Devoir 2 : Étude des chaînes de spin quantiques

PHQ404

Remise : 21 février 2024 à 23h45

1 Objectif

L'objectif de ce devoir est d'étudier les états de basse énergie du modèle de Heisenberg décrit à la section 8.4 des notes de David Sénéchal.

2 Comment présenter et remettre votre TP

Vous devez cloner le répertoire github dans l'organisation du cours au lien suivant : https://classroom.github.com/a/KuZsBMpG. Dans ce répertoire se trouvera votre code python, vos tests unitaires ainsi que votre rapport décrivant les méthodes utilisés et l'analyse de vos résultats. La structure des fichiers ne doit pas être modifiée, mais vous pouvez ajouter des fichiers si vous le désirez. Voici la structure de fichiers que votre répertoire devra garder :

Root
- → src
  - ↪ `fichier0.py`
  - ↪ `fichier1.py`
  - ↪ ...
- → tests
  - ↪ `test_fichier0.py`
  - ↪ `test_fichier1.py`
  - ↪ ...

↪ `.gitignore`
↪ `requirements.txt`
↪ `README.md`
↪ `rapport_devoir`*i*`-`*vos_noms*`.pdf`

Le fichier `requirements.txt` doit contenir les dépendances de votre projet. Le fichier `README.md` doit contenir les instructions pour installer et utiliser votre projet ainsi qu'une brève description du devoir et des méthodes utilisés dans le code. Le fichier `rapport_devoir`*i*`-`*vos_noms*`.pdf` doit contenir votre rapport en format pdf. L'utilisation de LaTeX est fortement recommandée pour la rédaction de votre rapport. Dans le dossier `src` se trouvera votre code python et dans le dossier `tests` se trouvera vos tests unitaires.

La remise et la correction automatique du code se fera à chaque `push` sur le répertoire github. Notez que seul le dernier `push` sera considéré pour la correction.

3 Énoncé

3.1 Construction du Hamiltonien

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

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

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

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

avec I 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`).

3.2 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 dans un fichier `test_hamiltonian.py`.

3.3 Calculs des basses énergies

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`. Ensuite, calculer ces énergies pour N de 2 à 20 inclusivement. Comme le calcul peut prendre un certain temps, il est suggéré de sauvegarder vos résultats intermédiaires. Faites les calculs en fixant la constant de couplage $J = 1$. Cela peut se faire à l'aide de la bibliothèque `numpy`, de la bibliothèque `pandas` ou manuellement.

3.4 Algorithme epsilon

Il s'agit ici d'implémenter une fonction qui prend en entrée un vecteur de points et retourne une estimation de la convergence de la série à l'aide de l'algorithme epsilon. L'algorithme est décrit à la section 8.5.4 des notes de David Sénéchal.

3.5 Vérification de l'algorithme espsilon

La série de Gregory

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

permet d'évaluer la constante π. Vérifiez que votre implémentation de l'algorithme epsilon appliquée à cette série permet bel et bien d'approximer π. Essayez avec un nombre variable de termes pour mesurer la qualité de l'approximation. Ajouter vos tests unitaires dans le fichier `test_epsilon.py`.

3.6 Extrapolation vers $N \to \infty$

Vous devez utiliser l'algorithme epsilon pour estimer la valeur de E_0/N et de $E_1 - E_0$ 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.

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

4 Vérification

Il est important de vérifer vos implémentations. En effet, vous devez vous assurer que vos méthodes fonctionnent correctement et pour ce faire, vous devez rouler et implémenter des tests unitaires qui testent chacune de vos classes et fonctions. De plus, vous devriez tester si les résultats obtenus sont logiques. Il serait aussi intéressant de retrouver vos vérifications dans votre rapport. Il est fortement recommandé d'ajouter des tests unitaires dans le dossier `tests`, mais les tests déjà implémentés ne doivent pas être modifiés.

5 Critères d'évaluation

70 points Pour le résultat de vos tests unitaires.

30 points Pour la qualité du rapport.