Comparação de desempenho de implementações Fortran e C: Corbino TBG

Arthur M. Passos[1], Calebe P. Bianchini[1]

[1]Faculdade de Computação e Informática - FCI
Universidade Presbiteriana Mackenzie - São Paulo, SP - Brasil

arthurmoreirap@gmail.com, calebe.bianchini@mackenzie.br

Resumo. *Este trabalho compara versões em Fortran e C de um código para simulação de transporte eletrônico em dispositivos de Grafeno Bicamada Torcido com geometria de Corbino utilizando intensivamente a biblioteca Intel MKL para manipulação de matrizes. O objetivo da reescrita foi modernizar o código, mantendo a fidelidade física dos resultados e comparar o desempenho das versões. Testes mostraram que a versão em C reduziu em 10% o tempo de execução e 25% o uso médio de CPU em relação à versão em Fortran, com consumo de memória RAM máximo equivalente. Os resultados destacam a eficácia da modernização e abrem perspectivas para otimizações futuras com outras bibliotecas, computação heterogênea e aplicação de outras técnicas.*

1. Introdução

O Grafeno Bicamada Torcido (TBG) tem atraído considerável interesse devido à formação de bandas quase planas, que dão origem a fenômenos eletrônicos exóticos, como estados isolantes correlacionados e supercondutividade [Nimbalkar e Kim 2020]. A geometria de Corbino, ao minimizar os efeitos dos estados de borda, permite a caracterização intrínseca das propriedades de transporte eletrônico do material, fornecendo *insights* essenciais para o entendimento desses fenômenos. A crescente complexidade dos dispositivos baseados em TBG e a demanda por simulações precisas de seus fenômenos de transporte têm apresentado desafios em termos do alto uso de tempo e recursos computacionais, impulsionado o desenvolvimento de códigos computacionais cada vez mais otimizados. Assim, simular com precisão o comportamento do sistema torna-se fundamental para a investigação de novos regimes de condutividade e para a validação de modelos teóricos [Barcelos et al, 2019]. Portanto, torna-se necessário atualizar e otimizar os códigos existentes para acompanhar essas tendências tecnológicas.

O presente trabalho apresenta um primeiro passo para a modernização de um código originalmente desenvolvido em Fortran – que realiza a simulação do transporte eletrônico em dispositivos de TBG em geometria de disco de Corbino – para a linguagem de programação C. A escolha pela reescrita do algoritmo para C visa explorar os benefícios em potencial de integração com outras bibliotecas e ferramentas computacionais modernas e ser mais amplamente adotada pela comunidade de desenvolvedores de *software* de alto desempenho [Silva et al, 2022].

Neste artigo, é detalhada a metodologia da reescrita do código, explorando as principais diferenças obtidas dessa reescrita para C. Além disso, foi realizada uma análise comparativa entre as versões em Fortran e C, utilizando métricas de tempo de execução e eficiência computacional. Os resultados demonstram que a versão em C não apenas mantém a fidelidade dos resultados físicos, mas também apresenta ganhos de

desempenho, o que a torna uma opção robusta e preferível para a simulação de sistemas TBG em configurações de disco de Corbino.

2. Base Teórica

2.1. Grafeno Bicamada Torcido (TBG)

O Grafeno Bicamada Torcido (TBG) representa um sistema de grande interesse na física da matéria condensada devido à peculiaridade de suas bandas eletrônicas. A rotação relativa entre as duas camadas de grafeno leva à formação de um padrão moiré que resulta no surgimento de bandas quase planas em ângulos de torção específicos, como o ângulo mágico. Essas bandas planas são responsáveis por uma variedade de fenômenos eletrônicos exóticos, incluindo estados isolantes correlacionados e supercondutividade. A investigação desses estados e a compreensão dos mecanismos de transporte em TBG demandam simulações precisas que considerem a complexidade das supercélulas de moiré, que podem atingir tamanhos significativos, tornando as simulações em larga escala um desafio computacional

2.2. Corbino

A geometria de Corbino é empregada como uma estratégia experimental para mitigar ou eliminar a contribuição dos estados de borda nas medições de transporte, permitindo o acesso às propriedades intrínsecas do interior (*bulk*) do material [Yan e Fuhrer, 2010]. Em dispositivos com essa configuração, as correntes elétricas fluem em um caminho anular entre um contato interno e um externo, eliminando os efeitos das bordas laterais presentes em outras geometrias. A simulação precisa do transporte em dispositivos de TBG com geometria de Corbino é, portanto, fundamental para a caracterização das propriedades fundamentais do material, a exploração de novos regimes de condutividade e a validação de modelos teóricos que buscam explicar os fenômenos observados. Métodos computacionais eficientes, como o framework "CAP-Chebyshev" que combina a teoria da aproximação de Chebyshev com um potencial de absorção complexo (CAP) para contabilizar contatos semi-infinitos, são desenvolvidos para lidar com as demandas computacionais dessas simulações em larga escala.

3. Metodologia

3.1. Procedimento de Reescrita do Código

A modernização do código para simulações de transporte eletrônico em dispositivos de TBG foi conduzida a partir de uma base originalmente escrita em Fortran, com forte uso da biblioteca Intel Math Kernel Library (MKL) para operações numéricas de alto desempenho. O código original em Fortran realiza operações fundamentais, como:

- Adição de matrizes esparsas em formato *Compressed Sparse Row* (CSR), crucial para o acoplamento eficiente entre diferentes regiões do sistema físico modelado;
- Multiplicação de matrizes densas e esparsas, utilizadas em etapas intermediárias da propagação de estados eletrônicos;
- Resolução de sistemas lineares esparsos via o *solver* Intel PARallel DIrect SOlver *(*PARDISO), essencial para o cálculo da função de Green resolvente com eficiência e estabilidade numérica.

Na implementação em linguagem C, foram utilizadas as mesmas funcionalidades da biblioteca MKL, garantindo a equivalência dos resultados físicos e numéricos simulados pelos dois códigos. No entanto, a transição entre as linguagens exigiu diversas adaptações estruturais, tanto na representação dos dados quanto nas convenções de indexação e armazenamento adotadas. As principais adaptações foram:

- Adaptação de tipos de dados: o tipo `Integer*8` amplamente utilizado em Fortran foi substituído por `long long int` em C, garantindo compatibilidade com os requisitos da MKL em arquiteturas modernas e suporte a índices de grande porte. Da mesma forma, o tipo `Complex*16` foi substituído por `MKL_Complex16`, garantindo a correta manipulação de números complexos nas operações vetoriais e matriciais.
- Indexação base-0 e base-1: Fortran utiliza indexação base-1 por padrão, enquanto C adota indexação base-0. Para preservar a integridade dos dados e o correto funcionamento das rotinas MKL, foi necessário ajustar todos os índices de estruturas de dados (como os vetores de índices de linha e coluna em estruturas CSR) para refletirem a nova base de indexação.
- Conversão de esquemas de armazenamento (*column-major* e *row-major*): Fortran armazena vetores em ordem *column-major*, enquanto C utiliza a ordem *row-major*. Para garantir que as operações envolvendo matrizes resultassem nos mesmos resultados, foi necessário adaptar o preenchimento e leitura das matrizes densas durante a multiplicação e em etapas de pré-processamento.

Essas adaptações foram realizadas de forma sistemática, com validação da consistência dos resultados entre as duas versões do código. Os resultados das simulações físicas, em ponto flutuante, foram validados e a margem de erro absoluta são da ordem de 10^{-12}.

3.2. Ambiente dos testes

Os experimentos foram realizados em uma máquina com CPU Intel Core i5-1235U, com 10 núcleos físicos e 12 processadores lógicos; com 32 GB de RAM, alocando 26GB para o WSL 2. O sistema operacional utilizado é o Windows 11 com o subsistema linux WSL Ubuntu 22.04.4 LTS. Foi usado o compilador C gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0 e compilador Intel® Fortran Compiler ifx (IFX) 2025.0.4 20241205. Em ambas as versões foi utilizada a flag O3 de nível de otimização.

Foram realizadas 10 execuções utilizando o programa linux `/usr/bin/time` para perfilamento.

4. Resultados

A média dos resultados está disposta na Tabela 1.

Tabela 1. Média das métricas das execuções

	user time	system time	elapsed time	cpu percent	max resident size
Fortran	21304s	365s	4682s	462%	776381 KB
C	14469s	241s	4222s	348%	790174 KB
Diferença	-32%	-37%	-10%	-25%	+1,7%

As métricas representam respectivamente: acumulado dos tempos em modo usuário por CPU, acumulado dos tempos em modo tempo em modo Kernel por CPU, tempo total, uso percentual médio de CPU, memória RAM máxima consumida.

A diminuição no percentual médio de uso da CPU (-25%) sugere que a versão em C, além de mais rápida no geral (-10% no tempo decorrido), foi computacionalmente mais eficiente, possivelmente devido a otimizações realizadas pelo compilador GCC ou a uma melhor interação do código C com a MKL e o sistema operacional em comparação com a versão Fortran compilada com IFX neste ambiente específico. Como não há diretivas de paralelismo explícitas, o uso de múltiplos núcleos se deve principalmente a chamadas da MKL em ambos os códigos.

Outro ponto de destaque é que a quantidade máxima de memória RAM utilizada pelas duas versões se manteve próxima. Reforçando a semelhança do nível de acesso e controle da memória das duas linguagens, fazendo com que os mesmos dados ocupem aproximadamente o mesmo espaço em memória sem sobrecargas consideráveis.

5. Conclusão

A modernização do código para C demonstrou-se eficaz ao reduzir o uso dos recursos computacionais e tempo de execução, preservando a fidelidade dos resultados físicos. Esses resultados incentivam a adoção de códigos modernos e otimizados para a simulação de sistemas complexos como os dispositivos de Grafeno Bicamada Torcido em geometria de Corbino, proporcionando uma ferramenta robusta e eficiente para a pesquisa em física da matéria condensada.

Trabalhos futuros incluem explorar a aplicação de hardwares especializados com computação heterogênea, como GPUs, e técnicas de otimização da manipulação de memória. Além disso, há espaço para comparar o desempenho da MKL com outras bibliotecas numéricas (como OpenBLAS, BLIS, cuSPARSE/cuSOLVER para GPUs) e avaliar diferentes *solvers* para sistemas lineares esparsos.

6. Referências

Barcelos, I. D.; Bechtel, H. A.; de Matos, C. J. S.; Bahamon, D. A.; Kaestner, B.; Maia, F. C. B.; Freitas, R. O. (2019). Probing Polaritons in 2D Materials with Synchrotron Infrared Nanospectroscopy. Advanced Optical Materials, volume 8.

Nimbalkar, A.; Kim, H. (2020). Opportunities and Challenges in Twisted Bilayer Graphene: A Review. Nano-Micro Letters.

Silva, G. P., Bianchini, C. P., & Costa, E. B. (2022). Programação Paralela e Distribuída: Com MPI, OpenMP e OpenACC para Computação de Alto Desempenho. Casa do Código.

Yan, J.; Fuhrer, M (2010). Charge Transport in Dual Gated Bilayer Graphene with Corbino Geometry. Nano Letters, v. 10, n. 11, p. 4521–4525.