Devoir 2 : Chaînes de spin quantiques

PHQ404

Remise : -

1 Objectif

L'objectif de ce devoir est de déterminer le gap d'excitation dans une chaîne de spins en interaction selon le modèle de Heisenberg (voir la section 8D des notes de cours de David Sénéchal).

2 Construction du hamiltonien

Nous considérons le hamiltonien du modèle de Heisenberg pour une chaine périodique de N spins 1/2 définie par l'équation

$$H_N = J \sum_{k=0}^{N-1} \mathbf{S_k} \cdot \mathbf{S_{k+1}}. \tag{1}$$

où J est la constante de couplage et $\mathbf{S}_k = (S_k^x, S_k^y, S_k^z)$ est le spin associé à l'atome k. Comme la chaine est périodique, nous avons que $\mathbf{S_N} = \mathbf{S_0}$. La première tâche est d'implémenter une fonction qui permet de construire ce hamiltonien selon le nombre de spins N et la constante de couplage J. La matrice représentant le hamiltonien doit être creuse. Vous devez utiliser le module `sparse` de la bibliothèque `scipy` pour la construire. Notez que pour $N > 2$, le hamiltonien peut être construit à partir de H_2. Par exemple, nous avons

$$H_3 = H_2 \otimes I_2 + I_2 \otimes H_2 + J \sum_{p \in \{X,Y,Z\}} S^p \otimes I_2 \otimes S^p, \tag{2}$$

avec I_2 la matrice identité de dimension 2. Finalement, bien que les matrices de Pauli soient complexes, le hamiltonien est réel. Ainsi, la matrice retournée par votre fonction doit contenir des nombres réels (`float`).

Voir le fichier `hamiltonian.py` et `utils.py`.

3 Vérification du hamiltonien

Il est important de vérifer votre implémentation avec des tests unitaires. Vous pouvez, par exemple, calculer à la main la matrice du hamiltonien pour $N = 3$ et un J de votre choix et

de comparer cette dernière à celle retournée par votre fonction. Le résultat de votre calcul doit se retrouver dans votre rapport et dans vos tests unitaires.

Voir le fichier `test_hamiltonian.py`.

4 Calcul du gap d'excitation

Vous devez implémenter une fonction qui permet d'obtenir les énergies de l'état fondamental E_0 et du premier état excité E_1 à l'aide de la bibliothèque `scipy`. Vous devez aussi calculer l'énergie fondamentale en fonction du nombre de spins E_0/N ainsi que le gap d'excitation, c'est-à-dire la différence d'énergie entre le premier état excité et l'état fondamentale $\Delta = E_1 - E_0$. Ensuite, calculer ces quantités pour N allant de 2 à 20 inclusivement. Faites les calculs en fixant la constant de couplage $J = 1$. Comme le calcul peut prendre un certain temps, il est suggéré de sauvegarder vos résultats intermédiaires. Cela peut se faire à l'aide de la bibliothèque `numpy`, de la bibliothèque `pandas` ou manuellement.

Voir le fichier `hamiltonian.py` et `physical-quantities.py`.

5 Implémentation de l'algorithme epsilon

Vous devez ici implémenter l'algorithme epsilon. Pour ce faire, vous devez écrire une fonction qui prend en entrée une suite de valeurs S_n $(n = 0, 1, 2, \cdots, N)$ sous la forme d'un vecteur et qui retourne une estimation de la convergence de la série à l'aide de l'algorithme epsilon. L'algorithme est décrit à la section 8D.5 des notes de David Sénéchal.

Voir le fichier `epsilon_algorithm.py`.

6 Vérification de l'algorithme espsilon

La série de Gregory pour la fonction $\arctan(x)$ lorsque $x = 1$ est donnée par

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots\right). \tag{3}$$

Celle-ci permet ainsi d'évaluer d'évaluer la constante $\frac{\pi}{4}$. Vérifiez que votre implémentation de l'algorithme epsilon appliquée à cette série permet bel et bien d'approximer $\frac{\pi}{4}$. Essayez avec un nombre variable de termes pour mesurer la qualité de l'approximation. Ajouter vos tests unitaires dans le fichier `test_epsilon.py`.

Voir le fichier `series.py` et `test_epsilon_algorithm.py`.

7 Extrapolation vers $N \to \infty$

Vous devez utiliser l'algorithme epsilon pour estimer la valeur de E_0/N et de $\Delta = E_1 - E_0$ lorsque $N \to \infty$. Vous devez comparer l'extrapolation en utilisant tous les points (tous les N), seulement les points pour N pair (les N pairs) et seulement les points pour N impair (les N impairs). Faites ensuite un graphique pour chaque quantité avec les valeurs calculées et un trait horizontal pour chacune des trois méthodes d'extrapolation.

Toutes les fonctions qui doivent être implémentées sont déjà définies dans les fichiers et retournent des `NotImplementedError`.

Des exemples de vérification sont fournis, mais il vous appartient d'ajouter un nombre suffisant de tests unitaires pertinents ainsi que des tests de validation scientifique afin d'obtenir l'ensemble des points.

8 Évaluation

Catégorie	**Critère d'évaluation**	**Pondération**
Code (60%)	Qualité du code (pylint)	10%
	Couverture de tests (codecov)	10%
	Tests publics (pytest)	10%
	Tests cachés (pytest)	20%
	Documentation, docstrings et typage	5%
	README, reproductibilité et historique Git	5%
Rapport (40%)	Résumé	3%
	Introduction	3%
	Théorie	6%
	Résultats	6%
	Discussion	6%
	Conclusion	3%
	Références	3%
	Vérification des résults	5%
	Présentation	5%