ÉCOLE
CENTRALE LYON

UE PRO tc2 2021-2022 – Stage Application

Simulation numérique d'équations de Schrödinger

Stage effectué du 19 avril 2022 au 12 août 2022 au Centre d'Enseignement et de Recherche en MathématIques et Calcul Scientifique (CERMICS) à l'École des Ponts et Chaussées ParisTech

École des Ponts
ParisTech

Auteur :
Matthias Personnaz (ECL20)

Encadrants :
Éric Cances (référent scientifique, ENPC)
Giacomo Casadei (référent pédagogique, ECL)

Version recompilée le
18 juin 2023

Résumé

Ce rapport décrit un stage effectué dans le cadre du cursus d'Ingénieur Généraliste de l'École Centrale de Lyon. Il présente la structure d'accueil, le travail réalisé, ainsi que les perspectives liées au processus de construction d'un projet professionnel. Le stage a été proposé par le tuteur scientifique et a consisté à tester numériquement et dans un cadre simplifié les limites de l'approximation de Born-Oppenheimer dans la simulation moléculaire. Cette approximation permet de découpler la dynamique des électrons de celle des noyaux atomiques, sans avoir à résoudre un problème bidimensionnel trop coûteux numériquement.

Pour atteindre cet objectif, la théorie des perturbations a été mise en œuvre en utilisant Julia, en justifiant chaque fois que possible les hypothèses mathématiques utilisées, leur cadre de validité, ainsi que les choix de conception. En utilisant la méthode employée, il a été démontré que la complexité asymptotique des perturbations sous certaines hypothèses est

$$\mathcal{O}\left(N\sqrt{E_{\max}(N)}\ln(E_{\max}(N))\right)$$

avec N la dimension de l'espace de Hilbert discret approchant le problème (taille de la grille) et $E_{\max}$ l'énergie maximale approchée sur cet espace. Les résultats numériques montrent que l'inégalité de Kato-Temple ne donne pas d'information utile. Enfin, en parallèle, en utilisant les bibliothèques de Julia, une heuristique en programmation dynamique a été mise au point pour alterner de manière optimale entre l'utilisation du CPU et du GPU. Ce travail de conception et de programmation pourra servir de base pour de futurs codes visant à évaluer de manière plus poussée les états propres de systèmes moléculaires complexes.

Abstract

This report describes an internship undertaken as part of the Engineering program at École Centrale de Lyon. The host structure, the work performed, and the perspectives related to the process of constructing a professional project are discussed. The internship was proposed by the scientific tutor and focused on numerically testing the limits of the Born-Oppenheimer approximation in molecular simulation. This approximation is used to decouple the dynamics of electrons from those of atomic nuclei in a simplified framework, without the need to solve a two-dimensional problem that would quickly become computationally intractable.

To achieve this goal, the theory of perturbations was implemented in Julia and the mathematical assumptions were justified as much as possible, along with their framework of validity and design choices. Using the chosen method, it was shown that the asymptotic complexity of the perturbations under certain assumptions is

$$\mathcal{O}\left(N\sqrt{E_{max}(N)}\ln(E_{max}(N))\right)$$

where N is the dimension of the discrete Hilbert space approximating the problem (grid size) and E_{max} is the maximum energy approximated on this space. Numerical results indicate that the Kato-Temple inequality does not provide any significant information. In addition, as a parallel project, a dynamic programming heuristic was developed using Julia's libraries that can be generalized to any computational algorithm to optimally alternate between CPU and GPU usage. This design and programming work can serve as a foundation for future codes to further evaluate the eigenstates of complex molecular systems.

Remerciements

Je remercie

- Éric Cances (professeur à l'ENPC, chercheur au CERMICS, tuteur scientifique) pour m'avoir accordé sa confiance en me sélectionnant pour ce stage, et expliqué ses aspects physiques et mathématiques, ainsi que pour ses indications à propos des méthodes préférentielles de l'état de l'art ;
- Paul Cazeaux (professeur et chercheur à l'université du Kansas) pour ses conseils pertinents permettant d'améliorer la complexité des algorithmes lors de notre rencontre à Roscoff ;
- Laurent Vidal (doctorant au CERMICS) pour ses conseils pratiques sur la programmation en Julia qui m'ont permis de gagner du temps au débuggage.

Dans ce document, les bibliothèques sont notées en rouge (comme `KrylovKit`), les classes en bleu (comme `CUDA.CUSPARSE`), et les fonctions en vert (par exemple, `eigvecs` ou `eigvals`).

Les codes qui ont servi à la génération des figures de ce rapport, ainsi que résoudre le problème posé en général, sont disponibles à l'adresse https://github.com/MatthiasPersonnaz/stage_CERMICS_BO.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Généralités en physique quantique, sous-jacents en simulation numérique et introduction du problème

1.1 Position du problème

Rappelons très succintement quelques éléments qui mettent en contexte le travail effectué lors de ce stage. La plupart de ce chapitre est fortement inspiré de, et résume certains des points importants de, [CTDL21b] et [CLBL21] et constitue un résumé de généralités en quantique[1] et sert à poser un cadre clair. Une partie importante du travail de ce stage a consisté à se familiariser avec, et étudier les fondements de la physique quantique au travers d'ouvrages de référence et d'articles jusqu'au niveau L3 environ, mais évidemment le but n'est pas ici de recopier ces cours et la littérature en entier, seulement d'en exposer les éléments les plus importants que j'ai dû m'approprier.

1.1.1 Fondamentaux succints sur le formalisme utilisé et le sens physique des simulations

Mécanique classique

En mécanique classique, le mouvement d'un système matériel quelconque peut se décrire notamment à l'aide du formalisme lagrangien qui met en jeu des coordonnées généralisées appelées *variables dynamiques fondamentales*, notées traditionnellement q_i. Le **lagrangien** $\mathcal{L}(q_i, \dot{q}_i, t)$ est une grandeur associée au système qui est la différence $E_c - E_p$. À ces coordonnées on associe également les *moments conjugués* $p_i = \frac{\partial \mathcal{L}}{\partial \dot{q}_i}$. Le **hamiltonien** donne l'énergie totale du système de points matériels $\mathcal{H}(p_i, q_i, t)$ et son évolution est dicté par les équations de Hamilton-Jacobi :

$$\begin{aligned} \frac{\mathrm{d}q_i}{\mathrm{d}t} &= +\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}p_i} \\ \frac{\mathrm{d}q_i}{\mathrm{d}t} &= -\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}q_i} \end{aligned} \tag{1.1}$$

Pour un point matériel seul, les trois variables dynamiques sont les trois coordonnées de

[1] Qui prolonge en particulier les notions effleurées dans les UEs PCM, GM et IDM du programme de l'ECL.

l'espace constituant le vecteur $\mathbf{r}$, et l'énergie totale du système s'écrit

$$\mathcal{H} = \frac{\mathbf{p}^2}{2m} + V(\mathbf{r}, t) \tag{1.2}$$

et les équations de Hamilton-Jacobi prennent la forme connue

$$\begin{array}{ll} \frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} & = \frac{\mathbf{p}}{m} \\ \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} & = -\nabla(V) \end{array} \tag{1.3}$$

Mécanique quantique

La meilleure théorie dont nous disposons qui puisse décrire le comportement fondamental de particules à des échelles très réduites est la physique quantique. À ces échelles, la mécanique classique ne fonctionne pas, mais certaines lois peuvent se vérifier en moyenne avec l'espérance de certains opérateurs. Avec les relations de De Broglie on associe à un corpuscule matériel d'énergie P et d'impulsion $\mathbf{p}$, une onde dont la pulsation $\omega = 2\pi\nu$ et le vecteur d'onde $\mathbf{k}$ sont donnés par les mêmes relations que pour les photons :

$$\left\{ \begin{array}{ll} E & = h\nu \quad = \hbar\omega \\ \mathbf{p} & = \hbar\mathbf{k} \end{array} \right. \tag{1.4}$$

Elle repose sur un formalisme et des postulats particuliers. Les postulats de la mécanique quantique stipulent notamment que :

- Un état quantique d'une particule à un instant donné est caractérisé par une fonction d'onde $|\psi(t)\rangle$ (un *ket* pour reprendre la notation de Dirac) qui appartient à un espace de Hilbert H (c'est-à-dire un espace préhilbertien complexe, muni d'un produit scalaire hermitien et de la norme associée), dont une facteur de phrase $e^{i\theta}$ n'affecte pas le sens physique. Cette fonction peut varier dans l'espace et le temps, on la note aussi $\psi(, t)$. En chaque endroit de l'espace accessible à la particule, $|\psi(\mathbf{x}, t)|^2$ est la densité de probabilité de présence de la particule.
- Toute grandeur physique d'intérêt, mesurable, est décrite par un opérateur $\mathcal{A}$ agissant dans l'espace des états : cet opérateur est appelé **observable**.
- Les valeurs permises pour cette grandeur physique sont les valeurs propres de l'opérateur $\mathcal{A}$, i.e. les nombres A tels que $\mathcal{A}|\psi\rangle = A|\psi\rangle$. Le spectre d'un observable peut également être continu ou discret. En outre la probabilité d'observation de chaque valeur n'est pas nécessairement également distribuée parmi toutes les valeurs permises.
- L'évolution de la fonction d'onde dans le temps est régie par l'équation de Schrödinger

$$\mathrm{i}\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}(t)|\psi(t)\rangle \tag{1.5}$$

  où $\hat{\mathcal{H}}$ est l'**opérateur hamiltonien**, associé à l'observable énergie totale du système et s'écrit

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\Delta + \hat{V}(\mathbf{r}) = \frac{\mathbf{p}^2}{2m}\Delta + \hat{V}(\mathbf{r}) \tag{1.6}$$

  où $\mathbf{p} = -\mathrm{i}\hbar\nabla$ est l'opérateur **impulsion**.

États stationnaire Lorsque le hamiltonien du système ne dépend pas du temps, une fonction d'onde de la forme

$$|\psi(\mathbf{r},t)\rangle = |\phi(\mathbf{r})\rangle\chi(t) \tag{1.7}$$

correspond à un état **stationnaire** de la fonction d'onde, de densité de probabilité indépendante du temps. Un tel état est d'énergie bien définie. La résolution, très courante, consiste à réinjecter 1.7 dans 1.5 et séparer les variables :

$$\mathrm{i}\hbar|\phi(\mathbf{r})\rangle\frac{\mathrm{d}\chi}{\mathrm{d}t}(t) = \chi(t)\hat{H}|\phi(\mathbf{r})\rangle \tag{1.8}$$

et séparer les variables des fonctions correspondantes aux états :

$$\mathrm{i}\hbar\frac{1}{\chi(t)}\frac{\mathrm{d}\chi}{\mathrm{d}t}(t) = \frac{1}{\phi(\mathbf{r})}\hat{H}\phi(\mathbf{r}) \tag{1.9}$$

comme chacun des deux membres ne dépend pas des variables présentes dans l'autre, l'équation est égale à une constante homogène à une énergie, que l'on désigne par E. L'équation différentielle qui s'en suit pour χ conduit à

$$\psi(\mathbf{r},t) = \exp\left(-\mathrm{i}\frac{E}{\hbar}t\right)\phi(\mathbf{r}) \tag{1.10}$$

où ϕ est normée dans vérifie

$$\hat{\mathcal{H}}\phi = E\phi \tag{1.11}$$

$\hat{H}$ étant un opérateur linéaire, si l'on peut en particulier trouver un base orthonormale dénombrable pour le produit scalaire hermitien, de l'ensemble des solutions au problème 1.11, de valeurs propres E_n et de vecteurs ϕ_n associés, alors toute solution peut se décomposer de manière unique sous la forme

$$\psi(\mathbf{r},t) = \sum_n c_n\phi_n(\mathbf{r})e^{-iE_nt/\hbar} \text{ avec } \sum_n |c_n|^2 = 1 \tag{1.12}$$

où les c_n sont des constantes complexes quelconques. C'est le cas par exemple pour l'oscillateur harmonique 1D dont on connait, grâce aux polynômes de Hermite et une démonstration dûe à Dirac.

États liés et états de diffusion On appelle un état lié un état $|\psi\rangle$ tel que

$$\int_{\mathbb{R}^d} |\psi|^2(\mathbf{x})\mathrm{d}\mathbf{x} = 1 \tag{1.13}$$

tandis qu'un état de diffusion est un vecteur propre tel que

$$\int_{\mathbb{R}^d} |\psi|^2(\mathbf{x})\mathrm{d}\mathbf{x} = +\infty \tag{1.14}$$

Ce stage fait beaucoup usage du résultat suivant :

Théorème 1.1.1. *Une base orthonormale d'états liés éléments propres solutions de l'équation de Schrödinger stationnaire*

$$\hat{\mathcal{H}} = \frac{\hat{p}^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \frac{1}{2}k\hat{x}^2 \tag{1.15}$$

où $\hat{p}$ et $\hat{x}$ vérifient $[\hat{x},\hat{p}] = \mathrm{i}\hbar$, est formée par les vecteurs (fonctions d'onde stationnaire)

$$\phi_n : x \mapsto \frac{1}{\sqrt{2^n n!}}\left(\frac{m\omega}{\pi\hbar}\right)^{\frac{1}{4}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right)\exp\left(-\frac{m\omega^2}{2\hbar}x^2\right) \tag{1.16}$$

ou encore, avec les paramètres m et k :

$$\phi_n : x \mapsto \frac{1}{\sqrt{2^n n!}} \frac{(km)^{\frac{1}{8}}}{\hbar^{\frac{1}{4}}} H_n\left(\frac{(km)^{\frac{1}{4}}}{\sqrt{\hbar}} x\right) \exp\left(-\frac{\sqrt{km}}{2\hbar} x^2\right) \tag{1.17}$$

associés aux valeurs propres (énergies)

$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega \tag{1.18}$$

où $\omega = \sqrt{\frac{k}{m}}$ et pour tout n, H_n représente le n-ième polynôme de Hermite *de forme physique*[2].

Ainsi dans cette base toute solution à l'équation instationnaire associée à cet hamiltonien se décompose sous la forme

$$\psi(\mathbf{r}, t) = \sum_n c_n \phi_n(\mathbf{r}) e^{-iE_n t/\hbar} \tag{1.19}$$

Lorsque l'on a affaire à un oscillateur harmonique en dimension supérieure, avec $V(x_1, \cdots, x_n) = \frac{1}{2}m(\omega_1 x_1^2 + \cdots + \omega_n x_n^2)$, on peut le traiter en cherchant des solutions séparables. Plus généralement lorsque le potentiel se met sous la $V(x, y) = V_x(x) + V_y(y)$ (en prenant une dimension égale à 2 par commodité, mais le propos se généralise), l'équation de Schrödinger s'écrit

$$-\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\Psi(x, y) + (V_x(x) + V_y(y))\Psi(x, y) = E\Psi(x, y) \tag{1.20}$$

donc en cherchant avec $\Psi(x, y) = \psi(x)\phi(y)$:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}(x) + V_x(x) - \frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi}{\mathrm{d}y^2}(y) + V_y(x) = E \tag{1.21}$$

que l'on peut réarranger de sorte à ce que le membre de gauche ne dépende que de la variable x et celui de droite, de la variable y, et ainsi on peut conclure que ϕ et ψ vérifient chacun une équation différentielle sur leur propre axe, que l'on peut résoudre indépendamment analytiquement si c'est possible (comme pour l'oscillateur harmonique) ou numériquement et obtenir par produit la solution séparable sur l'espace. Ce raisonnement occupera une place centrale dans la démarche employée pour le sujet de ce stage (sections suivantes) puisqu'une telle solution servira de base sur laquelle venir appliquer la théorie des perturbations.

Le sujet du stage s'intéresse plus particulièrement à l'état fondamental du hamiltonien considéré. À titre d'exemple, sur cette lignée, on a par exemple un résultat analytique pour un potentiel sous forme bilinéaire symétrique :

Proposition 1.1.2. *([All12], exercice du chapitre 7) Si $V(\mathbf{x}) = \mathbf{x}^\intercal A \mathbf{x}^\intercal$ avec $A \in \mathscr{S}_n^{++}(\mathbb{R})$, alors le niveau fondamental vérifiant $-\Delta u + Vu = Eu$ est donné par*

$$u(\mathbf{x}) = \exp(-\frac{1}{2}\mathbf{x}^\intercal A^{1/2}\mathbf{x}) \tag{1.22}$$

et d'énergie $E = \mathrm{tr}(A^{1/2})$.

[2] Les polynômes de Hermite existent en forme « physique » et forme « probabiliste », les deux étant utilisées comme leurs noms l'indiquent dans des contextes différents en probabilités ou en physique. Pour tout n la relation entre les deux formes est simplement le changement d'échelle et de variable donné par la relation $H_n^{proba} = 2^{n/2} H_n^{phys} \circ (\sqrt{2}X)$

dont j'ai rédigé la preuve :

Démonstration. On a en effet en notant $\mathbf{e}_i$ la base canonique :

$$\nabla u = \sum_{i=1}^{n} \left(\frac{\partial}{\partial x_i} \exp(-\frac{1}{2}\mathbf{x}^\intercal A^{1/2}\mathbf{x}) \cdot \mathbf{e}_i \right) \mathbf{e}_i = \nabla \left(-\frac{\mathbf{x}^\intercal A^{1/2}\mathbf{x}}{2} \right) u = -uA^{1/2}\mathbf{x}$$

d 'où

$$\begin{aligned} \Delta u &= \nabla \cdot (\nabla u) \\ &= -\nabla \cdot (uA^{1/2}\mathbf{x}) \end{aligned}$$

Or on a l'identité vectorielle $\nabla(f\mathbf{V}) = (\nabla f) \cdot \mathbf{V} + (\nabla \cdot \mathbf{V})f$ donc :

$$\begin{aligned} \Delta u &= -(\nabla u) \cdot A^{1/2}\mathbf{x} - (\nabla \cdot A^{1/2}\mathbf{x})u \\ &= (A^{1/2}\mathbf{x}u) \cdot A^{1/2}\mathbf{x} - \mathrm{tr}(A^{1/2})u \\ &= \mathbf{x}^\intercal A\mathbf{x}u - \mathrm{tr}(A^{1/2})u \end{aligned}$$

ce qui conclut. ■

Pour le prototypage des codes, cela permettra d'avoir une référence analytique pour ce type de fonctions.

Une telle solution analytique n'existe en général pas pour des problèmes plus « exotiques », comme pour beaucoup de problèmes d'intérêt en physique. C'est tout l'art des mathématiques appliquées, d'exprimer des solutions approchées à l'aide de raisonnements de l'ordre du sens physique (analyse dimensionnelle, comparaison des ordres de grandeur, négligence de certains phénomènes, etc). En physique quantique cependant, de tels raisonnements ne valent plus, et des méthodes plus fines doivent être employées pour parvenir aux résultats exploitables (calcul numérique, contrôle d'erreur, etc).

1.1.2 Motivations liées à la résolution des EDP en mécanique quantique

À l'échelle moléculaire, un système moléculaire se composant d'un certain nombre de noyaux et électrons, se trouve être très bien décrit par sa fonction d'onde.Comme on l'a vu l'évolution de la fonction d'onde est régie par l'équation de Schrödinger, *via* le hamiltonien, par exemple pour un électron autour d'un noyau :

$$H = -\frac{1}{2m_e}\frac{\partial^2}{\partial x^2} - \frac{Z_{\mathrm{nuc}}}{x - x_{\mathrm{nuc}}}$$

L'étude quantique de systèmes conservatifs est donc basée sur la résolution de l'équation aux valeurs propres de l'opérateur hamiltonien.

Pour des systèmes « suffisamment simples », on connaît des solutions analytique, par exemple l'oscillateur harmonique 1D, ou encore l'atome d'hydrogène non relativiste. Mais lorsque les systèmes se complexifient, le hamiltonien se complexifie spectaculairement, comprenant un nombre de terme augmentant typiquement en $\mathcal{O}(p^2)$ où p est le nombre de particules chargées en interaction. Il devient alors impossible de trouver les solutions analytiques. On peut lire dans [CTDL21a] dans l'introduction du chapitre XI :

> *Nous avons ainsi rencontré, dans les chapitres précédents, deux exemples importants de systèmes physiques (oscillateur harmonique et atome d'hydrogène) dont l'hamiltonien est suffisamment simple pour que son équation aux valeurs propres puisse être résolue exactement.*

> *Cependant, ceci ne se produit que pour un très petit nombre de problèmes ; en général, l'équation est trop compliquée pour que l'on puisse trouver ses solutions sous forme analytique : par exemple, on ne sait pas traiter exactement les atomes à plusieurs électrons, pas même celui d'hélium ; d'ailleurs, la théorie de l'atome d'hydrogène que nous avons développée au Chapitre VII (§ C) ne tient compte que de l'interaction électrostatique entre le proton et l'électron ; lorsqu'on ajoute à cette interaction principale des corrections d'origine relativiste (telles que les forces magnétiques), l'équation obtenue pour l'atome d'hydrogène lui-même n'est plus soluble analytiquement.*

Or, trouver les états de systèmes en chimie quantique se révèle particulièrement utile pour décrire les interactions d'atomes dans des molécules, dans toutes sortes de domaines aux applications très variées, avec à la clé l'accès au calcul de nombreuses propriétés physiques et chimiques du système comme la conformation spatiale, l'énergie de dissociation, la conductivité, etc. [Dus17]

Néanmoins, même pour des systèmes simples comme les a décrit l'ouvrage de Cohen-Tanoudji, uniquement composés de quelques particules, les équations deviennent vite extrêmement ardues car on dénombre 3 coordonnées d'espace pour chaque particule et une pour le spin. Il faut donc recourir à des approximations, des résolutions approchées ou numériques, tout comme dans beaucoup de domaines de la physique. Le physicien Paul Dirac écrivait en effet dès 1929 dans [DF29] :

> *The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation.*

1.1.3 Généralités brèves sur les opérateurs hamiltoniens et leur spectre

Dans cette sous-section, j'expose très brièvement quelques généralités concernant le formalisme utilisé pour parvenir à la résolution de problèmes aux valeurs propres à l'aide de différences finies. Ce qui suit provient des sources [CLBL21], qui reprend lui-même des éléments de [RS75], [RS79], [RS80b], [RS80a]. Les fondements mathématiques à la base de la mécanique quantique font partie de l'analyse fonctionnelle et la théorie des opérateurs, qui sont des domaines extrêmement vastes et moins triviaux que l'algèbre linéaire en dimension finie. Le but n'est pas ici de répliquer toute la connaissance disponible sur le sujet à travers de nombreux ouvrages, mais simplement de redonner les définitions incontournables pour justifier les choix de conception des codes faits dans le cadre du stage. Ce qui est suit est donc une succession de remarques, généralités et de définitions très disparates.

Lorsqu'on se pose un problème en physique quantique, on prend généralement pour cadre un espace de Hilbert, i.e. un espace vectoriel complexe muni d'un produit scalaire hermitien, antilinéaire à gauche (par conjugaison complexe) et linéaire à droite. Il peut être de dimension finie ou infinie. Typiquement, pour modéliser un système à quelques niveaux (comme les atomes d'une cavité laser), une dimension finie suffira, tandis que pour décrire des particules dans l'espace, les fonctions d'ondes de l'espace des états liés en tant qu'éléments de $L^2(\mathbb{R}^3)$, se révèlent adaptés. En dimension finie, $\mathbb{C}^d$ a également une structure hermitienne. Si l'on connaît très bien les propriétés de ces espaces en dimension finie avec l'algèbre linéaire, qui permet notamment de décrire le spectre de matrices hermitiennes correspondant à des opérateurs associés à des observables, il n'en est pas de même en dimension infinie, où de nombreux problèmes se posent. Tous les opérateurs ne sont pas forcément bien définis sur des espaces de Hilbert, à l'instar du laplacien $-\Delta$ qui intervient dans l'équation de Schrödinger. La stucture des spectres change également : ils ne sont plus nécessairement discret partout.

En annexe A sont ainsi donnés quelques théorèmes principaux qui fondent la théorie des opérateurs en physique quantique.

Le dernier théorème en particulier donne une interprétation du spectre d'un opérateur associé à un observable en mécanique quantique, en particulier, pour le hamiltonien, l'énergie peut prendre un ensemble de valeurs dans $\sigma(\hat{H})$ soit discrètes correspondant aux états liés, ou continues, il s'agit des états de diffusion.

Dans les applications concrètes en chimie quantique notamment, ce sont bien évidemment les états liés qui nous intéressent car ce sont précisément eux qui permettent de décrire des situations réelles (particules localisées, dans une molécule intègre, d'énergie bien déterminée, etc).

On souhaite tout particulièrement connaître les valeurs prises par l'observable hamiltonien dont les valeurs propres représentent l'énergie, dans le spectre discret.

Voici quelques résultats tirés de [RS80b] qui permettent de donner des résultats très forts.

L'ouvrage [RS80b] est extrêmement complet, et la théorie développée pour parvenir à ce résultat et les suivants sont compliqués et longs et repose abondamment sur des résultats d'analyse fonctionnelle. Ici, je retiens l'essentiel en coupant court à la la longue chaine de plusieurs centaines de théorèmes qui apparaissent dans [RS80b] pour parvenir aux conclusions principales et justifier l'approche numérique et résoudre le problème posé par le stage.

Théorème 1.1.3. *([RS80b], théorème XIII.47) Soit $V \in L^2_{loc}(\mathbb{R}^m)$ positif et tel que $V \to +\infty$ Alors $H = -\Delta + V$ est un opérateur possède un état fondamental non dégénéré et la fonction propre correspondante est strictement positive.*

Vient ensuite le théorème principal (numéro XIII.64), donné au chapitre « Opérateurs à résolvent compact », qui stipule en particulier qu'**un opérateur auto-adjoint vérifiant $\|A\psi\| \geq m\|\psi\|$ pour tout ψ et pour un certain $m > 0$ (*bounded from below operator*), a son opérateur résolvent $(A - z)^{-1}$ compact pour un certain $z \in \rho(A)$, si et seulement si son spectre est entièrement discret et réel, et tend vers l'infini sans point d'accumulation, i.e. qu'on peut trouver (ϕ_n) et (λ_n) indicés sur $\mathbb{N}$ tels que $A\phi_n = \lambda_n\phi_n$ et $\lambda_0 \leq \lambda_1 \leq \cdots$ avec $\lambda_n \to +\infty$.**

Théorème 1.1.4. *([RS80b], théorème XIII.67) Soit $V \in L^1_{loc}(\mathbb{R}^n)$ minoré tel que $V \to +\infty$ Alors $H = -\Delta + V$ est un opérateur à résolvent compact, possède un spectre purement discret de fonctions propres.*

On voit donc que les théorèmes 64 et 67 mis à la suite permettent de montrer que pour un potentiel localement intégrable minoré qui tend vers $+\infty$ inplique que le spectre est minoré et discret et tend vers $+\infty$.

Théorème 1.1.5. *([RS80b], théorème XIII.69) Soit $n \geq 3$ et $V = V_1 + V_2$ avec $V_2 \in L^{n/2}(\mathbb{R}^n) + L^\infty(\mathrm{R}^n)$, $V_1 > 0, V_1 \to +\infty$ et $V_1 \in L^1_{loc}(\mathbb{R}^n)$. Alors $H = -\Delta + V_1 + V_2$ est un opérateur à résolvent compact.*

Cela permet d'avoir le comportement qualitatif du spectre d'un opérateur en fonction du comportement qualitatif du potentiel, ce qui est très puissant comme cadre général. « Que demander de plus ? », écrit même l'auteur. Avec le théorème XIII.64 évoqué, cela permet de s'assurer que les valeurs propres de tels hamiltoniens vérifiant ces conditions avec des hypothèses de potentiels confinants sont bien discrètes.

En outre, [RS80b] fournit plusieurs théorèmes d'intégrabilité des fonctions propres du hamintonien en fonction du comportement de V (théorèmes XIII.70 à XIII.72), et [Sim75] également. Ces résultats sont à rapprocher de l'approximation connus sous le nom WKB pour Brillouin-Kramers-Wentzel qui est une méthode permettant d'approcher les fonctions propres dans un cadre semi-classique développée en 1826, basée sur des développements asymptotiques. Elle aboutit à une solution vérifiant asymptotiquement

$$\psi \sim \exp\left(-\int \sqrt{V - E}\mathrm{d}x\right) \tag{1.23}$$

Intuitivement, ces théorèmes donnent des inégalités de concentration sur notre fonction d'onde (donc sur la localisation de la particule), en fonction de la qualité « confinante » du potentiel. Lors du changement de variable, cet équivalent se révèlera utile.

En outre, il est également intéressant pour notre cadre numérique, car cela permet de justifier la discrétisation finie de l'espace, à savoir la troncature de notre grille. Nous n'avons de toute façon pas le choix : la mémoire de l'ordinateur est finie, mais ce résultat permet de s'assurer que la fonction d'onde atteint le zéro machine au delà d'une distance caractéristique raisonnablement faible vis-à-vis de la zone d'intérêt. Cela dit, comme le précise l'ouvrage, cette borne est loin d'être optimale en prenant l'exemple de l'oscillateur harmonique dont les fonctions propres décroissent en $\mathcal{O}(x^n e^{-x^2})$ (à changement d'échelle affine près) avec le nombre quantique principal n. Précisons un peu :

Théorème 1.1.6. *[Sim75] Soit $V \in \mathscr{C}^\infty(\mathbb{R}^n)$ et soit $\hat{H} = -\Delta + \hat{V}$. Supposons que ψ est une fonction propre de $\hat{H}$. Alors si $V(x) \geq c\|x^{2n}\| - d$ alors*

$$\forall \varepsilon > 0,\ \exists E > 0,\ \forall x,\ |\psi(x)| \leq D \exp(-(n+1)^{-1}(c+\varepsilon)^{1/2}|x|^{n+1}) \tag{1.24}$$

Appliquons ce théorème à un cas courant : supposons avoir un potentiel de confinement avec $c = \frac{1}{2}$, $d = 0$, $n = 1$ (cas du potentiel quadratique d'un oscillateur harmonique). On obtient pour un certain paramètre δ et un certain D ; $|\psi(x)| \leq D \exp(-\delta|x|^2)$. On connaît les fonctions propres de l'oscillateur harmonique, mais connaître les constantes de la fonction majorante dans le cas général n'est pas évident, et la démonstration de [Sim75] n'est pas constructive, donc je ne peux pas en exhiber directement a priori. Mais on peut numériquement estimer un seuil d'espace à partir duquel on négligera cette valeur.

1.1.4 Stratégies de résolution des états stationnaires pour des systèmes simples

Pour trouver les états stationnaires d'un hamiltonien d'un système simple proche de celui qui nous intéresse (expliqué en section 2.1), reprenons un exemple canonique : l'oscillateur harmonique quantique à deux dimensions. Conformément à ce qui précède, on peut écrire à nouveau son hamiltonien, composé d'un terme laplacien en $-\frac{\hbar^2}{2m}\Delta$ et d'un terme de potentiel $\hat{V}$. Ce hamiltonien est un opérateur hermitien sur $L^2(\mathbb{R}^2)$, et si V vérifie $V(x,y) \xrightarrow[\|x,y\|\to+\infty]{} +\infty$, il s'agit d'un potentiel de confinement. Alors le spectre de $-\frac{\hbar^2}{2m}\Delta + \hat{V}$ est entièrement discret et constitué d'une suite dénombrable croissante de valeurs indicée sur $\mathbb{N}$ tendant vers l'infini, d'après ce qui précède.

Problèmes et enjeux liés à la résolution numérique

Je tente de mettre en évidence dans ce paragraphe les difficultés numériques que l'on rencontre en discrétisant des EDP à l'aide des différences finies en particulier.

Il existe plusieurs stratégies pour approcher numériquement des solutions d'EDP où les variables de temps et d'espace sont séparables. Les chimistes utilisent en général la méthode de Rayleigh-Ritz qui consiste, comme en mécanique classique pour le calcul de structures, à « projeter »[3] une solution sur une « base »[4] de fonctions dans le but de trouver la solution qui satisfait les équations du problème avec les conditions aux limites ou conditions initiales. Cela donne lieu à beaucoup d'activité de recherche visant à optimiser ces bases (notamment, gaussiennes), et j'ai pu discuter au travers du stage avec des chercheurs ou doctorants qui orientent la leur dans cette voie.

[3] Léger abus de langage : il s'agit plutôt en fait de décomposer un ensemble de composantes que l'on choisit, et on appelle cela une base.
[4] Même remarque.

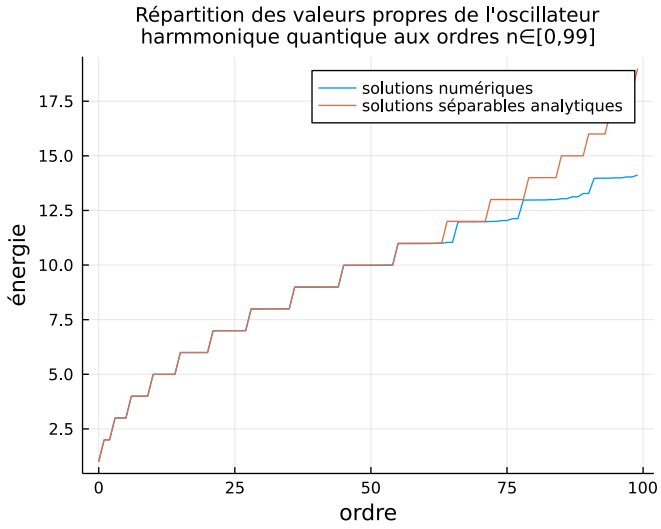
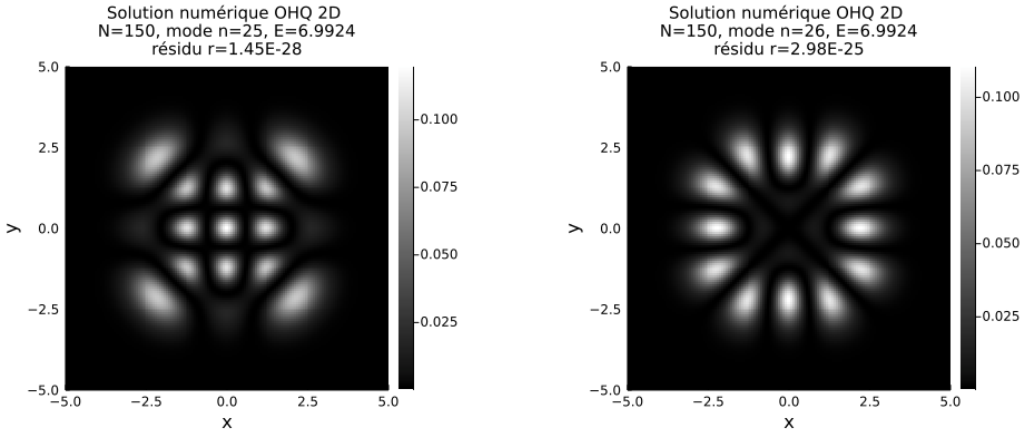


Figure 1.1 : Énergies analytiques et numériques pour le hamiltonien de l'oscillateur harmonique 2D : numériques (différences finies) et analytiques.

Un autre moyen classique, est de discrétiser une partie de l'espace à l'aide de différences finies ou d'éléments finis. Dans mon cas, j'ai utilisé les différences finies pour le problème unidimensionnel physique (section 2.1) et ferai néanmoins en sorte d'éviter d'avoir recours à une résolution 2D (ni en coordonnées d'espace ni en coordonnées généralisées) coûteuse en temps de calcul, mais je l'utiliserai pour avoir accès à une solution approchée de référence avec laquelle comparer les résultats.



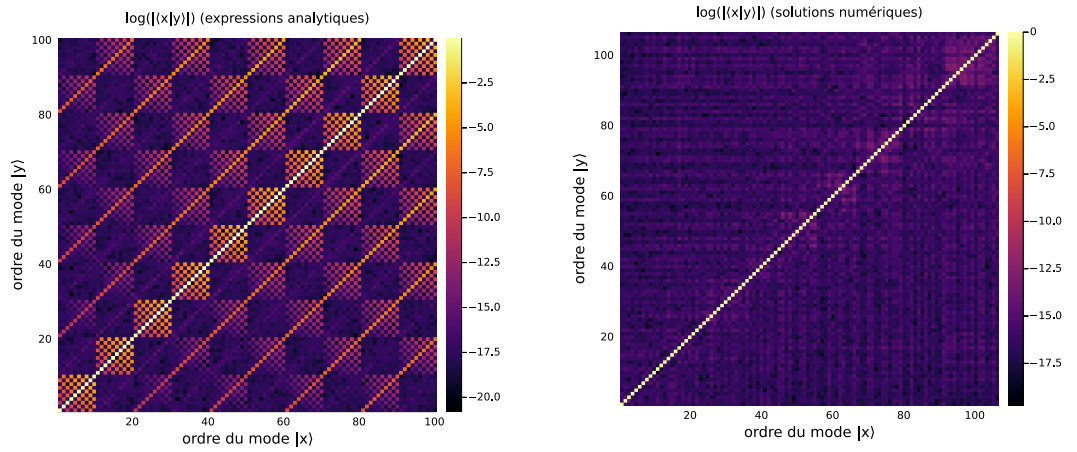
(a) 25ème mode **(b)** 26ème mode trouvé

Figure 1.2 : 2 états non séparables qui ont convergé, trouvés par la résolution numérique avec laplacien à 5 points des valeurs propres de l'oscillateur harmonique quantique 2D avec 150×150 points sur la grille et le potentiel $V(x, y) = \frac{1}{2}(x^2+y^2)$, et les paramètres $m = \hbar = 1$. La résolution numérique utilise une méthode basée sur les espaces de Krylov.

Ce genre de résolutions numériques pose plusieurs problèmes qui peuvent influencer les résultats pour les plus grands modes :

- D'abord, il y a bien-sûr l'erreur commise en raison de la discrétisation et le passage d'un problème dans $\mathcal{H} = L^2(\mathbb{R})$ espace hilbertien dans $\mathbb{R}^{N\times N}$ et l'approximation des dérivées partielles par les différences finies. Typiquement, les valeurs propres les plus petites sont les mieux approchées pour la résolution du laplacien numérique à 5 points en 2D (ou à 3 points en 1D), et on peut prouver que n'importe quelle valeur propre du spectre du laplacien continu peut être approchée en discrétisant ainsi l'espace suffisamment finement (voir annexes). Mais qu'en est-il lorsque l'on a un hamiltonien avec potentiel de confinement quelconque ? En illustration, la figure 1.1 montre les énergies calculées numériquement des 100 premiers modes de l'oscillateur harmonique quantique 2D avec 100 points de discrétisation d'espace, et les valeurs théoriques associées aux modes propres séparables. On voit sur cet exemple que les plus petites énergies sont correctement approchées, mais que la divergence survient soudainement. La plupart du temps, mes expériences ont montré que les solutions séparables qu'on connaît sont trouvées pour les petites énergies seulement. Mais la résolution aboutit à des solutions numériques non séparables plus « exotiques » pour les grandes valeurs propres (figure 1.2).
- Ensuite, un schéma à taille fixée ne permet pas de rendre compte des états liés de plus grande énergie, dont la fonction d'onde prend des valeurs significatives en dehors des frontières (tyipquement, au-delà de la précision machine), car l'ensemble de point satisfait nécessairement la condition de normalisation et la réalité physique n'est alors plus suffisamment bien approchée.
- Augmenter la précision requiert d'augmenter le nombre de points de discrétisation, ce qui a pour effet d'augmenter les temps de calcul ou l'instabilité numérique si le conditionnement devient mauvais, au risque de devoir recourir à des méthodes itératives dont la solution de départ peut influencer les solutions obtenues. En général, les vecteurs propres trouvés restent tributaires de l'algorithme de résolution, du bruit numérique, la forme du domaine, des effets d'échantillonnage, etc.

De la même manière, calculer un échantillonnage d'une solution séparable analytiquement exacte connue pose ses propres problèmes. Pour reprendre l'exemple canonique de l'oscillateur harmonique, les formules analytiques comportent dans les formules des facteurs difficiles à calculer (en $\mathcal{O}(2^n n!)$ dans le cas des polynômes de Hermite), ce qui peut provoquer une instabilité numérique, lorsque des termes grands ou facteurs grands se compensent pour donner de petites quantités. De plus, l'échantillonnage pose aussi ses propres problèmes, on obtient par exemple un repliement de spectre dans le calcul des intégrales de recouvrement approchées (c'est-à-dire, des produits scalaires des vecteurs-états entre eux) pour la solution exacte (figure 1.3). La conséquence est que certains modes ne sont pas orthogonaux entre eux à un seuil suffisant (seulement à $\approx 10^{-3}$ près pour les pires combinaisons, ce qui est énorme). Cela peut poser problème pour d'éventuels calculs qui exploitent massivement l'hypothèse d'orthogonalité des modes, comme c'est le cas pour la théorie des perturbations.



(a) modes analytiques échantillonnés **(b)** modes numériques

Figure 1.3 : Carte des produits scalaires entre les 100 premiers vecteurs propres séparables c'est-à-dire 10 modes possibles suivant chaque axe) échantillonnés d'après les formules analytiques et les vecteurs numériques trouvés, sur 150 points de discrétisation. On observe un repliement de spectre qui conduit à une mauvaise orthogonalité de certains modes entre eux. Pour des problèmes où l'orthogonalité est importante (comme la *sum over states* en théorie des perturbations), le choix de l'arbitrage entre les deux ensembles de vecteurs se pose, en parallèle de la précision sur les énergies associées (figure 1.1).

1.1.5 Généralités élémentaire et succinctes sur les problèmes d'analyse numérique les plus courants utilisés dans le contexte de ce stage

Problèmes matriciels rencontrés

En analyse numérique, on peut citer en particulier deux problèmes canoniques fondamentaux, qui sont immensément utiles en physique, ingénierie entre autres :

- Résoudre un système linéaire : apparaît dans un nombre incalculable d'applications en général. En physique et en EDP, on peut citer : dans la simulation d'une EDP dans le temps avec un schéma implicite (comme Crank-Nicholson).
- Résoudre un problème aux valeurs propres ou un problème aux valeurs propres généralisées pour une matrice (comme pour Schrödinger stationnaire).

Sources d'erreur On distingue également quel que soit le problème deux sources d'erreurs qui contribuent :

- **L'erreur d'arrondi**, due à la représentation discrète des nombres en machine ;
- **L'erreur de troncature**, due à l'arrêt en temps fini de méthodes itératives, ou à la troncature de sommes, etc.

Dans les deux cas, le choix optimal de la méthode dépend beaucoup du type de matrice que l'on a en présence. La matrice est-elle symétrique ? creuse ? réelle ? complexe ? hermitienne ? tridiagonale ? triangulaire ? une combinaison de ces derniers ? etc. Ce sont autant de cas pour lesquels l'usage d'algorithmes dédiés donneront des résultats bien plus rapides en coût de calcul et garantissant une meilleure stabilité numérique.

Algorithmes courants pour la résolution de systèmes linéaires

Pour résoudre des systèmes linéaires ou calculer une inverse, on connaît bien le pivot de Gauss dans le cas général, et les questions de stabilité numériques peuvent être plus ou moins garanties dans l'algorithme (en choisissant un pivot de plus grande valeur absolue possible, par exemple), ainsi que par des résultats analytiques, avec le conditionnement. Néanmoins, son coût de calcul est élevé, en $\mathcal{O}(n^3)$ pour une matrice $n \times n$.

D'autres algorithmes itératifs existent et présentent un compromis bien meilleur entre efficacité (vitesse de convergence) et précision, parmi lesquels on peut citer :

- Méthode de Gauss-Seidel
- Méthode de Jacobi

On peut également passer par des décompositions, dont on connaît des algorithmes efficaces pour les calculer. Ensuite, on utilise des algorithmes pour résoudre des systèmes linéaires à partir de la décomposition une fois acquise.

Une décomposition, pour une matrice A, est une factorisation ou un changement de base, qui exprime la matrice sous une forme aux propriétés intéressantes, ou plus explicites, plus facilement interprétables, plus stables numériquement, etc. On peut citer :

- Factorisation de Bunch-Kaufman pour une matrice symétrique quelconque complexe (écrire $PAP^\intercal = LDL^\intercal$ où P set une matrice de permutation, L une triangulaire inférieure et D diagonale par blocs 1×1 et 2×2)
- Factorisation de Cholesky pour une matrice hermitienne (écrire $A = LL^\star$ produit d'une triangulaire inférieure L à diagonale réelle et positive et sa conjuguée transposée)
- Décomposition LU (écrire $A = LU$, produit d'une triangulaire inférieure L et d'une triangulaire supérieure U)
- Décomposition QR (écrire $A = QR$, produit d'une orthogonale Q et d'une triangulaire supérieure R)
- Décomposition de Schur (écrire $PA = QUQ^{-1} = QUQ^\star$ changement de base orthogonal Q vers une triangulaire supérieure U)

Certaines décompositions sont plus avantageuses que d'autres suivant le problème. Se ramener à des matrices triangulaires est en particulier très intéressant pour résoudre un système linéaire, car on peut directement trouver les inconnues récursivement en descendant ou remontant les lignes d'équations, et cela permet d'éviter de recourir au pivot de Gauss plus coûteux. Par exemple, orsque l'on a une matrice hermitienne (très pertinent au regard des opérateurs hermitiens rencontrés en physique quantique), la décomposition de Cholesky est environ deux fois plus efficace que la décomposition LU[5].

Algorithmes courants pour extraire des éléments propres

Pour trouver des éléments propres analytiquement, en général on calcule d'abord le polynôme caractéristique, on tente de trouver ses racines, puis on résout un système linéaire dans un second temps.

Pour des matrices spéciales (par exemple, pour les matrices de Toeplitz, les matrices tridiagonales, les matrices de Vandermonde, etc), on peut parfois trouver par récurrence permet

[5] William H. ; Saul A. Teukolsky ; William T. Vetterling ; Brian P. Flannery (1992). Numerical Recipes in C : The Art of Scientific Computing (second ed.). Cambridge University England EPress. p. 994.

des formes closes ce qui permet d'aboutir à des résultats généraux. Il n'en est rien dans le cas général.

Appliquer ceci numériquement est une mauvaise idée. Plus précisément, tenter de calculer le polynôme caractéristique d'une matrice numériquement est en général une mauvaise idée, d'une part parce que l'opération est coûteuse[6], et d'autre part parce qu'on ne connaît pas de formule générale pour la résolution de polynômes de degré supérieur ou égal à 5 (théorème d'Abel-Rufini), et enfin de petites erreurs sur le calcul des coefficients peuvent amener à de grandes erreurs sur les valeurs propres trouvée.

Lorsque l'on veut calculer numériquement des valeurs propres, il est raisonnable de classer les solveurs propres en fonction de la taille et de la configuration non nulle de la matrice. Numériquement, trouver des valeurs propres nécessitent un processus itératif parmi lesquels on peut citer :

- Méthodes de la puissance itérée
- Méthodes d'Arnoldi, basées sur la construction de sous-espaces de Krylov

À titre d'exemples, voici en annexes C quelques résultats sur la localisation de spectres et les méthodes de la puissance que j'ai eu l'occasion de rencontrer durant ma scolarité, dans les cours de l'École Centrale de Lyon ou ailleurs qui portent sur l'obtention itérative d'élements propres.

Pour obtenir rapidement et de manière fiable des états propres cependant, j'utiliserai les codes éprouvés des nombreuses bibliothèques disponibles en Julia, en particulier `KrylovKit`, qui renvoie également des métadonnées sur le succès de convergence, les résidus, etc. à chaque appel de la fonction. Je ne m'attarderai pas davantage dans ce rapport (ni l'ai-je fait pendant le stage) sur les mathématiques sous-jacentes de ces algorithmes connus, car ce n'est pas vraiment l'objet. Il faut simplement se rappeler que ces méthodes sont itératives en appliquant la fonction linéaire un grand nombre de fois, et qu'elles construisent des bases orthogonales de sous-espaces, et donc qu'elles sont coûteuses en temps et en espace. Il convient de minimiser leurs appels, et de minimiser la dimension de Krylov choisie et la dimension des arguments.

Contrôle d'erreur a posteriori

Une fois que l'on dispose numériquement de vecteurs propres ou valeurs propres, il peut être intéressant, voire il est absolument nécessaire, de pouvoir contrôler l'erreur commise.

Pour connaître l'influence d'une perturbation sur les éléments propres d'une matrice, que ce soit des perturbations souhaitées (modification du hamiltonien dans la théorie des perturbations) ou non souhaitées (comme une erreur de troncature).

Théorème 1.1.7. *(Corollaire 3.3, [Saa11]) SI A est une matrice hermitienne, $\tilde{\lambda}$ une valeur propre approchée et $\tilde{u}$ son vecteur propre approché, et $r = A\tilde{u} - \tilde{\lambda}\tilde{u}$, alors il existe une valeur propre λ de A telle que*

$$|\lambda - \tilde{\lambda}| \leq \|r\|_2 \tag{1.25}$$

Enfin, nous avons l'inégalité de Kato-Temple qui porte sur un opérateur pas forcément en dimension finie, et donne un résultat fort, très utilisé dans le contexte de la mécanique quantique pour évaluer la précision d'états propres approchés d'opérateurs.

Théorème 1.1.8. *(Kato-Temple) Soit $\mathcal{H}$ un espace de Hilbert et A un opérateur auto-adjoint de A. Soit $\tilde{u}$ un vecteur propre approximatif normé de A et $\tilde{\lambda} = \langle A\tilde{u}|\tilde{u}\rangle$ sa valeur propre approchée. On définit r le résidu $r = A\tilde{u} - \tilde{\lambda}\tilde{u}$ et $\varepsilon = \|r\|$ la norme du résidu. Soit $]\alpha, \beta[$ un intervalle contenant $\tilde{\lambda}$. Si le seul point du spectre de A est λ, alors on a l'inégalité*

$$-\frac{\varepsilon^2}{\beta - \tilde{\lambda}} \leq \lambda - \tilde{\lambda} \leq \frac{\varepsilon^2}{\tilde{\lambda} - \alpha} \tag{1.26}$$

[6] Il faut calculer $n!$ produits de termes

1.1.6 *Simulations numériques : présentation des méthodes* hardware et software

Pour simuler numériquement les équations de SCHRÖDINGER stationnaires, j'ai eu recours eu langage JULIA qui m'a été proposé, que j'ai appris pour l'occasion, et qui faisait donc partie du travail compris dans le stage.

Présentation rapide de JULIA

Julia est un langage jeune, typé dynamiquement (c'est-à-dire que les variables peuvent changer de type), de haut niveau (c'est-à-dire, proche du langage humain), performant. La syntaxe sont proches du python, mais contrairement à ce dernier, Julia est un langage compilé, et son efficacité et sa rapidité est proche de celles du C, ce qui en fait un langage de premier choix pour le calcul scientifique car il combine ces deux avantages, ce qui est remarquable.

Pour les besoins des simulations j'ai eu l'occasion de travailler avec beaucoup de bibliothèques orientées calcul scientifique. Cela a été pour moi l'occasion de travailler sur les bonnes pratiques d'état de l'art en simulation numérique et sur l'optimisation[7] du code.

- `BenchmarkTools`, `Profile` (pour l'évaluation des performances du code)
- `SparseArrays`, `LinearAlgebra`, `SpecialMatrices`, pour la manipulation des tableaux ;
- `DSP`, `ImageFiltering`, `FFTW` pour la transformée de FOURIER rapide et la manipulation de noyaux de convolution (qui ont servi notamment pour étudier Julia au début sur des codes de propagation de paquets d'ondes, et obtenir des dérivées fréquentielles) ;
- `KrylovKit`, `IterativeSolvers`, `LinearMaps`, `LinearOperators` pour les algorithmes itératifs (ARNOLDI, KRYLOV, gradients conjugués, etc), pour algorithmes itératifs ; `Arpack`[8] pour résoudre des problèmes aux valeurs propres ;
- `Polynomials`, `SpecialPolynomials` pour la manipulation des polynômes de Hermite notamment.
- `CUDA` et ses sous-librairies `CUDA.CUSPARSE`, `CUDA.CUSOLVER`, `CUDA.CUFFT`, qui transcendent les paquets précédents (par exemple les méthodes itératives de `KrylovKit` et `IterativeSolvers` évoquées fonctionnent avec des vecteurs CUDA, on peut donc faire tourner leurs routines sur le processeur graphique, au prix d'une précisions simple au lieu de double sur des GPU grand public).
- `Graphs`, `DataStructures`, `MetaGraphsNext` pour l'utilisation de graphes étendus (dont les arêtes et sommets peuvent être surchargés par des *types composites* pour résoudre un problème personnalisé) dans la partie programmation dynamique pour optimiser les stratégies de calcul.

J'aurai l'occasion de revenir plus en détail sur l'usage de JULIA dans le cadre du problème qui nous intéresse.

Comme déjà évoqué, JULIA est un langage en devenir mais qui manque encore de documentation. J'ai pu m'en rendre compte lors de mes lectures de documentation, au travers des forums de développement. On y trouve encore l'évocation de bugs à améliorer, de fonctions à ajouter, de typages à corriger, etc. Un travail de fond important a alors consisté à tester les algorithmes sur des cas représentatifs pour voir lesquels étaient les plus rapides, car au delà des mathématiques qui fondent l'*algorithme*, le *programme* peut montrer des performances légèrement différentes selon le type de machine, la parallélisation, son matériel, etc.

[7] Par « optimisation » j'entends : parvenir à la solution donnée avec la précision requise en résuisant au maximum les ressources (en calcul, en temps) utilisées.

[8] Ce n'est pas une bibliothèque Julia à proprement parler mais une interface (*wrapper*) pour les routines Fortran 77 de résolution de problèmes aux valeurs propres en grande dimension.

Parallélisation sur GPU : généralités En sus de l'ensemble des algorithmes existants pour tel ou tel problème, certains sont hautement parallélisables, on peut alors les exécuter sur des GPU (*Graphic Processor Unit*). Destinés lors de leur conception à calculer des transformations géométriques de triangles pour l'affichage de graphiques, ils se sont rapidement avérés particulièrement utiles aux informaticiens pour l'algèbre linéaire et le Machine Learning en particulier. Parce qu'un processeur graphique dispose de très grandes quantités de cœurs, il accélère considérablement des calculs en les exécutant en même temps[9], alors qu'ils devraient être exécutés les uns après les autres sur un cœur de processeur classique. Nvidia fournit des API pour ses processeurs au travers de la bibliothèque CUDA (pour *Compute Unified Device Architecture*), avec des interfaces dans de nombreux langages dont C++ et JULIA, particulièrement CUSOLVER. On trouve de nombreux examples de *wrappers* en Julia sur Github.

Il est à noter que dans bien des cas, l'usage des fonctions sur CPU reste le plus efficace, plus rapide, et très bien optimisé par les compilateurs efficaces (en Julia, qui est amplement conçu pour le calcul scientifique, c'est le cas).

Cela s'explique par les temps de transit entre la mémoire du GPU et celle du CPU, qui dépassent généralement les temps de calcul pour des opérations sur des données de petite taille. En effet, le GPU était un processeur à part entière, il n'est pas synchronisé avec le CPU. Des temps d'attente peuvent aussi empirer les performances du code. En ce sens, **le calcul sur GPU se révèle en général très utile pour certains problèmes pouvant mettre en œuvre massivement et de manière répétée des calculs matriciels lourds mais son recours n'est absolument pas systématique. Le calcul à haute performance** (ou HPC pour *High-Performance Computing*), qui est une discipline de l'informatique, s'attache à exploiter intelligemment et au mieux le potentiel des GPU surtout pour des problèmes à grande échelle, en faisant un meilleur usage du matériel (cache, etc) y compris à bas niveau (proche de la machine) qui peuvent se révéler utile dans mon sujet de stage[10]. J'ai eu l'occasion de discuter sur ce point avec des chercheurs du Laboratoire de Chimie Théorique (LCT) de Sorbonne Université lors d'un séminaire organisé pendant mon stage, quoique cela ne fît absolument pas partie de mon travail.

En annexe C.3.1, j'ai mis un exemple très concret de calcul d'éléments propres pour le cas de la matrice du laplacien à 5 points dont on connaît les valeurs et vecteurs propres analytiquement, pour démontrer les avantages et inconvénients des deux paradigmes (GPU et CPU).

En annexe D.1, j'ai décrit tout un travail annexe réalisé pendant ce stage, qui repose sur l'idée d'alterner entre CPU et GPU pour certains calculs lourds dans un algorithme calculatoire, tout en choisissant un chemin minimisant le temps machine utilisé. Il s'agit pour seulement d'une heuristique, car la complexité combinatoire derrière le problème est prohibitive. Je n'ai finalement pas utilisé l'algorithme de décision réalisé car comme je le décris plus bas dans ce rapport, les étapes délicates qui prennent le plus de temps (à savoir le calcul de la référence pour l'état, ce qu'on s'interdit normalement de faire) sont plus précises et rapides sur mon CPU (machine grand public).

[9] https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#from-graphics-processing-to-general-purpose-parallel-computing

[10] Précisément, on m'a expliqué que contrôler mieux l'utilisation du cache dans les GPUs permet d'optimiser des calculs. Ça aurait par exemple pu être utile pour faire des produits creux d'un hamiltonien par un état avec un simple décalage des lignes entre elles, comme me l'a suggéré Paul Cazeaux.

Chapitre 2

Partie centrale du stage : simulation numérique d'équations de Schrödinger

Cette section porte cette fois sur le travail qui constitue le sujet de mon stage exposé linéairement. Je conçois les algorithmes nécessaires au fur et à mesure et j'applique les théorèmes connus des chapitres précédents pour parvenir à des résultats intéressants.

2.1 Définition du problème

Le sujet de ce stage s'intéresse à un système de trois particules dont l'une est beaucoup plus légère que les deux autres, lesquelles vivent sur un axe, en 1D.

Ce cas est fortement inspiré de l'ion H_2^+ (deux protons massifs de masses identiques m_p et un électron de masse m_e gravitant autour d'eux évoluant sur une droite). Les tests numériques seront d'ailleurs effectués pour cet ion.

On part du hamiltonien en coordonnées généralisées 1D : nous avons 3 particules évoluant en 1D donc 3 dimensions a priori :

$$\mathcal{H}(X_1, X_2, x, \mathbf{P}_1, \mathbf{P}_2, \mathbf{p}) = \frac{\mathbf{P}_1^2}{2M_1} + \frac{\mathbf{P}_2^2}{2M_2} + \frac{\mathbf{p}^2}{2m} + \mathcal{V}_1(x - X_1) + \mathcal{V}_2(x - X_2) + \mathcal{W}(X_2 - X_1) \quad (2.1)$$

En physique classique, on peut alors exprimer les quantités de mouvement en fonction des variables de position.

Le but ultime étant de trouver la configuration de l'état fondamental (de plus basse énergie).

2.2 Changement de coordonnées et réduction des DDL

Pour simplifier le problème, on va supposer que le système est isolé, la position du centre de masse ne nous intéresse alors pas.

Nous allons alors effectuer le changement de coordonnées $(X_1, X_2, x) \rightsquigarrow (X_G, R, r)$, avec X_G le barycentre de masse des trois particules :

$$X_G = \frac{M_1 X_1 + M_2 X_2 + mx}{M_1 + M_1 + m} \quad (2.2)$$

et R la distance séparant algébrique les deux noyaux, et r la distance algébrique de l'électron au centre de masse. En figure 2.1, sont représentés ces paramètres.

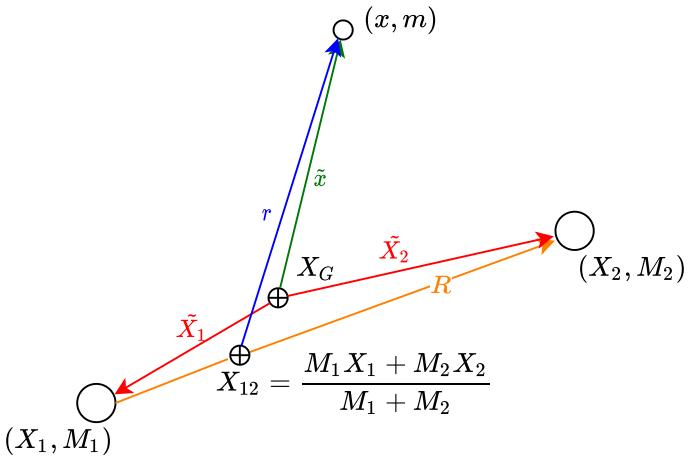


Figure 2.1 : Les deux paramétrages et les relations entre les paramètres. Le dessin « déplié » dans le plan 2D sert uniquement à la lisibilité. Les trois particules restent alignées, sur un unique axe.

Nous avons donc

$$\begin{bmatrix} X_G \\ r \\ R \end{bmatrix} = \begin{bmatrix} \frac{M_1}{M_1+M_2+m} & \frac{M_2}{M_1+M_2+m} & \frac{m}{M_1+M_2+m} \\ -\frac{M_1}{M_1+M_2} & -\frac{M_2}{M_1+M_2} & 1 \\ -1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} X_1 \\ X_2 \\ x \end{bmatrix} \tag{2.3}$$

Ce système est bien un changement de coordonnées inversible, car son déterminant vaut

$$-\frac{M_1}{M_1+M_2+m} - \frac{M_2}{M_1+M_2+m} - \frac{M_1 m}{(M_1+M_2+m)(M_1+M_2)} - \frac{M_2 m}{(M_1+M_2+m)(M_1+M_2)} < 0 \tag{2.4}$$

Dans un souci d'efficacité et de reproductibilité on inverse symboliquement le système par le calcul formel avec SageMath :

```
X_1, X_2, x, M_1, M_2, m = var('X_1, X_2, x, M_1, M_2, m')
M = matrix(SR, 3, 3,
    [M_1/(M_1+M_2+m), M_2/(M_1+M_2+m), m/(M_1+M_2+m),
    -M_1/(M_1+M_2), -M_2/(M_1+M_2), 1,
    -1, 1, 0]) # création matrice conversion (X_1, X_2, x) vers (X_G, r, R)
inv(M)
```

de là on déduit

$$\begin{bmatrix} X_1 \\ X_2 \\ x \end{bmatrix} = \begin{pmatrix} 1 & -\frac{m}{M_1+M_2+m} & -\frac{M_2}{M_1+M_2} \\ 1 & -\frac{m}{M_1+M_2+m} & \frac{M_1}{M_1+M_2} \\ 1 & \frac{M_1+M_2}{M_1+M_2+m} & 0 \end{pmatrix} \times \begin{bmatrix} X_G \\ r \\ R \end{bmatrix} \tag{2.5}$$

On peut ensuite expliciter les anciennes coordonnées en fonction des nouvelles :

```
nv_coord = matrix(SR, 3,1, var('X_G, r, R')) # nouveau paramétrage
an_coord = inv(M)*nv_coord
X_1 = an_coord[0,0].factor()
X_2 = an_coord[1,0].factor()
x   = an_coord[2,0].factor()
```

Avec les variables ainsi exprimées, on trouve parallèlement

$$\mathcal{V}_1(x - X_1) = \mathcal{V}_1\left(\frac{M_2R + M_1r + M_2r}{M_1 + M_2}\right) \tag{2.6}$$

$$\mathcal{V}_2(x - X_2) = \mathcal{V}_2\left(-\frac{M_1R - M_1r - M_2r}{M_1 + M_2}\right) \tag{2.7}$$

et

$$\mathcal{W}(X_2 - X_1) = \mathcal{W}(R) \tag{2.8}$$

enfin, en supposant que $\dot{X}_G = 0$, puisque notre système est isolé (cette hypothèse physique sera justifiée dans la section suivante), on peut réévaluer la partie cinétique du hamiltonien (chaque variable portée par un symbole arbitraire v en code ici désigne la quantité $\dot{v}$ en physique) :

```
Ec(X_1, X_2, x) = 1/2*M_1*X_1^2 + 1/2*M_2*X_2^2 + 1/2*m*x^2
Ec(0, r, R) # système isolé
```

qui permet d'écrire $\dot{X}_G = 0 \Rightarrow$

$$\frac{\mathbf{P}_1^2}{2M_1} + \frac{\mathbf{P}_2^2}{2M_2} + \frac{\mathbf{p}^2}{2m} = \frac{\left(M_1^2M_2 + M_1M_2^2 + M_1M_2m\right)\dot{R}^2 + \left(M_1^2m + 2\,M_1M_2m + M_2^2m\right)\dot{r}^2}{\left(M_1 + M_2 + m\right)\left(M_1 + M_2\right)} \tag{2.9}$$

Ainsi en posant en plus $X_G = 0$, on peut réécrire le hamiltonien classique comme fonction de deux variables r, R qui décrivent complètement l'état des trois particules avec μ_R et μ_r des **masses réduites** qui apparaissent dans l'équation 2.9 :

$$\tilde{\mathcal{H}}(r, R, p, P) = \frac{1}{2\mu_R}P^2 + \frac{1}{2\mu_r}p^2 + \underbrace{\mathcal{V}_1\left(\frac{M_2R + M_1r + M_2r}{M_1 + M_2}\right) + \mathcal{V}_2\left(-\frac{M_1R - M_1r - M_2r}{M_1 + M_2}\right) + \mathcal{W}(R)}_{\mathcal{Y}(r,R)} \tag{2.10}$$

où (P, p) sont les variables conjuguées respectives de (R, r), et μ_r et μ_R les masses réduites données par

$$\boxed{\mu_R = \frac{M_1^2M_2 + M_1M_2^2 + M_1M_2m}{\left(M_1 + M_2 + m\right)\left(M_1 + M_2\right)}} \tag{2.11}$$

$$\boxed{\mu_r = \frac{M_1^2m + 2\,M_1M_2m + M_2^2m}{\left(M_1 + M_2 + m\right)\left(M_1 + M_2\right)}} \tag{2.12}$$

D'où l'équivalent quantifié du hamiltonien :

$$\boxed{\tilde{H} = -\frac{1}{2\mu_R}\frac{\partial^2}{\partial R^2} - \frac{1}{2\mu_r}\frac{\partial^2}{\partial r^2} + \mathcal{Y}(r, R)} \tag{2.13}$$

2.3 Choix de fonctions numériques pour le hamiltonien de l'ion H_2^+

Conformément aux sections précédentes et l'introduction du chapitre, on désigne sur l'unique axe les variables-coordonnées scalaires généralisées

- r qui représente la coordonnée de l'électron par rapport au centre de masse ;
- R qui représente le rayon la molécule (distance séparant les deux noyaux).

Le référentiel barycentrique de la molécule est supposé galiléen car on suppose les noyaux non accélérés par des forces externes. On prend en compte les énergies cinétiques des noyaux et de l'électron respectivement dans ce référentiel barycentrique, ainsi qu'un potentiel décrivant leur interaction, qui complique la résolution et empêche *a priori* toute résolution à variables séparées.

On prend donc le hamiltonien 1D non relativiste de 2 coordonnées généralisées r et R indépendant du temps. On pose d'après le paragraphe précédent :

$$M = \frac{2m_p^3 + m_p^2 m_e}{2(2m_p + m_e)m_p} \tag{2.14}$$

et

$$m = \frac{2m_p^2 m_e}{m_p(2m_p + m_e)} \tag{2.15}$$

dans

$$\boxed{\hat{H} = \underbrace{-\frac{1}{2m}\frac{\partial^2}{\partial r^2}}_{\hat{T}_{\text{coord. él.}}} \quad \underbrace{-\frac{1}{2M}\frac{\partial^2}{\partial R^2}}_{\hat{T}_{\text{coord. nucl.}}} + \underbrace{V(r,R)}_{\text{potentiel d'interaction}}} \tag{2.16}$$

qui opère sur $L^2(\mathbb{R} \times \mathbb{R})$.

Pour notre étude, on prendra conventionnellement les potentiels d'interaction

$$\mathcal{V}^{(1)}_{\text{noy.-él.}}(d_{\text{prot.-él.}}) = \mathcal{V}^{(2)}_{\text{noy.-él.}}(d_{\text{prot.-él.}}) = -V_0 \exp\left(-\frac{d^2_{\text{prot.-él.}}}{2\sigma^2}\right) \tag{2.17}$$

qui représente un puits normalisé autour de chaque proton ; ainsi que

$$\mathcal{W}_{\text{prot.-prot.}}(d_{\text{prot.-prot.}}) = \frac{\beta}{\sqrt{\eta^2 + d^2_{\text{prot.-prot.}}}} \tag{2.18}$$

avec β, η, V_0, σ des paramètres à fixer à l'avance empiriquement, qui représente un potentiel électrostatique en $\frac{1}{d_{\text{prot.-prot.}}}$ rectifié (pour ne pas avoir de singularité).

Avec nos choix de masses nucléaires identiques, on a donc

$$V(r,R) = \mathcal{W}_{\text{prot.-prot.}}(R) + \mathcal{V}_{\text{noy.-él.}}\left(r + \frac{R}{2}\right) + \mathcal{V}_{\text{noy.-él.}}\left(r - \frac{R}{2}\right) \tag{2.19}$$

$$\boxed{V(r,R) = -V_0\left[\exp\left(-\frac{(r-\frac{R}{2})^2}{2\sigma^2}\right) + \exp\left(-\frac{(r+\frac{R}{2})^2}{2\sigma^2}\right)\right] + \frac{\beta}{\sqrt{\eta^2 + R^2}}} \tag{2.20}$$

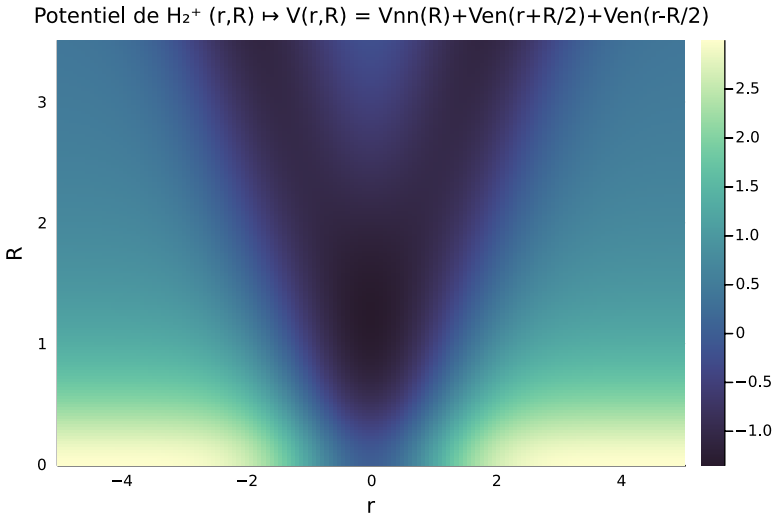


Figure 2.2 : Potentiel choisi pour la résolution avec les paramètres indiqués. Physiquement on retrouve bien notre modélisation : à R fixé, deux puits de potentiel sont visible selon l'axe des r, qui correspondent aux deux protons.

ou de manière plus compacte :

$$V(r,R) = -2V_0 \exp\left(-\frac{1}{2\sigma^2}\left(r^2 + \frac{R^2}{4}\right)\right)\cosh(rR) + \frac{\beta}{\sqrt{\eta^2 + R^2}} \tag{2.21}$$

2.3.1 Choix des paramètres

Pour la suite, je prendrai les paramètres $\beta = 1.5, \eta = 0.5, V_0 = 1.5, \sigma = 1$**,** $m_e = 1, m_p = 6$**.** Cela permet d'avoir quelques états propres de valeur propre négative, et jouer sur les paramètres permettrait de piloter le spectre du hamiltonien associé au potentiel. Les figures de ce rapport sont générées avec ces paramètres de sorte que la carte du potentiel choisi est donnée en figure 2.2.

2.4 Séparation des termes et approximation de Born-Oppenheimer

En règle générale, il est commode d'avoir un hamiltonien séparable, par exemple de la forme

$$\hat{H} = \hat{H}_{\text{él.}} + \hat{T}_{\text{noy.}} \tag{2.22}$$

Pour ce faire, on mettra en œuvre l'approximation de Born-Oppenheimer en découplant la dynamique noyau-électron du fait du grand rapport de masse entre l'électron et les noyaux. Les prémices de cette technique ont été proposées en 1927 par les deux physiciens éponymes dans l'article intitulé « Zur Quantentheorie der Molekln » (*Sur la théorie quantique des molécules*) [BO27]. L'hypothèse $m_p \gg m_e$ permet de supposer les noyaux fixes, ce qui permet de considérer R comme un hyperparamètre de notre problème, que l'on trouve d'abord en minimisant l'énergie fondamentale de la fonction d'onde de l'électron à R fixé, qui est appelée **énergie potentielle de surface**[1]. Une énergie potentielle de surface est usuellement définie comme l'énergie d'inter-

[1] Dans des situations où la molécule comprend beaucoup d'atomes, cette énergie de surface est une fonction des paramètres géométriques de la molécule, et l'analogie avec une « surface », ou un relief décrivant cette énergie en fonction de sa forme géométrique, donne sens à cette dénomination.

action d'une collection d'atomes fixes en configuration fixée donnée par des coordonnées d'espace généralisées. En chimie moléculaire il est utile de faire varier de manière infinitésimale ces coordonnées nucléaires (correspondant à R dans notre cas) puis de trouver les valeurs propres du système ainsi simplifié par rapport aux électrons, pour obtenir cette « carte » de l'énergie en fonction des positions nucléaires et obtenir ensuite la configuration la plus stable d'énergie minimale, car les propriétés de l'énergie de surface au voisinage de ce minimum régissent les spectres vibrationnel, rotationnel, et la stabilité de la conformation moléculaire.

Dans notre cas, la surface que l'on parcourt est une surface 1D puisqu'on a seulement un hyperparamètre, R. Ensuite, on approche la perturbation du hamiltonien au voisinage de R_0.

Cette approche est motivée par la connaissance des solutions analytiques (ou mêmes numériques) pour l'oscillateur harmonique quantique 1D, et l'expression de solutions analytiques à variables séparables pour le cas 2D, par opposition à trouver les états propres non séparables pour des systèmes 2D qui est souvent impossible analytiquement et représente une complexité de calcul bien supérieure numériquement.

On va donc faire apparaître deux oscillateurs harmoniques découplés. On va considérer, à $R \in \mathbb{R}$ fixé, l'énergie minimale $\mathcal{E}_0(R)$ du hamiltonien opérant sur la coordonnée électronique r uniquement :

$$\hat{H}_R = -\frac{1}{2m}\frac{\partial^2}{\partial r^2} + V(r, R) \tag{2.23}$$

dont on va trouver l'état fondamental d'énergie notée $E_0(R)$. Suivant cette méthode dans le cas qui nous intéresse, on définit donc

$$R_0 = \underset{R\in\mathbb{R}}{\operatorname{argmin}}\, E_0(R) = \underset{R\in\mathbb{R}}{\operatorname{argmin}} \left(\min \operatorname{Sp}(\hat{H}_R)\right) \tag{2.24}$$

avec à R fixé, le minimum du spectre bien défini à cause des théorèmes évoqués au chapitre précédent. L'existence d'un réel R minimisant l'énergie fondamentale n'est pas évidente ici avec mes connaissances en théorie des opérateurs.

2.4.1 Obtention de l'argument R minimisant l'énergie de surface

Théorème d'Ehrenfest pour la dérivée

Pour obtenir l'énergie de surface minimale E_0 et le paramètre R_0 permettant de l'atteindre, plusieurs méthodes s'offrent à nous. Il s'agit de minimiser une fonction, problème classique en informatique.

Nous allons exploiter une propriété intéressante que j'introduis pour le moment qualitativement : **lorsque l'on perturbe légèrement un hamiltonien, la modification infinitésimale (à l'ordre 1) sur l'état normé réel (donc le vecteur propre) d'un mode non dégénéré est orthogonale à celui-ci**. Ce résultat est bien connu, la preuve sera reproduite plus loin à la section 2.6.2.

Par conséquent, si on se donne un hamiltonien dépendant d'un paramètre λ et qu'on se place en un point λ, avec l'état fondamental $\psi(\lambda)$ et son énergie $E(\lambda)$, la dérivée l'observable énergie par rapport à λ est :

$$\frac{\mathrm{d}E}{\mathrm{d}\lambda}(\lambda_0) = \underbrace{2\left\langle \frac{\mathrm{d}\psi}{\mathrm{d}\lambda}(\lambda_0) \middle| \underbrace{H(\lambda_0)|\psi(\lambda_0)}_{E(\lambda_0)\psi(\lambda_0)} \right\rangle}_{=0} + \left\langle \psi(\lambda_0) \middle| \frac{\mathrm{d}H}{\mathrm{d}\lambda}(\lambda_0) \middle| \psi(\lambda_0) \right\rangle \tag{2.25}$$

L'égalité 2.25 est un cas particulier du théorème d'Ehrenfest, qui donne la dérivée d'un observable sans avoir à connaître la dérivée de l'état.

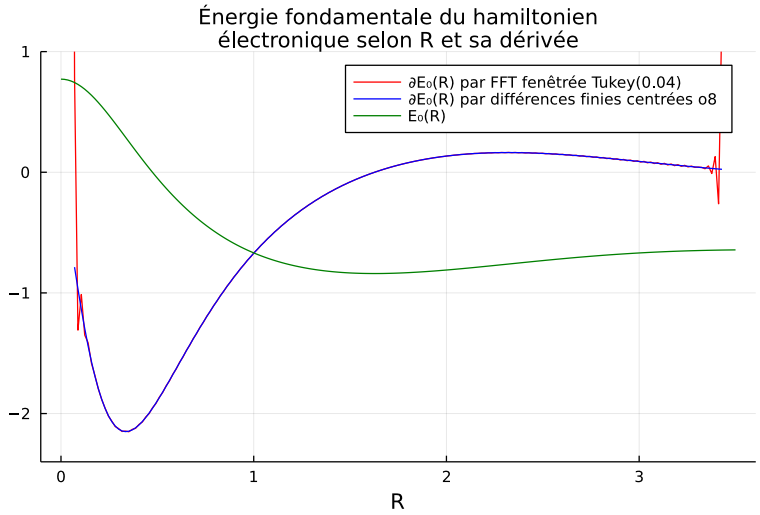


Figure 2.3 : Énergie fondamentale $R \mapsto E_0(R)$ et sa dérivée approchée de deux manières différentes en fonction de l'hyperparamètre R (distance entre les 2 nucléons). Chacune a un inconvénient : une dérivée fréquentielle sera difficilement précise en raison des parasites de haute fréquence dûe à la faible longueur du signal, même avec un fenêtrage et les différences finies ont un ordre de précision limité. Les deux nécessitent un échantillonnage qui est totalement bloquant si l'on veut évaluer la dérivée en un point précis hors de l'échantillonnage. Ce ne seront donc pas les approches choisies pour la descente de gradient dans la section 2.4.1.

Choix de l'algorithme d'approche : une méthode de quasi-Newton

On est donc dans un cas courant, en optimisation, où on connaît la fonction et sa dérivée (exprimée elle-même à l'aide de l'état, voire figure 2.3).

Partant de ceci, de nombreuses solutions s'offrent à nous[2] (liste non exaustive) :

1. Une dichotomie simple ;
2. Une descente de gradient sur $E(\lambda)$ parmi les nombreuses adaptations existantes selon l'adaptation des pas à chaque étape comme RMSprop, BFGS, ADA-{Max, M, grad, …} qui apparaissent dans la littérature florissante en Machine Learning notamment ;
3. La méthode de recherche de 0 de Newton-Raphson appliquée à $\frac{\mathrm{d}E}{\mathrm{d}\lambda}$ (nécéssiterait l'accès à $\frac{\mathrm{d}^2E}{\mathrm{d}\lambda^2}$) ;
4. Une méthode de recherche de 0 parmi lesdites quasi-Newton (Broyden, Davidon-Fletcher-Powell, etc.), méthode des sécantes, etc., appliquée à $\frac{\mathrm{d}E}{\mathrm{d}\lambda}$ qui s'affranchissent de la connaissance exacte de la jacobienne ou de la hessienne de $\frac{\mathrm{d}E}{\mathrm{d}\lambda}$.

Une solution évidente est la dichotomie pour trouver un zéro de la dérivée de $E(\lambda)$. Cette méthode présente une convergence linéaire (à chaque étape l'erreur est divisée par deux), c'est-à-dire que l'erreur suit une progression géométrique et on peut atteindre la précision machine en un temps logarithmique (mesuré en nombre d'étapes).

[2] Voir également https://ruder.io/optimizing-gradient-descent/index.html

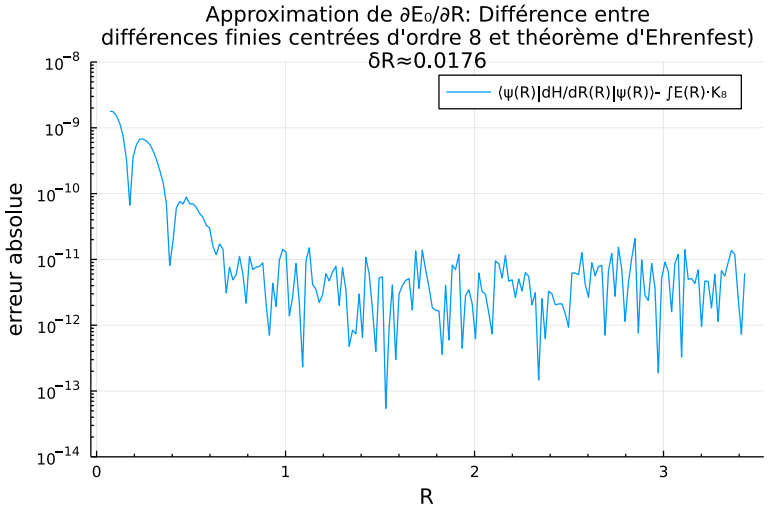


Figure 2.4 : Différence entre la dérivée de $E_0(R)$ calculée en différences finies centrées d'ordre 8 et avec le théorème d'Ehrenfest.

Quitte à devoir évaluer l'état et l'énergie (le plus coûteux) pour obtenir la dérivée (le moins coûteux une fois les précédents acquis) avec 2.25, autant utiliser la méthode des sécantes qui converge plus rapidement qu'une dichotomie. En voici un pseudo-code élémentaire :

Inputs x_0, x_1, f, ε
Processing $x_{k-1} := x_0$
$x_k := x_1$
$x_m := x_1$
while $|x_k - x_{k-1}| > \varepsilon$ do
$$x_k := \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)$$
$x_{k-1} := x_m$
end while
Outputs x_k

Il s'agit dans l'idée de la métode de Newton avec une approximation de la dérivée de f.

Compte tenu du caractère un peu particulier de la fonction $R \mapsto E_0(R)$ dont on ne connaît pas de forme algébrique close (mais qui nécessite l'appel à un algorithme d'éléments propres), je n'utiliserai pas les paquets Julia comme `Optim` ou `Roots`, mais je coderai la méthode des sécantes à la main appliquée à $\frac{\mathrm{d}E}{\mathrm{d}R}$. De même, puisqu'il sera nécessaire d'obtenir une évaluation assez précise de la dérivée de $R \mapsto E(R)$ en tous les points itérés de l'algorithme, il est malaisé d'évaluer ces dérivées par des méthodes de traitement du signal (comme les différences finies ou la dérivation fréquentielle avec `FFTW` et `DSP` qui a été testée, et qui nécessite d'ailleurs un fenêtrage) car elles se basent sur un échantillonnage. Utiliser ainsi le théorème d'Ehrenfest est par conséquent tout-à-fait justifié et même nécessaire. Voir figure 2.4.

L'algorithme 1 résume la démarche.

Algorithm 1 Calcul du paramètre nucléaire minimisant l'énergie fondamentale

Input : $\Lambda_r :\in \mathscr{M}_{n,n}(\mathbb{R})$; $R_g \in \mathbb{R}$ (laplacien suivant l'axe des r) ; $R^\dagger, R^\ddagger \in \mathbb{R}$ (points de départ) ; $\varepsilon > 0$ (précision) ; $\mathcal{V} : \mathbb{R} \to \mathbb{R}$ (fonction potentiel d'interaction noyau-électron) ; $\mathcal{V}'$ (sa dérivée) ; $\mathbf{r}_s$ (vecteur ordonné des r sur la grille)
Output : $R_0 = \operatorname{argmin}_{R\in\mathbb{R}} E_0(R)$ à ε près

procedure Méthode des sécantes appliquée à $\frac{\mathrm{d}E}{\mathrm{d}R}$
  $\partial E^\ddagger \leftarrow \varepsilon + 1$
  while $|\partial E^\ddagger| > \varepsilon$ **do**
    $\hat{H}^\dagger \leftarrow \Lambda_r + \operatorname{diag}\left(\mathcal{V}(\mathbf{r}_s + \frac{R^\dagger}{2}) + \mathcal{V}(\mathbf{r}_s - \frac{R^\dagger}{2})\right)$
    $\hat{H}^\ddagger \leftarrow \Lambda_r + \operatorname{diag}\left(\mathcal{V}(\mathbf{r}_s + \frac{R^\ddagger}{2}) + \mathcal{V}(\mathbf{r}_s - \frac{R^\ddagger}{2})\right)$
    $\psi^\dagger \leftarrow \operatorname{argmin} \operatorname{Sp}(\hat{H}^\dagger)$ ▷ Avec `KrylovKit`
    $\psi^\ddagger \leftarrow \operatorname{argmin} \operatorname{Sp}(\hat{H}^\ddagger)$
    $\partial\hat{H}^\dagger \leftarrow \operatorname{diag}\left(\frac{1}{2}\mathcal{V}'(\mathbf{r}_s + \frac{R^\dagger}{2}) - \frac{1}{2}\mathcal{V}'(\mathbf{r}_s - \frac{R^\dagger}{2})\right)$
    $\partial\hat{H}^\ddagger \leftarrow \operatorname{diag}\left(\frac{1}{2}\mathcal{V}'(\mathbf{r}_s + \frac{R^\ddagger}{2}) - \frac{1}{2}\mathcal{V}'(\mathbf{r}_s - \frac{R^\ddagger}{2})\right)$
    $\partial E^\dagger \leftarrow \langle\psi^\dagger|\partial\hat{H}^\dagger|\psi^\dagger\rangle$ ▷ Optimisé avec `dot`
    $\partial E^\ddagger \leftarrow \langle\psi^\ddagger|\partial\hat{H}^\ddagger|\psi^\ddagger\rangle$
    $(R^\ddagger, R^\dagger) \leftarrow \left(R^\ddagger - \frac{R^\ddagger - R^\dagger}{\partial E^\ddagger - \partial E^\dagger}\partial E^\ddagger, R^\ddagger\right)$ ▷ Descente du gradient de $\frac{\mathrm{d}E}{\mathrm{d}R}$
  end while
end procedurereturn $R^\ddagger$

Remarque 2.4.1. *En pratique, on peut baisser la complexité mémoire de l'algorithme 1 en panachant les calculs de chaque hamiltonien et de sa dérivée à stocker dans une même variable avant son utilisation. C'est ce que j'ai fait en pratique mais pour la compréhension visuelle, j'ai laissé l'algorithme tel quel ici. Ci-après, le code Julia correspondant avec ces optimisations pour mémoire.*

Code Julia de l'algorithme 1

```
Hr = copy(Λr); # initialisation opérateurs
dHr = Diagonal(zeros(N));
iter_nb = 0;
dE_rp = ϵ +1; # initialisation pour rentrer dans la boucle

while (iter_nb < itermax && abs(dE_rp) > ϵ)
    # calcul des états
    Hr[diagind(Hr)] = diag(Λr) + Vector(V_nucl_el.(rs, R_rp)) .+ V_nucl_nucl(R_rp);
    vals, vecs, infos = KrylovKit.eigsolve(Hr, N, 1, :SR, krylovdim=kdim1d,
    ↪  issymmetric=true, ishermitian=true);
    @assert infos.converged > 0;
    ψ_rp = Vector(vecs[1]);

    Hr[diagind(Hr)] = diag(Λr) + Vector(V_nucl_el.(rs, R_lp)) .+ V_nucl_nucl(R_rp);
    vals, vecs, infos = KrylovKit.eigsolve(Hr, N, 1, :SR, krylovdim=kdim1d,
    ↪  issymmetric=true, ishermitian=true);
    @assert infos.converged > 0;
    ψ_lp = Vector(vecs[1]);

    # calcul des dérivées de l'énergie
    dHr[diagind(dHr)] = dR_V_nucl_el.(rs, R_rp) .+ dV_nucl_nucl(R_rp);
    dE_rp = dot(ψ_rp,dHr,ψ_rp);


    dHr[diagind(dHr)] = dR_V_nucl_el.(rs, R_lp) .+ dV_nucl_nucl(R_lp);
    dE_lp = dot(ψ_lp,dHr,ψ_lp);

    # méthode des sécantes pour la dérivée de l'énergie par rapport à R
    R_rp_mem = R_rp;
    step = dE_rp*(R_rp-R_lp)/(dE_rp-dE_lp);
    R_rp -= step;
    R_lp = R_rp_mem;
    iter_nb += 1;

    @show iter_nb, step, dE_rp;
end
```

2.4.2 Scission du potentiel en un hamiltonien non perturbé et une perturbation

Dorénavant j'effectue le remplacement[3] de noms des variables $r \rightsquigarrow x$ et $R \rightsquigarrow y$ (et $R_0 = y_0$).

Le point charnière de ce stage se résume dans la décomposition suivante (proposée par le tuteur scientifique) :

$$\begin{aligned}\hat{H} = E_0(y_0) - \frac{1}{2M}\frac{\partial^2}{\partial y^2} + \frac{1}{2}\frac{\mathrm{d}^2 E_0}{\mathrm{d}y^2}(y_0)(y-y_0)^2 \\ - \frac{1}{2m}\frac{\partial^2}{\partial x^2} + V(x,y_0) - E_0(y_0) \\ + V(x,y) - V(x,y_0) - \frac{1}{2}\frac{\mathrm{d}^2 E_0}{\mathrm{d}y^2}(y_0)(y-y_0)^2 \end{aligned} \tag{2.26}$$

Le hamiltonien de départ $\hat{H}$ se décompose donc sous la forme de deux hamiltoniens non perturbés $\hat{H}_y^0$ et $\hat{H}_x^0$ agissant chacun respectivement sur une coordonnée généralisée (chacune

[3] Pour éviter toute ambiguïté liée à des erreurs de typographie

sur la même variable d'espace, selon un axe unidimensionnel), et un terme de couplage entre ces deux coordonnées $\hat{W}$, que l'on traitera comme une perturbation :

$$\hat{H} = \hat{H}_y^0 + \hat{H}_x^0 + \hat{W} \tag{2.27}$$

$$\begin{cases} \hat{H}_y^0 & = -\frac{1}{2M}\frac{\partial^2}{\partial y^2} + \frac{1}{2}\frac{\mathrm{d}^2 E_0}{\mathrm{d}y^2}(y_0)(y-y_0)^2 + E_0(y_0) \\ \hat{H}_x^0 & = -\frac{1}{2m}\frac{\partial^2}{\partial x^2} + V(x,y_0) - E_0(y_0) \\ \hat{W} & = V(x,y) - V(x,y_0) - \frac{1}{2}\frac{\mathrm{d}^2 E_0}{\mathrm{d}y^2}(y_0)(y-y_0)^2 \end{cases} \tag{2.28}$$

Ensuite, on se place en (x, y_0) pour écrire le développement limité de V dans le terme W :

$$V(x,y) - V(x,y_0) = \frac{\partial V}{\partial y}(x,y_0)(y-y_0) + \frac{1}{2}\frac{\partial^2 V}{\partial y^2}(x,y_0)(y-y_0)^2 + \underset{y\to y_0}{o}(y-y_0)^2 \tag{2.29}$$

Finalement,

$$\boxed{\hat{W} = \frac{\partial V}{\partial y}(x,y_0)(y-y_0) + \frac{1}{2}\left(\frac{\partial^2 V}{\partial y^2}(x,y_0) - \frac{\mathrm{d}^2 E_0}{\mathrm{d}y^2}(y_0)\right)(y-y_0)^2 + \text{h.o.t.}} \tag{2.30}$$

2.5 Changement de variable et adimensionnement

On peut alors traiter ce problème en essayant de trouver de manière approchée, une solution au problème simplifié $\hat{H} - \text{h.o.t.} = \hat{H}_y^0 + \hat{H}_x^0 + \hat{W} - \text{h.o.t.}$ pour lequel on néglige les termes d'ordre supérieurs ou égal à 3. Cela revient donc à prendre en lieu et place de $\hat{W}$, la perturbation tronquée $\tilde{\hat{W}}$ de ces termes suivante :

$$\tilde{\hat{W}} = \frac{\partial V}{\partial y}(x,y_0)(y-y_0) + \frac{1}{2}\left(\frac{\partial^2 V}{\partial y^2}(x,y_0) - \frac{\mathrm{d}^2 E_0}{\mathrm{d}y^2}(y_0)\right)(y-y_0)^2 \tag{2.31}$$

On remarque d'ailleurs que la perturbation W est elle-même une somme de termes séparable en les variables x et y.

À partir de là, on traite le terme $\hat{H}_x^0$ du hamiltonien non perturbé analytiquement puisqu'il s'agit d'un oscilateur harmonique, le terme $\hat{H}_y^0$ numériquement, à partir desquels on construit une solution $\Psi^{\text{HBO}}(x,y) = \psi_{y_0}^0(x)\phi^0(y)$ (HBO signifie *Harmonic Born-Oppenheimer*.) Puis, on applique la théorie des perturbations à Ψ^{HBO} et $\tilde{\hat{W}}$.

Nous aurons donc deux leviers d'action sur le développement : l'ordre du DSE en espace, et l'ordre de la perturbation.

Erreur commise Dans cette démarche on commet des erreurs dans l'approximation de l'état fondamental à trois niveaux :

1. En discrétisant l'espace de dimension infinie dans lequel existe la fonction d'onde, et en l'approchant par un « représentant »[4] dans un espace de dimension finie (qui ne recouvre pas tout l'espace), ainsi que ses dérivées (approchées par différences finies) ;
2. Lors de l'approximation de la perturbation $\hat{W}$ par $\tilde{\hat{W}}$ avec son développement limité ;
3. Lors de l'approximation de la fonction d'onde en tronquant le développement de la perturbation à un ordre fini.

[4] Abus de langage mathématique.

Adimensionnement du problème suivant la dimension nucléaire

Le but étant d'étudier la validité de Born-Oppenheimer lorsque la masse des protons devient grande devant celle de l'électron, on verra la fonction d'onde s'écraser suivant une dimension, car les protons deviennent peu à peu classiques. Pour ne pas impacter la simulation numérique, on va adimensionner le problème suivant la variable R.

Dans l'équation 2.28, on a vu que le hamiltonien suivant l'axe (Oy) (rappelons-le, $y \equiv R$), il s'agit d'un oscillateur. Comme vu dans le théorème 1.1.1, la solution fondamentale de l'oscillateur est une gaussienne de longueur caractéristique $\sqrt{\frac{2\hbar}{M\omega^2}}$ (cf. équation 1.17).

Pour adimensionner avec notre cas, à savoir la constante de raideur $K = M\omega^2 = \frac{\mathrm{d}^2 E_0}{\mathrm{d}y^2}(y_0)$, il est donc naturel[5] de poser $y \equiv R = R_0 + \varepsilon u$ avec $\varepsilon = (MK)^{-1/4}$, on a ainsi centré le problème autour de la distance nucléaire d'équilibre.

Les hamiltoniens 2.28 reparamétrés deviennent

$$\begin{cases} \mathfrak{h}_u &= K\varepsilon^2 \left(-\frac{1}{2}\frac{\partial^2}{\partial u^2} + \frac{1}{2}u^2\right) + E_0(R_0) \\ \mathfrak{h}_x &= -\frac{1}{2m}\frac{\partial^2}{\partial x^2} + V(x, R_0) - E_0(R_0) \qquad \text{(inchangé : } \mathfrak{h}_x = \hat{H}^0_x\text{)} \\ \mathfrak{w} &= V(x, R_0 + \varepsilon u) - V(x, R_0) - \frac{\varepsilon^2 K}{2}u^2 \end{cases} \tag{2.32}$$

puis le DSE en espace de la perturbation ici à l'ordre 2 :

$$\hat{\mathfrak{w}} = \varepsilon \frac{\partial V}{\partial y}(x, y_0)u + \frac{1}{2}\varepsilon^2 \left(\frac{\partial^2 V}{\partial y^2}(x, y_0) - K\right) u^2 \tag{2.33}$$

Par conséquent on a mis au point deux paramètres déterminants pour notre approximation : ε et l'ordre du DL pour $\mathfrak{w}$, sur lesquels on peut jouer.

2.6 Théorie des perturbations à tout ordre sur le système d'intérêt

2.6.1 Théorie des perturbations : présentation

Je présente ici une méthode d'approximation d'états propres en physique qui très connue et présente en particulier dans beaucoup d'ouvrages de mécanique quantique. Elle n'est pas exclusive à ce domaine, et repose sur des théorèmes mathématiques qui sont présentés ultérieurement. Elle trouve naturellement beaucoup d'applications en physique de manière générale. Les résultats qui suivent ne sont ainsi pas inédits : je reprends l'état de l'art en redémontrant certains points plus précisément que certains ouvrages.

Supposons que l'on dispose d'un hamiltonien $\hat{H} = \hat{H}^0 + \lambda\hat{W}$ dont les solutions stationnaires pour $\hat{H}^0$ sont connues et discrètes, notées $(\mathcal{E}_k, |\phi_k\rangle)_{k\in\mathbb{N}}$, et formant une base de l'espace hilbertien auquel ils appartiennent. La théorie des perturbations suppose que l'on peut approcher la solution de $\hat{H}$ à partir de celles de $\hat{H}^0$. Plus précisément, on fait l'hypothèse que le j-ième mode perturbé $(E(\lambda), |\psi(\lambda)\rangle)$ de $\hat{H}$ avec $|\psi(\lambda)\rangle$ **normé**, s'obtient par un développement en séries de Taylor :

$$E(\lambda) = \sum_{k=0}^{+\infty} \lambda^k E_k \qquad |\Psi(\lambda)\rangle = \sum_{k=0}^{+\infty} \lambda^k |\psi_k\rangle \tag{2.34}$$

où les $(E_k, |\psi_k\rangle)$ sont à déterminer. En général, dans la littérature, on se contente de l'ordre 1 ou 2, surtout pour un espace hilbertien de dimension infinie, et cela donne lieu à ce qu'on appelle

[5] C'est aussi ce que l'on trouve avec l'approximation BKW (1.23).

une *contamination* de l'état non perturbé considéré par les autres non perturbés après ajout de la perturbation.

Le fait que les états (resp. les énergies) soient développables en série entière autour de l'état non perturbée (resp. l'énergie non perturbée) n'est pas évident. En dimension finie, cela impose la différentiabilité de $\hat{H} \mapsto (E_j, |\psi_j\rangle)$ dans la direction $\hat{W}$ en $\hat{H}^0$ à tout ordre, mais [CTDL21a] admet cela comme une hypothèse commode.

Notons que pour que ces calculs soient exacts, il est nécessaire que la valeur propre E_j considérée ne soit pas dégénérée (de multiplicité au plus 1). Ce n'est pas très gênant pour les cas qui nous intéressent dans le cadre de ce stage, puisque nous l'appliquerons surtout à la plus petite valeur propre d'un système pour connaître son état fondamental, non dégénéré pour le hamiltonien qui nous intéresse.

En fait, cette hypothèse de développement en série entière est justifiée, au moins en dimension finie pour l'énergie dans [RS80b], le théorème XII.1 stipule que :

Théorème 2.6.1. *([RS80b], théorème XII.1) Si $F(\beta, \gamma) = \gamma^n + a_1(\beta)\gamma^{n-1} + \cdots + a_n(\beta)$ est un polynôme de degré n dont les coefficients sont analytiques en β (i.e., développables en série entière sur une boule de rayon non nul autour de tout point de $\mathbb{C}$), et soit γ_0 une racine simple de $\gamma \mapsto F(\beta_0, \gamma)$ pour un certain β_0, alors pour β suffisamment proche de β_0, il existe une unique racine $\gamma(\beta)$ de la fonction $\gamma \mapsto F(\beta, \gamma)$ proche de γ_0 et $\beta \mapsto \gamma(\beta)$ est analytique en β autour de β_0.*

Si on applique ce théorème au polynôme caractéristique $\det(\gamma I_n - (H_0 + \beta W))$ qui est de coefficient de terme de plus haut degré égal à 1 (en γ), on obtient que l'énergie propre obtenue suite à une perturbation « petite » du hamiltonien, est bien dévelopable en séries entière autour de l'énergie propre non-dégénérée correspondante au hamintonien non perturbé et le DSE de l'énergie dont les coefficients sont appelés coefficients de Rayleigh-Schrödinger. Ce même livre donne aussi un théorème justifiant l'existence du DSE pour le vecteur propre correspondant (théorème II.4, [RS80b]), que j'élude ici.

Remarquons que plus simplement, si on l'applique à $\det(\gamma I_n - H(\beta))$ avec H analytique en β, on obtient la condition de régularité nécessaire pour l'algorithme des sécantes fait à la section précédente, puisque la fonction de potentiel choisie dans 2.20 est analytique en R. Ainsi $R \mapsto E_0(R)$ est au moins de classe $\mathscr{C}^2$ ce qui permet d'appliquer la méthode des sécantes.

Identification à chaque ordre des termes de la série

Pour déterminer les $(E_k, |\psi_k\rangle)$, on écrit les égalités des termes de même ordre dans l'équation d'une perturbation en λ :

$$(\hat{H}^0 + \lambda \hat{W}) \left(\sum_{k=0}^{+\infty} \lambda^k |\psi_k\rangle \right) = \left(\sum_{k=0}^{+\infty} \lambda^k E_k \right) \left(\sum_{k=0}^{+\infty} \lambda^k |\psi_k\rangle \right) \tag{2.35}$$

Pour déterminer successivement les énergies propres et vecteurs propres de ce développement, on commence par identifier les termes d'ordre 1, puis 2, puis 3, etc., ce qui est licite, au moins en dimension finie, en raison de l'unicité des coefficients d'une série entière. Cela donne successivement comme dans [CTDL21a] :

$$(\hat{H}^0 - E_0)|\psi_0\rangle = 0 \,(\text{ordre } 0) \tag{2.36}$$

$$(\hat{H}^0 - E_0)|\psi_1\rangle + (\hat{W} - E_1)|\psi_0\rangle = 0 \,(\text{ordre } 1) \tag{2.37}$$

$$(\hat{H}^0 - E_0)|\psi_2\rangle + (\hat{W} - E_1)|\psi_1\rangle - E_2|\psi_0\rangle = 0 \,(\text{ordre } 2) \tag{2.38}$$

$$\vdots \tag{2.39}$$

$$(\hat{H}^0 - E_0)|\psi_q\rangle + (\hat{W} - E_1)|\psi_{q-1}\rangle - E_2|\psi_{q-2}\rangle + \cdots - E_q|\psi_0\rangle = 0 \,(\text{ordre } q) \tag{2.40}$$

Si l'on cherche à estimer un état non dégénéré $(\mathcal{E}_j, |\phi_j\rangle)$ après perturbation en le choisissant réel (phase nulle) et normé, on montre ainsi, en faisant $\lambda \to 0$, que $E_0 = \mathcal{E}_j$ et $|\psi_0\rangle = |\phi_j\rangle$ ce qui permet d'écrire plus précisément

$$(\hat{H}^0 + \lambda \hat{W}) \left(|\phi_j\rangle + \sum_{k=1}^{+\infty} \lambda^k |\psi_k\rangle \right) = \left(\mathcal{E}_j + \sum_{k=1}^{+\infty} \lambda^k E_k \right) \left(|\phi_j\rangle + \sum_{k=1}^{+\infty} \lambda^k |\psi_k\rangle \right) \tag{2.41}$$

c'est-à-dire que l'on a déjà déterminés $|\psi_0\rangle$ et E_0. Il ne reste plus qu'à trouver les autres.

2.6.2 Détermination des termes grâce à la méthode de la sum over states

On va montrer que cette méthode est peu adaptée en prenant l'exemple de l'oscillateur harmonique quantique 2D.

À partir d'ici, je vais décrire et prouver un algorithme à l'aide des formules systématiques à tout ordre, comme dans [CTDL21a] pour déterminer les états et énergies des développements de Taylor. Ce n'est donc pas un travail inédit, mais je m'efforce de justifier un peu mieux les égalités que ce que j'ai pu lire dans la littérature.

Détermination des termes de l'état propre

Pour exprimer les états $|\psi_k\rangle$, on va les décomposer dans la base orthonormale des états propres du hamiltonien non perturbé $|\phi_k\rangle$, les vecteurs $|\phi_j\rangle$, ce qui est toujours faisable car on a supposé être en dimension finie et s'être muni d'une base de vecteurs propres de cet espace.

Notons donc pour tout ordre q,

$$|\psi_q\rangle = \sum_k \alpha_k^{(q)} |\phi_k\rangle; \quad \alpha_k^{(q)} = \langle \psi_q | \phi_k \rangle \tag{2.42}$$

On détermine ensuite les $(\alpha_k^{(q)}), 1 \leq k \leq \dim \mathscr{H}, q \in \mathbb{N}$ par récurrence sur q. L'hypothèse de choix $|\psi(\lambda)\rangle$ **réel normé** est fondamentale car elle permet dans les calculs suivants de rendre la symétrie au produit scalaire de deux états et d'écrire pour tout q en développant $1 = \left\| \mathcal{O}(\lambda^{q+1}) + \sum_{i=0}^{q} \lambda^i \psi_i \right\|^2$, que :

$$\sum_{s=1}^{q} \lambda^s \sum_{i=0}^{s} \langle \psi_i | \psi_{s-i} \rangle + \mathcal{O}(\lambda^{q+1}) = 0 \tag{2.43}$$

et on impose donc

$$\sum_{s=1}^{q} \lambda^s \sum_{i=0}^{s} \langle \psi_i | \psi_{s-i} \rangle = 0 \tag{2.44}$$

- Dans un premier temps, Pour déterminer $\alpha_j^{(q)}$ (rappelons que j est l'indice du mode que l'on essaie d'estimer après perturbation), on a recours à l'astuce suivante comme on trouve dans la littérature, dont je rédige la preuve plus précisément et à tout ordre :

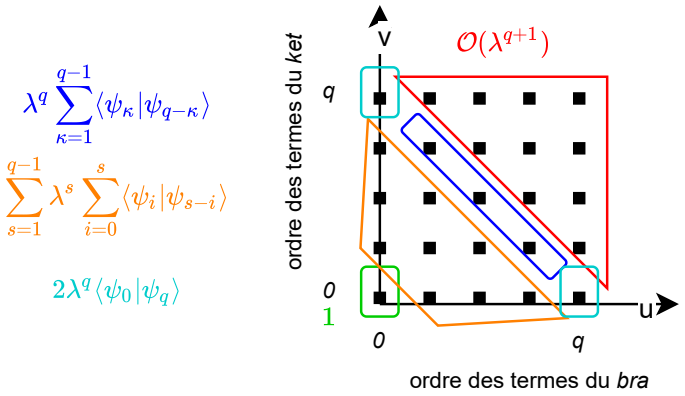


Figure 2.5 : Développement des termes de $\langle\psi(\lambda)|\psi(\lambda)\rangle$ à l'ordre q.

– Pour l'ordre 1, en écrivant $1 = \langle\psi(\lambda)|\psi(\lambda)\rangle = \langle\psi_0|\psi_0\rangle + \lambda\langle\psi_1|\psi_0\rangle + \lambda\langle\psi_0|\psi_1\rangle + \mathcal{O}(\lambda^2) = 1$ sachant que $|\psi_0\rangle = |\phi_j\rangle$ et que tous les états non perturbés sont normés et réels, et que les $|\psi\rangle$ sont également choisis réels, en faisant tendre λ vers 0, on déduit par croissance comparée que $2\Re(\langle\psi_0|\psi_1\rangle) = 2\langle\psi_1|\psi_0\rangle = 0$ autrement dit, $|\psi_1\rangle$ est orthogonal à $|\psi_0\rangle$, il n'y a pas de composante sur $|\phi_j\rangle$ à l'ordre 1, soit encore

$$\boxed{\alpha_j^{(1)} = 0} \tag{2.45}$$

– Pour l'ordre $q \geq 2$, en suivant le groupement des termes présenté dans la figure 2.5 :

$$1 = \langle\psi(\lambda)|\psi(\lambda)\rangle = \left\| \mathcal{O}(\lambda^{q+1}) + \sum_{i=0}^{q} \lambda^i \psi_i \right\|^2 = \mathcal{O}(\lambda^{q+1}) + 2\lambda^q\langle\psi_0|\psi_q\rangle + \lambda^q \sum_{i=1}^{q-1} \langle\psi_i|\psi_{q-i}\rangle$$
$$+ \underbrace{\langle\psi_0|\psi_0\rangle}_{=1} + \sum_{s=1}^{q-1} \sum_{\substack{u,v\geq 0 \\ u+v=s}} \lambda^s \langle\psi_u|\psi_v\rangle \tag{2.46}$$

On peut alors utiliser 2.44 dans le dernier terme de 2.46 et on obtient donc après soustraction de 1 des deux côtés et réarrangement :

$$2\lambda^q\langle\psi_0|\psi_q\rangle = -\lambda^q \sum_{i=1}^{q-1} \langle\psi_i|\psi_{q-i}\rangle + \mathcal{O}(\lambda^{q+1}) \tag{2.47}$$

Il ne reste plus qu'à diviser cette égalité par λ^q, puis en faisant $\lambda \to 0$ on obtient :

$$\boxed{\alpha_j^{(q)} = \langle\phi_j|\psi_q\rangle = -\frac{1}{2} \sum_{i=1}^{q-1} \langle\psi_i|\psi_{q-i}\rangle} \tag{2.48}$$

et on remarque que la formule reste en fait valable pour $q = 1$ (somme vide donne résultat nul).

- Dans un second temps on détermine les $\alpha_k^{(q)}$ avec $k \neq j$:

– Pour déterminer $\alpha_k^{(1)}$ on reprend 2.37 :

$$\langle\phi_k|\hat{H}^0 - E_0|\psi_1\rangle + \langle\phi_k|\hat{W} - E_1|\psi_0\rangle = 0 \tag{2.49}$$

ce qui permet d'extraire, en se souvenant que $E_0 = \mathcal{E}_j$ et que les ϕ_k sont vecteurs propres de $\hat{H}^0$, et que $\langle\psi_k|\psi_0\rangle = \langle\psi_k|\phi_j\rangle = 0$ avec $k \neq j$:

$$(\mathcal{E}_k - \mathcal{E}_j)\langle\phi_k|\psi_1\rangle + \langle\phi_k|\hat{W}|\phi_j\rangle = 0 \tag{2.50}$$

$$\boxed{\alpha_k^{(1)} = \frac{-\langle\phi_k|\hat{W}|\phi_j\rangle}{\mathcal{E}_k - \mathcal{E}_j} = \frac{-\langle\phi_k|\hat{W}|\psi_0\rangle}{\mathcal{E}_k - \mathcal{E}_j}} \tag{2.51}$$

– Puis, pour déterminer n'importe quel $\alpha_k^{(q)}$ on reprend 2.40 auquel on applique $\langle\phi_k|$:

$$\langle\phi_k|\hat{H}^0 - E_0|\psi_q\rangle + \langle\phi_k|\hat{W} - E_1|\psi_{q-1}\rangle - \sum_{i=2}^{q} E_i\langle\phi_i|\psi_{q-i}\rangle = 0 \tag{2.52}$$

d'où, avec les mêmes arguments :

$$\boxed{\alpha_k^{(q)} = \langle\phi_k|\psi_q\rangle = \frac{1}{\mathcal{E}_k - \mathcal{E}_j}\left[-\langle\phi_k|\hat{W} - E_1|\psi_{q-1}\rangle + \sum_{i=2}^{q} E_i\langle\phi_i|\psi_{q-i}\rangle\right]} \tag{2.53}$$

Détermination des termes d'énergie propre

Pour trouver les énergies E_k, on projette chaque équation du hamiltonien d'ordre k sur le vecteur $|\phi_j\rangle$.

- En appliquant $\langle\phi_j|$ à 2.37 (ordre 1) on obtient

$$\langle\phi_j|\hat{H}^0 - E_0|\psi_1\rangle + \langle\phi_j|\hat{W} - E_1|\psi_0\rangle = 0 \tag{2.54}$$

donc puisque $\langle\psi_1|\phi_j\rangle = 0$ et que les $|\phi_i\rangle$ sont des vecteurs propres pour $\hat{H}^0$, le premier terme du membre de gauche est nul donc cela donne

$$\boxed{E_1 = \langle\phi_j|\hat{W}|\phi_j\rangle} \tag{2.55}$$

- En appliquant $\langle\phi_j|$ à 2.40 (ordre $q \geq 2$) on obtient, toujours avec $E_0 = \mathcal{E}_j$:[6]

$$\langle\phi_j|\hat{H}^0 - \mathcal{E}_j|\psi_q\rangle + \langle\phi_j|\hat{W} - E_1|\psi_{q-1}\rangle - \sum_{i=2}^{q} E_i\langle\phi_j|\psi_{q-i}\rangle = 0 \tag{2.56}$$

puis comme $\hat{H}^0$ est auto-adjoint, $\langle\phi_j|\hat{H}^0 - \mathcal{E}_j|\psi_q\rangle = \langle\psi_q|\hat{H}^0 - \mathcal{E}_j|\phi_j\rangle = 0$, on peut extraire E_q :

$$\boxed{E_q = \langle\phi_j|\hat{W} - E_1|\psi_{q-1}\rangle - \sum_{i=2}^{q-1} E_i\langle\phi_j|\psi_{q-i}\rangle} \tag{2.57}$$

[6] Avec la même convention de somme nulle si son ensemble d'indices est vide.

Synthèse du calcul

On peut résumer les calculs à effectuer et leurs dépendances à chaque étape dans le diagramme 2.6. Ces 6 formules encadrées (ordre 1 + ordre $q \geq 2$) $\times$ (1 terme d'énergie + 2 termes de composantes vecteurs) formules sont très facilement automatisables en informatique et permettent ainsi de calculer à tout ordre la méthode des perturbations pour l'état propre. Mais dans le code, il faut faire attention à calculer le terme d'énergie avant les composantes, qui l'utilisent majoritairement.

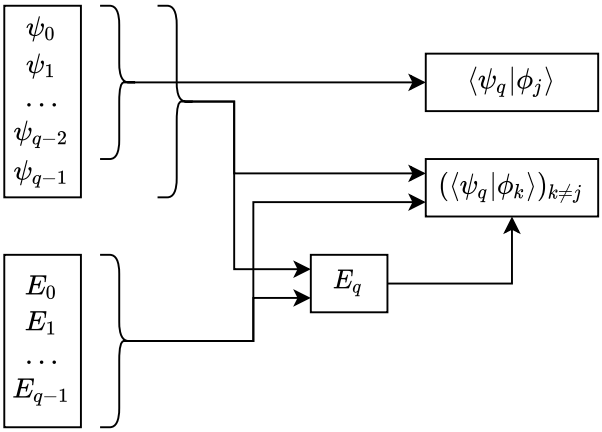


Figure 2.6 : Relations de causalité dans le calcul des termes des développements respectifs de l'énergie et de l'état propre après perturbation à tout ordre $q \geq 2$. En pratique il faudra donc calculer l'énergie à l'ordre q avant les coefficients du mode d'ordre q dans la base de décomposition.

Complexité asymptotique

En notant $N = n \times n$ la dimension de l'espace de Hilbert dans lequel nos états sont calculés (le nombre de points sur la grille), à l'étape $q \geq 2$, sachant les élements des étapes $\leq q-1$, le coût des étapes déterminantes en multiplications ou divisions pour obtenir le vecteur ψ_j et le terme d'énergie E_q est

$$c_q = \underbrace{1 + (q-1) \times \mathrm{N}}_{\alpha_j^{(q)}\,(2.48)} + \underbrace{N \times N^2 + (q-1) \times (1+N)}_{\alpha_{k\neq j}^{(q)}\,(2.53)} + \underbrace{N \times N^2 + (q-3) \times N}_{E_q\,(2.57)}$$
$$= 2N^3 + 3qN - 5N + 2q - 3 \quad (2.58)$$

et le coût total de calcul de l'étape 2 à l'étape $Q_{\max}$ est donc

$$\sum_{q=2}^{Q_{\max}} c_q = -2N^3 + \frac{1}{2}(3N+2)Q_{\max}^2 + \frac{1}{2}(4N^3 - 7N - 4)Q_{\max} + 2N + 1 = \boxed{\mathcal{O}(NQ_{\max}^2) + \mathcal{O}(N^3 Q_{\max})} \quad (2.59)$$

Remarque 2.6.2. *Le nombre N est le paramètre déterminant, car l'ordre maximal ne frôlera jamais la centaine (notamment parce que s'il y a convergence, elle est géométrique, et on fera en sorte qu'un nommbre d'étapes $\ll 100$ soit suffisant), tandis que $N \gg 10^3$ pour une simple grille 100×100.*

Si nous avons une grille 2D de taille $N = n \times n$, la majorité du coût en calcul pour de petits ordres est due principalement aux produits vectoriels, par conséquent en $\mathcal{O}(n^6 Q_{max})$ et bien-sûr,

ce coût s'ajoute à celui, non compté ici, qu'il faut pour diagonaliser le système, faisable en $\mathcal{O}(N^3)$ (LU ou Cholesky par exemple).

Implémentation : mise en œuvre et limites

Voici, pour l'illustration, un code spécimen représentatif des principales étapes des calculs, à l'ordre $q \geq 2$. On exploite des matrices qui servent pour l'occasion de liste de vecteurs verticaux (comme en notation mathématique) ce qui optimise l'accès mémoire de manière contiguë. Sans l'utilisation de tenseurs, on est forcé de recourir à des boucles, ce qui n'est pas très grave, car elles seront toujours de petite taille et contrairement à Python, leur exécution est, comme en C ou C++, rapide, mais pas facilement parallélisable sur GPU. Les calculs sont faits vectoriellement tant qu'ils peuvent l'être. Une solution serait de les représenter à l'aide de tenseurs, qui seraient creux.

L'autre gros défaut de cette méthode, est de nécessiter l'ensemble des vecteurs propres fortement orthogonaux[7] de $\hat{H}_0$ pour projeter ensuite sur ces vecteurs les termes de la perturbation. Cela peut être un problème pour de gros systèmes avec beaucoup de points de discrétisation, en terme de temps de calcul et de mémoire. De plus, en raison des problèmes exposés dans le paragraphe 1.1.4, obtenir les états des plus grandes énergies précisément en différences finies n'est pas acquis, et introduit des sources d'erreur qu'on ne maîtrise pas forcément bien. La méthode du paragraphe suivant, basée sur les gradients conjugués, élimine ce besoin.

[7] Au sens où les calculs utilisent cette hypothèse, donc une mauvaise orthogonalité des modes propres entre eux fausse le résultat perturbé.

Code specimen avec détail de la boucle à tout ordre $q \geq 2$

```
# ENTRÉES: \ W̃: perturbation
#          | lE: liste des énergies non perturbées,
#          \ lΦ: liste des modes non perturbés,
#          | N: nombre de points de discrétisation,
#          \ ind_approx: l'indice du mode à approcher en le perturbant,
#          | λ: coefficient devant Ŵ

lΨ = zeros(N^2, qmax); # liste des termes-vecteurs Ψ_q pour l'ordre q ∈ {1,2,...,q_max}
lE = zeros(qmax);      # liste des termes-énergie E_q pour l'ordre q ∈ {1,2,...,q_max}
dim = length(lℰ);      # nombre de modes sur lesquels on projette


ΦHBO = reshape(lΦ[ind_approx+1],N*N); # état propre HBO φ_j
EHBO = l [ind_approx+1]; # énergie propre HBO E_j
Ψc = zeros(N*N);         # vecteur corrigé  avec la théorie des perturbations
Ec = 0;                  # énergie corrigée avec la théorie des perturbations
Ψc += ΦHBO;
Ec += EHBO;

# ....
W̃mE1 = W̃ .- Diagonal(lE[1]*ones(N*N));
# ....

for q=2:qmax
    lE[q] = [formule 2.57] ΦHBO'*W̃mE1*lΨ[:,q-1] - sum([lE[r]* ΦHBO' * lΨ[:,q-r] for r
    ↪  in 2:q-1])
    Ec += (λ^q)*lE[q]; # ajout contribution ordre q en λ dans le DSE de E


    for k=1:dim
        αqj   = [formule 2.48] -1/2*sum([lΨ[:,r]'*lΨ[:,q-r] for r in 1:q-1]);
        αqknj = [formule 2.53] -1/(l [k] - EHBO)*( - lΦ[:,k]'*W̃mE1*lΨ[:,q-1] +
        ↪  sum([lE[r] * lΦ[:,r]'*lΨ[:,q-r] for r in 2:q-1]) + lE[q]*lΦ[:,q]' *
        ↪  ΦHBO);


        αqk = (k == ind_approx+1) ? αqj : αqknj; # disjonction selon k = j ou k ≠ j
        lΨ[:,q] += αqk * lΦ[:,k]; # calcul de Ψ_q et màj dans la liste des vecteurs
    end
    Ψc += λ^q*lΨ[:,q]; # ajout contribution ordre q en λ dans le DSE de |Ψ⟩
end

# ...

# SORTIES: \ ΦHBO, EHBO: solution non perturbée,
#          | lΨ, lE: liste des modes + énergies dans le DSE de la perturbation,
#          \ A = lΦ^T × lΨ:  décomposition des termes de lΨ sur la base des lΦ,
#          | Ψc, Ec: solution perturbée estimée (mode + énergie)
```

Démonstration sur l'exemple de l'oscillateur harmonique 2D

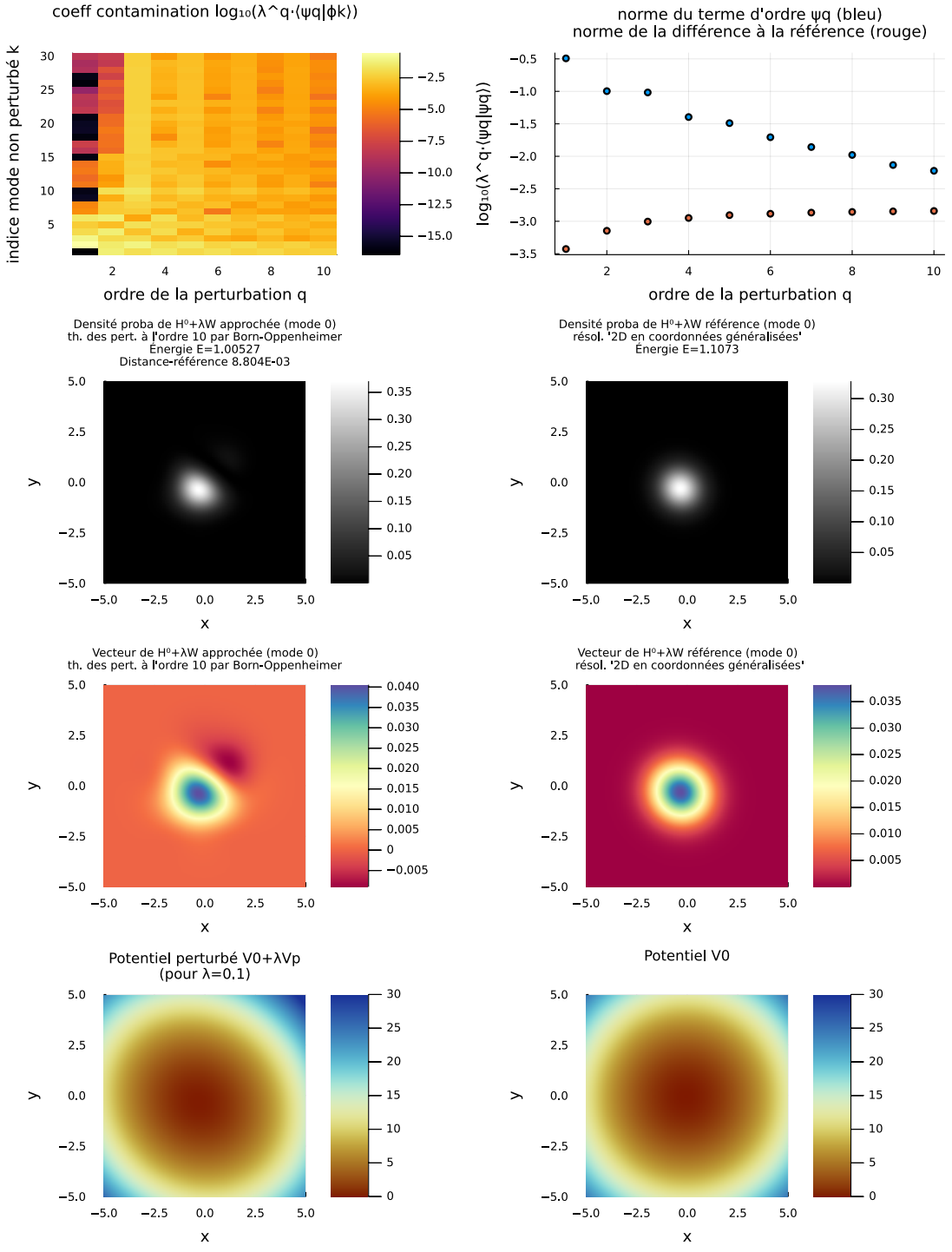


Figure 2.7 : Exemple du calcul du mode de l'état fondamental sur l'exemple : oscillateur harmonique 2D, dont la perturbation a été projetée uniquement sur les 30 premiers modes non perturbés par commodité. On constate dans ce cas une convergence mais la meilleure approximation reste l'ordre 1 (voir graphe de la norme à la différence). En clair, ces résultats ne sont pas un franc succès.

2.6.3 Détermination des termes par projection et descente de gradient

Gradients conjugués

Explicitons d'abord un résultat avec la méthode des gradients conjugués que l'on présente succintement. Étant donnée une matrice $A \in \mathscr{S}_m^{++}$ (symétrique positive définie) et un vecteur $\mathbf{b}$ dans $\mathbb{R}^m$, la méthode des gradients conjugués construit une suite de vecteurs $(\mathbf{x}^k)_k$ telle que

$$\mathbf{x}^k \xrightarrow[k\to+\infty]{} \underset{\mathbf{x}\in\mathbb{R}^m}{\operatorname{argmin}} f(\mathbf{x}) := \mathbf{x}^\star$$

où $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\intercal A\mathbf{x} - \mathbf{b}^\intercal\mathbf{x}$, ce qui est donc équivalent à résoudre $A\mathbf{x} = \mathbf{b}$.

L'algorithme est présent dans de nombreux ouvrages et références sur internet et se présente de la manière suivante en pseudo-code, que j'ai reproduit ici[8] :

Inputs $\mathbf{x}_0, \mathbf{b}, \mathbf{A}$
Processing $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
while $\|\mathbf{r}_k\| \geq \varepsilon$ do
$\alpha_k := \dfrac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\|r_{k+1}\| \leq \varepsilon$
return $\mathbf{x}_{k+1}$
end if
$\beta_k := \dfrac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end while
Outputs $\mathbf{x}_{k+1}$

Dans ce code, $\mathbf{r}$ représente $-\nabla(f)$.

En notant l'erreur $\mathbf{e}^k = \mathbf{x}^k - \mathbf{x}^\star$, cet algorithme a la propriété de vérifier pour tout k [Pol87] :

$$\|\mathbf{e}^k\| \leq 2\kappa(A)\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|\mathbf{e}^0\| \tag{2.60}$$

où $\kappa(A)$ désigne le conditionnement de A.

Cette méthode présente l'avantage d'être plus rapide asymptotiquement que d'autres algorithmes de résolution de systèmes linéaires comme celui de Jacobi ou Gauss-Seidel, même si elle reste coûteuse en opérations, car il faut faire de nombreux produits matriciels.

Proposition 2.6.3. *Si le vecteur initial* $\mathbf{x}_0$ *et* $\mathbf{b}$ *appartiennent à un sous-espace* V *stable par* $\mathbf{A}$*, alors la limite s'y trouve également.*

[8] Source : https://en.wikipedia.org/wiki/Conjugate_gradient_method

Démonstration. Montrons que pour tout k, $\mathbf{r}_k$, $\mathbf{x}_k$ et $\mathbf{p}_k$ sont dans V avec les hypothèses de la proposition. On a $\mathbf{p}_0, \mathbf{r}_0 \in V$ par combinaison linéaire. Par récurrence, on le suppose vrai à l'indice k. Alors en suivant les lignes de l'algorithme, $\mathbf{x}_{k+1} \in V$ par CL et $\mathbf{A}\mathbf{p}_k \in V$ par stabilité, donc par CL et par hypothèse de récurrence, $\mathbf{r}_{k+1} \in V$. Puis $\mathbf{p}_{k+1} \in V$ par CL avec $\mathbf{r}_{k+1}$ et $\mathbf{p}_k$.

Comme un sous-espace vectoriel de dimension finie est fermé, la limite appartient à V, ce qui conclut. ■

Proposition 2.6.4. *Dans la base des vecteurs propres $(\phi_1, \cdots, \phi_N)$, de $\hat{H}^0$, la matrice $(\hat{H}^0 - E_0)$ s'écrit* $\mathrm{diag}(0, E_1 - E_0, \cdots, E_N - E_0)$ *donc finalement nous avons en notant $\Pi^\perp$ le projecteur orthogonal sur l'orthogonal de ϕ_1 :*

$$M = \Pi^\perp(\hat{H}^0 - E_0)\Pi^\perp = \left[\begin{array}{c|c} 0 & 0^{N-1\intercal} \\ \hline 0^{N-1} & R \end{array}\right] \tag{2.61}$$

avec ainsi $Sp(\hat{H}^0 - E_0) = \{0\} \cup \mathrm{Sp}(R)$ et $\mathrm{Sp}(R) = \{E_1 - E_0, \cdots, E_N - E_0\} \subset \mathbb{R}_+^\star$.

La méthode des gradients conjugués avec M et n'importe quel $\mathbf{b} \in \mathrm{Vect}(\phi_j)^\intercal$ et $\mathbf{x}^0 = \mathbf{0}$, aboutit donc à une solution dans $\mathrm{Vect}(\phi_j)^\intercal$ *si le vecteur initial est* $\mathbf{0}$.

Comme

$$\mathbb{R}^N = \mathrm{Vect}(\phi_j) \oplus \underbrace{\mathrm{Vect}(\phi_j)^\perp}_{=\mathrm{Span}(R) \simeq \mathbb{R}^{N-1}} \tag{2.62}$$

on écrit de manière unique pour tout $\mathbf{v} \in \mathbb{R}^N, \mathbf{v} = \mathbf{v}_\parallel^k + \mathbf{v}_\perp^k$, et comme d'une part l'orthogonal est stable par R, et d'autre part le gradient de M est à valeurs dans cet orthogonal, on obtient que toutes les équations des gradients conjugués sont valables en remplaçant les vecteurs par leur composante indicée $\perp$.

Par conséquent, finalement, avec ces vecteurs de départ, on peut considérer uniquement l'algorithme dans l'hyperplan $\mathrm{Vect}(\phi_j)^\intercal$ *et ainsi obtenir la convergence en*

$$\mathcal{O}\left(\frac{\sqrt{\kappa(R)} - 1}{\sqrt{\kappa(R)} + 1}\right)^k = \mathcal{O}\left(\frac{\sqrt{E_N - E_1} - \sqrt{E_1}}{\sqrt{E_N - E_1} + \sqrt{E_1}}\right)^k \tag{2.63}$$

et la limite est bien dans l'orthogonal.

Pour éviter l'écueil de devoir connaître les vecteurs propres du hamiltonien non perturbé (ce qui nécessite de diagonaliser le système, on utilise une autre méthode, basée également sur des projections, mais **au lieu de calculer exactement chaque projection sur un ensemble limité de vecteurs propres non perturbés, on calcule approximativement (avec une métode itérative) les projections sur deux sous-espaces supplémentaires dont la somme directe recouvre l'espace**, grâce aux gradients conjugués.

Reprenons les formules du DSE de la perturbation à tout ordre :

$$\begin{aligned}
(\hat{H}^0 - E_0)|\psi_0\rangle &= 0 \,(\text{ordre } 0) \\
(\hat{H}^0 - E_0)|\psi_1\rangle + (\hat{W} - E_1)|\psi_0\rangle &= 0 \,(\text{ordre } 1) \\
(\hat{H}^0 - E_0)|\psi_2\rangle + (\hat{W} - E_1)|\psi_1\rangle - E_2|\psi_0\rangle &= 0 \,(\text{ordre } 2) \\
&\vdots \\
(\hat{H}^0 - E_0)|\psi_q\rangle + (\hat{W} - E_1)|\psi_{q-1}\rangle - E_2|\psi_{q-2}\rangle + \cdots - E_q|\psi_0\rangle &= 0 \,(\text{ordre } q)
\end{aligned}$$

Comme pour la méthode *sum over states*, on va distinguer des cas, mais seulement deux : projection sur $\mathrm{Vect}(\phi_j)$ (mode que l'on cherche à approcher après perturbation) ; et $\mathrm{Vect}(\phi_j)^\perp$.

Toutes les équations ci-dessus sont en particulier valables sur ces deux sous-espaces vectoriels par linéarité. Notons pour la suite $\Pi^{\|}$ le projecteur orthogonal sur $\mathrm{Vect}(\phi_j)$ et $\Pi^{\perp}$ celui sur $\mathrm{Vect}(\phi_j)^{\perp}$.

À l'ordre 1

On a vu précédemment que $\psi_1 \in \phi_0^{\perp}$, ce qui permet d'écrire $|\psi_1\rangle = \Pi^{\perp}|\psi_1\rangle$ et en projetant l'équation d'ordre 1 2.37 sur l'orthogonal à gauche il vient :

$$\Pi^{\perp}(\hat{H}^0 - E_0)\Pi^{\perp}|\psi_1\rangle = -\Pi^{\perp}\hat{W}|\psi_0\rangle \tag{2.64}$$

Puisque E_0 est une énergie de mode non dégénéré, l'application linéaire $\Pi^{\perp}(\hat{H}^0 - E_0)\Pi^{\perp}$ restreinte au sous-espace orthogonal est inversible. On a donc un système linéaire singulier dont seule une direction est nulle. En pratique, on utilise l'algorithme des gradients conjugués avec vecteur initial nul (qui appartient aux deux sous-espaces en somme directe).

On a déjà montré que l'énergie E_1 s'obtenait par 2.55 :

$$E_1 = \langle\psi_0|\hat{W}|\psi_0\rangle \tag{2.65}$$

À l'ordre $q \geq 2$

On part de l'ordre q en supposant les $|\psi_i\rangle$ avec $i < q$ connus, et comme précédemment, on exploite le fait que $(\hat{H}^0 - E_0)$ laisse stables les sous-espaces supplémentaires ce qui entraîne la commutation $(\hat{H}^0 - E_0)\Pi^{\perp} = \Pi^{\perp}(\hat{H}^0 - E_0)$ et $(\hat{H}^0 - E_0)\Pi^{\|} = \Pi^{\|}(\hat{H}^0 - E_0)$.

D'abord, nous écrivons

$$\underbrace{\langle\psi_0|\hat{H}^0 - E_0|\psi_q\rangle}_{=0} = -\langle\psi_0|\hat{W} - E_1|\psi_{q-1}\rangle + \langle\psi_0|\sum_{i=0}^{q-2} E_{q-i}|\psi_i\rangle \tag{2.66}$$

Cela permet de trouver l'énergie d'ordre q :

$$E_q = \langle\psi_0|\hat{W} - E_1|\psi_{q-1}\rangle - \sum_{i=1}^{q-2} E_{q-i}\langle\psi_0|\psi_i\rangle \tag{2.67}$$

Ensuite, on prend l'égalité à l'ordre q des termes du DSE que l'on projette sur l'orthogonal, ce qui fait apparaître le système linéaire suivant pour l'état $|\psi_q\rangle$.

$$\Pi^{\perp}(\hat{H}^0 - E_0)\Pi^{\perp}|\psi_q\rangle = -\Pi^{\perp}(\hat{W} - E_1)|\psi_{q-1}\rangle + \Pi^{\perp}\sum_{i=0}^{q-2} E_{q-i}|\psi_i\rangle \tag{2.68}$$

Tandis que la composante suivant son supplémentaire est donnée, comme pour la méthode précédente, par le fait que $|\Psi\rangle$ est normé à tout ordre : on a alors déjà écrit en 2.48 l'égalité qui nous manque :

$$\Pi^{\|}|\psi_q\rangle = \langle\psi_0|\psi_q\rangle = -\frac{1}{2}\sum_{i=1}^{q-1}\langle\psi_i|\psi_{q-i}\rangle \tag{2.69}$$

On a enfin bien-sûr :

$$\psi_q = \Pi^{\perp}\psi_q + \Pi^{\|}\psi_q \tag{2.70}$$

Là encore on se retrouve avec des équations dont la résolution est très facilement automatisables et dont le code sera d'ailleurs très succinct puisque des bibliothèques implémentent les gradients conjugués.

Complexité asymptotique et implémentation

À l'origine, une première version de l'algorithme que j'avais programmé mettait en œuvre cette méthode avec l'usage du GPU de manière naïve (i.e., reprenant les lignes de calcul point par point). C'était l'occasion d'apprendre à maîtriser un nouvel outil. L'algorithme et les hypothèses sont donnés en annexe (algorithme 2). Le paradigme principal était alors d'exploiter le GPU pour des calculs lourds (tels que les projections)

Ce choix est mauvais à plusieurs égards à ce stade de la conception :

- D'une part il requiert des bibliothèques « compliquées » pour piloter les opérations sur GPU et le matériel en lui-même, ce qui peut facilement dégénérer en transferts mémoire longs comme on le verra dans une section ultérieure ;
- D'autre part les GPU n'ont pas nécessairement des cœurs rapides à la précision flottante sur 64 bits.
- La plupart des produits coûteux entre matrices et vecteurs peuvent être facilement évités en n'assemblant pas les matrices de projecteurs.
- Certains cas ne sont pas nécessairement implémentés en Julia sur GPU : à l'été 2022, un produit matrice-matrice `sparse*dense` est supporté par Julia CUDA, mais pas `dense*sparse`.
- Enfin, mes tests ont montré que pour les routines disponibles employables pour résoudre mon problème, la sensibilité aux conditions initiales est grande, on doit donc savoir ce qu'on cherche à l'avance et le temps d'exécution est simplement plus long que sur CPU donc ça n'a pas beaucoup d'intérêt.

J'ai alors plus simplement fait le choix d'en rester aux seuls CPU, sans même aller jusqu'à choisir le GPU pour certaines étapes seulement (cf annexe tout mon travail effectué en section D.1).

Quant aux projections, il s'agit d'une instance particulièrement simple du problème de **parenthésage optimal des multiplications matricielles enchaînées** (dans notre cas, mettant en jeu uniquement des matrices et vecteurs). Plus clairement, pour effectuer le calcul $\Pi\mathbf{x}$ avec Π un projecteur selon un vecteur *normé* connu $\mathbf{y}$, on ne calcule pas naïvement $(\mathbf{y}\mathbf{y}^\intercal)\mathbf{x}$ (coût $\mathcal{O}(N^2)$), mais plutôt $\mathbf{y}(\mathbf{y}^\intercal\mathbf{x})$ (coût $\mathcal{O}(N)$). Ce calcul ne peut cependant pas être utilisé comme variable classique dans les expressions des calculs courants. Pour utiliser cette décomposition dans des routines telles que les gradients conjugués, on la formalise donc en construisant un objet `LinearMap` avec par exemple la commande `Π_par = LinearMap(x -> dot(Ψ,x)*Ψ, N²);`, que l'on peut ensuite appeler comme une fonction via une méthode associée dédiée, ou dans dans les fonctions routines du paquet `IterativeSolvers` en Julia, telles que `cg`. C'est à peu près la seule « difficulté technique » posée par ce choix, si on ne veut pas réécrire le code des gradients conjugués entièrement à la main.

Je me propose de montrer dans ce paragraphe que la complexité asymptotique de l'algorithme précédent (utilisant les gradients conjugués) est meilleure que celui basé sur la *sum over states*, sous l'hypothèse que la valeur propre maximale de la matrice du hamiltonien en dimension N tend vers $+\infty$ lorsque $N \to +\infty$, ce qui est le cas dans le théorème XIII.64 tiré de [RS80b].

Rappelons en premier lieu que R est la matrice qui apparaît dans 2.61, et est de taille $(N-1)\times(N-1)$. On peut donc indicer R par N. De plus comme précédemment, $\kappa(R_N)$ dénote son conditionnement en norme 2.

Proposition 2.6.5. *Pour tout $\eta > 0$, il existe N suffisamment grand tel que pour la grille discrétisée en N^2 points, la condition*

$$k \geq -\frac{\sqrt{\kappa(R_N)}}{2-\eta}\left[\ln\varepsilon - \ln(2\kappa(R_N)\|\mathbf{e}^0\|)\right] \tag{2.71}$$

est suffisante pour que la précision machine en précision double (FP64) $\varepsilon \approx 10^{-16}$ soit atteinte, i.e. $\|\mathbf{e}^k\| \leq \epsilon$ lors de l'application des gradients conjugués.

En d'autres termes, au voisinage de $N \to +\infty$ (i.e. pour N suffisamment grand), le nombre k d'itérations nécessaires des gradients conjugués vérifie

$$\boxed{k \underset{N\to+\infty}{=} \mathcal{O}\left(\sqrt{\kappa(R_N)}\ln\kappa(R_N)\right)} \tag{2.72}$$

ce qui est un résultat intéressant.

Démonstration. La condition nécessaire et suffisante sur le nombre d'itérations k pour atteindre la précision machine est (d'après 2.60) la suivante :

$$\underbrace{2\kappa(R)\left(\frac{\sqrt{\kappa(R)}-1}{\sqrt{\kappa(R)}+1}\right)^k\|\mathbf{e}^0\| \leq \varepsilon}_{\text{déf. précision machine atteinte}} \iff k\ln\left(\frac{\sqrt{\kappa(R)}-1}{\sqrt{\kappa(R)}+1}\right) \leq \ln\varepsilon - \ln\left(2\kappa(R)\|\mathbf{e}^0\|\right) \tag{2.73}$$

Nous commençons par écrire le développement asymptotique suivant :[9]

$$\frac{\sqrt{x}-1}{\sqrt{x}+1} \underset{x\to+\infty}{=} 1 - \frac{2}{\sqrt{x}} + o\left(\frac{1}{\sqrt{x}}\right)$$

Soit $1 > \eta' > 0$, nous avons donc au voisinage de $+\infty$ l'inégalité :

$$\frac{\sqrt{x}-1}{\sqrt{x}+1} \underset{x\to+\infty}{\leq} 1 - \frac{2-\eta'}{\sqrt{x}}$$

D'autre part nous avons le développement limité suivant avec une inégalité associée au voisinage de 0 :

$$\ln(1-(2-\eta')y) \underset{y\to0}{=} -(2-\eta')y + o(y) \underset{y\to0}{\leq} -(2-2\eta')y$$

Notons $\eta \longleftrightarrow 2\eta'$. En substituant y par $\frac{1}{\sqrt{x}}$, on obtient par croissance du log et par transitivité des inégalités :

$$\ln\left(\frac{\sqrt{x}-1}{\sqrt{x}+1}\right) \underset{x\to+\infty}{\leq} -\frac{2-\eta}{\sqrt{x}}$$

Comme $\kappa(R) = \frac{E_N-E_0}{E_1-E_0} \xrightarrow[N\to+\infty]{} +\infty$ (cf. équation 2.61), lorsque $\kappa(R)$ est grand, on obtient en substituant x par $\kappa(R)$ dans la précédente inégalité asymptotique :

$$\boxed{\ln\left(\frac{\sqrt{\kappa(R)}-1}{\sqrt{\kappa(R)}+1}\right) \underset{N\to+\infty}{\leq} -\frac{2-\eta}{\sqrt{\kappa(R)}}}$$

Ainsi, en supposant la matrice R suffisamment grande (i.e. en supposant N suffisamment grand, pour une la grille suffisamment fine), les énergies de plus en plus hautes pourront être

[9] On peut s'aider en général de PARI/gp

approchées et le conditionnement de R noté $\kappa(R)$ sera suffisamment grand pour écrire son développement asymptotique au voisinage de $+\infty$.

Par conséquent, la condition 2.71 ainsi réécrite :

$$\boxed{-k\frac{2-\eta}{\sqrt{\kappa(R)}} \leq \left[\ln\varepsilon - \ln(2\kappa(R)\|\mathbf{e}^0\|)\right]}$$

suffit à rendre vraie l'inégalité 2.73, définition de « précision machine atteinte » (ou précision souhaitée) :

$$k\ln\left(\frac{\sqrt{\kappa(R)}-1}{\sqrt{\kappa(R)}+1}\right) \leq -k\frac{(2-\eta)}{\sqrt{\kappa(R)}} \leq \ln\varepsilon - \ln\left(2\kappa(R)\|\mathbf{e}^0\|\right)$$

par simple transitivité des deux inégalités encadrées précédentes et par positivité de k.

Ceci conclut la démonstration. ■

Complexité de la méthode complète

Dénombrons alors les opérations (produits, divisions) coûteuses pour parvenir à la solution jusqu'à l'ordre $Q_{\max}$:

- Les opérations dans le terme de droite (RHS)[10] de 2.64 comptabilisent $3N$ produits exactement, puisque $\hat{W}$ est diagonale. Chaque application de l'opérateur du terme de gauche (LHS)[11] de 2.64, comptabilise au plus $10N$ produits. En effet, c'est sans assembler $\hat{H}^0 - E_0$ (pire cas), et en prenant en compte les projections non assemblées ($2N$ produits chacune) et le fait que $\hat{H}^0$ est creux (laplacien à 5 points). On a donc bien pour $\Pi^\perp(\hat{H}^0 - E_0)\Pi^\perp|\psi_1\rangle$, le total $2 + (5+1) + 2 = 10$.
- Le calcul de 2.65 comprend exactement $2N$ produits.
- Le calcul de 2.67 comprend exactement $3N + (q-2)(N+1) = (q+1)N + q - 2$ produits.
- Le terme RHS de 2.68 s'effectue en $4N + (2(q-1)+1)N = (2q+3)N$ produits. Chaque application de l'opérateur du LHS s'effectue comme vu précédemment en $10N$ produits.
- Enfin, le calcul du coefficient cherché dans 2.69 met en jeu qN produits.

Ainsi donc le coût total asymptotique de la méthode jusqu'à l'ordre $Q_{\max}$ pour obtenir la limite à la précision machine est donné par

$$\underbrace{10N + \mathcal{O}\left(N\sqrt{\kappa(R)}\ln\kappa(R)\right)}_{q=1} + \sum_{q=2}^{Q_{\max}}\left[(q+1)N + q - 2 + \mathcal{O}\left(N\sqrt{\kappa(R)}\ln\kappa(R)\right) + qN\right] \quad (2.74)$$

```
N, q, c, Q = var('N, q, c, Q')
10*N+c*N+sum((q+1)*N+q-2+c*N+q*N, q,2,Q).simplify()
```

[10] Right-Hand Side
[11] Left-Hand Side

Asymptotiquement, on aboutit à un nombre de produits qui croît comme :

$$\boxed{\mathcal{O}(NQ_{\max}^2) + \mathcal{O}\left(NQ_{\max}\sqrt{\kappa(R)}\ln\kappa(R)\right)} \tag{2.75}$$

Remarque 2.6.6. *Vis-à-vis de la grille, on a donc grandement amélioré en pratique la complexité de l'algorithme précédent (qui était en $\mathcal{O}(N^3 Q_{max})$).*

La clé est de décomposer le problème sur deux sous-espaces seulement (au lieu de N sous-espaces propres précédemment) en utilisant avantageusement des méthodes à convergence géométrique pour résoudre de manière les systèmes linéaires nécessaires.

En pratique cependant, comme le paramètre déterminant est N (en effet, vu que les constantes devant les termes sont asymptotiques comparables, l'ordre de perturbation Q_{max} restera en fait majoré et n'a aucune chance de frôler les mêmes ordres de grandeur que N), la complexité se réduit donc à

$$\mathcal{O}\left(N\sqrt{\kappa(R)}\ln\kappa(R)\right) \tag{2.76}$$

Pour une grille de taille $N = n \times n$ d'étendue fixée avec $\kappa(R) = \frac{E_N - E_0}{E_1 - E_0}$, on a donc $\mathcal{O}\left(n^2\sqrt{E_{max}(n)}\ln(E_{max}(n))\right)$. Il resterait à déterminer la croissance des énergies propres approchées en fonction de la taille de la grille pour obtenir une expression dépendant uniquement de n.

2.6.4 Application à l'ion H_2^+

Résumé de la procédure

Pour résumer, récapitulons les points importants de notre algorithme d'approche de l'état fondamental :

1. Choisir une famille de fonctions de potentiel et choisir les paramètres de sorte qu'à R fixé, le hamiltonien de l'électron ne possède qu'une ou deux valeurs propres négatives, et choisir les paramètres de masse (§2.3) ;
2. Évaluer l'argument minimisant l'énergie de surface R_0 (§2.4.1) et évaluer numériquement $\frac{\mathrm{d}^2 E}{\mathrm{d}\lambda^2}$ (avec un noyau de différences finies par exemple) ;
3. Calculer un développement limité du potentiel en espace suivant la direction autour de R_0 (§2.5) ;
4. Calculer suivant chaque axe les états fondamentaux mutuellement découplés par n'importe quelle méthode connue en analyse numérique (§1.1.5) ce qu'on suppose savoir faire car la taille des données reste selon chaque axe comparativement petite ;
5. Perturber le produit tensoriel (loi produit) de ces états à l'aide de l'algorithme des perturbations et des termes du DL précédent (§2.6.3) ;

On calcule donc numériquement le potentiel perturbateur tronqué 2.31 du potentiel de l'ion H_2^+ que l'on a calculé (éq. 2.20) avec les paramètres $\beta = 1.5, \eta = 0.5, V_0 = 1.5, \sigma = 1$, $m_e = 1, m_p \in [10^1, +\infty[$ comme déjà donnés plus haut.

Les figures qui suivent donnent une visualisation des résultats de la procédure entière décrite précédemment. **J'ai pris volontairement pour les graphes une masse très petite de proton afin de montrer l'approche HBO par rapport à la référence.**

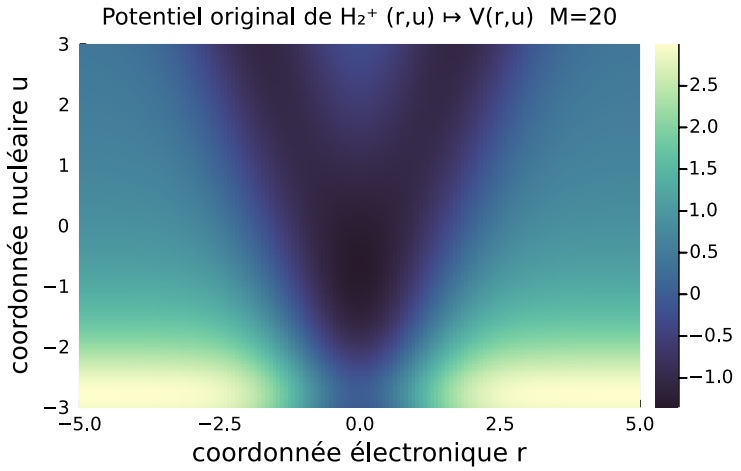


Figure 2.8 : Potentiel original de H_2^+

Figure 2.9 : Perturbation du potentiel HBO de H_2^+

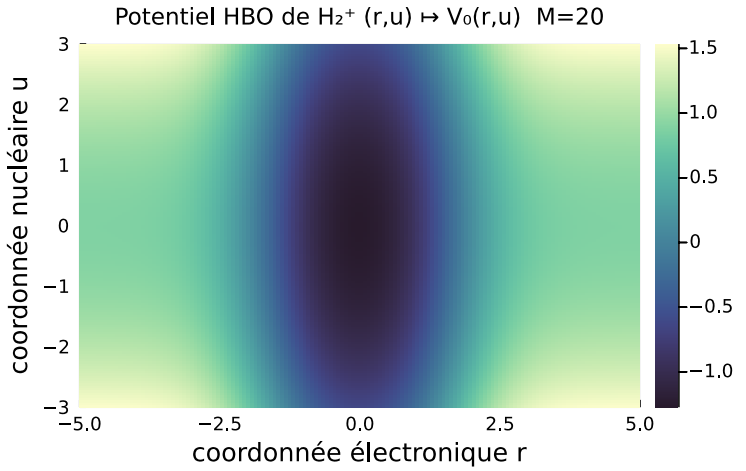


Figure 2.10 : Potentiel HBO de H_2^+, comme vu dans les équations, localement assimilable à un OHQ2D.

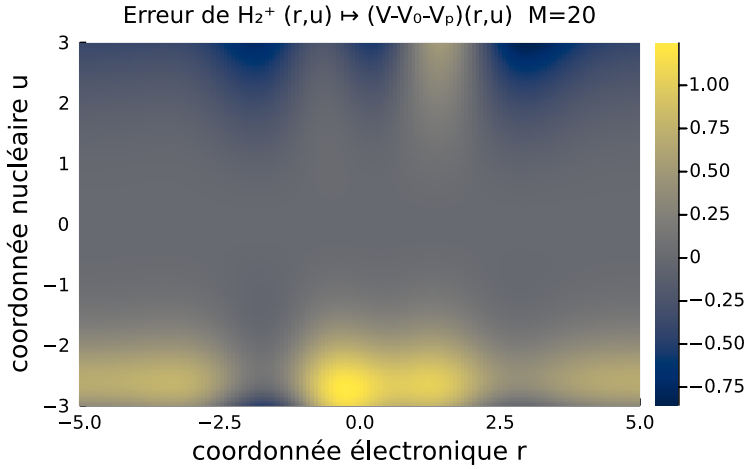
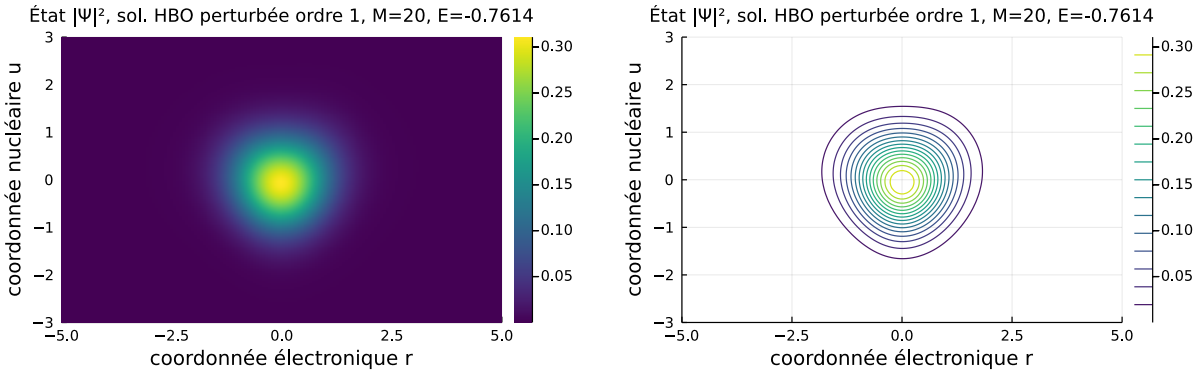
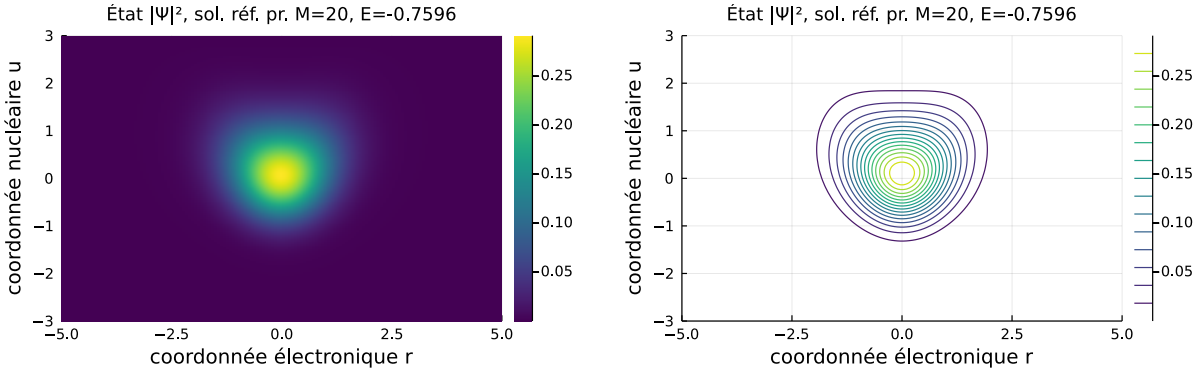


Figure 2.11 : Erreur commise sur le potentiel original avec la perturbation. Comme attendu, là où se concentre la probabilité de présence la plus forte, l'erreur est quasi-nulle. On s'attend donc à ce que l'état calculé avec perturbation, soit lui-même proche de la référence.



(a) densité **(b)** contour

Figure 2.12 : Solution HBO approchée par les gradients conjugués à l'ordre 1 en espace



(a) densité **(b)** contour

Figure 2.13 : Solution de référence, calculée en 2D (en trichant) par une méthode de Krylov

2.7 Métriques d'évaluation et critique des résultats

2.7.1 Métriques d'évaluation

Notations de ce paragraphe En reprenant les paramètres du problème, on a l'énergie fondamentale vraie $\mathcal{E}_0$ associée à l'état vrai ψ_0 ainsi que l'énergie fondamentale approchée par l'algorithme E_0 associée à l'état Ψ_0 obtenu par perturbation de Ψ^{HBO}.

Inégalité de Kato-Temple

On va appliquer l'inégalité de Kato-Temple 1.26 à l'état fondamental ψ_0 trouvé par les perturbations avec 2.32, en approchant le dénominateur par $K\varepsilon^2$. En effet, on postule que l'énergie de niveau 1 (la plus petite énergie propre strictement supérieure à l'énergie fondamentale est proche de l'énergie caractéristique de l'oscillateur harmonique quantique de $\mathfrak{h}_u$ dans 2.32 (phonon, ou mode de vibration des noyaux) en raison de l'hypothèse de masse électronique très petite $m \ll M$ dans le hamiltonien de départ 2.16, et que par conséquent la contribution en

énergie de celui-ci est négligeable devant celle des noyaux. Cette approximation étant de plus en plus valide à mesure que $\frac{m}{M} \to 0$, i.e. à mesure que $\varepsilon = (MK)^{-\frac{1}{4}} \to 0$ avec $m = 1$ fixé et $M \to +\infty$. **Autrement dit, dans ce raisonnement, on considère les deux hamiltoniens totalement découplés pour estimer le gap $\mathcal{E}_1 - \mathcal{E}_0$ (énergies véritables) du système total par $E_1 - E_0$ (énergies obtenues par perturbations), lui-même estimé par $K\varepsilon^2$. Cette hypothèse a été testée numériquement avant de l'injecter dans Kato-Temple, et il s'avère que l'erreur relative commise sur le gap est inférieure à 1‰, ce qui est tout-à-fait satisfaisant pour n'avoir pas à calculer un nouvel état.**

Ensuite, on compare les quantités $|\mathcal{E}_1 - E_1|$ et $\frac{\|\mathfrak{h}\Psi_0 - \langle\Psi_0|\mathfrak{h}|\Psi_0\rangle\Psi_0\|}{K\varepsilon^2}$ (notations reprises de 2.32, $\mathfrak{h} = -\frac{K\varepsilon^2}{2}\frac{\partial^2}{\partial u^2} - \frac{1}{2m}\frac{\partial^2}{\partial x^2} + V(x, R_0 + \varepsilon u)$) et on s'attend à avoir l'explication

$$|\mathcal{E}_0 - E_0| \leq \frac{\|\mathfrak{h}\Psi_0 - \langle\Psi_0|\mathfrak{h}|\Psi_0\rangle\Psi_0\|}{K\varepsilon^2} \tag{2.77}$$

Le but du jeu est de savoir si l'information contenue dans cette majoration est conséquente ou non.

Calcul des erreurs à la référence

En plus de cela, nous calculons

- une norme d'erreur à la solution de référence (qui est donc le vecteur Ψ_0 véritable) en norme 2 et en norme H1 (norme d'espace de Sololev) :

$$\| \underbrace{\psi_0}_{\text{véritable}} - \underbrace{\Psi_0}_{\text{perturbé}} \| \tag{2.78}$$

  obtenu par résolution 2D de la grille, ce qui serait normalement inaccessible, notre étude visant à ne pas y recourir). Nous calculons également le résidu simple de la solution perturbée

$$\|\mathfrak{h}\Psi_0 - E_0\Psi_0\| \tag{2.79}$$

- aussi bien-sûr l'erreur sur l'énergie (la valeur propre) :

$$| \underbrace{\mathcal{E}_0}_{\text{véritable}} - \underbrace{E_0}_{\text{perturbée}} | \tag{2.80}$$

Les résultats pour plusieurs masses ont été consignés dans la figure 2.14.

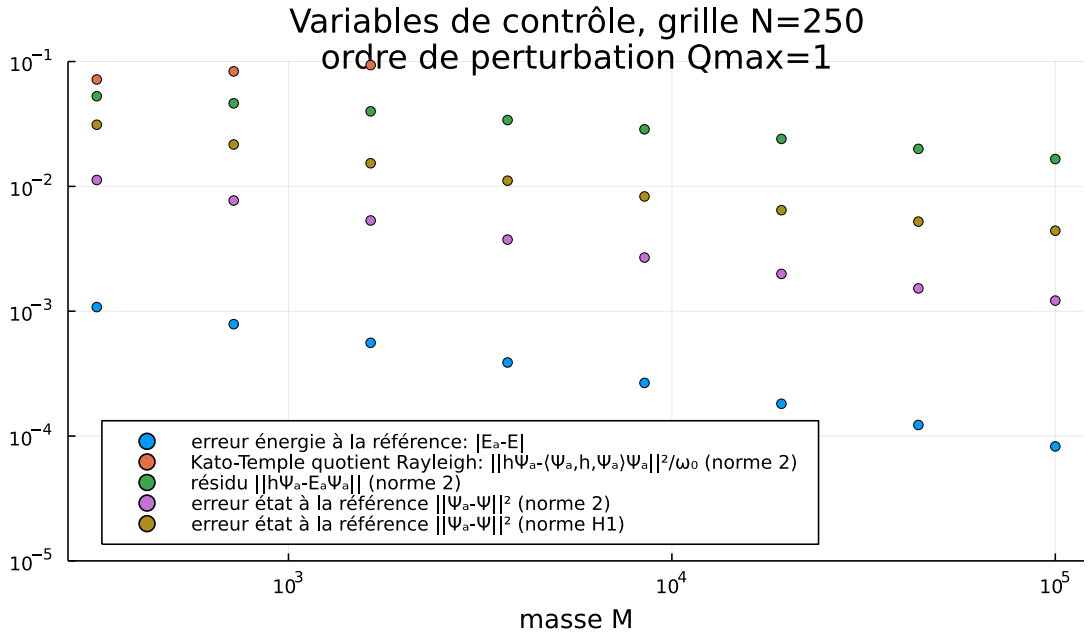


Figure 2.14 : Résultats des simulations pour les métriques décrites

2.7.2 Critique des résultats

Normalement, le travail effectué pendant ce stage d'application avec les codes permettrait de guider le travail de recherche pour trouver une explication analytique faite par Éric Cancès et Paul Cazeaux. On a donc approché le problème par l'observation et l'expérience d'abord, ce qui suit la démarche de la physique en tant que science expérimentale. Le travail présentement effectué est donc une bonne base pour de futurs travaux.

2.8 Conclusion

Dans tout le travail présenté, j'ai mis en pratique la méthode d'approximation physique qui m'a été prescrite pour approcher l'état et l'énergie propre d'un système moléculaire simplifié, à savoir H_2^+ unidimensionnel. En utilisant Julia, un langage adapté au calcul scientifique, j'ai pu programmer sans encombre l'algorithme à l'aide de quelques considération mathématiques annexes qui permettent d'accélérer et simplifier certaines étapes. À ma connaissance, comparer les résultats aux métriques était une idée originale de mon tuteur scientifique, ce qui devrait permettre de donner un éclairage sur la convergence de telles méthodes à l'avenir pour la communauté de scientifiques en chimie quantique et domaines proches.

Ouverture

Dans ce domaine (la chimie quantique et même plus généralement, l'étude des molécules, des protéines du vivant en biologie, etc), comme je l'ai déjà brièvement évoqué, on assiste également à une forte intervention du Machine Learning pour aider à résoudre plus efficacement des problèmes moléculaires (citons par exemple le programme AlphaFold de Google, qui calcule le repliement de protéines, très similaire à ce que l'on fait ici puisque c'est rechercher un état fondamental). Cela pourrait donc constituer encore une prolongation au sujet d'étude.

Annexes

Annexe A

Définitions et théorèmes principaux en théorie des opérateurs

Ces résultats sont issus de [CLBL21].

Définition A.0.1. *Une matrice complexe est dite hermitienne si elle est égale à sa conjuguée transposée.*

Définition A.0.2. *Un opérateur borné d'un espace de* Hilbert $\mathcal{H}$ *est une application linéaire* $A : \mathcal{H} \to \mathcal{H}$ *telle que*

$$\|A\| = \sup_{\psi \in \mathcal{H}\setminus\{0\}} \frac{\|A\psi\|}{\|\psi\|} < +\infty \tag{A.1}$$

Définition A.0.3. *(Adjoint d'un opérateur) On appelle adjoint* $A^\star$ *d'un opérateur* A *de domaine* $D(A)$ *sur* $\mathcal{H}$*, l'unique opérateur agissant sur le domaine*

$$D(A^\star) = \{\psi \in \mathcal{H}, \exists \chi_\psi \in \mathcal{H}, \forall \phi \in D(A), \langle A\phi|\psi\rangle = \langle \phi|\chi_\psi\rangle\} \tag{A.2}$$

tel que

$$A^\star : \psi \mapsto \chi_\psi \tag{A.3}$$

où $\langle A\phi|\psi\rangle = \langle \phi|\chi_\psi\rangle = \langle \phi|A^\star\psi\rangle$

Définition A.0.4. *(Opérateur auto-adjoint) Un opérateur est dit auto-adjoint si* $A^\star = A$*.*

Définition A.0.5. *(Résolvent et Spectre d'un opérateur linéaire) On appelle résolvent de l'opérateur linéaire* A *agissant sur* $\mathcal{H}$*, l'ensemble*

$$\rho(A) = \{z \in \mathbb{C}, (z - A) : D(A) \to \mathcal{H} \text{ inversible}\} \tag{A.4}$$

On appelle spectre de l'opérateur A *l'ensemble*

$$\sigma(A) = \mathbb{C}\backslash\rho(A) \tag{A.5}$$

Pour les matrices hermitiennes de $\mathbb{C}$, le spectre d'un opérateur auto-adjoint est réel.

Définition A.0.6. *(Valeurs spectrales et spectre ponctuel) On appelle spectre ponctuel d'un opérateur* A *sur* $D(A)$ *l'ensemble*

$$\sigma_p(A) = \{z \in \mathbb{C}, (z - A) : D(A) \to \mathcal{H} \text{ non-injectif}\} = \{z \in \mathbb{C}, \exists \psi \in D(A), \phi \neq 0, A\psi = z\psi\} \tag{A.6}$$

qui est donc l'ensemble de ses valeurs propres.

On note $\mathcal{H}_p$ la somme directe des espaces sous-espaces propres associés à toutes les valeurs propres ainsi obtenues :

$$\mathcal{H}_p = \bigoplus_{\lambda \in \sigma_p(A)} E_\lambda \tag{A.7}$$

avec la relation de Parseval

$$\sum_{\lambda \in \sigma_p(A)} \|\psi_\lambda\|^2 = \|\psi\|^2 < +\infty \tag{A.8}$$

Théorème A.0.7. *(Diagonalisation ponctuelle d'opérateurs) Si A est un opérateur auto-adjoint tel que $\mathcal{H}_p = \mathcal{H}$, alors A est diagonalisable en base orthonormée. Autrement dit, on peut trouver une suite $(\lambda_n)_{n\in\mathbb{N}}$ réels, et une suite $(\psi_n)_{n\in\mathbb{N}}$ normés deux à deux orthogonaux, tels que pour tout entier naturel n,*

$$A\psi_n = \lambda_n \psi_n \tag{A.9}$$

De plus,

$$\sigma(A) = \overline{\sigma_p(A)} \tag{A.10}$$

et

$$D(A) = \left\{ \psi \in \mathcal{H}, \sum_{n\in\mathbb{N}} (1 + |\lambda_n|^2) \cdot |\langle \psi_n | \psi \rangle|^2 < +\infty \right\} \tag{A.11}$$

En utilisant le formalisme de Dirac*, nous avons*

$$\psi = \sum_{n\in\mathbb{N}} \lambda_n |\psi_n\rangle\langle\psi_n| \tag{A.12}$$

Théorème A.0.8. *(Spectre d'un opérateur auto-adjoint quelconque) Soit A un opérateur auto-adjoint quelconque sur $\mathcal{H}$. et $\sigma_p(A)$ son spectre ponctuel, et*

$$\sum_{\lambda \in \sigma_p(A)} \ker(\lambda \mathrm{id} - A) \tag{A.13}$$

Alors,

- *Soit $\mathcal{H}_p = \mathcal{H}$, et A est diagonalisable ponctuellement dans une base dénombrable,*
- *Soit $\mathcal{H}_p \neq \mathcal{H}$, dans ce cas $\mathcal{H}_c = \mathcal{H}_p^\perp$ et $\mathcal{H}_c$ et $\mathcal{H}_p$ sont stables par A, et*

$$\sigma(A) = \overline{\sigma_p(A)} \cup \sigma_c(A) \tag{A.14}$$

Annexe B

Algorithmes sous-optimaux

B.0.1 Détail de l'algorithme pour calculer les perturbations avec les gradients conjugués sur GPU

Le code de l'algorithme est donné en 2.

- Les produits matrice-vecteur et matrice-matrice se font arbitrairement par convention sur GPU.
- Les produits vecteurs-vecteur peuvent se faire sur CPU
- Les transferts de mémoire GPU-CPU doivent être minimisés autant que possible
- les transferts de mémoire GPU-CPU doivent être substitués par des transferts GPU-GPU autant que possible
- Un même calcul coûteux (ex. produit matrice-matrice) ne doit jamais être refait plusieurs fois
- Un produit matrice-matrice `sparse*dense` est supporté par Julia CUDA à ce jour mais pas `dense*sparse`
- Tout stockage de variable coûteuse (ex. matrice pleine) comme résultat intermédiaire doit être évité autant que possible.

Algorithm 2 Théorie des perturbations avec les gradients conjugués

Input : $H^0, W \in \mathbb{R}^{N\times N}$ avec H^0 creuse $E_0 \in \mathbb{R}$; $\mathbf{\Psi}_0 \in \mathbb{R}^N \|\mathbf{\Psi}_0\|_2 = 1$ un hamiltonien, sa perturbation et les éléments propres d'un état non dégénéré de $\hat{H}^0$
Input : $Q_{\max} \in \mathbb{N}^\star$: ordre maximal du terme à calculer
Output : $(E_i)_{1\leq i\leq Q_{\max}}$, et $(\mathbf{\Psi}_i)_{1\leq i\leq Q_{\max}}$

▷ « $\triangle$ » signifie calcul sur CPU affecté sur mémoire CPU
▷ « $\boxplus$ » signifie calcul sur GPU affecté sur mémoire GPU
▷ « $\twoheadleftarrow$ » signifie transfert mém. GPU vers mém. CPU
▷ « $\hookleftarrow$ » signifie transfert mém. CPU vers mém. GPU

procedure CALCUL DES ORDRES SUR CPU/GPU
  $\mathbf{\Psi}_0, W, H^0 \hookleftarrow \Psi_0, W, H^0$ ▷ Copie sur mém. GPU en gardant les types `sparse`
  $\Pi^{\|} \leftarrow_{\boxplus} \mathbf{\Psi}_0 \times \mathbf{\Psi}_0^\intercal$ ▷ Création des projecteurs
  $\Pi^{\perp} \leftarrow_{\boxplus} \texttt{Diagonal}(\mathbf{1}_{\mathbb{R}^N}) - \Pi^{\|}$
  $P^{\perp} \leftarrow_{\boxplus} H^0 \times \Pi^{\perp}$ ▷ Produit `sparse-dense` seul supporté par Julia CUDA
  $P^{\|} \leftarrow_{\boxplus} H^0 \times \Pi^{\|}$
  $P^{\perp} \leftarrow_{\boxplus} \Pi^{\perp} \times P^{\perp} - E_0 \cdot \Pi^{\perp}$ ▷ Créations des opérateurs LHS
  $P^{\|} \leftarrow_{\boxplus} \Pi^{\|} \times P^{\|} - E_0 \cdot \Pi^{\|}$
  $l\mathbf{\Psi} \leftarrow_{\triangle} [\mathbf{0}_{\mathbb{R}^N}, \cdots, \mathbf{0}_{\mathbb{R}^N}]$ de longueur $Q_{\max}$ ▷ Affectation mémoire tous termes
  $lE \leftarrow_{\triangle} [0, \cdots, 0]$, de longueur $Q_{\max}$
  $R^{\perp} \leftarrow_{\boxplus} -P^{\perp} \times W$ ▷ Création opérateur RHS $\perp$
  $\mathbf{b} \leftarrow_{\boxplus} R^{\perp} \times \Psi_0$ ▷ Accumulateur RHS des GC GPU init. à l'o. 1 opti.
  $\mathbf{b}_{\text{acc}} \leftarrow_{\triangle} \mathbf{0}_{\mathbb{R}^N}$ ▷ Accumulateur RHS des GC CPU
  $\mathbf{\Psi}_{\text{acc1}} \leftarrow_{\boxplus} \mathbf{0}_{\mathbb{R}^N}$ ▷ Accumulateur Ψ_{q-1}
  $\mathbf{\Psi}_{\text{acc2}} \leftarrow_{\boxplus} \mathbf{0}_{\mathbb{R}^N}$ ▷ Accumulateur Ψ_q
  $\mathbf{\Psi}_{\text{acc1}} \leftarrow_{\boxplus} \text{cg}_{\boxplus}(P^{\perp}; \mathbf{b}; \varepsilon)$ ▷ Calcul GC ordre 1
  $\mathbf{\Psi}_{\text{acc2}} \leftarrow_{\boxplus} \mathbf{\Psi}_{\text{acc1}}$ ▷ $\Psi_{\text{acc1}} \equiv \Psi_{\text{acc2}} \equiv \psi_{q-1}$
  $l\mathbf{\Psi}[1] \twoheadleftarrow \mathbf{\Psi}_{\text{acc1}}$ ▷ Sauvegarde $\boxplus$
  $lE[1] \twoheadleftarrow \Psi_0^\intercal \times W \times \Psi_0$ ▷ & sauvegarde $\triangle$
  $R^{\|} \leftarrow_{\boxplus} -P^{\|} \times W + lE[1] \cdot P^{\|}$ ▷ Création opérateur RHS $\|$
  $R^{\perp} \leftarrow_{\boxplus} R^{\perp} + lE[1] \cdot P^{\perp}$ ▷ Màj opérateur RHS $\perp$
  $\mathbf{f} \twoheadleftarrow \Psi_0^\intercal (W - lE[1] \cdot \texttt{Diagonal}(\mathbf{1}_{\mathbb{R}^N}))$ ▷ Création forme linéaire premier terme **??**
  for $q \in [\![2, Q_{\max}]\!]$ **do** ▷ Boucle traitement termes ordre q
    $lE[q] \leftarrow_{\triangle} \mathbf{f} \times l\Psi[q-1]$
    for $i \in [\![1, q-2]\!]$ **do** ▷ Calcul de E_q
      $lE[q] \leftarrow_{\triangle} lE[q] - lE[q-i] \cdot \texttt{dot}(l\Psi_0, l\mathbf{\Psi}[i])$ ▷ Calcul **??**, avec `dot` sur CPU
    end for
    $\mathbf{b} \leftarrow_{\boxplus} \mathbf{0}_{\mathbb{R}^{\mathbb{N}}}$ ▷ Réinit. $\mathbf{b}$
    $\mathbf{b}_{\text{acc}} \leftarrow_{\triangle} \mathbf{0}_{\mathbb{R}^{\mathbb{N}}}$ ▷ Réinit. $\mathbf{b}_{\text{acc}}$
    for $i \in [\![1, q-2]\!]$ **do**
      $\mathbf{b}_{\text{acc}} \leftarrow_{\triangle} \mathbf{b}_{\text{acc}} + lE[q-i] \cdot l\mathbf{\Psi}[i]$ ▷ Calcul $\sum_{i=1}^{q-2} E_{q-i}|\psi_i\rangle$ dans 2.68 & 2.69
    end for
    $\mathbf{b}_{\text{acc}} \leftarrow_{\boxplus} \mathbf{b}_{\text{acc}} + lE[q] \cdot \mathbf{\Psi}_0$ ▷ Calcul $\sum_{i=0}^{q-2} E_{q-i}|\psi_i\rangle$ dans 2.68 & 2.69
    $\mathbf{b} \hookleftarrow \mathbf{b}_{\text{acc}}$ ▷ Recopie sur GPU pr utiliser ds les GC
    $\mathbf{\Psi}_{\text{acc1}} \leftarrow_{\boxplus} \text{cg}_{\boxplus}(P^{\perp}; \Pi^{\perp} \times \mathbf{b} + R^{\perp} \times \mathbf{\Psi}_{\text{acc1}}; \varepsilon)$ ▷ Calcul GC ordre q $\perp$
    $\mathbf{\Psi}_{\text{acc2}} \leftarrow_{\boxplus} \text{cg}_{\boxplus}(P^{\|}; \Pi^{\|} \times \mathbf{b} + R^{\|} \times \mathbf{\Psi}_{\text{acc2}}; \varepsilon) + \mathbf{\Psi}_{\text{acc1}}$ ▷ Calcul GC ordre q $\|$
    $\mathbf{\Psi}_{\text{acc1}} \leftarrow_{\boxplus} \mathbf{\Psi}_{\text{acc2}}$ ▷ màj $\mathbf{\Psi}_{\text{acc1}}$
    $l\mathbf{\Psi}[q] \twoheadleftarrow \mathbf{\Psi}_{\text{acc2}}$ ▷ & sauvegarde $\triangle$
  end for
end procedure

Annexe C

Théorèmes utiles en analyse numérique courante

C.1 Matrices du laplacien 1D et 2D

On va utiliser les différences finies avec des schémas de laplacien à 3 ou 5 points : les matrices obtenues sont hermitiennes (ou anti-hermitiennes) en particulier et des algorithmes efficaces existent en Julia pour traiter les matrices hermitiennes (inverse, résolution système linéaire, valeurs propres, etc).

Le schéma à trois points du laplacien 1D pour estimer le laplacien d'une fonction u à l'aide des points x_{i-1}, x_i, x_{i+1} espacés d'un pas d'espace h :

$$\left| u''(x_i) - \frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} \right| \leq \frac{h^2}{12} \max_{\xi \in [x_i - h, x_i + h]} |u^{(4)}(\xi)| \tag{C.1}$$

et pour le vecteur $U^{1\mathrm{D}} = (u_1, u_2, \cdots, u_N)^\top$ la matrice des valeurs du laplacien négatif $-\Delta$ présent dans l'équation de Schrödinger ainsi que dans le problème de Dirichlet prise que les points de l'axe vaut donc $\Lambda^{1\mathrm{D}} U^{1\mathrm{D}}$ avec

$$\Lambda^{1\mathrm{D}} = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & & \\ & & & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N \times N} \tag{C.2}$$

tandis que l'approximation du laplacien à 5 points vérifie

$$\begin{aligned} &\left| \Delta u(x_i, y_i) - \frac{u(x_{i+1}, y_i) + u(x_i, y_{i+1}) - 4u(x_i) + u(x_{i-1}, y_i), u(x_i, y_{i-1})}{h^2} \right| \\ &\qquad \leq \frac{h^2}{12} \max_{\xi \in [x_i - h, x_i + h]} |\partial^{(4)}_{xxxx} u(\xi, y_i)| + \frac{h^2}{12} \max_{\xi \in [y_i - h, y_i + h]} |\partial^{(4)}_{yyyy} u(x_i, \xi)| \end{aligned} \tag{C.3}$$

et pour le vecteur des points de la grille 2D *column-wise* (comme la convention du langage JULIA) $U^{2\mathrm{D}} = (u_{1,1}, u_{2,1}, \cdots, u_{N,1} | \cdots, \cdots, \cdots, | u_{1,N}, u_{2,N}, \cdots, u_{N,N})^\top$, la matrice des valeurs

du laplacien négatif $-\Delta$ présent dans l'équation de Schrödinger ainsi que dans le problème de Dirichlet prise que les points de la grille vaut donc $\Lambda^{2D}U^{2D}$ avec

$$\Lambda^{2D} = \begin{pmatrix} M & -I & & & & \\ -I & M & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & M & -I \\ & & & & -I & M \end{pmatrix} \in \mathbb{R}^{N^2 \times N^2} \tag{C.4}$$

et

$$M = \begin{pmatrix} 4 & -1 & & & & \\ -1 & 4 & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & & & \\ & & & 4 & -1 & \\ & & & -1 & 4 & \end{pmatrix} \in \mathbb{R}^{N \times N} \tag{C.5}$$

Et ces matrices Λ^{1D} et Λ^{2D} présentent l'intérêt considérable d'être creuses et hermitiennes, ce qui facilite grandement les calculs, les accélèrent, que ce soit pour la multiplication, en appliquant une méthode à base de puissances itérées, ou pour l'inversion, notamment grâce à la décomposition de Cholesky. Leurs valeurs propres et vecteurs propres sont notamment aussi connus analytiquement, on en trouve la preuve dans beaucoup de livres :[1]

$$\lambda_j^{1D} = \frac{4}{h^2}\sin^2\left(\frac{j\pi}{2N}\right), \qquad \lambda_{i,j}^{1D} = \frac{4}{h^2}\left[\sin^2\left(\frac{i\pi}{2N}\right) + \sin^2\left(\frac{j\pi}{2N}\right)\right] \tag{C.6}$$

ce qui permet d'avoir directement leur conditionnement en norme 2, puisqu'elles sont symétriques, il vaut le rapport de la plus grande valeur propre (en valeur absolue) sur la plus petite, ce qui fait que les deux matrices en 1D ou 2D ont un conditionnement asymptotique de $\kappa \sim_{N\to+\infty} \frac{4N^2}{\pi^2}$. Autrement dit plus on discrétise finement, pire est le conditionnement, ce qui n'est pas idéal.

Les vecteurs propres associés à Λ^{1D} sont les (x_i) pour $1 \leq i \leq N$ avec

$$(x_i)_k = \sin\left(\frac{ik\pi}{N+1}\right), \quad 1 \leq k \leq N \tag{C.7}$$

et ceux associés à Λ^{2D} sont les $(x_{i,j})$ pour $1 \leq i, j \leq N$ avec

$$(x_{i,j})_{k,l} = \sin\left(\frac{ik\pi}{N+1}\right)\sin\left(\frac{jl\pi}{N+1}\right), \quad 1 \leq k, l \leq N \tag{C.8}$$

On peut voir voir Λ^{2D} comme une « superposition » de deux Λ^{1D} dont chacun agirait indépendamment sur une dimension caractéristique d'un vecteur-matrice indicé sur deux dimension. C'est en fait exactement ça, mais par commodité en mathématiques (et en informatique), on déplie les vecteurs-matrices et on assemble les opérateurs agissant sur ces derniers. Cela se généralise En pratique, dans ce stage, pour la partie calcul de la solution de référence on travaille avec une superposition pondérée des deux opérateurs car les échelles d'espace et les masses réduites associées à chaque degré de liberté du problème ne sont pas les mêmes.

[1] Sources : https://www.ljll.math.upmc.fr/~frey/ftp/finite-differences.pdf ; https://who.rocq.inria.fr/Michel.Kern/Docs/MACS2_AnaNumAv_coursMK.pdf ; https://giref.ulaval.ca/~fkwok/docs/eig_notes.pdf

C.2 Recherches d'éléments propres

Méthodes de la puissance

Théorème C.2.1. *(Méthode de la puissance dans le cas général, sources multiples) Soi A une matrice de valeurs propres de modules ordonnés $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Soit u_0 un vecteur normé et définissons la suite par récurrence pour $t \in \mathbb{N}$, $u_{t+1} = \frac{Au_t}{\|Au_t\|}$.*

Si u_0 n'appartient pas aux sous-espaces propres des $(\lambda_i)_{i\geq 2}$, nous avons alors

$$\langle Au_t | u_t \rangle \xrightarrow[t\to+\infty]{} \lambda_1 \tag{C.9}$$

Remarque C.2.2. *Si $A \in \mathscr{S}_n^+(\mathbb{R})$ a pour valeur propre maximale connue λ_{max} et de valeur propre minimale λ_{min}, alors la méthode de la puissance appliquée à $A - \lambda_{max} I$ permet d'approcher $\lambda_{min} - \lambda_{max}$.*

Ces deux méthodes itératives très simples sont avantageuses car on sait faire le produit rapide de matrices (sur GPU, par exemple, si elles sont très grandes). C'est surtout très intéressant si la matrice en question est creuse, car le nombre de produits et sommes est encore plus réduit.

Théorème C.2.3. *(Méthode de la puissance dans le cas positif semi-défini[2]) Soit $A \in \mathscr{S}_n^+(\mathbb{R})$ de valeurs propres $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n$, soit $u_0 = (\frac{\varepsilon_1}{\sqrt{n}}, \cdots, \frac{\varepsilon_n}{\sqrt{n}})^\top$ où $(\varepsilon_i) \underset{i.i.d.}{\sim} \mathcal{U}(\{-1, +1\})$, soit v un vecteur tel que $\|v\| = 1$ et $Av = \lambda_1 v$, enfin définissons la suite par récurrence pour $t \in \mathbb{N}$, $u_{t+1} = \frac{Au_t}{\|Au_t\|}$.*

Alors avec une probabilité supérieure ou égale à $\frac{3}{16}$, nous avons

$$|\langle u_t | v \rangle| \geq 1 - 2n \left(\frac{\lambda_2}{\lambda_1}\right)^{2t} \tag{C.10}$$

Lorsque la matrice n'est pas particulièrement grande mais non creuse, les algorithmes reposent sur des factorisations avantageuses qui simplifient la recherche des éléments propres. Par exemple, la décomposition LU, ou celle de Cholesky pour les matrices hermitiennes.

C.3 Localisation de spectres

Il existe des théorèmes sur la localisation du spectre d'une matrice en général, par exemple les disques de Gershgorin, ce qui permet de « deviner » une valeur a priori :

Théorème C.3.1. *(Disques de Gershgorin) Soit A une matrice complexe carrée de terme a_{ij}. Toute valeur propre de A se trouve dans au moins un des disques suivante pour $i \in [\![1, n]\!]$:*

$$D_i = \left\{ z \in \mathbb{C}, |a_{ii} - z| \leq \sum_{j\neq i} |a_{ij}| \right\} = D(a_{ii}, R_i) \tag{C.11}$$

Et on peut faire de même avec la transposée, si la matrice n'est pas symétrique, ce qui permet de localiser le spectre dans l'intersection des unions de disques. Ce théorème se prouve facilement en écrivant l'égalité $Au = \lambda u$ pour un vecteur propre et d'appliquer ligne à ligne l'inégalité triangulaire en norme 2. Dans la même veine, on trouve également un résultat connu sous le nom d'ovales de Cassini.

[2]Source : cours INFO4220 2021-2022 de l'ENS Lyon, repris lui-même de http://theory.stanford.edu/~trevisan/expander-online/lecture03.pdf

Ainsi il est possible, étant donné une matrice, d'approcher les racines de son polynôme caractéristique (donc ses valeurs propres), ce qui permet également de trouver les racines de polynômes à l'aide de leurs matrices compagnon, tandis que passer par les polynômes pour trouver les valeurs propres est malaisé dès que le degré augmente. On peut aussi montrer que si un disque est isolé, il contient une seule valeur propre.[3]

Théorème C.3.2. *(Cours MDI210, Télécom Paris, 2022, §2.2.2) Soit A une matrice diagonalisable et P la matrice de passage dans la base de vecteurs propres et (λ_i) ses valeurs propres. Soit $\|\cdot\|$ une norme matricielle telle que pour toute matrice diagonale $D = diag(a_1, \cdots, a_n)$, on ait $\|D\| = \max(a_i)$. Alors pour toute matrice δA,*

$$Sp(A + \delta A) \subset \bigcup_{1 \leq i \leq n} D_i \tag{C.12}$$

où

$$D_i = \{z \in \mathbb{C}, |z - \lambda_i| \leq cond_{\|\cdot\|}(P)\|\delta A\|\} \tag{C.13}$$

Pour les matrices symétriques réelles, qui sont orthogonalement diagonalisables d'après le théorème spectral, le conditionnement de la matrice de passage est donc 1 en norme 2 et on a l'énoncé du cas particulier [MDI210 page 18] : si A est la matrice symétrique réelle originale et $\alpha_1 \leq \cdots \leq \alpha_n$ ses valeurs propres ordonnées, et $B = A+\delta A$ la matrice perturbée et $\beta_1 \leq \cdots \leq \beta_n$ ses valeurs propres perturbées, alors pour tout i on a $|\alpha_i - \beta_i| \leq \|\delta A\|_2$.

C.3.1 Exemples numériques et comparaison des performances

Voici par exemple un échantillon de codes specimen typiques qui montrent le calcul d'éléments propres grâce à certaines desdites bibliothèques en Julia, en plus de `eigvals` et `eigvecs` du paquet `LinearAlgebra` :

Calcul de valeur propre sur GPU CUDA avec csreigvsi

```
using CUDA
Λ_gpu = CUDA.CUSPARSE.CuSparseMatrixCSR{Float32}(Λ)
x_0 = CUDA.cu(rand(Float32, N))
CUDA.@time μ, x = CUDA.CUSOLVER.csreigvsi(Λ_gpu, convert(Float32,-0.82), x_0,
↪  convert(real(Float32), 1e-8), convert(Cint,100),'O')
```

Calcul de valeur propre sur CPU avec une méthode de Krylov

```
using KrylovKit
x_0 = rand(Float32, N);
@time vals, vecs, infos = KrylovKit.eigsolve(convert(SparseMatrixCSC{Float32, Int64},
↪  Λ), x0, 1, :SR, krylovdim=10);
```

Protocole de comparaison numérique sur un exemple de l'état de l'art

Comparons les méthodes itérées sur CPU et GPU à la matrice C.4, on peut comparer leur précision vis-à-vis de la référence dont on connaît une expression analytique. Je calcule avec `KrylovKit` sur la matrice représentée en précision simple et double l'état fondamental, et je fais de même sur le GPU avec la matrice en précision simple. Puis je compare à la référence calculée auparavant, exprimée en précision double. Les résultats pour une grille 80×80 sont dans les

[3] Source http://citron.9grid.fr/docs/gerschgorin.pdf

Λ^{2D} 80×80	Temps de calcul (secondes)	Erreur réf. vecteur propre (relatif, norme 2)	Erreur réf. valeur propre (relatif, abs)
GPU csreigvsi FP32	$\sim 5 \times 10^{-1}$	1×10^{-4}	2×10^{-6}
CPU Krylov FP32	$\sim 5 \times 10^{-2}$	2×10^{-6}	3×10^{-11}
CPU Krylov FP64	$\sim 8 \times 10^{-2}$	1×10^{-12}	6×10^{-14}

Table C.1 : Ordres de grandeur des résultats pour le calcul des éléments propres de Λ^{2D} de taille 80×80

Λ^{2D} 120×120	Temps de calcul (secondes)	Erreur réf. vecteur propre (relatif, norme 2)	Erreur réf. valeur propre (relatif, abs)
GPU csreigvsi FP32	~ 1.5	8×10^{-4}	4×10^{-6}
CPU Krylov FP32	~ 0.5	2×10^{-4}	3×10^{-7}
CPU Krylov FP64	~ 1.3	3×10^{-12}	1×10^{-12}

Table C.2 : Ordres de grandeur des résultats pour le calcul des éléments propres de Λ^{2D} de taille 120×120

tableaux C.1 et C.2. Les valeurs sont relatives (pour les énergies par rapport à la référence, et pour les vecteurs également par rapport à la norme de la référence). Les chiffres significatifs sont indicatifs (je n'ai pas fait d'étude statistique), mais les ordres de grandeur suffisent à conclure. De même, les comparaisons à la référence sont pris pour une référence utilisant les fonctions natives de Julia (sin, cos, etc) qui a priori n'apportent pas d'erreur aux chiffres des ordres de grandeur des erreurs (c'est-à-dire que je suppose que les valeurs retournées sont correctes à la précision machine près). *Il est assez clair que l'utilisation du GPU n'est pas très avantageuse ni en précision, ni en temps de calcul.* Ces tableaux sont uniquement là pour montrer que l'on ne gagne pas toujours à utiliser le GPU. Dans le problème du stage, on utilise des matrice avec potentiel (ici, ce n'est pas le cas). Ceci justifie que je n'aie jamais utilisé le GPU dans mes codes terminés pour simuler mon problème.

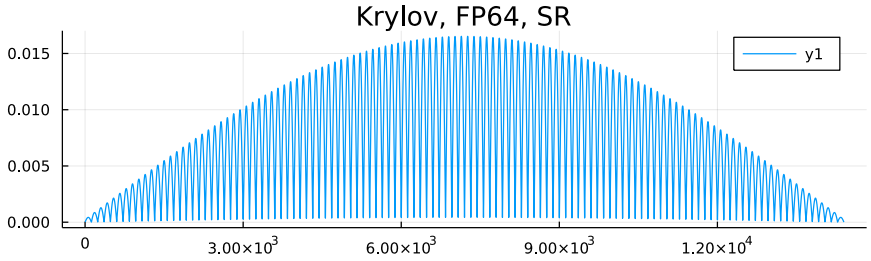


Figure C.1 : Vecteur propre normé de plus petite valeur propre du laplacien à 5 points 2D pour une grille 120×120, le vecteur est donc de dimension 14400. Ici calculé avec une méthode de Krylov en précision double.

Annexe D

Résultats partiels ou connexes

D.1 Optimisation de l'algorithms en fonction du *hardware*

Pour cette section, imaginons que nous disposions d'un algorithme constitué d'un ensemble d'instructions (une instruction consiste typiquement à un ou plusieurs calculs et un stockage dans la mémoire du résultat) numérotées de 1 à I, qui néanmoins doivent être exécutées dans un certain ordre bien précis.

Les données sur lesquelles nous travaillons sont des variables de taille n et d'ordre k avec k pouvant prendre plusieurs valeurs dans un ensemble $[\![0, k_{\max}]\!]$. Typiquement, nos variables peuvent être des vecteurs, des matrices, des tenseurs, etc occupant une taille mémoire en rapport avec leur ordre (matrice d'ordre 2 car stocke n^2 nombres, vecteur d'ordre 1 car n^1, etc), et nos opérations des produits matrice-matrice (ordre $k = 3$ car nécessite $\mathcal{O}(n^3)$ produits et sommes de réels), des produits matrice-vecteur (ordre $k = 2$ car nécessite $\mathcal{O}(n^2)$ produits et sommes de réels), des produits scalaire-matrice (ordre $k = 2$). Pour exécuter ces instructions nous avons deux agents, chacun ayant des caractéristiques différentes, l'un peut-être plus rapide pour certaines opérations, l'autre plus rapide pour d'autres, ou seulement asymptotiquement, etc.

Lorsqu'un algorithme tel que 2 a besoin d'être exécuté, supposons avoir le choix pour chaque instruction de demander à l'agent 1 dénoté A_1 ou l'agent 2 dénoté A_2 d'exécuter l'instruction. On peut répartir les tâches entre les deux agents. Mais il y a un hic : chaque agent dispose de sa propre mémoire sur laquelle il opère exclusivement. La communication est possible entre les deux mémoires mais tout transfert d'information se fait avec un certain coût en temps, qui peut avoisiner voire dépasser l'ordre de grandeur du coût en temps de calcul.

Nous venons de décrire une situation courante : celle d'un GPU et d'un CPU sur la carte mère. Le GPU est un ensemble d'un grand nombde de processeurs spécialisés qui, s'ils sont correctement utilisés, peuvent se révéler très rapide et efficaces à certains calculs parallélisables, tandis que le CPU est une unité très polyvalente, extrêmement rapide (souvent plus rapidement cadencée que les GPU), mais dont les capacités de parallélisations sont très limitées (souvent quelques cœurs contre quelques dizaines ou centaines de cœurs pour un GPU moderne), ce qui peut avoir raison de lui pour les très gros calculs. Si le GPU est très imposant, on pourrait stocker l'entièreté des données de départ et résultats générés sur sa mémoire, c'est le cas le plus favorisé en *machine learning* par exemple, lorsqu'un réseau de neurones complet doit être stocké sur le GPU afin de l'évaluer rapidement, estimer les gradients, et rétropropager le gradient pour l'entraîner. Mais si le GPU est limité, toutes les données en mémoire ne tiennent pas nécessairement.

On impose également que les variables n'aient pas le don d'ubiquité, pour s'interdire tout problème de synchronisation, qui fait partie de l'étude des réseaux.

Le problème devient maintenant clair : paralléliser de gros calculs opérant sur de grosses données est intéressant mais le transfert préalable et ultérieur à un calcul peut changer la donne. Le problème du choix est *a priori* complexe : il existe naïvement 2^I paramétrages (deux choix

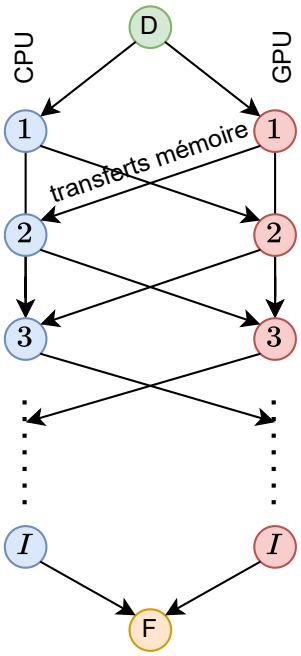


Figure D.1 : Modélisation des parcours possibles. Dans ce pseudo-graphe, les trajectoires sont mises en avant : ici les arêtes représentent les transferts mémoire et les nœuds représentent les calculs.

pour chaque instruction). **Il est important de noter que ce proglème ne se résout pas facilement par une approche « diviser pour régner « . En effet, un optimum global n'est pas constituée d'assemblage d'optima locaux : la raison est que la présence en mémoire d'une variable se propage indéfiniment dans le temps tant qu'elle n'est pas déplacée ou modifiée, expliquant la difficulté principale : les choix antérieurs de déplacement mémoire des variables impactent les étapes ultérieures.**

Je vais décrire une heuristique permettant de trouver une bonne solution à ce théorème avec quelques autres contraintes évidentes, orientées par le cas des calculs matriciels. Mais l'algorithme est généralisable à des cas moins spécifiques facilement.

On peut donc modéliser la situation par le pseudo-graphe D.1. Les arêtes représentent les transferts mémoire et les sommets les calculs des instructions (un sommet correspond à une ligne de l'algorithme effectuée sur un processeur. Néanmoins, il ne s'agit pas encore d'un graphe, car le coût en transfert mémoire en passant d'une étape de l'algorithme à une autre dépend du chemin précédemment emprunté, puisque l'on a supposé que les variables ne sont pas omniprésentes et ne sont bougées que lorsque c'est nécessaire.

Formalisation

Énumérons les hypothèses

1. La taille typique des données est n.
2. Chaque variable est indicée par $v \in [\![1, V]\!]$.
3. Au début, seules les variables utilisées au départ (en entrée de l'algorithme sont allouées dans la mémoire, et sont présentes sur le la mémoire de l'agent A_1 (CPU).

4. Chaque variable est mutable mais typée de type immuable.
5. Chaque variable v possède en attribut un ordre $k(v) \in [\![0, k_{\max}]\!]$.
6. Chaque instruction est indicée par $i \in [\![1, I]\!]$.
7. Chaque instruction i se compose d'une combinaison linéaire de coûts de calculs $\kappa_{i,j}$ d'ordres $k(i,j)$ compris dans $[\![1, k_{\max}]\!]$.
8. Chaque calcul j composant une instruction i se fait en un coût $c_{i,j}$ (on s'attend à ce que $c_{i,j} \propto n^{k(i,j)}$) et ainsi $c_i = \sum_j c_{i,j}$.
9. Il est possible de transférer les variables de la mémoire de l'agent A_1 vers celle de l'agent A_2 avec un coût proportionnel à l'ordre de la variable.
10. Une variable n'est transférée avant l'exécution de l'instruction qu'en cas de besoin : pour y accéder lors d'un calcul ou pour faire de la place pour en stocker d'autres sur celle de l'agent A_2.
11. Chaque instruction nécessite certaines variables en accès d'écriture ou de lecture pour effectuer ses calculs.
12. Toute variable ne peut pas coexister simultanément dans la mémoire de l'agent A_1 et de l'agent A_2 simultanément.
13. La mémoire $\mathcal{M}$ de l'agent A_2 (GPU) est limitée : $\mathcal{M} < +\infty$.
14. Chaque variable v occupe une taille mémoire m_v (on s'attend à ce que $m_v \propto n^{k(v)}$).
15. À chaque étape du calcul on associe son *état final* (état et étape seront donc synonymes), caractérisé uniquement par les variables présentes dans les mémoires respectives des agents A_1 et A_2 qui restent utiles pour la suite (stricte) de l'algorithme. C'est-à-dire que l'on libère les mémoires des variables dorénavant inutiles à la fin de chaque instruction.
16. À la fin de l'algorithme, certaines variables sont retournées en sortie de l'algorithme, et doivent être présentes sur la mémoire de l'agent A_1 (CPU).

Formalisons tout cela et définissons ainsi :

- La matrice $\mathbf{B} \in \mathscr{M}_{(V,I)}(\{0,1\})$ telle que $\mathbf{B}[v,i] = 1$ si et seulement si l'instruction i nécessite l'accès de la variable v en lecture ou écriture.
- La matrice $\mathbf{K} \in \mathscr{M}_{(k_{\max+1,I})}(\mathbb{N})$ telle que $\mathbf{K}[i,k]$ désigne ne nombre d'opérations d'ordre $k-1$ nécessaires à l'instruction i.
- Le vecteur $\mathbf{D} \in \{0,1\}^V$ tel que $\mathbf{D}[v] = 1$ si et seulement si la variable v fait partie des variables du début (en entrée de l'algorithme).
- Le vecteur $\mathbf{F} \in \{0,1\}^V$ tel que $\mathbf{F}[v] = 1$ si et seulement si la variable v fait partie des résultats retournés (sorties de l'algorithme).
- Le vecteur $\mathbf{M} \in \mathbb{R}_+^V$ tel que $\mathbf{M}[v]$ désigne l'occupation mémoire de la variable v.
- Le vecteur $\mathbf{T} \in \mathbb{R}_+^V$ tel que $\mathbf{T}[v]$ désigne le coût d'un transfert de la variable v entre le CPU et le GPU (dans un sens ou dans l'autre).
- Le vecteur $\mathbf{\Omega} \in \mathbb{N}^V$ tel que $\mathbf{\Omega}[v]$ désigne l'ordre de la variable v.

- Le vecteur $\mathbf{\Xi}_{\text{CPU}} \in \mathbb{R}_+^{k_{\max}+1}$ tel que $\mathbf{C}_{\text{CPU}}[k]$ désigne le coût d'une opération d'ordre $k-1$ sur le CPU.
- Le vecteur $\mathbf{\Xi}_{\text{GPU}} \in \mathbb{R}_+^{k_{\max}+1}$ tel que $\mathbf{C}_{\text{GPU}}[k]$ désigne le coût d'une opération d'ordre $k-1$ sur le GPU.
- Le vecteur $\mathbf{C}_{\text{CPU}} = \mathbf{K}^\intercal \times \mathbf{\Xi}_{\text{CPU}} \in \mathbb{R}_+^I$ tel que $\mathbf{C}_{\text{CPU}}[k]$ désigne le coût de l'étape i de l'algorithme sur le CPU.
- Le vecteur $\mathbf{C}_{\text{GPU}} = \mathbf{K}^\intercal \times \mathbf{\Xi}_{\text{GPU}} \in \mathbb{R}_+^I$ tel que $\mathbf{C}_{\text{GPU}}[k]$ désigne le coût de l'étape i de l'algorithme sur le GPU.

D.1.1 Approche par programmation dynamique

Une bonne idée pour résoudre le problème serait donc de construire un graphe qui décrive à chaque étape les choix correspondants. Pour obtenir un graphe, les poids doivent être des constantes. Il faudrait donc « déplier » ce pseudo-graphe : on obtiendrait ainsi un arbre binaire pour lequel on aurait relié toutes les feuilles à un dernier nœud unique. En pratique, il est possible que des nœuds en commun existent, si on aboutir à un même état par deux chemins différents. Il est donc peu judicieux (et peu efficace) de créer le pseudo-graphe en le dépliant. On va alors le construire directement de manière récursive. **Remarque importante : dès lors, sur le graphe déplié, les arêtes représenteront les coûts (en transfert mémoire et en calcul tous confondus, par exemple en équivalent de temps pour les deux) tandis que les nœuds représenteront les états de la mémoire (à profondeur constante) et la progression du calcul (à largeur constante).**

Une fois cela fait, il ne reste plus qu'à trouver le plus court chemin entre le premier nœud et le dernier.

Ce problème fait partie de la **programmation dynamique**, qui se définit grossièrement par une situation dans laquelle les décisions sont séquentielles et où notre système passe d'un état à un autre au fur et à mesure de ces décisions. Plus particulièrement on a besoin de trouver le plus court chemin dans un **graphe acyclique**. On peut y parvenir avec l'algorithme de Dijkstra, ou même encore d'une manière asymptotiquement plus efficace, récursivement, comme on le verra.

Construction du graphe

@algorithm 3 Construction du graphe des trajectoires de calcul

Input : $I, V, \mathbf{B}, \mathbf{K}, \mathbf{D}, \mathbf{F}, \mathbf{M}, \mathbf{\Omega}, \mathbf{T}, \mathbf{\Xi}_{\text{CPU}}, \mathbf{\Xi}_{\text{GPU}}$

▷ On note $\mathcal{N} = \times\{0,1\}^{2V} \times \mathbb{R}^2 \times \mathbb{Z} \times \{"CPU", "GPU"\}$ l'ensemble contenant les nœuds. Correspond au 6-uplet (présence CPU, présence GPU, occupation mémoire CPU, occupation mémoire GPU, état/profondeur, processeur). On note $p_{\text{CPU}}(i)$ et $p_{\text{GPU}}(i)$ respectivement les vecteurs de présence des variables 1 à V sur le CPU et le GPU à l'étape i.

Input : $h : \mathcal{N} \to H$ une fonction de hachage injective dans un ensemble H quelconque est l'ensemble des arêtes
Input : $\mathcal{D}$ une structure de dictionnaire de type $(\mathcal{K} \to \mathcal{V})$ où $\mathcal{K} \subset H$ et $\mathcal{V} \subset \mathbb{N}^+$
Input : $\mathcal{Q}$ une structure file (FIFO) de type $\mathcal{N}$
Input : $\mathcal{G}$ une structure de graphe orienté de type $(\mathcal{V} \times \mathcal{E})$ où $\mathcal{V} \subset \mathcal{N}$ est l'ensemble des nœuds et $\mathcal{E} \subset \mathbb{N}^+ \times \mathbb{N}^+$
Output : $G = (\mathcal{V} \times \mathcal{E})$ le graphe construit des états possibles de l'algorithme

procedure TRAITEMENT D'UN NOUVEAU NŒUD
  if $\texttt{haskey}(D, \eta)$ **then**
    $\mathrm{id} \leftarrow D[\eta]$
    $\texttt{addedgeweight}(G, (\mathrm{id}, s+1), (\mathbf{F} \wedge \neg \mathbf{p}_c)^\intercal \mathbf{T})$
  else
    $\mathrm{id} \leftarrow \mathrm{acc}$
    $\texttt{addnode}(G, \mathrm{id}, N_{\mathrm{new}})$
    $\texttt{addkeyvalue}(D, \eta, \mathrm{id})$
    $\texttt{enqueue}(Q, N_{\mathrm{new}})$
    $\texttt{addedgeweight}(G, (\mathrm{id}, 2), c)$
    $\mathrm{acc} \leftarrow \mathrm{acc} + 1$
  end if
end procedure

procedure CONSTRUCTION EN LARGEUR DU GRAPHE AVEC MÉMOÏSATION
  $\mathbf{C}_{\mathrm{CPU}} \leftarrow \mathbf{K}^\intercal \times \mathbf{\Xi}_{\mathrm{CPU}}$
  $\mathbf{C}_{\mathrm{GPU}} \leftarrow \mathbf{K}^\intercal \times \mathbf{\Xi}_{\mathrm{CPU}}$
  $G = \mathcal{G}(\emptyset, \emptyset)$ ▷ Création du graphe
  $Q = \mathcal{Q}(\emptyset)$ ▷ Création de la FIFO des nœuds
  $D = \mathcal{D}(\emptyset)$ ▷ Création du dictionnaire des nœuds

  for $\mathrm{id}, \mathbf{R}, s \in \texttt{zip}(\{1,2\}, \{\mathbf{D}, \mathbf{F}\}, \{0, I+1\})$ **do**
    $N \leftarrow (\mathbf{R}, \mathbf{0}^V, \mathbf{M}^\intercal \mathbf{R}, s, \text{"}CPU\text{"}) \in \mathcal{N}$ ▷ Création nœuds départ et arrivée
    $\texttt{addnode}(G, \mathrm{id}, N)$ ▷ Insérer ce nœud dans le graphe
    $\texttt{addkeyvalue}(D, h(N), \mathrm{id})$ ▷ Insérer ce nœud dans le dictionnaire
  end for
  $\texttt{enqueue}(Q, N)$ ▷ Enfiler ce nœud pour traitement ultérieur
  $\mathrm{acc} \leftarrow 3$ ▷ Accumulateur pour les identifiants (coloration)

  while $|q| > 0$ **do** ▷ Tant que la queue est non vide
    $N \leftarrow \texttt{dequeue}(q)$ ▷ Défiler le nœud à traiter en place
    $(\mathbf{p}_c, \mathbf{p}_g, m_c, m_g, s, t) \leftarrow N$ ▷ Extraire ses propriétés
    $\mathrm{id} \leftarrow D[h(N)]$ ▷ Déduire son identifiant
    if $s = I$ **then** ▷ S'il s'agit d'une feuille (fin de calcul)
      $\texttt{addedgeweight}(G, (\mathrm{id}, 2), c)$ ▷ Relier cette feuille au nœud final
    else
      $\mathbf{b} \leftarrow \mathbf{B}[:, s+1]$ ▷ Récupérer les variables nécessaires à l'étape suivante
      $\mathbf{b}_{\mathrm{ult}} = \mathrm{Vect}\{\texttt{Bool}\}((\mathbf{B}[:, s+2 : I] | \mathbf{F}) \times \mathbf{1}^{I-s}. > 0)$ ▷ $\in \{0,1\}^V$: besoins $\geq s+1$
      if then$(\mathbf{b} \vee \mathbf{p}_g)^\intercal \mathbf{M} \leq \mathcal{M}$ ▷ Calcul & import direct GPU possible
        $\mathbf{vi} \leftarrow \mathbf{b} \wedge \neg \mathbf{p}_g$ ▷ Variables importées sur GPU
        $c \leftarrow \mathbf{vi}^\intercal \mathbf{T} + \mathbf{C}_{\mathrm{GPU}}[s+1]$ ▷ Coût transferts + calculs GPU
        $\mathbf{p}_{g,\mathrm{new}} \leftarrow (\mathbf{b} \vee \mathbf{p}_g) \wedge \mathbf{b}_{\mathrm{ult}}$ ▷ Nouveau vecteur présence GPU
        $\mathbf{p}_{c,\mathrm{new}} \leftarrow (\mathbf{p}_c \wedge \neg \mathbf{vi}) \wedge \mathbf{b}_{\mathrm{ult}}$ ▷ Nouveau vecteur présence CPU
        $N_{\mathrm{new}} \leftarrow (\mathbf{p}_{c,\mathrm{new}}, \mathbf{p}_{g,\mathrm{new}}, \mathbf{p}_{c,\mathrm{new}}\mathbf{M}, \mathbf{p}_{g,\mathrm{new}}\mathbf{M}, \text{"}GPU\text{"})$ ▷ Créer le nœud-fils
        $\eta \leftarrow h(N_{\mathrm{new}})$ ▷ Calcul du hash du nœud-fils
        $\texttt{traitement}(G, \mathbf{id}, N_{\mathrm{new}}, D, \eta, c, s, \mathbf{F}, \mathbf{p}_c, \mathbf{T})$

      else if $\mathbf{b}^\intercal \mathbf{M} \leq \mathcal{M}$ **then** ▷ Calcul sur GPU possible si mémoire vidée
        $\mathbf{vie} \leftarrow \mathbf{b} \oplus \mathbf{p}_g$ ▷ Variables importées ou exportées
        $c \leftarrow \mathbf{vie}^\intercal \mathbf{T} + \mathbf{C}_{\mathrm{GPU}}[s+1]$
        $\mathbf{p}_{g,\mathrm{new}} \leftarrow \mathbf{b} \wedge \mathbf{b}_{\mathrm{ult}}$
        $\mathbf{p}_{c,\mathrm{new}} \leftarrow (\mathbf{p}_g \vee (\neg \mathbf{b} \wedge \mathbf{p}_g)) \wedge \mathbf{b}_{\mathrm{ult}}$

```
49 :                $N_{\text{new}} \leftarrow (\mathbf{p}_{c,\text{new}}, \mathbf{p}_{g,\text{new}}, \mathbf{p}_{c,\text{new}}\mathbf{M}, \mathbf{p}_{g,\text{new}}\mathbf{M}, \text{"}GPU\text{"})$
50 :                $\eta \leftarrow h(N_{\text{new}})$
51 :                traitement$(G, \mathbf{id}, N_{\text{new}}, D, \eta, c, s, \mathbf{F}, \mathbf{p}_c, \mathbf{T})$
52 :             else                                          ▷ Calcul sur CPU seulement possible
53 :                nothing
54 :             end if
55 :          end if
56 :       end while
57 : end procedure
```

Preuve de terminaison

L'algorithme 3 de création de l'arbre est en tous points similaire à un parcours d'arbre en largeur, avec marquage des sommets explorés (ou ici, créés). La boucle `while` est le point délicat qui empêche de trivialiser la preuve de terminaison.

Faisons quelques remarques :

(P1) Toutes les opérations de l'algorithme (chaque ligne) se font en temps fini donc chaque parcours de la boucle

(P2) Au cours d'un parcours de la boucle, un nœud est défilé, sinon l'exécution est terminée.

(P3) À cause de la vérification à l'aide de la fonction de hachage, les nœuds créés sont tous distincts 2 à 2, dans la file, le dictionnaire, ou le graphe.

(P4) Au cours d'un parcours de la boucle, si un défilement d'un nœud d'état $s < I$ a lieu, il y a au moins un et au plus deux enfilement(s) de nœuds fils d'état $s + 1$ dans la file.

(P5) Dans une boucle, ($\exists$ 1 enfilement et marquage d'un nœud d'état $s+1$) $\iff$ ($\exists$ 1 défilement d'un nœud d'ordre s).

(P6) Tout nœud d'état $s \geq I$ ne donne lieu à aucun ajout de nouveau nœud dans le graphe et dans le dictionnaire.

Démonstration. Par la propriété $P2$, prouver la terminaison de la boucle revient à montrer que la file se videra en temps fini.

Par la propriété $P1$, cela revient à montrer que la file se videra (simplement).

Par les propriétés $P5$ et $P6$, il ne peut pas exister dans la file des nœuds d'état $s > I$.

Par la propriété $P4$, on conclut qu'un nœud défilé n'est jamais réenfilé, car un nœud et un de ses fils diffèrent toujours au moins par leur état.

Par les propriétés $P3$ et $P4$, on peut aussi montrer immédiatement par récurrence qu'il existe au plus 2^s nœuds d'ordre s.

Avec ce qui précède et la propriété d'unicité $P3$, on conclut que peuvent exister dans la file et le dictionnaire, au plus $M = \sum_{s=1}^{I} 2^s$ nœuds, soit un nombre fini de nœuds, qui sont tous distincts.

Puisqu'on a montré qu'un nœud défilé n'est jamais enfilé à nouveau, qu'ils sont tous disctincts et en nombre fini, et qu'à chaque parcours de la boucle `while`, un nœud est défilé, on conclut que l'ensemble des nœuds défilés distincts est de cardinal strictement croissant et majoré par M au fur et à mesure des appels de la boucle. Il est donc nécessaire qu'il y ait un nombre fini de parcours de la boucle. On a montré que la file se vide finit par se vider en temps fini. ■

Preuve de correction ?

Cet algorithme est déjà une heuristique : il n'explore qu'une partie des trajectoires possibles. La raison est que dans le cas où la mémoire GPU est limitée trop occupée pour passer à l'étape suivante, on la vide complètement. Ceci n'est pas forcément toujours nécessaire. Mais modéliser tous les sous-ensemble de variables rapatriées sur la mémoire principale à chaque fois que ce blocage intervient ferait exploser la complexité combinatoire du problème (rajoutant potentiellement $2^{\neg \mathbf{B}[:,s]}$ fils au nœud en train d'être traité au lieu d'un seul avec l'algorithme), sous la contrainte que la mémoire libérée doit suffire à stocker $\mathbf{B}[:,s]$.

Par conséquent l'algorithme ne renvoie pas nécessairement une des trajectoires optimales possibles au sens des coûts.

Majoration optimale du nombre de nœuds

En notant

$$k(s) = \#\{v \in [\![1,V]\!], [\mathbf{B}|\mathbf{F}][v, s+1:I] \neq \mathbf{0}^{I-s}\} \tag{D.1}$$

$k(i)$ est le nombre de variables que l'on va réutiliser ultérieurement à la fin de l'instruction/étape i. On a la majoration

$$\#\{N \in V, \mathrm{prof}(N) = s\} \leq 2^{\max(k(s),s)} \tag{D.2}$$

Sans plus de contrainte, cette majoration est optimale avec cette modélisation. En effet, si l'on choisit $\mathcal{B} = \mathbb{I}$ (matrice identité) avec $I = V$ et $\mathbf{F} = (1,1,1,\cdots,1)^\intercal$, et $\mathcal{M} \geq \sum_{v=1}^{V} \mathbf{M}[v]$. Ce calcul n'a pas beaucoup d'intérêt mais il est facile de voir que l'arbre binaire associé est complet, avec l'ensemble des nœuds de profondeur I en bijection avec $\{0,1\}^I$ de cardinal 2^I.

Complexité du parcours

Pour trouver le plus court chemin entre le premier nœud représentant l'état initial 0, et le dernier représentant l'état final $I+1$, un algorithme naturel et efficace est d'utiliser celui de Dijkstra, dont la complexité pour un graphe orienté pondéré à poids **positifs** est $\mathcal{O}(E + V \log V)$ soit $\mathcal{O}(I2^I)$ puisque $V \leq 2^I$ et $E = \mathcal{O}(V)$.

On peut théoriquement mieux faire : Si l'on déplie le graphe en structure chaînée comme arbre binaire : tout arbre s'écrivant $A = (R, (A_1, c_1), (A_2, c_2))$ (racine, sous-arbre gauche et le poids de l'arête le reliant à la racine R, et de même pour le sous-arbre droit). Nous choisissons qu'une feuille est du type (F, c_f) pour une raison que je détaille ci-après.

Alors nous avons $G = (N, (G_c, c_c), (G_g, c_g))$ (Nœud, sous-arbre gauche ou CPU, sous-arbre droit chacun avec les coûts correspondant à leurs arêtes respectives), dont les feuilles (nœuds d'arêt) sont du type (F, c_f) affectées d'un coût final (correspond aux arêtes liant les nœuds de profondeur I et le nœud jaune dans la figure D.2).

On peut alors facilement trouver le chemin le plus court en $\mathcal{O}(2^I)$. En effet, la fonction T « longueur du chemin le plus court jusqu'au dernier nœud du graphe » poids minimal pour $T(G)$ s'écrit récursivement

$$T(G) = \min(c_g + T(G_g), c_c + T(G_c)) \tag{D.3}$$

Soit, en notant v le nombre de sommets d'un arbre binaire et C la fonction coût d'un appel à T :

$$C(v) = 2C\left(\frac{v}{2}\right) + \mathcal{O}(1) \tag{D.4}$$

Cette relation est bien connue et c'est un cas couvert par le Master Theorem et montre que la complexité de T est linéaire avec le nombre de sommets V soit $\mathcal{O}(2^I)$, ce qui constitue une amélioration du parcours de Dijikstra.

Un exemple : la fonction $\mathbb{R}^{N^2} \times \mathbb{R}^{N^2} \times \mathbb{R}^N \times \mathbb{R}^N : (R, S, u, v) \mapsto (u^\intercal RS(u+V), Sv)$

Considérons le cas de l'algorithme suivant avec $n = 10$, $k_{\max} = 3$, $\mathcal{M} = 2000$.

Algorithm 4 Calcul spécimen sur matrices et vecteurs

Input : R, S, u, v
Output : Les résultats des calculs matriciels $v^\intercal RS(u+v)$ et Sv
1 : $R \leftarrow RS$ ▷ n^3 produits de scalaires
2 : $u[1] \leftarrow u^\intercal R(u+v)$ ▷ $n^2 + n$ produits de scalaires
3 : $v \leftarrow Sv$ ▷ n^2 produits de scalaires
return u, v

On définit les matrices et vecteurs précédemment listés qui caractérisent l'algorithme

$$\mathbf{B}|\mathbf{F} = \begin{array}{c} \\ R: \\ S: \\ u: \\ v: \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{B}_3 & \mathbf{F} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array}, \qquad \mathbf{D} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{D.5}$$

(car au début, les 4 variables d'entrée sont données)

$$\mathbf{K} = \begin{array}{c} \\ \text{ordre } 0 \\ \text{ordre } 1 \\ \text{ordre } 2 \\ \text{ordre } 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \tag{D.6}$$

$$\mathbf{\Omega} = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} n^0 \\ n^1 \\ n^2 \\ n^3 \end{bmatrix}, \qquad \mathbf{\Xi}_{\text{CPU}} = \begin{bmatrix} n^0 \\ n^1 \\ n^2 \\ n^3 \end{bmatrix}, \qquad \mathbf{\Xi}_{\text{GPU}} = \begin{bmatrix} 1.0 \\ n^{1-1} \\ n^{2-1} \\ n^{3-1} \end{bmatrix} \tag{D.7}$$

(en faisant l'hypothèse que le GPU prarallélise toujours suivant une dimension)

Sur cet exemple, la résolution conduit à créer le graphe D.2, qui montre que la trajectoire optimale est de l'effectuer sur le GPU en entier, principalement car toutes les variables utiles à tout instant donné occupent une place mémoire inférieure ou égale à la limite du GPU. Avec des exemples plus compliqués, on peut générer des gros graphes ou le GPU devient parfois prohibitif.

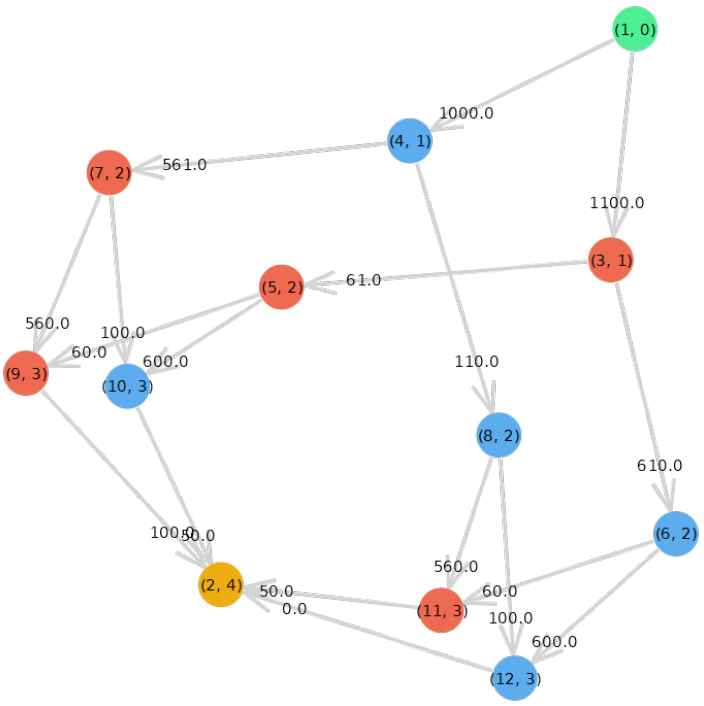


Figure D.2 : Graphe acyclique pour l'algorithme 4 avec les paramètres indiqués et $\mathbb{R}^{N^2} \times \mathbb{R}^{N^2} \times \mathbb{R}^N \times \mathbb{R}^N : (R, S, u, v) \mapsto (u^\intercal RS(u+v), Sv)$ (cas d'école). Chaque nœud est ici identifié par un couple (`identifiant`, `profondeur`) et représente un état du calcul. Sur une ligne de profondeur constante les différents nœuds représentent les différents scénarios d'états d'occupation de la mémoire. Les arêtes du graphe correspondent aux coûts de calcul et transfert mémoire pour passer d'un état à un autre. L'identifiant correspond à l'ordre d'insertion dans le graphe. Nous avons trois instructions donc bien une profondeur égale à 3. Ici, on voit seulement 4 nœuds de profondeur 3, chacun d'entre eux étant commun à 2 des $2^3 = 8$ scénarios de calcul. Les nœuds rouges correspondent au GPU et les bleus, au CPU.

D.2 Contrôle de l'erreur de bout en bout

Je me place dans ce paragraphe dans le cadre où l'on dispose d'un potentiel pour lequel on cherche l'énergie propre et la fonction d'onde de l'état fondamental non dégénéré, et j'essaie de donner des résultats pour majorer systématiquement tous les types d'erreurs commis (arrondi, troncature, discrétisation). Je me place ici dans l'hypothèse où le pas de discrétisation est le même selon les deux coordonnées généralisées ou deux directions d'espace, et où seule l'« ouverture »[1] de la grille, son étendue sur $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, est adaptable. La raison est que fixer h (pas d'espace) permet de connaître les valeurs propres, et donc le conditionnement, de la matrice du laplacien 2D en différences finies correspondant ce qui simplifie la majoration.

Dans ce qui suit, je note $\Psi^s_{(i,j)}$ la matrice de Ψ^s échantillonné sur la grille des (x_i, y_j), et $\psi_{(i,j)}$ et $\psi_{(i,j)}$ les matrice définies comme ci-dessus. De même je note $(\Delta\Psi^s)_{(i,j)}$ le laplacien de Ψ^s échantillonné sur la grille. Je note pareillement l'échantillonnage du

[1] Pour emprunter le terme à l'optique, par abus.

potentiel (connu) $V \equiv \mathcal{V}(x_i, y_j)$ qui est une matrice diagonale, de même taille que $\Lambda^{\mathbf{2D}}$, et qui représente sur la grille l'opérateur $\hat{\mathcal{V}}$. Ainsi Ψ^s désigne la fonction et $\Psi^s_{(i,j)}$ désigne la matrice des échantillons sur la grille. Notons qu'avec ces notations nous avons donc l'égalité vectorielle

$$-\frac{\hbar^2}{2m}(\Delta\Psi^s)_{(i,j)} + V\Psi^s_{(i,j)} = \mathcal{E}\Psi^s_{(i,j)} \tag{D.8}$$

qui est juste un échantillonnage sur notre grille de la version continue de l'égalité. Je m'en servirai après.

Puis je définis

- $\Psi^s \in L^2(\mathbb{R}^2, \mathbb{R})$ la fonction d'onde solution réelle du problème aux valeurs propres continu vérifiant bien-sûr $-\frac{\hbar^2}{2m}\Delta\Psi^s + \hat{\mathcal{V}}\Psi^s = \hat{\mathcal{E}}\Psi^s$ sur tout l'espace avec $\mathcal{E}$ l'énergie propre du fondamental (inaccessible analytiquement a priori) ;
- $\psi_{(i,j)} \in \mathbb{R}^{N\times N}$ la solution au problème matriciel discrétisé $-\frac{\hbar^2}{2m}\Lambda^{\mathrm{2D}}\psi_{(i,j)} + V\psi_{(i,j)} = E\psi_{(i,j)}$ du laplacien sur la grille en différences finies (problème aux valeurs propres) avec l'énergie fondamentale E (supposément proche de $\mathcal{E}$) ;

Il est à noter qu'en réalité, on calcule numériquement une approximation de E et une approximation de ψ, puisqu'on est en grande dimension, on a recours à des méthodes itératives très certainement, cf. les sections précédentes du rapport.

Puis on cherche à estimer naturellement l'erreur

$$\|\psi_{(i,j)} - \Psi^s_{(i,j)}\| \tag{D.9}$$

avec le sens de cette norme à préciser plus tard (en dimension finie, toutes les normes sont équivalentes donc on pourra passer de l'une à l'autre aisément, je passe de la norme infinie qui représente l'écart maximal, à la norme 2 ensuite).

On va décomposer cette erreur en plusieurs termes pour pouvoir la majorer. Le fait de discrétiser le problème va permettre de trouver une énergie fondamentale approchée E qui comme on l'a dit, est espérer proche de l'énergie du fondamental analytiquement exacte $\mathcal{E}$.

D.2.1 Erreur de discrétisation sur le vecteur

Première étape : majorer l'erreur faite sur le laplacien En utilisant le théorème 1.1.6, on sait que la fonction recherchée est à décroissance rapide au voisinage de $\|(x,y)\| \to +\infty$, et que $\forall\varepsilon > 0$, $\exists c > 0$, $\forall x$, $|\psi(x)| \leq D\exp(-\frac{1}{2}(c+\varepsilon)^{1/2}|x|^2)$ pour certains paramètres D, c, ε. On dénote donc par $\gamma > 0$ une fonction dépendant de la taille de la grille (étendue selon $x_{\max}$ en supposant que $x_{\max} \sim y_{\max}$) telle que, en dehors de la grille, $|\psi(x,y)| \leq \gamma$. On retient ainsi que $\gamma = \mathcal{O}(e^{-x^2_{\max}})$. On suppose raisonnablement que Ψ^s est de classe $\mathscr{C}^\infty$ et à décroissance rapide et que ses dérivées sont bornées. Par conséquent, l'approximation du laplacien à 5 points, si elle prenait même les points extérieurs à la grille aux points des frontières en dénotant $(\mathcal{L}\Psi^s)_{(i,j)}$, donnerait une erreur en norme ∞ telle que

$$\|(\Delta\Psi^s)_{(i,j)} - (\mathcal{L}\Psi^s)_{(i,j)}\|_\infty \leq \frac{h^2}{12}\left(\sup_{\mathbb{R}}\left|\frac{\partial^4\Psi^s}{\partial x^4}\right| + \sup_{\mathbb{R}}\left|\frac{\partial^4\Psi^s}{\partial y^4}\right|\right) \tag{D.10}$$

Mais la matrice du laplacien Λ^{2D} ne prend pas en compte les points extérieurs à la grille. Sur ces points Ψ^s prend des valeurs inférieures ou égales à γ, donc on va majorer en décomposant en

deux termes

$$\|\Lambda\Psi^s_{(i,j)} - (\Delta\Psi^s)_{(i,j)}\|_2 \leq \underbrace{\|\Lambda\Psi^s_{(i,j)} - (\mathcal{L}\Psi^s)_{(i,j)}\|_2}_{①} + \underbrace{\|(\mathcal{L}\Psi^s)_{(i,j)} - (\Delta\Psi^s)_{(i,j)}\|_2}_{②} \tag{D.11}$$

Le premier terme est l'erreur d'approximation du laplacien à 5 points, qui tend vers 0 lorsqu'on raffine le pas de discrétisation ; le second terme est l'erreur qu'on commet en ne prenant pas les points en dehors de la grille pour le laplacien des frontières.

On sait que $\|\cdot\|_2 \leq \sqrt{\dim}\|\cdot\|_\infty$, et ici dim $= N^2$ (puisque nous avons discrétisé sur N points de grille sur chaque axe en 2D).

D'une part, le résultat connu sur la convergence du laplacien à 5 points D.10 permet donc d'écrire en notant

$$① \leq N \cdot \|\Lambda\Psi^s_{(i,j)} - (\mathcal{L}\Psi^s)_{(i,j)}\|_\infty \leq \frac{Nh^2}{12}\left(\sup_{\mathbb{R}}\left|\frac{\partial^4\Psi^s}{\partial x^4}\right| + \sup_{\mathbb{R}}\left|\frac{\partial^4\Psi^s}{\partial y^4}\right|\right) \tag{D.12}$$

et je note dorénavant $S = \sup_{\mathbb{R}}\left|\frac{\partial^4\Psi^s}{\partial x^4}\right| + \sup_{\mathbb{R}}\left|\frac{\partial^4\Psi^s}{\partial y^4}\right|$.

D'autre part, l'erreur que l'on commet en ne prenant pas en compte les valeurs de Ψ^s en dehors de la grille concerne $4N$ points, tous sur lesquels Ψ^s prend des valeurs proches de 0 à γ près, c'est-à-dire qu'on peut en calculer un majorant en norme 2 en sommant le carré de ces erreurs. Chaque erreur individuelle vaut au plus $\frac{\gamma}{h^2}$ en effet, puisque le poids d'un point extracentré dans le laplacien à 5 points est de $\frac{1}{h^2}$:

$$② \leq \sqrt{4N\left(\frac{\gamma}{h^2}\right)^2} = 2\sqrt{N}\frac{\gamma}{h^2} \tag{D.13}$$

On conclut en réinjectant dans D.11 :

$$\boxed{\|\Lambda\Psi^s_{(i,j)} - (\Delta\Psi^s)_{(i,j)}\|_2 \leq \underbrace{\frac{Nh^2S}{12} + \frac{2\gamma\sqrt{N}}{h^2}}_{:=g}} \tag{D.14}$$

Seconde étape : erreur sur le vecteur de grille On écrit ensuite comme des perturbations les éléments numériques $\psi_{(i,j)} = \Psi^s_{(i,j)} + \mathbf{r}$ et $E = \mathcal{E} + e$, donc on part de l'égalité solution du problème discret

$$-\frac{\hbar^2}{2m}\Lambda\psi_{(i,j)} + V\psi_{(i,j)} = E\psi_{(i,j)} \tag{D.15}$$

que l'on réécrit

$$-\frac{\hbar^2}{2m}\Lambda(\Psi^s_{(i,j)} + \mathbf{r}) + V(\Psi^s_{(i,j)} + \mathbf{r}) = (\mathcal{E} + e)(\Psi^s_{(i,j)} + \mathbf{r}) \tag{D.16}$$

et réordonne ensuite, avec $E = \mathcal{E} + e$:

$$\underbrace{-\frac{\hbar^2}{2m}\Lambda\Psi^s_{(i,j)} + (V - \mathcal{E})\Psi^s_{(i,j)}}_{③}\ \underbrace{-e\Psi^s_{(i,j)}}_{④} = -\left(-\frac{\hbar^2}{2m}\Lambda + V - E\right)\mathbf{r} \tag{D.17}$$

Cette équation D.17 est une relation que vérifie l'erreur $\mathbf{r}$ en fonction de l'erreur que l'on fait en approchant le hamiltonien (en fait, la partie du hamiltonien qui corespond au laplacien) et

l'énergie. $\mathbf{r}$ est une quantité vectorielle inaccessible, car ψ^s lui-même est inaccessible : c'est la fonction d'onde qu'on tente d'approcher. La matrice devant $\mathbf{r}$ dans le terme de droite est singulière par construction, mais son rang est précisément $\mathbf{dim} - 1$ (i.e. $N^2 - 1$ si l'on est en 2D avec une discrétisation en N points suivant chaque dimension, et $N - 1$ si l'on est en 1D). A priori, rien ne permet de dire que $\mathbf{r} \in \psi_{(i,j)}^{\perp}$.

Or, comme on a l'égalité D.8, on reconnaît dans (3) une quantité presque nulle d'après l'étape précédente. En effet puisque l'évaluation de $\mathcal{V}$ sur la grille V est exacte (c'est seulement un échantillonnage), et de même pour $\mathcal{E}$ en E, l'écart à 0 est seulement dû à l'erreur sur le laplacien en différences finies, majoré dans l'équation D.14. Ainsi :

$$\left\| (3) \right\|_2 \leq \frac{\hbar^2}{2m} g \tag{D.18}$$

et en outre, $\Psi^s_{(i,j)} \leq D$ puisque Ψ^s est majoré par D :

$$\left\| (4) \right\|_2 \leq eD \tag{D.19}$$

donc le membre de gauche de D.17 est de norme 2 contrôlée par

$$\left\| -\frac{\hbar^2}{2m} \Lambda \Psi^s_{(i,j)} + (V - \mathcal{E}I)\Psi^s - e\Psi^s_{(i,j)} \right\|_2 \leq \frac{\hbar^2 g}{2m} + eD \tag{D.20}$$

À partir de là, on ne peut pas facilement déduire une majoration sur $\mathbf{r}$ car on ne connait pas a priori sa composante sur la direction singulière (droite engendrée par $\psi_{(i,j)}$) de $-\frac{\hbar^2}{2m}\Lambda + V - E$, qui rend cette matrice singulière.

D.2.2 Erreur de troncature sur l'énergie

Dans [RS80b], on peut lire à la fin du chapitre XII, qu'en dimension finie avec les notations précédentes, c'est-à-dire un hamiltonien H_0 qu'on perturbe avec W et en notant (ϕ_i) (qu'on indice de 1 à N par convention) les vecteurs propres de H_0 associés aux énergies E_i, alors l'opérateur

$$P(\lambda) = -\frac{1}{2\mathrm{i}\pi} \int_{|E-E_0|=\varepsilon} (H_0 + \lambda W - E)^{-1} \mathrm{d}E \tag{D.21}$$

est la projection sur le vecteur propre de valeur propre $E(\lambda)$ du hamiltonien perturbé $H_0 + \lambda W$. On peut donc tenter d'évaluer numériquement ce projecteur pour obtenir ensuite l'énergie perturbée.

L'ouvrage donne ensuite les formules pour la perturbation λW :

$$E = E_0 + \lambda \frac{\sum_{n=0}^{+\infty} a_n \lambda^n}{\sum_{n=0}^{+\infty} b_n \lambda^n} \tag{D.22}$$

avec en fait $E_0 = E_1$ (c'est la plus petite valeur propre, celle de l'état fondamental) et

$$a_n = -\frac{(-1)^{n+1}}{2\mathrm{i}\pi} \int_{|E-E_0|=\varepsilon} \langle \phi_0 | \qquad V(H_0 - E)^{-(n+1)} |\phi_0\rangle \, \mathrm{d}E \tag{D.23}$$

$$b_n = -\frac{(-1)^{n+1}}{2\mathrm{i}\pi} \int_{|E-E_0|=\varepsilon} \langle \phi_0 | \qquad (H_0 - E)^{-1} [V(H_0 - E)^{-1}]^n |\phi_0\rangle \, \mathrm{d}E \tag{D.24}$$

Si on parvient à calculer suffisamment de coefficients, on peut donc obtenir un approximant de PADÉ de l'énergie. La difficulté réside dans l'inverse, et dans l'intégrale bien-sûr. Puisque tous les facteurs sauf les inverses sont des constantes, il s'agit de calculer les intégrales

$$-\frac{(-1)^{n+1}}{2\pi}\int_0^{2\pi} V(H_0-(E_0+\varepsilon e^{\mathrm{i}t})I)^{-(n+1)}\,\mathrm{d}t \tag{D.25}$$

et

$$-\frac{(-1)^{n+1}}{2\pi}\int_0^{2\pi} (H_0-(E_0+\varepsilon e^{\mathrm{i}t}))^{-1}[V(H_0-(E_0+\varepsilon e^{\mathrm{i}t})I)^{-1}]^n\,\mathrm{d}t \tag{D.26}$$

L'ouvrage donne les calculs des premiers termes et précise que $b_0 = 1$ et $b_1 = 0$, or comme la seconde intégrale est considérablement moins évidente à calculer, je me concentre dans un premier temps sur la première.

Puisque H_0 est symétrique réelle, elle est diagonalisable en base orthonormée. On nomme P sa matrice de passage dans cette base. D'autre part, $(E_0+\varepsilon e^{it})I$ est diagonale, c'est une homotétie. Donc $\mathrm{Sp}(H_0-(E_0+\varepsilon e^{\mathrm{i}t})I) = \mathrm{Sp}(H_0)-E_0-\varepsilon e^{\mathrm{i}t} = \{E_1-E_0-\varepsilon e^{\mathrm{i}t}, E_2-E_0-\varepsilon e^{\mathrm{i}t},\cdots,E_N-E_0-\varepsilon e^{\mathrm{i}t}\}$. Notons $D=\mathrm{diag}(E_1,\cdots,E_N)$. Alors toujours en notant $E=E_0+\varepsilon e^{\mathrm{i}t}$, on a pour tout n entier naturel : on a

$$(H_0-E)^{-n} = (PDP^{-1}-E)^{-1} = P(D-E)^{-n}P^{-1} \tag{D.27}$$

$D-E$ est facile à inverser :

$$(D-E)^{-1} = \mathrm{diag}\left(\frac{1}{E_i-E_0-\varepsilon e^{\mathrm{i}t}}\right); \qquad (D-E)^{-n} = \mathrm{diag}\left(\frac{1}{(E_i-E_0-\varepsilon e^{\mathrm{i}t})^n}\right) \tag{D.28}$$

et on peut aussi facilement calculer l'intégrale sur t de $(D-E)^{-n}$ avec la formule de CAUCHY :

$$\int_0^{2\pi}\frac{1}{(E_i-E_0-\varepsilon e^{\mathrm{i}t})^n}\mathrm{d}t = \int_0^{2\pi}\frac{1}{(E_i-E_0-\varepsilon e^{\mathrm{i}t})^n}\mathrm{d}t \tag{D.29}$$

Bibliographie

[All12] Grégoire Allaire. *Analyse numérique et optimisation : une introduction à la modélisation mathématique et à la simulation numérique*. 2012. OCLC : 1040982058.

[BH06] M. Born and W. Heisenberg. Zur Quantentheorie der Molekeln. *Annalen der Physik*, 379 :1–31, March 2006.

[BO27] M. Born and R. Oppenheimer. Zur Quantentheorie der Molekeln. *Annalen der Physik*, 389(20) :457–484, 1927. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.19273892002.

[CLBL21] Eric Cances, Antoine Levitt, Laurent Brochard, and Anaël Lemaitre. Introduction to Quantum Mechanics, 2021.

[CTDL21a] Claude Cohen-Tannoudji, Bernard Diu, and Franck Laloë. *Mécanique Quantique - Tome 1 : Nouvelle édition*. 2021. OCLC : 1245157654.

[CTDL21b] Claude Cohen-Tannoudji, Bernard Diu, and Franck Laloë. *Mécanique Quantique - Tome 2 : Nouvelle édition*. 2021. OCLC : 1245042806.

[DF29] Paul Adrien Maurice Dirac and Ralph Howard Fowler. Quantum mechanics of many-electron systems. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 123(792) :714–733, April 1929. Publisher : Royal Society.

[Dus17] Geneviève Dusson. *Error estimation for linear and nonlinear eigenvalue problems arising from electronic structure calculation*. phdthesis, Université Pierre et Marie Curie - Paris VI, October 2017.

[Gir18] Alexandre Girouard, editor. *Spectral Theory and Applications*, volume 720 of *Contemporary Mathematics*. American Mathematical Society, Providence, Rhode Island, November 2018.

[Kat95] Toshio Kato. *Perturbation theory for linear operators*. Springer, Berlin ; Heidelberg, 1995. OCLC : 468302662.

[Pol87] Boris Teodorovich Poliak. *Introduction to optimization*. Optimization Software, New York, 1987. OCLC : 924709289.

[RS75] Michael Reed and Barry Simon. *Methods of modern mathematical physics. Vol. 2, Fourier analysis, self-adjointness*. Academic Press, New York, 1975. OCLC : 16565513.

[RS79] Michael Reed and Barry Simon. *Methods of modern mathematical physics. Vol. 3, Scattering theory*. Academic Press, New York ; London, 1979. OCLC : 1051306803.

[RS80a] Michael Reed and Barry Simon. *Methods of modern mathematical physics. Vol. 1, Functional analysis*. Academic Press, New York, 1980. OCLC : 1120894971.

[RS80b] Michael Reed and Barry Simon. *Methods of modern mathematical physics. Vol. 4, Analysis of operators*. Academic Press, San Diego [etc., 1980. OCLC : 490240607.

[Saa11] Y Saad. *Numerical methods for large eigenvalue problems*. Society for Industrial and Applied Mathematics, Philadelphia, 2011. OCLC : 707725938.

[Sim75] Barry Simon. Pointwise Bounds on Eigenfunctions and Wave Packets in N-Body Quantum Systems. III. *Transactions of the American Mathematical Society*, 208 :317–329, 1975. Publisher : American Mathematical Society.

[Tul00] John C. Tully. Perspective on "Zur Quantentheorie der Molekeln". *Theoretical Chemistry Accounts*, 103(3) :173–176, February 2000.

[Zha99] John Zeng Hui Zhang. *Theory and application of quantum molecular dynamics*. World Scientific, Farrer Road, Singapore ; River Edge, New Jersey, 1999. OCLC : 981439654.