# Aplicação *multithread* para a aceleração do método *Particle-in-Cell (PIC)*

**Maurício Ferreira de Araújo**[1], **Lui Habl**[1], **Daniel Sundfeld**[1]

[1]Faculdade de Ciências e Tecnologias em Engenharias – Universidade de Brasília (UnB)
Campus Gama, Setor Leste - 72.444-240 - Gama - DF - Brasil

ferreira.mauricio@aluno.unb.br, {lui.habl,daniel.sundfeld}@unb.br

***Resumo.*** *A aceleração de partículas baseada em plasma envolve um meio não linear e esse tipo de modelagem requer ferramentas de simulação adequadas. Particle-in-Cell (PIC) é uma importante técnica utilizada para se estudar a física dos plasmas. No entanto, essa simulação pode precisar de muito tempo e é desejável acelerar o resultado. Neste trabalho, propomos criar um simulador PIC para arquiteturas multicore, aproveitando o paralelismo entre vários núcleos e usando instruções SIMD. Nossos experimentos mostram que é possível otimizar a simulação em até 1,28x usando um computador portátil de 6 núcleos.*

## 1. Introdução

A simulação numérica de plasmas pode ser realizada por diferentes abordagens, como os modelos fluidos (baseados nas equações de magnetohidrodinâmica), os métodos híbridos e os modelos cinéticos de partículas. Entre estes, o método *Particle-in-Cell* (PIC) destaca-se por capturar com maior fidelidade os fenômenos microscópicos e as interações entre partículas e campos eletromagnéticos.

Apesar de sua precisão, o método PIC apresenta alto custo computacional, especialmente em simulações de larga escala, o que motiva a investigação de estratégias de paralelização e otimização de desempenho. Neste trabalho, é proposta a implementação de uma simulação PIC em C++, com posterior aceleração utilizando OpenMP, de modo a avaliar o impacto do paralelismo no tempo de execução e na escalabilidade do método.

## 2. *Particle-in-Cell* (PIC) Paralelo

O método PIC, estabelecido desde os trabalhos pioneiros de [Dawson 1962] e evoluindo ao longo das décadas [Taccogna et al. 2023, Birdsall and Langdon 1991], descreve a interação entre partículas e campos. Em sua forma discreta, essa interação é descrita pelas equações a seguir:

A abordagem PIC, em sua forma discreta, é matematicamente descrito pela interação entre a densidade das partículas, o potencial e o campo elétrico resultante:

A interpolação da densidade (Equação 1) é dada por:

$$n_j += \frac{x_{j+1} - r_i}{\Delta x} \quad \text{e} \quad n_{j+1} += \frac{r_i - x_j}{\Delta x} \tag{1}$$

O potencial elétrico (Equação 2) é obtido a partir da densidade usando uma aproximação de diferenças finitas para a Equação de Poisson:

$$\frac{d^2\phi(x_j)}{dx^2} \approx \frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(\Delta x)^2} \tag{2}$$

A Equação 3 apresenta o campo elétrico é calculado como o gradiente negativo do potencial, também por diferenças finitas:

$$\frac{d\phi(x_i)}{dx} \approx \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} \tag{3}$$

Por fim, a Equação 4 representa a interpolação do campo elétrico para a posição de uma única partícula:

$$E_i = \frac{x_{j+1} - r_i}{\Delta x} E_j + \frac{r_i - x_j}{\Delta x} E_{j+1} \tag{4}$$

A paralelização via *Single Instruction, Multiple Data* (SIMD) foi aplicada nas equações centrais do modelo 1D [Mocz 2020], especificamente na equação (1), correspondente à interpolação da densidade numérica de partículas na malha, e na equação (4), referente à interpolação do campo elétrico. Para as equações 2 (cálculo do potencial elétrico) e 3 (determinação do campo elétrico), foi utilizada uma biblioteca sequencial para a resolução de sistemas lineares com matrizes esparsas.

Para avaliar o ganho de desempenho da paralelização, as funções principais do modelo 1D foram otimizadas com a API OpenMP, utilizando diretivas de paralelismo de dados SIMD [Tanenbaum and Van Steen 2017]. Essa técnica é ideal pois permite que uma única instrução opere sobre múltiplos elementos dos vetores (índice i) de uma só vez, justificando-se pela natureza das equações que quando discretizadas iteram sobre eles. Assim, a divisão das iterações do loop entre múltiplas *threads* torna-se altamente eficiente.

## 3. Resultados Experimentais

Os testes foram realizados em um computador portátil equipado com um processador AMD Ryzen 5 5500U, com 6 núcleos (12 *threads*), 8 GB de memória RAM, sistema operacional Fedora 42 com o compilador GCC versão 15.2.1. Para executar a simulação, foi selecionado um conjunto $1.000.000$ de partículas em uma malha de $500$ posições. A Tabela 1 ilustra o tempo utilizado e resultados obtidos com o uso da paralelização de dados, mostrando que o algoritmo possui um ganho de desempenho em comparação à sua versão serial.

É possível perceber que o tempo de execução diminui consideravelmente em relação ao aumento de *threads*, ou seja, quanto mais *threads* for possível utilizar, menor o tempo gasto para execução. Dessa forma, alcançou-se uma melhoria de aproximadamente $28\%$ no desempenho.

A Figura 1 ilustra um *frame* da execução da simulação desenvolvida. O espaço de fase é plotado em um gráfico no qual o eixo $x$ é a posição da partícula, e o eixo $y$, por sua vez, a velocidade. As partículas em azul iniciaram a simulação com velocidade positiva, e as vermelhas, com velocidade negativa. A distribuição observada confirma o comportamento esperado do modelo PIC, indicando a correta evolução das partículas sob a interação malha-partícula durante a simulação paralela.

**Tabela 1. Resultados de Desempenho da Simulação de Partículas, Variando o Número de *Threads* ($N$).**

| Nº de *Threads* ($N$) | Tempo ($s$) | *Speedup* |
|---|---|---|
| 1 | 6,34 | 1,00 |
| 2 | 5,86 | 1,05 |
| 4 | 5,78 | 1,09 |
| 8 | 5,30 | 1,19 |
| 12 | 4,93 | 1,28 |

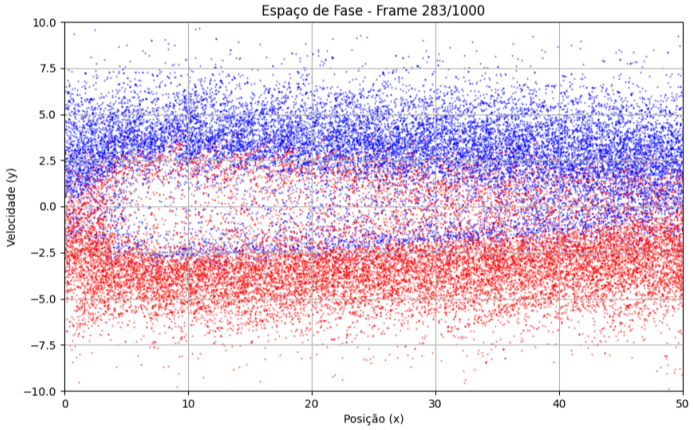


**Figura 1. Execução da simulação PIC paralela**

## 4. Conclusão e Trabalhos Futuros

Neste artigo, propusemos uma versão paralela para a simulação de plasma utilizando a abordagem *Particle-in-Cell* (PIC). Os resultados mostram que o tempo de execução pode ser reduzido de 6,34s para 4,93s utilizando 12 *threads*. Para trabalhos futuros, esperamos ampliar o número de simulações, buscando diminuir a duração da execução e, possivelmente, melhorar o *speedup*. Posteriormente, também planeja-se a criação de uma versão massivamente paralela utilizando a linguagem de programação CUDA para unidades de processamento gráfico (GPUs).

## Referências


Birdsall, C. K. and Langdon, A. B. (1991). *Plasma Physics via Computer Simulation*. McGraw-Hill, New York.

Dawson, J. M. (1962). One-dimensional plasma model. *Physical Review*, 128(2):622–633.

Mocz, P. (2020). Create your own plasma (pic) simulation with python. `https://medium.com/swlh/create-your-own-plasma-pic-simulation-with-python-39145c66578b`. Acesso em: 27 out. 2025.

Taccogna, F., Cichocki, F., Eremin, D., Fubiani, G., and Garrigues, L. (2023). Plasma propulsion modeling with particle-based algorithms. *Journal of Applied Physics*, 134(15):150901.

Tanenbaum, A. S. and Van Steen, M. (2017). *Sistemas Distribuídos: Princípios e Paradigmas*. Pearson, São Paulo, 3 edition.