

IFT - UNESP
INSTITUTO DE FÍSICA TEÓRICA

TESE DE MESTRADO

IFT-D.004/24

Transporte em Nanodispositivos na Presença de Interação Elétron-Elétron

Washington Francisco dos Santos Junior

Orientador

Dr. Alexandre Reily Rocha

São Paulo - SP
17 de abril de 2024

S237	Santos Junior, Washington Francisco dos Transporte em nanodispositivos na presença de interação elétron-elétron / Washington Francisco dos Santos Junior. – São Paulo, 2024 240 p: il. color. Dissertação (mestrado) – Universidade Estadual Paulista (Unesp), Instituto de Física Teórica (IFT), São Paulo Orientador: Alexandre Reily Rocha 1. Matéria condensada. 2. Física. 3. Nanoestruturas. I. Título
------	--

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Física Teórica (IFT), São Paulo. Dados fornecidos pelo autor(a).

TRANSPORTE EM NANODISPOSITIVOS NA PRESENÇA DE INTERAÇÃO ELÉTRON-ELÉTRON

Dissertação de Mestrado apresentada ao Instituto de Física Teórica do Câmpus de São Paulo, da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como parte dos requisitos para obtenção do título Mestre em Física, Especialidade Física Teórica.

Comissão Examinadora:

Prof. Dr. ALEXANDRE REILY ROCHA (Orientador)
Instituto de Física Teórica/UNESP

Prof. Dr. ANDERSON LUIZ DA ROCHA E BARBOSA
Universidade Federal Rural de Pernambuco/UFPRE

Prof. Dr. JOSÉ EDUARDO PADILHA E SOUSA
Universidade Federal do Paraná/UFPR

Conceito: Aprovado

São Paulo, 17 de abril de 2024.

*Dedico esta dissertação à
Minha mãe, Edneide Vasconcelos,
Minha avó, Adeilda Vasconcelos
e minhas sobrinhas, Maria e Layla.
Sem vocês, nada disso seria possível.*

Agradecimentos

Chegar até aqui não foi fácil, e foi maravilhoso encontrar pessoas ao longo do caminho que sempre estiveram dispostas a ajudar e se fazer presentes.

Agradeço a *Abbá*, Deus meu, por me conceder uma mente sempre aberta ao desconhecido e com sede de enxergar além do que é encoberto.

Agradeço à minha mãe e à minha avó pelo apoio e suporte quase incondicionais para perseguir os meus sonhos. Sem elas, eu jamais estaria aqui.

Agradeço a todos os meus amigos, em particular ao Eudes, um amigo que a vida e a academia me trouxeram e que sempre esteve ao meu lado, desempenhando um papel de extrema importância na minha carreira. Ao Pablito, Vivian e Renan, que sempre me incentivaram a me tornar uma pessoa melhor e que tentaram ao máximo me instruir com a sua experiência para que eu aprendesse com ela. Ao Rodrigo, João e Isa, pelas conversas e cafés pelas ladeiras; foi um grande prazer compartilhar esses momentos com vocês. Aos meus amigos não tão típicos, que o IFT me permitiu conhecer. Ao Jorge e à Ju, pelos passeios incríveis pelas ruas de São Paulo. A Luanna e Raquel, por todas as conversas, foi um prazer estar com vocês. Ao Ricardo e à Luisa, que sempre me incentivaram e estiveram perto em todos os momentos, mesmo distantes, uma das felicidades da minha vida é ter conhecido vocês. A Keline, Antonio, Fanie e Juninho, por estarem presente para escutar minhas reclamações.

E eu jamais poderia esquecer do meu orientador, Alexandre Rocha, por aceitar me orientar e estar comigo do começo ao fim do meu mestrado, por todo o seu suporte e compreensão nos meus dias ruins, pelos conselhos e longos debates que enriqueceram profundamente minha jornada, e pelas conversas que se estenderam além da vida profissional. Foi um imenso prazer compartilhar esses momentos.

Ao Ailton, meu orientador na graduação, e ao Pedro Hugo. Pessoas que me incentivaram a entrar e seguir na carreira acadêmica, mostrando-me o que é ser um físico e a beleza que existe por trás das equações.

A todos os meus amigos de Pernambuco e de Igarassu, é sempre um prazer estar com vocês, e espero estar junto pelas ruas da nossa terra em breve.

Agradeço às agências de fomento que deram suporte à produção da minha dissertação. Ao CNPq, por conceder a bolsa de mestrado. À Capes, IFT e Fapesp, pelo suporte às viagens e equipamentos. Ao INCT - Nanocarbono e INCT - Materials Informatics, pelo suporte. Aos Parques computacionais, Cenapad-SP e Grid-Unesp, sem os quais não seria possível desenvolver a parte computacional da minha dissertação.

*"Aquele loco que não pode errar
Aquele que você odeia amar nesse instante
Pele parda e ouço funk
E de onde vem os diamante? Da lama
Valeu mãe, negro drama"*

Racionais MC's,

Resumo

Estruturas à base de carbono, como os nanotubos, e mais recentemente o grafeno, têm atraído grande atenção da comunidade científica tanto do ponto de vista da ciência básica como de possíveis aplicações em fotônica e eletrônica. No caso particular do grafeno, uma estrutura bidimensional de carbono organizada em uma rede hexagonal, temos um material extremamente resistente com propriedades de transporte eletrônico e térmico praticamente incomparáveis. Para resolver o problema da ausência de um gap, ou para obter fios condutores, nanofitas de grafeno podem ser obtidas confinando o grafeno em uma dimensão. Essas estruturas unidimensionais retêm muitas das propriedades do grafeno em um fio em nanoescala. Além disso, nanofitas podem ser conectadas a outros materiais (também à base de carbono) para criar novos dispositivos. Este é o caso dos pontos quânticos abertos que podem funcionar como transistores de elétron único. A presença de defeitos nas nanofitas de grafeno desempenha um papel importante nas propriedades de transporte e, dependendo da concentração, pode levar à localização de Anderson. A presença de vacâncias nas nanofitas de grafeno pode gerar tanto efeitos de desordem quanto o surgimento de estados localizados.

Normalmente, ao estudar modelos desordenados, a interação elétron-elétron é negligenciada. Aqui, buscamos entender como a interação elétron-elétron altera as propriedades eletrônicas e de transporte de dispositivos quânticos, em particular pontos quânticos baseados em grafeno e nanofitas de grafeno desordenadas. Para tanto, realizamos cálculos utilizando o método da função de Green de fora de equilíbrio, descrevendo os estados de interação por meio de um modelo de Anderson para impurezas magnéticas juntamente com a chamada aproximação de não cruzamento (NCA) e uma generalização para a fórmula de Meir-Wingreen no regime de acoplamento não proporcional. Assim, por meio da combinação desses métodos, conseguimos obter fenômenos de transporte para sistemas interagentes. Observamos o surgimento de uma transmissão e corrente incoerentes devido ao processo de quebra de fase do transporte quântico coerente via a presença de interação elétron-elétron em ambos os sistemas. Além de uma descrição mais precisa, descrevemos fenômenos antes não descritos a partir de uma perspectiva puramente microscópica.

Palavras Chaves: Interação elétron-elétron; Nanodispositivos; Transporte Incoerente; Grafeno; Transporte Eletrônico.

Áreas do conhecimento: Física; Matéria Condensada; Transporte Eletrônico.

Abstract

Carbon-based structures, such as nanotubes, and more recently graphene, have attracted great attention from the scientific community both from a basic science perspective and for potential applications in photonics and electronics. In the particular case of graphene, a two-dimensional carbon structure organized in a hexagonal lattice, we have an extremely resilient material with electron and thermal transport properties that are practically unparalleled. To address the issue of the absence of a bandgap, or to obtain conductive wires, graphene nanoribbons can be obtained by confining graphene in one dimension. These one-dimensional structures retain many of the properties of graphene in a nano-scale wire. Furthermore, nanoribbons can be connected to other materials (also carbon-based) to create new devices. This is the case with open quantum dots that can function as single-electron transistors. The presence of defects in graphene nanoribbons plays an important role in the transport properties and, depending on the concentration, can lead to Anderson localization. The presence of vacancies in graphene nanoribbons can generate both disorder effects and the emergence of localized states.

Typically, when studying disordered models, electron-electron interaction is neglected. Here, we seek to understand how electron-electron interaction alters the electronic and transport properties of quantum devices, particularly graphene-based quantum dots and disordered graphene nanoribbons. To do so, we performed calculations using the non-equilibrium Green's function method, describing the interaction states through an Anderson model for magnetic impurities together with the so-called non-crossing approximation (NCA) and a generalization to the Meir-Wingreen formula in the non-proportional coupling regime. Thus, through the combination of these methods, we were able to obtain transport phenomena for interacting systems. We observed the emergence of incoherent transmission and current due to the process of quantum-coherent transport phase-break via the presence of electron-electron interaction in both systems. In addition to a more accurate description, we described phenomena not previously described from a purely microscopic perspective.

Keywords: Electronic transport; Graphene quantum dots; Electron-electron interaction; Non-equilibrium Green's functions

Knowledge areas: Physical; Condensed Matter; Electronic Transport.

Lista de Figuras

1.1	Escalas: Comparação das escalas de comprimento entre diferentes estruturas, incluindo aquelas criadas pelo homem e as encontradas na natureza. Imagem retirada da referência [1].	2
1.2	Evolução da escala dos transistores: Gráfico indicando a redução da escala de litografia dos transistores (em nanômetros) em função do ano de produção (em anos). Gerado com dados retirados da referência [2]	3
1.3	Visão esquemática da estrutura das bandas de condução e valência: Regiões em azul representam a banda de condução, e regiões em vermelho as banda de valência, caracterizando as propriedades eletrônicas de diferentes materiais (metais, semicondutores e isolantes) através da posição relativa entre elas e a energia de Fermi. <i>i)</i> Metais, caraterizado pela presença de um overlap entre as bandas de condução e valência, <i>ii)</i> Semicondutores, possuindo um pequeno gap (da ordem de grandeza da energia de Fermi) entre as bandas próximas da energia de Fermi. <i>iii)</i> Isolantes, possuindo um gap entre as bandas, grande em relação ao nível de Fermi.	4
1.4	Visão esquemática de uma héteroestrutura: Ilustração de um dispositivo eletrônico criado utilizando uma heteroestrutura, com um coletor e um emissor de carga (eletrodos) acoplados por uma região “ativa”. A base, constituída de uma material semicondutor. Figura extraída da Ref. [3]	4

1.5	Visão esquemática de um ponto quântico: Visão esquemática de um ponto quântico, onde os reservatórios esquerdo e direito são caracterizados por seus potenciais químicos μ_L e μ_R , respectivamente, acoplados a uma região central contendo níveis de energias discretos $(\dots, E_{N-1}, E_N, E_{N+1}, \dots)$. As propriedades eletrônicas e de transporte do sistema poder ser alteradas aplicando uma diferença de potencial $V_{bias} = \mu_L - \mu_R$ entre os reservatórios e/ou um potencial de <i>gate</i> V_g nos níveis de energia internos. a) Ausência de ressonância entre V_{bias} e os níveis internos. b) Ressonância entre V_{bias} e o nível interno E_N , e c) Ressonância entre V_{bias} e o nível interno E_{N+1} após aplicação de uma diferença de potencial V_g	6
1.6	Diagrama de estabilidade de um transistor de elétron único. Diagrama de estabilidade de um transistor de elétron único de nanofios de silício. A assinatura do bloqueio Coulombiano é o surgimento de curvas em forma de diamante. (a) Dentro dos diamantes, o bloqueio de Coulomb é estabelecido, enquanto fora, uma corrente flui através dos nanofio de silício. (b) Oscilações de Coulomb. Os picos de Coulomb ocorrem em tensões de V_g onde dois diamantes adjacentes se encontram. Imagem retirada da Ref. [4].	7
1.7	Bloqueio de Coulomb: (a) Oscilações do bloqueio de Coulomb de um ponto quântico suspenso, em relação às variações de V_g , (b) Gráfico em cinza, condutância diferencial em função de V_g e diferente de potencial V_{ds} (equivalente a V_{bias} na nomenclatura utilizada nesta dissertação) na presença de um campo magnético, mostrando diamantes de Coulomb. Imagem retirada da Ref. [5].	8
1.8	Diferentes materiais baseados em grafeno: Linha superior, estrutura cristalina de diferentes materiais baseados. Linha inferior, imagens de micrografia dessas diferentes estruturas. Imagem retirada da Ref. [6].	9

1.9	Arranjo experimental de um Transistor de Elétron Único, construído com estruturas baseadas em grafeno: a) Eletrodos SWNT para contato <i>bottom-up</i> com as GNRs. Esquema do conjunto GNR sintetizado em ultra-alto vácuo (UHV, do inglês) paralelo aos terraços de Au(788) (canto superior esquerdo). Esquema do conjunto de eletrodos SWNT paralelo em um substrato de SiO_2 (canto superior esquerdo). Esquema de um transistor baseado em uma única GNR com SWNTs como eletrodos (canto superior direito). Para maior clareza, apenas os GNRs mais próximos do nanogap são mostrados. b) Comportamento de carregamento de um único elétron em um QD (em particular, o QD metálico D3, da Ref. [7]). Escala de cores para a condutância diferencial em função de V_{FG} (equivalente a V_g na nomenclatura utilizada nesta dissertação) e V_{Bias} , com os potenciais de <i>gates</i> laterais $V_{SG1} = V_{SG2} = 4\text{V}$ fixos, mostrando comportamento de elétron único. Figura inferior direita, sendo um zoom ilustrado pela caixa branca, na figura inferior esquerda. Imagens retirada da Ref. [7].	11
2.1	Visão esquemática de estruturas cristalinas: Pontos (representando átomos ou moléculas) ligados por arestas formando estruturas periódicas, chamadas de redes cristalinas, onde $\mathbf{a}_1$, $\mathbf{a}'_1$, $\mathbf{a}_2$ e $\mathbf{a}'_2$ são vetores primitivos de acordo com a descrição da rede de Bravais [8]. a) Exemplo de uma rede quadrada. b) Rede hexagonal, também conhecida como "favo de mel", com A e B constituindo subredes que formam a célula unitária.	18
2.2	Visão esquemática do Modelo Tight-Binding: a) Os círculos azuis representam os sítios da nossa rede (aqui para uma rede 2×2), onde ϵ_1 e ϵ_2 representam as energias dos sítios 1 e 2, respectivamente, e $\gamma_{1,2}$ e $\gamma_{2,1}$ representam o acoplamento entre os sítios 1 e 2 e entre 2 e 1. b) Descrição de uma nanofita com simetria de rede hexagonal, onde $\hat{h}_0$ representa a estrutura de repetição mínima, também chamada de célula unitária, e $\hat{\gamma}_0$ representa o acoplamento entre células unitárias.	22

2.3	Visão esquemática de redes hexagonais: a) Uma representação dentro do espaço de posições, com A e B sendo as subredes e constituindo a célula unitária e $\mathbf{a}_1, \mathbf{a}_2$ os vetores primitivos da rede direta (posição). b) Representação dentro do espaço dos momentos, com $\mathbf{b}_1$ e $\mathbf{b}_2$ os vetores primitivos da rede recíproca (representação dos momentos). Γ, K_-, K_+, M , sendo pontos de simetrias no espaço de momentos.	25
2.4	Densidade de estados: densidade de estados para uma folha de grafeno em função da energia reescalada pelo constante de acoplamento.	27
2.5	Representação de vetores quirais que levam a vários tipos de nanotubos de carbono de parede única (SWNTs): (a) O vetor quiral $\mathbf{C} = 10\mathbf{a}_1 + 5\mathbf{a}_2$ (seta vermelha), conecta dois pontos em uma folha de grafeno e define a circunferência e a quiralidade de um nanotubo (10,5). (b, c, d) Exemplos de nanotubos zig-zag, quiral e armchair, com seus ângulos θ e diâmetros. Imagem retirada da Ref. [9].	27
2.6	Representação de uma nanofita de grafeno: Nanofitas de grafeno são obtidas a partir de uma folha de grafeno (representada em cinza), onde cortes são feitos em uma direção específica. A partir desses cortes, uma faixa estreita de grafeno é extraída, formando assim a nanofita de grafeno.	28
2.7	Célula unitária da nanofita de grafeno armchair: Ilustração da sequência de construção da célula unitária para diferentes famílias armchair. ϵ_i representa a energia do sítio, e são apresentadas as células unitárias para $N_a = 2, 3, 5$ e 8	29
2.8	Célula unitária da nanofita de grafeno zig-zag: Ilustração da sequência de construção da célula unitária para diferentes famílias zig-zag. ϵ_i representa a energia do sítio, e são apresentadas as células unitárias para $N_z = 1, 2, 3$ e 5	29
2.9	Nanofita de Grafeno Infinita: Representação da célula unitária (unidade de repetição mínima) das nanofitas de grafeno para os casos zig-zag, com $N_z = 3$, e armchair, $N_a = 5$. Aqui, γe^{-ika} é a constante de acoplamento com a célula anterior, e γe^{ika} para a próxima célula.	30

- 2.10 **Estruturas de bandas e Densidade de Estados:** Na coluna da esquerda, temos a relação de dispersão em função da energia, e na coluna da direita, temos a densidade de estados. As quantidades foram obtidas para nanofitas armchair (linha superior) e nanofitas (linha inferior) com $N_a = N_z = 4$. Nota-se que a nanofita armchair é semicondutora, enquanto a zig-zag é metálica, apresentando singularidade de van Hove (estados de borda) no nível de Fermi. . 31
- 2.11 **Estruturas de bandas e Densidade de Estados:** Na coluna da esquerda, temos a relação de dispersão em função da energia, e na coluna da direita, temos a densidade de estados. As quantidades foram obtidas para nanofitas armchair (linha superior) e nanofitas (linha inferior) com $N_a = N_z = 9$. Nota-se que a nanofita armchair é semicondutora, enquanto a zig-zag é metálica, apresentando singularidade de van Hove (estados de borda) no nível de Fermi. . 32
- 2.12 **Estruturas de bandas e Densidade de Estados:** Na coluna da esquerda, temos a relação de dispersão em função da energia, e na coluna da direita, temos a densidade de estados. As quantidades foram obtidas para nanofitas armchair (linha superior) e nanofitas (linha inferior) com $N_a = N_z = 30$. Nota-se que a nanofita armchair é semicondutora, enquanto a zig-zag é metálica, apresentando singularidade de van Hove (estados de borda) no nível de Fermi. . 33
- 2.13 **Densidade de estados para o sistema 2×2 :** densidade de estados em função de $E - \epsilon_0$ para dois orbitais atômicos *a*) desacoplados, $\gamma_{i,j} = 0$, e *b*) acoplados, com $\gamma_{i,j} = \gamma$. Aqui, tomamos $\eta \approx 10^{-4}$. . . 36
- 2.14 **Visão esquemática de um problema de transporte eletrônico:** As regiões em vermelho representam os eletrodos, onde $\hat{H}_L$ e $\hat{H}_R$ denotam os Hamiltonianos do eletrodo esquerdo e direito, respectivamente. Em cinza, temos a região de espalhamento, com $\hat{H}_S$ o seu Hamiltoniano e o acoplamento entre diferentes regiões representado por $\hat{\gamma}_{l,l'}$, onde $l = L, S, R$ 39

2.15	Visão esquemática do Método da função de Green recursiva: a) Uma cadeia linear infinita, onde os Hamiltonianos dos sítios $\hat{h}_m$ acoplam apenas com seus vizinhos próximos, e $\hat{\gamma}_{m,n}$ é o acoplamento. b) Representação esquemática do processo recursivo. Temos, a princípio, apenas um sítio $\hat{h}_1$ com sua função de green associada $\hat{g}_1$. Depois, ele é acoplado com o sítio posterior, obtendo a função de green efetiva entre os dois sítios $\hat{g}_1^2$. Esse processo é repetido novamente.	42
2.16	Funções de Green semi-infinita: Na linha superior, temos o negativo da parte imaginária da função de Green ($-\text{Im}[g]$), e na linha inferior, a parte real ($\text{Re}[g]$), em função da energia para uma cadeia atômica linear.	46
2.17	Auto-energias para uma nanofita de grafeno: Na linha superior tem o negativo da parte imaginária do traço sobre as auto-energias ($\text{Im}[\Sigma_{L/R}]$) e na linha inferior a parte real ($\text{Re}[\Sigma_{L/R}]$) em função da energia para uma nanofita semi-infinita de grafeno. Com a linha vermelha e preta referentes ao termos esquerdo (Σ_L) e direito (Σ_R) respectivamente, para $N_a = 20$	47
2.18	Auto-energias para uma nanotubo de carbono: Na linha superior, temos o negativo da parte imaginária do traço sobre as auto-energias ($\text{Im}[\Sigma_{L/R}]$), e na linha inferior, a parte real ($\text{Re}[\Sigma_{L/R}]$), em função da energia para um nanotubo de carbono semi-infinito. A linha vermelha e preta referem-se aos termos esquerdo (Σ_L) e direito (Σ_R), respectivamente, com $N_{cz} = 18$	48
2.19	Resistência e Condutância em função da temperatura: (a) Resistência em função da temperatura para um metal normal (curva em azul), um supercondutor (curva em verde) e um metal com impurezas magnéticas (curva em vermelho). b) Condutância para um ponto quântico semiconductor, com número de elétrons confinados no ponto sendo ímpar (curva vermelha) e par (curva azul). Observa-se o efeito Kondo: (a) o aumento da resistência em um metal dopado com impurezas magnéticas e (b) o aumento da condutância em um ponto quântico com número ímpar de elétrons confinados. Em ambos os casos, o fenômeno surge para o regime de baixas temperaturas [10]. Figura retira da Ref. [10]	52

2.20	Visão esquemática do Modelo de Anderson para impurezas: As regiões em cinza representam os banhos de elétrons, enquanto as regiões em vermelho representam a região de impurezas e as regiões em branco representam as barreiras de potencial. a) Descrição contendo múltiplos níveis internos, com níveis semi-preenchidos e completamente preenchidos, onde $\hat{c}_{k,\nu,\sigma}^\dagger$ ($\hat{c}_{k,\nu,\sigma}$) é o operador de criação (aniquilação) dos elétrons no banho, $\hat{f}_{k,\nu,\sigma}^\dagger$ ($\hat{f}_{k,\nu,\sigma}$) o operador de criação (aniquilação) dos elétrons na impureza, e $\hat{\gamma}_{k,\alpha}$ é o acoplamento entre as regiões. ϵ_α é a energia do nível α , U é o potencial de interação entre elétrons no mesmo nível, U' o termo de interação entre elétrons em estados diferentes. b) Descrição do caso de um único nível, c) e d) descrição para o caso de dois níveis.	54
2.21	Visão esquemática de um problema de transporte eletrônico na estrutura de uma molécula estendida: As regiões em vermelho representam os eletrodos, com $\hat{H}_L$ e $\hat{H}_R$ sendo os Hamiltonianos dos eletrodos esquerdo e direito, respectivamente. A região de espalhamento é dividida em sub-regiões, com $\hat{H}_I$ representando a região de interação, $\hat{H}_{EL/(ER)}$ representando a região à esquerda (direita) da região de interação. Todas essas regiões, juntamente com $\hat{H}_{\alpha\alpha/(\beta\beta)}$, compõem o que chamamos de região de molécula estendida.	59
2.22	Visão esquemática do ciclo autoconsistente: Ilustração do ciclo autoconsistente passo a passo adotado nesta dissertação. Aqui, o ciclo é uma extensão do DMFT usual.	79
2.23	Visão esquemática do problema de transporte eletrônico: $\mu_{L/(R)}$ e $T_{L/(R)}$ são o potencial químico e a temperatura, respectivamente do eletrodo esquerdo (direito), com uma diferença de potencia V_{bias} aplicada entre os eletrodos. a e b as amplitudes de entrada e saída respectivamente, da região de espalhamento.	80
2.24	Visão esquemática do problema de de Büttiker: Análogo ao problema de espalhamento, temos $\mu_{L/(R)}$ e $T_{L/(R)}$ os potencial químico e a temperatura, respectivamente do eletrodo esquerdo (direito), μ_ϕ e T_ϕ potencial químico e a temperatura do eletrodo fictício a e b as amplitudes de entrada e saída respectivamente, da região de espalhamento.	85

3.1	Lado (a), topo (b) e (c), um zoom da sobreposição entre as estruturas, vistas do dispositivo usado em nossos cálculos.: SWNTs como eletrodos direito e esquerdo, acoplados por uma NR para formar uma junção SWNT-NR-SWNT. O acoplamento com os eletrodos ocorre de maneira comensurada através de três células unitárias na NR de cada lado.	97
3.2	Densidade de Estados para a NR desacoplada: Densidade de Estados para a NR desacoplada (curva tracejada preta) e o módulo quadrado das funções de onda para os dois primeiros estados moleculares simétricos em relação ao nível de Fermi. $E_1 = -0.45$ eV, $E_2 = 0.45$ eV.	98
3.3	Densidade de Estados para a NR: Densidade de estados para a NR desacoplada (curva tracejada preta), densidade de estados para a NR acoplada (curva sólida vermelha) e o módulo quadrado das funções de onda para os dois estados moleculares degenerados no nível de Fermi. $E_{01} \approx E_{02} \approx 0$ eV.	102
3.4	Função de Hibridização: <i>a)</i> A parte real e <i>b)</i> o negativo da parte imaginária da função de hibridização para as impurezas de borda na ausência de interação. A curva tracejada preta refere-se a ψ_{AI}^{01} e a curva tracejada vermelha refere-se a ψ_{AI}^{02}	103
3.5	Densidade de estados para impurezas de borda: Densidade de estados para os estados de borda no nível de Fermi. Para o caso não interagente (curva tracejada) e para o caso interagente (curva sólida) com $U_z = 2$. A curva preta refere-se a ψ_{AI}^{01} e a curva vermelha a ψ_{AI}^{02}	104
3.6	Densidade de estados para impurezas de Anderson: Densidade de estados para os dois primeiros estados próximos ao nível de Fermi. Na ausência de interação <i>a)</i> , temos o estado duplamente ocupado (curva tracejada preta) e o estado com ocupação zero (curva tracejada vermelha). Na presença de interação ($U = 0.45$) <i>b)</i> , representado na figura por curvas sólidas, observamos o estado com ocupação zero em $E = 0.45$ (curva sólida vermelha), o estado com única ocupação em $E = -0.45$ (curva sólida preta) e o estado duplamente ocupado em $E = 0$ (curva sólida preta).	105

- 3.7 **Transmissão em função da energia:** A transmissão para o nanotubo (curva tracejada laranja) e para a nanofita (curva tracejada roxa) estão presentes como guias para comparação. Na configuração não interagente SWNT-NR-SWNT (curva sólida preta). A transmissão interagente (curva sólida verde), coerente (curva sólida vermelha) e incoerente (curva sólida azul) são apresentadas para a configuração SWNT-NR-SWNT interagente com potencial Coulombiano $U = 0.45\text{eV}$ 106
- 3.8 **a) Densidade de estados, b) Transmissão efetiva, c) Transmissão coerente e transmissão incoerente para múltiplos gates, e d) Corrente elétrica:** a) Temos a densidade de estados para os dois primeiros estados próximos ao nível de Fermi para valores de *gate* crescentes (diferentes colunas). Para o caso sem interação, a curva tracejada preta refere-se a ψ_{AI}^1 com dupla ocupação e a curva tracejada vermelha refere-se a ψ_{AI}^2 com ocupação zero. Na presença de interação com $U = 0.45\text{eV}$, a curva sólida preta refere-se a ψ^1_{AI} para $V_g < 0$ com dupla ocupação e para $V_g \geq 0$ com quebra de degenerescência e assumindo ocupação única e ocupação dupla para diferentes valores de energia. A curva sólida vermelha refere-se a ψ^2_{AI} com ocupação zero. b) e c) Temos a transmissão em função da energia para diferentes valores de *gate* (diferentes colunas), com a transmissão para o nanotubo (curva tracejada laranja) e nanofita (curva tracejada roxa) presentes como guias para comparação. Na configuração não interagente SWNT-NR-SWNT (curva sólida preta) e a transmissão efetiva (curva sólida verde) para interagente SWNT-NR-SWNT com potencial Coulombiano $U = 0.45\text{ eV}$. c) Transmissão coerente (curva sólida vermelha) e a transmissão incoerente (curva sólida azul) para interagente SWNT-NR-SWNT com potencial Coulombiano $U = 0.45\text{ eV}$. d) Comparação entre os termos de corrente efetiva (curva verde), o termo de corrente coerente (curva vermelha) e o termo de corrente incoerente (linha azul), para $U = 0.45\text{ eV}$ em função de V_{bias} para diferentes valores de *gate*. 107
- 3.9 **Corrente local:** Corrente local por sítio i , $i_i(E) = \sum_j (i_{ij}(E) \hat{r}_{ij} + i_{ji}(E) \hat{r}_{ji})$ para $E = 0\text{ eV}$, $V_g = 0\text{ eV}$ e $V_{bias} = 0.02\text{ eV}$. a) Na ausência de interação, e b) na presença de interação com $U = 0.45\text{ eV}$. 109

3.10	Diagrama de instabilidade para corrente e diferença entre os termos de corrente, $I_{\text{coh}} - I_{\text{incoh}}$: a) O diagrama de instabilidade para corrente, com $U = 0\text{eV}$, mostra o valor absoluto da corrente em função do <i>gate</i> (eixo x) e da polarização (eixo y). b) O diagrama de instabilidade para corrente efetiva, com $U = 0,45\text{eV}$, mostra o valor absoluto da corrente em função do <i>gate</i> (eixo x) e da polarização (eixo y). c) O diagrama de diferença entre os termos absolutos das correntes, com $U = 0,45\text{eV}$, é mostrado em função do <i>gate</i> (eixo x) e do viés (eixo y). Na região vermelha, $ I_{\text{coh}} $ é o termo dominante, na região azul, $ I_{\text{incoh}} $ é o termo dominante e na região branca, temos $ I_{\text{coh}} = I_{\text{incoh}} $.	110
4.1	Visão esquemática do dispositivo usado em nossos cálculos: Na região central, uma nanofita de grafeno (NR) armchair com vacâncias, e nas regiões em vermelho, os eletrodos esquerdo e direito. As regiões são fortemente acopladas entre si.	113
4.2	Transmissão e funções de onda para NR: A transmissão para a NR perfeita (curva tracejada roxa), a transmissão na presença de uma única vacância (curva preta) e o módulo quadrado das funções de onda para os dois estados moleculares próximos ao nível de Fermi. $E_1 \approx E_2 \approx 0\text{ eV}$. Por fim, o quadro em preto está aqui para ilustrar em qual orbital adicionamos a vacância.	115
4.3	Divisões das sub-regiões da NR: Visão das subdivisões feita sobre a NR conforme a estrutura da molécula estendida (Ver Sec. 2.7). Na região central da NR, temos a região de interação $\hat{H}_I$, que contém a vacância, $\hat{H}_\alpha$ ($\hat{H}_\beta$) a região à esquerda (direita) da região de interação, que não irá conter interação, e nas regiões em vermelho, os eletrodos esquerdo e direito. As regiões são fortemente acopladas entre si.	115
4.4	Densidade de Estados para a NR: a) Densidade de Estados para uma nanofita de grafeno na ausência de defeitos (curva tracejada roxa) e na presença de uma MI (curva preta). b) O módulo quadrado da função de onda para o estado molecular localizado no nível de Fermi ($E_1 \approx 0\text{ eV}$).	118

4.5	Densidade de estados para impureza magnética: densidade de estados para os estados de MI na representação de partícula única. (a) O estado duplamente ocupado (curva tracejada preta) na ausência de interação e (b) o estado com ocupação única em $E = 0\text{eV}$ e o estado duplamente ocupado em $E = 3.25\text{eV}$ na presença de interação, com $U = 3.25\text{eV}$ (curva sólida vermelha).	119
4.6	Transmissão em função da energia: a transmissão para a MI não interagente (curva tracejada preta), a transmissão efetiva (curva sólida verde), a transmissão coerente (curva sólida vermelha) e a transmissão incoerente (curva sólida azul) para a MI interagente com potencial Coulombiano $U = 3.25\text{eV}$. A transmissão para a nanofita na ausência de defeito (curva tracejada roxa) está presente como guia para comparação.	120
4.7	Média da transmissão em função da energia: Média Transmissão para a NR na ausência de defeitos (curva tracejada preta) e na presença de diferentes concentrações de vacâncias <i>i</i>) $d = 0.01$ (curva sólida anil) e <i>ii</i>) $d = 0.02$ (curva sólida laranja) na ausência de interação.	121
4.8	Rede e Densidade de Estados para a NR: A configuração da rede na presença de duas vacâncias e a densidade de estados associada à função de onda dos dois estados de impureza molecular para <i>a</i>) $L_v = 1$ e <i>b</i>) $L_v = 13$	121
4.9	Transmissão e função de hibridização para a NR não interagente: Na presença de uma única vacância, <i>a</i>) a transmissão e <i>b</i>) a função de hibridização. Ambos para diferentes configurações de impureza (diferentes colunas) com o quadro em preto ilustrando a configuração escolhida para aquela coluna. A transmissão para a nanofita na ausência de defeito (curva tracejada roxa) está presente como guia para comparação.	122

4.10 Transmissão em função da energia: <i>a)</i> transmissão na presença de uma única vacância não interagente (curva tracejada preta) e a transmissão efetiva (curva sólida verde), <i>b)</i> contribuições da transmissão coerente (curva sólida vermelha) e da transmissão incoerente (curva sólida azul) na presença de uma única vacância interagente com potencial Coulombiano $U = 3.25\text{eV}$. A transmissão para a nanofita na ausência de defeito (curva tracejada roxa) está presente como guia para comparação.	124
--	-----

Índice

Lista de Figuras	viii
1 Introdução	1
1.1 Nanociência	2
1.1.1 Transistores e pontos quânticos	4
1.1.2 Interação Elétron-Elétron	7
1.2 Grafeno	9
1.2.1 Nanoestruturas baseadas em grafeno	10
1.3 Estado da arte em transporte eletrônico	12
1.4 Estrutura da dissertação	15
2 Métodos	17
2.1 Estrutura Cristalina	18
2.2 Modelo Tight-binding	22
2.3 Grafeno	24
2.3.1 Nanotubos de Carbono e Nanofitas de Grafeno	27
2.4 Funções Green	34
2.4.1 Caso 2×2	35
2.5 Sistemas Abertos	38
2.5.1 Método da função de Green recursiva	42
2.5.2 Caso 3×3	43
2.6 Interação elétron-elétron	48
2.6.1 Modelo de Hubbard	48
2.6.2 Representação de pseudo partícula	55
2.7 Projeção sobre estados localizados	59
2.8 Teoria do campo médio dinâmico (DMFT)	78
2.9 Transporte Quântico Decoerente	80
2.9.1 Transporte Coerente: Formalismo de Landauer-Büttiker	81
2.9.2 Fenômeno de Decoerência: O problema de Büttiker	85
2.9.3 Transporte Interagente: Generalização da expressão de Meir-Wingreen	87
2.10 Conclusão	93

3	Transistor de elétron único	95
3.1	Metodologia	96
3.2	Resultados	102
3.3	Conclusão	110
4	Nanofita de Grafeno	112
4.1	Metodologia	113
4.2	Resultados	117
4.2.1	Impureza Magnética Única	118
4.2.2	Desordem por Vacância	121
4.2.3	Vacância única	123
4.3	Conclusão	124
5	Conclusão Geral e Perspectivas	126
	Referências	127
A	Script	138

Capítulo 1

Introdução

Historicamente, a introdução de novas tecnologias nos meios de produção sempre impulsionou mudanças significativas na sociedade [11]. O período de desenvolvimento e utilização de tecnologias como rodas d'água e turbinas hidráulicas [12], até o domínio e aplicação do carvão e vapor nos processos industriais [13], é chamado de 'Primeira Revolução Industrial' (entre os anos de 1760 a 1830) [13]. Entretanto, enquanto a primeira revolução industrial foi iniciada por avanços em um único campo, o período conhecido como 'Segunda Revolução Industrial', após 1870, foi o resultado de avanços tecnológicos em múltiplas áreas, como fertilizantes, eletricidade e transportes [14], o que levou ao desenvolvimento do motor de combustão interna baseado em gás e ar [14]. Analogamente, houve uma nova ruptura após 1960 com o desenvolvimento de tecnologias como transistores e microchips [15], possibilitando a confecção de eletrônicos e a automação industrial [15], o que levou à Terceira Revolução Industrial [15].

Devido a diversos fatores, em particular, a necessidade de redução no consumo de energia, a diminuição do tamanho dos dispositivos eletrônicos, o avanço no poder de processamento de computadores e a necessidade de processamento da alta taxa de dados existentes, a humanidade está sendo levada, uma vez mais, a um novo processo de revolução industrial [11]. A primeira menção a uma 'Quarta Revolução Industrial' foi feita na Alemanha durante a 'Hannover Fair' (uma conferência dedicada ao tema do desenvolvimento da indústria) em 2011, por Klaus Schwab [16]. Ao grande público este termo foi apresentado em 2015 no Fórum Econômico Mundial [16]. Este novo período também está sendo conhecido como a era da Indústria 4.0 [11], onde tecnologias como computação, biotecnologia, energia, ciência dos materiais e nanotecnologia (com as duas últimas áreas de interesse desta dissertação) desempenham um papel fundamental.

1.1 Nanociência

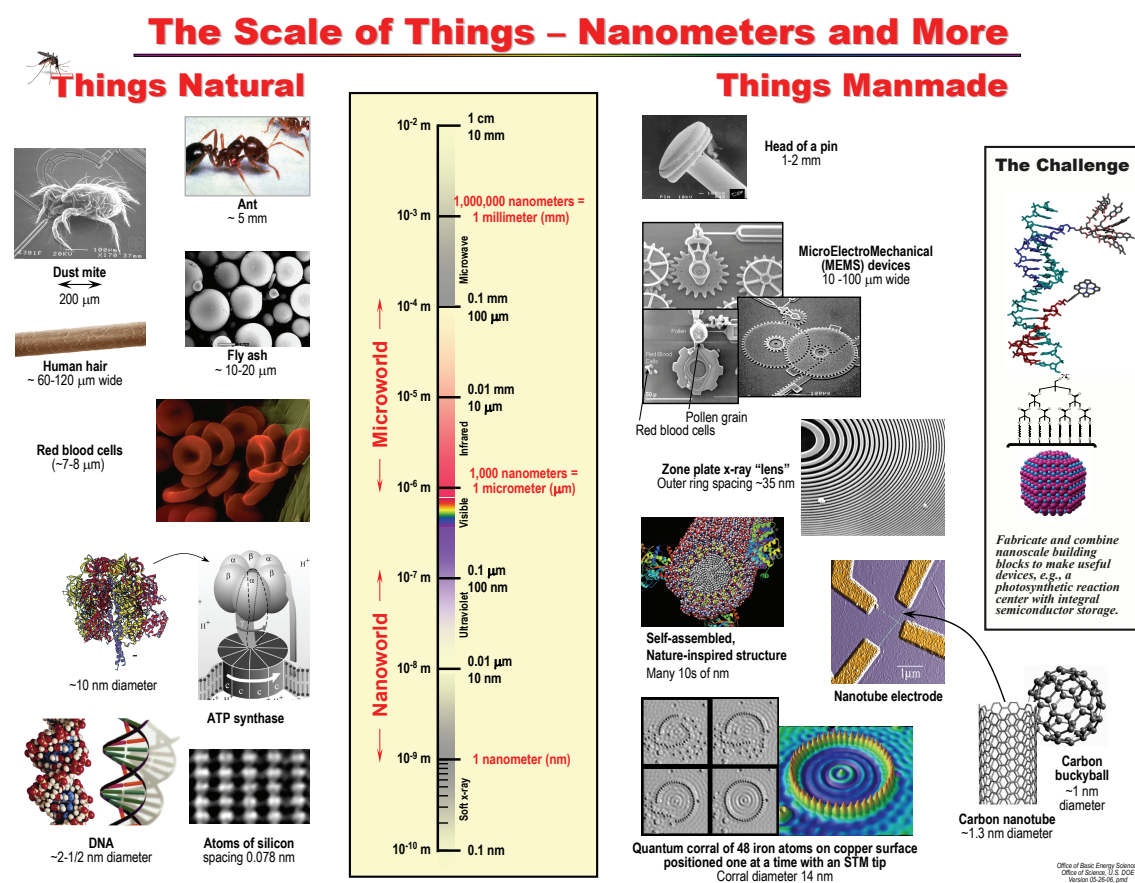


Figura 1.1: Escalas: Comparação das escalas de comprimento entre diferentes estruturas, incluindo aquelas criadas pelo homem e as encontradas na natureza. Imagem retirada da referência [1].

A necessidade de otimização do consumo energético, juntamente com a busca pela redução da escala dos dispositivos eletrônicos, provocou o surgimento de uma nova área do conhecimento conhecida como Nanociência, um dos pilares da Indústria 4.0. Esta ciência consiste na produção de conhecimento e produtos relacionados a fenômenos na escala de nanômetros (10^{-9} metro), como mostrado na Fig.1.1. Não é à toa que a atenção dos físicos, tanto experimentais quanto teóricos, nos últimos anos, tem se voltado para essa área em ascensão. De particular interesse desta dissertação está a possibilidade de construir nanodispositivos baseados em estruturas de carbono, como nanotubos [17] e, mais recentemente, o grafeno [18, 19], e caracterizá-los.

Um dos dispositivos de grande interesse na indústria e na comunidade acadêmica são os transistores. Propostos inicialmente em 1926 por Lilienfeld [20],

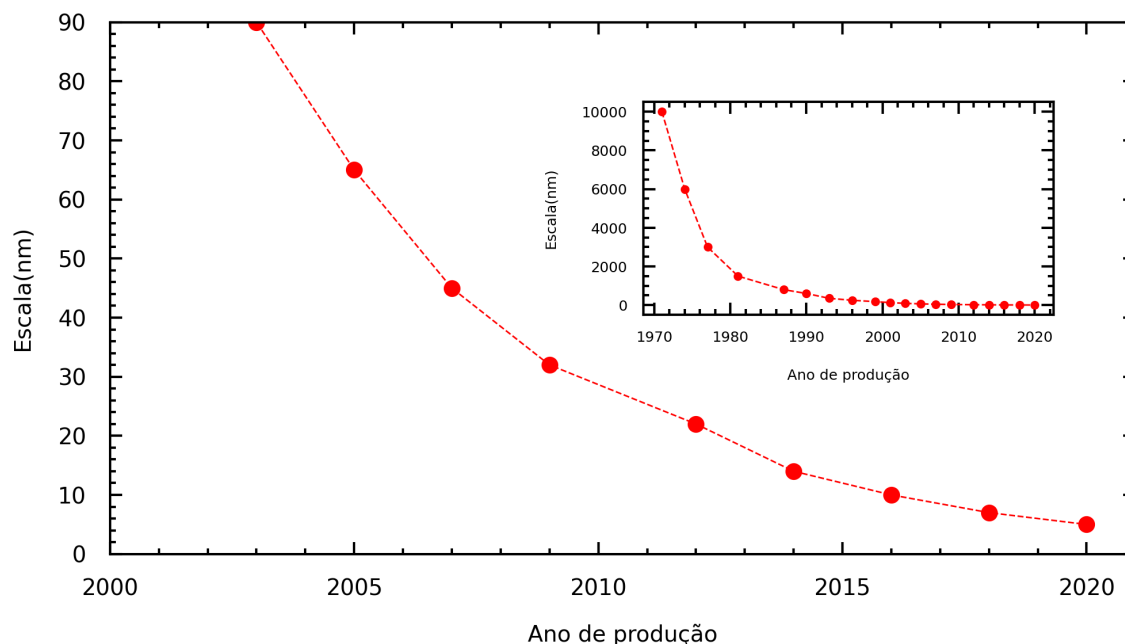


Figura 1.2: **Evolução da escala dos transistores:** Gráfico indicando a redução da escala de litografia dos transistores (em nanômetros) em função do ano de produção (em anos). Gerado com dados retirados da referência [2]

foram inventados por Bardeen, Brattain e Shockley [3] no Bell Labs e receberam o Prêmio Nobel em 1956. Eles rapidamente entraram em produção e substituíram as válvulas de vácuo, dando origem à era da eletrônica. Conforme aplicações mais complexas foram sendo elaboradas, Moore observou em 1965 [21] que o número de transistores em dispositivos eletrônicos estava dobrando a cada dois anos. Analogamente, como podemos observar na Figura 1.2, a escala dos transistores vem diminuindo exponencialmente, de acordo com a mesma lei de Moore para o número de dispositivos [2]. Este é, obviamente, um efeito notório; conforme tentamos colocar o mesmo número de transistores em um microchip, o tamanho individual de cada transistor deve diminuir para manter o tamanho do chip mais ou menos constante. Como consequência disso, nos últimos anos chegamos a dispositivos na escala de alguns nanômetros, trazendo novos desafios, tanto de uma perspectiva de produção quanto teórica. Neste último, principalmente devido ao fato de que nessa escala, as flutuações quânticas regem as propriedades macroscópicas dos sistemas, demandando uma descrição teórica realística em uma abordagem microscópica.

1.1.1 Transistores e pontos quânticos

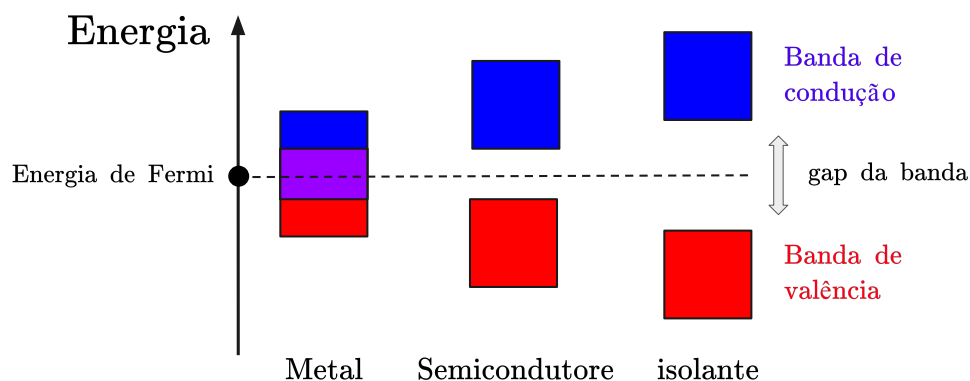


Figura 1.3: **Visão esquemática da estrutura das bandas de condução e valência:** Regiões em azul representam a banda de condução, e regiões em vermelho as banda de valência, caracterizando as propriedades eletrônicas de diferentes materiais (metais, semicondutores e isolantes) através da posição relativa entre elas e a energia de Fermi. *i)* Metais, caracterizado pela presença de um overlap entre as bandas de condução e valência, *ii)* Semicondutores, possuindo um pequeno gap (da ordem de grandeza da energia de Fermi) entre as bandas próximas da energia de Fermi. *iii)* Isolantes, possuindo um gap entre as bandas, grande em relação ao nível de Fermi.

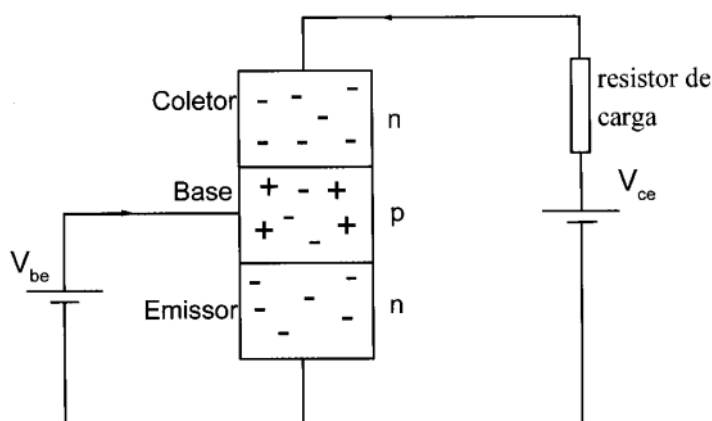


Figura 1.4: **Visão esquemática de uma heteroestrutura:** Ilustração de um dispositivo eletrônico criado utilizando uma heteroestrutura, com um coletor e um emissor de carga (eletrodos) acoplados por uma região "ativa". A base, constituída de um material semicondutor. Figura extraída da Ref. [3]

Tipicamente, os transistores são fabricados com materiais semicondutores. A compreensão do transistor “clássico” pode ser obtida por meio da teoria de bandas [22]. Nos metais, o nível de Fermi está localizado no meio da banda e, portanto, não há diferenças significativas de energia entre os estados da banda de valência e de condução. Por outro lado, nos isolantes, a diferença entre as bandas é grande em relação ao nível de Fermi (ver Fig.1.3), o que torna necessário aplicar altas diferenças de potencial (da ordem de dezenas de elétrons volts) para excitar elétrons entre as bandas. Os semicondutores, por sua vez, podem ser classificados como materiais cujo gap entre as bandas é relativamente pequeno, assim, aplicar pequenas diferenças de potencial (da ordem de algumas unidades de elétrons volts) é suficiente para mover elétrons da banda de valência para a de condução (ver Fig.1.3). Devido às propriedades dos semicondutores, J. Bardeen e W. Brattain observaram que ao construir uma heteroestrutura, ou seja, um dispositivo caracterizado por diferentes tipos de materiais em contato, como ilustrado na Fig.1.4, sendo a base o material semicondutor (como o germânio), ocorre o “efeito transistor”, que consiste na amplificação de um sinal elétrico aplicado a um eletrodo [23] e controlado por uma tensão de *gate* V_g (V_{be} na Fig. 1.4). Inicialmente, este dispositivo foi chamado de “transfer resistor”, já que um transistor é responsável por transferir a corrente de um circuito elétrico para outro, como ilustrado na Fig.1.4.

Devido à atual escala dos nanodispositivos, torna-se necessário, na confecção de dispositivos, considerar fenômenos de natureza quântica. Em particular, a miniaturização faz com que o número e o arranjo de átomos se torne um fator importante, de modo que o conceito de bandas não pode mais ser aplicado em todo o dispositivo, ao mesmo tempo, em que os efeitos quânticos se tornam cada vez mais importantes. Entre as décadas de 1960 e 1970 novas técnicas experimentais desenvolvidas permitiram a construção de redes cristalinas artificiais com interfaces quase perfeitas, onde os efeitos de quantização são essenciais. Em paralelo, houve também a síntese de pontos quânticos (QD, da sigla em inglês), integrados a dispositivos eletrônicos e ópticos. Este último feito (no caso de pontos quânticos orgânicos) levou Mounqi G. Bawendi, Louis E. Brus e Aleksey Yekimov a serem laureados com o Prêmio Nobel de Química de 2023. Pontos quânticos pode ser considerados “átomos artificiais” uma vez que são estruturas (quase)-0D e apresentam níveis de energias discretos. Assim, os QDs surgem como uma boa alternativa para a confecção de transistores quânticos, onde o número de elétrons passando pelo dispositivo é pequeno, podendo chegar ao limite unitário.

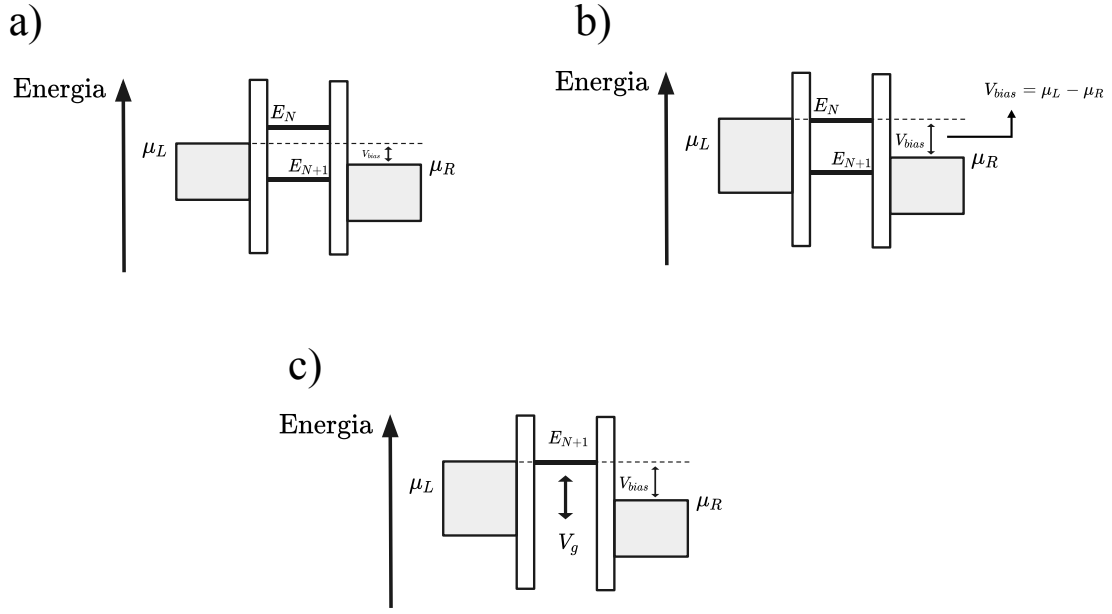


Figura 1.5: **Visão esquemática de um ponto quântico:** Visão esquemática de um ponto quântico, onde os reservatórios esquerdo e direito são caracterizados por seus potenciais químicos μ_L e μ_R , respectivamente, acoplados a uma região central contendo níveis de energias discretos $(\dots, E_{N-1}, E_N, E_{N+1}, \dots)$. As propriedades eletrônicas e de transporte do sistema poder ser alteradas aplicando uma diferença de potencial $V_{bias} = \mu_L - \mu_R$ entre os reservatórios e/ou um potencial de *gate* V_g nos níveis de energia internos. a) Ausência de ressonância entre V_{bias} e os níveis internos. b) Ressonância entre V_{bias} e o nível interno E_N , e c) Ressonância entre V_{bias} e o nível interno E_{N+1} após aplicação de uma diferença de potencial V_g .

De maneira simplificada, um QD aberto pode ser descrito por um sistema subdividido em três regiões, como ilustrado na Fig. 1.5. Com μ_L (μ_R) representando o reservatório de elétrons à esquerda (direita) da região central, acoplado por meio de uma barreira de potencial, e na região central contendo níveis discretos de energia, a interação Coulombiana entre elétrons dentro do QD faz com que exista apenas corrente elétrica entre os reservatórios quando a energia dos estados discretos está em ressonância com a diferença de potencial aplicada entre os eletrodos (V_{bias}). A corrente elétrica é nula caso contrário. Este efeito é conhecido como bloqueio de Coulomb [22]. Podemos ainda aplicar uma diferença de potencial nos níveis internos por meio de um campo perpendicular (V_g) para acessar níveis de energia com diferentes valores de ocupação eletrônica e, conseqüentemente, obter corrente elétrica para diferentes valores de V_g , como ilustrado nas Fig. 1.5, Fig. 1.6 e Fig. 1.7. A assinatura do bloqueio de Coulomb é o chamado diamante de Coulomb, que surge no diagrama de fase (G, V_{bias}, V_g) , conforme ilustrado

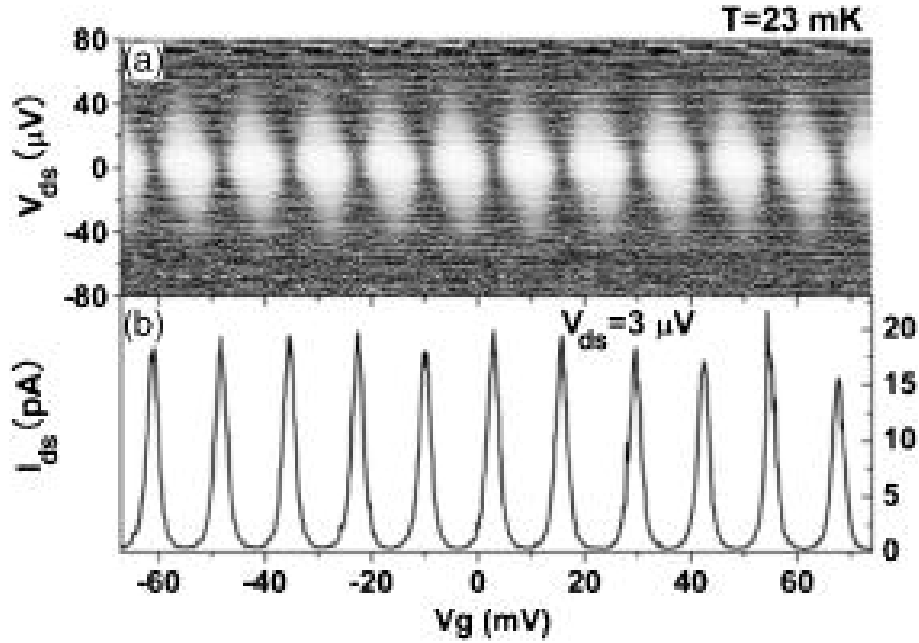


Figura 1.6: **Diagrama de estabilidade de um transistor de elétron único.** Diagrama de estabilidade de um transistor de elétron único de nanofios de silício. A assinatura do bloqueio Coulombiano é o surgimento de curvas em forma de diamante. (a) Dentro dos diamantes, o bloqueio de Coulomb é estabelecido, enquanto fora, uma corrente flui através dos nanofio de silício. (b) Oscilações de Coulomb. Os picos de Coulomb ocorrem em tensões de V_g onde dois diamantes adjacentes se encontram. Imagem retirada da Ref. [4].

na Fig. 1.7. Descrever de maneira teórica esse comportamento para dispositivos baseados em redes cristalinas é um desafio, ao ser necessário descrever os níveis de energia internos de maneira precisa. Para isso, é necessária uma descrição atômica considerando os níveis de energia e, mais importante, a interação entre os elétrons internos ao material, o que é tipicamente desconsiderado. Assim, a presença de interação elétron-elétron e suas consequências em nanodispositivos, especificamente em nanodispositivos baseados em semicondutores, ainda é um problema em aberto.

1.1.2 Interação Elétron-Elétron

Uma das quatro forças fundamentais da natureza é regida pela Lei de Coulomb, que descreve a interação entre partículas eletricamente carregadas. Essa lei se revelou crucial para a compreensão da natureza tal como a conhecemos. Em particular, quando consideramos a presença de elétrons - partículas com carga elétrica negativa -, eles interagem entre si por meio da interação Coulombiana

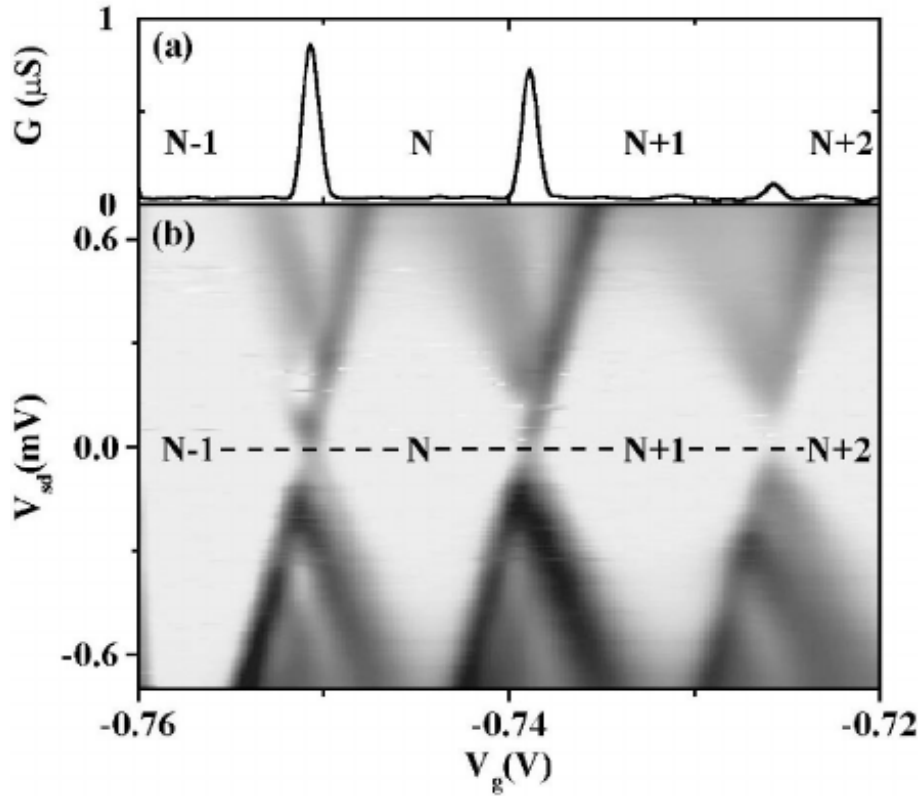


Figura 1.7: **Bloqueio de Coulomb**: (a) Oscilações do bloqueio de Coulomb de um ponto quântico suspenso, em relação às variações de V_g , (b) Gráfico em cinza, condutância diferencial em função de V_g e diferente de potencial V_{ds} (equivalente a V_{bias} na nomenclatura utilizada nesta dissertação) na presença de um campo magnético, mostrando diamantes de Coulomb. Imagem retirada da Ref. [5].

repulsiva, frequentemente chamada de interação elétron-elétron. Essa interação é essencial para descrever a natureza dos níveis de energia atômicos em uma perspectiva microscópica [24].

No caso de elétrons livres em materiais, que podem ser tratados como um sistema de muitos corpos, uma abordagem simplificada os trata como partículas não interagentes (ou interagentes por meio de um potencial efetivo) [24]. Isso é válido em diversos semicondutores e na maioria dos metais, onde a energia de interação Coulombiana entre elétrons é muito menor do que sua energia cinética [25]. Nesses casos, teorias como a teoria das bandas se mostram eficazes. Essa teoria é uma metodologia baseada em uma aproximação de campo médio, em que se supõe que cada elétron se move no potencial médio do cristal, uma média da interação de todos os elétrons no sólido. O sucesso da física da matéria condensada foi amplamente impulsionado pela teoria das bandas, que simplifica o problema ao tratar cada elétron de forma independente.

No entanto, essa abordagem não é necessariamente válida para sistemas com interações entre elétrons. Uma das consequências mais importantes dessas interações é a correlação eletrônica, na qual o movimento de um elétron afeta diretamente o comportamento dos demais [25]. Em dispositivos nanométricos, isso pode levar a fenômenos como o bloqueio de Coulomb (já mencionado na Sessão anterior), no qual a presença de um elétron em uma região impede a passagem de outros elétrons devido à repulsão eletrostática [25]. Essa interação pode resultar em características não-lineares no transporte de corrente, como patamares de condutância discretos, e até mesmo na quebra da coerência de fase eletrônica.

Como mencionado anteriormente, a compreensão da interação elétron-elétron e suas consequências em nanodispositivos continua sendo um desafio em aberto na física da matéria condensada.

1.2 Grafeno

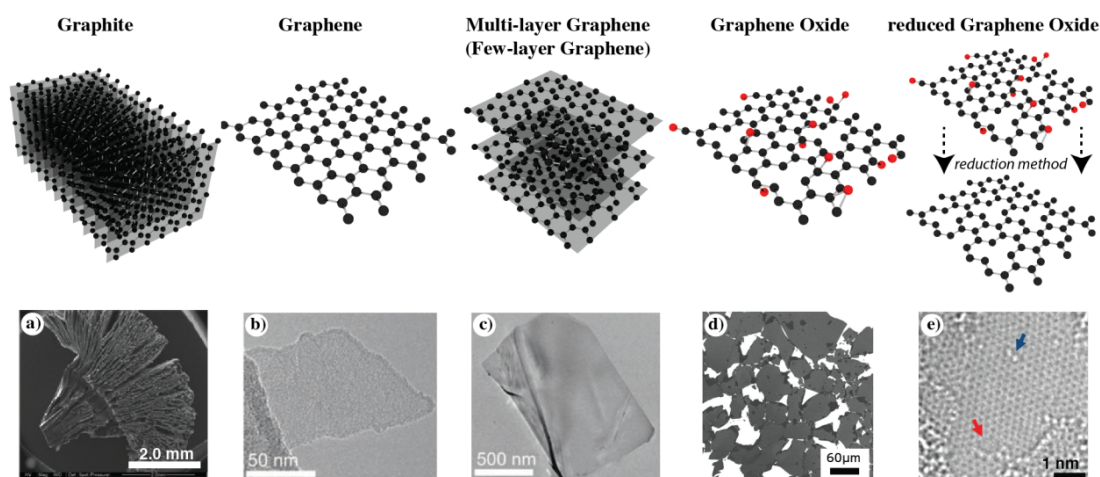


Figura 1.8: **Diferentes materiais baseados em grafeno:** Linha superior, estrutura cristalina de diferentes materiais baseados. Linha inferior, imagens de micrografia dessas diferentes estruturas. Imagem retirada da Ref. [6].

Um dos materiais de grande interesse na comunidade, devido às suas "maravilhosas propriedades", é o grafeno. Este material, inicialmente considerado impossível de existir, teve sua produção bem-sucedida apresentada em meados de 2004 [18]. Este trabalho rendeu, seis anos depois, em 2010, o Prêmio Nobel a Andre Geim e Konstantin Novoselov por "experimentos inovadores sobre o material bidimensional grafeno". Isso impulsionou, após 2004, um aumento exponencial na produção de trabalhos envolvendo os tópicos de "Graphene" e "production

of Graphene"[26], e fez surgir uma nova área de física de materiais voltada ao estudo de sistemas bidimensionais. Isso ocorre principalmente devido às suas propriedades e à possibilidade de aplicação em novas tecnologias.

O grafeno consiste em uma monocamada de carbono arranjada em uma rede cristalina em duas dimensões (2D) com a estrutura de favo de mel (*honeycomb* em inglês), como ilustrado na Fig. 1.8. Uma das desvantagens, entretanto, é a falta de um *gap* eletrônico, dificultando uma série de aplicações em eletrônica. Nesse sentido, avanços experimentais permitiram a fabricação de nanofitas de grafeno (NR). Essas estruturas são tiras de grafeno quase unidimensionais e podem ser fabricadas com comprimentos variados e, tipicamente, são semicondutoras com um *band gap* que pode ser controlado pela largura e altura da fita [27]. Devido a isso, elas surgiram como uma alternativa para a confecção de nanotransistores [7, 28, 29, 30, 31]. Enquanto métodos chamados *top-down*, que utilizam litografia e *etching*, produzem fitas com corrugações nas bordas, recentemente NR derivadas quimicamente, utilizando métodos chamados *bottom-up*, foram obtidas [32, 33]. Estas nanofitas são atômicamente precisas.

Um novo desafio, no caso de nanoestruturas, em geral, e em grafeno em particular, é a produção de amostras com alto grau de "pureza". É comum na produção em larga escala a existência de defeitos, sendo eles vacâncias ou a existência de átomos de outra natureza na monocamada (ver Fig. 1.8). Os defeitos em sistemas atomísticos resultam em significativas mudanças na estrutura eletrônica. Devido à presença de defeitos, surgem estados localizados, que, à medida que o sistema cresce (chegando à ordem de grandezas de centenas ou até milhares de nanômetros), podem levá-lo a um efeito de localização dos estados; conseqüentemente, o transporte de carga no material é suprimido [34, 35, 36]. Além disso, alterações nas configurações de carga podem levar a alterações significativas no transporte eletrônico; assim, a presença de interação entre elétrons não pode ser desprezada para uma descrição mais realista.

1.2.1 Nanoestruturas baseadas em grafeno

Como mencionado anteriormente, devido à atual busca pela confecção de nanodispositivos, uma alternativa explorada é a utilização de fenômenos que ocorrem na escala atômica, como o comportamento tipo QD, para a construção de nanotransistores [5]. Além disso, devido à natureza molecular e às suas propriedades "maravilhosas", o grafeno e seus alotrópicos, como nanotubos de carbono e

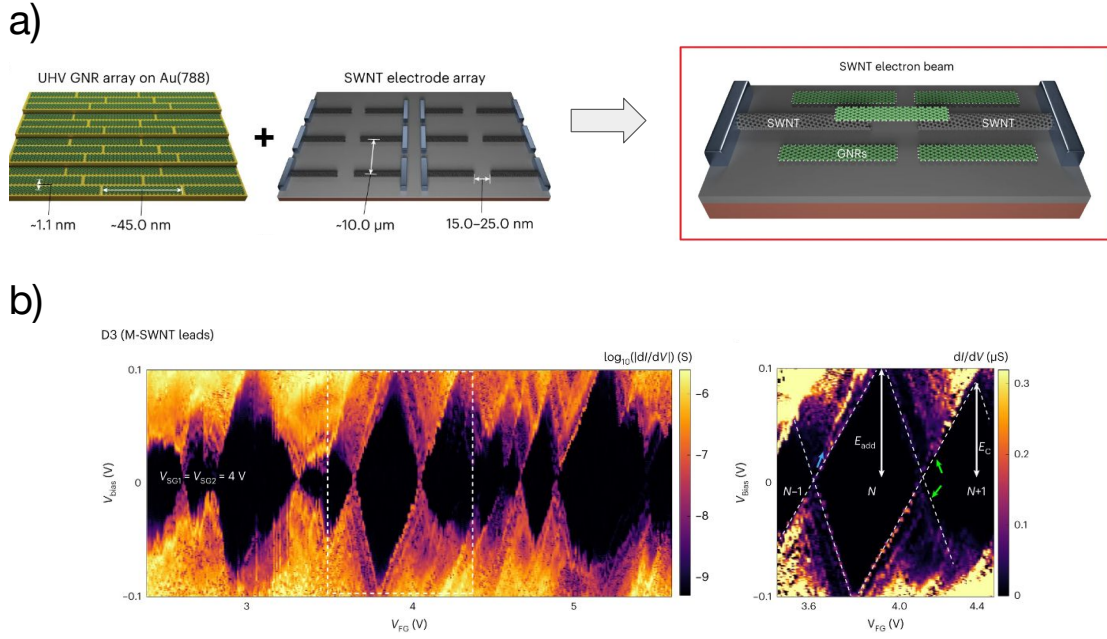


Figura 1.9: Arranjo experimental de um Transistor de Elétron Único, construído com estruturas baseadas em grafeno: a) Eletrodos SWNT para contato *bottom-up* com as GNRs. Esquema do conjunto GNR sintetizado em ultra-alto vácuo (UHV, do inglês) paralelo aos terraços de Au(788) (canto superior esquerdo). Esquema do conjunto de eletrodos SWNT paralelo em um substrato de SiO_2 (canto superior esquerdo). Esquema de um transistor baseado em uma única GNR com SWNTs como eletrodos (canto superior direito). Para maior clareza, apenas os GNRs mais próximos do nanogap são mostrados. b) Comportamento de carregamento de um único elétron em um QD (em particular, o QD metálico D3, da Ref. [7]). Escala de cores para a condutância diferencial em função de V_{FG} (equivalente a V_g na nomenclatura utilizada nesta dissertação) e V_{Bias} , com os potenciais de *gates* laterais $V_{SG1} = V_{SG2} = 4\text{ V}$ fixos, mostrando comportamento de elétron único. Figura inferior direita, sendo um zoom ilustrado pela caixa branca, na figura inferior esquerda. Imagens retirada da Ref. [7].

nanofitas de grafeno, surgem como alternativas para materiais base na construção de nanodispositivos [7, 28]. Entretanto, uma questão difícil de superar é a caracterização do acoplamento entre o dispositivo baseado em grafeno e os contatos, pois, em dispositivos que geralmente utilizam metal na fabricação, surge uma barreira de potencial resultante, chamada de barreira Schottky [37]. Neste sentido, uma série de artigos foram publicados propondo a confecção de sistemas constituídos exclusivamente por carbono, como, por exemplo, QDs, e utilizando esses para a construção de nanodispositivos [7, 28, 29, 30, 31, 38]. Seguindo a ideia proposta por Santos *et al.* [39], recentemente Jian Zhang *et al.* [7] propuseram construir uma arquitetura de dispositivo multi-*gate* com nanotubos de carbono de parede

única (SWNTs) como eletrodos acoplados fracamente através de uma nanofita de grafeno (NR) [7], como ilustrado na Fig.1.9(a). Podemos observar na Fig.1.9(b), a configuração SWNT-NR-SWNT retém o comportamento de um QD, com a assinatura do bloqueio de Coulomb e os diamantes de Coulomb correspondentes. Enquanto, para baixos valores de V_{bias} , é possível controlar o transporte de carga, chegando ao regime de transmissão de um único elétron. Assim, Jian Zhang *et al.* tiveram sucesso ao confeccionar um transistor de elétron único baseado em grafeno.

1.3 Estado da arte em transporte eletrônico

Para compreender o funcionamento de transistores e outros dispositivos eletrônicos, é fundamental introduzir uma teoria de transporte eletrônico. Essa teoria descreve como os elétrons se movem através de materiais condutores e semicondutores, influenciando o desempenho e as características dos dispositivos [40].

A teoria de transporte eletrônico abrange diversos conceitos, incluindo a condução elétrica em metais, semicondutores e isolantes, bem como fenômenos como difusão, dispersão e efeitos quânticos. Ela também é crucial para entender como os dispositivos eletrônicos, como transistores, diodos e circuitos integrados, funcionam e como podem ser projetados e otimizados para diversas aplicações [41].

De maneira simplificada, mas sem perda de generalidade, uma amostra de determinado material em contato com dois (ou mais) reservatórios de elétrons, e caracterizada por uma corrente de carga (elétrons, buracos ou spin) indo de um reservatório para o outro é uma descrição esquemática de um sistema de transporte [40]. No caso de cargas, a corrente surge devido a diferenças entre os reservatórios e depende das propriedades internas, bem como das propriedades do material e da maneira como as diferentes partes são acopladas entre si. Assim, o domínio e o controle com certa precisão das propriedades de transporte podem resultar na confecção de diversos tipos de dispositivos, como diodos, transistores e conversores de energia, que estão no centro das aplicações industriais [41].

Proposta por Fourier em 1822 [42] no intuito de descrever o transporte de calor entre dois reservatórios térmicos com diferentes temperaturas através de um material, a chamada lei de Fourier descreve a corrente térmica e é dada por [42]:

$$I = -\kappa_c \frac{\Delta T}{L}, \quad (1.1)$$

onde L é o tamanho do material na direção da corrente, ΔT é a diferença de temperatura entre os reservatórios e κ_c é a condutividade térmica. De maneira geral, a condutividade é uma função da temperatura T e do tamanho do sistema L , tornando a corrente não linear na temperatura [43]. Analogamente, a corrente elétrica é qualquer movimento de portadores de carga elétrica em um fio, onde os portadores de carga são elétrons, e é uma medida da quantidade de carga que passa por qualquer ponto do fio por unidade de tempo [22]. Considerando apenas a dependência com o tamanho, a corrente pode ser escrita como:

$$I \propto \frac{1}{L^\alpha}. \quad (1.2)$$

Para $\alpha = 0$, a corrente não depende do tamanho, entrando no que chamamos de regime balístico. Para $\alpha = 1$, resgatamos a lei de Fourier com a corrente decaindo $\propto 1/L$, conhecido como limite difusivo. Com $0 < \alpha < 1$, o regime é considerado sub-difusivo e quando $\alpha > 1$, o regime é super-difusivo. Por fim, no limite extremo $\alpha \rightarrow \infty$, temos que $I \rightarrow 0$ e o sistema se torna isolante, podendo ser expresso como $I \propto \exp(-L/L_0)$, com L_0 sendo o comprimento de localização de Anderson.

Em particular, para o sistema de transporte de carga eletrônica, de maneira análoga ao térmico, temos também escalas características. Porém, aqui, diferente do caso de Fourier, dependemos não apenas dos parâmetros macroscópicos, mas também dos parâmetros microscópicos do sistema. Devido a isso, esse fenômeno de transporte é comumente chamado de transporte quântico, ou, em outros casos, transporte eletrônico em sistemas mesoscópicos. A princípio, existe uma sutil diferença entre ambos; enquanto sistemas mesoscópicos estão tipicamente associados à descrição quase-clássica de sistemas contínuos ou com densidade espectral suave, ou quando alguns observáveis do sistema assumem características universais, como as flutuações universais da condutância [22]. Entretanto, recentemente, com a nanoeletrônica, tornou-se cada vez mais necessária uma descrição em nível atômico. Assim, tipicamente, quando existe uma descrição microscópica do sistema, esta é chamada de transporte em nanoescala, ou apenas transporte quântico [25]. É importante mencionar que existem casos onde esses conceitos se misturam. Portanto, a melhor maneira de descrever esses sistemas de uma maneira geral é olhando para suas escalas características. Tipicamente, elas são dadas por

$$\lambda_B < l_p < l_\phi < l_e. \quad (1.3)$$

onde: l_p, τ_p - comprimento de espalhamento elástico (caminho livre médio) e o

tempo associado, l_e, τ_e - comprimento de espalhamento inelástico (energia de relaxação) e tempo, l_ϕ, τ_ϕ - comprimento de coerência de fase quântica e tempo, λ_B - comprimento de onda de de Broglie (que depende da energia cinética, para elétrons no metal é caracterizada em relação à superfície de Fermi).

Em metais e semicondutores à temperatura ambiente, esses comprimentos são tipicamente muito pequenos, o que permite uma descrição via abordagem semiclássica. Por outro lado, no grafeno e em materiais baseados em grafeno, como nanotubos de carbono e nanofitas de grafeno, mesmo à temperatura ambiente, os fenômenos de transporte ainda podem assumir características quânticas, exigindo, assim, uma descrição microscópica.

Assim, o comprimento de onda de de Broglie λ_B e o comprimento de decoerência de fase quântica l_ϕ são a mais importante escala para a classificação dos fenômenos de transporte. Portanto, quando

$$\lambda_B \ll L \ll l_\phi, \quad (1.4)$$

temos que o movimento do elétron no sistema preserva a coerência de fase quântica e, apesar de não poder ser descrito por equações clássicas (devido à natureza quasiclássica desse sistema), pode então ser aproximado por trajetórias clássicas com correção vindas de flutuações quânticas, sendo estes tipicamente chamados de sistemas mesoscópicos. Por outro lado, quando temos

$$L \approx \lambda_B, \quad (1.5)$$

ou seja, a escala do sistema é comparável com o comprimento de onda do elétron e, portanto, os efeitos de quantização como a discretização dos níveis não podem mais ser desprezados. Isso ocorre em sistemas onde estudamos o transporte eletrônico através de junções entre moléculas, pontos quânticos e na presença de interação Coulombiana, por exemplo.

A teoria de Landauer-Büttiker (LB) [44] foi desenvolvida para estudar fenômenos de transporte em sistemas que preservam a coerência, ou seja, na ausência de interações (elétron-elétron e elétron-fônon) e estabeleceu a relação entre observáveis de transporte, como a condutância e corrente elétrica, e as amplitudes de espalhamento das funções de onda. Desta forma, proporciona uma metodologia para descrever alguns dispositivos, como pontos quânticos, nanofios, nanofitas e, posteriormente, é validada até mesmo para sistemas desordenados e quanticamente caóticos, desde que as interações sejam desconsideradas.

Entretanto, na presença de interação (quebra da coerência de fase), esta abordagem não é mais válida. Outra metodologia que surge, mais completa, é o método das funções de Green fora de equilíbrio (NEGF, da sigla em inglês), proposto há quase 50 anos, e é ideal para lidar com sistemas fortemente fora de equilíbrio, ou seja, na presença de eletrodos com diferenças de potenciais finitas entre eles, sendo ainda possível incorporar às NEGFs fenômenos de interação (apesar de não triviais).

Na ausência de interação, os observáveis de transporte são dados pela expressão de Fisher-Lee [45], que é equivalente à teoria de Landauer-Büttiker, mas expressa em termos das NEGFs. Entretanto, na presença de interação elétron-elétron e/ou elétron-fônon, dominam a dinâmica do sistema. Em particular, quando temos a interação elétron-elétron surgem fenômenos como bloqueio Coulombiano [24] e efeito Kondo [10], sendo necessário incorporar os fenômenos de interação diretamente nas NEGFs. Isto foi proposto por Meir-Wingreen [46] em uma generalização da relação de Fisher-Lee para esses sistemas, porém apenas quando os acoplamentos com os eletrodos são proporcionais entre si. Isso significa que para sistemas onde a relação de acoplamento proporcional não é respeitada, não se tinha uma descrição genérica. Uma solução foi proposta recentemente por Amorim-Rocha [47], generalizando a expressão de Meir-Wingreen para o regime de acoplamento não proporcional, e que será abordada nesta dissertação.

Recentemente, sistemas baseados em grafeno têm atraído a atenção da comunidade científica, de perspectiva experimental, para confecção de uma nova classe de nanodispositivos [7, 28, 29, 30, 31]. Utilizando semicondutores e pontos quânticos para gerar nanotransistores construídos de junção de materiais com nanotubos [7], nanofitas de grafeno [28] e o próprio grafeno. Sistemas estes onde as interações e defeitos são partes fundamentais de suas propriedades eletrônicas e de transporte. Assim, com a construção de diferentes métodos abordados nesta dissertação, nos permitir estudar essas classes de materiais na vanguarda da nanociência e descrevê-los, de uma perspectiva teórica, de maneira precisa.

1.4 Estrutura da dissertação

Nesta dissertação, desenvolvemos uma ferramenta computacional para estudar fenômenos de transporte eletrônico em nanodispositivos na presença de interação elétron-elétron e suas consequências, especialmente o surgimento de uma corrente eletrônica decoerente devido à quebra na coerência de fase eletrônica causada

pela interação entre os elétrons. Além de contribuir com novos resultados para a compreensão de um problema ainda em aberto.

No Capítulo 1, apresentamos as abordagens teóricas utilizadas. Começamos com uma breve revisão dos conceitos de matéria condensada e física de muitos corpos. Em seguida, discutimos o método das funções de Green recursivas para um hamiltoniano geral e para sistemas abertos (com quebra de simetria translacional). Introduzimos também uma descrição da interação elétron-elétron em redes cristalinas, particularmente através do Modelo de Anderson para impurezas magnéticas (AIM) combinado com uma representação de pseudo-partícula, que permite a resolução do AIM por meio de uma expansão diagramática. Desenvolvemos uma projeção sobre estados de impurezas moleculares que transforma uma rede cristalina escrita em uma base de orbitais atômicos para uma base de orbitais moleculares, e utilizamos a teoria do campo médio dinâmico (DMFT) para "amarrar" os diferentes modelos por meio de um ciclo autoconsistente. Por fim, apresentamos a teoria de transporte eletrônico na presença de interação elétron-elétron, incluindo a generalização das equações de Fisher-Lee na presença de acoplamento não proporcional entre os eletrodos.

No Capítulo 2, apresentamos resultados para um transistor de elétron único baseado na junção de diferentes materiais de grafeno, utilizando nanotubos de carbono como eletrodos e uma nanofita de grafeno semicondutora como centro espalhador.

No Capítulo 3, também aplicamos nossa ferramenta a uma nanofita de grafeno na presença de defeitos, como vacâncias e impurezas de Anderson, e apresentamos os resultados para esse sistema.

Por fim, no último capítulo, tiramos conclusões sobre os resultados obtidos e discutimos suas implicações gerais.

Capítulo 2

Métodos

Como mencionado anteriormente, para descrever adequadamente nanodispositivos em um formalismo teórico apropriado, é necessário considerar a presença de interações elétron-elétron através da repulsão Coulombiana. Normalmente, isso é negligenciado [39] devido à complexidade envolvida na solução de um problema de muitos corpos. Aqui, concentramo-nos especificamente em duas classes: *i*) dispositivos 1D, nanofitas de grafeno na presença de desordem gerada por vacâncias, e *ii*) dispositivos quase-0D, pontos quânticos abertos gerados pela junção de diferentes estruturas baseadas em grafeno (nanotubos de carbono e nanofitas de grafeno).

Neste capítulo, discutimos a combinação de métodos que nos permitiu descrever esses sistemas de maneira adequada. Utilizamos o modelo tight-binding (MTB) (Sec. 2.2), o método de funções de Green recursivas (Subsec. 2.5.1), o modelo de Anderson para impurezas magnéticas (AIM, da sigla em inglês) (Subsec. 2.6.1), a representação de pseudopartícula (Subsec. 2.6.2), a projeção sobre estados localizados (Sec. 2.7) e, por fim, empregamos a teoria do campo médio dinâmico (DMFT, da sigla em inglês) (Sec. 2.8) para integrar todas as metodologias em um ciclo autoconsistente. Essa abordagem, combinada com a generalização das equações de Fisher-Lee (Subsec. 2.9.3) para o transporte eletrônico interagente no regime de acoplamento não proporcional, permitiu observar a influência da interação elétron-elétron nesses dispositivos.

2.1 Estrutura Cristalina

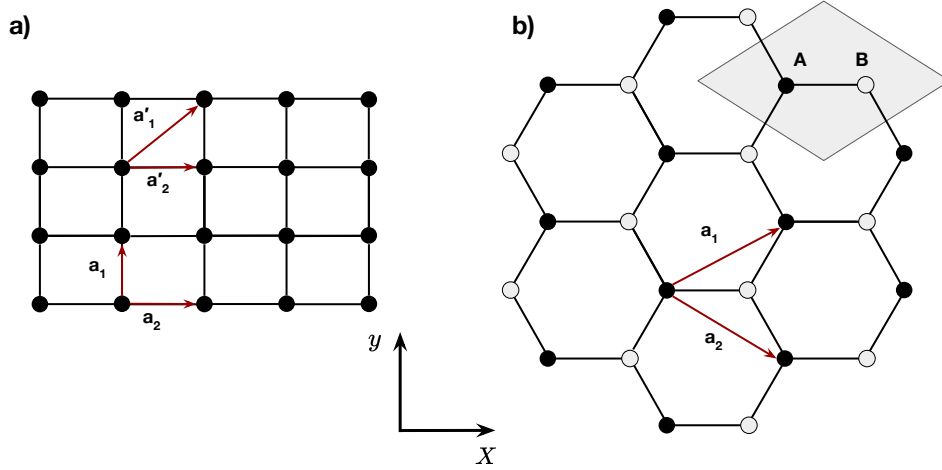


Figura 2.1: **Visão esquemática de estruturas cristalinas:** Pontos (representando átomos ou moléculas) ligados por arestas formando estruturas periódicas, chamadas de redes cristalinas, onde $\mathbf{a}_1$, $\mathbf{a}'_1$, $\mathbf{a}_2$ e $\mathbf{a}'_2$ são vetores primitivos de acordo com a descrição da rede de Bravais [8]. a) Exemplo de uma rede quadrada. b) Rede hexagonal, também conhecida como "favo de mel", com A e B constituindo subredes que formam a célula unitária.

Muitos sólidos são constituídos de arranjos regulares de átomos formando uma estrutura cristalina perfeita. Portanto, elétrons movendo sobre esses sólidos são sujeitos a um potencial periódico. Uma maneira convencional de descrever redes cristalinas é através da chamada rede de Bravais [8], representando um conjunto de pontos correspondentes às posições discretas no espaço, dados por

$$\{\mathbf{R} | \mathbf{R} = n_1 \mathbf{a}_1 + n_2 \mathbf{a}_2 + n_3 \mathbf{a}_3; \quad n_i \in \mathbb{N}\}. \quad (2.1)$$

Aqui $\mathbf{a}_i$ (com $i = 1, 2, 3$) é o chamado vetor primitivo em três dimensões. Temos que a rede de Bravais é periódica e invariante por translação discretas

$$\mathbf{R} \rightarrow \mathbf{R} + \mathbf{T}, \quad (2.2)$$

com

$$\mathbf{T} = L\mathbf{a}_1 + M\mathbf{a}_2 + N\mathbf{a}_3, \quad (2.3)$$

sendo L, M, N valores inteiros.

Constrói-se então uma estrutura cristalina, fixando um conjunto de posições em relação a cada um dos pontos da rede de Bravais, representando átomos, íons ou moléculas. Na Figura 2.1, podem ser vistos dois exemplos de estruturas cristalinas que podemos construir usando redes de Bravais. Na Figura 2.1(a), temos uma rede quadrada onde dois exemplos de vetores primitivos são representados,

$$\mathbf{a}_1 = (0, a_1), \quad \mathbf{a}_2 = (a_2, 0) \quad \text{ou} \quad \mathbf{a}'_1 = (\sqrt{a_1^2 + a_2^2}, y), \quad \mathbf{a}'_2 = (a_1, 0), \quad (2.4)$$

resultando na mesma estrutura. Já na Figura 2.1(b), vemos um exemplo de uma rede triangular ou hexagonal, onde definimos a chamada célula unitária com dois sítios (A e B), que, de modo geral, podem ser diferentes. Escolhemos os vetores primitivos

$$\mathbf{a}_1 = a \left(\frac{\sqrt{3}}{2}, \frac{1}{2} \right), \quad \mathbf{a}_2 = a \left(\frac{\sqrt{3}}{2}, -\frac{1}{2} \right), \quad (2.5)$$

Com a sendo a distância entre os pontos A e B.¹ A estrutura completa forma a rede cristalina de favo de mel, presente em vários materiais de interesse físico, em particular, o grafeno [48].

Dada a simetria de translação, podemos definir quantidades físicas, $f(\mathbf{X})$, que possuam a mesma simetria da sua respectiva rede de Bravais, ou seja,

$$f(\mathbf{X}) = f(\mathbf{X} + \mathbf{R}). \quad (2.6)$$

Podemos expressar a invariância sobre translações de maneira mais explícita através da transformada de Fourier

$$\begin{aligned} f(\mathbf{X}) &= f(\mathbf{X} + \mathbf{R}), \\ \sum_{\mathbf{q}} f(\mathbf{q}) e^{i\mathbf{q} \cdot \mathbf{X}} &= \sum_{\mathbf{q}} f(\mathbf{q}) e^{i\mathbf{q} \cdot (\mathbf{X} + \mathbf{R})}, \\ e^{i\mathbf{q} \cdot \mathbf{X}} &= e^{i\mathbf{q} \cdot (\mathbf{X} + \mathbf{R})}. \quad \forall \mathbf{q} \end{aligned}$$

Como consequência, temos

$$\mathbf{q} \cdot \mathbf{R} = 2\pi n, \quad n \in \mathbb{N}. \quad (2.7)$$

¹Neste caso, a rede possui 2 átomos na base, uma vez que não é possível encontrar uma representação mínima contendo apenas 1 átomo.

Podemos escrever $\mathbf{q}$ em uma base arbitrária de vetores

$$\mathbf{q} = \sum_j l_j \mathbf{b}_j \quad l_j \in \mathbb{N}. \quad (2.8)$$

Usando as relações (2.1), (2.7) e (2.8) temos,

$$\mathbf{a}_i \cdot \mathbf{b}_j = 2\pi \delta_{i,j}. \quad (2.9)$$

Isto permite escrever quantidades físicas invariantes sob a simetria da rede de Bravais, na representação de momentos $\mathbf{q}$, em termos dos vetores primitivos $\mathbf{b}$, definindo assim a base

$$\{\mathbf{Q} | \mathbf{Q} = n_1 \mathbf{b}_1 + n_2 \mathbf{b}_2 + n_3 \mathbf{b}_3; \quad n_i \in \mathbb{N}\} \quad (2.10)$$

chamada rede recíproca.

Finalmente, podemos considerar um elétron livre na presença de uma rede cristalina. O Hamiltoniano de uma partícula livre é dado por

$$H_0 = \frac{\mathbf{p}^2}{2m'}, \quad (2.11)$$

e o operador de translação arbitrário

$$T(\mathbf{a}) = e^{\frac{\mathbf{p}}{\hbar} \cdot \mathbf{a}}. \quad (2.12)$$

Como $H_0 |\Psi_0\rangle = E_0 |\Psi_0\rangle$ e $[T(\mathbf{a}), H_0] = 0$, temos que T e H_0 possuem os mesmos autovalores, sendo eles $e^{i\mathbf{q} \cdot \mathbf{a}}$, dados em termos de q , vetores de onda arbitrários, que são constantes de movimento. Escrevemos os efeitos da simetria cristalina adicionando um potencial periódico que depende unicamente da posição, $V(\mathbf{r}) = V(\mathbf{r} + \mathbf{R})$, ao Hamiltoniano

$$H = \frac{\mathbf{p}^2}{2m} + V(\mathbf{r}). \quad (2.13)$$

A equação de Schrödinger então é dada por

$$\left[-\frac{\hbar^2}{2m} \nabla^2 + V(\mathbf{r}) \right] \Psi(\mathbf{r}) = E \Psi(\mathbf{r}). \quad (2.14)$$

Na presença de um potencial periódico, H é invariante sob translações e, portanto,

H e $T(\mathbf{R})$ tem uma base comum de autofunções e satisfazem

$$\Psi(\mathbf{r} + \mathbf{R}) = T(\mathbf{R})\Psi(\mathbf{r}) = e^{i\mathbf{q} \cdot \mathbf{R}}\Psi(\mathbf{r}). \quad (2.15)$$

Expressando $\mathbf{q}$ explicitamente em termos dos vetores primitivos da rede recíproca, e tomando $\mathbf{q} = \mathbf{Q} + \mathbf{k}$, analogamente à relação 2.2. Como consequência

$$T(\mathbf{R})\Psi(\mathbf{r}) = e^{i\mathbf{k} \cdot \mathbf{R}}\Psi(\mathbf{r}), \quad (2.16)$$

que é o chamado teorema de Floquet. Expressando a função de onda e o potencial no espaço dos momentos, temos

$$\begin{aligned} \Psi(\mathbf{r}) &= \sum_{\mathbf{q}} \Psi(\mathbf{q}) e^{i\mathbf{q} \cdot \mathbf{r}} \\ V(\mathbf{r}) &= \sum_{\mathbf{Q}} V(\mathbf{Q}) e^{i\mathbf{Q} \cdot \mathbf{r}}. \end{aligned} \quad (2.17)$$

Devido à simetria de V , sua transformada de Fourier é uma soma apenas sobre vetores da rede recíproca. Desta forma, temos a equação de Schrödinger na base de momentos

$$-\frac{\hbar^2 \mathbf{q}^2}{2m} \Psi(\mathbf{q}) + \sum_{\mathbf{K}} V(\mathbf{K}) \Psi(\mathbf{q} - \mathbf{K}) = E \Psi(\mathbf{q}). \quad (2.18)$$

Aqui $\mathbf{Q} = \mathbf{q} - \mathbf{K}$, com $\mathbf{K}$ um vetor que pertence à rede recíproca. Usando a relação $\mathbf{q} = \mathbf{k} + \mathbf{Q}$, com $\mathbf{k}$ pertencendo à primeira zona de Brillouin,

$$\left(\frac{\hbar^2}{2m} |\mathbf{k} + \mathbf{Q}|^2 - E \right) \Psi(\mathbf{k} + \mathbf{Q}) + \sum_{\mathbf{K}} V(\mathbf{K}) \Psi(\mathbf{k} + \mathbf{Q} - \mathbf{K}) = 0. \quad (2.19)$$

Assim, temos um sistema de equações matriciais. Fixando o valor de $\mathbf{k}$, nota-se que este apenas se acopla com translações discretas. Usando o teorema de Floquet da Eq. 2.16, a expressão se reduz a,

$$\Psi_{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{Q}} \Psi(\mathbf{k} + \mathbf{Q}) e^{i(\mathbf{k} + \mathbf{Q}) \cdot \mathbf{r}} = e^{i\mathbf{k} \cdot \mathbf{r}} \sum_{\mathbf{Q}} \Psi(\mathbf{k} + \mathbf{Q}) e^{i\mathbf{Q} \cdot \mathbf{r}} = e^{i\mathbf{k} \cdot \mathbf{r}} \psi_{\mathbf{k}}(\mathbf{r}). \quad (2.20)$$

De maneira geral, temos que a função de onda para um Hamiltoniano sujeito ao potencial periódico da rede cristalina é

$$\Psi_{\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{k} \cdot \mathbf{r}} \psi_{\mathbf{k}}(\mathbf{r}), \quad (2.21)$$

com um termo oscilante, análogo a uma partícula livre, e um termo periódico na rede que depende unicamente da posição da partícula em relação aos pontos da rede de Bravais. Definimos, também k como o momento cristalino. Este é o chamado teorema de Bloch [8].

2.2 Modelo Tight-binding

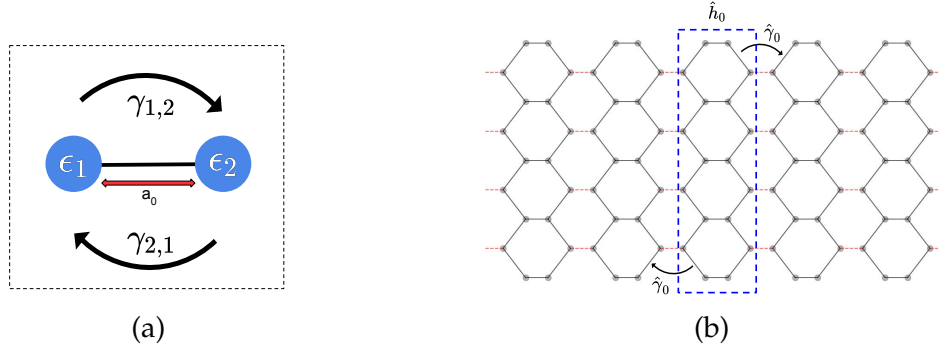


Figura 2.2: **Visão esquemática do Modelo Tight-Binding:** a) Os círculos azuis representam os sítios da nossa rede (aqui para uma rede 2×2), onde ϵ_1 e ϵ_2 representam as energias dos sítios 1 e 2, respectivamente, e $\gamma_{1,2}$ e $\gamma_{2,1}$ representam o acoplamento entre os sítios 1 e 2 e entre 2 e 1. b) Descrição de uma nanofita com simetria de rede hexagonal, onde $\hat{h}_0$ representa a estrutura de repetição mínima, também chamada de célula unitária, e $\hat{\gamma}_0$ representa o acoplamento entre células unitárias.

Considerando uma rede cristalina com átomos separados por uma distância a_0 e acoplados entre si, uma descrição simples que podemos fazer é o chamado modelo tight-binding (MTB). Neste modelo, escrevemos a função de onda do sistema como uma combinação linear de funções de ondas de orbitais atômicos centrados em cada um dos sítios. Desta forma, o Hamiltoniano de um sistema qualquer na base dos orbitais atômicos localizados é dado por um termo dos sítios isolados e um termo de acoplamento,

$$\hat{H} = \sum_i \hat{h}_i + \sum_{i,j} \hat{\gamma}_{i,j} = \sum_i \epsilon_i |i\rangle \langle i| + \sum_{i \neq j} \gamma_{i,j} |i\rangle \langle j|. \quad (2.22)$$

Como ilustrado pela Fig. 2.2(a), o termo ϵ_i é a energia associada ao orbital atômico no sítio i , enquanto $\gamma_{i,j}$ é o termo de acoplamento entre os orbitais i e j , de modo que, ambos são escalares.² Suponha que a forma de interação - que pode ser

²Em geral, $\langle i|$ pode estar associado a vários índices diferentes, como $\langle i| = \langle i, \alpha, \sigma|$, indicando

dados por um potencial efetivo- e é conhecida, o termo dos sítios isolados e o acoplamento podem ser resolvido em termos [8],

$$\epsilon_i = \int d\mathbf{r} \psi_i^\dagger(\mathbf{r}) \hat{h}_i \psi_i(\mathbf{r}) \quad \text{e} \quad \gamma_{i,j} = \int d\mathbf{r} \psi_i^\dagger(\mathbf{r}) \hat{\gamma}_{\mathbf{R}}(\mathbf{r}) \psi_j(\mathbf{r} + \mathbf{R}), \quad (2.23)$$

com ψ_i a autofunção de $\hat{h}_i$.

Na presença de simetria translacional, a unidade de repetição mínima da rede é a célula unitária, como ilustrado na Fig. 3.1(b) para o caso particular de uma rede com simetria hexagonal. Isso torna conveniente escrever o Hamiltoniano em termos da célula unitária. Então, nessa base, que pode ter graus de liberdade internos, temos:

$$\hat{H} = \sum_m \hat{h}_m |m\rangle \langle m| + \sum_{m \neq n} \hat{\gamma}_{m,n} |m\rangle \langle n|, \quad (2.24)$$

com $\hat{h}_m$ o Hamiltoniano “intramolecular”, descrito através da unidade de repetição mínima dada pela Eq. 2.22, e $\hat{\gamma}_{m,n}$ o acoplamento entre as células unitárias, como ilustrado na Fig. 2.2(b). Com isso, é possível expressar o problema de maneira análoga a uma cadeia linear com N orbitais atômicos, incorporando novos graus de liberdade internos para cada orbital. Na forma matricial, isso resulta em uma matriz em blocos que, de maneira geral e independente da simetria local, possui a seguinte estrutura: ³

$$\hat{H} = \begin{pmatrix} \hat{h}_1 & \hat{\gamma}_{1,2} & \dots & 0 & \hat{\gamma}_{1,N} \\ \hat{\gamma}_{2,1} & \hat{h}_2 & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \hat{\gamma}_{N-1,N} \\ \hat{\gamma}_{N,1} & 0 & \dots & \hat{\gamma}_{N,N-1} & \hat{h}_N \end{pmatrix}, \quad (2.25)$$

com $\hat{\gamma}_{1,N}$ e $\hat{\gamma}_{N,1}$ as condições de periodicidade do sistema. Para obter a função de onda, tomamos uma translação arbitrária R sobre $|i\rangle$,

$$T(\mathbf{R}) |i, \mathbf{k}\rangle = e^{i\mathbf{k} \cdot \mathbf{R}} |i, \mathbf{k}\rangle, \quad (2.26)$$

com $\mathbf{k}$ pertencendo à primeira zona Brillouin. A base em termos do momento

que para cada sítio i existem diferentes orbitais atômicos α e projeções de spin σ .

³No caso de um sistema infinito, temos $N \rightarrow \infty$, e para o caso periódico, $\hat{h}_i = \hat{h}_j$ e $\hat{\gamma}_{i,j} = \hat{\gamma}_{i+1,j+1}$ para todo i e j .

cristalino, então é dada por:

$$|i, \mathbf{k}\rangle = \frac{1}{\sqrt{N}} \sum_{\mathbf{R}_i} e^{i\mathbf{k} \cdot \mathbf{R}_i} |i, \mathbf{R}_i\rangle. \quad (2.27)$$

Aqui, N é o número total de pontos na rede de Bravais. Projetando o Hamiltoniano em termos dos momentos,

$$\hat{H} |i, \mathbf{k}\rangle = \hat{E}_i |i, \mathbf{k}\rangle - \frac{1}{\sqrt{N}} \sum_{\langle \mathbf{R}_i \mathbf{R}_j \rangle} \hat{\gamma}_{i,j} e^{i\mathbf{k} \cdot \mathbf{R}_i} |i, \mathbf{R}_j\rangle. \quad (2.28)$$

e dado a relação $\hat{H} |i, \mathbf{k}\rangle = \epsilon_i(\mathbf{k}) |i, \mathbf{k}\rangle$ podemos obter os autovalores.

De maneira simplificada, considerando apenas 1 orbital por celular unitária, ou seja, com $\hat{h} = \epsilon_i$ e o acoplamento constante $\hat{\gamma}_{m,n} = \gamma$, podemos obter a relação de dispersão, de maneira analítica,

$$\epsilon_i(\mathbf{k}) = \epsilon_i - \gamma \sum_{\mathbf{R}_i} e^{i\mathbf{k} \cdot \mathbf{R}_i}. \quad (2.29)$$

E, por definição, a densidade de estados é:

$$D(E) = \sum_{i, \mathbf{k}} \delta(E - \epsilon_i(\mathbf{k})). \quad (2.30)$$

2.3 Grafeno

Um caso de extrema importância neste trabalho é o grafeno [48], que consiste em uma estrutura com simetria hexagonal, onde cada ponto da rede corresponde a um átomo de carbono que se conecta com outros três. A rede tem uma estrutura hexagonal, correspondendo a uma rede bipartida com a estrutura de duas sub-redes triangulares, A e B , como ilustrado na Fig. 2.3, onde um átomo em uma sub-rede se conecta com 3 átomos da outra. Definindo a célula unitária com dois átomos, cada um pertencente à sub-rede A e B , os vetores primitivos da rede são dados por [48].

$$\mathbf{a}_1 = a \left(\frac{\sqrt{3}}{2}, \frac{1}{2} \right), \quad \mathbf{a}_2 = a \left(\frac{\sqrt{3}}{2}, -\frac{1}{2} \right), \quad (2.31)$$

Com $a = \sqrt{3}a_{cc}$, onde a_{cc} é a distância entre os átomos de carbono, que tipicamente no grafeno é de 1.42 Å. Assim, os vetores primitivos na rede recíproca são dados

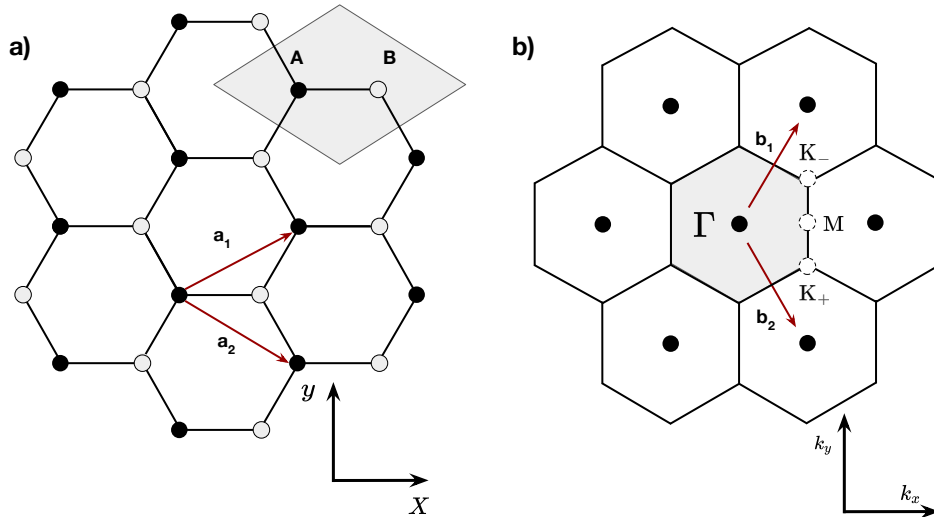


Figura 2.3: **Visão esquemática de redes hexagonais:** a) Uma representação dentro do espaço de posições, com A e B sendo as subredes e constituindo a célula unitária e $\mathbf{a}_1, \mathbf{a}_2$ os vetores primitivos da rede direta (posição). b) Representação dentro do espaço dos momentos, com $\mathbf{b}_1$ e $\mathbf{b}_2$ os vetores primitivos da rede recíproca (representação dos momentos). Γ, K_-, K_+, M , sendo pontos de simetrias no espaço de momentos.

por:

$$\mathbf{b}_1 = b \left(\frac{1}{2}, \frac{\sqrt{3}}{2} \right), \quad \mathbf{b}_2 = b \left(\frac{1}{2}, -\frac{\sqrt{3}}{2} \right), \quad (2.32)$$

Com $b = 4\pi/a\sqrt{3}$. Mostrado na Fig. 2.3(b), temos a primeira zona de Brillouin (região sombreada em cinza). Esta zona de Brillouin, de formato hexagonal, é construída como a célula de Wigner-Seitz [48] da rede recíproca. Dos seus pontos dos vértices, dois deles são desiguais (os outros podem ser escritos como um destes dois mais um vetor de rede recíproca). Esses dois pontos especiais são denotados por K_+ e K_- . Outro ponto de alta simetria é aquele marcado com M na Fig. 2.3(b). Escrevemos eles como:

$$\mathbf{K}_+ = \frac{4\pi}{3a} \left(\frac{\sqrt{3}}{2}, -\frac{1}{2} \right), \quad \mathbf{K}_- = \frac{4\pi}{3a} \left(\frac{\sqrt{3}}{2}, \frac{1}{2} \right), \quad \mathbf{M} = \frac{2\pi}{\sqrt{3}a} (1, 0). \quad (2.33)$$

Devido a essa simetria, as ligações entre os carbonos geram orbitais híbridos, sendo eles os orbitais σ , que fazem a ligação carbono-carbono e são fortemente acoplados, e os orbitais p_z , que representam os orbitais próximos ao nível de Fermi. Assim, é conveniente modelar esses sistemas usando o modelo tight-binding (TB) para descrever apenas o orbital livre, já que, no caso de fenômenos de transporte, estes

estão tipicamente confinados a esta região de energia. Na nossa descrição, temos efetivamente um orbital por sítio ($\epsilon_{i,j} = \epsilon_0$) e com o mesmo acoplamento entre todos os sítios ($\gamma_{i,j} = \gamma$) [49], consistindo apenas acoplamento entre primeiros vizinhos.

Devido à ortogonalidade das funções de onda e à periodicidade do sistema [48], a equação de Schrödinger pode ser tratada como um problema de diagonalização de uma matriz 2×2 ,

$$\begin{pmatrix} \hat{H}_{A,A}(\mathbf{k}) & \hat{H}_{A,B}(\mathbf{k}) \\ \hat{H}_{B,A}(\mathbf{k}) & \hat{H}_{B,B}(\mathbf{k}) \end{pmatrix} \begin{pmatrix} \hat{c}_A \\ \hat{c}_B \end{pmatrix} = E(\mathbf{k}) \begin{pmatrix} \hat{c}_A \\ \hat{c}_B \end{pmatrix}. \quad (2.34)$$

O Hamiltoniano na base de momentos assume a forma:

$$\hat{H}(\mathbf{k}) = \begin{pmatrix} 0 & -\gamma(1 + e^{-i\mathbf{k} \cdot (\mathbf{a}_1 + \mathbf{a}_2)}) \\ -\gamma(1 + e^{i\mathbf{k} \cdot (\mathbf{a}_1 + \mathbf{a}_2)}) & 0 \end{pmatrix}, \quad (2.35)$$

de modo que obtemos a relação de dispersão diagonalizando $\hat{H}(\mathbf{k})$ (com $\mathbf{k} = (k_x, k_y)$),

$$E_{\pm}(k_x, k_y) = \pm\gamma \sqrt{1 + 4 \cos \frac{\sqrt{3}k_x a}{2} \cos \frac{k_y a}{2} + 4 \cos^2 \frac{k_y a}{2}}. \quad (2.36)$$

Importante notar que para os valores de $\mathbf{k}$ em que $E_{\pm} = 0$. Expandindo $E_{\pm}$ para $\mathbf{k}$ na vizinhança de $\mathbf{K}_+$ (ou $\mathbf{K}_-$), temos $\mathbf{k} = \mathbf{K}_+ + \delta\mathbf{k}$ ($\mathbf{k} = \mathbf{K}_- + \delta\mathbf{k}$). Por fim, temos a relação de dispersão [48],

$$E_{\pm}(\delta\mathbf{k}) = \pm\hbar v_F |\delta\mathbf{k}|, \quad (2.37)$$

aqui,

$$v_F = \frac{\sqrt{3}\gamma a}{2\hbar} \quad (2.38)$$

é a velocidade de grupo eletrônica. Desta forma, temos a formação de dois cones que se tocam próximo de $E_{\pm} = 0$, chamados pontos de Dirac, e a dispersão do grafeno se assemelha à de um férmion de Dirac sem massa [48]. Além disso, a partir da relação de dispersão, obtemos a densidade de estados dada pela Eq. 2.30, e apresentado na Fig. 2.4

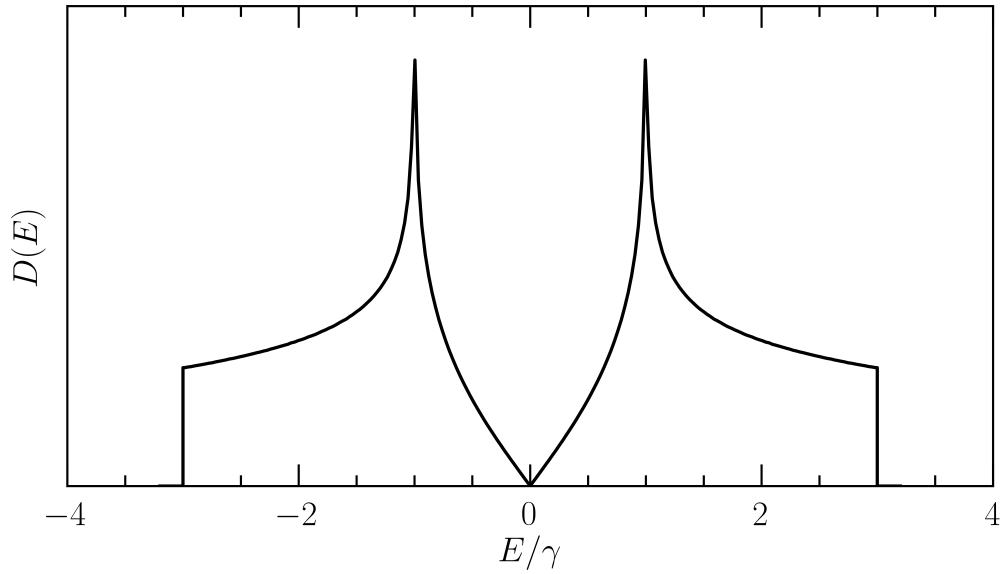


Figura 2.4: **Densidade de estados:** densidade de estados para uma folha de grafeno em função da energia reescalada pelo constante de acoplamento.

2.3.1 Nanotubos de Carbono e Nanofitas de Grafeno

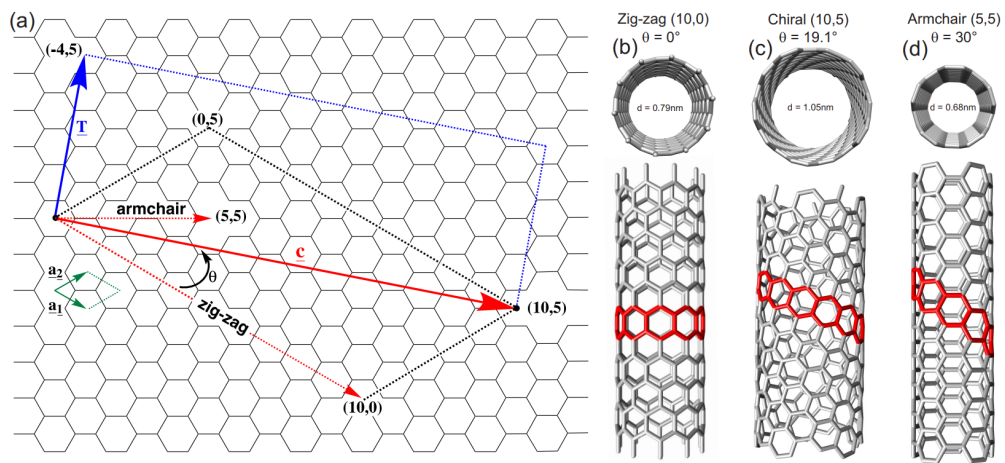


Figura 2.5: **Representação de vetores quirais que levam a vários tipos de nanotubos de carbono de parede única (SWNTs):** (a) O vetor quiral $\mathbf{C} = 10\mathbf{a}_1 + 5\mathbf{a}_2$ (seta vermelha), conecta dois pontos em uma folha de grafeno e define a circunferência e a quiralidade de um nanotubo (10,5). (b, c, d) Exemplos de nanotubos zig-zag, quiral e armchair, com seus ângulos θ e diâmetros. Imagem retirada da Ref. [9].

O grafeno é a estrutura predecessora de um conjunto de alótropos do carbono [50]. O mais conhecido é o abundante grafite. No entanto, a partir de cortes apropriados na folha de grafeno, é possível obter outras estruturas nanoscópicas

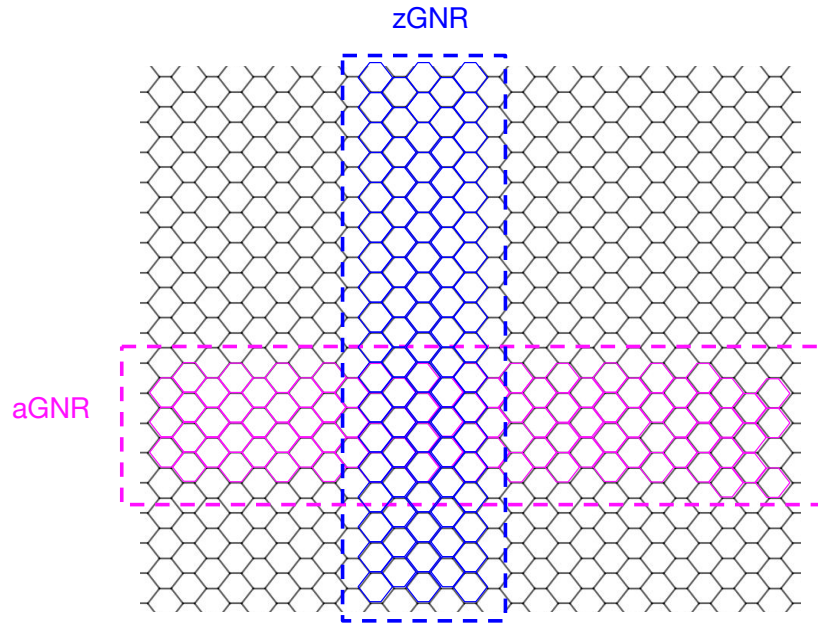


Figura 2.6: **Representação de uma nanofita de grafeno:** Nanofitas de grafeno são obtidas a partir de uma folha de grafeno (representada em cinza), onde cortes são feitos em uma direção específica. A partir desses cortes, uma faixa estreita de grafeno é extraída, formando assim a nanofita de grafeno.

de dimensões inferiores. O primeiro caso, obtido experimentalmente antes mesmo do grafeno, são os nanotubos de carbono (CNT, da sigla em inglês), descobertos por Iijima [17] em 1991.

É possível gerar uma estrutura de nanotubos de carbono de parede única (SWNT, da sigla em inglês) removendo uma tira de grafeno e enrolando-a [51], como ilustrado na Fig. 2.5. Sua estrutura pode ser especificada pelo vetor quiral $\mathbf{C}$ [48], que conecta duas células unitárias distintas (veja Fig. 2.5) em uma folha de grafeno. Portanto, o vetor quiral pode ser especificado por dois números inteiros (n e m), $\mathbf{C} = n\mathbf{a}_1 + m\mathbf{a}_2$, e representa a posição relativa do par de átomos na rede de grafeno que forma um tubo quando enrolado. O par (n, m) rotula exclusivamente SWNTs.

Outro caso de interesse são as nanofitas de grafeno, que são obtidas apenas cortando uma folha de grafeno [48], como ilustrado na Fig. 2.6. Da mesma forma que os CNTs, podem ser obtidas diferentes estruturas dependendo da maneira como são cortadas. Em particular, temos nanofitas com borda armchair e zig-zag, que possuem as estruturas ilustradas nas Fig. 2.7 e Fig. 2.8. Elas são caracterizadas pelo número inteiro N_a (N_z), que representa a família armchair (zig-zag) em termos de suas respectivas células unitárias e possuem diferentes propriedades eletrônicas

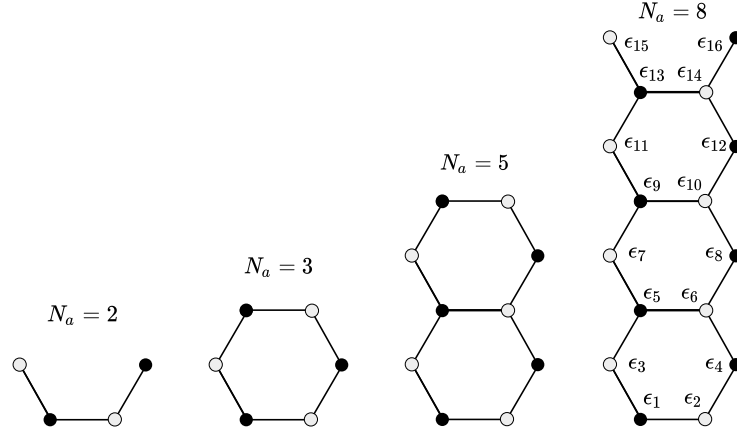


Figura 2.7: **Célula unitária da nanofita de grafeno armchair:** Ilustração da sequência de construção da célula unitária para diferentes famílias armchair. ϵ_i representa a energia do sítio, e são apresentadas as células unitárias para $N_a = 2, 3, 5$ e 8 .

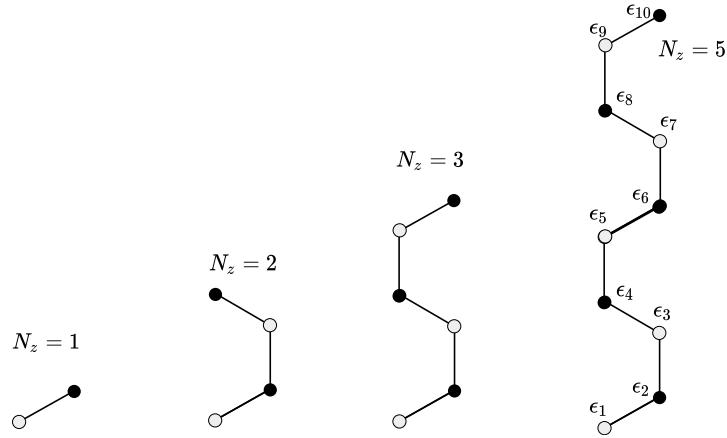


Figura 2.8: **Célula unitária da nanofita de grafeno zig-zag:** Ilustração da sequência de construção da célula unitária para diferentes famílias zig-zag. ϵ_i representa a energia do sítio, e são apresentadas as células unitárias para $N_z = 1, 2, 3$ e 5 .

[48].

O Hamiltoniano de uma célula unitária para uma nanofita zig-zag na descrição do MTB é dado, para o caso com $N_z = 2$, por:

$$\hat{h}_{N_z=2} = \begin{pmatrix} \epsilon_1 & \gamma & 0 & 0 \\ \gamma & \epsilon_2 & \gamma & 0 \\ 0 & \gamma & \epsilon_3 & \gamma \\ 0 & 0 & \gamma & \epsilon_4 \end{pmatrix},$$

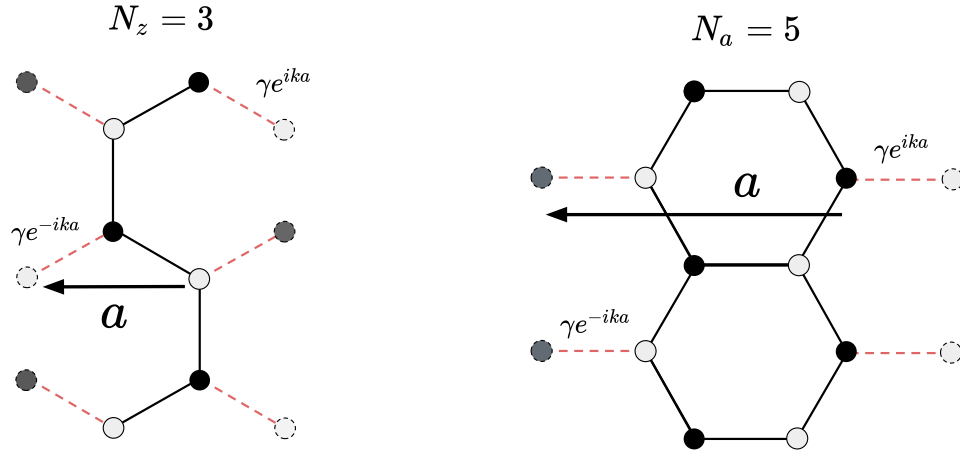


Figura 2.9: **Nanofita de Grafeno Infinita:** Representação da célula unitária (unidade de repetição mínima) das nanofitas de grafeno para os casos zig-zag, com $N_z = 3$, e armchair, $N_a = 5$. Aqui, γe^{-ika} é a constante de acoplamento com a célula anterior, e γe^{ika} para a próxima célula.

e para uma nanofita armchair, com $N_a = 5$, por:

$$\hat{h}_{N_a=5} = \begin{pmatrix} \epsilon_1 & \gamma & \gamma & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma & \epsilon_2 & 0 & \gamma & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma & 0 & \epsilon_3 & 0 & \gamma & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma & 0 & \epsilon_4 & 0 & \gamma & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma & 0 & \epsilon_5 & \gamma & \gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma & \gamma & \epsilon_6 & 0 & \gamma & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma & 0 & \epsilon_7 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 0 & 0 & \gamma & 0 & \epsilon_8 & 0 & \gamma \\ 0 & 0 & 0 & 0 & 0 & 0 & \gamma & 0 & \epsilon_9 & \gamma \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma & \gamma & \epsilon_{10} \end{pmatrix}, \quad (2.39)$$

O Hamiltoniano completo na base de momentos ao longo da fita, seguindo a representação da Fig. 2.9 para a simetria zig-zag, é dado por:

$$\begin{aligned} \hat{H}_{N_z}(k) = & \sum_{m,i} \epsilon_i |m,i\rangle \langle m,i| + \gamma |m,i\rangle \langle m,i+1| + \gamma |m,i\rangle \langle m,i-1| \\ & + e^{-ika} \gamma \sum_{m,l} |m,l\rangle \langle m-1,l+1| + e^{ika} \gamma \sum_{m,l} |m,l\rangle \langle m+1,l+1| \\ & + e^{-ika} \gamma \sum_{m,j} |m,j\rangle \langle m-1,j-1| + e^{ika} \gamma \sum_{m,j} |m,j\rangle \langle m+1,j-1|. \end{aligned} \quad (2.40)$$

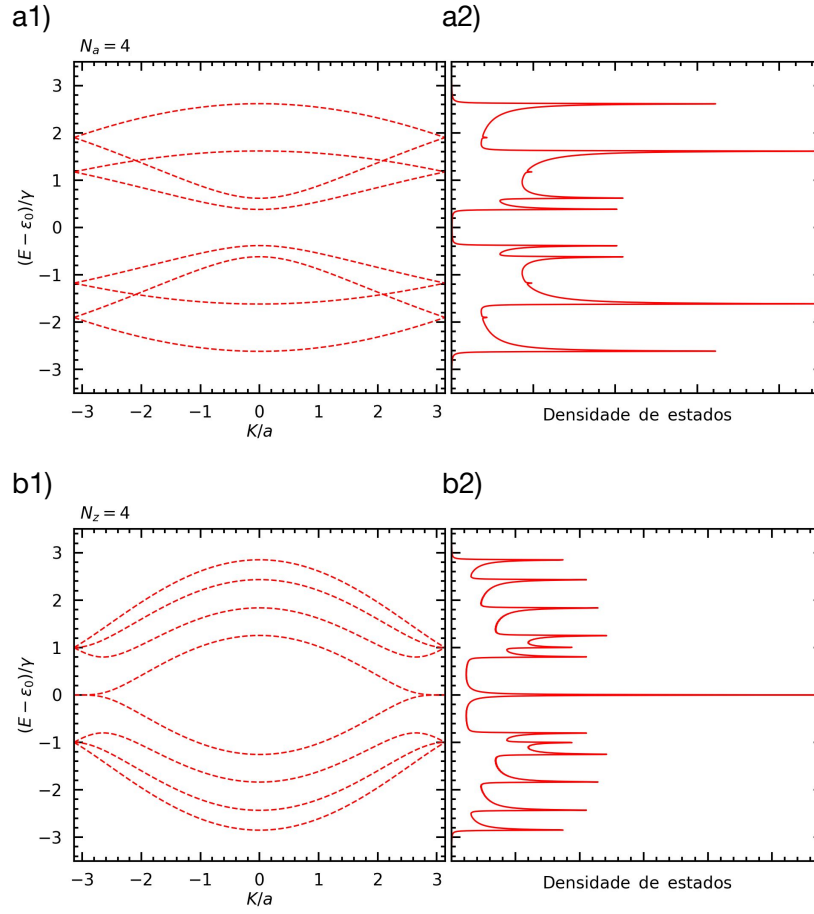


Figura 2.10: **Estruturas de bandas e Densidade de Estados:** Na coluna da esquerda, temos a relação de dispersão em função da energia, e na coluna da direita, temos a densidade de estados. As quantidades foram obtidas para nanofitas armchair (linha superior) e nanofitas (linha inferior) com $N_a = N_z = 4$. Nota-se que a nanofita armchair é semicondutora, enquanto a zig-zag é metálica, apresentando singularidade de van Hove (estados de borda) no nível de Fermi.

Aqui, l representa apenas os valores pares de i enquanto j representa os ímpares. Analogamente, para a simetria armchair,

$$\begin{aligned}
 \hat{H}_{N_a}(k) = & \sum_{m,i} \epsilon_i |m,i\rangle \langle m,i| + \gamma \sum_{2l+1} |m,l\rangle \langle m,l+2| + \gamma \sum_{2l+1} |m,l\rangle \langle m,l-2| + \\
 & + \gamma \sum_{2j+1} |j,m\rangle \langle j+2| + \gamma \sum_{2j+1} |m,j\rangle \langle m,j-2| \\
 & + \gamma \sum_{2l+2} |m,l\rangle \langle m,l+2| + \gamma \sum_{2l+1} |m,l\rangle \langle m,l-2|, \quad l = 1, 3, 5, \dots
 \end{aligned}$$

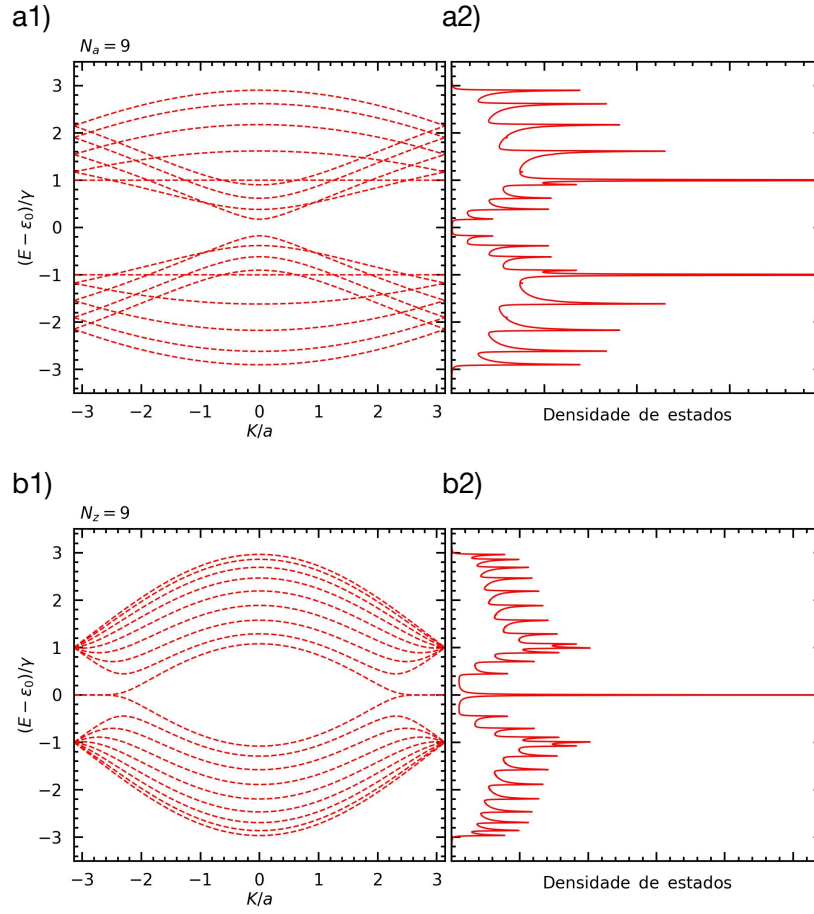


Figura 2.11: **Estruturas de bandas e Densidade de Estados:** Na coluna da esquerda, temos a relação de dispersão em função da energia, e na coluna da direita, temos a densidade de estados. As quantidades foram obtidas para nanofitas armchair (linha superior) e nanofitas (linha inferior) com $N_a = N_z = 9$. Nota-se que a nanofita armchair é semicondutora, enquanto a zig-zag é metálica, apresentando singularidade de van Hove (estados de borda) no nível de Fermi.

$$\begin{aligned}
 & +\gamma \sum_{2j+1} |m, j\rangle \langle m, j+1| + \gamma \sum_{2j+1} |m, j\rangle \langle m, j-2|, \quad j = 2, 4, 6, \dots \\
 & +e^{-ika} \gamma \sum_p |m, p\rangle \langle m-1, p+1| + e^{ika} \gamma \sum_q |m, q\rangle \langle m+1, q-1| \\
 & \quad p = 3, 7, 11, \dots, \quad q = 4, 8, 10, \dots
 \end{aligned} \tag{2.41}$$

Na forma matricial, para $N_z = 2$ e $N_a = 5$,

$$\hat{H}_{N_z=2} = \begin{pmatrix} \epsilon_1 & \gamma(1 + e^{-ika}) & 0 & 0 \\ \gamma(1 + e^{ika}) & \epsilon_2 & \gamma & 0 \\ 0 & \gamma & \epsilon_3 & \gamma(1 + e^{ika}) \\ 0 & 0 & \gamma(1 + e^{-ika}) & \epsilon_4 \end{pmatrix},$$

$$\hat{H}_{N_a=5} = \begin{pmatrix} \epsilon_1 & \gamma & \gamma & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma & \epsilon_2 & 0 & \gamma & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma & 0 & \epsilon_3 & \gamma e^{-ika} & \gamma & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma & \gamma e^{ika} & \epsilon_4 & 0 & \gamma & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma & 0 & \epsilon_5 & \gamma & \gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma & 0 & \epsilon_6 & \gamma & \gamma & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma & 0 & \epsilon_7 & \gamma e^{-ika} & \gamma & 0 \\ 0 & 0 & 0 & 0 & 0 & \gamma & \gamma e^{ika} & \epsilon_8 & 0 & \gamma \\ 0 & 0 & 0 & 0 & 0 & 0 & \gamma & 0 & \epsilon_9 & \gamma \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma & \gamma & \epsilon_{10} \end{pmatrix},$$

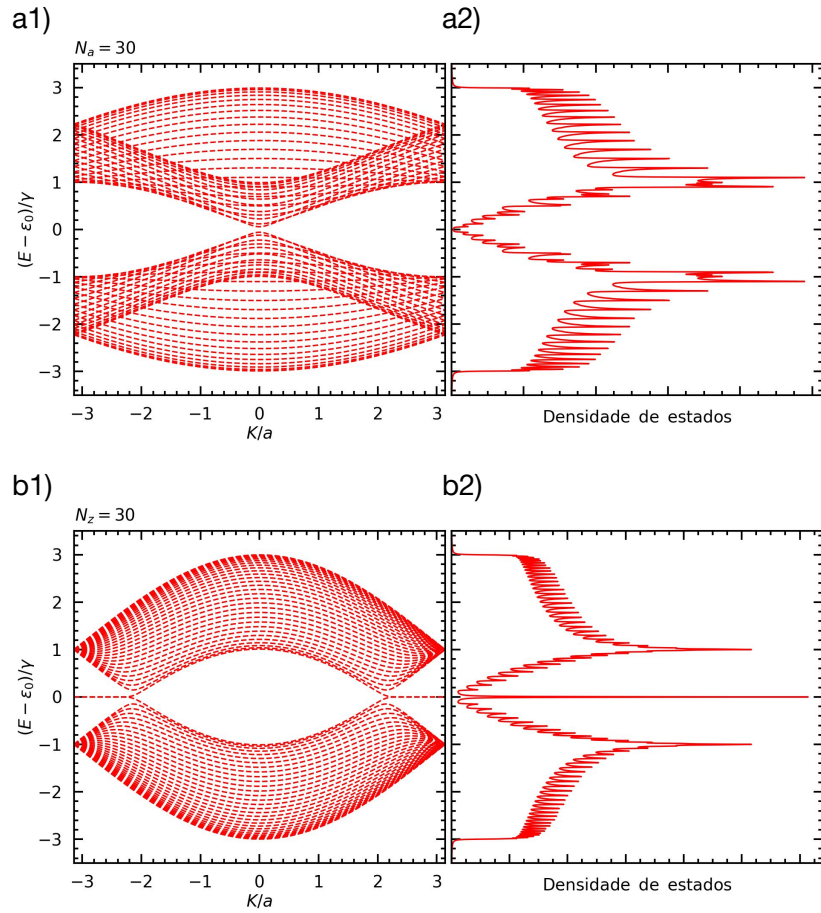


Figura 2.12: **Estruturas de bandas e Densidade de Estados:** Na coluna da esquerda, temos a relação de dispersão em função da energia, e na coluna da direita, temos a densidade de estados. As quantidades foram obtidas para nanofitas armchair (linha superior) e nanofitas (linha inferior) com $N_a = N_z = 30$. Nota-se que a nanofita armchair é semicondutora, enquanto a zig-zag é metálica, apresentando singularidade de van Hove (estados de borda) no nível de Fermi.

Em particular, ao tomar $\epsilon_i = \epsilon_0$ e $\gamma_{\langle i,j \rangle} = \gamma$,⁴ diagonalizamos o Hamiltoniano $\hat{H}$, e obtemos a relação de dispersão e a densidade de estados somando sobre todos os momentos na primeira zona de Brillouin.

Observamos nas Fig. 2.10, Fig. 2.11 e Fig. 2.12 divergências nas densidades de estados, conhecidas como singularidades de van Hove [52], que estão diretamente associadas aos valores onde há abertura ou fechamento de novas bandas. Para valores muito altos de N_z/a , mesmo que as estruturas de banda aparentem ser diferentes devido ao *folding* da zona de Brillouin, a densidade de estados converge para a DOS do grafeno (mostrada anteriormente na Fig. 2.4) e o cone é recuperado.

2.4 Funções Green

Como estamos interessados em sistemas abertos, nosso Hamiltoniano tem dimensão infinita, ao mesmo tempo, a simetria de translação é quebrada. Assim, buscamos uma forma alternativa de resolver o problema. Um operador importante para este trabalho é $\hat{G}^R$ [53], chamado de função de Green retardada, que está relacionado com $\hat{H}$ em uma base de orbitais localizados, através da equação $[E - \hat{H}]\hat{G}^R = \mathbb{1}_N$, onde $\mathbb{1}_N$ é a matriz identidade de ordem N . A função de Green avançada $\hat{G}^A$ é dada por $\hat{G}^A = (\hat{G}^R)^\dagger$. Explicitamente, $\hat{G}^R$ é definido como:

$$\hat{G}^R = \lim_{\eta \rightarrow 0^+} [(E + i\eta)\mathbb{1}_N - \hat{H}]^{-1} = \lim_{\eta \rightarrow 0^+} \sum_i \frac{1}{E + i\eta - \epsilon_i} |i\rangle \langle i|, \quad (2.42)$$

onde ϵ_i e $|i\rangle$ são ou autovalores e os autoestados de $\hat{H}$.

Usando a identidade,

$$\pi\delta(x) = \lim_{\eta \rightarrow 0^+} \frac{\eta}{x^2 + \eta^2}, \quad (2.43)$$

é fácil notar que a densidade de estados (Eq. 2.30) em termos da função de Green ($\hat{G}$)⁵ é dada por [24],

$$D(E) = -\frac{1}{\pi} \text{Im}\{\text{Tr } \hat{G}\}. \quad (2.44)$$

Uma das vantagens de uma descrição por meio de funções de Green é o fato de podermos resolver um Hamiltoniano complexo a partir de soluções para um problema mais simples conhecido. Assim, conseguimos expressar a função de

⁴Aqui $\langle i,j \rangle$ indica primeiros vizinhos

⁵Iremos omitir o sobrescrito R , assim $\hat{G} = \hat{G}^R$, a menos que seja necessário fazer um contraponto entre as funções retardada e avançada.

Green de maneira mais interessante através da equação de Dyson,

$$\hat{G} = \hat{G}_0 + \hat{G}_0 \hat{V} \hat{G}, \quad (2.45)$$

com $\hat{G}_0 = \lim_{\eta \rightarrow 0^+} [(E + i\eta)\mathbb{1}_N - \hat{H}_0]^{-1}$, representando o sistema desacoplado e $\hat{V}$ carregando a informação dos acoplamentos (interação). Na representação matricial, temos:

$$\hat{G}_0 = \begin{pmatrix} \hat{g}_1 & 0 & \dots & \dots & 0 \\ 0 & \hat{g}_2 & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & \dots & 0 & \hat{g}_N \end{pmatrix}, \quad (2.46)$$

com $\hat{g}_i = \lim_{\eta \rightarrow 0^+} [(E + i\eta)\mathbb{1} - \hat{h}_i]^{-1} e$

$$\hat{H}_0 = \begin{pmatrix} \hat{h}_1 & 0 & \dots & \dots & 0 \\ 0 & \hat{h}_2 & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & \dots & 0 & \hat{h}_N \end{pmatrix}, \quad (2.47)$$

$$\hat{V} = \begin{pmatrix} 0 & \hat{\gamma}_{1,2} & \dots & 0 & \hat{\gamma}_{1,N} \\ \hat{\gamma}_{2,1} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \hat{\gamma}_{N-1,N} \\ \hat{\gamma}_{N,1} & 0 & \dots & \hat{\gamma}_{N,N-1} & 0 \end{pmatrix}.^6 \quad (2.48)$$

2.4.1 Caso 2×2

Um exemplo interessante é o caso onde o sistema tem dimensões $N = 2$, e o Hamilton é expresso em termos de orbitais atômicos. Fixando $\gamma_{i,j} = \gamma$ (para primeiros vizinhos),

$$\hat{H} = \begin{pmatrix} \epsilon_1 & \gamma \\ \gamma & \epsilon_2 \end{pmatrix}. \quad (2.49)$$

⁶Nesta dissertação, iremos considerar um sistema aberto e, portanto, $\hat{\gamma}_{1,N} = \hat{\gamma}_{N,1} = 0$ e $N \rightarrow \infty$.

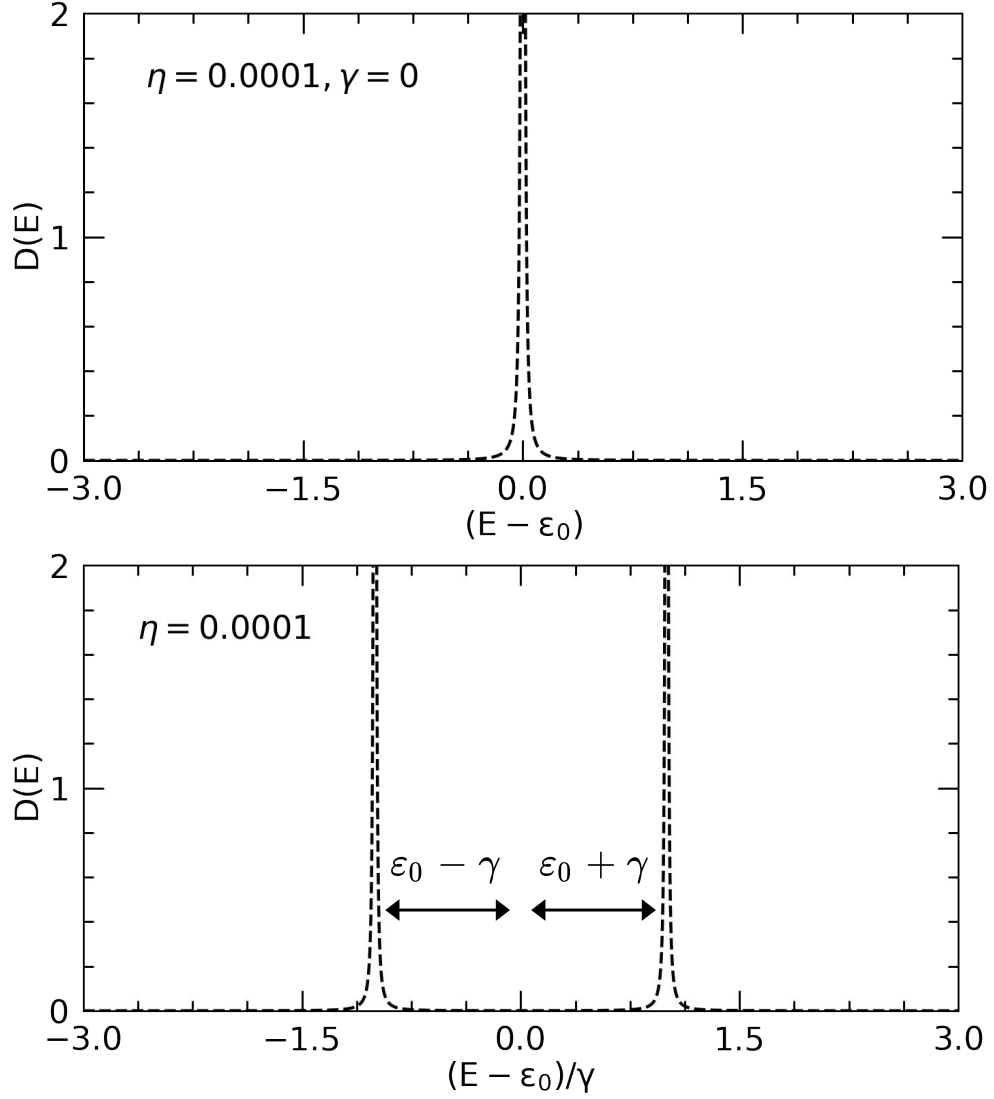


Figura 2.13: **Densidade de estados para o sistema 2×2** : densidade de estados em função de $E - \epsilon_0$ para dois orbitais atômicos *a*) desacoplados, $\gamma_{i,j} = 0$, e *b*) acoplados, com $\gamma_{i,j} = \gamma$. Aqui, tomamos $\eta \approx 10^{-4}$.

Utilizando a equação de Dyson (2.45), calculamos a função de Green,

$$\begin{pmatrix} G_{1,1} & G_{1,2} \\ G_{2,1} & G_{2,2} \end{pmatrix} = \begin{pmatrix} g_1 & 0 \\ 0 & g_2 \end{pmatrix} + \begin{pmatrix} g_1 & 0 \\ 0 & g_2 \end{pmatrix} \begin{pmatrix} 0 & \gamma \\ \gamma & 0 \end{pmatrix} \begin{pmatrix} G_{1,1} & G_{1,2} \\ G_{2,1} & G_{2,2} \end{pmatrix}. \quad (2.50)$$

Realizando os produtos,

$$\begin{pmatrix} G_{1,1} & G_{1,2} \\ G_{2,1} & G_{2,2} \end{pmatrix} = \begin{pmatrix} g_1 & 0 \\ 0 & g_2 \end{pmatrix} + \begin{pmatrix} g_1 \gamma G_{2,1} & g_1 \gamma G_{2,2} \\ g_2 \gamma G_{1,1} & g_2 \gamma G_{1,2} \end{pmatrix}, \quad (2.51)$$

temos, que

$$G_{1,1} = g_1 + g_1 \gamma G_{2,1}, \quad (2.52)$$

$$G_{2,1} = g_2 \gamma G_{1,1}, \quad (2.53)$$

$$G_{2,2} = g_3 + g_2 \gamma G_{1,2}, \quad (2.54)$$

$$G_{1,2} = g_1 \gamma G_{2,2}. \quad (2.55)$$

Temos equações para os termos da diagonal para $G_{1,1}$

$$\begin{aligned} G_{1,1} &= g_1 + g_1 \gamma g_2 \gamma G_{1,1} \\ g_1^{-1} G_{1,1} &= 1 + \gamma g_2 \gamma G_{1,1} \\ G_{1,1} &= [g_1^{-1} - \gamma g_2 \gamma]^{-1}, \end{aligned} \quad (2.56)$$

e, analogamente,

$$G_{2,2} = [g_2^{-1} - \gamma g_1 \gamma]^{-1}. \quad (2.57)$$

Por definição, no caso em que os orbitais são desacoplados ($\gamma = 0$), a partir de 2.56 e 2.57, temos $G_{i,i}^0 = g_i$, onde $g_i = 1/(E + i\eta - \epsilon_i)$. Escrevendo de uma maneira mais conveniente,

$$\begin{aligned} g_i &= \lim_{\eta \rightarrow 0^+} \frac{1}{E + i\eta - \epsilon_i} \frac{E - i\eta - \epsilon_i}{E - i\eta - \epsilon_i} = \lim_{\eta \rightarrow 0^+} \frac{E - i\eta - \epsilon_i}{(E - \epsilon_i)^2 + \eta^2} \\ g_i &= \lim_{\eta \rightarrow 0^+} \left[\frac{E - \epsilon_i}{(E - \epsilon_i)^2 + \eta^2} - i \frac{\eta}{(E - \epsilon_i)^2 + \eta^2} \right]. \end{aligned} \quad (2.58)$$

Assim, temos a densidade de estados diretamente de 2.58. De maneira geral $G_{1,1}$ é dada por:

$$G_{1,1} = \lim_{\eta \rightarrow 0^+} \frac{1}{E + i\eta - \epsilon_1 - \gamma^2 g_2} = \lim_{\eta \rightarrow 0^+} \frac{1}{E + i\eta - \epsilon_1 - \gamma^2 \frac{1}{E + i\eta - \epsilon_2}},$$

de modo que,

$$Re(G_{1,1}) = \frac{(E - \epsilon_1)[(E - \epsilon_2)^2 + \eta^2] - \gamma^2(E - \epsilon_2)}{\eta^2[2\gamma^2 + (E - \epsilon_1)^2 + (E - \epsilon_2)^2] + [\gamma^2 - (E - \epsilon_1)(E - \epsilon_2)]^2 + \eta^4}, \quad (2.59)$$

e

$$Im(G_{1,1}) = -\frac{\eta[(E - \epsilon_2)^2 + \gamma^2 + \eta^2]}{\eta^2[2\gamma^2 + (E - \epsilon_1)^2 + (E - \epsilon_2)^2] + [\gamma^2 - (E - \epsilon_1)(E - \epsilon_2)]^2 + \eta^4}. \quad (2.60)$$

Analogamente,

$$Re(G_{2,2}) = \frac{(E - \epsilon_2)[(E - \epsilon_1)^2 + \eta^2] - \gamma^2(E - \epsilon_1)}{\eta^2[2\gamma^2 + (E - \epsilon_1)^2 + (E - \epsilon_2)^2] + [\gamma^2 - (E - \epsilon_1)(E - \epsilon_2)]^2 + \eta^4}, \quad (2.61)$$

e

$$Im(G_{2,2}) = -\frac{\eta[(E - \epsilon_1)^2 + \gamma^2 + \eta^2]}{\eta^2[2\gamma^2 + (E - \epsilon_1)^2 + (E - \epsilon_2)^2] + [\gamma^2 - (E - \epsilon_1)(E - \epsilon_2)]^2 + \eta^4}. \quad (2.62)$$

A partir disso, podemos calcular a densidade de estados da Eq. 2.42

$$D(E) = -\frac{1}{\pi}[Im(G_{1,1}) + Im(G_{2,2})]. \quad (2.63)$$

Por simplicidade, mas ainda contendo informações importantes sobre o problema, tomamos $\epsilon_1 = \epsilon_2 = \epsilon_0$,

$$D(E, \eta) = \frac{2}{\pi} \left\{ \frac{\eta[(E - \epsilon_0)^2 + \eta^2 + \gamma^2]}{(E - \epsilon_0)^4 + 2(E - \epsilon_0)^2(\eta^2 - \gamma^2) + (\gamma^2 + \eta^2)^2} \right\} \quad (2.64)$$

com

$$D(E) = \lim_{\eta \rightarrow 0^+} D(E, \eta). \quad (2.65)$$

Observando a solução na Fig. 2.13 fixando $\eta = 0.0001$, notamos que a densidade de estados é formada por duas funções δ em $\epsilon_0 \pm \gamma$. No caso em que $\gamma = 0$, os estados são degenerados em ϵ_0 . Quando ligamos o acoplamento, o efeito é um *split* nos níveis em relação ao caso desacoplado, levando os níveis para $\epsilon_0 - \gamma$ e $\epsilon_0 + \gamma$. O mesmo resultado pode ser obtido diagonalizando o Hamiltoniano completo $\hat{H}$.

2.5 Sistemas Abertos

Até este ponto, tratamos apenas de sistemas periódicos. No entanto, para estudar propriedades eletrônicas e de transporte em um dispositivo realista, precisamos descrever o que chamamos de sistema aberto. Vamos acoplar a parte ativa

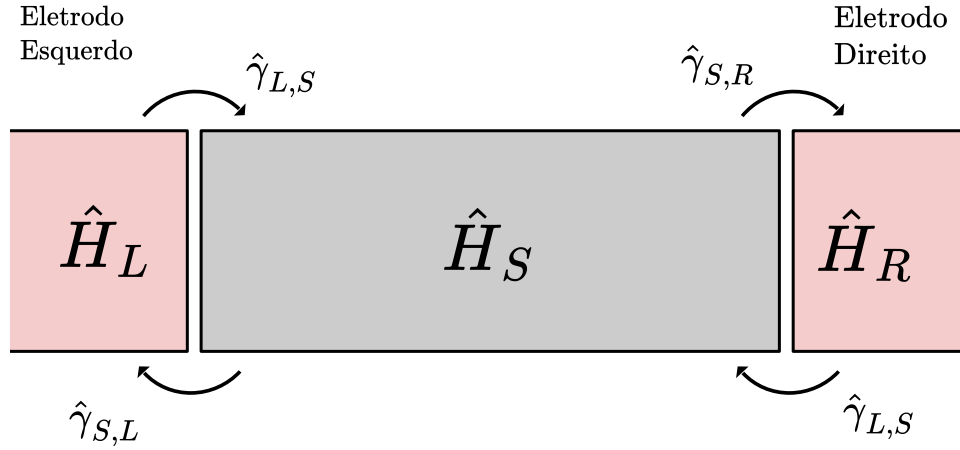


Figura 2.14: **Visão esquemática de um problema de transporte eletrônico:** As regiões em vermelho representam os eletrodos, onde $\hat{H}_L$ e $\hat{H}_R$ denotam os Hamiltonianos do eletrodo esquerdo e direito, respectivamente. Em cinza, temos a região de espalhamento, com $\hat{H}_S$ o seu Hamiltoniano e o acoplamento entre diferentes regiões representado por $\hat{\gamma}_{l,l'}$, onde $l = L, S, R$.

do dispositivo a reservatórios de elétrons e dividir o sistema resultante em três regiões [54]: $\hat{H}_L$ e $\hat{H}_R$, denominadas eletrodos esquerdo e direito, respectivamente, e a região central $\hat{H}_S$, a região de espalhamento, como ilustrado na Fig. 2.14. Inicialmente, consideramos apenas funções de onda vindas do infinito, dadas pelos estados de Bloch (2.21). Aqui, consideramos que as funções de onda provenientes de $-\infty$ e $+\infty$ são originárias dos eletrodos, que podem ter estrutura interna. Em particular, para este trabalho, consideramos estruturas cristalinas. Essas estruturas podem ser iguais ou diferentes da estrutura da região de espalhamento. Assim, $\hat{H}_L$ e $\hat{H}_R$ têm dimensões semi-infinitas, enquanto $\hat{H}_S$ possui dimensões finitas. O Hamiltoniano completo é dado por:

$$\hat{H} = \hat{H}_L + \hat{H}_S + \hat{H}_R + \hat{\gamma}_L + \hat{\gamma}_R. \quad (2.66)$$

Aqui, $\hat{\gamma}_{L/(R)} = \hat{\gamma}_{L/(R),S} + \hat{\gamma}_{S,L/(R)}$ é o acoplamento entre a região de espalhamento e o eletrodo esquerdo (direito). Aqui, a princípio, não temos mais simetria de translação, portanto, não é possível utilizar o teorema de Bloch. No entanto, como mostramos na seção anterior, podemos utilizar funções de Green para tratar esse problema. Para a função de Green, usaremos a equação de Dyson (2.45). Tomando

$\hat{g}_i = \lim_{\eta \rightarrow 0^+} [(E + i\eta)\mathbb{1}_N - \hat{H}_i]^{-1}$, para $i = L, R, S$, temos:

$$\hat{H}_0 = \begin{pmatrix} \hat{H}_L & 0 & 0 \\ 0 & \hat{H}_S & 0 \\ 0 & 0 & \hat{H}_R \end{pmatrix} \quad \text{e} \quad \hat{G}_0 = \begin{pmatrix} \hat{g}_L & 0 & 0 \\ 0 & \hat{g}_S & 0 \\ 0 & 0 & \hat{g}_R \end{pmatrix}. \quad (2.67)$$

A matriz que carrega a informação dos acoplamentos é dada por:

$$\hat{V} = \begin{pmatrix} 0 & \hat{\gamma}_{L,S} & 0 \\ \hat{\gamma}_{S,L} & 0 & \hat{\gamma}_{S,R} \\ 0 & \hat{\gamma}_{R,S} & 0 \end{pmatrix}, \quad (2.68)$$

e a função de Green completa,

$$\begin{aligned} \hat{G} &= \begin{pmatrix} \hat{G}_L & \hat{G}_{L,S} & \hat{G}_{L,R} \\ \hat{G}_{S,L} & \hat{G}_S & \hat{G}_{S,R} \\ \hat{G}_{R,L} & \hat{G}_{R,S} & \hat{G}_R \end{pmatrix} \\ &= \left[\begin{pmatrix} (E + i\eta)\mathbb{1}_{N_L} & 0 & 0 \\ 0 & (E + i\eta)\mathbb{1}_{N_S} & 0 \\ 0 & 0 & (E + i\eta)\mathbb{1}_{N_R} \end{pmatrix} - \begin{pmatrix} \hat{H}_L & \hat{\gamma}_{L,S} & 0 \\ \hat{\gamma}_{S,L} & \hat{H}_S & \hat{\gamma}_{S,R} \\ 0 & \hat{\gamma}_{R,S} & \hat{H}_R \end{pmatrix} \right]^{-1}. \end{aligned} \quad (2.69)$$

Realizando as operações de forma análoga à Eq. 2.56 e expressando diretamente o termo da região de espalhamento, obtemos:

$$\hat{G}_S = \lim_{\eta \rightarrow 0^+} [(E + i\eta)\mathbb{1}_{N_S} - \hat{H}_S - \Sigma_L - \Sigma_R]^{-1}, \quad (2.70)$$

com

$$\begin{aligned} \Sigma_L &= \hat{\gamma}_{L,S} \hat{g}_L \hat{\gamma}_{S,L} = \hat{\gamma}_{L,S} [(E + i\eta)\mathbb{1}_{N_L} - \hat{H}_L]^{-1} \hat{\gamma}_{S,L}, \\ \Sigma_R &= \hat{\gamma}_{R,S} \hat{g}_R \hat{\gamma}_{S,R} = \hat{\gamma}_{R,S} [(E + i\eta)\mathbb{1}_{N_R} - \hat{H}_R]^{-1} \hat{\gamma}_{S,R}, \end{aligned} \quad (2.71)$$

as chamadas auto-energias dos eletrodos. Assim, a função de Green da região de espalhamento contém toda a informação de interesse. Note que as auto-energias possuem a mesma dimensão de $\hat{H}_S$, porém dependem explicitamente da energia, E . Isso ocorre porque a parte de $\hat{g}_{L/R}$ que contribui é efetivamente o termo de $\hat{H}_{L/R}$ que acopla com a região de espalhamento. Desta forma, só precisamos das funções de Green de superfície do eletrodo.⁷ Na próxima subseção, discutiremos

⁷Consideramos aqui que o acoplamento ocorre apenas entre a região de espalhamento e um número finito de estados nos eletrodos. Uma descrição utilizando o modelo TB é particularmente

como calcular explicitamente as funções de Green de superfície e as auto-energias dos eletrodos de maneira geral e independente da estrutura interna.

apropriada, já que é localizada no espaço real.

2.5.1 Método da função de Green recursiva

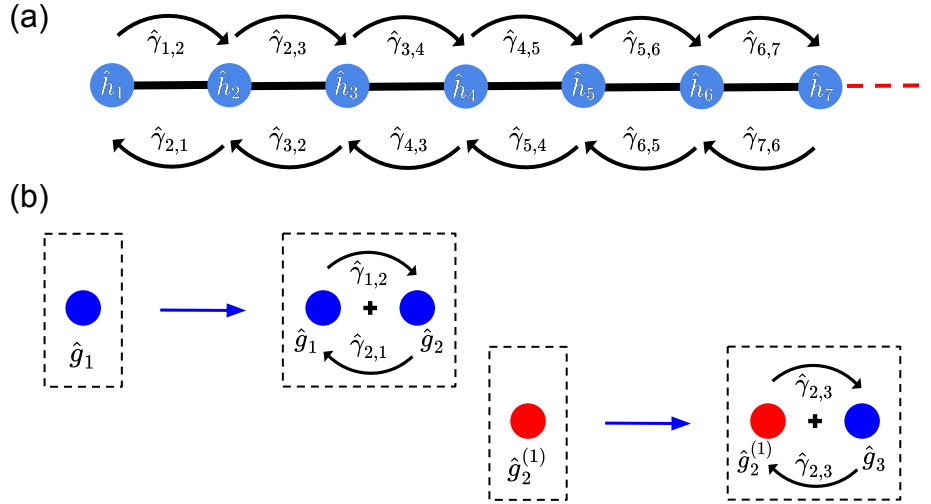


Figura 2.15: **Visão esquemática do Método da função de Green recursiva:** a) Uma cadeia linear infinita, onde os Hamiltonianos dos sítios $\hat{h}_m$ acoplam apenas com seus vizinhos próximos, e $\hat{\gamma}_{m,n}$ é o acoplamento. b) Representação esquemática do processo recursivo. Temos, a princípio, apenas um sítio $\hat{h}_1$ com sua função de green associada $\hat{g}_1$. Depois, ele é acoplado com o sítio posterior, obtendo a função de green efetiva entre os dois sítios $\hat{g}_1^2$. Esse processo é repetido novamente.

De maneira explícita, temos as auto-energias Σ_L e Σ_R em termos de suas respectivas funções de Green de superfície g_L e g_R , da seguinte maneira:

$$\Sigma_L = \hat{\gamma}_{L,S} \hat{g}_L \hat{\gamma}_{S,L} \quad \text{e} \quad \Sigma_R = \hat{\gamma}_{R,S} \hat{g}_R \hat{\gamma}_{S,R}. \quad (2.72)$$

O método que usaremos para calcular as funções de Green de superfície $\hat{g}_L$ e $\hat{g}_R$ é chamado método recursivo das funções de Green [55]. Inicialmente, representamos uma cadeia linear com acoplamento entre os primeiros vizinhos, como ilustrado na Fig. 2.15(a). $\hat{h}_m$ é o Hamiltoniano intracelular com sítios acoplados internamente, dada pela Eq. 2.24, e $\hat{\gamma}_{m,n}$ é o acoplamento entre as células m e n . Em princípio, o acoplamento entre as células é arbitrário. No entanto, consideramos acoplamentos apenas entre células vizinhas. Aqui apresentaremos o método de maneira geral, que vale para sistemas 1D e ordem inferior. Em particular, temos interesse em redes 1D com simetria translacional, e para esses casos $\hat{h}_m$ é a estrutura de repetição mínima, ou seja, é a célula unitária, que para o grafeno é dada por $\hat{h}_m = \hat{h}_0$ como ilustrado na Fig. 2.2(b). O interesse é descrever elétrons vindos do infinito em um eletrodo com estrutura cristalina, e em equilíbrio termodinâmico.

Teremos então um sistema com dimensões infinitas. Olhando para a representação matricial de $\hat{h}$, temos:

$$\hat{h}_{L/R} = \begin{pmatrix} \hat{h}_1 & \hat{\gamma}_{1,2} & \dots & \dots & 0 \\ \hat{\gamma}_{2,1} & \hat{h}_2 & \vdots & \vdots & 0 \\ \vdots & \vdots & \ddots & \hat{\gamma}_{N-1,N} & \vdots \\ \vdots & \vdots & \hat{\gamma}_{N,N-1} & \hat{h}_N & \vdots \\ 0 & \dots & \dots & \dots & \ddots \end{pmatrix}. \quad (2.73)$$

Separando os termos locais (diagonais) dos termos de acoplamento (fora da diagonal), obtemos:

$$\hat{H}_0 = \begin{pmatrix} \hat{h}_1 & 0 & \dots & \dots & 0 \\ 0 & \hat{h}_2 & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \ddots \end{pmatrix}, V = \begin{pmatrix} 0 & \hat{\gamma}_{1,2} & \dots & \dots & 0 \\ \hat{\gamma}_{2,1} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \ddots \end{pmatrix}.$$

Com isso, podemos usar a equação de Dyson (Eq. 2.45) para obter a função de Green de superfície.

2.5.2 Caso 3×3

Para uma compreensão mais clara, consideraremos inicialmente o caso de uma rede 3×3 . A função de Green do sistema desacoplado e a matriz de acoplamento são, respectivamente, dadas por:

$$\hat{G}_0 = \begin{pmatrix} \hat{g}_1 & 0 & 0 \\ 0 & \hat{g}_2 & 0 \\ 0 & 0 & \hat{g}_3 \end{pmatrix},$$

$$\hat{V} = \begin{pmatrix} 0 & \hat{\gamma}_{1,2} & 0 \\ \hat{\gamma}_{2,1} & 0 & \hat{\gamma}_{2,3} \\ 0 & \hat{\gamma}_{3,2} & 0 \end{pmatrix}.$$

A função de Green total do sistema apresenta a seguinte estrutura:

$$\hat{G} = \begin{pmatrix} \hat{G}_{1,1} & \hat{G}_{1,2} & \hat{G}_{1,3} \\ \hat{G}_{2,1} & \hat{G}_{2,2} & \hat{G}_{2,3} \\ \hat{G}_{3,1} & \hat{G}_{3,2} & \hat{G}_{3,3} \end{pmatrix}.$$

Eliminamos os graus de liberdade de $\hat{h}_1$ reescrevendo $\hat{g}_2$ com a contribuição explícita de $\hat{h}_1$. Isso equivale a resolver primeiro o sistema 2×2 . Neste caso,

$$\hat{G} = \begin{pmatrix} \hat{G}_{1,1} & \hat{G}_{1,2} \\ \hat{G}_{2,1} & \hat{G}_{2,2} \end{pmatrix}.$$

A equação de Dyson tem a estrutura:

$$\begin{pmatrix} \hat{G}_{1,1} & \hat{G}_{1,2} \\ \hat{G}_{2,1} & \hat{G}_{2,2} \end{pmatrix} = \begin{pmatrix} \hat{g}_1 & 0 \\ 0 & \hat{g}_2 \end{pmatrix} + \begin{pmatrix} \hat{g}_1 & 0 \\ 0 & \hat{g}_2 \end{pmatrix} \begin{pmatrix} 0 & \hat{\gamma}_{1,2} \\ \hat{\gamma}_{2,1} & 0 \end{pmatrix} \begin{pmatrix} \hat{G}_{1,1} & \hat{G}_{1,2} \\ \hat{G}_{2,1} & \hat{G}_{2,2} \end{pmatrix}.$$

Realizando os produtos e escrevendo explicitamente os termos locais da função de Green,

$$\hat{G}_{1,1} = \hat{g}_1 + \hat{g}_1 \hat{\gamma}_{1,2} \hat{G}_{2,1},$$

$$\hat{G}_{2,1} = \hat{g}_2 \hat{\gamma}_{2,1} \hat{G}_{1,1},$$

$$\hat{G}_{2,2} = \hat{g}_2 + \hat{g}_2 \hat{\gamma}_{2,1} \hat{G}_{1,2},$$

$$\hat{G}_{1,2} = \hat{G}_{1,1} \hat{\gamma}_{1,2} \hat{G}_{2,2},$$

de modo que,

$$\hat{g}_1^{(1)} = \hat{G}_{1,1} = [(\hat{g}_1)^{-1} - \hat{\gamma}_{1,2} \hat{g}_2 \hat{\gamma}_{2,1}]^{-1},$$

$$\hat{g}_2^{(1)} = \hat{G}_{2,2} = [(\hat{g}_2)^{-1} - \hat{\gamma}_{2,1} \hat{g}_1 \hat{\gamma}_{1,2}]^{-1}.$$

Repetindo o procedimento e acoplando o segmento 3 ao segmento 2 efetivo, obtemos:⁸

$$\hat{G}_0^{(1)} = \begin{pmatrix} \hat{g}_2^{(1)} & 0 \\ 0 & \hat{g}_3 \end{pmatrix},$$

$$\hat{V} = \begin{pmatrix} 0 & \hat{\gamma}_{2,3} \\ \hat{\gamma}_{3,2} & 0 \end{pmatrix}.$$

⁸Notamos que não é necessario fazer o processo recursivo sobre $\hat{g}_1^{(1)}$, já que $\hat{\gamma}_{1,3} = 0$.

Podemos repetir o procedimento com a equação de Dyson, como ilustrado pela Fig. 2.15(b). Por fim, o termo $\hat{G}_{3,3}$ é dado por:

$$\hat{g}_3^{(2)} = \hat{G}_{3,3} = [(E + i\eta)\mathbb{1}_{N_3} - \hat{h}_3 - \hat{\gamma}_{3,2}\hat{g}_2^{(1)}\hat{\gamma}_{2,3}]^{-1}.$$

Isto é equivalente a obter $G_{3,3}$ diretamente da equação de Dyson escrita em termos das matrizes dadas pela Eq. 2.5.2, resolvendo o sistema 3×3 . Assim, obtemos a função de Green local para uma camada arbitrária N iterando $N - 1$ vezes, o que corresponde a decimar os graus de liberdade do sistema. De maneira geral, o termo da função de Green local, para um sistema de ordem N arbitrária, é dado por:

$$\hat{G}_{N,N} = \hat{g}_N^{(N-1)} = [(E + i\eta)\mathbb{1}_{N_N} - \hat{h}_N - \hat{\gamma}_{N,N-1}\hat{g}_{N-1}^{(N-2)}\hat{\gamma}_{N-1,N}]^{-1}. \quad (2.74)$$

A função de Green de superfície é dada pelo último termo no limite em que $N \rightarrow \infty$, ou seja,

$$\hat{g}_L = \lim_{N \rightarrow \infty} \hat{G}_{N,N}.$$

De forma análoga, obtemos $\hat{g}_R$ acoplando os segmentos no sentido da direita para esquerda. Analogamente, se iteramos no sentido contrário, obtemos $\hat{g}_R$ (Fig. 2.15(a)). Do ponto de vista numérico, é possível notar que o cálculo das auto-energias pode ser implementado como um processo iterativo. Como o sistema é cristalino, há garantia de convergência conforme aumentamos o número de camadas.

Olhando para alguns exemplos de interesse:

- Cadeia linear semi-infinita: temos aqui uma cadeia com um orbital por sítio, portanto, $\hat{h}_m = \epsilon_0$ e $\hat{\gamma}_{m,n} = \gamma_0$ são escalares. Conseguimos obter analiticamente as funções de Green de superfície nos dois sentidos, e devido às simetrias do problema, elas são idênticas e dadas por:

$$g_{L/R} = \frac{e^{-ik}}{\gamma} \quad \text{com} \quad k = \arccos \frac{E - \epsilon_0}{2\gamma}. \quad (2.75)$$

Mostramos o comportando de maneira explícita na Fig. 2.16.

- Nanofita de Grafeno semi-infinito: Analogamente a uma cadeia linear, com cada célula unitária tendo estrutura interna. Em particular, para nanofitas armchair, temos $\hat{h}_0 = \hat{H}_{N_a}$, $\hat{\gamma}_{m,n} = \hat{\gamma}_0$ e $\hat{\gamma}_{n,m} = \hat{\gamma}_0^T$, as matrizes seguindo

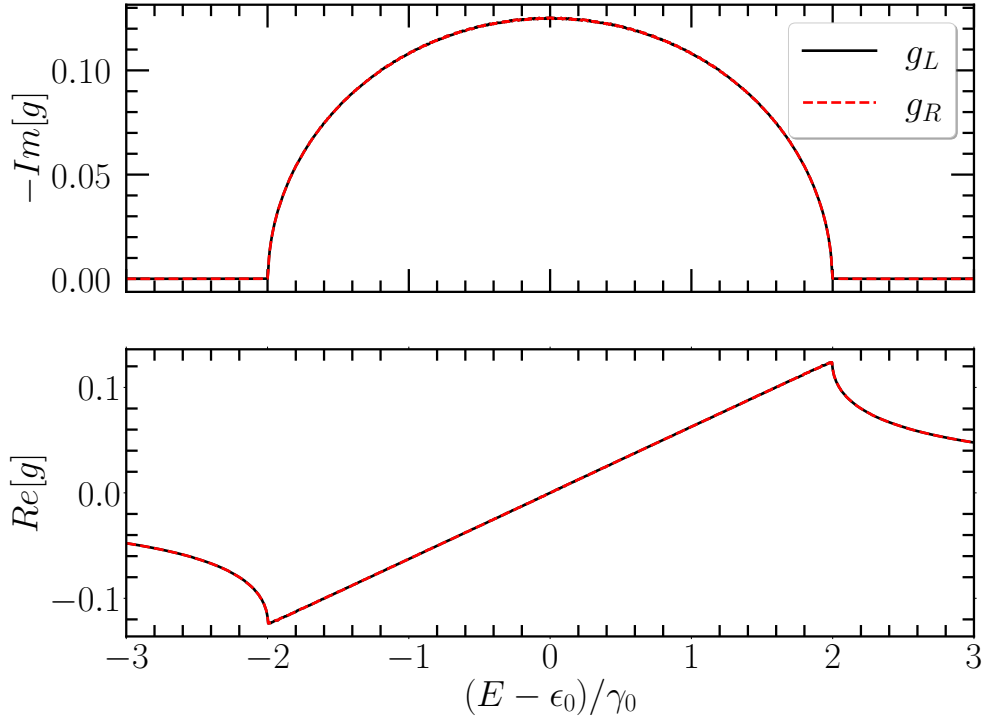


Figura 2.16: **Funções de Green semi-infinita:** Na linha superior, temos o negativo da parte imaginária da função de Green ($-\text{Im}[g]$), e na linha inferior, a parte real ($\text{Re}[g]$), em função da energia para uma cadeia atômica linear.

a mesma construção da célula unitária da nanofita (apresentada na Subsec. 2.3.1), acoplando células em segmentos diferentes como ilustrado na Fig. 2.2(b). Considerando que o acoplamento à região de espalhamento tem a mesma estrutura do acoplamento entre elementos da fita, obtemos a auto-energia $\Sigma_L = \gamma_0 \hat{g}_L \gamma_0^T$ e $\Sigma_R = \Sigma_L^T$. Calculamos o traço sobre as auto-energias, como mostrado na Figura 2.17, de modo que a parte imaginária corresponde à densidade de estados na superfície.

- Nanotubo de carbono semi-infinito: As células unitárias dos nanotubos de carbono são essencialmente as mesmas das nanofitas de grafeno, porém com o primeiro elemento da célula unitária acoplado ao último, ou seja,

$$\hat{H}_{N_{ca}=2} = \begin{pmatrix} \epsilon_1 & \gamma & 0 & \gamma \\ \gamma & \epsilon_2 & \gamma & 0 \\ 0 & \gamma & \epsilon_3 & \gamma \\ \gamma & 0 & \gamma & \epsilon_4 \end{pmatrix},$$

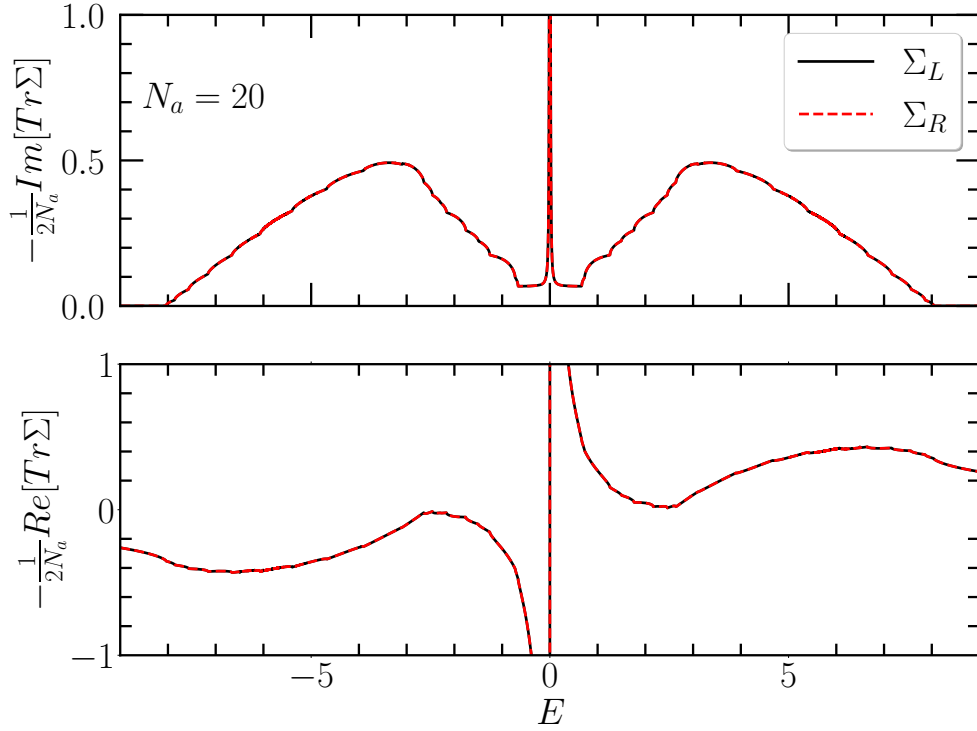


Figura 2.17: **Auto-energias para uma nanofita de grafeno:** Na linha superior tem o negativo da parte imaginária do traço sobre as auto-energias ($Im[\Sigma_{L/R}]$) e na linha inferior a parte real ($Re[\Sigma_{L/R}]$) em função da energia para uma nanofita semi-infinita de grafeno. Com a linha vermelha e preta referentes aos termos esquerdo (Σ_L) e direito (Σ_R) respectivamente, para $N_a = 20$.

De maneira geral

$$\hat{H}_{N_{cz}} = \hat{H}_{N_a} + \gamma \otimes \mathbb{I}_{0,2N_a}, \quad (2.76)$$

e

$$\hat{H}_{N_{ca}} = \hat{H}_{N_z} + \gamma \otimes \mathbb{I}_{0,2N_z}, \quad (2.77)$$

com

$$\mathbb{I}_{0,N} = \begin{pmatrix} 0 & \dots & \dots & 1 \\ \vdots & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \vdots \\ 1 & \dots & \dots & 0 \end{pmatrix}, \quad (2.78)$$

a matriz com todos os elementos nulos, por exceção dos termos extremos da antidiagonal, onde o índice N refere-se à dimensão $N \times N$ da matriz.⁹ Assim, obtemos a auto-energia analogamente ao caso anterior, como mostrado na

⁹Note que a nomenclatura 'zig-zag' e 'armchair' é invertida no caso de nanotubos.

Fig. 2.18.

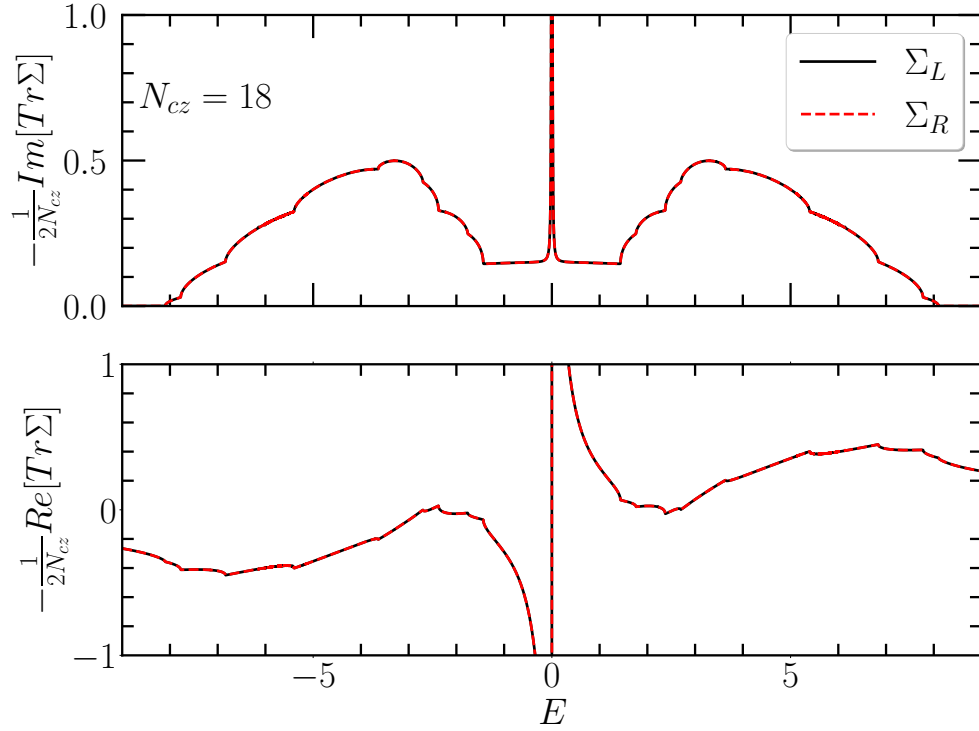


Figura 2.18: **Auto-energias para um nanotubo de carbono:** Na linha superior, temos o negativo da parte imaginária do traço sobre as auto-energias ($\text{Im}[\Sigma_{L/R}]$), e na linha inferior, a parte real ($\text{Re}[\Sigma_{L/R}]$), em função da energia para um nanotubo de carbono semi-infinito. A linha vermelha e preta referem-se aos termos esquerdo (Σ_L) e direito (Σ_R), respectivamente, com $N_{cz} = 18$.

2.6 Interação elétron-elétron

Nas seções anteriores, discutimos métodos para descrever propriedades eletrônicas na presença de um potencial periódico. No entanto, um ingrediente importante que precisamos considerar em uma descrição mais realista é a presença da interação entre elétrons por meio da repulsão Coulombiana. Nesta seção, discutiremos como descrever a interação elétron-elétron em estruturas cristalinas.

2.6.1 Modelo de Hubbard

O modelo de Hubbard (MH) consiste na descrição de uma rede em que os elétrons interagem entre si através da repulsão Coulombiana, podendo ocorrer interações no mesmo ou em diferentes orbitais. O Hamiltoniano do MH é dado por

uma descrição TB em segunda quantização, adicionando um termo de interação:

$$\hat{H} = \sum_{i,j} \langle i|h|j \rangle \hat{c}_i^\dagger \hat{c}_j + \sum_{i,j,k,l} \langle ik|v|jl \rangle \hat{c}_i^\dagger \hat{c}_k \hat{c}_j^\dagger \hat{c}_l. \quad (2.79)$$

Os índices i, j, k, l pertencem à base completa de partícula única. Se (i) considerarmos apenas um orbital por sítio, e (ii) o potencial Coulombiano ($v_{i,k,j,l} = U_c$), temos:

$$\hat{H}^0 = \sum_{i,\sigma} \epsilon_i \hat{n}_{i\sigma} + \sum_{i,j} \gamma_{ij} \hat{c}_i^\dagger \hat{c}_j - \mu \sum_i (\hat{n}_{i\uparrow} + \hat{n}_{i\downarrow}), \quad (2.80)$$

e a interação Coulombiana será dada por [56, 57]:

$$\hat{H}^{ee} = \frac{U}{2} \sum_i \hat{n}_{i\uparrow} \hat{n}_{i\downarrow} + \frac{2U' - J}{4} \sum_{\sigma,\sigma',i,j} \hat{n}_{i,\sigma} \hat{n}_{j,\sigma'} - J \sum_{i,j} \vec{S}_i \cdot \vec{S}_j - \frac{J}{2} \sum_{\sigma,i} \hat{c}_{i,\sigma}^\dagger \hat{c}_{j,\sigma}^\dagger \hat{c}_{j,\sigma} \hat{c}_{i,\sigma}. \quad (2.81)$$

Analogamente ao modelo TB, ϵ_i é a energia do i -ésimo orbital, γ_{ij} é o acoplamento entre diferentes orbitais, e μ é o potencial químico do sistema não interagente. Para o Hamiltoniano interagente, U é a intensidade do potencial de repulsão Coulombiana entre elétrons com projeções de spin diferentes no mesmo orbital, U' e J são a intensidade do potencial entre elétrons em diferentes órbitas com projeções de spin iguais ou diferentes. Em particular, J está relacionado ao acoplamento de Hund [57]. Além disso, $U' = U - 2J$ e os coeficientes seguem a hierarquia $U > U' > J$.

De maneira geral, o Hamiltoniano completo do sistema é

$$\hat{H} = \hat{H}^0 + \hat{H}^{ee}, \quad (2.82)$$

uma soma entre os termos não interagente e o termo que carrega toda a informação da interação entre os elétrons do sistema. Se considerarmos ainda que só existe interação entre elétrons no mesmo orbital, $U' = J_H = 0$, e por simplicidade tomamos $\mu = 0$, temos então o modelo de Hubbard em sua forma usual

$$H = \sum_i \epsilon_i (\hat{n}_{i\uparrow} + \hat{n}_{i\downarrow}) + \sum_i \gamma_{ij} \hat{c}_i^\dagger \hat{c}_j + U \sum_i \hat{n}_{i\uparrow} \hat{n}_{i\downarrow}. \quad (2.83)$$

Apesar de ser uma simplificação das equações 2.79 e 2.82, ainda é difícil de solucionar. Inicialmente, ao examinarmos alguns limites com soluções analíticas, conseguimos observar alguns efeitos da presença da interação entre elétrons.

- Limite de delocalização: $U/t \ll 1$

Nesse limite, podemos ignorar o termo de interação, e a expressão para Hamiltoniano tem a estrutura:

$$\hat{H} = \sum_i \epsilon_i (\hat{n}_{i\uparrow} + \hat{n}_{i\downarrow}) + \sum_i \gamma_{ij} \hat{c}_i^\dagger \hat{c}_j. \quad (2.84)$$

Usando os estados de Bloch para um sistema cristalino,

$$|k\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle e^{ikR_i}, \quad (2.85)$$

e realizando a transformada inversa,

$$|i\rangle = \frac{1}{\sqrt{N}} \sum_k |k\rangle e^{-ikR_i}. \quad (2.86)$$

Escrevendo os operadores de criação e aniquilação em termos dos estados de Bloch na representação dos momentos, temos:

$$\begin{aligned} \hat{c}_i^\dagger &= \frac{1}{\sqrt{N}} \sum_k \hat{c}_k^\dagger e^{-ikR_i}, \\ \hat{c}_i &= \frac{1}{\sqrt{N}} \sum_k \hat{c}_k e^{-ikR_i}. \end{aligned} \quad (2.87)$$

Por fim, o Hamiltoniano nessa base é dado por:

$$H = \sum_k \epsilon_k \hat{c}_k^\dagger \hat{c}_k, \quad (2.88)$$

com $\epsilon_k = \frac{1}{\sqrt{N}} \sum_{ij} \gamma_{ij} e^{-ik(R_i - R_j)}$. Considerando um sistema unidimensional com um único orbital por sítio e o acoplamento apenas entre primeiros vizinhos, obtemos a relação de dispersão no espaço de momentos,

$$\epsilon_k = \epsilon_0 + 2\gamma \cos ka, \quad (2.89)$$

correspondendo à mesma relação apresentada na seção 2.4.1. Aqui, $k \in [-\frac{\pi}{2a}, \frac{\pi}{2a}]$ correspondendo à primeira zona de Brillouin e a é a distância entre os átomos da rede. Considerando ainda a presença de duas projeções de spin por sítio ($\sigma = \uparrow, \downarrow$), mas com uma única partícula por sítio, como consequência, temos uma banda parcialmente preenchida, também conhecida como *half-filled*, pois todos os sítios estão preenchidos com uma única projeção

de spin e $\langle \hat{n}_{\uparrow} \rangle = \langle \hat{n}_{\downarrow} \rangle = \frac{1}{2}$. Temos, nesse caso, um sistema metálico, com o nível de Fermi em ϵ_0 . Nesse limite, a interação pode ser tratada como campo médio, onde $\bar{\epsilon}_i = \epsilon_i + \langle \hat{n}_{i,\sigma} \rangle$.

- Limite de localização: $U/t \gg 1$

Nesse limite, a interação domina o sistema e podemos ignorar o acoplamento entre sítios diferentes.

$$H = \sum_i \epsilon_i (\hat{n}_{i\uparrow} + \hat{n}_{i\downarrow}) + U \sum_i \hat{n}_{i\uparrow} \hat{n}_{i\downarrow} \quad (2.90)$$

Se olharmos para as possíveis ocupações do sistema, temos:

- 0 ocupação: $\langle 0 | \hat{n}_{i,\sigma} | 0 \rangle = 0 \rightarrow E_i = 0$,
- ocupação $|\uparrow\rangle$ ou $|\downarrow\rangle$: $\langle \hat{n}_{i,\sigma} \rangle = 1 \rightarrow E_i = \epsilon_i$,
- ocupação $|\uparrow\downarrow\rangle$ ou $|\downarrow\uparrow\rangle$: $\langle \sigma\bar{\sigma} | \hat{n}_{i,\sigma} + \hat{n}_{i,\bar{\sigma}} | \sigma\bar{\sigma} \rangle = 2 \rightarrow E_i = 2\epsilon_0 + U$.

Supondo que inicialmente temos apenas um elétron por sítio, como estamos no limite de interação forte $U \gg t$, comparativamente o sistema é fracamente acoplado. Então, se quisermos mover um elétron de um sítio para outro com projeções de spin diferentes, estaremos saindo de um estado com ocupação 1 para um duplamente ocupado, e para isso teremos um custo energético de U . Portanto, nesse limite, U faz o papel de acoplamento efetivo entre os sítios. Em temperatura ambiente, o sistema está fracamente acoplado e tem um gap de $E(|\uparrow\downarrow\rangle) - E(|\uparrow\rangle) = \epsilon_0 + U$, tornando-o assim um isolante, conhecido como isolante de Mott [58]. A estrutura de bandas para esse sistema é chamada de bandas de Hubbard [56], pois é decorrente da presença da interação entre elétrons e não do acoplamento entre os sítios.

- Limite transição de fase metal/isolante de Mott: $U \approx t$

Resolver o modelo de Hubbard no limite intermediário, mesmo para o caso onde existe apenas interação entre elétrons no mesmo sítio, é extremamente complicado. Devido ao fato de estarmos interessados particularmente nesse limite, olharemos para outro modelo chamado de Modelo de Anderson para impurezas magnéticas, que em um primeiro momento é diferente e mais simples que o modelo de Hubbard.

Modelo de Anderson para impurezas magnéticas

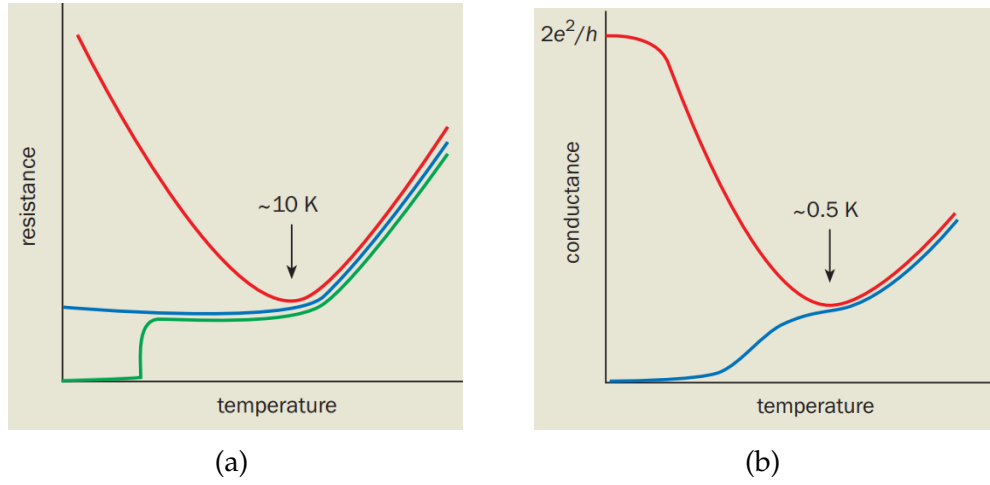


Figura 2.19: **Resistência e Condutância em função da temperatura:** (a) Resistência em função da temperatura para um metal normal (curva em azul), um supercondutor (curva em verde) e um metal com impurezas magnéticas (curva em vermelho). (b) Condutância para um ponto quântico semiconductor, com número de elétrons confinados no ponto sendo ímpar (curva vermelha) e par (curva azul). Observa-se o efeito Kondo: (a) o aumento da resistência em um metal dopado com impurezas magnéticas e (b) o aumento da condutância em um ponto quântico com número ímpar de elétrons confinados. Em ambos os casos, o fenômeno surge para o regime de baixas temperaturas [10]. Figura retira da Ref. [10]

É conhecido que a resistência elétrica para materiais puros aumenta à medida que o sistema é levado a altas temperaturas [10]. Isso ocorre devido a fenômenos de ruído térmico. Por sua vez, à medida que a temperatura diminui, o ruído térmico tende a zero e a resistividade satura após um certo valor. Alguns materiais se tornam supercondutores e a resistência vai a zero (Fig. 2.19(a)). No entanto, foi observado que, para alguns materiais, à medida que a temperatura diminui, a resistividade volta a aumentar após um valor crítico (Fig. 2.19(a)). Devido à ausência de ruído térmico em baixas temperaturas, a explicação deveria ser puramente microscópica. Juan Kondo [59] observou que isso ocorria devido à presença de concentrações de impurezas magnéticas e, usando teoria de perturbação, mostrou que as impurezas agem como centros de espalhamento e adicionam um termo logarítmico na resistividade em função da temperatura., dado por

$$\mathcal{R} = \mathcal{R}_0 + aT^2 + bT^5 + c[1 - \rho(\epsilon_F J)] \ln T. \quad (2.91)$$

Aqui, $\mathcal{R}_0$ é a resistividade característica do material que não depende da tempera-

tura, a é o coeficiente que carrega a contribuição da interação elétron-elétron, b é proveniente da interação elétron-fônon, c é devido à concentração de impurezas magnéticas no sistema, $\rho(\epsilon_F J)$ é a densidade de estados no nível de Fermi e J é o acoplamento efetivo de troca entre o spin da impureza e dos elétrons na banda de condução. Este é o conhecido efeito Kondo [10].

Um modelo que surgiu para explicar como se dá a interação entre metais e estados localizados de impurezas magnéticas usando uma abordagem puramente microscópica é o chamado modelo de Anderson para impurezas magnéticas (AIM, da sigle em inglês).

O Hamiltoniano do AIM multinível é escrito da seguinte maneira,

$$\hat{\mathcal{H}} = \hat{\mathcal{H}}_{\text{imp}} + \hat{\mathcal{H}}_{\text{bath}} + \hat{\mathcal{V}}_{\text{hyb}}, \quad (2.92)$$

com [60]

$$\hat{\mathcal{H}}_{\text{imp}} = \sum_{\alpha, \sigma} \epsilon_{\alpha} \hat{f}_{\alpha, \sigma}^{\dagger} \hat{f}_{\alpha, \sigma} + \frac{1}{2} \sum_{\alpha, \alpha', \beta, \beta', \sigma, \sigma'} U_{\alpha, \alpha', \beta, \beta'} \hat{f}_{\alpha, \sigma}^{\dagger} \hat{f}_{\alpha', \sigma'}^{\dagger} \hat{f}_{\beta', \sigma'} \hat{f}_{\beta, \sigma}, \quad (2.93)$$

$$\hat{\mathcal{H}}_{\text{bath}} = \sum_{k, \nu, \sigma} \epsilon_{k, \nu} \hat{c}_{k, \nu, \sigma}^{\dagger} \hat{c}_{k, \nu, \sigma}, \quad (2.94)$$

$$\hat{\mathcal{V}}_{\text{hyb}} = \sum_{\alpha, k, \nu, \sigma} \hat{\gamma}_{\alpha, k, \nu, \sigma} \left(\hat{f}_{\alpha, \sigma}^{\dagger} \hat{c}_{k, \nu, \sigma} + \hat{c}_{k, \nu, \sigma}^{\dagger} \hat{f}_{\alpha, \sigma} \right). \quad (2.95)$$

Ele consiste então em um Hamiltoniano dividido em duas regiões. A primeira é $\hat{\mathcal{H}}_{\text{bath}}$, contendo o que chamamos de banho, pois temos elétrons não interagentes em equilíbrio térmico. A segunda região, $\hat{\mathcal{H}}_{\text{imp}}$, é a impureza que pode conter múltiplos níveis internos, ou seja, com uma estrutura análoga a uma molécula com diversos orbitais moleculares, com $\hat{\mathcal{V}}$ sendo o acoplamento entre as regiões. Os níveis podem estar semi-preenchidos ou completamente preenchidos, e assim todos os elétrons desta região tendem a interagir entre si (Fig. 2.20(a)). Olhando então apenas para o termo de interação elétron-elétron entre os níveis, e considerando que é de natureza Coulombiana [60, 61], temos:

$$\frac{1}{2} \sum_{\alpha, \alpha', \beta, \beta', \sigma, \sigma'} U_{\alpha, \alpha', \beta, \beta'} \hat{f}_{\alpha, \sigma}^{\dagger} \hat{f}_{\alpha', \sigma'}^{\dagger} \hat{f}_{\beta', \sigma'} \hat{f}_{\beta, \sigma} \quad (2.96)$$

$$\approx U \sum_{\alpha} n_{\alpha \uparrow} n_{\alpha \downarrow} + U' \sum_{\alpha \neq \beta} n_{\alpha} n_{\beta} - J_H S_f^2 + J_H (N_f - D - N_f^2/4),$$

com $n_{\alpha \sigma} = \hat{f}_{\alpha, \sigma}^{\dagger} \hat{f}_{\alpha, \sigma}$ como o operador número do nível α com projeção de spin σ ,

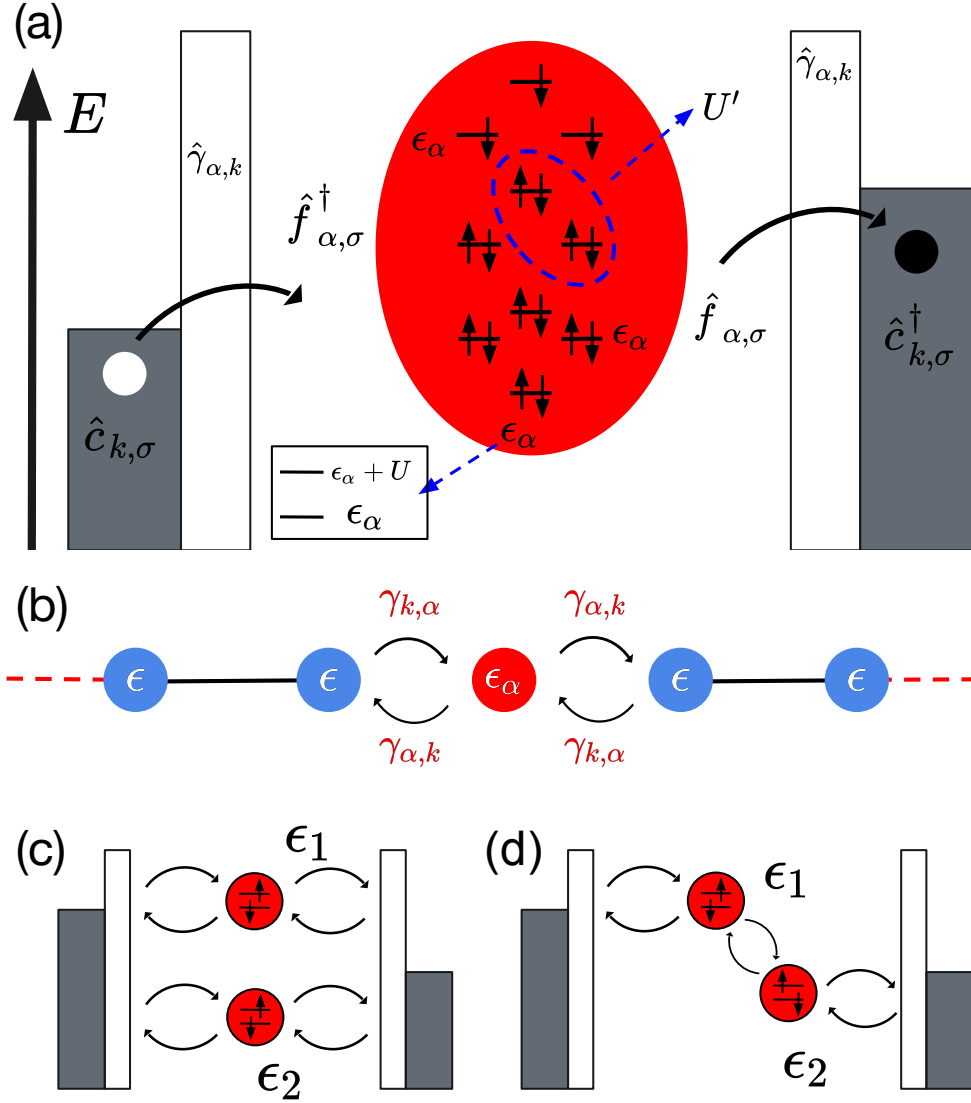


Figura 2.20: **Visão esquemática do Modelo de Anderson para impurezas:** As regiões em cinza representam os banhos de elétrons, enquanto as regiões em vermelho representam a região de impurezas e as regiões em branco representam as barreiras de potencial. a) Descrição contendo múltiplos níveis internos, com níveis semi-preenchidos e completamente preenchidos, onde $\hat{c}_{k,\nu,\sigma}^\dagger$ ($\hat{c}_{k,\nu,\sigma}$) é o operador de criação (aniquilação) dos elétrons no banho, $\hat{f}_{k,\nu,\sigma}^\dagger$ ($\hat{f}_{k,\nu,\sigma}$) o operador de criação (aniquilação) dos elétrons na impureza, e $\hat{\gamma}_{k,\alpha}$ é o acoplamento entre as regiões. ϵ_α é a energia do nível α , U é o potencial de interação entre elétrons no mesmo nível, U' o termo de interação entre elétrons em estados diferentes. b) Descrição do caso de um único nível, c) e d) descrição para o caso de dois níveis.

$n_\alpha = n_{\alpha\uparrow} + n_{\alpha\downarrow}$ sendo a soma entre o operador número com diferentes projeções de spin σ no mesmo nível, $U = U_{\alpha\alpha\alpha\alpha}$ o potencial de interação entre elétrons no mesmo nível, $U' = U_{\alpha\beta\alpha\beta}$ e $J_H = U_{\alpha\beta\beta\alpha}$ os termos de interação entre elétrons

em estados diferentes $\alpha \neq \beta$. Nota-se que U' media a interação que resulta em elétrons com a mesma projeção de spin inicial, enquanto J_H altera a projeção dos spins. Portanto, J_H é o acoplamento da regra de Hund, $\hat{N}_f = \sum_i \hat{n}_{i,f}$ é o número total de elétrons, D é o número de estados duplamente ocupados e S_f o spin total sobre todos os estados interagentes, como ilustrado na Fig. 2.20(a).

Alguns casos particulares que valem a pena serem comentados são o caso de um único nível (SAIM, da sigla em inglês) e para dois níveis (DAIM, da sigla em inglês). Para o SAIM, automaticamente não há interações entre elétrons em níveis diferentes, então $U' = J_H = 0$, e o sistema se torna mais simples (ver Fig. 2.20(b)).

$$\hat{\mathcal{H}}_{\text{Simp}} = \epsilon_f \hat{n}_{\hat{f}} + U \hat{n}_{f\uparrow} \hat{n}_{f\downarrow}. \quad (2.97)$$

Para DAIM, além dos dois níveis, podemos ter interações entre eles, ou seja, a priori $U' \neq 0$ e $J_H = 0$ (ver Fig. 2.20(c) e Fig. 2.20(d))

$$\begin{aligned} \hat{\mathcal{H}}_{\text{Dimp}} = & \epsilon_{f,1} \hat{n}_{f,1} + \epsilon_{f,2} \hat{n}_{f,2} + U(\hat{n}_{f,1\uparrow} \hat{n}_{f,1\downarrow} + \hat{n}_{f,2\uparrow} \hat{n}_{f,2\downarrow}) \\ & + U'[(\hat{n}_{f,1\uparrow} + \hat{n}_{f,1\downarrow}) + (\hat{n}_{f,2\uparrow} + \hat{n}_{f,2\downarrow})]. \end{aligned} \quad (2.98)$$

A princípio, temos os mesmos limites apresentados para o modelo de Hubbard anteriormente em 2.6, mas com um número finito (tipicamente 1) de impurezas.

2.6.2 Representação de pseudo partícula

Além dos problemas na resolução do modelo de Anderson para o regime $U \approx t$, tipicamente, a interação de Coulomb efetiva, $U_{\alpha,\alpha',\beta,\beta'}$, entre os elétrons na impureza é grande em comparação com o acoplamento entre os elétrons da impureza e o banho, $|\hat{\mathcal{V}}_{\text{hyb}}|_{\alpha,k,\nu,\sigma} = \hat{\gamma}_{\alpha,k,\nu,\sigma}$, de modo que utilizar teoria de perturbação no termo de interação é um problema. Uma alternativa é escrever o sistema em uma descrição de pseudo-partículas [62, 63] e utilizar teoria de perturbação no termo de hibridização $|\hat{\mathcal{V}}_{\text{hyb}}|$. Podemos assim desenvolver uma expansão diagramática, e consideraremos apenas os diagramas sem cruzamentos (NCA, da sigla em inglês) [62, 63], que é uma boa aproximação para temperaturas acima da temperatura de Kondo, T_K , e para o regime $U \approx t$ [64]. Isso resulta em uma boa descrição da transição de fase metal/isolante de Mott. Finalmente, a título de informação, os diagramas com um único cruzamento (OCA, da sigla em inglês) [62, 63] apresentam uma melhor descrição para temperaturas abaixo de T_K [64].

Inicialmente, a estratégia é diagonalizar $\hat{\mathcal{H}}_{\text{imp}}$ dado pela Eq. 2.93. Para isso, es-

crevemos o Hamiltoniano das impurezas em termos de pseudo-partículas, através da seguinte relação com os operadores de criação (aniquilação) $\hat{f}^\dagger$ ($\hat{f}$),

$$\hat{f}_\sigma^\dagger = \hat{s}_\sigma^\dagger \hat{b} + \hat{d}_{\sigma\bar{\sigma}}^\dagger \hat{s}_\sigma, \quad (2.99)$$

com $\hat{b}_\sigma^\dagger$, $\hat{s}_\sigma^\dagger$ e $\hat{d}_{\sigma\bar{\sigma}}^\dagger$ operadores de criação de pseudo-partículas (com seus respectivos operadores de aniquilação e $\sigma = \uparrow$ ou $\downarrow$), atuando no vacuo dessa representação da seguinte maneira

$$\hat{b}_\sigma^\dagger |0_{PP}\rangle = |0\rangle, \quad \hat{s}_\sigma^\dagger |0_{PP}\rangle = |\sigma\rangle \quad \text{e} \quad \hat{d}_{\sigma\bar{\sigma}}^\dagger |0_{PP}\rangle = |\sigma\bar{\sigma}\rangle. \quad (2.100)$$

Assim, representando estados associado a ocupação zero, um e dupla ocupação respectivamente. Aqui $\hat{b}_\sigma^\dagger$ e $\hat{d}_{\sigma\bar{\sigma}}^\dagger$ operadores seguem as regras de comutação para bósons, e $\hat{s}_\sigma^\dagger$ as regras de comutação para férmios. Impondo as condições, para eliminar estados não físicos, obtemos a seguinte relação,

$$\hat{f}_\sigma^\dagger \hat{f}_\sigma = \hat{s}_\sigma^\dagger \hat{s}_\sigma + \hat{d}_{\sigma\bar{\sigma}}^\dagger \hat{d}_{\sigma\bar{\sigma}}, \quad (2.101)$$

e a conservação de carga através da contração

$$Q = \hat{b}^\dagger \hat{b} + \sum_{\sigma} \hat{s}_\sigma^\dagger \hat{s}_\sigma + \hat{d}_{\sigma\bar{\sigma}}^\dagger \hat{d}_{\sigma\bar{\sigma}} = 1. \quad (2.102)$$

Para um modelo de Anderson multinível é conveniente escrever em uma notação compacta os operadores de pseudo-partícula, como $\hat{a}_p$, com p indicando o número de ocupação. Assim, $\hat{a}_p^\dagger = \hat{b}_\sigma^\dagger$ ou $\hat{d}_{\sigma\bar{\sigma}}^\dagger$ e $\hat{a}_p = \hat{s}_\sigma$ para p ímpar e par receptivamente. Consequentemente,

$$\hat{a}_p^\dagger |0_{PP}\rangle = |p\rangle \quad \text{e} \quad \hat{a}_p |p\rangle = |0_{PP}\rangle, \quad (2.103)$$

relacionado, de maneira geral, com $\hat{f}$ por

$$\hat{f}_{\alpha\sigma} = \sum_{p,p'} (F^{\alpha\sigma})_{pp'} \hat{a}_p^\dagger \hat{a}_{p'}, \quad (2.104)$$

com $(F^{\alpha\sigma})_{pp'} = \langle p | \hat{f}_{\alpha\sigma} | p' \rangle$, que são os elementos da matriz do operador de criação de impureza-elétrons. Desta forma, obtemos um Hamiltoniano para as impurezas

de Anderson desacoplado do banho de elétrons e diagonal,

$$\hat{\mathcal{H}}_{\text{imp}} = \sum_p e_p \hat{a}_p^\dagger \hat{a}_p. \quad (2.105)$$

O Hamiltoniano completo é dado pela transformação da Eq. 2.92,

$$\hat{\mathcal{H}} = \sum_p e_p \hat{a}_p^\dagger \hat{a}_p + \sum_{k,\nu,\sigma} \epsilon_{k,\nu} \hat{c}_{k,\nu,\sigma}^\dagger \hat{c}_{k,\nu,\sigma} + \sum_{pp',k,\nu,\sigma} \left(\hat{\gamma}_{\alpha,k,\nu,\sigma} (F^{\alpha\sigma})_{pp'} \hat{c}_{k,\nu,\sigma}^\dagger \hat{a}_p^\dagger \hat{a}_{p'} + \text{h.c.} \right). \quad (2.106)$$

Aqui, a hibridização com os elétrons do banho dada pelo último termo passa a ser um termo análogo à interação para as Pseudo-partículas. Devido às regras de comutação fermiônica e bosônica, pode-se desenvolver uma expansão em termos de uma perturbação diagramática na interação das Pseudo-partículas. Em particular, o SAIM (2.97), na descrição de pseudo-partícula, assume a seguinte forma, em termos dos operadores de pseudo-partícula,

$$\begin{aligned} \hat{\mathcal{H}} = & \sum_{\sigma} e_0 \hat{s}_{\sigma}^\dagger \hat{s}_{\sigma} + (2e_0 + U) \hat{d}_{\sigma\bar{\sigma}}^\dagger \hat{d}_{\sigma\bar{\sigma}} + \sum_{k,\nu,\sigma} \epsilon_{k,\nu} \hat{c}_{k,\nu,\sigma}^\dagger \hat{c}_{k,\nu,\sigma} \\ & + \sum_{k,\nu,\sigma} \left(\hat{\gamma}_{\alpha,k,\nu,\sigma} \hat{c}_{k,\nu,\sigma}^\dagger \hat{s}_{\sigma}^\dagger \hat{b} + \text{h.c.} \right) + \left(\hat{\gamma}_{\alpha,k,\nu,\sigma} \hat{c}_{k,\nu,\sigma}^\dagger \hat{d}_{\sigma\bar{\sigma}}^\dagger \hat{s}_{\bar{\sigma}} + \text{h.c.} \right), \end{aligned} \quad (2.107)$$

com, $\hat{\mathcal{H}}_{\text{imp}}$ bilinear. Nessa representação, o termo de hibridização $\hat{\mathcal{V}}_{\text{hyb}}$, se torna mais complicado. A carga da PP (Q) é restrita a um, ou seja, só podemos ter uma PP por vez. Adicionamos a restrição Lagrangiana $\lambda(Q - 1)$ ao Hamiltoniano onde o multiplicador de Lagrange λ desempenha o papel de um potencial químico para a PP (negativo). O valor esperado de um observável físico A é calculado no Ensemble grand-canônico e é então projetado no subespaço físico com $Q = 1$ através do truque de Abrikosov [62]:

$$\langle A \rangle = \lim_{\lambda \rightarrow \infty} \frac{\langle QA \rangle_G(\lambda)}{\langle Q \rangle_G(\lambda)}. \quad (2.108)$$

O fator Q no numerador permite remover o subespaço (não físico) $Q = 0$. No entanto, para todos os observáveis locais físicos A (atuando no subespaço de impurezas) o valor esperado de A no subespaço $Q = 0$ é sempre zero. Portanto, basta calcular

$$\langle A \rangle = \lim_{\lambda \rightarrow \infty} \frac{\langle A \rangle_G(\lambda_0)}{\langle Q \rangle_G(\lambda_0)}. \quad (2.109)$$

A função de Green é dada por,

$$G_p(E) = \lim_{\eta \rightarrow 0^+, \lambda \rightarrow \infty} [(E + i\eta) - e_p - \lambda - \Sigma_p(E)]^{-1}, \quad (2.110)$$

onde Σ_p é a auto-energia da PP que descreve a interação dinâmica com as demais PPs induzida pelo acoplamento com o banho de elétrons. Com as auto-energias obedecendo às seguintes equações autoconsistentes [65]

$$\Sigma_b(E) = \int_{-\infty}^{\infty} \frac{d\epsilon}{\pi} f(\epsilon) \sum_{\sigma} \Delta_{\sigma}(\epsilon) G_{s_{\sigma}}(\epsilon + E) \Lambda_{\sigma}^{(0)}(E, \epsilon), \quad (2.111)$$

$$\Sigma_{s_{\sigma}}(E) = \int_{-\infty}^{\infty} \frac{d\epsilon}{\pi} f(\epsilon) [\Delta_{\sigma}(-\epsilon) G_b(\epsilon + E) \Lambda_{\sigma}^{(0)}(\epsilon + E, -\epsilon) + \Delta_{\sigma\bar{\sigma}}(\epsilon) G_{d_{\sigma\bar{\sigma}}}(\epsilon + E) \Lambda_{\sigma\bar{\sigma}}^{(2)}(\epsilon + E, \epsilon)], \quad (2.112)$$

$$\Sigma_{d_{\sigma\bar{\sigma}}}(E) = \int_{-\infty}^{\infty} \frac{d\epsilon}{\pi} f(\epsilon) [\Delta_{\sigma}(-\epsilon) G_{s_{\bar{\sigma}}}(\epsilon + E) \Lambda_{\sigma\bar{\sigma}}^{(0)}(E, -\epsilon) + \Delta_{\bar{\sigma}}(-\epsilon) G_{s_{\sigma}}(\epsilon + E) \Lambda_{\sigma\bar{\sigma}}^{(2)}(E, -\epsilon)]. \quad (2.113)$$

Aqui, $\Delta_{\sigma}(\epsilon) = \pi \sum_{\alpha, k, \nu} |\hat{\gamma}_{\alpha, k, \nu, \sigma}|$ é a função de hibridização, que essencialmente é a função espectral do acoplamento impureza-banho. As funções de Green das PPs em termos das suas respectivas auto-energias é dada explicitamente por

$$G_b(E) = \lim_{\eta \rightarrow 0^+, \lambda \rightarrow \infty} [(E + i\eta) - \lambda - \Sigma_b(E)]^{-1}, \quad (2.114)$$

$$G_{s_{\sigma}}(E) = \lim_{\eta \rightarrow 0^+, \lambda \rightarrow \infty} [(E + i\eta) - \lambda - e_0 - \Sigma_{s_{\sigma}}(E)]^{-1}, \quad (2.115)$$

$$G_{d_{\sigma\bar{\sigma}}}(E) = \lim_{\eta \rightarrow 0^+, \lambda \rightarrow \infty} [(E + i\eta) - \lambda - 2e_0 - U - \Sigma_{\sigma\bar{\sigma}}(E)]^{-1}. \quad (2.116)$$

Ainda, as funções Λ contem as correções dadas pelos vértices nos diagramas,

$$\Lambda_{\sigma}^0(E, E') = 1 + \int_{-\infty}^{\infty} \frac{d\epsilon}{\pi} f(\epsilon) \Delta_{\sigma}(\epsilon) G_{s_{\sigma}}(E + \epsilon) G_{d_{\sigma\bar{\sigma}}}(E + E' + \epsilon) \quad (2.117)$$

$$\Lambda_{\sigma}^2(E, E') = 1 + \int_{-\infty}^{\infty} \frac{d\epsilon}{\pi} f(-\epsilon) \Delta_{\sigma}(\epsilon) G_{s_{\sigma}}(E - \epsilon) G_{d_{\sigma\bar{\sigma}}}(E - E' - \epsilon). \quad (2.118)$$

Descartando as correções de vértices, que correspondem ao conjunto $\Lambda_{\sigma\bar{\sigma}}^2(E, E') = \Lambda_{\sigma}^0(E, E') = 1$, obtemos a chamada aproximação não cruzada (NCA) [62]. Se incluirmos as correções de vértices chegaremos à aproximação de cruzamento único (OCA) [62]. Dada a solução, obtemos então uma auto-energia com a informação da interação e escrevendo o sistema na representação inicial, obtemos por fim a

auto-energia de interação elétron-elétron, através da relação,

$$\Sigma_{ee} = G^{-1} - G_0^{-1}. \quad (2.119)$$

2.7 Projecção sobre estados localizados

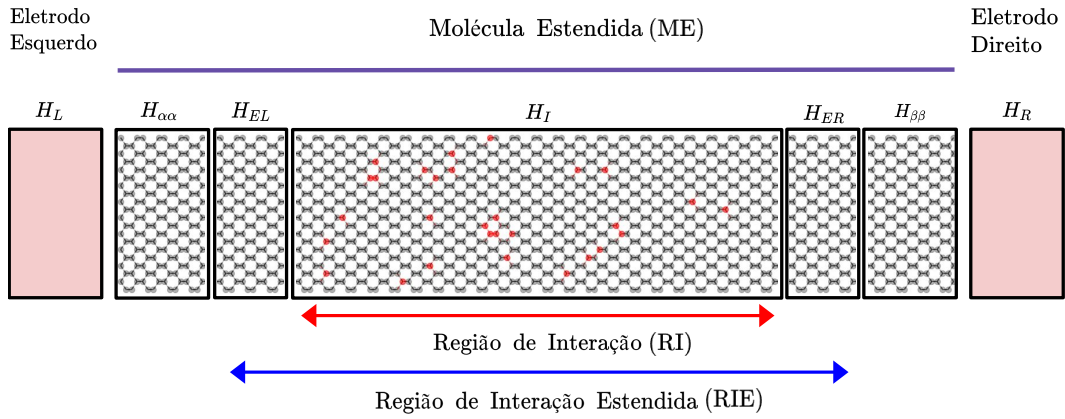


Figura 2.21: **Visão esquemática de um problema de transporte eletrônico na estrutura de uma molécula estendida:** As regiões em vermelho representam os eletrodos, com $\hat{H}_L$ e $\hat{H}_R$ sendo os Hamiltonianos dos eletrodos esquerdo e direito, respectivamente. A região de espalhamento é dividida em sub-regiões, com $\hat{H}_I$ representando a região de interação, $\hat{H}_{EL/(ER)}$ representando a região à esquerda (direita) da região de interação. Todas essas regiões, juntamente com $\hat{H}_{\alpha\alpha/(\beta\beta)}$, compõem o que chamamos de região de molécula estendida.

De maneira geral consideramos, um sistema parcialmente dopado com impurezas, sendo aqui, modelado como um orbital (ou múltiplos orbitais) atômico com energia diferente dos sítios da rede. Por um lado, a interação e-e deve considerar, em princípio, todos os sítios do sistema. Por outro lado, impurezas levam a estados localizados onde a interação e-e é mais importante e, portanto, podem ser tratadas como Impurezas de Anderson (IA). É conveniente tratar o problema na base dessas “impurezas” que, essencialmente, são estados localizados. É interessante notar que vacâncias também pode ser tratadas desta forma, levando a energia do sítio para infinito ou descrevendo os estados moleculares removendo um orbital atômico.

Nesta seção, descreveremos o formalismo para a projeção sobre estados locali-

zados em um conjunto de orbitais moleculares interagentes, utilizando a linguagem de funções de Green fora do equilíbrio [66]. Faremos isso para mapear o problema que envolve múltiplos orbitais em outro contendo um número reduzido, sendo esses os orbitais de interesse.

Inicialmente, como ilustrado na Fig. 2.21, o Hamiltoniano da região de espalhamento tem a seguinte estrutura:

$$\hat{H} = \hat{H}_{\alpha\alpha} + \hat{H}_{EL} + \hat{H}_I + \hat{H}_{ER} + \hat{H}_{\beta\beta}. \quad (2.120)$$

Conforme esquematicamente indicado na Fig. 2.21, a região de espalhamento é subdividida em um total de 4 subsistemas [66]: *i*) Molécula Estendida (ME), contendo todos os orbitais da região de espalhamento, com uma base contendo N orbitais; *ii*) Região de Interação Estendida (RIE), contendo o subconjunto de orbitais pertencentes à região de interação ($\hat{H}_I$) e às regiões à esquerda ($\hat{H}_{EL}$) e à direita ($\hat{H}_{ER}$) da região de interação, com NRI orbitais; *iii*) A Região de Interação (RI) ($\hat{H}_I$), que contém um subconjunto de orbitais, onde ocorrem todos os processos de interação, e contendo NI orbitais; e por fim, *iv*) As Impurezas de Anderson (AI, da sigla em inglês), um subconjunto da RI, contendo uma pequena quantidade de orbitais interagentes, onde um conjunto de funções de onda ψ_i ($i \in [1, N_{AI}]$) define as AI interagentes, com N_{AI} igual ao número de estados interagentes ($N_{AI} < N$). Ordenados por tamanho decrescente, os 4 subsistemas são então: ME, RIE, RI, AI (onde a AI é um subsistema da RI, o RI é um subsistema da RIE, e o RIE é um subsistema da ME; veja a Fig. 2.21).

O Hamiltoniano da ME é expandido sobre os orbitais de base $|\phi_\nu\rangle$,

$$\mathcal{H}_{\mu\nu} = \langle \phi_\mu | \hat{\mathcal{H}} | \phi_\nu \rangle, \quad (2.121)$$

os quais podem ser não-ortogonais. Em geral, a matriz de overlap é dada por

$$\mathcal{S}_{\mu\nu} = \langle \phi_\mu | \phi_\nu \rangle, \quad (2.122)$$

de modo que, as matrizes $\mathcal{H}$ e $\mathcal{S}$ têm dimensões $N \times N$.

O primeiro passo no esquema de projeção é definir o conjunto de orbitais moleculares, ψ_i , dentro da ME que constituem o AI. Estes são, em princípio, arbitrários e podem basear-se na intuição física. No caso mais simples, por exemplo, pode-se simplesmente tomar um conjunto de orbitais d ou f de uma impureza magnética, ou então as funções de onda *HOMO* ou *LUMO* de uma molécula

ligada a eletrodos metálicos. O método descrito aqui funcionará para ambos os casos, e consideraremos uma situação genérica de orbitais moleculares como uma combinação de orbitais atômicos qualquer.

Matriz de projeção direta

Uma maneira prática de extrair um conjunto de orbitais moleculares é extrair das matrizes $\mathcal{H}$ e $\mathcal{S}$, apenas os sub-blocos que denotamos como $\mathcal{H}_I$ e $\mathcal{S}_I$, e calcular os autovalores ϵ_i e os autovetores ψ_i de tais sub-blocos, resolvendo $\mathcal{H}_I \psi_i = \epsilon_i \mathcal{S}_I \psi_i$. Observe que cada um dos ψ_i é um vetor de dimensão N_{RI} . Escolhemos então o conjunto que compõe a AI: $\{\psi_1, \psi_2, \dots, \psi_{AI}\}$, que definem a matriz básica de projeção

$$U_\Psi = (\psi_1, \psi_2, \dots, \psi_{AI}), \quad (2.123)$$

com dimensão $N_{RI} \times N_{AI}$. Os orbitais de base do AI, portanto, abrangem os orbitais da molécula, que neste caso corresponde a RI. Nós então construímos a RIE adicionando a ele todos os orbitais dentro da ME que tenham um overlap finito com a RI. Em uma configuração típica de transporte de dois terminais, subdividimos a matriz de overlap da ME como

$$\mathcal{S} = \begin{pmatrix} \mathcal{S}_{\alpha\alpha} & \mathcal{S}_{\alpha E} & 0 & 0 & 0 \\ \mathcal{S}_{\alpha E}^\dagger & \mathcal{S}_{EL} & \mathcal{S}_{EL,I} & \mathcal{S}_{EL,ER} & 0 \\ 0 & \mathcal{S}_{EL,I}^\dagger & \mathcal{S}_I & \mathcal{S}_{I,ER} & 0 \\ 0 & \mathcal{S}_{EL,ER}^\dagger & \mathcal{S}_{I,ER}^\dagger & \mathcal{S}_{ER} & \mathcal{S}_{E\beta} \\ 0 & 0 & 0 & \mathcal{S}_{E\beta}^\dagger & \mathcal{S}_{\beta\beta} \end{pmatrix}. \quad (2.124)$$

onde $\mathcal{S}_I$ é uma matriz $N_{RI} \times N_{RI}$, $\mathcal{S}_{EL}$ inclui todos os orbitais N_{EL} da RIE à esquerda do RI (e é, portanto, uma matriz $N_{EL} \times N_{EL}$), e $\mathcal{S}_{ER}$ inclui todos os orbitais N_{ER} da RIE à direita da RI (e é, portanto, uma matriz $N_{ER} \times N_{ER}$). Observe que $N_E = N_{EL} + N_{ER} + N_{RI}$. $\mathcal{S}_{\alpha\alpha}$ ($\mathcal{S}_{\beta\beta}$) inclui todos os orbitais N_α (N_β) à esquerda (direita) do RIE, de modo que $N = N_\alpha + N_\beta + N_E$. As dimensões dos blocos de matriz fora da diagonal, bem como dos blocos de matriz iguais a zero, são determinadas a partir das dimensões dos blocos diagonais e, portanto, não são

fornecidas explicitamente. Observe que aqui definimos o sub-bloco de overlap $\mathcal{S}_{\alpha,ER} = \mathcal{S}_{EL,\beta} = 0$. Este é geralmente o caso em configurações práticas, no entanto, o método de projeção geral também é válido se esses sub-blocos da matriz de overlap forem diferentes de zero. No entanto, exigimos que $\mathcal{S}_{\alpha\beta} = 0$. Observe também que por construção $\mathcal{S}_{\alpha,I}$ e $\mathcal{S}_{I,\beta}$ são 0. Em uma configuração prática de transporte, a célula de simulação (a ME) inclui uma célula unitária dos terminais em cada extremidade, e é importante que o RIE não inclua esses orbitais para que o overlap entre o AI e as células unitárias dos terminais em ME seja 0. O Hamiltoniano do problema pode escrito de maneira análoga.

Introduzimos então um novo conjunto da base $\{|\tilde{\phi}_\mu\rangle\}$, que é idêntico ao conjunto original de $\{|\phi_\mu\rangle\}$, com exceção dos orbitais de base N_{AI} , que possuem um grande overlap com o conjunto $\{\psi_1, \psi_2, \dots, \psi_{AI}\}$, que são substituídos pelo ψ_i correspondente do AI. Observe que, por construção, os orbitais de base substituídos sempre fazem parte do RI (uma vez que ψ_i possui elementos diferentes de zero apenas nos orbitais da base da RI). Esta transformação é alcançada com a matriz de projeção

$$W_0 = \begin{pmatrix} 0 & 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ U_\Psi & 0 & 0 & U_{NI} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}. \quad (2.125)$$

Aqui e a seguir denotamos uma matriz identidade de dimensão m como $1_{m \times m} = 1_m$. As dimensões da matriz U_Ψ são $N_{RI} \times N_{AI}$, enquanto a matriz U_{NI} tem dimensão $N_{RI} \times N_{NI}$, e é uma matriz identidade com colunas N_{AI} removidas conforme descrito acima. Coletamos todos os vetores unitários removidos ao formar U_{NI} e introduzimos em uma matriz $U_\perp$ com dimensões $N_{RI} \times N_{AI}$ com a seguinte relação $U_\perp^\dagger U_{NI} = 0$.

As matrizes de overlap e Hamiltonianas na nova base, $\tilde{\mathcal{S}}$ e $\tilde{\mathcal{H}}$, são então obtidas

$$\tilde{\mathcal{S}} = W_0^\dagger \mathcal{S} W_0, \quad (2.126)$$

$$\tilde{\mathcal{H}} = W_0^\dagger \mathcal{H} W_0. \quad (2.127)$$

A estrutura geral de $\tilde{\mathcal{S}}$ é

$$\tilde{\mathcal{S}} = \begin{pmatrix} \tilde{\mathcal{S}}_{AI} & 0 & \tilde{\mathcal{S}}_{AI,EL} & \tilde{\mathcal{S}}_{AI,NI} & \tilde{\mathcal{S}}_{AI,ER} & 0 \\ 0 & \mathcal{S}_{\alpha\alpha} & \mathcal{S}_{\alpha E} & 0 & 0 & 0 \\ \tilde{\mathcal{S}}_{AI,EL}^\dagger & \mathcal{S}_{\alpha E}^\dagger & \mathcal{S}_{EL} & \tilde{\mathcal{S}}_{NI,EL}^\dagger & \mathcal{S}_{EL,ER} & 0 \\ \tilde{\mathcal{S}}_{AI,NI}^\dagger & 0 & \tilde{\mathcal{S}}_{NI,EL} & \tilde{\mathcal{S}}_{NI} & \tilde{\mathcal{S}}_{NI,ER} & 0 \\ \tilde{\mathcal{S}}_{AI,ER}^\dagger & 0 & \mathcal{S}_{EL,ER}^\dagger & \tilde{\mathcal{S}}_{NI,ER}^\dagger & \mathcal{S}_{ER} & \mathcal{S}_{E\beta} \\ 0 & 0 & 0 & 0 & \mathcal{S}_{E\beta}^\dagger & \mathcal{S}_{\beta\beta} \end{pmatrix}, \quad (2.128)$$

onde $\tilde{\mathcal{S}}_{AI}$ tem dimensão $N_{AI} \times N_{AI}$, e $\tilde{\mathcal{S}}_{NI}$ com dimensão $N_{NI} \times N_{NI}$. Os demais sub-blocos possuem dimensões idênticas às suas contrapartes no caso não transformado. Os valores explícitos dos sub-blocos são os seguintes

$$\tilde{\mathcal{S}} = \begin{pmatrix} U_\Psi^\dagger \mathcal{S}_I U_\Psi & 0 & U_\Psi^\dagger \mathcal{S}_{I,EL} & U_\Psi^\dagger \mathcal{S}_I U_{NI} & U_\Psi^\dagger \mathcal{S}_{I,ER} & 0 \\ 0 & \mathcal{S}_{\alpha\alpha} & \mathcal{S}_{\alpha E} & 0 & 0 & 0 \\ \mathcal{S}_{I,EL}^\dagger U_\Psi & \mathcal{S}_{\alpha E}^\dagger & \mathcal{S}_{EL} & \mathcal{S}_{I,EL}^\dagger U_{NI} & \mathcal{S}_{EL,ER} & 0 \\ U_{NI}^\dagger \mathcal{S}_I^\dagger U_\Psi & 0 & U_{NI}^\dagger \mathcal{S}_{I,EL} & U_{NI}^\dagger \mathcal{S}_I U_{NI} & U_{NI}^\dagger \mathcal{S}_{I,ER} & 0 \\ \mathcal{S}_{I,ER}^\dagger U_\Psi & 0 & \mathcal{S}_{EL,ER}^\dagger & \mathcal{S}_{I,ER}^\dagger U_{NI} & \mathcal{S}_{ER} & \mathcal{S}_{E\beta} \\ 0 & 0 & 0 & 0 & \mathcal{S}_{E\beta}^\dagger & \mathcal{S}_{\beta\beta} \end{pmatrix}, \quad (2.129)$$

com $\tilde{\mathcal{H}}$ tendo uma estrutura análoga. As matrizes $\tilde{\mathcal{S}}_{AI}$ e $\tilde{\mathcal{H}}_{AI}$ são as matrizes de overlap e Hamiltoniana da AI, respectivamente. Por construção, os orbitais de

base AI têm overlap 0 com a célula unitária dos contatos incluída em ambos os lados da ME. Entretanto, as funções de onda da base da AI não são ortogonais aos orbitais da RIE. Se o solucionador de impurezas de Anderson puder lidar com tais sistemas não ortogonais, então será possível passar esta matriz diretamente para tal solucionador. A maioria dos *solvers* de IA, entretanto, exigem ortogonalidade com o banho de elétrons, de modo que realizamos uma segunda transformação que ortogonaliza os orbitais do banho da RIE aos da IA. É importante ressaltar que queremos que os orbitais da AI permaneçam inalterados, de modo que apenas os orbitais das regiões EL, NI e ER sejam modificados. Podemos conseguir isso com a seguinte matriz de transformação,

$$W_1 = \begin{pmatrix} 1_{N_{AI}} & 0 & -W_A & -W_B & -W_C & 0 \\ 0 & 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1_{N_{NI}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}, \quad (2.130)$$

com

$$\begin{aligned} W_A &= \tilde{\mathcal{S}}_{AI}^{-1} \tilde{\mathcal{S}}_{AI,EL}, \\ W_B &= \tilde{\mathcal{S}}_{AI}^{-1} \tilde{\mathcal{S}}_{AI,NI}, \\ W_C &= \tilde{\mathcal{S}}_{AI}^{-1} \tilde{\mathcal{S}}_{AI,ER}. \end{aligned} \quad (2.131)$$

Isso leva ao Hamiltoniano transformado $\tilde{\mathcal{H}} = W_1^\dagger \mathcal{H} W_1$ e a matriz de overlap

$\tilde{\tilde{S}} = W_1^\dagger \tilde{S} W_1$, com a nova estrutura

$$\tilde{\tilde{S}} = \begin{pmatrix} \tilde{S}_{AI} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathcal{S}_{\alpha\alpha} & \mathcal{S}_{\alpha E} & 0 & 0 & 0 \\ 0 & \mathcal{S}_{\alpha E}^\dagger & \tilde{S}_{EL} & \tilde{S}_{NI,EL}^\dagger & \tilde{S}_{EL,ER} & 0 \\ 0 & 0 & \tilde{S}_{NI,EL} & \tilde{S}_{NI} & \tilde{S}_{NI,ER} & 0 \\ 0 & 0 & \tilde{S}_{EL,ER}^\dagger & \tilde{S}_{NI,ER}^\dagger & \tilde{S}_{ER} & \mathcal{S}_{E\beta} \\ 0 & 0 & 0 & 0 & \mathcal{S}_{E\beta}^\dagger & \mathcal{S}_{\beta\beta} \end{pmatrix}, \quad (2.132)$$

$$\tilde{\tilde{\mathcal{H}}} = \begin{pmatrix} \tilde{\mathcal{H}}_{AI} & 0 & \tilde{\mathcal{H}}_{AI,EL} & \tilde{\mathcal{H}}_{AI,NI} & \tilde{\mathcal{H}}_{AI,ER} & 0 \\ 0 & \mathcal{H}_{\alpha\alpha} & \mathcal{H}_{\alpha E} & 0 & 0 & 0 \\ \tilde{\mathcal{H}}_{AI,EL}^\dagger & \mathcal{H}_{\alpha E}^\dagger & \tilde{\mathcal{H}}_{EL} & \tilde{\mathcal{H}}_{NI,EL}^\dagger & \tilde{\mathcal{H}}_{EL,ER} & 0 \\ \tilde{\mathcal{H}}_{AI,NI}^\dagger & 0 & \tilde{\mathcal{H}}_{NI,EL} & \tilde{\mathcal{H}}_{NI} & \tilde{\mathcal{H}}_{NI,ER} & 0 \\ \tilde{\mathcal{H}}_{AI,ER}^\dagger & 0 & \tilde{\mathcal{H}}_{EL,ER}^\dagger & \tilde{\mathcal{H}}_{NI,ER}^\dagger & \tilde{\mathcal{H}}_{ER} & \mathcal{H}_{\beta E}^\dagger \\ 0 & 0 & 0 & 0 & \mathcal{H}_{\beta E} & \mathcal{H}_{\beta\beta} \end{pmatrix}, \quad (2.133)$$

Temos então a AI ortogonal a todos os orbitais do banho. Como mais uma etapa¹⁰ podemos diagonalizar o overlap de AI $\tilde{S}_{AI} = U_\psi^\dagger \mathcal{S}_I U_\psi$ e o Hamiltoniano $\tilde{\mathcal{H}}_{AI} = U_\psi^\dagger \mathcal{H}_I U_\psi$. Para este objetivo, primeiro ortogonalizamos os orbitais no AI por uma transformação de Löwdin calculando a raiz quadrada do inverso da matriz de overlap, $\tilde{S}_{AI}^{-\frac{1}{2}}$, e então resolvendo todos os autovetores, $U_{AI\psi}$, e autovalores do sistema resultante $(\tilde{S}_{AI}^{-\frac{1}{2}} \tilde{\mathcal{H}}_{AI} \tilde{S}_{AI}^{-\frac{1}{2}}) U_{AI\psi} = U_{AI\psi} \epsilon_{AI,D}$. Aqui $U_{AI\psi}$ é uma matriz de dimensão $N_{AI} \times N_{AI}$ com cada coluna igual a um autovetor associado a um

¹⁰A princípio, é opcional, mas para alguns solucionadores de AI é necessário, e em particular para o de interesse deste trabalho.

autovalor do sistema, e $\epsilon_{AI,D}$ é uma matriz diagonal com dimensão $N_{AI} \times N_{AI}$, onde os elementos diagonais são os autovalores correspondentes. A matriz de transformação resultante para a nova base atua apenas nos orbitais do AI, e é

$$W_2 = \begin{pmatrix} W_{2,AI} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1_{N_{NI}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}, \quad (2.134)$$

com

$$W_{2,AI} = \tilde{\mathcal{S}}_{AI}^{-\frac{1}{2}} U_{AI} \psi. \quad (2.135)$$

As matrizes transformadas são $\overline{\mathcal{H}} = W_2^\dagger \tilde{\mathcal{H}} W_2$ e $\overline{\mathcal{S}} = W_2^\dagger \tilde{\mathcal{S}} W_2$ e mantêm as estruturas não nulas de $\tilde{\mathcal{H}}$ e $\tilde{\mathcal{S}}$, por exceção dos elementos referentes a $\overline{\mathcal{S}}_{AI}$ e $\overline{\mathcal{H}}_{AI}$ que se tornam diagonais.

A matriz de transformação total, W , então é

$$W = W_0 W_1 W_2, \quad (2.136)$$

que ao realizamos a multiplicação, resulta em

$$W = \begin{pmatrix} 0 & 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ W_{I,AI} & 0 & W_{I,EL} & W_{I,NI} & W_{I,ER} & 0 \\ 0 & 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}, \quad (2.137)$$

de modo que $\overline{\mathcal{H}} = W^\dagger \mathcal{H} W$ e $\overline{\mathcal{H}} = W^\dagger \mathcal{S} W$. Aqui apresentamos

$$W_{I, AI} = U_\psi W_{2, AI}, \quad (2.138)$$

$$W_{I, EL} = -W_\psi \mathcal{S}_{I, EL}, \quad (2.139)$$

$$W_{I, NI} = (1_{NI} - W_\psi \mathcal{S}_I) U_{NI}, \quad (2.140)$$

e

$$W_{I, ER} = -W_\psi \mathcal{S}_{I, ER}. \quad (2.141)$$

com

$$W_\psi = U_\psi (U_\psi^\dagger \mathcal{S}_I U_\psi)^{-1} U_\psi^\dagger. \quad (2.142)$$

As equações explícitas relacionando as matrizes originais com as transformadas

$$\overline{\mathcal{S}} = \begin{pmatrix} 1_{AI} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathcal{S}_{\alpha\alpha} & \mathcal{S}_{\alpha E} & 0 & 0 & 0 \\ 0 & \mathcal{S}_{\alpha E}^\dagger & \overline{\mathcal{S}}_{EL} & \overline{\mathcal{S}}_{NI, EL}^\dagger & \overline{\mathcal{S}}_{EL, ER} & 0 \\ 0 & 0 & \overline{\mathcal{S}}_{NI, EL} & \overline{\mathcal{S}}_{NI} & \overline{\mathcal{S}}_{NI, ER} & 0 \\ 0 & 0 & \overline{\mathcal{S}}_{EL, ER}^\dagger & \overline{\mathcal{S}}_{NI, ER}^\dagger & \overline{\mathcal{S}}_{ER} & \mathcal{S}_{E\beta} \\ 0 & 0 & 0 & 0 & \mathcal{S}_{E\beta}^\dagger & \mathcal{S}_{\beta\beta} \end{pmatrix} \quad (2.143)$$

com

$$\overline{\mathcal{S}}_{EL} = \mathcal{S}_{EL} + 2\mathcal{S}_{I, EL}^\dagger W_{I, EL}^\dagger \mathcal{S}_I W_{I, EL}, \quad (2.144)$$

$$\overline{\mathcal{S}}_{NI, EL} = W_{I, NI}^\dagger (\mathcal{S}_{I, EL} + \mathcal{S}_I W_{I, EL}), \quad (2.145)$$

$$\overline{\mathcal{S}}_{EL, ER} = \mathcal{S}_{EL, ER} + 2\mathcal{S}_{I, EL}^\dagger W_{I, ER} + W_{I, EL}^\dagger \mathcal{S}_I W_{I, ER}, \quad (2.146)$$

$$\overline{\mathcal{S}}_{NI} = W_{I, NI}^\dagger \mathcal{S}_I W_{I, NI}, \quad (2.147)$$

$$\overline{\mathcal{S}}_{NI, ER} = W_{I, NI}^\dagger (\mathcal{S}_{I, ER} + \mathcal{S}_I W_{I, ER}), \quad (2.148)$$

$$\overline{\mathcal{S}}_{ER} = \mathcal{S}_{ER} + 2\mathcal{S}_{I, ER}^\dagger W_{I, ER} + W_{I, ER}^\dagger \mathcal{S}_I W_{I, ER} + W_{I, ER}^\dagger \mathcal{S}_I W_{I, ER}. \quad (2.149)$$

Para o Hamiltoniano temos

$$\bar{\mathcal{H}} = \begin{pmatrix} \epsilon_{AI,D} & 0 & \bar{\mathcal{H}}_{AI,EL} & \bar{\mathcal{H}}_{AI,NI} & \bar{\mathcal{H}}_{AI,ER} & 0 \\ 0 & \mathcal{H}_{\alpha\alpha} & \mathcal{H}_{\alpha E} & 0 & 0 & 0 \\ \bar{\mathcal{H}}_{AI,EL}^\dagger & \mathcal{H}_{\alpha E}^\dagger & \bar{\mathcal{H}}_{EL} & \bar{\mathcal{H}}_{NI,EL}^\dagger & \bar{\mathcal{H}}_{EL,ER} & 0 \\ \bar{\mathcal{H}}_{AI,NI}^\dagger & 0 & \bar{\mathcal{H}}_{NI,EL} & \bar{\mathcal{H}}_{NI} & \bar{\mathcal{H}}_{NI,ER} & 0 \\ \bar{\mathcal{H}}_{AI,ER}^\dagger & 0 & \bar{\mathcal{H}}_{EL,ER}^\dagger & \bar{\mathcal{H}}_{NI,ER}^\dagger & \bar{\mathcal{H}}_{ER} & \mathcal{H}_{\beta E}^\dagger \\ 0 & 0 & 0 & 0 & \mathcal{H}_{\beta E} & \mathcal{H}_{\beta\beta} \end{pmatrix}, \quad (2.150)$$

$$\epsilon_{AI,D} = W_{I,AI}^\dagger \mathcal{H}_I W_{I,AI}, \quad (2.151)$$

$$\bar{\mathcal{H}}_{AI,EL} = W_{I,AI}^\dagger (\mathcal{H}_{I,EL} + \mathcal{H}_I W_{I,EL}), \quad (2.152)$$

$$\bar{\mathcal{H}}_{AI,NI} = W_{I,AI}^\dagger \mathcal{H}_I W_{I,NI}, \quad (2.153)$$

$$\bar{\mathcal{H}}_{AI,ER} = W_{I,AI}^\dagger (\mathcal{H}_{I,ER} + \mathcal{H}_I W_{I,ER}), \quad (2.154)$$

$$\bar{\mathcal{H}}_{EL} = \mathcal{H}_{EL} + \mathcal{H}_{I,EL}^\dagger W_{I,EL} + W_{I,EL}^\dagger \mathcal{H}_{I,EL} + W_{I,EL}^\dagger \mathcal{H}_I W_{I,EL}, \quad (2.155)$$

$$\bar{\mathcal{H}}_{NI,EL} = W_{I,NI}^\dagger (\mathcal{H}_{I,EL} + \mathcal{H}_I W_{I,EL}), \quad (2.156)$$

$$\bar{\mathcal{H}}_{EL,ER} = \mathcal{H}_{EL,ER} + \mathcal{H}_{I,EL}^\dagger W_{I,ER} + W_{I,EL}^\dagger \mathcal{H}_{I,ER} + W_{I,EL}^\dagger \mathcal{H}_I W_{I,ER}, \quad (2.157)$$

$$\bar{\mathcal{H}}_{NI} = W_{I,NI}^\dagger \mathcal{H}_I W_{I,NI}, \quad (2.158)$$

$$\bar{\mathcal{H}}_{NI,ER} = W_{I,NI}^\dagger (\mathcal{H}_{I,ER} + \mathcal{H}_I W_{I,ER}), \quad (2.159)$$

e

$$\bar{\mathcal{H}}_{ER} = \mathcal{H}_{ER} + \mathcal{H}_{I,ER}^\dagger W_{I,ER} + W_{I,ER}^\dagger \mathcal{H}_{I,ER} + W_{I,ER}^\dagger \mathcal{H}_I W_{I,ER}. \quad (2.160)$$

Matriz de projeção inversa

Antes de passar para a transformação correspondente da função de Green, G , e das auto-energias, é necessário olharmos para a inversa de W que será utilizada

em vários momentos diferentes:

$$W^{-1} = W_2^{-1} W_1^{-1} W_0^{-1}. \quad (2.161)$$

A inversa de W_0 é

$$W_0^{-1} = \begin{pmatrix} 0 & 0 & A & 0 & 0 \\ 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ 0 & 0 & B & 0 & 0 \\ 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}, \quad (2.162)$$

onde A é uma matriz $N_{AI} \times N_{IR}$, e B é uma matriz $N_{NI} \times N_{IR}$, obtidas avaliando

$$\begin{pmatrix} A \\ B \end{pmatrix} = (U_\psi \quad U_{NI})^{-1}. \quad (2.163)$$

Se lembrarmos que todos os vetores unitários removidos ao formar U_{NI} são combinados em uma matriz $U_\perp$ ($U_\perp^\dagger U_{NI} = 0$), então podemos fornecer expressões analíticas para A e B como

$$A = (U_\perp^\dagger U_\psi)^{-1} U_\perp^\dagger \quad (2.164)$$

$$B = U_{NI}^\dagger \left(1_{N_{IR}} - U_\psi (U_\perp^\dagger U_\psi)^{-1} U_\perp^\dagger \right). \quad (2.165)$$

Observe que, como para sistemas grandes, as matrizes U_{NI} e $U_\perp$ consistem principalmente em zeros, e a matriz $(U_\perp^\dagger U_\psi)$ tem apenas dimensão $N_{AI} \times N_{AI}$, o cálculo dos elementos diferentes de zero de W_0^{-1} utilizando as relações analíticas para A e

B é rápido ($\text{ordem}(N)$). A inversa de W_1 é

$$W_1^{-1} = \begin{pmatrix} 1_{N_{AI}} & 0 & W_A & W_B & W_C & 0 \\ 0 & 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1_{N_{NI}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}, \quad (2.166)$$

e a inversa de W_2 é

$$W_2^{-1} = \begin{pmatrix} W_{2,AI}^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1_{N_{NI}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}. \quad (2.167)$$

Multiplicando todos os componentes obtemos

$$W^{-1} = \begin{pmatrix} 0 & W_{iEL} & W_{iI} & W_{iER} & 0 \\ 1_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 1_{NEL} & 0 & 0 & 0 \\ 0 & 0 & B & 0 & 0 \\ 0 & 0 & 0 & 1_{NER} & 0 \\ 0 & 0 & 0 & 0 & 1_{N_\beta} \end{pmatrix}, \quad (2.168)$$

onde,

$$W_{iEL} = W_{2,AI}^{-1} W_{i\psi} \mathcal{S}_{I,EL}, \quad (2.169)$$

$$W_{iI} = W_{2,AI}^{-1} W_{i\psi} \mathcal{S}_I, \quad (2.170)$$

e

$$W_{iER} = W_{2,AI}^{-1} W_{i\psi} \mathcal{S}_{I,ER}, \quad (2.171)$$

com

$$W_{i\psi} = \left(U_\psi^\dagger \mathcal{S}_I U_\psi \right)^{-1} U_\psi^\dagger. \quad (2.172)$$

Auto-energias e funções de Green

Passamos agora para a projeção da função de Green não interagente, $G = [E\mathcal{S} - \mathcal{H} - \Sigma_R - \Sigma_L]^{-1}$, na nova base, partindo da sua definição, podemos expressá-la na base transformada

$$\bar{G} = [E\bar{\mathcal{S}} - \bar{\mathcal{H}} - \bar{\Sigma}_R - \bar{\Sigma}_L]^{-1}. \quad (2.173)$$

Um fato importante é que a transformação é projetada de tal forma que não afeta as camadas limites da ME, de modo que temos

$$\bar{\Sigma}_R = W^\dagger \Sigma_R W = \Sigma_R \quad (2.174)$$

$$\bar{\Sigma}_L = W^\dagger \Sigma_L W = \Sigma_L \quad (2.175)$$

Podemos, portanto, escrever $\bar{G} = W^{-1}[ES - \mathcal{H} - \Sigma_R - \Sigma_L]W^{-1^\dagger}$, de modo que

$$\bar{G} = W^{-1}GW^{-1^\dagger}. \quad (2.176)$$

É importante ressaltar que apenas os elementos até o RIE são diferentes para $\bar{G}$ e G , enquanto todos os outros elementos permanecem inalterados. Em um sistema grande, a transformação precisa, portanto, ser aplicada apenas a uma sub-região relativamente pequena, tornando-a adequada para simulações de Ordem(N) em grande escala.

Se escrevermos a matriz G com dimensões $N \times N$ completa na forma de sub-blocos

$$G = \begin{pmatrix} G_{\alpha\alpha} & G_{\alpha,EL} & G_{\alpha,I} & G_{\alpha,ER} & G_{\alpha\beta} \\ G_{EL,\alpha} & G_{EL,EL} & G_{EL,I} & G_{EL,ER} & G_{EL,\beta} \\ G_{I,\alpha} & G_{I,EL} & G_I & G_{I,ER} & G_{I,\beta} \\ G_{ER,\alpha} & G_{ER,EL} & G_{ER,I} & G_{ER} & G_{ER,\beta} \\ G_{\beta\alpha} & G_{\beta,EL} & G_{\beta,I} & G_{\beta,ER} & G_{\beta\beta} \end{pmatrix}, \quad (2.177)$$

obtemos para a função de Green projetada na AI:

$$\bar{G}_{AI} = (W_{iEL}, W_{iI}, W_{iER}) \begin{pmatrix} G_{EL} & G_{EL,I} & G_{EL,ER} \\ G_{I,EL} & G_I & G_{I,ER} \\ G_{ER,EL} & G_{ER,I} & G_{ER} \end{pmatrix} \begin{pmatrix} W_{IEL}^\dagger \\ W_{iI}^\dagger \\ W_{iER}^\dagger \end{pmatrix}. \quad (2.178)$$

Com $\bar{G}_{AI}$ contendo os elementos do RIE. Considerando a estrutura matricial de $\bar{\mathcal{S}}$ e $\bar{\mathcal{H}}$, podemos escrever $\bar{G}_{AI}$ equivalentemente como

$$\bar{G}_{AI} = [E - \epsilon_{AI,D} - \bar{\Delta}_{AI}]^{-1}, \quad (2.179)$$

com a chamada função de hibridização $\bar{\Delta}_{AI}$ dada por

$$\bar{\Delta}_{AI} = \bar{\mathcal{H}}_{AI,NE} \bar{G}_{NE}^B \bar{\mathcal{H}}_{AI,NE}^\dagger. \quad (2.180)$$

Aqui introduzimos,

$$\bar{\mathcal{H}}_{AI,NE} = (\bar{\mathcal{H}}_{AI,EL}, \bar{\mathcal{H}}_{AI,NI}, \bar{\mathcal{H}}_{AI,ER}), \quad (2.181)$$

$$\bar{G}_{NE}^B = \begin{pmatrix} \bar{G}_{EL}^B & \bar{G}_{EL,NI}^B & \bar{G}_{EL,ER}^B \\ \bar{G}_{NI,EL}^B & \bar{G}_{NI}^B & \bar{G}_{NI,ER}^B \\ \bar{G}_{ER,EL}^B & \bar{G}_{ER,NI}^B & \bar{G}_{ER}^B \end{pmatrix}, \quad (2.182)$$

que é o sub-bloco que se estende sobre a região estendida não interagente (“NE”) da Função de Green do banho não interagente, $\bar{G}^B$,

$$\bar{G}^B = \begin{pmatrix} \bar{G}_{\alpha\alpha}^B & \bar{G}_{\alpha,El}^B & \bar{G}_{\alpha,I}^B & \bar{G}_{\alpha,ER}^B & \bar{G}_{\alpha\beta}^B \\ \bar{G}_{EL,\alpha}^B & \bar{G}_{EL,EL}^B & \bar{G}_{EL,I}^B & \bar{G}_{EL,ER}^B & \bar{G}_{EL,\beta}^B \\ \bar{G}_{I,\alpha}^B & \bar{G}_{I,EL}^B & \bar{G}_I^B & \bar{G}_{I,ER}^B & \bar{G}_{I,\beta}^B \\ \bar{G}_{ER,\alpha}^B & \bar{G}_{ER,EL}^B & \bar{G}_{ER,I}^B & \bar{G}_{ER}^B & \bar{G}_{ER,\beta}^B \\ \bar{G}_{\beta\alpha}^B & \bar{G}_{\beta,EL}^B & \bar{G}_{\beta,I}^B & \bar{G}_{\beta,ER}^B & \bar{G}_{\beta\beta}^B \end{pmatrix}. \quad (2.183)$$

Isto inclui todos os orbitais dentro da ME, exceto os estados AI, e é definida como

$$\bar{G}^B = [E\bar{\mathcal{S}}^B - \bar{\mathcal{H}}^B - \Sigma_R - \Sigma_L]^{-1}, \quad (2.184)$$

com

$$\bar{\mathcal{S}}^B = \begin{pmatrix} \mathcal{S}_{\alpha\alpha} & \mathcal{S}_{\alpha E} & 0 & 0 & 0 \\ \mathcal{S}_{\alpha E}^\dagger & \bar{\mathcal{S}}_{EL} & \bar{\mathcal{S}}_{EL,NI} & \bar{\mathcal{S}}_{EL,ER} & 0 \\ 0 & \bar{\mathcal{S}}_{EL,NI}^\dagger & \bar{\mathcal{S}}_{NI} & \bar{\mathcal{S}}_{NI,ER} & 0 \\ 0 & \bar{\mathcal{S}}_{EL,ER}^\dagger & \bar{\mathcal{S}}_{NI,ER}^\dagger & \bar{\mathcal{S}}_{ER} & \mathcal{S}_{E\beta} \\ 0 & 0 & 0 & \mathcal{S}_{E\beta}^\dagger & \mathcal{S}_{\beta\beta} \end{pmatrix}. \quad (2.185)$$

e

$$\overline{\mathcal{H}}^B = \begin{pmatrix} \mathcal{H}_{\alpha\alpha} & \mathcal{H}_{\alpha E} & 0 & 0 & 0 \\ \mathcal{H}_{\alpha E}^\dagger & \overline{\mathcal{H}}_{EL} & \overline{\mathcal{H}}_{EL,NI} & \overline{\mathcal{H}}_{EL,ER} & 0 \\ 0 & \overline{\mathcal{H}}_{EL,NI}^\dagger & \overline{\mathcal{H}}_{NI} & \overline{\mathcal{H}}_{NI,ER} & 0 \\ 0 & \overline{\mathcal{H}}_{EL,ER}^\dagger & \overline{\mathcal{H}}_{NI,ER}^\dagger & \overline{\mathcal{H}}_{ER} & \mathcal{H}_{E\beta} \\ 0 & 0 & 0 & \mathcal{H}_{E\beta}^\dagger & \mathcal{H}_{\beta\beta} \end{pmatrix}. \quad (2.186)$$

De modo que, $\overline{\mathcal{H}}^B$ e $\overline{\mathcal{S}}^B$, se relacionam com $\overline{\mathcal{H}}$ e $\overline{\mathcal{S}}$, através da relação,

$$\overline{\mathcal{S}} = \begin{pmatrix} 1_{AI} & 0 \\ 0 & \overline{\mathcal{S}}^B \end{pmatrix}, \quad (2.187)$$

$$\overline{\mathcal{H}} = \begin{pmatrix} \epsilon_{AI,D} & \overline{\mathcal{H}}_{AI,B} \\ \overline{\mathcal{H}}_{B,AI} & \overline{\mathcal{H}}^B \end{pmatrix}, \quad (2.188)$$

com $\overline{\mathcal{H}}_{AI,B}$ e $\overline{\mathcal{H}}_{B,AI}$, sub-blocos quem contem o acoplamento entre os orbitais da região AI e da ME, mostrado na Eq. 2.150. Observe que, para sistemas grandes, $\mathcal{H}_{\alpha\alpha}$ e $\mathcal{H}_{\beta\beta}$ são matrizes muito maiores que os sub-blocos da região NE, de modo que usando o algoritmo da função de Green recursiva podemos calcular $\overline{\Delta}_{AI}$ em operações de Ordem(N). Observamos também que, em geral, $\overline{\Delta}_{AI}$ é uma matriz $N_{AI} \times N_{AI}$ não nula.

Pode-se mostrar que de fato as Eq. 2.206 e Eq. 2.179 são equivalentes. Para obter $\overline{\Delta}_{AI}$, pode-se, portanto, usar primeiro a Eq. 2.206 para avaliar $\overline{G}_{AI}$ e depois obter $\overline{\Delta}_{AI}$, como

$$\overline{\Delta}_{AI} = E - \epsilon_{AI,D} - \overline{G}_{AI}^{-1}. \quad (2.189)$$

No entanto, especialmente para E grande, esta forma indireta de obter $\overline{\Delta}_{AI}$ é significativamente menos precisa. A razão é que para E grande, as quantidades E e $\overline{G}_{AI}^{-1}$ são, ambos, muito grandes, enquanto a diferença resultante é pequena, com um termo principal proporcional a $1/E$. Por outro lado, desde que a região de interesse esteja em valores de E próximo ao nível de Fermi, ambas são uma opção viável.

Auto-energia de Muitos corpos

De maneira geral, para resolvermos um problema interagente, (independe da sua natureza) sobre AI é necessário o cálculo da função de Green de muitos corpos, $\bar{G}_{AI}^{MB}$. Isso leva à definição da auto-energia de muitos corpos, $\bar{\Sigma}_{AI}^{MB}$, como

$$\bar{\Sigma}_{AI}^{MB} = \bar{G}_{AI}^{-1} - (\bar{G}_{AI}^{MB})^{-1}. \quad (2.190)$$

As duas $\bar{G}_{AI}^{MB}$ e $\bar{\Sigma}_{AI}^{MB}$ podem ser obtidas usando um dos **solvers** de impurezas de Anderson disponíveis. Uma parte importante é que após obtida a solução devemos realizar a transformação inversa e voltar a uma base de orbitais atômicos (ou outra base na qual ela foi inicialmente escrita). Como estamos considerando o caso de uma única RI, a auto-energia completa de muitos corpos é zero para todos os elementos, exceto para aqueles da AI

$$\bar{\Sigma}^{MB} = \begin{pmatrix} \bar{\Sigma}_{AI}^{MB} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0_{N_{EL}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0_{N_{NI}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0_{N_\beta} \end{pmatrix}. \quad (2.191)$$

Neste caso podemos, portanto, aplicar o análogo ao que fizemos com a função de Green, para a $\bar{\Sigma}^{MB}$ completa

$$\bar{\Sigma}^{MB} = \bar{G}^{-1} - (\bar{G}^{MB})^{-1}. \quad (2.192)$$

como $\bar{G} = W^{-1}GW^{-1\dagger}$ e $\bar{G}^{MB} = W^{-1}G^{MB}W^{-1\dagger}$, nós obtemos $\bar{\Sigma}^{MB} = W^\dagger(G^{-1} - (G^{MB})^{-1})W = W^\dagger\Sigma^{MB}W$, de modo que

$$\Sigma^{MB} = W^{-1\dagger}\bar{\Sigma}^{MB}W^{-1}. \quad (2.193)$$

A estrutura explícita de Σ^{MB} é dada por

$$\Sigma^{MB} = \begin{pmatrix} 0_{N_\alpha} & 0 & 0 & 0 & 0 \\ 0 & \Sigma_{EL}^{MB} & \Sigma_{EL,I}^{MB} & \Sigma_{EL,ER}^{MB} & 0 \\ 0 & \Sigma_{I,EL}^{MB} & \Sigma_I^{MB} & \Sigma_{I,ER}^{MB} & 0 \\ 0 & \Sigma_{ER,EL}^{MB} & \Sigma_{ER,I}^{MB} & \Sigma_{ER}^{MB} & 0 \\ 0 & 0 & 0 & 0 & 0_{N_\beta} \end{pmatrix}. \quad (2.194)$$

O bloco diferente de zero de Σ^{MB} se estende sobre o RIE (denotamos como Σ_E^{MB}), e sua forma explícita é

$$\Sigma^{MB} = \begin{pmatrix} \Sigma_{EL}^{MB} & \Sigma_{EL,I}^{MB} & \Sigma_{EL,ER}^{MB} \\ \Sigma_{I,EL}^{MB} & \Sigma_I^{MB} & \Sigma_{I,ER}^{MB} \\ \Sigma_{ER,EL}^{MB} & \Sigma_{ER,I}^{MB} & \Sigma_{ER}^{MB} \end{pmatrix} = \begin{pmatrix} W_{IEL}^+ \\ W_{iI}^+ \\ W_{iER}^+ \end{pmatrix} \bar{\Sigma}^{MB}(W_{IEL}, W_{iI}, W_{iER}). \quad (2.195)$$

Após calcular Σ^{MB} pode-se então obter a função de Green interagente como

$$G = [ES - \mathcal{H} - \Sigma_R - \Sigma_L - \Sigma^{MB}]^{-1}. \quad (2.196)$$

Com essa formulação para a auto-energia de muitos corpos, pode ser utilizada em qualquer tipo de interação (seja elétron-elétron ou elétron-fônon, por exemplo), desde que o método aplicado a resolver a autoneergia seja o seu respectivo *solver*. Em particular para o nosso trabalho, consideraremos apenas a interação elétron-elétron, assim $\Sigma^{MB} = \Sigma^{ee}$.

Overlap ortogonal

Até aqui, apresentamos como construir as matrizes de projeções de maneira geral. A metodologia pode ser implementada em códigos avançados de estrutura eletrônica que utilizam bases localizadas, como, por exemplo, o código SIESTA, que se baseia na teoria do funcional de densidade.

Em particular, um caso de interesse é quando os estados da base já são inicialmente ortogonais entre si e, portanto, a matriz de overlap é a identidade, ou

seja,

$$\mathcal{S}_{\mu\nu} = \langle \phi_\mu | \phi_\nu \rangle = \delta_{\mu,\nu}, \quad (2.197)$$

que em muitos casos é uma boa aproximação. De maneira explícita

$$\mathcal{S} = \begin{pmatrix} 1_{AI} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1_{\alpha\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{EL} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1_{NI} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1_{ER} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{\beta\beta} \end{pmatrix}. \quad (2.198)$$

Olhando diretamente para os sub-blocos da matriz inversa da projeção total W^{-1} com

$$W_{i\psi} = U_\psi^\dagger, \quad (2.199)$$

$$W_{2,AI}^{-1} = 1_{N_{AI}}, \quad (2.200)$$

temos,

$$W_{iEL} = 0, \quad (2.201)$$

$$W_{iI} = U_\psi^\dagger, \quad (2.202)$$

$$W_{iER} = 0, \quad (2.203)$$

e

$$W_{iER} = 0, \quad (2.204)$$

de maneira que

$$W^{-1} = \begin{pmatrix} 0 & 0 & U_{\psi}^{\dagger} & 0 & 0 \\ 1_{N_{\alpha}} & 0 & 0 & 0 & 0 \\ 0 & 1_{N_{EL}} & 0 & 0 & 0 \\ 0 & 0 & B & 0 & 0 \\ 0 & 0 & 0 & 1_{N_{ER}} & 0 \\ 0 & 0 & 0 & 0 & 1_{N_{\beta}} \end{pmatrix}. \quad (2.205)$$

Isto é o equivalente a fazermos apenas a primeira rotação com W_0^{-1} , pois as demais operações tinham como objetivo diagonalizar o overlap entre AI e as demais regiões. Como consequência, nesse caso em particular, $W^{-1} = W_0^{-1}$, com $A = U_{\psi}^{\dagger}$ e $B = U_{NI}$.

Obtemos a função de Green projetada sobre AI, com a nova projeção explícita.

$$\bar{G}_{AI} = (0, U_{\psi}^{\dagger}, 0) \begin{pmatrix} G_{EL} & G_{EL,I} & G_{EL,ER} \\ G_{I,EL} & G_I & G_{I,ER} \\ G_{ER,EL} & G_{ER,I} & G_{ER} \end{pmatrix} \begin{pmatrix} 0 \\ U_{\psi} \\ 0 \end{pmatrix}, \quad (2.206)$$

$$\bar{G}_{AI} = U_{\psi}^{\dagger} G_I U_{\psi}. \quad (2.207)$$

Concluimos que, na presença de overlap diagonal, projetar a função de Green completa sobre AI é o equivalente a projetar apenas a função de Green da RI diretamente sobre a base das AI, U_{ψ} .

2.8 Teoria do campo médio dinâmico (DMFT)

A Teoria do Campo Médio Dinâmico (DMFT, da sigla em inglês) consiste em mapear a função de Green do modelo de Hubbard na do modelo de Anderson para impurezas magnéticas, corrigida pela função de hibridização, ou seja, [67]

$$[E\mathbb{I} - H_{\mathbf{H}} - \Sigma_{ee}]_{i,j}^{-1} = [E\mathbb{I} - H_{\mathbf{AI}} - \bar{\Delta}_{bath} - \Sigma_{ee}]_{i,j}^{-1}. \quad (2.208)$$

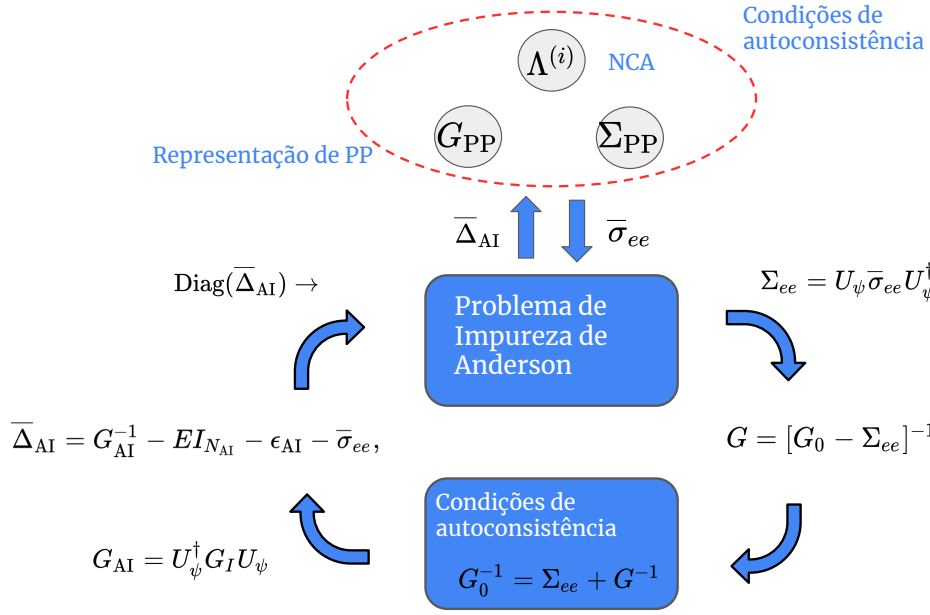


Figura 2.22: **Visão esquemática do ciclo autoconsistente:** Ilustração do ciclo autoconsistente passo a passo adotado nesta dissertação. Aqui, o ciclo é uma extensão do DMFT usual.

Importante notar aqui que a relação é local, assim saímos de um problema multi-orbital interagente para um modelo de Anderson para um único nível interagente, e, portanto, para considerar a rede completa, é necessário construir um ciclo autoconsistente.

Como já visto acima, definimos a função de Green retardada e interagente, por

$$G = [E - \mathcal{H} - \Sigma_R - \Sigma_L - \Sigma_{ee}]^{-1}, \quad (2.209)$$

e a não interagente, como $G_0 = G(\Sigma_{ee} = 0)$. Importante notar que dessa maneira, a equação de Dyson (Eq. 2.45) nos fornece a seguinte relação $G = [G_0^{-1} - \Sigma_{ee}]^{-1}$. Como ilustrado na Fig. 2.22, No primeiro ciclo definimos a base com estados localizados de interesse U_ψ com N_{AI} estados, tomamos G_0 e projetando, temos:

$$G_{AI} = [\epsilon_{AI} - EI_{N_{AI}} - \overline{\Delta_{AI}}]. \quad (2.210)$$

A função de Green nessa base, por construção, tem a estrutura da função de Green do Modelo de Anderson para impurezas magnéticas com N_{AI} níveis. Dessa relação, obtemos então a função de Híbridização $\overline{\Delta_{AI}}$. Aqui ela carrega a informação dos eletrodos Σ_L e Σ_R nessa base, além dos estados que não foram projetados e se acoplam com os da base. Posteriormente, por simplicidade, redefinimos

$\overline{\Delta_{AI}} = \text{Diag}(\overline{\Delta_{AI}})$ com apenas os elementos da diagonal, devido à natureza do *solver* usado [60]. Usamos então *solver* para impurezas de Anderson. Para o nosso caso, consiste em mudarmos a base mais uma vez, para a base de PP e realizando uma expansão diagramática na hibridização, descartando as correções de vértices (NCA), e obtendo a auto-energia de interação de maneira autoconsistente e retornando para a base das AI. Obtemos então Σ^{MB} e fazendo mais uma vez a projeção inversa, obtemos $\Sigma_{ee} = U_\psi \Sigma^{MB} U_\psi^T$, a auto-energia na base dos orbitais atômicos. Por fim, obtemos a função de Green interagente. E devido ao fato de escolhermos apenas os termos da diagonal de $\overline{\Delta_{AI}}$, Σ^{MB} também é diagonal. Por isso precisamos repetir o ciclo para obter autoconsistência na aproximação.

2.9 Transporte Quântico Decoerente

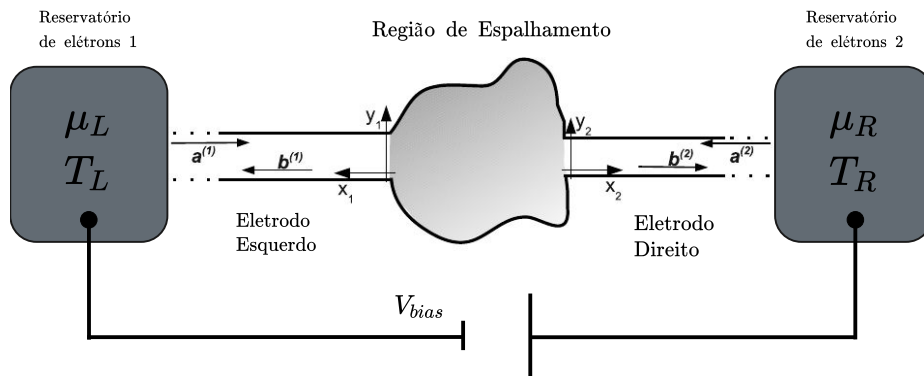


Figura 2.23: **Visão esquemática do problema de transporte eletrônico:** $\mu_{L/(R)}$ e $T_{L/(R)}$ são o potencial químico e a temperatura, respectivamente do eletrodo esquerdo (direito), com uma diferença de potencial V_{bias} aplicada entre os eletrodos. **a** e **b** as amplitudes de entrada e saída respectivamente, da região de espalhamento.

Até aqui, olhamos de maneira genérica para as propriedades eletrônicas de nanosistemas sem considerar observáveis específicos. O transporte eletrônico em nanodispositivos é uma área de interesse da comunidade científica, principalmente devido ao seu grande número de aplicações em tecnologias, como é o caso dos transistores (MOSFETs, por exemplo) [2]. Este problema traz consigo vários desafios para o campo da física da matéria condensada fora do equilíbrio. A fim de modelar este problema, inicialmente temos uma região de espalhamento (onde

os efeitos de interesse acontecem) conectada a reservatórios de elétrons por meio dos eletrodos. Os reservatórios são grandes o suficiente para serem considerados em equilíbrio termodinâmico com temperaturas fixas T_L e T_R , com μ_L e μ_R o potencial químico em cada reservatório. Por fim, aplicando uma diferença de potencial $V_{bias} = \mu_R - \mu_L$ quebramos o equilíbrio do sistema, resultando no transporte de carga (e energia) pela região de espalhamento. Dependendo dos elementos que compõem a estrutura interna da região de espalhamento, isso leva a diversos tipos de processos. Em particular, podemos classificar o problema em função do comprimento: *i*) Transporte coerente, para sistemas menores que o comprimento de onda característico do elétron. *ii*) Transporte incoerente, na presença de fenômenos que quebram a coerência de fase quântica eletrônica, como interação elétron-fônon e elétron-elétron.

2.9.1 Transporte Coerente: Formalismo de Landauer-Büttiker

Proposto por R. Landauer e Büttiker [44, 68, 69] a ideia central é que o problema de transporte eletrônico (a condutância) está associado a probabilidade de transmissão de elétrons espalhados por um potencial. O formalismo de espalhamento associa as amplitudes de entrada e saída de uma região para um sistema de multi-canaís na presença de coerência de fase quântica. Como ilustrado na Fig. 2.23, o confinamento transversal em sistemas de baixa dimensionalidade produz modos discretos. Assim, a função de onda total é uma combinação linear de ondas planas (ou estados de Bloch) entrando e saindo da região de espalhamento e dadas por

$$\Psi(\vec{r}_l) = \sum_{n=1}^{N_l} \Psi_n(\vec{r}_l) = \sum_{n=1}^{N_l} \left[a_n^{(l)} \frac{e^{-ik_n^{(l)} x_l}}{(\hbar k_n^{(l)} / m)^{1/2}} + b_n^{(l)} \frac{e^{ik_n^{(l)} x_l}}{(\hbar k_n^{(l)} / m)^{1/2}} \right] \chi_n(y_l). \quad (2.211)$$

O sinal positivo (negativo) corresponde a ondas saindo (entrando) da (na) região de espalhamento, $a_n^{(l)}$ ($b_n^{(l)}$) são as amplitudes da onda entrando (saindo) da região e $l = L, R$ refere aos eletrodos correspondentes. Definindo os vetores

$$\mathbf{a}^{(l)} = (a_1^{(l)}, \dots, a_{N_l}^{(l)})^T, \quad \mathbf{b}^{(l)} = (b_1^{(l)}, \dots, b_{N_l}^{(l)})^T, \quad (2.212)$$

de maneira geral

$$\mathbf{a} = (\mathbf{a}^{(L)}, \mathbf{a}^{(R)})^T, \quad \mathbf{b} = (\mathbf{a}^{(L)}, \mathbf{a}^{(R)})^T. \quad (2.213)$$

O vetor $\mathbf{a}$ ($\mathbf{b}$) carrega, portanto, a informação de todas as amplitudes entrando (saindo) na (da) região de espalhamento. A matriz $\mathbf{S}$ que associa estas amplitudes,

é dada por

$$\mathbf{b} = \mathbf{S}\mathbf{a}, \quad (2.214)$$

e possui a seguinte estrutura para o caso multi-canais

$$\mathbf{S} = \begin{pmatrix} \mathbf{r}_{N_L \times N_L} & \mathbf{t}'_{N_L \times N_R} \\ \mathbf{t}_{N_R \times N_L} & \mathbf{r}'_{N_R \times N_R} \end{pmatrix}, \quad (2.215)$$

com $N_{(l)}$ o números de canais abertos no eletrodo l . $\mathbf{t}$ e $\mathbf{t}'$ as matrizes de transmissão, com $\mathbf{t}$ referente ao que é transmitido do eletrodo L para R e $\mathbf{t}'$ no sentido contrário. $\mathbf{r}$ é a matriz de reflexão referente ao que entra por L e volta por L e $\mathbf{r}'$ é análogo no eletrodo R . Assim, podemos obter os coeficientes τ_i e computar a transmissão total diretamente da fórmula de Landauer

$$T = \sum_i \tau_i = \text{Tr}\{\mathbf{t}\mathbf{t}'\}. \quad (2.216)$$

Na presença de uma diferença de potencial aplicada entre os reservatórios, a corrente é dada por [44]

$$I = ev \frac{\delta n}{\delta E} (\mu_2 - \mu_1), \quad (2.217)$$

com v a velocidade de Fermi e $\frac{\delta n}{\delta E}$ a densidade de estados com velocidade positiva. Para o caso unidimensional, $\delta n / \delta k = 1 / \pi$, e consequentemente

$$\frac{\delta n}{\delta E} = \frac{1}{\pi \hbar v}. \quad (2.218)$$

Considerando que existe uma probabilidade de transmissão e assumindo que para temperaturas não nulas os reservatórios seguem a distribuição de Fermi-Dirac $f_{(l)}(E) = 1 / (1 + e^{(E - \mu_{(l)}) / k_B T_{(l)}})$, temos

$$I_i = \frac{2e}{\hbar} \tau_i (f_R - f_L). \quad (2.219)$$

Para o caso multidimensional [44]

$$I = \frac{2e}{\hbar} \int T(E) (f_R - f_L) dE. \quad (2.220)$$

Finalmente, a condutância no regime de resposta linear, é dada por

$$\mathcal{G} = \lim_{V \rightarrow 0} \frac{I}{V} = \frac{2e}{\hbar} \int T \left[-\frac{df}{dE} \right] dE. \quad (2.221)$$

Aqui, observamos que, aparentemente, todas as quantidades são expressas dependendo apenas dos eletrodos, os quais são estados assintóticos. Assim, em um primeiro momento, as configurações microscópicas e internas da região de espalhamento são eliminados (implícitos) na descrição do problema. Uma maneira de explicitar os observáveis em termos da descrição que leve em consideração as propriedades internas da região de espalhamento é utilizando funções de Green.

Expressão de Fisher-Lee

Como relacionado por Fisher e Lee [45], inicialmente escrevemos a função de Green retardada na representação de posição. Temos,

$$G(\vec{r}_{(l)}, \vec{r}'_{(l)}) = \sum \frac{\Psi_n(\vec{r}_{(l)}) \Psi_n^*(\vec{r}'_{(l)})}{E - \epsilon_n + i\eta}. \quad (2.222)$$

De maneira análoga, na ausência de fenômenos que quebrem a coerência de fase, podemos usar as relações feitas anteriormente (2.211, 2.213 e 2.215), expressando 2.222 em termos das amplitudes de saída e entrada. A função de Green para os modos longitudinais é dada por

$$G_{l,l'}(x_l, x'_l) = \sum_{n'} (a_{n'}^{(l')} + b_{n'}^{(l')}) \mathcal{X}_n(y_l) e^{ik_{n'}|x-x'|}, \quad (2.223)$$

e para os modos transversais, por

$$G_{l,l'}(y_l, y'_l) = \sum_{n',n} [\delta_{nn'} a_{n'}^{(l')} + S'_{nn'} b_{n'}^{(l')}] \mathcal{X}_n(y_l). \quad (2.224)$$

Sabemos que as amplitudes são dadas por [22]

$$a_n^{(l)} = b_n^{(l)} = -\frac{i}{\hbar v_n} \mathcal{X}_n(y_l), \quad (2.225)$$

assim definimos uma nova matriz $S'_{nn'}$ em termos das amplitudes de espalhamento, e reescalada pela razão entre as velocidades

$$S'_{nn'} = \sqrt{v_{n'}/v_n} S_{nn'}. \quad (2.226)$$

Aqui, $v_n = \hbar k_n / m$ é a velocidade associada ao momento e à massa (m) da partícula. Por fim, para expressar os elementos da matrix S , integrando sobre y_l e $y_{l'}$ e usando

as relações de ortogonalidade das funções de onda, temos [22]

$$S_{nn'} = -\delta_{nn'} + i\hbar\sqrt{\nu_n\nu_{n'}} \int \int \mathcal{X}_n(y_l) [G_{l,l'}(y_l, y_{l'})] \mathcal{X}_{n'}(y_{l'}) dy_l dy_{l'}. \quad (2.227)$$

Esta é a chamada relação de Fisher-Lee. Calculando a função transmissão diretamente da matriz de espalhamento nessa expressão, ou seja, $T_{l,l'} = \sum_{nn'} S_{nn'} S_{n'n'}$, com $n \neq n'$. Temos:

$$T_{l,l'} = -\sum_{nn'} \hbar^2 \nu_n \nu_{n'} \int \mathcal{X}_n(y_l) [G_{l,l'}(y_l, y_{l'})] \mathcal{X}_{n'}(y_{l'}) \mathcal{X}_n(y_l) [G_{l',l}(y_{l'}, y_l)] \mathcal{X}_n(y_l) dy_l dy_{l'} dy_{l'} dy_l. \quad (2.228)$$

Usando as seguintes relações, para a função Green de superfície

$$g_{(l)}(l_i, l_i) = -\frac{1}{\gamma} \sum_n \mathcal{X}_n(l_i) \exp[ik_n a] \mathcal{X}_n(l_i), \quad (2.229)$$

e, portanto, para os elementos das matrizes das auto-energias,

$$\Sigma_{(l)}(i, j) = \gamma^2 g_{(l)}(l_i, l_i), \quad (2.230)$$

conseguimos então definir a parte imaginária das auto-energias diretamente das funções de onda transversais,

$$\Gamma_l(i, j) = i[\Sigma_l(i, j) - \Sigma_l^\dagger(i, j)] = \mathcal{X}_{n'}(l_i) \hbar \nu_n \mathcal{X}_{n'}(l_j). \quad (2.231)$$

Aqui, Γ_l é a função espectral da auto-energia do eletrodo l . Por fim, simplificamos a transmissão que assume a seguinte forma matricial,

$$T_{l,l'} = \text{Tr} \left\{ \Gamma_l G \Gamma_{l'} G^\dagger \right\}, \quad (2.232)$$

Esta é a expressão de Fisher-Lee usual para dois terminais [22]. Vale notar que aqui, diferente da expressão anterior, escrita em termos das quantidades que dependem apenas dos eletrodos, temos a transmissão em termos das quantidades locais e dependendo das simetrias internas do sistema. Foi demonstrado por Büttiker [44], que é fácil generalizar essa expressão para um problema de mult-terminais, de modo que a transmissão total no limite de $T \rightarrow 0$ é uma soma sobre permutações de todos os terminais possíveis seguindo a expressão de Fisher-Lee. Assim, se

tivermos três terminais, com $l = L, R, T_o$, a transmissão total sobre L é [44]

$$T_L = \text{Tr}\left\{\Gamma_L G \Gamma_R G^\dagger\right\} + \text{Tr}\left\{\Gamma_L G \Gamma_{T_o} G^\dagger\right\}. \quad (2.233)$$

2.9.2 Fenômeno de Decoerência: O problema de Büttiker

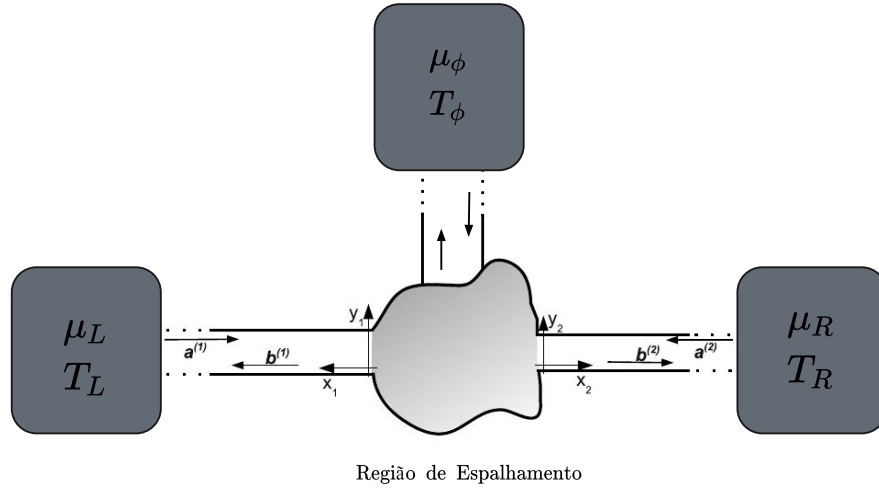


Figura 2.24: **Visão esquemática do problema de de Büttiker:** Análogo ao problema de espalhamento, temos $\mu_{L/(R)}$ e $T_{L/(R)}$ os potencial químico e a temperatura, respectivamente do eletrodo esquerdo (direito), μ_ϕ e T_ϕ potencial químico e a temperatura do eletrodo fictício **a** e **b** as amplitudes de entrada e saída respectivamente, da região de espalhamento.

Até aqui, discutimos o formalismo de transporte, porém apenas para o regime de processos coerentes. Para sistemas interagentes, podemos ter processos que ocorrem durante o espalhamento e quebram a coerência de fase da função de onda. Büttiker propôs [70, 69] uma maneira de olhar para os fenômenos de quebra de fase em um sistema de dois terminais, com uma descrição seguindo a metodologia de Landauer, através de um sistema análogo a três terminais, como mostrado no esquema da Fig. 2.24, onde todos os processos de quebra de fase ocorrem em um eletrodo fictício ϕ . Assim, do formalismo de Landauer-Büttiker para um sistema de três terminais, expressamos a corrente exatamente, por [22].

$$I = I_{\text{coerente}} + I_{\text{não-coerente}}, \quad (2.234)$$

com

$$I_{\text{coerente}}^R = \frac{2e}{\hbar} \int T_{R,L}(f_R - f_L) dE, \quad (2.235)$$

$$I_{\text{não-coerente}}^R = \frac{2e}{\hbar} \int T_{R,\phi}(f_R - f_\phi) dE. \quad (2.236)$$

de forma análoga podemos escrever a corrente para o terminal da esquerda,

$$I_{\text{coerente}}^L = \frac{2e}{\hbar} \int T_{L,R}(f_L - f_R) dE, \quad (2.237)$$

$$I_{\text{não-coerente}}^L = \frac{2e}{\hbar} \int T_{L,\phi}(f_L - f_\phi) dE. \quad (2.238)$$

Aqui, $T_{ll'}$ é a expressão de Fisher-Lee, dada pela Eq. 2.232, e f_ϕ é a distribuição de Fermi-Dirac do eletrodo fictício. O ponto fundamental aqui é que o elétron ao entrar/sair de um eletrodo termalizado, simulando um processo de espalhamento inelástico.

Uma primeira aproximação que pode ser feita, é considerar que existem apenas processos elásticos, mas o transporte pode ser não-coerente, ou seja, como o eletrodo ϕ é fictício, a densidade por energia de corrente líquida através dele deve ser zero ($i_\phi = 0$). Este *constraint* é fundamental na construção do modelo, de modo que

$$i_\phi = \frac{2e}{\hbar} \text{Tr} \left\{ \Gamma_\phi G \Gamma_R G^\dagger \right\} (f_\phi - f_R) + \frac{2e}{\hbar} \text{Tr} \left\{ \Gamma_\phi G \Gamma_L G^\dagger \right\} (f_\phi - f_L), \quad (2.239)$$

e temos,

$$i_\phi = 0, \quad \forall E. \quad (2.240)$$

Como a densidade de corrente total no eletrodo fictício é nula, isso implica dizer que a corrente total não-coerente também é nula. De 2.239,

$$f_\phi = \frac{1}{T_{\phi R}(E) + T_{\phi L}(E)} [T_{\phi R}(E) f_R + T_{\phi L}(E) f_L]. \quad (2.241)$$

Desta forma é possível obter o potencial químico $\mu_\phi(E)$ ¹¹ que satisfaz esta condição, e rescrevendo a corrente total

$$I = \frac{2e}{\hbar} \int \tilde{T}_{RL}(E) (f_R - f_L) dE, \quad (2.242)$$

com

$$\tilde{T}_{RL}(E) = T_{RL}(E) + \frac{T_{R\phi}(E) T_{\phi L}(E)}{T_{R\phi}(E) + T_{\phi L}(E)} \quad (2.243)$$

¹¹Note que μ_ϕ tem uma dependência com a energia e a equação deve ser resolvida para cada valor de E

usando 2.240, temos

$$I = \frac{2e}{\hbar} \int \left[\left(T_{RL}(E) + \frac{T_{R\phi}(E)T_{\phi L}(E)}{T_{R\phi}(E) + T_{\phi L}(E)} \right) (f_R - f_L) \right] dE. \quad (2.244)$$

Assim, para esse caso, dentro da amostra ocorrem processos de quebra de fase, mas todo o processo de transporte é elástico. Isso é análogo ao problema puramente coerente, mas agora em termos de uma transmissão efetiva, onde o primeiro termo carrega a informação dos processos coerentes e o segundo termo, dos processos não-coerentes.

Um próximo passo para ir além da perda de fase é impor $\int i_\phi dE = 0$. Neste caso, a equação 2.240 torna-se uma integral e resultará em um potencial químico único. Neste cenário, aparentemente, não é possível associar uma transmissão efetiva à condutância.

2.9.3 Transporte Interagente: Generalização da expressão de Meir-Wingreen

A solução do problema de Büttiker, à primeira vista, é geral para sistemas onde ocorre quebra de coerência de fase no transporte eletrônico, podendo ser estendida para o problema interagente. Entretanto, a expressão para o transporte ainda é problemática, pois precisamos saber, nessa descrição, a forma funcional da distribuição eletrônica no eletrodo fictício e como ele se acopla à região de espalhamento. Uma alternativa é tratarmos o problema interagente diretamente, como feito por Meir-Wingreen [46]. Usando o formalismo de Keldysh [71] e considerando que todos os processos de interação ocorrem internamente na região de espalhamento, a corrente sobre um dos eletrodos, obtida por Meir-Wingreen [46], é dada por

$$I_l = \frac{ie}{\hbar} \int \frac{dE}{2\pi} \text{Tr} \left\{ \Gamma_l \left[G^< + (G - G^+) f_l \right] \right\} \quad (2.245)$$

Aqui, $G^<$ é a função de Green *lesser* [24, 72, 73], G (G^+) é a função de Green retardada (avançada) contendo a auto-energia de interação, dada por

$$G = \lim_{\eta \rightarrow 0^+} [(E + i\eta)\mathbb{1}_N - H - \Sigma_L - \Sigma_R - \Sigma_{ee}]^{-1}. \quad (2.246)$$

Importante notar que a densidade de estados pode ser escrita como $D(E) = \frac{-1}{\pi} \text{ImTr}\{G\} = \frac{-1}{\pi} \text{Tr}(G - G^\dagger)$. Temos que a corrente total [46]

$$I = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} \text{Tr}\left\{[G - G^\dagger] (f_L \Gamma_L - f_R \Gamma_R)\right\} + \text{Tr}\{G^< (\Gamma_L - \Gamma_R)\}. \quad (2.247)$$

Para o caso não interagente, temos explicitamente [71]

$$G^< = iG(f_L \Gamma_L + f_R \Gamma_R)G, \quad (2.248)$$

$$G - G^\dagger = -iG(\Gamma_L + \Gamma_R)G. \quad (2.249)$$

Note que, nas duas quantidades, não aparece a informação da distribuição dos reservatórios de elétrons, dependendo unicamente das quantidades internas do sistema. Reescrevendo a corrente total

$$I = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} \text{Tr}\left\{[G \Gamma_R G^\dagger \Gamma_L] (f_L - f_R)\right\}. \quad (2.250)$$

Obtemos, assim, a corrente de Fisher-Lee (FL), conforme esperado para o caso não interagente. Entretanto, para o caso interagente, devido à conservação do fluxo de carga, temos $I_L = -I_R$. Isso implica que a corrente total é $I = xI_L + xI_R - I_R = xI_L - (1 - x)I_R$, com x sendo uma variável arbitrária. A corrente total então terá a forma

$$I = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} \text{Tr}\left\{[G - G^\dagger] (xf_L \Gamma_L - (1 - x)f_R \Gamma_R)\right\} + \text{Tr}\{G^< (x\Gamma_L - (x - 1)\Gamma_R)\}, \quad (2.251)$$

e para o regime de acoplamento proporcional (AP), dado por $\Gamma_L = \lambda \Gamma_R$, em que λ é um fator de proporcionalidade arbitrário,

$$I = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} \text{Tr}\left\{[G - G^\dagger] (x\lambda f_L - (1 - x)f_R) \Gamma_R\right\} + \text{Tr}\{G^< (x\lambda - (x - 1)) \Gamma_R\}. \quad (2.252)$$

Com o intuito, então, de simplificar sem perda de generalidade, removemos a dependência explícita de $G^<$ impondo que,

$$x\lambda - (x - 1) = 0 \quad \rightarrow \quad x = \frac{1}{\lambda - 1} \quad \text{e} \quad 1 - x = \frac{\lambda}{\lambda - 1}. \quad (2.253)$$

Eliminando o segundo termo e reescrevendo a expressão, temos

$$I = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} \text{Tr} \left\{ [G - G^\dagger] \left[\frac{1}{1 + \lambda} (f_L \Gamma_L + f_R \Gamma_R) - f_R \Gamma_R \right] \right\}, \quad (2.254)$$

Se considerarmos ainda que os acoplamentos são inversíveis (o que nem sempre é verdade), podemos escrever a constante de proporcionalidade como $\lambda = \Gamma_L / \Gamma_R$. Simplificando, temos, por fim, a corrente total

$$I = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} \text{Tr} \left\{ \frac{\Gamma_L \Gamma_R}{\Gamma_L + \Gamma_R} [G - G^\dagger] \right\} (f_L - f_R). \quad (2.255)$$

Esta é a expressão de Meir-Wingreen. Note que essa expressão é para o caso de partículas não interagentes, pois não temos uma dependência direta com $G^<$. Entretanto, podemos computar a transmissão na presença de interação, pois $\Gamma_L \Gamma_R / (\Gamma_L + \Gamma_R)$ age como uma velocidade de grupo efetiva, e $G - G^\dagger$ está diretamente associada à densidade de estados e contém informações sobre a interação e-e. Assim, no $\text{Tr}\{\}$, temos a densidade de estados multiplicada pela velocidade de grupo associada a uma partícula saindo de um reservatório para o outro com energia $(f_L - f_R)$. No entanto, nessa expressão, não conseguimos diferenciar os processos inelásticos e elásticos, exceto pela presença do termo de auto-energia de interação, além do fato de estarmos restritos ao regime de acoplamento proporcional (AP).

Generalização da expressão de Meir-Wingreen

Para estudarmos os casos de interesse, é necessário, então, generalizar a expressão de Meir-Wingreen (MW) para o regime de acoplamento não proporcional, proposto por Amorim e Rocha [47]. Usando a expressão para corrente (2.247) e a conservação de carga, obtemos a seguinte condição

$$I_L + I_R = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} \text{Tr} \left\{ [G - G^\dagger] (f_L \Gamma_L + f_R \Gamma_R) \right\} + \text{Tr} \{ G^< (\Gamma_L - \Gamma_R) \} = 0. \quad (2.256)$$

Definimos o integrando como uma função que depende apenas da densidade de estados dos eletrodos direito e esquerdo e do Hamiltoniano, da seguinte forma:

$$A(\rho_L, \rho_R, H) = \text{Tr} \left\{ [G - G^\dagger] (f_L \Gamma_L + f_R \Gamma_R) \right\} + \text{Tr} \{ G^< (\Gamma_L - \Gamma_R) \}. \quad (2.257)$$

Para expressar as relações entre as funções de Green de maneira explícita, definimos o funcional,

$$\mathcal{L} = \int \frac{dE}{2\pi} A(\rho_L, \rho_R, H). \quad (2.258)$$

Fazendo uma variação em um dos parâmetros $\rho'_L(E)]_{mn} = \rho_L(E)]_{mn} + \delta_{m,i}\delta_{n,j}\delta\rho(E)$, levamos a uma variação no funcional

$$\delta\mathcal{L} = \int \frac{dE}{2\pi} \frac{\partial[A(\rho_L, \rho_R, H)]}{\partial[\rho_L(E)]_{ij}} \delta\rho(E). \quad (2.259)$$

Assumindo a conservação do fluxo de corrente e que a Eq. 2.256 é uma consequência da suposição de conservação do fluxo de carga em estado estacionário e, como tal, é válida independentemente do sistema para o qual é aplicada, temos [47]

$$\mathcal{L}(\rho_L, \rho_R, H) = \mathcal{L}(\rho'_L, \rho_R, H) = \delta\mathcal{L} \quad (2.260)$$

com isso, obtemos que

$$\left. \frac{\partial[A(\rho_L, \rho_R, H)]}{\partial[\rho_L(E)]_{ij}} \right|_{(\rho_L, \rho_R, H)} = 0. \quad (2.261)$$

Integrando com relação a $[\rho_L(E)]_{ij}$, obtemos [47]

$$A(\rho_L, \rho_R, H) = 0, \quad (2.262)$$

a relação mais importante, ao permitir relacionar diretamente os termos das funções de Green ($G^<, G, G^+$). Para escrever a relação de maneira mais conveniente, reescrevamos $G^<$ em termos da auto-energia de interação *lesser*, $\Sigma_{int}^<$. Note que até então não mencionamos qual tipo de interação está presente, pois estamos tratando do transporte de maneira geral, na presença de qualquer tipo de interação. Temos

$$G^< = G\Sigma^<G^+ = G(i\Gamma_R f_R + i\Gamma_L f_L + \Sigma_{int}^<)G^+ \quad (2.263)$$

e

$$G - G^+ = G(i\Gamma_R - i\Gamma_L - i\Gamma_{int}). \quad (2.264)$$

Aqui, $\Gamma_{int} = i(\Gamma_{int}^r - \Gamma_{int}^a)$. Reescrevendo $A(\rho_L, \rho_R, H)$,

$$\begin{aligned} & \text{Tr}\{i(\Gamma_R + \Gamma_L)G(\Gamma_R f_R + \Gamma_L f_L)G^+ - (\Gamma_R f_R + \Gamma_L f_L)G(\Gamma_R + \Gamma_L)G^+\} \\ & + \text{Tr}\{(\Gamma_R + \Gamma_L)G\Sigma_{int}^<G^+ - i(\Gamma_R f_R + \Gamma_L f_L)G\Gamma_{int}G^+\} = 0. \end{aligned} \quad (2.265)$$

Em particular, para o caso de simetria de reversão temporal, ou seja, $g_l = [g_l]^T$ e $G_l = [G_l]^T$, o primeiro termo é eliminado e ficamos com

$$\text{Tr} \left\{ (\Gamma_R + \Gamma_L) G \Sigma_{int}^< G^\dagger - i(\Gamma_R f_R + \Gamma_L f_L) G \Gamma_{int} G^\dagger \right\} = 0. \quad (2.266)$$

De maneira geral, expressar $\Sigma_{int}^<$, requer condições adicionais. Algumas propriedades que esperamos que ela obedeça são

$$\Sigma_{int}^< = -[\Sigma_{int}^<]^T \quad (2.267)$$

$$\Sigma_{int}^< \Big|_{f_L=f_L=f} = -(\Sigma_{int}^r - \Sigma_{int}^a) f, \quad (2.268)$$

$$\Sigma_{int}^< \Big|_{no-interagente} = 0, \quad (2.269)$$

$$\lim_{\Gamma_{L(R)} \rightarrow 0} \Sigma_{int}^< = -(\Sigma_{int}^r - \Sigma_{int}^a) f_{L(R)}, \quad (2.270)$$

$$\Sigma_{int}^> - \Sigma_{int}^< = \Sigma_{int}^r - \Sigma_{int}^a. \quad (2.271)$$

Assim, uma solução que satisfaz todas essas condições é:

$$\begin{aligned} \Sigma_{int}^< = & -(\sigma_{int}^r - \sigma_{int}^a) \frac{f_R + f_L}{2} + \\ & i [(\Gamma_L + \Gamma_R) G]^{-1} \left[\Gamma_L^{\frac{1}{2}} G \Gamma_{int} G^\dagger \Gamma_L^{\frac{1}{2}} - \Gamma_R^{\frac{1}{2}} G \Gamma_{int} G^\dagger \Gamma_R^{\frac{1}{2}} \right] \left[G^\dagger (\Gamma_L + \Gamma_R)^{\frac{1}{2}} \right]^{-1} \frac{f_R - f_L}{2}. \end{aligned} \quad (2.272)$$

Essa solução, que não necessariamente é única, pode ser vista como a generalização do teorema da flutuação fora do equilíbrio e, a princípio, pode ser usada para expressar a transmissão nesse regime. Em particular, em equilíbrio, o teorema da flutuação e resposta, dado pela relação, $\lim_{\Gamma_{L(R)} \rightarrow 0} \Sigma_{int}^< = -(\Sigma_{int}^r - \Sigma_{int}^a) f_{L(R)}$ é recuperado. Nesse caso,

$$[\Sigma_{int}^<]_{ii}(E) = f_{FD}(E) [\Gamma_{int}(E)]_{ii}. \quad (2.273)$$

O segundo termo é uma medida da taxa de dispersão de partículas com energia E devido ao nível i , que é proporcional à ocupação térmica (f_{FD}) e inversamente proporcional ao tempo de vida $[\Gamma_{int}(E)]_{ii} = \tau^{-1}$ de uma partícula de energia E devido ao nível i . No regime de não equilíbrio esses termos se tornam extremamente complicados. Uma maneira de simplificarmos é considerando processos de transporte descorrelacionados, ou seja, descartando correlações entre os processos de tunelamento coerentes e incoerentes. Nesse caso, a única diferença entre o

regime de equilíbrio (Eq. 2.273) e fora de equilíbrio é uma alteração na ocupação média.

Usando a Relação dada pela Eq. 2.266, obtemos uma solução mais conveniente,

$$[\Sigma_{int}^<]_{ij} = i \frac{[G(\Gamma_R f_R + \Gamma_L f_L)G^+]_{ij}}{[G(\Gamma_R + \Gamma_L)G^+]_{ij}} [\Gamma_{int}]_{ij}, \quad (2.274)$$

Observamos que na aproximação de transporte não correlacionado, o regime fora de equilíbrio é descrito por uma função de distribuição que considera não apenas as flutuações térmicas, mas também o desequilíbrio populacional causado pelos eventos de tunelamento em diferentes potenciais químicos. Assim, expressando a transmissão, e aqui especificando que a interação que ocorre é apenas de natureza elétron-elétron, $\Gamma_{int} = \Gamma_{ee}$ e $\Sigma_{int} = \Sigma_{ee}$. Obtemos,

$$T_{eff} = \text{Tr} \left\{ \Gamma_L G \Gamma_R G^+ \right\} + \sum_{i,j} \frac{[G \Gamma_R G^+]_{ij} + [G \Gamma_L G^+]_{ij}}{[G(\Gamma_R + \Gamma_L)G^+]_{ij}} [\Gamma_{ee}]_{ij}. \quad (2.275)$$

A corrente, então, é dada por

$$I = \frac{ie}{2\hbar} \int \frac{dE}{2\pi} T_{eff}(E) (f_L - f_R). \quad (2.276)$$

Aqui, a transmissão é dada por uma transmissão efetiva, onde no primeiro termo temos uma expressão análoga à expressão de Fisher-Lee, porém com a função de Green interagente e um termo de correção que depende diretamente do termo de interação elétron-elétron. Assim, devido à natureza coerente do termo de Fisher-Lee, podemos associar essa expressão de maneira análoga ao problema de Büttiker. Entretanto, aqui o segundo termo está relacionado ao tunelamento sequencial e, portanto, aos processos de tunelamento incoerente. Assim, temos uma expressão para a transmissão em sistemas interagentes, dada por

$$T_{int} = T_{eff} = T_{coerente} + T_{incoerente}, \quad (2.277)$$

com

$$T_{coerente} = \text{Tr} \left\{ \Gamma_L G \Gamma_R G^+ \right\}, \quad T_{incoerente} = \sum_{i,j} \frac{[G \Gamma_R G^+]_{ij} + [G \Gamma_L G^+]_{ij}}{[G(\Gamma_R + \Gamma_L)G^+]_{ij}} [\Gamma_{ee}]_{ij}. \quad (2.278)$$

De maneira direta para a corrente, $I = I_{coerente} + I_{incoerente}$. Temos, por fim, uma

expressão que contabiliza a influência da interação elétron-elétron e a quebra de coerência de fase devido à sua presença, além de estar conforme o modelo geral proposto por Büttiker, porém aqui considerando os estados interagentes quantizados.

Em particular, na presença de simetria de paridade, a transmissão assume a seguinte forma [47]

$$T_{\text{eff}} = \text{Tr}\left\{\Gamma_R G \Gamma_L G^\dagger\right\} + \text{Tr}\left[\Gamma_R G \Gamma_{ee} G^\dagger\right], \quad (2.279)$$

onde mais uma vez temos uma expressão análoga à de Fisher-Lee, porém a interação agora fazendo um papel análogo a um terceiro terminal, como mais uma vez proposto por Büttiker.

Por fim, uma vez que temos $\Sigma_{int}^<$, uma grandeza que podemos expressar diretamente é a densidade local de corrente, dada por:

$$I_{ij}(E) = \frac{2e^2}{\hbar} \text{Im}\{H_{ij} G_{ij}^<\}, \quad (2.280)$$

com

$$G^< = G \Sigma^< G^\dagger = G \Sigma_0^< G^\dagger + G \Sigma_{ee}^< G^\dagger. \quad (2.281)$$

aqui $\Sigma_0^< = \Gamma_L f_L + \Gamma_R f_R$.

2.10 Conclusão

Neste capítulo, descrevemos um conjunto de ferramentas que, combinadas, nos permitem estudar as propriedades eletrônicas e de transporte em nanodispositivos na presença de interação elétron-elétron. Utilizando o modelo tight-binding para descrever o Hamiltoniano do sistema na região de espalhamento do dispositivo e definindo a célula unitária com a simetria dos eletrodos na representação de orbitais atômicos localizados. Após a definição da simetria dos eletrodos, empregamos o método Recursivo das Funções de Green para obter as auto-energias dos eletrodos, resultando na função de Green completa para um sistema acoplado a dois terminais.

Na ausência de interação, descrevemos os observáveis de transporte, como transmissão e corrente elétrica na presença de uma diferença de potencial. Por outro lado, para o sistema na presença de interação, precisamos primeiro escolher os estados moleculares de interesse que conterão a interação. Após a definição dos

estados, realizamos uma projeção sobre o sistema na base dos estados moleculares de interesse, resultando em um Hamiltoniano com a estrutura do modelo de Impurezas de Anderson multi-níveis. Que, a princípio, para obter a auto-energia de interação pode se escolher o *solver* de interesse, aqui, escolhemos utilizar da representação de pseudo-partícula para tal feito. Nessa descrição, obtemos um Hamiltoniano bilinear no termo de interação, enquanto o acoplamento com os eletrodos se torna mais complicado, consistindo nos elementos fora da diagonal do Hamiltoniano. Realizamos uma expansão diagramática no acoplamento e descartamos as correções de vértices (conhecida como aproximação de não cruzamento) para obter a auto-energia de interação na representação de pseudo-partícula. Em seguida, realizamos a projeção inversa para obter a auto-energia de interação na representação de orbitais atômicos localizados, permitindo assim obter a função de Green integrante completa.

Entretanto, nem a expressão de Fisher-Lee para o transporte, nem a de Meir-Wingreen, que é válida apenas para o regime de acoplamento proporcional, são válidas para o regime de interesse. Para lidar com isso, foi necessário generalizar a expressão de Meir-Wingreen para o caso interagente no regime de acoplamento não proporcional, com algumas aproximações. Isso nos permite analisar os observáveis de transporte para os sistemas de interesse, que serão discutidos nos capítulos seguintes.

Por fim, criamos uma biblioteca utilizando a linguagem de programação Python, seguindo a estrutura lógica dos métodos apresentados aqui, para cada um dos modelos discutidos. A biblioteca está integralmente apresentada no Apêndice.

Capítulo 3

Transistor de Elétron Único Baseado em Nanofitas de Grafeno

Estruturas baseadas em carbono, como os nanotubos [17], e, mais recentemente, o grafeno [18, 74, 19], têm atraído grande atenção da comunidade científica tanto do ponto de vista da ciência básica [75, 76] quanto de possíveis aplicações em fotônica [77, 78] e eletrônica [79, 80, 81]. Como já discutido nos capítulos anteriores, no caso particular do grafeno, apresenta-se como um material extremamente forte com propriedades elétricas [19, 79] e de transporte térmico incomparáveis [82].

Uma das desvantagens, entretanto, é a falta de um *gap* eletrônico, o que dificulta uma série de aplicações em eletrônica. Nesse sentido, avanços experimentais permitiram a fabricação de nanofitas de grafeno (NR) derivadas quimicamente, que são atômicamente precisas [32, 33]. Essas estruturas quase unidimensionais são tipicamente semicondutoras, com um *band gap* que pode ser controlado pela largura e família da fita [27]. Outra questão difícil de superar é a caracterização do acoplamento entre o dispositivo baseado em grafeno e os contatos. A barreira Schottky resultante pode ser um problema para dispositivos que geralmente usam metal para fabricação [37].

Neste sentido, a ideia de sistemas constituídos exclusivamente por carbono está se tornando cada vez mais comum [39, 83, 7, 28, 29, 30, 31, 38]. Seguindo essa linha, uma estrutura constituída por junções de nanotubos de carbono e uma NR foi teoricamente proposta por Santos *et al.* [39]. Nessa proposta, os autores consideraram uma estrutura fortemente acoplada, sem interação, e usaram um nanotubo descompactado para imitar as junções. Recentemente, Jian Zhang *et al.* [7] utilizaram essa ideia para construir uma arquitetura de dispositivo multi-*gate* com nanotubos de carbono de parede única (SWNTs) como eletrodos [7]. Esta configuração SWNT-NR-SWNT retém muitas das propriedades dos transistores de elétron único, e os autores observaram o bloqueio de Coulomb e os diamantes de Coulomb correspondentes [7]. Embora a presença deste efeito tenha

sido observada anteriormente em estruturas baseadas em grafeno [84, 85], neste trabalho, o acoplamento com o eletrodo foi muito melhor controlado.

Para descrever completamente esses dispositivos, uma descrição teórica adequada precisa considerar a presença de interações elétron-elétron através da repulsão de Coulomb. Como já mencionado anteriormente, isso normalmente é negligenciado [39] porque envolve a solução de um problema de muitos corpos. Foram desenvolvidas técnicas para resolver esse problema, como, por exemplo, a aproximação de campo médio [86], o grupo de renormalização funcional (fRG) [87], o grupo de renormalização numérica (NRG) [88], e as aproximações de não cruzamento (NCA) [60, 89] e de cruzamento único (OCA) [60, 61]. No entanto, elas normalmente se concentram em modelos Hamiltonianos para impurezas locais, contendo um pequeno número de orbitais.

Este não é o caso aqui, já que o comportamento do sistema semelhante a um ponto quântico surge do fraco acoplamento entre o nanotubo e a nanofita, e os estados relevantes podem ser espacialmente deslocalizados ao longo de toda a fita. Por isso, nesta parte da dissertação, simulamos um transistor de elétron único baseado em nanotubos de carbono como eletrodos, acoplado a uma nanofita de grafeno, semelhante à que foi obtida experimentalmente [7]. Aqui, buscamos entender como a interação elétron-elétron altera as propriedades eletrônicas e de transporte deste dispositivo. Para tanto, realizamos cálculos usando o método da função de Green fora de equilíbrio [90], descrevendo os estados de interação por meio de um modelo de impureza de Anderson multi-orbital (AIM) juntamente com a chamada aproximação de não cruzamento (NCA) [64] e uma generalização para a fórmula de Meir-Wingreen para transporte incoerente [47]. Estes métodos foram todos descritos em detalhes no Capítulo 2. Através da combinação deles, descrevemos corretamente os fenômenos de transporte para sistemas interagentes, incluindo diamantes de Coulomb neste problema multi-orbital. Mostramos que somos capazes de explicar os diamantes de Coulomb e, o mais importante, que as contribuições incoerentes para a corrente dominam o transporte para uma ampla gama de potenciais externos e tensões de *gate* aplicados.

3.1 Metodologia

A configuração utilizada ao longo do capítulo é apresentada na Figura 3.1, seguindo o arranjo observado nos experimentos [7]. Colocamos dois SWNTs (metálicos) com borda zig-zag semi-infinitos (18,0) separados por uma distância

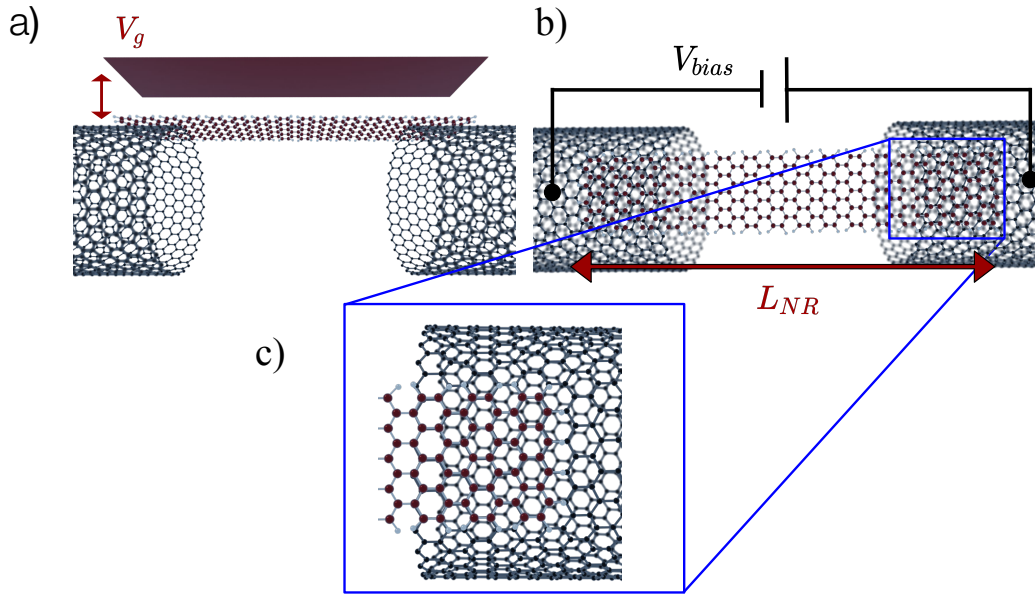


Figura 3.1: Lado (a), topo (b) e (c), um zoom da sobreposição entre as estruturas, vistas do dispositivo usado em nossos cálculos.: SWNTs como eletrodos direito e esquerdo, acoplados por uma NR para formar uma junção SWNT-NR-SWNT. O acoplamento com os eletrodos ocorre de maneira comensurada através de três células unitárias na NR de cada lado.

de aproximadamente $20 \text{ \AA}$.¹ Estes constituem os eletrodos esquerdo e direito. Em seguida, tomamos um segmento finito de uma nanofita de grafeno de 9 armchair, que consiste em $L_{NR} = 16$ células unitárias acopladas entre si. Fazemos com que as bordas da NR se sobreponham parcialmente a cada um dos eletrodos. Note que as estruturas consideradas aqui são comensuráveis; portanto, consideramos um empilhamento do tipo AA entre a nanofita e o nanotubo, que é mantido unido mediante um acoplamento fraco do tipo van der Waals.

O sistema é descrito usando um Hamiltoniano de tight-binding com um único orbital por sítio representando os orbitais p_z em cada átomo de carbono [48]. Dividimos nosso sistema em três regiões conforme descrito pela primeira vez por Caroli *et al* [54, 91], ou seja, eletrodos semi-infinitos esquerdo e direito e uma região de espalhamento. Este último consiste na NR e nos segmentos finitos de nanotubos.

O Hamiltoniano da região de espalhamento $\hat{H}_S$ é dividido novamente em três segmentos, $\hat{H}_\alpha$ e $\hat{H}_\beta$, que possuem a estrutura de um SWNT finito, e $\hat{H}_{NR}$, que

¹O caráter metálico do nanotubo pode ser observado na autoenergia de superfície apresentada na Figura 2.18.

representa nossa NR:

$$\hat{H}_S = \hat{H}_\alpha + \hat{H}_{NR} + \hat{H}_\beta + \hat{\gamma}_\alpha + \hat{\gamma}_\beta, \quad (3.1)$$

onde $\hat{\gamma}_{\alpha/\beta} = \gamma_{\alpha/\beta} \hat{I}_{\alpha/\beta} \otimes \hat{I}_{NR}$ é a matriz de acoplamento de van der Waals entre o segmento de nanotubo correspondente e a NR.

Os parâmetros de tight-binding do sistema foram escolhidos de acordo com a literatura [49]. Assim, tomamos a energia local $\epsilon = 0$ eV, o acoplamento entre primeiros vizinhos $\gamma_{i,j} = \gamma_0 = 2,7$ eV, e o termo de acoplamento entre orbitais na borda armchair γ_b na NR são ligeiramente alterados para $\gamma_b = \gamma_0 + 0,12\gamma_0$, resultante da hidrogenação. Por último, tomamos $\gamma_{\alpha/\beta} = 0,15$ eV. Também podemos aplicar uma tensão de *gate* V_g deslocando os níveis da NR e um potencial externo V_{bias} entre os eletrodos, como pode ser visto na Fig. 3.1(b).

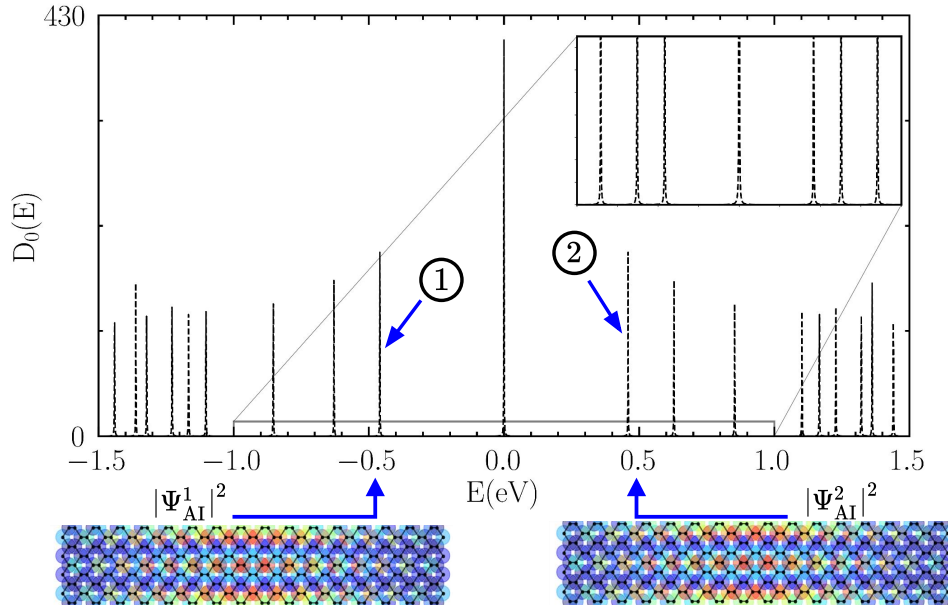


Figura 3.2: **Densidade de Estados para a NR desacoplada:** Densidade de Estados para a NR desacoplada (curva tracejada preta) e o módulo quadrado das funções de onda para os dois primeiros estados moleculares simétricos em relação ao nível de Fermi. $E_1 = -0.45$ eV, $E_2 = 0.45$ eV.

Em equilíbrio, a função de Green retardada não interagente, é dada por:

$$\lim_{\eta \rightarrow 0^+} G_0 = [(E + i\eta)I - H_S - \Sigma_L - \Sigma_R]^{-1}, \quad (3.2)$$

É a quantidade-chave, onde Σ_L (Σ_R) é a autoenergia para o eletrodo esquerdo (direito). Dada a estrutura cristalina dos eletrodos e utilizando a equação de Dyson,

as autoenergias podem ser obtidas através de um método recursivo [55].

A densidade de estados projetada nos estados orbitais da NR é dada por:

$$D_0(E) = -\frac{1}{\pi} \text{Tr} [Im\{G_0\}_{NR}]. \quad (3.3)$$

Ao mesmo tempo, a corrente não interegente é dada pela fórmula de Buttiker-Landauer:

$$I_0 = \frac{2e}{\hbar} \int T_0(E)(f_R - f_L)dE, \quad (3.4)$$

onde

$$T_0 = \text{Tr} [\Gamma_R G_0^\dagger \Gamma_L G_0], \quad (3.5)$$

é a transmissão, incluindo apenas os processos coerentes, e

$$\Gamma_{L/R} = i(\Sigma_{L/R} - \Sigma_{L/R}^\dagger). \quad (3.6)$$

No caso de um segmento isolado da NR, obtêm-se estados moleculares quantizados, como observamos na Figura 3.2. Além disso, os orbitais moleculares próximos ao nível de Fermi estão deslocalizados em toda a estrutura. Como veremos melhor, o acoplamento entre NR e os SWNTs é fraco, portanto o efeito geral do acoplamento aos eletrodos é um pequeno alargamento dos estados. Isto indica que ele retém a maioria do caráter molecular do sistema desacoplado. Isso também significa que a aplicação de um potencial externo V_{bias} e de tensões de *gate* podem levar a mudanças significativas no estado de carga do sistema. Nesse caso, a interação elétron-elétron pode ter um impacto significativo, como já mencionado.

Em princípio, a inclusão de interações exigiria um cálculo de muitos corpos considerando o Hamiltoniano completo, mais notavelmente, toda a região de espalhamento. Aqui, para resolver esse problema, primeiro definimos a base dos estados moleculares de impureza de Anderson, com N_{AI} estados. Este é um subconjunto dos estados moleculares da NR que conterão interações.

$$U_\psi = (\psi_{AI}^1, \psi_{AI}^2, \dots, \psi_{AI}^n)^T. \quad (3.7)$$

A interação elétron-elétron é incluída aqui considerando um modelo de impureza multi-orbital de Anderson para o ψ_{AI}^i selecionado, conforme dado por:

$$\hat{H}_{NR} |\psi_{AI}^i\rangle = \epsilon_{AI}^i |\psi_{AI}^i\rangle. \quad (3.8)$$

Portanto, adicionamos

$$\hat{H}^{ee} = \sum_{i,j,k,l}^{N_{AI}} U_{i,j,k,l} \hat{f}_{i,\sigma}^\dagger \hat{f}_{i,\sigma'}^\dagger \hat{f}_{l,\sigma'} \hat{f}_{k,\sigma}, \quad (3.9)$$

para o Hamiltoniano total. Os operadores de criação (aniquilação) $\hat{f}_{i,\sigma}^\dagger$ ($\hat{f}_{i,\sigma}$) atuam nos orbitais moleculares da AI, onde σ é a projeção do spin ($\sigma = +$ ou $-$). Aqui, por simplicidade, tomamos

$$U_{i,j,k,l} = \begin{cases} U, & \text{se } l = k = j = i \\ 0, & \text{caso contrário} \end{cases} \quad (3.10)$$

com a interação $U = 0,45\text{eV}$. Estamos considerando que os orbitais são ortogonais entre si, portanto, podemos projetar diretamente a função de Green da região de espalhamento sobre os estados AI

$$G_{AI} = \begin{pmatrix} 0_\alpha \\ U_\psi \\ 0_\beta \end{pmatrix}^\dagger \left[(E + i\eta)I - H_S - \Sigma_L - \Sigma_R - \Sigma_{ee} \right]^{-1} \begin{pmatrix} 0_\alpha \\ U_\psi \\ 0_\beta \end{pmatrix}. \quad (3.11)$$

que inclui a interação elétron-elétron através da autoenergia Σ_{ee} . Ao mesmo tempo, definimos a função de Green para os estados da IA

$$G_{AI} = [EI_{N_{AI}} - \epsilon_{AI} - \bar{\Delta}_{AI} - \bar{\sigma}_{ee}]^{-1}, \quad (3.12)$$

com a chamada função de hibridização $\bar{\Delta}_{AI}$, dada por:

$$\bar{\Delta}_{AI} = EI_{N_{AI}} - \epsilon_{AI} - \bar{\sigma}_{ee} - G_{AI}^{-1}. \quad (3.13)$$

Devido ao fato de o acoplamento entre a região de espalhamento e os eletrodos ser dado por uma interação do tipo van der Waals, os estados da AI são fracamente acoplados aos eletrodos, e a função de hibridização pode ser utilizada no solucionador de impurezas. Isso define o conjunto de equações para o campo médio dinâmico que pode ser iterado de forma autoconsistente. Em nosso código, usamos a chamada aproximação não cruzada (NCA) para obter a autoenergia para a interação elétron-elétron ($\bar{\sigma}_{ee}$) na base da AI. Podemos então realizar a projeção

inversa para obter a autoenergia na representação dos orbitais atômicos,

$$\Sigma_{ee} = U_\psi \bar{\sigma}_{ee} U_\psi^T. \quad (3.14)$$

Obtemos então a função de Green retardado que descreve o problema interagente

$$G = [G_0^{-1} - \Sigma_{ee}]^{-1}, \quad (3.15)$$

e podemos calcular as propriedades eletrônicas e de transporte para o problema interagente.

Aqui, consideramos a expressão de Fisher-Lee para o regime de acoplamento não proporcional [47], como apresentada na Subseção 2.9.3, assumindo a forma

$$I = \frac{ie}{\hbar} \int \frac{dE}{2\pi} \left\{ \text{Tr} [\Gamma_R G^\dagger \Gamma_L G] + \text{Tr} [\Gamma_R G^\dagger \Gamma_{ee} G] \right\} (f_R - f_L), \quad (3.16)$$

com $\Gamma_{ee} = i(\Sigma_{ee} - \Sigma_{ee}^\dagger)$, podemos identificar a transmissão efetiva por um termo coerente e um termo incoerente. Assim, a corrente total é dada por $I = I_{\text{coh}} + I_{\text{incoh}}$

Para abordar a corrente local, podemos escrever o termo fora de equilíbrio de $G^<$ como a soma de um termo interagente e um não interagente.

$$G^< = G^r \Sigma^< G^a = G^r \Sigma_0^< G^a + G^r \Sigma_{ee}^< G^a,$$

com $\Sigma_0^< = \Gamma_L f_L + \Gamma_R f_R$ dado pelo formalismo Buttiker-Landauer, e o termo de interação $[\Sigma_{ee}]_{ij}^<$, dado pela aproximação [47]:

$$[\Sigma_{ee}]_{ij}^< = i \frac{[G(\Gamma_R f_R + \Gamma_L f_L)G^\dagger]_{ij}}{[G(\Gamma_R + \Gamma_L)G^\dagger]_{ij}} [\Gamma_{ee}]_{ij}. \quad (3.17)$$

Obtemos a generalização da corrente local:

$$I_{ij}(E) = \frac{2e^2}{\hbar} \text{Im} \{ H_{ij} G_{ij}^< \}, \quad (3.18)$$

novamente, aqui temos $I_{ij}(E) = I_{ij}^{\text{coh}}(E) + I_{ij}^{\text{inc}}(E)$, onde um termo representa a corrente local puramente devido à interação elétron-elétron, e o outro devido ao transporte coerente.

3.2 Resultados

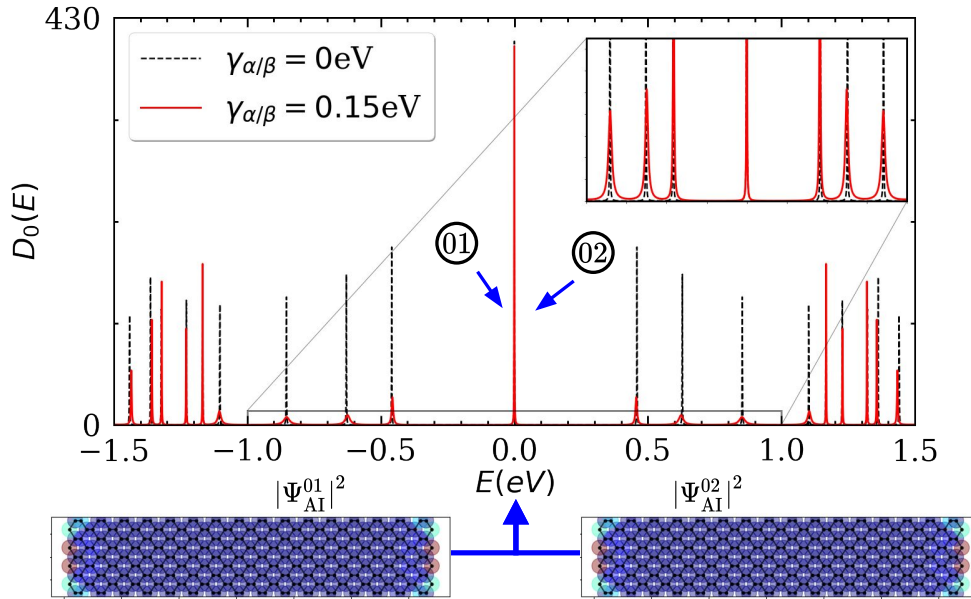


Figura 3.3: **Densidade de Estados para a NR:** Densidade de estados para a NR desacoplada (curva tracejada preta), densidade de estados para a NR acoplada (curva sólida vermelha) e o módulo quadrado das funções de onda para os dois estados moleculares degenerados no nível de Fermi. $E_{01} \approx E_{02} \approx 0$ eV.

A escolha dos estados de impurezas moleculares, que conterão a interação elétron-elétron, é inicialmente arbitrária e depende apenas de argumentos físicos, podendo variar para cada sistema. No caso específico da junção SWNT-NR-SWNT apresentada neste capítulo, discutimos também nesta seção como essa escolha foi feita.

Observe que na densidade de estados, apresentada na Fig.3.2 e Fig.3.3, no caso em que o segmento da nanofita está desacoplado dos eletrodos ($\gamma_{\alpha/\beta} = 0$ eV), os estados moleculares são quantizados. Quando consideramos $\gamma_{\alpha/\beta} = 0.15$ eV, tal que $\gamma_{\alpha/\beta} < \gamma_0$, simulando um acoplamento do tipo van der Waals, observamos um pequeno alargamento dos estados. Com exceção dos estados degenerados no nível de Fermi, que permanecem desacoplados e localizados espacialmente na borda do zig-zag (borda transversal) da fita. Isso indica que a fita retém a maioria do caráter molecular do sistema desacoplado.

Assim, inicialmente, não é necessário considerar os estados degenerados para nossos cálculos. No entanto, para estabilizar a estrutura, conforme discutido na literatura [49], consideramos os efeitos da hidrogenação na borda do zig-zag, o que implica em considerar a interação elétron-elétron com um potencial Coulombiano

na borda, $U_z = 2$. Para isso, selecionamos os estados de borda (Ver Fig. 3.3) e definimos a base de projeção,

$$U_{0\psi} = (\psi_{AI}^{01}, \psi_{AI}^{02})^T. \quad (3.19)$$

Utilizando a metodologia já discutida anteriormente, calculamos a função de hibridização na representação de partícula única, na ausência de interação, conforme mostrado na Fig. 3.4, onde $-\frac{1}{\pi}Im[\bar{\Delta}_{AI}]$ representa a função espectral dos estados da borda. Observe que para $-2 < E < 2$, os estados de borda estão aproxima-

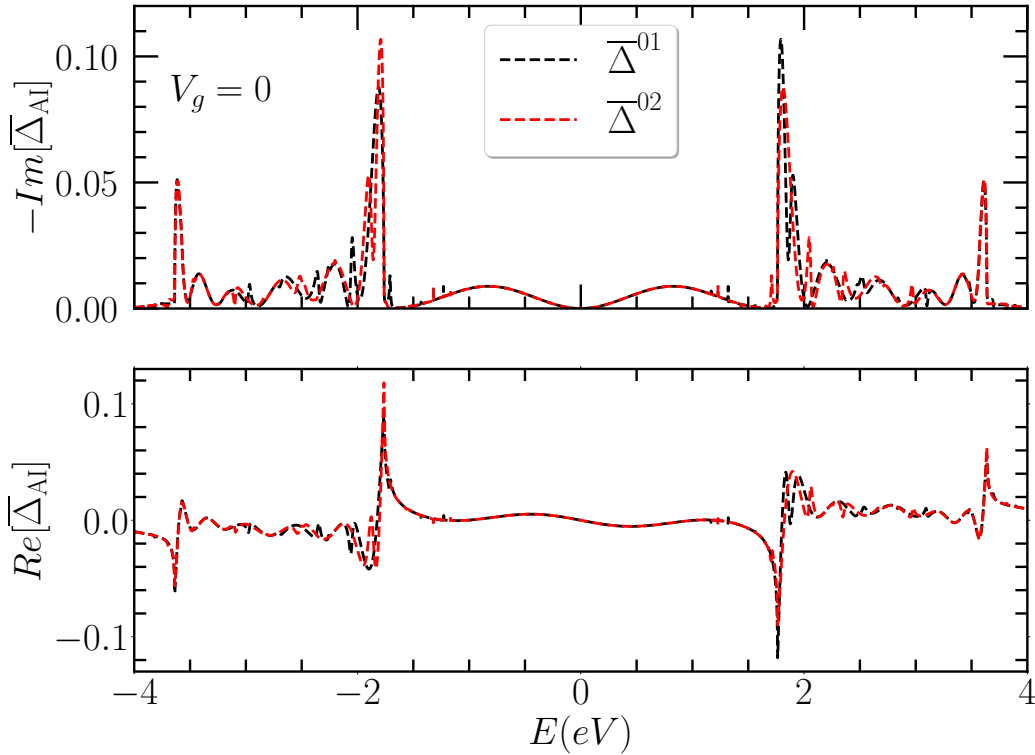


Figura 3.4: **Função de Hibridização:** a) A parte real e b) o negativo da parte imaginária da função de hibridização para as impurezas de borda na ausência de interação. A curva tracejada preta refere-se a ψ_{AI}^{01} e a curva tracejada vermelha refere-se a ψ_{AI}^{02} .

mente desacoplados dos demais estados moleculares. Olhando para a densidade de estados na ausência do *gate* para os estados de impureza molecular da borda não interagentes na Fig. 3.5, como esperado, eles são estados moleculares desacoplados da NR. Entretanto, quando ativamos a interação devido à hidrogenação das bordas, vemos que a densidade de estados é ligeiramente comprimida, o que implica que os estados passam a ser "mais" desacoplados em relação à ausência da hidrogenação das bordas.

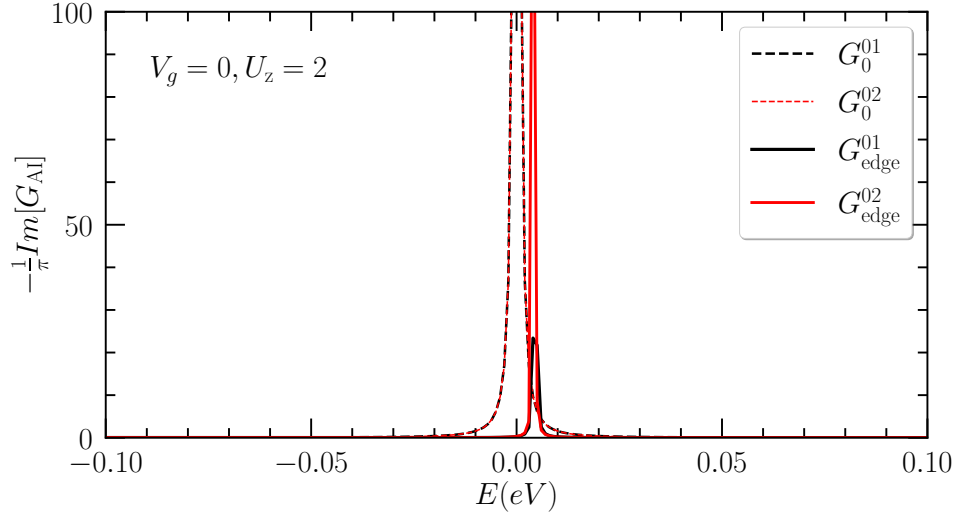


Figura 3.5: **Densidade de estados para impurezas de borda:** Densidade de estados para os estados de borda no nível de Fermi. Para o caso não interagente (curva tracejada) e para o caso interagente (curva sólida) com $U_z = 2$. A curva preta refere-se a ψ_{AI}^{01} e a curva vermelha a ψ_{AI}^{02} .

Para escolher os estados de impurezas moleculares ψ^i_{AI} de maneira apropriada, usamos o seguinte critério: selecionamos um intervalo de energia e, com base nele, escolhemos os estados. Na ausência de V_{bias} e devido à natureza quantizada, ao fixarmos um estado com um valor E_i , ao considerarmos a interação, a solução será uma boa aproximação para o problema completo para valores em torno de E_i . Entretanto, na presença de V_{bias} , ao observarmos a expressão para a corrente elétrica dada pela Eq. 3.16, temos uma integral de $T(E)$ no intervalo de $-V_{bias}/2$ a $V_{bias}/2$, o que significa que todos os estados dentro desse intervalo contribuem para a corrente. Portanto, precisamos considerar todos eles para uma descrição física precisa. Analogamente, na presença de V_g , os estados são deslocados para $E_i + V_g$, então é necessário considerar os estados que entram no intervalo de $-V_g$ a V_g .

Fixando então o intervalo de interesse como $-0.5 < V_{bias} < 0.5$ e $-0.5 < V_g < 0.5$. Considerar todos os estados entre $-0.5 < E < 0.5$ nos oferecer uma boa aproximação nesses intervalos. Finalmente, podemos desconsiderar os estados degenerados no nível de Fermi e suas contribuições devido à hidrogenação da borda para o intervalo de energia de interesse.

Como nos experimentos [7], fixamos a temperatura do sistema em aproximadamente 9K. Como já foi apontado, uma vez que assumimos que o acoplamento na junção é fraco e o sistema retém muito do caráter molecular (ver Fig. 3.3). Assim,

selecionamos um subconjunto de orbitais moleculares, para os quais incluímos interações elétron-elétron.

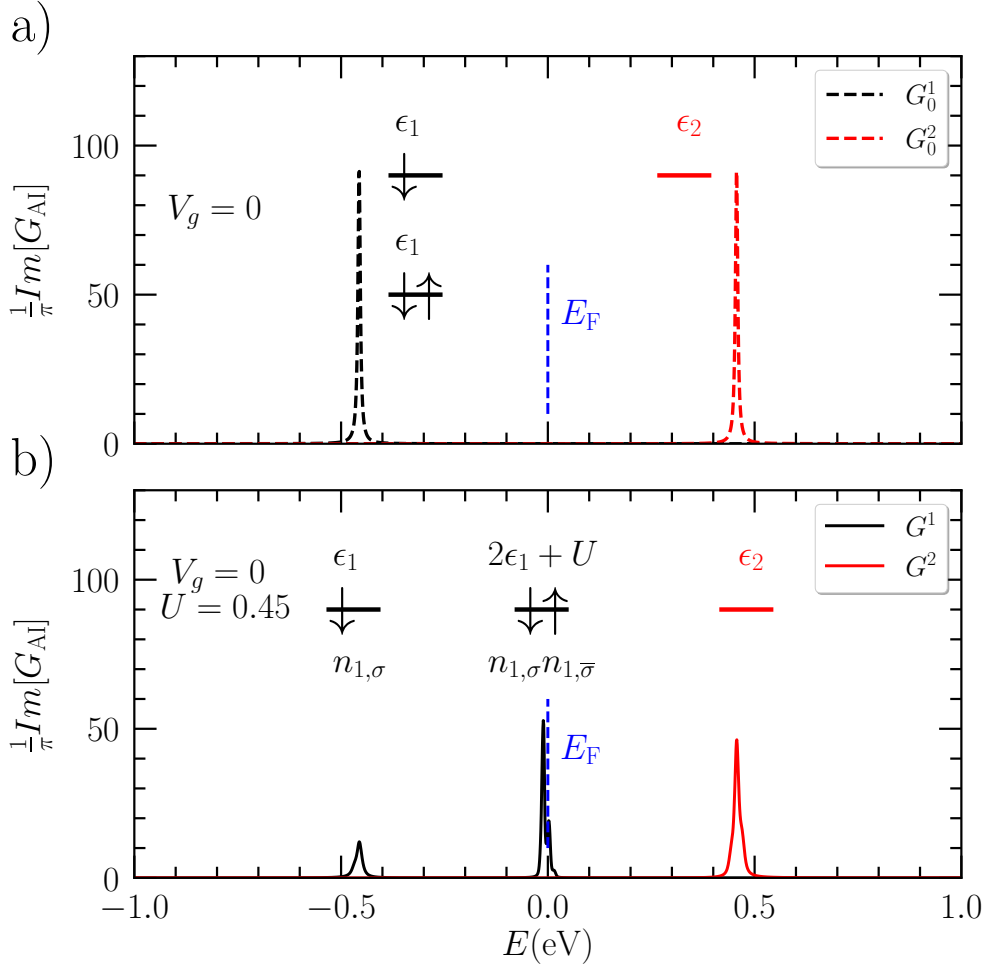


Figura 3.6: **Densidade de estados para impurezas de Anderson:** Densidade de estados para os dois primeiros estados próximos ao nível de Fermi. Na ausência de interação *a*), temos o estado duplamente ocupado (curva tracejada preta) e o estado com ocupação zero (curva tracejada vermelha). Na presença de interação ($U = 0.45$) *b*), representado na figura por curvas sólidas, observamos o estado com ocupação zero em $E = 0.45$ (curva sólida vermelha), o estado com única ocupação em $E = -0.45$ (curva sólida preta) e o estado duplamente ocupado em $E = 0$ (curva sólida preta).

Aqui consideramos os dois estados que estão simetricamente posicionados em relação ao nível de Fermi ($E_1 = -0.45$ eV, $E_2 = 0.45$ eV). Definimos G^i e G_0^i como o i -ésimo elemento diagonal de G_{AI} com e sem interação, respectivamente.

Olhando inicialmente para a densidade de estados na ausência do *gate* para os estados de impureza molecular não interagentes na Figura 3.6(a), observamos, como mencionado anteriormente, são estados moleculares desacoplados da NR

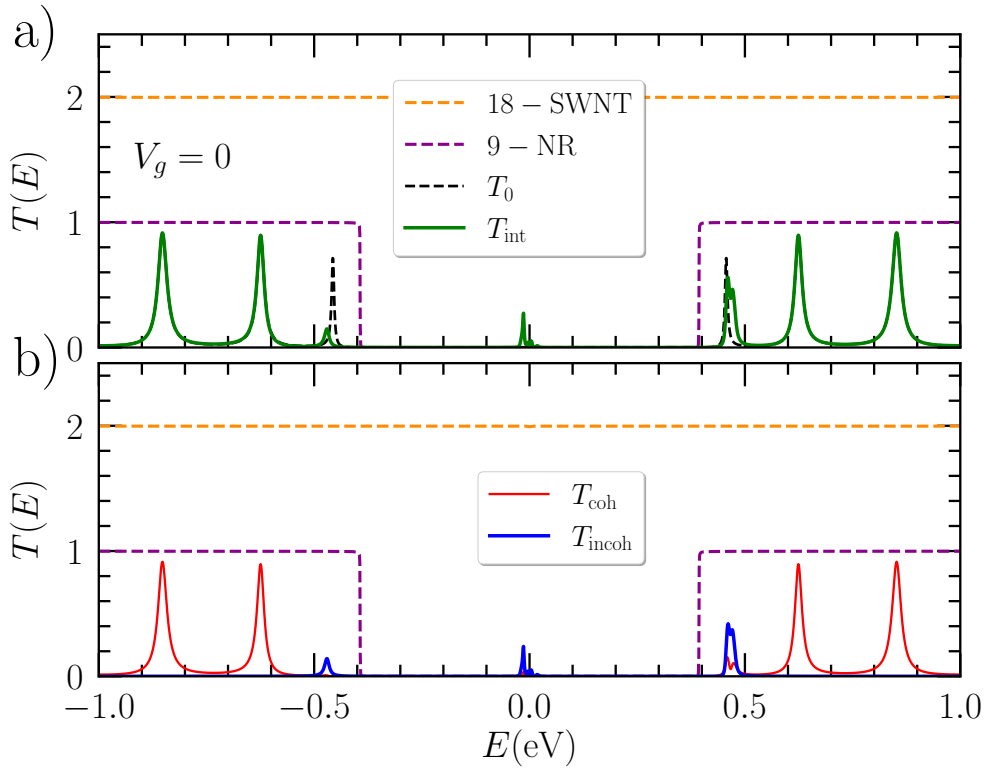


Figura 3.7: **Transmissão em função da energia:** A transmissão para o nanotubo (curva tracejada laranja) e para a nanofita (curva tracejada roxa) estão presentes como guias para comparação. Na configuração não interagente SWNT-NR-SWNT (curva sólida preta). A transmissão interagente (curva sólida verde), coerente (curva sólida vermelha) e incoerente (curva sólida azul) são apresentadas para a configuração SWNT-NR-SWNT interagente com potencial Coulombiano $U = 0.45\text{eV}$.

com ligeiro alargamento. Além disso, o nível logo abaixo de E_F permanece duplamente ocupado. Quando ativamos a interação (ver Figura 3.6(b)), notamos que esta configuração tem outro valor energético. O nível duplamente ocupado do primeiro estado de partícula é deslocado em U e agora está muito próximo do nível de Fermi.²

Calculamos a influência do estado carregado na transmissão. Como mostrado na Figura 3.7(a), a probabilidade de transmissão próxima ao nível de Fermi surge devido ao estado duplamente ocupado no primeiro orbital molecular, que não está presente no caso não interagente. Além disso, quando olhamos para as diferentes contribuições para este pico próximo do nível de Fermi na Figura 3.7(b), notamos que ele provém principalmente do termo não coerente na transmissão. É

²Para o segundo estado de partícula única, ϵ_2 o mesmo irá acontecer de modo que a energia do estado será dada por $\epsilon_2 + U$ de maneira análoga.

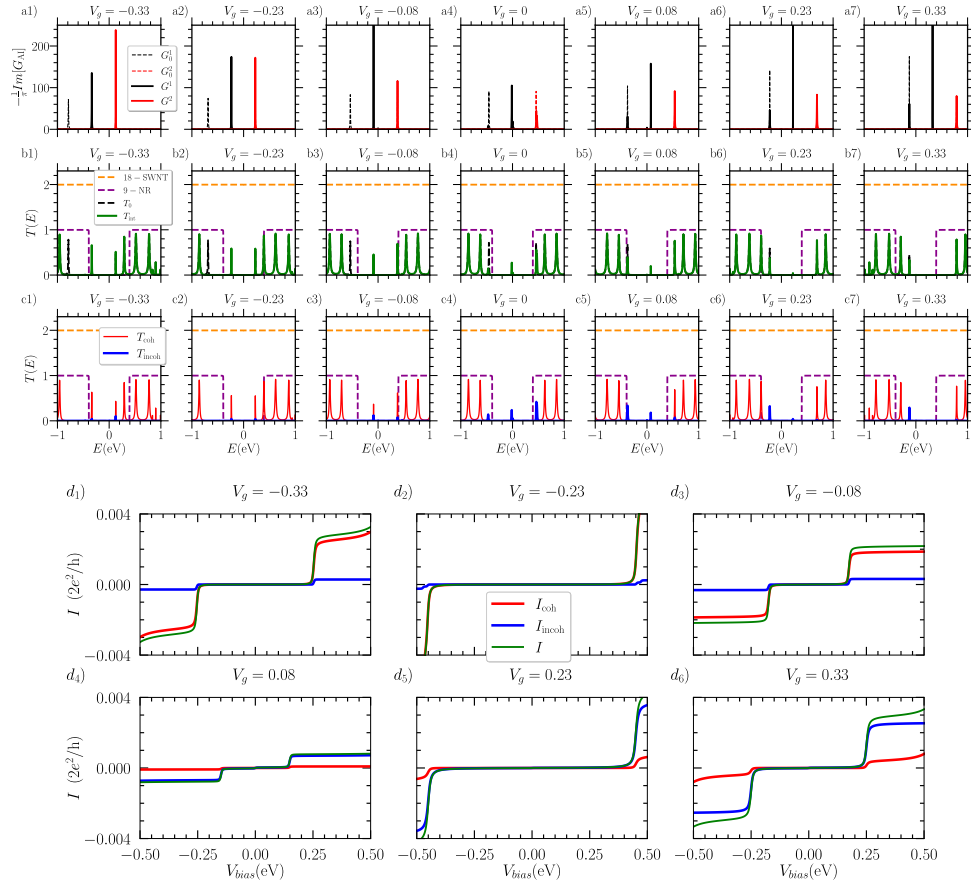


Figura 3.8: a) Densidade de estados, b) Transmissão efetiva, c) Transmissão coerente e transmissão incoerente para múltiplos *gates*, e d) Corrente elétrica: a) Temos a densidade de estados para os dois primeiros estados próximos ao nível de Fermi para valores de *gate* crescentes (diferentes colunas). Para o caso sem interação, a curva tracejada preta refere-se a ψ_{AI}^1 com dupla ocupação e a curva tracejada vermelha refere-se a ψ_{AI}^2 com ocupação zero. Na presença de interação com $U = 0.45\text{eV}$, a curva sólida preta refere-se a ψ_{AI}^1 para $V_g < 0$ com dupla ocupação e para $V_g \geq 0$ com quebra de degenerescência e assumindo ocupação única e ocupação dupla para diferentes valores de energia. A curva sólida vermelha refere-se a ψ_{AI}^2 com ocupação zero. b) e c) Temos a transmissão em função da energia para diferentes valores de *gate* (diferentes colunas), com a transmissão para o nanotubo (curva tracejada laranja) e nanofita (curva tracejada roxa) presentes como guias para comparação. Na configuração não interagente SWNT-NR-SWNT (curva sólida preta) e a transmissão efetiva (curva sólida verde) para interagente SWNT-NR-SWNT com potencial Coulombiano $U = 0.45\text{ eV}$. c) Transmissão coerente (curva sólida vermelha) e a transmissão incoerente (curva sólida azul) para interagente SWNT-NR-SWNT com potencial Coulombiano $U = 0.45\text{ eV}$. d) Comparação entre os termos de corrente efetiva (curva verde), o termo de corrente coerente (curva vermelha) e o termo de corrente incoerente (linha azul), para $U = 0.45\text{ eV}$ em função de V_{bias} para diferentes valores de *gate*.

importante salientar aqui que, embora o termo coerente no caso interagente tenha a mesma forma funcional que a relação de Fisher-Lee para o caso não interagente, consideramos a função de Green com interação completa e, portanto, os efeitos de muitos corpos então também presente neste termo. Isto significa que, no regime aqui considerado - acima da temperatura de Kondo - temos que considerar contribuições de tunelamento coerentes e sequenciais.

Além disso, ao considerar um *gate* externo, é possível controlar as posições dos diferentes níveis. Isso é mostrado na Figura 3.8. Para a função espectral projetada nos dois AIs (Figura 3.8(a)) em valores de *gate* negativos, o comportamento do sistema pode ser descrito por uma abordagem de campo médio para a interação, o nível de impureza completamente preenchido é deslocado para cima em U , e esta é a única contribuição para a transmissão proveniente desse nível (ver Figura 3.8(b)). Isto é corroborado pelo fato de que a maior contribuição para a transmissão de zero-bias vem do termo não coerente, como mostrado na Figura 3.8(c). Para valores positivos, à medida que a tensão aumenta, o nível duplamente ocupado é inicialmente fixado em E_F , mas é posteriormente suprimido na transmissão, portanto, não dá nenhuma contribuição para as propriedades gerais de transporte do sistema. Na região intermediária, o pico de transmissão só pode ser contabilizado se o termo incoerente for incluído, como já discutido.

Também podemos aplicar uma bias externa ao dispositivo e desviar o foco do caso estritamente à zero-bias. Aqui, isso é feito considerando que as autoenergias de interação e as funções de Green não mudam à medida que a bias é aplicada. Assim, a corrente é dada pela integral na Eq. 3.16 dentro da janela de bias apropriada, e a corrente local pela Eq. 3.18. Primeiro, olhamos para a corrente local mostrada na Fig. 3.9 para $E = 0$ eV, $V_g = 0$ eV e $V_{bias} = 0.02$ eV, o que nos permite ter uma ideia melhor dos efeitos do estado duplamente ocupado na densidade de corrente para valores pequenos de V_{bias} . Na ausência de interação (Fig. 3.9(a)), temos uma corrente que localmente é zero. Quando consideramos a interação elétron-elétron (Figura 3.9(b)), devido à presença do estado duplamente ocupado, a densidade eletrônica permite o acoplamento entre os eletrodos. Assim, para esta pequena diferença de potencial, uma corrente local através do ponto quântico surge, gerando assim a transmissão de elétrons de um lado para o outro, resultando em uma corrente total diferente de zero no sistema.

Finalmente, para a corrente total, observamos na Fig. 3.8(d) uma competição entre os termos coerentes e incoerentes. Para alguns valores de *gate*, I_{coh} é o termo dominante, enquanto para outros a contribuição de I_{incoh} é mais intensa. Esses

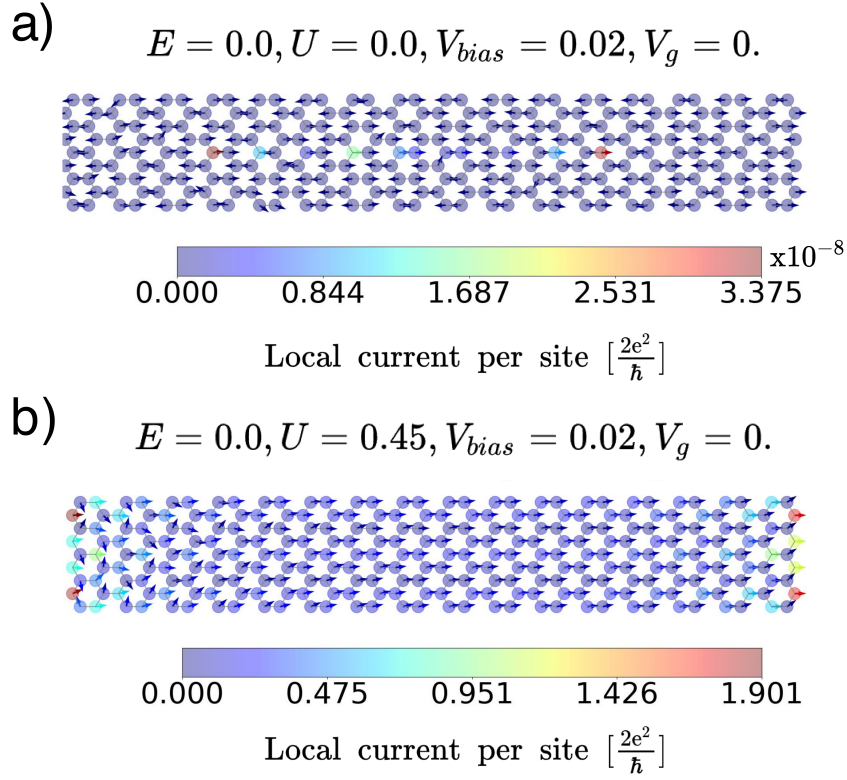


Figura 3.9: **Corrente local:** Corrente local por sítio i , $i_i(E) = \sum_j (i_{ij}(E) \hat{r}_{ij} + i_{ji}(E) \hat{r}_{ji})$ para $E = 0$ eV, $V_g = 0$ eV e $V_{bias} = 0.02$ eV. a) Na ausência de interação, e b) na presença de interação com $U = 0.45$ eV.

resultados também corroboram o argumento de que uma descrição de campo médio é válida quando o nível está duplamente ocupado.

Ao varrer tanto o V_{bias} quanto V_g , podemos entender o comportamento global do dispositivo traçando o chamado diagrama de instabilidade, ou seja, a corrente em função do V_{bias} e do $gate$, comparando o caso não interagente (Fig. 3.10(a)) com o interagente (Fig. 3.10(b)). Podemos observar que, para $U = 0$ eV, a corrente na região central do diagrama é zero. Quando consideramos $U = 0.45$ eV, vemos o surgimento dos diamantes de Coulomb e uma boa descrição do bloqueio de Coulomb para as regiões de corrente-zero. Isso está de acordo com os resultados experimentais [7] e demonstra que, para baixos valores de V_{bias} , estamos no regime de transmissão de um único elétron. Apresentamos $|I_{coh}| - |I_{incoh}|$ na Fig. 3.10(c). Assim, vemos que na região onde $Im[\Sigma_{ee}] \neq 0$, o termo incoerente é dominante, e observa-se o surgimento de fenômenos de decoerência. O mesmo efeito foi observado por Lozano-Negro *et al.* [92].

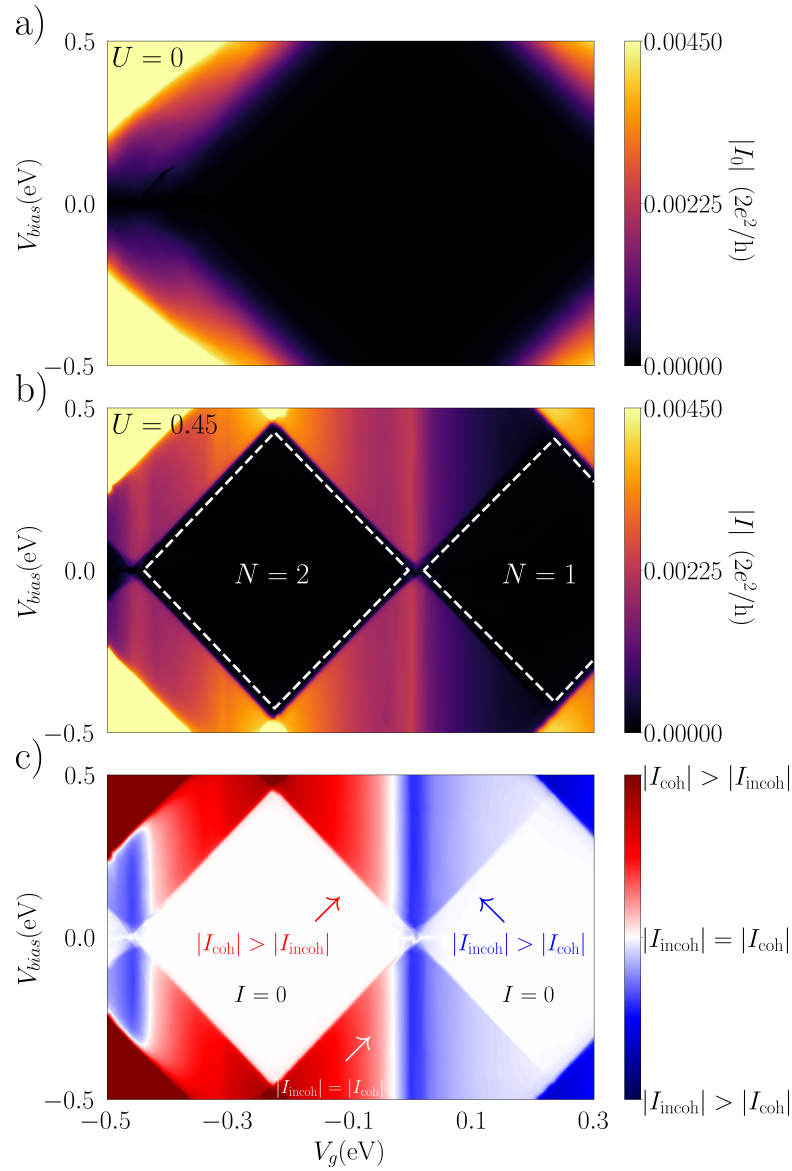


Figura 3.10: **Diagrama de instabilidade para corrente e diferença entre os termos de corrente, $|I_{coh}| - |I_{incoh}|$:** a) O diagrama de instabilidade para corrente, com $U = 0$ eV, mostra o valor absoluto da corrente em função do *gate* (eixo x) e da polarização (eixo y). b) O diagrama de instabilidade para corrente efetiva, com $U = 0,45$ eV, mostra o valor absoluto da corrente em função do *gate* (eixo x) e da polarização (eixo y). c) O diagrama de diferença entre os termos absolutos das correntes, com $U = 0,45$ eV, é mostrado em função do *gate* (eixo x) e do viés (eixo y). Na região vermelha, $|I_{coh}|$ é o termo dominante, na região azul, $|I_{incoh}|$ é o termo dominante e na região branca, temos $|I_{coh}| = |I_{incoh}|$.

3.3 Conclusão

Neste capítulo, estudamos de forma mais realista a interação elétron-elétron em dispositivos quânticos. Inicialmente, observamos que considerar a presença

da interação nos estados moleculares em vez dos orbitais atômicos proporciona uma melhor descrição física e simplifica o problema. A presença da interação elétron-elétron mostrou-se essencial para descrever o bloqueio de Coulomb de forma mais precisa.

Além disso, notamos que a inclusão do termo de transporte incoerente é crucial para descrever sistemas interagentes. Como demonstrado neste capítulo, existem regiões onde o processo incoerente governa a dinâmica do transporte eletrônico. No entanto, quando a autoenergia de interação é real ($Im[\Sigma_{ee}] = 0$) e, portanto, $\Gamma_{ee} = 0$, o termo incoerente é nulo. Nesses casos, o termo BL torna-se dominante, como esperado.

Portanto, ainda é necessário investigar outros sistemas e observar a competição entre esses termos para determinar se essa abordagem fornecerá uma regra geral para sistemas interagentes no regime de acoplamento não proporcional.

Capítulo 4

Nanofita de Grafeno Desordenada na Presença de Interação Elétron-Elétron

Estruturas baseadas em carbono, como os nanotubos [17], e mais recentemente o grafeno [18, 74, 19], têm atraído grande atenção da comunidade científica, do ponto de vista da ciência básica [75, 76]. Como já mencionado, a ausência de um gap no grafeno é um dos problemas que impedem a utilização deste material em diversas aplicações em eletrônica. Isso pode ser mitigado reduzindo a dimensionalidade do grafeno realizando cortes em direções específicas, formando nanofitas [48].

A presença de defeitos em nanofitas de grafeno desempenha um papel importante nas propriedades de transporte [26, 93] e, dependendo da concentração, leva a um fenômeno conhecido como localização de Anderson [94], fenômeno este descrito inicialmente por P. W. Anderson [95]. Onde, a presença de desordem em sistemas como nanofitas leva à formação de estados localizados, os quais, combinados com a natureza ondulatória, resultam em fenômenos de interferência destrutiva. Devido a isso, a probabilidade de transporte eletrônico decai exponencialmente à medida que as dimensões do sistema aumentam na direção do transporte de carga [95].

Normalmente, ao estudar modelos desordenados [26], a interação elétron-elétron é negligenciada [96]. Aqui, buscamos entender como a interação elétron-elétron altera as propriedades eletrônicas e de transporte de nanofitas de grafeno desordenadas. Para tanto, realizamos cálculos usando o método da função de Green fora de equilíbrio [90], descrevendo os estados interagentes por meio de um modelo de impureza Anderson multi-orbital (AIM) juntamente com a chamada aproximação de não cruzamento (NCA) [64] e uma generalização para a fórmula de Meir-Wingreen para transporte incoerente [47]. Estes métodos foram todos descritos em detalhes no Cap. 2. Através da combinação desses métodos, descrevemos corretamente fenômenos de transporte para sistemas interagentes.

4.1 Metodologia

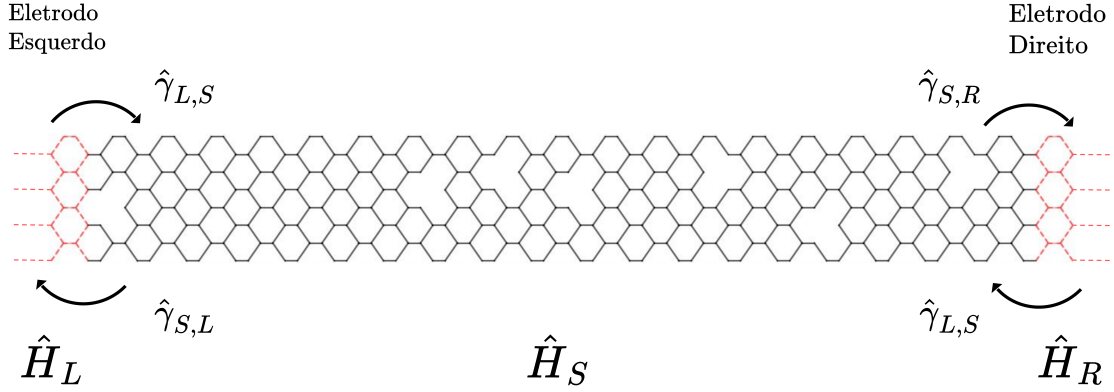


Figura 4.1: **Visão esquemática do dispositivo usado em nossos cálculos:** Na região central, uma nanofita de grafeno (NR) armchair com vacâncias, e nas regiões em vermelho, os eletrodos esquerdo e direito. As regiões são fortemente acopladas entre si.

A configuração base utilizada ao longo do capítulo é apresentada na Fig. 4.1. O sistema é descrito usando um Hamiltoniano tight-binding com um único orbital por sítio representando os orbitais p_z em cada átomo de carbono [48]. Dividimos nosso sistema em três regiões conforme descrito pela primeira vez por Caroli *et al* [54, 91], ou seja, eletrodos semi-infinitos esquerdo e direito e uma região de espalhamento, onde $\hat{H}_S$ é o Hamiltoniano da região de espalhamento. Tomamos então um segmento finito de uma nanofita de grafeno de 8 armchair que consiste em $LNR = 18$ células unitárias acopladas entre si. Os eletrodos esquerdo e direito ($\hat{H}_L$ e $\hat{H}_R$) possuem a mesma simetria da região de espalhamento. Assim, o sistema é de natureza metálica [49].

O Hamiltoniano completo, é dado por:

$$\hat{H} = \hat{H}_L + \hat{H}_S + \hat{H}_R + \hat{\gamma}_L + \hat{\gamma}_R. \quad (4.1)$$

Aqui, $\hat{\gamma}_{L/(R)} = \hat{\gamma}_{L/(R),S} + \hat{\gamma}_{S,L/(R)}$ é a matriz de acoplamento entre a região de espalhamento e o elemento de superfície, conforme descrito na Sec. 2.5, do respectivo eletrodo

Os parâmetros tight-binding do sistema foram escolhidos da seguinte maneira: tomamos a energia de sítio $\epsilon = 0$ eV (conforme a literatura [49]), e o acoplamento entre primeiros vizinhos $\gamma_{i,j} = \gamma_0 = 3\text{eV}$.¹

¹Aqui, embora tipicamente para o grafeno seja $\gamma_0 = 2,7\text{eV}$, escolhemos ligeiramente maior para

Em equilíbrio, a função de Green retardada não interagente é dada por:

$$\lim_{\eta \rightarrow 0^+} G_0 = [(E + i\eta)I - H_S - \Sigma_L - \Sigma_R]^{-1}, \quad (4.2)$$

que é a quantidade chave para as propriedades eletrônicas do sistema, e Σ_L (Σ_R) é a auto-energia para o eletrodo esquerdo (direito). Dada a estrutura cristalina dos eletrodos e utilizando a equação de Dyson, as auto-energias podem ser obtidas através de um método recursivo, conforme discutido no capítulo anterior.

A densidade de estados projetada nos orbitais da região de espalhamento é dada por:

$$D_0(E) = -\frac{1}{\pi} \text{Tr} [\text{Im}\{G_0\}]. \quad (4.3)$$

Ao mesmo tempo, a transmissão não interagente, que inclui apenas os processos coerentes, é dada pela equação de Büttiker-Landauer:

$$T_0 = \text{Tr} [\Gamma_R G_0^\dagger \Gamma_L G_0], \quad (4.4)$$

e

$$\Gamma_{L/R} = i(\Sigma_{L/R} - \Sigma_{L/R}^\dagger). \quad (4.5)$$

Neste trabalho, os defeitos considerados serão vacâncias. No caso em que adicionamos uma única vacância no segmento da NR, como ilustrado na Fig.4.2, obtemos a transmissão para o caso não interagente na presença e na ausência da vacância. No caso sem vacância, a transmissão é quantizada (curva tracejada preta). Entretanto, na presença da vacância, observamos que a intensidade da transmissão (Fig.4.2, curva sólida em roxo) diminui em relação ao caso da nanofita perfeita (Fig.4.2, curva tracejada preta), e, portanto, temos um modo propagante a menos. Para a região próxima ao nível Fermi, $E \approx 0\text{eV}$, temos o surgimento de uma antirressonância. Além disso, os orbitais moleculares são localizados espacialmente próximo à vacância (Fig.4.2). Isso indica que a interação elétron-elétron, nestes orbitais moleculares, pode levar a mudanças significativas no estado de carga do sistema.

A inclusão de interações é tipicamente desconsiderada para sistemas no regime metálico em temperaturas acima da temperatura de Kondo [97]. Entretanto, temos a presença de estados localizados em que a interação não pode ser negligenciada.

facilitar a observação de alguns resultados, os quais não dependem diretamente do valor de γ_0 desde que $|\gamma_0| > \epsilon$.

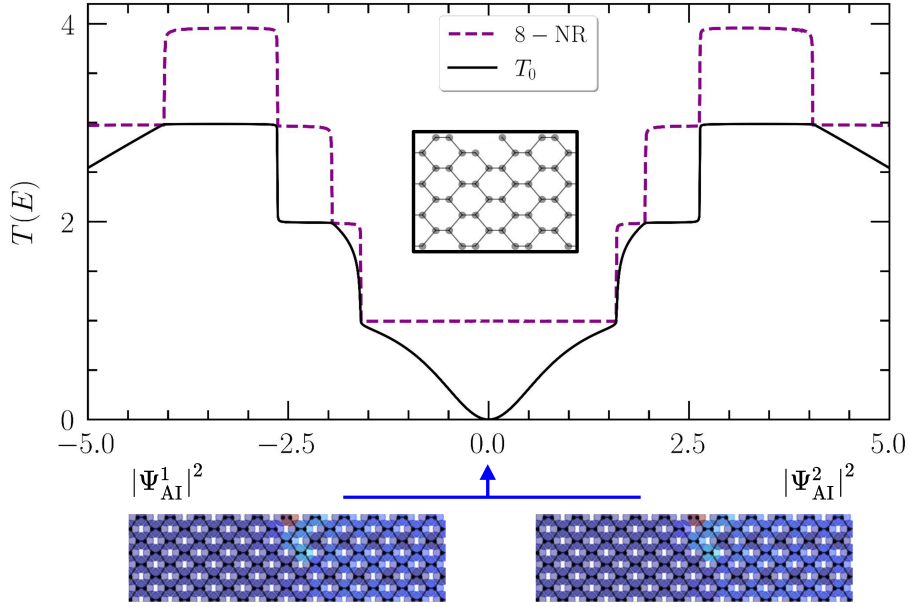


Figura 4.2: **Transmissão e funções de onda para NR:** A transmissão para a NR perfeita (curva tracejada roxa), a transmissão na presença de uma única vacância (curva preta) e o módulo quadrado das funções de onda para os dois estados moleculares próximos ao nível de Fermi. $E_1 \approx E_2 \approx 0$ eV. Por fim, o quadro em preto está aqui para ilustrar em qual orbital adicionamos a vacância.

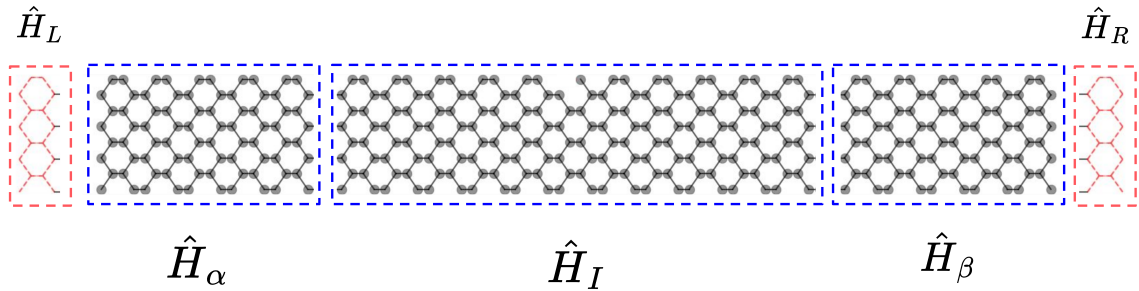


Figura 4.3: **Divisões das sub-regiões da NR:** Visão das subdivisões feita sobre a NR conforme a estrutura da molécula estendida (Ver Sec. 2.7). Na região central da NR, temos a região de interação $\hat{H}_I$, que contém a vacância, $\hat{H}_\alpha$ ($\hat{H}_\beta$) a região à esquerda (direita) da região de interação, que não irá conter interação, e nas regiões em vermelho, os eletrodos esquerdo e direito. As regiões são fortemente acopladas entre si.

Para resolver esse problema, primeiro dividimos o Hamiltoniano da região de espalhamento, como ilustrado na Fig. 4.3, e de acordo com a descrição apresentada na Sec. 2.7:

$$\hat{H}_S = \hat{H}_\alpha + \hat{H}_I + \hat{H}_\beta + \hat{\gamma}_\alpha + \hat{\gamma}_\beta. \quad (4.6)$$

Aqui, $\hat{H}_I$ é a chamada região de interação (RI), que contém os orbitais atômicos

próximos à vacância e que não acoplam diretamente com os eletrodos. Assim, escolhemos $L_I = 11$, considerando 5 células unitárias para a esquerda e para a direita da vacância.² Ao mesmo tempo, $\hat{H}_\alpha$ ($\hat{H}_\beta$) contém os orbitais à esquerda (direita) da RI, que incluem os orbitais que acoplam diretamente ao eletrodo esquerdo (direito). Definimos a base dos estados moleculares de impureza de Anderson, com N_{AI} estados. Este é um subconjunto dos estados moleculares da NR que surgem devido à presença da vacância e aqueles que contêm interações elétron-elétron.

$$U_\psi = (\psi_{AI}^1, \psi_{AI}^2, \dots, \psi_{AI}^n)^T. \quad (4.7)$$

A interação elétron-elétron é incluída aqui considerando um modelo de impureza multi-orbital de Anderson para os ψ_{AI}^i ,

$$\hat{H}_I |\psi_{AI}^i\rangle = \epsilon_{AI}^i |\psi_{AI}^i\rangle. \quad (4.8)$$

Portanto, adicionamos:

$$\hat{H}^{ee} = \sum_{i,j,k,l}^{N_{AI}} U_{i,j,k,l} \hat{f}_{i,\sigma}^\dagger \hat{f}_{i,\sigma'}^\dagger \hat{f}_{l,\sigma'} \hat{f}_{k,\sigma}, \quad (4.9)$$

ao Hamiltoniano da região de interação. Os operadores de criação (aniquilação) $\hat{f}_{i,\sigma}^\dagger$ ($\hat{f}_{i,\sigma}$) atuam nos orbitais moleculares da AI, onde σ é a projeção do spin ($\sigma = +$ ou $-$). Aqui, por simplicidade, tomamos

$$U_{i,j,k,l} = \begin{cases} U, & \text{se } l = k = j = i \\ 0, & \text{caso contrario} \end{cases} \quad (4.10)$$

com a interação $U = 3,5\text{eV}$. Estamos considerando que os orbitais são ortogonais entre si, portanto, podemos projetar diretamente a função de Green da região de espalhamento sobre os estados AI (ver Sec. 2.7.),

$$G_{AI} = \begin{pmatrix} 0_\alpha \\ U_\psi \\ 0_\beta \end{pmatrix}^\dagger \left[(E + i\eta)I - H_S - \Sigma_L - \Sigma_R - \Sigma_{ee} \right]^{-1} \begin{pmatrix} 0_\alpha \\ U_\psi \\ 0_\beta \end{pmatrix}, \quad (4.11)$$

que inclui a interação elétron-elétron através da auto-energia Σ_{ee} . Utilizando a

²É importante notar que podemos extrair apenas esse segmento porque as funções de onda ψ_{AI}^i dos estados moleculares vão a zero à medida que os orbitais atômicos ficam mais distantes da vacância, como já mencionado e mostrado na Fig. 4.2.

metodologia apresentada no capítulo anterior (Cap. 3), definimos a função de hibridização $\bar{\Delta}_{AI}$,

$$\bar{\Delta}_{AI} = EI_{N_{AI}} - \epsilon_{AI} - \bar{\sigma}_{ee} - G_{AI}^{-1}. \quad (4.12)$$

Aqui, temos que os estados da AI são fracamente acoplados aos demais orbitais da região de espalhamento; consequentemente, são fracamente acoplados aos eletrodos, e a função de hibridização pode ser utilizada no solucionador de impurezas. Isso define o conjunto de equações para o campo médio dinâmico que pode ser iterado de forma autoconsistente. Em nosso código, usamos a chamada aproximação de não cruzamento (NCA) e obtemos a auto-energia para a interação elétron-elétron ($\bar{\sigma}_{ee}$) na base da AI. Realizando a projeção inversa, obtemos a auto-energia na representação dos orbitais atômicos,

$$\Sigma_{ee} = U_{\psi} \bar{\sigma}_{ee} U_{\psi}^T. \quad (4.13)$$

Obtemos então a função de Green retardada que descreve o problema interagente,

$$G = [G_0^{-1} - \Sigma_{ee}]^{-1}, \quad (4.14)$$

e podemos calcular as propriedades eletrônicas e de transporte para o problema considerando a interação elétron-elétron.

Aqui, consideramos a transmissão dada pela expressão de Fisher-Lee para o regime de acoplamento não proporcional [47] (como apresentada na Subseção 2.9.3).

$$T_{\text{int}} = \text{Tr} [\Gamma_R G^\dagger \Gamma_L G] + \text{Tr} [\Gamma_R G^\dagger \Gamma_{ee} G], \quad (4.15)$$

com $\Gamma_{ee} = i(\Sigma_{ee} - \Sigma_{ee}^\dagger)$, e podemos identificar a transmissão efetiva por um termo coerente e um termo incoerente, $T_{\text{int}} = T_{\text{coh}} + T_{\text{incoh}}$.

4.2 Resultados

No intuito de entendermos melhor os efeitos da interação devido à presença de impurezas, inicialmente consideramos uma única impureza magnética (um orbital atômico fracamente acoplado com a nanofita) e posteriormente estudamos a presença de vacâncias. Como observaremos a seguir, ambas têm uma descrição semelhante. A única diferença relevante é o fato de que, para o caso da impureza magnética, temos apenas um estado localizado.

4.2.1 Impureza Magnética Única

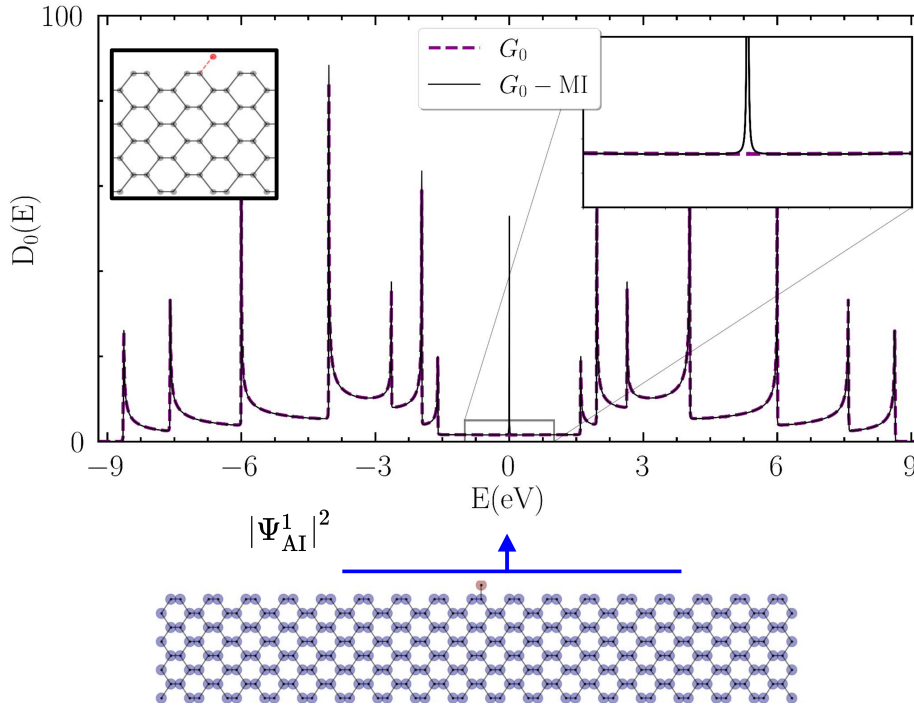


Figura 4.4: **Densidade de Estados para a NR:** a) Densidade de Estados para uma nanofita de grafeno na ausência de defeitos (curva tracejada roxa) e na presença de uma MI (curva preta). b) O módulo quadrado da função de onda para o estado molecular localizado no nível de Fermi ($E_1 \approx 0$ eV).

Adicionamos, portanto, uma impureza magnética (MI, da sigla em inglês) no segmento da NR (como ilustrado na Fig. 4.4(a)), de modo que o novo orbital atômico (ϵ_{MI}) e o vizinho mais próximo são fracamente acoplados. Adicionando ao Hamiltoniano da região de interação,

$$\begin{pmatrix} 0 & \dots & \dots & \dots & 0 \\ \vdots & \ddots & \gamma_{S,MI} & \vdots & \vdots \\ \vdots & \gamma_{MI,S} & \epsilon_{MI} & \vdots & \vdots \\ \vdots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 \end{pmatrix}_{(N+1) \times (N+1)}$$

Assim, escolhemos $\gamma_{MI,S} = \gamma_{S,MI} = 1/6\gamma_0$. Com esses parâmetros, obtemos a densidade de estados para o caso não interagente na presença da MI, mostrado na Fig. 4.4, e vemos o surgimento de uma ressonância na DOS que corresponde a um estado localizado em torno da MI. Mais uma vez, isso indica que a interação

elétron-elétron, neste orbital molecular, pode levar a mudanças significativas no estado de carga do sistema.

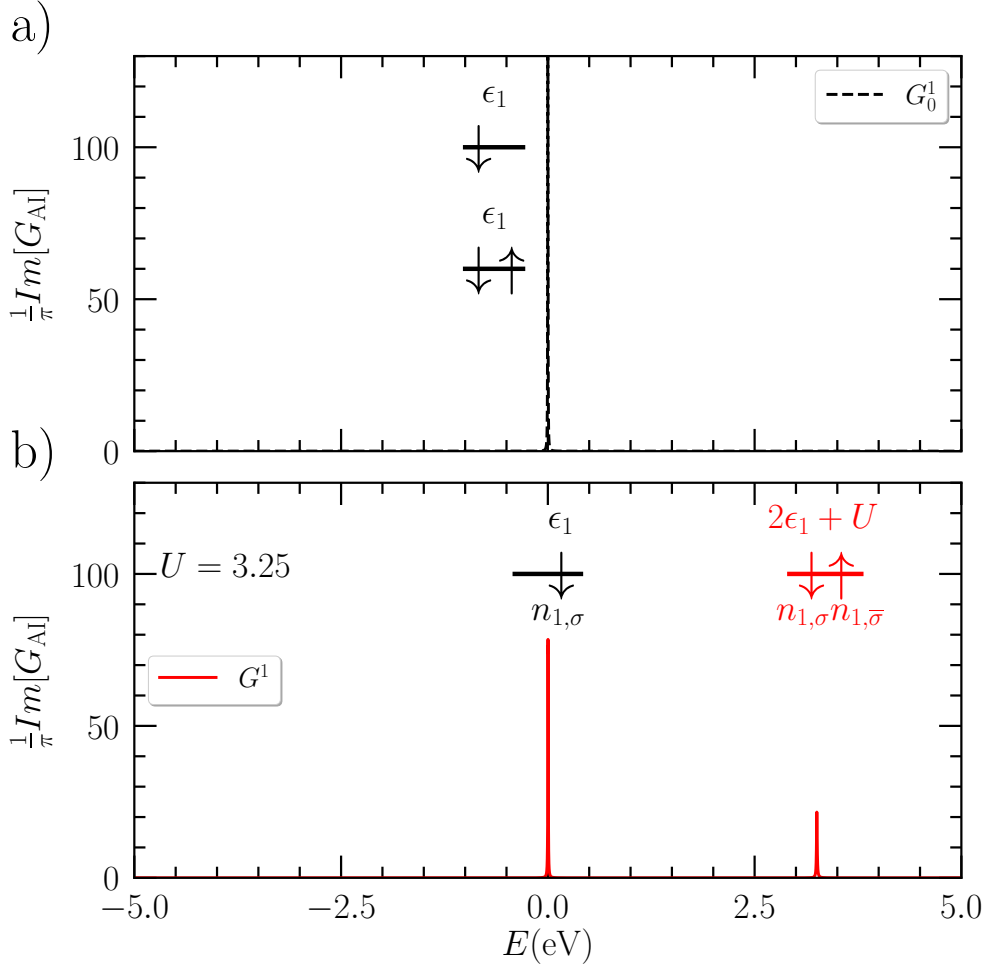


Figura 4.5: **Densidade de estados para impureza magnética:** densidade de estados para os estados de MI na representação de partícula única. (a) O estado duplamente ocupado (curva tracejada preta) na ausência de interação e (b) o estado com ocupação única em $E = 0\text{eV}$ e o estado duplamente ocupado em $E = 3.25\text{eV}$ na presença de interação, com $U = 3.25\text{eV}$ (curva sólida vermelha).

Aqui consideramos o estado que surge devido a MI, posicionado no nível Fermi ($E_1 \approx 0\text{eV}$). Assim, como temos apenas um estado na base AI, ($\dim(G_{\text{AI}}) = 1$), todos os objetos na base da AI são escalares. Definimos G^1 e G_0^1 como o elemento de G_{AI} com e sem interação, respectivamente.

Olhando inicialmente para a densidade de estados para os estados da MI não interagentes na Fig. 4.5(a), observamos que ele é um estado fracamente acoplado à NR. Quando ativamos a interação (ver Fig. 4.5(b)), observa-se que o nível duplamente ocupado do primeiro estado de partícula única é deslocado em U . Ao

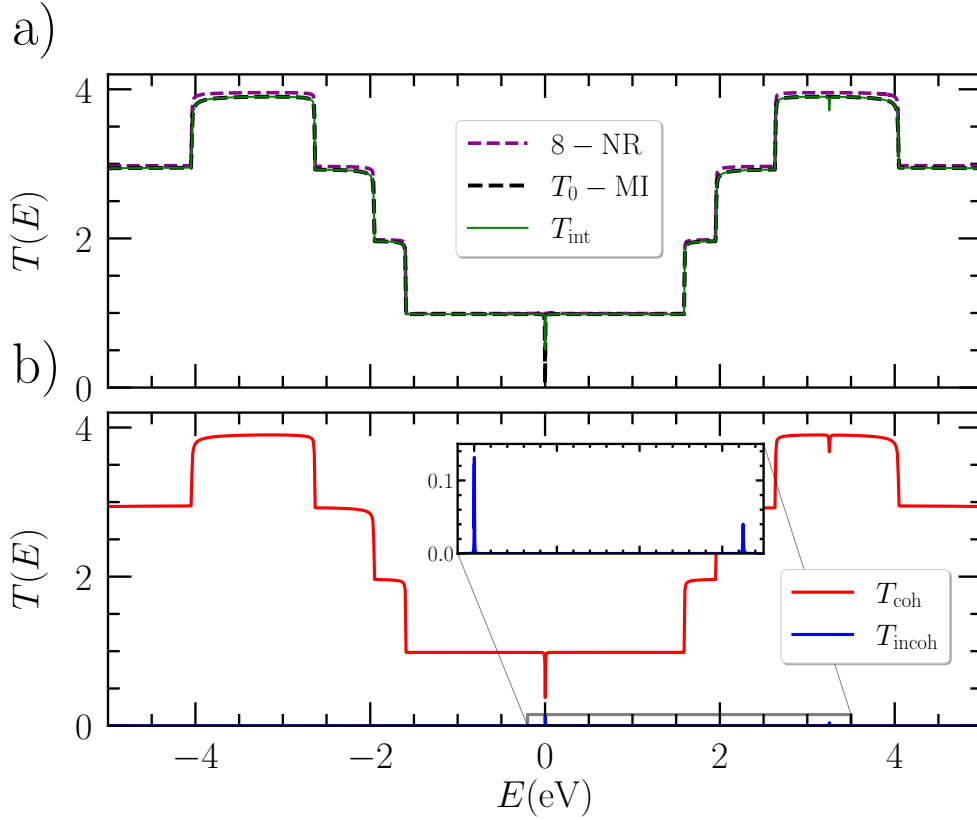


Figura 4.6: **Transmissão em função da energia:** a transmissão para a MI não interagente (curva tracejada preta), a transmissão efetiva (curva sólida verde), a transmissão coerente (curva sólida vermelha) e a transmissão incoerente (curva sólida azul) para a MI interagente com potencial Coulombiano $U = 3.25\text{eV}$. A transmissão para a nanofita na ausência de defeito (curva tracejada roxa) está presente como guia para comparação.

olhar para a influência do estado carregado na transmissão, como mostrado na Fig. 4.6(a), surgem duas antirressonâncias na transmissão efetiva: uma no nível de Fermi ($E = 0$), associada ao estado unicamente ocupado, e outra em $E = U$, devido ao estado duplamente ocupado. Observando as diferentes contribuições na Fig. 4.6(b), a transmissão coerente é dominante para todo o espectro; entretanto, para os valores onde temos antirressonâncias, observamos o surgimento de contribuições dos termos incoerentes da transmissão. Isso indica que a presença do tunelamento sequencial pode ser uma regra importante nesses sistemas, sendo intensificado na presença de uma vacância.

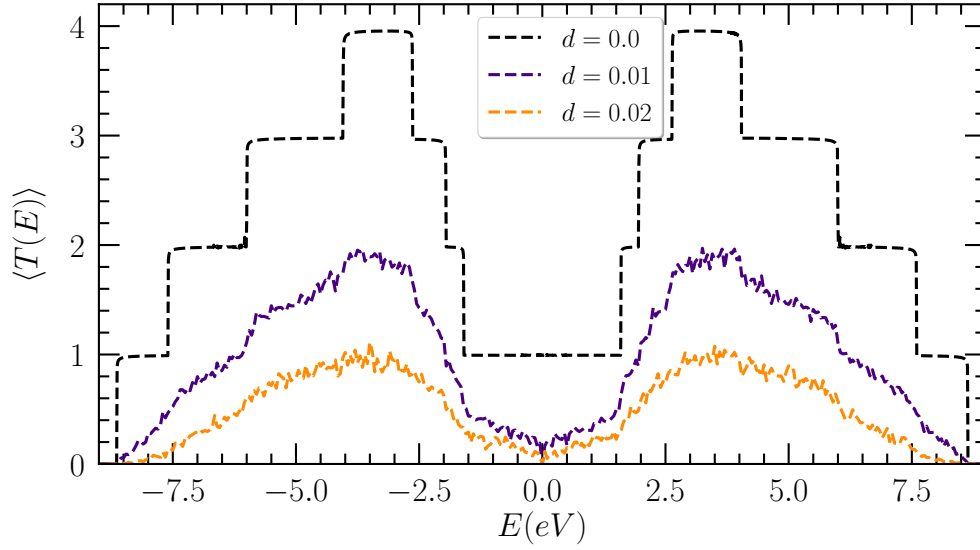


Figura 4.7: **Média da transmissão em função da energia:** Média Transmissão para a NR na ausência de defeitos (curva tracejada preta) e na presença de diferentes concentrações de vacâncias *i*) $d = 0.01$ (curva sólida anil) e *ii*) $d = 0.02$ (curva sólida laranja) na ausência de interação.

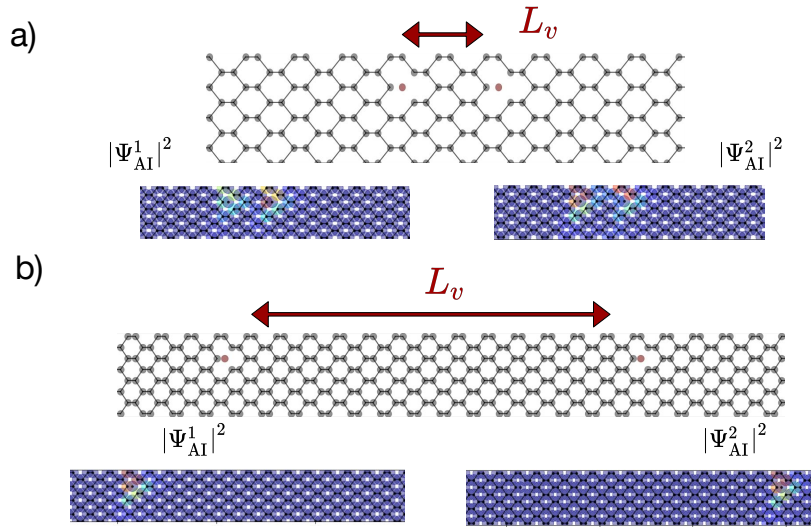


Figura 4.8: **Rede e Densidade de Estados para a NR:** A configuração da rede na presença de duas vacâncias e a densidade de estados associada à função de onda dos dois estados de impureza molecular para *a*) $L_v = 1$ e *b*) $L_v = 13$.

4.2.2 Desordem por Vacância

Adicionamos a presença de desordem através da presença de vacâncias. Definimos então a concentração,

$$d = \frac{N_{\text{def}}}{N_T}, \quad (4.16)$$

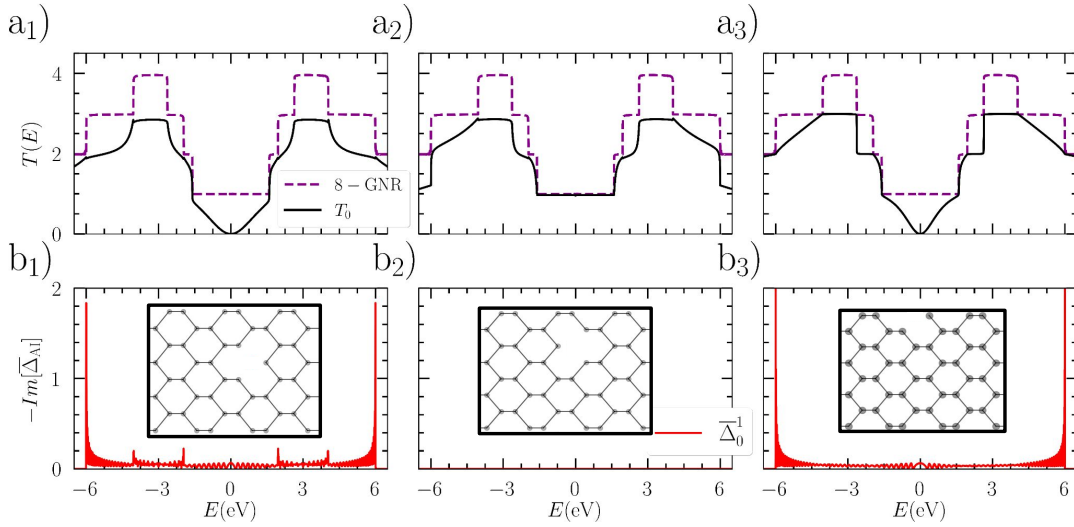


Figura 4.9: **Transmissão e função de hibridização para a NR não interagente:** Na presença de uma única vacância, *a*) a transmissão e *b*) a função de hibridização. Ambos para diferentes configurações de impureza (diferentes colunas) com o quadro em preto ilustrando a configuração escolhida para aquela coluna. A transmissão para a nanofita na ausência de defeito (curva tracejada roxa) está presente como guia para comparação.

Com N_{def} sendo o número de vacâncias na nanofita, N_T o número total de orbitais atômicos e d a concentração de vacâncias na nanofita. A partir disso, calculamos a transmissão para algumas concentrações de desordem na ausência de interação. Foram realizadas 1000 configurações e calculadas as médias sobre a transmissão, mostrada na Figura 4.7. Temos que, mesmo para baixa concentração de vacâncias, a transmissão é suprimida, e à medida que a concentração aumenta, a transmissão tende a diminuir ainda mais, como esperado na presença de desordem [22] e que leva à localização de Anderson [95].

Observa-se que para vacâncias próximas (Fig. 4.8(a)), não é possível obter funções de onda localizadas em torno de uma única vacância, o que indica que esses estados moleculares de impurezas estão acoplados entre si. Assim, para considerar a presença de interação, seria necessário considerar todas as configurações de impurezas. No entanto, ao escolher $L_v > 7$, os estados de impurezas são desacoplados entre si (Fig. 4.8(a)).

Por fim, ao considerar diferentes configurações de defeitos para o caso de uma vacância única, mostrado na Fig. 4.9, escolhemos 3 diferentes posições e obtivemos a transmissão (Fig. 4.9(a)) e a função de hibridização (Fig. 4.9(b)),

ambos na ausência de interação. Observamos a presença da antirressonância na transmissão para dois casos (a_1) e b_3). Nestes casos, as funções de hibridização das AIs são aproximadamente iguais (b_1) e b_3), principalmente para $-2 < E < 2$. Enquanto para a_2), a intensidade da transmissão diminui, mas apenas longe do nível de Fermi, mantendo assim as propriedades metálicas do material. Além disso, vemos que a função de hibridização b_2) é nula para todo o espectro de energia, indicando que esse estado de impureza molecular está desacoplado dos demais orbitais moleculares do sistema. Assim, ao obter a função de hibridização de uma configuração na presença de uma única vacância e utilizando o *solver*, temos a auto-energia de interação para este caso como uma boa aproximação para a auto-energia de diferentes configurações de vacâncias. Entretanto, para as vacâncias que não produzem influência na transmissão e suas funções de hibridização são nulas, isso indica que os efeitos da interação elétron-elétron, nesses casos, são nulos. Isso nos permite resolver o caso desordenado na presença de interação.

4.2.3 Vacância única

Enquanto o caso de uma impureza magnética serve como um *benchmark* para a nossa metodologia, como mencionado abaixo, a introdução de vacâncias permite olharmos para a situação em que temos mais de um estado localizado. Nos voltamos então para o caso na presença de uma única vacância. Seguindo a densidade de estados apresentada na Figura 4.2, considerando os dois estados localizados próximos à vacância e que estão na região de energia próximo ao nível de Fermi.

A transmissão em função da energia é apresentada na Figura 4.10. Conforme já discutido, a presença de vacâncias leva a uma antirressonância na região do nível de Fermi, e no caso não interagente a transmissão vai a zero. Ao ligarmos a interação, observamos que não há mais supressão do transporte. Notamos também que este efeito vem quase exclusivamente do termo incoerente da transmissão (ver Figura 4.2(b)). Então, temos que a intensidade de T_{incoh} está associada à intensidade da antirressonância na ausência de interação. Porém, diferente da presença da MI, temos que o regime metálico é praticamente restaurado próximo ao nível de Fermi devido à presença da interação elétron-elétron.

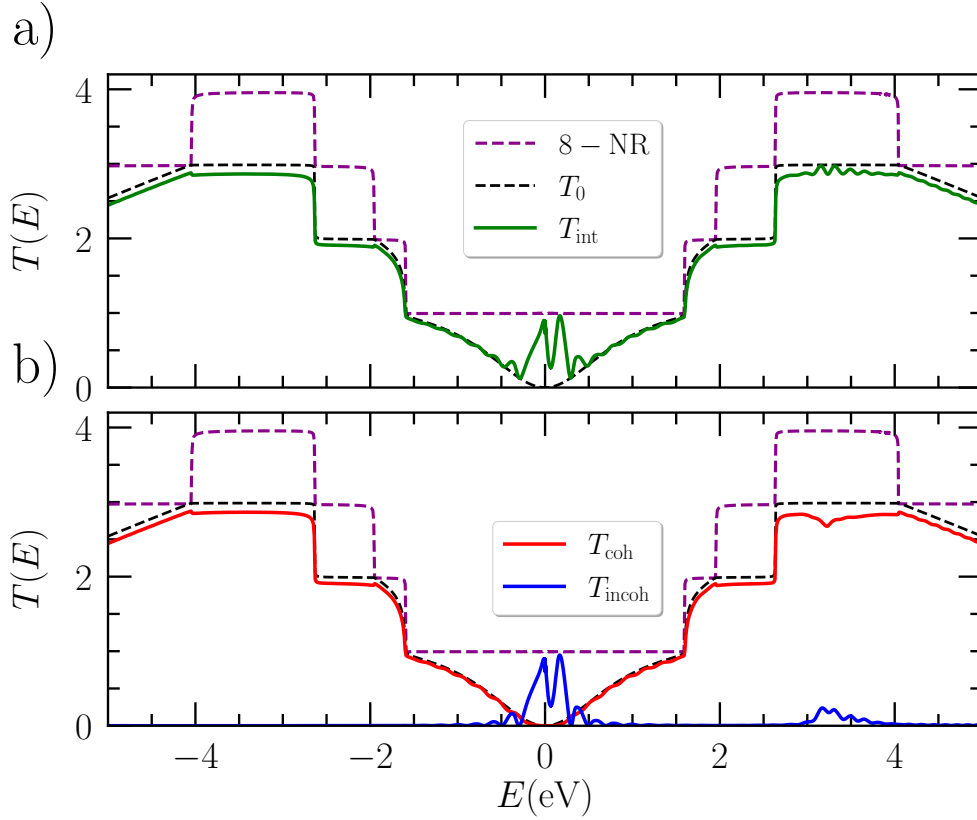


Figura 4.10: **Transmissão em função da energia:** *a)* transmissão na presença de uma única vacância não interagente (curva tracejada preta) e a transmissão efetiva (curva sólida verde), *b)* contribuições da transmissão coerente (curva sólida vermelha) e da transmissão incoerente (curva sólida azul) na presença de uma única vacância interagente com potencial Coulombiano $U = 3.25\text{eV}$. A transmissão para a nanofita na ausência de defeito (curva tracejada roxa) está presente como guia para comparação.

4.3 Conclusão

Neste capítulo, estudamos a influência de defeitos em nanofitas de grafeno, classificando sua influência em dois tipos: *i)* defeitos que levam à formação de estados localizados fracamente acoplados aos estados moleculares da rede, onde a interação elétron-elétron não pode ser desprezada; *ii)* defeitos que resultam em estados localizados e desacoplados dos demais estados moleculares da rede, onde os efeitos da interação elétron-elétron são nulos. Observamos que a presença da interação gera processos de transmissão decoerente, onde, para determinadas regiões do espectro de energia, o termo de transmissão incoerente é dominante e, em alguns casos, restaura a natureza metálica do sistema. Assim, a interação

também pode facilitar os processos de transmissão. Por fim, concluímos os estudos preliminares sobre como considerar a presença de interação em nanofitas de grafeno desordenadas.

Argumentamos ainda que uma descrição molecular é mais natural para esses sistemas, e para baixas concentrações de defeitos, uma aproximação pode ser feita mapeando um sistema desordenado com múltiplos defeitos interagentes acoplados, médias sobre diferentes configurações de defeitos interagentes desacoplados entre si, o que pode ser uma ferramenta para estudar efeitos de localização, incluindo a interação elétron-elétron.

Capítulo 5

Conclusão Geral e Perspectivas

Nesta dissertação, propomos a construção de um ferramental para o estudo e caracterização do transporte quântico decoerente devido à interação elétron-elétron, combinando diferentes modelos teóricos. Desenvolvemos uma biblioteca em Python que abrange todos os métodos de interesse, incluindo o método da função de Green fora de equilíbrio, a descrição dos estados de interação por meio de um modelo de Anderson para impurezas magnéticas, uma generalização da fórmula de Meir-Wingreen para o regime de acoplamento não proporcional e a utilização de um *solver* externo para a abordagem da aproximação de não cruzamento.

Esses métodos foram aplicados a uma nanofita de grafeno na presença de vacâncias e à simulação de um transistor de elétron único, baseado na junção de diferentes materiais de grafeno. Observamos que o comportamento global desses sistemas é semelhante, pois a interação quebra a coerência de fase quântica no transporte, levando à transmissão de carga incoerentemente. Isso pode, em alguns casos, facilitar a mobilidade eletrônica no material. Além disso, os métodos propostos descrevem adequadamente fenômenos como o bloqueio Coulombiano e os diamantes de Coulomb associados. Portanto, esse fenômeno pode ser uma regra importante para o estudo de fenômenos de transporte eletrônico, exigindo mais estudos em diferentes materiais e dispositivos. Estudos desses sistemas análogos já estão sendo realizados por nós. Além disso, exploraremos, no caso do transistor de elétron único, o espaço de fase dos parâmetros e também adicionaremos estados moleculares mais distantes em relação ao nível de Fermi. Enquanto no caso da nanofita desordenada, investigaremos casos com múltiplas vacâncias e a dependência com a concentração de impurezas. Assim, dispomos de uma ferramenta poderosa para simular sistemas baseados em grafeno na presença de interações.

Referências

- [1] Office of Basic Energy Sciences (BES). Scale of things chart.
- [2] R.K. Ratnesh, A. Goel, G. Kaushik, H. Garg, Chandan, M. Singh, and B. Prasad. Advancement and challenges in mosfet scaling. *Materials Science in Semiconductor Processing*, 134:106002, 2021.
- [3] Adenilson J Chiquito and Francesco Lanciotti Jr. O transistor, 50 anos. *Revista Brasileira de Ensino de Física*, 20(4):309–314, 1998.
- [4] Shaoyun Huang, Naoki Fukata, Maki Shimizu, Tomohiro Yamaguchi, Takashi Sekiguchi, and Koji Ishibashi. Classical coulomb blockade of a silicon nanowire dot. *Applied Physics Letters*, 92:213110–213110, 05 2008.
- [5] Jochen Kirschbaum, E. Hohberger, Robert Blick, W. Wegscheider, and Max Bichler. Integrating suspended quantum dot circuits for applications in nanomechanics. *Applied Physics Letters*, 81:280 – 282, 08 2002.
- [6] Jason Stafford, Andrius Patapas, Nwachukwu Uzo, Omar Matar, and Camille Petit. Towards scale-up of graphene production via non-oxidizing liquid exfoliation methods. *AIChE Journal*, 64, 03 2018.
- [7] Jian Zhang, Liu Qian, Gabriela Barin, Abdalghani Daaoub, Peipei Chen, Klaus Müllen, Sara Sangtarash, Pascal Ruffieux, Roman Fasel, Hatef Sadeghi, Jin Zhang, Michel Calame, and Mickael Perrin. Contacting individual graphene nanoribbons using carbon nanotube electrodes. *Nature Electronics*, 6:1–10, 08 2023.
- [8] Eduardo C. Marino. *Quantum Field Theory Approach to Condensed Matter Physics*. Cambridge University Press, 2017.
- [9] David R. Barbero and Samuel D. Stranks. Functional single-walled carbon nanotubes and nanoengineered networks for organic- and perovskite-solar-cell applications. *Advanced Materials*, 28(44):9668–9685, 2016.
- [10] Leo Kouwenhoven and Leonid Glazman. Revival of the kondo effect. *Physics World*, 14(1):33, jan 2001.

- [11] Georgios Tsaramirsis, Antreas Kantaros, Izzat Aldarraj, Dimitrios Piromalis, Charalampos Apostolopoulos, Athanasia Pavlopoulou, Muath Alrammal, Zamhar Bin Awang Ismail, Seyed Buhari, Milos Stojmenovic, Hatem Tamimi, Princy Randhawa, Akshet Patel, and Fazal Khan. A modern approach towards an industry 4.0 model: From driving technologies to management. *Journal of Sensors*, 2022:1–18, 06 2022.
- [12] Pierre-Louis Viollet. From the water wheel to turbines and hydroelectricity. technological evolution and revolutions. *Comptes Rendus Mécanique*, 345(8):570–580, 2017. A century of fluid mechanics: 1870–1970.
- [13] Nicholas F. R. Crafts. The first industrial revolution: A guided tour for growth economists. *The American Economic Review*, 86(2):197–201, 1996.
- [14] Joel Mokyr and Robert H. Strotz. The second industrial revolution , 1870-1914. 2000.
- [15] Min Xu, Jeanne David, and Suk Kim. The fourth industrial revolution: Opportunities and challenges. *International Journal of Financial Research*, 9:90, 02 2018.
- [16] Klaus Schwab. *The Fourth Industrial Revolution*. Crown Publishing Group, USA, 2017.
- [17] S. Iijima et al. Helical microtubules of graphitic carbon. *nature*, 354(6348):56–58, 1991.
- [18] K. S. Novoselov, A. K. Geim, S. V. Morozov, D. Jiang, Y. Zhang, S. V. Dubonos, I. V. Grigorieva, and A. A. Firsov. Electric field effect in atomically thin carbon films. *Science*, 306(5696):666–669, oct 2004.
- [19] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim. The electronic properties of graphene. *Reviews of Modern Physics*, 81(1):109–162, January 2009. Publisher: American Physical Society.
- [20] J.E. Lilienfeld. Method and apparatus for controlling electric current. *US Patent*, 1926.
- [21] Gordon E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35, 2006.

- [22] Supriyo Datta. *Transmission function, S-matrix and Green's functions*, page 117–174. Cambridge Studies in Semiconductor Physics and Microelectronic Engineering. Cambridge University Press, 1995.
- [23] MA Omar. Addison-Wesley, 1975.
- [24] Henrik Bruus and Karsten Flensberg. *Many-Body Quantum Theory in Condensed Matter Physics*, volume 38. 01 2004.
- [25] Dmitry A. Ryndyk (auth.). Theory of quantum transport at nanoscale: An introduction. *Springer Series in Solid-State Sciences №184*, 2016.
- [26] Jason Stafford, Omar Matar, and Camille Petit. Producing graphene at scale, January 2019.
- [27] Néstor Merino-Díez, Aran Garcia-Lekue, Eduard Carbonell-Sanromà, Jingcheng Li, Martina Corso, Luciano Colazzo, Francesco Sedona, Daniel Sánchez-Portal, Jose I. Pascual, and Dimas G. de Oteyza. Width-dependent band gap in armchair graphene nanoribbons reveals fermi level pinning on au(111). *ACS Nano*, 11(11):11661–11668, 2017. PMID: 29049879.
- [28] Jian Zhang, Gabriela Borin Barin, Roman Furrer, Cheng-Zhuo Du, Xiao-Ye Wang, Klaus Müllen, Pascal Ruffieux, Roman Fasel, Michel Calame, and Mickael L. Perrin. Determining the number of graphene nanoribbons in dual-gate field-effect transistors. *Nano Letters*, 23(18):8474–8480, 2023. PMID: 37671914.
- [29] Jian Zhang, Liu Qian, Gabriela Borin Barin, Peipei Chen, Klaus Müllen, Pascal Ruffieux, Roman Fasel, Jin Zhang, Michel Calame, and Mickael L. Perrin. Double quantum dots in atomically-precise graphene nanoribbons. *Materials for Quantum Technology*, 3(3):036201, sep 2023.
- [30] Jian Zhang, Liu Qian, Gabriela Borin Barin, Abdalghani H. S. Daaoub, Peipei Chen, Klaus Müllen, Sara Sangtarash, Pascal Ruffieux, Roman Fasel, Hatef Sadeghi, Jin Zhang, Michel Calame, and Mickael L. Perrin. Ultimately-scaled electrodes for contacting individual atomically-precise graphene nanoribbons, 2022.
- [31] Jian Zhang, Oliver Braun, Gabriela Borin Barin, Sara Sangtarash, Jan Overbeck, Rimah Darawish, Michael Stiefel, Roman Furrer, Antonis Olziersky,

- Klaus Müllen, Ivan Shorubalko, Abdalghani H. S. Daaoub, Pascal Ruffieux, Roman Fasel, Hatef Sadeghi, Mickael L. Perrin, and Michel Calame. Tunable quantum dots from atomically precise graphene nanoribbons using a multi-gate architecture. *Advanced Electronic Materials*, 9(4):2201204, 2023.
- [32] Xiaoling Li, Xinran Wang, Li Zhang, Sangwon Lee, and Hongjie Dai. Chemically derived, ultrasmooth graphene nanoribbon semiconductors. *Science*, 319:1229 – 1232, 2008.
- [33] Twinkle, Anjali, Sudhir Kumar, J.K. Goswamy, Parveen Kumar, and Suresh Kumar. Chemically derived graphene nanoribbons from carbon nanotubes for supercapacitor application. *Materials Today: Proceedings*, 50:1511–1515, 2022. 2nd International Conference on Functional Material, Manufacturing and Performances (ICFMMP-2021).
- [34] Jian Li, Rui-Lin Chu, J. K. Jain, and Shun-Qing Shen. Topological anderson insulator. *Phys. Rev. Lett.*, 102:136806, Apr 2009.
- [35] C. W. Groth, M. Wimmer, A. R. Akhmerov, J. Tworzydło, and C. W. J. Beenakker. Theory of the topological anderson insulator. *Phys. Rev. Lett.*, 103:196805, Nov 2009.
- [36] Juntao Song, Haiwen Liu, Hua Jiang, Qing-feng Sun, and X. C. Xie. Dependence of topological anderson insulator on the type of disorder. *Phys. Rev. B*, 85:195125, May 2012.
- [37] Raymond Tung. The physics and chemistry of the schottky barrier height. *Appl. Phys. Rev.*, 1, 02 2014.
- [38] Wenhao Huang, Oliver Braun, David I. Indolese, Gabriela Borin Barin, Guido Gandus, Michael Stiefel, Antonis Olziersky, Klaus Müllen, Mathieu Luisier, Daniele Passerone, Pascal Ruffieux, Christian Schönenberger, Kenji Watanabe, Takashi Taniguchi, Roman Fasel, Jian Zhang, Michel Calame, and Mickael L. Perrin. Edge contacts to atomically precise graphene nanoribbons. *ACS Nano*, 17(19):18706–18715, 2023. PMID: 37578964.
- [39] H. Santos, L. Chico, and L. Brey. Carbon nanoelectronics: Unzipping tubes into graphene ribbons. *Phys. Rev. Lett.*, 103:086801, Aug 2009.

- [40] Gabriel T. Landi, Dario Poletti, and Gernot Schaller. Nonequilibrium boundary-driven quantum systems: Models, methods, and properties. *Rev. Mod. Phys.*, 94:045006, Dec 2022.
- [41] Nianbei Li, Jie Ren, Lei Wang, Gang Zhang, Peter Hänggi, and Baowen Li. Colloquium: Phononics: Manipulating heat flow with electronic analogs and beyond. *Rev. Mod. Phys.*, 84:1045–1066, Jul 2012.
- [42] J.B.J. Fourier. *Théorie analytique de la chaleur*. Landmarks of Science. Didot, 1822.
- [43] Giuliano Benenti, Giulio Casati, Carlos Mejía-Monasterio, and Michel Peyrard. *From Thermal Rectifiers to Thermoelectric Devices*, pages 365–407. Springer International Publishing, Cham, 2016.
- [44] M. Büttiker, Y. Imry, R. Landauer, and S. Pinhas. Generalized many-channel conductance formula with application to small rings. *Phys. Rev. B*, 31:6207–6215, May 1985.
- [45] Daniel S. Fisher and Patrick A. Lee. Relation between conductivity and transmission matrix. *Phys. Rev. B*, 23:6851–6854, Jun 1981.
- [46] Yigal Meir and Ned S. Wingreen. Landauer formula for the current through an interacting electron region. *Phys. Rev. Lett.*, 68:2512–2515, Apr 1992.
- [47] Felipe Amorim and Alexandre Reily Rocha. Generalized fisher-lee transmission for interacting systems. *in preparation*.
- [48] Luis E. F. Foa Torres, Stephan Roche, and Jean-Christophe Charlier. *Introduction to Graphene-Based Nanomaterials: From Electronic Structure to Quantum Transport*. Cambridge University Press, 2014.
- [49] Young-Woo Son, Marvin L. Cohen, and Steven G. Louie. Energy gaps in graphene nanoribbons. *Phys. Rev. Lett.*, 97:216803, Nov 2006.
- [50] Abdullah Anwar, Bashar Mohammed, Mubarak Wahab, and M.S. Liew. Enhanced properties of cementitious composite tailored with graphene oxide nanomaterial - a review. *Developments in the Built Environment*, 1:100002, 10 2019.

- [51] R Saito, G Dresselhaus, and M S Dresselhaus. *Physical Properties of Carbon Nanotubes*. PUBLISHED BY IMPERIAL COLLEGE PRESS AND DISTRIBUTED BY WORLD SCIENTIFIC PUBLISHING CO., 1998.
- [52] Léon Van Hove. The occurrence of singularities in the elastic frequency distribution of a crystal. *Phys. Rev.*, 89:1189–1193, Mar 1953.
- [53] Frank E. Harris George B. Arfken, Hans J. Weber. *Mathematical Methods for Physicists, Seventh Edition: A Comprehensive Guide*. Academic Press, 7 edition, 2012.
- [54] C Caroli, R Combescot, P Nozieres, and D Saint-James. Direct calculation of the tunneling current. *Journal of Physics C: Solid State Physics*, 4(8):916, jun 1971.
- [55] Mariana Odashima, Beatriz Prado, and Edson Vernek. Pedagogical introduction to equilibrium green's functions: Condensed-matter examples with numerical implementations. *Revista Brasileira de Ensino de Física*, 39:2017, 10 2016.
- [56] J. Hubbard. Electron correlations in narrow energy bands. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 276:238 – 257, 1963.
- [57] Th. Pruschke and R. Bulla. Hund's coupling and the metal-insulator transition in the two-band hubbard model. *The European Physical Journal B*, 44(2):217–224, mar 2005.
- [58] Sindhunil Barman Roy. Mott insulators and related phenomena: a basic introduction. In *Mott Insulators*, 2053-2563, pages 3–1 to 3–35. IOP Publishing, 2019.
- [59] Jun Kondo. Resistance Minimum in Dilute Magnetic Alloys. *Progress of Theoretical Physics*, 32(1):37–49, 07 1964.
- [60] K. Haule, S. Kirchner, J. Kroha, and P. Wölfle. Anderson impurity model at finite coulomb interaction u: Generalized noncrossing approximation. *Physical Review B*, 64(15), sep 2001.
- [61] Kristjan Haule, Chuck-Hou Yee, and Kyoo Kim. Dynamical mean-field theory within the full-potential methods: Electronic structure of CeIrIn_5 , CeCoIn_5 , and CeRhIn_5 . *Phys. Rev. B*, 81:195107, May 2010.

- [62] S E Barnes. New method for the anderson model. ii. the $u=0$ limit. *Journal of Physics F: Metal Physics*, 7(12):2637, dec 1977.
- [63] Piers Coleman. New approach to the mixed-valence problem. *Phys. Rev. B*, 29:3035–3044, Mar 1984.
- [64] Bruno Max de Souza Melo, Luis GGV Dias da Silva, Alexandre Reily Rocha, and Caio Lewenkopf. Qualitative comparison of anderson impurity solvers applied to transport in quantum dots. *Journal of Physics: Condensed Matter*, 32(9):095602, 2019.
- [65] C. N. Sposetti, L. O. Manuel, and P. Roura-Bas. Qualitative breakdown of the noncrossing approximation for the symmetric one-channel anderson impurity model at all temperatures. *Phys. Rev. B*, 94:085139, Aug 2016.
- [66] Andrea Droghetti and Ivan Rungger. Quantum transport simulation scheme including strong correlations and its application to organic radicals adsorbed on gold. *Phys. Rev. B*, 95:085131, Feb 2017.
- [67] Antoine Georges, Gabriel Kotliar, Werner Krauth, and Marcelo J. Rozenberg. Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions. *Rev. Mod. Phys.*, 68:13–125, Jan 1996.
- [68] Markus Büttiker. Four-terminal phase-coherent conductance. *Physical review letters*, 57 14:1761–1764, 1986.
- [69] M. Buttiker. Symmetry of electrical conduction. *IBM Journal of Research and Development*, 32(3):317–334, 1988.
- [70] M. Büttiker. Role of quantum coherence in series resistors. *Phys. Rev. B*, 33:3020–3026, Mar 1986.
- [71] C Caroli, R Combescot, D Lederer, P Nozieres, and D Saint-James. A direct calculation of the tunnelling current. ii. free electron description. *Journal of Physics C: Solid State Physics*, 4(16):2598, nov 1971.
- [72] Hartmut Haug and A. P. Jauho. Quantum kinetics in transport and optics of semiconductors. 2004.
- [73] Michael G. Cottam and Zahra Haghshenasfard. *Many-Body Theory of Condensed Matter Systems: An Introductory Course*. Cambridge University Press, 2020.

- [74] SUPARNA DUTTASINHA and Kostya S. Novoselov. The rise of graphene. *Nature materials*, 6 3:183–91, 2007.
- [75] Adrian Bachtold, Peter Hadley, Takeshi Nakanishi, and Cees Dekker. Logic circuits with carbon nanotube transistors. *Science*, 294(5545):1317–1320, 2001.
- [76] Xinran Wang, Yijian Ouyang, Xiaolin Li, Hailiang Wang, Jing Guo, and Hongjie Dai. Room-temperature all-semiconducting sub-10-nm graphene nanoribbon field-effect transistors. *Phys. Rev. Lett.*, 100:206803, May 2008.
- [77] F. Bonaccorso, Z. Sun, T. Hasan, and A. C. Ferrari. Graphene photonics and optoelectronics. *Nature Photonics*, 4(9):611–622, aug 2010.
- [78] Cesar E. P. Villegas, P. B. Mendonça, and A. R. Rocha. Optical spectrum of bottom-up graphene nanoribbons: towards efficient atom-thick excitonic solar cells. *Scientific Reports*, 4(1):6579, October 2014. Number: 1 Publisher: Nature Publishing Group.
- [79] S. Das Sarma, Shaffique Adam, E. H. Hwang, and Enrico Rossi. Electronic transport in two-dimensional graphene. *Rev. Mod. Phys.*, 83:407–470, May 2011.
- [80] Vivian M. de Menezes, Alexandre R. Rocha, Ivana Zanella, Ronaldo Mota, Adalberto Fazzio, and Solange B. Fagan. Electronic transport properties of ascorbic acid and nicotinamide adsorbed on single-walled carbon nanotubes. *Chemical Physics Letters*, 506(4):233–238, 2011.
- [81] Ernane de Freitas Martins, Rodrigo G. Amorim, Gustavo Troiano Feliciano, Ralph Hendrik Scheicher, and Alexandre Reily Rocha. The role of water on the electronic transport in graphene nanogap devices designed for dna sequencing. *Carbon*, 158:314–319, 2020.
- [82] Mir Mohammad Sadeghi, Michael Thompson Pettes, and Li Shi. Thermal transport in graphene. *Solid State Communications*, 152(15):1321–1330, 2012. Exploring Graphene, Recent Research Advances.
- [83] Chun-Yu Lin, Zhenghang Zhao, Jianbing Niu, and Zhenhai Xia. Synthesis, properties and applications of 3d carbon nanotube–graphene junctions. *Journal of Physics D: Applied Physics*, 49(44):443001, oct 2016.

- [84] Dong-Keun Ki and Alberto F. Morpurgo. Crossover from coulomb blockade to quantum hall effect in suspended graphene nanoribbons. *Phys. Rev. Lett.*, 108:266601, Jun 2012.
- [85] C. Stampfer, J. Güttinger, F. Molitor, D. Graf, T. Ihn, and K. Ensslin. Tunable coulomb blockade in nanostructured graphene. *Applied Physics Letters*, 92(1):012102, 2008.
- [86] Y Claveau, B Arnaud, and S Di Matteo. Mean-field solution of the hubbard model: the magnetic phase diagram. *European Journal of Physics*, 35(3):035023, apr 2014.
- [87] Lorenz Bartosch, Hermann Freire, Jose Juan Ramos Cardenas, and Peter Kopietz. A functional renormalization group approach to the anderson impurity model. *Journal of Physics: Condensed Matter*, 21(30):305602, jul 2009.
- [88] A C Hewson and D Meyer. Numerical renormalization group study of the anderson-holstein impurity model. *Journal of Physics: Condensed Matter*, 14(3):427–445, dec 2001.
- [89] Bruno Max de Souza Melo, Luis G. G. V. Dias da Silva, Alexandre Reily Rocha, and Caio Lewenkopf. Quantitative comparison of Anderson impurity solvers applied to transport in quantum dots. *Journal of Physics: Condensed Matter*, 32(9):095602, December 2019. Publisher: IOP Publishing.
- [90] A. R. Rocha, V. M. García-Suárez, S. Bailey, C. Lambert, J. Ferrer, and S. Sanvito. Spin and molecular electronics in atomically generated orbital landscapes. *Phys. Rev. B*, 73:085414, Feb 2006.
- [91] C Caroli, R Combescot, D Lederer, P Nozieres, and D Saint-James. A direct calculation of the tunnelling current. ii. free electron description. *Journal of Physics C: Solid State Physics*, 4(16):2598, nov 1971.
- [92] Fabricio S Lozano-Negro, Marcos A Ferreyra-Ortega, Denise Bendersky, Lucas Fernández-Alcázar, and Horacio M Pastawski. Simulating a catalyst induced quantum dynamical phase transition of a heyrovsky reaction with different models for the environment. *Journal of Physics: Condensed Matter*, 34(21):214006, mar 2022.

- [93] Mehmet Gokhan Sensoy. The influence of vacancy-induced local strain on the transport properties in armchair and zigzag graphene nanoribbons. *Materials Research Express*, 6(4):045057, jan 2019.
- [94] Gerald Schubert, Jens Schleede, and Holger Fehske. Anderson disorder in graphene nanoribbons: A local distribution approach. *Phys. Rev. B*, 79:235116, Jun 2009.
- [95] P. W. Anderson. Absence of diffusion in certain random lattices. *Phys. Rev.*, 109:1492–1505, Mar 1958.
- [96] Melinda Y. Han, Juliana C. Brant, and Philip Kim. Electron transport in disordered graphene nanoribbons. *Phys. Rev. Lett.*, 104:056801, Feb 2010.
- [97] Kenneth G. Wilson. The renormalization group: Critical phenomena and the kondo problem. *Rev. Mod. Phys.*, 47:773–840, Oct 1975.
- [98] G. Kotliar, Sahana Murthy, and M. J. Rozenberg. Compressibility Divergence and the Finite Temperature Mott Transition. *Physical Review Letters*, 89(4):046401, July 2002. Publisher: American Physical Society.
- [99] S Datta. *Electronic Transport in Mesoscopic Systems*. Cambridge University Press, Cambridge, 1 edition, 1995.
- [100] A. R. Rocha, V. García-Suárez, S. Bailey, C. Lambert, J. Ferrer, and S. Sanvito. Spin and molecular electronics in atomically generated orbital landscapes. *Phys. Rev. B*, 73(8), February 2006.
- [101] Mads Brandbyge, José-Luis Mozos, Pablo Ordejón, Jeremy Taylor, and Kurt Stokbro. Density-functional method for nonequilibrium electron transport. *Phys. Rev. B*, 65:165401, Mar 2002.
- [102] Fabricio S. Lozano-Negro, Emilio Alvarez Navarro, Nahum C. Chávez, Francesco Mattiotti, Fausto Borgonovi, Horacio M. Pastawski, and G. Luca Celardo. Universal stability towards decoherence in quantum diffusive 1d chains. 2023.
- [103] Abhijit Kar. *Nanoelectronics and Materials Development*. ExLi4EvA, 2016.
- [104] Eduardo Gracia-Espino, Florentino López-Urías, Yehrin Kim, T. Hayashi, H. Muramatsu, M. Endo, H. Terrones, Mauricio Terrones, and Mildred

- Dresselhaus. Novel carbon-based nanomaterials graphene and graphitic nanoribbons[1], 08 2015.
- [105] Zhe Kan, Mahfuza Khatun, and Antonio Cancio. Quantum transport in zigzag graphene nanoribbons in the presence of vacancies. *Journal of Applied Physics*, 125(16):164305, 04 2019.

Apêndice A

Script

Neste apêndice, apresentamos a biblioteca construída como um dos resultados desta dissertação. Assim, aqui está presente o código em Python por extenso.

```
1
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat May 22 19:38:34 2021
5
6 @author: Washington
7 """
8
9
10 import numpy as np
11 import random
12 import matplotlib.pyplot as plt
13 import matplotlib.cm as cm
14 #import matplotlib as mt
15 from numba import njit
16 from sympy.matrices import Matrix
17 from mayavi import mlab
18
19
20 @njit(nogil=True)
21 def fitaZ(k,N,gamma,e0,e1):
22     #aneis = 2
23     #N = aneis*4
24     #e0 = 1
25     #gamma = 2
26     #k = 1
27     #j = complex(0,1)
28     #h0 = np.dot(np.identity(2*N,complex),0)
29     #h1 = np.dot(np.identity(2*N,complex),0)
30     #h2 = np.dot(np.identity(2*N,complex),0)
31     h0 = np.zeros((2*N,2*N),dtype=np.complex_)
32     h1 = np.zeros((2*N,2*N),dtype=np.complex_)
```

```

33     h2 = np.zeros((2*N,2*N),dtype=np.complex_)
34     b = 0
35     #vi = np.dot(np.identity(2*N,complex),0)
36     h0[1][0] = gamma*np.exp(-1j*k)
37     h2[0][1] = gamma*np.exp(1j*k)
38     for i3 in np.arange(1,2*N+1):
39         if i3%2 != 0:
40             h1[(i3-1)][(i3-1)] = e0
41         else:
42             h1[(i3-1)][(i3-1)] = e1
43         if i3 > 1:
44             h1[(i3-1)][(i3-1)-1] = gamma
45         if i3 < 2*N:
46             h1[(i3-1)][(i3-1)+1] = gamma
47         if i3 > 2:
48             if b == 0:
49                 #licacaoentre+
50                 if i3%2 != 0:
51                     h0[(i3-1)][(i3-1)+1] = gamma*np.exp(-1j*k)
52                 else:
53                     h2[(i3-1)][(i3-1)-1] = gamma*np.exp(1j*k)
54                     b =1
55             else:
56                 if i3%2 == 0:
57                     h0[(i3-1)][(i3-1)-1] = gamma*np.exp(-1j*k)
58                     b = 0
59                 else:
60                     h2[(i3-1)][(i3-1)+1] = gamma*np.exp(1j*k)
61
62
63     #H = h0+h1+h2
64     return (h0,h1,h2)
65
66 #
=====
67 # @njit(nogil=True)
68 # def fitaA(k,N,gamma,e0,e1,b=0,res_hop=0):
69 #     #aneis = 2
70 #     #N = 2+aneis*4
71 #     #e0 = 1
72 #     #e1
73 #     #gamma = 2

```

```

74 #     #k = 1
75 #     if N<2:
76 #         N=2
77 #     #j = complex(0,1)
78 #     #h0 = np.dot(np.identity(2*N,complex),0)
79 #     #h1 = np.dot(np.identity(2*N,complex),0)
80 #     #h2 = np.dot(np.identity(2*N,complex),0)
81 #     h0 = np.zeros((2*N,2*N),dtype=np.complex_)
82 #     h1 = np.zeros((2*N,2*N),dtype=np.complex_)
83 #     h2 = np.zeros((2*N,2*N),dtype=np.complex_)
84 #     #b = 0
85 #
86 #     hidro_rescala = 0.12
87 #
88 #     for i4 in np.arange(1,2*N+1):
89 #         #print(i)
90 #         if b == 0:
91 #             if i4%2!=0:
92 #                 h1[(i4-1)][(i4-1)] = e0
93 #                 #ligacaoentrecel-1
94 #                 h0[(i4-1)][(i4-1)+1] = gamma*np.exp(-1j*k)
95 #                 if i4 == 1:
96 #                     h0[(i4-1)][(i4-1)+1] = gamma*np.exp(-1j*k) +
97 #                     hidro_rescala*gamma*np.exp(-1j*k)
98 #                 else:
99 #                     h1[(i4-1)][(i4-1)] = e1
100 #                     #ligacaoentrecel+1
101 #                     h2[(i4-1)][(i4-1)-1] = gamma*np.exp(1j*k)
102 #                     if i4 == 2:
103 #                         h2[(i4-1)][(i4-1)-1] = gamma*np.exp(1j*k) +
104 #                         hidro_rescala*gamma*np.exp(-1j*k)
105 #                     b = 1
106 #             else:
107 #                 if i4%2 != 0:
108 #                     h1[(i4-1)][(i4-1)] = e1
109 #                     #ligacao-somente-x
110 #                     h1[(i4-1)][(i4-1)+1] = gamma
111 #                 else:
112 #                     h1[(i4-1)][(i4-1)] = e0
113 #                     #ligacao-somente-x
114 #                     h1[(i4-1)][(i4-1)-1] = gamma
115 #                     b = 0
116 #             if i4 < 2*N-1:

```



```

115 #             h1[(i4-1)][(i4-1)+2] = gamma
116 #             if i4 > 2:
117 #                 h1[(i4-1)][(i4-1)-2] = gamma
118 #
119 #             #H = h0 + h1 + h2
120 #             return (h0,h1,h2)
121 #
122 #
=====

123
124
125 @njit(nogil=True)
126 def fitaA(k,N,gamma,e0,e1,b=0,res_hop=0):
127     #aneis = 2
128     #N = 2+aneis*4
129     #e0 = 1
130     #e1
131     #gamma = 2
132     #k = 1
133     if N<2:
134         N=2
135     #j = complex(0,1)
136     #h0 = np.dot(np.identity(2*N,complex),0)
137     #h1 = np.dot(np.identity(2*N,complex),0)
138     #h2 = np.dot(np.identity(2*N,complex),0)
139     h0 = np.zeros((2*N,2*N),dtype=np.complex_)
140     h1 = np.zeros((2*N,2*N),dtype=np.complex_)
141     h2 = np.zeros((2*N,2*N),dtype=np.complex_)
142     #b = 0
143     hidro_rescala = res_hop
144
145     for i4 in np.arange(1,2*N+1):
146         #print(i)
147         if b == 0:
148             if i4%2!=0:
149                 h1[(i4-1)][(i4-1)] = e0
150                 #ligacaoentrecel-1
151                 h0[(i4-1)][(i4-1)+1] = gamma
152
153                 if i4 == 1 or i4 == 2*N-1 or i4 == 2*N :
154                     h0[(i4-1)][(i4-1)+1] = gamma + hidro_rescala*gamma
155

```

```

156         else:
157             h1[(i4-1)][(i4-1)] = e1
158             #ligacaoentrecel+1
159             h2[(i4-1)][(i4-1)-1] = gamma
160
161             if i4 == 2 or i4 == 2*N-1 or i4 == 2*N :
162                 h2[(i4-1)][(i4-1)-1] = gamma + hidro_rescala*gamma
163                 b = 1
164
165         else:
166             if i4%2 != 0:
167                 h1[(i4-1)][(i4-1)] = e1
168                 #ligacao-somente-x
169                 h1[(i4-1)][(i4-1)+1] = gamma
170                 if i4 == 1 or i4 == 2*N-1 or i4 == 2*N:
171                     h1[(i4-1)][(i4-1)+1] = gamma + hidro_rescala*gamma
172
173             else:
174                 h1[(i4-1)][(i4-1)] = e0
175                 #ligacao-somente-x
176                 h1[(i4-1)][(i4-1)-1] = gamma
177                 if i4 == 2 or i4 == 2*N-1 or i4 == 2*N:
178                     h1[(i4-1)][(i4-1)-1] = gamma + hidro_rescala*gamma
179                 b = 0
180             if i4 < 2*N-1:
181                 h1[(i4-1)][(i4-1)+2] = gamma
182
183             if i4 > 2:
184                 h1[(i4-1)][(i4-1)-2] = gamma
185
186         #H = h0 + h1 + h2
187         return (h0*np.exp(-1j*k), h1, h2*np.exp(1j*k))
188
189
190 @njit(nogil=True)
191 def G(g,v):
192     G = np.linalg.inv(np.linalg.inv(g)-v)
193     return G
194
195 @njit(nogil=True)
196 def G1(H,vl,vr,E,erro):
197     a = len(H[0])
198     E1 = np.eye(a, dtype=np.complex_)

```

```

199     E1 = E*E1#np.dot(E,E1)
200     inH = np.linalg.inv(E1-H)
201     #V = v1 + vr
202     #zero = np.zeros((a, a),dtype=np.complex_)
203
204     G0l = np.zeros((2*a, 2*a),dtype=np.complex_)
205     G0l[:a, :a] = inH#np.block([[inH,zero],[zero,inH]])
206     G0l[a:, a:] = inH
207     V1 = np.zeros((2*a, 2*a),dtype=np.complex_)
208     V1[:a, a:] = vr#np.block([[zero,vr],[v1,zero]])
209     V1[a:, :a] = v1
210
211     Gn1 = G(G0l,V1)
212     gnl= np.zeros((a, a),dtype=np.complex_)
213
214     G0r = np.zeros((2*a, 2*a),dtype=np.complex_)
215     G0r[:a, :a] = inH#np.block([[inH,zero],[zero,inH]])
216     G0r[a:, a:] = inH
217     Vr = np.zeros((2*a, 2*a),dtype=np.complex_)
218     Vr[:a, a:] = vr#np.block([[zero,vr],[v1,zero]])
219     Vr[a:, :a] = v1
220
221     Gnr = G(G0r,Vr)
222     gnr= np.zeros((a, a),dtype=np.complex_)
223     for i5 in np.arange(a):
224         for j5 in np.arange(a):
225             gnl[i5][j5] = Gn1[i5+a][j5+a]
226             gnr[i5][j5] = Gnr[i5][j5]
227
228     for k2 in np.arange(200000):
229         err = np.trace(gnl)
230         G0l = np.zeros((2*a, 2*a),dtype=np.complex_)
231         G0l[:a, :a] = gnl
232         G0l[a:, a:] = inH
233         #G0l = np.block([[gnl,zero],[zero,inH]])
234
235         Gn1 = G(G0l,V1)
236         gnl = np.zeros((a, a),dtype=np.complex_)
237
238         G0r = np.zeros((2*a, 2*a),dtype=np.complex_)
239         #G0r = np.block([[inH,zero],[zero,gnr]])
240         G0r[:a, :a] = inH
241         G0r[a:, a:] = gnr

```

```

242
243     Gnr = G(G0r,Vr)
244     gnr = np.zeros((a, a),dtype=np.complex_)
245     for i in np.arange(a):
246         for j in np.arange(a):
247             gnl[i][j] = Gnl[i+a][j+a]
248             gnr[i][j] = Gnr[i][j]
249     if np.abs(err-np.trace(gnl)) <= erro:
250         #print("1 - erro = %.3f"%abs(err-np.trace(gnr)))
251         #print("2 - N   de intera   es = %d"%(k2))
252         #print("3 -energia = %.4f"%(E.real))
253         break
254     else:
255         continue
256
257     #Gf = np.block([[gnl,zero,zero],[zero,inH,zero],[zero,zero,gnr]])
258     #Gf = np.zeros((3*a, 3*a),dtype=np.complex_)
259     #Gf[:a, :a] = gnl
260     #Gf[a:, a:] = inH
261     #Gf[2*a:, 2*a:] = gnr
262
263     #vf = np.block([[zero,vr,zero],[vl,zero,vr],[zero,vl,zero]])
264     #vf = np.zeros((3*a, 3*a),dtype=np.complex_)
265     #vf[:a, a:] = vr
266     #vf[a:, :a] = vl
267     #vf[a:, 2*a:] = vr
268     #vf[2*a:, a:] = vl
269
270
271     #Gf = G(Gf,vf)
272     #green = np.zeros((a, a),dtype=np.complex_)
273     #for i1 in np.arange(a):
274     #    for j1 in np.arange(a):
275     #        green[i1][j1] = Gf[i1+a][j1+a]
276
277     return gnl,gnr
278
279
280 def GL(H1,vl,vr,E,L):
281     a = len(H1[0])
282     E1 = np.identity(a,complex)
283     E1 = np.dot(E,E1)
284     inH = np.linalg.inv(E1-H1)

```

```

285     #V = vl + vr
286     zero = np.zeros((a, a), complex)
287
288     G0l2 = np.block([[inH, zero], [zero, inH]])
289     Vl2 = np.block([[zero, vr], [vl, zero]])
290     Gn12 = G(G0l2, Vl2)
291     gnl2 = np.zeros((a, a), complex)
292
293     #G0r = np.block([[inH, zero], [zero, inH]])
294     #Vr = np.block([[zero, vr], [vl, zero]])
295     #Gnr = G(G0r, Vr)
296     #gnr = np.zeros((a, a), complex)
297     for i6 in range(a):
298         for j6 in range(a):
299             gnl2[i6][j6] = Gn12[i6+a][j6+a]
300             #gnr[i5][j5] = Gnr[i5][j5]
301
302
303     G0l2 = np.block([[gnl2, zero], [zero, inH]])
304     Gn12 = G(G0l2, Vl2)
305     gnl2 = np.zeros((a, a), complex)
306
307     #G0r = np.block([[inH, zero], [zero, gnr]])
308     #Gnr = G(G0r, Vr)
309     #gnr = np.zeros((a, a), complex)
310     for i in range(a):
311         for j in range(a):
312             gnl2[i][j] = Gn12[i+a][j+a]
313             #gnr[i][j] = Gnr[i][j]
314
315     Gf = np.block([[gnl2, zero, zero], [zero, inH]])
316     vf = np.block([[zero, vr], [vl, zero]])
317     Gf = G(Gf, vf)
318     #green = np.zeros((a, a), complex)
319     #for i1 in range(a):
320     #     for j1 in range(a):
321     #         green[i1][j1] = Gf[i1+a][j1+a]
322
323     return Gf
324 #
=====
325 #

```

```

326 # def HL(L,N,h,vl,vr):
327 #     #N = 3
328 #     #L = 10
329 #     #vl,h,vr = fitaZ(0,N,1,0,0)
330 #     h1 = h
331 #     h0 = h
332 #     for i in np.arange(L-1):
333 #         a = i*len(h)
334 #         zeror = np.zeros((a, len(h1)),dtype=np.complex_)
335 #         zerol = np.zeros((len(h1), a),dtype=np.complex_)
336 #         #Vr = np.block([[zeror],[vr]])
337 #         #Vl = np.block([[zerol,vl]])
338 #         #h0 = np.block([[h0,Vr],[Vl,h1]])
339 #         Vr = np.block([[zeror],[vr]])
340 #         Vl = np.block([[zerol,vl]])
341 #         h0 = np.block([[h0,Vr],[Vl,h1]])
342 #
343 #         #print(i)
344 #     return h0
345 #

```

```

346
347 @njit(nogil=True)
348 def HL(L,N,h,vl,vr):
349     #N = 3
350     #L = 10
351     #vl,h,vr = fitaZ(0,N,1,0,0)
352     h1 = h
353     h0 = h
354     d1 = len(h1)
355     d2 = len(vr)
356     d3 = len(vl)
357     for i in np.arange(L-1):
358         a = i*len(h)
359         #zeror = np.zeros((a, d1),dtype=np.complex_)
360         #zerol = np.zeros((d1, a),dtype=np.complex_)
361         #Vr = np.block([[zeror],[vr]])
362         #Vl = np.block([[zerol,vl]])
363         #h0 = np.block([[h0,Vr],[Vl,h1]])
364         Vr = np.zeros((a+d2, d1),dtype=np.complex_)
365         Vr[a:, :d2] = vr
366

```

```

367     V1 = np.zeros((d1, a+d3), dtype=np.complex_)
368     V1[:d3, a:] = v1
369
370     dh = len(h0)
371     h2 = np.zeros((h0.shape[0]+V1.shape[0], h0.shape[1]+Vr.shape
372 [1]), dtype=np.complex_)
373     h2[:h0.shape[0], :h0.shape[1]] = h0
374     h2[:h0.shape[0], h0.shape[1]:] = Vr
375     h2[h0.shape[0]:, :h0.shape[1]] = V1
376     h2[h0.shape[0]:, h0.shape[1]:] = h1
377
378     h0 = h2[:, :]
379
380     #h0 = np.block([[h0, Vr], [V1, h1]])
381     if L == 0:
382         h0 = np.zeros((0, 0), dtype=np.complex_)
383     return h0
384
385 def Hd(L, N, E):
386     #N = 3
387     #L = 10
388     vl, h, vr = fitaZ(0, N, 1, 0, 0)
389     h1 = h
390     hn = h
391     hi = h
392     #a = len(h)
393     vri = vr
394     vli = vl
395     for i in range(100):
396         print(i)
397         h1 = h1 + np.dot(vri, np.dot(np.linalg.inv(E-hn), vli))
398         hn = h + np.dot(vl, np.dot(np.linalg.inv(E-hn), vr))
399         vri = np.dot(vri, np.dot(np.linalg.inv(E-hi), vr))
400         vli = np.dot(vl, np.dot(np.linalg.inv(E-hi), vli))
401
402         hi = hn
403         #print(i)
404         h0 = np.block([[h1, vri], [vli, hn]])
405
406     return h0
407
408 #@njit(nogil=True)

```

```

409 def Exm(Ter_a, Ter_el, Ter_i, Ter_er, Ter_b, Nt, Lt, V_ex, L_ex):
410
411     Ha, Hel, Hi, Her, Hb = Ter_a[0], Ter_el[0], Ter_i[0], Ter_er[0], Ter_b
412     [0]
413
414     vla, vlel, vli, vler, vlb = Ter_a[1], Ter_el[1], Ter_i[1], Ter_er[1],
415     Ter_b[1]
416
417     vra, vrel, vri, vrer, vrb = Ter_a[2], Ter_el[2], Ter_i[2], Ter_er[2],
418     Ter_b[2]
419
420     Na, Nel, Ni, Ner, Nb = Nt[0], Nt[1], Nt[2], Nt[3], Nt[4]
421
422     La, Lel, Li, Ler, Lb = Lt[0], Lt[1], Lt[2], Lt[3], Lt[4]
423
424     ha = HL(La, Na, Ha, vla, vra)
425     hel = HL(Lel, Nel, Hel, vlel, vrel)
426     hi = HL(Li, Ni, Hi, vli, vri)
427     her = HL(Ler, Ner, Her, vler, vrer)
428     hb = HL(Lb, Nb, Hb, vlb, vrb)
429
430     na, nel, ni, ner, nb = len(Ha), len(Hel), len(Hi), len(Her), len(Hb)
431
432     adin, eldin, idin, erdin, bdin = na*La, nel*Lel, ni*Li, ner*Ler, nb*Lb
433
434     N = adin+eldin+idin+erdin+bdin
435
436     H = np.zeros((N,N), dtype=np.complex_)
437
438     H[:adin, :adin] = ha
439     H[adin:adin+eldin, adin:adin+eldin] = hel
440     H[adin+eldin:adin+eldin+idin, adin+eldin:adin+eldin+idin] = hi
441     H[adin+eldin+idin:adin+eldin+idin+erdin, adin+eldin+idin:adin+eldin+
442     idin+erdin] = her
443
444     H[N-bdin:N, N-bdin:N] = hb
445
446     Idin = (adin+eldin, adin+eldin+idin)
447
448     La_el, Lel_i, Li_er, Ler_b = L_ex[0], L_ex[1], L_ex[2], L_ex[3]

```



```

448
449     va_el, vel_i, vi_er, ver_b = V_ex[0],V_ex[1],V_ex[2],V_ex[3]
450
451
452
453     if La == 0:
454         adin = eldin
455     if Lel == 0:
456         eldin = idin
457     if Li == 0:
458         idin = erdin
459     if Ler == 0:
460         erdin = bdin
461     if Lb == 0:
462         bdin = erdin
463
464
465
466
467
468     Va_el, Vel_i, Vi_er, Ver_b = np.zeros((adin,eldin),dtype=np.
complex_),np.zeros((eldin,idin),dtype=np.complex_),np.zeros((idin,
erdin),dtype=np.complex_),np.zeros((erdin,bdin),dtype=np.complex_)
469     #print(Va_el.shape, Vel_i.shape, Vi_er.shape, Ver_b.shape)
470     #print(Va_el.shape[0],Va_el.shape[1])
471     #print(ver_b.shape[0],Ver_b.shape[0])
472
473
474
475
476
477
478     for la_el in np.arange(La_el):
479         Va_el[adin-(La_el-la_el)*na:adin-(La_el-la_el)*na+va_el.shape
[0],la_el*nel:la_el*nel+va_el.shape[1]] = va_el
480
481     for lel_i in np.arange(Lel_i):
482         Vel_i[eldin-(Lel_i-lel_i)*nel:eldin-(Lel_i-lel_i)*nel+vel_i.
shape[0],lel_i*ni:lel_i*ni+vel_i.shape[1]] = vel_i
483
484     for li_er in np.arange(Li_er):
485         Vi_er[idin-(Li_er-li_er)*ni:idin-(Li_er-li_er)*ni+vi_er.shape
[0],li_er*ner:li_er*ner+vi_er.shape[1]] = vi_er

```

```

486
487
488     for ler_b in np.arange(Ler_b):
489         Ver_b[erdin-(Ler_b-ler_b)*ner:erdin-(Ler_b-ler_b)*ner+ver_b.
shape[0],ler_b*nb:ler_b*nb+ver_b.shape[1]] = ver_b
490
491
492
493     if La == 0:
494         Va_el = np.zeros((0,0),dtype=np.complex_)
495     if Lel == 0:
496         Vel_i = np.zeros((0,0),dtype=np.complex_)
497     if Ler == 0:
498         Vi_er = np.zeros((0,0),dtype=np.complex_)
499     if Lb == 0:
500         Ver_b = np.zeros((0,0),dtype=np.complex_)
501
502
503
504     #adin,eldin,idin,erdin,bdin = na*La,nel*Lel,ni*Li,ner*Ler,nb*Lb
505
506     #Vel_i[eldin-nel:eldin,:ni] = vel_i
507
508     #Vi_er[idin-ni:idin,:ner] = vi_er
509
510     #Ver_b[erdin-ner:erdin,:nb] = ver_b
511
512
513     #adin,eldin,idin,erdin,bdin = na*La,nel*Lel,ni*Li,ner*Ler,nb*Lb
514
515     #
=====
516     # H[:Va_el.shape[0],adin:adin+Va_el.shape[1]] = Va_el
517     # H[adin:adin+Va_el.shape[1],:Va_el.shape[0]] = (Va_el).T
518     #
519     # H[adin:adin+Vel_i.shape[0],adin+eldin:adin+eldin+Vel_i.shape[1]]
= Vel_i
520     # H[adin+eldin:adin+eldin+Vel_i.shape[1],adin:adin+Vel_i.shape[0]]
= (Vel_i).T
521     #
522     # H[adin+eldin:adin+eldin+Vi_er.shape[0],adin+eldin+idin:adin+
eldin+idin+Vi_er.shape[1]] = Vi_er

```

```

523 #     #H[adin+eldin+idin:adin+eldin+idin+Vi_er.shape[1],adin+eldin:adin
    +eldin+Vi_er.shape[0]] = (Vi_er).T
524 #
525 #     #H[adin+eldin+idin:adin+eldin+idin+Ver_b.shape[0],adin+eldin+idin
    +erdin:adin+eldin+idin+erdin+Ver_b.shape[1]] = Ver_b
526 #     #H[adin+eldin+idin+erdin:adin+eldin+idin+erdin+Ver_b.shape[1],
    adin+eldin+idin:adin+eldin+idin+Ver_b.shape[0]] = (Ver_b).T
527 #
528 #
    =====

529
530 H[:,Va_el.shape[0],adin:adin+Va_el.shape[1]] = Va_el
531 H[adin:adin+Va_el.shape[1],:Va_el.shape[0]] = (Va_el).T
532
533
534 H[adin:adin+Vel_i.shape[0],adin+eldin:adin+eldin+Vel_i.shape[1]] =
    Vel_i
535 H[adin+eldin:adin+eldin+Vel_i.shape[1],adin:adin+Vel_i.shape[0]] =
    (Vel_i).T
536
537 H[adin+eldin:adin+eldin+Vi_er.shape[0],adin+eldin+idin:adin+eldin+
    idin+Vi_er.shape[1]] = Vi_er
538 H[adin+eldin+idin:adin+eldin+idin+Vi_er.shape[1],adin+eldin:adin+
    eldin+Vi_er.shape[0]] = (Vi_er).T
539
540
541 H[N-bdin-erdin:(N-bdin-erdin)+Ver_b.shape[0],N-bdin:N-bdin+Ver_b.
    shape[1]] = Ver_b
542 H[N-bdin:N-bdin+Ver_b.shape[1],N-bdin-erdin:(N-bdin-erdin)+Ver_b.
    shape[0]] = (Ver_b).T
543
544
545 return H,hi,Idin
546
547
548
549 @njit(nogil=True)
550 def Reduce(N,L,E,h):
551     #N = 2
552     #L = 3
553     #L = 5
554     #v1,h0,vr = fitaZ(0,N,1,0,0)

```

```

555     #h = HL(L,N,h0,vl,vr)
556     #print(h)
557     dh = len(h)
558     n = 2*N#len(h0)
559
560     #e = 1+1j*0.1
561     #E1 = np.identity(n,complex)
562     #E = np.dot(e,E1)
563
564     h1 = h[:n,:n]
565     hn = h[n:2*n,n:2*n]
566     hi = hn
567
568     vri = h[0*n:n,n:2*n]
569     vli = h[n:2*n,:n]
570
571
572
573     #print(vri == vr)
574     #print(h1 == h0)
575     #print(h[2*n:3*n,2*n:3*n] == h0)
576     #print(vli)
577
578     #print(vri == vr2)
579     #print(vli == vl2)
580     #print(L*N)
581     for i in np.arange(1,L-1):
582         #print(i)
583         vr2 = h[i*n:(i+1)*n,(i+1)*n:(i+2)*n]
584         vl2 = h[(i+1)*n:(i+2)*n,i*n:(i+1)*n]
585         #print(vr == vr2)
586         invhn = np.linalg.inv(E-hn)
587         #print(vri,invhn,vli)
588         h1 = h1 + np.dot(vri,np.dot(invhn,vli))
589         hn = h[(1+i)*n:(2+i)*n,(1+i)*n:(2+i)*n] + np.dot(vl2,np.dot(
invhn,vr2))
590
591         invhi = np.linalg.inv(E-hi)
592         vri = np.dot(vri,np.dot(invhi,vr2))
593         vli = np.dot(vl2,np.dot(invhi,vli))
594
595         hi = hn
596         #print(i)

```

```

597     hf = np.zeros((2*len(h1),2*len(h1)),dtype=np.complex_)
598     hf[:len(h1),:len(h1)] = h1
599     hf[:len(h1),len(h1):] = vri
600     hf[len(h1):,:len(h1)] = vli
601     hf[len(h1):,len(h1):] = hn
602
603     #hf = np.block([[h1,vri],[vli,hn]])
604     #print(N)
605     #print(len(hf))
606     return hf
607
608
609
610
611
612 def ReduceAI(v,L,Ai,E,h):
613     #N = 2
614     #L = 3
615     #L = 5
616     #vl,h0,vr = fitaZ(0,N,1,0,0)
617     #h = HL(L,N,h0,vl,vr)
618     #print(h)
619     dh = len(h)
620     n = len(v)
621
622     #e = 1+1j*0.1
623     #E1 = np.identity(n,complex)
624     #E = np.dot(e,E1)
625
626     h1 = h[:n,:n]
627     hn = h[n:2*n,n:2*n]
628     hi = hn
629
630     vri = h[0*n:n,n:2*n]
631     vli = h[n:2*n,:n]
632
633
634
635     #print(vri == vr)
636     #print(h1 == h0)
637     #print(h[2*n:3*n,2*n:3*n] == h0)
638     #print(vli)
639

```

```

640     #print(vri == vr2)
641     #print(vli == vl2)
642     #print(L*N)
643     for i in range(1,Ai-1):
644         print(i)
645         vr2 = h[i*n:(i+1)*n, (i+1)*n:(i+2)*n]
646         vl2 = h[(i+1)*n:(i+2)*n, i*n:(i+1)*n]
647         #print(vr == vr2)
648         h1 = h1 + np.dot(vri, np.dot(np.linalg.inv(E-hn), vli))
649         hn = h[(1+i)*n:(2+i)*n, (1+i)*n:(2+i)*n] + np.dot(vl2, np.dot(np.
linalg.inv(E-hn), vr2))
650
651         vri = np.dot(vri, np.dot(np.linalg.inv(E-hi), vr2))
652         vli = np.dot(vl2, np.dot(np.linalg.inv(E-hi), vli))
653
654         hi = hn
655         #print(i)
656     hf = np.block([[h1, vri], [vli, hn]])
657     #print(N)
658     #print(len(hf))
659     return hf
660
661
662
663
664 #
=====
665 #
666 #
667 # def ReduceAI(N,L,E,h,pos):
668 #     #N = 2
669 #     #L = 3
670 #     #L = 5
671 #     #vl,h0,vr = fitaZ(0,N,1,0,0)
672 #     #h = HL(L,N,h0,vl,vr)
673 #     #print(h)
674 #     dh = len(h)
675 #     n = 2*N#len(h0)
676 #
677 #     #e = 1+1j*0.1
678 #     #E1 = np.identity(n,complex)
679 #     #E = np.dot(e,E1)

```

```

680 #
681 #     h1 = h[:n,:n]
682 #     hn = h[n:2*n,n:2*n]
683 #     hi = hn
684 #
685 #     vri = h[0*n:n,n:2*n]
686 #     vli = h[n:2*n,:n]
687 #
688 #
689 #
690 #     #print(vri == vr)
691 #     #print(h1 == h0)
692 #     #print(h[2*n:3*n,2*n:3*n] == h0)
693 #     #print(vli)
694 #
695 #     #print(vri == vr2)
696 #     #print(vli == vl2)
697 #     #print(L*N)
698 #     for i in range(1,pos-1):
699 #         #print(i)
700 #         vr2 = h[i*n:(i+1)*n,(i+1)*n:(i+2)*n]
701 #         vl2 = h[(i+1)*n:(i+2)*n,i*n:(i+1)*n]
702 #         #print(vr == vr2)
703 #         h1 = h1 + np.dot(vri,np.dot(np.linalg.inv(E-hn),vli))
704 #         hn = h[(1+i)*n:(2+i)*n,(1+i)*n:(2+i)*n] + np.dot(vl2,np.dot(
np.linalg.inv(E-hn),vr2))
705 #
706 #         vri = np.dot(vri,np.dot(np.linalg.inv(E-hi),vr2))
707 #         vli = np.dot(vl2,np.dot(np.linalg.inv(E-hi),vli))
708 #
709 #         hi = hn
710 #         #print(i)
711 #         i = i+1
712 #         vrj = h[i*n:(i+1)*n,(i+1)*n:(i+2)*n]
713 #         vlj = h[(i+1)*n:(i+2)*n,i*n:(i+1)*n]
714 #         h2 = hn
715 #         hn = h[(i+1)*n:(i+2)*n,(i+1)*n:(i+2)*n]
716 #         for j in range(i,L-1):
717 #             print(j)
718 #             vr2 = h[j*n:(j+1)*n,(j+1)*n:(j+2)*n]
719 #             vl2 = h[(j+1)*n:(j+2)*n,i*n:(j+1)*n]
720 #             #print(vr == vr2)
721 #             h2 = h2 + np.dot(vrj,np.dot(np.linalg.inv(E-hn),vlj))

```

```

722 #         hn = h[(1+j)*n:(2+j)*n, (1+j)*n:(2+i)*n] + np.dot(vl2, np.dot(
        np.linalg.inv(E-hn), vr2))
723 #         vrj = np.dot(vrj, np.dot(np.linalg.inv(E-hi), vr2))
724 #         vlj = np.dot(vl2, np.dot(np.linalg.inv(E-hi), vlj))
725 #
726 #         hi = hn
727 #         di = len(h1)
728 #         hf = np.block([[h1, vri, np.zeros((di, di), complex)], [vli, h2, vrj], [
        np.zeros((di, di), complex), vlj, hn]])
729 #         #print(N)
730 #         #print(len(hf))
731 #         return hf
732 #
733 #
734 #
735 #
=====

736
737 def plotZ(N, h):
738     fig = plt.subplots()
739     a = len(h)
740     n = 2*N
741     L = int(a/n)
742     pos = []
743     c = 1
744     for l in range(L):
745         #print(l)
746         b = 1
747         posy = 1
748         for i in range(c, c+n):
749             if i%2 == 0:
750                 x = b+2*1
751                 y = posy
752                 h[i-1][i-1]
753                 if h[i-1][i-1].real != 0:
754                     print(i)
755                     plt.scatter(x, y, color = "red", s = 50, alpha=0.6)
756                 else:
757                     plt.scatter(x, y, color = "gray", s = 50, alpha=0.6)
758                 pos.append([x, y])
759             else:
760                 x = b+2*1

```



```

761         y = posy
762         h[i-1][i-1]
763         if h[i-1][i-1].real != 0:
764             plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
765             #print(i)
766         else:
767             plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
768             pos.append([x,y])
769             if b == 1:
770                 b = 0
771             else:
772                 b = 1
773             posy = posy + 1
774     c = c + n
775     for k in range(1,a+1):
776         for j in range(1,a+1):
777             if h[k-1][j-1] != 0:
778                 #print(k,j)
779                 if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
780                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
781 [j-1][1]],'k--',color="red",alpha = 0.4)
782                 else:
783                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
784 [j-1][1]],color = "black",alpha = 0.4)
785
786 plt.show()
787
788 def plotA(N,h,b=0):
789     fig = plt.subplots()
790     a = len(h)
791     n = 2*N
792     L = int(a/n)
793     pos = []
794     c = 1
795     X = []
796     Y = []
797     for l in range(L):
798         #print(l)
799         bi = 1+b
800         bp = 4-b
801         posy = 1
802         for i in range(c,c+n):
803             #print(i)

```

```

802         if i%2 != 0:
803             x = bi+4*l
804             #print(x)
805             y = posy
806             h[i-1][i-1]
807             if h[i-1][i-1].real != 0:
808                 #print(i)
809                 plt.scatter(x,y,color = "red",s = 100,alpha=0.6)
810             else:
811                 plt.scatter(x,y,color = "gray",s = 100,alpha=0.6)
812             pos.append([x,y])
813             X.append(x)
814             Y.append(y)
815             if bi == 1:
816                 bi = 2
817             else:
818                 bi = 1
819         else:
820             x = bp+4*l
821             y = posy
822             h[i-1][i-1]
823             if h[i-1][i-1].real != 0:
824                 plt.scatter(x,y,color = "red",s = 100,alpha=0.6)
825                 #print(i)
826             else:
827                 plt.scatter(x,y,color = "gray",s =100,alpha=0.6)
828             pos.append([x,y])
829             X.append(x)
830             Y.append(y)
831             if bp == 4:
832                 bp = 3
833             else:
834                 bp = 4
835             posy = posy + 1
836     c = c + n
837
838     for k in range(1,a+1):
839         for j in range(1,a+1):
840             if h[k-1][j-1] != 0:
841                 #print(k,j)
842                 if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
843                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]], 'k--', color="red", alpha = 0.4)

```

```

844         else:
845             plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
846 [j-1][1]],color = "black",alpha = 0.4)
847 plt.scatter(X,Y,color = "gray",s =100,alpha=0.6)
848
849
850 def plotA_mult(N,H):
851     fig = plt.subplots()
852     nn = -1
853     LL = [0]
854     print()
855     for h in H:
856         #print(LL)
857         shift = sum(LL)
858         #print(shift)
859         nn = nn + 1
860         a = len(h)
861         n = 2*(N[nn])
862
863         L = int(a/n)
864         pos = []
865         c = 1
866         for l in range(L):
867             #print(l)
868             bi = 1
869             bp = 4
870             posy = 1
871             for i in range(c,c+n):
872                 #print(i)
873                 if i%2 != 0:
874                     x = bi+4*l+shift
875                     y = posy
876                     h[i-1][i-1]
877                     if h[i-1][i-1].real != 0:
878                         #print(i)
879                         plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
880                     else:
881                         plt.scatter(x,y,color = "gray",s = 50,alpha
882 =0.6)
883
884                     pos.append([x,y])
885                     if bi == 1:
886                         bi = 2

```

```

885         else:
886             bi = 1
887     else:
888         x = bp+4*L+shift
889         y = posy
890         h[i-1][i-1]
891         if h[i-1][i-1].real != 0:
892             plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
893             #print(i)
894         else:
895             plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
896         pos.append([x,y])
897         if bp == 4:
898             bp = 3
899         else:
900             bp = 4
901         posy = posy + 1
902     c = c + n
903     for k in range(1,a+1):
904         for j in range(1,a+1):
905             if h[k-1][j-1] != 0:
906                 #print(k,j)
907                 if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
908                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j-1][1]],'k--',color="red",alpha = 0.4)
909                 else:
910                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j-1][1]],color = "black",alpha = 0.4)
911
912     LL.append(4*L)
913
914
915
916 def plotA_Exm(LL,N,h,b=0):
917     fig = plt.subplots()
918     pos = []
919     #nn = -1
920     shift = 0
921     L = 0
922     c = 1
923     for l0 in np.arange(len(LL)):
924
925         shift = shift + 4*L

```

```

926     #print(shift)
927     #nn = nn + 1
928     a = len(h)
929     n = 2*N[10]#2*(N[nn])
930
931     L = LL[10]#int(a/n)
932     #print(10)
933     #print(L,N[10])
934
935
936     for l in range(L):
937         #print(l)
938         bi = 1+b
939         bp = 4-b
940         posy = 1
941         for i in range(c,c+n):
942             #print(i)
943             if i%2 != 0:
944                 x = bi+4*l+shift
945                 y = posy
946                 h[i-1][i-1]
947                 if h[i-1][i-1].real != 0:
948                     #print(i)
949                     plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
950                     #print(h[i-1][i-1].real,i-1)
951                 else:
952                     plt.scatter(x,y,color = "gray",s = 50,alpha
=0.6)
953
954                 pos.append([x,y])
955                 if bi == 1:
956                     bi = 2
957                 else:
958                     bi = 1
959             else:
960                 x = bp+4*l+shift
961                 y = posy
962                 h[i-1][i-1]
963                 if h[i-1][i-1].real != 0:
964                     plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
965                     #print(i)
966                 else:
967                     plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
968                 pos.append([x,y])

```

```

968         if bp == 4:
969             bp = 3
970         else:
971             bp = 4
972             posy = posy + 1
973         c = c + n
974     Din = len(h)
975     dc = 0
976     dc = np.linspace(0, 1, 210)
977     dci = 0
978     for k in range(1, Din+1):
979         for j in range(1, Din+1):
980             if h[k-1][j-1] != 0:
981                 #print(k, j)
982                 if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
983                     plt.plot([pos[k-1][0], pos[j-1][0]], [pos[k-1][1], pos
984 #
=====
985 #                 if k > j:
986 #                     #print(k)
987 #                     co = cm.rainbow(dc[dci])
988 #                     plt.scatter(pos[k-1][0], pos[k-1][1], color=co)
989 #                     plt.scatter(pos[j-1][0], pos[j-1][1], color=co)
990 #                     dci = dci + 1
991 #
=====
992         else:
993             plt.plot([pos[k-1][0], pos[j-1][0]], [pos[k-1][1], pos
994 [j-1][1]], 'k--', color = "black", alpha = 0.4)
995
996
997
998 def plotq(N, h):
999     a = len(h)
1000     n = N
1001     L = int(a/n)
1002     pos = []
1003     c = 1
1004     for l in range(L):

```

```

1005     #print(l)
1006     bi = 1
1007     posy = 1
1008     for i in range(c,c+n):
1009         x = bi+1
1010         y = posy
1011         #print(x,y)
1012         h[i-1][i-1]
1013         if h[i-1][i-1].real != 0:
1014             #print(i)
1015             plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1016         else:
1017             plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1018         posy = posy+1
1019         pos.append([x,y])
1020     #c = c + n
1021     for k in range(1,a+1):
1022         for j in range(1,a+1):
1023             if h[k-1][j-1] != 0:
1024                 #print(k,j)
1025                 if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
1026                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
1027 [j-1][1]],'k--',color="black",alpha = 0.4)
1028                 else:
1029                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
1030 [j-1][1]],color = "black",alpha = 0.4)
1031
1032 def plotqS(N,h):
1033     a = len(h)
1034     n = 2*N
1035     L = int(a/n)
1036     pos = []
1037     c = 1
1038
1039     x0 = 0
1040     x1 = 1
1041     for l in range(L):
1042         #print(l)
1043
1044         #print(x0,x1)
1045         posy = 0
1046         for i in range(c,c+n):

```

```

1046         #print(i)
1047         y = posy
1048         if i%2 != 0:
1049             x = x0
1050         else:
1051             x = x1
1052             posy = posy+1
1053         if h[i-1][i-1].real != 0:
1054             #print(i)
1055             plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1056         else:
1057             plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1058         pos.append([x,y])
1059         #print(x,y,posy,i)
1060     x0 = x0+2
1061     x1 = x1+2
1062
1063     c = c + n
1064     for k in range(1,a+1):
1065         for j in range(1,a+1):
1066             if h[k-1][j-1] != 0:
1067                 #print(k,j)
1068                 if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
1069                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
1070 [j-1][1]],'k--',color="black",alpha = 0.4)
1071                 else:
1072                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
1073 [j-1][1]],color = "black",alpha = 0.4)
1074
1075 def plotqadc(N,h1,r):
1076     rm = 0
1077     ra = 0
1078     h = h1
1079     for ind in r:
1080         #print(ind)
1081         h = np.delete(h,ind-1*rm,0)
1082         h = np.delete(h,ind-1*rm,1)
1083         rm = rm +1
1084     a = len(h)
1085     n = N
1086     L = int(a/n)
1087     pos = []

```



```

1087     c = 1
1088     at = 0
1089     for l in range(L):
1090         #print(l)
1091         bi = 1
1092         posy = 1
1093         for i in range(c,c+n):
1094             x = bi+1
1095             y = posy
1096             #print(x,y)
1097             h[i-1][i-1]
1098             if h[i-1][i-1].real != 0:
1099                 #print(i)
1100                 plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1101             else:
1102                 plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1103             posy = posy+1
1104             pos.append([x,y])
1105             if len(r)>=1:
1106                 #print(at)
1107                 if r[0]-1== at:
1108                     #print(i)
1109                     xa = x#-1*rm
1110                     plt.scatter(xa+0.5,y+0.5,color = "red",s = 50,alpha
=0.6)
1111                     pos.append([xa+0.5,y+0.5])
1112                     ra = ra + 1
1113                     r = np.delete(r,0,0)
1114                 at = at + 1
1115                 #print(at)
1116             #c = c + n
1117     a2 = len(h1)
1118     for k in range(1,a2+1):
1119         for j in range(1,a2+1):
1120             if h1[k-1][j-1] != 0:
1121                 #print(k,j)
1122                 if h1[k-1][j-1] != 1:
1123                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],color="red",alpha = 0.4)
1124                 else:
1125                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],color = "black",alpha = 0.4)
1126

```

```
1127
1128 def plotAdc(N,h1,r):
1129     rm = 0
1130     ra = 0
1131     ei = []
1132     h = h1
1133     for ind in r:
1134         #print(ind)
1135         ei.append(h[ind-1*rm][ind-1*rm])
1136         h = np.delete(h,ind-1*rm,0)
1137         h = np.delete(h,ind-1*rm,1)
1138         rm = rm +1
1139     a = len(h)
1140     n = 2*N
1141     L = int(a/n)
1142     pos = []
1143     c = 1
1144     for l in range(L):
1145         #print(l)
1146         bi = 1
1147         bp = 4
1148         posy = 1
1149         for i in range(c,c+n):
1150             #print(i)
1151             if i%2 != 0:
1152                 x = bi+4*1
1153                 y = posy
1154                 h[i-1][i-1]
1155                 if h[i-1][i-1].real != 0:
1156                     #print(i)
1157                     plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1158                 else:
1159                     plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1160                 pos.append([x,y])
1161                 if bi == 1:
1162                     bi = 2
1163                 else:
1164                     bi = 1
1165             else:
1166                 x = bp+4*1
1167                 y = posy
1168                 h[i-1][i-1]
1169                 if h[i-1][i-1].real != 0:
```

```

1170         plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1171         #print(i)
1172     else:
1173         plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
1174     pos.append([x,y])
1175     if bp == 4:
1176         bp = 3
1177     else:
1178         bp = 4
1179     posy = posy + 1
1180     if len(r)>=1:
1181         # print(i)
1182         if i == (r[0] - ra):
1183             print(i)
1184             xa = x-1*rm
1185             if ei[0] == 0.0:
1186                 plt.scatter(xa+3,y+1,color = "black",s = 50,
alpha=0.6)
1187             else:
1188                 plt.scatter(xa+3,y+1,color = "red",s = 50,alpha
=0.6)
1189             pos.append([xa+3,y+1])
1190             ra = ra + 1
1191             r = np.delete(r,0,0)
1192             ei = np.delete(ei,0,0)
1193     c = c + n
1194     a2 = len(h1)
1195     for k in range(1,a2+1):
1196         for j in range(1,a2+1):
1197             if h1[k-1][j-1] != 0:
1198                 #print(k,j)
1199                 if h1[k-1][j-1] != 0 and h1[k-1][j-1] != 1:
1200                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],'k--',color="r",alpha = 0.4)
1201                 else:
1202                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],color = "black",alpha = 0.4)
1203
1204
1205 def plotRm(N,h,r):
1206     rm = 0
1207     ra = 0
1208     ei = []

```

```

1209     #h = h1
1210     for ind in r:
1211         #print(ind)
1212         #ei.append(h[ind-1*rm][ind-1*rm])
1213         #h = np.delete(h,ind-1*rm,0)
1214         #h = np.delete(h,ind-1*rm,1)
1215         rm = rm +1
1216     a = len(h)
1217     n = 2*N
1218     L = int(a/n)
1219     pos = []
1220     c = 1
1221     for l in range(L):
1222         #print(l)
1223         bi = 1
1224         bp = 4
1225         posy = 1
1226         for i in range(c,c+n):
1227             #print(i)
1228             if i%2 != 0:
1229                 x = bi+4*1
1230                 y = posy
1231                 h[i-1][i-1]
1232                 if h[i-1][i-1].real != 0:
1233                     #print(i)
1234                     plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1235                 else:
1236                     plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1237                 pos.append([x,y])
1238                 if bi == 1:
1239                     bi = 2
1240                 else:
1241                     bi = 1
1242             else:
1243                 x = bp+4*1
1244                 y = posy
1245                 h[i-1][i-1]
1246                 if h[i-1][i-1].real != 0:
1247                     plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1248                     #print(i)
1249                 else:
1250                     plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
1251                 pos.append([x,y])

```

```

1252         if bp == 4:
1253             bp = 3
1254         else:
1255             bp = 4
1256         posy = posy + 1
1257         if len(r)>=1:
1258             # print(i)
1259             if i == (r[0] - ra):
1260                 print(i)
1261                 xa = x-1*rm
1262                 #if ei[0] == 0.0:
1263                     # plt.scatter(xa+3,y+1,color = "black",s = 50,
alpha=0.6)
1264                 #else:
1265                     # plt.scatter(xa+3,y+1,color = "red",s = 50,
alpha=0.6)
1266                 #pos.append([xa+3,y+1])
1267                 ra = ra + 1
1268                 r = np.delete(r,0,0)
1269                 #ei = np.delete(ei,0,0)
1270         c = c + n
1271         a2 = len(h)
1272         for k in range(1,a2+1):
1273             for j in range(1,a2+1):
1274                 if h[k-1][j-1] != 0:
1275                     #print(k,j)
1276                     if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
1277                         plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],'k--',color="r",alpha = 0.4)
1278                     else:
1279                         plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],color = "black",alpha = 0.4)
1280
1281
1282
1283 def plotAdc2(N,h1,r,ax):
1284     rm = 0
1285     ra = 0
1286     h = h1
1287     for ind in r:
1288         #print(ind)
1289         h = np.delete(h,ind-1*rm,0)
1290         h = np.delete(h,ind-1*rm,1)

```

```
1291     rm = rm +1
1292     a = len(h)
1293     n = 2*N
1294     L = int(a/n)
1295     pos = []
1296     c = 1
1297     for l in range(L):
1298         #print(l)
1299         bi = 1
1300         bp = 4
1301         posy = 1
1302         for i in range(c,c+n):
1303             #print(i)
1304             if i%2 != 0:
1305                 x = bi+4*l
1306                 y = posy
1307                 h[i-1][i-1]
1308                 if h[i-1][i-1].real != 0:
1309                     #print(i)
1310                     ax.scatter(x,y,color = "red",s = 50,alpha=0.6)
1311                 else:
1312                     ax.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1313                 pos.append([x,y])
1314                 if bi == 1:
1315                     bi = 2
1316                 else:
1317                     bi = 1
1318             else:
1319                 x = bp+4*l
1320                 y = posy
1321                 h[i-1][i-1]
1322                 if h[i-1][i-1].real != 0:
1323                     ax.scatter(x,y,color = "red",s = 50,alpha=0.6)
1324                     #print(i)
1325                 else:
1326                     ax.scatter(x,y,color = "gray",s =50,alpha=0.6)
1327                 pos.append([x,y])
1328                 if bp == 4:
1329                     bp = 3
1330                 else:
1331                     bp = 4
1332                 posy = posy + 1
1333         if len(r)>=1:
```

```

1334         # print(i)
1335         if i == (r[0] - ra):
1336             print(i)
1337             xa = x-1*rm
1338             ax.scatter(xa+1,y+1,color = "red",s = 50,alpha=0.6)
1339             pos.append([xa+1,y+1])
1340             ra = ra + 1
1341             r = np.delete(r,0,0)
1342     c = c + n
1343     a2 = len(h1)
1344     for k in range(1,a2+1):
1345         for j in range(1,a2+1):
1346             if h1[k-1][j-1] != 0:
1347                 #print(k,j)
1348                 if h1[k-1][j-1] != 0 and h1[k-1][j-1] != 1:
1349                     ax.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[
j-1][1]],'k--',color="r",alpha = 0.4)
1350                 else:
1351                     ax.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[
j-1][1]],color = "black",alpha = 0.4)
1352
1353 #
=====

1354 # def plotAdc(N,hf,r):
1355 #     x = 0
1356 #     y = 0
1357 #     pos = []
1358 #     Posx = []
1359 #     Posy = []
1360 #     l = 1
1361 #     hs = len(r)+1
1362 #     for hi in range(hs):
1363 #         print(hi)
1364 #         if hi == 0:
1365 #             h = hf[:,r, :r]
1366 #             print(len(h))
1367 #         if hi == 1:
1368 #             plt.scatter(x,y+1,color = "red",s = 50,alpha=0.6)
1369 #             pos.append([x,y+1])
1370 #             Posx.append(x)
1371 #             Posy.append(y+1)
1372 #             h = hf[r+1:, r+1:]

```

```

1373 #         print(len(h))
1374 #     a = len(h)
1375 #     n = 2*N
1376 #     L = int(a/n)
1377 #     c = 1
1378 #     if hi == 0:
1379 #         ran = range(L)
1380 #     if hi == 1:
1381 #         ran = range(l+1,L+1+1)
1382 #     for l in ran:
1383 #         #print(l)
1384 #         bi = 1
1385 #         bp = 4
1386 #         posy = 1
1387 #
1388 #         for i in range(c,c+n):
1389 #             #print(i)
1390 #             if i%2 != 0:
1391 #                 x = bi+4*l
1392 #                 y = posy
1393 #                 #h[i-1][i-1]
1394 #                 if h[i-1][i-1].real != 0:
1395 #                     #print(i)
1396 #                     plt.scatter(x,y,color = "red",s = 50,alpha
=0.6)
1397 #                 else:
1398 #                     plt.scatter(x,y,color = "gray",s = 50,alpha
=0.6)
1399 #                 pos.append([x,y])
1400 #                 Posx.append(x)
1401 #                 Posy.append(y)
1402 #                 if bi == 1:
1403 #                     bi = 2
1404 #                 else:
1405 #                     bi = 1
1406 #             else:
1407 #                 x = bp+4*l
1408 #                 y = posy
1409 #                 #h[i-1][i-1]
1410 #                 if h[i-1][i-1].real != 0:
1411 #                     plt.scatter(x,y,color = "red",s = 50,alpha
=0.6)
1412 #                 #print(i)

```



```

1413 #                 else:
1414 #                     plt.scatter(x,y,color = "gray",s =50,alpha
=0.6)
1415 #                 pos.append([x,y])
1416 #                 Posx.append(x)
1417 #                 Posy.append(y)
1418 #                 if bp == 4:
1419 #                     bp = 3
1420 #                 else:
1421 #                     bp = 4
1422 #                 posy = posy + 1
1423 #                 c = c + n
1424 #                 #plt.scatter(Posx,Posy)
1425 #                 din = len(hf)
1426 #                 for k in range(din):
1427 #                     for j in range(din):
1428 #                         if hf[k][j] != 0:
1429 #                             #print(k,j)
1430 #                             # if hf[k-1][j-1] != 0 and hf[k-1][j-1] != 1:
1431 #                             #     plt.plot([pos[k-1][0],pos[j-1][0]], [pos[k-1][1],
pos[j-1][1]], 'k--',color="r",alpha = 0.4)
1432 #                             plt.plot([pos[k][0],pos[j][0]], [pos[k][1],pos[j][1]],
color = "black",alpha = 0.4)
1433 #                             #     plt.plot([Posx[k],Posy[k]], [Posx[j],Posy[j]],color =
"black",alpha = 0.4)
1434 #
1435 #
1436 #
=====
1437
1438
1439 #
=====

1440 # def densityAdc(N,hf,r,D):
1441 #     x = 0
1442 #     y = 0
1443 #     pos = []
1444 #     Posx = []
1445 #     Posy = []
1446 #     l = 1
1447 #     for hi in range(2):

```

```

1448 #         print(hi)
1449 #         if hi == 0:
1450 #             h = hf[:r, :r]
1451 #             print(len(h))
1452 #         if hi == 1:
1453 #             #plt.scatter(x,y+1,color = "red",s = 50,alpha=0.6)
1454 #             pos.append([x,y+1])
1455 #             Posx.append(x)
1456 #             Posy.append(y+1)
1457 #             h = hf[r+1:, r+1:]
1458 #             print(len(h))
1459 #         a = len(h)
1460 #         n = 2*N
1461 #         L = int(a/n)
1462 #         c = 1
1463 #         if hi == 0:
1464 #             ran = range(L)
1465 #         if hi == 1:
1466 #             ran = range(l+1,L+1+1)
1467 #         for l in ran:
1468 #             #print(l)
1469 #             bi = 1
1470 #             bp = 4
1471 #             posy = 1
1472 #
1473 #             for i in range(c,c+n):
1474 #                 #print(i)
1475 #                 if i%2 != 0:
1476 #                     x = bi+4*l
1477 #                     y = posy
1478 #                     #h[i-1][i-1]
1479 #                     #if h[i-1][i-1].real != 0:
1480 #                         #print(i)
1481 #                         # plt.scatter(x,y,color = "red",s = 50,alpha
1482 # =0.6)
1483 #                         #else:
1484 #                         # plt.scatter(x,y,color = "gray",s = 50,alpha
1485 # =0.6)
1486 #                         pos.append([x,y])
1487 #                         Posx.append(x)
1488 #                         Posy.append(y)
1489 #                         if bi == 1:
1490 #                             bi = 2

```

```

1489 #             else:
1490 #                 bi = 1
1491 #             else:
1492 #                 x = bp+4*1
1493 #                 y = posy
1494 #                 #h[i-1][i-1]
1495 #                 #if h[i-1][i-1].real != 0:
1496 #                 #     plt.scatter(x,y,color = "red",s = 50,alpha
=0.6)
1497 #                 #print(i)
1498 #             #else:
1499 #                 #plt.scatter(x,y,color = "gray",s =50,alpha
=0.6)
1500 #                 pos.append([x,y])
1501 #                 Posx.append(x)
1502 #                 Posy.append(y)
1503 #                 if bp == 4:
1504 #                     bp = 3
1505 #                 else:
1506 #                     bp = 4
1507 #                 posy = posy + 1
1508 #                 c = c + n
1509 #                 #plt.scatter(Posx,Posy)
1510 #                 fig, axs = plt.subplots()
1511 #                 plt.scatter(Posx,Posy,s =300,c=D, cmap=plt.cm.jet,norm =None,
alpha=0.4)#plt.cm.jet
1512 #                 plt.colorbar()
1513 #                 plt.scatter(Posx,Posy,color = "black",s = 10,alpha = 0.8)
1514 #                 din = len(hf)
1515 #                 for k in range(din):
1516 #                     for j in range(din):
1517 #                         if hf[k][j] != 0:
1518 #                             #print(k,j)
1519 #                             # if hf[k-1][j-1] != 0 and hf[k-1][j-1] != 1:
1520 #                             #     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],
pos[j-1][1]],'k--',color="r",alpha = 0.4)
1521 #                             plt.plot([pos[k][0],pos[j][0]],[pos[k][1],pos[j][1]],
color = "black",alpha = 0.4)
1522 #                             #     plt.plot([Posx[k],Posy[k]],[Posx[j],Posy[j]],color =
"black",alpha = 0.4)
1523 #
1524 #

```

```

1525 #
=====
1526
1527 def densityqAdc(N,h1,r,D):
1528     rm = 0
1529     ra = 0
1530     h = h1
1531     for ind in r:
1532         #print(ind)
1533         h = np.delete(h,ind-1*rm,0)
1534         h = np.delete(h,ind-1*rm,1)
1535         rm = rm +1
1536     a = len(h)
1537     n = N
1538     L = int(a/n)
1539     pos = []
1540     Posx = []
1541     Posy = []
1542     c = 1
1543     at = 0
1544     for l in range(L):
1545         #print(l)
1546         bi = 1
1547         posy = 1
1548         for i in range(c,c+n):
1549             x = bi+1
1550             y = posy
1551             #print(x,y)
1552             #h[i-1][i-1]
1553             #if h[i-1][i-1].real != 0:
1554                 #print(i)
1555                 #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1556             #else:
1557                 #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1558             posy = posy+1
1559             pos.append([x,y])
1560             Posx.append(x)
1561             Posy.append(y)
1562             if len(r)>=1:
1563                 #print(at)
1564                 if r[0]-1== at:
1565                     #print(i)

```

```

1566         xa = x#-1*rm
1567         #plt.scatter(xa+0.5,y+0.5,color = "red",s = 50,
alpha=0.6)
1568         pos.append([xa+0.5,y+0.5])
1569         Posx.append(xa+0.5)
1570         Posy.append(y+0.5)
1571         ra = ra + 1
1572         r = np.delete(r,0,0)
1573         at = at + 1
1574         #print(at)
1575         #c = c + n
1576         a2 = len(h1)
1577         fig, axs = plt.subplots()
1578         plt.scatter(Posx,Posy,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha
=0.4)#plt.cm.jet
1579         plt.colorbar()
1580         plt.scatter(Posx,Posy,color = "black",s = 10,alpha = 0.8)
1581         for k in range(1,a2+1):
1582             for j in range(1,a2+1):
1583                 if h1[k-1][j-1] != 0:
1584                     #print(k,j)
1585                     if h1[k-1][j-1] != 1:
1586                         plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],color="red",alpha = 0.4)
1587                     else:
1588                         plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],color = "black",alpha = 0.4)
1589
1590
1591 def densityqAdc2(N,h1,r,D,ax):
1592     rm = 0
1593     ra = 0
1594     h = h1
1595     for ind in r:
1596         #print(ind)
1597         h = np.delete(h,ind-1*rm,0)
1598         h = np.delete(h,ind-1*rm,1)
1599         rm = rm +1
1600     a = len(h)
1601     n = N
1602     L = int(a/n)
1603     pos = []
1604     Posx = []

```

```

1605     Posy = []
1606     c = 1
1607     at = 0
1608     for l in range(L):
1609         #print(l)
1610         bi = 1
1611         posy = 1
1612         for i in range(c,c+n):
1613             x = bi+1
1614             y = posy
1615             #print(x,y)
1616             #h[i-1][i-1]
1617             #if h[i-1][i-1].real != 0:
1618                 #print(i)
1619                 #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1620             #else:
1621                 #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1622             posy = posy+1
1623             pos.append([x,y])
1624             Posx.append(x)
1625             Posy.append(y)
1626             if len(r)>=1:
1627                 #print(at)
1628                 if r[0]-1== at:
1629                     #print(i)
1630                     xa = x#-1*rm
1631                     #plt.scatter(xa+0.5,y+0.5,color = "red",s = 50,
alpha=0.6)
1632                     pos.append([xa+0.5,y+0.5])
1633                     Posx.append(xa+0.5)
1634                     Posy.append(y+0.5)
1635                     ra = ra + 1
1636                     r = np.delete(r,0,0)
1637                 at = at + 1
1638                 #print(at)
1639                 #c = c + n
1640             a2 = len(h1)
1641             #fig, axs = plt.subplots()
1642             ax.scatter(Posx,Posy,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha
=0.4)#plt.cm.jet
1643             #ax.colorbar()
1644             #plt.colorbar()
1645             ax.scatter(Posx,Posy,color = "black",s = 10,alpha = 0.8)

```

```

1646     for k in range(1,a2+1):
1647         for j in range(1,a2+1):
1648             if h1[k-1][j-1] != 0:
1649                 #print(k,j)
1650                 if h1[k-1][j-1] != 1:
1651                     ax.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[
1652 j-1][1]],color="red",alpha = 0.4)
1653                 else:
1654                     ax.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[
1655 j-1][1]],color = "black",alpha = 0.4)
1656
1657
1658
1659
1660 def densityZAdc(N,h1,r,D):
1661
1662     rm = 0
1663     ra = 0
1664     h = h1
1665     for ind in r:
1666         #print(ind)
1667         h = np.delete(h,ind-1*rm,0)
1668         h = np.delete(h,ind-1*rm,1)
1669         rm = rm +1
1670
1671     a = len(h)
1672     n2 = 2*N
1673     L = int(a/n2)
1674     pos = []
1675
1676     c = 1
1677     X = []
1678     Y = []
1679     Z = []
1680
1681     for l in range(L):
1682         #print(l)
1683         b = 1
1684         posy = 1
1685         for i in range(c,c+n2):
1686             if i%2 == 0:

```

```

1687         #print(i)
1688         x = b+2*1
1689         y = posy
1690         pos.append([x,y])
1691         X.append(x)
1692         Y.append(y)
1693         #Z.append(sum(np.abs(psi[i-1].real)))
1694         #Z.append(h1[i-1][i-1].real)
1695         #if density == "psi":
1696
1697     else:
1698         #print(i)
1699         x = b+2*1
1700         y = posy
1701         pos.append([x,y])
1702         X.append(x)
1703         Y.append(y)
1704         #Z.append(sum(np.abs(psi[i-1].real)))
1705         #Z.append(h1[i-1][i-1].real)
1706         if b == 1:
1707             b = 0
1708         else:
1709             b = 1
1710         posy = posy + 1
1711
1712     if len(r)>=1:
1713         # print(i)
1714         if i == (r[0] - ra):
1715             print(i)
1716             xa = x-1*rm
1717             #plt.scatter(xa+2,y+1,color = "red",s = 50,alpha
=0.6)
1718             pos.append([xa+1,y+1])
1719             X.append(xa+1)
1720             Y.append(y+1)
1721             ra = ra + 1
1722             r = np.delete(r,0,0)
1723
1724     c = c + n2
1725     #normalize =mt.colors.Normalize(vmin=0, vmax=1, clip=False)
1726     #nor = sum(Z)
1727     #Z = Z/nor
1728     a2 = len(h1)

```



```

1729     fig, axs = plt.subplots()
1730     plt.scatter(X,Y,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha=0.4) #
plt.cm.jet
1731     plt.colorbar()
1732     plt.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
1733     for k in range(1,a2+1):
1734         for j in range(1,a2+1):
1735             if h1[k-1][j-1] != 0:
1736                 #print(k-1,j-1)
1737                 plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
-1][1]],color = "black",alpha = 0.4)
1738     plt.show()
1739
1740
1741
1742
1743 def densityAdc(N,h1,r,D):
1744     rm = 0
1745     ra = 0
1746     h = h1
1747     for ind in r:
1748         #print(ind)
1749         h = np.delete(h,ind-1*rm,0)
1750         h = np.delete(h,ind-1*rm,1)
1751         rm = rm +1
1752     a = len(h)
1753     n = 2*N
1754     L = int(a/n)
1755     pos = []
1756     Posx = []
1757     Posy = []
1758     c = 1
1759     for l in range(L):
1760         #print(l)
1761         bi = 1
1762         bp = 4
1763         posy = 1
1764         for i in range(c,c+n):
1765             #print(i)
1766             if i%2 != 0:
1767                 x = bi+4*1
1768                 y = posy
1769                 h[i-1][i-1]

```

```

1770         #if h[i-1][i-1].real != 0:
1771             #print(i)
1772             # plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1773         #else:
1774             # plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1775         pos.append([x,y])
1776         Posx.append(x)
1777         Posy.append(y)
1778         if bi == 1:
1779             bi = 2
1780         else:
1781             bi = 1
1782     else:
1783         x = bp+4*1
1784         y = posy
1785         h[i-1][i-1]
1786         #if h[i-1][i-1].real != 0:
1787             #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1788             #print(i)
1789         #else:
1790             # plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
1791         pos.append([x,y])
1792         Posx.append(x)
1793         Posy.append(y)
1794         if bp == 4:
1795             bp = 3
1796         else:
1797             bp = 4
1798         posy = posy + 1
1799     if len(r)>=1:
1800         # print(i)
1801         if i == (r[0] - ra):
1802             print(i)
1803             xa = x-1*rm
1804             #plt.scatter(xa+2,y+1,color = "red",s = 50,alpha
1805             #0.6)
1806             pos.append([xa+1,y+1])
1807             Posx.append(xa+1)
1808             Posy.append(y+1)
1809             ra = ra + 1
1810             r = np.delete(r,0,0)
1811         c = c + n
1812         a2 = len(h1)

```

```

1812     fig, axs = plt.subplots()
1813     plt.scatter(Posx,Posy,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha
1814     =0.4) #plt.cm.jet
1815     plt.colorbar()
1816     plt.scatter(Posx,Posy,color = "black",s = 10,alpha = 0.8)
1817     for k in range(1,a2+1):
1818         for j in range(1,a2+1):
1819             if h1[k-1][j-1] != 0:
1820                 #print(k,j)
1821                 #if h1[k-1][j-1] != 0 and h1[k-1][j-1] != 1:
1822                     #plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
1823                     -1][1]],'k--',color="r",alpha = 0.4)
1824                 #else:
1825                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
1826                     -1][1]],color = "black",alpha = 0.4)
1827
1828 def densityAdc2(N,h1,r,D,ax):
1829     rm = 0
1830     ra = 0
1831     h = h1
1832     for ind in r:
1833         #print(ind)
1834         h = np.delete(h,ind-1*rm,0)
1835         h = np.delete(h,ind-1*rm,1)
1836         rm = rm +1
1837
1838     a = len(h)
1839     n = 2*N
1840     L = int(a/n)
1841     pos = []
1842     Posx = []
1843     Posy = []
1844     c = 1
1845     for l in range(L):
1846         #print(l)
1847         bi = 1
1848         bp = 4
1849         posy = 1
1850         for i in range(c,c+n):
1851             #print(i)
1852             if i%2 != 0:
1853                 x = bi+4*1
1854                 y = posy
1855                 h[i-1][i-1]

```

```

1852         #if h[i-1][i-1].real != 0:
1853             #print(i)
1854             # plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1855         #else:
1856             # plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1857         pos.append([x,y])
1858         Posx.append(x)
1859         Posy.append(y)
1860         if bi == 1:
1861             bi = 2
1862         else:
1863             bi = 1
1864     else:
1865         x = bp+4*1
1866         y = posy
1867         h[i-1][i-1]
1868         #if h[i-1][i-1].real != 0:
1869             #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1870             #print(i)
1871         #else:
1872             # plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
1873         pos.append([x,y])
1874         Posx.append(x)
1875         Posy.append(y)
1876         if bp == 4:
1877             bp = 3
1878         else:
1879             bp = 4
1880         posy = posy + 1
1881     if len(r)>=1:
1882         # print(i)
1883         if i == (r[0] - ra):
1884             print(i)
1885             xa = x-1*rm
1886             #plt.scatter(xa+2,y+1,color = "red",s = 50,alpha
1887             =0.6)
1888             pos.append([xa+1,y+1])
1889             Posx.append(xa+1)
1890             Posy.append(y+1)
1891             ra = ra + 1
1892             r = np.delete(r,0,0)
1893         c = c + n
1894         a2 = len(h1)

```

```

1894     #fig, axs = plt.subplots()
1895     ax.scatter(Posx,Posy,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha
1896     =0.4)#plt.cm.jet
1897     #ax.colorbar()
1898     #matplotlib.colorbar.ColorbarBase(ax=ax, cmap=coolwarm, values=
1899     sorted(v),
1900     #                                     orientation="horizontal")
1901     ax.scatter(Posx,Posy,color = "black",s = 10,alpha = 0.4)
1902     for k in range(1,a2+1):
1903         for j in range(1,a2+1):
1904             if h1[k-1][j-1] != 0:
1905                 #print(k,j)
1906                 #if h1[k-1][j-1] != 0 and h1[k-1][j-1] != 1:
1907                 #plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
1908                 -1][1]],'k--',color="r",alpha = 0.4)
1909                 #else:
1910                 ax.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
1911                 -1][1]],color = "black",alpha = 0.2)
1912
1913 def densityA(N,h,D,b=0):
1914     a = len(h)
1915     n = 2*N
1916     L = int(a/n)
1917     pos = []
1918     c = 1
1919     X = []
1920     Y = []
1921     for l in range(L):
1922         #print(l)
1923         bi = 1+b
1924         bp = 4-b
1925         posy = 1
1926         for i in range(c,c+n):
1927             if i%2 != 0:
1928                 x = bi+4*i
1929                 y = posy
1930                 h[i-1][i-1]
1931                 #if h[i-1][i-1].real != 0:
1932                 #    print(i)
1933                 #    plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1934                 #else:
1935                 #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)

```

```

1933         pos.append([x,y])
1934         X.append(x)
1935         Y.append(y)
1936         if bi == 1:
1937             bi = 2
1938         else:
1939             bi = 1
1940     else:
1941         x = bp+4*1
1942         y = posy
1943         h[i-1][i-1]
1944         #if h[i-1][i-1].real != 0:
1945             #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1946             # print(i)
1947         #else:
1948             #plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
1949         pos.append([x,y])
1950         X.append(x)
1951         Y.append(y)
1952         if bp == 4:
1953             bp = 3
1954         else:
1955             bp = 4
1956         posy = posy + 1
1957     c = c + n
1958     fig, axs = plt.subplots()
1959     plt.scatter(X,Y,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha=0.4) #
1960     plt.cm.jet
1961     plt.colorbar()
1962     plt.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
1963     for k in range(1,a+1):
1964         for j in range(1,a+1):
1965             if h[k-1][j-1] != 0:
1966                 #print(k,j)
1967                 plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j-1][1]],color = "black",alpha = 0.4)
1968
1969 def densityA2(N,h,D,axes):
1970     a = len(h)
1971     n = 2*N
1972     L = int(a/n)
1973     pos = []

```

```
1974     c = 1
1975     X = []
1976     Y = []
1977     for l in range(L):
1978         #print(l)
1979         bi = 1
1980         bp = 4
1981         posy = 1
1982         for i in range(c,c+n):
1983             if i%2 != 0:
1984                 x = bi+4*l
1985                 y = posy
1986                 h[i-1][i-1]
1987                 #if h[i-1][i-1].real != 0:
1988                     # print(i)
1989                     # plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
1990                 #else:
1991                     #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
1992                 pos.append([x,y])
1993                 X.append(x)
1994                 Y.append(y)
1995                 if bi == 1:
1996                     bi = 2
1997                 else:
1998                     bi = 1
1999             else:
2000                 x = bp+4*l
2001                 y = posy
2002                 h[i-1][i-1]
2003                 #if h[i-1][i-1].real != 0:
2004                     #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2005                     # print(i)
2006                 #else:
2007                     #plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
2008                 pos.append([x,y])
2009                 X.append(x)
2010                 Y.append(y)
2011                 if bp == 4:
2012                     bp = 3
2013                 else:
2014                     bp = 4
2015                 posy = posy + 1
2016     c = c + n
```

```

2017     #fig, axs = plt.subplots()
2018     axes.scatter(X,Y,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha=0.4) #
plt.cm.jet
2019     #plt.colorbar()
2020     axes.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
2021     for k in range(1,a+1):
2022         for j in range(1,a+1):
2023             if h[k-1][j-1] != 0:
2024                 #print(k,j)
2025                 axes.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
-1][1]],color = "black",alpha = 0.4)
2026
2027
2028 def current0(N,h,Gn,b=0):
2029     #fig = plt.subplots()
2030     #fig = mlab.figure(size=(1024, 1024),bgcolor = (1,1,1), fgcolor =
(0.5, 0.5, 0.5))
2031     figz = (70,10)
2032
2033     fig1, ax1 = plt.subplots(figsize=figz)
2034
2035
2036     a = len(h)
2037     n = 2*N
2038     L = int(a/n)
2039     pos = []
2040     c = 1
2041     X = []
2042     Y = []
2043     Z = []
2044     if b==1:
2045         b = 3/4
2046     for l in range(L):
2047         #print(l)
2048         bi = 1+b
2049         bp = 4-b
2050         posy = 1
2051         for i in range(c,c+n):
2052             #print(i)
2053             if i%2 != 0:
2054                 x = bi+(9/2)*1
2055                 #print(x)
2056                 y = posy

```



```

2057 #
=====

2058 #         h[i-1][i-1]
2059 #         if h[i-1][i-1].real != 0:
2060 #             #print(i)
2061 #             #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2062 #             mlab.points3d(x,y,0)
2063 #         else:
2064 #             #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
2065 #             mlab.points3d(x,y,0)
2066 #
=====

2067         pos.append([x,y])
2068         X.append(x)
2069         Y.append(y)
2070         Z.append(0)
2071         if bi == 1:
2072             bi = 7/4
2073         else:
2074             bi = 1
2075     else:
2076         x = bp+(9/2)*1
2077         y = posy
2078 #
=====

2079 #         h[i-1][i-1]
2080 #         if h[i-1][i-1].real != 0:
2081 #             #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2082 #             mlab.points3d(x,y,0)
2083 #             #print(i)
2084 #         else:
2085 #             #plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
2086 #             mlab.points3d(x,y,0)
2087 #
=====

2088         pos.append([x,y])
2089         X.append(x)
2090         Y.append(y)
2091         Z.append(0)

```

```

2092         if bp == 4:
2093             bp = 13/4
2094         else:
2095             bp = 4
2096             posy = posy + (3/4)*(np.sqrt(2))
2097     c = c + n
2098     #mlab.points3d(X,Y,Z,scale_factor=0.55,color=(0.5,0.5,0.5))
2099     for k in range(1,a+1):
2100         for j in range(1,a+1):
2101             if h[k-1][j-1] != 0:
2102                 #print(k,j)
2103                 if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
2104                     ax1.plot([pos[k-1][0],pos[j-1][0]], [pos[k-1][1],
2105 pos[j-1][1]], 'k--',color="gray",alpha = 0.4)
2106                     #mlab.plot3d([pos[k-1][0],pos[j-1][0]], [pos[k
2107 -1][1],pos[j-1][1]], [0,0],line_width=3,tube_radius=0.1)
2108                 else:
2109                     ax1.plot([pos[k-1][0],pos[j-1][0]], [pos[k-1][1],
2110 pos[j-1][1]],color = "black",alpha = 0.4)
2111                     #mlab.plot3d([pos[k-1][0],pos[j-1][0]], [pos[k
2112 -1][1],pos[j-1][1]], [0,0],line_width=3,tube_radius=0.1)
2113 #
2114 # =====
2115 #
2116 #     Il = [0]
2117 #     oo = 50
2118 #     for o in range(1,a):
2119 #         I = (h[oo][o])*(Gn[oo][o])
2120 #         Il.append(I.imag)
2121 #     plt.scatter(X,Y,s =300,c=Il, cmap=plt.cm.jet,norm =None,alpha
2122 =0.4)
2123 #     plt.colorbar()
2124 #
2125 # =====
2126
2127
2128
2129     Il = []
2130     Xv = []
2131     Yv = []
2132     Zv = []
2133     U = []
2134     V = []
2135     W = []

```

```

2126
2127
2128     Ic = []
2129     Xc = []
2130     Yc = []
2131     Zc = []
2132     Uc = []
2133     Vc = []
2134     Wc = []
2135
2136
2137     for k in range(1,a+1):
2138         for j in range(1,a+1):
2139             if h[k-1][j-1] != 0:
2140                 I = ((h[k-1][j-1])*(Gn[k-1][j-1])).imag
2141                 x1,x2 = pos[k-1][0],pos[j-1][0]
2142                 y1,y2 = pos[k-1][1],pos[j-1][1]
2143                 iv = np.array([[x2-x1],[y2-y1]])
2144                 #norm = np.linalg.norm(iv)
2145                 #iv = I*(1/norm)*(iv)
2146
2147                 xv = iv[0]
2148                 yv = iv[1]
2149
2150                 nor1 = np.sqrt(x1**2+y1**2)
2151                 nor2 = np.sqrt(x2**2+y2**2)
2152
2153                 cang = (x1*x2 + y1*y2)/(nor1*nor2)
2154
2155                 ang = np.arccos(cang)
2156
2157                 xv = iv[0]
2158                 yv = iv[1]
2159
2160
2161                 Xi = xv#I*np.cos(ang)
2162                 Yi = yv#I*np.sin(ang)
2163
2164
2165                 Xv.append(x1)
2166                 Yv.append(y1)
2167                 Zv.append(0)
2168                 u = Xi#xv[0]

```

```

2169         v = Yi#yv[0]
2170         U.append(u)
2171         V.append(v)
2172         W.append(0)
2173         Il.append(I)
2174         #print(u,v)
2175         #mlab.quiver3d(x1,y1,0,u,v,0,scale_factor=1, line_width
=2, mode='2ddash', scale_mode='vector',colormap='viridis')#colormap
=
2176         #mlab.outline()
2177     Xv,Yv,Zv,U,V,W = np.array(Xv),np.array(Yv),np.array(Zv),np.array(U)
,np.array(V),np.array(W)
2178     bg = 0
2179     #be = 15
2180     pc1 = [15,17,17,15] #valor de componetes vetoriais em cada celula
hexagona
2181     pc2 = [15+1,17+1,17+1,15+1]
2182     for l in range(L):
2183         xc = 2.6 + (9/2)*l
2184         zc = 0
2185
2186         if l == 0 or l == 3:
2187             pc = pc1
2188         else:
2189             pc = pc2
2190         for nn in range(4):
2191
2192             be = bg + pc[nn]
2193
2194             yc = 2.1 + 2*nn*(3/4)*(np.sqrt(2))
2195             Xc.append(xc)
2196             Yc.append(yc)
2197             Zc.append(zc)
2198
2199             ln = len(Xv[bg:be])
2200
2201             #Xv[bg:be] = xc*np.ones((1,ln)) #colocando todos as
componetes no centro do hexagono
2202             #Yv[bg:be] = yc*np.ones((1,ln))
2203
2204             uc = U[bg:be]
2205             vc = V[bg:be]
2206             wc = W[bg:be]

```

```

2207
2208         #ic = Il[bg:be]
2209
2210         us = sum(uc)
2211         vs = sum(vc)
2212
2213         Uc.append(us)
2214         Vc.append(vs)
2215         Wc.append(sum(wc))
2216
2217         ic = np.sqrt(us**2 + vs**2)
2218
2219         Ic.append(ic)
2220
2221         bg = be - 6
2222         bg = bg + 6
2223
2224         #mlab.quiver3d(Xv,Yv,Zv,U,V,W,scale_factor=0.5)
2225         #ax1.scatter(Xc, Yc)
2226         #ax1.hexbin(Xc, Yc,gridsize = 50,linewidths = 5,cmap = "magma")
2227         import matplotlib.colors as mcolors
2228         import matplotlib.colorbar as mcolorbar
2229
2230
2231         #res = 0.000000000000000000000001
2232         #sIc = res*15.11461430450805#max(Ic)
2233
2234         #print(sIc)
2235
2236         # Ic = np.array(Ic)/sIc
2237
2238         #max(Ic)
2239
2240         #nz = mcolors.Normalize()
2241         #nz.autoscale(I)
2242
2243
2244         #plt.scatter(Xc,Yc,s =2500, marker='H',c=Ic, cmap=plt.cm.jet,norm =
None,alpha=0.2)
2245         #plt.scatter(Xv,Yv,s =2500, marker='H',c=I, cmap=plt.cm.jet,norm =
None,alpha=0.2)
2246         #clb=plt.colorbar()
2247         #plt.clim(vmin = 0, vmax = 10)

```

```

2248
2249 #
=====

2250 #     fz = 20
2251 #
2252 #     clb.ax.tick_params(labelsize=fz)
2253 #     clb.ax.tick_params(labelsize=fz)
2254 #     clb.set_label(r'|$Current$|', rotation = 270, fontsize=fz, labelpad
= 40)
2255 #
2256 #     #clb=plt.colorbar(ticks=[0,0,0.0045])
2257 #     clb.ax.tick_params(labelsize=fz)
2258 #
=====

2259 #clb.set_label(r'$|I_{inc}$|$', rotation = 270, fontsize=fz, labelpad
= 40)
2260 #clb.ax.set_yticklabels([r'$|I_{inc}$|>|I_{coh}$|$', r'$|I_{inc}$|=|I_{
coh}$|$', r'$|I_{coh}$|>|I_{inc}$|$', fontsize=fz)
2261
2262
2263
2264 #
=====

2265 #     pts = mlab.quiver3d(Xc,Yc,Zc,Uc,Vc,Wc, scalars=Ic, scale_factor
=0.5)#0.5
2266 #     #mlab.contour3d(Xc,Yc,Zc,Ic)
2267 #
2268 #     pts.glyph.scale_mode = "scale_by_scalar"
2269 #     pts.glyph.color_mode = "color_by_scalar"
2270 #     pts.glyph.glyph_source.glyph_source.center = [0,0,0]
2271 #     mlab.colorbar(object=pts, title="Local current")
2272 #
=====

2273
2274 #ax1.set_title('Arrows scale with plot width, not view')
2275
2276
2277 #width=0.003,scale=750#,width=0.003,scale=600
2278

```

```

2279     nz = mcolors.Normalize()
2280     nz.autoscale(I1)
2281
2282
2283     Q = ax1.quiver(Xv, Yv, np.array(U), np.array(V), color=cm.jet(nz(I1)),
2284                   ,width=0.0015, scale=110)
2285
2286
2287     plt.scatter(Xv, Yv, marker='o', s=0, c=I1, cmap=plt.cm.jet, norm=None,
2288               alpha=0.4)
2289
2290     #Q = ax1.quiver(Xv, Yv, np.array(U), np.array(V), color=cm.jet(nz(I)),
2291                   ,width=0.002, scale=0.0002)
2292
2293     #Q = ax1.quiver(Xc, Yc, np.array(Uc), np.array(Vc), color=cm.jet(nz(Ic)),
2294                   ,width=0.003, scale=400) # units='x', pivot='mid', width=0.3,
2295                   ,length=5, scale=7) #, width=0.022) #, width=0.022) #, scale = 0.015)
2296
2297     #qk = ax1.quiverkey(Q, 0.9, 0.9, 1, r'$1 \frac{m}{s}$', labelpos='E',
2298                   ,coordinates='figure')
2299
2300     ax1.set_xlim(0, 72)
2301     ax1.set_ylim(0, 11)
2302
2303
2304     fz = 48
2305
2306
2307     lb = 0
2308
2309
2310
2311     ax1.spines['left'].set_linewidth(lb)
2312     ax1.spines['right'].set_linewidth(lb)
2313     ax1.spines['top'].set_linewidth(lb)
2314     ax1.spines['bottom'].set_linewidth(lb)
2315
2316
2317     ax1.set_yticklabels([])
2318     ax1.set_xticklabels([])
2319
2320
2321
2322
2323     ax1.xaxis.set_ticks_position('both')
2324     ax1.yaxis.set_ticks_position('both')
2325     ax1.tick_params(axis='both', which='major', labelsize=fz, direction =
2326                   "in", length = 0, width = 30)
2327
2328
2329
2330
2331
2332
2333     #clim = np.linspace(min(I1), max(I1), 5)
2334     clim = np.linspace(0, max(I1), 5)
2335     clb = plt.colorbar(ticks=clim, location="bottom")
2336     clb.ax.tick_params(labelsize=fz)

```

```

2315     clb.ax.tick_params(labelsize=fz)
2316     clb.set_label("Local current",rotation = 0,fontsize=fz,labelpad =
2317     5)
2318     plt.clim(vmin = 0, vmax = max(I1))
2319     plt.suptitle(r'$E = 0.0, U = 0.45, V_{g} = 0.02 , V_{g} = 0$',
2320     fontsize = fz,x=0.5,y=0.95)
2321 #
=====
2322 #
2323 #     ax1.xaxis.set_ticks_position('both')
2324 #     ax1.yaxis.set_ticks_position('both')
2325 #     ax1.tick_params(axis='both', which='major',labelsize=fz,direction
2326 #     = "in",length = 0, width = 30)
2327 #
2328 #     #clim = np.linspace(min(It),max(It),5)
2329 #     clim = np.linspace(0,max(It),5)
2330 #     clb = plt.colorbar(ticks=clim,location="bottom")
2331 #     clb.ax.tick_params(labelsize=fz)
2332 #     clb.ax.tick_params(labelsize=fz)
2333 #     clb.set_label("Local current",rotation = 0,fontsize=fz,labelpad =
2334 #     5)
2335 #     plt.clim(vmin = 0, vmax = max(It))
2336 #
2337 #     plt.suptitle(r'$E = 0.0, U = 0.0, V_{g} = 0.02 , V_{g} = 0$',
2338 #     fontsize = fz,x=0.5,y=0.95)
2339 #
2340 #
=====
2341
2342
2343
2344     #cax,_ = mcolorbar.make_axes(plt.gca())
2345     #cb = mcolorbar.ColorbarBase(cax, cmap=cm.jet, norm=nz)
2346
2347 #
=====

```



```

2348 #     Xq,Yq,Uq,Vq,Iq = np.nan*np.zeros((13,4*L)),np.nan*np.zeros((13,4*
      L)),np.nan*np.zeros((13,4*L)),np.nan*np.zeros((13,4*L)),np.nan*np.
      zeros((13,4*L))
2349 #     Xa = sorted(set(X))
2350 #     #print(Xq.shape)
2351 #     #print(X)
2352 #     #print(len(X))
2353 #     #print(Xa[0],X[9])
2354 #     for numb in range(len(Xa)):
2355 #         aux = 0
2356 #         for numb2 in range(len(Xv)):
2357 #             if Xa[numb] == Xv[numb2]:
2358 #                 #print(aux,numb,numb2)
2359 #                 Xq[aux][numb] = Xv[numb2]
2360 #                 Yq[aux][numb] = Yv[numb2]
2361 #                 Uq[aux][numb] = U[numb2]
2362 #                 Vq[aux][numb] = V[numb2]
2363 #                 Iq[aux][numb] = Il[numb2]
2364 #                 aux = aux + 1
2365 #
=====
2366 #
=====

2367 #     cell = [16,]
2368 #     for cel in range()
2369 #
2370 #
=====

2371 #print(Xq[:,20])
2372 #plt.quiver(Xq,Yq,Uq,Vq,Iq)#,cmap=plt.cm.jet,scale=20)#, units="xy
    ")
2373 # mlab.quiver3d(Xv,Yv,Zv,U,V,W)
2374 #print(len(Xv),len(Yv),len(Zv),len(U),len(V),len(W))
2375 #plt.streamplot(Xq, Yq, Uq, Vq, color=np.log10(Iq),cmap="autumn")
2376 #plt.scatter(X,Y,s =300)#,c=Il, cmap=plt.cm.jet,norm =None,alpha
    =0.4)
2377 # plt.colorbar()
2378
2379

```

```

2380
2381
2382
2383
2384
2385 def current (N,h,Gn,b=0) :
2386     #fig = plt.subplots()
2387     #fig = mlab.figure(size=(1024, 1024),bgcolor = (1,1,1), fgcolor =
        (0.5, 0.5, 0.5))
2388
2389     figz = (70,10)
2390
2391     fig1, ax1 = plt.subplots(figsize=figz)
2392
2393     a = len(h)
2394     n = 2*N
2395     L = int(a/n)
2396     pos = []
2397     c = 1
2398     X = []
2399     Y = []
2400     Z = []
2401     if b==1:
2402         b = 3/4
2403     for l in range(L):
2404         #print(l)
2405         bi = 1+b
2406         bp = 4-b
2407         posy = 1
2408         for i in range(c,c+n):
2409             #print(i)
2410             if i%2 != 0:
2411                 x = bi+(9/2)*l
2412                 #print(x)
2413                 y = posy
2414 #
=====
2415 #             h[i-1][i-1]
2416 #             if h[i-1][i-1].real != 0:
2417 #                 #print(i)
2418 #                 #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2419 #                 mlab.points3d(x,y,0)

```

```

2420 #         else:
2421 #             #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
2422 #             mlab.points3d(x,y,0)
2423 #
=====
2424         pos.append([x,y])
2425         X.append(x)
2426         Y.append(y)
2427         Z.append(0)
2428         if bi == 1:
2429             bi = 7/4
2430         else:
2431             bi = 1
2432     else:
2433         x = bp+(9/2)*l
2434         y = posy
2435 #
=====
2436 #         h[i-1][i-1]
2437 #         if h[i-1][i-1].real != 0:
2438 #             #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2439 #             mlab.points3d(x,y,0)
2440 #             #print(i)
2441 #         else:
2442 #             #plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
2443 #             mlab.points3d(x,y,0)
2444 #
=====
2445         pos.append([x,y])
2446         X.append(x)
2447         Y.append(y)
2448         Z.append(0)
2449         if bp == 4:
2450             bp = 13/4
2451         else:
2452             bp = 4
2453         posy = posy + (3/4)*(np.sqrt(2))
2454     c = c + n
2455     #mlab.points3d(X,Y,Z,scale_factor=0.55,color=(0.5,0.5,0.5))
2456     for k in range(1,a+1):

```

```

2457     for j in range(1,a+1):
2458         if h[k-1][j-1] != 0:
2459             #print(k,j)
2460             if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
2461                 ax1.plot([pos[k-1][0],pos[j-1][0]], [pos[k-1][1],
pos[j-1][1]], 'k--', color="gray", alpha = 0.6)
2462                 #mlab.plot3d([pos[k-1][0],pos[j-1][0]], [pos[k
-1][1],pos[j-1][1]], [0,0], line_width=3, tube_radius=0.1)
2463             else:
2464                 ax1.plot([pos[k-1][0],pos[j-1][0]], [pos[k-1][1],
pos[j-1][1]], color = "black", alpha = 0.6)
2465                 #mlab.plot3d([pos[k-1][0],pos[j-1][0]], [pos[k
-1][1],pos[j-1][1]], [0,0], line_width=3, tube_radius=0.1)
2466 #
=====

2467
2468 It = []
2469 Xt = []
2470 Yt = []
2471 Zt = []
2472 Ut = []
2473 Vt = []
2474 Wt = []
2475
2476 for k in range(1,a+1):
2477     Il = []
2478     Xv = []
2479     Yv = []
2480     Zv = []
2481     U = []
2482     V = []
2483     W = []
2484
2485     #entrando
2486
2487     x1,y1 = pos[k-1][0],pos[k-1][1]
2488     for j in range(1,a+1):
2489         if h[k-1][j-1] != 0:
2490             I = ((h[k-1][j-1])*(Gn[k-1][j-1])).imag
2491
2492             x2,y2 = pos[j-1][0],pos[j-1][1]
2493

```

```

2494         iv = np.array([[ (x2-x1) ], [ (y2-y1) ]])
2495         norm = np.linalg.norm(iv)
2496         iv = (1/norm)* (iv)
2497
2498
2499         nor1 = np.sqrt (x1**2+y1**2)
2500         nor2 = np.sqrt (x2**2+y2**2)
2501
2502         cang = (x1*x2 + y1*y2)/(nor1*nor2)
2503
2504         ang = np.arccos (cang)
2505
2506         xv = iv[0]
2507         yv = iv[1]
2508
2509
2510         Xi = I*np.cos (ang)
2511         Yi = I*np.sin (ang)
2512
2513         fac = 1#np.sqrt (Xi**2+Yi**2)
2514
2515         Il.append(I)
2516         Xv.append(x1)
2517         Yv.append(y1)
2518         Zv.append(0)
2519         u = Xi/fac#xv[0]
2520         v = Yi/fac#yv[0]
2521         U.append(u)
2522         V.append(v)
2523         W.append(0)
2524
2525
2526     #entrando
2527     for j in range (1,a+1):
2528         if h[j-1][k-1] != 0:
2529             I = ((h[j-1][k-1])*(Gn[j-1][k-1])).imag
2530
2531             x2,y2 = pos[j-1][0],pos[j-1][1]
2532
2533             iv = np.array([[ -(x2-x1) ], [ -(y2-y1) ]])
2534             norm = np.linalg.norm(iv)
2535             iv = (1/norm)* (iv)
2536

```

```
2537
2538     nor1 = np.sqrt(x1**2+y1**2)
2539     nor2 = np.sqrt(x2**2+y2**2)
2540
2541     cang = (x1*x2 + y1*y2)/(nor1*nor2)
2542
2543     ang = np.arccos(cang)
2544
2545     xv = iv[0]
2546     yv = iv[1]
2547
2548
2549     Xi = I*np.cos(ang)
2550     Yi = I*np.sin(ang)
2551
2552     fac = 1#np.sqrt(Xi**2+Yi**2)
2553
2554     Il.append(I)
2555     Xv.append(x1)
2556     Yv.append(y1)
2557     Zv.append(0)
2558     u = Xi/fac#xv[0]
2559     v = Yi/fac#yv[0]
2560     U.append(u)
2561     V.append(v)
2562     W.append(0)
2563
2564
2565     Xv,Yv,Zv,U,V,W,Il = np.array(Xv),np.array(Yv),np.array(Zv),np.
array(U),np.array(V),np.array(W),np.array(Il)
2566     Xt.append(x1)
2567     Yt.append(y1)
2568
2569     fact = np.sqrt(np.sum(Il**2))
2570
2571     ut = np.sum(U)/fact
2572     vt = np.sum(V)/fact
2573
2574
2575     norM = np.sqrt(ut**2+vt**2)
2576
2577
2578     Ut.append(ut/norM)
```

```

2579     Vt.append(vt/norM)
2580     It.append(np.sqrt(ut**2+vt**2))
2581
2582
2583
2584     import matplotlib.colors as mcolors
2585     import matplotlib.colorbar as mcolorbar
2586
2587     nz = mcolors.Normalize()
2588     nz.autoscale(It)
2589
2590     fz = 48
2591
2592
2593
2594
2595     #width=0.003,scale=750#,width=0.003,scale=600
2596     Q = ax1.quiver(Xt, Yt,np.array(Ut), np.array(Vt),color=cm.jet(nz(It)),width=0.002,scale=70)
2597     #Q = ax1.quiver(Xc, Yc,np.array(Uc), np.array(Vc),color=cm.jet(nz(Ic)),width=0.003,scale=400)# units='x', pivot='mid', width=0.3,
2598     #length=5,scale=7)#,width=0.022)#,width=0.022)#,scale = 0.015)
2599     #qk = ax1.quiverkey(Q, 0.9, 0.9, 1, r'$1 \frac{m}{s}$', labelpos='E',coordinates='figure')
2600     ax1.set_xlim(0,80)
2601     ax1.set_ylim(0,11)
2602
2603     plt.scatter(Xt,Yt, marker='o',s=400,c=It, cmap=plt.cm.jet,norm = None,alpha=0.4)
2604
2605
2606     lb = 0
2607
2608     ax1.spines['left'].set_linewidth(lb)
2609     ax1.spines['right'].set_linewidth(lb)
2610     ax1.spines['top'].set_linewidth(lb)
2611     ax1.spines['bottom'].set_linewidth(lb)
2612
2613     ax1.set_yticklabels([])
2614     ax1.set_xticklabels([])
2615
2616

```

```

2617 ax1.xaxis.set_ticks_position('both')
2618 ax1.yaxis.set_ticks_position('both')
2619 ax1.tick_params(axis='both', which='major', labels=fz, direction =
    "in", length = 0, width = 30)
2620
2621
2622 #clim = np.linspace(min(It),max(It),5)
2623 clim = np.linspace(0,max(It),5)
2624 clb = plt.colorbar(ticks=clim,location="bottom")
2625 clb.ax.tick_params(labels=fz)
2626 clb.ax.tick_params(labels=fz)
2627 clb.set_label("Local current",rotation = 0,fontsize=fz,labelpad =
    5)
2628 plt.clim(vmin = 0, vmax = max(It))
2629
2630 plt.suptitle(r'$E = 0.0, U = 0.0, V_{g} = 0.02, V_{g} = 0$', fontsize
    = fz, x=0.5, y=0.95)
2631
2632
2633 #clb.ax.tick_params(labels=fz)
2634
2635
2636
2637
2638
2639 #plt.show()
2640
2641
2642
2643 def current_D(N,h,Gn,b=0):
2644     #fig = plt.subplots()
2645     #fig = mlab.figure(size=(1024, 1024),bgcolor = (1,1,1), fgcolor =
    (0.5, 0.5, 0.5))
2646
2647     figz = (70,10)
2648
2649     fig1, ax1 = plt.subplots(figsize=figz)
2650
2651     a = len(h)
2652     n = 2*N
2653     L = int(a/n)
2654     pos = []
2655     c = 1

```



```

2656     X = []
2657     Y = []
2658     Z = []
2659     if b==1:
2660         b = 3/4
2661     for l in range(L):
2662         #print(l)
2663         bi = 1+b
2664         bp = 4-b
2665         posy = 1
2666         for i in range(c,c+n):
2667             #print(i)
2668             if i%2 != 0:
2669                 x = bi+(9/2)*l
2670                 #print(x)
2671                 y = posy
2672 #
=====

2673 #             h[i-1][i-1]
2674 #             if h[i-1][i-1].real != 0:
2675 #                 #print(i)
2676 #                 #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2677 #                 mlab.points3d(x,y,0)
2678 #             else:
2679 #                 #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
2680 #                 mlab.points3d(x,y,0)
2681 #
=====

2682         pos.append([x,y])
2683         X.append(x)
2684         Y.append(y)
2685         Z.append(0)
2686         if bi == 1:
2687             bi = 7/4
2688         else:
2689             bi = 1
2690     else:
2691         x = bp+(9/2)*l
2692         y = posy
2693 #
=====

```

```

2694 #             h[i-1][i-1]
2695 #             if h[i-1][i-1].real != 0:
2696 #                 plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2697 #                 mlab.points3d(x,y,0)
2698 #                 #print(i)
2699 #             else:
2700 #                 plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
2701 #                 mlab.points3d(x,y,0)
2702 #
=====

2703         pos.append([x,y])
2704         X.append(x)
2705         Y.append(y)
2706         Z.append(0)
2707         if bp == 4:
2708             bp = 13/4
2709         else:
2710             bp = 4
2711         posy = posy + (3/4)*(np.sqrt(2))
2712         c = c + n
2713         #mlab.points3d(X,Y,Z,scale_factor=0.55,color=(0.5,0.5,0.5))
2714         for k in range(1,a+1):
2715             for j in range(1,a+1):
2716                 if h[k-1][j-1] != 0:
2717                     #print(k,j)
2718                     if h[k-1][j-1] != 0 and h[k-1][j-1] != 1:
2719                         ax1.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],
pos[j-1][1]],'k--',color="gray",alpha = 0.6)
2720                         #mlab.plot3d([pos[k-1][0],pos[j-1][0]],[pos[k
-1][1],pos[j-1][1]],[0,0],line_width=3,tube_radius=0.1)
2721                     else:
2722                         ax1.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],
pos[j-1][1]],color = "black",alpha = 0.6)
2723                         #mlab.plot3d([pos[k-1][0],pos[j-1][0]],[pos[k
-1][1],pos[j-1][1]],[0,0],line_width=3,tube_radius=0.1)
2724 #
=====

2725     X = np.array(X)
2726     Y = np.array(Y)
2727     Xd = sorted(set(X))

```

```

2728     Yd = sorted(set(Y))
2729     Xad,Yad = np.meshgrid(Xd,Yd)
2730     Zad = np.zeros((Xad.shape[0],Xad.shape[1]))
2731
2732     d0 = np.diag(Gn)
2733
2734     D = (np.abs(d0)**2).real
2735 #
=====
2736 #     c,d = 0,0
2737 #
2738 #     for k in range(len(X)):
2739 #         x = X[k]
2740 #         y = Y[k]
2741 #         z = D[k]
2742 #         Zad[d][c] = z
2743 #         if k < len(X) -1 and x != X[k+1]:
2744 #             c = c + 1
2745 #             d = -1
2746 #             d = d + 1
2747 #
=====
2748     plt.scatter(X,Y, marker='o',s=400,c=D, cmap=plt.cm.jet,norm =None,
2749               alpha=0.4)
2749 # plt.scatter(Xad,Yad)
2750 #
=====
2751 #     It = []
2752 #     Xt = []
2753 #     Yt = []
2754 #     Zt = []
2755 #     Ut = []
2756 #     Vt = []
2757 #     Wt = []
2758 #
2759 #     for k in range(1,a+1):
2760 #         Il = []
2761 #         Xv = []
2762 #         Yv = []
2763 #         Zv = []

```

```

2764 #         U = []
2765 #         V = []
2766 #         W = []
2767 #
2768 #         #entrando
2769 #
2770 #         x1,y1 = pos[k-1][0],pos[k-1][1]
2771 #         for j in range(1,a+1):
2772 #             if h[k-1][j-1] != 0:
2773 #                 I = ((h[k-1][j-1])*(Gn[k-1][j-1])).imag
2774 #
2775 #                 x2,y2 = pos[j-1][0],pos[j-1][1]
2776 #
2777 #                 iv = np.array([(x2-x1)],[(y2-y1)]])
2778 #                 norm = np.linalg.norm(iv)
2779 #                 iv = (1/norm)*(iv)
2780 #
2781 #
2782 #                 nor1 = np.sqrt(x1**2+y1**2)
2783 #                 nor2 = np.sqrt(x2**2+y2**2)
2784 #
2785 #                 cang = (x1*x2 + y1*y2)/(nor1*nor2)
2786 #
2787 #                 ang = np.arccos(cang)
2788 #
2789 #                 xv = iv[0]
2790 #                 yv = iv[1]
2791 #
2792 #
2793 #                 Xi = I*np.cos(ang)
2794 #                 Yi = I*np.sin(ang)
2795 #
2796 #                 fac = 1#np.sqrt(Xi**2+Yi**2)
2797 #
2798 #                 Il.append(I)
2799 #                 Xv.append(x1)
2800 #                 Yv.append(y1)
2801 #                 Zv.append(0)
2802 #                 u = Xi/fac#xv[0]
2803 #                 v = Yi/fac#yv[0]
2804 #                 U.append(u)
2805 #                 V.append(v)
2806 #                 W.append(0)

```

```

2807 #
2808 #
2809 #     #entrando
2810 #     for j in range(1,a+1):
2811 #         if h[j-1][k-1] != 0:
2812 #             I = ((h[j-1][k-1])*(Gn[j-1][k-1])).imag
2813 #
2814 #             x2,y2 = pos[j-1][0],pos[j-1][1]
2815 #
2816 #             iv = np.array([[-(x2-x1)],[-(y2-y1)]])
2817 #             norm = np.linalg.norm(iv)
2818 #             iv = (1/norm)*(iv)
2819 #
2820 #
2821 #             nor1 = np.sqrt(x1**2+y1**2)
2822 #             nor2 = np.sqrt(x2**2+y2**2)
2823 #
2824 #             cang = (x1*x2 + y1*y2)/(nor1*nor2)
2825 #
2826 #             ang = np.arccos(cang)
2827 #
2828 #             xv = iv[0]
2829 #             yv = iv[1]
2830 #
2831 #
2832 #             Xi = I*np.cos(ang)
2833 #             Yi = I*np.sin(ang)
2834 #
2835 #             fac = 1#np.sqrt(Xi**2+Yi**2)
2836 #
2837 #             Il.append(I)
2838 #             Xv.append(x1)
2839 #             Yv.append(y1)
2840 #             Zv.append(0)
2841 #             u = Xi/fac#xv[0]
2842 #             v = Yi/fac#yv[0]
2843 #             U.append(u)
2844 #             V.append(v)
2845 #             W.append(0)
2846 #
2847 #
2848 #             Xv,Yv,Zv,U,V,W,Il = np.array(Xv),np.array(Yv),np.array(Zv),np
                .array(U),np.array(V),np.array(W),np.array(Il)

```

```

2849 #         Xt.append(x1)
2850 #         Yt.append(y1)
2851 #
2852 #         fact = np.sqrt(np.sum(I1**2))
2853 #
2854 #         ut = np.sum(U) / fact
2855 #         vt = np.sum(V) / fact
2856 #
2857 #
2858 #         norM = np.sqrt(ut**2+vt**2)
2859 #
2860 #
2861 #         Ut.append(ut/norM)
2862 #         Vt.append(vt/norM)
2863 #         It.append(np.sqrt(ut**2+vt**2))
2864 #
=====
2865 #
=====
2866 #
2867 #
2868 #
2869 #     import matplotlib.colors as mcolors
2870 #     import matplotlib.colorbar as mcolorbar
2871 #
2872 #     nz = mcolors.Normalize()
2873 #     nz.autoscale(It)
2874 #
2875 #     fz = 48
2876 #
2877 #
2878 #
2879 #
2880 #     #width=0.003,scale=750#,width=0.003,scale=600
2881 #     Q = ax1.quiver(Xt, Yt,np.array(Ut), np.array(Vt),color=cm.jet(nz(
2882 #         It)),width=0.002,scale=70)
2883 #     #Q = ax1.quiver(Xc, Yc,np.array(Uc), np.array(Vc),color=cm.jet(nz(
2884 #         Ic)),width=0.003,scale=400)# units='x', pivot='mid', width=0.3,
2885 #         length=5,scale=7)#,width=0.022)#,width=0.022)#,scale = 0.015)
2886 #     #qk = ax1.quiverkey(Q, 0.9, 0.9, 1, r'$1 \frac{m}{s}$', labelpos
2887 #         ='E',coordinates='figure')

```

```

2884 #     ax1.set_xlim(0,80)
2885 #     ax1.set_ylim(0,11)
2886 #
2887 #
2888 #     plt.scatter(Xt,Yt, marker='o',s=400,c=It, cmap=plt.cm.jet,norm =
None,alpha=0.4)
2889 #
2890 #
2891 #     lb = 0
2892 #
2893 #     ax1.spines['left'].set_linewidth(lb)
2894 #     ax1.spines['right'].set_linewidth(lb)
2895 #     ax1.spines['top'].set_linewidth(lb)
2896 #     ax1.spines['bottom'].set_linewidth(lb)
2897 #
2898 #     ax1.set_yticklabels([])
2899 #     ax1.set_xticklabels([])
2900 #
2901 #
2902 #     ax1.xaxis.set_ticks_position('both')
2903 #     ax1.yaxis.set_ticks_position('both')
2904 #     ax1.tick_params(axis='both', which='major',labelsize=fz,direction
= "in",length = 0, width = 30)
2905 #
2906 #
2907 #     #clim = np.linspace(min(It),max(It),5)
2908 #     clim = np.linspace(0,max(It),5)
2909 #     clb = plt.colorbar(ticks=clim,location="bottom")
2910 #     clb.ax.tick_params(labelsize=fz)
2911 #     clb.ax.tick_params(labelsize=fz)
2912 #     clb.set_label("Local current",rotation = 0,fontsize=fz,labelpad =
5)
2913 #     plt.clim(vmin = 0, vmax = max(It))
2914 #
2915 #     plt.suptitle(r'$E = 0.0, U = 0.0,V_{g} = 0.02 ,V_{g} = 0$',
fontsize = fz,x=0.5,y=0.95)
2916 #
2917 #
=====

2918
2919 #clb.ax.tick_params(labelsize=fz)
2920

```

```
2921
2922
2923
2924
2925     #plt.show()
2926
2927
2928
2929 def densityA_mult(N,H,D):
2930
2931     X = []
2932     Y = []
2933
2934     for h in H:
2935
2936
2937         a = len(h)
2938         ind = H.index(h)
2939         n = 2*(N[ind])
2940         L = int(a/n)
2941         pos = []
2942         c = 1
2943
2944         for l in range(L):
2945             #print(l)
2946             bi = 1
2947             bp = 4
2948             posy = 1
2949             for i in range(c,c+n):
2950                 if i%2 != 0:
2951                     x = bi+4*l
2952                     y = posy
2953                     h[i-1][i-1]
2954                     #if h[i-1][i-1].real != 0:
2955                         # print(i)
2956                         # plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2957                 #else:
2958                     #plt.scatter(x,y,color = "gray",s = 50,alpha=0.6)
2959                     pos.append([x,y])
2960                     X.append(x)
2961                     Y.append(y)
2962                 if bi == 1:
2963                     bi = 2
```



```

2964         else:
2965             bi = 1
2966         else:
2967             x = bp+4*1
2968             y = posy
2969             h[i-1][i-1]
2970             #if h[i-1][i-1].real != 0:
2971                 #plt.scatter(x,y,color = "red",s = 50,alpha=0.6)
2972             # print(i)
2973             #else:
2974                 #plt.scatter(x,y,color = "gray",s =50,alpha=0.6)
2975             pos.append([x,y])
2976             X.append(x)
2977             Y.append(y)
2978             if bp == 4:
2979                 bp = 3
2980             else:
2981                 bp = 4
2982                 posy = posy + 1
2983             c = c + n
2984             fig, axs = plt.subplots()
2985             plt.scatter(X,Y,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha
=0.4) #plt.cm.jet
2986             plt.colorbar()
2987             plt.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
2988             for k in range(1,a+1):
2989                 for j in range(1,a+1):
2990                     if h[k-1][j-1] != 0:
2991                         #print(k,j)
2992                         plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos
[j-1][1]],color = "black",alpha = 0.4)
2993
2994
2995
2996 def densityZ(N,h,D):
2997
2998
2999
3000     a = len(h)
3001     n2 = 2*N
3002     L = int(a/n2)
3003     pos = []
3004

```

```

3005     c = 1
3006     X = []
3007     Y = []
3008     Z = []
3009
3010     for l in range(L):
3011         #print(l)
3012         b = 1
3013         posy = 1
3014         for i in range(c, c+n2):
3015             if i%2 == 0:
3016                 #print(i)
3017                 x = b+2*1
3018                 y = posy
3019                 pos.append([x,y])
3020                 X.append(x)
3021                 Y.append(y)
3022                 #Z.append(sum(np.abs(psi[i-1].real)))
3023                 #Z.append(h1[i-1][i-1].real)
3024                 #if density == "psi":
3025
3026             else:
3027                 #print(i)
3028                 x = b+2*1
3029                 y = posy
3030                 pos.append([x,y])
3031                 X.append(x)
3032                 Y.append(y)
3033                 #Z.append(sum(np.abs(psi[i-1].real)))
3034                 #Z.append(h1[i-1][i-1].real)
3035                 if b == 1:
3036                     b = 0
3037                 else:
3038                     b = 1
3039                 posy = posy + 1
3040             c = c + n2
3041
3042     #normalize =mt.colors.Normalize(vmin=0, vmax=1, clip=False)
3043     #nor = sum(Z)
3044     #Z = Z/nor
3045     fig, axs = plt.subplots()
3046     plt.scatter(X,Y,s =300,c=D, cmap=plt.cm.jet,norm =None,alpha=0.4) #
3047     plt.cm.jet
3048     plt.colorbar()

```

```

3047 plt.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
3048 for k in range(1,a+1):
3049     for j in range(1,a+1):
3050         if h[k-1][j-1] != 0:
3051             #print(k-1,j-1)
3052             plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j-1][1]],color = "black",alpha = 0.4)
3053 plt.show()
3054
3055
3056
3057
3058
3059 def density(N,L,en,defect,density = "stats"):
3060     (v0,h,v1) = fitaZ(0,N,1,0,0)
3061     er = 0.01
3062     eta = 0.01
3063     e = en+1j*eta
3064     gl,gr = G1(h,v0,v1,e,er)
3065     h = HL(L,N,h,v0,v1)#Gn.H(L,N,h1,v0,v1)
3066     #if defect == "Y" or "y" or "s" or "S":
3067     #h = remov(h,v0,defect)
3068     din = len(h[0])
3069     E1 = np.identity(din,complex)
3070     E1 = np.dot(e,E1)
3071
3072     #a = L*len(h1)
3073     n = len(gl)
3074     m = len(h)
3075     zq1 = np.zeros((m-n,m-n),complex)
3076     z1 = np.zeros((n,m-n),complex)
3077
3078     sigl = np.dot(np.dot(v0,gl),v1)
3079
3080     sigL = np.block([[sigl,z1],[np.transpose(z1),zq1]])
3081
3082     sigr = np.dot(np.dot(v1,gr),v0)
3083
3084     Vrc = np.block([[z1,v1],[zq1,np.transpose(z1)]])
3085
3086     Vlc = np.block([[np.transpose(z1),zq1],[v0,z1]])
3087
3088     Vc = Vrc + Vlc

```

```

3089
3090     sigR = np.block([[zq1,np.transpose(z1)],[z1,sigr]])
3091     h1 = h
3092     G2 = np.linalg.inv(E1-h1-sigR-sigL)
3093     E, psi = np.linalg.eigh(h+Vc)
3094     #print(E)
3095     #print(psi)
3096     h1 = np.diag(E)
3097     n2 = 2*N
3098     #L = int(din/n2)
3099     pos = []
3100     c = 1
3101     X = []
3102     Y = []
3103     Z = []
3104     if density == "prob":
3105         print(E)
3106         ener = input("Posição da energia: ")
3107         print(E[int(ener)-1])
3108     for l in range(L):
3109         #print(l)
3110         b = 1
3111         posy = 1
3112         for i in range(c,c+n2):
3113             if i%2 == 0:
3114                 #print(i)
3115                 x = b+2*1
3116                 y = posy
3117                 pos.append([x,y])
3118                 X.append(x)
3119                 Y.append(y)
3120                 if density == "prob":
3121                     Z.append((psi[int(ener)-1][i-1]*np.conj(psi[int(ener)
3122 )-1][i-1])).real)
3123                 if density == "stats":
3124                     Z.append((-1/(np.pi))*G2[i-1][i-1].imag)
3125                     #Z.append(sum(np.abs(psi[i-1].real)))
3126                     #Z.append(h1[i-1][i-1].real)
3127                     #if density == "psi":
3128
3129             else:
3130                 #print(i)
3131                 x = b+2*1

```

```

3131         y = posy
3132         pos.append([x,y])
3133         X.append(x)
3134         Y.append(y)
3135         if density == "prob":
3136             Z.append((psi[int(ener)-1][i-1]*np.conj(psi[int(ener)
3137             )-1][i-1])).real)
3138         if density == "stats":
3139             Z.append((-1/(np.pi))*G2[i-1][i-1].imag)
3140             #Z.append(sum(np.abs(psi[i-1].real)))
3141             #Z.append(h1[i-1][i-1].real)
3142         if b == 1:
3143             b = 0
3144         else:
3145             b = 1
3146         posy = posy + 1
3147         c = c + n2
3148         #normalize =mt.colors.Normalize(vmin=0, vmax=1, clip=False)
3149         #nor = sum(Z)
3150         #Z = Z/nor
3151         fig, axs = plt.subplots()
3152         plt.scatter(X,Y,s =300,c=Z, cmap=plt.cm.jet,norm =None,alpha=0.4) #
3153         plt.cm.jet
3154         plt.colorbar()
3155         plt.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
3156         if density == "prob":
3157             plt.suptitle("energy = %.6f"%(E[int(ener)-1]))
3158         for k in range(1,din+1):
3159             for j in range(1,din+1):
3160                 if h[k-1][j-1] != 0:
3161                     #print(k-1,j-1)
3162                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
3163                     -1][1]],color = "black",alpha = 0.2)
3164         plt.show()
3165         #return E,psi
3166
3167 def density2(N,L,en,defect,density = "stats"):
3168     (v0,h,v1) = fitaZ(0,N,1,0,0)
3169     er = 0.01
3170     eta = 0.01
3171     e = en+1j*eta
3172     gl,gr = G1(h,v0,v1,e,er)

```

```

3171     h = HL(L,N,h,v0,v1) #Gn.H(L,N,h1,v0,v1)
3172     #if defect == "Y" or "y" or "s" or "S":
3173     h = remov(h,v0,defect)
3174     din = len(h[0])
3175     E1 = np.identity(din,complex)
3176     E1 = np.dot(e,E1)
3177
3178     #a = L*len(h1)
3179     n = len(g1)
3180     m = len(h)
3181     zq1 = np.zeros((m-n,m-n),complex)
3182     z1 = np.zeros((n,m-n),complex)
3183
3184     sigl = np.dot(np.dot(v0,g1),v1)
3185
3186     sigL = np.block([[sigl,z1],[np.transpose(z1),zq1]])
3187
3188     sigr = np.dot(np.dot(v1,gr),v0)
3189
3190     sigR = np.block([[zq1,np.transpose(z1)],[z1,sigr]])
3191     h1 = h
3192     G2 = np.linalg.inv(E1-h1-sigR-sigL)
3193     E, psi = np.linalg.eigh(h1)
3194     #print(E)
3195     #print(psi)
3196     h1 = np.diag(E)
3197     n2 = 2*N
3198     #L = int(din/n2)
3199     pos = []
3200     c = 1
3201     X = []
3202     Y = []
3203     Z = []
3204     if density == "prob":
3205         print(E)
3206         ener = input("Posição da energia: ")
3207         print(E[int(ener)-1])
3208     for l in range(L):
3209         #print(l)
3210         b = 1
3211         posy = 1
3212         for i in range(c,c+n2):
3213             if i%2 == 0:

```

```

3214         #print(i)
3215         x = b+2*l
3216         y = posy
3217         pos.append([x,y])
3218         X.append(x)
3219         Y.append(y)
3220         if density == "prob":
3221             Z.append((psi[int(ener)-1][i-1]*np.conj(psi[int(ener
)-1][i-1])).real)
3222         if density == "stats":
3223             Z.append((-1/(np.pi))*G2[i-1][i-1].imag)
3224             #Z.append(sum(np.abs(psi[i-1].real)))
3225             #Z.append(h1[i-1][i-1].real)
3226             #if density == "psi":
3227
3228     else:
3229         #print(i)
3230         x = b+2*l
3231         y = posy
3232         pos.append([x,y])
3233         X.append(x)
3234         Y.append(y)
3235         if density == "prob":
3236             Z.append((psi[int(ener)-1][i-1]*np.conj(psi[int(ener
)-1][i-1])).real)
3237         if density == "stats":
3238             Z.append((-1/(np.pi))*G2[i-1][i-1].imag)
3239             #Z.append(sum(np.abs(psi[i-1].real)))
3240             #Z.append(h1[i-1][i-1].real)
3241         if b == 1:
3242             b = 0
3243         else:
3244             b = 1
3245         posy = posy + 1
3246         c = c + n2
3247     #normalize =mt.colors.Normalize(vmin=0, vmax=1, clip=False)
3248     #nor = sum(Z)
3249     #Z = Z/nor
3250     fig = plt.figure()
3251     ax = plt.axes(projection='3d')
3252     ax.scatter3D(X, Y, Z, cmap='Greens')
3253     #plt.scatter(X,Y,s =300,c=Z, cmap=plt.cm.jet,norm =None,alpha=0.4)#
plt.cm.jet

```

```

3254     plt.colorbar()
3255     plt.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
3256     for k in range(1,din+1):
3257         for j in range(1,din+1):
3258             if h[k-1][j-1] != 0:
3259                 #print(k,j)
3260                 plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j-1][1]],color = "black",alpha = 0.2)
3261     plt.show()
3262
3263
3264 def densityE(pos,density = "stats"):
3265     N=6
3266     L=20
3267     (v0,h,v1) = fitaZ(0,N,1,0,0)
3268
3269     arq = open('sig.out','r')
3270
3271     aur = []
3272     aui = []
3273     enl = []
3274     for line in arq:
3275         line = line.strip()
3276         a,b,c = line.split()
3277         enl.append(float(a))
3278         aur.append(float(b))
3279         aui.append(float(c))
3280     arq.close()
3281
3282     en = enl[pos]
3283     si = aur[pos] + 1j*aui[pos]
3284
3285     er = 0.01
3286     eta = 0.001/2
3287     e = en+1j*eta
3288     gl,gr = G1(h,v0,v1,e,er)
3289     h = HL(L,N,h,v0,v1) #Gn.H(L,N,h1,v0,v1)
3290     #if defect == "Y" or "y" or "s" or "S":
3291     h = remov(h,v0,1)
3292     din = len(h[0])
3293     E1 = np.identity(din,complex)
3294     E1 = np.dot(e,E1)
3295

```



```

3296
3297     SigE = np.identity(din, complex)
3298
3299     SigE[139][138] = si
3300     SigE[139][140] = si
3301     SigE[139][150] = si
3302
3303     SigE[138][139] = si
3304     SigE[140][139] = si
3305     SigE[150][139] = si
3306
3307     #a = L*len(h1)
3308     n = len(g1)
3309     m = len(h)
3310     zq1 = np.zeros((m-n, m-n), complex)
3311     z1 = np.zeros((n, m-n), complex)
3312
3313     sigl = np.dot(np.dot(v0, g1), v1)
3314
3315     sigL = np.block([[sigl, z1], [np.transpose(z1), zq1]])
3316
3317     sigr = np.dot(np.dot(v1, gr), v0)
3318
3319     Vrc = np.block([[z1, v1], [zq1, np.transpose(z1)]])
3320
3321     Vlc = np.block([[np.transpose(z1), zq1], [v0, z1]])
3322
3323     Vc = Vrc + Vlc
3324
3325     sigR = np.block([[zq1, np.transpose(z1)], [z1, sigr]])
3326     h1 = h
3327     G2 = np.linalg.inv(E1-h1-sigR-sigL-SigE)
3328     E, psi = np.linalg.eigh(h+Vc)
3329     #print(E)
3330     #print(psi)
3331     h1 = np.diag(E)
3332     n2 = 2*N
3333     #L = int(din/n2)
3334     pos = []
3335     c = 1
3336     X = []
3337     Y = []
3338     Z = []

```

```

3339     if density == "prob":
3340         print(E)
3341         ener = input("Posição da energia: ")
3342         print(E[int(ener)-1])
3343     for l in range(L):
3344         #print(l)
3345         b = 1
3346         posy = 1
3347         for i in range(c,c+n2):
3348             if i%2 == 0:
3349                 #print(i)
3350                 x = b+2*l
3351                 y = posy
3352                 pos.append([x,y])
3353                 X.append(x)
3354                 Y.append(y)
3355                 if density == "prob":
3356                     Z.append((psi[int(ener)-1][i-1]*np.conj(psi[int(ener)
3357 )-1][i-1])).real)
3358                     if density == "stats":
3359                         Z.append((-1/(np.pi))*G2[i-1][i-1].imag)
3360                         #Z.append(sum(np.abs(psi[i-1].real)))
3361                         #Z.append(h1[i-1][i-1].real)
3362                         #if density == "psi":
3363             else:
3364                 #print(i)
3365                 x = b+2*l
3366                 y = posy
3367                 pos.append([x,y])
3368                 X.append(x)
3369                 Y.append(y)
3370                 if density == "prob":
3371                     Z.append((psi[int(ener)-1][i-1]*np.conj(psi[int(ener)
3372 )-1][i-1])).real)
3373                     if density == "stats":
3374                         Z.append((-1/(np.pi))*G2[i-1][i-1].imag)
3375                         #Z.append(sum(np.abs(psi[i-1].real)))
3376                         #Z.append(h1[i-1][i-1].real)
3377                     if b == 1:
3378                         b = 0
3379                     else:
3380                         b = 1

```

```

3380         posy = posy + 1
3381         c = c + n2
3382         #normalize =mt.colors.Normalize(vmin=0, vmax=1, clip=False)
3383         #nor = sum(Z)
3384         #Z = Z/nor
3385         fig, axs = plt.subplots()
3386         plt.scatter(X,Y,s =300,c=Z, cmap=plt.cm.jet,norm =None,alpha=0.4) #
3387         plt.colorbar()
3388         plt.scatter(X,Y,color = "black",s = 10,alpha = 0.8)
3389         plt.suptitle("Energy = %.3f"%(en))
3390         if density == "prob":
3391             plt.suptitle("energy = %.6f"%(E[int(ener)-1]))
3392         for k in range(1,din+1):
3393             for j in range(1,din+1):
3394                 if h[k-1][j-1] != 0:
3395                     #print(k-1,j-1)
3396                     plt.plot([pos[k-1][0],pos[j-1][0]],[pos[k-1][1],pos[j
3397                     -1][1]],color = "black",alpha = 0.2)
3398
3399
3400 def alea(h,v):
3401     a = len(v)
3402     b = len(h)
3403     #if b-a-2 > a-2:
3404         #r = random.sample(range(a+2, b-a-2), n)
3405     r = [30+a]#[6,10,12]
3406     #r = range(a,b,10)
3407     for i in r:
3408         if i < b-a:
3409             h[i-1][i-1] = 500
3410             print(i)
3411
3412     return h
3413
3414 def remov(h,v,nu,r1):
3415     a = len(v)
3416     b = len(h)
3417     #if b-a-2 > a-2:
3418         #r = random.sample(range(a+2, b-a-2), n)
3419     r = [r1]#[8*a]#[35+1*3*a]#[20+10*a,20+20*a]#[6,10,12]
3420     #print(nu)

```

```

3421     #for ind in range(nu):
3422         #r.append(90+4*40*(ind))
3423     h0 = h
3424     for k in r:
3425         #print(k)
3426         for i in range(b):
3427             if h0[k-1][i-1] != 0:
3428                 print(k-1,i-1)
3429                 h0[k-1][i-1] = 0
3430                 h0[i-1][k-1] = 0
3431     h0[r1-1][r1-1] = 100
3432
3433     #print(i)
3434
3435     return h0
3436
3437 def removA(h,v,nu):
3438     a = len(v)
3439     b = len(h)
3440     #if b-a-2 > a-2:
3441         #r = random.sample(range(a+2, b-a-2), n)
3442     #r = [30+49*a+13]#[30+8*3*a]#[30+16*3*a]#[20+10*a,20+20*a
3443 ]#[6,10,12]
3444     r = [51*a]#[51*a]#[30+49*a+2]#[,30+49*a+1]
3445     #print(nu)
3446     #for ind in range(nu):
3447         #r.append(90+4*40*(ind))
3448     h0 = h
3449     for k in r:
3450         print(k)
3451         for i in range(b):
3452             if h0[k-1][i-1] != 0:
3453                 print(k-1,i-1)
3454                 h0[k-1][i-1] = 0
3455                 h0[i-1][k-1] = 0
3456
3457         #print(i)
3458
3459     return h0
3459 #def remo(h,v,nu):
3460 #     a = len(v)
3461 #     b = len(h)
3462 #     #if b-a-2 > a-2:

```

```

3463     #r = random.sample(range(a+2, b-a-2), n)
3464 # r = [30+1*3*a]#[20+10*a,20+20*a]#[6,10,12]
3465 #print(nu)
3466 #for ind in range(nu):
3467     #r.append(90+4*40*(ind))
3468 #h0 = np.zeros((b-1,b-1),complex)
3469 #for k in r:
3470     # print(k)
3471     # for i in range(b-1):
3472         # for j in range(b-1):
3473             # if i != k-1 and j != k-1:
3474                 # h0[i][j] = h[i][j]
3475 # return h0
3476
3477 @njit(nogil=True)
3478 def remov2(h,v,nu):
3479     a = len(v)
3480     b = len(h)
3481     #if b-a-2 > a-2:
3482         #r = random.sample(range(a+2, b-a-2), n)
3483     r = np.random.randint(a, high=b-a, size=nu,dtype=np.int8)
3484     h0 = h
3485     for k in r:
3486         for i in np.arange(b):
3487             h0[k-1][i-1] = 0
3488             h0[i-1][k-1] = 0
3489
3490     #print(i)
3491
3492     return h0
3493
3494 @njit(nogil=True)
3495 def removf(h,v,nu,L):
3496     a = len(v)
3497     b = len(h)
3498     #if b-a-2 > a-2:
3499         #r = random.sample(range(a+2, b-a-2), n)
3500     #r = np.random.randint(a, high=b-a, size=nu,dtype=np.int8)
3501     r = np.random.randint(3,L-2,nu)#nu
3502     #print(r)
3503     h0 = h
3504     for k in r:
3505         h0[k-1][k-1] = 100+0*1j

```

```

3506         for i in np.arange(b):
3507             h0[k-1][i-1] = 0+0*1j
3508             h0[i-1][k-1] = 0+0*1j
3509
3510         #print(i)
3511
3512         return h0
3513
3514 def adc(h,v,i,gam2,r):
3515     a = len(v)
3516     b = len(h)
3517     #if b-a-2 > a-2:
3518         #r = random.sample(range(a+2, b-a-2), n)
3519     #if i == 1:
3520         #    r = 30+49*a+2#41#[6,10,12]
3521     #if i == 2:
3522         #    r = 30+49*a+2 + 4*a+1
3523     print(r)
3524     h0 = np.zeros((b+1,b+1),complex)
3525     #h0[41][41] = 0
3526     #h0[41][40] = 1/4
3527     #h0[40][41] = 1/4
3528     #for i in range(b+1):
3529         #    for j in range(b+1):
3530             #        if i < r and j < r:
3531                 #            h0[i][j] = h[i][j]
3532             #        if i > r and j > r:
3533                 #            h0[i][j] = h[i-1][j-1]
3534     h0[:r, :r] = h[:r, :r]
3535     h0[r+1:, r+1:] = h[r:, r:]
3536
3537     h0[:r, r+1:] = h[:r, r:]
3538     h0[r+1:, :r] = h[r:, :r]
3539
3540     h0[r][r-1] = gam2
3541     h0[r-1][r] = gam2
3542
3543     return h0
3544
3545 def adcl(h,v,n,gam2,r):
3546     a = len(v)
3547     b = len(h)
3548     #if b-a-2 > a-2:

```

```

3549         #r = random.sample(range(a+2, b-a-2), n)
3550     #
=====
3551     #     if n ==1:
3552     #         r = 8*a
3553     #     if n == 2:
3554     #         r = 9*a+1
3555     #
=====
3556         #41#[6,10,12]
3557     #print(r)
3558     h0 = np.zeros((b+1,b+1),dtype=np.complex_)
3559     #h0[41][41] = 0
3560     #h0[41][40] = 1/4
3561     #h0[40][41] = 1/4
3562     #for i in range(b+1):
3563     #     for j in range(b+1):
3564     #         if i < r and j < r:
3565     #             h0[i][j] = h[i][j]
3566     #         if i > r and j > r:
3567     #             h0[i][j] = h[i-1][j-1]
3568     h0[:r, :r] = h[:r, :r]
3569     h0[r+1:, r+1:] = h[r:, r:]
3570
3571     h0[:r, r+1:] = h[:r, r:]
3572     h0[r+1:, :r] = h[r:, :r]
3573
3574     h0[r][r-1] = gam2
3575     h0[r-1][r] = gam2
3576     #
=====
3577     #     h0[r][r-1] = 1/50
3578     #     h0[r-1][r] = 1/50
3579     #
=====
3580
3581     return h0
3582
3583 def dalea(h,v,d):

```

```

3584     a = len(v)
3585     b = len(h)
3586     t = int((b*d))
3587     r = []
3588     for x in range(t):
3589         va = random.randint(a, b-a)
3590         r.append(va)
3591     for i in r:
3592         if i < b-a:
3593             h[i][i] = 0.5
3594             #print(i)
3595
3596     return h
3597
3598 def dremov(h,v,d):
3599     a = len(v)
3600     b = len(h)
3601     t = int((b*d))
3602     r = []
3603     for x in range(t):
3604         va = random.randint(a, b-a)
3605         r.append(va)
3606     h0 = h
3607     for k in r:
3608         for i in range(b):
3609             h0[k-1][i-1] = 0
3610             h0[i-1][k-1] = 0
3611     return h0
3612
3613 #
=====
3614 # @njit(nogil=True)
3615 # def cond_def(): #gera corretamente condutancia x energia
3616 #     N = 4
3617 #     gamma = 3
3618 #     #L = 10
3619 #
3620 #
3621 #
3622 #     eta = 0.001
3623 #     ldos = []
3624 #     e2 = []

```



```
3625 #     er = 0.001
3626 #
3627 #
3628 #
3629 #
3630 #     #l2 = [5]#= range(10,1000,40)
3631 #     energy = np.arange(-3,3,0.01)
3632 #
3633 #
3634 #     #l1 = []
3635 #     #med = []
3636 #     #for l in l2:
3637 #         (v0,h1,v1) = fitaA(0,N,gamma,0,0)
3638 #
3639 #         #print(l)
3640 #         #L = l
3641 #
3642 #         #ldos1 = []
3643 #         #ldos1 = []
3644 #         #ldos2 = []
3645 #         #ldos3 = []
3646 #         #ldosn = []
3647 #         #ldosadc = []
3648 #         L = 20
3649 #         d = 0
3650 #         cond = []
3651 #         for en in energy:
3652 #             print(en)
3653 #             e = en+1j*eta
3654 #             gl,gr = G1(h1,v0,v1,e,er)
3655 #             #for ka in range(1):
3656 #                 b = len(h1[0])
3657 #                 E1 = np.identity(b,dtype=np.complex_)
3658 #                 E1 = np.dot(e,E1)
3659 #
3660 #                 hl = HL(L,N,h1,v0,v1)
3661 #                 #print(len(hl))
3662 #                 hl = dremov(hl,v0,d)
3663 #                 #print(len(hl))
3664 #                 hef = Reduce(N,L,E1,hl)#Hd(2,N,E1)
3665 #                 #print(len(hef))
3666 #
3667 #                 a1 = len(hl)
```

```

3668 #         n = len(g1)
3669 #         m = len(hef)
3670 #         #z2 = np.zeros((n,n),complex)
3671 #         #zq1 = np.zeros((m-n,m-n),complex)
3672 #         #z1 = np.zeros((n,m-n),complex)
3673 #         ze = np.zeros((a1,a1),dtype=np.complex_)
3674 #
3675 #         z1 = np.zeros((a1,m),dtype=np.complex_)
3676 #         z2 = np.zeros((a1,a1),dtype=np.complex_)
3677 #         zm = np.zeros((m,m),dtype=np.complex_)
3678 #         znm = np.zeros((a1,m-n),dtype=np.complex_)
3679 #
3680 #
3681 #         sigl = np.dot(np.dot(v0,g1),v1)
3682 #         sigr = np.dot(np.dot(v1,gr),v0)
3683 #
3684 #         #sigR = np.block([[ze,ze],[ze,sigr]])
3685 #
3686 #         #sigL = np.block([[sigl,ze],[ze,ze]])
3687 #         Sig = np.block([[sigl,ze],[ze,sigr]])
3688 #
3689 #         h = hef+Sig #np.block([[E1-h1,-v1],[-v0,E1-h1]])
3690 #
3691 #         G = np.linalg.inv(np.block([[E1,ze],[ze,E1]]))-h
3692 #
3693 #
3694 #         G13 = np.zeros((a1,a1),complex)
3695 #         for i2 in range(a1):
3696 #             for j2 in range(a1):
3697 #                 G13[i2][j2] = G[i2][a1+j2]
3698 #
3699 #         Gamr = np.dot(1j,(sigr - np.transpose(np.conj(sigr))))
3700 #         Gaml = np.dot(1j,(sigl - np.transpose(np.conj(sigl))))
3701 #         GamR = np.block([[ze,ze],[ze,Gamr]])
3702 #         GamL = np.block([[Gaml,ze],[ze,ze]])
3703 #         g1 = np.dot(Gaml,G13)
3704 #         g2 = np.dot(Gamr,np.transpose(np.conj(G13)))
3705 #         g = np.dot(g1,g2)
3706 #
3707 #
3708 #         cond.append(np.trace(g).real)
3709 #         e2.append(en)
3710 #

```

```
3711 #         #plt.plot(e2,cond)
3712 #         return e2,cond
3713 #
3714 #
3715 #
3716 #
=====

3717
3718
3719 @njit(nogil=True)
3720 def cond(var): #gera corretamente condutancia x energia, fita com
                 desordem
3721
3722
3723     en = var[0]
3724     L = int(var[1])
3725     d = var[2]
3726     #u0 = var[3]
3727
3728     N = 8
3729     gamma = 3
3730     #L = 10
3731
3732
3733
3734     eta = 0.0001
3735     #ldos = []
3736     e2 = []
3737     er = 0.0001
3738
3739
3740
3741
3742     #l2 = [5]#= range(10,1000,40)
3743     #energy = np.arange(-3,3,0.03)
3744
3745
3746     #l1 = []
3747     #med = []
3748     #for l in l2:
3749         (v0,h1,v1) = fitaA(0,N,gamma,0,0)
3750
```

```

3751     #print(l)
3752     #L = l
3753
3754     #ldos1 = []
3755     #ldos1 = []
3756     #ldos2 = []
3757     #ldos3 = []
3758     #ldosn = []
3759     #ldosadc = []
3760     #L = 20
3761     #d = 0.02
3762     nu_at = 2*N*L
3763     nu = int(nu_at*d)
3764     #print(nu)
3765     #conf = []
3766     #en = energy
3767     #print(en)
3768     e = en+lj*eta
3769     gl,gr = G1(h1,v0,v1,e,er)
3770     #for ka in range(1):
3771     b = len(h1[0])
3772     E1 = np.eye(b, dtype=np.complex_)
3773     E1 = e*E1
3774     cond = []
3775     #for inte in np.arange(1000):
3776     h1 = HL(L,N,h1,v0,v1)
3777
3778
3779     #print(len(h1))
3780     h1 = removf(h1,v0,nu,L)#dremov(h1,v0,d)
3781     #print(len(h1))
3782     hef = Reduce(N,L,E1,h1)#Hd(2,N,E1)
3783     #print(len(hef))
3784
3785     a1 = len(h1)
3786     n = len(gl)
3787     m = len(hef)
3788     #z2 = np.zeros((n,n),complex)
3789     #zq1 = np.zeros((m-n,m-n),complex)
3790     #z1 = np.zeros((n,m-n),complex)
3791 #
=====

```

```

3792 #         ze = np.zeros((a1,a1),dtype=np.complex_)
3793 #
3794 #         z1 = np.zeros((a1,m),dtype=np.complex_)
3795 #         z2 = np.zeros((a1,a1),dtype=np.complex_)
3796 #         zm = np.zeros((m,m),dtype=np.complex_)
3797 #         znm = np.zeros((a1,m-n),dtype=np.complex_)
3798 #
=====
3799
3800
3801 sigl = np.dot(np.dot(v0,gl),v1)
3802 sigr = np.dot(np.dot(v1,gr),v0)
3803
3804 #sigR = np.block([[ze,ze],[ze,sigr]])
3805
3806 #sigL = np.block([[sigl,ze],[ze,ze]])
3807 #Sig = np.block([[sigl,ze],[ze,sigr]])
3808 A = len(sigl)
3809 Sig = np.zeros((2*n,2*n),dtype=np.complex_)
3810 Sig[:A,:A] = sigl
3811 #Sig[:A,A:] = vri
3812 #Sig[A,:A] = vli
3813 Sig[A:A:] = sigr
3814
3815 h = hef+Sig #np.block([[E1-h1,-v1],[-v0,E1-h1]])
3816 E2 = np.eye(m,dtype=np.complex_)
3817 E2 = e*E2
3818 #G = np.linalg.inv(np.block([[E1,ze],[ze,E1]])-h)
3819 G = np.linalg.inv(E2-h)
3820
3821
3822 G13 = np.zeros((a1,a1),dtype=np.complex_)
3823 for i2 in np.arange(a1):
3824     for j2 in np.arange(a1):
3825         G13[i2][j2] = G[i2][a1+j2]
3826
3827 #Gamr = np.dot(1j,(sigr - np.transpose(np.conj(sigr))))
3828 Gamr = 1j*(sigr - np.transpose(np.conj(sigr)))
3829 Gaml = 1j*(sigl - np.transpose(np.conj(sigl)))
3830 #Gaml = np.dot(1j,(sigl - np.transpose(np.conj(sigl))))
3831 #GamR = np.block([[ze,ze],[ze,Gamr]])
3832 #GamL = np.block([[Gaml,ze],[ze,ze]])

```

```

3833     g1 = np.dot(Gaml,G13)
3834     g2 = np.dot(Gamr,np.transpose(np.conj(G13)))
3835     g = np.dot(g1,g2)
3836
3837     con=np.trace(g).real
3838     cond.append(con)
3839     #conf.append(np.average(cond))
3840     #print(np.average(cond))
3841     e2.append(en)
3842
3843     #arq1 = open("decimal_cond_N_%d_d%.2f.data"%(N,d),"w")
3844     #for i in range(len(e2)):
3845         #    arq1.write("%.40f %.40f \n" %(e2[i],conf[i]))
3846     #arq1.close()
3847     #print(nu)
3848     #plt.plot(e2,conf)
3849     #plt.plot(e2,cond)
3850     return en,con
3851
3852
3853
3854 #
=====
3855 # @njit(nogil=True)
3856 # def cond2(en): #gera corretamente condutancia x energia, fita com
    desordem
3857 #     N = 8
3858 #     gamma = 3
3859 #     #L = 10
3860 #
3861 #
3862 #
3863 #     eta = 0.0001
3864 #     #ldos = []
3865 #     e2 = []
3866 #     er = 0.0001
3867 #
3868 #
3869 #
3870 #
3871 #     #l2 = [5]#= range(10,1000,40)
3872 #     #energy = np.arange(-3,3,0.03)

```

```

3873 #
3874 #
3875 #     #l1 = []
3876 #     #med = []
3877 #     #for l in l2:
3878 #         (v0,h1,v1) = fitaA(0,N,gamma,0,0)
3879 #
3880 #     #print(l)
3881 #     #L = l
3882 #
3883 #     #ldos1 = []
3884 #     #ldos1 = []
3885 #     #ldos2 = []
3886 #     #ldos3 = []
3887 #     #ldosn = []
3888 #     #ldosadc = []
3889 #     L = 20
3890 #     d = 0.02
3891 #     nu_at = 2*N*L
3892 #     nu = int(nu_at*d)
3893 #     #print(nu)
3894 #     conf = []
3895 #
3896 #     print(en)
3897 #     e = en+1j*eta
3898 #     gl,gr = G1(h1,v0,v1,e,er)
3899 #     #for ka in range(1):
3900 #     b = len(h1[0])
3901 #     E1 = np.eye(b, dtype=np.complex_)
3902 #     E1 = e*E1
3903 #     cond = []
3904 #     for inte in np.arange(1000):
3905 #         h1 = HL(L,N,h1,v0,v1)
3906 #         #print(len(h1))
3907 #         h1 = remov2(h1,v0,nu)#dremov(h1,v0,d)
3908 #         #print(len(h1))
3909 #         hef = Reduce(N,L,E1,h1)#Hd(2,N,E1)
3910 #         #print(len(hef))
3911 #
3912 #         a1 = len(h1)
3913 #         n = len(gl)
3914 #         m = len(hef)
3915 #         #z2 = np.zeros((n,n),complex)

```

```

3916 #         #zq1 = np.zeros((m-n,m-n),complex)
3917 #         #z1 = np.zeros((n,m-n),complex)
3918 # #
=====
3919 # #         ze = np.zeros((a1,a1),dtype=np.complex_)
3920 # #
3921 # #         z1 = np.zeros((a1,m),dtype=np.complex_)
3922 # #         z2 = np.zeros((a1,a1),dtype=np.complex_)
3923 # #         zm = np.zeros((m,m),dtype=np.complex_)
3924 # #         znm = np.zeros((a1,m-n),dtype=np.complex_)
3925 # #
=====

3926 #
3927 #
3928 #         sigl = np.dot(np.dot(v0,g1),v1)
3929 #         sigr = np.dot(np.dot(v1,gr),v0)
3930 #
3931 #         #sigR = np.block([[ze,ze],[ze,sigr]])
3932 #
3933 #         #sigL = np.block([[sigl,ze],[ze,ze]])
3934 #         #Sig = np.block([[sigl,ze],[ze,sigr]])
3935 #         A = len(sigl)
3936 #         Sig = np.zeros((2*n,2*n),dtype=np.complex_)
3937 #         Sig[:A,:A] = sigl
3938 #         #Sig[:A,A:] = vri
3939 #         #Sig[A,:A] = vli
3940 #         Sig[A,A:] = sigr
3941 #
3942 #         h = hef+Sig #np.block([[E1-h1,-v1],[-v0,E1-h1]])
3943 #         E2 = np.eye(m,dtype=np.complex_)
3944 #         E2 = e*E2
3945 #         #G = np.linalg.inv(np.block([[E1,ze],[ze,E1]])-h)
3946 #         G = np.linalg.inv(E2-h)
3947 #
3948 #
3949 #         G13 = np.zeros((a1,a1),dtype=np.complex_)
3950 #         for i2 in np.arange(a1):
3951 #             for j2 in np.arange(a1):
3952 #                 G13[i2][j2] = G[i2][a1+j2]
3953 #
3954 #         #Gamr = np.dot(1j,(sigr - np.transpose(np.conj(sigr))))

```



```

3955 #         Gamr = 1j*(sigr - np.transpose(np.conj(sigr)))
3956 #         Gaml = 1j*(sigl - np.transpose(np.conj(sigl)))
3957 #         #Gaml = np.dot(1j, (sigl - np.transpose(np.conj(sigl))))
3958 #         #GamR = np.block([[ze, ze], [ze, Gamr]])
3959 #         #Gaml = np.block([[Gaml, ze], [ze, ze]])
3960 #         g1 = np.dot(Gaml, G13)
3961 #         g2 = np.dot(Gamr, np.transpose(np.conj(G13)))
3962 #         g = np.dot(g1, g2)
3963 #
3964 #
3965 #         cond.append(np.trace(g).real)
3966 #         conf.append(np.average(cond))
3967 #         #print(np.average(cond))
3968 #         e2.append(en)
3969 #
3970 #         #arq1 = open("decimal_cond_N_%d_d%.2f.data"%(N,d),"w")
3971 #         #for i in range(len(e2)):
3972 #             # arq1.write("%.40f %.40f \n" %(e2[i],conf[i]))
3973 #         #arq1.close()
3974 #         #print(nu)
3975 #         #plt.plot(e2,conf)
3976 #         #plt.plot(e2,cond)
3977 #         return e2,conf,cond
3978 #

```

```

3979
3980
3981 def cond1(): #gera corretamente condutancia x L, fita com desordem
3982     N = 4
3983     gamma = 1
3984     #L = 10
3985
3986
3987
3988     eta = 0.0001
3989     ldos = []
3990     l2 = []
3991     er = 0.001
3992
3993
3994
3995

```

```

3996     #l2 = [5]#= range(10,1000,40)
3997
3998
3999
4000     #l1 = []
4001     #med = []
4002     #for l in l2:
4003         (v0,h1,v1) = fitaZ(0,N,gamma,0,0)
4004
4005     #print(l)
4006     #L = l
4007
4008     #ldos1 = []
4009     #ldos1 = []
4010     #ldos2 = []
4011     #ldos3 = []
4012     #ldosn = []
4013     #ldosadc = []
4014     #L = 10
4015     en = -1.1
4016     d = 0.02
4017     L = range(300,500,40)
4018     conf = []
4019
4020     for l in L:
4021         print(l)
4022         ex = 0
4023         e = en+1j*eta
4024         gl,gr = G1(h1,v0,v1,e,er)
4025         #for ka in range(1):
4026             b = len(h1[0])
4027             E1 = np.identity(b,complex)
4028             E1 = np.dot(e,E1)
4029             cond = []
4030             for inte in range(750):
4031                 hl = HL(l,N,h1,v0,v1)
4032                 #print(len(hl))
4033                 hl = dremov(hl,v0,d)
4034                 #print(len(hl))
4035                 hef = Reduce(N,l,E1,hl) #Hd(2,N,E1)
4036                 #print(len(hef))
4037
4038                 a1 = len(hl)

```

```

4039     n = len(g1)
4040     m = len(hef)
4041     #z2 = np.zeros((n,n),complex)
4042     #zq1 = np.zeros((m-n,m-n),complex)
4043     #z1 = np.zeros((n,m-n),complex)
4044     ze = np.zeros((a1,a1),complex)
4045
4046     #z1 = np.zeros((a1,m),complex)
4047     #z2 = np.zeros((a1,a1),complex)
4048     #zm = np.zeros((m,m),complex)
4049     #znm = np.zeros((a1,m-n),complex)
4050
4051
4052     sigl = np.dot(np.dot(v0,g1),v1)
4053     sigr = np.dot(np.dot(v1,gr),v0)
4054
4055     #sigR = np.block([[ze,ze],[ze,sigr]])
4056
4057     #sigL = np.block([[sigl,ze],[ze,ze]])
4058     Sig = np.block([[sigl,ze],[ze,sigr]])
4059
4060     h = hef+Sig #np.block([[E1-h1,-v1],[-v0,E1-h1]])
4061
4062     G = np.linalg.inv(np.block([[E1,ze],[ze,E1]])-h)
4063
4064
4065     G13 = np.zeros((a1,a1),complex)
4066     for i2 in range(a1):
4067         for j2 in range(a1):
4068             G13[i2][j2] = G[i2][a1+j2]
4069
4070     Gamr = np.dot(1j,(sigr - np.transpose(np.conj(sigr))))
4071     Gaml = np.dot(1j,(sigl - np.transpose(np.conj(sigl))))
4072     GamR = np.block([[ze,ze],[ze,Gamr]])
4073     GamL = np.block([[Gaml,ze],[ze,ze]])
4074     g1 = np.dot(Gaml,G13)
4075     g2 = np.dot(Gamr,np.transpose(np.conj(G13)))
4076     g = np.dot(g1,g2)
4077
4078     con = np.trace(g).real
4079     if con != 0.0:
4080         cond.append(np.log(con))

```

```
4081         arqc = open("dist_dec_cond_N_%d_d%.2f_L_%d.dat"%(N,d,l)
, "a")
4082         arqc.write("%.40f \n" %(np.log(con)))
4083         arqc.close()
4084     else:
4085         ex =ex + 1
4086         conf.append(np.average(cond))
4087         #print(np.average(cond))
4088         l2.append(1)
4089         print(l,np.average(cond),ex)
4090
4091     arq1 = open("decimal_condxL_N_%d_d%.2f.data"%(N,d), "w")
4092     for i in range(len(l2)):
4093         arq1.write("%.40f %.40f \n" %(l2[i],conf[i]))
4094     arq1.close()
4095
4096     arq2 = open("dist_dec_cond_N_%d_d%.2f.data"%(N,d), "w")
4097     for i in range(len(cond)):
4098         arq2.write("%.40f \n" %(cond[i]))
4099     arq2.close()
4100     #plt.plot(e2,conf)
4101     #plt.plot(e2,cond)
4102     return l2,conf,cond
```