UFRRJ

INSTITUTO DE CIÊNCIAS EXATAS
CURSO DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

DISSERTAÇÃO

Implementação Computacional do Método da Matriz Densidade Tight-Binding para Cristais de Silício através de Algoritmos de Gradiente Conjugado Não Linear e do Hamiltoniano de Kwon

Fernanda Lúcia Sá Ferreira

2017

INSTITUTO DE CIÊNCIAS EXATAS
CURSO DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

IMPLEMENTAÇÃO COMPUTACIONAL DO MÉTODO DA MATRIZ DENSIDADE TIGHT-BINDING PARA CRISTAIS DE SILÍCIO ATRAVÉS DE ALGORITMOS DE GRADIENTE CONJUGADO NÃO LINEAR E DO HAMILTONIANO DE KWON

FERNANDA LÚCIA SÁ FERREIRA

Sob a Orientação do Professor
Moisés Augusto da Silva Monteiro de Araújo

Dissertação submetida como requisito parcial para obtenção do grau de **Mestre em Ciências**, no Curso de Pós-Graduação em Modelagem Matemática e Computacional, Área de Concentração em Modelagem Matemática e Computacional

Seropédica, RJ
Março de 2017

Universidade Federal Rural do Rio de Janeiro
Biblioteca Central / Seção de Processamento Técnico

Ficha catalográfica elaborada
com os dados fornecidos pelo(a) autor(a)

SA FERREIRA, FERNANDA LUCIA, 1981-
S383i Implementação Computacional do Método da Matriz Densidade Tight-Binding para Cristais de Silício através de Algoritmos de Gradiente Conjugado Não Linear e do Hamiltoniano de Kwon / FERNANDA LUCIA SA FERREIRA. - 2017.
84 f.

Orientador: Moisés Augusto da Silva Monteiro de Araújo. Dissertação(Mestrado). -- Universidade Federal Rural do Rio de Janeiro, Programa de Pós Graduação em Modelagem Matemática e Computacional, 2017.

1. Tight-Binding. 2. Método da Matriz Densidade Tight-Binding - DMTB. 3. Gradiente Conjugado Não Linear - NLCG. I. da Silva Monteiro de Araújo, Moisés Augusto , 1974-, orient. II Universidade Federal Rural do Rio de Janeiro. Programa de Pós-Graduação em Modelagem Matemática e Computacional III. Título.

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO DE CIÊNCIAS EXATAS
CURSO DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

FERNANDA LÚCIA SÁ FERREIRA

Dissertação submetida como requisito parcial para obtenção do grau de **Mestre em Ciências**, no Curso de Pós-Graduação em Modelagem Matemática e Computacional, área de Concentração em Modelagem Matemática e Computacional.

DISSERTAÇÃO APROVADA EM 31/03/2017

Moisés Augusto da Silva Monteiro de Araújo (Dr.) UFRRJ
(Orientador)

Duílio Tadeu da Conceição Junior (Dr.) UFRRJ

Welfington Wallace Miguel Melo (Dr.) CEFET-RJ

Ao pequeno João Danilo.

"Embora ninguém possa voltar atrás e fazer um novo começo, qualquer um pode começar agora e fazer um novo fim."

Chico Xavier

Resumo

FERREIRA, Fernanda Lúcia Sá. **Implementação Computacional do Método da Matriz Densidade Tight-Binding para Cristais de Silício através de Algoritmos de Gradiente Conjugado Não Linear e do Hamiltoniano de Kwon.** 2017 75p Dissertação (Mestrado em Modelagem Matemática e Computacional). Instituto de Ciências Exatas, Universidade Federal Rural do Rio de Janeiro, Seropédica, RJ,2017.

Para se investigar propriedades eletrônicas e estruturais de sistemas cristalinos são usados os cálculos por primeiros princípios e metodologias semi-empíricas como o *tight-binding*. A metodologia *tight-binding* está presente em diversos simuladores computacionais, porém envolve um processo de diagonalização de matrizes, o que pode requerer complexidade computacional (de pior caso) $O(N^3)$, onde N é o número de átomos no sistema cristalino em questão. Nesse trabalho apresentamos o método da matriz densidade *tight-binding* - DMTB, desenvolvido em [Li et al., 1993]. Esse método tem potencial para ter um custo computacional mais baixo, o que permite que sejam tratados sistemas com milhares de átomos. Nessa dissertação apresentamos o cálculo de energia mínima para cristais de silício, via DMTB. Para esse fim, fizemos algumas modificações no método DMTB, tornado-o compatível com ideias apresentadas em [Millam and Scuseria, 1997], além de também apresentarmos a parametrização feita por Kwon [Kwon et al., 1994] para o hamiltoniano tight-binding cristalino do silício. Para a mininização presente no método DMTB, usamos métodos do tipo Gradiente Conjugado Não Linear, sobre os quais dedicamos um capítulo dessa dissertação. O resultado final desse trabalho foi um algoritmo computacional em C++, sob orientação a objeto, para o cálculo da energia mínima do Silício. Diversos testes foram feitos para a validação do mesmo, e os resultados obtidos estão coerentes com os presentes na literatura.

Palavras-chaves: Tight-Binding, Método da Matriz Densidade *Tight-Binding* - DMTB, Gradiente Conjugado Não Linear - NLCG

Abstract

FERREIRA, Fernanda Lúcia Sá. **Computational Implementation of the Matrix Density Tight-Binding Method for Silicon Crystals by Nonlinear Conjugate Gradient Algorithms and Kwon Hamiltonian.** 2017 75p Dissertation (Master Science in Mathematical and Computational Modeling). Instituto de Ciências Exatas, Universidade Federal Rural do Rio de Janeiro, Seropédica, RJ,2017.

In order to investigate electronic and structural properties of crystalline systems, calculations by first principles and semi-empirical methodologies such as tight-binding are used. The tight-binding methodology is present in several computational simulators, but involves a process of matrix diagonalization which may require computational complexity (the worst case) $O(N^3)$ where N is the number of atoms in the crystalline system in question. In this work, we present the density-tight matrix method - DMTB developed in [Li et al., 1993]. This method has potential to have a lower computational cost which allows systems with thousands of atoms to be treated. In this dissertation, we present the calculation of minimum energy for silicon crystals via DMTB. For this purpose, some modifications were done in the DMTB method, making it compatible with the ideas presented in [Millam and Scuseria, 1997]. Besides this, the parameterization made by Kwon [Kwon et al., 1994] for the crystalline tight-binding hamiltonian of silicon was presented. For the existing mininization in the DMTB method, we used methods based on the Nonlinear Conjugate Gradient. A chapter of this dissertation was dedicated to them. The final result of this work was a computational algorithm in C ++, under object orientation, to calculate the minimum energy silicon. Several tests were done for its validation and the results obtained are consistent with those presented in the literature.

Key-words: Tight-Binding, Density Matrix *Tight-Binding* - DMTB, Nonlinear Conjugate Gradient - NLCG

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Símbolos

Ψ	Letra grega maiúscula Psi
Φ	Letra grega maiúscula Phi
ψ	Letra grega minúscula psi
φ	Letra grega minúscula phi
ϵ	Letra grega minúscula epsilon
ρ	Letra grega minúscula rho
μ	Letra grega minúscula mu
α	Letra grega minúscula alpha
θ	Letra grega minúscula theta
λ	Letra grega minúscula lambda
$\mathcal{M}_{n\times n}$	Notação para espaço vetorial das matrizes $n \times n$
$*$	Notação para a operação conjugação
^{T}	Notação para a operação transposição de matrizes
$\cdot$	Notação para a operação produto escalar
eV	Notação para a unidade de medida Elétrons-Volt

Lista de Siglas

TB	*Método tight-binding*
DMTB	*Método da matriz densidade tight-binding*
DFT	*Teoria do Funcional de Densidade*
CG	*Gradiente Conjugado*
NLCG	*Gradiente Conjugado Não Linear*

Sumário

Introdução

A comunidade científica tem demonstrado interesse crescente nas últimas décadas em materiais com escalas entre 1 e 100 nanômetros (nm), usualmente chamados de materiais nanoestruturados. Esse crescimento nas pesquisas ocorre principalmente devido ao grande número de aplicações, que vão desde a eletrônica [Hammes, 2011], passando pela engenharia de materiais [Nero, 1999] até aplicações em biologia molecular [Frauenheim et al., 2000].

Para se investigar propriedades eletrônicas e estruturais de sistemas cristalinos, são usados os cálculos por primeiros princípios e metodologias semi-empíricas como o *tight-binding*. (O livro [Dorsett et al., 2000] apresenta um tratado sobre os métodos de primeiros princípios e [Freed, 1995] constrói uma ponte entre as metodologias de primeiro princípios e as semi-empíricas). Um exemplo de uso desses cálculos está no estudo teórico dos defeitos cristalinos. Defeitos em cristais alteram várias propriedades eletrônicas e estruturais dos materiais conforme cita [Padilha, 1997]. Por outro lado, conhecendo tais propriedades, é possível se investigar quais tipos de defeitos o cristal possuí. Um tipo particular de defeito que recebe muita atenção na área de Física da Matéria Condensada é o defeito linear. Esses defeitos são conhecidos como as discordâncias cristalinas (para detalhes veja [Araújo, 2006] e [Oliveira, 2014]). O deslocamento de discordâncias, sob certas condições, é responsável por um processo chamado *encruamento*, onde um material dúctil se torna ainda mais duro e resistente. Este é um dos motivos que coloca o estudo dos defeitos em cristais como ponto de grande interesse da Ciência dos Materiais.

A simulação computacional é uma ferramenta bastante eficiente para o estudo de novos materiais nanoestruturados. Como os materiais de interesse possuem escalas entre 1 nm e 100 nm, as simulações se baseiam em programas computacionais escritos para resolver as equações quanto-mecânicas que os modelam, e assim, determinar propriedades energéticas, estruturais e eletrônicas dos materiais. A metodologia *tight-binding* está presente em diversos simuladores computacionais, porém envolve um processo de di-

agonalização de matrizes, o que pode requerer complexidade computacional (de pior caso) $O(N^3)$, onde N é o número de átomos no sistema cristalino em questão (maiores detalhes veja em [Ordejón, 1998]). Se esse número for muito alto, o uso da metodologia *tight-binding* está comprometido, já que o custo computacional não o tornará viável e isso nos impede de fazer simulações mais realistas.

[Li et al., 1993], desenvolveram o método da matriz densidade *tight-binding* - DMTB. No mesmo ano, usando técnicas diferentes, [Daw, 1993] obteve um método bastante similar a este. O DMTB está diretamente ligado ao tratamento *tight-binding*, onde os estados eletrônicos cristalinos podem ser descritos em termos de orbitais atômicos e o hamiltoniano cristalino pode ser escrito de forma parametrizada. Esse método tem potencial para ter um custo computacional mais baixo, o que permite que sejam tratados sistemas com milhares de átomos. Porém, essa metodologia se limita ao cálculo de energia total no estado fundamental e geometrias de equilíbrio [Araújo, 2006]. Nessa dissertação apresentamos o cálculo de energia mínima para cristais de silício, via DMTB, que é um dos semicondutores de grande importância para o desenvolvimento tecnológico, pois é utilizado na fabricação de microchips e em diversos equipamentos eletrônicos que utilizamos hoje.

Durante esse período do mestrado, foi formado um grupo de estudo com alunos do PPGMMC sob a supervisão do professor orientador desse trabalho. Nesse grupo, nos dedicamos ao estudo de várias vertentes do método DMTB. Em um primeiro momento foi feito um estudo teórico da metodologia e a seguir passamos a fazer uma implementação computacional do DMTB. Na dissertação de mestrado de [Filho, 2017] consta em detalhes a modelagem físico-matemática do DMTB e também uma apresentação da mecânica quântica necessária para se entender o método. A modelagem apresentada não é mais a original, mas sim uma versão desenvolvida por nosso grupo de estudo.

Nessa dissertação o foco é apresentar, de uma maneira autocontida e compreensível, uma implementação computacional para o DMTB e também apresentar as ferramentas matemáticas inerentes a implementação apresentada. Sendo assim, a dissertação está organizada da seguinte forma:

No primeiro capítulo apresentamos uma coleção de informações necessárias para a compreensão do método DMTB. Começamos falando sobre estruturas cristalinas, passando por uma breve descrição da metodologia *tight-binding* e chegando ao método DMTB. Nesse trabalho não usamos exatamente o modelo DMTB, proposto por [Li et al., 1993]. Fizemos algumas modificações no método, tornado-o compatível com ideias apresentadas em [Millam and Scuseria, 1997]. Nesse capítulo também apresentamos a para-

metrização feita por Kwon [Kwon et al., 1994] para o hamiltoniano tight-binding cristalino do silício.

No segundo capítulo fazemos uma exposição do método do Gradiente Conjugado para minimização de funções. Começamos o capítulo com a abordagem clássica do Gradiente Conjugado Linear e depois passamos para a exposição do Gradiente Conjugado Não-Linear, apresentando as técnicas que serão usadas nesse trabalho. Convém observar que grande parte desse capítulo foi desenvolvida independente do método DMTB, podendo ser usado como referência didática para iniciantes no assunto. Encerramos esse capítulo adaptando um método de Gradiente Conjugado Não Linear para o uso no DMTB, incluindo o processo de vetorização que irá nos permitir aplicar essa metodologia de otimização em uma função originalmente com domínio matricial.

No terceiro capítulo tratamos exatamente da implementação do método do DMTB. O capítulo começa com uma justificativa do porquê decidimos desenvolver nosso simulador em C++ sob o paradigma da orientação à objeto. Posteriormente falamos sobre as bibliotecas externas que utilizamos e detalhamos as classes que desenvolvemos para criar o programa, incluindo as estratégias utilizadas e pseudo-códigos de trechos fundamentais.

E por fim, no quarto capítulo apresentamos diversas propostas de trabalhos futuros que vão na direção de transformar o simulador computacional desenvolvido durante esse estudo de mestrado em um laboratório virtual de propriedades eletrônicas e estruturais em sistemas cristalinos.

Capítulo 1

Fundamentos Teóricos

Resolver equações quanto-mecânicas para se obter propriedades estruturais e eletrônicas de um sistema formado por muitos elétrons não é uma tarefa trivial. Essas equações modelam sistemas sob várias hipóteses simplificadoras (como a aproximação de Born - Oppenheimer e modelos de partículas independentes, por exemplo) e ainda sim, não possuem soluções analíticas. Os cálculos quânticos basicamente são feitos de duas maneiras:

- **Técnicas *ab initio* ou de Primeiros Princípios:** Como o próprio nome diz, os cálculos são feitos "como no início", levando em consideração somente os números atômicos e as suas posições. A Teoria do Funcional de Densidade - DFT, é uma dessas técnicas e é bastante precisa, porém tem complexidade computacional alta, podendo chegar a $O(N^3)$, onde N é o número de elétrons envolvidos, como informado por [Ordejón, 1998].

- **Técnicas semi-empíricas:** São técnicas baseadas em aproximações feitas em técnicas *ab initio*, que envolvem dados experimentais e "intuição química" como diz [Bowler, 1997]. Os problemas quanto-mecânicos tem sua dificuldade bastante reduzida nesse método. Nas técnicas semi-empíricas a precisão dos resultados também é reduzida, porém em troca, ganhamos eficiência computacional. A técnica do *tight-binding* - TB é um exemplo. Outro exemplo é o DMTB, cujo custo computacional é bem mais baixo que na técnica DFT, podendo até mesmo ter complexidade reduzida a $O(N)$, onde N é o número de elétrons envolvidos, como por exemplo nas estratégias apresentadas por [Ordejón, 1998].

O livro [Dorsett et al., 2000] apresenta um tratado bastante detalhado sobre os métodos de primeiros

princípios e [Freed, 1995] apresenta a construção de um elo entre as metodologias de primeiros princípios e as semi-empíricas.

Devido ao custo computacional, quando se trabalha com sistemas de muitos elétrons, os métodos semi-empíricos são os mais indicados. [Li et al., 1993] e [Daw, 1993], desenvolveram metodologias semelhantes para tratar desses problemas. O método proposto por [Li et al., 1993] se chama método da matriz densidade *tight-binding* - DMTB. Este método está diretamente ligado ao tratamento *tight-binding*, onde os estados eletrônicos cristalinos podem ser descritos em termos de orbitais atômicos e o hamiltoniano cristalino pode ser escrito de forma parametrizada. Como vantagem, esse método apresenta possibilidade de implementação com um custo computacional mais baixo, quando comparado ao método *tight-binding*.

O objetivo dessa dissertação não é desenvolver o DMTB, e sim apresentar um algoritmo computacional para a implementação desta metodologia. Especificamente, nesse trabalho usamos o método DMTB para o cálculo da energia mínima de cristais de silício. Nesse capítulo apresentaremos brevemente os conceitos teóricos relacionados ao DMTB necessários para permitir a compreensão do algoritmo apresentado no capítulo 3. Para um exposição mais detalhada sobre o DMTB, na versão específica que trabalhamos aqui e a teoria quântica que o cerca, veja [Filho, 2017].

1.1 Estrutura Cristalina

Um cristal ideal é construído pela repetição regular de blocos elementares idênticos formados um ou mais átomos. A definição precisa de estrutura cristalina é muito importante, pois permite seu tratamento teórico e computacional.

A geometria de um cristal é dada pela *Redes de Bravais*. Segundo [Aguiar Junior, 2006], o cristalógrafo francês Bravais, em 1848, determinou matematicamente, que átomos poderiam ser arranjadas no espaço através de no máximo 14 arranjos, esses arranjos ficaram conhecidos como os 14 sólidos de Bravais. Todos os materiais cristalinos conhecidos até hoje pertencem a um destes 14 arranjos tridimensionais. Os 14 sólidos de Bravais estão ilustrados na figura 1.2.

Uma propriedade importante das redes de Bravais é translação. A rede pode ser representada por um vetor de translação

$$r' = u_1 a_1 + u_2 a_2 + u_3 a_3,$$

onde a_1, a_2 e a_3 são os chamados *vetores primitivos da rede* e u_1, u_2 e u_3 são inteiros que identificam quantas vezes os vetores a_1, a_2 e a_3 foram repetidos. Uma célula primitiva é definida pelos vetores primitivos da rede, representando os pontos da rede.

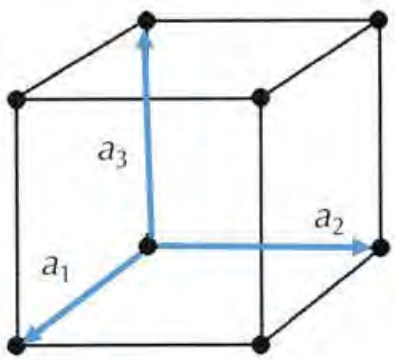

Figura 1.1: Vetores Primitivos em um Cristal

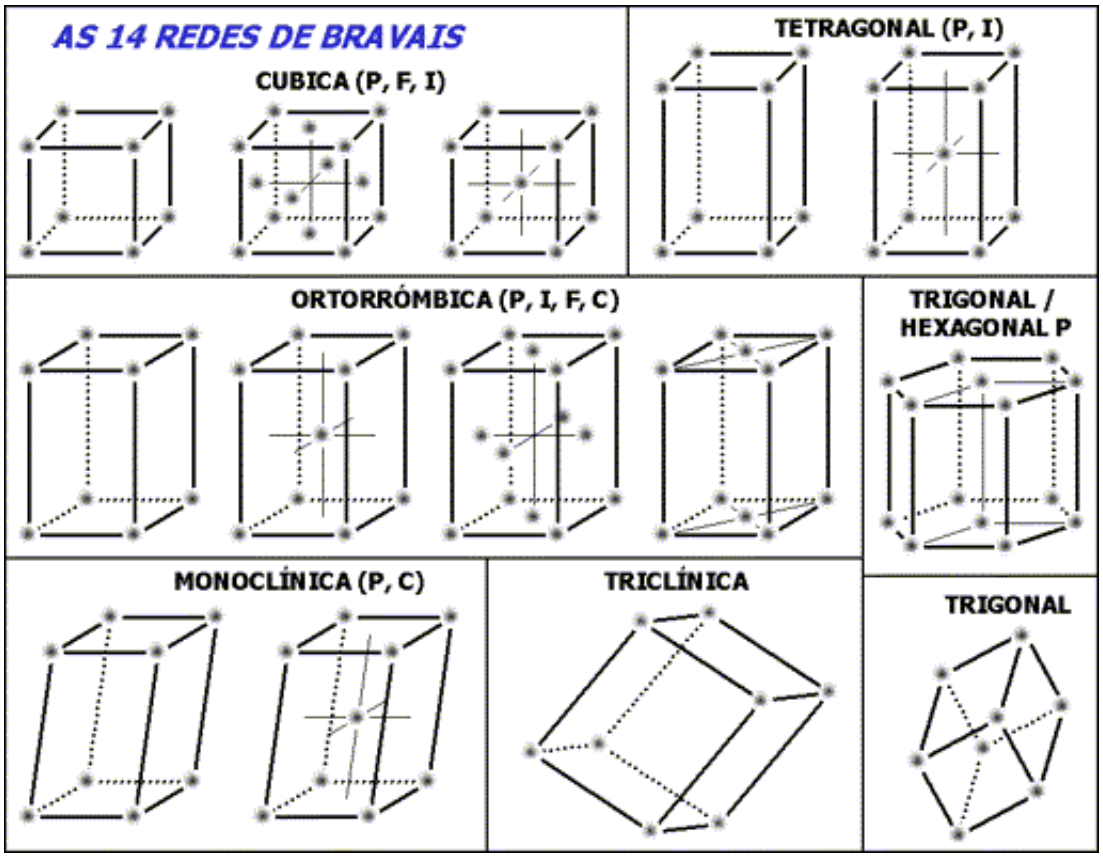

Figura 1.2: As redes de Bravais - Imagem retirada de [Aguiar Junior, 2006]

Chamamos de *base* o conjunto de posições, obtidas através dos vetores primitivos da rede, onde estão localizados um ou mais átomos.

Assim, a estrutura cristalina é uma configuração geométrica dada pela Rede de Bravais, com a inclusão dos átomos da base. Na figura 1.5 temos um exemplo de uma estrutura cristalina plana triangular com vetores primitivos a_1 e a_2 com base formada pelas posições A e A'.

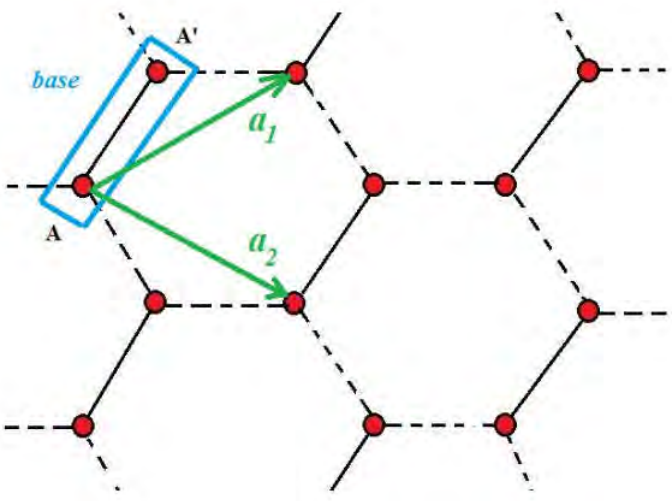

Figura 1.3: Estrutura cristalina plana triangular com vetores primitivos a_1 e a_2 com base formada pelas posições A e A'.

Em [Filho, 2017] se encontra uma descrição breve, porém mais completa e focada em nossos interesses, sobre estruturas cristalinas. Maiores detalhes sobre esse assunto podem ser vistos nos livros clássicos [Kittel, 1971] ou [Ashcroft and Mermin, 1976a].

É comum considerarmos os vetores primitivos com norma 1. Posteriormente multiplica-se cada um desses vetores por fatores para se ter o tamanho real do mesmo. Em uma rede cúbica somente é necessário um fator, chamado de *parâmetro de rede*.

No algoritmo computacional que implementamos, os cálculos foram realizados baseados em uma simulação de estrutura cristalina por super-células, onde sistemas formados por um número muito grande de átomos são descritos através de repetições no espaço de estruturas menores *(as super-células)*, com apenas N átomos.

1.2 O método *Tight-Binding*

Nesta seção vamos apresentar brevemente o *método Tight-Binding*, sem nos aprofundarmos na Mecânica Quântica que o cerca. Maiores detalhes podem se ser vistos em [Ashcroft and Mermin, 1976b] e [Filho, 2017].

Em mecânica quântica os estados de um sistema são determinados por uma função, chamada de função de onda. Funções de onda são soluções para a *equação de Schrödinger*. Para uma nanopartícula de massa constante m, sujeita a uma energia potencial $U(t, x, y, z)$, a *equação de Schrödinger* é

$$i\hbar\frac{\partial\psi}{\partial t}(t, x, y, z) = \widehat{H}\psi(t, x, y, z), \tag{1.1}$$

com

$$\widehat{H} = -\frac{\hbar^2}{2m}\Delta + U(t, x, y, z),$$

onde i representa a unidade imaginária, Δ representa o operador Laplaciano e $\hbar$ é uma constante conhecida como *constante de Dirac*, cujo valor é $1,054 \times 10^{-27} erg\dot{s}$. O operador $\widehat{H}$ é chamado de operador Hamiltoniano.

Quando a energia potencial independe do tempo, podemos usar a técnica de separação de variáveis para resolução. Supondo que $\psi(t, x, y, z) = T(t)\varphi(x, y, z)$, se conclui que existe uma constante $\epsilon \in \mathbb{R}$ tal que

$$i\hbar\frac{T'(t)}{T(t)} = \frac{\widehat{H}\varphi(x, y, z)}{\varphi(x, y, z)} = \epsilon.$$

Logo o problema se resume a encontrar as autofunções do operador Hamiltoniano através da chamada *equação de Schrödinger* independente do tempo, dada por

$$\hat{H}\varphi(x, y, z) = \epsilon\varphi(x, y, z).$$

Para cada estado quântico de um sistema podemos medir várias de suas propriedades, entre elas a energia. Chamamos essas propriedades físicas mensuráveis de observáveis. Para cada observável existe um único operador quântico correspondente. Os valores permitidos para um observável são obtidos através de um problema de autovalor para o operador quântico associado. Quando se trata do problema estacionário, o operador quântico associado a energia mecânica de uma única nanopartícula de massa m é o operador

Hamiltoniano. Sendo assim, os autovalores ϵ na Equação de Schrödinger independente do tempo, são os valores permitidos para a energia.

Considerando que todas as interações eletrônicas entre os diversos componentes do sistema estudado podem ser agrupadas em um único potencial eletrônico que possui a periodicidade $\vec{T}$ da rede cristalina, $U(\vec{r}) = U(\vec{r}+\vec{T})$, as autofunções devem satisfazer o Teorema de Bloch. O teorema está presente em diversos livros, como por exemplo em [Ashcroft and Mermin, 1976b]. Um demonstração do mesmo, bastante clara e autocontida pode ser vista em [Filho, 2017]. Abaixo segue o enunciado do teorema.

Teorema 1.1. (Teorema de Bloch) Considere a equação Schrödinger independente do tempo e U um potencial que possui a mesma periodicidade da rede cristalina. Existe $k \in R^3$ de modo que a aplicação φ é uma autofunção se, e somente se,

$$\varphi_{\vec{k}}((x,y,z)+\vec{T}) = e^{i\vec{k}\cdot\vec{T}}\varphi_{\vec{k}}(x,y,z). \tag{1.2}$$

O vetor $\vec{k} = (k_1, k_2, k_3)$ é um vetor cujo o significado físico depende do conceito de redes recíprocas e teoria de bandas, assuntos não abordados nessa dissertação. As funções que satisfazem a equação 1.2 são chamadas de *funções de Bloch.*

Note que apenas falamos do operador Hamiltoniano de um única nanopartícula. Quando falamos em sólidos cristalinos, como é o nosso caso, o operador Hamiltoniano se altera porque a energia do sistema se altera. Vamos considerar inicialmente a situação de um átomo com n elétrons. Para escrever o operador Hamiltoniano é preciso considerar:

- K_N - representa o operador energia cinética do núcleo;
- K_e - repesenta a soma dos operadores energia cinética de cada elétron;
- U_{eN} - representa a soma dos potenciais coulombianos atrativos de cada par elétron-núcleo;
- U_{ee} - representa a soma dos potenciais coulombianos repulsivos de cada par elétron-elétron.

Assim, passamos a ter

$$\hat{H} = K_N + K_e + U_{eN} + U_{ee}.$$

Se incluirmos vários átomos, a única diferença no operador Hamiltoniano apresentado acima, é que iremos precisar incluir termos referentes aos núcleos agora presentes no sistema. Logo, K_N irá representar o operador energia cinética dos núcleos, U_{eN} precisará contabilizar a interação de todos os pares elétron-núcleo e será preciso adicionar uma energia U_{NN} que irá representar a soma dos potenciais coulombianos repulsivos de cada par núcleo-núcleo.

Vamos supor que o operador hamiltoniano para um único átomo tem espectro limitado por n auto-valores, possuindo então n auto-funções, que chamaremos de orbitais, e que a estrutura cristalina possui N átomos localizados em $T_1, T_2, \cdots, T_N$.

Para cada orbital j, definimos o j-ésimo *orbital de Bloch* como sendo a seguinte combinação linear de orbitais atômicos

$$\Phi_j(\vec{k}, \vec{x}) = \frac{1}{\sqrt{N}} \sum_{r=1}^{N} e^{i\vec{k}\cdot\vec{T}_r} \varphi_j(\vec{x} - \vec{T}_r), \quad j = 1, \cdots, n.$$

No método *tight-binding*, se propõe que as auto-funções Ψ_j do operador de Hamiltoniano sejam dadas por

$$\Psi_j(\vec{k}, \vec{x}) = \sum_{j'=1}^{n} C_{jj'}(\vec{k}) \Phi_{j'}(\vec{k}, \vec{x}), \quad j = 1, \cdots, n$$

onde os coeficientes $C_{jj'}$ da combinação linear precisam ser determinados. A acurácia desse proposta está intimamente ligada com o número n de orbitais usados usados nessa representação.

O j-ésimo auto-valor $E_j(\vec{k})$ do operador hamiltoniano nessa proposta pode então ser determinado por

$$E_j(\vec{k}) = \frac{\left\langle \Psi_j | \hat{H} | \Psi_j \right\rangle}{\langle \Psi_j | \Psi_j \rangle},$$

onde a notação $\left\langle f | \hat{H} | g \right\rangle$ significa

$$\left\langle f | \hat{H} | g \right\rangle = \iiint_{\mathbb{R}^3} f^*(x, y, z) \hat{H} g(x, y, z) dx dy dz,$$

com f^* representando o conjugado da função complexa f e $\langle f | f \rangle$ significa

$$\langle f | f \rangle = \iiint_{\mathbb{R}^3} f^*(x, y, z) f(x, y, z) dx dy dz.$$

Temos então que

$$\begin{aligned} E_j(\vec{k}) &= \frac{\left\langle \Psi_j | \hat{H} | \Psi_j \right\rangle}{\langle \Psi_j | \Psi_j \rangle} \\ &= \frac{\sum_{jj'=1}^{n} C^*_{jj'} C_{jj'} \left\langle \Phi_j | \hat{H} | \Phi_{j'} \right\rangle}{\sum_{jj'=1}^{n} C^*_{jj'} C_{jj'} \langle \Phi_j | \Phi_j \rangle} \\ &= \frac{\sum_{jj'=1}^{n} C^*_{jj'} C_{jj'} H_{jj'}}{\sum_{jj'=1}^{n} C^*_{jj'} C_{jj'} S_{jj'}}, \end{aligned}$$

onde as matrizes $H_{jj'}$ e $S_{jj'}$ são chamadas de matrizes integrais de *transferência* e *overlap* respectivamente.

Para se encontrar a energia no estado fundamental de um cristal é preciso encontrar $C_{jj'}$ que minimizam $E_j(\vec{k})$. Para cada j fixado, fazendo $\frac{\partial E_j(\vec{k})}{\partial C_{jj}} = 0$ se obtêm a equação matricial

$$[H - E_j(\vec{k})S]C_j = 0,$$

onde C_j é o vetor coluna definido por $C_j = [C_{j1}\ C_{j2} \cdots C_{jn}]^T$.

Admitindo que esse sistema matricial homogêneo não tenha somente a solução única, é necessário que a matriz $[H - E_j(\vec{k})S]$ não seja inversível, o que é equivalente a exigir que

$$det[H - E_j(\vec{k})S] = 0, \quad j = 1, 2, \cdots, n$$

Esta última equação é chamada de *equação secular*. Se a base de orbitais de Bloch formar um conjunto ortonormal, a matriz de overlap S será a matriz identidade e então a equação secular se torna o polinômio característico da matriz H, dado por

$$det[H - EI] = 0,$$

onde H agora é chamada de matriz do hamiltoniano cristalino e E é a energia mínima.

Uma vez conhecendo a matriz do hamiltoniano do sistema, o método *tight-binding* se resume ao cálculo da equação secular apenas por resolução de sistemas de equações lineares. Note que embora a estratégia seja simples, possui alto custo computacional dependendo do tamanho da matriz H. Nessa metodologia

se faz uso de representações indiretas para o hamiltoniano, chamadas *parametrizações semi-empíricas do hamiltoniano cristalino.* Especificamente, nesse trabalho onde fazemos um estudo de caso para o silício, usamos a parametrização proposta por Kwon [Kwon et al., 1994].

1.2.1 O Hamiltoniano de Kwon para Cristais de Silício

O silício é o segundo elemento mais abundante da crosta terrestre, superado apenas pelo oxigênio. Segundo [Ciências, 1999], o silício é um elemento extremamente importante na indústria eletrônica, já que quando dopado com elementos como boro, fósforo e arsênio, forma materiais semicondutores, fundamentais na construção de chips de computadores e vários outros componentes de circuitos eletrônicos. Além disso, o silício tem grande aplicação na indústria metalúrgica sendo usado na produção de ligas de aços, latões e bronzes. O silício também é utilizado na fabricação de tijolos e de diferentes concretos, na fabricação de vidros especiais, dentre outros.

A célula primitiva do Silício é formada por uma estrutura cúbica face centrada (figura 1.4). A base é formada por dois átomos, um na origem do cubo e outro a 1/4 da diagonal do cubo. Seu parâmetro de rede é $a = 5,43$.

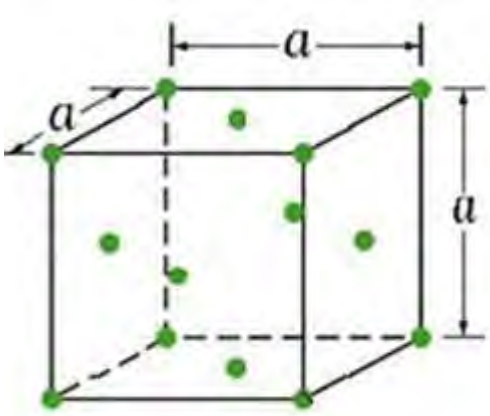

Figura 1.4: Célula Cúbica de Face Centrada

O silício pertence ao grupo 14 da Classificação Periódica dos Elementos e possui $1s^2 2s^2 2p^6 3s^2 3p^2$ como configuração eletrônica, possuindo camada de valência composta por $3s^2 3p^2$.

Como visto na seção anterior, precisamos de uma parametrização do hamiltoniano cristalino para

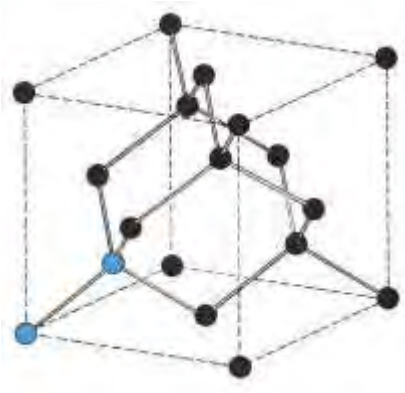

Figura 1.5: Célula Primitiva do Silício

utilizar a metodologia *tight-binding*. Os elementos da matriz hamiltoniana são dados por

$$\left\langle \Psi_m | \hat{H} | \Psi_n \right\rangle = \frac{1}{N} \sum_{R=1}^{N} e^{ikR} \int \varphi_m^*(\vec{x}) \hat{H} \varphi_n(\vec{x} - T_R) dx dy dx.$$

Os pioneiros na parametrização da matriz hamiltoniana foram Slater e Koster , que em [Slater and Koster, 1954] propuseram trocar as integrais nos elementos da matriz de transferência, por parâmetros independentes de k que reproduzem corretamente os valores de energia. Nessa proposta a matriz hamiltoniana possuirá 4 índices, m e n para indicar os orbitais atômicos e i e j fazendo referência aos átomos onde estão localizados os orbitais m e n. Podemos denotar seus elementos então por H_{m^i,n^j}.

Nessa seção apresentaremos a parametrização proposta por Kwon [Kwon et al., 1994] para o Silício. Como o propósito dessa dissertação é a implementação computacional do método DMTB (e para isso será necessário a matriz do Hamiltoniano - veja seção 1.3), faremos uma abordagem baseada em [Nakano, 2015] para explicitar os elementos da matriz do hamiltoniano cristalino para o silício.

Vamos representar a estrutura eletrônica dos átomos de silício como combinações lineares de 4 orbitais atômicos por átomo, um orbital $3s$ e três orbitais $3p$: $3p_x$, $3p_y$ e $3p_z$, centrados em cada átomo. Para encontrar a matriz do hamiltoniano, precisamos encontrar os elementos de matriz que dependem desses orbitais atômicos em diferentes distâncias interatômicas, conforme as definições de orbital de Bloch e de matriz de transferência. A matriz hamiltoniana é uma matriz $NM \times NM$, onde N é o número de átomos na base do cristal e M é o número de orbitais. Essa matriz pode ser vista como uma matriz de matrizes 4×4, onde para cada entrada ij temos entradas correspondentes as iterações entre os orbitais do átomo i e do átomo j. Por exemplo, se estivermos somente com 2 átomos de silício, a estrutura básica da matriz

do hamiltoniano será como na figura 1.6.

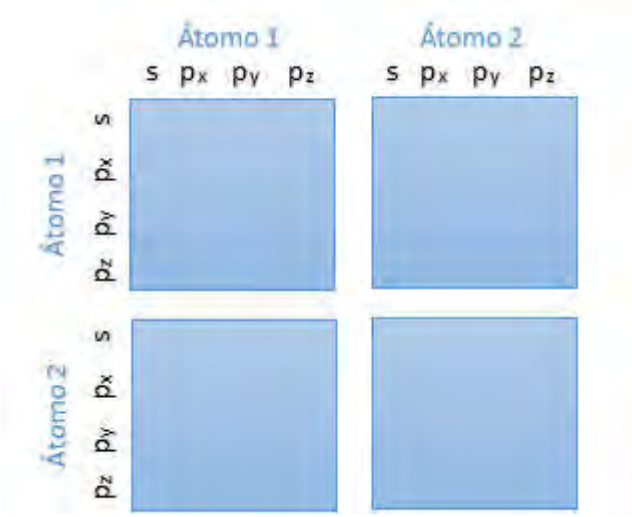

Figura 1.6: Estrutura básica da matriz do Hamiltoniano com 2 átomos

Cada bloco da matriz é formado por elementos referentes a interação dos orbitais na seguinte forma:

$$\begin{bmatrix} \langle s|\hat{H}|s\rangle & \langle s|\hat{H}|p_x\rangle & \langle s|\hat{H}|p_y\rangle & \langle s|\hat{H}|p_z\rangle \\ \langle p_x|\hat{H}|s\rangle & \langle p_x|\hat{H}|p_x\rangle & \langle p_x|\hat{H}|p_y\rangle & \langle p_x|\hat{H}|p_z\rangle \\ \langle p_y|\hat{H}|s\rangle & \langle p_y|\hat{H}|p_x\rangle & \langle p_y|\hat{H}|p_y\rangle & \langle p_y|\hat{H}|p_z\rangle \\ \langle p_z|\hat{H}|s\rangle & \langle p_z|\hat{H}|p_x\rangle & \langle p_z|\hat{H}|p_y\rangle & \langle p_z|\hat{H}|p_z\rangle \end{bmatrix}.$$

Sendo assim, se um sistema possui N átomos de silício então a matriz do Hamiltoniano será uma matriz $4N \times 4N$. Passaremos a explicitar cada entrada desse matriz, para isso serão necessários os parâmetros de Kwon, listados nas tabelas 1.1 e 1.2. No modelo da parametrização proposto por Kwon em [Kwon et al., 1994] as integrais de transferência são parametrizadas decompondo o orbital p em componentes normais p_n e paralelas p_d ao eixo de ligação dos átomos. Assim, para construir os elementos de matriz do hamiltoniano, precisamos descrever p_x, p_y e p_z em relação p_n e p_d.

r_0 **(Å)**	**n**	E_s **(eV)**	E_p **(eV)**
2,360352	2	-5,25	1,2

Tabela 1.1: Parâmetros de Kwon

Segundo [Paxton and Sutton, 1989], os orbitais s, p_x, p_y e p_z são ortogonais, sendo assim, os blocos

que formam a diagonal da matriz do hamiltoniano $(i = j)$ são blocos de matrizes diagonais. As entradas da diagonal principal são $\left\langle s|\hat{H}|s\right\rangle$, $\left\langle p_x|\hat{H}|p_x\right\rangle$, $\left\langle p_y|\hat{H}|p_y\right\rangle$ e $\left\langle p_z|\hat{H}|p_z\right\rangle$. Esses dados foram tabelados por Kwon [Kwon et al., 1994], valendo E_s o primeiro e E_p os demais (veja a tabela 1.1). Dessa forma, os blocos para $i = j$ são

$$\begin{bmatrix} E_s & 0 & 0 & 0 \\ 0 & E_p & 0 & 0 \\ 0 & 0 & E_p & 0 \\ 0 & 0 & 0 & E_p \end{bmatrix}. \tag{1.3}$$

Para deduzir as fórmulas para os blocos fora da diagonal principal $(i \neq j)$ precisamos calcular as integrais de transferência correspondentes. Segundo [Nakano, 2015], consideramos somente quatro integrais de transferência que correspondem as possíveis ligações químicas entre átomos de silício descritas na figura 1.7.

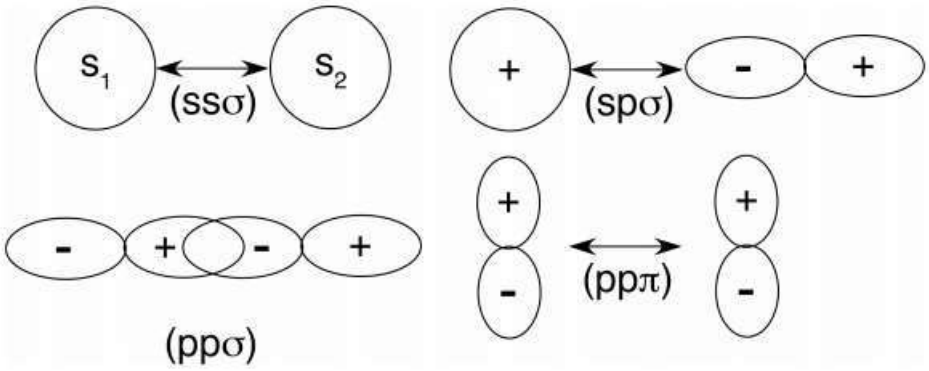

Figura 1.7: Ligações entre átomos de silício - Figura retirada de [Nakano, 2015]

λ	**ss**σ	**sp**σ	**pp**σ	**pp**π
$h_\lambda(r_0)(Ev)$	-2,038	1,745	2,75	-1,075
n_λ	9,5	8,5	7,5	7,5
$r_\lambda(A)$	3,4	3,55	3,7	3,7

Tabela 1.2: Parâmetros Eletrônicos de Kwon

A primeira entrada dessa matriz é dada por $\left\langle s_1|\hat{H}|s_2\right\rangle$. Segundo [Kwon et al., 1994], essa integral pode ser expressa de acordo com a equação 1.4. Os parâmetros necessários para estimar essa integral estão na segunda coluna da tabela 1.2.

$$h_\lambda(r) = h_\lambda(r_0)\left(\frac{r_0}{r}\right)^n exp\left(n\left[-\left(\frac{r}{r_\lambda}\right)^{n_\lambda} + \left(\frac{r_0}{r_\lambda}\right)^{n_\lambda}\right]\right), \tag{1.4}$$

com r sendo a norma do vetor d.

Em seguida, vamos calcular as integrais entre orbitais s e orbitais p. Para isso, vamos usar a notação p_α, onde α pode ser x, y ou z e p_n é a componente normal de p_α em relação ao eixo de ligação entre os átomos d e p_d é a componente paralela de p_α com relação ao mesmo eixo. (veja figura 1.8).

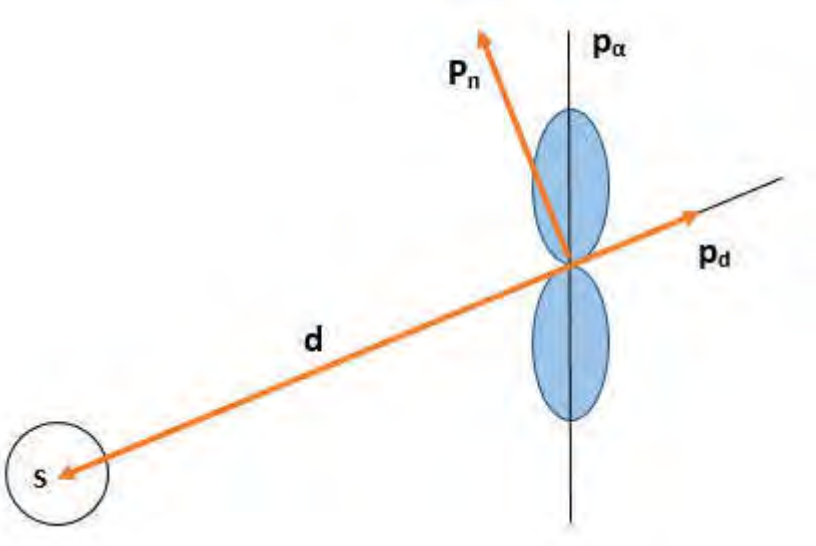

Figura 1.8: Ligações entre Orbital S com Orbital P

Dessa forma é possível escrever

$$p_\alpha = (\vec{\alpha}\cdot\vec{n})p_n + (\vec{\alpha}\cdot\vec{d})p_d,$$

onde $\cdot$ representa o produto escalar entre os vetores.

Assim,

$$\begin{aligned}
\left\langle s_1|\hat{H}|p_{2\alpha}\right\rangle &= \left\langle s_1|\hat{H}|(\vec{\alpha_2}\cdot\vec{n})p_{2n} + (\vec{\alpha_2}\cdot\vec{d})p_{2d}\right\rangle \\
&= \left\langle s_1|\hat{H}|(\vec{\alpha_2}\cdot\vec{n})p_{2n}\right\rangle + \left\langle s_1|\hat{H}|(\vec{\alpha_2}\cdot\vec{d})p_{2d}\right\rangle \\
&= (\vec{\alpha_2}\cdot\vec{n})\left\langle s_1|\hat{H}|p_{2n}\right\rangle + (\vec{\alpha_2}\cdot\vec{d})\left\langle s_1|\hat{H}|p_{2d}\right\rangle \\
&= (\vec{\alpha_2}\cdot\vec{d})\left\langle s_1|\hat{H}|p_{2d}\right\rangle \\
&= (\vec{\alpha_2}\cdot\vec{d})h_{sp\sigma}
\end{aligned}$$

Sendo a distância entre os orbitais dada pelo vetor unitário $\vec{d} = (d_x, d_y, d_z)$, então:

$$\left\langle s_1|\hat{H}|p_{2x}\right\rangle = ((1,0,0)\cdot\vec{d})h_{sp\sigma} = d_x h_{sp\sigma},$$

$$\left\langle s_1|\hat{H}|p_{2y}\right\rangle = ((0,1,0)\cdot\vec{d})h_{sp\sigma} = d_y h_{sp\sigma},$$

$$\left\langle s_1|\hat{H}|p_{2z}\right\rangle = ((0,0,1)\cdot\vec{d})h_{sp\sigma} = d_z h_{sp\sigma}.$$

Os parâmetros necessários para estimar integral $\left\langle s_1|\hat{H}|p_{2\alpha}\right\rangle$ estão na terceira coluna da tabela 1.2, e sua expressão é dada na equação 1.4. Temos também que

$$\left\langle p_{1x}|\hat{H}|s_2\right\rangle = -d_x h_{sp\sigma},$$

$$\left\langle p_{1y}|\hat{H}|s_2\right\rangle = -d_y h_{sp\sigma},$$

$$\left\langle p_{1z}|\hat{H}|s_2\right\rangle = -d_z h_{sp\sigma}.$$

Já que a única diferença no cálculo dessas integrais é a orientação do vetor $\vec{d}$. Novamente os parâmetros necessários para estimar essa integral estão na segunda coluna da tabela 1.2.

Por fim, vamos calcular as integrais entre orbitais p. Observe a figura 1.9.

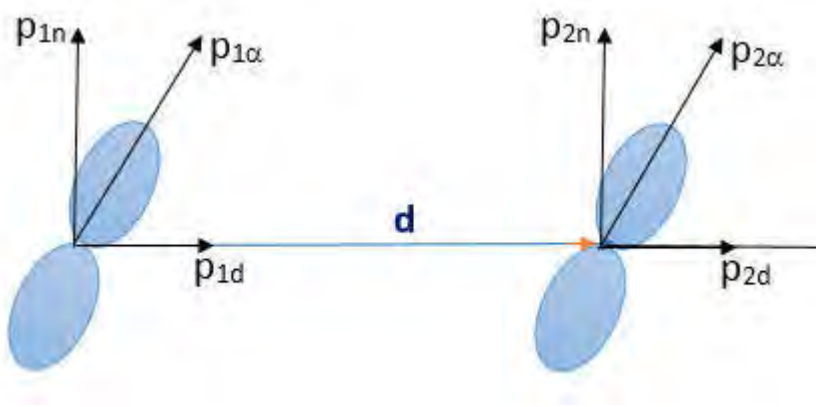

Figura 1.9: Ligações entre Orbitais P

Temos que,

$$\begin{aligned}
\left\langle p_{1\alpha}|\hat{H}|p_{2\alpha}\right\rangle &= \left\langle(\vec{\alpha_1}\cdot\vec{n})p_{1n}+(\vec{\alpha_1}\cdot\vec{d})p_{1d}|\hat{H}|(\vec{\alpha_2}\cdot\vec{n})p_{2n}+(\vec{\alpha_2}\cdot\vec{d})p_{2d}\right\rangle \\
&= (\vec{\alpha_1}\cdot\vec{n})(\vec{\alpha_2}\cdot\vec{n})\left\langle p_{1n}|\hat{H}|p_{2n}\right\rangle+(\vec{\alpha_1}\cdot\vec{d})(\vec{\alpha_2}\cdot\vec{n})\left\langle p_{1d}|\hat{H}|p_{2n}\right\rangle \\
&+ (\vec{\alpha_1}\cdot\vec{n})(\vec{\alpha_2}\cdot\vec{d})\left\langle p_{1n}|\hat{H}|p_{2d}\right\rangle+(\vec{\alpha_1}\cdot\vec{d})(\vec{\alpha_2}\cdot\vec{d})\left\langle p_{1d}|\hat{H}|p_{2d}\right\rangle \\
&= (\vec{\alpha_1}\cdot\vec{n})(\vec{\alpha_2}\cdot\vec{n})\left\langle p_{1n}|\hat{H}|p_{2n}\right\rangle+(\vec{\alpha_1}\cdot\vec{d})(\vec{\alpha_2}\cdot\vec{d})\left\langle p_{1d}|\hat{H}|p_{2d}\right\rangle \\
&= (\vec{\alpha_1}\cdot\vec{n})(\vec{\alpha_2}\cdot\vec{n})h_{pp\pi}+(\vec{\alpha_1}\cdot\vec{d})(\vec{\alpha_2}\cdot\vec{d})h_{pp\sigma}.
\end{aligned}$$

Note que

$$\begin{aligned}
(\vec{\alpha_1}\cdot\vec{n})(\vec{\alpha_2}\cdot\vec{n}) &= (\vec{\alpha_1}\cdot\vec{n})(\vec{\alpha_2}\cdot\vec{n})(\vec{n}\cdot\vec{n}) \\
&= ((\vec{\alpha_1}\cdot\vec{n})\vec{n})\cdot((\vec{\alpha_2}\cdot\vec{n})\vec{n}) \\
&= (\vec{\alpha_1}-(\vec{\alpha_1}\cdot\vec{d})\vec{d})\cdot(\vec{\alpha_2}-(\vec{\alpha_2}\cdot\vec{d})\vec{d})
\end{aligned}$$

Usando essa expressão na equação 1.5 ficamos com

$$\begin{aligned}
\left\langle p_{1\alpha}|\hat{H}|p_{2\alpha}\right\rangle &= (\vec{\alpha_1}\cdot\vec{n})(\vec{\alpha_2}\cdot\vec{n})h_{pp\pi}+(\vec{\alpha_1}\cdot\vec{d})(\vec{\alpha_2}\cdot\vec{d})h_{pp\sigma} \\
&= (\vec{\alpha_1}-(\vec{\alpha_1}\cdot\vec{d})\vec{d})\cdot(\vec{\alpha_2}-(\vec{\alpha_2}\cdot\vec{d})\vec{d})h_{pp\pi}+(\vec{\alpha_1}\cdot\vec{d})(\vec{\alpha_2}\cdot\vec{d})h_{pp\sigma}
\end{aligned}$$

Assim,

$$\begin{aligned}
\left\langle p_{1x}|\hat{H}|p_{2x}\right\rangle &= ((1,0,0)-((1,0,0)\cdot\vec{d})\vec{d})\cdot((1,0,0)-((1,0,0)\cdot\vec{d})\vec{d})h_{pp\pi} \\
&+ ((1,0,0)\cdot\vec{d})((1,0,0)\cdot\vec{d})h_{pp\sigma} \\
&= ((1-d_x^2,-d_xd_y,-d_xd_z)\cdot(1-d_x^2,-d_xd_y,-d_xd_z))h_{pp\pi}+d_x^2h_{pp\sigma} \\
&= ((1-d_x^2)^2+{d_x}^2{d_y}^2+{d_x}^2{d_z}^2)h_{pp\pi}+d_x^2h_{pp\sigma} \\
&= (1-d_x^2)h_{pp\pi}+d_x^2h_{pp\sigma}.
\end{aligned}$$

$$\begin{aligned}
\left\langle p_{1y}|\hat{H}|p_{2y}\right\rangle &= ((0,1,0)-((0,1,0)\cdot\vec{d})\vec{d})\cdot((0,1,0)-((0,1,0)\cdot\vec{d})\vec{d})h_{pp\pi} \\
&+ ((0,1,0)\cdot\vec{d})((0,1,0)\cdot\vec{d})h_{pp\sigma} \\
&= ((-d_yd_x,1-d_y^2,-d_yd_z)\cdot(-d_yd_x,1-d_y^2,-d_yd_z))h_{pp\pi}+d_y^2h_{pp\sigma} \\
&= ({d_y}^2{d_x}^2+(1-d_y^2)^2+{d_y}^2{d_z}^2)h_{pp\pi}+d_y^2h_{pp\sigma} \\
&= (1-d_y^2)h_{pp\pi}+d_y^2h_{pp\sigma}.
\end{aligned}$$

$$\begin{aligned}
\left\langle p_{1z}|\hat{H}|p_{2z}\right\rangle &= ((0,0,1)-((0,0,1)\cdot\vec{d})\vec{d})\cdot((0,0,1)-((0,0,1)\cdot\vec{d})\vec{d})h_{pp\pi} \\
&+ ((0,0,1)\cdot\vec{d})((0,0,1)\cdot\vec{d})h_{pp\sigma} \\
&= ((-d_zd_x,-d_zd_y,1-d_z^2)\cdot(-d_zd_x,-d_zd_y,1-d_z^2)h_{pp\pi}+d_z^2h_{pp\sigma} \\
&= ({d_z}^2{d_x}^2+{d_z}^2{d_y}^2+(1-d_z^2)^2)h_{pp\pi}+d_z^2h_{pp\sigma} \\
&= (1-d_z^2)h_{pp\pi}+d_z^2h_{pp\sigma}.
\end{aligned}$$

$$\begin{aligned}
\left\langle p_{1x}|\hat{H}|p_{2y}\right\rangle &= ((1,0,0)-((1,0,0)\cdot\vec{d})\vec{d})\cdot((0,1,0)-((0,1,0)\cdot\vec{d})\vec{d})h_{pp\pi} \\
&+ ((1,0,0)\cdot\vec{d})((0,1,0)\cdot\vec{d})h_{pp\sigma} \\
&= ((1-d_x^2,-d_xd_y,-d_xd_z)\cdot(-d_yd_x,1-d_y^2,-d_yd_z))h_{pp\pi}+d_xd_yh_{pp\sigma} \\
&= (-(1-d_x^2)d_yd_x-(1-d_y^2)d_xd_y+d_xd_yd_z^2)h_{pp\pi}+d_xd_yh_{pp\sigma} \\
&= -d_xd_yh_{pp\pi}+d_xd_yh_{pp\sigma}.
\end{aligned}$$

$$\begin{aligned}
\left\langle p_{1x}|\hat{H}|p_{2z}\right\rangle &= ((1,0,0)-((1,0,0)\cdot\vec{d})\vec{d})\cdot((0,0,1)-((0,0,1)\cdot\vec{d})\vec{d})h_{pp\pi} \\
&+ ((1,0,0)\cdot\vec{d})((0,0,1)\cdot\vec{d})h_{pp\sigma} \\
&= ((1-d_x^2,-d_xd_y,-d_xd_z)\cdot(-d_zd_x,-d_zd_y,1-d_z^2))h_{pp\pi}+d_xd_zh_{pp\sigma} \\
&= (-(1-d_x^2)d_zd_x+d_xd_y^2d_z-(1-d_z^2)d_xd_z)h_{pp\pi}+d_xd_zh_{pp\sigma} \\
&= -d_zd_xh_{pp\pi}+d_xd_zh_{pp\sigma}.
\end{aligned}$$

De modo análogo se obtêm as demais entradas da matriz hamiltoniana. Explicitamente, a matriz hamiltoniana H, está dada abaixo.

$$\begin{bmatrix} h_{ss\sigma} & d_x h_{sp\sigma} & d_y h_{sp\sigma} & d_z h_{sp\sigma} \\ -d_x h_{sp\sigma} & d_x^2 h_{pp\sigma} + (1-d_x^2) h_{pp\pi} & d_x d_y (h_{pp\sigma} - h_{pp\pi}) & d_x d_z (h_{pp\sigma} - h_{pp\pi}) \\ -d_y h_{sp\sigma} & d_y d_x (h_{pp\sigma} - h_{pp\pi}) & d_y^2 h_{pp\sigma} + (1-d_y^2) h_{pp\pi} & d_y d_z (h_{pp\sigma} - h_{pp\pi}) \\ -d_z h_{sp\sigma} & d_z d_x (h_{pp\sigma} - h_{pp\pi}) & d_z d_y (h_{pp\sigma} - h_{pp\pi}) & d_z^2 h_{pp\sigma} + (1-d_z^2) h_{pp\pi} \end{bmatrix}. \tag{1.5}$$

Uma informação importante sobre a parametrização Kwon é que o hamiltoniano é transferível, ou seja, independe da estrutura cristalina que estamos usando. Os parâmetros de Kwon já incorporam as informações da espécie química e as variações que ocorrem se desejarmos variar o parâmetro de rede do Silício aparecem explicitamente, visto que as entradas das matriz dependem diretamente da distância entre os átomos.

1.3 O método DMTB

O método *tight-binding* que falamos brevemente na seção 1.2, fornece bons resultados acerca das propriedades eletrônicas e estruturais de sistemas cristalino quando se compara com resultados experimentais. Infelizmente, esse método exige um grande custo computacional o que o deixa inviável para o estudo de sistemas com um grande número de átomos. Esses sistemas precisam ser tratados com uma metodologia com um menor custo computacional, mas que ainda leve em conta uma descrição quântica das propriedades dos elétrons.

O método da matriz densidade DMTB foi desenvolvido por [Li et al., 1993] em 1992 está diretamente ligado ao tratamento *tight binding*. No mesmo ano, usando técnicas diferentes, [Daw, 1993] obteve um método bastante similar a este. Em 1996 [Millam and Scuseria, 1997], fez um pequena modificação nessa metodologia para cristais de carbono.

Nessa seção fazemos uma breve descrição do método, nos preocupando com importantes detalhes quando observados no ponto de vista computacional. Para maiores detalhes do método veja [Filho, 2017].

Para o método DMTB precisamos do *operador densidade* (*ou matriz densidade*). Nessa dissertação não trataremos profundamente desse operador, usaremos somente suas propriedades para desenvolvimento do método. Para detalhes sobre o operador densidade veja [Amaral et al., 2011]. Para uma abordagem da metodologia Tight-Binding sobre o ponto de vista do operador densidade recomendamos [Paxton and Sutton, 1989].

Sendo ρ a matriz densidade, podemos escrever o número de elétrons como - N_e e a energia eletrônica total *(parte atrativa da energia total do sistema)* - E como

$$N_e = tr(\rho) \tag{1.6}$$

e

$$E = tr(\rho H). \tag{1.7}$$

Nessa metodologia geralmente a energia total do sistema com N_e elétrons é dada por

$$E_{total}(\rho) = E(\rho) + E_{rep} + NE_0, \tag{1.8}$$

onde E_{rep} corresponde ao potencial repulsivo, N é o número de átomos do sistema e E_0 é uma constante de energia por átomo.

De acordo com [Kwon et al., 1994], a formulação parametrizada para E_{rep} para supercélulas formadas por átomos de silício é dada por

$$E_{rep} = \sum_i f\left(\sum_j \phi(r_{ij})\right),$$

com

$$f(x) = C_1 x + C_2 x^2 + C_3 x^3 + C_4 x^4$$

e

$$\phi_(r) = \left(\frac{r_0}{r}\right)^m exp\left(m\left[-\left(\frac{r}{d_c}\right)^{m_c} + \left(\frac{r_0}{d_c}\right)^{m_c}\right]\right).$$

Onde r é a norma do vetor d, vetor que corresponde a distância orbitais e $r_0 = 2,360253$ conforme listado na tabela 1.1.

As demais constantes necessárias para o cálculo de E_{rep} estão nas tabelas 1.3 e 1.4.

C_1 **(eV)**	C_2 **(eV)**	C_3 **(eV)**	C_4 **(eV)**
$2,1604385$	$-0,1384393$	$5,8398423 \times 10^{-3}$	$-8,0263577 \times 10^{-5}$

Tabela 1.3: Parâmetros de Kwon em f

Na equação 1.8, note que a única parcela da energia total que depende de ρ é E. Logo, para minimizar a energia total é preciso minimizar a equação 1.7 com a restrição de manter o número de partículas dado pela

E_0 **(eV)**	m	m_c	r_c **(Å)**
8,7393204	6,8755	13,017	3,66995

Tabela 1.4: Parâmetros de Kwon usados em $\phi(r)$

equação 1.6 fixo. Com isso, usando o Teorema dos Multiplicadores de Lagrange [Lima, 2008], o objetivo é minimizar o potencial

$$\Omega(\rho) = tr(\rho H) + \mu(tr(\rho) - N_e), \tag{1.9}$$

onde μ é a constante lagrangiana que força que a quantidade de elétrons seja fixa.

Sabemos que a matriz densidade é local no espaço real, ou seja,

$$\rho_{ij} \to 0 \qquad \text{quando} \qquad R_{ij} \to \infty,$$

onde R_{ij} é distância entre os orbitais i e j. Para o cálculo da energia do sistema, a principio seria necessário percorrer todos os elementos da matriz densidade, porém isso gastaria um alto tempo de processamento, embora não ofereça ganhos físicos. Uma boa alternativa é usar a seguinte aproximação:

$$\rho_{ij} = 0 \qquad \text{quando} \qquad R_{ij} > R_c$$

onde R_c é um raio de corte previamente definido.

Além disso, a matriz densidade é uma matriz idempotente, ou seja

$$\rho^2 = \rho.$$

Isso implica que o autovalores de ρ podem ser somente 0 ou 1. É preciso então impor, mesmo que indiretamente, a restrição da idempotencia. Essa imposição vem na forma de uma purificação proposta por MacWeeney [McWeeny, 1960]

$$\tilde{\rho} = 3\rho^2 - 2\rho^3.$$

Note que dessa forma, se λ for um autovalor para ρ, então $3\lambda^2 - 2\lambda^3$ será um autovalor para $\tilde{\rho}$ e conforme podemos ver na figura 1.3, quando temos autovalores originais entre -0.5 e 1.5 a purificação os leva para valores entre 0 e 1.

A partir desse momento consideramos a matriz $\tilde{\rho}$ como a matriz densidade propriamente dita (com significado físico) e ρ como uma matriz densidade tentativa.

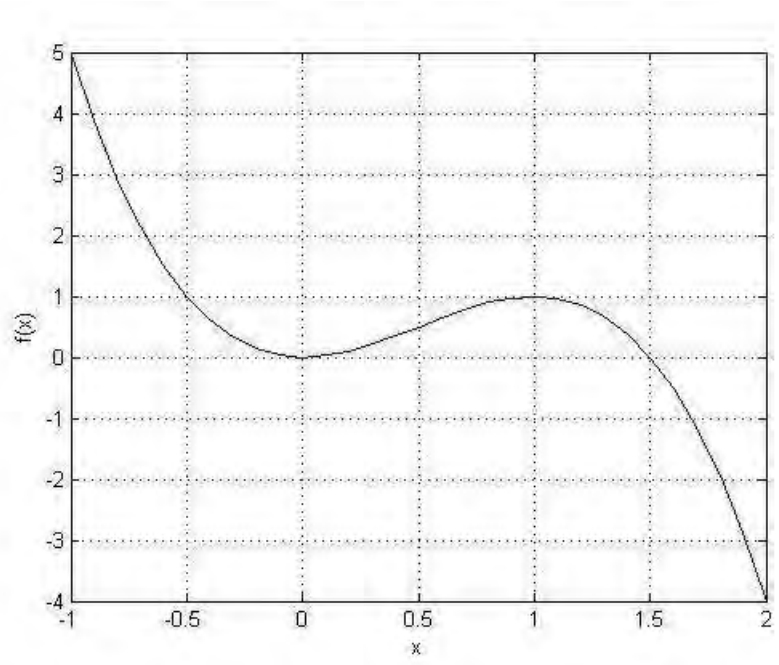

Figura 1.10: Gráfico da função de purificação de MacWeeney

Com essa purificação, o potencial segundo [Millam and Scuseria, 1997] passará a ser escrito como

$$\Omega(\rho) = tr(\tilde{\rho}H) + \mu(tr(\rho) - N_e). \tag{1.10}$$

É esse funcional que é minimizado no método DMTB. No capítulo 2 apresentamos detalhadamente o método do gradiente conjugado não-linear que foi o método escolhido para fazer a implementação do método.

Capítulo 2

O Método do Gradiente Conjugado

Os métodos de gradientes conjugados estão bastante presentes em artigos de Física da Matéria Condensada.Podemos citar, entre outros, [Nightingale et al., 1993], [Ordejón et al., 1995], [Ordejón et al., 1996], [I. Morrison and Payne, 1997], [Pfrommer et al., 1999], [Jiang and Yang, 2004] e [VandeVondele et al., 2012]. Nesse capítulo apresentaremos os Métodos do Gradiente Conjugado Linear e do Gradiente Conjugado Não Linear . Ao longo deste capítulo, usaremos resultados da Álgebra Linear e do Cálculo Vetorial. Como referências para tais resultados recomendamos [Lang, 1966], [Lima, 2008], [Hoffman and Kunze, 1971] e [Spivak, 1965].

Conforme visto no fim do capítulo 1, o método DMTB recai no problema de minimizar o funcional $\Omega : \mathcal{M}_{n\times n} \to \mathbb{R}$, onde $\mathcal{M}_{n\times n}$ representa o espaço vetorial das matrizes $n \times n$. Esse fato faz com que existam diversas maneiras de se concluir a aplicação do método DMTB, visto que o método dependerá intimamente da estratégia de otimização escolhida. No decorrer desse capítulo vamos mostrar como usar o método dos gradientes conjugados não linear para esse fim.

Consideremos o problema de minimização irrestrita

$$\min_{x\in\mathbb{R}^n} f(x),$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é de classe C^∞.

Na década de 50, Hestenes e Stiefel, desenvolveram o Método dos Gradientes Conjugados (CG), que é um método iterativo para resolução de sistemas lineares de muitas equações. Posteriormente, na década de 60, Fletcher e Reeves [Fletcher and Reeves, 1964] adaptaram o CG para ser utilizado em problemas de otimização, introduzindo assim o primeiro Método do Gradiente Conjugado Não Linear (NLCG). Ao

longo dos anos, várias variantes desse método foram criadas e hoje em dia são largamente utilizadas. Uma das mais comuns é a que foi a desenvolvida por Polak e Ribiere em 1969 [Polak and Ribiere, 1969]. Na próxima seção discutiremos brevemente o método do Gradiente Conjugado Linear e posteriormente, com mais profundidade, o conjunto de técnicas que chamamos de método do Gradiente Conjugado Não Linear, já que este é o método que usaremos nessa dissertação.

2.1 O Gradiente Conjugado Linear

Um sistema linear com n equações e n incógnitas pode ser escrito de forma matricial como $Ax = b$, onde $A \in \mathcal{M}_{n\times n}$ e $x, b \in \mathbb{R}^n$, sendo esses vetores tomados como vetores coluna. Se a matriz A dos coeficientes do sistema for inversível, a única solução do sistema é dada por $x = A^{-1}b$. Porém, o cálculo de A^{-1} é uma tarefa de alto custo computacional.

Quando os sistemas lineares que precisam ser resolvidos são de grandes proporções *(da ordem de milhares ou até milhões)*, precisa-se de métodos alternativos, com complexidade mais baixa. Com esse objetivo, surge a brilhante estratégia de resolver o sistema linear $Ax = b$ através de um problema de otimização. Vamos definir a função

$$f(x) = \frac{x \cdot Ax}{2} - b \cdot x + c. \tag{2.1}$$

O próximo passo é calcular o gradiente dessa função, para isso precisamos calcular sua derivada com relação a cada variável. Ou seja, sendo $x = (x_1, x_2, \cdots, x_n)$, vamos calcular $\dfrac{\partial f}{\partial x_l}$, para $l = 1, 2, \cdots, n$.

Vamos começar os cálculos desenvolvendo o produto $x \cdot Ax$.

$$\begin{aligned} x \cdot Ax &= \sum_{p=1}^{n} x_p \left(\sum_{k=1}^{n} a_{pk} x_k \right) \\ &= \sum_{p=1}^{n} \sum_{k=1}^{n} a_{pk} x_p x_k \\ &= \sum_{p\neq l}^{n} \sum_{k=1}^{n} a_{pk} x_p x_k + \sum_{k=1}^{n} a_{lk} x_l x_k \\ &= \sum_{p\neq l}^{n} \sum_{k=1}^{n} a_{pk} x_p x_k + \sum_{k\neq 1}^{n} a_{lk} x_l x_k + a_{lk} x_l^2 \end{aligned}$$

Sua derivada parcial com respeito a variável x_l é calculada a seguir.

$$\begin{aligned}\frac{\partial(x \cdot Ax)}{\partial x_l} &= \sum_{p \neq l}^{n} x_p a_{pl} + \sum_{k=1}^{n} a_{lk} x_k + a_{ll} x_l \\ &= \sum_{p=1}^{n} x_p a_{pl} + \sum_{k=1}^{n} a_{lk} x_k \\ &= \sum_{r=1}^{n} x_r a_{rl} + a_{lr} x_r \\ &= \sum_{r=1}^{n} (a_{rl} + a_{lr}) x_r.\end{aligned}$$

A derivada parcial do fator $b \cdot x$ em relação a x_l é obviamente b, assim concluímos que

$$\nabla f(x) = \frac{A + A^T}{2} x - b.$$

Adicionando a hipótese da matriz A ser simétrica, ficamos com

$$\nabla f(x) = Ax - b.$$

Sendo assim, se encontrarmos um valor x^* que anula ∇f, encontramos a solução do sistema linear. Os métodos de otimização para resolver sistemas lineares consistem então em buscar um mínimo ou máximo x_0 para a função $f(x)$, pois sendo assim $\nabla f(x_0) = 0$ e portanto $Ax_0 = b$. Porém, para que se garanta que o método de otimização convirja é preciso que se saiba se, de fato, a função em questão possui um máximo ou mínimo. Essa garantia é obtida quando se impõe que a matriz A seja negativa definida [1] (veja figura 2.2) ou positiva definida [2] respectivamente (veja figura 2.1). Não há perda de generalidade em supor que A é positiva definida, pois se a matriz A por negativa definida, basta definir uma matriz C como $C = -A$ e resolver o sistema linear $Cy = b$. Assim, como a matriz C será positiva definida é possível resolver o problema por otimização e solução do sistema original $Ax = b$ será $x = -y$. Com isso, vamos considerar como hipótese para o método dos gradientes conjugados que A é simétrica e positiva definida.

Para garantir a existência de um mínimo para esse problema, é preciso analisar a matriz Hessiana - $\mathcal{H}$.

[1] Dizemos que uma matriz A, $n \times n$, é uma matriz **negativa definida** quando para todo vetor não nulo $v \in \mathbb{R}^n$ vale que $v \cdot Av < 0$.

[2] Dizemos que uma matriz A, $n \times n$, é uma matriz **positiva definida** quando para todo vetor não nulo $v \in \mathbb{R}^n$ vale que $v \cdot Av > 0$.

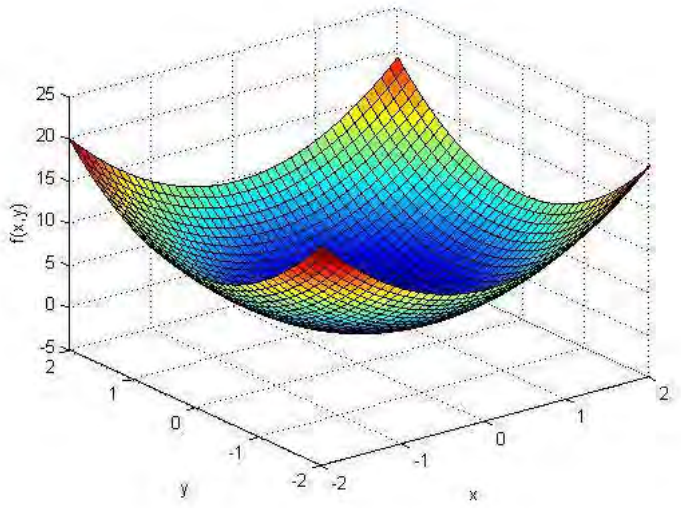

Figura 2.1: Gráfico de uma função $f(x)$ com A uma matriz positiva definida: $f(x,y) = 5x^2 + 5y^2 - 0.5x - 0.5y$.

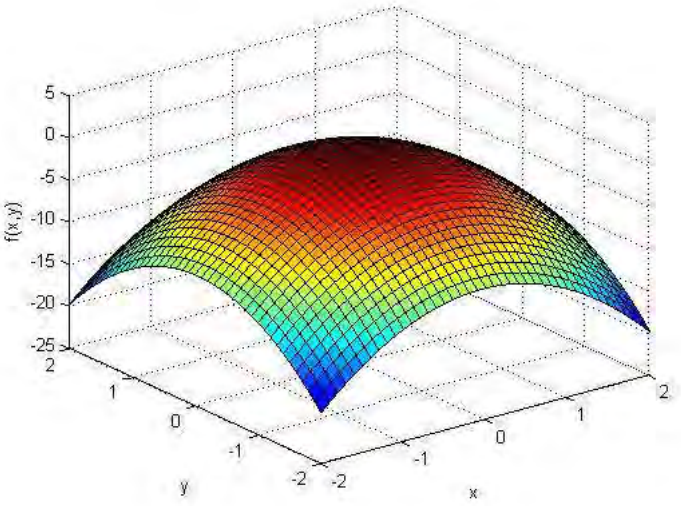

Figura 2.2: Gráfico de uma função $f(x)$ com A uma matriz negativa definida: $f(x,y) = -5x^2 - 5y^2 - 0.5x - 0.5y$.

Sabemos que $\mathcal{H}_{ij} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$. Logo, como

$$\frac{\partial f}{\partial x_i} = \sum_{k=1}^{n} a_{ik} x_i - b_i$$

temos que

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = a_{ij}.$$

Assim temos que

$$\mathcal{H}(x) = A.$$

Dessa forma, sabemos que um ponto crítico x para f sempre será um ponto de mínimo, já que A é positiva definida. O resultado que garante esse fato pode ser visto no Teorema 5, na página 157 de [Lima, 2008]. Uma maneira simples de tentar se resolver esse problema é buscar uma sequência de pontos $(x_k)_{k\in\mathbb{N}}$ tais que $f(x_{k+1}) < f(x_k)$. Para a construção dessa sequência considera-se uma direção de busca d_k, um passo α_k e forma-se o iterando

$$x_{k+1} = x_k + \alpha_k d_k. \tag{2.2}$$

A partir desse momento devemos analisar cuidadosamente como deve ser feita a escolha para α_k e d_k de modo a se obter $f(x_{k+1}) < f(x_k)$.

Definição 2.1. Dizemos que d_k é **direção de descida** da função $f : \mathbb{R}^n \to \mathbb{R}$ a partir do ponto x_k, se existe $\delta > 0$ tal que

$$f(x_k + \alpha d_k) \leq f(x_k), \quad \text{sempre que } \alpha \in (0, \delta).$$

Como a derivada direcional é a taxa de variação de uma função sobre uma direção, podemos reformular o conceito de direção de descida:

Teorema 2.2. Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função cuja derivada direcional no ponto x_k na direção d_k exista. Então d_k é uma direção de descida para f, a partir de x_k, se, e somente se, $\frac{\partial f}{\partial d_k}(x_k) \leq 0$.

Demonstração. Como a derivada direcional de f no ponto x_k, na direção de d_k existe, podemos escrever

$$\frac{\partial f}{\partial d_k}(x_k) = \lim_{t\to 0} \frac{f(x_k + td_k) - f(x_k)}{t} = \lim_{t\to 0^+} \frac{f(x_k + td_k) - f(x_k)}{t}.$$

Daí, $\frac{\partial f}{\partial d_k}(x_k) \leq 0$ se, e somente se, existe $\delta > 0$ tal que $f(x_k + td_k) \leq f(x_k)$ sempre que $t \in (0, \delta)$, que é justamente a definição de direção de descida. □

Usando o fato de que, sendo f diferenciável,

$$\frac{\partial f}{\partial d_k}(x_k) = \nabla f(x_k) \cdot d_k,$$

temos a seguinte caracterização para as direções de descida de funções diferenciáveis:

Teorema 2.3. Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função diferenciável. Então d_k é uma direção de descida para f, a partir de x_k, se, e somente se, $\nabla f(x_k) \cdot d_k \leq 0$.

Uma consequência imediata do teorema 2.3 é que $-\nabla f(x_k)$ é uma direção de descida, já que

$$\nabla f(x_k) \cdot (-\nabla f(x_k)) = -||\nabla f(x_k)||^2 \leq 0.$$

No **Método do Gradiente** (ou declive máximo - *steepest descent*) é utilizado $d_k = -\nabla f(x_k)$ como direção de descida. Agora é preciso definir o passo α_k.Temos que α_k deve ser mínimo da função $\varphi : \mathbb{R} \to \mathbb{R}$ dada por $\varphi(\alpha) = f(x_k + \alpha d_k)$. Derivando φ temos,

$$\varphi'(\alpha) = (\nabla f(x_k + \alpha d_k)) \cdot d_k. \tag{2.3}$$

Como $\nabla f(x) = Ax - b$, vemos que

$$\begin{aligned} \nabla f(x_k + \alpha d_k) &= A(x_k + \alpha d_k) - b \\ &= Ax_k - b + \alpha A d_k \\ &= \nabla f(x_k) + \alpha A d_k. \end{aligned} \tag{2.4}$$

Usando a equação 2.4 na expressão 2.3 e fazendo $\varphi'(\alpha) = 0$ obtemos

$$(\nabla f(x_k) + \alpha A d_k) \cdot d_k = 0,$$

de onde segue que o α_k candidato ao mínimo é

$$\alpha_k = -\frac{\nabla f(x_k) \cdot d_k}{d_k \cdot A d_k} \tag{2.5}$$

$$= \frac{d_k \cdot d_k}{d_k \cdot A d_k} \tag{2.6}$$

Resta verificar que de fato, este α_k minimiza φ. Para isso precisamos calcular a derivada segunda de φ.

$$\begin{aligned} \varphi''(\alpha) &= ((\nabla f(x_k + \alpha d_k)) \cdot d_k)' \\ &= ((A(x_k + \alpha d_k) - b) \cdot d_k)' \\ &= ((x_k + \alpha d_k) \cdot A d_k - b \cdot d_k)' \\ &= d_k \cdot A d_k > 0, \end{aligned}$$

já que A é positiva definida. Assim temos α_k de fato é mínimo de φ.

O próximo método é conhecido como **Método das Direções Conjugadas**. Vamos então definir direções conjugadas.

Definição 2.4. Dizemos que um conjunto de vetores $\{d_0, d_1, \cdots, d_{n-1}\}$ de $\mathbb{R}^n$ é **A-ortogonal** ou que os vetores são **conjugados** se

$$d_i \cdot Ad_j = 0,$$

para todo $i, j = 0, \cdots, n-1$ com $i \neq j$.

Note que se usarmos nessa definição a matriz identidade I recaímos no conceito de ortogonalidade.

Na iteração 2.2 vamos usar uma base $\{d_0, d_1, \cdots, d_{n-1}\}$ de direções A-ortogonais de $\mathbb{R}^n$. A construção do passo α_k é feita do mesmo modo que no Método do Gradiente, onde obtemos

$$\alpha_k = -\frac{\nabla f(x_k) \cdot d_k}{d_k \cdot Ad_k}. \tag{2.7}$$

Teorema 2.5. Se $\{d_0, d_1, \cdots, d_{n-1}\}$ é um conjunto de direções A-ortogonais e α_k é como na equação 2.7, então para $k = 0, 1, \cdots, n-1$ temos que $\nabla f(x_{k+1})$ é ortogonal a d_k.

Demonstração. Calculando o produto escalar de $\nabla f(x_{k+1})$ por d_k obtemos

$$\nabla f(x_{k+1}) \cdot d_k = \nabla f(x_k + \alpha_k d_k) \cdot d_k = \varphi'(\alpha_k) = 0,$$

o que prova o teorema. □

Teorema 2.6. Um método de direções conjugadas converge para a solução x^* em no máximo n iterações.

Demonstração. A demonstração desse teorema pode ser vista em [Nocedal and Wright, 2006] - teorema 5.1. □

Note que no método das direções conjugadas partimos de uma base de direções conjugadas, sem especificar quais são esses vetores. O **Método do Gradiente Conjugado Linear** é um caso particular do método das direções conjugadas, onde usamos um conjunto de direções conjugadas obtidas através do gradiente.

Fixemos então $x_0 \in \mathbb{R}^n$ e fazemos

$$d_0 = -\nabla f(x_0) \text{ e } d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k. \tag{2.8}$$

A escolha de β_k deve ser feita de modo que $\{d_0, d_1, \cdots, d_{n-1}\}$ seja A-conjugado. Como visto em [Borges, 1985], isso é equivalente a escolher β_k de modo que duas direções consecutivas sejam A-conjugadas. Para isso,

$$\begin{aligned} 0 &= d_k \cdot Ad_{k+1} \\ &= d_k \cdot A(-\nabla f(x_{k+1}) + \beta_k d_k) \\ &= -d_k \cdot A\nabla f(x_{k+1}) + d_k \cdot A\beta_k d_k. \end{aligned}$$

De onde segue que

$$\beta_k = \frac{d_k \cdot A\nabla f(x_{k+1})}{d_k \cdot Ad_k}. \tag{2.9}$$

Algoritmo 1: Gradiente Conjugado Linear

Entrada: $f, \nabla f, x_0, tol$ { campo escalar, gradiente do campo, estimativa inicial e tolerância para o critério de parada}

Saída: vetor $\vec{x}$ que minimiza o campo escalar

início
  $x := x_0$;
  $d_0 := -\nabla f(x_0)$;
  $d := d_0$;
  enquanto $||\nabla f(x)|| > tol$ **faça**
    $\alpha := -\dfrac{\nabla f(x) d}{d \cdot Ad}$;
    $x := x + \alpha d$;
    $\beta := \dfrac{d \cdot A\nabla f(x)}{d \cdot Ad}$;
    $d := -\nabla f(x) + \beta d$;
  fim
fim
retorna x

2.2 O Gradiente Conjugado Não Linear

O método do Gradiente Conjugado descrito na seção anterior é um método desenvolvido para resolver sistemas lineares $Ax = b$, recaindo na minimização de funções quadráticas. Com a inspiração na técnica para minimização no Gradiente Conjugado é possível criar uma coleção de métodos, que chamamos de Gradiente Conjugado Não Linear (NLCG), para a minimização irrestrita de funções gerais.

Mais precisamente, considerando o problema de minimizar um campo escalar $f(x)$ irrestritamente, vamos definir iterativamente uma sequência de pontos x_k dada por:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.10}$$

onde

$$d_0 = -\nabla f(x_0) \text{ e } d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k. \tag{2.11}$$

Os passos α_k são obtido através de uma busca unidirecional. Esta metodologia iterativa é chamada de Gradiente Conjugado Não Linear se, ao tomarmos f como sendo a função quadrática definida em 2.1, recaímos no Gradiente Conjugado Linear, com β_k dado pela fórmula 2.9.

Existem várias fórmulas conhecidas para β_k. Essas fórmulas, em geral, não são equivalentes. As mais populares, e as que iremos tratar nesse trabalho, são as fórmulas de Fletcher-Reeves e Polak-Ribiere. Ao contrário do método do Gradiente Conjugado Linear, que possui propriedades de convergência bem conhecidas, os métodos não lineares podem possuir comportamento bizarro, inclusive, muitas vezes, os algoritmos não produzem sequências convergindo para o mínimo. Algumas vezes o NLCG na formulação Fletcher-Reeves é tão eficiente quanto na formulação Polak-Ribiére mas, na grande maioria das vezes, é mais lento segundo [Gilbert and Nocedal, 1992]. Porém, a convergência global do método usando buscas unidimensionais exatas, foi garantida em [Zoutendijk, 1970]. [Al-Baali, 1985] também provou a convergência global desse método com buscas unidimensionais inexatas, porém sob certas condições que serão vistas na seção 2.2.2. Por convergência global queremos dizer que

$$\lim_{k\to\infty} ||\nabla f(x_k + \alpha_k d_k)|| = 0.$$

Na prática, NLCG com a formulação Polak-Ribiere é mais utilizado que NLCG com Fletcher-Reeves, no entanto [Powell, 1984] mostrou que esse método pode entrar em loop infinito caso utilizado com buscas unidimensionais exatas. Existem várias modificações no método de Polak-Ribiere que garantem sua convergência global. [Zhang et al., 2012] ilustra algumas delas.

2.2.1 Busca Unidirecional Exata

O termo busca unidirecional exata não possui um significado único na literatura. Um significado é que busca unidirecional exata é feita de forma direta, se encontrando o mínimo da função $\varphi(\alpha) = f(x_k + \alpha d_k)$. Um outro significado, mais fraco, é que na busca unidirecional exata encontra-se um α tal que $\varphi'(\alpha) = 0$. Que, pela regra da cadeia, temos $\varphi'(\alpha) = \nabla f(x_k + \alpha d_k) \cdot d_k = 0$. Portanto, neste significado, a busca unidirecional exata determina um passo α tal que $\nabla f(x_{k+1})$ é perpendicular ao vetor d_k. Em qualquer um desses dois significados, estabelecemos a seguinte proposição:

Proposição 2.7. O NLCG com busca direcional exata sempre gera direções de descida.

Demonstração. Admitamos que em cada iteração é possível obter o passo α com uma busca exata. Fixemos uma iteração k. Então, temos que

$$d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k.$$

Tomando o produto escalar por $\nabla f(x_{k+1})$ obtemos que

$$\nabla f(x_{k+1}) \cdot d_{k+1} = -||\nabla f(x_{k+1})||^2 + \beta_k \nabla f(x_{k+1}) \cdot d_k.$$

Como $\nabla f(x_{k+1}) \cdot d_k = 0$, segue que

$$\nabla f(x_{k+1}) \cdot d_{k+1} = -||\nabla f(x_{k+1})||^2 < 0,$$

portanto, d_{k+1} é direção de descida pra f em x_{k+1}. □

Particularmente, neste trabalho, ao usar essa estratégia para o funcional obtido no método DMTB para o silício, recaímos em achar o mínimo de uma função polinomial de grau 3. Então, obter os pontos críticos, caso existam, no nosso caso é uma tarefa trivial e gera uma fórmula fechada para ser usada em todas as iterações.

No entanto, os pontos críticos podem não existir ou o teste da derivada segunda para a caracterização do mínimo pode ser inconclusivo. Na seção 2.3 apresentaremos explicitamente o valor obtido para α_k na busca linear exata. Em geral, mesmo quando é possível se determinar o mínimo da função sobre a semi reta a partir de x_k na direção de descida d_k, por requerer um processo de minimização unidimensional em cada iteração, a busca exata pode ser bastante ineficiente em problemas de larga escala. Por esse motivo, na

prática, a maioria das implementações computacionais de buscas lineares abrem mão da precisão, usando as chamadas buscas lineares inexatas.

[Millam and Scuseria, 1997] em seu trabalho com Carbono, usou somente busca linear exata ($\nabla f(x_{k+1}) \cdot d_k = 0$). Ao longo de nossos experimentos numéricos com Silício, observamos que nem sempre é possível usar busca exata, pois o nosso funcional em questão restrito as semiretas nem sempre possui pontos críticos. Sendo assim, nesse trabalho, optamos por utilizar uma abordagem mista. Ou seja, ora usando busca liner exata quando possível e inexata nos demais casos.

Conforme comentamos anteriormente, para encontrarmos os pontos críticos de φ nesse trabalho, precisamos resolver $a\alpha^2 + b\alpha + c = 0$, o que é trivial do ponto de vista algébrico: basta usar a fórmula resolvente para equações polinomiais de grau 2. Porém, caso nessa equação tenhamos b^2 muito maior do que $4ac$, teremos um grave problema ao usar a fórmula resolvente, já que $b^2 - 4ac \approx b^2$. Por consequência, teríamos que uma das raízes seria nula, o que não é verdade. Esse fenômeno numérico é conhecido como *cancelamento subtrativo*, e existem algumas propostas na literatura para contornar esse problema.

Aqui usaremos a seguinte estratégia:

$$\alpha_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad \text{e} \quad \alpha_2 = \frac{c}{a\alpha_1} \quad \text{caso } b \geq 0$$

ou

$$\alpha_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{e} \quad \alpha_2 = \frac{c}{a\alpha_1} \quad \text{caso } b < 0.$$

Observe que sendo $b^2 - 4ac \approx b^2$ temos que $\sqrt{b^2 - 4ac} \approx |b|$. Assim, temos que $\alpha_1 \approx \dfrac{-2b}{2a}$ e $\alpha_2 = \dfrac{c}{a\alpha_1}$, mostrando que evitamos o fenômeno do cancelamento subtrativo.

2.2.2 Busca Unidirecional Inexata - As condições fortes de Wolfe

Existem muitas metodologias para buscas inexatas. Quando o cálculo de ∇f não é complicado, as chamadas "regras de Wolfe" são consideradas as mais eficientes estratégias de busca unidirecional. Apresentaremos a seguir as condições fortes de Wolfe, que são aplicadas para determinar o valor do passo α_k e essenciais em muitos teoremas de convergência global do NLCG.

A primeira Condição de Wolfe, também chamada de ***Regra de Armijo*** consiste em uma busca unidirecional inexata para determinar o comprimento do passo α_k de modo que se consiga obter um decréscimo

suficiente na função f a partir de x_k na direção d_k. Mais precisamente, vamos considerar um ponto $x_k \in \mathbb{R}^n$, uma direção de descida d_k e $c_1 \in (0,1)$. A Regra de Armijo consiste em encontrar α_k tal que

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k) \cdot d_k, \tag{2.12}$$

Geometricamente, a Regra de Armijo diz que somente são aceitáveis os valores de α_k aqueles cujo $f(x_k + \alpha_k d_k)$ fiquem abaixo da reta $l(\alpha_k) = f(x_k) + c_1 \alpha_k \nabla f(x_k) \cdot d_k$. Para referência, observe que $\nabla f(x_k) \cdot d_k$ é o coeficiente angular da reta tangente ao gráfico de $\varphi(\alpha_k) = f(x_k + \alpha_k d_k)$ em $\alpha_k = 0$. Como d_k é direção de descida e $c_1 \in (0,1)$, temos que $\nabla f(x_k) \cdot d_k < c_1 \nabla f(x_k) \cdot d_k$. A figura 2.3 ilustra essa situação.

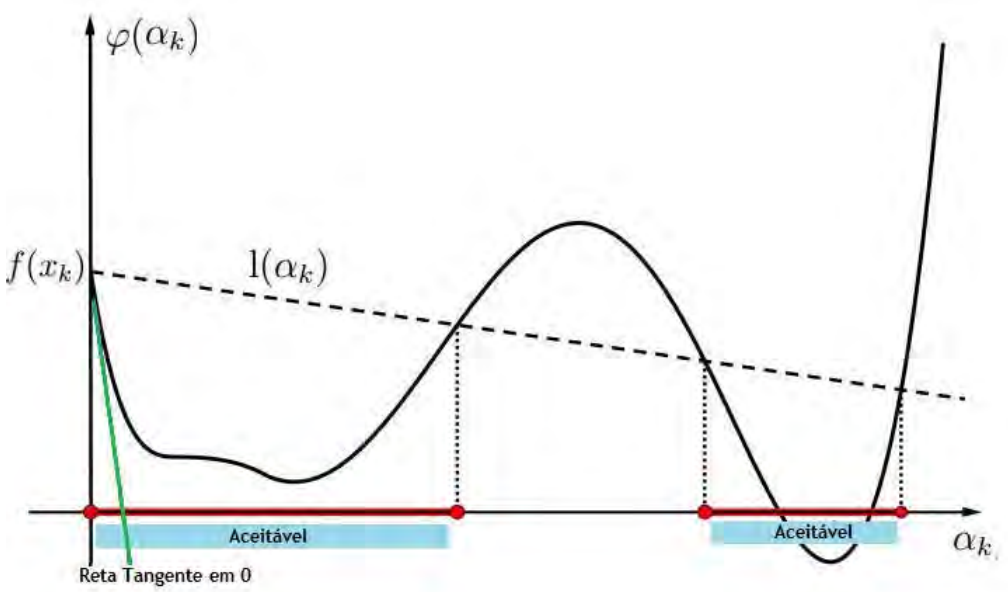

Figura 2.3: Valores aceitáveis para α_k na Condição de Armijo

O teorema a seguir garante que existem intervalos de comprimentos de passos α_k que satisfazem a Regra Armijo.

Teorema 2.8. Considere uma função diferenciável $f : \mathbb{R}^n \to \mathbb{R}$, um ponto $x_k \in \mathbb{R}^n$, uma direção de descida $d_k \in \mathbb{R}^n$ e $c_1 \in (0,1)$. Então existe $\delta > 0$ tal que

$$f(x_k + t d_k) \leq f(x_k) + c_1 t \nabla f(x_k) \cdot d_k,$$

para todo $t \in (0, \delta)$.

Demonstração. Como d_k é uma direção de descida temos que $\nabla f(x_k) \cdot d_k \leq 0$. No caso $\nabla f(x_k) \cdot d_k = 0$ o resultado segue diretamente da definição de direção de descida. Suponhamos que $\nabla f(x_k) \cdot d_k < 0$. Temos que

$$\nabla f(x_k) \cdot d_k = \frac{\partial f}{\partial d_k}(x_k) = \lim_{t \to 0} \frac{f(x_k + td_k) - f(x_k)}{t}.$$

E como $c_1 \in (0, 1)$, vemos que

$$\nabla f(x_k) \cdot d_k < c_1 \nabla f(x_k) \cdot d_k.$$

Logo existe $\delta > 0$ tal que, no intervalo $(0, \delta)$ seja válido que

$$\frac{f(x_k + td_k) - f(x_k)}{t} \leq c_1 \nabla f(x_k) \cdot d_k.$$

Portanto

$$f(x_k + td_k) \leq f(x_k) + c_1 t \nabla f(x_k) \cdot d_k,$$

sempre que $t \in (0, \delta)$, como desejávamos provar. □

Vemos então que podemos escolher α_k como sendo qualquer número do intervalo $(0, \delta)$. Então, podemos começar com um α_k inicial qualquer e formarmos a sequência $\theta^n \alpha_k$, onde θ é um parâmetro fixo tomado no intervalo $(0, 1)$. Desta forma, a sequência converge para zero, garantindo que, em algum momento, estaremos no intervalo $(0, \delta)$, independente do valor desconhecido de δ. O que faremos é tomar o primeiro momento onde isso ocorre. Veja o algoritmo 2, que é conhecido com algoritmo de *backtracking*.

Algoritmo 2: Regra de Armijo

Entrada: $f, \nabla f, x, d, c_1, \theta, a$ { campo escalar, gradiente do campo, estimativa, direção de descida, parâmetro de Armijo - $c_1 \in (0, 1)$, fator de decaimento - $\theta \in (0, 1)$, valor inicial do passo}

Saída: o passo ajustado pela Regra de Armijo

```
início
    enquanto f(x + αd) > f(x) + c1 α ∇f(x) · d faça
        α = θα;
    fim
fim
retorna a
```

Como dissemos, no algoritmo o que estamos fazendo é diminuir α_k até que possamos chegar nas condição de teorema 2.8, então poderíamos tomar qualquer fator multiplicativo entre 0 e 1, como, por exemplo, $\theta = 0,8$.

Somente a condição de decrescimento dada pela Condição de Armijo, em geral, não é suficiente para a convergência do NLCG. E mesmo na presença da convergência global, pode-se gerar valores de passo α_k extremamente pequenos, o que compromete a eficiência do algoritmo. Além disso, para garantir a eficácia do método, é preciso ter, de fato, direções de descida d_k.

Lembremos que

$$\nabla f(x_{k+1}) \cdot d_{k+1} = -||\nabla f(x_{k+1})||^2 + \beta_k \nabla f(x_{k+1}) \cdot d_k.$$

Portanto, dependendo dos valores de β_k , é possível que $\nabla f(x_{k+1}) \cdot d_{k+1} \geq 0$. Sendo assim é necessário impor mais um condição, dessa vez envolvendo $\nabla f(x_{k+1}) \cdot d_{k+1}$. A condição que apresentaremos a seguir é chamada de **Condição de Curvatura**.

$$|\nabla f(x_k + \alpha_k d_k) \cdot d_k| \leq -c_2 \nabla f(x_k) \cdot d_k, \tag{2.13}$$

com $0 < c_1 < c_2 < \frac{1}{2}$, sendo c_1 a constante da Condição de Armijo, equação 2.12.

Podemos reescrever 2.13 como:

$$\nabla f(x_k + \alpha_k d_k) \cdot d_k \leq -c_2 \nabla f(x_k) \cdot d_k \tag{2.14}$$

$$\nabla f(x_k + \alpha_k d_k) \cdot d_k \geq c_2 \nabla f(x_k) \cdot d_k \tag{2.15}$$

As duas desigualdades limitam os valores de α_k aceitáveis através da comparação da inclinação das retas tangentes ao gráfico de $\varphi(\alpha_k) = \nabla f(x_k + \alpha_k d_k)$, com um múltiplo do coeficiente angular da reta tangente ao gráfico de $\varphi(\alpha_k)$ em $\alpha_k = 0$. No gráfico 2.4 estão destacados em vermelho os valores para α_k que satisfazem a condição de curvatura.

A Condição de Armijo, equação 2.12, em conjunto com a Condição de Curvatura, equação 2.13, formam as chamadas ***Condições Fortes de Wolfe***.

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k) \cdot d_k$$

$$|\nabla f(x_k + \alpha_k d_k) \cdot d_k| \geq -c_2 \nabla f(x_k) \cdot d_k$$

com $0 < c_1 < c_2 < \frac{1}{2}$.

Na figura 2.5 unimos a representação geométrica da Condição de Armijo com a Condição de Curvatura. Observe que a inclusão da Condição de Armijo tende a descartar valores de α_k que estão próximos de máximos locais.

Algoritmo 3: Condições Fortes de Wolfe

Entrada: $f, \nabla f, x_k, d_k, c_1, c_2, \alpha_0, \alpha_{max}$ {campo escalar, gradiente do campo, estimativa, direção de descida, parâmetro de Armijo - $c_1 \in (0, \frac{1}{2})$, parâmetro de Wolfe - $c_2 \in (c_1, \frac{1}{2})$, valor inicial do passo, valor máximo pra o passo}

Saída: o passo ajustado pelas condições fortes de Wolfe

início
  $\alpha_0 := 0$;
  Escolha $\alpha_1 \in (0, \alpha_{max})$;
  enquanto *Não encontrar* α **faça**
    Calcule $f(x_k + \alpha_i d_k)$;
    se $f(x_k + \alpha_i d_k) > f(x_k) + c_1 \alpha_i \nabla f(x_k) \cdot d_k$ *ou* $[|f(x_k + \alpha_i d_k)| \geq |f(x_k + \alpha_{i-1} d_k)|$ *e* $i > 1]$ **então**
      Obtenha α através da rotina zoom(α_{i-1}, α_i);
    senão
      Calcule $\nabla f(x_k + \alpha_i d_k) \cdot d_k$;
      se $|\nabla f(x_k + \alpha_i d_k) \cdot d_k| \leq -c_2 \nabla f(x_k) \cdot d_k$ **então**
        Faça $\alpha = \alpha_i$;
      senão
        se $\nabla f(x_k + \alpha_i d_k) \cdot d_k \geq 0$ **então**
          Obtenha α através da rotina zoom(α_i, α_{i-1});
        senão
          Faça $\alpha_{i+1} = \dfrac{\alpha_i + \alpha_{max}}{2}$;
          i=i+1;
        fim
      fim
    fim
  fim
fim
retorna α

Algoritmo 4: Zoom

Entrada: $f, \nabla f, x_k, d_k, c_1, c_2, \alpha_l, \alpha_u$ {campo escalar, gradiente do campo, estimativa, direção de descida, parâmetro de Armijo - $c_1 \in (0, \frac{1}{2})$, parâmetro de Wolfe - $c_2 \in (c_1, \frac{1}{2})$, valores que limitam α}

Saída: o passo desejado

início
  Faça $\alpha_j = \dfrac{\alpha_l + \alpha_u}{2}$;
  enquanto *Não encontrar* α **faça**
    Calcule $f(x_k + \alpha_j d_k)$;
    se $f(x_k + \alpha_j d_k) > f(x_k) + c_1 \alpha_j \nabla f(x_k) \cdot d_k$ *ou* $f(x_k + \alpha_j d_k) \geq f(x_k + \alpha_l d_k)$ **então**
      $\alpha_u = \alpha_j$;
    senão
      Calcule $\nabla f(x_k + \alpha_j d_k) \cdot d_k$;
      se $|\nabla f(x_k + \alpha_j d_k) \cdot d_k| \leq -c_2 \nabla f(x_k) \cdot d_k$ **então**
        Faça $\alpha = \alpha_j$;
      senão
        se $\nabla f(x_k + \alpha_j d_k) \cdot d_k (\alpha_u - \alpha_l) \geq 0$ **então**
          $\alpha_u = \alpha_l$;
          $\alpha_l = \alpha_j$;
        senão
      fim
    fim
  fim
fim
retorna α

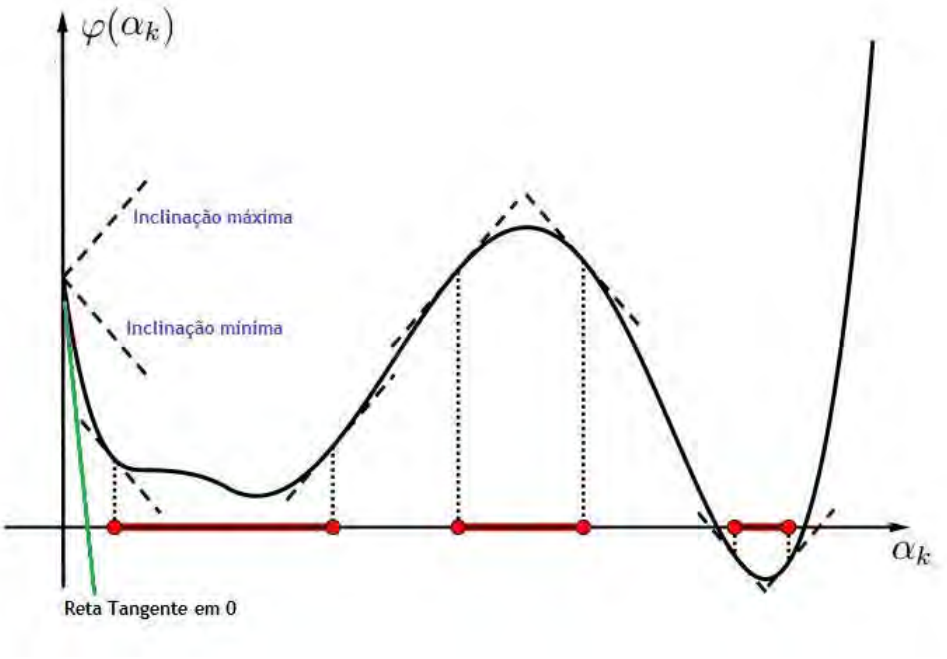

Figura 2.4: Valores de α_k que satisfazem a Condição de Curvatura

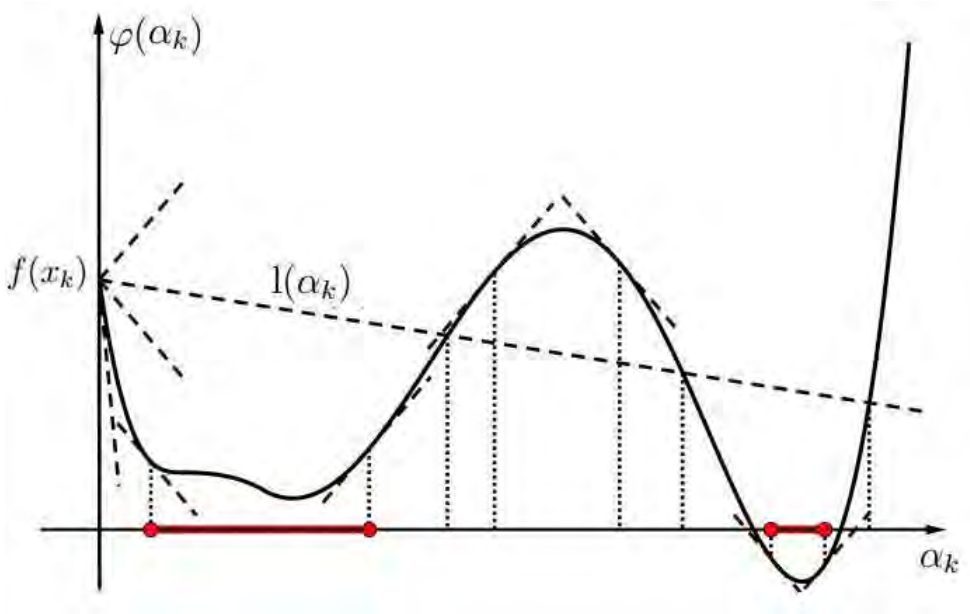

Figura 2.5: Valores de α_k que satisfazem as Condições Fortes de Wolfe

O lema 3.1 de [Nocedal and Wright, 2006] garante a existência de um passo α_k satisfazendo as condições fortes de Wolfe. No algoritmo 3 apresentamos uma possível implementação para as mesmas. A rotina *zoom* presente no algoritmo 3 está detalhada no algoritmo 4. Esse algoritmo é construído de maneira a for-

mar uma sequência monótona α_i. Então dois casos podem acontecer, ou o algoritmo segue até encontrar um α aceitável ou entra na rotina *zoom* se encontrar um intervalo onde o α possa ser detectado. No *zoom*, caso α_j não satisfaça a Condição de Armijo, ou tenha o decaimento desejado, ou ainda a função passe a ser crescente a partir desse comprimento de passo, o intervalo que contém α é reduzido continuamente, até que um α aceitável seja encontrado. O teorema a seguir garante a convergência do algoritmo 3.

Teorema 2.9. Se $0 < c_1 < c_2 < \frac{1}{2}$ então o algoritmo 3 encontra α após um número finito de iterações.

Demonstração. A demonstração desse resultado pode ser encontrada em [Al-Baali and Fletcher, 1986]. □

Como falamos anteriormente, existe uma coleção de métodos que chamamos de Gradiente Conjugado Não Linear. As variações estão na busca do passo α_k e também na escrita de β_k. Discutimos anteriormente as estratégias que usamos nesse trabalho para encontrar α_k. Uma fórmula para β_k foi desenvolvida para o caso do Gradiente Conjugado Linear. Agora vamos reescrever essa fórmula de duas maneiras, encontrando a modificação de Fletcher-Reevers e a modificação de Polak-Ribiere, que são as versões que usamos nesse trabalho.

2.2.3 Modificação de Fletcher Reevers

A grande modificação proposta por Fletcher and Reevers, foi reescrever β_k de modo que este dependa somente da função f e não da matriz A.

Teorema 2.10. Seja β_k como na equação 2.9. Então

$$\beta_k = \frac{\nabla f(x_{k+1}) \cdot \nabla f(x_{k+1})}{\nabla f(x_k) \cdot \nabla f(x_k)}. \tag{2.16}$$

Demonstração. Vamos começar observando que da equação 2.4 temos que

$$Ad_k = \frac{\nabla f(x_k + \alpha_k d_k) - \nabla f(x_k)}{\alpha_k}. \tag{2.17}$$

Além disso, vemos que β_k como definido em 2.9 pode ser escrito como

$$\beta_k = \frac{d_k \cdot A\nabla f(x_{k+1})}{d_k \cdot Ad_k} = \frac{(\nabla f(x_{k+1}) \cdot (Ad_k))^T}{d_k \cdot Ad_k}.$$

Usando 2.17 nessa última equação, obtemos

$$\beta_k = \frac{\left(\nabla f(x_{k+1}) \cdot \left(\frac{\nabla f(x_k + \alpha_k d_k) - \nabla f(x_k)}{\alpha_k}\right)\right)^T}{d_k \left(\frac{\nabla f(x_k + \alpha_k d_k) - \nabla f(x_k)}{\alpha_k}\right)},$$

de onde segue que

$$\beta_k = \frac{(\nabla f(x_{k+1}) \cdot (\nabla f(x_{k+1}) - \nabla f(x_k)))^T}{d_k \cdot (\nabla f(x_{k+1}) - \nabla f(x_k))}.$$

Usando o teorema 2.5, ficamos com

$$\beta_k = \frac{\nabla f(x_{k+1}) \cdot \nabla f(x_{k+1}) - \nabla f(x_{k+1}) \cdot \nabla f(x_k)}{-d_k \cdot \nabla f(x_k)}.$$

De acordo com [Borges, 1985], no caso do método do Gradiente Conjugado Linear, temos que $\nabla f(x_{k+1}) \cdot \nabla f(x_j) = 0$ para $j = 0, 1, \cdots, k$. Usando esse resultado, temos em particular que $\nabla f(x_{k+1}) \cdot \nabla f(x_k) = 0$. Assim, para concluir a demonstração do teorema, basta provarmos que $-d_k \cdot \nabla f(x_k) = \nabla f(x_k) \cdot \nabla f(x_k)$.

Note que, usando a definição de d_k, ficamos com

$$-d_k \cdot \nabla f(x_k) = -(-\nabla f(x_k) + \beta_{k-1} d_{k-1}) \cdot \nabla f(x_k) = \nabla f(x_k) \cdot \nabla f(x_k) - \beta_{k-1} d_{k-1} \cdot \nabla f(x_k).$$

Como d_{k-1} e $\nabla f(x_k)$ são ortogonais pelo teorema 2.5, concluímos que

$$-d_k \cdot \nabla f(x_k) = \nabla f(x_k) \cdot \nabla f(x_k).$$

Assim,

$$\beta_k = \frac{\nabla f(x_{k+1}) \cdot \nabla f(x_{k+1})}{\nabla f(x_k) \cdot \nabla f(x_k)},$$

como desejávamos. □

A partir de agora, o β_k escrito como no teorema 2.10 será denotado por β_k^{FR}.

Em [Zoutendijk, 1970] foi provado que a modificação de Fletcher-Reevers converge com buscas exatas. Para garantir a convergência global do NLCG com Fletcher-Reevers, usando busca inexata, foi preciso incluir as condições fortes de Wolfe. Com elas é possível provar que vale a condição de decrescimento suficiente, dada na equação 2.18.

$$\nabla f(x_k) \cdot d_k \leq -c||\nabla f(x_k)||^2, \quad c > 0 \tag{2.18}$$

Devido a este fato, [Al-Baali, 1985] garantiu a convergência global do método. A seguir vemos o teorema que garante a convergência. Uma demonstração desse teorema pode ser encontrada em [Gilbert and Nocedal, 1992

Teorema 2.11. Suponha que $\mathcal{L} = \{x | f(x) \leq f(x_1)\}$ é limitado e que em alguma vizinhança $\mathcal{N}$ de $\mathcal{L}$ a função f é continuamente derivável e seu gradiente é uma função de Lipschitz contínua [o que significa que existe uma constante $L > 0$ de modo que

$$||\nabla f(x) - \nabla f(\bar{x})|| \leq ||x - \bar{x}||,$$

para todo $x, \bar{x} \in \mathcal{N}$]. Considere a iteração dada pelas equações 2.10 - 2.11, de modo que $|\beta_k| \leq \beta_k^{FR}$ e onde o passo α_k satisfaz as condições fortes de Wolfe com $0 < c_1 < c_2 < \frac{1}{2}$. Então

$$\lim_{k \to +\infty} ||\nabla f(x_k)|| = 0.$$

No algoritmo 5 temos a implementação do Método de Fletcher Reevers.

Algoritmo 5: Gradiente Conjugado Não-Linear - Método de Fletcher Reevers

Entrada: $f, \nabla f, x_0$, *tol*, it_{max} { campo escalar, gradiente do campo, estimativa inicial, tolerância para o critério de parada e número máximo de iterações}

Saída: vetor x que minimiza o campo escalar

início
- $it := 0$,
- $x := x_0$;
- $d_0 := -\nabla f(x_0)$;
- $d := d_0$;
- $G := d_0$;
- **enquanto** $||G|| > tol$ *ou* $it < it_{max}$ **faça**
  - $G_0 := \nabla f(x_0)$;
  - Encontre α
  - $x := x + \alpha d$;
  - $G = \nabla f(x)$;
  - $b^{FR} = \dfrac{G \cdot G}{G_0 \cdot G_0}$;
  - $d := -\nabla f(x) + \beta^{FR} d$;
  - $x_0 := x$;
  - $it := it + 1$;
- **fim**

fim

retorna x

2.2.4 Modificação de Polak - Ribiere

Existem várias variantes para o método de Fletcher-Reevers fazendo somente modificações em β_k. Um das mais importantes é amodificação feita por Polak-Ribiere, que propõe uma outra escrita para β_k:

Teorema 2.12. Seja β_k como na equação 2.9. Então

$$\beta_k = \frac{\nabla f(x_{k+1}) \cdot (\nabla f(x_{k+1}) - \nabla f(x_k))}{\nabla f(x_k) \cdot \nabla f(x_k)}. \tag{2.19}$$

Demonstração. A prova desse teorema é idêntica a demonstração apresentada em 2.10, se diferenciando apenas por nesse caso não usar o ortogonalidade entre $\nabla f(x_{k+1})$ e $\nabla f(x_k)$. □

O β_k com a escrita como no teorema 2.12 passará a ser denotado por β_k^{PR}.

Quando a função que se deseja minimizar é não quadrática, porém fortemente convexa (veja [Ribeiro and Karas, 2011]), e se usa uma busca exata para se determinar α_k, os métodos de Fletcher-Reevers e Polak-Ribiere são idênticos. Porém, se usarmos busca inexata em funções não lineares mais gerais, pode-se observar mudanças significativas entre os dois métodos. Segundo [Nocedal and Wright, 2006], o método Polak-Ribiere costuma ser mais robusto que o método de Fletcher-Reevers. Porém, no método de Polak-Ribiere as condições fortes de Wolfe não garantem que d_k é uma direção de descida. Na verdade, existe muito a ser discutido sobre a convergência global do NLCG com Polak-Ribiere. Em [Zhang et al., 2012] podemos encontrar alguns resultados que garantem a convergência sob certas condições.

Uma proposta bastante presente na literatura, desenvolvida em [Powell, 1986], é a de modificar β, fazendo

$$\beta_k^+ = max\{\beta_k^{PR}, 0\}.$$

Como

$$\nabla f(x_k) \cdot d_k = -||\nabla f(x_k)||^2 + \beta_{k-1}^+ \nabla f(x_k) \cdot d_{k-1},$$

e $\beta_{k-1}^+ \geq 0$, se $\nabla f(x_k) \cdot d_{k-1} > 0$ podemos não gerar direções de descida. Dessa forma, é preciso impor a condição de decrescimento suficiente na busca linear - equação 2.18, com $0 < c \leq 1$ em todas as iterações. A implementação apresentada para as condições fortes de Wolfe no algoritmo 3 incorpora as estratégias propostas em [Moré and Thuente, 1994] para se garantir decrescimento suficiente durante a busca linear

inexata. O teorema 2.11 garante a convergência global do NLCG com essa proposta de β_k. Note que se estivermos com buscas exatas, temos $\nabla f(x_k) \cdot d_{k-1} = 0$, fazendo com que 2.18 seja satisfeita para $c = 1$, então também podemos garantir a convergência do método.

Algoritmo 6: Gradiente Conjugado Não-Linear - Método de Polak-Ribiere

Entrada: $f, \nabla f, x_0, tol, it_{max}$ { campo escalar, gradiente do campo, estimativa inicial, tolerância para o critério de parada e número máximo de iterações}

Saída: vetor x que minimiza o campo escalar

início
- $it := 0$,
- $x := x_0$;
- $d_0 := -\nabla f(x_0)$;
- $d := d_0$;
- $G := d_0$;
- **enquanto** $||G|| > tol$ *ou* $it < it_{max}$ **faça**
  - $G_0 := \nabla f(x_0)$;
  - Encontre a ; `/* Com busca unidimencional exata ou inexata conforme visto no algoritmo 3 */`
  - $x := x + ad$;
  - $G = \nabla f(x)$;
  - $b^{PR} = \dfrac{G \cdot (G - G_0)}{G_0 \cdot G_0}$;
  - **se** $b^{PR} < 0$ **então**
    - $b := 0$;
  - **senão**
    - $b = b^{PR}$;
  - **fim**
  - $d := -G + bd$;
  - $x_0 := x$;
  - $it := it + 1$;
- **fim**

fim

retorna x

2.3 O método DMTB com o Gradiente Conjugado Não-Linear

No fim do capítulo 1 chegamos a equação 1.10 que deverá ser minimizada a fim de se encontrar a energia do sistema. Optamos por usar o NLCG para fazer essa minimização por este método ter potencial para ser

um método de ordem de complexidade n, quando estivermos com sistemas de n átomos. Neste trabalho, o objetivo foi criar um laboratório virtual para o estudo do DMTB. Sendo assim, nos preocupamos em fazer inicialmente uma abordagem compreensível, e que possa ser facilmente ampliada para outros elementos que não o Silício, onde estamos fazendo nosso estudo de caso. A diminuição da ordem de complexidade do algoritmo será tratada em trabalhos futuros. Nessa seção mostramos como aplicar o NLGC no DMTB.

Inicialmente note que a função de desejamos minimizar é uma função matricial

$$\Omega(\rho) = tr(\tilde{\rho}H) + \mu(tr(\rho) - N_e I).$$

Para poder usar o método dos gradientes conjugados é preciso olhar para essa função de forma vetorial. Para isso podemos identificar uma matriz $n \times n$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

como o vetor em $\mathbb{R}^{n^2}$,

$$v_A = \begin{bmatrix} a_{11} \\ a_{12} \\ \cdots \\ a_{1n} \\ a_{21} \\ a_{22} \\ \cdots \\ a_{2n} \\ \cdots \\ \cdots \\ \cdots \\ a_{n1} \\ a_{n2} \\ \cdots \\ a_{nn} \end{bmatrix}.$$

Por exemplo, a matriz

$$A = \begin{bmatrix} 1 & 3 & -2 & 3 \\ -1 & 0 & 4 & -1 \\ 1 & 1 & -3 & 2 \end{bmatrix}$$

passa a ser escrita como o vetor

$$v_A = \begin{bmatrix} 1 \\ 3 \\ -2 \\ 3 \\ -1 \\ 0 \\ 4 \\ -1 \\ 1 \\ 1 \\ -3 \\ 2 \end{bmatrix}.$$

Assim nosso problema é minimizar

$$\Omega : \mathbb{R}^{n^2} \longrightarrow \mathbb{R}$$
$$\rho \longmapsto tr(\tilde{\rho}H) + \mu(tr(\rho) - N_e).$$

Claramente, $\nabla\Omega(\rho)$ também será uma função definida em $\mathbb{R}^{n^2}$. Porém, para evitar elevar os custos computacionais, vamos fazer o cálculo de $\nabla\Omega(\rho)$ ainda de forma matricial.

Admitindo a vetorização definida nas linhas acima, temos que:

$$\nabla\Omega(\rho) = 3(\rho H + H\rho)^T - 2(\rho^2 H + \rho H \rho + H\rho^2)^T + \mu I. \tag{2.20}$$

Para demonstração desse fato é necessária a dedução de regras para derivação de funções com domínio matricial, o que foge o objetivo dessa dissertação. No entanto, a demonstração detalhada dessa expressão pode ser encontrada em [Filho, 2017].

A partir daí é possível usar tanto a modificação de Fletcher-Reevers (algoritmo 5) quanto a modificação de Polak-Ribiere (algoritmo 6) para minimizar a função.

Nesse trabalho usamos uma abordagem mista para a determinação do passo α_k. Começamos com a estratégia da busca linear exata para sua determinação. Vamos então a determinação dos pontos críticos de

$\Omega(\rho+\alpha d)$, lembrando que estamos pensando em Ω como função de ρ. Antes vamos desenvolver $\Omega(\rho+\alpha d)$.

$$\begin{aligned}
\Omega(\rho+\alpha d) &= tr(3(\rho^2+\rho\alpha d+\alpha d\rho+\alpha^2 d^2)-2(\rho^3+\rho\alpha d\rho+\alpha d\rho^2+\alpha^2 d^2\rho+\rho^2\alpha d \\
&+ \rho\alpha^2 d^2+\alpha d\rho\alpha d+\alpha^3 d^3)H)+\mu(tr(\rho+\alpha d)-N_e) \\
&= 3tr(\rho^2 H)+3\alpha tr(\rho dH)+3\alpha tr(d\rho H)+3\alpha^2 tr(d^2 H)-2tr(\rho^3 H) \\
&- 2\alpha tr(\rho d\rho H)-2\alpha tr(d\rho^2 H)-2\alpha^2 tr(d^2\rho H)-2\alpha tr(\rho^2 dH) \\
&- 2\alpha^2 tr(\rho d^2 H)-2\alpha^2 tr(d\rho dH)-2\alpha^3 tr(d^3 H)+\mu tr(\rho)+\alpha\mu tr(d)-\mu N_e.
\end{aligned}$$

Organizando os fatores podemos escrever

$$\Omega(\rho+\alpha d)=\mathbf{A}+\mathbf{B}\alpha+\mathbf{C}\alpha^2+\mathbf{D}\alpha^3,$$

onde

$$\mathbf{A}=3tr(\rho^2 H)-2tr(\rho^3 H)+\mu(tr(\rho)-N_e),$$

$$\mathbf{B}=3tr((\rho d+d\rho)H)-2tr((d\rho^2+\rho d\rho+\rho^2 d)H)+\mu tr(d),$$

$$\mathbf{C}=3tr(d^2 H)-2tr((\rho d^2+d\rho d+d^2\rho)H),$$

$$\mathbf{D}=-2tr(d^3 H).$$

Logo,

$$\Omega'(\rho+\alpha d)=\mathbf{B}+2\mathbf{C}\alpha+3\mathbf{D}\alpha^2.$$

Para achar os pontos críticos vamos usar a estratégia vista na seção 2.2.1 para evitar o cancelamento subtrativo. Então teremos

$$\alpha_1=\frac{-\mathbf{C}-\sqrt{\mathbf{C}^2-3\mathbf{DB}}}{3\mathbf{D}} \quad \text{e} \quad \alpha_2=\frac{\mathbf{B}}{3\mathbf{D}\alpha_1} \quad \text{caso } \mathbf{C}\geq 0$$

ou

$$\alpha_1=\frac{-\mathbf{C}+\sqrt{\mathbf{C}^2-3\mathbf{DB}}}{3\mathbf{D}} \quad \text{e} \quad \alpha_2=\frac{\mathbf{B}}{3\mathbf{D}\alpha_1} \quad \text{caso } \mathbf{C}<0.$$

Um ponto interessante de observar é que uma função polinomial cúbica possui no máximo um ponto de mínimo. Digamos que f possua x_1 e x_2 como pontos mínimos, com $x_1<x_2$ Então, existem intervalos em torno x_1 e x_2 onde a derivada de f antes era negativa e passa a ser positiva.

$$f'(a,x_1)<0 \quad \text{e} \quad f'(x_1,b)>0$$

$$f'(c, x_2) < 0 \quad \text{e} \quad f'(x_2, d) > 0$$

Pelo teorema do valor intermediário, entre b e c obrigatoriamente f' precisa passar por zero novamente, o que é absurdo pois estaríamos encontrando uma terceira raiz para um polinômio de grau 2. Para nós essa informação é bastante útil pois uma vez que concluímos que, por exemplo, x_1 é mínimo, não é mais preciso gastar tempo computacional para testar x_2.

Por fim, para verificar qual dos pontos críticos é ponto mínimo, se existir, é preciso analisar a derivada segunda de Ω que é dada por

$$\Omega''(\rho + \alpha d) = 2\mathbf{C} + 6\mathbf{D}\alpha.$$

O uso da busca linear exata é uma boa estratégia, porém pode ser possível que a função não tenha pontos críticos, ou que esses não sejam mínimos. Nesse caso precisamos continuar pela busca linear inexata.

Algumas considerações devem ser feitas sobre o algorítimo:

1. A matriz teste de densidade ρ_0 deve ser tal que seu traço seja o número correto de elétrons.

2. A purificação de MacWenney é proposta pra que matriz densidade obtida tenha, ou fique próxima de ter, a condição de idempotência. A matriz ρ obtida a cada iteração do método não necessariamente tem essa característica, então a purificamos novamente.

3. A constante Lagrangiana μ é atualizada a cada iteração de modo que o gradiente de Ω tenha traço nulo. Isso garante que em cada atualização de ρ que se preserve o número n de elétrons. Para determinar μ vamos usar a equação 2.20, a linearidade do traço de matrizes e o fato de que o traço de um produto de matrizes não depende da ordem que o produto é realizado.

$$\begin{gathered}
\text{tr}(3\ (\rho H + H\rho)^T - 2(\rho^2 H - \rho H \rho - H\rho^2)^T + \mu I) = \\
3\text{tr}(\ (\rho H + H\rho)^T) - 2tr((\rho^2 H - \rho H \rho - H\rho^2)^T) + \mu tr(I) = \\
3\text{tr}(\rho H + H\rho) - 2tr(\rho^2 H - \rho H \rho - H\rho^2) + \mu tr(I) = \\
3\ \text{tr}(\rho H) + 3tr(H\rho) - 2tr(\rho^2 H) - 2tr(\rho H \rho) - 2tr(H\rho^2) + \mu n = \\
\text{-6 tr}(\rho^2 H) + 6tr(\rho H) + \mu n = 0.
\end{gathered}$$

De onde segue que μ é dado por

$$\mu = \frac{6tr(\rho^2 H) - 6tr(\rho H)}{n}. \tag{2.21}$$

4. A expressão obtida acima para a constante lagrangiana μ é fundamental para ajustar ρ de modo que seu traço seja o número de elétrons. Esse ajuste será feito a cada passo da iteração, sendo assim teremos sempre que

$$\mu(tr(\rho) - N_e) = 0.$$

Logo, em nossos algoritmos por escrever diretamente

$$\Omega(\rho) = 3tr(\rho^2 H) - 2tr(\rho^3 H).$$

Capítulo 3

Implementação Computacional do Método DMTB

Nesse capítulo apresentamos uma implementação computacional para o Método DMTB, exposto no capítulo 1. Antes de falarmos dos algoritmos computacionais faremos uma breve descrição dos motivos que nos levaram a escolher a linguagem C++ para fazer essa implementação.

3.1 Computação Científica em C++

Tradicionalmente o Fortran é a linguagem preferida em Computação Científica. No entanto os modelos matemáticos que tem sido desenvolvidos nas Ciências Aplicadas estão cada vez mais complexos, fazendo com que seja necessário o uso de diversos *packages* para que a simulação computacional seja possível. A complexidade dos modelos matemáticos, o que induz a necessidade do desenvolvimento ser realizado em equipes de pesquisadores, tem motivado programadores a optarem pelo paradigma da orientação a objetos (geralmente usando C++), ao invés da clássica programação estruturada, para produzir tais *packages* ou mesmo para fazer toda a implementação.

Em Física, especificamente em Física da Matéria Condensada, observa-se que a linguagem preferida até hoje é o Fortran, embora o C++, com o paradigma da orientação a objeto, esteja em ascensão. Por exemplo, em University of Maryland existe o curso Physics 165: Introduction to Programming for the Physical Sciences desde 2011 onde é lecionado programação orientada a objeto em C++ (*homepage da disciplina:* https://umdphysics.umd.edu/academics/courses/973-physics-165-introduction-to-programming-for-the-physical-sciences.html). O curso Advanced Computational Physics - Special Topics in Con-

densed Matter Physics em Rutgers University tem o objetivo de ampliar o conhecimento dos alunos em C++ em técnicas e aplicações especificas de Física da Matéria Condensada (homepage da disciplina: http://www.physics.rutgers.edu/ haule/681/). Citamos também o artigo "Object-oriented programming for science" [Chung, 2008] e "Physics0.01 : Object-Oriented Programming for Exact Diagonalization" em [Chung, 2003]. Os livros [Troyer, 2010], [Yang, 2001], [Yevick, 2005] e [Bueno, 2003] também ilustram bem a a ascensão do C++ entre cientistas.

Nosso objeto nessa seção não é argumentar que o Fortran supera o C++ nem vice-versa. O objetivo aqui é apontar pontos positivos e negativos de ambos e esclarecer porquê escolhemos fazer a implementação usando C++ e não o Fortran.

Muitos pesquisadores descartam o uso do C++ por acreditarem que o Fortran ou outras linguagens são mais rápidas sempre. Segundo [Pitt-Francis and Whiteley, 2012], isso nem sempre acontece. A escolha pelo Fortran também costuma acontecer por ela ser uma linguagem mais simples que o C++ e principalmente, por já existir um número enorme de programas inteiros e *packages* desenvolvidos nessa linguagem que possuem alto desempenho. Esse fato desencoraja pesquisadores a usarem novas linguagens de programação.

O C++ possui uma riqueza de bibliotecas bem desenvolvidas e testadas por décadas. Então fazendo uso desses recursos em seus próprios códigos estamos nos apoiando em décadas de experiência e aperfeiçoamento. Além disso, para o C++ existe uma gama enorme de *open-source* e também de ferramentas comerciais, assim é possível reunir recursos para que o C++ fique cada vez mais adequado ao problema que desejamos resolver.

Por último, o que torna o uso do C++ bastante atraente é o paradigma da Orientação a Objeto. Programando sob esse paradigma o programa é extremamente mais fácil de se elaborado, compreendido, estendido e modificado. Para programas curtos, a programação estruturada desempenha um bom papel, porém para programas mais elaborados, com milhares ou milhões de linhas, desenvolvido por diversos pesquisadores a escolha da linguagem C++ Orientada a Objeto é a mais adequada. Como este é exatamente o caso do código desenvolvido nesse trabalho, o C++ foi a nossa escolha.

3.2 Bibliotecas e Classes

3.2.1 Bibliotecas Externas

- Boost

  Devido a grande quantidade de implementações envolvendo Álgebra Linear, optamos por fazer uso da Biblioteca uBLAS da *Boost* EEEE . A *Boost* é uma coleção de bibliotecas de auto nivel que estendem a funcionalidade da linguagem de programação C++. Várias bibliotecas presentes na *Boost* já foram aceitas no TR1*(Tecnical Report 1)* liberados pelo comite ANSI C++. A *Boost* conta com um grande número de desenvolvedores em todo o mundo, sendo que muitos destes fazem parte do comite ANSI C++.

  Dentre as bibliotecas presentes na *Boost* está a uBLAS, uma biblioteca de Álgebra linear com suporte BLAS para matrizes. A uBLAS cobre as operações básicas de álgebra linear com vetores e matrizes, como adição e subtração de vetores e matrizes, produtos de matrizes , produto de matrizes por um escalar, transposição de matrizes, entre outros. Todos esses recursos foram utilizados em nossa implementação.

  Para maiores detalhes sobre as bibliotecas da *Boost* recomendamos seu site oficial, www.boost.org.

3.2.2 Classes Construídas

A classe CCristal.h

A classe CCristal.h reune atributos referentes ao cristal, que são: os vetores primitivos, parâmetro de rede, número de átomos na supercélula e um vetor das coordenadas desses átomos.

Nessa classe, temos um método que faz a leitura externa de estruturas cristalinas chamado **LeSuperCelula()**. Aqui estamos admitindo um padrão para entrada de dados cristalinos, que é a forma:

```
5.43
0.000000 0.500000 0.500000
0.500000 0.000000 0.500000
0.500000 0.500000 0.000000
2
0.000000 0.000000 0.000000
0.500000 0.500000 0.500000
```

Figura 3.1: Formato da entrada de dados cristalinos

Sendo o primeiro valor o parâmetro de rede, as três linhas seguintes representam os vetores primitivos da rede cristalina, na quarta linha temos o número de átomos na supercélula e em seguida, os vetores com as coordenadas desses átomos.

A classe CHamiltoniano.h

Repare que o método DMTB apresentado na seção 1.3 é um método que pode ser usado sempre que se conhece a matriz do hamiltoniano para a estrutura cristalina. Como a proposta dessa implementação é que ela seja um laboratório virtual para o estudo de propriedades eletrônicas e estruturais em cristais, preparamos essa classe de modo que ela possa facilmente ser atualizada com diversas parametrizações do Hamiltoniano. Nesse trabalho, como o foco foi o cálculo da energia mínima no Silício, a classe CHamiltoniano possui como atributos os parâmetros de Kwon, apresentados nas tabelas 1.1 e 1.2 e um elemento da classe CCristal, sobre o qual iremos calcular a matriz do hamiltoniano.

O principal método da classe se chama **Kwon()** e é o método que, de fato, constrói a matriz do hamiltoniano do Kwon, apresentado na seção 1.2.1. A matriz hamiltoniano de Kwon é uma matriz $4N \times 4N$ onde N é número de átomos de silício que tem o sistema. Essa matriz pode ser pensada como uma matriz de $N \times N$, com cada entrada (i, j) sendo uma matriz 4×4, que representa a interação entre os orbitais (Veja 1.2.1). Os blocos 4×4 são aqueles apresentados em 1.3 e 1.5.

Outro fator importante que entra nesse trecho é o raio de corte para o hamiltoniano. Essa estratégia é um recurso para se tentar diminuir o custo computacional do programa. Nesse trabalho estamos usando um raio de corte fixo: 0.78*(parâmetro de rede). O algoritmo 8 contém o trecho do código referente ao raio de corte.

A classe CDmtb.h

A classe CDmtb é a principal classe do nosso programa. Dentre seus atributos, temos a matriz densidade eletrônica ρ, o raio de corte para a matriz densidade e um objeto da classe CHamiltoniano que será responsável pela passagem da matriz do hamiltoniano cristalino. A classe CDmtb também possui um atributo matricial, para receber o hamiltoniano caso este venha diretamente de uma arquivo de entrada. É nesse método que implementamos o que foi exposto no seção 2.3.

Devemos lembrar que a estratégia que usamos para construir a matriz do hamiltoniano a torna uma

Algoritmo 7: Estratégia para Implementação de Kwon();

Entrada: número de átomos do sistema, parâmetros de Kwon
Saída: matriz $hkwon$ do hamiltoniano

início
  para cada $i < n$ **faça**
    para cada $j < n$ **faça**
      se *i=j* **então**
        $hkwon \leftarrow$ matriz nula $n \times n$;
        $hkwon(i,j)(0,0) \leftarrow E_s$;
        $hkwon(i,j)(1,1) \leftarrow E_p$;
        $hkwon(i,j)(2,2) \leftarrow E_p$;
        $hkwon(i,j)(3,3) \leftarrow E_s$;
      senão
        Inserir a matriz 4×4 definida pela fórmula 1.5
      fim
    fim
  fim
fim
retorna $hkwon$

Algoritmo 8: Raio de Corte para o Hamiltoniano

Entrada: D, A {a distância entre o i-ésimo e j-ésimo átomo, parâmetro de rede}
Saída: matriz $hkwon$ do hamiltoniano já ajustada com o raio de corte

início
  Seja D
  Seja A o parâmetro de rede;
  para cada $i < n$ **faça**
    para cada $j < n$ **faça**
      se $D < 0,78 * A$ **então**
        Inserir a matriz 4×4 definida pela fórmula 1.5
      senão
        $hkwon(i,j) \leftarrow$ matriz nula 4×4;
      fim
    fim
  fim
fim
retorna $hkwon$

matriz com entradas matriciais e isso impossibilita a utilização de operações matriciais comuns entre a matriz do hamiltoniano e a matriz densidade, que é uma matriz com entradas numéricas. Criamos então um método chamado **SetH()** que converte a matriz do hamiltoniano, antes $n \times n$ cuja as entradas eram matrizes 4×4, em uma matriz $4n \times 4n$ com entradas numéricas. Em termos computacionais, estamos com a matriz do hamiltoniano indexada por 4 índices, e precisamos convertê-la pra uma matriz usual, indexada por apenas 2 índices. Com isto, essa última terá sua ordem quadruplicada. Vejamos um exemplo numérico:

$$\left[\begin{array}{cccc|cccc} 1 & -1 & 0 & 1 & 2 & 3 & 1 & -2 \\ -1 & 5 & 1 & 0 & 0 & -3 & -1 & 3 \\ 5 & 0 & 0 & -1 & -2 & 5 & 4 & -6 \\ -2 & 3 & 2 & 4 & 4 & -1 & 0 & 0 \\ \hline 0 & -3 & 4 & -1 & 5 & 5 & 6 & -1 \\ 5 & -3 & 0 & 9 & 5 & 2 & -3 & 4 \\ 4 & 0 & 4 & 0 & 0 & -3 & 7 & 0 \\ 0 & 0 & -2 & -1 & 7 & 4 & 3 & -1 \end{array}\right]$$

Figura 3.2: Matriz 2×2 com entradas da forma de matrizes 4×4 que pode ser vista como uma matriz de ordem 8

Para ilustrar a estratégia utilizada, observe essa matriz genérica 2×2, onde cada entrada é uma matriz 4×4. Para cada i, j o bloco é indicado por A^{ij}, com $i, j = 0, 1$. Uma vez estando no bloco A^{ij} é preciso de mais dois índices para indicar a posição dos elementos em cada bloco, podemos escrever então $A^{ij}(k, l)$ com $i, j = 0, 1$ e com $k, l = 0, 1, 2, 3$.

Desejamos transforma-lá em uma matriz clássica $A = (a_{mn})$, com $m, n = 0, 1, \cdot, 7$.

$$\begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & a(0,3) & a(0,4) & a(0,5) & a(0,6) & a(0,7) \\ a(1,0) & a(1,1) & a(1,2) & a(1,3) & a(1,4) & a(1,5) & a(1,6) & a(1,7) \\ a(2,0) & a(2,1) & a(2,2) & a(2,3) & a(2,4) & a(2,5) & a(2,6) & a(2,7) \\ a(3,0) & a(3,1) & a(3,2) & a(3,3) & a(3,4) & a(3,5) & a(3,6) & a(3,7) \\ a(4,0) & a(4,1) & a(4,2) & a(4,3) & a(4,4) & a(4,5) & a(4,6) & a(4,7) \\ a(5,0) & a(5,1) & a(5,2) & a(5,3) & a(5,4) & a(5,5) & a(5,6) & a(5,7) \\ a(6,0) & a(6,1) & a(6,2) & a(6,3) & a(6,4) & a(6,5) & a(6,6) & a(6,7) \\ a(7,0) & a(7,1) & a(7,2) & a(7,3) & a(7,4) & a(7,5) & a(7,6) & a(7,7) \end{bmatrix}.$$

Agora vamos relacionar os índices, para isso precisamos analisar como delimitar os blocos nessa última

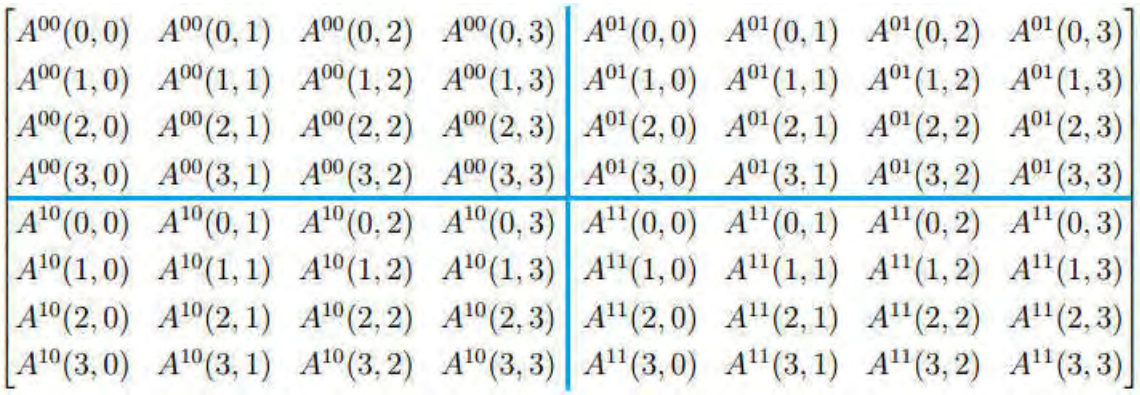

Figura 3.3: Matriz 2×2 com entradas da forma de matrizes 4×4 escrita de forma genérica, ilustrando a necessidade de 4 índices.

matriz. Repare que quando estamos com $0 \leq m \leq 3$ e $0 \leq n \leq 3$, construímos o bloco A^{00}. Quando estamos com $0 \leq m \leq 3$ e $4 \leq n \leq 7$, construímos o bloco A^{01} e assim sucessivamente a cada variação de 4 números nos índices m e n. Daí podemos retirar a seguinte relação com os índices:

$$m = 4i + k$$

$$n = 4j + l.$$

Assim fazendo a divisão inteira de m e n por 4 vamos obter respectivamente i e j. O par com o resultado da divisão será correspondente ao bloco que o elemento pertence. E por fim, encontrando o resto da divisão inteira de m e n por 4 obtemos k e l. O par com os restos da divisão indica a posição do elemento dentro do bloco. Por exemplo, para saber qual elemento colocar na posição $(6, 5)$, iremos procurar no bloco $(1, 1)$ e posição $(2, 1)$. Assim, $a(6, 5) = A^{11}(2, 1)$. No algoritmo 8 temos essa implementação. Nesse algoritmo *div* representa a divisão inteira e *mod* o resto da divisão inteira.

Feita essa transformação na matriz do hamiltoniano, o próximo passo é aplicar o método gradiente conjugado não linear apresentado no capítulo 2. Nesse simulador implementamos a modificação de Fletcher-Reeves (algoritmo 5) e também a modificação de Polak-Ribiere (algoritmo 6). É importante lembrar que para isso foi preciso fazer uma vetorização da função $\Omega(\rho)$. No capítulo 2, seção 2.3 explicamos como foi escolhida a vetorização e também a estratégia usada para se ganhar tempo computacional nos cálculo do NLGC. Para o cálculo dos produtos escalares que aparecem na determinação de b_k para a atualização da direção de descida d_k, vamos usar a proposição 4.1 de [Filho, 2017]. A proposição diz que se v_A e v_B forem

Algoritmo 9: Matriz de blocos $n \times n$ em Matriz $4n \times 4n$

Entrada: $hkwon$ {matriz do hamiltoniano}
Saída: matriz h do hamiltoniano já como matriz $4n \times 4n$

```
início
    para cada i < 4n faça
        para cada j < 4n faça
            h(i, j) ← hkown(i div 4, j div 4)(i mod 4, j mod 4);
        fim
    fim
fim
retorna h
```

vetorizações respectivamente das matrizes A e B, então teremos que

$$v_A \cdot v_B = tr(A^T B).$$

Esta proposição é bastante útil pois reduz o custo computacional ao já fazer os cálculos com as matrizes, sem ter que vetoriza-las primeiro.

Feitas essas considerações, o esquema de aplicação do método dos gradientes conjugados, já incluindo as particularidades citadas nas seção 2.3, fica conforme o algoritmo 10.

O método responsável por esse processo em nosso simulador é **DMTB()**.

Para finalizar, temos os algoritmos 12 e 13 que calculam respetivamente a energia eletrônica mínima e a energia mínima total.

A relação entre as classes

A figura 3.4 mostra a relação entre as classes construídas. A classe CDMTB tem como atributo um objeto da classe CHamiltoniano, e esse por sua vez, possui como atributo um objeto da classe CCristal.

Embora nesse trabalho tenhamos implementado apenas o cálculo da energia mínima para o silício, o programa foi construído de forma a facilmente poder ser estendido para outros elementos (basta acrescentar na classe CCristal) e para outras parametrizações de hamiltoniano (basta incorpora-las na classe CHamiltoniano). A classe DMTB é geral e não específica para o silício com o hamiltoniano de Kwon. As informações sobre o elemento e sobre o hamiltoniano são advindas do atributo que a classe CHamiltoniano possui.

Algoritmo 10: Método DMTB com Gradiente Conjugado Não-Linear

Entrada: $\Omega, \nabla\Omega, \rho_0, tol, it_{max}$ { campo escalar, gradiente do campo, estimativa inicial, tolerância para o critério de parada e número máximo de iterações}

Saída: vetor x que minimiza o campo escalar

início

  $it = 0$

  $\rho := \rho_0$;

  $d_0 := -\nabla\Omega(\rho_0)$; /* Os gradientes são calculados pela equação 2.20 */

  $d := d_0$;

  $G := d_0$;

  enquanto $||G|| > tol$ *ou* $it < it_{max}$ **faça**

    $G_0 := \nabla\Omega(\rho_0)$;

    Encontre a ; /* Com busca unidirecional exata ou inexata. Veja algoritmo 11 */

    $\rho = \rho + ad$;

    Purifique ρ

    $\rho = 3\rho^2 - 2\rho^3$; /* Purificação de McWenney */

    $G = \nabla\Omega(\rho)$;

    Calcule b

    $b^{FR} = \dfrac{tr(G^T G)}{tr({G_0}^T G_0)}$; /* Para o Método de Fletcher Reevers */

    $b^{PR} = \dfrac{tr(G^T (G - G_0))}{tr({G_0}^T G_0)}$; /* Para o Método de Polak-Ribiere */

    $d := -G + bd$;

    Atualize μ

    $\mu = \dfrac{6tr(\rho^2 H) - 6tr(\rho H)}{n}$; /* Seguindo a equação 2.21 */

    $\rho_0 = \rho$;

    it=it+1;

  fim

fim

retorna ρ

Algoritmo 11: Algoritmo para determinar o passo a

Entrada: $\Omega, \nabla\Omega$, **A, B, C, D**, c_1, c_2, c { campo escalar, gradiente do campo, **A, B, C, D** são coeficientes de Ω quando vista como funcão polinomial de grau 3 em a, parâmetros das condições fortes de Wolfe}

Saída: passo a

início
  wolfe:=0;
  se $\boldsymbol{C} \geq 0$ **então**
    $a_1 = \frac{-C-\sqrt{C^2-3DB}}{3D}$;
    $a_2 = \frac{B}{3Da_1}$;
  senão
    $a_1 = \frac{-C+\sqrt{C^2-3DB}}{3D}$;
    $a_2 = \frac{B}{3Da_1}$;
  fim
fim
retorna ρ

Algoritmo 12: Calculo da Energia Eletrônica Mínima

Entrada: H, ρ { hamiltoniano e matriz densidade que realiza o mínimo}

Saída: E -Energia mínima do sistema

início
  $E := tr(\rho H)$;
fim
retorna E;

Algoritmo 13: Cálculo da Energia Total Mínima

Entrada: E, E_{rep}, N, E_0 { Energia eletrônica mínima, Energia repulsiva, número de átomos e constante de energia}

Saída: E_{total} -Energia total do sistema

início
  $E_{total} := E + E_{rep} + NE_0$;
fim
retorna E_{total};

Figura 3.4: Diagrama de Classes do Simulador

O simulador

No simulador o usuário entra com o arquivo que possui a estrutura cristalina, os cálculos são realizados internamente, e por fim o usuário informa o nome que deseja salvar o arquivo contendo o hamiltoniano cristalino, a matriz ρ que minimiza a energia e o valor da energia.

Considerações Finais

Trabalhos de implementação como este, costumam ser tarefas árduas, pois em um primeiro momento é necessário desvendar toda a teoria que cerca o problema e depois criar estratégias para um bom funcionamento do algoritmo. Além disso, quando o volume de dados é enorme,como nos casos que aparecem na Física da Matéria Condensada, é fundamental criar algoritmos com complexidade mais baixa possível. É importante observar que os mesmos autores de [Li et al., 1993] desenvolveram um simulador,bastante eficiente e que vem sendo utilizado e aperfeiçoado por anos. Em nenhum momento nossa intensão foi construir nesse curto período de tempo um simulador do mesmo nível que este, mas sim criar nosso pró-

Algoritmo 14: Programa Principal

```
1 CCristal diamante ;          /* Cria um objeto da classe CCristal chamado
  diamante */
2 diamante.LeSuperCelular ; /* Faz leitura do arquivo externo com os dados
  do cristal */
3 CHamiltonian kwon(diamante) ;  /* Cria um objeto da classe CHamiltoniano
  chamado kwon tendo diamante da CCristal como atributo */
4 kwon.Kwon() ;        /* Aplica ao objeto kwon o método que constroi o
  hamiltoniano */
5 CDmtb dmtb(kwon) ;     /* Cria um objeto da classe CDmtb chamado dmtb
  tendo kwon da classe CHamiltoniano como atributo */
6 dmtb.SetH() ;    /* Converte a matriz do hamiltoniano para uma matriz
  4n × 4n */
7 dmtb.DMTB(0.0001,128) ;        /* Executa o método DMTB com tolerância de
  0,0001 e 128 iterações máximas. */
8 Salva as informações obtidas em um arquivo externo;
```

prio simulador para que tenhamos expertise e no futuro possamos atingir nosso projeto final que é criar um laboratório virtual para o estudo de propriedades eletrônicas e estruturais em cristais.

Para validar o simulador que desenvolvemos nesse trabalho, decidimos verificar o comportamento da energia obtida quando variamos o parâmetro de rede do Silício. O parâmetro de rede do Silício, medido de forma experimental na temperatura ambiente de 300k,é de 5.43 A. Espera-se então, que nosso simulador possa encontrar que o menor valor para a energia no estado fundamental ocorra neste valor para o parâmetro de rede.

Portanto, de início foi necessário executar nosso simulador supondo o Silício com diversos parâmetros de rede. Isso não foi um problema, pois estamos usando a parametrização de Kwon para o Hamiltoniano e esta parametrização é transferível,ou seja, independe da estrutura cristalina que estamos usando. Os parâmetros de Kwon já incorporam as informações da espécie química e as variações que ocorrem quando variamos o parâmetro de rede aparecem explicitamente,visto que entradas das matriz dependem diretamente da distância entre os átomos.

Para fazer esse teste usamos 10 valores para a energia obtidos pelo DMTB, com parâmetro de rede variando de $5.2A$ até $5.65A$ com variação de $0.05A$. Os valores obtidos estão tabelados em 3.1 e 3.2.

Em seguida, fizemos um ajuste polinomial para esses pontos. Usamos mínimos quadrados e ajustamos

Parâmetro de Rede (A)	5.2	5.25	5.3	5.35	5.4
Energia (eV)	-4.462575	-4.499944	-4.528473	-4.548335	-4.559587

Tabela 3.1: Parâmetro de Rede x Energia

Parâmetro de Rede (A)	5.45	5.5	5.55	5.6	5.65
Energia (eV)	-4.562314	-4.556653	-4.542799	-4.521004	-4.491575

Tabela 3.2: Parâmetro de Rede x Energia

os pontos através da parábola de equação

$$y = 1.6773x^2 - 18.2607x + 45.1371.$$

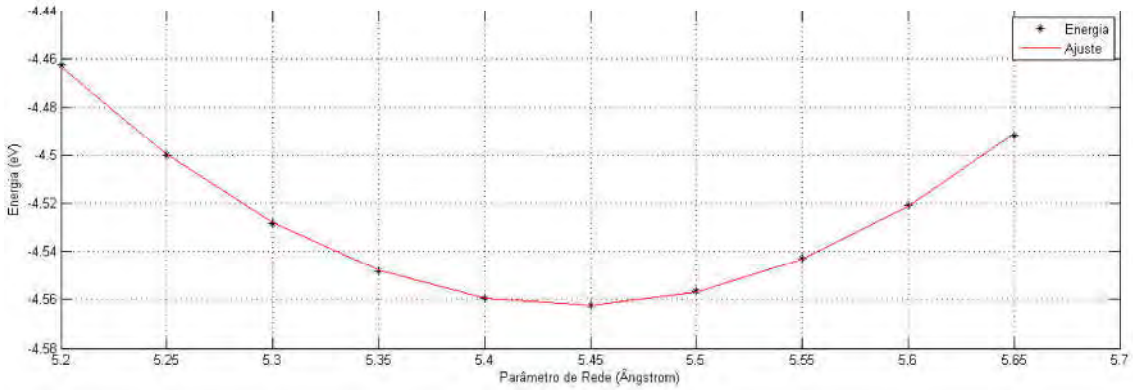

Figura 3.5: Parâmetro de Rede x Energia

O mínimo desta parábola ocorre em $x = 5.4433$ Angstrons. Quando comparamos com o parâmetro de rede experimental, $5.43A$, obtemos um erro relativo de apenas 0,2%, mostrando que nossos cálculos estão coerentes com os presentes na literatura.

A base para esse trabalho foi o artigo [Li et al., 1993]. Nele, a proposta era que a complexidade computacional do método fosse de ordem N, onde N é o número de átomos do sistema. Na verdade, esse método possui um grande potencial para ser implementado dessa forma, porém ainda não aperfeiçoamos nosso simulador para este objetivo.

Umas das chaves para reduzir a complexidade do algoritmo é o raio de corte da matriz densidade. Vamos admitir então que se tenha uma implementação para o NLCG com complexidade N. Levando em consideração o processo de vetorização que estamos usando, a versão adaptada para o DMTB fica com complexidade N^2.

No entanto, sabe-se que

$$\lim_{R_{ij}\to\infty} \rho_{ij} = 0,$$

onde R_{ij} é a distância entre os orbitais i e j.

Diante deste fato, o grupo de Vanderbilt propôs a introdução de um raio de corte $R_c > 0$ de modo que se $R_{ij} > R_c$ então $\rho_{ij} = 0$. Com isso, nosso processo de vetorização queantes produzia uma vetor com M^2N^2 posições produz um vetor com NLM^2 coordenadas, onde M é o número de orbitais por átomo e L é o números de átomos presente numa esfera de raio R_c centrada em qualquer átomo. Tendo este vetor como entrada, o algoritmo NLCG faria uso de NLM^2 passos, apresentando, portanto, complexidade $O(N)$.

Referências Bibliográficas

[Aguiar Junior, 2006] Aguiar Junior, F. F. (2006). Trabalho de curso: Software visual das estruturas cristalinas dos sólidos em três dimensões. Master's thesis, UCB.

[Al-Baali, 1985] Al-Baali, M. (1985). Descent property and global convergence of the fletcher—reeves method with inexact line search. *IMA Journal of Numerical Analysis*, 5(1):121–124.

[Al-Baali and Fletcher, 1986] Al-Baali, M. and Fletcher, R. (1986). An efficient line search for nonlinear least squares. *Journal of Optimization Theory and Applications*, 48(3):359–377.

[Amaral et al., 2011] Amaral, B., Baraviera, T., and Cunha, M. (2011). *Mecânica Quântica para Matemáticos em Formação.* 28º Colóquio Brasileiro de Matemática. IMPA.

[Araújo, 2006] Araújo, M. M. (2006). *Teses de Doutorado: Estudos Teóricos Sobre Discordâncias Cristalinas em Silício.* PhD thesis, UFMG.

[Ashcroft and Mermin, 1976a] Ashcroft, N. and Mermin, N. (1976a). *Solid state physics.* Science: Physics. Saunders College.

[Ashcroft and Mermin, 1976b] Ashcroft, N. and Mermin, N. (1976b). *Solid state physics.* Science: Physics. Saunders College.

[Borges, 1985] Borges, P. R. T. (1985). Dissertação de mestrado: Aspéctos sobre a aplicação de métodos de gradientes conjugados em otimização irrestrita diferenciável. Master's thesis, COPPE/UFRJ.

[Bowler, 1997] Bowler, D. P. (1997). *PhD Thesis: A theoretical investigation of gas source growth of the Si(001) surface.* PhD thesis, Wolfson College.

[Bueno, 2003] Bueno, A. D. (2003). *Programação Orientada a Objeto com C++.* NOVATEC.

[Chung, 2003] Chung, M. (2003). Physics0.01: Object-oriented programming for exact diagonalization. *eprint arXiv:cond-mat/030253*.

[Chung, 2008] Chung, M. (2008). Science code .net: Object-oriented programming for science. *Science of Computer Programming*, 71(3):242 – 247.

[Ciências, 1999] Ciências, S. (1999). Tabela periódica v2.5.

[Daw, 1993] Daw, M. S. (1993). Model for energetics of solids based on the density matrix. *Phys. Rev. B*, 47:10895–10898.

[Dorsett et al., 2000] Dorsett, H., White, A., Science, D., (Australia), T. O., Aeronautical, and Division, M. R. L. A. W. S. (2000). *Overview of Molecular Modelling and Ab Initio Molecular Orbital Methods Suitable for Use with Energetic Materials*. General document. DSTO Aeronautical and Maritime Research Laboratory.

[Filho, 2017] Filho, J. M. B. (2017). Dissertação de mestrado: Desenvolvimento de uma nova proposta matemÁtica e computacional para a metodologia da matriz densidade tight-binding ordem-n. Master's thesis, UFRRJ.

[Fletcher and Reeves, 1964] Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154.

[Frauenheim et al., 2000] Frauenheim, T., Seifert, G., Elsterner, M., Hajnal, Z., Jungnickel, G., Porezag, D., Suhai, S., and Scholz, R. (2000). A self-consistent charge density-functional based tight-binding method for predictive materials simulations in physics, chemistry and biology. *physica status solidi (b)*, 217(1):41–62.

[Freed, 1995] Freed, K. F. (1995). *Structure and Dynamics of Atoms and Molecules: Conceptual Trends*, chapter Building a Bridge between Ab Initio and Semiempirical Theories of Molecular Electronic Structure. Springer Netherlands.

[Gilbert and Nocedal, 1992] Gilbert, J. C. and Nocedal, J. (1992). Global convergence properties of conjugate gradient methods for optimization.

[Hammes, 2011] Hammes, I. (2011). Dissertação de mestrado: Bundles de nanotubos de carbono:teoria e experimento. Master's thesis, UFF.

[Hoffman and Kunze, 1971] Hoffman, K. and Kunze, R. (1971). *Linear algebra.* Prentice-Hall mathematics series. Prentice-Hall.

[I. Morrison and Payne, 1997] I. Morrison, J.-C. Li, S. S. X. and Payne, M. C. (1997). Ab-initio total energy studies of the static and dynamical properties of ice ih. *The Journal of Physical Chemistry B*, 101(32):6146–6150.

[Jiang and Yang, 2004] Jiang, H. and Yang, W. (2004). Conjugate-gradient optimization method for orbital-free density functional calculations. *The Journal of Chemical Physics*, 121(5):2030–2036.

[Kittel, 1971] Kittel, C. (1971). *Introduction to solid state physics.* Wiley.

[Kwon et al., 1994] Kwon, I., Biswas, R., Wang, C. Z., Ho, K. M., and Soukoulis, C. M. (1994). Transferable tight-binding models for silicon. *Phys. Rev. B*, 49:7242–7250.

[Lang, 1966] Lang, S. (1966). *Linear Algebra.* Addison-Wesley Publishing Company.

[Li et al., 1993] Li, X.-P., Nunes, R. W., and Vanderbilt, D. (1993). Density-matrix electronic-structure method with linear system-size scaling. *Phys. Rev. B*, 47:10891–10894.

[Lima, 2008] Lima, E. L. L. (2008). *Curso de análise Volume II - Décima Edição.* IMPA.

[McWeeny, 1960] McWeeny, R. (1960). Some recent advances in density matrix theory. *Rev. Mod. Phys.*, 32:335–369.

[Millam and Scuseria, 1997] Millam, J. M. and Scuseria, G. E. (1997). Liner scaling density matrix. *J. Chem. Phys.*, 106:5569–5577.

[Moré and Thuente, 1994] Moré, J. J. and Thuente, D. J. (1994). Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3):286–307.

[Nakano, 2015] Nakano, A. (2015). Notas de aula: Tight-binding model of electronic structures. http://cacs.usc.edu/education/phys516/04TB.pdf. Acessado em 20-03-2015.

[Nero, 1999] Nero, J. D. (1999). *Tese de Doutorado: Aplicação de Métodos de Estrutura Eletrônica em Polímeros Visando o Desenvolvimento de Novos Materiais.* PhD thesis, UNICAMP.

[Nightingale et al., 1993] Nightingale, M. P., Viswanath, V. S., and Müller, G. (1993). Computation of dominant eigenvalues and eigenvectors: A comparative study of algorithms. *Phys. Rev. B*, 48:7696–7699.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical optimization.* Springer Series in Operations Research and Financial Engineering. Springer, Berlin. NEOS guide http://www-fp.mcs.anl.gov/otc/Guide/.

[Oliveira, 2014] Oliveira, A. C. A. (2014). Dissertação de mestrado: Modelagem computacional da interação entre discordâncias parciais a 90 graus e a superfície (111) do silício. Master's thesis, UFRRJ.

[Ordejón, 1998] Ordejón, P. (1998). Order-n tight-binding methods for electronic-structure and molecular dynamics. *Computational Materials Science*, 12(3):157 – 191.

[Ordejón et al., 1996] Ordejón, P., Artacho, E., and Soler, J. M. (1996). Self-consistent order-n density-functional calculations for very large systems. *Phys. Rev. B*, 53:R10441–R10444.

[Ordejón et al., 1995] Ordejón, P., Drabold, D. A., Martin, R. M., and Itoh, S. (1995). Linear scaling method for phonon calculations from electronic structure. *Phys. Rev. Lett.*, 75:1324–1327.

[Padilha, 1997] Padilha, A. F. (1997). *Materiais de engenharia: microestrutura, propriedades.* São Paulo: Hemus.

[Paxton and Sutton, 1989] Paxton, A. and Sutton, A. (1989). A tight-binding study of grain boundaries in silicon. *Acta Metallurgica*, 37(7):1693 – 1715.

[Pfrommer et al., 1999] Pfrommer, B. G., Demmel, J., and Simon, H. (1999). Unconstrained energy functionals for electronic structure calculations. *Journal of Computational Physics*, 150(1):287 – 298.

[Pitt-Francis and Whiteley, 2012] Pitt-Francis, J. and Whiteley, J. (2012). *Guide to Scientific Computing in C++.* Undergraduate Topics in Computer Science. Springer London.

[Polak and Ribiere, 1969] Polak, E. and Ribiere, G. (1969). Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 3(R1):35–43.

[Powell, 1984] Powell, M. J. D. (1984). Nonconvex minimization calculations and the conjugate gradient method. In *Lecture Notes in Mathematics 1066*, pages 122–141. Springer–Verlag, Berlin.

[Powell, 1986] Powell, M. J. D. (1986). Convergence properties of algorithms for nonlinear optimization. *SIAM Rev.*, 28(4):487–500.

[Ribeiro and Karas, 2011] Ribeiro, A. A. and Karas, E. W. (2011). Um curso de otimização. Curitiba.

[Slater and Koster, 1954] Slater, J. C. and Koster, G. F. (1954). Simplified lcao method for the periodic potential problem. *Phys. Rev. B*, 94:1498–1524.

[Spivak, 1965] Spivak, M. (1965). *Calculus on Manifolds.* Perseus Books Publishing L.L.C.

[Troyer, 2010] Troyer, M. (2010). Computational quantum physics. ETH Zurich, SS.

[VandeVondele et al., 2012] VandeVondele, J., Borstnik, U., and Hutter, J. (2012). Linear scaling self-consistent field calculations with millions of atoms in the condensed phase. *Journal of Chemical Theory and Computation*, 8(10):3565–3573.

[Yang, 2001] Yang, D. (2001). *C++ and Object-Oriented Numeric Computing for Scientists and Engineers.* Springer.

[Yevick, 2005] Yevick, D. (2005). *A First Course in Computational Physics and Object-Oriented Programming with C++ Hardback with CD-ROM.* Cambridge University Press.

[Zhang et al., 2012] Zhang, Y., Zheng, H., and Zhang, C. (2012). Global convergence of a modified prp conjugate gradient method. *Procedia Engineering*, 31:986 – 995.

[Zoutendijk, 1970] Zoutendijk, G. (1970). Nonlinear programming, computational methods. *Integer and nonlinear programming*, 143(1):37–86.