Méthodes mathématiques pour ordinateur Vision, robotique et graphisme

Notes de cours pour CS 205A, automne 2013

Justin Salomon
Département d'informatique
Université de Stanford

Contenu

Première partie

Préliminaires

Chapitre 0

Révision de mathématiques

Dans ce chapitre, nous passerons en revue les notions pertinentes de l'algèbre linéaire et du calcul multivariable qui figureront dans notre discussion sur les techniques de calcul. Il se veut un examen des documents de base avec un parti pris pour les idées et les interprétations couramment rencontrées dans la pratique ; le chapitre peut être sauté en toute sécurité ou utilisé comme référence par les élèves ayant une solide formation en mathématiques.

0.1 Préliminaires : nombres et ensembles

Plutôt que d'envisager des discussions algébriques (et parfois philosophiques) du type « Qu'est-ce qu'un nombre ? », nous nous appuierons sur l'intuition et le bon sens mathématique pour définir quelques ensembles : • Les nombres naturels N = {1, 2, 3 , . . .}

• Les entiers Z = {. . . , -2, -1, 0, 1, 2, . . .}

• Les nombres rationnels Q = {a/b : a, b Z} • Les nombres [1]

réels R englobant Q ainsi que les nombres irrationnels comme π et $\sqrt{2}$ • Les nombres complexes C = {a + bi : a, b R }, où nous pensons que i satisfait i = $\sqrt{-1}$.

Il convient de reconnaître que notre définition de R est loin d'être rigoureuse. La construction des nombres réels peut être un sujet important pour les praticiens des techniques de cryptographie qui utilisent des systèmes de numération alternatifs, mais ces complexités ne sont pas pertinentes pour la discussion en cours.

Comme avec tous les autres ensembles, N, Z, Q, R et C peuvent être manipulés à l'aide d'opérations génériques pour générer de nouveaux ensembles de nombres. En particulier, rappelons que nous pouvons définir le "produit euclidien" de deux ensembles A et B comme

$$A \times B = \{(a, b) : a \in A \text{ et } b \in B\}.$$

On peut prendre des puissances d'ensembles en écrivant

$$A^n = \underbrace{A \times A \times \cdots \times A}_{n \text{ fois}}.$$

[1] C'est la première de nombreuses fois que nous utiliserons la notation {A : B} ; les accolades doivent désigner un ensemble et les deux-points peuvent être lus comme "tel que". Par exemple, la définition de Q peut être lue comme "l'ensemble des fractions a/b telles que a et b sont des entiers". Comme second exemple, on pourrait écrire N = {n Z : n > 0}.

Cette construction donne ce qui deviendra notre ensemble de nombres préféré dans les chapitres à venir :

$$R^n = \{(a1, a2, \ldots, an) : ai \quad R \text{ pour tout } i\}$$

0.2 Espaces vectoriels

Les cours d'introduction à l'algèbre linéaire pourraient facilement s'intituler "Introduction aux espaces vectoriels de dimension finie". Bien que la définition d'un espace vectoriel puisse sembler abstraite, nous trouverons de nombreuses applications concrètes qui satisfont toutes aux aspects formels et peuvent ainsi bénéficier de la machinerie que nous développerons.

0.2.1 Définition des espaces vectoriels

Nous commençons par définir un espace vectoriel et donnons un certain nombre d'exemples :

Définition 0.1 (Espace vectoriel). Un espace vectoriel est un ensemble V fermé par multiplication et addition scalaires.

Pour nos besoins, un scalaire est un nombre dans R, et les opérations d'addition et de multiplication satisfont aux axiomes usuels (commutativité, associativité, etc.). Il est généralement simple de repérer des espaces vectoriels dans la nature, notamment les exemples suivants :

Exemple 0.1 (Rn comme espace vectoriel). L'exemple le plus courant d'espace vectoriel est Rn . Ici, l'addition et la multiplication scalaire se produisent composant par composant :

$$(1, 2) + (-3, 4) = (1 - 3, 2 + 4) = (-2, 6)$$
$$10 \cdot (-1, 1) = (10 \cdot -1, 10 \cdot 1) = (-10, 10)$$

Exemple 0.2 (Polynômes). Un deuxième exemple important d'espace vectoriel est « l'anneau » de polynômes avec des entrées de nombres réels, noté R[x]. Un polynôme p R[x] est une fonction p : R → R prenant la forme[2]

$$p(x) = \sum_k \text{kakx}\_ .$$

L'addition et la multiplication scalaire sont effectuées de la manière habituelle, par exemple si $p(x) = x^2 + 2x - 1$ et $q(x) = x + 3x^3$, les exemples ci-dessus notent que des alors $2p(x) + 5q(x) = 5x^3 + 2^2 + 6x - 3$, qui est un autre polynôme. Soit dit en passant, car -5) fonctions comme $p(x) = (x - 1)(x + 1) + x$ bien qu'ils ne soient pas explicitement $(X^3$ sont encore des polynômes même écrits sous la forme ci-dessus.

Les éléments v V d'un espace vectoriel V sont appelés vecteurs, et une somme pondérée de la forme ∑i aivi , où ai R et vi V, est connue comme une combinaison linéaire des vi . Dans notre deuxième exemple, les "vecteurs" sont des fonctions, bien que nous n'utilisions normalement pas ce langage pour discuter de R[x]. Un k avec la séquence points de vue serait d'identifier le polynôme ∑k akx (a0, a1, a2, · · ·); rappelez-vous que les façon de relier ces deux polynômes ont un nombre fini de termes, donc la séquence se terminera éventuellement par une chaîne de zéros.

2La notation f : A → B signifie que f est une fonction qui prend en entrée un élément de l'ensemble A et sort un élément de l'ensemble B. Par exemple, f : R → Z prend en entrée un nombre réel dans R et sort un entier Z , comme cela pourrait être le cas pour f(x) = x, la fonction « arrondir vers le bas ».

0.2.2 Portée, indépendance linéaire et bases

Supposons que nous partions des vecteurs v1, . . . ,vk V pour l'espace vectoriel V. D'après la définition 0.1, nous avons deux manières de partir de ces vecteurs et de construire de nouveaux éléments de V : l'addition et la multiplication scalaire. L'idée de span est qu'il décrit tous les vecteurs que vous pouvez atteindre via ces deux opérations :

Définition 0.2 (Span). L'étendue d'un ensemble S V de vecteurs est l'ensemble

$$\text{span } S \equiv \{a_1v_1 + \cdots + a_kv_k : k \geq 0, v_i \in V \text{ pour tout } i, \text{ et } a_i \in \mathbb{R} \text{ pour tout } i\}.$$

Notez que l'étendue S est un sous-espace de V, c'est-à-dire un sous-ensemble de V qui est en lui-même un espace vectoriel. Nous pouvons donner quelques exemples :

Exemple 0.3 (Mixologie). Le « puits » typique d'un bar à cocktails contient au moins quatre ingrédients à la disposition du barman : vodka, tequila, jus d'orange et grenadine. En supposant que nous ayons ce puits simple, nous pouvons représenter les boissons sous forme de points dans R4 avec un emplacement pour chaque ingrédient. Par exemple, un "lever de soleil de tequila" typique peut être représenté à l'aide du point (0, 1,5, 6, 0,75), représentant les quantités de vodka, de tequila, de jus d'orange et de grenadine (en onces), resp.

L'ensemble des boissons pouvant être préparées avec le puits typique est contenu dans

$$\text{étendue } \{(1, 0, 0, 0),(0, 1, 0, 0),(0, 0, 1, 0),(0, 0, 0, 1)\},$$

c'est-à-dire toutes les combinaisons des quatre ingrédients de base. Cependant, un barman cherchant à gagner du temps pourrait remarquer que de nombreuses boissons ont le même rapport jus d'orange/grenadine et mélanger les bouteilles. Le nouveau puits simplifié peut être plus facile à verser mais peut faire fondamentalement moins de boissons :

$$\text{étendue } \{(1, 0, 0, 0),(0, 1, 0, 0),(0, 0, 6, 0{,}75)\}$$

Exemple 0.4 (Polynômes). Définir le $p_k(x) \equiv x^k$. Ensuite, il est facile de voir que

$$\mathbb{R}[x] = \text{étendue } \{p_k : k \geq 0\}.$$

Assurez-vous de bien comprendre la notation pour voir pourquoi c'est le cas.

L'ajout d'un autre élément à un ensemble de vecteurs n'augmente pas toujours la taille de sa plage. Pour exemple, dans R2 , il est clair que

$$\text{durée } \{(1, 0),(0, 1)\} = \text{durée } \{(1, 0),(0, 1),(1, 1)\}.$$

Dans ce cas, on dit que l'ensemble {(1, 0),(0, 1),(1, 1)} est linéairement dépendant :

Définition 0.3 (Dépendance linéaire). Nous proposons trois définitions équivalentes. Un ensemble S V de vecteurs est linéairement dépendant si :

1. Un des éléments de S peut être écrit comme une combinaison linéaire des autres éléments, ou S contient zéro.
2. Il existe une combinaison linéaire non vide d'éléments vk S donnant ∑ ck = 0 pour tout k. $\sum_{k=1}^{m} c_kv_k = 0$ où
3. Il existe v S tel que span S = span S\{v}. Autrement dit, nous pouvons supprimer un vecteur de S sans affectant sa portée.

Si S n'est pas linéairement dépendant, on dit qu'il est linéairement indépendant.

Fournir une preuve ou une preuve informelle que chaque définition est équivalente à ses homologues (de manière « si et seulement si ») est un exercice valable pour les élèves moins à l'aise avec la notation et les mathématiques abstraites.

Le concept de dépendance linéaire conduit à une idée de « redondance » dans un ensemble de vecteurs. En ce sens, il est naturel de se demander quelle taille d'ensemble nous pouvons choisir avant d'ajouter un autre vecteur ne peut éventuellement augmenter l'étendue. En particulier, supposons que nous ayons un ensemble linéairement indépendant S V, et que nous choisissions maintenant un vecteur supplémentaire v V. L'ajout de v à S conduit à l'un des deux résultats possibles :

1. L'envergure de S {v} est plus grande que l'envergure de S.
2. L'ajout de v à S n'a aucun effet sur la durée.

La dimension de V n'est rien de plus que le nombre maximal de fois où nous pouvons obtenir le résultat 1, ajouter v à S et répéter.

Définition 0.4 (Dimension et base). La dimension de V est la taille maximale |S| d'un ensemble S V linéairement indépendant tel que span S = V. Tout ensemble S satisfaisant cette propriété est appelé une base de V.

Exemple 0.5 (Rn). La base standard de Rn est l'ensemble des vecteurs de la forme

$$ek \equiv (0, \ldots, \underbrace{0}_{k-1 \text{ emplacements}}, 1, \underbrace{0}_{n-k \text{ emplacements}}, \ldots, 0).$$

C'est-à-dire que ek a tous les zéros à l'exception d'un seul dans le k-ième emplacement. Il est clair que ces vecteurs sont linéairement indépendants et forment une base ; par exemple dans R3 tout vecteur (a, b, c) peut s'écrire ae1 + be2 + ce3. Ainsi, la dimension de Rn est n, comme on pouvait s'y attendre.

Exemple 0.6 (Polynômes). Il est clair que l'ensemble {1, x, x , . . .} est un ensemble linéairement indépendant de polynômes couvrant R[x]. Remarquez que cet ensemble est infiniment grand, et donc la dimension de R[x] est ∞.

0.2.3 Notre objectif : Rn

L'espace vectoriel Rn est particulièrement important pour notre propos . Ce n'est rien , le soi-disant Eu à n dimensions de plus que l'ensemble des axes de coordonnées rencontrés dans les cours de mathématiques du secondaire :

- R1 ≡ R est la droite numérique
- R2 est le plan bidimensionnel de coordonnées (x, y) • R3 représente l'espace tridimensionnel de coordonnées (x, y, z)

Presque toutes les méthodes de ce cours traiteront des transformations et des fonctions sur Rn .

Pour plus de commodité, nous écrivons généralement les vecteurs dans Rn sous forme de "colonne", comme suit

$$(a1, \ldots, une) \equiv \begin{matrix} a1 \\ a2 \\ \vdots \\ un \end{matrix}$$

Cette notation inclura des vecteurs en tant que cas particuliers de matrices discutés ci-dessous.

Contrairement à certains espaces vectoriels, Rn possède non seulement une structure d'espace vectoriel, mais aussi une construction supplémentaire qui fait toute la différence : le produit scalaire.

Définition 0.5 (Produit scalaire). Le produit scalaire de deux vecteurs a = (a1, . . . , an) et b = (b1, . . . , bn) dans Rn est donné par

$$un \cdot b = \sum_{k=1}^{n} akbk .$$

Exemple 0.7 (R2). Le produit scalaire de (1, 2) et (−2, 6) est 1 · −2 + 2 · 6 = −2 + 12 = 10.

Le produit scalaire est un exemple de métrique, et son existence donne une notion de géométrie à Rn . Par exemple, nous pouvons utiliser le théorème de Pythagore pour définir la norme ou la longueur d'un vecteur comme la racine carrée

$$a2 \equiv \sqrt{une 2 1 + \cdots + une n} = \sqrt{un \cdot un}.$$

Alors, la distance entre deux pointsa,b Rn est simplement b −a2.

Les produits scalaires donnent non seulement des notions de longueurs et de distances mais aussi d'angles. Rappelons l'identité suivante de la trigonométrie fora,b R3 :

$$a \cdot b = ab \cos \theta$$

où θ est l'angle entre a et b. Pour n ≥ 4, cependant, la notion « d'angle » est beaucoup plus difficile à visualiser pour Rn . Nous pourrions définir l'angle θ entre a et b comme étant la valeur θ donnée par

$$\theta \equiv \arccos \frac{un \cdot b}{un B} .$$

Nous devons faire nos devoirs avant de faire une telle définition! En particulier, rappelons que cosinus out met des valeurs dans l'intervalle [−1, 1], nous devons donc vérifier que l'entrée de l'arc cosinus (également noté cos−1) est dans cet intervalle ; heureusement, l'inégalité bien connue de Cauchy-Schwarz a ·b ≤ ab garantit exactement cette propriété.

Lorsque a = cb pour un certain c R, on a θ = arccos 1 = 0, comme on pouvait s'y attendre : l'angle entre les vecteurs parallèles est nul. Qu'est-ce que cela signifie pour les vecteurs d'être perpendiculaires? Remplaçons θ = 90◦ . Ensuite nous avons

$$0 = \cos 90^\circ$$

$$= \frac{un \cdot b}{un B}$$

Multiplier les deux côtés par ab motive la définition :

Définition 0.6 (Orthogonalité). Deux vecteurs sont perpendiculaires, ou orthogonaux, quanda ·b = 0.

Cette définition est quelque peu surprenante d'un point de vue géométrique. En particulier, nous avons réussi à définir ce que signifie être perpendiculaire sans aucune utilisation explicite des angles. Cette construction facilitera la résolution de certains problèmes pour lesquels la non-linéarité du sinus et du cosinus aurait pu créer des maux de tête dans des contextes plus simples.

Mis à part 0,1. Il y a de nombreuses questions théoriques à méditer ici, dont certaines que nous aborderons dans les prochains chapitres lorsqu'elles seront plus motivées :

- Tous les espaces vectoriels admettent-ils des produits scalaires ou des structures similaires ?
- Tous les espaces vectoriels de dimension finie admettent-ils des produits scalaires ?
- Quel pourrait être un produit scalaire raisonnable entre éléments de R[x] ?

Les étudiants intrigués peuvent consulter des textes sur l'analyse réelle et fonctionnelle.

0.3 Linéarité

Une fonction entre des espaces vectoriels qui préserve la structure est connue sous le nom de fonction linéaire :

Définition 0.7 (Linéarité). Supposons que V et V soient des espaces vectoriels. Alors, L : V → V est linéaire s'il vérifie les deux critères suivants pour tout v,v1,v2 V et c R :

- L préserve les sommes : L[v1 +v2] = L[v1] + L[v2]
- L préserve les produits scalaires : L[cv] = cL[v]

Il est facile de générer des cartes linéaires entre espaces vectoriels, comme nous pouvons le voir dans les exemples suivants :

Exemple 0.8 (Linéarité dans Rn). L'application suivante f : R2 → R3 est linéaire :

$$f(x, y) = (3x, 2x + y, -y)$$

On peut vérifier la linéarité comme suit :

- Conservation de la somme :

$$\begin{aligned} f(x1 + x2, y1 + y2) &= (3(x1 + x2), 2(x1 + x2) + (y1 + y2), -(y1 + y2)) \\ &= (3x1, 2x1 + y1, -y1) + (3x2, 2x2 + y2, \\ &\quad -y2) = f(x1, y1) + f(x2, y2) \end{aligned}$$

- Conservation du produit scalaire :

$$f(cx, cy) = (3cx, 2cx + cy, -cy) = c(3x, 2x + y, -y) = c\, f(x, y)$$

En revanche, g(x, y) ≡ xy2 n'est pas linéaire. Par exemple, g(1, 1) = 1 mais g(2, 2) = 8 = 2 · g(1, 1), donc cette forme ne préserve pas les produits scalaires.

Exemple 0.9 (Intégration). Le L « fonctionnel » suivant de R[x] à R est linéaire :

$$L[p(x)] \equiv \int_0^1 p(x)\, dx.$$

Cet exemple un peu plus abstrait associe des polynômes p(x) à des nombres réels L[p(x)]. Par exemple, nous pouvons écrire

$$L[3x^2 + x - 1] = \int_0^1 (3x^2 1 + x - 1)\, dx = 2 -\cdot$$

La linéarité provient des faits bien connus suivants du calcul :

$$\int_0^1 c \cdot f(x)\, dx = c \int_0^1 f(x)\, dx$$

$$\int_0^1 [f(x) + g(x)]\, dx = \int_0^1 f(x)\, dx + \int_0^1 g(x)\, dx$$

Nous pouvons écrire une forme particulièrement agréable pour les applications linéaires sur Rn . Rappelons que le vecteur a = (a1, . . . , an) est égal à la somme ∑k akek , où ek est le k-ième vecteur de base standard. Alors, si L est linéaire on sait :

$$\begin{aligned} L[a] &= L \sum_k a_k e_k \text{ pour la base standard} e_k \\ &= \sum_k L[a_k e_k] \text{ par préservation de la somme} \\ &= \sum_k a_k L[e_k] \text{ par préservation du produit scalaire} \end{aligned}$$

Cette dérivation montre le fait important suivant :

L est entièrement déterminé par son action sur la base standard vectorsek .

Autrement dit, pour tout vectora , nous pouvons utiliser la somme ci-dessus pour déterminer L[a] en combinant linéairement Rn L[e1], . . . ,L[fr].

Exemple 0.10 (Développer une carte linéaire). Rappelons l'application de l'exemple 0.8 donnée par f(x, y) = (3x, 2x + y, −y). On a f(e1) = f(1, 0) = (3, 2, 0) et f(e2) = f(0, 1) = (0, 1, −1). Ainsi, la formule ci-dessus montre :

$$f(x, y) = x\, f(e_1) + y\, f(e_2) = x \begin{matrix} 3 \\ 2 \\ 0 \end{matrix} + y \begin{matrix} 0 \\ 1 \\ -1 \end{matrix}$$

0.3.1 Matrices

L'expansion des cartes linéaires ci-dessus suggère l'un des nombreux contextes dans lesquels il est utile de stocker plusieurs vecteurs dans la même structure. Plus généralement, disons que nous avons n vecteurs v1, . . . ,vn Rm. Nous pouvons écrire chacun sous la forme d'un vecteur colonne :

$$v_1 = \begin{matrix} v_{11} \\ v_{21} \\ \vdots \\ v_{m1} \end{matrix} \quad ,v_2 = \begin{matrix} v_{12} \\ v_{22} \\ \vdots \\ v_{m2} \end{matrix} \quad , \cdots ,v_n = \begin{matrix} v_{1n} \\ v_{2n} \\ \vdots \\ v_{mn} \end{matrix}$$

Les transporter séparément peut être fastidieux sur le plan de la notation, donc pour simplifier les choses, nous les combinons simplement en une seule matrice m × n :

$$\begin{vmatrix} & & \\ v1 & v2 & \cdots & vn \\ & & \end{vmatrix} = \begin{matrix} v11 & v12 & \cdots & v1n \\ v21 & v22 & \cdots & v2n \\ \vdots & \vdots & \vdots & \vdots \\ vm1 & vm2 & \cdots & vmn \end{matrix}$$

Nous appellerons l'espace de telles matrices $\mathbb{R}^{m\times n}$.

Exemple 0.11 (Matrice d'identité). Nous pouvons stocker la base standard pour $\mathbb{R}^n$ dans la « matrice d'identité » n × n $I_{n\times n}$ donné par :

$$\text{Dans}\times n \equiv \begin{vmatrix} & & \\ e1 & e2 & \cdots & fr \\ & & \end{vmatrix} = \begin{matrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{matrix}$$

Puisque nous avons construit des matrices comme des moyens pratiques de stocker des ensembles de vecteurs, nous pouvons utiliser la multiplication pour exprimer comment ils peuvent être combinés linéairement. En particulier, une matrice dans $\mathbb{R}^{m\times n}$ peut être multipliée par un vecteur colonne dans $\mathbb{R}^n$ comme suit :

$$\begin{vmatrix} & & \\ v1 & v2 & \cdots & vn \\ & & \end{vmatrix} \begin{matrix} c1 \\ c2 \\ \vdots \\ CN \end{matrix} \equiv c1v1 + c2v2 + \cdots + cnvn$$

L'expansion de cette somme donne la formule explicite suivante pour les produits matrice-vecteur :

$$\begin{matrix} v11 & v12 & \cdots & v1n \\ v21 & v22 & \cdots & v2n \\ \vdots & \vdots & \vdots & \vdots \\ vm1 & vm2 & \cdots & vmn \end{matrix} \begin{matrix} c1 \\ c2 \\ \vdots \\ CN \end{matrix} = \begin{matrix} c1v11 + c2v12 + \cdots + cnv1n \\ c1v21 + c2v22 + \cdots + cnv2n \\ \vdots \\ c1vm1 + c2vm2 + \cdots + cnvmn \end{matrix}$$

Exemple 0.12 (Multiplication de matrice identité). Il est clair que pour tout $x \in \mathbb{R}^n$, $x = I_{n\times n}x$, où $I_{n\times n}$ est la matrice identité de l'exemple 0.11. nous pouvons écrire

Exemple 0.13 (Carte linéaire). Nous revenons une fois de plus à l'expression de l'exemple 0.8 pour montrer une autre forme alternative :

$$f(x, y) = \begin{matrix} 3 & 0 & 2 \\ 1 & & \\ 0 & -1 & \end{matrix} \begin{matrix} x \\ \\ y \end{matrix}$$

Nous définissons de la même manière un produit entre une matrice dans $M \in \mathbb{R}^{m\times n}$ et une autre matrice dans $\mathbb{R}^{n\times p}$ en concaténant des produits matrice-vecteur individuels :

$$M \begin{vmatrix} & & \\ c1 & c2 & \cdots & cn \\ & & \end{vmatrix} \equiv \begin{vmatrix} & & \\ Mc1 & Mc2 & \cdots & Mcn \\ & & \end{vmatrix}$$

Exemple 0.14 (Mixologie). Poursuivant l'exemple 0.3, supposons que nous fassions une tequila sunrise et une seconde con coction avec des parties égales des deux liqueurs dans notre puits simplifié. Pour savoir combien d'ingrédients de base sont contenus dans chaque commande, nous pourrions combiner les recettes pour chaque colonne et utiliser la matrice multiplication:

	Puits 1	Puits 2	Puits 3
Vodka	1	0	0
Tequila	0	1	0
JO	0	0	6
Grenade	0	0	0,75

.

	Boire 1	Boire 2
	0	0,75
	1.5	0,75
	1	2

=

	Boire 1	Boire 2	
	0	0,75	Vodka
	1,5	0,75	Tequila
	6	12	JO
	0,75	1.5	Grenadine

En général, on utilisera des majuscules pour représenter les matrices, comme $A \in \mathbb{R}^{m\times n}$. Nous utiliserons le notation $A_{ij} \in \mathbb{R}$ pour désigner l'élément de A à la ligne i et à la colonne j.

0.3.2 Scalaires, vecteurs et matrices

Il n'est pas surprenant que nous puissions écrire un scalaire sous la forme d'un vecteur 1×1 c $\mathbb{R}^{1\times 1}$. Similaire, comme nous l'avons déjà suggéré au §0.2.3, si on écrit les vecteurs dans $\mathbb{R}^n$ sous forme de colonnes, on peut les considérer comme des matrices $n \times 1$ $v \in \mathbb{R}^{n\times 1}$. Notez que les produits matrice-vecteur peuvent être interprétés facilement dans ce contexte ; Par exemple, si $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, etb $\mathbb{R}^m$, alors on peut écrire des expressions comme

$$\underset{m\times n}{A}\ \underset{n\times 1}{x} = \underset{m\times 1}{b}$$

Nous allons introduire un opérateur supplémentaire sur les matrices qui est utile dans ce contexte :

Définition 0.8 (Transposer). La transposée d'une matrice $A \in \mathbb{R}^{m\times n}$ est une matrice $A \in \mathbb{R}^{n\times m}$ d'éléments $(A^\top)_{ij} = A_{ji}$.

Exemple 0.15 (Transposition). La transposée de la matrice

$$A = \begin{matrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{matrix}$$

est donné par

$$A^\top = \begin{matrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{matrix}.$$

Géométriquement, nous pouvons considérer la transposition comme le retournement d'une matrice sur sa diagonale.

Ce traitement unifié des scalaires, des vecteurs et des matrices combiné à des opérations telles que la trans position et la multiplication peut conduire à des dérivations astucieuses d'identités bien connues. Par exemple,

on peut calculer les produits scalaires des vecteursa,b Rn en faisant la série d'étapes suivante :

$$\text{un} \cdot b = \sum_{k=1}^{n} \text{akbk}$$

$$= a1\ a2 \cdots \text{une} \begin{matrix} b1 \\ b2 \\ \vdots \\ \text{milliards} \end{matrix}$$

$$= ab$$

De nombreuses identités importantes de l'algèbre linéaire peuvent être dérivées en enchaînant ces opérations avec quelques règles :

$$(\text{UNE}\ ) = \text{UNE}$$

$$(A + B) = A + B$$

$$(AB) = BA$$

Exemple 0.16 (Norme résiduelle). Supposons que nous ayons une matrice A et deux vecteurs x et b. Si nous souhaitons savoir dans quelle mesure Ax se rapproche de b, nous pourrions définir un résidu r ≡ b − Ax ; ce résidu est nul exactement lorsque Ax = b. Sinon, nous pourrions utiliser la norme r2 comme approximation de la relation entre Ax et b. Nous pouvons utiliser les identités ci-dessus pour simplifier :

$$r \begin{matrix} 2 \\ 2 \end{matrix} = b - \text{Hache} \begin{matrix} 2 \\ 2 \end{matrix}$$

$$= (b - Ax) \cdot (b - Ax) \text{ comme expliqué au §0.2.3} = (b$$

$$- Ax)\ (b - Ax) \text{ par notre expression pour le produit scalaire ci-dessus} = (b$$

$$-x\ A\ )(b - Ax) \text{ par propriétés de transposition} = bb - b\ Ax - x\ A\ b$$

$$+x\ A\ Ax \text{ après multiplication}$$

Les quatre termes du côté droit sont des scalaires ou, de manière équivalente, des matrices 1 × 1. Les scalaires considérés comme des matrices bénéficient trivialement d'une belle propriété supplémentaire c = c, puisqu'il n'y a rien à transposer ! Ainsi, nous pouvons écrire

$$x\ A\ b = (x\ A\ b) = b\ Ax$$

Cela nous permet de simplifier encore plus notre expression :

$$r \begin{matrix} 2 \\ 2 \end{matrix} = bb - 2b\ Axe + x\ A\ Axe$$

$$= \text{Hache} \begin{matrix} 2 \\ 2 \end{matrix} - 2b\ Ax + b \begin{matrix} 2 \\ 2 \end{matrix}$$

Nous aurions pu dériver cette expression en utilisant des identités de produits scalaires, mais les étapes intermédiaires ci-dessus s'avéreront utiles dans notre discussion ultérieure.

0.3.3 Problème modèle : Ax =b

Dans le cours d'introduction à l'algèbre, les élèves passent un temps considérable à résoudre des systèmes linéaires tels que les suivants pour des triplets (x, y, z) :

$$\begin{aligned} 3x + 2y + 5z &= 0 \\ -4x + 9y - 3z &= -7 \\ 2x - 3y - 3z &= 1 \end{aligned}$$

Nos constructions au §0.3.1 nous permettent d'encoder de tels systèmes de manière plus propre :

$$\begin{bmatrix} 3 & 2 & 5 \\ -4 & 9 & -3 \\ 2 & -3 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ -7 \\ 1 \end{bmatrix}$$

Plus généralement, on peut écrire des systèmes linéaires d'équations sous la forme Ax = b en suivant le même schéma ci-dessus ; ici, le vecteur x est inconnu alors que A et b sont connus. Un tel système d'équations n'est pas toujours garanti d'avoir une solution. Par exemple, si A ne contient que des zéros, il est clair qu'aucun x ne satisfera Ax =b chaque fois que b =0. Nous reporterons une considération générale sur le moment où une solution existe à notre discussion sur les solveurs linéaires dans les prochains chapitres.

Une interprétation clé du système Ax =b est qu'il adresse la tâche :

Ecrireb comme une combinaison linéaire des colonnes de A.

Pourquoi? Rappelons du §0.3.1 que le produit Ax code une combinaison linéaire des colonnes de A avec des poids contenus dans les éléments de x. Ainsi, l'équation Ax =b demande que la combinaison linéaire Ax soit égale au vecteurb donné. Compte tenu de cette interprétation, nous définissons l'espace des colonnes de A comme étant l'espace des côtés droitsb pour lequel le système a une solution :

Définition 0.9 (Espace colonne). L'espace des colonnes d'une matrice A $\in$ $R^{m\times n}$ est l'étendue des colonnes de A. On peut écrire comme

$$\text{col } A \equiv \{Ax : x \in R^n\}.$$

Un cas important est un peu plus facile à considérer. Supposons que A soit carré, on peut donc écrire A $\in$ $R^{n\times n}$. De plus, supposons que le système Ax = b ait une solution pour tous les choix de b. Le so par notre seule condition onb est qu'il soit membre de R^n conclure, interprétation ci-dessus de Ax =b nous pouvons que les colonnes de A s'étendent sur R^n .

Dans ce cas, puisque le système linéaire est toujours résoluble supposons que nous insérons la base standard e1, . . . ,en pour produire les vecteurs x1, . . . ,xn satisfaisant Axk = ek pour chaque k. Ensuite, nous pouvons "empiler" ces expressions pour montrer :

$$A \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ax_1 & Ax_2 & \cdots & Ax_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_n \\ | & | & & | \end{bmatrix} = I_{n\times n},$$

où $I_{n\times n}$ est la matrice identité de l'exemple 0.11. On appellera la matrice à colonnes xk l'inverse A^{-1} , qui satisfait

$$AA^{-1} = A^{-1}A = I_{n\times n}.$$

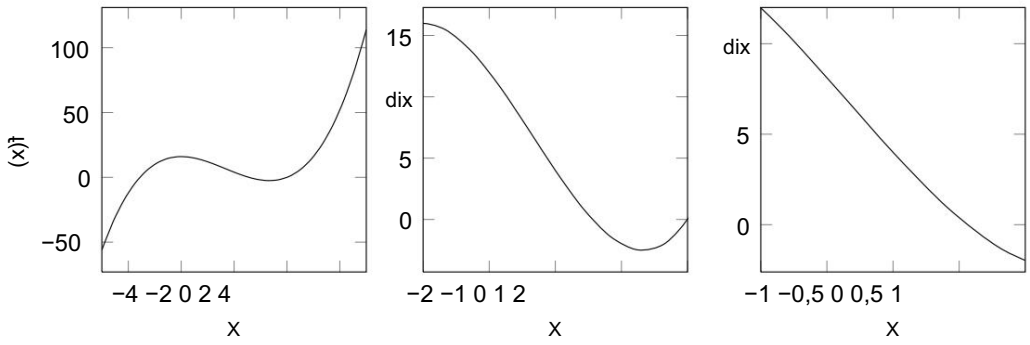


Figure 1 : Plus nous zoomons sur f(x) = x 3 2 + x − 8x + 4, plus il ressemble à une ligne.

Il est aussi facile de vérifier que (A = A. Lorsqu'un tel inverse existe, il est facile de résoudre le système Ax =b. En particulier, on trouve :

$$x = \ln \times nx = (A - 1A)x = A^{-1}(Ax) = A - 1b$$

0.4 Non-linéarité : calcul différentiel

Alors que la beauté et l'applicabilité de l'algèbre linéaire en font une cible d'étude clé, les non-linéarités abondent dans la nature et nous devons souvent concevoir des systèmes informatiques capables de gérer ce fait de la vie. Après tout, au niveau le plus élémentaire, le carré de la célèbre relation E = mc2 la rend moins susceptible d'être analysée linéairement.

0.4.1 Différenciation

Alors que de nombreuses fonctions sont globalement non linéaires, elles présentent localement un comportement linéaire. Cette idée de « linéarité locale » est l'une des principales motivations du calcul différentiel. Par exemple, la figure 1 montre que si vous zoomez suffisamment près d'une fonction lisse, cela finit par ressembler à une ligne. La dérivée f (x) d'une fonction f(x) : R → R n'est rien de plus que la pente de la droite d'approximation, calculée en trouvant la pente des droites passant par des points de plus en plus proches de x :

$$= \lim_{y \to x} \frac{f(y) - f(x)\ f(x)}{y - x}$$

On peut exprimer la linéarité locale en écrivant f(x + ∆x) = f(x) + ∆x · f (x) + O(∆x 2).

Si la fonction f prend plusieurs entrées, alors elle peut s'écrire f(x) : Rn → R pour x Rn ; en d'autres termes, à chaque point x = (x1, . . . , xn) dans l'espace à n dimensions f attribue un seul nombre f(x1, . . . , xn). Notre idée de linéarité locale s'effondre quelque peu ici, car les lignes sont des objets unidimensionnels. Cependant, la fixation de toutes les variables sauf une revient au cas du calcul à une seule variable. Par exemple, on pourrait écrire g(t) = f(t, x2, . . . , xn), où l'on fixe simplement des constantes x2, . . . , xn. Alors, g(t) est une fonction différentiable d'une seule variable. Bien sûr, nous aurions pu mettre t dans n'importe lequel des emplacements d'entrée pour f , donc en général nous faisons la définition suivante de la dérivée partielle de f :

f Définition 0.10 (Dérivée partielle). La k-ième dérivée partielle de f , notée ∂x_k , reliant f à $\frac{\partial}{\partial x_k}$ est donné par différent ième variable d'entrée :

$$\frac{\partial f}{\partial x_k}(x_1, \ldots, x_n) \equiv \frac{d}{dt} f(x_1, \ldots, x_{k-1}, t, x_{k+1}, \ldots, x_n)|_{t=x_k}$$

La notation « $|_{t=x_k}$ » doit être lue comme « évaluée à $t = x_k$ ».

Exemple 0.17 (Relativité). La relation $E = mc^2$ peut être considérée comme une fonction de m et c vers E. Ainsi, on pourrait écrire $E(m, c) = mc^2$, donnant les dérivées

$$\frac{\partial E}{\partial m} = c^2$$
$$\frac{\partial E}{\partial c} = 2mc$$

En utilisant le calcul à une variable, on peut écrire :

$$f(x + \Delta x) = f(x_1 + \Delta x_1, x_2 + \Delta x_2, \ldots, x_n + \Delta x_n)$$
$$= f(x_1, x_2 + \Delta x_2, \ldots, x_n + \Delta x_n) + \frac{\partial}{\partial x_1}\Delta x_1 + O(\Delta x_1^2) \text{ par calcul à une variable}$$
$$= f(x_1, \ldots, x_n) + \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}\Delta x_k + O(\Delta x_k^2) \text{ en répétant ceci n fois}$$
$$= f(x) + \nabla f(x) \cdot \Delta x + O(\Delta x^2)$$

où nous définissons le gradient de f comme

$$\nabla f \equiv \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right) \in \mathbb{R}^n$$

De cette relation, il est facile de voir que f peut être différenciée dans n'importe quelle direction v ; on peut évaluer cette dérivée $D_v f$ comme suit :

$$D_v f(x) \equiv \frac{d}{dt} f(x + tv)|_{t=0} = \nabla f(x) \cdot v$$

Exemple 0.18 ($\mathbb{R}^2$). Soit $f(x, y) = x^2y^3$. Alors,

$$\frac{\partial}{\partial x} F = 2xy^3$$
$$\frac{\partial}{\partial y} F = 3x^2y^2$$

Ainsi, on peut écrire $\nabla f(x, y) = (2xy^3 , 3x^2y^2)$. La dérivée de f en (1, 2) dans la direction (−1, 4) est donnée par $(-1, 4) \cdot \nabla f(1, 2) = (-1, 4) \cdot (16, 12) = 32$.

Exemple 0.19 (Fonctions linéaires). Il est évident mais intéressant de noter que le gradient de $f(x) \equiv a \cdot x + c = (a_1x_1 + c_1, \ldots, a_nx_n + c_n)$ est un.

Exemple 0.20 (Formes quadratiques). Prenez n'importe quelle matrice A , et définissons f(x) ≡ x Ax. Expansion
Rn×n cette fonction montre élément par élément

$$f(x) = \sum_{ij} Aijxixj ;$$

développer f et vérifier explicitement cette relation vaut la peine. Prenons k {1, . . . , n}. Ensuite, nous pouvons séparer tous les termes contenant xk :

$$f(x) = Akkx_k^2 + xk \sum_{je=k} Aikxi + \sum_{j=k} Akjxj + \sum_{i,j=k} Aijxixj$$

Avec cette factorisation, il est facile de voir

$$\frac{\partial f}{\partial xk} = 2Akkxk + \sum_{je=k} Aikxi + \sum_{j=k} Akjxj$$
$$= \sum_{je=1}^{n} (Aik + Aki)xi$$

Cette somme n'est rien de plus que la définition de la multiplication matrice-vecteur ! Ainsi, nous pouvons écrire

$$f(x) = Ax + A x.$$

Nous avons généralisé de f : R → R à f : Rn → R. Pour atteindre une généralité complète, nous voudrions considérer f : Rn → Rm. En d'autres termes, f prend n nombres et produit m nombres. Heureusement, cette extension est simple, car nous pouvons considérer f comme une collection de fonctions à valeur unique f1, . . . , fm : Rn → R fusionnés en un seul vecteur. C'est-à-dire que nous écrivons :

$$f(x) = \begin{matrix} f1(x) \\ f2(x) \\ \vdots \\ fm(x) \end{matrix}$$

Chaque fk peut être différentié comme précédemment, donc au final on obtient une matrice de dérivées partielles appelée le jacobien de f :

Définition 0.11 (jacobien). Le jacobien de f : Rn → Rm est la matrice D f Rm×n d'entrées

$$(D f)ij \equiv \frac{\partial fi}{\partial xj}.$$

Exemple 0.21 (Fonction simple). Supposons f(x, y) = (3x, −xy2 , x + y). Alors,

$$ré f(x, y) = \begin{matrix} 3 & \\ -y^2 & 0 \\ 1 & -2xy \end{matrix} 1$$

Assurez-vous que vous pouvez dériver ce calcul à la main.

Exemple 0.22 (Multiplication matricielle). Sans surprise, le jacobien de f(x) = Ax pour la matrice A est donné par D f(x) = A.

Nous rencontrons ici un point commun de confusion. Supposons qu'une fonction a une entrée vectorielle et une sortie scalaire, c'est-à-dire f : Rn → R. Nous avons défini le gradient de f comme un vecteur colonne, donc pour aligner cette définition avec celle du jacobien, nous devons écrire

$$\text{ré } f = \quad f .$$

0.4.2 Optimisation

Rappel du calcul à une seule variable minima et maxima de f : R → R doit se produire aux points x satisfaisant f (x) = 0. Bien sûr, cette condition est nécessaire plutôt que suffisante : il peut exister des points x avec f (x) = 0 qui ne sont ni des maxima ni des minima. Cela dit, trouver de tels points critiques de f peut être une étape d'un algorithme de minimisation de fonction, tant que l'étape suivante garantit que le x résultant est en fait un minimum/maximum.

Si f : Rn → R est minimisé ou maximisé en x, il faut s'assurer qu'il n'existe pas une seule direction ∆x à partir de x dans laquelle f diminue ou augmente, resp. D'après la discussion du §0.4.1, cela signifie qu'il faut trouver des points pour lesquels f = 0.

Exemple 0.23 (Fonction simple). Supposons que f(x, y) = x 2x + 8y. ^{2} + 2xy + 4y ^{2} $\partial x\ \partial y$ $\frac{\partial F}{}$ ∂ F . Alors, = 2x + 2y et $\frac{}{}$ = Ainsi, les points critiques de f satisfont :

$$2x + 2y = 0 \; 2x$$
$$+ 8y = 0$$

Clairement ce système est résolu en (x, y) = (0, 0). En effet, c'est le minimum de f , comme on peut le voir plus clairement en écrivant f(x, y) = (x + y) ^{2} + 3 ans^2.

Exemple 0.24 (Fonctions quadratiques). Supposons que f(x) = x Ax + bx + c. Ensuite, à partir des exemples de la section précédente, nous pouvons écrire f(x) = (A + A)x +b. Ainsi, les points critiques x de f vérifient (A + A)x +b = 0.

Contrairement au calcul à une seule variable, lorsque nous effectuons un calcul sur Rn , nous pouvons ajouter des contraintes à notre optimisation. La forme la plus générale d'un tel problème ressemble à :

$$\text{minimiser } f(x) \text{ tel}$$
$$\text{que } g(x) = 0$$

Exemple 0.25 (Zones rectangulaires). Supposons qu'un rectangle ait une largeur w et une hauteur h. Un problème de géométrie classique consiste à maximiser l'aire avec un périmètre fixe 1 :

$$\text{maximiser } wh$$
$$\text{tel que } 2w + 2h - 1 = 0$$

Lorsque nous ajoutons cette contrainte, nous ne pouvons plus nous attendre à ce que les points critiques satisfassent f(x) = 0, puisque ces points pourraient ne pas satisfaire g(x) = 0.

Pour l'instant, supposons g : Rn → R. Considérons l'ensemble des points S0 ≡ {x : g(x) = 0}. Évidemment, deux x,y S0 satisfont la relation g(y) − g(x) = 0 − 0 = 0. Supposons y = x + ∆x pour petit ∆x.
Alors, g(y) − g(x) = g(x) · ∆x + O(∆x). En d'autres termes, si nous commençons à x satisfaisant g(x) = 0, alors si nous déplaçons dans la direction ∆x g(x) · ∆x ≈ 0 pour continuer à satisfaire cette relation.

Rappelons maintenant que la dérivée de f dans la direction v en x est donnée par f ·v. Si x est un minimum du problème d'optimisation contrainte ci-dessus, alors tout petit déplacement x vers x + v devrait entraîner une augmentation de f(x) à f(x + v). Puisque nous ne nous soucions que des déplacements v préservant la contrainte g(x + v) = c, d'après notre argument ci-dessus, nous voulons f · v = 0 pour tout v satisfaisant g(x)·v = 0. En d'autres termes, f et g doivent être parallèles, une condition que nous pouvons écrire comme f = λ g pour un certain λ R.

Définir

$$\Lambda(x, \lambda) = f(x) - \lambda g(x).$$

Alors, les points critiques de Λ sans contraintes vérifient :

$$0 = \frac{\partial \Lambda}{\partial \lambda} = -g(x)$$
$$0 = \quad x\Lambda = \quad f(x) - \lambda \quad g(x)$$

En d'autres termes, les points critiques de Λ satisfont g(x) = 0 et f(x) = λ g(x), exactement les conditions d'optimalité que nous avons dérivées !

L'extension aux contraintes multivariées donne ce qui suit :

Théorème 0.1 (Méthode des multiplicateurs de Lagrange). Les points critiques du problème d'optimisation contrainte ci-dessus sont des points critiques sans contrainte de la fonction multiplicateur de Lagrange

$$\Lambda(x,\lambda) \equiv f(x) - \lambda \cdot g(x),$$

par rapport à x et λ.

Exemple 0.26 (Zone de maximisation). En continuant l'exemple 0.25, nous définissons la fonction multiplicateur de Lagrange Λ(w, h, λ) = wh − λ(2w + 2h − 1). En différenciant, on trouve :

$$\frac{\partial \Lambda}{\partial w} = h - 2\lambda \; 0 = w - 2\lambda$$
$$\frac{\partial \Lambda}{\partial h} = 0$$
$$\frac{\partial \Lambda}{\partial \lambda} = 1 - 2w - 2h \; 0$$

Ainsi, les points critiques du système satisfont

$$\begin{matrix} 0 & 1 & -2 \\ 1 & 0 & -2 \\ 2 & 2 & 0 \end{matrix} \begin{matrix} w \\ h \\ \lambda \end{matrix} = \begin{matrix} 0 \\ 0 \\ 1 \end{matrix}$$

La résolution du système montre w = h = 1/4 et λ = 1/8. En d'autres termes, pour un périmètre fixe, le rectangle d'aire maximale est un carré.

Exemple 0.27 (Problèmes propres). Supposons que A est une matrice symétrique définie positive, c'est-à-dire A = A (symétrie) et x Ax > 0 pour tout x Rn\{0} (définie positive). Souvent on souhaite minimiser x Ax sous réserve de x = 1 pour une matrice donnée A Rn×n ; notez que sans la contrainte, le minimum a lieu trivialement à x = 0. On définit la fonction multiplicateur de Lagrange

$$\Lambda(x, \lambda) = x\,Ax - \lambda(x \quad {}^{2}_{2} - 1)$$
$$= x\,Ax - \lambda(x\,x - 1).$$

En différenciant par rapport à x, on trouve

$$0 = \quad x\Lambda = 2Ax - 2\lambda x$$

Autrement dit, x est un vecteur propre de la matrice A :

$$Ax = \lambda x.$$

0.5 Problèmes

Problème 0.1. Soit C1 (R) l'ensemble des fonctions f : R → R qui admettent une dérivée première f (x). Pourquoi 1 C (R) un espace vectoriel ? Montrer que C1 (R) est de dimension ∞.

Problème 0.2. Supposons que les lignes de A Rm×n soient données par les transposées de r1, . . . ,rm Rn et les colonnes de A Rm×n sont données par c1, . . . ,cn Rm. C'est,

$$Un = \begin{bmatrix} - & r1 & - \\ - & r2 & - \\ & \vdots & \\ - & rm & - \end{bmatrix} = \begin{bmatrix} | & | & | \\ c1 & c2 & \cdots & cn \\ | & | & | \end{bmatrix}.$$

Donner des expressions pour les éléments de AA et AA en fonction de ces vecteurs.

Problème 0.3. Donner un système linéaire d'équations satisfaites par les minima de l'énergie f(x) = Ax −b par rapport à x, pour x Rn et b A Rm×n. Ce système est appelé les « équations normales » et apparaîtra ailleurs dans ces notes ; même ainsi, cela vaut la peine de travailler et de comprendre pleinement la dérivation.

Problème 0.4. Supposons A, B Rn×n . Formuler une condition pour que les vecteurs x Rn soient des points critiques Hache du sujet à Bx = 1. Donner également une forme alternative pour les valeurs optimales de Ax .

Problème 0.5. Fixons un vecteura Rn\{0} et définissons f(x) =a · x. Donner une expression pour le maximum de f(x) sous réserve de x = 1.

Chapitre 1

Numériques et analyse d'erreurs

En étudiant l'analyse numérique, nous passons du traitement des entiers et des longs aux flottants et aux doubles.
Cette transition apparemment innocente comprend un énorme changement dans la façon dont nous devons penser à la conception et à la mise en œuvre des micros algorithmiques. Contrairement aux bases des algorithmes discrets, on ne peut plus s'attendre à nos algorithmes pour produire des solutions exactes dans tous les cas. "Big O" et le comptage des opérations ne font pas toujours règne suprême; au lieu de cela, même en comprenant les techniques les plus élémentaires, nous sommes obligés d'étudier le compromis entre le timing, l'erreur d'approximation, etc.

1.1 Stocker des nombres avec des parties fractionnaires

Rappelons que les ordinateurs stockent généralement les données au format binaire. En particulier, chaque chiffre d'un positif entier correspond à une puissance différente de deux. Par exemple, nous pourrions convertir 463 en binaire à l'aide du tableau suivant :

1	1	1	0	0	1			1	1	1
8^2	7^2	6^2		5^2		4^2	3^2	2^2	1^2	2^0

En d'autres termes, cette notation encode le fait que 463 peut être décomposé en puissances de deux uniquement comme :

$$463 = 2^{8} + 2^{7} + 2^{6} + 2^{3} + 2^{2} + 2^{1} + 2^{0}$$
$$= 256 + 128 + 64 + 8 + 4 + 2 + 1$$

Mis à part les problèmes de débordement, tous les entiers positifs peuvent être écrits sous cette forme en utilisant un nombre fini de chiffres. Les nombres négatifs peuvent également être représentés de cette façon, soit en introduisant un signe avant-coureur bit ou en utilisant l'astuce du "complément à deux".

Une telle décomposition inspire une extension simple aux nombres qui incluent des fractions : simplement inclure des puissances négatives de deux. Par exemple, décomposer 463,25 est aussi simple que d'ajouter deux emplacements :

1	1	1	0	0	1	3	2		1	1	1.	0	0	2	1
8^2	7^2	6^2		5^2		4^2			2^2	1^2			2^{-1}		2^{-2}

Cependant, tout comme dans le système décimal, représenter des parties fractionnaires de nombres de cette manière est pas aussi bien que de représenter des nombres entiers. Par exemple, écrire la fraction 1/3 en binaire donne l'expression :

$$\frac{1}{3} = 0{,}0101010101\ldots$$

De tels exemples montrent qu'il existe des nombres à toutes les échelles qui ne peuvent pas être représentés à l'aide d'un chaîne binaire finie. En fait, des nombres comme π = 11.00100100001 . . .2 ont des expansions infiniment longues quelle que soit la base (entière) que vous utilisez !

Pour cette raison, lors de la conception de systèmes informatiques qui effectuent des calculs sur R au lieu de Z, nous sont obligés de faire des approximations pour presque toutes les représentations numériques raisonnablement efficaces. Cela peut entraîner de nombreux points de confusion lors du codage. Par exemple, considérez le C++ suivant fragment:

```
double x = 1,0 ;
y double = x / 3,0 ;
if ( x == y *3.0) cout << Ils sont égaux ! "                    ;
sinon cout <<              Ils ne sont PAS égaux.                ;
```

Contrairement à l'intuition, ce programme affiche "Ils ne sont PAS égaux". Pourquoi? La définition de y fait une approximation à 1/3 car il ne peut pas être écrit comme une chaîne binaire de fin, arrondissant à un nombre proche qu'il peut représenter. Ainsi, y*3.0 ne multiplie plus 3 par 1/3. Une façon de réparer ce problème est ci-dessous:

```
double x = 1,0 ;
y double = x / 3,0 ;
if ( fabs (x - y *3.0) < numeric_limits < double >:: epsilon ) cout << else cout << Ils ne sont PAS égaux .    Ils sont égaux ! "                ;
                                                                                                                ;
```

Ici, nous vérifions que x et y*3.0 sont dans une certaine tolérance l'un de l'autre plutôt que de vérifier égalité exacte. Ceci est un exemple d'un point très important:

Rarement, voire jamais, l'opérateur == et ses équivalents doivent être utilisés sur des valeurs fractionnaires.
Au lieu de cela, une certaine tolérance doit être utilisée pour vérifier si les nombres sont égaux.

Bien sûr, il y a ici un compromis : la taille de la tolérance définit une ligne entre l'égalité et « proche-mais-pas-le-même », qui doit être choisi avec soin pour une application donnée.

Nous considérons ci-dessous quelques options pour représenter les nombres sur un ordinateur.

1.1.1 Représentations en virgule fixe

L'option la plus simple pour stocker des fractions consiste à ajouter un point décimal fixe. C'est-à-dire, comme dans l'exemple ci-dessus, nous représentons des valeurs en stockant des coefficients 0/1 devant des puissances de deux qui vont de 2−k à 2 pour certains k, Z. Par exemple, représentant toutes les valeurs non négatives entre 0 et 127,75 par incréments de 1/4 est aussi simple que de prendre k = 2 et = 7 ; dans cette situation, nous représentons ces valeurs utilisant 9 chiffres binaires, dont deux après la virgule décimale.

Le principal avantage de cette représentation est que presque toutes les opérations arithmétiques peuvent être effectuée en utilisant les mêmes algorithmes que pour les nombres entiers. Par exemple, il est facile de voir que

$$une + b = (une \cdot 2^{k} + b \cdot 2^{k}) \cdot 2^{-k}.$$

Multiplier notre représentation fixe par 2k garantit que le résultat est intégral, donc cette observation montre essentiellement que l'addition peut être effectuée en utilisant l'addition d'entiers essentiellement en "ignorant" la virgule décimale. Ainsi, plutôt que d'utiliser du matériel spécialisé, l'entier préexistant L'unité arithmétique et logique (ALU) effectue rapidement des calculs en virgule fixe.

L'arithmétique en virgule fixe peut être rapide, mais elle peut souffrir de sérieux problèmes de précision. En particulier, il arrive souvent que la sortie d'une opération binaire telle que la multiplication ou la division nécessite plus de bits que les opérandes. Par exemple, supposons que nous incluions une décimale de précision et

souhaite réaliser le produit 1/2 · 1/2 = 1/4. Nous écrivons 0,12 × 0,12 = 0,012, qui est tronqué à 0. Dans ce système, il est assez simple de combiner des nombres à virgule fixe de manière raisonnable et d'obtenir un résultat déraisonnable.

En raison de ces inconvénients, la plupart des principaux langages de programmation n'incluent pas par défaut un type de données décimal à virgule fixe. La vitesse et la régularité de l'arithmétique à pont fixe peuvent cependant constituer un avantage considérable pour les systèmes qui privilégient la synchronisation à la précision. En fait, certaines unités de traitement graphique (GPU) bas de gamme n'implémentent que ces opérations car quelques décimales de précision suffisent pour de nombreuses applications graphiques.

1.1.2 Représentations en virgule flottante

L'un des nombreux défis numériques dans la rédaction d'applications scientifiques est la variété des échelles qui peuvent apparaître. Seuls les chimistes traitent des valeurs comprises entre 9,11 × 10−31 et 6,022 × 1023. Une opération aussi innocente qu'un changement d'unités peut provoquer une transition brutale entre ces régimes : la même observation écrite en kilogrammes par année-lumière sera considérablement différente en mégatonnes par seconde. En tant qu'analystes numériques, notre travail consiste à écrire des logiciels qui peuvent passer d'une échelle à l'autre sans imposer au client des restrictions artificielles sur ses techniques.

Quelques notions et observations de l'art de la mesure scientifique sont pertinentes pour un tel discussion. Premièrement, évidemment l'une des représentations suivantes est plus compacte que l'autre :

6,022 × 1023 = 602, 200, 000, 000, 000, 000, 000, 000

De plus, en l'absence d'équipement scientifique exceptionnel, la différence entre 6,022 × 1023 et 6,022 × 1023 + 9,11 × 10−31 est négligeable. Une façon d'arriver à cette conclusion est de dire que 6,022 × 1023 n'a que trois chiffres de précision et représente probablement une gamme de mesures possibles [6,022 × 1023 − ε, 6,011 × 1023 + ε] pour un certain ε 0,001 × 1023 .

Notre première observation a pu compactifier notre représentation de 6,022 × 1023 en l'écrivant en notation scientifique. Ce système numérique sépare les chiffres "intéressants" d'un nombre de son ordre de grandeur en l'écrivant sous la forme a × 10b pour certains a 1 et b Z. Nous appelons ce format la forme à virgule flottante d'un nombre, car contrairement à la configuration en virgule fixe du §1.1.1, ici la virgule décimale « flotte » vers le haut. On peut décrire les systèmes à virgule flottante en utilisant quelques paramètres (CITE) :

- La base β N ; pour la notation scientifique expliquée ci-dessus, la base est 10
- La précision p N représentant le nombre de chiffres dans le développement décimal
- La plage des exposants [L, U] représentant les valeurs possibles de b

Une telle extension ressemble à :

$$\underbrace{\pm}_{\text{signe}} \underbrace{(d_0 + d_1 \cdot \beta^{-1} + d_2 \cdot \beta^{-2} + \cdots + d_{p-1} \cdot \beta^{1-p})}_{\text{mantisse}} \times \underbrace{\beta^{b}}_{\text{exposant}}$$

où chaque chiffre dk est dans l'intervalle [0, β − 1] et b [L, U].

Les représentations en virgule flottante ont une curieuse propriété qui peut affecter le logiciel de manière inattendue : leur espacement est irrégulier. Par exemple, le nombre de valeurs représentables entre β et β^2 est le même que celui entre β^2 et β^3 même si généralement $\beta^3 - \beta^2 > \beta^2 - \beta$. Pour comprendre la précision possible avec un système numérique donné, nous définirons la précision machine εm comme la

plus petit εm > 0 tel que 1 + εm soit représentable. Alors, des nombres comme β + εm ne sont pas exprimables dans le système de numération car εm est trop petit !

La norme de loin la plus courante pour le stockage des nombres à virgule flottante est la norme IEEE 754. Cette norme spécifie plusieurs classes de nombres à virgule flottante. Par exemple, un nombre à virgule flottante double précision est écrit en base β = 2 (comme la plupart des nombres sur l'ordinateur), avec un seul bit de signe ± , 52 chiffres pour d et une plage d'exposants entre -1022 et 1023. La norme spécifie également comment stocker ±∞ et des valeurs telles que NaN, ou "not-a-number", réservées aux résultats de calculs tels que 10/0. Une précision supplémentaire peut être obtenue en écrivant des valeurs à virgule flottante normalisées et en supposant que le chiffre le plus significatif d0 est 1 et en ne l'écrivant pas.

La norme IEEE comprend également des options convenues pour traiter le nombre fini de valeurs pouvant être représentées en fonction d'un nombre fini de bits. Par exemple, une stratégie commune non biaisée pour arrondir les calculs est d'arrondir au plus proche, lié à pair, ce qui rompt les liens équidistants en arrondissant à la valeur à virgule flottante la plus proche avec un bit encore moins significatif (le plus à droite). Notez qu'il existe de nombreuses stratégies tout aussi légitimes pour arrondir; en choisir un seul garantit que les logiciels scientifiques fonctionneront à l'identique sur toutes les machines clientes implémentant le même standard.

1.1.3 Options plus exotiques

À l'avenir, nous supposerons que les valeurs décimales sont stockées au format à virgule flottante, sauf indication contraire. Ceci, cependant, ne veut pas dire que d'autres systèmes numériques n'existent pas, et pour des applications spécifiques, un choix alternatif pourrait être nécessaire. Nous reconnaissons certaines de ces situations ici.

Le casse-tête lié à l'ajout de tolérances pour tenir compte des erreurs d'arrondi peut être inacceptable pour certaines applications. Cette situation apparaît dans les applications de géométrie computationnelle, par exemple lorsque la différence entre des lignes presque parallèles et complètement parallèles peut être une distinction difficile à faire. Une solution pourrait être d'utiliser l'arithmétique de précision arbitraire, c'est-à-dire d'implémenter l'arithmétique sans arrondi ni erreur d'aucune sorte.

L'arithmétique à précision arbitraire nécessite une implémentation spécialisée et une réflexion approfondie sur les types de valeurs que vous devez représenter. Par exemple, il se peut que les nombres rationnels Q soient suffisants pour une application donnée, ce qui peut être écrit sous la forme de rapports a/b pour a, b Z.
Les opérations arithmétiques de base peuvent être effectuées dans Q sans aucune perte de précision. Par exemple, il est facile de voir

$$\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd} \qquad \frac{a}{b} \div \frac{c}{d} = \frac{ad}{bc}.$$

L'arithmétique dans les rationnels exclut l'existence d'un opérateur racine carrée, puisque des valeurs comme $\sqrt{2}$ sont irrationnelles. De plus, cette représentation n'est pas unique, puisque par exemple a/b = 5a/5b.

D'autres fois, il peut être utile de mettre l'erreur entre parenthèses en représentant les valeurs à côté des estimations d'erreur sous la forme d'une paire a, ε R ; nous considérons le couple (a,ε) comme l'intervalle a ± ε. Ensuite, les opérations arithmétiques mettent également à jour non seulement la valeur, mais également l'estimation de l'erreur, comme dans

$$(x \pm \varepsilon 1) + (y \pm \varepsilon 2) = (x + y) \pm (\varepsilon 1 + \varepsilon 2 + \text{erreur}(x + y)),$$

où le terme final représente une estimation de l'erreur induite par l'addition de x et y.

1.2 Comprendre l'erreur

À l'exception des systèmes à précision arbitraire décrits au §1.1.3, presque toutes les représentations informatisées de nombres réels avec des parties fractionnaires sont obligées d'employer des arrondis et d'autres schémas d'approximation. Ce schéma représente l'une des nombreuses sources d'approximations généralement rencontrées dans les systèmes numériques :

- L'erreur de troncature vient du fait que nous ne pouvons représenter qu'un sous-ensemble fini de tout l'ensemble possible de valeurs dans R ; par exemple, nous devons tronquer des séquences longues ou infinies au-delà de la virgule décimale jusqu'au nombre de bits que nous sommes prêts à stocker.
- L'erreur de discrétisation provient de nos adaptations informatisées du calcul, de la physique et d'autres aspects des mathématiques continues. Par exemple, on fait une approximation

$$\frac{dx}{} \approx \frac{y(x + \varepsilon) - y(x)}{\varepsilon}.$$

  Nous apprendrons que cette approximation est légitime et utile, mais selon le choix de ε , elle peut ne pas être complètement correcte.

- L'erreur de modélisation provient de descriptions incomplètes ou inexactes des problèmes que l'on souhaite résoudre. Par exemple, une simulation pour prédire le temps en Allemagne peut choisir de négliger le battement collectif des ailes de papillon en Malaisie, bien que le déplacement de l'air par ces papillons puisse être suffisant pour perturber quelque peu les conditions météorologiques ailleurs.
- L'erreur constante empirique provient de mauvaises représentations des constantes physiques ou mathématiques. Par exemple, nous pouvons calculer π en utilisant une séquence de Taylor que nous terminons tôt, et même les scientifiques peuvent même ne pas connaître la vitesse de la lumière à plus d'un certain nombre de chiffres.
- L'erreur de saisie peut provenir d'approximations générées par l'utilisateur des paramètres d'un système donné (et de fautes de frappe !). La simulation et les techniques numériques peuvent être utilisées pour répondre à des questions de type "et si", dans lesquelles des choix exploratoires de configurations d'entrée sont choisis uniquement pour avoir une idée du comportement d'un système.

Exemple 1.1 (Physique computationnelle). Supposons que nous concevions un système pour simuler des planètes pendant qu'elles tournent autour de la terre. Le système résout essentiellement l'équation de Newton F = ma en intégrant les forces dans le temps. Des exemples de sources d'erreurs dans ce système peuvent inclure :

- Erreur de troncature : utilisation de la virgule flottante IEEE pour représenter les paramètres et la sortie du système et troncature lors du calcul du produit ma
- Erreur de discrétisation : remplacement de l'accélération a par une différence divisée
- Erreur de modélisation : négliger de simuler les effets de la lune sur le mouvement de la terre dans le plan planétaire système
- Erreur empirique : ne saisir la masse de Jupiter qu'à quatre chiffres
- Erreur de saisie : l'utilisateur peut souhaiter évaluer le coût de l'envoi de déchets dans l'espace plutôt que de risquer une accumulation de style Wall-E sur Terre, mais ne peut qu'estimer la quantité de déchets que le gouvernement est prêt à jeter de cette manière.

1.2.1 Erreur de classification

Compte tenu de notre discussion précédente, les deux nombres suivants pourraient être considérés comme ayant la même quantité d'erreur potentielle :

$$1 \pm 0{,}01$$

$$105 \pm 0{,}01$$

Bien qu'il ait la taille comme plage [1 - 0,01, 1 + 0,01], la plage [105 - 0,01, 105 + 0,01] semble coder une mesure plus fiable car l'erreur 0,01 est beaucoup plus petite par rapport à 105 qu'à 1.

La distinction entre ces deux classes d'erreur est décrite en différenciant ab
erreur de solution et erreur relative :

Définition 1.1 (Erreur absolue). L'erreur absolue d'une mesure est donnée par la différence entre la valeur approximative et sa valeur réelle sous-jacente.

Définition 1.2 (Erreur relative). L'erreur relative d'une mesure est donnée par l'erreur absolue divisée par la valeur vraie.

Une façon de faire la distinction entre ces deux types d'erreur est l'utilisation d'unités par rapport à des pourcentages.

Exemple 1.2 (Erreur absolue et relative). Voici deux énoncés équivalents sous des formes contrastées :

Absolu : 2 pouces ± 0,02 pouces

Relatif : 2 à ± 1 %

Dans la plupart des applications, la vraie valeur est inconnue ; après tout, si ce n'était pas le cas, l'utilisation d'une approximation au lieu de la vraie valeur pourrait être une proposition douteuse. Il existe deux façons populaires de résoudre ce problème. La première consiste simplement à être conservateur lors de la réalisation des calculs : à chaque étape, prenez l'estimation d'erreur la plus grande possible et propagez ces estimations vers l'avant si nécessaire. De telles estimations conservatrices sont puissantes en ce sens que lorsqu'elles sont petites, nous pouvons être très confiants que notre solution est utile.

Une résolution alternative a à voir avec ce que vous pouvez mesurer. Par exemple, supposons que nous souhaitions résoudre l'équation $f(x) = 0$ pour x étant donné une fonction $f : R \to R$. Nous savons qu'il existe quelque part une racine $x0$ satisfaisant exactement $f(x0) = 0$, mais si nous le savions root notre algorithme ne serait pas nécessaire en premier lieu. En pratique, notre système de calcul peut produire un xest satisfaisant $f(xest) = \varepsilon$ pour un certain ε avec $|\varepsilon|$ 1. Nous ne pourrons peut-être pas évaluer la différence $x0 - xest$ puisque $x0$ est inconnu. D'autre part, simplement en évaluant f nous pouvons calculer $f(xest) - f(x0) \equiv f(xest)$ puisque nous savons $f(x0) = 0$ par définition. Cette valeur donne une certaine notion d'erreur pour notre calcul.

Cet exemple illustre la distinction entre erreur avant et erreur arrière. L'erreur directe faite par une approximation définit très probablement notre intuition pour l'analyse des erreurs comme la différence entre la solution approchée et la solution réelle, mais comme nous l'avons discuté, il n'est pas toujours possible de calculer. L'erreur en arrière, cependant, a la particularité d'être calculable mais pas notre objectif exact lors de la résolution d'un problème donné. Nous pouvons ajuster notre définition et notre interprétation de l'erreur vers l'arrière à mesure que nous abordons différents problèmes, mais une définition appropriée si vague est la suivante :

Définition 1.3 (Erreur rétrograde). L'erreur vers l'arrière est donnée par la quantité qu'un énoncé de problème devrait changer pour réaliser une approximation donnée de sa solution.

Cette définition est quelque peu obtuse, nous illustrons donc son utilisation dans quelques exemples.

Exemple 1.3 (Systèmes linéaires). Supposons que nous souhaitions résoudre le système linéaire n × n Ax = b. Appelons la vraie solution x0 ≡ A −1b. En réalité, en raison d'une erreur de troncature et d'autres problèmes, notre système donne une solution proche xest. L'erreur directe de cette approximation est évidemment mesurée à l'aide de la différence xest -x0 ; en pratique cette valeur est impossible à calculer car on ne connaît pas x0. En réalité, xest est la solution exacte d'un système modifié Ax = best for best ≡ Axest ; ainsi, nous pourrions mesurer l'erreur vers l'arrière en termes de différenceb −best. Contrairement à l'erreur directe, cette erreur est facilement calculable sans inverser A, et il est facile de voir que xest est une solution au problème exactement lorsque l'erreur arrière (ou avant) est nulle.

Exemple 1.4 (Résolution d'équations, CITE). Supposons que nous écrivions une fonction pour trouver les racines carrées de nombres positifs qui génère $\sqrt{2} \approx 1{,}4$. L'erreur directe est $|1{,}4 - 1{,}41421 \cdots| \approx 0{,}0142$. Notez que $1{,}42 = 1{,}96$, donc l'erreur vers l'arrière est $|1{,}96 - 2| = 0{,}04$.

Les deux exemples ci-dessus démontrent un modèle plus large selon lequel l'erreur vers l'arrière peut être beaucoup plus facile à calculer que l'erreur vers l'avant. Par exemple, l'évaluation de l'erreur directe dans l'exemple 1.3 nécessitait l'inversion d'une matrice A tandis que l'évaluation de l'erreur inverse ne nécessitait qu'une multiplication par A. De même, dans l'exemple 1.4, la transition de l'erreur directe à l'erreur inverse a remplacé le calcul de la racine carrée par la multiplication.

1.2.2 Conditionnement, stabilité et précision

Dans presque tous les problèmes numériques, zéro erreur vers l'arrière implique zéro erreur vers l'avant et vice versa. Ainsi, un logiciel conçu pour résoudre un tel problème peut sûrement se terminer s'il trouve qu'une solution candidate n'a aucune erreur de retour. Mais que se passe-t-il si l'erreur vers l'arrière est différente de zéro mais petite ?

Cela implique-t-il nécessairement une petite erreur vers l'avant ? De telles questions motivent l'analyse de la plupart des techniques numériques dont l'objectif est de minimiser l'erreur vers l'avant mais qui, en pratique, ne peuvent mesurer que l'erreur vers l'arrière.

Nous souhaitons analyser les changements d'erreur vers l'arrière par rapport à l'erreur vers l'avant afin que nos algorithmes puissent dire avec confiance en utilisant uniquement l'erreur vers l'arrière qu'ils ont produit des solutions acceptables. Cette relation peut être différente pour chaque problème que nous souhaitons résoudre, donc à la fin nous faisons la classification approximative suivante :

- Un problème est insensible ou bien conditionné lorsque de petites quantités d'erreur vers l'arrière impliquent de petites quantités d'erreur vers l'avant. En d'autres termes, une petite perturbation de l'énoncé d'un problème bien conditionné ne produit qu'une petite perturbation de la vraie solution.

- Un problème est sensible ou mal conditionné alors que ce n'est pas le cas.

Exemple 1.5 (ax = b). Supposons comme exemple jouet que nous voulons trouver la solution x0 ≡ b/a à l'équation linéaire ax = b pour a, x, b R. L'erreur directe d'une solution potentielle x est donnée par x − x0 tandis que l'erreur arrière est donnée par b − ax = a(x − x0). Ainsi, quand |a| 1, le problème est bien conditionné puisque de petites valeurs d'erreur vers l'arrière a(x − x0) impliquent des valeurs encore plus petites de x − x0 ; au contraire, quand |a| 1 le problème est mal conditionné, car même si a(x − x0) est petit, l'erreur directe x − x0 ≡ 1/a · a(x − x0) peut être grande compte tenu du facteur 1/a .

Nous définissons le nombre de condition comme une mesure de la sensibilité d'un problème :

Définition 1.4 (Numéro de condition). Le nombre de condition d'un problème est le rapport entre la quantité de modification de sa solution et la quantité de modification de son énoncé sous de petites perturbations. Alternativement, c'est le rapport entre l'erreur avant et l'erreur arrière pour de petits changements dans l'énoncé du problème.

Exemple 1.6 (ax = b, deuxième partie). En continuant l'exemple 1.5, nous pouvons calculer le nombre de condition exactement :

$$c = \frac{\text{erreur vers l'avant}}{\text{erreur vers l'arrière}} = \frac{x - x0}{un(x - x0)} \equiv \frac{1}{un}$$

En général, le calcul des nombres de condition est presque aussi difficile que le calcul de l'erreur directe, et donc leur calcul exact est probablement impossible. Même ainsi, il est souvent possible de trouver des limites ou des approximations pour les nombres de conditions pour aider à évaluer dans quelle mesure une solution peut être fiable.

Exemple 1.7 (Recherche de racine). Supposons qu'on nous donne une fonction lisse f : R → R et que nous voulions trouver des valeurs x avec f(x) = 0. Notez que f(x + ∆) ≈ f(x) + ∆f (x). Ainsi, une approximation du nombre de conditions pour trouver x pourrait être

$$\frac{\text{modification de l'erreur vers l'avant}}{\text{modification de l'erreur vers l'arrière}} = \frac{(x + \Delta) - x\, f(x}{+ \Delta) - f(x)\, \Delta}$$

$$\approx \frac{}{\Delta f(x)}$$

$$= \frac{1}{f(x)}$$

Notez que cette approximation s'aligne sur celle de l'exemple 1.6. Bien sûr, si nous ne connaissons pas x, nous ne pouvons pas évaluer f (x), mais si nous pouvons regarder la forme de f et borner | f | près de x, nous avons une idée de la situation la plus défavorable.

Les erreurs avant et arrière sont des mesures de la précision d'une solution. Dans un souci de répétabilité scientifique, nous souhaitons également dériver des algorithmes stables qui produisent des solutions auto-cohérentes à une classe de problèmes. Par exemple, un algorithme qui génère des solutions très précises seulement un cinquième du temps peut ne pas valoir la peine d'être mis en œuvre, même si nous pouvons revenir en arrière en utilisant les techniques ci-dessus pour vérifier si la solution candidate est bonne.

1.3 Aspects pratiques

L'infinitude et la densité des nombres réels R peuvent provoquer des bugs pernicieux lors de l'implémentation d'algorithmes numériques. Alors que la théorie de l'analyse des erreurs introduite au §1.2 nous aidera éventuellement à mettre des garanties sur la qualité des techniques numériques introduites dans les prochains chapitres, il convient de noter avant de poursuivre un certain nombre d'erreurs courantes et de « pièges » qui imprègnent les implémentations des méthodes numériques.

Nous avons délibérément introduit le plus gros contrevenant au début du §1.1, que nous répétons dans une police plus grande pour une

emphase bien méritée : rarement, voire jamais, l'opérateur == et ses équivalents ne doivent être utilisés sur des valeurs fractionnaires.

Trouver un remplacement approprié pour == et les conditions correspondantes pour terminer une méthode numérique dépend de la technique considérée. L'exemple 1.3 montre qu'une méthode de résolution Ax = b peut se terminer lorsque le résidu b − Ax est nul ; puisque nous ne voulons pas vérifier explicitement si A*x==b, en pratique les implémentations vérifieront norm(A*xb)<epsilon. Notez que cet exemple illustre deux techniques :

- L'utilisation de l'erreur vers l'arrièreb − Ax plutôt que l'erreur vers l'avant pour déterminer quand terminer, et
- Vérifier si l'erreur vers l'arrière est inférieure à epsilon pour éviter le prédicat interdit ==0.

Le paramètre epsilon dépend de la précision de la solution souhaitée ainsi que de la résolution du système numérique utilisé.

Un programmeur utilisant ces types de données et ces opérations doit être vigilant lorsqu'il s'agit de détecter et de prévenir les opérations numériques médiocres. Par exemple, considérons l'extrait de code suivant pour calculer la norme x2 pour un vecteur x Rn représenté sous la forme d'un tableau 1D x[] :

```
double normeCarré = 0 ; for ( int je
= 0; je < n ; je ++) normSquared += x [ je ]*
                x [ je ];
return sqrt ( normSquared );
```

Il est facile de voir qu'en théorie mini |xi | ≤ x2/ √ n ≤ maxi |xi |, c'est-à-dire que la norme de x est de l'ordre des valeurs des éléments contenus dans x. Caché dans le calcul de x2, cependant, se trouve l'expression x[i]*x[i]. S'il existe i tel que x[i] soit de l'ordre de DOUBLE MAX, le produit x[i]*x[i] débordera même si x2 est toujours dans l'intervalle des doubles. Un tel débordement est facilement évitable en divisant x par sa valeur maximale, en calculant la norme et en multipliant :

```
double maxElement = epsilon ; // ne veut pas diviser par zéro ! pour ( int je = 0; je < n ; je ++)

                maxElement = max ( maxElement for ( int    ,    fabs ( x [ je ]));
je = 0; je < n ; je ++) {
                double mise à l'échelle = x [ i ] / maxElement ;
                normSquared += mis à l'échelle * mis à l'échelle ;
}
return sqrt ( normSquared ) * maxElement ;
```

Le facteur d'échelle supprime le problème de débordement en s'assurant que les éléments additionnés ne sont pas supérieurs à 1.

Ce petit exemple montre l'une des nombreuses circonstances dans lesquelles un seul caractère de code peut conduire à un problème numérique non évident. Alors que notre intuition des mathématiques continues est suffisante pour générer de nombreuses méthodes numériques, nous devons toujours revérifier que les opérations que nous employons sont valides d'un point de vue discret.

1.3.1 Exemple à plus grande échelle : sommation

Nous donnons maintenant un exemple d'un problème numérique causé par l'arithmétique à précision finie qui peut être résolu en utilisant une astuce algorithmique moins qu'évidente.

Supposons que nous souhaitions additionner une liste de valeurs à virgule flottante, facilement une tâche requise par les systèmes de comptabilité, d'apprentissage automatique, de graphisme et de presque tous les autres domaines. Un extrait de code pour accomplir cette tâche qui apparaît sans aucun doute dans d'innombrables applications se présente comme suit :

```
double somme = 0 ; pour
( int je = 0; je < n ; je ++) somme += x [ je ];
```

Avant de poursuivre, il convient de noter que pour la grande majorité des applications, il s'agit d'une technique parfaitement stable et certainement mathématiquement valide.

Mais qu'est-ce qui peut mal tourner ? Considérons le cas où n est grand et la plupart des valeurs x[i] sont petites et positives. Dans ce cas, lorsque i est suffisamment grand, la somme des variables sera grande par rapport à x[i]. Finalement, la somme pourrait être si grande que x[i] n'affecte que les bits d'ordre le plus bas de la somme, et dans le cas extrême, la somme pourrait être suffisamment grande pour que l'ajout de x[i] n'ait aucun effet. Bien qu'une seule erreur de ce type puisse ne pas être un gros problème, l'effet cumulé de la répétition de cette erreur pourrait dépasser le montant auquel nous pouvons faire confiance.

Pour comprendre mathématiquement cet effet, supposons que le calcul d'une somme a + b puisse être décalé d'autant que ε > 0. Ensuite, la méthode ci-dessus peut clairement induire une erreur de l'ordre de nε, qui croît linéairement avec n. En fait, si la plupart des éléments x[i] sont de l'ordre de ε, alors la somme n'est absolument pas fiable ! C'est un résultat décevant : l'erreur peut être aussi grande que la somme elle-même.

Heureusement, il existe de nombreuses façons de faire mieux. Par exemple, ajouter les plus petites valeurs en premier peut aider à tenir compte de leur effet cumulé. Les méthodes par paires ajoutant récursivement des paires de valeurs à partir de x[] et construisant une somme sont également plus stables, mais elles peuvent être difficiles à implémenter aussi efficacement que la boucle for ci-dessus. Heureusement, un algorithme de Kahan (CITE) fournit une méthode de « sommation compensée » facile à mettre en œuvre et presque aussi rapide.

L'observation utile ici est que nous pouvons en fait suivre une approximation de l'erreur en somme au cours d'une itération donnée. En particulier, considérons l'expression

$$((a + b) - a) - b.$$

Évidemment cette expression est algébriquement nulle. Numériquement, cependant, cela peut ne pas être le cas. En particulier, la somme (a + b) peut arrondir le résultat pour le maintenir dans le domaine des valeurs à virgule flottante. Soustraire a et b un à un donne alors une approximation de l'erreur induite par cette opération ; notez que les opérations de soustraction sont probablement mieux conditionnées car le passage des grands nombres aux petits ajoute des chiffres de précision en raison de l'annulation.

Ainsi, la technique de Kahan procède comme suit :

```
double somme = 0 ;
compensation double = 0 ; // une approximation de l'erreur

for ( int i = 0; i < n ; i ++) { // essayez de rajouter à
                la fois x [ i ] et la partie manquante double nextTerm = x [ i ] + compensation ;


                // calcule le résultat de sommation de cette itération double nextSum = sum +
                nextTerm ;

                // calcule la compensation comme la différence entre le terme que vous vouliez // ajouter et le résultat réel compensation =
                nextTerm - ( nextSum - sum );


                somme = sommesuivante ;
}
```

Au lieu de simplement maintenir la somme, nous gardons maintenant une trace de la somme ainsi qu'une compensation d'approximation de la différence entre la somme et la valeur souhaitée. A chaque itération, on essaie de rajouter

cette compensation en plus de l'élément courant de x[], puis nous recalculons la compensation pour tenir compte de la dernière erreur.

L'analyse de l'algorithme de Kahan nécessite une comptabilité plus minutieuse que l'analyse de la technique incrémentale plus simple. Vous découvrirez une dérivation d'une expression d'erreur à la fin de ce chapitre ; le résultat mathématique final sera que l'erreur s'améliore de nε à $O(\varepsilon + n\varepsilon^2)$, une amélioration considérable lorsque $0 < \varepsilon \ll 1$.

La mise en œuvre de la sommation de Kahan est simple mais fait plus que doubler le nombre d'opérations du programme résultant. De cette façon, il existe un compromis implicite entre vitesse et précision que les ingénieurs logiciels doivent faire lorsqu'ils décident quelle technique est la plus appropriée.

Plus largement, l'algorithme de Kahan est l'une des nombreuses méthodes qui contournent l'accumulation d'erreur numérique au cours d'un calcul composé de plus d'une opération. D'autres exemples incluent l'algorithme de Bresenham pour la rastérisation des lignes (CITE), qui utilise uniquement l'arithmétique entière pour tracer des lignes même lorsqu'elles croisent des lignes et des colonnes de pixels à des emplacements non entiers, et la transformée de Fourier rapide (CITE), qui utilise efficacement la partition binaire astuce de sommation décrite ci-dessus.

1.4 Problèmes

Problème 1.1. Voici un problème.

Partie II

Algèbre linéaire

Chapitre 2

Systèmes linéaires et LU Décomposition

Au chapitre 0, nous avons discuté d'une variété de situations dans lesquelles des systèmes linéaires d'équations Ax = b apparaissent dans la théorie mathématique et dans la pratique. Dans ce chapitre, nous abordons le problème de base de front et explorons des méthodes numériques pour résoudre de tels systèmes.

2.1 Solvabilité des systèmes linéaires

Comme introduit au §0.3.3, les systèmes d'équations linéaires comme

$$\begin{aligned} 3x + 2y &= 6 \\ -4x + y &= 7 \end{aligned}$$

peut s'écrire sous forme matricielle comme dans

$$\begin{bmatrix} 3 & 2 \\ -4 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \end{bmatrix}.$$

Plus généralement, on peut écrire des systèmes de la forme Ax =b pour $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, et $b \in \mathbb{R}^m$.

La solvabilité du système doit tomber dans l'un des trois cas suivants :

1. Le système peut ne pas admettre de solutions, comme dans :

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

   Ce système demande que x = −1 et x = 1 simultanément, évidemment deux conditions incompatibles.

2. Le système peut admettre une solution unique ; par exemple, le système au début de cette section est résolu par (x, y) = (−8/11, 45/11).

3. Le système peut admettre une infinité de solutions, par exemple 0x =0. Remarquons que si un système Ax =b admet deux solutions x0 et x1, alors il admet automatiquement une infinité de solutions de la forme cx0 + (1 − c)x1 pour c R, puisque

$$A(cx0 + (1 - c)x1) = cAx0 + (1 - c)Ax1 = cb + (1 - c)b = b.$$

Ce système linéaire serait étiqueté sous-déterminé.

En général, la solvabilité d'un système dépend à la fois de A et de onb. Par exemple, si nous modifions le système insoluble ci-dessus pour qu'il soit

$$\begin{matrix} dix & x & & 1 \\ dix & y & = & 1 \end{matrix},$$

alors le système passe de l'absence de solutions à une infinité de la forme (1, y). En fait, toute matrice A admet un second membre b tel que Ax = b est résoluble, car Ax = 0 peut toujours être résolu par x ≡ 0 quel que soit A. Rappelons du §0.3.1 que la multiplication matrice-vecteur peut être vue comme combinant linéairement les colonnes de A avec des poids de x. Ainsi, comme mentionné au §0.3.3, on peut s'attendre à ce que Ax =b soit résoluble exactement lorsque b est dans l'espace des colonnes de A.

D'une manière générale, la « forme » de la matrice A Rm×n a une influence considérable sur la solvabilité de Ax =b. Rappelons que les colonnes de A sont des vecteurs à m dimensions. Considérons d'abord le cas où A est "large", c'est-à-dire lorsqu'il a plus de colonnes que de lignes (n > m). Chaque colonne est un vecteur dans Rm, donc au plus l'espace des colonnes peut avoir une dimension m. Puisque n > m, les n colonnes de A doivent alors être linéairement dépendantes ; ceci implique qu'il existe un x0 =0 tel que Ax0 =0. Alors, si nous pouvons résoudre Ax = b pour x, alors A(x + αx0) = Ax + αAx0 = b +0 = b, montrant qu'il existe en fait une infinité de solutions x à Ax = b. En d'autres termes, nous avons montré qu'aucun système matriciel large n'admet une solution unique.

Lorsque A est "grand", c'est-à-dire lorsqu'il a plus de lignes que de colonnes (m > n), alors les n colonnes ne peuvent pas s'étendre sur Rm. Ainsi, il existe un vecteur b0 Rm\col A. Par définition, ce b0 ne peut pas satisfaire Ax =b0 pour tout x. En d'autres termes, toute matrice haute A admet des systèmes Ax =b0 qui ne sont pas résolubles.

Les deux situations ci-dessus sont loin d'être favorables à la conception d'algorithmes numériques. Par exemple, si un système linéaire admet de nombreuses solutions, nous devons d'abord définir quelle solution est souhaitée par l'utilisateur : après tout, la solution x + 1031x0 pourrait ne pas être aussi significative que x − 0,1x0. D'un autre côté, dans le cas haut même si Ax = b est résoluble pour un b particulier, toute petite perturbation Ax =b + εb0 n'est plus résoluble ; cette situation peut apparaître simplement parce que les procédures d'arrondi discutées dans le dernier chapitre ne peuvent qu'approximer A et b en premier lieu.

Compte tenu de ces complications, nous ferons dans ce chapitre quelques hypothèses simplificatrices :

- Nous ne considérerons que le carré A Rn×n ·
- Nous supposerons que A est non singulier, c'est-à-dire que Ax =b est résoluble pour tout b.

Rappelons du §0.3.3 que la condition de non-singularité revient à demander que les colonnes de A s'étend sur Rn et implique l'existence d'une matrice A^{-1} satisfaisant $A^{-1}A = AA^{-1} = I_{n\times n}$.

Une observation trompeuse est de penser que résoudre Ax =b équivaut à calculer explicitement la matrice puis A^{-1} à multiplier pour trouver x ≡ A −1b. Bien que cette stratégie de solution soit certainement valable, elle peut représenter une quantité considérable d'exagérations : après tout, nous ne nous intéressons qu'aux n 2 valeurs de x plutôt qu'aux n . De plus, même lorsque A se comporte bien, il se peut qu'écrire A valeurs dans A^{-1} donne des difficultés numériques qui peuvent être contournées.

2.2 Stratégies de solution ad hoc

Dans l'algèbre d'introduction, nous abordons souvent le problème de la résolution d'un système linéaire d'équations comme une forme d'art. La stratégie consiste à "isoler" les variables, en écrivant itérativement des formes alternatives du système linéaire jusqu'à ce que chaque ligne soit de la forme x = const.

Lors de la formulation d'algorithmes systématiques pour résoudre des systèmes linéaires, il est instructif de réaliser un exemple de ce processus de résolution. Considérez le système suivant :

$$\begin{aligned} y - z &= -1 \\ 3x - y + z &= 4 \\ x + y - 2z &= -3 \end{aligned}$$

En parallèle, nous pouvons maintenir une version matricielle de ce système. Plutôt que d'écrire Ax = b explicitement, nous pouvons économiser un peu d'espace en écrivant la matrice "augmentée" ci-dessous :

$$\left[\begin{array}{cccc} 0 & 1 & -1 & -1 \\ 3 & -1 & 1 & 4 \\ 1 & 1 & -2 & -3 \end{array}\right]$$

Nous pouvons toujours écrire des systèmes linéaires de cette façon tant que nous convenons que les variables restent à gauche des équations et les constantes à droite.

Peut-être souhaitons-nous d'abord traiter de la variable x. Pour plus de commodité, nous pouvons permuter les lignes du système afin que la troisième équation apparaisse en premier :

$$\begin{aligned} x + y - 2z &= -3 \\ y - z &= -1 \\ 3x - y + z &= 4 \end{aligned} \qquad \left[\begin{array}{cccc} 1 & 1 & -2 & -3 \\ 0 & 1 & -1 & -1 \\ 3 & -1 & 1 & 4 \end{array}\right]$$

Nous pouvons alors substituer la première équation dans la troisième pour éliminer le terme 3x. Cela revient à mettre à l'échelle la relation x + y − 2z = −3 par −3 et à ajouter le résultat à la troisième équation :

$$\begin{aligned} x + y - 2z &= -3 \\ y - z &= -1 \\ -4y + 7z &= 13 \end{aligned} \qquad \left[\begin{array}{cccc} 1 & 1 & -2 & -3 \\ 0 & 1 & -1 & -1 \\ 0 & -4 & 7 & 13 \end{array}\right]$$

De même, pour éliminer y de la troisième équation, nous pouvons multiplier la deuxième équation par 4 et ajouter le résultat à la troisième :

$$\begin{aligned} x + y - 2z &= -3 \\ y - z &= -1 \\ 3z &= 9 \end{aligned} \qquad \left[\begin{array}{cccc} 1 & 1 & -2 & -3 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 3 & 9 \end{array}\right]$$

Nous avons maintenant isolé z ! Ainsi, nous pouvons redimensionner la troisième ligne de 1/3 pour obtenir une expression pour z :

$$\begin{aligned} x + y - 2z &= -3 \\ y - z &= -1 \\ z &= 3 \end{aligned} \qquad \left[\begin{array}{cccc} 1 & 1 & -2 & -3 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

Maintenant, nous pouvons substituer z = 3 dans les deux autres équations pour supprimer z de toutes sauf la dernière ligne :

$$\begin{array}{l} x + y = 3 \\ y = 2 \\ z = 3 \end{array} \qquad \left|\begin{array}{cccc} 1 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right.$$

Enfin, nous effectuons une substitution similaire pour y pour terminer la résolution :

$$\begin{array}{l} x = 1 \\ y = 2 \\ z = 3 \end{array} \qquad \left|\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right.$$

Cet exemple est peut-être quelque peu pédant, mais en repensant à notre stratégie, nous obtenons quelques observations importantes sur la manière dont nous pouvons résoudre des systèmes linéaires :

- Nous avons écrit des systèmes successifs Aix = bi qui peuvent être vus comme des simplifications de l'original Hache =b.
- Nous avons résolu le système sans jamais écrire A^{-1}.
- Nous avons utilisé à plusieurs reprises quelques opérations simples : mise à l'échelle, ajout et permutation des lignes du système.
- Les mêmes opérations ont été appliquées à A et b. Si nous avons mis à l'échelle la k-ième ligne de A, nous avons également mis à l'échelle la k-ième rangée deb. Si nous ajoutons les lignes k et de A, nous ajoutons les lignes k et ofb.
- Moins évidemment, les étapes de résolution ne dépendaient pas de b. C'est-à-dire que toutes nos décisions sur la façon de résoudre ont été motivées par l'élimination des valeurs non nulles dans A plutôt que par l'examen des valeurs inb;b vient de faire le tour.
- Nous avons terminé lorsque nous avons réduit le système à In×nx =b.

Nous utiliserons toutes ces observations générales sur la résolution de systèmes linéaires à notre avantage.

2.3 Encodage des opérations sur les lignes

En revenant à l'exemple du §2.2, on constate que la résolution du système linéaire n'impliquait en réalité que l'application de trois opérations : la permutation, la mise à l'échelle des lignes et l'ajout de l'échelle d'une ligne à une autre. En fait, nous pouvons résoudre n'importe quel système linéaire de cette façon, il vaut donc la peine d'explorer ces opérations plus en détail.

2.3.1 Permutation

Notre première étape au §2.2 a consisté à échanger deux des lignes du système d'équations. Plus généralement, nous pourrions indexer les lignes d'une matrice en utilisant les nombres 1, . . . , M. Ensuite, une permutation de ces lignes peut être écrite comme une fonction σ telle que la liste σ(1), . . . , σ(m) couvre le même ensemble d'indices.

Si ek est la k-ième fonction de base standard, alors il est facile de voir que le produit e k A donne la k-ième ligne de la matrice A. Ainsi, nous pouvons « empiler » ou concaténer ces vecteurs ligne verticalement pour donner une matrice permutant les lignes selon σ :

$$P\sigma \equiv \begin{matrix} - & e_{\sigma(1)} & - \\ - & e_{\sigma(2)} & - \\ & \cdots & \\ - & e_{\sigma(m)} & - \end{matrix}$$

Autrement dit, le produit PσA est exactement la matrice A avec des lignes permutées selon σ.

Exemple 2.1 (Matrices de permutation). Supposons que nous souhaitions permuter des lignes d'une matrice dans R3 × 3 avec σ(1) = 2, σ(2) = 3 et σ(3) = 1. Selon notre formule, nous aurions

$$P\sigma = \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{matrix}$$

D'après l'exemple 2.1, nous pouvons voir que Pσ a des uns dans les positions de la forme (k, σ(k)) et des zéros ailleurs. La paire (k, σ(k)) représente l'énoncé « Nous aimerions que la ligne k de la matrice de sortie soit la ligne σ(k) de la matrice d'entrée ». Sur la base de cette description d'une matrice de permutation, il est facile de voir que l'inverse de Pσ est la transposée P puisque celaσéchange simplement les rôles des lignes et des colonnes - maintenant nous prenons la ligne σ(k) de l'entrée et la mettons dans ligne k de la sortie. Autrement dit, P σ Pσ = Im×m.

2.3.2 Mise à l'échelle des lignes

Supposons que nous écrivions une liste de constantes a1, . . . , am et cherche à mettre à l'échelle la k-ième ligne d'une matrice A par ak . Ceci est évidemment accompli en appliquant la matrice d'échelle Sa donnée par :

$$Sa \equiv \begin{matrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & & \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \cdots & & a_m \end{matrix}$$

En supposant que tous les ak satisfont ak = 0, il est facile d'inverser Sa en « réduisant » :

$$S_{un}^{-1} = S1/a \equiv \begin{matrix} 1/a_1 & 0 & 0 & \cdots & 1/a_2 & 0 & \cdot \\ 0 & & & & & & \\ \vdots & \vdots & & \ddots & & \vdots & \\ 0 & & 0 & \cdots & 1/a_m & & \end{matrix}$$

2.3.3 Élimination

Enfin, supposons que nous souhaitions mettre à l'échelle la ligne k par une constante c et ajouter le résultat à la ligne . Cette opération peut sembler moins naturelle que les deux précédentes mais est en réalité assez pratique : c'est la seule que nous

besoin de combiner des équations de différentes lignes du système linéaire ! Nous allons réaliser cette opération à l'aide d'une « matrice d'élimination » M telle que le produit MA applique cette opération à la matrice A.

Rappelons que le produit $e_k A$ sélectionne la k-ième ligne de A. Ensuite, la prémultiplication par e donne un matriceee k A, qui est nul sauf que la -ième ligne est égale à la k-ième de A.

Exemple 2.2 (Construction de la matrice d'élimination). Prendre

$$A = \begin{matrix} 1\,2\,3 \\ 4\,5\,6 \\ 7\,8\,9 \end{matrix}$$

Supposons que nous souhaitions isoler la troisième ligne de A $R_{3\times 3}$ et la déplacer vers la ligne deux. Comme discuté ci-dessus, cette opération est accomplie en écrivant :

$$e_{2e_3} A = \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \begin{matrix} 0\,0\,1 \end{matrix} \begin{matrix} 1\,2\,3\,4 \\ 5\,6 \\ 7\,8\,9 \end{matrix}$$

$$= \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \begin{matrix} 7\,8\,9 \end{matrix}$$

$$= \begin{matrix} 0\,0\,0 \\ 7\,8\,9 \\ 0\,0\,0 \end{matrix}$$

Bien sûr, nous avons multiplié de droite à gauche ci-dessus, mais nous aurions tout aussi bien pu regrouper le produit en $(e_{2e_3})A$.
La structure de ce produit est facile à voir :

$$e_{2e_3} = \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \begin{matrix} 0\,0\,1 = \end{matrix} \begin{matrix} 0\,0\,0\,0 \\ 0\,1 \\ 0\,0\,0 \end{matrix}$$

Nous avons réussi à isoler la ligne k et à la déplacer vers la ligne . Notre opération d'élimination originale voulait ajouter c fois la ligne k à la ligne A = k (In×n + cee ; que nous pouvons maintenant accomplir comme la somme A + cee $_k$)A.

Exemple 2.3 (Résolution d'un système). Nous pouvons maintenant encoder chacune de nos opérations de la section 2.2 en utilisant les matrices que nous avons construites ci-dessus :

1. Permutez les lignes pour déplacer la troisième équation vers la première ligne :

$$P = \begin{matrix} 0\,0\,1 \\ 1\,0\,0 \\ 0\,1\,0 \end{matrix}$$

2. Mettre à l'échelle la ligne 1 par -3 et ajouter le résultat à la ligne 3 : E1 = I3×3 − 3e3e 1

3. Mettre à l'échelle la ligne deux par 4 et ajouter le résultat à la ligne trois : E2 = I3×3 + 4e3e 2

4. Mettre à l'échelle la rangée trois par 1/3 : S = diag(1, 1, 1/3)

5. Mettez à l'échelle la rangée trois par 2 et ajoutez-la à la rangée un : E3 = I3×3 + 2e1e3

6. Ajoutez la ligne trois à la ligne deux : E4 = I3×3 +e2e 3

7. Mettre à l'échelle la ligne trois par -1 et ajouter le résultat à la ligne un : E5 = I3×3 −e1e 3

Ainsi, l'inverse de A dans la section 2.2 satisfait

$$A^{-1} = E5E4E3SE2E1P.$$

Assurez-vous de comprendre pourquoi ces matrices apparaissent dans l'ordre inverse !

2.4 Élimination gaussienne

La séquence d'étapes choisie dans la section 2.2 n'était en aucun cas unique : il existe de nombreux chemins différents qui peuvent conduire à la solution de Ax =b. Nos démarches, cependant, ont suivi la stratégie d'élimination gaussienne, un algorithme célèbre pour résoudre des systèmes linéaires d'équations.

Plus généralement, supposons que notre système ait la "forme" suivante :

$$[A|b] = \left[\begin{array}{ccccc|c} \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \end{array}\right]$$

L'algorithme procède par phases décrites ci-dessous.

2.4.1 Substitution avant

Considérez l'élément en haut à gauche de notre matrice :

$$[A|b] = \left[\begin{array}{ccccc|c} \textcircled{\times} & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \end{array}\right]$$

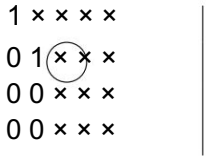
Nous appellerons cet élément notre premier pivot et supposerons qu'il est non nul ; s'il est égal à zéro, nous pouvons permuter les lignes pour que ce ne soit pas le cas. Nous appliquons d'abord une matrice de mise à l'échelle pour que le pivot soit égal à un :

$$\left[\begin{array}{ccccc|c} \textcircled{1} & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \end{array}\right]$$

Maintenant, nous utilisons la ligne contenant le pivot pour éliminer toutes les autres valeurs en dessous dans la même colonne :

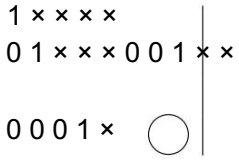
$$\left[\begin{array}{ccccccc|c} \textcircled{1} & \times & \times & \times & 0 & \times & \times & \\ \times & 0 & \times & \times & \times & 0 & \times & \\ \times & \times & & & & & & \end{array}\right]$$

Nous déplaçons maintenant notre pivot vers la ligne suivante et répétons une série d'opérations similaires :



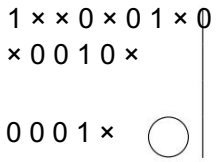
Remarquez qu'une belle chose se passe ici. Une fois le premier pivot éliminé de toutes les autres lignes, la première colonne est zéro sous la ligne 1. Cela signifie que nous pouvons ajouter en toute sécurité des multiples de la ligne deux aux lignes en dessous sans affecter les zéros de la colonne un.

Nous répétons ce processus jusqu'à ce que la matrice devienne triangulaire supérieure :

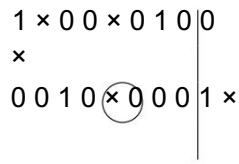


2.4.2 Substitution arrière

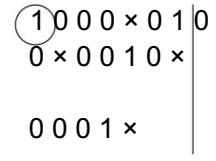
L'élimination des × restants du système est maintenant un processus simple. Maintenant, nous procédons dans l'ordre inverse des rangées et éliminons en arrière. Par exemple, après la première série d'étapes de rétrosubstitution, il nous reste la forme suivante :



De même, la seconde itération donne :



Après notre dernière étape d'élimination, il nous reste notre forme souhaitée :



Le membre de droite est maintenant la solution du système linéaire Ax =b.

2.4.3 Analyse de l'élimination gaussienne

Chaque opération de ligne dans l'élimination gaussienne - mise à l'échelle, élimination et permutation de deux lignes - prend évidemment un temps O (n) , car vous devez parcourir tous les n éléments d'une ligne (ou

deux) de A. Une fois que nous avons choisi un pivot, nous devons faire n substitutions avant ou arrière dans les au-dessous ou au-dessus de ce pivot, resp. ; cela signifie que le travail pour un seul pivot au total est $O(n^2$ rangées). Au total, nous choisissons un pivot par ligne, en ajoutant un facteur final de n. Ainsi, il est assez facile de voir que la gaussienne l'élimination tourne en $O(n^3)$ temps.

Une décision qui a lieu lors de l'élimination gaussienne dont nous n'avons pas discuté est le choix des pivots. Rappelons que nous pouvons permuter les lignes du système linéaire comme bon nous semble avant d'effectuer une rétro-substitution ou une substitution directe. Cette opération est nécessaire pour pouvoir traiter toutes les matrices A possibles. Par exemple, considérons ce qui se passerait si nous n'utilisions pas le pivotement sur les suivantes matrice:

$$U_n = \begin{matrix} (0) & 1 & 1 \\ & 0 & \end{matrix}$$

Notez que l'élément encerclé est exactement zéro, donc nous ne pouvons pas nous attendre à diviser la ligne un par n'importe quel nombre pour remplacer ce 0 par un 1. Cela ne signifie pas que le système n'est pas résoluble, cela signifie simplement que nous devons faire pivoter, accompli en échangeant le première et deuxième rangées, pour mettre un non-zéro dans cet emplacement.

Plus généralement, supposons que notre matrice ressemble à :

$$U_n = \begin{matrix} (\varepsilon) & 1 \\ dix & \end{matrix},$$

où $0 < \varepsilon$

1. Si nous ne pivotons pas, alors la première itération de l'élimination gaussienne donne :

$$\tilde{A} = \begin{matrix} (1) & 1/\varepsilon & 0 \\ & -1/\varepsilon & \end{matrix},$$

Nous avons transformé une matrice A qui ressemble presque à une matrice de permutation (en fait, un $A \approx -1$, un moyen très simple de résoudre le système !) en un système avec des valeurs potentiellement énormes $1/\varepsilon$.

Cet exemple montre qu'il existe des cas où l'on peut souhaiter pivoter même si cela n'est pas nécessaire à proprement parler. Puisque nous mettons à l'échelle par l'inverse de la valeur du pivot, il est clair que l'option la plus stable numériquement est d'avoir un grand pivot : les petits pivots ont de grands inverses, mettant à l'échelle les nombres à de grandes valeurs dans des régimes susceptibles de perdre en précision. Il existe deux stratégies de pivotement bien connues :

1. Le pivotement partiel parcourt la colonne actuelle et permute les lignes de la matrice de sorte que la plus grande valeur absolue apparaît sur la diagonale.

2. Le pivotement complet itère sur toute la matrice et permute les lignes et les colonnes pour obtenir la plus grande valeur possible sur la diagonale. Remarquez que la permutation des colonnes d'une matrice est une opération valide : elle correspond à changer l'étiquetage des variables dans le système, ou à post-multiplier A par une permutation.

Exemple 2.4 (Pivotement). Supposons qu'après la première itération de l'élimination gaussienne, il nous reste la matrice suivante :

$$\begin{matrix} 1 & 10 & -10 \\ 0 & 0{,}1 & 9 \\ 0 & 4 & 6.2 \end{matrix}$$

Si nous implémentons le pivotement partiel, nous ne regarderons que dans la deuxième colonne et échangerons les deuxième et troisième lignes ; notez que nous laissons le 10 dans la première ligne puisque celui-ci a déjà été visité par l'algorithme :

$$\begin{matrix} 1 & 10 & -10 \\ 0 & 4 & 6{,}2 \\ 0 & 0{,}1 & 9 \end{matrix}$$

Si nous implémentons le pivotement complet, nous déplacerons le 9 :

$$\begin{matrix} 1 & -10 & 10 \\ 0 & 9 & 0{,}1 \\ 0 & 6{,}2 & 4 \end{matrix}$$

De toute évidence, le pivotement complet donne les meilleurs chiffres possibles, mais le coût est une recherche plus coûteuse d'éléments volumineux dans la matrice.

2.5 Factorisation LU

Il arrive souvent que l'on souhaite résoudre une suite de problèmes Ax1 =b1, Ax2 =b2, ... Comme nous l'avons déjà discuté, les étapes de l'élimination gaussienne pour résoudre Ax = bk dépendent principalement de la structure de A plutôt que des valeurs d'un bk particulier . Puisque A est maintenu constant ici, nous souhaiterons peut-être "se souvenir" des étapes que nous avons suivies pour résoudre le système afin qu'à chaque fois qu'un newb nous soit présenté, nous n'ayons pas à repartir de zéro.

Consolidant cette suspicion que nous pouvons déplacer certains des $O(n^3)$ pour l'élimination gaussienne en temps de précalcul, rappelons le système triangulaire supérieur résultant après l'étape de substitution directe :

$$\left[\begin{array}{cccc|c} 1 & \times & \times & \times & \times \\ 0 & 1 & \times & \times & \times \\ 0 & 0 & 1 & \times & \times \\ 0 & 0 & 0 & 1 & \times \end{array}\right]$$

En fait, résoudre ce système par rétrosubstitution ne prend que $O(n^2)$ le temps ! Pourquoi? Substitution arrière dans ce cas est beaucoup plus facile grâce à la structure des zéros dans le système. Par exemple, dans la première série de rétrosubstitutions on obtient la matrice suivante :

$$\left[\begin{array}{cccccc|c} 1 & \times & \times & 0 & \times & 0 & \times & 0 & \times & 0 & 1 \\ 0 & \times & & 1 & & & & & & & \\ & & & 0 & & & & & & & \\ 0 & & 0 & & 0 & & 1 & \times & & & \end{array}\right]$$

Puisque nous savons que les valeurs (entourées) à gauche du pivot sont nulles par construction, nous n'avons pas besoin de les copier explicitement. Ainsi, cette étape n'a pris que O(n) temps plutôt que $O(n^2)$ pris par substitution directe.

Maintenant, notre prochain pivot fait une substitution similaire :

$$\left[\begin{array}{cccccc|c} 1 & \times & 0 & 0 & \times & & \\ 0 & & 1 & 0 & 0 & \times & \\ 0 & & 0 & 1 & 0 & \times & 0 & 0 & 1 & \times \\ 0 & & & & & & & & & \end{array}\right]$$

Encore une fois, les zéros des deux côtés du 1 n'ont pas besoin d'être copiés explicitement.

Ainsi, nous avons trouvé :

Observation. Alors que l'élimination gaussienne prend O(n ^{3}) temps, résoudre des systèmes triangulaires prend O(n^2) temps.

2.5.1 Construction de la factorisation

Rappelons du §2.3 que toutes les opérations d'élimination gaussienne peuvent être considérées comme pré-multipliant Ax = b par différentes matrices M pour obtenir un système plus simple (MA)x = Mb. Comme nous l'avons démontré dans l'exemple 2.3, de ce point de vue chaque étape de l'élimination gaussienne représente un système (Mk · · · M2M1A)x = Mk · · · M2M1 b . Bien sûr, stocker explicitement ces matrices Mk comme n × n objets est exagéré, mais garder cette interprétation à l'esprit d'un point de vue théorique simplifie beaucoup de nos calculs.

Après la phase de substitution vers l'avant de l'élimination gaussienne, nous nous retrouvons avec une matrice triangulaire supérieure, que nous pouvons appeler U Rn×n . Du point de vue de la multiplication matricielle, nous pouvons écrire:

$$M_k \cdots M_1 A = U,$$

ou équivalent,

$$\begin{aligned} A &= (M_k \cdots M_1)^{-1} U \\ &= (M_1^{-1} M_2^{-1} \cdots M_k^{-1}) U \\ &\equiv LU, \text{ si on fait la définition } L \equiv M_1^{-1} M_2^{-1} \cdots M_{k-1}^{-1}. \end{aligned}$$

Nous ne savons encore rien de la structure de L, mais nous savons que les systèmes de la forme Uy = d sont plus faciles à résoudre car U est triangulaire supérieur. Si L est également gentil, on pourrait résoudre Ax =b en deux étapes, en écrivant (LU)x =b, ou x = U−1L −1b : 1. Résoudre Ly =b pour y,

donnant y = L −1b.

2. Maintenant que nous avons y, résolvons Ux = y, donnant x = U−1y = U−1 (L −1b) = (LU) −1b = A −1b.
Nous savons déjà que cette étape ne prend que O(n ^{2}) temps.

Notre tâche restante est de nous assurer que L a une belle structure qui rendra la résolution de Ly = b plus facile que la résolution de Ax = b. Heureusement - et sans surprise - nous trouverons que L est triangulaire inférieur et peut donc être résolu en utilisant O(n ^{2}) substitution vers l'avant.

Pour voir cela, supposons pour l'instant que nous n'implémentons pas de pivotement. Ensuite, chacune de nos matrices Mk est soit une matrice d'échelle ou a la structure

$$M_k = I_{n\times n} + c e_e e_k,$$

où > k puisque nous n'avons effectué que la substitution directe. N'oubliez pas que cette matrice a un objectif spécifique : mettre à l'échelle la ligne k par c et ajouter le résultat à la ligne . Cette opération est évidemment facile à annuler : mettre à l'échelle la ligne k par c et soustraire le résultat de la ligne . On peut vérifier cela formellement :

$$\begin{aligned} (I_{n\times n} + c e_e e_k)(I_{n\times n} - c e_e e_k) &= I_{n\times n} + (-c e_e e_k + c e_e e_k) - c^2 e_e e_e e_k e_k \\ &= I_{n\times n} - c^2 e_e (e_k e_e) e_k \\ &= I_{n\times n} \text{ puisque } e_k \cdot e_k = 0, \text{ et } k = \end{aligned}$$

Ainsi, la matrice L est le produit de matrices d'échelle et de matrices de la forme M−1 = In×n + cee triangle$_k$ sont inférieur quand > k. Les matrices de mise à l'échelle sont diagonales et la matrice M est triangulaire inférieure. Vous montrerez dans l'exercice 2.1 que le produit des matrices triangulaires inférieures est triangulaire inférieur, montrant à son tour que L est triangulaire inférieur au besoin.

Nous avons montré que s'il est possible d'effectuer une élimination gaussienne de A sans utiliser de pivotement, on peut factoriser A = LU dans le produit de matrices triangulaires inférieures et supérieures. Les substitutions avant et arrière prennent chacune un temps $O(n^2)$, donc si cette factorisation peut être calculée à l'avance, la résolution linéaire peut être effectuée plus rapidement que la pleine $O(n^3)$ Élimination gaussienne. Vous montrerez dans Exercice 2.2 que se passe-t-il lorsque nous effectuons LU avec pivotement ; pas de changements majeurs sont nécessaires.

2.5.2 Mise en œuvre de LU

Une implémentation simple de l'élimination gaussienne pour résoudre Ax = b est assez simple à formuler. En particulier, comme nous l'avons vu précédemment, nous pouvons former la matrice augmentée (A |b) et appliquer les opérations de ligne une par une à ce bloc n × (n + 1) jusqu'à ce qu'il ressemble à (In×nĀ^1|b). Ce processus est cependant destructeur, c'est-à-dire qu'au final nous ne nous soucions que de la dernière colonne de la matrice augmentée et n'avons conservé aucune trace de notre chemin vers la solution. Un tel comportement n'est clairement pas acceptable pour la factorisation LU.

Examinons ce qui se passe lorsque nous multiplions deux matrices d'élimination :

$$(In\times n - cee_{k})(In\times n - cpepe_{k}) = In\times n - cee_{k} - cpepe_{k}$$

Comme dans notre construction de l'inverse d'une matrice d'élimination, le produit des deux termes ei s'annule puisque la base standard est orthogonale. Cette formule montre qu'après la mise à l'échelle du pivot à 1, le produit des matrices d'élimination utilisées pour remplacer ce pivot a la forme :

$$M = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \bigcirc & 0 \\ 0 & \times & 1 & 0 \\ 0 & \times & 0 & 1 \end{matrix} \quad 0 \\ 0 \\ ,$$

où les valeurs × sont les valeurs utilisées pour éliminer le reste de la colonne. La multiplication des matrices de cette forme ensemble montre que les éléments sous la diagonale de L proviennent simplement des coefficients utilisés pour accomplir la substitution.

Nous pouvons prendre une décision finale pour garder les éléments le long de la diagonale de L dans la factorisation LU égaux à 1. Cette décision est légitime, puisque nous pouvons toujours post-multiplier un L par une matrice d'échelle S prenant ces éléments à 1 et écrivez LU = (LS)(S −1U) sans affecter le modèle triangulaire de L ou U. Avec cette décision en place, nous pouvons compresser notre stockage de L et U en une seule matrice n × n dont le triangle supérieur est U et qui est égal à L sous la diagonale ; les éléments diagonaux manquants de L sont tous 1.

Nous sommes maintenant prêts à écrire un pseudocode pour la stratégie de factorisation LU la plus simple dans laquelle nous ne permutons pas les lignes ou les colonnes pour obtenir des pivots :

```
// Prend en entrée une matrice n - par - n A [i , j ]
// Modifie A en place pour obtenir la factorisation LU compacte décrite ci-dessus

pour pivot de 1 à n {
        valeurpivot = A [ pivot , pivot ] ; // Mauvaise hypothèse que c'est non nul !
```

```
        for removalRow from ( pivot +1) to n { // Élimine les valeurs sous le pivot
                // De combien mettre à l'échelle la ligne pivot pour éliminer la valeur dans le courant
                // rangée ; remarquez nous ne nous inquiétons pas de mettre à l'échelle la ligne de pivot à 1 ,
                // valeur pivot
                échelle = A [ éliminationRow , pivot ]/ pivotValue ;

                // Puisque nous / soustrayons / redimensionnons la ligne pivot de la ligne actuelle
                // lors de l'élimination gaussienne // stocké dans L , nous / ajoutons / il dans l'opération inverse

                A [eliminationRow, pivot] = échelle ;

                // Maintenant, comme dans l'élimination gaussienne, effectuez l'opération de ligne sur le reste
                // de la ligne : cela deviendra U
                pour éliminationCol de ( pivot +1) à n
                        A [ éliminationLigne , ColElimination ] -= A [ pivot , ColElimination ]* échelle ;
        }
}
```

2.6 Problèmes

Problème 2.1. Le produit des choses triangulaires inférieures est triangulaire inférieur; le produit des matrices pivot semble correct

Problème 2.2. Mettre en œuvre LU avec pivotement

Problème 2.3. LU non carré

chapitre 3

Concevoir et analyser linéaire Systèmes

Maintenant que nous avons quelques méthodes pour résoudre des systèmes linéaires d'équations, nous pouvons les utiliser pour résoudre une variété de problèmes. Dans ce chapitre, nous explorerons quelques-unes de ces applications et les techniques analytiques associées pour caractériser les types de solutions auxquelles nous pouvons nous attendre.

3.1 Solution des systèmes carrés

Au début du chapitre précédent, nous avons fait plusieurs hypothèses sur les types de systèmes linéaires que nous allions résoudre. Bien que cette restriction ne soit pas triviale, en fait, de nombreuses applications, sinon la plupart, des solveurs linéaires peuvent être posées en termes de systèmes linéaires carrés inversibles. Nous explorons ci-dessous quelques applications contrastées.

3.1.1 Régression

Nous commencerons par une application simple apparaissant dans l'analyse de données connue sous le nom de régression. Supposons que nous réalisions une expérience scientifique et souhaitions comprendre la structure de nos résultats expérimentaux. Une façon de modéliser une telle relation pourrait être d'écrire les variables indépendantes de l'expérience dans un vecteur $x \in \mathbb{R}^n$ et de considérer la variable dépendante comme une fonction $f(x) : \mathbb{R}^n \to \mathbb{R}$. Notre objectif est de prédire la sortie de f sans effectuer l'expérience complète.

Exemple 3.1 (Expérience biologique). Supposons que nous souhaitions mesurer l'effet des engrais, de la lumière du soleil et de l'eau sur la croissance des plantes. Nous pourrions faire un certain nombre d'expériences en appliquant différentes quantités d'engrais (en cm3), de lumière solaire (en watts) et d'eau (en ml) et en mesurant la hauteur de la plante après quelques jours. Nous pourrions modéliser nos observations sous la forme d'une fonction $f : \mathbb{R}^3 \to \mathbb{R}$ prenant en compte les trois paramètres que nous souhaitons tester et produisant la hauteur de la plante.

En régression paramétrique, on fait une hypothèse simplificatrice sur la structure de f . Par exemple, supposons que f est linéaire :

$$f(x) = a_1x_1 + a_2x_2 + \cdots + a_nx_n.$$

Ensuite, notre but devient plus concret : estimer les coefficients a_k .

Supposons que nous fassions une série d'expériences qui montrent la forme $x^{(k)} \to y^{(k)} \equiv f(x^{(k)})$. En vous connectant à notre $x \to y$ pour f , nous obtenons une série d'énoncés :

$$\begin{aligned} y^{(1)} &= f(x^{(1)}) = a_1 x_1^{(1)} + a_2 x_2^{(1)} + \cdots + a_n x_n^{(1)} \\ y^{(2)} &= f(x^{(2)}) = a_1 x_1^{(2)} + a_2 x_2^{(2)} + \cdots + a_n x_n^{(2)} \\ &\vdots \end{aligned}$$

Notez que contrairement à notre notation Ax =b, les inconnues ici sont les ak et non les variables x. Si on fait n observations, on peut écrire :

$$\begin{bmatrix} - & x^{(1)} & - \\ & x^{(2)} & - \\ & \vdots & \\ - & x^{(n)} & - \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

En d'autres termes, si nous effectuons n essais de notre expérience et les écrivons dans les colonnes d'une matrice $X \in \mathbb{R}^{n \times n}$ et écrivons les variables dépendantes dans un vecteur $y \in \mathbb{R}^n$, les coefficients a peuvent être récupérés en résolvant X a = y .

En fait, nous pouvons généraliser notre approche à d'autres formes non linéaires plus intéressantes pour la fonction f . Ce qui compte ici, c'est que f soit une combinaison linéaire de fonctions. En particulier, supposons que f(x) prend la forme suivante :

$$f(x) = a_1 f_1(x) + a_2 f_2(x) + \cdots + a_m f_m(x),$$

où $f_k : \mathbb{R}^n \to \mathbb{R}$ et on souhaite estimer les paramètres ak . Alors, par une dérivation parallèle $(x^{(k)}) \to y^{(k)}$ étant donné m observations de la forme on peut trouver les paramètres en résolvant :

$$\begin{bmatrix} f_1(x^{(1)}) & f_2(x^{(1)}) & \cdots & f_m(x^{(1)}) \\ f_1(x^{(2)}) & f_2(x^{(2)}) & \cdots & f_m(x^{(2)}) \\ \vdots & \vdots & \cdots & \vdots \\ f_1(x^{(m)}) & f_2(x^{(m)}) & \cdots & f_m(x^{(m)}) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Autrement dit, même si les f sont non linéaires, nous pouvons apprendre les poids ak en utilisant des techniques purement linéaires.

Exemple 3.2 (Régression linéaire). Le système Xa = y peut être récupéré à partir de la formulation générale en prenant $f_k(x) \equiv x_k$.

Exemple 3.3 (Régression polynomiale). Supposons que nous observons une fonction d'une seule variable f(x) et que nous souhaitions l'écrire sous la forme d'un polynôme de degré n

$$f(x) \equiv a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n .$$

Étant donné n paires $x^{(k)} \to y^{(k)}$, nous pouvons résoudre les paramètresa via le système

$$\begin{bmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \cdots & (x^{(1)})^n \\ 1 & x^{(2)} & (x^{(2)})^2 & \cdots & (x^{(2)})^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x^{(n)} & (x^{(n)})^2 & \cdots & (x^{(n)})^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} .$$

En d'autres termes, nous prenons $f_k(x) = x^k$ dans notre forme générale ci-dessus. Incidemment, la matrice du côté gauche de cette relation est connue sous le nom de matrice de Vandermonde, qui possède de nombreuses propriétés spéciales spécifiques à sa structure.

Exemple 3.4 (Oscillation). Supposons que nous souhaitions trouver a et φ pour une fonction $f(x) = a\cos(x + \varphi)$. Rappelez-vous de la trigonométrie que nous pouvons écrire $\cos(x + \varphi) = \cos x \cos \varphi - \sin x \sin \varphi$. Ainsi, étant donné deux points d'échantillonnage, nous pouvons utiliser la technique ci-dessus pour trouver $f(x) = a_1 \cos x + a_2 \sin x$, et en appliquant cette identité, nous pouvons écrire

$$2 \text{ un} = \text{un } 1 \quad + \text{un } \frac{2}{2}$$

$$a2\ \varphi = -\arctan \frac{}{a1}$$

Cette construction peut être étendue pour trouver $f(x) = \sum_k a_k \cos(x + \varphi_k)$, donnant une façon de motiver la transformée de Fourier discrète de f .

3.1.2 Moindres carrés

Les techniques du §3.1.1 fournissent des méthodes intéressantes pour trouver un f continu correspondant exactement à un ensemble de paires de données $x_k \to y_k$. Il y a deux inconvénients liés à cette approche :

- Il peut y avoir une erreur dans la mesure des valeurs x_k et y_k . Dans ce cas, une relation approchée $f(x_k) \approx y_k$ peut être acceptable voire préférable à une relation exacte $f(x_k) = y_k$.
- Remarquez que s'il y avait m fonctions f_k au total, alors nous avions besoin d'exactement m observations$x_k \to y_k$. Des observations supplémentaires devraient être jetées, ou nous devrions changer la forme de f .

Ces deux problèmes sont liés au problème plus large du sur-ajustement : l'ajustement d'une fonction avec n degrés de liberté à n points de données ne laisse aucune place à l'erreur de mesure.

Plus généralement, supposons que l'on veuille résoudre le système linéaire $Ax = b$ pour x. Si on note la ligne k de A comme r_k , alors notre système ressemble

$$\begin{array}{ccccc} b_1 & & -r_1 & - & x_1 \\ b_2 & & -r_2 & - & x_2 \\ \vdots & & \vdots & \cdots & \vdots \\ \text{milliards} & = & -r_n & - & x_n \end{array} \quad = \quad \begin{array}{c} r_1 \cdot x \\ r_2 \cdot x \\ \vdots \\ r_n \cdot x \end{array}$$

par définition de la multiplication matricielle.

Ainsi, chaque ligne du système correspond à une observation de la forme $m_k \cdot x = b_k$. C'est-à-dire qu'une autre façon d'interpréter le système linéaire $Ax = b$ consiste en n déclarations de la forme "Le produit scalaire de x avec r_k est b_k ".

De ce point de vue, un grand système $Ax = b$ avec $A \in \mathbb{R}^{m\times n}$ et $m > n$ code simplement plus de n de ces observations de produits scalaires. Cependant, lorsque nous faisons plus de n observations, elles peuvent être incompatibles ; comme expliqué au §2.1, les grands systèmes n'admettront probablement pas de solution. Dans notre montage « expérimental » expliqué plus haut, cette situation pourrait correspondre à des erreurs de mesure des couples $x_k \to y_k$.

Quand on ne peut pas résoudre $Ax = b$ exactement, on peut relâcher un peu le problème pour approximer $Ax \approx b$. En particulier, on peut demander que le résidu $b - Ax$ soit le plus petit possible en minimisant le

norme b − Ax. Remarquez que s'il existe une solution exacte au système linéaire, alors cette norme est minimisée à zéro, puisque dans ce cas nous avons b − Ax = b − b = 0. Minimiser b − Ax revient à minimiser b − Ax ^{2}, que nous avons étendu dans l'exemple 0.16 à :

$$\text{b − Hache } ^2 = \text{x A Axe − 2b Axe + b } ^2.1$$

Le gradient de cette expression par rapport à x doit être nul à son minimum, donnant le système suivant :

$$0 = 2A\ Ax − 2A\ b$$

Ou de façon équivalente : A Ax = A b.

Cette fameuse relation est digne d'un théorème :

Théorème 3.1 (Équations normales). Les minima du résidu b − Ax pour A Rm×n (sans restriction sur m ou n) satisfont AAx = A b.

Si au moins n lignes de A sont linéairement indépendantes, alors la matrice AA Rn×n est inversible. Dans ce cas, le résidu minimum se produit (uniquement) en (A A) −1A b, ou de manière équivalente, résoudre le problème des moindres carrés est aussi simple que résoudre le système linéaire carré A Ax = A b du théorème 3.1.
Ainsi, nous avons étendu notre ensemble de stratégies de résolution à A Rm×n avec m ≥ n en appliquant uniquement des techniques pour les matrices carrées.

Le cas sous-déterminé m < n est considérablement plus difficile à traiter. En particulier, on perd la possibilité d'une solution unique à Ax = b. Dans ce cas, nous devons faire une hypothèse supplémentaire sur x pour obtenir une solution unique, par exemple qu'elle a une petite norme ou qu'elle contient beaucoup de zéros. Chacune de ces hypothèses de régularisation conduit à une stratégie de solution différente ; nous en explorerons quelques-unes dans les exercices qui accompagnent ce chapitre.

3.1.3 Exemples supplémentaires

Une compétence importante est d'être capable d'identifier les systèmes linéaires "dans la nature". Ici, nous énumérons rapidement quelques exemples supplémentaires.

Alignement

Supposons que nous prenions deux photographies de la même scène à partir de positions différentes. Une tâche courante en vision par ordinateur et en graphisme consiste à les assembler. Pour ce faire, l'utilisateur (ou un système automatique) peut marquer un certain nombre de points xk ,yk R2 tels que xk dans l'image un corresponde à yk dans l'image deux. Bien sûr, des erreurs probables ont été commises lors de la mise en correspondance de ces points, nous souhaitons donc trouver une transformation stable entre les deux images en suréchantillonnant le nombre de paires nécessaires (x, y).

En supposant que notre appareil photo a un objectif standard, les projections de l'appareil photo sont linéaires, donc un raisonnable comme La supposition est qu'il existe un A R2×2 et un vecteur de translationb R2 tels que

$$yk ≈ Axk + b.$$

[1] Il peut être utile de revenir aux préliminaires du chapitre 0 à ce stade pour examen.

Nos variables inconnues ici sont A et b plutôt que xk et yk .

Dans ce cas, on peut trouver la transformation en résolvant :

$$\min_{\text{Un B}} \sum_{k=1}^{p} (Axk + b) - yk \quad ^2.$$

Cette expression est encore une fois une somme d'expressions linéaires au carré dans nos inconnues A et b, et par une dérivation similaire à notre discussion du problème des moindres carrés, elle peut être résolue linéairement.

Déconvolution

Souvent, nous prenons accidentellement des photos quelque peu floues. Bien qu'une photo complètement floue puisse être une cause perdue, s'il y a un flou localisé ou à petite échelle, nous pourrons peut-être récupérer une image plus nette en utilisant des techniques informatiques. Une stratégie simple est la déconvolution, expliquée ci-dessous.

Nous pouvons penser à une photographie comme un point , où p est le nombre de pixels ; bien sûr, si le dans la photo Rp est en couleur, nous pouvons avoir besoin de trois valeurs (RVB) par pixel, ce qui donne une technique similaire dans R3p Quoi qu'il en soit, de nombreux flous d'image simples sont linéaires, par exemple la convolution gaussienne ou les opérations faisant la moyenne des pixels avec leur voisins sur l'image. Dans le traitement d'image, ces opérations linéaires ont souvent d'autres propriétés spéciales comme l'invariance par décalage, mais pour nos besoins, nous pouvons considérer le flou comme un opérateur linéaire x → G x.

Supposons que nous prenions une photo floue x0 Rp . Ensuite, on pourrait essayer de récupérer l'image nette x Rp en résolvant le problème des moindres carrés

$$\min_{x \quad Rp} \quad x0 - G \quad x \quad ^2.$$

Autrement dit, nous demandons que lorsque vous brouillez x avec G, vous obteniez la photo observée x0. Bien sûr, de nombreuses images nettes peuvent donner le même résultat flou sous G, nous ajoutons donc souvent des termes supplémentaires à la minimisation ci-dessus en demandant que x0 ne varie pas énormément.

3.2 Propriétés spéciales des systèmes linéaires

Notre discussion sur l'élimination gaussienne et la factorisation LU a conduit à une méthode complètement générique pour résoudre des systèmes linéaires d'équations. Bien que cette stratégie fonctionne toujours, nous pouvons parfois gagner en vitesse ou en avantages numériques en examinant le système particulier que nous résolvons. Nous discutons ici de quelques exemples courants où en savoir plus sur le système linéaire peut simplifier les stratégies de solution.

3.2.1 Matrices définies positives et factorisation de Cholesky

Comme le montre le théorème 3.1, la résolution d'un problème des moindres carrés Ax ≈ b donne une solution x satisfaisant le système linéaire carré (A A)x = A b. Indépendamment de A, la matrice AA a quelques propriétés spéciales qui rendent ce système spécial.

Tout d'abord, il est facile de voir que AA est symétrique, puisque

$$(\text{UNE UNE}) = \text{UNE } (\text{UNE }) = \text{UNE UNE}.$$

Ici, nous avons simplement utilisé les identités (AB) = BA et (A) = A. Nous pouvons exprimer cette symétrie indexée en écrivant (A A)ij = (A A)ji pour tous les indices i, j. Cette propriété implique qu'il suffit de stocker uniquement les valeurs de AA sur ou au-dessus de la diagonale, puisque le reste des éléments peut être obtenu par symétrie.

De plus, AA est une matrice semi-définie positive, telle que définie ci-dessous :

Définition 3.1 (Positive (Semi-)Définie). Une matrice B Rn×n est semi-définie positive si pour tout x Rn x Bx , ≥ 0. B est définie positive si x Bx > 0 si x =0.

Il est facile de montrer que AA est semi-défini positif, puisque :

$$x\,A\,Ax = (Ax)\,(Ax) = (Ax)\cdot(Ax) = Ax \qquad {}_2^2 \geq 0.$$

En effet, si les colonnes de A sont linéairement indépendantes, alors AA est définie positive.

Plus généralement, supposons que l'on veuille résoudre un système défini positif symétrique Cx = d. Comme nous l'avons déjà exploré, nous pourrions factoriser en LU la matrice C, mais en fait nous pouvons faire un peu mieux. On écrit C Rn×n sous la forme d'une matrice bloc :

$$C = \begin{matrix} c11 & v^{\top} \\ C^{\tilde{}} & \end{matrix}$$

où v Rn−1 et C˜ R(n−1)×(n−1) . Grâce à la structure particulière de C, nous pouvons faire la remarque suivante :

$$Ce1 = 1\,0\cdots 0 \text{ et } 1 \qquad \begin{matrix} c11 & v^{\top} \\ C^{\tilde{}} & \end{matrix} \qquad \begin{matrix} 1 \\ 0 \\ \vdots \\ 0 \end{matrix}$$

$$= 1\,0\cdots 0 \qquad {}^{c11}_{v}$$

$$= c11$$

$$> 0 \text{ puisque C est définie positive et e1 } \neq 0.$$

Cela montre que, en ignorant les problèmes numériques, nous n'avons pas besoin d'utiliser le pivotement pour garantir que c11 = 0 pour la première étape d'élimination gaussienne.

En continuant avec l'élimination gaussienne, nous pouvons appliquer une matrice de substitution directe E, qui a génériquement la forme

$$E = \begin{matrix} 1/\sqrt{c11} & 0 \\ r & je(n-1)\times(n-1) \end{matrix}.$$

Ici, le vecteurr Rn−1 contient les multiples de la ligne 1 pour annuler le reste de la première colonne de C. Nous mettons également à l'échelle la ligne 1 de 1/ √ c11 pour des raisons qui apparaîtront bientôt !

De par sa conception, après substitution vers l'avant, nous connaissons le produit

$$CE = \begin{matrix} \sqrt{c11} & v^{\top}/\sqrt{c11} \\ 0 & D \end{matrix}$$

pour un certain D R(n−1)×(n−1) .

C'est ici que l'on s'écarte de l'élimination gaussienne : plutôt que de passer à la deuxième ligne, on peut post-multiplier par E pour obtenir un produit ECE :

$$\begin{aligned} CEE &= (CE)E \sqrt{} \\ &= \begin{matrix} c11 & v / \sqrt{c11} \\ 0 & D \end{matrix} \quad \begin{matrix} & \\ 1/\sqrt{c11} & r \\ 0 & je(n-1)\times(n-1) \end{matrix} \\ &= \begin{matrix} dix \\ 0 & D\tilde{} \end{matrix} \end{aligned}$$

Autrement dit, nous avons éliminé la première ligne et la première colonne de C ! De plus, il est facile de vérifier que la matrice D˜ est également définie positive.

Nous pouvons répéter ce processus pour éliminer toutes les lignes et colonnes de C de manière symétrique. Avis que nous avons utilisé à la fois la symétrie et la définition positive pour dériver la factorisation, puisque

- la symétrie nous a permis d'appliquer le même E aux deux côtés, et
- la définition positive garantit que c11 > 0, impliquant donc que $\sqrt{c11}$ existe.

En fin de compte, similaire à la factorisation LU, nous obtenons maintenant une factorisation C = LL pour une matrice triangulaire inférieure L. Ceci est connu sous le nom de factorisation de Cholesky de C. Si prendre les racines carrées le long de la diagonale pose des problèmes numériques, une factorisation LDL associée , où D est une matrice diagonale, évite ce problème et est simple à déduire de la discussion ci-dessus.

La factorisation de Cholesky est importante pour plusieurs raisons. Plus important encore, il faut la moitié de la mémoire pour stocker L que la factorisation LU de C ou même C lui-même, puisque les éléments au-dessus de la diagonale sont nuls, et comme dans LU, résoudre Cx = d est aussi simple que la substitution avant et arrière . Vous explorerez d'autres propriétés de la factorisation dans les exercices.

Au final, le code pour la factorisation de Cholesky peut être très succinct. Pour dériver un particulier com sous forme de pacte, supposons que nous choisissions une ligne arbitraire k et écrivions L sous forme de bloc isolant cette ligne :

$$L = \begin{matrix} L11 & 0 & 0 \\ k & & \\ L31 & k & L33 \end{matrix} \quad \begin{matrix} kk & 0 \\ & \\ & \end{matrix}$$

Ici, L11 et L33 sont toutes deux des matrices carrées triangulaires inférieures. Alors, la réalisation d'un produit donne :

$$\begin{aligned} LL = & \begin{matrix} L11 & 0 & 0 \\ k & & \\ L31 & k & L33 \end{matrix} \quad \begin{matrix} kk & 0 \\ & \\ & \end{matrix} \quad \begin{matrix} L_{11} & kL \\ 0 & k \\ 0 & 0 \end{matrix} \quad \begin{matrix} kk & (\_31 \\ & k) \\ L & 33 \end{matrix} \\ & \begin{matrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{matrix} \\ = & \begin{matrix} L & 11 \\ kk & + k \times \end{matrix} \quad \begin{matrix} & \\ 2 & kk \end{matrix} \quad \begin{matrix} \times \\ \times \\ \times \end{matrix} \end{aligned}$$

Nous laissons de côté les valeurs du produit qui ne sont pas nécessaires à notre dérivation.

Au final, on sait qu'on peut écrire C = LL. L'élément central du produit montre :

$$_{kk} = ckk - k \quad \begin{matrix} 2 \\ 2 \end{matrix}$$

où k Rk−1 contient les éléments de la k-ième ligne de L à gauche de la diagonale. En outre, l'élément central gauche du produit montre

$$L11k = ck$$

où ck contient les éléments de C dans la même position ask . Comme L11 est triangulaire inférieur, ce système peut être résolu par substitution vers l'avant !

Notez que notre discussion ci-dessus donne un algorithme pour calculer la factorisation de Cholesky de haut en bas, puisque L11 sera déjà calculé au moment où nous atteignons la ligne k. Nous fournissons pseudocode ci-dessous, adapté du CITE :

```
// Prend en entrée une matrice n - par - n A [i , j ]
// Modifie A en place pour obtenir la factorisation de Cholesky dans son triangle inférieur

pour k de 1 à n {
        // Back - remplace pour trouver l_k
        for i from 1 to k -1 { // élément i de l_k
                somme = 0 ;
                pour j de 1 à i -1
                        somme += UNE [je , j ]* UNE [k , j ] ;
                Un [k , je ] = ( UNE [k   , je ] - somme )/ A [je , je ];
        }

        // Appliquer la formule pour l_kk
        normeCarré = 0
        pour j de 1 à i -1
                normSquared += A [k , j ]^2;
        Un [k , k ] = sqrt ( UNE [k       , k ] - normeCarré );
}
```

Comme pour la factorisation LU, cet algorithme fonctionne clairement en $O(n^3)$ temps.

3.2.2 Faiblesse

De nombreux systèmes linéaires d'équations bénéficient naturellement de propriétés de parcimonie, ce qui signifie que la plupart des les entrées de A dans le système Ax = b sont exactement nulles. La rareté peut refléter une structure particulière dans un problème donné, y compris les cas d'utilisation suivants :

- En traitement d'images, de nombreux systèmes de retouche et de compréhension de photos expriment des relations entre les valeurs des pixels et celles de leurs voisins sur la grille de l'image. Une image peut être un point dans Rp pour p pixels, mais en résolvant Ax =b pour une nouvelle image de taille p, A Rp×p peut avoir seulement O(p) plutôt que $O(p^2)$ non nuls puisque chaque ligne n'implique qu'un seul pixel et ses voisins haut/bas/gauche/droite.

- En apprentissage automatique, un modèle graphique utilise une structure de graphe G ≡ (V, E) pour exprimer des distributions de probabilité sur plusieurs variables. Chaque variable est représentée par un nœud v V de le graphe, et l'arête e E représente une dépendance probabiliste. Les systèmes linéaires apparaissant dans ce contexte a souvent une ligne par sommet v V avec des valeurs non nulles uniquement dans les colonnes impliquant v et ses voisins.

- En géométrie computationnelle, les formes sont souvent exprimées à l'aide de collections de triangles liés ensemble dans un maillage. Les équations pour le lissage de surface et d'autres tâches relient à nouveau les positions et d'autres valeurs à un sommet donné avec celles de leurs voisins dans le maillage.

Exemple 3.5 (Paramétrage harmonique). Supposons que nous souhaitions utiliser une image pour texturer un maillage triangulaire. Un maillage peut être représenté comme une collection de sommets V R3 reliés entre eux par des arêtes E V × V pour former des triangles. Étant donné que les sommets de la géométrie sont dans R3 , nous devons trouver un moyen de les mapper sur le plan de l'image pour stocker la texture sous forme d'image. Ainsi, nous devons attribuer des coordonnées de texture t(v) R2 sur le plan image à chaque v V. Voir la figure NUMBER pour une illustration.

Une stratégie pour créer cette carte implique une seule résolution linéaire. Supposons que le maillage a une topologie de disque, c'est-à-dire qu'il peut être mappé à l'intérieur d'un cercle dans le plan. Pour chaque sommet vb sur la frontière du maillage, nous pouvons spécifier la position de vb en le plaçant sur un cercle. A l'intérieur, on peut demander que la position de la texture map soit la moyenne de ses positions voisines :

$$1\ t(v) = \left|\frac{}{n(v)}\right| \sum_{w \quad n(v)} t(w)$$

Ici, n(v) V est l'ensemble des voisins de v V sur le maillage. Ainsi, chaque v V est associé à une équation linéaire, soit en le fixant sur le bord, soit en demandant que sa position soit égale à la moyenne de ses positions voisines. Ce |V| × |V| système d'équations conduit à une stratégie de paramétrisation stable connue sous le nom de paramétrisation harmonique ; la matrice du système n'a que des O(|V|) non nuls dans les cases correspondant aux sommets et à leurs voisins.

Bien sûr, si A Rn×n est parcimonieux au point qu'il contient O(n) plutôt que O(n valeurs, il n'y a aucune raison^{2}) différent de zéro de stocker A comme une matrice n × n. Au lieu de cela, les techniques de stockage de matrice creuse ne stockent que les O(n) non nuls dans une structure de données plus raisonnable, par exemple une liste de triplets ligne/colonne/valeur Le choix d'une structure de données matricielle implique de considérer les opérations probables qui se produiront sur la matrice, y compris éventuellement la multiplication, l'itération sur des non nuls , ou en itérant sur des lignes ou des colonnes individuelles.

Malheureusement, il est facile de voir que la factorisation LU d'un A clairsemé peut ne pas aboutir à des matrices L et U clairsemées ; cette perte de structure limite considérablement l'applicabilité de l'utilisation de ces méthodes pour résoudre Ax = b lorsque A est grand mais clairsemé. Heureusement, il existe de nombreux solveurs clairsemés directs adaptant LU à des matrices clairsemées qui peuvent produire une factorisation de type LU sans induire beaucoup de remplissage ou des valeurs non nulles supplémentaires ; la discussion de ces techniques sort du cadre de ce texte. Alternativement, des techniques itératives ont été utilisées pour obtenir des solutions approximatives aux systèmes linéaires; nous reporterons la discussion de ces méthodes aux chapitres suivants.

Certaines matrices sont non seulement creuses mais aussi structurées. Par exemple, un système tridiagonal de équations linéaires a le modèle suivant de valeurs non nulles :

```
× ×
× × ×
    × × ×
        × × ×
            × ×
```

Dans les exercices qui suivent ce chapitre, vous dériverez une version spéciale de l'élimination gaussienne pour traiter cette structure en bandes simple.

Dans d'autres cas, les matrices peuvent ne pas être creuses mais admettre une représentation clairsemée. Pour l'examen ple, considérons la matrice cyclique :

abcddabccdabbcda

Bien entendu, cette matrice peut être stockée en utilisant uniquement les valeurs a, b, c, d. Les techniques spécialisées pour cette classe et d'autres classes de matrices sont bien étudiées et souvent plus efficaces que l'élimination gaussienne générique.

3.3 Analyse de sensibilité

Comme nous l'avons vu, il est important d'examiner la matrice d'un système linéaire pour savoir si elle a des propriétés spéciales qui peuvent simplifier le processus de résolution. La parcimonie, la définition positive, la symétrie, etc., peuvent fournir des indices sur le bon solveur à utiliser pour un problème particulier.

Même si une stratégie de solution donnée peut fonctionner en théorie, il est tout aussi important de comprendre dans quelle mesure nous pouvons faire confiance à la réponse à un système linéaire donnée par un solveur particulier. Par exemple, en raison de l'arrondi et d'autres effets discrets, il se peut qu'une implémentation de l'élimination gaussienne pour résoudre Ax = b produise une solution x0 telle que 0 < Ax0 −b 1 ; en d'autres termes, x0 ne résout le système qu'approximativement.

Une façon de comprendre la probabilité de ces effets d'approximation consiste à effectuer une analyse de sensibilité. Dans cette approche, on se demande ce qui arriverait à x si au lieu de résoudre Ax = b on résolvait en réalité un système d'équations perturbé (A + δA)x = b + δb. Il existe deux manières de visualiser les conclusions de ce type d'analyse :

1. Nous faisons probablement des erreurs en représentant A et b grâce à l'arrondi et à d'autres effets. Cette analyse montre alors la meilleure précision possible à laquelle on peut s'attendre pour x compte tenu des erreurs commises représentant le problème.

2. Si notre solveur génère une approximation x0 de la solution de Ax =b, c'est une solution exacte du système Ax0 = b0 si nous définissons b0 ≡ Ax0 (assurez-vous de bien comprendre pourquoi cette phrase n'est pas une tautologie !). Comprendre comment les changements dans x0 affectent les changements dans b0 montre à quel point le système est sensible aux réponses légèrement incorrectes.

Notez que notre discussion ici est similaire et en fait motivée par nos définitions de l'erreur avant et arrière dans les chapitres précédents.

3.3.1 Normes matricielles et vectorielles

Avant de pouvoir discuter de la sensibilité d'un système linéaire, nous devons être quelque peu prudents pour définir ce que cela signifie pour un changement δx d'être "petit". Généralement, on souhaite mesurer la longueur, ou norme, d'un vecteur x. Nous avons déjà rencontré la binorme d'un vecteur :

$$2x2 \equiv x1 \_ \_ \_ \quad + x \; \frac{2}{2} \; 2 + \cdots + x \; n$$

pour x Rn . Cette norme est populaire grâce à son lien avec la géométrie euclidienne, mais ce n'est en aucun cas la seule norme sur Rn . Plus généralement, nous définissons une norme comme suit :

Définition 3.2 (Norme vectorielle). Une norme vectorielle est une fonction · : Rn → [0, ∞) satisfaisant les conditions suivantes conditions :

- x = 0 si et seulement si x =0.

• cx = |c|x pour tout scalaire c R et vecteur x Rn .

• x +y ≤ x + y pour tout x,y Rn .

Alors que nous utilisons les deux indices · 2 pour désigner la norme à deux d'un vecteur, sauf indication contraire, nous utiliserons la notation x pour désigner la norme à deux de x. Outre cette norme, il existe de nombreux autres exemples : • La p-norme xp, pour p

≥ 1, donnée par :

$$xp \equiv (|x1|\, p + |x2|\, p + \cdots + |xn|\, p)^{1/p}$$

La norme 1 ou « taxicab », donnée par

$$x1 \equiv \sum_{k=1}^{n} |xk\,|.$$

Cette norme reçoit son surnom parce qu'elle représente la distance parcourue par un taxi entre deux points dans une ville où les routes ne sont que nord/sud et est/ouest.

• La ∞-norme x∞ donnée par :

$$x\infty \equiv \max(|x1|, |x2|, \cdots, |xn|).$$

Dans un certain sens, de nombreuses normes sur Rn sont les mêmes. En particulier, supposons que l'on dise que deux normes sont équivalentes lorsqu'elles vérifient la propriété

suivante : Définition 3.3 (Normes équivalentes). Deux normes · et · sont équivalentes s'il existe des constantes clow et chigh telles que

$$clowx \le x \le chighx$$

pour tout x Rn·

Cette condition garantit que jusqu'à certains facteurs constants, toutes les normes s'accordent sur les vecteurs sont "petits" et "grands". En fait, nous allons énoncer sans démonstration un célèbre théorème d'analyse : Théorème

3.2 (Équivalence des normes sur Rn). Toutes les normes sur Rn sont équivalentes.

Ce résultat quelque peu surprenant implique que toutes les normes vectorielles ont le même comportement approximatif, mais le choix d'une norme pour analyser ou énoncer un problème particulier peut faire une énorme différence.
Par exemple, sur R3 , la norme ∞ considère que le vecteur (1000, 1000, 1000) a la même norme que (1000, 0, 0) alors que la norme 2 est certainement affectée par les valeurs non nulles supplémentaires.

Comme on perturbe non seulement les vecteurs mais aussi les matrices, il faut aussi pouvoir prendre la norme d'une matrice. Bien entendu, la définition de base d'une norme ne change pas sur Rn×m. Pour cette raison, nous pouvons "dérouler" n'importe quelle matrice dans Rm × n en un vecteur dans Rnm pour adopter n'importe quelle norme vectorielle aux matrices. L'une de ces normes est la norme de Frobenius, donnée par

$$AFro \equiv \sqrt{\sum_{je,j} un\ ij.^{2}}$$

De telles adaptations des normes vectorielles ne sont cependant pas toujours très significatives. En particulier, la priorité pour comprendre la structure d'une matrice A est souvent son action sur les vecteurs, c'est-à-dire les résultats probables lorsque A est multiplié par un x arbitraire. Avec cette motivation, on peut définir la norme induite par une norme vectorielle comme suit :

Définition 3.4 (Norme induite). La norme sur Rm×n induite par une norme · sur Rn est donnée par

$$A \equiv \max\{Ax : x = 1\}.$$

Autrement dit, la norme induite est la longueur maximale de l'image d'un vecteur unitaire multipliée par A.

Puisque les normes vectorielles satisfont cx = |c|x, il est facile de voir que cette définition équivaut à exiger

$$A \equiv \max_{x \quad Rn\{0\}} \frac{\text{Hache}}{X}.$$

De ce point de vue, la norme de A induite par · est le plus grand rapport réalisable de la norme de Ax par rapport à celle de l'entrée x.

Cette définition générale rend quelque peu difficile le calcul de la norme A étant donné une matrice A et un choix de · . Heureusement, les normes matricielles induites par de nombreuses normes vectorielles populaires peuvent être simplifiées. Nous énonçons certaines de ces expressions sans preuve :

- La norme un induite de A est la somme maximale de n'importe quelle colonne de A :

$$A1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |aij|$$

- La norme ∞ induite de A est la somme maximale de n'importe quelle ligne de A :

$$A\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |aij|$$

- La double norme induite, ou norme spectrale, de A Rn×n est la racine carrée de la plus grande valeur propre de A A. Autrement dit,

$$^{UN}{}_{2}^{2} = \max\{\lambda : \text{il existe } x \quad R \quad {}^{n} \text{ avec } A\ Ax = \lambda x\}$$

Au moins les deux premières normes sont relativement faciles à calculer ; nous reviendrons sur la troisième en discutant des problèmes de valeurs propres.

3.3.2 Numéros d'état

Maintenant que nous disposons d'outils pour mesurer l'action d'une matrice, nous pouvons définir le nombre conditionnel d'un système linéaire en adaptant notre définition générique des nombres conditionnels du chapitre 1. Nous suivons le développement présenté dans CITE.

Supposons qu'on perturbe δA d'une matrice A et une perturbation δb correspondante. Pour chaque ε ≥ 0, en ignorant les aspects techniques de l'inversibilité, nous pouvons écrire un vecteur x(ε) comme solution à

$$(A + \varepsilon \cdot \delta A)x(\varepsilon) = b + \varepsilon \cdot \delta b.$$

Si nous différencions les deux côtés par rapport à ε et appliquons la règle du produit, nous obtenons le résultat suivant :

$$\delta A \cdot x + (A + \varepsilon \cdot \delta A)\frac{dx}{d\varepsilon} = \delta b$$

En particulier, lorsque ε = 0 on trouve

$$\delta A \cdot x(0) + A = \delta b \frac{dx}{d\varepsilon}_{\varepsilon=0}$$

ou équivalent,

$$\frac{dx}{d\varepsilon}_{\varepsilon=0} = A^{-1}(\delta b - \delta A \cdot x(0)).$$

En utilisant le développement de Taylor, on peut écrire

$$x(\varepsilon) = x + \varepsilon x(0) + O(\varepsilon^2),$$

où l'on définit $x(0) = \text{système } \frac{dx}{d\varepsilon}_{\varepsilon=0}$. Ainsi, nous pouvons étendre l'erreur relative commise en résolvant le perturbé :

$$\frac{x(\varepsilon) - x(0)}{x(0)} = \frac{\varepsilon x(0) + O(\varepsilon x(0)^2)}{} \text{ par le développement de Taylor}$$

$$= \frac{\varepsilon A^{-1}(\delta b - \delta A \cdot x(0)) + O(\varepsilon x(0)^2)}{} \text{ par la dérivée que nous avons calculée}$$

$$\leq \frac{|\varepsilon|}{x(0)}(A^{-1}\delta b + A^{-1}\delta A \cdot x(0))) + O(\varepsilon^2)$$

par l'inégalité triangulaire $A + B \leq A + B$

$$\leq |\varepsilon| A^{-1} \frac{\delta b}{x(0)} + \delta A + O(\varepsilon^2) \text{ par l'identité } AB \leq AB$$

$$= |\varepsilon| A^{-1} \frac{\delta b}{\text{Hache}(0)} + \frac{\delta A}{} + O(\varepsilon^2)$$

$$\leq |\varepsilon| A^{-1} \frac{\delta b}{\text{Hache}(0)} + \frac{\delta A}{} + O(\varepsilon^2) \text{ puisque } Ax(0) \leq Ax(0)$$

$$= |\varepsilon| A^{-1} \frac{\delta b}{b} + \frac{\delta A}{} + O(\varepsilon^2) \text{ puisque } Ax(0) \leq Ax(0)$$

puisque par définition, $Ax(0) = b$

Ici, nous avons appliqué certaines propriétés de la norme matricielle qui découlent des propriétés correspondantes pour les vecteurs. Remarquons que la somme $D \equiv \delta b/b + \delta A/A$ code les perturbations relatives de A et b. De ce point de vue, au premier ordre nous avons borné l'erreur relative de perturbation du système par ε en utilisant un facteur $\kappa \equiv AA^{-1}$:

$$\frac{x(\varepsilon) - x(0)}{x(0)} \leq \varepsilon \cdot D \cdot \kappa + O(\varepsilon^2)$$

De cette façon, la quantité κ borne le conditionnement des systèmes linéaires impliquant A. Pour cette raison, nous faisons la définition suivante :

Définition 3.5 (Matrix condition number). Le nombre conditionnel de A Rn×n pour une norme matricielle donnée · est

$$\text{cond } A \equiv AA^{-1}.$$

Si A n'est pas inversible, on prend cond A ≡ ∞.

Il est facile de voir que cond A ≥ 1 pour tout A, que la mise à l'échelle de A n'a aucun effet sur son numéro de condition et que le numéro de condition de la matrice d'identité est 1. Ces propriétés contrastent avec le déterminant, qui peut être mis à l'échelle vers le haut ou vers le bas. que vous mettez à l'échelle A.

Si · est induit par une norme vectorielle et A est inversible, alors on a

$$\begin{aligned} A^{-1} &= \max_{x=0} \frac{A^{-1}x}{x} && \text{par définition} \\ &= \max_{y=0} \frac{y}{Ay} && \text{en substituant } y = Ax \\ &= \min_{y=0} \frac{Ay}{y}^{-1} && \text{en prenant la réciproque} \end{aligned}$$

Dans ce cas, le nombre conditionnel de A est donné par :

$$\text{cond } A = \max_{x=0} \frac{Ax}{x} \min_{y=0} \frac{Ay}{y}^{-1}$$

En d'autres termes, cond A mesure l'étirement maximum au minimum possible d'un vecteur x sous A.

Plus généralement, une propriété de stabilité souhaitable d'un système Ax = b est que si A orb est perturbé, la solution x ne change pas considérablement. Notre motivation pour cond A montre que lorsque le nombre de conditions est petit, le changement de x est petit par rapport au changement de A orb, comme illustré dans la figure NUMBER. Sinon, une petite modification des paramètres du système linéaire peut entraîner de grandes déviations en x ; cette instabilité peut amener les solveurs linéaires à faire de grosses erreurs dans x en raison de l'arrondi et d'autres approximations pendant le processus de résolution. peut être aussi difficile que

La norme A^{-1} de calculer l'inverse complet A bornant le nombre de condition $^{-1}$. Une façon de baisser est d'appliquer l'identité $A^{-1}x \le A^{-1}x$. Ainsi, pour tout $x = 0 \ge A^{-1}x/x$. Ainsi, $^{-1}$ on peut écrire A

$$\text{cond } A = AA^{-1} \ge \frac{AA^{-1}x}{x}.$$

Ainsi, nous pouvons borner le nombre conditionnel en résolvant $A^{-1}x$ pour certains vecteurs x ; bien sûr, la nécessité d'un solveur linéaire pour trouver $A^{-1}x$ crée une dépendance circulaire sur le nombre conditionnel pour évaluer la qualité de l'estimation ! Lorsque · est induit par la norme à deux, nous fournirons dans les prochains chapitres des estimations plus fiables.

3.4 Problèmes

Quelque chose comme:

- Régression du noyau comme exemple du §3.1.1
- Solution de moindre norme à Ax =b, la matrice des moindres carrés est inversible sinon
- Versions variationnelles de la régularisation de Tikhonov/régression "ridge" (pas l'approche habituelle à cela, mais peu importe); complétant l'histoire sous-déterminée de cette façon
- 1 • L approches de régularisation pour le contraste - dessinez une image de pourquoi s'attendre à la rareté, dessinez cercles unitaires, montrer que la norme p n'est pas une norme pour p < 1, prendre la limite lorsque p → 0
- Mini-Riesz - dériver la matrice pour le produit interne, à utiliser pour montrer comment faire pivoter l'espace
- résolution tridiagonale
- propriétés du numéro de condition

Chapitre 4

Espaces de colonne et QR

Une manière d'interpréter le problème linéaire Ax = b pour x consiste à écrire b comme une combinaison linéaire des colonnes de A avec des poids donnés en x. Cette perspective ne change pas lorsque nous permettons à A Rm×n d'être non carré, mais la solution peut ne pas exister ou être unique selon la structure de l'espace des colonnes. Pour ces raisons, certaines techniques de factorisation des matrices et d'analyse des systèmes linéaires recherchent des représentations plus simples de l'espace des colonnes pour lever l'ambiguïté de la solvabilité et s'étendre plus explicitement que les factorisations basées sur les lignes comme LU.

4.1 La structure des équations normales

Comme nous l'avons montré, une condition nécessaire et suffisante pour que x soit une solution du problème des moindres carrés Ax ≈ b est que x satisfasse aux équations normales (A A)x = A b. Ce théorème suggère que la résolution des moindres carrés est une extension assez simple des techniques linéaires. Des méthodes telles que la factorisation de Cholesky montrent également que la structure particulière des problèmes des moindres carrés peut être utilisée à l'avantage du solveur.

Il existe cependant un gros problème limitant l'utilisation de cette approche. Pour l'instant, supposons que A soit carré ; alors on peut écrire :

$$\begin{aligned} \text{cond } AA &= AA\,(AA\ AA)^{-1} \\ &\approx AA\ AA^{-1}\,(A\ )^{-1} \quad \text{selon le choix de } \cdot \\ &= \text{UN}^{\,2}\ {}_{\text{UN}}{}^{-1\,2} \\ &= (\text{cond } A)^2 \end{aligned}$$

C'est-à-dire que le nombre conditionnel de AA est approximativement le carré du nombre conditionnel de A! Ainsi, alors que les stratégies linéaires génériques peuvent fonctionner sur AA lorsque le problème des moindres carrés est "facile", lorsque les colonnes de A sont presque linéairement dépendantes, ces stratégies sont susceptibles de générer une erreur considérable car elles ne traitent pas A directement.

Intuitivement, l'une des principales raisons pour lesquelles cond A peut être grand est que les colonnes de A peuvent sembler « similaires ». Considérez chaque colonne de A comme un vecteur dans Rm. Si deux colonnes ai et aj satisfont ai ≈ alors la longueur résiduelle aj , des moindres carrés b − Ax ne souffrirait probablement pas beaucoup si nous remplaçons des multiples de ai par des multiples de aj ou vice versa. Cette large gamme de solutions presque - mais pas complètement - équivalentes donne un mauvais conditionnement. Alors que le vecteur résultant x est instable, cependant, le produit

Ax reste pratiquement inchangé, du fait de la conception de notre substitution. Par conséquent, si nous souhaitons résoudre Ax ≈b simplement pour écrire b dans l'espace des colonnes de A, l'une ou l'autre solution suffirait.

Pour résoudre ces problèmes mal conditionnés, nous utiliserons une stratégie alternative avec une plus grande attention à l'espace des colonnes de A plutôt que d'employer des opérations sur les lignes comme dans l'élimination gaussienne. De cette façon, nous pouvons identifier explicitement ces quasi-dépendances et les traiter de manière numériquement stable.

4.2 Orthogonalité

Nous avons déterminé quand le problème des moindres carrés est difficile, mais nous pouvons également nous demander quand il est le plus simple. Si nous pouvons réduire un système au cas simple sans induire de problèmes de conditionnement en cours de route, nous aurons trouvé une manière plus stable de contourner les problèmes expliqués au §4.1.

Évidemment, le système linéaire le plus facile à résoudre est In×nx = b : La solution est simplement x ≡ b ! Il est peu probable que nous entrions explicitement ce système linéaire particulier dans notre solveur, mais nous pouvons le faire accidentellement lors de la résolution des moindres carrés. En particulier, même lorsque A = In × n - en fait, A n'a pas besoin d'être une matrice carrée - nous pouvons, dans des circonstances particulièrement chanceuses, trouver que la matrice normale AA satisfait AA = In × n. Pour éviter toute confusion avec le cas général, nous utiliserons la lettre Q pour représenter une telle matrice.

Prier simplement que QQ = In×n donnera probablement une stratégie de solution souhaitable, mais nous pouvons examiner ce cas pour voir comment il devient si favorable. Ecrire les colonnes de Q sous la forme de vecteurs q1, · · · ,qn Rm. Ensuite, il est facile de vérifier que le produit QQ a la structure suivante :

$$QQ = \begin{matrix} - & q_1 & - \\ - & q_2 & - \\ & \vdots & \\ - & q_n & - \end{matrix} \begin{matrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{matrix} = \begin{matrix} q_1 \cdot q_1 & q_1 \cdot q_2 & \cdots & q_1 \cdot q_n \\ q_2 \cdot q_2 & \cdots & q_2 \cdot q_n & \\ \vdots & \vdots & \cdots & \vdots \\ q_n \cdot q_1 & q_n \cdot q_2 & \cdots & q_n \cdot q_n \end{matrix}$$

Si l'expression de droite est égale à In×n, on obtient la relation suivante :

$$q_i \cdot q_j = \begin{matrix} 1 \text{ quand } i = j \\ 0 \text{ quand } i \neq j \end{matrix}$$

En d'autres termes, les colonnes de Q sont de longueur unitaire et orthogonales entre elles. On dit qu'ils forment une base orthonormée pour l'espace des colonnes de Q :

Définition 4.1 (Orthonormale ; matrice orthogonale). Un ensemble de vecteurs {v1, · · · ,vk} est orthonormé si vi = 1 pour tout i et vi ·vj = 0 pour tout i = j. Une matrice carrée dont les colonnes sont orthonormées est appelée matrice orthogonale.

Nous avons motivé notre discussion en demandant quand nous pouvons nous attendre à QQ = In×n. Maintenant, il est facile de voir que cela se produit lorsque les colonnes de Q sont orthonormées. De plus, si Q est carré et inversible avec QQ = In×n, alors simplement en multipliant les deux côtés de cette expression par Q−1 on trouve Q−1 = Q. Ainsi, résoudre Qx = b dans ce cas est aussi simple que de multiplier des deux côtés par la transposition Q.

L'orthonormalité a également une forte interprétation géométrique. Rappelons du chapitre 0 que nous pouvons considérer deux vecteurs orthogonaux a et b comme étant perpendiculaires. Ainsi, un ensemble orthonormé de vecteurs

est simplement un ensemble de vecteurs perpendiculaires de longueur unitaire dans · Si Q est orthogonal, alors son action fait Rn n'affectant pas la longueur des vecteurs :

$$\|Qx\|^2 = x^\top Q^\top Qx = x^\top I_n x = x \cdot x = \|x\|^2$$

De même, Q ne peut pas affecter l'angle entre deux vecteurs, puisque :

$$(Qx) \cdot (Qy) = x^\top Q^\top Qy = x^\top I_n y = x \cdot y$$ De ce point de

vue, si Q est orthogonal, alors Q représente une isométrie de Rn c'est-à-dire qu'il conserve les longueurs et les angles. En d'autres termes, il peut faire pivoter ou refléter des vecteurs mais ne peut pas les mettre à l'échelle ou les cisailler. À un niveau élevé, l'algèbre linéaire des matrices orthogonales est plus facile car leur action n'affecte pas la géométrie de l'espace sous-jacent de manière non triviale.

4.2.1 Stratégie pour les matrices non orthogonales

Sauf circonstances particulières, la plupart de nos matrices A lors de la résolution de Ax =b ou du problème des moindres carrés correspondant ne seront pas orthogonales, donc la machinerie du §4.2 ne s'applique pas directement. Pour cette raison, nous devons faire quelques calculs supplémentaires pour relier le cas général au cas orthogonal.

Soit une matrice $A \in \mathbb{R}^{m\times n}$, et dénotons son espace de colonne comme col A ; rappelons que col A représente l'étendue des colonnes de A. Supposons maintenant qu'une matrice $B \in \mathbb{R}^{n\times n}$ soit inversible. On peut faire une observation simple sur l'espace colonne de AB par rapport

à celui de A : Lemme 4.1 (Invariance de l'espace colonne). Pour tout $A \in \mathbb{R}^{m\times n}$ et inversible $B \in \mathbb{R}^{n\times n}$

$$\text{col } A = \text{col } AB.$$

Preuve. Supposons$b \in$ col A. Alors, par définition de la multiplication par A il existe x avec Ax = b. Alors, (AB) · (B −1x) = Ax =b, so$b \in$ col AB. Inversement, prenons $c \in$ col AB, donc il existe y avec (AB)y = c. Alors, A · (By) = c, montrant que c est dans la colonne A. □

Rappelons la description « matrice d'élimination » de l'élimination gaussienne : Nous sommes partis d'une matrice A et avons appliqué des matrices d'opérations ligne Ei telles que la suite A, E1A, E2E1A, . . . représenté des systèmes linéaires séquentiellement plus faciles. Le lemme ci-dessus suggère une stratégie alternative pour les situations dans lesquelles nous nous soucions de l'espace des colonnes : appliquer des opérations de colonne à A par post-multiplication jusqu'à ce que les colonnes soient orthonormées. Autrement dit, on obtient un produit Q = AE1E2 · · · Ek tel que Q soit orthonormé. Tant que les Ei sont inversibles, le lemme montre que col Q = col A. L'inversion de ces opérations donne une factorisation $A = QR$ pour $R = E_k^{-1} E_{k-1}^{-1} \cdots E_1^{-1}$.

Comme dans la factorisation LU, si nous concevons R avec soin, la solution du plus petit les problèmes de carrés Ax ≈b peuvent simplifier. En particulier, lorsque A = QR, on peut écrire la solution de A Ax = A b comme suit :

$$\begin{aligned}
x &= (A^\top A)^{-1} A^\top b \\
&= (R^\top Q^\top QR)^{-1} R^\top Q^\top b \quad \text{puisque } A = QR \\
&= (R^\top R)^{-1} R^\top Q^\top b \quad \text{puisque } Q \text{ est orthogonal} \\
&= R^{-1} (R^\top)^{-1} R^\top Q^\top b \quad \text{puisque } (AB)^{-1} = B^{-1} A^{-1} \\
&= R^{-1} Q^\top b
\end{aligned}$$

Ou de manière équivalente, $Rx = Q^\top b$

Ainsi, si nous concevons R comme une matrice triangulaire, alors résoudre le système linéaire Rx = Qb est aussi simple qu'une rétrosubstitution.

Notre tâche pour le reste du chapitre est de concevoir des stratégies pour une telle factorisation.

4.3 Orthogonalisation de Gram-Schmidt

Notre première approche pour trouver des factorisations QR est la plus simple à décrire et à mettre en œuvre mais peut souffrir de problèmes numériques. Nous l'utilisons ici comme stratégie initiale, puis nous l'améliorerons avec de meilleures opérations.

4.3.1 Projections

Supposons que nous ayons deux vecteurs a et b. Ensuite, nous pourrions facilement demander "Quel multiple de a est le plus proche de b ?" Mathématiquement, cette tâche équivaut à minimiser ca −b sur tous les c R possibles. Si nous considérons a et b comme des matrices n × 1 et c comme une matrice 1 × 1, alors ce n'est rien de plus qu'un problème non conventionnel des moindres carrésa · c ≈b. Dans ce cas, les équations normales montrenta a · c = ab, ou

$$c = \frac{un \cdot b}{un \cdot un} = \frac{un \cdot b}{un^2}.$$

On note cette projection de b sur a comme :

$$\text{projet}_{un} \cdot bb = ca = \frac{une}{une \cdot une} une = \frac{un \cdot b}{un^2}$$

Évidemment proj $_{un}$ b est parallèle à a. Qu'en est-il du reste b − proj $_{un}$b? Nous pouvons faire un simple calcul pour savoir:

$$\begin{aligned} a \cdot (b - \text{proj}_{un} b) &= a \cdot b - a \cdot \frac{un \cdot b}{un^2} \\ &= a \cdot b - \frac{une}{un^2} b (une \cdot a) \\ &= a \cdot b - a \cdot b \\ &= 0 \end{aligned}$$

Ainsi, nous avons décomposéb en une composante parallèle à a et une autre composante orthogonale toa.

Supposons maintenant que aˆ1, aˆ2, · · · , aˆk soient orthonormés ; nous mettrons des chapeaux sur des vecteurs de longueur unitaire. Alors, pour tout i unique, nous pouvons voir :

$$\text{projaˆi } b = (aˆi \cdot b)aˆi$$

Le terme de norme n'apparaît pas car aˆi = 1 par définition. On pourrait projeterb sur l'étendue {aˆ1, · · · , aˆk} en minimisant l'énergie suivante sur c1, . . . , ck R :

$$\left\| c1\hat{a}1 + c2\hat{a}2 + \cdots + ck\,\hat{a}k - b \right\|^2 = \sum_{je=1}^{k}\sum_{j=1}^{k} cicj(\hat{a}i \cdot \hat{a}j) - 2b \cdot \sum_{je=1}^{k} cia\hat{i} + b \cdot b$$

en appliquant $v^2 = v \cdot v$

$$= \sum_{je=1}^{k} c_{je}^2 - 2ci\, b \cdot \hat{a}i + \|b\|^2 \quad \text{puisque les } \hat{a}i \text{ sont orthonormés}$$

Notez que la deuxième étape ici n'est valide qu'en raison de l'orthonormalité. Au minimum, la dérivée par rapport à ci est nulle pour chaque ci , ce qui donne :

$$ci = \hat{a}i \cdot b$$

Ainsi, nous avons montré que lorsque aˆ1, · · · , aˆk sont orthonormés, la relation suivante est vraie :

$$\text{projspan}\{\hat{a}1,\cdots,\hat{a}k\}b = (\hat{a}1 \cdot b)\hat{a}1 + \cdots + (\hat{a}k \cdot b)\hat{a}k$$

Ceci est simplement une extension de notre formule de projection, et par une preuve similaire, il est facile de voir que

$$\hat{a}i \cdot (b - \text{projspan}\{\hat{a}1,\cdots,\hat{a}k\}b) = 0.$$

C'est-à-dire que nous avons séparéb en une composante parallèle à l'étendue des aˆi et un résidu perpendiculaire.

4.3.2 Orthogonalisation de Gram-Schmidt

Nos observations ci-dessus conduisent à un algorithme simple d'orthogonalisation, ou à la recherche d'une base orthogonale {aˆ1, · · · , aˆk} dont l'étendue est la même que celle d'un ensemble de vecteurs d'entrée linéairement indépendants {v1, · · · ,vk} :

1 jeu

$$\hat{a}1 \equiv \frac{v1}{\|v1\|}.$$

Autrement dit, nous prenons aˆ1 comme un vecteur unitaire parallèle à v1.

2. Pour i de 2 à k,

(a) Calculer la projection

$$pi \equiv \text{projspan}\{\hat{a}1,\cdots,\hat{a}i-1\}\, vi\ .$$

Par définition aˆ1, · · · , aˆi−1 sont orthonormés, donc notre formule ci-dessus s'applique.

(b) Définir

$$\hat{a}i \equiv \frac{vi - pi}{\|vi - pi\|}.$$

Cette technique, connue sous le nom d'« orthogonalisation de Gram-Schmidt », est une application directe de notre discussion ci-dessus. La clé de la preuve de cette technique est de remarquer que span {v1, · · · ,vi} = span {aˆ1, · · · , aˆi} pour chaque i {1, · · · , k}. L'étape 1 en fait clairement le cas pour i = 1, et pour i > 1 la définition de aˆi à l'étape 2b supprime simplement la projection sur les vecteurs que nous avons déjà vus.

Si nous partons d'une matrice A dont les colonnes sont v1, · · · ,vk , alors nous pouvons implémenter Gram Schmidt comme une série d'opérations de colonne sur A. Diviser la colonne i de A par sa norme équivaut à post-multiplier A par ak × k matrice diagonale. De même, soustraire la projection d'une colonne sur les colonnes orthonormées à sa gauche comme à l'étape 2 équivaut à post-multiplier par une matrice triangulaire supérieure : Assurez-vous de comprendre pourquoi c'est le cas ! Ainsi, notre discussion au §4.2.1 s'applique, et nous pouvons utiliser Gram-Schmidt pour obtenir une factorisation A = QR.

Malheureusement, l'algorithme de Gram-Schmidt peut introduire de sérieuses instabilités numériques dues à l'étape de soustraction. Par exemple, supposons que nous fournissions les vecteursv1 = (1, 1) et v2 = (1 + ε, 1) comme entrée de Gram-Schmidt pour un certain 0 < ε 1. Notez qu'une base évidente pour l'étendue {v1,v2} est { (1, 0), (0, 1)}. Mais, si on applique Gram-Schmidt, on obtient :

$$aˆ1 = \frac{v1}{v1} = \frac{1}{\sqrt{2}} \begin{matrix}1\\1\end{matrix}$$

$$p2 = \frac{2+\varepsilon}{2} \begin{matrix}1\\1\end{matrix}$$

$$v2 - p2 = \begin{matrix}1+\varepsilon\\1\end{matrix} - \frac{2+\varepsilon}{2} \begin{matrix}1\\1\end{matrix} = \frac{1}{2} \begin{matrix}\varepsilon\\-\varepsilon\end{matrix}$$

Notez que v2 − p2 = (√ 2/2) · ε, donc le calcul de aˆ2 nécessitera une division par un scalaire de l'ordre de ε. La division par de petits nombres est une opération numérique instable qu'il faut éviter.

4.4 Transformations du foyer

Au §4.2.1, nous avons motivé la construction de la factorisation QR par des opérations de post-multiplication et de colonnes. Cette construction est raisonnable dans le contexte de l'analyse des espaces de colonnes, mais comme nous l'avons vu dans notre dérivation de l'algorithme de Gram-Schmidt, les techniques numériques résultantes peuvent être instables.

Plutôt que de commencer par A et de post-multiplier par des opérations de colonne pour obtenir Q = AE1 · · · Ek , cependant, nous pouvons préserver notre stratégie de haut niveau de l'élimination gaussienne. Autrement dit, nous pouvons commencer par A et pré-multiplier par des matrices orthogonales Qi pour obtenir Qk · · · Q1A = R; ces Q agiront comme des opérations de ligne, éliminant les éléments de A jusqu'à ce que le produit résultant R soit triangulaire supérieur. Ensuite, grâce à l'orthogonalité des Q, nous pouvons écrire A = QR, obtenant le facteur QR
isation puisque le produit de matrices orthogonales est orthogonal.

Les matrices d'opération de ligne que nous avons utilisées dans l'élimination gaussienne et LU ne suffiront pas pour la factorisation QR car elles ne sont pas orthogonales. Un certain nombre d'alternatives ont été suggérées; nous présenterons une stratégie commune introduite en 1958 par Alston Scott Householder.

L'espace des matrices orthogonales n × n est très grand, nous devons donc trouver un espace de Qi plus petit avec lequel il est plus facile de travailler. D'après nos discussions géométriques au §4.2, nous savons que les matrices orthogonales doivent conserver les angles et les longueurs, donc intuitivement, elles ne peuvent tourner et refléter que les vecteurs.

Heureusement, les réflexions peuvent être faciles à écrire en termes de projections, comme l'illustre la figure NUMBER. Supposons que nous ayons un vecteur b que nous souhaitons réfléchir sur un vecteur v. Nous avons montré que le résidu $r \equiv b - \text{proj}_v b$ est perpendiculaire à v. Comme dans la figure NUMÉRO, la différence $2\text{proj}_v b - b$ réfléchit b sur v.

Nous pouvons développer notre formule de réflexion comme suit :

$$\begin{aligned} 2\text{proj}_v \cdot bb - b &= 2\frac{v}{v \cdot v} v - b \text{ par définition de projection} \\ &= 2v \cdot \frac{vb}{vv} - b \text{ en notation matricielle} \\ &= \frac{2vv}{vv} - En\times n\ b \\ &\equiv -Hv\ b \text{ où le négatif est introduit pour s'aligner avec d'autres traitements} \end{aligned}$$

Ainsi, nous pouvons penser à refléter b sur v comme appliquer un opérateur linéaire $-Hv$ à b ! Bien sûr, Hv sans le négatif est toujours orthogonal, nous l'utiliserons donc à partir de maintenant.

Supposons que nous effectuions la première étape de la substitution directe lors de l'élimination gaussienne. Ensuite, on souhaite prémultiplier A par une matrice qui ramène la première colonne de A, que l'on notera a, à un multiple du premier vecteur identité e1. Autrement dit, on veut pour un $c \quad R$:

$$\begin{aligned} ce1 &= Hva \\ &= En\times n - \frac{2vv}{vv}\ _{un} \\ &= a - 2v\ \frac{\text{Virginie}}{vv} \end{aligned}$$

Déplacement des termes dans les émissions

$$v = (a - ce1) \cdot \frac{vv}{2va}$$

Autrement dit, v doit être parallèle à la différence $a - ce1$. En fait, la mise à l'échelle de v n'affecte pas la formule de Hv , nous pouvons donc choisir $v = a - ce1$. Ensuite, pour que notre relation tienne, nous devons avoir

$$\begin{aligned} 1 &= \frac{vv}{2va} \\ &= \frac{{}_{un}{}^{2} - 2ce1 \cdot une^{2}}{+ c\ 2(a \cdot a - ce1} \\ \text{Ou, } 0 &= un^{2\,2-c} \cdot a) = \quad c = \pm a \end{aligned}$$

Avec ce choix de c, nous avons montré :

$$HvA = \begin{matrix} c \times \times \times \\ 0 \times \times \times \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ 0 \times \times \times \end{matrix}$$

Nous venons de franchir une étape proche de l'élimination directe en utilisant uniquement des matrices orthogonales !

En continuant, dans la notation de CITE lors de la k-ième étape de triangularisation, nous avons un vecteur a que l'on peut scinder en deux composantes :

$$un = \begin{matrix} a1 \\ a2 \end{matrix}$$

Ici,a1 Rk eta2 Rm−k . On veut trouver v tel que

$$Hva = \begin{matrix} a1 \\ 0 \\ \vdots \\ 0 \end{matrix}$$

En suivant une dérivation parallèle à celle ci-dessus, il est facile de montrer que

$$v = \begin{matrix} 0 \\ a2 \end{matrix} - cek$$

accomplit exactement cette transformation lorsque c = ±a2 ; on choisit généralement le signe de c pour éviter l'annulation en lui faisant prendre le signe opposé à celui de la k-ième valeur ina.

L'algorithme pour Householder QR est donc assez simple. Pour chaque colonne de A, on calcule v en annulant les éléments inférieurs de la colonne et on applique Hv à A. Le résultat final est une matrice triangulaire supérieure R = Hvn · · · Hv1 A. La matrice orthogonale Q est donnée par le produit H qui peut être stocké implicitement sous la forme d'une liste de vecteurs v, qui tient dans le triangle inférieur comme v1 illustré ci-dessus.

4.5 Factorisation QR réduite

Nous terminons notre discussion en revenant au cas le plus général Ax ≈ b lorsque A Rm×n n'est pas carré. Notez que les deux algorithmes dont nous avons parlé dans ce chapitre peuvent factoriser des matrices non carrées A en produits QR, mais le résultat est quelque peu différent :

- Lors de l'application de Gram-Schmidt, on fait des opérations colonnes sur A pour obtenir Q par orthogonalisation. Pour cette raison, la dimension de A est celle de Q, donnant Q Rm×n et R Rn×n
- En utilisant les réflexions de Householder, on obtient Q comme le produit d'un nombre de m × m matrices de réflexion, laissant R Rm×n ·

Supposons que nous soyons dans le cas typique des moindres carrés, pour lesquels m n. Nous préférons toujours utiliser la méthode Householder en raison de sa stabilité numérique, mais maintenant la matrice m × m Q pourrait être trop grande pour être stockée ! Heureusement, nous savons que R est triangulaire supérieur. Par exemple, considérons la structure d'une matrice 5 × 3 R :

$$R = \begin{matrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

Il est facile de voir que tout ce qui se trouve en dessous du carré n × n supérieur de R doit être nul, ce qui donne un simplifi cation:

$$A = QR = Q1\ Q2 \begin{matrix} R1 \\ 0 \end{matrix} = Q1R1$$

Ici, Q1 Rm×n et R1 Rn×n contiennent toujours le triangle supérieur de R. C'est ce qu'on appelle le « réduit ». Factorisation QR de A, puisque les colonnes de Q1 contiennent une base pour l'espace colonne de A plutôt que pour l'ensemble de Rm ; il prend beaucoup moins de place. Notez que la discussion du §4.2.1 s'applique toujours, de sorte que la factorisation QR réduite peut être utilisée pour les moindres carrés de la même manière.

4.6 Problèmes

- tridiagonalisation avec Householder
- Données
- QR sous-déterminé

Chapitre 5

Vecteurs propres

Intéressons-nous maintenant à un problème non linéaire concernant les matrices : trouver leurs valeurs propres et leurs vecteurs propres. Les vecteurs propres x et leurs valeurs propres correspondantes λ d'une matrice carrée A sont déterminés par l'équation Ax = λx. Il existe de nombreuses façons de voir que ce problème est non linéaire.
Par exemple, il existe un produit d'inconnues λ et x, et pour éviter la solution triviale x = 0 on contraint x = 1 ; cette contrainte est circulaire plutôt que linéaire. Grâce à cette structure, nos méthodes de recherche d'espaces propres seront considérablement différentes des techniques de résolution et d'analyse de systèmes linéaires d'équations.

5.1 Motivations

Malgré la forme arbitraire de l'équation Ax = λx, le problème de la recherche de vecteurs propres et de valeurs propres se pose naturellement dans de nombreuses circonstances. Nous motivons notre discussion avec quelques exemples ci-dessous.

5.1.1 Statistiques

Supposons que nous ayons des machines pour collecter plusieurs observations statistiques sur une collection d'éléments. Par exemple, dans une étude médicale, nous pouvons collecter l'âge, le poids, la tension artérielle et la fréquence cardiaque de 100 patients. Ensuite, chaque patient i peut être représenté par un point xi dans R4 stockant ces quatre valeurs.

Bien sûr, ces statistiques peuvent présenter une forte corrélation. Par exemple, les patients dont la pression artérielle est élevée peuvent être susceptibles d'avoir un poids ou une fréquence cardiaque plus élevés. Pour cette raison, bien que nous ayons collecté nos données dans R4 , elles peuvent en réalité, dans une certaine mesure approximative, vivre dans un espace dimensionnel inférieur capturant mieux les relations entre les différentes variables.

Pour l'instant, supposons qu'il existe en fait un espace unidimensionnel se rapprochant de notre ensemble de données. Ensuite, nous nous attendons à ce que tous les points de données soient presque parallèles à un vecteur v, de sorte que chacun puisse être écrit comme xi ≈ civ pour différents ci R. D'avant, nous savons que la meilleure approximation de xi parallèle à v est proj $_v$ XI :

$$\text{projet}_v = \frac{xi \cdot v\, xi}{v \cdot v} v \text{ par définition}$$

$$= (xi \cdot \hat{v})\hat{v} \text{ puisque } v \cdot v = v^2$$

Ici, nous définissons v̂ ≡ v/v. Bien sûr, la grandeur de v n'a pas d'importance pour le problème à résoudre, il est donc raisonnable de chercher dans l'espace des vecteurs unitaires v̂.

En suivant le modèle des moindres carrés, nous avons un nouveau problème d'optimisation :

$$\text{minimiser} \sum_i \left\| x_i - \text{proj}_{\hat{v}} x_i \right\|^2$$
$$\text{tel que } \|\hat{v}\| = 1$$

Nous pouvons simplifier un peu notre objectif d'optimisation :

$$\sum_i \left\| x_i - \text{proj}_{\hat{v}} x_i \right\|^2 = \sum_i \left\| x_i - (x_i \cdot \hat{v})\hat{v} \right\|^2 \quad \text{par définition de projection}$$
$$= \sum_i \|x_i\|^2 - (x_i \cdot \hat{v})^2 \quad \text{puisque } \|\hat{v}\| = 1 \text{ et } \|w\|^2 = w \cdot w$$
$$= \text{const.} - \sum_i (x_i \cdot \hat{v})^2$$

Cette dérivation montre que l'on peut résoudre un problème d'optimisation équivalent :

$$\text{maximiser } \|X\hat{v}\|^2$$
$$\text{tel que } \|\hat{v}\|^2 = 1,$$

où les colonnes de X sont les vecteurs x_i. Notez que $\|X\hat{v}\|^2 = \hat{v}^\top X^\top X \hat{v}$, donc par exemple 0.27 le vecteur $\hat{v}$ correspond au vecteur propre de $X^\top X$ avec la valeur propre la plus élevée. Le vecteur $\hat{v}$ est connu comme la première composante principale de l'ensemble de données.

5.1.2 Équations différentielles

De nombreuses forces physiques peuvent être écrites en fonction de la position. Par exemple, la force entre deux particules aux positions x et y dans $\mathbb{R}^3$ exercée par un ressort peut être écrite comme $k(x - y)$ par la loi de Hooke ; ces forces de ressort sont utilisées pour approximer les forces qui maintiennent le tissu ensemble dans de nombreux systèmes de simulation. Bien que ces forces ne soient pas nécessairement linéaires en position, nous les approchons souvent de manière linéaire. En particulier, dans un système physique à n particules, coder les positions de toutes les particules simultanément dans un vecteur $X \in \mathbb{R}^{3n}$. Ensuite, si nous supposons une telle approximation, nous pouvons écrire que les forces dans le système sont approximativement $F \approx AX$ pour une certaine matrice A.

Rappelez-vous la deuxième loi du mouvement de Newton F = ma, ou la force est égale à la masse multipliée par l'accélération. Dans notre contexte, nous pouvons écrire une matrice de masse diagonale $M \in \mathbb{R}^{3n \times 3n}$ contenant la masse de chaque particule du système. Alors, on connaît $F = MX''$ où prime désigne la différenciation dans le temps. Bien sûr, $X = (X')'$, donc au final on a un système d'équations du premier ordre :

$$\frac{d}{dt}\begin{bmatrix} X \\ V \end{bmatrix} = \begin{bmatrix} 0 & I_{3n\times 3n} \\ M^{-1}A & 0 \end{bmatrix}\begin{bmatrix} X \\ V \end{bmatrix}$$

Ici, nous calculons simultanément les deux positions dans $X \in \mathbb{R}^{3n}$ et les vitesses $V \in \mathbb{R}^{3n}$ de toutes les n particules en fonction du temps.

Plus généralement, les équations différentielles de la forme $x' = Ax$ apparaissent dans de nombreux contextes, notamment la simulation de tissus, de ressorts, de chaleur, de vagues et d'autres phénomènes. Supposons que nous connaissions les vecteurs propres

x1, . . . ,xk de A, tel que Axi = λixi . Si nous écrivons la condition initiale de l'équation différentielle en termes de vecteurs propres, comme

$$x(0) = c_1x_1 + \cdots + c_kx_k ,$$

alors la solution de l'équation peut s'écrire sous forme fermée :

$$x(t) = c_1e^{\lambda_1 t}x_1 + \cdots + c_ke^{\lambda_k t}x_k ,$$

Cette solution est facile à vérifier à la main. Autrement dit, si nous écrivons les conditions initiales de cette équation différentielle en termes de vecteurs propres de A, alors nous connaissons sa solution pour tous les instants t ≥ 0 gratuitement. Bien sûr, cette formule n'est pas la fin de l'histoire de la simulation : trouver l'ensemble complet des vecteurs propres de A coûte cher, et A peut changer avec le temps.

5.2 Enrobage spectral

Supposons que nous ayons une collection de n éléments dans un ensemble de données et une mesure wij ≥ 0 de la similarité de chaque paire d'éléments i et j ; nous supposerons wij = wji. Par exemple, on nous donne peut-être une collection de photographies et on utilise wij pour comparer la similarité de leurs distributions de couleurs. Nous pourrions souhaiter trier les photographies en fonction de leur similitude pour simplifier la visualisation et l'exploration de la collection.

Un modèle pour ordonner la collection pourrait consister à attribuer un numéro xi à chaque élément i, en demandant que des objets similaires reçoivent des numéros similaires. Nous pouvons mesurer à quel point une affectation regroupe des objets similaires en utilisant l'énergie

$$E(x) = \sum_{ij} w_{ij}(x_i - x_j)^2 .$$

Autrement dit, E(x) demande que les éléments i et j avec des scores de similarité élevés wij soient mappés sur des valeurs proches.

Bien sûr, minimiser E(x) sans contrainte donne un minimum évident : xi = const. pour tout moi. Ajouter une contrainte x = 1 ne supprime pas cette solution constante ! En particulier, prendre xi = 1/ √ n pour tout i donne x = 1 et E(x) = 0 de façon inintéressante. Ainsi, nous devons également supprimer ce cas:

$$\begin{aligned} &\text{minimiser } E(x) \\ &\text{tel que } x^2 = 1 \\ &\qquad 1 \cdot x = 0 \end{aligned}$$

Notez que notre deuxième contrainte demande que la somme de x soit nulle.

Encore une fois on peut simplifier l'énergie :

$$\begin{aligned} E(x) &= \sum_{ij} w_{ij}(x_i - x_j)^2 \\ &= \sum_{ij} w_{ij}(x_i^2 - 2x_ix_j + x_j^2) \\ &= \sum_i a_i x_i^2 - 2\sum_{ij} w_{ij}x_ix_j + \sum_j b_j x_j^2 \\ &\qquad \text{pour } a_i \equiv \sum_j w_{ij} \text{ et } b_j \equiv \sum_i w_{ij} \\ &= x^\top(A - 2W + B)x \text{ où } \mathrm{diag}(A) = a \text{ et } \mathrm{diag}(B) = b = x^\top(2A - 2W)x \text{ par} \\ &\quad \text{symétrie de } W \end{aligned}$$

Il est facile de vérifier que 1 est un vecteur propre de 2A − 2W de valeur propre 0. Plus intéressant, le vecteur propre correspondant à la deuxième plus petite valeur propre correspond à la solution de notre objectif de minimisation ci-dessus ! (À FAIRE : ajouter la preuve KKT de la conférence)

5.3 Propriétés des vecteurs propres

Nous avons établi une variété d'applications nécessitant le calcul de l'espace propre. Avant de pouvoir explorer les algorithmes à cette fin, cependant, nous examinerons de plus près la structure du problème des valeurs propres.

Nous pouvons commencer par quelques définitions qui sont probablement évidentes à ce stade :

Définition 5.1 (Valeur propre et vecteur propre). Un vecteur propre x = 0 d'une matrice A Rn×n est tout vecteur vérifiant Ax = λx pour un certain λ R ; le λ correspondant est appelé valeur propre. Les valeurs propres complexes et les vecteurs propres satisfont les mêmes relations avec λ C et x Cn .

Définition 5.2 (Spectre et rayon spectral). Le spectre de A est l'ensemble des valeurs propres de A. Le rayon spectral ρ(A) est la valeur propre λ maximisant |λ|.

L'échelle d'un vecteur propre n'a pas d'importance. En particulier, la mise à l'échelle d'un vecteur propre x par c donne A(cx) = cAx = cλx = λ(cx), donc cx est un vecteur propre avec la même valeur propre. Nous restreignons souvent notre recherche en ajoutant une contrainte x = 1. Même cette contrainte ne lève pas complètement l'ambiguïté, puisque maintenant ±x sont les deux vecteurs propres avec la même valeur propre.

Les propriétés algébriques des vecteurs propres et des valeurs propres pourraient facilement remplir un livre. Nous limiterons notre discussion à quelques théorèmes importants qui affectent la conception des algorithmes numériques ; nous suivrons le développement de CITE AXLER. Tout d'abord, nous devons vérifier que chaque matrice a au moins un vecteur propre afin que notre recherche ne soit pas vaine. Notre stratégie habituelle est de remarquer que si λ est une valeur propre telle que Ax = λx, alors (A − λIn×n)x =0 ; ainsi, λ est une valeur propre exactement lorsque la matrice A − λIn×n n'est pas de plein rang.

Lemme 5.1 (théorème CITE 2.1). Toute matrice A Rn×n a au moins un vecteur propre (complexe).

Preuve. Prenons n'importe quel vecteur x Rn\{0}. L'ensemble {x, Ax, A 2x, · · · A nx} doit être linéairement dépendant car il contient n + 1 vecteurs à n dimensions. Donc, il existe des constantes c0, . . . , cn R avec cn = 0 tel que

$$0 = c0x + c1Ax + \cdots + cnA^{n}x.$$

On peut écrire un polynôme

$$f(z) = c0 + c1z + \cdots + cnz^{n}.$$

D'après le théorème fondamental de l'algèbre, il existe n racines zi C telles

$$\text{que } f(z) = cn(z - z1)(z - z2)\cdots(z - zn).$$

Ensuite nous avons:

$$\begin{aligned} 0 &= c0x + c1Ax + \cdots + cnA^{n}x \\ &= (c0\, In\times n + c1A + \cdots + cnA^{n})x \\ &= cn(A - z1\, In\times n)\cdots(A - zn\, In\times n)x \text{ par notre factorisation} \end{aligned}$$

Ainsi, au moins un A − zi In×n a un espace nul, montrant qu'il existe v avec Av = ziv, au besoin.

□

Il y a un fait supplémentaire qui vaut la peine d'être vérifié pour motiver notre discussion sur le calcul des vecteurs propres. tation :

Lemme 5.2 (Proposition CITE 2.2). Les vecteurs propres correspondant à différentes valeurs propres doivent être linéairement indépendants.

Preuve. Supposons que ce ne soit pas le cas. Alors il existe des vecteurs propres x1, · · · ,xk de valeurs propres distinctes λ1, · · · , λk linéairement dépendantes. Ceci implique qu'il existe des coefficients c1, . . . , ck pas tous nuls avec 0 = c1x1 + · · · + ckxk . Si on prémultiplie par la matrice (A − λ2 In×n)· · ·(A − λk In×n), on trouve :

$$\begin{aligned} 0 &= (A - \lambda_2 I_{n\times n}) \cdot \cdot (A - \lambda_k I_{n\times n})(c_1x_1 + \cdot \cdot \cdot + c_kx_k) \\ &= c_1(\lambda_1 - \lambda_2) \cdot \cdot (\lambda_1 - \lambda_k)x_1 \text{ puisque } Ax_i = \lambda_i x_i \end{aligned}$$

Puisque tous les λi sont distincts, cela montre que c1 = 0. Une preuve similaire montre que le reste des ci doit être nul, ce qui contredit la dépendance linéaire. □

Ce lemme montre qu'une matrice n × n peut avoir au plus n valeurs propres distinctes, puisqu'un ensemble de n valeurs propres produit n vecteurs linéairement indépendants. Le nombre maximal de vecteurs propres linéairement indépendants correspondant à une seule valeur propre λ est appelé multiplicité géométrique de λ.

Il n'est pas vrai, cependant, qu'une matrice doit avoir exactement n vecteurs propres linéairement indépendants. C'est le cas de nombreuses matrices, que nous appellerons non défectueuses :

Définition 5.3 (Non défectueux). Une matrice A Rn×n est non défectueuse ou diagonalisable si ses vecteurs propres couvrent Rn .

Nous appelons une telle matrice diagonalisable pour la raison suivante : Si une matrice est diagonalisable, alors elle a n vecteurs propresx1, . . . ,xn Rn avec les valeurs propres correspondantes (éventuellement non uniques) λ1, . . . , λn. Prenons les colonnes de X comme vecteurs xi , et définissons D comme la matrice diagonale de valeurs propres λ1, . . . , λn le long de la diagonale. Alors, par définition des valeurs propres on a AX = XD ; c'est simplement une version "empilée" de Axi = λixi . Autrement dit,

$$D = X^{-1}AX,$$

signifiant que A est diagonalisé par une transformation de similarité A → X −1AX : Définition

5.4 (Matrices similaires). Deux matrices A et B sont semblables s'il existe T avec B = T −1AT.

Des matrices similaires ont les mêmes valeurs propres, puisque si Bx = λx, alors T −1ATx = λx. De manière équivalente, A(Tx) = λ(Tx), montrant que Tx est un vecteur propre de valeur propre λ.

5.3.1 Matrices définies symétriques et positives

Comme on pouvait s'y attendre, compte tenu de notre attention particulière aux matrices normales AA, les matrices définies symétriques et/ou positives bénéficient d'une structure de vecteurs propres particulière. Si nous pouvons vérifier l'une ou l'autre de ces propriétés, des algorithmes spécialisés peuvent être utilisés pour extraire plus rapidement leurs vecteurs propres.

Tout d'abord, nous pouvons prouver une propriété des matrices symétriques qui efface le besoin de matrices complexes arithmétique. On commence par faire une généralisation des matrices symétriques aux matrices en Cn×n :

Définition 5.5 (Complexe conjugué). Le conjugué complexe d'un nombre z ≡ a + bi C est $\bar{z} \equiv a - bi$.

Définition 5.6 (Transposition conjuguée). La transposée conjuguée de A Cm×n est AH ≡ A¯.

Définition 5.7 (Matrice hermitienne). Une matrice A Cn×n est hermitienne si A = A H.

Remarquons qu'une matrice symétrique A Rn×n est automatiquement hermitienne car elle n'a pas de partie complexe. Avec cette légère généralisation en place, nous pouvons prouver une propriété de symétrie pour les valeurs propres. Notre preuve utilisera le produit scalaire des vecteurs dans Cn , donné par

$$x,y = \sum x_i \bar{y}_i ,$$

où x,y Cn . Remarquons qu'encore une fois cette définition coïncide avec x ·y lorsque x,y Rn . Pour la plupart, les propriétés de ce produit scalaire coïncident avec celles du produit scalaire sur Rn sauf que v, w = w ,v. , un notable

Lemme 5.3. Toutes les valeurs propres des matrices hermitiennes sont réelles.

Preuve. Supposons que A Cn×n est hermitienne avec Ax = λx. En mettant à l'échelle, nous pouvons supposer ^{2} = x,x = x 1. Alors, nous avons :

λ = λx,x puisque x est de norme 1
= λx,x par linéarité de ·, ·
= Ax,x puisque Ax = λx
= (Ax) x¯ par définition de = x ·, ·
(A¯x) en développant le produit et en utilisant ab = a¯ ¯b = x, A Hx
par définition de A H et ·, ·
H = x, Ax puisque A = A =
λ¯x,x puisque Ax = λx = λ¯
puisque x est de norme 1

Ainsi λ = λ¯ , ce qui ne peut arriver que si λ R, au besoin. □

Les matrices symétriques et hermitiennes bénéficient également d'une propriété d'orthogonalité particulière pour leur eigenvec teurs :

Lemme 5.4. Les vecteurs propres correspondant à des valeurs propres distinctes de matrices hermitiennes doivent être orthogonaux.

Preuve. Supposons que A Cn×n est hermitienne, et supposons λ = μ avec Ax = λx et Ay = μy. Par le lemme précédent nous connaissons λ, μ R. Alors, Ax,y = λx,y. Mais puisque A est hermitienne on peut aussi écrire Ax,y = x, A Hy = x, Ay = μx,y. Ainsi, λx,y = μx,y. Puisque λ = μ, on doit avoir x,y = 0.

□

Enfin, nous pouvons énoncer sans preuve un résultat suprême de l'algèbre linéaire, le théorème spectral. Ce théorème stipule qu'aucune matrice symétrique ou hermitienne ne peut être défectueuse, ce qui signifie qu'une matrice n × n satisfaisant cette propriété a exactement n vecteurs propres orthogonaux.

Théorème 5.1 (Théorème spectral). Supposons que A Cn×n est hermitien (si A Rn×n , supposons qu'il est symétrique). Alors, A admet exactement n vecteurs propres orthonormés x1, · · · ,xn de valeurs propres (éventuellement répétées) λ1, . . . , λn . Autrement dit, il existe une matrice orthonormée X de vecteurs propres et une matrice diagonale D de valeurs propres telles que D = X AX.

Ce théorème implique que tout vecteur y Rn peut être décomposé en une combinaison linéaire des vecteurs propres d'une matrice hermitienne A. De nombreux calculs sont plus faciles dans cette base, comme indiqué ci-dessous :

Exemple 5.1 (Calcul par vecteurs propres). Prenez x1, . . . ,xn Rn étant les vecteurs propres de longueur unitaire de la matrice symétrique A Rn×n . Supposons que nous souhaitions résoudre Ay =b. Nous pouvons écrire

$$b = c_1x_1 + \cdots + c_nx_n,$$

où ci =b · xi par orthonormalité. Il est facile de deviner la solution suivante :

$$\frac{c_1}{\lambda_1}x_1 + \cdots + \frac{c_n}{\lambda_n}x_n. = y$$

On retrouve notamment :

$$\begin{aligned} Ay &= A\left(\frac{c_1}{\lambda_1}x_1 + \cdots + \frac{c_n}{\lambda_n}x_n\right) \\ &= \frac{c_1}{\lambda_1}Ax_1 + \cdots + \frac{c_n}{\lambda_n}Ax_n \\ &= c_1x_1 + \cdots + c_nx_n \\ &= b, \text{ au choix.} \end{aligned}$$

Le calcul ci-dessus est à la fois un résultat positif et négatif. Il montre que, étant donné les vecteurs propres de A symétrique, les opérations comme l'inversion sont simples. D'un autre côté, cela signifie que trouver l'ensemble complet des vecteurs propres d'une matrice symétrique A est "au moins" aussi difficile que de résoudre Ax = b.

De retour de notre incursion dans les nombres complexes, nous revenons aux nombres réels pour prouver un dernier fait utile quoique simple sur les matrices définies positives :

Lemme 5.5. Toutes les valeurs propres des matrices définies positives sont non négatives.

Preuve. Soit A Rn×n définie positive, et supposons Ax = λx avec x = 1. Par définition positive, on connait x Ax ≥ 0. Mais, x Ax = x (λx) = λx = λ, au besoin. □

5.3.2 Propriétés spécialisées1

Polynôme caractéristique

Rappelons que le déterminant d'une matrice det A satisfait la relation que det A = 0 si et seulement si A est inversible. Ainsi, une façon de trouver les valeurs propres d'une matrice est de trouver les racines du polynôme caractéristique

$$p_A(\lambda) = \det(A - \lambda I_{n\times n}).$$

Nous ne définirons pas les déterminants dans notre discussion ici, mais la simplification de pA révèle qu'il s'agit d'un polynôme de degré n en λ. Cela fournit une raison alternative pour laquelle il y a au plus n valeurs propres distinctes, puisqu'il y a au plus n racines de cette fonction.

A partir de cette construction, nous pouvons définir la multiplicité algébrique d'une valeur propre comme sa multiplicité en tant que racine de pA. Il est facile de voir que la multiplicité algébrique est au moins aussi grande que la multiplicité géométrique

1Cette section peut être ignorée si les lecteurs manquent de connaissances suffisantes, mais elle est incluse par souci d'exhaustivité.

multiplicité. Si la multiplicité algébrique est 1, la racine est dite simple, car elle correspond à un seul vecteur propre linéairement dépendant de tous les autres. Les valeurs propres pour lesquelles les multiplicités algébriques et géométriques ne sont pas égales sont dites défectueuses.

En analyse numérique, nous évitons de discuter du déterminant d'une matrice. Bien qu'il s'agisse d'une construction théorique pratique, son utilisation pratique est limitée. Les déterminants sont difficiles à calculer. En fait, les algorithmes de valeurs propres ne tentent pas de trouver les racines de pA car cela nécessiterait l'évaluation d'un déterminant. De plus, le déterminant det A n'a rien à voir avec le conditionnement de A, donc un déterminant proche de zéro de det(A − λIn×n) pourrait ne pas montrer que λ est presque une valeur propre de A.

Forme normale de la Jordanie

Nous ne pouvons diagonaliser une matrice que lorsqu'elle a un espace propre complet. Toutes les matrices, cependant, sont similaires à une matrice en forme normale de Jordan, qui a la forme suivante :

- Les valeurs non nulles sont sur les entrées diagonales aii et sur la « superdiagonale » ai(i+1) .
- Les valeurs diagonales sont des valeurs propres répétées autant de fois que leur multiplicité ; la matrice est bloc diagonale autour de ces clusters.
- Les valeurs hors diagonale sont 1 ou 0.

Ainsi, la forme ressemble à ce qui suit

$$\begin{pmatrix} \lambda_1 & 1 & & & & \\ & \lambda_1 & 1 & & & \\ & & \lambda_1 & & & \\ & & & \lambda_2 & 1 & \\ & & & & \lambda_2 & \\ & & & & & \lambda_3 \\ & & & & & & \ddots \end{pmatrix}$$

La forme normale de Jordan est intéressante théoriquement car elle existe toujours, mais la structure 1/0 est discrète et instable sous perturbation numérique.

5.4 Calcul des valeurs propres

Le calcul et l'estimation des valeurs propres d'une matrice est un problème bien étudié avec de nombreuses solutions potentielles. Chaque solution est adaptée à une situation différente, et l'obtention d'un conditionnement ou d'une vitesse maximale nécessite l'expérimentation de plusieurs techniques. Ici, nous couvrons quelques-unes des solutions les plus populaires et les plus simples au problème des valeurs propres fréquemment rencontré dans la pratique.

5.4.1 Itération de puissance

Pour l'instant, supposons que A Rn×n est symétrique. Alors, par le théorème spectral nous pouvons écrire les vecteurs propres x1, . . . ,xn Rn ; nous les trions de sorte que leurs valeurs propres correspondantes satisfassent |λ1| ≥ |λ2| ≥ · · · ≥ |λn|.

Supposons que nous prenions un vecteur arbitraire v. Puisque les vecteurs propres de A s'étendent sur Rn, nous pouvons écrire:

$$v = c_1x_1 + \cdots + c_nx_n.$$

Alors,

$$\begin{aligned} Av &= c_1Ax_1 + \cdots + c_nAx_n \\ &= c_1\lambda_1x_1 + \cdots + c_n\lambda_nx_n \text{ puisque } Ax_i = \lambda_ix_i \\ \lambda_2 \lambda_n & \\ \cdots + c_nx_n \lambda_1 \lambda_1 &= \lambda_1 \left(c_1x_1 + c_2x_2 + \cdots\right) \end{aligned}$$

$$A^2v = \lambda_1^2 \left(c_1x_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^2 c_2x_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^2 c_nx_n\right)$$

$$\vdots$$

$$A^kv = \lambda_1^k \left(c_1x_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k c_2x_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k c_nx_n\right)$$

Notez que lorsque $k \to \infty$, le rapport $(\lambda_i/\lambda_1) \to 0^k$ sauf si $\lambda_i = \lambda_1$, puisque λ_1 a la plus grande amplitude de toute valeur propre par définition. Ainsi, si x est la projection de v sur l'espace des vecteurs propres de valeurs propres λ_1, alors lorsque $k \to \infty$ l'approximation suivante est de plus en plus exacte :

$$\lambda_1^k A^k v \approx x.$$

Cette observation conduit à un algorithme extrêmement simple pour calculer un vecteur propre x de A correspondant à la plus grande valeur propre λ_1 :

1. Soit $v_1 \in \mathbb{R}^n$ un vecteur non nul arbitraire.
2. Itérer jusqu'à convergence pour k croissant :

$$v_k = Av_{k-1}$$

Cet algorithme, appelé itération de puissance, va produire des vecteurs v_k de plus en plus parallèles au x_1 recherché. Il est garanti de converger, même lorsque A est asymétrique, bien que la preuve de ce fait soit plus compliquée que la dérivation ci-dessus. La seule fois où cette technique peut échouer, c'est si nous choisissons accidentellement v_1 tel que $c_1 = 0$, mais les chances que cela se produise sont minces, voire nulles.

Bien sûr, si $|\lambda_1| > 1$, alors $v_k \to \infty$ comme $k \to \infty$, une propriété indésirable pour l'arithmétique à virgule flottante. Rappelez-vous que nous ne nous soucions que de la direction du vecteur propre plutôt que de sa magnitude, donc la mise à l'échelle n'a aucun effet sur la qualité de notre solution. Ainsi, pour éviter cette situation de divergence, nous pouvons simplement normaliser à chaque étape, produisant l'algorithme d'itération de puissance normalisé :

1. Soit $v_1 \in \mathbb{R}^n$ un vecteur non nul arbitraire.
2. Itérer jusqu'à convergence pour k croissant :

$$w_k = Av_{k-1}$$

$$v_k = \frac{w_k}{\|w_k\|}$$

Remarquez que nous n'avons pas décoré la norme · avec un indice particulier. Mathématiquement, n'importe quelle norme suffira pour éviter le problème de divergence, puisque nous avons montré que toutes les normes sur Rn sont équivalentes. En pratique, on utilise souvent la norme de l'infini · ∞ ; dans ce cas il est facile de vérifier que w k → |λ1|.

5.4.2 Itération inverse

Nous avons maintenant une stratégie pour trouver la valeur propre de plus grande grandeur λ1. Supposons que A soit inversible, de sorte que nous puissions évaluer y = A −1v en résolvant Ay = v en utilisant les techniques décrites dans les chapitres précédents.

Si Ax = λx, alors x = λA −1x, ou de manière équivalente

$$UN^{-1}x = \_ \quad \frac{1}{\lambda} \text{fois.}$$

avons montré que 1/λ est une valeur propre de A alors |b| ≥ |a| pour $^{-1}$ avec vecteur propre x. Notez que si |a| ≥ |b| Ainsi, nous tout a, b $^{-1}$R, la plus petite valeur propre de A est donc la plus grande . Cette observation donne une stratégie pour trouver vecteur propre de magnitude de A $^{-1}$ λn plutôt que λ1
appelé itération de puissance inverse :

1. Soit v1 Rn un vecteur non nul arbitraire.
2. Itérer jusqu'à convergence pour k croissant :
   (a) Résoudre pour w k : Aw k = vk−1
   (b) Normaliser : vk = $\frac{wkwk}{\_\_\_}$

Nous résolvons à plusieurs reprises des systèmes d'équations en utilisant la même matrice A, ce qui est une application parfaite des techniques de factorisation des chapitres précédents. Par exemple, si nous écrivons A = LU, alors nous pourrions formuler une version équivalente mais considérablement plus efficace de l'itération de puissance inverse :

1. Facteur A = LU
2. Soit v1 Rn un vecteur non nul arbitraire.
3. Itérer jusqu'à convergence pour k croissant :
   (a) Résoudre pour yk par substitution vers l'avant : Lyk = vk−1 (b)
   Résoudre pour w k par substitution vers l'arrière : Uw k = yk
   (c) Normaliser : vk = $\frac{wkwk}{\_\_\_}$

5.4.3 Changement de vitesse

Supposons que λ2 est la valeur propre avec la deuxième plus grande amplitude de A. Compte tenu de notre dérivation originale de l'itération de puissance, il est facile de voir que l'itération de puissance converge le plus rapidement lorsque |λ2/λ1| est faible, car dans ce cas la puissance (λ2/λ1) ^{k} décroît rapidement. En revanche, si ce rapport est proche de 1, plusieurs itérations d'itération de puissance peuvent être nécessaires avant qu'un seul vecteur propre ne soit isolé.

Si les valeurs propres de A sont λ1, . . . , λn, alors il est facile de voir que les valeurs propres de A − σIn×n sont λ1 − σ, . . . , λn − σ. Ensuite, une stratégie pour faire converger rapidement les itérations de puissance est de choisir σ tel que :

$$\frac{\lambda 2 - \sigma}{\lambda 1 - \sigma} < \frac{\lambda 2}{\lambda 1} .$$

Bien sûr, deviner un tel σ peut être un art, puisque les valeurs propres de A ne sont évidemment pas connues initialement. De même, si on pense que σ est proche d'une valeur propre de A, alors A − σIn×n a une valeur propre proche de 0 que l'on peut révéler par itération inverse.

Une stratégie qui utilise cette observation est connue sous le nom d'itération du quotient de Rayleigh. Si nous avons une estimation fixe à un vecteur propre x de A, alors par NOMBRE l'approximation des moindres carrés de la valeur propre correspondante σ est donnée par

$$\sigma \approx \frac{x \text{ Hache}}{X \frac{2}{2}} .$$

Cette fraction est connue sous le nom de quotient de Rayleigh. Ainsi, on peut tenter d'augmenter la convergence en itérant comme suit :

1. Prendre v1 Rn comme un vecteur non nul arbitraire ou une estimation initiale d'un vecteur propre.

2. Itérer jusqu'à convergence pour k croissant :

   (a) Écrire l'estimation actuelle de la valeur propre

$$\sigma k = \frac{v\ k-1 A v k-1}{v k-1 \frac{2}{2}}$$

(b) Résoudre pour w k : (A − σk In×n)w k (c) Normaliser : vk = $\frac{wkwk}{---}$ = vk−1

Cette stratégie converge beaucoup plus rapidement avec une bonne estimation de départ, mais la matrice A − σk In × n est différente à chaque itération et ne peut pas être préfactorisée en utilisant LU ou la plupart des autres stratégies. Ainsi, moins d'itérations sont nécessaires mais chaque itération prend plus de temps !

5.4.4 Trouver plusieurs valeurs propres

Jusqu'à présent, nous avons décrit des techniques pour trouver une seule paire valeur propre/vecteur propre : alimentez-la pour trouver la plus grande valeur propre, itération inverse pour trouver la plus petite et passez aux valeurs cibles intermédiaires. Bien sûr, pour de nombreuses applications, une seule valeur propre ne suffira pas. Heureusement, nous pouvons également étendre nos stratégies pour gérer ce cas.

Déflation

Rappelez-vous notre stratégie d'itération de puissance : choisissez un v1 arbitraire et multipliez-le itérativement par A jusqu'à ce que seule la plus grande valeur propre λ1 survive. Prenez x1 comme vecteur propre correspondant.

Nous n'avons cependant pas tardé à écarter un mode de défaillance peu probable de cet algorithme, lorsque v1 ·x1 = 0. Dans ce cas, peu importe combien de fois vous prémultipliez par A vous ne récupérerez jamais un vecteur

parallèle à x1, puisque vous ne pouvez pas amplifier une composante nulle. La probabilité de choisir un tel v1 est exactement nulle, donc dans tous les cas sauf les plus pernicieux, l'itération de puissance reste sûre.

Nous pouvons inverser cet inconvénient pour formuler une stratégie permettant de trouver plus d'une valeur propre lorsque A est symétrique. Supposons que nous trouvions x1 et λ1 via une itération de puissance comme précédemment. Maintenant, nous redémarrons l'itération de puissance, mais avant de commencer le projet x1 sur v1. Alors, puisque les vecteurs propres de A sont orthogonaux, l'itération de puissance récupérera la deuxième plus grande valeur propre !

En raison de problèmes numériques, il se peut que l'application de A à un vecteur introduit une petite composante parallèle à x1. En pratique on peut éviter cet effet en projetant à chaque itération. Au final, cette stratégie donne l'algorithme suivant pour calculer les valeurs propres par ordre de grandeur décroissante :

- Pour chaque valeur propre souhaitée = 1, 2, . . .
  1. Soit v1 Rn un vecteur non nul arbitraire.
  2. Itérer jusqu'à convergence pour k croissant : (a)

     Projeter les vecteurs propres que nous avons déjà calculés :

     $$uk = vk-1 - \text{projspan}\{x1,...,x-1\}\ vk-1$$

     (b) Multiplier Auk = w k (c)

     Normaliser : vk = $\frac{wkwk}{---}$
  3. Ajouter le résultat de l'itération à l'ensemble des xi

La boucle interne équivaut à une itération de puissance sur la matrice AP, où P projette x1, . . . ,x−1 .
Il est facile de voir que AP a les mêmes vecteurs propres que A ; ses valeurs propres sont λ , . . . , λn avec les valeurs propres restantes prises à zéro.

Plus généralement, la stratégie de déflation consiste à modifier la matrice A pour que l'itération puissance révèle un vecteur propre que vous n'avez pas encore calculé. Par exemple, AP est une modification de A afin que les grandes valeurs propres que nous avons déjà calculées soient mises à zéro.

Notre stratégie de projection échoue si A est asymétrique, car dans ce cas ses vecteurs propres peuvent ne pas être orthogonaux. D'autres stratégies de déflation moins évidentes peuvent fonctionner dans ce cas. Par exemple, supposons Ax1 = λ1x1 avec x1 = 1. Prenons H comme matrice de Householder telle que Hx1 = e1, le premier vecteur de base standard. Les transformées de similarité une fois de plus n'affectent pas l'ensemble des vecteurs propres, nous pourrions donc essayer de conjuguer par H. Considérez ce qui se passe lorsque nous multiplions HAH par e1 :

HAHe1 = HAHe1 puisque H est symétrique
= HAx1 puisque Hx1 =e1 et H = ^{2} = En×n
λ1Hx1 puisque Ax1 = λ1x1 = λ1e1 par définition de H

Ainsi, la première colonne de HAH est λ1e1, montrant que HAH a la structure suivante (CITE BRUYÈRE):

$$HAH = \begin{matrix} \lambda 1 & b \\ 0 & B \end{matrix} .$$

La matrice B R(n−1)×(n−1) a pour valeurs propres λ2, . . . , λn. Ainsi, une autre stratégie de déflation consiste à construire des matrices B de plus en plus petites avec chaque valeur propre calculée à l'aide d'une itération de puissance.

La déflation a l'inconvénient que nous devons calculer chaque vecteur propre séparément, ce qui peut être lent et peut accumuler des erreurs si les valeurs propres individuelles ne sont pas exactes. Nos stratégies restantes tentent de trouver plus d'un vecteur propre à la fois.

Rappelons que des matrices similaires A et B = T −1AT doivent avoir les mêmes valeurs propres. Ainsi, un algorithme tentant de trouver les valeurs propres de A peut appliquer librement des transformations de similarité à A. Bien sûr, appliquer T en général peut être une proposition difficile, car cela nécessiterait effectivement d'inverser T, nous cherchons donc des matrices T dont les inverses sont faciles à appliquer.

L'une de nos motivations pour dériver la factorisation QR était que la matrice Q est orthogonale, satisfaisant Q−1 = Q. Ainsi, Q et Q−1 sont tout aussi simples à appliquer, ce qui fait des matrices orthogonales de bons choix pour les transformations de similarité.

Mais quelle matrice orthogonale Q choisir ? Idéalement, Q devrait impliquer la structure de A tout en étant simple à calculer. On ne sait pas comment appliquer stratégiquement des transformations simples comme les matrices Householder pour révéler plusieurs valeurs propres,[2] mais nous savons comment générer un tel Q simplement en factorisant A = QR. Alors, on pourrait conjuguer A par Q pour trouver : $Q^{-1}AQ = Q^{\infty}AQ = Q^{\infty}(QR)Q = (Q^{\infty}Q)RQ = RQ$

Étonnamment, conjuguer A = QR par la matrice orthogonale Q revient à écrire le produit RQ !

Sur la base de ce raisonnement, dans les années 1950, plusieurs groupes de mathématiciens européens ont émis l'hypothèse dimensionné le même algorithme itératif élégant pour trouver les valeurs propres d'une matrice A :

1. Prenez A1 = A.
2. Pour k = 1, 2, . . .
   (a) Facteur Ak = QkRk . (b)
   Ecrire Ak+1 = RkQk .

Par notre dérivation ci-dessus, les matrices Ak ont toutes les mêmes valeurs propres que A. De plus, supposons que les Ak convergent vers un certain A^{∞}. Alors, on peut factoriser $A^{\infty} = Q^{\infty}R^{\infty}$, et par convergence on sait $A^{\infty} = Q^{\infty}R^{\infty} = R^{\infty}Q^{\infty}$. Par NOMBRE, les valeurs propres de R^{∞} sont simplement les valeurs le long de la diagonale de R^{∞}, et par NOMBRE le produit $R^{\infty}Q^{\infty} = A^{\infty}$ doit à son tour avoir les mêmes valeurs propres. Enfin, par construction A^{∞} a les mêmes valeurs propres que A. Ainsi, nous avons montré que si l'itération QR converge, elle fait apparaître les valeurs propres de A de manière directe.

Bien sûr, la dérivation ci-dessus suppose qu'il existe A^{∞} avec $A^k \to A^{\infty}$ lorsque $k \to \infty$. En fait, l'itération QR est une méthode stable garantie de converger dans de nombreuses situations importantes, et la convergence peut même être améliorée en changeant de stratégie. Nous ne dériverons pas de conditions exactes ici, mais nous pouvons plutôt donner une idée de la raison pour laquelle cette stratégie apparemment arbitraire devrait converger. Nous donnons ci-dessous une intuition pour le cas symétrique $A = A^{\infty}$ qui est plus facile à analyser grâce à l'orthogonalité des vecteurs propres dans ce cas.

Supposons que les colonnes de A soient données par a1, . . . ,an, et considérons la matrice A^k pour grand k. Nous peut écrire:

$$A^k = A^{k-1} \cdot A = \begin{vmatrix} | & | & & | \\ A^{k-1}a_1 & A^{k-1}a_2 & \cdots & A^{k-1}a_n \\ | & | & & | \end{vmatrix}$$

2Les techniques plus avancées, cependant, font exactement cela !

Par notre dérivation de l'itération de puissance, la première colonne de A^k est en général parallèle au vecteur propre x1 avec la plus grande amplitude $|\lambda 1|$ puisque nous avons pris un vecteur a1 et l'avons multiplié par A plusieurs fois. En appliquant notre intuition à partir de la déflation, supposons que nous projetions a1 sur la deuxième colonne ^{k} . Ce de A vecteur doit être presque parallèle à x2, puisqu'il s'agit de la deuxième valeur propre la plus dominante ! Procédure inductivement, factoriser $A^k = QR$ donnerait un ensemble de vecteurs proches comme les colonnes de Q, par ordre décroissant de valeur propre, avec les valeurs propres correspondantes le long de la diagonale de R.

Bien sûr, le calcul de A^k pour un grand k prend le nombre conditionnel de A à la puissance k, donc QR sur la matrice résultante est susceptible d'échouer ; cela est clair puisque toutes les colonnes de A devraient ressembler à x1 pour un grand k. Nous pouvons cependant faire le constat suivant :

$$
\begin{aligned}
A &= Q_1R_1 \\
A^2 &= (Q_1R_1)(Q_1R_1) \\
&= Q_1(R_1Q_1)R_1 \\
&= Q_1Q_2R_2R_1 \text{ en utilisant la notation d'itération QR ci-dessus, puisque } A_2 = R_1Q_1 \\
&\vdots \\
A^k &= Q_1Q_2 \cdots Q_kR_kR_{k-1} \cdots R_1
\end{aligned}
$$

Le regroupement des variables Qi et des variables Ri séparément fournit une factorisation QR de A, nous nous attendons à ce ^{k} . Ainsi, que les colonnes de $Q_1 \cdots Q_k$ convergent vers les vecteurs propres de A.

Par un argument similaire, on peut trouver

$$
\begin{aligned}
A &= Q_1R_1 \\
A_2 &= R_1Q_1 \text{ par notre construction d'itération QR} \\
&= Q_2R_2 \text{ par définition de la factorisation} \\
A_3 &= R_2Q_2 \text{ de l'itération QR} \\
&= Q_2^{-1}A_2Q_2 \text{ puisque } A_2 = Q_2R_2 \\
&= Q_2^{-1}R_1Q_1Q_2 \text{ puisque } A_2 = R_1Q_1 \\
&= Q_2^{-1}Q_1^{-1}AQ_1Q_2 \text{ puisque } A = Q_1R_1 \\
&\vdots \\
A_{k+1} &= Q_k^{-1} \cdots Q_1^{-1}AQ_1 \cdots Q_k \text{ par induction} \\
&= (Q_1 \cdots Q_k)^{-1}A(Q_1 \cdots Q_k)
\end{aligned}
$$

où Ak est la k-ième matrice de l'itération QR. Ainsi, Ak+1 est simplement la matrice A conjuguée par le produit $\bar{Q}_k \equiv Q_1 \cdots Q_k$. Nous avons soutenu précédemment que les colonnes de $\bar{Q}_k$ convergent vers les vecteurs propres de A. Ainsi, puisque la conjugaison par la matrice des vecteurs propres donne une matrice diagonale de valeurs propres, nous savons $A_{k+1} = \bar{Q}_k^{-1}A\bar{Q}$ aura des valeurs propres approchées de A le long de sa diagonale lorsque $k \to \infty$.

Notre justification de l'itération QR consistait à analyser les colonnes de A de k comme k → ∞ comme extension
l'itération de puissance. Plus généralement, pour un vecteurb , nous pouvons examiner la matrice dite de Krylov

$$K_k = \begin{vmatrix} | & | & | & & | \\ b & Ab & A^2b & \cdots & A^{k-1}b \\ | & | & | & & | \end{vmatrix}.$$

Les méthodes analysant Kk pour trouver des vecteurs propres et des valeurs propres sont généralement connues sous le nom de méthodes de sous-espace de Krylov. Par exemple, l'algorithme d'itération d'Arnoldi utilise l'orthogonalisation de Gram-Schmidt pour maintenir une base orthogonale {q1, . . . ,qk} pour l'espace colonne de Kk :

1. Commencez par prendre q1 comme un vecteur de norme unitaire arbitraire
2. Pour k = 2, 3, . . .
   (a) Takeak = Aqk−1 (b) Projetez les q que vous avez déjà calculés :

$$b_k = a_k - \text{proj}_{\text{span}\{q_1,\ldots,q_{k-1}\}} a_k$$

   (c) Renormaliser pour trouver le prochain qk = bk/bk.

La matrice Qk dont les colonnes sont les vecteurs trouvés ci-dessus est une matrice orthogonale avec le même espace de colonne que Kk , et les estimations de valeurs propres peuvent être récupérées à partir de la structure de $Q_k^* A Q_k$. L'utilisation de Gram-Schmidt rend cette technique instable et le timing s'aggrave progressivement à mesure que k augmente, cependant, de nombreuses extensions sont nécessaires pour la rendre réalisable. Par exemple, une stratégie consiste à exécuter certaines itérations d'Arnoldi, à utiliser la sortie pour générer une meilleure estimation du q1 initial et à redémarrer. Les méthodes de cette classe conviennent aux problèmes nécessitant plusieurs vecteurs propres à l'une des extrémités du spectre sans calculer l'ensemble complet.

5.5 Sensibilité et conditionnement

Comme averti, nous n'avons décrit que quelques techniques de valeurs propres issues d'une littérature riche et ancienne. Presque toutes les techniques algorithmiques ont été expérimentées pour trouver des spectres, des méthodes itératives à la recherche de racine sur le polynôme caractéristique aux méthodes qui divisent les matrices en blocs pour un traitement parallèle.

Tout comme dans les solveurs linéaires, nous pouvons évaluer le conditionnement d'un problème aux valeurs propres indépendamment de la technique de résolution. Cette analyse peut aider à comprendre si un schéma itératif simpliste réussira à trouver les vecteurs propres d'une matrice donnée ou si des méthodes plus complexes sont nécessaires ; il est important de noter que le conditionnement d'un problème aux valeurs propres n'est pas le même que le nombre de conditions de la matrice pour résoudre les systèmes, car ce sont des problèmes distincts.

Supposons qu'une matrice A ait un vecteur propre x de valeur propre λ. L'analyse du conditionnement du problème des valeurs propres implique d'analyser la stabilité de x et λ aux perturbations dans A. À cette fin, nous pourrions perturber A par une petite matrice δA, modifiant ainsi l'ensemble des vecteurs propres. En particulier, on peut écrire les vecteurs propres de A + δA comme des perturbations des vecteurs propres de A en résolvant le problème

$$(A + \delta A)(x + \delta x) = (\lambda + \delta\lambda)(x + \delta x).$$

L'expansion des deux côtés donne:

$$Ax + A\delta x + \delta A \cdot x + \delta A \cdot \delta x = \lambda x + \lambda \delta x + \delta \lambda \cdot x + \delta \lambda \cdot \delta x$$

En supposant que δA est petit, nous supposerons3 que δx et δλ sont également petits. Les produits entre ces variables sont alors négligeables, ce qui donne l'approximation suivante :

$$Ax + A\delta x + \delta A \cdot x \approx \lambda x + \lambda \delta x + \delta \lambda \cdot x$$

Puisque Ax = λx, nous pouvons soustraire cette valeur des deux côtés pour trouver :

$$A\delta x + \delta A \cdot x \approx \lambda \delta x + \delta \lambda \cdot x$$

Nous appliquons maintenant une astuce analytique pour compléter notre dérivation. Puisque Ax = λx, on sait (A − λIn×n)x =0, donc A − λIn×n n'est pas de rang complet. La transposée d'une matrice n'est de rang complet que si la matrice est de rang complet, donc nous savons que (A − λIn×n) = A − λIn×n a aussi un vecteur d'espace nul y. Ainsi A y = λy; on peut appeler y le vecteur propre gauche correspondant à x. Nous pouvons multiplier à gauche notre estimation de perturbation ci-dessus par y :

$$y\,(A\delta x + \delta A \cdot x) \approx y\,(\lambda \delta x + \delta \lambda \cdot x)$$

Puisque A y = λy, on peut simplifier :

$$y\,\delta A \cdot x \approx \delta \lambda y\, x$$

Réorganiser les rendements :

$$\approx \quad \frac{y\,(\delta A)x\,\delta \lambda}{yx}$$

Supposons x = 1 et y = 1. Alors, si nous prenons les normes des deux côtés, nous trouvons :

$$|\delta \lambda| \quad \frac{\delta A2\,|}{y \cdot x|}$$

Donc, en général, le conditionnement du problème aux valeurs propres dépend de la taille de la perturbation δA - comme prévu - et de l'angle entre les vecteurs propres gauche et droit x et y. Nous pouvons utiliser 1/x·y comme nombre de condition approximatif. Notez que x = y lorsque A est symétrique, ce qui donne un nombre conditionnel de 1 ; cela reflète le fait que les vecteurs propres des matrices symétriques sont orthogonaux et donc séparés au maximum.

5.6 Problèmes

3Cette hypothèse devrait être vérifiée dans un traitement plus rigoureux !

Chapitre 6

Décomposition en valeurs singulières

Au chapitre 5, nous avons dérivé un certain nombre d'algorithmes pour calculer les valeurs propres et les vecteurs propres des matrices A Rn×n . Après avoir développé cette machinerie, nous complétons notre discussion initiale sur l'algèbre linéaire numérique en dérivant et en utilisant une factorisation matricielle finale qui existe pour toute matrice A Rm×n : la décomposition en valeurs singulières (SVD).

6.1 Dérivation de la SVD

Pour A Rm×n , nous pouvons penser à la fonction x → Ax comme une carte prenant des points dans Rn à des points dans Rm. De ce point de vue, on peut se demander ce qu'il advient de la géométrie de Rn dans le processus, et en particulier de l'effet de A sur les longueurs et les angles entre les vecteurs.

En appliquant notre point de départ habituel pour les problèmes aux valeurs propres, nous pouvons demander l'effet que A a sur les longueurs des vecteurs en examinant les points critiques du rapport

$$R(x) = \frac{\text{Hache}}{X}$$

sur différentes valeurs de x. La mise à l'échelle x n'a pas d'importance, puisque

$$R(\alpha x) = \frac{A \cdot \alpha x}{\alpha x} = \frac{|a|}{|a|} \cdot \frac{\text{Hache}}{X} = \frac{\text{Hache}}{X} = R(x).$$

Ainsi, nous pouvons restreindre notre recherche à x avec x = 1. De plus, puisque R(x) ≥ 0, nous pouvons plutôt considérer [R(x)]2 = Ax 2 = x A Ax. Or, comme nous l'avons montré dans les chapitres précédents, les points critiques de x A Ax soumis à x = 1 sont exactement les vecteurs propres xi vérifiant A Axi = λixi ; notons λi ≥ 0 et xi · xj = 0 quand i = j puisque AA est symétrique et semi-défini positif.

Sur la base de notre utilisation de la fonction R, la base {xi} est raisonnable pour étudier les effets géométriques de A. Revenant à cet objectif initial, définissons yi ≡ Axi . Nous pouvons faire une observation supplémentaire sur yi révélant une structure de valeurs propres encore plus forte :

$$\begin{aligned}
\lambda_i y_i &= \lambda_i \cdot Ax_i \text{ par définition de } y_i \\
&= A(\lambda_i x_i) \\
&= A(A\,Ax_i) \text{ puisque } x_i \text{ est un vecteur propre de } AA \\
&= (AA)(Ax_i) \text{ par associativité} \\
&= (AA)y_i
\end{aligned}$$

Ainsi, nous avons deux cas :

1. Lorsque λi = 0, alors yi = 0. Dans ce cas, xi est un vecteur propre de AA et yi = Axi est AAxi = $\sqrt{\lambda_i}$ un vecteur propre correspondant de AA avec yi = Axi = Axi $\sqrt{\lambda_i}$xi. $\|y_i\|_2 = \|x_i\|$

2. Lorsque λi = 0, yi =0.

Une preuve identique montre que si y est un vecteur propre de AA, alors x ≡ A y est soit nul soit un vecteur propre de AA de même valeur propre.

Soit k le nombre de valeurs propres strictement positives λi > 0 discuté ci-dessus. Par notre construction ci-dessus, nous pouvons prendre x1, . . . ,xk Rn comme étant les vecteurs propres de AA et les vecteurs propres correspondants y1, . . . ,yk Rm de AA tel que

$$A\,Ax = \lambda_i x_i$$

$$AAy_i = \lambda_i y_i$$

pour les valeurs propres λi > 0 ; ici on normalise tel que xi = yi = 1 pour tout i. Suivant la notation traditionnelle, on peut définir des matrices $\bar{V}$ Rn×k et $\bar{U}$ Rm×k dont les colonnes sont des xi et des yi, resp .

Nous pouvons examiner l'effet de ces nouvelles matrices de base sur A. Prenons ei comme le i-ème vecteur de base standard. Alors,

$$\begin{aligned}
\bar{U}A\bar{V}e_i &= \bar{U}Ax_i \text{ par définition de } \bar{V} \\
&= \frac{1}{\lambda_i}\bar{U}A(\lambda_i x_i) \text{ puisque nous avons supposé } \lambda_i > 0 \\
&= \frac{1}{\lambda_i}\bar{U}A(A\,Ax_i) \text{ puisque } x_i \text{ est un vecteur propre de AA } \lambda_i\ 1 \\
&= \frac{1}{\lambda_i}\bar{U}(AA)Ax_i \text{ par associativité } \lambda_i\ 1 \\
&= \frac{1}{\lambda_i}\bar{U}(AA)y_i \text{ puisque nous avons redimensionné pour que} \\
&\quad y_i = 1\ \sqrt{\lambda_i} = \lambda_i\bar{U}y_i \text{ puisque } AAy_i = \lambda_i y_i \\
&= \lambda_i e_i
\end{aligned}$$

Soit $\bar{\Sigma} = \mathrm{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_k})$. Alors, la dérivation ci-dessus montre que $\bar{U}A\bar{V} = \bar{\Sigma}$.

Compléter les colonnes de $\bar{U}$ et $\bar{V}$ en U Rm×m et V Rn×n en ajoutant les vecteurs orthonormés xi et yi avec A Axi =0 et AAyi =0, resp. Dans ce cas, il est facile de montrer UAVei =0 et/ou UAV =0 . Ainsi, si nous prenons e_j

$$\Sigma_{ij} \equiv \begin{cases} \sqrt{\lambda_i} & i = j \text{ et } i \le k \text{ sinon} \\ 0 & \end{cases}$$

alors nous pouvons étendre notre relation précédente pour montrer UAV = Σ, ou de manière équivalente

$$A = U\Sigma V .$$

Cette factorisation est exactement la décomposition en valeurs singulières (SVD) de A. Les colonnes de U couvrent l'espace des colonnes de A et sont appelées ses vecteurs singuliers gauches ; les colonnes de V s'étendent sur son espace de ligne et sont les bons vecteurs singuliers. Les éléments diagonaux σi de Σ sont les valeurs singulières de A ; elles sont généralement triées de telle sorte que σ1 ≥ σ2 ≥ · · · ≥ 0. U et V sont des matrices orthogonales.

Le SVD fournit une caractérisation géométrique complète de l'action de A. Puisque U et V sont orthogonaux, ils peuvent être considérés comme des matrices de rotation ; en tant que matrice diagonale, Σ met simplement à l'échelle les coordonnées individuelles. Ainsi, toutes les matrices A Rm×n sont composées d'une rotation, d'une échelle et d'une seconde rotation.

6.1.1 Calcul de la SVD

Rappelons que les colonnes de V sont simplement les vecteurs propres de AA, elles peuvent donc être calculées en utilisant les techniques discutées dans le chapitre précédent. Puisque A = UΣV, on sait AV = UΣ. Ainsi, les colonnes de U correspondant à des valeurs singulières non nulles dans Σ sont simplement des colonnes normalisées de AV ; les colonnes restantes satisfont AAui = 0, qui peut être résolu en utilisant la factorisation LU.

Cette stratégie n'est en aucun cas l'approche la plus efficace ou la plus stable pour calculer la SVD, mais elle fonctionne raisonnablement bien pour de nombreuses applications. Nous omettrons les approches plus spécialisées pour trouver le SVD, mais notons que beaucoup sont de simples extensions de l'itération de puissance et d'autres stratégies que nous avons déjà couvertes et qui fonctionnent sans former explicitement AA ou AA.

6.2 Applications du SVD

Nous consacrons la suite de ce chapitre à introduire de nombreuses applications de la SVD. Le SVD apparaît d'innombrables fois dans la théorie et la pratique de l'algèbre linéaire linéaire numérique, et son importance ne peut guère être exagérée.

6.2.1 Résolution de systèmes linéaires et de la pseudo-inverse

Dans le cas particulier où A Rn×n est carré et inversible, il est important de noter que la SVD peut être utilisée pour résoudre le problème linéaire Ax =b. En particulier, on a UΣV x =b, ou

$$x = V\Sigma^{-1}U\,b.$$

Dans ce cas Σ est une matrice diagonale carrée, donc Σ^{-1} est simplement la matrice dont les entrées diagonales sont $1/\sigma_i$.

Le calcul de la SVD est beaucoup plus coûteux que la plupart des techniques de résolution linéaire que nous avons présentées au chapitre 2, donc cette observation initiale est surtout d'intérêt théorique. Plus généralement, supposons que nous souhaitions trouver une solution des moindres carrés à Ax ≈ b, où A Rm×n n'est pas nécessairement carré. De notre discussion des équations normales, nous savons que x doit satisfaire A Ax = A b. Jusqu'à présent, nous avons pour la plupart ignoré le cas où A est "court" ou "sous-déterminé", c'est-à-dire lorsque A a plus de colonnes que de lignes. Dans ce cas, la solution des équations normales est non unique.

Pour couvrir les trois cas, on peut résoudre un problème d'optimisation de la forme suivante :

$$\begin{aligned} &\text{minimiser } \|x\|^2 \\ &\text{tel que } A^\top Ax = A^\top b \end{aligned}$$

En d'autres termes, cette optimisation demande que x satisfasse les équations normales avec le moins de norme possible.
Maintenant, écrivons A = UΣV . Alors,

$$\begin{aligned} AA &= (U\Sigma V)(U\Sigma V) \\ &= V\Sigma U\, U\Sigma V \text{ puisque } (AB) = BA \\ &= V\Sigma\, \Sigma V \text{ puisque U est orthogonal} \end{aligned}$$

Ainsi, demander que A Ax = A b revient au même que demander

$$V\Sigma\, \Sigma V x = V\Sigma U b$$

Ou de manière équivalente, $\Sigma y = d$

si on prend d ≡ Ub et y ≡ V x. Notez que y = x puisque U est orthogonal, donc notre optimisation devient :

$$\begin{aligned} &\text{minimiser } y \text{ tel} \quad {}^{2} \\ &\text{que } \Sigma y = d \end{aligned}$$

Puisque Σ est diagonal, cependant, la condition Σy = d indique simplement σiyi = di ; donc, chaque fois que σi = 0, nous devons avoir yi = di/σi . Lorsque σi = 0, il n'y a pas de contrainte sur yi , donc puisque nous minimisons y ^{2} autant prendre yi = 0. En d'autres termes, la solution à cette optimisation est y = Σ +d, où Σ + Rn×m a la forme suivante :

$$\Sigma^{+}_{ij} \equiv \quad 1/\sigma i \text{ i = j, } \sigma i = 0, \text{ et i} \leq k \text{ 0 sinon}$$

Cette forme donne à son tour x = Vy = VΣ +d = VΣ +Ub.
Forts de cette motivation, nous formulons la définition suivante :

Définition 6.1 (Pseudoinverse). La pseudo-inverse de A = UΣV Rm×n est A+ ≡ VΣ +U Rn×m.

Notre dérivation ci-dessus montre que la pseudo-inverse de A jouit des propriétés suivantes :

- -1 • Lorsque A est carré et inversible, A + $= UN$.
- • Lorsque A est surdéterminé, A +b donne la solution des moindres carrés à Ax ≈b.
- • Lorsque A est sous-déterminé, A +b donne la solution des moindres carrés à Ax ≈ b avec une norme (euclidienne).

De cette façon, nous sommes enfin capables d'unifier les cas sous-déterminés, pleinement déterminés et surdéterminés de Ax ≈b.

6.2.2 Décomposition en produits extérieurs et approximations de rang bas

Si on développe le produit A = UΣV , il est facile de montrer que cette relation implique :

$$UNE = \sum_{je=1} \sigma iuiv\, je ,$$

où ≡ min{m, n}, et ui et vi sont les ième colonnes de U et V, resp. Notre somme ne va qu'à min{m, n} puisque nous savons que les colonnes restantes de U ou V seront mises à zéro par Σ.

Cette expression montre que toute matrice peut être décomposée comme la somme des produits extérieurs de vecteurs :

Définition 6.2 (Produit extérieur). Le produit extérieur de u Rm et v Rn est la matrice u v ≡ uv Rm×n .

Supposons que nous souhaitions écrire le produit Ax. Alors, à la place, nous pourrions écrire :

$$Ax = \sum_{je=1} \sigma iuiv\ je\ x$$

$$= \sum_{je=1} \sigma iui(v \quad X)$$

$$= \sum_{je=1} \sigma i(vi \cdot x)ui \text{ puisque } x \cdot y = xy$$

Ainsi, appliquer A à x revient à combiner linéairement les vecteurs ui avec des poids σi(vi · x). Cette stratégie de calcul de Ax peut apporter des économies considérables lorsque le nombre de valeurs de σi non nulles est relativement faible. De plus, nous pouvons ignorer les petites valeurs de σi , tronquant effectivement cette somme pour approximer Ax avec moins de travail.

De même, à partir du §6.2.1, nous pouvons écrire la pseudo-inverse de A sous la forme :

$$A_{+} = \sum_{\sigma i=0} \frac{viu\ je}{\sigma i}.$$

Évidemment, nous pouvons appliquer la même astuce pour évaluer A +x, et en fait nous pouvons approximer A +x en n'évaluant que les termes de la somme pour lesquels σi est relativement petit. En pratique, nous calculons les valeurs singulières σi comme racines carrées des valeurs propres de AA ou AA, et des méthodes comme l'itération de puissance peuvent être utilisées pour révéler un ensemble partiel plutôt que complet de valeurs propres. Ainsi, si nous devons résoudre un certain nombre de problèmes de moindres carrés Axi ≈ bi pour différents bi et que nous nous contentons d'une approximation de xi , il peut être utile de calculer d'abord les plus petites valeurs de σi et d'utiliser l'approximation ci-dessus. Cette stratégie évite également d'avoir à calculer ou à stocker la matrice A + complète et peut être précise lorsque A a une large gamme de valeurs singulières.

Revenant à notre notation originale A = UΣV, notre argument ci-dessus montre effectivement qu'une approximation potentiellement utile de A est A˜ ≡ UΣ˜ V où Σ˜ arrondit les petites valeurs de Σ à zéro.
Il est facile de vérifier que l'espace colonne de A˜ a une dimension égale au nombre de valeurs non nulles sur la diagonale de Σ˜ En fait, cette approximation n'est pas une estimation ad hoc mais résout plutôt un problème d'optimisation difficile post par le fameux suivant théorème (énoncé sans preuve) :

Théorème 6.1 (Eckart-Young, 1936). Supposons que A soit obtenu à partir de A = UΣV en tronquant tous sauf les plus grandes valeurs singulières σi de A à zéro. Alors A minimise les A − A˜Fro et A − A˜2 sous réserve des k deux contraintes que l'espace colonne de A a au plus la dimension k.

6.2.3 Normes matricielles

La construction de la SVD nous permet également de revenir à notre discussion sur les normes matricielles du §3.3.1. Par exemple, rappelons que nous avons défini la norme de Frobenius de A comme

$$\|A\|_{Fro}^2 \equiv \sum_{ij} a_{ij}^2.$$

Si on écrit $A = U\Sigma V^T$, on peut simplifier cette expression :

$$\begin{aligned}
\|A\|_{Fro}^2 &= \sum_j \|Ae_j\|_2^2 \quad \text{puisque ce produit est la } j\text{ème colonne de } A \\
&= \sum_j \|U\Sigma V^T e_j\|_2^2, \quad \text{remplaçant le SVD} \\
&= \sum_j \|\Sigma V^T e_j\|_2^2 \quad \text{depuis } \|Qx\|_2^2 = x^T Q^T Q x = x^T x \text{ et } U \text{ est orthogonal} \\
&= \|\Sigma V^T\|_{Fro}^2 \quad \text{par la même logique} \\
&= \|V\Sigma\|_{Fro}^2 \quad \text{puisqu'une matrice et sa transposée ont la même norme de Frobenius} \\
&= \sum_j \|V\Sigma e_j\|_2^2 = \sum_j \|\sigma_j V e_j\|_2^2 \quad \text{par diagonalité de } \Sigma \\
&= \sum_j \sigma_j^2 \quad \text{puisque } V \text{ est orthogonal}
\end{aligned}$$

Ainsi, la norme de Frobenius de $A \in \mathbb{R}^{m\times n}$ est la somme des carrés de ses valeurs singulières.

Ce résultat est d'un intérêt théorique, mais en pratique la définition de base de la norme de Frobe nius est déjà simple à évaluer. Plus intéressant, rappelons que la double norme induite de A est donnée par

$$\|A\|_2^2 = \max\{\lambda : \text{il existe } x \in \mathbb{R}^n \text{ avec } A^T Ax = \lambda x\}.$$

Maintenant que nous avons étudié les problèmes de valeurs propres, nous réalisons que cette valeur est la racine carrée de la plus grande valeur propre de $A^T A$, ou de manière équivalente

$$\|A\|_2 = \max\{\sigma_i\}.$$

En d'autres termes, nous pouvons lire la norme à deux de A directement à partir de ses valeurs propres.

De même, rappelons que le nombre conditionnel de A est donné par $\text{cond } A = \|A\|_2\|A^{-1}\|_2$. Par nos doivent être les dérivation de A^+, les valeurs singulières de A^{-1} réciproques des valeurs singulières de A.
En combinant cela avec notre simplification des rendements $\|A\|_2$:

$$\text{cond } A = \frac{\sigma_{max}}{\sigma_{min}}.$$

Cette expression donne une stratégie pour évaluer le conditionnement de A. Bien entendu, le calcul de σ_{min} nécessite de résoudre des systèmes $Ax = b$, un processus qui en lui-même peut souffrir d'un mauvais conditionnement de A ; si cela pose problème, le conditionnement peut être borné et approché en utilisant diverses approximations des valeurs singulières de A.

6.2.4 Le problème de Procuste et l'alignement

De nombreuses techniques de vision par ordinateur impliquent l'alignement de formes tridimensionnelles. Par exemple, supposons que nous ayons un scanner tridimensionnel qui collecte deux nuages de points du même objet rigide à partir de vues différentes. Une tâche typique peut consister à aligner ces deux nuages de points dans un seul cadre de coordonnées.

Puisque l'objet est rigide, nous nous attendons à ce qu'il y ait une matrice de rotation R et une translationt R3 telles que la rotation du premier nuage de points de R puis la translation de byt aligne les deux ensembles de données. Notre travail consiste à estimer ces deux objets.

Si les deux balayages se chevauchent, l'utilisateur ou un système automatisé peut marquer n points correspondants qui correspondent entre les deux balayages ; on peut les stocker dans deux matrices X1, X2 R3×n . Alors, pour chaque colonne x1i de X1 et x2i de X2, on attend Rx1i +t = x2i . Nous pouvons écrire une fonction énergétique mesurant à quel point cette relation est vraie :

$$E \equiv \sum_{i} \left\| Rx1i + t - x2i \right\|^2.$$

Si on fixe R et qu'on minimise par rapport à tot, l'optimisation de E devient évidemment un problème de moindres carrés. Maintenant, supposons que nous optimisions pour R sans fixer. Cela revient à minimiser $\|RX1 - X\|_{Pour}^2$ où les t_2 ans, colonnes de X c'est-à-dire t_2 sont ceux de X2 translatés part, sous réserve que R soit une matrice de rotation 3 × 3, que RR = I3×3. C'est ce qu'on appelle le problème orthogonal de Procuste.

Pour résoudre ce problème, nous allons introduire la trace d'une matrice carrée comme suit :

Définition 6.3 (Trace). La trace de A Rn×n est la somme de sa diagonale :

$$\mathrm{tr}(A) \equiv \sum_{i} a_{ii}.$$

Il est simple de vérifier que $\|A\|_{Pour}^2 = \mathrm{tr}(A^t A)$. Ainsi, nous pouvons simplifier E comme suit :

$$\begin{aligned} \|RX1 - X_2^t\|_{Pour}^2 &= \mathrm{tr}((RX1 - X_2^t)^t (RX1 - X_2^t)) \\ &= \mathrm{tr}(X_1^t X_1 - X_1^t R^t X_2^t - X_2 R X_1 + X_2 X_2^t) \\ &= \text{const.} - 2\mathrm{tr}(X_2 R^t X_1) \\ &\quad \text{puisque } \mathrm{tr}(A + B) = \mathrm{tr}\, A + \mathrm{tr}\, B \text{ et } \mathrm{tr}(A^t) = \mathrm{tr}(A) \end{aligned}$$

Ainsi, on souhaite maximiser $\mathrm{tr}(X_2^t RX1)$ avec RR = I3×3. Dans les exercices, vous prouverez que tr(AB) = tr(BA). Ainsi notre objectif peut se simplifier légèrement en tr(RC) avec $C \equiv X1X_2^t$. En appliquant la SVD, si on décompose C = UΣV alors on peut simplifier encore plus :

$$\begin{aligned} \mathrm{tr}(RC) &= \mathrm{tr}(RU\Sigma V) \text{ par définition} = \mathrm{tr}((VRU)\Sigma) \text{ puisque } \mathrm{tr}(AB) = \mathrm{tr}(BA) \\ &= \mathrm{tr}(\tilde{R}\Sigma) \text{ si on définit } \tilde{R} = VRU, \text{ qui est aussi orthogonal} \\ &= \sum \sigma_i \tilde{r}_{ii} \text{ puisque } \Sigma \text{ est diagonal} \end{aligned}$$

Puisque $\tilde{R}$ est orthogonal, ses colonnes ont toutes une longueur unitaire. Ceci implique que $\tilde{r}_{ii} \le 1$, car sinon la norme de la colonne i serait trop grande. Puisque $\sigma_i \ge 0$ pour tout i, cet argument montre qu'on peut maximiser tr(RC) en prenant $\tilde{R} = I3\times3$. En annulant nos substitutions, on obtient $R = VR\tilde{U} = VU$.

Plus généralement, nous avons montré ce qui suit :

Théorème 6.2 (Procuste orthogonal). La matrice orthogonale R minimisant RX − Y VU, où SVD ^{2} est donné par est appliqué au facteur XY = UΣV .

Pour en revenir au problème d'alignement, une stratégie typique est une approche alternée :

1. Fixer R et minimiser E par rapport à tot.
2. Fixer la résultante et minimiser E par rapport à R sous réserve de RR = I3×3.
3. Revenez à l'étape 1.

L'énergie E diminue à chaque pas et converge donc vers un minimum local. Comme nous n'optimisons jamais et R simultanément, nous ne pouvons pas garantir que le résultat soit la plus petite valeur possible de E, mais en pratique cette méthode fonctionne bien.

6.2.5 Analyse en composantes principales (ACP)

Rappelons la configuration du §5.1.1 : Nous souhaitons trouver une approximation en basse dimension d'un ensemble de points de données, que nous pouvons stocker dans une matrice X Rn×k pour k observations en n dimensions. Précédemment, nous avons montré que si l'on n'admet qu'une seule dimension, la meilleure direction possible est donnée par le vecteur propre dominant de XX.

Supposons plutôt que nous soyons autorisés à projeter sur l'étendue de d vecteurs avec d ≤ min{k, n} et que nous souhaitions choisir ces vecteurs de manière optimale. On pourrait les écrire dans une matrice n × d C ; puisque nous pouvons appliquer Gram-Schmidt à n'importe quel ensemble de vecteurs, nous pouvons supposer que les colonnes de C sont orthonormées, montrant CC = Id×d . Puisque C a des colonnes orthonormées, par les équations normales, la projection de X sur l'espace des colonnes de C est donnée par CCX.

Dans cette configuration, on souhaite minimiser X − CCXFro sous réserve de CC = Id×d . On peut simplifier un peu notre problème :

$$\begin{aligned} \|X - CCX\|_{\text{Pour}}^2 &= \operatorname{tr}((X - CCX)^\top (X - CCX)) \text{ puisque } \|A\|_{\text{Pour}}^2 = \operatorname{tr}(A^\top A) \\ &= \operatorname{tr}(X^\top X - 2X^\top CCX + X^\top CCCCX) = \\ &\text{const.} - \operatorname{tr}(X^\top CCX) \text{ puisque } CC = I_{d\times d} \\ &= -\|CX\|_{\text{Pour}}^2 + \text{const.} \end{aligned}$$

Ainsi, de manière équivalente, nous pouvons ^{2}Bof ; pour les statisticiens, cela montre quand les lignes de X ont maximiser CX signifie zéro que nous souhaitons maximiser la variance de la projection C X.

Maintenant, supposons que nous factorisions X = UΣV . Alors, on souhaite maximiser C = C˜ΣFro = UΣV Fro Σ˜ CFro par orthogonalité de V si on prend C˜ = CU. Si les éléments de C˜ sont c˜ij, alors l'expansion de cette norme donne

$$\|\Sigma \tilde{C}\|_{\text{Pour}}^2 = \sum_{ij} \sigma_i^2 \tilde{c}_{ij}^2.$$

Par orthogonalité des colonnes de C˜, on sait ∑i c˜ = 1 pour tout j et, puisque C˜ peut avoir moins de n colonnes, $^2_{ij}$ ∑j c˜ vaut au plus 1 dans la somme ci-dessus, donc si on a alors clairement le maximum est atteint en prenant $^2_{ij} \le 1$. Ainsi, le coefficient à côté de σ_j^2 notre changement de , les colonnes de C˜ à trier telles que σ1 ≥ σ2 ≥ · · · bee1, . . . , éd . En annulant coordonnées, nous voyons que notre choix de C devrait être les d premières colonnes de U.

Nous avons montré que la SVD de X peut être utilisée pour résoudre un tel problème d'analyse en composantes principales (ACP). En pratique, les lignes de X sont généralement décalées pour avoir une moyenne nulle avant d'effectuer la SVD ; comme le montre la figure NUMBER, cela centre l'ensemble de données sur l'origine, fournissant des vecteurs PCA plus significatifs ui .

6.3 Problèmes

Partie III

Techniques non linéaires

Chapitre 7

Systèmes non linéaires

Malgré tous nos efforts, il n'est tout simplement pas possible d'exprimer tous les systèmes d'équations dans le cadre linéaire que nous avons développé au cours des derniers chapitres. Il n'est guère nécessaire de motiver l'utilisation des logarithmes, des exponentielles, des fonctions trigonométriques, des valeurs absolues, des polynômes, etc. dans des problèmes pratiques, mais sauf dans quelques cas particuliers, aucune de ces fonctions n'est linéaire. Lorsque ces fonctions apparaissent, nous devons employer une machinerie plus générale quoique moins efficace.

7.1 Problèmes à variable unique

Nous commençons notre discussion en considérant les problèmes d'une seule variable scalaire. En particulier, étant donnée une fonction f(x) : R → R, on souhaite développer des stratégies pour trouver des points x R tels que f(x) = 0 ; nous appelons x une racine de f . Les problèmes à une seule variable en algèbre linéaire ne sont pas intéressant; après tout, nous pouvons résoudre l'équation ax − b = 0 sous forme fermée comme particulièrement = x équation non linéaire comme$^{2 + e}$ b/a. Résoudre un cos y − 3 = 0, cependant, est beaucoup moins évident (d'ailleurs, la solution y est y = ±1.30246 . . .).

7.1.1 Caractérisation des problèmes

Nous ne pouvons plus supposer que f est linéaire, mais sans aucune hypothèse sur sa structure, il est peu probable que nous progressions dans la résolution de systèmes à une variable. Par exemple, un solveur est assuré de ne pas trouver les zéros de f(x) donné par

$$f(x) = \begin{array}{c} -1 \; x \leq 0 \\ 1 \; x > 0 \end{array}$$

Ou pire:

$$f(x) = \begin{array}{c} -1 \; x \; Q \; 1 \\ \text{sinon} \end{array}$$

Ces exemples sont triviaux en ce sens qu'un client rationnel d'un logiciel de recherche de racine ne s'attendrait probablement pas à ce qu'il réussisse dans ce cas, mais des cas beaucoup moins évidents ne sont pas beaucoup plus difficiles construire.

Pour cette raison, nous devons ajouter quelques hypothèses de « régularisation » sur le fait que f donne un aperçu de la possibilité de concevoir des techniques de recherche de racines. Ces hypothèses typiques sont présentées ci-dessous, classées par ordre croissant de force :

- Continuité : Une fonction f est continue si elle peut être dessinée sans lever la plume ; plus formellement, f est continue si la différence f(x) − f(y) s'annule lorsque x → y.
- Lipschitz : Une fonction f est Lipschitz continue s'il existe une constante C telle que | f(x) − f(y)| ≤ C|x − y| ; Les fonctions de Lipschitz n'ont pas besoin d'être différentiables mais sont limitées dans leurs taux de changement.
- Dérivabilité : Une fonction f est différentiable si sa dérivée f existe pour tout x.

toutes les dérivées de f k si elle est différentiable k fois et chacune de ces k dérivées est continue ; • C : Une fonction est C ∞ indique que C existent et sont continues.

Au fur et à mesure que nous ajoutons des hypothèses de plus en plus fortes sur f , nous pouvons concevoir des algorithmes plus efficaces pour résoudre f(x) = 0. Nous illustrerons cet effet en considérant quelques algorithmes ci-dessous.

7.1.2 Continuité et bissection

Supposons que tout ce que nous savons de f, c'est qu'il est continu. Dans ce cas, nous pouvons énoncer un théorème intuitif du calcul standard à une seule variable :

Théorème 7.1 (Théorème des valeurs intermédiaires). Supposons f : [a, b] → R est continue. Supposons que f(x) < u < f(y). Alors, il existe z entre x et y tel que f(z) = u.

En d'autres termes, la fonction f doit atteindre toutes les valeurs entre f(x) et f(y).

Supposons qu'on nous donne en entrée la fonction f ainsi que deux valeurs et r telles que f() · f(r) < 0 ; notez que cela signifie que f() et f(r) ont des signes opposés. Ensuite, par le théorème des valeurs intermédiaires, nous savons que quelque part entre et r il y a une racine de f ! Cela fournit une stratégie de bissection évidente pour trouver x

1. Calculer c = +r/2.
2. Si f(c) = 0, renvoie x = c.
3. Si f() · f(c) < 0, prendre r ← c. Sinon, prenez ← c.
4. Si |r − | < ε, renvoie x ≈ c.
5. Revenez à l'étape 1

Cette stratégie divise simplement l'intervalle [, r] en deux de manière itérative, en gardant à chaque fois le côté dans lequel une racine est connue pour exister. De toute évidence, d'après le théorème des valeurs intermédiaires, il converge inconditionnellement, en ce sens que tant que f() · f(r) < 0, les deux et r sont finalement garantis converger vers une racine valide x .

7.1.3 Analyse de la recherche de racine

La bissection est la technique la plus simple mais pas nécessairement la plus efficace pour trouver les racines. Comme avec la plupart des méthodes aux valeurs propres, la bissection est intrinsèquement itérative et peut ne jamais fournir de solution X exacte. Nous pouvons cependant nous demander à quel point la valeur ck de c à la k-ième itération est proche de la racine x que nous espérons calculer. Cette analyse fournira une base de comparaison avec d'autres méthodes.

En général, supposons que nous puissions établir une borne d'erreur Ek telle que l'estimation xk de la racine lors de la k-
X ième itération d'une méthode de recherche de racine vérifie |xk − x | < Ek . Évidemment tout algorithme avec Ek → 0
représente un schéma convergent ; la vitesse de convergence, cependant, peut être caractérisée par la vitesse à laquelle Ek
s'approche de 0. sont dans l'intervalle [k ,rk], une borne supérieure
sont données par Ek ≡ | rk − k |. Puisque nous divisons de Par exemple, en bissection puisque les erreurs ck et x
l'intervalle en deux à chaque itération, nous savons Ek+1 = 1/2Ek . Comme Ek+1 est linéaire dans Ek , on dit que la bissection
présente une convergence linéaire.

7.1.4 Itération en virgule fixe

La bissection est garantie de converger vers une racine pour toute fonction continue f , mais si nous en savons plus sur f, nous pouvons formuler des algorithmes qui peuvent converger plus rapidement.

Par exemple, supposons que nous souhaitions trouver x satisfaisant g(x) = x ; bien sûr, cette configuration est équivalente au problème de recherche de racine puisque résoudre f(x) = 0 revient à résoudre f(x) + x = x.
Cependant, comme information supplémentaire, nous pourrions également savoir que g est Lipschitz avec une constante C < 1.

Le système g(x) = x suggère une stratégie potentielle que nous pourrions émettre :

1. Prendre x0 comme estimation initiale d'une racine.
2. Itérer xk = g(xk−1).

Si cette stratégie converge, le résultat est clairement un point fixe de g satisfaisant aux critères ci-dessus.

Heureusement, la propriété de Lipschitz garantit que cette stratégie converge vers une racine s'il en existe une.
Si on prend Ek = |xk − x |, alors on a la propriété suivante :

$$
\begin{aligned}
Ek = |xk - x| &= |g(xk-1) - g(x)| \text{ par conception du schéma itératif et définition de } x \le C|xk-1 - x| \text{ puisque g est Lipschitz} \\
&= CEk-1
\end{aligned}
$$

L'application de cette déclaration de manière inductive montre que ^{k} |E0| → 0 comme k → ∞. Ainsi, l'itération en virgule fixe Ek ≤ C converge vers le x souhaité !

En fait, si g est Lipschitz avec constante C < 1 dans un voisinage [x − δ, x + δ], alors tant que x0 est choisi dans cet intervalle, l'itération en virgule fixe convergera. Ceci est vrai puisque notre expression pour Ek ci-dessus montre qu'elle se rétrécit à chaque itération. et |g (x)| < 1. Par

Un cas important se produit lorsque g est C^{1} sachant continuité de g dans ce cas, on + δ] où |g (x)| < 1 − ε pour tout x N, qu'il existe un voisinage N = [x − δ, x pour un choix d'un ε > 0.1
suffisamment petit Prenons n'importe quel x, y N. Alors, nous avons

$$
\begin{aligned}
|g(x) - g(y)| &= |g(\theta)| \cdot |x - y| \text{ par le théorème de la valeur moyenne du calcul de base, pour certains } \theta \in [x, y] \\
&< (1 - \varepsilon)|x - y|
\end{aligned}
$$

Cela montre que g est Lipschitz avec une constante 1 − ε < 1 dans N. Ainsi, lorsque g est continûment différentiable et g (x) < 1, l'itération en virgule fixe convergera vers x lorsque la supposition initiale x0 est proche.

1Cette affirmation est difficile à analyser : assurez-vous de bien la comprendre !

Jusqu'à présent, nous avons peu de raisons d'utiliser l'itération en virgule fixe : nous avons montré qu'elle est garantie de converger uniquement lorsque g est Lipschitz, et notre argument sur les Ek montre une convergence linéaire comme une bissection. Il existe cependant un cas où l'itération en virgule fixe offre un avantage.

Supposons que g soit différentiable avec g (x) = 0. Alors, le terme du premier ordre disparaît dans la série de Taylor pour g, laissant derrière :

$$1\ g(xk) = g(x\ ) + g\ (x\ )(xk - x\ )\ 2\ ^{2} + O\ (xk - x\ )\ ^{3}\ .$$

Ainsi, dans ce cas nous avons :

$$\begin{aligned} Ek &= |xk - x\ | \\ &= |g(xk-1) - g(x\ )| \text{ comme avant} \\ &= \tfrac{1}{1}\ |g\ (x\ )|(xk-1 - x\ )\ 2\ ^{2} + O((xk-1 - x\ )\ ^{3}) \text{ de l'argument de Taylor} \\ &\leq \bar{}\ (|g\ (x\ )| + \varepsilon)|(xk-1 - x\ )\ ^{2} \text{ pour un certain } \varepsilon \text{ tant que } xk-1 \text{ est proche de } x \\ &= \tfrac{2}{2}\ 1\ (|g\ (x\ )| + \varepsilon)E\ _{k-1}^{2} \end{aligned}$$

Ainsi, dans ce cas Ek est quadratique dans Ek−1 , on dit donc que l'itération en point fixe peut avoir une convergence quadratique ; notez que cette preuve de convergence quadratique n'est valable que parce que nous connaissons déjà Ek → 0 à partir de notre preuve de convergence plus générale. Cela implique que Ek → 0 beaucoup plus rapidement, nous aurons donc besoin de moins d'itérations pour atteindre une racine raisonnable.

Exemple 7.1 (Convergence d'itération en virgule fixe).

7.1.5 Méthode de Newton

Nous resserrons une fois de plus notre classe de fonctions pour dériver une méthode qui a une convergence quadratique plus cohérente. Maintenant, supposons à nouveau que nous souhaitions résoudre f(x) = 0, mais maintenant nous supposons $^{1,\ un}$ que f est C une condition légèrement plus stricte que Lipschitz.

En un point xk R, puisque f est maintenant différentiable on peut l'approximer en utilisant une droite tangente :

$$f(x) \approx f(xk) + f\ (xk)(x - xk)$$

La résolution de cette approximation pour f(x) ≈ 0 donne une racine

$$xk+1 = xk - \frac{f(xk)\ f}{(xk)}\ .$$

L'itération de cette formule est connue sous le nom de méthode de recherche de racine de Newton et revient à résoudre de manière itérative une approximation linéaire du problème non linéaire.

Remarquez que si nous définissons

$$g(x) = x - f\ (x)\ \frac{f(x)}{\ }\ ,$$

alors la méthode de Newton revient à itérer en point fixe sur g. En différenciant, on trouve : − f(x)f (x) par la

$$(x) = 1 - \frac{f\ (x)\ g\ ^{2}\ \text{règle du}}{2\ \text{quotient}\ f\ (x)\ f(x)f\ (x)}$$

$$= \frac{}{f\ (x)\ \ ^{2}}$$

itération en est une racine simple, ce qui signifie f (x) = 0. Alors, g (x) = 0, et par notre dérivation de Supposons x virgule fixe ci-dessus, nous savons que la méthode de Newton converge quadratiquement vers x pour un suffisamment proche estimation initiale. Ainsi, lorsque f est différentiable avec une racine simple, la méthode de Newton fournit une formule d'itération en virgule fixe dont la convergence quadratique est garantie ; quand x n'est pas simple, cependant, la est convergence peut être linéaire ou pire.

La dérivation de la méthode de Newton suggère d'autres méthodes dérivées en utilisant plus de termes dans la série de Taylor. Par exemple, la « méthode de Halley » ajoute des termes impliquant f aux itérations, et une classe de « méthodes de Householder » prend un nombre arbitraire de dérivées. Ces techniques offrent une convergence d'ordre encore plus élevé au prix d'avoir à évaluer des itérations plus complexes et la possibilité de modes de défaillance plus exotiques. D'autres méthodes remplacent les séries de Taylor par d'autres formes de base ; par exemple, l'interpolation fractionnaire linéaire utilise des fonctions rationnelles pour mieux approximer les fonctions avec une structure asymptote.

7.1.6 Méthode sécante

Un problème d'efficacité que nous n'avons pas encore abordé est le coût de l'évaluation de f et de ses dérivées.
Si f est une fonction très compliquée, nous pouvons souhaiter minimiser le nombre de fois que nous devons calculer f ou pire f . Des ordres de convergence plus élevés aident à résoudre ce problème, mais nous pouvons également concevoir des méthodes numériques qui évitent d'évaluer des dérivées coûteuses.

Exemple 7.2 (Conception). Supposons que nous concevions une fusée et que nous souhaitions connaître la quantité de carburant à ajouter au moteur. Pour un nombre de gallons x donné, on peut écrire une fonction f(x) donnant la hauteur maximale de la fusée ; nos ingénieurs ont précisé que nous souhaitons que la fusée atteigne une hauteur h, nous devons donc résoudre f(x) = h. L'évaluation de f(x) implique de simuler une fusée lors de son décollage et de surveiller sa consommation de carburant, ce qui est une proposition coûteuse, et bien que nous puissions soupçonner que f est différentiable, nous pourrions ne pas être en mesure d'évaluer f dans un laps de temps pratique.

Une stratégie pour concevoir des méthodes à faible impact consiste à réutiliser les données autant que possible. Pour exemple, nous pourrions facilement approximer :

$$\approx \quad \frac{f(xk) - f(xk-1)\ f\ (xk)}{xk - xk-1}.$$

Autrement dit, puisque nous avons dû calculer f(xk−1) dans l'itération précédente, nous utilisons simplement la pente de f(xk) pour approximer la dérivée. Certes, cette approximation fonctionne bien, en particulier lorsque les xk sont proches de la convergence.

Brancher notre approximation dans la méthode de Newton révèle un nouveau schéma itératif :

$$xk+1 = xk - \frac{f(xk)(xk - xk-1)}{f(xk) - f(xk-1)}$$

Notez que l'utilisateur devra fournir deux suppositions initiales x0 et x−1 pour démarrer ce schéma, ou peut exécuter une seule itération de Newton pour le démarrer.

L'analyse de la méthode de la sécante est un peu plus compliquée que les autres méthodes que nous considérons car elle utilise à la fois f(xk) et f(xk−1) ; la preuve de sa convergence sort du cadre de notre discussion. Fait intéressant, l'analyse des erreurs révèle que l'erreur diminue à un taux de $1+\sqrt{5}/2$ (le « nombre d'or »), entre linéaire et quadratique ; puisque la convergence est proche de celle de la méthode de Newton sans qu'il soit nécessaire d'évaluer f , la méthode de la sécante peut fournir une alternative solide.

7.1.7 Techniques hybrides

Une ingénierie supplémentaire peut être effectuée pour tenter de combiner les avantages de différents algorithmes de recherche de racine. Par exemple, nous pourrions faire les observations suivantes sur deux méthodes dont nous avons discuté :

- La bissection est garantie de converger inconditionnellement mais ne le fait qu'à un taux linéaire.
- La méthode de la sécante converge plus rapidement lorsqu'elle atteint une racine, mais dans certains cas, elle peut ne pas converger.

Supposons que nous ayons encadré une racine de f(x) dans un intervalle [k ,rk] comme dans la bissection. On peut dire que quand | gardons une trace de xk et xk−1 , est donné par xk = k f(k)| < | f(rk)| et xk = rk sinon. Si notre estimation actuelle de x nous alors nous pourrions prendre xk+1 comme la prochaine estimation de la racine donnée par la méthode de la sécante. Si xk est en dehors de l'intervalle [k ,rk], cependant, nous pouvons le remplacer par k+rk/2. Cette correction garantit que xk+1 [k ,rk], et quel que soit le choix nous pouvons mettre à jour vers une parenthèse valide [k+1 ,rk+1] comme dans la bissection en examinant le signe de f(xk+1). Cet algorithme est connu sous le nom de « méthode de Dekker ».

La stratégie ci-dessus tente de combiner la convergence inconditionnelle de la bissection avec les estimations de racine plus fortes de la méthode sécante. Dans de nombreux cas, il réussit, mais son taux de convergence est quelque peu difficile à analyser ; des modes de défaillance spécialisés peuvent réduire cette méthode à une convergence linéaire ou pire - en fait, dans certains cas, la bissection peut étonnamment converger plus rapidement ! D'autres techniques, par exemple la « méthode de Brent », font plus souvent des étapes de bissection pour éviter ce cas et peuvent présenter un comportement garanti au prix d'une mise en œuvre un peu plus complexe.

7.1.8 Cas à variable unique : résumé

Nous avons maintenant présenté et analysé un certain nombre de méthodes pour résoudre f(x) = 0 dans le cas à une seule variable. Il est probablement évident à ce stade que nous n'avons fait qu'effleurer la surface de ces techniques ; de nombreux schémas itératifs de recherche de racine existent, tous avec des garanties, des taux de convergence et des mises en garde différents. Quoi qu'il en soit, à travers nos expériences, nous pouvons faire un certain nombre d'observations :

- En raison de la forme générique possible de f , il est peu probable que nous puissions trouver exactement les racines x et nous nous contentons plutôt de schémas itératifs.
- On souhaite que la suite xk d'estimations racine atteigne x aussi vite que possible. Si Ek est une borne d'erreur, alors nous pouvons caractériser un certain nombre de situations de convergence en supposant Ek → 0 comme k → ∞. Une liste complète des conditions qui doivent être remplies lorsque k est suffisamment grand est ci-dessous :
  1. Convergence linéaire : Ek+1 ≤ CEk pour un certain C < 1 2.
  Convergence superlinéaire : Ek+1 ≤ CEr pour r > 1 (maintenant nous n'exigeons pas C < 1 car si Ek est assez petit, la k puissance r peut s'annuler les effets de C)
  3. Convergence quadratique : Ek+1 ≤ CE2 4.
  Convergence cubique : Ek+1 ≤ CE3 k (et ainsi de suite)
- Une méthode peut converger plus rapidement mais à chaque itération individuelle nécessiter des calculs supplémentaires ; pour cette raison, il peut être préférable de faire plus d'itérations d'une méthode plus simple que moins d'itérations d'une méthode plus complexe.

7.2 Problèmes multivariables

Certaines applications peuvent nécessiter la résolution d'un problème plus général f(x) =0 pour une fonction f : Rn → Rm. Nous avons déjà vu un exemple de ce problème lors de la résolution de Ax =b, ce qui équivaut à trouver les racines de f(x) ≡ Ax −b, mais le cas général est considérablement plus difficile. En particulier, les stratégies comme la bissection sont difficiles à étendre puisque nous garantissons désormais largement que m valeurs différentes sont toutes nulles simultanément.

7.2.1 Méthode de Newton

Heureusement, l'une de nos stratégies s'étend de manière simple. Rappelons que pour f : Rn → Rm on peut écrire la matrice jacobienne, qui donne la dérivée de chaque composante de f dans chacune des directions de coordonnées :

$$(D\,f)_{ij} \equiv \frac{d\,f_i}{dx_j}$$

Nous pouvons utiliser le jacobien de f pour étendre notre dérivation de la méthode de Newton à plusieurs dimensions. En particulier, l'approximation au premier ordre de f est donnée par :

$$f(x) \approx f(x_k) + ré\, f(x_k) \cdot (x - x_k).$$

En substituant le f(x) =0 souhaité , on obtient le système linéaire suivant pour l'itération suivante xk+1 :

$$D\,f(x_k) \cdot (x_{k+1} - x_k) = -f(x_k)$$

Cette équation peut être résolue en utilisant la pseudo-inverse lorsque m < n ; quand m > n on peut tenter les moindres carrés mais l'existence d'une racine et la convergence de cette technique sont toutes deux peu probables. Lorsque D f est carré, cependant, correspondant à la méthode , on obtient l'itération typique de Newton f : Rn → Rn :

$$x_{k+1} = x_k - [D\,f(x_k)]^{-1} f(x_k), \text{ où}$$

comme toujours nous ne calculons pas explicitement la matrice [D f(xk)]−1 mais l'utilisons plutôt pour signaler la résolution d'un système linéaire.

La convergence des méthodes à virgule fixe comme la méthode de Newton qui itèrent xk + 1 = g (xk) nécessite que la valeur propre d'amplitude maximale du jacobien Dg soit inférieure à 1. Après avoir vérifié cette hypothèse, un argument similaire au cas unidimensionnel montre que la méthode de Newton peut pour laquelle D f(x) est non quadratique près des racines x singulière. avoir une convergence

7.2.2 Rendre Newton plus rapide : Quasi-Newton et Broyen

Lorsque m et n augmentent, la méthode de Newton devient très coûteuse. A chaque itération, une matrice D f(xk) différente doit être inversée ; parce qu'il change si souvent, pré-factoriser D f(xk) = LkUk n'aide pas.

Certaines stratégies quasi-Newton tentent d'appliquer différentes stratégies d'approximation pour simplifier les itérations individuelles. Par exemple, une approche directe pourrait réutiliser D f des itérations précédentes tout en recalculant f(xk) sous l'hypothèse que la dérivée ne change pas très rapidement. Nous reviendrons sur ces stratégies lorsque nous discuterons de l'application de la méthode de Newton à l'optimisation.

Une autre option consiste à tenter de mettre en parallèle notre dérivation de la méthode de la sécante. De même que la méthode de la sécante contient toujours la division, de telles approximations ne supprimeront pas nécessairement la nécessité d'inverser une matrice, mais elles permettent d'effectuer une optimisation sans calculer explicitement le Jacobien D f . De telles extensions ne sont pas totalement évidentes, puisque les différences divisées ne produisent pas une matrice jacobienne approximative complète.

Rappelons cependant que la dérivée directionnelle de f dans la direction v est donnée par Dv f = D f · v. Comme pour la méthode sécante, nous pouvons utiliser cette observation à notre avantage en demandant que notre approximation J d'un Jacobien satisfasse

$$J \cdot (x_k - x_{k-1}) \approx f(x_k) - f(x_{k-1}).$$

La méthode de Broyden est une telle extension de la méthode sécante qui garde trace non seulement d'une estimation xk de x mais aussi d'une matrice Jk estimant le jacobien ; les estimations initiales J0 et x0 doivent toutes deux être fournies. Supposons que nous ayons une estimation précédente Jk−1 du Jacobien de l'itération précédente. Nous avons maintenant un nouveau point de données xk auquel nous avons évalué f(xk), nous voudrions donc mettre à jour Jk−1 vers un nouveau jacobien Jk en tenant compte de cette nouvelle observation. Un modèle raisonnable consiste à demander que la nouvelle approximation soit aussi similaire à l'ancienne sauf dans la direction xk − xk−1 :

$$\begin{aligned} &\text{minimiser}_{J_k} \|J_k - J_{k-1}\|_{\text{Pour}}^2 \text{ tel} \\ &\quad \text{que } J_k \cdot (x_k - x_{k-1}) = f(x_k) - f(x_{k-1}) \end{aligned}$$

Pour résoudre ce problème, définissons $\Delta J \equiv J_k - J_{k-1}$, $d \equiv f(x_k) - f(x_{k-1}) - J_{k-1} \cdot \Delta x$. $\Delta x \equiv x_k - x_{k-1}$ ces substitutions donne la forme suivante :

$$\begin{aligned} &\text{minimiser}_{\Delta J} \|\Delta J\|_{\text{Pour}}^2 \text{ tel} \\ &\quad \text{que } \Delta J \cdot \Delta x = d \end{aligned}$$

Si l'on prend λ comme multiplicateur de Lagrange, cette minimisation revient à trouver les points critiques du Lagrangien Λ :

$$\Lambda = \|\Delta J\|_{\text{Pour}}^2 + \lambda (\Delta J \cdot \Delta x - d)$$

La différenciation par rapport à $(\Delta J)_{ij}$ montre :

$$0 = \frac{\partial \Lambda}{\partial (\Delta J)_{ij}} = 2(\Delta J)_{ij} + \lambda_i (\Delta x)_j \quad \Delta J = -\frac{1}{2} \lambda (\Delta x)$$

La substitution dans $\Delta J \cdot \Delta x = d$ montre $\lambda(\Delta x)(\Delta x) = -2d$, ou de façon équivalente $\lambda = -2d/\Delta x^2$. Enfin, on peut substituer pour trouver :

$$\Delta J = -\frac{1}{2} \lambda (\Delta x) = \Delta x \frac{d(\Delta x)}{\Delta x^2}$$

L'expansion de nos spectacles de substitution :

$$\begin{aligned} J_k &= J_{k-1} + \Delta J \\ &= J_{k-1} + \frac{d(\Delta x)}{\Delta x^2} \\ &= J_{k-1} + \frac{(f(x_k) - f(x_{k-1}) - J_{k-1} \cdot \Delta x)}{\|x_k - x_{k-1}\|^2} (x_k - x_{k-1}) \end{aligned}$$

Ainsi, la méthode de Broyden alterne simplement entre cette mise à jour et le Newton f(xk) correspondant. Une efficacité pas xk+1 = xk − J_k^{-1} supplémentaire dans certains cas peut être obtenue en gardant une trace de la matrice J_k^{-1} explicitement plutôt que la matrice Jk , qui peut être mise à jour à l'aide d'une formule similaire.

7.3 Conditionnement

Nous avons déjà montré dans l'exemple 1.7 que le nombre conditionnel de recherche de racine dans une seule variable est :

$$\text{condx} \quad f = \frac{1}{|f(x)|}$$

Comme l'illustre la figure NUMBER, ce numéro de condition montre que la meilleure situation possible pour la recherche de racine se produit lorsque f change rapidement près de x puisque dans ce cas, perturber x fera prendre à f des valeurs éloignées de 0.

L'application d'un argument identique lorsque f est multidimensionnelle montre un nombre conditionnel de $D f(x)^{-1}$. Notez que lorsque D f n'est pas inversible, le nombre conditionnel est infini. Cette bizarrerie préserve f(x) = se produit parce que la perturbation de premier ordre 0, et en effet une telle condition peut de x crée des cas difficiles de recherche de racine comme celui illustré à la figure NUMBER.

7.4 Problèmes

De nombreuses possibilités, dont :

- De nombreux schémas d'itération en virgule fixe possibles pour un problème de recherche de racine donné, version graphique de l'itération en virgule fixe
- Itération de champ moyen en ML
- Méthode de Muller - racines complexes
- Méthodes itératives d'ordre supérieur - Méthodes Householder
- Interprétation des éléments propres comme recherche de racine
- Convergence de la méthode sécante
- Racines de polynômes
- Méthode Newton-Fourier (!)
- "Méthode de Newton modifiée en cas de convergence non quadratique"
- Convergence –¿ rayon spectral pour Newton multidimensionnel ; convergence quadratique
- Mise à jour Sherman-Morrison pour Broyden

Chapitre 8

Optimisation sans contrainte

Dans les chapitres précédents, nous avons choisi d'adopter une approche largement variationnelle pour dériver des algorithmes standards pour l'algèbre linéaire computationnelle. Autrement dit, nous définissons une fonction objectif, éventuellement avec des contraintes, et posons nos algorithmes comme un problème de minimisation ou de maximisation. Un échantillon de notre discussion précédente est répertorié ci-dessous :

Problème	Objectif	Contraintes
Moindres carrés	$E(x) = Ax - b$	Aucun
Projeter sur a	$E(c) = ca - b$	Aucun
Vecteurs propres de la matrice symétrique A	$E(x) = x^\top Ax$	$\|x\| = 1$
Pseudoinverse	$E(x) = x$	$A^\top Ax = A^\top b$
Analyse en composantes principales	$E(C) = X - CC^\top X\|_{Fro}$ où $C^\top C = Id_{k\times d}$	
Étape de Broyden	$E(J_k) = J_k - J_{k-1}$ Pour	$J_k \cdot (x_k - x_{k-1}) = f(x_k) - f(x_{k-1})$

Évidemment, la formulation des problèmes de cette manière est une approche puissante et générale. Pour cette raison, il est intéressant de concevoir des algorithmes qui fonctionnent en l'absence d'une forme spéciale pour l'énergie E, de la même manière que nous avons développé des stratégies pour trouver des racines de f sans connaître la forme de f a priori.

8.1 Optimisation sans contrainte : Motivation

Dans ce chapitre, nous allons considérer des problèmes non contraints, c'est-à-dire des problèmes qui peuvent être posés comme minimisant ou maximisant une fonction $f : \mathbb{R}^n \to \mathbb{R}$ sans aucune exigence sur l'entrée. Il n'est pas difficile de rencontrer de tels problèmes dans la pratique ; nous énumérons quelques exemples ci-dessous.

Exemple 8.1 (Moindres carrés non linéaires). Supposons qu'on nous donne un certain nombre de paires (x_i, y_i) telles que $f(x_i) \approx y_i$, et que nous souhaitions trouver la meilleure approximation de f dans une classe particulière. Par exemple, on peut s'attendre à ce que f soit exponentielle, auquel cas on devrait pouvoir écrire $f(x) = ce^{ax}$ pour certains c et certains a ; notre travail consiste à trouver ces paramètres. Une stratégie simple pourrait consister à tenter de minimiser l'énergie suivante :

$$E(a, c) = \sum_i (y_i - ce^{ax_i})^2.$$

Cette forme de E n'est pas quadratique en a, donc nos méthodes linéaires des moindres carrés ne s'appliquent pas.

Exemple 8.2 (Estimation du maximum de vraisemblance). En apprentissage automatique, le problème de l'estimation des paramètres consiste à examiner les résultats d'une expérience randomisée et à essayer de les résumer à l'aide d'une distribution de probabilité d'une forme particulière. Par exemple, nous pouvons mesurer la taille de chaque élève d'une classe, ce qui donne une liste de hauteurs hi pour chaque élève i. Si nous avons beaucoup d'élèves, nous pourrions modéliser la distribution des tailles d'élèves à l'aide d'une distribution normale :

$$g(h; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(h-\mu)^2/2\sigma^2},$$

où μ est la moyenne de la distribution et σ est l'écart type.

Sous cette distribution normale, la probabilité que nous observions la taille hi pour l'élève i est donnée par g(hi ; μ, σ), et sous l'hypothèse (raisonnable) que la taille de l'élève i est probabiliste indépendante de celle de l'élève j, la probabilité d'observer l'ensemble des hauteurs observées est donnée par le produit

$$P(\{h1, \ldots, hn\}; \mu, \sigma) = \prod_i g(hi; \mu, \sigma).$$

Une méthode courante pour estimer les paramètres μ et σ de g est de maximiser P vu comme une fonction de μ et σ avec {hi} fixé ; c'est ce qu'on appelle l'estimation du maximum de vraisemblance de μ et σ. En pratique, on optimise généralement le log vraisemblance (μ, σ) ≡ log P({h1, . . . , hn}; μ, σ) ; cette fonction a les mêmes maxima mais bénéficie de meilleures propriétés numériques et mathématiques.

Exemple 8.3 (Problèmes géométriques). De nombreux problèmes de géométrie rencontrés en graphisme et en vision ne se réduisent pas aux énergies des moindres carrés. Par exemple, supposons que nous ayons un certain nombre de points x1, . . . ,xk R3 . Si nous souhaitons regrouper ces points, nous pourrions souhaiter les résumer avec un seul x minimisant :

$$E(x) \equiv \sum_i x - xi2.$$

Le x R3 minimisant E est connu comme la médiane géométrique de {x1, . . . ,xk}. Remarquez que la norme de la différence x −xi dans E n'est pas au carré, donc l'énergie n'est plus quadratique dans les composantes de x.

Exemple 8.4 (Équilibres physiques, adapté du CITE). Supposons que nous attachions un objet à un ensemble de ressorts ; chaque ressort est ancré au point xi R3 et a une longueur propre Li et une constante ki . En l'absence de gravité, si notre objet est situé à la position p R3 le réseau de sources a de l'énergie potentielle

$$E(p) = \frac{1}{2} \sum_i ki\,(p - xi2 - Li)^2$$

Les équilibres de ce système sont donnés par des minima de E et reflètent les points p auxquels les forces du ressort sont toutes équilibrées. De tels systèmes d'équations sont utilisés pour visualiser les graphes G = (V, E), en attachant des sommets dans V avec des ressorts pour chaque paire dans E.

8.2 Optimalité

Avant de discuter de la manière de minimiser ou de maximiser une fonction, nous devons être clairs sur ce que nous recherchons ; notez que maximiser f est la même chose que minimiser −f , donc le problème de minimisation est suffisant pour notre considération. Pour un f particulier : Rn → R et x Rn nous devons en possible f(x). conditions d'optimalité déduire que a la plus petite valeur

qui vérifient que x Bien sûr, idéalement on aimerait trouver des optima globaux de f :

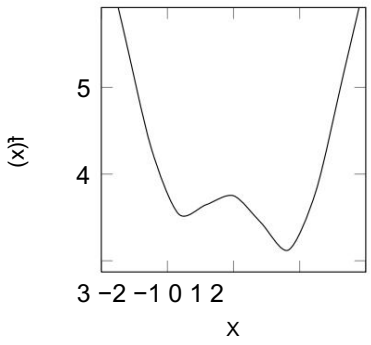


Figure 8.1 : Une fonction f(x) avec plusieurs optima.

Définition 8.1 (minimum global). Le point x f(x) ≤ Rn est un minimum global de f : Rn → R si f(x) pour tout x Rn .

Trouver un minimum global de f sans aucune information sur la structure de f nécessite effectivement une recherche dans l'obscurité. Par exemple, supposons qu'un algorithme d'optimisation identifie le minimum local près de x = −1 dans la fonction de la figure 8.1. Il est presque impossible de réaliser qu'il existe un deuxième minimum inférieur près de x = 1 simplement en devinant les valeurs de x - pour autant que nous sachions, il peut y avoir un troisième minimum encore inférieur de f à x = 1000 !

Ainsi, dans de nombreux cas on se satisfait en trouvant un minimum local :

Définition 8.2 (Minimum local). Le point x f(x) pour Rn est un minimum local de f : Rn → R si f(x) ≤ < ε tout x Rn satisfaisant x −x pour un certain ε > 0.

exige que x rayon ε. Notez que les atteint la plus petite valeur dans un voisinage défini par la Cette définition algorithmes d'optimisation locale ont une limitation sévère en ce sens qu'ils ne peuvent pas garantir qu'ils produisent la valeur la plus basse possible de f , comme dans la figure 8.1 si le minimum local gauche est atteint ; de nombreuses stratégies, heuristiques et autres, sont appliquées pour explorer le paysage des valeurs x possibles pour aider à acquérir la confiance qu'un minimum local a la meilleure valeur possible.

8.2.1 Optimalité différentielle

Une histoire familière du calcul à une et plusieurs variables est que trouver des minima et des maxima potentiels d'une fonction f : Rn → R est plus simple lorsque f est différentiable. Rappelons que le vecteur gradient f = (∂ f/∂x1, . . . , ∂ f/∂xn) pointe dans la direction dans laquelle f augmente le plus ; le vecteur − f pointe dans la direction de plus grande décroissance. Une façon de voir cela est de se rappeler que près de a , f ressemble au point de la fonction linéaire x0 Rn

$$f(x) \approx f(x0) + \quad f(x0) \cdot (x - x0).$$

Si on prend x −x0 = α f(x0), alors on trouve :

$$f(x0 + \alpha \quad f(x0)) \approx f(x0) + \alpha \quad f(x0)^{2}$$

Lorsque f(x0) > 0, le signe de α détermine si f augmente ou diminue.

Il n'est pas difficile de formaliser l'argument ci-dessus pour montrer que si x0 est un minimum local, alors on doit avoir f(x0) =0. Notez que cette condition est nécessaire mais pas suffisante : maxima et selle

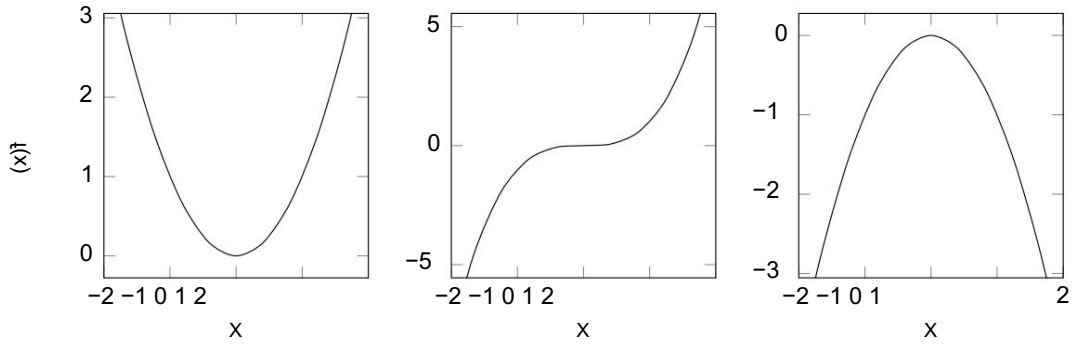


Figure 8.2 : Les points critiques peuvent prendre plusieurs formes ; nous montrons ici un minimum local, un point de selle et un maximum local.

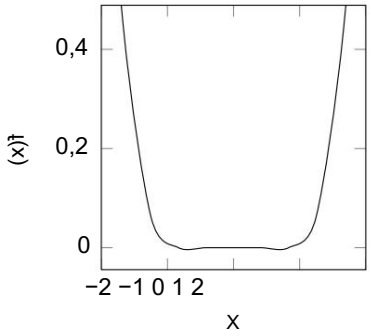


Figure 8.3 : Une fonction avec de nombreux points stationnaires.

points ont aussi f(x0) = 0 comme illustré sur la Figure 8.2. Même ainsi, cette observation sur les minima de fonctions différentiables donne une stratégie commune pour trouver la racine :

1. Trouver les points xi satisfaisant f(xi) =0.

2. Vérifiez lequel de ces points est un minimum local par opposition à un maximum ou un point de selle.

Compte tenu de leur rôle important dans cette stratégie, nous donnons aux points que nous recherchons un nom particulier :

Définition 8.3 (Point stationnaire). Un point stationnaire de f : Rn → R est un point x Rn vérifiant f(x) =0.

Autrement dit, notre stratégie de minimisation peut consister à trouver des points stationnaires de f puis à éliminer ceux qui ne sont pas des minima.

Il est important de garder à l'esprit quand nous pouvons nous attendre à ce que nos stratégies de minimisation réussissent. Dans la plupart des cas, comme ceux de la figure 8.2, les points stationnaires de f sont isolés, ce qui signifie qu'on peut les écrire dans une liste discrète {x0,x1, . . .}. Un cas dégénéré, cependant, est illustré à la figure 8.3 ; ici, tout l'intervalle [−1/2, 1/2] est composé de points fixes, ce qui rend impossible de les considérer un par un. Pour la plupart, nous ignorerons les problèmes de cas dégénérés, mais nous y reviendrons lorsque nous considérerons le conditionnement du problème de minimisation.

Supposons que nous identifions un point x R comme un point stationnaire de f et que nous souhaitions maintenant vérifier s'il s'agit d'un minimum local. Si f est deux fois différentiable, une stratégie que nous pouvons employer est d'écrire son hessien

matrice:

$$Hf(x) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} & \frac{\partial^2}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n} \\ \frac{\partial^2}{\partial x_2 \partial x_1} & \frac{\partial^2}{\partial x_2^2} & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} & \frac{\partial^2}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2}{\partial x_n^2} \end{bmatrix}$$

Nous pouvons ajouter un autre terme à notre développement de Taylor de f pour voir le rôle de Hf :

$$f(x) \approx f(x_0) + f(x_0) \cdot (x - x_0) + \frac{1}{2} (x - x_0) Hf(x - x_0)$$

Si on substitue un point fixe x , alors par définition on sait :

$$f(x) \approx f(x) + (x - x)^T Hf(x - x) \frac{1}{2}$$

Si Hf est définie positive, alors cette expression montre f(x) ≥ f(x), et donc x est un minimum local.
Plus généralement, une des quelques situations peut se produire :

- Si Hf est définie positive, alors x est un minimum local de f .
- Si Hf est définie négative, alors x est un maximum local de f .
- Si Hf est indéfini, alors x est un point selle de f .
- Si Hf n'est pas inversible, des bizarreries telles que la fonction de la figure 8.3 peuvent se produire.

Vérifier si une matrice est définie positive peut être accompli en vérifiant si sa factorisation de Cholesky existe ou - plus lentement - en vérifiant que toutes ses valeurs propres sont positives. Ainsi, lorsque le hessien de f est connu, nous pouvons vérifier l'optimalité des points stationnaires en utilisant la liste ci-dessus; de nombreux algorithmes d'optimisation, y compris ceux dont nous parlerons, ignorent simplement le cas final et informent l'utilisateur, car il est relativement peu probable.

8.2.2 Optimalité via les propriétés de la fonction

Occasionnellement, si nous connaissons plus d'informations sur f : Rn → R , nous pouvons fournir des conditions d'optimalité plus fortes ou plus faciles à vérifier que celles ci-dessus.

Une propriété de f qui a de fortes implications pour l'optimisation est la convexité, illustrée dans la figure NUMBER :

Définition 8.4 (Convexe). Une fonction f : Rn → R est convexe lorsque pour tout x,y Rn et α (0, 1) la relation suivante est vérifiée :

$$f((1 - \alpha)x + \alpha y) \leq (1 - \alpha)f(x) + \alpha f(y).$$

Lorsque l'inégalité est stricte, la fonction est strictement convexe.

La convexité implique que si vous reliez dans Rn deux points par une droite, les valeurs de f le long de la droite sont inférieures ou égales à celles que vous obtiendriez par interpolation linéaire.

Les fonctions convexes bénéficient de nombreuses propriétés fortes, dont la plus fondamentale est la suivante :

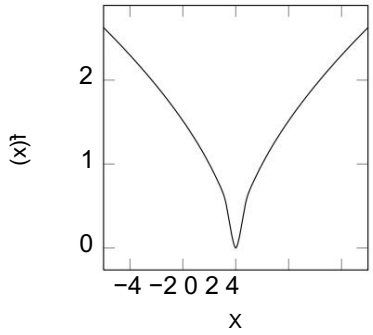


Figure 8.4 : Une fonction quasiconvexe.

Proposition 8.1. Un minimum local d'une fonction convexe f : Rn → R est nécessairement un minimum global.

Preuve. Soit x un tel minimum local et supposons qu'il existe x Alors, pour α (0, = x avec f(x) < f(x). 1),

$$f(x + \alpha(x - x)) \leq (1 - \alpha)f(x) + \alpha f(x) \text{ par convexité}$$
$$< f(x) \text{ puisque } f(x) < f(x)$$

Mais prendre α → 0 montre que x ne peut pas être un minimum local. □

Cette proposition et les observations associées montrent qu'il est possible de vérifier si vous avez atteint un minimum global d'une fonction convexe simplement en appliquant l'optimalité du premier ordre. Ainsi, il est intéressant de vérifier à la main si une fonction en cours d'optimisation se trouve être convexe, situation qui se présente étonnamment souvent en calcul scientifique ; une condition suffisante qui peut être plus facile à vérifier lorsque f est deux fois différentiable est que Hf est définie positive partout.

D'autres techniques d'optimisation ont des garanties sous d'autres hypothèses sur f . Pour l'examen ple, une version plus faible de la convexité est la quasi-convexité, obtenue lorsque

$$f((1 - \alpha)x + \alpha y) \leq \max(f(x), f(y)).$$

Un exemple de fonction quasiconvexe est illustré à la Figure 8.4 ; bien qu'elle n'ait pas la forme en « bol » caractéristique d'une fonction convexe, elle possède un optimum unique.

8.3 Stratégies unidimensionnelles

Comme dans le dernier chapitre, nous commencerons par une optimisation unidimensionnelle de f : R → R puis étendrons nos stratégies à des fonctions plus générales f : Rn → R.

8.3.1 Méthode de Newton

Notre stratégie principale pour minimiser les fonctions différentiables f : Rn → R sera de trouver sta sont des points satisfaisant f(x) = 0. En supposant que nous pouvons vérifier si les points stationnaires tionnaires x maxima, minima ou points de selle comme post- étape de traitement, nous allons nous concentrer sur le problème de la recherche des points fixes x

Pour cela, supposons que f : R → R soit différentiable. Ensuite, comme dans notre dérivation de Newton méthode de recherche de racine, nous pouvons approximer :

$$1\ f(x) \approx f(xk) + f\ (xk)(x - xk) + f\ (xk)(x - xk)\ 2^{2}.$$

L'approximation de droite est une parabole dont le sommet est situé en xk − f (xk)/f (xk).
Bien sûr, en réalité f n'est pas nécessairement une parabole, donc la méthode de Newton itère simplement la formule

$$xk+1 = xk - \frac{f\ (xk)}{f\ (xk)}.$$

Cette technique est facilement analysable étant donné le travail que nous avons déjà fait pour comprendre la méthode de recherche de racines de Newton dans le chapitre précédent. En particulier, une autre façon de dériver la formule ci-dessus provient de la recherche de racine sur f (x), puisque les points stationnaires satisfont f (x) = 0.
Ainsi, dans la plupart des cas, la méthode d'optimisation de Newton présente une convergence quadratique, à condition que l'estimation initiale x0 soit suffisamment proche de x .

Une question naturelle à se poser est de savoir si la méthode de la sécante peut être appliquée de manière analogue. Notre dérivation de la méthode de Newton ci-dessus trouve les racines de f , donc la méthode sécante pourrait être utilisée pour éliminer l'évaluation de f mais pas f ; les situations dans lesquelles nous connaissons f mais pas f sont relativement rares. Un parallèle plus approprié consiste à remplacer les segments de droite utilisés pour approximer f dans la méthode sécante par des paraboles. Cette stratégie, connue sous le nom d'interpolation parabolique successive, minimise également une approximation quadratique de f à chaque itération, mais plutôt que d'utiliser f(xk), f (xk) et f (xk) pour construire l'approximation qu'elle utilise f(xk), f(xk−1), et f(xk−2). La dérivation de cette technique est relativement simple et elle converge de manière superlinéaire.

8.3.2 Recherche de Golden Section

Nous avons sauté la bissection dans notre parallèle des techniques de recherche de racine à variable unique. Il y a plusieurs raisons à cette omission. Notre motivation pour la bissection était qu'elle n'utilisait que l'hypothèse la plus faible sur f nécessaire pour trouver des racines : la continuité. Cependant, le théorème de la valeur intermédiaire ne s'applique pas aux minima de manière intuitive, il semble donc qu'une approche aussi simple n'existe pas.

Il est toutefois utile de disposer d'au moins une stratégie de minimisation qui n'exige pas la dérivabilité de f comme hypothèse sous-jacente ; après tout, il existe des fonctions non différentiables qui ont des minima clairs, comme f(x) ≡ |x| à x = 0. À cette fin, une hypothèse alternative pourrait être que f est unimodulaire :

Définition 8.5 (Unimodulaire). Une fonction f : [a, b] → R est unimodulaire s'il existe x [a, b] tel que f soit décroissante pour x [a, x] et croissante pour x [x , b].

En d'autres termes, une fonction unimodulaire diminue pendant un certain temps, puis commence à augmenter ; aucun minimum localisé n'est autorisé. Notez que des fonctions comme |x| ne sont pas différentiables mais sont toujours unimodulaires.

Supposons que nous ayons deux valeurs x0 et x1 telles que a < x0 < x1 < b. Nous pouvons faire deux observations qui nous aideront à formuler une technique d'optimisation :

- Si f(x0) ≥ f(x1), alors on sait que f(x) ≥ f(x1) pour tout x [a, x0]. Ainsi, l'intervalle [a, x0] peut être écarté dans notre recherche d'un minimum de f .

• Si f(x1) ≥ f(x0), alors on sait que f(x) ≥ f(x0) pour tout x [x1, b], et donc on peut écarter [x1, b].

Cette structure suggère une stratégie potentielle de minimisation commençant par l'intervalle [a, b] et supprimant itérativement des pièces selon les règles ci-dessus.

Un détail important demeure cependant. Notre garantie de convergence pour l'algorithme de bissection provenait du fait que nous pouvions supprimer la moitié de l'intervalle en question à chaque itération.
On pourrait procéder de la même manière en supprimant à chaque fois un tiers de l'intervalle ; cela nécessite deux évaluations de f lors de chaque itération aux nouveaux emplacements x0 et x1 . Si l'évaluation de f est coûteuse, cependant, nous pouvons souhaiter réutiliser les informations des itérations précédentes pour éviter au moins une de ces deux évaluations.

Pour l'instant a = 0 et b = 1 ; les stratégies que nous dérivons ci-dessous fonctionneront plus généralement par déplacement et mise à l'échelle. En l'absence de plus d'informations sur f , autant faire un choix symétrique x0 = α et x1 = 1 − α pour un certain α (0, 1/2). Supposons que notre itération supprime l'intervalle le plus à droite [x1, b]. Alors, l'intervalle de recherche devient [0, 1 − α], et nous connaissons f(α) de l'itération précédente. La prochaine itération divisera [0, 1 − α] tel que x0 = α(1 − α) et x1 = (1 − α) ^{2}. Si nous souhaite réutiliser f(α) de l'itération précédente, on pourrait poser (1 − α) ^{2} = α, donnant :

$$a = \frac{1}{2}(3 - \sqrt{5})$$

$$1 - \alpha = 2^{-1}(\sqrt{5} - 1)$$

La valeur de 1 − α ≡ τ ci-dessus est le nombre d'or ! Il permet de réutiliser l'une des évaluations de fonction des itérations précédentes ; un argument symétrique montre que le même choix de α fonctionne si nous avions supprimé l'intervalle de gauche au lieu de celui de droite.

L'algorithme de recherche du nombre d'or utilise cette construction (CITE):

1. Prendre τ ≡ $\frac{1}{2}(\sqrt{5} - 1)$., et initialiser a et b pour que f soit unimodulaire sur [a, b].
2. Faire une subdivision initiale x0 = a + (1 − τ)(b − a) et x1 = a + τ(b − a).
3. Initialisez f0 = f(x0) et f1 = f(x1).
4. Itérer jusqu'à ce que b − a soit suffisamment petit :
   (a) Si f0 ≥ f1, alors supprimez l'intervalle [a, x0] comme suit :
   • Déplacer le côté gauche : a ← x0 • Réutiliser l'itération précédente : x0 ← x1, f0 ← f1 • Générer un nouvel échantillon : x1 ← a + τ(b − a), f1 ← f(x1)
   (b) Si f1 > f0, alors supprimez l'intervalle [x1, b] comme suit :
   • Déplacer vers la droite : b ← x1 • Réutiliser l'itération précédente : x1 ← x0, f1 ← f0 • Générer un nouvel échantillon : x0 ← a + (1 − τ)(b − a), f0 ← f(x0)

Cet algorithme converge clairement de manière inconditionnelle et linéaire. Lorsque f n'est pas globalement unimodale, il peut être difficile de trouver [a, b] tel que f soit unimodale sur cet intervalle, ce qui limite quelque peu les applications de cette technique ; généralement [a, b] est deviné en essayant de mettre entre parenthèses un minimum local de f .

8.4 Stratégies multivariables

Nous continuons dans notre parallèle de notre discussion sur la recherche de racines en élargissant notre discussion aux problèmes multivariables. Comme pour la recherche de racines, les problèmes à plusieurs variables sont considérablement plus difficiles que les problèmes à une seule variable, mais ils apparaissent tellement de fois dans la pratique qu'ils méritent une attention particulière.

Ici, nous ne considérerons que le cas où f : Rn → R est différentiable. Les méthodes d'optimisation plus proches de la recherche du nombre d'or pour les fonctions non différentiables ont des applications limitées et sont difficiles à formuler.

8.4.1 Descente de gradient

Rappelons-nous de notre discussion précédente que f(x) pointe dans la direction de « la montée la plus raide » de f en x ; de même, le vecteur − f(x) est la direction de la « descente la plus raide ». Si rien d'autre, cette définition garantit que lorsque f(x) =0, pour petit α > 0 nous devons avoir

$$f(x - \alpha \quad f(x)) \leq f(x).$$

Supposons que notre estimation actuelle de l'emplacement du minimum de f soit xk . Ensuite, on peut souhaiter choisir xk+1 de sorte que f(xk+1) < f(xk) pour une stratégie de minimisation itérative. Une façon de simplifier la recherche de xk+1 serait d'utiliser un de nos algorithmes unidimensionnels du §8.3 sur un problème plus simple. En particulier, considérons la fonction gk(t) ≡ f(xk − t f(xk)), qui restreint f à la droite passant par xk parallèle à f(xk). Grâce à notre discussion sur le gradient, nous savons qu'un petit t entraînera une diminution de f .

L'algorithme de descente de gradient résout itérativement ces problèmes unidimensionnels pour améliorer notre estimation de xk :

1. Choisissez une estimation initiale x0
2. Itérer jusqu'à convergence de xk :
   (a) Prendre gk(t) ≡ f(xk − t f(xk)) (b)
   Utiliser un algorithme unidimensionnel pour trouver t minimisant gk sur tout t ≥ 0 ("line search")
   (c) Prendre xk+1 ≡ xk − t f(xk)

Chaque itération de descente de gradient diminue f(xk), donc les valeurs objectives convergent. L'algorithme ne se termine que lorsque f(xk) ≈ 0, montrant que la descente de gradient doit au moins atteindre un minimum local ; cependant, la convergence est lente pour la plupart des fonctions f . Le processus de recherche de ligne peut être remplacé par une méthode qui diminue simplement l'objectif d'une quantité non négligeable si sous-optimale, bien qu'il soit plus difficile de garantir la convergence dans ce cas.

8.4.2 Méthode de Newton

Parallèlement à notre dérivation du cas à variable unique, nous pouvons écrire une approximation en série de Taylor de f : Rn → R en utilisant son hessien Hf :

$$f(x) \approx f(xk) + \quad f(xk) \cdot (x - xk) + \frac{1}{2} (x - xk) \cdot Hf(xk) \cdot (x - xk)$$

En différenciant par rapport à x et en fixant le résultat égal à zéro, on obtient le schéma itératif suivant :

$$x_{k+1} = x_k - [Hf(x_k)]^{-1} \quad f(x_k)$$

Il est facile de revérifier que cette expression est une généralisation de celle du §8.3.1, et encore une fois elle converge quadratiquement lorsque x0 est proche d'un minimum.

La méthode de Newton peut être plus efficace que la descente de gradient selon l'objectif d'optimisation f . Rappelons que chaque itération de descente de gradient nécessite potentiellement de nombreuses évaluations de f lors de la procédure de recherche de ligne. Par contre, il faut évaluer et inverser le Hessien Hf à chaque itération de la méthode de Newton. Notez que ces facteurs n'affectent pas le nombre d'itérations mais affectent le temps d'exécution : il s'agit d'un compromis qui peut ne pas être évident via une analyse traditionnelle.

Il est intuitif pourquoi la méthode de Newton converge rapidement lorsqu'elle est proche d'un optimum. En particulier, la descente de gradient n'a aucune connaissance de Hf ; il procède de manière analogue à la descente d'une pente en ne regardant que ses pieds. En utilisant Hf , la méthode de Newton a une image plus large de la forme de f à proximité.

Lorsque Hf n'est pas défini positif, cependant, l'objectif local peut ressembler à une selle ou à un pic plutôt qu'à un bol. Dans ce cas, sauter à un point stationnaire approximatif peut ne pas avoir de sens.
Ainsi, des techniques adaptatives pourraient vérifier si Hf est définie positive avant d'appliquer un pas de Newton ; s'il n'est pas défini positif, les méthodes peuvent revenir à la descente de gradient pour trouver une meilleure approximation du minimum. Alternativement, ils peuvent modifier Hf , par exemple en projetant sur la matrice définie positive la plus proche.

8.4.3 Optimisation sans dérivées : BFGS

La méthode de Newton peut être difficile à appliquer aux fonctions compliquées f : Rn → R. La dérivée seconde de f peut être considérablement plus impliquée que la forme de f , et Hf change à chaque itération, ce qui rend difficile la réutilisation du travail des itérations précédentes. De plus, Hf a une taille n × n, donc le stockage de Hf nécessite $O(n^2)$ l'espace, ce qui peut être inacceptable.

Comme dans notre discussion sur la recherche de racine, les techniques de minimisation qui imitent la méthode de Newton mais utilisent des dérivées approximatives sont appelées méthodes de quasi-Newton. Souvent, ils peuvent avoir des propriétés de convergence tout aussi fortes sans qu'il soit nécessaire de réévaluer explicitement et même de factoriser le hessien à chaque itération. Dans notre discussion, nous suivrons le développement de (CITE NO CEDAL AND WRIGHT).

Supposons que l'on souhaite minimiser f : Rn → R en utilisant un schéma itératif. Proche de l'estimation actuelle xk de la racine, on pourrait estimer f avec un modèle quadratique :

$$f(x_k + \delta x) \approx f(x_k) + \quad f(x_k) \cdot \delta x + \quad \frac{1}{2} (\delta x) B_k(\delta x).$$

Remarquez que nous avons demandé que notre approximation soit en accord avec f au premier ordre en xk ; comme dans la méthode de Broyden pour trouver les racines, cependant, nous permettrons à notre estimation du Hessian Bk de varier.

Ce modèle quadratique est minimisé en prenant $\delta x = -B_k^{-1} \quad f(x_k)$. Dans le cas où δx2 est grand et que nous ne souhaitons pas faire un pas aussi considérable, nous nous permettrons de mettre à l'échelle cette différence par une taille de pas αk , ce qui donne

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \quad f(x_k).$$

Notre objectif est de trouver une estimation raisonnable de Bk+1 en mettant à jour Bk , afin de pouvoir répéter ce processus.

La Hessienne de f n'est rien de plus que la dérivée de f , on peut donc écrire une condition de type sécante sur Bk+1 :

$$B_{k+1}(x_{k+1} - x_k) = f(x_{k+1}) - f(x_k).$$

On substituera $s_k \equiv x_{k+1} - x_k$ et $y_k \equiv f(x_{k+1}) - f(x_k)$, donnant une condition équivalente $B_{k+1}s_k = y_k$.

Compte tenu de l'optimisation en cours, nous souhaitons que Bk ait deux propriétés :

- Bk devrait être une matrice symétrique, comme la Hessienne Hf .
- Bk devrait être positif (semi-)défini, de sorte que nous recherchons des minima plutôt que des maxima ou points de selle.

La condition de symétrie est suffisante pour éliminer la possibilité d'utiliser l'estimation de Broyden que nous avons développée au chapitre précédent.

La contrainte définie positive conditionne implicitement la relation entre sk et En particulier, en prémultipliant la relation $B_{k+1}s_k = y_k$ par s $B_{k+1}s_k = s_k y_k$. Pour y_k . $_k$ montre s_k B_{k+1} pour être défini positif, il faut alors avoir$s_k \cdot y_k > 0$. Cette observation peut guider notre choix de α_k ; il est facile de voir qu'elle est valable pour $\alpha_k > 0$ suffisamment petit.

Supposons que sk et yk satisfassent notre condition de compatibilité. Avec cela en place, nous pouvons écrire une optimisation à la Broyden conduisant à une possible approximation Bk+1 :

$$\begin{aligned} &\text{minimiser}_{B_{k+1}} \ \|B_{k+1} - B_k\| \ \text{tel que} \\ &B_{k+1} = B_{c+1} \\ &B_{k+1}s_k = y_k \end{aligned}$$

Pour un choix approprié des normes $\|\cdot\|$, cette optimisation donne le célèbre DFP (Davidon Fletcher-Powell) schéma itératif.

Plutôt que de travailler sur les détails du schéma DFP, nous passons à une méthode plus populaire connue sous le nom de formule BFGS (Broyden-Fletcher-Goldfarb-Shanno), qui apparaît dans de nombreux systèmes modernes. Notez que—ignorant notre choix de α_k pour l'instant—notre approximation du second ordre a été minimisée en prenant $\delta x = -B_{k-1}^{-1} \nabla f(x_k)$. Ainsi, au final le comportement de notre schéma itératif est dicté par la matrice inverse B_{k-1}^{-1}. Demander que $B_{k+1} - B_k$ est petit peut encore impliquer des différences relativement mauvaises entre l'action de B_k^{-1} et celui de B_{k+1}^{-1} !

Avec cette observation à l'esprit, le schéma BFGS apporte une légère modification à la dérivation ci-dessus. Plutôt que de calculer Bk à chaque itération, on peut calculer directement son inverse $H_k \equiv B_k^{-1}$. Maintenant notre condition $B_{k+1}s_k = y_k$ est inversée tosk = Hk+1yk ; la condition que Bk soit symétrique revient à demander que Hk soit symétrique. Nous résolvons une optimisation

$$\begin{aligned} &\text{minimiser}_{H_{k+1}} \ \|H_{k+1} - H_k\| \ \text{tel que} \\ &H_{k+1} = H_{k+1} \\ &s_k = H_{k+1}y_k \end{aligned}$$

Cette construction a l'avantage de ne pas nécessiter d'inversion de matrice pour calculer $\delta x = -H_k \nabla f(x_k)$.

Pour dériver une formule pour Hk+1 , nous devons décider d'une norme matricielle $\|\cdot\|$. Comme dans notre discussion précédente, la norme de Frobenius ressemble le plus à l'optimisation des moindres carrés, ce qui rend probable que nous puissions générer une expression de forme fermée pour Hk + 1 plutôt que d'avoir à résoudre la minimisation ci-dessus en tant que sous-programme d'optimisation BFGS.

La norme de Frobenius, cependant, présente un sérieux inconvénient pour les matrices hessiennes. Rappelons que la matrice Hessienne a des entrées (Hf)ij = ∂ fi/∂xj . Souvent les quantités xi pour différents i peuvent avoir des unités différentes ; par exemple, envisagez de maximiser le profit (en dollars) réalisé en vendant un cheeseburger de rayon r (en pouces) et de prix p (en dollars), ce qui donne f : (pouces, dollars) → dollars. Mettre au carré ces différentes quantités et les additionner n'a pas de sens.

Supposons que nous trouvions une matrice définie positive symétrique W telle que Wsk = yk ; nous vérifierons dans les exercices qu'une telle matrice existe. Une telle matrice ramène les unités de sk = xk+1 − xk à celles de yk = f(xk+1) − f(xk). En nous inspirant de notre expression $A = \text{Tr}(A^2)$, nous pouvons définir une norme de Frobenius pondérée d'une matrice A comme

$$\|A\|_W^2 \equiv \text{Tr}(AWAW)$$

Il est simple de vérifier que cette expression a des unités cohérentes lorsqu'elle est appliquée à notre optimisation pour Hk+1 . Lorsque W et A sont symétriques avec des colonnes w i andai , respectivement, développer l'expression ci-dessus montre :

$$\|A\|_W^2 = \sum_{ij} (w_i \cdot a_j)(w_j \cdot a_i).$$

Ce choix de norme combiné au choix de W donne une formule particulièrement propre pour Hk+1 et yk : étant donné Hk ,sk ,

$$H_{k+1} = (I_{n\times n} - \rho_k s_k y_k^T) H_k (I_{n\times n} - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

où ρk ≡ 1/y·s. Nous montrons dans l'annexe de ce chapitre comment dériver cette formule.

L'algorithme BFGS évite le besoin de calculer et d'inverser une matrice hessienne pour f , mais il nécessite toujours $O(n^2)$ stockage pour Hk. Une variante utilisée couramment sous le nom de L-BFGS ("Limited-Memory BFGS") évite ce problème en gardant un historique limité des vecteurs yk andsk et en appliquant Hk en développant sa formule récursivement.Cette approche peut effectivement avoir de meilleures propriétés numériques malgré son utilisation compacte de l'espace ; en particulier, les anciens vecteurs yk etsk peuvent ne plus être pertinents et doivent être ignorés.

8.5 Problèmes

Liste d'idées :

- Dériver Gauss-Newton
- Méthodes stochastiques, AdaGrad
- Algorithme VSCG
- Conditions de Wolfe pour la descente en pente ; brancher sur BFGS
- Formule de Sherman-Morrison-Woodbury pour Bk pour BFGS
- Prouver que BFGS converge ; montrer l'existence d'une matrice W
- Algorithme de gradient réduit (généralisé)
- Numéro de condition pour l'optimisation

Annexe : Dérivation de la mise à jour BFGS1

Notre optimisation pour Hk+1 a l'expression suivante du multiplicateur de Lagrange (pour faciliter la notation, nous prenons Hk+1 ≡ H et Hk = H) :

$$\Lambda \equiv \sum_{ij} (w_{je} \cdot (h_j - h_{\ j}))(w_j \cdot (h_i - h_{\ \cdot})) - \sum_{je<j} \alpha_{ij}(H_{ij} - H_{ji}) - \lambda (H y_k - s_k)$$

$$= \sum_{ij} (w_{je} \cdot (h_j - h_{\ j}))(w_j \cdot (h_i - h_{\ \cdot})) - \sum_{ij} \alpha_{ij} H_{ij} - \lambda (H y_k - s_k) \text{ si on suppose } \alpha_{ij} = -\alpha_{ji}$$

Prendre des dérivées pour trouver des points critiques montre (pour $y \equiv y_k, s \equiv s_k$) :

$$0 = \frac{\partial \Lambda}{\partial H_{ij}} = \sum 2w_i(w_j \cdot (h - h_{\ \cdot})) - \alpha_{ij} - \lambda_i y_j$$

$$= 2\sum w_i(W(H - H_{\ \cdot}))_j - \alpha_{ij} - \lambda_i y_j$$

$$= 2\sum (W(H - H_{\ \cdot}))_j w_i - \alpha_{ij} - \lambda_i y_j \text{ par symétrie de W}$$

$$= 2(W(H - H_{\ \cdot})W)_{ji} - \alpha_{ij} - \lambda_i y_j$$

$$= 2(W(H - H_{\ \cdot})W)_{ij} - \alpha_{ij} - \lambda_i y_j \text{ par symétrie de W et H}$$

Ainsi, sous forme matricielle, nous avons la liste de faits suivante :

$$0 = 2W(H - H_{\ \cdot})W - A - \lambda y^{\top}, \text{ où } A_{ij} = \alpha_{ij}$$

$$\text{UNE} = -A,\ W = W,\ H = H^{\top},(H_{\ \cdot}) = H$$

$$Hy = s,\ Ws = y$$

Nous pouvons obtenir une paire de relations en utilisant la transposition combinée avec la symétrie de H et W et l'asymétrie de A :

$$0 = 2W(H - H_{\ \cdot})W - A - \lambda y^{\top}$$

$$0 = 2W(H - H_{\ \cdot})W + A - y\lambda^{\top} = 0 =$$

$$4W(H - H_{\ \cdot})W - \lambda y^{\top} - y\lambda^{\top}$$

La post-multiplication de cette relation par montre :

$$0 = 4(y - WH_{\ \cdot}\ y) - \lambda(y^{\top} \cdot s) - y(\lambda^{\top} \cdot s)$$

Maintenant, prenez le produit scalaire avec :

$$0 = 4(y^{\top} \cdot s) - 4(y^{\top} H_{\ \cdot}\ y) - 2(y^{\top} \cdot s)(\lambda^{\top} \cdot s)$$

Ceci montre:

$$\lambda^{\top} \cdot s = 2\rho y^{\top}(s - H_{\ \cdot}\ y), \text{ pour } \rho \equiv 1/y^{\top} \cdot s$$

1Remerciements particuliers à Tao Du pour le débogage de plusieurs parties de cette dérivation.

Maintenant, nous substituons ceci dans notre égalité vectorielle :

$$0 = 4(y - WH\ y) - \lambda(y \cdot s) - y(\lambda \cdot s) \text{ d'avant} =$$

$$4(y - WH\ y) - \lambda(y \cdot s) - y[2\rho y\ (s - H\ y)] \text{ de notre simplification}$$

$$= \lambda = 4\rho(y - WH\ y) - 2\rho^2 y\ (s - H\ y)y$$

La post-multiplication par y montre :

$$\lambda y = 4\rho(y - WH\ y)y - 2\rho^2 y\ (s - H\ y)yy$$

Prenant la transposition,

$$y\lambda = 4\rho y(y - y\ H\ W) - 2\rho^2 y\ (s - H\ y)yy$$

En combinant ces résultats et en divisant par quatre, on obtient :

$$\frac{1}{4}(\lambda y + y\lambda) = \rho(2yy - WH\ yy - yy\ H\ W) - \rho^2 y\ (s - H\ y)yy$$

Maintenant, nous allons pré- et post-multiplier par W^{-1} . Puisque $Ws = y$, on peut écrire de manière équivalente $= W^{-1}y$; de plus, par symétrie de W on sait alors $y\ W^{-1} = s$. L'application de ces identités à l'expression ci-dessus montre :

$$\frac{1}{4}W^{-1}(\lambda y + y\lambda)W^{-1} = 2\rho ss - \rho H\ ys - \rho sy\ H - \rho^2 (y\ s)ss + \rho^2 (y\ H\ y)ss$$

$$= 2\rho ss - \rho H\ ys - \rho sy\ H - \rho ss + s\rho^2 (y\ H\ y)s \text{ par définition de } \rho$$

$$= \rho ss - \rho H\ ys - \rho sy\ H + s\rho^2 (y\ H\ y)s$$

Enfin, nous pouvons conclure notre dérivation de l'étape BFGS comme suit :

$$0 = 4W(H - H\ )W - \lambda y - y\lambda \text{ d'avant}$$

$$= H = 4\frac{1}{=} - W^{-1}(\lambda y + y\lambda)W^{-1} + H$$

$$\rho ss - \rho H\ ys - \rho sy\ H + s\rho^2 (y\ H\ y)s + H \text{ du dernier paragraphe}$$

$$= H\ (I - \rho ys) + \rho ss - \rho sy\ H\ (I + (\rho sy)H\ (\rho ys)$$

$$= H - \rho ys) + \rho ss - \rho sy\ H\ (I - \rho ys) = \rho ss + (I$$

$$- \rho sy)H\ (I - \rho ys)$$

Cette expression finale est exactement l'étape BFGS introduite dans le chapitre.

Chapitre 9

Optimisation contrainte

Nous poursuivons notre réflexion sur les problèmes d'optimisation en étudiant le cas contraint. Ces problèmes prennent la forme générale suivante :

$$\text{minimiser } f(x) \text{ tel}$$
$$\text{que } g(x) = 0 \; h(x) \geq 0$$

Ici, $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, et $h : \mathbb{R}^n \to \mathbb{R}^p$. Évidemment, cette forme est extrêmement générique, il n'est donc pas difficile de prédire que les algorithmes pour résoudre de tels problèmes en l'absence d'hypothèses supplémentaires sur f , g ou h peuvent être difficiles à formuler et sont sujets à des dégénérescences telles que des minima locaux et le manque de convergence. En fait, cette optimisation encode d'autres problèmes que nous avons déjà envisagés ; si nous prenons $f(x) \equiv 0$, alors cette optimisation contrainte devient une recherche de racine sur g, tandis que si nous prenons $g(x) = h(x) \equiv 0$ alors elle se réduit à une optimisation sans contrainte sur f .

Malgré ces perspectives quelque peu sombres, les optimisations pour le cas contraint général peuvent être utiles lorsque f , g et h n'ont pas de structure utile ou sont trop spécialisés pour mériter un traitement spécialisé. De plus, quand f est heuristique de toute façon, trouver simplement un faisablex pour lequel $f(x) < f(x_0)$ pour une estimation initiale x_0 est valable. Une application simple dans ce domaine serait un système économique dans lequel f mesure les coûts ; évidemment, nous souhaitons minimiser les coûts, mais si x_0 représente la configuration actuelle, tout x diminuant f est une sortie précieuse.

9.1 Motivations

Il n'est pas difficile de rencontrer des problèmes d'optimisation sous contrainte dans la pratique. En fait, nous avons déjà énuméré de nombreuses applications de ces problèmes lorsque nous avons discuté des vecteurs propres et des valeurs propres, puisque ce problème peut être posé comme la recherche des points critiques de $x^\top Ax$ sous réserve de $\|x\|_2 = 1$; bien sûr, le cas particulier du calcul des valeurs propres admet des algorithmes spéciaux qui en font un problème plus simple.

Ici, nous listons d'autres optimisations qui n'apprécient pas la structure des problèmes aux valeurs propres :

Exemple 9.1 (Projection géométrique). De nombreuses surfaces S dans $\mathbb{R}^3$ peuvent s'écrire implicitement sous la forme $g(x) = 0$ pour un certain g. Par exemple, la sphère unitaire résulte de la prise de $g(x) \equiv \|x\|_2^2 - 1$, alors qu'un cube peut

être construit en prenant g(x) = x1 − 1. En fait, certains environnements de modélisation 3D permettent aux utilisateurs de spécifier des objets "blobby", comme dans la figure NUMBER, comme des sommes

$$g(x) \equiv c + \sum \quad -bix-xi\ aie\ {}^{2}_{2}.$$

Supposons qu'on nous donne un point y R3 et que nous souhaitions trouver le point le plus proche sur S de y. Ce problème est résolu en utilisant la minimisation contrainte suivante :

$$\text{minimiserx } x - y2 \text{ tel que}$$
$$g(x) = 0$$

Exemple 9.2 (Fabrication). Supposons que vous disposiez de m matériaux différents ; vous avez si unités de chaque matériau i en stock. Vous pouvez fabriquer k produits différents ; le produit j vous donne un profit pj et utilise cij du matériau i pour le fabriquer. Pour maximiser les profits, vous pouvez résoudre l'optimisation suivante pour le montant total xj que vous devriez fabriquer pour chaque article j :

$$\text{maximiserx } pjx\sum_{j=1}^{k}$$
$$\text{tel que } xj \geq 0 \quad j \quad \{1, \ldots, k\}$$
$$\sum_{j=1}^{k} cijxj \leq si \quad i \quad \{1, \ldots, m\}$$

La première contrainte garantit que vous ne faites pas de nombres négatifs d'aucun produit, et la seconde garantit que vous n'utilisez pas plus que votre stock de chaque matériau.

Exemple 9.3 (Moindres carrés non négatifs). Nous avons déjà vu de nombreux exemples de problèmes de moindres carrés, mais parfois des valeurs négatives dans le vecteur de solution peuvent ne pas avoir de sens. Par exemple, en infographie, un modèle animé peut être exprimé sous la forme d'une structure osseuse déformante plus une « peau » maillée ; pour chaque point de la peau, une liste de poids peut être calculée pour approximer l'influence des positions des articulations osseuses sur la position des sommets de la peau (CITE). Ces poids doivent être contraints d'être non négatifs pour éviter un comportement dégénéré pendant que la surface se déforme. Dans un tel cas, on peut résoudre le problème des « moindres carrés non négatifs » :

$$\text{minimiserx } Ax - b2 \text{ tel que}$$
$$xi \geq 0 \quad i$$

Des recherches récentes consistent à caractériser la parcimonie des solutions des moindres carrés non négatifs, qui ont souvent plusieurs valeurs xi satisfaisant xi = 0 exactement (CITE).

Exemple 9.4 (Ajustement groupé). En vision par ordinateur, supposons que nous prenions une photo d'un objet sous plusieurs angles. Une tâche naturelle consiste à reconstruire la forme tridimensionnelle de l'objet. Pour ce faire, nous pourrions marquer un ensemble de points correspondant sur chaque image ; en particulier, nous pouvons prendre xij R2 comme étant la position du point caractéristique j sur l'image i. En réalité, chaque point caractéristique a une position yj R3 dans l'espace, que nous aimerions calculer. De plus, nous devons trouver les positions des caméras elles-mêmes, que nous pouvons représenter comme

matrices de projection inconnues Pi . Ce problème, connu sous le nom d'ajustement groupé, peut être abordé à l'aide d'une stratégie d'optimisation :

$$\text{minimiseryj ,Pi} \sum_{ij} \left\| \text{Piyj} - \text{xij} \right\|_2^2$$

$$\text{tel que Pi soit orthogonal} \quad \text{i}$$

La contrainte d'orthogonalité garantit que les transformations de la caméra sont raisonnables.

9.2 Théorie de l'optimisation contrainte

Dans notre discussion, nous supposerons que f , g et h sont différentiables. Certaines méthodes existent qui ne font que de faibles hypothèses de continuité ou de Lipschitz, mais ces techniques sont assez spécialisées et nécessitent une réflexion analytique avancée.

Bien que nous n'ayons pas encore développé d'algorithmes pour l'optimisation générale sous contrainte, nous avons implicitement fait usage de la théorie de tels problèmes en considérant les méthodes aux valeurs propres. Plus précisément, rappelons la méthode des multiplicateurs de Lagrange, introduite dans le théorème 0.1. Dans cette technique, les points critiques f(x) soumis à g(x) sont caractérisés comme des points critiques de la fonction multiplicatrice de La grange non contrainte $\Lambda(x,\lambda) \equiv f(x) - \lambda \cdot g(x)$ par rapport aux deux λ et x simultanément.
Ce théorème nous a permis de fournir des interprétations variationnelles des problèmes aux valeurs propres ; plus généralement, il donne un critère alternatif (nécessaire mais pas suffisant) pour que x soit un point critique d'une optimisation sous contrainte d'égalité.

Cependant, trouver simplement un x satisfaisant les contraintes peut être un défi considérable. Nous pouvons séparer ces questions en faisant quelques définitions :

Définition 9.1 (Point réalisable et ensemble des réalisables). Un point réalisable d'un problème d'optimisation contraint est tout point x satisfaisant g(x) =0 et h(x) ≥0. L'ensemble des possibles est l'ensemble de tous les points x satisfaisant ces contraintes.

Définition 9.2 (Point critique de l'optimisation contrainte). Un point critique d'une optimisation contrainte est un point satisfaisant aux contraintes qui est également un maximum, un minimum ou un point de selle local de f dans l'ensemble des possibles.

Les optimisations contraintes sont difficiles car elles résolvent simultanément des problèmes de recherche de racine (la contrainte g(x) =0), des problèmes de satisfiabilité (la contrainte h(x) ≥0) et de minimisation (la fonction f). Ceci mis à part, pour pousser nos techniques différentielles à une généralité complète, nous devons trouver un moyen d'ajouter des contraintes d'inégalité au système multiplicateur de Lagrange. Supposons que nous ayons trouvé le minimum de l'optimisation, noté x . Pour chaque contrainte d'inégalité hi(x) ≥ 0, nous avons deux options :

- hi(x) = 0 : une telle contrainte est active, indiquant probablement que si la contrainte était supprimée, l'optimum pourrait changer.
- hi(x) > 0 : Une telle contrainte est inactive, c'est-à-dire qu'au voisinage de x si nous avions supprimé cette contrainte nous aurions toujours atteint le même minimum.

Bien sûr, nous ne savons pas quelles contraintes seront actives ou inactives en x jusqu'à ce qu'elles soient calculées.

Si toutes nos contraintes étaient actives, alors nous pourrions changer notre contrainte h(x) ≥0 en une égalité sans affecter le minimum. Cela pourrait motiver l'étude du système multiplicateur de Lagrange suivant :

$$\Lambda(x,\lambda,\mu) \equiv f(x) - \lambda \cdot g(x) - \mu \cdot h(x)$$

Cependant, nous ne pouvons plus dire que x est un point critique de Λ, car des contraintes inactives supprimeraient les termes ci-dessus. En ignorant cette question (importante !) pour le moment, nous pourrions procéder aveuglément et demander des points critiques de ce nouveau Λ par rapport à x, qui satisfassent ce qui suit :

$$0 = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

Ici, nous avons séparé les composants individuels de g et h et les avons traités comme des fonctions scalaires pour éviter une notation complexe.

Une astuce astucieuse peut étendre cette condition d'optimalité aux systèmes à contraintes d'inégalité. Remarquez que si nous avions pris μj = 0 chaque fois que hj est inactif, cela supprime les termes non pertinents des conditions d'optimalité. En d'autres termes, nous pouvons ajouter une contrainte sur les multiplicateurs de Lagrange :

$$\mu_j h_j(x) = 0.$$

Avec cette contrainte en place, nous savons qu'au moins l'un de μj et hj(x) doit être nul, et donc notre condition d'optimalité du premier ordre tient toujours !

Jusqu'à présent, notre construction n'a pas fait la distinction entre la contrainte hj(x) ≥ 0 et la contrainte hj(x) ≤ 0. Si la contrainte est inactive, elle aurait pu être abandonnée sans affecter le résultat de l'optimisation localement, donc nous considérons le cas où la contrainte est active. Intuitivement1, dans ce cas, nous nous attendons à ce qu'il y ait un moyen de diminuer f en violant la contrainte. Localement, la direction dans laquelle f décroît est $-\nabla f(x)$ et la direction dans laquelle hj décroît est $-\nabla h_j(x)$. Ainsi, à partir de x, nous pouvons encore diminuer f en violant la contrainte hj(x) ≥ 0 lorsque $\nabla f(x) \cdot \nabla h_j(x) \geq 0$.

Bien sûr, les produits de gradients de f et hj sont difficiles à travailler. Cependant, rappelons qu'à notre condition x^* d'optimalité du premier ordre nous dit :

$$\nabla f(x^*) = \sum_i \lambda_i \nabla g_i(x^*) + \sum_{\mu_j \text{ actif}} \mu_j \nabla h_j(x^*)$$

Les valeurs μj inactives sont nulles et peuvent être supprimées. En fait, on peut supprimer les contraintes g(x) = 0 en ajoutant des contraintes d'inégalité g(x) ≥ 0 et g(x) ≤ 0 à h ; c'est une commodité mathématique pour écrire une preuve plutôt qu'une manœuvre numériquement sage. Ensuite, en prenant des produits scalaires avec ∇h_k pour tout k fixe, on obtient :

$$\sum_{\mu_j \text{ actif}} \mu_j \nabla h_j(x^*) \cdot \nabla h_k(x^*) = \nabla f(x^*) \cdot \nabla h_k(x^*) \geq 0$$

La vectorisation de cette expression montre $Dh(x^*)Dh(x^*)^T \mu \geq 0$. Puisque $Dh(x^*)Dh(x^*)^T$ est semidef positif inite, cela implique $\mu^T \geq 0$. Ainsi, l' observation $\nabla f(x^*) \cdot \nabla h_j(x^*) \geq 0$ se manifeste simplement par fait que μj ≥ 0.

Nos observations peuvent être formalisées pour prouver une condition d'optimalité du premier ordre pour les optimisations contraintes par les inégalités :

1Vous ne devez pas considérer notre discussion comme une preuve formelle, puisque nous ne considérons pas beaucoup de cas limites.

Théorème 9.1 (conditions de Karush-Kuhn-Tucker (KKT)). Le vecteur x Rn est un point critique pour minimiser f sous réserve de g(x) =0 et h(x) ≥0 lorsqu'il existe λ Rm et μ Rp tels que :

• 0 = f(x) − ∑i λi gi(x) − ∑j μj hj(x) (“stationnarité”)

• g(x) =0 et h(x) ≥0 (« faisabilité primale ») •

μjhj(x) = 0 pour tout j (« écart complémentaire ») •

μj ≥ 0 pour tout j (« faisabilité duale ») ”)

Notez que lorsque h est supprimé, ce théorème se réduit au critère du multiplicateur de Lagrange.

Exemple 9.5 (Optimisation simple[2]). Supposons que nous souhaitions résoudre

$$\begin{aligned} &\text{maximiser } xy \\ &\text{tel que } x + yx, y \geq^2 \leq 2 \\ &0 \end{aligned}$$

Dans ce cas, nous n'aurons pas de λ et trois μ. On prend f(x, y) = −xy, h1(x, y) ≡ 2− x − y et h3(x, y) = y. ^{2}, h2(x, y) = x, Les conditions KKT sont :

$$\begin{aligned} \text{Stationnarité : } 0 &= -y + \mu 1 - \mu 2\ 0 \\ &= -x + 2\mu 1y - \mu 3 \\ \text{Faisabilité primaire : } x + yx, &^2 \leq 2 \\ y &\geq 0 \\ \text{Écart complémentaire : } \mu 1(2 - x - y\ \mu 2x = 0 &^2) = 0 \\ \mu 3y &= 0 \end{aligned}$$

Double faisabilité : μ1, μ2, μ3 ≥ 0

Exemple 9.6 (Programmation linéaire). Considérez l'optimisation :

$$\begin{aligned} &\text{minimiserx } b \cdot x \\ &\text{tel que } Ax \geq c \end{aligned}$$

Remarque L'exemple 9.2 peut être écrit de cette façon. Les conditions KKT pour ce problème sont :

$$\begin{aligned} \text{Stationnarité : } &A\mu = b \\ \text{Faisabilité primaire : } &Ax \geq c \\ \text{Ecart complémentaire : } &\mu i(ai \cdot x - ci) = 0 \quad i, \text{ où} a \text{ est la ligne i de A} \\ \text{Double faisabilité : } &\mu \geq 0 \end{aligned}$$

Comme dans le cas des multiplicateurs de Lagrange, nous ne pouvons pas supposer que tout x satisfaisant les conditions KKT minimise automatiquement f sous les contraintes, même localement. Une façon de vérifier l'optimalité locale est d'examiner la Hessienne de f restreinte au sous-espace de Rn dans lequel x peut se déplacer sans violer les contraintes ; si ce hessien "réduit" est défini positif alors l'optimisation a atteint un minimum local.

2Tiré de http://www.math.ubc.ca/~israel/m340/kkt2.pdf

9.3 Algorithmes d'optimisation

Un examen attentif des algorithmes d'optimisation sous contraintes sort du cadre de notre discussion ; heureusement, il existe de nombreuses implémentations stables de ces techniques et beaucoup peut être accompli en tant que « client » de ce logiciel plutôt que de le réécrire à partir de zéro. Même ainsi, il est utile d'esquisser quelques approches potentielles pour avoir une idée du fonctionnement de ces bibliothèques.

9.3.1 Programmation quadratique séquentielle (SQP)

Semblable à BFGS et à d'autres méthodes que nous avons considérées dans notre discussion sur l'optimisation sans contrainte, une stratégie typique d'optimisation contrainte consiste à approximer f , g et h avec des fonctions plus simples, à résoudre l'optimisation approchée et à itérer.

Supposons que nous ayons une estimation xk de la solution au problème d'optimisation sous contrainte. Nous pourrions appliquer une expansion de Taylor du second ordre à f et une approximation du premier ordre à g et h pour définir une itération suivante comme suit :

$$x_{k+1} \equiv x_k + \operatorname{argmin}_d \frac{1}{2} d H f(x_k) d + \nabla f(x_k) \cdot d + f(x_k)$$

$$\text{tel que } g_i(x_k) + \nabla g_i(x_k) \cdot d = 0$$

$$h_i(x_k) + \nabla h_i(x_k) \cdot d \geq 0$$

L'optimisation pour trouver d a un objectif quadratique avec des contraintes linéaires, pour lesquelles l'optimisation peut être considérablement plus facile en utilisant l'une des nombreuses stratégies. C'est ce qu'on appelle un programme quadratique.

Bien sûr, cette approximation de Taylor ne fonctionne qu'au voisinage du point optimal. Lorsqu'une bonne estimation initiale x0 n'est pas disponible, ces stratégies échoueront probablement.

Contraintes d'égalité Lorsque les seules contraintes sont les égalités et que h est supprimé, le programme quadratique pour d a des conditions d'optimalité du multiplicateur de Lagrange dérivées comme suit :

$$\Lambda(d,\lambda) \equiv \frac{1}{2} d H f(x_k) d + \nabla f(x_k) \cdot d + f(x_k) + \lambda \left(g(x_k) + \nabla g(x_k) d\right)$$

$$= 0 = \nabla_\Lambda = H f(x_k) d + \nabla f(x_k) + [\nabla g(x_k)]\lambda$$

En combinant cela avec la condition d'égalité, on obtient un système linéaire :

$$\begin{bmatrix} H f(x_k) & [\nabla g(x_k)] \\ \nabla g(x_k) & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ -g(x_k) \end{bmatrix}$$

Ainsi, chaque itération de programmation quadratique séquentielle en présence de seules contraintes d'égalité peut être accomplie en résolvant ce système linéaire à chaque itération pour obtenir $x_{k+1} \equiv x_k + d$. Il est important de noter que le système linéaire ci-dessus n'est pas défini positif, donc à grande échelle, il peut être difficile à résoudre.

Les extensions de cette stratégie fonctionnent comme BFGS et des approximations similaires fonctionnent pour une optimisation sans contrainte, en introduisant des approximations du Hessian Hf . La stabilité peut également être introduite en limitant la distance parcourue en une seule itération.

Contraintes d'inégalité Des algorithmes spécialisés existent pour résoudre des programmes quadratiques plutôt que des programmes non linéaires généraux, et ceux-ci peuvent être utilisés pour générer des étapes de SQP. Une stratégie notable consiste à conserver un « ensemble actif » de contraintes qui sont actives au minimum par rapport à d ; alors les méthodes contraintes d'égalité ci-dessus peuvent être appliquées en ignorant les contraintes inactives. Les itérations d'optimisation de l'ensemble actif mettent à jour l'ensemble actif de contraintes en ajoutant des contraintes violées à l'ensemble actif et en supprimant les contraintes d'inégalité hj pour lesquelles f · hj ≤ 0 comme dans notre discussion des conditions KKT.

9.3.2 Méthodes barrières

Une autre option pour minimiser en présence de contraintes est de changer les contraintes en termes d'énergie. Par exemple, dans le cas des contraintes d'égalité, nous pourrions minimiser un objectif "augmenté" comme suit :

$$f\rho(x) = f(x) + \rho g(x) \quad {}^{2}_{2}$$

Notez que prendre ρ → ∞ forcera g(x) à être aussi petit que possible, donc finalement nous atteindrons g(x) ≈0. Ainsi, la méthode barrière d'optimisation contrainte applique des techniques itératives d'optimisation sans contrainte à fρ et vérifie dans quelle mesure les contraintes sont satisfaites ; s'ils ne sont pas dans une tolérance donnée, ρ est augmenté et l'optimisation continue en utilisant l'itération précédente comme point de départ.

Les méthodes barrières sont simples à mettre en œuvre et à utiliser, mais elles peuvent présenter des modes de défaillance pernicieux. En particulier, à mesure que ρ augmente, l'influence de f sur la fonction objectif diminue et la Hessienne de fρ devient de plus en plus mal conditionnée.

Les méthodes de barrière peuvent également être appliquées aux contraintes d'inégalité. Ici on doit s'assurer que hi(x) ≥ 0 pour tout i ; les choix typiques de fonctions de barrière pourraient inclure 1/hi(x) (la "barrière inverse") ou − log hi(x) (la "barrière logarithmique").

9.4 Programmation convexe

D'une manière générale, les méthodes comme celles que nous avons décrites pour l'optimisation sous contrainte offrent peu ou pas de garanties sur la qualité de la sortie. Certes, ces méthodes sont incapables d'obtenir des minima globaux sans une bonne estimation initiale x0, et dans certains cas, par exemple lorsque le Hessian
près de x n'est pas défini positif, ils peuvent ne pas converger du tout.

Il existe une exception notable à cette règle, qui apparaît dans un certain nombre d'optimisations importantes : la programmation convexe. L'idée ici est que lorsque f est une fonction convexe et que l'ensemble des possibles lui-même est convexe, alors l'optimisation possède un minimum unique. Nous avons déjà défini une fonction convexe, mais nous devons comprendre ce que cela signifie pour un ensemble de contraintes d'être
convexe:

Définition 9.3 (Ensemble convexe). Un ensemble S Rn est convexe si pour tout x,y S, le point tx + (1 − t)y est aussi dans S pour tout t [0, 1].

Comme le montre la figure NUMBER, intuitivement un ensemble est convexe si sa forme limite ne peut pas se plier à la fois vers l'intérieur et vers l'extérieur.

Exemple 9.7 (Cercles). Le disque {x Rn : x2 ≤ 1} est convexe, alors que le cercle unitaire {x Rn : x2 = 1} ne l'est pas.

Il est facile de voir qu'une fonction convexe a un minimum unique même lorsque cette fonction est restreinte à un domaine convexe. En particulier, si la fonction avait deux minima locaux, alors la ligne de points entre ces minima doit donner des valeurs de f non supérieures à celles des extrémités.

De fortes garanties de convergence sont disponibles pour les optimisations convexes qui garantissent de trouver le minimum global tant que f est convexe et que les contraintes sur g et h forment un ensemble réalisable convexe. Ainsi, un exercice précieux pour presque tous les problèmes d'optimisation consiste à vérifier s'il est convexe, car une telle observation peut augmenter la confiance dans la qualité de la solution et les chances de succès d'un facteur important.

Un nouveau domaine appelé programmation convexe disciplinée tente d'enchaîner des règles simples sur la convexité pour générer des optimisations convexes (CITE CVX), permettant à l'utilisateur final de combiner des termes d'énergie convexes simples et des contraintes tant qu'ils satisfont aux critères rendant l'optimisation finale convexe. Les déclarations utiles sur la convexité dans ce domaine incluent les suivantes :

- L'intersection des ensembles convexes est convexe ; ainsi, l'ajout de plusieurs contraintes convexes est une opération autorisée.
- La somme des fonctions convexes est convexe.
- Si f et g sont convexes, alors $h(x) \equiv \max\{ f(x), g(x)\}$.
- Si f est une fonction convexe, l'ensemble $\{x : f(x) \leq c\}$ est convexe.

Des outils tels que la bibliothèque CVX aident à séparer la mise en œuvre d'objectifs convexes assortis de leur minimisation.

Exemple 9.8 (Programmation convexe).

- Le problème des moindres carrés non négatifs de l'exemple 9.3 est convexe car $Ax - b_2$ est un problème convexe fonction de x et l'ensemble $x \geq 0$ est convexe.
- Le problème de programmation linéaire de l'exemple 9.6 est convexe car il a un objectif linéaire et des contraintes.
- On peut inclure x1 dans un objectif d'optimisation convexe en introduisant une variable y. Pour ce faire, on ajoute des contraintes $y_i \geq x_i$ et $y_i \geq -x_i$ pour chaque i et un objectif $\sum_i y_i$. Cette somme a des termes au moins aussi grands que $|x_i|$ et que l'énergie et les contraintes sont convexes. Au minimum, nous devons avoir $y_i = |x_i|$ puisqu'on a contraint $y_i \geq |x_i|$ et nous souhaitons minimiser l'énergie. Les bibliothèques convexes « disciplinées » peuvent effectuer de telles opérations dans les coulisses sans révéler ces substitutions à l'utilisateur final.

Un exemple particulièrement important d'optimisation convexe est la programmation linéaire de l'exemple 9.6.
Le célèbre algorithme du simplexe garde une trace des contraintes actives, résout le x résultant à l'aide d'un système linéaire et vérifie si l'ensemble actif doit être mis à jour ; aucune approximation de Taylor n'est nécessaire car l'ensemble objectif et réalisable sont donnés par des machines linéaires. Les stratégies de programmation linéaire des points intérieurs telles que la méthode des barrières sont également efficaces pour ces problèmes. Pour cette raison, les programmes linéaires peuvent être résolus à grande échelle - jusqu'à des millions ou des milliards de variables ! - et apparaissent souvent dans des problèmes tels que la planification ou la tarification.

9.5 Problèmes

- Dériver le simplexe ?
- Dualité de programmation linéaire

Chapitre 10

Solveurs linéaires itératifs

Dans les deux chapitres précédents, nous avons développé des stratégies pour résoudre une nouvelle classe de problèmes en impliquant la minimisation d'une fonction f(x) avec ou sans contraintes sur x. Ce faisant, nous avons assoupli notre point de vue de l'algèbre linéaire numérique et en particulier de l'élimination gaussienne selon laquelle nous devons trouver une solution exacte à un système d'équations et nous nous sommes plutôt tournés vers des schémas itératifs qui sont garantis pour approcher de mieux en mieux le minimum d'une fonction au fur et à mesure qu'ils itérer de plus en plus. Même si nous ne trouvons jamais exactement le minimum, nous savons que nous finirons par trouver un x0 avec f(x0) ≈0 avec des niveaux de qualité arbitraires, en fonction du nombre d'itérations que nous exécutons.

Nous avons maintenant une reprise de notre problème préféré de l'algèbre linéaire numérique, résolvant Ax = b pour x, mais appliquez une approche itérative plutôt que de vous attendre à trouver une solution sous forme fermée. Cette stratégie révèle une nouvelle classe de solveurs de systèmes linéaires capables de trouver des approximations fiables de x en très peu d'itérations. Nous avons déjà suggéré comment aborder cela dans notre discussion sur l'algèbre linéaire, en suggérant que les solutions aux systèmes linéaires sont des minima de l'énergie Ax − b entre autres. 22,

Pourquoi s'embêter à dériver encore une autre classe de solveurs de systèmes linéaires ? Jusqu'à présent, la plupart de nos approches directes nous obligent à représenter A sous la forme d'une matrice n × n complète, et des algorithmes tels que LU, QR ou la factorisation de Cholesky prennent tous environ O(n tiales raisons d'essayer 3) temps. Il y a deux cas à garder à l'esprit pour poten
des schémas itératifs :

1. Lorsque A est creux, des méthodes comme l'élimination gaussienne ont tendance à induire un remplissage, ce qui signifie que même) si A contient O(n) valeurs non nulles, les étapes intermédiaires d'élimination peuvent introduire O(n valeurs non nulles. Cette 2 propriété peut rapidement provoquer un manque de mémoire dans les systèmes d'algèbre linéaire. En revanche, les algorithmes de ce chapitre exigent seulement que vous sachiez appliquer A aux vecteurs, ce qui peut être fait en temps proportionnel au nombre de valeurs non nulles dans une matrice.

2. Nous pouvons vouloir vaincre le O(n 3) runtime des techniques standard de factorisation matricielle. En particulier, si un schéma itératif peut découvrir une solution assez précise de Ax = b en quelques itérations, les temps d'exécution peuvent être considérablement réduits.

Notez également que bon nombre des méthodes d'optimisation non linéaires dont nous avons parlé, en particulier celles qui dépendent d'un pas de type Newton, nécessitent de résoudre un système linéaire à chaque itération ! Ainsi, la formulation du solveur le plus rapide possible peut faire une différence considérable lors de la mise en œuvre de méthodes d'optimisation à grande échelle qui nécessitent une ou plusieurs résolutions linéaires par itération. En fait, dans ce cas, une résolution inexacte mais rapide d'un système linéaire pourrait être acceptable, car elle alimente de toute façon une technique itérative plus large.

Veuillez noter qu'une grande partie de notre discussion est due au CITE, bien que notre développement puisse être un peu plus court compte tenu du développement dans les chapitres précédents.

10.1 Descente en gradient

Nous concentrerons notre discussion sur la résolution de Ax =b où A a trois propriétés :

1. A Rn×n est carré
2. A est symétrique, c'est-à-dire que A = A
3. A est définie positive, c'est-à-dire que pour tout x =0, x Ax > 0

Vers la fin de ce chapitre, nous allons assouplir ces hypothèses. Pour l'instant, notez que nous pouvons remplacer Ax = b par les équations normales A Ax = A b pour satisfaire ces critères, bien que, comme nous l'avons vu, cette substitution puisse créer des problèmes de conditionnement numérique.

10.1.1 Dérivation du schéma itératif

Dans ce cas, il est facile de vérifier que les solutions de Ax = b sont des minima de la fonction f(x) donnée par la forme quadratique

$$f(x) \equiv \frac{1}{2} x Ax - bx + c \quad \text{pour tout}$$

c R. En particulier, prendre la dérivée de f montre

$$f(x) = Ax - b,$$

et le réglage f(x) =0 donne le résultat souhaité.

Plutôt que de résoudre f(x) = 0 directement comme nous l'avons fait dans le passé, supposons que nous appliquions stratégie de descente de gradient à cette minimisation. Rappelons l'algorithme de descente de gradient de base :

1. Calculer la direction de recherche d k ≡ − f(xk−1) =b − Axk−1 .
2. Définir xk ≡ xk−1 + αkd k , où αk est choisi tel que f(xk) < f(xk−1)

Pour une fonction générique f , décider de la valeur de αk peut être un problème difficile de « recherche de ligne » unidimensionnel, se résumant à minimiser f(xk−1 + αkd k) en fonction d'une seule variable αk ≥ 0. Pour notre choix particulier de la forme quadratique $f(x) = \frac{1}{2} x Ax - bx + c$, cependant, nous pouvons effectuer une recherche linéaire sous forme fermée. En particulier, définir

$$\begin{aligned} g(\alpha) &\equiv f(x + \alpha d) \\ &= \frac{1}{2}(x + \alpha d) A(x + \alpha d) - b(x + \alpha d) + c \\ &= \frac{1}{2}(x Ax + 2\alpha x Ad + \alpha^2 d Ad) - bx - \alpha b d + c \text{ par symétrie de A} \\ &= \frac{\alpha^2}{2} dAd + \alpha(x Ad - b d) + \text{const.} \end{aligned}$$

$$= \frac{dg}{d\alpha}(\alpha) = \alpha dAd + d(Ax - b)$$

Ainsi, si l'on veut minimiser g par rapport à α, on choisit simplement

$$a = \frac{d(b - Ax)\, dAd}{}$$

En particulier, pour la descente de gradient nous avons choisi $d_k = b - Ax_k$, donc en fait α_k prend une belle forme :

$$ak = \frac{jj\ kk}{d_{\text{Annonce}_{kk}}}$$

En fin de compte, notre formule de recherche de ligne donne le schéma de descente de gradient itératif suivant pour résoudre Ax =b dans le cas défini positif symétrique :

$$d_k = b - Axk-1$$

$$ak = \frac{jj\ kk}{d_{k\ \text{Annonce}_k}}$$

$$xk = xk-1 + \alpha k d\ k$$

10.1.2 Convergence

Par construction, notre stratégie de descente de gradient décroît f(xk) lorsque $k \to \infty$. Même ainsi, nous n'avons pas montré que l'algorithme atteint la valeur minimale possible de f , et nous n'avons pas été en mesure de caractériser combien d'itérations nous devrions exécuter pour atteindre un niveau de confiance raisonnable que Axk ≈b .

Une stratégie simple pour comprendre la convergence de l'algorithme de descente de gradient pour notre choix de f est d'examiner le changement d'erreur vers l'arrière d'une itération à l'autre.[1] Supposons que la solution que nous recherchons soit Ax =b. Ensuite, x nous pouvons étudier le rapport de retour erreur d'itération en itération :

$$Rk \equiv \frac{f(xk) - f(x\quad)}{f(xk-1) - f(x\quad)}$$

Évidemment, borner Rk < β < 1 pour certains β montre que la descente de gradient converge.

Pour plus de commodité, nous pouvons développer f(xk) :

$$\begin{aligned}
f(xk) &= f(xk-1 + \alpha k d\ k) \text{ par notre schéma itératif} \\
&= \frac{1}{2}(xk-1 + \alpha k d\ k)\, A(xk-1 + \alpha k d\ k) - b\ (xk-1 + \alpha k d\ k) + c \\
&= f(xk-1) + \alpha k d\ k\, Axk-1 + \alpha\ k\ 2\ 1\ (b\ d_k\ Ad\ k - \alpha k\ bd\ k \text{ par définition de f} \\
&= f(xk-1) + \alpha k d\quad k\ d\ k) + \quad \frac{-}{2}\alpha\ d_k\ Ad\ k - \alpha k\ bd\ k \text{ puisque } d_k = b - Axk-1 \\
&= f(xk-1) - \alpha k d\quad {}_{k} + \alpha\ kk\ 2\ d\ d_{\text{Annonce}_{kk}}
\end{aligned}$$

[1] Cet argument est présenté par exemple dans http://www-personal.umich.edu/~mepelman/teaching/IOE511/Handouts/511notes07-7.pdf.

$$= f(xk-1) - d \frac{jj\ kk}{k\ \text{Annonce}_k} jj\ k + k\ 2 \frac{1}{} \frac{jj\ kk}{d_k\ \text{Annonce}_k}^2 d \text{ Ad } k \text{ par définition de } \alpha k\ k$$

$$= f(xk-1) - \frac{(d^k\ dk)^2}{d\ \text{Annonce}_{kk}} 1 + \frac{(j_k\ dk)^2}{2j\ \text{Annonce}_{kk}} = f(xk-1) - 2d \frac{(d_k\ dk)^2}{\text{Annonce}_{kk}}$$

Ainsi, nous pouvons revenir à notre fraction :

$$Rc = \frac{f(xk-1) - \frac{(d_k\ d_k)^2}{2j_k\ \text{Annonce}_k} - f(x}{f(xk-1) - f(x\ \ )} \text{ par notre formule pour } f(xk)$$

$$= 1 - \frac{(d_k\ dk)^2}{2d\ k\ \text{Ad } k(f(xk-1) - f(x\ \ ))}$$

Remarquez que $Ax\ \ = b$, donc on peut écrire :

$$f(xk-1) - f(x\ \ ) = \frac{1}{}\ x\ k-1Axk-1 - b\ xk-1 + c - 2\ \ \ \ \frac{1}{}\ (x\ \ )\ b - bx\ 2\ \ \ + c$$

$$= \frac{1}{}\ x\ k-1Axk-1 - b\ xk-1 - 2\ 1\ \ \frac{1}{2} b \text{ UNE } -1b$$

$$= -(Axk-1 - b)\ A\ 2\ 1\ d^{-1}\ (Axk-1 - b) \text{ par symétrie de A}$$

$$= \frac{}{k\ 2}\ A^{-1}\ d\ k \text{ par définition de } d_k$$

Ainsi,

$$Rk = 1 - 2d\ k \frac{(d_k\ dk)^2}{\text{Ad } k(f(xk-1) - f(x\ \ ))}$$

$$= 1 - \text{ré} \frac{(d_k\ \text{ré}\ k)^2}{\text{Ad}_{k \cdot d\ k\ k}\ A-1d\ k} \text{ par notre dernière simplification}$$

$$= 1 - \text{ré} \frac{d_k\ d_k}{k\ \text{Annonce}_k} \cdot \frac{d_k\ d_k}{d\ A-1d\ k\ k}$$

$$\leq 1 - \min dAd \frac{1}{d=1} \quad \min_{d=1} \frac{1}{dA-1d} \text{ puisque cela rend le deuxième terme plus petit}$$

$$1 - \max d=1 \quad dAd = \max_{d=1}\ {}^{-1}\ dA\ {}^{-1}\ d$$

$\sigma\min = \frac{1 -}{\sigma\max\ 1}$ où $\sigma\min$ et $\sigma\max$ sont les valeurs singulières minimale et maximale de A

$$= 1 - \frac{}{\text{conde A}}$$

Il a fallu une quantité considérable d'algèbre, mais nous avons prouvé un fait important :

La convergence de la descente de gradient sur f dépend du conditionnement de A.

C'est-à-dire que plus A est conditionné, plus la descente de gradient convergera rapidement. De plus, puisque cond A ≥ 1, nous savons que notre stratégie de descente de gradient ci-dessus converge inconditionnellement , vers x bien que la convergence puisse être lente lorsque A est mal conditionné.

La figure NUMBER illustre le comportement de descente de gradient pour des matrices bien et mal conditionnées. Comme vous pouvez le voir, la descente de gradient peut avoir du mal à trouver le minimum de notre fonction quadratique f lorsque les valeurs propres de A ont une large dispersion.

10.2 Dégradés conjugués

Rappelons que résoudre Ax = b pour A Rn×n a pris la ^{3}) temps. Réexaminer la descente de gradient stratégie O(n ci-dessus, on voit que chaque itération ^{2}) temps, puisqu'il faut calculer matrice-vecteur prend O(n produits entre A, xk−1 et d k . Ainsi, si la descente de gradient prend plus de n itérations, on On aurait aussi bien pu appliquer l'élimination gaussienne, qui retrouvera la solution exacte dans le même laps de temps.Malheureusement, nous ne sommes pas en mesure de montrer que la descente de gradient doit prendre un nombre fini d'itérations, et en fait, dans les cas mal conditionnés, elle peut prendre un grand nombre d'itérations pour trouver le minimum.

Pour cette raison, nous allons concevoir un algorithme dont la convergence est garantie en au plus n étapes,) en préservant le O(n ^{3} synchronisation du pire cas pour la résolution de systèmes linéaires. En chemin, nous trouverons que cet algorithme présente en fait de meilleures propriétés de convergence dans l'ensemble, ce qui en fait un choix raisonnable même si nous ne l'exécutons pas jusqu'à la fin.

10.2.1 Motivations

Notre dérivation de l'algorithme des gradients conjugués est motivée par une observation assez simple. Supposons que nous connaissions la solution x dans Ax =b. Ensuite, nous pouvons écrire notre forme quadratique f d'une manière différente :

$$
\begin{aligned}
f(x) &= \tfrac{1}{2} x^{T} Ax - bx + c \quad \text{par définition} \\
&= \tfrac{1}{2}(x - x^{*})^{T} A(x - x^{*}) + x^{*T} Ax - \tfrac{1}{2}(x - x^{*})^{T} Ax^{*} - bx + c \\
&\qquad \text{en additionnant et en soustrayant les mêmes termes} \\
&= \tfrac{1}{2}(x - x^{*})^{T} A(x - x^{*}) + bx - \tfrac{1}{2}(x - x^{*})^{T} b - bx + c \quad \text{puisque } Ax^{*} = b \\
&= \tfrac{1}{2}(x - x^{*})^{T} A(x - x^{*}) + \text{const.} \quad \text{puisque les termes } xb \text{ s'annulent}
\end{aligned}
$$

Ainsi, à décalage constant f est égal au produit x inconnu mais $\tfrac{1}{2}(x - x^{*})^{T} A(x - x^{*})$. Bien sûr, on le fait cette observation nous montre la nature de f ; c'est simplement mesurer la distance ≡ v Av. de x à x par rapport à la « A-norme » v En fait, $^2_{UN}$

puisque A est symétrique et défini positif, même s'il peut être lent à réaliser en pratique, on sait qu'il peut être factorisé en utilisant la stratégie de Cholesky comme A = LL. Avec cette factorisation en main, f prend une forme encore plus agréable :

$$f(x) = \tfrac{1}{2} L^{T}(x - x^{*}) \; ^2 + \text{const.}^2$$

Puisque L est une matrice inversible, cette norme est bien une mesure de distance entre x et x · −

Définir y ≡ L x et y ≡ L x Alors, de ce nouveau point de vue, on minimise 2 . Bien sûr, si nous f(y) = . − 2 y −y pouvions vraiment arriver à ce point via la factorisation de Cholesky, l'optimisation f serait extrêmement facile, mais pour dériver un schéma pour cette minimisation sans L, nous considérons la possibilité de minimiser f en utilisant uniquement les recherches de ligne dérivées au §10.1.1. −

Nous faisons une observation simple sur la minimisation de notre fonction simplifiée f en utilisantune telle strat egie, illustrée à la figure NUMÉRO :

−

Proposition 10.1. Supposons que {w 1, . . . , w n} sont orthogonaux à Rn . Ensuite, f est minimisée en au plus n pas par recherche de ligne dans la direction w 1, puis la direction w 2, et ainsi de suite.

Preuve. Soient les colonnes de Q Rn×n les vecteurs w i ; Q est une matrice orthogonale. Puisque Q f(y) = y −y est orthogonal, en d'autres termes, nous tournons pour que w 1 soit le premier vecteur de base standard, w 2 soit le second, et ainsi de suite. on peut écrire 2 = Qy − Qy 2 2 Si on écrit z ≡ Qy et après la deuxième itération

z ≡ Qy et , alors clairement après la première itération on doit avoir z1 = z 1 , z2 = z 2 , ainsi de suite. Après n pas on atteint zn = z n , donnant le résultat souhaité. □

−

Ainsi, l'optimisation de f peut toujours être accomplie à l'aide de recherches sur n lignes tant que ces recherches sont dans des directions orthogonales. −

Tout ce que nous avons fait pour passer de f à f est de faire pivoter les coordonnées en utilisant L . Une telle transformation linéaire des lignes droites en lignes droites, donc faire une recherche de ligne à faire f le long d'un vecteur w est équivalent prend une recherche de ligne le long de (L) −1w sur notre fonction quadratique originale f . Inversement, si on fait n recherches linéaires sur f dans les directions vi telles que L vi ≡ w i soient orthogonales, alors d'après la proposition 10.1 on doit avoir trouvé x . Notez que demander w i · w = 0 revient au même que demander j

0 = w je · w j = (L vi) (L vj) = v je (LL)vj = v je Avj .

Nous venons de démontrer un corollaire important de la proposition 10.1. Définissez les vecteurs conjugués comme suit :

Définition 10.1 (Vecteurs A-conjugués). Deux vecteurs v, w sont A -conjugués si v Aw = 0.

Ensuite, sur la base de notre discussion, nous avons montré:

Proposition 10.2. Supposons que {v1, . . . ,vn} sont A-conjugués. Ensuite, f est minimisé en au plus n pas par recherche de ligne dans la direction v1, puis la direction v2, et ainsi de suite.

À un niveau élevé, l'algorithme des gradients conjugués applique simplement cette proposition, générant et recherchant le long de directions A-conjuguées plutôt que de se déplacer le long de − f . Notez que ce résultat peut sembler quelque peu contre-intuitif : nous ne nous déplaçons pas nécessairement le long de la direction de descente la plus raide, mais demandons plutôt que notre ensemble de directions de recherche satisfasse un critère global pour nous assurer de ne pas répéter le travail. Cette configuration garantit la convergence dans un nombre fini d'itérations et reconnaît la structure de f en termes de f discuté ci-dessus.

Rappelons que nous avons motivé les directions A-conjuguées en notant qu'elles sont orthogonales après application de L à partir de la factorisation A = LL. De ce point de vue, on a affaire à deux produits scalaires : xi · xj et yi · yj ≡ (L xi) · (L xj) = x i LLxj = x i Axj . Ces deux produits figureront dans notre discussion ultérieure en quantités égales, nous désignons donc le "produit interne A" par

u,vA ≡ (L u) · (L v) = u Moy.

10.2.2 Sous-optimalité de descente de gradient

Jusqu'à présent, nous savons que si nous pouvons trouver n directions de recherche A-conjuguées, nous pouvons résoudre Ax = b en n étapes via des recherches linéaires le long de ces directions. Il reste à découvrir une stratégie pour trouver ces directions aussi efficacement que possible. Pour ce faire, nous examinerons une autre propriété de l'algorithme de descente de gradient qui inspirera une approche plus raffinée.

Supposons que nous soyons en xk lors d'une méthode itérative de recherche linéaire sur f(x) ; on appellera direction de descente la plus raide de f en xk le résidu rk ≡ b − Axk . Nous ne pouvons pas décider de faire une recherche de ligne le long de rk comme dans la descente de gradient, car les directions de gradient ne sont pas nécessairement A-conjuguées. Ainsi, en généralisant légèrement, nous trouverons xk+1 via une recherche de ligne le long d'une direction encore indéterminée vk+1 .

À partir de notre dérivation de la descente de gradient, nous devrions choisir xk+1 = xk + αk+1vk+1 , où αk+1 est donné par

$$ak+1 = \frac{v_{k+1}\, rk}{v_{k+1} Avk+1}.$$

En appliquant cette expansion de xk+1 , nous pouvons écrire une formule de mise à jour alternative pour le résidu :

rk+1 =b − Axk+1 =b −

A(xk + αk+1vk+1) par définition de xk+1 = (b − Axk) −

αk+1Avk+1 =rk − αk+1Avk+1 par

définition ofrk

Cette formule est valable quel que soit notre choix de vk+1 et peut être appliquée à n'importe quelle méthode de recherche de ligne itérative.

Dans le cas de la descente de gradient, cependant, nous avons choisi vk+1 ≡ rk . Ce choix donne une relation de récurrence :

rk+1 =rk − αk+1Ark .

Cette formule simple conduit à une proposition instructive :

Proposition 10.3. Lors de la descente de gradient sur f , span{r0, . . . ,rk} = étendue{r0, Ar0, . . . , Un kr0}.

Preuve. Cette affirmation découle inductivement de notre formule forrk+1 ci-dessus. □

La structure que nous découvrons commence à ressembler beaucoup aux méthodes du sous-espace de Krylov évoquées au chapitre 5 : ce n'est pas une erreur !

La descente de gradient arrive à xk en se déplaçant le long de r0, puis r1, et ainsi de suite jusqu'à rk . Ainsi, au final on sait que l'itérée xk de descente de gradient sur f se situe quelque part dans le plan x0 + A k−1r0}, par la proposition 10.3. {r0, Ar0, . . . , il n'est pas vrai que si nous exécutons la descenteMalheureusement, c'est l'étendue {r0,r1, . . . ,rk−1} = x0 + étendue de gradient, l'itération xk est optimale dans ce sous-espace. En d'autres termes, en général, il peut arriver que

$$-x0 = \min_{v \;\text{span}\{r0,Ar0,...,Ak-1r0\}} xk \quad f(x0 + v) \arg$$

Idéalement, changer cette inégalité en égalité garantirait que la génération de xk+1 à partir de xk n'« annule » aucun travail effectué pendant les itérations 1 à k − 1.

Si nous réexaminons notre preuve de la proposition 10.1 en gardant ce fait à l'esprit, nous pouvons faire une observation suggérant comment nous pourrions utiliser la conjugaison pour améliorer la descente de gradient. En particulier, une fois que zi passe à z, il ne change jamais de valeur dans une itération future. En d'autres termes, après rotation de z à i , x la proposition suivante est vraie :

Proposition 10.4. Considérons xk comme le k-ième itéré du processus de la proposition 10.1 après une recherche le long de vk . Alors,

$$xk - x0 = \arg\min_{v \text{ étendue } \{v1,\ldots,vk\}} f(x0 + v)$$

Ainsi, dans le meilleur des mondes possibles, dans une tentative de surpasser la descente de gradient, nous pourrions espérer que A trouvera des directions A-conjuguées {v1, . . . ,vn} tel que l'étendue {v1, . . . ,vk} = étendue {r0, Ar0, . . . , pour chaque k; alors k−1r0} notre schéma itératif est garanti de ne pas faire pire que la descente de gradient au cours d'une itération donnée. Mais, avidement, nous souhaitons le faire sans orthogonalisation ni stockage de plus d'un nombre fini de vecteurs à la fois.

10.2.3 Génération de directions A-conjuguées

Bien sûr, étant donné n'importe quel ensemble de directions, nous pouvons les rendre A-orthogonales en utilisant une méthode comme l'orthogonalisation de Gram Schmidt. Malheureusement, l'orthogonalisation de {r0, Ar0, . . .} pour trouver l'ensemble des directions de recherche coûte cher et nous obligerait à maintenir une liste complète des directions vk ; cette construction dépasserait probablement les besoins en temps et en mémoire même de l'élimination gaussienne.
Nous révélerons une dernière observation sur Gram-Schmidt qui rend les gradients conjugués traitables en générant des directions conjuguées sans processus d'orthogonalisation coûteux.

En ignorant ces problèmes, nous pourrions écrire une « méthode des directions conjuguées » comme suit :

Mettre à jour le sens de recherche (mauvaise étape de Gram-Schmidt) : $vk = A^{k-1}r0 - \sum_{je<k} \frac{A\,k-1r0,viA}{vi\ ,viA} vi$

Recherche de ligne : $\alpha k = \frac{v\,k\ rk-1}{v_k\ Avk}$

Mise à jour de l'estimation : $xk = xk-1 + \alpha kvk$

Mettre à jour le résidu : $rk = rk-1 - \alpha kAvk$

Ici, nous calculons la k-ième direction de recherche vk simplement en projetant v1, . . . ,vk−1 sur le vecteur A k−1r0. Cet algorithme a évidemment la propriété span {v1, . . . ,vk} = étendue {r0, Ar0, . . . , A k−1r0} suggéré au §10.2.2, mais pose deux problèmes :

1. Semblable à l'itération de puissance pour les vecteurs propres, la puissance A k−1r0 est susceptible de ressembler principalement au premier vecteur propre de A, rendant la projection de plus en plus mal conditionnée
2. Nous devons conserver v1, . . . ,vk−1 autour pour calculer vk ; ainsi, chaque itération de cet algorithme nécessite plus de mémoire et de temps que la précédente.

Nous pouvons résoudre le premier problème de manière relativement simple. En particulier, en ce moment nous projetons les directions de recherche précédentes sur A k−1r0, mais en réalité nous pouvons projeter les directions précédentes de n'importe quel vecteur w tant que

$$w \quad \text{envergure } \{r0, Ar0, \ldots, A^{k-1}r0\} \backslash \text{span } \{r0, Ar0, \ldots, A^{k-2}r0\},$$

c'est-à-dire tant que w a une composante dans la nouvelle partie de l'espace.

Un choix alternatif de w avec cette propriété est le résidurk−1 . Cette propriété découle de la mise à jour résiduelle rk = rk−1 − αkAvk ; dans cette expression, on multiplie vk par A, en introduisant la nouvelle puissance de A dont on a besoin. Ce choix imite aussi plus fidèlement l'algorithme de descente de gradient, qui prenait vk =rk−1 . Ainsi, nous pouvons mettre à jour un peu notre algorithme :

$$\text{Mettre à jour le sens de recherche (mauvais Gram-Schmidt sur les résidus) : } vk = rk-1 - \sum_{je<k} \frac{rk-1\ ,viA\ vi}{vi\ ,viA}$$

$$\text{Recherche de ligne : } \alpha k = \frac{v_k\ rk-1}{v_k\ Avk}$$

$$\text{Mise à jour de l'estimation : } xk = xk-1 + \alpha kvk$$

$$\text{Mettre à jour le résidu : } rk = rk-1 - \alpha kAvk$$

Maintenant, nous ne faisons pas d'arithmétique impliquant le vecteur mal conditionné A k−1r0 mais nous avons toujours le problème de "mémoire" ci-dessus.

En fait, l'observation surprenante concernant l'étape d'orthogonalisation ci-dessus est que la plupart des termes de la somme sont exactement nuls ! Cette observation étonnante permet à chaque itération de gradients conjugués de se produire sans augmenter l'utilisation de la mémoire. Nous mémorisons ce résultat dans une proposition :

Proposition 10.5. Dans la méthode « direction conjuguée » ci-dessus, rk ,vA = 0 pour tout < k.

Preuve. Nous procédons par induction. Il n'y a rien à prouver pour le cas de base k = 1, supposons donc k > 1 et que le résultat est valable pour tout k < k. Par la formule de mise à jour résiduelle, nous savons :

$$rk\ ,vA = rk-1\ ,vA - \alpha kAvk\ ,vA = rk-1\ ,vA - \alpha kvk\ ,\ AvA,$$

où la seconde égalité découle de la symétrie de A.

Supposons d'abord < k − 1. Alors le premier terme de la différence ci-dessus est nul par récurrence.
De plus, par construction Av span {v1, . . . ,v+1}, donc puisque nous avons construit nos directions de recherche pour qu'elles soient A-conjuguées, nous savons que le deuxième terme doit également être nul.

Pour conclure la preuve, on considère le cas = k − 1. En utilisant la formule de mise à jour des résidus, on sait :

$$Avk-1 = \frac{1}{\alpha k-1}(rk-2 - rk-1)$$

La prémultiplication de byrk montre :

$$rk\ ,vk-1A = \frac{1}{\alpha k-1} r_k\ (rk-2 - rk-1)$$

La différence rk−2 −rk−1 vit dans l'étendue {r0, Ar0, . . . , la propositionA k−1r0}, par la formule de mise à jour des résidus. 10.4 montre que xk est optimal dans ce sous-espace. Puisque rk = − f(xk), cela implique A k−1r0}, car sinon il existerait une pour passer de xk à diminuer f . direction qu'il faut avoir rk span {r0, Ar0, . . . , dans le sous-espace

En particulier, cela montre le produit scalaire au-dessus de rk ,vk−1A = 0, comme on le souhaite. □

Ainsi, notre preuve ci-dessus montre que nous pouvons trouver une nouvelle direction vk comme suit :

$$vk = rk-1 - \sum_{je<k} \frac{rk-1\ ,viA}{vi\ ,viA} vi \text{ par la formule de Gram-Schmidt}$$

$$= rk-1 - \frac{rk-1\ ,vk-1A}{vk-1\ ,vk-1A} vk-1 \text{ car les termes restants s'annulent } vk-1\ ,vk-1A$$

Puisque la sommation sur i disparaît, le coût du calcul de vk ne dépend pas de k.

10.2.4 Formulation de l'algorithme des gradients conjugués

Maintenant que nous avons une stratégie qui produit des directions de recherche A-conjuguées avec relativement peu d'effort de calcul, nous appliquons simplement cette stratégie pour formuler l'algorithme des gradients conjugués.

En particulier, supposons que x0 est une estimation initiale de la solution de Ax = b, et prenons r0 ≡ b − Ax0.

Par commodité, prenons v0 ≡ 0. Ensuite, nous mettons à jour itérativement xk−1 en xk en utilisant une série d'étapes pour k = 1, 2, . . .:

$$\text{Mettre à jour le sens de recherche : } vk = rk-1 - \frac{rk-1\ ,vk-1A}{vk-1\ ,vk-1A} vk-1$$

$$\text{Recherche de ligne : } \alpha k = \frac{v_k\ rk-1}{v_k\ Avk}$$

$$\text{Mise à jour de l'estimation : } xk = xk-1 + \alpha k vk$$

$$\text{Mettre à jour le résidu : } rk = rk-1 - \alpha k Avk$$

Ce schéma itératif n'est qu'un ajustement mineur de l'algorithme de descente de gradient mais possède de nombreuses propriétés souhaitables par construction :

- f(xk) est majoré par celui du k-ième itéré de descente de gradient
- L'algorithme converge vers x en n étapes
- A chaque étape, l'itération xk est optimale dans le sous-espace couvert par les k premières directions de recherche

Afin d'obtenir une qualité numérique maximale des gradients conjugués, nous pouvons essayer de simplifier les valeurs numériques des expressions ci-dessus. Par exemple, si nous insérons la mise à jour de la direction de recherche dans la formule pour αk , par orthogonalité nous pouvons écrire :

$$ak = \frac{r_{k-1}\ rk-1}{v_k\ Avk}$$

Le numérateur de cette fraction est maintenant garanti non négatif sans précision numérique problèmes.

De même, nous pouvons définir une constante βk pour diviser la mise à jour de la direction de recherche en deux étapes :

$$\beta k \equiv - \frac{rk-1\ ,vk-1A}{vk-1\ ,vk-1A}$$

$$vk = rk-1 + \beta k vk-1$$

Nous pouvons simplifier notre formule pour βk :

$$\beta_k \equiv -\frac{r_{k-1}, v_{k-1} A}{v_{k-1}, v_{k-1} A}$$

$$= -\frac{r_{k-1} A v_{k-1}}{v_{k-1} A v_{k-1}} \text{ par définition de } v_{\cdot, \cdot U n}$$

$$= -\frac{r_{k-1}(r_{k-2} - r_{k-1})}{\alpha_{k-1} v_{k-1} A v_{k-1}} \text{ puisque } r_k = r_{k-1} - \alpha_k A v_k$$

$$= \frac{r_{k-1} r_{k-1}}{v_{k-1} A v_{k-1}} \text{ par un calcul inférieur à } \alpha_{k-1} v$$

$$= \frac{r_{k-1} r_{k-1}}{r_{k-2} r_{k-2}} \text{ par notre dernière formule pour } \alpha_k$$

Cette expression révèle que βk ≥ 0, une propriété qui n'aurait peut-être pas été vérifiée après des problèmes de précision numérique. Nous avons un calcul restant ci-dessous : (rk−2

$$r_{k-2} r_{k-1} = r_{k-2} - \alpha_{k-1} A v_{k-1}) \text{ par notre formule de mise à jour résiduelle}$$

$$= r_{k-2} r_{k-2} - \frac{r_{k-2} r_{k-2}}{v_{k-1} A v_{k-1}} r_{k-2} A v_{k-1} \text{ par notre formule pour } \alpha_k$$

$$= r_{k-2} - v_{k-1} A v_{k-1} \frac{r_{k-2} r_{k-2}}{v_{k-1} A v_{k-1}} \text{ par la mise à jour de } v_k \text{ et la A-conjugaison des } k-2 \; r_{k-2}$$

$$= 0, \text{ selon les besoins.}$$

Avec ces simplifications, nous avons une version alternative des gradients conjugués :

$$\text{Mettre à jour le sens de recherche : } \beta_k = \frac{r_{k-1} r_{k-1}}{r_{k-2} r_{k-2}}$$

$$v_k = r_{k-1} + \beta_k v_{k-1}$$

$$\text{Recherche de ligne : } \alpha_k = \frac{r_{k-1} r_{k-1}}{v_k A v_k}$$

$$\text{Mise à jour de l'estimation : } x_k = x_{k-1} + \alpha_k v_k$$

$$\text{Mettre à jour le résidu : } r_k = r_{k-1} - \alpha_k A v_k$$

Pour des raisons numériques, parfois plutôt que d'utiliser la formule de mise à jour forrk, il est conseillé d'utiliser la formule résiduelle =b − Axk . Cette formule nécessite une multiplication matrice-vecteur supplémentaire mais répare la «dérive» numérique causée par l'arrondi à précision finie. Notez qu'il n'est pas nécessaire de stocker une longue liste de résidus précédents ou de directions de recherche : les gradients conjugués occupent une quantité constante d'espace d'une itération à l'autre.

10.2.5 Conditions de convergence et d'arrêt

Par construction, l'algorithme des gradients conjugués (CG) est garanti de ne pas converger plus lentement que la descente de gradient sur f , tout en n'étant pas plus difficile à mettre en œuvre et en ayant un certain nombre d'autres

propriétés positives. Une discussion détaillée de la convergence CG sort du cadre de notre discussion, mais en général, l'algorithme se comporte mieux sur des matrices avec des valeurs propres uniformément réparties sur une petite plage. Une estimation grossière parallèle à notre estimation au §10.1.2 montre que l'algorithme CG satisfait :

$$\frac{f(xk) - f(x\quad)}{f(x0) - f(x\quad)} \leq 2\quad \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}^{k}$$

où κ ≡ cond A. Plus généralement, le nombre d'itérations nécessaires pour que le gradient conjugué atteigne une valeur d'erreur donnée peut généralement être borné par une fonction de $\sqrt{\kappa}$, tandis que les bornes de convergence de descente de gradient sont proportionnelles à κ.

Nous savons qu'il est garanti que les gradients conjugués convergent vers x exactement en n étapes, mais lorsque n est grand, il peut être préférable de s'arrêter plus tôt. En fait, la formule de βk se divisera par zéro lorsque le résidu devient très court, ce qui peut entraîner des problèmes de précision numérique près du minimum de f . Ainsi, en pratique, CG est généralement arrêté lorsque le rapport rk/r0 est suffisamment petit.

10.3 Préconditionnement

Nous disposons maintenant de deux schémas itératifs puissants pour trouver des solutions à Ax = b lorsque A est symétrique et définie positive : la descente de gradient et les gradients conjugués. Les deux stratégies convergent inconditionnellement, ce qui signifie que quelle que soit la supposition initiale x0 avec suffisamment d'itérations, nous nous rapprocherons arbitrairement de la vraie ; en fait, les gradients conjugués garantissent que nous atteindrons x solution x exactement en un nombre fini d'itérations. Bien sûr, le temps nécessaire pour atteindre une solution de Ax = b pour ces deux méthodes est directement proportionnel au nombre d'itérations nécessaires pour atteindre x dans une tolérance acceptable. Ainsi, il est logique d'ajuster une stratégie autant que possible pour minimiser le nombre d'itérations pour la convergence.

À cette fin, nous remarquons que nous sommes en mesure de caractériser les taux de convergence des deux algorithmes et de nombreuses autres techniques itératives connexes en termes de nombre de conditions cond A. Autrement dit, plus la valeur de cond A est petite, moins cela devrait prendre de temps. pour résoudre Ax = b. Notez que cette situation est quelque peu différente pour l'élimination gaussienne, qui prend le même nombre d'étapes quel que soit A ; en d'autres termes, le conditionnement de A affecte non seulement la qualité de la sortie des méthodes itératives mais aussi la vitesse à laquelle x est approché.

Bien entendu, pour toute matrice inversible P, il est vrai que résoudre PAx = Pb équivaut à résoudre Ax = b. L'astuce, cependant, est que le numéro de condition de PA n'a pas besoin d'être le même que celui de A; à l'extrême (inatteignable), bien sûr, si nous prenions $P = A^{-1}$, nous éliminerions complètement les problèmes de conditionnement ! Plus généralement, supposons $P \approx A^{-1}$ cond A, et il peut donc être conseillé . Ensuite, on attend cond PA d'appliquer P avant de résoudre le système linéaire. Dans ce cas, nous appellerons P un préconditionneur.

Si l'idée du préconditionnement semble séduisante, deux problèmes subsistent :

1. Alors que A peut être symétrique et défini positif, le produit PA en général ne jouira pas ces propriétés.
2. Il faut trouver $P \approx A^{-1}$ qui est plus facile à calculer que A^{-1} lui-même.

Nous abordons ces questions dans les sections ci-dessous.

10.3.1 CG avec préconditionnement

Nous concentrerons notre discussion sur les gradients conjugués car ils ont de meilleures propriétés de convergence, bien que la plupart de nos constructions s'appliquent assez facilement à la descente de gradient également. De ce point de vue, si nous examinons nos constructions au §10.2.1, il est clair que notre construction de CG dépend fortement à la fois de la symétrie et de la définition positive de A, donc exécuter CG sur PA ne convergera généralement pas hors de la boîte.

Supposons cependant que le préconditionneur P soit lui-même symétrique et défini positif. C'est une hypothèse raisonnable puisque A doit satisfaire ces propriétés. Alors, on peut encore écrire $a = EE$. Nous faisons le constat suivant :
Factorisation de Cholesky de l'inverse P Proposition^{-1}

10.6. Le numéro de condition de PA est le même que celui de $E^{-1}AE^{-}$.

Preuve. Nous montrons que PA et $E^{-1}AE^{-}$ ont les mêmes valeurs singulières ; le numéro de condition est le rapport entre la valeur singulière maximale et minimale, donc cette déclaration est plus que suffisante. En particulier, il est clair que $E^{-1}AE^{-}$ est symétrique et définie positive, donc ses vecteurs propres sont ses valeurs singulières. Ainsi, supposons $E^{-1}AE^{-}x = \lambda x$. Nous connaissons P . Ainsi, si nous prémultiplions les deux côtés de notre expression de vecteur propre par $E^{-1} = EE$, donc $P^{-1} = E^{-}E^{-1}$, nous trouvons $PAE^{-}x = \lambda E^{-}x$.
Définir $y \equiv E^{-}x$ montre $PAy = \lambda y$. Ainsi, PA et $E^{-1}AE^{-}$ ont tous deux des espaces propres complets et des valeurs propres identiques. □

Cette proposition implique que si nous faisons CG sur la matrice définie positive symétrique $E^{-1}AE^{-}$, nous recevrons les mêmes avantages de conditionnement que nous aurions si nous pouvions itérer sur PA. Comme dans notre preuve de la proposition 10.6, nous pourrions accomplir notre nouvelle résolution pour $y = E x$ en deux étapes :

1. Résolvez $E^{-1}AE^{-}y = E^{-1}b$.
2. Résoudre $x = E^{-}y$.

Trouver E ferait partie intégrante de cette stratégie mais est probablement difficile, mais nous prouverons bientôt qu'il n'est pas nécessaire.

En ignorant le calcul de E, nous pourrions accomplir l'étape 1 en utilisant CG comme suit :

$$\text{Mettre à jour le sens de recherche : } \beta_k = \frac{r_{k-1}\, r_{k-1}}{r_{k-2}\, r_{k-2}}$$

$$v_k = r_{k-1} + \beta_k v_{k-1}$$

$$\text{Recherche de ligne : } \alpha_k = \frac{r_{k-1}\, r_{k-1}}{v_k\, E^{-1}AE^{-}v_k}$$

$$\text{Mise à jour de l'estimation : } y_k = y_{k-1} + \alpha_k v_k$$

$$\text{Mettre à jour le résidu : } r_k = r_{k-1} - \alpha_k E^{-1}AE^{-}v_k$$

Ce schéma itératif convergera selon le conditionnement de notre matrice $E^{-1}AE^{-}$.

Définissons $\tilde{r}_k \equiv Er_k$, $\tilde{v}_k \equiv E^{-}v_k$, et $\tilde{x}_k \equiv Ey_k$. Si nous rappelons la relation $P^{-1} = E^{-}E$, réécrivez , nous pouvons notre itération de gradients conjugués préconditionnés en utilisant ces nouvelles variables :

$$\text{Mettre à jour le sens de recherche : } \beta_k = \frac{\tilde{r}_{k-1} P \tilde{r}_{k-1}}{\tilde{r}_{k-2} P \tilde{r}_{k-2}}$$

$$\tilde{v}_k = P\tilde{r}_{k-1} + \beta_k \tilde{v}_{k-1}$$

$$\text{Recherche de ligne : } \alpha_k = \frac{r_{k-1}^{T} P r_{k-1}}{\tilde{v}_k^{T} A \tilde{v}_k}$$

$$\text{Mise à jour de l'estimation : } x_k = x_{k-1} + \alpha_k \tilde{v}_k$$

$$\text{Mettre à jour le résidu : } \tilde{r}_k = \tilde{r}_{k-1} - \alpha_k A \tilde{v}_k$$

Cette itération ne dépend pas de la factorisation de Cholesky de P effectuée en utilisant P^{-1} du tout, mais au lieu de cela peut uniquement des applications de P et A. Il est facile de voir que le schéma $x_k \to x$ bénéficie des avantages , être si en fait ce du préconditionnement sans avoir besoin de factoriser le préconditionneur.

En remarque, un préconditionnement encore plus efficace peut être effectué en remplaçant A par PAQ pour une deuxième matrice Q, bien que cette deuxième matrice nécessitera des calculs supplémentaires pour s'appliquer. Cet exemple représente un compromis courant : si un préconditionneur lui-même prend trop de temps à s'appliquer dans une seule itération de CG ou d'une autre méthode, cela peut ne pas valoir le nombre réduit d'itérations.

10.3.2 Préconditionneurs communs

Trouver de bons préconditionneurs dans la pratique est autant un art qu'une science. Trouver le meilleur dépend de la structure approximation P de A ainsi P^{-1} de A, de l'application particulière à portée de main et
de suite. Cependant, même des approximations grossières peuvent aider considérablement la convergence, si rarement des applications de CG apparaissent qui n'utilisent pas de préconditionneur.

La meilleure stratégie pour formuler P est souvent spécifique à l'application, et un problème d'approximation technique intéressant implique la conception et le test de P assortis pour le meilleur préconditionneur.
Voici deux stratégies courantes :

- Un préconditionneur diagonal (ou « Jacobi ») prend simplement P comme la matrice obtenue en inversant les éléments diagonaux de A ; c'est-à-dire que P est la matrice diagonale avec les entrées 1/aii. Cette stratégie peut atténuer la mise à l'échelle non uniforme d'une ligne à l'autre, qui est une cause fréquente de mauvais conditionnement.

- Le préconditionneur inverse approché creux est formulé en résolvant un sous-problème $\min_P \|SAP - I\|_{Fro}$, où P est contraint d'être dans un ensemble S de matrices sur lesquelles il est moins difficile d'optimiser un tel objectif. Par exemple, une contrainte courante est de prescrire un modèle de parcimonie pour P, par exemple uniquement des non-zéros sur la diagonale ou où A a des non-zéros.

- Le préconditionneur incomplet de Cholesky facteurs $A \approx L\ L$ puis se rapproche de A^{-1} en résolvant les problèmes de substitution avant et arrière appropriés. Par exemple, une stratégie populaire consiste à passer par les étapes de la factorisation de Cholesky mais à ne sauvegarder que la sortie dans les positions (i, j) où $a_{ij} = 0$.

- Les valeurs non nulles dans A peuvent être considérées comme un graphe, et la suppression d'arêtes dans le graphe ou le regroupement de nœuds peut déconnecter des composants assortis ; le système résultant est une diagonale de bloc après la permutation des lignes et des colonnes et peut donc être résolu à l'aide d'une séquence de résolutions plus petites. Une telle stratégie de décomposition de domaine peut être efficace pour des systèmes linéaires issus d'équations différentielles telles que celles considérées dans le chapitre NUMBER.

Certains préconditionneurs sont accompagnés de limites décrivant les changements apportés au conditionnement de A après son remplacement par PA, mais pour la plupart, ce sont des stratégies heuristiques qui doivent être testées et affinées.

10.4 Autres schémas itératifs

Les algorithmes que nous avons développés en détail dans ce chapitre s'appliquent pour résoudre Ax = b lorsque A est carré, symétrique et défini positif. Nous nous sommes concentrés sur ce cas car il apparaît si souvent dans la pratique, mais il existe des cas où A est asymétrique, indéfini ou même rectangulaire. Il n'entre pas dans le cadre de notre discussion de dériver des algorithmes itératifs dans chaque cas, car beaucoup nécessitent une analyse spécialisée ou un développement avancé, mais nous résumons ici certaines techniques de haut niveau (CITE EACH):

- Les méthodes de découpage décomposent A = M − N et notent que Ax = b est équivalent à Mx = Nx +b. Si M est facile à inverser, alors un schéma en virgule fixe peut être dérivé en écrivant Mxk = Nxk−1 +b (CITE) ; ces techniques sont faciles à mettre en oeuvre mais ont une convergence dépendant du spectre de la matrice G = M-1N et en particulier peuvent diverger lorsque le rayon spectral de G est supérieur à un. Un choix populaire de M est la diagonale de A. Des méthodes telles que la sur-relaxation successive (SOR) pondèrent ces deux termes pour une meilleure convergence.
- La méthode du résidu de l'équation normale du gradient conjugué (CGNR) applique simplement l'algorithme CG aux équations normales A Ax = A b. Cette méthode est simple à mettre en œuvre et garantie de converger tant que A est de rang complet, mais la convergence peut être lente en raison d'un mauvais conditionnement de AA comme discuté au chapitre NUMBER.
- La méthode de l'erreur de l'équation normale du gradient conjugué (CGNE) résout de manière similaire AAy =b ; alors la solution de Ax =b est simplement A y.
- Des méthodes telles que MINRES et SYMMLQ s'appliquent aux matrices A définies symétriques mais pas nécessairement positives en remplaçant notre forme quadratique f(x) par g(x) ≡ b − Ax2 ; cette fonction g est minimisée aux solutions de Ax = b quelle que soit la définition de A.
- Compte tenu du mauvais conditionnement de CGNR et CGNE, les algorithmes LSQR et LSMR minimisent également g(x) avec moins d'hypothèses sur A, en permettant notamment la solution des systèmes de moindres carrés.
- Les méthodes généralisées incluant GMRES, QMR, BiCG, CGS et BiCGStab résolvent Ax =b avec la seule mise en garde que A est carré et inversible. Ils optimisent des énergies similaires mais doivent souvent stocker plus d'informations sur les itérations précédentes et peuvent devoir factoriser des matrices intermédiaires pour garantir une convergence avec une telle généralité.
- Enfin, les méthodes de Fletcher-Reeves, Polak-Ribi`ere et autres reviennent au problème plus général de la minimisation d'une fonction non quadratique f , en appliquant des étapes de gradient conjugué pour trouver de nouvelles directions de recherche de ligne. Les fonctions f qui sont bien approchées par les quadratiques peuvent être minimisées très efficacement en utilisant ces stratégies, bien qu'elles n'utilisent pas nécessairement le hessien ; par exemple, la méthode de Fletcher-Reeves remplace simplement le résidu dans les itérations CG par le gradient négatif − f . Il est possible de caractériser la convergence de ces méthodes lorsqu'elles sont accompagnées de stratégies de recherche linéaire suffisamment efficaces.

Beaucoup de ces algorithmes sont presque aussi faciles à implémenter que CG ou descente de gradient, et de nombreuses implémentations existent qui nécessitent simplement la saisie de A et b. De nombreux algorithmes énumérés ci-dessus nécessitent l'application à la fois de A et de A, ce qui peut être un défi technique dans certains cas. En règle générale, plus une méthode est généralisée - c'est-à-dire que moins une méthode fait d'hypothèses sur la structure de la matrice A - plus elle est susceptible de nécessiter d'itérations pour compenser ce manque d'hypothèses. Cela dit, il n'y a pas de règles absolues simplement en regardant Un schéma itératif le plus réussi, bien qu'il existe une discussion théorique limitée comparant les avantages et les inconvénients de chacune de ces méthodes (CITE).

10.5 Problèmes

- Dériver CGNR et/ou CGNE
- Déduire MINRES
- Dériver Fletcher-Reeves
- Diapositive 13 de http://math.ntnu.edu.tw/~min/matrix_computation/Ch4_Slide4_CG_2011.pdf

Partie IV

Fonctions, dérivées et intégrales

Chapitre 11

Interpolation

Jusqu'à présent, nous avons dérivé des méthodes pour analyser les fonctions f , par exemple en trouvant leurs minima et leurs racines. Évaluer f(x) à un x Rn particulier peut être coûteux, mais une hypothèse fondamentale des méthodes que nous avons développées dans les chapitres précédents est que nous pouvons obtenir f(x) quand nous le voulons, quel que soit x.

Il existe de nombreux contextes où cette hypothèse n'est pas réaliste. Par exemple, si nous prenons une photo avec un appareil photo numérique, nous recevons une grille n × m de valeurs de couleur de pixel échantillonnant le continuum de lumière entrant dans un objectif d'appareil photo. Nous pourrions penser à une photographie comme une fonction continue de la position de l'image (x, y) à la couleur (r, g, b), mais en réalité nous ne connaissons la valeur de l'image qu'à nm emplacements séparés sur le plan de l'image. De même, dans l'apprentissage automatique et les statistiques, on ne nous donne souvent des échantillons d'une fonction qu'aux points où nous avons collecté des données, et nous devons les interpoler pour avoir des valeurs ailleurs ; dans un contexte médical, nous pouvons surveiller la réponse d'un patient à différentes doses d'un médicament, mais nous ne pouvons prédire ce qui se passera qu'à une dose que nous n'avons pas essayée explicitement.

Dans ces cas, avant de pouvoir minimiser une fonction, trouver ses racines ou même calculer des valeurs f(x) à des emplacements arbitraires x, nous avons besoin d'un modèle pour interpoler f(x) à tout Rn (ou à un sous-ensemble de celui-ci) étant donné un ensemble d'échantillons f(xi). Bien sûr, les techniques résolvant ce problème d'interpolation sont intrinsèquement approximatives, puisque nous ne connaissons pas les vraies valeurs de f , nous cherchons donc plutôt à ce que la fonction interpolée soit lisse et serve de prédiction «raisonnable» des valeurs de la fonction.

Dans ce chapitre, nous supposerons que les valeurs f(xi) sont connues avec une entière certitude ; dans ce cas, nous pourrions aussi bien penser que le problème étend f au reste du domaine sans perturber la valeur à aucun des emplacements d'entrée. Dans le chapitre NUMBER (WRITE ME IN 2014), nous examinerons le problème de régression, dans lequel la valeur f(xi) est connue avec une certaine incertitude, auquel cas nous pouvons renoncer complètement à faire correspondre f(xi) en faveur de rendre f plus lisse .

11.1 Interpolation en une seule variable

Avant de considérer le cas le plus général, nous allons concevoir des méthodes d'interpolation de fonctions d'une seule variable f : R → R. En entrée, nous prendrons un ensemble de k couples (xi , yi) avec l'hypothèse f(xi) = yi ; notre travail est de trouver f(x) pour x {x1, . . . , xk}.

Notre stratégie dans cette section et dans d'autres s'inspirera de l'algèbre linéaire en écrivant f(x) dans une base. Autrement dit, l'ensemble de toutes les fonctions possibles f : R → R est beaucoup trop grand pour fonctionner et comprend de nombreuses fonctions qui ne sont pas pratiques dans un cadre informatique. Ainsi, nous simplifions l'espace de recherche en forçant f à être écrit comme une combinaison linéaire de blocs de construction plus simples

fonctions de base. Cette stratégie est déjà familière dans le calcul de base : le développement de Taylor écrit des fonctions sur la base de polynômes, tandis que les séries de Fourier utilisent le sinus et le cosinus.

11.1.1 Interpolation polynomiale

L'interpolation la plus simple est peut-être de supposer que f(x) est dans R[x], l'ensemble des polynômes. Les polynômes sont lisses et il est simple de trouver un polynôme de degré k - 1 à travers k points d'échantillonnage.

En fait, l'exemple 3.3 donne déjà les détails d'une telle technique d'interpolation. Comme un rappel, supposons que l'on souhaite trouver f(x) ≡ a0 + a1x + a2x ; ici nos inconnues sont les valeurs a0, . . . , ak−1 . Brancher l'expression yi = f(xi) pour chaque i montre que le vecteur a satisfait le système k × k Vandermonde :

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{k-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{k-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{k-1} & x_{k-1}^2 & \cdots & x_{k-1}^{k-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{k-1} \end{bmatrix}$$

Ainsi, la réalisation d'une interpolation polynomiale de degré-k peut être accomplie en utilisant une résolution linéaire ak × k en appliquant nos stratégies génériques des chapitres précédents, mais en fait, nous pouvons faire mieux.

Une façon de penser à notre forme pour f(x) est qu'elle est écrite dans une base. Tout comme une base de Rn est un ensemble de n vecteurs linéairement indépendants v1, . . . ,vn, ici l'espace des polynômes de degré x k−1}. C'est peut-l'étendue des monômes {1, x, x R[x], mais notre choix ^{2}, . . . , être la base la plus évidente car k − 1 est écrit dans actuel a peu de propriétés qui le rendent utile pour le problème d'interpolation.
Une façon de voir ce problème est de tracer la séquence des fonctions 1, x, x k qui ^{2}, ^{3x}, . . . pour x [0, 1] ; dans ce ressembler. intervalle, il est facile de voir que lorsque k devient grand, les fonctions sont toutes à se

En continuant à appliquer notre intuition de l'algèbre linéaire, nous pouvons choisir d'écrire notre polynôme dans une base plus adaptée au problème à résoudre. Cette fois, rappelons qu'on nous donne k paires (x1, y1), . . . ,(xk , yk). Nous utiliserons ces points (fixes) pour définir la base d'interpolation de Lagrange φ1, . . . , φk en écrivant :

$$\varphi_i(x) \equiv \frac{\prod_{j=i}(x - x_j)}{\prod_{j=i}(x_i - x_j)}$$

Bien qu'elle ne soit pas écrite dans la base 1, x, x ^{2}, . . . , x^{k-1}, il est facile de voir que chaque φi est toujours un poly nomial de degré k − 1. De plus, la base de Lagrange a la propriété souhaitable suivante :

$$\varphi_i(x) = \begin{cases} 1 & \text{quand } = i \\ 0 & \text{sinon.} \end{cases}$$

Ainsi, trouver l'unique polynôme de degré k − 1 correspondant à nos paires (xi , yi) est facile dans la base de Lagrange :

$$f(x) \equiv \sum y_i \varphi_i(x)$$

En particulier, si on substitue x = xj on trouve : f(xj) = ∑

$$y_i \varphi_i(x_j)$$

= yj par notre expression pour φi(x) ci-dessus.

Ainsi, dans la base de Lagrange, nous pouvons écrire une formule fermée pour f(x) qui ne nécessite pas de résoudre le système de Vandermonde. L'inconvénient, cependant, est que chaque φi(x) prend un temps O(k) pour être évalué en utilisant la formule ci-dessus pour un x donné, donc trouver f(x) prend ^{2}) temps ; si on trouve les coefficients explicitement O(n ai du système de Vandermonde, cependant, l'évaluation le temps peut être réduit à O(n).

Hormis le temps de calcul, la base de Lagrange présente un inconvénient numérique supplémentaire. Notez que le dénominateur est le produit de plusieurs termes. Si les xi sont proches les uns des autres, alors le produit peut inclure de nombreux termes proches de zéro, nous divisons donc par un nombre potentiellement petit.
Comme nous l'avons vu, cette opération peut créer des problèmes numériques que nous souhaitons éviter.

Une base de polynômes de degré k − 1 qui tente de faire un compromis entre la qualité numérique des monômes et l'efficacité de la base de Lagrange est la base de Newton, définie comme suit :

$$\psi_i(x) = \prod_{j=1}^{i-1} (x - x_j)$$

On définit ψ1(x) ≡ 1. Remarquons que ψi(x) est un polynôme de degré i − 1. Par définition de ψi , il est clair que ψi(x) = 0 pour tout < i. Si on veut écrire f(x) = ∑i ciψi(x) et écrire cette observation plus explicitement, on trouve :

$$\begin{aligned} f(x_1) &= c_1\psi_1(x_1) \\ f(x_2) &= c_1\psi_1(x_2) + c_2\psi_2(x_2) \\ f(x_3) &= c_1\psi_1(x_3) + c_2\psi_2(x_3) + c_3\psi_3(x_3) \\ &\vdots \quad \vdots \end{aligned}$$

En d'autres termes, nous pouvons résoudre forc le système triangulaire inférieur suivant :

$$\begin{bmatrix} \psi_1(x_1) & 0 & 0 & \cdots & 0 \\ \psi_1(x_2) & \psi_2(x_2) & 0 & \cdots & 0 \\ \psi_1(x_3) & \psi_2(x_3) & \psi_3(x_3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \psi_1(x_k) & \psi_2(x_k) & \psi_3(x_k) & \cdots & \psi_k(x_k) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \text{beurk} \end{bmatrix}$$

Ce système peut être résolu en O(n ^{2}) temps en utilisant la substitution vers l'avant, plutôt que le O(n ^{3}) temps nécessaire pour résoudre le système de Vandermonde.

Nous avons maintenant trois stratégies d'interpolation de k points de données à l'aide d'un polynôme de degré k − 1 en l'écrivant dans les bases monôme, Lagrange et Newton. Tous trois représentent des compromis différents entre qualité numérique et rapidité. Une propriété importante, cependant, est que la fonction interpolée résultante f(x) est la même dans chaque cas. Plus explicitement, il existe exactement un polynôme de degré k − 1 passant par un ensemble de k points, donc puisque tous nos interpolants sont de degré k − 1, ils doivent avoir la même sortie.

11.1.2 Bases alternatives

Bien que les fonctions polynomiales se prêtent particulièrement à l'analyse mathématique, il n'y a aucune raison fondamentale pour laquelle notre base d'interpolation ne peut pas consister en différents types de fonctions. Par exemple, un résultat suprême de l'analyse de Fourier implique qu'une grande classe de fonctions est bien approchée par des sommes de fonctions trigonométriques cos(kx) et sin(kx) pour k N. Une construction

comme le système Vandermonde s'applique toujours dans ce cas, et en fait l'algorithme Fast Fourier Transform (qui mérite une discussion plus large) montre comment effectuer une telle interpolation encore plus rapidement.

Une plus petite extension du développement du §11.1.1 est aux fonctions rationnelles de la forme :

$$f(x) = \frac{p0 + p1x + p2x^2 + \cdots + pmx^m}{q0 + q1x + q2x^2 + \cdots + qnx^n}$$

Remarquez que si on nous donne k paires (xi , yi), alors nous aurons besoin de m + n + 1 = k pour que cette fonction soit bien définie. Un degré de liberté supplémentaire doit être fixé pour tenir compte du fait que la même fonction rationnelle peut être exprimée de plusieurs façons par une mise à l'échelle identique du numérateur et du dénominateur.

Les fonctions rationnelles peuvent avoir des asymptotes et d'autres modèles non réalisables en utilisant uniquement des polynômes, de sorte qu'elles peuvent être des interpolations souhaitables pour les fonctions qui changent rapidement ou qui ont des pôles. En fait, une fois m et n fixés, les coefficients pi et qi peuvent toujours être trouvés en utilisant des techniques linéaires en multipliant les deux côtés par le dénominateur :

$$yi(q0 + q1xi + q2xi^2 + \cdots + qnxi^n) = p0 + p1xi + p2xi^2 + \cdots + pmxi^m$$

Encore une fois, les inconnues dans cette expression sont les p et les q.

La flexibilité des fonctions rationnelles, cependant, peut causer certains problèmes. Par exemple, considérons l'exemple suivant : Exemple 11.1

(Échec de l'interpolation rationnelle, Bulirsch-Stoer §2.2). Supposons que nous souhaitions trouver f(x) avec les points de données suivants : (0, 1), (1, 2), (2, 2). Nous pourrions choisir m = n = 1. Alors, nos conditions linéaires deviennent :

$$q0 = p0$$

$$2(q0 + q1) = p0 + p1$$

$$2(q0 + 2q1) = p0 + 2p1$$

Une solution non triviale à ce système est :

$$p0 = 0$$

$$p1 = 2$$

$$q0 = 0$$

$$q1 = 1$$

Cela implique la forme suivante pour f(x) :

$$f(x) = \frac{2x}{x}$$

Cette fonction a une dégénérescence à x = 0, et en fait l'annulation de x au numérateur et au dénominateur ne donne pas f(0) = 1 comme on pourrait le souhaiter.

Cet exemple illustre un phénomène plus vaste. Notre système linéaire pour trouver les p et les q peut rencontrer des problèmes lorsque le dénominateur résultant ∑ px a une racine à l'un des xi fixes .
On peut montrer que lorsque c'est le cas, aucune fonction rationnelle n'existe avec le choix fixe de m et n interpolant les valeurs données. Une résolution partielle typique dans ce cas est présentée dans (CITE), qui incrémente m et n alternativement jusqu'à ce qu'une solution non triviale existe. D'un point de vue pratique, cependant, la nature spécialisée de ces méthodes est un bon indicateur que des stratégies d'interpolation alternatives peuvent être préférables lorsque les méthodes rationnelles de base échouent.

11.1.3 Interpolation par morceaux

Jusqu'à présent, nous avons construit nos stratégies d'interpolation en combinant des fonctions simples sur tout R. Lorsque le nombre k de points de données devient élevé, cependant, de nombreuses dégénérescences deviennent apparentes. Par exemple, la figure NUMBER montre des exemples dans lesquels l'ajustement de polynômes de haut degré aux données d'entrée peut produire des résultats inattendus. De plus, la figure NUMBER illustre comment ces stratégies sont non locales, ce qui signifie que la modification d'une seule valeur yi dans les données d'entrée peut modifier le comportement de f pour tous les x, même ceux qui sont éloignés du xi correspondant . D'une certaine manière, cette propriété est irréaliste : nous nous attendons à ce que seules les données d'entrée proches d'un x donné affectent la valeur de f(x), en particulier lorsqu'il existe un grand nuage de points d'entrée.

Pour ces raisons, lorsque nous concevons un ensemble de fonctions de base φ1, . . . , φk , une propriété souhaitable est non seulement qu'ils sont faciles à utiliser, mais aussi qu'ils ont un support compact :

Définition 11.1 (Support compact). Une fonction g(x) a un support compact s'il existe C R tel que g(x) = 0 pour tout x avec |x| > C

Autrement dit, les fonctions prises en charge de manière compacte n'ont qu'une plage finie de points dans laquelle elles peuvent prendre des valeurs non nulles.

Une stratégie courante pour construire des bases d'interpolation avec un support compact consiste à le faire par morceaux. En particulier, une grande partie de la littérature sur l'infographie dépend de la construction de polynômes par morceaux, qui sont définis en décomposant R en un ensemble d'intervalles et en écrivant un polynôme différent dans chaque intervalle. Pour ce faire, nous allons ordonner nos points de données de sorte que x1 < x2 < · · · < xk . Ensuite, deux exemples simples d'interpolations par morceaux sont les suivants :

- Constante par morceaux (Figure NUMBER) : Pour un x donné, trouvez le point de données xi en minimisant |x − xi | et définissons f(x) = yi .
- Linéaire par morceaux (Figure NOMBRE) : Si x < x1, prendre f(x) = y1, et si x > xk prendre f(x) = yk . Sinon, trouvez un intervalle avec x [xi , xi+1] et définissez

$$f(x) = y_{i+1} \cdot \frac{x - x_i}{x_{i+1} - x_i} + y_i \cdot \frac{x_{i+1} - x}{x_{i+1} - x_i}.$$

Plus généralement, on peut écrire un polynôme différent dans chaque intervalle [xi , xi+1]. Remarquez notre modèle jusqu'à présent : les polynômes constants par morceaux sont discontinus, tandis que les fonctions linéaires par morceaux sont continues. Il est facile de voir que les quadratiques par morceaux peuvent être C et ainsi de suite. Cette continuité accrue et cette différenciation accrues se produisent même si chaque yi bénéficie d'un support local ; cette théorie est élaborée en détail dans la construction de «splines», ou courbes interpolant entre des points donnés des valeurs de fonction et des tangentes.

Cette continuité accrue a cependant ses propres inconvénients. Avec chaque degré supplémentaire de dérivabilité, nous posons une hypothèse de lissage plus forte sur f . Cette hypothèse peut être irréaliste : de nombreux phénomènes physiques sont vraiment bruyants ou discontinus, et cette régularité accrue peut affecter négativement les résultats d'interpolation. Un domaine dans lequel cet effet est particulièrement clair est lorsque l'interpolation est utilisée en conjonction avec des outils de simulation physique. La simulation d'écoulements de fluides turbulents avec des fonctions surlissées peut éliminer les phénomènes discontinus comme les ondes de choc qui sont souhaitables en sortie.

Ces problèmes mis à part, les polynômes par morceaux peuvent toujours être écrits comme des combinaisons linéaires de fonctions de base. Par exemple, les fonctions suivantes servent de base aux fonctions constantes par morceaux :

$$\varphi_i(x) = \begin{cases} 1 & \text{quand } x_{i-1}+x_i \le x < \frac{x_i+x_{i+1}}{2} \\ 0 & \text{sinon} \end{cases}$$

Cette base place simplement la constante 1 près de xi et 0 ailleurs ; l'interpolation constante par morceaux d'un ensemble de points (xi , yi) s'écrit f(x) = ∑i yiφi(x). De même, la base dite "chapeau" illustrée dans la figure NUMBER couvre l'ensemble des fonctions linéaires par morceaux avec des arêtes vives à nos points de données xI :

$$\psi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{quand } x_{i-1} < x \le x_i \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & \text{quand } x_i < x \le x_{i+1} \\ 0 & \text{sinon} \end{cases}$$

Encore une fois, par construction, l'interpolation linéaire par morceaux des points de données donnés est f(x) = ∑i yiψi(x).

11.1.4 Processus gaussiens et krigeage

Non couvert dans CS 205A, automne 2013.

11.2 Interpolation multivariable

De nombreuses extensions des stratégies ci-dessus existent pour interpoler une fonction à partir de points de données (xi , yi) où xi Rn peut maintenant être multidimensionnel. Cependant, les stratégies d'interpolation dans ce cas ne sont pas aussi claires, car il est moins évident de partitionner Rn en un petit nombre de régions autour de xi . Pour cette raison, un modèle courant consiste à interpoler en utilisant des fonctions d'ordre relativement bas, c'est-à-dire à préférer des stratégies d'interpolation simplistes et efficaces à celles qui produisent des fonctions C ∞ .

Si tout ce qui nous est donné est l'ensemble d'entrées et de sorties (xi , yi), alors une stratégie constante par morceaux pour l'interpolation consiste à utiliser l'interpolation au plus proche voisin. Dans ce cas f(x) prend simplement la valeur yi correspondant à xi minimisant x − xi2 ; les implémentations simples parcourent tout i pour trouver cette valeur, bien que les structures de données comme les arbres kd puissent trouver plus rapidement les voisins les plus proches. Tout comme les interpolations constantes par morceaux divisent R en intervalles autour des points de données xi , la stratégie du plus proche voisin divise Rn en un ensemble de cellules de Voronoi :

Définition 11.2 (Cellule de Voronoï). Étant donné un ensemble de points S = {x1,x2, . . . ,xk} Rn , la cellule de Voronoï correspondant à un xi spécifique est l'ensemble Vi ≡ {x : x −xi2 < x −xj2 pour tout j = i}. Autrement dit, c'est l'ensemble des points les plus proches de xi que de tout autre xj dans S.

La figure NUMBER montre un exemple de cellules de Voronoi concernant un ensemble de points de données dans . Ces des cellules R2 ayant de nombreuses propriétés favorables ; par exemple, ce sont des polygones convexes et sont localisés autour de chaque xEn fait, la connectivité des cellules de Voronoi est un problème bien étudié en géométrie computationnelle menant à la construction de la célèbre triangulation de Delaunay.

Il existe de nombreuses options pour l'interpolation continue des fonctions sur Rn chacune avec ses propres avantages et inconvénients. Si nous souhaitons étendre notre stratégie du plus proche voisin ci-dessus, par exemple, nous pourrions calculer plusieurs voisins les plus proches de x et interpoler f(x) en fonction de x −

xi2 pour chaque plus proche voisin xi . Certaines structures de données « k plus proches voisins » peuvent accélérer les requêtes lorsque vous souhaitez rechercher plusieurs points dans un ensemble de données les plus proches d'un x donné.

Une autre stratégie apparaissant fréquemment dans la littérature sur l'infographie est l'interpolation barycentrique. Supposons que nous ayons exactement n + 1 points d'échantillonnage (x1, y1), . . . ,(xn+1, yn+1), où xi Rn et comme , toujours on souhaite interpoler les valeurs de y à tout Rn ; par exemple, sur le plan, on nous donnerait trois valeurs associées aux sommets d'un triangle. Tout point x Rn peut être écrit de manière unique comme une combinaison linéaire $x = \sum_{i=1}^{n+1} a_i x_i$ avec une contrainte supplémentaire que $\sum_i a_i = 1$; en d'autres termes, nous écrivons x comme une moyenne pondérée des points xi . L'interpolation barycentrique dans ce cas s'écrit simplement $f(x) = \sum_i a_i(x) y_i$.

Sur le plan R2 , l'interpolation barycentrique a une interpolation géométrique simple dans les zones de triangle tournant, illustrée sur la figure NUMBER. De plus, il est facile de vérifier que la fonction interpolée résultante f(x) est affine, c'est-à-dire qu'elle peut s'écrire f(x) = c + d · x pour certains c R et d Rn

.

En général, le système d'équations que nous souhaitons résoudre pour l'interpolation barycentrique à certains x Rn est :

$$\sum_i a_i x_i = x$$

$$\sum_i a_i = 1$$

En l'absence de dégénérescences, ce système fora est inversible lorsqu'il y a n + 1 points xi . En présence de plus de xi , cependant, le système pour a devient sous-déterminé. Cela signifie qu'il existe plusieurs façons d'écrire un x donné comme une moyenne pondérée des xi .

Une solution à ce problème est d'ajouter plus de conditions sur le vecteur des pondérations moyennesa. Cette stratégie aboutit à des coordonnées barycentriques généralisées, un sujet de recherche en mathématiques et en ingénierie modernes. Les contraintes typiques de ona demandent qu'il soit lisse en fonction de Rn et non négatif à l'intérieur de l'ensemble des xi lorsque ces points définissent un polygone ou un polyèdre. La figure NUMBER montre un exemple de coordonnées barycentriques généralisées calculées à partir de points de données sur un polygone avec plus de n + 1 points.

Une résolution alternative du problème sous-déterminé pour les coordonnées barycentriques concerne l'idée d'utiliser des fonctions par morceaux pour l'interpolation; nous limiterons notre discussion ici à xi R2 pour simplifier, bien que les extensions à des dimensions supérieures soient relativement évidentes.
Plusieurs fois, on nous donne non seulement l'ensemble de points xi mais aussi une décomposition du domaine qui nous intéresse (dans ce cas, un sous-ensemble de R2) en objets à n + 1 dimensions utilisant ces points comme sommets. Par exemple, la figure NUMBER montre une telle mosaïque d'une partie de R2 en triangles.
L'interpolation dans ce cas est simple : l'intérieur de chaque triangle est interpolé à l'aide de coordonnées barycentriques.

Exemple 11.2 (ombrage). En infographie, l'une des représentations les plus courantes d'une forme est un ensemble de triangles dans un maillage. Dans le modèle d'ombrage par sommet, une couleur est calculée pour chaque sommet d'un maillage. Ensuite, pour rendre l'image à l'écran, ces valeurs par sommet sont interpolées en utilisant une interpolation barycentrique à l'intérieur des triangles. Des stratégies similaires sont utilisées pour la texturation et d'autres tâches courantes. La figure NUMBER montre un exemple de ce modèle d'ombrage simple. En aparté, un problème spécifique à l'infographie est l'interaction entre les transformations de perspective et les stratégies d'interpolation.
L'interpolation barycentrique de la couleur sur une surface 3D, puis la projection de cette couleur sur le plan image n'est pas la même chose que la projection de triangles sur le plan image et ensuite l'interpolation des couleurs à l'intérieur du triangle; ainsi, les algorithmes de ce domaine doivent appliquer une correction de perspective pour tenir compte de cette erreur.

Étant donné un ensemble de points dans R2 , le problème de la triangulation est loin d'être trivial, et les algorithmes pour effectuer ce type de calcul s'étendent souvent mal à Rn . Ainsi, dans les dimensions supérieures, les stratégies du plus proche voisin ou de régression deviennent préférables (voir chapitre NUMÉRO).

L'interpolation barycentrique conduit à une généralisation des fonctions chapeau linéaires par morceaux du §11.1.3 illustrées à la figure NUMBER. Rappelons que notre sortie interpolatoire est entièrement déterminée par les valeurs yi aux sommets des triangles. En fait, nous pouvons considérer f(x) comme une combinaison linéaire ∑i yiφi(x), où chaque φi(x) est la fonction barycentrique par morceaux obtenue en mettant un 1 sur xi et 0 partout ailleurs, comme sur la figure NUMÉRO . Ces fonctions chapeau triangulaires forment la base de la «méthode des éléments finis du premier ordre», que nous explorerons dans les prochains chapitres; des constructions spécialisées utilisant des polynômes d'ordre supérieur sont appelées «éléments d'ordre supérieur» peuvent être utilisées pour garantir la différentiabilité le long des bords du triangle.

Une décomposition alternative et tout aussi importante du domaine de f se produit lorsque les points xi apparaissent sur une grille régulière dans Rn . Les exemples suivants illustrent des situations où c'est le cas :

Exemple 11.3 (Traitement d'images). Comme mentionné dans l'introduction, une photographie numérique typique est représentée par une grille m × n d'intensités de couleur rouge, verte et bleue. Nous pouvons penser que ces valeurs vivent sur un réseau en Z × Z. Supposons que nous souhaitions faire pivoter l'image d'un angle qui n'est pas un multiple de 90◦ , cependant. Ensuite, comme illustré dans la figure NUMBER, nous devons rechercher des valeurs d'image à des positions potentiellement non entières, nécessitant l'interpolation des couleurs à R × R.

Exemple 11.4 (Imagerie médicale). La sortie typique d'un appareil d'imagerie par résonance magnétique (IRM) est une grille de valeurs am × n × p représentant la densité de tissu à différents points; théoriquement, le modèle typique de cette fonction est f : R3 → R. Nous pouvons extraire la surface externe d'un organe particulier, illustré à la figure NUMBER, en trouvant l'ensemble de niveaux {x : f(x) = c} pour un certain c. Trouver cet ensemble de niveaux nous oblige à étendre f à l'ensemble de la grille de voxels pour trouver exactement où il croise c.

Les stratégies d'interpolation basées sur la grille appliquent généralement les formules unidimensionnelles du §11.1.3 une dimension à la fois. Par exemple, les schémas d'interpolation bilinéaire dans R2 interpolent linéairement une dimension à la fois pour obtenir la valeur de sortie :

Exemple 11.5 (Interpolation bilinéaire). Supposons que f prenne les valeurs suivantes :

- f(0, 0) = 1
- f(0, 1) = −3
- f(1, 0) = 5
- f(1, 1) = −11

et qu'entre f est obtenu par interpolation bilinéaire. Pour trouver f($\frac{1}{4}, \frac{1}{2}$), on interpole d'abord en x1 pour trouver :

$$F\left(\frac{1}{4}, 0\right) = \frac{3}{4} f(0,0) + \frac{1}{4} f(1,0) = 2$$

$$F\left(\frac{1}{4}, 1\right) = \frac{3}{4} F(0,1) + \frac{1}{4} F(1,1) = -5$$

Ensuite, on interpole en x2 :

$$F\left(\frac{1}{4}, \frac{1}{2}\right) = \frac{1}{2} f\left(\frac{1}{4}, f(0) + 2^{-1}\right)\left(\frac{1}{4}, 3(1) = -2\right)$$

Une propriété importante de l'interpolation bilinéaire est que nous recevons la même sortie en interpolant d'abord en x2 et ensuite en x1.

Les méthodes d'ordre supérieur comme l'interpolation bicubique et Lanczos utilisent à nouveau plus de termes polynomiaux mais sont plus lentes à calculer. En particulier, dans le cas d'images interpolées, les stratégies bicubiques nécessitent plus de points de données que le carré des valeurs de fonction les plus proches d'un point x ; cette dépense supplémentaire peut ralentir les outils graphiques pour lesquels chaque recherche en mémoire entraîne un temps de calcul supplémentaire.

11.3 Théorie de l'interpolation

Jusqu'à présent, notre traitement de l'interpolation a été assez heuristique. Bien que nous nous appuyions sur notre intuition pour savoir ce qu'une interpolation «raisonnable» pour un ensemble de valeurs de fonction est pour la plupart une stratégie acceptable, des problèmes subtils peuvent survenir avec différentes méthodes d'interpolation qu'il est important de reconnaître.

11.3.1 Algèbre linéaire des fonctions

Nous avons commencé notre discussion en posant diverses stratégies d'interpolation comme bases différentes pour l'ensemble des fonctions f : R → R. Cette analogie avec les espaces vectoriels s'étend à une théorie géométrique complète des fonctions, et en fait les premiers travaux dans le domaine de l'analyse fonctionnelle essentiellement étend la géométrie de Rn à des ensembles de fonctions. Ici, nous discuterons des fonctions d'une variable, bien que de nombreux aspects de l'extension à des fonctions plus générales soient faciles à réaliser.

Tout comme nous pouvons définir les notions d'étendue et de combinaison linéaire pour les fonctions, pour a, b R fixes, nous pouvons définir un produit interne des fonctions f(x) et g(x) comme suit :

$$\langle f, g \rangle \equiv \int_{un}^{b} f(x)g(x)\,dx.$$

Tout comme le produit interne A des vecteurs nous a aidés à dériver l'algorithme des gradients conjugués et avait beaucoup en commun avec le produit scalaire, le produit interne fonctionnel peut être utilisé pour définir des méthodes d'algèbre linéaire pour traiter les espaces de fonctions et comprendre leur étendue. Nous définissons également une norme d'une fonction comme étant $\|f\| \equiv \sqrt{\langle f, f \rangle}$.

Exemple 11.6. Fonction produit scalaire Soit $p_n(x) = x^b = 1^n$ être le n-ième monôme. Alors, pour a = 0 et on a :

$$\begin{aligned} \langle p_n, p_m \rangle &= \int_0^1 n x \cdot m \cdot x\,dx \\ &= \int_0^1 x^{n+m}\,dx \\ &= \frac{1}{n+m+1} \end{aligned}$$

Notez que cela montre:

$$\frac{pn}{pn}, \frac{pm}{pm} = \frac{pn,}{pm} = \frac{}{\frac{pnpm\,(2n+1)}{(2m+1)\,n+m+1}}$$

Cette valeur est d'environ 1 lorsque n ≈ m mais n = m, confirmant notre affirmation précédente selon laquelle les monômes "se chevauchent" considérablement sur [0, 1].

Étant donné ce produit interne, nous pouvons appliquer l'algorithme de Gram-Schmidt pour trouver une base orthonormale pour l'ensemble des polynômes. Si on prend a = −1 et b = 1, on obtient les polynômes de Legendre, tracés sur la figure NUMBER :

$$P0(x) = 1$$
$$P1(x) = x$$
$$P2(x) = \frac{1}{2}(3x^2 - 1)$$
$$P3(x) = (5\frac{1}{x}2\,1^3 - 3x)$$
$$P4(x) = \frac{}{8}(35x^4 - 30x^2 + 3)$$
$$\vdots \quad \vdots$$

Ces polynômes ont de nombreuses propriétés utiles grâce à leur orthogonalité. Par exemple, supposons que l'on veuille approximer f(x) avec une somme ∑i aiPi(x). Si l'on veut minimiser f − ∑i aiPi dans la norme fonctionnelle, c'est un problème des moindres carrés ! Par orthogonalité de la base de Legendre pour R[x], une simple extension de nos méthodes de projection montre :

$$ai = \frac{f\,,\,Pi}{Pi\,,\,Pi}$$

Ainsi, l'approximation de f à l'aide de polynômes peut être accomplie simplement en intégrant f par rapport aux membres de la base de Legendre ; dans le chapitre suivant nous apprendrons comment cette intégrale pourrait être réalisée approximativement.

Étant donné une fonction positive w(x), On peut définir un produit scalaire plus général ·, ·w en écrivant

$$f\,,\,gw = \int_{un}^{b} w(x)f(x)g(x)\,dx.$$

Si on prend $w(x) = \sqrt{\frac{1}{1-x^2}}$ avec a = −1 et b = 1, alors l'application de Gram-Schmidt donne le Chebyshev polynômes :

$$T0(x) = 1$$
$$T1(x) = x$$
$$T2(x) = 2x^2 - 1$$
$$T3(x) = 4x^3 - 3x$$
$$T4(x) = 8x^4 - 8x^2 + 1$$
$$\vdots \quad \vdots$$

En fait, une identité surprenante tient pour ces polynômes :

$$T_k(x) = \cos(k \arccos(x)).$$

Cette formule peut être vérifiée en la vérifiant explicitement pour T0 et T1, puis en appliquant inductivement l'observation :

$$\begin{aligned} T_{k+1}(x) &= \cos((k+1)\arccos(x)) \\ &= 2x\cos(k\arccos(x)) - \cos((k-1)\arccos(x)) \text{ par l'identité} \\ &\qquad \cos((k+1)\theta) = 2\cos(k\theta)\cos(\theta) - \cos((k-1)\theta) \\ &= 2xT_k(x) - T_{k-1}(x) \end{aligned}$$

Cette formule de « récurrence à trois termes » permet également de générer facilement les polynômes de Chebyshev.

Comme l'illustre la figure NUMBER, grâce à la formule trigonométrique des polynômes de Chebyshev, il est facile de voir que les minima et les maxima de Tk oscillent entre +1 et −1.
De plus, ces extrema sont situés en $\cos(i\pi/k)$ (les dits « points de Chebyshev ») pour i de 0 à k ; cette belle distribution d'extrema évite les phénomènes oscillatoires comme celui illustré à la figure NUMBER lors de l'utilisation d'un nombre fini de termes polynomiaux pour approximer une fonction. En fait, des traitements plus techniques de l'interpolation polynomiale recommandent de placer les xi pour l'interpolation près des points de Chebyshev afin d'obtenir une sortie lisse.

11.3.2 Approximation via des polynômes par morceaux

Supposons que l'on veuille approximer une fonction f(x) avec un polynôme de degré n sur un intervalle [a, b]. Définissons Δx comme étant l'espacement b − a. Une mesure de l'erreur d'une approximation est en fonction de Δx, qui devrait disparaître lorsque $\Delta x \to 0$. Ensuite, si nous approximons f avec des polynômes par morceaux, ce type d'analyse nous indique à quelle distance nous devons espacer les polynômes pour obtenir un niveau d'approximation souhaité.

Par exemple, supposons que nous approchions f avec une constante c = f(interpolation. $\frac{a+b}{2}$), comme dans la constante par morceaux Si nous supposons | f (x)| < M pour tout x [a, b], nous avons :

$$\max_{x \in [a,b]} |f(x) - c| \le \Delta x \max_{x \in [a,b]} M \text{ par le théorème de la valeur moyenne}$$

$$\le M\Delta x$$

Ainsi, nous nous attendons à une erreur $O(\Delta x)$ lors de l'utilisation d'une interpolation constante par morceaux.

Supposons plutôt que nous approchions f en utilisant une interpolation linéaire par morceaux, c'est-à-dire en prenant

$$\tilde{f}(x) = f(a) + \frac{b - X}{b - une} \qquad \frac{x - une}{- une} f(b). \; \tilde{b}$$

Par le théorème de la valeur moyenne, nous savons f (x) = f (θ) pour un certain θ [a, b]. L'écriture du développement de Taylor autour de θ montre $f(x) = f(\theta) + f(\theta)(x - \theta) + O(\Delta x^2)$ sur [a, b], alors que nous pouvons réécrire notre linéaire l'approximation lorsque f(x) = f(θ) + f (θ)(x − θ). Ainsi, la soustraction de ces deux expressions montre que). Il n'est pas difficile de l'erreur d'approximation de f diminue à $O(\Delta x^2)$ avec un polynôme prédire cette erreur d' approximation , bien qu'en pratique la de degré n fait que $O(\Delta x^{n+1})$ des approximations linéaires convergence quadratique par morceaux suffit pour la plupart des applications.

11.4 Problèmes

Idées :

- Méthode de Horner pour évaluer les polynômes
- Stratégie récursive pour les coefficients polynomiaux de Newton.
- Splines, de Casteljeau
- Vérifier l'interpolation de la zone triangulaire de l'interpolation barycentrique

Chapitre 12

Intégration numérique et Différenciation

Dans le chapitre précédent, nous avons développé des outils pour remplir des valeurs raisonnables d'une fonction f(x) étant donné un échantillon de valeurs (xi , f(xi)) dans le domaine de f . Bien entendu ce problème d'interpolation est utile en lui-même pour compléter des fonctions dont on sait qu'elles sont continues ou dérivables mais dont les valeurs ne sont connues qu'en un ensemble de points isolés, mais dans certains cas on souhaite alors étudier les propriétés de ces fonctions. En particulier, si nous souhaitons appliquer des outils de calcul à f , être capable d'approximer nous devons ses intégrales et ses dérivées.

En fait, il existe de nombreuses applications dans lesquelles l'intégration et la différenciation numériques jouent un rôle clé dans le calcul. Dans le cas le plus simple, certaines fonctions bien connues sont définies comme des intégrales. Par exemple, la "fonction d'erreur" utilisée comme distribution cumulée d'une courbe gaussienne ou en cloche s'écrit :

$$\operatorname{erf}(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

Des approximations de erf(x) sont nécessaires dans de nombreux contextes statistiques, et une approche raisonnable pour trouver ces valeurs consiste à effectuer numériquement l'intégrale ci-dessus.

D'autres fois, les approximations numériques des dérivées et des intégrales font partie d'un système plus vaste. Par exemple, les méthodes que nous développerons dans les prochains chapitres pour approximer les solutions aux équations différentielles dépendront fortement de ces approximations. De même, en électrodynamique computationnelle, les équations intégrales résolvant une fonction inconnue φ étant donné un noyau K et une sortie f apparaissent dans la relation :

$$f(x) = \int_{Rn} K(x,y)\varphi(y)\, dy.$$

Ces types d'équations doivent être résolues pour estimer les champs électriques et magnétiques, mais à moins que φ et K ne soient très spéciaux, nous ne pouvons espérer trouver une telle intégrale sous forme fermée, mais résoudre seule cette équation pour la fonction inconnue φ.

Dans ce chapitre, nous développerons diverses méthodes d'intégration et de différenciation numériques à partir d'un échantillon de valeurs de fonctions. Ces algorithmes sont généralement des approximations assez simples, donc pour les comparer, nous développerons également des stratégies qui évaluent la performance attendue de différentes méthodes.

12.1 Motivations

Il n'est pas difficile de formuler des applications simples d'intégration et de différenciation numériques compte tenu de la fréquence à laquelle les outils de calcul apparaissent dans les formules et techniques de base de la physique, des statistiques et d'autres domaines. Nous suggérons ici quelques endroits moins évidents où l'intégration et la différenciation apparaissent.

Exemple 12.1 (Échantillonnage à partir d'une distribution). Supposons qu'on nous donne une distribution de probabilité p(t) sur l'intervalle [0, 1] ; c'est-à-dire que si nous échantillonnons au hasard des valeurs en fonction de cette distribution, nous nous attendons à ce que p(t) soit proportionnel au nombre de fois que nous tirons une valeur proche de t. Une tâche courante consiste à générer des nombres aléatoires distribués comme p(t).

Plutôt que de développer une méthode spécialisée pour le faire à chaque fois que nous recevons un nouveau p(t), il est possible de faire une observation utile. Nous définissons la fonction de distribution cumulative de p comme étant

$$F(t) = \int_0^t p(x)\,dx.$$

Alors, si X est un nombre aléatoire uniformément distribué dans [0, 1], on peut montrer que $F^{-1}(X)$ est distribué comme p, où F^{-1} est l'inverse de F. Ainsi, si on peut approximer F ou F^{-1} on peut générer des nombres aléatoires selon une distribution arbitraire p ; cette approximation revient à intégrer p, ce qui peut devoir être fait numériquement lorsque les intégrales ne sont pas connues sous forme fermée.

Exemple 12.2 (Optimisation). Rappelons que la plupart de nos méthodes pour minimiser et trouver les racines d'une fonction f dépendaient non seulement des valeurs f(x) mais aussi de son gradient $\nabla f(x)$ et même de Hessian H_f . Nous avons vu que des algorithmes comme BFGS et la méthode de Broyden construisent des approximations grossières des dérivées de f au cours du processus d'optimisation. Lorsque f a des fréquences élevées, cependant, il peut être préférable d'approximer ∇f près de l'itération courante xk plutôt que d'utiliser des valeurs de points potentiellement éloignés x pour < k.

Exemple 12.3 (Rendu). L'équation de rendu du lancer de rayons et d'autres algorithmes pour un rendu de haute qualité est une intégrale indiquant que la lumière quittant une surface est égale à l'intégrale de la lumière entrant dans la surface dans toutes les directions entrantes possibles après avoir été réfléchie et diffusée; il stipule essentiellement que l'énergie lumineuse doit être conservée avant et après l'interaction de la lumière avec un objet. Les algorithmes de rendu doivent se rapprocher de cette intégrale pour calculer la quantité de lumière émise par une surface réfléchissant la lumière dans une scène.

Exemple 12.4 (Traitement d'image). Supposons que nous pensions à une image en fonction de deux variables I(x, y). De nombreux filtres, y compris les flous gaussiens, peuvent être considérés comme des convolutions, données par

$$(I * g)(x, y) = \int I(u, v)g(x - u, y - v)\,du\,dv.$$

Par exemple, pour brouiller une image, nous pourrions prendre g comme une gaussienne ; dans ce cas $(I * g)(x, y)$ peut être considéré comme une moyenne pondérée des couleurs de I près du point (x, y). En pratique, les images sont des grilles discrètes de pixels, donc cette intégrale doit être approchée.

Exemple 12.5 (Règle de Bayes). Supposons que X et Y soient des variables aléatoires à valeur continue ; nous pouvons utiliser P(X) et P(Y) pour exprimer les probabilités que X et Y prennent des valeurs particulières. Parfois, connaître X peut affecter notre connaissance de Y. Par exemple, si X est la tension artérielle d'un patient et Y est le poids d'un patient,

alors savoir qu'un patient a un poids élevé peut suggérer qu'il souffre également d'hypertension artérielle. On peut donc aussi écrire des distributions de probabilités conditionnelles P(X|Y) (lire « la probabilité de X étant donné Y ») exprimant de telles relations.

Une fondation de la théorie moderne des probabilités stipule que P(X|Y) et P(Y|X) sont liés comme suit : P(Y|

$$P(X|Y) = \frac{X)P(X)}{P(Y|X)P(X)\,dY}$$

L'estimation de l'intégrale dans le dénominateur peut être un problème sérieux dans les algorithmes d'apprentissage automatique où les distributions de probabilité prennent des formes complexes. Ainsi, des schémas d'intégration approximatifs et souvent randomisés sont nécessaires pour les algorithmes de sélection de paramètres qui utilisent cette valeur dans le cadre d'une technique d'optimisation plus large.

12.2 Quadrature

Nous allons commencer par considérer le problème de l'intégration numérique, ou quadrature. Ce problème - dans une seule variable - peut être exprimé comme suit : "Étant donné un échantillonnage de n points à partir d'une fonction f(x), trouver une approximation de $\int_{un}^{b} f(x)\,dx$". Dans la section précédente, nous avons présenté plusieurs situations qui se résumer exactement à cette technique.

Il existe quelques variantes du problème qui nécessitent un traitement ou une adaptation légèrement différente. tion :

- Les extrémités a et b peuvent être fixes, ou nous pouvons souhaiter trouver un schéma de quadrature qui peut efficacement approcher les intégrales pour de nombreuses paires (a, b) .
- Nous pouvons être en mesure d'interroger f(x) à n'importe quel x mais souhaiter approximer l'intégrale en utilisant relativement peu d'échantillons, ou nous pouvons recevoir une liste de paires précalculées (xi , f(xi)) et sommes contraints d'utiliser ces données points dans notre approximation.

Ces considérations doivent être gardées à l'esprit lorsque nous concevons des algorithmes assortis pour le problème de quadrature.

12.2.1 Quadrature interpolatoire

Plusieurs des stratégies d'interpolation développées dans le chapitre précédent peuvent être étendues aux méthodes de quadrature en utilisant une observation très simple. Supposons que nous écrivions une fonction f(x) en termes d'un ensemble de fonctions de base φi(x) :

$$f(x) = \sum_{je} a_i \varphi_i(x).$$

On peut alors trouver l'intégrale de f comme suit :

$$\int_{un}^{b} f(x)\,dx = \int_{un}^{b} \sum_{je} a_i \varphi_i(x)\,dx \quad \text{par définition de } f$$

$$= \sum_{je} a_i \int_{un}^{b} \varphi_i(x)\,dx$$

$$= \sum_{je} c_i a_i \quad \text{si on fait la définition } c_i \equiv \int_{un}^{b} \varphi_i(x)\,dx$$

En d'autres termes, l'intégration de f consiste simplement à combiner linéairement les intégrales des fonctions de base qui composent f .

Exemple 12.6 (Monômes). Supposons que nous écrivions $f(x) = \sum_k a_k x^k$. Nous savons

$$\int_0^1 x^k \, dx = \frac{1}{x^{k+1}},$$

donc en appliquant la dérivation ci-dessus, nous savons

$$\int_0^1 f(x) \, dx = \sum_k \frac{a_k}{k+1}.$$

En d'autres termes, dans notre notation ci-dessus, nous avons défini $c_k = \frac{1}{k+1}$.

Les schémas où nous intégrons une fonction en interpolant des échantillons et en intégrant la fonction interpolée sont appelés règles de quadrature interpolatoires ; presque toutes les méthodes que nous allons présenter ci-dessous peuvent s'écrire de cette façon. Bien sûr, nous pouvons être confrontés à un problème de poule et d'œuf, si l'intégrale $\varphi_i(x) \, dx$ elle-même n'est pas connue sous forme fermée. Certaines méthodes dans les éléments finis d'ordre supérieur traitent ce problème en mettant du temps de calcul supplémentaire pour faire une approximation numérique de haute qualité de l'intégrale d'un seul φ_i , puis puisque tous les φ ont une forme similaire, appliquez des formules de changement de coordonnées à écrire des intégrales pour les fonctions de base restantes. Cette intégrale canonique peut être approximée hors ligne à l'aide d'un schéma de haute précision, puis réutilisée.

12.2.2 Règles de quadrature

Si on nous donne un ensemble de paires $(x_i , f(x_i))$, notre discussion ci-dessus suggère la forme suivante pour une règle de quadrature pour approximer l'intégrale de f sur un intervalle :

$$Q[f] \equiv \sum_i w_i f(x_i).$$

Différents poids w_i donneront différentes approximations de l'intégrale, qui, nous l'espérons, deviendront de plus en plus similaires à mesure que nous échantillonnerons les x_i de manière plus dense.

En fait, même la théorie classique de l'intégration suggère que cette formule est un point de départ raisonnable. Par exemple, l'intégrale de Riemann présentée dans de nombreux cours d'introduction au calcul prend la forme :

$$\int_a^b f(x) = \lim_{\Delta x_k \to 0} \sum_k f(\tilde{x}_k)(x_{k+1} - x_k)$$

Ici, l'intervalle [a, b] est partitionné en morceaux $a = x_1 < x_2 < \cdots < x_n = b$, où $\Delta x_k = x_{k+1} - x_k$ et $\tilde{x}_k$ est tout point de $[x_k , x_{k+1}]$. Pour un ensemble fixe de x_k avant de prendre la limite, cette intégrale peut clairement s'écrire sous la forme $Q[f]$ ci-dessus.

De ce point de vue, les choix de $\{x_i\}$ et $\{w_i\}$ déterminent complètement une stratégie de quadrature. Il existe de nombreuses façons de déterminer ces valeurs, comme nous le verrons dans la section suivante et comme nous l'avons déjà vu pour la quadrature interpolatoire.

Exemple 12.7 (Méthode des coefficients indéterminés). Supposons que nous fixons $x_1, \ldots, x_n$ et souhaitent trouver un ensemble raisonnable de poids d'accompagnement w_i de sorte que $\sum_i w_i f(x_i)$ soit une approximation appropriée de l'intégrale

désactivé . Une alternative à la stratégie de fonction de base répertoriée ci-dessus consiste à utiliser la méthode des coefficients indéterminés. Dans cette stratégie, nous choisissons n fonctions f1(x), . . . , fn(x) dont les intégrales sont connues, et demander que notre règle de quadrature récupère exactement les intégrales de ces fonctions :

$$\begin{aligned}
\int_a^b f1(x)\,dx &= w1\,f1(x1) + w2\,f1(x2) + \cdots + wn\,f1(xn)\\
\int_a^b f2(x)\,dx &= w1\,f2(x1) + w2\,f2(x2) + \cdots + wn\,f2(xn)\\
&\vdots \qquad\qquad \vdots\\
\int_a^b fn(x)\,dx &= w1\,fn(x1) + w2\,fn(x2) + \cdots + wn\,fn(xn)
\end{aligned}$$

Cela crée un système linéaire n × n d'équations pour les wi .

Un choix courant est de prendre fk(x) = x les $^{k-1}$, c'est-à-dire pour s'assurer que le schéma de quadrature récupère intégrales des polynômes d'ordre inférieur. Nous savons

$$\int_a^b kx\_\ k+1\,dx = \frac{b \quad - une^{k+1}}{k+1}.$$

Ainsi, nous obtenons le système linéaire d'équations suivant pour les wi :

$$\begin{aligned}
w1 + w2 + \cdots + wn &= b - une\\
b\ x1w1 + x2w2 + \cdots + xnwn &= \frac{2\,2 - une}{2}\\
^{2x}1w1 + x\,^{2}2w2 + \cdots + x\quad ^{2}nwn &= \frac{b\ 3\ 3 - un}{2}\\
&\vdots \qquad\qquad \vdots\\
1x^{n-}\ ^{1}\ w1 + x\,2\,w2 + \cdots + x\quad ^{n-1}\ n\quad wn &= \frac{2\,2\,b - une}{2}
\end{aligned}$$

Ce système est exactement le système de Vandermonde discuté au §11.1.1.

12.2.3 Quadrature de Newton-Cotes

Règles de quadrature lorsque les $_{je}$s sont régulièrement espacés dans [a, b] sont appelés quadrature de Newton-Cotes règles x. Comme l'illustre la figure NUMÉRO, il existe deux choix raisonnables d'échantillons régulièrement espacés :

- La quadrature fermée de Newton-Cotes place les xi en a et b. En particulier, pour k {1, . . . , n} nous prendre

$$xk \equiv une + \frac{(k-1)(b-une)}{n-1}.$$

• La quadrature ouverte de Newton-Cotes ne place pas de xi en a ou en b :

$$xk \equiv une + \frac{k(b - une)}{n + 1}.$$

Après avoir fait ce choix, les formules de Newton-Cotes appliquent simplement une interpolation polynomiale pour approximer l'intégrale de a à b ; le degré du polynôme doit évidemment être n - 1 pour garder la règle de quadrature bien définie.

En général, nous garderons n relativement petit. De cette façon, nous évitons les phénomènes d'oscillation et de bruit qui se produisent lors de l'ajustement de polynômes de degré élevé à un ensemble de points de données. Comme dans l'interpolation polynomiale par morceaux, nous enchaînerons ensuite de petits morceaux en règles composites lors de l'intégration sur un grand intervalle [a, b].

Règles fermées. Les stratégies de quadrature de Newton-Cotes fermées nécessitent n ≥ 2 pour éviter de diviser par zéro. Deux stratégies apparaissent souvent en pratique :

• La règle trapézoïdale est obtenue pour n = 2 (donc x1 = a et x2 = b) en interpolant linéairement de f(a) à f(b). Il stipule que

$$\int_{un}^{b} dx \approx (b - a)\, 2 \qquad \frac{f(a) + f(b)\ f(x)}{}.$$

• La règle de Simpson vient du fait que n = 3, nous avons donc maintenant

$$x1 = un$$

$$b\ x2 = \frac{une +}{2}$$

$$x3 = b$$

L'intégration de la parabole qui passe par ces trois points donne

$$\int_{un}^{b} une\ f(x)\ dx \approx \frac{b -}{6} \quad f(a) + 4\ f \qquad \frac{un + b}{2} \quad + f(b)\ .$$

Règles ouvertes. Les règles ouvertes pour la quadrature permettent la possibilité de n = 1, donnant la règle simpliste du point médian :

$$\int_{un}^{b} f(x)\ dx \approx (b - a) f \qquad \frac{un + b}{2} \quad .$$

Des valeurs plus grandes de n produisent des règles similaires à la règle de Simpson et à la règle trapézoïdale.

Intégration composite. En général, nous pourrions souhaiter intégrer f(x) avec plus d'une, deux ou trois valeurs xi . Il est évident comment construire une règle composite à partir des règles médianes ou trapézoïdales ci-dessus, comme illustré dans la figure NUMBER ; additionnez simplement les valeurs le long de chaque intervalle. Par exemple, si nous subdivisons [a, b] en k intervalles, alors nous pouvons prendre $\Delta x \equiv \frac{b-a}{k}$ et $xi \equiv a + i\Delta x$. Alors, la règle du point médian composite est :

$$\int_{un}^{b} f(x)\ dx \approx f \quad \sum_{je=1}^{k} \quad \frac{xi+1 + xi\ 2}{} \quad \Delta x$$

De même, la règle du trapèze composite est :

$$\int_{un}^{b} f(x)\,dx \approx \sum_{je=1}^{k} \frac{f(xi) + f(xi+1)}{2}\,\Delta x$$

$$= \Delta x \left[\frac{1}{2} f(a) + f(x1) + f(x2) + \cdots + f(xk-1) + \frac{1}{2} f(b) \right]$$

en séparant les deux valeurs moyennes de f dans la première ligne et en réindexant

Un traitement alternatif de la règle du point médian composite consiste à appliquer la formule de quadrature interpolatoire du §12.2.1 à l'interpolation linéaire par morceaux ; de même, la version composite de la règle trapézoïdale provient d'une interpolation linéaire par morceaux.

La version composite de la règle de Simpson, illustrée dans la figure NUMBER, enchaîne trois points à la fois pour faire des approximations paraboliques. Les paraboles adjacentes se rencontrent à des xi indexés pairs et peuvent ne pas partager de tangentes. Cette sommation, qui n'existe que lorsque n est pair, devient :

$$\int_{un}^{b} f(x)\,dx \approx \frac{\Delta x}{3} \left[f(a) + 2 \sum_{je=1}^{n-2-1} f(x2i) + 4 \sum_{je=1}^{n/2} f(x2i-1) + f(b) \right]$$

$$= \frac{\Delta x}{3} [\, f(a) + 4\, f(x1) + 2\, f(x2) + 4\, f(x3) + 2\, f(x4) + \cdots + 4\, f(xn-1) + f(b)]$$

Précision. Jusqu'à présent, nous avons développé un certain nombre de règles de quadrature qui combinent efficacement le même ensemble de f(xi) de différentes manières pour obtenir différentes approximations de l'intégrale de f . Chaque approximation est basée sur une hypothèse d'ingénierie différente, il n'est donc pas clair qu'une de ces règles soit meilleure qu'une autre. Ainsi, nous devons développer des estimations d'erreur caractérisant leur comportement respectif. Nous utiliserons nos intégrateurs Newton-Cotes ci-dessus pour montrer comment de telles comparaisons pourraient être effectuées, comme présenté dans CITE.

Considérons d'abord la règle de quadrature du point médian sur un seul intervalle [a, b]. Définir $c \equiv \frac{1}{2}(a + b)$. Le La série de Taylor de f autour de c est :

$$1\, f(x) = f(c) + f\,(c)(x - c) + f\,(c)(x - c)^2 \quad 1\,2\, f\,(c)(x - c) + 6 \quad 1\,3\, f\,(c)(x - c) + 24 \quad 4 + \cdots$$

Ainsi, par symétrie autour de c les termes impairs disparaissent :

$$\int_{un}^{b} f(x)\,dx = (b - une) f(c) + 24 \quad \frac{1}{} f\,(c)(b - une) \quad 3 + \frac{1}{1920} f\,(c)(b - une) \quad 5 + \cdots$$

Notez que le premier terme de cette somme est exactement l'estimation de $\int_{un}^{b} f(x)\,dx$ fourni par le milieu la règle, donc cette règle est précise jusqu'à $O(\Delta x^3)$.

Maintenant, brancher a et b dans notre série de Taylor pour f sur c montre :

$$f(a) = f(c) + f\,(c)(a - c) + \frac{1}{} F\,(c)(a - c)^2\, 1\,2\, F\,(c)(a - c) + 6\, 1\, F\,(c) \quad 3 + \cdots$$

$$F(b) = F(c) + F\,(c)(b - c) + F\,(c)(b - c)^2 \quad 2 + (b - c) \quad 6 \quad 3 + \cdots$$

En les additionnant et en multipliant les deux côtés par b−a/2, on obtient :

$$\frac{f(a) + f(b)}{2}\,1\,(b - a) = f(c)(b - a) + f\,(c)\left((a - c)^2\, 4 + (b - c)^2\right) + \cdots$$

Le terme f (c) disparaît par définition de c. Notez que le côté gauche est l'estimation de l'intégrale de la règle trapézoïdale, et le côté droit est en accord avec notre série de Taylor pour f(x) dx jusqu'au terme cubique. Autrement dit, la règle trapézoïdale est aussi $O(\Delta x^3)$ précis dans un seul intervalle.

Arrêtons-nous ici pour noter un résultat initialement surprenant : les règles du trapèze et du point médian ont le même ordre de précision ! En fait, l'examen du terme de troisième ordre montre que la règle du point médian est environ deux fois plus précise que la règle trapézoïdale. Ce résultat semble contre-intuitif, puisque la règle trapézoïdale utilise une approximation linéaire alors que la règle du milieu est constante.
Comme l'illustre la figure NUMBER, cependant, la règle du point médian récupère en fait l'intégrale des fonctions linéaires, ce qui explique son degré supplémentaire de précision.

Un argument similaire s'applique à la recherche d'une estimation d'erreur pour la règle de Simpson. [ÉCRIRE EXPLANATION ICI ; Omettre DE 205A]. En fin de compte, nous trouvons que la règle de Simpson a une erreur comme $O(\Delta x^5)$

Une mise en garde importante s'applique à ce type d'analyse. En général, le théorème de Taylor ne s'applique que lorsque Δx est suffisamment petit. Si les échantillons sont éloignés les uns des autres, les inconvénients de l'interpolation polynomiale s'appliquent et les phénomènes oscillatoires décrits dans la section NUMBER peuvent entraîner des résultats instables pour les schémas d'intégration d'ordre élevé.

Ainsi, revenant au cas où a et b sont éloignés, nous divisons maintenant [a, b] en intervalles de largeur Δx et appliquons n'importe laquelle de nos règles de quadrature à l'intérieur de ces intervalles. Notez que notre nombre total d'intervalles est $b-a/\Delta x$, nous devons donc multiplier nos estimations d'erreur par $1/\Delta x$ dans ce cas. En particulier, les ordres de précision suivants sont valables :

- Milieu composé : $O(\Delta x^2)$
- Trapèze composé : $O(\Delta x^2)$
- Simpson composée : $O(\Delta x^4)$

12.2.4 Quadrature gaussienne

Dans certaines applications, nous pouvons choisir les emplacements xi auxquels f est échantillonné. Dans ce cas, nous pouvons optimiser non seulement les poids pour la règle de quadrature mais aussi les emplacements xi pour obtenir la meilleure qualité. Cette observation conduit à des règles de quadrature difficiles mais théoriquement attrayantes.

Les détails de cette technique sortent du cadre de notre discussion, mais nous fournissons un chemin simple vers sa dérivation. En particulier, comme dans l'exemple 12.7, supposons que l'on souhaite optimiser x1, . . . , xn et w1, . . . , wn simultanément pour augmenter l'ordre d'un schéma d'intégration. Nous avons maintenant 2n au lieu de n connus, nous pouvons donc appliquer l'égalité pour 2n exemples :

$$\int_a^b f1(x)\,dx = w1\,f1(x1) + w2\,f1(x2) + \cdots + wn\,f1(xn)$$

$$\int_a^b f2(x)\,dx = w1\,f2(x1) + w2\,f2(x2) + \cdots + wn\,f2(xn)$$

$$\vdots \qquad \vdots$$

$$\int_a^b f2n(x)\,dx = w1\,fn(x1) + w2\,fn(x2) + \cdots + wn\,fn(xn)$$

Maintenant, les xi et les wi sont inconnus, donc ce système d'équations n'est plus linéaire. Par exemple, si nous souhaitons optimiser ces valeurs pour des polynômes sur l'intervalle [−1, 1] nous aurions

doivent résoudre le système de polynômes suivant (CITE) :

$$w1 + w2 = \int_{-1}^{1} 1\,dx = 2$$

$$w1x1 + w2x2 = \int_{-1}^{1} x\,dx = 0$$

$$2\,w1x_1\,{}_2 + w2x_2 = \int_{-1}^{1} 2\,2\,dx = x\,3$$

$$3\,3 + w2x_{w1x\,1\,2} = \int_{-1}^{1} {}_{3x\_}\,dx = 0$$

Il se peut que des systèmes comme celui-ci aient des racines multiples et d'autres dégénérescences qui dépendent non seulement du choix des fi (typiquement des polynômes) mais aussi de l'intervalle sur lequel nous approchons une intégrale. De plus, ces règles ne sont pas progressives, en ce sens que l'ensemble des xi pour n points de données n'a rien de commun avec ceux pour k points de données lorsque k = n, il est donc difficile de réutiliser les données pour obtenir une meilleure estimation. D'autre part, lorsqu'elles sont applicables, la quadrature gaussienne a le degré le plus élevé possible pour n fixé. Les règles de quadrature de Kronrod tentent d'éviter ce problème en optimisant la quadrature avec 2n + 1 points tout en réutilisant les points gaussiens.

12.2.5 Quadrature adaptative

Comme nous l'avons déjà montré, il existe certaines fonctions f dont les intégrales sont mieux approchées avec une règle de quadrature donnée que d'autres ; par exemple, les règles médianes et trapézoïdales intègrent des fonctions linéaires avec une précision totale tandis que des problèmes d'échantillonnage et d'autres problèmes peuvent survenir si f oscille rapidement.

Rappelez-vous que la règle de quadrature gaussienne suggère que le placement des xi peut avoir un effet sur la qualité d'un schéma de quadrature. Il reste cependant une information que nous n'avons pas utilisée : les valeurs f(xi). Après tout, ceux-ci déterminent la qualité de notre schéma de quadrature.

Dans cet esprit, les stratégies de quadrature adaptative examinent l'estimation actuelle et génèrent de nouveaux xi où l'intégrande est plus compliquée. Les stratégies d'intégration adaptative comparent souvent la sortie de plusieurs techniques de quadrature, par exemple trapèze et point médian, avec l'hypothèse qu'elles concordent là où l'échantillonnage de f est suffisant (voir la figure NOMBRE). S'ils ne sont pas d'accord avec une certaine tolérance sur un intervalle donné, un point d'échantillonnage supplémentaire est généré et les estimations intégrales sont mises à jour.

AJOUTEZ PLUS DE DÉTAILS OU UN EXEMPLE ; DISCUTER DE L'ALGORITHME RÉCURSIF ; JARS ET GAUTSCHI

12.2.6 Variables multiples On

souhaite souvent intégrer des fonctions f(x) où x Rn . Par exemple, lorsque n = 2, nous pourrions intégrer sur un rectangle en calculant

$$\int_{un}^{b}\int_{c}^{d} f(x, y)\,dx\,dy.$$

Plus généralement, comme l'illustre la figure NUMBER¡, nous pourrions souhaiter trouver une intégrale $\int_{\Omega} f(x)\,dx$, où Ω est un sous-ensemble de Rn .

Une « malédiction de la dimensionnalité » rend l'intégration exponentiellement plus difficile à mesure que la dimension augmente. En particulier, le nombre d'échantillons de f nécessaires pour obtenir une précision de quadrature comparable pour une intégrale dans Rk augmente comme $O(n^k)$. Cette observation peut être décourageante, mais elle est quelque peu raisonnable : plus il y a de dimensions d'entrée pour f , plus il faut d'échantillons pour comprendre son comportement dans toutes les dimensions.

La stratégie d'intégration la plus simple dans Rk est l'intégrale intégrée. Par exemple, si f est une fonction tion de deux variables, supposons que nous voulions trouver $\int_c^d \int_a^b f(x,y)\,dx\,dy$. Pour y fixe, nous pouvons approximer l'intégrale interne sur x en utilisant une règle de quadrature unidimensionnelle ; puis, nous intégrons ces valeurs sur y en utilisant une autre règle de quadrature. Évidemment, les deux schémas d'intégration induisent une certaine erreur, nous devrons donc peut-être échantillonner les xi plus densément que dans une dimension pour obtenir la qualité de sortie souhaitée.

Alternativement, tout comme nous avons subdivisé [a, b] en intervalles, nous pouvons subdiviser Ω en triangles et rectangles en 2D, polyèdres ou boîtes en 3D, et ainsi de suite et utiliser des règles de quadrature interpolatoires simples dans chaque morceau. Par exemple, une option populaire consiste à intégrer la sortie de l'interpolation barycentrique à l'intérieur des polyèdres, puisque cette intégrale est connue sous forme fermée.

Lorsque n est élevé, cependant, il n'est pas pratique de diviser le domaine comme suggéré. Dans ce cas, nous pouvons utiliser la méthode de Monte Carlo randomisée. Dans ce cas, on génère simplement k points aléatoires $x_i \in \Omega$ avec, par exemple, une probabilité uniforme. La moyenne des valeurs f(xi) donne une approximation de $\int_\Omega f(x)\,dx$ qui converge comme $1/\sqrt{k}$ – indépendamment de la dimension de Ω ! Ainsi, en grandes dimensions l'estimation de Monte Carlo est préférable aux méthodes de quadrature déterministes ci-dessus.

PLUS DE DÉTAILS SUR LA CONVERGENCE DE MONTE CARLO ET LE CHOIX DES DISTRIBUTIONS PLUS DE Ω

12.2.7 Conditionnement

Jusqu'à présent, nous avons considéré la qualité d'une méthode de quadrature utilisant des valeurs de précision $O(\Delta x^k)$; évidemment par cette métrique, un ensemble de poids de quadrature avec un grand k est préférable.

Une autre mesure, cependant, équilibre les mesures de précision obtenues à l'aide des arguments de Taylor. En particulier, rappelons que nous avons écrit notre règle de quadrature sous la forme $Q[f] \equiv \sum_i w_i f(x_i)$. Supposons que nous perturbons f en un autre $\hat{f}$. Définissons $\|f - \hat{f}\|_\infty = \max_{x \in [a,b]} |f(x) - \hat{f}(x)|$. Alors,

$$\frac{|Q[f] - Q[\hat{f}]|}{\|f - \hat{f}\|_\infty} = \frac{|\sum_i w_i (f(x_i) - \hat{f}(x_i))|}{\|f - \hat{f}\|_\infty}$$

$$\leq \frac{\sum_i |w_i|\,|f(x_i) - \hat{f}(x_i)|}{\|f - \hat{f}\|_\infty} \quad \text{par l'inégalité triangulaire}$$

$$\leq \|w\|_\infty \quad \text{puisque } |f(x_i) - \hat{f}(x_i)| \leq \|f - \hat{f}\|_\infty \text{ par définition.}$$

Ainsi, la stabilité ou le conditionnement d'une règle de quadrature dépend de la norme de l'ensemble des poids w.

En général, il est facile de vérifier qu'à mesure que l'on augmente l'ordre de précision en quadrature, le conditionnement w se détériore parce que les wi prennent de grandes valeurs négatives ; cela contraste avec le cas tout positif, où le conditionnement est borné par b − a parce que $\sum_i w_i = b - a$ pour les schémas polynomiaux interpolatoires et la plupart des méthodes d'ordre inférieur n'ont que des coefficients positifs (CHECK). Ce fait est le reflet de la même intuition que nous ne devrions pas interpoler des fonctions utilisant des polynômes d'ordre supérieur. Ainsi, en pratique, nous préférons généralement la quadrature composite aux méthodes d'ordre élevé, qui peuvent fournir de meilleures estimations mais peuvent être instables en cas de perturbation numérique.

12.3 Différenciation

L'intégration numérique est un problème relativement stable. en ce que l'influence de toute valeur unique f(x) sur $\int_a^b f(x)\,dx$ se réduit à zéro lorsque a et b s'éloignent. L'approximation de la dérivée d'une fonction f (x), en revanche, n'a pas une telle propriété de stabilité. Du point de vue de l'analyse de Fourier, on peut montrer que l'intégrale f(x) a généralement des fréquences inférieures à f , tandis que la différenciation pour produire f amplifie les hautes fréquences de f , ce qui rend les contraintes d'échantillonnage, le conditionnement et la stabilité particulièrement difficiles pour l'approximation de f .

Malgré les circonstances difficiles, les approximations des dérivées sont généralement relativement faciles à calculer et peuvent être stables en fonction de la fonction à portée de main. En fait, lors du développement de la règle de la sécante, de la méthode de Broyden, etc., nous avons utilisé des approximations simples des dérivées et des gradients pour aider à guider les routines d'optimisation.

Ici, nous nous concentrerons sur l'approximation de f pour f : R → R. La recherche de gradients et de jacobiens est souvent accomplie en différenciant une dimension à la fois, ce qui réduit effectivement le problème unidimensionnel que nous considérons ici.

12.3.1 Différenciation des fonctions de base

Le cas le plus simple de différenciation concerne les fonctions construites à l'aide de routines d'interpolation. Comme au §12.2.1, si on peut écrire $f(x) = \sum_i a_i \varphi_i(x)$ alors par linéarité on sait

$$f'(x) = \sum_i a_i \varphi_i'(x)$$

En d'autres termes, on peut considérer les fonctions φ_i comme une base pour les dérivées de fonctions écrites dans la base φ_i !

Un exemple de cette procédure est illustré à la figure NUMBER. Ce phénomène relie souvent différents schémas d'interpolation. Par exemple, les fonctions linéaires par morceaux ont des dérivées constantes par morceaux, les fonctions polynomiales ont des dérivées polynomiales de degré inférieur, etc. nous reviendrons sur cette structure lorsque nous considérerons les discrétisations des équations aux dérivées partielles. En attendant, il est utile de savoir dans ce cas que f est connu avec une certitude totale, bien que, comme dans la figure NUMBER, ses dérivées puissent présenter des discontinuités indésirables.

12.3.2 Différences finies

Un cas plus courant est que nous avons une fonction f(x) que nous pouvons interroger mais dont les dérivées sont inconnues. Cela se produit souvent lorsque f prend une forme complexe ou lorsqu'un utilisateur fournit f(x) en tant que sous-programme sans informations analytiques sur sa structure.

La définition de la dérivée suggère une approche raisonnable :

$$f'(x) \equiv \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Comme on pouvait s'y attendre, pour un h fini > 0 avec petit |h| l'expression dans la limite fournit une valeur possible se rapprochant de f (x).

Pour étayer cette intuition, on peut utiliser les séries de Taylor pour écrire :

$$f(x+h) = f(x) + f'(x)h + f''(x)h^2 - \cdots$$

La réorganisation de cette expression montre :

$$(x) = \frac{f(x + h) - f(x)}{} f + O(h) h$$

Ainsi, l'approximation de différence directe suivante de f a une convergence linéaire :

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

De même, l'inversion du signe de h montre que les différences en arrière ont également une convergence linéaire :

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

Nous pouvons en fait améliorer la convergence de notre approximation en utilisant une astuce. Par Taylor's théorème on peut écrire :

$$f(x + h) = f(x) + f (x)h + \frac{1}{2} f (x) h^2 + \frac{1}{6} \frac{1}{} f (x) h^3 + \cdots$$

$$f(x - h) = f(x) - f (x)h + \frac{1}{2} f (x) h^2 - \frac{1}{6} f (x) h^3 + \cdots$$

$$1 = f(x + h) - f(x - h) = 2 f (x)h + f (x)h^3 + \cdots$$

$$= \frac{f(x + h) - f(x - h) =}{} f (x) + O(h^2 2h)$$

Ainsi, cette différence centrée donne une approximation de f (x) à convergence quadratique ; c'est l'ordre de convergence le plus élevé que nous pouvons espérer atteindre avec une différence divisée. Nous pouvons cependant obtenir plus de précision en évaluant f en d'autres points, par exemple x + 2h, bien que cette approximation ne soit pas beaucoup utilisée en pratique en faveur d'une simple diminution de h.

La construction d'estimations de dérivées d'ordre supérieur peut se faire par des constructions similaires. Pour exemple, si nous additionnons les développements de Taylor de f(x + h) et f(x − h) nous voyons

$$f(x + h) + f(x - h) = 2 f(x) + f (x)h f(x + ^2 + O(h^3)$$

$$= \frac{h) - 2 f(x) + f(x - h) = f (x) +}{} O(h h 2^2)$$

Pour prédire des combinaisons similaires pour des dérivées plus élevées, une astuce consiste à remarquer que notre deuxième la formule dérivée peut être factorisée différemment :

$$\frac{f(x + h) - 2 f(x) + f(x - h) h}{2} = \frac{\frac{f(x+h)-f(x) h}{} - \frac{f(x)-f(x-h) h}{}}{h}$$

Autrement dit, notre approximation de la dérivée seconde est une "différence finie de différences finies".
Une façon d'interpréter cette formule est illustrée à la figure NUMBER. Lorsque nous calculons l'approximation de la différence directe de f entre x et x + h, nous pouvons penser que cette pente vit à x + h/2 ; nous pouvons de même utiliser les différences en arrière pour placer une pente à x − h/2. Trouver la pente entre ces valeurs remet l'approximation sur x.

Une stratégie qui peut améliorer la convergence des approximations ci-dessus est l'extrap olation de Richardson. Comme exemple d'un modèle plus général, supposons que nous souhaitions utiliser des différences directes pour approximer f . Définir

$$D(h) \equiv h \frac{f(x + h) - f(x)}{}.$$

Évidemment, D(h) tend vers f (x) lorsque h → 0. Plus spécifiquement, cependant, d'après notre discussion au §12.3.2, nous savons que D(h) prend la forme :

$$D(h) = f(x) + \frac{1}{2} f(x)h + O(h^2)$$

Supposons que nous connaissions D(h) et D(αh) pour un certain 0 < α < 1. Nous savons :

$$D(\alpha h) = f(x) + \frac{1}{2} f(x)\alpha h + O(h^2)$$

On peut écrire ces deux relations dans une matrice :

$$\begin{matrix} 1 & \frac{1}{2}h \\ 1 & \frac{1}{2}ah \end{matrix} \begin{matrix} f(x) \\ f(x) \end{matrix} = \begin{matrix} D(h) \\ D(ah) \end{matrix} + O(h^2)$$

Ou équivalent,

$$\begin{matrix} f(x) \\ f(x) \end{matrix} = \begin{matrix} \frac{1}{2}h \\ 1\ 1\ \frac{1}{2}ah \end{matrix}^{-1} \begin{matrix} D(h) \\ D(ah) \end{matrix} + O(h^2)$$

C'est-à-dire que nous avons pris une approximation O(h) de f (x) en utilisant D(h) et l'avons ^{2}) environ transformée en une imation O(h ! Cette technique intelligente est une méthode d'accélération de séquence, car elle améliore l'ordre de convergence des approximation D(h) La même astuce s'applique plus généralement à de nombreux autres problèmes en écrivant une approximation D(h) = a + bhn + O(h m) où m > n, où a est la quantité que l'on espère estimer et b est le terme suivant dans le développement de Taylor. En fait, l'extrapolation de Richardson peut même être appliquée de manière récursive pour faire des approximations d'ordre de plus en plus élevé.

12.3.3 Choix de la taille de pas

Contrairement à la quadrature, la différenciation numérique a une curieuse propriété. Il semble que toute méthode que nous choisissons puisse être arbitrairement précise simplement en choisissant un h suffisamment petit. Cette observation est attrayante du point de vue que nous pouvons obtenir des approximations de meilleure qualité sans temps de calcul supplémentaire. Le hic, cependant, c'est qu'il faut diviser par h et comparer de plus en plus de valeurs similaires f(x) et f(x + h) ; en arithmétique à précision finie, additionner et/ou diviser par des valeurs proches de zéro induit des problèmes et des instabilités numériques. Ainsi, il existe une gamme de valeurs h qui ne sont pas assez grandes pour induire une erreur de discrétisation significative et pas assez petites pour faire des problèmes numériques ; La figure NUMBER montre un exemple de différenciation d'une fonction simple en arithmétique à virgule flottante IEEE.

12.3.4 Grandeurs intégrées

Non couvert dans CS 205A, automne 2013.

12.4 Problèmes

- Quadrature gaussienne - contient toujours des points médians, stratégie utilisant des polynômes orthogonaux
- Quadrature adaptative
- Applications de l'extrapolation de Richardson ailleurs

Chapitre 13

Équations différentielles ordinaires

Nous avons motivé le problème de l'interpolation au chapitre 11 en passant de l'analyse à la recherche de fonctions. Autrement dit, dans des problèmes comme l'interpolation et la régression, l'inconnue est une fonction f , et le travail de l'algorithme est de remplir les données manquantes.

Nous poursuivons cette discussion en considérant des problèmes similaires impliquant le remplissage de valeurs de fonction. Ici, notre inconnue continue d'être une fonction f , mais plutôt que de simplement deviner les valeurs manquantes, nous aimerions résoudre des problèmes de conception plus complexes. Par exemple, considérons les problèmes suivants :

- Trouver f se rapprochant d'une autre fonction f0 mais satisfaisant des critères supplémentaires (lissage, continuité, basse fréquence, etc.).
- Simuler une relation dynamique ou physique comme f(t) où t est le temps.
- Trouver f avec des valeurs similaires à f0 mais certaines propriétés en commun avec une fonction différente g0.

Dans chacun de ces cas, notre inconnue est une fonction f , mais nos critères de réussite sont plus impliqués que "correspond à un ensemble donné de points de données".

Les théories des équations différentielles ordinaires (ODEs) et des équations aux dérivées partielles (PDEs) étudient le cas où l'on souhaite trouver une fonction f(x) à partir d'informations ou de relations entre ses dérivées. Remarquez que nous avons déjà résolu une version simple de ce problème dans notre discussion sur la quadrature : étant donné f (t), les méthodes de quadrature fournissent des moyens d'approximer f(t) en utilisant l'intégration.

Dans ce chapitre, nous considérerons le cas d'une équation différentielle ordinaire et en particulier des problèmes aux valeurs initiales. Ici, l'inconnue est une fonction f(t) : R → Rn et on donne une équation satisfaite par f et ses dérivées ainsi que f(0) ; notre objectif est de prédire f(t) pour t > 0. Nous fournirons plusieurs exemples d'EDO apparaissant dans la littérature informatique, puis procéderons à la description de techniques de résolution courantes.

Pour mémoire, nous utiliserons la notation f pour désigner la dérivée d f/dt de f : [0, ∞) → Rn . Notre le but sera de trouver f(t) étant donné les relations entre t, f(t), f (t), f (t), etc.

13.1 Motivations

Les ODE apparaissent dans presque toutes les parties de l'exemple scientifique, et il n'est pas difficile de rencontrer des situations pratiques nécessitant leur solution. Par exemple, les lois fondamentales du mouvement physique sont données par une ODE :

Exemple 13.1 (Deuxième loi de Newton). Dans la continuité du §5.1.2, rappelez-vous que la deuxième loi du mouvement de Newton stipule que F = ma, c'est-à-dire que la force totale sur un objet est égale à sa masse multipliée par son accélération. Si nous simulons n particules simultanément, alors nous pouvons penser à combiner toutes leurs positions en un vecteur x R3n . De même, nous pouvons écrire une fonction F(t,x,x) R3n prenant le temps, les positions des particules et leurs vitesses et renvoyant la force totale sur chaque particule. Cette fonction peut prendre en compte les interrelations entre les particules (par exemple les forces gravitationnelles ou les ressorts), les effets externes comme la résistance au vent (qui dépend de x), les forces externes variant avec le temps t, etc.

Ensuite, pour trouver les positions de toutes les particules en fonction du temps, on souhaite résoudre l'équation x = F(t,x,x)/m. On nous donne généralement les positions et les vitesses de toutes les particules au temps t = 0 comme condition de départ.

Exemple 13.2 (repliement des protéines). A plus petite échelle, les équations régissant les mouvements des molécules sont aussi des équations différentielles ordinaires. Un cas particulièrement difficile est celui du repliement des protéines, dans lequel la structure géométrique d'une protéine est prédite en simulant des forces intermoléculaires au fil du temps. Ces forces prennent de nombreuses formes souvent non linéaires qui continuent de défier les chercheurs en biologie computationnelle.

Exemple 13.3 (Descente en gradient). Supposons que nous souhaitions minimiser une fonction énergétique E(x) sur tout x. Nous avons appris au chapitre 8 que − E(x) pointe dans la direction où E diminue le plus à un x donné, nous avons donc effectué une recherche linéaire le long de cette direction à partir de x pour minimiser E localement. Une option alternative populaire dans certaines théories est de résoudre une ODE de la forme x = − E(x) ; en d'autres termes, pensez à x en fonction du temps x(t) qui tente de diminuer E en descendant une pente.

Par exemple, supposons que l'on veuille résoudre Ax =b pour A définie positive symétrique. On sait par §10.1.1 x Ax −bx que cela équivaut à minimiser E(x) ≡ x = − f(x) =b $\frac{1}{2}$ + c. Ainsi, nous pourrions tenter de résoudre l'ODE − Ax. Comme t → ∞, on s'attend à ce que x(t) satisfasse de mieux en mieux le système linéaire.

Exemple 13.4 (Simulation de foule). Supposons que nous écrivions un logiciel de jeu vidéo nécessitant une simulation réaliste de foules virtuelles d'humains, d'animaux, de vaisseaux spatiaux, etc. Une stratégie pour générer un mouvement plausible, illustrée dans la figure NUMBER, consiste à utiliser des équations différentielles. Ici, la vitesse d'un membre de la foule est déterminée en fonction de son environnement ; par exemple, dans les foules humaines, la proximité d'autres humains, la distance aux obstacles, etc. peuvent affecter la direction dans laquelle un agent donné se déplace. Ces règles peuvent être simples, mais dans l'ensemble, leur interaction est complexe. Des intégrateurs stables pour les équations différentielles sous-tendent cette machinerie, car nous ne souhaitons pas avoir un comportement sensiblement irréaliste ou non physique.

13.2 Théorie des ODE

Un traitement complet de la théorie des équations différentielles ordinaires sort du cadre de notre discussion, et nous renvoyons le lecteur au CITE pour plus de détails. Ceci mis à part, nous mentionnons ici quelques faits saillants qui seront pertinents pour notre développement dans les sections futures.

13.2.1 Notions de base

Le problème de valeur initiale ODE le plus général prend la forme suivante :

$$\text{Trouver } f(t) : R \to R^n$$

$$\text{Satisfaisant } F[t, f(t), f'(t), f''(t), \ldots, f^{(k)}(t)] = 0 \text{ Étant}$$

$$\text{donné } f(0), f'(0), f''(0), \ldots, f^{(k-1)}(0)$$

Ici, F est une relation entre f et toutes ses dérivées ; nous utilisons $f^{()}$ pour désigner la dérivée -ième de f . Nous pouvons considérer les ODE comme déterminant l'évolution de f au cours du temps t ; nous connaissons f et ses dérivées au temps zéro et souhaitons le prédire à l'avenir.

Les ODE prennent plusieurs formes même dans une seule variable. Par exemple, notons $y = f(t)$ et supposons $y \in R^1$. Ensuite, des exemples d'ODE incluent :

- $y = 1 + \cos t$: cette ODE peut être résolue en intégrant les deux côtés, par exemple en utilisant la quadrature méthodes
- $y = ay$: cette ODE est linéaire en y
- $y = ay + e^t$: Cet ODE dépend du temps et de la position
- $y + 3y - y = t$: cette ODE implique plusieurs dérivées de y
- $y \sin y = e^{ty}$: Cet ODE est non linéaire en y et t.

De toute évidence, les ODE les plus générales peuvent être difficiles à résoudre. Nous restreindrons la majeure partie de notre discussion au cas des ODE explicites, dans lesquelles la dérivée d'ordre le plus élevé peut être isolée :

Définition 13.1 (ODE explicite). Une ODE est explicite si elle peut s'écrire sous la forme

$$f^{(k)}(t) = F[t, f(t), f'(t), f''(t), \ldots, f^{(k-1)}(t)].$$

Par exemple, une forme explicite de la deuxième loi de Newton est $x''(t) = \frac{1}{m}a(t, x(t), x'(t))$.

Étonnamment, en généralisant l'astuce du §5.1.2, en fait, toute ODE explicite peut être convertie en une équation du premier ordre $f'(t) = F[t, f(t)]$, où f a une sortie multidimensionnelle. Cette observation implique que nous n'avons pas besoin de plus d'une dérivée dans notre traitement des algorithmes ODE. Pour voir cette relation, rappelons simplement que $d^2y/dt^2 = d/dt(dy/dt)$. Ainsi, on peut définir une variable intermédiaire $z \equiv dy/dt$, et comprendre d^2y/dt^2 comme dz/dt avec la contrainte $z = dy/dt$. Plus généralement, si l'on veut résoudre le problème explicite

$$f^{(k)}(t) = F[t, f(t), f'(t), f''(t), \ldots, f^{(k-1)}(t)],$$

où $f : R \to R^n$, alors on définit $g(t) : R \to R^{kn}$ en utilisant l'ODE du premier ordre :

$$\frac{d}{dt}\begin{bmatrix} g_1(t) \\ g_2(t) \\ \vdots \\ g_{k-1}(t) \\ g_k(t) \end{bmatrix} = \begin{bmatrix} g_2(t) \\ g_3(t) \\ \vdots \\ g_k(t) \\ F[t, g_1(t), g_2(t), \ldots, g_{k-1}(t)] \end{bmatrix}$$

Ici, on note gi(t) : R → Rn pour contenir n composantes de g. Alors, g1(t) satisfait l'ODE d'origine. Pour le voir, nous vérifions simplement que notre équation ci-dessus implique g2(t) = g 1 (t), g3(t) = g 2 (t) = g 1 (t), et ainsi de suite. Ainsi, la réalisation de ces substitutions montre que la ligne finale code l'ODE d'origine.

L'astuce ci-dessus va simplifier notre notation, mais il faut prendre garde de bien comprendre que cette approche ne banalise pas les calculs. En particulier, dans de nombreux cas, notre fonction f(t) n'aura qu'une seule sortie, mais l'ODE sera en plusieurs dérivées. Nous remplaçons ce cas par une dérivée et plusieurs sorties.

Exemple 13.5 (extension ODE). Supposons que nous souhaitions résoudre y = 3y − 2y + y où y(t) : R → R. Cette équation est équivalente à :

$$\frac{d}{dt}\begin{matrix} y \\ z \\ w \end{matrix} = \begin{matrix} 0\ 1\ 0 \\ 0\ 0\ 1 \\ 1\ -2\ 3 \end{matrix} \begin{matrix} y \\ z \\ w \end{matrix}$$

Tout comme notre astuce ci-dessus nous permet de ne considérer que les ODE du premier ordre, nous pouvons restreindre encore plus notre notation aux ODE autonomes. Ces équations sont de la forme f (t) = F[f(t)], c'est-à-dire que F ne dépend plus de t. Pour ce faire, nous pourrions définir

$$g(t) \equiv \begin{matrix} f(t) \\ g^{-}(t) \end{matrix} .$$

Ensuite, nous pouvons résoudre l'ODE suivante pour g à la place :

$$g\,(t) = \begin{matrix} f\,(t) \\ g^{-}\,(t) \end{matrix} = \begin{matrix} F[\,f(t),\, g^{-}(t)] \\ 1 \end{matrix} .$$

En particulier, $g^{-}(t) = t$ en supposant que l'on prend $g^{-}(0) = 0$.

Il est possible de visualiser le comportement des ODE de plusieurs manières, illustrées dans la figure NUMBER. Par exemple, si l'inconnue f(t) est une fonction d'une seule variable, alors nous pouvons penser que F[f(t)] fournit la pente de f(t), comme le montre la figure NUMBER. Alternativement, si f(t) a une sortie dans R2 , nous ne pouvons plus visualiser la dépendance au temps t, mais nous pouvons dessiner l'espace des phases, qui montre la tangente de f(t) à chaque (x, y) R2 .

13.2.2 Existence et unicité

Avant de pouvoir passer aux discrétisations du problème de la valeur initiale, nous devons brièvement reconnaître que tous les problèmes d'équations différentielles ne sont pas solubles. De plus, certaines équations différentielles admettent plusieurs solutions.

Exemple 13.6 (ODE insoluble). Considérons l'équation y = 2y/t, avec y(0) = 0 donné ; notez que nous ne divisons pas par zéro car y(0) est prescrit. Réécrire comme

$$\frac{1}{y}\frac{dt}{} = \frac{2}{t}$$

et l'intégration par rapport à t des deux côtés montre :

$$\text{en } |y| = 2 \ln t + c \text{ Ou}$$

de manière équivalente, y = Ct2 pour un C R. Remarquons que y(0) = 0 dans cette expression, ce qui contredit nos conditions initiales. Ainsi, cet ODE n'a pas de solution avec les conditions initiales données.

Exemple 13.7 (Solutions non uniques). Maintenant, considérons la même ODE avec y(0) = 0. Considérons y(t) donné par y(t) = Ct2 pour tout C R. Alors, y (t) = 2Ct. Ainsi,

$$\frac{2}{\text{ans}} = \frac{2Ct2}{t} = 2Ct = y\,(t),$$

montrant que l'ODE est résolu par cette fonction indépendamment de C. Ainsi, les solutions de ce problème ne sont pas uniques.

Heureusement, il existe une riche théorie caractérisant le comportement et la stabilité des solutions aux équations différentielles. Notre développement dans le chapitre suivant aura un ensemble plus fort de conditions nécessaires à l'existence d'une solution, mais en fait sous des conditions faibles sur f il est possible de montrer qu'une ODE f (t) = F[f(t)] a une solution. Par exemple, un tel théorème garantit l'existence locale d'une solution :

Théorème 13.1 (Existence locale et unicité). Supposons que F soit continue et Lipschitz, c'est-à-dire que F[y] − F[x]2 ≤ Ly −x2 pour un certain L. Alors, l'ODE f (t) = F[f(t)] admet exactement une solution pour tout t ≥ 0 quelles que soient les conditions initiales.

Dans notre développement ultérieur, nous supposerons que l'ODE que nous essayons de résoudre satisfait les conditions d'un tel théorème ; cette hypothèse est assez réaliste dans la mesure où, au moins localement, il faudrait un comportement assez dégénéré pour briser des hypothèses aussi faibles.

13.2.3 Équations du modèle

Une façon d'acquérir une intuition pour le comportement des ODE est d'examiner le comportement des solutions à certaines équations modèles simples qui peuvent être résolues sous forme fermée. Ces équations représentent des linéarisations d'équations plus pratiques, et donc localement elles modélisent le type de comportement auquel nous pouvons nous attendre.

Nous commençons avec les ODE dans une seule variable. Compte tenu de nos simplifications au §13.2.1, l'équation la plus simple avec laquelle on pourrait s'attendre à travailler serait y = F[y], où y(t) : R → R. Prendre une approximation linéaire donnerait des équations de type y = ay + b. La substitution $y^{-} \equiv y + b/a$ montre : $y^{-} = y = ay + b = a(y^{-} - b/a) + b = ay^{-}$. Ainsi, dans nos équations modèles, la constante b induit simplement un décalage, et pour notre étude phénoménologique dans cette section, nous pouvons supposer b = 0.

Par l'argument ci-dessus, nous pouvons localement comprendre le comportement de y = F[y] en étudiant le linéaire équation y = ay. En fait, l'application d'arguments standard du calcul montre que

$$y(t) = Ceat\ .$$

Évidemment, il y a trois cas, illustrés dans la figure NUMBER :

1. a > 0 : Dans ce cas, les solutions deviennent de plus en plus grandes ; en fait, si y(t) et yˆ(t) vérifient tous les deux ODE avec des conditions de départ légèrement différentes, car t → ∞ elles divergent.

2. a = 0 : le système dans ce cas est résolu par des constantes ; les solutions avec des points de départ différents restent à la même distance.

3. a < 0 : Alors, toutes les solutions de l'ODE tendent vers 0 lorsque t → ∞.

On dit que les cas 2 et 3 sont stables, dans le sens où perturber légèrement y(0) donne des solutions qui se rapprochent de plus en plus avec le temps ; le cas 1 est instable, car une petite erreur dans la spécification du paramètre d'entrée y(0) sera amplifiée au fur et à mesure que le temps t avance. Les ODE instables génèrent des problèmes de calcul mal posés ; sans un examen attentif, nous ne pouvons pas nous attendre à ce que les méthodes numériques génèrent des solutions utilisables dans ce cas, car même les sorties théoriques sont si sensibles aux perturbations de l'entrée.
D'un autre côté, les problèmes stables sont bien posés puisque les petites erreurs en y(0) diminuent avec le temps.

En passant à plusieurs dimensions, nous pourrions étudier l'équation linéarisée

$$y = Oui.$$

Comme expliqué au §5.1.2, si y1, · · · ,yk sont des vecteurs propres de A de valeurs propres λ1, . . . , λk et y(0) = alors
$c_1y_1 + \cdots + c_ky_k$,

$$y(t) = c_1e^{\lambda_1 t}y_1 + \cdots + c_ke^{\lambda_k t}y_k.$$

En d'autres termes, les valeurs propres de A prennent la place de a dans notre exemple unidimensionnel. À partir de ce résultat, il n'est pas difficile de deviner qu'un système multivariable est stable exactement lorsque son rayon spectral est inférieur à un.

En réalité, nous souhaitons résoudre y = F[y] pour les fonctions générales F. En supposant que F est différentiable, nous pouvons écrire F[y] ≈ F[y0] + JF(y0)(y − y0), donnant l'équation modèle ci-dessus après un quart de travail. Ainsi, pour de courtes périodes de temps, nous nous attendons à un comportement similaire à l'équation du modèle. De plus, les conditions du théorème 13.1 peuvent être considérées comme une limite sur le comportement de JF, fournissant une connexion à des théories moins localisées d'ODE.

13.3 Schémas de pas de temps

Nous procédons maintenant à la description de plusieurs méthodes de résolution de l'ODE non linéaire y = F[y] pour des fonctions potentiellement non linéaires F. En général, étant donné un « pas de temps » h, nos méthodes seront utilisées pour générer des estimations de y(t + h) étant donné y(t). L'application itérative de ces méthodes génère des estimations de y0 ≡ y(t), y1 ≡ y(t + h), y2 ≡ y(t + 2h), y3 ≡ y(t + 3h), etc. Notez que puisque F n'a pas de dépendance t, le mécanisme de génération de chaque étape supplémentaire est le même que le premier, donc pour la plupart, nous n'aurons besoin de décrire qu'une seule étape de ces méthodes. Nous appelons des méthodes pour générer des approximations des intégrateurs y(t) , reflétant le fait qu'ils intègrent les dérivées dans l'équation d'entrée.

L'idée de stabilité est d'une importance capitale pour notre réflexion. Tout comme les ODE peuvent être stables ou instables, les discrétisations le peuvent aussi. Par exemple, si h est trop grand, certains schémas accumuleront l'erreur à un taux exponentiel ; à l'opposé, d'autres méthodes sont stables en ce que même si h est grand, les solutions resteront bornées. La stabilité, cependant, peut rivaliser avec la précision ; souvent les schémas stables dans le temps sont de mauvaises approximations de y(t), même s'il est garanti qu'ils n'ont pas de comportement sauvage.

13.3.1 Euler avant

Notre première stratégie ODE provient de notre construction du schéma de différenciation directe au §12.3.2 :

$$F[y_k] = y(t) = \frac{y_{k+1} - y_k}{h} + O(h)$$

Résoudre cette relation pour yk+1 montre

$$y_{k+1} = y_k + hF[y_k] + O(h^2) \approx y_k + hF[y_k].$$

Ainsi, le schéma d'Euler direct applique la formule à droite pour estimer yk+1 . C'est l'une des stratégies les plus efficaces pour le pas de temps, car elle évalue simplement F et ajoute un multiple du résultat à yk . Pour cette raison, nous l'appelons une méthode explicite, c'est-à-dire qu'il existe une formule explicite pour yk+1 en termes de yk et F.

L'analyse de la précision de cette méthode est assez simple. Notez que notre approximation), donc chaque de yk+1 est O(h ^{2}étape induit une erreur quadratique. Nous appelons cette erreur erreur de troncature localisée car il s'agit de l'erreur induite par une seule étape ; le mot « troncature » fait référence au fait que nous avons tronqué une série de Taylor pour obtenir cette formule. Bien sûr, notre iterateyk peut déjà être inexact grâce aux erreurs de troncature accumulées des itérations précédentes. Si nous intégrons de t0 à t avec O(1/h) étapes, alors notre erreur totale ressemble à O(h); cette estimation représente l'erreur de troncature globale, et nous écrivons donc généralement que le schéma d'Euler avant est "précis au premier ordre".

La stabilité de cette méthode nécessite un peu plus de considération. Dans notre discussion, nous travaillerons sur la stabilité des méthodes dans le cas à une variable y = ay, avec l'intuition que des déclarations similaires se répercutent sur les équations multidimensionnelles en remplaçant a par le rayon spectral. Dans ce cas, nous savons

$$y_{k+1} = y_k + ahy_k = (1 + ah)y_k .$$

Autrement dit, yk = (1 + ah) ky0. Ainsi, l'intégrateur est stable lorsque |1 + ah| ≤ 1, car sinon |yk | → ∞ exponentiellement. En supposant a < 0 (sinon le problème est mal posé), on peut simplifier :

$$\begin{aligned} |1 + ah| \le 1 \quad & -1 \le 1 + ah \le 1 \quad -2 \\ & \le ah \le 0 \\ & 0 \le h \le \frac{2}{|a|}, \text{ puisque } a < 0 \end{aligned}$$

Ainsi, Euler en avant admet une restriction de pas de temps pour la stabilité donnée par notre condition finale sur h. En d'autres termes, la sortie d'Euler vers l'avant peut exploser même lorsque y = ay est stable si h n'est pas assez petit. La figure NUMBER illustre ce qui se passe lorsque cette condition est respectée ou violée.
En plusieurs dimensions, nous pouvons remplacer cette restriction par une restriction analogue utilisant le rayon spectral de A. Pour les ODE non linéaires, cette formule donne un guide de stabilité au moins localement dans le temps ; globalement h peut devoir être ajusté si le jacobien de F devient moins conditionné.

13.3.2 Euler en arrière

De même, nous aurions pu appliquer le schéma de différenciation vers l'arrière à yk+1 pour concevoir un intégrateur ODE :

$$F[y_{k+1}] = y(t) = h \frac{y_{k+1} - y_k}{} + O(h)$$

Ainsi, nous résolvons le système d'équations potentiellement non linéaire suivant pour yk+1 :

$$y_k = y_{k+1} - hF[y_{k+1}].$$

Parce que nous devons résoudre cette équation pour yk+1 , Euler en arrière est un intégrateur implicite.

Cette méthode est précise au premier ordre comme Euler en avant par une preuve identique. La stabilité de cette méthode contraste cependant considérablement avec l'avant Euler. Considérant à nouveau l'équation modèle y = ay, on écrit :

$$y_k \quad y_k = y_{k+1} - hay_{k+1} = \quad y_{k+1} = \frac{}{1 - ha} \cdot$$

Parallèlement à notre argument précédent, Euler en arrière est stable sous la condition suivante :

$$\frac{1}{|} \leq 1 \qquad |1 - ha| \geq 1 \; |1 - ha$$

$$1 - ha \leq -1 \text{ ou } 1 - ha \geq 1$$

$$2 \qquad h \leq \text{ou} - h \geq 0, \text{ pour } a < 0$$

Évidemment, nous prenons toujours h ≥ 0, donc Euler en arrière est inconditionnellement stable.

Bien sûr, même si Euler en arrière est stable, il n'est pas nécessairement précis. Si h est trop grand, yk s'approchera beaucoup trop rapidement de zéro. Lors de la simulation de tissus et d'autres matériaux physiques qui nécessitent beaucoup de détails à haute fréquence pour être réalistes, l'Euler en arrière peut ne pas être un choix efficace.
De plus, nous devons inverser F[·] pour résoudre yk+1 .

Exemple 13.8 (Euler arrière). Supposons que l'on veuille résoudre y = Ay pour A Rn×n . Ensuite, pour trouver yk+1 on résout le système suivant :

$$y_k = y_{k+1} - hAy_{k+1} = \quad y_{k+1} = (I_{n\times n} - hA)^{-1} y_k .$$

13.3.3 Méthode trapézoïdale

Supposons que yk soit connu à l'instant tk et que yk+1 représente la valeur à l'instant tk+1 = tk + h. Supposons que nous connaissions également yk+1/2 à mi-chemin entre ces deux étapes. Alors, par notre dérivation de la différenciation centrée nous savons :

$$y_{k+1} = y_k + hF[y_{k+1/2}] + O(h^3)$$

De notre dérivation de la règle trapézoïdale :

$$\frac{F[y_{k+1}] + F[y_k]}{2} = F[y_{k+1/2}] + O(h^2)$$

La substitution de cette relation donne notre premier schéma d'intégration de second ordre, la méthode trapézoïdale pour l'intégration des ODE :

$$y_{k+1} = y_k + h\frac{F[y_{k+1}] + F[y_k]}{2}$$

Comme Euler à rebours, cette méthode est implicite puisqu'il faut résoudre cette équation pour yk+1 .

En effectuant à nouveau une analyse de stabilité sur y = ay, on retrouve dans ce cas des pas de temps de la méthode trapézoïdale solve

$$y_{k+1} = y_k + \frac{1}{2} - ha(y_{k+1} + y_k)$$

Autrement dit,

$$y_k = \frac{1 + \frac{1}{2}ha}{1 - \frac{1}{2}hak}\, y_0.$$

La méthode est donc stable lorsque

$$\frac{\text{ha } 1 + \frac{1}{2}}{1 - \frac{1}{2}\text{ha}} < 1.$$

Il est facile de voir que cette inégalité est vraie chaque fois que a < 0 et h > 0, montrant que la méthode du trapèze est inconditionnellement stable.

Malgré son ordre de précision supérieur avec une stabilité maintenue, la méthode trapézoïdale présente cependant certains inconvénients qui la rendent moins populaire que l'Euler en arrière. En particulier, considérez les
rapport

$$R \equiv \frac{\text{oui+1}}{\text{beurk}} = \frac{1 + \frac{1}{2}\text{Ha}}{2\,1\,1 - \frac{1}{2}\text{Ha}}$$

Lorsque a < 0, pour h suffisamment grand, ce rapport finit par devenir négatif ; en fait, comme h → ∞, on a R → −1. Ainsi, comme l'illustre la figure NUMBER, si les pas de temps sont trop grands, la méthode d'intégration trapézoïdale a tendance à présenter un comportement oscillatoire indésirable qui n'est pas du tout comme ce à quoi on pourrait s'attendre pour des solutions de y = ay .

13.3.4 Méthodes Runge-Kutta

Une classe d'intégrateurs peut être dérivée en faisant l'observation suivante :

$$\text{yk+1} = \text{yk} + \int_{tk}^{tk+\Delta t} \text{y (t) dt par le théorème fondamental du calcul}$$

$$= \text{yk} + \int_{tk}^{tk+\Delta t} \text{F[y(t)] dt}$$

Bien sûr, l'utilisation pure et simple de cette formule ne fonctionne pas pour formuler une méthode de pas de temps puisque nous ne connaissons pas y(t), mais une application prudente de nos formules de quadrature du chapitre précédent peut générer des stratégies réalisables.

Par exemple, supposons que nous appliquions la méthode trapézoïdale pour l'intégration. Ensuite, on trouve :

$$\text{h yk+1} = \text{yk} + \frac{}{2}\,(\text{F[yk]} + \text{F[yk+1]}) + \text{O(h}^{3}\text{)}$$

C'est la formule que nous avons écrite pour la méthode trapézoïdale au §13.3.3.

Si nous ne souhaitons pas résoudre pour yk+1 implicitement, cependant, nous devons trouver une expression à approxi).
mat F[yk+1]. En utilisant la méthode d'Euler, cependant, nous savons que yk+1 = yk + hF[yk] + ^{2}Fabrication
O(h cette substitution pour yk+1 n'affecte pas l'ordre d'approximation du pas de temps trapézoïdal ci-dessus, donc nous pouvons écrire :

$$\text{yk+1} = \text{yk} + \frac{\text{h}}{2} - (\text{F[yk]} + \text{F[yk + hF[yk]]}) + \text{O(h}^{3}\text{)}$$

Ignorer le O(h second ^{3}) termes donne une nouvelle stratégie d'intégration connue sous le nom de méthode de Heun, qui est ordre précis et explicite.

Si nous étudions le comportement de stabilité de la méthode de Heun pour y = ay pour a < 0, nous savons :

$$h\ y_{k+1} = y_k + \frac{}{} (ay_k + a(y_k + hay_k))$$

$$+ 2 \qquad \frac{2}{} h\ a(2 + ha)\ y_k = 1$$

$$1 = 1 + ha + h\ 2 \quad \frac{}{} \quad \frac{2\ 2\ un}{beurk}$$

Ainsi, la méthode est stable lorsque

$$1 - 1 \le 1 + ha + h\ 2\ 2 - 2 \quad \frac{2\ 2\ un}{} \le 1$$

$$-4 \le 2ha + h \quad \frac{une}{} \le 0$$

L'inégalité de droite montre h ≤ a < 0, donc la $\frac{}{2}$ et celui de gauche est toujours vrai pour h > 0 et |a| ,

condition de stabilité est h ≤ La méthode de Heun $\frac{}{2\ |a|}$.

est un exemple de méthode de Runge-Kutta dérivée en appliquant des méthodes de quadrature à l'intégrale ci-dessus et en remplaçant les pas d'Euler dans F[·]. Forward Euler est une méthode Runge-Kutta précise du premier ordre, et la méthode de Heun est du second ordre. Une méthode populaire de Runge Kutta du quatrième ordre (abrégé « RK4 ») est donnée par :

$$= y_k + 6 \qquad \frac{h}{} (k1 + 2k2 + 2k3 + k4)\ y_{k+1}$$

où$k1 = F [y_k]$

$$k2 = F\ y_k + \frac{1}{2} hk1$$

$$k3 = F\ y_k + \frac{1}{2} hk2$$

$$k4 = F\ y_k + hk3$$

Cette méthode peut être dérivée en appliquant la règle de Simpson pour la quadrature.

Les méthodes de Runge-Kutta sont populaires car elles sont explicites et donc faciles à évaluer tout en offrant un haut degré de précision. Le coût de cette précision, cependant, est que F[·] doit être évalué plusieurs fois. De plus, les stratégies de Runge-Kutta peuvent être étendues à des méthodes implicites capables de résoudre des équations rigides.

13.3.5 Intégrateurs exponentiels Une classe

d'intégrateurs qui atteint une grande précision lorsque F[·] est approximativement linéaire consiste à utiliser explicitement notre solution à l'équation du modèle. En particulier, si nous résolvions l'ODE y = Ay, en utilisant les vecteurs propres de A (ou toute autre méthode) nous pourrions trouver une solution explicite y(t) comme expliqué au §13.2.3. Nous écrivons généralement $y_{k+1} = e^{Ah} y_k$, encode notre exponentiation des valeurs propres (en fait nous pouvons trouver une matrice e Maintenant, si nous écrivons où e^{Ah} à partir de cette expression qui résout l'ODE au temps h).

$$y = Ay + G[y],$$

où G est une fonction non linéaire mais petite, nous pouvons obtenir une précision assez élevée en intégrant explicitement la partie A, puis en rapprochant la partie G non linéaire séparément. Par exemple, l'intégrateur exponentiel du premier ordre applique Euler vers l'avant au terme G non linéaire :

$$Ah\ y_{k+1} = e\ y_k - A^{-1}(1 - e\ Ah)G[y_k]$$

L'analyse révélant les avantages de cette méthode est plus complexe que ce que nous avons écrit, mais intuitivement il est clair que ces méthodes se comporteront particulièrement bien lorsque G est petit.

13.4 Méthodes multivaleurs

Les transformations du §13.2.1 nous ont permis de simplifier considérablement la notation de la section précédente en réduisant toutes les ODE explicites à la forme y = F[y]. En fait, bien que tous les ODE explicites puissent être écrits de cette façon, il n'est pas certain qu'ils le soient toujours.

En particulier, lorsque nous avons réduit les ODE d'ordre k en ODE de premier ordre, nous avons introduit un certain nombre de variables représentant les premières à k − 1ères dérivées de la sortie souhaitée. En fait, dans notre solution finale, nous ne nous soucions que de la dérivée zéro, c'est-à-dire de la fonction elle-même, donc les ordres de précision sur les variables temporaires sont moins importants.

De ce point de vue, considérons la série de Taylor

$$2\ y(t_k + h) = y(t_k) + hy\ (t_k) + y\ (t_k) + \frac{h}{2} + \mathcal{O}(h^3)$$

Si on ne connaît que y jusqu'à $O(h^2)$, cela n'affecte pas notre approximation, puisque y est multiplié par h. De même, si nous ne connaissons y que jusqu'à O(h), cette approximation n'affectera pas les termes de la série de Taylor ci-dessus car elle sera multipliée par h 2/2. Ainsi, nous considérons maintenant meth « multivaleur » (k) (t) = F[t,y intégrer y différentes dérivées de (t),y (t), . . . ,y (k−1) (t)] avec une précision d'ordre différent pour ods, conçue pour la fonction y.

Etant donné l'importance de la seconde loi de Newton F = ma, nous nous restreindrons au cas y = F[t,y,y]; de nombreuses extensions existent pour le cas moins courant du k-ième ordre. Nous introduisons un vecteur « vitesse » v(t) = y (t) et un vecteur « accélération »a. Par notre réduction précédente, nous souhaitons résoudre le système du premier ordre suivant :

$$\begin{aligned} y\ (t) &= v(t)\ v \\ (t) &= a(t)\ a(t) = \\ & F[t,y(t),v(t)] \end{aligned}$$

Notre objectif est de dériver un intégrateur spécifiquement adapté à ce système.

13.4.1 Systèmes Newmark

Nous commençons par dériver la fameuse classe des intégrateurs de Newmark.[1] Soit y_k , v_k et a_k les vecteurs position, vitesse et accélération au temps t_k ; notre but est d'avancer jusqu'au temps $t_{k+1} \equiv t_k + h$.

[1] Nous suivons le développement dans http://www.stanford.edu/group/frg/course_work/AA242B/CA-AA242B-Ch7.pdf.

Utilisez y(t), v(t) et a(t) pour désigner les fonctions du temps en supposant que nous commençons à tk . Alors évidemment nous pouvons écrire

$$v_{k+1} = v_k + \int_{t_k}^{t_{k+1}} a(t)\,dt$$

On peut aussi écrire yk+1 comme une intégrale impliquant a(t), en suivant quelques étapes :

$$\begin{aligned}
y_{k+1} &= y_k + \int_{t_k}^{t_{k+1}} v(t)\,dt \\
&= y_k + [tv(t)]_{t_k}^{t_{k+1}} - \int_{t_k}^{t_{k+1}} ta(t)\,dt && \text{après intégration par parties} \\
&= y_k + t_{k+1}v_{k+1} - t_k v_k - \int_{t_k}^{t_{k+1}} ta(t)\,dt && \text{en développant le terme de différence} \\
&= y_k + hv_k + t_{k+1}v_{k+1} - t_{k+1}v_k - \int_{t_k}^{t_{k+1}} ta(t)\,dt && \text{en ajoutant et en soustrayant } hv_k \\
&= y_k + hv_k + t_{k+1}(v_{k+1} - v_k) - \int_{t_k}^{t_{k+1}} ta(t)\,dt && \text{après factorisation} \\
&= y_k + hv_k + t_{k+1}\int_{t_k}^{t_{k+1}} a(t)\,dt - \int_{t_k}^{t_{k+1}} ta(t)\,dt && \text{puisque } v'(t) = a(t) \\
&= y_k + hv_k + \int_{t_k}^{t_{k+1}} (t_{k+1} - t)a(t)\,dt
\end{aligned}$$

Supposons que nous choisissions $\tau \in [t_k, t_{k+1}]$. On peut alors écrire les expressions forak etak+1 en utilisant la série de Taylor autour de τ :

$$\begin{aligned}
a_k &= a(\tau) + a'(\tau)(t_k - \tau) + O(h^2) \\
a_{k+1} &= a(\tau) + a'(\tau)(t_{k+1} - \tau) + O(h^2)
\end{aligned}$$

Pour toute constante $\gamma \in \mathbb{R}$, si nous mettons à l'échelle la première équation de 1 − γ et la seconde de γ et additionnons les résultats, nous trouvons :

$$\begin{aligned}
a(\tau) &= (1-\gamma)a_k + \gamma a_{k+1} + a'(\tau)((\gamma - 1)(t_k - \tau) - \gamma(t_{k+1} - \tau)) + O(h^2) \\
&= (1-\gamma)a_k + \gamma a_{k+1} + a'(\tau)(\tau - h\gamma - t_k) + O(h^2) \text{ après substitution } t_{k+1} = t_k + h
\end{aligned}$$

Au final, nous souhaitons intégrera de tk à tk+1 pour obtenir le changement de vitesse. Ainsi, on calcule :

$$\begin{aligned}
\int_{t_k}^{t_{k+1}} a(\tau)\,d\tau &= (1-\gamma)ha_k + \gamma h a_{k+1} + \int_{t_k}^{t_{k+1}} a'(\tau)(\tau - h\gamma - t_k)\,d\tau + O(h^3) \\
&= (1-\gamma)ha_k + \gamma h a_{k+1} + O(h^2),
\end{aligned}$$

où la deuxième étape est vérifiée car l'intégrande est O(h) et l'intervalle d'intégration est de largeur h. En d'autres termes, nous savons maintenant :

$$v_{k+1} = v_k + (1-\gamma)ha_k + \gamma h a_{k+1} + O(h^2)$$

Pour faire une approximation similaire pour yk+1 , on peut écrire

$$\begin{aligned}
\int_{t_k}^{t_{k+1}} (t_{k+1} - t)a(t)\,dt &= \int_{t_k}^{t_{k+1}} (t_{k+1} - \tau)((1-\gamma)a_k + \gamma a_{k+1} + a'(\tau)(\tau - h\gamma - t_k))\,d\tau + O(h^3) \\
&= \frac{1}{2}(1-\gamma)h^2 a_k + \left(\frac{1}{2} - \gamma\right)h^2 a_{k+1} + O(h^2) \text{ par un argument similaire.}
\end{aligned}$$

Ainsi, nous pouvons utiliser notre relation précédente pour montrer :

$$y_{k+1} = y_k + hv_k + \int_{t_k}^{t_{k+1}} (t_{k+1} - t)a(t)\,dt \text{ d'avant}$$

$$= y_k + hv_k + \left(\frac{1}{2} - \beta\right)h^2 a_k + \beta h^2 a_{k+1} + O(h^2)$$

Ici, nous utilisons β au lieu de γ (et absorbons un facteur de deux dans le processus) car le γ que nous choisissons pour approximer y_{k+1} n'a pas à être le même que celui que nous choisissons pour approximer v_{k+1} .

Après toute cette intégration, nous avons dérivé la classe des schémas de Newmark, avec deux entrées paramètres γ et β, qui ont une précision de premier ordre par la preuve ci-dessus :

$$y_{k+1} = y_k + hv_k + \left(\frac{1}{2} - \beta\right)h^2 a_k + \beta h^2 a_{k+1}$$

$$v_{k+1} = v_k + (1 - \gamma)ha_k + \gamma ha_{k+1}$$

$$a_k = F[t_k, y_k, v_k]$$

Différents choix de β et γ conduisent à différents schémas. Par exemple, considérez les exemples suivants :

- β = γ = 0 donne l'intégrateur d'accélération constante :

$$y_{k+1} = y_k + hv_k + \frac{1}{2}h^2 a_k$$

$$v_{k+1} = v_k + ha_k$$

  Cet intégrateur est explicite et tient exactement lorsque l'accélération est une fonction constante.

- β = 1/2, γ = 1 donne l'intégrateur d'accélération implicite constante :

$$y_{k+1} = y_k + hv_k + \frac{1}{2}h^2 a_{k+1}$$

$$v_{k+1} = v_k + ha_{k+1}$$

  Ici, la vitesse est échelonnée implicitement en utilisant Euler vers l'arrière, ce qui donne une précision de premier ordre. La mise à jour de y, cependant, peut s'écrire

$$y_{k+1} = y_k + \frac{1}{2}h(v_k + v_{k+1}),$$

  montrant qu'il est mis à jour localement par la règle du point médian ; ceci est notre premier exemple d'un schéma où les mises à jour de vitesse et de position ont différents ordres de précision. Même ainsi, il est possible de montrer que cette technique est cependant globalement précise au premier ordre en y.

- β = 1/4, γ = 1/2 donne le schéma trapézoïdal du second ordre suivant après un peu d'algèbre :

$$x_{k+1} = x_k + \frac{1}{2}h(v_k + v_{k+1})$$

$$v_{k+1} = v_k + \frac{1}{2}h(a_k + a_{k+1})$$

- β = 0, γ = 1/2 donne un schéma de différenciation centrale précis du second ordre. Dans le canonique forme, nous avons

$$x_{k+1} = x_k + hv_k + h\, a_k \frac{1}{2}$$

$$v_{k+1} = v_k + \frac{1}{2} h(a_k + a_{k+1})$$

La méthode tire son nom du fait que la simplification des équations ci-dessus conduit à la forme alternative :

$$v_{k+1} = \frac{y_{k+2} - y_k}{2h}$$

$$y_{k+1} - \frac{2y_{k+1} + y_k}{h^2} a_{k+1} =$$

Il est possible de montrer que les méthodes de Newmark sont inconditionnellement stables lorsque $4\beta > 2\gamma > 1$ et que la précision du second ordre se produit exactement lorsque $\gamma = 1/2$ (CHECK).

13.4.2 Grille décalée

Une autre façon d'obtenir une précision de second ordre iny consiste à utiliser des différences centrées autour du temps $t_{k+1/2} \equiv t_k + h/2$:

$$y_{k+1} = y_k + hv_{k+1/2}$$

Plutôt que d'utiliser des arguments de Taylor pour essayer de déplacer $v_{k+1/2}$, nous pouvons simplement stocker les vitesses v à des demi-points sur la grille des pas de temps.

Ensuite, nous pouvons utiliser une mise à jour similaire pour avancer les vitesses :

$$v_{k+3/2} = v_{k+1/2} + ha_{k+1} .$$

Notez que cette mise à jour est également précise au second ordre pour x, car si nous remplaçons nos expressions par $v_{k+1/2}$ et $v_{k+3/2}$, nous pouvons écrire :

$$a_{k+1} = \frac{1}{h^2} (y_{k+2} - 2y_{k+1} + y_k)$$

Enfin, une simple approximation suffit pour le terme d'accélération puisqu'il s'agit d'un terme d'ordre supérieur :

$$a_{k+1} = F\left[t_{k+1}, x_{k+1}, \frac{1}{2}(v_{k+1/2} + v_{k+3/2})\right]$$

Cette expression peut être substituée dans l'expression pour $v_{k+3/2}$.

Lorsque F[·] ne dépend pas de v, la méthode est parfaitement explicite :

$$y_{k+1} = y_k + hv_{k+1/2}$$

$$a_{k+1} = F[t_{k+1}, y_{k+1}]$$

$$v_{k+3/2} = v_{k+1/2} + ha_{k+1}$$

C'est ce qu'on appelle la méthode d'intégration saute-mouton, grâce à la grille de temps échelonnée et au fait que chaque point médian est utilisé pour mettre à jour la vitesse ou la position suivante.

Sinon, si la mise à jour de la vitesse dépend de v, la méthode devient implicite.

Souvent, la dépendance à la vitesse est symétrique ; par exemple, la résistance au vent change simplement de signe si vous inversez la direction dans laquelle vous vous déplacez. Cette propriété peut conduire à des matrices symétriques dans l'étape implicite de mise à jour des vitesses, permettant d'utiliser des gradients conjugués et des méthodes itératives rapides associées pour résoudre.

13.5 À faire

- Définir l'ODE rigide
- Donner un tableau des méthodes de pas de temps pour F[t;y]
- Utilisez la notation y de manière plus cohérente

13.6 Problèmes

- TVD RK
- Méthodes multi-étapes/multi-valeurs à la Heath
- Intégration Verlet
- Intégrateurs symplectiques

Chapitre 14

Équations aux dérivées partielles

Notre intuition pour les équations différentielles ordinaires découle généralement de l'évolution temporelle des systèmes physiques. Des équations comme la deuxième loi de Newton déterminant le mouvement des objets physiques dans le temps dominent la littérature sur ces problèmes de valeur initiale; des exemples supplémentaires proviennent de concentrations chimiques réagissant au fil du temps, de populations de prédateurs et de proies interagissant de saison en saison, etc. Dans chaque cas, la configuration initiale — par exemple les positions et les vitesses des particules dans un système à l'instant zéro — est connue, et la tâche consiste à prédire le comportement au fil du temps.

Dans ce chapitre, cependant, nous envisageons la possibilité de coupler des relations entre différentes dérivées d'une fonction. Il n'est pas difficile de trouver des exemples où ce couplage est nécessaire. Par exemple, lors de la simulation de quantités de fumée ou de gaz telles que des « gradients de pression », la dérivée de la pression d'un gaz dans l'espace, détermine comment le gaz se déplace dans le temps ; cette structure est raisonnable puisque le gaz diffuse naturellement des régions à haute pression vers les régions à basse pression. Dans le traitement d'images, les dérivées se couplent encore plus naturellement, puisque les mesures sur les images ont tendance à se produire simultanément dans les directions x et y.

Les équations couplant entre elles des dérivées de fonctions sont appelées équations aux dérivées partielles. Ils font l'objet d'une théorie riche mais fortement nuancée digne d'un traitement à plus grande échelle, notre objectif ici sera donc de résumer les idées clés et de fournir suffisamment de matériel pour résoudre les problèmes apparaissant couramment dans la pratique.

14.1 Motivations

Les équations aux dérivées partielles (EDP) apparaissent lorsque l'inconnue est une fonction f : Rn → Rm. On nous donne une ou plusieurs relations entre les dérivées partielles de f , et le but est de trouver un f qui satisfait aux critères. Les EDP apparaissent dans presque toutes les branches des mathématiques appliquées, et nous en énumérons quelques-unes ci-dessous.

En aparté, avant d'introduire des PDE spécifiques, nous devrions introduire quelques notations. En particulier, il existe quelques combinaisons de dérivées partielles qui apparaissent souvent dans le monde des EDP. Si f : R3 → R et v : R3 → R3, alors les opérateurs suivants méritent d'être rappelés :

$$\text{Dégradé :} \quad \nabla f \equiv \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \right)$$

$$\text{Divergence :} \quad \cdot v \equiv + + \frac{\partial v1}{\partial x1} \frac{\partial v2}{\partial x2} \frac{\partial v3}{\partial x3} \frac{\partial v3}{\partial v3} \frac{\partial v2}{\partial x2} \frac{\partial x3}{\partial 2} \frac{\partial 2}{\partial}$$

$$\text{Courbure :} \quad \times v \equiv \frac{2 \text{ fff}}{} - \frac{}{}, \frac{\partial v1}{\partial x3} - \frac{\partial v3}{\partial x1}, \frac{\partial v2}{\partial x1} - \frac{\partial v1}{\partial x2}$$

$$\text{Laplacien :} \quad 2 f \equiv + + \frac{\partial x 2}{\partial x 2} \frac{\partial x 2}{\partial x 2} \frac{1}{3} 2$$

Exemple 14.1 (Simulation fluide). L'écoulement de fluides comme l'eau et la fumée est régi par les équations de Navier Stokes, un système d'EDP dans de nombreuses variables. En particulier, supposons qu'un fluide se déplace dans une région $\Omega \quad R3$. Nous définissons les variables suivantes, illustrées dans la figure NUMBER :

- $t \quad [0, \infty)$: Temps
- $v(t) : \Omega \to R3$: La vitesse du fluide
- $\rho(t) : \Omega \to R$: La masse volumique du fluide
- $p(t) : \Omega \to R$: La pression du fluide
- $f(t) : \Omega \to R3$: Forces externes comme la gravité sur le fluide

Si le fluide a une viscosité μ, alors si nous supposons qu'il est incompressible, les équations de Navier-Stokes indiquent :

$$\rho \frac{\partial v}{\partial t} + v \cdot \quad v = - \quad p + \mu \quad 2 v + f$$

Ici, $2v = \partial 2v1/\partial x 2 + \partial 2v2/\partial x 2 + \partial 2v3/\partial x 2$; nous considérons le gradient comme un gradient dans l'espace plutôt que $\partial f \partial f \partial x1 , \partial x2 ,$ le temps, c'est- à-dire $\frac{}{1} \frac{}{} \frac{\partial f}{\partial x3}$). Ce système d'équations détermine la dynamique temporelle du mouvement des fluides et $f = ($ peut en fait être construit en appliquant la deuxième loi de Newton au suivi des "particules" de fluide. Son énoncé, cependant, n'implique pas seulement des dérivées dans le temps $\frac{}{\partial t}$ mais aussi des dérivées dans l'espace , ce qui en fait une EDP.

Exemple 14.2 (équations de Maxwell). Les équations de Maxwell déterminent l'interaction des champs électriques E et des champs magnétiques B dans le temps. Comme pour les équations de Navier-Stokes, nous considérons le gradient, la divergence et la courbure comme prenant des dérivées partielles dans l'espace (et non dans le temps t). Alors, le système de Maxwell (sous forme "forte") peut s'écrire :

$$\text{Loi de Gauss pour les champs électriques :} \quad \cdot E = \frac{\rho}{\varepsilon 0}$$

$$\text{Loi de Gauss pour le magnétisme :} \quad \cdot B = 0$$

$$\text{Loi de Faraday :} \quad \times E = - \frac{\partial B}{\partial t}$$

$$\text{Loi d'Ampère :} \quad \times B = \mu 0 J + \varepsilon 0 \frac{\partial E}{\partial t}$$

Ici, $\varepsilon 0$ et $\mu 0$ sont des constantes physiques et J code la densité de courant électrique. Tout comme les équations de Navier Stokes, les équations de Maxwell reliaient les dérivées des grandeurs physiques dans le temps t à leurs dérivées dans l'espace (données par les termes de boucle et de divergence).

Exemple 14.3 (équation de Laplace). Supposons que Ω soit un domaine dans R2 de bord ∂Ω et qu'on nous donne une fonction g : ∂Ω → R, illustrée sur la figure NUMBER. On peut vouloir interpoler g à l'intérieur de Ω. Lorsque Ω est une forme irrégulière, cependant, nos stratégies d'interpolation du chapitre 11 peuvent échouer.

Supposons que nous définissions f(x) : Ω → R comme étant la fonction d'interpolation. Ensuite, une stratégie inspirée de notre approche des moindres carrés consiste à définir une fonctionnelle énergétique :

$$E[f] = \int_\Omega \|\nabla f(x)\|_2^2 \, dx$$

C'est-à-dire que E[f] mesure la « dérivée totale » de f mesurée en prenant la norme de son gradient et en intégrant cette quantité sur tout Ω. Les fonctions f extrêmement fluctuantes auront des valeurs élevées de E[f] puisque la pente f sera grande en de nombreux endroits ; les fonctions lisses et basses fréquences f , par contre, auront E[f] petit puisque leur pente sera petite partout.1 Alors, on pourrait demander que f interpole g en étant aussi lisse que possible à l'intérieur de Ω en utilisant l'optimisation suivante :

$$\begin{aligned} &\text{minimiser}_f \; E[f] \\ &\text{tel que } f(x) = g(x) \quad x \in \partial\Omega \end{aligned}$$

Cette configuration ressemble aux optimisations que nous avons résolues dans les exemples précédents, mais maintenant notre inconnue est une fonction f plutôt qu'un point dans Rn !

Si f minimise E, alors E[f + h] ≥ E[f] pour toutes les fonctions h(x). Cette affirmation est vraie même pour de petites perturbations E[f + εh] lorsque ε → 0. En divisant par ε et en prenant la limite comme ε → 0, nous devons avoir $\frac{d}{d\varepsilon} E[f + \varepsilon h]|_{\varepsilon=0} = 0$; c'est comme si les dérivées directionnelles d'une fonction étaient égales à zéro pour trouver ses minima. On peut simplifier :

$$\begin{aligned} E[f + \varepsilon h] &= \int_\Omega \|\nabla f(x) + \varepsilon \nabla h(x)\|_2^2 \, dx \\ &= \int_\Omega \left(\|\nabla f(x)\|_2^2 + 2\varepsilon \nabla f(x) \cdot \nabla h(x) + \varepsilon^2 \|\nabla h(x)\|_2^2 \right) dx \end{aligned}$$

Spectacles différenciants :

$$\begin{aligned} \frac{d}{d\varepsilon} E[f + \varepsilon h] &= \int_\Omega \left(2 \nabla f(x) \cdot \nabla h(x) + 2\varepsilon \|\nabla h(x)\|_2^2 \right) dx \\ \frac{d}{d\varepsilon} E[f + \varepsilon h]|_{\varepsilon=0} &= 2 \int_\Omega [\nabla f(x) \cdot \nabla h(x)] \, dx \end{aligned}$$

Cette dérivée doit être nulle pour tout h, donc en particulier on peut choisir h(x) = 0 pour tout x ∈ ∂Ω. Alors, en appliquant l'intégration par parties, on a :

$$\frac{d}{d\varepsilon} E[f + \varepsilon h]|_{\varepsilon=0} = -2 \int_\Omega h(x) \, \nabla^2 f(x) \, dx$$

Cette expression doit être égale à zéro pour toutes (toutes !) les perturbations h, donc nous devons avoir $\nabla^2 f(x) = 0$ pour tout x ∈ Ω\∂Ω (une preuve formelle sort du cadre de notre discussion). Autrement dit, le problème d'interpolation ci-dessus

1La notation E[·] utilisée ici ne signifie pas « espérance » comme elle pourrait le faire dans la théorie des probabilités, mais est simplement une fonctionnelle « énergie » ; c'est la notation standard dans les domaines de l'analyse fonctionnelle.

peut être résolu en utilisant l'EDP suivante :

$$2 f(x) = 0 \; f(x) = g(x) \quad x \quad \partial\Omega$$

Cette équation est connue sous le nom d'équation de Laplace, et elle peut être résolue en utilisant des méthodes linéaires définies positives creuses comme ce que nous avons vu au chapitre 10. Comme nous l'avons vu, elle peut être appliquée à des problèmes d'interpolation pour des domaines irréguliers Ω ; en outre, E[f] peut être augmenté pour mesurer d'autres propriétés de f , par exemple dans quelle mesure f se rapproche d'une fonction bruyante f0, pour dériver des PDE associées en mettant en parallèle l'argument ci-dessus.

Exemple 14.4 (Équation d'Eikonal). Supposons que Ω Rn est une région fermée de l'espace. Ensuite, nous pourrions considérer que d(x) est une fonction mesurant la distance d'un point x0 à x complètement dans Ω. Lorsque Ω est convexe, on peut écrire d sous forme fermée :

$$d(x) = x - x02.$$

Comme l'illustre la figure NUMBER, cependant, si Ω est non convexe ou un domaine plus compliqué comme une surface, les distances deviennent plus compliquées à calculer. Dans ce cas, les fonctions de distance d satisfont la condition localisée connue sous le nom d'équation eikonale :

$$d2 = 1.$$

Si nous pouvons le calculer, d peut être utilisé pour des tâches telles que la planification de trajectoires de robots en minimisant la distance qu'ils doivent parcourir avec la contrainte qu'ils ne peuvent se déplacer qu'en Ω.

Des algorithmes spécialisés connus sous le nom de méthodes de marche rapide sont utilisés pour trouver des estimations de d étant donné x0 et Ω en intégrant l'équation eikonale. Cette équation est non linéaire dans la dérivée d, donc les méthodes d'intégration pour cette équation sont quelque peu spécialisées, et la preuve de leur efficacité est complexe. Fait intéressant mais sans surprise, de nombreux algorithmes pour résoudre l'équation eikonale ont une structure similaire à l'algorithme de Dijkstra pour calculer les chemins les plus courts le long des graphes.

Exemple 14.5 (Analyse harmonique). Différents objets réagissent différemment aux vibrations, et en grande partie ces réponses sont fonction de la géométrie des objets. Par exemple, les violoncelles et les pianos peuvent jouer la même note, mais même un musicien inexpérimenté peut facilement distinguer les sons qu'ils produisent. D'un point de vue mathématique, on peut prendre Ω R3 comme une forme représentée soit comme une surface soit comme un volume.
Si nous fixons les bords de la forme, alors son spectre de fréquence est donné par les solutions du problème aux valeurs propres différentielles suivant :

$$2 f = \lambda f$$

$$f(x) = 0 \quad x \quad \partial\Omega, \text{ où}$$

2 est le laplacien de Ω et ∂Ω est la frontière de Ω. La figure NUMBER montre des exemples de ces fonctions sur différents domaines Ω.

Il est facile de vérifier que sin kx résout ce problème lorsque Ω est l'intervalle [0, 2π], pour k Z. En particulier, le Laplacien à une dimension est $\partial 2/\partial x 2$, et ainsi on peut vérifier :

$$\frac{\partial 2}{\partial x} \partial \sin kx = k \cos kx \; \partial x 2$$

$$2 = -k \sin kx$$

$$\sin k \cdot 0 = 0$$

$$\sin k \cdot 2\pi = 0$$

Ainsi, les fonctions propres sont sin kx avec des valeurs propres −k 2 ·

14.2 Définitions de base

En utilisant la notation de CITE, nous supposerons que notre inconnue est une fonction f : Rn → R. Pour les équations jusqu'à trois variables, nous utiliserons la notation en indice pour désigner les dérivées partielles :

$$f_x \equiv \frac{\partial f}{\partial x},$$

$$f_y \equiv \frac{\partial f}{\partial y \partial},$$

$$f_{xy} \equiv \frac{2 f}{\partial x \partial y},$$

et ainsi de suite.

Les dérivées partielles sont généralement exprimées comme des relations entre deux ou plusieurs dérivées de comme f , dans ce qui suit :

- Linéaire, homogène : fxx + fxy − fy = 0 •

Linéaire : fxx − y fyy + f = xy2

- Non linéaire : $f_{xx}^{2} = f_{xy}$

Généralement, on souhaite vraiment trouver f : Ω → R pour un certain Ω Rn . Tout comme les EDO ont été énoncées comme des problèmes de valeur initiale, nous énoncerons la plupart des EDP comme des problèmes de valeur aux limites. C'est-à-dire que notre travail sera de remplir f à l'intérieur de Ω étant donné les valeurs sur sa frontière. En fait, nous pouvons penser au problème de la valeur initiale de l'ODE de cette manière : le domaine est Ω = [0, ∞), avec la frontière ∂Ω = {0}, où nous fournissons les données d'entrée. La figure NUMBER illustre des exemples plus complexes. Les conditions aux limites de ces problèmes prennent plusieurs formes :

- Les conditions de Dirichlet spécifient simplement la valeur de f(x) sur ∂Ω
- Les conditions de Neumann spécifient les dérivées de f(x) sur ∂Ω
- Les conditions Mixed ou Robin combinent ces deux

14.3 Équations du modèle

Rappelons-nous du chapitre précédent que nous avons pu comprendre de nombreuses propriétés des ODE en examinant une équation modèle y = ay. Nous pouvons tenter de poursuivre une approche similaire pour les EDP, même si nous constaterons que l'histoire est plus nuancée lorsque les dérivés sont liés entre eux.

Comme pour l'équation du modèle pour les ODE, nous étudierons le cas linéaire à variable unique. Nous nous limiterons également aux systèmes du second ordre, c'est-à-dire aux systèmes contenant au plus la dérivée seconde de u ; le modèle ODE était du premier ordre mais ici nous avons besoin d'au moins deux ordres pour étudier comment les dérivées interagissent de manière non triviale.

Une EDP linéaire du second ordre a la forme générale suivante :

$$c = 0 \; a_{ij} \frac{\partial F}{\partial x_i \partial x_j} \frac{\partial F}{\partial x_i} + \sum b_i +$$

Formellement, nous pourrions définir « l'opérateur de gradient » comme suit :

$$\equiv \quad \frac{\partial}{\partial x1}, \frac{\partial}{\partial x2}, \ldots, \frac{\partial}{\partial xn} \quad .$$

Vous devez vérifier que cet opérateur est une notation raisonnable dans la mesure où des expressions telles que f , · v et ×v fournissent toutes les expressions appropriées. Dans cette notation, la PDE peut être considérée comme prenant une forme matricielle :

$$(\quad A \quad + \quad \cdot b + c)f = 0.$$

Cette forme a beaucoup en commun avec notre étude des formes quadratiques dans les gradients conjugués, et en fait nous caractérisons généralement les PDE par la structure de A :

- Si A est définie positive ou négative, le système est elliptique.
- Si A est semi-défini positif ou négatif, le système est parabolique.
- Si A n'a qu'une seule valeur propre de signe différent des autres, le système est hyperbolique.
- Si A ne satisfait aucun des critères, le système est ultrahyperbolique.

Ces critères sont énumérés approximativement dans l'ordre du niveau de difficulté de résolution de chaque type d'équation. Nous considérons les trois premiers cas ci-dessous et fournissons des exemples de comportement correspondant ; les équations ultrahyperboliques n'apparaissent pas aussi souvent dans la pratique et nécessitent des techniques hautement spécialisées pour leur résolution.

À FAIRE : Réduction à la forme canonique via la matière propre de A (pas dans 205A)

14.3.1 EDP elliptiques

Tout comme les matrices définies positives permettent des algorithmes spécialisés comme la décomposition de Cholesky et les gradients conjugués qui simplifient leur inversion, les EDP elliptiques ont une structure particulièrement forte qui conduit à des techniques de résolution efficaces.

Le modèle elliptique PDE est l'équation de Laplace, donnée par 2 f = g pour une fonction donnée g comme dans l'exemple 14.3. Par exemple, à deux variables, l'équation de Laplace devient

$$fxx + fyy = g.$$

La figure NUMBER illustre quelques solutions de l'équation de Laplace pour différents choix de u et f sur un domaine bidimensionnel.

Les équations elliptiques ont de nombreuses propriétés importantes. L'idée de régularité elliptique, selon laquelle les solutions des EDP elliptiques sont automatiquement des fonctions lisses dans C ∞(Ω), revêt une importance théorique et pratique particulière. Cette propriété n'est pas immédiatement évidente : une EDP du second ordre dans f nécessite seulement que f soit deux fois différentiable pour avoir un sens, mais en fait, sous des conditions faibles, elles sont automatiquement infiniment différentiables. Cette propriété donne l'intuition physique que les équations elliptiques représentent des équilibres physiques stables comme la pose de repos d'une feuille de caoutchouc étirée.
Les équations elliptiques du second ordre sous la forme ci-dessus sont également garanties d'admettre des solutions, contrairement aux PDE sous certaines autres formes.

Exemple 14.6 (Poisson dans une variable). L'équation de Laplace avec g = 0 reçoit le nom spécial d'équation de Poisson. Dans une variable, on peut écrire f (x) = 0, qui se résout trivialement par f(x) = αx + β. Cette équation est suffisante pour examiner d'éventuelles conditions aux limites sur [a, b] :

- Les conditions de Dirichlet pour cette équation spécifient simplement f(a) et f(b) ; il y a évidemment une droite unique qui passe par (a, f(a)) et (b, f(b)), qui fournit la solution de l'équation.

- Les conditions de Neumann spécifieraient f (a) et f (b). Mais, f (a) = f (b) = α pour f(x) = αx + β. De cette manière, les valeurs aux limites des problèmes de Neumann peuvent être soumises aux conditions de compatibilité nécessaires pour admettre une solution. De plus, le choix de β n'affecte pas les conditions aux limites, donc lorsqu'elles sont satisfaites, la solution n'est pas unique.

14.3.2 EDP paraboliques

En continuant à mettre en parallèle la structure de l'algèbre linéaire, les systèmes d'équations semi-définis positifs ne sont que légèrement plus difficiles à traiter que les systèmes définis positifs. En particulier, les matrices semi-définies positives admettent un espace nul qui doit être traité avec précaution, mais dans les directions restantes, les matrices se comportent de la même manière que le cas défini.

L'équation de la chaleur est le modèle parabolique PDE. Supposons que f(0; x, y) est une distribution de chaleur dans une région Ω R2 au temps t = 0. Ensuite, l'équation de la chaleur détermine comment la chaleur se diffuse au cours du temps t en fonction f(t; x, y) :

$$\frac{\partial f}{} = \alpha \quad 2 f , \partial t$$

où α > 0 et nous considérons à nouveau 2 comme le Laplacien dans les variables d'espace x et y, c'est-à-dire 2 = ∂ 2/∂x 2 + ∂ 2/∂y 2 . Cette équation doit être parabolique, puisqu'il y a le même coefficient α devant fxx et fyy, mais ftt ne figure pas dans l'équation.

La figure NUMBER illustre une interprétation phénoménologique de l'équation de la chaleur. Nous pouvons penser que 2 f mesure la convexité de f , comme dans la figure NUMBER(a). Ainsi, l'équation de la chaleur augmente u avec le temps lorsque sa valeur est « évasée » vers le haut, et diminue f dans le cas contraire. Cette rétroaction négative est stable et conduit à l'équilibre lorsque t → ∞.

Il y a deux conditions aux limites nécessaires pour l'équation de la chaleur, qui viennent toutes deux avec interprétations physiques directes :

- La distribution de chaleur f(0; x, y) au temps t = 0 en tous points (x, y) Ω

- Comportement de f quand t > 0 aux points (x, y) ∂Ω. Ces conditions aux limites décrivent le comportement à la frontière du domaine. Les conditions de Dirichlet fournissent ici f(t; x, y) pour tout t ≥ 0 et (x, y) ∂Ω, correspondant à la situation dans laquelle un agent extérieur fixe les températures au bord du domaine. Ces conditions peuvent se produire si Ω est un morceau de papier d'aluminium placé à côté d'une source de chaleur dont la température n'est pas affectée de manière significative par le papier d'aluminium comme un grand réfrigérateur ou un four. En revanche, les conditions de Neumann spécifient la dérivée de f dans la direction normale à la frontière ∂Ω, comme dans la figure NUMBER ; ils correspondent à la fixation du flux de chaleur sur Ω provoqué par différents types d'isolants.

14.3.3 EDP hyperboliques

L'équation finale du modèle est l'équation d'onde, correspondant au cas de la matrice indéfinie :

$$\frac{\partial 2}{\partial t 2} f - c 2 \quad 2 f = 0$$

L'équation d'onde est hyperbolique car la dérivée seconde dans le temps a un signe opposé aux deux dérivées spatiales. Cette équation détermine le mouvement des ondes à travers un milieu élastique comme une feuille de caoutchouc ; par exemple, il peut être dérivé en appliquant la deuxième loi de Newton à des points sur un morceau d'élastique, où x et y sont des positions sur la feuille et f(t; x, y) est la hauteur du morceau d'élastique au temps t.

La figure NUMBER illustre une solution unidimensionnelle de l'équation d'onde. Le comportement des vagues contraste considérablement avec la diffusion de la chaleur dans la mesure où l'énergie $t \to \infty$ peut ne pas se diffuser. En particulier, les ondes peuvent rebondir indéfiniment sur un domaine. Pour cette raison, nous verrons que les stratégies d'intégration implicites peuvent ne pas être appropriées pour intégrer des EDP hyperboliques car elles ont tendance à amortir le mouvement.

Les conditions aux limites de l'équation d'onde sont similaires à celles de l'équation de la chaleur, mais maintenant nous devons spécifier à la fois f(0; x, y) et ft(0; x, y) au temps zéro :

- Les conditions à t = 0 spécifient la position et la vitesse de l'onde au temps initial.
- Les conditions aux limites sur Ω déterminent ce qui se passe aux extrémités du matériau. Les conditions de Dirichlet correspondent à la fixation des côtés de l'onde, par exemple en pinçant une corde de violoncelle, qui est maintenue à plat à ses deux extrémités sur l'instrument. Les conditions de Neumann correspondent à laisser intactes les extrémités de l'onde comme l'extrémité d'un fouet.

14.4 Dérivés en tant qu'opérateurs

Dans les EDP et ailleurs, nous pouvons considérer les dérivées comme des opérateurs agissant sur les fonctions de la même manière que les matrices agissent sur les vecteurs. Notre choix de notation reflète souvent ce parallèle : La dérivée d f/dx ressemble au produit d'un opérateur d/dx et d'une fonction f . En fait, la différenciation est un opérateur linéaire au même titre que la multiplication matricielle, puisque pour tout f , g : R → R et a, b R

$$\frac{d}{dx}(af(x) + bg(x)) = a\frac{d}{dx}f(x) + b\frac{d}{dx}g(x).$$

En fait, lorsque nous discrétisons des EDP pour une solution numérique, nous pouvons réaliser cette analogie complètement. Par exemple, considérons une fonction f sur [0, 1] discrétisée à l'aide de n + 1 échantillons régulièrement espacés, comme dans la figure NUMBER. Rappelons que l'espace entre deux échantillons est h = 1/n. Au chapitre 12, nous avons développé une approximation pour la dérivée seconde f (x) :

$$f(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h)$$

Supposons que nos n échantillons de f(x) sur [0, 1] soient y0 ≡ f(0), y1 ≡ f(h), y2 ≡ f(2h), . . . , yn = f(nh).
Ensuite, l'application de notre formule ci-dessus donne une stratégie pour approximer f à chaque point de la grille :

$$y_k'' \equiv \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2}$$

Autrement dit, la dérivée seconde d'une fonction sur une grille de points peut être calculée à l'aide du gabarit $1 \quad -2 \quad 1$ illustré à la figure NUMBER(a).

Une subtilité que nous n'avons pas abordée est ce qui se passe à y 0 et y n , puisque la formule ci-dessus nécessiterait y−1 et yn+1. En fait, cette décision encode les conditions aux limites introduites au §14.2.
En gardant à l'esprit que y0 = f(0) et yn = f(1), des exemples de conditions aux limites possibles pour f incluent :

• Conditions aux limites de Dirichlet : y−1 = yn+1 = 0, c'est-à-dire fixer simplement la valeur de y au-delà de points de terminaison

• Conditions aux limites de Neumann : y−1 = y0 et yn+1 = yn, codant la condition aux limites f (0) = f (1) = 0.

• Conditions aux limites périodiques : y−1 = yn et yn+1 = y0, faisant l'identification f(0) = f(1)

Supposons que nous empilions les échantillons yk en un vecteur y Rn+1 et les échantillons y k dans une seconde vecteur w Rn+1 . Alors, notre construction ci-dessus, il est facile de voir que h 2w = L1y, où L1 est l'un des choix ci-dessous :

$$\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \quad \begin{bmatrix} -1 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -1 \end{bmatrix} \quad \begin{bmatrix} -2 & 1 & & & & 1 \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 1 & & & & 1 & -2 \end{bmatrix}$$

Dirichlet Neumann Périodique

Autrement dit, la matrice L peut être considérée comme une version discrétisée de l'opérateur y Rn+1 $\frac{d}{2\,dx2}$ agissant sur plutôt que des fonctions f : [0, 1] → R.

Nous pouvons écrire une approximation similaire pour 2 f lorsque nous échantillonnons f : [0, 1] × [0, 1] → R avec une grille de valeurs, comme dans la figure NUMBER. En particulier, rappelez-vous que dans ce cas 2 f = fxx + fyy, nous pouvons donc en particulier additionner les dérivées secondes x et y comme nous l'avons fait dans l'exemple unidimensionnel ci-dessus. Cela conduit à un gabarit 1— − 2—1 doublé , comme dans la figure NUMBER. Si nous numérotons nos échantillons comme yk, ≡ f(kh, h), alors notre formule pour le Laplacien de f est dans ce cas :

$$(\ 2\ y)k, \equiv h^{-2}\ (y(k-1), + yk,(-1) + y(k+1), + yk,(+1) - 4yk,)$$

Si nous combinons à nouveau nos échantillons de y et y en y et w, puis en utilisant une construction et un choix similaires de conditions aux limites, nous pouvons à nouveau écrire h 2w = L2y. Cette grille bidimensionnelle laplacienne L2 apparaît dans de nombreuses applications de traitement d'image, où (k,) est utilisé pour indexer les pixels sur une image.

Une question naturelle à poser après la discussion ci-dessus est de savoir pourquoi nous avons sauté au Laplacien dérivé second dans notre discussion ci-dessus plutôt que de discrétiser la dérivée première f (x). En principe, il n'y a aucune raison pour laquelle nous ne pourrions pas créer des matrices similaires D mettant en œuvre des approximations de différence avant, arrière ou symétriques de f . Quelques détails techniques, cependant, rendent cette tâche un peu plus difficile, comme détaillé ci-dessous.

Plus important encore, nous devons décider quelle approximation de la dérivée première utiliser. Si nous écrivons y k (yk+1 − yk), 1 comme la différence avant h − par exemple, alors nous serons dans la position anormalement asymétrique d'avoir besoin d'une condition aux limites en yn mais pas en y0. En revanche, on pourrait utiliser la différence symétrique (yk+1 − yk−1), mais cette discrétisation souffre d'un piquet plus subtil $\frac{1}{2}$ problème illustré dans la figure NUMÉRO. En particulier, cette version de y k ignore la valeur de yk et ne regarde que ses voisins yk−1 et yk+1 , ce qui peut créer des artefacts puisque chaque ligne de D n'implique yk que pour k pair ou impair mais pas les deux.

Si nous utilisons des dérivées avant ou arrière pour éviter les problèmes de clôture, nous perdons un ordre de précision et souffrons également des asymétries décrites ci-dessus. Comme pour l'intégration saute-mouton

Dans l'algorithme §13.4.2, une façon d'éviter ces problèmes est de penser que les dérivées vivent sur des demi-points de grille, illustrés dans la figure NUMBER. Dans le cas unidimensionnel, ce changement correspond à (yk+1 étiqueter la différence sur les $\frac{1}{h}$ yk) comme yk+1/2 . Cette technique consistant à placer différentes dérivées pour sommets, les arêtes et les centres des cellules de la grille est particulièrement courante dans la simulation des fluides, qui maintient les pressions, les vitesses des fluides, etc. à des emplacements qui simplifient les calculs.

Ces subtilités mises à part, notre principale conclusion de cette discussion est que si nous discrétisons une fonction f(x) en gardant une trace des échantillons (xi , yi) alors les approximations les plus raisonnables des dérivées de f seront calculables comme un produit Lx pour une certaine matrice L. Cette observation complète l'analogie : « Les dérivés agissent sur les fonctions comme les matrices agissent sur les vecteurs. Ou en notation d'examen standardisée : Dérivées : Fonctions :: Matrices : Vecteurs

14.5 Résolution numérique des EDP

Il reste encore beaucoup à dire sur la théorie des EDP. Les questions d'existence et d'unicité ainsi que la possibilité de caractériser des solutions à des EDP variées conduisent à des discussions nuancées utilisant des aspects avancés de l'analyse réelle. Bien qu'une compréhension complète de ces propriétés soit nécessaire pour prouver rigoureusement l'efficacité des discrétisations PDE, nous en avons déjà assez pour suggérer quelques techniques qui sont utilisées dans la pratique.

14.5.1 Résolution d'équations elliptiques

Nous avons déjà fait l'essentiel du travail de résolution des EDP elliptiques au §14.4. En particulier, supposons que l'on veuille résoudre une EDP linéaire elliptique de la forme Lf = g. Ici L est un opérateur différentiel ; par exemple, pour résoudre l'équation de Laplace on prendrait L ≡ 2 le Laplacien. Puis, au §14.4 nous avons montré que si l'on discrétise f en prenant un ensemble d'échantillons dans un vecteur y avec yi = f(xi), alors une approximation correspondante de Lf peut s'écrire Ly pour une certaine matrice L. Si on discrétise aussi g en utilisant des échantillons dans un vecteurb, puis la résolution de l'EDP elliptique Lf = g est approchée en résolvant le système linéaire Ly =b.

Exemple 14.7 (discrétisation EDP elliptique). Supposons que l'on souhaite approcher les solutions de f (x) = g(x) sur [0, 1] avec des conditions aux limites f(0) = f(1) = 0. On approchera f(x) avec un vecteur y Rn échantillonnage f comme suit :

$$\begin{matrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{matrix} \quad \begin{matrix} f(h) \\ f(2h) \\ \vdots \\ f(nh) \end{matrix}$$

où h = 1/n+1. Nous n'ajoutons pas d'échantillons à x = 0 ou x = 1 car les conditions aux limites y déterminent les valeurs. Nous utiliserons b pour conserver un ensemble analogue de valeurs pour g(x).

Compte tenu de nos conditions aux limites, nous discrétisons $f(x) \overset{1}{\text{en}} h\ 2\ Ly$, où L est donné par :

$$L \equiv \begin{matrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & \end{matrix}$$

Ainsi, notre solution approchée de l'EDP est donnée par $y = h\ 2L\ -1b$.

Tout comme les EDP elliptiques sont les EDP les plus simples à résoudre, leurs discrétisations utilisant des matrices comme dans l'exemple ci-dessus sont les plus simples à résoudre. En fait, généralement la discrétisation d'un opérateur elliptique L est une matrice définie positive et creuse parfaitement adaptée aux techniques de résolution dérivées du chapitre 10.

Exemple 14.8 (Opérateurs elliptiques comme matrices). Considérons la matrice L de l'exemple 14.7. On peut montrer que L est définie négative (et donc le système défini positif $-Ly = -h\ 2b$ peut être résolu en utilisant des gradients conjugués) en remarquant que $-L = DD$ pour la matrice $D \quad R^{(n+1)\times n}$ donnée par :

$$ré = \begin{matrix} 1 & & & & & \\ -1 & 1 & -1 & & & \\ & 1 & & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & & -1 \end{matrix}$$

Cette matrice n'est rien de plus que la dérivée première aux différences finies, donc cette observation est parallèle au fait que $d\ 2\ f/dx2 = d/dx(d\ f/dx)$. Ainsi, $x\ Lx = -x\ DDx = -Dx \le 0$, montrant que L est semi-défini négatif. Il est facile de voir $Dx = 0$ exactement quand $x = 0$, complétant la preuve que L est en fait définie négative.

14.5.2 Résolution d'équations paraboliques et hyperboliques

Les équations paraboliques et hyperboliques introduisent généralement une variable temporelle dans la formulation, elle aussi différenciée mais potentiellement d'ordre inférieur. Étant donné que les solutions aux équations paraboliques admettent de nombreuses propriétés de stabilité, les techniques numériques pour traiter cette variable de temps sont souvent stables et bien conditionnées; en revanche, il faut faire plus attention pour traiter le comportement hyperbolique et empêcher l'amortissement du mouvement au fil du temps.

Méthodes semi-discrètes L'approche la plus directe pour résoudre des équations simples dépendant du temps consiste probablement à utiliser une méthode semi-discrète. Ici, nous discrétisons le domaine mais pas la variable temporelle, conduisant à une ODE qui peut être résolue en utilisant les méthodes du chapitre 13.

Exemple 14.9 (Équation de chaleur semi-discrète). Considérons l'équation de la chaleur à une variable, donnée par $ft = fxx$, où $f(t; x)$ représente la chaleur à la position x et au temps t. En tant que données de limite, l'utilisateur fournit une

fonction f0(x) telle que f(0; x) ≡ f0(x); nous attachons également la frontière x {0, 1} à un réfrigérateur et appliquons ainsi f(t; 0) = f(t; 1) = 0.

Supposons que nous discrétisons la variable x en définissant :

$$f_1(t) \equiv f(h; t)$$
$$f_2(t) \equiv f(2h; t)$$
$$\vdots \qquad \vdots$$
$$f_n(t) \equiv f(nh; t),$$

où, comme dans l'exemple 14.7, nous prenons h = 1/n+1 et omettons les échantillons en x {0, 1} puisqu'ils sont fournis par les conditions aux limites.

En combinant ces fi , nous pouvons définir f(t) : R → Rn comme étant la version semi-discrète de f où nous avons échantillonné dans l'espace mais pas dans le temps. Par notre construction, l'approximation PDE semi-discrète est l'ODE donnée par f (t) = L f(t).

L'exemple précédent montre une instance d'un modèle très général pour les équations paraboliques. Lorsque nous simulons des phénomènes continus comme la chaleur se déplaçant à travers un domaine ou des produits chimiques diffusant à travers une membrane, il y a généralement une variable temporelle puis plusieurs variables spatiales qui se différencient de manière elliptique. Lorsque nous discrétisons ce système de manière semi-discrète, nous pouvons alors utiliser des stratégies d'intégration ODE pour leur solution. En effet, de la même manière que la matrice utilisée pour résoudre une équation linéaire elliptique comme au §14.5.1 est généralement définie positive ou négative, lorsqu'on écrit une EDP parabolique semi-discrète f = L f , la matrice L est généralement définie négative . Cette observationimplique que f résolvant cet ODE continu est inconditionnellement stable, puisque les valeurs propres négatives sont amorties dans le temps.

Comme indiqué dans le chapitre précédent, nous avons plusieurs choix pour résoudre l'ODE en temps résultant d'une discrétisation spatiale. Si les pas de temps sont petits et limités, des méthodes explicites peuvent être acceptables. Les solveurs implicites sont souvent appliqués à la résolution d'EDP paraboliques ; le comportement diffusif d'Euler implicite peut générer des imprécisions mais, d'un point de vue comportemental, il semble similaire à la diffusion fournie par l'équation de la chaleur et peut être acceptable même avec des pas de temps assez grands. Les EDP hyperboliques peuvent nécessiter des étapes implicites pour la stabilité, mais des intégrateurs avancés tels que les « intégrateurs symplectiques » peuvent empêcher le lissage excessif causé par ces types d'étapes.

Une approche contrastée consiste à écrire les solutions de systèmes semi-discrets f = L f en termes de vecteurs propres de L. Supposons v1, . . . ,vn sont des vecteurs propres de L de valeurs propres λ1, . . . , λn et que l'on sait f(0) = c1v1 + · · · + cnvn. Rappelons ensuite que la solution de f = L f est donnée par :

$$f(t) = \sum_{i} c_i v_i e^{\lambda_i t}$$

Cette formule n'est pas nouvelle au-delà du §5.1.2, que nous avons introduit lors de notre discussion sur les vecteurs propres et les valeurs propres. Les vecteurs propres de L, cependant, peuvent avoir une signification physique dans le cas d'une EDP semi-discrète, comme dans l'exemple 14.5, qui a montré que les vecteurs propres des Laplaciens L correspondent à différentes vibrations résonnantes du domaine. Ainsi, cette approche par vecteurs propres peut être appliquée pour développer, par exemple, des «approximations basse fréquence» des données de valeur initiale en tronquant la somme ci-dessus sur i, avec l'avantage que la dépendance de t est connue exactement sans pas de temps.

Exemple 14.10 (Fonctions propres du Laplacien). La figure NUMBER montre les vecteurs propres de la matrice L de l'exemple 14.7. Les vecteurs propres à faibles valeurs propres correspondent à des fonctions basse fréquence sur [0, 1] avec des valeurs fixées aux extrémités et peuvent être de bonnes approximations de f(x) lorsqu'il est relativement lisse.

Méthodes entièrement discrètes Alternativement, nous pourrions traiter les variables d'espace et de temps de manière plus démocratique et les discrétiser toutes les deux simultanément. Cette stratégie donne un système d'équations à résoudre plus comme §14.5.1. Cette méthode est facile à formuler en mettant en parallèle le cas elliptique, mais les systèmes linéaires d'équations résultants peuvent être importants si la dépendance entre les pas de temps a une portée globale.

Exemple 14.11 (Diffusion de chaleur entièrement discrète). Explicite, implicite, Crank-Nicolson. Non couvert dans CS 205A.

Il est important de noter qu'au final, même les méthodes semi-discrètes peuvent être considérées comme totalement discrètes dans la mesure où la méthode ODE à pas de temps discrétise toujours la variable t ; la différence concerne principalement la classification de la manière dont les méthodes ont été dérivées. Un avantage des techniques semi-discrètes, cependant, est qu'elles peuvent ajuster le pas de temps pour t en fonction de l'itération actuelle, par exemple si les objets se déplacent rapidement dans une simulation physique, il peut être judicieux de prendre plus de pas de temps et de résoudre ce mouvement. Certaines méthodes ajustent même la discrétisation du domaine des valeurs x au cas où plus de résolution serait nécessaire à proximité de discontinuités locales ou d'autres artefacts.

14.6 Méthode des éléments finis

Non couvert dans 205A.

14.7 Exemples pratiques

Au lieu d'un traitement rigoureux de toutes les techniques d'EDP couramment utilisées, nous fournissons dans cette section des exemples d'où elles apparaissent dans la pratique en informatique.

14.7.1 Traitement d'image de domaine de gradient

14.7.2 Filtrage préservant les contours

14.7.3 Fluides basés sur le réseau

14.8 À faire

- En savoir plus sur l'existence/l'unicité
- Conditions CFL
- Théorème d'équivalence laxiste
- Cohérence, stabilité et amis

14.9 Problèmes

- Show 1d Laplacian peut être factorisé comme DD pour la première matrice de dérivée D
- Résoudre les EDP du premier ordre

Remerciements

[La discussion va ici]

Je dois un grand merci aux étudiants du CS 205A de Stanford, automne 2013 pour avoir détecté de nombreuses fautes de frappe et erreurs dans le développement de ce livre. Voici une liste sans aucun doute incomplète des étudiants qui ont contribué à cet effort : Tao Du, Lennart Jansson, Miles Johnson, Luke Knepper, Minjae Lee, Nisha Masharani, John Reyna, William Song, Ben-Han Sung, Martina Troesch, Ozhan Turgut, Patrick Ward, Joongyeub Yeo et Yang Zhao.

Remerciements particuliers à Jan Heiland et Tao Du pour avoir aidé à clarifier la dérivation du rythme de l'algorithme BFGS.

[Plus de discussion ici]