# Notes de cours VISI601, L3 CMI Informatique
# Algorithmique numérique

Jacques-Olivier Lachaud
LAMA, Université de Savoie
https://www.lama.univ-savoie.fr/wiki
(suivre VISI601)

4 mars 2024

# Préambule

Ce document donne quelques éléments pour un cours introductif à l'algorithmique numérique. Il cherche à montrer les méthodes de résolution standards des problèmes que l'on trouve en calcul scientifique, et qui provienne le plus souvent de problèmes concrets en physique, mécanique, biologie, mais aussi de problèmes classiques en informatique, en imagerie, géométrie, réseaux, etc. Ce cours suppose comme prérequis une certaine familiarité avec l'analyse (fonctions, dérivabilité, intégration), avec l'algèbre linéaire (vecteurs, matrices), avec la programmation Python et un peu de complexité algorithmique (notations $O$, $\Theta$, $\Omega$).

Pour les travaux pratiques, nous utiliserons le langage Python, ainsi que les modules scientifiques Python (`scipy.org`, scipy, matplotlib). Utilisez la commande suivante (avec `python3` installé sur votre machine).

```
python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose
```

# Plan

1. Introduction : du problème mathématique à la résolution numérique
2. Fondements du calcul scientifique
3. Algèbre linéaire
4. Fonctions : approximation, intégration, interpolation

# 1 Introduction : du problème mathématique à la résolution numérique

Pour citer Quarteroni, Sacco et Saleri [QSS10], "le calcul scientifique est une discipline qui consiste à développer, analyser et appliquer des méthodes relevant de domaines mathématiques aussi variés que l'analyse, l'algèbre linéaire, la géométrie, la théorie de l'approximation, les équations fonctionnelles, l'optimisation ou le calcul différentiel. Les méthodes numériques trouvent des applications naturelles dans de nombreux problèmes posés par la physique, les sciences biologiques, les sciences de l'ingénieur, l'économie, et la finance", mais aussi l'informatique elle-même, comme en imagerie, en traitement des données géométriques, ou dans l'analyse des réseaux sociaux.

## 1.1 Modélisation d'un problème concret

Dans ce cours, nous ne nous intéresserons pas aux méthodes pour traduire un problème concret en un modèle mathématique, domaine appelé *modélisation mathématique*. Il n'y a pas de méthode générale, et c'est très dépendant du domaine visé. Le plus souvent, nous étudierons un problème déjà formalisé sous forme mathématique, et nous montrerons quels outils mathématiques peuvent aider à sa résolution, mais aussi quels algorithmes seront nécessaires pour trouver des solutions numériques, qui sont proches de la solution réelle. De plus, l'efficacité et la stabilité des méthodes numériques seront aussi des critères importants. On voit donc que l'on se situe à la croisée des mathématiques et de l'informatique.

**Remarque 1** *Certains problèmes ont des solutions analytiques (e.g. trouver les racines d'un polynôme du second degré). C'est-à-dire que l'on est capable de décrire la ou les solutions avec des formules mathématiques. Ces formules mathématiques impliquent ensuite des calculs sur ordinateurs comme application numérique. Les formules simples introduisent peu d'erreurs numériques, mais des formules plus complexes peuvent en engendrer.*

*La plupart des problèmes n'ont pas de solutions analytiques. Néanmoins, il faut souvent chercher le bon outil d'analyse ou la bonne méthode d'approximation pour résoudre le problème même de façon purement numérique. Analyse mathématique et résolution numérique sont intimement liés.*

## 1.2 Un mot sur les notations

On manipulera beaucoup des fonctions $f$ de $\Omega \subset \mathbb{R}$ dans $\mathbb{R}$. Dans ce cas sa dérivée par rapport à sa seule variable $x$ (mettons) est souvent notée $f'(x)$.

On manipulera aussi des fonctions du plan $g(x, y)$ avec $(x, y) \in \Omega \subset \mathbb{R}^2$. Dans ce cas là, les deux dérivées partielles de $g$ en un point $(x_0, y_0)$ seront notées :

$$\frac{\partial g}{\partial x}(x_0, y_0) \text{ ou } \partial_x g(x_0, y_0), \quad \frac{\partial g}{\partial y}(x_0, y_0) \text{ ou } \partial_y g(x_0, y_0),$$

Si on désigne uniquement les fonctions dérivées partielles, on omettra le point d'application $(x_0, y_0)$.

On utilisera aussi parfois des fonctions $u(x, y, t)$ qui dépendent à la fois de l'espace et du temps $t$. La dérivée partielle par rapport au temps s'écrira naturellement : $\frac{\partial u}{\partial t}$ ou $\partial_t u$.

Les opérateurs gradient $\nabla\cdot$, laplacien $\Delta\cdot$, divergence $\nabla \cdot \cdot$ ou rotationnel $\nabla \times \cdot$, ne sont liés qu'aux dérivées partielles en espace des fonctions (la dérivée en temps n'intervient pas).

Ainsi, le laplacien $\Delta u$ de $u$ peut s'écrire en coordonnées cartésiennes $\Delta u = \frac{\partial^2 u}{\partial x \partial x} + \frac{\partial^2 u}{\partial y \partial y}$.

Le gradient d'une fonction $f$ de $\mathbb{R}^2$ dans $\mathbb{R}$, s'écrit comme le vecteur $[\partial_x f, \partial_y f]^\intercal$.

## 1.3 Quelques exemples de problèmes concrets

### 1.3.1 Une équation polynomiale implicite

On cherche l'ensemble des points $(x, y)$ du plan tel que

$$x^2 + y^2 - r^2 = 0, \tag{1}$$

avec $r \in \mathbb{R}, r \geq 0$. Evidemment on reconnait le disque de centre $(0, 0)$ et rayon $r$.

Notez qu'on peut écrire ce problème sous une forme plus générale. On cherche les $\mathbf{x} \in \mathbb{R}^n$ tels que $\|\mathbf{x}\|^2 = r^2$. On préférera souvent la forme suivante :

$$S := \{\mathbf{x} \in \mathbb{R}^n, f(\mathbf{x}) = 0\}, \quad \text{with} \quad f(\mathbf{x}) = \|\mathbf{x}\| - r^2. \tag{2}$$

Notez en fait que $S$ est aussi la solution de $f^2(\mathbf{x}) = 0$ et on voit qu'il n'y a pas unicité de la formulation.

**Résolution analytique.** Telle quelle, (2) ne nous permet pas d'énumérer (et de façon de plus en précise) les points de $S$. Une façon est de l'écrire sous une forme $y = \pm\sqrt{r^2 - x^2}$, et pour $x \in [-1, 1]$, nous obtenons des points sur $S$. Néanmoins, la distribution des points n'est pas uniforme, et cette solution ne marche que dans le plan.

Une autre façon (aussi dans le plan) est de remarquer que $S$ doit être une courbe dans le plan et de proposer la "bonne" paramétrisation de cette courbe. On remarque en effet que $\forall t, (r\cos t, r\sin t) \in S$, ce qui nous permet de générer facilement des points répartis uniformément sur $S$. Comme cette fonction est périodique de période $2\pi$, choisir les $t_k = \frac{k2\pi}{n}$, $k \in \{0, \ldots, n-1\}$ donnent les points $(x_k, y_k) = (r\cos t_k, r\sin t_k)$ uniformément répartis sur $S$.

**Résolution numérique** L'approche précédente est très spécifique au cercle. On peut penser qu'il existe des méthodes plus génériques pour trouver les solutions à un problème de type $f(\mathbf{x}) = 0$. Voilà une façon de faire avec Python.

```
%matplotlib inline
import numpy as np
import matplotlib as mp
import matplotlib.pyplot as plt

def f(x,y):
    return x**2+y**2-5**2

# generate 2 2d grids for the x & y bounds
```

```
dx, dy = 0.5, 0.5
y, x = np.mgrid[slice(-7, 7 + dy, dy), slice(-7, 7 + dx, dx)]
z = f(x,y)
# Display
ax = plt.subplot()
ax.imshow(z)
ax.contour(z,[0],colors='red')
```

En réalité, cette méthode parait fonctionner bien, mais si vous essayez d'afficher les zeros de la fonction suivante, vous verrez que l'algorithme `contour` échoue complètement.

```
def f2(x,y):
    return f(x,y)*f(x,y)
```

Quelques questions :

- Pourquoi la résolution numérique de $f(\mathbf{x}) = 0$ et $f^2(\mathbf{x}) = 0$ est très différente ?
- Si vous supposez que $f$ change de signe autour de 0, comment peut-on décider si $f$ a un zéro entre deux points d'une grille ?
- En déduire un algorithme pour tracer les contours dans une grille.

### 1.3.2 L'équation de la chaleur

En mathématiques et en physique théorique, l'équation de la chaleur est une équation aux dérivées partielles parabolique, pour décrire le phénomène physique de conduction thermique, introduite initialement en 1807 par Joseph Fourier.

Si on note $D$ le coefficient de diffusion thermique (supposé constant), alors la température $T$ se diffuse dans un domaine fixe $\Omega$ en suivant l'équation :

$$\frac{\partial T}{\partial t} = D\Delta T, \tag{3}$$

ou plus précisément

$$\forall t \geq 0, \forall \mathbf{x} \in \Omega, \frac{\partial T}{\partial t}(t, \mathbf{x}) = D\Delta T(t, \mathbf{x}), \tag{4}$$

avec $\Delta\cdot$ l'opération Laplacien (la somme des dérivées secondes de la fonction).

Il faut ensuite fermer le système :

1. en précisant une condition initiale, $\forall \mathbf{x} \in \Omega, T(0, \mathbf{x}) = T_{\text{init}}(\mathbf{x})$,
2. en mettant des conditions aux limites sur le bord $\partial\Omega$ du domaine (de normale extérieure $\mathbf{n}$, par exemple
   - condition de Dirichlet : $T(\mathbf{x}, t) = T_{\text{bord}}(\mathbf{x}, t)$,
   - condition de Neumann : $\frac{\partial T}{\partial \mathbf{n}}(\mathbf{x}, t) = \mathbf{n}(\mathbf{x}) \cdot \nabla T(\mathbf{x}, t) = f(\mathbf{x}, t)$, $f$ donné.

**Résolution analytique.** Joseph Fourier a proposé une méthode qui a ensuite trouvé une quantité considérable d'autres applications : c'est les fameuses séries de Fourier (dans le cas d'un domaine mono-dimensionnel) et plus généralement la transformée de Fourier.

**Résolution numérique.** On peut approcher ce type de problèmes par des méthodes de type différences finies ou éléments finis.

### 1.3.3 Les équations proie-prédateur

En mathématiques, les équations de prédation de Lotka-Volterra, que l'on désigne aussi sous le terme de "modèle proie-prédateur", sont un couple d'équations différentielles non-linéaires du premier ordre, et sont couramment utilisées pour décrire la dynamique de systèmes biologiques dans lesquels un prédateur et sa proie interagissent. Elles ont été proposées indépendamment par Alfred James Lotka en 1925 et Vito Volterra en 1926.

Soient

- $t$ le temps,
- $x(t)$ l'effectif de proies en fonction du temps, $x(0) = x_0$,
- $y(t)$ l'effectif de prédateurs en fonction du temps, $y(0) = y_0$,

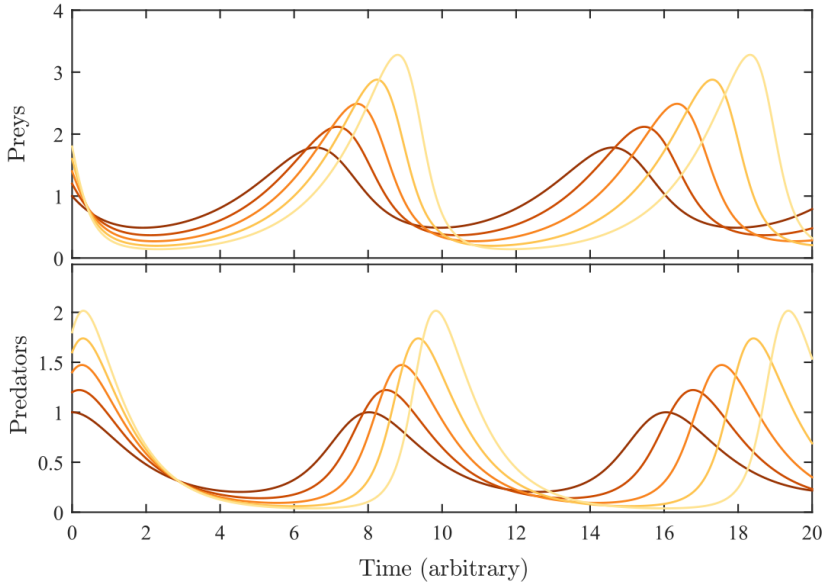


FIGURE 1 – Résolution numérique du système d'équation proie-prédateur (5) de Lotka-Volterra. Paramètres $\alpha = 2/3$, $\beta = 4/3$, $\gamma = 1$, $\delta = 1$, conditions initiales variant de 0.9 à 1.8 par pas de 0.3. Source : Brice2000, Wikipedia

On note que les dérivées $\frac{\mathrm{d}x(t)}{\mathrm{d}t}$ et $\frac{\mathrm{d}y(t)}{\mathrm{d}t}$ représentent l'accroissement de population en fonction du temps. Les paramètres suivants caractérisent les interactions entre les deux espèces :

- $\alpha$ : taux de reproduction intrinsèque des proies (constant, indépendant du nombre de prédateurs) ;
- $\beta$ : taux de mortalité des proies dû aux prédateurs rencontrés ;
- $\delta$ : taux de reproduction des prédateurs en fonction des proies rencontrées et mangées ;
- $\gamma$ : taux de mortalité intrinsèque des prédateurs (constant, indépendant du nombre de proies).

Les équations s'écrivent alors :

$$\begin{cases} \frac{\mathrm{d}x(t)}{\mathrm{d}t} = x(t)(\alpha - \beta y(t)) \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} = y(t)(\delta x(t) - \gamma) \end{cases} \tag{5}$$

**Résolution analytique.** Il n'y a pas d'expression simple des effectifs $x(t)$ et $y(t)$ en fonction du temps. On peut montrer que la fonction $F : (x, y) \mapsto \beta y + \delta x - \alpha \ln y - \gamma \ln x$ est alors une intégrale première du mouvement : $t \mapsto F(x(t), y(t))$ est constante.

Les solutions maximales sont périodiques, et leur trajectoire est fermée bornée.

**Résolution numérique.** Typiquement, on approcherait les dérivées avec des différences finies. Un pas de temps suffisamment petit permet alors d'avoir des solutions approchées raisonnables. Voir la Figure 1 pour une approximation numérique à l'aide de la méthode de Runge-Kutta d'ordre 4.

### 1.3.4 L'intersection de deux droites

Soient $a_{11}x + a_{12}y = b_1$ et $a_{21}x + a_{22}y = b_2$ deux droites dans le plan. On cherche si ces deux droites ont au moins un point commun, et si oui, on veut au moins une solution.

**Résolution analytique.** Le problème précédent se réduit à un problème d'algèbre linéaire. On écrit le système précédent sous forme de matrices/vecteurs :

$$\mathbf{A} := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \mathbf{b} := \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \mathbf{x} := \begin{bmatrix} x \\ y \end{bmatrix} \qquad \mathbf{Ax} = \mathbf{b}. \tag{6}$$

On peut ensuite calculer le déterminant ou le rang de la matrice pour savoir s'il y a une solution, et ensuite inverser la matrice pour trouver la solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

**Résolution numérique.** Dans le cas général des systèmes linéaires, il y a plusieurs façons de faire. Ici on utiliserait l'algèbre linéaire et probablement les règles de Cramer efficaces pour des systèmes de taille $\leq 4$. Sinon on utiliserait plutôt un algorithme de type Gauss-Jordan ou des variantes pour des matrices plus spécifiques.

Attention, un système linéaire peut être très instable (analytiquement) et peut aussi présenté des problèmes numériques de résolution.

### 1.3.5 Débruitage d'image par formulation variationnelle

Si $g$ est une image (une fonction de $\Omega$ dans $\mathbb{R}$) qui a été dégradée (par exemple lors de son acquisition, de sa numérisation). On cherche à reconstruire une fonction $u$ plus lisse et pas trop loin de la donnée $g$. Une façon de faire est de modéliser le problème sous forme variationnelle. On associe una valeur $E$ (souvent appelée *énergie*) à $u$ (définie dans un bon espace) ainsi :

$$E(u) := \int_\Omega \|u(\mathbf{x}) - g(\mathbf{x})\|^2 d\mathbf{x} + \lambda \int_\Omega \|\nabla u(\mathbf{x})\|^p d\mathbf{x}, \tag{7}$$

avec $\lambda > 0$ une constante dépendant de l'intensité de la dégradation de l'image (plus elle est forte, plus $u$ sera lisse), et $p \geq 1$ indique le degré de régularité souhaité. Lorsque $p = 1$, on parle de modèle Variation Totale : l'image reconstruite $u$ a tendance a être lisse par morceaux. Si $p > 1$ (généralement $p = 2$) l'image reconstruite est lisse partout.

L'objectif est alors de trouver la fonction $u^*$ qui *minimise* $E$. Mathématiquement, il faut définir proprement l'espace des fonctions où on cherche la solution. Pour $p > 1$, les fonctions continues et dérivables dans le domaine seraient une possibilité. Pour $p = 1$, il faut des espaces de fonctions plus complexes.

**Résolution analytique.** Il n'y en a pas toujours. On transformerait le problème de minimisation précédent en système d'équations aux dérivées partielles, par exemple via les équations d'Euler-Lagrange. Pour $p = 2$, cela se ramène à une diffusion $u - g - \lambda\Delta u = 0$.

**Résolution numérique.** Pour $p > 1$, on utiliserait les équations d'Euler-Lagrange discrétisées par éléments finis. On pourrait ainsi linéariser l'équation précédente ($\mathbf{I}$ est la matrice identité et $\mathbf{A}$ la matrice du Laplacien par différences finies) et résoudre

$$(\mathbf{I} - \lambda\mathbf{A})\mathbf{u} = \mathbf{g}. \tag{8}$$

Une autre façon est de chercher un point fixe de la fonction $f(u) = g + \lambda\Delta u$.

Pour $p = 1$ on utiliserait plutôt des algorithmes d'optimisation type point intérieur.

### 1.3.6 Résoudre des équations aux dérivées partielles

Le calcul scientifique est très utilisé pour résoudre les équations aux dérivées partielles, omniprésentes en science. La fonction $u$ est spatiale ou spatio-temporelle dans les équations suivantes, la fonction $\vec{v}$ est un champ de vecteurs.

- équation de la chaleur : $\partial_t u = \alpha\Delta u$
- progation d'une onde : $\partial^2_{tt} u = \Delta u$
- équation de Laplace : $\Delta u = 0$, équation de Poisson : $\Delta u = f$
- flots dans les fluides :
  - flot de Darcy : $\vec{v} + \frac{k}{\mu}\nabla p = 0, \nabla \cdot \vec{v} = 0$
  - fluide incompressible (Euler) : $\partial_t \vec{v} + \vec{v} \cdot \nabla\vec{v} = -\frac{\nabla p}{\rho} + g$
  - fluide compressible (Navier-Stokes) : $\rho(\partial_t \vec{v} + \vec{v} \cdot \nabla\vec{v}) = -\nabla p + \nabla \cdot \vec{v} + f$
- élasticité :
  - torsion d'une barre (Euler-Bernouilli) : $\rho A\partial^2_{tt} u + \partial^2_{xx}[EI\partial^2_{xx} u] = p(x,t)$
  - vibration d'une plaque mince (Kirchhoff) : $\rho\partial^2_{tt} u + D\Delta\Delta u = q(x,y,t)$

— loi de l'électromagnétisme (Maxwell)

$$\nabla \cdot \vec{B} = 0, \quad \nabla \cdot \vec{E} = \rho,$$
$$\nabla \times \vec{E} = -\partial_t \vec{B}, \quad \nabla \times \vec{B} = \vec{j} + \partial_t \vec{E}.$$

Toutes ces équations ont des méthodes numériques dédiées. Il n'y a pas de méthode générale pour les résoudre.

### 1.3.7 Recherche de formes optimales

En géométrie, on cherche souvent à savoir quelles formes vont optimiser telle ou telle caractéristique. Ces problèmes se mettent souvent sous forme variationnelle. Numériquement on cherchera à trouver la meilleure forme par des méthodes itératives, en général après linéariser le problème et ramener sous une forme de système linéaire.

Exemples :

— On peut chercher la forme qui minimise son aire à volume constant. Si $\mathrm{Aire}(S) := \int_S dA$, alors on cherche à résoudre

$$X^* := \arg \min_{X \subset \mathbb{R}^3} \mathrm{Aire}(\partial X) \quad \text{avec} \quad \int_X dV = 1. \tag{9}$$

Il se trouve que l'on sait que c'est la boule qui minimise cette énergie. On en déduit d'ailleurs l'inégalité isopérimétrique.

— Problème du nid d'abeille. On cherche dans un domaine (rectangulaire) de volume $n$ entier le découpage en $n$ objets connexes de volume 1 qui minimise l'aire entre les objets. On regarde ensuite à quoi tend cet assemblage lorsque le domaine remplit tout le plan.
En 2D, il s'agit du problème du nid d'abeille (honeycomb), résolu analytiquement en 1999 par T. Hales. On se doutait que le tuilage hexagonal était le meilleur, c'est le cas.
En 3D, Kelvin a conjecturé en 1887 que la solution est un pavage formé d'octaèdres tronquées avec des faces légèrement non planes. Cela a été invalidé en 1994 par Weire et Phellan, avec une structure hétérogène à 8 polyèdres (6 d'une même sorte, 2 d'une autre). On ne sait pas si c'est la meilleure à ce jour.

# 2 Le calcul scientifique

Ces quelques exemples concrets montrent qu'il n'y a pas de méthodes génériques pour aborder tous ces problèmes. On dispose juste de plein d'outils pour aborder ces problèmes. Notons de suite l'importance de la résolution des systèmes linéaires, des approximations des dérivées de fonctions, des équations de type $f(x) = 0$, et des problèmes de minimisations.

On se donne maintenant quelques définitions qui vont permettre de quantifier la difficulté des problèmes, la qualité et/ou la convergence des solutions numériques.

## 2.1 Problèmes bien posés et problèmes mal posés

Considérons le problème suivant. Soient $d$ des données et on cherche $x$ tel que :

$$F(x, d) = 0, \tag{10}$$

où $F$ est une relation fonctionnelle entre les inconnues $x$ et les données $d$. Attention $F$ peut être à valeur scalaire (une équation) ou vectorielle (système d'équations). Les variables $x$ et $d$ peuvent être des scalaires, des vecteurs ou des fonctions.

(10) est un problème *direct* si $F$ et $d$ sont les données et $x$ les inconnues. (10) est un problème *inverse* si $F$ et $x$ sont données et on recherche des données $d$ qui vérifie $F$. Enfin (10) est un problème d'*identification* si $x$ et $d$ sont donnés, mais que $F$ n'est pas connu.

Exemples :

- On note que l'intersection de deux droites est un problème direct qui rentre typiquement dans le cadre précédent, avec $F(x, (\mathbf{A}, \mathbf{b})) = 0$.
- Comme exemple notable de problème inverse, on peut citer le scanner X (ou tomodensitométrie par rayon X). On connait la relation (fonction d'absorption) $F_1$ qui fabrique une image $x_1$ à partir de rayons X envoyés d'une direction $\theta_1$ dans des données $d$ (le patient). En notant $F(d, x) = (F_1(d, x_1), \ldots, F_n(d, x_n))$, on voit qu'il s'agit d'un problème inverse.
- Les problèmes d'identification occupent beaucoup le physicien, le mécanicien ou le biologiste, puisqu'il s'agit de trouver des lois qui relient des quantités observés.

Le problème (10) est dit *bien posé* si la solution $x$ *existe*, est *unique* et dépend *continûment des données*. On parle aussi de problème *bien posé au sens de Hadamard*, ou de problème *stable*.

On ne peut espérer résoudre numériquement des problèmes intrinsèquement mal posés. Si un problème est mal posé, on le *régularise* pour le rendre bien posé.

Exemples :

- Pour la plupart des données, le problème de l'intersection de 2 droites est bien posé, mais lorsque les pentes sont trop proches, le problème devient numériquement de plus en plus mal posé. Lorque les 2 droites sont parallèles ou confondues, le problème devient mal posé.
- Le problème de Dirichlet pour l'équation de Laplace et l'équation de la chaleur avec spécification de conditions initiales sont des formulations bien posées.
- Déterminer le nombre de racines réelles des polynômes est typiquement mal posé en général. Ainsi la famille de polynômes $P_a(x) = x^2 + ax + 1$ a 0 racines pour $a < 2$, 1 racine pour $a = 2$, 2 racines pour $a > 2$. Une petite perturbation des données $a$ autour de 2 entraîne une grande perturbation du résultat (0,1, ou 2).

La dépendance continue par rapport aux données signifie que de petites perturbations $\delta d$ admissibles sur les données $d$ impliquent de "petites" perturbations $\delta x$ sur la solution $x$, avec

$$F(x + \delta x, d + \delta d) = 0. \tag{11}$$

La dépendance continue aux données s'écrit ainsi. Il existe deux constantes positives $\eta_0$ et $K_0$ dépendant possiblement des données tels que :

$$\text{si } \|\delta d\| \leq \eta_0 \text{ alors } \|\delta x\| \leq K_0 \|\delta d\|.$$

NB : Les normes utilisées peuvent être différentes.

## 2.2 Conditionnement

Si $D$ est l'ensemble des données possibles, on peut associer à un problème deux quantités qui traduisent si un problème est plus ou moins bien posé.

Le *conditionnement relatif* $K_{rel}(d)$ et le *conditionnement absolu* $K_{abs}(d)$ sont donnés par :

$$K_{rel}(d) = \sup \left\{ \frac{\|\delta x\|/\|x\|}{\|\delta d\|/\|d\|}, \delta d \neq 0, d + \delta d \in D \right\} \tag{12}$$

$$K_{abs}(d) = \sup \left\{ \frac{\|\delta x\|}{\|\delta d\|}, \delta d \neq 0, d + \delta d \in D \right\}. \tag{13}$$

Le conditionnement absolu est utilisé lorsque $d = 0$ ou $x = 0$.

On dit que le problème (10) est *mal conditionné* si $K_{rel}(d)$ est "grand" pour toute donnée admissible $d$ (avec "petit" et "grand" qui dépendent du contexte). Un bon ou mauvais conditionnement est indépendant de la méthode numérique choisie. En revanche le bon ou le mauvais conditionnement d'un problème peut avoir un gros impact sur la qualité de solution trouvée par une méthode numérique.

Attention, mal conditionné n'implique pas forcément mal posé, car une transformation du problème peut parfois conduire à un problème équivalent bien posé.

## 2.3 Résolvante

Si notre problème admet une unique solution, alors il existe une application appelée *résolvante*, notée $G$, de l'ensemble des données vers l'ensemble des solutions, telle que

$$x = G(d), \quad \text{autrement dit} \quad F(G(d), d) = 0. \tag{14}$$

Cela vient du théorème des fonctions implicites. Si on suppose $G$ différentiable en $d$ et on note $G'(d)$ sa dérivée par rapport à $d$ (sa matrice jacobienne évaluée en $d$ si $G$ est de $\mathbb{R}^n$ vers $\mathbb{R}^m$), alors on peut écrire à l'aide d'un développement au premier ordre :

$$G(d + \delta d) - G(d) = G'(d)\delta d + o(\|\delta d\|).$$

On trouve facilement

$$K_{rel}(d) \approx \|G'(d)\| \frac{\|d\|}{\|G(d)\|}, \quad K_{abs}(d) \approx \|G'(d)\|. \tag{15}$$

Le symbole $\|\cdot\|$ désigne la norme matricielle subordonnée à la norme vectorielle choisie.

Exemple : Si on observe un système d'équations linéaires $\mathbf{Ax} = \mathbf{b}$ où $\mathbf{A}$ est une matrice $n \times n$ inversible fixée et $\mathbf{b}$ représente les données $(d)$ et $\mathbf{x}$ les inconnues. Si on ne perturbe que $\mathbf{b}$, alors $d = \mathbf{b}$, $\mathbf{x} = G(d) = \mathbf{A}^{-1}\mathbf{b}$ et $G'(d) = \mathbf{A}^{-1}$. D'où

$$\begin{aligned} K_{rel}(d) &\approx \|\mathbf{A}^{-1}\| \frac{\|\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|} \\ &= \|\mathbf{A}^{-1}\| \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} \\ &\leq \|\mathbf{A}^{-1}\|\|\mathbf{A}\| =: \mathrm{K}(\mathbf{A}), \end{aligned}$$

où $\mathrm{K}(\mathbf{A})$ est le conditionnement de la matrice $\mathbf{A}$ (voir chapitre algèbre linéaire). On verra par exemple que la stabilité d'un système linéaire lorsque $\mathbf{A}$ est symétrique définie positive est le ratio de la plus grande valeur propre sur la plus petite.

## 2.4 Stabilité et convergence des méthodes numériques

Une méthode numérique pour approcher la solution d'un problème bien posé consiste à effectuer une suite de problèmes approchés (avec $n$ un paramètre comme la résolution par exemple).

$$F_n(x_n, d_n) = 0 \quad n \geq 1 \tag{16}$$

On cherche à ce que $x_n \to x$ lorsque $n \to \infty$.

La méthode numérique est *consistante* si $F_n$ tend bien vers $F$ avec la bonne donnée et la bonne solution, i.e.

$$F_n(x, d) = F_n(x, d) - F(x, d) \to 0 \quad \text{lorsque } n \to \infty. \tag{17}$$

Une méthode est *fortement consistante* si $F_n(x, d) = 0$ pour toute valeur de $n$.

On note que pour certaine méthode itérative $F_n$ nécessite des solutions antérieures, donc a une forme (au lieu de (16)) :

$$F_n(x_n, x_{n-1}, \ldots, x_{n-q}, d_n) = 0 \quad n \geq q. \tag{18}$$

La propriété de consistance forte s'écrit alors plutôt $F_n(x, x, \ldots, x, d) = 0$.

Exemples : Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction dont on cherche à approcher une racine simple. La *méthode de Newton* consiste à

- choisir un $x_0$,
- pour $n \geq 1$, $x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$.

On pose $F_n(x_n, x_{n-1}, f) := x_n - x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$. On vérifie que si $\alpha$ est une racine de $f$, i.e. $f(\alpha) = 0$, alors $F_n(\alpha, \alpha, f) = 0$. La *méthode de Newton* est donc fortement consistante.

Exemples : Si maintenant on choisit la méthode $x_n = x_{n-1} - \alpha \frac{f(x_{n-1})}{f'(x_{n-1})}$, où $\alpha$ est un coefficient positif, est-ce que nous avons toujours la consistance (forte ou pas) ?

Exemples : On cherche à trouver le minimum d'une fonction, quelle méthode numérique peut-on envisager ?

On note que la plupart des méthodes numériques qui tronquent une opération faisant intervenir une limite (e.g. une intégrale) ne peuvent pas être fortement consistantes.

Une méthode numérique est *bien posée* ou *stable* s'il existe, pour tout $n$, une unique solution $x_n$ correspondant à une donnée $d_n$ et si $x_n$ varie continûment en fonctions des données.

Similairement à (11), la dépendance continue s'écrit alors sous la forme :

$$F_n(x_n + \delta x_n, d_n + \delta d_n) = 0. \tag{19}$$

On définit alors similairement la dépendance avec deux constantes $\eta_0 = \eta_0(d_n)$ et $K_0 = K_0(d_n)$ :

$$\text{si } \|\delta d_n\| \leq \eta_0 \text{ alors } \|\delta x_n\| \leq K_0 \|\delta d_n\|.$$

On introduit encore de la même façon les conditionnements relatif $K_n(d_n)$ et absolu $K_{abs,n}(d_n)$. On a encore une application $G_n$ appelée *resolvante* (numérique) telle que :

$$x_n = G_n(d_n), \quad \text{autrement dit} \quad F_n(G_n(d_n), d_n) = 0. \tag{20}$$

Exemples :

- L'addition, i.e. l'application $f : \mathbb{R}^2 \to \mathbb{R}$, $f(a,b) = a+b$, est bien conditionnée pour des nombres positifs. En effet, le vecteur gradient s'écrit $f'(a,b) = [1,1]^\intercal$. Avec la norme vectorielle $\|\cdot\|_1$, on montre que le conditionnement relatif $K_{rel}(a,b) \approx (|a|+|b|)/|a+b| = 1$.
- la soustraction de deux nombres presque égaux est quant à elle mal conditionnée par la même formule.

Le but d'une méthode numérique est évidemment d'essayer d'approcher la vraie solution au problème $x$. La suite de solution intermédiaire $x_n$ doit donc converger vers $x$. Ecrit formellement, cela donne :

La méthode numérique (16) est dite *convergente* vers le problème (10) ssi :

$$\begin{aligned} &\forall \epsilon > 0, \exists n_0 = n_0(\epsilon), \exists \delta = \delta(n_0, \epsilon) \text{ tels que} \\ &\forall n \geq n_0, \forall \delta_n : \|\delta_n\| \leq \delta \Rightarrow \|x(d) - x_n(d+\delta_n)\| \leq \epsilon, \end{aligned} \tag{21}$$

où $d$ est une donnée possible du problème (10), $x(d)$ est la solution correspondante et $x_n(d+\delta_n)$ est la solution donnée par la méthode numérique pour $d+\delta_n$.

En fait, pour établir (21), il suffit de vérifier que

$$\|x(d+\delta_n) - x_n(d+\delta_n)\| \leq \frac{\epsilon}{2}.$$

L'*erreur absolue* et l'*erreur relative* sont définies respectivement par $E(x_n) = |x - x_n|$ et $E_{rel}(x_n) = |x - x_n|/|x|$ pour $x \neq 0$. On remplace évidemment par la norme appropriée lorsque $x$ et $x_n$ sont des vecteurs ou autres.

## 2.5 Relations entre stabilité et convergence

Si le problème (10) est bien posé, la *stabilité* (ou bien posé) est une condition *nécessaire* pour que la méthode numérique (16) soit convergente.

Si la méthode numérique (16) est de plus consistante avec le problème (10), alors la stabilité (ou bien posé) est une condition *suffisante* pour que la méthode numérique (16) soit convergente.

Il y a donc équivalence pour une méthode numérique consistante entre les propriétés de stabilité et de convergence (théorème de Lax-Richtmyer).

## 2.6 Erreurs dans les modèles numériques

Premièrement, il existe en général un problème physique, noté $PP$, avec une solution exacte $x_\phi$. Deuxièmement, en plus du modèle mathématique (10) (de solution $x$) et de la méthode numérique (16) (de solutions $x_n$), il y a les solutions effectivement calculées sur ordinateur $\hat{x}_n$.

On définit alors les erreurs suivantes :

- l'*erreur globale* $e := \|\hat{x}_n - x_\phi\|$
- l'*erreur de modélisation* $e_m := \|x - x_\phi\|$, erreurs de mesures + erreurs modèle
- l'*erreur du modèle numérique* $e_c := \|\hat{x}_n - x\|$, erreurs méthode num. + calcul
- l'*erreur de discrétisation* $e_n := \|x_n - x\|$, erreur de la méthode num. (continu vers discret)
- l'*erreur d'arrondie* $e_a := \|\hat{x}_n - x_n\|$, erreur de représentation des nombres (quantification).

On a $e = e_m + e_c$. L'erreur $e_m$ tient compte des erreurs possibles sur les données (mesures physiques) ainsi que les approximations faites dans le modèle.

L'erreur $e_c$ tient compte des *erreurs de troncature* de discrétisation $e_n$ et des erreurs d'arrondi ou d'approximation $e_a$ dues aux calculs sur ordinateur avec des représentations partielles des nombres.

Une bonne méthode numérique est une méthode dont on pourra baisser arbitrairement les erreurs $e_n$ (et parfois $e_a$) en augmentant l'effort de calcul. Evidemment, cette augmentation d'effort sera lié à la complexité des algorithmes mis en oeuvre lors de la résolution numérique du problème. Des algorithmes trop coûteux ne permettront pas la résolution pratique du problème avec la précision voulue.

## 2.7 Représentation des nombres en machine

On n'utilise pas en général le système positionnel de représentation des nombres, c'est-à-dire, dans une base donnée, on écrit tous les chiffres à gauche et à droite de la virgule jusqu'à n'avoir que des chiffres nuls après.

Exemple : 159000000.0, $-0.0000000143$ sont des écritures positionnelles.

On utilise un système à virgule flottante.

Exemple : $1.59 \times 10^6$, $-1.43 \times 10^{-8}$ sont des écritures décimales à virgule flottante. La mantisse décrit les chiffres non nuls, l'exposant donne le décalage.

Les types `float` ou `double` (standard IEC559 par l'IEEE) sont des représentation binaires à virgules flottante. En reprenant un tableau de Wikipedia, voilà les précisions de ces types.

| Précision | Encodage | Signe | Exposant | Mantisse | Valeur d'un nombre | Précision | Chiffres signif. |
|---|---|---|---|---|---|---|---|
| Simple | 32 bits | 1 bit | 8 bits | 23 bits | $(-1)^S \times M \times 2^{(E-127)}$ | 24 bits | environ 7 |
| Double | 64 bits | 1 bit | 11 bits | 52 bits | $(-1)^S \times M \times 2^{(E-1023)}$ | 53 bits | environ 16 |

Comme c'est des nombres binaires, le premier chiffre est toujours 1 (non codé). On a des codages spéciaux de $+0$ et $-0$ ( !), de $+\infty$ et $-\infty$ et des erreurs NaN (not a number).

Exemples : Codons le nombre $x = -47,625$ en flottant 32 bits. D'abord on met le signe à 1 (négatif) et on l'enlève de $x$. Ensuite on code en binaire $47,625$. On a ainsi :

$$\begin{aligned} 47,625 &= 32 + 8 + 4 + 2 + 1 + 0,5 + 0,125 \\ &= 2^5 + 2^3 + 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-3} \\ &= 101111,101 \qquad \text{(en base 2)} \end{aligned}$$

Il faut ensuite l'exprimer sous la forme $1,\ldots \times 2^n$, ce qui donne :

$$101111,101 = 1,01111101 \times 2^5$$

Les bits $\boxed{01111101\,0\cdots0}$ (23 bits au total) représente la mantisse $M$. Enfin, comme on est en précision 32 bits, il y a 8 bits pour l'exposant, et l'exposant 0 correspond à $e = 2^7 - 1 = 127$. L'exposant $E$ est donc $n + e = 5 + 127 = 132 =_2 10000100$. En machine on a donc :

$$x = -47,625 = \underbrace{1}_{\text{signe}}\ \underbrace{10000100}_{\text{exposant E (8 bits)}}\ \underbrace{01111101\,0\cdots0}_{\text{mantisse M (23 bits)}}$$

On note qu'on peut montrer que l'erreur d'arrondi *relative* de l'opération $fl$ d'un nombre réel $x$ quelonque est $E_{rel}(x) = |x - fl(x)|/|x| \leq \frac{1}{2}2^{1-t} =: \mathfrak{u}$ avec $t$ nb de bits de la mantisse. Le nombre $\mathfrak{u}$ est appelé *unité d'arrondi* ou *précision machine*. Il est indépendant du nombre $x$ considéré.

On montre aussi que l'erreur d'arrondi *absolue* $E(x) = |x - fl(x)| = \frac{1}{2}2^{e-t}$ où $e$ est l'exposant courant. L'erreur $\mathfrak{ulp}(x) := 2^{e-t}$ est appelée *unit in the last place*.

On note que les opérations machine $+$ ou $*$ ne sont pas associatives. Les erreurs numériques pourraient se compenser mais elles ont tendance à s'accumuler !

Exemples : Calculez $f(x) = \frac{1}{1-\sqrt{1-x^2}}$ au voisinage de 0 à l'aide de python ou matlab (par exemple testez $0,0001$, $0,00001$, $0,000001$, $0,0000001$, $0,00000001$). Que constatez-vous ? Où se situe le problème ? Quelles seraient les valeurs attendues au voisinage de 0 ?
Proposez une autre écriture de la même fonction, qui retourne des valeurs beaucoup plus précises au voisinage de 0.

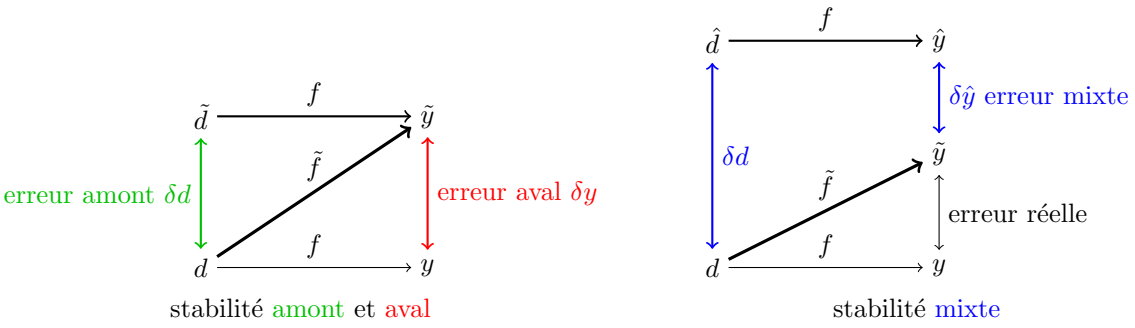


FIGURE 2 – Illustration des erreurs mesurées pour les stabilités amont et mixte.

Exercice : Erreurs d'arrondis
Analyser la stabilité par rapport aux propagation d'arrondis des deux codes suivants pour évaluer $f(x) = (e^x - 1)/x$ quand $|x| \ll 1$.

```
import numpy as np
def f1(x):
    if x == 0:
      return 1
    else:
      return (np.exp(x)-1)/x
```

```
def f2(x):
    y = np.exp(x)
    if y == 0:
      return 1
    else:
      return (y-1)/np.log(y)
```

## 2.8 Stabilités amont, mixte et aval d'un calcul numérique $f$

Voir la Figure 2.

(En anglais, stabilité amont *backward stability*, stabilité aval *forward stability*, stabilité mixte *mixed stability*.)

Soit $d$ les données idéales et $y$ les sorties idéales d'un calcul $f$. En réalité, on ne réalise qu'un calcul numérique approché $\tilde{f}$ et la sortie est $\tilde{y}$. A première vue, nous sommes intéressés par la précision du résultat, donc l'erreur aval $\delta y = \tilde{y} - y$. Une fonction est *stable aval* si et seulement si $\frac{\|\delta y\|}{\|y\|} \leq O(\mathfrak{u})$. En effet, comme les données sont précises à l'arrondi près $\mathfrak{u}$ (car c'est la précision de la fonction $fl(x)$), on espère que la sortie de nos fonctions ait cette précision.

Exemples : La fonction numérique $fl(x)$ qui fabrique le nombre flottant le plus proche de $x$ est stable aval (dans la limite des puissances bien sûr). En effet, on la compare à l'identité $i$ et on obtient :

$$\frac{|fl(d) - i(d)|}{|i(d)|} = \frac{|fl(d) - d|}{|d|} \leq \mathfrak{u}.$$

Malheureusement très peu de calculs sont stables en aval, tout simplement parce qu'on montre que la stabilité aval est aussi directement influencée par le conditionnement du problème, et donc l'erreur peut souvent grandir quasi-arbitrairement.

On cherche donc plutôt si il existe une donnée approchée $\tilde{d}$ telle que $f(\tilde{d}) = \tilde{f}(d)$. Selon Higham, "donne exactement la bonne réponse à presque la bonne question", puisqu'on a

$$\tilde{y} = \tilde{f}(d) = f(\tilde{d}).$$

L'erreur amont est $\tilde{d} - d$. Le calcul numérique est *stable en amont* si on $\frac{\|\tilde{d}-d\|}{\|d\|} \leq O(\mathfrak{u})$. Une autre façon de définir la stabilité amont est de dire que $f^{-1}(\tilde{y})$ est proche de $d$. La stabilité amont nous dit dans quelle mesure on a résolu le bon problème. Lorque la méthode est stable amont, notre solution calculée est aussi proche de la bonne qu'on peut espérer, car son écart à la solution réelle correspond à une donnée qui était très proche de toutes façons.

Parfois cette définition de stabilité est difficile à atteindre, et on cherche à avoir une définition qui mélange erreur amont et erreur aval. Un calcul numérique a une *stabilité mixte* lorsque

$$\frac{\|\hat{d}-d\|}{\|d\|} \leq O(\mathfrak{u}) \quad \Rightarrow \quad \frac{\|\tilde{f}(d)-f(\hat{d})\|}{\|f(\hat{d})\|} \leq O(\mathfrak{u}).$$

Souvent on n'a pas l'exacte bonne réponse (i.e. $f(\hat{d}) \neq f(d)$), ni l'exacte donnée, et la stabilité mixte garantit que si la donnée est proche, le résultat est proche. Selon Higham, "donne presque la bonne réponse à presque la bonne question".

On montre aisément que stabilité amont implique stabilité mixte.

On note que la précision d'une méthode stable en amont (et d'ailleurs aussi d'une méthode stable mixte) est liée au conditionnement relatif :

$$\begin{aligned}
\frac{\|\delta y\|}{\|y\|} &= \frac{\|\tilde{f}(d)-f(d)\|}{\|f(d)\|} \\
&= \frac{\|f(\tilde{d})-f(d)\|}{\|f(d)\|} && (\text{car } \tilde{f}(d)=f(\tilde{d})) \\
&\leq \frac{\|\tilde{d}-d\|\|f'(d)\|+o(\|\tilde{d}-d\|)}{\|f(d)\|} && (\text{Taylor})
\end{aligned}$$

Or

$$K_{rel}(d) = \sup_{\delta d}\left\{\frac{\|\delta y\|/\|y\|}{\|\delta d\|/\|d\|}\right\} = \frac{\|f'(d)\|\|d\|}{\|f(d)\|}$$

D'où

$$\begin{aligned}
\frac{\|\delta y\|}{\|y\|} &\leq \frac{\|\tilde{d}-d\|}{\|d\|}K_{rel}(d) \\
&\leq K_{rel}(d)O(\mathfrak{u}) && (\text{stabilité amont})
\end{aligned}$$

La précision des méthodes stables amont est donc directement liée à la précision machine pour modéliser les données et au conditionnement relatif du problème.

La stabilité amont est beaucoup plus fréquente et simple à démontrer que la stabilité aval. On peut être stable amont et avoir une erreur aval non bornée (par exemple parce que le conditionnement devient infini proche de certaines données).

Exemples :

- La fonction addition $+$ (en flottant $\oplus$) est stable aval pour des valeurs $y$ loin de $-x$. On montre ainsi que $\frac{|x\oplus y-(x+y)|}{|x+y|} \leq 2\frac{|x|+|y|}{|x+y|}\mathfrak{u}$.
  Par exemple si, $y = -x(1+\mu)$, on montre que $\frac{|x|+|y|}{|x+y|} \leq 2$ si et seulement si $\mu \leq -2/3$ ou $\mu \geq 2$. Ce qui montre que pour ces valeurs $(x, y)$, la fonction $\oplus$ est stable aval avec une erreur relative bornée par $4\mathfrak{u}$.
- La fonction soustraction $-$ (en flottant $\ominus$) est stable amont. En effet, soit $f(x_1, x_2) = x_1 - x_2$, alors

$$\begin{aligned}\tilde{f}(x_1, x_2) &= \mathrm{fl}(x_1) \ominus \mathrm{fl}(x_2) \\ &= (x_1(1+\epsilon_1) - x_2(1+\epsilon_2))(1+\epsilon_3) && (\epsilon_i = O(\mathfrak{u})) \\ &= x_1(1+\epsilon_4) - x_2(1+\epsilon_5) \\ &= f(\tilde{x}_1, \tilde{x}_2).\end{aligned}$$

  Attention, autour de 0, l'erreur aval peut néanmoins tendre vers l'infini car le conditionnement devient infini.
- Le produit scalaire $\mathbf{x}^\intercal\mathbf{y}$ est stable amont
- La plupart des opérations d'algèbre linéaire sont stables amont : $\mathbf{Ax}$, $\mathbf{AB}$, $\mathbf{QR}$, $\mathbf{LU}$, SVD
- La fonction addition $+1$ $f(x) = x + 1$ est stable mixte mais pas stable amont !

$$\begin{aligned}\tilde{f}(x) &= \mathrm{fl}(x) \oplus 1 = (x(1+\epsilon_1) + 1)(1+\epsilon_2) \\ f(\tilde{x}) &= \tilde{x} + 1.\end{aligned}$$

  Si $x \approx 0$, $f(\tilde{x}) = 1 \neq 1 + \epsilon_2 = \tilde{f}(x)$.
- La fonction produit externe (ou tensoriel) $\mathbf{x}_1\mathbf{x}_2^\intercal$ est stable mais n'est pas stable amont. En effet, il est peu probable que le rang de la fonction numérique soit 1 et on ne peut trouver de perturbations de $\mathbf{x}_1$ et $\mathbf{x}_2$ qui donne la perturbation de sortie.

## 2.9 Complexité algorithmique

Voir le cours INFO602 et les notes de cours associées. On utilisera les notations $O$, $\Theta$ et $\Omega$ pour des tailles de données tendant vers l'infini.

Une opération de calcul sur des nombres flottants sera appelée *flop*. Parfois, pour être plus précis que la simple complexité, on essaiera de compter précisément le nombre de flops effectués par un calcul.

# 3 Algèbre linéaire numérique

Un système linéaire de $m$ équations à $n$ inconnues $x_j$ est constitué de $m$ relations linéaires $\sum_{j=1}^{n} a_{ij}x_i = b_i$, $i = 1, \ldots, m$. Les $a_{ij}$ sont les coefficients du systèmes et les $b_i$ les composantes du second membre. On écrit plutôt ce système sous forme matricielle :

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{22}$$

Possiblement les coefficients peuvent avoir valeur dans n'importe quel corps, mais on utilisera uniquement les réels ici. Donc $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m\times n}$ et $\mathbf{b} = (b_i) \in \mathbb{R}^m$ et $\mathbf{x} = (x_j) \in \mathbb{R}^n$. Lorsque $m = n$, la matrice $\mathbf{A}$ est carrée et on dit qu'elle est d'*ordre* $n$.

A toute application linéaire $f$ est associée une unique matrice $\mathbf{A}_f$ telle que $f(\mathbf{x}) = \mathbf{A}_f\mathbf{x}$. Inversement n'importe quelle matrice $\mathbf{A}$ définit une application linéaire $g$ telle que $g(\mathbf{x}) := \mathbf{A}\mathbf{x}$.

Il est connu que dans le cas où $n$ est carré, on a existence et unicité de la solution lorsque une des conditions équivalentes est vraies :

1. $\mathbf{A}$ est inversible,
2. $\mathrm{rg}(\mathbf{A}) = n$ (son rang est $n$),
3. le système homogène $\mathbf{A}\mathbf{x} = \mathbf{0}$ admet seulement la solution nulle.

On peut savoir si une matrice est inversible en calculant son déterminant.

## 3.1 Trace et déterminant d'une matrice

Soit $\mathbf{A}$ une matrice d'ordre $n$. La trace de $\mathbf{A}$ est définie comme la somme des éléments de sa diagonale : $\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}$.

Le déterminant de $\mathbf{A}$ est un scalaire obtenu en effectuant tous les produits de coefficients avec un élément par ligne sur toutes les permutations possibles $P$ de taille $n$, modulo le signe de la permutation :

$$\det(\mathbf{A}) = \sum_{\pi\in P} \mathrm{signe}(\pi) a_{1\pi(1)}a_{2\pi(2)}\ldots a_{n\pi(n)}. \tag{23}$$

Il est alors évident que le déterminant change seulement de signe lors d'un échange de lignes ou de colonnes de la matrice.

> Quel est l'interprétation géométrique du déterminant d'une matrice $\mathbf{A}$ vue comme la juxtaposition de $n$ vecteurs colonnes ou lignes ?

On note aussi les propriétés suivantes :

$$\begin{aligned} \det(\mathbf{A}) = \det(\mathbf{A}^\intercal), \quad \det(\mathbf{A}\mathbf{B}) = \det(\mathbf{A})\det(\mathbf{B}), \\ \det(\mathbf{A}^{-1}) = 1/\det(\mathbf{A}), \quad \det(\alpha\mathbf{A}) = \alpha^n\det(\mathbf{A}). \end{aligned}$$

Si jamais deux vecteurs ligne ou colonne ne sont pas indépendants, alors le déterminant est nul. Un échange de ligne ou de colonne inverse le signe du déterminant.

On note $\mathbf{A}_{ij}$ la matrice d'ordre $n-1$ obtenue en enlevant la colonne et la ligne passant par $a_{ij}$. On appelle *mineur* associé à $a_{ij}$ le déterminant de $\mathbf{A}_{ij}$. On appelle *cofacteur de* $a_{ij}$ la quantité $\Delta_{ij} = (-1)^{i+j}\det(\mathbf{A}_{ij})$.

La loi de Laplace donne un moyen de calculer par récurrence le déterminant d'une matrice :

$$\det(\mathbf{A}) = \begin{cases} a_{11} & \text{si } n = 1, \\ \sum_{i=1}^{n} \Delta_{ij}a_{ij} & \text{pour } n > 1. \end{cases} \tag{24}$$

Enfin, on peut calculer l'inverse d'une matrice avec la formule

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}[\Delta_{ij}]^\intercal. \tag{25}$$

Quelques questions :

— Quelle est la complexité du calcul du déterminant avec (24) ?

— Même question pour l'inverse d'une matrice avec (25).

## 3.2 Calcul naïf du déterminant et de la solution à (22)

On pourrait tenter de résoudre le système (22), en vérifiant que le système est inversible via (24), puis en utilisant les formules de Cramer :

$$x_j = \frac{\Delta_j}{\det(\mathbf{A})}, \quad j = 1, \ldots, n, \tag{26}$$

où $\Delta_j$ est le déterminant de la matrice $\mathbf{A}$ où on mit $\mathbf{b}$ à la place de sa $j$-ème colonne.

Cette formule a peu d'utilité pratique car son coût est supérieur à $(n+1)!$ flops.

De plus, le déterminant a peu d'intérêt pratique dans la détermination de l'existence d'une solution à $\mathbf{Ax} = \mathbf{b}$. Par exemple, considérez la matrice d'ordre $n$ diagonale $\mathrm{diag}(a, \ldots, a)$ où $a$ est un réel non nul.

Si $a$ vaut $10^{-4}$ et $n = 100$, le déterminant vaut $10^{-400}$, qui n'est pas représentable sur un double. Le déterminant numérique est alors 0 et le système est déclaré non inversible. Or il est très facile d'inverser $\mathbf{A}$.

Son conditionnement est d'ailleurs de 1.

## 3.3 Exemples de problèmes typiques

On prend les matrices et vecteur suivants :

$$\mathbf{A} = \begin{bmatrix} -\frac{1}{2} & 1 \\ -\frac{1}{2} + d & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 4 \\ 1 & 2 + e \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Si on prend $|d| \ll 1$ alors le système $\mathbf{Ax} = \mathbf{b}$ est *mal conditionné*.

Si on cherche à résoudre ce système en changeant de base, on pourrait chercher à résoudre plutôt $\mathbf{BAx} = \mathbf{Bb}$. Ici la nouvelle base change le vecteur $(1, 0)$ en $(2, 4)$ et le vecteur $(0, 1)$ en $(4, 2 + e)$. En faisant cela, la matrice $\mathbf{BA}$ a pour déterminant $-6ed$. On peut donc rendre le système mieux conditionné en augmentant $e$.

En revanche, la résolution numérique est impactée. Pour des valeurs très innocentes de $d$, un mauvais $e$ peut rendre une méthode PLU très instable numériquement. C'est par exemple le cas pour $d = \frac{3}{2}$ et $|e| \ll 1$. En effet la construction du système triangulaire donne :

$$\begin{bmatrix} 3 & 6 \\ 0 & -e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 6 \\ -e \end{bmatrix}$$

## 3.4 Conditionnement d'une matrice

On définit le *conditionnement d'une matrice* $\mathbf{A}$ d'ordre $n$ (inversible) ainsi :

$$\mathrm{K}(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|, \tag{27}$$

pour une norme matricielle subordonnée. En général, on prend les $p$-normes et on note $K_p(\mathbf{A})$ son conditionnement par rapport à cette norme. Les cas utilisés sont pour $p = 1, 2, \infty$ et les valeurs peuvent être différentes. Il s'agit bien du conditionnement relatif au problème $\mathbf{Ax} = \mathbf{b}$.

En effet, on le calcule en supposant que seul le membre de droite $\mathbf{b}$ est perturbé :

$$\begin{aligned}
K_{rel}(\mathbf{b}) &= \sup\left\{\frac{\|\delta\mathbf{x}\|/\|\mathbf{x}\|}{\|\delta\mathbf{b}\|/\|\mathbf{b}\|}\right\} && \text{(cf (12))}\\
&= \sup\left\{\frac{\|\mathbf{A}^{-1}\delta\mathbf{b}\|/\|\mathbf{x}\|}{\|\delta\mathbf{b}\|/\|\mathbf{b}\|}\right\} && (\text{car } \mathbf{A}\delta\mathbf{x} = \delta\mathbf{b})\\
&= \frac{\|\mathbf{A}^{-1}\|/\|\mathbf{x}\|}{1/\|\mathbf{b}\|} && (\text{car } \|\mathbf{A}^{-1}\delta\mathbf{b}\| \leq \|\mathbf{A}^{-1}\|\|\delta\mathbf{b}\|)\\
&= \frac{\|\mathbf{A}^{-1}\|\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} && (\text{car } \mathbf{b} = \mathbf{A}\mathbf{x})\\
&= \|\mathbf{A}^{-1}\|\|\mathbf{A}\| && (\text{car } \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|)\\
&=: \mathrm{K}(\mathbf{A}).
\end{aligned}$$

On peut aussi trouver des relations de perturbations avec un $\delta\mathbf{A}$. On prouve ainsi la propriété suivante. Si $\mathbf{x}$ est la solution exacte du système $\mathbf{A}\mathbf{x} = \mathbf{b}$ et si on considère que la solution $\mathbf{x} + \delta\mathbf{x}$ est la solution numérique obtenue du système perturbé $(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$, $\mathbf{b} \neq 0$, alors

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\mathrm{K}(\mathbf{A})}{1 - \mathrm{K}(\mathbf{A})\|\delta\mathbf{A}\|/\|\mathbf{A}\|}\left(\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}\right). \tag{28}$$

On trouve des matrices avec à la fois un petit déterminant et un petit conditionnement (exemple $\mathrm{diag}(10^{-4}, \ldots, 10^{-4})$) et des matrices avec un grand déterminant et un grand conditionnement.

---

*Exercices*

1. Montrez que pour $\mathbf{A}$ d'ordre $n$, on a les relations suivantes entre conditionnements :

$$\begin{aligned}
\frac{1}{n}\mathrm{K}_2(\mathbf{A}) \leq \mathrm{K}_1(\mathbf{A}) \leq n\mathrm{K}_2(\mathbf{A}),\\
\frac{1}{n}\mathrm{K}_\infty(\mathbf{A}) \leq \mathrm{K}_2(\mathbf{A}) \leq n\mathrm{K}_\infty(\mathbf{A}),\\
\frac{1}{n}\mathrm{K}_1(\mathbf{A}) \leq \mathrm{K}_\infty(\mathbf{A}) \leq n\mathrm{K}_1(\mathbf{A}).
\end{aligned}$$

2. Soit la matrice $\mathbf{B}$ suivante : $b_{ii} = 1$, $b_{ij} = -1$ si $i < j$, $b_{ij} = 0$ si $i > j$. Montrez que $\det(\mathbf{B}) = 1$ et $\mathrm{K}_\infty(\mathbf{B}) = n2^{n-1}$.

---

## 3.5 Résolution de systèmes triangulaires

Il est beaucoup plus facile de résoudre des systèmes triangulaires que des systèmes pleins. Sur ces systèmes (on note $\mathbf{L}$ ou $\mathbf{U}$ la matrice concernée), une condition nécessaire et suffisante d'inversibilité est la non nullité des termes de la diagonale.

Dans le cas $\mathbf{L}\mathbf{x} = \mathbf{b}$,

$$\begin{bmatrix} l_{11} & 0 & \ldots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \ldots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

la résolution s'écrit (*formule de descente*) :

$$x_1 = \frac{b_1}{l_{11}}$$
$$x_i = \frac{1}{l_{ii}} \left( b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right).$$

Pour traiter $\mathbf{Ux} = \mathbf{b}$, le principe est le même sauf qu'on part de $x_n$ pour remonter jusqu'à $x_1$ (*formule de remontées*).

On note qu'il faut $n^2$ opérations (additions, multiplications, divisions) pour chacun de ces algorithmes. Chaque $b_i$ est utilisé exactement une fois.

Ces algorithmes de *substitution directe* lisent les lignes de $L$ ou $U$. On dit qu'ils sont "orientés lignes". On peut écrire des versions colonnes de ces algorithmes (en se plaçant du point de vue de $\mathbf{b}$). Dans le cas où les matrices sont creuses (et représentées par ligne ou par colonne), ce choix a une influence considérable sur le temps de calcul.

Le principe est de voir que $x_1 = \frac{b_1}{l_{11}}$. Ensuite, on change le système pour mettre du côté droit tous les termes avec $x_1$. Le système devient :

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & l_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1/l_{11} \\ b_2 - x_1 l_{21} \\ \vdots \\ b_n - x_1 l_{n1} \end{bmatrix},$$

Cela a un sens car $x_1$ est connu et on peut donc mettre à jour $\mathbf{b}$ à droite. On procède ensuite similairement pour chaque terme. A la fin, on obtient $\mathbf{Ix} = \mathbf{b}'$. L'équivalence des systèmes intermédiaires impliquent que $\mathbf{x} = \mathbf{b}'$ contient le résultat attendu.

Ces algorithmes sont exacts d'un point de vue mathématique, mais il peut y avoir des erreurs d'arrondi d'un point de vue numérique. On rappelle que l'erreur d'arrondi $\mathfrak{u}$ lié à la représentation des nombres en virgule flottante vaut $2^{-t}$ où $t$ est la taille de mantisse.

On considère que la solutions numérique $\hat{\mathbf{x}}$ de $\mathbf{Lx} = \mathbf{b}$ résoud en fait exactement un système linéaire perturbé $(\mathbf{L} + \delta\mathbf{L})\hat{\mathbf{x}} = \mathbf{b}$, où $\delta\mathbf{L}$ est une perturbation de la matrice. On montre que

$$\|\delta\mathbf{L}\| \leq \frac{n\mathfrak{u}}{1 - n\mathfrak{u}} \|\mathbf{L}\| \tag{29}$$

En utilisant le résultat d'analyse a priori (28), on montre alors que

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq n\mathfrak{u}K(\mathbf{L}) + O(\mathfrak{u}^2). \tag{30}$$

La précision de systèmes triangulaires par substitution directe est donc très élevée, car $\mathbf{u}$ est très petit. On sait même que les erreurs sont encore plus petites si les diagonales sont dominantes.

## 3.6 Inversion de matrices triangulaires

On peut calculer explicitement l'inverse d'une matrice triangulaire inférieure ou supérieure en utilisant les formules de descente ou de remontée. Par exemple pour $\mathbf{Lx} = \mathbf{b}$, en notant $\mathbf{V}$ sa matrice inverse (triangulaire inférieure aussi) formée des colonnes $\mathbf{V}_j$, on observe comme $\mathbf{LV} = \mathbf{I}$ :

$$\forall 1 \leq j \leq n, \mathbf{LV}_j = \mathbf{e}_j$$

avec $\mathbf{e}_j = (0, \dots, 0, 1, 0, \dots, 0)^\intercal$ non nul seulement en $j$.

Le nombre total d'opérations est donc de $n^3$ flops. On peut faire un peu mieux en exploitant le fait que la moitié de $\mathbf{V}$ est nulle. On aboutit à un algorithme qui prend environ $\frac{1}{3}n^3 + \frac{3}{4}n^2$ flops. L'ordre de grandeur est inchangé.

## 3.7 Méthode d'élimination de Gauss (factorisation LU)

C'est la méthode à utiliser pour résoudre des systèmes linéaires où $\mathbf{A}$ est dense, sans structure particulière. Comme on sait qu'il est facile de résoudre des systèmes triangulaires, on va chercher à déomposer la matrice $\mathbf{A}$ en un produit de matrices triangulaires $\mathbf{L}$ et $\mathbf{U}$. Ainsi

$$\begin{bmatrix} 3 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 3 & 5 \\ 0 & -3 \end{bmatrix}$$

La solution de $\mathbf{Ax} = \mathbf{b}$ sera cherchée sous la forme

$$\mathbf{Ax} = \mathbf{L} \underbrace{\mathbf{Ux}}_{\mathbf{y}} = \mathbf{Ly} = \mathbf{b}.$$

On résoud donc le système triangulaire inférieur $\mathbf{Ly} = \mathbf{b}$, puis le système triangulaire supérieur $\mathbf{Ux} = \mathbf{y}$.

Le principe de l'élimination de Gauss est tout simplement de soustraire la première équation à toutes les autres de manière à mettre à 0 les termes de la 1ère colonne sous la diagonale. Ensuite, on procède de même sur les autres colonnes.

On note $\mathbf{A}^{(1)} = \mathbf{A}$ et $\mathbf{b}^{(1)}$ les matrice/vecteur initiaux. Ensuite, les itérés seront notés $\mathbf{A}^{(k)}$ et $\mathbf{b}^{(k)}$. On note que seules leurs parties $k+1 : n$ diffèrent de l'état précédent.

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix},$$

$$\begin{aligned} \text{avec} \quad a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}, & i,j &= 2, \dots, n \\ b_i^{(2)} &= b_i^{(1)} - m_{i1} b_1^{(1)}, & i &= 2, \dots, n. \end{aligned}$$

On appelle *multiplicateur* la quantité $m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$, pour $i = 2, \dots, n$, et $m_{ii} = 1$, $m_{ij} = 0$ pour $j > i$.

> On a soustrait la ligne 1 de $\mathbf{A}$ à toutes les autres lignes, en veillant à la multiplier par le multiplicateur correspondant à la ligne, de façon à annuler toute la première colonne de $\mathbf{A}$ (sauf en ligne 1).

Le système $\mathbf{A}^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$ est équivalent au précédent. On procède récursivement sur les sous-matrices $[k : n] \times [k : n]$ de $\mathbf{A}$ et sous-vecteur de $\mathbf{b}$. Ce procédé fonctionne tant qu'aucun élément diagonal est nul.

On montre que cet algorithme effectue environ $\frac{2}{3}n^3 + 2n^2$ flops (en comptant la formule de remontée pour résoudre le système triangulaire supérieure).

Les multiplicateurs forment une matrice triangulaire inférieure. On montre que

$$\mathbf{A} = \mathbf{MU}. \tag{31}$$

On obtient bien la décomposition $\mathbf{A} = \mathbf{LU}$ par cet algorithme.

Quelques explications. On note $M_k$ la matrice $\mathbf{I}$ avec la colonne $k$ égale à $-m_{ik}$. On vérifie simplement que $\mathbf{A}^{(k+1)} = M_k \mathbf{A}^{(k)}$. Ensuite, on a $M_k^{-1}$ est égal à $\mathbf{I}$ avec la colonne $k$ égale à $m_{ik}$. Du coup,

$$\begin{aligned} & \underbrace{M_{n-1} M_{n-2} \cdots M_1}_{\mathbf{M}} \mathbf{A} = \mathbf{U} \\ \Leftrightarrow \quad & \mathbf{A} = \underbrace{M_1^{-1} \cdots M_{n-2}^{-1} M_{n-1}^{-1}}_{\mathbf{M}^{-1}} \mathbf{U} \end{aligned}$$

On montre ensuite assez simplement (par orthogonalité des vecteurs colonnes de la matrice identité), que $M_i^{-1}M_j^{-1}$ est la matrice identité plus les deux colonnes significatives de $M_i^{-1}$ et $M_j^{-1}$ à leur place, *lorsque* $i < j$. Il s'ensuit que $\mathbf{M}^{-1} = M_1^{-1} \cdots M_{n-2}^{-1}M_{n-1}^{-1}$ est une matrice triangulaire inférieure avec diagonale unitaire, notée $\mathbf{L}$.

**Théorème 1** *La factorisation* $\mathbf{LU}$ *de* $\mathbf{A}$ *avec diagonale identité pour* $\mathbf{L}$ *existe si et seulement si les sous-matrices principales d'ordre* $i$ *(de* $1$ *à* $n-1$*) sont inversibles.*

On note aussi que les matrices à diagonale dominante en ligne ou colonne sont factorisables sous cette forme.

On montre que les erreurs numériques de la factorisation $\mathbf{LU}$ sont données par :

$$\|\delta\mathbf{A}\| \leq \frac{n\mathbf{u}}{1-2n\mathbf{u}}\|\mathbf{A}\|. \tag{32}$$

On voit aussi que la perturbation peut être grande à cause de petits pivots, qui induisent de grands multiplicateurs. Or, vu la forme des matrices, $\|\mathbf{A}\| = \|\mathbf{LU}\| \leq \|\mathbf{L}\|\|\mathbf{U}\|$. Or $\|\mathbf{L}\|_1 = \max\sum_i |m_{ij}|$ et $\|\mathbf{L}^{-1}\|_\infty \geq \|\mathbf{L}\|$. Les erreurs numériques seront donc très dépendantes des multiplicateurs.

---

*Exercices*

1. Soient deux matrices triangulaires unitaires inférieures $L_k$ et $L_{k'}$, telles que seules leurs colonnes respectives $k$ et $k'$ ont des éléments non nuls. Pour $k < k'$, que vaut $L_kL_{k'}$ ? Que vaut en revanche $L_{k'}L_k$ ?
2. On note $\mathbf{m}_k = (0, \ldots, 0, m_{k+1,k}, \ldots, m_{n,k})^\intercal$ et $\mathbf{e}_k$ le $k$-ème vecteur colonne de la matrice identité d'ordre $n$. Avec le résultat précédent, montrez que $(\mathbf{I}+\mathbf{m}_i\mathbf{e}_i{}^\intercal)(\mathbf{I}+\mathbf{m}_j\mathbf{e}_j{}^\intercal) = (\mathbf{I}+\mathbf{m}_i\mathbf{e}_i{}^\intercal+\mathbf{m}_j\mathbf{e}_j{}^\intercal)$ si $i < j$. Que vaudrait le produit inverse ?
   En déduire que $\Pi_{i=1}^n(\mathbf{I}+\mathbf{m}_i\mathbf{e}_i{}^\intercal) = \mathbf{I}+\sum_{i=1}^n \mathbf{m}_i\mathbf{e}_i{}^\intercal$.
   Dans l'élimination de Gauss, on a

$$\begin{aligned}\mathbf{M} &= \mathbf{I}+\sum_{i=1}^n \mathbf{m}_i\mathbf{e}_i{}^\intercal \\ &= \Pi_{i=1}^n(\mathbf{I}+\mathbf{m}_i\mathbf{e}_i{}^\intercal) \\ &= \Pi_{i=1}^n(\mathbf{I}-\mathbf{m}_i\mathbf{e}_i{}^\intercal)^{-1} \\ &= \left(\Pi_{i=n}^1(\mathbf{I}-\mathbf{m}_i\mathbf{e}_i{}^\intercal)\right)^{-1}.\end{aligned}$$

---

## 3.8 Elimination de Gauss avec choix du pivot

On a vu dans les sections précédentes que des instabilités numériques apparaissent pour des multiplicateurs trop grands, ce qui est lié à un pivot trop petit par rapport aux éléments de sa colonne. Rien n'empêche de choisir un autre élément comme pivot. Le *pivotage partiel* choisit le pivot dans la même colonne $\mathbf{A}_{k:n,k}$ (en dessous du pivot $a_{kk}$ usuel), le *pivotage total* choisit le pivot dans la sous-matrice $\mathbf{A}_{k:n,k:n}$ (en dessous ou à droite du pivot usuel). On prend en général l'élément le plus grand en valeur absolue. Ainsi les multiplicateurs seront tous inférieurs ou égaux à 1.

On s'intéresse au pivotage partiel, et soit $a_{lk}$ le pivot choisi, avec $k \leq l \leq n$. On va simplement échanger les lignes $k$ et $l$ pour amener cette valeur sur la ligne usuelle. Ensuite on procède à l'identique. Algébriquement, la permutation des lignes se fait à l'aide d'une matrice $\mathbf{P}$ égale à l'identité sauf que le lignes $k$ et $l$ ont été échangées.

Notons $P_k$ les matrices "échanges de lignes" effectuées à chaque étape.

$$\underbrace{M_{n-1}P_{n-1}M_{n-2}P_{n-2}\cdots M_1P_1}_{\mathbf{M}}\mathbf{A} = \mathbf{U}$$
$$\Leftrightarrow \quad (\mathbf{MP}^{-1})\mathbf{PA} = \mathbf{U}$$
$$\Leftrightarrow \quad \mathbf{PA} = \underbrace{\mathbf{P}P_1^{-1}M_1^{-1}\cdots P_{n-2}^{-1}M_{n-2}^{-1}P_{n-1}^{-1}M_{n-1}^{-1}}_{\mathbf{L}}\mathbf{U}$$

On montre que la matrice $\mathbf{L}$ résultante est bien triangulaire inférieure. Algorithmiquement, on met à jour à chaque itération $k$ la matrice des permutations $\mathbf{P}$ en échangeant ligne $k$ et $r$ (évidemment si $k \neq r$). De plus on échange ligne $k$ et $r$ de $\mathbf{U}$ (entre $k$ et $n$) et de $\mathbf{L}$ (entre 0 et $k$). Le reste est inchangé.

## 3.9 Factorisation $LDM^\intercal$ et de Cholesky

Si une matrice $\mathbf{A}$ a une décomposition $\mathbf{LU}$, il est facile de voir qu'elle peut avoir une décomposition avec deux matrices triangulaires inférieures unitaire $\mathbf{L}$ et $\mathbf{M}$, et une matrice diagonale $\mathbf{D}$ :

$$\mathbf{A} = \mathbf{LDM}^\intercal. \tag{33}$$

Il suffit de choisir $\mathbf{D}$ comme la diagonale des éléments de $\mathbf{U}$ et de poser $\mathbf{M}^\intercal = \mathbf{D}^{-1}\mathbf{U}$.

Si cette factorisation a peu d'intérêt pratique pour des matrices $\mathbf{A}$ générale, elle est très intéressante pour des matrices symétriques. Dans ce cas, $\mathbf{M} = \mathbf{L}$, et il n'est plus nécessaire que de calculer la moitié des coefficients de $\mathbf{LU}$.

**Théorème 2** *Si* $\mathbf{A}$ *est symétrique définie positive, alors il existe une unique matrice triangulaire inférieure* $\mathbf{H}$ *de termes diagonaux strictement positifs telle que* $\mathbf{A} = \mathbf{HH}^\intercal$ *(dite* factorisation de Cholesky*). On peut la calculer ainsi, colonne après colonne :*

$$h_{11} = \sqrt{a_{11}}$$
$$h_{j1} = a_{1j}/h_{11}, \ \textit{pour } j = 2, \ldots, n,$$

*et pour* $i = 2, \ldots, n$

$$h_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} h_{ik}^2\right)^{\frac{1}{2}},$$
$$h_{ji} = \left(a_{ij} - \sum_{k=1}^{i-1} h_{ik}h_{jk}\right)/h_{ii}, \ \textit{pour } i+1 \leq j \leq n$$

Cet algorithme ne coûte que $n^3/3$ flops et est stable par rapport aux erreurs d'arrondi. On montre que $\|\delta\mathbf{A}\|_2 \leq 8n(n+1)\mathfrak{u}\|\mathbf{A}\|_2$.

On vérifie que cet algorithme fonctionne en formant la $j$-ème colonne du produit $\mathbf{A} = \mathbf{HH}^\intercal$ :

$$\mathbf{A}_{1:n,j} = \sum_{k=1}^{j} \mathbf{H}_{jk}\mathbf{H}_{1:n,k}$$
$$\Leftrightarrow \mathbf{H}_{jj}\mathbf{H}_{1:n,j} = \mathbf{A}_{1:n,j} - \sum_{k=1}^{j-1} \mathbf{H}_{jk}\mathbf{H}_{1:n,k} =: \mathbf{z}$$

Le terme de droite est connu et on peut calculer $\mathbf{z}$. Ensuite, on voit facilement que $\mathbf{H}_{ij} = 0$ pour $i < j$, $\mathbf{H}_{jj}\mathbf{H}_{jj} = z_j$, et $\mathbf{H}_{ij} = z_i/\mathbf{H}_{jj}$ pour $i > j$.

---

*Exercices*

1. Si on veut écrire une fonction pour calculer la factorisation de Cholesky d'une matrice creuse $\mathbf{A}$, quelle type de matrice creuse faut-il choisir pour $\mathbf{A}$ et $\mathbf{H}$ ?

---

## 3.10 Considérations numériques

Si $\mathbf{A}^{-1}$ est calculé exactement, alors la solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ induit une erreur $\|\mathbf{b} - \mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|\mathrm{K}(\mathbf{A})O(\mathfrak{u})$, car le conditionnement de la multiplication est aussi $\mathrm{K}(\mathbf{A})$ (voir TD).

---

*Exercices*

1. Montrons le résultat précédent. Soit $f(\mathbf{d}) = \mathbf{B}\mathbf{d}$ la fonction produit d'une matrice $\mathbf{B}$ avec un vecteur $\mathbf{d}$. Soit $\tilde{f}$ la fonction numérique correspondant à $f$. On note d'abord que $f$ est stable amont comme produit matrice/vecteur. La stabilité amont implique que $\exists\tilde{\mathbf{d}}$, $\frac{\|\tilde{\mathbf{d}}-\mathbf{d}\|}{\|\mathbf{d}\|} \leq O(\mathfrak{u})$ et $\tilde{f}(\mathbf{d}) = f(\tilde{\mathbf{d}})$.

   Du coup, la stabilité amont est reliée au conditionnement relatif de $f$ ainsi (voir chapitre précédent, avec $\delta\mathbf{y} = \tilde{f}(\mathbf{d}) - f(\mathbf{d})$) :

$$\frac{\|\delta\mathbf{y}\|}{\|\mathbf{y}\|} \leq \frac{\|\tilde{\mathbf{d}} - \mathbf{d}\|}{\|\mathbf{d}\|} K_{rel}(\mathbf{d}) \leq K_{rel}(\mathbf{d})O(\mathfrak{u})$$

   Or le conditionnement relatif est borné ainsi :

$$\begin{aligned} K_{rel}(\mathbf{d}) &= \frac{\|f'(\mathbf{d})\|\|\mathbf{d}\|}{\|f(\mathbf{d})\|} = \frac{\|\mathbf{B}\|\|\mathbf{d}\|}{\|\mathbf{B}\mathbf{d}\|} = \frac{\|\mathbf{B}\|\|\mathbf{B}^{-1}\mathbf{y}\|}{\|\mathbf{y}\|} \\ &\leq \frac{\|\mathbf{B}\|\|\mathbf{B}^{-1}\|\|\mathbf{y}\|}{\|\mathbf{y}\|} = \|\mathbf{B}\|\|\mathbf{B}^{-1}\| = \mathrm{K}(\mathbf{B}). \end{aligned}$$

   Pour conclure, la solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ induit une erreur (avec $\mathbf{b} \equiv \mathbf{d}$, $\mathbf{x} \equiv \mathbf{y}$ et $\mathbf{A}^{-1} \equiv \mathbf{B}$) :

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} = \frac{\|\delta\mathbf{y}\|}{\|\mathbf{y}\|} \leq K_{rel}(\mathbf{d})O(\mathfrak{u}) \leq \mathrm{K}(\mathbf{A}^{-1})O(\mathfrak{u}) = \mathrm{K}(\mathbf{A})O(\mathfrak{u}).$$

   On peut aussi mesurer l'erreur (ou résidu) entre $\mathbf{b}$ et $\mathbf{A}\tilde{\mathbf{x}}$

$$\|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\| = \|\mathbf{A}(\mathbf{x} - \tilde{\mathbf{x}})\| \leq \|\mathbf{A}\|\|\mathbf{x}\|\mathrm{K}(\mathbf{A})O(\mathfrak{u}).$$

   Si on normalise le résidu, on voit bien que le conditionnement détermine l'ampleur possible des erreurs.

$$\frac{\|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\|}{\|\mathbf{A}\|\|\mathbf{x}\|} \leq \mathrm{K}(A)O(\mathfrak{u}).$$

---

Or avec PLU, l'erreur est $\|\mathbf{b} - \mathbf{A}\mathbf{x}\| \leq O(\|\mathbf{A}\| + \|\mathbf{U}\|)\|\mathbf{x}\|\mathfrak{u}$, ce qui est toujours plus favorable.

## 3.11 Considérations algorithmiques

Dans la plupart des problèmes concrets, les matrices $\mathbf{A}$ peuvent être très grandes (plusieurs millions de lignes et colonnes parfois), mais elles sont souvent très creuses. En effet, les matrices expriment des relations entre des inconnues. Or, dans les systèmes physiques, il est rare que des éléments très éloignés aient une influence directe les uns sur les autres. Si on note $n_{nz}$ le nombre d'éléments différents de 0 dans la matrice $\mathbf{A}$, il faut mettre au point des algorithmes et des structures de données qui tiennent compte de ce fait. On a le théorème suivant :

**Théorème 3 (Gilbert, Peierls)** *La décomposition LU d'une matrice* $\mathbf{A}$ *avec* $n_{nz}$ *coefficients non nuls peut être implémentée de manière à s'exécuter en temps* $O(flops(\mathbf{LU}))$*, où* $flops(\mathbf{LU})$ *compte le nombre de multiplications à coefficients non nuls dans le produit* $\mathbf{LU}$*.*

Si $n$ est l'ordre de $\mathbf{A}$, $n^*$ le nombre d'éléments non nuls dans la factorization, alors :

$$n \leq n_{nz} \leq n^* \leq n^2 \quad \text{et} \quad n^* \leq flops(\mathbf{LU}) \leq n^3.$$

Pour les matrices denses, $n_{nz} = n^* = n^2$ et $flops(\mathbf{LU}) = n^3$. Pour les matrices creuses, dans l'exemple typique de la maille bidimensionnelle, on a $n_{nz} = \Theta(n)$, $n^* = \Theta(n \log n)$, et $flops(\mathbf{LU}) = n^{\frac{3}{2}}$.

Néanmoins, l'algorithme n'est pas trivial et la constante dépend de la structure des relations entre éléments.

On note les points suivants :

— même si $\mathbf{A}$ est creux, $\mathbf{L}$ et $\mathbf{U}$ ne sont pas forcément creux (il y a même un facteur de remplissage quasi minimum qui double le nombre de coefficients non nuls par ligne).

— en général $\mathbf{A}^{-1}$ est plein.

— si $\mathbf{A}$ est symétrique et inversible, alors la largeur de bande de $\mathbf{A}$ est conservée dans $\mathbf{L}$ et $\mathbf{U}$.

On cherche donc à minimiser la largeur de bande.

L'algorithme de Cuthill-McKee (1969) procède ainsi. D'abord on choisit un sommet périphérique (un sommet avec un degré faible), $R := (\{x\})$.

Ensuite pour $i = 1, 2, \ldots$ on itère les étapes suivantes tant que $|R| < n$.

— On construit l'ensemble d'adjacence $A_i$ de $R_i$ (avec $R_i$ la i-ème composante de $R$) en excluant les sommets déjà dans $R$,

$$A_i := \mathrm{Adj}(R_i) \setminus R$$

— On trie $A_i$ par degrés croissants.

— On rajoute $A_i$ à la fin de la liste $R$

En d'autres termes, on numérote les sommets en fonction d'un parcours en largeur particulier où les sommets voisins sont visités dans l'ordre du degré le plus petit au plus grand.

L'algorithme Reverse Cuthill-McKee (1969) inverse juste l'ordre des sommets à la fin.

Cela nous donne une matrice de permutation des sommets que l'on applique sur la matrice $\mathbf{A}$ avant de résoudre LU.

## 3.12 Représentations des matrices creuses

Il faut trouver des structures de données qui permettent :

— économiser la mémoire utilisée, i.e. proportionnel à $n_{nz}$

— accéder efficacement à un élément

— accéder efficacement à tous les éléments d'une ligne / d'une colonne

— modifier les éléments d'une ligne / d'une colonne

— extraire une ligne / une colonne

— faire des multiplications matrice / vecteur ou matrice / matrice

On ne peut avoir des structures de données idéales pour tous ces points. Les trois représentations les plus utilisées sont :

**COO** format triplet ou coordonnées. On stocke des triplets valeur, colonne, ligne.

Pratique pour construire une matrice creuse. Sinon, inefficace.

**CSR** format ligne compressée. Il utilise trois tableaux : le tableau 'data' des valeurs de taille nnz, le tableau 'indices' de taille nnz qui contient la colonne de chaque valeur, le tableau 'indptr' de taille nombre de lignes+1, qui stocke le numéro du premier élément dans sa ligne.

- opérations arithmétiques efficaces CSR + CSR, CSR * CSR, etc. - extraction d'une ligne rapide - produit matrices / vecteurs rapides - mais assez lent d'extraire une colonne ou de faire des mises à jour

**CSC** format colonne compressée. Même principe que CSR, mais par colonne.

- opérations arithmétiques efficaces CSR + CSR, CSR * CSR, etc. - extraction d'une colonne rapide - produit matrices / vecteurs rapides (CSR plus rapide) - mais assez lent d'extraire une ligne ou de faire des mises à jour

Par exemple, l'algorithme SuperLU utilise les formats suivants pour ses calculs : A CSC, L CSR, U CSC.

---

*Exercices*

1. Ecrire le pseudo-code pour accéder à l'élément $(i, j)$ d'une matrice CSR $A$. Quelle est la complexité de l'opération ?
2. Ecrire le pseudo-code pour faire la multiplication matrice CSR / vecteur. Quelle est la complexité de l'opération ?

---

## 3.13 Méthodes itératives pour la solution de systèmes linéaires

Elles donnent une solution à un système $\mathbf{Ax} = \mathbf{b}$ en un nombre infini d'itération en calculant le résidu $\mathbf{Ax} - \mathbf{b}$ à chaque itération. Dans le cas d'une matrice pleine, leur coût par itération est de l'ordre $n^2$ alors que les méthodes directes ont un coût $\frac{2}{3}n^3$. Elles peuvent être intéressantes lorsqu'il faut peu d'itérations pour converger, ou dans le cas des matrices creuses.

On construit une suite convergente de vecteur $\{\mathbf{x}^{(k)}\}$ telle que $\mathbf{x} = \lim_{k\to+\infty} \mathbf{x}^{(k)}$, avec $\mathbf{x}$ la solution de $\mathbf{Ax} = \mathbf{b}$. En pratique on arrête l'algorithme selon un certain critère qui ne peut pas dépendre de $\mathbf{x}$ qui est inconnu.

On s'intéresse aux méthodes itératives de la forme :

$$\mathbf{x}^{(0)} \text{ donné}, \quad \mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{f}. \tag{34}$$

$B$ est une matrice d'ordre $n$ appelée matrice d'itération et $\mathbf{f}$ dépend de $\mathbf{b}$.

On dit qu'une méthode est *consistante* si $\mathbf{x} = \mathbf{Bx} + \mathbf{f}$. Autrement dit $\mathbf{f} = (\mathrm{Id} - \mathbf{B})\mathbf{A}^{-1}\mathbf{b}$.

On a le théorème suivant :

**Théorème 4** *Si la méthode itérative est consistante, la suite de vecteur $\mathbf{x}^{(k)}$ de (34) converge vers $\mathbf{x}$ pour toute valeur de $\mathbf{x}^{(0)}$ si et seulement si le rayon spectral de $\mathbf{B}$ est strictement plus petit que 1.*

> Exemple : On montre par exemple que, pour le système linéaire $2\mathrm{Id}\mathbf{x} = \mathbf{b}$, la méthode itérative $\mathbf{x}^{(k+1)} = -\mathbf{x}^{(k)} + \mathbf{b}$ ne converge pas pour $\mathbf{x}^{(0)} \neq \mathbf{b}/2$.
> Par exemple, $\mathbf{x}^{(0)} = \mathbf{0}$, $\mathbf{x}^{(1)} = \mathbf{b}$, $\mathbf{x}^{(2)} = \mathbf{0}$, etc.
> Par exemple, $\mathbf{x}^{(0)} = -\mathbf{b}$, $\mathbf{x}^{(1)} = 2\mathbf{b}$, $\mathbf{x}^{(2)} = -\mathbf{b}$, etc.

On rappelle que le rayon spectral $\rho(\mathbf{A})$ est plus petit ou égal à n'importe quelle norme consistante sur $\mathbf{A}$. Si $\mathbf{A}$ est hermitienne, il y a égalité entre $\|\mathbf{A}\|_2 = \rho(\mathbf{A})$. De plus $\lim_{m\to+\infty} \|\mathbf{A}^m\|^{1/m} = \rho(\mathbf{A})$.

On se restreint aux méthodes itératives linéaires. On va chercher à décomposer $\mathbf{A}$ sous la forme $\mathbf{A} = \mathbf{P} - \mathbf{N}$ avec $\mathbf{P}$ inversible, appelé *préconditionneur*. Sachant $\mathbf{x}^{(0)}$, on résoud à chaque itération le système suivant :

$$\mathbf{P}\mathbf{x}^{(k+1)} = \mathbf{N}\mathbf{x}^{(k)} + \mathbf{b}. \tag{35}$$

Avec les notations ci-dessous, on a $\mathbf{B} = \mathbf{P}^{-1}\mathbf{N}$ et $\mathbf{f} = \mathbf{P}^{-1}\mathbf{b}$. En notant $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ le *résidu*, on peut aussi écrire (35) sous la forme :

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \mathbf{P}^{-1}\mathbf{N}\mathbf{x}^{(k)} + \mathbf{P}^{-1}\mathbf{b} \\ &= \mathbf{B}\mathbf{x}^{(k)} + \mathbf{P}^{-1}(\mathbf{r}^{(k)} + (\mathbf{P} - \mathbf{N})\mathbf{x}^{(k)}) \\ &= \mathbf{x}^{(k)} + \mathbf{P}^{-1}\mathbf{r}^{(k)}.\end{aligned} \tag{36}$$

### 3.13.1 Méthode de Jacobi

Elle ne fonctionne que si les coefficients diagonaux sont non nuls. Son principe est de calculer $x_i^{(k+1)}$ à partir des $(x_j^{(k)})_{j\neq i}$.

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n. \tag{37}$$

La méthode de Jacobi consiste à choisir, dans l'équation (35), $\mathbf{P} = \mathbf{D}$, et $\mathbf{N} = \mathbf{D} - \mathbf{A}$ avec $\mathbf{D}$ la matrice ne contenant que la diagonale de $\mathbf{A}$. On découpe aussi $\mathbf{N} = \mathbf{E} + \mathbf{F}$ en deux matrices $\mathbf{E}$ et $\mathbf{F}$ où $\mathbf{E}$ est la matrice triangulaire inférieure telle que $e_{ij} = -a_{ij}$ et $\mathbf{F}$ est la matrice triangulaire supérieure telle que $f_{ij} = -a_{ij}$.

On peut calculer la matrice d'itération de Jacobi :

$$\mathbf{B}_J = \mathbf{P}^{-1}\mathbf{N} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A}) = \mathbf{Id} - \mathbf{D}^{-1}\mathbf{A}.$$

On peut aussi moduler les pas par $\omega$ dans cette méthode (méthode de surrelaxation, ou *Jacobi over relaxation (JOR)*) :

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right) + (1 - \omega) x_i^{(k)}, \quad i = 1, \ldots, n. \tag{38}$$

Et la matrice d'itération de JOR est alors :

$$\mathbf{B}_J(\omega) = \omega \mathbf{B}_J + (1 - \omega)\mathbf{Id}. \tag{39}$$

On peut l'écrire aussi très simplement sous la forme (36) (avec le résidu) :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega \mathbf{D}^{-1}\mathbf{r}^{(k)}.$$

Il est alors évident que cette méthode est consistante pour tout $\omega \neq 0$ et coïncide avec Jacobi pour $\omega = 1$.

### 3.13.2 Méthode de Gauss-Seidel

Elle diffère de la méthode de Jacobi car elle utilise à l'itération $k+1$ des valeurs de $\mathbf{x}^{(k)}$ et des valeurs déjà calculées de $\mathbf{x}^{(k+1)}$.

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n. \tag{40}$$

On peut l'écrire avec la forme $\mathbf{P}, \mathbf{N}$ ainsi : $\mathbf{P} = \mathbf{D} - \mathbf{E}$, $\mathbf{N} = \mathbf{F}$. Sa matrice d'itération est :

$$\mathbf{B}_{GS} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{F}.$$

On peut aussi introduire la méthode de *sur-relaxation successive (SOR)* comme pour la méthode de Jacobi :

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) + (1 - \omega) x_i^{(k)}, \quad i = 1, \ldots, n. \tag{41}$$

On peut trouver que sa matrice d'itération est

$$\mathbf{B}_{GS}(\omega) = (\mathrm{Id} - \omega \mathbf{D}^{-1}\mathbf{E})^{-1}[(1 - \omega)\mathrm{Id} + \omega \mathbf{D}^{-1}\mathbf{F}].$$

On peut l'écrire aussi très simplement sous la forme (36) (avec le résidu) :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \left( \frac{1}{\omega}\mathbf{D} - \mathbf{E} \right)^{-1} \mathbf{r}^{(k)}.$$

La méthode SOR est consistante pour tout $\omega \neq 0$, correspond avec Gauss-Seidel pour $\omega = 1$. Si $\omega$ est entre 0 et 1 on parle de sous-relaxation et si $\omega > 1$ on parle de sur-relaxation.

### 3.13.3 Convergences des méthodes itératives

Les méthodes de Jacobi et Gauss-Seidel sont dominantes pour toute matrice à coefficients diagonaux dominants.

**Théorème 5** *Si* $\mathbf{A}$ *est une matrice à coefficients diagonaux dominants stricts, les méthodes de Jacobi et Gauss-Seidel sont convergentes.*

*Preuve.* (Pour Jacobi seulement). Si $\mathbf{A}$ est à diagonale dominante stricte, on a $\forall i, |a_{ii}| > \sum_{j\neq i} |a_{ij}|$. On a donc $\|\mathbf{B}_J\|_\infty = \|\mathbf{D}^{-1}(\mathbf{D}-\mathbf{A})\| = \max_i \sum_{j\neq i} |a_{ij}|/|a_{ii}| < 1$. Le théorème 4 permet de conclure. □

Exemple : Dans le cas où la dominance diagonale n'est pas stricte, Jacobi peut ne pas converger. Prenons $\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$, alors Jacobi s'écrit :

$$\mathbf{x}^{(k+1)} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \mathbf{x}^{(k)} + \mathbf{b}.$$

On voit que si on cherche $\mathbf{Ax} = \mathbf{0}$, et que l'on part avec $\mathbf{x}^{(0)} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal$, on obtient successivement :

$$\mathbf{x}^{(1)} = \begin{bmatrix} 0 & -1 \end{bmatrix}^\intercal, \qquad \mathbf{x}^{(2)} = \begin{bmatrix} -1 & 0 \end{bmatrix}^\intercal, \qquad \mathbf{x}^{(3)} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\intercal, \qquad \mathbf{x}^{(4)} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal.$$

**Théorème 6** *Si* $\mathbf{A}$ *et* $2\mathbf{D} - \mathbf{A}$ *sont sym. def. pos. (sdp) alors la méthode de Jacobi est convergente et* $\rho(\mathbf{B}_J) = \|\mathbf{B}_J\|_\mathbf{A} = \|\mathbf{B}_J\|_\mathbf{D}$.

**Théorème 7** *Si* $\mathbf{A}$ *est sdp, la méthode JOR est convergente si* $0 < \omega < 2/\rho(D^{-1}\mathbf{A})$.

**Théorème 8** *Si* $\mathbf{A}$ *est sdp, la méthode de Gauss-Seidel converge de manière monotone pour la norme* $\|\cdot\|_\mathbf{A}$.

Si $\mathbf{A}$ est une matrice tridiagonale sdp (et plus généralement pour des matrices avec la $A$-propriété) alors les méthodes de Jacobi et Gauss-Seidel sont convergentes avec $\rho(\mathbf{B}_{GS}) = \rho^2(\mathbf{B}_J)$. La méthode de Gauss-Seidel converge alors plus vite que celle de Jacobi. Dans le cas général, on ne peut rien conclure sur les vitesses de convergence.

**Théorème 9** *Si la méthode de Jacobi converge, alors la méthode JOR converge pour* $0 < \omega \leq 1$.

*Preuve.* Il suffit de remarquer que les valeurs propres $\mu_k$ de $\mathbf{B}_J(\omega)$ s'écrivent en fonction des valeurs propres $\lambda_k$ de $\mathbf{B}_J$ grâce à la relation (39) ($\mathbf{B}_J(\omega) = \omega\mathbf{B}_J + (1-\omega)\mathbf{Id}$) :

$$\mu_k = \omega\lambda_k + (1-\omega).$$

Avec $\lambda_k = r_k e^{i\theta_k}$, on trouve

$$\begin{aligned} |\mu_k|^2 &= \omega^2 r_k^2 + 2\omega(1-\omega) r_k \cos(\theta_k) + (1-\omega)^2 \\ &\leq (\omega r_k + (1-\omega))^2 \leq 1. \end{aligned}$$

Le rayon spectral de $\mathbf{B}_J(\omega)$ est donc inférieure ou égal 1. □

**Théorème 10** *Si* $\mathbf{A}$ *est spd alors la méthode SOR converge si et seulement si* $0 < \omega < 2$ *et sa convergence est monotone pour* $\|\cdot\|_\mathbf{A}$.

Si $\mathbf{A}$ est à diagonale dominante stricte alors SOR converge si $0 < \omega \leq 1$.

**Remarque 2** *On peut rendre symétrique les méthodes Gauss-Seidel et SOR en faisant une itération normale suivie d'une itération rétrograde (c'est la matrice $F$ qui est placée à gauche). Du coup la matrice de préconditionnement devient symétrique.*

### 3.13.4 Autres méthodes itératives

Ce sont des variantes des méthodes précédentes, qui ne sont parfois applicables que sur des matrices sdp :

- les méthodes de Richardson stationnaires ou instationnaires (le paramètre de relaxation peut varier à chaque pas)
- introduction de préconditionneur sur les méthodes précédentes (matrice diagonale, factorisation incomplète LU)
- la méthode du gradient (méthode de Richardson avec pas fixe dans le cas où est symétrique) et la méthode du gradient à pas optimal lorsqu'on adapte le pas à chaque itération.
- la méthode du gradient conjugué (qui garantit qu'on explore des directions orthogonales deux à deux) pour les matrices sdp

### 3.13.5 Critères d'arrêt sur les méthodes itératives

On montre que

$$\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leq \frac{\|\mathbf{B}\|}{1 - \|\mathbf{B}\|} \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|.$$

On peut minorer $\|\mathbf{B}\|$ par le ratio des deux variations $\delta_{k+1}/\delta_k$ avec $\delta_{k+1} = \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$. On peut donc estimer l'erreur $\epsilon^{(k)}$ à l'étape $k$ comme :

$$\epsilon^{(k+1)} \approx \frac{\delta_{k+1}^2}{\delta_k - \delta_{k+1}}.$$

On peut aussi utiliser le résidu et on obtient $\epsilon^{(k)} \geq \frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|}$.

# 4 Fonctions : approximation, intégration, interpolation

Dans ce chapitre, nous nous intéressons à quelques méthodes numériques pour approcher les fonctions, leurs dérivées ou leurs intégrales, ainsi qu'à construire des fonctions qui approchent des points. Ce chapitre n'a rien d'exhaustif, la litérature étant très abondante sur le sujet.

Sauf explicitement mentionné, dans tout le chapitre, $f$ est une fonction définie sur un intervalle $[a, b]$, à valeur dans un espace vetoriel normé réel (souvent $\mathbb{R}$), continue et dérivable autant de fois que nécessaire (on dira de classe $C^k$ pour $k$-fois dérivable et de $k$-ème dérivée continue).

## 4.1 Formules de Taylor, Taylor-Young, Taylor-Lagrange

Un outil naturel pour approcher les fonctions est d'utiliser des polynômes, qui ont sont des fonctions faciles à évaluer et à manipuler ensuite.

Soit $c \in [a, b]$ alors la formule de Taylor ou Taylor-Young, ou développement de Taylor en $c$, est

$$\forall x \in [a, b], \quad f(x) = f(c) + \frac{f'(c)}{1!}(x - c) + \frac{f^{(2)}(c)}{2!}(x - c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x - c)^n + R_n(x), \quad (42)$$

où le *reste* $R_n(x)$ est une fonction négligeable devant $(x-c)^n$ au voisinage de $c$. On a $R_n(x) = o((x-c)^n)$ et même $R_n(x) = O((x - c)^{n+1})$ si $f$ de classe $C^{n+1}$.

Nous utiliserons volontiers la formule de Taylor-Lagrange qui, elle, donne une écriture explicite du reste à l'ordre $n$ (lorsque $f$ est de classe $C^{n+1}$). Pour tout $x \in [a, b] \setminus \{c\}$, il existe un nombre réel $\xi$ strictement compris entre $c$ et $x$ tel que

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n + 1)!}(x - c)^{n+1}. \quad (43)$$

La plupart de ces formules s'étendent en dimension supérieure, ainsi que pour des fonctions à valeur complexe.

Exemples :

- Si $f(x) = \sin(x)$, alors $f(x) = f(0) + \frac{x}{1!}f'(0) + \frac{x^2}{2!}f''(0) + \frac{x^3}{3!}f'''(0) + \frac{x^4}{4!}f'''(0) + o(x^4)$, d'où $f(x) = x - \frac{x^3}{6} + o(x^4)$
- Si $f(x) = \frac{1}{1-x}$ alors $f(x) = 1 + x + x^2 + \cdots + x^n + o(x^n)$
- Si $f(\mathbf{x})$ est de $\mathbb{R}^n$ vers $\mathbb{R}$ alors $f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + J_f(\mathbf{x})\mathbf{h} + \frac{1}{2}\mathbf{h}^\intercal \mathbb{H}_f(\mathbf{x})\mathbf{h} + o(\|\mathbf{h}\|^2)$, avec $J_f(\mathbf{x})$ le jacobien de $f$ au point $\mathbf{x}$ (un vecteur ligne ici) et $\mathbb{H}_f(\mathbf{x})$ sa matrice hessienne au point $\mathbf{x}$.
- on vérifie facilement que si $f$ est un polynôme de degré $n$ alors son développement de Taylor est égal à lui-même. Son développement en 0 correspond à ses monômes dans l'ordre des degrés.

## 4.2 Approximation des dérivées

Dans une résolution numérique d'équations différentielles, on aura très souvent besoin d'approcher les dérivées à partir de la valeur d'une fonction en quelques points. Pour $i$ de 0 à $n$, soient $x_i := x + ih$ avec $h = (b - a)/n$ le pas entre les points. On cherche à approcher $u_i := f'(x_i)$ à partir des valeurs $f(x_k)$ pour $k$ entre $-m$ et $m$. On cherche ainsi les $\beta_k$ tels que :

$$hu_i = \sum_{k=-m}^{m} \beta_k f(x_{i-k}) \quad (44)$$

On appelle *stencil* le voisinage des points impliqués dans ce calcul. Plus il est grand, plus le coût de calcul sera important mais on pourra espérer une précision plus grande.

### 4.2.1 Méthodes des différences finies

D'après (43), nous avons :

$$\begin{aligned} f(x) &= f(x_i) + \frac{f'(x_i)}{1!}(x - x_i) + R_1(x - x_i) \\ \Leftrightarrow f'(x_i) &= \frac{f(x) - f(x_i)}{x - x_i} + \frac{R_1(x - x_i)}{x - x_i}, \end{aligned}$$

avec $R_1(x-x_i) = \frac{f^{(2)}(\xi_i)}{2!}(x-x_i)^2$, et $\xi_i \in ]x_i, x[$.

En prenant $x = x_{i+1}$ et en notant que $x_{i+1} - x_i = h$, on définit par analogie

$$u_i^{FD} := \frac{1}{h}\left(f(x_{i+1}) - f(x_i)\right), \tag{45}$$

appelée *différence finie progressive* (ou *forward difference*) en $x_i$. Dans (44), on a pris $m = 1$, $\beta_{-1} = 1$, $\beta_0 = -1$, $\beta_1 = 0$.

On note immédiatement que $u_i^{FD} - f'(x_i) = h\frac{f^{(2)}(\xi_i)}{2!}$, ce qui signifie que $u^{FD}$ est proche de $f'$ à l'ordre 1.

De façon analogue, on peut définir les *différences finies rétrogrades* (ou *backward difference*) ainsi

$$u_i^{BD} := \frac{1}{h}\left(f(x_i) - f(x_{i-1})\right). \tag{46}$$

Cela revient à prendre, dans (44), $m = 1$, $\beta_{-1} = 0$, $\beta_0 = 1$, $\beta_1 = -1$. L'erreur est la même que pour les différences finies progressives, c'est-à-dire une approximation d'ordre 1.

On peut aussi approcher la dérivée en utilisant le développement de Taylor en $x_i$ aux points $x_{i+1}$ et $x_{i-1}$ :

$$\begin{aligned} f(x_{i+1}) &= f(x_i) + hf'(x_i) + \frac{h^2}{2}f''(x_i) + R_2(h), \\ f(x_{i-1}) &= f(x_i) - hf'(x_i) + \frac{h^2}{2}f''(x_i) + R_2'(-h), \end{aligned} \tag{47}$$

en soustrayant l'une à l'autre, on élimine certains termes, ce qui donne :

$$f(x_{i+1}) - f(x_{i-1}) = 2hf'(x_i) + R_2(h) - R_2'(-h). \tag{48}$$

On définit donc les *différences finies centrées* (ou *centered differences*) ainsi :

$$u_i^{CD} := \frac{1}{2h}\left(f(x_{i+1}) - f(x_{i-1})\right). \tag{49}$$

Cela revient à prendre, dans (44), $m = 1$, $\beta_{-1} = 1/2$, $\beta_0 = 0$, $\beta_1 = -1/2$. D'après (48), on note immédiatement que les différences finies centrées sont une approximation des dérivées à l'ordre 2 :

$$\begin{aligned} u_i^{CD} - f'(x_i) &= \frac{1}{2h}(R_2(h) - R_2'(h)) = \frac{1}{2h}\frac{f^{(3)}(\xi_i) + f^{(3)}(\mu_i)}{3!}(h^3) \\ &= \frac{h^2}{12}(f^{(3)}(\xi_i) + f^{(3)}(\mu_i)). \end{aligned}$$

Géométriquement, les différences finies progressives approchent la dérivée par la pente de la droite entre $(x_i, f(x_i))$ et $(x_{i+1}, f(x_{i+1}))$, les différences finies rétrogrades approchent la dérivée par la pente de la droite entre $(x_{i-1}, f(x_{i-1}))$ et $(x_i, f(x_i))$, les différences finies centrées approchent la dérivée par la pente de la droite entre $(x_{i-1}, f(x_{i-1}))$ et $(x_{i+1}, f(x_{i+1}))$.

On peut construire facilement des schémas d'approximation d'ordre plus élevé en étendant le stencil et en annulant intelligemment les termes dans les développements de Taylor. On peut aussi construire des approximations de dérivées d'ordre supérieur avec les mêmes principes.

Par exemple, en additionnant cette fois-ci les équations de (47) (et en prenant le développement à l'ordre juste supérieur), on obtient :

$$f(x_{i+1}) + f(x_{i-1}) = 2f(x_i) + h^2 f''(x_i) + R_3(h) + R_3'(-h).$$

On voit ainsi que l'on peut approcher $f''(x_i)$ par la difference finie centrée suivante :

$$v_i^{CD} := \frac{1}{h^2}(f(x_{i+1}) - 2f(x_i) + f(x_{i-1})), \tag{50}$$

où $v_i^{CD} - f''(x_i) = \frac{h^2}{24}(f^{(4)}(\xi_i) + f^{(4)}(\mu_i))$.

---

*Exercices*

1. Si $f$ est une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$, comment approcher ses dérivées partielles de façon progressive, rétrograde ou centrée ?
2. Quid de ses dérivées secondes ? On cherche notamment à trouver l'expression du Laplacien de $f$, i.e. $\Delta f := \frac{\partial^2 f}{\partial x \partial x} + \frac{\partial^2 f}{\partial y \partial y}$.

   (Aide : pour chaque point $(x_i, y_i)$ on écrit les fonctions $g_i(x) := f(x, y_i)$ et $h_i(x) := f(x_i, y)$ et on utilise les résultats précédents.)
3. Aux bords, il faut faire attention à exprimer les dérivées avec la même précision qu'à l'intérieur du domaine. Trouvez une expression de $f'(x_0)$ en fonction de $f(x_0)$, $f(x_1)$, et $f(x_2)$ à l'ordre 2.

   (Aide : on écrit le développement de Taylor en $x_0$ au points $x_1$ et et $x_2$ et on annule le terme impliquant la dérivée seconde.)
4. Trouvez une méthode pour approcher $f(x_{i+\frac{1}{2}})$ à l'ordre 2.
5. Que vaut $\int_{x_i}^{x_{i+1}} f'(x)dx$ ? En déduire un lien entre les différences finies progressives et ces intégrales.

---

## 4.3 Différences finies compactes

On peut étendre l'idée des différences finies (cf (44)) en utilisant plusieurs approximations des dérivées. Cela donne à chercher des coefficients $\alpha_k, \beta_k$ dans l'équation suivante :

$$h \sum_{k'=-m'}^{k'} \alpha_{k'} u_i = \sum_{k=-m}^{m} \beta_k f(x_{i-k}). \tag{51}$$

Cela s'appelle les différences finies compactes. En général, il ne faut pas que $m'$ soit trop grand (car il faudra inverser le système linéaire pour le résoudre). Si $m' = 0$, on retombe sur les schémas précédents. Pour $m = 1$, on écrit l'équation précédente sous la forme simplifiée suivante :

$$\alpha u_{i-1} + u_i + \alpha u_{i+1} = \frac{\beta}{2h}(f_{i+1} - f_{i-1}) + \frac{\gamma}{4h}(f_{i+2} - f_{i-2}). \tag{52}$$

On peut montrer qu'il existe des familles de paramètres qui donnent des schémas précis à des ordres 2, 4, ou 6.

Un schéma très utilisé est $\alpha = 1/4$, $\beta = 3/2$, $\gamma = 0$, précis à l'ordre 4. Le seul schéma à l'ordre 6 correspond à $\alpha = 1/3$, $\beta = 14/9$, $\gamma = 1/9$.

On voit immédiatement que l'on doit résoudre un système de la forme $\mathbf{Au} = \mathbf{Bf}$. Attention aux bords, où il faut en général calculer des coefficients spécifiques.

## 4.4 Intégration numérique ou formules de quadrature

On cherche à approcher la valeur de $I(f) := \int_a^b f(x)dx$, dans le cas où on ne peut le faire analytiquement. On appelle formule de quadrature une méthode pour approcher numériquement $I(f)$. Très souvent, l'idée est de remplacer $f$ par une suite de fonctions $f_n$, dont l'intégrale est facilement calculable. On pose $I_n(f) = I(f_n)$, i.e.

$$I_n(f) = \int_a^b f_n(x)dx, \quad n \geq 0.$$

On note *erreur de quadrature* $E_n(f) = I(f) - I_n(f)$. Si $f$ et $f_n$ sont continues sur l'intervalle, on a immédiatement

$$|E_n(f)| \leq \int_a^b |f(x) - f_n(x)|dx \leq (b-a)\|f - f_n\|_\infty$$

Donc si $f_n$ se rapproche de $f$, $E_n(f)$ tend vers 0. Il est donc naturel de prendre pour $f_n$ des polynômes d'interpolation en des points de $f$. On parle de formules de quadrature de Lagrange. Si on utilise aussi les dérivées, on parle de formules de quadrature d'Hermite.

### 4.4.1 Quadratures interpolatoires

Les formules ci-dessous sont liées au *polynôme interpolatoire de Lagrange* $L_f^n$. Soient $(x_i)_{i=0,\dots,n}$ $n+1$ points sur $[a,b]$. On note $l_i$ et $L_f$ les polynômes de degré $n$ définis ainsi :

$$l_i(x) = \Pi_{j=0,j\neq i}^n \frac{x-x_j}{x_i-x_j}, \quad i = 0,\dots,n \tag{53}$$

$$L_f^n(x) = \sum_{i=0}^n f(x_i) l_i(x) \tag{54}$$

Si jamais $f$ était un polynôme de degré $n$, alors tout polynôme de Lagrange utilisant au moins $n+1$ points est égal à $f$, et donc l'utiliser conduit à un erreur de quadrature nulle. On dit qu'un polynôme d'interpolation à $n+1$ points a un degré d'exactitude $n$.

On note que, pour le polynôme de Lagrange, $I_n(f) = \sum_{i=0}^n f(x_i) \int_a^b l_i(x)dx$, qui est un cas particulier de

$$I_n(f) = \sum_{i=0}^n \alpha_i f(x_i). \tag{55}$$

### 4.4.2 Formules du rectangle ou du point milieu

Si on prend un seul point, on prend généralement le point milieu de $[a,b]$ et on aboutit à la formule du rectangle ou du point milieu (ici $\alpha_0 = (b-a)$, $x_0 = \frac{a+b}{2}$) :

$$I_0(f) = (b-a) f\left(\frac{a+b}{2}\right) \tag{56}$$

On trouve que l'erreur se comporte ainsi, si $h = (b-a)$ et $f$ est $C^2$ :

$$E_0(f) = \frac{h^3}{24} f''(\xi)$$

On écrit $f(x) = f(c) + f'(c)(x-c) + f''(\xi(x))(x-c)^2/2$ (Taylor-Lagrange) et on utilise le théorème généralisé de la moyenne, i.e. $\exists \xi \in ]a,b[, f(c)\int_a^b g(x)dx = \int_a^b f(x)g(x)dx$.

Attention, l'erreur est beaucoup moins favorable si on ne prend pas le point au milieu de l'intervalle.

Ces formules se généralisent pour $m$ points dans l'intervalle $[a,b]$ en les plaçant au milieu de rectangles de largeur $H = (b-a)/m$.

$$I_{0,m}(f) = H \sum_{k=0}^{m-1} f(x_k). \tag{57}$$

On montre que (linéarité de l'intégrale, découpage en intervalle, et théorème de la moyenne discrète) :

$$E_{0,m}(f) = (b-a)\frac{H^2}{24} f''(\xi)$$

### 4.4.3 Formules du trapèze

On utilise cette fois-ci le polynôme de Lagrange interpolatoire de degré 1 avec les points aux bords de l'intervalle, i.e. $a$ et $b$. On a $x_0 = a, x_1 = b, \alpha_0 = \alpha_1 = (b-a)/2$.

$$I_1(f) = \frac{b-a}{2}(f(a) + f(b)). \tag{58}$$

On montre alors, si $h = (b-a)$ :

$$E_1(f) = -\frac{h^3}{12} f''(\xi)$$

On peut aussi décomposer l'intervalle en $m$ sous intervalles de taille $H$ et collationner les formules du trapèze. On voit que la formule se simplifie ainsi :

$$I_{1m},(f) = H\left(\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{m-1})\frac{1}{2}f(x_m)\right). \tag{59}$$

dont l'erreur est

$$E_{1,m}(f) = -(b-a)\frac{H^2}{12}f''(\xi).$$

### 4.4.4 Formules de Cavalieri-Simpson

On utilise cette fois le polynôme d'interpolation de degré 2 aux points $x_0 = a, x_1 = (a+b)/2, x_2 = b, \alpha_0 = \alpha_2 = (b-a)/6, \alpha_1 = 4(b-a)/6$. La formule est alors :

$$I_1(f) = \frac{b-a}{6}(f(a) + 4f((a+b)/2) + f(b)). \tag{60}$$

Si $f$ est $C^4$, l'erreur s'écrit :

$$E_2(f) = -\frac{h^5}{1440}f^{(4)}(\xi)$$

On peut aussi placer $2m+1$ points dans l'intervalle, avec $H = (b-a)/m$, on trouve

$$I_{2,m}(f) = \frac{H}{6}\left(f(x_0) + 2\sum_{r=1}^{m-1} f(x_{2r}) + 4\sum_{s=0}^{m-1} f(x_{2s+1}) + f(x_{2m})\right), \tag{61}$$

dont l'erreur est

$$E_{2,m}(f) = -\frac{b-a}{180}(H/2)^4 f^{(4)}(\xi).$$

On peut bien sûr étendre ces formules pour de polynômes de degré plus important, appelées alors formules de Newton-Cotes. Pour résumer :

| formule | nb de points | degré | exactitude | erreur |
|---|---|---|---|---|
| point milieu | 1 | 0 | 1 | $\frac{h^3}{24}f''(\xi)$ |
| trapèze | 2 | 1 | 1 | $-\frac{h^3}{12}f''(\xi)$ |
| simpson | 3 | 2 | 3 | $-\frac{h^5}{1440}f^{(4)}(\xi)$ |
| point milieu composite | m | 0 | 1 | $(b-a)\frac{H^2}{24}f''(\xi)$ |
| trapèze composite | m+1 | 1 | 1 | $-(b-a)\frac{H^2}{12}f''(\xi)$ |
| simpson composite | 2m+1 | 2 | 3 | $-\frac{b-a}{180}(H/2)^4 f^{(4)}(\xi)$ |

De façon assez suprenante, les formules du point milieu donnent de meilleurs résultats que les formules du trapèze, de degré supérieur. Cela est du à la symétrie des fonctions suffisamment lisses.

# 5 Applications à la résolution d'EDP

Ce chapitre présente quelques applications des outils précédents à la résolution numérique de problèmes classiques (type équations aux dérivées partielles dépendantes du temps).

## 5.1 Equation de la chaleur

On recherche une fonction $u = u(x,t)$ avec $x \in [0,1]$ et $t$ positif qui suit l'équation de la chaleur :

$$\partial_t u = \nu \partial^2_{xx} u + f, \quad 0 < x < 1, t > 0, \tag{62}$$

Avec des conditions aux limites $u(0,t) = u(1,t) = 0$ pour $t > 0$ (conditions type Dirichlet) et une donnée initiale $u(x,0) = u_0(x)$.

On montre que cette équation régit l'évolution de la température en espace et en temps dans une barre métallique avec une conductance thermique homogène $\nu$ et les extrémités maintenues à 0 degrés. La fonction $f$ décrit quant à elle l'apport de chaleur constant en fonction de la position (densité de chaleur linéique). Dans un problème physique, avec $u$ la température, d'autres termes comme la densité de masse ou la capacité calorifique interviendraient :

$$\partial_t u - \frac{\lambda}{\rho C_P} \partial^2_{xx} u = \frac{S}{\rho C_P}, \quad 0 < x < 1, t > 0, \tag{63}$$

avec $\rho$ la masse volumique, $C_P$ la chaleur spécifique à pression constante, $\lambda$ la conductivité thermique supposée indépendante de la température.

Analytiquement, on peut montrer que les solutions de cette équation sont liées aux coefficients $c_k := 2\int_0^1 u_0(x)\sin(k\pi x)dx$ des sinus de la série de Fourier de $u_0$ pour $f = 0$ et des conditions de Dirichlet. Pour les conditions de Neumann, ce serait les coefficients $d_k := 2\int_0^1 u_0(x)\cos(k\pi x)dx$ associés aux cosinus. On aurait donc pour Dirichlet ($u = 0$ aux bords)

$$u(x,t) = \sum_{k=1}^{\infty} c_k e^{-(k\pi)^2 t} \sin(k\pi x). \tag{64}$$

et pour Neumann ($\partial_x u(0,t) = \partial_x u(1,t) = 0$) :

$$u(x,t) = \frac{d_0}{2} + \sum_{k=1}^{\infty} d_k e^{-(k\pi)^2 t} \cos(k\pi x). \tag{65}$$

Les solutions ont donc une décroissance exponentielle en temps sans apport de chaleur.

**Discrétisation par différences finies.** On commence par une discrétisation seulement spatiale de $u$ sur l'intervalle $[0,1]$ par $n+1$ points $x_i$ distants de $h = \frac{1}{n}$. On note $u_i(t)$ une approximation de $u(x_i,t)$. Le schéma de Dirichlet s'écrit alors, avec une approximation de la dérivée seconde par différence finie centrée (cf (50)) :

$$\begin{aligned}
\partial_t u_i(t) &= \frac{\nu}{h^2}(u_{i-1}(t) - 2u_i(t) + u_{i+1}(t)) + f_i(t), \quad i = 1, \dots, n-1, \forall t > 0, \\
u_0(t) &= u_n(t) = 0, \quad \forall t > 0, \\
u_i(0) &= u_0(x_i).
\end{aligned}$$

On peut écrire ce système sous forme matricielle/vectorielle, en notant $\mathbf{u}(t)$ le vecteur colonne $[u_0(t), \dots, u_n(t)]^\intercal$, et en écrivant l'approximation de la dérivée seconde sous forme d'une matrice tridiagonale $\mathbf{A} := \frac{1}{h^2}[0 \dots, -1, 2, -1, \dots 0]$ d'ordre $n-1$.

$$\begin{aligned}
\partial_t \mathbf{u}(t) &= -\nu \mathbf{A}\mathbf{u}(t) + \mathbf{f}(t), \forall t > 0, \\
\mathbf{u}(0) &= \mathbf{u}_0.
\end{aligned}$$

On note que $\mathbf{A}$ est à diagonale dominante et définie positive ($\mathbf{x}^\intercal \mathbf{A}\mathbf{x} > 0$ grâce aux coefficients aux bords).

Il faut intégrer en temps les solutions de l'équation précédente. Le principe est d'utiliser des différences finies en temps sur le membre gauche ($t^k = \Delta t k$). Se pose ensuite la question de quel temps utiliser à droite : $t^k$, $t^{k+1}$ ou une combinaison des deux. On utilise souvent le $\theta$-schéma, qui combine $\theta$ du temps $t^{k+1}$ à $1-\theta$ du temps $t^k$, pour $\theta \in [0,1]$. Cela donne

$$\begin{aligned}\frac{\mathbf{u}^{k+1}-\mathbf{u}^k}{\Delta t} &= -\nu\mathbf{A}(\theta\mathbf{u}^{k+1}+(1-\theta)\mathbf{u}^k)+\theta\mathbf{f}^{k+1}+(1-\theta)\mathbf{f}^k, \text{pour } k=0,1,\dots\\ \mathbf{u}(0) &= \mathbf{u}_0.\end{aligned}$$

que l'on réécrit ainsi :

$$\begin{aligned}(\mathrm{Id}+\theta\nu\Delta t\mathbf{A})\mathbf{u}^{k+1} &= (\mathrm{Id}-(1-\theta)\nu\Delta t\mathbf{A})\mathbf{u}^k+\mathbf{g}^{k+1}, \text{pour } k=0,1,\dots\\ \mathbf{u}(0) &= \mathbf{u}_0.\end{aligned}$$

Si $\theta = 0$, il s'agit d'un système que l'on peut résoudre de manière explicite (*schéma d'Euler explicite*), tandis que si $\theta > 0$, il faut résoudre un système linéaire. Le schéma $\theta = 1$ correspond à la *méthode d'Euler implicite*. Pour $theta = 1/2$, on parle de *schéma de Crank-Nicholson*.

**Convergence de $theta = 0$.** Pour prouver la convergence du schéma d'Euler, il suffit de prouver sa stabilité car la méthode est consistante. On se place sous l'hypothèse $f = 0$. On obtient

$$\mathbf{u}^k = (\mathrm{Id}-\nu\Delta t\mathbf{A})^k\mathbf{u}^0.$$

Normalement, quelle que soit la donnée $\mathbf{u}^0$, il faut que la sortie $\mathbf{u}^k$ tende vers 0. Or $(\mathrm{Id}-\nu\Delta t\mathbf{A})^k$ tend vers 0 si sa plus grande valeur propre est plus petite que 1 en module, donc si

$$\rho(\mathrm{Id}-\nu\Delta t\mathbf{A}) < 1.$$

On prouve que les valeurs propres de $\mathbf{A}$ sont $\mu_j = \frac{4}{h^2}\sin^2(j\pi h/2)$, pour $j = 1,\dots,n-1$. D'où

$$\begin{aligned}&\rho(\mathrm{Id}-\nu\Delta t\mathbf{A}) < 1\\ \Leftrightarrow\ &\max_j|1-\nu\Delta t\mu_j| < 1\\ \Leftrightarrow\ &\forall j,(1-\nu\Delta t\mu_j)^2 < 1\\ \Leftrightarrow\ &\forall j,\nu^2\Delta t^2\mu_j^2 < 2\nu\Delta t\mu_j\\ \Leftrightarrow\ &\forall j,\nu\Delta t|\mu_j| < 2\\ \Leftarrow\ &\nu\Delta t\frac{4}{h^2} < 2\\ \Leftrightarrow\ &\Delta t < \frac{h^2}{2\nu}\end{aligned}$$

Le schéma est donc conditionnellement stable. Il ne faut pas des pas de temps trop importants en fonction du carré de l'espacement entre points, au risque d'avoir des oscillations.

**Convergence de $\theta = 1$.** On montre alors que

$$\mathbf{u}^k = ((\mathrm{Id}+\nu\Delta t\mathbf{A})^{-1})^k\mathbf{u}^0.$$

Or toutes les valeurs propres de $(\mathrm{Id}+\nu\Delta t\mathbf{A})^{-1}$ sont réelles et strictement inférieures à 1 pour tout $\Delta t$. Le schéma est dit inconditionnellement stable.

**Autres $\theta$-schémas.** On montre que les $\theta$-schémas sont conditionnellement stables pour $\theta < \frac{1}{2}$ et inconditionnellement stables pour $\theta > \frac{1}{2}$.

**Erreur d'approximation des $\theta$-schémas.** On montre que l'erreur de troncature locale du $\theta$-schéma est de l'ordre de $\Delta t + h^2$ si $\theta \neq 1/2$, et $\Delta t^2 + h^2$ si $\theta = 1/2$.

Ce dernier schéma est donc le meilleur pour optimiser stabilité et précision.

---

*Exercices*

1. Prouvez que les valeurs propres de $\mathbf{B} := (\mathrm{Id} + \nu \Delta t \mathbf{A})^{-1}$ sont $1/(1 + \nu \Delta t \mu_j)$. Utilisez le fait que $\mathbf{B}\mathbf{B}^{-1} = \mathrm{Id}$.
2. Dans le cas d'une grille de taille $m \times m$, quel serait l'opérateur $\mathbf{A}$ pour des conditions aux bords de Dirichlet ? Pour des conditions aux bords de Neumann ?

---

# A Quelques rappels d'algèbre linéaire

## A.1 Définitions élémentaires

On appelle *transposée* d'une matrice $\mathbf{A} \in \mathbb{R}^{m\times n}$ la matrice $n \times m$, notée $A^\intercal$, obtenue en échangeant les lignes et les colonnes de $\mathbf{A}$.

On a les relations : $(\mathbf{A}^\intercal)^\intercal = \mathbf{A}$, $(\mathbf{A}+\mathbf{B})^\intercal = \mathbf{A}^\intercal + \mathbf{B}^\intercal$, $(\mathbf{AB})^\intercal = \mathbf{B}^\intercal\mathbf{A}^\intercal$, $\alpha(\mathbf{A})^\intercal = \alpha\mathbf{A}^\intercal$ pour un scalaire $\alpha$. Si $\mathbf{A}$ est inversible, alors $(\mathbf{A}^\intercal)^{-1} = (\mathbf{A}^{-1})^\intercal$.

Une matrice (carrée) d'ordre $n$ est *symétrique* si $\mathbf{A} = \mathbf{A}^\intercal$ et antisymétrique si $\mathbf{A} = -\mathbf{A}^\intercal$. Elle est *orthogonale* si $\mathbf{A}\mathbf{A}^\intercal = \mathbf{A}^\intercal\mathbf{A} = \mathbf{I}$, autrement dit elle est son propre inverse.

## A.2 Matrices particulières

Les matrices *diagonales par blocs* sont des matrices de la forme $\mathbf{D} = \mathrm{diag}(\mathbf{D}_1, \ldots, \mathbf{D}_n)$ où les $\mathbf{D}_i$ sont des matrices carrées, de tailles éventuellement différentes. On note que $\det(\mathbf{D}) = \det(\mathbf{D}_1) \times \cdots \times \det(\mathbf{D}_n)$.

Une matrice $\mathbf{A}$, de taille $m \times n$, est *trapézoïdale supérieure* si $a_{ij} = 0$ pour $i > j$, et *trapézoïdale inférieure* si $a_{ij} = 0$ pour $i < j$. Si $m = n$, de telles matrices sont appelées *triangulaires supérieures* et *triangulaires inférieures* respectivement.

On note les propriétés importantes suivantes :

— le déterminant d'une matrice triangulaire est le produit de ses éléments diagonaux,
— l'inverse d'une matrice triangulaire inférieure est aussi triangulaire inférieure,
— le produit de deux matrices triangulaires inférieures est aussi triangulaire inférieure,
— le produit de deux matrices triangulaires inférieures avec des éléments égaux à 1 sur la diagonale est aussi triangulaire inférieure avec des éléments égaux à 1 sur la diagonale est aussi triangulaire inférieure.

Toutes ces propriétés sont vraies en substituant inférieure par supérieure.

## A.3 Valeurs propres et vecteurs propres

Soit $\mathbf{A}$ une matrice carré réelle (ou complexe). On dit que $\lambda \in \mathbb{C}$ est *valeur propre* de $\mathbf{A}$ s'il existe un vecteur non nul $\mathbf{x} \in \mathbf{C}^n$ tel que $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$.

Le vecteur $\mathbf{x}$ est alors appelé *vecteur propre* associé à $\mathbf{A}$. L'ensemble des valeurs propres de $\mathbf{A}$ forme le *spectre* de $\mathbf{A}$, noté $\sigma(\mathbf{A})$. Le plus grand module de valeur propre $\rho(\mathbf{A})$ est appelé *rayon spectral*.

On peut calculer une valeur propre $\lambda$ associé à un vecteur propre donné $\mathbf{x}$ via le *quotient de Rayleigh* :

$$\lambda = \frac{\mathbf{x}^*\mathbf{A}\mathbf{x}}{\mathbf{x}^*\mathbf{x}} \tag{66}$$

On note aussi que les valeurs propres sont solutions de l'équation

$$p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}) = 0. \tag{67}$$

$p_{\mathbf{A}}(\lambda)$ est appelé polynôme caractéristique de $\mathbf{A}$. Il est de degré $n$. Il existe donc $n$ valeurs propres dans $\mathbb{C}$, pas forcément distinctes.

On montre que $\det(\mathbf{A})$ est le produit des valeurs propres et que $\mathrm{tr}(A)$ est la somme des valeurs propres.

La plus grande valeur propre en module de $\mathbf{A}$ est appelé *rayon spectral* de $\mathbf{A}$, noté $\rho(\mathbf{A})$.

On montre facilement $\rho(\mathbf{A}^*) = \rho(\mathbf{A})$, $\rho(\alpha\mathbf{A}) = |\alpha|\rho(\mathbf{A})$, $\rho(\mathbf{A}^k) = (\rho(\mathbf{A}))^k$, pour $\alpha \in \mathbb{C}, k \in \mathbb{N}$.

## A.4 Normes matricielles

Soit $\|\cdot\|$ une norme vectorielle. On peut définir une norme matricielle associée à cette norme vectorielle ainsi : $\|\mathbf{A}\| = \sup_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}$. On l'appelle *norme matricielle subordonnée à la norme vectorielle* $\|\cdot\|$.

*Les normes vectorielles usuelles sont les p-normes. On obtient les normes matricielles subordonnées suivantes.*

*La 1-norme est la norme "somme des colonnes" :* $\|A\|_1 = \max_{j=1\ldots n} \sum_{i=1}^{m} |a_{ij}|$.

*La $\infty$-norme est la norme "somme des lignes" :* $\|A\|_\infty = \max_{i=1\dots m} \sum_{j=1}^{n} |a_{ij}|$.
*La* 2*-norme ou* norme spectrale *est associé à la plus grande valeur singulière de* $\mathbf{A}$.

# Références

*[QSS10] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri.* Numerical mathematics*, volume 37. Springer Science & Business Media, 2010.*