Universidade Federal do Rio Grande do Sul
Instituto de Física

Rodrigo Weber Pereira

Simulação Computacional em Física de Plasmas

Porto Alegre
2017

Rodrigo Weber Pereira

Simulação Computacional em Física de Plasmas

Trabalho de Conclusão de Curso apresentado
ao Instituto de Física da Universidade Federal
do Rio Grande do Sul como requisito parcial
para a obtenção do grau de bacharel em Física.

Universidade Federal do Rio Grande do Sul
Instituto de Física

Orientador: Prof. Dr. Fernando Haas

Porto Alegre
2017

Sumário

Resumo

Neste trabalho desenvolvem-se técnicas analíticas para a simulação computacional de plasmas não colisionais, visando o entendimento do seu comportamento coletivo, com o desenvolvimento do algoritmo ES1. Através da aplicação da transformada contínua de Fourier nas equações de Maxwell é possível conectar a densidade de carga com a energia potencial no espaço de frequências através de uma simples relação algébrica. Podemos definir que o plasma está contido em um domínio discretizado, onde todas as quantidades espaciais, exceto a posição das partículas, são calculadas apenas em um número finito de pontos da grade. Isso permite que se obtenha numericamente não só a transformada discreta de Fourier (DFT, do inglês Discrete Fourier Transform) destas quantidades, mas também uma relação entre as transformadas contínua e discreta, possibilitando uma maneira de obter o campo elétrico em cada ponto do espaço através de técnicas de interpolação de primeira ordem desenvolvidas no trabalho. Desenvolve-se também, para fins de eficiência computacional, a técnica conhecida como FFT (Fast Fourier Transform) que permite reduções significativas no tempo de simulação. Como alternativa ao método por transformadas, desenvolve-se uma maneira de resolver numericamente as equações de Maxwell que governam o sistema, através de um método matricial que envolve a solução de um sistema linear, providas as condições de contorno.

Abstract

In this work, analytical techniques for a computational simulation of a non-collisional plasma are developed, aiming the understanding of its collective behaviour, with the development of the ES1 algorithm. By applying the continuous Fourier transform in the Maxwell equations, it is possible to connect the charge density with the potential energy in the frequency domain through a simple algebraic relation. We can define that the plasma is contained in a discrete domain, where all the spatial quantities, except the position of the particles, are calculated only in a finite number of grid points. This allows one to obtain numerically not only a Discrete Fourier Transform (DFT) but also a relation between continuous and discrete transformations, enabling a way to obtain the electric field at each point of space through first order interpolation techniques developed in this work. A technique known as Fast Fourier Transform (FFT) is also developed for computational efficiency, which allows significant reductions on the simulation elapsing time. As an alternative to the Fourier method, a numerical way of solving the Maxwell's equations that govern the system is developed through the use of a matrix method, involving a solution of a linear system, given the boundary conditions.

1 Introdução

A falta de textos introdutórios é um fator de risco para qualquer campo de estudo na física hoje. Particularmente na física de plasmas, sente-se a necessidade, advinda da falta de experiência dos estudantes neste tema, de uma maior interlocução da teoria com a prática. Infelizmente, principalmente devido a razões de ordem financeira, são poucos os centros de educação que dispõem de laboratórios ou até mesmo grupos de pesquisa na área de física de plasmas, de forma que muitos estudantes tem pouco ou nenhum contato com plasmas durante toda a sua vida acadêmica. Embora sendo a simulação computacional um trabalho teórico, a possibilidade de resolver a evolução temporal de um sistema complexo por meio destas simulações pode muitas vezes fornecer uma intuição ao estudante, estando esta, muitas vezes, intimamente relacionada com a *prática*. A ideia por trás deste trabalho é o desenvolvimento de um algoritmo amplamente difundido chamado ES1[1], usado na simulação computacional de plasmas eletrostáticos unidimensionais. A solução de problemas envolvendo física de plasma através de recursos computacionais é uma realidade recente, não porque os computadores em si se desenvolveram há pouco tempo, mas porque as técnicas matemáticas usadas na solução destes problemas não eram computacionalmente eficientes, isto é, nem com o melhor computador disponível seria possível obter a solução do problema mais simples de física de plasma em um tempo razoável. Isso acontece por que todos os problemas da física de plasma envolvem muitas partículas, aumentando muito o trabalho computacional. O *turning point* ocorreu pela década de 60, quando uma técnica chamada Transformada de Fourier Rápida causou uma verdadeira revolução neste aspecto. Esta técnica, embora não única, é a base para o programa ES1, e será desenvolvida em detalhes neste trabalho. Com ela, as simulações hoje não se restringem a computadores de alta capacidade: os códigos envolvidos podem ser rodados em um computador de mesa por um estudante de graduação ou no âmbito de pequenos grupos de pesquisa nas universidades sem acesso a caros recursos computacionais como *clusters*.

Apesar da natureza unidimensional dos problemas que o algoritmo ES1 se dispõe a resolver, é possível extrair resultados úteis para a física de plasmas, sendo o seu correto entendimento um caminho natural para o desenvolvimentos de algoritmos mais complexos, em duas e até mesmo três dimensões, onde a introdução de campos magnéticos induzidos e externos faz sentido. A ideia por trás de toda a simulação é a obtenção de uma visão am-

[1] Electrostatic one dimensional 1D code.

pla sobre a dinâmica do problema em questão. O programa em si armazena todas as quantidades de utilidade física, como as posições, velocidades, densidade de carga, potencial e campo elétrico em todos os pontos do plasma, podendo o programador escolher quais destes parâmetros serão armazenados para uma posterior análise. Os primeiros resultados de utilidade prática em uma simulação são os qualitativos: a análise qualitativa do resultado da simulação, através da análise de gráficos relevantes para o problema em questão, dá o *insight* necessário para a investigação aprofundada (quantitativa) do fenômeno em questão, de forma que o programador pode selecionar regiões de interesse, nas quais o mar de dados gerado pelo programa mostrará sua utilidade. É impossível escrever um programa otimizado para a análise de todos os problemas físicos, pois a quantidade de dados necessária seria infinita. Em vez disso, para cada problema específico que o programador se dispõe a resolver, existirá um conjunto de parâmetros e quantidades úteis a serem armazenadas pelo programa, de forma que sua análise posterior seja possível. A simulação computacional se apresenta não só como uma ferramenta poderosa na análise dos problemas que a Física de Plasmas se dispõe a resolver, que em sua maioria não possuem soluções analíticas, como também é um instrumento pedagógico útil na compreensão fenomenológica desta grande área.

Na próxima seção estabelecemos os objetivos deste trabalho de maneira mais precisa, apontando as vantagens e limitações de se trabalhar com o algoritmo ES1. Na seção 3 abordamos os fundamentos físicos e as técnicas matemáticas para o desenvolvimento do programa. O enfoque é no entendimento dos conceitos necessários para escrever o algoritmo e busca-se caminhos alternativos para se obter os mesmos resultados através do uso de técnicas que exigem mais ou menos poder computacional. Ao final desta seção elucidamos algumas informações que podem ser tiradas diretamente do programa, independente do tipo de plasma em estudo. O trabalho é encerrado com a conclusão na seção 4.

2 Objetivos

O objetivo do programa ES1 é simular o comportamento coletivo de plasmas não colisionais ($N_D \gg 1$ e $L \gg \lambda_D$)[2] em comprimentos de onda λ maiores que o comprimento de onda de Debye λ_D, ou seja, $\lambda > \lambda_D$, com

$$\lambda_D = \sqrt{\frac{\epsilon_0 k_B T}{ne^2}}. \tag{2.1}$$

Nessa expressão, $\epsilon_0 \approx 8,85.10^{-12} \frac{C^2}{Nm^2}$ é a constante de permissividade do vácuo, $k_B \approx 1,38.10^{-23} \frac{J}{K}$ é a constante de Boltzmann, n é a densidade numérica de partículas do plasma, $e \approx 1,6.10^{-19} C$ é a carga fundamental do elétron e T é a temperatura do plasma. Usualmente encontram-se, na bibliografia, plasmas de densidades $n = 10^6/cm^3$ a $n = 10^{22}/cm^3$, com energias variando de $0.01eV$ a $10^5 eV$. Considerando que o número de partículas dentro do cubo de Debye é dada por

$$N_D = n\lambda_D^3, \tag{2.2}$$

podemos obter, por exemplo, para a ionosfera da terra, $N_D \approx 10^4$. Para experimentos de fusão de plasmas confinados observa-se até mesmo $N_D \approx 10^6$! Isso significa que o número de partículas envolvidas na simulação pode ser tão grande a ponto de ser proibitivo. O algoritmo ES1, no entanto, se propõe a reproduzir a essência do comportamento do plasma, sem maiores detalhes, tornando a simulação possível. O plasma que iremos tratar é unidimensional, logo não precisamos satisfazer o requerimento tridimensional, digamos $N_D = 10^6$, mas apenas o unidimensional $N_D \approx (10^6)^{\frac{1}{3}} = 10^2$, reduzindo, em muito, o esforço computacional.

O algoritmo ES1 é um conjunto de instruções lógicas, que pode ser adaptado às necessidades ou limitações do computador em que será rodada a simulação do plasma em questão, por isso a necessidade de se desenvolver múltiplos caminhos para se conseguir o mesmo resultado. Isso é feito ao longo do trabalho, onde o usuário do programa pode decidir qual método será usado no cálculo da densidade de carga $\rho(x,t)$, campo elétrico $E(x,t)$ e na integração das equações de movimento. Para cada caso são dadas duas alternativas, ou possibilidades de solução. As técnicas serão desenvolvidas de forma linear, sendo apresentadas à medida em que forem sendo necessárias na solução do problema.

O foco do trabalho é se tornar um instrumento útil ao programador, na hora

[2] Neste trabalho, L é o comprimento do sistema unidimensional e N_D é o número de partículas dentro do cubo de Debye.

não só de escrever o programa, mas também de compreender todas as etapas e ferramentas matemáticas que compõem o programa.

3 O Programa ES1

O algoritmo ES1 que vamos estudar é uma forma de resolver problemas típicos da física de plasmas para o caso unidimensional eletrostático, ou seja, despreza-se a presença de campos magnéticos $\mathbf{B}(\mathbf{r}, t)$ induzidos pelas partículas, e considera-se apenas a dimensão espacial x. Abaixo, vamos elucidar em linhas gerais o fluxo de funcionamento do programa e, a seguir, detalhamos cada uma das etapas de forma a construir o conhecimento necessário para escrever um algoritmo capaz de rodar o programa em si em qualquer linguagem de programação.

3.1 O Ciclo Computacional: Comentários Gerais

O ponto de partida para a solução do problema eletrostático unidimensional é a discretização da região em consideração. Divide-se a região unidimensional $(0, L)$, onde L é o comprimento do sistema, em intervalos de comprimento Δx. Desta forma, cria-se uma espécie de "grade matemática"com N pontos, onde todas as quantidades relevantes do problema serão calculadas *apenas* nos pontos desta grade. Cada um dos N pontos da grade define uma *célula* de comprimento Δx, com o ponto da grade no centro. A figura 1 elucida como seria a construção de uma grade para o caso bidimensional. No programa ES1 a grade é evidentemente unidimensional, com pontos da forma $X_0, X_1, ..., X_{N-1}$. As posições das partículas podem assumir qualquer valor no intervalo $(0, L)$, mas a densidade de carga $\rho(x, t)$, campo elétrico $E(x, t)$ e força $F(x, t)$ serão conhecidos *apenas* nos pontos da grade. As diferentes espécies de partículas carregadas do plasma estão distribuídas inicialmente dentro da grade, com velocidades iniciais dadas. O programa precisa saber exatamente o que o usuário gostaria de ter inicialmente no sistema, por isso, um dos dados de entrada obrigatórios é a distribuição $f(x, v, t = 0)$, para cada espécie, fornecida pelo programador. Geralmente, no entanto, se conhece a densidade de carga, ou a densidade de partículas do sistema; por isso desenvolve-se mais a frente um método genérico para obter as posições e velocidades, a partir de uma perturbação arbitrária na distribuição de carga inicial dada.
Com as posições das partículas definidas inicialmente é possível, *a priori*, calcular a densidade de carga local em cada um dos pontos da grade no tempo $t = 0$:

$$\rho(X_0, 0) \quad , \quad \rho(X_1, 0) \quad \dots \quad , \quad \rho(X_{N-1}, 0). \tag{3.1}$$

Isso será feito tanto com um método de ordem zero chamado *NGP*, quanto com um método de primeira ordem chamado *CIC*, a ser explicado mais à

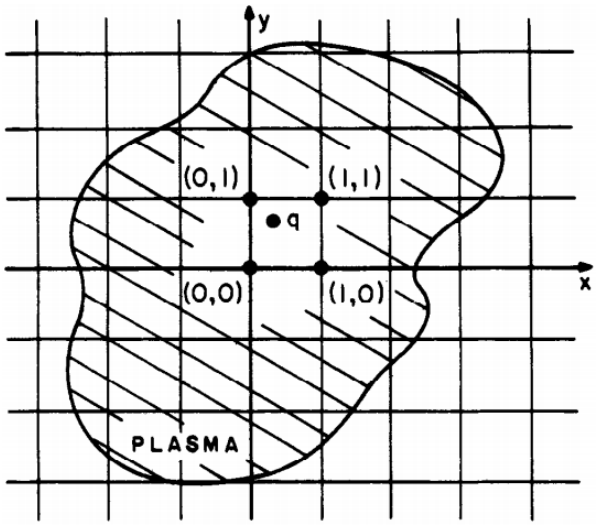

Figura 1: A grade matemática é definida na região do plasma para que a densidade $\rho(x, y, t)$ possa ser medida. Cada carga q localizada no interior de uma célula irá contribuir para a densidade de carga das respectivas células vizinhas. Fonte: Ref. [1], p. 10.

frente. A escolha do método a ser usado vai depender do custo computacional que o programador está disposto a pagar. De posse da densidade de carga inicial, usa-se as equações de Maxwell conjuntamente com uma técnica envolvendo a Transformada de Fourier Rápida, para obter o campo elétrico em cada um dos pontos da grade também no tempo $t = 0$:

$$E(X_0, 0) \quad , \quad E(X_1, 0) \ \ldots \quad , \quad E(X_{N-1}, 0). \tag{3.2}$$

O programador também pode optar por usar um método matricial, que usa as condições de contorno do problema para obter os mesmo resultados para o campo elétrico[3]. Aqui, vemos o quão importante é a grade em si: as quantidades até agora citadas são conhecidas apenas nos pontos da grade, não estando definidas fora dela! Este é o regime de aproximação do programa ES1, que obviamente se torna melhor à medida em que os pontos da grade se aproximam uns dos outros. No entanto, para usar o campo elétrico obtido nas equações de movimento (segunda lei de Newton), é preciso conhecê-lo na

[3]Embora igualmente precisa, esta técnica mostra-se computacionalmente mais dispendiosa que o método das transformadas.

posição x_i da $i-$ésima partícula, ou seja, é preciso conhecer as quantidades

$$E(x_i, 0) \qquad i = 0, ..., N_P, \tag{3.3}$$

onde N_P é o número de partículas. Para obter $E(x_i, 0)$ a partir dos campos $E(X_j, 0)$ da grade, usa-se um método de interpolação (mostrado mais à frente) que leva em consideração os campos em células vizinhas ao ponto x_i de interesse. De posse do campo elétrico na posição da partícula, obtém-se a força $F(x_i, 0)$ simplesmente multiplicando pela carga q, o que nos permite usar as equações de movimento para prever as posições x_i e velocidades v_i das partículas em um tempo futuro $t + \Delta t$. Aqui opta-se por um método numérico de segunda ordem chamado *Leapfrog*, em contraste com o método de Euler usual, que é de primeira ordem. Se desejado, o programador pode incluir a aplicação de um campo elétrico externo E_0 na direção x, de forma que a força sobre a partícula no passo anterior seria:

$$F(x_i, 0) = q[E(x_i, 0) + E_0].$$

Estes passos encerram o primeiro ciclo computacional. Com as novas posições e velocidades das partículas é possível recalcular a densidade de carga com os métodos *NGP* ou *CIC*, depois os campos, a força e finalmente as novas posições, repetindo estes passos quantas vezes for desejado pelo programador. A figura 2 apresenta um esquema que resume o ciclo até aqui descrito.

3.2 Obtendo $\rho(x, t)$ na Grade

Vamos apresentar duas maneiras de computar a densidade de carga $\rho(X, t)$ nos pontos da grade. A escolha de qual delas é a mais adequada deve ser avaliada pelo programador, com base na disponibilidade de tempo e capacidade computacional.

3.2.1 Método *NGP*

A sigla *NGP* significa *Nearest Grid Point* e é um método muito simples para calcular a densidade de carga nos pontos da grade. Simplesmente contamos o número de partículas de cada espécie que se encontram dentro de uma distância de $\pm\frac{\Delta x}{2}$ em torno do $j-$ésimo ponto da grade (um comprimento de célula) e designamos este número à variável $N(j)$. Desta forma, em uma dimensão e para apenas uma espécie de partículas com carga q, a densidade de partículas neste ponto da grade é simplesmente

$$n(X_j, t) = \frac{N(j)}{\Delta x}, \tag{3.4}$$

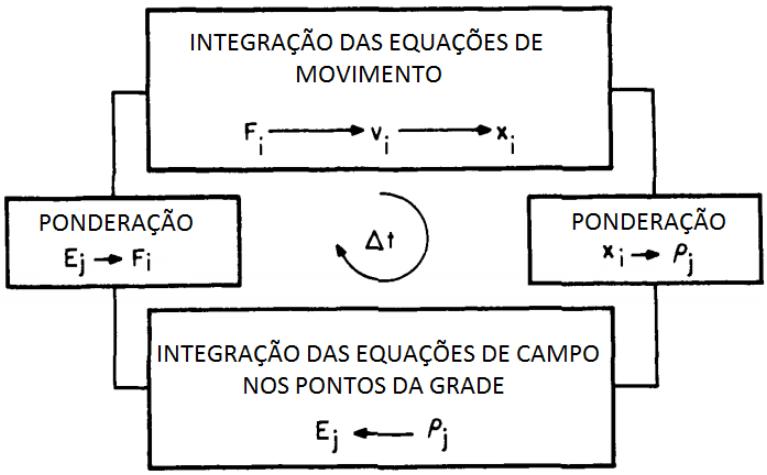

Figura 2: Um ciclo típico num passo de tempo Δt. As partículas são representadas pelo índice $i = 1, .., N_P$. Os pontos da grade pelo índice $j = 0, 1, ..., N - 1$. Fonte: Ref. [1], p. 11.

podendo esta ser transformada em uma densidade de carga $\rho(X_j, t)$ multiplicando pela carga da partícula:

$$\rho(X_j, t) = qn(X_j, t) = \frac{qN(j)}{\Delta x}. \tag{3.5}$$

Incluir mais espécies no plasma em estudo não é problema algum, basta contar a densidade de cada espécie separadamente e somar, com atenção ao sinal da carga. Como mencionado anteriormente, o método *NGP* é um método de ordem zero. Computacionalmente a contagem de partículas em cada célula é rápida, pois é exigido apenas uma olhada na posição de cada partícula para saber onde ela se encontra e qual é a sua carga, para assim designar um $N(j)$ em cada célula.

Dois efeitos podem ser notados com o uso desta técnica. Primeiro, vemos que a figura 3 evidencia o fato de que este método dá um formato retangular à partícula, como se ela tivesse um comprimento Δx. Ou seja, as partículas deixam de ser pontuais para ter um tamanho finito Δx. Uma vez que colisões (ou *encontros próximos*) entre partículas são raros[4], este efeito quase não altera os efeitos básicos da física de plasma a serem estudados. Segundo, nota-se que, quando uma partícula x_i entra ou sai das fronteiras $X_j \pm \frac{\Delta x}{2}$ da célula j, a densidade de carga aumenta ou diminui bruscamente, causando

[4]Quase todas as colisões ocorrem com grande parâmetro de impacto.

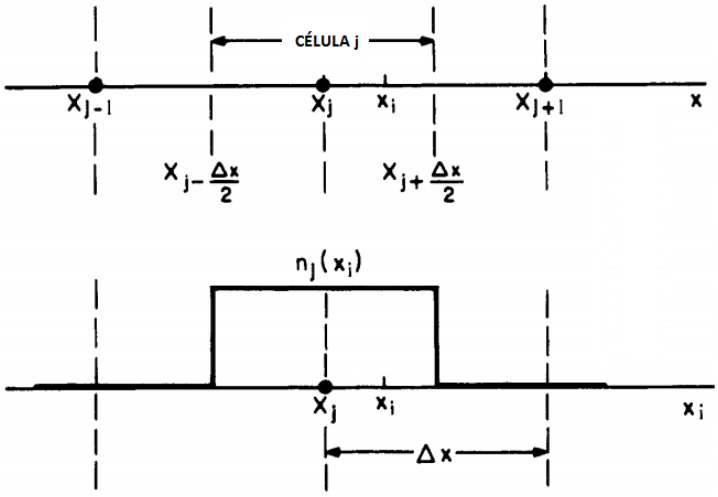

Figura 3: Método *NGP* de ponderação. A densidade resultante pode ser interpretada como um "formato efetivo"da partícula. Fonte: Ref. [1], p. 20.

uma espécie de ruído, tanto no espaço quanto no tempo, que podem ser uma grande fonte de erro nesta aproximação.

3.2.2 Método *CIC*

A sigla *CIC* significa *Cloud in Cell*. Este método suaviza a densidade de partículas em comparação com o *NGP*, às custas de mais tempo computacional, sendo considerado um método de primeira ordem. Aqui, adota-se um modelo no qual as partículas assumem o formato de "nuvens", com tamanho finito, que podem passar livremente umas através das outras. Isso implica que estamos desprezando qualquer tipo efeito colisional, mas para uma primeira aproximação a hipótese é válida. Esse trabalho não tem intenção de incluir um termo colisional nas suas equações, já que este é um tratamento usualmente reservado para sistemas tridimensionais. Deseja-se estender a contribuição à densidade de carga devido a uma partícula na célula j para as partículas vizinhas. Isso pode ser feito se assumirmos que a partícula é uma nuvem de carga com densidade uniforme e comprimento Δx, como pode ser visto na figura 4. Isto é, se a contribuição de carga à célula for proporcional à fração da nuvem que está sobre a célula, então é fácil ver que de toda a carga q da nuvem, a fração $q(j)$ de carga designada à célula j é apenas

$$q(j) = q\Big[\frac{\Delta x - (x_i - X_j)}{\Delta x}\Big] = q\Big[\frac{X_{j+1} - x_i}{\Delta x}\Big], \tag{3.6}$$

onde no último passo foi usada a relação $X_{j+1} = \Delta x + X_j$. Já a fração $q(j+1)$ de carga designada à célula $j+1$ é o que resta de carga:

$$q(j+1) = q - q(j) = q\Big[\frac{x_i - X_j}{\Delta x}\Big]. \tag{3.7}$$

Considere o que acontece quando uma partícula atravessa a fronteira de duas células vizinhas. O simples exame das equações 3.6 e 3.7 mostra que se a partícula está sobre a célula j ($x_i = X_j$), então tem-se:

$$q(j) = q \qquad q(j+1) = 0.$$

Quando a partícula está entre duas células $x_i = X_j + \frac{\Delta x}{2}$ (na fronteira das duas), então:

$$q(j) = \frac{q}{2} \qquad q(j+1) = \frac{q}{2}.$$

E finalmente, se ela passa para a posição $x_i = X_{j+1}$ tem-se:

$$q(j) = 0 \qquad q(j+1) = q.$$

Ou seja, este método suaviza a contribuição de carga à medida em que as partículas entram ou abandonam as células, evitando o ruído presente no método *NGP*. O maior custo computacional vem do fato de que é preciso acessar mais de um ponto de grade para cada partícula, duas vezes por passo: a carga total de uma célula j será a soma de todos os $q(j)$ das partículas vizinhas a esta célula, considerando as cargas com sinal de espécie presente.

3.3 Obtendo $E(x,t)$ na Grade

Uma vez definida a densidade de carga em todos os pontos da grade é possível obter o campo elétrico usando apenas a primeira equação de Maxwell, a saber:

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \qquad \text{(Lei de Gauss)}. \tag{3.8}$$

Da teoria eletromagnética, sabemos que o campo elétrico pode ser obtido da relação

$$\mathbf{E} = -\nabla\phi - \frac{\partial \mathbf{A}}{\partial t}, \tag{3.9}$$

onde $\phi = \phi(\mathbf{r}, t)$ é o potencial elétrico e $\mathbf{A} = \mathbf{A}(\mathbf{r}, t)$ é o potencial vetor, valendo a relação $\mathbf{B} = \nabla \times \mathbf{A}$. Estamos lidando com um problema eletrostático

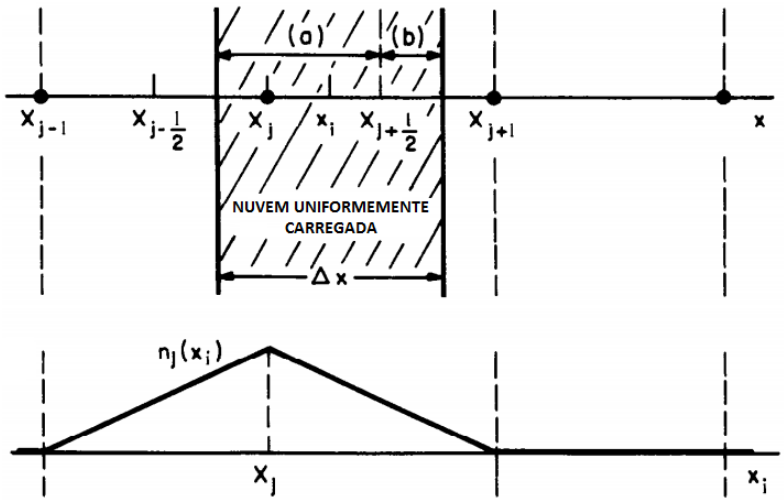

Figura 4: Método *CIC* de ponderação. A parte (a) da nuvem contribui para a fração $q(j)$, enquanto a parte (b) da nuvel contribui para a fração $q(j+1)$. O resultado é uma maior suavização da densidade de carga. Fonte: Ref. [1], p. 21.

com campo elétrico irrotacional da forma $\mathbf{E} = E(x,t)\hat{x}$, implicando que sempre existe $\phi(x,t)$ tal que a equação 3.9 pode ser escrita simplesmente como

$$\mathbf{E} = -\nabla\phi, \tag{3.10}$$

respeitadas as condições de regularidade usuais. De fato, o potencial escalar vem de $E = -\frac{\partial\phi}{\partial x}$. Considerando que não há campo magnético, isto permite escolher um potencial vetor $\mathbf{A} = 0$. As equações 3.8 e 3.10, para o nosso problema unidimensional, podem ser escritas como:

$$\frac{\partial E_x}{\partial x} = \frac{\rho}{\epsilon_0}, \tag{3.11}$$

$$E_x = -\frac{\partial\phi}{\partial x}. \tag{3.12}$$

Substituindo uma equação na outra obtém-se a Equação de Poisson unidimensional:

$$\frac{\partial^2\phi}{\partial x^2} = -\frac{\rho}{\epsilon_0}. \tag{3.13}$$

A partir daqui vamos explorar dois métodos para resolver as equações acima. Um deles usa álgebra linear para resolver um sistema linear nas variáveis ϕ_i, dado as condições de contorno, e o outro usa uma técnica recente chamada Transformada Rápida de Fourier, ou FFT[5].

[5]Do inglês *Fast Fourier Transform*.

3.3.1 Solução Matricial

É possível obter o campo elétrico E_x através das equações 3.12 e 3.13, providas as condições de contorno

$$\phi_0 = \phi(0) = V_0 \qquad \phi_{N-1} = \phi(L) = V_L,$$

conforme vamos explicitar agora. Primeiro, notamos pela figura 5 que a equação 3.12 pode ser escrita como

$$E_x = -\frac{\partial \phi}{\partial x} \implies E_j = -\Big[\frac{\phi_{j+1} - \phi_{j-1}}{2\Delta x}\Big], \tag{3.14}$$

onde $E_j = (E_x)_j$, ou seja, a componente x do campo elétrico no $j-$ésimo ponto da grade. Aqui simplesmente aplicamos a definição de derivada usando os pontos do domínio disponíveis. Note que optamos por uma definição de derivada "centralizada", diferentemente do que é feito no método de Euler[6]. A equação de Poisson (3.13) envolve uma derivada segunda, e pode ser calculada da seguinte forma:

$$\frac{\partial^2 \phi}{\partial x^2} = -\frac{\rho}{\epsilon_0} \implies \phi''_j = -\frac{\rho_j}{\epsilon_0} = \frac{\phi'_{j+\frac{1}{2}} - \phi'_{j-\frac{1}{2}}}{\Delta x} = \frac{\frac{\phi_{j+1}-\phi_j}{\Delta x} - \left(\frac{\phi_j - \phi_{j-1}}{\Delta x}\right)}{\Delta x},$$

que resulta em:

$$\frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(\Delta x)^2} = -\frac{\rho_j}{\epsilon_0}. \tag{3.15}$$

Para simplificar a notação, defina:

$$p_j \equiv -\frac{\rho_j (\Delta x)^2}{\epsilon_0}.$$

Escrevendo a equação 3.15 para $j = 1, 2, \ldots, N-2$ obtém-se:

$$\begin{aligned}
j = 1 \qquad & \underbrace{\phi_0}_{V_0} - 2\phi_1 + \phi_2 = p_1 \\
j = 2 \qquad & \phi_1 - 2\phi_2 + \phi_3 = p_2 \\
\vdots \qquad & \qquad\qquad \vdots \\
j = N-2 \qquad & \phi_{N-3} - 2\phi_{N-2} + \underbrace{\phi_{N-1}}_{V_L} = p_{N-2}.
\end{aligned}$$

[6] Neste caso a definição seria: $E_j = -\Big[\frac{\phi_{j+1}-\phi_j}{\Delta x}\Big]$.

As relações acima podem ser escrítas na forma matricial:

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \ddots & \vdots \\ 0 & & \cdots & & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N-3} \\ \phi_{N-2} \end{pmatrix} = \begin{pmatrix} p_1 - V_0 \\ p_2 \\ \vdots \\ p_{N-3} \\ p_{N-2} - V_L \end{pmatrix} \tag{3.16}$$

Sendo que os $p_j^{'}$s são conhecidos, temos $N-2$ equações independentes e $N-2$ variáveis, o que implica que o sistema é, em princípio, solúvel. Repare que ϕ_0 e ϕ_{N-1} já são conhecidos (condições de contorno). Aqui é possível usar uma variedade de métodos para resolver o sistema, cabendo ao programador decidir qual será usado.

De posse do potencial elétrico em todos os pontos da grade, usa-se a equação 3.14 em cada ponto para calcular o campo elétrico E_j em todos os pontos da grade.

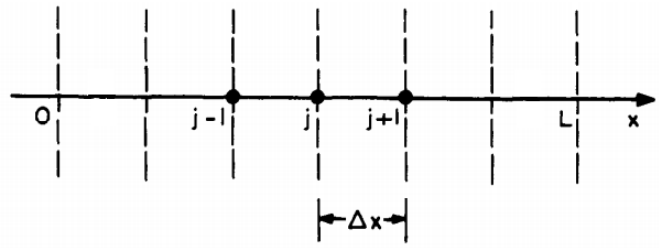

Figura 5: Grade unidimensional com os pontos uniformemente espaçados localizados em $X_j = j\Delta x$. Fonte: Ref. [1], p. 16.

3.3.2 Solução por Transformada Rápida de Fourier - FFT

Uma maneira muito mais prática e rápida de achar o campo elétrico E_x é o uso do algoritmo que iremos descrever agora. A moderna FFT foi desenvolvida por volta dos anos 1960[7] e revolucionou a área de processamento de sinais, solução de equações diferenciais parciais e diversos outros ramos da matemática aplicada. No programa ES1 ela é a parte principal, sendo, na verdade, o fator que possibilita as simulações em física de plasmas em

[7]Descobriu-se posteriormente que o matemático Gauss, em 1805, já havia inventado este método quando ele precisava interpolar a órbita dos asteroides *Pallas* e *Juno* a partir de dados observacionais.

computadores de mesa, e por isso será dada toda a atenção necessária ao desenvolvimento da teoria.

A abordagem para a obtenção do campo elétrico é diferente da que foi feita na seção anterior: a ideia chave para a solução do problema usando a FFT é assumir que $\rho(x)$, $\phi(x)$ e $E_x(x) = E(x)$ possuem transformada de Fourier[8] da forma

$$\rho(k) = \mathcal{F}\{\rho(x)\}$$
$$\phi(k) = \mathcal{F}\{\phi(x)\}$$
$$E(k) = \mathcal{F}\{E(x)\},$$

onde optamos por usar o núcleo e^{-ikx} da transformada de Fourier contínua:

$$f(k) = \mathcal{F}\{f(x)\} = \int_{-\infty}^{\infty} f(x)e^{-ikx}dx. \tag{3.17}$$

Nessa equação, k é o vetor de onda associado a este núcleo da transformada. Ao assumir que as funções tem transformada de Fourier estamos também assumindo que elas são periódicas, mas isso não causa problema, como vamos mostrar. O primeiro a se fazer é aplicar a transformada de Fourier nos dois lados da equação de Poisson:

$$\frac{\partial^2\phi}{\partial x^2} = -\frac{\rho}{\epsilon_0} \implies \mathcal{F}\Big\{\frac{\partial^2\phi}{\partial x^2}\Big\} = \mathcal{F}\Big\{-\frac{\rho}{\epsilon_0}\Big\}.$$

O termo do lado direito da igualdade é imediatamente obtido:

$$\mathcal{F}\Big\{-\frac{\rho}{\epsilon_0}\Big\} = \int_{-\infty}^{\infty}\Big(-\frac{\rho(x)}{\epsilon_0}\Big)e^{-ikx}dx = -\frac{1}{\epsilon_0}\int_{-\infty}^{\infty}\rho(x)e^{-ikx}dx = -\frac{\rho(k)}{\epsilon_0}.$$

Já o termo do lado esquerdo se obtém através de integração por partes duas vezes seguidas:

$$\mathcal{F}\Big\{\frac{\partial^2\phi}{\partial x^2}\Big\} = \int_{-\infty}^{\infty}\Big(\frac{\partial^2\phi}{\partial x^2}\Big)e^{-ikx}dx = \underbrace{\frac{\partial\phi}{\partial x}e^{-ikx}\Big|_{-\infty}^{\infty}}_{=0} + ik\int_{-\infty}^{\infty}\frac{\partial\phi}{\partial x}e^{-ikx}dx$$

$$= ik\Big[\underbrace{\phi e^{-ikx}\Big|_{-\infty}^{\infty}}_{=0} + ik\int_{-\infty}^{\infty}\phi e^{-ikx}dx\Big] = -k^2\mathcal{F}\{\phi(x)\} = -k^2\phi(k).$$

[8] Não é sempre necessário assumir que $E(x)$ possui transformada de Fourier, como vamos explicar mais a frente.

Os termos identicamente nulos assim o são pois assume-se que

$$\phi(\infty) = \phi(+\infty) = 0$$

$$\frac{\partial\phi}{\partial x}\Big|_{x=\infty} = \frac{\partial\phi}{\partial x}\Big|_{x=-\infty} = 0.$$

Juntando os resultados, obtemos:

$$\phi(k) = -\frac{\rho(k)}{\epsilon_0 k^2}. \tag{3.18}$$

Esta equação representa uma conexão direta entre a densidade de carga e o potencial elétrico no $k-$espaço. A fim de explorar esta relação extremamente simples entre estas duas quantidades, temos que perceber o seguinte: assumimos desde o começo que existe um número N de pontos na grade nos quais $\rho(X_j)$ para $j = 0, 1, \ldots, N-1$ são conhecidos, ou seja, temos uma coleção discreta e finita de pontos, ao invés de uma distribuição contínua com infinitos pontos. Isso nos permite aplicar uma Transformada de Fourier Discreta (DFT[9]) com o *mesmo núcleo* da transformada contínua sobre esta coleção de pontos. Para exemplificar, considere a nossa densidade de carga $\rho(x)$ como sendo uma função periódica $\rho(x) = \rho(x + L)$; isso não representará nenhuma dificuldade como vamos ver à frente. Temos na verdade uma coleção de pontos da forma $\rho(X_j)$. Por definição, a Transformada de Fourier Discreta deste conjunto de pontos seria

$$\rho(k) = \Delta x \sum_{j=0}^{N-1} \rho(X_j) e^{-ikX_j}, \tag{3.19}$$

onde $X_j = j\Delta x$. A sua transformada inversa é dada por (a soma é em $k = n\frac{2\pi}{L}$)

$$\rho(X_j) = \frac{1}{L} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \rho(k) e^{ikX_j}. \tag{3.20}$$

A condição de periodicidade é fundamental pois é ela quem determina os possíveis valores que o vetor de onda k pode assumir, veja:

$$\rho(x) = \rho(x + L) \;\Rightarrow\; \rho(X_j) = \rho(X_j + L).$$

Sendo a função periódica, tem que valer:

[9] Do inglês *Discrete Fourier Transform*.

$$\Delta x \sum_{j=0}^{N-1} \rho(X_j) e^{-ikX_j} = \Delta x \sum_{j=0}^{N-1} \underbrace{\rho(X_j + L)}_{\rho(X_j)} e^{-ik(X_j+L)}$$

$$\Delta x \sum_{j=0}^{N-1} \rho(X_j) e^{-ikX_j} = e^{-ikL} \Delta x \sum_{j=0}^{N-1} \rho(X_j) e^{-ikX_j}$$

$$e^{-ikL} = 1 \implies \cos(kL) - i\sin(kL) = 1 \Leftrightarrow kL = 2\pi n \Rightarrow k = n\frac{2\pi}{L}$$

A condição de periodicidade, essencial para o correto funcionamento do programa, será implementada ao elucidarmos o funcionamento do algoritmo de Leapfrog. A ideia central agora é conectar $\rho(k)$ da equação 3.18 com o $\rho(k)$ da equação 3.19, ou seja, conectar a transformada de Fourier contínua com a discreta, já que o valor de k nas duas deve ser o mesmo.

A Transformada Rápida de Fourier (FFT) entra no algoritmo neste momento. Note que para calcular a Transformada de Fourier Discreta (DFT) dada pela equação 3.19 para apenas um k, digamos k_1, é necessário multiplicar N termos da forma $\rho(X_j)$ com $\exp(-i\frac{2\pi X_j}{L}); j = 0, \ldots, N-1$, e em seguida somar N termos da forma $\rho(X_j)e^{-i\frac{2\pi X_j}{L}}; j = 0, \ldots, N-1$ entre eles, totalizando $2N$ operações matemáticas. Desejamos obter não apenas $\rho(k_1)$, mas também $\rho(k_2), \rho(k_3), ..., \rho(k_N)$ etc! Logo, repete-se o processo por mais $N-1$ vezes, totalizando $2N \times N$ operações. Ou seja, a quantidade de operações matemáticas (envolvendo soma e multiplicação de números complexos) é da ordem de

$$\mathcal{O}(N^2).$$

Vamos mostrar agora que através da FFT é possível fatorar o somatório presente na definição da DFT (Eq. 3.19), e através do uso de propriedades especiais do núcleo da transformada de Fourier, reduzir a ordem de grandeza das operações matemáticas necessárias para apenas:

$$\mathcal{O}(N \log_2 N).$$

O ganho em termos de eficiência computacional pode parecer, à primeira vista, pequeno. Mas considere valores crescentes de N, conforme a tabela 1. De fato, para $N = 1000$ a FFT representa uma economia de apenas algumas dezenas de milhares de operações. No entanto, conforme se vê pela tabela, para $N = 10^6$ a DFT faz 5×10^4 vezes o número de operações do que a técnica FFT! Para expressar esta diferença em termos palpáveis, considere que um computador realiza cada operação em $1ns$. Então para $N = 10^9$,

Tabela 1: Comparação entre a quantidade de computação necessária para o cálculo da DFT com e sem a FFT.

N	1000	10^6	10^9
N^2	10^6	10^{12}	10^{18}
$N \log_2 N$	10^4	20×10^6	30×10^9

o tempo computacional envolvido para calcular a DFT seria de $31,2$ anos! Em contraste, se usarmos o algoritmo da FFT, podemos reduzir este tempo de computação para apenas 30 segundos. Motivados por esta constatação, vamos explicar o procedimento para a construção do algoritmo e mostrar que de fato precisa-se de aproximadamente $N \log_2 N$ operações para realizar a DFT.

A FFT explora as propriedades da exponencial e^{-ikx}. A partir daqui, vamos assumir que $N = 2^m$, onde m é um inteiro qualquer. A necessidade desta restrição ficará evidente em seguida. Para descarregar a notação, defina

$$e^{-ikX_j} = e^{-ikj\Delta x} = \Big[\underbrace{e^{-i\Delta x}}_{W_{\Delta x}}\Big]^{kj} \equiv W_{\Delta x}^{kj}.$$

Nesta notação, a DFT como a definimos pode ser escrita como

$$\rho(k) = \Delta x \sum_{j=0}^{N-1} \rho(X_j) W_{\Delta x}^{kj}.$$

Separando a soma nos elementos ímpares e pares temos:

$$\rho(k) = \Delta x \Big[\sum_{j \text{ PAR}} \rho(X_j) W_{\Delta x}^{kj} + \sum_{j \text{ ÍMPAR}} \rho(X_j) W_{\Delta x}^{kj} \Big].$$

Tomando $j = 2r$ para os elementos pares e $j = 2r + 1$ para os ímpares fica:

$$\rho(k) = \Delta x \Big[\sum_{r=0}^{\frac{N}{2}-1} \rho(X_{2r}) W_{\Delta x}^{k(2r)} + \sum_{r=0}^{\frac{N}{2}-1} \rho(X_{2r+1}) W_{\Delta x}^{k(2r+1)} \Big].$$

Colocando o termo $W_{\Delta x}^{k}$ para fora da soma, e notando que $W_{\Delta x}^{2kr} = W_{2\Delta x}^{kr}$, temos:

$$\rho(k) = \Delta x \Big[\sum_{r=0}^{\frac{N}{2}-1} \rho(X_{2r}) W_{2\Delta x}^{kr} + W_{\Delta x}^{k} \sum_{r=0}^{\frac{N}{2}-1} \rho(X_{2r+1}) W_{2\Delta x}^{kr} \Big].$$

Perceba que, se definirmos

$$\rho_P(k) = \sum_{r=0}^{\frac{N}{2}-1} \rho(X_{2r}) W_{2\Delta x}^{kr} \tag{3.21}$$

$$\rho_I(k) = \sum_{r=0}^{\frac{N}{2}-1} \rho(X_{2r+1}) W_{2\Delta x}^{kr}, \tag{3.22}$$

então $\rho(k)$ pode ser escrito como:

$$\rho(k) = \Delta x \Big[\rho_P(k) + W_{\Delta x}^{k} \rho_I(k) \Big]. \tag{3.23}$$

Ou seja, expressamos $\rho(k)$ como uma soma de duas outras DFTs. Note que $\rho_P(k)$ é uma DFT com $\frac{N}{2}$ pontos associada aos elementos pares de $\rho(X_j)$, enquanto $\rho_I(k)$ é uma DFT com $\frac{N}{2}$ pontos associada aos elementos ímpares de $\rho(X_j)$. A exigência de que tenhamos apenas $N = 2^m$ vem da necessidade da divisão $\frac{N}{2}$ (aparece no somatório), que só pode ser feita resultando um número inteiro para todo N se o mesmo for um número par. Considere a figura 6, que esquematiza o processo para $N = 2^3 = 8$. A partir da

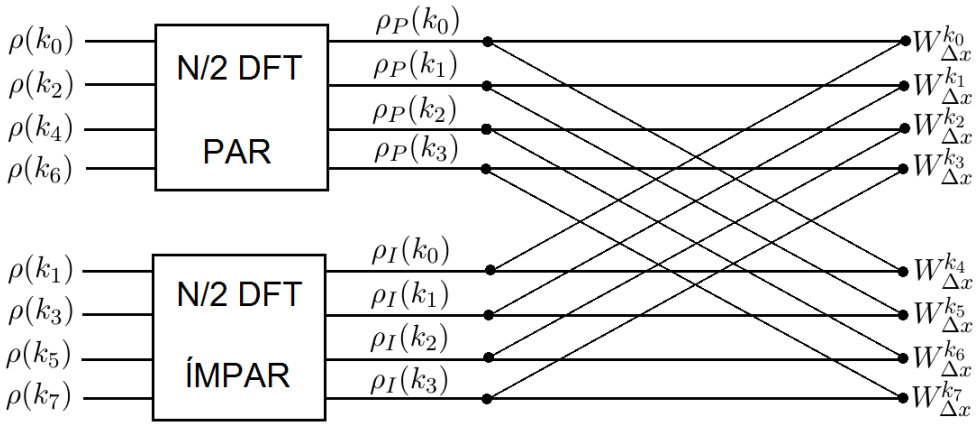

Figura 6: Representa o ciclo de cálculo de $\rho(k)$ para $N = 8$.

figura 6 é possível visualizar e fazer a contagem do custo computacional envolvido neste processo: cada DFT com $\frac{N}{2}$ elementos consome, como visto, aproximadamente $\left(\frac{N}{2}\right)^2$ operações. Temos duas DFTs, logo o número de operações é $2 \times \left(\frac{N}{2}\right)^2$. Como temos que somar o termo $W_{\Delta x}^{k}$ ao final do

cálculo de cada DFT, soma-se ainda N operações. Logo, tem-se um total de aproximadamente

$$\frac{N^2}{2} + N.$$

Este resultado mostra que uma única aplicação desta fatoração já corta o número de operações aproximadamente pela metade! Esta divisão em duas DFTs já nos economizou algum tempo computacional, mas é possível ir mais além. Podemos aplicar a mesma fatoração às DFTs $\rho_P(k)$ e $\rho_I(k)$, e reduzir mais ainda o tempo computacional, continuando este processo até a fatoração máxima, ou seja:

$$\frac{N}{2}\ ,\ \frac{N}{4}\ ,\ \cdots\ ,\ \frac{N}{2^{p-1}}\ ,\ \frac{N}{2^p} = 1.$$

Vemos, portanto, que $p = \log_2 N$ é o número máximo de vezes que podemos dividir a nossa $\rho(k)$ original. Se contarmos o custo computacional frente às sucessivas divisões, obtemos o número de operações necessárias para o cálculo, conforme mostrado na Tabela 2.

Tabela 2: Esquema explicitando o número de operações necessárias para o cálculo de uma DFT, com N pontos, em função da fatoração da transofrmada.

Fatoração	Nº Ptos. na DFT	Nº de Operações Necessárias
1	$\frac{N}{2}$	$2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N$
2	$\frac{N}{4}$	$2\left(2\left(\frac{N}{4}\right)^2 + \frac{N}{2}\right) + N = \frac{N^2}{4} + 2N$
3	$\frac{N}{8}$	$2\left[2\left(2\left(\frac{N}{8}\right)^2 + \frac{N}{4}\right) + \frac{N}{2}\right] + N = \frac{N^2}{8} + 3N$
$\vdots$	$\vdots$	$\vdots$
p	$\frac{N}{2^p} = 1$	$\frac{N^2}{2^p} + Np = \frac{N^2}{N} + N\log_2 N = N + N\log_2 N$

Para a fatoração máxima p, vê-se que a ordem de grandeza do número de operações é

$$\mathcal{O}(N \log_2 N),$$

conforme tínhamos afirmado. Repare que a exigência $N = 2^m$ foi essencial para chegarmos na fatoração máxima.

3.3.3 Interpolação do Campo Elétrico

Uma vez obtido o campo elétrico $E(X_j)$ em todos os pontos da grade, temos que usar esta informação para calcular o campo elétrico $E(x_i)$ no ponto em que a $i-$ésima partícula se encontra. A maneira mais fácil de fazer isso seria considerar o campo elétrico da célula em que a partícula se encontra como sendo o campo elétrico na posição da partícula, o que funcionaria bem no limite $N \to \infty$. Este método é considerado como de ordem zero, e conduz aos mesmos tipos de complicações já discutidas no método *NGP*. Convencionalmente, adota-se um método de interpolação inteiramente análogo ao que foi feito na obtenção da densidade de carga $\rho(X_j)$ pelo método *CIC*, ou seja, se faz uma média ponderada dos campos elétricos experimentados pela partícula na posição $x_i \in (X_j, X_{j+1})$ pela relação:

$$E(x_i) = \Big[\frac{X_{j+1} - x_i}{\Delta x}\Big] E(X_j) + \Big[\frac{x_i - X_j}{\Delta x}\Big] E(X_{j+1}). \tag{3.24}$$

Esta técnica é também de primeira ordem e, como já discutido, suaviza as contribuições com diferentes valores nos pontos da grade.

3.4 Integração das Equações de Movimento

Conhecendo as posições e velocidades iniciais de todas as partículas do sistema (informadas pelo programador), podemos calcular a densidade de carga $\rho(x, t = 0)$ e o campo elétrico $E(x, t = 0)$ através das técnicas descritas anteriormente. Isso nos permite usar as equações de movimento

$$m\frac{\mathrm{d}v}{\mathrm{d}t} = F \tag{3.25}$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v \tag{3.26}$$

para prever as posições e velocidades de cada uma delas em um tempo posterior $t = 0 + \Delta t$ com boa aproximação, desde que Δt seja pequeno. Isso pode ser feito por diversos métodos de integração numérica, destacando-se os Métodos de Euler (1^{a} Ordem), Leapfrog (2^{a} Odem) e Runge-Kutta (4^{a} Ordem). A diferença entre eles está no custo computacional envolvido: queremos escolher um método simples e direto, que exija o mínimo de esforço computacional mas que capture as propriedades essenciais do sistema. O

Método de Runge-Kutta é o preferido na grande maioria das simulações numéricas envolvendo poucas partículas, mas no nosso caso estamos lidando com um sistema de muitas partículas, e o uso deste método anularia o ganho computacional já obtido com a FFT. Por isso, vamos elucidar apenas os dois primeiros métodos.

Sobre o intervalo Δt a ser usado no programa, o programador deve naturalmente ter cuidado na sua escolha pois este deve ser compatível com a frequência de oscilação ω do plasma estudado. Por outro lado, um intervalo de tempo muito pequeno pode comprometer a eficiência da simulação, aumentando em muito o tempo necessário para rodar o código.

3.4.1 Método de Euler

O método de Euler é a maneira mais simples de resolver as equações de movimento. Simplesmente avançamos as quantidades conhecidas pelo programa num tempo arbitrário t

$$x(t) = x_t \quad v(t) = v_t \quad F(t) = F_t$$

para um tempo posterior $t + \Delta t$

$$x_{t+\Delta t} \quad v_{t+\Delta t} \quad F_{t+\Delta t}$$

através das duas equações de *diferença finita*, vindas diretamente das equações 3.25 e 3.26:

$$m\frac{v_{t+\Delta t} - v_t}{\Delta t} = F_t \quad \Longrightarrow \quad v_{t+\Delta t} = v_t + F_t\frac{\Delta t}{m} \tag{3.27}$$

$$\frac{x_{t+\Delta t} - x_t}{\Delta t} = v_t \quad \Longrightarrow \quad x_{t+\Delta t} = x_t + v_t\Delta t. \tag{3.28}$$

As duas equações acima determinam as posições e velocidades de todas as partículas no tempo posterior, já que todas as quantidades no tempo anterior são conhecidas *a priori*. Com os novos valores de posição e velocidade é possível recalcular a densidade de carga e o campo elétrico através dos métodos estudados e usar as equações 3.27 e 3.28, novamente substituindo as quantidades novas:

$$x_{t+\Delta t} \to x_t \quad v_{t+\Delta t} \to v_t \quad F_{t+\Delta t} \to F_t.$$

A cada iteração, incrementa-se o tempo por uma quantidade Δt e, repetindo este processo, chegamos até o tempo de simulação T_{sim} desejado.

3.4.2 Método de Leapfrog

O método de integração Leapfrog resolve as equações de movimento de forma muito semelhante ao método de Euler. A diferença é que as duas equações de diferença finita, provindas das equações 3.25 e 3.26, são escritas na forma

$$m\frac{v_{\text{novo}} - v_{\text{velho}}}{\Delta t} = F_{\text{velho}} \tag{3.29}$$

$$\frac{x_{\text{novo}} - x_{\text{velho}}}{\Delta t} = v_{\text{novo}}. \tag{3.30}$$

As equações acima são diferentes das equações de diferença finita do método de Euler, pois se evidencia que o tempo está centralizado, ou seja, não se conhece a velocidade e as posições e força no mesmo instante de tempo t, como podemos ver esquematicamente na figura 7. O computador avança x_t

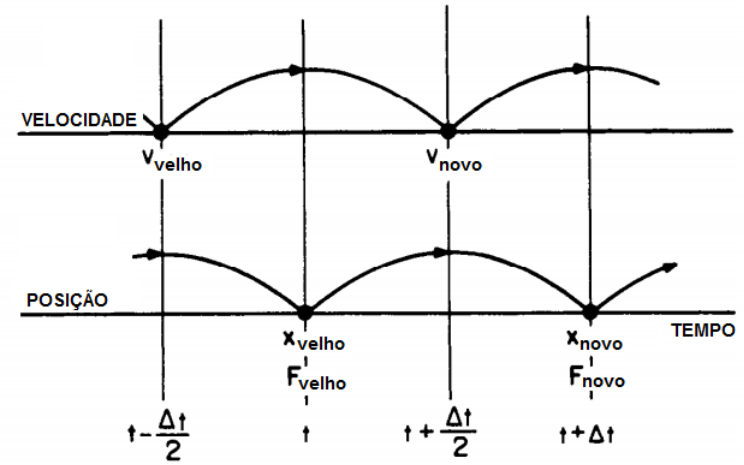

Figura 7: Esquema do método de integração Leapfrog mostrando a centralização temporal da força F e posição x durante o avanço da velocidade v. Fonte: Ref. [1], p. 13.

e v_t para $x_{t+\Delta t}$ e $v_{t+\Delta t}$, mesmo que x e v não sejam conhecidos ao mesmo tempo. Sendo que as condições iniciais para as posições e velocidades das partículas são fornecidas ao programa no instante $t = 0$, a fim de que o método funcione corretamente, é preciso alterar estas condições iniciais para poder encaixá-las no fluxo de cálculo do programa. As posições iniciais das partículas já estão adequadas para o uso, pois $x(0) = x_{\text{velho}}$; a força sobre as partículas também, pois $F(0) = F_{\text{velho}}$. A velocidade $v(0)$ da condição

inicial deve ser trazida de volta no tempo até $v(0-\frac{\Delta t}{2}) = v_{\text{velho}}$ com o uso da equação 3.29:

$$m\frac{v(0)-v_{\text{velho}}}{\left(-\frac{\Delta t}{2}\right)} = F_{\text{velho}} \qquad \Longrightarrow \qquad v_{\text{velho}} = v(0) + \frac{F_{\text{velho}}\Delta t}{2m}.$$

Usando v_{velho} na mesma equação, podemos obter $v(0+\frac{\Delta t}{2}) = v_{\text{novo}}$:

$$m\frac{v_{\text{novo}}-v_{\text{velho}}}{\Delta t} = F_{\text{velho}} \qquad \Longrightarrow \qquad v_{\text{novo}} = v_{\text{velho}} + \frac{F_{\text{velho}}\Delta t}{m}.$$

Repare que este passo é necessário apenas uma única vez, só para deslocar temporalmente a velocidade a fim de encaixá-la no fluxo de cálculo do método. Feito isso, usamos as equações de diferença finita para avançar temporalmente a posição e velocidade, procedendo de maneira iterativa, mas respeitando a ordem de cálculo abaixo:

$$x_{\text{novo}} = x_{\text{velho}} + \Delta t v_{\text{novo}}$$

Obtém $F_{\text{velho}} = F(x_{\text{novo}})$ com a FFT ou com o Método Matricial

$$v_{\text{novo}} \rightarrow v_{\text{velho}}$$
$$v_{\text{novo}} = v_{\text{velho}} + \frac{F_{\text{velho}}\Delta t}{m}$$
$$x_{\text{novo}} \rightarrow x_{\text{velho}}.$$

Como no método de Euler, cada vez que as operações acima descritas são executadas, o tempo é incrementado por Δt e o programador pode repetir este processo até o tempo de simulação T_{sim} desejado. Perceba também que o uso do Método de Leapfrog exige cuidados especiais ao calcular as energias a partir de v (cinética) e x (potencial), pois estas devem ser ajustadas para que sejam calculadas no mesmo tempo; do contrário veríamos uma violação explícita da conservação da energia. Outro ponto a ser observado é quando a posição da partícula é colocada fora do sistema pelo método, ou seja, o caso em que durante a execução do programa se tem

$$x_{novo} < 0 \qquad \text{ou} \qquad x_{novo} > L.$$

Neste caso, o programa deve reposicionar a partícula no intervalo $0 \leq x \leq L$. Isso é feito simplesmente movendo-a por um período inteiro L para a esquerda ou direita, garantindo a periodicidade do sistema.

3.5 Inicializando o Programa ES1

Para rodar o programa, o usuário deve informar a posição e velocidade inicial de cada partícula do plasma, por espécie. Com base nestes dados, obteremos a dinâmica resultante do sistema usando as técnicas descritas acima. Uma maneira de fazer isso é estimar uma função densidade[10] $f(x, v, t = 0)$ para cada espécie do sistema e, com base nela, colocar as partículas distribuídas uniformemente pelo comprimento L. No entanto, na maioria das vezes, a teoria nos fornece a densidade de carga $\rho(x, t = 0)$ ou densidade de partículas $n(x, t = 0)$ do sistema. Por isso, vamos desenvolver uma técnica para derivar as posições das partículas em um plasma uniforme, a partir das perturbações na densidade de carga, e determinar todas as quantidades relevantes, para o caso de perturbações senoidais de pequena amplitude.

Sejam duas partículas vizinhas com posições iniciais x_{0a} e x_{0b}. Neste caso, a densidade (de ordem zero) n_0 de partículas em um plasma uniforme é:

$$n_0 = \frac{1}{x_{0b} - x_{0a}}. \tag{3.31}$$

Agora perturbamos as posições de ordem zero para obter

$$x_a = x_{0a} + \delta x_a \qquad x_b = x_{0b} + \delta x_b,$$

de forma que para algum x tal que $x_a < x < x_b$, a nova densidade é da

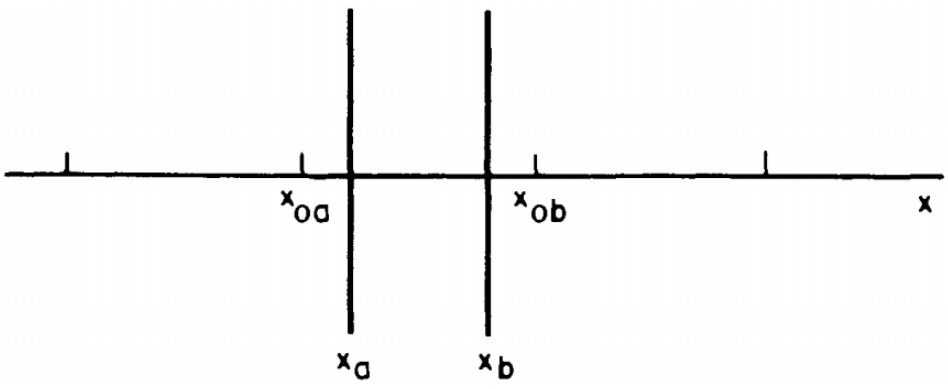

Figura 8: As posições não perturbadas, x_{0a} e x_{0b}, e perturbadas, x_a e x_b, de duas partículas vizinhas. Fonte: Ref. [1], p. 82.

[10] Esta função densidade deve fornecer a probabilidade de haver uma partícula nos intervalos $(x, x + \Delta x)$ e $(v, v + \Delta v)$ no tempo $t = 0$.

forma:

$$n(x) = n_0 + n_1(x) = \frac{1}{x_b - x_a} = \frac{1}{x_{0b} + \delta x_b - x_{0a} - \delta x_a}$$

$$= \frac{1}{\underbrace{x_{0b} - x_{0a}}_{n_0}} \left(\frac{1}{1 + \frac{\delta x_b - \delta x_a}{x_{0b} - x_{0a}}} \right) = \frac{n_0}{1 + \underbrace{\frac{\delta x_b - \delta x_a}{x_{0b} - x_{0a}}}_{\frac{\partial \delta x}{\partial x_0}}} = \frac{n_0}{1 + \frac{\partial \delta x}{\partial x_0}}.$$

Ou seja:

$$\frac{\partial \delta x}{\partial x_0} = -\frac{n_1(x)}{n_0 + n_1(x)}. \tag{3.32}$$

Assumindo que n_0 e $n_1(x)$ são conhecidos, podemos encontrar a perturbação $\delta x = x - x_0$ resolvendo a equação acima. Se estamos interessados em perturbações de pequena amplitude, tem-se $|n_1(x)| \ll n_0$, de forma que a equação 3.32 vira[11]:

$$\frac{\partial \delta x}{\partial x} \approx -\frac{n_1(x)}{n_0}. \tag{3.33}$$

A relação acima é de aplicabilidade geral, de forma que o programador precisa apenas definir a perturbação desejada. Para exemplificar, suponha uma perturbação senoidal em um plasma de uma única espécie

$$n_1(x) = A \sin(kx),$$

onde A é a amplitude da perturbação, sendo uma quantidade dada. Integrando a equação 3.33, temos:

$$\delta x = x - x_0 = \int \Big(-\frac{A \sin(kx)}{n_0} \Big) dx = \frac{A}{n_0 k} \cos(kx)$$

$$x = x_0 + \frac{A}{n_0 k} \cos(kx). \tag{3.34}$$

Desta forma o problema de colocar as partículas nos seus lugares está resolvido. As velocidades iniciais das partículas são geralmente obtidas a partir de uma distribuição Maxwelliana onde não existe correlação com as posições das partículas, mas o programador também pode optar por escrever a velocidade diretamente da definição

$$v = \frac{\delta x}{\Delta t} = \frac{A}{n_0 k \Delta t} \cos(kx), \tag{3.35}$$

[11] Use $x \to x_0$.

onde o Δt da equação acima não é necessariamente o mesmo Δt usado pelo programador quando roda o programa: este intervalo temporal é usado apenas uma vez apenas para definir as posições e velocidades iniciais das partículas, podendo este passo ser executado por uma subrotina à parte do programa. Repare que aqui o programador deve controlar a velocidade máxima produzida pela perturbação, controlando a amplitude A:

$$v_{max} = \frac{A}{kn_0\Delta t} \implies A = kn_0\Delta t v_{max}.$$

Repare que uma onda pode ser excitada no plasma em estudo, desde que $k_{min} \leq k \leq k_{max}$, ou seja, se k for um vetor de onda compatível com a relação[12]

$$k = n\frac{2\pi}{L},$$

onde $n \in (-\frac{N}{2}, \ldots, \frac{N}{2} - 1)$. Na figura 9 é possível identificar o efeito de uma perturbação senoidal em um sistema.

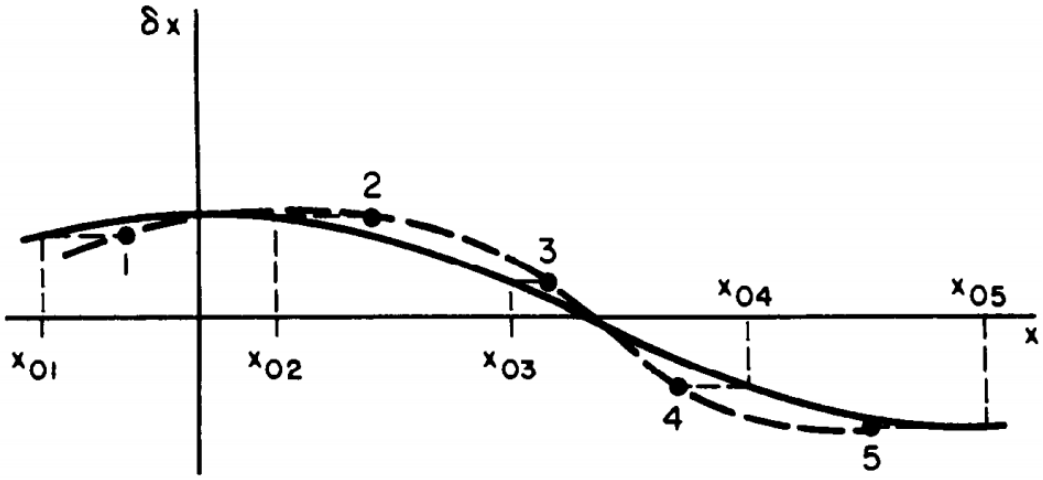

Figura 9: Posições x_{0i} uniformemente espaçadas com uma perturbação senoidal obtidas da relação $x_i = x_{0i} + \delta x(x_{0i})$. Fonte: Ref. [1], p. 83.

3.6 Informações Tiradas do Programa ES1

A ideia por trás de qualquer simulação computacional nesta área é ganhar *insight* sobre a física do plasma em questão. O objetivo de uma si-

[12]Mostramos esta relação na seção Solução por Transformada Rápida de Fourier - FFT.

mulação em particular pode ser o estudo de oscilações, instabilidades, transporte, difusão, etc. Portanto, o tipo de informação a ser tirada do programa deve ser orientado para o fenômeno em estudo. Se o espaço em disco não for um problema, deve-se, em princípio, agrupar toda a informação gerada pelo programa ES1 (x, v, ρ, ϕ, E, etc.) em cada passo Δt para um diagnóstico, junto ou separadamente com o programa. Isso é feito, no entanto, após uma análise do tipo mais comum de saída de dados do programa: os gráficos. Usualmente utiliza-se deste recurso de duas formas: *snapshots* (são gráficos instantâneos feitos durante a simulação, em tempos específicos determinados) e gráficos *história* (gráficos que mostram a evolução de uma quantidade ao longo da simulação). Através destes gráficos, o usuário do programa deve ser capaz de aprender algo sobre o comportamento qualitativo do sistema e, com isso, selecionar regiões de interesse para fazer uma análise quantitativa com o uso dos parâmetros (x, v, ρ, ϕ, E, etc.) gerados. Quantidades interessantes de serem plotadas na forma de *snapshots* seriam:

1. Espaço de fase, $v(x)$.
2. Densidade de velocidades, $f(v)$.
3. Densidade de cargas, $\rho(X)$.
4. Potencial elétrico, $\phi(X)$
5. Campo elétrico, $E(X)$
6. A densidade de energia, $\frac{1}{2}\rho(X)\phi(X)$.

Repare que ρ, ϕ, E e a densidade de energia são quantidades de grade, logo, tem-se a informação apenas nestes pontos. Já as quantidades interessantes de serem plotadas apenas ao final da simulação, na forma de um gráfico *história* seriam:

1. Energia eletrostática, $\frac{\Delta x}{2}\sum_{i=0}^{N-1}\rho(X_i)\phi(X_i)$[13].
2. Energia cinética por espécies, $\sum_i \frac{1}{2}m_i v_i^2$.
3. Energia de Drift[14], $\sum_i \frac{1}{2}m_i\langle v_i\rangle^2$.
4. Energia termal por partícula, $\sum_i \frac{1}{2}m_i(\langle v_i^2\rangle - \langle v_i\rangle^2)$.

[13] Da relação $\int_v \frac{1}{2}\rho(\mathbf{r})\phi(\mathbf{r})d^3r \underbrace{\rightarrow}_{\text{1D}} \int_0^L \frac{1}{2}\rho(x)\phi(x)dx \approx \frac{\Delta x}{2}\sum_{i=0}^{N-1}\rho(X_i)\phi(X_i)$.

[14] Dá uma ideia da mudança gradual na energia cinética do sistema ao longo do tempo.

5. Energia total do sistema.

Naturalmente, outras quantidades além das listadas podem ser de interesse do usuário do programa. Por exemplo, no estudo da difusão, deseja-se monitorar a média do desvio quadrado da velocidade e posição das partículas, de forma que $[v_i - v_i(t=0)]^2$ e $[x_i - x_i(t=0)]^2$ sejam armazenados a cada passo e, então, plotadas. No estudo de ondas no plasma, pode-se desejar resolver as frequências de oscilação, de modo que uma análise de Fourier em relação ao tempo seja feita numa quantidade como $\phi(k,t)$, talvez precisando do armazenamento desta quantidade ao longo do programa. Desta forma, o usuário do programa, assim como um físico experimental no laboratório, irá acumular dados suficientes para verificar a correção de sua previsão teórica, ou obter o *insight* desejado para trabalhar com este problema em particular.

4 Conclusão

O desenvolvimento deste trabalho foi orientado para os fundamentos da simulação computacional em física de plasmas e, neste sentido, o presente estudo conclui o seu objetivo de se tornar uma ferramenta de base para estudos mais avançados nesta grande área. As dificuldades inerentes ao problema de simulação computacional de sistemas de muitas partículas foi superada por meio do uso de técnicas analíticas relativamente modernas, visando sempre a eficiência computacional, mas também levando em conta as possíveis limitações físicas indissociáveis deste processo por meio de caminhos alternativos menos dispendiosos do ponto de vista computacional. Os fundamentos do algoritmo ES1 foram elucidados sistematicamente, com enfoque no entendimento dos aspectos teóricos e, em todos os pontos, fazendo a conexão na forma de pseudolinguagem com o algoritmo em si. A vantagem obtida ao se proceder desta maneira foi que o conteúdo se tornou acessível a qualquer programador, independentemente da linguagem de programação dominada, facilitando a difusão do conhecimento. O enfoque no entendimento da teoria em si foi de suma importância para alcançar os objetivos do trabalho, tendo em vista que as mesmas técnicas aqui desenvolvidas são usadas na simulação de plasmas mais complexos em duas e três dimensões.

Referências

[1] BIRDSALL, C. K., A. B. Langdon. Plasma Physics via Computer Simulation. 1ª Edição. New York: Taylor & Francis, c2005.

[2] CHEN, F. F. Introduction to Plasma Physics and Controlled Fusion. 2ª Edição. New York : Plenun Press, c1974.

[3] KOSTOMAROV, D.P., Y.N. Dnestrovskii. Numerical Simulation of Plasmas. 1ª Edição. Springer-Verlag Berlin Heildelberg, 1986.

[4] WAKATANI, M., K. Nishikawa. Plasma Physics. 3ª Edição. Springer-Verlag Berlin Heildelberg, 2000.