Modélisation Mathématique

Notes de Cours

Master M1 — 2025–2026

Yaë Ulrich Gaba

« Essentiellement, tous les modèles sont faux, mais certains sont utiles. »
— George E. P. Box

25 mars 2026

Table des matières

Préface

« Tous les modèles sont faux, mais certains sont utiles. »

— George E. P. Box

La modélisation mathématique est l'art et la science de traduire des phénomènes du monde réel en structures mathématiques rigoureuses, de les analyser pour en déduire des prédictions, et de confronter ces prédictions aux observations. Ce cours s'adresse aux étudiants de deuxième et troisième année de licence (L2/L3) en mathématiques, physique, biologie quantitative ou sciences de l'ingénieur, disposant d'une base solide en analyse, algèbre linéaire et équations différentielles.

Objectifs pédagogiques

À l'issue de ce cours, l'étudiant sera capable de :

1. **Formuler** un problème concret sous forme d'un modèle mathématique, en identifiant variables, paramètres et hypothèses.
2. **Analyser** qualitativement et quantitativement les équations obtenues : points d'équilibre, stabilité, bifurcations, comportement asymptotique.
3. **Simuler** numériquement les modèles à l'aide de Python (`scipy`, `numpy`, `matplotlib`).
4. **Valider** un modèle en le confrontant aux données expérimentales et en discutant ses limites.
5. **Communiquer** les résultats de manière claire et rigoureuse, tant par écrit que par des visualisations.

Philosophie du cours

Nous adoptons une démarche *du concret vers l'abstrait*. Chaque chapitre part d'un phénomène observable — croissance d'une population, propagation d'une épidémie, diffusion de la chaleur, fluctuation d'un prix — et construit pas à pas le modèle mathématique associé. Les outils théoriques (analyse dimensionnelle, théorie de la stabilité, processus stochastiques) sont introduits *au moment où ils deviennent nécessaires*, et non comme des prérequis abstraits.

Modèle ≠ Réalité

Un modèle est toujours une simplification. La citation de Box en exergue doit rester présente à l'esprit tout au long du cours. Nous insisterons systématiquement sur les **hypothèses** de chaque modèle et sur les conséquences de leur violation.

Structure du cours

Le cours est organisé en dix chapitres, suivant une progression logique :

Chapitre 1 — Le processus de modélisation. Méthodologie générale, étapes de la modélisation, classification des modèles.

Chapitre 2 — Analyse dimensionnelle et mise à l'échelle. Théorème de Buckingham Π, adimensionnement, ordres de grandeur.

Chapitre 3 — Dynamique des populations. Modèles de Malthus, Verhulst, Lotka–Volterra ; stabilité et portraits de phase.

Chapitre 4 — Modèles épidémiologiques. SIR, SEIR ; nombre de reproduction de base $\mathcal{R}_0$; stratégies de vaccination.

Chapitre 5 — Systèmes dynamiques discrets et chaos. Application logistique, exposants de Lyapunov, diagrammes de bifurcation.

Chapitre 6 — Modèles de diffusion et de transport. Équation de la chaleur, advection-diffusion, loi de Fick.

Chapitre 7 — Modèles économiques et financiers. Modèle de Solow, Black–Scholes, équilibre offre-demande.

Chapitre 8 — Optimisation et modèles variationnels. Programmation linéaire, conditions KKT, calcul des variations.

Chapitre 9 — Modèles stochastiques. Marches aléatoires, chaînes de Markov, mouvement brownien, EDS.

Chapitre 10 — Études de cas et projets. Applications intégratives couvrant plusieurs domaines.

Prérequis

- **Analyse :** suites, séries, calcul différentiel et intégral (une et plusieurs variables), équations différentielles ordinaires.
- **Algèbre linéaire :** espaces vectoriels, matrices, valeurs propres, diagonalisation.
- **Probabilités :** espaces probabilisés, variables aléatoires, espérance, variance, loi normale.
- **Informatique :** bases de Python ; les compléments nécessaires seront introduits au fil du cours.

Conventions et notations

- $\mathbb{R}$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{C}$: ensembles de nombres usuels.
- $\mathbb{E}[X]$, $\mathrm{Var}(X)$: espérance et variance d'une variable aléatoire X.
- $\|\cdot\|$, $|\cdot|$: norme et valeur absolue.
- Vecteurs en gras : $\mathbf{x}$, $\mathbf{v}$.
- Dérivées temporelles : $\dot{x} = dx/dt$.
- Grandeurs adimensionnées : $\tilde{x} = x/x_0$.

Organisation pratique

Chaque chapitre contient :

- Des **définitions**, **théorèmes** et **propositions** numérotés.
- Des **exemples détaillés** illustrant les concepts.
- Des encadrés **Formules clés**, **Attention**, **Bonne pratique** et **Intuition**.
- Des **simulations Python** complètes et commentées.
- Des **exercices** de difficulté croissante.

Logiciels

Bibliothèques Python utilisées

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint, solve_ivp
from scipy.optimize import minimize, linprog
from scipy.linalg import eig
```

Remerciements

Ce cours s'inspire des ouvrages de C. Lin et L. Segel (*Mathematics Applied to Deterministic Problems in the Natural Sciences*), J. D. Murray (*Mathematical Biology*), S. H. Strogatz (*Nonlinear Dynamics and Chaos*) et E. A. Bender (*An Introduction to Mathematical Modeling*).

Bonne lecture et bonne modélisation !

Chapitre 1

Le Processus de Modélisation

Galilée écrivait que le livre de la nature est écrit en langage mathématique. Mais entre la nature et les équations, il y a un pont à construire : c'est le *modèle*. Un modèle est une simplification délibérée de la réalité, qui capture les mécanismes essentiels tout en ignorant les détails superflus. L'art de la modélisation — car c'est bien un art autant qu'une science — consiste à choisir le bon niveau d'abstraction : assez de détails pour que le modèle soit utile, assez de simplicité pour qu'il soit traitable.

« La science est faite de données comme une maison est faite de pierres ; mais une accumulation de données n'est pas plus une science qu'un tas de pierres n'est une maison. »

— Henri Poincaré

Ce premier chapitre présente la démarche générale de modélisation mathématique. Nous décrivons les étapes clés, de l'observation du phénomène à la validation du modèle, et nous classifions les grandes familles de modèles mathématiques.

1.1 Qu'est-ce qu'un modèle mathématique ?

Définition 1.1 (Modèle mathématique)**.** Un **modèle mathématique** est une représentation formelle d'un système réel (physique, biologique, économique, etc.) à l'aide d'objets mathématiques (équations, inégalités, graphes, distributions de probabilité) qui capturent les relations essentielles entre les grandeurs du système.

Remarque 1.2. Un modèle n'est jamais unique : le même phénomène peut être décrit par différents modèles, de niveaux de complexité variés. Le choix du modèle dépend de la question posée et des données disponibles.

1.2 Les étapes du processus de modélisation

Le processus de modélisation se décompose en cinq grandes étapes, formant un cycle itératif.

1. **Observation et identification du problème.** On délimite le phénomène à étudier, on identifie les grandeurs pertinentes (variables d'état, paramètres, entrées/sorties) et on rassemble les données disponibles.

2. **Formulation des hypothèses.** On énonce explicitement les simplifications adoptées : quelles interactions sont négligées, quelles grandeurs sont supposées constantes, quel est le domaine de validité spatial et temporel.

3. **Construction du modèle mathématique.** On traduit les hypothèses en équations. Cela peut donner :

   - une équation différentielle ordinaire (EDO),
   - une équation aux dérivées partielles (EDP),
   - un système d'équations algébriques,
   - un programme d'optimisation,
   - un modèle stochastique.

4. **Résolution et analyse.** On résout le modèle analytiquement (quand c'est possible) ou numériquement. On étudie le comportement qualitatif : existence et unicité des solutions, stabilité des équilibres, dépendance aux paramètres.

5. **Validation et interprétation.** On compare les prédictions du modèle aux données expérimentales. Si l'accord est insuffisant, on revient à l'étape 2 pour modifier les hypothèses.

Le cycle de modélisation

La modélisation n'est pas un processus linéaire mais un **cycle**. Après la validation, on revient souvent raffiner les hypothèses, enrichir le modèle ou en réduire la complexité. Un bon modélisateur effectue plusieurs tours de ce cycle.

1.3 Classification des modèles

Définition 1.3 (Types de modèles)**.** On distingue les modèles selon plusieurs critères :

1. **Déterministe vs. stochastique.** Un modèle déterministe prédit une trajectoire unique pour des conditions initiales données ; un modèle stochastique incorpore de l'aléatoire.

2. **Continu vs. discret.** Le temps et/ou l'espace peuvent varier de façon continue ou discrète.

3. **Statique vs. dynamique.** Un modèle statique décrit un état d'équilibre ; un modèle dynamique décrit l'évolution temporelle.

4. **Linéaire vs. non linéaire.** La linéarité des relations entre variables conditionne fortement les méthodes d'analyse.

5. **Paramétrique vs. non paramétrique.** Selon que la structure du modèle est fixée a priori ou déduite des données.

Exemple 1.4. Considérons la chute d'un objet. En négligeant la résistance de l'air, le modèle est :

$$m\ddot{y} = -mg, \qquad y(0) = h, \quad \dot{y}(0) = 0.$$

C'est un modèle déterministe, continu, dynamique et linéaire. Sa solution est $y(t) = h - \frac{1}{2}gt^2$.

Si l'on ajoute la résistance de l'air proportionnelle à $\dot{y}^2$, on obtient un modèle **non linéaire** :

$$m\ddot{y} = -mg + k\dot{y}^2.$$

1.4 Principes fondamentaux de construction

1.4.1 Lois de conservation

De nombreux modèles reposent sur des **lois de conservation** : conservation de la masse, de l'énergie, de la quantité de mouvement, ou du nombre d'individus.

Théorème 1.5 (Bilan générique)**.** *Pour une quantité $Q(t)$ contenue dans un domaine Ω, la loi de conservation générique s'écrit :*

$$\frac{dQ}{dt} = \textit{(flux entrant)} - \textit{(flux sortant)} + \textit{(source)} - \textit{(puits)}.$$

Exemple 1.6. Soit $N(t)$ la taille d'une population. Si le taux de naissance est b et le taux de mortalité est d (par individu par unité de temps), la loi de conservation donne :

$$\frac{dN}{dt} = bN - dN = (b - d)N.$$

C'est le modèle de Malthus, que nous étudierons en détail au chapitre 3.

1.4.2 Lois constitutives

Les lois constitutives relient les flux aux grandeurs d'état.

Exemple 1.7.

- **Loi de Fourier :** le flux de chaleur est proportionnel au gradient de température : $\mathbf{q} = -\kappa \nabla T$.
- **Loi de Fick :** le flux de diffusion est proportionnel au gradient de concentration : $\mathbf{J} = -D\nabla c$.
- **Loi de Hooke :** la contrainte est proportionnelle à la déformation : $\sigma = E\varepsilon$.

1.4.3 Principes variationnels

Certains modèles sont obtenus en minimisant (ou en rendant stationnaire) une fonctionnelle.

Définition 1.8 (Fonctionnelle d'action)**.** En mécanique, le principe de moindre action énonce que la trajectoire réelle rend stationnaire la fonctionnelle

$$\mathcal{S}[q] = \int_{t_1}^{t_2} L(q, \dot{q}, t)\, dt,$$

où L est le lagrangien du système. Les équations d'Euler–Lagrange qui en découlent seront étudiées au chapitre 8.

1.5 Qualités d'un bon modèle

Critères de qualité

Un bon modèle satisfait un compromis entre :

- **Simplicité** (parcimonie) : le nombre de paramètres doit rester raisonnable (principe du rasoir d'Occam).
- **Précision** : les prédictions doivent être suffisamment proches des observations.
- **Robustesse** : de petites variations des paramètres ne doivent pas entraîner de changements dramatiques des prédictions.
- **Interprétabilité** : les paramètres doivent avoir une signification physique ou biologique.

Proposition 1.9 (Principe de parcimonie)**.** Étant donnés deux modèles de précision comparable, on préfère celui qui possède le moins de paramètres libres. Ce principe est formalisé par des critères d'information tels que l'AIC (Akaike Information Criterion) :

$$\mathrm{AIC} = 2k - 2\ln \hat{L},$$

où k est le nombre de paramètres et $\hat{L}$ la vraisemblance maximale.

1.6 Sensibilité et incertitude

Définition 1.10 (Analyse de sensibilité)**.** L'analyse de sensibilité étudie comment la sortie y d'un modèle varie lorsqu'un paramètre p change. L'**indice de sensibilité locale** est défini par :

$$S_p = \frac{p}{y}\frac{\partial y}{\partial p}.$$

Un indice $|S_p| \gg 1$ signale que le modèle est très sensible au paramètre p.

Exemple 1.11. Pour le modèle exponentiel $y(t) = y_0 e^{rt}$, on a

$$S_r = \frac{r}{y}\frac{\partial y}{\partial r} = \frac{r}{y_0 e^{rt}}\, y_0 t e^{rt} = rt.$$

La sensibilité au taux r croît linéairement avec le temps : de petites erreurs sur r se traduisent par de grandes erreurs sur y à long terme.

Analyse de sensibilité en Python

```
import numpy as np
import matplotlib.pyplot as plt

def model(t, r, y0=1.0):
    return y0 * np.exp(r * t)
```

```
t = np.linspace(0, 10, 200)
r_nom = 0.5
dr = 0.05  # perturbation de 10%

y_nom = model(t, r_nom)
y_plus = model(t, r_nom + dr)
y_minus = model(t, r_nom - dr)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

ax1.plot(t, y_nom, 'b-', label=f'$r = {r_nom}$')
ax1.fill_between(t, y_minus, y_plus, alpha=0.3,
                 label=f'$r = {r_nom} \\pm {dr}$')
ax1.set_xlabel('$t$'); ax1.set_ylabel('$y(t)$')
ax1.legend(); ax1.set_title('Solutions')

S_r = r_nom * t  # indice de sensibilité
ax2.plot(t, S_r, 'r-')
ax2.set_xlabel('$t$'); ax2.set_ylabel('$S_r$')
ax2.set_title('Indice de sensibilité $S_r = rt$')
plt.tight_layout(); plt.savefig('sensibilite.pdf')
```

1.7 Ajustement aux données

Une fois le modèle construit, il faut en estimer les paramètres à partir des données.

Définition 1.12 (Moindres carrés)**.** Étant données des observations (t_i, y_i^{obs}), $i = 1, \ldots, n$, et un modèle $y(t; \mathbf{p})$ dépendant d'un vecteur de paramètres $\mathbf{p}$, l'estimateur des **moindres carrés** est :

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_{i=1}^{n} \big(y_i^{\text{obs}} - y(t_i; \mathbf{p})\big)^2.$$

Ajustement par moindres carrés

```
import numpy as np
from scipy.optimize import curve_fit

# Données synthétiques : croissance exponentielle bruitée
np.random.seed(42)
t_data = np.linspace(0, 5, 20)
r_true, y0_true = 0.8, 2.0
y_data = y0_true * np.exp(r_true * t_data) \
         + np.random.normal(0, 1, len(t_data))

def modele(t, y0, r):
    return y0 * np.exp(r * t)

popt, pcov = curve_fit(modele, t_data, y_data, p0=[1, 0.5])
```

```
print(f"Paramètres estimés : y0 = {popt[0]:.3f}, r = {popt[1]:.3f}")
print(f"Écarts-types : {np.sqrt(np.diag(pcov))}")
```

1.8 Exemple complet : refroidissement d'une tasse de café

Illustrons le processus complet sur un exemple classique.

Étape 1 : Observation. Une tasse de café chaude posée sur une table refroidit progressivement. On mesure la température $T(t)$ au cours du temps.

Étape 2 : Hypothèses.

- La température du café est homogène (modèle à paramètres groupés).
- La température ambiante T_a est constante.
- Le flux de chaleur est proportionnel à l'écart de température (loi de Newton du refroidissement).

Étape 3 : Modèle. La loi de Newton donne :

$$\frac{dT}{dt} = -k(T - T_a), \qquad T(0) = T_0,$$

où $k > 0$ est le coefficient de transfert thermique.

Étape 4 : Résolution. L'EDO est linéaire du premier ordre. Sa solution est :

$$T(t) = T_a + (T_0 - T_a)\, e^{-kt}.$$

Loi de Newton du refroidissement

$$T(t) = T_a + (T_0 - T_a)\, e^{-kt}, \qquad k > 0.$$

- T_0 : température initiale.
- T_a : température ambiante.
- k : constante de refroidissement (en s^{-1}).
- Temps de demi-refroidissement : $t_{1/2} = \ln 2 / k$.

Étape 5 : Validation.

Simulation et ajustement — refroidissement

```
import numpy as np
```

```python
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Données expérimentales (simulées)
np.random.seed(0)
t_exp = np.arange(0, 31, 2)  # minutes
T_a, T0_true, k_true = 22.0, 90.0, 0.07
T_exp = T_a + (T0_true - T_a) * np.exp(-k_true * t_exp) \
        + np.random.normal(0, 1.5, len(t_exp))

def newton_cooling(t, T0, k):
    return T_a + (T0 - T_a) * np.exp(-k * t)

popt, pcov = curve_fit(newton_cooling, t_exp, T_exp, p0=[85, 0.05])
print(f"T0 = {popt[0]:.1f} °C,  k = {popt[1]:.4f} /min")

t_fine = np.linspace(0, 30, 300)
plt.figure(figsize=(8, 4))
plt.scatter(t_exp, T_exp, c='red', label='Données')
plt.plot(t_fine, newton_cooling(t_fine, *popt), 'b-',
         label='Modèle ajusté')
plt.axhline(T_a, color='gray', ls='--', label=f'$T_a = {T_a}$ °C')
plt.xlabel('Temps (min)'); plt.ylabel('Température (°C)')
plt.legend(); plt.title('Refroidissement de Newton')
plt.tight_layout(); plt.savefig('refroidissement.pdf')
```

1.9 Limites et pièges courants

Pièges à éviter

1. **Sur-ajustement (overfitting).** Un modèle avec trop de paramètres peut s'ajuster parfaitement aux données d'entraînement mais échouer en prédiction.

2. **Extrapolation abusive.** Un modèle validé sur un intervalle $[0, T]$ ne doit pas être extrapolé sans précaution à $t \gg T$.

3. **Confusion corrélation/causalité.** Un bon ajustement ne prouve pas un lien causal.

4. **Oubli des unités.** Ne jamais additionner des grandeurs de dimensions différentes (voir chapitre 2).

1.10 Exercices

Exercice 1.1. Un réservoir contient initialement $V_0 = 1000$ L d'eau pure. On y verse une solution salée de concentration $c_{\text{in}} = 30$ g/L au débit $q_{\text{in}} = 5$ L/min. Le mélange, supposé homogène, sort au débit $q_{\text{out}} = 5$ L/min.

1. Écrire l'EDO vérifiée par la quantité de sel $Q(t)$ dans le réservoir.
2. Résoudre l'EDO et déterminer la concentration à l'équilibre.
3. Tracer $Q(t)$ pour $t \in [0, 400]$ min en Python.
4. Discuter ce qui change si $q_{\text{out}} \neq q_{\text{in}}$.

Exercice 1.2. On considère le modèle $y(t) = A\sin(\omega t + \varphi)$ pour décrire un signal périodique.

1. Générer des données synthétiques bruitées avec $A = 3$, $\omega = 2\pi/5$, $\varphi = \pi/4$ et un bruit gaussien d'écart-type 0,5.
2. Ajuster le modèle par moindres carrés.
3. Calculer les intervalles de confiance à 95 % pour A, ω et φ.

Exercice 1.3. Pour le modèle de refroidissement de Newton :

1. Montrer que la température $T(t)$ est strictement décroissante si $T_0 > T_a$ et strictement croissante si $T_0 < T_a$.
2. Calculer l'indice de sensibilité S_k et interpréter.
3. Modifier le modèle pour tenir compte du rayonnement (loi de Stefan–Boltzmann : flux $\propto T^4 - T_a^4$). Résoudre numériquement et comparer.

Exercice 1.4. On observe la décroissance d'une substance radioactive. Le nombre d'atomes $N(t)$ satisfait $dN/dt = -\lambda N$.

1. Montrer que la demi-vie est $t_{1/2} = \ln 2/\lambda$.
2. Écrire un programme Python qui, à partir de données (t_i, N_i), estime λ par moindres carrés et par régression linéaire sur $\ln N$.
3. Comparer les deux méthodes et discuter leurs avantages.

Chapitre 2

Analyse Dimensionnelle et Mise à l'Échelle

En 1883, Osborne Reynolds publie une étude sur l'écoulement de fluides dans des tuyaux. Il découvre que le passage du régime laminaire au régime turbulent ne dépend pas séparément de la vitesse, du diamètre ou de la viscosité, mais d'un unique *nombre sans dimension* — le nombre de Reynolds. Cette observation illustre le pouvoir de l'*analyse dimensionnelle* : en identifiant les combinaisons adimensionnelles des paramètres d'un problème, on réduit drastiquement sa complexité.

2.1 Introduction

L'analyse dimensionnelle est un outil puissant qui exploite la cohérence des unités physiques pour simplifier les modèles, réduire le nombre de paramètres et découvrir des lois d'échelle. Ce chapitre présente le théorème de Buckingham Π, les techniques d'adimensionnement et leurs applications.

2.2 Dimensions et unités

Définition 2.1 (Dimension physique)**.** Toute grandeur physique possède une **dimension** exprimée en fonction des dimensions fondamentales. Dans le système SI, les dimensions de base sont :

Grandeur	Symbole dimensionnel	Unité SI
Longueur	L	m
Masse	M	kg
Temps	T	s
Température	Θ	K
Quantité de matière	N	mol
Intensité électrique	I	A

Définition 2.2 (Homogénéité dimensionnelle)**.** Une équation physique est dite **dimensionnellement homogène** si chaque terme a la même dimension. C'est une condition nécessaire (mais non suffisante) de validité.

Exemple 2.3. La vitesse v a pour dimension $[v] = LT^{-1}$. L'énergie cinétique $E_c = \frac{1}{2}mv^2$ a pour dimension $[E_c] = ML^2T^{-2}$, qui est bien celle d'une énergie.

Remarque 2.4. Les arguments de fonctions transcendantes (exp, ln, sin, etc.) doivent être sans dimension. Ainsi, dans $e^{-t/\tau}$, le rapport t/τ est adimensionnel.

2.3 Le théorème de Buckingham Π

Théorème 2.5 (Buckingham Π)**.** *Soit une relation physique faisant intervenir n grandeurs $q_1, \ldots, q_n$ dont les dimensions s'expriment à l'aide de k dimensions fondamentales indépendantes. Alors la relation peut s'écrire sous la forme d'une équation entre $n - k$ groupes adimensionnels $\Pi_1, \ldots, \Pi_{n-k}$:*

$$f(\Pi_1, \Pi_2, \ldots, \Pi_{n-k}) = 0.$$

Remarque 2.6. Le nombre k est le rang de la **matrice dimensionnelle**, c'est-à-dire la matrice dont les colonnes contiennent les exposants des dimensions fondamentales pour chaque grandeur.

2.3.1 Méthode pratique

1. Lister toutes les n grandeurs intervenant dans le problème.

2. Construire la matrice dimensionnelle et calculer son rang k.

3. Former $n - k$ groupes adimensionnels Π_j en combinant les grandeurs.

4. Écrire la loi physique sous forme $\Pi_1 = \phi(\Pi_2, \ldots, \Pi_{n-k})$.

Exemple 2.7 (Période d'un pendule simple)**.** La période τ dépend de la longueur ℓ, de la masse m, de l'accélération gravitationnelle g et de l'amplitude θ_0 (sans dimension). Les grandeurs dimensionnées sont τ, ℓ, m, g ($n = 4$). Les dimensions fondamentales sont M, L, T ($k = 3$). Matrice dimensionnelle :

$$\begin{pmatrix} & \tau & \ell & m & g \\ M & 0 & 0 & 1 & 0 \\ L & 0 & 1 & 0 & 1 \\ T & 1 & 0 & 0 & -2 \end{pmatrix}$$

Le rang est $k = 3$ (mais la colonne de m est linéairement indépendante et m apparaît seule dans M, donc m disparaît). On obtient $n - k = 1$ groupe :

$$\Pi_1 = \tau\sqrt{g/\ell}.$$

Conclusion : $\tau = C(\theta_0)\sqrt{\ell/g}$ pour une certaine fonction C. Pour de petites oscillations, $C \approx 2\pi$. La masse n'intervient pas.

2.4 Adimensionnement des équations

Définition 2.8 (Adimensionnement)**. Adimensionner** une équation consiste à introduire des variables sans dimension en divisant chaque grandeur par une échelle caractéristique :

$$\tilde{x} = \frac{x}{x_c}, \qquad \tilde{t} = \frac{t}{t_c}, \qquad \tilde{u} = \frac{u}{u_c}.$$

L'équation adimensionnée ne fait plus apparaître que des **nombres sans dimension** qui révèlent l'importance relative des différents termes.

Exemple 2.9 (Équation de l'oscillateur harmonique amorti)**.** L'équation

$$m\ddot{x} + c\dot{x} + kx = 0$$

fait intervenir m (masse), c (amortissement), k (raideur). Posons x_c une amplitude caractéristique et $t_c = \sqrt{m/k}$ (période naturelle). Avec $\tilde{x} = x/x_c$, $\tilde{t} = t/t_c$:

$$\frac{d^2\tilde{x}}{d\tilde{t}^2} + 2\zeta\,\frac{d\tilde{x}}{d\tilde{t}} + \tilde{x} = 0, \qquad \zeta = \frac{c}{2\sqrt{mk}}.$$

Le seul paramètre restant est le **facteur d'amortissement** ζ, qui contrôle qualitativement le comportement :

- $\zeta < 1$: oscillations amorties (sous-critique),
- $\zeta = 1$: amortissement critique,
- $\zeta > 1$: régime sur-critique (pas d'oscillation).

Nombres adimensionnels classiques

Nom	Expression	Signification
Reynolds	$\mathrm{Re} = \rho v L/\mu$	inertie / viscosité
Péclet	$\mathrm{Pe} = vL/D$	advection / diffusion
Mach	$\mathrm{Ma} = v/c_s$	vitesse / son
Froude	$\mathrm{Fr} = v/\sqrt{gL}$	inertie / gravité
Damköhler	$\mathrm{Da} = \tau_{\text{transport}}/\tau_{\text{réaction}}$	transport / réaction

2.5 Lois d'échelle et similitude

Définition 2.10 (Similitude)**.** Deux systèmes physiques sont dits en **similitude** s'ils partagent les mêmes nombres adimensionnels. L'étude d'un modèle réduit (maquette) peut alors être transposée au système réel.

Théorème 2.11 (Invariance d'échelle)**.** *Si une grandeur y dépend de variables $x_1, \dots, x_n$ et si la relation est une loi de puissance :*

$$y = C\, x_1^{a_1}\, x_2^{a_2} \cdots x_n^{a_n},$$

alors cette loi est invariante par changement d'unités si et seulement si les exposants vérifient les contraintes dimensionnelles.

Exemple 2.12 (Loi de traînée)**.** La force de traînée F_D sur un objet dépend de ρ (densité du fluide), v (vitesse), L (taille caractéristique) et μ (viscosité). On a $n = 5$, $k = 3$ (M, L, T), donc $n - k = 2$ groupes :

$$\Pi_1 = \frac{F_D}{\rho v^2 L^2}, \qquad \Pi_2 = \frac{\rho v L}{\mu} = \mathrm{Re}.$$

Ainsi $C_D = F_D/(\frac{1}{2}\rho v^2 A)$ est une fonction de Re seulement (pour une géométrie donnée).

2.6 Application : équation de diffusion

Considérons l'équation de diffusion en une dimension :

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \qquad u(x, 0) = u_0(x),$$

sur un domaine de longueur L. Posons $\tilde{x} = x/L$, $\tilde{t} = Dt/L^2$, $\tilde{u} = u/U$ (où U est une échelle de concentration). L'équation devient :

$$\frac{\partial \tilde{u}}{\partial \tilde{t}} = \frac{\partial^2 \tilde{u}}{\partial \tilde{x}^2}.$$

Tous les paramètres ont disparu : le temps caractéristique de diffusion est $t_c = L^2/D$.

Temps de diffusion en Python

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Diffusion 1D par méthode des lignes
L = 1.0    # longueur du domaine
D = 0.01   # coefficient de diffusion
N = 100    # nombre de points
dx = L / (N + 1)
x = np.linspace(dx, L - dx, N)

# Condition initiale : pic gaussien
u0 = np.exp(-((x - 0.5)**2) / (2 * 0.02**2))

def rhs(t, u):
    """Laplacien discret avec conditions de Dirichlet nulles."""
    d2u = np.zeros_like(u)
    d2u[0] = (0 - 2*u[0] + u[1]) / dx**2
    d2u[-1] = (u[-2] - 2*u[-1] + 0) / dx**2
    d2u[1:-1] = (u[:-2] - 2*u[1:-1] + u[2:]) / dx**2
    return D * d2u

sol = solve_ivp(rhs, [0, 5], u0, t_eval=[0, 0.5, 1, 2, 5],
                method='RK45', max_step=0.01)

plt.figure(figsize=(8, 5))
```

```
for i, t_val in enumerate(sol.t):
    plt.plot(x, sol.y[:, i], label=f'$t = {t_val}$')
plt.xlabel('$x$'); plt.ylabel('$u(x,t)$')
plt.legend(); plt.title('Diffusion 1D')
plt.tight_layout(); plt.savefig('diffusion_1d.pdf')
```

2.7 Perturbations régulières et ordres de grandeur

L'adimensionnement révèle souvent un petit paramètre $\varepsilon \ll 1$ qui permet une **approximation perturbative**.

Définition 2.13 (Développement perturbatif régulier)**.** On cherche la solution sous la forme

$$\tilde{u} = \tilde{u}_0 + \varepsilon\, \tilde{u}_1 + \varepsilon^2\, \tilde{u}_2 + \cdots$$

et on identifie les termes ordre par ordre.

Exemple 2.14. L'oscillateur faiblement non linéaire :

$$\ddot{x} + x + \varepsilon x^3 = 0, \qquad 0 < \varepsilon \ll 1.$$

À l'ordre zéro : $\ddot{x}_0 + x_0 = 0 \Rightarrow x_0 = A\cos t$. À l'ordre un :

$$\ddot{x}_1 + x_1 = -x_0^3 = -A^3\cos^3 t = -\frac{A^3}{4}(3\cos t + \cos 3t).$$

Le terme en $\cos t$ est résonnant (terme séculaire). Cela signale que le développement naïf échoue à long terme ; il faut utiliser la méthode des échelles multiples ou la méthode de Lindstedt–Poincaré.

Termes séculaires

Un développement perturbatif régulier peut contenir des termes qui croissent sans borne avec le temps (termes séculaires). Cela invalide l'approximation sur des temps longs et nécessite des techniques spécialisées.

2.8 Analyse d'ordres de grandeur

Proposition 2.15 (Simplification par ordres de grandeur)**.** Après adimensionnement, si un terme $\varepsilon\, f(\tilde{x})$ est multiplié par $\varepsilon \ll 1$ tandis que les autres termes sont d'ordre $O(1)$, on peut négliger ce terme en première approximation.

Exemple 2.16 (Écoulement à bas Reynolds)**.** Pour un fluide newtonien incompressible, les équations de Navier–Stokes adimensionnées contiennent le terme $\frac{1}{\mathrm{Re}}\nabla^2\mathbf{v}$. Si $\mathrm{Re} \gg 1$, la viscosité est négligeable (écoulement d'Euler). Si $\mathrm{Re} \ll 1$, l'inertie est négligeable (écoulement de Stokes) :

$$\nabla p = \mu \nabla^2 \mathbf{v}.$$

2.9 Exercices

Exercice 2.1. En utilisant l'analyse dimensionnelle, déterminer la dépendance de la fréquence f d'oscillation d'une goutte sphérique en fonction de sa masse m, de son rayon R et de la tension de surface γ.

Exercice 2.2. Adimensionner l'équation de Lotka–Volterra :

$$\dot{x} = ax - bxy, \qquad \dot{y} = -cy + dxy,$$

en posant $\tilde{x} = bx/c$, $\tilde{y} = dy/a$, $\tilde{t} = at$. Vérifier qu'on obtient un système ne dépendant que d'un seul paramètre $\alpha = c/a$.

Exercice 2.3. La vitesse de sédimentation v_s d'une sphère de rayon R et de masse volumique ρ_s dans un fluide de viscosité μ et de masse volumique ρ_f sous gravité g est donnée par la formule de Stokes.

1. Retrouver la dépendance $v_s \propto R^2$ par analyse dimensionnelle.
2. Pour quelles valeurs de Re cette formule est-elle valide ?

Exercice 2.4. Considérons l'équation de la chaleur avec source :

$$\rho c_p \frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} + Q_0 e^{-x/\ell}.$$

1. Identifier les échelles caractéristiques x_c, t_c, T_c.
2. Adimensionner l'équation et identifier les nombres sans dimension.
3. Discuter les régimes limites.

Exercice 2.5. En utilisant le théorème Π de Buckingham, montrer que la perte de charge Δp dans un tuyau de longueur L et de diamètre d, pour un fluide de masse volumique ρ et de viscosité μ s'écoulant à la vitesse v, s'écrit :

$$\frac{\Delta p}{\rho v^2} = \phi\left(\mathrm{Re},\, \frac{L}{d}\right).$$

Chapitre 3

Modèles de Dynamique des Populations

3.1 Introduction

En 1798, le révérend Thomas Malthus publie son célèbre *Essai sur le principe de population*, où il affirme que la population croît géométriquement tandis que les ressources ne croissent qu'arithmétiquement. Cette vision pessimiste — la « catastrophe malthusienne »— allait inspirer Darwin et devenir le premier modèle mathématique de dynamique des populations : l'exponentielle. Mais Malthus avait tort sur un point crucial : la croissance ne peut pas être exponentielle indéfiniment. En 1838, Pierre-François Verhulst introduit un terme de saturation qui mène à la célèbre équation logistique. Puis, dans les années 1920, Alfred Lotka et Vito Volterra, travaillant indépendamment — l'un sur des réactions chimiques, l'autre sur la pêche en Adriatique — découvrent le modèle proie-prédateur qui porte leurs noms.

Ce chapitre retrace cette aventure intellectuelle et développe les modèles fondamentaux — Malthus, Verhulst, Lotka–Volterra — en insistant sur la dérivation à partir de principes premiers, l'analyse qualitative (portraits de phase, stabilité) et les simulations numériques.

3.2 Croissance exponentielle : le modèle de Malthus

Définition 3.1 (Modèle de Malthus)**.** Soit $N(t)$ la taille d'une population au temps t. Si le taux de naissance b et le taux de mortalité d sont constants et proportionnels à la taille de la population :

$$\frac{dN}{dt} = rN, \qquad r = b - d, \qquad N(0) = N_0.$$

Théorème 3.2 (Solution du modèle de Malthus)**.** *La solution est* $N(t) = N_0\, e^{rt}$.

- *Si* $r > 0$ *: croissance exponentielle (explosion).*
- *Si* $r = 0$ *: population constante.*
- *Si* $r < 0$ *: décroissance exponentielle (extinction).*

Remarque 3.3. Le modèle de Malthus est réaliste uniquement sur des périodes courtes. Aucune population réelle ne peut croître exponentiellement indéfiniment.

Modèle de Malthus en Python

```python
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 10, 200)
N0 = 100

for r in [-0.3, 0, 0.2, 0.5]:
    N = N0 * np.exp(r * t)
    plt.plot(t, N, label=f'$r = {r}$')

plt.xlabel('Temps $t$'); plt.ylabel('Population $N(t)$')
plt.legend(); plt.title('Modèle de Malthus')
plt.ylim(0, 2000)
plt.tight_layout(); plt.savefig('malthus.pdf')
```

3.3 Croissance logistique : le modèle de Verhulst

Définition 3.4 (Modèle logistique de Verhulst)**.** Pour modéliser la limitation des ressources, Verhulst (1838) propose :

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right), \qquad N(0) = N_0,$$

où $K > 0$ est la **capacité de charge** de l'environnement.

Interprétation du terme logistique

Le facteur $(1 - N/K)$ modélise la compétition intraspécifique : quand $N \ll K$, la croissance est presque exponentielle ; quand N approche K, le taux de croissance effectif tend vers zéro.

Théorème 3.5 (Solution de l'équation logistique)**.** *L'EDO logistique est séparable. Sa solution est :*

$$N(t) = \frac{K}{1 + \left(\frac{K}{N_0} - 1\right)e^{-rt}}.$$

On a $\lim_{t\to+\infty} N(t) = K$ *pour tout* $N_0 > 0$.

Démonstration. On sépare les variables :

$$\frac{dN}{N(1 - N/K)} = r\,dt.$$

Décomposition en éléments simples :

$$\frac{1}{N(1-N/K)} = \frac{1}{N} + \frac{1/K}{1 - N/K}.$$

Intégration :

$$\ln N - \ln(1 - N/K) = rt + C.$$

En posant $C = \ln \frac{N_0}{1 - N_0/K}$, on obtient le résultat. □

3.3.1 Analyse qualitative

Les points d'équilibre sont les solutions de $rN(1 - N/K) = 0$: $N^* = 0$ et $N^* = K$.

Proposition 3.6 (Stabilité des équilibres logistiques)**.**

- $N^* = 0$ est **instable** (pour $r > 0$).
- $N^* = K$ est **asymptotiquement stable**.

On le vérifie par linéarisation : $f'(N) = r(1 - 2N/K)$, d'où $f'(0) = r > 0$ (instable) et $f'(K) = -r < 0$ (stable).

Croissance logistique et portrait de phase

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

r, K = 0.5, 1000

def logistic(N, t):
    return r * N * (1 - N / K)

t = np.linspace(0, 30, 500)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

for N0 in [10, 50, 200, 500, 1200, 1500]:
    N = odeint(logistic, N0, t).flatten()
    ax1.plot(t, N, label=f'$N_0={N0}$')

ax1.axhline(K, color='gray', ls='--', label=f'$K={K}$')
ax1.set_xlabel('$t$'); ax1.set_ylabel('$N(t)$')
ax1.legend(fontsize=8); ax1.set_title('Trajectoires')

# Portrait de phase (champ de direction 1D)
N_vals = np.linspace(0, 1600, 300)
dNdt = r * N_vals * (1 - N_vals / K)
ax2.plot(N_vals, dNdt, 'b-')
ax2.axhline(0, color='gray', ls='--')
ax2.plot([0, K], [0, 0], 'ro', markersize=8)
ax2.set_xlabel('$N$'); ax2.set_ylabel("$dN/dt$")
ax2.set_title('Champ de vitesse')
plt.tight_layout(); plt.savefig('logistique.pdf')
```

3.4 Le modèle proie–prédateur de Lotka–Volterra

Définition 3.7 (Système de Lotka–Volterra)**.** Soient $x(t)$ la population de proies et $y(t)$ celle de prédateurs. Le système classique de Lotka–Volterra est :

$$\dot{x} = ax - bxy, \tag{3.1}$$

$$\dot{y} = -cy + dxy, \tag{3.2}$$

où $a, b, c, d > 0$.

- a : taux de croissance intrinsèque des proies.
- b : taux de prédation.
- c : taux de mortalité des prédateurs.
- d : efficacité de conversion proie $\to$ prédateur.

3.4.1 Points d'équilibre

On annule le second membre :

$$x(a - by) = 0, \qquad y(-c + dx) = 0.$$

Proposition 3.8 (Équilibres de Lotka–Volterra)**.** Il y a deux points d'équilibre :

1. $E_0 = (0, 0)$: extinction totale.
2. $E^* = (c/d,\ a/b)$: coexistence.

3.4.2 Stabilité linéaire

La jacobienne du système est :

$$J(x, y) = \begin{pmatrix} a - by & -bx \\ dy & -c + dx \end{pmatrix}.$$

Théorème 3.9 (Stabilité des équilibres de Lotka–Volterra)**.** *1. En $E_0 = (0, 0)$: $J = \begin{pmatrix} a & 0 \\ 0 & -c \end{pmatrix}$, valeurs propres $\lambda_1 = a > 0$, $\lambda_2 = -c < 0$. E_0 est un **point selle** (instable).*

*2. En $E^* = (c/d, a/b)$: $J = \begin{pmatrix} 0 & -bc/d \\ da/b & 0 \end{pmatrix}$, valeurs propres $\lambda = \pm i\sqrt{ac}$. E^* est un **centre** (trajectoires périodiques).*

3.4.3 Intégrale première et trajectoires fermées

Théorème 3.10 (Intégrale première)**.** *Le système de Lotka–Volterra admet l'intégrale première :*

$$H(x, y) = dx - c \ln x + by - a \ln y = \text{constante}.$$

Les trajectoires dans le plan de phase sont donc des courbes de niveau de H, qui sont des courbes fermées entourant E^*.

Démonstration. On calcule $\frac{dH}{dt}$:

$$\begin{aligned} \frac{dH}{dt} &= d\dot{x} - \frac{c}{x}\dot{x} + b\dot{y} - \frac{a}{y}\dot{y} \\ &= d(ax - bxy) - \frac{c}{x}(ax - bxy) + b(-cy + dxy) - \frac{a}{y}(-cy + dxy) \\ &= dax - dbxy - ca + cby - bcy + bdxy + ac - adx = 0. \end{aligned}$$

□

Simulation de Lotka–Volterra

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

a, b, c, d = 1.0, 0.1, 1.5, 0.075

def lotka_volterra(t, z):
    x, y = z
    return [a*x - b*x*y, -c*y + d*x*y]

t_span = (0, 40)
z0 = [40, 9]
sol = solve_ivp(lotka_volterra, t_span, z0,
                t_eval=np.linspace(0, 40, 1000),
                method='RK45', rtol=1e-10)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(13, 5))

ax1.plot(sol.t, sol.y[0], 'b-', label='Proies $x(t)$')
ax1.plot(sol.t, sol.y[1], 'r-', label='Prédateurs $y(t)$')
ax1.set_xlabel('$t$'); ax1.set_ylabel('Population')
ax1.legend(); ax1.set_title('Séries temporelles')

# Portrait de phase avec plusieurs conditions initiales
for x0 in [10, 20, 30, 40, 60]:
    sol_i = solve_ivp(lotka_volterra, (0, 50), [x0, 9],
                      t_eval=np.linspace(0, 50, 2000),
                      method='RK45', rtol=1e-10)
    ax2.plot(sol_i.y[0], sol_i.y[1], 'b-', alpha=0.6)

ax2.plot(c/d, a/b, 'ro', markersize=8, label="$E^*$")
ax2.set_xlabel('Proies $x$'); ax2.set_ylabel('Prédateurs $y$')
ax2.legend(); ax2.set_title('Portrait de phase')
plt.tight_layout(); plt.savefig('lotka_volterra.pdf')
```

3.5 Extensions du modèle de Lotka–Volterra

3.5.1 Réponse fonctionnelle de Holling

Définition 3.11 (Réponses fonctionnelles)**.** La réponse fonctionnelle $g(x)$ modélise le taux de prédation par prédateur en fonction de la densité de proies :

- **Type I (linéaire) :** $g(x) = bx$ (Lotka–Volterra classique).
- **Type II (saturation) :** $g(x) = \frac{bx}{1+bhx}$ où h est le temps de manipulation.
- **Type III (sigmoïde) :** $g(x) = \frac{bx^2}{1+bhx^2}$.

Le système avec réponse de type II (modèle de Rosenzweig–MacArthur) est :

$$\dot{x} = rx\left(1 - \frac{x}{K}\right) - \frac{bxy}{1 + bhx}, \tag{3.3}$$

$$\dot{y} = y\left(\frac{dbx}{1 + bhx} - c\right). \tag{3.4}$$

Paradoxe de l'enrichissement

Rosenzweig (1971) a montré que l'augmentation de la capacité de charge K peut déstabiliser l'équilibre de coexistence par une bifurcation de Hopf, provoquant des oscillations de grande amplitude qui risquent de mener à l'extinction.

3.5.2 Compétition interspécifique

Définition 3.12 (Modèle de compétition de Lotka–Volterra)**.** Pour deux espèces en compétition :

$$\dot{N}_1 = r_1 N_1\left(1 - \frac{N_1 + \alpha_{12} N_2}{K_1}\right), \tag{3.5}$$

$$\dot{N}_2 = r_2 N_2\left(1 - \frac{N_2 + \alpha_{21} N_1}{K_2}\right), \tag{3.6}$$

où α_{ij} mesure l'effet compétitif de l'espèce j sur l'espèce i.

Proposition 3.13 (Principe d'exclusion compétitive)**.** Si $\alpha_{12}K_2/K_1 > 1$ et $\alpha_{21}K_1/K_2 > 1$, les deux espèces ne peuvent pas coexister de façon stable (exclusion compétitive de Gause). L'issue dépend des conditions initiales.

3.6 Modèles discrets : l'équation de Beverton–Holt

Définition 3.14 (Modèle de Beverton–Holt)**.** Pour des populations à générations non chevauchantes :

$$N_{t+1} = \frac{RN_t}{1 + (R-1)N_t/K},$$

où $R > 1$ est le taux de reproduction et K la capacité de charge.

Proposition 3.15. L'équilibre non trivial $N^* = K$ est globalement stable pour $R > 1$. Le modèle de Beverton–Holt est l'analogue discret de l'équation logistique et ne présente *pas* de chaos, contrairement à l'application logistique $N_{t+1} = RN_t(1 - N_t/K)$ étudiée au chapitre 5.

3.7 Limites des modèles de populations

Hypothèses et limites

- **Homogénéité spatiale :** les modèles précédents ignorent la structure spatiale. En réalité, les populations sont distribuées dans l'espace (voir cha-

pitre 6).

- **Déterminisme :** pour de petites populations, les fluctuations stochastiques deviennent importantes (chapitre 9).
- **Structure d'âge :** les modèles de Leslie introduisent une structure par classes d'âge.
- **Paramètres constants :** en réalité, r et K peuvent dépendre du temps (saisonnalité, changement climatique).

3.8 Exercices

Exercice 3.1. Montrer que la population logistique atteint la moitié de sa capacité de charge au temps $t_{1/2} = \frac{1}{r} \ln \frac{K-N_0}{N_0}$.

Exercice 3.2. Ajouter un terme de récolte constante h au modèle logistique : $\dot{N} = rN(1 - N/K) - h$.

1. Trouver les équilibres en fonction de h.
2. Déterminer la récolte maximale soutenable $h_{\max}$.
3. Tracer le diagramme de bifurcation N^* vs. h.
4. Simuler en Python pour $h < h_{\max}$, $h = h_{\max}$, $h > h_{\max}$.

Exercice 3.3. Pour le système de Lotka–Volterra, vérifier numériquement que $H(x, y)$ est conservée au cours du temps. Comparer les résultats obtenus avec les méthodes `RK45` et `RK23` de `solve_ivp` pour différentes tolérances.

Exercice 3.4. Étudier le modèle de Rosenzweig–MacArthur avec les paramètres $r = 1$, $b = 0.5$, $h = 0.5$, $d = 0.5$, $c = 0.3$ pour $K \in \{5, 10, 20, 50\}$.

1. Déterminer les équilibres et leur stabilité.
2. Tracer les portraits de phase.
3. Identifier la valeur critique de K pour la bifurcation de Hopf.

Exercice 3.5. Deux espèces de bactéries sont en compétition dans un environnement confiné, avec $r_1 = 0.8$, $r_2 = 0.6$, $K_1 = 500$, $K_2 = 400$, $\alpha_{12} = 0.7$, $\alpha_{21} = 1.2$.

1. Déterminer les équilibres et analyser leur stabilité.
2. Simuler le système pour différentes conditions initiales.
3. Conclure : y a-t-il coexistence ou exclusion ?

Chapitre 4

Modèles Épidémiologiques

4.1 Introduction

En 1927, William Ogilvy Kermack et Anderson Gray McKendrick publient un article qui allait fonder l'épidémiologie mathématique : leur modèle SIR divise la population en trois compartiments — Susceptibles, Infectieux, Rétablis — et prédit, avec une simplicité remarquable, le déroulement d'une épidémie. Le nombre de reproduction de base R_0 émerge naturellement : si $R_0 > 1$, l'épidémie se propage ; si $R_0 < 1$, elle s'éteint. Ce seuil, devenu célèbre lors de la pandémie de COVID-19, est un résultat purement mathématique qui guide les politiques de santé publique.

Les modèles épidémiologiques compartimentaux permettent de comprendre la propagation des maladies infectieuses et d'évaluer l'impact des interventions (vaccination, quarantaine). Ce chapitre développe les modèles SIR et SEIR à partir de principes premiers, analyse le nombre de reproduction de base $\mathcal{R}_0$ et propose des simulations Python.

4.2 Le modèle SIR de Kermack–McKendrick

Définition 4.1 (Compartiments SIR)**.** On divise la population totale N (constante) en trois compartiments :

- $S(t)$: **Susceptibles** (individus pouvant être infectés).
- $I(t)$: **Infectieux** (individus contagieux).
- $R(t)$: **Rétablis** (individus immunisés ou décédés).

La contrainte de conservation est $S(t) + I(t) + R(t) = N$.

4.2.1 Dérivation du modèle

Hypothèses du modèle SIR

- La population est homogène et bien mélangée.
- Le taux de transmission est proportionnel au produit SI (action de masse).
- Les infectieux guérissent au taux constant γ.
- Il n'y a ni naissance, ni mort (sauf par la maladie), ni démographie.

Les équations du modèle SIR sont :

$$\frac{dS}{dt} = -\beta \frac{SI}{N}, \tag{4.1}$$

$$\frac{dI}{dt} = \beta \frac{SI}{N} - \gamma I, \tag{4.2}$$

$$\frac{dR}{dt} = \gamma I, \tag{4.3}$$

où $\beta > 0$ est le taux de transmission et $\gamma > 0$ le taux de guérison.

Paramètres du modèle SIR

Paramètre	Signification	Unité
β	Taux de transmission	temps^{-1}
γ	Taux de guérison	temps^{-1}
$1/\gamma$	Durée moyenne d'infection	temps
$\mathcal{R}_0 = \beta/\gamma$	Nombre de reproduction de base	sans dimension

4.3 Le nombre de reproduction de base $\mathcal{R}_0$

Définition 4.2 (Nombre de reproduction de base)**.** Le **nombre de reproduction de base** $\mathcal{R}_0$ est le nombre moyen de cas secondaires produits par un individu infectieux dans une population entièrement susceptible :

$$\mathcal{R}_0 = \frac{\beta}{\gamma}.$$

Théorème 4.3 (Seuil épidémique)**.** *Une épidémie se déclenche si et seulement si* $\mathcal{R}_0 > 1$*. Plus précisément,* $dI/dt > 0$ *au début de l'épidémie si et seulement si* $\mathcal{R}_0 S(0)/N > 1$*.*

Démonstration. Au début, $S \approx N$. L'équation (4.2) donne :

$$\left.\frac{dI}{dt}\right|_{t=0} \approx (\beta - \gamma) I_0 = \gamma(\mathcal{R}_0 - 1) I_0.$$

Donc $dI/dt > 0$ si et seulement si $\mathcal{R}_0 > 1$. □

Remarque 4.4. Quelques valeurs typiques de $\mathcal{R}_0$: rougeole $\approx$ 12–18, grippe $\approx$ 1,5–3, COVID-19 (souche originale) $\approx$ 2,5–3,5, Ebola $\approx$ 1,5–2,5.

4.4 Analyse qualitative du modèle SIR

4.4.1 Réduction à deux équations

Grâce à $R = N - S - I$, il suffit d'étudier le système (S, I) dans le triangle $\{S \geq 0,\ I \geq 0,\ S + I \leq N\}$.

4.4.2 Taille finale de l'épidémie

Théorème 4.5 (Équation de la taille finale). *Si $I(0) = I_0 \ll N$ et $S(0) \approx N$, la fraction finale de susceptibles $s_\infty = S(\infty)/N$ vérifie l'équation transcendante :*

$$\ln s_\infty = \mathcal{R}_0(s_\infty - 1).$$

Démonstration. En divisant dS/dI :

$$\frac{dS}{dI} = \frac{-\beta SI/N}{\beta SI/N - \gamma I} = \frac{-\beta S/N}{\beta S/N - \gamma}.$$

Pour $S = Ns$ où $s = S/N$:

$$\frac{ds}{dI} = \frac{-\mathcal{R}_0 s}{\mathcal{R}_0 s - 1} \cdot \frac{1}{N}.$$

En intégrant et en utilisant $I(\infty) = 0$, $S(\infty) + R(\infty) = N$, on obtient l'équation de la taille finale. □

Simulation du modèle SIR

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

N = 10000
beta, gamma = 0.3, 0.1  # R0 = 3
S0, I0, R0_init = N - 10, 10, 0

def sir(t, y):
    S, I, R = y
    dS = -beta * S * I / N
    dI = beta * S * I / N - gamma * I
    dR = gamma * I
    return [dS, dI, dR]

sol = solve_ivp(sir, [0, 200], [S0, I0, R0_init],
                t_eval=np.linspace(0, 200, 1000),
                method='RK45')

plt.figure(figsize=(10, 5))
plt.plot(sol.t, sol.y[0], 'b-', label='$S(t)$')
plt.plot(sol.t, sol.y[1], 'r-', label='$I(t)$')
plt.plot(sol.t, sol.y[2], 'g-', label='$R(t)$')
plt.xlabel('Temps (jours)'); plt.ylabel('Nombre d\'individus')
plt.legend(); plt.title(f'Modèle SIR ($\\mathcal{{R}}_0 =
↪ {beta/gamma:.1f}$)')
plt.tight_layout(); plt.savefig('sir.pdf')
```

4.5 Vaccination et immunité de groupe

Théorème 4.6 (Seuil d'immunité de groupe). *Si une fraction p de la population est vaccinée (et immunisée), alors la fraction de susceptibles est $S(0)/N = 1 - p$. L'épidémie*

ne se déclenche pas si $(1-p)\mathcal{R}_0 < 1$*, soit :*

$$p > p_c = 1 - \frac{1}{\mathcal{R}_0}.$$

Exemple 4.7. Pour la rougeole ($\mathcal{R}_0 \approx 15$) : $p_c = 1 - 1/15 \approx 93{,}3\%$. Il faut vacciner plus de 93% de la population.

Effet de la vaccination

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

N = 10000
beta, gamma = 0.3, 0.1

def sir_vacc(t, y):
    S, I, R = y
    return [-beta*S*I/N, beta*S*I/N - gamma*I, gamma*I]

fig, axes = plt.subplots(2, 2, figsize=(12, 8))
vaccination_rates = [0, 0.3, 0.6, 0.75]

for ax, p in zip(axes.flat, vaccination_rates):
    S0 = N * (1 - p) - 10
    R0_init = N * p
    I0 = 10
    sol = solve_ivp(sir_vacc, [0, 200], [S0, I0, R0_init],
                    t_eval=np.linspace(0, 200, 500))
    ax.plot(sol.t, sol.y[1], 'r-', lw=2)
    ax.set_title(f'$p = {p:.0%}$, $I_{{\\max}} = {max(sol.y[1]):.0f}$')
    ax.set_xlabel('Temps'); ax.set_ylabel('$I(t)$')

plt.suptitle(f'Effet de la vaccination ($\\mathcal{{R}}_0 =
↪  {beta/gamma:.1f}$)')
plt.tight_layout(); plt.savefig('vaccination.pdf')
```

4.6 Le modèle SEIR

Définition 4.8 (Modèle SEIR)**.** On ajoute un compartiment **Exposé** (E) pour les individus infectés mais pas encore contagieux (période de latence $1/\sigma$) :

$$\frac{dS}{dt} = -\beta \frac{SI}{N}, \tag{4.4}$$

$$\frac{dE}{dt} = \beta \frac{SI}{N} - \sigma E, \tag{4.5}$$

$$\frac{dI}{dt} = \sigma E - \gamma I, \tag{4.6}$$

$$\frac{dR}{dt} = \gamma I. \tag{4.7}$$

Le nombre de reproduction de base reste $\mathcal{R}_0 = \beta/\gamma$.

Simulation du modèle SEIR

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

N = 10000
beta, sigma, gamma = 0.5, 0.2, 0.1
S0, E0, I0, R0_init = N - 10, 0, 10, 0

def seir(t, y):
    S, E, I, R = y
    dS = -beta * S * I / N
    dE = beta * S * I / N - sigma * E
    dI = sigma * E - gamma * I
    dR = gamma * I
    return [dS, dE, dI, dR]

sol = solve_ivp(seir, [0, 200], [S0, E0, I0, R0_init],
                t_eval=np.linspace(0, 200, 1000))

plt.figure(figsize=(10, 5))
plt.plot(sol.t, sol.y[0], 'b-', label='$S$')
plt.plot(sol.t, sol.y[1], 'orange', label='$E$')
plt.plot(sol.t, sol.y[2], 'r-', label='$I$')
plt.plot(sol.t, sol.y[3], 'g-', label='$R$')
plt.xlabel('Temps'); plt.ylabel('Population')
plt.legend(); plt.title(f'Modèle SEIR ($\\mathcal{{R}}_0 =
↪  {beta/gamma}$)')
plt.tight_layout(); plt.savefig('seir.pdf')
```

4.7 Extensions et modèles avancés

4.7.1 Modèle SIR avec démographie

Définition 4.9 (SIR avec démographie)**.** En ajoutant naissance (taux μ) et mort naturelle (taux μ) :

$$\frac{dS}{dt} = \mu N - \beta \frac{SI}{N} - \mu S, \tag{4.8}$$

$$\frac{dI}{dt} = \beta \frac{SI}{N} - (\gamma + \mu) I, \tag{4.9}$$

$$\frac{dR}{dt} = \gamma I - \mu R. \tag{4.10}$$

Le nombre de reproduction de base devient $\mathcal{R}_0 = \beta/(\gamma + \mu)$.

Proposition 4.10 (Équilibre endémique)**.** Pour $\mathcal{R}_0 > 1$, il existe un équilibre endémique

(maladie présente en permanence) :

$$S^* = \frac{N}{\mathcal{R}_0}, \qquad I^* = \frac{\mu N}{\gamma + \mu}\left(1 - \frac{1}{\mathcal{R}_0}\right).$$

4.7.2 Modèle SIRS (perte d'immunité)

Si l'immunité est temporaire (durée moyenne $1/\delta$), les rétablis redeviennent susceptibles :

$$\frac{dR}{dt} = \gamma I - \delta R, \qquad \frac{dS}{dt} = -\beta \frac{SI}{N} + \delta R.$$

Ce modèle peut produire des oscillations.

4.8 Ajustement aux données réelles

Ajustement SIR à des données

```
import numpy as np
from scipy.integrate import solve_ivp
from scipy.optimize import minimize

# Données synthétiques réalistes
np.random.seed(42)
N = 10000; beta_true, gamma_true = 0.3, 0.1
sol_true = solve_ivp(lambda t, y: [-beta_true*y[0]*y[1]/N,
                     beta_true*y[0]*y[1]/N - gamma_true*y[1],
                     gamma_true*y[1]],
                     [0, 100], [N-10, 10, 0],
                     t_eval=np.arange(0, 100, 7))
I_obs = sol_true.y[1] + np.random.normal(0, 50, len(sol_true.t))
I_obs = np.maximum(I_obs, 0)

def cost(params):
    b, g = params
    if b <= 0 or g <= 0: return 1e10
    sol = solve_ivp(lambda t, y: [-b*y[0]*y[1]/N,
                    b*y[0]*y[1]/N - g*y[1], g*y[1]],
                    [0, 100], [N-10, 10, 0],
                    t_eval=np.arange(0, 100, 7))
    return np.sum((sol.y[1] - I_obs)**2)

result = minimize(cost, [0.2, 0.05], method='Nelder-Mead')
print(f"beta = {result.x[0]:.4f}, gamma = {result.x[1]:.4f}")
print(f"R0 estimé = {result.x[0]/result.x[1]:.2f}")
```

4.9 Limites des modèles compartimentaux

Limites à connaître

- **Homogénéité :** la population est supposée bien mélangée ; en réalité, les contacts sont structurés (réseaux sociaux).
- **Paramètres constants :** β peut varier avec les saisons, les mesures de santé publique, le comportement.
- **Déterminisme :** pour de petits nombres de cas, les modèles stochastiques sont plus adaptés.
- **Hétérogénéité :** âge, géographie, comorbidités influencent la transmission et la sévérité.

4.10 Exercices

Exercice 4.1. Pour le modèle SIR avec $N = 10000$, $\beta = 0.4$, $\gamma = 0.1$:

1. Calculer $\mathcal{R}_0$ et le seuil d'immunité de groupe.
2. Simuler l'épidémie et déterminer le pic épidémique.
3. Vérifier numériquement l'équation de la taille finale.

Exercice 4.2. Comparer les dynamiques SIR et SEIR pour les mêmes valeurs de β et γ, avec $\sigma = 0.2$. Comment la période de latence affecte-t-elle le pic et la durée de l'épidémie ?

Exercice 4.3. Implémenter un modèle SIR stochastique (Gillespie) et comparer avec le modèle déterministe pour $N = 100$ et $N = 10000$.

Exercice 4.4. Modéliser une campagne de vaccination progressive : $p(t)$ augmente linéairement de 0 à $p_{\max}$ sur une durée T_v. Comparer l'impact sur le nombre total de cas pour différentes valeurs de T_v.

Exercice 4.5. Étudier le modèle SIRS avec $\beta = 0.5$, $\gamma = 0.1$, $\delta = 0.01$.

1. Trouver l'équilibre endémique.
2. Montrer numériquement l'existence d'oscillations amorties.
3. Comparer avec un modèle SIR standard.

CHAPITRE 4. *MODÈLES ÉPIDÉMIOLOGIQUES*

Chapitre 5

Systèmes Dynamiques Discrets et Chaos

En 1963, Edward Lorenz, météorologue au MIT, découvre par accident que son modèle atmosphérique simplifié est extrêmement sensible aux conditions initiales : une différence infime dans les données de départ produit des trajectoires radicalement différentes. C'est l'« effet papillon », et la naissance de la théorie du chaos. Mais le chaos n'est pas le désordre : il obéit à des règles précises, engendre des structures fractales magnifiques (l'attracteur de Lorenz, l'ensemble de Mandelbrot), et possède une théorie mathématique rigoureuse. Ce chapitre explore ce monde fascinant à travers les systèmes dynamiques discrets — les itérations de fonctions — où le chaos apparaît dans toute sa pureté.

5.1 Introduction

Les systèmes dynamiques discrets, définis par des itérations $x_{n+1} = f(x_n)$, constituent un cadre remarquablement simple dans lequel émergent des comportements complexes : bifurcations, doublement de période et chaos déterministe. Ce chapitre étudie en profondeur l'application logistique, les exposants de Lyapunov et les diagrammes de bifurcation.

5.2 Itérations et orbites

Définition 5.1 (Système dynamique discret)**.** Un **système dynamique discret** en dimension un est défini par une application $f : \mathbb{R} \to \mathbb{R}$ et la récurrence :

$$x_{n+1} = f(x_n), \qquad x_0 \text{ donné.}$$

L'**orbite** de x_0 est la suite $(x_0, x_1, x_2, \ldots)$.

Définition 5.2 (Point fixe et cycle)**.**

- Un **point fixe** est un point x^* tel que $f(x^*) = x^*$.
- Un **cycle de période** p est un ensemble $\{x_1^*, \ldots, x_p^*\}$ tel que $f^p(x_i^*) = x_i^*$ et $f^k(x_i^*) \neq x_i^*$ pour $0 < k < p$.

Théorème 5.3 (Stabilité d'un point fixe)**.** *Soit x^* un point fixe de f avec f dérivable en x^*.*

- *Si $|f'(x^*)| < 1$, le point fixe est **asymptotiquement stable** (attracteur).*

- *Si $|f'(x^*)| > 1$, le point fixe est **instable** (répulseur).*
- *Si $|f'(x^*)| = 1$, la stabilité dépend des termes d'ordre supérieur.*

Exemple 5.4 (Diagramme en escalier)**.** Pour $f(x) = \cos(x)$, le point fixe vérifie $x^* = \cos(x^*) \approx 0{,}7391$. On a $f'(x^*) = -\sin(x^*) \approx -0{,}6736$, donc $|f'(x^*)| < 1$: le point fixe est stable.

Diagramme en escalier (toile d'araignée)

```
import numpy as np
import matplotlib.pyplot as plt

def cobweb(f, x0, n_iter, ax, xmin=0, xmax=1):
    x = np.linspace(xmin, xmax, 500)
    ax.plot(x, f(x), 'b-', lw=2, label='$f(x)$')
    ax.plot(x, x, 'k--', lw=1, label='$y=x$')
    xn = x0
    for _ in range(n_iter):
        xn1 = f(xn)
        ax.plot([xn, xn], [xn, xn1], 'r-', lw=0.8)
        ax.plot([xn, xn1], [xn1, xn1], 'r-', lw=0.8)
        xn = xn1
    ax.set_xlabel('$x_n$'); ax.set_ylabel('$x_{n+1}$')

fig, ax = plt.subplots(figsize=(6, 6))
cobweb(np.cos, 0.1, 30, ax, xmin=-0.5, xmax=1.5)
ax.set_title('Diagramme en escalier pour $f(x) = \\cos(x)$')
plt.tight_layout(); plt.savefig('cobweb.pdf')
```

5.3 L'application logistique

Définition 5.5 (Application logistique)**.** L'**application logistique** est définie par :

$$f_r(x) = rx(1-x), \qquad x \in [0,1], \quad r \in [0,4].$$

Elle modélise la croissance d'une population avec saturation en temps discret.

5.3.1 Points fixes

Les points fixes vérifient $x = rx(1-x)$, soit :

$$x_1^* = 0, \qquad x_2^* = 1 - \frac{1}{r} \quad (r > 1).$$

Proposition 5.6 (Stabilité des points fixes)**.** On a $f_r'(x) = r(1-2x)$.

- $x_1^* = 0$: $f_r'(0) = r$. Stable si $r < 1$, instable si $r > 1$.
- $x_2^* = 1 - 1/r$: $f_r'(x_2^*) = 2 - r$. Stable si $1 < r < 3$.

5.4 Cascade de doublements de période

Théorème 5.7 (Scénario de Feigenbaum)**.** *Lorsque r augmente au-delà de 3, le point fixe x_2^* devient instable et donne naissance à un cycle de période 2 (bifurcation par doublement de période). Ce processus se répète : $2 \to 4 \to 8 \to \cdots$ Les valeurs de r aux bifurcations convergent géométriquement vers $r_\infty \approx 3{,}5699$ avec le rapport universel de Feigenbaum :*

$$\delta = \lim_{n\to\infty} \frac{r_n - r_{n-1}}{r_{n+1} - r_n} \approx 4{,}6692.$$

Remarque 5.8. La constante de Feigenbaum δ est **universelle** : elle ne dépend pas de la forme précise de f, pourvu que f ait un unique maximum quadratique.

Diagramme de bifurcation de l'application logistique

```python
import numpy as np
import matplotlib.pyplot as plt

r_vals = np.linspace(2.5, 4.0, 2000)
n_skip = 300   # transitoire
n_plot = 200   # points à tracer

r_list, x_list = [], []

for r in r_vals:
    x = 0.5
    for _ in range(n_skip):
        x = r * x * (1 - x)
    for _ in range(n_plot):
        x = r * x * (1 - x)
        r_list.append(r)
        x_list.append(x)

plt.figure(figsize=(12, 7))
plt.scatter(r_list, x_list, s=0.01, c='black', alpha=0.5)
plt.xlabel('$r$'); plt.ylabel('$x^*$')
plt.title('Diagramme de bifurcation - Application logistique')
plt.tight_layout(); plt.savefig('bifurcation.pdf', dpi=200)
```

5.5 Chaos déterministe

Définition 5.9 (Chaos (Devaney))**.** Un système dynamique discret $x_{n+1} = f(x_n)$ est dit **chaotique** au sens de Devaney s'il satisfait :

1. **Sensibilité aux conditions initiales :** $\exists\, \varepsilon > 0$ tel que pour tout x_0 et tout voisinage U de x_0, $\exists\, y_0 \in U$ et $\exists\, n$ tel que $|f^n(x_0) - f^n(y_0)| > \varepsilon$.

2. **Transitivité topologique :** pour tout couple d'ouverts non vides U, V, $\exists\, n$ tel que $f^n(U) \cap V \neq \emptyset$.

3. **Densité des orbites périodiques :** les points périodiques sont denses.

Théorème 5.10. *L'application logistique* $f_4(x) = 4x(1-x)$ *est chaotique sur* $[0,1]$.

Chaos et prévisibilité

Un système chaotique est **déterministe** : il n'y a pas d'aléatoire. Cependant, la sensibilité aux conditions initiales rend les prédictions à long terme impossibles en pratique, car toute erreur de mesure est amplifiée exponentiellement.

5.6 Exposants de Lyapunov

Définition 5.11 (Exposant de Lyapunov)**.** L'**exposant de Lyapunov** d'une orbite $(x_0, x_1, \ldots)$ sous f est :

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} \ln |f'(x_k)| \,.$$

- $\lambda < 0$: convergence vers un attracteur (point fixe ou cycle stable).
- $\lambda = 0$: comportement marginalement stable.
- $\lambda > 0$: **chaos** (sensibilité aux conditions initiales).

Calcul de l'exposant de Lyapunov

```
import numpy as np
import matplotlib.pyplot as plt

def lyapunov_exponent(r, x0=0.5, n_skip=1000, n_calc=5000):
    x = x0
    for _ in range(n_skip):
        x = r * x * (1 - x)
    lyap = 0.0
    for _ in range(n_calc):
        deriv = abs(r * (1 - 2 * x))
        if deriv == 0:
            return -np.inf
        lyap += np.log(deriv)
        x = r * x * (1 - x)
    return lyap / n_calc

r_vals = np.linspace(2.5, 4.0, 2000)
lyap_vals = [lyapunov_exponent(r) for r in r_vals]

plt.figure(figsize=(12, 4))
plt.plot(r_vals, lyap_vals, 'b-', lw=0.5)
plt.axhline(0, color='red', ls='--')
plt.xlabel('$r$'); plt.ylabel('$\\lambda$')
plt.title('Exposant de Lyapunov de l\'application logistique')
plt.tight_layout(); plt.savefig('lyapunov.pdf')
```

5.7 Fenêtres de périodicité

Remarque 5.12. Au sein de la région chaotique ($r > r_\infty$), il existe des **fenêtres de périodicité** où l'orbite redevient périodique. La plus grande est la fenêtre de période 3 autour de $r \approx 3{,}83$.

Théorème 5.13 (Li–Yorke, 1975)**.** *Si f admet un point périodique de période 3, alors f admet des orbites périodiques de* ***toute*** *période (« period three implies chaos »).*

5.8 L'application de Hénon

Définition 5.14 (Application de Hénon)**.** L'**application de Hénon** est un système dynamique discret en dimension 2 :

$$\begin{cases} x_{n+1} = 1 - a x_n^2 + y_n, \\ y_{n+1} = b x_n. \end{cases}$$

Pour $a = 1{,}4$ et $b = 0{,}3$, elle possède un attracteur étrange de dimension fractale $\approx 1{,}26$.

Attracteur de Hénon

```
import numpy as np
import matplotlib.pyplot as plt

a, b = 1.4, 0.3
n_iter = 50000
x, y = np.zeros(n_iter), np.zeros(n_iter)
x[0], y[0] = 0.1, 0.1

for n in range(n_iter - 1):
    x[n+1] = 1 - a * x[n]**2 + y[n]
    y[n+1] = b * x[n]

plt.figure(figsize=(10, 6))
plt.scatter(x[1000:], y[1000:], s=0.1, c='blue', alpha=0.5)
plt.xlabel('$x$'); plt.ylabel('$y$')
plt.title("Attracteur de Hénon ($a=1.4$, $b=0.3$)")
plt.tight_layout(); plt.savefig('henon.pdf', dpi=200)
```

5.9 Chaos en temps continu : aperçu du système de Lorenz

Définition 5.15 (Système de Lorenz)**.** Le système de Lorenz modélise la convection atmosphérique :

$$\dot{x} = \sigma(y - x), \tag{5.1}$$
$$\dot{y} = \rho x - y - xz, \tag{5.2}$$
$$\dot{z} = xy - \beta z. \tag{5.3}$$

Pour $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, le système présente un **attracteur étrange** en forme de papillon.

Attracteur de Lorenz

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from mpl_toolkits.mplot3d import Axes3D

sigma, rho, beta_l = 10, 28, 8/3

def lorenz(t, state):
    x, y, z = state
    return [sigma*(y-x), rho*x - y - x*z, x*y - beta_l*z]

sol = solve_ivp(lorenz, [0, 50], [1, 1, 1],
                t_eval=np.linspace(0, 50, 20000),
                method='RK45', rtol=1e-10)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.plot(sol.y[0], sol.y[1], sol.y[2], lw=0.3, color='blue')
ax.set_xlabel('$x$'); ax.set_ylabel('$y$'); ax.set_zlabel('$z$')
ax.set_title('Attracteur de Lorenz')
plt.tight_layout(); plt.savefig('lorenz.pdf')
```

5.10 Exercices

Exercice 5.1. Pour l'application logistique avec $r = 3{,}2$:

1. Trouver analytiquement les points du cycle de période 2.
2. Vérifier par simulation.
3. Montrer que le cycle est stable en calculant $(f_r^2)'$.

Exercice 5.2. Écrire un programme qui calcule numériquement les premières valeurs de bifurcation $r_1, r_2, \ldots, r_6$ et vérifier la convergence vers le rapport de Feigenbaum $\delta \approx 4{,}6692$.

Exercice 5.3. Tracer l'exposant de Lyapunov de l'application logistique en fonction de r et vérifier que $\lambda > 0$ dans les régions chaotiques et $\lambda < 0$ dans les fenêtres de périodicité.

Exercice 5.4. Pour l'application de Hénon avec différentes valeurs de a (en fixant $b = 0{,}3$), tracer l'attracteur et calculer les deux exposants de Lyapunov.

Exercice 5.5. Étudier la sensibilité aux conditions initiales dans le système de Lorenz : tracer deux trajectoires partant de $(1, 1, 1)$ et $(1{,}001, 1, 1)$ et mesurer la divergence exponentielle.

Exercice 5.6. Considérer l'application tente $f(x) = \min(2x, 2(1-x))$ sur $[0, 1]$.

1. Montrer que l'exposant de Lyapunov vaut $\lambda = \ln 2$.

2. Tracer le diagramme en escalier.

3. Pourquoi cette application est-elle exactement conjuguée à l'application logistique f_4 ?

Chapitre 6

Modèles de Diffusion et de Transport

6.1 Introduction

La diffusion est partout : un colorant se disperse dans l'eau, la chaleur se répand dans un métal, un polluant se dissémine dans l'atmosphère. Ce phénomène fondamental, étudié pour la première fois quantitativement par Adolf Fick en 1855 (par analogie avec la loi de Fourier pour la chaleur), obéit à l'équation de diffusion — la même équation de la chaleur, mais interprétée en termes de concentration plutôt que de température. Couplée à un champ de vitesse, elle devient l'équation d'advection-diffusion, qui modélise le transport de substances dans un milieu en mouvement. Ce chapitre développe ces modèles et leurs applications.

Les phénomènes de diffusion et de transport sont omniprésents : conduction thermique, dispersion de polluants, migration cellulaire. Ce chapitre dérive l'équation de la chaleur à partir de la loi de Fick, étudie l'équation d'advection-diffusion et propose des méthodes numériques pour les EDP paraboliques.

6.2 Loi de Fick et équation de diffusion

Définition 6.1 (Loi de Fick)**.** Le flux de diffusion J d'une substance de concentration $c(x,t)$ est proportionnel et opposé au gradient de concentration :

$$J = -D\frac{\partial c}{\partial x},$$

où $D > 0$ est le **coefficient de diffusion** (en m^2/s).

Théorème 6.2 (Équation de diffusion (chaleur))**.** *En combinant la loi de Fick avec la conservation de la masse $\partial c/\partial t = -\partial J/\partial x$, on obtient :*

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}.$$

C'est l'équation de la chaleur en une dimension spatiale.

Démonstration. Conservation locale :

$$\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} = -\frac{\partial}{\partial x}\left(-D\frac{\partial c}{\partial x}\right) = D\frac{\partial^2 c}{\partial x^2}.$$

□

Solution fondamentale (noyau de la chaleur)

Pour le problème sur $\mathbb{R}$ avec condition initiale $c(x,0) = \delta(x)$:

$$c(x,t) = \frac{1}{\sqrt{4\pi D t}} \exp\left(-\frac{x^2}{4Dt}\right), \qquad t > 0.$$

C'est une gaussienne dont l'écart-type croît comme $\sigma(t) = \sqrt{2Dt}$.

Diffusion et marche aléatoire

La diffusion est la limite macroscopique d'une marche aléatoire. Si une particule effectue des pas de longueur ℓ au rythme Δt, le coefficient de diffusion est $D = \ell^2/(2\Delta t)$.

6.3 Conditions aux limites

Définition 6.3 (Conditions aux limites classiques)**.** Sur un domaine $[0, L]$, les conditions aux limites les plus courantes sont :

- **Dirichlet :** $c(0,t) = c_0$, $c(L,t) = c_L$ (concentration imposée).
- **Neumann :** $\partial c/\partial x(0,t) = 0$ (flux nul, paroi isolante).
- **Robin :** $-D\partial c/\partial x + hc = hc_\infty$ (transfert convectif).

6.4 Méthode des différences finies

Définition 6.4 (Discrétisation)**.** On discrétise $[0, L]$ en $N+1$ points $x_j = j\Delta x$ et le temps en pas Δt. La dérivée seconde est approchée par :

$$\left.\frac{\partial^2 c}{\partial x^2}\right|_{x_j} \approx \frac{c_{j-1} - 2c_j + c_{j+1}}{(\Delta x)^2}.$$

Théorème 6.5 (Schéma explicite (FTCS))**.** *Le schéma Forward Time, Central Space est :*

$$c_j^{n+1} = c_j^n + \mu(c_{j-1}^n - 2c_j^n + c_{j+1}^n), \qquad \mu = \frac{D\Delta t}{(\Delta x)^2}.$$

Ce schéma est stable si et seulement si $\mu \leq 1/2$.

Condition CFL

La condition de stabilité $\mu = D\Delta t/(\Delta x)^2 \leq 1/2$ impose une relation entre les pas de temps et d'espace. Pour des maillages fins, Δt doit être très petit, rendant le schéma explicite coûteux. Les schémas implicites (Crank–Nicolson) levènt cette contrainte.

Diffusion 1D — schéma explicite

```python
import numpy as np
import matplotlib.pyplot as plt

L, D, T_final = 1.0, 0.01, 2.0
Nx = 50; dx = L / Nx
dt = 0.4 * dx**2 / D  # CFL : mu = 0.4 < 0.5
Nt = int(T_final / dt)
mu = D * dt / dx**2

x = np.linspace(0, L, Nx + 1)
c = np.exp(-((x - 0.5)**2) / (2 * 0.05**2))  # CI gaussienne

plt.figure(figsize=(10, 6))
plt.plot(x, c, label='$t=0$')

for n in range(Nt):
    c_new = c.copy()
    c_new[1:-1] = c[1:-1] + mu * (c[:-2] - 2*c[1:-1] + c[2:])
    c_new[0] = 0; c_new[-1] = 0  # Dirichlet
    c = c_new
    if (n+1) * dt in [0.05, 0.2, 0.5, 1.0, 2.0]:
        plt.plot(x, c, label=f'$t={((n+1)*dt):.2f}$')

plt.xlabel('$x$'); plt.ylabel('$c(x,t)$')
plt.legend(); plt.title('Diffusion 1D (schéma explicite)')
plt.tight_layout(); plt.savefig('diffusion_explicite.pdf')
```

6.5 Équation d'advection-diffusion

Définition 6.6 (Équation d'advection-diffusion)**.** Lorsque la substance est également transportée par un champ de vitesse v :

$$\frac{\partial c}{\partial t} + v\frac{\partial c}{\partial x} = D\frac{\partial^2 c}{\partial x^2}.$$

Le nombre de Péclet $\mathrm{Pe} = vL/D$ mesure l'importance relative de l'advection par rapport à la diffusion.

Proposition 6.7 (Régimes limites)**.**

- $\mathrm{Pe} \ll 1$: la diffusion domine (équation de la chaleur).
- $\mathrm{Pe} \gg 1$: l'advection domine (équation de transport).
- $\mathrm{Pe} \sim 1$: les deux effets sont comparables.

Advection-diffusion

```python
import numpy as np
import matplotlib.pyplot as plt
```

```
L, D, v = 1.0, 0.005, 0.5
Nx = 200; dx = L / Nx
dt = 0.001
Nt = 1000

x = np.linspace(0, L, Nx + 1)
c = np.exp(-((x - 0.2)**2) / (2 * 0.03**2))

plt.figure(figsize=(10, 5))
plt.plot(x, c, 'k--', label='$t=0$')

for n in range(Nt):
    c_new = c.copy()
    # Upwind pour l'advection + centré pour la diffusion
    c_new[1:-1] = (c[1:-1]
                   - v * dt / dx * (c[1:-1] - c[:-2])
                   + D * dt / dx**2 * (c[:-2] - 2*c[1:-1] + c[2:]))
    c_new[0] = 0; c_new[-1] = 0
    c = c_new
    if (n+1) in [200, 500, 800, 1000]:
        plt.plot(x, c, label=f'$t={(n+1)*dt:.2f}$')

plt.xlabel('$x$'); plt.ylabel('$c(x,t)$')
plt.legend(); plt.title(f'Advection-diffusion (Pe = {v*L/D:.0f})')
plt.tight_layout(); plt.savefig('advection_diffusion.pdf')
```

6.6 Diffusion en dimensions supérieures

Définition 6.8 (Équation de diffusion en 2D/3D)**.** En d dimensions spatiales :

$$\frac{\partial c}{\partial t} = D\nabla^2 c = D\sum_{i=1}^{d} \frac{\partial^2 c}{\partial x_i^2}.$$

La solution fondamentale en d dimensions est :

$$c(\mathbf{x}, t) = \frac{1}{(4\pi D t)^{d/2}} \exp\left(-\frac{\|\mathbf{x}\|^2}{4Dt}\right).$$

6.7 Modèle de réaction-diffusion

Définition 6.9 (Équation de réaction-diffusion)**.** En ajoutant un terme de réaction $f(c)$:

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} + f(c).$$

Exemple 6.10 (Équation de Fisher–KPP)**.** Avec $f(c) = rc(1-c)$ (croissance logistique), on obtient l'équation de Fisher :

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} + rc(1-c).$$

Elle admet des **fronts d'onde progressifs** de vitesse $v_{\min} = 2\sqrt{Dr}$.

Théorème 6.11 (Vitesse minimale du front de Fisher)**.** *L'équation de Fisher–KPP admet des solutions en onde progressive $c(x,t) = \phi(x - vt)$ pour toute vitesse $v \geq v_{\min} = 2\sqrt{Dr}$. Pour des conditions initiales à support compact, le front se propage à la vitesse $v_{\min}$.*

Front d'onde de Fisher

```python
import numpy as np
import matplotlib.pyplot as plt

L, D, r = 20.0, 1.0, 1.0
Nx = 400; dx = L / Nx
dt = 0.4 * dx**2 / D
x = np.linspace(0, L, Nx + 1)

# CI : front initial
c = np.where(x < 2, 1.0, 0.0)

plt.figure(figsize=(10, 5))
times_to_plot = [0, 2, 4, 6, 8]
t_current = 0

for t_target in times_to_plot:
    while t_current < t_target:
        c_new = c.copy()
        c_new[1:-1] = (c[1:-1]
            + dt * D * (c[:-2] - 2*c[1:-1] + c[2:]) / dx**2
            + dt * r * c[1:-1] * (1 - c[1:-1]))
        c_new[0] = 1.0; c_new[-1] = 0.0
        c = c_new
        t_current += dt
    plt.plot(x, c, label=f'$t = {t_target}$')

v_min = 2 * np.sqrt(D * r)
plt.xlabel('$x$'); plt.ylabel('$c(x,t)$')
plt.title(f'Front de Fisher ($v_{{\\min}} = {v_min:.2f}$)')
plt.legend(); plt.tight_layout()
plt.savefig('fisher_front.pdf')
```

6.8 Systèmes de Turing

Définition 6.12 (Instabilité de Turing)**.** Un système de réaction-diffusion à deux espèces :

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + f(u,v), \tag{6.1}$$

$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + g(u,v), \tag{6.2}$$

peut présenter une **instabilité de Turing** : un équilibre homogène stable en l'absence de diffusion peut devenir instable en présence de diffusion si $D_v \gg D_u$. Cela donne naissance à des **motifs spatiaux** (taches, rayures).

Remarque 6.13. L'instabilité de Turing nécessite typiquement un activateur (diffusion lente, D_u petit) et un inhibiteur (diffusion rapide, D_v grand).

6.9 Exercices

Exercice 6.1. Vérifier que le noyau de la chaleur $c(x,t) = \frac{1}{\sqrt{4\pi D t}} \exp(-x^2/(4Dt))$ est bien solution de l'équation de diffusion.

Exercice 6.2. Résoudre l'équation de la chaleur sur $[0, L]$ avec conditions de Dirichlet homogènes par séparation des variables. Montrer que la solution générale est :

$$c(x,t) = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) \exp\left(-\frac{n^2\pi^2 D}{L^2} t\right).$$

Exercice 6.3. Implémenter le schéma de Crank–Nicolson pour l'équation de diffusion 1D et comparer avec le schéma explicite en termes de précision et de stabilité.

Exercice 6.4. Simuler l'équation d'advection-diffusion pour différentes valeurs de Pe et observer la transition entre le régime diffusif et le régime advectif.

Exercice 6.5. Pour l'équation de Fisher–KPP, vérifier numériquement que la vitesse du front converge vers $v_{\min} = 2\sqrt{Dr}$ pour des conditions initiales à support compact.

Exercice 6.6. Implémenter un système de réaction-diffusion de type Schnakenberg en 2D et observer la formation de motifs de Turing.

Chapitre 7

Modélisation Économique et Financière

7.1 Introduction

Les économistes ont longtemps envié la physique pour la précision de ses modèles. En 1956, Robert Solow propose un modèle de croissance économique d'une élégance newtonienne : le capital s'accumule, la main-d'œuvre croît, et la technologie progresse, le tout régi par une équation différentielle dont l'état stationnaire éclaire les disparités de richesse entre nations. Ce modèle lui vaudra le prix Nobel en 1987. Mais l'économie est aussi le domaine de l'incertitude : les marchés financiers oscillent, les prix des options défient l'intuition. C'est là que le modèle de Black–Scholes, enraciné dans le calcul stochastique d'Itô, révolutionne la finance en 1973. Ce chapitre tisse le lien entre équations différentielles déterministes et stochastiques appliquées à l'économie et à la finance, du modèle de Solow à l'évaluation des options.

7.2 Modèle de croissance de Solow

Définition 7.1 (Modèle de Solow)**.** Le modèle de Solow décrit la croissance économique à long terme. Soit $K(t)$ le capital, $L(t)$ le travail et $Y(t)$ la production. La fonction de production est de type Cobb–Douglas :

$$Y = AK^{\alpha}L^{1-\alpha}, \qquad 0 < \alpha < 1,$$

où A est la productivité totale des facteurs (PTF). L'accumulation du capital vérifie :

$$\dot{K} = sY - \delta K,$$

où $s \in (0,1)$ est le taux d'épargne et $\delta > 0$ le taux de dépréciation.

7.2.1 Capital par tête

Supposons $L(t) = L_0 e^{nt}$ (croissance démographique au taux n). En posant $k = K/L$ (capital par tête) et $y = Y/L$:

Théorème 7.2 (Équation fondamentale de Solow)**.** *Le capital par tête vérifie :*

$$\dot{k} = sAk^{\alpha} - (n+\delta)k.$$

Démonstration. $\dot{k} = \dot{K}/L - Kn/L = (sY - \delta K)/L - nk = sy - \delta k - nk = sAk^{\alpha} - (n + \delta)k$. □

Proposition 7.3 (Équilibre de Solow)**.** L'équilibre $k^* > 0$ vérifie $sAk^{*\alpha} = (n+\delta)k^*$, d'où :

$$k^* = \left(\frac{sA}{n+\delta}\right)^{1/(1-\alpha)}.$$

Cet équilibre est **globalement stable** pour $k > 0$.

Équilibre de Solow

$$k^* = \left(\frac{sA}{n+\delta}\right)^{1/(1-\alpha)}, \qquad y^* = A\left(\frac{sA}{n+\delta}\right)^{\alpha/(1-\alpha)}.$$

Simulation du modèle de Solow

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

A, alpha = 1.0, 0.3
s, delta, n = 0.2, 0.05, 0.02

def solow(t, k):
    return s * A * k**alpha - (n + delta) * k

k_star = (s * A / (n + delta))**(1 / (1 - alpha))
print(f"Équilibre : k* = {k_star:.4f}")

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(13, 5))

for k0 in [0.5, 1, 2, 5, 10, 15]:
    sol = solve_ivp(solow, [0, 100], [k0],
                    t_eval=np.linspace(0, 100, 500))
    ax1.plot(sol.t, sol.y[0], label=f'$k_0={k0}$')

ax1.axhline(k_star, color='gray', ls='--', label=f'$k^*={k_star:.2f}$')
ax1.set_xlabel('$t$'); ax1.set_ylabel('$k(t)$')
ax1.legend(fontsize=8); ax1.set_title('Convergence vers $k^*$')

# Diagramme de Solow
k_vals = np.linspace(0, 15, 200)
ax2.plot(k_vals, s * A * k_vals**alpha, 'b-', label='$sAk^\\alpha$')
ax2.plot(k_vals, (n + delta) * k_vals, 'r-', label='$(n+\\delta)k$')
ax2.plot(k_star, (n+delta)*k_star, 'ko', markersize=8)
ax2.set_xlabel('$k$'); ax2.set_ylabel('Investissement / Dépréciation')
ax2.legend(); ax2.set_title('Diagramme de Solow')
plt.tight_layout(); plt.savefig('solow.pdf')
```

7.3 Équilibre offre-demande

Définition 7.4 (Modèle d'ajustement des prix)**.** Soient $D(p)$ la demande et $S(p)$ l'offre en fonction du prix p. Le prix s'ajuste proportionnellement à l'excès de demande :

$$\dot{p} = \kappa\big(D(p) - S(p)\big), \qquad \kappa > 0.$$

Exemple 7.5 (Fonctions linéaires)**.** Avec $D(p) = a - bp$ et $S(p) = c + dp$ $(a, b, c, d > 0)$:

$$\dot{p} = \kappa\big((a - c) - (b + d)p\big).$$

L'équilibre est $p^* = (a - c)/(b + d)$. C'est un équilibre globalement stable (l'équation est linéaire avec coefficient négatif devant p).

7.3.1 Modèle du cobweb (toile d'araignée)

Définition 7.6 (Modèle du cobweb)**.** En temps discret, les producteurs décident leur offre sur la base du prix passé :

$$S_t = S(p_{t-1}), \qquad p_t = D^{-1}(S_t).$$

Proposition 7.7. Si $|S'(p^*)/D'(p^*)| < 1$, le modèle converge vers l'équilibre. Si $|S'(p^*)/D'(p^*)| > 1$, les oscillations divergent.

7.4 Modèle de Black–Scholes

Définition 7.8 (Mouvement brownien géométrique)**.** Le prix d'un actif financier $S(t)$ suit un mouvement brownien géométrique :

$$dS = \mu S\, dt + \sigma S\, dW_t,$$

où μ est le rendement moyen, σ la volatilité et W_t un mouvement brownien standard.

Théorème 7.9 (Équation de Black–Scholes)**.** *Le prix $V(S, t)$ d'une option européenne sur un actif de prix S vérifie l'EDP :*

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0,$$

où r est le taux sans risque.

Formule de Black–Scholes pour un call européen

$$C(S, t) = S\,\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2),$$

où K est le prix d'exercice, T la maturité, Φ la fonction de répartition de la loi normale, et

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)(T - t)}{\sigma\sqrt{T - t}}, \qquad d_2 = d_1 - \sigma\sqrt{T - t}.$$

Formule de Black–Scholes en Python

```python
import numpy as np
from scipy.stats import norm

def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * norm.cdf(d1) - K * np.exp(-r*T) * norm.cdf(d2)

def black_scholes_put(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return K * np.exp(-r*T) * norm.cdf(-d2) - S * norm.cdf(-d1)

# Exemple
S0, K, T, r, sigma = 100, 100, 1.0, 0.05, 0.2
print(f"Call = {black_scholes_call(S0, K, T, r, sigma):.4f}")
print(f"Put  = {black_scholes_put(S0, K, T, r, sigma):.4f}")

# Les Grecs
import matplotlib.pyplot as plt
S_range = np.linspace(60, 140, 200)
calls = [black_scholes_call(S, K, T, r, sigma) for S in S_range]

fig, ax = plt.subplots(figsize=(8, 5))
ax.plot(S_range, calls, 'b-', lw=2, label='Call')
ax.plot(S_range, np.maximum(S_range - K, 0), 'r--', label='Payoff')
ax.set_xlabel('$S$'); ax.set_ylabel('Prix du call')
ax.legend(); ax.set_title('Prix du call européen (Black-Scholes)')
plt.tight_layout(); plt.savefig('black_scholes.pdf')
```

7.5 Simulation Monte Carlo

Définition 7.10 (Évaluation par Monte Carlo)**.** On simule M trajectoires du prix $S(T)$ et on estime le prix de l'option par la moyenne actualisée des payoffs :

$$\hat{C} = e^{-rT} \frac{1}{M} \sum_{i=1}^{M} \max(S_i(T) - K,\ 0).$$

Monte Carlo pour le call européen

```python
import numpy as np

S0, K, T, r, sigma = 100, 100, 1.0, 0.05, 0.2
M = 100000
np.random.seed(42)

Z = np.random.standard_normal(M)
```

```
S_T = S0 * np.exp((r - 0.5*sigma**2)*T + sigma*np.sqrt(T)*Z)
payoffs = np.maximum(S_T - K, 0)
C_mc = np.exp(-r*T) * np.mean(payoffs)
se = np.exp(-r*T) * np.std(payoffs) / np.sqrt(M)

print(f"Prix MC = {C_mc:.4f} ± {1.96*se:.4f} (IC 95%)")
print(f"Prix BS = {black_scholes_call(S0, K, T, r, sigma):.4f}")
```

7.6 Modèle de Markowitz

Définition 7.11 (Optimisation de portefeuille)**.** Pour n actifs de rendements espérés $\boldsymbol{\mu} \in \mathbb{R}^n$ et de matrice de covariance Σ, le portefeuille optimal au sens de Markowitz minimise la variance pour un rendement cible r_c :

$$\min_{\mathbf{w}} \ \mathbf{w}^\top \Sigma \mathbf{w} \quad \text{s.c.} \quad \mathbf{w}^\top \boldsymbol{\mu} = r_c, \quad \mathbf{w}^\top \mathbf{1} = 1.$$

Proposition 7.12 (Frontière efficiente)**.** L'ensemble des portefeuilles optimaux forme une parabole dans le plan (risque, rendement), appelée **frontière efficiente**.

7.7 Modèle de réseau d'entreprises

Définition 7.13 (Modèle entrée-sortie de Leontief)**.** Soient n secteurs économiques. La matrice $A = (a_{ij})$ donne la quantité de bien i nécessaire pour produire une unité de bien j. L'équilibre de production vérifie :

$$\mathbf{x} = A\mathbf{x} + \mathbf{d} \quad \Leftrightarrow \quad \mathbf{x} = (I - A)^{-1}\mathbf{d},$$

où $\mathbf{d}$ est la demande finale.

Proposition 7.14. Si le rayon spectral de A vérifie $\rho(A) < 1$, alors $(I - A)$ est inversible et $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k \geq 0$.

7.8 Limites des modèles financiers

Limites du modèle de Black–Scholes

- La volatilité σ n'est pas constante (smile de volatilité).
- Les rendements réels ont des queues lourdes (événements extrêmes plus fréquents que prévu par la loi normale).
- Le marché n'est pas toujours liquide et sans friction.
- Les crises financières révèlent des corrélations non linéaires que les modèles gaussiens ignorent.

7.9 Exercices

Exercice 7.1. Pour le modèle de Solow avec $A = 1$, $\alpha = 0{,}3$, $s = 0{,}25$, $\delta = 0{,}05$, $n = 0{,}01$:

1. Calculer k^* et y^*.
2. Étudier l'effet d'une augmentation du taux d'épargne s.
3. Trouver le taux d'épargne optimal (règle d'or de Phelps).

Exercice 7.2. Simuler le modèle du cobweb avec $D(p) = 100 - 2p$ et $S(p) = -10 + 3p$. Déterminer si les oscillations convergent ou divergent.

Exercice 7.3. Calculer les « Grecs » du call européen (Delta, Gamma, Vega, Theta, Rho) analytiquement et numériquement.

Exercice 7.4. Comparer le prix Monte Carlo d'un put européen avec la formule de Black–Scholes pour différents nombres de trajectoires M. Tracer l'erreur en fonction de M et vérifier la convergence en $O(1/\sqrt{M})$.

Exercice 7.5. Pour un portefeuille de 3 actifs avec rendements et covariances donnés, tracer la frontière efficiente de Markowitz.

Chapitre 8

Optimisation et Modèles Variationnels

8.1 Introduction

La nature est économe. La lumière suit le chemin le plus rapide (principe de Fermat), une bulle de savon minimise sa surface (problème de Plateau), une chaînette adopté la forme qui minimise son énergie potentielle. Cette idée — que les lois de la physique sont des problèmes d'optimisation — a fasciné les mathématiciens depuis Euler et Lagrange au XVIII$^{\text{e}}$ siècle. Le *calcul des variations*, qui cherche la fonction optimisant une fonctionnelle, est l'outil qui en découle. Mais l'optimisation irrigue aussi les sciences appliquées : programmation linéaire pour la logistique, conditions de Karush–Kuhn–Tucker pour l'optimisation sous contraintes, méthodes numériques pour les problèmes de grande taille. Ce chapitre couvre ces trois facettes, de la théorie classique aux algorithmes modernes.

8.2 Programmation linéaire

Définition 8.1 (Problème de programmation linéaire)**.** Un **problème de programmation linéaire** (PL) consiste à minimiser une fonction linéaire sous contraintes linéaires :

$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{c}^\top \mathbf{x} \quad \text{s.c.} \quad A\mathbf{x} \le \mathbf{b}, \quad \mathbf{x} \ge \mathbf{0}.$$

Théorème 8.2 (Théorème fondamental de la PL)**.** *Si un problème de PL admet une solution optimale, alors il en admet une qui est un* ***sommet*** *(point extrême) du polyèdre des contraintes.*

Exemple 8.3 (Problème de production)**.** Une usine fabrique deux produits x_1, x_2 avec les contraintes de ressources et les profits unitaires suivants :

$$\begin{aligned} \max \quad & 40x_1 + 30x_2 \\ \text{s.c.} \quad & x_1 + x_2 \le 40, \\ & 2x_1 + x_2 \le 60, \\ & x_1, x_2 \ge 0. \end{aligned}$$

Programmation linéaire avec scipy

```python
import numpy as np
from scipy.optimize import linprog

# min c^T x  =>  max 40x1 + 30x2  =>  min -40x1 - 30x2
c = [-40, -30]
A_ub = [[1, 1], [2, 1]]
b_ub = [40, 60]
bounds = [(0, None), (0, None)]

result = linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=bounds,
                 method='highs')
print(f"Solution optimale : x1 = {result.x[0]:.1f}, "
      f"x2 = {result.x[1]:.1f}")
print(f"Profit maximal : {-result.fun:.1f}")
```

Définition 8.4 (Dualité)**.** Le **problème dual** associé au PL primal $\min \mathbf{c}^\top \mathbf{x}$ s.c. $A\mathbf{x} \geq \mathbf{b}$, $\mathbf{x} \geq 0$ est :

$$\max_{\mathbf{y} \geq 0} \mathbf{b}^\top \mathbf{y} \quad \text{s.c.} \quad A^\top \mathbf{y} \leq \mathbf{c}.$$

Théorème 8.5 (Théorème de dualité forte)**.** *Si le primal et le dual ont des solutions réalisables, alors les valeurs optimales sont égales :* $\mathbf{c}^\top \mathbf{x}^* = \mathbf{b}^\top \mathbf{y}^*$.

8.3 Optimisation non linéaire sans contraintes

Définition 8.6 (Conditions nécessaires d'optimalité)**.** Pour $f : \mathbb{R}^n \to \mathbb{R}$ de classe $\mathcal{C}^2$, si $\mathbf{x}^*$ est un minimum local, alors :

1. $\nabla f(\mathbf{x}^*) = \mathbf{0}$ (condition du premier ordre).
2. $\nabla^2 f(\mathbf{x}^*)$ est semi-définie positive (condition du second ordre).

Définition 8.7 (Méthode de descente de gradient)**.** La méthode de descente de gradient itère :

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k),$$

où $\alpha_k > 0$ est le pas d'apprentissage.

Descente de gradient en Python

```python
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return (x[0]-1)**2 + 10*(x[1]-x[0]**2)**2  # Rosenbrock

def grad_f(x):
    dx0 = 2*(x[0]-1) - 40*x[0]*(x[1]-x[0]**2)
    dx1 = 20*(x[1]-x[0]**2)
```

```
    return np.array([dx0, dx1])

x = np.array([-1.0, 1.0])
alpha = 0.002
trajectory = [x.copy()]

for _ in range(5000):
    x = x - alpha * grad_f(x)
    trajectory.append(x.copy())

trajectory = np.array(trajectory)
print(f"Minimum trouvé : x = ({x[0]:.6f}, {x[1]:.6f})")
print(f"f(x) = {f(x):.8f}")

# Visualisation
x1 = np.linspace(-2, 2, 200)
x2 = np.linspace(-1, 3, 200)
X1, X2 = np.meshgrid(x1, x2)
Z = (X1-1)**2 + 10*(X2-X1**2)**2

plt.figure(figsize=(8, 6))
plt.contour(X1, X2, Z, levels=np.logspace(-1, 3, 30), cmap='viridis')
plt.plot(trajectory[:, 0], trajectory[:, 1], 'r-', lw=0.5)
plt.plot(1, 1, 'r*', markersize=15)
plt.xlabel('$x_1$'); plt.ylabel('$x_2$')
plt.title('Descente de gradient sur la fonction de Rosenbrock')
plt.tight_layout(); plt.savefig('gradient_descent.pdf')
```

8.4 Optimisation sous contraintes : conditions KKT

Théorème 8.8 (Conditions de Karush–Kuhn–Tucker)**.** *Considérons le problème :*

$$\min_{\mathbf{x}} \; f(\mathbf{x}) \quad s.c. \quad g_i(\mathbf{x}) \le 0,\; i = 1, \dots, m, \qquad h_j(\mathbf{x}) = 0,\; j = 1, \dots, p.$$

Si $\mathbf{x}^*$ est un minimum local et qu'une condition de qualification des contraintes est satisfaite, alors il existe des multiplicateurs $\lambda_i \ge 0$ et μ_j tels que :

1. $\nabla f(\mathbf{x}^*) + \sum_i \lambda_i \nabla g_i(\mathbf{x}^*) + \sum_j \mu_j \nabla h_j(\mathbf{x}^*) = \mathbf{0}$ *(stationnarité).*
2. $g_i(\mathbf{x}^*) \le 0$ *pour tout i (faisabilité primale).*
3. $\lambda_i \ge 0$ *pour tout i (faisabilité duale).*
4. $\lambda_i g_i(\mathbf{x}^*) = 0$ *pour tout i (complémentarité).*

Exemple 8.9. Minimiser $f(x, y) = x^2 + y^2$ sous la contrainte $x + y \ge 1$. Le lagrangien est $\mathcal{L} = x^2 + y^2 - \lambda(x + y - 1)$. Conditions KKT : $2x = \lambda$, $2y = \lambda$, $\lambda(x + y - 1) = 0$, $\lambda \ge 0$. D'où $x = y = \lambda/2$. Si $\lambda > 0$: $x + y = 1$, donc $\lambda = 1$, $x^* = y^* = 1/2$.

Optimisation sous contraintes avec `scipy`

```python
from scipy.optimize import minimize

def objective(x):
    return x[0]**2 + x[1]**2

constraints = [{'type': 'ineq', 'fun': lambda x: x[0] + x[1] - 1}]
x0 = [0.0, 0.0]
result = minimize(objective, x0, method='SLSQP',
                  constraints=constraints)
print(f"Solution : x = ({result.x[0]:.4f}, {result.x[1]:.4f})")
print(f"f(x*) = {result.fun:.4f}")
```

8.5 Calcul des variations

Définition 8.10 (Problème du calcul des variations)**.** On cherche la fonction $y : [a, b] \to \mathbb{R}$ rendant stationnaire la fonctionnelle :

$$J[y] = \int_a^b F(x, y, y')\, dx,$$

avec conditions aux limites $y(a) = y_a$, $y(b) = y_b$.

Théorème 8.11 (Équation d'Euler–Lagrange)**.** *Si y^* rend J stationnaire, alors y^* satisfait :*

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\frac{\partial F}{\partial y'} = 0.$$

Démonstration. Soit η une variation admissible ($\eta(a) = \eta(b) = 0$). Posons $y_\varepsilon = y^* + \varepsilon\eta$. La condition de stationnarité est :

$$\frac{d}{d\varepsilon} J[y_\varepsilon]\Big|_{\varepsilon=0} = 0.$$

On calcule :

$$\int_a^b \left(\frac{\partial F}{\partial y}\eta + \frac{\partial F}{\partial y'}\eta' \right) dx = 0.$$

Intégration par parties du second terme (les termes de bord s'annulent) et application du lemme fondamental du calcul des variations donne l'équation d'Euler–Lagrange. □

Exemple 8.12 (Géodésiques dans le plan)**.** La longueur d'une courbe $y(x)$ entre (a, y_a) et (b, y_b) est $J[y] = \int_a^b \sqrt{1 + y'^2}\, dx$. Ici $F = \sqrt{1 + y'^2}$, $\partial F/\partial y = 0$. L'équation d'Euler–Lagrange donne $y''/(1 + y'^2)^{3/2} = 0$, soit $y'' = 0$: les géodésiques sont des droites.

Exemple 8.13 (Brachistochrone)**.** Trouver la courbe $y(x)$ le long de laquelle une bille glisse sans frottement de $(0, 0)$ à (x_1, y_1) en un temps minimal. Le temps de descente est :

$$T[y] = \int_0^{x_1} \frac{\sqrt{1 + y'^2}}{\sqrt{2gy}}\, dx.$$

L'équation d'Euler–Lagrange conduit à une **cycloïde**.

Brachistochrone numérique

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

# Discrétisation : y aux N points intérieurs
N = 30
x1, y1 = 1.0, 0.5
x = np.linspace(0, x1, N + 2)
dx = x[1] - x[0]
g = 9.81

def travel_time(y_inner):
    y = np.concatenate([[0], y_inner, [y1]])
    dy = np.diff(y) / dx
    y_mid = 0.5 * (y[:-1] + y[1:])
    y_mid = np.maximum(y_mid, 1e-10)  # éviter division par zéro
    ds = np.sqrt(dx**2 + (np.diff(y))**2)
    v = np.sqrt(2 * g * y_mid)
    return np.sum(ds / v)

y0_inner = np.linspace(0, y1, N + 2)[1:-1]
result = minimize(travel_time, y0_inner, method='L-BFGS-B',
                  bounds=[(0.001, 2.0)] * N)
y_opt = np.concatenate([[0], result.x, [y1]])

plt.figure(figsize=(8, 5))
plt.plot(x, y_opt, 'b-', lw=2, label='Brachistochrone numérique')
plt.plot(x, np.linspace(0, y1, N+2), 'r--', label='Droite')
plt.gca().invert_yaxis()
plt.xlabel('$x$'); plt.ylabel('$y$')
plt.legend(); plt.title('Courbe brachistochrone')
plt.tight_layout(); plt.savefig('brachistochrone.pdf')
```

8.6 Problèmes isopérimétriques

Théorème 8.14 (Problème isopérimétrique)**.** *Parmi toutes les courbes fermées de périmètre fixé L, celle qui enferme l'aire maximale est le* ***cercle****.*

Définition 8.15 (Multiplicateur de Lagrange pour les fonctionnelles)**.** Pour un problème variationnel avec contrainte intégrale $\int_a^b G(x, y, y')\,dx = C$, on forme le lagrangien augmenté :

$$\tilde{J}[y] = \int_a^b \big[F(x, y, y') + \lambda G(x, y, y')\big]\,dx,$$

et on applique l'équation d'Euler–Lagrange à $\tilde{F} = F + \lambda G$.

8.7 Exercices

Exercice 8.1. Résoudre graphiquement et avec `linprog` le problème :

$$\begin{aligned} \max \quad & 5x_1 + 4x_2 \\ \text{s.c.} \quad & 6x_1 + 4x_2 \leq 24,\ x_1 + 2x_2 \leq 6,\ x_1, x_2 \geq 0. \end{aligned}$$

Exercice 8.2. Montrer que la méthode de descente de gradient converge pour une fonction f fortement convexe et L-lisse avec un pas $\alpha = 1/L$. Quelle est la vitesse de convergence ?

Exercice 8.3. Résoudre par les conditions KKT : $\min\ x_1^2 + x_2^2$ s.c. $x_1 + 2x_2 = 3$, $x_1 \geq 0$.

Exercice 8.4. Trouver l'équation de la chaînette (courbe d'un câble suspendu) en minimisant l'énergie potentielle gravitationnelle $J[y] = \int_{-a}^{a} \rho g y \sqrt{1 + y'^2}\, dx$ sous la contrainte de longueur fixée $\int \sqrt{1 + y'^2}\, dx = L$.

Exercice 8.5. Implémenter la méthode de Newton pour l'optimisation sans contraintes et comparer sa convergence avec la descente de gradient sur la fonction de Rosenbrock.

Exercice 8.6. Résoudre numériquement le problème de la brachistochrone pour différentes positions du point d'arrivée et comparer avec la solution analytique (cycloïde).

Chapitre 9

Modèles Stochastiques

« Le hasard n'est que la mesure de notre ignorance. »
— Henri Poincaré

Jusqu'ici, nos modèles étaient déterministes : connaissant l'état initial et les lois d'évolution, on pouvait prédire le futur avec certitude. Mais le monde réel est bruité. Les marchés financiers fluctuent, les gènes mutent aléatoirement, les molécules s'agitent sous l'effet thermique, et même les épidémies ne se propagent pas de manière déterministe. Modéliser ces phénomènes exige d'intégrer l'aléa dans les équations. Ce chapitre introduit les modèles stochastiques — des marches aléatoires aux équations différentielles stochastiques d'Itô — et montre comment le hasard, loin d'être un obstacle à la compréhension, devient un ingrédient essentiel de la modélisation. Les modèles stochastiques décrivent l'évolution de systèmes soumis à des fluctuations aléatoires, ce qui est essentiel en biologie, en théorie des files d'attente et en physique.

9.1 Processus de naissance et de mort

Définition 9.1 (Processus de naissance et de mort)**.** Un **processus de naissance et de mort** est un processus de Markov à temps continu $\{X(t)\}_{t\geq 0}$ à valeurs dans $\mathbb{N}$ dont les seules transitions possibles depuis l'état n sont :

- $n \to n+1$ (naissance) avec taux $\lambda_n \geq 0$,
- $n \to n-1$ (mort) avec taux $\mu_n \geq 0$, où $\mu_0 = 0$.

Intuition

Imaginez une population de bactéries dans une boîte de Pétri. Chaque bactérie se divise (naissance) ou meurt indépendamment des autres. Le nombre total de bactéries fluctue aléatoirement autour d'une tendance générale. Le processus de naissance et de mort capture exactement cette dynamique.

Proposition 9.2 (Équations de Kolmogorov)**.** Soit $p_n(t) = \mathbb{P}(X(t) = n)$. Les probabilités d'état satisfont le système d'équations différentielles :

$$\frac{dp_n}{dt} = \lambda_{n-1}\, p_{n-1}(t) + \mu_{n+1}\, p_{n+1}(t) - (\lambda_n + \mu_n)\, p_n(t), \quad n \geq 1,$$

avec $\dfrac{dp_0}{dt} = \mu_1\, p_1(t) - \lambda_0\, p_0(t)$.

Démonstration. On conditionne sur l'état à l'instant t et on utilise les propriétés markoviennes. En un intervalle $[t, t+h]$, avec h petit :

$$p_n(t+h) = p_n(t)\big(1 - (\lambda_n + \mu_n)h\big) + p_{n-1}(t)\,\lambda_{n-1}\,h + p_{n+1}(t)\,\mu_{n+1}\,h + o(h).$$

En soustrayant $p_n(t)$, divisant par h et passant à la limite $h \to 0$, on obtient l'équation annoncée. □

Théorème 9.3 (Distribution stationnaire)**.** *Si les taux vérifient* $\sum_{n=1}^{\infty} \prod_{k=0}^{n-1} \frac{\lambda_k}{\mu_{k+1}} < \infty$, *alors il existe une distribution stationnaire* $\pi = (\pi_n)_{n \geq 0}$ *donnée par :*

$$\pi_n = \pi_0 \prod_{k=0}^{n-1} \frac{\lambda_k}{\mu_{k+1}}, \qquad \pi_0 = \left(1 + \sum_{n=1}^{\infty} \prod_{k=0}^{n-1} \frac{\lambda_k}{\mu_{k+1}}\right)^{-1}.$$

Exemple 9.4. Considérons un processus de naissance pure avec $\lambda_n = \lambda\, n$ et $\mu_n = 0$ (modèle de Yule). Partant de $X(0) = 1$, la population croît exponentiellement : $\mathbb{E}[X(t)] = e^{\lambda t}$.

9.2 Théorie des files d'attente

Définition 9.5 (File M/M/1)**.** Une file **M/M/1** est un processus de naissance et de mort avec taux d'arrivée constant $\lambda_n = \lambda$ et taux de service constant $\mu_n = \mu$ pour $n \geq 1$. « M » désigne « markovien » (arrivées et services exponentiels).

Théorème 9.6 (Équilibre de la file M/M/1)**.** *Soit* $\rho = \lambda/\mu$ *l'intensité du trafic. Si* $\rho < 1$, *la file M/M/1 admet une distribution stationnaire géométrique :*

$$\pi_n = (1 - \rho)\,\rho^n, \quad n \geq 0.$$

Le nombre moyen de clients dans le système est $L = \dfrac{\rho}{1-\rho}$ *et le temps moyen de séjour est* $W = \dfrac{1}{\mu - \lambda}$.

Démonstration. Par le théorème 9.3, $\pi_n = \pi_0 \rho^n$. La condition de normalisation $\sum_{n=0}^{\infty} \pi_n = 1$ donne $\pi_0 = 1 - \rho$ lorsque $\rho < 1$. Le nombre moyen suit de $L = \sum_{n=0}^{\infty} n\,\pi_n = (1-\rho)\sum_{n=1}^{\infty} n\,\rho^n = \rho/(1-\rho)$. La formule de Little $L = \lambda W$ donne W. □

Formules clés

Résumé des files d'attente

$$\text{M/M/1} : \quad L = \frac{\rho}{1-\rho}, \quad W = \frac{1}{\mu - \lambda}, \quad L_q = \frac{\rho^2}{1-\rho}$$

$$\text{M/M/c} : \quad \pi_0 = \left[\sum_{k=0}^{c-1} \frac{(c\rho)^k}{k!} + \frac{(c\rho)^c}{c!\,(1-\rho)}\right]^{-1}, \quad \rho = \frac{\lambda}{c\mu} < 1$$

Définition 9.7 (File M/M/c)**.** Une file **M/M/c** possède c serveurs identiques en parallèle. Les taux de service effectifs sont $\mu_n = \min(n, c)\,\mu$.

Remarque 9.8. La formule d'Erlang C donne la probabilité d'attente dans une file M/M/c :

$$C(c, \lambda/\mu) = \frac{(c\rho)^c}{c!\,(1-\rho)} \cdot \pi_0.$$

Elle est largement utilisée dans le dimensionnement des centres d'appels.

9.3 Marches aléatoires

Définition 9.9 (Marche aléatoire simple)**.** Une **marche aléatoire simple** sur $\mathbb{Z}$ est une suite $(S_n)_{n\geq 0}$ définie par $S_0 = 0$ et $S_n = \sum_{k=1}^{n} X_k$, où les (X_k) sont i.i.d. avec $\mathbb{P}(X_k = +1) = p$ et $\mathbb{P}(X_k = -1) = 1 - p$.

Théorème 9.10 (Récurrence de la marche aléatoire)**.** *La marche aléatoire simple sur $\mathbb{Z}$ est récurrente si et seulement si $p = 1/2$. Plus généralement, la marche aléatoire simple sur $\mathbb{Z}^d$ est récurrente si et seulement si $d \leq 2$ (théorème de Pólya).*

Idée de la preuve pour $d = 1$. Si $p = 1/2$, la probabilité de retour à l'origine est $\sum_{n=1}^{\infty} \mathbb{P}(S_{2n} = 0) = \sum_{n=1}^{\infty} \binom{2n}{n} 2^{-2n}$. Par la formule de Stirling, $\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi n}}$, donc la série diverge, ce qui assure la récurrence. Si $p \neq 1/2$, $\mathbb{E}[X_k] = 2p - 1 \neq 0$ et la loi des grands nombres montre que $S_n \to \pm\infty$. □

Exemple 9.11. La marche aléatoire modélise la position d'une particule en suspension dans un fluide (mouvement brownien discret). Elle modélise également la fortune d'un joueur dans un jeu équitable.

9.4 Algorithme de Gillespie

Définition 9.12 (Algorithme de Gillespie (SSA))**.** L'**algorithme de simulation stochastique** (SSA) de Gillespie est une méthode exacte pour simuler la trajectoire d'un système chimique stochastique. À chaque étape, on détermine :

1. le temps τ avant la prochaine réaction (exponentiel de paramètre $a_0 = \sum_j a_j$),
2. la réaction j qui se produit (avec probabilité a_j / a_0),

où a_j est la *propensité* de la réaction j.

Attention

L'algorithme de Gillespie simule *chaque* réaction individuellement. Pour des systèmes avec un grand nombre de molécules, il devient très coûteux. On préfère alors des méthodes approchées comme le τ-leaping.

Exemple 9.13. Considérons la réaction de production-dégradation d'un ARN messager : $\emptyset \xrightarrow{k_1} \text{ARN} \xrightarrow{k_2} \emptyset$, avec propensités $a_1 = k_1$ et $a_2 = k_2\, n$ où n est le nombre de molécules d'ARN. À l'équilibre stochastique, n suit une loi de Poisson de paramètre k_1/k_2.

```python
import numpy as np

def gillespie_production_degradation(k1, k2, n0, t_max):
    """Simulation de Gillespie pour production-degradation."""
    t, n = 0.0, n0
    times, states = [t], [n]
    while t < t_max:
        a1 = k1            # propension production
        a2 = k2 * n        # propension degradation
```

```
        a0 = a1 + a2
        if a0 == 0:
            break
        tau = np.random.exponential(1.0 / a0)
        t += tau
        if np.random.uniform() < a1 / a0:
            n += 1          # production
        else:
            n -= 1          # degradation
        times.append(t)
        states.append(n)
    return np.array(times), np.array(states)
```

9.5 Simulation de Monte Carlo

Définition 9.14 (Méthode de Monte Carlo)**.** Une **méthode de Monte Carlo** est une technique de calcul qui utilise des échantillons aléatoires pour estimer une quantité déterministe. Pour estimer $\theta = \mathbb{E}[g(X)]$, on génère $X_1, \dots, X_N$ i.i.d. selon la loi de X et on calcule :

$$\hat{\theta}_N = \frac{1}{N} \sum_{i=1}^{N} g(X_i).$$

Théorème 9.15 (Convergence de Monte Carlo)**.** *Par la loi forte des grands nombres,* $\hat{\theta}_N \to \theta$ *p.s. De plus, par le TCL, l'erreur d'estimation est d'ordre* $O(1/\sqrt{N})$ *:*

$$\sqrt{N}\,(\hat{\theta}_N - \theta) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \mathrm{Var}(g(X))).$$

Intuition

La convergence en $O(1/\sqrt{N})$ est *indépendante de la dimension* du problème. C'est l'avantage majeur de Monte Carlo par rapport aux méthodes déterministes (quadrature), dont la convergence se dégrade avec la dimension (« fléau de la dimension »).

Proposition 9.16 (Réduction de variance par variables antithétiques)**.** Si g est monotone et $X \sim \mathcal{U}(0,1)$, l'estimateur

$$\hat{\theta}_N^{\text{anti}} = \frac{1}{N} \sum_{i=1}^{N/2} \frac{g(U_i) + g(1-U_i)}{2}$$

a une variance inférieure ou égale à celle de l'estimateur brut.

Exercice 9.1. Estimer $\int_0^1 e^x\,dx$ par Monte Carlo avec $N = 10\,000$ échantillons. Comparer la précision obtenue avec et sans variables antithétiques.

Exercice 9.2. Simuler une file M/M/1 avec $\lambda = 3$ et $\mu = 5$ sur un horizon de temps $T = 1000$. Estimer le nombre moyen de clients dans le système et comparer avec la valeur théorique $L = 3/2$.

Exercice 9.3. Implémenter l'algorithme de Gillespie pour le modèle de Lotka–Volterra stochastique : $X \xrightarrow{\alpha} 2X$, $\quad X+Y \xrightarrow{\beta} 2Y$, $\quad Y \xrightarrow{\gamma} \emptyset$. Comparer les trajectoires stochastiques au modèle déterministe.

Chapitre 10

Études de Cas et Projets

« Tous les modèles sont faux, mais certains sont utiles. »
— George E. P. Box

Cette maxime de George Box résume à elle seule la philosophie de la modélisation mathématique. Un modèle n'est pas la réalité : c'est une simplification délibérée, conçue pour capturer l'essentiel d'un phénomène tout en ignorant les détails non pertinents. Mais un bon modèle est *utile* : il prédit, il explique, il guide la décision. Ce chapitre final met en pratique l'ensemble des outils développés dans les chapitres précédents à travers des études de cas complètes : de la formulation du problème à l'analyse de sensibilité, en passant par le choix du modèle, sa calibration et sa validation.

Ce chapitre met en œuvre les outils développés dans les chapitres précédents à travers cinq études de cas concrètes. Chaque étude illustre le cycle complet de modélisation : formulation, analyse mathématique, calibration sur données et validation.

10.1 Modélisation épidémiologique : SIR et SEIR

Définition 10.1 (Modèle SIR)**.** Le modèle **SIR** (Susceptible–Infecté–Rétabli) de Kermack–McKendrick décrit la propagation d'une maladie dans une population de taille N :

$$\frac{dS}{dt} = -\beta \frac{SI}{N}, \quad \frac{dI}{dt} = \beta \frac{SI}{N} - \gamma I, \quad \frac{dR}{dt} = \gamma I,$$

où $\beta > 0$ est le taux de transmission et $\gamma > 0$ le taux de guérison.

Théorème 10.2 (Seuil épidémique)**.** *Soit $R_0 = \beta/\gamma$ le* nombre de reproduction de base. *Si $R_0 > 1$ et $S(0) \approx N$, alors $I(t)$ croît initialement (épidémie). Si $R_0 < 1$, le nombre d'infectés décroît monotonement.*

Démonstration. Au début de l'épidémie, $S \approx N$, donc $dI/dt \approx (\beta - \gamma)\, I = \gamma(R_0 - 1)\, I$. Si $R_0 > 1$, cette équation linéarisée donne une croissance exponentielle de I. Si $R_0 < 1$, I décroît exponentiellement. □

Intuition

Le nombre R_0 représente le nombre moyen de personnes qu'un individu infecté contamine durant toute sa période infectieuse (de durée moyenne $1/\gamma$). L'épidémie

CHAPITRE 10. ÉTUDES DE CAS ET PROJETS

se propage uniquement si chaque malade contamine en moyenne plus d'une personne ($R_0 > 1$).

Définition 10.3 (Modèle SEIR)**.** Le modèle **SEIR** ajoute un compartiment « Exposé » (incubation) :

$$\frac{dS}{dt} = -\beta \frac{SI}{N}, \quad \frac{dE}{dt} = \beta \frac{SI}{N} - \sigma E, \quad \frac{dI}{dt} = \sigma E - \gamma I, \quad \frac{dR}{dt} = \gamma I,$$

où $1/\sigma$ est la durée moyenne d'incubation.

Formules clés

Paramètres épidémiologiques clés

$$R_0 = \frac{\beta}{\gamma} \quad \text{(SIR)}, \qquad R_0 = \frac{\beta}{\gamma} \quad \text{(SEIR, même formule)}$$

$$R_{\text{eff}}(t) = R_0 \cdot \frac{S(t)}{N} \quad \text{(nombre de reproduction effectif)}$$

Taille finale : S_∞ est solution de $S_\infty = S_0\, e^{-R_0(N - S_\infty)/N}$

Exemple 10.4. Calibration du modèle SIR sur les données de la grippe dans un pensionnat anglais (1978) : 763 élèves, données journalières sur 14 jours. Par moindres carrés, on estime $\beta \approx 1.66$ et $\gamma \approx 0.44$, soit $R_0 \approx 3.78$.

```python
import numpy as np
from scipy.integrate import odeint
from scipy.optimize import minimize

def sir_model(y, t, beta, gamma, N):
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return [dSdt, dIdt, dRdt]

# Donnees observees (infectes par jour)
data_I = np.array([1, 3, 8, 28, 76, 222, 293, 257, 237,
                   192, 126, 70, 28, 12])
t_data = np.arange(len(data_I))
N = 763

def objective(params):
    beta, gamma = params
    y0 = [N - 1, 1, 0]
    sol = odeint(sir_model, y0, t_data, args=(beta, gamma, N))
    return np.sum((sol[:, 1] - data_I)**2)

result = minimize(objective, [1.5, 0.5], method='Nelder-Mead')
```

10.2 Dynamique des populations : Lotka–Volterra

Définition 10.5 (Modèle de Lotka–Volterra)**.** Le modèle proies–prédateurs de Lotka–Volterra s'écrit :

$$\frac{dx}{dt} = \alpha\, x - \beta\, x\, y, \qquad \frac{dy}{dt} = \delta\, x\, y - \gamma\, y,$$

où $x(t)$ est la population de proies, $y(t)$ celle de prédateurs, et $\alpha, \beta, \gamma, \delta > 0$.

Proposition 10.6 (Intégrale première)**.** Le système de Lotka–Volterra admet l'intégrale première :

$$H(x, y) = \delta\, x - \gamma\, \ln x + \beta\, y - \alpha\, \ln y.$$

Les trajectoires dans le plan de phase sont donc des courbes fermées entourant le point d'équilibre $(\gamma/\delta,\, \alpha/\beta)$.

Démonstration. On calcule $\frac{dH}{dt} = \delta\, \dot{x} - \frac{\gamma}{x}\, \dot{x} + \beta\, \dot{y} - \frac{\alpha}{y}\, \dot{y}$. En substituant les équations du système, tous les termes s'annulent : $dH/dt = 0$. □

Exemple 10.7. Les données historiques de la Compagnie de la Baie d'Hudson (fourrures de lièvres et de lynx, 1845–1935) montrent des oscillations déphasées caractéristiques du modèle de Lotka–Volterra. La calibration par moindres carrés donne des périodes d'environ 10 ans.

Attention

Le modèle de Lotka–Volterra classique est structurellement instable : il ne possède pas de cycle limite attractif. Toute perturbation modifie l'amplitude des oscillations. Pour un modèle plus réaliste, on ajoute une capacité de charge (logistique) ou une réponse fonctionnelle de type Holling.

10.3 Modèles de trafic routier

Définition 10.8 (Modèle LWR)**.** Le modèle de **Lighthill–Whitham–Richards** (LWR) est une loi de conservation scalaire pour la densité de véhicules $\rho(x, t)$:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}\big[Q(\rho)\big] = 0,$$

où $Q(\rho) = \rho\, v(\rho)$ est le flux et $v(\rho)$ est la vitesse en fonction de la densité.

Proposition 10.9 (Relation de Greenshields)**.** Le modèle de Greenshields suppose $v(\rho) = v_{\max}(1 - \rho/\rho_{\max})$, ce qui donne un flux parabolique :

$$Q(\rho) = v_{\max}\, \rho \left(1 - \frac{\rho}{\rho_{\max}}\right).$$

Le flux maximal (capacité) est $Q_{\max} = v_{\max}\, \rho_{\max}/4$ atteint en $\rho = \rho_{\max}/2$.

Remarque 10.10. Les solutions du modèle LWR peuvent développer des discontinuités (*chocs*), qui correspondent physiquement à la formation de bouchons. La condition de Rankine–Hugoniot donne la vitesse du choc : $s = \dfrac{Q(\rho_R) - Q(\rho_L)}{\rho_R - \rho_L}$.

10.4 Introduction à la modélisation climatique

Définition 10.11 (Modèle de bilan énergétique)**.** Le modèle de **bilan énergétique** de Budyko–Sellers décrit la température moyenne T de la Terre :

$$C\,\frac{dT}{dt} = (1-\alpha(T))\,Q - \varepsilon\,\sigma\,T^4,$$

où C est la capacité thermique, Q le flux solaire incident, $\alpha(T)$ l'albédo (dépendant de la température via la couverture de glace) et $\varepsilon\,\sigma\,T^4$ le rayonnement infrarouge émis.

Théorème 10.12 (Bifurcation et « Terre boule de neige »)**.** *Le modèle de Budyko–Sellers admet génériquement trois équilibres : un état chaud (peu de glace), un état « boule de neige » (Terre entièrement glacée) et un équilibre instable intermédiaire. Lorsque le flux solaire Q diminue, une bifurcation noeud-selle conduit à une transition brutale vers l'état glacé.*

Intuition

L'albédo de la glace est élevé (~ 0.7) tandis que celui de l'océan est faible (~ 0.06). Plus il fait froid, plus la glace s'étend, plus la Terre refléchit le soleil, plus il fait froid… C'est une rétroaction positive qui peut mener à un emballement.

10.5 Comparaison et validation de modèles

Définition 10.13 (Critère d'information d'Akaike (AIC))**.** Soit $\hat{L}$ la vraisemblance maximale d'un modèle à k paramètres. Le **critère d'information d'Akaike** est :

$$\mathrm{AIC} = -2\,\ln \hat{L} + 2k.$$

On privilégie le modèle avec l'AIC le plus faible.

Définition 10.14 (Critère d'information bayésien (BIC))**.** Le **critère BIC** (ou critère de Schwarz) est :

$$\mathrm{BIC} = -2\,\ln \hat{L} + k\,\ln n,$$

où n est le nombre d'observations. Le BIC pénalise davantage la complexité que l'AIC pour $n \geq 8$.

Proposition 10.15 (Validation croisée)**.** La **validation croisée** k-fold estime l'erreur de prédiction en découpant les données en k parties. L'erreur estimée est :

$$\mathrm{CV}(k) = \frac{1}{k}\sum_{j=1}^{k} \mathrm{Erreur}_j,$$

où Erreur_j est l'erreur du modèle ajusté sur les $k-1$ autres parties et évalué sur la partie j.

Attention

Un bon ajustement aux données d'entraînement ne garantit pas un bon pouvoir prédictif. Le *surapprentissage* (overfitting) se produit lorsque le modèle capture le bruit plutôt que le signal. Toujours valider sur des données indépendantes.

Formules clés

Résumé des critères de comparaison

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \quad \text{(coefficient de détermination)}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad \text{(erreur quadratique moyenne)}$$

$$\text{AIC} = -2\ln \hat{L} + 2k, \qquad \text{BIC} = -2\ln \hat{L} + k \ln n$$

Exercice 10.1. Implémenter le modèle SEIR avec les paramètres de la COVID-19 ($\sigma^{-1} \approx 5{,}2$ jours, $\gamma^{-1} \approx 10$ jours, $R_0 \approx 2{,}5$). Simuler la dynamique pour une ville de 100 000 habitants et étudier l'effet du confinement (R_0 réduit).

Exercice 10.2. Calibrer un modèle de Lotka–Volterra sur les données lièvres–lynx de la Compagnie de la Baie d'Hudson. Estimer les paramètres par moindres carrés non linéaires et évaluer la qualité de l'ajustement par R^2 et RMSE.

Exercice 10.3. Comparer les modèles SIR, SEIR et SIR avec vaccination sur un jeu de données épidémiques. Utiliser l'AIC et le BIC pour déterminer quel modèle est le plus parcimonieux.

Exercice 10.4. Résoudre numériquement le modèle LWR avec la relation de Greenshields pour simuler la formation d'un bouchon à un feu rouge. Utiliser un schéma de Godunov et visualiser l'évolution de la densité $\rho(x, t)$ en espace et en temps.

Bibliographie

[1] Murray, J.D., *Mathematical Biology*, 3rd ed., 2 vols., Springer, 2002–2003.

[2] Strogatz, S.H., *Nonlinear Dynamics and Chaos*, 2nd ed., Westview, 2015.

[3] Fowler, A.C., *Mathematical Models in the Applied Sciences*, Cambridge, 1997.