Master 2 Modèles non-linéaires

Mémoire de fin d'étude / Rapport de stage

Micromagnétisme hiérarchique pour la modélisation des propriétés des matériaux

Auteur :
Romain Claveau

Maître de stage :
Pascal Thibaudeau

1er Avril 2019 — 28 Juin 2019

Remerciements

Je tiens tout d'abord à remercier mon maître de stage, Pascal, pour les nombreuses discussions, à la fois sur le sujet du stage mais aussi sur d'autres aspects physiques, que nous avons eues. Sa clarté et sa grande connaissance dans le domaine m'ont permis de me plonger dans le sujet sans réelles difficultés.

Je remercie aussi l'ensemble de l'équipe du laboratoire qui a su me mettre très rapidement à l'aise par un accueil assez chaleureux. Les moments de détente en leur compagnie furent des plus plaisants.

Je remercie aussi les deux doctorants du laboratoire, Geoffroy et Thomas, pour les nombreuses discussions, physiques ou non, que nous avons eues autour d'un café.

Finalement, je remercie le centre CEA Le Ripault pour m'avoir accueilli tout au long de ces trois mois de stage.

Table des matières

Table des figures

Chapitre 1

Introduction

Ce mémoire porte sur la dynamique de l'aimantation au sein des matériaux. Depuis de nombreuses années, l'équation de Landau-Lifshitz-Gilbert est utilisée afin de décrire la dynamique de l'aimantation d'un matériau lorsque celui-ci est en contact avec un bain thermique. Ce contact serait traduit par un phénomène de perte d'énergie, et par un terme phénoménologique ajouté par Gilbert à l'équation de la précession.

Cependant, cette équation possède une propriété qui vient contredire cette vocation : nous pouvons rapidement constater que $\vec{m} \cdot d\vec{m}/dt = 0$, entraînant la conservation de la norme de l'aimantation au cours du temps. Cette propriété est associée à la conservation de l'énergie au sein du système, ce qui est impossible si une perte est effective. Le formalisme développé par Yoichiro Nambu en 1972 [1], généralisant la mécanique hamiltonienne, permet de montrer que ce terme ne traduit pas une perte mais plutôt un décalage de l'énergie dans l'espace des phases.

Partant de ce constat, il convient d'être pragmatique : il nous faut trouver une nouvelle hiérarchie d'équations permettant de modéliser convenablement un système d'aimantation en interaction avec des fluctuations thermiques. En utilisant une variable gaussienne aléatoire $\tilde{\omega}_i$ afin de modéliser les fluctuations thermiques et en considérant la moyenne sur le bruit, une nouvelle hiérarchie d'équations a été découverte : la dynamique de Landau-Lifshitz-Bloch [2]. Ces équations traduisent la non-conservation de la norme de l'aimantation, et par extension, la non-conservation de l'énergie au sein du système à cause de l'interaction avec un bain thermique.

Notre travail consiste dans un premier à réutiliser un code micromagnétique développé avec Python (et NumPy) par Claas Abert [3] et de l'améliorer en changeant l'intégrateur (Méthode d'Euler explicite à méthode de Runge-Kutta explicite d'ordre 4). Ceci fait, il conviendra de vérifier que les dynamiques générées sont équivalentes.

Puis, nous nous intéresserons aux équations de Landau-Lifshitz-Bloch et développerons un programme permettant de résoudre numériquement ces 21 équations simultanément, en considérant un champ interne nul (les interactions micromagnétiques seront négligées) et un champ externe (magnétique) constant. Nous comparerons ensuite la dynamique obtenue avec celle calculée par le logiciel de calcul formel Maple, afin de tester la validité de notre intégrateur numérique (qui sera aussi une méthode de Runge-Kutta explicite d'ordre 4).

Enfin, ces deux étapes passées, nous pourrons nous intéresser à la dynamique de Landau-Lifshitz-Bloch en considérant l'ensemble des interactions, internes et externe, afin de décrire la dynamique de l'aimantation à température finie.

Finalement, nous utiliserons une approche atomistique afin de décrire la dynamique de structures topologiques stables, baptisées *skyrmions*, en température finie.

Chapitre 2

Approche analytique

2.1 Micromagnétisme

Le micromagnétisme permet de décrire la dynamique de l'aimantation au sein d'un matériau soumis à un champ magnétique extérieur. Il convient de remarquer que même sans ce champ, le matériau est soumis à des interactions internes qui doivent être prises en compte afin de trouver l'état d'équilibre de l'aimantation. Cet état d'équilibre peut être trouvé en minimisant l'énergie E du système, composée de l'énergie d'échange E_{exch}, de l'énergie d'anisotropie E_{ani}, de l'énergie de Zeeman E_{zee} et de l'énergie de désaimantation E_{demag} :

$$E = E_{exch} + E_{anis} + E_{zee} + E_{demag} \tag{2.1}$$

En définissant l'aimantation réduite du système $\vec{m} = \frac{\vec{M}}{|\vec{M}|}$, nous pouvons obtenir le champ effectif ressenti par l'aimantation comme :

$$\vec{H}_{eff} = -\frac{1}{\mu_0 M_s}\frac{\delta^2 E}{\delta \vec{m} \delta V} \tag{2.2}$$

avec M_s l'aimantation à saturation du système, V le volume du système et $\mu_0 = 4\pi \times 10^{-7}$ H/m, la perméabilité du vide.

2.1.1 Développement des termes d'énergie internes et externe

2.1.1.1 Energie d'échange

Afin de trouver la forme analytique de ces différentes énergies, il convient de s'intéresser aux "bases" du micromagnétisme. Dans le cas des matériaux ferromagnétiques, les électrons ayant une fonction d'onde recouvrant d'autres fonctions d'onde électroniques (overlapping) favorisent un alignement parallèle de leur spin grâce à une interaction d'échange. Ainsi, l'alignement des aimants élémentaires $\vec{m}_I$ à la position $\vec{r}_I$ peut être considéré comme parallèle pourvu que la distance entre les deux spins est petite devant une longueur caractéristique notée λ :

$$\vec{m}_I \approx \vec{m}_K \text{ pour } |\vec{r}_I - \vec{r}_K| < \lambda \tag{2.3}$$

Bien que cet effet soit purement quantique, le micromagnétisme ne peut pas être considéré comme tel car il prend aussi en compte des effets classiques. Plus précisément, le micromagnétisme permet

d'étendre l'électrodynamique classique en y ajoutant des termes d'origine quantique. Ainsi, l'expression de départ de l'énergie d'échange provient du Hamiltonien de Heisenberg mettant en jeu le spin $\vec{S}$ et la constante d'interaction entre les sites I et K J_{IK}.

$$E_{I,K} = -J_{IK}\vec{S}_I \cdot \vec{S}_K \tag{2.4}$$

Nous introduisons ensuite le fait que $S = |\vec{S}_I| = |\vec{S}_K|$ et construisons les vecteurs unitaires $\vec{n}_i = \vec{S}_I/S$ et $n_K = \vec{S}_K/S$ associés. Ainsi l'énergie d'échange se réécrit comme :

$$E_{I,K} = -J_{IK}S^2\vec{n}_I \cdot \vec{n}_K \tag{2.5}$$

Le produit scalaire $\vec{n}_I \cdot \vec{n}_K$ peut être évité en introduisant une nouvelle variable $\vec{n} = \vec{n}_I - \vec{n}_K$. Ceci nous permet donc d'écrire $\vec{n}^2 = \vec{n}_I^2 + \vec{n}_K^2 - 2\vec{n}_I \cdot \vec{n}_K$ et donc d'obtenir $\vec{n}_I \cdot \vec{n}_K = \frac{1}{2}(\vec{n}_I^2 + \vec{n}_K^2 - \vec{n}^2)$. En utilisant le fait que $\vec{n}_I$ et $\vec{n}_K$ soient des vecteurs unitaires, nous pouvons finalement obtenir $\vec{n}_I \cdot \vec{n}_K = 1 - \frac{1}{2}(\vec{n}_I - \vec{n}_K)^2$. Ainsi, en considérant que l'énergie total d'échange s'écrit comme la somme sur les plus proches voisins, nous obtenons :

$$E_{I,K}^{tot} = -\sum_{\langle I,K \rangle} J_{IK}S^2 \left(1 - \frac{1}{2}(\vec{n}_I - \vec{n}_K)^2\right) \tag{2.6}$$

Cette expression de l'énergie est discrète et nous aimerions maintenant la généraliser pour une valeur de $\vec{m}$ continue (car cela doit être le cas dans le cadre du micromagnétisme). Nous considérons donc :

$$\vec{m}(\vec{r}) \cdot \vec{m}(\vec{r} + \delta\vec{r}) \approx 1 - \frac{1}{2}\left(\vec{m}(\vec{r}) \cdot \delta\vec{m}(\vec{r})\right)^2 \tag{2.7}$$

Afin d'obtenir l'énergie totale d'interaction d'échange entre les spins, il suffit de sommer sur les plus proches voisins et d'intégrer ensuite sur l'ensemble du volume du matériau, noté Ω :

$$E_{exch}^{tot} = \int_\Omega \sum_I A_I \vec{m}(\vec{r}) \cdot \delta\vec{m}(\vec{r}))d\vec{r} \tag{2.8}$$

En utilisant l'expression 2.7, nous pouvons obtenir :

$$E_{exch}^{tot} = \int_\Omega \sum_I A_I d\vec{r} - \frac{1}{2}\int_\Omega \sum_I A_I \left(\vec{m}(\vec{r}) \cdot \delta\vec{m}(\vec{r})\right)^2 d\vec{r} \tag{2.9}$$

Le premier terme obtenu met en jeu A_I qui est la constante d'échange contenant les magnitudes des différents spins. Celui-ci peut être considéré comme constant, qui peut ensuite être supprimé de l'expression lors d'une référence. Dans le second terme, nous obtenons une matrice A_I qui n'est pas diagonale. Cependant, en utilisant le fait que la matrice est symétrique et qu'une rotation appropriée du système permet sa diagonalisation [4], nous pouvons écrire :

$$E_{exch}^{tot} = -\frac{1}{2}\int_\Omega \sum_I A_I \left(\delta\vec{m}(\vec{r})\right)^2 d\vec{r} \tag{2.10}$$

Lors de l'étude de matériaux isotropiques (aucune direction n'est privilégiée), la constante d'échange ne dépend pas des coordonnées spatiales, nous permettant de la sortir de la somme et de l'intégrale dans l'expression ci-dessus. Ceci nous permet d'écrire :

$$E_{exch}^{tot} = -\frac{A}{2}\int_{\Omega} \left(\delta\vec{m}(\vec{r})\right)^2 d\vec{r} \tag{2.11}$$

Bien que nous soyons partis du Hamiltonien de Heisenberg, postulant des spins localisés, cette expression permet de déterminer l'énergie d'échange de la plupart des matériaux et la constante d'échange A peut être déterminée expérimentalement.

2.1.1.2 Energie de désaimantation

Intéressons-nous maintenant à un autre terme contribuant à l'énergie micromagnétique : l'énergie démagnétisante. Celle-ci provient de l'existence d'un champ induit qui s'oppose à l'aimantation du matériau. Ainsi, nous pouvons constater la création de petits dipôles magnétiques (plus précisément plusieurs zones avec des spins orientés parallèlement les uns aux autres mais qui sont dans le sens inverse que ceux dans les zones alentours) s'annulant mutuellement. Alors que l'énergie d'échange tend à orienter les spins et donc à favoriser la même direction que l'aimantation, l'énergie démagnétisante, comme son nom l'indique, tend à désaimanter le matériau. Afin de comprendre l'existence d'un tel terme, penchons-nous sur les équations de Maxwell. En supposant que nous avons un courant $\vec{J}$ nul, nous pouvons écrire que :

$$\begin{cases} \vec{\nabla}\cdot\vec{B} = 0 \\ \vec{\nabla}\times\vec{H} = \vec{0} \end{cases} \tag{2.12}$$

avec $\vec{B} = \mu_0\left(\vec{H} + \vec{M}\right)$. On pose que le champ vectoriel $\vec{H}$ comme le gradient d'un champ scalaire u tel que $\vec{H} = -\vec{\nabla}u$. En utilisant le fait que $\vec{\nabla}\cdot\vec{\nabla}u \equiv \Delta u$, nous pouvons obtenir :

$$\begin{cases} \vec{H} = -\vec{\nabla}u \\ \Delta u = \vec{\nabla}\cdot\vec{M} \end{cases} \tag{2.13}$$

On considère que le champ u s'annule à l'infini. Asymptotiquement, nous pouvons écrire que $u(\vec{r}) = \mathcal{O}\left(1/|\vec{r}|\right)$. Remarquons ensuite que la première équation de 2.14 n'est rien d'autre que l'équation de Poisson, qui peut donc être résolue en utilisant la fonction de Green du Laplacien :

$$u(\vec{r}) = -\frac{1}{4\pi}\int_{\Omega} \frac{\vec{\nabla}'\cdot\vec{M}(\vec{r'})}{|\vec{r}-\vec{r'}|} d\vec{r'} \tag{2.14}$$

Le vecteur $\vec{M}(\vec{r})$ désigne la forme de l'aimantation au point $\vec{r}$. Dans le cadre d'un corps magnétique parfait, l'aimantation est uniforme au sein du matériau et égale à l'aimantation de saturation M_s, et nulle à l'extérieur de celui-ci. En considérant que Ω désigne le volume du matériau, nous avons :

$$\vec{M}(\vec{r}) = \begin{cases} M_s & \text{si } \vec{r} \in \Omega \\ 0 & \text{sinon} \end{cases} \tag{2.15}$$

Cette condition particulière entraîne une discontinuité à l'interface, notée $\partial\Omega$, comprise entre Ω et $\mathbb{R}^3\backslash\Omega$. Afin de contourner ce problème, nous pouvons considérer un milieu entourant notre matériau dont le volume serait noté Ω_{med} et qui aurait la particularité de faire décroître la courbe de l'aimantation afin de joindre les deux autres milieux. Nous obtenons ainsi :

$$u(\vec{r}) = -\frac{1}{4\pi}\left(\int_{\Omega}\frac{\vec{\nabla}'\cdot\vec{M}(\vec{r'})}{|\vec{r}-\vec{r'}|}d\vec{r'} + \int_{\Omega_{med}}\frac{\vec{\nabla}'\cdot\vec{M}(\vec{r'})}{|\vec{r}-\vec{r'}|}d\vec{r'}\right) \tag{2.16}$$

Afin de pouvoir utiliser le théorème de Green, nous devons transformer un peu l'expression ci-dessus. Il suffit de remarquer que :

$$\int_{\Omega_{med}}\frac{\vec{\nabla}'\cdot\vec{M}(\vec{r'})}{|\vec{r}-\vec{r'}|}d\vec{r'} = \int_{\Omega_{med}}\vec{\nabla}'\cdot\left(\frac{\vec{M}(\vec{r'})}{|\vec{r}-\vec{r'}|}\right)d\vec{r'} - \int_{\Omega_{med}}\vec{M}(\vec{r'})\cdot\vec{\nabla}'\left(\frac{1}{|\vec{r}-\vec{r'}|}\right)d\vec{r'} \tag{2.17}$$

Le théorème de Gauss stipule que :

$$\int_{\Omega_{med}}\vec{\nabla}'\cdot\left(\frac{\vec{M}(\vec{r'})}{|\vec{r}-\vec{r'}|}\right)d\vec{r'} = \int_{\partial\Omega_{med}}\frac{\vec{M}(\vec{r'})\cdot\vec{n}}{|\vec{r}-\vec{r'}|}d\vec{s'} \tag{2.18}$$

Afin de négliger les intégrales sur Ω_{med}, il suffit de considérer que l'aimantation décroît très vite dans cette zone, ainsi la région Ω_{med} peut devenir infiniment petite sans que cela change le résultat de l'équation 2.17. Ainsi, la partie droite du terme de droite de l'équation 2.17 s'annule. Il ne reste plus que l'intégrale, en utilisant l'équation 2.18, sur le bord $\partial\Omega_{med}$, qui n'est qu'un contour fermé. Comme l'aimantation s'annule sur ces bords, l'intégrale s'annule elle aussi. Ainsi, le terme de $u(\vec{r})$ dépendant de $\partial\Omega_{med}$ s'annule naturellement sans pour autant changer la physique considérée. Nous obtenons simplement :

$$u(\vec{r}) = -\frac{1}{4\pi}\left(\int_{\partial\Omega}\frac{\vec{M}(\vec{r'})\cdot\vec{n}}{|\vec{r}-\vec{r'}|}d\vec{s'} - \int_{\Omega}\vec{M}(\vec{r'})\cdot\vec{\nabla}'\left(\frac{1}{|\vec{r}-\vec{r'}|}\right)d\vec{r'}\right) \tag{2.19}$$

Sachant que l'aimantation est parallèle à la surface, dû à l'annulation de la composante normale, nous obtenons $\vec{M}(\vec{r'})\cdot\vec{n} = 0$. Ainsi :

$$u(\vec{r}) = \frac{1}{4\pi}\int_{\Omega}\vec{M}(\vec{r'})\cdot\vec{\nabla}'\left(\frac{1}{|\vec{r}-\vec{r'}|}\right)d\vec{r'} \tag{2.20}$$

Afin d'obtenir le champ démagnétisant $\vec{H}_{dem}(\vec{r})$, il suffit de reprendre l'expression 2.13 et d'injecter 2.20 à l'intérieur. Nous obtenons finalement :

$$\vec{H}_{dem}(\vec{r}) = -\frac{1}{4\pi}\int_{\Omega}\vec{M}(\vec{r'})\cdot\vec{\nabla}\otimes\vec{\nabla}'\left(\frac{1}{|\vec{r}-\vec{r'}|}\right)d\vec{r'} \tag{2.21}$$

On dénote $\bar{\bar{N}}(\vec{r}-\vec{r'}) = -\frac{1}{4\pi}\vec{\nabla}\otimes\vec{\nabla}'\left(\frac{1}{|\vec{r}-\vec{r'}|}\right)$, ce qui nous permet d'obtenir la convolution suivante :

$$\vec{H}_{demag}(\vec{r}) = \int_{\Omega}\bar{\bar{N}}(\vec{r}-\vec{r'})\vec{M}(\vec{r'})d\vec{r'} \tag{2.22}$$

Les lois de l'électrodynamique classique [5] stipulent que :

$$E_{demag} = -\frac{\mu_0}{2}\int\int_{\Omega}\vec{M}(\vec{r})\bar{\bar{N}}(\vec{r}-\vec{r'})\vec{M}(\vec{r'})d\vec{r}d\vec{r'} \tag{2.23}$$

2.1.1.3 Energie d'anisotropie

L'énergie d'anisotropie provient de l'interaction spin-orbite dans les matériaux magnétiques dont la structure cristalline favorise certains axes d'alignement de l'aimantation [6]. Cette énergie est purement locale du fait qu'elle ne dépende exclusivement que de l'aimantation locale. Cette forme d'énergie étant assez complexe à décrire et mettant en jeu la forme du matériau, nous nous limiterons à sa forme dans le cas orthorhombique. L'énergie d'anisotropie s'écrit ainsi :

$$E_{ani} = \sum_{i,k} K_{ik} m_i m_k \tag{2.24}$$

où K_{ik} est un tenseur symétrique d'ordre deux définissant les principaux axes d'aimantation. Ainsi, dans le cas orthorhombique tridimensionnel, K est diagonal, donc :

$$E_{ani} = K_1 m_1^2 + K_2 m_2^2 + K_3 m_3^3 \tag{2.25}$$

2.1.1.4 Energie de Zeeman

L'énergie de Zeeman est créée par un champ magnétique extérieur et a pour expression :

$$E_{zee} = -\mu_0 \int_\Omega \vec{M} \cdot \vec{H}_{zee} d\vec{r} \tag{2.26}$$

avec $\vec{H}_{zee}$ le champ magnétique extérieur, appliqué au matériau.

2.1.2 Dynamique de l'aimantation

L'équation de Landau-Lifshitz peut être obtenue en appliquant le formalisme de Lagrange et en considérant un Lagrangien bien particulier, qui n'est malheureusement pas entièrement justifiable par la théorie classique de variables commutantes [20]. Ainsi, il est intéressant d'utiliser la mécanique quantique afin d'approcher plus rigoureusement ce problème.

2.1.2.1 Equation de Landau-Lifshitz-Gilbert

En mécanique quantique, les opérateur $\hat{S}_i$ permettent la caractérisation des différentes composantes du spin. Il est pertinent de rappeler que nous avons une équivalence entre les crochets de Poisson et le commutateur de deux objets via la relation $\{\cdot\} \equiv \frac{1}{i\hbar}[\cdot]$. Lorsqu'un objet ne dépend pas explicitement du temps, sa dynamique est donnée par le relation :

$$\frac{d}{dt}A = \{A, H\} + \cancelto{= 0}{\frac{\partial A}{\partial t}} \tag{2.27}$$

Ainsi, pour un opérateur $\hat{A}$ ne dépendant pas explicitement du temps, nous avons, dans la représentation de Heisenberg, la relation suivante :

$$\frac{d}{dt}\hat{A} = \frac{1}{i\hbar}[\hat{A}, \hat{H}] + \cancelto{= 0}{\left(\frac{\partial \hat{A}}{\partial t}\right)} \tag{2.28}$$

Ainsi, pour l'opérateur de spin $\hat{S}_j$, nous avons :

$$\frac{d}{dt}\hat{S}_j = \frac{1}{i\hbar}[\hat{S}_j, \hat{H}] \tag{2.29}$$

Sachant que notre Hamiltonien, doit nous donner des contributions faisant apparaître des opérateurs de spin, il est intéressant de le développer en fonction de ceux-ci. Nous pouvons écrire :

$$[\hat{S}_j, \hat{H}] = -\sum_k \frac{\partial H}{\partial S_k}[\hat{S}_k, \hat{S}_j] + \mathcal{O}(\hbar^2) \tag{2.30}$$

Sachant que nous avons $[\hat{S}_k, \hat{S}_j] = -i\hbar\epsilon_{jkl}\hat{S}_l$ avec ϵ_{jkl} le symbole de Levi-Civita, nous avons :

$$\frac{d}{dt}\hat{S}_j = \sum_{k,l} \epsilon_{jkl}\frac{\partial H}{\partial S_k}\hat{S}_l + \mathcal{O}(\hbar^2) \tag{2.31}$$

En considérant maintenant la notation vectorielle $\hat{\vec{S}}$, nous pouvons reconnaître la somme dans 2.31 comme un produit tensoriel :

$$\frac{d}{dt}\hat{\vec{S}} = \frac{\delta\vec{H}}{\delta\vec{S}} \times \hat{\vec{S}} + \mathcal{O}(\hbar^2) \tag{2.32}$$

Le passage à la limite classique s'effectue en prenant $\hbar \to 0$, annulant le résidu en $\mathcal{O}(\hbar)$. Sachant que $|\vec{M}| = \frac{Ng\mu_B}{V}\left|\frac{\vec{L}+\vec{S}}{\hbar}\right|$, et que $\vec{L}$ est négligeable devant $\vec{S}$, nous obtenons une relation de proportionnalité entre $\vec{M}$ et $\vec{S}$. Ainsi $\delta\vec{H}/\delta\vec{S}$ doit être proportionnel au champ effectif $\vec{H}_{eff}$. Cette proportionnalité explique la similarité entre la relation 2.2 et l'équation de la précession :

$$\frac{d\vec{m}}{dt} = \gamma\mu_0\vec{H}_{eff} \times \vec{m} \tag{2.33}$$

Avec $\gamma = g\mu_B/\hbar$ le facteur gyromagnétique. Cependant, ceci n'est pas l'équation de Landau-Lifshitz. En effet, pour l'obtenir, il suffit d'ajouter le terme traduisant des "pertes", l'équation de la précession conservant totalement l'énergie. Ce terme supplémentaire, entièrement phénoménologique, a été introduit par Gilbert. L'équation de Landau-Lifshitz s'écrit ainsi comme [7] :

$$\frac{d\vec{m}}{dt} = -\gamma\mu_0\left(\vec{m} \times \vec{H}_{eff}\right) - \lambda\vec{m} \times \frac{d\vec{m}}{dt} \tag{2.34}$$

Equation, qui après quelques manipulations vectorielles, devient :

$$(1+\lambda^2)\frac{d\vec{m}}{dt} = -\gamma\mu_0\vec{m} \times \vec{H}_{eff} - \gamma\mu_0\lambda\vec{m} \times \left(\vec{m} \times \vec{H}_{eff}\right) \tag{2.35}$$

2.1.2.2 Dissipation d'énergie et systèmes quantiques ouverts

Bien qu'il fût inclus de façon totalement phénoménologique dans l'équation de précession, le terme de dissipation de l'équation de Landau-Lifshitz-Gilbert peut maintenant être expliqué à l'aide de la mécanique quantique. Cependant, nous ne pouvons pas utiliser la mécanique quantique dites "fermée" stipulant que les opérateurs $\hat{N}$ et $\hat{H}$ commutent, entraînant la conservation du nombre de particules dans l'espace considéré. L'idée est de considérer un espace local, ouvert, plongé dans un

espace bien plus grand, fermé, comme l'Univers par exemple. Ainsi, le nombre totale de particules est conservé dans l'Univers mais pas forcément dans l'espace local. La mécanique quantique "ouverte" s'intéresse à l'espace local, qui ne conserve pas le nombre de particules. En utilisant ce principe-là, il est possible de retrouver l'équation de Landau-Lifshitz-Gilbert en faisant certains choix qui sont physiquement justifiables. Introduisons l'opérateur non-hermitien suivant[8] :

$$\hat{\mathcal{H}} = \hat{H} - i\lambda\hat{\Gamma} \tag{2.36}$$

$\hat{H}$ désigne l'Hamiltonien du système quantique fermé, $\hat{\Gamma}$ un opérateur hermitien que l'on considérera quelconque et λ une constante. Il a été remarqué [9] que l'introduction d'un Hamiltonien non-hermitien conduisait à une absorption ou une dissipation d'énergie. A cause de la non-conservation de la norme, nous faisons le changement $\hat{\Gamma} \Rightarrow \hat{\Gamma} - \langle\hat{\Gamma}\rangle$. En effet :

$$n = \langle\psi(t)|\psi(t)\rangle = e^{2\lambda\langle\hat{\Gamma}\rangle t}\langle\psi_0|e^{-2\lambda\langle\Gamma\rangle t}|\psi_0\rangle = e^{2\lambda\langle\hat{\Gamma}\rangle t}e^{-2\lambda\langle\hat{\Gamma}\rangle t} = 1 \tag{2.37}$$

En considérant que $|\psi(t)\rangle = e^{-i\mathcal{H}t}|\psi_0\rangle$. Nous pouvons ainsi écrire l'équation de Schrödinger correspondante :

$$i\hbar\frac{\partial|\psi(t)\rangle}{\partial t} = \left(\hat{H} - i\lambda\left(\hat{\Gamma} - \langle\hat{\Gamma}\rangle\right)\right)|\psi(t)\rangle \tag{2.38}$$

Nous définissons maintenant l'opérateur densité comme $\hat{\rho} = |\psi(t)\rangle\langle\psi(t)|$. Le théorème de Liouville stipule que :

$$\frac{\partial}{\partial t}\hat{\rho} = \frac{i}{\hbar}[\hat{\rho}, \hat{\mathcal{H}}] \tag{2.39}$$

Nous faisons l'approximation que $\hat{\Gamma} = \hat{H}$ [10], qui est en partie justifiable physiquement. En effet, en considérant l'Univers comme un espace fermé $\mathcal{E}_{tot}$ et notre espace local $\mathcal{E}_{sub}$ comme un sous-espace ouvert en interaction avec $\mathcal{E}_{tot}$ qui est vu comme un bain thermique. Ainsi, en considérant que $\hat{\mathcal{H}} = \hat{H} - i\lambda(\hat{H} - \langle\hat{H}\rangle)$, nous considérons que la variable λ, traduisant l'amortissement du système, contient toutes les informations de l'interaction de $\mathcal{E}_{tot}$ avec $\mathcal{E}_{sub}$. La dynamique de cette interaction peut être obtenue formellement en utilisant la projection de Zwanzig-Mori [11][12]. Nous pouvons définir deux sous-espaces de fonctions Ω_{slow} et Ω_{fast} contenant respectivement des variables dites "lentes" $A(\Omega)$ (composantes de Fourier avec de grandes longueurs d'ondes) et des variables dites "rapides" (composantes de Fourier avec de petites longueurs d'ondes). Par définition, les variables rapides sont orthogonales à toute fonction $G(A(\Omega))$ dépendante de $A(\Omega)$. Ainsi, toute fonction de l'espace Ω peut se décomposer comme $f(\Omega) = G(A(\Omega)) + R(\Omega)$. Le projecteur de Zwanzig-Mori doit respecter la condition suivante :

$$\mathcal{P}\left[G(A(\Omega))\right] = G(A(\Omega)) \text{ avec } \mathcal{P}^2 = \mathcal{P} \tag{2.40}$$

Ainsi, nous obtenons $R(\Omega) = (1 - \mathcal{P})f(\Omega)$. Prenons l'équation de Liouville :

$$\frac{\partial}{\partial t}\sigma(\Omega, t) = \hat{L}\sigma(\Omega, t) \tag{2.41}$$

Avec $\rho(\Omega,t)=\rho_0(\Omega)\sigma(\Omega,t)$ et $\hat{L}=\frac{i}{\hbar}[\sigma,\mathcal{H}]$. L'idée est de décomposer cette équation sur les sous-espaces Ω_{slow} et Ω_{fast}. Pour cela, nous introduisons $\rho_1=\mathcal{P}\sigma$ et $\rho_2=(1-\mathcal{P})\sigma$ afin de pouvoir projeter l'équation 2.41. Nous obtenons ainsi :

$$\begin{cases}\frac{\partial}{\partial t}\rho_1 &= \mathcal{P}\hat{L}\left(\rho_1+\rho_2\right)\\ \frac{\partial}{\partial t}\rho_2 &= \left(1-\mathcal{P}\right)\hat{L}\left(\rho_1+\rho_2\right)\end{cases} \tag{2.42}$$

Nous voulons séparer la dynamique sur deux sous-espaces afin de caractériser plus facilement les effets collectifs dans Ω_{slow} et mettre de côté les fluctuations rapides dans Ω_{fast}. Afin de résoudre ce système, nous pouvons tout d'abord résoudre formellement la deuxième équation en considérant la forme suivante :

$$\left(1-\mathcal{P}\right)\sigma=e^{(1-\mathcal{P})\hat{L}t}\left(1-\mathcal{P}\right)\sigma(t=0)+\int_0^t dt' e^{(1-\mathcal{P})\hat{L}t'}\left(1-\mathcal{P}\right)\hat{L}\mathcal{P}\sigma(t-t') \tag{2.43}$$

En injectant cette forme dans la première équation, et en supposant que le terme non-homogène s'annule, nous obtenons :

$$\frac{\partial}{\partial t}\mathcal{P}\sigma=\mathcal{P}\hat{L}\mathcal{P}\sigma+\mathcal{P}\hat{L}\int_0^t dt' e^{(1-\mathcal{P})\hat{L}t'}\left(1-\mathcal{P}\right)\hat{L}\mathcal{P}\sigma(t-t') \tag{2.44}$$

On dénote $\mathcal{K}(t)=\mathcal{P}\hat{L}e^{(1-\mathcal{P})\hat{L}t'}\left(1-\mathcal{P}\right)\hat{L}$, ce qui nous permet d'écrire l'équation de Nakajima–Zwanzig :

$$\frac{\partial}{\partial t}\rho_1=\mathcal{P}\hat{L}\rho_1+\int_0^t dt'\mathcal{K}(t')\rho_1(t-t') \tag{2.45}$$

En remplaçant $\hat{L}$ par le commutateur de σ et $\hat{\mathcal{H}}$, nous remarquons que l'équation de la densité se présente comme l'équation 2.38, c'est-à-dire avec un opérateur global $\hat{\mathcal{O}}\rho_1=\mathcal{P}\hat{L}\rho_1+\int_0^t dt'\mathcal{K}(t')\rho_1(t-t')$, non-hermitien. Ainsi, l'argument avancé par Wieser est justifiable physiquement mais le fait de prendre $\hat{\Gamma}=\hat{H}$ est une approximation (en réalité, il faudrait utiliser plus rigoureusement le formalisme de Zwanzig afin de déterminer la forme de l'opérateur $\hat{\Gamma}=\hat{\mathcal{O}}$). Ainsi, en faisant l'approximation $\hat{\Gamma}=\hat{H}$, nous considérons que la variable λ contient toute l'information sur l'interaction entre le sous-espace ouvert $\mathcal{E}_{sub}$ et l'espace fermé $\mathcal{E}_{tot}$, l'équation 2.38 se réécrit comme :

$$i\hbar\frac{\partial|\psi(t)\rangle}{\partial t}\approx\left(\hat{H}-i\lambda\left(\hat{H}-\langle\hat{H}\rangle\right)\right)|\psi(t)\rangle \tag{2.46}$$

L'équation de Liouville associée à cette équation de Schrödinger peut s'écrire comme :

$$\frac{\partial}{\partial t}\hat{\rho}=\frac{i}{\hbar}[\hat{\rho},\hat{H}]-\frac{\lambda}{\hbar}[\hat{\rho},[\hat{\rho},\hat{H}]] \tag{2.47}$$

Comme vu dans l'équation 2.30, l'Hamiltonien peut être décomposé en fonctions des opérateurs de spin. Ceux-ci vont faire apparaître des symboles de Levi-Civita, permettant la construction de produits vectoriels. Nous obtenons ainsi un analogue directe entre l'équation de Schrödinger 2.47 sous la forme de Liouville et l'équation de Landau-Lifshitz-Gilbert 2.35. Nous avons donc réussi à montrer que le terme phénoménologique de Gilbert, peut être retrouvé en se plaçant dans un système quantique ouvert et en utilisant le formalisme de projection de Zwanzig.

2.2 Aimantation en présence de fluctuations thermiques

Les précédentes équations obtenues sont valables seulement dans le cas où notre système serait isolé et n'interagirait pas avec l'extérieur, et ce même si nous avons rajouté un terme d'amortissement. Afin de considérer plus rigoureusement l'interaction d'un système avec l'extérieur, il est intéressant de se tourner vers le formalisme stochastique, permettant l'introduction du formalisme statistique et de variables aléatoires, représentant les fluctuations d'un bain thermique.

2.2.1 Généralités

Le formalisme stochastique permet de considérer des variables aléatoires, permettant de décrire des fluctuations dans un système physique, suivant une loi de probabilité. Afin de décrire l'interaction entre un système et un bain thermique, il est commun d'utiliser ce formalisme, et de considérer que désormais ce sont les valeurs moyennes des variables qui sont observables, et intéressantes. L'introduction de variables moyennes entraîne l'utilisation de fonction de corrélation de différentes variables, ayant elles-mêmes leur dynamique propre. Celles-ci doivent être prises en compte afin de résoudre la dynamique du système complet [21].

2.2.2 Dynamique de Landau-Lifshitz-Bloch

2.2.2.1 Variable gaussienne aléatoire et bruit thermique

Nous considérons une variable aléatoire $\tilde{\omega}_i$ définie comme $\langle \tilde{\omega}_i \tilde{\omega}_j \rangle = \frac{D}{\tau}\delta_{ij} \exp\left(-\frac{|t-t'|}{\tau}\right)$ avec $\langle \tilde{\omega}_i \rangle = 0$. Cette variable Gaussienne aléatoire nous permet de modéliser un bruit thermique auquel est soumis notre système. On définit la variable s_i comme la composante i du vecteur unitaire pointant dans la même direction que l'aimantation $\vec{m}$, et le rapport entre le champ effectif et sa pulsation par $\hbar\omega_i = -\delta H_{eff}/\delta s_i$ définit comme la composante i de la pulsation du champ effectif, nous pouvons écrire l'équation stochastique de Landau-Lifshitz-Gilbert :

$$\frac{\partial}{\partial t} s_i = \frac{1}{1+\lambda^2}\epsilon_{ijk} s_k \left[\omega_j + \tilde{\omega}_j - \lambda \epsilon_{jlm}\omega_l s_m\right] \tag{2.48}$$

En prenant la moyenne sur le bruit $\tilde{\omega}$ de l'équation ci-dessus et en considérant que ω n'est pas corrélé à s, ni à $\tilde{\omega}$, mais est une fonction de s, nous pouvons obtenir :

$$\frac{\partial}{\partial t}\langle s_i \rangle = \frac{1}{1+\lambda^2}\epsilon_{ijk}\left[\omega_j \langle s_k \rangle + \langle \tilde{\omega}_j s_k \rangle - \lambda \epsilon_{jlm}\omega_l \langle s_k s_m \rangle\right] \tag{2.49}$$

Afin de trouver la dynamique de $\langle \tilde{\omega}_j s_k \rangle$ et $\langle s_k s_m \rangle$, nous utilisons la méthode de Shapiro-Loginov [13], stipulant que :

$$\frac{\partial}{\partial t}\langle \alpha\beta[\alpha] \rangle = \langle \alpha \frac{\partial}{\partial t}\beta[\alpha] \rangle - \frac{1}{\tau}\langle \alpha\beta[\alpha] \rangle \tag{2.50}$$

avec α jouant le rôle de la variable Gaussienne aléatoire et $\beta[\alpha]$ une fonction dépendante de α. Cette méthode appliquée, nous obtenons :

$$\frac{\partial}{\partial t}\langle \tilde{\omega}_i s_j \rangle = \langle \tilde{\omega}_i \frac{\partial}{\partial t} s_j \rangle - \frac{1}{\tau}\langle \tilde{\omega}_i s_j \rangle \tag{2.51}$$

Cependant, cette relation ne s'applique que lorsqu'une fonction de corrélation met en jeu une variable aléatoire. Nous ne pouvons donc pas appliquer directement cette méthode pour la fonction de corrélation liant s_i et s_j, mais nous pouvons tout de même écrire :

$$\frac{\partial}{\partial t}\langle s_i s_j\rangle = \langle s_i \frac{\partial s_j}{\partial t}\rangle + \langle \frac{\partial s_i}{\partial t} s_j\rangle \tag{2.52}$$

En reprenant l'expression 2.48 et en l'insérant dans les équations 2.51 et 2.52, nous pouvons obtenir la dynamique de $\langle\tilde{\omega}_i s_j\rangle$ et $\langle s_i s_j\rangle$. Ainsi :

$$\frac{\partial}{\partial t}\langle s_i\rangle = \frac{1}{1+\lambda^2}\epsilon_{ijk}\left[\omega_j\langle s_k\rangle + \langle\tilde{\omega}_j s_k\rangle - \lambda\epsilon_{jlm}\omega_l\langle s_k s_m\rangle\right] \tag{3.53.a}$$

$$\frac{\partial}{\partial t}\langle\tilde{\omega}_i s_j\rangle = -\frac{1}{\tau}\langle\tilde{\omega}_i s_j\rangle + \frac{1}{1+\lambda^2}\left[\epsilon_{jkl}\omega_k\langle\tilde{\omega}_j s_l\rangle + \epsilon_{jkl}\langle\tilde{\omega}_i\tilde{\omega}_k s_n\rangle + \lambda\epsilon_{jkl}\epsilon_{lmn}\omega_m\langle\tilde{\omega}_i s_k s_n\rangle\right] \tag{3.53.b}$$

$$\frac{\partial}{\partial t}\langle s_i s_j\rangle = \frac{1}{1+\lambda^2}\epsilon_{jkl}\left[\omega_k\langle s_i s_l\rangle + \langle\tilde{\omega}_k s_i s_l\rangle - \lambda\epsilon_{lmn}\omega_m\langle s_i s_l s_n\rangle\right] + (i \leftrightarrow j) \tag{3.53.c}$$

2.2.2.2 Approximation de la fermeture gaussienne

L'équation 3.53.a ne pose pas de problème car elle ne met en jeu que la variable moyennée $\langle s_k\rangle$ ainsi que les fonctions de corrélation à deux points $\langle\tilde{\omega}_j s_k\rangle$ et $\langle s_k s_m\rangle$ qui ont chacune leur dynamique caractérisée par la hiérarchie ci-dessus. Les éléments problématiques proviennent des équations 3.53.b et 3.53.c qui reposent sur des fonctions de corrélation à trois points, respectivement $\langle\tilde{\omega}_i s_k s_n\rangle$ et $\langle\tilde{\omega}_i\tilde{\omega}_k s_n\rangle$ pour l'équation 3.53.b ainsi que $\langle\tilde{\omega}_i s_i s_l\rangle$ et $\langle s_i s_l s_n\rangle$ pour l'équation 3.53.c. Ces fonctions de corrélations à trois points n'ayant pas leur dynamique propre, du moins dans cette hiérarchie d'équations, nous nous devons d'effectuer une approximation afin de les décomposer à l'ordre inférieur. La relation de fermeture invoquée dans [2] est donc utilisée :

$$\langle\langle s_i s_j s_k\rangle\rangle = \langle s_i s_j s_k\rangle - \langle s_i\rangle\langle s_j s_k\rangle - \langle s_j\rangle\langle s_i s_k\rangle - \langle s_k\rangle\langle s_i s_j\rangle + 2\langle s_i\rangle\langle s_j\rangle\langle s_k\rangle = 0 \tag{3.54}$$

Où $\langle\langle\circ\rangle\rangle$ désigne le cumulant[21]. Il convient de remarquer qu'il nous est possible de dériver une autre hiérarchie d'équations, en considérant que le cumulant stochastique d'ordre 4 $\langle\langle s_i s_j s_k s_l\rangle\rangle$ en plus de celui d'ordre 3, s'annule, donnant ainsi naissance à des termes d'ordre 2 et d'ordre 1 supplémentaires [14]. Cependant, ces termes supplémentaires ajoutent de la complexité à des équations loin d'être triviales. En appliquant la relation 3.54, nous pouvons décomposer les cumulants d'ordre 3 des équations 3.53.b et 3.53.c :

$$\langle\tilde{\omega}_i s_j s_k\rangle = \langle s_j\rangle\langle\tilde{\omega}_i s_k\rangle + \langle s_k\rangle\langle\tilde{\omega}_i s_j\rangle \tag{3.55.a}$$

$$\langle\tilde{\omega}_i\tilde{\omega}_j s_k\rangle = \frac{D}{\tau}\delta_{ij}\langle s_k\rangle \tag{3.55.b}$$

$$\langle s_i s_j s_k\rangle = \langle s_i\rangle\langle s_j s_k\rangle + \langle s_j\rangle\langle s_i s_k\rangle + \langle s_k\rangle\langle s_i s_j\rangle - 2\langle s_i\rangle\langle s_j\rangle\langle s_k\rangle \tag{3.55.c}$$

2.2.2.3 Dynamique stochastique du système

Il est important de préciser que 3.55.b n'est vraie que si l'on considère $\tilde{\omega}_i$ et $\tilde{\omega}_j$ au même temps t. Dans le cas contraire, nous serions contraints de considérer que $\langle\tilde{\omega}_i(t)\tilde{\omega}_j(t')s_k(t)\rangle = \frac{D}{\tau}\delta_{ij}\exp\left(-\frac{|t-t'|}{\tau}\right)\langle s_k(t)\rangle$. Cependant, nous travaillons à temps égal. Ainsi, en insérant les expressions ci-dessus dans la hiérarchie obtenue précédemment, nous sommes à même d'obtenir la hiérarchie complète, fermée par la relation 3.54.

$$\frac{\partial}{\partial t}\langle s_i\rangle = \frac{1}{1+\lambda^2}\epsilon_{ijk}\left[\omega_j\langle s_k\rangle + \langle\tilde{\omega}_j s_k\rangle - \lambda\epsilon_{jlm}\omega_l\langle s_k s_m\rangle\right] \tag{3.56.a}$$

$$\begin{aligned}\frac{\partial}{\partial t}\langle\tilde{\omega}_i s_j\rangle = &-\frac{1}{\tau}\langle\tilde{\omega}_i s_j\rangle + \frac{1}{1+\lambda^2}\left[\epsilon_{jkl}\omega_k\langle\tilde{\omega}_j s_l\rangle + \epsilon_{jil}\frac{D}{\tau}\langle s_l\rangle\right]\\ &-\frac{\lambda}{1+\lambda^2}\left[\lambda\epsilon_{jkl}\epsilon_{kmn}\omega_m\left(\langle s_l\rangle\langle\tilde{\omega}_i s_n\rangle + \langle s_n\rangle\langle\tilde{\omega}_i s_l\rangle\right)\right]\end{aligned} \tag{3.56.b}$$

$$\begin{aligned}\frac{\partial}{\partial t}\langle s_i s_j\rangle = &+\frac{1}{1+\lambda^2}\epsilon_{jkl}\left[\omega_k\langle s_i s_l\rangle + \epsilon_{jkl}\left(\langle s_i\rangle\langle\tilde{\omega}_k\rangle + \langle s_l\rangle\langle\tilde{\omega}_k s_i\rangle\right)\right]\\ &-\frac{\lambda}{1+\lambda^2}\epsilon_{jkl}\epsilon_{kmn}\omega_m\left[\langle s_i\rangle\langle s_l s_n\rangle + \langle s_l\rangle\langle s_i s_n\rangle\right]\\ &-\frac{\lambda}{1+\lambda^2}\epsilon_{jkl}\epsilon_{kmn}\omega_m\left[\langle s_n\rangle\langle s_i s_l\rangle - 2\langle s_i\rangle\langle s_l\rangle\langle s_n\rangle\right]\\ &+(i\leftrightarrow j)\end{aligned} \tag{3.56.c}$$

Afin de décrire la dynamique de l'aimantation au sein d'un matériau en contact avec un bain thermique, nous devons résoudre simultanément ce système de vingt-et-une équations différentielles, hautement non-linéaires. Lorsque nous nous penchons sur la première équation du système 3.56.c et en considérant nos deux fonctions de corrélations comme des tenseurs, nous pouvons remarquer une forte analogie avec le modèle de Landau-Lifshitz-Bloch présentant des amortissements transverse et longitudinal [15] (ce qui est également observé ici). Ainsi, il est pertinent de qualifier la hiérarchie ci-dessus de modèle dynamique de Landau-Lifshitz-Bloch (dLLB) [2].

2.3 Construction des intégrateurs numériques

Tournons-nous maintenant vers les intégrateurs numériques, qui seront au cœur de notre travail. Ceux-ci permettent d'intégrer des systèmes d'équations de façon numérique. Bien entendu, ces intégrateurs ne permettent pas de trouver la solution analytique exacte comme pourrait le permettre un logiciel de calcul formel. Cependant, ils permettent d'obtenir une dynamique approchée de la dynamique exacte, en faisant des approximations, permettant de trouver un bon compromis entre temps de calcul et précision. Un bon intégrateur permet donc d'obtenir une dynamique approchée en un temps raisonnable.

2.3.1 Développement de Magnus

2.3.1.1 Généralités

Le développement de Magnus [16] [17] permet de représenter la solution d'une équation différentielle (linéaire et homogène) d'ordre 1 sous la forme d'une exponentielle. En général, les différents

intégrateurs que nous utilisons considèrent différentes approximations du développement de Magnus, d'où son intérêt lors de l'étude de solution numérique. Ainsi, en considérant un opérateur linéaire $A(t)$, nous avons :

$$Y'(t) = A(t)Y(t) \tag{3.57}$$

avec $Y(t_0) = Y_0$. Nous définissons le propagateur $U(t, t_0)$ tel que $Y(t) = U(t, t_0)Y_0$. Ainsi, nous obtenons l'expression suivante :

$$U'(t) = A(t)U(t) \tag{3.58}$$

avec $U(t_0) = \mathbb{I}_{N\times N}$. L'idée ensuite est de chercher une représentation exponentielle du propagateur U. Nous écrivons $U(t) = \exp(\Omega(t))$, avec $\Omega(t_0) = \mathbb{O}_{N\times N}$. L'opérateur $\Omega(t)$ peut lui-même être représenté sous la forme d'une série :

$$\Omega(t) = \sum_{k=1}^{+\infty} \Omega_k(t) \tag{3.59}$$

Les différents termes $\Omega_k(t)$ sont :

$$\Omega_1(t) = \int_0^t A(t_1)dt_1 \tag{3.60.a}$$

$$\Omega_2(t) = \frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2[A(t_1), A(t_2)] \tag{3.60.b}$$

$$\Omega_2(t) = \frac{1}{6}\int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} ([A(t_1), [A(t_2), A(t_3)]] + [A(t_3), [A(t_2), A(t_1)]]) \tag{3.60.c}$$

etc. Il convient de remarquer que même si les premiers termes semblent relativement simples à calculer, les termes suivants deviennent très complexes si nous n'avons pas de relation de commutation simple. Il est intéressant de remarquer la proximité de ces développements avec la formule de Baker–Campbell–Hausdorff, très utilisée en mécanique quantique et en théorie quantique des champs : le développement de Magnus est une généralisation de cette formule. Il permet aussi de justifier l'apparition de l'opérateur "time-ordering" dans le développement de Dyson [18].

2.3.1.2 Construction de l'intégrateur explicite d'Euler

Comme mentionné plus haut, le développement de Magnus est utilisé pour la construction d'intégrateurs numériques. Ceux-ci permettent ensuite la résolution de la dynamique du système considéré. Ce n'est pas une partie à négliger car un mauvais intégrateur donnera systématiquement de mauvais résultats, et ce dans la plupart des cas.

Afin de souligner ce point, considérons un système physique simple : le pendule. Sa dynamique est donnée par les équations de Hamilton $\dot{q} = p$ et $\dot{p} = -\sin(q)$. Nous cherchons alors à tracer $z(t) = (q(t), p(t))^T$ et définissons $f(z) = (p, -\sin(q))$. Les méthodes employées sont les suivantes :

- Méthode explicite d'Euler : $z_{k+1} = z_k + hf(z_k)$
- Méthode implicite d'Euler : $z_{k+1} = z_k + hf(z_{k+1})$
- Méthode symplectique d'Euler : $z_{k+1} = z_k + hf(q_k, p_{k+1})$
- Méthode du point milieu implicite : $z_{k+1} = z_k + hf((z_{k+1} + z_k)/2)$

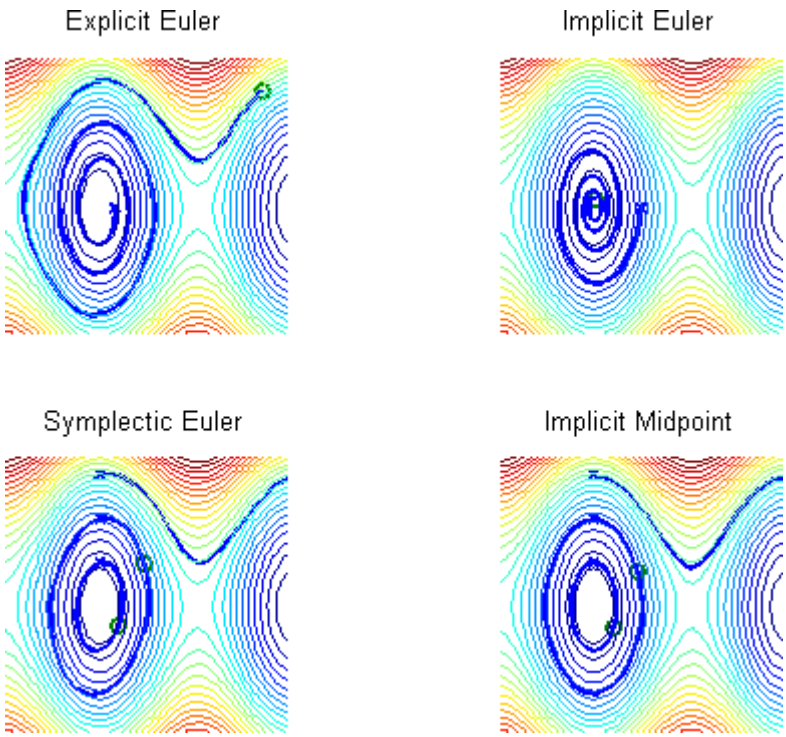


FIGURE 2.1 – Comparaison des dynamiques générées par différentes méthodes d'intégrations (voir *https://upload.wikimedia.org/wikipedia/commons/1/1d/Pendulumtrajectories.png*)

Nous remarquons que seuls les deux derniers intégrateurs donnent la bonne dynamique. Afin de comprendre pourquoi, attardons-nous un peu plus sur la méthode explicite d'Euler. Celle-ci stipule que la dynamique d'un système via la relation $z_{k+1} = z_k + hf(z_k)$. Cette méthode est construite à partir de plusieurs approximations. Tout d'abord, considérons le développement de Magnus :

$$U(t) = \exp\left(\Omega(t)\right) = \exp\left(\Omega_1(t) + \Omega_2(t) + ...\right) \tag{3.61}$$

Nous commençons par faire l'approximation $U(t) \approx \exp\left(\Omega_1(t)\right) = \exp\left(\int_0^t A(t_1)dt_1\right)$. Nous considérons ensuite l'intégration de 0 à τ avec $\tau << 1$. Ceci nous permet d'écrire $U(\tau) = \exp\left(\int_0^\tau A(t_1)dt_1\right)$. La deuxième approximation vient du développement à l'ordre 1 de l'exponentielle, ce qui nous permet d'obtenir la relation suivante :

$$U(\tau) = 1 + \int_0^\tau A(t_1)dt_1 \tag{3.62}$$

Vient finalement une dernière approximation qui consiste à considérer que l'opérateur A est constant lors de l'intégration sur un petit temps τ. Nous obtenons ainsi :

$$U(\tau) = 1 + A(t_0)\tau \tag{3.63}$$

En appliquant cet opérateur à $Y(t_0)$, nous obtenons la relation explicite d'Euler :

$$Y(\tau) = Y_0 + \tau Y'(t_0) \tag{3.64}$$

Ainsi, la méthode d'Euler repose sur trois approximations du développement de Magnus, ce qui permet d'expliquer à la fois sa mauvaise précision et sa mauvaise stabilité. La méthode implicite

d'Euler repose elle-aussi sur un schéma d'ordre 1 pour résoudre la dynamique du système, ce qui correspond aussi à un développement au premier ordre de la série de Magnus. Sachant que le terme $\Omega_1(t)$ ne permet pas trouver la bonne solution (car nous forçons la solution à prendre la forme d'une exponentielle), nous sommes obligés, si nous voulons une solution approchée, de considérer les termes supérieurs du développement de Magnus qui assurent systématiquement la correction de la représentation exponentielle : la représentation exacte de la solution est alors trouvée lorsque nous considérons tous les termes du développement, ce qui est impossible en pratique.

2.3.1.3 Construction de l'intégrateur explicite de Runge-Kutta

L'intégrateur de Runge-Kutta explicite d'ordre 4 est une méthode d'intégration numérique développée par Runge [19] permettant d'intégrer des équations différentielles et systèmes d'équations différentielles d'ordre 1. Celui-ci est très utilisé car réputé pour être relativement stable et précis. Afin d'étudier son fonctionnement, considérons l'équation différentielle $dy/dx = f(y)$.

La méthode de Runge-Kutta explicite d'ordre 4 stipule que :

$$y_{n+1} = y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{3.65}$$

avec les coefficients k_s donnés par :

$$\begin{aligned} k_1 &= hf(y) \\ k_2 &= hf\left(y + \frac{k_1}{2}\right) \\ k_3 &= hf\left(y + \frac{k_2}{2}\right) \\ k_4 &= hf\left(y + k_3\right) \end{aligned}$$

Ainsi, en injectant ces coefficients dans la relation ci-dessus, nous obtenons l'analogue du développement en série de Taylor à l'ordre 4 de la fonction y :

$$y_{n+1} = y_n + a(y_n)h + b(y_n)h^2 + c(y_n)h^3 + d(y_n)h^4 + \mathcal{O}(h^5) \tag{3.66}$$

Cette méthode peut aussi être vue comme une approximation du développement de Magnus [17].

2.3.2 Intégrateur symplectique

Un intégrateur symplectique est un intégrateur numérique qui conserve la structure de l'équation. Ainsi, si une équation conserve la norme d'un certain vecteur, celle-ci sera automatiquement conservée lors de l'utilisation d'un intégrateur symplectique. Cependant, celui-ci ne conserve pas forcément l'énergie du système, ce qui pour des temps de simulation assez longs, induit des erreurs. En revanche, lorsque nous nous intéressons à des "petits" temps, nous pouvons obtenir la conservation de l'énergie et de la structure de l'équation.

2.3.2.1 Généralités

Les intégrateurs symplectiques sont construits afin de résoudre les équations de Hamilton :

$$\dot{p} = -\frac{\partial H}{\partial q} \text{ et } \dot{q} = \frac{\partial H}{\partial p} \tag{3.67}$$

L'évolution temporelle des équations de Hamilton étant symplectomorphique, la 2-forme $dp \wedge dq$ est conservée au cours du temps. L'intégrateur symplectique conserve lui aussi cette 2-forme.

2.3.2.2 Construction de l'intégrateur de l'équation de Landau-Lifshitz-Gilbert

Considérons l'équation de Landau-Lifshitz-Gilbert :

$$(1+\lambda^2)\frac{d\vec{m}}{dt} = -\gamma\mu_0\vec{m} \times \vec{H}_{eff} - \gamma\mu_0\lambda\vec{m} \times \left(\vec{m} \times \vec{H}_{eff}\right) \tag{3.68}$$

En effectuant le produit scalaire avec $\vec{m}$ de chaque côté de l'équation, nous pouvons annuler le terme de droite. Nous obtenons alors le fait que $\vec{m} \times d\vec{m}/dt = 0$, entraînant la conservation de la norme de $\vec{m}$. Développons le triple produit vectoriel, en utilisant la relation $\vec{a} \times (\vec{b} \times \vec{c}) = (\vec{a} \cdot \vec{c})\vec{b} - (\vec{a} \cdot \vec{b})\vec{c}$. Nous pouvons écrire :

$$(1+\lambda^2)\frac{d\vec{m}}{dt} = -\gamma\mu_0\vec{m} \times \vec{H}_{eff} - \gamma\mu_0\lambda\left((\vec{m} \cdot \vec{m})\vec{H}_{eff} - (\vec{m} \cdot \vec{H}_{eff})\vec{m}\right) \tag{3.69}$$

Nous pouvons remarquer que la partie de droite de l'équation se décompose suivant trois vecteurs distincts : $\vec{m}$, $\vec{H}_{eff}$ et $\vec{m} \times \vec{H}_{eff}$. Considérant ceci, nous pouvons en déduire qu'il est possible de décomposer le vecteur $d\vec{m}/dt$, ainsi que le vecteur $\vec{m}$ sur cette base de vecteurs. Nous cherchons ainsi :

$$\vec{m} = A(t)\vec{m}_0 + B(t)\frac{\vec{H}_{eff}}{H_{eff}} + C(t)\frac{\vec{H}_{eff}}{H_{eff}} \times \vec{m}_0 \tag{3.70}$$

Sachant qu'à l'instant initial, nous avons $\vec{m} = \vec{m}_0$, nous obtenons les conditions $A(t_0) = 1$, $B(t_0) = 0$ et $C(t_0) = 0$. Nous cherchons ensuite les équations régissant $A(t)$, $B(t)$ et $C(t)$. Pour cela, nous considérons l'équation sur la dérivée temporelle de $\vec{m}$:

$$\frac{d\vec{m}}{dt} = \dot{A}(t)\vec{m}_0 + \dot{B}(t)\vec{H}_{eff} + \dot{C}(t)\vec{H}_{eff} \times \vec{m}_0 \tag{3.71}$$

Il suffit maintenant d'injecter l'expression 3.66 dans la relation 3.65 afin d'identifier les dérivées temporelles des fonctions A, B et C. Posons $\xi = \vec{m}(0) \cdot \vec{H}_{eff}/H_{eff}$ ainsi que $\chi = \vec{H}_{eff} \cdot \vec{H}_{eff}/H_{eff}^2$. Nous obtenons alors :

$$\dot{A}(t) = -\gamma\mu_0\chi C(t) + \gamma\mu_0\lambda\left(A^2(t)\xi + A(t)B(t)\chi\right) \tag{3.72.a}$$

$$\dot{B}(t) = -\gamma\mu_0\xi C(t) - \gamma\mu_0\lambda\left(1 - A(t)B(t)\xi - B^2(t)\chi\right) \tag{3.72.b}$$

$$\dot{C}(t) = -\gamma\mu_0 A(t) + \gamma\mu_0\lambda\left(A(t)C(t)\xi + B(t)C(t)\chi\right) \tag{3.72.c}$$

Afin de trouver les fonctions $A(t)$, $B(t)$ et $C(t)$, nous devons résoudre ce système de trois équations couplées. Ceci peut être fait aisément de façon numérique. Dans le cas où $\gamma\mu_0\lambda = 0$, c'est-à-dire lorsqu'il n'y a pas "d'amortissement", le système est soluble analytiquement :

$$\dot{A}(t) = -\gamma\mu_0\chi C(t) \tag{3.73.a}$$

$$\dot{B}(t) = -\gamma\mu_0\xi C(t) \tag{3.73.b}$$

$$\dot{C}(t) = -\gamma\mu_0 A(t) \tag{3.73.c}$$

Ce système se résolvant aisément, nous obtenons les solutions suivantes :

$$A(t) = 2\sinh(\gamma\mu_0\sqrt{\chi}t) + e^{-\gamma\mu_0\sqrt{\chi}t} \tag{3.74.a}$$

$$B(t) = -\frac{\gamma\mu_0\xi}{\sqrt{\chi}}\left(t - \frac{e^{-\gamma\mu_0\sqrt{\chi}t}}{\gamma\mu_0\sqrt{\chi}} - \frac{2\sinh(\gamma\mu_0\sqrt{\chi}t)}{\gamma\mu_0\sqrt{\chi}} + \frac{1}{\gamma\mu_0\sqrt{\chi}}\right) \tag{3.75.b}$$

$$C(t) = \frac{1}{\sqrt{\chi}}\left(1 + e^{-\gamma\mu_0\sqrt{\chi}t} - 2\cosh(\gamma\mu_0\sqrt{\chi}t)\right) \tag{3.76.c}$$

Nous pouvons constater que ces solutions vérifient les conditions initiales $A(t=0) = 1$, $B(t=0) = 0$ et $C(t=0) = 0$. Si maintenant nous considérons le cas où $\lambda \neq 0$, nous pouvons nous servir des expressions précédentes et utiliser un développement en série de α. Ceci n'est alors valable seulement si nous considérons $\lambda \ll 1$, ce qui n'est pas forcément notre cas. Ainsi, il est plus astucieux de projeter notre système sur une sphère, en utilisant les coordonnées sphériques. Nous cherchons alors $\vec{m}(t) = m_0\sin(\theta)\cos(\phi)\hat{e}_r + m_0\sin(\theta)\sin(\phi)\hat{e}_\theta + m_0 cos(\theta)\hat{e}_\phi$. En injectant cette forme dans l'équation 3.65, nous pouvons obtenir la dynamique suivante :

$$\phi(t) \propto H_{eff,z}t \tag{3.77.a}$$

$$\theta(t) \propto \arctan(e^{-H_{eff,z}t}) \tag{3.77.b}$$

Ici, nous avons considéré la normalisation de $m_0 = 1$. En reprenant la décomposition du vecteur $\vec{m}$ sur la base sphérique, et en utilisant les relations entre la base sphérique et cartésienne :

$$\hat{e}_r = \sin(\theta)\cos(\phi)\hat{e}_x + \sin(\theta)\sin(\phi)\hat{e}_y + \cos(\theta)\hat{e}_z \tag{3.78.a}$$

$$\hat{e}_\theta = \cos(\theta)\cos(\phi)\hat{e}_x + \cos(\theta)\sin(\phi)\hat{e}_y - \sin(\theta)\hat{e}_z \tag{3.78.b}$$

$$\hat{e}_\phi = -\sin(\phi)\hat{e}_x + \cos(\phi)\hat{e}_y \tag{3.78.c}$$

Nous pouvons retrouver l'expression des composantes de $\vec{m} = m_x\hat{e}_x + m_y\hat{e}_y + m_z\hat{e}_z$ dans la base cartésienne :

$$m_x = \frac{e^{-2H_{eff,z}t}\cos^2(H_{eff,z}t)}{e^{-2H_{eff,z}t}+1} + \frac{e^{-H_{eff,z}t}\sin(H_{eff,z}t)\cos(H_{eff,z}t)}{e^{-2H_{eff,z}t}+1} - \frac{\sin(H_{eff,z}t)}{\sqrt{e^{-2H_{eff,z}t}+1}} \tag{3.79.a}$$

$$m_y = \frac{e^{-H_{eff,z}t}\cos(H_{eff,z}t)\sin(H_{eff,z}t)}{e^{-2H_{eff,z}t}+1} + \frac{e^{-2H_{eff,z}t}\sin^2(H_{eff,z}t)}{e^{-2H_{eff,z}t}+1} + \frac{\cos(H_{eff,z}t)}{\sqrt{e^{-2H_{eff,z}t}+1}} \tag{3.79.b}$$

$$m_z = \frac{e^{-H_{eff,z}t}\cos(H_{eff,z}t)}{e^{-2H_{eff,z}t}+1} - \frac{e^{-2H_{eff,z}t}\sin(H_{eff,z}t)}{e^{-2H_{eff,z}t}+1} \tag{3.79.c}$$

Bien que ces expressions paraissent compliquées, elles sont relativement simples à obtenir : nous avons la dynamique exacte de l'équation de Landau-Lifshitz-Gilbert (en considérant des relations de proportionnalité pour $\phi(t)$ et $\theta(t)$ afin de réduire la taille des expressions à manipuler). Il convient de remarquer que lorsque nous complexifions l'expression de la dynamique de l'aimantation, l'intégration analytique du système devient impossible, et nous avons recours à l'intégration numérique afin de déterminer les coefficients $A(t)$, $B(t)$ et $C(t)$. De manière générale, il est toujours intéressant de chercher un intégrateur symplectique afin d'intégrer "correctement" le système considéré, mais en pratique, il est parfois très difficile de le trouver, citons par exemple le cas de la dynamique de Landau-Lifshitz-Bloch qui requiert une intégration d'un vecteur mais aussi de matrices (fonctions de corrélations), qui ne sont pas décomposables sur une base comme nous l'avons fait pour l'équation de Landau-Lifshitz-Gilbert. Ainsi, si la recherche d'un intégrateur symplectique s'avère presque perdue pour certain problème, il reste intéressant de se tourner vers le développement de Magnus et d'effectuer les calculs pour les premiers ordres afin d'obtenir un bon intégrateur, à défaut d'en avoir un symplectique.

Chapitre 3

Approche numérique

3.1 Généralités

A mesure que la complexité des systèmes et des équations augmentent, nous avons de plus en plus recours à la simulation numérique comme première expérience avant la réelle expérimentation, permettant d'économiser à la fois du temps mais aussi des matériaux. La puissance de calcul grandissante des différents ordinateurs permet maintenant la simulation de systèmes d'assez grande taille (typiquement un système de 10^4 à 10^6 spins, par exemple).

Bien qu'il n'y ait pas de réel coût matériel sinon l'achat de la machine (en plus, bien sûr, du coût de l'électricité et du personnel), deux contraintes importantes s'ajoutent naturellement : le temps de calcul et la précision. Il est primordial de prendre ces deux paramètres en compte afin de développer une solution viable sur le long terme. Vient ensuite le choix du langage de programmation, en lien avec l'objectif en question : prototypage ou calcul. Le prototypage s'effectuera plus facilement avec une langage de haut niveau comme Python, alors que le C sera beaucoup plus intéressant pour effectuer les calculs, car plus bas niveau.

Nous avons choisi le langage Python, et la bibliothèque NumPy, afin d'avoir accès à une simplicité d'écriture que ne permettait pas les langages de bas niveau (typiquement manipulation de très gros tableaux, de vecteurs, de matrices, etc.). En plus de cela, un code micromagnétique était développé dans ce langage [3]. L'objectif était donc de reprendre ce code, et de l'adapter ensuite pour étudier nos différents problèmes.

3.2 Code micromagnétique et équation de Landau-Lifshitz-Gilbert

Commençons par l'étude de l'équation de Landau-Lifshitz-Gilbert développée dans la première partie de l'approche analytique. Celle-ci se caractérise par l'étude d'une assemblée de "spins", soumis à des champs internes et externes. Contrairement à une dynamique atomistique, la dynamique micromagnétique est beaucoup plus lente à cause de certaines approximations que nous verrons un peu plus loin. Le tout est que nous ne pouvons pas générer n'importe quelle configuration initiale. L'objectif, en reprenant le code *70 lines of Numpy*, était dans un premier temps de changer l'intégrateur d'Euler, très instable, par un intégrateur de Runge-Kutta d'ordre 4, plus commode, pour enfin implémenter un intégrateur symplectique, conservant naturellement la norme de l'aimantation.

3.2.1 Code micromagnétique et calcul des champs internes

L'aimantation étant soumise à différentes interactions, nous devons dans un premier trouver l'expression atomistique des champs internes et externes, pour ensuite faire l'approximation micromagnétique.

3.2.1.1 Calcul du champ d'échange

Le champ d'échange correspond à l'interaction d'un spin avec ses plus proches voisins. Ceux-ci ont tendance à s'orienter dans la même direction. L'énergie d'échange peut être obtenue en considérant l'Hamiltonien de Heisenberg 2.4 Le produit scalaire entre les aimantations réduites $\vec{m}_i$ et $\vec{m}_j$ peut se décomposer en faisant apparaître l'angle entre ces deux vecteurs, que l'on notera θ_{ij}. Nous avons :

$$E_{ex} = -M_s^2 \sum_{\langle I,K \rangle} J_{IK} m_I m_J \cos(\theta_{IK}) \tag{3.1}$$

Nous supposons que les vecteurs ont une norme constante, que l'on dénote par s, et que l'interaction n'est constante qu'entre les mêmes premiers voisins, permettant de réduire $J_{IK} \equiv J$:

$$E_{ex} = -Js^2 M_s^2 \sum_{\langle I,K \rangle} \cos(\theta_{IK}) \tag{3.2}$$

L'idée est ensuite de considérer l'approximation micromagnétique, stipulant que $\theta_{IK} \ll 1$. Ainsi, nous pouvons effectuer le développement limité du cosinus $\cos(\theta_{IK}) = 1 - \theta_{IK}^2/2! + \mathcal{O}(\theta_{IK}^4)$.

$$E_{ex} = -Js^2 M_s^2 \sum_{\langle I,K \rangle} \left(1 + \mathcal{O}(\theta_{IK}^4)\right) + \frac{Js^2 M_s^2}{2} \sum_{\langle I,K \rangle} \theta_{IK}^2 \tag{3.3}$$

Nous pouvons ensuite remarquer que $|\vec{m}_I - \vec{m}_K|^2 = m_I^2 + m_K^2 - 2\vec{m}_I \cdot \vec{m}_K = 2s^2 - 2s^2 \cos(\theta_{IK})$. En utilisant le développement limité du cosinus, nous avons $|\vec{m}_I - \vec{m}_K|^2 = 2s^2 - 2s^2 \left(1 - \theta_{IK}^2/2 + \mathcal{O}(\theta_{IK}^4)\right)$. Ainsi, nous obtenons $\theta_{IK}^2 = |\vec{m}_I - \vec{m}_K|^2/s^2 + \mathcal{O}(\theta_{IK}^4)$. En réinjectant dans l'expression 3.4, celle-ci devient :

$$E_{ex} = -Js^2 M_s^2 \sum_{\langle I,K \rangle} \left(1 + \frac{\mathcal{O}(\theta_{IK}^4)}{2}\right) + \frac{JM_s^2}{2} \sum_{\langle I,K \rangle} |\vec{m}_I - \vec{m}_K|^2 \tag{3.4}$$

Si maintenant nous posons que $\vec{m}_I$ et $\vec{m}_K$ sont très proches, nous pouvons considérer que le vecteur $\vec{m}_K$ peut être trouvé à partir du vecteur voisin $\vec{m}_I$ en lui ajoutant un gradient du vecteur d'aimantation $\vec{m}$: $\vec{m}_K \approx \vec{m}_I + (\vec{s}_I \cdot \vec{\nabla})\vec{m}$, avec $\vec{s}_I$ l'aimantation réduite au point I, définie par $\vec{s}_I = \vec{m}_I/|\vec{m}_I|$. Nous pouvons donc écrire $|\vec{m}_K - \vec{m}_I|^2 = [(\vec{s}_I \cdot \vec{\nabla})\vec{m}]^2$. Ainsi :

$$E_{ex} = -Js^2 M_s^2 \sum_{\langle I,K \rangle} \left(1 + \frac{\mathcal{O}(\theta_{IK}^4)}{2}\right) + \frac{JM_s^2}{2} \sum_{\langle I \rangle} \left[(\vec{s}_I \cdot \vec{\nabla})\vec{m}\right]^2 \tag{3.5}$$

Par une renormalisation des différents paramètres du système, il est possible de supprimer le terme constant dans la somme de gauche. De plus, la considération que $\theta_{IK} \ll 1$ nous permet de nous

affranchir des corrections d'ordre supérieur du cosinus, mais en veillant à toujours rester dans le domaine de cette approximation. Finalement, nous obtenons :

$$E_{ex} = \frac{JM_s^2}{2} \sum_{\langle I \rangle} \left[(\vec{s}_I \cdot \vec{\nabla}) \vec{m} \right]^2 \tag{3.6}$$

Afin de trouver le champ d'échange, nous devons effectuer la dérivée fonctionnelle de l'énergie par rapport à l'aimantation. Cependant, il est plus astucieux de se rendre compte que nous avons la relation $\Delta(UV) = U\Delta(V) + V\Delta(U) + 2(\vec{\nabla}U) \cdot (\vec{\nabla}V)$. Dans notre cas, ceci revient à écrire $\Delta(\vec{m}^2) = 2\vec{m}\Delta(\vec{m}) + 2(\vec{\nabla}\vec{m}) \cdot (\vec{\nabla}\vec{m})$. Ainsi, nous avons :

$$\vec{H}_{ex,i} = -\frac{1}{\mu_0} \frac{\delta^2 E_{ex}}{\delta \vec{m}_I \delta V} = -\frac{JM_s^2}{2\mu_0 V} \frac{\delta}{\delta \vec{m}_I} \left(\sum_{\langle K \rangle} \left(\frac{1}{2} \Delta \vec{m}^2 - \vec{m}_K \Delta \vec{m}_K \right) \right) \tag{3.7}$$

La norme de $\vec{m}$ étant préservée, son carré est lui aussi constant. Ainsi, son Laplacien s'annule. Nous obtenons donc le champ d'échange :

$$\vec{H}_{ex,i} = \frac{JM_s^2}{2\mu_0 V} \sum_{\langle K \rangle} \delta_{IK} \Delta \vec{m}_K = \frac{JM_s^2}{2\mu_0 V} \Delta \vec{m}_I \tag{3.8}$$

Le Laplacien peut être calculé de façon numérique, en utilisant la méthode des différences finies. Celle-ci stipule que :

$$\Delta u(x,y) \approx \frac{u(x-dx,y) - 2u(x,y) + u(x+dx,y)}{dx^2} + \frac{u(x,y-dy) - 2u(x,y) + u(x,y+dy)}{dy^2}$$

Ainsi, pour un ensemble discret de points, nous obtenons :

$$\Delta u_{i,j} \approx \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}}{a^2}$$

Avec a la distance entre deux sites. Finalement, le champ d'échange pour un ensemble discret d'éléments, en deux dimensions, peut s'écrire comme :

$$\vec{H}_{ex}^{(I,J)} = \frac{JM_s^2}{2\mu_0 V} \left(\frac{\vec{m}_{I-1,J} + \vec{m}_{I+1,J} + \vec{m}_{I,J-1} + \vec{m}_{I,J+1} - 4\vec{m}_{I,J}}{a^2} \right) \tag{3.9}$$

3.2.1.2 Calcul du champ de démagnétisation

Le champ de démagnétisation est un champ créé par les interactions dipôle-dipôle des aimants élémentaires. Il est l'expression de l'énergie de l'aimantation dans un champ magnétique créé par l'aimantation elle-même. Rappelons que celui-ci est donné par l'expression suivante :

$$\vec{H}_{demag}(\vec{r}) = M_s \int_\Omega \bar{\bar{N}}(\vec{r} - \vec{r'}) \vec{m}(\vec{r'}) d\vec{r'} \tag{3.10}$$

Cette expression n'est rien d'autre que la convolution du tenseur de démagnétisation et de l'aimantation. Son calcul peut être assez coûteux mais il est intéressant de remarquer qu'il est possible

de simplifier son expression. Considérons tout d'abord la transformation de Fourier du champ de démagnétisation :

$$\mathcal{F}_{\vec{\omega}}\left[\vec{H}_{dem}(\vec{r})\right] = M_s \mathcal{F}_{\vec{\omega}}\left[\bar{\bar{N}}(\vec{r}) * \vec{m}(\vec{r})\right] \tag{3.11}$$

Nous savons que la transformée de Fourier d'une convolution est le produit des transformées de Fourier. Ainsi :

$$\mathcal{F}_{\vec{\omega}}\left[\vec{H}_{dem}(\vec{r})\right] = M_s \mathcal{F}_{\vec{\omega}}\left[\bar{\bar{N}}(\vec{r})\right] \mathcal{F}_{\vec{\omega}}\left[\vec{m}(\vec{r})\right] \tag{3.12}$$

Sachant que nous avons une expression analytique pour $\bar{\bar{N}}(\vec{r})$, nous pouvons calculer sa transformée de Fourier sans avoir recours au calcul numérique. En effet, nous savons que :

$$\bar{\bar{N}}(\vec{r}) = -\frac{1}{4\pi}\vec{\nabla} \otimes \vec{\nabla}' \frac{1}{|\vec{r}|} \tag{3.13}$$

De plus, nous avons la relation, par le théorème de la divergence, que $\mathcal{F}_{\vec{\omega}}\left[\vec{\nabla} f\right] = \vec{\omega}\mathcal{F}_{\vec{\omega}}\left[f\right]$. Ainsi, nous pouvons écrire :

$$\mathcal{F}_{\vec{\omega}}\left[\bar{\bar{N}}(\vec{r})\right] = -\frac{1}{4\pi}\mathcal{F}_{\vec{\omega}}\left[\frac{1}{|\vec{r}|}\right] \vec{\omega} \otimes \vec{\nabla}' \tag{3.14}$$

La transformée de Fourier de l'inverse de la norme de $\vec{r}$ peut se trouver en utilisant le théorème des résidus, en considérant une intégrale de quatrième type :

$$\mathcal{F}_{\vec{\omega}}\left[\frac{1}{|\vec{r}|}\right] = -\frac{2\ln\left(|\vec{\omega}|\right) - 2\gamma\mu_0}{\sqrt{2\pi}} \tag{3.15}$$

avec $\gamma \approx 0.577216$, la constante d'Euler-Mascheroni. Ainsi, nous obtenons :

$$\tilde{N}(\vec{\omega}) \equiv \mathcal{F}_{\vec{\omega}}\left[\bar{\bar{N}}(\vec{r})\right] = \frac{1}{\sqrt{8\pi^3}}\left(\ln(|\vec{\omega}|) - \gamma\mu_0\right) \vec{\omega} \otimes \vec{\nabla}' \tag{3.16}$$

Afin de trouver la valeur de $\vec{H}_{dem}$ dans l'espace réel, il nous suffit de prendre la transformée inverse du produit de $\tilde{m}$ et $\tilde{N}$. Sachant que nous n'avons pas d'expression analytique exacte de l'aimantation $\vec{m}$, nous sommes contraints d'utiliser une approche numérique afin de calculer sa transformée de Fourier, et prendre la transformée de Fourier inverse de l'ensemble.

Le langage de programmation *Python* possède une librairie nommée *NumPy* permettant la manipulation d'objets mathématiques divers et variés comme des vecteurs, des tenseurs, etc. Cette librairie permet le calcul des transformées de Fourier discrètes (voir *https://docs.scipy.org/doc/numpy/reference/routines.fft.html*). Il est même possible d'utiliser une fonction permettant d'optimiser les données en entrée afin d'effectuer des transformées de Fourier rapides (voir *https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft.html*).

3.2.2 Vérification de l'intégrateur

Une étape essentielle lors d'une approche numérique d'un problème est le test du programme que l'on développe. Pour cela, il suffit de considérer un cas classique et déjà traité, de prendre les mêmes paramètres de simulation, et comparer les différentes dynamiques obtenues. Si le programme développé fonctionne correctement, les dynamiques doivent en théorie se superposer.

Ainsi, notre première étape consiste à reprendre le code micromagnétique *70 lines of NumPy* et changer l'intégrateur d'Euler par un intégrateur de Runge-Kutta d'ordre 4. Bien que l'intégrateur d'Euler soit plus rapide, celui-ci n'est pas très stable et induit des erreurs dans les valeurs calculées qui peuvent rapidement diverger. L'intégrateur de Runge-Kutta d'ordre 4 nous permet d'avoir une meilleure approximation de la dynamique tout en conservant un temps de calcul raisonnablement faible.

3.2.2.1 Composantes et normes de l'aimantation

Afin de tester la validité de notre programme et de l'implémentation de notre intégrateur explicite de Runge-Kutta d'ordre 4, nous avons comparé la dynamique produite par *70 lines of NumPy* qui est en accord avec le problème n°4 de μmag (voir *https://www.ctcms.nist.gov/~rdm/mumag.org.html*).

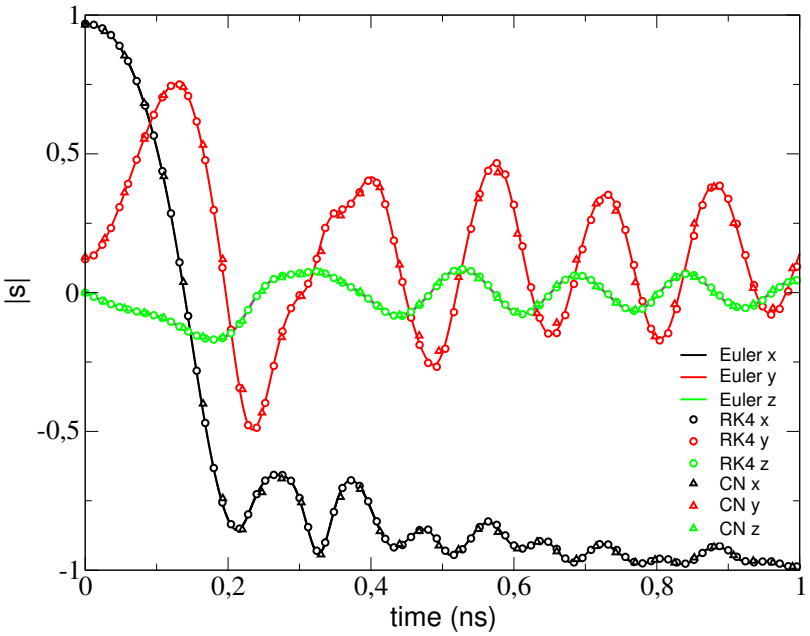


FIGURE 3.1 – Comparaison de la dynamique de Landau-Lifshitz-Gilbert de l'aimantation préparée dans un état s puis relaxée avec un coefficient $\alpha = 0.02$, pour un champ magnétique dirigé suivant un angle de 170°. Nous comparons la dynamique calculée par différents intégrateurs : Runge-Kutta explicite d'ordre 4, Euler explicite et Crank-Nicholson semi-implicite.

La dynamique produite par l'intégrateur semi-implicite de Crank-Nicholson (représentée par des triangles) a été effectuée par Gonçalo Albuquerque, Jaques Miltat et André Thiaville, en considérant des cellules ayant une taille de 2nm, alors que la dynamique produite par l'intégrateur d'Euler (représentée par un trait plein) et l'intégrateur explicite de Runge-Kutta d'ordre 4 (représentée par des cercles) portait sur des cellules de 5nm.

Nous observons que les dynamiques sont en accord sur la prédiction de la dynamique de Landau-Lifshitz-Gilbert, les quelques écarts provenant de la stabilité des différents intégrateurs numériques (la méthode semi-implicite de Crank-Nicholson étant beaucoup plus stable que les deux autres).

Ainsi, nous en concluons que l'intégrateur explicite de Runge-Kutta d'ordre 4 permet d'obtenir de très bons résultats et permet un bon compromis entre la précision de la méthode Crank-Nicholson et la vitesse de la méthode d'Euler, lors de la simulation de la dynamique de Landau-Lifshitz-Gilbert.

3.3 Code stochastique et dynamique de Landau-Lifshitz-Bloch sans interactions internes

La première étape validée, nous pouvons maintenant nous intéresser à des phénomènes plus complexes. Nous allons donc maintenant nous intéresser à la dynamique de Landau-Lifshitz-Bloch, qui est une généralisation de l'équation de Landau-Lifshitz-Gilbert en prenant en compte les effets de fluctuations thermiques liées à une connexion avec un bain thermique. Cette dynamique peut être résolue en intégrant simultanément 21 équations : 3 sur les composantes $\langle s_i \rangle$, 9 sur les composantes de $\langle \tilde{\omega}_i s_j \rangle$ et 9 sur les composantes $\langle s_i s_j \rangle$.

3.3.1 Généralités

A cause de la complexité des équations à intégrer, notre intégrateur se doit d'être assez stable. En effet, une erreur dans le calcul d'une seule composante se propagerait instantanément dans toutes les autres valeurs. De plus, l'intégration instantanée des 21 équations nécessite de revoir un minimum la structure de l'intégrateur que nous utilisons. En effet, pour une seule équation, l'intégrateur explicite de Runge-Kutta d'ordre 4 se lit :

Algorithm 1 Intégrateur explicite de Runge-Kutta d'ordre 4 pour une seule fonction

procedure RK4
    $N_{spins} \leftarrow$ Array[]
    $\vec{s}_{current} \leftarrow \vec{s}$
    while $t < t_{max}$ **do**
        for n dans N_{spins} **do**
            $\vec{a}_1 \leftarrow f(t, \vec{s}_n)$
            $\vec{a}_2 \leftarrow f(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_1)$
            $\vec{a}_3 \leftarrow f(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_2)$
            $\vec{a}_4 \leftarrow f(t + h, \vec{s}_n + h\vec{a}_3)$

            $\vec{s}_{current,n} \leftarrow \vec{s}_n + \frac{h}{6}\left(\vec{a}_1 + 2\vec{a}_2 + 2\vec{a}_3 + \vec{a}_4\right)$
        $\vec{s} \leftarrow \vec{s}_{current}$

Pour la dynamique de Landau-Lifshitz-Bloch, nous devons aussi intégrer les autres composantes des différentes fonctions de corrélations. Il est primordial de s'assurer que chaque composante est intégrée au même temps que les autres.

L'intégration simultanée de toutes les composantes s'effectue en considérant le calcul de celles-ci à chaque pas intermédiaire afin qu'elles soient réactualisées pour le calcul au pas suivant. Cette procédure permet d'intégrer un système d'équations couplées à l'aide d'un intégrateur explicite de Runge-Kutta d'ordre 4. Il convient de remarquer qu'ici, nous utilisons la notation vectorielle $\vec{\cdot}$ et tensorielle $\overline{\overline{\cdot}}$ afin de simplifier l'écriture. En réalité, nous devons effectuer des boucles supplémentaires afin de parcourir chaque composante.

Algorithm 2 Intégrateur explicite de Runge-Kutta d'ordre 4 pour plusieurs fonctions

procedure RK4
  $N_{spins} \leftarrow$ Array[]
  $\vec{s}_{current} \leftarrow \vec{s}$
  $\bar{\bar{\omega}}_{current} \leftarrow \bar{\bar{\omega}}$
  $\bar{\bar{S}}_{current} \leftarrow \bar{\bar{S}}$
  while $t < t_{max}$ **do**
    for n dans N_{spins} **do**
      $\vec{a}_1 \leftarrow f(t, \vec{s}_n, \bar{\omega}_n, \bar{\bar{S}}_n)$
      $\bar{\bar{b}}_1 \leftarrow g(t, \vec{s}_n, \bar{\omega}_n, \bar{\bar{S}}_n)$
      $\bar{\bar{c}}_1 \leftarrow h(t, \vec{s}_n, \bar{\omega}_n, \bar{\bar{S}}_n)$

      $\vec{a}_2 \leftarrow f(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_1, \bar{\omega}_n + \frac{h}{2}\bar{\bar{b}}_1, \bar{\bar{S}}_n + \frac{h}{2}\bar{\bar{c}}_1)$
      $\bar{\bar{b}}_2 \leftarrow g(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_1, \bar{\omega}_n + \frac{h}{2}\bar{\bar{b}}_1, \bar{\bar{S}}_n + \frac{h}{2}\bar{\bar{c}}_1)$
      $\bar{\bar{c}}_2 \leftarrow h(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_1, \bar{\omega}_n + \frac{h}{2}\bar{\bar{b}}_1, \bar{\bar{S}}_n + \frac{h}{2}\bar{\bar{c}}_1)$

      $\vec{a}_3 \leftarrow f(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_2, \bar{\omega}_n + \frac{h}{2}\bar{\bar{b}}_2, \bar{\bar{S}}_n + \frac{h}{2}\bar{\bar{c}}_2)$
      $\bar{\bar{b}}_3 \leftarrow g(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_2, \bar{\omega}_n + \frac{h}{2}\bar{\bar{b}}_2, \bar{\bar{S}}_n + \frac{h}{2}\bar{\bar{c}}_2)$
      $\bar{\bar{c}}_3 \leftarrow h(t + \frac{h}{2}, \vec{s}_n + \frac{h}{2}\vec{a}_2, \bar{\omega}_n + \frac{h}{2}\bar{\bar{b}}_2, \bar{\bar{S}}_n + \frac{h}{2}\bar{\bar{c}}_2)$

      $\vec{a}_4 \leftarrow f(t + h, \vec{s}_n + h\vec{a}_3, \bar{\omega}_n + h\bar{\bar{b}}_3, \bar{\bar{S}}_n + h\bar{\bar{c}}_3)$
      $\bar{\bar{b}}_4 \leftarrow g(t + h, \vec{s}_n + h\vec{a}_3, \bar{\omega}_n + h\bar{\bar{b}}_3, \bar{\bar{S}}_n + h\bar{\bar{c}}_3)$
      $\bar{\bar{c}}_4 \leftarrow h(t + h, \vec{s}_n + h\vec{a}_3, \bar{\omega}_n + h\bar{\bar{b}}_3, \bar{\bar{S}}_n + h\bar{\bar{c}}_3)$

      $\vec{s}_{current,n} \leftarrow \vec{s}_n + \frac{h}{6}\left(\vec{a}_1 + 2\vec{a}_2 + 2\vec{a}_3 + \vec{a}_4\right)$
      $\bar{\omega}_{current,n} \leftarrow \bar{\omega}_n + \frac{h}{6}\left(\bar{\bar{b}}_1 + 2\bar{\bar{b}}_2 + 2\bar{\bar{b}}_3 + \bar{\bar{b}}_4\right)$
      $\bar{\bar{S}}_{current,n} \leftarrow \bar{\bar{S}}_n + \frac{h}{6}\left(\bar{\bar{c}}_1 + 2\bar{\bar{c}}_2 + 2\bar{\bar{c}}_3 + \bar{\bar{c}}_4\right)$
    $\vec{s} \leftarrow \vec{s}_{current}$
    $\bar{\bar{\omega}} \leftarrow \bar{\bar{\omega}}_{current}$
    $\bar{\bar{S}} \leftarrow \bar{\bar{S}}_{current}$

3.3.2 Vérification de l'intégrateur

Comme pour le cas du code micromagnétique, nous devons vérifier que notre intégrateur intègre le système comme il se doit. Comme la dynamique de Landau-Lifshitz-Bloch est une nouvelle hiérarchie d'équations et qu'il n'y a pas réellement d'éléments de comparaison avec les différents articles traitant du sujet, nous allons intégrer le système à l'aide du logiciel de calcul formel *Maple*.

Nous n'avons pas de réel moyen de vérifier que *Maple* lui aussi intègre de façon convenable le système mais nous partons du principe qu'il a été testé maintes fois et que sa fiabilité n'est plus à prouver. Partant de ceci, nous choisissons un jeu de paramètres et comparons la dynamique obtenue par *Maple* et par le code que nous avons développé.

Nous commençons par comparer les composantes et la norme de l'aimantation $\langle s_i \rangle$. La dynamique obtenue par notre programme est caractérisée par des traits pleins tandis que la dynamique obtenue par *Maple* est représentée par des cercles.

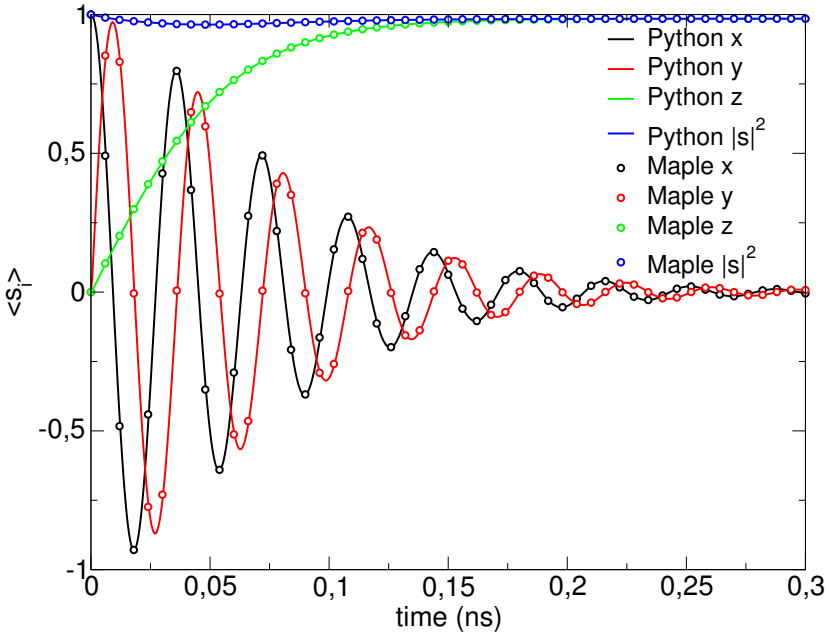
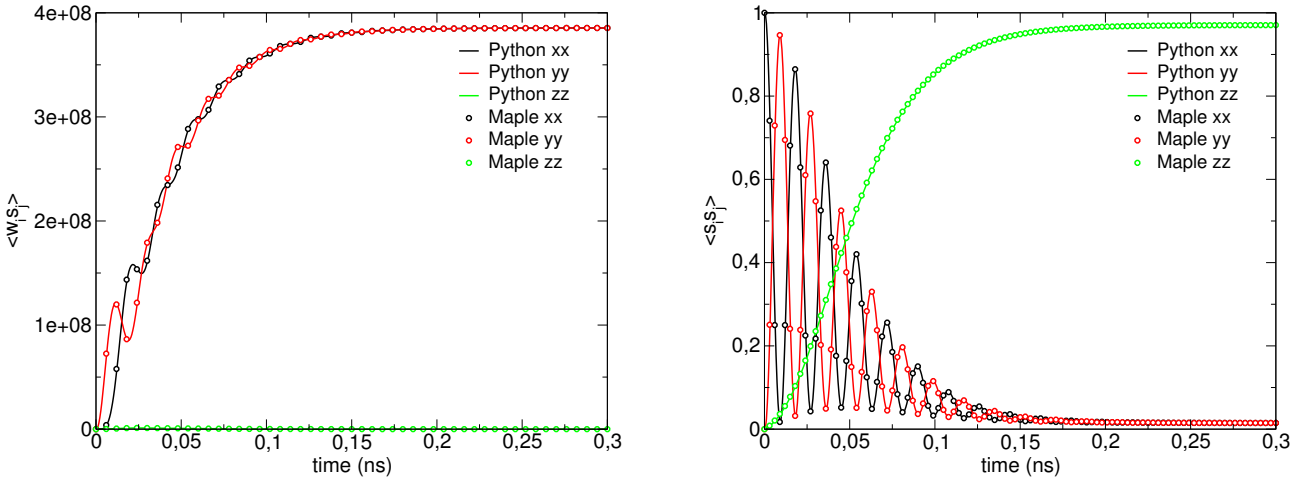


FIGURE 3.2 – Composantes et normes de l'aimantation $\langle s_i \rangle$



(a) Fonction de corrélation $\langle \tilde{\omega}_i s_j \rangle$

(b) Fonction de corrélation $\langle s_i s_j \rangle$

FIGURE 3.3 – Composantes diagonales des fonctions de corrélation

Nous remarquons que notre dynamique est en parfaite adéquation avec la dynamique produite par le logiciel *Maple*. Ainsi, notre programme, et plus particulièrement notre intégrateur, semble fonctionner correctement et calcule convenablement la dynamique de Landau-Lifshitz-Bloch.

3.4 Dynamique complète de Landau-Lifshitz-Bloch avec interactions internes

3.4.1 Généralités

Intéressons-nous maintenant à une généralisation de nos deux programmes précédents : le code micromagnétique avec la dynamique de Landau-Lifshitz-Gilbert, et le code atomistique avec la dynamique de Landau-Lifshitz-Bloch. Bien qu'il suffise de changer l'équation de Landau-Lifshitz-Gilbert

par le système de Landau-Lifshitz-Bloch, un autre problème vient se dresser : les unités. En effet, les physiciens travaillant en micromagnétisme utilisent des unités comme les Ampères par mètre, les secondes et des constantes fondamentales comme μ_0 alors que notre dynamique de Landau-Lifshitz-Bloch utilise des unités beaucoup plus naturelles comme les radians par gigahertz, et des nanosecondes. De plus, le code micromagnétique utilise les champs alors que le code atomistique utilise des pulsations. La conversion des unités est donc notre première tâche dans cette généralisation.

3.4.2 Influence de la température

Nous avons donc un programme permettant d'effectuer des calculs micromagnétiques, mais en considérant la hiérarchie d'équations de Landau-Lifshitz-Bloch plutôt que l'équation de Landau-Lifshitz-Gilbert.

Alors que la plupart des chercheurs utilisent l'équation de Landau-Lifshitz-Gilbert et gèrent la température de façon stochastique, entraînant des coûts de calcul et de stockage en mémoire assez énormes, la dynamique de Landau-Lifshitz-Bloch a le bon goût de proposer une approche équivalente mais de façon déterministe, en considérant seulement deux équations supplémentaires caractérisant l'influence du bruit thermique.

Il a été montré que le formalisme développé permettait de gérer à la fois de faibles et forts temps de corrélation (approximation markovienne ou non-markovienne) [2] et la superposition de la dynamique générée avec des simulations stochastiques permet de montrer une équivalence du traitement du bruit thermique.

Nous cherchons maintenant à voir l'influence de la température sur l'aimantation moyenne au sein du matériau. Naïvement, nous nous attendons à obtenir une courbe de Curie, surtout lorsque nous construirons de grandes assemblées d'aimants élémentaires.

Lors de notre étude, nous avons fait varier la taille de notre système de 8 spins à 1000 spins. En plus cela, nous avons effectué pour plusieurs assemblées différentes des calculs avec une tolérance très faible, permettant d'obtenir des résultats avec une excellente précision. Nous avons remarqué que les résultats obtenus avec une tolérance très faible et ceux avec une tolérance plus élevée se superposaient. Ainsi, ceci conforte nos différents résultats produits.

Bien que nous nous attendions à obtenir une courbe de Curie, et voir que la taille du système avait une réelle importance sur la dynamique de l'aimantation en température finie, nous avons constaté que nos systèmes étaient trop petits pour accommoder des parois magnétiques. Ce sont ces parois qui permettent d'obtenir une dynamique produisant cette courbe. En effet, notre système le plus grand ne mesure qu'une cinquantaine de nanomètres et ces fameuses parois s'accommodent pour une taille autour du micromètre.

Nous remarquons donc que quelque soit la taille du système, l'aimantation décroît comme une exponentielle décroissante lorsque nous augmentons la température. De plus, les différentes courbes se superposent, mettant en avant le fait que notre système se comporte comme une particule chargée soumise à une certaine température. Nous sommes donc dans le régime paramagnétique et nous constatons bien que l'aimantation du système suit la loi de Langevin. Le fait que notre système soit foncièrement ferromagnétique mais qu'il décrive une dynamique paramagnétique le fait tomber dans le cas du **superparamagnétisme**.

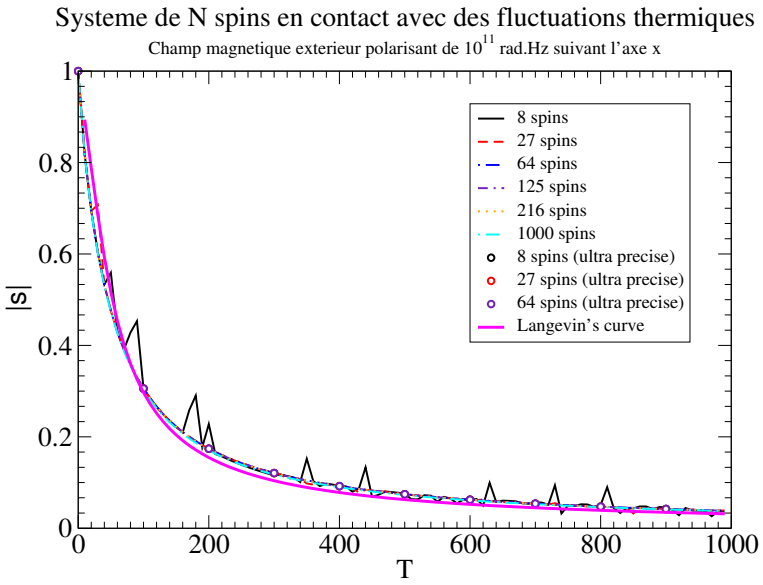


FIGURE 3.4 – Influence de la température sur l'aimantation pour différentes tailles de systèmes

Comme nous le remarquons sur ce graphique, la taille du système n'influence pas la dynamique de l'aimantation. En réalité, la courbe de Curie peut être obtenue lorsque nous considérons un système de taille infinie, ce qui est très loin de notre cas de "seulement" 1000 spins ! Il paraît donc assez naturel de tomber dans ce cas.

L'étude en température est donc très limitée sur nos systèmes car nous tombons systématiquement dans le même cas de figure. Afin d'obtenir des dynamiques plus intéressantes, nous devons réussir à stabiliser des structures magnétiques que nous pourrons ensuite étudier à température finie.

3.5 Dynamique d'un gaz de *Skyrmions*

3.5.1 Généralités

Les *Skyrmions* aussi appelés *vortex magnétiques* sont des structures topologiquement stables qui peuvent être obtenues en considérant l'influence d'un autre champ interne appelé **échange anti-symétrique de Dzyaloshinskii–Moriya**. Cette interaction est décrite par :

$$H_{DMI} = \sum_{\langle I,K \rangle} \vec{D}_{IK} \cdot \left(\vec{S}_I \times \vec{S}_K \right) \tag{3.17}$$

Avec $\vec{S}_I$ un vecteur unitaire sans dimension, pointant dans la même direction que l'aimantation, $\vec{D}_{ij}$ se décomposant comme $\vec{D}_{ij} = D\vec{r}_{ij}$, $\vec{r}_{ij}$ étant le vecteur unitaire reliant les sites i et j. La pulsation effective du champ peut être trouvée en considérant la relation suivante :

$$\vec{\omega}_{DMI,I} = -\frac{1}{\hbar}\frac{\delta H_{DMI}}{\delta \vec{S}_I} = -\frac{D}{\hbar}\sum_{\langle K \rangle} \vec{S}_K \times \vec{r}_{IK} \tag{3.18}$$

L'influence de l'interaction antisymétrique vient contrer à la fois l'interaction d'échange mais aussi le champ magnétique externe. Cette interaction permet d'expliquer le fait qu'à température nulle,

certains matériaux antiferromagnétique exhibent un moment magnétique non-nul. Cet effet est attribué à "l'inclinaison" (*Spin canting*) d'un petit angle des spins plutôt que d'être parfaitement parallèles.

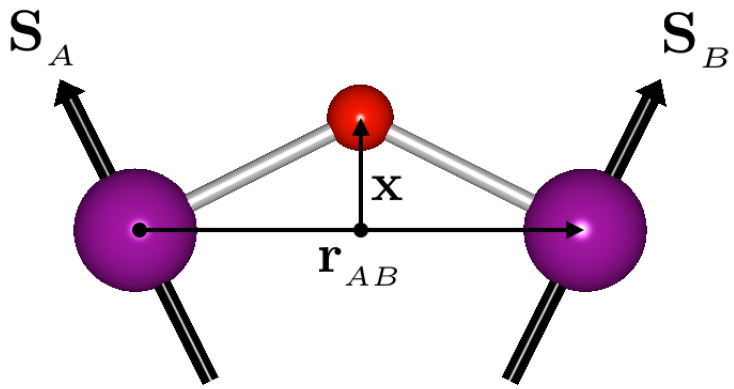


FIGURE 3.5 – *Spin canting* décrit par l'interaction antisymétrique (voir *https://upload.wikimedia.org/wikipedia/commons/1/1a/Dzyloshinskii_Moriya_antisymmetric_exchange.jpg*).

Le fait que l'alignement parallèle des spins n'est pas favorisé par cette interaction antisymétrique implique qu'il y aura une concurrence entre les deux champs internes : le champ d'échange favorise un alignement parallèle des spins alors que le champ DMI non. Cette compétition permet la création de structures magnétiques inattendues agissant chacune comme des particules.

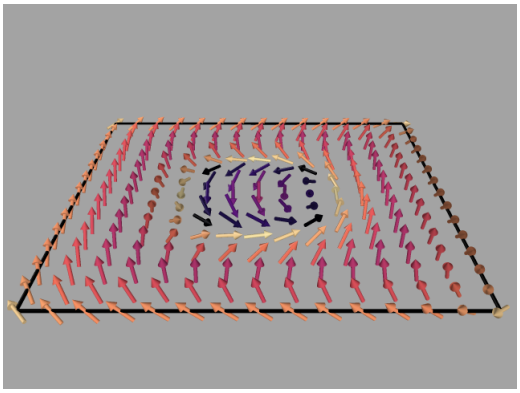


FIGURE 3.6 – Un vortex créé avec la dynamique de Landau-Lifshitz-Gilbert sur un réseau de 15×15 spins, initialement tous *down*, soumis à un champ magnétique suivant z de l'ordre de 10 T. Les champs internes pris en compte sont le champ d'échange avec $J = 10$ meV et le champ DMI avec $D = 8$ meV.

Sachant que chaque *skyrmion* agit comme une particule, il est intéressant de s'intéresser à leur dynamique lorsque nous soumettons notre système à une certaine température.

3.5.2 Influence de la température sur un gaz de *skyrmions*

Nous appelons "gaz de *skyrmions*" une assemblée de *skyrmions* stabilisés sur une structure carrée bidimensionnelle à température nulle. Afin de s'intéresser à la thermodynamique de ce gaz, il est intéressant d'avoir un indicateur permettant de quantifier le nombre de *skyrmions* présents dans

notre système. Ce nombre, aussi appelé "charge topologique" est définie par la relation suivante :

$$Q = \frac{1}{4\pi} \int_{\Omega} \vec{m} \cdot \left(\frac{\partial \vec{m}}{\partial x} \times \frac{\partial \vec{m}}{\partial y} \right) dxdy \tag{3.19}$$

Cette relation nous permet donc de compter le nombre de *skyrmions* apparaissant dans notre système. Afin de voir l'influence de la température sur ceux-ci, nous pouvons tracer la charge topologique Q en fonction de la température. Cependant, il est aussi intéressant de regarder l'influence du champ magnétique extérieur sur la stabilité des skyrmions lorsqu'ils sont soumis à une température finie.

3.5.2.1 Influence de la taille du système sur la température critique pour un champ magnétique constant

Intéressons-nous à l'influence de la taille du système sur la température critique. Contrairement aux matériaux ordinaires, les gaz de *skyrmions* sont des structures métastables très sensibles à la moindre variation d'intensité. Sachant que chaque *skyrmion* agit comme une particule, il doit y avoir une sorte d'interaction entre-eux et nous attendons des réponses différentes en fonction du nombre de *skyrmions* présents dans notre assemblée de spins, lorsque nous augmenterons progressivement la température.

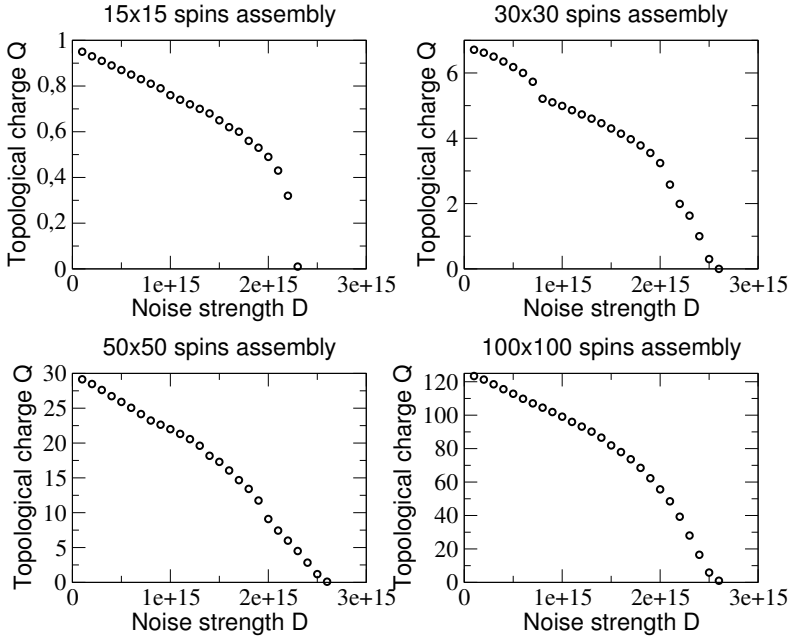


FIGURE 3.7 – Charge topologique totale d'assemblées de 15×15, 30×30, 50×50 et 100×100 spins en fonction de la température pour un champ magnétique externe constant de 25 T suivant l'axe z. L'énergie d'échange aux premiers voisins est de 10 meV et l'interaction antisymétrique aux premiers voisins est de 8 meV.

Après simulation, il apparaît que la taille du système importe peu sur la température critique, traduisant la disparition de la totalité des *skyrmions*. Ceci est sans doute dû au fait que notre rampe de température est assez importante que, nos systèmes étant assez petits, la différence doit être visible mais pour des variation plus faibles. Nous observons déjà un changement lorsque nous considérons une assemblée de 15×15 spins et une assemblée de 100×100 spins : la chute de la charge topologique totale est beaucoup moins abrupte pour une plus grande assemblée de spins. Il serait

intéressant de simuler une très grande assemblée (par exemple 500×500 voire 1000×1000) pour vérifier cette hypothèse. Cependant, nous nous limiterons à ces résultats dans ce mémoire.

Comme nous le verrons dans la partie suivante, nous travaillons avec un bruit gaussien coloré ayant un temps d'autocorrélation $\tau = 10^{-2}$ ns. L'influence de la température est directement liée à ce temps d'autocorrélation, qui lui même est lié à la couleur du bruit thermique, ainsi il est assez délicat de donner des températures critiques en Kelvins sachant que nous fixons arbitrairement l'intensité des interactions et ne nous basons pas réellement sur des constantes de matériaux connues. L'étude menée permet seulement d'estimer, en quelque sorte, l'influence des différents paramètres, mais pas de donner des chiffres précis. Ceux-ci pourront être obtenus en reprenant des intensités d'interactions mesurées.

Une première chose que nous pouvons remarquer est l'influence directe de la prise en compte de l'interaction antisymétrique : lorsque nous ne considérions que l'énergie d'échange et un champ magnétique externe, de par la petitesse de nos systèmes, nous tombions dans le cas du superparamagnétisme. Cependant, même si nous tracions l'aimantation en fonction de la température, et non pas la charge topologique, nous obtiendrions des courbes de Langevin car nous restons toujours dans la limite superparamagnétique, la forme ferromagnétique subsistant après la disparition des *skyrmions*, jusqu'à une température de transition où l'on constate l'apparition d'une phase désordonnée.

3.5.2.2 Influence du temps d'autocorrélation sur la température critique

Avant de se lancer dans les simulations, penchons-nous sur les équations de Landau-Lifshitz-Bloch et particulièrement sur l'équation gérant la connexion thermique. Bien que nous nous intéressons à un système d'équations couplées, seules les équations gouvernées par 3.56.b font intervenir la force du bruit D et le temps d'autocorrélation τ. En effet, pour rappel, celles-ci se lisent :

$$\begin{aligned}\frac{\partial}{\partial t}\langle\tilde{\omega}_i s_j\rangle = &-\frac{1}{\tau}\langle\tilde{\omega}_i s_j\rangle + \frac{1}{1+\lambda^2}\left[\epsilon_{jkl}\omega_k\langle\tilde{\omega}_j s_l\rangle + \epsilon_{jil}\frac{D}{\tau}\langle s_l\rangle\right] \\ &- \frac{\lambda}{1+\lambda^2}\left[\lambda\epsilon_{jkl}\epsilon_{kmn}\omega_m\left(\langle s_l\rangle\langle\tilde{\omega}_i s_n\rangle + \langle s_n\rangle\langle\tilde{\omega}_i s_l\rangle\right)\right]\end{aligned} \tag{3.20}$$

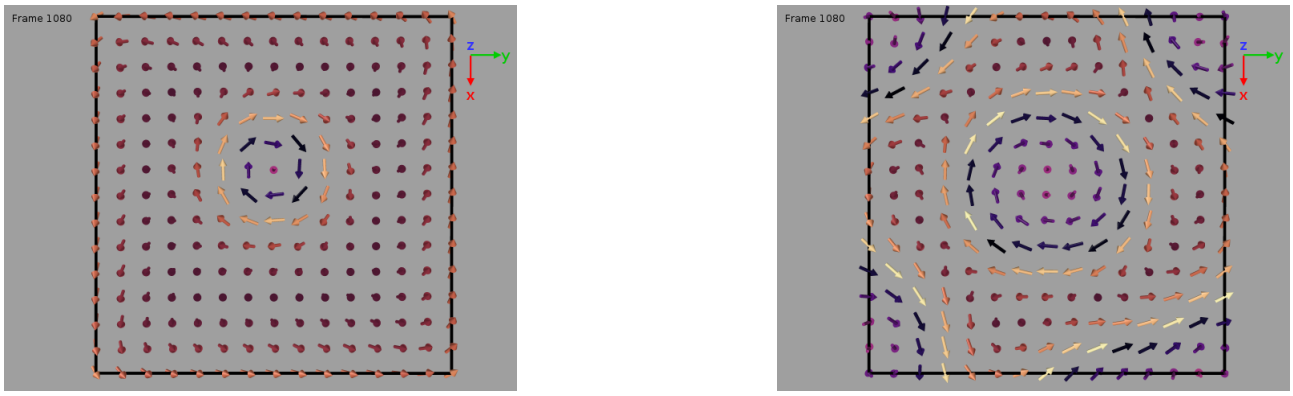
Nous remarquons que le facteur intervenant dans l'équation n'est pas D ni τ mais D/τ. Ainsi, le coefficient auquel il faut s'intéresser n'est pas directement τ mais D/τ car il est possible d'obtenir des coefficients égaux pour des D et τ différents.

Nous remarquons ainsi que le temps d'autocorrélation va jouer directement un rôle sur la force de la connexion thermique : un faible temps d'autocorrélation augmentera l'effet de la température sur le système et nous aurons des températures critiques beaucoup plus faibles alors que pour un fort temps d'autocorrélation la température critique sera beaucoup plus haute.

Le temps d'autocorrélation étant directement lié à la couleur du bruit, il convient de trouver celui qui est le plus adapté au problème. Étant donné que nous ne travaillons pas avec temps d'autocorrélation nul, le bruit utilisé est coloré (voir *https://en.wikipedia.org/wiki/Colors_of_noise*). Cependant, il est assez difficile de caractériser les différents paramètres du bruit et ne nous intéresserons pas aux différents procédés mis en œuvre pour les déterminer.

3.5.2.3 Influence du champ magnétique externe sur la température critique

Intéressons-nous maintenant à l'influence du champ magnétique externe sur la température critique. Avant de mettre en place une rampe de température dans nos simulations, nous avons pu constater que le champ magnétique permettait de contrôler la taille des *skyrmions*. Nous les stabilisons d'abord avec un champ magnétique de 25 T puis relaxons le système en le faisant varier. Nous pouvons ainsi observer les figures ci-dessous :



(a) Champ magnétique de 20 T

(b) Champ magnétique nul

FIGURE 3.8 – Taille d'un *skyrmion* en fonction du champ magnétique extérieur appliqué lors de la relaxation du système.

Il est alors intéressant de s'intéresser à la stabilité des *skyrmions* lorsque nous augmentons ou diminuons le champ magnétique extérieur lors de la relaxation du système et de la montée en température. Sachant que les *skyrmions* obtenus avec un champ magnétique plus faible ont une taille plus importante, il semble raisonnable de penser qu'ils seront moins affectés par la température. Afin de d'infirmer ou de confirmer notre hypothèse, nous nous concentrons sur une assemblée de 15×15 spins permettant d'accommoder exactement un *skyrmion*. Nous faisons ensuite varier le champ magnétique extérieur, appliqué lors de la relaxation, puis nous faisons varier la température.

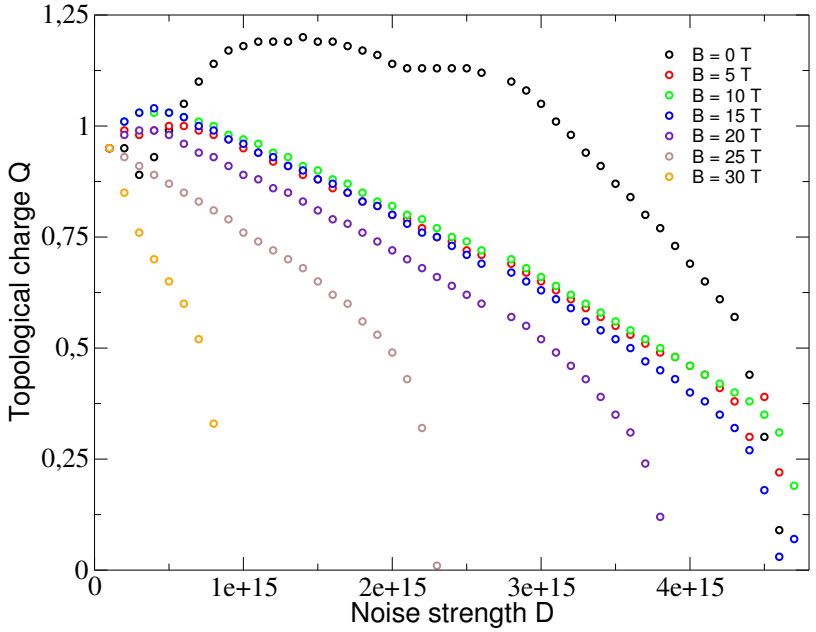


FIGURE 3.9 – Influence du champ magnétique appliqué lors de la relaxation sur la stabilité d'un *skyrmion*.

La figure ci-dessus montre qu'en effet, plus nous appliquons un champ magnétique faible, plus le *skyrmion* reste stable à haute température. Nous pouvons remarquer que les températures critiques sont quasiment les mêmes pour des champs magnétiques de 0 T, 5 T, 10 T et 15 T car le *skyrmion* résiste jusqu'à la température de transition du matériau, à partir de laquelle l'ordre ferromagnétique devient complètement désordonné à cause de la forte température. Nous pouvons donc imaginer qu'en prenant un matériau avec une température de transition plus haute, le *skyrmion* resterait stable à plus haute température lorsque nous considérons des champs magnétiques très faibles.

Afin de confirmer que nous nous intéressons à une transition de phase du second ordre, nous devons calculer l'exposant critique lié aux courbes tracées ci-dessus. Nous cherchons une courbe ayant l'allure suivante au voisinage de T_c :

$$Q(T) \propto \left(\frac{T_c - T}{T_c}\right)^{\beta} \tag{3.21}$$

En relevant la température critique T_c pour nos différentes courbes et en utilisant un logiciel permettant de trouver le coefficient β donnant une courbe la plus proche de notre simulation, nous pouvons obtenir le graphique suivant :

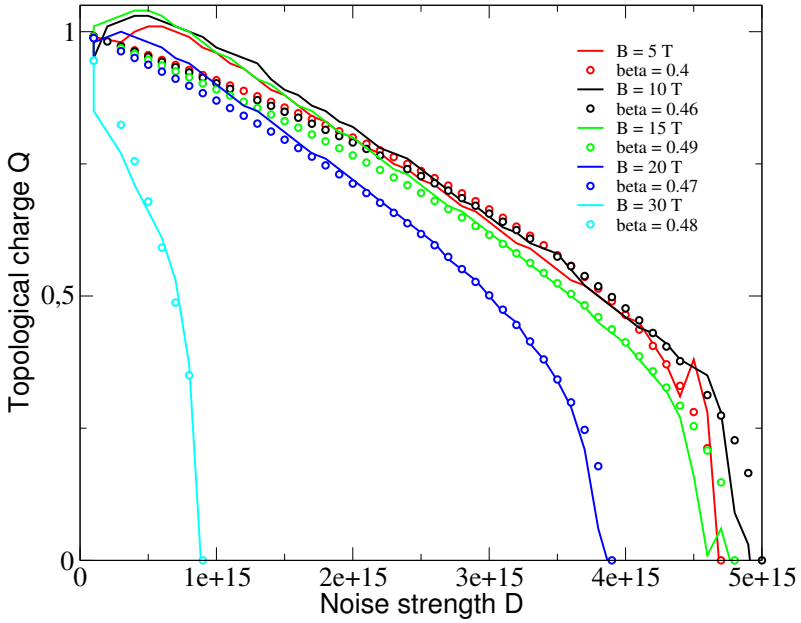


FIGURE 3.10 – Calcul de l'exposant critique en fonction du champ magnétique extérieur appliqué lors de la relaxation.

Nous relevons en moyenne un coefficient $\beta = 0.45 \pm 0.05$. Une bonne chose est que nous avons déjà le bon ordre de grandeur. Cependant, nous ne sommes pas exactement en accord avec le modèle d'Ising bidimensionnel qui prédit $\beta = 0.125$ mais sommes assez proche de la théorie de Landau qui prédit $\beta = 0.5$. La théorie de Landau étant une théorie de champ moyen, il est normal que nous nous approchions de celle-ci puisque nous avons fait une approximation afin d'obtenir l'équation de Landau-Lifshitz-Bloch : les champs internes et externes sont supposés décorrélés du spin et du bruit, entraînant une approximation proche de celle du champ moyen. Nos résultats sont donc cohérents et prouvent le fait que la connexion thermique est correctement gérée par la nouvelle hiérarchie d'équations de Landau-Lifshitz-Bloch, permettant l'étude de gaz de *skyrmions* à température finie.

Chapitre 4

Discussion

Dans ce mémoire, nous nous sommes intéressés dans un premier temps à la description micromagnétique de l'aimantation, avec un champ d'échange, un champ démagnétisant et un champ magnétique extérieur, en considérant l'équation de Landau-Lifshitz-Gilbert. Bien qu'un terme phénoménologique ait été ajouté par Gilbert afin de considérer des pertes liées à une connexion thermique, nous avons pu remarquer, en utilisant le formalisme des crochets de Nambu, que cette "perte" n'était en fait qu'un "décalage" dans l'espace des phases et ne traduisait pas réellement des pertes d'énergie (un indice flagrant était le fait que la norme de l'aimantation était conservée). Nous en avons profité pour changer l'intégrateur d'Euler par un intégrateur de Runge-Kutta d'ordre 4. Nous avons pu comparer les différentes dynamiques produites et prouver que notre intégrateur était correctement implémenté.

Nous nous sommes ensuite intéressés à une nouvelle hiérarchie d'équations, baptisée *dynamique de Landau-Lifshitz-Bloch*, qui permet de gérer convenablement la connexion thermique, tout en s'affranchissant du coût numérique du calcul stochastique. Nous avons commencé par nous intéresser aux cas où seul le champ magnétique externe était pris en compte, et avons comparé la dynamique obtenue avec la dynamique calculée par le logiciel de calcul formel *Maple*. Les dynamiques se superposant, nous en avons conclu que notre programme permettait d'intégrer correctement les équations de la dynamique de Landau-Lifshitz-Bloch. Nous avons ensuite inclus le calcul des champs micromagnétiques internes afin de tester l'influence de la température sur l'aimantation de notre système. Bien que nous nous attendions à obtenir une courbe de Curie, car notre matériau est ferromagnétique, nous sommes tombés dans le cas du superparamagnétisme à cause de la petitesse de notre système, et avons obtenu une courbe de Langevin, signe d'une dynamique paramagnétique.

Nous nous sommes donc tournés vers la simulation atomistique plutôt que micromagnétique et avons essayé d'accommoder des structures métastables à température nulle nommées *skyrmions*. Nous avons pris en compte une interaction interne supplémentaire : l'interaction antisymétrique de Dzyaloshinskii–Moriya qui favorise un petit décalage angulaire entre les spins voisins, et permet l'émergence de ces fameux *skyrmions*. Nous avons cherché dans un premier temps à les accommoder en utilisant l'équation de Landau-Lifshitz-Gilbert, ce qui était fait dans la grande majorité des articles s'intéressant aux *skyrmions*. Parce que l'équation de Landau-Lifshitz-Gilbert ne porte que sur trois composantes, sa résolution numérique est beaucoup plus rapide que la dynamique de Landau-Lifshitz-Bloch qui porte sur vingt-et-une composantes. Nous avons donc privilégié le développement de l'algorithme avec la dynamique la plus rapide à simuler afin de produire plus rapidement des résultats puis avons adapté notre code pour la dynamique la plus lente à simuler. Cependant, il n'était pas dit que nous allions pouvoir obtenir des *skyrmions* avec la dynamique de Landau-Lifshitz-Bloch, car même en température nulle, nous considérons la corrélation entre les spins et sa dynamique dans le temps, agissant comme un effet de d'amortissement supplémentaire, équivalent à l'amortissement produit par le terme de Gilbert.

Les *skyrmions* obtenus, nous avons pu tester leur stabilité en fonction de différents paramètres de simulation : force du bruit thermique, temps d'autocorrélation du bruit thermique, champ magnétique extérieur. Nous avons utilisé le jeu de paramètres suivant pour accommoder des *skyrmions* dans des plaquettes de 15×15, 30×30, 50×50 et 100×100 spins, à température nulle :

- Energie d'échange aux premiers voisins : $J = 10$ meV.
- Energie antisymétrique de Dzyaloshinskii–Moriya aux premiers voisins : $D = 8$ meV.
- Energie de Zeeman à chaque spin : $B = 25$ T.

Nous avons ensuite activé une rampe de température de 0.5×10^8 rad.GHz^2 afin de voir son influence sur la stabilité des *skyrmions*. Il convient de remarquer que cette rampe de température n'est pas du tout réalisable expérimentalement mais qu'elle convient parfaitement pour nos simulations. Malheureusement, nous n'avons pas eu de grosses surprises : nos assemblées restaient trop petites pour qu'il y ait une grosse variation au niveau de la température critique. Notre intérêt s'est alors porté sur le temps d'autocorrélation du bruit thermique. Nous voulions voir s'il avait une grande influence sur la température critique. Après une rapide étude des équations, il ressort que le coefficient réellement important est la force du bruit D divisé par le temps d'autocorrélation τ. Cependant, la caractérisation du type de bruit et de ses différents paramètres est hors de notre portée car beaucoup trop technique, nous nous sommes donc contentés de fixer la valeur de τ et de faire varier D. Ainsi, bien que nous ayons une relation explicite entre D et la température T, nous n'avons pas cherché immédiatement des valeurs de matériaux connus, pour reproduire des températures de transition de phases expérimentale. Nous avons d'ailleurs constaté que la diminution de plusieurs ordres de grandeur, des valeurs des constantes d'échange et d'anisotropie, n'avait pas directement d'impact sur la dynamique des *skyrmions*, mais simplement sur la durée de la simulation. C'est un phénomène attendu car la pulsation des champs est proportionnelle à la valeur de ces constantes.

Finalement, nous avons eu l'idée de stabiliser les *skyrmions*, comme nous le faisions précédemment, à température nulle et à champ magnétique extérieur constant. Puis, une fois les structures stabilisées, nous avons modifié l'intensité du champ magnétique et laisser le système relaxer vers son nouvel état d'équilibre. Nous avons pu remarquer que cette procédure permettait de contrôler la taille des *skyrmions* : plus le champ magnétique était faible, plus leur rayon était grand, et inversement, jusqu'à une valeur critique où le champ magnétique devient trop fort ($40 \sim 50$ T pour nos paramètres de simulation) pour que les *skyrmions* restent des solutions stables de notre dynamique. Une fois l'état d'équilibre relaxé obtenu, nous avons mis en place une rampe de température et avons vu une réelle différence : les *skyrmions* de plus grosse taille résistaient beaucoup mieux à la température et leur température critique devenait beaucoup plus grande. Avec nos paramètres de simulation, nous avons remarqué que les *skyrmions*, relaxés avec des champs magnétiques d'intensité nulle, de 5 T et 10 T, restaient stables jusqu'à la température de Curie du matériau ! Il est cependant certain qu'en utilisant d'autres paramètres de simulation, il aurait été possible de voir des différences de température critique même pour des champs magnétiques très faibles. Afin de confirmer le fait que nous nous intéressions bien à un phénomène critique, plus exactement à une transition de phase du second ordre, nous avons décidé de calculer, pour chaque dynamique obtenue après relaxation avec des champs différents, l'exposant critique β associé à l'équation $Q(D) \propto ((D_c - D)/D_c)^\beta$ avec D la force du bruit (proportionnelle à la température) et D_c la force critique du bruit (proportionnelle à la température critique). Le fait que nous obtenions $\beta_{dLLB} = 0.45 \pm 0.05$ est en bon accord avec le fait que nos champs, par approximation, soient décorrélés du spin et du bruit, et donc soient assimilable à un champ moyen, dont l'exposant critique est prédit par la théorie de Landau $\beta_{Landau} = 0.5$. La disparition des *skyrmions* constitue donc une transition de phase, la phase "skyrmionique" devenant une phase ordonnée à part entière.

Tirons une conclusion plus générale de notre étude : **la dynamique de Landau-Lifshitz-Bloch permet d'effectuer correctement la connexion thermique entre un système et un bain et les *skyrmions* émergent comme des solutions stables de cette dynamique.**

Chapitre 5

Pour aller plus loin

Bien que nous ayons vu une multitude de choses et tiré plusieurs résultats, il est tout à fait possible (et même nécessaire) d'aller encore plus loin. Pour cela, il est possible d'améliorer plusieurs points :

- **Interactions supplémentaires** : Bien que nous ayons pris en compte plusieurs interactions internes et externes, il est possible de raffiner notre étude et de considérer d'autres champs ainsi que d'autres phénomènes : champ démagnétisant, champ d'anisotropie magnétocristalline, champ électrique, transfert de spin, etc.

- **Raffinement du modèle de Landau-Lifshitz-Bloch** : Ce nouveau modèle semble être assez efficace pour prévoir la dynamique d'un système en interaction avec un bain thermique. Cependant, celui-ci repose sur l'approximation de fermeture gaussienne $\langle\langle a_i b_j c_k \rangle\rangle = 0$ et le fait que les champs soient décorrélés du spin et du bruit thermique $\langle \omega_i a_j \rangle = \omega_i \langle a_j \rangle$. Si nous voulons obtenir une dynamique plus fine, il suffit de ne pas considérer ces différentes approximations, ou d'autres d'ordre supérieur. La dynamique obtenue sera sûrement beaucoup plus complexe et portera sur beaucoup plus de composantes mais elle sera aussi moins approximative. Tout dépend des phénomènes que nous cherchons à modéliser, et la précision requise.

- **Accélérer, simplifier et automatiser les procédures** : Le temps de calcul a été le plus grand frein (avec la recherche de *bug*) lors de ce stage. Afin d'automatiser certaines tâches, il serait intéressant de créer des fichiers de configuration avec les différentes informations de simulation et d'avoir un algorithme permettant d'interpréter ce fichier et lancer la simulation adéquate. Cette simplification permettrait de gagner du temps dans la gestion des différentes simulations et permettrait même de faire plusieurs simulations à la suite, en spécifiant par exemple, plusieurs fichiers de configuration. Bien que ceci ne permette pas de gagner du temps de calcul, nous gagnons en temps de gestion. Pour le temps de calcul, la question est bien plus épineuse mais il existe une solution : le calcul en parallèle. Le fait que *Python* soit un langage de haut niveau permet un prototypage rapide mais de mauvaises performances. Il existe un compilateur nommé *Numba* permettant de compiler du *Python* afin d'exploiter le calcul sur GPU et CPU multicœurs (voir *https://devblogs.nvidia.com/numba-python-cuda-acceleration/*). L'idéal serait d'utiliser la puissance de CUDA (Compute Unified Device Architecture), ce qui est fait par *Spirit* qui permet de faire de la simulation atomistique (voir *https://spirit-code.github.io/*).

- **Structures 3D** : Notre étude était basée entièrement sur des structures 2D, représentant de très fines couches de matériaux. Cependant, la troisième dimension spatiale doit être prise en compte et doit sûrement avoir une influence sur la dynamique de l'aimantation. De multiples questions se posent : les *skyrmions* restent-ils des solutions stables en 3D ? Si oui, ont-ils une forme similaire ? Quels sont les nouveaux états stables et quelle est leur dynamique ?

Bibliographie

[1] Yoichiro Nambu
Generalized Hamilton Dynamics.
https://journals.aps.org/prd/pdf/10.1103/PhysRevD.7.2405

[2] J. Tranchida, P. Thibaudeau, and S. Nicolis
Closing the hierarchy for non-Markovian magnetization dynamics.
https://arxiv.org/pdf/1506.00544.pdf

[3] Claas Abert, Florian Bruckner, Christoph Vogler, Roman Windl, Raphael Thanhoffer, and Dieter Suess
A full-fledged micromagnetic code in less than 70 lines of NumPy.
https://arxiv.org/pdf/1411.7188.pdf

[4] W. Döring
Mikromagnetismus.
https://link.springer.com/chapter/10.1007/978-3-642-46035-7_3

[5] A. Friedman
Mathematics in Industrial Problems.
https://link.springer.com/chapter/10.1007/978-1-4615-7405-7_17

[6] Alex Hubert, Rudolf Schäfer
Magnetic Domains.
https://link.springer.com/book/10.1007/978-3-540-85054-0

[7] M. Lakshmanan
The fascinating world of the Landau–Lifshitz–Gilbert equation : an overview.
https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2010.0319

[8] R. Wieser
Description of a dissipative quantum spin dynamics with a Landau-Lifshitz/Gilbert like damping and complete derivation of the classical Landau-Lifshitz equation.
https://doi.org/10.1140/epjb/e2015-50832-0

[9] R. L. Liboff, M. A. Porter
Energy absorption and dissipation in quantum systems.
https://www.sciencedirect.com/science/article/pii/S0167278904002404

[10] R. Wieser
Comparison of Quantum and Classical Relaxation in Spin Dynamics.
https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.110.147201

[11] R. Zwanzig
Memory Effects in Irreversible Thermodynamics.
https://journals.aps.org/pr/abstract/10.1103/PhysRev.124.983

[12] R. Zwanzig
Ensemble Method in the Theory of Irreversibility.
https://aip.scitation.org/doi/10.1063/1.1731409

[13] V.E.Shapiro, V.M.Loginov
"Formulae of differentiation" and their use for solving stochastic equations.
https://www.sciencedirect.com/science/article/pii/037843717890198X

[14] J. Tranchida, P. Thibaudeau, and S. Nicolis
Hierarchies of Landau-Lifshitz-Bloch equations for nanomagnets : A functional integral framework.
https://journals.aps.org/pre/abstract/10.1103/PhysRevE.98.042101

[15] U. Atxitia1, D. Hinzke and U. Nowak
Fundamentals and applications of the Landau–Lifshitz–Bloch equation.
https://iopscience.iop.org/article/10.1088/1361-6463/50/3/033003

[16] W. Magnus
On the exponential solution of differential equations for a linear operator.
https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160070404

[17] S. Blanes, F. Casas, J. A. Oteo, and J. Ros
The Magnus expansion and some of its applications.
https://www.sciencedirect.com/science/article/pii/S0370157308004092

[18] F. J. Dyson
The Radiation Theories of Tomonaga, Schwinger, and Feynman.
https://journals.aps.org/pr/abstract/10.1103/PhysRev.75.486

[19] C. Runge
Ueber die numerische Auflösung von Differentialgleichungen.
https://doi.org/10.1007/BF01446807

[20] A. Frydryszak
LAGRANGIAN MODELS OF PARTICLES WITH SPIN : THE FIRST SEVENTY YEARS.
https://arxiv.org/abs/hep-th/9601020

[21] N. G. Van Kampen
Stochastic Processes in Physics and Chemistry.
https://www.sciencedirect.com/book/9780444529657/stochastic-processes-in-physics-and-chemistry

Appendices

Annexe A

Code micromagnétique sLLG avec intégrateur de Runge-Kutta explicite d'ordre 4

```
1  # Based on "70 lines of Numpy" (Copyright (C) 2014 Claas Abert)
2  # Solving the sLLG equation : (1 + l^2)ds/dt = w x s + l * s x (w x s)
3
4
5  ########################
6  # Importing librairies #
7  ########################
8  import numpy as np
9  from math import asinh, atan, sqrt, pi, fmod, log10
10
11 ######################################
12 # Setup mesh and material constraints #
13 ######################################
14 n     = (100, 25, 1)
15 dx    = (5e-9, 5e-9, 3e-9)
16 gamma = 2.211e5
17 eps   = 1e-18 # Small number to avoid division by 0
18 mu0   = 4e-7 * pi
19 ms    = 8e5
20 A     = 1.3e-11
21
22 ############################
23 # Defining Newell functions #
24 ############################
25 def f(p):
26   x, y, z = abs(p[0]), abs(p[1]), abs(p[2])
27   return + y / 2.0 * (z**2 - x**2) * asinh(y / (sqrt(x**2 + z**2) + eps)) \
28          + z / 2.0 * (y**2 - x**2) * asinh(z / (sqrt(x**2 + y**2) + eps)) \
29          - x*y*z * atan(y*z / (x * sqrt(x**2 + y**2 + z**2) + eps))       \
30          + 1.0 / 6.0 * (2*x**2 - y**2 - z**2) * sqrt(x**2 + y**2 + z**2)
31
32 def g(p):
33   x, y, z = p[0], p[1], abs(p[2])
34   return + x*y*z * asinh(z / (sqrt(x**2 + y**2) + eps))
       \
35          + y / 6.0 * (3.0 * z**2 - y**2) * asinh(x / (sqrt(y**2 + z**2) + eps))
              \
36          + x / 6.0 * (3.0 * z**2 - x**2) * asinh(y / (sqrt(x**2 + z**2) + eps))
              \
```

```
37          - z**3 / 6.0 * atan(x*y / (z * sqrt(x**2 + y**2 + z**2) + eps))
                \
38          - z * y**2 / 2.0 * atan(x*z / (y * sqrt(x**2 + y**2 + z**2) + eps))
                \
39          - z * x**2 / 2.0 * atan(y*z / (x * sqrt(x**2 + y**2 + z**2) + eps))
                \
40          - x*y * sqrt(x**2 + y**2 + z**2) / 3.0
41
42  ################################
43  # Setup demagnetization tensor #
44  ################################
45
46  def set_n_demag(c, permute, func):
47    print "Setup magnetization tensor"
48    it = np.nditer(n_demag[:,:,:,c], flags=['multi_index'], op_flags=['writeonly'
        ])
49    while not it.finished:
50      value = 0.0
51      for i in np.rollaxis(np.indices((2,)*6), 0, 7).reshape(64, 6):
52        idx = map(lambda k: (it.multi_index[k] + n[k]) % (2*n[k]) - n[k], range(3)
            )
53        value += (-1)**sum(i) * func(map(lambda j: (idx[j] + i[j] - i[j+3]) * dx[j
            ], permute))
54      it[0] = - value / (4 * pi * np.prod(dx))
55      it.iternext()
56
57  n_demag = np.zeros([1 if i==1 else 2*i for i in n] + [6])
58  for i, t in enumerate(((f,0,1,2),(g,0,1,2),(g,0,2,1),(f,1,2,0),(g,1,2,0),(f
        ,2,0,1))):
59    set_n_demag(i, t[1:], t[0])
60
61  m_pad     = np.zeros([1 if i==1 else 2*i for i in n] + [3])
62  f_n_demag = np.fft.rfftn(n_demag, axes = filter(lambda i: n[i] > 1, range(3)))
63
64  ########################################################################
65  # Computing effective field (Demagnetization field + Exchange field) #
66  ########################################################################
67  def h_eff(m):
68    # demag field
69    m_pad[:n[0],:n[1],:n[2],:] = m
70    f_m_pad = np.fft.rfftn(m_pad, axes = filter(lambda i: n[i] > 1, range(3)))
71    f_h_demag_pad = np.zeros(f_m_pad.shape, dtype=f_m_pad.dtype)
72    f_h_demag_pad[:,:,:,0] = (f_n_demag[:,:,:,(0, 1, 2)]*f_m_pad).sum(axis = 3)
73    f_h_demag_pad[:,:,:,1] = (f_n_demag[:,:,:,(1, 3, 4)]*f_m_pad).sum(axis = 3)
74    f_h_demag_pad[:,:,:,2] = (f_n_demag[:,:,:,(2, 4, 5)]*f_m_pad).sum(axis = 3)
75    h_demag = np.fft.irfftn(f_h_demag_pad, axes = filter(lambda i: n[i] > 1, range
          (3)))[:n[0],:n[1],:n[2],:]
76
77    # exchange field
78    h_ex = - 2 * m * sum([1/x**2 for x in dx])
79    for i in range(6):
80      h_ex += np.repeat(m, 1 if n[i%3] == 1 else [i/3*2] + [1]*(n[i%3]-2) + [2-i
          /3*2], axis = i%3) / dx[i%3]**2
81
82    h = ms * h_demag + 2 * A / (mu0 * ms) * h_ex
83
84    return h
85
86  ############################################################
87  # Computing the magnetization dynamics using an integrator #
```

```
88 ##########################################################
89
90 def F(m, h, gamma, alpha):
91     return gamma / (1 + alpha**2) * np.cross(h, m) + gamma * alpha / (1 + alpha
       **2) * np.cross(m, np.cross(h, m))
92
93 def RK4(m, min, max, gamma, alpha, step, write):
94     t = 0
95     dt = (max - min) / step
96     c = 0
97
98     if write == "yes":
99         with open('dynamics.dat', 'w') as f:
100            while t < max:
101                h = h_eff(m) + h_zee # Computing effective field depending on
                       the magnetization m
102
103                k1 = dt*F(m, h, gamma, alpha)
104                k2 = dt*F(m + k1 / 2, h, gamma, alpha)
105                k3 = dt*F(m + k2 / 2, h, gamma, alpha)
106                k4 = dt*F(m + k3, h, gamma, alpha)
107                m += (k1 + 2*k2 + 2*k3 + k4) / 6
108
109                m /= np.repeat(np.sqrt((m*m).sum(axis=3)), 3).reshape(m.shape)
110
111                if fmod(c, 100) == 0: # Printing 1 point for 100 points computed
112                    m_vec = map(lambda i: np.mean(m[:,:,:,i]), range(3))
113                    print "%.15f %f %f %f" % (t/max, m_vec[0], m_vec[1], m_vec
                           [2])
114                    f.write("%.15f %f %f %f\n" % (t/max, m_vec[0], m_vec[1],
                           m_vec[2]))
115
116                t += dt
117                c += 1
118    else:
119        while t < max:
120            h = h_eff(m) + h_zee # Computing effective field depending on the
                   magnetization m
121
122            k1 = dt*F(m, h, gamma, alpha)
123            k2 = dt*F(m + k1 / 2, h, gamma, alpha)
124            k3 = dt*F(m + k2 / 2, h, gamma, alpha)
125            k4 = dt*F(m + k3, h, gamma, alpha)
126            m += (k1 + 2*k2 + 2*k3 + k4) / 6
127
128            m /= np.repeat(np.sqrt((m*m).sum(axis=3)), 3).reshape(m.shape)
129
130            m_vec = map(lambda i: np.mean(m[:,:,:,i]), range(3))
131            print "%.15f %f %f %f" % (t/max, m_vec[0], m_vec[1], m_vec[2])
132
133            t += dt
134
135
136    return m
137
138 def sLLG(m, gamma, alpha, step, write):
139     min = 0
140     max = 1e-9
141
142     m = RK4(m, min, max, gamma, alpha, step, write)
```

```
143
144         return m
145
146     # Initial value of m
147     m = np.zeros(n + (3,))
148     m[1:-1,:,:,0]    = 1.0
149     m[(-1,0),:,:,1] = 1.0
150
151     # Settings magnetization in s-state
152     print "Computing s-state"
153     step = 5000
154     h_zee = np.zeros(n + (3,))
155     alpha = 1
156     m = sLLG(m, gamma, alpha, step, "no")
157
158     # Switching
159     print "Setting up Zeeman field"
160     step = 50000
161     h_zee = np.tile([-24.6e-3/mu0, +4.3e-3/mu0, 0.0], np.prod(n)).reshape(m.shape)
162     alpha = 0.02
163     m = sLLG(m, gamma, alpha, step, "yes")
```

Annexe B

Code micromagnétique dLLB avec intégrateur de Runge-Kutta explicite d'ordre 4 pour un champ constant

```
# Based on "70 lines of Numpy" (Copyright (C) 2014 Claas Abert)
# Solving the sLLG equation : (1 + l^2)ds/dt = w x s + l * s x (w x s)


########################
# Importing librairies #
########################
import numpy as np
from math import asinh, atan, sqrt, pi

#####################################
# Setup mesh and material constraints #
#####################################
n     = (1, 1, 1)
dx    = (5e-9, 5e-9, 3e-9)
gamma = 2.211e5
#gamma = 1
eps   = 1e-25 # Small number to avoid division by 0
mu0   = 4e-7 * pi
ms    = 8e5
A     = 1.3e-11
D     = 1e9 # 1 rad.GHz
tau   = 1e-11 # Weak correlation time

#############################################################
# Computing the magnetization dynamics using an integrator #
#############################################################

def computing_W(): # Computing <w x s> with w the random Gaussian variable
    value = np.zeros(n + (3,3,))
    value[:,:,:,1,2] = -D
    value[:,:,:,2,1] = D
    return value


def F(s, h, W, S, tau):
    value = np.zeros(3)
    for i in range(3):
        for k in range(3):
```

```
40              value[i] -= coeff * alpha * (h[k] * S[k][i] - h[i] * S[k][k])
41              for j in range(3):
42                  value[i] += coeff * (epsilon[i][j][k] * h[j] * s[k] + epsilon[i
                        ][j][k] * W[j][k])
43      return value
44
45  def G(s, h, W, S, tau):
46      value = np.zeros((3, 3))
47      for i in range(3):
48          for j in range(3):
49              value[i][j] -= W[i][j] / tau
50              for l in range(3):
51                  value[i][j] += coeff * D * epsilon[j][i][l] * s[l] / tau - coeff
                        * alpha * (h[l] * (s[l] * W[i][j] + s[j] * W[i][l]) - 2 * h[
                        j] * s[l] * W[i][l])
52                  for k in range(3):
53                      value[i][j] += coeff * (epsilon[j][k][l] * h[k] * W[i][l])
54      return value
55
56  def H(s, h, W, S, tau):
57      value = np.zeros((3, 3))
58      for i in range(3):
59          for j in range(3):
60              for l in range(3):
61                  value[i][j] -= coeff * alpha * (h[l] * (s[i] * S[l][j] + s[l] *
                        S[i][j] + s[j] * S[i][l] - 2 * s[i] * s[j] * s[l]) - h[j] * (
                        s[i] * S[l][l] + 2 * s[l] * S[i][l] - 2 * s[i] * s[l] * s[l])
                         + h[l] * (s[j] * S[l][i] + s[l] * S[j][i] + s[i] * S[j][l] -
                         2 * s[j] * s[i] * s[l]) - h[i] * (s[j] * S[l][l] + 2 * s[l]
                        * S[j][l] - 2 * s[j] * s[l] * s[l]))
62                  for k in range(3):
63                      value[i][j] += coeff * (epsilon[j][k][l] * h[k] * S[i][l] +
                            epsilon[j][k][l] * (s[i] * W[k][l] + s[l] * W[k][i]) +
                            epsilon[i][k][l] * h[k] * S[j][l] + epsilon[i][k][l] * (s
                            [j] * W[k][l] + s[l] * W[k][j]))
64      return value
65
66  def RK6(min, max, s, S, W, gamma, alpha, step):
67      t = 0
68      dt = (max - min) / step
69      c = 0
70      Q = 1e-1
71
72      # Computing W
73      W = computing_W()
74
75      with open('dllb_zeeman_rk4_s.dat', 'w') as f:
76          while t < max:
77              h = h_zee # Computing effective field depending on the magnetization
                    s
78
79              vec_s = map(lambda i: np.mean(s[:,:,:,i]), range(3))
80
81              s_sqr = sqrt(vec_s[0]**2 + vec_s[1]**2 + vec_s[2]**2)
82
83              print "%.8f\t%f\t%f\t%f\t%f" % (t*1e9, vec_s[0], vec_s[1], vec_s[2],
                    s_sqr)
84              f.write("%.8f\t%f\t%f\t%f\t%f\n" % (t*1e9, vec_s[0], vec_s[1], vec_s
                    [2], s_sqr))
85
```

```
86              for a in range(n[0]):
87                  for b in range(n[1]):
88                      for c in range(n[2]):
89                          a1 = dt * F(s[a][b][c], h[a][b][c], W[a][b][c], S[a][b][
                                c], tau)
90                          b1 = dt * G(s[a][b][c], h[a][b][c], W[a][b][c], S[a][b][
                                c], tau)
91                          c1 = dt * H(s[a][b][c], h[a][b][c], W[a][b][c], S[a][b][
                                c], tau)
92
93                          a2 = dt * F(s[a][b][c] + a1/2, h[a][b][c], W[a][b][c] +
                                b1/2, S[a][b][c] + c1/2, tau)
94                          b2 = dt * G(s[a][b][c] + a1/2, h[a][b][c], W[a][b][c] +
                                b1/2, S[a][b][c] + c1/2, tau)
95                          c2 = dt * H(s[a][b][c] + a1/2, h[a][b][c], W[a][b][c] +
                                b1/2, S[a][b][c] + c1/2, tau)
96
97                          a3 = dt * F(s[a][b][c] + a2/2, h[a][b][c], W[a][b][c] +
                                b2/2, S[a][b][c] + c2/2, tau)
98                          b3 = dt * G(s[a][b][c] + a2/2, h[a][b][c], W[a][b][c] +
                                b2/2, S[a][b][c] + c2/2, tau)
99                          c3 = dt * H(s[a][b][c] + a2/2, h[a][b][c], W[a][b][c] +
                                b2/2, S[a][b][c] + c2/2, tau)
100
101                         a4 = dt * F(s[a][b][c] + a3, h[a][b][c], W[a][b][c] + b3
                                , S[a][b][c] + c3, tau)
102                         b4 = dt * G(s[a][b][c] + a3, h[a][b][c], W[a][b][c] + b3
                                , S[a][b][c] + c3, tau)
103                         c4 = dt * H(s[a][b][c] + a3, h[a][b][c], W[a][b][c] + b3
                                , S[a][b][c] + c3, tau)
104
105                         s += (a1 + 2 * a2 + 2 * a3 + a4) / 6
106                         W += (b1 + 2 * b2 + 2 * b3 + b4) / 6
107                         S += (c1 + 2 * c2 + 2 * c3 + c4) / 6
108
109             # Variable step
110             vec_h = map(lambda i: np.mean(h[:,:,:,i]), range(3))
111             norm_h = sqrt(vec_h[0] * vec_h[0] + vec_h[1] * vec_h[1] + vec_h[2] *
                    vec_h[2])
112
113             dt = Q/norm_h
114
115             t += dt
116             c += 1
117
118     return s
119
120 # Initial values
121 s = np.zeros(n + (3,))
122 s[:,:,:,:] = 0
123 S = np.zeros(n + (3, 3,))
124 S[:,:,:,:,:] = 0
125 W = np.zeros(n + (3, 3,))
126 W[:,:,:,:,:] = 0
127 s[:,:,:,0]   = 1.0 # s = (1, 0, 0)
128 S[:,:,:,0,0] = 1.0
129
130 # Levi-Civita symbol
131 epsilon = np.zeros((3,3,3))
132
```

```
133 for a in range(3):
134   for b in range(3):
135     for c in range(3):
136       if [a,b,c] == [0, 1, 2] or [a,b,c] == [1,2,0] or [a,b,c] == [2,0,1]:
137         epsilon[a][b][c] = 1
138       elif a == b or a == c or b == c:
139         epsilon[a][b][c] = 0
140       else:
141         epsilon[a][b][c] = -1
142
143 # Switching
144 print "Setting up Zeeman field"
145 step = 1e3
146 h_zee = np.tile([0.0, 0.0, 176 * 1e9], np.prod(n)).reshape(s.shape) # h_zee =
        (0, 0, 176 rad.GHz)
147 alpha = 0.1
148 coeff = 1 / (1 + alpha * alpha)
149 s = RK6(0.0, 3e-10, s, S, W, gamma, alpha, step)
```

Annexe C

Calcul de la charge topologique des *skyrmions* en fonction de la température

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
import os
import numpy as np
from math import asinh, atan, sqrt, pi, sin, cos, floor, fmod, acos
import random
from scipy.spatial import Delaunay
#import numba

hbar_ev = 6.582119514 * 1e-7
hbar = 1.05457148 * 1e-34
kboltzmann = 1.380649 * 1e-23

sys.path.append(os.path.dirname(os.path.realpath("create_configuration.py")))
sys.path.append(os.path.dirname(os.path.realpath("load_configuration.py")))

import create_configuration
import load_configuration

# Creating configuration
filename_config = "configuration_15_15_1_strict_random.json"
filename_output = "influence_temperature_skyrmions_15_15_1.dat"
filename_output_charge = "charge_skyrmions_15_15_1.dat"

#create_configuration.create_configuration(500, 500, 1, "strict", "random",
    filename_config)
data = load_configuration.load_configuration(filename_config)

n      = (data[0], data[1], data[2])
m = np.asarray(data[3])
W = np.asarray(data[4])
S = np.asarray(data[5])

def compute_Q_2(s):
  Q = 0.0

  for a in range(n[0]):
    for b in range(n[1]):
      dx = 0.0
```

```
41        dy = 0.0
42
43        if a == 0:
44          dx = (s[a+1][b][0] - s[a][b][c])/(1.0)
45        elif a > 0 and a < n[0]-1:
46          dx = (s[a+1][b][0] - s[a-1][b][0])/(2.0)
47        else:
48          dx = (s[a-1][b][0] - s[a][b][c])/(1.0)
49
50        if b == 0:
51          dy = (s[a][b+1][0] - s[a][b][c])/(1.0)
52        elif b > 0 and b < n[1]-1:
53          dy = (s[a][b+1][0] - s[a][b-1][0])/(2.0)
54        else:
55          dy = (s[a][b-1][0] - s[a][b][c])/(1.0)
56
57        Q += np.dot(s[a][b][c], np.cross(dx, dy))
58
59    return -Q/(4 * pi)/0.667
60
61  # Computing topological charge
62  def compute_Q(s):
63
64    # Delaunay triangulation
65    points = []
66    A = 0.0
67
68    for a in range(n[0]):
69      for b in range(n[1]):
70        points.append([a + 1e-15 * random.random(), b + 1e-15 * random.random()])
71
72    points = np.array(points)
73
74    tri = Delaunay(points).simplices
75
76    for a in tri:
77      counter_clockwise = False
78      counter = 0
79      permutations = [
80        [a[0], a[1], a[2]],
81        [a[0], a[2], a[1]],
82        [a[1], a[0], a[2]],
83        [a[1], a[2], a[0]],
84        [a[2], a[0], a[1]],
85        [a[2], a[1], a[0]]
86      ]
87
88      pt = permutations[counter]
89
90      while(counter_clockwise == False and counter < 5):
91        rij = np.array([points[pt[1]][0] - points[pt[0]][0],points[pt[1]][1] -
              points[pt[0]][1], 0])
92
93        rik = np.array([points[pt[2]][0] - points[pt[0]][0],points[pt[2]][1] -
              points[pt[0]][1], 0])
94
95        r_normal = np.array([0, 0, 1])
96
97        if np.sign(np.dot(r_normal, np.cross(rij, rik))) == -1:
98          counter_clockwise = True
```

```
99          break
100
101       counter += 1
102       pt = permutations[counter]
103
104     ni = s[points[pt[0]][0]][points[pt[0]][1]][0]
105     nj = s[points[pt[1]][0]][points[pt[1]][1]][0]
106     nk = s[points[pt[2]][0]][points[pt[2]][1]][0]
107
108     A_tmp = ((1 + np.dot(ni, nj) + np.dot(ni, nk) + np.dot(nj, nk)) / sqrt(2 *
            (1 + np.dot(ni, nj)) * (1 + np.dot(ni, nk)) * (1 + np.dot(nj, nk))))
109
110     #A_tmp = ((1 + np.dot(ni, nj) + np.dot(ni, nk) + np.dot(nj, nk)) / sqrt(2 *
            (1 + np.dot(ni, nj)) * (1 + np.linalg.norm(ni) * np.linalg.norm(nj)) * (1
             + np.linalg.norm(ni) * np.linalg.norm(nk))))
111
112     if A_tmp > 1: A_tmp = 1
113     if A_tmp < -1: A_tmp = -1
114
115     A_tmp = 2 * np.sign(np.dot(ni, np.cross(nj, nk))) * acos(A_tmp)
116
117     A += A_tmp
118
119   return A / (4 * pi)
120
121 def h_eff(s, l):
122         h_ex_value = np.zeros(n + (3,))
123         J = 10 * 1e-3
124
125         for a in range(n[0]):
126             for b in range(n[1]):
127                 for c in range(n[2]):
128                     for i in range(len(l[a][b][c])):
129                         h_ex_value[a][b][c] += s[l[a][b][c][i][0]][l[a][b][c][i
                                ][1]][l[a][b][c][i][2]]
130
131         return 0.5 * 1e9 * J * h_ex_value / hbar_ev
132
133     # Computing Dzyaloshinskii-Moriya field
134 def h_dmi(s, l):
135     h_dmi_value = np.zeros(n + (3,))
136     k = 8 * 1e-3
137
138     for a in range(n[0]):
139         for b in range(n[1]):
140             for c in range(n[2]):
141                 for i in range(len(l[a][b][c])):
142                     aprime = l[a][b][c][i][0]
143                     bprime = l[a][b][c][i][1]
144                     cprime = l[a][b][c][i][2]
145                     r_ij = np.array([a - aprime, b - bprime, c - cprime])
146                     r_ij_norm = sqrt(r_ij[0]**2 + r_ij[1]**2 + r_ij[2]**2)
147                     r_ij = r_ij / r_ij_norm
148                     h_dmi_value[a][b][c] += np.cross(s[aprime][bprime][cprime],
                            r_ij)
149
150     return 0.5 * 1e9 * k * h_dmi_value / hbar_ev
151
152 def h_dd(s):
153   h_dd_value = np.zeros(n + (3,))
```

```
154   D = 0.5 * 1e-3
155
156   for a in range(n[0]):
157     for b in range(n[1]):
158       for aprime in range(n[0]):
159         for bprime in range(n[1]):
160           if(a != aprime and b != bprime):
161             r_ij = np.array([a - aprime, b - bprime, 0])
162             r_ij_norm = sqrt(r_ij[0]**2 + r_ij[1]**2 + r_ij[2]**2)
163             r_ij = r_ij / r_ij_norm
164
165             if r_ij_norm < 2:
166               h_dd_value[a][b][c] += (3 * r_ij * np.dot(s[aprime][bprime][0],
                      r_ij) - s[aprime][bprime][0]) / r_ij_norm**3
167
168   return -1e9 * D * h_dd_value / hbar_ev
169
170 def PC(a, b):
171   if a == n[0]: a = 0
172   if a == -1: a = n[0]-1
173   if b == n[1]: b = 0
174   if b == -1: b = n[1]-1
175
176   return [a,b]
177
178 def list_neighbors(condition, s):
179   neighbors = [[ [0 for col in range(n[2])] for col in range(n[1])] for row in
        range(n[0])]
180
181   if condition == "periodical":
182     for a in range(n[0]):
183       for b in range(n[1]):
184         current_neighbors = []
185
186         current_neighbors.append([PC(a-1,b)[0], PC(a-1,b)[1], 0])
187         current_neighbors.append([PC(a+1,b)[0], PC(a+1,b)[1], 0])
188         current_neighbors.append([PC(a,b-1)[0], PC(a,b-1)[1], 0])
189         current_neighbors.append([PC(a,b+1)[0], PC(a,b+1)[1], 0])
190
191         neighbors[a][b][c] = current_neighbors
192
193   if condition == "strict":
194     for a in range(n[0]):
195       for b in range(n[1]):
196         for c in range(n[2]):
197           current_neighbors = []
198           for i in range(-1, 2, 1):
199             for j in range(-1, 2, 1):
200               for k in range(-1, 2, 1):
201
202                 aprime = a + i
203                 bprime = b + j
204                 cprime = c + k
205
206                 distance = sqrt((a - aprime)**2 + (b - bprime)**2 + + (c -
                      cprime)**2)
207
208                 if distance < 1.1 and aprime >= 0 and aprime <= n[0]-1 and
                      bprime >= 0 and bprime <= n[1]-1 and cprime >= 0 and cprime
                      <= n[2]-1 and [a,b,c] != [aprime, bprime, cprime]:
```

```
209                     current_neighbors.append([aprime, bprime, cprime])
210             neighbors[a][b][c] = current_neighbors
211
212
213     return neighbors
214
215 def h_ani(s):
216     h_ani_value = np.zeros(n + (3,))
217     d = np.array([0.0, 0.0, 1.0])
218
219     K = 0.0 * 1e-3
220
221     for a in range(n[0]):
222         for b in range(n[1]):
223             h_ani_value[a][b][c] = np.dot(d, s[a][b][c])
224
225     return -0.5 * K * 1e9 * h_ani_value / hbar_ev
226
227 ############################################################
228 # Computing the magnetization dynamics using an integrator #
229 ############################################################
230
231 def F(s, h, W, S, tau):
232     value = np.zeros(3)
233     for i in range(3):
234         for k in range(3):
235             value[i] += coeff * alpha * (h[i] * S[k][k] - h[k] * S[k][i])
236             for j in range(3):
237                 value[i] += coeff * (epsilon[i][j][k] * h[j] * s[k] + epsilon[i
                        ][j][k] * W[j][k])
238     return value
239
240 def G(s, h, W, S, tau):
241     value = np.zeros((3, 3))
242     for i in range(3):
243         for j in range(3):
244             value[i][j] -= W[i][j] / tau
245             for l in range(3):
246                 value[i][j] += coeff * D * epsilon[j][i][l] * s[l] / tau - coeff
                        * alpha * (h[l] * (s[l] * W[i][j] + s[j] * W[i][l]) - 2 * h[
                        j] * s[l] * W[i][l])
247                 for k in range(3):
248                     value[i][j] += coeff * (epsilon[j][k][l] * h[k] * W[i][l])
249     return value
250
251 def H(s, h, W, S, tau):
252     value = np.zeros((3, 3))
253     for i in range(3):
254         for j in range(3):
255             for l in range(3):
256                 value[i][j] -= coeff * alpha * (h[l] * (s[i] * S[l][j] + s[l] *
                        S[i][j] + s[j] * S[i][l] - 2 * s[i] * s[j] * s[l]) - h[j] * (
                        s[i] * S[l][l] + 2 * s[l] * S[i][l] - 2 * s[i] * s[l] * s[l])
                         + h[l] * (s[j] * S[l][i] + s[l] * S[j][i] + s[i] * S[j][l] -
                         2 * s[j] * s[i] * s[l]) - h[i] * (s[j] * S[l][l] + 2 * s[l]
                        * S[j][l] - 2 * s[j] * s[l] * s[l]))
257                 for k in range(3):
258                     value[i][j] += coeff * (epsilon[j][k][l] * h[k] * S[i][l] +
                            epsilon[j][k][l] * (s[i] * W[k][l] + s[l] * W[k][i]) +
                            epsilon[i][k][l] * h[k] * S[j][l] + epsilon[i][k][l] * (s
```

```
                            [j] * W[k][l] + s[l] * W[k][j]))
259     return value
260
261 def RK4(m, new_m, min, max, gamma, step, l):
262     global D
263
264     t = 0
265     dt = (max - min) / step
266     counter = 0
267     E = 1e-5
268
269     # Printing initial configuration
270     m_vec = map(lambda i: np.mean(m[:,:,:,i]), range(3))
271     m_norm = sqrt(m_vec[0]**2 + m_vec[1]**2 + m_vec[2]**2)
272     with open(filename_output_charge, 'w') as f2, open(filename_output, 'w') as
            f:
273         #Q = compute_Q_2(m)
274
275         #f2.write(("%f %f %.2f\n") % (t/max, D, Q))
276
277         while t < max:
278             m = np.copy(new_m)
279             W = np.copy(new_W)
280             S = np.copy(new_S)
281
282             #Q = compute_Q_2(m)
283
284             h = h_eff(m, l) + h_dmi(m, l) + h_zee # Computing effective field
                    depending on the magnetization m
285
286             m_vec = map(lambda i: np.mean(m[:,:,:,i]), range(3))
287             m_norm = sqrt(m_vec[0]**2 + m_vec[1]**2 + m_vec[2]**2)
288
289             # Printing
290             print "%f %f %f %f %f %f" % (t/max, m_vec[0], m_vec[1], m_vec[2],
                    m_norm, D)
291             #print "%f %f %.2f" % (t/max, D, Q)
292
293
294             f.write("%i\n" % (n[0]*n[1]*n[2]))
295             f.write("Time=%f\n" % (t/max))
296
297
298             #f2.write(("%f %f %.2f\n") % (t/max, D, Q))
299
300
301             for a in range(n[0]):
302                 for b in range(n[1]):
303                   for c in range(n[2]):
304                     f.write("%i\t%i\t%i\t%f\t%f\t%f\t%f\n" % (a, b, c, m[a][b][c
                            ][0], m[a][b][c][1], m[a][b][c][2], D))
305
306
307             for a in range(n[0]):
308                 D = 1e15 * a / (n[0]-1)
309                 for b in range(n[1]):
310                     for c in range(n[2]):
311                       a1 = dt * F(m[a][b][c], h[a][b][c], W[a][b][c], S[a][b][c
                              ], tau)
312                       b1 = dt * G(m[a][b][c], h[a][b][c], W[a][b][c], S[a][b][c
```

```
                        ], tau)
313                     c1 = dt * H(m[a][b][c], h[a][b][c], W[a][b][c], S[a][b][c
                        ], tau)
314
315                     a2 = dt * F(m[a][b][c] + a1/2, h[a][b][c], W[a][b][c] + b1
                        /2, S[a][b][c] + c1/2, tau)
316                     b2 = dt * G(m[a][b][c] + a1/2, h[a][b][c], W[a][b][c] + b1
                        /2, S[a][b][c] + c1/2, tau)
317                     c2 = dt * H(m[a][b][c] + a1/2, h[a][b][c], W[a][b][c] + b1
                        /2, S[a][b][c] + c1/2, tau)
318
319                     a3 = dt * F(m[a][b][c] + a2/2, h[a][b][c], W[a][b][c] + b2
                        /2, S[a][b][c] + c2/2, tau)
320                     b3 = dt * G(m[a][b][c] + a2/2, h[a][b][c], W[a][b][c] + b2
                        /2, S[a][b][c] + c2/2, tau)
321                     c3 = dt * H(m[a][b][c] + a2/2, h[a][b][c], W[a][b][c] + b2
                        /2, S[a][b][c] + c2/2, tau)
322
323                     a4 = dt * F(m[a][b][c] + a3, h[a][b][c], W[a][b][c] + b3,
                        S[a][b][c] + c3, tau)
324                     b4 = dt * G(m[a][b][c] + a3, h[a][b][c], W[a][b][c] + b3,
                        S[a][b][c] + c3, tau)
325                     c4 = dt * H(m[a][b][c] + a3, h[a][b][c], W[a][b][c] + b3,
                        S[a][b][c] + c3, tau)
326
327                     new_m[a][b][c] += (a1 + 2 * a2 + 2 * a3 + a4) / 6
328                     new_W[a][b][c] += (b1 + 2 * b2 + 2 * b3 + b4) / 6
329                     new_S[a][b][c] += (c1 + 2 * c2 + 2 * c3 + c4) / 6
330
331             m_vec_new = map(lambda i: np.mean(new_m[:,:,:,i]), range(3))
332             m_norm_new = sqrt(m_vec_new[0]**2 + m_vec_new[1]**2 + m_vec_new
                    [2]**2)
333
334             print abs(m_norm_new - m_norm)
335             print counter
336
337             if counter == 200:
338                 break
339
340             h_vec = map(lambda i: np.mean(h[:,:,:,i]), range(3))
341             h_norm = sqrt(h_vec[0]**2 + h_vec[1]**2 + h_vec[2]**2)
342
343             dt = 1e-14
344
345             t += dt
346             counter += 1
347     return m
348
349 def dLLB(m, m_new, gamma, step, l):
350     min = 0
351     max = 1e-9
352
353     m = RK4(m, new_m, min, max, gamma, step, l)
354
355     return m
356
357 # Levi-Civita symbol
358 epsilon = np.zeros((3,3,3))
359
360 for a in range(3):
```

```
361     for b in range(3):
362       for c in range(3):
363         if [a,b,c] == [0, 1, 2] or [a,b,c] == [1,2,0] or [a,b,c] == [2,0,1]:
364           epsilon[a][b][c] = 1
365         elif a == b or a == c or b == c:
366           epsilon[a][b][c] = 0
367         else:
368           epsilon[a][b][c] = -1
369
370 # Parameters
371 tau = 1e-11
372 step = 1e5
373 alpha = 1.0
374 coeff = 1 / (1 + alpha * alpha)
375 D = 1e14
376
377 # Copying dynamics at t-1
378 new_m = np.copy(m)
379 new_W = np.copy(W)
380 new_S = np.copy(S)
381
382 l = list_neighbors("strict", m)
383
384 # Switching
385 E = 1e-5
386 print "Setting up Zeeman field"
387 h_zee = np.tile([0.0, 0.0, 176e9 * 0], np.prod(n)).reshape(m.shape)
388 m = dLLB(m, new_m, 0.0, step, l)
```