

MODÉLISATION NUMÉRIQUE EN PHYSIQUE

Notes de cours
par

Paul Charbonneau
Département de Physique
Université de Montréal

Décembre 2016

MINI-PRÉFACE

Ces notes contiennent toute la matière couverte dans le cadre du cours PHY-3075, **Modélisation Numérique en Physique**, offert par le Département de Physique de l'Université de Montréal.

Vous avez en main la version 2.0 de ces Notes de cours, dont la première mouture fut écrite et testée l'an passé. Je tiens à remercier la cohorte étudiante de l'hiver 2016, dont les commentaires et suggestions (et détection d'erreurs ici et là) ont conduit à la version améliorée (on l'espère) de cette année. Remerciements particuliers à Guillaume Emond, François Hardy, et François Labonville; ainsi qu'à mes collègues Jean-Yves Lapointe et Jean-François Arguin, pour m'avoir éclairé la lanterne sur l'électrophysiologie des membranes et sur le boson de Higgs, respectivement; à Louis-Guillaume Gagnon pour m'avoir dirigé vers le concours d'apprentissage machine pour la détection du boson de Higgs, et finalement à mon ex-postdoc Antoine Strugarek pour son aide toujours efficace et éclairée face aux subtilités du Python.

Il reste certainement beaucoup à améliorer. Je compte sur vos commentaires et suggestions, sur le cours en général, et sur ces Notes en particulier.

Table des Matières

1 Équations différentielles ordinaires	7
1.1 Systèmes d'équations couplées	7
1.2 Les méthodes de Runge-Kutta	8
1.3 Exemple 1: chaînes de désintégration nucléaire	9
1.4 Exemple 2: un système nonlinéaire réactif	15
1.5 Runge-Kutta avec pas adaptif	19
1.6 Transmission des signaux dans les neurones	19
1.6.1 Les neurones	23
1.6.2 L'équation du câble	25
1.6.3 Solutions à l'équation du câble	27
1.7 Projet: le modèle de Hodgkin-Huxley	29
1.7.1 Les canaux ioniques et le courant transmembranaire	29
1.7.2 Formulation mathématique du modèle de Hodgkin-Huxley	29
1.7.3 Traitement numérique	31
1.7.4 Le potentiel d'action	31
1.7.5 Explorations numériques	35
1.8 Bibliographie	36
2 Équations différentielles aux dérivées partielles	37
2.1 Méthodes explicites: diffusion	38
2.1.1 L'équation de diffusion	38
2.1.2 Forward-in-Time, Centered-in-Space (FTCS)	38
2.2 Exemple: formation de structures par réaction-diffusion	39
2.3 Méthodes explicites: advection	46
2.3.1 Forward-in-Time, Centered-in-Space (FTCS), bis	48
2.3.2 Analyse de stabilité de von Newmann	48
2.3.3 La méthode de Lax	50
2.3.4 La méthode de Lax-Wendroff	52
2.4 Méthodes implicites	53
2.4.1 Méthode d'Euler implicite	53
2.4.2 La méthode de Crank-Nicolson	55
2.5 Quel algorithme utiliser ?	56
2.6 Au delà du 1D	57
2.7 Techniques de discrétisation spatiale	60
2.7.1 Différences finies	60
2.7.2 Éléments finis	61
2.7.3 Méthodes spectrales	61
2.7.4 Méthodes de volumes finis	61
2.8 Les plasmas en tokamak	62
2.8.1 La fusion nucléaire	62
2.8.2 Trajectoires de particules chargées dans un champ magnétique	63
2.8.3 Le confinement plasma en tokamak	64

2.9	Projet: barrière de confinement plasma	67
2.9.1	Transport turbulent et diffusion nonlinéaire	67
2.9.2	Établissement de l'effet barrière	69
2.9.3	Traitement numérique	69
2.9.4	Explorations numériques	70
2.10	Bibliographie	72
3	Calcul sur réseau	75
3.1	Calculs sur réseaux	75
3.2	Exemple: les tremblements de terre	78
3.2.1	La tectonique des plaques (en bref)	78
3.2.2	Le modèle de Burridge-Knopoff	80
3.2.3	Traitement numérique	84
3.2.4	Une simulation représentative	84
3.2.5	Comportement du modèle	90
3.3	Le ferromagnétisme	92
3.4	Le modèle d'Ising	93
3.4.1	Traitement numérique	94
3.4.2	L'algorithme de Metropolis	95
3.4.3	Résultats représentatifs	97
3.4.4	La température de Curie	100
3.4.5	Magnétisation extérieure	102
3.5	Projet: gravure magnétique par laser	102
3.6	Bibliographie	107
4	Les problèmes à N-corps	109
4.1	La dynamique moléculaire	109
4.1.1	Le potentiel de Lennard-Jones	109
4.1.2	Adimensionalisation du problème	110
4.2	Traitement numérique	111
4.2.1	L'algorithme de Verlet	112
4.2.2	Conditions initiales et limites	112
4.2.3	Dynamique moléculaire en Python	113
4.2.4	Quelques test simples	117
4.3	Exemple: simulation d'un liquide	121
4.3.1	Considérations thermodynamiques	121
4.3.2	Condensation, solidification et liquéfaction	122
4.3.3	Fluides stratifiés par la gravité	124
4.4	Matière sombre et rotation des galaxies	128
4.5	Simulation dynamique d'une galaxie	131
4.5.1	Adimensionalisation	131
4.5.2	Conditions limites et initiales	132
4.5.3	Calcul du potentiel gravitationnel	133
4.5.4	Astuces algorithmiques	140
4.5.5	Quelques résultats représentatifs	140
4.6	Projet: la matière sombre	142
4.7	Bibliographie	143
5	Les algorithmes évolutifs	145
5.1	Optimisation locale versus globale	145
5.1.1	Méthodes de grimpe	145
5.1.2	La grimpe stochastique	147
5.2	La puissance de la sélection cumulative	151
5.3	Un algorithme évolutif de base	156

5.3.1	L'algorithme	156
5.3.2	Initialisation	156
5.3.3	Évaluation	157
5.3.4	Classement	157
5.3.5	Sélection	157
5.3.6	Reproduction	159
5.3.7	Remplacement	160
5.3.8	Test de convergence	160
5.4	La diffraction 2D, troisième tour...	161
5.5	Les algorithmes génétiques	161
5.6	Modélisation d'une orbite Keplerienne	163
5.6.1	Les étoiles binaires	163
5.6.2	De la vitesse radiale à l'orbite Keplerienne	164
5.6.3	Une solution par algorithme évolutif	166
5.7	Projet: mutation adaptive et l'orbite de ρ CrB	167
5.8	Bibliographie	169
6	Les réseaux de neurones artificiels	171
6.1	Les réseaux de neurones (artificiels)	172
6.2	Entraîner et tester le réseau	175
6.2.1	Un problème de classification de structures	175
6.2.2	La rétropropagation	177
6.2.3	La phase d'entraînement	179
6.2.4	La phase de test	182
6.3	Le boson de Higgs en une page	185
6.4	Projet: trouver le boson de Higgs	187
6.5	Bibliographie	188



Chapitre 1

Équations différentielles ordinaires

La plupart des logiciels mathématiques et des librairies associées aux langages de programmation contiennent un ou plusieurs intégrateurs “boîtes noires” pour les équations différentielles ordinaires (ci-après EDO), qui sont très performants et faciles d’utilisation. Pourquoi donc un chapitre sur ce sujet? Parce qu’il est important de comprendre ce qu’il y a dans la boîte noire, et d’apprécier qu’il existe des situations où une application à l’aveuglette de ces boîtes noires peut s’avérer désastreuse.

Après un bref rappel (surtout notationnel) sur la reformulation des EDOs d’ordre élevé en systèmes d’EDO couplées d’ordre un (§1.1), ce chapitre introduit un intégrateur robuste, soit la méthode dite de Runge-Kutta d’ordre 4 (§§1.2 et 1.5), appliquée ensuite à trois exemples physiques: les chaînes de désintégration radioactives (§1.3), les systèmes réactifs autocatalytiques (§1.4), et la propagation des signaux dans les neurones (§§1.6 et 1.7).

1.1 Systèmes d’équations couplées

On considère dans ce qui suit une EDO prototypique de la forme

$$\frac{du}{dt} = g(t, u) , \quad (1.1)$$

où en général la fonction $g(t, u)$ peut dépendre de t ou u de manière nonlinéaire. Géométriquement, g représente la pente de la fonction $u(t)$. On se limitera ici à des situations où une condition initiale est donnée à $t = 0$, et le problème consiste à avancer la solution $u(t)$ dans le temps.

On peut généraliser l’éq. (1.1) à un système d’EDOs couplées; par exemple le très fameux système chaotique de Lorenz ($u \equiv [x, y, z]$):

$$\frac{dx}{dt} = -\sigma(x + y) , \quad (1.2)$$

$$\frac{dy}{dt} = -xz + rx - y , \quad (1.3)$$

$$\frac{dz}{dt} = xy - bz , \quad (1.4)$$

où σ , r et b sont des constantes données, peut s’écrire de manière compacte comme

$$\frac{d}{dt} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_u = \underbrace{\begin{pmatrix} -\sigma(x + y) \\ -xz + rx - y \\ xy - bz \end{pmatrix}}_{g(u)} . \quad (1.5)$$

Les système d'EDO impliquant des dérivées plus élevées que un peuvent habituellement se convertir facilement en un système d'EDOs couplées d'ordre 1; considérons par exemple la célèbre Loi de Newton, écrite sous forme différentielle:

$$m \frac{\partial^2 \mathbf{x}}{\partial t^2} = \mathbf{F}(\mathbf{x}, \dots), \quad \mathbf{x} \equiv (x, y, z), \quad (1.6)$$

où les “...” indiquent que la force pourrait fort bien dépendre d'autre chose que la simple position de l'objet (e.g., sa vitesse). On introduit maintenant la vitesse comme variable secondaire:

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{v}, \quad \mathbf{v} \equiv (u, v, w), \quad (1.7)$$

ce qui permet d'exprimer (1.6) sous la forme:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \\ z \\ u \\ v \\ w \end{pmatrix} = \begin{pmatrix} u \\ v \\ w \\ F_x/m \\ F_y/m \\ F_z/m \end{pmatrix}, \quad (1.8)$$

soit un système de six EDOs d'ordre 1, où en général les composantes de la force peuvent dépendre des six variables du problème et du temps t . Dans tous les cas, ce qui importe c'est que les quantités aux membres de droite puissent être calculées.

1.2 Les méthodes de Runge-Kutta

Fondamentalement, les méthodes numériques de solutions d'EDO sont toutes basées sur le développement en série de Taylor. Commençons par introduire la discrétisation suivante pour la variable indépendante t :

$$t \rightarrow t_n, \quad t_{n+1} = t_n + h, \quad n = 0, 1, 2, \dots \quad (1.9)$$

Le développement de Taylor s'écrit alors sous la forme:

$$u(t+h) = u(t) + h \frac{du}{dt} + \frac{h^2}{2} \frac{d^2 u}{dt^2} + \frac{h^3}{6} \frac{d^3 u}{dt^3} + \dots = \sum_{n=0}^{\infty} \frac{h^n}{n!} \frac{d^n u}{dt^n}. \quad (1.10)$$

Si on ne conserve que les premier deux termes au membre de droite de ce développement, et utilisant (1.1) pour remplacer du/dt , on arrive immédiatement à la *méthode d'Euler*:

$$u_{n+1} = u_n + h g(t_n, u_n) + O(h^2) \quad (1.11)$$

où le terme $O(h^2)$ indique que tous les termes de puissance h^2 et plus dans (1.10) ont été négligés. La méthode d'Euler est par conséquent dite “d'ordre un”. Conceptuellement simple et codable en 1 ligne d'instructions dans une boucle, la méthode d'Euler s'avère plutôt imprécise. Par exemple, il est facile de vérifier qu'elle surestime toujours une fonction convexe, et sous-estime une fonction concave.

La précision de la méthode d'Euler peut être améliorée en produisant un meilleur estimé de la pente. Par exemple, on peut utiliser la moyenne des pentes aux pas n et $n+1$:

$$u_{n+1} = u_n + h \frac{1}{2} (g(t_n, u_n) + g(t_{n+1}, u_{n+1})), \quad (1.12)$$

mais comme la valeur u_{n+1} n'est pas connue, on l'estime à l'aide de la méthode d'Euler, ce qui conduit à:

$$u_{n+1} = u_n + h \frac{1}{2} (g(t_n, u_n) + g(t_{n+1}, u_n + h g(t_n, u_n))). \quad (1.13)$$

C'est la *méthode de Heun*. Une autre option est d'utiliser la méthode d'Euler pour estimer la pente à mi-pas:

$$u_{n+1} = u_n + h g(t_n + \frac{h}{2}, u_n + \frac{h}{2} g(t_n, u_n)) ; \quad (1.14)$$

C'est la méthode dite de mi-pas. Notons que les algorithmes (1.13) et (1.14) deviennent identiques si g est une fonction linéaire de u .

Considérons maintenant les quatre estimés suivants de la pente:

$$g_1 = g(t_n, u_n) , \quad (1.15)$$

$$g_2 = g(t_n + \frac{h}{2}, u_n + \frac{h}{2} g_1) , \quad (1.16)$$

$$g_3 = g(t_n + \frac{h}{2}, u_n + \frac{h}{2} g_2) , \quad (1.17)$$

$$g_4 = g(t_n + h, u_n + h g_3) ; \quad (1.18)$$

le premier (g_1) est le même que celui utilisé par la méthode d'Euler. Le second (g_2) utilise ce premier estimé (g_1) pour calculer la pente non pas à t_n , mais plutôt à $t_n + h/2$, soit à mi-chemin du pas de temps devant être effectué. Le troisième (g_3) estimé répète ce calcul, mais utilisant cette fois ce nouvel estimé (g_2). Le dernier (g_4) utilise l'estimé g_3 pour évaluer la pente à $t_n + h$, soit sur le pas complet. Le calcul de u_{n+1} se fait encore une fois en ne conservant que les premier deux termes du développement de Taylor, mais utilise cette fois une moyenne pondérée de nos quatre estimés de la pente:

$$u_{n+1} = u_n + \frac{h}{6} (g_1 + 2g_2 + 2g_3 + g_4) + O(h^5) . \quad (1.19)$$

C'est la *méthode de Runge-Kutta*. Les choix des estimés de pente et de leur pondération dans l'éq. (1.19) peuvent paraître arbitraire, mais il sont cruciaux: ils sont tels qu'en bout de ligne (i.e., l'évaluation de (1.19)), les termes des développement de Taylor leur étant associés s'annulent aux ordre h^2 , h^3 et h^4 , impliquant que la méthode de Runge-Kutta est d'ordre 4 même si elle n'est basée que sur les termes en h^0 et h^1 du développement de Taylor. D'autres combinaisons d'estimés de pente et de pondération peuvent conduire au même effet, mais les éqs. (1.15)–(1.19) représentent la version classique de la méthode de Runge-Kutta, et c'est celle qui sera utilisée dans tout ce qui suit. Notons finalement que la méthode de mi-pas (1.14) est, fondamentalement, une méthode Runge-Kutta d'ordre deux.

1.3 Exemple 1: chaines de désintégration nucléaire

Que ce soit dans le cadre de la saine gestion des déchets de centrales nucléaires ou la production de radioisotopes pour usage médical, le calcul des chaines de désintégration nucléaire n'est pas un sujet qui est prêt de se démoder. Il nous offre de plus un premier exemple simple d'application de la méthode de Runge-Kutta.

La radioactivité naturelle a été découverte accidentellement en 1896 par Henri Becquerel (1852-1908), qui remarqua le noircissement d'une plaque photographique par un sel d'uranium (voir Figure 1.1). Le sujet attira rapidement l'attention de ses collègues Pierre Curie (1859–1906) et Marie Skłodowska-Curie (1867-1934), le trio consacrant la décennie suivante à l'étude de ce phénomène, récoltant au passage le Prix Nobel de Physique en 1903.

Les premières études expérimentales systématiques identifièrent trois modes de désintégration, en fonction de la déviation des faisceaux de particules émises par les noyaux radioactifs durant leur passage dans un champ électrique ou magnétique. Ces trois modes sont:

1. Désintégration- α : émission d'un noyau de ${}^4\text{He}$. Cette désintégration est associée à la force nucléaire forte, et se fait par effet tunnel à travers la barrière de potentiel nucléaire:

$$(A, Z) \rightarrow (A - 4, Z - 2) + {}^4\text{He} , \quad (1.20)$$

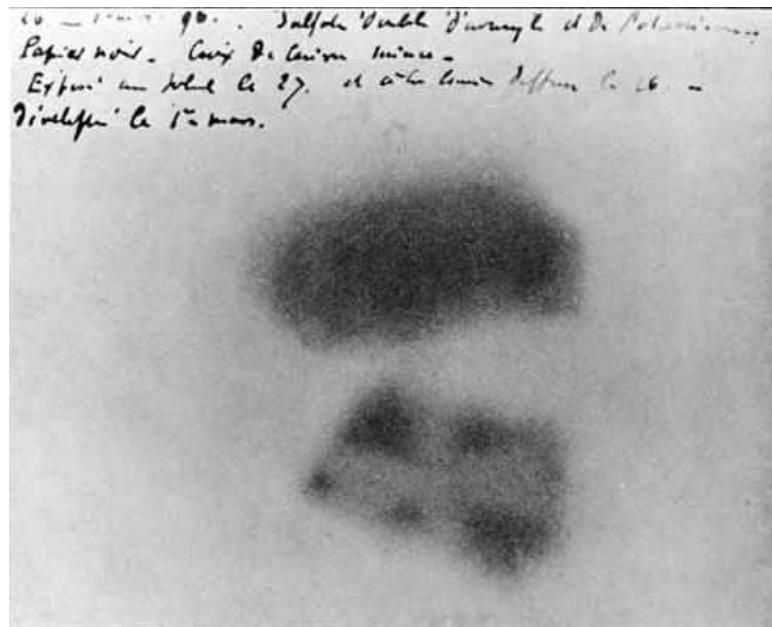
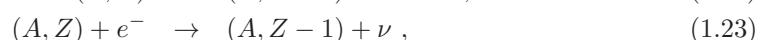
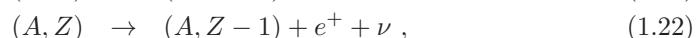
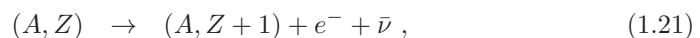


Figure 1.1: Noircissement d'une plaque photographique de Becquerel par deux échantillons de sels d'uranium déposés sur la plaque. Celui du bas avait été déposé sur une croix de Malte métallique ayant protégé partiellement la plaque, produisant une ombre (en négatif) clairement visible ici. Image en domaine public tirée de la page Wikipedia sur Henri Becquerel: https://en.wikipedia.org/wiki/Henri_Becquerel.

2. Désintégration- β : il existe trois formes communes de cette désintégration, contrôlée par la force nucléaire faible:



soit, respectivement, l'émission d'un électron, d'un positron, ou encore la capture d'un électron atomique. Ces trois formes de désintégration- β exigent l'émission d'un neutrino ou antineutrino afin de conserver le nombre leptonique.

3. La désintégration- γ , soit l'émission d'un photon très énergétique suite à la désexcitation spontanée d'un noyau s'étant retrouvé dans un état métastable, habituellement suite à une désintégration antérieure de type α ou β :



Le spectre de ces rayons γ nous donne accès à la structure quantique des orbitales nucléaires, tout comme la spectroscopie classique nous renseigne sur la structure atomique.

La Figure 1.2 illustre la répartition des différents isotopes nucléaires connus dans le plan $[Z, N]$, où sur ce diagramme Z est le nombre de protons et N le nombre de neutrons. Le mode primaire de désintégration des isotopes instables est indiqué par le code couleur. On notera que les isotopes stables (en noir) contiennent plus de neutrons que de protons, dès que $Z > 20$. La physique nucléaire sous-jacente peut bien être compliquée, mais du point de vue empirique la désintégration nucléaire est simple: quel que soit son mode de désintégration, un noyau

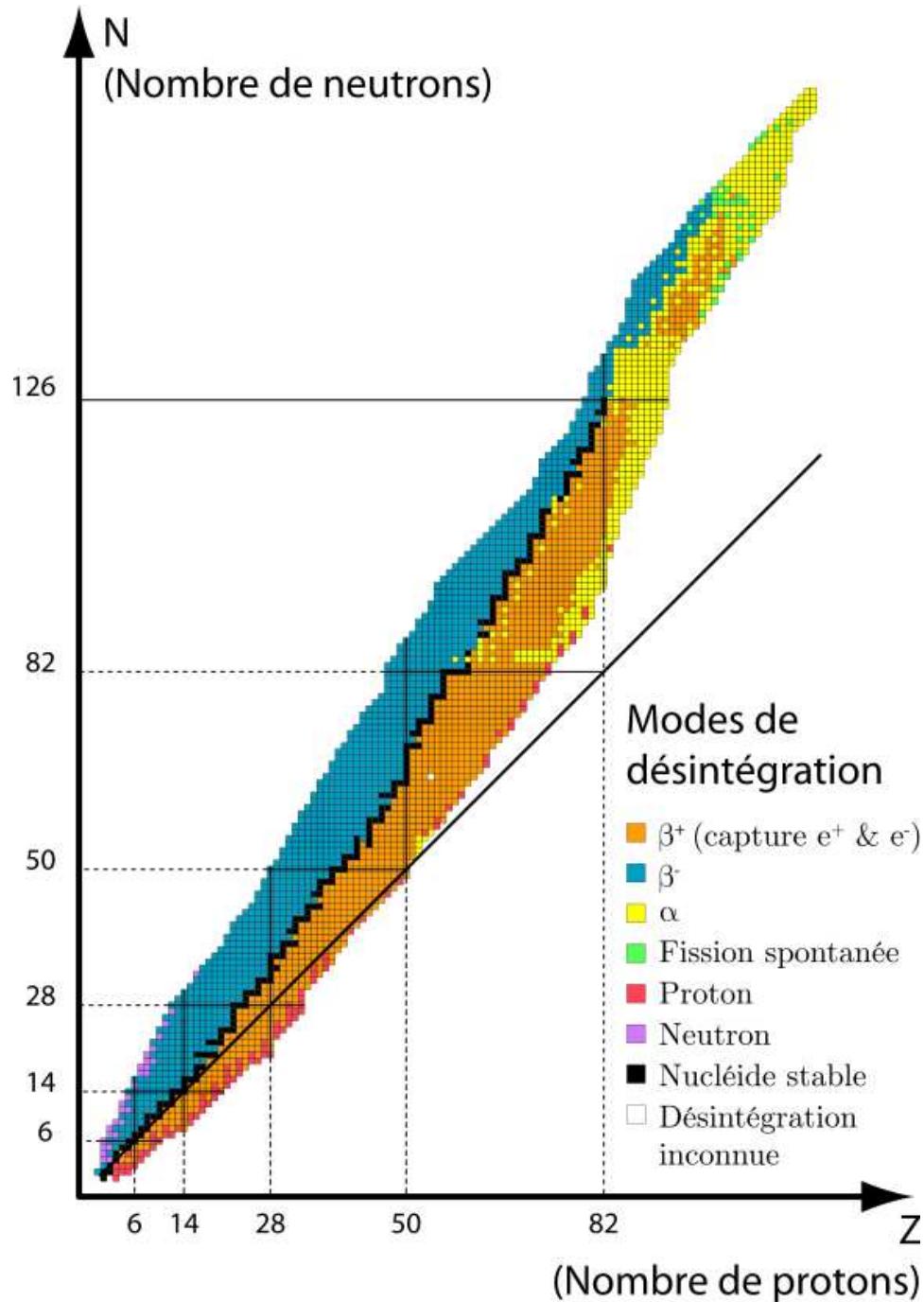


Figure 1.2: Les isotopes nucléaires et leurs modes de désintégration primaire. Ici le mode codé β^- inclut les deux modes $Z \rightarrow Z - 1$ dans les éqs. (1.21), soit l'émission d'un positron et la capture électronique. Image en domaine public tirée de la page Wikipedia sur la désintégration radioactive: https://en.wikipedia.org/wiki/Radioactive_decay.

instable donné à une probabilité fixe de se désintégrer, et cette probabilité est indépendante de la présence d'autres noyaux dans les environs, et des conditions physiques du milieu. Même dans l'environnement extrême d'une explosion supernova, le ^{56}Ni se désintègre au même taux que celui mesuré en laboratoire. Dans le cas d'un seul canal de désintégration, le taux de variations du nombre N de noyaux radioactifs d'un isotope donné est décrit par

$$\frac{dN(t)}{dt} = -\lambda N , \quad (1.25)$$

où le taux de désintégration λ est inversement proportionnel à la demi-vie de l'isotope: $\lambda = \ln(2)/\tau_{1/2}$. C'est Ernest Rutherford (1871-1937) qui a le premier établi expérimentalement l'universalité de cette loi de désintégration. Dans le cas d'une chaîne de désintégration simple, soit $1 \rightarrow 2 \rightarrow 3$ où "3" est stable, on écrirait:

$$\frac{dN_1}{dt} = -\lambda_1 N_1 , \quad (1.26)$$

$$\frac{dN_2}{dt} = -\lambda_2 N_2 + \lambda_1 N_1 , \quad (1.27)$$

$$\frac{dN_3}{dt} = \lambda_2 N_2 , \quad (1.28)$$

ou encore, sous notre forme générique:

$$\frac{d}{dt} \underbrace{\begin{pmatrix} N_1 \\ N_2 \\ N_3 \end{pmatrix}}_u = \underbrace{\begin{pmatrix} -\lambda_1 N_1 \\ -\lambda_2 N_2 + \lambda_1 N_1 \\ \lambda_2 N_2 \end{pmatrix}}_{g(u)} \quad (1.29)$$

Notons ici que le membre de droite est linéaire par rapport aux N 's. À strictement parler les N 's sont des entiers, mais on peut reexprimer ces EDOs en termes des fractions f de chaque isotope par rapport au total, i.e., $f_k = N_k / \sum_j N_j$. Pour une telle cascade à trois espèces où la quantité totale d'atomes est conservée (dans le sens que $\sum f_k = 1$), il est possible d'obtenir une solution analytique, ce que je vous laisse comme petit défi personnel. La Figure 1.3 montre trois solutions Runge-Kutta du système ci-dessus, pour la cascade de désintégration- β suivante:



avec les demi-vies telles qu'indiquées ci-dessus (en heures), et une condition initiale de pur ^{76}Kr . On pose donc $(1, 2, 3) \equiv (\text{Kr}, \text{Br}, \text{Se})$ dans l'éq. (1.29). La solution est suivie sur 100 heures, avec ici un pas de temps plutôt grand: $h = 5\text{ hr}$, soit une fraction substantielle ($\sim 1/3$) de la plus petite demi-vie.

Ici l'espèce-mère, le ^{76}Kr , décroît exponentiellement selon la relation $f(t)/f(0) = \exp(-\lambda_{\text{Kr}} t)$, et offre donc la possibilité de mesurer l'erreur numérique (ϵ):

$$\epsilon(t_n) = |f_{\text{Kr}}(t_n) - \exp(-\lambda_{\text{Kr}} t_n)| . \quad (1.30)$$

Cette séquence temporelle de l'erreur numérique est portée en graphique au bas de la Figure 1.3, avec les courbes correspondantes pour des solutions Runge-Kutta utilisant un pas 2 et 4 fois plus petits, tel qu'indiqué. Il est remarquable que même pour $h = 5\text{ hr} = 20$ pas de temps pour couvrir 100 heures,— notre solution Runge-Kutta demeure précise à 10^{-5} ou mieux.

La méthode Runge-Kutta étant d'ordre 4, on s'attend à ce qu'une diminution du pas par un facteur deux conduise à une diminution de l'erreur par un facteur 2^4 ; Le trait pointillé retrace la courbe d'erreur pour $h = 5$ divisée par un facteur numérique $4^4 = 256$; sa quasi-coincidence avec la courbe d'erreur pour $h = 1.25$ confirme ici l'ordre 4 de la méthode de Runge-Kutta.

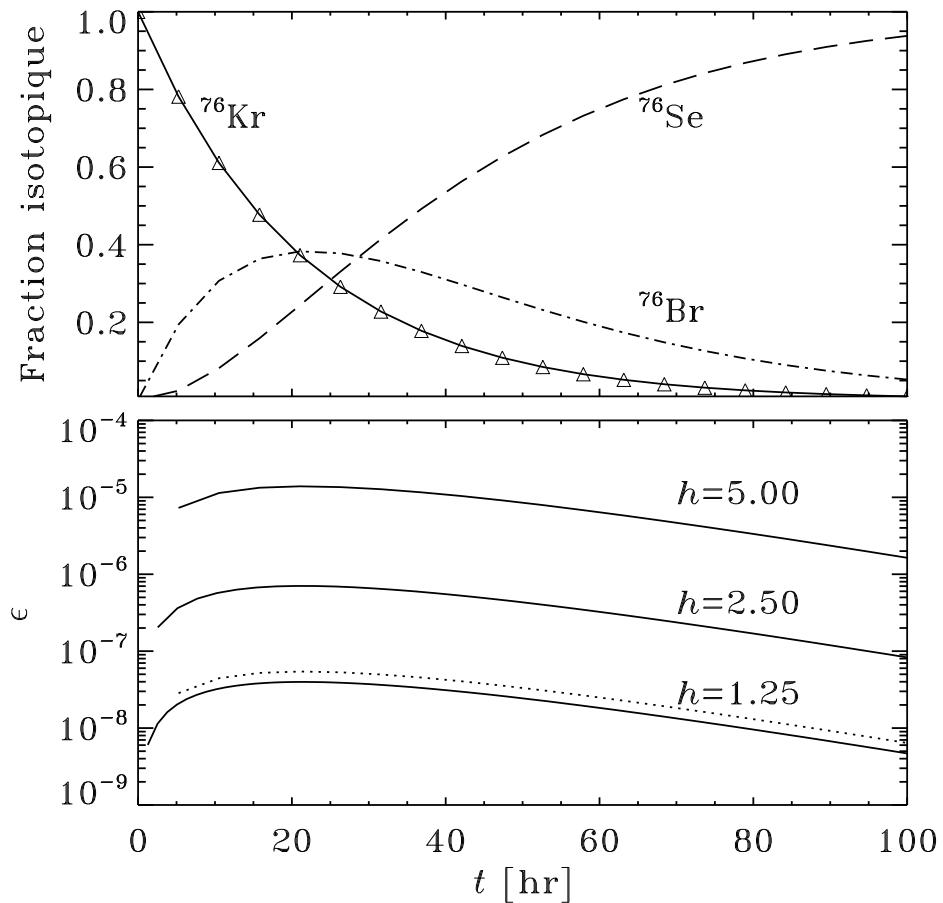


Figure 1.3: La chaîne de désintégration $^{76}\text{Kr} \rightarrow ^{76}\text{Br} \rightarrow ^{76}\text{Se}$. La partie du haut montre l'évolution temporelle des fractions isotopiques des trois isotopes impliqués, obtenue par la méthode Runge-Kutta avec pas $h = 5\text{ hr}$. La partie du bas montre l'évolution temporelle de l'erreur sur la fraction de ^{76}Kr , pour trois tailles de pas. Le trait pointillé est la courbe d'erreur pour $h = 5$ divisée par un facteur $4^4 = 256$.

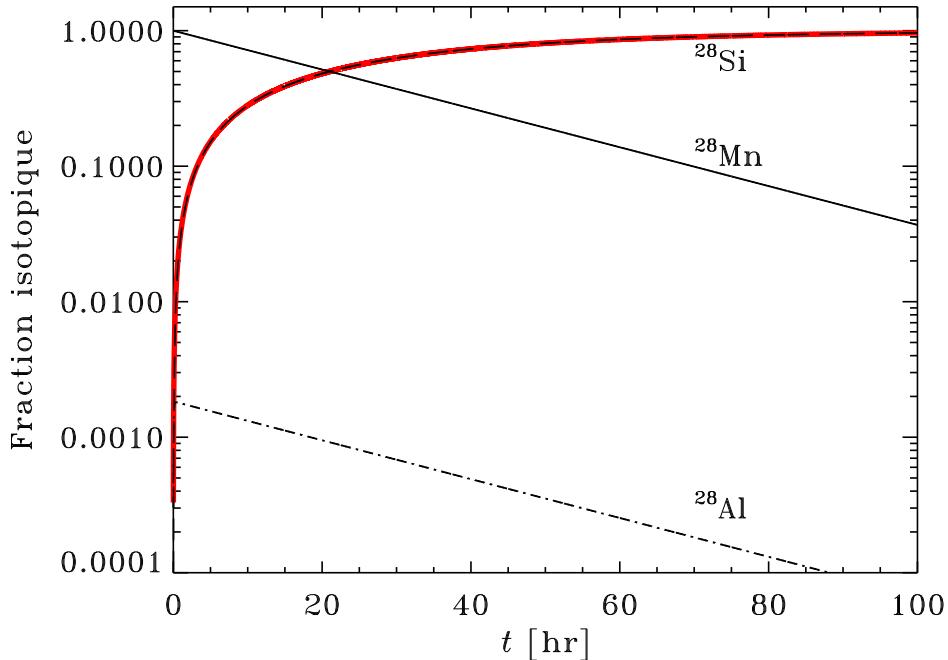


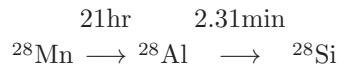
Figure 1.4: La chaîne de désintégration $^{28}\text{Mn} \rightarrow ^{28}\text{Al} \rightarrow ^{28}\text{Si}$. Solution par Runge-Kutta avec $h = 10^{-2}$ hr, un pas de temps minuscule par rapport à la durée d'intégration, imposé par la très courte demi-vie du ^{28}Al . Le trait en rouge est une solution approximative supposant que ^{28}Mn se désintègre directement vers ^{28}Si (voir texte).

On pourrait imaginer une autre mesure plus “physique” de l’erreur, basée sur le fait que le nombre total de noyaux est ici conservé; ceci implique que

$$f_1(t) + f_2(t) + f_3(t) = 1 . \quad (1.31)$$

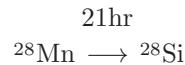
et donc qu’une mesure de l’erreur globale pourrait se définir comme $E(t) = 1 - (f_1 + f_2 + f_3)$; pour les trois solutions de la Figure 1.3, cette quantité demeure inférieure à 10^{-15} en tout temps, mais nous savons déjà (viz. la Fig. 1.3) que les erreurs numériques sont beaucoup plus grandes, variant de $\sim 10^{-5}$ à 10^{-8} dépendant de h . Ici, les erreurs numériques affectant les diverses espèces isotopiques sont fortement couplées, i.e., une erreur sous-estimant ^{76}Kr à un t_n quelconque entraîne une surestimation quasi-identique de ^{76}Br , de manière telle que ces erreurs s’annulent (presque) lorsqu’on calcule la somme des deux espèces.

La chaîne de désintégration que nous venons de considérer est “facile” à traiter numériquement; les équations sont linéaires, et leurs temps caractéristiques respectifs sont semblables. Considérons maintenant la chaîne de désintégration suivante, toujours de type β :



soit $(1, 2, 3) \equiv (\text{Mn}, \text{Al}, \text{Si})$ dans l’éq. (1.29). Cette fois il existe une énorme disparité entre les deux demi-vies. Celà n’empêche pas la méthode Runge-Kutta de fonctionner, mais le pas de temps doit être suffisamment petit pour bien capturer la seconde désintégration. La solution montrée en Figure 1.4 utilise maintenant 10^4 pas de taille $h = 10^{-2}$ hr $\equiv 0.6$ min, soit un quart de la demi-vie du ^{28}Al . Si le but de l’exercice est de calculer à quel taux ^{28}Si est produit, et qu’on ne s’intéresse pas particulièrement à l’évolution temporelle du ^{28}Al , alors on peut se

permettre une simplification qui permettra l'usage d'un pas de temps beaucoup plus grand: remplacer la chaîne à trois espèces ci-dessus simplement par



Ceci revient à supposer que la désintégration du ^{28}Al est instantanée par rapport à celle du ^{28}Mn . Autrement dit, on suppose que ^{28}Al demeure en tout temps en équilibre, d'où

$$N_{\text{Al}} = \frac{\lambda_{\text{Mn}}}{\lambda_{\text{Al}}} N_{\text{Mn}} , \quad (1.32)$$

ce qui réduit le système à trois EDO (1.29) à:

$$\frac{d}{dt} \begin{pmatrix} N_1 \\ N_3 \end{pmatrix} = \begin{pmatrix} -\lambda_1 N_1 \\ \lambda_1 N_1 \end{pmatrix} . \quad (1.33)$$

Le trait rouge sur la Figure 1.4 est une solution (analytique) de cette version simplifiée; elle diffère de la solution véritable par $\simeq 10^{-3}$ ou moins à tous les stades de l'évolution. Une solution numérique Runge-Kutta utilisant 20 pas de temps ($h = 5 \text{ hr}$) montre une erreur numérique de l'ordre de 10^{-5} , et à l'échelle des axes de ce graphiques, la solution d'équilibre pour ^{28}Al donnée par l'éq. (1.32) demeure indistinguables de la solution numérique complète. Quand ce genre de simplification est possible (i.e., on ne s'intéresse vraiment pas à ^{28}Al et un haut niveau de précision sur les autres espèces n'est pas nécessaire), le gain en terme de temps d'exécution est immense.

Il est plus que temps de passer à une chaîne de désintégration plus complexe, soit celle du Radon 211, ce sournois poison de nos sous-sols... Sa chaîne de désintégration est illustrée à la Figure 1.5 dans le "tableau périodique isotopique", soit le plan défini par la passe atomique A (axe horizontal) et la charge électrique Z (axe vertical). Les flèches verticales correspondent aux désintégration- β (émission d'un positron) et les flèches diagonales à la désintégration- α (émission d'un noyau d'Hélium)¹. Ce qui est nouveau ici est que ^{211}Rn a accès à deux modes de désintégration, soit la désintégration- α (probabilité 0.26) et la désintégration- β (probabilité 0.74). Je vous laisse en exercice d'établir la forme générique du système d'EDOs (soit l'équivalent de (1.29) décrivant cette double chaîne de désintégration).

On constate ici que l'une des demi-vies ($^{211}\text{Po} \rightarrow ^{207}\text{Pb}$) est vraiment beaucoup plus courte, et une autre ($^{207}\text{Bi} \rightarrow ^{207}\text{Pb}$) beaucoup plus longue, que toutes les autres. Il serait approprié ici d'appliquer le même truc que précédemment, soit supposer que le ^{211}At se désintègre directement en ^{207}Pb , avec une demi-vie de 7.2hr, ce qui évacue ^{211}Po du problème. Une solution Runge-Kutta ($h = 1 \text{ hr}$) de ce système réduit est présentée au bas de la Figure 1.5. Au bout des 200 heures couvertes par ce calcul, il ne reste effectivement plus que ^{207}Bi qui soit toujours radioactif, mais en couvrir une demi-vie de 30 ans demanderait 262980 pas de temps de $h = 1 \text{ hr}$; il serait beaucoup plus efficace d'augmenter substantiellement le pas de temps après quelques centaines d'heures. La section 1.5 introduit plus loin une technique simple permettant un ajustement automatique du pas de temps, qui fonctionnerait très bien ici.

1.4 Exemple 2: un système nonlinéaire réactif

Passons maintenant à un problème fortement nonlinéaire. Considérons la chaîne réactive suivante, impliquant les réactants A, B, C, X et Y :



¹Les divers isotopes du Radon sont eux-même produit par la chaîne de désintégration de l'Uranium et du Thorium, donc La Fig. 1.5 n'est en fait qu'une petite partie d'un chaîne de désintégration beaucoup plus complexe.

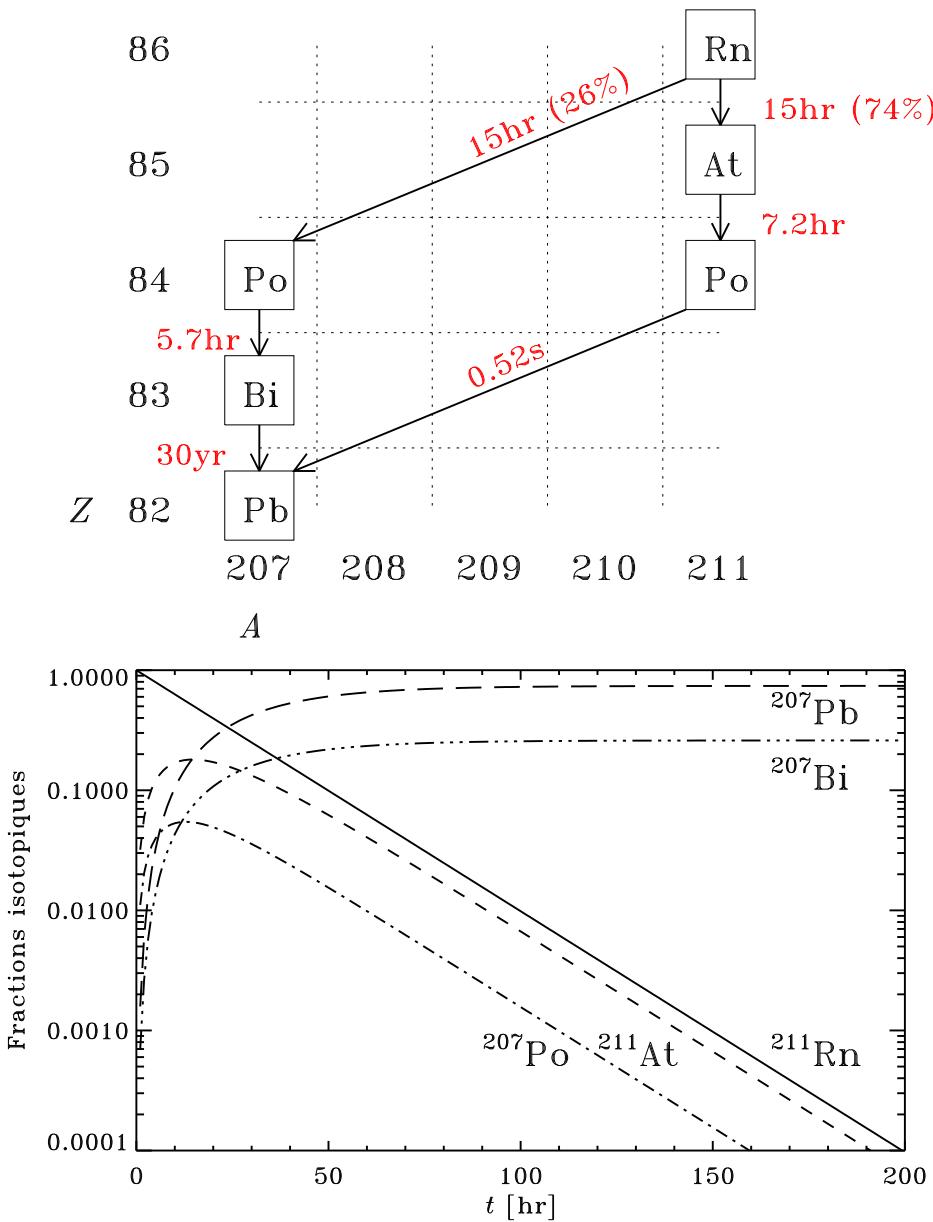


Figure 1.5: La double chaîne de désintégration du ^{211}Rn . Le diagramme du haut indique les deux chaînes de réactions, avec les demi-vies indiquées en rouge. Le diagramme du bas est une solution approximative supposant que ^{211}Po se désintègre directement vers ^{207}Pb (voir texte).



Contrairement aux chaines de désintégration nucléaires considérées précédemment, cette chaîne est *nonlinéaire*, car certains réactants réagissent entre eux (e.g., les termes $B + X$, $2X$, etc.), et *autocatalytique*, car certains réactants se régénèrent au fil de la chaîne (e.g., X dans la troisième réaction ci-dessus). La chimie et la biochimie regorgent de ce genre de réactions.

Encore une fois on peut écrire un système général d'EDOs décrivant l'évolution temporelle des concentrations des réactants:

$$\frac{d}{dt} \underbrace{\begin{pmatrix} A \\ B \\ X \\ Y \\ C \end{pmatrix}}_u = \underbrace{\begin{pmatrix} -\lambda_1 A \\ -\lambda_2 BX \\ \lambda_1 A - \lambda_2 BX + \lambda_3 X^2 Y - \lambda_4 X \\ \lambda_2 BX - \lambda_3 (X^2 Y) \\ \lambda_4 X \end{pmatrix}}_{g(u)} \quad (1.38)$$

Comprenez bien comment on est passé de (1.34) à (1.38); la subtilité vient du fait que la réaction 3 produit un gain net de un X , à un taux donné par le produit X^2Y . Comparez aussi à la forme équivalente (1.29) pour la chaîne de désintégration du ^{76}Kr .

Imaginons maintenant que la chaîne de réaction ci-dessus représente un processus chimique où les réactants X et Y baignent dans un réservoir illimité de A et B ; les concentrations de ces derniers peuvent donc être considérées constantes. Si on considère de plus que les taux de réaction sont $\lambda_k = 1$ pour les quatre réactions, le système ci-dessus se réduit effectivement à

$$\frac{dX}{dt} = A - (B + 1)X + X^2Y, \quad (1.39)$$

$$\frac{dY}{dt} = BX - X^2Y. \quad (1.40)$$

puisque C est calculable a posteriori une fois l'évolution de X connue. Je vous laisse déjà vérifier que la solution d'équilibre ($d/dt = 0$) est donnée ici par:

$$X_{\text{eq}} = A, \quad Y_{\text{eq}} = B/A. \quad (1.41)$$

La Figure 1.6 illustre deux séries de solutions de ce système obtenues par Runge-Kutta, pour $A = B = 1$ (en haut) et $A = 1, B = 3$ en bas. Les conditions initiales sur X et Y ont été choisies arbitrairement dans les deux cas. À $A = B = 1$, et quelle que soit la grandeur du pas, la solution converge vers la solution d'équilibre à partir de la condition initiale $(X_0, Y_0) = (1.5, 1.5)$. La solution d'équilibre est ici l'attracteur de l'espace de phase. Il est remarquable qu'on puisse converger ici avec un pas $h = 1$, qui est égal à l'inverse du taux de réaction puisqu'on a posé $\lambda = 1$ ici. La méthode de Runge-Kutta semble incroyablement robuste... Ce n'est cependant plus le cas pour des solutions ayant $A = 1$ et $B = 3$. Ces solutions convergent non pas vers la solution d'équilibre, mais plutôt vers une solution périodique correspondant à l'orbite tracée en gris sur la Figure. Cette fois, la taille du pas a un effet très marqué, particulièrement dans le coin inférieur droit de la trajectoire, où les solutions ayant $h \gtrsim 0.1$ éprouvent beaucoup de difficulté à "tourner le coin" même si elles parviennent à demeurer sur l'attracteur. De plus, dans ce cas spécifique, une solution Runge-Kutta utilisant un pas $h \gtrsim 0.31$ diverge dès le premier pas de temps.

Le message ici est que dans le cas d'un problème nonlinéaire, le choix d'un pas de temps approprié peut devenir problématique et est souvent difficile à établir a priori. On peut évidemment y aller par essai-erreur, ou simplement se farcir un pas de temps minuscule, mais il est possible de faire beaucoup mieux en ajustant le pas de temps durant l'évolution de la solution. Voyons comment.

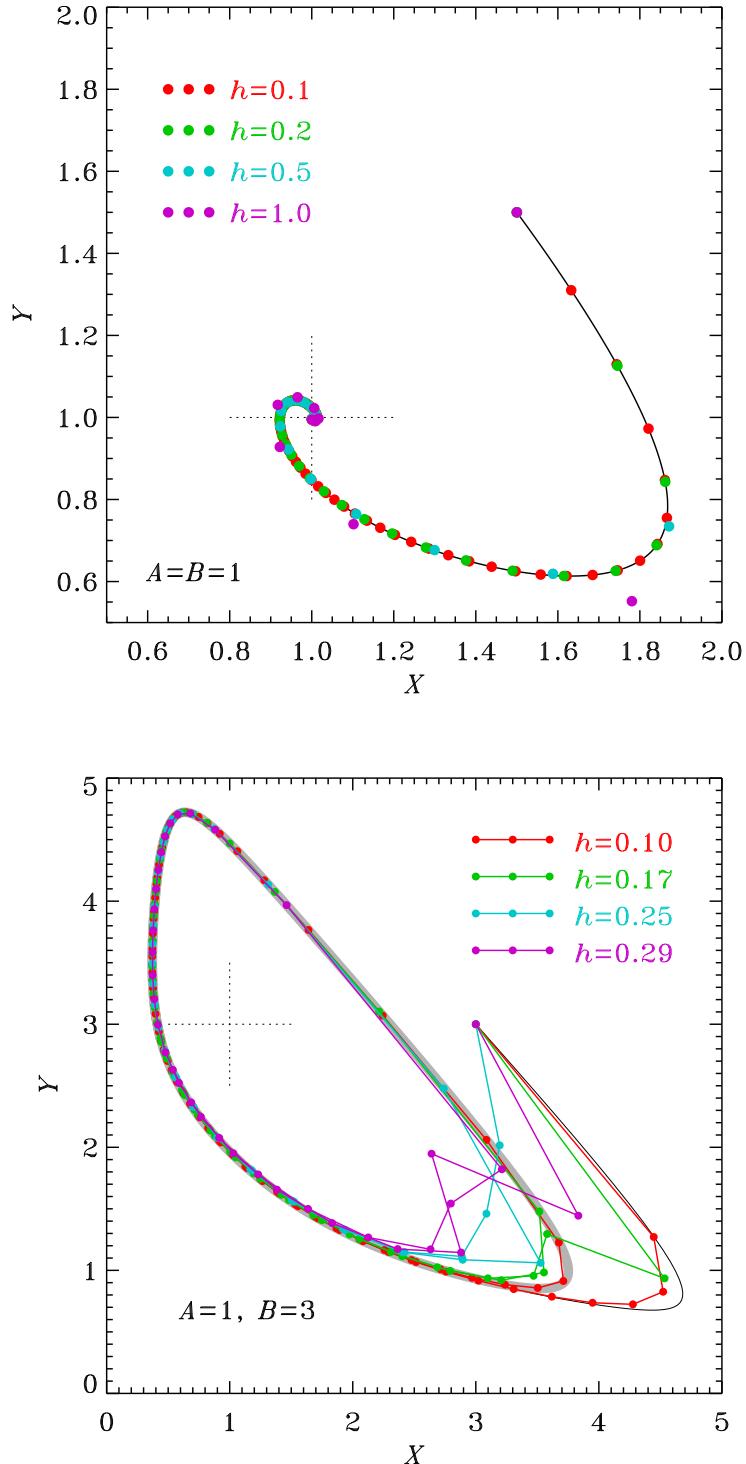


Figure 1.6: Solutions Runge-Kutta utilisant diverses grandeurs de pas h , comme indiqué par le code couleur, pour le problème d'oscillation nonlinéaire décrit par les éqs. (1.34). Les solutions sont présentées ici sous la forme de trajectoires dans l'espace de phase $[X, Y]$ du système. Le graphique du haut montre des solutions pour $A = B = 1$, et celui du bas pour $A = 1$ et $B = 3$. La croix pointillée indique la solution d'équilibre (1.41).

1.5 Runge-Kutta avec pas adaptif

Dénotons par u_1 une solution Runge-Kutta calculée en utilisant un pas h , et par u_2 une solution calculée en faisant deux pas de longueur $h/2$. La seconde est présumément plus précise que la première, et leur différence Δ est une mesure de l'erreur de troncation:

$$\Delta = |u_2 - u_1| \quad (1.42)$$

Supposons maintenant que l'on soit en mesure de spécifier, a priori, un niveau maximum d'erreur qui soit tolérable; par exemple, dans le cas des chaines de réactions autocatalytiques considérées précédemment, on pourrait vouloir insister que l'erreur demeure inférieure à $\epsilon = 10^{-5}$. La stratégie d'ajustement du pas de temps est la suivante:

1. si $\Delta < \epsilon/2$: on accepte la solution u_2 et on multiplie le pas par un facteur 1.5;
2. si $\epsilon/2 < \Delta < \epsilon$: on accepte u_2 et on continue avec le même pas h ;
3. si $\Delta > \epsilon$: on refuse u_2 , on divise le pas par 1.5, et on reprend le calcul de u_1 et u_2 .

Le choix du facteur 1.5 est arbitraire jusqu'à un certain point; bien des bouquins suggèrent 2.0, mais mon expérience personnelle est que ceci conduit parfois à une alternance de croissance/décroissance du pas, tandis que 1.5 donne une variation plus graduelle et régulière du pas h .

Cette stratégie est facile à coder si on a affaire à une seule EDO; mais pour un système couplé de plusieurs EDOs, Δ et ϵ deviennent des vecteurs, et l'ajustement doit se faire sur la base de la composante ayant le plus grand Δ par rapport à l' ϵ correspondant. La Figure 1.7 donne un listing en Python d'un code solutionnant le problème nonlinéaire de la §1.4 de cette façon,

La Figure 1.8 montre le résultat de l'application de cette technique au problème d'oscillation nonlinéaire de la §1.4. La Figure 1.9 (graphique du haut) porte en graphique la même solution, cette fois sous la forme de séquence temporelles des variables X (en rouge) et Y (vert). La taille variable des pas h est indiquée par les tirets verticaux au bas du graphique. Ici le pas h varie entre $h = 0.022$ et $h = 0.563$ au cours de la période simulée, et produit en 82 pas de temps une solution dont la précision globale est environ la même qu'une solution de 954 pas fixe de $h = 0.011$. Il faut cependant ne pas perdre de vue que la solution à pas adaptif effectue plus d'évaluations du membre de droite de (1.39), donc en terme de temps d'exécution le gain net inférieur à $954/82$ par un facteur pouvant aller de 1.375 à 1.66, dépendant du détail de l'implémentation algorithmique. Le gain n'en demeure pas moins substantiel.

L'ajustement adaptif du pas permet d'obtenir des solutions utiles même en utilisant des tolérances ϵ surprenamment grandes; le graphique au bas de la Figure 1.9 en montre exemple à $\epsilon = 10^{-2}$, toujours pour le même problème. L'ajustement adaptif permet de réduire le pas là où la solution tend à être la plus instable à l'amplification des erreurs de troncation, soit ici vers $t = 0.5$ et 7.5 . La solution est imprécise, mais demeure stable. Cependant, une comparaison attentive avec le graphique du haut ($\epsilon = 10^{-5}$) révèle que les caractéristiques globales du cycle nonlinéaire sont bien capturées, incluant en particulier la période de l'oscillation et l'amplitude des pics; pas mal du tout pour une solution de 27 pas de temps!

1.6 Transmission des signaux dans les neurones

Notre dernier exemple de solutions d'EDOs combine forte nonlinéarité et disparité des temps caractéristiques pour un système à quatre variables dans un contexte biophysique: l'électrodynamique des membranes des neurones. Le décodage sémantique et la compréhension de la phrase que vous êtes en train de lire aura probablement impliqué quelques $\sim 10^{11}$ signaux électriques entre vos neurones dans les $\sim 8\text{--}10$ secondes requise pour en arriver ici.

```

1 import numpy as np
2 # FONCTION CALCULANT LE COTE DROIT DU SYSTEME DE DEUX EDO
3 def g(t0,u):
4     A,B=1.,3.                                # parametres dans (1.39)--(1.40)
5     gX=A-(B+1.0)*u[0]+u[0]**2*u[1]          # RHS Eq (1.39)
6     gY=B*u[0]-u[0]**2*u[1]                  # RHS Eq (1.40)
7     eval=np.array([gX,gY])                  # vecteur RHS (pentes)
8     return eval
9 # END FONCTION G
10 # FONCTION CALCULANT UN SEUL PAS DE RUNGE-KUTTA D'ORDRE 4
11 def rk(h,t0,uu):
12     g1=g(t0,uu)                            # Eq (1.15)
13     g2=g(t0+h/2.,uu+h*g1/2.)            # Eq (1.16)
14     g3=g(t0+h/2.,uu+h*g2/2.)            # Eq (1.17)
15     g4=g(t0+h,uu+h*g3)                 # Eq (1.18)
16     unew=uu+h/6.* (g1+2.*g2+2.*g3+g4) # Eq (1.19)
17     return unew
18 # END FONCTION RK
19 # OSCILLATIONS NONLINEAIRES: 2 EDOS NONLINEAIRES COUPLEES
20 nMax=1000                                # nombre maximal de pas de temps
21 eps =1.e-5                                 # tolerance
22 tfin=10.                                  # duree d'integration
23 t=np.zeros(nMax)                          # tableau temps
24 u=np.zeros([nMax,2])                      # tableau solution
25 u[0,:]=np.array([3.,3.])                  # condition initiale
26 nn=0                                      # compteur iterations temporelles
27 h=0.1                                     # pas initial
28 while (t[nn] < tfin) and (nn < nMax):    # boucle temporelle
29     u1 =rk(h, t[nn],u[nn,:])              # pas pleine longueur
30     u2a=rk(h/2.,t[nn],u[nn,:])          # premier demi-pas
31     u2 =rk(h/2.,t[nn],u2a[:])           # second demi-pas
32     delta=max(abs(u2[0]-u1[0]),abs(u2[1]-u1[1])) # Eq (1.42)
33     if delta > eps:                   # on rejette
34         h/=1.5                         # reduction du pas
35     else:                           # on accepte le pas
36         nn=nn+1                        # compteur des pas de temps
37         t[nn]=t[nn-1]+h                # le nouveau pas de temps
38         u[nn,:]=u2[:]                 # la solution a ce pas
39         if delta <= eps/2.: h*=1.5   # on augmente le pas
40     print("{0}, t {1}, X {2}, Y {3}.".format(nn,t[nn],u[nn,0],u[nn,1]))
41 # fin boucle temporelle
42 # END

```

Figure 1.7: Code Python pour la solution du problème d'oscillations réactives nonlinéaires par Runge-Kutta d'ordre 4 avec pas adaptif. Notons qu'ici le temps (variable $t[nn]$) est fourni en argument à la fonction `RK` qui le passe à la fonction `G...` qui LANCE ET COMPTE !!!! ... s'cusez on continue..., même si cette dernière ne l'utilise pas, histoire de demeurer général (viz. l'éq. (1.1)). A noter également, Python permet aux fonctions de retourner un tableau (ici de dimension 1 et taille 2), ce qui n'est pas le cas de tous les langages de programmation.

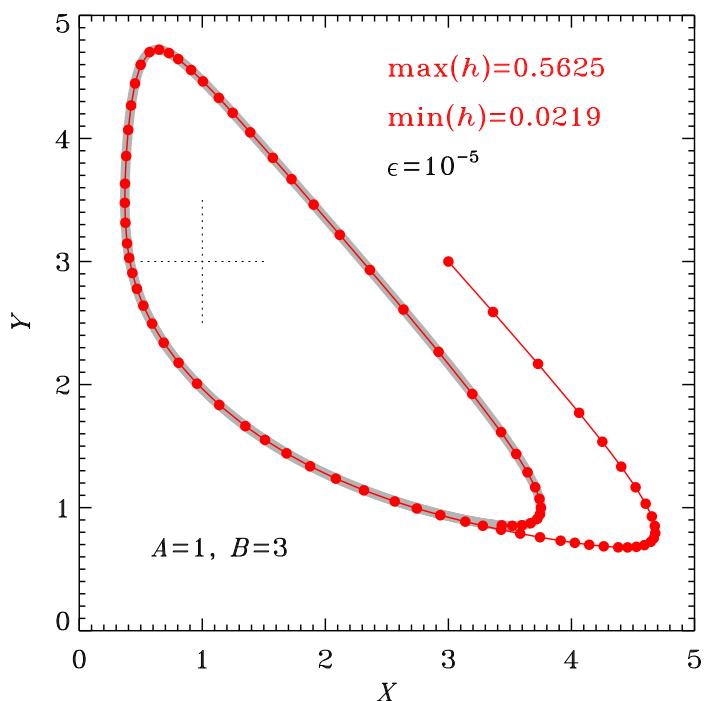


Figure 1.8: Solution Runge-Kutta avec pas adaptif ($\epsilon = 10^{-5}$), pour le problème d'oscillation nonlinéaire de la §1.4. Comparer ceci au graphique du bas de la Fig. 1.6. La taille du pas de temps h varie par plus d'un facteur 20 ici.

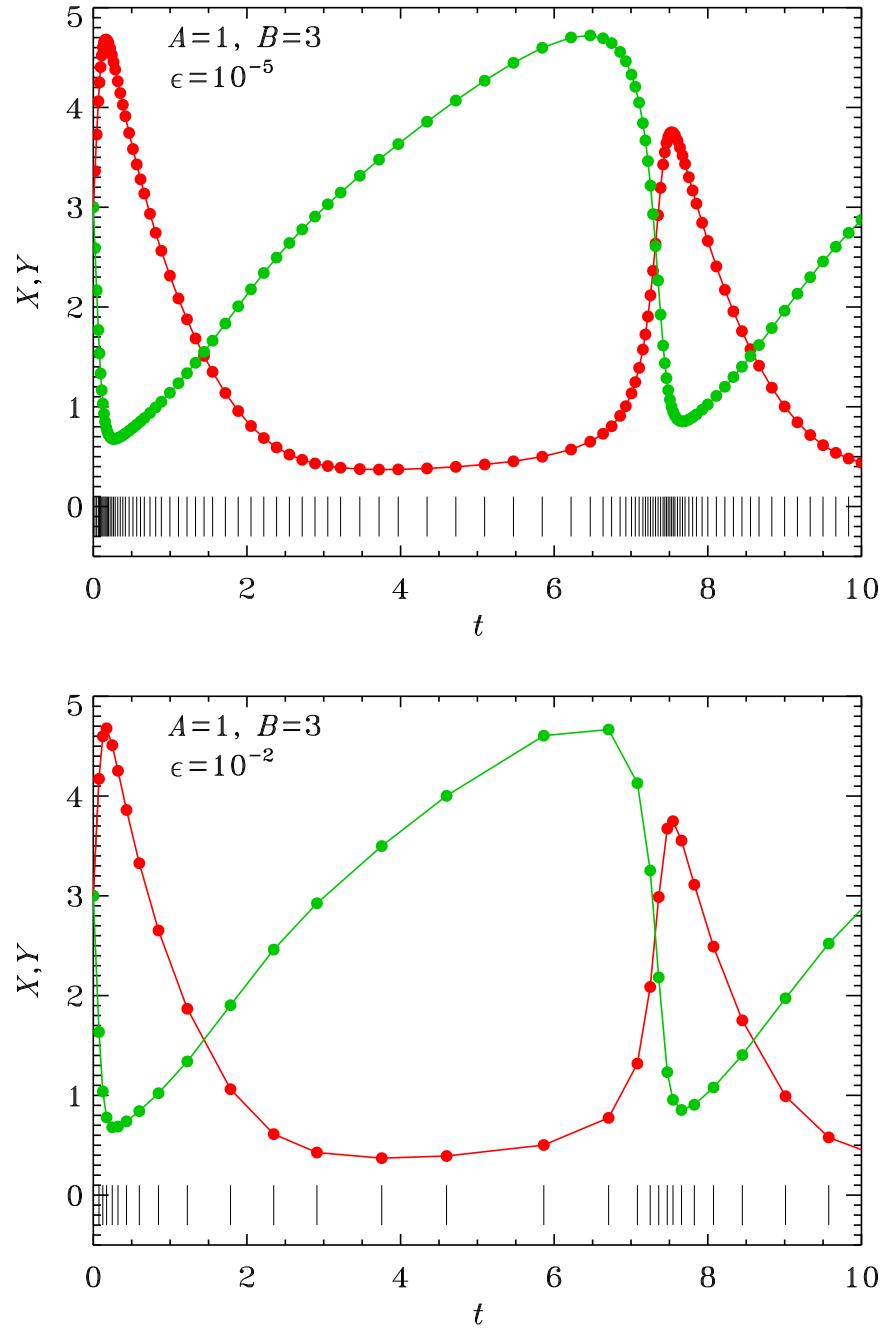


Figure 1.9: Solution Runge-Kutta avec pas adaptif pour le problème d'oscillation nonlinéaire de la §1.4, portée en graphique ici en terme de séquences temporelles pour les variables X (rouge) et Y (vert). Les tirets verticaux au bas des graphiques indiquent la taille des pas temporels. La solution du haut est la même que sur la Fig. 1.8 ($\epsilon = 10^{-5}$), tandis que celle du bas utilise une tolérance $\epsilon = 10^{-2}$.

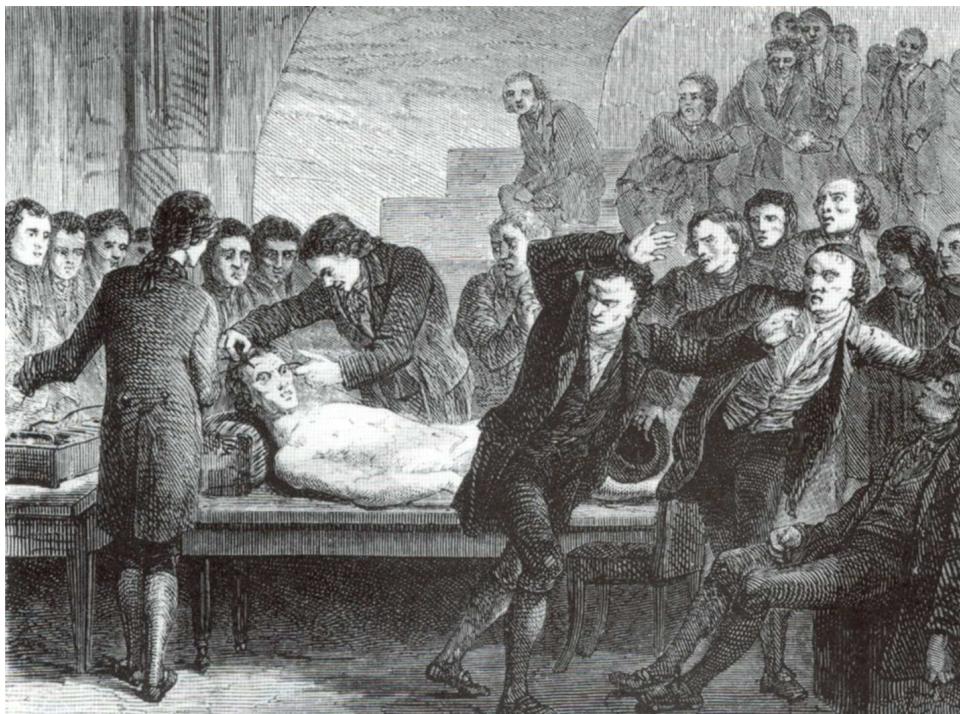


Figure 1.10: Spectacle électrisant populaire au début du 19^e siècle: des courants électriques sont utilisés pour faire grimacer ou gesticuler un cadavre relativement frais. Tiré de l'ouvrage *Landmark in Western Science*, par P. Whitfield.

1.6.1 Les neurones

On considère généralement que l'application des techniques de la physique expérimentale à l'étude du système nerveux remonte aux expériences de Luigi Galvani (1737-1798), qui en 1791 démontre qu'une stimulation électrique pouvait déclencher la contraction musculaire dans des pattes de grenouilles fraîchement amputées de leur malchanceuses propriétaires. Étudié en plus de détail par Alessandro Volta (1745-1827) dans les années qui suivirent, ce phénomène de l'électricité animale, connu alors sous le nom de "galvanisme" devint rapidement populaire et conduit à l'idée que la distinction entre le vivant et le non-vivant ne pourrait bien être qu'une question d'électricité. Dans les premières décennies du dix-neuvième siècle, il était courant de voir organisés des spectacles publics (payants) où un "expérimentateur" utilisait des électrodes pour faire cligner des yeux ou bouger les mains à un cadavre relativement frais (voir Figure 1.10).

Il a depuis été démontré que la propagation des signaux électriques chez le vivant est effectuée par un groupe de cellules particulières, les *neurones*, dont la grande majorité, chez les vertébrés du moins, se retrouvent dans le cerveau. Voici une petite liste de dix choses à savoir sur les neurones, question de culture générale:

1. Nombre de neurones dans le cerveau humain: 10^{10} — 10^{11} .
2. Interconnexions: 100—1000/neurone, soit 10^{14} — 10^{15}
3. Croissance: nulle! les neurones ne se divisent pas après la naissance, et ne peuvent que mourir par la suite.
4. Croissance de la connectivité: 10^6 /sec les premiers dix années de vie (gain net); taux maximal 10^7 /sec de un à trois ans.

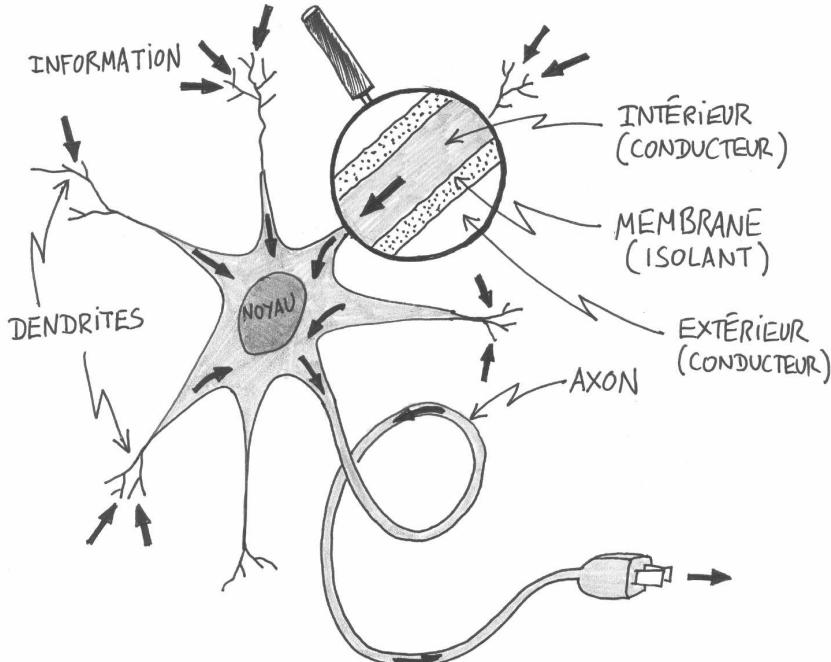


Figure 1.11: Représentation schématique d'un neurone générique. Les dendrites captent l'information provenant d'autres neurones, et l'intègrent en un signal de sortie envoyé dans l'axone. Les flèches noires indiquent la direction de transmission de l'information.

5. Consommation énergétique: $\simeq 25\%$ du corps
6. Masse du système nerveux: 1—2% du corps
7. Dimension (axone): diamètre $\simeq 1\text{--}2\mu\text{m}$, longueur $\simeq 0.01\text{ mm--}1\text{ m}$.
8. Epaisseur des membranes: $\sim 40\text{--}70\text{\AA}$ ($1\text{\AA} = 10^{-10}\text{ m}$)
9. Différence de potentiel intérieur-extérieur: -50mV à -90mV
10. Résistance: $\sim 1000\text{ Ohm cm}^2$; capacité: $\sim 1\text{ }\mu\text{F/cm}^2$

La Figure 1.11 offre une représentation schématique d'un neurone générique. Le corps de la cellule, contenant le noyau, est entouré d'extensions souvent très ramifiées appelées *dendrites*. Une (et une seule) de ces extensions est souvent plus robuste et/ou plus longue que toutes les autres dendrites; c'est *l'axone*. Des expériences ont démontré que les dendrites captent l'information provenant d'autres neurones, et la “fusionnent” en un signal qui est envoyé le long de l'axone, au bout duquel une ou plusieurs connexions sont établies avec une ou plusieurs dendrites appartenant à un ou plusieurs autres neurones. Le traitement de l'information neuronale se fait donc dans la direction indiquée par les flèches noires sur la Figure 1.11.

En première approximation, plusieurs neurones effectuent une forme *d'intégration* du signal qu'elles reçoivent de leur dendrites, et produisent un signal de sortie (I_A) dans l'axone qui est une fonction nonlinéaire du signal d'entrée intégré. Mathématiquement, on écrirait:

$$I_A = f(S), \quad S = \sum_{k=1}^N w_k I_k, \quad (1.43)$$

où I_k est le signal provenant de la $k^{\text{ème}}$ connexion dendritique (ou *synapse*), et w_k est un coefficient numérique mesurant la sensibilité de cette connexion. Notons que certaines synapses

sont caractérisées par un $w_k < 0$, i.e., la présence d'un signal à une telle synapse a un effet *inhibiteur* sur la production du signal de sortie dans l'axone. La nature de la fonction $f(S)$ est telle qu'elle tend vers une valeur constante quand S est petit, à une autre valeur constante quand S est grand, et varie de manière graduelle et continue entre ces deux valeurs (pensez par exemple à la fonction $\arctan(S)$).

L'équation (1.43) est une représentation extrêmement grossière de l'opération d'un neurone, mais demeure à la base de la conception des réseaux de neurones sur ordinateurs, couramment utilisés en intelligence artificielle et en classification automatisée. On y reviendra d'ailleurs au chapitre 6.

Du point de vue électrique, la membrane cellulaire d'un neurone s'avère être un isolant respectable, tandis que l'intérieur et l'extérieur d'un neurone, riche en électrolytes divers, sont de bons conducteurs. Il est donc tentant de considérer les dendrites et axones comme étant des fils électriques, et le traitement et transfert de l'information par le neurone comme étant du à la combinaison et propagation de courants électriques. Développons maintenant cette hypothèse, et examinons-en les conséquences.

1.6.2 L'équation du câble

L'idée générale que nous explorerons ici est d'appliquer une différence de potentiel ΔV d'un bout à l'autre d'une dendrite (correspondant à un signal provenant d'un autre neurone), et de produire un courant électrique s'écoulant le long de la dendrite afin de "faire suivre" le signal reçu jusqu'à la base de la dendrite et, ultimement, jusqu'au bout de l'axone. Trois problèmes se présentent:

1. l'intérieur du neurone, bien que relativement bon conducteur, a néanmoins une résistance non-négligeable;
2. la membrane est un bon isolant, dans le sens qu'elle offre une résistance très grande, mais néanmoins finie, ce qui implique qu'un faible courant peut tout de même traverser de l'intérieur à l'extérieur du neurone;
3. structurellement, la membrane d'une dendrite représente une couche cylindrique isolante séparant un cylindre conducteur (l'intérieur de la dendrite) d'un milieu extérieur également conducteur, ce qui implique que la membrane possède une capacité.

Un circuit électrique combinant tous ces éléments, et donc opérationnellement analogue à une section de dendrite, est illustré sur la Figure 1.12. Dans une telle situation il est naturel de mesurer les conductance et capacité *par unité de longueur* le long de la dendrite.

Il s'agit maintenant d'obtenir une équation décrivant la variation du potentiel V en fonction de la distance x le long de la dendrite, et du temps t . Considérons tout d'abord le courant produit par une différence de potentiel V appliquée à travers la membrane en un point x donné. Se référant à la Figure 1.12, on voit que le courant membranaire total (i_m) sera la somme des courants traversant la résistance (i_r) et la capacité (i_c), soit:

$$i_m = i_c + i_r = \frac{dQ}{dt} + \frac{V}{r_m}. \quad (1.44)$$

Notez bien qu'il s'agit ici d'un courant *par unité de longueur* en x . Puisque $Q = c_m V$, on peut réécrire ceci comme:

$$i_m = c_m \frac{dV}{dt} + \frac{V}{r_m}. \quad (1.45)$$

Considérons maintenant le courant s'écoulant *le long* de la dendrite lorsque sujette à une différence de potentiel ΔV établie sur une distance Δx à l'intérieur de la dendrite. Le courant s'écoulera dans une direction à l'intérieur, et dans l'autre à l'extérieur de façon à boucler le

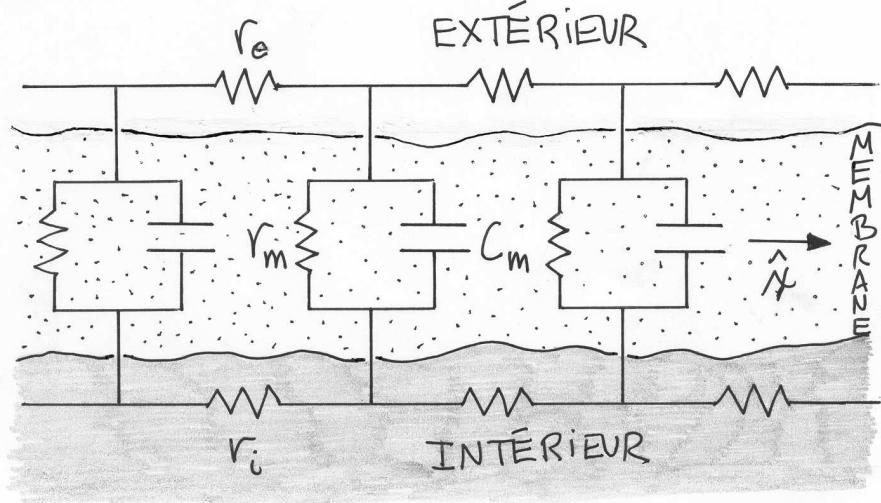


Figure 1.12: Modèle électrique d'une membrane de neurone. La membrane possède une capacité (c_m) ainsi qu'une résistance (r_m), qu'on mesure toutes deux par unités de longueur dans la direction de l'axe de la dendrite ou de l'axone (\hat{x} ici). Un courant peut également circuler à l'extérieur du neurone (résistance par unité de longueur r_e Ohm m $^{-1}$), ainsi qu'à l'intérieur (résistance r_i Ohm m $^{-1}$). Le défi est d'appliquer une différence de potentiel le long de la dendrite afin de produire un courant le long de celle-ci, malgré les pertes à travers la membrane.

circuit. Ce circuit contient donc deux résistances en séries, et l'application de la Loi d'Ohm conduit directement à:

$$i = \frac{\Delta V}{R} = \frac{\Delta V}{(r_i + r_e)\Delta x} = \frac{1}{(r_i + r_e)} \frac{dV}{dx} . \quad (1.46)$$

En général, le courant variera donc dans la direction x . Dans une situation stationnaire ($\partial/\partial t \equiv 0$), la conservation de la charge requiert qu'un courant traverse la membrane, pour compenser la variation du courant en x ; la grandeur de ce courant est donnée par

$$\frac{di}{dx} = i_m , \quad (1.47)$$

donc,

$$i_m = \frac{1}{(r_i + r_e)} \frac{d^2V}{dx^2} . \quad (1.48)$$

Mais il s'agit ici du même courant membranaire que nous avions considéré précédemment; nous pouvons donc supposer que les membres de droite des équations (1.45) et (1.48) sont égaux, ce qui produit l'équation aux dérivées partielles suivante:

$$\tau \frac{\partial V(x, t)}{\partial t} = \lambda^2 \frac{\partial^2 V(x, t)}{\partial x^2} - V(x, t) , \quad (1.49)$$

où l'on a défini les quantités

$$\tau = r_m c_m , \quad (1.50)$$

$$\lambda^2 = \frac{r_m}{r_i + r_e} . \quad (1.51)$$

L'équation (1.49) est une variante de l'équation dite "du câble", dérivée pour la première fois par W. Thompson (1824–1907; officiellement Lord Kelvin à partir de 1892) à l'époque de l'installation du premier câble télégraphique transatlantique. Il s'agit de toute évidence d'une équation différentielle linéaire aux dérivées partielles (EDP).

1.6.3 Solutions à l'équation du câble

Il est habituellement pas mal plus difficile de solutionner analytiquement une EDP, même linéaire, qu'une équation différentielle ordinaire. Cependant, pour des EDP ayant une structure "diffusion" (c.-à-d. une dérivée temporelle d'ordre un combinée à une dérivée spatiale d'ordre deux), la technique de la Transformée de Laplace permet souvent d'obtenir une solution. Dans le cas de l'équation (1.49) il est préférable d'introduire tout d'abord une nouvelle fonction $U(x, t)$ telle que

$$V(x, t) = \exp(-t/\tau)U(x, t). \quad (1.52)$$

Substituant cette expression dans l'éq. (1.49) conduit à l'EDP suivante:

$$\tau \frac{\partial U(x, t)}{\partial t} = \lambda^2 \frac{\partial^2 U(x, t)}{\partial x^2}, \quad (1.53)$$

qui a maintenant la forme d'une équation de diffusion classique pour $U(x, t)$. Considérons maintenant l'application d'un forage harmonique en U imposé à $x = 0$ dans un neurone initialement à potentiel zéro:

$$U(x, t) = 0, \quad U(0, t) = \sin(\omega t). \quad (1.54)$$

On peut démontrer, par la technique de la transformée de Laplace, que l'équation (1.53) sujette aux conditions (1.54) admet une solution de la forme:

$$U(x, t) = \exp(-kx) \sin(\omega t - kx), \quad (1.55)$$

où l'on a défini une nouvelle constante:

$$k = \frac{1}{\lambda} \sqrt{\frac{1}{2}\omega\tau}. \quad (1.56)$$

La solution à notre équation du câble originale (1.49) s'obtient en remultipliant le préfacteur temporel de chaque côté de l'éq. (1.55). Qu'impliquent ces résultats? Vous aurez reconnu (on espère...) le facteur $\sin(\omega t - kx)$ comme étant du type à décrire la propagation d'une onde plane de fréquence ω et nombre d'onde k . Cependant, le préfacteur $\exp(-kx)$ indique que cette onde est atténuée exponentiellement sur une longueur caractéristique k^{-1} , qui de surcroit dépend de la fréquence ω (cf. éq. 1.56). La Figure 1.13 illustre quelques solutions pour des fréquences de plus en plus grandes. L'atténuation exponentielle du signal, de plus en plus marquée à mesure que la fréquence augmente, y est clairement apparente.

Quelles seraient des valeurs typiques pour ce facteur d'atténuation, dans le cas des neurones? Le tableau ci-dessous liste les facteurs d'atténuation correspondant à un axone de longueur 1 m (vous en avez des comme ça dans la moelle épinière...) pour des fréquences d'oscillations imposées de plus en plus grandes.

Le résultat est en fait aberrant. Pour produire un signal d'un mV (valeur typique pour les neurones) au bout d'un axone d'un mètre de long, la différence de potentiel à l'entrée doit être de 250 kV! Même ceux d'entre nous ayant les personnalités les plus électrisantes auraient de la difficulté à produire —et surtout supporter— ce genre de différence de potentiel.

Que doit-on en conclure? Tout simplement que les neurones ne transmettent **pas** l'information à l'aide de courants électriques simples (AC ou DC) se propageant à l'intérieur des dendrites et axones. Des expériences effectuées dans les années 1950 ont démontré, à la surprise générale à l'époque, que c'est la membrane elle-même qui fait tout le travail. Plutôt que d'agir comme un

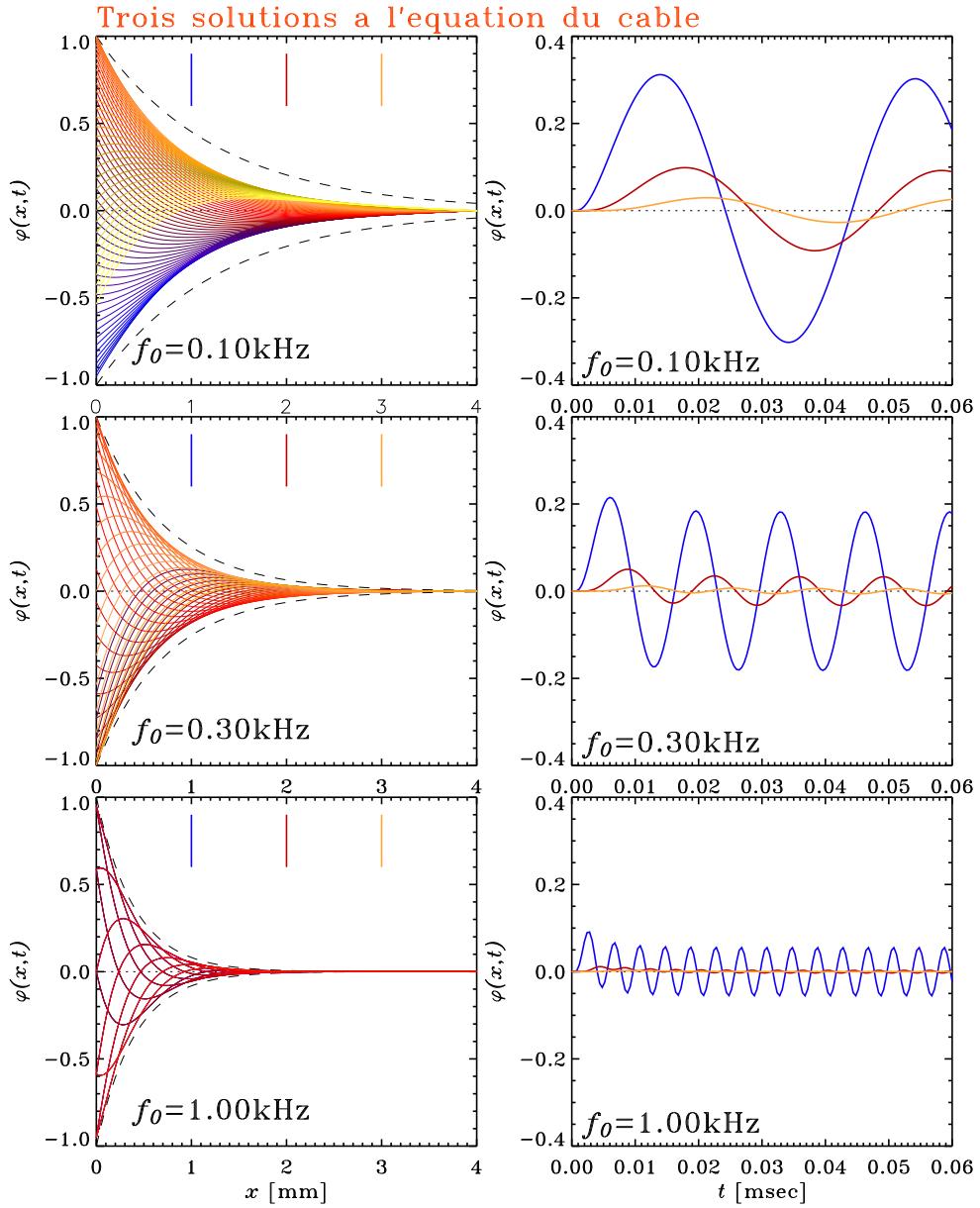


Figure 1.13: Trois solutions à l'équation du câble (1.49) pour un forçage harmonique à $x = 0$ à trois fréquences différentes, allant en augmentant du haut vers le bas. Toutes les solutions ont $\tau = 1\text{ msec}$ et $\lambda = 1\text{ mm}$. La colonne de gauche présente une série de profils $V(x)$ à différents t , et la colonne de droite trois séquences temporelles $V(t)$ extraites aux trois positions indiquées par des tirets verticaux sur la colonne de gauche. Le trait pointillé noir correspond au facteur d'atténuation spatiale $\exp(-kx)$.

Tableau 1.1: Propriétés des courants membranaires passifs

f (Hz)	$\omega = 2\pi f$	τ [s]	λ [mm]	k	$\exp(-kL)$
3	18.8	10^{-3}	5	19.4	$\sim 4 \times 10^{-9}$
30	188.	10^{-3}	5	61.3	$\sim 10^{-27}$
300	1884	10^{-3}	5	194	$\sim 5 \times 10^{-85}$

simple isolant passif, la membrane contrôle activement le passage d'ions Na^+ et K^+ , et donc la différence de potentiel entre l'intérieur de la cellule et l'extérieur. L'information peut donc être propagée le long des dendrites et axones sous la forme d'une "onde de dépolarisation", voyageant à des vitesses de 5–20 mètres par seconde, et dépassant parfois 120 m s^{-1} pour certains type d'axones.

1.7 Projet: le modèle de Hodgkin-Huxley

Le projet qui clot ce chapitre est basé sur un modèle de la dynamique membranaire développé et publié en 1952 par Alan Hodgkin (1914–1998) et Andrew Huxley (1917–2012), et qui leur a valu le Prix Nobel de Physiologie/Médecine en 1963, rien de moins.

1.7.1 Les canaux ioniques et le courant transmembranaire

Notre modèle électrique de la Fig. 1.12 est en fait correct, à un ingrédient crucial près: les canaux ioniques. Ce sont de petites machines moléculaires, incorporées à même la membrane, et qui contrôlent le passage des ions à travers celle-ci. Cette régulation peut se faire dans le sens de la différence de potentiel transmembranaire, dans lequel cas la conductance effective de la membrane est augmentée; ou dans le sens contraire, dans lequel cas elles agissent comme une force électromotrice qui augmente la différence de potentiel, comme le ferait une batterie; on parle alors de pompes ioniques. Les canaux ioniques peuvent donc produire des courants traversant la membrane vers l'extérieur ou l'intérieur, le changement de direction se produisant à une valeur de V appelée *potentiel de repos*, qui varie d'un type de canal ionique à un autre.

Les deux types de canaux considérées dans l'étude de Hodgkin et Huxley étant ceux associés au Na^+ et K^+ , l'équation (1.45) devient:

$$i_m(t) = c_m \frac{dV}{dt} + i_K + i_{\text{Na}} + \frac{V}{r_m}, \quad (1.57)$$

où cette fois les courants sont mesurés par unité de surface. La difficulté du problème vient du fait que les propriétés (conductance) des canaux ioniques dépendent de manière complexe de la différence de potentiel transmembranaire, et donc leurs contributions au courant transmembranaire se retrouvent à être des fonctions très fortement nonlinéaires en V .

1.7.2 Formulation mathématique du modèle de Hodgkin-Huxley

Travaillant sur l'axone du calmar (axone dont la grande taille facilite grandement la manipulation expérimentale), Hodgkin et Huxley ont effectué une grande série de mesures en variant systématiquement les concentrations de Sodium et Potassium, ainsi que le potentiel transmembranaire. Sur la base de ces données expérimentales, ils ont pu proposer un modèle mathématique pour la variation temporelle du potentiel membranaire et de la conductance des canaux ioniques. Attention, il s'agit bien ici d'un modèle *local* de la dynamique membranaire, on considère ce qui se passe à un point de la dendrite ou de l'axone, la dépendance spatiale ayant été évacuée du problème. Expérimentalement, ceci est réalisé en insérant un fil fait d'un matériau de très haute conductivité à l'intérieur de l'axone, assurant ainsi que le

potentiel transmembranaire devient le même tout le long de l’axone. Le modèle est décrit par quatre EDOs d’ordre 1:

$$I_a = c_m \frac{dV}{dt} + g_K n^4 (V - V_K) + g_{\text{Na}} m^3 h (V - V_{\text{Na}}) + g_L (V - V_L) , \quad (1.58)$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n , \quad (1.59)$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m , \quad (1.60)$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h , \quad (1.61)$$

Notons pour commencer qu’avec le potentiel mesuré en mV, le temps en ms, le courant en $\mu\text{A cm}^{-2}$, et la capacitance en $\mu\text{F cm}^{-2}$, l’éq. (1.58) arrive au niveau des unités SI! La quantité I_a représente un courant appliqué à travers la membrane par un agent extérieur; expérimentalement, c’est ainsi qu’on force le système à quitter son état d’équilibre. Les fonctions-barrière adimensionnelles n , m et h sont toutes trois $\in [0, 1]$, et définissent conjointement la réponse nonlinéaire des canaux ioniques au potentiel transmembranaire V . Chaque canal contient une ou plusieurs petites molécules pouvant bloquer ou ouvrir le canal, la probabilité d’ouverture étant déterminée par la valeur du potentiel transmembranaire. Le canal pour le potassium contient 4 de ces molécules, d’où le fait que le courant ionique y étant associé soit proportionnel à n^4 , tandis que pour le canal sodium il n’y a que trois de ces molécules-barrières, ce qui conduit à $i_{\text{Na}} \propto m^3$. Notons que l’effet des canaux ioniques autres que Na et K, et de toutes les sources de courants indépendantes du potentiel transmembranaire (incluant la conductance de la membrane, $\propto 1/r_m$) ont été regroupés dans le dernier terme au membre de droite de l’éq. (1.58). Ce terme est par ailleurs linéaire par rapport au potentiel transmembranaire, tout comme le serait une simple conductance passive genre Loi d’Ohm². Le tableau ci-dessous liste les valeurs des potentiels de repos³ (V_x) et coefficients de conductance (g_x) pour les trois termes de courants du modèle ($x \equiv \text{K}, \text{Na}, \text{L}$). On voit déjà que lorsque les canaux ioniques du Na et K sont ouverts, ils domineront le courant transmembranaire ($g_K, g_{\text{Na}} \gg g_L$). La capacitance membranaire est fixée à $1 \mu\text{F cm}^{-2}$ dans tout ce qui suit.

Tableau 1.2: Potentiels de repos et coefficients de conductance

Canal	V_x [mV]	g_x [ms cm^{-2}]
K	-12	36
Na	115	120
L	10.6	0.3

Les dépendances en n , m , et h des courants ioniques, ainsi que les formes analytiques suivantes pour les fonctions $\alpha(V)$ et $\beta(V)$, ont toutes été déterminées par Hodgkin et Huxley par ajustement aux données expérimentales (ils ne l’ont vraiment pas eu gratos, leur Prix Nobel!):

$$\alpha_n(V) = \frac{(0.1 - 0.01V)}{(\exp(1 - 0.1V) - 1)} , \quad \beta_n(V) = 0.125 \exp(-V/80) , \quad (1.62)$$

²Merci à mon collègue Jean-Yves Lapointe d’avoir pris le temps de m’expliquer tout ça; j’espère bien que mon résumé de ses explications leur rend justice.

³Hodgkin & Huxley ont choisi le point zéro de leur potentiel tel qu’une membrane “au repos” est caractérisée pour un potentiel $V = 0$; la valeur mesurée expérimentalement est en fait -65mV, cependant la choix de ce point zéro n’influence aucunement le comportement dynamique du système.

$$\alpha_m(V) = \frac{(2.5 - 0.1V)}{(\exp(2.5 - 0.1V) - 1)}, \quad \beta_m(V) = 4 \exp(-V/18), \quad (1.63)$$

$$\alpha_h(V) = 0.07 \exp(-V/20), \quad \beta_h(V) = \frac{1}{\exp(3 - 0.1V) + 1}. \quad (1.64)$$

avec V mesuré en milliVolt dans tous les cas. Remarquons pour une différence de potentiel fixe, les éqs. (1.59)–(1.61) sont découpés l'une de l'autre. Une solution d'équilibre ($d/dt = 0$) est obtenue directement en posant les membres de gauche égaux à zéro:

$$x_{\text{Eq}}(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)}, \quad x \equiv n, m, h. \quad (1.65)$$

Il est facile de montrer que, toujours à V fixe, toute perturbation par rapport à ces solutions d'équilibre relaxera vers l'équilibre avec un temps caractéristique donné par

$$\tau_x(V) = \frac{1}{\alpha_x(V) + \beta_x(V)}, \quad x \equiv n, m, h. \quad (1.66)$$

La Figure 1.14 porte en graphique les variations de ces valeurs d'équilibre et temps caractéristiques, sur un intervalle physiologiquement réaliste de différence de potentiel. On peut déjà remarquer que la variable m , associée aux canaux Sodium (viz. éq. (1.60)), a un temps de relaxation beaucoup plus court que n et h au voisinage du potentiel d'équilibre ($V = 0$).

1.7.3 Traitement numérique

Le traitement numérique du modèle de Hodgkin-Huxley est conceptuellement simple: il s'agit de solutionner un système de quatre EDOs couplées nonlinéairement, à partir d'une condition initiale donnée. Notre méthode de Runge-Kutta est tout à fait appropriée ici. La seule manœuvre mathématique requise est de réécrire l'éq. (1.58) sous la forme

$$\frac{dV}{dt} = \frac{1}{c_m} (I_a - g_K n^4 (V - V_K) - g_{Na} m^3 h (V - V_{Na}) - g_L (V - V_L)), \quad (1.67)$$

Ceci, de concert avec les éqs. (1.59)–(1.61), est un système de quatre EDOs ayant bien la forme générique requise (cf. l'éq. (1.1)):

$$\underbrace{\frac{d}{dt} \begin{pmatrix} V \\ n \\ m \\ h \end{pmatrix}}_u = \underbrace{\begin{pmatrix} \frac{1}{c_m} (I_a - g_K n^4 (V - V_K) - g_{Na} m^3 h (V - V_{Na}) - g_L (V - V_L)) \\ \alpha_n(V)(1 - n) - \beta_n(V)n \\ \alpha_m(V)(1 - m) - \beta_m(V)m \\ \alpha_h(V)(1 - h) - \beta_h(V)h \end{pmatrix}}_{g(u)}. \quad (1.68)$$

Les temps de relaxation associées aux réponses des canaux ioniques (voir Fig. 1.14, et en particulier la variable m) nous indiquent déjà que le pas de temps ne devrait probablement pas dépasser ~ 0.01 ms. Il est également important d'utiliser en condition initiale, pour les variables n, m, h , les valeurs d'équilibre associées à celle du potentiel membranaire initial (tel que calculé via les éqs. (1.65)–(1.66)), de manière à ce que le système soit globalement en équilibre avant de commencer à forcer le système via le terme I_a .

1.7.4 Le potentiel d'action

Bien que le but de ce projet soit justement de vous faire explorer le comportement du modèle de Hodgkin-Huxley, cette section en discute quelques propriétés importantes, afin d'aider à vous lancer là-dedans et vous donner quelques résultat numériques pouvant servir à valider votre propre version du modèle.

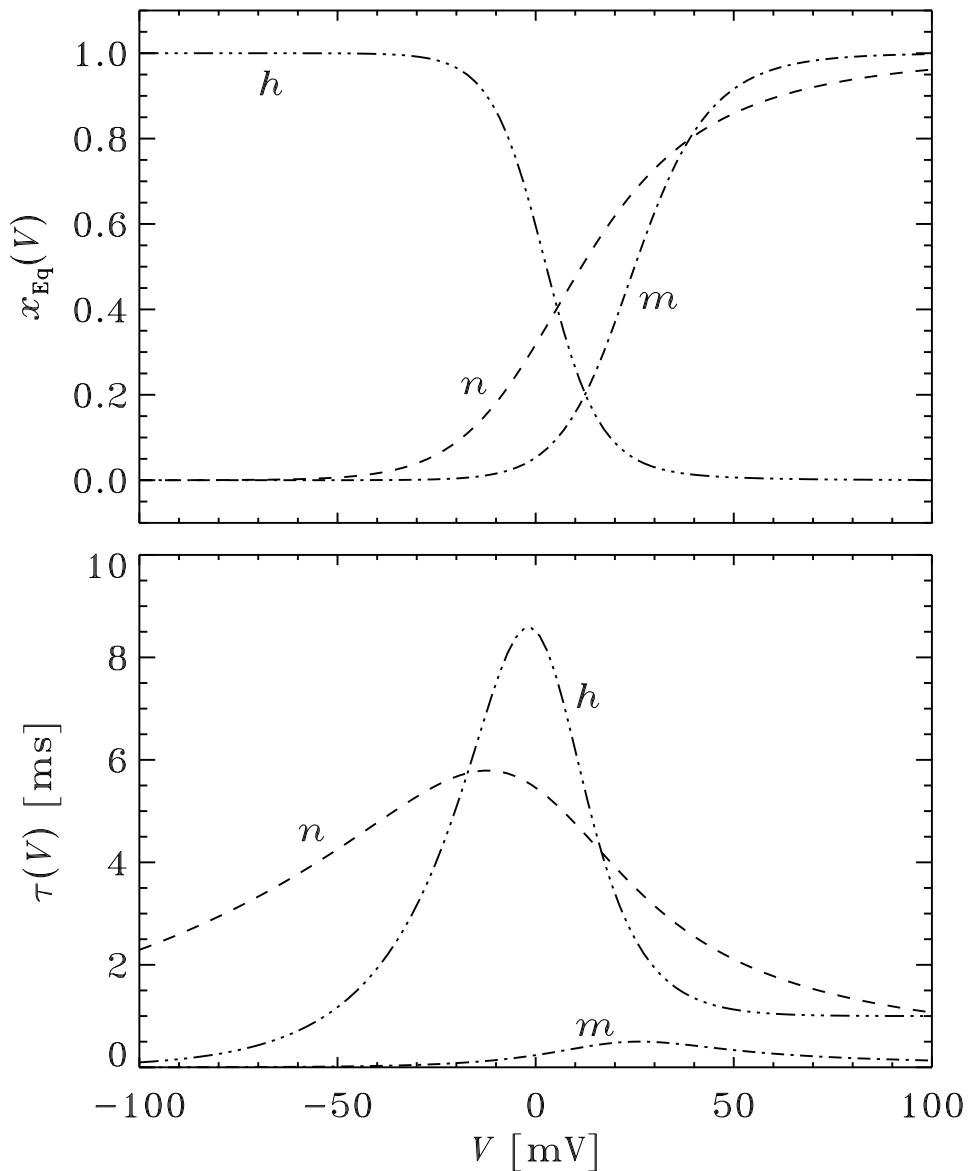


Figure 1.14: Valeurs d'équilibre (haut) et temps de relaxation (bas) pour les trois variables adimensionnelles n, m, h du modèle de Hodgkin-Huxley, en fonction de la valeur du potentiel transmembranaire V . L'échelle de potentiel utilisée par Hodgkin et Huxley fixe le potentiel d'équilibre de la membrane non-active à $V = 0$.

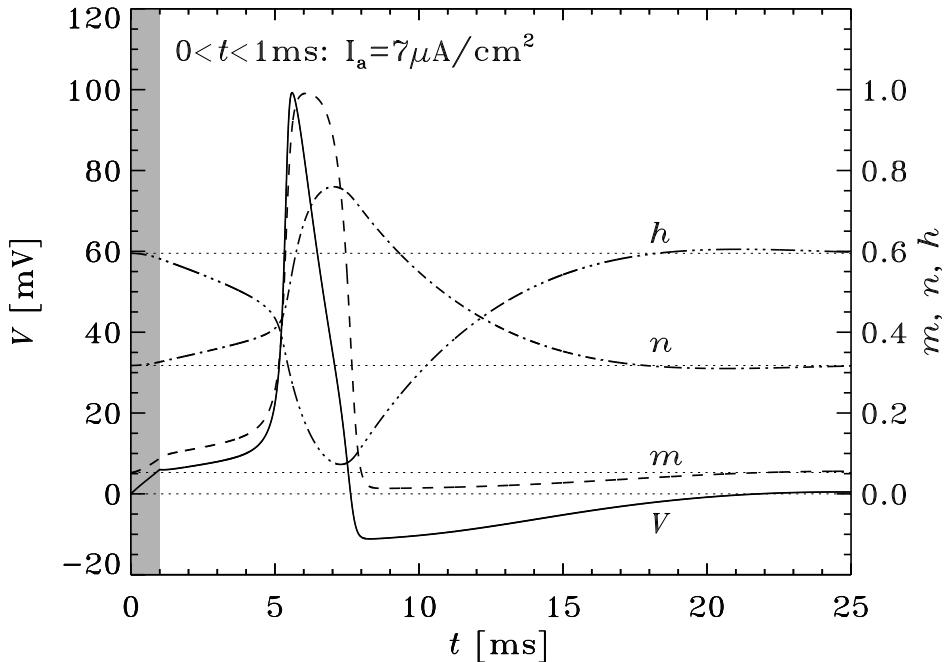


Figure 1.15: Réponse du potentiel membranaire et des fonctions-barrière à l’application d’un courant de $7 \mu\text{A cm}^{-2}$, pendant 1 ms (bande grise). Potentiel initial nul et fonctions-barrière à l’état d’équilibre correspondant (traits pointillés horizontaux).

Une des propriétés les plus importantes du modèle de Hodgkin-Huxley —en fait le modèle a été développé dans le but premier de reproduire cette propriété— est la production de pulse de voltage transmembranaire sous application externe d’un courant à travers la membrane. Ceci est illustré à la Figure 1.15, dans le cas d’une solution partant d’un état d’équilibre à $V = 0$, le système étant sujet à l’application d’un courant transmembranaire $I_a = 7 \mu\text{A cm}^{-2}$ pendant une milliseconde. Durant la phase d’application du courant externe (bande grise), le potentiel croît linéairement, comme on s’y attendrait, et les fonctions-barrière dévient légèrement de leurs valeurs d’équilibre. Une fois que I_a est retombé à zéro, le potentiel continue d’augmenter, quoique plus lentement, car les conductances des canaux ioniques sont plus grandes qu’elles ne l’étaient dans l’état d’équilibre initial. Cette croissance s’emballe à $t \simeq 5$ ms produisant un pic de potentiel (ou vague de dépolarisation) appelé *potentiel d’action*, après quoi le potentiel chute tout aussi rapidement *sous* sa valeur d’équilibre. Après $\simeq 25$ ms, le potentiel et les fonctions barrières sont tous revenus à leurs valeurs d’équilibre respectives.

Un aspect crucial de cette dynamique membranaire est que la production d’un potentiel d’action n’est possible qu’au delà d’un certain seuil du courant appliqué, comme le montre la Figure 1.16. Pour les paramètres du modèle de Hodgkin-Huxley utilisés ici, ce seuil se situe entre 6.9 et $7 \mu\text{A cm}^{-2}$, *et ne dépend pas de la durée d’application du courant extérieur*. Une fois ce seuil dépassé, la forme et l’amplitude du potentiel d’action ne dépendent que très peu de la grandeur du courant appliquée. Cependant, un fois le seuil de déclenchement du potentiel d’action dépassé, la durée de l’application du pulse de courant a un effet sur le comportement à plus long terme du modèle. Ceci est illustré à la Figure 1.17, qui montre deux simulations partant de l’état dééquilibré $V = 0$, pour des courants de 7 et $10 \mu\text{A cm}^{-2}$ appliquées de manière continue dans le temps à partir de $t = 0$. Dans le premier cas deux potentiels d’action sont produits, après quoi la membrane revient lentement à son état d’équilibre. Mais pour $I_a = 10 \mu\text{A cm}^{-2}$, un train périodique de potentiel d’action est produit, qui ne cessera que lorsque le courant appliqué sera ramené à zéro.

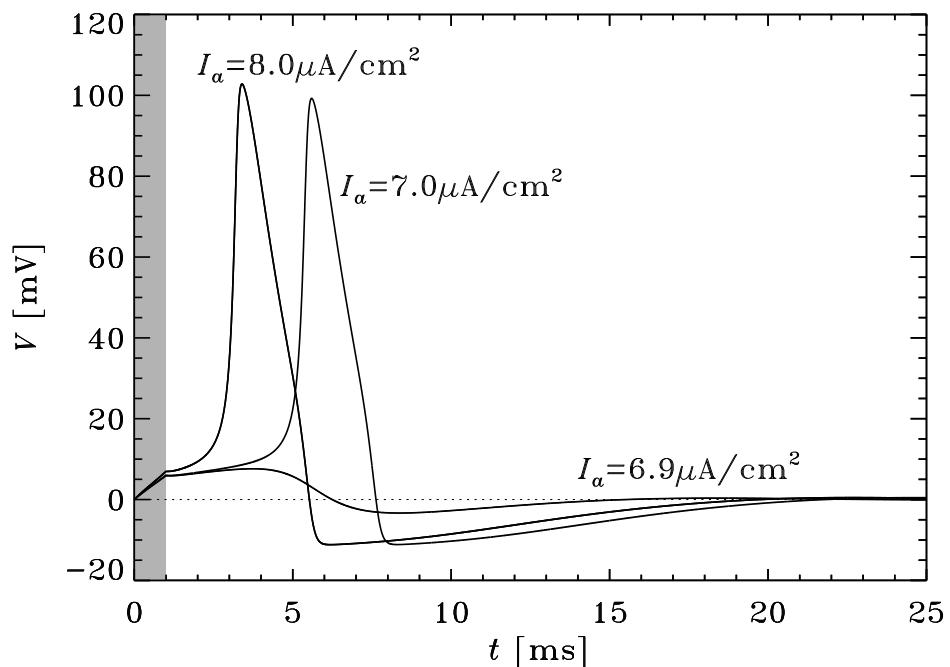


Figure 1.16: Réponse du potentiel membranaire à l'application d'un courant de 6.9 , 7.0 et $8 \mu\text{A}$ cm^{-2} , dans les trois cas pendant 1 ms (bande grise). Potentiel initial nul et fonctions-barrière à l'état d'équilibre correspondant. La production d'un potentiel d'action se fait à partir d'un certain seuil $\sim 7 \mu\text{A cm}^{-2}$, mais une fois ce seuil dépassé son amplitude ne dépend que très peu de I_a ou de la durée du pulse de courant (voir texte).

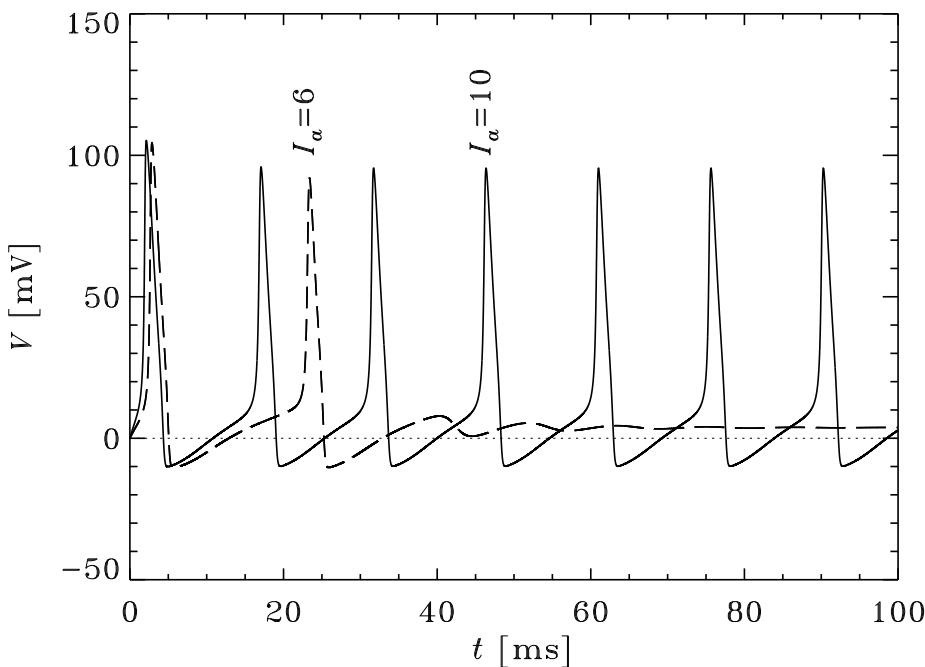


Figure 1.17: Réponse du potentiel membranaire à l'application d'un courant continu de 6 et $10 \mu\text{A cm}^{-2}$. Dans le second cas un train périodique de potentiels d'action est produit. Potentiel initial nul et fonctions-barrière à l'état d'équilibre correspondant.

1.7.5 Explorations numériques

Le projet-devoir associé à ce chapitre consiste à explorer le comportement et le type de solutions produites par le modèle de Hodgkin-Huxley. Les points suivants ne sont que des pistes pour vos explorations numériques. Votre rapport doit couvrir à la fois les aspects numériques et physiques pertinents.

1. Solutionnez numériquement le modèle de Hodgkin-Huxley en utilisant la méthode de Runge-Kutta d'ordre 4, avec pas adaptif (§1.5)
2. Mesurer les fréquence et amplitude des trains de potentiels d'action en fonction de la grandeur d'un courant I_a appliqué en mode continu, comme sur la Fig. 1.17.
3. On a vu qu'une fois le potentiel d'action déclenché, un certain intervalle de temps est requis avant que le potentiel revienne à sa valeur d'équilibre. Travaillez avec des pulses de $10 \mu\text{A cm}^{-2}$ sur 1ms, déterminer à partir de quel intervalle de temps deux pulses successifs peuvent déclencher deux potentiels d'action. Répétez l'exercice, mais cette fois avec un second pulse d'amplitude plus élevée, soit $12 \mu\text{A cm}^{-2}$, $15 \mu\text{A cm}^{-2}$, et $20 \mu\text{A cm}^{-2}$, toujours sur 1ms. Existe-t-il un intervalle temporel durant lequel il est simplement impossible de déclencher un second potentiel d'action, quelle que soit la grandeur du courant appliquée ?
4. Stimulez le modèle avec un courant appliqué dont la grandeur varie aléatoirement toutes les 2 ms. Tirez vos nombres aléatoires d'une distribution Gaussienne avec déviation standard de $3 \mu\text{A cm}^{-2}$ et centrée sur le potentiel d'équilibre $V = 0$.
5. La sclérose en plaque est une maladie auto-immunitaire dégueulasse où la membrane des dendrites et axones est lentement détruite, dégradant inexorablement ses propriétés

isolantes et rendant ainsi de plus en plus difficile le maintien la différence de potentiel transmembranaire nécessaire au déclenchement des potentiels d'action. Dans le cadre du modèle de Hodgkin-Huxley, son effet peut être simulé en augmentant le coefficient de conductance g_L . Déterminez à quelle valeur de g_L la production de potentiels d'action cesse, en fonction de la grandeur et durée du courant appliqué.

6. On a déjà noté que temps caractéristique de la fonction-barrière m est beaucoup plus court que ceux pour m et n (viz. Fig. 1.14); réduisez le modèle à trois EDOs en supposant que m demeure en tout temps à l'équilibre. Ce modèle réduit génère-t-il toujours des potentiels d'action semblable à ceux de la version originale?

1.8 Bibliographie

Ce chapitre, comme tous les autres qui suivent, est une production originale, dans le sens qu'il a été écrit de A à Z par tonton ici présent. Certaines sections peuvent toutefois être inspirées, parfois fortement, de matériel publié (même si dans de tels cas j'ai toujours tout recalculé et produit mes propres Figures et diagrammes). Les sections bibliographiques à la fin de chaque chapitre visent à attribuer le crédit là où il est du, et à fournir des références plus avancées pour ceux/celles désirant approfondir les sujets couverts.

Pas mal tous les bouquins d'analyse numérique traitent des méthodes de Runge-Kutta; voir par exemple:

Press, W.H., Teukolsy, S.A., Vetterling, W.T., & Flannery, B.P., *Numerical Recipes*, seconde éd., Cambridge University Press (1992).

La section 16.2 de cet ouvrage contient d'ailleurs une bonne discussion des techniques simples et avancées d'ajustement adaptif du pas pour les méthodes de type Runge-Kutta. L'exemple des chaines de désintégration nucléaire est tiré pas mal directement de

Gould, H., & Tobochnik, J., *An Introduction to Computer Simulation Methods*, seconde éd., Addison-Wesley (1996),

mais pour en savoir plus sur ce sujet voir le chapitre 5 de l'ouvrage suivant, qui demeure une référence classique dans le domaine:

Segré, E., *Nuclei and Particles*, seconde éd., W.A. Benjamin (1977).

L'article original de Hodgkin et Huxley mérite toujours d'être lu:

Hodgkin, A.L., & Huxley, A., *J. Physiology*, **117**(4), 500-544 (1952).

La présentation de la section 1.7 est fortement inspirée de l'excellent ouvrage suivant:

Gerstner, W., & Kistler, W.M., *Spiking Neuron Models*, Cambridge University Press (2002), dont une version préliminaire demeure disponible en domaine public sur le web (décembre 2015):

<http://lcn.epfl.ch/~gerstner/SPNM/SPNM.html>

Publié plus récemment par les mêmes auteurs, et couvrant passablement plus large que le précédent:

Gerstner, W., & Kistler, W.M., Naud, R., & Paninski, L., *Neuronal Dynamics*, Cambridge University Press (2014),

Le modèle de Hodgkin-Huxley, ainsi que ses versions dérivées, demeure un grand favori de la mathématique biologique; voir par exemple le chapitre 9 dans:

Keener, J., & Sneyd, J., *Mathematical Physiology*, Springer (1998).

Chapitre 2

Équations différentielles aux dérivées partielles

Les équations aux dérivée partielles (ci-après EDP) impliquent des dérivées par rapport à plus d'une variable indépendante. On en a déjà croisé un exemple au chapitre précédent sous la forme de l'équation du câble (1.49), où la différence de potentiel V dépend de la position x et du temps t .

La solution numérique d'EDPs (ou de systèmes d'EDPs couplées) pourrait facilement occuper tout le cours. Nous nous en tiendrons ici à des méthodes applicables aux EDPs de la forme générale:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(\alpha(x) \frac{\partial u}{\partial x} \right) + \beta(x) \frac{\partial u}{\partial x} + \gamma(x)u + s(t, x) \quad (2.1)$$

Avec $\alpha \equiv \lambda^2/\tau$, $\beta = 0$, $\gamma = -1/\tau$ et $s = 0$, on retrouve l'éq. (1.49). L'équation (2.1) est en fait une version 1D de *l'équation de transport*, dont la forme multidimensionnelle générale a l'air de:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha \nabla u) + \boldsymbol{\beta} \cdot \nabla u + \gamma u + s \quad (2.2)$$

Si on identifie α à un coefficient de diffusion, $\boldsymbol{\beta}$ à un écoulement, γ à un taux de désintégration, et s à une source spatialement localisée, cette dernière PDE décrit la dispersion dans l'environnement d'une substance radioactive libérée localement par un "incident" à la Three Mile Island ou Tchernobyl, ou Fukushima... Ceux/celles parmi vous ayant eu la bonne idée de s'inscrire en PHY-3140 en entendront parler en plus en détail en février.

Le plan de match est le suivant; on examinera d'abord séparément la définition et le comportement de divers algorithmes appliquées à différentes sous-classes de l'éq. (2.1), soit les équations de diffusion (§2.1) et d'advection (§2.3) pure, avec un détour (§2.2) par les systèmes couplés d'équations de réaction-diffusion. La section §2.4 introduit les méthodes dites implicites, numériquement plus lourdes mais souvent essentielles pour traiter de problèmes dominés par la diffusion et/ou impliquant un grand écart de temps caractéristiques.

On s'en tient dans ce chapitre aux méthodes de discréétisation spatiale basées sur les différences finies, mais la §2.7 offre un rapide survol d'autres classes de techniques, avec références en fin de chapitre pour ceux/elles désirant approfondir la chose, ainsi qu'un survol aussi bref de la généralisation des méthodes introduites tout au long du chapitre à des problèmes en plus d'une dimension spatiale (§2.6). Le chapitre clot avec un sujet demeurant très d'actualité en physique des plasmas (et en génie physique), soit le confinement des plasmas en tokamak (§2.8).

2.1 Méthodes explicites: diffusion

2.1.1 L'équation de diffusion

On considère pour commencer une classe d'EDP appellées *équations de diffusion*; en une dimension spatiale ces équations prennent la forme

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2}, \quad (2.3)$$

où D est un *coefficient de diffusion* (unités: $\text{m}^2 \text{s}^{-1}$). La version multidimensionnelle générale s'écrirait:

$$\frac{\partial u}{\partial t} = \nabla \cdot (D \nabla u), \quad (2.4)$$

L'éq. (2.4) est une forme spécifique d'une *équation de conservation*:

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{F}) = 0, \quad (2.5)$$

où la quantité $\mathbf{F} = -D \nabla u$ correspond ici au *flux diffusif*. Ceci décrit des phénomènes physiques aussi divers que la conduction thermique, la dispersion passive des polluants, ou la pénétration d'un champ magnétique dans une substance électriquement conductrice. Revenant à la forme 1D (2.3) avec D constant, la solution d'équilibre ($\partial/\partial t = 0$) se calcule facilement:

$$\frac{\partial^2 u_{\text{eq}}}{\partial x^2} = 0, \quad \rightarrow \quad u_{\text{eq}}(x) = ax + b. \quad (2.6)$$

où la valeur des constantes d'intégrations a et b est fixée par les conditions limites du problème. Supposons que $x \in [0, 5]$ avec $u(0) = 1$ and $u(5) = 0$; on a alors

$$u_{\text{eq}}(x) = 1 - x/5. \quad (2.7)$$

Notez que cette solution d'équilibre est indépendante de la valeur du coefficient de diffusion D ou de la condition initiale, mais se retrouve plutôt fixée par les conditions limites. Cependant le temps requis pour atteindre cette solution d'équilibre, lui, dépend de la forme de la condition initiale et de la grandeur de D . Ce temps peut être estimé par *analyse dimensionnelle* de l'éq. (2.3). Ceci consiste à y remplacer la dérivée spatiale par $1/\tau$ et la dérivée seconde spatiale par $1/L^2$, où L est une longueur caractéristique associée au profil spatial de la solution, et τ le temps caractéristique recherché. L'éq. (2.9) devient

$$\frac{u}{\tau} \sim \frac{uD}{L^2}, \quad (2.8)$$

et on obtient ainsi

$$\tau = \frac{L^2}{D}. \quad (2.9)$$

Notons déjà que ce temps caractéristique diminue quadratiquement avec l'échelle spatiale, ce qui implique que les petites échelles spatiales sont plus fortement affectées par la diffusion que les grandes.

2.1.2 Forward-in-Time, Centered-in-Space (FTCS)

Il s'agit maintenant de discréteriser l'éq. (2.3). On considère des mailles spatiales et temporelles équidistantes ¹:

$$x_j = j \times \Delta x, \quad j = 0, 1, \dots, J-1, \quad (2.10)$$

$$t^n = n \times \Delta t, \quad n = 0, 1, \dots, N. \quad (2.11)$$

¹ ATTENTION: ici et dans tout ce qui suit, les noeuds d'une maille spatiale de J points sont numérotés de 0 à $J-1$, en accord avec la convention Python sur la numérotation des éléments d'un tableau. Cependant, au niveau de la discréétisation temporelle l'indice "0" demeurera assigné à la condition initiale; c'est pourquoi, dans l'éq. (2.10), l'indice j va de 0 jusqu'à $J-1$, mais l'indice n de 0 jusqu'à N .

avec $n = 0$ correspondant à la condition initiale, et les conditions limites imposées à x_0 et x_{J-1} .

L'approche la plus simple est certainement d'introduire une différence finie avant pour la dérivée temporelle, et centrée d'ordre 2 pour la dérivée seconde en x :

$$\frac{u^{n+1} - u^n}{\Delta t} = D \frac{u_{j+1} - 2u_j + u_{j-1}}{(\Delta x)^2}. \quad (2.12)$$

Si les u 's au membre de gauche sont évalués à la position x_j , et ceux au membre droit évalués au pas de temps n , alors on obtient immédiatement un algorithme explicite permettant d'avancer la solution d'un pas temporel:

$$u_j^{n+1} = u_j^n + \left(\frac{D \Delta t}{(\Delta x)^2} \right) (u_{j+1}^n - 2u_j^n + u_{j-1}^n), \quad [\text{FTCS}] \quad (2.13)$$

Cet algorithme est connu sous la très subtile appellation “Forward-in-Time-Centered-in-Space”, et habituellement abbrévié FTCS. Il représente, conceptuellement, l'équivalent de la méthode d'Euler pour les EDPs.

La Figure 2.1 présente une solution FTCS utilisant comme condition initiale une quasi-discontinuité au centre du domaine spatial ($x \in [0, 5]$), décrit par un profil en fonction d'erreur:

$$u(x, t=0) = \frac{1}{2} \left[1 - \operatorname{erf} \left(\frac{x - 2.5}{0.1} \right) \right]. \quad (2.14)$$

Ce profil initial présente une variation spatialement localisée et très raide, quoique continue, la largeur de la transition entre $u = 1$ et $u = 0$ étant déterminée par le dénominateur dans l'argument de la fonction d'erreur “erf(...)" . Les conditions limites sont $u(0, t) = 1$, $u(5, t) = 0$, on a posé ici $D = 1$, et l'intégration est poussée jusqu'à $t = 2$ (trait rouge), soit suffisamment loin pour avoir pratiquement atteint la solution d'équilibre $u_{\text{eq}}(x) = 1 - x/5$. L'aplatissement du fort gradient initial est tout à fait typique des processus diffusifs. Il faut remarquer comment l'évolution initiale du profil est rapide au début, mais ralentit à mesure que le profil s'aplanit, l'atteinte de l'état d'équilibre exigeant plus de deux unités de temps ici. Ce “ralentissement” apparent de l'évolution diffusive se comprend facilement en terme du temps de diffusion (2.9) obtenu précédemment; initialement, la longueur caractéristique appropriée pour la solution est la largeur du profil de fonction d'erreur, soit 2×0.1 , ce qui conduit à $\tau = 0.04$ pour $D = 1$; cependant, une fois l'évolution poussée à $t = 1$, cette échelle caractéristique est plutôt de l'ordre de ~ 2 , donc $\tau = 4$.

L'algorithme FTCS s'avère très robuste pour traiter les problèmes de diffusion. La Figure 2.2 en montre un exemple extrême, soit une condition initiale complètement aléatoire, où les $u(x_j, 0)$ sont tirés d'une distribution aléatoire uniforme dans l'intervalle $[0, 1]$, avec comme conditions limites $u(0, t) = u(5, t) = 0$. La solution d'équilibre est donc ici $u_{\text{eq}}(x) = 0$. Ici l'échelle de variation spatiale de la condition initiale est celle de l'intervalle de maille ($\Delta x = 0.05$), et donc on peut s'attendre à ce que notre différence finie centrée dans (2.12) offre une très mauvaise approximation de la dérivée seconde au membre de droite de (2.3). C'est bien le cas, mais malgré cette imprécision initiale l'algorithme FTCS parvient à évoluer cette condition initiale vers la solution stationnaire attendue. Ici l'erreur initiale est amortie durant l'évolution subséquente; comme on le verra plus loin, ce ne sera pas toujours le cas.

La Figure 2.2 illustre aussi très bien une autre caractéristique typique des problèmes diffusifs: plus l'amplitude diminue, plus l'échelle spatiale caractérisant la solution augmente. Si on imagine la condition initiale comme la superposition de modes de Fourier, $\propto \exp(ikx)$, alors chaque mode de dissipe sur un temps caractéristique $(Dk^2)^{-1}$; les modes à grands k se dissipent donc plus rapidement, ce qui explique la croissance rapide de l'échelle spatiale “typique” des variations de $u(x, t)$: d'un point de maille ($k = 2\pi/\Delta x$) à $t = 0$, à $k = 2\pi/10$ à $t \sim 1$.

2.2 Exemple: formation de structures par réaction-diffusion

Alan M. Turing (1912–1954) était un mathématicien absolument hors pair. Durant les dernières années de sa vie, il s'est intéressé à un problème tout à fait fondamental en embryologie:

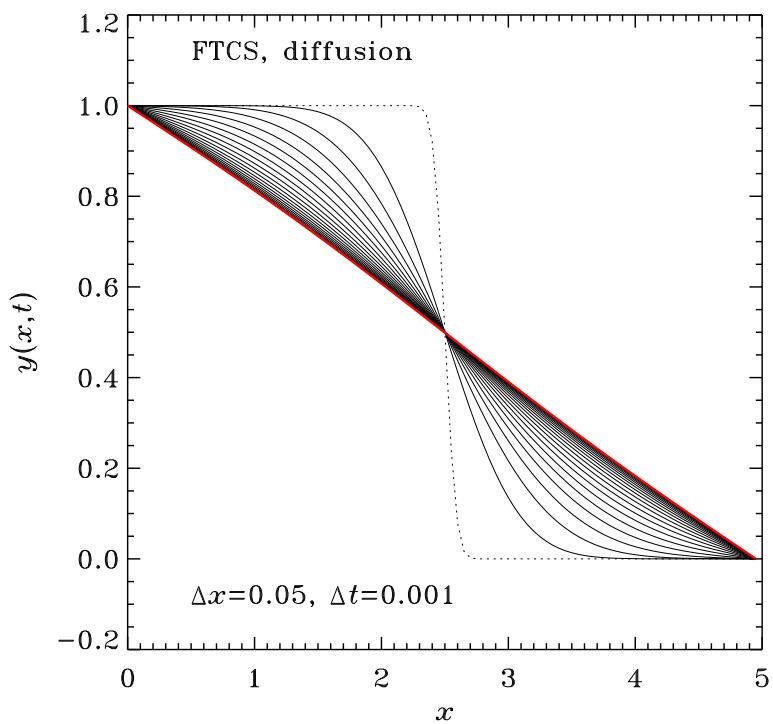


Figure 2.1: Solution numérique de l'équation de diffusion (2.3) avec (2.28) comme condition initiale (courbe pointillée), obtenue par la méthode FTCS. Les 20 courbes sont tracées à une cadence de 0.1 unités de temps. La solution à $t = 2$ (trait rouge) est à toutes fins pratiques identique à la solution d'équilibre $u_{\text{eq}}(x) = 1 - x/5$.

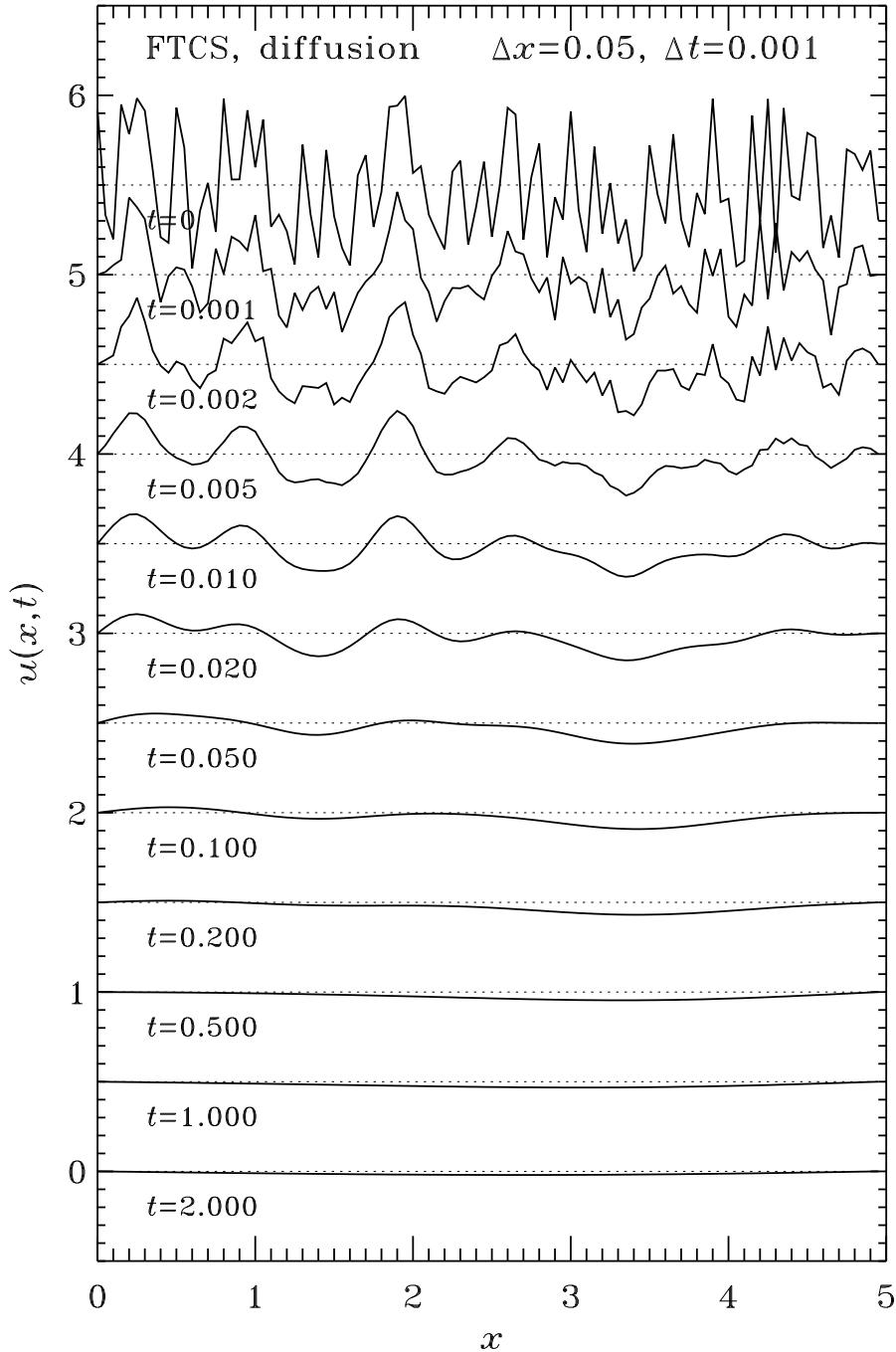


Figure 2.2: Solution FTCS de l'équation de diffusion (2.3), cette fois pour une condition initiale aléatoire. Les courbes sont décalées verticalement vers le bas pour chaque pas de temps successivement tracé, à partir de la condition initiale en haut. Notez bien que l'intervalle temporel entre chaque courbe tracée n'est pas du tout régulier, mais suit une cadence approximativement logarithmique.

comment peuvent se développer des structures ayant un haut niveau de différentiation et de symétrie à partir d'un embryon qui, dans les stades initiaux de développement, n'est qu'une sphère de cellules identiques. Expérimentalement, il était déjà accepté que la différentiation cellulaire s'organise en fonction de gradients chimiques s'établissant dans l'embryon, mais le mécanisme physico-chimique responsable demeurait un mystère. Comme on vient de le voir, un mécanisme bien compris et agissant certainement dans l'embryon, la diffusion, a l'effet contraire, soit *aplanir* les gradients. L'intuition de génie de Turing fut la réalisation qu'en présence de diffusion, certaines classes de réactions chimiques autocatalytiques, du genre de celles que nous avons déjà brièvement considérées (§1.4) peuvent conduire à des états d'équilibre caractérisés par des gradients spatiaux organisés de manière spatialement cohérente à l'échelle globale du système.

Turing s'est intéressé à des chaînes de réactants décrites par des systèmes nonlinéaires couplés de la forme générale:

$$\frac{\partial X}{\partial t} = \mu \frac{\partial^2 X}{\partial x^2} + g_1(X, Y), \quad (2.15)$$

$$\frac{\partial Y}{\partial t} = \nu \frac{\partial^2 Y}{\partial x^2} + g_2(X, Y), \quad (2.16)$$

où les $X(x, t)$ et $Y(x, t)$ représentent les concentrations de deux réactants de la chaîne. Le premier terme aux membres de droite est notre terme de diffusion linéaire classique, tandis que les termes nonlinéaires g_1, g_2 qui couplent les deux équations sont du type que nous avions considéré au chapitre précédent dans notre étude des oscillations réactives sans dépendance spatiale (§1.4; voir en particulier les éqs. (1.39)). Un aspect important de la formation de structures spatiales dans ce genre de système réaction-diffusion est que les réactants ne doivent pas diffuser pas à la même vitesse, i.e., $\mu \neq \nu$.

Dans son article de 1952 intitulé *The chemical basis of morphogenesis*, Turing présente une analyse mathématique par linéarisation des éqs. (2.15) en une dimension spatiale, ce qui était déjà tout un tour de force; mais en plus, prenant avantage de son accès à un des tout premier ordinateur, construit à l'Université de Manchester, Turing pu également calculer et présenter des solutions numériques du système nonlinéaire, à l'époque une *terra incognita* complète².

Depuis Turing, des générations de mathématicien(ne)s ont exploré le comportement de ce genre d'EDPs nonlinéaires couplées, et ont pu mettre en évidence une vaste gamme de comportements. La bibliographie en fin de chapitre donne quelques points d'entrée dans cette volumineuse littérature. Par soucis de rectitude historique, nous allons travailler ici avec les formes exactes de nonlinéarités choisies par Turing pour l'exemple numérique qu'il présente dans son article de 1952:

$$g_1(X, Y) = \frac{1}{32}(-7X^2 - 50XY + 57), \quad (2.17)$$

$$g_2(X, Y) = \frac{1}{32}(7X^2 + 50XY - 2Y - 55). \quad (2.18)$$

Notons déjà que la présence des termes en X^2 et XY indique que certaines des réactions impliquées sont de type autocatalytique. Les facteurs numériques 32, 55, etc., peuvent paraître particuliers mais ont été choisis avec soin par Turing, afin de demeurer dans une partie de l'espace des paramètres du modèle où les solutions ne divergent pas. L'ordinateur de Manchester était le nec plus ultra du temps, mais avait tout de même des capacités très limitées; Turing a donc choisi de solutionner ses équations en une dimension spatiale en supposant un domaine périodique; ceci revient à supposer que les cellules forment un anneau fermé. Donc, en terme d'une discrétisation 1D habituelle (e.g., l'éq. (2.10)), ceci revient à supposer que le point x_{J-1} est le même que le point x_0 .

Puisque les éqs. (2.17) n'incluent pas de dépendance explicite en x (seulement implicitement via les dépendances en x des réactants X et Y), en vertu de la périodicité spatiale supposée la

²Notez bien que j'utilise ici le Latin par pour faire instruit, mais bien pour ne pas tomber dans l'ambiguité phonétique “terre inconnue” versus “terrain connu”.

solution d'équilibre ($\partial/\partial t = 0$) est ici:

$$X_{\text{eq}}(x) = 1, \quad Y_{\text{eq}}(x) = 1. \quad (2.19)$$

Cette solution d'équilibre est utilisée comme condition initiale, en lui ajoutant une composante aléatoire de (relativement) faible amplitude (a):

$$X(x, 0) = X_{\text{eq}}(x) + a(2r - 1), \quad r \in [0, 1], \quad (2.20)$$

$$Y(x, 0) = Y_{\text{eq}}(x) + a(2r - 1), \quad r \in [0, 1]. \quad (2.21)$$

Nous avons déjà vu comment discréteriser les termes de type diffusifs à l'aide de l'algorithme FTCS; conservant l'idée d'évaluer explicitement tous les termes au membre de droite au pas t^n , on écrira donc:

$$g_1(X_j^n, Y_j^n) = \frac{1}{32}(-7(X_j^n)^2 - 50X_j^n Y_j^n + 57), \quad (2.22)$$

$$g_2(X_j^n, Y_j^n) = \frac{1}{32}(7(X_j^n)^2 + 50X_j^n Y_j^n - 2Y_j^n - 55). \quad (2.23)$$

La version à deux équations de notre algorithme FTCS devient alors:

$$\begin{aligned} \begin{pmatrix} X_j^{n+1} \\ Y_j^{n+1} \end{pmatrix} &= \begin{pmatrix} X_j^n \\ Y_j^n \end{pmatrix} + \frac{\Delta t}{(\Delta x)^2} \begin{pmatrix} \mu(X_{j+1}^n - 2X_j^n + X_{j-1}^n) \\ \nu(Y_{j+1}^n - 2Y_j^n + Y_{j-1}^n) \end{pmatrix} \\ &+ \frac{\Delta t}{32} \begin{pmatrix} -7(X_j^n)^2 - 50X_j^n Y_j^n + 57 \\ 7(X_j^n)^2 + 50X_j^n Y_j^n - 2Y_j^n - 55 \end{pmatrix}, \quad j = 1, \dots, J-2. \end{aligned} \quad (2.24)$$

Cet algorithme FTCS est dans un premier temps appliqué aux points de maille $1, \dots, J-2$, après quoi l'hypothèse de périodicité en x est invoquée pour assigner des valeurs aux noeuds 0 et $J-1$

$$x_0^{n+1} = x_{J-2}^{n+1}, \quad x_{J-1}^{n+1} = x_1^{n+1} \quad (2.25)$$

Notez que ceci assure que la dérivée spatiale sont également périodiques, du moins à l'ordre de la différence finie centrée utilisée pour discréteriser le terme de diffusion. Voilà, tout baigne dans l'huile, on peut foncer...

La Figure 2.3 présente une solution FTCS aux équations de Turing, utilisant des valeurs de paramètres $D = 10^{-2}$, $\mu = D$, et $\nu = D/2$, et amplitude initiale de bruit $a = 0.15$. L'évolution initiale est semblable à celle caractérisant le problème purement diffusif de la Fig. 2.2, soit un amortissement rapide des fluctuations de petites échelles spatiales, conduisant à une baisse générale de l'amplitude. Cependant, après environ une unité de temps, la solution atteint un état d'équilibre qui diffère légèrement mais significativement de $X = Y = 1$: une oscillation de type harmonique en $\exp(ikx)$ est maintenant présente, avec $k = 2\pi$ ici, et demeure stable en amplitude même si la solution est poussée jusqu'à $t = 50$. La phase spatiale de cette nouvelle solution d'équilibre est déterminé par le détail de la condition initiale (aléatoire), mais pas son amplitude ni son nombre d'onde.

La Figure 2.4 montre une seconde solution, identique en tout point à celle de la Figure 2.3, à l'exception du coefficient de diffusion D qui est ici réduit à une valeur $D = 0.00075$. Le système se stabilise encore ici à une solution d'équilibre présentant une modulation harmonique, cette fois avec un nombre d'onde $k = 8\pi$. Je vous laisse vérifier qu'un mode avec $k = 4\pi$ est produit à $D = 0.0025$, $k = 6\pi$ à $D = 0.0015$, et $k = 10\pi$ à $D = 0.0005$. La Figure 2.5 montre cette dernière structure (à droite) et celle correspondant à la Fig. 2.3 (à gauche). La teinte de gris encode ici la concentration du réactant X . Dans le cas du mode $k = 2\pi$, la solution présente un gradient vertical de concentration, qui peut servir à définir le "devant" et "l'arrière" de l'organisme embryonnaire. Dans le cas $k = 10\pi$, la symétrie pentaradiale produite pourrait

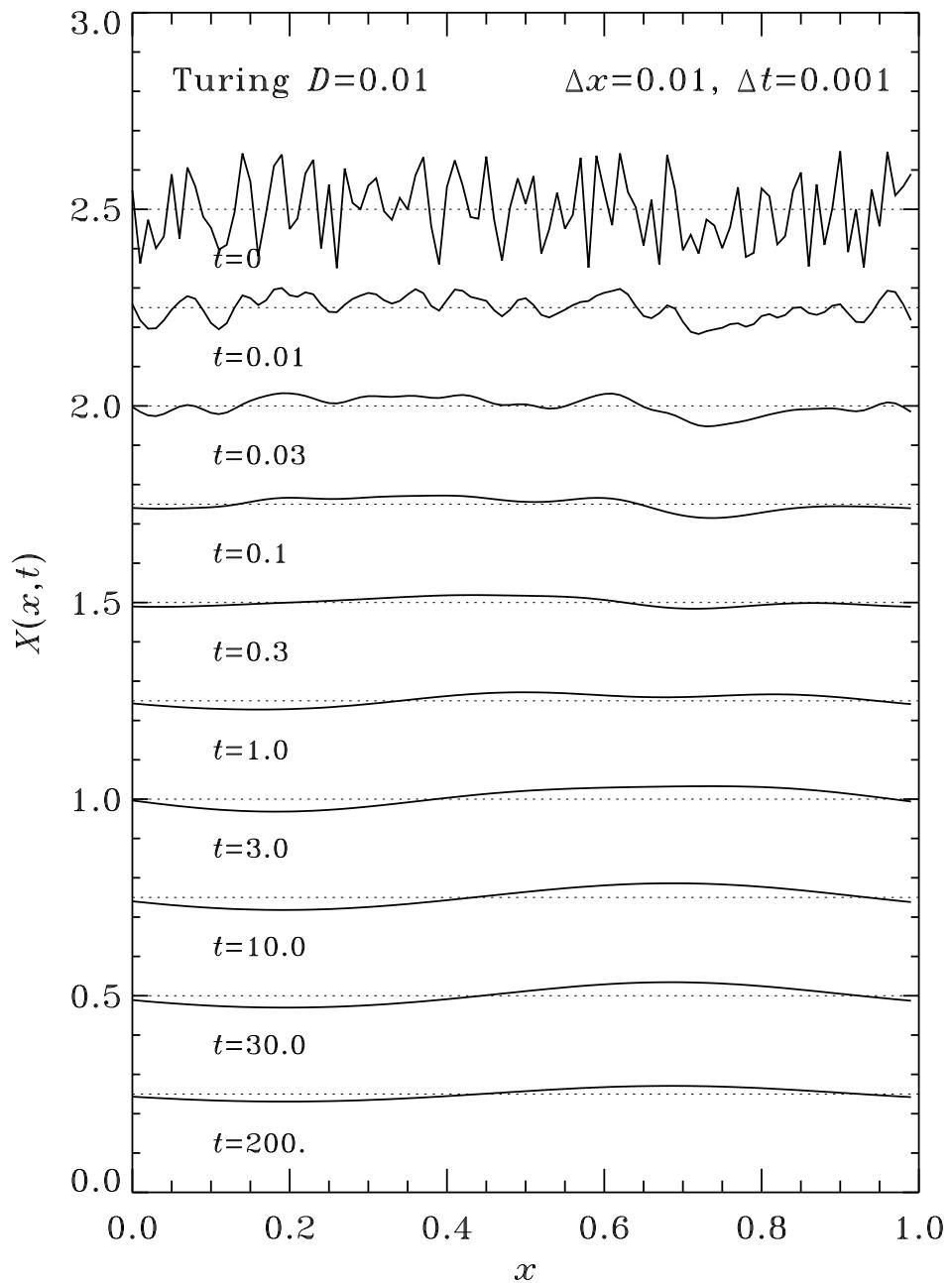


Figure 2.3: Solution FTCS des équations de Turing (2.15)–(2.17), à partir d'une solution initiale donnée par l'éq. (2.20). Le format est semblable à celui de la Fig. (2.2)

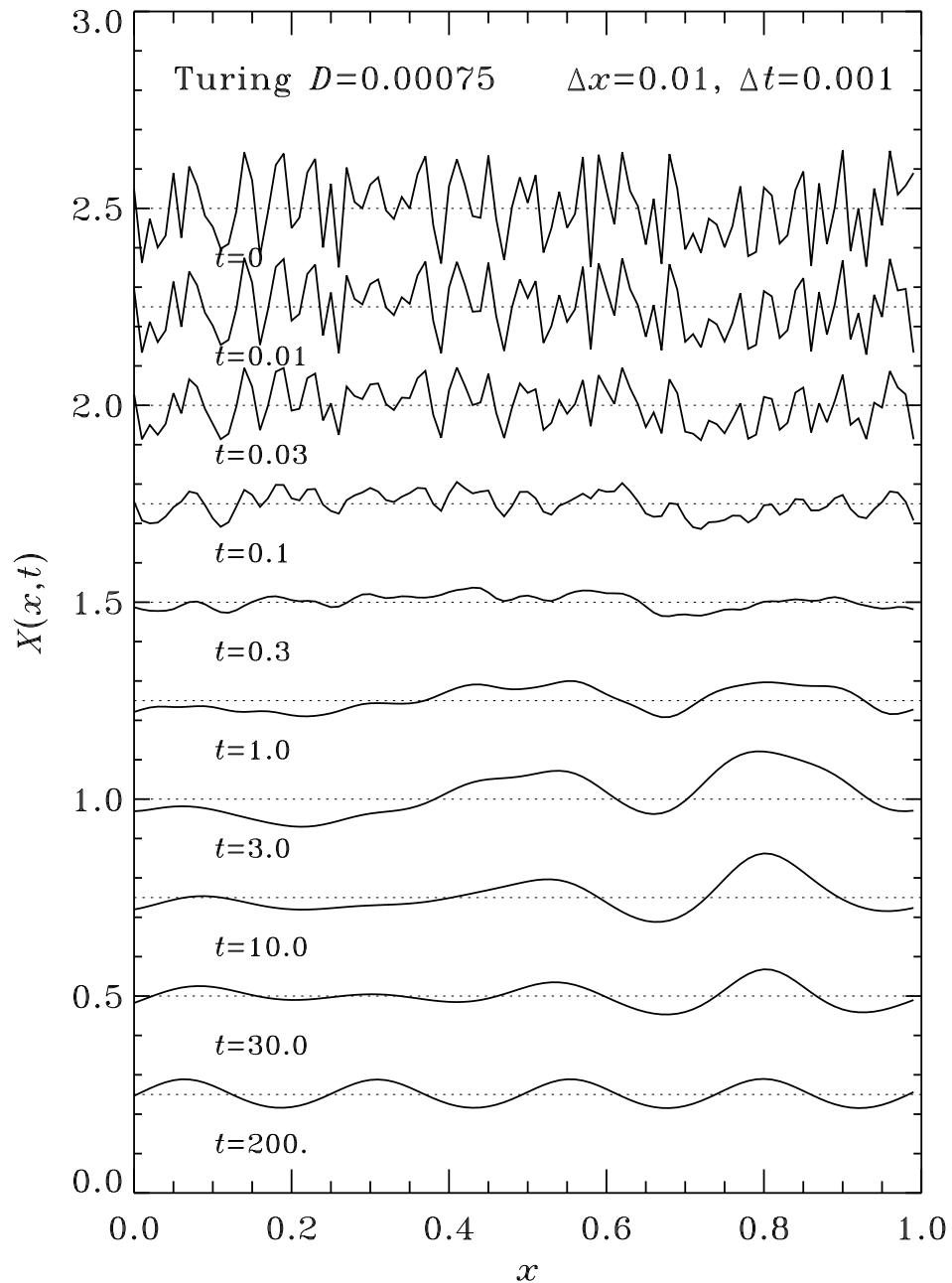


Figure 2.4: Identique à la Figure 2.3, y compris le détail de la condition initiale, sauf que cette fois on a posé $D = 0.00075$.

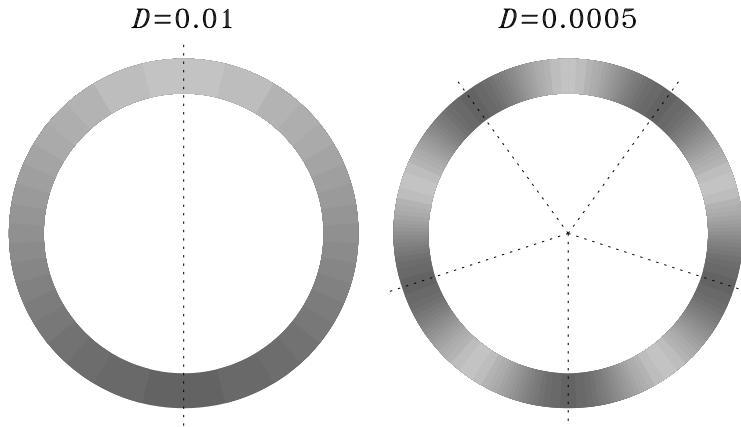


Figure 2.5: Deux solutions d'équilibre des équations de Turing, représentées cette fois dans leur géométrie annulaire véritable. La première ($D = 0.01$) dévelope un axe de symétrie permettant de différencier le “haut” et le “bas” de l’anneau; la seconde ($D = 0.0005$) développe une symétrie pentaradiale, couramment observée chez plusieurs classes de végétaux et d’invertébrés.

servir de patron au développement de cette symétrie à l'échelle de l'organisme; Turing proposait même que le développement des tentacules de l'hydre pouvait débuter de cette façon.

Dans tous les cas considérés ici —et par Turing dans son article de 1952— les grandeurs des gradients de concentrations produit à l'équilibre demeurent modestes, mais ce qui compte c'est de produire “spontanément” un gradient permettant d'imposer un pattern spatial cohérent à un processus de croissance. L'intérêt du genre de processus physico-chimique étudié par Turing dépasse largement le “simple” développement embryonnaire; Et du point de vue mathématico-physics, les solutions stationnaires obtenues ci-dessus sont exemple de *bris spontané de symétrie*.

2.3 Méthodes explicites: advection

Passons maintenant à une autre classe d'EDP, soit les équations dites d'advection:

$$\frac{\partial u}{\partial t} = -v \frac{\partial u}{\partial x} . \quad (2.26)$$

Il s'agit encore une fois ici d'une équation de conservation du type général (2.5), avec cette fois $\mathbf{F} = \mathbf{v}u$ correspondant au *flux advectif* et la vitesse v supposée constante dans cette forme 1D. Cette EDP accepte des solutions analytiques de la forme générale

$$u(x, t) = f(x - vt) , \quad (2.27)$$

où f peut être n'importe quelle fonction d'une variable, incluant même une ou plusieurs discontinuités. Ceci décrit une condition initiale de la forme $f(x, t_0)$ se déplaçant dans la direction des x croissant (pour $v > 0$; le contraire sinon) tout en préservant la forme du profil initial. La Figure 2.6 montre une telle solution analytique, dans le cas de notre profil en fonction d'erreur utilisé précédemment, cette fois avec la quasi-discontinuité initialement localisée à $x = 0.5$:

$$u(x, t = 0) = \frac{1}{2} \left[1 - \text{erf} \left(\frac{x - 0.5}{0.1} \right) \right] . \quad (2.28)$$

avec $v = 0.4$ dans l'éq. (2.26). Ici la solution est suivie sur un intervalle de temps $t_f - t_0 = 10$, et la solution est tracée sur une cadence $\delta t = 1$. Le profil initial conserve sa forme tout en se déplaçant vers la droite, pour un déplacement final de 4 unités spatiales.

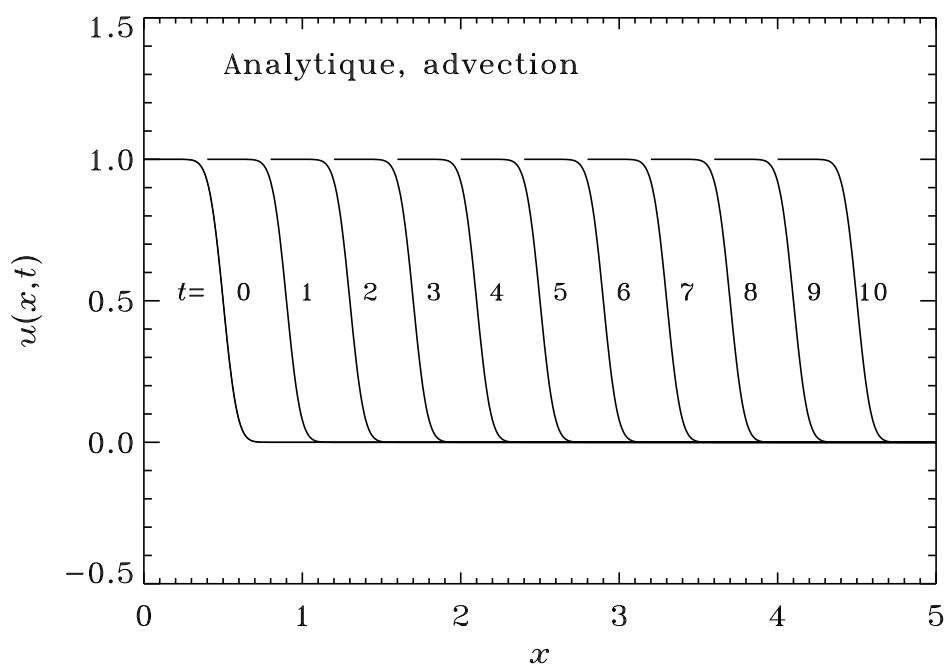


Figure 2.6: Solution analytique de l'équation d'advection (2.26) avec $v = 0.4$, conditions limites $u(x = 0, t) = 1$ et $u(x = 5, t) = 0$, et avec (2.28) comme condition initiale. La solution est tracée à une cadence temporelle $\delta t = 1$ et est calculée sur un intervalle de 10 unités de temps, au bout desquels le profil initial s'est déplacé vers la droite de $\delta x = 4$ unités spatiales tout en conservant sa forme.

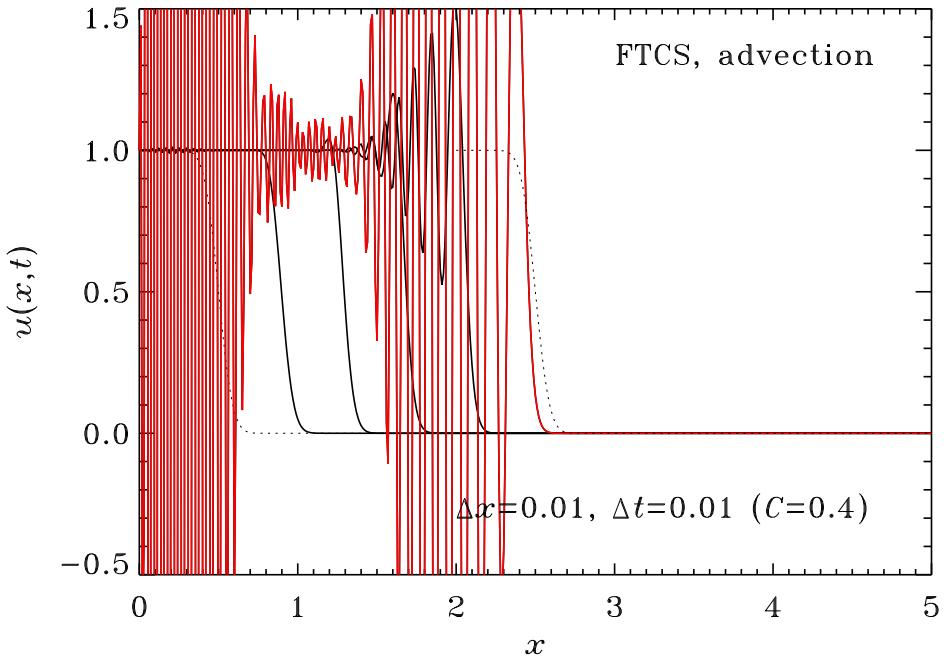


Figure 2.7: Solution numérique de l'équation d'advection (2.26) avec (2.28) comme condition initiale, obtenue par la méthode FTCS. La solution est tracée à une cadence temporelle $\delta t = 1$, soit la même que sur la Fig. 2.6. Le trait rouge montre la solution à $t = 2$.

2.3.1 Forward-in-Time, Centered-in-Space (FTCS), bis

L'approche numérique à l'éq. (2.26) la plus simple est encore une fois d'introduire une différence finie avant pour la dérivée temporelle, et centrée pour la dérivée spatiale, comme nous l'avions fait pour l'équation de diffusion pure:

$$\frac{u^{n+1} - u^n}{\Delta t} = -v \frac{u_{j+1} - u_{j-1}}{2\Delta x} \quad (2.29)$$

Si les u 's au membre de droite sont évalués au pas de temps n , alors on retombe sur l'algorithme explicite FTCS nous permettant d'avancer la solution d'un pas temporel:

$$u_j^{n+1} = u_j^n - \left(\frac{v\Delta t}{2\Delta x} \right) (u_{j+1}^n - u_{j-1}^n), \quad [\text{FTCS}] \quad (2.30)$$

La Figure 2.7 illustre une solution FTCS de l'éq. (2.26), en principe comparable à la solution analytique de la Fig. 2.6. Le début de l'évolution se développe comme on l'anticiperait, le profil initial se déplaçant vers la droite à vitesse 0.4. Cependant, déjà à $t = 0.8$ (seconde courbe ne trait plein), une petite bosse est visible en aval du front. À $t = 1.2$, cette bosse est devenue une substantielle oscillation dont l'amplitude est maximale au front et décroît graduellement en aval, et augmente inexorablement avec le temps. À $t = 2.0$ (trait rouge), la solution est complètement foutue, avec les oscillations perdurant à très haute amplitude jusqu'à $x = 0$. Remarquablement, le front se retrouve presqu'exactement à la position anticipée selon la solution analytique (pointillés).

2.3.2 Analyse de stabilité de von Newmann

La divergence de la solution présentée à la Fig. 2.7 est due au développement d'une instabilité non pas physique, mais d'origine purement numérique. L'existence d'une elle instabilité peut

être établie à l'aide une technique d'analyse proposée par John von Neumann (1903-1957), un autre des grands pionniers de l'âge informatique. L'idée est de voir sous quelles conditions une petite perturbation du genre

$$u_j^n = \xi^n \exp(ik j \Delta x) \quad (2.31)$$

est amplifiée ou amortie par un schéma numérique donné. Il faut bien comprendre ce que veut dire cette expression. Le produit $j \Delta x$, c'est la version discrétisée de notre variable spatiale x , donc la quantité $\exp(ik j \Delta x)$ correspond à un mode harmonique de longueur d'onde $\propto k^{-1}$. L'amplitude (complexe) de ce mode est donné par le terme ξ^n , et croira avec le temps —tel que discrétisé par l'indice n — si $|\xi|^2 \equiv \xi^* \xi > 1$, ou le “*” indique le conjugué complexe. Remarquez bien que selon cette notation,

$$u_j^{n+1} = \xi^{n+1} \exp(ik j \Delta x) = \xi \times (\xi^n \exp(ik j \Delta x)) , \quad (2.32)$$

$$u_{j-1}^n = \xi^n \exp(ik(j-1) \Delta x) = \exp(-ik \Delta x) \times (\xi^n \exp(ik j \Delta x)) , \quad (2.33)$$

etc. La stabilité d'un schéma numérique comme l'algorithme FTCS (2.30) peut donc être testée y en substituant l'éq. (2.31), et en vérifiant que $|\xi| < 1$ pour toutes les valeurs de k qui peuvent être casées dans le domaine spatial. Dans le cas du schéma FTCS cette substitution donne ici:

$$\xi^{n+1} \exp(ik j \Delta x) =$$

$$\xi^n \exp(ik j \Delta x) - \left(\frac{v \Delta t}{2 \Delta x} \right) (\xi^n \exp(ik(j+1) \Delta x) - \xi^n \exp(ik(j-1) \Delta x)) . \quad (2.34)$$

On divise tout ça par $\xi^n \exp(ik j \Delta x)$, ce qui conduit à

$$\xi = 1 - \left(\frac{v \Delta t}{2 \Delta x} \right) \underbrace{\left(\exp(ik \Delta x) - \exp(-ik \Delta x) \right)}_{2i \sin(k \Delta x)} , \quad (2.35)$$

$$= 1 - i \left(\frac{v \Delta t}{\Delta x} \right) \sin(k \Delta x) , \quad (2.36)$$

d'où

$$|\xi|^2 \equiv \xi^* \xi = 1 + \left(\frac{v \Delta t}{\Delta x} \right)^2 \sin^2(k \Delta x) \geq 1 \forall k . \quad (2.37)$$

On en conclut que la méthode FTCS est *inconditionnellement instable* lorsqu'appliquée au problème d'advection pure (i.e., l'éq. (2.26)). Le résultat désastreux de la Figure 2.7 est donc expliqué, mathématiquement parlant.

Vous n'aurez pas oublié (j'espère...) que la méthode FTCS s'était avérée en toute apparence stable dans le cas de l'équation de diffusion (2.3). L'analyse de von Neumann s'effectue encore en y substituant l'éq. (2.31) dans (2.12):

$$\xi^{n+1} \exp(ik j \Delta x) = \xi^n \exp(ik j \Delta x)$$

$$+ \left(\frac{D \Delta t}{(\Delta x)^2} \right) (\xi^n \exp(ik(j+1) \Delta x) - 2\xi^n \exp(ik j \Delta x) + \xi^n \exp(ik(j-1) \Delta x)) . \quad (2.38)$$

ce qui conduit maintenant à:

$$\xi = 1 + \left(\frac{D \Delta t}{(\Delta x)^2} \right) \left(\underbrace{\exp(ik \Delta x) + \exp(-ik \Delta x)}_{2 \cos(k \Delta x)} - 2 \right)$$

$$\begin{aligned}
&= 1 - 2 \left(\frac{D\Delta t}{(\Delta x)^2} \right) \underbrace{\left(1 - \cos(k\Delta x) \right)}_{2 \sin^2(k\Delta x/2)} \\
&= 1 - \frac{4D\Delta t}{(\Delta x)^2} \sin^2 \left(\frac{k\Delta x}{2} \right). \tag{2.39}
\end{aligned}$$

La méthode FTCS demeurera donc stable ($|\xi| < 1$) tant que

$$-1 < 1 - \frac{4D\Delta t}{(\Delta x)^2} \sin^2 \left(\frac{k\Delta x}{2} \right) < 1; \tag{2.40}$$

La seconde inégalité est immédiatement assurée, et la première conduit à la contrainte:

$$\frac{D\Delta t}{(\Delta x)^2} < \frac{1}{2}. \tag{2.41}$$

L'algorithme FTCS est donc *conditionnellement stable* dans le cas d'un problème de diffusion pure. Il y a un message important ici: la stabilité d'un algorithme numérique dépend généralement du problème auquel on applique cet algorithme !

L'équation (2.41) indique qu'une diminution du pas spatial Δx exige une diminution du pas temporel, Δt devant diminuer *quadratiquement* avec le pas spatial Δx pour garder fixe la valeur du membre de gauche. Avec $D = 1$, $\Delta x = 0.05$ et $\Delta t = 10^{-3}$, on a $D\Delta t/(\Delta x)^2 = 0.4$, donc la solution FTCS de la Figure 2.1 est dangereusement près de la limite de stabilité (évidemment que c'était arrangé avec le gars des vues...!). Si une grande précision est recherchée (Δx très petit), le pas temporel devient excessivement petit. Il faudra alors trouver autre chose. On y reviendra plus loin (§2.4), pour le moment essayons de nous trouver un algorithme qui ne nous pète pas au nez pour notre problème d'advection pure.

2.3.3 La méthode de Lax

L'algorithme FTCS peut être stabilisé par une modification à prime abord très *ad hoc*. Sur examen du membre de droite de l'éq. (2.30), on constate que la dérivée est estimée par une différence finie centrée sur j , tandis que le premier terme au membre de droite est simplement évalué à j . Évaluons plutôt ce dernier comme une moyenne sur les points de maille $j - 1$ et $j + 1$:

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) - \left(\frac{v\Delta t}{2\Delta x} \right) (u_{j+1}^n - u_{j-1}^n), \quad [\text{Lax}] \tag{2.42}$$

Cette fois l'application de l'analyse de stabilité de von Neumann, soit la substitution de l'éq. (2.31) dans l'expression ci-dessus, conduit à:

$$\xi = \cos(k\Delta x) - i \frac{v\Delta t}{\Delta x} \sin(k\Delta x) \tag{2.43}$$

d'où

$$|\xi|^2 = \cos^2(k\Delta x) + \left(\frac{v\Delta t}{\Delta x} \right)^2 \sin^2(k\Delta x), \tag{2.44}$$

La contrainte $|\xi| < 1$ conduit au *critère de stabilité de Courant-Friedrich-Lowy*, ou d'usage plus courant (!), *Critère de Courant*:

$$C \equiv \frac{|v|\Delta t}{\Delta x} \leq 1. \tag{2.45}$$

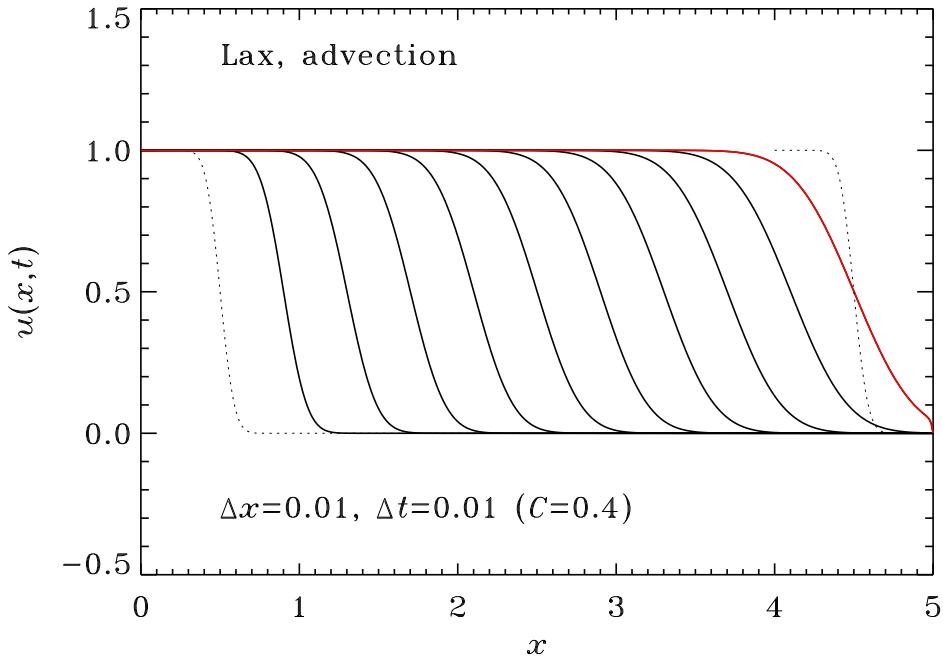


Figure 2.8: Solution numérique de l'équation d'advection (2.26) avec (2.28) comme condition initiale, obtenue par la méthode de Lax. La solution à $t = 10$ est tracée en rouge, et le trait pointillé donne la solution analytique correspondante.

où C est le *Nombre de Courant*. L'analyse suggère donc que l'algorithme de Lax devrait produire une solution stable tant que le pas de temps est choisi suffisamment petit, par rapport à la vitesse v d'advection et du pas spatial Δx , en accord avec le critère de Courant (2.45). La Figure 2.8 montre une solution utilisant des pas spatial et temporel identiques à la solution FTCS de la Figure 2.7, correspondant donc à $C = 0.4$. La solution est définitivement stable, et la quasi-discontinuité de la condition initiale est transportée vers la droite, comme il se doit, mais s'aplanit graduellement avec le temps. L'origine de ce comportement “surdiffusif” émerge en soustrayant u_j^n de chaque côté de l'éq. (2.46):

$$u_j^{n+1} - u_j^n = \frac{1}{2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n) - \left(\frac{v\Delta t}{2\Delta x}\right)(u_{j+1}^n - u_{j-1}^n); \quad (2.46)$$

en divisant ensuite par Δt et en réorganisant algébriquement les termes au membre de droite, on arrive à:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -v \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + \frac{(\Delta x)^2}{2\Delta t} \left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} \right). \quad (2.47)$$

Ceci représente une discréttisation par différence finie centrée d'ordre deux de l'EDP suivante:

$$\frac{\partial u}{\partial t} = -v \frac{\partial u}{\partial x} + \left(\frac{(\Delta x)^2}{2\Delta t} \right) \frac{\partial^2 u}{\partial x^2}. \quad (2.48)$$

Ceci n'est plus l'équation de pure advection que l'on croyait solutionner! Le second terme au membre de droite est un terme de type diffusion, avec la quantité $(\Delta x)^2/2\Delta t$ jouant le rôle du coefficient D dans l'éq. (2.3) traitée précédemment. Avec $\Delta x = 0.01$ et $\Delta t = 0.01$, on a $D = 0.005$ ici et une échelle spatiale initiale $L \sim 0.1$, conduisant donc à un temps de diffusion (2.9) $\tau = 2$, soit cinq fois moins que la durée du calcul sur la Fig. 2.8. Il n'est donc pas

surprenant de voir le fort gradient initial s'aplanir au fil du temps. Ce qui est important de noter ici est que l'apparition de ce terme diffusif supplémentaire inattendu est entièrement dûe à la manœuvre *ad hoc* de Lax consistant à évaluer le u_j^n au membre de droite de l'algorithme FTCS comme une moyenne centrée des points voisins (cf. éq. 2.46). Ce choix, qui semblait justifiable du point de vue strictement numérique, a effectivement introduit l'équivalent d'un processus diffusif physique; l'algorithme de Lax est stable, mais ne solutionne plus le problème physique tel qu'initiallement posé, soit l'équation d'advection pure (2.26)!

2.3.4 La méthode de Lax-Wendroff

Examinons un dernier algorithme applicable non seulement à notre équation d'advection, mais en fait à toute EDP pouvant s'exprimer sous la forme conservative introduite précédemment. L'idée sous-jacente à cet algorithme nous sera par ailleurs fort utile une fois arrivé au projet qui clot ce chapitre.

Commençons par reécrire l'éq. (2.26) sous la forme conservative équivalente

$$\frac{\partial u}{\partial t} = -\frac{\partial F}{\partial x}. \quad (2.49)$$

où $F = vu$ est le *flux advectif* de la quantité u . L'algorithme de Lax-Wendroff est basé sur une évaluation de ce flux à mi-pas dans l'espace et dans le temps. La première étape est d'évaluer u de manière centrée spatialement et à mi-pas temporellement, via l'algorithme de Lax:

$$u_{j+1/2}^{n+1/2} = \frac{1}{2}(u_{j+1}^n + u_j^n) - \left(\frac{v\Delta t}{2\Delta x}\right)(u_{j+1}^n - u_j^n); \quad (2.50)$$

ensuite, on calcule les flux correspondants; ici c'est simplement:

$$F_{j+1/2}^{n+1/2} = v u_{j+1/2}^{n+1/2} \quad (2.51)$$

Finalement, u est calculé de manière véritablement centrée dans l'espace et le temps, soit:

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}(F_{j+1/2}^{n+1/2} - F_{j-1/2}^{n+1/2}). \quad (2.52)$$

Cet algorithme se généralise facilement à l'équation de diffusion (2.3) et remplaçant le flux advectif $F = vu$ par le flux diffusif $F = -D\partial u/\partial x$, ou encore à des situations incluant advection et diffusion, et exprimant le flux comme la somme de contributions advective et diffusive: $F = vu - D\partial u/\partial x$.

Revenant au cas de l'advection pure, cette fois l'application de l'analyse de stabilité de von Neumann donne

$$\xi = 1 - iC \sin(k\Delta x) - C^2(1 - \cos(k\Delta x)), \quad (2.53)$$

où $C = v\Delta t/\Delta x$ est le nombre de Courant introduit précédemment. Comme d'habitude, la stabilité requiert que

$$|\xi|^2 = 1 - C^2(1 - C^2)(1 - \cos(k\Delta x))^2 < 1; \quad (2.54)$$

ce qui conduit encore une fois au critère de Courant, $C < 1$.

La Figure 2.9 présente une solution Lax-Wendroff équivalente à celle présentées pour la méthode de Lax à la Fig. 2.8. La comparaison entre ces deux Figures est intéressante et instructive. La méthode de Lax-Wendroff évite le comportement surdiffusif de l'algorithme de Lax, le prix à payer étant une erreur croissant très lentement et demeurant localisée en aval immédiat de la discontinuité.

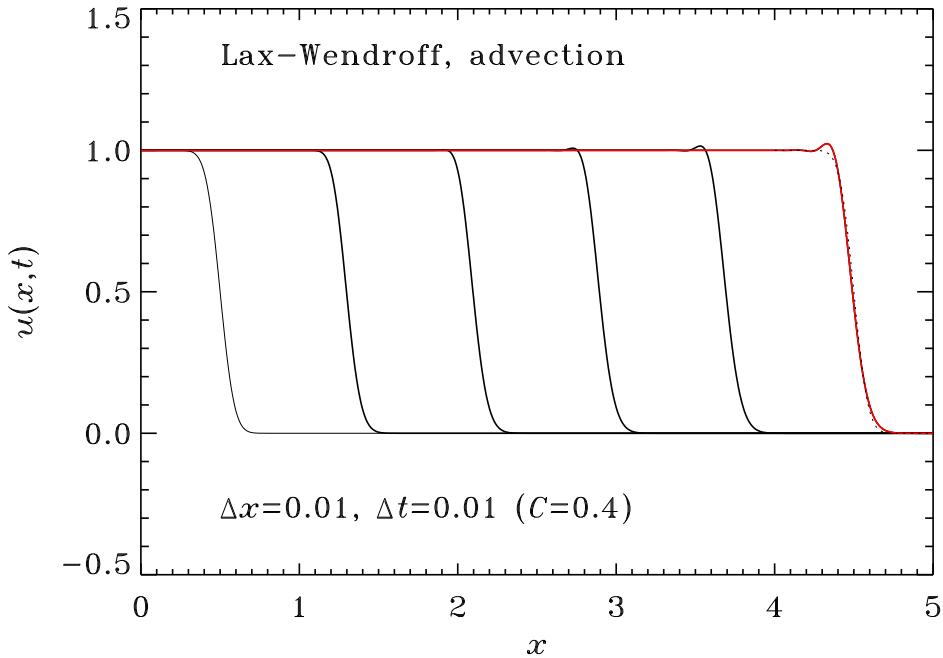


Figure 2.9: Solution numérique de l'équation d'advection (2.26) avec (2.28) comme condition initiale, obtenue par la méthode de Lax-Wendroff. La solution est tracée à une cadence temporelle $\delta t = 2$. La solution à $t = 10$ est tracée en rouge, et le trait pointillé donne la solution analytique correspondante.

2.4 Méthodes implicites

Le critère de Courant, que ce soit pour une PDE décrivant la diffusion (éq. (2.3)) ou l'advection (éq. (2.26)) a comme conséquence que tout raffinement de la maille spatiale (Δx) doit s'accompagner d'un raffinement de la maille temporelle (δt) afin de maintenir la stabilité. Ceci demeure le cas pour des PDEs (ou systèmes de PDE couplées) décrivant des systèmes physiques plus complexes que l'advection ou diffusion pures, par exemple les écoulements fluides, si on s'en tient aux méthodes explicites dans le temps. Le critère de Courant peut donc souvent devenir très contraignant pour des simulations numériques. Les méthodes implicites dans le temps permettent ce transfert d'information global, et réussissent donc (souvent mais pas toujours) à s'affranchir du critère de Courant.

2.4.1 Méthode d'Euler implicite

Revenons à notre problème de diffusion pure, tel que décrit par l'éq. (2.3), et reprenons l'idée de bon vieil algorithme FTCS, mais cette fois en évaluant le membre de droite non pas au temps n , mais plutôt au pas $n + 1$; l'éq. (2.13) devient:

$$u_j^{n+1} = u_j^n + \left(\frac{D\Delta t}{(\Delta x)^2} \right) (u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}) \quad [\text{Euler implicite}] . \quad (2.55)$$

Il n'est plus possible de calculer directement les u^{n+1} comme auparavant, puisque le membre de droite couple maintenant les triades successives de points $(j-1, j, j+1)$. L'éq. (2.55) décrit maintenant un système d'équation linéaires pour les u^{n+1} 's, pouvant s'exprimer sous la forme matricielle:

$$[K_{ij}]\{u_j^{n+1}\} = \{r_i^n\} , \quad (2.56)$$

```

1 import numpy as np
2 def tridiag(a,b,c,r,u,J):
3 #-----
4 # Solveur tridiagonal de Press et al. (1992), Section 2.4
5 # a[J],b[J],c[J] contiennent les trois diagonales de la matrice
6 # r[J] est le vecteur RHS
7 # u[J] a l'entrée est la solution au pas n, en sortie à n+1
8 #-----
9     gam=np.zeros(J)                      # vecteur de travail
10    bet=b[0]
11    if bet == 0: return 1
12    u[0]=r[0]/bet
13    for j in range(1,J-1):               # décomposition LU
14        gam[j]=c[j-1]/bet
15        bet =b[j]-a[j]*gam[j]
16        if bet == 0: return 1
17        u[j] =(r[j]-a[j]*u[j-1])/bet
18    for j in range(J-2,0,-1):           # backsubstitution
19        u[j]-= gam[j+1]*u[j+1]
20    return 0
21 # END tridiag

```

Figure 2.10: Solveur pour système tridiagonal d'équations linéaires. Adaptation Python de la routine `tridiag` discutée à la §2.4 de Press et al. (voir bibliographie). Notez bien que Python numérote de 0 à $J - 1$ les éléments d'un tableau de taille J .

où $r_i^n = u_i^n$ ici, et dans le cas présent de l'équation de diffusion pure discrétisée par différences finies centrées d'ordre deux, la matrice $[K]$ est tridiagonale:

$$[K_{ij}] = \begin{pmatrix} b_0 & c_0 & & & & & 0 \\ a_1 & b_1 & c_1 & & & & \\ & a_2 & b_2 & c_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_{J-2} & b_{J-2} & c_{J-2} & \\ 0 & & & & a_{J-1} & b_{J-1} & \end{pmatrix} \quad (2.57)$$

avec

$$a_i = -D\Delta t / (\Delta x)^2, \quad i = 1, \dots, J-1, \quad (2.58)$$

$$b_i = 1 + 2D\Delta t / (\Delta x)^2, \quad i = 0, \dots, J-1, \quad (2.59)$$

$$c_i = -D\Delta t / (\Delta x)^2, \quad i = 0, \dots, J-2. \quad (2.60)$$

La nécessité de solutionner un tel système d'équations algébriques à chaque pas de temps classe l'algorithme (2.55) comme étant *implicite*, en contraste à ceux des §2.1—2.3, qui sont catégorisés comme explicites. Cependant, un tel système tridiagonal est facile à solutionner numériquement; la Figure 2.10 offre un exemple de code Python le faisant, traduit et adaptée de la routine équivalente en FORTRAN présentée dans l'ouvrage de Press et al. cité en bibliographie en fin de chapitre. Comme les coefficients de la matrice $[K]$ ne dépendent ni de u (le problème étant linéaire) ni de t , la matrice $[K]$ —ou plutôt les vecteurs $\{a_i\}$, $\{b_i\}$ et $\{c_i\}$ définissant ses trois diagonales— peut être construite en placee en mémoire avant le début de l'itération temporelle, après quoi l'exécution d'un pas de temps ici implique

1. le calcul du vecteur au membre de droite; ici il ne contient que la solution u_n au pas de temps précédent;

2. la solution du système algébrique même.

L'imposition des conditions limites mérite qu'on s'y attarde; ce n'est pas aussi simple que de solutionner le système (2.56), en ensuite imposer les valeurs limites au premier et dernier point de maille. Pour que les conditions limites puissent être incorporées dans la solution du système linéaire, on doit directement modifier la matrice $[K]$ et le vecteur RHS $\{r\}$. Supposons que l'on doive imposer $u^* = 1$ à $x = 0$; il suffit de remplacer l'élément b_0 par un "1", mettre à zéro c_0 et a_1 , et insérer la valeur requise ($u_0^* = 1$) au membre de droite: $r_0 = 1$. La solution du système produira alors $u_0^{n+1} = 1$ comme désiré; cependant la mise à zéro de a_1 doit être compensée en soustrayant $a_1 u_0$ à r_1 ; bref, le système linéaire devient:

$$\begin{pmatrix} 1 & 0 & & & 0 \\ 0 & b_1 & c_1 & & \\ a_2 & b_2 & c_2 & & \\ \ddots & \ddots & \ddots & & \\ & a_{J-2} & b_{J-2} & c_{J-2} & \\ 0 & & a_{J-1} & b_{J-1} & \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{J-2}^{n+1} \\ u_{J-1}^{n+1} \end{pmatrix} = \begin{pmatrix} u_0^* \\ r_1 - a_1 u_0^* \\ r_2 \\ \vdots \\ r_{J-2} \\ r_{J-1} \end{pmatrix} \quad (2.61)$$

Une manœuvre semblable au point de maille x_{J-1} doit être effectuée pour imposer la seconde condition limite.

Même si notre nouvel algorithme est implicite, l'analyse de von Newmann s'applique comme auparavant. La substitution de (2.31) dans (2.55) conduit à

$$\xi = \left(1 + 4 \frac{D\Delta t}{(\Delta x)^2} \sin^2(k\Delta x/2) \right)^{-1}; \quad (2.62)$$

ce qui implique que $|\xi| < 1$ pour n'importe quelle combinaison de Δt et Δx ! Alleluia Banzai et toute cette sorte de chose !

L'algorithme demeure stable même si le critère de Courant n'est pas satisfait, parce que le processus de solution du système algébrique couple tous les noeuds; effectivement, l'information de *tous* les noeuds spatiaux au pas n est utilisée dans le calcul de la solution à chaque point de maille au pas $n + 1$.

Il ne faut cependant pas perdre de vue que même si cet algorithme implicite est inconditionnellement stable, l'utilisation de pas temporel et/ou spatiaux trop grands va nous produire une solution imprécise; la stabilité numérique est une condition nécessaire, mais non suffisante, à la précision.

2.4.2 La méthode de Crank-Nicolson

L'algorithme implicite (2.55) fonctionne et est stable, mais s'avère surdissipatif, un peu comme la méthode de Lax. Une version implicite moins dissipative peut être produite en évaluant le membre de droite comme la moyenne du terme aux pas n et $n + 1$, de manière à le centrer temporellement (un peu comme dans Lax-Wendroff). On écrit donc:

$$u_j^{n+1} = u_j^n + \left(\frac{D\Delta t}{(\Delta x)^2} \right) \frac{1}{2} [(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}) + (u_{j+1}^n - 2u_j^n + u_{j-1}^n)] \quad (2.63)$$

C'est l'algorithme de Crank-Nicolson, qui conduit maintenant au système matriciel:

$$[K_{ij}]\{u_j^{n+1}\} = \underbrace{\{u_i^n\} - (2[\delta_{ij}] - [K_{ij}])\{u_j^n\}}_{\equiv\{r_i^n\}}. \quad (2.64)$$

avec cette fois les éléments diagonaux de la matrice $[K_{ij}]$ donnés par $a_i = c_i = -D\Delta t/2(\Delta x)^2$, $b_i = 1 + D\Delta t/(\Delta x)^2$, et $[\delta_{ij}]$ la matrice-identité. Le membre de droite $\{r_i\}$ est encore un vecteur de quantités connues, mais cette fois il implique à chaque pas de temps un calcul plus

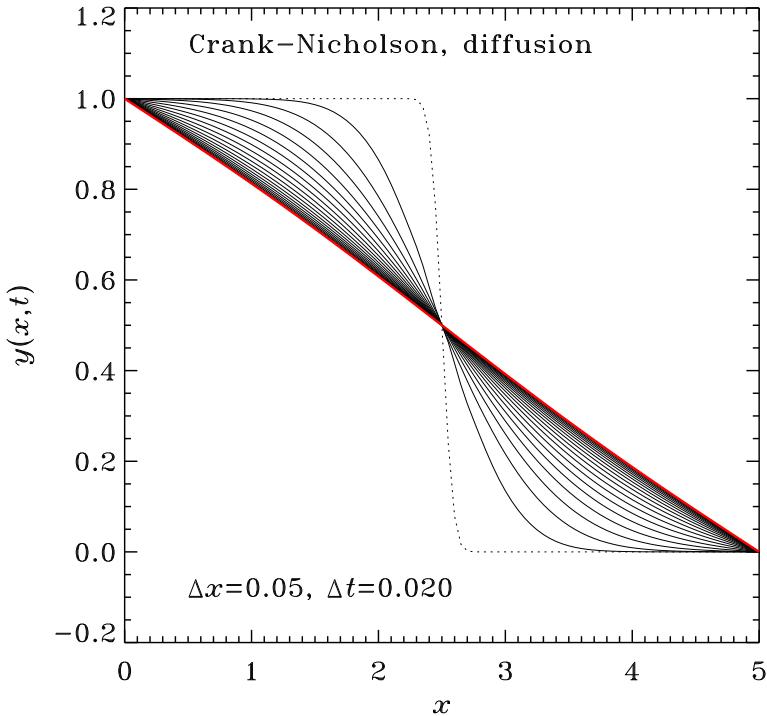


Figure 2.11: Solution numérique au problème de diffusion de la §2.1.2, cette fois à l'aide de la méthode implicite de Crank-Nicolson. Notez le pas de temps 20 fois plus grand que pour solution FTCS de la Fig. 2.1.

élaboré, soit la multiplication de la matrice $[K]$ au vecteur-solution du pas de temps précédent. La Figure 2.11 montre une solution Crank-Nicolson à notre problème diffusif prototypique. La solution est indistinguables de celle obtenue par la méthode FTCS (cf. Fig. 2.1); pourquoi alors s'estre tapé la solution matricielle d'un système algébrique ? Parce que le pas de temps est ici 20 fois plus grand !

La méthode- Θ est une généralisation de l'approche implicite introduite ci-dessus. Elle consiste à évaluer le membre de droite de notre EDP comme une moyenne pondérée aux pas de temps n et $n + 1$, la pondération étant contrôlée par le facteur numérique $0 \leq \Theta \leq 1$. On se retrouve avec un système algébrique semblable à celui obtenu pour Crank-Nicolson (viz. l'éq. (2.63)), sauf que la pondération des deux groupes de termes dans la parenthèse carrée au membre de droite est maintenant Θ et $(1 - \Theta)$ pour les contributions aux pas $n + 1$ et n respectivement, plutôt qu'une pondération égale (à $1/2$). La limite $\Theta = 0$ nous ramène directement à FTCS, $\Theta = 1$ correspond à l'algorithme Euler implicite (l'éq. (2.55)) et poser $\Theta = 1/2$ nous ramène évidemment à Crank-Nicolson. Voir les références données en fin de chapitre pour plus de détail.

2.5 Quel algorithme utiliser ?

On en a fait du chemin n'est-ce-pas de la Figs. 2.1 à 2.11 ? Que doit-on conclure de cet étourdissante tournée des algorithmes numériques pour EDPs? Essayons de condenser tout ça en quelques “recommandations”:

1. Pour des problèmes avec des temps caractéristiques bien définis et où la diffusion est peu

importante, une méthode explicite est habituellement préférable.

2. Parmi les méthodes explicites, Lax-Wendroff performe habituellement très bien, même en présence de fort gradients dans la solution.
3. Si la diffusion est importante et/ou il existe une grande disparité de temps caractéristiques, une méthode implicite est habituellement préférable; mais attention, prendre un trop grand pas de temps peut conduire à une solution physiquement imprécise, même si elle est stable!
4. Parmi les méthodes implicites, Crank-Nicolson est recommandable, sauf pour des problèmes caractérisés par de forts gradients, dans lequel cas la méthode- Θ avec $\Theta = 2/3$ est souvent préférable.
5. Les termes sources nonlinéaires de type réactif (i.e., n'impliquant pas de dérivées spatiales) peuvent être traités de manière explicite, mais en s'assurant bien que le pas de temps demeure confortablement inférieur au temps caractéristique de réaction.

Notons finalement que pour la majorité des “vrais” problèmes, particulièrement en plus d’une dimension spatiale, l’erreur de discréétisation spatiale est dominante; donc pas besoin de se casser la tête pour coder une discréétisation temporelle d’ordre élevé. Des méthodes d’ordre deux temporellement, comme Lax-Wendroff ou Crank-Nicolson, sont habituellement appropriées.

2.6 Au delà du 1D

Toutes les méthodes discutées à date l’ont été dans le cas de problèmes en une dimension spatiale, mais toutes peuvent se généraliser à deux ou trois dimensions; l’effort requis peut cependant varier...

Considérons une fonction $u(t, x, y)$ de deux variables spatiales, discréétisée sur une maille cartésienne régulière en deux dimensions spatiales (voir Figure 2.12):

$$x \rightarrow \{x_0, x_1, \dots, x_{N-1}\}, \quad x_{j+1} > x_j, \quad (2.65)$$

$$y \rightarrow \{y_0, y_1, \dots, y_{M-1}\}, \quad y_{k+1} > y_k. \quad (2.66)$$

Une maille est dite **cartésienne** si toutes les lignes $x = \text{constante}$ sont des droites parallèles les unes aux autres, les $y = \text{constante}$ le sont également, et chaque famille de droites est orthogonale à l’autre. On identifie chaque noeud du maillage 2D par une paire d’indices (j, k) , et on écrit pour la fonction discréétisée:

$$u_{j,k}^n \equiv u(t_n, x_j, y_k), \quad (2.67)$$

Toutes les méthodes explicites introduites aux sections 2.1 et 2.3 se généralisent facilement à un maillage 2D. Par exemple, dans le cas de la méthode FTCS appliquée à l’équation de diffusion (2.3), on aurait:

$$u_{j,k}^{n+1} = u_{j,k}^n + \left(\frac{D\Delta t}{(\Delta x)^2} \right) (u_{j,k+1}^n - 2u_{j,k}^n + u_{j,k-1}^n) + \left(\frac{D\Delta t}{(\Delta y)^2} \right) (u_{j+1,k}^n - 2u_{j,k}^n + u_{j-1,k}^n) \quad (2.68)$$

ou encore, pour un maille où $\Delta x = \Delta y \equiv \Delta$:

$$u_{j,k}^{n+1} = u_{j,k}^n + \left(\frac{D\Delta t}{\Delta^2} \right) (u_{j,k+1}^n + u_{j+1,k}^n - 4u_{j,k}^n + u_{j,k-1}^n + u_{j-1,k}^n); \quad (2.69)$$

tandis que pour FTCS appliqué à l’équation d’advection pure (2.26) on aurait:

$$u_{j,k}^{n+1} = u_{j,k}^n - \Delta t \left(v_x \frac{u_{j,k+1}^n - u_{j,k-1}^n}{2\Delta x} + v_y \frac{u_{j+1,k}^n - u_{j-1,k}^n}{2\Delta y} \right). \quad (2.70)$$

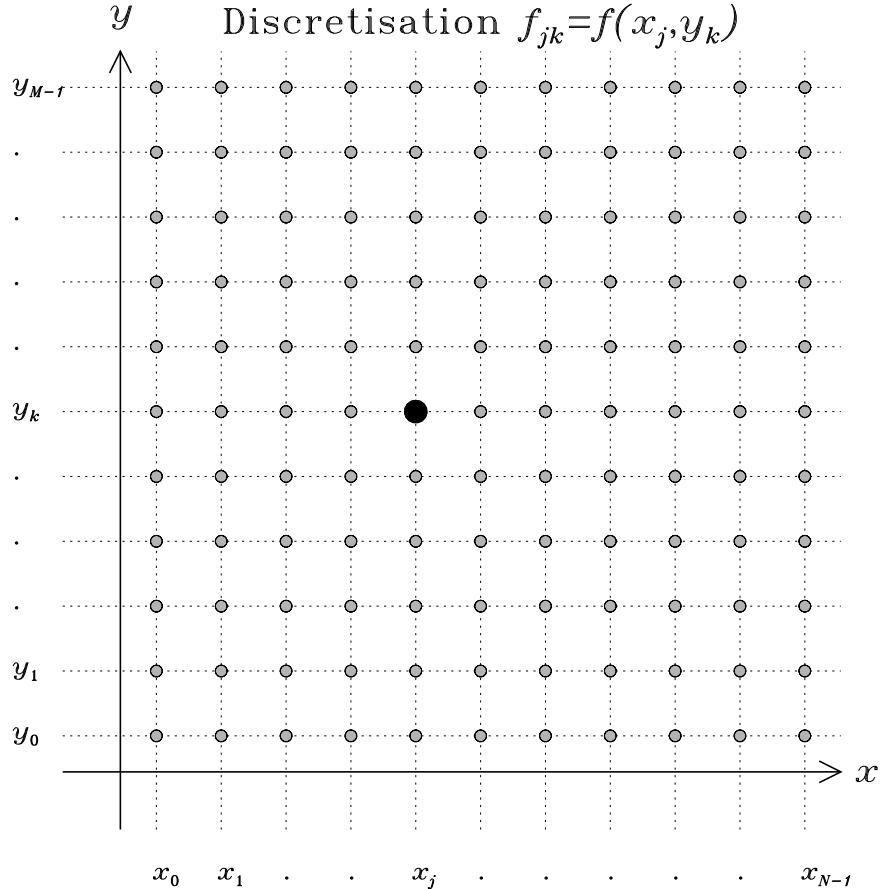


Figure 2.12: Discréétisation d'une fonction de deux variables $u(x, y)$ sur un maillage Cartésien régulier, ici équidistant dans les direction x et y et avec $N = M = 11$. Le noeud (j, k) est indiqué en noir. Je vous laisse imaginer comment ceci se généralise à des fonctions de plus de deux variables.

Attention, dans le cas des méthodes de Lax (§2.3.3) et Lax-Wendroff (§2.3.4), il faut bien s'assurer d'évaluer les u^n et les flux (respectivement) au centre des cellules, i.e. à $x_k \pm \Delta x/2$ et $y_j \pm \Delta y/2$.

D'un point de vue pratique les choses se corsent passablement plus quand on passe à la généralisation multidimensionnelle des méthodes implicites. Quelle que soit la dimensionnalité du problème, les discréétisations spatiale et temporelle produisent un système matriciel du type donné par (2.56), qui dans le cas de la maille de la Fig. 2.12 impliquera $S = M \times N$ valeurs de $u_{j,k}$, et une matrice K de taille $S \times S$. Supposons que l'on choisit d'ordonner les valeurs du vecteur $\{u\}$ de manière séquentielle en x , soit la ligne de N noeud pour j fixé à $j = 0$ pour $y = y_0$, suivi de la ligne de N noeuds pour $y = y_1$, etc. Une discréétisation spatiale telle qu'effectuée aux membres de droite de (2.68) ou (2.70) ci-dessus couple trois noeuds contigus dans le vecteur $\{u\}$, soit $u_{j,k-1}, u_{j,k}, u_{j,k+1}$, et deux autres qui ne le sont pas: $u_{j-1,k}, u_{j+1,k}$; en conséquence de quoi la matrice K ne sera plus tridiagonale, mais adoptera une forme en bande, ayant la forme de trois diagonales comme auparavant (éq. 2.57), et deux autres diagonales séparées par $N - 1$ zéros horizontalement et verticalement par rapport au groupe de trois diagonales centrales. La largeur de bande, ici $N + 1$, est déterminés par le nombre de point de mailles dans la direction

primaire de numérotation globale, ici N puisqu'on numérote séquentiellement en x . Dans le cas de la méthode d'Euler implicite, la matrice a donc la forme suivante:

$$[K_{i,j}] = \begin{pmatrix} b_0 & c_0 & 0 & \dots & 0 & d_0 & 0 & \dots & 0 \\ a_1 & b_2 & c_1 & 0 & & 0 & d_1 & 0 & \vdots \\ 0 & a_2 & b_2 & c_2 & 0 & & 0 & d_2 & 0 \\ \vdots & & \ddots & \ddots & \ddots & & & \ddots & \\ 0 & & 0 & a_{N-1} & b_{N-1} & c_{N-1} & 0 & & \\ e_N & 0 & & 0 & a_N & b_N & c_N & 0 & \\ 0 & e_{N+1} & 0 & & 0 & a_{N+1} & b_{N+1} & c_{N+1} & 0 \\ & & & & & & \ddots & \ddots & \ddots \\ \vdots & & 0 & e_{S-2} & 0 & & 0 & a_{S-2} & b_{S-2} & c_{S-2} \\ 0 & \dots & & 0 & e_{S-1} & 0 & \dots & 0 & a_{S-1} & b_{S-1} \end{pmatrix} \quad (2.71)$$

avec, pour notre problème de diffusion pure à D constant dans l'espace:

$$a_i = -D\Delta t/(\Delta x)^2, \quad i = 1, \dots, S-1, \quad (2.72)$$

$$b_i = 1 + 2D\Delta t/(\Delta x)^2, \quad i = 0, \dots, S-1, \quad (2.73)$$

$$c_i = -D\Delta t/(\Delta x)^2, \quad i = 0, \dots, S-2, \quad (2.74)$$

$$d_i = -D\Delta t/(\Delta x)^2, \quad i = 0, \dots, S-N-1, \quad (2.75)$$

$$e_i = -D\Delta t/(\Delta x)^2, \quad i = N, \dots, S-1. \quad (2.76)$$

tandis que pour une équation d'advection pure du genre

$$\frac{\partial u}{\partial t} = -v_x \frac{\partial u}{\partial x} - v_y \frac{\partial u}{\partial y}, \quad (2.77)$$

on aurait:

$$a_i = -v_x \Delta t / \Delta x, \quad i = 1, \dots, S-1, \quad (2.78)$$

$$b_i = 1, \quad i = 0, \dots, S-1, \quad (2.79)$$

$$c_i = v_x \Delta t / \Delta x, \quad i = 0, \dots, S-2, \quad (2.80)$$

$$d_i = v_y \Delta t / \Delta x, \quad i = 0, \dots, S-N-1, \quad (2.81)$$

$$e_i = -v_y \Delta t / \Delta x, \quad i = N, \dots, S-1. \quad (2.82)$$

où en général les composantes- x et y de la vitesse, v_x et v_y , peuvent dépendre de x et y . La solution de ce système se fait par décomposition LU; voir le chapitre 2 de l'ouvrage de Press et al. cité en bibliographie en fin de chapitre. L'application des conditions limites doit se faire encore une fois en modifiant directement la matrice et son vecteur coté droit, exactement comme discuté à la §2.4, sauf que cette fois la manoeuvre s'applique à tous les noeuds sur les frontières du domaine.

Si cette perspective de jongler ainsi avec des grosses matrices vous perturbe à outrance, vous avez aussi la possibilité d'adopter la méthode dite *alternate directions implicit* (ADI), également discutée dans Press et al. (§19.3). Ceci consiste à traiter séparément les deux directions spatiales, en solutionnant séquentiellement pour un pas $\Delta t/2$. Dans le cas de l'équation de diffusion 2D:

$$u_{j,k}^{n+1} = u_{j,k}^n + \frac{1}{2} \left(\frac{D\Delta t}{(\Delta x)^2} \right) \underbrace{(u_{j,k+1}^{n+1} - 2u_{j,k}^{n+1} + u_{j,k-1}^{n+1})}_{\text{implicite}}$$

$$+ \frac{1}{2} \left(\frac{D\Delta t}{(\Delta y)^2} \right) \underbrace{(u_{j+1,k}^n - 2u_{j,k}^n + u_{j-1,k}^n)}_{\text{explicite}}, \quad j = 0, \dots, M-1, \quad (2.83)$$

et ensuite pour le $\Delta t/2$ suivant:

$$\begin{aligned} u_{j,k}^{n+1} &= u_{j,k}^n + \frac{1}{2} \left(\frac{D\Delta t}{(\Delta x)^2} \right) \underbrace{(u_{j,k+1}^n - 2u_{j,k}^n + u_{j,k-1}^n)}_{\text{explicite}} \\ &\quad + \frac{1}{2} \left(\frac{D\Delta t}{(\Delta y)^2} \right) \underbrace{(u_{j+1,k}^{n+1} - 2u_{j,k}^{n+1} + u_{j-1,k}^{n+1})}_{\text{implicite}} , \quad k = 0, \dots, N-1 . \end{aligned} \quad (2.84)$$

L'idée est que chacun de ces problème est décrit par un système matriciel tridiagonal du genre de (2.57). Autrement dit, il faut solutionner $2 \times N \times M$ problèmes 1D à chaque pas de temps; c'est plus facile à coder, plus rapide à l'exécution, mais moins précis; *No Free Lunch!*

2.7 Techniques de discrétisation spatiale

Prenons un peu de recul sur tout ça et considérons une EDP linéaire avec coefficients indépendants du temps. On peut écrire ceci sous la forme:

$$\frac{\partial u}{\partial t} = \mathcal{L}u + s , \quad (2.85)$$

où \mathcal{L} est un opérateur linéaire qui peut combiner des dérivés première, seconde, etc., ainsi que des coefficients dépendant de la position. Le processus de discrétisation spatiale, quel qu'il soit, transforme cette EDP en un système d'EDOs linéaires de la forme:

$$[C_{ij}] \frac{\partial \{u_j\}}{\partial t} = [L_{ij}] \{u_j\} + \{s_i\} , \quad i, j = 0, \dots, J-1 . \quad (2.86)$$

L'apparition de la matrice $[C]$ au membre de gauche peut paraître surprenante, mais on y reviendra sous peu. L'introduction d'une discrétisation temporelle convertit finalement cet ensemble d'EDO en un système algébrique linéaire qui doit être solutionné à chaque pas de temps:

$$[K_{ij}]^n \{u_j^{n+1}\} = \{F_i^n\} , \quad i, j = 0, \dots, N-1 , \quad (2.87)$$

sauf si la discrétisation temporelle choisie est explicite, dans lequel cas la matrice $[K]$ se réduit à la matrice identité et il ne s'agit plus que de calculer le membre de droite $\{F_i^n\}$ à chaque pas de temps.

2.7.1 Différences finies

Tous les algorithmes considérés jusqu'ici dans ce chapitre, qu'ils soient explicites ou implicites, effectuent la discrétisation spatiale via des formules de différences finies centrées d'ordre deux. En une dimension spatiale, ceci conduit typiquement à une matrice $[L]$ qui est tridiagonale. L'utilisation de différences finies centrée d'ordre plus élevé, par exemple d'ordre 4, conduirait à un système pentadiagonal. Dans un cas comme dans l'autre la matrice $[C]$ est la matrice identité, i.e., $C_{ij} = \delta_{ij}$.

Dans tous les cas, l'erreur de discrétisation spatiale est dominée par l'erreur de troncation associée aux formules de différence finies choisies. L'erreur de discrétisation spatiale est donc déterminée par la précision à laquelle les dérivées sont évaluées par ces différences finies.

2.7.2 Éléments finis

Une autre approche à la discréétisation spatiale, les *éléments finis*, consiste à subdiviser le domaines en sous-domaines contigus, et dans chacun de ces sous-domaines on représente la solution par une interpolation polynomiale du genre:

$$u_{(e)}(x, t) = \sum_i a_i(t) \varphi_i(x), \quad (2.88)$$

où les φ sont des polynômes de forme analytique connue (habituellement des polynômes d'interpolation de Lagrange). Ici toute la dépendance temporelle est contenu dans les coefficients numériques a définissant l'interpolation. La substitution de l'éq. (2.88) dans l'EDP conduit de nouveau à un système d'EDO de la forme (2.86), où les u ne sont plus la valeur de la solution aux points de maille x_j , comme dans le cas des différences finies, mais plutôt les coefficients a définissant l'interpolation. La taille des matrices $[L]$ et $[C]$, et du vecteur solution $\{u_j\}$, se retrouve déterminée à la fois par le nombre d'éléments utilisés pour couvrir le domaine spatial, et par l'ordre de l'interpolation choisi pour chaque élément.

Si on utilise une interpolation linéaire, alors la matrice $[L]$ se retrouve encore une fois tridiagonale, mais cette fois la matrice $[C]$ est également tridiagonale. La précision de la discréétisation spatiale est maintenant déterminée par le degré auquel l'interpolation représente bien la solution dans l'élément; puisque les fonctions d'interpolation ont une forme analytique connue, les dérivées spatiales sont évalués exactement!

Pour en savoir plus sur la méthode des éléments finis, voir le bouquin de Burnett cité en bibliographie à la fin de ce chapitre.

2.7.3 Méthodes spectrales

Les méthodes spectrales représentent également la solution via une forme d'interpolation, mais utilisant cette fois des fonctions ou polynômes qui sont définis globalement sur le domaine spatial. Dans le cas d'un problème 1D, une série de Fourier est le développement habituellement choisi. La substitution de la série de Fourier dans l'EDP conduit de nouveau à un système d'EDO pour les coefficients des termes de Fourier. Comme avec les éléments finis, les dérivés spatiales sont évalués analytiquement, et l'erreur de discréétisation spatiale est déterminé par le degré auquel la série de Fourier parvient à bien représenter la fonction sur tout le domaine.

La taille des matrices $[L]$ et $[C]$, et du vecteur solution $\{u_j\}$ est maintenant déterminée par le nombre de termes conservés dans la série de Fourier; plus la solution développe de fort gradients, plus on doit conserver de termes pour avoir une bonne précision. Ici les matrices $[C]$ et $[K]$ sont pleines, ce qui alourdit substantiellement le processus de solution du système (2.87), mais pour une solution relativement lisse les méthodes spectrales permettent d'obtenir des solutions précises tout en conservant relativement peu de termes de Fourier, donc ces matrices se retrouvent souvent de beaucoup plus petite taille que celles générées par une discréétisation par différences finies ou éléments finis ayant une précision comparable.

Pour en savoir plus sur les méthodes spectrales, voir le bouquin de Canuto et al. cité en bibliographie à la fin de ce chapitre.

2.7.4 Méthodes de volumes finis

Dans le cas d'EDP pouvant s'exprimer comme un système d'équation de conservation (viz. eq. 2.5), les méthodes dites de volumes finis sont souvent avantageuses car elles peuvent être formulées de manière à conserver la quantité transportée à un haut niveau de précision numérique, même dans des situations où cette variable présente des discontinuités (e.g., un choc hydrodynamique).

Considérons une cellule d'un maillage 3D, et intégrons l'éq. 2.5 sur le volume de la cellule:

$$\int_V \frac{\partial u}{\partial t} dV = - \int_V \nabla \cdot \mathbf{F} dV, \quad (2.89)$$

Commutant le opérateurs au membre de gauche et appliquant le théorème de la divergence au membre de droite, ceci devient:

$$\frac{\partial}{\partial t} \int_V u dV = - \oint_S \mathbf{F} \cdot \hat{\mathbf{n}} dS . \quad (2.90)$$

La forme plus simple d'une méthode de volume fini consiste à évaluer u au membre de gauche au centre de la cellule, et d'introduire des formules de quadratures —équivalent des différences finies pour le calcul des intégrales— pour évaluer les six contributions à l'intégrale de surface au membre de droite, sujet à une contrainte de continuité sur le flux $\mathbf{F} \cdot \hat{\mathbf{n}}$ entre cellules contigues. L'introduction subséquente d'une différence finie pour la dérivée temporelle nous ramène encore une fois à un système d'équations algébriques pour les u évalués au centre des cellules. Sous cette formulation intégrale il n'y a plus de dérivées à évaluer, mais la précision de la discréétiation est déterminée par celle du calcul des intégrales de surface; cependant, la conservation du flux à travers les surfaces délimitant deux cellules contigues est assurée par construction.

Pour en savoir plus sur les méthodes de volumes finis, voir le bouquin de LeVeque cité en bibliographie à la fin de ce chapitre.

2.8 Les plasmas en tokamak

2.8.1 La fusion nucléaire

La fusion nucléaire de quatre atomes d'Hydrogène en un atome d'Hélium libère un substantiel 27.7 Mev d'énergie, soit 6.7 Mev par nucléon, en raison de la très grande énergie de liaison du noyau ${}^4\text{He}$, elle-même due au fait que deux protons et deux neutrons remplissent complètement les premiers niveaux quantiques nucléaires protoniques et neutroniques. En comparaison, la fission d'un noyau de ${}^{235}\text{U}$ par bombardement neutronique, bien qu'elle libère ~ 200 Mev, ne produit que $\simeq 0.9$ Mev par nucléon. La fusion nucléaire est donc énergétiquement préférable autant pour les applications explosives que contrôlées. En particulier, la fusion de l'Hydrogène est souvent considérée comme la solution-miracle à tous nos problèmes énergétiques: de l'Hydrogène il y en plein les océans, et en bonus, son "déchet", le ${}^4\text{He}$ est un gaz non-radioactif et chimiquement inerte, qui s'évapore naturellement de l'atmosphère terrestre. Les déchets et sous-produits de la fission nucléaire, par contre, sont plus difficiles et dangereux à gérer; on n'a qu'à penser à la demi-vie de 2.41×10^4 années du Plutonium-239...

La fusion thermonucléaire de l'hydrogène en Hélium est le processus produisant la luminosité de la majorité des étoiles observables. Dans le cœur du soleil ce brûlage de l'Hydrogène se produit via une séquence de réactions nucléaires appelée *chaîne pp*:



où $D \equiv {}^2\text{H}$ est le Deutérium. L'énergie totale libérée est de 24.7 MeV, un bilan auquel doit s'ajouter 2×1.02 MeV résultant de l'annihilation quasi-instantané du positron produit par la première réaction avec un électron libre. Tout ça conduit à une production énergétique d'environ 276.5 W m^{-3} au centre du cœur solaire, et à une puissance totale de $3.86 \times 10^{26} \text{ W}$ pour l'ensemble du cœur. L'énergie des neutrinos, elle, est perdue car la très grande majorité de ceux-ci sortent de l'étoile sans interagir avec quoique ce soit.

La chaîne ci-dessus domine aux conditions physiques du cœur solaire; cependant, des variantes de la troisième réaction impliquant ${}^3\text{He} + {}^4\text{He}$ peuvent devenir importantes à plus hautes températures.

Le principal obstacle à ces réactions est la force de répulsion Coulombienne, particulièrement difficile à contrer chez les noyaux légers dont le rapport p/n avoisine l'unité. Les étoiles contournent ce problème en brûlant l'Hydrogène en leurs cœur, où la température et la densité

sont très élevées — $\rho \simeq 150 \text{ g cm}^{-3}$ et $T \simeq 1.5 \times 10^7 \text{ K}$ au centre du soleil— favorisant ainsi des collisions suffisamment fréquentes et énergétiques pour vaincre la barrière de Coulomb.

Plusieurs des grands projets tokamak actuellement en développement sont basés sur la réaction de fusion Deutérium-Tritium:



où la présence de neutrons favorise la réaction de fusion, car ces derniers contribuent à la force nucléaire (attractive et de courte portée) mais pas à la force électrostatique (répulsive et de longue portée, $\propto 1/r^2$). C'est tout de même 3.52 MeV par nucléon, soit un petit peu plus de la moitié du 6.7 MeV par nucléon de la chaîne pp complète. Une idée particulièrement intéressante à l'étude est de "récupérer" le neutron produit, et de s'en servir pour produire du Tritium à partir du Deutérium, ce dernier existant en (relativement) grande quantité dans l'eau naturelle (rapport $D/H \simeq 1.56 \times 10^{-4}$); ça peut paraître petit, mais les usines de production de Deuterium pour les réacteurs nucléaires CANDU peuvent produire de l'ordre de 1000 tonnes métriques d'eau lourde (D_2O) par année, ce qui serait largement suffisant pour les besoins en Deutérium de la fusion en tokamak.

2.8.2 Trajectoires de particules chargées dans un champ magnétique

Reproduire des températures "solaires" dans un réacteur pose un problème immédiat: aucun métal connu n'y résisterait. Techniquelement, ce problème est complexe: il s'agit (1) de produire la réaction de fusion, (2) tout en confinant le plasma de manière à contrôler la production d'énergie, (3) tout en pouvant extraire rapidement la chaleur du système, pour ultimement propulser des turbines, par exemple. Seul le premier de ces problèmes est pertinent aux applications plus explosives de la fusion thermonucléaire... Et, toujours du côté pratique, il faudrait bien produire plus d'énergie que la quantité requise pour chauffer (et confiner) le plasma aux températures et densités requises.

Arrivent à la rescoussse les champs magnétiques. Vous savez qu'une particule de charge électrique q et masse m se déplaçant à vitesse \mathbf{v} dans un champ magnétique \mathbf{B} ressent une force de Lorentz $\propto q\mathbf{v} \times \mathbf{B}$ qui tendra à contraindre la particule à un mouvement en spirale le long d'une ligne de champ magnétique. Pour un champ magnétique spatiallement uniforme et d'intensité B_0 , le mouvement dans le plan perpendiculaire à \mathbf{B} prend la forme d'un cercle au *rayon de Larmor* (r_L):

$$r_L = \frac{v_\perp}{qB_0}, \quad (2.95)$$

où v_\perp est la grandeur de la projection de \mathbf{v} dans ce plan, et la fréquence angulaire de rotation de la particule dans ce plan est donnée par la *fréquence cyclotron*:

$$\omega_c = \frac{qB_0}{m} \quad (2.96)$$

qui elle, on le notera, est indépendante de la vitesse de v_\perp . Si on suppose $\mathbf{B} = B_0\hat{\mathbf{e}}_x$, alors les particules décrivent une trajectoire donnée par

$$y(t) = y_0 + r_L \sin(\omega_c t), \quad z(t) = z_0 + r_L \cos(\omega_c t), \quad (2.97)$$

où (y_0, z_0) est le *centre de gyration*. Comme la force magnétique n'a aucune composante dans la direction parallèle à \mathbf{B} , toute composante de vitesse parallèle à \mathbf{B} (v_\parallel), si présente, demeure constante; c'est comme ça qu'on peut se retrouver avec un mouvement en spirale, à moins évidemment que \mathbf{v} soit exactement perpendiculaire à \mathbf{B} .

En présence d'une force extérieure additionnelle \mathbf{F} (e.g. la gravité, ou un champ électrique), une dérive du centre de gyration apparaît. Ajoutons à la configuration traitée ci-dessus une

force pointant dans la direction- y : $\mathbf{F} = F\hat{\mathbf{e}}_y$; les équations du mouvement d'une particule deviennent:

$$\frac{\partial^2 z}{\partial t^2} = \omega_c \frac{\partial y}{\partial t}, \quad (2.98)$$

$$\frac{\partial^2 y}{\partial t^2} = -\omega_c \frac{\partial z}{\partial t} + \frac{F}{qB_0}. \quad (2.99)$$

où on a utilisé la définition de la fréquence cyclotron (2.96). La solution de ces équations, si la particules est initialement positionnée à $(y, z) = 0$ avec vitesse nulle, est

$$z(t) = \frac{F}{q\omega_c B_0} (\omega_c t - \sin(\omega_c t)), \quad (2.100)$$

$$y(t) = \frac{F}{q\omega_c B_0} (1 - \cos(\omega_c t)), \quad (2.101)$$

La somme du carré de ces deux équations conduit à la trajectoire:

$$(z(t) - (F/qB_0)t)^2 + (y(t) - R)^2 = R^2, \quad (2.102)$$

où on a défini $R \equiv F/q\omega_c B$. Ceci décrit une trajectoire *cycloïde*, soit une trajectoire suivant un cercle dont le centre se déplace à une vitesse $v_G = F/qB_0$ dans la direction- z . Notez que cette vitesse de dérive du centre de gyration est ici perpendiculaire à la fois à \mathbf{B} et à la direction de la force extérieure! Dans une situation plus générale où la force \mathbf{F} est orientée arbitrairement par rapport à \mathbf{B} , la vitesse de dérive est donnée par:

$$\mathbf{v}_G = \frac{1}{q} \frac{\mathbf{F} \times \mathbf{B}}{B^2}; \quad (2.103)$$

donc, par exemple, en présence d'un champ électrique on aurait la fameuse "dérive $\mathbf{E} \times \mathbf{B}$ " si chère à nos collègues de plasma.

2.8.3 Le confinement plasma en tokamak

Imaginons maintenant un long solénoïde; vous avez déjà vu que loin des bords du solénoïde le champ intérieur est orienté parallèlement à l'axe du solénoïde et d'intensité uniforme $B = \mu_0 n I$, où μ_0 est la perméabilité du vide, I le courant circulant dans la bobine solénoidale, et n le nombre de tours par unité de longueur caractérisant cette bobine. Pour un courant électrique suffisamment intense, et donc un champ magnétique interne intense lui aussi, le rayon de Larmor se retrouvera beaucoup plus petit que le diamètre du solénoïde, et les constituants microscopiques d'un plasma chaud (fait d'atomes ionisés) ne pourront se déplacer globalement que dans la direction de l'axe du solénoïde; le plasma est ainsi confiné radialement... mais pas longitudinalement; pour confiner longitudinalement le plus simple est de plier le solénoïde pour lui donner la forme d'un tore circulaire. Une réacteur plasma confinant le plasma de cette façon est appelé *tokamak*, contraction de "chambre magnétique toroidale" en Russe.

Dans une telle configuration tokamak, le champ est orienté selon la direction "toroidale" $\hat{\mathbf{e}}_\phi$, mais son intensité décroît avec le rayon cylindrique mesuré à partir de l'axe de symétrie du tore:

$$\mathbf{B}(s) = \frac{\mu_0 n R}{s} \hat{\mathbf{e}}_\phi. \quad (2.104)$$

Or voilà un problème: en présence d'un gradient du champ magnétique (ici dans la direction $\hat{\mathbf{e}}_s$), Les trajectoires des charges dans le plan du tore ne seront plus circulaires, dérivant plutôt graduellement vers l'extérieur du tore. Revenons, en coordonnées cartésiennes, à notre particule chargée orbitant dans le plan $[y, z]$ dans un champ magnétique orienté en x mais dont l'intensité varie selon y . Si l'échelle spatiale de cette variation est beaucoup plus petite que le rayon de

Larmor, alors on peut utiliser les premier deux termes d'un développement en série de Taylor pour exprimer la variation en y du champ magnétique le long de l'orbite de la particule comme:

$$B_x(y, z, t) = B_0 \pm \underbrace{r_L \cos(\omega_c t)}_{\equiv \Delta y} \frac{\partial B_x}{\partial y}, \quad (2.105)$$

et donc la composante- y de la force magnétique sera:

$$F_y(y, z, t) = -qv_z(y, z, t)B_x(y) = -q\underbrace{v_\perp \cos(\omega_c t)}_{v_z} \left(B_0 \pm r_L \cos(\omega_c t) \frac{\partial B_x}{\partial y} \right). \quad (2.106)$$

Calculant la moyenne de ceci sur une orbite de la particule, on obtient:

$$\bar{F}_y = \mp \frac{qv_\perp r_L}{2} \frac{\partial B_x}{\partial y}, \quad (2.107)$$

où le facteur $1/2$ provient de la moyenne de $\cos^2(\omega_c t)$ sur une orbite, soit l'intervalle $[0, 2\pi]$. Précisons que le terme en $B_0 \cos(\omega_c t)$ au membre de droite de l'éq. (2.106), pour sa part, se moyenne à zéro sur une orbite complète. En vertu de l'éq. (2.103), on aura donc ici:

$$\mathbf{v}_\nabla = \frac{1}{q} \frac{\bar{F}_y}{|B|} \hat{\mathbf{e}}_z = \mp \frac{v_\perp r_L}{2B} \frac{\partial B_x}{\partial y} \hat{\mathbf{e}}_z, \quad (2.108)$$

puisque $\mathbf{F} = F_y \hat{\mathbf{e}}_y$ est ici perpendiculaire à $\mathbf{B} = B_x(y) \hat{\mathbf{e}}_x$; ou encore, pour une orientation arbitraire du champ magnétique et de son gradient:

$$\mathbf{v}_\nabla = \pm \frac{v_\perp r_L}{2} \frac{\mathbf{B} \times \nabla B}{B^2}. \quad (2.109)$$

Dans le contexte tokamak, ∇B pointe dans la direction s , et \mathbf{B} dans la direction ϕ , et donc cette vitesse de dérive est dans la direction verticale z (toujours en coordonnées cylindriques).

À ce problème s'en ajoute un second, soit que le mouvement global de la particule autour de l'axe de symétrie du tore produira une force centrifuge pointant dans une direction perpendiculaire à \mathbf{B} , soit en $+\hat{\mathbf{e}}_s$, en tout point de la gyration de la particule autour de la ligne de champ. Ceci entraînera également une dérive verticale:

$$\mathbf{v}_R = \pm \frac{mv_\parallel}{qB^2} \frac{\mathbf{s} \times \mathbf{B}}{s^2}, \quad (2.110)$$

qui s'additionne à la dérive verticale associée au gradient de B , telle que donnée par l'éq. (2.109). Les constituants microscopiques du plasma aura donc tendance à vouloir s'accumuler dans le haut (charges positives) ou le bas (charge négatives) du tore, ce qui nuit clairement au confinement du plasma au centre du tore!

Ces deux problèmes peuvent être contournés en introduisant une seconde série de courants électriques circulant cette fois dans la direction toroidale, de manière à produire une composante magnétique dans le plan du tore, prenant la forme de ligne de champ fermées et approximativement circulaires. Le champ magnétique total est maintenant donné par la somme vectorielle du champ toroidal original et de ce nouveau champ poloidal (le principe de superposition!), ce qui conduit à une configuration torsadée, comme illustré schématiquement sur la Figure 2.13. Cette configuration torsadée a comme conséquence que dans leur parcours le long du tore, les particules n'orbitent plus à une position fixe dans la section, mais décrivent maintenant une trajectoire fermée autour du centre de la section. Durant une révolution dans la section du tore, sous l'influence de la dérive verticale les particules passeront donc autant de temps à être poussées vers le centre de la section que vers l'extérieur, ce qui permettra ainsi de contrer la dérive verticale.

En pratique, comment produire une telle composante magnétique poloidale ? On peut utiliser le déplacement des charges le long du tore comme source de courant pour induire

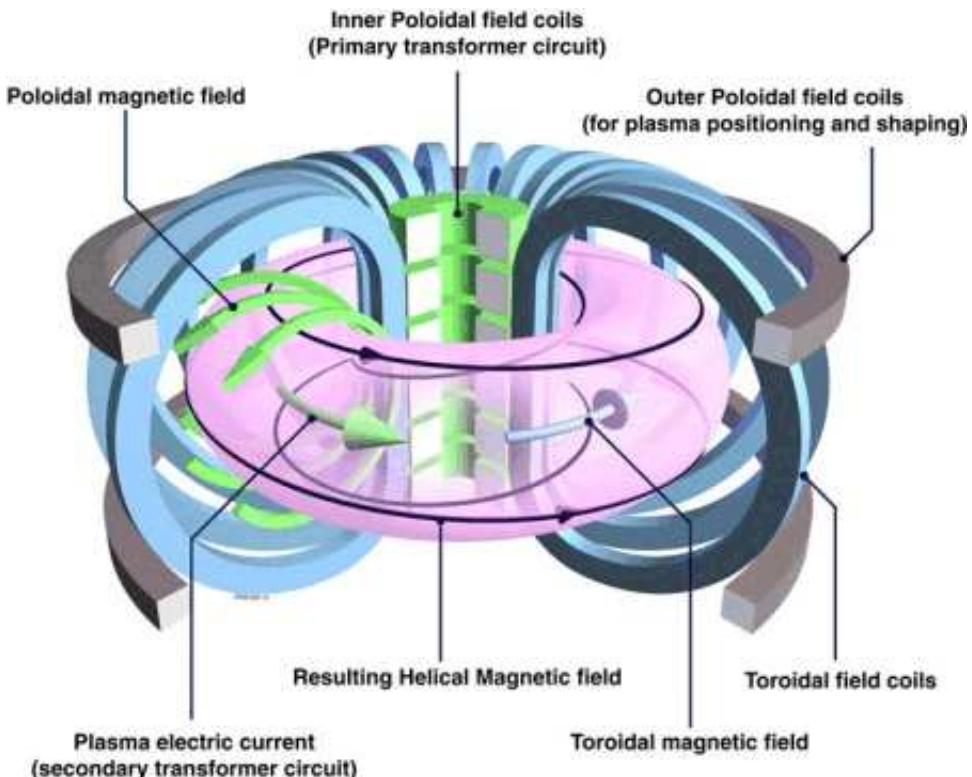


Figure 2.13: Confinement de plasma en tokamak. Les multiples systèmes de bobines, aimants et boucles de courant visent à confiner le plasma sur des trajectoires hélicoïdales à l'intérieur du tore. En rose, le plasma! Image tirée de http://www.ccf.ac.uk/How_fusion_works.aspx.

un champ poloidal; ce courant interne est typiquement soutenu par induction, une bobine étant placée le long de l'axe de symétrie du tore, et agissant comme le circuit primaire d'un transformateur. Des boucles de courants toroidaux extérieures au tore peuvent également être utilisées, afin de contrôler activement le confinement plasma (comme sur la Fig. 2.13).

La quantité de plasma requise pour soutenir la fusion dans la génération de tokamak présentement en développement se mesure en grammes, ce qui est minuscule. Cependant, dans le régime physique où se produisent les réactions de fusion la vitesse thermique des constituants de ce plasma est très élevée, ce qui place le plasma dans le régime collisionnel même si sa densité est très faible. Le plasma est également sujet à diverses instabilités qui produiront de la turbulence, comme le montrent à la fois les mesures directes ainsi que les simulations numériques dites gyrocinétiques, qui suivent l'évolution des distributions des positions et vitesses des particules circulant dans le tokamak. La Figure 2.14 montre un exemple d'une telle simulation, où l'échelle de couleur code la densité électronique. La structure fine est associée à la présence de tourbillons turbulents.

La turbulence peut devenir un gros problème dans un tokamak. Elle tendra à accélérer la dissipation du champ magnétique, et causera l'expansion du plasma et amplifiera le transfert de chaleur vers les parois du tokamak, réduisant ainsi la température centrale. Ajoutons à ça que si la fusion se produit, présumément là où le confinement est maximal, soit le long de l'axe du tore, le chauffage intense et localisé y étant associé va produire de très forts gradients de température et pression pointant aussi vers la périphérie du tore. Ce gradient doit être équilibré par quelque chose, si on veut espérer produire une configuration stationnaire (plutôt qu'explosive), où la fusion est *contrôlée* afin de pouvoir extraire la chaleur du système aussi rapidement qu'elle est produite. C'est cet aspect du problème que vous étudierez dans le cadre

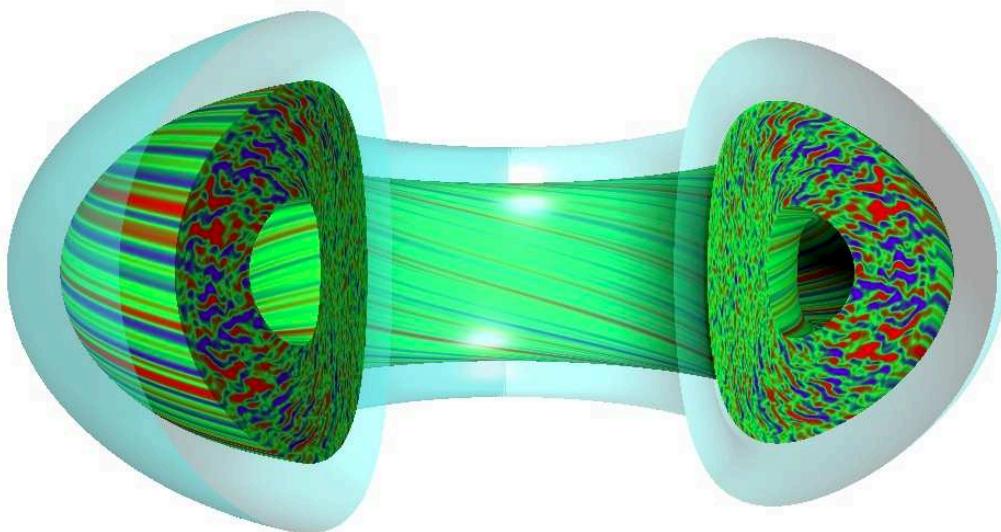


Figure 2.14: Simulation gyrocinétique d'un plasma en tokamak (code GYRO de General Atomics, É.-U.). Ce type de simulation calcule l'évolution temporelle de la densité des différents types de particules présentes (e^- , D , ^3H , etc) dans un espace 5-dimensionnel $[x, y, z, v_\perp, v_\parallel]$. Sur cette image la couleur code les fluctuations de la densité électronique, qui montre une élongation marquée le long des lignes du champ magnétique torsadé. Image tirée de <https://fusion.gat.com/theory>.

du projet qui clot ce chapitre !

2.9 Projet: barrière de confinement plasma

Le but du confinement est de maintenir le plus longtemps possible le plasma et la chaleur dans une région la plus compacte possible au centre de la section du tokamak, afin d'assurer les températures et densités les plus hautes possibles et ainsi favoriser la fusion.

Ce problème de confinement sera étudié dans une configuration géométrique simplifiée: on supposera invariance le long du tore (i.e., axisymétrie en coordonnées cylindriques, donc aucune dépendance en ϕ). On supposera de plus que la section du tokamak est circulaire, que la fusion ne se produit que très près du centre géométrique de cette section, et que les propriétés du plasma ne dépendent que de la distance r mesurée par rapport à ce centre. Le problème se réduit donc à une dimension spatiale.

2.9.1 Transport turbulent et diffusion nonlinéaire

Deux processus physiques contribuent au transport de la chaleur du centre de la section du tokamak vers ses bords: les collisions inter-particules (diffusion microscopique) et le transport associé aux mouvements turbulents dans le plan de la section (diffusion macroscopique); tant que l'échelle des tourbillons turbulents demeure beaucoup plus petite que le rayon de la section, ce transport peut se modéliser comme un processus diffusif où l'on suppose que le flux de chaleur est proportionnel au gradient de température:

$$\mathbf{Q} = -\chi \nabla T \quad (2.111)$$

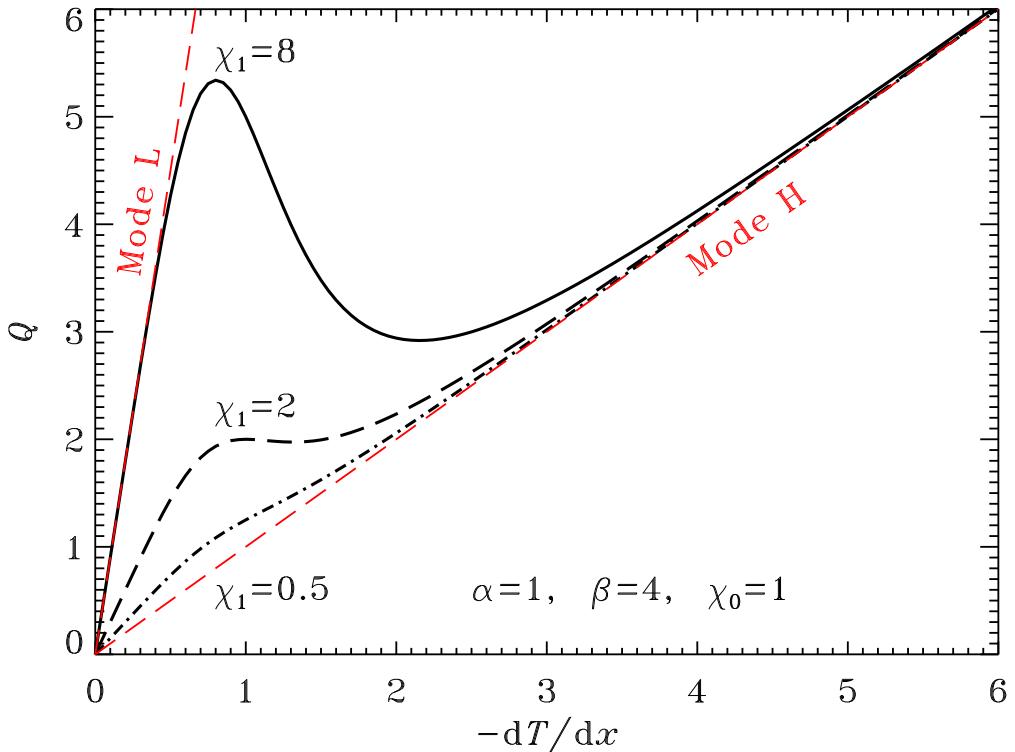


Figure 2.15: Variation du flux de chaleur en fonction du gradient de température dans le cas d'un transport turbulent en 1D, dont la conductivité nette donnée par l'éq. (2.112), avec les valeurs de paramètres $\chi_0 = 1$, $\alpha = 1$ et $\beta = 4$ dans l'éq. (2.112), et quelques valeurs distinctes de χ_1 , tel qu'indiqué. Si $\chi_1/\chi_0 \gg 1$, la relation devient non-monotone, avec deux branches de croissance du flux de chaleur selon dT/dx (voir texte).

avec le coefficient de conductivité thermique total donné par

$$\chi(T) = \chi_0 + \chi_1 \left[1 + \alpha \left(\frac{dT}{dx} \right)^\beta \right]^{-1}. \quad (2.112)$$

Ici la contribution χ_0 correspond à la diffusion microscopique (linéaire); et la contribution (nonlinéaire en T) χ_1 est celle associée au transport turbulent. En général, on s'attend à ce que $\chi_1 \gg \chi_0$. Le terme proportionnel à la puissance β du gradient de température entre les parenthèses carrées modélise la suppression de la turbulence par le cisaillement des écoulements de plasma dans la section du tore, ceux-ci étant propulsé par une force magnétique produite en réaction au gradient de pression induit par la présence d'un gradient de température. Il y a beaucoup de physique aussi compliquée que mal comprise là-dedans, mais le modèle de turbulence sur lequel est basée l'éq. (2.112) demeure un genre de standard à ce stade des choses.

La Figure 2.15 montre la variation du flux de chaleur Q en fonction du gradient de température, tel que donné par l'éq. (2.112), avec les valeurs de paramètres $\alpha = 1$, $\beta = 4$ et $\chi_0 = 1$, et quelques valeurs représentatives de χ_1 . Le profil non-monotone produit dans le régime $\chi_1/\chi_0 \gtrsim 2$ est une conséquence directe de la réduction du transport turbulent lorsque $-dp/dx$ dépasse $\alpha^{-1} = 1$ ici. Initialement le flux de chaleur augmente très rapidement avec $-dT/dx$, selon une pente donnée par $\chi_0 + \chi_1$ ($= 9$ ici). Mais dès qu'on dépasse la valeur α^{-1} , la réduction de l'efficacité du transport turbulent est ici rapide en raison du choix $\beta = 4$ (viz. eq. (2.112)). Une fois

le transport turbulent complètement éliminé, il ne reste que la contribution linéaire au flux ($\propto \chi_0$), qui cause la montée beaucoup plus lente de Q au delà de $-\partial T/\partial x \gtrsim 3$.

2.9.2 Établissement de l'effet barrière

On a donc deux “branches” dans la relation Q vs dT/dx portée en graphique à la Fig. 2.15: sur la première (à gauche), le flux Q croît rapidement avec le gradient de température, conséquence du transport turbulent; on a donc ici un mauvais confinement de la chaleur au centre. C'est le soi-disant “mode L” (pour “Low confinement”). Sur la seconde branche (à droite), on confine beaucoup mieux, dans le sens qu'on y trouve un gradient de température beaucoup plus élevé pour un Q donné; c'est le “mode H”, pour “High confinement”. Si la fusion se déclenche au centre du tore, T y croîtra très rapidement, conduisant à un très fort gradient local de température. On voudrait donc, idéalement, s'assurer qu'une fois la fusion déclenchée, l'évolution thermique du système l'emmène à se stabiliser (dans le sens d'une solution d'équilibre) sur le mode H plutôt que L.

L'objectif est donc de comprendre sous quelles conditions, en présence d'une source de chaleur au centre du tore (soyons optimiste, on suppose que la fusion opère!), le système peut transiter “naturellement” vers le mode H, où la turbulence est réduite à zéro (ou pas loin) et le transport de chaleur n'implique plus que le terme linéaire $\propto \chi_0$.

On étudiera l'effet barrière en géométrie Cartésienne, un peu comme si on solutionnait le long d'un diamètre de la section du tore. Le problème est décrit par une équation de diffusion de la chaleur ayant la forme:

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left[\chi(T) \frac{\partial T}{\partial x} \right] + H(x) , \quad x \in [-10, 10] \quad (2.113)$$

où la conductivité totale est donnée par l'éq. (2.112) ci-dessus, et H est une source de chaleur intense et concentrée au centre du tore (ici $x = 0$) et qu'on supposera prendre la forme d'un profil Gaussien:

$$H(x) = H_0 \exp \left(-\frac{x^2}{\sigma^2} \right) , \quad \sigma = 0.1 \quad (2.114)$$

avec $\sigma = 1$ dans tout ce qui suit, et où l'amplitude H_0 est un paramètre du modèle. Ce terme “simule” la libération de chaleur, fortement concentrée au centre de la section du tore, résultant de la fusion nucléaire. L'intégrale de ce terme source sur l'ensemble du domaine spatial définit la quantité de chaleur qui est produite par la fusion en régime stationnaire, et qui doit être extraite du système pour que l'ensemble du plasma à l'intérieur du tokamak demeure globalement en équilibre thermique.

Pour ce qui est des conditions initiale et limites on pose simplement:

$$T(x, 0) = 1 , \quad T(\pm 10, t) = 1 . \quad (2.115)$$

2.9.3 Traitement numérique

L'équation (2.113) est une équation de type diffusion, avec un terme source au membre de droite. Cependant ici la conductivité dépend de la position via la dépendance nonlinéaire de χ sur T , et donc doit également être discrétisée. Réécrivons l'expression ci-dessus sous forme conservative

$$\frac{\partial T}{\partial t} = -\frac{\partial F}{\partial x} + H(x) , \quad F = -\chi(T) \frac{\partial T}{\partial x} , \quad (2.116)$$

où F est le flux diffusif. Dans l'esprit Lax-Wendroff, évaluons ce flux à mi-pas spatialement, de telle sorte que

$$\frac{\partial F}{\partial x} = \frac{F_{j+1/2} - F_{j-1/2}}{\Delta x} , \quad (2.117)$$

avec

$$F_{j+1/2} = \frac{1}{2}(\chi_j + \chi_{j+1}) \frac{T_{j+1} - T_j}{\Delta x} \quad (2.118)$$

$$F_{j-1/2} = \frac{1}{2}(\chi_{j-1} + \chi_j) \frac{T_j - T_{j-1}}{\Delta x}. \quad (2.119)$$

Substituant ceci dans l'éq. (2.117) on arrive à:

$$\frac{\partial F}{\partial x} = \frac{1}{2(\Delta x)^2} \left((\chi_j + \chi_{j-1})T_{j-1} - (\chi_{j+1} + 2\chi_j + \chi_{j-1})T_j + (\chi_j + \chi_{j+1})T_{j+1} \right) \quad (2.120)$$

Maintenant la discréttisation temporelle; l'idée ici sera de toujours évaluer le terme nonlinéaire (la conductivité $\chi(T)$) à partir du pas de temps connu, mais d'évaluer les T_j entre parenthèses à mi-pas, dans le sens de Crank-Nicolson. On se retrouve donc avec un schéma centré dans le temps et l'espace:

$$\begin{aligned} \frac{T_j^{n+1} - T_j^n}{\Delta t} &= \frac{1}{4(\Delta x)^2} \left[\overbrace{((\chi_j^n + \chi_{j-1}^n)T_{j-1}^n - (\chi_{j+1}^n + 2\chi_j^n + \chi_{j-1}^n)T_j^n + (\chi_j^n + \chi_{j+1}^n)T_{j+1}^n)}^{\text{explicite}} \right. \\ &\quad \left. + \underbrace{((\chi_j^n + \chi_{j-1}^n)T_{j-1}^{n+1} - (\chi_{j+1}^n + 2\chi_j^n + \chi_{j-1}^n)T_j^{n+1} + (\chi_j^n + \chi_{j+1}^n)T_{j+1}^{n+1})}_{\text{implicite}} \right] + H(x_j). \end{aligned} \quad (2.121)$$

Je vous laisse le plaisir (!) de calculer la forme des éléments des trois diagonales de ce systèmes d'équations algébriques linéaires (les a_i , b_i et c_i dans l'éq. (2.58)). L'utilisation d'un schéma implicite de type Crank-Nicolson, qui minimise la diffusion numérique, s'impose en vertu de la nature du problème considéré ici: si on cherche à simuler numériquement la suppression dynamique d'un processus diffusif, on ne voudra certainement pas faire ça avec un schéma numérique qui introduit une diffusion artificielle!

La Figure 2.16 monte un exemple de solution numérique de l'éq. (2.113), pour les valeurs de paramètres telles que données dans la légende. Attention, ici les pas de temps tracés ne sont pas du tout espacés également dans le temps, mais à peu près uniformément en $\log(t)$, comme sur les Figs. 2.3 et 2.4. Même si poussée jusqu'à $t = 25$, cette solution est encore bien loin de son état d'équilibre ($dT/dt = 0$); il faudrait pousser au moins aussi loin que le "long" temps diffusif, soit celui associé à χ_0 : $\tau = L^2/\chi_0 = 100$ ici, avec χ_0 et $L = 10$. Remarquez les changements discontinus dans le gradient des profils de température. L'un de ces bris de pente se propage vers les frontières du système, tandis que le second demeure près du centre; c'est ce second bris de pente qui est responsable de l'établissement de la barrière de confinement.

La Figure 2.17 montre une vue différente de l'établissement de l'effet barrière. On se concentre ici sur une petite région du domaine près du centre de la section ($0 \leq x \leq 1.5$), correspondant à la région ombragée en gris sur la Fig. 2.16. Les courbes en rouge correspondent au profil du gradient de température dT/dx , tandis que celles en vert montrent les variations du coefficient de conductivité thermique totale $\chi(T(x, t))$ (voir l'éq. (2.112)). La variation rapide de dT/dx se produit là où la partie "turbulente" ($\propto \chi_1$) du coefficient de conductivité se met à chuter drastiquement, quand dT/dx dépasse α^{-1} ($= 1$ ici). Ce processus débute vers $x \simeq \pm 1$, soit là où le terme source chute rapidement vers zéro; dans l'intervalle $-1 \leq x \leq 1$ il y a fort chauffage, mais beaucoup moins au-delà, donc le gradient de température croît le plus rapidement à $x = \pm\sigma = \pm 1$, même si le terme source a son amplitude maximale à $x = 0$.

2.9.4 Explorations numériques

Le projet-devoir associé à ce chapitre consiste à explorer le comportement et le type de solutions produites par le modèle de confinement diffusif nonlinéaire introduit ci-dessus. Les points suivants ne sont que des pistes pour vos explorations numériques. Votre rapport doit couvrir à la fois les aspects numériques et physiques pertinents.

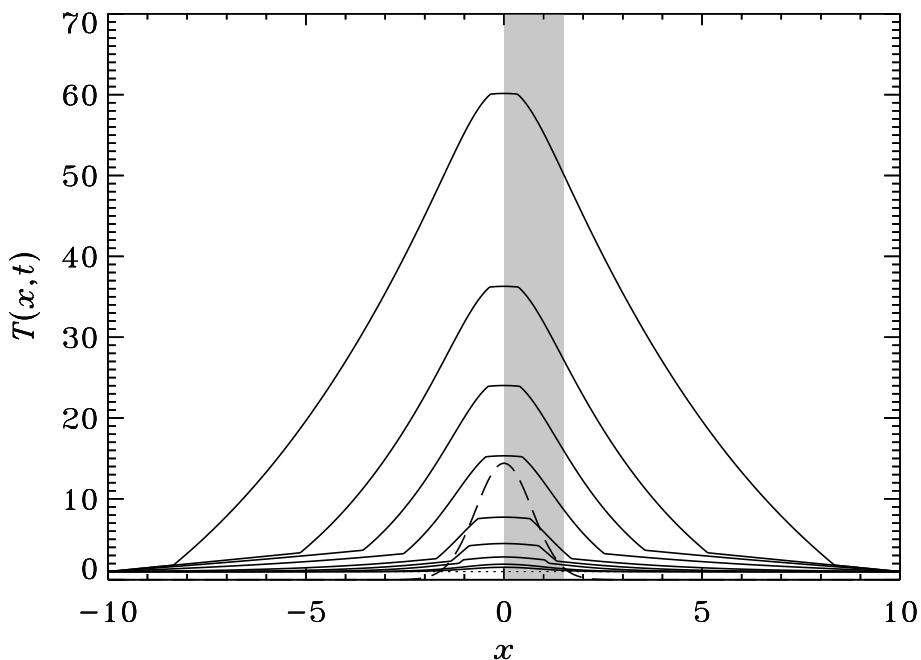


Figure 2.16: Évolution du profil de température pour une solution de l'éq. (2.112) avec valeurs de paramètres $\chi_0 = 1$, $\chi_1 = 8$, $\alpha = 1$, $\beta = 4$, et $H_0 = 14.4$. Le trait en tirets montre le profil Gaussien du terme source. Les courbes sont tracées aux temps $t = 0.05, 0.1, 0.25, 0.5, 2.5, 5.0, 10.0$, et 25.0 (allant du bas vers le haut). Cette solution numérique par Crank-Nicolson utilise des pas $\Delta x = 0.02$ et $\Delta t = 5 \times 10^{-4}$.

1. Calculez l'intégrale de l'éq. (2.114) sur le domaine, pour des H_0 dans intervalle [2., 22.]; retournant à la Fig. 2.15, pouvez-vous “prédir” le développement (ou pas) d'une barrière simplement à partir de la grandeur de cette quantité ?
2. Calculez des solutions d'équilibre (i.e. pousser l'évolution jusqu'à ce que $dT/dt \simeq 0$), pour différentes valeurs de $1 \leq \chi_1 \leq 15$ et différents $2 \leq H_0 \leq 22$. Quelles sont les conditions nécessaires pour produire un confinement en mode H ?
3. La Figure 2.17 montre clairement qu'une fois établie, la barrière de confinement migre lentement vers le centre du système ($x = 0$). Calculez la vitesse de déplacement de cette barrière, et montrez qu'en bonne première approximation cette vitesse $\propto \sqrt{t}$. Comment interprétez-vous ce résultat ?
4. Portez en graphique sur la Fig. (2.15) les valeurs de flux de chaleur et de gradient de température pour chaque point de maille d'une solution à l'équilibre avec effet barrière. Ensuite répétez cette manœuvre pour une seconde solution où H_0 a été abaissé au point de perdre l'effet barrière. Comment interprétez-vous les différences entre ces deux diagrammes ?
5. Produisez une solution à l'équilibre avec effet barrière. Ensuite, posez H_0 et laissez le système diffuser jusqu'à ce que $T(x) = 1$ partout (retour à la première condition initiale). Cette seconde évolution retrace-t-elle exactement, à l'inverse, celle se développant quand le terme source est “allumé” sur une solution où $T(x) = 1$ partout ? Comment expliquez-vous les différences ?
6. À regarder la Figure 2.16 il est clair que la solution est symétrique par rapport à $x =$

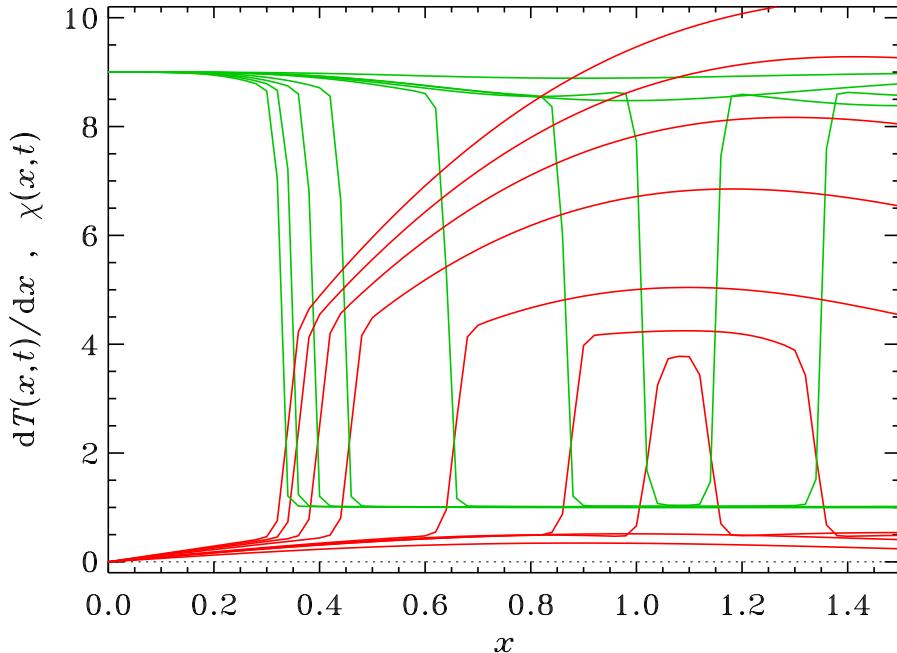


Figure 2.17: Évolution spatiotemporelle du gradient de température (en rouge) et conductivité thermique totale (en vert) près du centre du domaine, pour la solution de la Fig. 2.16. Les courbes sont tracées dans une petite partie du domaine près du centre, correspondant à la zone grise sur la Fig. 2.16, et aux mêmes pas de temps.

0, donc c'est du gros gaspillage de ressources numériques que de modéliser le système dans $x \in [-10, 10]$. Modifiez la simulation de manière à pouvoir calculer seulement dans $x \in [0, 10]$. Réfléchissez bien à la condition limite à imposer à $x = 0$, et à la manière de l'incorporer dans la méthode Crank-Nicolson.

2.10 Bibliographie

La solution numériques des équations aux dérivées partielles a produit une très volumineuse littérature. J'ai beaucoup appris du vieux classique qui suit, toujours très clair et pertinent:

Lapidus, L., & Pinder, G.F., *Numerical solution of partial differential equations in science and engineering*, Wiley (1982).

La présentation des méthodes explicites de la §2.3 et du critère de Courant suivent de près l'excellente présentation de ces sujets se trouvant à la §19.1 de:

Press, W.H., Teukolsy, S.A., Vetterling, W.T., & Flannery, B.P., *Numerical Recipes*, seconde éd., Cambridge University Press (1992);

la méthode Crank-Nicolson y est également discutée plus loin au même chapitre. La méthode Θ est discutée dans l'ouvrage de Burnett ci-dessus, mais sur ce sujet voir également:

Wood, W.L., *Practical time-stepping schemes*, Oxford University Press (1990)

L'article de Turing cité à la §2.2 est le suivant:

Turing, A.M., *Phil. Trans. Roy. Soc. London B*, **237**, 37–72 (1952),

et est republié au chapitre 15 de l'excellent ouvrage:

Copeland, B.J. (éd.), *The essential Turing*, Oxford University Press (2013).

Une des applications les plus spectaculaires et convaincantes des systèmes réaction-diffusion est probablement l'explication des patterns de taches, bandes, etc., observées dans le monde animal. Voir par exemple le chapitre 15 dans

Murray, J.D., *Mathematical Biology*, Springer (1989)

Sur les méthodes d'éléments finis, mon ouvrage préféré demeure:

Burnett, D.S., *Finite Element Analysis*, Addison-Wesley (1988),

tandis qu'au niveau des méthodes spectrales, le bouquin suivant est une excellente introduction:

Canuto, C., Hussaini, M.Y., Quarteroni, A., & Zang, T.A., *Spectral Methods in Fluid Dynamics*, Springer (1988).

Une comparaison intéressante et instructive des forces et faiblesses (relatives) des méthode de différences finies, éléments finis, et spectrales est présentée au chapitre 6 de

Fletcher, C.A.J., *Computational Galerkin Methods*, Springer (1984).

Au niveau des méthodes de volumes finis et techniques semblables, voir e.g.,

Leveque, R.J., *Numerical methods for Conservation Laws*, Birkhäuser (1992).

Le confinement des plasmas en tokamak est un sujet gigantesque. Le modèle décrit à la section 2.8 trouve son origine dans:

Hinton, F.L., & Staebler, G.M., *Phys. Fluids B*, **5**, 1281–1288 (1993),

Malkov, M.A., & Diamond, P.H., *Phys. Plasmas*, **15**, 122301 (2008).

Si vous avez besoin de vous rafraîchir la mémoire sur le calcul des champs magnétiques dans des solénoides ou bobines de formes diverses, voir le chapitre 5 de

Griffith, D.J., *Introduction to electrodynamics*, 3^e éd., Prentice Hall (1999).

Au niveau de la physique du confinement plasma, le vieux classique suivant est toujours tout à fait recommandable:

Chen, F.F., *Introduction to Plasma Physics and controlled Fusion*, 2^e éd., Springer (1983).

Pour plus de détails sur la simulation gyrocinétique de la Fig. 2.14, voir

<https://fusion.gat.com/theory/gyro>.



Chapitre 3

Calcul sur réseau

La formulation des lois physiques sous la forme d'équations différentielles, comme on l'a étudié aux deux chapitres précédents, remonte habituellement à l'expression de lois de conservation pour des quantités physiques d'intérêt: quantité de mouvement, énergie, masse, etc. Une telle formulation peut devenir problématique quand le système évolue sur une très grande gamme d'échelles spatiales. Il devient alors impossible en pratique de bien résoudre les petites échelles tout en définissant la simulation à l'échelle globale du système. Même si on a accès à un ordinateur ayant suffisamment de mémoire pour le faire, les contraintes de stabilité numérique du genre du critère de Courant se retrouvent à imposer un pas de temps souvent minusculement plus petit que la durée sur laquelle on voudrait pousser la simulation.

Il existe en fait quelques options pour se déétriper de ce problème: méthodes dites multigrid, couplages de simulations distinctes opérant sur des échelles différentes, etc. Nous considérerons ici une solution à la fois simple et extrême, soit le *calcul sur réseau*. Il s'agit essentiellement de remplacer les lois physiques par des "règles évolutives" simples capturant néanmoins (on l'espère) la physique essentielle du problème. Parce que ces règles sont simples, autant conceptuellement qu'au niveau de leur formulation numérique, il devient possible d'effectuer des simulations couvrant une vaste gamme d'échelles spatiales et/ou temporelles.

Le but de ce chapitre est donc de présenter une introduction au calcul sur réseau. Après une brève prélude surtout notationnel (§3.1), on examinera en détail un exemple tiré de la géophysique, soit une version sur réseau du modèle de faille séismique de Burridge-Knopoff (§3.2). Notre second exemple sera le modèle d'Ising de la magnétisation (§3.4), qui nous conduira au projet final qui clôt ce chapitre, soit la gravure magnétique par laser.

3.1 Calculs sur réseaux

Un *réseau* est constitué d'un ensemble de *noeuds* interconnectés. Il convient de distinguer deux caractéristiques des réseaux: la *géométrie* et la *connectivité*. La géométrie réfère aux positions des noeuds dans l'espace physique —où ici "espace physique" peut inclure des espaces de phase plutôt abstraits. La connectivité réfère aux liens dynamiques entre les noeuds, i.e., quels noeuds interagissent avec quels autres. Dans tout ce qui suit on travaillera avec des réseaux cartésiens réguliers, puisque ceux-ci sont triviaux à représenter numériquement en terme de tableaux ayant les même dimensions que celles du réseau, mais au niveau du comportement dynamique c'est la connectivité qui est cruciale.

En pratique la connectivité est déterminée par un *stencil de voisinage* donnant la position *relative* des voisins "connectés" par rapport à un noeud donné. Par exemple, travaillant sur un réseau Cartésien régulier en deux dimensions spatiales, les 4 voisins les plus rapprochés d'un noeud (i, j) sont les noeuds $(i, j + 1)$, $(i + 1, j)$, $(i, j - 1)$ et $(i - 1, j)$, où les indices i et j numérotent les positions verticales et horizontales des noeuds dans le réseau 2D. C'est le *stencil de von Neumann*. En Python/numpy ça pourrait avoir l'air de ceci:

```
dx=np.array([0,1,0,-1],dtype='int')
dy=np.array([-1,0,1,0],dtype='int')
```

Si on ajoute les quatre voisins en diagonale, on obtient le *stencil de Moore*:

```
dx=np.array([0,1,0,-1,-1,1,1,-1],dtype='int')
dy=np.array([-1,0,1,0,-1,-1,1,1],dtype='int')
```

Le stencil à 6-voisins qui suit définit une connectivité de type von Neumann, mais pour un réseau *triangulaire*:

```
dx=np.array([0,1,0,-1,-1,1],dtype='int')
dy=np.array([-1,0,1,0,-1,1],dtype='int')
```

Je vous laisse imaginer en quoi ceci peut être considéré “triangulaire”... Quelle que soit la connectivité, ces stencils permettent d’effectuer des calculs et/ou tests impliquant le voisinage d’un noeud. Par exemple, l’exemple suivant calcule la somme de tous les voisins du noeud (i, j) dans un réseau 2D grid:

```
sum=0.
for k in range(0,4): sum+=grid[i+dx[k],j+dy[k]]
```

De telles instructions se retrouverait normalement imbriquées dans une double boucle en i et j sur les dimensions du réseau. Cette approche numérique fait cependant face à un problème, illustré sur la Figure 3.1, toujours pour le stencil de von Neumann: à moins d’introduire des stencils modifiés spécifiques aux noeuds de bord, ou encore de surcharger le code d’instructions de type **if...else**, on aura débordement de tableau pour les noeuds de bords, qui n’ont que trois voisins (et seulement deux pour les noeuds de coin).

La façon la plus propre de régler ce problème est d’introduire une couche de *noeuds fantômes* en périphérie du réseau, comme l’illustre la Figure 3.2 (cercles gris). Le réseau est maintenant de dimensions 12×12 , mais les calculs ne se font que sur les 10×10 noeuds du réseau original. Mais maintenant, du point de vue du stencil tous les noeuds sont des noeuds “intérieurs”.

On doit évidemment maintenant assigner une valeur numérique à la variable nodale aux noeuds fantômes; cette information est habituellement fournie par les conditions limites du problème. Parfois il suffira de fixer la valeur des noeuds fantômes à une valeur qui demeure fixe au cours du calcul; parfois la spécification des conditions limites est plus complexe. La Figure 3.3 montre comment utiliser les noeuds fantômes pour imposer des conditions limites périodiques sur les deux dimensions du réseau.

À regarder les Figs. 3.1—3.3 il est difficile de ne pas penser à un maillage 2D servant à discréteriser une quelconque EDP par différences finies (cf. Fig. 2.12!), et donc que la variable nodale est simplement une fonction discrétisée exactement comme au chapitre précédent; mais il ne faut vraiment pas le voir comme ça. Ici un “noeud” peut être une structure passablement complexe en soi (une molécule, un grain de sable, un arbre, etc.), et les règles d’évolution peuvent capturer des processus dynamiques également complexes (une réaction chimique, un échange de particules, une collision anélastique, une décharge électrique, etc.). C’est précisément en faisant abstraction du *détail* de ces structures et processus qu’on en arrive à construire une simulation qui capture les comportements globaux résultant de ces interactions, sans en capturer les détails à l’échelle “microscopique”.

Tout ça peut paraître bien abstrait, donc passons immédiatement à un exemple spécifique qui permettra de mieux apprécier cette distinction entre un calcul sur réseau, et la solution numérique d’une EDP discrétisée sur une maille cartésienne.

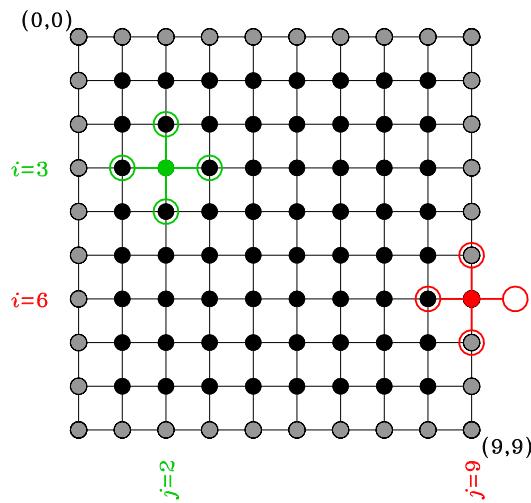


Figure 3.1: Utilisation correcte (en vert) et incorrecte (en rouge) d'un stencil 4-voisins sur un réseau Cartésien de dimensions 10×10 . Le stencil fonctionne bien pour les noeuds intérieurs (en noir) mais causera des débordements de tableaux pour les noeuds de bords (en gris) à moins d'utiliser des stencils modifiés adaptés spécifiquement à ces noeuds.

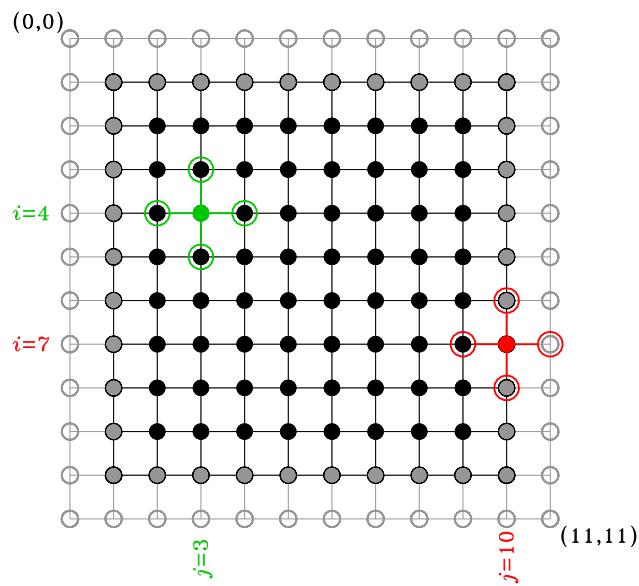


Figure 3.2: Le même réseau que ci-dessus, mais cette fois incluant une couche de noeuds fantômes en périphérie (cercles gris). Le réseau complet est maintenant de dimensions 12×12 , mais le stencil 4-voisins peut maintenant être utilisé sur les “vrais” noeuds de bords (en gris).

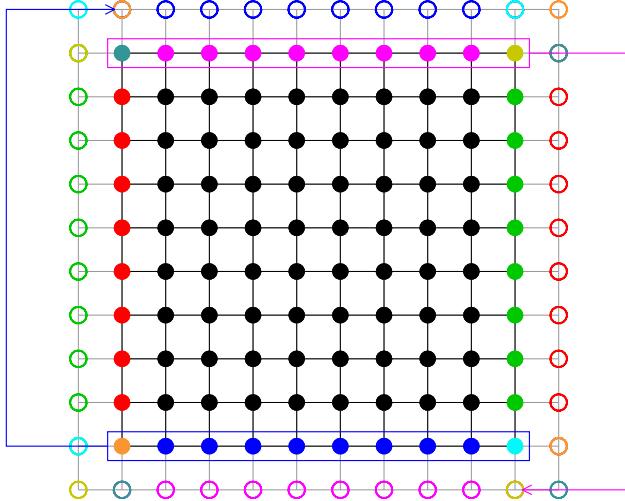


Figure 3.3: Conditions limites périodiques via les noeuds fantômes. Les noeuds de bord du réseau original 10×10 (en gris sur la Fig. 3.1) sont recopiés dans la rangée de noeuds fantômes du côté opposé du réseau 12×12 , tel qu'indiqué par le code couleur. Les noeuds de coin doivent être recopiés trois fois: horizontalement, verticalement, et diagonalement. Ce processus n'affecte pas les noeuds intérieur (noir) du réseau original 10×10 .

3.2 Exemple: les tremblements de terre

3.2.1 La tectonique des plaques (en bref)

La *tectonique des plaques* est la grande théorie unificatrice de la géophysique contemporaine, un peu comme l'évolution en biologie ou la mécanique quantique en physique. Plus d'un siècle aura été nécessaire pour y arriver, mais on comprend maintenant que la croûte terrestre est fragmentée en une douzaine de plaques tectoniques majeures (voir la Figure 3.4), d'une centaine de kilomètres d'épaisseur, flottant sur un magma fluide appelé *asténosphère*. Ce fluide est en mouvement, un peu comme de l'eau dans un chaudron juste avant qu'elle ne se mette à bouillir, en raison du flux de chaleur résiduelle provenant du noyau terrestre, et de l'asténosphère même via la radioactivité naturelle. Cet écoulement devient horizontal sous les plaques tectoniques, et la force visqueuse y étant associée tend à déplacer ces dernières dans la direction de l'écoulement. La friction entre les plaques s'oppose à cette force visqueuse tendant à les déplacer les unes par rapport aux autres. Ces *lignes de failles* sont donc sujettes à de forts stress mécaniques. Le roc réagit en se déformant élastiquement au début, mais il vient un moment où la friction statique entre les bords des plaques ne peut plus équilibrer la force visqueuse; les plaques se déplacent alors soudainement, causant un tremblement de terre.

L'intensité des tremblements de terre est mesurée par leur *magnitude* (m), qui est essentiellement une mesure de l'amplitude des ondes séismiques relevées par les sismographes. Il s'avère que pour un ensemble de tremblements de terre, cette magnitude est distribuée en loi de puissance. Plus spécifiquement, pour un intervalle de temps donné le nombre N de tremblements de terre ayant une magnitude plus grande que m est donné par:

$$N(>m) \propto m^{-b}, \quad (3.1)$$

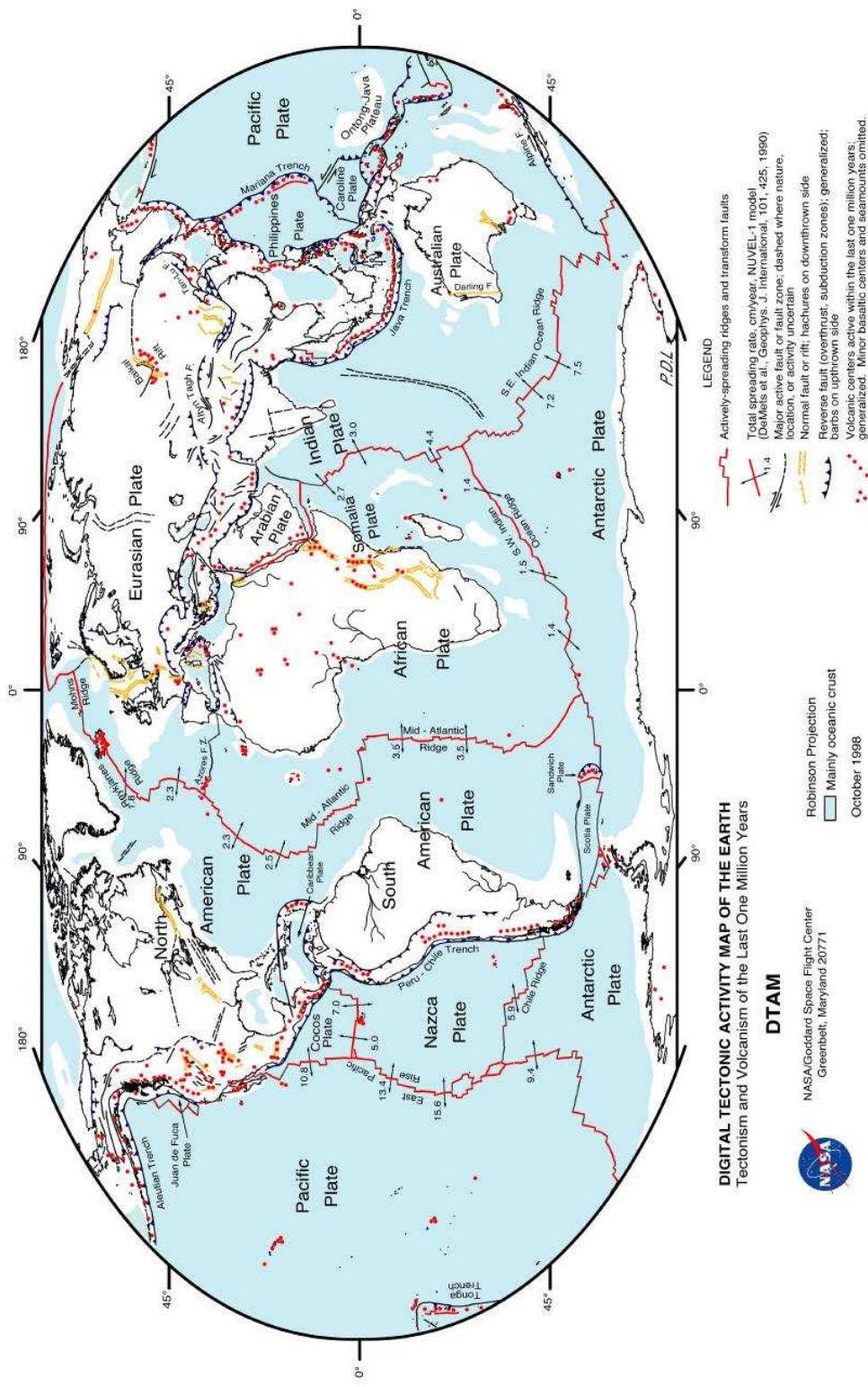


Figure 3.4: Les plaques tectoniques terrestres et leurs mouvements. Certaines glissent horizontalement l'une contre l'autre, d'autre s'éloignent l'une de l'autre, et certaines se rapprochent avec l'une montant par dessus l'autre. Image de la NASA, en domaine public; voir <http://earthobservatory.nasa.gov/Features/Tectonics>.

C'est la très célèbre *Loi de Gutenberg-Richter*. La valeur numérique de l'indice b varie un peu d'une faille à l'autre, mais $b \simeq 1$ pour la plupart d'entre elles. Une telle loi de puissance indique que la dynamique des tremblements de terre est invariante par rapport à la taille de ces derniers. C'est là une propriété remarquable, qui s'avère pouvoir être reproduite par le petit modèle sur réseau dans lequel il est maintenant temps de se lancer.

3.2.2 Le modèle de Burridge-Knopoff

Le modèle mécanique des lignes de failles séismiques mis au point par Burridge et Knopoff (voir bibliographie en fin de chapitre), aussi connu sous le nom de “stick-slip”, est défini comme un assemblage 2D de blocs, chacun connecté à ses voisins immédiats par des ressorts, le tout étant pris en sandwich entre deux grande plaques, celle du bas au repos et celle du haut se déplaçant à une vitesse donnée V . Chaque bloc est également connectés à la plaque du haut par un autre ressort dit primaire. La Figure 3.5 illustre cette configuration pour le bloc (i, j) et ses quatre voisins immédiats $(i-1, j)$, $(i+1, j)$, $(i, j-1)$ et $(i, j+1)$. Le déplacement de la plaque du haut simule l'écoulement du magma de l'asténosphère, et la force visqueuse que ce mouvement exerce sur les plaques tectoniques. Ce déplacement étire graduellement les ressorts primaires, augmentant inexorablement la composante- x de la force agissant sur chaque bloc. On supposera ici que cette force est donnée par la Loi de Hooke habituelle:

$$F_x = K\Delta x , \quad (3.2)$$

où K est la constante du ressort et le déplacement $\Delta x \equiv x_{i,j}$ est défini comme la distance entre le centre du bloc et le point d'ancrage de son ressort primaire sur la plaque du haut (voir Fig. 3.5). On dénotera par K et K_L les constantes associées aux ressorts internes et primaires, respectivement.

La composante- x de la force totale agissant sur le bloc (i, j) est donc donnée par:

$$F_{i,j}^n = K(x_{i-1,j}^n - x_{i,j}^n) + K(x_{i+1,j}^n - x_{i,j}^n) + K(x_{i,j-1}^n - x_{i,j}^n) + K(x_{i,j+1}^n - x_{i,j}^n) - K_L x_{i,j}^n \quad (3.3)$$

$$= K(x_{i-1,j}^n + x_{i+1,j}^n + x_{i,j-1}^n + x_{i,j+1}^n - 4x_{i,j}^n) - K_L x_{i,j}^n \quad (3.4)$$

où, en anticipation de ce qui suivra sous peu, l'indice n dénote une itération temporelle. Notons déjà qu'un déplacement $x_{i,j}$ sera en général une quantité négative, puisqu'on a choisi de le mesurer à partir du point d'ancrage sur la plaque supérieure (voir Fig. 3.5). Par conséquent, un terme du genre $-K_L x_{i,j}^n$ dans l'éq. (3.3) est positif, et indique que le ressort primaire exerce sur le bloc une force orientée dans la direction positive de l'axe- x , comme il se doit.

Une force importante manque dans l'éq. (3.3): la friction statique entre le bloc et la plaque du bas. Tant que cette force parvient à équilibrer la force produite par l'ensemble des ressorts, le bloc demeure au repos. Mais comme le déplacement de la plaque supérieure se fait à vitesse constante, inévitablement on atteint un point où la force de friction devient insuffisante pour tenir le bloc au repos, et ce dernier glissera dans la direction- x . L'idée dynamique cruciale du modèle de Burridge-Knopoff est que le bloc glissera très rapidement jusqu'à une position où la force totale associée aux divers ressorts est nulle:

$$F_{i,j}^{n+1} = K(x_{i-1,j}^n + x_{i+1,j}^n + x_{i,j-1}^n + x_{i,j+1}^n - 4x_{i,j}^{n+1}) - K_L x_{i,j}^{n+1} = 0 . \quad (3.5)$$

La variation de la force totale agissant sur le bloc est donc donnée par:

$$\delta F_{i,j} \equiv F_{i,j}^{n+1} - F_{i,j}^n = (4K + K_L)(x_{i,j}^{n+1} - x_{i,j}^n) \quad (3.6)$$

Puisque $F_{i,j}^{n+1} = 0$ par hypothèse (soit l'éq. (3.5)), le membre de droite de cette expression doit donc être égal à $-F_{i,j}^n$. Considérons maintenant le bloc voisin $(i+1, j)$. Si l'on suppose que seul le bloc (i, j) a glissé, le changement de la force sur le bloc $(i+1, j)$ est simplement donné par:

$$\delta F_{i+1,j} = K(x_{i,j}^{n+1} - x_{i,j}^n) . \quad (3.7)$$

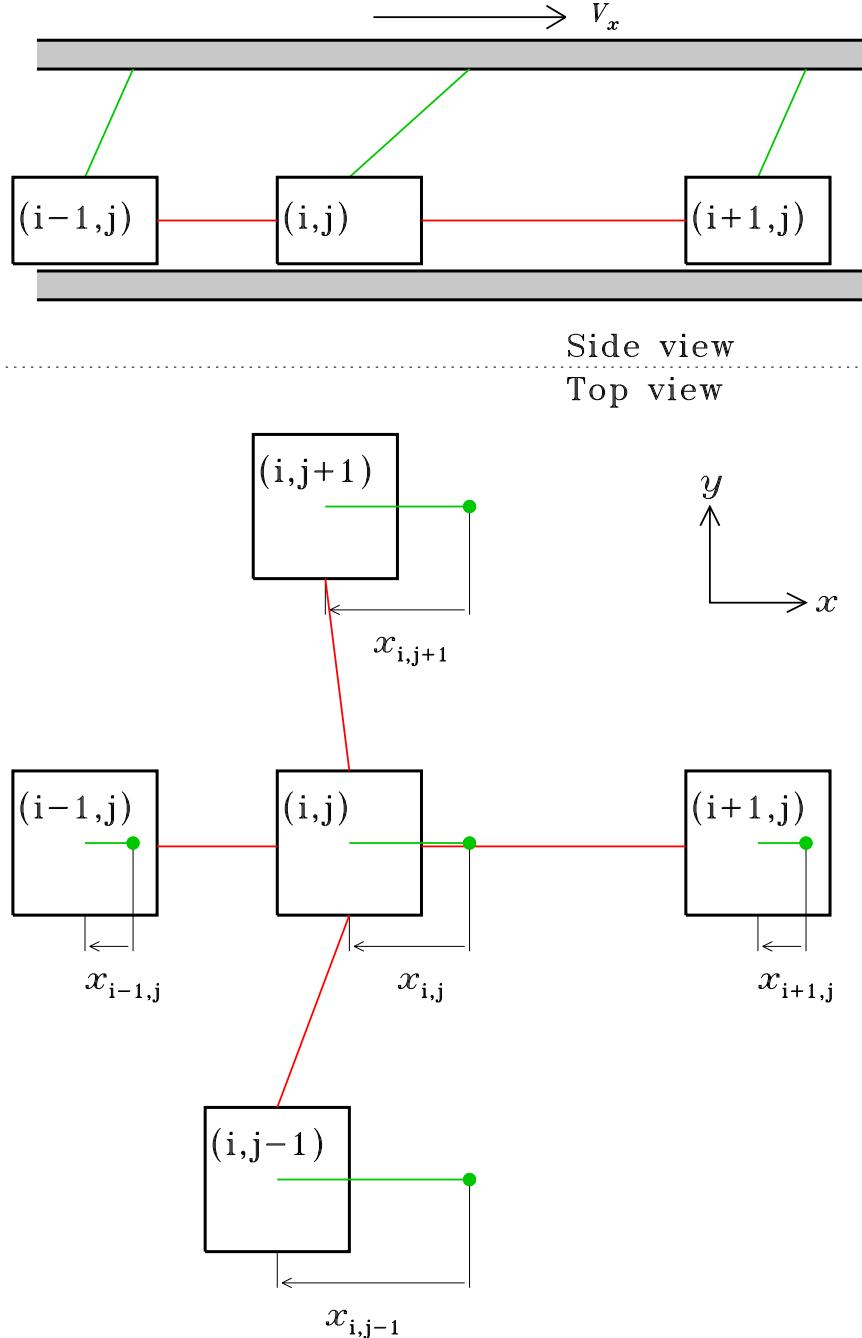


Figure 3.5: Le modèle “stick-slip” de Burridge-Knopoff, vu ici de coté et du dessus. La plaque du bas est supposée au repos, mais celle du haut se déplace à vitesse constante V dans la direction x . Les traits verts représentent les ressorts primaires connectant les blocs à la plaque du haut, et les rouges les ressorts internes connectant les blocs entre eux. Le déplacement des blocs (x) est mesuré à partir du point d’ancrage des ressorts primaires sur la plaque du haut, tel qu’indiqué par les points verts sur la projection du bas.

Ceci doit être égal à $\delta F_{i,j}$, Newton oblige. Substituant l'équation (3.6) pour $x_{i,j}^{n+1} - x_{i,j}^n$ dans l'éq. (3.7), on obtient:

$$\delta F_{i+1,j} = \alpha F_{i,j}^n , \quad (3.8)$$

où

$$\alpha = \frac{K}{4K + K_L} . \quad (3.9)$$

On en conclut que la force sur le bloc $(i+1, j)$ varie en proportion à la force agissant sur le bloc (i, j) avant son glissement. Ce résultat tient également pour les trois autres voisins. En général le bloc glissera également dans la direction- y , sauf si par hasard $x_{i,j+1} = x_{i,j-1}$; en terme de l'évolution à long terme du système, ce déplacement peut être ignoré car en moyenne il sera nul, en raison du fait que la plaque du haut se déplace dans la direction- x . De plus, l'expression ci-dessus ne tient que pour les blocs "intérieurs"; pour ceux situés sur les bords de l'assemblage, l'éq. (3.9) devient $\alpha = K/(3K + K_L)$ et $\alpha = K/(2K + K_L)$ pour les quatre blocs de coin. Dans tous les cas, la valeur numérique de la constante α est déterminée par le rapport des constantes des ressorts interne et primaire. Notons en particulier que

$$\lim_{K \ll K_L} \alpha \rightarrow 0 , \quad \lim_{K \gg K_L} \alpha \rightarrow \frac{1}{4} . \quad (3.10)$$

Notre tâche est maintenant de définir un modèle sur réseau qui capture correctement la dynamique du modèle de Burridge-Knopoff. On suivra ici l'approche proposée par Olami, Feder et Christensen (abrévié ci-après OFC; voir bibliographie en fin de chapitre). La clef est ici d'utiliser comme variable nodale la force $F_{i,j}$ agissant sur le noeud (i, j) , plutôt que la position $x_{i,j}$ du bloc, le choix qui aurait été peut-être plus intuitif.

Le modèle OFC est défini sur un réseau Cartésien de taille $N \times N$, avec stencil de voisinage droite+gauche+haut+bas. La variable nodale est dénotée $F_{i,j}^n$, la paire d'indices (i, j) identifiant un noeud et l'indice n l'itération temporelle. Cette variable nodale est sujette à un forçage déterministe d'amplitude constante. Plus précisément, à chaque itération temporelle un petit incrément δF est ajouté à chaque noeud du réseau:

$$F_{i,j}^{n+1} = F_{i,j}^n + \delta F , \quad \forall i, j . \quad (3.11)$$

Ceci est l'équivalent de l'augmentation de la force sur chaque bloc causée par le déplacement de la plaque supérieure dans le modèle de Burridge-Knopoff. Si la variable nodale dépasse un seuil pré-établi F_c

$$F_{i,j}^n > F_c , \quad (3.12)$$

correspondant à la force de friction statique entre le bloc et la plaque du bas, alors le noeud relaxe vers un état de force nulle en redistribuant sa force aux noeuds voisins:

$$F_{i,j}^{n+1} = 0 , \quad (3.13)$$

$$F_{nn}^{n+1} = F_{nn}^n + \alpha F_{i,j}^n , \quad 0 \leq \alpha \leq 0.25 , \quad (3.14)$$

où $nn \equiv (i+1, j)(i-1, j)(i, j+1)(i, j-1)$, et α est la constante de proportionnalité apparaissant aux éqs. (3.8)–(3.9), i.e., elle mesure la fraction de la force agissant sur les noeuds instables qui est transmise à la plaque du haut, et donc "perdue" du point de vue du réseau, plutôt que d'être redistribuée aux noeuds voisins. Cette redistribution restaure évidemment la stabilité au noeud (i, j) , mais un ou plusieurs de ses quatre voisins peuvent avoir été poussés au delà du seuil de stabilité par l'apport provenant de (i, j) , ce qui peut conduire à des redistributions aux itérations temporelles subséquentes, et possiblement à une *avalanche* de redistributions se propageant en cascade dans le réseau. La figure 3.6 illustre schématiquement ce processus de

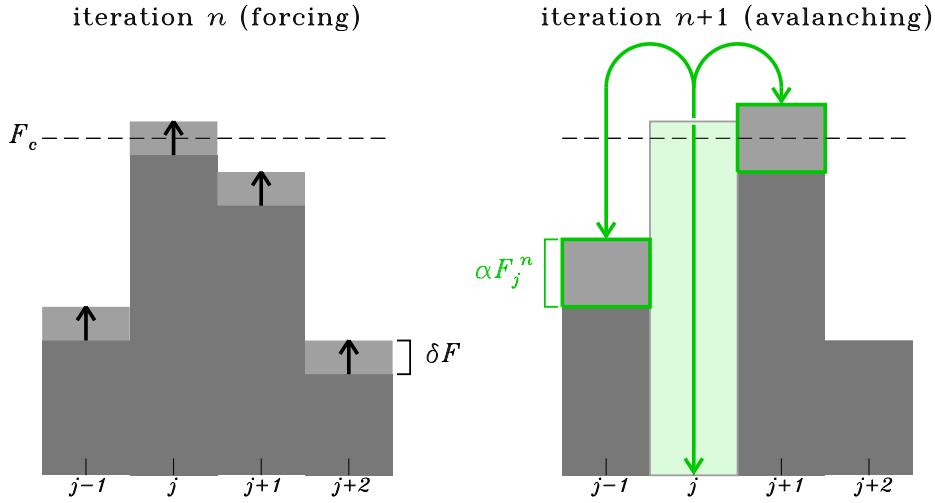


Figure 3.6: Opération de la règle de redistribution définie par les éqs. (3.13)–(3.14), ici simplifiées à une dimension spatiale. Le réseau est stable partout au début de l’itération n (à gauche), mais le forçage uniforme (flèches noires; voir l’éq. (3.11)) pousse le noeud j au delà du seuil de stabilité F_c (ligne en tirets). À l’itération suivante (à droite) le noeud j est ramené à zéro et une fraction αF_j^n de sa valeur antérieure F_j^n est redistribuée à part égale aux deux noeuds voisins. La forçage stoppe durant la redistribution, ce qui classe le modèle dans la catégorie dite “stop-and-go”.

redistribution, dans une situation 1D où le noeud j dépasse le seuil de stabilité suite à l’addition d’un incrément δF (à gauche). La redistribution en découlant (à droite) pousse le noeud $j + 1$ au delà du seuil de stabilité, ce qui conduira à une nouvelle redistribution d’une quantité αF_{j+1}^{n+1} vers les noeuds j and $j + 2$ à l’itération temporelle suivante, après quoi le réseau sera de nouveau partout stable.

En terme de modèle sur réseau, ceci ne semble pas différer trop de certains modèles semblables avec lesquels certain(e)s d’entre vous ont peut-être déjà fait connaissance, comme le modèle dit “Tas-de-Sable” en PHY-1234. Une différence importante relève du fait que pour $\alpha < 0.25$ dans l’éq. (3.14), le modèle OFC est *non-conservatif*: la somme de la variable nodale est plus petite après la redistribution qu’avant. Ici le choix $\alpha = 0$ correspond à un découplage complet de chaque bloc de ses voisins (la constante de ressort $K = 0$), dans lequel cas la force $F_{i,j}^n$ à un noeud instable est entièrement transférée à la plaque supérieure via le ressort primaire. Ce n’est qu’à la limite opposée, $\alpha = 0.25$, soit un rapport de constantes $K_F/K \ll 1$, que toute la force $F_{i,j}^n$ est redistribuée aux noeuds voisins suite à un glissement, rendant alors la redistribution conservative.

On notera également ici que les règles de forçage, de stabilité et de redistribution sont toutes déterministes. La seule stochasticité présente est introduite au niveau de la condition initiale: à $n = 0$, la variable nodale est initialisée à partir d’une distribution uniforme de nombre aléatoires couvrant l’intervalle de stabilité:

$$F_{i,j}^0 = r , \quad i, j = 1, \dots, N , \quad r \in [0, F_c] . \quad (3.15)$$

Le modèle OFC opère habituellement en mode dit “stop-and-go”, i.e., le forçage est suspendu durant les itérations temporelles où le réseau est instable quelquepart et redistribue la variable nodale, et ne reprend qu’une fois la stabilité rétablie. Physiquement, ceci implique une séparation d’échelles temporelles entre le forçage et la redistribution. C’est une hypothèse qui se justifie très bien dans le contexte de la tectonique des plaques, considérant que les plaques

se déplacent normalement à une vitesse moyenne de l'ordre du centimètre par année (la vitesse à quelle poussent nos ongles!), versus le mètre par seconde durant un déplacement déclenchant un tremblement de terre.

3.2.3 Traitement numérique

La Figure 3.7 présente une formulation numérique minimale en Python du modèle OFC; “minimale” dans le sens qu’ici on a donné priorité à la clarté du code, plutôt qu’à la vitesse d’exécution. Notez bien les points suivants:

1. La simulation est structurée en terme d’une boucle temporelle inconditionnelle simple (débutant à ligne 19), itérant la simulation sur un nombre prédéfini d’itérations `nIter`.
2. Le test de stabilité (ligne 24) est effectué en premier pour tous les noeuds du réseau, et, le cas échéant, la quantité de variable nodale à redistribuer est accumulée dans le tableau `move` (lignes 25 et 26); ce n’est qu’une fois tous les noeuds testés que la mise à jour du réseau `force` est effectuée (ligne 31); une telle *mise à jour synchrone* est nécessaire afin de ne pas introduire de biais spatial artificiel dans la propagation des avalanches.
3. Les tableaux `force` et `move` sont définis de dimensions $(N+2) \times (N+2)$ (lignes 13 et 20), bien que le réseau soit de dimensions $N \times N$. Les 2 rangées et colonnes supplémentaires sont ces fameux noeuds fantômes (voir Fig. 3.2) qui évitent les débordement de tableaux lors de la redistribution (ligne 26). C’est pourquoi les boucles sur les deux dimensions du réseau roulent de 1 à N (lignes 22–23). Les noeuds fantômes sont initialisés à zéro et conservent cette valeur tout le long de la simulation.
4. Les deux tableaux 1D `dx` et `dy` (lignes 11 et 12) définissent les stencil 4-voisins (voir Fig. 3.1). L’équation (3.14) est codée via une boucle implicite sur les éléments de ces tableaux de stencil servant à identifier les noeuds où est redistribuée (dans `move`) la variable nodale des noeuds instables (ligne 26).
5. Le forçage (ligne 33) s’applique à tout le noeuds du réseau, mais seulement si aucun noeud instable n’a été détecté durant l’itération. Forcing takes place at all nodes (line 32), but only
6. La fonction de sommation de tableau de la Librairie Python `numpy` est utilisée pour calculer la force totale appliquée sur le réseau à chaque itération (ligne 37), et le tableau 1D `tsav` accumule la séquence temporelle du nombre de noeuds instables à chaque itération.

3.2.4 Une simulation représentative

Examinons en détail une simulation spécifique mais représentative du comportement du modèle OFC. On travaille sur un réseau de dimensions 128×128 , avec $F_c = 1$, $\delta F = 10^{-4}$ et $\alpha = 0.15$, cette dernière valeur correspondant à une redistribution significativement non-conservative: 40% de la variable nodale est “perdue” à chaque redistribution.

Il est important de pousser la simulation jusqu’à ce qu’un état *statistiquement stationnaire* soit atteint. Pour les paramètres utilisés ici, ceci requiert quelques millions d’itérations temporelles!

Dans le présent contexte une bonne mesure de “l’énergie” libérée par une avalanche est la force collectivement dissipée par toutes les redistributions ayant lieu au cours de l’avalanche. En pratique, ceci sera presque exactement égal au nombre total de redistributions (en comptabilisant individuellement les redistributions se répétant aux mêmes noeuds), puisque chaque redistribution dissipe essentiellement la même force, soit $(1 - 4\alpha) \times F_c$, tant qu’on opère dans le régime $\delta F/F_c \ll 1$.

La Figure 3.8 montre des portions d’une séquence temporelle de la force dissipée par les avalanches dans notre simulation représentative, une fois qu’elle a atteint un état statistiquement stationnaire. Le graphique du haut montre un segment de 40000 itérations, celui du milieu

```

1 # MODELE OLAMI-CHRISTENSEN-FEDER 2D DES FAILLES TECTONIQUES
2 import numpy as np
3 #-----
4 N      =128                      # taille du reseau
5 NN     =4                         # nombre de voisins immediats
6 fc     =5.                        # seuil d'instabilite
7 deltaf=1.e-4                     # amplitude du forage
8 alpha  =0.15                      # parametre de conservation
9 nIter =100000                     # nombre d'iterations temporelles
10 #-----
11 dx=np.array([-1,0,1,0])          # stencil de voisinage
12 dy=np.array([0,-1,0,1])
13 force=np.zeros([N+2,N+2])        # tableau force
14 toppling=np.zeros(nIter,dtype='int') # sequence temporelle avalanches
15 for i in range(1,N+1):
16     for j in range(1,N+1):
17         force[i,j]=fc*(np.random.uniform()) # force initiale aleatoire
18
19 for iter in range(0,nIter):       # iteration temporelle
20     move =np.zeros([N+2,N+2])      # remise a zero du tableau
21     # balayage du reseau pour identifier les noeuds instables
22     for i in range(1,N+1):
23         for j in range(1,N+1):
24             if force[i,j] >= fc:      # le noeud i,j est instable
25                 move[i,j]=-force[i,j]    # Eq (3.13): a zero
26                 move[i+dx[:,],j+dy[:,]]+=alpha*force[i,j] # Eq (3.14): aux voisins
27                 toppling[iter]+=1           # cumul des noeuds instables
28     # fin du balayage du reseau
29
30     if toppling[iter] > 0:          # il y a eu instabilite
31         force+=move                # mise a jour du reseau
32     else:                          # pas d'instabilite
33         force[:,,:]+=deltaf        # Eq (3.11): forage du reseau
34
35     # commandes graphiques pourraient etre inserees ici
36
37     totalf=force.sum()            # force totale sur le reseau
38     print("{0}, force {1}, toppl {2}.".format(iter,totalf,toppling[iter]))
39 # fin de l'iteration temporelle
40 # END

```

Figure 3.7: Code Python minimal pour le modèle Olami-Feder-Christensen; étant donc ici un modèle numérique sur réseau du modèle mécanique de la dynamique de failles tectoniques développé par Burridge et Knopoff.

200000 itérations, et celui du bas 2×10^6 itérations. On y observe des avalanches couvrant une vaste gamme d'amplitude, allant d'une seule redistribution à un peu plus de 4000 pour la plus grande avalanche sur le graphique du bas. Le graphique du haut montre une récurrence claire du même pattern d'avalanches, suggérant un comportement de type périodique, de période $\simeq 10960$ itérations ici. En y regardant de plus près on constate que la périodicité n'est pas parfaite, particulièrement au niveau des plus petites avalanches. Si on pousse à des intervalles plus longs (graphiques du milieu et du bas), on constate que la quasi-périodicité des grandes avalanches s'étend sur un intervalle temporel fini, le pattern évoluant lentement d'un type de périodicité à une autre. Le graphique du milieu capture une telle transition, la taille de la plus grande avalanche passant de 750 dans le premier tier de la séquence, à plus de 1800 dans le dernier tier. Néanmoins, une périodicité demeure présente même sur les plus longues échelles temporelles, comme on peut le constater sur le graphique du bas, même si l'amplitude des plus grandes avalanches change de manière intermittente d'un épisode quasipériodique au suivant.

La Figure 3.9 montre trois séquences temporelles de la variable nodale $F_{i,j}$ à trois noeuds spécifiques du réseau. Un cycle de récurrence y est clairement apparent dans les trois cas. La variable nodale croît lentement en réponse au forçage, à un taux qui est le même pour les trois noeuds, mais cette croissance est parfois interrompue par des sauts de grandeur αF_c vers le haut, se produisant quand le noeud reçoit un incrément de force d'un voisin ayant atteint le seuil d'instabilité. Éventuellement chaque noeud atteint ce seuil, et retombe à zéro. Ici les noeuds "bleu" et "vert" dépassent ce seuil suite à l'instabilité d'un noeud voisin, mais le "rouge" l'atteint via le forçage. Les segments verticaux colorés sur le graphique du haut de la Fig. 3.8 indiquent les temps où ces trois noeuds atteignent le seuil d'instabilité. Aucun de ces trois noeuds ne prend part ici aux grandes avalanches périodiques, mais les noeuds verts et bleus sont impliqués dans des avalanches de taille suffisamment élevée pour être distinguables à l'échelle de ce graphique.

La période de récurrence, $\simeq 10960$ itérations ici, est dangereusement près du nombre d'itérations $t = (\delta F)^{-1} = 10^4$ requises pour que le forçage pousse le noeud de zéro jusqu'au seuil d'instabilité. Ce n'est qu'une partie de l'histoire cependant, puisque tous les noeuds, au cours de leur croissance sautent à quelques reprises de $\alpha F_c = 0.15$ en réponse à l'avalanche d'un noeud voisin (viz. Fig. 3.9). De plus, la simulation opère en mode "stop-and-go", sous une seule boucle temporelle, ce qui implique que le forçage n'agit pas si même un seul noeud est instable quelquepart sur le réseau. On doit donc soustraire les itérations passées en mode "avalanche" de la période de récurrence mesurée à partir de la séquence temporelle. Pour la simulation considérée ici, 63% des iterations sont en mode avalanche, ce qui conduit à une période de récurrence "corrigée" de $\simeq 4000$ iterations, donc une croissance de $F_{i,j}$ par 0.4 sous l'influence du forçage seul, ce qui est cohérent avec ce que l'on peut observer sur la Fig. 3.9. Autrement dit, dans le cadre d'un cycle de récurrence, un noeud "moyen" reçoit quatre incrémentations $+0.15$ d'un voisin instable, et le reste du forçage.

Mais comment une évolution totalement déterministe à partir d'une condition initiale aléatoire peut-elle produire un comportement (quasi)périodique sur des échelles de temps (relativement) courtes, mais apériodiques sur de plus longues échelles ? La réponse est cachée dans les règles de redistribution couplant les noeuds au moment d'une avalanche. La Figure 3.10 montre la condition initiale pour la simulation des Figs. 3.8 et 3.9 (en haut à gauche), et trois instantanés de l'état du réseau loin dans la simulation, séparés de 10^6 itérations. Le pattern aléatoire initial est graduellement remplacé par une mosaique de domaines contigus de formes irrégulières, à l'intérieur desquels la variable nodale est approximativement constante. Les formes et tailles de ces domaines varient lentement au cours de la simulation, lorsqu'une avalanche atteint une de leurs frontières.

C'est l'existence de ces domaines spatiaux qui explique la quasipériodicité observée dans les séquences temporelles de la Fig. 3.8. Que ce soit via le forçage ou par une avalanche provenant d'un domaine voisin, tous les noeuds d'un domaine vont typiquement avalanacher en même temps, le domaine en entier se retrouvant ensuite avec des valeurs nodales de zéro. Plus le domaine est grand, plus l'avalanche qu'il produit est grande. Puisque tous les noeuds du domaine se retrouvent à zéro de manière synchronisée, le domaine se rebâtit à nouveau,

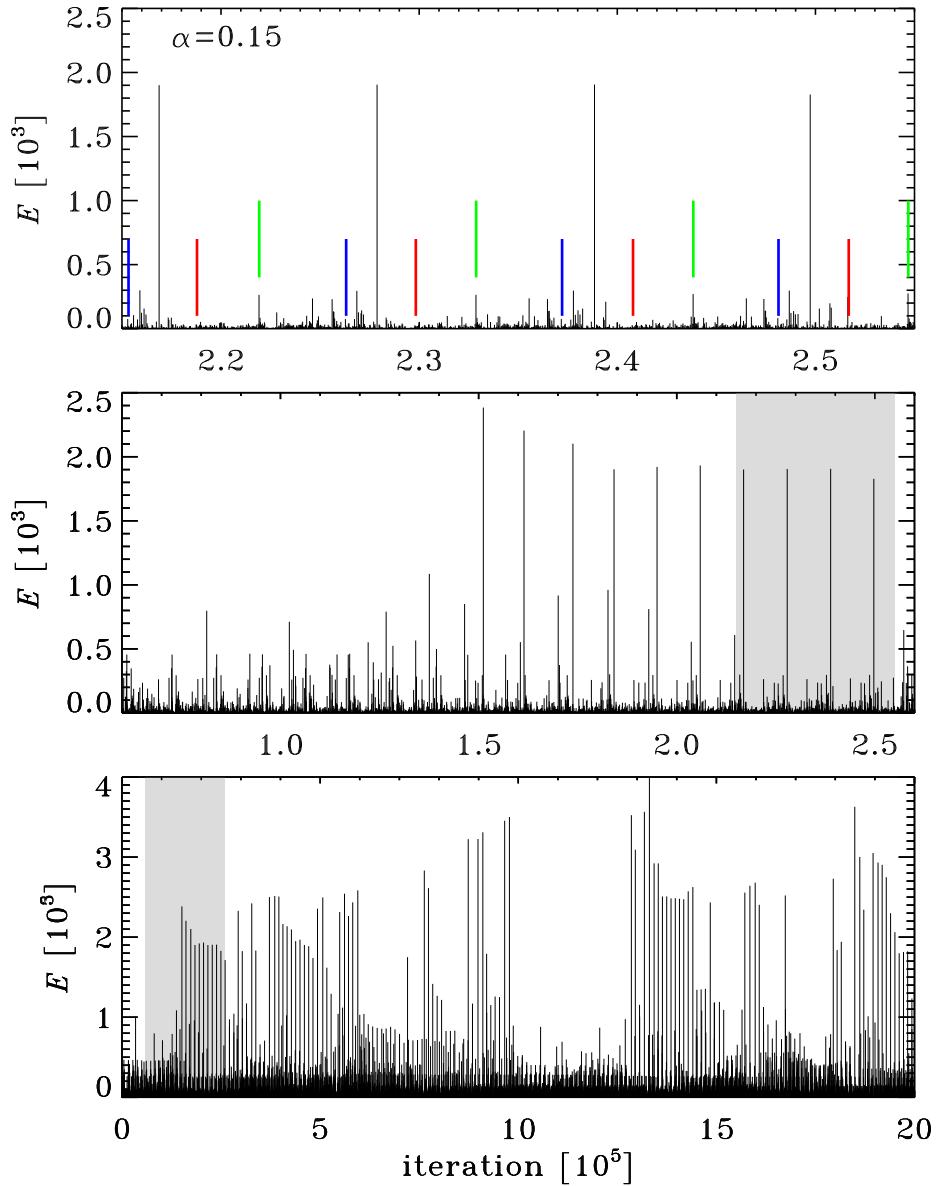


Figure 3.8: Trois portions de la séquence temporelle de l'énergie dissipée par les avalanches dans une simulation sur réseau 128×128 avec paramètre de conservation $\alpha = 0.15$ et forçage $\delta F = 10^{-4}$. Le graphique du haut couvre 4×10^4 itérations, celui du milieu 5 fois plus, et celui du bas un autre 10 fois plus. Les régions ombragées en gris indiquent l'intervalle couvert par le graphique précédent. Les tirets colorés sur le graphique du haut indiquent les itérations où les trois noeuds test de la Figure 3.9 ci-dessous dépassent le seuil d'instabilité. On observe ici une claire périodicité, avec période de $\simeq 10960$ itérations.

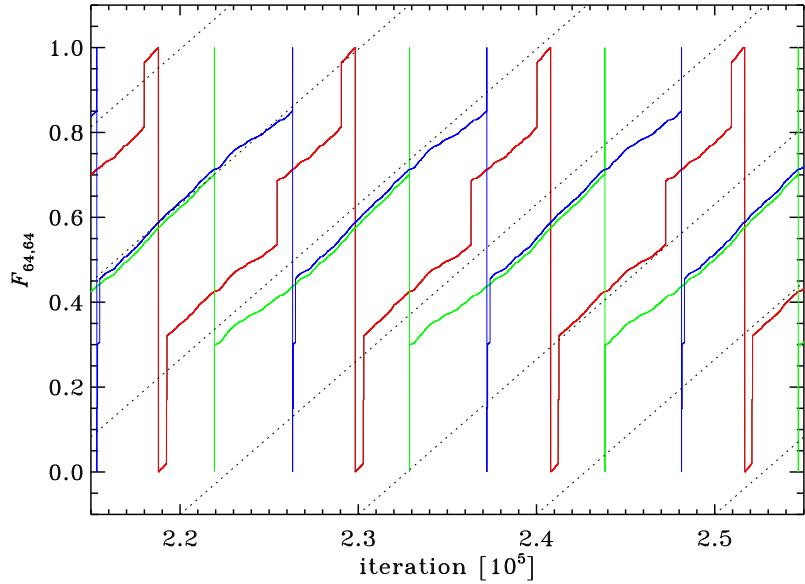


Figure 3.9: Séquences temporelles de $F_{i,j}$ pour trois noeuds à l'intérieur du réseau, extraites de la simulation de la Fig. 3.8 (128×128 , $\delta F = 10^{-4}$, $\alpha = 0.15$). Ces séquences couvrent le même intervalle temporel que le graphique du haut sur la Fig. 3.8. Les noeuds en question sont $(i, j) = (64, 64)$, $(32, 32)$, et $(64, 76)$, en rouge, vert et bleu respectivement. Les droites pointillées indiquent un taux de croissance égal à $\delta F = 10^{-4}$ par itération, corrigé pour le nombre d'itération passées en état d'avalanche.

déclenchant une nouvelle avalanche une fois que les noeuds ont de nouveau atteint le seuil d'instabilité. La lente évolution des tailles et formes de ces domaines résulte en une lente variation du pattern des grandes avalanches quasipériodiques, comme on le voit sur le graphique central de la Fig. 3.8, et est responsable de l'apérioridicité observée sur les très longues échelles temporelles.

Il est facile de comprendre comment un domaine de noeuds déjà “synchronisés” le demeurera durant l'évolution subséquente du système. Comment cette synchronisation s'établit à partir d'une condition initiale aléatoire l'est pas mal moins. Considérons deux noeuds voisins $F_{(1)}, F_{(2)}$ ayant les valeurs

$$F_{(1)}^n = F + \Delta, \quad F_{(2)}^n = F - \Delta, \quad \rightarrow \quad |F_{(2)}^n - F_{(1)}^n| = 2\Delta. \quad (3.16)$$

Si les deux noeuds avalancheont simultanément, alors la règle de redistribution (3.13) ne produira pas des valeurs nulles pour ces deux noeuds, mais plutôt:

$$F_{(1)}^{n+1} = \alpha(F - \Delta), \quad F_{(2)}^{n+1} = \alpha(F + \Delta), \quad \rightarrow \quad |F_{(2)}^{n+1} - F_{(1)}^{n+1}| = 2\alpha\Delta, \quad (3.17)$$

où l'on a encore supposé $F_c = 1$, mais sans perte de généralité. La différence (ou gradient) entre les deux noeuds après la redistribution a donc chuté d'un facteur α (≤ 0.25). On retrouve ici un processus de type diffusif, puisque la quantité transportée du noeud élevé vers le noeud moins élevé, $(1 - \alpha)2\Delta$, est linéairement proportionnelle au “gradient” initial 2Δ ; tout comme le flux diffusif classique. Une fois que les deux noeuds sont synchronisés (dans le sens $F_{(1)}^n = F_{(2)}^n$), d'où $\Delta = 0$ dans l'éq. (3.16)), la redistribution ne peut plus briser cette synchronisation; et le forçage déterministe non plus. La seule option est la déstabilisation d'un troisième noeud voisin transférant un incrément de force à un noeud mais pas à l'autre. Autrement dit, un domaine de noeuds synchrones ne peut évoluer qu'à ses frontières avec un autre domaine de

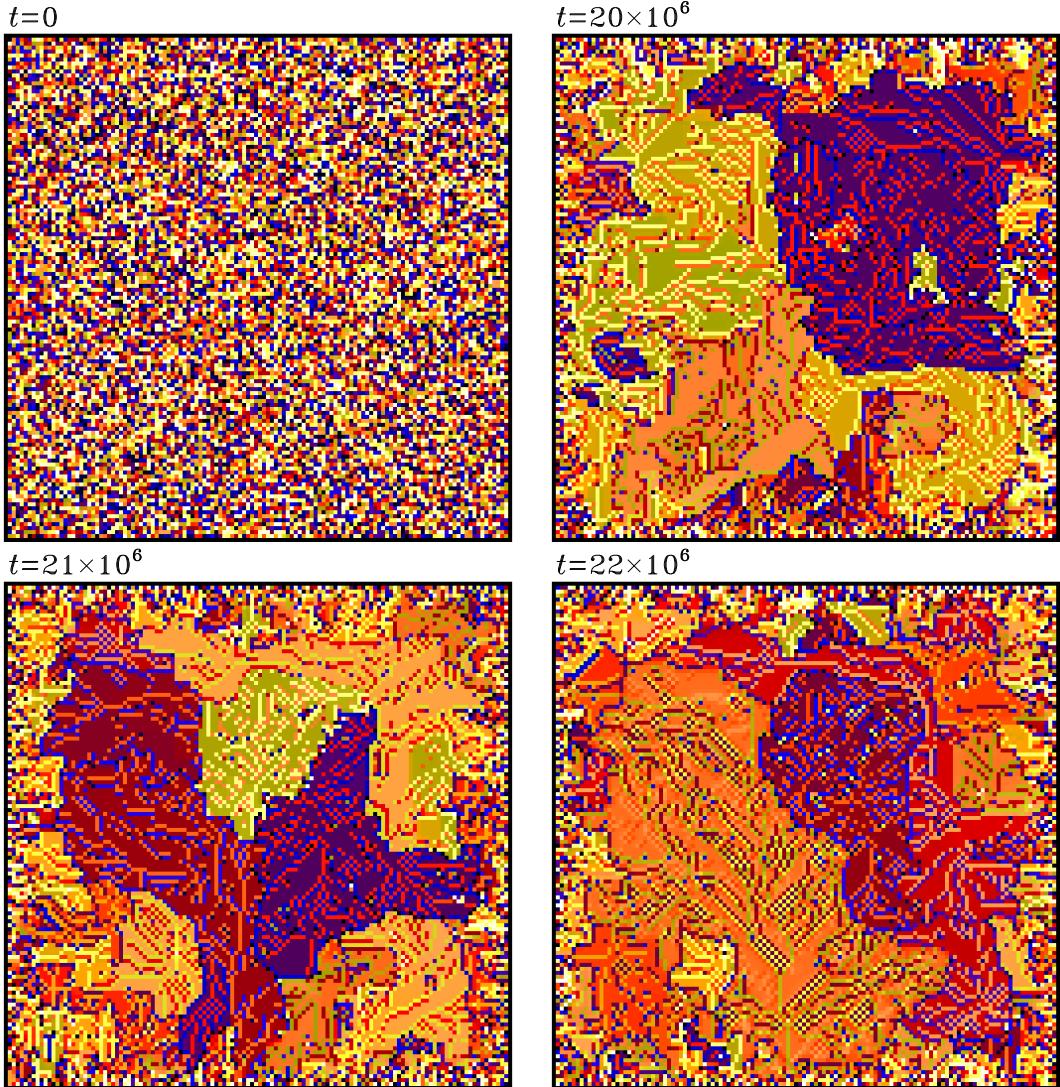


Figure 3.10: Quatre instantanés du réseau OFC, toujours pour la même simulation que sur les Figs. 3.8 et 3.9 (128×128 , $\delta F = 10^{-4}$, $\alpha = 0.15$). L'échelle de couleur encode la valeur de la variable nodale $F_{i,j}$ à chaque noeud. L'Image en haut à gauche est la condition initiale aléatoire, et les trois instantanés suivants sont extrait loin dans la phase statistiquement stationnaire, à une cadence de 10^6 itérations, tel qu'indiqué. On remarquera le développement de “domaines” spatiaux où $F_{i,j}$ est approximativement constant, et dont les frontières évoluent lentement avec le temps.

valeurs nodales distinctes. Plus le domaine est grand, le plus longtemps il peut survivre à la lente érosion de ses frontières d'un cycle de récurrence à l'autre.

3.2.5 Comportement du modèle

Le modèle OFC n'implique que trois paramètres: le paramètre de conservation α (éq. (3.14)), le paramètre de forçage δF (éq. (3.11)), et le seuil d'instabilité F_c (éq. (3.12)). En fait, pour un α et F_c donnés, ce qui importe vraiment est le rapport $\delta F/F_c$; tant que ce dernier est $\ll 1$, correspondant à un forçage lent, le comportement du modèle est essentiellement le même. Le choix de δF impose cependant une échelle temporelle au système puisqu'il détermine le temps d'attente moyen entre deux grandes avalanches successives.

En jouant un peu avec le modèle on réalise rapidement que changer la valeur de α a une forte influence sur la dynamique globale du système, affectant à la fois les tailles et patterns de récurrence des grandes avalanches. Dans le cas limite $\alpha = 0$ où les noeuds sont découplés les uns des autres, chaque noeud croit linéairement de δF par itération à partir de sa valeur initiale aléatoire, et atteint le seuil d'instabilité indépendamment de l'évolution des noeuds voisins, à une cadence de $(\delta F)^{-1}$ itérations. Le réseau est parfaitement périodique, toutes les avalanches sont de taille unitaire (à moins que deux noeuds aient des valeurs initiales aléatoires différant par moins que δF), et la distribution initiale aléatoire est éternellement "gelée" sur le réseau.

Ce n'est plus le cas si $\alpha > 0$ et un couplage se produit entre les noeuds lors d'avalanches. Plus α augmente, plus la période de récurrence diminue, ici de 6435 itérations à $\alpha = 1.0$, 4002 à $\alpha = 1.5$, chutant à 2165 iterations pour $\alpha = 0.2$. En y réfléchissant un peu, c'est une tendance qui s'explique facilement, surtout au regard de notre discussion antérieure de la Fig. 3.9. Plus α est grand, plus la variable nodale saute haut quand un voisin est déstabilisé. Moins d'itérations de forçage sont alors requises pour atteindre le seuil d'instabilité. La quasipériodicité des avalanches, quant à elle, disparaît graduellement quand α approche 0.25, et disparaît complètement à $\alpha = 0.25$.

Il est tout à fait remarquable que malgré la quasipériodicité caractérisant les avalanches dans une large plage de paramètres du modèle, ces avalanches montrent une invariance d'échelle au niveau de leurs tailles. Ceci est démontré par la Fig. 3.11, qui présente les fonctions de densité de probabilité pour des simulations ayant $\alpha = 0.1$, 0.2 et 0.25 . La dernière de celles-ci est conservative, et présente une FDP prenant la forme d'une loi de puissance couvrant ici quatre ordres de grandeurs en taille (E) d'avalanches, avec une pente logarithmique de -1.19 :

$$f(E) = f_0 E^{-\beta} ; \quad (3.18)$$

Par la définition même d'une FDP, la quantité $f(E)dE$ mesure la probabilité d'observer une avalanche de taille dans l'intervalle $[E, E + dE]$. Les simulations non-conservatives ($\alpha < 0.25$) montrent également des FDP en loi de puissance, mais de pente de plus en plus raides et d'étendues de moins en moins large à mesure que α diminue.

Ces tendances viennent du fait que pour qu'un état statistiquement stationnaire puisse être soutenu, la variable nodale doit être soit dissipée localement, soit évacuée aux frontières du réseau, globalement au même taux auquel elle croît en réponse au forçage. Pour un taux de forçage fixe, un bas taux de dissipation interne ($\alpha \rightarrow 0.25$) exige donc qu'un plus grand nombre d'avalanches atteignent les frontières du réseau, tandis que pour un taux de dissipation plus élevés les avalanches peuvent plus facilement s'arrêter quelque part à l'intérieur du réseau. Les grandes avalanches doivent donc être plus fréquentes dans la limite $\alpha \rightarrow 0.25$, ce qui se traduit en une distribution de tailles en loi de puissance plus aplatie, en accord avec la Fig. 3.11.

La valeur numérique de l'indice β a des conséquences qui sont loin d'être triviales. L'énergie totale libérée par des avalanches (\mathcal{E}) se distribuant selon l'éq. (3.18) sera donnée par

$$\mathcal{E} = \int_{E_{\min}}^{E_{\max}} f(E) E dE = f_0 \int_{E_{\min}}^{E_{\max}} E^{1-\beta} dE = f_0 \left[\frac{E^{2-\beta}}{2-\beta} \right]_{E_{\min}}^{E_{\max}}, \quad (3.19)$$

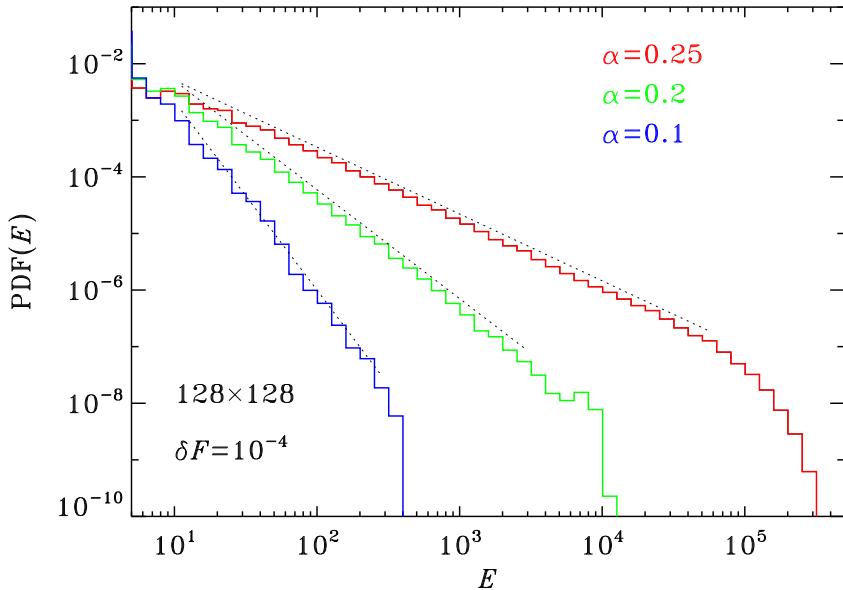


Figure 3.11: Fonction de densité de probabilité de la taille des avalanches dans des simulations OFC pour trois valeurs du paramètre de conservation α , tel qu’indiqué par le code couleur. Toutes les distributions sont bien décrisées par une loi de puissance sur une large plage de tailles, avec la pente logarithmique s’adoucissant quand α augmente: -3.34 à $\alpha = 0.1$, -1.92 à $\alpha = 0.2$, jusqu’à -1.18 à $\alpha = 0.25$. Les segments de droites pointillées indiquent l’étendue de chaque distribution sur laquelle est calculée la pente logarithmique. Les trois simulations sont effectuées sur un réseau 128×128 avec $\delta F = 10^{-4}$ et $F_c = 1$, et les distributions sont bâties à partir de séquences temporelles longues de 5×10^6 itération, durant lesquelles plus de 10^6 avalanches ont eu lieu.

où E_{\min} et E_{\max} sont les tailles minimale et maximale des avalanches possibles dans le système (ici 1 et N^2 , respectivement). Typiquement, on aura $E_{\min} \ll E_{\max}$, et donc

$$\mathcal{E} = \begin{cases} \frac{f_0 E_{\max}^{2-\beta}}{2-\beta}, & \beta < 2, \\ \frac{f_0 E_{\min}^{2-\beta}}{\beta-2}, & \beta > 2. \end{cases} \quad (3.20)$$

Autrement dit, les très nombreuses petites avalanches dominent la dynamique si $\beta > 2$, mais dans le cas contraire la dynamique (plus spécifiquement ici, la relaxation des stress tectoniques), est dominée par les très rares grandes avalanches.

La loi (de puissance) de Gutenberg-Richter est décrite par une pente logarithmique $b \sim -1$ dans la distribution *cumulative*, ce qui implique une loi de puissance d’indice $\simeq -2$ pour une FDP du genre de celles portées en graphique sur la Fig. 3.11. Si on prend cette Figure au mot, on en concluerait que le modèle OFC nous indique que la tectonique des plaques opère en mode non-conservatif, avec α quelque part dans l’intervalle 1.5–2.0. C’est la partie de l’espace des paramètres où les plus grandes avalanches dominent encore la dynamique, et où la récurrence quasipériodique des avalanche se développe de manière marquée. Cette possibilité trouve un certain support dans les données séismiques, qui indiquent que certaines failles, comme très fameuse faille San Andreas en Californie, produisent des tremblements de terre d’une taille “caractéristique” à intervalles passablement réguliers.

Une telle quasipériodicité est évidemment très intéressante pour la prédiction des tremblements de terre. Mais attention, il s’agit ici d’un jeu très dangereux. Une équipe de séismologues gouvernementaux italiens l’a réalisé à la dure, quand ils se sont retrouvés traduits en justice et

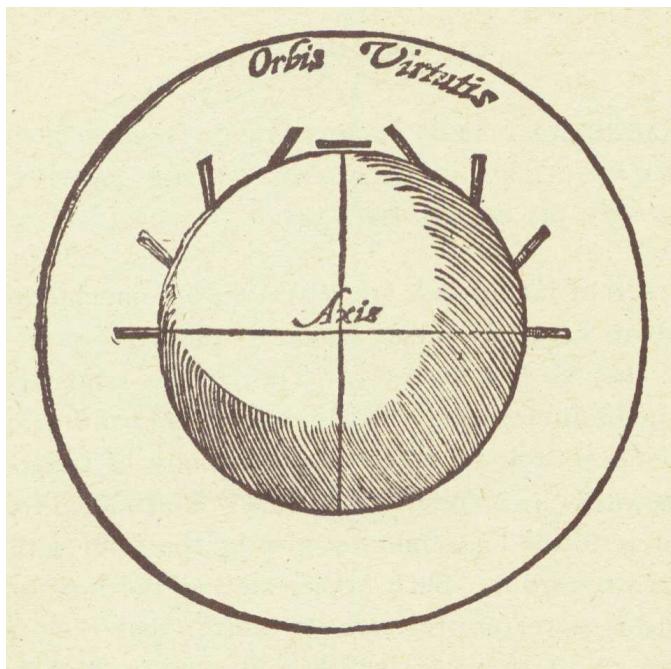


Figure 3.12: Inclinaison d'aiguilles métalliques placées à la surface d'un aimant taillé en forme de sphère, avec l'axe magnétique horizontal ici. Le pattern d'inclinaison des aiguilles étant le même que celui observé pour la déclinaison magnétique géographique, Gilbert en conclut que la Terre est un gigantesque aimant. Reproduit de la traduction anglaise de 1892 par P. Fleury Mottelay, Edward Brothers Inc. (p. 285).

déclarés coupables d'homicides involontaires multiples et condamnés à six ans de prison pour ne pas avoir réussi à prédire le tremblement de terre qui a rasé le village de L'Aquila le 6 avril 2009 (309 morts); ce n'est que 5 ans plus tard que ce verdict de culpabilité fut finalement renversé par une cour d'appel!

3.3 Le ferromagnétisme

Les minéraux magnétiques et les substances magnétisables (ou pas) sont connus depuis l'antiquité. Un aspect du magnétisme particulièrement troublant pour les Grands Penseurs du temps était “l'action à distance” qui caractérise l'attraction ou la répulsion entre les pôles de deux aimants. Ceci leur a taillé une place de choix dans les grands systèmes philosophiques “alternatifs” de tous acabit, ainsi que dans les croyances populaires sur les effets spirituels et physiologiques des aimants.

Le grand ménage dans tout ça est habituellement attribué à William Gilbert, qui en 1600 a publié un ouvrage intitulé *De Magnete*, visant à démêler les faits établis scientifiquement des superstitions relatives au magnétisme. Dans cet ouvrage, Gilbert établit (entre autre) sur la base de mesures géographiques de la déclinaison magnétique, que la Terre elle-même est un gigantesque aimant, avec les pôles approximativement alignés avec son axe de rotation (voir Fig. 3.12). À peu près à la même époque, et à la recherche d'un mécanisme physique pouvant expliquer les mouvements des planètes, Johannes Kepler suggérait que le Soleil est lui aussi un aimant, et que c'est l'extension dans le milieu interplanétaire de ce champ magnétique du soleil (en rotation) qui entraîne les planètes dans leur mouvement orbital¹.

¹Kepler avait raison sur le fait que le soleil est magnétisé, et qu'une force Soleil-Terre est nécessaire pour expliquer le mouvement orbital; il n'avait simplement pas identifié la bonne force !

L'étude expérimentale du magnétisme s'est accélérée au dix-neuvième siècle, avec la découverte par Hans Christian Oersted et les travaux subséquents de Louis-Marie Ampère qu'un courant électrique peut produire un champ magnétique, le tout culminant avec les travaux de Michael Faraday et sa découverte du processus inverse, soit l'induction magnétique; l'unification de l'électricité et du magnétisme par James Clerk Maxwell venant couronner le tout sur le plan théorique, la relativité n'y changeant d'ailleurs absolument rien.

L'explication contemporaine du ferromagnétisme (aimants permanents) et du paramagnétisme (substances polarisables magnétiquement) trouve son origine dans la mécanique quantique de l'atome, plus précisément dans les configurations orbitales montrant un moment dipolaire magnétique produit par le spin d'un électron non-pairé. À l'échelle globale du système, le magnétisme est produit par l'alignement de ces moments magnétiques atomiques, qui peuvent se retrouver "figés" lors de la solidification d'un fluide (ferromagnétisme), ou s'aligner sous l'influence d'un champ magnétique imposé de l'extérieur (paramagnétisme).

3.4 Le modèle d'Ising

Développé en 1925 par Ernst Ising, alors étudiant au doctorat sous la supervision de Wilhelm Lenz, le modèle statistique maintenant dit d'Ising est devenu un chouchou de la physique statistique, particulièrement dans le contexte des transitions de phase. Le modèle original avait en fait été développé pour étudier la transition de phase se produisant à la température dite de Curie, sous laquelle un métal passe "spontanément" de paramagnétique à ferromagnétique.

On ne considère que deux états possibles de spin s pour chaque atome: \uparrow et \downarrow , auxquels on assigne les valeurs numériques $+1$ et -1 , respectivement. Deux spins i, j voisins ont une énergie d'interaction $= -J s_i s_j$, où $J (> 0$ dans tout ce qui suit) est la constante d'interaction spin-spin. L'énergie d'interaction est donc limitée à quelques valeurs: $\uparrow\uparrow \equiv (+1)(+1) = -J$; $\uparrow\downarrow \equiv (+1)(-1) = +J$; $\downarrow\uparrow \equiv (-1)(+1) = +J$; et $\downarrow\downarrow \equiv (-1)(-1) = -J$. Les état d'énergie minimaux sont donc $\uparrow\uparrow$ et $\downarrow\downarrow$, soit des spins alignés.

Si le spin i spin interagit avec un nombre nn de spins voisins équidistants, l'énergie d'interaction est donnée par

$$E_i = -J \sum_{j=1}^{nn} s_i s_j , \quad (3.21)$$

où la somme n'implique que les voisins immédiats du spin i , et on supposera que J est identique pour toutes les paires de spins.

L'idée centrale du modèle d'Ising est que chaque spin se renverse "spontanément" si cette inversion diminue l'énergie d'interaction, comme on s'y attendrait dans un système fermé tendant inexorablement vers son état d'énergie minimale. La moitié supérieure de la Figure 3.13 illustre l'idée, pour un réseau 2D cartésien avec connectivité de von Neumann (4 voisins). C'est le genre de réseau et connectivité qui seront utilisés dans tout ce qui suit. Ici l'inversion du spin en rouge est permise car elle réduit l'énergie d'interaction avec les quatre voisins de $+2J$ à $-2J$; la transition inverse (de la gauche vers la droite) ne l'est cependant pas. La définition de cette règle d'évolution sur un réseau est la simplicité même:

- À chaque itération temporelle et pour chaque spin i , calculer l'énergie d'interaction E , ainsi que celle associée à la configuration où le spin i est inversé (E'); si $E' \leq E$, inverser le spin i , sinon il conserve son orientation.

Une simulation du modèle d'Ising est typiquement initialisée sous la forme d'une distribution aléatoire de spins, et la règle ci-dessus est itérée temporellement jusqu'à ce que le système atteigne (en principe) une configuration stable. Pour un réseau $N \times N$ où chaque noeud peut prendre l'une de deux valeurs, \uparrow ou \downarrow , il existe 2^{N^2} états distincts, dont seulement deux (\uparrow partout ou \downarrow partout) qui minimisent globalement l'énergie. Le défi consiste donc à atteindre

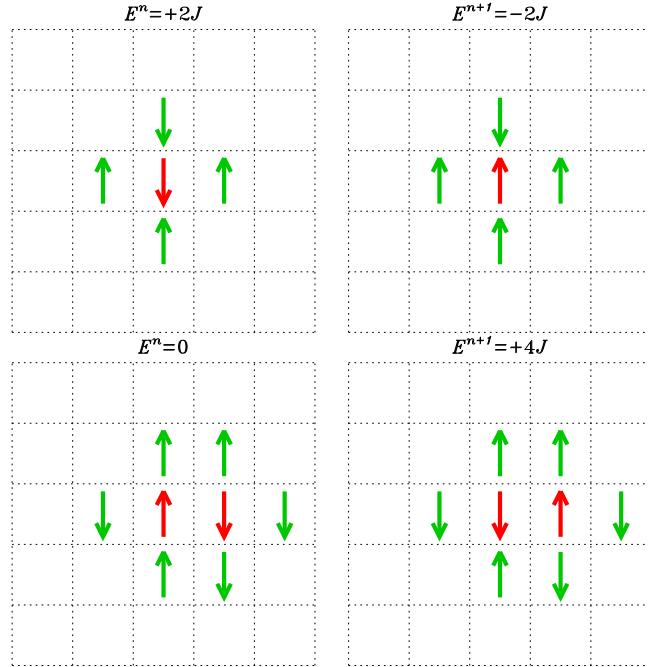


Figure 3.13: Exemple d’application de la règle d’inversion des spins dans le modèle d’Ising en 2D. Les deux diagrammes du haut illustrent une situation où l’inversion du spin central (en rouge) réduit l’énergie d’interaction de $+2J$ à $-2J$; les deux diagrammes du bas montrent une situation pathologique où l’inversion synchrone des deux spins voisins (en rouge) ferait augmenter l’énergie d’interaction totale des deux spins de 0 à $+4J$, une situation à éviter (voir texte).

un de ces deux états, à partir d’une condition initiale aléatoire. Pas du tout évident, comme on le verra sous peu.

En présence d’un champ magnétique imposé de l’extérieur, il faut aussi prendre en considération l’énergie d’interaction du spin avec ce champ; l’équation (3.21) doit alors être remplacée par:

$$E_i = -J \sum_{j=1}^{nn} s_i s_j - H s_i , \quad (3.22)$$

où $H = \mu_0 B$. Le membre de droite de cette expression est effectivement l’Hamiltonien du système. Clairement, si $|H| > 4J$ tous les spins s’aligneront contre le champ extérieur, les énergies d’interaction spin-spin étant insuffisantes pour soutenir un désalignement par rapport au champ appliqué. Pour des H plus faible, le comportement est plus intéressant, comme aurons l’occasion de l’explorer plus loin, ainsi que dans le cadre du projet.

3.4.1 Traitement numérique

La dynamique très simplifiée associée à la règle d’évolution du modèle d’Ising peut conduire à des situations problématiques, comme l’illustrent les deux diagrammes au bas de la Figure 3.13. Ici les deux spin rouges à gauche ont chacun deux voisins \uparrow et deux \downarrow , pour une énergie d’interaction $E = 0$ dans les deux cas; la règle ci-dessus, appliquée de manière synchrone, inversera les deux spins, conduisant maintenant à une énergie d’interaction $+4J$ pour chacun; l’énergie a *augmenté* au lieu de diminuer ! La mise à jour synchrone des noeuds du réseau nous joue ici un mauvais tour.

La solution à ce problème consiste à “diviser” les noeuds du réseau en deux groupes positionnés comme les cases blanches et noires sur un échiquier. On calcule d’abord les inversions devant se produire pour les spins situés sur les cases blanches, on les effectue à mesure, ensuite on répète pour les spins sur les cases noires; le fragment de code Python listé à la Figure 3.14 montre comment faire ça en pratique. Le truc est de balayer sur le second indice du tableau `spin[i,j]` par incrément de 2, en alternant la première valeur de l’indice j entre 1 et 2 en fonction de la valeur (paire ou impaire) de l’indice i ; c’est le rôle des deux instructions utilisant l’opérateur modulus (“%”) pour calculer l’indice de départ `ij` pour la boucle sur les j (lignes 19 et 28). Notons également que la mise à jour de `spin[i,j]` peut maintenant se faire immédiatement à l’intérieur de ces boucles, puisqu’il n’y a plus de conflit possible entre un noeud (i,j) et ses quatre voisins.

Mis à part cette différence dans le test et la mise à jour de la variable nodale, la simulation est structurée comme celle du modèle OFC des tremblements de terre (Fig. 3.7): boucle temporelle inconditionnelle sur `nIter` itérations, initialisation aléatoire ($s_i^0 = \pm 1$ équiprobable), et définition d’un cadre de noeuds fantômes pour imposer les conditions limites, ici périodiques verticalement et horizontalement (voir la Fig. 3.3).

Les deux quantités primaires calculables à partir des sorties des simulations sont, à chaque itération temporelle, l’énergie moyenne par spin:

$$e = \frac{1}{N^2} \sum_i \left(-J \sum_{j=1}^{nn} s_i s_j - H s_i \right) , \quad (3.23)$$

et la magnétisation moyenne par spin:

$$m = \frac{1}{N^2} \sum_i s_i . \quad (3.24)$$

En l’absence d’un champ extérieur imposé ($H = 0$), il existe deux états (dégénérés) d’énergie minimale pour le réseau dans son ensemble: tous les spins \uparrow ou tous \downarrow , conduisant à $e = -4J$ et $m = \pm 1$. Rouler le modèle sous cette forme révèle rapidement que le système n’atteint jamais l’un ou l’autre de ces états d’énergie minimale; on stabilise plutôt dans un état caractérisé par une énergie par spin substantiellement inférieure à $-4J$, et une magnétisation tout aussi substantiellement inférieure à $|m| = 1$. Pour une condition initiale (aléatoire) donnée, il n’existe pas nécessairement une trajectoire dans l’espace de phase reliant la condition initiale à un des deux états d’énergie minimale, le long de laquelle l’énergie diminue de manière monotone. Autrement dit, le système atteint un minimum secondaire en énergie, et ne peut simplement pas s’en décoincer.

3.4.2 L’algorithme de Metropolis

La règle d’évolution introduite ci-dessus pour le modèle d’Ising est complètement déterministe: toujours réduire l’énergie. À température finie cependant, il est possible qu’un apport localisé d’énergie associée à une perturbation thermique permette occasionnellement une transition “spontanée” vers un état d’énergie plus élevée, l’effet devenant plus important le plus la température est élevée. Cette idée est à la base de l’*algorithme de Metropolis*. Une inversion du spin i peut se produire avec probabilité

$$p = \min \left[1, \exp(-\Delta E / kT) \right] , \quad (3.25)$$

où $\Delta E = E' - E$, k est (en principe) la constante de Boltzmann, et T la température. Notons déjà que si $\Delta E \leq 0$, $p = 1$ et donc le spin s’inversera à tous les coups, comme dans la version déterministe du modèle d’Ising. Mais si $\Delta E > 0$, on aura une probabilité finie qu’un spin s’inverse même si cela augmente l’énergie du système, cette probabilité tendant vers zéro dans la limite $T \rightarrow 0$. D'où vient cette énergie? tout simplement du bain de chaleur qui maintient la température à sa valeur constante T .

```

1 # MODELE D'ISING SUR RESEAU 2D AVEC CONNECTIVITE VON NEUMANN (4 VOISINS)
2 import numpy as np
3 #-----
4 N      =32                      # taille du reseau
5 nIter =1000                     # nombre d'iterations temporelles
6 ...
7 #-----
8 spin=np.zeros([N+2,N+2])        # le reseau des spins
9 ...
10 for i in range(1,N+1):          # autres definitions de tableaux
11     for j in range(1,N+1):       # initialisation aleatoire [-1,1]
12         spin[i,j]=-1+2*(np.random.random_integers(0,1))
13     ...
14             # periodicite sur condition initiale
15 for iter in range(0,nIter):     # boucle temporelle
16
17     # balayage des noeuds blancs du reseau
18     for i in range(1,N+1):
19         ij= (i % 2)+1           # on alterne le noeud de depart
20         for j in range(ij,N+1,2): # chaque deux noeuds
21             ...
22             ...
23             ...
24         # calcul de E, E'
25         ...
26         ...
27         ...
28         ...
29         ...
30         ...
31         ...
32         ...
33     # test probabiliste (Metropolis)
34         ...
35         ...
36         ...
37     # mise a jour de spin[i,j]
38
39     # fin du balayage sur les noeuds blancs
40
41     # balayage des noeuds noirs du reseau
42     for i in range(1,N+1):
43         ij= ((i+1) % 2)+1       # on alterne le noeud de depart
44         for j in range(ij,N+1,2): # chaque deux noeuds
45             ...
46             ...
47             ...
48         # calcul de E, E'
49         ...
50         ...
51         ...
52         ...
53     # test probabiliste (Metropolis)
54         ...
55         ...
56         ...
57     # mise a jour de spin[i,j]
58
59     # fin du balayage sur les noeuds noirs
60
61     # conditions limites periodiques
62
63     ...
64
65     # fin boucle temporelle
66 # END

```

Figure 3.14: Fragment de code Python pour effectuer la mise à jour du réseau de manière séquentielle sur les deux familles de noeuds “blancs” et “noirs”, disposés relativement les uns par rapport aux autres comme les cases sur un échiquier. Notez la taille $(N + 2) \times (N + 2)$ dans la définition du tableau `spin`, afin d’y inclure un cadre de noeuds fantômes, et ne pas oublier que la fonction Python `range(A,B)` contrôlant les boucles inconditionnelles itère de l’indice A jusqu’à la *position* B, soit l’indice B-1 (ackpht...).

Cette simple modification permet à notre simulation de s'extraire des minima secondaires dans lesquelles elle se serait sinon coincée durant sa quête de l'énergie minimale pour l'ensemble du réseau.

3.4.3 Résultats représentatifs

Il est temps d'examiner quelques solutions. On pose, sans perte de généralité, $J = 1$, $H = 0$ et $k = 1$ (désolé Herr Prof. Dr. Boltzmann...) pour les résultats numériques présentés dans ce qui suit.

La Figure 3.15 montre la variation temporelle de l'énergie totale par spin sur réseau $N \times N = 64 \times 64$ pour trois simulations débutant de la même condition initiale aléatoire, pour des températures $T = 1.5$, 2.35 et 5.0 , tel qu'indiqué par le code couleur. Aucun champ extérieur n'est appliqué ici, i.e., $H = 0$ dans les éqs. (3.22) et (3.23). L'énergie décroît comme il se doit, se stabilisant à une valeur qui augmente avec la température, comme on aurait pu s'y attendre puisqu'une température plus grande tend à produire plus d'inversions de spin, augmentant ainsi l'énergie d'interaction. Ici les énergies se stabilisent en une cinquantaine d'itérations temporelles, fluctuant par la suite. Cependant, comme le montre la Fig. 3.16, la magnétisation ne se stabilise pas nécessairement aussi vite; elle le fait ici à $T = 1.5$ et $T = 3.5$, les états résultants étant ferromagnétiques et non-ferromagnétiques, respectivement. Le comportement à $T = 2.35$ est qualitativement différent; bien que l'énergie semble s'être stabilisée, la magnétisation ne l'est pas, sa variation temporelle ressemblant à une marche aléatoire. Ces variations indiquent un changement substantiel et continu dans le pattern d'alignement des spins sur le réseau; comment l'énergie peut-être demeurer stable sous ces conditions ?

La Figure 3.17 montre les configurations du réseau à une itération temporelle donnée (et arbitrairement choisie à 5000 itérations, soit bien au delà de la relaxation initiale de l'énergie totale) pour les trois simulations des Figs. 3.15 et 3.16. À $T = 1.5$ l'ensemble du réseau est magnétisé positivement, les quelques pixels noirs correspondant à des inversions ponctuelles de spin associées à la température finie du système. Désaligner un spin entouré de 4 voisins alignés représente un $\Delta e = +8J$, ce qui se produira, selon l'algorithme de Metropolis, avec une probabilité donnée par $\exp(-\Delta E/T) \simeq 0.005$ ici; sur un réseau de $64 \times 64 = 4096$ spins, on s'attendrait donc, à n'importe quelle itération, à avoir $0.005 \times 64^2 \simeq 21$ spins désalignés; on en a 26 ici, ce qui conforme aux attentes considérant la nature stochastique de ces inversions.

À l'autre extrême, $T = 5$, le désalignement d'un spin aligné se produit avec probabilité $p = 0.2$ à chaque itération. On est presque en régime aléatoire ici. La configuration au bas de la Fig. 3.17 semble aléatoire au premier coup d'œil, mais un examen approfondi révèle qu'en moyenne, un spin positif à une plus grande probabilité d'avoir un spin positif que négatif comme voisin (et vice versa). Autrement dit, le système est désordonné à l'échelle du réseau, mais il existe une échelle de coupure sous laquelle le système demeure localement ordonné.

À $t = 2.35$ le système est caractérisé par la présence de *domaines de magnétisation* soit positive ou négative. Les plus grands de ces domaines couvrent la dimension linéaire du réseau, mais contiennent également des domaines de signes opposés présentant une variété de tailles. De plus, les tailles et formes de ces domaines évoluent lentement, sans jamais véritablement se stabiliser, et avec une polarité prenant parfois le dessus sur l'autre durant des périodes prolongées. C'est ce qui explique la variations erratique de la magnétisation observée à cette température sur la Fig. 3.16 (courbe verte).

On est ici dans une situation où, à certaine température, l'état du réseau montre des corrélations spatiotemporelles de longue portée, en raison de l'existence de ces domaines; autrement dit, l'état moyen du réseau à l'itération n est fortement dépendant de l'historique du système, i.e., de son évolution antérieure, remontant jusqu'à la condition initiale (aléatoire!) de la simulation. Dans une telle situation, le calcul des propriétés moyennes du réseau doit se faire par *moyenne d'ensemble*.

La Figure 3.18 montre les variations de la magnétisation produite dans un tel ensemble de simulations sur réseau maintenant 32×32 , mais avec $J = 1$ et $H = 0$ comme auparavant. À chaque température, 100 simulations de 1000 itérations sont calculées, chacune débutant d'une

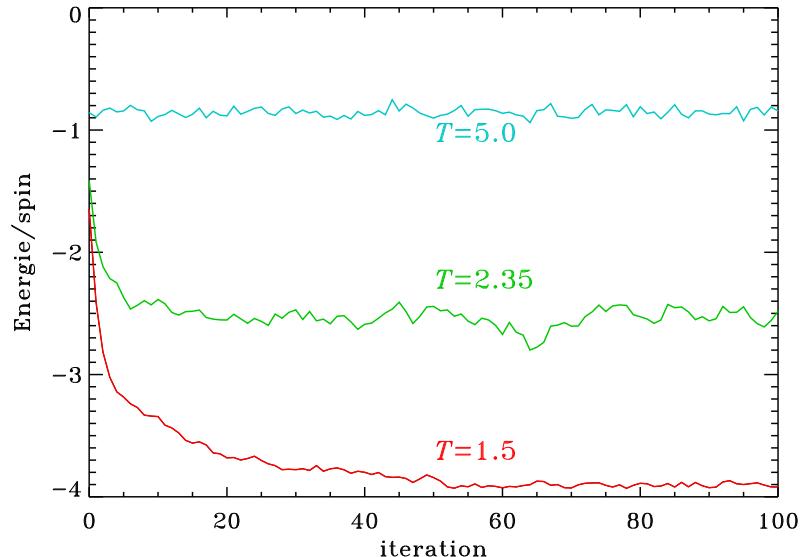


Figure 3.15: Séquences temporelles de l'énergie moyenne par spin dans trois simulations sur réseau 64×64 avec $J = 1$ et $H = 0$, pour trois températures différentes, tel qu'indiqué. Un réseau où tous les spins sont parfaitement alignés aurait une énergie par spin = -4 ici.

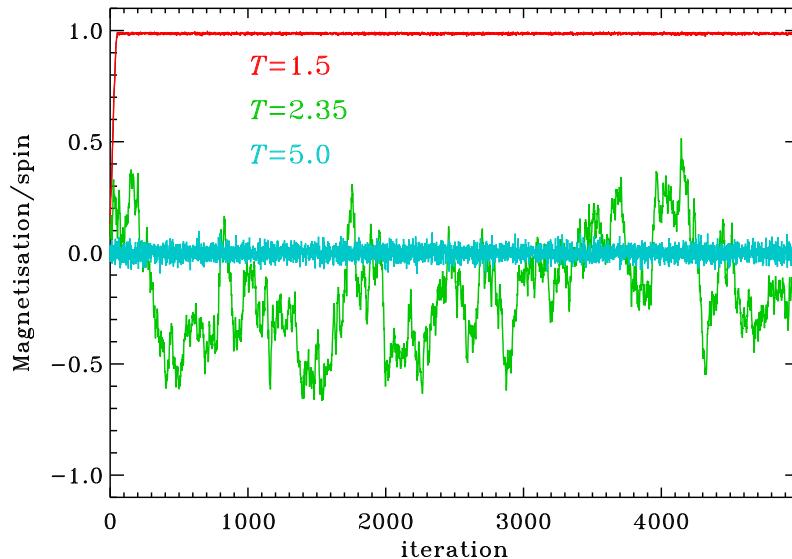


Figure 3.16: Séquences temporelles de la magnétisation m pour les trois même simulations que ci-dessus. Notez l'échelle temporelle poussée ici à 5000 itérations, versus seulement 100 sur la Fig. 3.15.

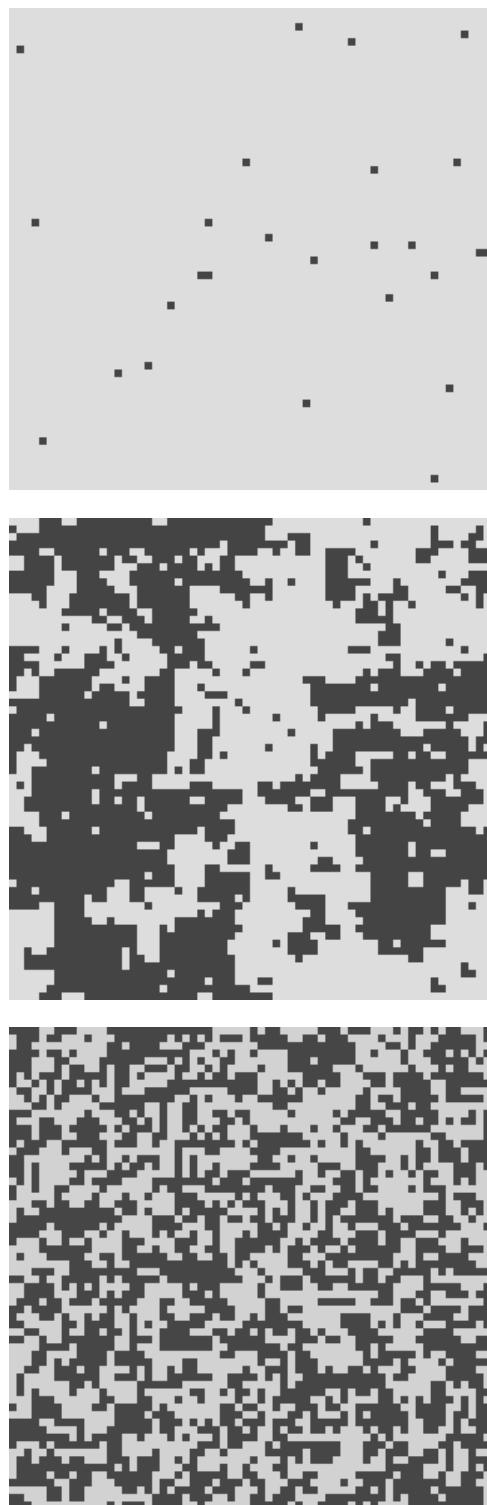


Figure 3.17: Distribution spatiale des spins dans des simulations du modèle d’Ising sur réseau 64×64 , pour $T = 1.5, 2.325$ et 5.0 , de haut en bas. Les spins $\uparrow (+1)$ sont en gris, et les $\downarrow (-1)$ en noir. Des animations de ces solutions sont disponibles sur la page web du cours.

réalisation distincte de la condition initiale aléatoire. Pour $T \lesssim 2$, les simulations se retrouvent fortement ferromagnétiques (presque tous les spins alignés) soit positivement ou négativement, de manière équiprobable; laquelle des deux polarités devenant dominante étant déterminé par le détail de la condition initiale (aléatoire!)². La magnétisation décroît ensuite rapidement dans l'intervalle $2.2 \lesssim T \lesssim 2.5$, pour tendre vers zéro dans tous les cas pour $T \gtrsim 3$. La grande dispersion de la magnétisation dans l'intervalle $2.2 \lesssim T \lesssim 2.5$ n'est *pas* causée par une durée trop courte des simulations; on est ici dans le régime de température où la magnétisation ne se stabilise jamais (cf. courbe verte sur la Fig. 3.16).

3.4.4 La température de Curie

Que la magnétisation disparaisse quand T devient suffisamment élevé, soit; on pouvait s'y attendre considérant son effet stochastiquement perturbateur sur l'alignement mutuel des spins voisins. Mais l'aspect abrupt de cette transition est plus surprenant. La Figure 3.19 représente différemment ces mêmes résultats, cette fois sous la forme d'une moyenne d'ensemble de la valeur absolue de la magnétisation à chaque valeur de température:

$$\langle |m| \rangle = \frac{1}{100} \sum_{k=1}^{100} |m_k| , \quad (3.26)$$

où m_k est la magnétisation du k^{eme} membre de l'ensemble. Ces moyennes d'ensemble à chaque température sont indiquées par les points noirs sur la Fig. 3.19. Les barres verticales donnant l'intervalle $\pm\sigma$, la déviation standard σ étant définie de la manière habituelle:

$$\sigma^2 = \frac{1}{100} \sum_{k=1}^{100} \left(m_k^2 - \langle |m| \rangle^2 \right) . \quad (3.27)$$

On voit maintenant bien la rapidité de la transition de $\langle |m| \rangle \simeq 1$ à $\langle |m| \rangle \simeq 0$, se produisant à la *Température de Curie* (T_c), indiquée par le trait vertical en tirets sur la Fig. 3.19. La valeur de cette température critique peut être déterminée de différentes manières; ici elle correspond à la température pour laquelle $\langle |m| \rangle = 0.5$, ou encore où la dérivée $|d\langle |m| \rangle/dT|$ est maximale. Pour un réseau cartésien 2D à connectivité 4-voisins et de taille $N \rightarrow \infty$, on peut montrer que la valeur de la température de Curie est donnée par l'expression

$$\lim_{N \rightarrow \infty} T_c = \frac{2}{\ln(1 + \sqrt{2})} \simeq 2.269185... \quad (3.28)$$

On s'en tire déjà pas trop mal sur la Fig. 3.19, avec seulement $N = 32$!

Avec ces résultats en main, on peut calculer à partir du modèle plusieurs quantités physiques d'intérêt. La capacité thermique du système peut se calculer directement à partir de la Fig. 3.19 selon:

$$c(m, H) = \left(\frac{\partial e}{\partial T} \right)_H , \quad (3.29)$$

tandis qu'en présence d'un champ extérieur H , la *susceptibilité magnétique* est définie comme:

$$\chi(m, H) = \left(\frac{\partial m}{\partial H} \right)_T . \quad (3.30)$$

On ne le démontrera pas ici (voir l'ouvrage de Christensen & Moloney cité en fin de chapitre), mais ces quantités peuvent aussi être obtenues via les mesures des fluctuations de e et m selon:

$$c(m, H) = \frac{1}{n} \frac{1}{kT^2} \left(\langle e^2 \rangle - \langle e \rangle^2 \right) , \quad (3.31)$$

²On remarquera sur la Fig. 3.18 qu'à $T = 1.5$, deux simulations n'ont pas atteint un état de fort alignement; elles sont "coincées" dans un minimum secondaire de l'énergie, et ne peuvent s'en sortir que suite à une fluctuation thermique, peu probables ici en raison de la (relativement) faible température; il aurait fallu pousser cette simulation plus loin dans le temps.

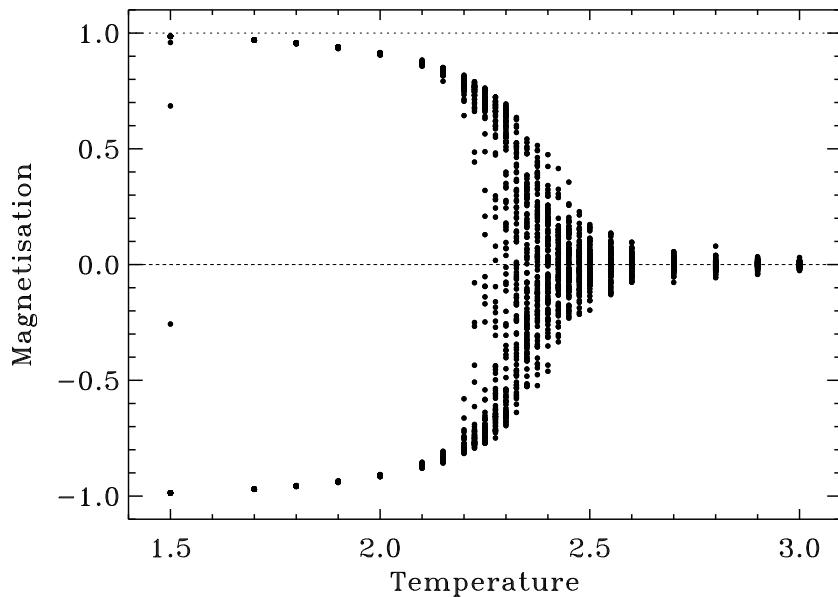


Figure 3.18: Magnétisation produite au bout de 1000 itérations temporelles, mesurée dans un ensemble de 100 simulations à chaque valeur de la température. Réseau 32×32 avec $J = 1$ et $H = 0$ dans tous les cas. Chaque point correspond à la moyenne de m sur les dernières 500 itération de chaque simulation.

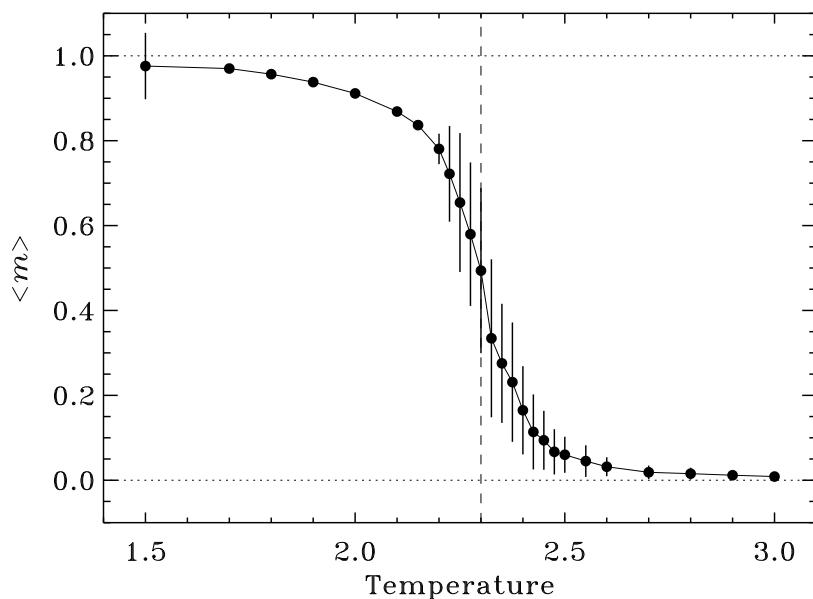


Figure 3.19: Transition de phase dans le modèle d'Ising. Chaque point représente maintenant une moyenne d'ensemble de la valeur absolue de la magnétisation, effectuée sur les 100 réalisations indépendantes d'une simulation à une température T donnée comme ci-dessus. Les barres verticales indiquent la déviation standard $\pm\sigma$ par rapport à cette magnétisation moyenne. Le trait vertical en tirets indique (approximativement) la température de Curie (voir texte).

$$\chi(m, H) = \frac{1}{n} \frac{1}{kT} \left(\langle m^2 \rangle - \langle m \rangle^2 \right), \quad (3.32)$$

où n est ici le nombre de “mesures” de m ou e .

Le passage du paramagnétisme ($T > T_c$) au ferromagnétisme ($T < T_c$), dans le modèle d’Ising comme dans la réalité, représente un exemple d’une *transition de phase*.

3.4.5 Magnétisation extérieure

Il est plus que temps d’introduire la possibilité d’une magnétisation imposée extérieurement, i.e., $H \neq 0$ dans l’éq. (3.22). Sous la température de Curie, même un champ extérieur très faible peu avoir un fort impact sur la magnétisation finale atteinte par le réseau. En effet, le moindre champ non-nul lève l’ambiguité caractérisant la règle d’évolution du modèle quand un spin est entouré de deux voisins \uparrow et deux \downarrow ; l’orientation finale, dans ce cas ambigu lorsque $H = 0$, maintenant sera toujours celle du champ extérieur. Il est facile de vérifier que pour toute température finie, à partir d’une condition initiale aléatoire la polarité de la magnétisation finale sera pratiquement toujours la même que celle de H .

La chose devient plus intéressante —et complexe— lorsqu’on examine le comportement du système à des faibles niveaux de magnétisation (dans le sens $|H| < 4$) mais à températures T pas trop loin de la température de Curie. Il y aurait beaucoup à explorer dans ce contexte, comme vous pourrez le constater dans le cadre du projet. En guise d’exemple on s’en tiendra ici à l’action d’un champ magnétique extérieur variant sinusoidalement dans le temps:

$$H(t) = H_0 \sin\left(\frac{2\pi t}{P}\right), \quad t = 0, 1, 2, \dots, \quad (3.33)$$

où le temps t et la période d’oscillation P sont tous deux mesurés en itérations. La Figure 3.20 montre quatre exemples de l’évolution temporelle de la magnétisation, pour différentes combinaisons de H_0 et T , tel qu’indiqué. Quand H_0 est (relativement) grand et T nettement sous la température de Curie, comme en (A) et (B), le système inverse sa polarité magnétique en phase avec l’inversion du champ extérieur appliqué, sans cependant en reproduire la forme sinusoïdale. L’inversion de la magnétisation moyenne est en effet très rapide, et se produit quand la magnétisation extérieure approche son amplitude maximale. Si on réduit légèrement T et H_0 , on observe en (B) que l’inversion de la magnétisation moyenne retarde par rapport au pic de la magnétisation extérieure. On note également deux inversions “ratées” à $t \simeq 900$ et 4200, et un examen attentif de la séquence temporelle révèle également que l’intervalle de temps entre deux inversions de polarité dévie parfois de manière significative de la valeur attendue, soit $P/2 = 250$ itérations pour cette simulation. Ces irrégularités deviennent beaucoup plus importantes à mesure que H_0 diminue, comme on peut le constater en (C). Cependant, à la température de Curie, en (D), même une magnétisation extérieure très faible ($H_0 = 0.01$) induit une périodicité bien marquée dans la magnétisation. Le système, étant ici à son point critique, montre une sensibilité extrême à toute influence extérieure même de très faible amplitude.

La Figure 3.21 montre l’évolution de ces quatre mêmes simulations, cette fois tracée dans un espace de phase $[H(t), m(t)]$. Les solutions s’y déplacent dans le sens antihoraire. On notera comment la solution (C), ratant son inversion de polarité, retrace son chemin vers la droite, plutôt que transiter rapidement vers le bas ($m \simeq -1$) comme elle le ferait lors d’une inversion de la polarité positive à négative.

3.5 Projet: gravure magnétique par laser

L’application d’un champ magnétique spatiallement localisé sur une substance paramagnétisable sous sa température de Curie peut, pour un H suffisamment grand, induire une magnétisation localisée qui demeure une fois le champ extérieur H disparu. On peut ainsi encoder un “bit” ($0 \equiv$ pas de magnétisation, $1 \equiv$ magnétisation localisée). C’est le principe des vieux rubans magnétiques, technologie numérique remontant au Pléistocène (plus ou moins...) et reprise sur

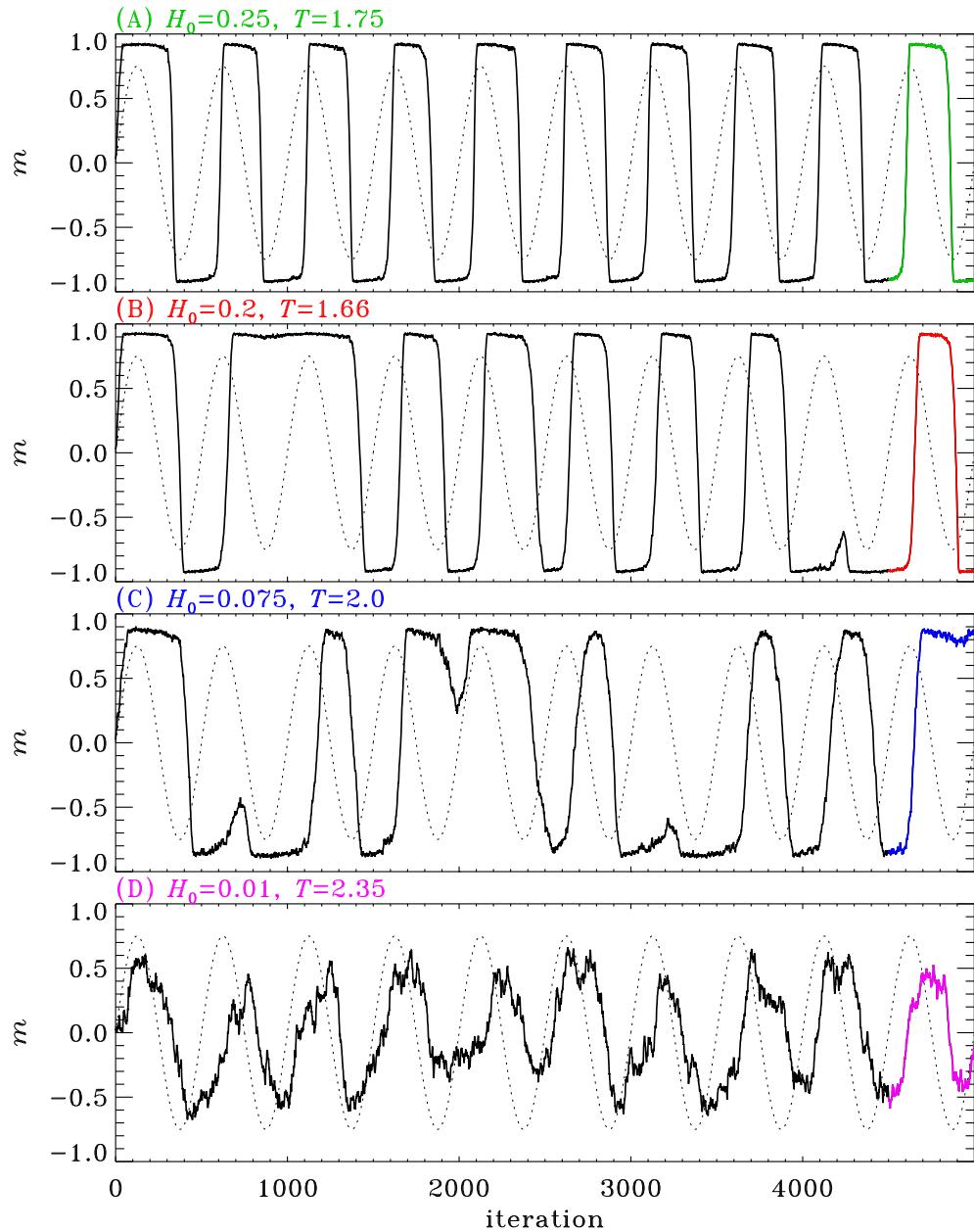


Figure 3.20: Séquences temporelles de la magnétisation moyenne par spin (traits pleins) dans un réseau 64×64 avec $J = 1$, sujet à une magnétisation extérieure variant sinusoidalement dans le temps (traits pointillés). Les quatre solutions sont obtenues avec des amplitudes de magnétisation extérieure H_0 et température T telles qu'indiquées. Dans tous les cas la condition initiale est aléatoire, et la période de la magnétisation extérieure est $P = 500$ itérations.

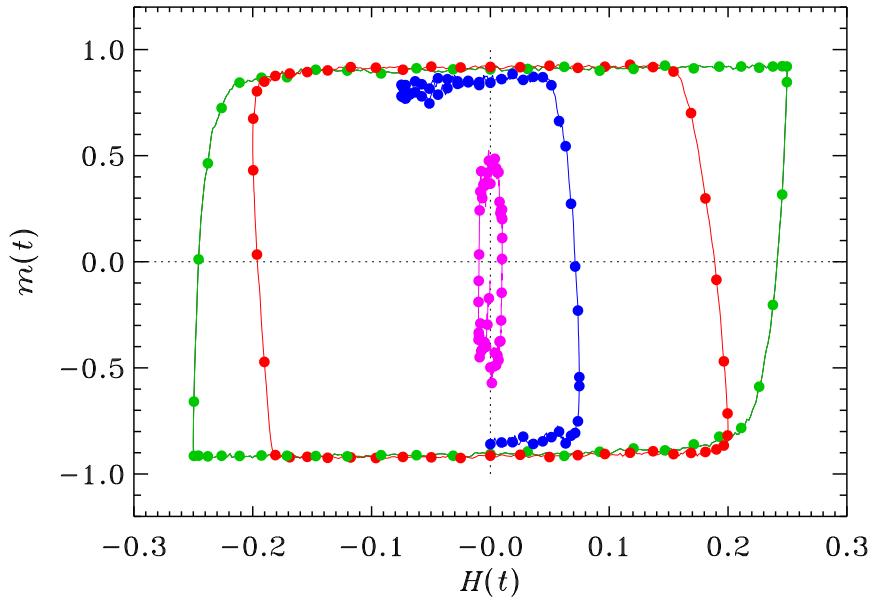


Figure 3.21: Trajectoires des quatre solutions de la Figure 3.20, dans un espace de phase défini par les séquences temporelles $m(t)$ et $H(t)$. Seule la période couverte par le dernier cycle d'oscillation de la magnétisation extérieure $H(t)$ est ici porté en graphique, selon le même code couleur qu'utilisé sur la Fig. 3.20. Les solutions évoluent dans le sens antihoraire, et les points sont tracés à une cadence de 10 itérations.

les disquettes de type “floppy” (âge du bronze environ...) et les premières générations de disques durs (un peu après la soi-disant Révolution Copernicienne, toujours approximativement...). D'un point de vue pratique, on veut atteindre une densité élevée (nombre de bits encodés par unité de surface sur le disque), ce qui implique produire un champ magnétique très intense et très localisé spatialement; ce qui est loin d'être évident, technologiquement parlant. Une alternative plus viable —et c'est celle utilisée aujourd'hui— consiste à imposer une faible magnétisation globale, et utiliser un faisceau laser pour pousser localement la température au delà de la température de Curie T_c , permettant l'alignement des spins avec la magnétisation extérieure. Cette magnétisation spatialement localisée devient ensuite permanente quand la région du disque ainsi chauffée se refroidit sous la température de Curie. C'est là l'idée que vous explorerez dans le cadre de ce projet à l'aide du modèle d'Ising³.

Dans un premier temps, on supposera que le laser chauffe une région circulaire du disque, qu'on représentera par un réseau cartésien de type Ising en 2D tel qu'introduit précédemment. On supposera que le disque est initialement partout à une température T_0 et que le profil temporel de chauffage peut être représenté par une gaussienne; donc on aura, à la position du spin (j, k) sur le réseau et à l'itération t :

$$T(j, k, t) = T_0 + \Delta T f(j, k) \exp\left(-\frac{(t - t_0)^2}{\sigma^2}\right) \quad (3.34)$$

avec le profil spatial $f(j, k)$ donné par:

$$f(j, k) = \begin{cases} 1 & \text{si } (j - j_0)^2 + (k - k_0)^2 \leq R^2 \\ 0 & \text{sinon} \end{cases}, \quad (3.35)$$

³Ce projet est inspiré en partie par une exploration numérique effectuée par Guillaume Emond, étudiant en PHY-3075 à la session d'hiver 2016.

où R est le rayon de la région chauffée par le laser, (j_0, k_0) le centre de cette région (mesuré en unités de distance inter-site), σ est la durée du pulse, t_0 l'itération à laquelle le pulse atteint son intensité maximale, et ΔT la hausse maximale de température produite par le chauffage.

La Figure 3.22 montre trois exemples de simulation effectuées sur un réseau 64×64 avec $(j_0, k_0) = (32, 32)$, $R = 10$, $t_0 = 100$, et $\sigma = 10$. Le disque a été “initialisé” à une polarité magnétique positive (tous les spins $+1 \equiv \uparrow$) et le champ ensuite appliqué est $H = -0.2$. La courbe en noir montre la variation temporelle de la magnétisation moyenne par spin, pour une simulation ayant $T_0 = 0.1$, et $\Delta T = 2.4$. Au pic du pulse, on a donc $T = 2.5$, correspondant à $T/T_c = 1.059$, soit légèrement au dessus de la température de Curie (cf. Fig. 3.19). Le profil temporel de chauffage est indiqué par le trait pointillé. La Figure 3.23 montre la distribution spatiale de la magnétisation à 200 itérations. Tout marche à merveille, on a produit un bit à peu près carré (bien que le chauffage suivait un profil circulaire), bien localisé spatialement et qui demeure stable une fois la température retombée à sa valeur initiale. Caramba, Banzai, et toute cette sorte de chose !

Ce n'est évidemment pas si simple; les courbes en rouge et vert sur la Fig. 3.22 montrent des séquences temporelles de la magnétisation moyenne par spin, dans deux simulations en tout point identique à la première, sauf que maintenant $T_0 = 0.5$ et $\Delta T = 2.0$; notons que la température maximale au pic du pulse demeure ainsi $T = 2.5$. Les deux simulations ne diffèrent qu'au niveau de l'initialisation du générateur de nombres aléatoires utilisé dans le contexte de l'algorithme de Metropolis. Dans le premier cas (vert) le bit s'érode lentement pour finalement disparaître, tandis que dans le second (rouge) il “envahit” tout le réseau, conduisant éventuellement à $m \simeq -1.0$ au bout de 2000 itérations. La température du substrat joue clairement un rôle important ici, même si T est substantiellement sous T_c .

Le but de ce projet et de vous faire découvrir si et comment il est possible de produire ainsi un bit stable, sous des conditions physiques plus现实的 than celles introduites ci-dessus. En pratique, on aimerait utiliser une magnétisation extérieure de la plus faible amplitude possible, ainsi qu'un pulse de faible amplitude, de manière à minimiser les besoins énergétiques. Un pulse de courte durée favorise aussi une écriture rapide du bit, et le système doit pouvoir opérer à une température finie. Finalement, on aimerait bien que le bit ne soit pas trop étendu spatialement, de manière à atteindre une haute densité dans le stockage de l'information. Gardant toutes ces contraintes en tête, les étapes sont les suivantes:

1. En guise de préliminaire, travaillez avec le modèle de base décrit ci-dessus, et examinez si et comment la production et la persistance du bit est possible, quand vous variez la durée du pulse, la température maximale, la taille de la région chauffée, la température basale, l'intensité du champ magnétique appliqué, etc. Il s'agit ici d'un exercice de réchauffement, pour éviter les entorses au cerveau.
2. L'idée qu'un pulse laser puisse imposer un profil de température spatialement constant sur un petit élément de surface est irréaliste; la diffusion de chaleur va jouer! Solutionnez numériquement l'équation de diffusion linéaire (2.3), avec coefficient de diffusion D constant, par la méthode FTCS, sur une maille 64×64 en utilisant le même profil spatial de chauffage que pour vos expériences précédentes avec le modèle d'Ising (soit les éqs. (3.34)–(3.35)). Il s'agit donc ici d'introduire un terme source $S(x, y, t)$ au membre de droite de l'équation de diffusion de la chaleur (soit l'éq. (2.113) avec $H = 0$ et χ constant), ce terme source ayant les mêmes dépendances spatiale et temporelle que dans vos expériences préliminaires en gravure magnétique. Considérez un domaine carré de côté $L = 64$ avec conditions limites $T = T_0$, ce T_0 étant le même qu'introduit précédemment.
3. Maintenant, répétez vos expériences de gravure magnétique en solutionnant *simultanément* l'équation de diffusion pour la chaleur afin de déterminer la température influençant localement chaque spin, en fonction de la position et du temps. Existe-t-il toujours un régime de paramètres où vous pouvez produire un bit stable ?
4. Introduire un nouveau stencil impliquant les 8 voisins de chaque spin, avec dans le calcul de l'énergie d'interaction une pondération de 2 pour les voisins immédiats (haut-bas-

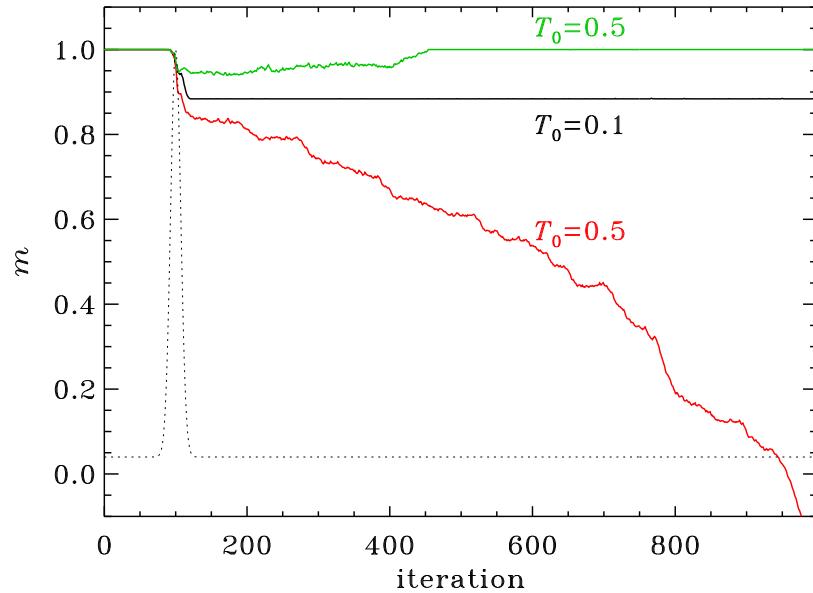


Figure 3.22: Séquences temporelles de la magnétisation moyenne par spin sur un réseau 64×64 avec $J = 1$, initialement magnétisé positivement (\uparrow) partout, et sujet à un champ extérieur $H = -0.2$ et un pulse thermique de forme circulaire en son centre (voir texte). Les paramètres du profil de chauffage (voir éqs. (3.34)–(3.35)) sont $(j_0, k_0) = (32, 32)$, $R = 10$, $t_0 = 100$, et $\sigma = 10$. La courbe en noir utilise $T_0 = 0.1$ et $\Delta T = 2.4$, tandis que pour les deux courbes colorées $T_0 = 0.5$ et $\Delta T = 2.0$, ces deux simulations ne différant qu’au niveau de la séquence de nombres aléatoires utilisée.

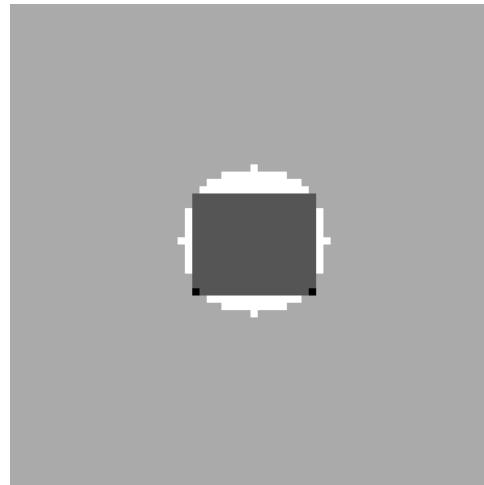


Figure 3.23: Bit produit par la simulation à $T_0 = 0.1$ (trait noir sur la Figure ci-dessus), superposé au profil spatial de chauffage, en blanc. Le gris clair correspond à une magnétisation positive (\uparrow) et gris foncé à \downarrow .

gauche-droite) et 1 pour les quatre voisins en diagonale. En posant $J = 1/3$ on aura de nouveau que l'énergie d'interaction pour un ensemble de spin alignés sera $|E| = 4$, ce qui facilitera la comparaison avec les résultats obtenus avec le stencil standard. Répétez vos expériences de gravure magnétique avec pulse thermique, et comparez aux résultats obtenus avec le stencil standard à quatre voisins. Votre bit a-t-il la même forme ? la même stabilité ?

5. (suite du précédent) et/ou: travaillez avec un réseau triangulaire (six voisins, pondération égale); en posant $J = 2/3$, un ensemble de spins alignés aura de nouveau une énergie d'interaction par spin de $|E| = 4$.
6. etc...

3.6 Bibliographie

Si le sujet vous intéresse, les deux vieux classiques géophysiques qui suivent sont définitivement à lire:

Gutenberg, B., & Richter, C.F., *Bull. Seismol. Soc. Am.*, **34**, 185 (1944).
 Burridge, R., & Knopoff, L., *Bull. Seismol. Soc. Am.*, **57**, 341 (1967).

Je suis loin d'être un expert en tremblements de terre ou en tectonique des plaques; c'est probablement la raison qui fait que j'ai trouvé intéressante la page Wikipedia sur le sujet:

<http://en.wikipedia.org/wiki/Earthquake>

Sur la reformulation sur réseau du modèle de Burridge-Knopoff, voir:

Carlson, J.M., & Langer, J.S., *Phys. Rev. A*, **40**, 6470 (1989),
 Olami, Z., Feder, H.J.S., & Christensen, K., *Phys. Rev. Lett.*, **68**, 1244 (1992),
 Hergarten, S., *Self-Organized Criticality in Earth Systems*, Berlin: Springer, chap. 7 (2002).

La présentation de la §3.2.2 suit de près celle faite à la §3.10 de l'ouvrage suivant, qui offre de surcroit une analyse mathématique complète du modèle OFC, ainsi que du modèle d'Ising:

Christensen, K., & Moloney, N.R., *Complexity and Criticality*, Imperial College Press (2005).

Sur ce sujet voir également:

Sethna, J.P., *Statistical Mechanics: Entropy, Order Parameters, and Complexity*, Oxford University Press (2006).

Si le calcul analytique de l'éq. (3.28) vous intrigue, attachez vos tuques avec de la broche et allez lire:

Binder, K., & Luijten, E., *Phys. Rep.*, **344**, 179 (2001).



© castierman - geluck

Chapitre 4

Les problèmes à N -corps

Le problème à deux corps (en physique), c'est facile, vous savez comment vous y prendre depuis le secondaire. Mais dans la vraie vie, on a souvent plus de deux corps en interaction, que ce soit de l'infiniment petit (atomes/molécules) à l'infiniment grand (étoiles et galaxies dans l'univers). Quand N est vraiment très grand, style nombre d'Avogadro et plus, la mécanique statistique permet de calculer les propriétés globales du système à partir de sa fonction de partition, donc ça va encore ici. La situation demeure cependant problématique dans le régime intermédiaire, quand N est grand mais pas assez pour s'en tirer avec la physique statistique. Seule une approche numérique est alors possible; c'est la simulation à N -corps.

Ce chapitre s'ouvre sur une forme particulière de simulations à N -corps, soit la *dynamique moléculaire*, qui nous plongera dans le domaine de l'infiniment petit. Le projet qui clot ce chapitre, par contre, se centre sur une simulation à N -corps de la formation d'une galaxie. De l'infiniment petit à l'infiniment grand dans le même chapitre, Vive La Simulation à N -corps !!

4.1 La dynamique moléculaire

La dynamique moléculaire est une approche très “Laplaciennne” à l'évolution d'un système physique: on solutionne les équations du mouvement du sieur Newton, soit $\mathbf{F} = m\mathbf{a}$, pour chaque constituant microscopique du système. Cette approche “force brute” a évidemment ses limites, à la fois pratiques¹, mais aussi conceptuelles, comme on le verra plus loin. Néanmoins, cette approche s'avère tout de même très utile pour bon nombre d'applications en physique des matériaux, par exemple.

L'élément critique dans ce genre de simulations est évidemment la spécification des interactions entre les particules, plus précisément la nature et formulation des forces agissant sur et entre elles.

4.1.1 Le potentiel de Lennard-Jones

La collision entre deux atomes partiellement ou complètement ionisés est dominée par la force électrostatique, et peut se traiter de manière entièrement classique, à moins que les énergies cinétiques soient suffisamment élevées pour que les deux atomes parviennent à s'approcher à des distances comparable au rayon de leurs nuages électroniques respectifs. L'interaction “collisionnelle” entre deux atomes ou molécules électriquement neutres, même à basses énergies, est un processus très complexe qui implique des effets quantiques associés à la configuration des orbitales électroniques et des niveaux d'excitation interne des molécules. Cependant il existe des représentations empiriques de ces effets qui ont pu être bien validées expérimentalement pour

¹En date de l'été 2015, le supercalculateur ayant la plus grande capacité de mémoire RAM atteignait $\sim 10^3$ TB, donc au gros maximum on pourrait suivre 2×10^{13} atomes ou molécules (48B chacun, soit 3 coordonnées spatiales et 3 composantes de vitesse encodées à 64 bit). Avec ça on simule à peu près un cinquantième d'un millimètre cube d'air dans une pièce à TPN... et en étant très, très patient!

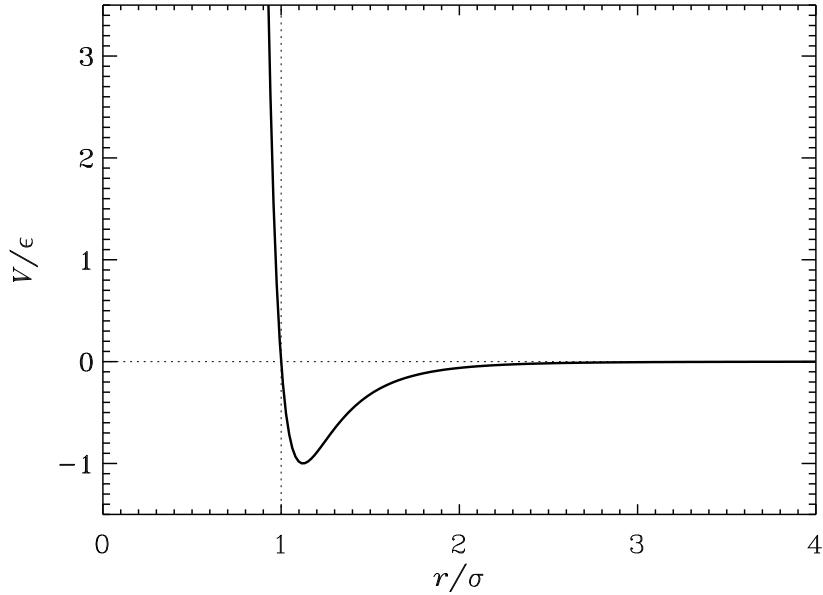


Figure 4.1: Le potentiel de Lennard-Jones, tel que décrit par l'éq. (4.1). Notons que $V = 0$ à $r = \sigma$, que le potentiel atteint sa valeur minimale $V = -\epsilon$ à $r/\sigma = 2^{1/6}$, et que $|V/\epsilon| \lesssim 0.005$ pour $r/\sigma > 3$.

certaines substances. Le *potentiel de Lennard-Jones* en est un exemple, applicable à plusieurs liquides à (relativement) basse température. On associe à chaque “particule” du système un potentiel donné par:

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right]. \quad (4.1)$$

La force agissant sur une particule k associée au potentiel ainsi produit par une particule j est donnée par le gradient de ce potentiel:

$$\mathbf{F}_k = -\nabla V(r_{jk}), \quad (4.2)$$

où r_{jk} est la distance entre les particules j et k , et le signe moins est ajouté explicitement de manière à ce qu'un potentiel négatif (positif) corresponde à une force d'attraction (de répulsion). La Figure 4.1 porte en graphique le potentiel de Lennard-Jones en fonction de la distance radiale. Le potentiel est fortement répulsif pour $r/\sigma < 1$, et attractif au-delà. La partie répulsive du potentiel représente ultimement l'effet du principe d'exclusion de Pauli. Quand les nuages électroniques de deux atomes se rapprochent trop, certains électrons doivent sauter à des niveaux d'énergie plus élevés afin de ne pas se retrouver avec les mêmes nombres quantiques, donc ils augmentent leur énergie; ceci se fait aux dépens de l'énergie cinétique des atomes, en les freinant, et donc agit globalement comme une force de répulsion inter-atomique. L'attraction capture la polarisation mutuelle des atomes/molécules; c'est la force de van der Waals.

Le choix de l'exposant 6 pour la contribution attractive du potentiel est motivé physiquement ($\text{dipôle} \times \text{dipôle} \equiv r^{-3} \times r^{-3} = r^{-6}$); l'exposant 12 pour la partie répulsive l'est passablement moins, mais ce qui importe vraiment pour la suite est que cet exposant soit significativement plus grand que 6.

4.1.2 Adimensionalisation du problème

Il sera pratique d'effectuer nos simulations dans un système d'unités adimensionnalisées, afin de minimiser les erreurs numériques associées aux produits et/ou quotients de quantités très

grandes et/ou très petites. Le potentiel de Lennard-Jones sous fournit déjà une longueur caractéristique, soit le paramètre σ , et une énergie (potentielle) caractéristique ϵ . Les forces peuvent alors s'exprimer en unités de ϵ/σ . On peut aussi choisir la masse d'un atome/molécule m comme unité de masse, dans lequel cas on peut exprimer les vitesses en unités de $\sqrt{\epsilon/m}$ et le temps en unités de $\sigma\sqrt{m/\epsilon}$.

Les simulations de la section §4.3 ci-dessous s'appliquent à l'Argon liquide, pour lequel le potentiel de Lennard-Jones offre une représentation bien validée des interactions interatomiques. Le tableau 4.1 liste les paramètres physiques correspondants, ainsi que les valeurs des facteurs d'adimensionalisation (en SI) pour les quantités physiques d'intérêt dans ce qui suit. Oczou,

Tableau 4.1: Adimensionalisation du problème

Quantité physique	Unités	Valeur pour l'Argon (SI)
Longueur	σ	$3.4 \times 10^{-10} \text{ m}$
Énergie	ϵ	$1.65 \times 10^{-21} \text{ J}$
Masse	m	6.69^{-26} kg
Temps	$\sigma(m/\epsilon)^{1/2}$	$2.17 \times 10^{-12} \text{ s}$
Vitesse	$(\epsilon/m)^{1/2}$	$1.57 \times 10^2 \text{ m s}^{-1}$
Force	ϵ/σ	$4.85 \times 10^{-12} \text{ N}$
Température	ϵ/k_B	120 K

petit rappel de la constante de Boltzmann: $k_B = 1.38 \times 10^{-23} \text{ J K}^{-1}$.

Avec σ comme unité de distance “atomique”, la densité dans nos simulations sera déterminée à la fois par le nombre de particules utilisées, ainsi que par la taille caractéristique L du domaine de simulation (un carré de taille $L \times L$ dans tout ce qui suit). Il est amusant de considérer la durée, en unités physiques, des simulations effectuées à la §4.3 ci-dessous; on simulera au plus sur 10^4 temps adimensionalisés, ce qui correspond à un minuscule $\simeq 10^{-8}$ secondes !

4.2 Traitement numérique

Il s'agit maintenant de solutionner les équations du mouvement pour chacune des particules d'un ensemble de N particules intergissant mutuellement via une force donnée par le gradient du potentiel de Lennard-Jones:

$$\frac{\partial^2 x}{\partial t^2} = -\frac{\partial V}{\partial x}, \quad (4.3)$$

$$\frac{\partial^2 y}{\partial t^2} = -\frac{\partial V}{\partial y}, \quad (4.4)$$

n'ayant pas oublié que selon notre choix d'unités, la masse des particules $m \equiv 1$. La procédure habituelle serait de convertir ceci en un système de quatre EDOs d'ordre 1 en utilisant la vitesse comme variable secondaire:

$$\frac{\partial v_x}{\partial t} = -\frac{\partial V}{\partial x}, \quad (4.5)$$

$$\frac{\partial v_y}{\partial t} = -\frac{\partial V}{\partial y}, \quad (4.6)$$

$$\frac{\partial x}{\partial t} = v_x, \quad (4.7)$$

$$\frac{\partial y}{\partial t} = v_y. \quad (4.8)$$

Ici cependant, comme le potentiel ne dépend que des coordonnées spatiales, on peut faire mieux.

4.2.1 L'algorithme de Verlet

L'*algorithme de Verlet* prend avantage de l'indépendance du potentiel sur la vitesse pour intégrer de manière explicite et à l'ordre deux les éqs. (4.3). Il s'agit ici d'un calcul en trois étapes séquentielles devant être effectuées à chaque pas de temps:

1. Étape 1: pour toutes les particules, calcul des positions à $n + 1$:

$$x^{n+1} = x^n + v_x^n(\Delta t) + \frac{1}{2}a_x^n(\Delta t)^2 , \quad (4.9)$$

$$y^{n+1} = y^n + v_y^n(\Delta t) + \frac{1}{2}a_y^n(\Delta t)^2 , \quad (4.10)$$

2. Étape 2: pour toutes les particules, calcul des accélérations à $n + 1$:

$$a_x^{n+1} = -\frac{\partial V(x^{n+1}, y^{n+1})}{\partial x} , \quad (4.11)$$

$$a_y^{n+1} = -\frac{\partial V(x^{n+1}, y^{n+1})}{\partial y} . \quad (4.12)$$

3. Étape 3: pour toutes les particules, calcul des vitesses à $n + 1$:

$$v_x^{n+1} = v_x^n + \frac{\Delta t}{2}(a_x^n + a_x^{n+1}) , \quad (4.13)$$

$$v_y^{n+1} = v_y^n + \frac{\Delta t}{2}(a_y^n + a_y^{n+1}) , \quad (4.14)$$

On notera que seules les composantes des accélérations doivent être conservées en mémoire aux pas de temps n et $n + 1$; celles des positions et vitesses sont immédiatement mises à jour et conservées dans un seul tableau par composante.

L'algorithme de Verlet s'avère être d'ordre deux par rapport à la discréétisation temporelle. Les équations (4.9)–(4.10) sont équivalentes à un développement en série de Taylor pour $x(t + \Delta t)$ où les trois premier termes de la série sont conservés; donc l'avance des x, y est précise à l'ordre deux dans le temps (pour un Δt suffisamment petit, évidemment). Puisque le potentiel ne dépend que de x, y , on peut alors calculer *sans approximation* le potentiel, et donc les accélérations, au pas $n + 1$; ce qui permet ensuite d'avancer les vitesses (éqs. (4.13)–(4.14)) par la méthode d'Euler explicite, en évaluant le membre de droite centré dans le temps, ce qui confère au calcul des vitesses un ordre deux dans le temps; c'est effectivement une méthode Runge-Kutta d'ordre deux ici. L'algorithme de Verlet est donc applicable à toute force ne dépendant pas de la vitesse, ce qui inclue la gravité et la force électrostatique.

4.2.2 Conditions initiales et limites

La spécification des conditions initiales est plus délicate que l'on pourrait l'imaginer. Pour une simulation de type dynamique moléculaire, une initialisation aléatoire des positions et vitesses des N particules peut conduire à des gros problèmes si deux particules se retrouvent ainsi initialement très en deçà du rayon de répulsion du potentiel de Lennard-Jones ($r/\sigma = 2^{1/6}$); dès le premier pas de temps les deux particules accélèrent à de très hautes vitesses, et les collisions subséquentes avec les autres particules font “éclater” le système. Il faut donc prendre une condition initiale où toutes les particules sont à une distance d'au moins σ les unes des autres, et leur donner des vitesses suffisamment basses pour éviter des interactions trop énergétiques dans

les phases initiales de relaxation de la condition initiale; au vu de notre adimensionalisation, des vitesses de l'ordre de l'unité sont habituellement appropriées.

Au niveau des conditions limites, comme nous nous retrouvons forcés en pratique à utiliser des N beaucoup plus bas que souhaité, l'imposition de conditions limites périodiques permet au moins de minimiser les effets de bord. L'idée est illustrée schématiquement sur la Figure 4.2. Le domaine de simulation est indiqué par le carré noir, et contient ici $N = 16$ particules. On suppose que ce carré se répète horizontalement et verticalement à l'infini, comme des tuiles carrées couvrant un plancher (carrés pointillés). Chaque carré contient des copies des particules du carré de base, situées aux même positions (relatives) et se déplaçant avec les mêmes vitesses. Donc, si une des particules du carré de base traverse le bord de droite, une de ses homologues entre du côté gauche, de telle sorte que le nombre total de particules dans chaque carré demeure constant.

Si la périodicité est facile à imposer au niveau de la position des particules, elle l'est moins sur le calcul des forces; la particule homologue entrant dans le carré de base par la gauche doit déjà avant son entrée, via la portée finie de son potentiel, exercer une force sur les autres particules situés près du bord gauche du carré de base.

Il existe plusieurs manières de s'assurer que les forces soient elles aussi périodiques. Dans une situation où le domaine est significativement plus grand que la portée du potentiel (dans notre notation, $L \gg 3\sigma$ pour le potentiel de Lennard-Jones), une approche pratique consiste à bâtir un “tampon” de particules “fantômes” en périphérie du carré de base, correspondant à la zone en gris sur la Fig. 4.2. Chaque particule du carré de base située à moins d'une distance d'interaction déterminée par la portée —ou la troncation— du potentiel est recopiée à une distance L dans la direction opposée, comme l'indique le code couleur sur la Fig. 4.2. Notez que les particules situées près des coins du carré de base se voient recopiées trois fois: horizontalement, verticalement et diagonalement (e.g. les particules orange, violettes, et bleu ciel). Dans le cas spécifique de la Figure 4.2 le calcul des forces implique maintenant $16 + 20$ particules, mais l'algorithme de Verlet n'est appliqué qu'aux 16 particules du carré de base.

La Figure 4.3 donne un exemple d'une fonction Python bâtant un tel tampon. Ici N est le nombre de particules dans le carré de base, la variable `nb` est le nombre total final de particules, `LL` est la longueur du carré de base, et `DB` la distance d'interaction déterminant la largeur du tampon. Les tableaux `x` et `y` contiennent les positions des particules du carré de base, et les tableaux `xb` et `yb` celles de toutes les particules, carré de base et tampon. On verra plus loin pourquoi il est avantageux de procéder ainsi.

4.2.3 Dynamique moléculaire en Python

La Figure 4.4 offre une implémentation en Python d'une simulation de dynamique moléculaire en deux dimensions spatiales sur un carré périodique, utilisant les formes adimensionnelles des variables physiques, tel qu'introduites précédemment. Ce code n'est pas nécessairement optimal au niveau du temps d'exécution, mais demeure (relativement) lisible. Notez bien (en comprenez bien) les points suivants:

1. Le code principal (lignes 37–50) n'est guère plus qu'un codage à peu près direct des trois étapes de l'algorithme de Verlet, avec périodicité sur les positions introduite aux lignes 42–43 à l'aide de l'opérateur modulo (“%” en Python). Toute la complexité du calcul est cachée dans la fonction `calca`, qui calcule les accélérations.
2. La variable `DB` est initialisée (ligne 9) à trois unités adimensionnelles de distance, soit trois fois le paramètre σ du potentiel de Lennard-Jones.
3. La fonction `tampon` crée des versions fantômes de toutes les particules situées à moins d'une distance `DB` des bords du domaine périodique (viz. Fig. 4.2).
4. La fonction `calca` cumule les contributions des forces associées à chaque particule j sur chaque particule k , en bouclant sur les paires distinctes j, k . Les forces et potentiels sont remis à zéro en entrée de la fonction (ligne 23), donc à chaque itération temporelle.

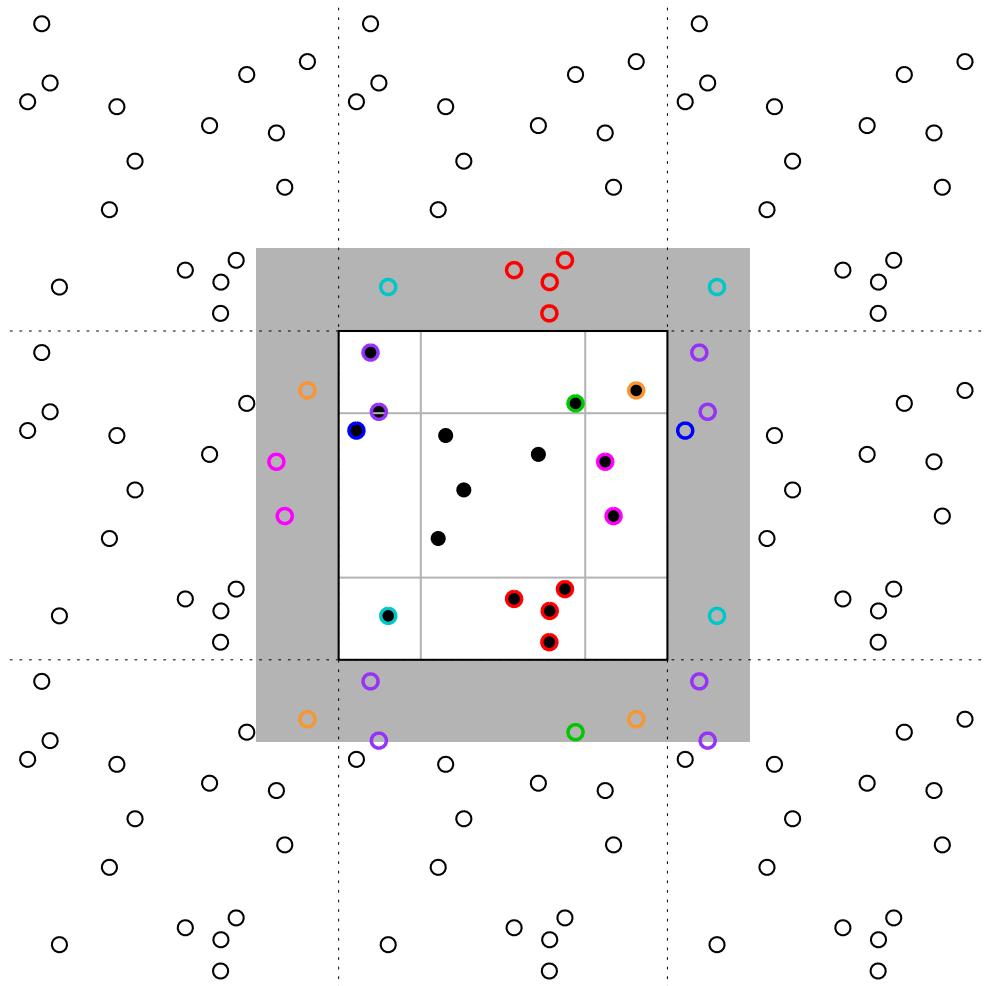


Figure 4.2: Définition d'une zone tampon pour assurer un calcul correct des forces dans un système périodique en x et y . Chaque particule (points noirs) située à moins d'une distance r_b d'un des bord de la cellule périodique (carré noir) est dupliquée à une distance L dans la direction opposée, les particules de coin se retrouvant ainsi recopiées trois fois; examinez attentivement les couleurs sur ce diagramme afin de bien saisir ce processus de duplication.

```

1 # FONCTION BATISSANT UN TAMPON POUR CALCUL DES FORCES
2 def tampon(x,y,xb,yb):
3
4     xb[0:N]=x[0:N]                                # les particules primaires
5     yb[0:N]=y[0:N]
6     nb=N
7     for j in range(0,N):                         # boucle sur particules
8         if x[j] <= DB:                          # ajout a droite
9             xb[nb]=x[j]+LL
10            yb[nb]=y[j]
11            nb+=1
12         if x[j] >= (LL-DB):                   # ajout a gauche
13             xb[nb]=x[j]-LL
14            yb[nb]=y[j]
15            nb+=1
16         if y[j] <= DB:                          # ajout en haut
17             xb[nb]=x[j]
18            yb[nb]=y[j]+LL
19            nb+=1
20         if y[j] >= (LL-DB):                   # ajout a gauche
21             xb[nb]=x[j]
22            yb[nb]=y[j]-LL
23            nb+=1
24         if x[j] <= DB and y[j] <= DB:        # ajout en haut a droite
25             xb[nb]=x[j]+LL
26            yb[nb]=y[j]+LL
27            nb+=1
28         if x[j] <= DB and y[j] >= (LL-DB):    # ajout en bas a droite
29             xb[nb]=x[j]+LL
30            yb[nb]=y[j]-LL
31            nb+=1
32         if x[j] >= (LL-DB) and y[j] >= (LL-DB): # ajout en bas a gauche
33             xb[nb]=x[j]-LL
34            yb[nb]=y[j]-LL
35            nb+=1
36         if x[j] >= (LL-DB) and y[j] <= DB:      # ajout en haut a gauche
37             xb[nb]=x[j]-LL
38            yb[nb]=y[j]+LL
39            nb+=1
40     return nb                                    # nombre total de particules
41 # END FONCTION TAMPON

```

Figure 4.3: Fonction Python pour la construction d'un tampon de particules fantômes (voir texte). Les variables LL et DB sont supposées globales, et doivent être initialisées dans le code principal appelant cette fonction (voir Fig. 4.4 plus bas).

```

1 # SIMULATION DYNAMIQUE MOLECULAIRE, POTENTIEL LENNARD-JONES
2 # NORMALISATION: LONGUEUR=SIGMA, ENERGIE=EPSILON, MASSE=1 (VOIR SEC. 4.1.2)
3 #=====
4 import numpy as np
5 LL    =6.                      # TAILLE DU DOMAINE, EN UNITES DE SIGMA
6 N     =16                     # NOMBRE DE PARTICULES
7 NITER=2000                   # NOMBRE DE PAS DE TEMPS
8 DT    =1.0d-2                 # TAILLE DU PAS DE TEMPS
9 DB    =3.                      # RAYON DE TRONCATION DU POTENTIEL
10 #-----
11 def force(d2,xk,xj,yk,yj,fxk,fyk,potk):
12 # FONCTION CALCULANT LA FORCE DE J SUR K (SELON POTENTIEL LENNARD-JONES)
13     dx,dy=xk-xj,yk-yj          # differences en x et y
14     qq=(1./d2)**3              # =(sigma/r)**6 (sigma=1)
15     f=24.0*( qq-2.*qq**2 )/d2 # la grandeur de la force /r
16     ffxk,fyk=-f*dx,-f*dy      # ses composantes x,y
17     potk=4.0*(qq**2-qq)        # le potentiel sur k par j (epsilon=1)
18     return
19 # END FONCTION FORCE
20 #-----
21 # FONCTION CALCULANT L'ACCELERATION ET POTENTIEL POUR TOUTES LES PARTICULES
22 def calca(x,y,xb,yb,nb,fx,fy,pot):
23     fx,fy,pot=np.zeros(N),np.zeros(N),np.zeros(N)
24     for k in range(0,N):         # boucle sur particules primaires
25         for j in range(k+1,nb):   # demie-boucle sur les particules
26             d2=(x[k]-xb[j])**2+(y[k]-yb[j])**2 # distance au carre entre j et k
27             if d2 <= DB*DB:           # dans le rayon de troncation
28                 force(d2,x[k],xb[j],y[k],yb[j],fxk,fyk,potk)
29                 fx[k] +=fxk          # force-x de j sur k
30                 fy[k] +=fyk          # force-y de j sur k
31                 pot[k]+=potk        # contribution de j au potentiel sur k
32                 if j < N:            # j est une particule primaire
33                     fx[j]=-fxk        # force-x de k sur j: action-reaction !
34                     fy[j]=-fyk        # force-y de k sur j: action-reaction !
35     return
36 # END FONCTION COMPUTE
37 #-----
38 # PROGRAMME PRINCIPAL
39 ...                         # declarations, condition initiale, etc
40 for it in range(0,NITER):    # boucle temporelle
41     x=x+vx*DT+0.5*ax*DT**2  # positions a n+1 (Verlet, etape 1)
42     y=y+vy*DT+0.5*ay*DT**2
43     x = (LL+x) % LL          # periodicite sur positions
44     y = (LL+y) % LL
45     nb=tampon(x,y,xb,yb)    # construction du tampon
46     calca(x,y,xb,yb,nb,fx,fy,pot) # accelerations a n+1 (Verlet, etape 2)
47     vx=vx+0.5*(ax+fx)*DT    # vitesses a n+1 (Verlet, etape 3)
48     vy=vy+0.5*(ay+fy)*DT
49     ax,ay=fx,fy              # on conserve les accelerations
50     ...
51 # END

```

Figure 4.4: Code Python pour la dynamique moléculaire sous le potentiel de Lennard-Jones, dans un domaine cartésien périodique en x, y . Voir explications dans le texte.

5. Dans la fonction `calca`, la première boucle (débutant ligne 24) opère sur les N particules primaires (`range(0,N)`), tandis que la seconde (débutant ligne 25) inclue les particules fantômes du tampon (`range(k+1,nb)`); chaque itération de cette seconde boucle calcule les force et potentiel agissant sur la particule k dus à la particule j .
6. Cette seconde boucle commence à $k + 1$ pour ne pas calculer deux fois la force entre les particules k et j ; la contribution de la force agissant sur la particule j due à la particule k est de signe opposé à celle de k sur j (action-réaction!), et n'est comptabilisée que pour les particules primaires (i.e., excluant les particules tampon via l'instruction `if` à la ligne 32).
7. Avec N particules primaires et un total de `nb` incluant celles du tampon, on devra calculer $(1/2) \times N \times nb$ vecteurs-force par itération temporelle; il sera important d'être efficace; ici le calcul de la force n'est effectué que si la distance entre j et k (calculée à la ligne 26) est inférieure au rayon de troncation `DB`, qui est le même que celui utilisé précédemment pour déterminer la largeur du tampon.
8. La force produite par j sur k est calculée dans la fonction `force`; on remarquera la définition d'une variable interne `qq`, pour réduire le nombre d'opération mathématiques effectuées par la fonction. Pour la même raison, la distance (au carré) `d2` ayant déjà calculée dans `calca`, on la passe en argument à `force` plutôt que de la recalculer à l'interne. Ce genre de petites manœuvres réduit significativement le temps d'exécution, surtout quand N est grand;
9. Toujours pour minimiser les opérations mathématiques, la force est pré-divisée par r , de manière à ce que le calcul des composantes en x et y n'implique qu'une multiplication par `dx` et `dy` (plutôt que par $\cos \theta_{ij} \equiv \Delta x / r_{ij}$ et $\sin \theta_{ij} \equiv \Delta y / r_{ij}$).

4.2.4 Quelques test simples

Comme toujours il est important de bien tester l'algorithme de Verlet avant de se lancer dans des calculs sérieux. On considère ici deux test simples: interactions ballistiques entre deux particules, et inversion temporelle pour un système de 16 particules.

La Figure 4.5 montre deux exemples d'interactions à deux particules sous l'influence du potentiel de Lennard-Jones, avec troncation à $r/\sigma = 3$. Les deux conditions initiales diffèrent au niveau de leur paramètre d'impact; on a pour la condition initiale “rouge”:

$$x_1^0 = 3, v_{x1}^0 = 1, y_1^0 = 10.25, v_{y1}^0 = 0, x_2^0 = 17, v_{x2}^0 = -1, y_2^0 = 9.75, v_{y2}^0 = 0, \quad (4.15)$$

et pour la condition initiale “verte”:

$$x_1^0 = 3, v_{x1}^0 = 1, y_1^0 = 11, v_{y1}^0 = 0, x_2^0 = 17, v_{x2}^0 = -1, y_2^0 = 9, v_{y2}^0 = 0, \quad (4.16)$$

La première a donc un paramètre d'impact $b \equiv y_1^0 - y_2^0 = 0.5$, et la seconde $b = 2.0$; le premier paramètre d'impact est plus petit que le rayon σ (cercle pointillé) sous lequel joue la forte répulsion du potentiel de Lennard-Jones, tandis que le second demeure dans la portée attractive du potentiel (cercle en tiret, $r/\sigma = 3$). On observe donc des déviations qualitativement très différentes suite aux deux “collisions”.

Un bon indicateur de la précision numérique de ces solutions est l'énergie totale du système (E), qui ici doit être conservée. L'énergie totale compte deux contributions, cinétique et potentielle:

$$E = \underbrace{\sum_{i=1}^N \frac{v_{xi}^2 + v_{yi}^2}{2}}_{E_K} + \underbrace{\frac{1}{2} \sum_{i \neq j} V(r_{ij})}_{E_P}, \quad (4.17)$$

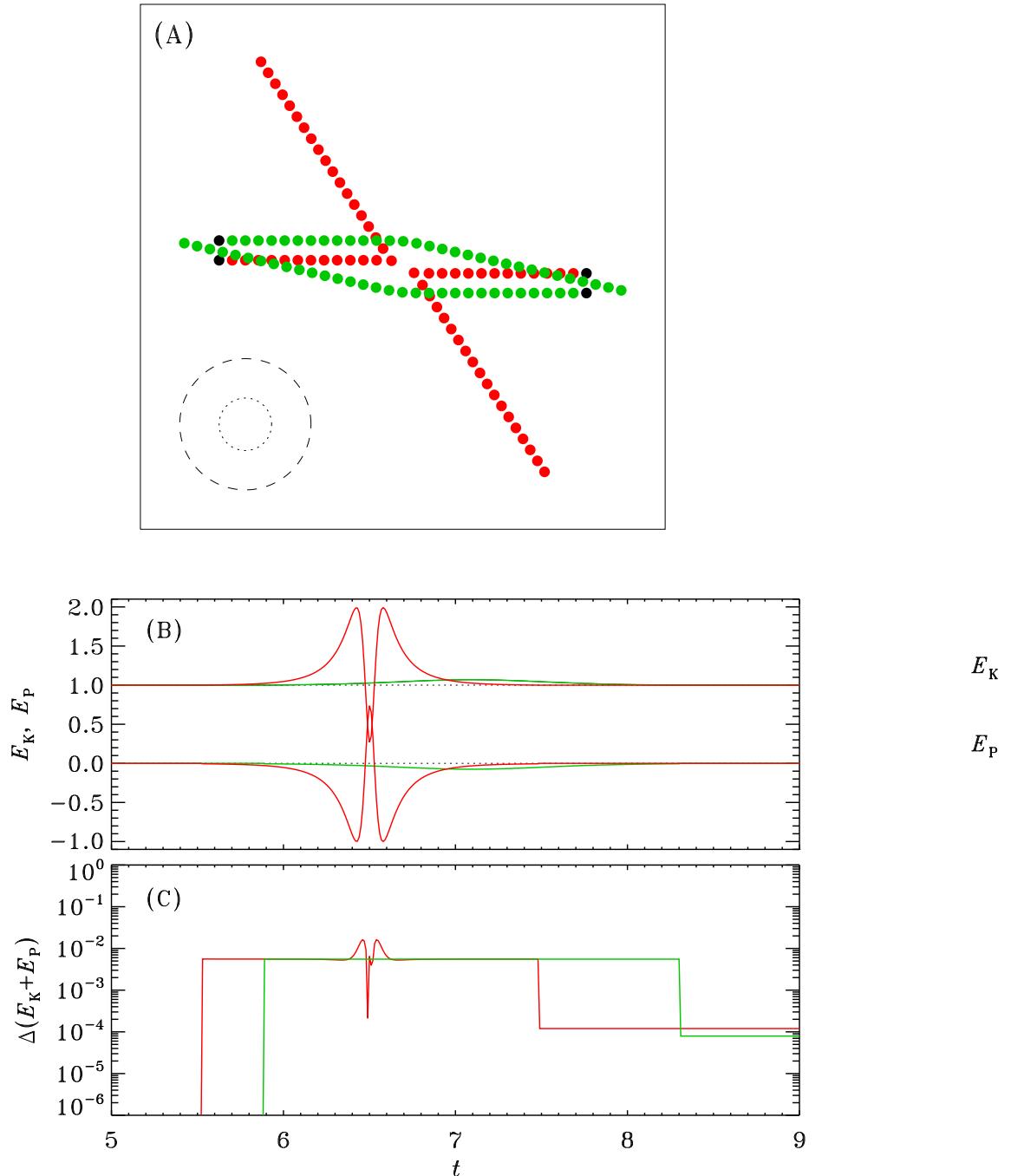


Figure 4.5: Collisions à deux particules dont l’interaction est décrite par le potentiel de Lennard-Jones. La partie (A) montre deux “collisions” ayant des paramètres d’impact $b \equiv \Delta y = 0.5$ (rouge) et 2.0 (vert). Les points noirs indiquent les positions initiales des particules, qui à $t = 0$ se déplacent horizontalement l’une vers l’autre avec vitesse $v_x^0 = \pm 1$. Ici $L = 20$ et $\Delta t = 10^{-2}$, et le potentiel Lennard-Jones est tronqué à $r/\sigma = 3$. Les positions des particules sont tracées à une cadence de $50\Delta t = 0.5$. La partie (B) montre l’évolution des énergies potentielle (E_P) et cinétique (E_K), sur un intervalle de temps couvrant les deux “collisions”, et la partie (C) la variation temporelle de l’erreur en énergie (voir éq. (4.19)).

où la seconde somme ne doit compter qu'une seule fois la contribution de la paire ij , d'où le facteur $1/2$, et on n'oublie pas que $m = 1$ dans notre système adimensionnel. Dans le cas des systèmes à deux particules de la Fig. 4.5 cette expression se réduit à:

$$E = \sum_{i=1}^2 \frac{v_{xi}^2 + v_{yi}^2}{2} + V(r_{12}) , \quad (4.18)$$

avec $r_{12}^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$. Les parties (B) et (C) de la Fig. 4.5 montrent les variations temporelles des énergies cinétique, potentielle (en B) et totale (en C) pour les deux collisions en (A). À l'oeil en (B), les interactions impliquent bien une conversion cinétique \rightarrow potentielle \rightarrow cinétique, comme il se doit, mais la partie (C) montre que l'énergie totale E n'est pas parfaitement conservée; on y a porté en graphique sur une échelle verticale logarithmique la quantité:

$$\Delta E = |E(t) - E(0)| , \quad (4.19)$$

mesurant ici le niveau de non-conservation de l'énergie. Les sauts se produisant à $t \simeq 5.5$ et 5.9 sont une conséquence directe de la troncation du potentiel à $r/\sigma = 3$ ici; réduire le pas de temps n'y change pas grand chose. On notera cependant que l'erreur retombe substantiellement, quoique pas jusqu'à zéro, quand les particules ressortent du rayon de troncation du potentiel, à $t \simeq 7.5$ et $t \simeq 8.3$. Pour la collision "rouge", l'erreur monte également un signal complexe à $t \simeq 6.5$, correspondant au moment où les particules entrent à l'intérieur de leur rayons de répulsion ($r \lesssim \sigma$); ici une réduction du pas de temps à $\Delta t = 10^{-3}$ élimine presque complètement ce signal.

Ceci suggère que la troncation du potentiel de Lennard-Jones à une distance finie n'est vraiment pas une bonne idée. On le fera cependant dans tout ce qui suit, afin de réduire le temps de calcul dans les systèmes à plusieurs particules.

Un autre test très intéressant et instructif de l'algorithme de Verlet est l'*inversion temporelle*: on évolue une distribution de particules pendant un intervalle de temps P , puis on inverse instantanément les vitesses de toutes les particules et on évolue pendant un autre intervalle de temps P ; les particules devraient revenir à leur positions initiales, avec leurs vitesses inversées par rapport à celles de la condition initiale. La rangée du haut sur la Figure 4.6 montre le résultat de cet exercice. À $t = 0$ on assigne des vitesses aléatoires à $N = 16$ particules distribuées aux intersections d'un quadrillage 4×4 (en noir à gauche), les composantes des vitesses étant extraites de distributions uniformes $\in [-2, 2]$. On a ici $L = 6$ et $\Delta t = 10^{-2}$. Après 600 pas de temps, soit à $t = 6$ (en rouge), on inverse les vitesses, et on poursuit la simulation jusqu'à $t = 12$ (en vert). On retrouve bien la condition initiale, avec les vitesses inversées.

La rangée centrale montre une simulation de la même inversion temporelle à $t = 6$ mais calculée ici avec un pas de temps dix fois plus petit, soit $\Delta t = 10^{-3}$; si ça marchait bien à $\Delta t = 10^{-2}$ on serait en droit de s'attendre à ce que ça fonctionne aussi à $\Delta t = 10^{-3}$; et c'est bien le cas ici. Mais, holà, comparons les deux solutions à $t = 6$, au moment de l'inversion (colonne centrale, en rouge): complètement différentes ! Kosséssa !?

À cause de l'accumulation des erreurs numériques lors des "collisions", les solutions pour $\Delta t = 10^{-2}$ et 10^{-3} se retrouvent à suivre des trajectoires divergentes dans l'espace de phase (à 64 dimensions!) du système. Cependant, la grande part de ces erreurs provient de la troncation du potentiel de Lennard-Jones à $r/\sigma = 3$, et celles-ci sont en grande partie symétriques par rapport à l'entrée et sortie du rayon 3σ (viz. Fig. 4.5C), et donc sont réversibles temporellement. Les deux trajectoires dans l'espace de phase sont différentes, mais chacune est ici réversible à un haut niveau de précision; d'où le retour à la condition initiale dans les deux cas, même si les solutions sont complètement différentes à $t = 6$ au moment de l'inversion.

Si on pousse plus loin dans le temps, il viendra inévitablement un moment où l'inversion temporelle se dégradera. La rangée du bas de la Figure 4.6 montre une simulation à $\Delta t = 10^{-2}$, comme la première, mais avec cette fois l'inversion se produisant deux fois plus loin dans le temps, à $t = 12$. Cette fois, à $t = 24$ on remarque des différences notables dans les positions

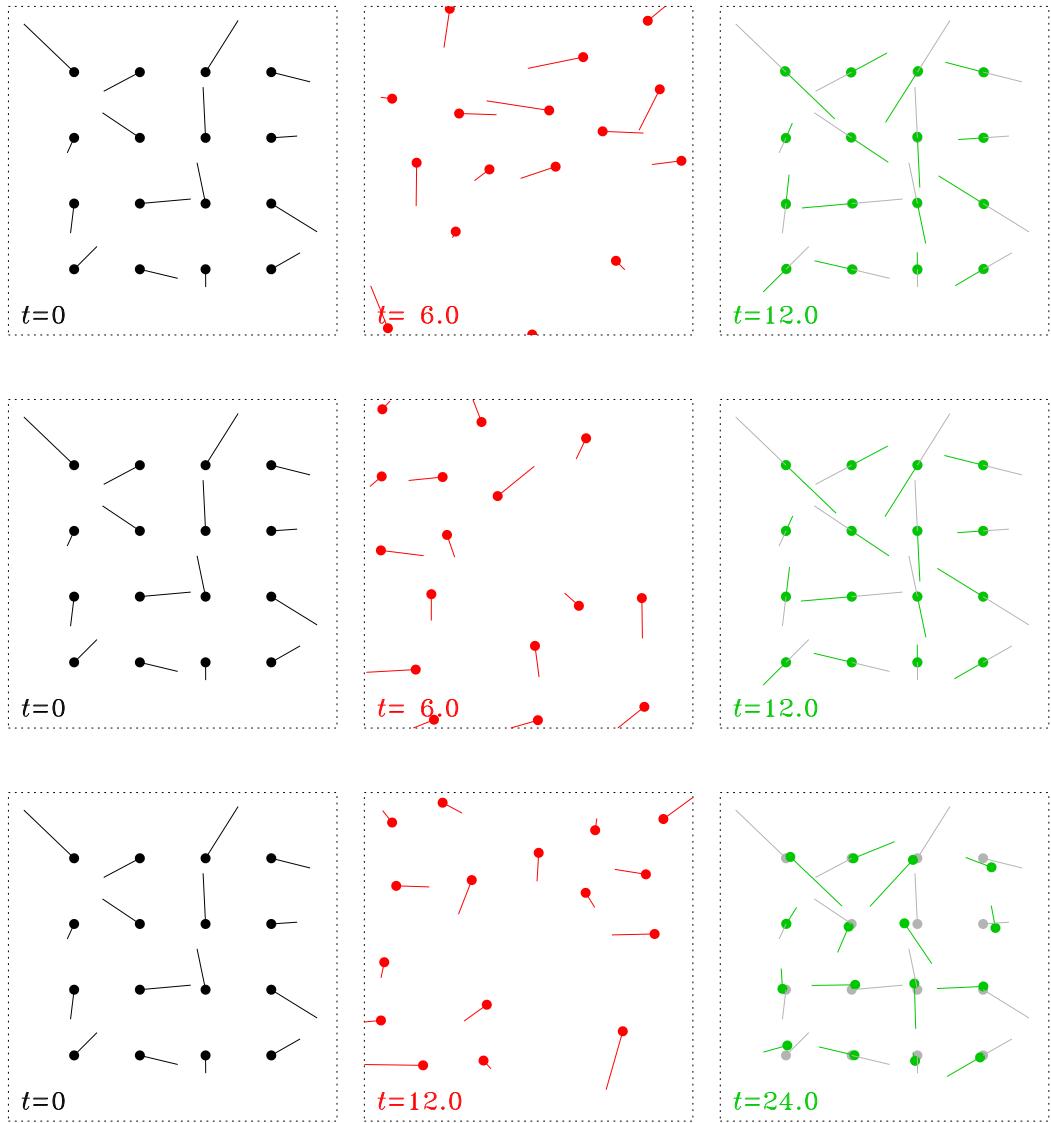


Figure 4.6: Test d'inversion temporelle sur $N = 16$ particules avec condition initiale aléatoire sur les vitesses. Rangée du haut: $\Delta t = 10^{-2}$ et inversion à $t = 6$; Rangée du centre: $\Delta t = 10^{-3}$ et inversion à $t = 6$; Rangée du bas: $\Delta t = 10^{-2}$ et inversion à $t = 12$. La condition intiale est identique pour les trois simulations. En fin de simulation (colonne de droite), les positions et vitesses de la condition initiale sont retracées en gris, pour faciliter la comparaison. Dans tous les cas $L = 6$ et troncation du potentiel à $r/\sigma = 3$.

et vitesses (inversées) par rapport à la condition initiale. On pourrait y voir l'accumulation inexorable d'erreurs de conservation de l'énergie au moment des interactions (viz. Fig. 4.5C), mais ce n'est pas vraiment le cas. Si on examine l'évolution de la solution en détail, on réalise que le bris de l'inversion temporelle peut être retracé à une interaction rapprochée spécifique, se produisant à $t = 7.97$, durant laquelle les particules 8 et 13 s'approchent à une distance $r_{ij} = 0.918\sigma$ l'une de l'autre; ceci introduit une erreur de conservation de l'ordre de $\simeq 10^{-2}$, qui, comme sur la Fig. 4.5C, n'est *pas* symétrique temporellement par rapport au moment de plus forte approche (pic dans l'énergie potentielle), en raison d'erreurs numériques associées à la discréétisation. Dans la phase "retour" de la simulation ($12 \leq t \leq 24$), une autre erreur d'amplitude similaire est introduite, mais décorrélée de celle introduite durant la première moitié de la simulation.

4.3 Exemple: simulation d'un liquide

Conceptuellement, la dynamique moléculaire telle qu'introduite dans ce chapitre repose sur deux bases d'une extrême simplicité: $\mathbf{F} = m\mathbf{a}$ et une forme spécifique pour un potentiel déterminant les interactions dynamiques particule-particule. Passons maintenant à la simulation et exploration, via la dynamique moléculaire, de certains comportements observés dans des vrais fluides.

4.3.1 Considérations thermodynamiques

Malgré l'arbitraire (relatif) des conditions initiales utilisées pour les simulations de la Figs. 4.6, le système atteint rapidement un équilibre (statistique) caractérisé par des transferts entre les énergies potentielles et cinétiques durant les interactions particules-particules, conservant l'énergie totale du système à un degré raisonnable de précision même pour des pas de temps relativement grands, $\sim 10^{-2}$ dans nos unités adimensionnalisées. Mais peut-on vraiment prétendre qu'on est vraiment en train de simuler un vrai liquide comme l'Argon sur la base d'une simulation n'impliquant que 16 particules se déplaçant dans un carré périodique ?

La Figure 4.7 montre la fonction de densité de probabilité des vitesses v_i ($i = 1, \dots, N$), où $v_i^2 \equiv \mathbf{v}_i \cdot \mathbf{v}_i$. Cette fonction est construite en échantillonnant 100 fois les vitesses des $N = 16$ particules dans une simulation identique à celle de la Figure 4.6, à une cadence de 200 pas de temps. L'allure de cette distribution devrait vous rappeler quelque chose...

Une fois à l'équilibre, le sieur Boltzmann nous informe qu'il est possible de caractériser la vitesse moyenne des particules en terme d'une quantité globale que l'on nomme *la température*. Pour un système de N particules se déplaçant en deux dimensions spatiales, on a

$$\langle T \rangle = \frac{1}{2N} \sum_{i=1}^N \langle v_i^2(t) \rangle , \quad (4.20)$$

où les $\langle \dots \rangle$ indiquent une moyenne temporelle, et on n'oublie surtout pas que notre choix d'unités implique que $m_i \equiv 1$. Pour la simulation de la Fig. 4.7, on trouve $T = 1.208$. Pour un système n'impliquant que peu de particules (genre $N = 16$!), la température moyenne ainsi définie fluctue en général beaucoup si la moyenne temporelle se fait sur un intervalle court, mais des valeurs stables peuvent être extraites si on moyenne sur une période de temps beaucoup plus longue que le temps moyen entre deux collisions (pour une particule-test donnée).

À l'équilibre, on s'attend également à ce que la distribution des vitesses ait la forme d'une distribution de Maxwell-Boltzmann:

$$P(v)dv = T^{-1} \exp(-v^2/2T) v dv \quad (4.21)$$

Notons qu'il s'agit ici de la version normalisée de la distribution, i.e.,

$$\int_0^\infty P(v)dv = 1 , \quad (4.22)$$

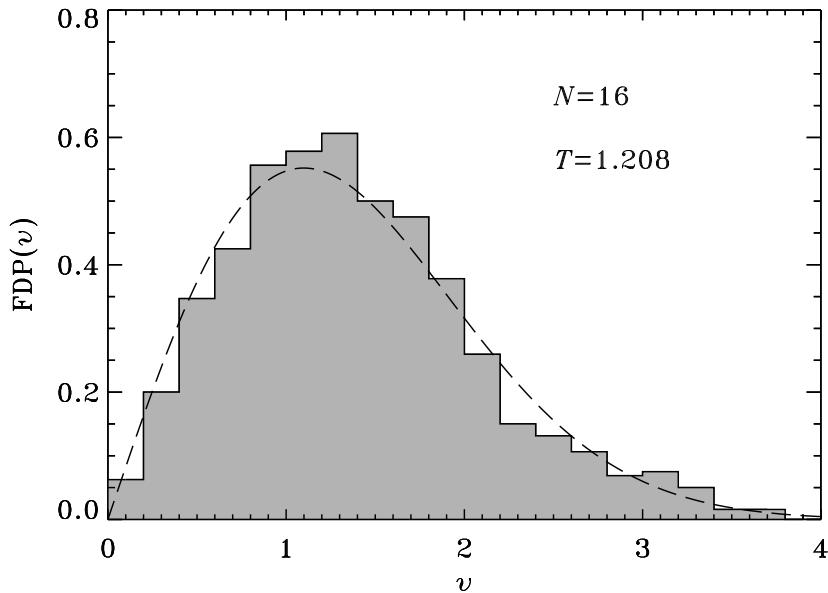


Figure 4.7: Distributions de vitesses dans une simulation $N = 16$ utilisant la même condition initiale que sur la Fig. 4.6. Encore une fois $L = 6$ et on tronque le potentiel de Lennard-Jones à $r/\sigma = 3$. La courbe en tiret est une distribution de Maxwell-Boltzmann (éq. (4.21)) pour $T = 1.208$, cette dernière valeur étant calculée selon l'éq. (4.20).

d'où le préfacteur T^{-1} dans l'éq. (4.21). Le trait en tiret sur la Fig. 4.7 correspond à cette distribution, évaluée pour $T = 1.208$, tel que calculé via l'éq. (4.20). C'est bien une distribution de Maxwell-Boltzmann !! C'est bôôôôô...

4.3.2 Condensation, solidification et liquéfaction

Ayant établi un lien clair entre l'énergie cinétique moyenne de nos particules et la température, il devient possible d'étudier le comportement du système en fonction de la température. Plus spécifiquement, le système peut être “refroidi” ou “réchauffé” et réduisant ou augmentant les vitesses de nos N particules. La Figure montre une série d'instantanés des positions de $N = 81$ particules dans une simulation $L = 20$, initialisée avec des vitesses (orientées aléatoirement) dont la grandeur conduit à une température $T \simeq 1.8$ une fois le système en équilibre. À partir de $t = 20$, à chaque intervalle $\Delta t = 1$ les vitesses de toutes les particules sont artificiellement réduites par un facteur multiplicatif légèrement sous l'unité:

$$v_x \rightarrow 0.99 \times v_x, \quad v_y \rightarrow v_y \times 0.99, \quad (t \bmod 1) = 0 \quad (4.23)$$

ce qui correspond donc à refroidir le système. La Figure 4.8 montre l'évolution du système sous la forme d'instantanés des positions des particules, mais l'animation disponible sur la Page Web du cours est hautement préférable pour mieux en saisir toute la dynamique.

Initialement, avec $T \simeq 1.8$ ici l'énergie cinétique moyenne des particules est plus grande que la profondeur du puit attractif du potentiel de Lennard-Jones. Les interactions entre particules sont dominées par la partie répulsive du potentiel, et à toutes fins pratiques ressemblent à des collisions élastiques entre sphères rigides (comme la trajectoire en rouge sur la Fig. 4.5A). C'est la phase “gazeuse” de la simulation. La force attractive commence à se faire sentir lorsque T chute sous la valeur de l'unité (dans nos unités adimensionnelles), et à $T \simeq 0.75$ les particules commencent à se condenser en petits groupes genre “microgouttelettes”, sous l'influence de la partie attractive du potentiel de Lennard-Jones. On entre ici dans la phase “liquide” de

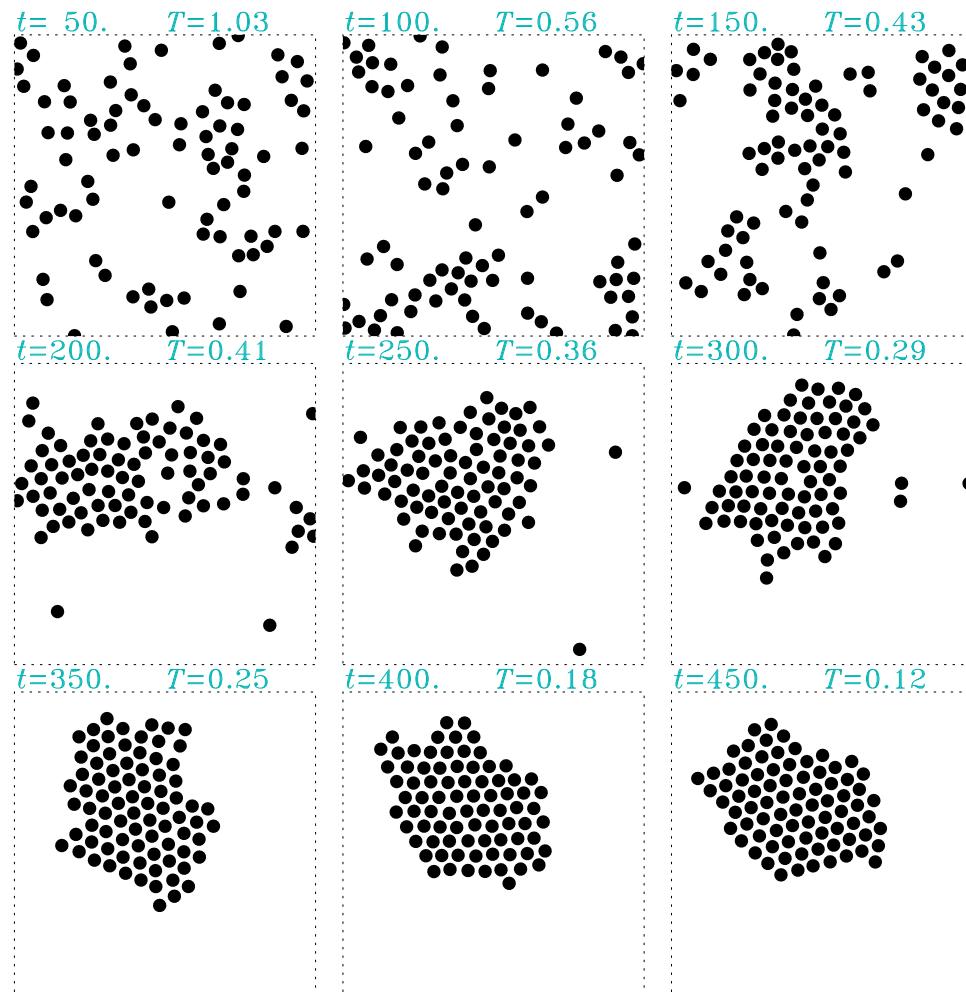


Figure 4.8: Instantanés d'une simulation de condensation/solidification utilisant $N = 81$ particules. Simulation à $L = 20$ et $\Delta t = 10^{-2}$. La température diminue de 1% à toutes les unités de temps adimensionnelles. Une animation de cette simulation est disponible sur la page web du cours.

la simulation. Une fois sous $T \simeq 0.4$ on a formé une “gouttelette” dont la forme change continuellement. Lorsque T chute sous $\simeq 0.25$, la forme de la gouttelette de stabilise, et les particules s’organisent en configuration triangulaire: chaque particule intérieure a six voisins séparés angulairement de $\pi/3$, soit la configuration de densité maximale pour des sphères en 2D. Ici chaque particule est en équilibre sous l’influence de la force attractive de ses six voisins, et l’énergie cinétique se retrouve dans des mouvements oscillatoires par rapport à cette position d’équilibre. L’effet net en est un de vibration dans la structure interne de la gouttelette. On a atteint ici la phase “solide” de la simulation.

Qu’arrive-t-il si on inverse ce processus ? La Figure 4.9 en montre résultat. On débute avec le solide résultant du processus de refroidissement de la simulation précédente (spécifiquement, la structure à $t = 500$, qui est essentiellement identique à celle à $t = 450$ dans le coin inférieur droit de la Fig. 4.8). Encore une fois à chaque intervalle $\Delta t = 1$, soit tous les 100 pas de temps, les vitesses de toutes les particules sont augmentée par un facteur multiplicatif légèrement au delà l’unité, soit plus spécifiquement ici:

$$v_x \rightarrow 1.01 \times v_x , \quad v_y \rightarrow v_y \times 1.01 , \quad (t \bmod 1) = 0 \quad (4.24)$$

On est donc ici en train de réchauffer le système ! La Figure 4.9 en illustre l’évolution, sous la forme d’instantanés des positions des particules, comme sur la Fig. 4.8; et encore ici l’animation disponible via la page web du cours permet de mieux en apprécier la description qui suit.

À mesure que la température augmente le mouvement vibratoire des particules augmente mais la structure conserve son organisation en réseau triangulaire, ainsi que sa forme globale. À $T \simeq 0.25$ les particules à la surface de la gouttelette commencent à se déplacer le long de sa surface, produisant un changement de la forme globale même si les noeuds intérieurs conservent leur configuration triangulaire régulière. Cette déformation de la gouttelette et les déplacements surfaciques s’accentuent inexorablement à mesure qu’augmente la température. Une fois atteint $T \simeq 0.5$, des particules commencent à se détacher de la gouttelette, et celle-ci finit par se désagréger au delà de $T \simeq 2/3$. À ce stade on observe toujours de plus petits regroupements de particules (des “microgouttelettes”) qui persistent parfois durant plusieurs unités de temps, mais une fois au delà de $T \simeq 1$ on se retrouve de nouveau dans une configuration “gazeuse” dominée par les collisions élastiques entre particules individuelles.

Considérant la réversibilité temporelle de nos simulations de dynamique moléculaire (viz. 4.6), on pourrait s’attendre à ce que le processus de condensation/solidification soit réversible, i.e., que la Fig. 4.9 soit équivalente à une inversion temporelle de la Fig. 4.8; il s’avère que ce n’est pas tout à fait le cas, comme le montre la Figure 4.10. On y a porté en graphique l’énergie potentielle par particule versus la température, les deux étant moyennées sur les blocs de 100 pas de temps durant lesquels la température n’est pas artificiellement réduite ou augmentée. Le processus de condensation solidification correspond à un déplacement du coin supérieur droit au coin inférieur gauche le long de la courbe en bleu, et la fonte/évaporation au déplacement dans la direction inverse sur la courbe en rouge. Les courbes sont bruyantes (surtout en température) malgré la moyenne sur 100 pas de temps, mais il demeure clair qu’elles ne se superposent pas; le “retour” de la phase solide à gazeuse durant le réchauffement (courbe rouge) se fait à notablement plus grande énergie potentielle (en valeur absolue) durant la phase liquide, que durant le refroidissement (courbe bleue). L’intégrale de la différence d’énergie entre les deux courbes correspond à la *chaleur latente* du système.

Les termes “solide”, “liquide”, etc, ont systématiquement été mis entre guillemets dans la discussion ci-dessus, car il y a des limites à l’interprétation physique que l’on peut attacher à nos résultats. Cependant, au moins en termes qualitatifs, ce modèle demeure une représentation passablement réaliste du comportement de certains fluides, comme l’Argon liquide.

4.3.3 Fluides stratifiés par la gravité

Notre modèle de dynamique moléculaire peut aussi être utilisé pour explorer (un peu artificiellement) un phénomène se produisant sur des échelles beaucoup plus grandes qu’interatomique, soit la stratification d’un fluide par la gravité. Par soucis de simplicité, on continuera ici de

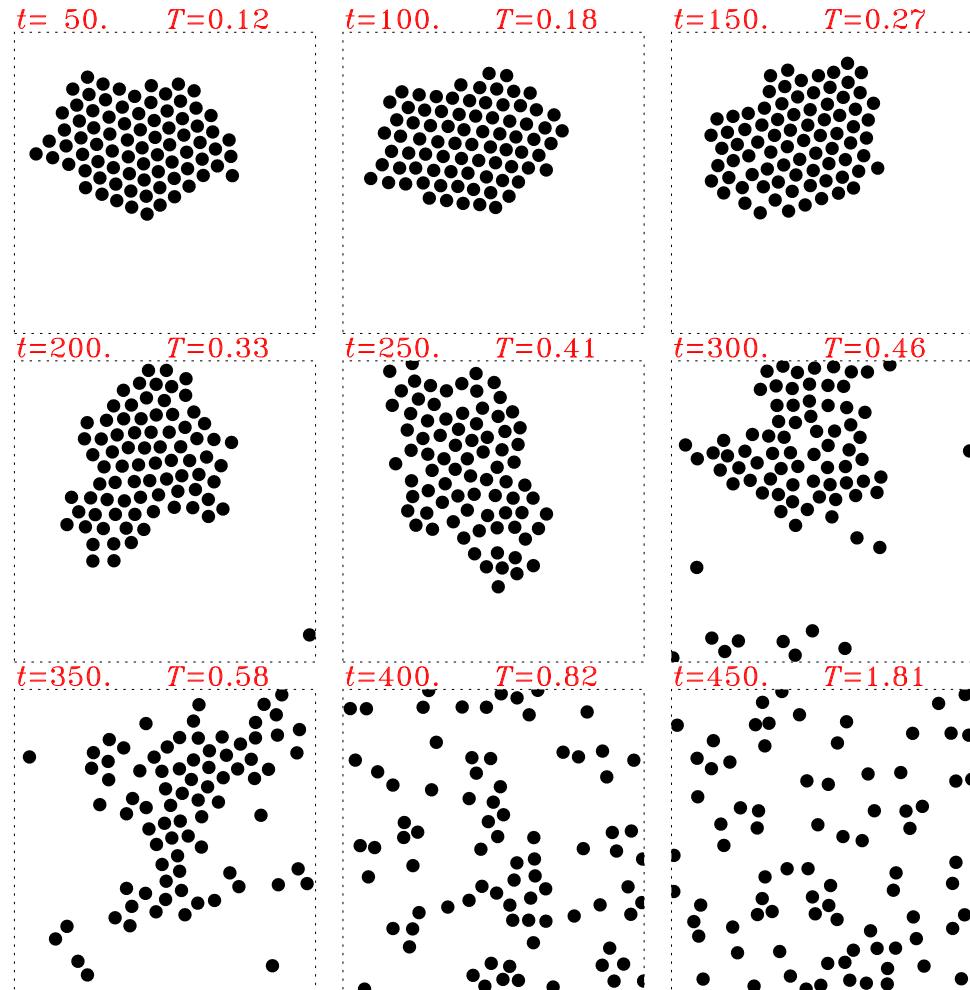


Figure 4.9: Identique en format à la Fig. 4.8, mais cette fois pour une simulation de fonte/évaporation. La température augmente de 1% à toutes les unités de temps adimensionnel. Encore une fois, $N = 81$ et troncation du potentiel à $r/\sigma = 3$. Une animation de cette simulation est disponible sur la page web du cours.

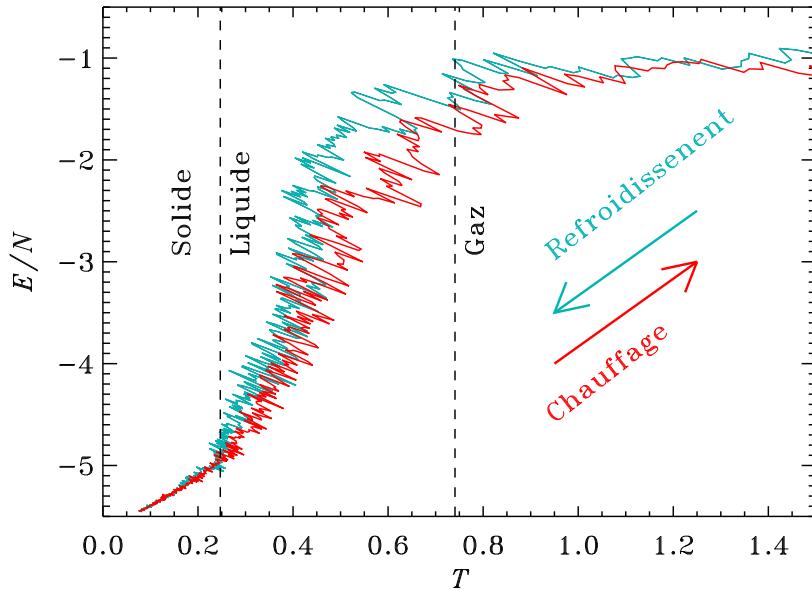


Figure 4.10: Variation de l'énergie potentielle par particule versus la température, pour les simulations de condensation/solidification de la Fig. 4.8 (en bleu) et de fonte/évaporation de la Fig. 4.9 (en rouge). Les températures et énergies sont moyennées sur les blocs de 1 unité de temps, délimités par les hausses ou baisses de températures imposées dans ces simulations. Les températures aux transitions entre les phases “solide”, “liquide” et “gaz” (droites verticales en tirets) sont estimées ici “à l'oeil”.

travailler avec la même version adimensionnalisée du modèle qu'auparavant, sauf qu'il faut imaginer ici qu'il ne s'agit plus ici d'Argon liquide aux alentours de 120 K, mais plutôt d'azote moléculaire à TPN ou pas loin.

On considère maintenant un domaine périodique horizontalement, comme auparavant, mais cette fois dans la verticale on place une “surface” rigide à $y = 0$, et on laisse ouvert le haut du domaine ($y = L$). La fonction `tampon` doit être modifiée pour n'inclure que les tampons latéraux (et sans les coins). Si la coordonnée- y d'une particule chute sous zéro, on introduit maintenant une “réflexion” instantanée, selon l'algorithme:

$$\text{si } y_i \leq 0 : \quad y_i \rightarrow |y_i| , \quad v_{yi} \rightarrow -v_{yi} , \quad (4.25)$$

qui simule assez bien une collision élastique avec une surface plane rigide.

Il s'agit maintenant d'introduire la gravité dans la simulation de dynamique moléculaire. Si on s'en tient à une gravité constante à la surface de la Terre (par exemple), il ne s'agit que d'ajouter l'accélération correspondante à celle résultant du calcul des forces inter-atomiques; i.e., on rajoute entre les lignes 33 et 34 l'instruction suivante:

`fy[:,]=g`

bien alignée avec l'instruction `return` qui suit, pour ne pas se ramasser à l'intérieur de l'une ou l'autre boucle sur les particules ! (la foutue convention d'indentation des boucles dans Python...)

On considère maintenant une série de simulations à $N = 81$ particules dans un domaine $L = 20$, incluant la gravité comme décrit ci-dessus. Avec les σ et ϵ du potentiel de Lennard-Jones comme unités de longueur et d'énergie, la gravité terrestre $g = 9.8 \text{ m s}^{-2}$ devrait être absolument minuscule ici lorsqu'exprimée en unités adimensionnelles (je vous laisse le calculer!).

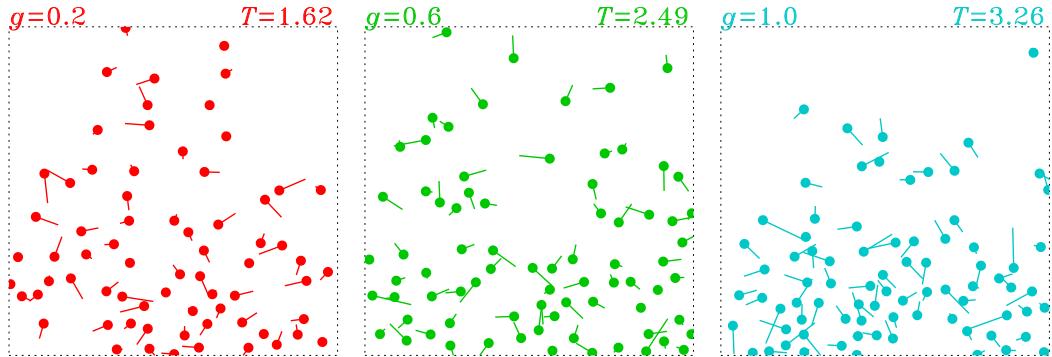


Figure 4.11: Solutions à $t = 100$ pour trois simulations de dynamique moléculaire périodiques horizontalement mais sur “plancher” rigide, avec gravité allant de 0.2 (à gauche) à 1.0 (à droite), tel qu’indiqué. À cause des énergies potentielles gravitationnelles initiales augmentant (linéairement) avec la gravité, à l’équilibre les trois solutions se retrouvent avec des températures différentes (voir texte). Une animation de la solution à $g = 0.6$ est disponible sur la page Web du cours.

Cependant, aux fins purement didactiques de ce qui suit, on utilisera des valeurs de l’ordre de l’unité.

La Figure 4.11 montre la distribution des $N = 81$ particules et leur vecteurs-vitesse à la fin ($t = 10$) de trois simulations utilisant des gravités croissantes, de gauche à droite. Le domaine est ici de taille $L = 20$, et la condition initiale consiste à répartir nos 81 particules aux intersections d’un quadrillage régulier de taille 9×9 , un peu comme le quadrillage 4×4 des conditions initiales sur la Figure 4.6. On observe bien une tendance à l’accumulation des particules au bas du domaine, tendance devenant de plus en plus marquée à mesure que g augmente. Les températures moyennes, telles que calculées via l’éq. (4.20) pour ce seul pas de temps, i.e., sans moyenne temporelle; basé sur seulement 81 particules, ici ces valeurs peuvent donc fluctuer significativement d’un pas de temps à l’autre, dans le genre $\pm 30\%$. Néanmoins, l’augmentation de la température moyenne avec la gravité est attendue, résultant de la plus grande énergie potentielle gravitationnelle présente dans la condition initiale.

Si on divise maintenant le domaine en tranches horizontales et qu’on effectue le décompte du nombre de particules situées dans chacune de ces tranches, on peut calculer un profil de densité en fonction de la hauteur y . Ces déterminations devront se faire en moyennant sur un grand nombre de pas de temps si l’on veut extraire des valeurs stables de nos simulations à $N = 81$ particules, comme on l’avait fait pour les distributions de vitesses sur la Fig. (4.7). La Figure 4.12 montre le résultat d’un tel exercice, pour les trois simulations de la Fig. 4.11. On y constate que la densité chute bien avec la hauteur, et ce de manière un peu plus marquée quand la gravité est plus grande.

Dans une atmosphère hydrostatique à gravité constante, la dynamique se réduit à un équilibre entre le gradient vertical de pression et la gravité:

$$\frac{\partial p}{\partial z} = -\rho g , \quad (4.26)$$

où ρ est la densité (km m^{-3}) de la substance. Pour un gaz parfait isotherme on aurait $p = \rho kT/m$, d'où

$$\frac{\partial \rho}{\partial z} = -\frac{\rho}{H} , \quad \rightarrow \quad \frac{\rho(z)}{\rho_0} = \exp\left(-\frac{z}{H}\right) \quad (4.27)$$

où ρ_0 est la densité à $y = 0$ et $H = kT/mg$ est la *hauteur de colonne*. Le graphique de la Fig. 4.12 étant tracé avec une échelle verticale logarithmique, la pente constante des histogrammes reflète bel et bien une décroissance exponentielle avec l’altitude. Banzai !!

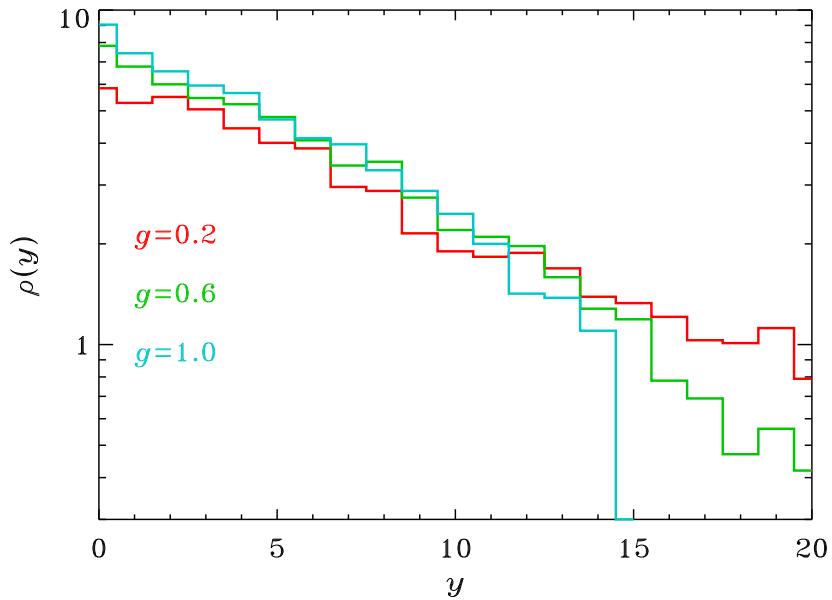


Figure 4.12: Variation de la densité de particule en fonction de la coordonnée verticale y , calculée dans une série de simulations ayant des accélérations gravitationnelles g allant de 0.2 à 1.0, selon le code couleur. Notez l'échelle logarithmique verticale, indiquant qu'ici la densité décroît exponentiellement avec l'altitude.

En terme de nos unités adimensionnalisées, la hauteur de colonne se réduit à $H = T/m$, et donc pour les trois solutions de la Fig. 4.11 on a $H = 8.1, 4.14$ et 3.26 , en passant de $g = 0.2$ (à gauche) à $g = 1$ (à droite). Ceci est cohérent avec l'allure générale de la Fig. 4.11, où l'on constate que la densité de particules chute plus rapidement avec y à mesure que g augmente.

4.4 Matière sombre et rotation des galaxies

On passe maintenant de l'infiniment petit à l'infiniment grand: de la dynamique microscopique des fluides à celles des étoiles composant une galaxie. La Figure 4.13 en montre un exemple particulièrement joli, soit la galaxie NGC6946, située à environ 22 million d'années-lumière de chez nous. À ces distances il est impossible d'y distinguer des étoiles individuelles —sauf si elles explosent en supernova. Toutes les étoiles visibles sur cette photographie font partie de la Voie Lactée. L'émission lumineuse de NC6946 est la composante diffuse, et inclue l'émission radiative des étoiles, du gaz interstellaire, etc.

NGC6946 est catégorisée comme une galaxie de type spirale, pour des raisons qui je l'espère sont évidentes. Ceci suggère un mouvement de rotation dans un plan, qui est en fait observé. Ces observations sont effectuées surtout dans le domaine radio du spectre électromagnétique, prenant avantage du fait que le décalage Doppler induit par la rotation varie en $\Delta\nu/\nu$, et que l'Hydrogène atomique abondamment présent dans le milieu interstellaire galactique émet copieusement dans le domaine radio. Il est en fait possible de détecter le HI bien au delà de l'étendue visible du disque.

Dans une galaxie spirale, comme NGC6946, on suppose qu'on a affaire à N étoiles confinées dans un disque très mince, réduisant ainsi le problème à deux dimensions spatiales. Supposons, en première approximation, que chaque étoile est en orbite circulaire autour du centre galactique. De telles orbites sont caractérisées par un équilibre entre les forces centrifuge et gravitationnelle. Travailant en coordonnées polaires (r, θ) , pour la n -ième étoile, de masse m



Figure 4.13: Image composite visible+infrarouge de la galaxie NGC6946, capturée ici par le télescope Japonais Subaru. Cette galaxie de type spirale est à environ 22 millions d'années-lumière de la Voie Lactée, et est environ trois fois plus petite (en diamètre) que cette dernière. Image téléchargée de apod.nasa.gov/apod/ap120109.html

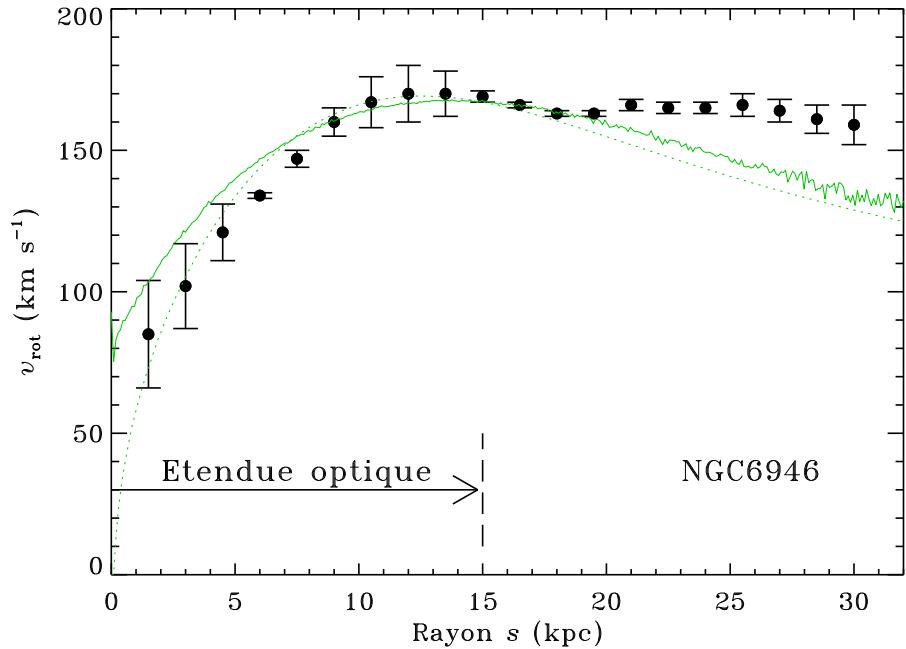


Figure 4.14: Courbe de rotation de la galaxie NGC6946, obtenue par observations radio dans la raie du HI. Les données sont tirées de Carignan et al. (1990, A&A **234**, 43–52). Les courbes en vert correspondent aux courbes de rotation initiale (pointillés) et finale (trait plein) de la simulation discutée à la §4.5.5, légèrement ajustée en amplitude et rayon aux données sous 15 kpc.

et située à un rayon r_n du centre, on a:

$$\frac{m v_\theta^2}{r_n} = \frac{G M(r_n)m}{r_n^2} \quad (4.28)$$

où $M(r_n)$ la masse totale contenue dans la partie du disque intérieure à l'étoile. Si on ne considère que la contribution des étoiles on écrirait simplement:

$$M(r_n) = \sum_{k=1}^N \begin{cases} m_k & \text{si } r_k < r_n \\ 0 & \text{sinon} \end{cases}. \quad (4.29)$$

On pourrait ajouter d'autres contributions, associées à la distribution de gaz, mais pour l'instant on s'en tient aux étoiles. Pour celles en périphérie du disque galactique, l'équation (4.28) se réduit à une orbite Keplérienne classique:

$$v_\theta(r_n) = \left(\frac{G M_G}{r_n} \right)^{\frac{1}{2}}, \quad (4.30)$$

où M_G est la masse totale de la galaxie, maintenant une constante. On peut tirer de ceci la troisième Loi de Kepler, soit la proportionnalité $P_n^2 \propto r_n^3$ entre la période orbitale P_n et le rayon orbital r_n . L'éq. (4.30) prédit donc $v_\theta \propto r^{-1/2}$ en périphérie du disque stellaire et au delà.

La Figure 4.14 montre une courbe de la vitesse de rotation v_θ dans le plan du disque, déduite d'observations radio de la raie HI dans NGC6946. La constance de la vitesse de rotation au delà du disque visible ($r \sim 15$ kPc) indique la présence d'une densité de masse significative à ces distances. Pour que l'éq. (4.28) conduise à v_θ constant à grand r , on doit avoir $M(r) \propto r$,

ce qui implique $\rho_g(r) \propto r^{-2}$ pour un halo sphérique. Il ne peut s'agir uniquement de la masse du halo gazeux produisant l'émission HI; sa densité peut être déduite à partir de l'intensité de l'émission radio observée, et cette densité se retrouve trop faible pour ajouter suffisamment au potentiel gravitationnel pour maintenir la constance de la courbe de rotation jusqu'à 30 kpc (quoiqu'ici elle y contribue tout de même au niveau de 20–30% au delà du disque stellaire, selon les modèles de masse actuels de NCG6946).

Cette situation n'est pas unique à NGC6946, mais comme l'ont démontré l'astronome Vera Rubin et collaborateurs au début des années 1970's, s'est retrouvée observée dans une majorité de galaxies, y compris de types autres que spirale, à mesure que se sont accumulées les observations radio en HI. C'était la première indication non-ambigüe de la présence de *matière sombre* s'étendant au delà de la masse lumineuse dans les galaxies² —la composante stellaire, ainsi que le halo gazeux gaz produisant l'émission HI. Par la suite l'observation de son effet de lentille gravitationnelle a confirmé la grande étendue spatiale des “halos” de matière sombre galactiques, et que la majorité de la masse d'une galaxie typique se retrouve sous la forme de matière sombre, indétectable visuellement ou spectroscopiquement, mais contribuant au potentiel gravitationnel. La nature physique de la matière sombre demeure inconnue à ce jour, et est sujette à moult spéculations et tentatives de détection directe. Il s'agit ici d'un des grands mystères de l'astrophysique contemporaine, et son identification demeure un sujet chaud autant du point de vue de l'astrophysique que de la physique des particules (voir références en fin de chapitre).

4.5 Simulation dynamique d'une galaxie

Il s'agit maintenant de simuler la dynamique d'une galaxie afin d'examiner de plus près cette question de courbe de rotation décroissante (ou pas). En principe il s'agit ici encore une fois d'un problème à N -corps, comme pour la dynamique moléculaire traitée précédemment. Cependant, comme la force gravitationnelle en est une de longue portée ($\propto 1/r^2$), on ne pourra pas s'en tirer en tronquant le potentiel gravitationnel à courte distance, comme on l'avait fait avec Lennard-Jones. Pour des simulations qui compteront maintenant de l'ordre de $N = 10^5$ — 10^6 particules, il est clairement hors de question de commencer à calculer $N^2/2$ distances à chaque pas de temps! Les méthodes de simulation à N -corps de type “particle-in-cell” (qu'on peut très libéralement traduire par “particule-en-boîte”) permettent de s'en tirer à pas mal meilleur compte. Voyons comment.

4.5.1 Adimensionalisation

Introduisons tout d'abord un choix d'unités physiques appropriées au problème. On se limitera ici à simplement choisir des unités de longueur, temps, et masse, après quoi on exprimera les constantes physiques du problème dans ces unités. Le tableau 4.2 compile un choix pertinent ici.

Tableau 4.2: Choix d'unités physiques

Quantité	Unité	(SI)
Masse	Masse du soleil (M_\odot)	1.99×10^{30} kg
Longueur	kiloParsec (kpc)	3.086×10^{19} m
Temps	Rotation galactique (P_\odot)	7.5×10^{15} s

L'échelle de temps P_\odot mérite une explication; c'est simplement la période de rotation du Soleil autour du centre galactique (évaluée en fait entre 225 et 250 millions d'années). Notons

²Sa présence avait été suggérée dans les amas galactique 40 ans plus tôt, sous l'hypothèse que ces amas devaient être gravitationnellement liés, ce qui demandait beaucoup plus de masse que celle des galaxies de l'amas.

que dans ces unités, la constante gravitationnelle prend la valeur:

$$G = 6.67 \times 10^{-11} \frac{\text{m}^3}{\text{kg s}^2} \equiv 2.277 \times 10^{-7} \frac{\text{kpc}^3}{M_{\odot} \text{P}^2} \quad (4.31)$$

4.5.2 Conditions limites et initiales

Le problème est solutionné dans un domaine carré d'étendue $x, y \in [-L, L]$, avec $L = 30$ (kpc) dans tout ce qui suit. Occasionnellement une particule-étoile peut se retrouver à sortir du domaine; quand cela se produit, on la fait simplement “rebondir” vers l'intérieur. Avec v_r la vitesse de la particule dans la direction radiale et (x, y) sa position, on impose:

$$v_r \rightarrow -v_r \text{ et } \begin{cases} x \rightarrow 2L - x & \text{si } x > L \\ x \rightarrow -2L - x & \text{si } x < -L \\ y \rightarrow 2L - x & \text{si } y > L \\ y \rightarrow -2L - x & \text{si } y < -L \end{cases} \quad (4.32)$$

La spécification des conditions initiales s'avère encore une fois délicate. Il se s'agit pas simplement de “lancer” N particules-étoile dans la simulation, avec des vitesses aléatoires, et espérer former un système gravitationnellement lié; la majorité des étoiles se feraient éventuellement éjecter du système.

Aux fins de nos simulations, une condition initiale acceptable consiste à distribuer nos particules-étoiles selon un profil circulairement symétrique avec concentration centrale, et leur assigner une vitesse angulaire correspondant à une orbite circulaire Képlérienne, telle que donnée par l'éq. (4.30), avec M correspondant à la masse *intérieure* au rayon de l'orbite circulaire de chaque particule. Pour ce faire il sera pratique de diviser le domaine en N_a anneaux concentriques contigus de largeur Δr centrés sur notre galaxie. On dénotera par $\Delta M(r_k)$ la masse contenue dans le k -ième anneau, et par $M(r_k)$ la masse contenue dans la portion du disque intérieur au rayon r_k de chaque anneau. Le fragment de code présenté sur la Figure 4.15 effectue cette initialisation. La procédure est la suivante:

1. La première boucle (lignes 13–18) extrait les coordonnées cartésiennes (x, y) de chacune des N particules-étoile de distribution gaussiennes de largeur à mi-hauteur r_D ; la distance radiale au centre est ensuite calculée (ligne 16), après quoi la particule est assignée à l'anneau correspondant (ligne 17): l'indice est entreposé dans le tableau `iann[N]`, et sa contribution à la masse ΔM (tableau `deltaM[N_a]`) de l'anneau comptabilisée (ligne 18);
2. La seconde boucle (lignes 21–23) calcule la masse totale (tableau `Mr[N_a]`) située sous chaque anneau, incluant la masse de ce dernier. La fréquence angulaire de rotation Képlérienne correspondant à cette masse est aussi calculée (ligne 23).
3. La troisième et dernière boucle calcule finalement les vitesses radiale (`v_r`) et azimutale (`v_a`) de chaque particule-étoile, en supposant pour chacune une orbite Képlérienne circulaire.

Notons qu'ici chaque particule-étoile a une masse de $10^5 M_{\odot}$; une petite galaxie comme celle que l'on tente de simuler ici compterait typiquement quelques 10^{11} étoiles, pas question de faire une simulation à $N = 10^{11}$ particules! En augmentant artificiellement la masse de nos particules, on se retrouvera avec une dynamique galactique raisonnable, avec une courbe de rotation qui, on le verra, est assez proche de celle de NGC6946 sur la Fig. 4.14, même avec “seulement” 10^5 – 10^6 particules. Le calcul de la “masse” d'une particule-étoile (variable `metoile`, ligne 4) est tel que la masse totale de la galaxie sera toujours $10^{11} M_{\odot}$, quelque soit le N choisi. La *densité surfacique* (σ , en M_{\odot}/kpc^2) se calcule facilement une fois la masse dans chaque anneau connue:

$$\sigma(r_k) = \frac{\Delta M}{2\pi r_k \Delta r}, \quad (4.33)$$

```

1 Ggrav=2.277e-7          # G en kpc**3/ M_sol / P**2
2 N=1000000                # nombre de particules-etoiles
3 r_D=6                     # largeur du disque gaussien, en kpc
4 metoile=1.e6/(N/1.e5)     # masse d'une particule-etoile
5 x,y,rayon=np.zeros(N),np.zeros(N),np.zeros(N)

6
7 N_a=500                  # nombre d'anneaux
8 deltaM=np.zeros(N_a)       # masse dans chaque anneau
9 Mr =np.zeros(N_a)          # masse cumulative sous chaque anneau
10 Omega =np.zeros(N_a)        # vitesse angulaire (Keplerienne)
11 iann =np.zeros(N,dtype='int') # anneau pour chaque particule
12 dr=6*r_D/N_a              # largeur radiale des anneaux
13 for k in range(0,N):      # initialisation des positions
14     x[k]=np.random.normal(0.,r_D)
15     y[k]=np.random.normal(0.,r_D)
16     rayon[k]=np.sqrt(x[k]**2+y[k]**2)
17     iann[k]=int(rayon[k]/dr) # anneau ou tombe chaque particule
18     deltaM[iann[k]]+=metoile # incremente masse dans cet anneau

19
20 Mr[0]=deltaM[0]
21 for j in range(1,N_a):    # masse cumulative sous chaque anneau
22     Mr[j]=Mr[j-1]+deltaM[j]
23     Omega[j]=np.sqrt( Ggrav*Mr[j]/(j*dr)**3 ) # Eq. (4.28)
24 Omega[0]=Omega[1]

25
26 v_r=np.zeros(N)           # vitesse radiale
27 v_a=np.zeros(N)           # vitesse azimutale
28 for k in range(0,N):      # initialisation des vitesses
29     v_r[k]=0.               # orbites circulaires, v_r=0
30     v_a[k]=Omega[iann[k]]*rayon[k]

```

Figure 4.15: Prescription d'une condition initiale à l'équilibre Képlérien pour un ensemble de N particules-étoiles distribuées spatialement de manière Gaussienne dans le plan cartésien (voir texte).

où r_k est le rayon central du k -ième anneau.

La Figure 4.16 montre les profils radiaux des diverses quantités définies ci-dessus pour notre condition initiale Képlérienne incluant ici $N = 10^6$ particules distribuées avec une largeur de Gaussienne $r_D = 6$ kpc, et utilisant 500 anneaux pour le calcul des profils de masse. La partie (A) montre la distribution de la masse, tandis qu'en (B) on porte en graphique la vitesse angulaire Képlérienne Ω (en rouge), ainsi que la moyenne des vitesses azimutales dans chaque anneau, v_{rot} , en fonction de r , toujours pour 500 anneaux, reconstruites à partir de la position des N particules.

4.5.3 Calcul du potentiel gravitationnel

On arrive finalement à cette fameuse histoire de “particules en boîte”. L'idée, en une phrase, est de calculer *localement* les forces agissant sur chaque particules via le potentiel gravitationnel produit par l'ensemble de N particules formant notre “galaxie”.

Le graphique au haut de la Figure 4.17 montre la distribution spatiale de $N = 10^5$ particules-étoiles dans un domaine 2D cartésien $(x, y) \in [-L, L]$ avec $L = 30$. La première étape consiste à diviser le domaine en cellules contigues à l'aide d'un quadrillage régulier d'intervalle $\Delta x =$

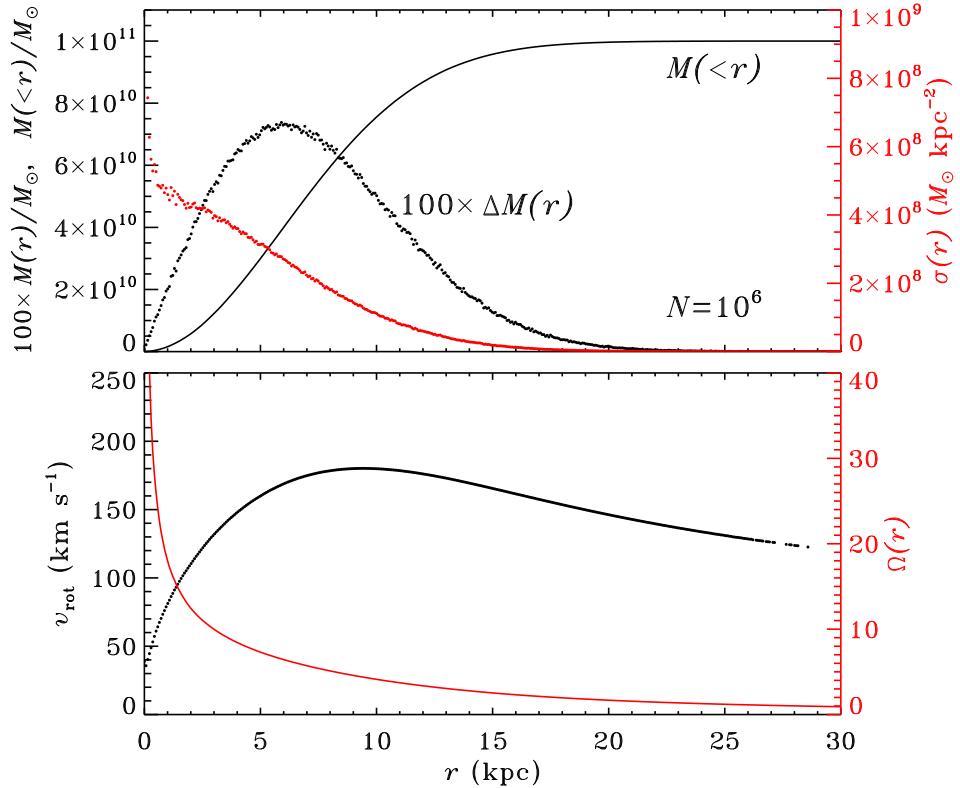


Figure 4.16: Conditions initiales en quasi-équilibre Képlérien, pour une simulation de $N = 10^6$ particules. La partie (A) montre la distribution radiale de la masse (noir) et de la densité surfacique (rouge), telle que produite par la procédure décrite dans le texte. La partie (B) montre le profil radial de la vitesse de rotation (noir) et de la vitesse angulaire (rouge), correspondant à un profil Képlérien en équilibre. Les paramètres définissant cette condition initiale sont données à la Fig. 4.15.

$\Delta y \equiv \Delta = 2L/M$, tel qu'indiqué en rouge. Définissons les lignes de quadrillage des $M \times M$ cellules comme:

$$x_i = i \times \Delta, \quad y_j = j \times \Delta, \quad i, j = 0, \dots, M \quad (4.34)$$

Notez bien: $M + 1$ lignes de quadrillage sont requises pour définir M cellules, dans chaque dimension spatiale. Ensuite, on compte simplement le nombre $n(k, l)$ de particules dans chaque cellule, et on définit la densité surfacique σ dans la cellule (k, l) comme:

$$\sigma(\mathbf{x}_k, \mathbf{y}_l) = \frac{n \times m}{\Delta^2} \quad (4.35)$$

où m est la masse associée à une particule, et les $\mathbf{x}_k, \mathbf{y}_l$ définissent un **second quadrillage** dont les intersections se retrouvent au centre des cellules:

$$x_k = (k + 1/2) \times \Delta, \quad y_l = (l + 1/2) \times \Delta, \quad k, l = 0, \dots, M - 1 \quad (4.36)$$

La localisation de chacune des N particules dans l'une ou l'autre des $M \times M$ cellules utilisées pour définir la densité peut s'effectuer efficacement en prenant avantage du fait que ces cellules définissent un quadrillage régulier. Comprenez bien comment les instructions Python suivantes

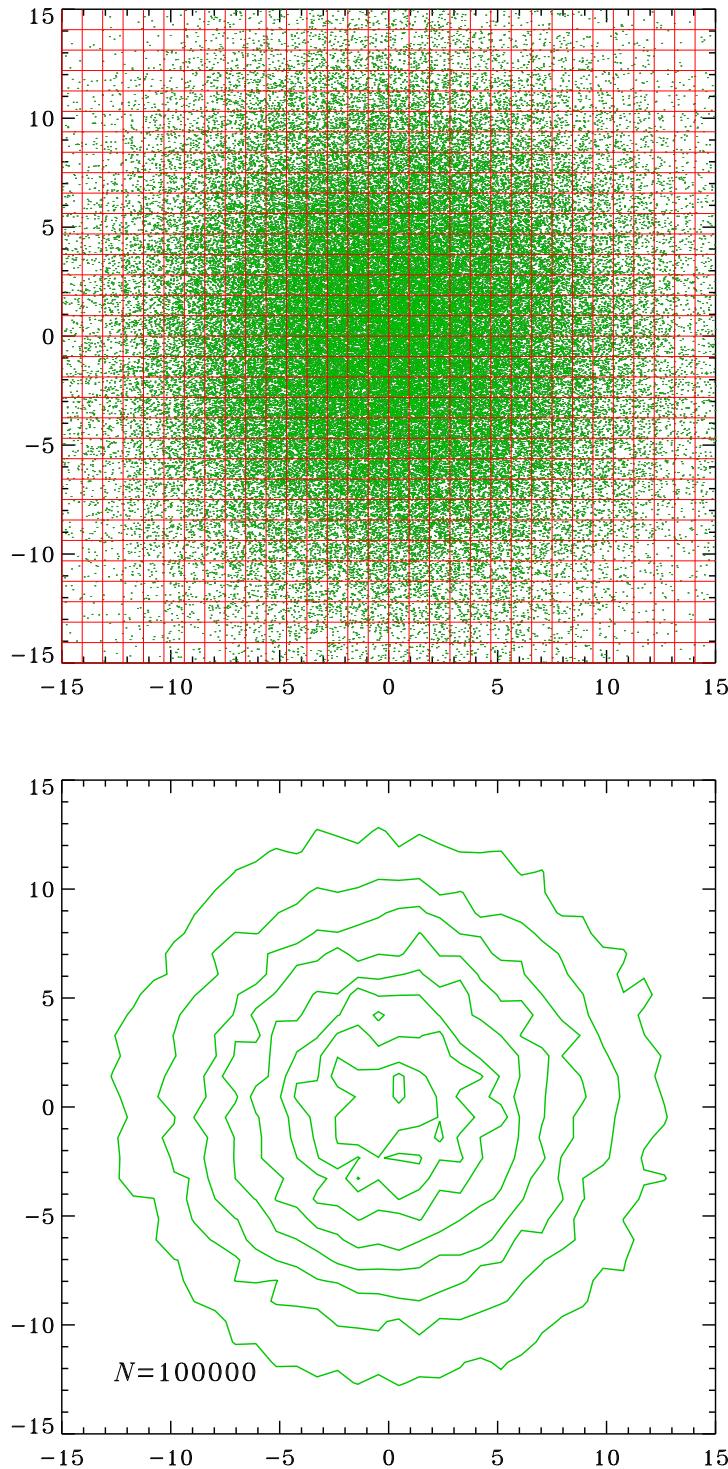


Figure 4.17: Distribution spatiale de $N = 10^5$ particules-étoiles (en haut), et isocontours de la densité surfacique y correspondant (en bas). La densité est calculée via le décompte des particules situées dans chaque case d'un quadrillage régulier en x et y , indiqué en rouge sur le diagramme du haut. Le domaine total couvre ici $x, y \in [-L, L]$ avec $L = 30\text{ kpc}$, et le quadrillage utilisé pour calculer la densité compte ici $M \times M = 64 \times 64$ cellules.

calculent les coordonnées de cellules (k, l) où se trouvent les n particule dont les positions ($\in [-L, L]$) sont emmagasinées dans les tableaux $x[N]$ et $y[N]$; la densité $\text{sigma}[k, 1]$ de cette cellule est ensuite incrémentée en conséquence:

```
sigma=np.zeros([M,M])      # densite sur M X M cellules, initialisee a zero
for n in range(0,N):        # boucle sur les N particules
    k=int( (x[n]+L)/Delta ) # numero de cellule en x
    l=int( (y[n]+L)/Delta ) # numero de cellule en y
    sigma[k,l]+=metoile    # cumul de la masse dans la cellule
sigma/=Delta^2              # conversion en densite surfacique
```

Le diagramme du bas sur la Figure 4.17 montre les isocontours de densité correspondant à la distribution de particules sur le diagramme du haut. Avec seulement $N = 10^5$ particules ici, ces isocontours de densité ne sont pas très lisses, mais reproduisent néanmoins la forme circulaire de la distribution de particules.

La prochaine étape consiste à calculer le potentiel gravitationnel associé à cette distribution en densité. Ceci est défini, de manière tout à fait générale, par l'intégrale:

$$\Phi(\mathbf{r}) = - \int_V \frac{G\sigma(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dS' . \quad (4.37)$$

Attention, ici l'intégrale se fait sur la coordonnée \mathbf{r}' mesurant la position de l'élément de masse $\sigma(\mathbf{r}')dS'$, et définit le potentiel à une position \mathbf{r} . L'évaluation numérique de cette intégrale n'est pas triviale et mérite qu'on s'y attarde. L'idée centrale, illustrée à la Figure 4.18 est d'utiliser les valeurs de densités calculées au centre des cellules (points vert) pour calculer le potentiel gravitationnel aux coins des cellules (points bleus), soit les intersections du quadrillage primaire définissant les frontières des cellules (en rouge ici et sur la Fig. 4.17). Si on considère la densité constante dans chacune des cellules, alors l'intégrale ci-dessus évaluée numériquement (en deux dimensions spatiales) devient une double somme:

$$\Phi(\mathbf{x}_i, \mathbf{y}_j) = - \sum_{k=0}^{M-1} \sum_{l=0}^{M-1} \frac{G\sigma(\mathbf{x}_k, \mathbf{y}_l)}{((\mathbf{x}_i - \mathbf{x}_k)^2 + (\mathbf{y}_j - \mathbf{y}_l)^2)^{1/2}} \Delta^2 , \quad i, j = 0, \dots, M . \quad (4.38)$$

Notez bien qu'à $M \times M$ cellules sont associées $(M + 1) \times (M + 1)$ intersections, et n'oubliez pas que Python commence la numérotation des tableaux à l'indice zéro ! La Figure 4.19 montre des isocontours (en bleu) du potentiel gravitationnel calculé de cette façon, cette fois sur le domaine complet $(x, y) \in [-L, L]$. Les isocontours du potentiel sont beaucoup plus lisses que ceux de la densité (cf. Fig. 4.17), conséquence de la nature intégrale du potentiel, qui "moyenne" les irrégularités de l'intégrand. On peut aussi vérifier ici qu'à grande distance, on a bien $\Phi(r) = -G(N \times m)/r$. Notez bien que le fait que l'on calcule le potentiel aux coins des cellules à partir des densités définies aux centres de ces mêmes cellules garantit ici que l'on n'aura jamais $|\mathbf{r} - \mathbf{r}'| = 0$ au dénominateur de l'intégrand dans l'éq. (4.37). De plus, le calcul des distances au dénominateur de la forme discréteisée de l'intégrale peut se faire directement en fonction des indices des deux quadrillages; puisque $\mathbf{x}_k = \mathbf{x}_i + \Delta/2$ et idem pour y , on a

$$\frac{d^2(\mathbf{x}_i, \mathbf{y}_j, \mathbf{x}_k, \mathbf{y}_l)}{\Delta^2} \equiv \frac{(\mathbf{x}_i - \mathbf{x}_k)^2 + (\mathbf{y}_j - \mathbf{y}_l)^2}{\Delta^2} \equiv (i - k - 1/2)^2 + (j - l - 1/2)^2 \quad (4.39)$$

Le fragment de code Python suivant effectue cette intégrale du potentiel:

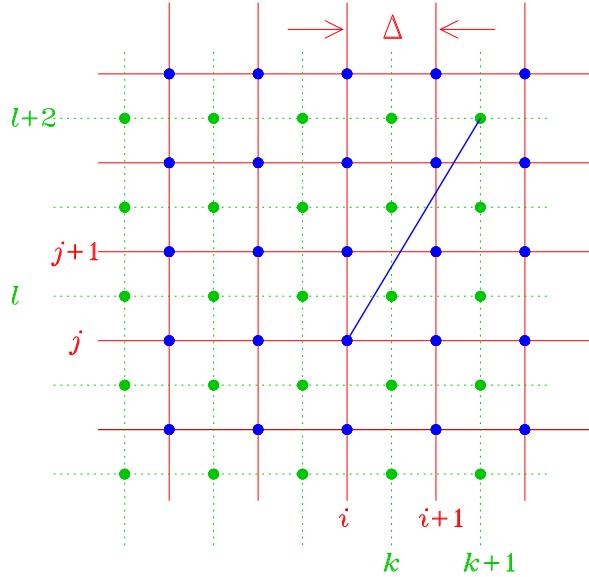


Figure 4.18: Double quadrillage utilisé pour le calcul du potentiel gravitationnel. Les coins des $M \times M$ cellules (en bleu) définissent une grille (i, j) de taille $(M + 1) \times (M + 1)$, tandis que les centres des cellules définissent une grille (k, l) de taille $M \times M$ décalée horizontalement et verticalement de la première par une distance $\Delta/2$. Le segment de droite bleu indique la distance $d(x_i, y_j, x_{k+1}, y_{l+2})$ dans la notation de l'éq. (4.39).

```

pot=np.zeros([M+1,M+1])                      # re-initialisation du potentiel
for k in range(0,M):                          # boucles sur MXM cellules
    for l in range(0,M):
        if sigma[k,l] > 0.:                  # cellule ne contribue pas si vide
            for i in range(0,M+1):           # boucles sur (M+1)X(M+1) coins
                for j in range(0,M+1):
                    d=np.sqrt((i-k-0.5)**2+(j-l-0.5)**2) # distance d(i,j,k,l)
                    pot[i,j]+=sigma[k,l]/d # contribution a l'integrale
pot*=(-Ggrav*metoile*Delta)                   # les constantes a la fin

```

La multiplication finale par Δ vient du fait que l'élément de surface impliqué dans l'intégrale est $\propto \Delta^2$, mais la distance telle que calculée à l'intérieur des boucles est exprimée en unités de Δ ; on évite ici des opérations de multiplication et division inutiles en multipliant par $\Delta^2/\Delta = \Delta$ seulement à la toute fin, à l'extérieur des quatre boucles imbriquées. Pour cette même raison, la multiplication par G et m est également effectuée à l'extérieur des boucles. Notez aussi que la structure des boucles ci-dessous calcule la contribution de la densité de chaque cellule (les deux boucles externes, en k, l) à chaque noeud i, j via les deux boucles internes, ces dernières étant exécutées seulement si la densité surfacique dans la cellule (k, l) est non-nulle. Toutes ces manœuvres servent à réduire le nombre d'opérations arithmétiques impliquées dans le calcul du potentiel, car les opérations internes aux quatre boucles sont effectuées ici $\simeq M^4$ fois, en principe à chaque pas de temps ! Comprenez bien tout ça avant de passer à la suite; le coeur du projet consistera effectivement à ajouter une contribution additionnelle à ce potentiel gravitationnel.

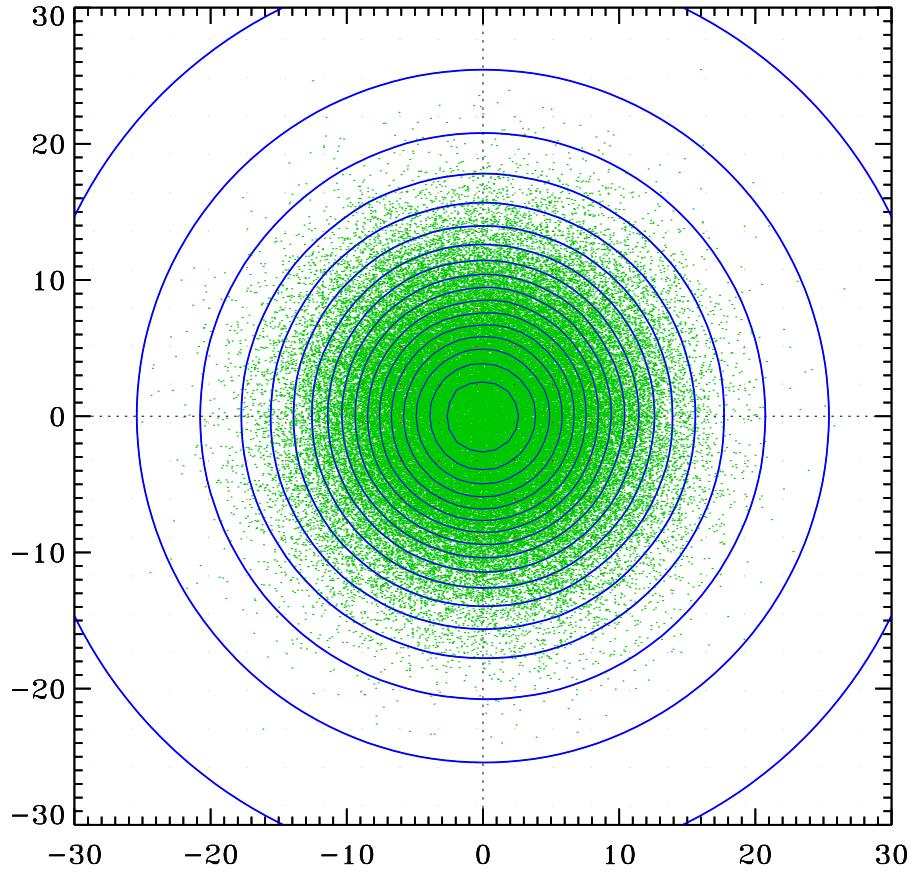


Figure 4.19: Isocontours du potentiel gravitationnel (en bleu) associé à la distribution de particules de la Figure 4.17, tel que calculé par l’intermédiaire de la densité dans chaque cellule du quadrillage des particules. Malgré l’aspect très bruité de la densité (cf. Fig. 4.17), les isocontours sont ici presque parfaitement circulaires.

La dernière étape consiste à évaluer la force gravitationnelle au centre des cellules via $\mathbf{F} = -\nabla\Phi$. Ce calcul se fait par le biais d’une *interpolation bilinéaire* du potentiel; i.e., on écrit, pour chaque cellule (k, l) :

$$\Phi^*(x^*, y^*) = \sum_{m=0}^1 \sum_{n=0}^1 \Phi(\textcolor{red}{x}_{k+m}, \textcolor{red}{y}_{l+n}) \phi_m(x^*) \phi_n(y^*) , \quad (4.40)$$

où le potentiel est déjà connu aux coins des cellules, et (x^*, y^*) est la position par rapport au coin inférieur gauche de la cellule (k, l) , normalisée sur l’intervalle $[0, 1]$ en x et y ; autrement dit, pour une particule positionnée à (x, y) , on aurait:

$$x^* = (x \bmod \Delta)/\Delta , \quad y^* = (y \bmod \Delta)/\Delta . \quad (4.41)$$

Les fonctions d’interpolation, quand à elles, sont définies comme:

$$\phi_0(x) = (1 - x) , \quad \phi_1(x) = x , \quad (4.42)$$

et idem en y ; notons qu’une telle interpolation garantit que $\Phi^*(x_k, y_l) = \Phi(x_k, y_l)$. Comme ces fonctions ont une forme analytique, les dérivées du potentiel à (x^*, y^*) se calculent directement

en fonction des dérivées de ces fonctions d'interpolation, e.g.,

$$\frac{\partial \Phi^*(x^*, y^*)}{\partial x} = \sum_{m=0}^1 \sum_{n=0}^1 \Phi(\textcolor{red}{x_{k+m}}, \textcolor{red}{y_{l+n}}) \frac{\partial \phi_m(x^*)}{\partial x} \phi_n(y^*) , \quad (4.43)$$

et tient pour $\partial \Phi / \partial y$. Développant explicitement la double somme apparaissant dans l'éq. (4.43), et dans son homologue pour F_y , après un peu d'algèbre on arrive à:

$$F_x \equiv -\frac{\partial \Phi(x^*, y^*)}{\partial x} = \quad (4.44)$$

$$- \left[[(\Phi(x_{k+1}, y_l) - \Phi(x_k, y_l))(1 - y^*) + (\Phi(x_{k+1}, y_{l+1}) - \Phi(x_k, y_{l+1}))y^*] \right] / \Delta , \quad (4.45)$$

$$F_y \equiv -\frac{\partial \Phi(x^*, y^*)}{\partial y} = \quad (4.46)$$

$$- \left[[(\Phi(x_k, y_{l+1}) - \Phi(x_k, y_l))(1 - x^*) + (\Phi(x_{k+1}, y_{l+1}) - \Phi(x_{k+1}, y_l))x^*] \right] / \Delta . \quad (4.47)$$

La division par Δ est requise ici car l'interpolation (4.40) est définie sur un carré unitaire en (x, y) . En Python, ça pourrait avoir l'air de ceci:

```
xs=(x % Delta)/Delta      # coordonnees relatives dans cellule
ys=(y % Delta)/Delta
ix=np.trunc((x+L)/Delta)  # coin de la cellule correspondante
iy=np.trunc((x+L)/Delta)
for n in range(0,N):      # boucle sur les particules
    f_x[n]=-( (pot[ix[n]+1, iy[n]]-pot[ix[n], iy[n]])*(1-ys[n]) + ...etc...
    f_y[n]= ...etc...
```

Notez qu'on calcule tout d'abord la position normalisée (x^*, y^*) de chaque particule dans chaque cellule dont le coin est donné par la paire de tableaux (ix, iy) . Il ne s'agit plus que d'invoquer le sieur Newton sous la forme $\mathbf{a} = \mathbf{F}/m$, et appliquer l'algorithme de Verlet (4.9)–(4.14) pour avancer d'un pas de temps les positions et vitesses des particules.

Il y a encore ici matière à réflexion; les particules situées près du centre galactique décrivent des orbites de très petit rayon, ce qui est difficile à suivre avec précision en coordonnées cartésiennes à moins d'utiliser un très petit pas de temps. Il est alors judicieux d'implémenter l'algorithme de Verlet en coordonnées polaires, même si le calcul du potentiel se fait en coordonnées cartésiennes:

```
# conversion cartesien a polaire
rayon=np.sqrt(x^2+y^2)
angle=np.atan(y,x)
f_r=( np.cos(angle)*f_x+np.sin(angle)*f_y + v_a**2/rayon )
f_a=(-np.sin(angle)*f_x+np.cos(angle)*f_y/rayon + v_r*v_a/rayon**2)
# algorithme de Verlet
rayon+=v_r*dt+0.5*(f_r0/metoile)*dt^2
angle+=v_a*dt+0.5*(f_a0/metoile)*dt^2
v_r +=dt*(f_r0+f_r)/2.
v_a +=dt*(f_a0+f_a)/2.
f_r0,f_a0=f_r,f_a
# reconversion des positions en cartesien
x,y=np.cos(angle)*rayon,np.sin(angle)*rayon
```

Toutes ces opérations étant effectuées “vectoriellement” sur toutes les composantes des tableaux impliqués. Notez bien la transformation des composantes cartésiennes f_x, f_y de la force gravitationnelle; le passage aux coordonnées polaires fait apparaître des termes dits métriques impliquant les vitesses radiales et azimutales, provenant de la variation temporelle des vecteurs

unitaires associés au système polaire lorsqu'on suit une particule dans son mouvement. Ce sont les équivalents des forces centrifuge et Coriolis. Donc, les positions et le calcul du potentiel et des composantes de la force gravitationnelle se font en coordonnées cartésiennes, mais la dynamique est calculée en coordonnées polaires; hybride à prime abord curieux, mais qui fonctionne très bien!

4.5.4 Astuces algorithmiques

Les étapes de calcul décrites ci-dessus devront être effectuées à tous les pas de temps; on doit donc bien réfléchir à minimiser le nombre d'opération arithmétiques requises. Si on repasse sur les étapes de calcul devant être répétées à chaque pas de temps, il est clair que la plus exigeante en temps de calcul est le calcul du potentiel aux $(M + 1) \times (M + 1)$ coins des $M \times M$ cellules; c'est un calcul qui augmente en M^4 , et typiquement, pour une bonne résolution du potentiel, on voudrait $M \sim 10^2$, afin d'avoir (en moyenne) au moins 10^2 particules par cellules pour une simulation à $N = 10^6$. On trouve alors que le calcul du potentiel gobe plus de 80% du temps de calcul par pas de temps.

Pour un pas de temps judicieusement choisi, la majorité des particules-étoiles traverseront une cellules en plusieurs pas de temps; on s'attend donc que la distribution spatiale de la densité surfacique, et donc du potentiel gravitationnel, évolue très lentement. Il devient alors possible de recalculer la densité et le potentiel à une cadence plus faible, par exemple tous les dix pas de temps, sans affecter significativement l'évolution galactique. C'est une stratégie qui vous sauvera beaucoup de temps, une simulation typique comme celles décrite ci-dessous pouvant compter plusieurs milliers de pas de temps.

4.5.5 Quelques résultats représentatifs

Un premier exercice de validation consiste ici à évoluer notre galaxie à partir de la condition initiale de la Fig. 4.16, sur un intervalle de temps de l'ordre d'une période de rotation. Toutes les particules devraient gentiment tourner sur une orbite circulaire, sans changer leur distance au centre, tout en conservant leur vitesse azimutale. Obligatoire!

Une condition initiale plus réaliste consiste en fait à introduire une composante aléatoire de faible amplitude à la vitesse initiale des particules. Ceci peut se faire, par exemple, en remplaçant les lignes 29 et 30 à la Fig. 4.15 par les instructions:

```
vrot =Omega[iann[k]]*rayon[k]
v_r[k]=np.random.normal(0.,vrot*0.1)
v_a[k]=np.random.normal(0.,vrot*0.1) +0.95*vrot
```

Autrement dit, la composante aléatoire de la vitesse est de l'ordre de 10% de la vitesse orbitale. Cette condition initiale n'est plus en équilibre Képlérien, en conséquence de quoi les particules décriront maintenant des orbites elliptiques d'excentricité diverses, variant d'une particule à l'autre. Afin d'éviter de produire ainsi trop de trajectoires non-liées, la partie circulaire de la vitesse orbitale est réduite à 95% de sa valeur d'équilibre.

La Figure 4.20 montre quelques exemples de trajectoires choisies aléatoirement parmi les $N = 10^5$ particules-étoiles d'une simulation utilisant un quadrillage de $M \times M = 81 \times 81$ pour le calcul de la densité et du potentiel, et couvre un intervalle de $t/P = 0.5$ avec 2000 pas de temps de $\Delta = 2.5 \times 10^{-4} P$. L'étendue initiale de la distribution de particules est $r_D = 12$ kpc, sur un domaine de demie-largeur $L = 30$ kpc, comme sur la Fig. 4.17. La distribution finale des particules-étoile est tracée à la fin de la simulation (points noirs). Remarquez comment l'orbite rouge, qui est quasi-circulaire, décrit une spirale vers le centre galactique, manifestation et conséquence de la tendance à former une condensation centrale près du centre galactique.

Les orbites n'étant plus circulaires, le profil de densité surfacique et le potentiel gravitationnel évolueront donc vers une distribution de particules (et de vitesses) différent des distributions initiales. Un résultat représentatif est présenté à la Figure 4.21, pour une simulation identique à celle de la Fig. 4.20, sauf que cette fois on a $N = 10^6$. On a porté en graphique ici les

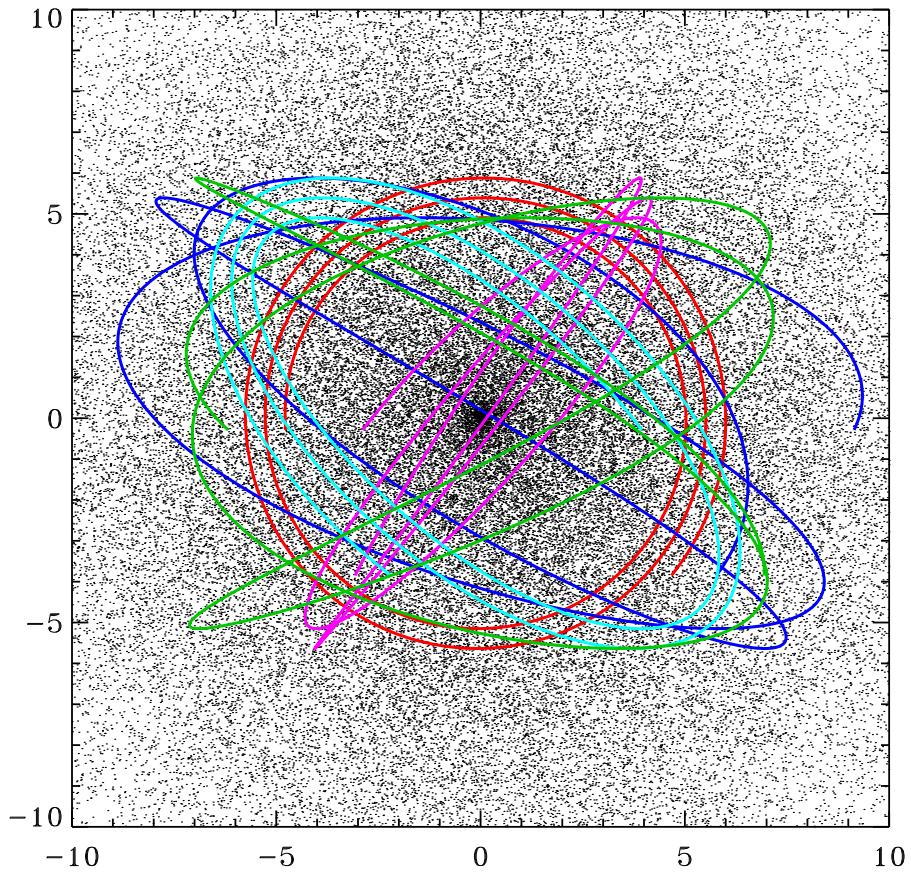


Figure 4.20: Trajectoires de 5 particules-étoiles choisies aléatoirement dans une simulation incluant une composante thermique à la vitesse initiale. Certaines particules conservent une orbite relativement circulaire, tandis que d'autres se retrouvent sur des orbites à très haute excentricité (voir texte). Les points noirs correspondent aux positions des particules-étoile à la fin de la simulation, soit $t/P = 0.5$; Notez la condensation centrale près du “centre galactique”.

profils radiaux initiaux (trait plein) et finaux (points) de la vitesse azimutale moyennées sur des anneaux concentriques comme auparavant (en noir), et de la densité surfacique (en rouge).

On voit très bien ici comment le réajustement dynamique de la galaxie a conduit à la formation d'une forte concentration centrale, la densité surfacique à $r = 0$ ayant doublé par rapport à celle caractérisant la condition initiale. Les particules-étoiles ayant migré vers les régions centrales ont également augmenté leur vitesse orbitale, et les celles sur des orbites de forte excentricité atteignent leur vitesse azimutale maximale à au périhélium de leur orbite autour du centre galactique. La combinaison de ces deux effets conduit à une hausse significative de la courbe de rotation dans les premiers $\simeq 3$ kpc. Ce genre de concentration centrale est observée dans les vraies galaxies spirale (c'est le “bulbe”). De même, les particules-étoiles ayant reçu un excès de quantité de mouvement orbital de la composante thermique ont migré vers les régions extérieures tout en conservant leur moment cinétique, conduisant à une légère hausse de la courbe de rotation au-delà de $\simeq 12$ kpc. On est cependant encore très loin d'une

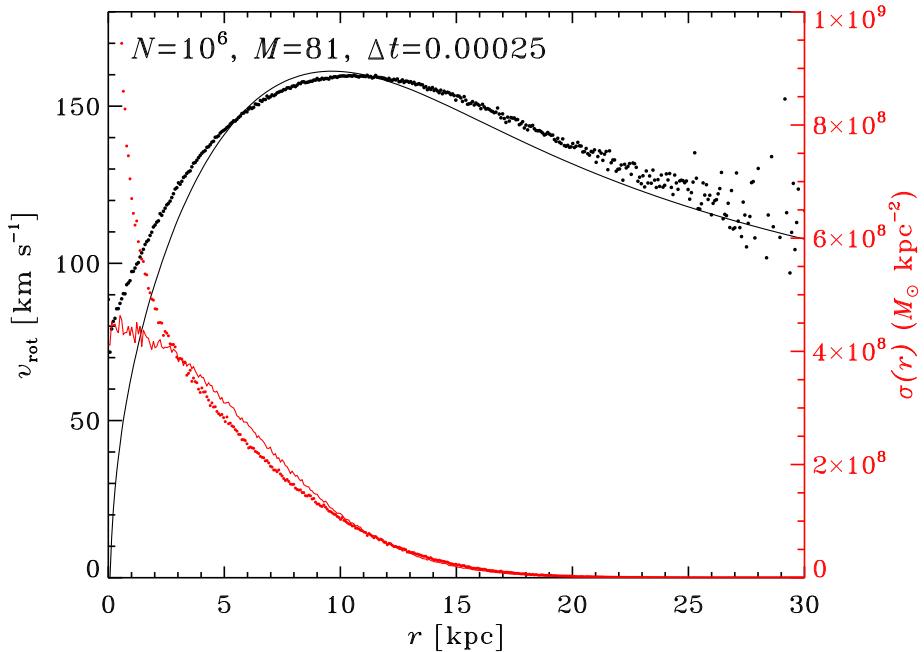


Figure 4.21: Profils initiaux (traits continus) et finaux (points) de la vitesse moyenne azimutale (noir) et densité surfacique (rouge) pour une simulation de $N = 10^6$ particules. La dispersion de v_{rot} débutant vers $r \gtrsim 20$ kpc est une conséquence du faible nombre de particules présents dans les anneaux correspondant, et de la condition limite “réfléctrice” imposée aux frontières du domaine (cf. éq. (4.32)).

courbe de rotation plate, car malgré cette hausse la décroissance en $r^{-1/2}$ est maintenue.

4.6 Projet: la matière sombre

Le but du projet est d’ajouter à la simulation une contribution au potentiel gravitationnel provenant d’un halo de matière sombre s’étendant en périphérie de la galaxie, suffisamment dense pour “redresser” la courbe de rotation. Ce halo de matière sombre est diffus, et on supposera qu’il n’est pas influencé par la position des étoiles dans la galaxie. Autrement dit, vous n’avez pas à recalculer à chaque pas de temps sa contribution au potentiel. Cependant il sera pratique de comptabiliser sa contribution en terme d’une densité surfacique (également fixe et spécifiée *a priori*), de manière à pouvoir conserver le protocole de calcul du potentiel gravitationnel introduite ci-dessus.

Plusieurs formes analytiques peuvent être définies pour décrire la variation avec le rayon de la densité surfacique (σ_H) du halo de matière sombre; par exemple:

$$\sigma_H(r) = \frac{\sigma_0}{1 + (r/r_H)^\alpha} \quad [M_\odot \text{kpc}^{-2}] . \quad (4.48)$$

Ce profil est défini par trois paramètres, soit la densité (surfacique) centrale σ_0 , l’étendue radiale r_H du halo, et un exposant α (> 0) contrôlant le taux de décroissance de la densité à grandes distances (dans le sens $r \gg r_H$). Les étapes de la modélisation sont les suivantes:

1. Modifiez votre condition initiale de manière à être en équilibre Képlérien, et répétez une simulation incluant une composante aléatoire de vitesse, comme à la §4.5.5;

2. Trouvez une (ou plusieurs) combinaisons de paramètres $[\alpha, r_H, \sigma_H]$ qui conduisent à une courbe de rotation demeurant plate jusqu'aux étoiles les plus éloignées du centre.
3. Comment la masse totale de votre halo se compare-t-elle à la masse de la composante stellaire du disque ?

Voici quelques autres idées qu'il pourrait être intéressant d'explorer (avec ou sans halo de matière sombre dans la simulation):

1. Une mesure très physique qui permet ici de vérifier la précision numérique de votre simulation est encore la conservation de l'énergie (potentielle + cinétique). Codez ça et vérifiez le niveau de conservation de l'énergie quand vous changez le pas de temps, et/ou la cadence de recalculation du potentiel gravitationnel.
2. Répétez une simulation, en version avec et sans halo, débutant cette fois d'une distribution ellipsoïdale de particules (mais toujours en équilibre Képlérien); explorez les similarités et différences par rapport à une simulation équivalente débutant d'une condition initiale axisymétrique (circulaire) comme auparavant.
3. Dans la même veine que le précédent, définissez une condition initiale où les étoiles sont distribuées avec une densité surfacique uniforme dans un disque de rayon r_D , et assignez aux étoiles une vitesse angulaire égale à 90% de sa valeur Képlérienne, et incluant une composante "thermique", comme à la §4.5.5. La galaxie évolue-t-elle vers une configuration finale semblable à celle utilisant un profil initial Gaussien pour la densité surfacique ?
4. Si vous voulez vraiment vous amuser: simulez la collision de deux (petites) galaxies. L'idée est d'introduire dans la simulation une seconde galaxie, ses étoiles orbitant son centre, et ajouter à cette vitesse orbitale une (faible) vitesse linéaire conduisant à un déplacement de cette seconde galaxie vers la première. Variez la vitesse de la seconde galaxie et le paramètre d'impact de la collision, soit la distance d'approche minimale entre les deux centres galactiques. Vous devrez recalculer le potentiel à chaque pas de temps, ou pas loin, durant la collision même! Si vous manquez d'inspiration, voir l'article de Toomre & Toomre cité en bibliographie ci-dessous.

4.7 Bibliographie

Ce chapitre est pas mal de mon cru, mais tout l'aspect adimensionalisation et Argon liquide est emprunté assez directement au chapitre 8 de l'excellent ouvrage suivant, déjà cité à quelques reprises au fil des chapitres précédents:

Gould, H., & Tobochnik, J., *An Introduction to Computer Simulation Methods*, seconde éd., Addison-Wesley (1996).

Son chapitre 3 présente également une dérivation formelle de l'algorithme de Verlet. Sur ce sujet voir également le chapitre 8 dans:

Pang, T., *An Introduction to Computational Physics*, seconde éd., Cambridge University Press (2006).

Sur les processus de liquéfaction/solidification dans les simulations (3D) de dynamique moléculaire de l'Argon liquide, voir:

Matsukoa, H., Hirokawa, T., Matsui, M., & Doyama, M., *Phys. Rev. Lett.* **69**, 297-300 (1992).

Vous serez probablement surpris(e) de l'apprendre, mais la première étude de "cristallographie théorique" basé sur l'assemblage compact de sphères rigides remonte à nul autre que l'astronome Johannes Kepler, qui avait tenté d'y retrouver l'origine de la symétrie des flocons de neige, rien de moins ! Si ça vous intrigue, voir

Kepler, J., *The six-cornered snowflake*, trad. et réimpress. Oxford University Press (1966). Les données pour la courbe de rotation de NGC6946 reproduite sur la Fig. 4.14 sont tirées de:

Carignan, C., Charbonneau, P., Boulanger, F., & Viallefond, F., *Astron. Astrophys.*, **234**, 43–52 (1990),

mais pour une vision plus récente et plus large du contexte observationnel, voir plutôt:

de Blok, W.J.G., Walter, F., Brinks, E., Trachternach, C., Oh, S.-H., & Kennicutt, R.C. Jr, *Astron. J.*, **136**, 2648–2719 (2008);

cet article fait partie dans un numéro spécial de l'*Astronomical Journal* qui contient plusieurs autres articles d'intérêt portant sur les modèles de masse galactique, incluant les contributions de la matière sombre. Pour ce qui est de l'angle physique des particules, mon collègue Viktor Zacek a écrit il y a quelques années un excellent petit article de revue sur la matière sombre et sa détection, à un niveau très accessible. Publié dans un acte de congrès difficile à trouver, l'article est cependant disponible sur ArXiv:

<https://arxiv.org/pdf/0707.0472.pdf>

Plus récent et un peu plus technique, et aussi recommandé par Viktor et disponible sur ArXiv, l'article de revue suivant par K. Garrett et G. Düda:

<https://arxiv.org/pdf/1006.2483.pdf>

Le protocole de simulation de la §4.5 est inspiré du vieux classique

Hohl, F., *Astrophys. J.*, **168**, 343–359 (1971);

Si Frank Hohl pouvait déjà simuler une galaxie de 10^5 particules-étoiles en 1971 avec les ordinateurs du temps, vous devriez bien pouvoir en faire 10^6 maintenant, donc pas de chialage ! Datant de la même époque mais traitant de collisions/interactions entre galaxies, cet autre vieux classique mérite encore d'être lu:

Toomre, A., & Toomre, J., *Astrophys. J.*, **178**, 623–666 (1972).

Pour en savoir plus sur les méthodes de simulations à N -corps spécifiquement dans le contexte de problèmes de type gravitationnel, voir:

Trenti, M., & Hut, P., *Scholarpedia*, **3**(5), 3930 (2008),

qui est disponible en ligne:

[www.scholarpedia.org/article/N-body_simulations_\(gravitational\)](http://www.scholarpedia.org/article/N-body_simulations_(gravitational))

Chapitre 5

Les algorithmes évolutifs

Une des tâches les plus courante en sciences est d'ajuster les paramètres d'un modèle de manière à offrir la meilleure représentation possible d'un ensemble de données expérimentales. Cet ajustement se fait typiquement de manière à minimiser les différences entre les données et les "prédictions" du modèle pour un ensemble donné de valeurs des paramètres définissant le comportement du modèle. Ceci définit la classe de problème dits d'*optimisation*. Vous avez fait ça "à bras" dans le projet du chapitre 4, en variant les paramètres du halo de matière sombre pour reproduire le mieux possible la forme de la courbe de rotation galactique. Dans ce chapitre, on verra comment faire mieux.

5.1 Optimisation locale versus globale

Il convient de premièrement distinguer entre l'optimisation locale et globale. Cette importante distinction ressort aisément d'un problème d'optimisation conceptuellement simple, soit la recherche des maxima en intensité dans un patron de diffraction.

L'intensité lumineuse $I(x, y)$ du patron de diffraction produit par un front d'ondes planes de longueur d'onde λ sur une ouverture carrée de coté d est donnée par l'expression:

$$\frac{I(x, y)}{I_0} = \left(\frac{\sin x}{x} \frac{\sin y}{y} \right)^2, \quad (5.1)$$

avec I_0 l'intensité centrale, et les x et y étant des quantités normalisées mesurant la position sur l'écran:

$$x = \frac{\pi d}{\lambda} \sin(\theta_x), \quad y = \frac{\pi d}{\lambda} \sin(\theta_y). \quad (5.2)$$

Les θ_x et θ_y sont les angles sous-tendus par la direction de l'onde incidente et le segment de droite reliant le centre de l'ouverture à la position sur l'écran où est mesurée l'intensité lumineuse, projetés sur les directions orthogonales dans le plan de l'écran. La Figure 5.1 illustre la surface correspondant à cette fonction de deux variables. L'objectif de l'optimisation est de localiser numériquement le point le plus élevé dans ce paysage, défini ici dans un espace des paramètres bidimensionnels. Ceci définit donc un problème d'optimisation tout à fait classique.

5.1.1 Méthodes de grimpe

Fondamentalement, la vaste majorité des méthodes classiques d'optimisation sont basées sur le concept de la **grimpe**: on commence par choisir un point de départ dans l'espace des paramètres, ici une paire (x, y) (possiblement choisie aléatoirement) dans le paysage bidimensionnel; on explore le voisinage local de ce point (x, y) et on se déplace dans une direction ascendante dans la valeur de la fonction $f(x, y)$. Le processus est répété itérativement jusqu'à

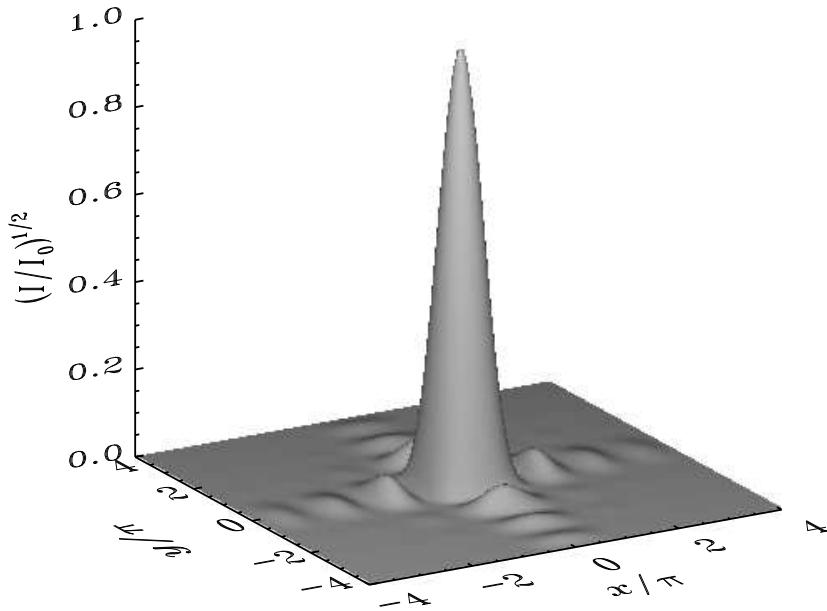


Figure 5.1: Le patron d'intensité $I(x, y)$ produit par la diffraction d'une onde plane à travers une ouverture carrée, représenté sous la forme d'une surface dans l'espace des paramètres $[x, y]$. Le maximum est à $I(0, 0)/I_0 = 1$.

ce qu'on se retrouve à un point (x^*, y^*) où toutes les directions partant de ce point soient descendantes. Le maximum est alors localisé. Le problème avec ce genre de méthodes de grimpe est qu'elles sont (habituellement) *locales*, c'est à dire qu'elles n'utilisent que l'information locale (le gradient de la fonction à maximiser au point courant (x, y)) pour effectuer le prochain pas. Dans le cas du profil de diffraction 2D de la Figure 5.1, la grimpe convergera sur le maximum d'intensité le plus rapproché de la position de départ, et donc ratera souvent le maximum global à $(x, y) = (0, 0)$.

On peut contourner ce problème à l'aide de la *grimpe itérée*, qui consiste simplement à faire rouler à répétition une méthode de grimpe, chaque fois avec un point de départ différent (habituellement choisi aléatoirement); on accumule une liste des maxima ainsi détectés, et ce jusqu'à ce qu'on ne trouve plus de nouveau maximum. Décider quand arrêter est évidemment l'aspect crucial de cette approche. La Figure 5.2 illustre l'idée, toujours dans le cas de notre patron de diffraction 2D. Les trajectoires conduisant aux maxima sont tracées pour une série de 50 points de départ (x^0, y^0) (indiqués par des \bullet colorés) choisis aléatoirement, et les contours en noir correspondent à des courbes de niveau $I(x, y) = \text{constante}$. Dans tous les cas la méthode de grimpe fait son travail et livre la marchandise, dans le sens que les grimpes se terminent toutes sur un maximum, mais dans bien des cas il ne s'agit pas du maximum le plus élevé à $(0, 0)$, mais plutôt de l'un ou l'autre des 8 autres maxima secondaires présents ici dans l'espace des paramètres. La probabilité de localiser ainsi le maximum global evient proportionnelle à la fraction de la surface de l'espace des paramètres occupée par la “bassin d'attraction” du maximum global, ici 1/9 dans le cas de la Fig. 5.2.

Bref, le maximum sur lequel converge la grimpe devient fonction de sa position de départ, une situation embarrassante puisque dans bien des cas nous n'avons pas idée de la position du maximum global, et donc ne sommes pas en mesure de fournir à l'algorithme un point de

départ qui garantisse de mener à l'extremum global.

La Figure 5.2 illustre à merveille la distinction entre l'optimisation **locale** et l'optimisation **globale**. Cette dernière, qui consiste à trouver le maximum absolu dans tout l'espace des paramètres, est beaucoup plus coriace numériquement. Parfois, comme dans le cas de notre problème de diffraction, l'existence d'extrama secondaires est due à la nonlinéarité du modèle; dans d'autres cas c'est le bruit dans les données expérimentales qui conduit à un espace de paramètre multimodal. À l'heure actuelle il n'existe aucun algorithme qui *garantisse* la détection d'un maximum global !.

5.1.2 La grimpe stochastique

La *grimpe stochastique* est une variation de type Monte Carlo sur le thème de la grimpe itérée. L'idée générale est la même que pour la méthode de grimpe itérée de la §5.1.1, avec l'importante différence que les direction et grandeur du pas, plutôt que d'être choisies d'une manière déterministe sur la base du gradient de la fonction, le sont de manière aléatoire; mais quand même pas n'importe comment. On cherche à produire des pas qui puissent avoir une taille comparable à celle de l'espace des paramètres, de manière à pouvoir "explorer" et se décoincer d'un extremum local; d'un autre coté, une bonne localisation d'un extremum demande un pas d'une taille de l'ordre du niveau de précision requis. En résumé, on veut générer une distribution de pas couvrant à la fois les grandes et les petites tailles. Une distribution ayant cette propriété est la distribution dite gaussienne.

Évidemment, si les pas de la grimpe sont choisis aléatoirement, plusieurs conduiront à une diminution de la valeur de la fonction f à maximiser (dans le contexte d'une maximisation), donc seul les pas conduisant à une augmentation de f sont "acceptés"¹.

La grimpe stochastique performe vraiment bien pour plusieurs problèmes d'optimisation globale de difficulté modérée; au point où je me permets de vous en donner à la Figure 5.3 un exemple d'implémentation en Python utilisant un pas adaptif. Examinez bien la série d'instructions à l'intérieur de la boucle extérieure (débutant à la ligne 21); La boucle **while** interne (débutant à la ligne 24) peut générer jusqu'à 20 pas aléatoires, et si (et seulement si!) aucun de ces pas ne conduit à une diminution de la fonction, alors la variance de la distribution gaussienne de taille de pas est diminuée de moitié (ligne 37), et on passe à une nouvelle itération sans changer l'origine de ce pas. Cependant si un des 20 pas a réduit la valeur de la fonction alors on accepte le pas (lignes 34–35) et on passe à l'itération suivante sans toucher à σ .

La Figure 5.4 montre le résultat de 10 grimpes stochastiques, chacune effectuée selon le code de la Figure 5.3, appliquée à notre désormais familier problème de recherche du maximum central du patron de diffraction d'une ouverture carrée. Pour des points de départ choisis aléatoirement, on s'attendrait ici à ce que la grimpe itérée produise un taux de succès de 1/9. Avec la grimpe stochastique, on obtient ici un très honorable 8/10. Remarquez comment certaines solutions se "coincent" pendant quelques itérations sur un extremum secondaire, mais finissent par atterrir sur le pic central suite à l'action d'un pas de taille substantiel; on voit ici l'avantage d'avoir utilisé une distribution gaussienne de taille de pas; bien que leur probabilité soit faible, de temps en temps un pas de taille substantiellement plus grand que la variance de la distribution sera produit, ce qui permet de mieux échantillonner l'espace environnant, tout en ne freinant pas trop la convergence vers l'extremum (qu'il soit local ou global).

Que ce soit en version itérée classique ou stochastique, chaque "essai" d'une grimpe itérée opère indépendamment des autres; autrement dit, l'information sur la fonction à optimiser accumulée le long d'une trajectoire n'est pas prise en considération lors du calcul des essais suivants. Les *algorithmes évolutifs* offrent une manière de conserver et utiliser cette information dans la recherche du maximum global, tout en améliorant les propriétés exploratoires de la grimpe stochastique. Leur inspiration provient en fait de l'évolution en biologie !

¹Les méthodes dites de *recuit simulé* ("Simulated annealing" dans la littérature anglophone) acceptent un pas qui diminue la fonction à maximiser selon une probabilité variant en $\exp(-\Delta f/T)$; vous reconnaîtrez ici l'algorithme de Metropolis introduit dans notre étude du modèle d'Ising. Voir les références en fin de chapitre si vous voulez en apprendre plus sur ces méthodes d'optimisation globale d'inspiration thermodynamique.

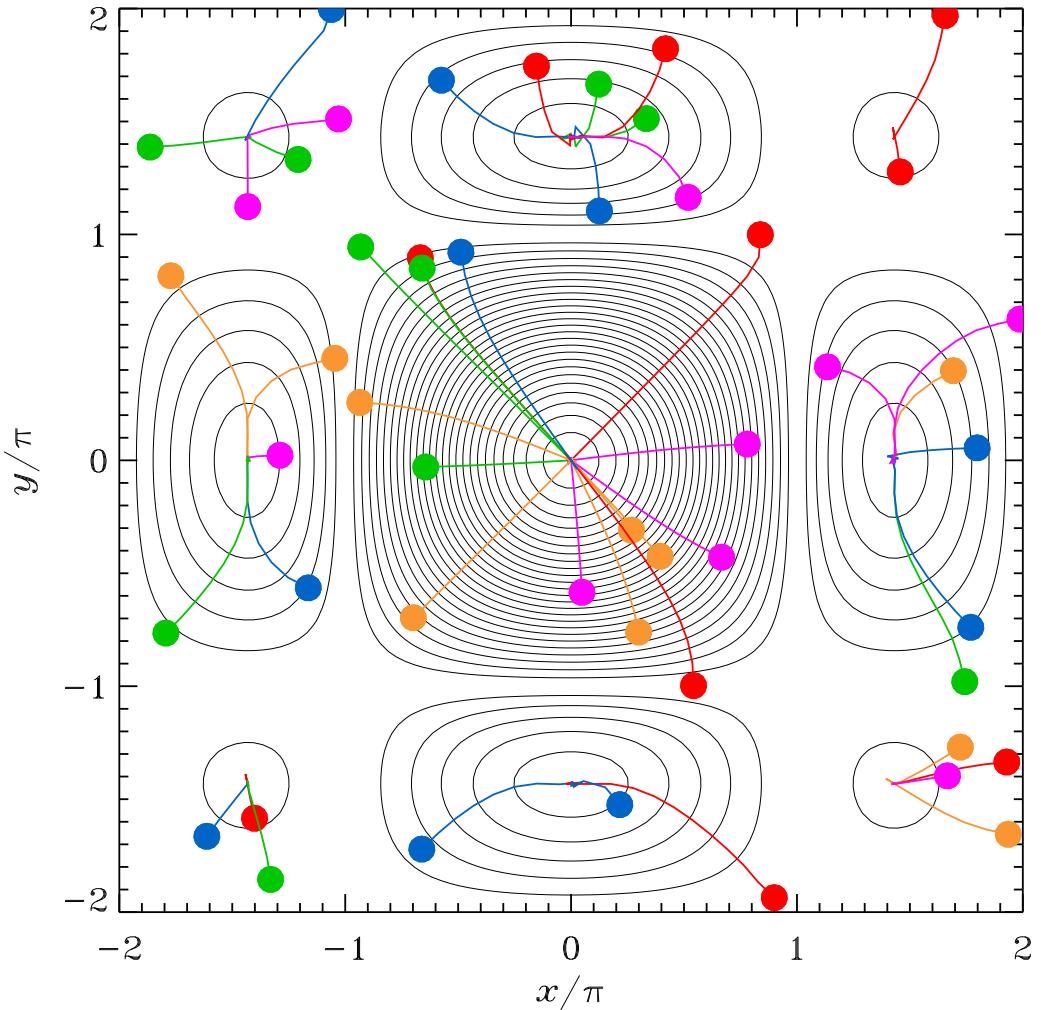


Figure 5.2: Application de la grimpe itérée à la recherche du pic central du patron de diffraction d'une ouverture carrée. Les traits noirs sont des courbes de niveau de la fonction à maximiser, soit l'éq. (5.1); son maximum est à $(x^*, y^*) = (0, 0)$, où $I(x, y)/I_0 = 1$. Les points colorés correspondent à 50 positions de départ (x^0, y^0) choisies aléatoirement, et les trajectoires en émanant résultent de l'application répétée d'une méthode de grimpe simple du genre que certains d'entre vous ont rencontré en PHY-1234. 35 des points de départ sur 50 convergent vers un maximum secondaire.

```

1 # MAXIMUM GLOBAL DU PATRON DE DIFFRACTION 2D PAR GRIMPE STOCHASTIQUE
2 #=====
3 import numpy as np
4 IterMx = 200          # nombre maximal de pas
5 NEssai = 20            # nombre d'essais par pas
6 epsilon=1.e-6          # precision requise sur la valeur du maximum
7 PI=3.1415926536       # Pi !
8 -----
9 # FONCTION A MAXIMISER: INTENSITE LUMINEUSE
10 def diffrac2d(x,y):
11     return (np.sin(x)/x)**2 *(np.sin(y)/y)**2
12 -----
13 # PROGRAMME PRINCIPAL (POUR UNE SEULE GRIMPE!)
14 x,y=3.1*PI,2.3*PI           # essai initial
15 pas=1.0                     # Taille initiale du pas
16 f  =diffrac2d(x,y)          # valeur au (x,y) initial
17 fn =f/2.                     # initialisations bidons...
18 delta=2.*epsilon            # ...pour le test de sortie
19 iter=0                      # compteur de pas
20
21 while (delta > epsilon) and (iter < IterMx): # Boucle de grimpe
22     j=0                         # compteur d'essais
23
24     while (fn <= f) and (j < NEssai): # Boucle d'essais de pas
25         g1 = np.random.normal(0.,1.)   # Aleatoires gaussien sigma=1
26         g2 = np.random.normal(0.,1.)   # Aleatoires gaussien sigma=1
27         xn = x + pas*g1             # pas en x
28         yn = y + pas*g2             # pas en y
29         fn = diffrac2d(xn,yn)        # valeur au nouveau (xn,yn)
30         j+=1                        # incremente compteur d'essais
31 # FIN boucle d'essais de pas
32
33     if fn > f:                  # On accepte le pas
34         delta = abs( fn-f )      # Pour le test de sortie
35         x,y,f=xn,yn,fn          # Deplacement du grimpeur
36     else:
37         pas/=2.                  # Diminution du pas
38
39     iter+=1                      # incremente compteur de pas
40     print("Iter={0}, Maximum: {1}, x={2}, y={3}.".format(iter,f,x,y))
41 # FIN boucle de grimpe
42 # END
43

```

Figure 5.3: Code Python pour la grimpe stochastique. Notez que chaque pas implique sa propre boucle conditionnelle `while`, qui tente ici 20 pas stochastiques avant de refuser le pas et en réduire la grandeur, si aucun de ces 20 essais n'a conduit à une augmentation de la fonction-mérite.

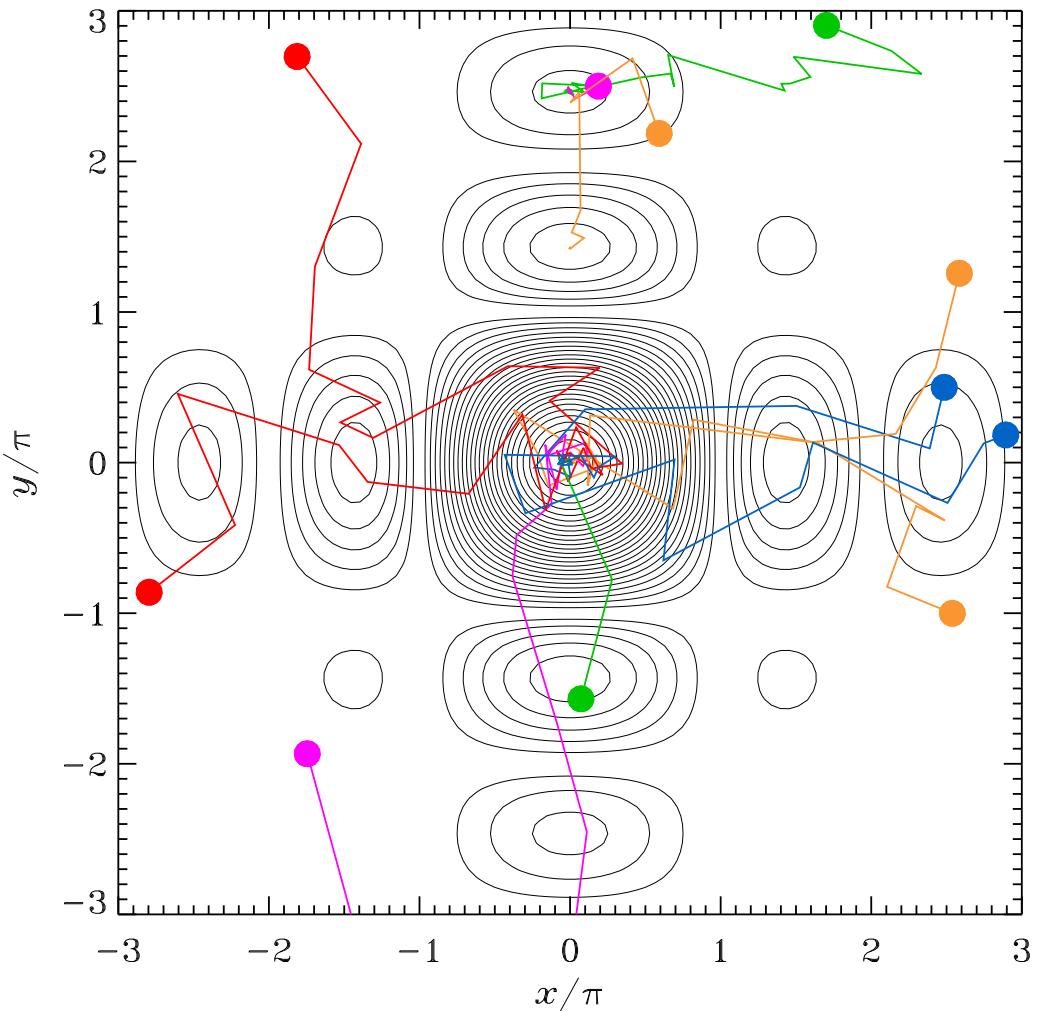


Figure 5.4: Recherche du maximum central du patron de diffraction 2D par grimpe stochastique (ici itérée). Remarquez comment certaines grimpes se coincent temporairement sur un extrémum secondaire, mais parviennent à s'en dépêtrer suite à la production d'un pas de longueur substantielle. Le taux de succès est ici de 8/10, ce qui est beaucoup mieux que le 1/9 attendu de la grimpe itérée (voir Figure 5.2).

5.2 La puissance de la sélection cumulative

L'idée qu'un processus de sélection puisse accélérer une recherche du genre grimpe stochastique semble évidente, mais ce qui l'est pas mal moins et l'impact d'un tel processus de sélection sur une succession de générations de solutions-test. Un exemple très simple illustre ceci à merveille. Prenons la petite phrase suivante, tirée d'un ouvrage bien connu d'un auteur qui vaut la peine d'être lu:

D E S S I N E M O I U N M O U T O N

Cette phrase compte 21 caractères, tirés d'un alphabet de 27 lettres (on compte les espaces blancs comme un caractère et un membre de l'alphabet). Installons maintenant le proverbial stagiaire d'été face à un clavier d'ordi, avec une bonne réserve de bananes, et la mission de reproduire cette phrase cible. L'idée est de laisser notre stagiaire taper aléatoirement des séquences de 21 caractères, jusqu'à temps qu'il/elle tombe sur la bonne. Voici un exemple de 10 de ces essais, avec le nombre de lettres correctes indiqué dans la colonne de droite:

H C W O X O I Q B E D D D P L I E G U V E	0
I U X Y U M O A D U C O T F V W A G E	0
L F C C P I K A Q Q A H Z J R S V R R T	0
C X Q R Z F Q J Z O B C B N K Y L D S R U	2
S C N C A I G F S F G K W U K T K O U	1
X D D W G O O X X R P E I F K D B W K F Q	0
Z O Q N A A V M O L H F P P B R E B A	0
Y P F V W Y I U X Q J H P C C A R Y N D	0
A O A R G G E V B E N G J B H Q Z K G	0
Y J Y E O Q F E L Y A N L R J A Y V V Q F	0

Ca ne ressemble absolument pas à la phrase cible, quoiqu'en y regardant bien on constate que le quatrième essai a en fait reproduit deux des lettres au bon endroit dans la séquence, soit le "O" du "MOI" et le "N" du "UN". Considérant la longueur de la séquence et la taille de l'alphabet , produire une phrase avec deux lettres correctes en 10 essais est tout à fait probable. Il sera utile, en anticipation de ce qui suit, de se rafraîchir la mémoire sur ce genre de calcul de probabilités. Supposons que l'on veuille calculer la probabilité d'obtenir au moins une lettre correcte, à n'importe quelle position dans une phrase-essai spécifique. La probabilité se calcule comme suit:

- $1/27$: probabilité qu'une lettre soit correcte; donc
- $1 - 1/27$: probabilité qu'une lettre soit incorrecte; donc
- $(1 - 1/27)^{21}$: probabilité que toutes les 21 lettres soient incorrectes; donc
- $1 - (1 - 1/27)^{21} = 0.547$: probabilité qu'au moins une lettre soit correcte.

Donc, on en fait été un peu malchanceux dans nos dix essais ci-dessus, car nous n'avons que trois phrases sur 10 qui ont au moins une lettre correcte.

Bon, ceci dit, la probabilité p de reproduire correctement les 21 lettres est donnée par

$$p = \left(\frac{1}{27}\right)^{21} = 8.737 \times 10^{-31}, \quad (5.3)$$

soit une chance sur $\sim 10^{30}$. En physique, évidemment, on n'a pas peur des gros chiffres; 10^{30} , après tout, c'est seulement la moitié de la masse du soleil mesurée en kilogrammes... Mais il faut tout de même apprécier que 10^{30} , c'est vraiment gros. Si je vous donne 10^{30} grains de sable de qualité-plage (diamètre 0.2mm) pour en faire un tas, votre tas recouvrera la Terre entière d'une épaisseur de 8 kilomètres. Et votre tâche est de trouver, dans ce tas, un seul et

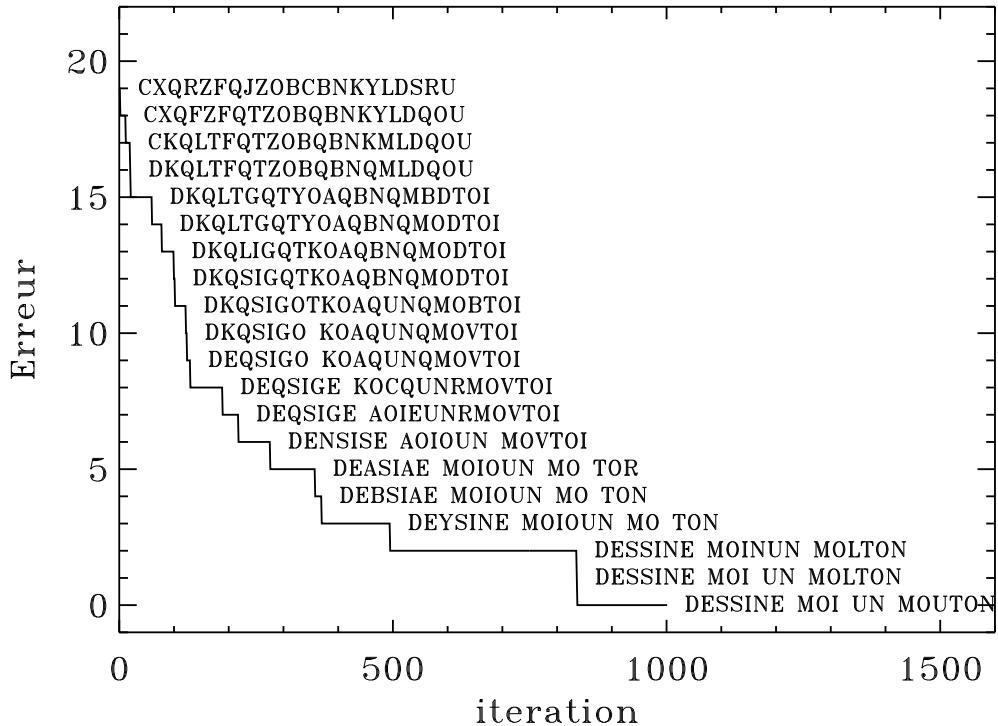


Figure 5.5: L'évolution du Petit Prince par sélection artificielle.

unique grain très spécifique. Je n'ai pas fait le calcul, mais je soupçonne fort que même avec une armée de 7 milliard de stagiaires d'être entraîné(e)s à examiner un grain de sable à la seconde, la quantité de bananes requises pour trouver ce fameux grain St-Exupériesque est beaucoup, beaucoup plus grande que la masse de l'Univers, matière sombre incluse. Bref, bonne chance...

Considérons maintenant la variation suivante; de nos 10 essais aléatoires, on choisit le meilleur (on devrait peut-être dire plutôt le moins pire, à ce stade), soit ici le quatrième; on en fait 10 copies conformes; puis, dans chaque copie, on introduit des "mutations", de la manière suivante: chaque lettre se fait remplacer par une autre lettre choisie aléatoirement dans l'alphabet, avec une probabilité p_m . Si cette probabilité $p_m \ll 1$, alors on peut imaginer que seule une ou deux lettres seront remplacées dans chaque phrase. Une fois ce processus complété pour les 10 phrases recopiés, on rechoisit la meilleure dans cette nouvelle "population", et on recommence le cycle de copie/mutation. Vous percevez j'espère l'analogie avec l'évolution en biologie; le choix du meilleur, c'est la sélection naturelle (en version assez extrême); la copie en 10 nouvelles phrases, c'est la reproduction (en version asexuée); les mutations, c'est la variabilité génétique produite par les erreurs de copies de l'ADN.

On peut facilement imaginer que ce processus accélérera la reproduction de notre phase cible, mais je doute fort que vous réalisiez jusqu'à quel point cette accélération est substantielle. La Figure 5.5 montre la variation de l'erreur (mesurée ici comme le nombre de lettres incorrectes) en fonction du nombre de "générations" de 10 phrases produites. La phrase cible est atteinte ici en 835 itérations, durant lesquelles 8350 phrases ont été "évaluées". On est très très loin du 10^{30} de la recherche aléatoire!

Une minute de réflexion devrait suffire pour comprendre que la mutation est ce qui nous fait progresser rapidement ici; donc il devrait être possible de faire mieux que 835 itérations en augmentant p_m , dont la valeur était de 10^{-2} pour la "solution" de la Figure 5.5. La Figure 5.6 montre trois convergences vers la phrase-cible, pour $p_m = 10^{-3}, 10^{-2}$ et 10^{-1} . À $p_m = 10^{-3}$, la convergence est plus lente qu'à 10^{-2} , comme on s'y attendait; mais à $p_m = 0.1$, la convergence,

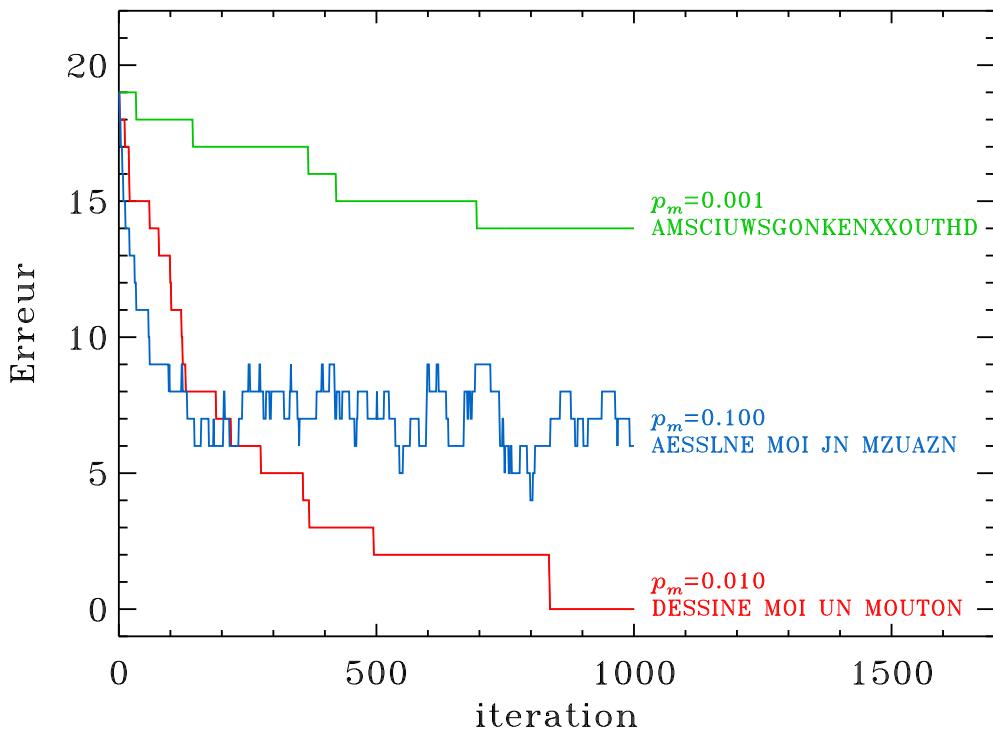


Figure 5.6: Impact de la probabilité de mutation sur l'évolution linguistique. Si la mutation est trop élevée, la convergence sature (voir texte).

bien que très rapide au début, sature rapidement à un niveau d'erreur substantiel. Le problème est ici que la mutation ne fait pas seulement remplacer les mauvaises lettres par des bonnes, elle fait aussi l'inverse, et la convergence sature lorsque la probabilité totale de détruire une “bonne” lettre devient égale à celle de “produire” une bonne lettre.

Imaginons qu'on en est rendu à n lettres correctes; quelle est la probabilité de passer à $n + 1$ lettres correctes, étant donné une probabilité de mutation p_m ? Allons-y par étapes: on commence par calculer la probabilité de produire par mutation une nouvelle lettre correcte dans une copie de la phrase à n lettres correctes, dans une phrase de N caractères et un alphabet de A lettres. Le calcul se fait comme suit:

- p_m/A : probabilité d'avoir une mutation (p_m) et (\times) que cette mutation produire la bonne lettre ($1/A$); donc:
- $1 - p_m/A$: probabilité de ne pas produire la bonne lettre; donc:
- $(1 - p_m/A)^{N-n}$ probabilité de ne produire aucune lettre correcte parmi les $N - n$ lettres incorrectes; donc
- $1 - (1 - p_m/A)^{N-n}$ probabilité de produire au moins une bonne lettre parmi les lettres incorrectes.

Même si ceci se produit, pour avoir progrès vers la phrase-cible il faut aussi qu'aucune des bonnes lettres existantes dans la phrase où une nouvelle correcte vient d'être produite ne soit détruite par une mutation. Calculons cette probabilité:

- $(A - 1)/A$: probabilité de produire une mauvaise lettre; donc:

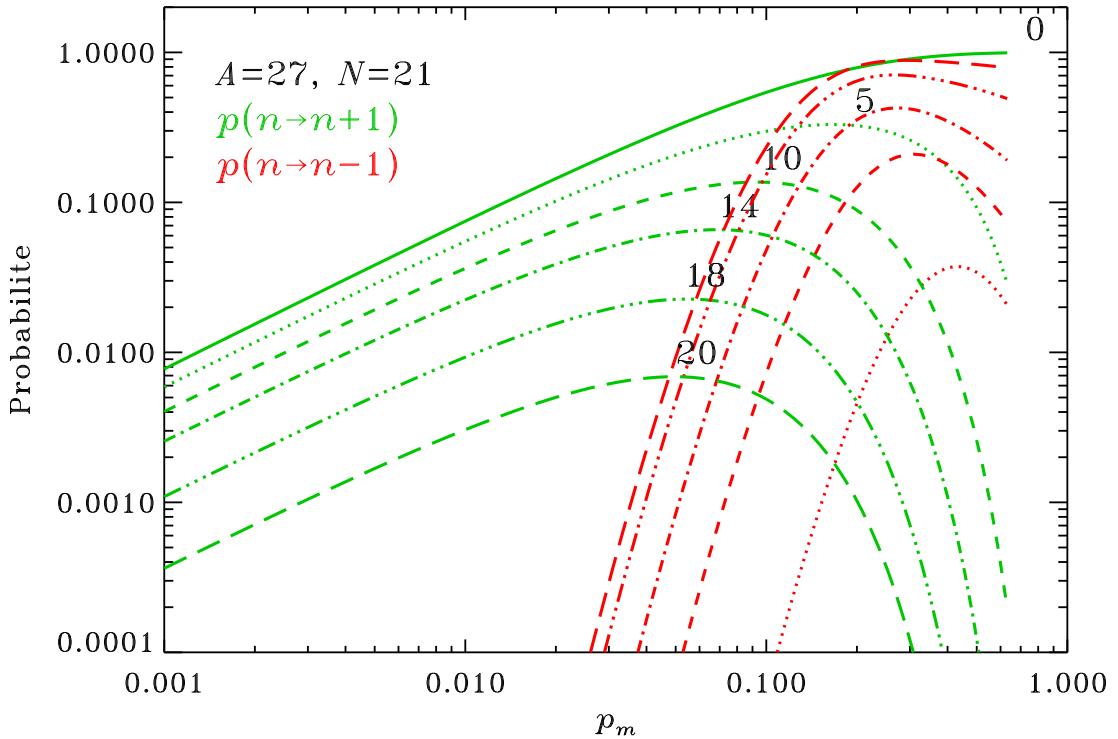


Figure 5.7: Les probabilités de progression $p(n \rightarrow n+1)$ en rouge, et de régression $p(n \rightarrow n-1)$ en vert, telles que données par les éqs. (5.5) et (5.6), en fonction de la probabilité de mutation p_m , pour quelques valeurs du nombre n de lettres correctes, indiqué au dessus de chaque courbe. Quel que soit la probabilité de mutation, la probabilité de progression (régression) décroît (croît) à mesure que n augmente (voir texte).

- $(A - 1)p_m/A$: probabilité d'avoir une mutation et que cette mutation produise une mauvaise lettre; donc:
- $1 - (A - 1)p_m/A$: probabilité de ne pas détruire une bonne lettre donnée; donc
- $(1 - (A - 1)p_m/A)^n$: probabilité de ne détruire aucune des n bonnes lettres.

La probabilité de progresser de n à $n + 1$ bonnes lettres dans une phrase est donc donnée par:

$$\left[1 - \left(1 - \frac{p_m}{A} \right)^{(N-n)} \right] \times \left[\left(1 - \frac{(A-1)p_m}{A} \right)^n \right] \quad (5.4)$$

et donc, la probabilité de progresser de n à $n + 1$ bonnes lettres dans au moins une des 10 phrases-essai sera donnée par:

$$p(n \rightarrow n+1) = 1 - \left(1 - \left[1 - \left(1 - \frac{p_m}{A} \right)^{(N-n)} \right] \times \left[\left(1 - \frac{(A-1)p_m}{A} \right)^n \right] \right)^{10} \quad (5.5)$$

Les courbes en vert sur la Figure 5.7 montrent comment varie cette probabilité en fonction de la probabilité de mutation p_m , pour des phrases ayant un nombre croissant de lettres correctes, soit le n dans l'éq. (5.5), indiqué au dessus du maximum de chaque courbe.

La probabilité de progression décroît quand n augmente, et ce à toutes probabilités de mutation p_m . Ceci explique la forme des courbes de convergence sur les Figs. 5.6: convergence

rapide au début, ralentissant par la suite. On notera cependant que cette décroissance de $p(n \rightarrow n+1)$ avec n est plus rapide à grand p_m , et que le maximum de $p(n \rightarrow n+1)$ se déplace graduellement vers les plus basses valeurs de p_m à mesure que n augmente; il n'y a donc pas une valeur de p_m qui soit optimale à tous les stades du processus évolutif ici.

Pour que la convergence vers la phrase-cible se produise, il faut aussi que cette probabilité de progression demeure supérieure à la probabilité de régression $p(n \rightarrow n-1)$, soit passer de n à $n-1$ lettres correctes, et ce jusqu'à ce que $n = N$. La probabilité de régression est donnée par le produit de la probabilité de détruire au moins une bonne lettre dans une phrase multipliée par la probabilité de ne pas produire une bonne lettre dans cette même phrase, le tout exposant 10 puisque la régression doit se faire dans les 10 phrases recopiées pour que l'itération dans son ensemble régresse. Je vous laisse calculer que ça donne:

$$p(n \rightarrow n-1) = \left(\left[\left(1 - \frac{p_m}{A} \right)^{(N-n)} \right] \times \left[1 - \left(1 - \frac{(A-1)p_m}{A} \right)^n \right] \right)^{10}. \quad (5.6)$$

Ces probabilités de régression sont tracées en rouge sur la Figure 5.7, le codage des courbes (pointillés, tirets, etc.) suivant celui des courbes de probabilités de progression. La probabilité de mutation où se produit l'intersection des courbes rouge et verte pour le même n donne le taux de mutation auquel le processus évolutif saturera au nombre correspondant de lettres correctes; par exemple, les courbes en tiret-points ($n = 14$ lettres correctes) se croisent à $p_m \simeq 0.1$; et sur la Fig. 5.6 on constate que la convergence à ce p_m sature bien à $21 - 14 = 7$ lettres incorrectes ! Banzaaaiii !!

On peut également conclure de la Figure 5.7 que dans une phrase de $N = 21$ caractères tirées d'un alphabet de $A = 27$ lettres, on pourra atteindre la phrase-cible avant de saturer tant que la probabilité de mutation $p_m \lesssim 0.04$. Si on est dans ce régime, on peut aussi utiliser les résultats de la Figure 5.7 pour estimer le nombre d'itérations requises pour atteindre la phrase cible. Une bonne limite inférieure peut être obtenue en calculant le nombre d'itérations requises pour passer de $n = N-1$ à N lettres correctes, soit l'inverse de la probabilité associée à la courbe $n = 20$ (en long tirets) à la valeur de p_m utilisée; pour l'exemple de Fig. 5.5, avec $p_m = 10^{-2}$, on a $p(n \rightarrow n+1) \simeq 3 \times 10^{-3}$, d'où le nombre d'itérations requises ~ 330 ; à $p_m = 10^{-3}$, cela monterait à ~ 3000 itérations. Un estimé plus précis du nombre d'itérations requises pour atteindre la phrase-cible peut être obtenu via l'éq. (5.5) en calculant:

$$\sum_{n=0}^{N-1} \frac{1}{p(n \rightarrow n+1)}. \quad (5.7)$$

On obtient 10005 et 1154 itérations pour $p_m = 10^{-3}$ et 10^{-2} , respectivement, ce qui se compare bien aux temps de convergence observés.

Tout ça pour dire: la mutation c'est bénéfique, mais seulement à petites doses; et le dosage peut être délicat et dépend habituellement du problème considéré. Si on passe à un alphabet hexadécimal ($A = 16$) et à des phrases de 128 caractères, une probabilité $p_m = 10^{-2}$ ne sera pas nécessairement appropriée.

Évidemment, comme représentation du processus d'évolution biologique cet exemple n'est pas très réaliste: (1) on ne choisit ici que la meilleure de toutes les solutions-test, ce qui représente une forme de sélection plutôt extrême; (2) la "reproduction" produit ici des copies quasi-conformes du parent; (3) la qualité des solutions est mesurée par rapport à un absolu connu *a priori*, ce qui n'est pas du tout le cas en biologie (n'en déplaise à Teilhard de Chardin et son fan club). Il n'en demeure pas moins qu'il est possible de bâtir des méthodes d'optimisation globale qui incorporent ces idées dans un contexte purement numérique. Ses éléments-cléf sont: (1) un processus de sélection des solutions, basé sur une mesure de leur "qualité"; (2) la reproduction des solutions qui sont jugées les meilleures, d'une manière qui préserve au moins en partie leurs caractéristiques avantageuses (**héritabilité**); et (3) l'injection de **variabilité** dans la population, habituellement via le processus de reproduction, afin de pouvoir explorer les coins de l'espace des paramètres qui ne sont pas "échantillonnés" par la population initiale. Par

exemple, dans l'exemple littéraire ci-dessus, aucune des phrases-tests n'avait de "D" en première position; aucun mécanisme de reproduction basé uniquement sur un échange de lettres entre solutions n'aurait pu, en soi, conduire à la phrase-cible.

5.3 Un algorithme évolutif de base

Voyons maintenant comment appliquer cette idée à notre désormais familier problème de recherche du maximum global du patron de diffraction d'une ouverture carrée.

Nous considérons dans ce qui suit un algorithme évolutif très simple. Les choix spécifiques faits ci-dessous au niveau de la triade des processus de sélection/héritabilité/variabilité ne sont pas uniques, et certainement pas optimaux, mais ils conduisent à un algorithme relativement facile à coder et dont la performance est tout à fait acceptable.

5.3.1 L'algorithme

Le calcul évolutif procède de manière itérative et agit sur une population de N solutions-test au problème. Dans sa forme de base discutée ici, il ne requiert que la capacité d'assigner à chaque solution-test une mesure de qualité permettant de distinguer les meilleures solutions-test des pires. On peut voir cette mesure de qualité (ici, l'intensité lumineuse telle que donnée par l'éq. (5.1) comme une fonction des paramètres du problème (ici les positions (x, y)), et vu sous cet angle le calcul évolutif n'est qu'une autre méthode d'optimisation de cette mesure de qualité, quelle qu'elle soit. Un algorithme de base a l'air de ceci:

1. **Initialisation:** on produit une population de N solutions (ici, des paires (x, y)) complètement aléatoires;
2. **Évaluation:** On évalue la qualité des membres de la population;
3. **Classement:** On classe les solutions en ordre croissant de qualité;
4. **Test:** Si la meilleure solution satisfait au critère de convergence , on arrête ici; sinon on continue;
5. **Sélection:** On choisit deux bonnes solutions, en fonction de leur rang;
6. **Reproduction:** Les deux solutions choisies se reproduisent en deux nouvelles solutions;
7. **Itération:** Les étapes 5 et 6 sont répétées jusqu'à ce que l'on ait produit un nouvel ensemble de N solutions,
8. **Remplacement:** On remplace l'ancienne population par la nouvelle.
9. **Prochaine génération:** Retour à l'étape 2.

Examinons maintenant le détail des différentes étapes de cet algorithme évolutif.

5.3.2 Initialisation

L'évolution débute habituellement avec une population aléatoire; par exemple, pour le problème de recherche du maximum de diffraction en 2D, chaque membre de la population est un point (x, y) , dont les valeurs devraient être extraites d'une distribution aléatoire uniforme dans l'intervalle considéré, par exemple $[-3\pi, 3\pi]$ comme sur la Figure 5.4. Cet échantillonnage initial est important, car il sera la source du "pool génétique" sur lequel travaillera le processus évolutif. On veut que ce pool soit le plus varié possible.

Dans le cas de notre problème de diffraction 2D, l'initialisation de la population se ferait en Python sous `numpy` comme suit:

```
L=3.*PI                      # etendue du domaine en [x,y]
for i in range(0,N):
    x[i]=numpy.random.uniform(-L,L)
    y[i]=numpy.random.uniform(-L,L)
```

où l'instruction `numpy.random.uniform(-L,L)` invoque le générateur de nombres aléatoires distribuée uniformément (ici dans l'intervalle $[-L, +L]$) de la librairie `numpy` de Python, et la variable `L` contrôle ici l'étendue de la région de l'espace des paramètres à explorer. Ce bout de code presuppose que les tableaux `x[N]` et `y[N]` et la variable `PI` ont été définis de manière appropriée.

On peut limiter le domaine où l'on recherche le maximum, mais pas trop tout de même

5.3.3 Évaluation

Il faut maintenant assigner une mesure de qualité (“fitness”) à chaque membre de la population. Dans le cas de la recherche de maximum du patron de diffraction 2D, ce pourrait être simplement la valeur absolue (ou le carré) de l'évaluation de l'éq. (5.1); Une fonction Python effectuant cette évaluation est déjà incluse dans le code illustrant la méthode de grimpe stochastique (Fig. 5.3). En général, c'est à vous de définir une mesure quantitative de la qualité qui soit appropriée au problème. Cette fonction est le seul point de contact entre l'algorithme évolutif et votre problème d'optimisation !

5.3.4 Classement

Dans le processus de sélection qui sera décrit plus bas, il s'avèrera pratique d'avoir accès aux N solutions formant la population, classées en fonction de leur mesure de qualité. Le classement est un sujet classique couvert dans tous les cours de programmation. Vous pourrez en trouver une discussion dans le *Numerical Recipes* déjà souvent cité. Un exemple illustrera mieux le concept qu'un long discours. Considérons le tableau 1D suivant, de longueur $N = 5$:

```
x=[3,8,7,2,5]
```

Son tableau de rang, également de longueur N , est le suivant:

```
rang(x)=[3,0,4,2,1]
```

L'élément `rang[0]` contient l'indice correspondant au plus petit élément du tableau `x`, soit ici `x[3]=2` (sous la convention de numérotation Python). De même, `rang[1]` contient l'indice du second plus petit, soit `x[0]=3`, et `rang[N-1]` contient l'indice du plus grand élément, `x[1]=8` pour notre exemple ci-dessus.

La plupart des langages de programmation contiennent des fonctions calculant le rang des éléments d'un tableau. En Python, la plus simple d'utilisation est la fonction `argsort` de la librairie `numpy`. Soit `f[N]` un tableau de longueur N contenant les valeurs numériques de la mesure de qualité; sous `numpy`, l'instruction suivante:

```
rang = numpy.argsort(f)
```

indices du tableau trié (sans trier le tableau original)

retourne un tableau `rang[N]` équivalent à celui introduit plus haut, et qui contient les positions des rangs 1,2,... des éléments de `f[N]`. ATTENTION: `rang[N-1]`, et non `rang[N]`, contient l'indice du plus grand élément d'un tableau de longueur N ; donc, on accède au plus grand élément de `f` via l'instruction `f[rang[N-1]]`. Pour notre problème de diffraction en 2D, cet élément est situé à une position $(x, y) = (x[rang[N-1]], y[rang[N-1]])$.

5.3.5 Sélection

Il existe plusieurs mécanismes de sélection possibles. Ce n'est en général *pas* une bonne idée de choisir seulement le meilleur membre de la population pour produire la génération suivante, car cela peut souvent conduire à une convergence prématurée sur un extrémum secondaire. Il

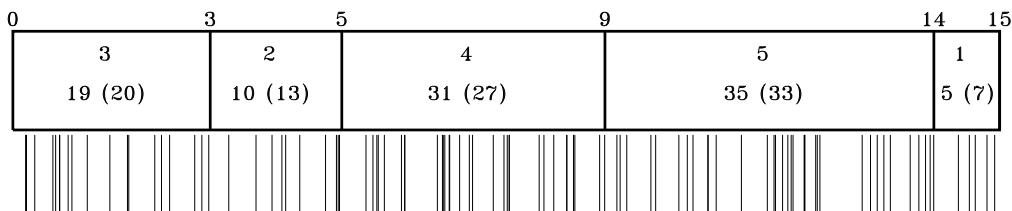


Figure 5.8: Représentation schématique de l'opération de l'algorithme de la roulette, ici pour une population de $N = 5$ individus, dont le tableau de rang est $[3, 2, 4, 5, 1]$. La barre horizontale est faite de cinq sections, chacune de largeur correspondant au rang de l'individu, pour une largeur totale $S = \sum_{n=1}^N n = 15$ ici, avec la somme cumulative des rangs indiquée au haut de la barre. Les tirets verticaux sont 100 nombres aléatoires tirés d'une distribution uniforme couvrant l'intervalle $[0, S]$. Le nombre de ces aléatoires tombant dans chaque section est indiqué, suivi, entre parenthèse, de la valeur attendue, ici $100 \times (\text{rang}/15)$.

est essentiel de préserver un bon niveau de variabilité dans la population, donc les solutions qui ne sont pas les meilleures ne doivent pas être éliminées trop rapidement du “pool génétique”.

Voici une première possibilité simple à coder et qui fonctionne bien: les “parents” sont choisis par paires (la reproduction étant définitivement plus agréable à deux que seul...), le premier aléatoirement parmi les $N/5$ meilleurs de la population, le second aléatoirement parmi la population en entier. Le facteur $1/5$ est ici un choix empirique, et détermine la **pression sélective**: une valeur numérique plus grande (e.g., $1/2$) donnerait moins d'influence aux meilleurs de la population, tandis que $1/N$ ferait que seul le meilleur serait choisi à tous les coups comme premier parent.

```
i1=rang[ numpy.random_integers(0.8*N,N)-1 ] # dans les premiers 20%
i2=i1
while (i2==i1):
    i2= np.random_integers(0,N-1) # n'importe qui sauf i1
```

Ici les entiers `i1` et `i2` identifient, dans les tableaux `x[N]` et `y[N]`, les position des paramètres x et y des deux parents choisis pour la reproduction. Notez bien que la fonction `numpy` retourne en entier aléatoire contenu dans un intervalle incluant les bornes données en argument, et que le “`-1`” reflète le fait que le rang N est contenu dans l’élément $N-1$ du tableau dont on a effectué le classement. La boucle `while` est requise pour empêcher que les deux parents correspondent au même individu; pour une population de petite taille, de tels croisements “consanguins” pourraient rapidement conduire à une convergence prématurée sur un extrémum suboptimal.

Une approche différente consiste à assigner à chaque individu une *probabilité* d’être choisi(e) comme géniteur(trice), en proportion à son rang. L'algorithme dit “de la Roulette” (**Roulette Wheel Algorithm**) effectue une telle sélection. De manière imagée, chaque individu se voit assigner une section d'une roue de casino, la largeur de la section étant proportionnelle au rang. On fait alors tourner la roue; plus un individu a une section de grande largeur angulaire, plus la probabilité que la roue s'y arrête est grande. La Figure 5.8 illustre l'idée schématiquement, la roue étant ici remplacée par une barre horizontale faite de N sections contigues, chacune ayant une largeur égale au rang de l'individu correspondant. La longueur totale de cette barre sera donc donnée par

$$S = \sum_{n=1}^N n = \frac{N \times (N + 1)}{2}, \quad (5.8)$$

égale à 15 pour $N = 5$ sur la Figure 5.8. On tire alors un aléatoire d'une distribution uniforme d'étendue $[0, S]$, et la section correspondante de la barre indique l'individu choisi. Ce processus

de sélection est stochastique, dans le sens que le nombre de fois qu'un individu k est choisi n'approche la fraction théorique $\text{rang}[k]/S$ que dans la limite où le nombre de tir d'aleatoire devient très grand (dans le sens $\gg N^2$).

Tout cela peut sembler compliqué à coder, mais en fait ça peut se faire en quelques lignes de code, avec un générateur de nombre aléatoire. Sous forme de fonction Python, ça pourrait avoir l'air de ceci:

```
# FONCTION DE SELECTION PAR ALGORITHME DE LA ROULETTE
def roulette(rang,N):
    r=N*(N+1)/2. *np.random.uniform() # aleatoire uniforme dans [0,S]
    scumul=0.
    for k in range(0,N):
        scumul+=(1.+rang[k]))           # some cumulative des rangs
        if scumul >= r: return k       # on a trouve
    # END fonction roulette
```

Remarquez comment l'identification est effectuée en calculant la somme cumulative des rangs, et quand cette somme $> r$ alors l'individu k correspondant est choisi (l'addition de "1" au rang dans la somme cumulative vient du fait que ce dernier, en Python, va de 0 à $N - 1!$). La procédure de sélection devient alors:

```
i1=roulette(rang,N)                      # un premier parent
i2=i1
while (i2==i1):
    i2= roulette(rang,N)                  # un second, n'importe qui sauf i1
```

Avec ce second algorithme de sélection, même le pire individu de la population conserve une (petite) chance d'être choisi comme parent; c'est parfois une bonne chose, car ceci aide à conserver un bon niveau de variabilité dans la population, ce qui en retour aide à prévenir une convergence prématuée sur un optimum secondaire.

5.3.6 Reproduction

La reproduction est un processus en deux étapes: le **croisement**, qui consiste à partager le "matériel génétique" des deux parents entre les deux "enfants"; et la **mutation**, qui consiste à introduire une perturbation au matériel génétique des enfants. Ces opérations ont lieu avec des probabilités p_c et p_m , respectivement. L'expérience montre que $0.5 \leq p_c \leq 1.0$ fonctionne bien (on choisira $p_c = 0.8$ dans ce qui suit), mais il est définitivement préférable d'avoir $p_m \ll 1$. Sous la librairie `numpy` en Python, on peut formuler ce genre de test probabiliste avec l'instruction:

```
if numpy.random.uniform() < pc:
    ...      # instructions effectuant le croisement
```

où, utilisée sans bornes explicites comme ci-dessus, la fonction `numpy.random.uniform()` retourne un nombre aléatoire extrait d'une distribution uniforme dans l'intervalle $[0, 1]$.

Toujours dans le cas de notre problème de diffraction 2D, le matériel génétique est simplement les valeurs de x et y définissant chaque membre de la population. Donc, soit les points (x_1^n, y_1^n) , (x_2^n, y_2^n) définissant les deux parents de la génération n choisis à l'étape "sélection", et deux nombres aléatoires r_1 , r_2 extraits d'une distribution uniforme dans l'intervalle $[0, 1]$. Le processus de croisement produit deux rejetons à partir des deux parents, selon les expressions suivantes

$$x_1^{n+1} = r_1 x_1^n + (1 - r_1) x_2^n, \quad x_2^{n+1} = (1 - r_1) x_1^n + r_1 x_2^n, \quad (5.9)$$

$$y_1^{n+1} = r_2 y_1^n + (1 - r_2) y_2^n, \quad y_2^{n+1} = (1 - r_2) y_1^n + r_2 y_2^n. \quad (5.10)$$

Notez que cette forme de moyenne pondérée aléatoirement entre les x et y des deux parents a comme effet que les rejetons se retrouvent automatiquement dans les intervalles:

$$\min(x_1^n, x_n^2) \leq x_1^{n+1}, x_2^{n+1} \leq \max(x_1^n, x_2^n), \quad (5.11)$$

$$\min(y_1^n, y_n^2) \leq y_1^{n+1}, y_2^{n+1} \leq \max(y_1^n, y_2^n). \quad (5.12)$$

Pour chacun des paramètres, le croisement n’opère qu’avec une probabilité p_c ; si la procédure de test refuse le croisement, alors les deux rejetons héritent chacun de l’une des valeurs de paramètre des parents; par exemple, si le croisement est refusé pour la variable x , on aurait alors:

$$x_1^{n+1} = x_1^n, \quad x_2^{n+1} = x_2^n, \quad (5.13)$$

La mutation consiste à ajouter une “perturbation” aux paramètres x ou y de chacun des rejetons, mais si seulement un test probabiliste du genre de celui introduit pour le croisement a été satisfait. On effectue un test par paramètre par rejeton). On aurait par exemple:

$$x_1^{n+1} = x_1^n + g(\sigma), \quad (5.14)$$

où $g(\sigma)$ est un nombre aléatoire extrait d’une distribution gaussienne de moyenne nulle et variance σ , comme on l’avait introduit dans la grimpe stochastique.

5.3.7 Remplacement

Une fois les rejetons produits par le processus décrit ci-dessus, on doit les insérer dans la population. La stratégie dite de **remplacement générationnel complet** consiste à accumuler les rejetons dans des tableaux temporaires (disons $xn[N]$, $yn[N]$ pour notre problème de diffraction 2D), et une fois qu’on en a produit N ils remplacent la population-parent (tableaux $x[N]$, $y[N]$) à la fin de l’itération générationnelle.

Une stratégie additionnelle très utile, l'**élitisme**, consiste à recopier intact le meilleur individu de la population-parent (l’individu de rang $rang[N-1]$) dans la population-rejeton, afin de s’assurer que les aléas du processus reproductif n’en viennent pas à faire reculer le processus évolutif. Si les parents sont “contenus” dans les tableaux x et y et les rejetons dans les tableaux xn , yn , un processus de remplacement générationnel complet avec élitisme pourrait avoir l’air de ceci:

```
x[:,],y[:,]=xn[:,],yn[:] # remplacement generationnel
x[rang[0]],y[rang[0]]=xtop,ytop # meilleur parent remplace pire rejeton
xtop,ytop=x[rang[N-1]],y[rang[N-1]] # sauvegarde du meilleur futur parent...
fstop=f[rang[N-1]] # et de sa fitness
```

Ici le tableau $rang$ contient les rangs de la population de rejetons. Notez bien, les paramètres du meilleur individu sont sauvegardés dans les variables $xtop$, $ytop$, de manière à ce qu’à l’itération générationnelle suivante, le pire rejeton puisse être remplacé par le meilleur parent.

5.3.8 Test de convergence

Comme dans toute méthode d’optimisation globale, il n’y a pas de choix unique et universel pour le critère d’arrêt. Typiquement, on itère soit pour un nombre de générations fixé *a priori* (boucle inconditionnelle), ou tant que la meilleure solution ne change plus de manière significative pendant un nombre prédéterminé d’itération (genre 100. Ou mille; ou plus...?).

5.4 La diffraction 2D, troisième tour...

Appliquons maintenant le calcul évolutif au problème de la recherche du maximum central du patron de diffraction 2D. La Figure 5.9 illustre la distribution des membres de la population en fonction du temps, résultant de l’application de l’algorithme évolutif décrit ci-dessus. Remarquez que la population initiale n’inclut aucun membre ayant “atterri” sur les flancs du pic central, ce qui implique que même avec ces 20 essais, la grimpe classique aurait échoué ici. Cependant, dès la seconde génération un croisement a produit un individu qui s’est retrouvé au flanc du pic central, et l’effet combiné de la sélection, des croisements et des mutations durant les 3–4 générations subséquentes fait converger l’ensemble de la population sur le pic central, quelque peu hors-centre (génération 6). Comme les x et y des différents individus sont maintenant presqu’identiques, le croisement n’accomplit plus grand chose, et c’est l’effet cumulatif des mutations, couplé à la sélection avec élitisme, qui déplace lentement la population vers l’extremum global à $(0, 0)$.

La Figure 5.10 montre la convergence de l’algorithme évolutif, pour différentes probabilités de mutation p_m . L’erreur ε à une génération est ici mesurée en fonction du meilleur individu (x^*, y^*) de la population, selon

$$\varepsilon = |1 - I(x^*, y^*)| \quad (5.15)$$

On retrouve l’effet déjà noté dans le dessin du mouton: la mutation est requise, mais doit demeurer suffisamment basse sinon la convergence est freinée. À $p_m = 0.1$ on converge à une précision de 10^{-6} en une quarantaine de générations, pour un total de près de 800 évaluations de la fonction $I(x, y)$, ce qui est comparable à la grimpe stochastique de la Figure 5.4. Mais ici, le pic central est localisé à tous les coups! De surcroit, la performance globale de l’approche évolutive se dégrade beaucoup plus lentement que celle de toutes les autres techniques d’optimisation décrites précédemment à mesure que la dimensionnalité de l’espace de recherche augmente.

Vous imaginerez sans difficultés que le comportement évolutif de l’algorithme peut être fortement influencé non seulement par les valeurs choisies pour p_c et p_m , mais aussi par d’autres facteurs comme la taille de la population, la pression de sélection, le choix des distributions statistiques contrôlant la taille des mutations, etc. Les choix ci-dessus sont bons, mais certainement pas optimaux, dans le sens que d’autres valeurs des divers paramètres évolutifs pourraient bien accélérer la convergence *pour ce problème de diffraction 2D*. Cependant, les meilleures valeurs de paramètres évolutifs pour un autre problème d’optimisation seraient fort probablement différentes. En général, il est préférable de s’en tenir à des valeurs qui, sans être vraiment optimales pour aucun problème spécifique, donnent des résultats acceptables pour une vaste gamme de problèmes très différents. Une méthode d’optimisation ayant cette propriété est dite **robuste**.

Typiquement, le facteur le plus critique pour la performance d’un algorithme évolutif est le choix de la probabilité de mutation, et de l’intervalle d’amplitude de cette dernière (e.g., le σ dans l’éq. (5.14)). Il existe plusieurs stratégies visant à contrôler de manière adaptive ces paramètres, en fonction de la distribution de la population dans l’espace des paramètres. L’exploration numérique suggérée dans le cadre du projet qui clot ce chapitre vise à vous en faire explorer quelques-unes.

Notons finalement qu’en posant $p_c = 0$ et $p_m = 1$, on se retrouve avec un algorithme qui commence fort à ressembler à une grimpe stochastique opérant en mode parallèle.

5.5 Les algorithmes génétiques

Les **algorithmes génétiques** sont une classe d’algorithmes évolutifs qui poussent encore plus loin l’analogie biologique, cette fois au niveau de l’encodage des paramètres du modèle et des opérateurs de reproduction. Plutôt que d’effectuer la reproduction (§5.3.6) par moyenne stochastique (éqs. (5.9)–(5.10)) et perturbation ((éq. 5.14)), les paramètres définissant chaque

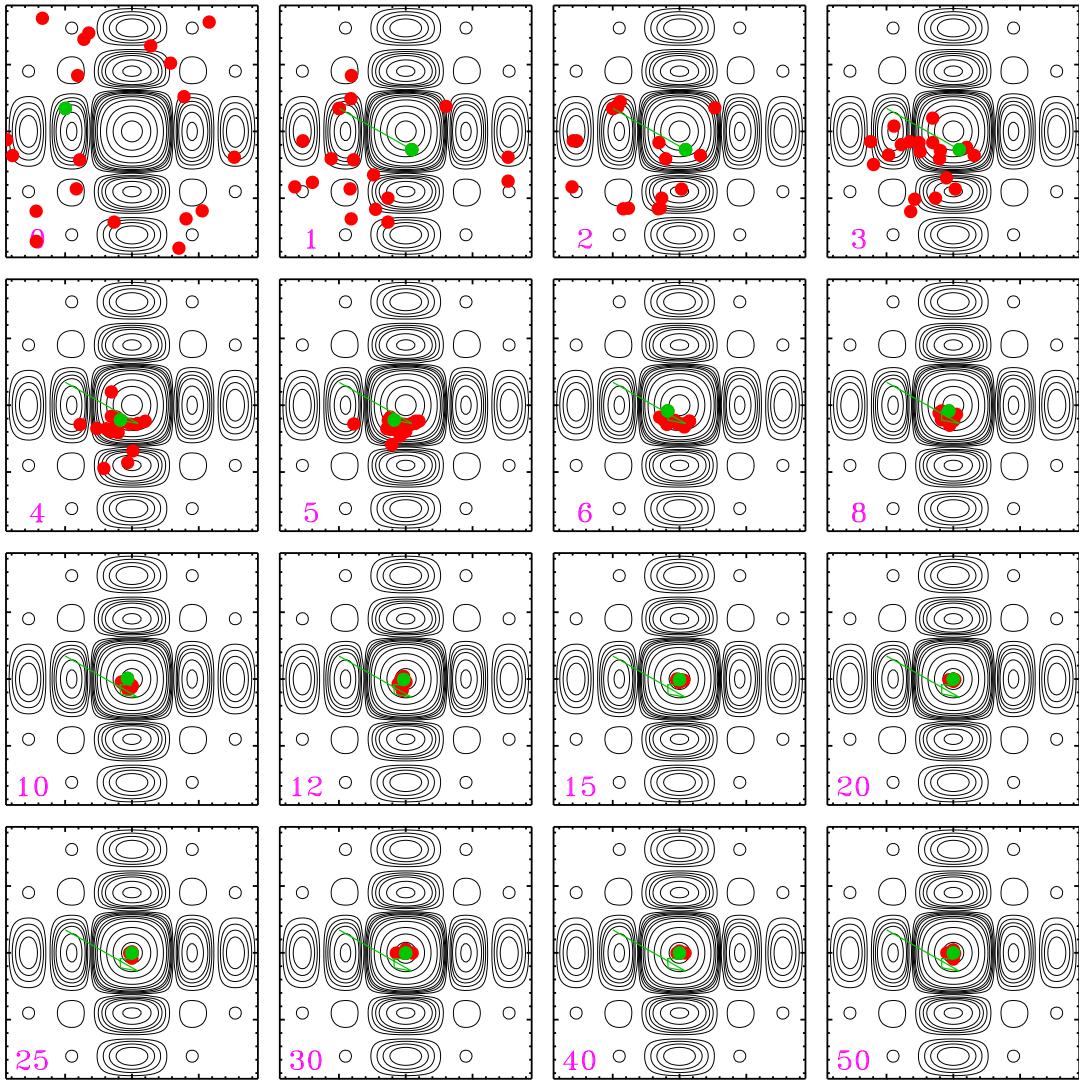


Figure 5.9: Recherche du maximum central du patron de diffraction 2D par calcul évolutif. L'espace de recherche couvre ici l'intervalle $[-3\pi, 3\pi]$ en x et y . Le point coloré en vert correspond au meilleur individu de sa génération (numérotée dans le coin inférieur gauche de chaque image). Notez comment aucun des membres de la population initiale aléatoire n'est tombé suffisamment près du pic central pour qu'une grimpe classique ne conduise à l'extremum global à $(x, y) = (0, 0)$.

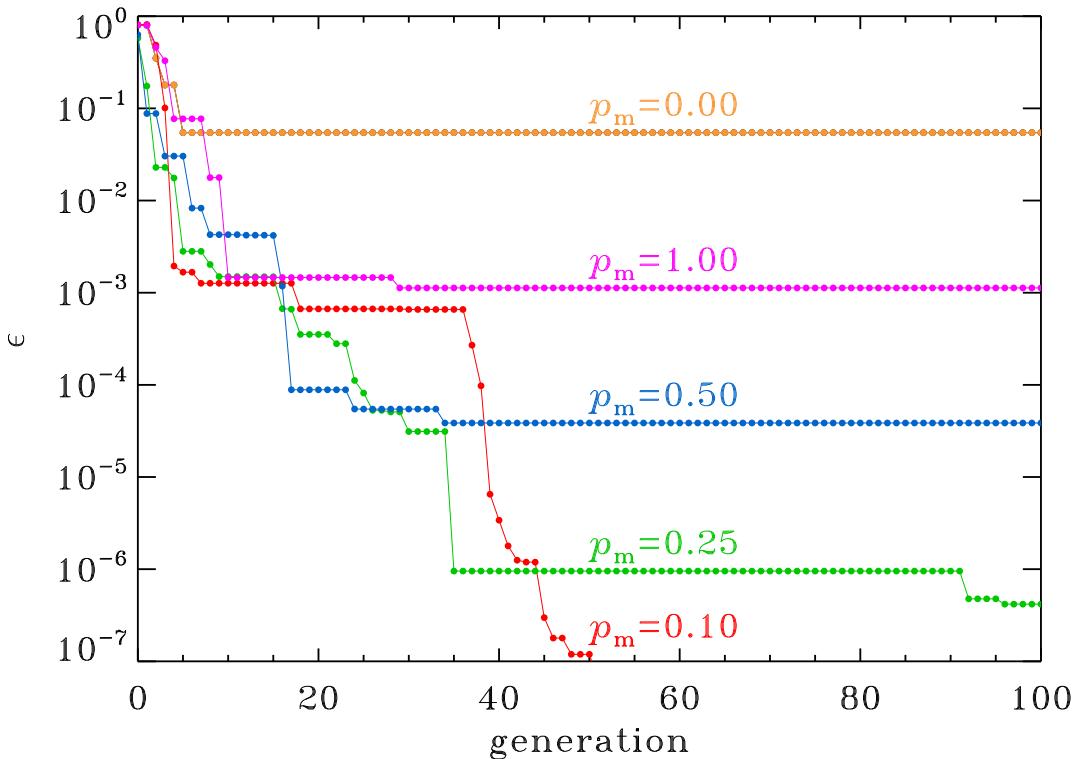


Figure 5.10: Convergence de l'algorithme évolutif de base pour différentes probabilités de mutation p_m , tel qu'indiqué, toujours pour le problème de diffraction 2D.

“individu” sont encodés en une chaîne de bits (le “chromosome”), et le croisement entre deux individus est effectué en coupant les deux chaînes à un bit choisi aléatoirement, et en interchangeant les fragments de chaînes situés plus loin que le point de coupe. Les deux chaînes résultantes, qui incorporent maintenant des morceaux *intacts* des chromosomes parentaux. Le processus de mutation consiste maintenant à inverser ($0 \rightarrow 1$ ou $1 \rightarrow 0$) un ou plusieurs bits choisis aléatoirement sur chaque chaîne. Ces chaînes sont finalement décodées pour produire les valeurs numériques des paramètres définissant l’instance du modèle associée à chacun des deux rejetons.

L’avantage de procéder ainsi vient du fait que des blocs de bits contigus qui confèrent à leur porteurs une mesure de qualité supérieure à la moyenne voient leur fréquence augmenter géométriquement dans le pool génétique. Ceci est quantifié et formalisé par le **théorème fondamental** des algorithmes génétiques, dû à John Holland, un des doyens-fondateurs de ces méthodes. Voir la bibliographie en fin de chapitre si vous êtes intéressé(e)s à en apprendre plus là-dessus.

5.6 Modélisation d'une orbite Keplerienne

5.6.1 Les étoiles binaires

Plus de moitié des étoiles répertoriées dans le voisinage galactique du soleil font partie d'un système binaire. Cette haute fréquence de la binarité est une conséquence de la conservation du moment cinétique, qui tend à causer la fragmentation des nuages protostellaires dans les stades avancés de leur effondrement gravitationnel. Certains systèmes binaires peuvent être séparées visuellement à l'aide d'un télescope de taille modeste, la première ayant été découverte de

cette manière par Riccioli en 1650! Cependant, la vaste majorité des binaires connues ont été identifiées par le décalage Doppler de leurs raies spectrales, causé par le mouvement orbital. Pour des vitesses orbitales non-relativistes, le décalage en longueur d'onde λ est donné par

$$\frac{\Delta\lambda}{\lambda} \simeq \frac{V}{c}, \quad (5.16)$$

où c est la vitesse de la lumière et V la *vitesse radiale*, soit la projection de la vitesse orbitale sur la ligne de visée. La première *binaire spectroscopique* fut détectée de cette façon en 1889 et, coïncidence amusante, était l'étoile Mizar A, soit un des deux membres de la binaire visuelle découverte en 1650 par Riccioli! Et les spectaculaires améliorations des télescopes, particulièrement en optique adaptive, ont depuis permis de résoudre visuellement l'orbite de cette binaire spectroscopique; on n'arrête pas le progrès...

La Figure 5.11 montre une série de mesures de vitesse radiale de l'étoile η Bootis, une célèbre binaire spectroscopique. La composante dite *primaire* du système est ici beaucoup plus brillante que sa compagne, et donc domine le spectre, ce qui fait que les variations observées caractérisent le mouvement d'orbital d'une seule des deux étoiles du système. Ces données sont typiques d'encore bien des mesures de vitesses radiales: elles couvrent un nombre restreint d'orbites, ne sont pas distribuées uniformément dans le temps, et leur qualité/précision varie d'une mesure à l'autre même quand, comme ici, presque toutes les mesures ont été prises par le même instrument vissé au même télescope, en raison de variations dans les conditions météo (et, pour les données de la Fig. 5.11, par la qualité inégale des plaques photographiques utilisées!). La découverte des premières exoplanètes il y a une vingtaine d'années a causé un regain d'intérêt envers les mesures de vitesses radiales, et les méthodes utilisées pour en extraire les paramètres orbitaux du système. Cette information permet éventuellement de calculer la masse des composantes du système, information cruciale pour valider (entre autre) les modèles de la structure et évolution des étoiles, et jauger de l'habitabilité potentielle des exoplanètes.

5.6.2 De la vitesse radiale à l'orbite Keplerienne

On sait depuis Kepler, et on comprend depuis Newton, que deux corps massifs orbitent autour de leur centre de masse le long d'orbites en général de forme elliptique, le carré de la période orbitale (P) étant proportionnel au cube de l'axe semi-majeur (a) de l'ellipse; la constante de proportionnalité étant donnée, à une constante bien connue près, par la masse totale du système:

$$\frac{P^2}{a^3} = \frac{4\pi^2}{G(M_1 + M_2)} \quad (5.17)$$

La détermination des variations attendues de la vitesse radiale observée (V) pour un système binaire dont l'axe semi-majeur de l'ellipse est orienté arbitrairement dans l'espace par rapport à la ligne de visée se réduit à un problème quelque peu laborieux de trigonométrie sphérique; le résultat en est:

$$V(t) = V_0 + K(\cos(\omega + v(t)) + e \cos(\omega)). \quad (5.18)$$

La quantité V_0 est la vitesse radiale du centre de masse du système, après soustraction de la vitesse orbitale de la Terre autour du soleil. La quantité K est l'amplitude de vitesse, qui se retrouve définie en terme des paramètres orbitaux:

$$K = \frac{2\pi}{P} \frac{a \sin i}{(1 - e^2)^{1/2}}, \quad (5.19)$$

où e est l'excentricité de l'orbite, et i l'inclinaison du plan de l'orbite par rapport au plan du ciel ($i = 0$ correspond à une orbite dans le plan du ciel, et $i = \pi/2$ à une orbite vue par la tranche). L'anomalie de la vitesse projetée v est reliée à l'anomalie excentrique E via la relation:

$$\tan \frac{v}{2} - \sqrt{\frac{1+e}{1-e}} \tan \frac{E}{2} = 0; \quad (5.20)$$

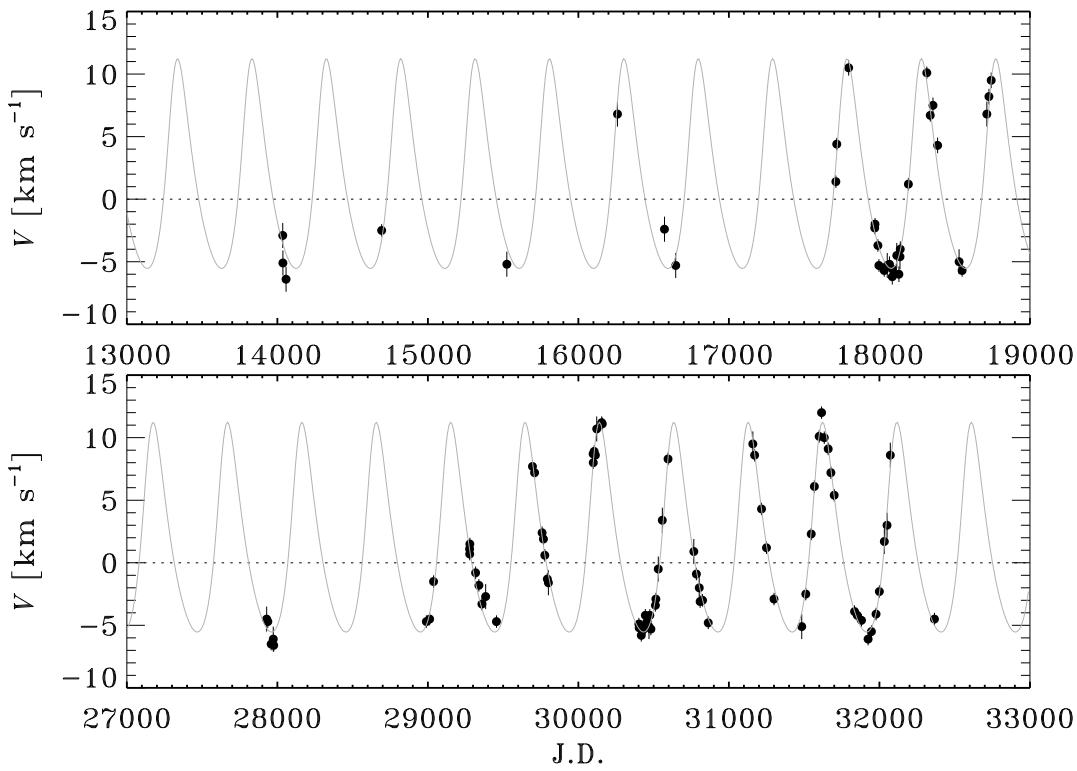


Figure 5.11: Variation de la vitesse radiale (V) observée dans la binaire spectroscopique η Boötis. Le temps est mesuré selon un standard astronomique appelé *date julienne* ($1\text{JD} \equiv 1$ jour; $\text{JD}=30000$ correspond ici au 06/01/1941; pour avoir la date julienne officielle, ajouter 2400000). Les données sont tirées de Bertiau (1957, ApJ, **125**, 696). La vitesse orbitale de la Terre a été soustraite des données. Le trait gris pâle est une solution obtenue par monocle à l'aide d'un algorithme évolutif. Ces données sont disponibles sur la page web du cours.

et l'anomalie excentrique E est reliée au temps via *l'équation de Kepler*:

$$E - e \sin(E) = \frac{2\pi}{P}(t - \tau), \quad (5.21)$$

où τ est le temps de passage au périhélium. Cette expression qui doit donc être solutionnée pour E à des e , τ et t donnés. Ceci définit un problème (nonlinéaire) de recherche de racine, que vous aurez à solutionner numériquement dans ce qui suit; Kepler, évidemment, faisait tout ça crayon-papier, comme un grand...

Bon, on respire quelques fois par le nez et on repasse de nouveau sur le développement ci-dessus et on fait le bilan des paramètres inconnus dont nous devrons ajuster les valeurs pour reproduire au mieux les observations de vitesses radiales de la Fig. 5.11. Il y en a un total de six:

1. P : la période orbitale;
2. τ : le temps de passage au périhélium;
3. ω : la longitude du périhélium;
4. e l'excentricité orbitale;
5. K l'amplitude de vitesse orbitale;

6. V_0 la vitesse radiale du centre de masse.

Il sera pratique pour la suite de regrouper ces paramètres dans un tableau de longueur 6:

$$\mathbf{u} = (P, \tau, \omega, e, K, V_0) \quad (5.22)$$

Chaque réalisation de ce tableau représente donc une solution-essai, la matière première sur laquelle opère notre algorithme évolutif!

5.6.3 Une solution par algorithme évolutif

L'idée est maintenant d'utiliser un algorithme évolutif pour ajuster une orbite aux données. Il faut d'abord établir l'étendue de l'espace des paramètres (ici 6-dimensionnel) dans lequel on distribuera aléatoirement notre population initiale. Certains paramètres peuvent être contraints partiellement à partir des données, tandis que d'autre sont bornés par définition. Le tableau 5.1 donne les intervalles utilisés pour obtenir la solution de la Fig. 5.11.

Tableau 5.1: Paramètres du problème de Kepler

Paramètre	position	intervalle	unités
P	$u[0]$	$200 \leq P \leq 800$	J.D.; estimé des données
τ	$u[1]$	$t_0 \leq \tau \leq t_0 + P$	J.D.
ω	$u[2]$	$0 \leq \omega \leq 2\pi$	radian; intervalle admissible
e	$u[3]$	$0 \leq e \leq 1$	adimensionnel; intervalle admissible
K	$u[4]$	$0 \leq K \leq \max(V_j) - \min(V_j)$	km s^{-1} estimé des données
V_0	$u[5]$	$\min(V_j) \leq V_0 \leq \max(V_j)$	km s^{-1} estimé des données

On passe maintenant à la définition de la fonction mérite ($f(\mathbf{u})$). C'est là toujours un aspect clef, car celle-ci représente le seul point de contact entre l'algorithme évolutif et le problème physique. Ici on basera la fonction mérite sur le χ^2 calculé entre les observations et les "prédictions" du modèle de Kepler pour un \mathbf{u} donné:

$$\chi^2(\mathbf{u}) = \frac{1}{N-6} \sum_{j=0}^{N-1} \left(\frac{V_j^{\text{obs}} - V(t_j; \mathbf{u})}{\sigma_j} \right)^2 \quad (5.23)$$

où les V_j^{obs} représentent les $j = 0, 1, \dots, N-1$ mesures observationnelles, les $V(t_j; \mathbf{u})$ sont les vitesses calculées selon le modèle de Kepler aux t_j correspondant aux observations pour un \mathbf{u} donné, et les σ_j sont les estimés d'erreur sur les observations. Le préfacteur $1/(N-6)$ correspond au nombre de degrés de liberté du fit, ce qui fait que la "valeur-cible" pour le χ^2 est alors ~ 1 ; ceci correspond à une déviation quadratique moyenne entre les données et le modèle égale à l'estimé d'erreur sur les données. Faire mieux n'est pas vraiment justifiable physiquement ou statistiquement: s'acharner à pousser sous $\chi^2 = 1$, c'est "fitter le bruit". Il ne faut pas trop s'inquiéter non plus si vous ne parvenez pas à atteindre $\chi^2 \simeq 1$: il y a toujours des erreurs systématiques cachées quelque part, et donc les erreurs ne sont habituellement pas distribuées de manière Gaussienne; et de manière plus générale, les estimés d'erreur σ_j sont souvent sous-estimés !

Comme notre algorithme évolutif est conçu de manière à *maximiser* la fonction mérite (f), on peut définir celle-ci par exemple comme:

$$f(\mathbf{u}) = \frac{1}{\chi^2(\mathbf{u})}. \quad (5.24)$$

La définition exacte de f n'est pas critique, dans le sens que c'est le rang des solutions-essais, basé sur leur valeur de f , qui détermine la probabilité de sélection; autrement dit, prendre la

racine carrée du membre de droite de l'éq. (5.24), ou le multiplier par 42, ne changera rien au rang, et donc à la probabilité de sélection.

Il s'agit donc maintenant de bâtir une fonction qui calcule une valeur de f pour un vecteur solution-essai donné (viz. éq. 5.22). Les étapes sont les suivantes:

1. Pour un t_j donné, et une solution-essai (un tableau \mathbf{u}), solutionner le problème de recherche de racine défini par l'éq. (5.21); la bisection fonctionne très bien ici (voir Fig. 5.12 ci-dessous), mais vous devez donner un intervalle initial de recherche suffisamment large, soit de l'ordre de $\pm P$; ici $[-300., 300.]$ fonctionne bien. Attention, du point de vue de ce calcul t_j est considéré fixe.
2. Connaissant maintenant $E(t_j)$, calculer $v(t_j)$ via l'éq. (5.20).
3. Connaissant maintenant $v(t_j)$, calculer $V(t_j)$ via l'éq. (5.18).
4. Une fois ces trois étapes complétées pour tous les t_j , calculer le χ^2 pour cette solution-essai via l'éq. (5.23), et la fonction-mérite f via (5.24).

Le trait en gris pâle sur la Fig. 5.11 montre une solution obtenue à l'aide d'un algorithme évolutif structuré tel que décrit ci-dessus. Le vecteur solution est donné ici par

$$\mathbf{u} \equiv (P, \tau, \omega, e, K, V_0) = (494.20, 14299.0, 328.86, 0.2626, 8.3836, 1.0026); \quad (5.25)$$

voir le Tableau 5.1 pour les unités.

5.7 Projet: mutation adaptive et l'orbite de ρ CrB

Tout au long des §5.2 et 5.3 il a été répété *ad nauseam* que le choix du taux de mutation est un aspect critique de tout algorithme évolutif; mais l'algorithme évolutif introduit à la §5.3 utilise un taux de mutation fixe. L'exploration numérique qui clot ce chapitre vise à vous faire développer un opérateur de mutation qui soit adaptif, i.e., qui s'ajuste automatiquement en fonction du niveau de convergence de la solution.

Votre première tâche est de construire un algorithme évolutif utilisant les différents éléments introduits à la §5.3. Vous pouvez valider cet algorithme sur le problème de diffraction 2D introduit en début de chapitre. Vous devriez pouvoir reproduire le comportement général caractérisant les Figs. 5.9 et 5.10, mais pas le détail, en raison des aspects stochastiques de l'algorithme. Assurez-vous de structurer votre code de manière à ce que son seul point de contact avec le problème traité soit le calcul de la fonction-mérite; de cette manière, en passant à la suite vous n'aurez qu'à substituer dans votre code une nouvelle fonction.

Il s'agit maintenant de mettre au point une procédure d'ajustement adaptif de la mutation; vous pouvez ajuster la probabilité de mutation (p_m), ou encore son amplitude, via la largeur de la distribution gaussienne utilisée (le σ dans l'éq. (5.14)). La décision d'adapter (ou pas) les paramètres de mutation se fait à partir d'une mesure de la concentration ou dispersion des membres de la population dans l'espace des paramètres. On peut imaginer (au moins) deux classes de critères:

1. basé sur la valeur de la fonction mérite: on mesure la différence (absolue ou relative) entre la valeur de la fonction-mérite du meilleur individu et de celui de rang médian dans la population:

$$f(\text{rang}[N - 1]) - f(\text{rang}[N/2]),$$

et quand cette différence tombe sous un certain seuil, on augmente la mutation; ou,

2. basé sur la distance dans l'espace des paramètres entre le meilleur et le médian, toujours basé sur le rang:

$$\sum_k (x_k(\text{rang}[N - 1]) - x_k(\text{rang}[N/2]))^2,$$

```

1 # FONCTION BISSEC: RECHERCHE DE RACINE DANS [x1,x2] PAR BISSECTION
2 def bissec(x1,x2,epsilon,ecc,P,tau,tj):
3     itermax=40           # nombre maximal d'iterations de la bisection
4     delta=x2-x1
5     k    =0               # compteur d'iteration de la bisection
6     while (delta > epsilon) and (k <= itermax):
7         xm=0.5*(x1+x2)   # essai de racine
8         fm=func(xm,ecc,tj,P,tau)
9         f2=func(x2,ecc,tj,P,tau)
10        if fm*f2 > 0.:   # racine a droite
11            x2=xm
12        else:             # racine a gauche
13            x1=xm
14        delta=x2-x1
15        k+=1
16        return xm          # la racine recherchée
17 # END fonction bissec
18 #-----
19 # PROGRAMME PRINCIPAL (fragment)
20 ...                      # initialisations, etc.
21 epsilon=1.e-5            # précision recherchée sur la racine
22 ...                      # boucle sur les t_j, etc.
23 for j in range(0,N_data): # boucle sur chaque point de donnée
24     x1    =-300.           # borne inf. de l'intervalle de recherche
25     x2    = 300.           # borne sup. de l'intervalle de recherche
26     E=bissec(x1,x2,epsilon,ecc,P,tau,t[j])
27     ...                  # la suite du calcul du V(t_j) pour ce E
28 # END

```

Figure 5.12: Fonction Python pour la recherche de racine par bisection, avec un fragment de code en montrant l'instruction d'appel. Ici la fonction `func` correspond au membre de gauche de l'éq. (5.21), `x1` et `x2` ($> x1$) définissent l'intervalle à l'intérieur duquel on recherche la racine, `epsilon` est la précision demandée sur la racine, et sur sortie de la boucle `xm` est la racine recherchée. Ce fragment de code pourrait être imbriqué dans un boucle sur les t_j , ou prendre la forme d'une fonction Python invoquée à l'intérieur de la boucle sur les t_j .

où la somme se fait dans une version normalisée en $[0, 1]$ dans chacune des k dimensions de l'espace des paramètres.

L'idée d'augmenter les effets de la mutation quand la solution semble avoir convergé est d'augmenter la probabilité de se décoincer d'un extremum secondaire. Attention, il est essentiel ici d'utiliser l'élitisme, tel qu'introduit à la §5.3.7, afin de ne pas régresser en détruisant votre meilleure solution avec une mutation trop intense.

Développez votre propre stratégie d'adaptation de la mutation, et testez la sur le problème de diffraction 2D ($x, y \in [-5\pi, 5\pi]$); en ensuite sur une version (artificielle) en 5D:

$$f(\mathbf{x}) = \prod_{i=1}^5 \left(\frac{\sin x_i}{x_i} \right), \quad x_i \in [-5\pi, 5\pi].$$

Une fois convaincu(e) que votre algorithme évolutif fonctionne correctement, solutionnez le problème de modélisation de données offert par la Fig. 5.11. Comparez la performance de

votre algorithme avec celle d'une version avec mutation fixe. Les données de vitesses radiales, incluant les estimés d'erreurs, sont disponibles via la page Web du cours.

Une fois convaincu(e) que ça marche pour η Boo, votre dernière tâche est de calculer les paramètres orbitaux du système étoile+exoplanète ρ CrB, une des premières exoplanètes découverte sur la base de mesures de vitesses radiales. Les données de vitesses radiales, ainsi que les estimés d'erreurs leur étant associés, sont également disponibles via la page Web du cours.

5.8 Bibliographie

L'exemple de l'évolution de la phrase littéraire de la §5.2 est adapté de l'excellent ouvrage “grand public”:

Dawkins, R., *The Blind Watchmaker*, W.W. Norton (1986),

mais j'ai changé la phrase-cible, celle de Dawkins étant tirée de Shakespeare, comme on s'y attendrait d'un Brit d'Oxford.

Les algorithmes évolutifs sont un sujet sur lequel j'ai déjà pas mal travaillé dans le passé, et sont en fait une des rares techniques d'optimisation d'intérêt qui ne soit pas discutée dans *Numerical Recipes* auquel j'ai si souvent fait référence. L'ouvrage suivant demeure une bonne introduction au sujet:

Bäck, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press (1996).

Sur l'optimisation globale en général, et plus spécifiquement sur l'optimisation par algorithmes génétiques, si vous voulez en savoir plus là-dessus, j'ai écrit il y a quelques années une petite introduction au sujet dont je ne suis pas mécontent:

Charbonneau, P., *An introduction to genetic algorithms for numerical optimization*, NCAR Technical Note 450+IA, National Center for Atmospheric Research (2002)

Ca peut s'obtenir du NCAR via le Web:

<http://www.hao.ucar.edu/modeling/pikaia/pikaia.php>

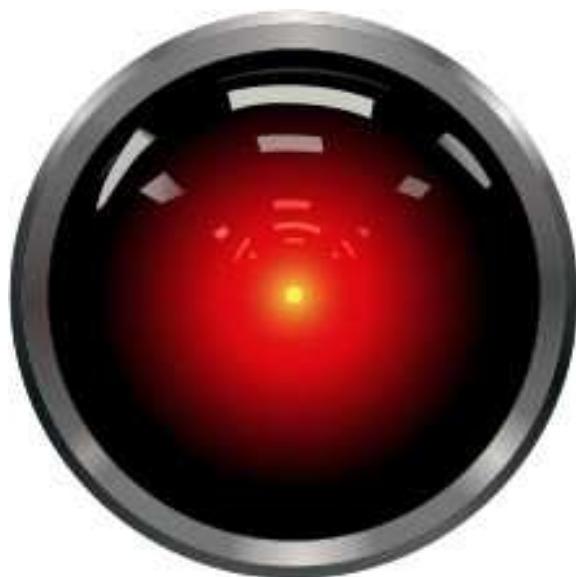
sous “Oslo Tutorial” ...ou vous pouvez passer à mon bureau ramasser une des copies papier que j'ai déménagé à Montréal... Le grand classique théorique dans le domaine, et solidement technique, demeure

Holland, J.H., *Adaptation in natural and artificial systems*, 2^e éd., MIT Press (1992).

Tous les bouquins d'introduction à l'astronomie discutent en long en et large des orbites Képlériennes et des systèmes binaires, mais la majorité s'arrêtent à l'éq. (5.17) et bien peu présentent une dérivation formelle des éqs. (5.18)–(5.21); si le sujet vous intrigue, voir par exemple

Smart, W.M., *Textbook on Spherical Astronomy*, 5^e éd., Cambridge University Press (1971) (c'est un sujet multi-centenaire!). Pour une brève et accessible introduction aux étoiles binaires et leurs modes de détection, voir:

http://www.atnf.csiro.au/outreach/education/senior/astrophysics/binary_types.html



Chapitre 6

Les réseaux de neurones artificiels

Le chapitre précédent était motivé par une tâche des plus courantes en sciences, physique incluse: ajuster un modèle à des données; ceci supposait implicitement qu'on savait déjà quel modèle physique devait être ajusté aux dites données. Ce n'est pas toujours le cas; on doit parfois d'abord, sur la base de certaines caractéristiques associées au données, choisir parmi plusieurs modèles potentiellement applicables aux données. Autrement dit, on doit d'abord *classer* les données avant de les modéliser. Par exemple, dans le cas du problème de calcul d'orbites, avant d'appliquer le modèle physique décrit à la §5.6.2 on peut imaginer devoir d'abord se convaincre que les variations observées de la vitesse radiales obtenue spectroscopiquement sont bien associées au mouvement orbital de deux corps autour de leur centre de masse respectif, plutôt qu'au mouvement de trois corps ou plus, comme dans un système multiple, ou encore à un processus physique complètement différent, comme les pulsations stellaires.

De manière tout à fait générale, un problème de classification consiste à assigner une *catégorie* à un ensemble de données associées à une instance spécifique (et habituellement bruitée) d'un phénomène observé. Quelques exemples spécifiques clarifieront ce concept mieux que ne pourrait le faire un long discours:

1. Sur la base d'une image pixellisée (possiblement bruitée) de quelque chose, décider si l'image contient un ou plusieurs objets/structures possiblement d'intérêt.
2. Sur la base d'un scan pixellisé d'une enveloppe, extraire automatiquement les six lettres et chiffres du code postal;
3. Sur la base d'un scan aux rayons-X, décider si *oui* ou *non* une tumeur est présente;
4. Sur la base d'un ensemble de symptômes physiologiques (température corporelle, décompte de globules blancs, présence/absence de sécrétion nasales, etc.) poser un diagnostic médical (e.g., allergie, Ebola, peste bubonique, etc);
5. Sur la base de mesure des vitesses et énergies des particules secondaires produites par une collision très énergétiques au CERN, décider quelles particules primaires ont été produites par la collision.

Il existe plusieurs approches possibles à ce genre de problèmes de classification. Si le nombre de données en entrée et le nombre de catégories possibles n'est pas trop élevé, il est possible de tabuler toutes les combinaisons possibles d'associations donnée → catégories; c'est rarement une approche pratique cependant, sauf pour les tâches de classification particulièrement simples. Une autre option est offerte par les *systèmes experts* basées sur une série de critères de sélection (déterministes ou probabilistes). Ceux-ci se traduisent en *arbres de décision*, qui en pratique prennent la forme d'une longue série d'instruction imbriqués du genre **if...else...** C'est le

principe des clefs de classification taxonomique en biologie. Coder ainsi un système expert peut rapidement devenir un cauchemar de programmation si le nombre de données en entrées est grand.

6.1 Les réseaux de neurones (artificiels)

Les *réseaux de neurones artificiels* offrent une option attrayante et numériquement efficace qui demeure applicable à des systèmes impliquant un nombre relativement élevé de données en entrée. De plus, comme on le verra plus loin les données sont traitées globalement en parallèle par le réseau, plutôt que séquentiellement comme dans un arbre de décision, ce qui confère aux réseaux de neurones une capacité de *généralisation*, qui est typiquement absente des systèmes experts conventionnels.

En toute généralité, un réseau de neurone est constitué d'une série d'unités de traitement (ci-après "neurones") qui reçoivent des informations d'autres neurones, et effectuent un calcul interne à partir de ces informations pour produire un signal de sortie qui est retransmis aux autres neurones. Dans tout ce qui suit on considérera un réseau composé de neurones organisées en *couches*, où chaque neurone d'une couche reçoit l'information des neurones dans la couche à sa "gauche" et transmet son information aux neurones dans la couche à sa "droite". De tels réseaux sont appelés *perceptron*, ou encore plus couramment *réseaux feed-forward*, car ils traitent l'information d'une manière unidirectionnelle, de la couche d'entrée à la couche de sortie via les couches internes. La Figure 6.1 illustre cette architecture, dans le cas d'un petit réseau composé de trois couches où la couche d'entrée compte 4 unités, l'unique couche interne compte deux neurones, et la couche de sortie est faite d'un seul neurone. À proprement parler les unités de la couche d'entrée ne sont pas des neurones dans le sens de celles des couches internes et de sortie, car elles n'effectuent habituellement aucun calcul; elles ne font que distribuer l'information aux neurones de la première couche interne. C'est pourquoi un symbole différent a été utilisé sur la Fig. 6.1. On assigne parfois aux unités de la couche d'entrée une tâche de pré-traitement (e.g., normalisation) des données en entrée, dans lequel cas elles se retrouvent elles aussi à exécuter un calcul, mais ce ne sera pas le cas pour les réseaux considérés dans ce chapitre. Établissons pour commencer la notation utilisée dans ce qui suit:

- I_i : donnée présentée au neurone i de la couche d'entrée;
- w_{ij}^{IH} : poids de la connection entre le neurone i de la couche d'entrée et le neurone j de la couche interne;
- S_j^H : signal produit par le neurone j de la couche interne;
- w_{jk}^{HO} : poids de la connection entre le neurone j de la couche interne et le neurone j de la couche de sortie;
- b_k^O : biais appliqué au neurone k de la couche de sortie;
- S_k^O : signal produit par le neurone k de la couche de sortie;
- D_k : signal-cible pour le neurone k de la couche de sortie (dans les phases d'entraînement ou de test);
- e_j : signal d'erreur associé à un neurone quelconque j (couche interne ou de sortie).
- δ_j : gradient d'erreur associé à un neurone quelconque j (couche interne ou de sortie).

Les unités de la couche d'entrée ne font habituellement aucun calcul; les données I_i qui leur sont présentées sont transmises aux neurones de la couche interne, chacune de ces dernières calculant une combinaison linéaire des I_i pondérée par les poids w_{ij}^{IH} associés à chaque connexion (voir Fig. 6.1). Le résultat de ce calcul (a_j^H) est donné en argument à une *fonction d'activation*

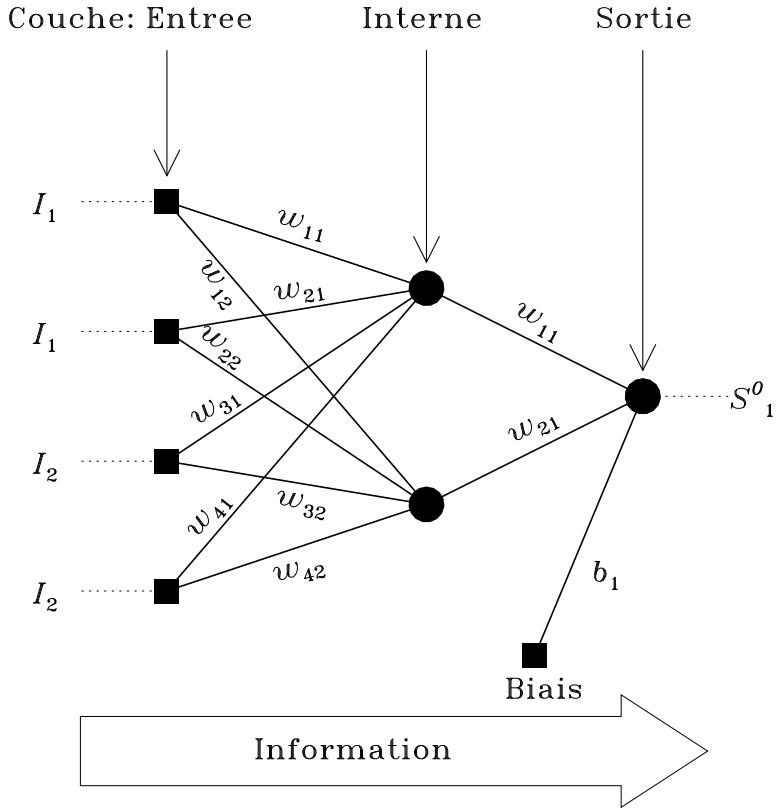


Figure 6.1: Un réseau de neurones artificiel de type “feed-forward”. Ce petit réseau compte 4 unités d’entrée, une couche cachée composée de deux neurones, et une neurone de sortie sujette à un biais. L’information se propage de la gauche vers la droite dans un tel réseau, chaque couche traitant l’information provenant des couches précédentes selon les éqs. (6.1)–(6.3). Les “IH” et “OH” ont été omis sur les poids w afin d’alléger le diagramme.

φ^H qui calcule le signal S_j^H associé au neurone j de la couche interne; pour le réseau de la Fig. 6.1 on aurait donc:

$$a_j^H = \sum_{i=1}^4 w_{ij}^{IH} I_i , \quad j = 1, 2 . \quad (6.1)$$

$$S_j^H = \varphi(a_j^H) , \quad j = 1, 2 . \quad (6.2)$$

Le même calcul est répété pour les neurones de la couche de sortie (avec une addition possible discutée plus bas). Pour le petit réseau de la Fig. 6.1 la couche de sortie de contient qu’un seul neurone, et donc:

$$a_1^O = b_1^O + \sum_{i=1}^2 w_{i1}^{HO} S_i^H . \quad (6.3)$$

$$S_1^O = \varphi(a_1^O) . \quad (6.4)$$

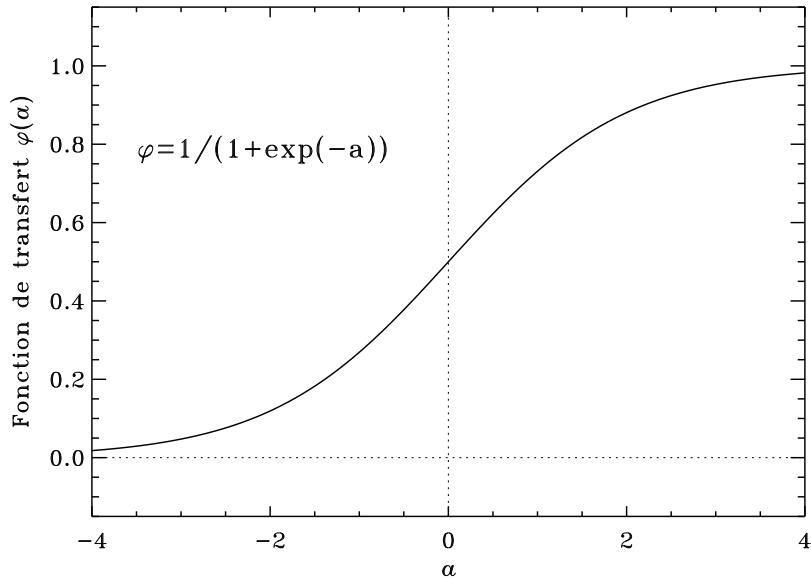


Figure 6.2: Une forme analytique possible (et couramment utilisée) pour la fonction d’activation $\varphi(a)$: la fonction logistique, telle que donnée par l’éq. (6.5). Cette fonction d’activation restreint le signal produit par un neurone à l’intervalle $[0, 1]$, et sera utilisée dans tout ce qui suit.

Le rôle de la fonction d’activation est de contraindre le signal produit par les neurones à un intervalle prédéfini, ici $[0, 1]$. Ceci aide à empêcher qu’une seule branche du circuit neuronique (de la couche d’entrée à la couche de sortie) en vienne à dominer le comportement du réseau. Plusieurs fonctions mathématiques peuvent s’acquitter de cette tâche (voir Fig. 6.2). Dans tout ce qui suit on se limitera à la *fonction logistique* (en trait plein sur la Fig. 6.2):

$$\varphi(a) = \frac{1}{1 + \exp(-a)} . \quad (6.5)$$

La couche interne peut paraître superflue, mais elle est en fait essentielle à la capacité de généralisation du réseau. Elle agit globalement comme un détecteur/encodeur des patrons présents dans les données, qu’elles testent de manière parallèle chaque fois qu’un nouvel ensemble de données est présenté au réseau. Cette déclaration à saveur quelque peu mystique se verra graduellement validée à mesure que l’on progressera dans ce chapitre! Il n’y rien d’essentiel au fait que la couche interne soit constituée de deux neurones sur la Fig. 6.1. Deux est probablement un minimum, mais pour un réseau de type feed-forward il serait inhabituel que la couche interne ait plus de neurones que la couche d’entrée. De même, un réseau feed-forward pourrait (et parfois devrait!) inclure plus d’une couche interne.

Maintenant au sujet de l’addition promise plus haut; On notera, sur la Fig. 6.1, comment le neurone de la couche de sortie reçoit un signal d’un neurone interne au réseau, dont le signal (par définition) est toujours $+1$. L’ajout de telle *unité de biais* n’est pas obligatoire ou toujours utile dans un réseau feed-forward, mais peut le devenir si par exemple on veut que le réseau puisse assigner une classe “par défaut” en présence d’un signal ne contenant aucun pattern encodé par les poids du réseau. Un tel biais peut aussi être appliquée aux autres neurones du réseau, et permet ainsi d’ajuster le “point-zéro” de l’activation des neurones; ceci est souvent utile, particulièrement au niveau des neurones de la couche cachée recevant leurs signaux de la couche d’entrée, si ces signaux d’entrée couvrent une vaste plage de valeurs numériques systématiquement négatives ou positives.

Il faut noter que pour un ensemble de poids w^{IH} et w^{HO} (et les poids de biais b , le cas échéant) donné, le calcul effectué par chaque neurone est donc complètement déterministe, étant entièrement déterminé par les valeurs numériques des signaux d'entrée. Le signal de sortie est donc lui aussi complètement déterministe.

6.2 Entrainer et tester le réseau

6.2.1 Un problème de classification de structures

Un type de problèmes où les réseaux de neurones peuvent s'avérer très utiles (et performant) est la *classification*. Étant donnée une série de caractéristiques en entrée (i.e., un vecteur I_i présenté au réseau), assigner à ces données une *catégorie* (e.g., pomme, orange, banane, etc). Nous faisons ça continuellement dans la vie de tous les jours; imaginez vous revenant à votre appart tard le soir par une ruelle mal éclairée. Un animal bouge quelquepart près d'une poubelle. En un coup d'oeil, vous jaugez la taille approximative de l'animal, sa manière de bouger, la longueur de son poil, couleur peut-être, une odeur dans l'air, et en une fraction de seconde vous “identifiez” l'animal: chien, chat, raton laveur, mouffette, alien, etc.

Dans ce qui suit nous allons considérer un problème de classification conceptuellement simple, de nature dite *binaire*. Le problème est illustré à la Figure 6.3: les “données” du problème sont des chaînes de 12 bit (chaque élément valant 0 ou 1). Pour chaque chaîne de ce genre, le problème de classification consiste à répondre:

- OUI ou NON, existe-t-il dans la chaîne au moins un groupe d'au moins 5 bits contigus valant 1.

Ceci correspond à une version particulièrement simple d'un problème courant en analyse d'image, soit l'identification automatique de structures potentiellement “intéressantes”. Pour ce problème de classification, un réseau feed-forward minimal aurait donc 12 unités d'entrée, et un neurone de sortie, dont le signal de sortie pourrait encoder la solution (réponse) au problème posé ci-dessus:

$$s_1^O = \begin{cases} 0.9 & \text{OUI} \\ 0.1 & \text{NON} \end{cases} \quad (6.6)$$

Le choix des valeurs [0.1, 0.9] plutôt que les valeurs plus “binaires” [0, 1] est une conséquence de notre choix de la fonction logistique (6.5) comme nonlinéarité d'activation neuronique. Cette fonction n'atteint les valeurs de zéro et un que dans les limites $a \rightarrow \pm\infty$, ce qui demanderait des poids de valeurs élevées, et pousserait nos neurones dans leur régime saturé, ce qui nuirait au processus d'entraînement introduit plus bas.

Dans ce qui suit on utilisera une couche interne composé de 6 neurones. Aucun biais ne sera introduit aux unités d'entrée ou aux neurones des couches interne et de sortie. Le “comportement” d'un tel réseau feed-forward est donc défini par les $12 \times 6 + 6 \times 1 = 78$ poids de connexion (les w^{IH} et w^{HO} des éqs. (6.1) et (6.3)). Il s'agit maintenant de choisir ces 78 valeurs de poids de manière à obtenir le comportement désiré. C'est un problème d'optimisation!

Si ceux-celles d'entre nous ayant grandi au Québec, ou du moins en Amérique du Nord, savent si rapidement distinguer un chat d'une mouflette sous des conditions d'éclairage suboptimales, c'est qu'au fil des nos années à nous promener partout le soir nous avons eu plusieurs occasions de voir passer, et identifier, des chats, chiens, rats-laveurs, etc. Notre solution quasi-instantanée du problème de classification “c'est quoi la petite bête qui bouge dans le noir” est *basé sur nos expériences antérieures*, soit le fait que nous ayions déjà fait face à d'autres instances de cette situation, différentant certainement dans les détail, mais comportant des similarités qui nous permettent de *généraliser* à partir de ces expériences antérieures; autrement dit, nous n'avons typiquement aucune difficulté à identifier rapidement une mouflette même dans une ruelle sombre où nous mettons les pieds pour la première fois.

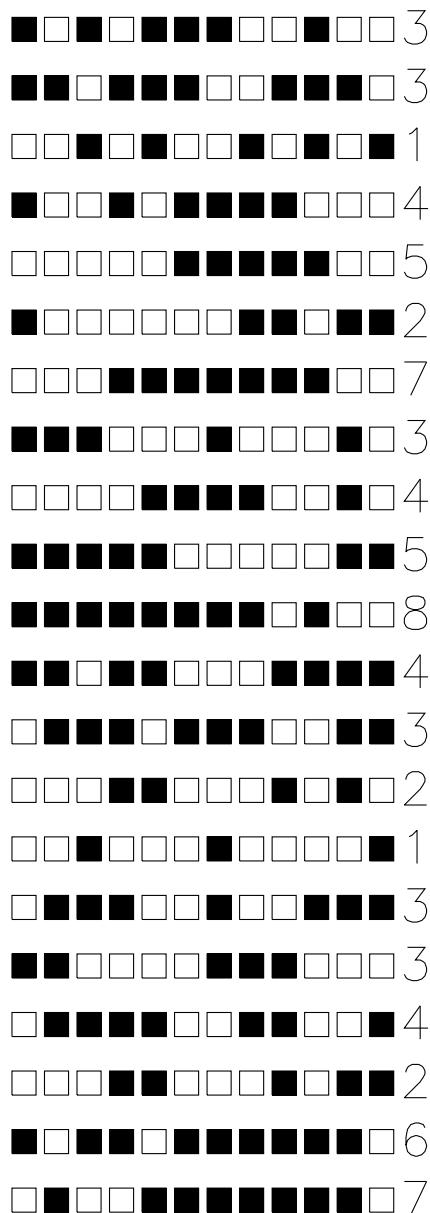


Figure 6.3: Quelques exemples d'instances du problème de classification binaire décrit dans le texte. Chacun des bit d'une chaîne de douze bits peut valoir zéro (carré blanc) ou un (carré noir). La tâche de classification consiste à identifier si, oui ou non, chaque chaîne contient un groupe de bit= 1 contigus, de longueur ≥ 5 . Le chiffre dans la colonne de droite indique ici la taille du plus grand groupe de ce genre, dans chaque exemple présenté

Le choix des poids des connexions du réseau sera donc fait sur la base d'un ensemble d'exemples (l'ensemble dit *d'entraînement*) pour lesquels nous connaissons déjà la classification correcte, dénotée D^m dans ce qui suit. À chacun de ces exemples est associée un tableau de données en entrée (les I_i^m , avec ici $i = 1, \dots, 12$). Supposons que l'on dispose de M exemples de ce genre. Pour chaque exemple on peut définir l'erreur comme la différence entre le signal de sortie du réseau et la réponse désirée, pour chacune des N_O neurones de la couche de sortie:

$$e_k = D_k^m - S_k^m , \quad k = 1, 2, \dots, N_O , \quad (6.7)$$

où S^m est le signal de sortie du réseau lorsque qu'on lui présente les données I^m . Ça, c'est pour un exemple $[I^m, D^m]$. L'erreur associée à l'application du réseau sur tout l'ensemble d'entraînement est habituellement définie selon une moyenne quadratique des e_k^m de l'ensemble:

$$E = \frac{1}{2} \sum_{m=1}^M (e_k^m)^2 . \quad (6.8)$$

où le facteur $1/2$ est ajouté en anticipation de ce qui suit. Il s'agira donc de choisir les poids de manière à minimiser E dans un espace de dimensions égales au nombre de poids requis par l'architecture du réseau (78 ici), pour un ensemble d'entraînement donné. Plusieurs techniques sont possibles (y compris le calcul évolutif!), mais ici on se limitera à l'algorithme classique: la *rétropropagation*.

6.2.2 La rétropropagation

La rétropropagation est effectivement une méthode de grimpe, où l'on cherche à modifier les poids de manière à minimiser E . Comme toute méthode de grimpe qui se respecte, cet ajustement se fait sur la base d'un calcul du gradient de l'erreur dans l'espace des paramètres; exprimé symboliquement, on veux calculer

$$\frac{\partial E}{\partial \mathbf{w}} , \quad (6.9)$$

où \mathbf{w} représente ici tous les poids du réseau. Appliquons la dérivée en chaîne à cette expression, imaginant l'évaluer par rapport à un seul poids w_{ji} pondérant le signal provenant d'un neurone i dans le calcul du signal de sortie du neurone j :

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial S_j} \frac{\partial S_j}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} , \quad (6.10)$$

et examinons chacune des dérivées au membre de droite. En vertu de l'éq. (6.8) —et de son fameux facteur $1/2$ — et avec j remplaçant le k , on a:

$$\frac{\partial E}{\partial e_j} = e_j ; \quad (6.11)$$

en vertu de l'éq. (6.7) on a:

$$\frac{\partial e_j}{\partial S_j} = -1 ; \quad (6.12)$$

et en vertu des éqs. (6.4)–(6.5):

$$\frac{\partial S_j}{\partial a_j} = \varphi'(a_j) , \quad (6.13)$$

où ici $\varphi' \equiv \partial \varphi / \partial a$; et en vertu de l'éq. (6.3) on a:

$$\frac{\partial a_j}{\partial w_{ji}} = S_i , \quad (6.14)$$

où ici S_i est le signal provenant du neurone i connecté au neurone j via la connection de poids w_{ji} . Substituant tout ceci dans (6.10) on obtient:

$$\frac{\partial E}{\partial w_{ji}} = -e_j \varphi'(a_j) S_i . \quad (6.15)$$

L'ajustement du poids est alors donné par:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} , \quad (6.16)$$

où $\eta (> 0)$ est le *paramètre d'apprentissage*, et le signe moins apparaît car on veut se déplacer dans la direction contraire au gradient puisqu'on cherche ici à *minimiser* l'erreur. Il sera pratique de réécrire cette expression sous la forme:

$$\Delta w_{ji} = -\eta \delta_j S_i , \quad (6.17)$$

où

$$\delta_j = e_j \varphi'(a_j) . \quad (6.18)$$

est le *gradient d'erreur* associé au neurone j . On voit ici le danger mentionné précédemment (viz. l'éq. 6.6) d'entrainer le réseau à reproduire des valeurs de sortie de zéro ou de un, exigeant donc $a \rightarrow \pm\infty$; dans ce régime, on a $\varphi'(a_j) \rightarrow 0$, et l'apprentissage s'en retrouve fortement ralenti. Notons que pour la fonction d'activation sigmoidale (6.5), on a

$$\varphi'(a_j) \equiv \frac{\partial \varphi}{\partial a_j} = \frac{\exp(-a_j)}{[1 + \exp(-a_j)]^2} = S_j(1 - S_j) , \quad (6.19)$$

où la dernière égalité résulte du fait que $S_j = \varphi(a_j)$. L'éq. (6.18) devient donc

$$\delta_j = e_j S_j(1 - S_j) . \quad (6.20)$$

Le détail de l'application de la règle d'ajustement des poids (les éqs. (6.16) et (6.20)) n'est cependant pas le même si le neurone j est dans la couche de sortie ou dans une couche interne. Pour un neurone dans la couche de sortie e_j est donné directement par l'éq. (6.7) et on utilise donc directement l'éq. (6.20); mais pour une neurone d'une couche interne, la définition du signal d'erreur est plus complexe car l'erreur du signal de sortie doit être "répartie" entre les neurones internes ayant contribué un signal aux neurones de sortie. Pour ce faire on utilise une pondération des gradients d'erreur δ 's associés à ces neurones:

$$\delta_j = \left(\sum_k \delta_k w_{kj} \right) S_j(1 - S_j) , \quad [j \text{ interne}] \quad (6.21)$$

où la somme sur les k se fait sur les connexions entre le neurone j et les neurones positionnées dans la couche suivante (à droite sur la Fig. 6.1). Par exemple, pour un réseau à une seule couche interne et un seul neurone dans la couche de sortie, comme sur la Fig. 6.1, la somme sur les k se ferait sur cet unique neurone de sortie.

Dans un cas comme dans l'autre le calcul du gradient δ_j exige de connaître les δ associés aux neurones sur lesquels se connecte le neurone j . Le calcul des ajustements de poids Δw_{ji} (qui exige de connaître les δ_j) devra donc se faire en débutant dans la couche de sortie, et continuant couche par couche vers la couche d'entrée, soit dans la direction *contraire* à celle de la propagation des signaux d'entrée dans le réseau; c'est d'ailleurs pourquoi l'algorithme s'appelle "rétropropagation"!

La Figure 6.4 présente un exemple d'implémentation de cet algorithme de rétropropagation, applicable à un réseau sans unités de biais, et dont l'architecture n'inclut qu'une seule couche interne. La fonction `ffnn` calcule un signal de sortie (tableau `so`) à partir d'un signal d'entrée

(tableau `ivec`). La fonction `backprop` ajuste les poids du réseau en propageant le signal d'erreur (tableau `err`) qui lui est fourni en entrée. La généralisation de ces fonctions à plusieurs couches internes et laissée en exercice (aha...!). À noter, les fonctions `ffnn` et `backprop` ne retournent aucune valeur; elles utilisent et agissent sur des variables définies globalement au tout début du code, et qui sont donc accessibles à toutes les unités de programme définis par la suite dans le script/code source (du moins en Python).

L'ajustement des poids du réseau est donc un processus en trois étapes:

1. Calcul du signal de sortie S^O suite à la présentation des données d'entrée I_i au réseau (par la fonction `ffnn`). Durant cette étape tous les poids du réseau demeurent fixes.
2. Calcul du signal d'erreur e_k (éq. (6.7)) associé à chaque neurone définissant le signal de sortie. Ceci peut se faire dans le code principal.
3. Rétropropagation de ce signal d'erreur dans le réseau (fonction `backprop`), avec ajustement des poids effectué couche par couche selon les éqs. (6.16) et (6.18) ou (6.21), selon le cas (neurone de sortie vs interne).

La phase d'entraînement du réseau demandera l'application répétée de cette séquence, selon la procédure décrite dans ce qui suit.

6.2.3 La phase d'entraînement

Le code Python présenté à la Fig. 6.5 montre un protocole typique d'entraînement d'un réseau feed-forward par rétropropagation. Ce code presuppose que les `nset` membres de l'ensemble d'entraînement sont emmagasinés dans deux tableaux, le premier (`tset[nset,ni]`) contenant les signaux d'entrée de tous les membres, et le second (`oset[nset,no]`) le signal de sortie correspondant. Ces données seraient typiquement lues à partir d'un fichier pré-existant, ou encore produites par une série d'instructions supplémentaires ajoutées au début de ce code.

Avant le début de l'entraînement, les poids du réseau sont initialisés à des valeurs aléatoires restreintes à l'intervalle $[-0.5, 0.5]$ (lignes 22–27). L'entraînement est structuré en terme d'une boucle externe sur les itérations d'entraînement (débutant à la ligne 29), et d'une boucle interne sur les membres de l'ensemble d'entraînement (débutant à la ligne 33). Pour chaque membre de l'ensemble, on calcule d'abord le signal de sortie par un appel à la fonction `ffnn` de la Figure 6.4 (ligne 35), ensuite on calcule le signal d'erreur `err` associé à chaque neurone de la couche de sortie (lignes 36–38), et finalement on effectue la rétropropagation (appel à la fonction `backprop`, ligne 39). L'erreur rms totale d'entraînement est calculée à la fin de chaque itération (ligne 42). Plusieurs choses importantes sont à remarquer ici:

1. L'ordre dans lequel les membres de l'ensemble d'entraînement sont présentés au réseau varie aléatoirement d'une itération d'entraînement à la suivante. On accomplit ceci en utilisant le rang des éléments d'un tableau de longueur `nset` initialisé aléatoirement pour établir la séquence de présentation; c'est le rôle de la fonction `randomize`, qui est appelée au début de chaque itération.
2. La rétropropagation est appliquée séquentiellement pour chacun des membres de l'ensemble d'entraînement, donc tous les poids du réseau sont ajustés `nset` fois par itération d'entraînement;
3. Ce fragment de code, comme les fonctions de la Fig. 6.4, peut fonctionner pour un nombre `no > 1` de neurones dans la couche de sortie, mais ici est utilisé avec `no = 1`.

La Figure 6.6 montre le résultat de l'entraînement d'un réseau feed-forward appliqué au problème de classification binaire de la Figure 6.3. On a utilisé ici une couche de 6 neurones internes, un paramètre d'entraînement $\eta = 0.1$, et 100 membres (précalculés) pour l'ensemble d'entraînement, dont la moitié contiennent des groupes de plus de 5 bits non-nuls contigus (classe "1") et l'autre moitié non (classe "0"). Le trait noir montre l'évolution de l'erreur rms d'entraînement, définie ici comme

```

1 # La retropropagation pour un reseau feed-forward avec une couche interne
2 #=====
3 import numpy as np
4 ni,nh,no =12,6,1           # nombre d'unités d'entrée, interne et de sortie
5 wih =np.zeros([ni,nh])      # poids des connexions entrée vers interne
6 who =np.zeros([nh,no])      # poids des connexions interne vers sortie
7 ivec =np.zeros(ni)          # signal en entrée
8 sh =np.zeros(nh)            # signal des neurones interne
9 so =np.zeros(no)            # signal des neurones de sortie
10 err =np.zeros(no)          # signal d'erreur des neurones de sortie
11 deltao=np.zeros(no)        # gradient d'erreur des neurones de sortie
12 deltah=np.zeros(nh)        # gradient d'erreur des neurones internes
13 eta =0.1                  # parametre d'apprentissage
14 -----
15 # Fonction d'activation sigmoidale
16 def actv(a):
17     return 1./(1.+exp(-a))      # Eq. (6.5)
18 -----
19 # Derivee de la fonction d'activation sigmoidale
20 def dactv(s):
21     return s*(1.-s)            # Eq. (6.19)
22 -----
23 # fonction reseau feed-forward, calcul signal de sortie
24 def ffnn(ivec):
25     for ih in range(0,nh):      # couche d'entrée a couche interne
26         sum=0.
27         for ii in range(0,ni):
28             sum+=wih[ii,ih]*ivec[ii]    # Eq. (6.1)
29             sh[ih]=actv(sum)          # Eq. (6.2)
30     for io in range(0,no):      # couche interne a couche de sortie
31         sum=0.
32         for ih in range(0,nh):
33             sum+=who[ih,io]*sh[ih]    # Eq. (6.3) avec b_1=0
34             so[io]=actv(sum)          # Eq. (6.4)
35     return
36 # END fonction ffnn
37 -----
38 # retropropagation du signal d'erreur et ajustement des poids du reseau
39 def backprop(err):
40     for io in range(0,no):      # couche de sortie a couche interne
41         deltao[io]=err[io]* dactv(so[io])      # Eq. (6.20)
42         for ih in range(0,nh):
43             who[ih,io]+=eta*deltao[io]*sh[ih]    # Eq. (6.17) pour les WHO
44     for ih in range(0,nh):      # couche interne a couche de sortie
45         sum=0.
46         for io in range(0,no):
47             sum+=deltao[io]*who[ih,io]
48             deltah[ih]=dactv(sh[ih])*sum          # Eq. (6.21)
49         for ii in range(0,ni):
50             wih[ii,ih]+=eta*deltah[ih]*ivec[ii] # Eq. (6.17) pour les WIH
51     return
52 # END fonction backprop

```

Figure 6.4: Code Python minimal pour l'algorithme de rétropropagation (voir texte).

```

1 ...
2 # Le code de la Figure 6.4 doit preceder ce qui suit
3 #-----
4 # fonction de melange
5 def randomize(n):
6     dumvec=np.zeros(n)
7     for k in range(0,n):
8         dumvec[k]=np.random.uniform() # tableau de nombre aleatoires
9     return np.argsort(dumvec)      # retourne le tableau de rang
10 # END fonction randomize
11 #=====
12 # MAIN: Entrainement d'un reseau par retropropagation
13 nset =100           # nombre de membres dans ensemble d'entrainement
14 niter =1000          # nombre d'iterations d'entrainement
15 oset =np.zeros([nset,no]) # sortie pour l'ensemble d'entrainement
16 tset =np.zeros([nset,ni]) # vecteurs-entree l'ensemble d'entrainement
17 rmserr=np.zeros(niter)   # erreur rms d'entrainement
18
19 # lecture/initialisation de l'ensemble d'entrainement
20 ...                  # diverses instructions...
21 # initialisation aleatoire des poids
22 for ii in range(0,ni):      # poids entree-interne
23     for ih in range(0,nh):
24         wih[ii,ih]=np.random.uniform(-0.5,0.5)
25 for ih in range(0,nh):      # poids interne-sortie
26     for io in range(0,no):
27         wih[ih,io]=np.random.uniform(-0.5,0.5)
28
29 for iter in range(0,niter):    # boucle sur les iteration d'entrainement
30     sum=0.
31     rvec=randomize(nset)        # melange des membres
32
33     for itrain in range(0,nset): # boucle sur l'ensemble d'entrainement
34         itt=rvec[itrain]          # le membre choisi...
35         ivec=tset[itt,:]
36         ffnn(ivec)              # calcule signal de sortie
37         for io in range(0,no):   # signaux d'erreur sur neurones de sortie
38             err[io]=oset[itt,io]-so[io]
39             sum+=err[io]**2       # cumul pour calcul de l'erreur rms
40         backprop(err)          # retropropagation
41 # END boucle sur ensemble
42
43     rmserr[iter]=np.sqrt(sum/nset/no) # erreur rms a cette iteration
44 # END boucle sur iterations d'entrainement
45
46 # Maintenant la phase de test irait ci-dessous...
47 # END MAIN

```

Figure 6.5: Fragment de code Python minimal pour l'entraînement d'un réseau feed-forward par rétropropagation. Ce fragment de code devrait suivre celui de la Figure 6.4, et il lui manque les instructions lisant d'un fichier (ou calculant) les membres de l'ensemble d'entraînement. Les variables et tableaux `no`, `so`, `err`, etc., sont définis au début du code de la Fig. 6.4, et donc sont accessibles globalement.

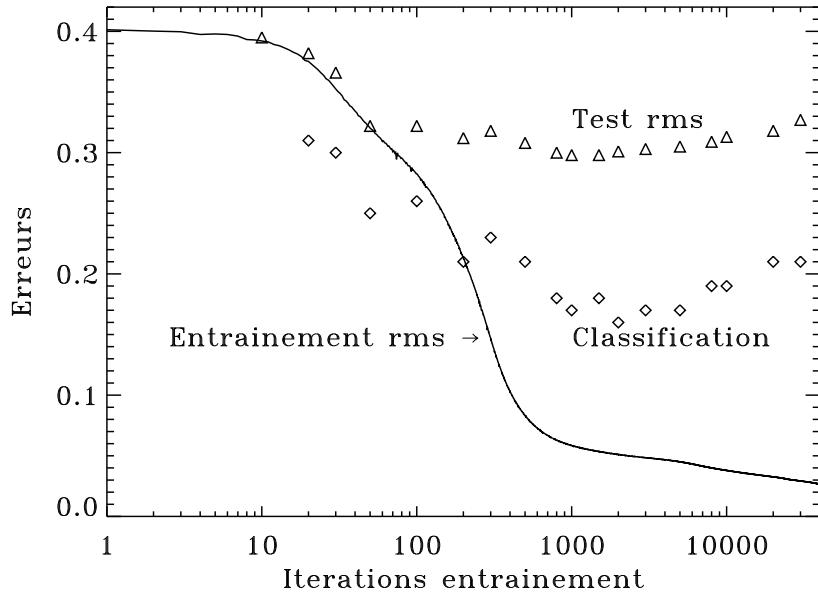


Figure 6.6: Évolution des erreurs r.m.s. à l’entraînement (trait noir) et durant la phase de test (\triangle) en fonction du nombre d’itérations dans la phase d’entraînement, pour le problème de classification décrit à la §6.2.1. L’erreur de classification (\diamond) est calculée selon l’éq. (6.25) ci-dessous. Le réseau utilisé ici a une couche de 6 neurones internes, et aucunes unités de biais. Le paramètre d’apprentissage est $\eta = 0.1$, et les ensembles d’entraînement et de test comptent chacun 100 membres.

$$\langle E \rangle = \left(\frac{1}{M} \sum_{m=1}^M (D^m - S_1^m)^2 \right)^{1/2}. \quad (6.22)$$

Dans une situation où la couche de sortie compte plus d’un neurone, on aurait plutôt:

$$\langle E \rangle = \left(\frac{1}{M} \sum_{m=1}^M \sum_{k=1}^{\text{no}} (D_k^m - S_k^m)^2 \right)^{1/2} \quad (6.23)$$

L’erreur décroît à mesure que l’entraînement progresse (heureusement!), mais cette décroissance est loin d’être régulière (remarquez bien l’échelle horizontale logarithmique sur la Fig. 6.6 !).

Après environ 10^3 itérations d’entraînement ici, le réseau a appris à “reconnaître” les membres de l’ensemble d’entraînement, dans le sens que l’écart rms entre les valeurs de sortie et les valeur désirées a chuté sous ~ 0.05 ; autrement dit, là où le réseau doit prédire un “0.1” il prédit en moyenne dans l’intervalle 0.1 ± 0.05 , et là où il doit prédire un “0.9” il prédit en moyenne dans l’intervalle 0.9 ± 0.05 ; c’est pas mal du tout, mais l’entraînement ne peut être déclaré un succès que si le réseau a appris à généraliser, dans le sens de bien prédire les valeurs de sortie pour des vecteurs d’entrée qui ne font pas partie de l’ensemble d’entraînement. La *phase de test* vise à vérifier si c’est bien le cas.

6.2.4 La phase de test

Considérons maintenant ce qui se passe si on présente au réseau, maintenant entraîné, un nouvel ensemble de $L = 100$ exemples distinct de l’ensemble utilisé pour l’entraînement. Encore une fois les signaux de sorties pour ces nouveaux exemples sont considérés connus, et donc il

est encore possible de définir une erreur rms selon l'éq. (6.22). Les triangles sur la Fig. 6.6 montrent l'erreur rms moyenne sur cet ensemble test, calculé pour des nombre croissant d'itération d'entraînement sur le même ensemble d'entraînement que précédemment; la distance verticale entre les triangle et la courbe en trait plein donne donc une idée du niveau de généralisation atteint par le réseau. Les deux erreurs rms se suivent de près jusqu'à une centaine d'itérations environ, mais ensuite l'erreur sur l'ensemble-test décroît beaucoup plus lentement jusqu'à ce que, à partir de $\simeq 10^3$ itérations d'entraînement, elle se remette à augmenter lentement, et ce même si l'erreur rms sur l'ensemble d'entraînement, elle, continue à diminuer.

Cette situation correspond à un *sur-entraînement* du réseau; ce dernier ne généralise plus, mais "apprend" plutôt des spécificités de l'ensemble d'entraînement, afin de réduire l'erreur d'entraînement, au dépend de sa capacité à généraliser. Comme ces spécificités ne sont typiquement pas présentes dans l'ensemble test, l'erreur rms sur ce dernier augmente. Ce genre de situation est tout à fait typique de l'entraînement des réseaux de neurones artificiels (et, en fait, de l'apprentissage-machine en général)¹. Donc, ici il serait approprié de stopper l'entraînement après $\simeq 10^3$ itérations, même si l'erreur d'entraînement diminue toujours.

La validation du réseau sur un ensemble-test après l'entraînement est donc *essentielle* pour s'assurer d'éviter le surentrainement.

Évidemment, dans le cadre d'un problème de classification binaire comme celui considéré ici, un réseau qui produit un signal de sortie $S_1^O = 0.7$ erre par 0.2, mais du point de vue de la classification il pourrait être justifié de déclarer que le réseau indique que OUI, il y a un bloc d'au moins 5 bits non-nuls contigus dans la chaîne. Si c'est le cas, du point de vue de la classification binaire ceci compte comme une classification correcte. On pourrait donc définir une mesure d'erreur binaire alternative où le signal de sortie est remplacé par

$$S^* = \begin{cases} 1 & \text{si } S_1^O > 0.5 \\ 0 & \text{si } S_1^O < 0.5 \end{cases} \quad (6.24)$$

et baser le calcul de l'erreur sur les S^* . L'erreur moyenne de classification est alors définie comme

$$\langle E_C \rangle = \frac{1}{L} \sum_{l=1}^L |S_*^l - D^l| \quad (6.25)$$

où la somme est effectuée sur les L membres de l'ensemble-test. Les losanges sur la Figure 6.6 montrent comment cette erreur varie en fonction du nombre d'itérations à l'entraînement. La tendance générale est la même que l'erreur rms, indiquant que notre mesure alternative de l'erreur n'est pas déraisonnable. Si on stoppe l'entraînement à 10^3 itérations, l'erreur de classification est d'environ 17%. On en conclut que 83% des membres de l'ensemble-test ont été correctement catégorisés, une "note" très honorable finalement...

Mais holà, réfléchissons un peu; l'ensemble-test, comme l'ensemble d'entraînement, contient 50 exemples de chacune des deux catégories; donc, en tirant à pile ou face, on devrait (statistiquement) déjà atteindre un taux de succès de 50% pour une classification binaire entièrement aléatoire. 83% c'est certainement mieux, mais peut-être pas aussi bon qu'on aurait pu l'imaginer à prime abord.

La Figure 6.7 montre, sous forme d'histogrammes, les distributions des signaux de sortie produits par les membres de l'ensemble-test, au préalable subdivisés en fonction des deux classes de sortie (0 ou 1, indiquant absence ou présence d'au moins 5 bits non-nuls contigus dans la chaîne présentée au réseau). Les deux distributions piquent aux valeurs de sortie très basses (classe 0) et très élevées (classe 1), comme on pouvait l'espérer, mais les deux distributions montrent une "queue" s'étirant vers les valeurs de sortie médianes. On constate ici (graphique

¹...et probablement même de l'apprentissage en général; mémoriser à la perfection les solutions de 50 problèmes de TP, est-ce-que ça vous aide toujours sur un examen ? ou c'est préférable d'avoir saisi la structure logique et physique sous-jacente aux problèmes vus en TP (i.e., avoir "compris" la matière) ? J'imagine que la réponse dépend du cours... et du prof...

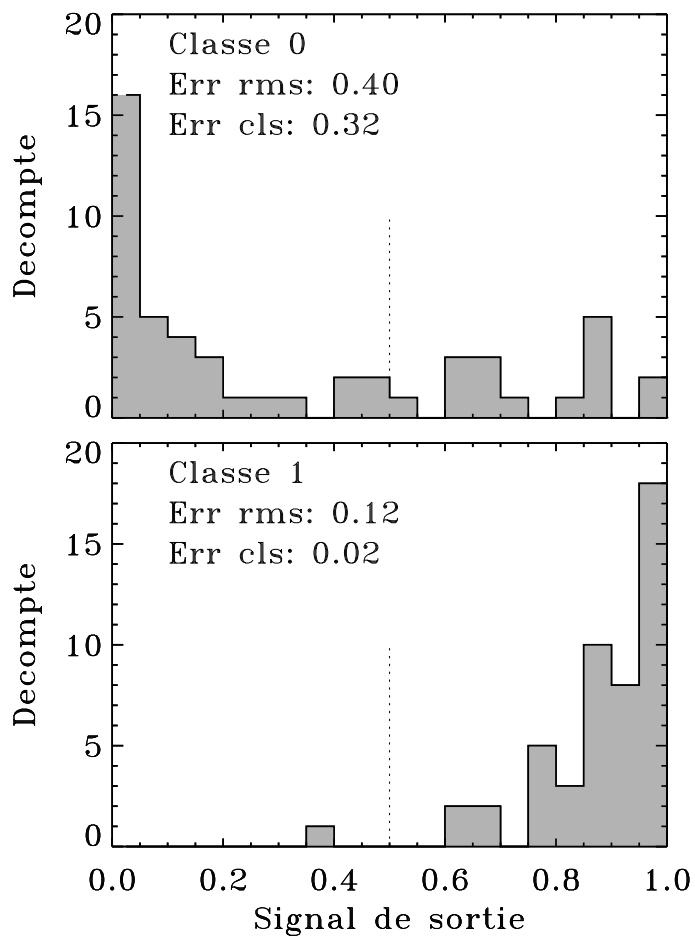


Figure 6.7: Distributions des signaux de sortie produits par l’ensemble-test, subdivisés entre les deux classes-cible ($0 \equiv$ pas de groupes d’au moins 5 bits non-nuls contigus; $1 \equiv$ au moins un groupe d’au moins 5 bits non-nuls contigus). Le trait pointillé indique la frontière de classification (viz. eq. (6.24)). Le réseau performe beaucoup mieux lorsque présentés avec les entrées de classe 1, et ce même si les deux classes sont présentes en nombres égaux dans les ensembles d’entraînement et de test.

du bas) que la performance du réseau est néanmoins excellente pour les membres de l'ensemble-test appartenant à la classe 1: erreur rms de seulement 0.12, et un seul exemple sur 50 mal classé. Malgré une distribution bien piquée près de zéro, les exemples-test de classe 0 (graphique du haut) sont clairement ceux qui dominent l'erreur totale sur la Fig. 6.6: erreur rms de 0.4 et un tiers des exemples mal classés. Le réseau a même réussi à produire dans deux cas un signal de sortie $> 0.9!$ Notre réseau a clairement plus de facilité à détecter la présence d'un groupe de 5 bits non-nuls contigus que son absence, et ce malgré le fait que les ensembles d'entraînement et de test contiennent tous les deux des nombres égaux d'exemples de classe 0 et 1. Ce genre "d'asymétrie" dans la performance de classification n'est pas du tout rare dans ce genre de problème. On peut la quantifier en définissant les quatre catégories suivantes:

1. **Vrai Positif (VP):** La structure (ici, de 5-bit non-nuls contigus) est présente et est correctement classée comme présente.
2. **Vrai Négatif (VN):** La structure est absente et est correctement classée comme absente.
3. **Faux Positif (FP):** La structure est absente mais est incorrectement classée comme présente.
4. **Faux Négatif (FN):** La structure est présente mais est incorrectement classée comme absente.

On représente habituellement ceci de manière compacte sous la forme d'une matrice de dimension 2×2 ; par exemple, pour les résultats de la Fig. 6.7, on aurait:

$$\begin{matrix} 0 & 1 \\ 0 & \left(\begin{matrix} VN & FP \\ FN & VP \end{matrix} \right) \\ 1 & \end{matrix} = \begin{matrix} 0 & 1 \\ 1 & \left(\begin{matrix} 34 & 16 \\ 1 & 49 \end{matrix} \right) \end{matrix} \quad (6.26)$$

Une classification parfaite aura donc les deux éléments hors-diagonale nuls. Cependant, en fonction de l'application considérée, certaines catégories d'erreur peuvent être plus importantes à minimiser; par exemple, dans la prédiction des tremblements de terre, un Faux Positif (fausse alarme) est moins grave qu'un Faux Négatif (tremblement de terre non-prédit; pensez à ces pauvres séismologues italiens...!). Dans une telle situation il serait alors judicieux de définir une mesure de performance globale en attribuant un plus grande pondération à la "case" FN qu'à la case FP.

Il est évidemment possible d'améliorer la performance du notre réseau. Avant même d'essayer, la première chose à faire serait d'interchanger les ensembles d'entraînement et de test, répéter les procédures d'entraînement et de test, et recalculer les mesures d'erreurs rms et de classification. Il est en effet possible que le réseau se soit coincé en entraînement dans une configuration suboptimale due à la présence fortuite, dans l'ensemble d'entraînement, d'exemples quasi-contradictoires (vecteurs d'entrée presque identiques, mais classes différentes). Si c'est le cas, augmenter la taille de l'ensemble d'entraînement est habituellement la première chose à essayer, quand c'est possible de le faire.

Une fois convaincu(e) que la performance n'est pas dépendante de manière sensible du détail de l'ensemble d'entraînement (et/ou de l'ensemble-test), la première chose à essayer serait de varier le nombre de neurones dans la couche cachée, autant à la hausse qu'à la baisse. Si un examen des exemples qui sont mal classés indique que l'une ou l'autre des deux catégories domine l'erreur de classification, comme c'est le cas ici, l'ajout d'une unité de biais à la neurone de sortie (comme sur la Fig. 6.1) et/ou aux neurones internes serait également une approche à explorer. Mais il est maintenant plus que temps de plutôt passer au contexte physique qui motive tout ceci: rien de moins que la recherche du boson de Higgs, Mesdames-zé-Messieurs...

6.3 Le boson de Higgs en une page

Le bien nommé *modèle standard* de la physique des particules est sans nul doute un des grands succès de la physique de la seconde moitié du vingtième siècle. Combinant 6 quarks et 6 leptons

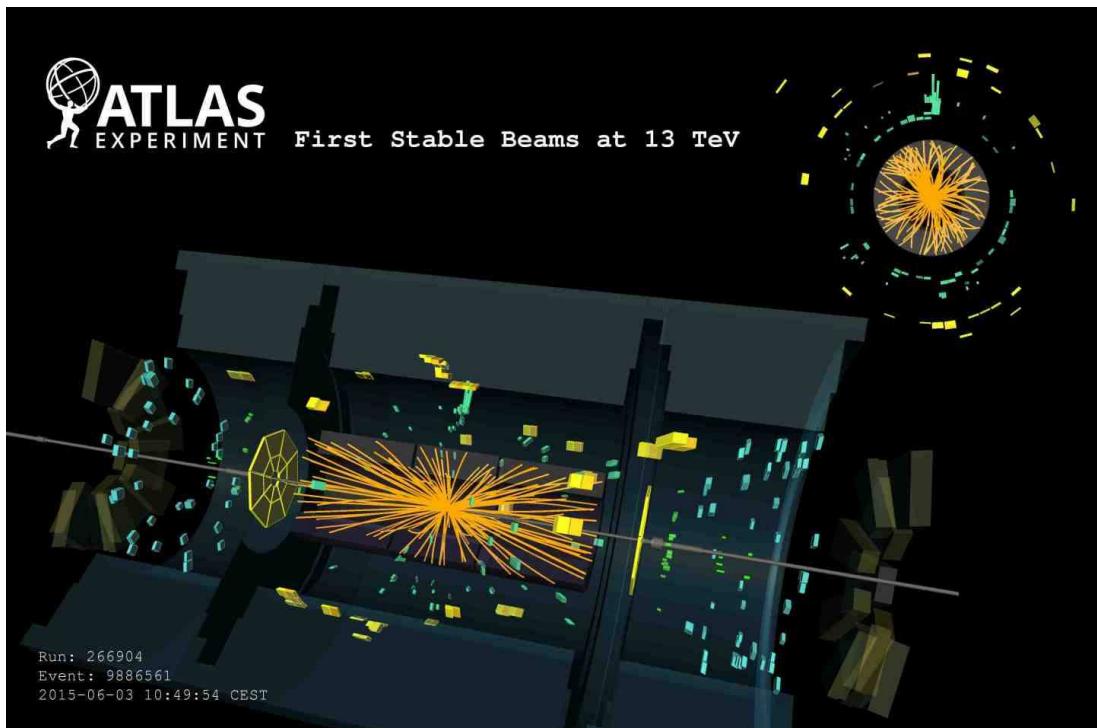


Figure 6.8: Exemple d'une reconstruction des trajectoires des produits secondaires de désintégration d'une collision $p+p$ à $\simeq 13$ TeV dans le détecteur ATLAS au CERN. Les différents volumes colorés indiquent des éléments du détecteur ayant été excités suite à la collision. Les trajectoires en orange sont ensuite reconstruites à partir de ces mesures. L'encart en haut à gauche montre la même collision vue le long de l'axe du LHC. Les trajectoires courbées sont celles de particules chargées déviées par le très intense champ magnétique à l'intérieur d'ATLAS. Image en domaine public tirée de <http://atlas.web.cern.ch/>

dans le cadre de la théorie quantique des champs (effectivement une unification de la relativité restreinte et de la mécanique quantique), la physique des particules expérimentale n'a pas encore produit de résultat qui soit discordant par rapport aux aspects fondamentaux du modèle. Du point de vue expérimental, le modèle standard semble incontestable.

Ce n'est pas que le modèle soit parfait. Du côté théorique, les symétries du Lagrangien du modèle standard ont la facheuse conséquence qu'il se retrouve impossible pour les particules d'avoir une masse... ce qui est en conflit assez direct et incontestable avec les résultats expérimentaux. Dès les années 1960, les théoriciens des particules ont utilisé le truc habituel, soit imaginer de nouvelles particules/champs pouvant briser cette symétrie du Lagrangien; apparaît ainsi sur la carte le champ de Higgs, et l'existence de particules ayant une masse devient alors "possible" théoriquement. Pas moins de trois bosons massifs sont associés au champ de Higgs, dont l'un est celui qu'on appelle maintenant boson de Higgs. Sa masse ne peut être qu'estimée par le modèle standard (quelquepart entre 100 et 250 GeV), mais si la masse est connue ses sections efficaces de production et désintégration se calculent (relativement) facilement. Le boson de Higgs même n'est pas observable directement, ce sont ses produits de désintégration qui révèlent sa (très très brève) présence. La furieuse envie de le détecter a été un des grands points de motivation pour la construction du LHC au CERN, et en particulier du détecteur ATLAS (voir Fig. 6.8).

Considérant l'énergie des collisions atteintes au LHC du CERN, on s'attendait à ce que quelques 10^5 bosons de Higgs soient produits durant la première année d'opération (2011-2012). Ça peut paraître beaucoup, mais le problème c'est qu'à ces énergies il y a tout un zoo de

particules massives autres que le Higgs, incluant d'autres bosons massifs comme le W , qui peuvent aussi être produits, et générer des cascades de désintégration difficile à distinguer de celles du Higgs. On s'attendait en fait à quelque chose comme $\simeq 10^{16}$ collisions énergétiques n'impliquant *pas* le Higgs durant la même période d'opération du LHC. Donc, parmi $\simeq 10^{16}$ cascades de désintégration observées, impliquant des groupes très semblables de particules secondaires, il faut identifier celles provenant de la production/désintégration du Higgs. Voici tout un problème de classification de données expérimentales!

Des divers canaux de désintégrations possibles du Higgs, deux sont relativement "propres", dans le sens qu'ils laissent les signatures les moins ambiguës, dans le sens d'être moins difficile à distinguer des produits secondaires de collisions n'impliquant pas le Higgs: ce sont les deux canaux:

$$H \rightarrow \gamma + \gamma \quad [0.2\%] \quad (6.27)$$

$$H \rightarrow Z + Z^* \rightarrow (\mu^- + \mu^+) + (e^- + e^+) \quad [0.014\%] \quad (6.28)$$

Le prix à payer —il y en a toujours un!— est la faible probabilité de branchement de ces canaux; des $\sim 10^5$ bosons de Higgs attendus en une année d'opération du LHC seuls 200 se désintégreront selon le premier de ces canaux, et 14 suivant le second. Ce sont néanmoins ces deux canaux qui ont produit en 2012 les premières détections les plus statistiquement crédibles du boson de Higgs.

6.4 Projet: trouver le boson de Higgs

Le projet qui clot ce chapitre est une version simplifiée du concours Kaggle lancé il y a maintenant près de quatre ans par l'équipe du CERN: le "Higgs boson machine learning challenge". L'idée était d'améliorer les méthodes de détection et d'analyse de données pour un autre canal de désintégration du boson de Higgs, en deux leptons τ :

$$H \rightarrow \tau^- + \tau^+ \quad (6.29)$$

C'est un canal dominant sous ~ 125 GeV, et il demeure important ($\gtrsim 1\%$) jusqu'à ~ 160 GeV. Le problème est que le boson Z , également produit en abondance par les collisions $p+p$ au LHC sans passer par la production intermédiaire du H , se désintègre aussi en $\tau^- + \tau^+$; et la masse du Z , 91 GeV, est à peine plus petite que la masse attendue du H . Pour compliquer encore plus la chose, tous ces leptons τ ont des temps de vie très courts, et donc ne sont pas mesurés directement par ATLAS, mais indirectement via leurs propres produits de désintégration. Le défi est donc d'identifier la présence du Higgs comme particule primaires sur la bases des caractéristiques des trajectoires des produits de désintégration observés par ATLAS.

Vous travaillerez à partir de 13 mesures dites dérivées des trajectoires de particules (l'annexe B du document cité en fin de chapitre liste en détail, mais en fait la nature exacte de ces mesures n'est pas essentielle à votre travail de classification).

L'idée de ce projet est la simplicité même: je vous fournis sur la page web du cours un ensemble de 2×10^3 exemples de mesures déjà classées, dans un fichier nommé `trainingset.txt`. Ce fichier contient une ligne d'en-tête, ensuite 2000 lignes de 14 chiffres, chacune correspondant à un événement-exemple. Les 13 premiers chiffres définissent le vecteur d'entrées (donc `ni=13`), et le quatorzième est un bit (0 ou 1) établissant la classification binaire de l'exemple: "0" indique un événement classé "pas Higgs" et "1" classé "Higgs". Vous définissez vos ensembles d'entraînement et de test à partir de ça. Vous construisez ensuite un réseau feed-forward, vous l'entraînez, et une fois convaincu que votre réseau fonctionne bien vous lui présentez une seconde série de 1000 mesures-événements, que je vous fournis aussi (fichier `examset.txt`, celle-là non-classées).

Vous devez m'envoyer par courriel (paulchar@astro.umontreal.ca) un fichier ASCII (`.txt`) contenant une séquence de 1000 0 ou 1, *dans le même ordre de présentation que dans le fichier examset.txt*. **JE N'ACCEPTERAI PAS** de fichiers en format word, excel, rtf, ou utilisant

tout autre forme d'encodage! Je calculerai (et vous ferai parvenir par retour de courriel) votre score de classification, qui comptera pour la moitié de la note sur votre rapport de projet. Votre rapport même peut être bref; il doit contenir une description de votre réseau, et une présentation de vos statistiques d'erreur dans les phases d'entraînement et de classement, afin d'appuyer la crédibilité de vos classifications. Quelques suggestions de départ:

1. Commencez avec une seule couche interne de 8 neurones. Augmentez ou diminuez au besoin.
2. Commencez sans biais sur les neurones. Ajoutez un biais au besoin.
3. Les 13 entrées définissant un événement ont des grandeurs numériques passablement différentes. Il pourrait être utile de les ramener à un intervalle $[0, 1]$ (par exemple) en les renormalisant par rapport aux valeurs extrêmes présentes dans l'ensemble.
4. Commencez avec un ensemble d'entraînement relativement petit, genre 100-200 membres. Augmentez seulement au besoin.
5. Commencez avec un ensemble-test de la même taille que votre ensemble d'entraînement; augmentez pour affiner vos statistiques finales.

Bonne chance (même si ce n'est pas vraiment une question de chance...)

6.5 Bibliographie

Il existe une littérature gigantesque sur les réseaux de neurones artificiels; quand j'ai appris le sujet (de manière essentiellement autodidacte), j'ai tiré grand bénéfice des deux ouvrages suivants, dans lesquels je me suis d'ailleurs joyeusement (et nostalgiquement, snif) replongé dans le cadre de la préparation de ce chapitre:

Anderson, J.A., *An Introduction to Neural Networks*, MIT Press (1995).

Haykin, S., *Neural Networks: a comprehensive Foundation*, MacMillan Publishing (1994).

La dérivation de l'algorithme de rétropropagation présentée à la §6.2.2 suit d'ailleurs de près celle faite dans l'ouvrage de Haykin.

Le rôle et l'importance du boson de Higgs "...est un des sujets les plus difficiles du modèle standard...", citant ici verbatim mon collègue Jean-François Arguin à qui j'avais posé la question. La brève discussion de la §6.3 s'est inspirée d'une conférence du vendredi donnée par Jean-François il y a quelques années, dont les slides demeurent disponible sur le Web (mars 2016):

www.lps.umontreal.ca/~arguin/udem_withcomments.pdf

Plus spécifiquement sur les données utilisées dans le cadre du projet (et de la compétition Kaggle), voir le document:

https://higgsml.lal.in2pr.fr/files/2014/04/documentation_v1.8.pdf