UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE ENGENHARIA MECÂNICA
ENGENHARIA MECÂNICA

TIAGO GASPAR DA ROSA

ALGORITMO DE SOLUÇÃO PARA O MÉTODO DAS ABSCISSAS PRESCRITAS PARA MÉTODOS DE LATTICE-BOLTZMANN EM DUAS DIMENSÕES EUCLIDEANAS

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA
2019

TIAGO GASPAR DA ROSA

ALGORITMO DE SOLUÇÃO PARA O MÉTODO DAS ABSCISSAS PRESCRITAS PARA MÉTODOS DE LATTICE-BOLTZMANN EM DUAS DIMENSÕES EUCLIDEANAS

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de TCC 2, do curso de Engenharia Mecânica da Coordenação de Engenharia Mecânica (COEME) da Universidade Tecnológica Federal do Paraná (UTFPR) como requisito parcial para obtenção do título de Engenheiro Mecânico.

Orientador: Prof. PhD. Christian Naaktgeboren

GUARAPUAVA
2019

TERMO DE APROVAÇÃO

ALGORITMO DE SOLUÇÃO PARA O MÉTODO DAS ABSCISSAS PRESCRITAS PARA MÉTODOS DE LATTICE-BOLTZMANN EM DUAS DIMENSÕES EUCLIDEANAS

por

TIAGO GASPAR DA ROSA

Este Trabalho de Conclusão de Curso foi apresentado em 03 de julho de 2019 como requisito parcial para a obtenção do título de Bacharel em Engenharia Mecânica. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Dr. Christian Naaktgeboren
Prof. Orientador

_____
Dr.ª Raquel da Cunha Ribeiro da Silva
Membro Titular

_____
Dr. Sérgio Dalmás
Membro titular

_____
Dr.ª Aldo Przybysz
Coordenador do curso de Engenharia Mecânica

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

Resumo

ROSA, Tiago Gaspar da; ALGORITMO DE SOLUÇÃO PARA O MÉTODO DAS ABSCISSAS PRESCRITAS PARA MÉTODOS DE LATTICE-BOLTZMANN EM DUAS DIMENSÕES EUCLIDEANAS . 108 f. Trabalho de Conclusão de Curso, Engenharia Mecânica, Universidade Tecnológica Federal do Paraná, Guarapuava, 2019.

Os métodos mesoscópicos de Lattice Boltzmann atuam pela resolução numérica da equação de Boltzmann para uma função de distribuição probabilística de partículas, permitindo a recuperação de grandezas macroscópicas de um sistema em não equilíbrio. As partículas são distribuidas e interagem em lattices - redes - que discretizam um espaço de velocidades e o próprio espaço. O conjunto formado pelos pesos, velocidades discretas e a escala do lattice é denominado *stencil*. A discretização do espaço de velocidades a partir da equação contínua é um passo crítico na derivação das redes de Boltzmann. Há diferentes modos de conduzi-la, incluindo o método das abcissas prescritas, que trabalha com um conjunto de velocidades discretas adimensionais pré-definidas. O problema, uma quadratura para determinação dos pesos e a escala do lattice, equivale a um produto interno tensorial entre dois polinômios tensoriais de Hermite no espaço discreto. Não foi encontrado pelo autor bibliografia detalhada sobre como resolver as quadraturas, assim este trabalho investigará algoritmos para solucioná-las.

Palavras-chaves:Métodos de Lattice Boltzmann. *Stencil*. Abcissas Prescritas. Tensores de Hermite. Equação de Boltzmann.

Abstract

ROSA, Tiago Gaspar da; SOLUTION ALGORITHM FOR THE PRESCRIBED ABSCISSAS METHOD FOR LATTICE-BOLTZMANN METHODS IN TWO EUCLIDEAN DIMENSIONS. 108 f. Undergraduate Thesis, Mechanical Engeneering, Federal University of Technology - Paraná, Guarapuava, 2019.

The lattice Boltzmann methods, mesoscopic, act by the numerical solution of the Boltzmann equation for a particle probabilistic distribution function, allowing the retrieval of the system macroscopic quantities. The particles are distributed and interact in lattices that discretize a velocity space and a spatial space. The set formed by the weights, discrete velocities and the scale of lattice is called stencil. The velocity space from the continuous equation discretization is a critical step in the Boltzmann stencils derivation. There are different ways to conduct it, such that one, prescribed abscissas method, works with a set of predefined dimensionless discrete velocities. The problem, a quadrature for the weights and the lattice scale determination, is equivalent to an internal product tensor between two Hermite tensor polynomials in the discrete space. There is no detailed in the bibliography about how to solve the quadratures, so this work will investigate algorithms to solve them, which lead to the creation of new lattice-Boltzmann methods.

Key-words:Lattice Boltzmann Methods. *Stencil*. Prescribed Abscissas. Hermite Tensors. Boltzmann Equation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BBGKY	Bogoliubov, Born, H. S. Green, Kirkwood e Yvon
BGK	Bhatnagar-Gross-Kroos
BGL	Boltzmann Gas Limit
LBM	Lattice Boltzmann Method
LI	Linearmente Independente
MPA	Método das Abcissas Prescritas
PPT	Plano de Produto Tensorial
CAS	Computer Algebra System
ID	Interpolative Decomposition

Lista de símbolos

a_{Eq}	Equação para determinação da escala a do *lattice*
d_{Eq}	Densidade final de equações L.I. em comparação ao total de equações inspecionadas
LI_{Eq}	Número de equações linearmente independentes
T_{Eq}	Total de equações inspecionadas por caminho percorrido
LI_{Max}	Quantidade de componentes L.I. que fazem o sistema ser possível e determinado
N_{Max}	Máximo N, tal que $m+n \leq 2N$, numericamente equivalente à máxima ordem possível para o *lattice*. $N_{Max} = 6$ para o caso estudado
$n(a)_{Max}$	Número de equações candidatas à a_{Eq}
c_T	Velocidade térmica de referência
c	Velocidade de som gás ideal
γ	Coeficiente de expansão isentrópica
p	Paridade da soma $m+n$ nos algoritmos de iteração.
f	Função distribuição de probabilidade

Sumário

Capítulo 1

Introdução

Métodos numéricos agregam importância na engenharia, em razão das restrições de uso de soluções analíticas a condições de aplicações muito específicas e com pouco respaldo prático. Dentro dessa vertente, incorporam-se os métodos com aplicação tradicional para solução de problemas de engenharia no campo da fluidodinâmica computacional, como as diferenças finitas, os volumes finitos e elementos finitos, que foram precursores de métodos mais modernos, como os de *lattice* Boltzmann.

Os métodos de *lattice* Boltzmann são mesoscópicos e atuam pela resolução numérica da equação de Boltzmann para uma distribuição probabilística de partículas μ-scópicas. As grandezas macroscópicas, como densidade, velocidade e energia cinética, são recuperadas pelo cálculo dos momentos hidrodinâmicos de uma função de distribuição de massa no espaço de velocidades.

Originado a partir da teoria cinética dos gases, esse método usa uma abordagem intermediária entre a Euleriana e a Lagrangiana, na qual partículas são distribuídas em *lattices*, também chamados de redes, que discretizam um espaço de velocidades e o próprio espaço físico, de modo que partículas interagem em colisões binárias e se deslocam para outras redes vizinhas após um incremento de tempo (BAZARIN, 2018). O conjunto formado pelos pesos, velocidades discretas e a escala a do *lattice* é chamado *lattice*-Boltzmann *stencil*. Cada *stencil* formulado define um novo método.

A discretização do espaço de velocidades a partir da equação contínua é um passo crítico na derivação das redes de Boltzmann – LBE – (MATTILA; HEGELE JR.; PHILIPPI, 2014). Há diferentes modos de conduzi-la, tal que um, em especial, trabalha-se com um conjunto de velocidades discretas adimensionais pré-definidas, recebendo a denominação de método das abcissas prescritas. A importância da coincidência entre as velocidades discretas e posições discretas está na eliminação de artefatos como difusão numérica espúria.

O problema consiste em uma quadratura, na qual o conjunto de pesos deve ser encontrado. Além disso, é requerido um fator de escala relacionado à velocidade do som na rede, de forma a cumprir uma condição de isotropia (MATTILA; HEGELE JR.; PHILIPPI,

2014).

Publicações recentes mostram que resolver a quadratura é equivalente a encontrar o produto interno tensorial entre dois polinômios tensoriais de Hermite no espaço discreto, que coincidirá com o produto interno no espaço contínuo. Além do mais, a equivalência entre os produtos nos dois espaços, para uma ordem N, deve assegurar a ortogonalidade dos polinômios de Hermite tensorial no espaço discreto (MATTILA; HEGELE JR.; PHILIPPI, 2014).

Quanto a ordem N, é importante ressaltar que seu incremento é intrínseco a quantidade de físicas que o método é capaz de solucionar. Apesar de uma vasta gama de aplicações ser satisfatoriamente descrita pela equação de Navier-Stokes, especialmente escoamentos Newtonianos monofásicos, incongruências aparecem em regimes de escoamentos mais complexos, envolvendo múltiplas fases, altos números de Mach ou de Knudsen. Em tais situações, descrições hidrodinâmicas de alta ordem são necessárias (SHAN XUE-FENG YUAN, 2006). No entanto, a complexidade computacional de solução das quadraturas cresce exponencialmente com esse aumento.

Compete a este projeto de pesquisa investigar algoritmos de solução para o método das abcissas prescritas, que enseja o desenvolvimento de novos métodos de geração de redes de Boltzmann. Em decorrência, fará uso do resultado outrora citado na determinação computacional dos pesos e velocidades em *stencils* de ordem progressiva como parte da otimização do código desenvolvido.

1.1 Objetivos

1.1.1 Objetivos gerais

São objetivos gerais ilustrar o método das abcissas prescritas e propor um algoritmo de solução de menor complexidade computacional em relação a abordagens de "força bruta", definidas como a inspeção de todos os casos possíveis no interior do produto interno tensorial da Equação 3.48 e explicados de acordo com a Seção 3.1.4.

1.1.2 Objetivos específicos

O cumprimento do objetivo descrito acima será dividido nas etapas:

1. Revisão bibliográfica e metodológica.
2. Ilustrar o MPA em stencils de ordens progressivas.
3. Propor e testar diferentes algoritmos de solução do MPA, incluindo Benchmarking.
4. Selecionar melhor proposta.
5. Realizar estudos de caso: re-derivar, com Benchimarking, *stencils* existentes na literatura.

1.2 Justificativa

Essa investigação se faz pertinente, à medida que não foram encontradas publicações que expliquem de maneira detalhada como resolver as quadraturas, apesar de soluções de *stencils* serem frequentemente apresentadas na literatura.

Capítulo 2

Revisão Bibliográfica

Uma recapitulação de conceitos introdutórios aos métodos Lattice Boltzmann é apresentada.

2.1 Teoria Cinética

A Mecânica estatística de equilíbrio e de não-equilíbrio têm em seus objetivos a determinação de propriedades termo-fluidas básicas de sistemas que são caracterizados por um grande número de graus de liberdade.

O postulado básico da mecânica estatística de equilíbrio e não-equilíbrio afirma que todas as propriedades macroscópicas intensivas de um sistema podem ser descritas em termos do estado microscópico dele. No entanto, devido à alta complexidade computacional em se trabalhar com um grande número de GDL's, há a busca por abordagens intermediárias, denominadas mesoscópicas.

Em nível molecular, a descrição usualmente discorre em termos de coordenadas espaciais e momento das moléculas que constituem o sistema de interesse. Se a descrição for estatístico-mecânica irá requerer $6N$ variáveis, em que N é o número de moléculas do sistema - cerca de 10^{23} por mol. Tais variáveis tomam, usualmente, $3N$ coordenadas espaciais e $3N$ momentos conjugados das moléculas constituintes.

Buscou-se expor a teoria cinética dos gases nesta seção com base em Harris (2004) e no seu decorrer se supõe partículas sem estrutura, isto é, as quais são completamente determinadas por uma massa pontual no espaço, tal que os efeitos de orientação, vibração, e todos os fenômenos que são relacionados com a estrutura das moléculas são negligenciáveis.

Assim, o estado de um sistema em qualquer instante de tempo pode ser especificado por um ponto único do espaço de dimensão $6N$, denominado espaço de fase. Um ponto desse espaço determina o micro-estado de um sistema.

A função de distribuição de massa F_N evolui, no ponto do espaço de fase $\{\mathbf{x}\} =$

$\{\mathbf{q}\},\{\mathbf{p}\}$ em relação ao tempo, conforme a equação de Liouville:

$$dF_N = \frac{\partial F_N}{\partial t}dt + \sum_{i=1}^{N} \frac{\partial F_N}{\partial \mathbf{q}_i} \cdot d\mathbf{q}_i + \sum_{i=1}^{N} \frac{\partial F_\mathbf{N}}{\partial \mathbf{p}_i} \cdot d\mathbf{p}_i \tag{2.1}$$

que especifica mudanças infinitesimais desenvolvida na trajetória do sistema no espaço de fase. Ela associa uma mudança local, no ponto $\{\mathbf{x}\}$ de F_N, a uma mudança total $\frac{dF_N}{dt}$ ao longo da trajetória da vizinhança de $\{\mathbf{x}\}$. O teorema de Liouville declara que:

$$\frac{dF_N}{dt} = 0 \tag{2.2}$$

ao longo da trajetória de qualquer ponto de fase a densidade de probabilidade na vinhança desse ponto se mantêm constante com o tempo. Uma vez que F_N permaneça constante — Equação 2.2 — qualquer função F_N terá a mesma propriedade.

Uma descrição estatístico-mecânica completa conduz a um problema computacionalmente complicado por causa do grande número de graus de liberdade. Felizmente, boas aproximações podem ser extraídas com base nas primeiras funções de distribuição de probabilidade reduzida, que considera uma amostragem do total de partículas. A função de distribuição reduzida para R partículas $F_R(\mathbf{x}_1,\mathbf{x}_2, \cdot\ ,\ \mathbf{x}_R,t)$ é definida como:

$$F_R(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_R,t) \equiv \int d\mathbf{x}_{R+1}\cdots d\mathbf{x}_N F_N(\mathbf{x}_1,\cdots,\mathbf{x}_R,\cdots,\mathbf{x}_N,t) \tag{2.3}$$

As duas primeiras funções para $R = 1$ e 2 são:

$$F_1(\mathbf{x}_1,t) = \int d\mathbf{x}_2\cdots d\mathbf{x}_\mathbf{N} F_N(\mathbf{x}_1,\cdots,\mathbf{x}_\mathbf{N},t) \tag{2.4}$$

e

$$F_2(\mathbf{x}_1,\mathbf{x}_2,t) = \int d\mathbf{x}_3 \cdot\cdot\ \ d\mathbf{x}_N F_N(\mathbf{x}_1,\cdots,\mathbf{x}_N,t) \tag{2.5}$$

Sendo $F_1(x_1,t)\,dx_1$ a probabilidade de encontrar a molécula 1 no volume diferencial $d\mathbf{x}_1$ sobre o ponto de fase $\mathbf{x}_1$ no tempo t. A densidade numérica e a densidade de partículas podem ser definidas, respectivamente, como:

$$f_N(\mathbf{x}_1,t) = NF_1(\mathbf{x}_1,t) \tag{2.6}$$

e

$$f(\mathbf{x}_1,t) = NmF_1(\mathbf{x}_1,t) \tag{2.7}$$

Em que m é a massa de uma única molécula, $f_N(\mathbf{x}_1,t)\,d\mathbf{x}_1$ o número esperado de moléculas e $f(\mathbf{x}_1,t)\,d\mathbf{x}_1$ o de massa no elemento de volume incremental $d\mathbf{x}_1$ sobre o ponto de fase $\mathbf{x}_1$ no tempo t.

F_R deve ser simétrica em seus $\mathbf{x}_1,\mathbf{x}_2, \cdot\ ,\ ,\mathbf{x}_R$ argumentos para ser válida no caso especial de $F = N$. A fim de se descrever o comportamento dessa função é necessária uma equação que forneça seu desenvolvimento temporal, obtida a partir da equação de Liouville.

Como, $\dot{\mathbf{q}}_i = \mathbf{p}_i/m$ e $\dot{\mathbf{p}}_i = \mathbf{F}_i$, $\mathbf{F}_i$ sendo a força de campo que atua na i-ésima molécula que, na hipótese de forças externas ausentes, é somente a força exercida por i em outra molécula do sistema. Assume-se que essa interação é derivada de um potencial entre duas partículas. Dessa forma, a força exercida na i-ésima molécula pela j-ésima resulta do potencial ϕ_{ij}, função de $\left|\mathbf{r}_{ij}\right| = |\mathbf{q}_i - \mathbf{q}|$. Então:

$$\mathbf{F}_i = -\sum_{j=1\neq i}^{N} \frac{\partial \phi_i}{\partial \mathbf{q}_i} \tag{2.8}$$

A equação para F_R pode ser obtida de 2.1 por sua integração sobre as fases $\mathbf{x}_{R+1}, \quad \cdots, \mathbf{x}_N$. Da combinação dos resultados anteriores, deduz-se para F_R a equação:

$$\begin{aligned} \frac{\partial F_R}{\partial t} \quad &+ \quad \sum_{i=1}^{R} \frac{\mathbf{p}_i}{m} \cdot \frac{\partial F_R}{\partial \mathbf{q}_i} - \sum_{i,j-1} \frac{\partial \phi_{ij}}{\partial \mathbf{q}_i} \cdot \frac{\partial F_R}{\partial \mathbf{p}_i} \\ &= \quad (N-R) \int d\mathbf{x}_{R+1} \sum_{i=1}^{R} \frac{\partial \phi_{iR+1}}{\partial \mathbf{q}_i} \cdot \frac{\partial F_{R+1}}{\partial \mathbf{p}_i} \end{aligned} \tag{2.9}$$

que produz uma cadeia de equações de evolução chamada hierarquia BBGKY, cuja sigla homenageia os autores independentes: Bogoliubov, Born, H. S. Green, Kirkwood e Yvon. As duas primeiras equações dessa hierarquia:

$$\frac{\partial F_1}{\partial t} + \frac{\mathbf{p}_1}{m} \cdot \frac{\partial F_1}{\partial \mathbf{q}_1} = (N-1) \int d\mathbf{x}_2 \phi'_{12} \cdot \frac{\partial F_2}{\partial \mathbf{p}_1} \tag{2.10}$$

$$\begin{aligned} \frac{\partial F_2}{\partial t} \quad &+ \quad \frac{\mathbf{p}_1}{m} \cdot \frac{\partial F_2}{\partial \mathbf{q}_1} + \frac{\mathbf{p}_2}{m} \cdot \frac{\partial F_2}{\partial \mathbf{q}_2} - \dot{\phi}'_{12} \cdot \left(\frac{\partial}{\partial \mathbf{p}_1} - \frac{\partial}{\partial \mathbf{p}_2} \right) F_2 \\ &= \quad (N-2) \int d\mathbf{x}_3 \left\{ \phi'_{13} \cdot \left(\frac{\partial}{\partial \mathbf{p}_1} - \frac{\partial}{\partial \mathbf{p}_3} \right) + \phi'_{23} \cdot \left(\frac{\partial}{\partial \mathbf{p}_2} - \frac{\partial}{\partial \mathbf{p}_3} \right) \right\} F_3 \end{aligned} \tag{2.11}$$

evidenciam o problema de se determinar uma formulação fechada para F_1, removendo sua dependência de F_2 e, portanto, dos demais membros da hierquia.

Uma forma fechada para F_1 foi descoberta por Boltzmann em 1872, embora sem que um vínculo fosse estabelecido por ele com 2.1. Assim, a redução da primeira equação de BBGKY para a equação de Boltzmann é feita com base em um conjunto de suposições que restringem sua aplicabilidade a sistemas que:

1. Tem densidade suficiente baixa tanto que apenas colisões moleculares binárias são consideradas.

2. A influência espacial nas propriedades dos gases é baixa o sufiente para colisões serem localizadas no espaço físico.

3. O potencial interpartícula tem alcance curto o suficiente para a primeira declaração ser válida, isto é, uma partícula qualquer não pode interagir com mais de uma partícula no mesmo instante particular.

Na derivação da equação de Boltzmann, as suposições acima traduzem-se em termos matemáticos que especificam o regime físico ou limite, chamado de Boltzmann Gas Limit (BGL). Apenas dentro desse limite que a teoria clássica da equação de Boltzmann tem significado.

A derivação da equação procede-se de forma similar a utilizada na dedução das equações originadas da hierarquia BBGKY com a distinção de que o F_R é função da coordenada espacial e da velocidade, ao invés do momento. O ponto de fase é $[\mathbf{y}_R = \mathbf{q}_R, \mathbf{v}_R]$.

A integração da equação de Liouville sobre $D1$ fornece a equação BBGKY para F_1^σ, com a possibilidade das substituições de F_1 por F_1^σ e F_2 por F_2^σ:

$$\frac{\partial F_1^*}{\partial t} + \mathbf{v}_1 \cdot \frac{\partial F_1^*}{\partial q_1} + (N-1) \oint_{s_2} dv_2 d\mathbf{S}_2 \cdot (\mathbf{v}_1 - \mathbf{v}_2) F_2^* - (N-1) \frac{\partial}{\partial \mathbf{v}_1} \cdot \int_{|\mathbf{q}_1 - \mathbf{q}_2| > 0} d\mathbf{y}_2 \phi'_{12} F_{2\sigma} = 0 \tag{2.12}$$

na qual S_i é a esfera $|\mathbf{q}_i - \mathbf{q}_1| = \sigma$.

Uma mudança de coordenada que contempla o plano perpendicular ao vetor $\mathbf{V} = \mathbf{v}_2 - \mathbf{v_1}$ com origem em $\mathbf{q}_1$ e definindo um sistema coordenadas polares r_1 e no plano para que cada ponto de S_2 corresponda a um ponto do disco $0 \leq r \leq \sigma$ e $0 \leq \varepsilon \leq 2\pi$. Após essa alteração e a realização de alguns pressupostos adicionais, a equação:

$$\frac{\partial F_1^\sigma}{\partial t} + \mathbf{v}_i \cdot \frac{\partial F_1^\sigma}{\partial \mathbf{q}_1} = (N-1) \int d\omega d\mathbf{v}_2 V \left[F_2^\sigma (y_1, y_2, +t) - F_2^\sigma (y_1, y_2, -t) \right] \tag{2.13}$$

torna-se a forma fechada para F_1, conhecida como equação de Boltzmann. O lado direito dessa equação integral-diferencial não linear inclui um operado integral em v, com parâmetros t e $\mathbf{q}$, denominado operador de colisão.

2.2 Equação de Boltzmann

A equação de Boltzmann, de autoria de Ludwig Boltzmann, apresenta formulação adimensional contínua expressa por:

$$\frac{df^*}{dt^*} = \frac{\partial f^*}{\partial t^*} + \xi^* \cdot \nabla_{\mathbf{x}} f^* + \mathbf{g}^* \cdot \nabla_\xi f^* = \Omega^* \tag{2.14}$$

Sendo f a função distribuição de probabilidade, ξ vetor de velocidade da partícula, g força externa atuante e Ω um operador de colisão, sendo todos os parâmetros adimensionais, indicados por *. Detalhes sobre adimensionalização estão disponíveis no apêndice B.9 de Bazarin (2018).

A interpretação física da equação é de evolução temporal de um sistema de partículas à nível mesoscópico, ou seja, uma descrição intrínseca a uma função de distribuição de probabilidade (BAZARIN, 2018).

Além disso, a validade desse modelo requer o cumprimento das hipóteses de colisões binárias entre partículas, não-correlação de velocidades antes e depois da colisão e força de interação molecular preponderante nas colisões entre partículas, desprezando-se as forças externas (KREMER, 2005).

2.3 Discretização da Equação de Boltzmann

Conforme mencionado anteriormente, a equação contínua (2.14) é inapta para propósito de simulação computacional, tal que sua discretização é necessária. O método *lattice*-Boltzmann é a forma discretizada de (2.14) no espaço-tempo e espaço de velocidades. (HE; LUO, 1997) apresenta um procedimento geral para executar essa derivação.

Um exemplo de equação discreta de evolução do sistema, isto é a equação de *lattice*-Boltzmann (LBE), que determina a função de distribuição para partícula única é (MATTILA; HEGELE JR.; PHILIPPI, 2014):

$$f_i(\mathbf{r}+\delta_t\mathbf{c}_i, t+\delta_t) = f_i(\mathbf{r},t) + \delta_t\Omega_i(\mathbf{r},t), \quad i=0,\ldots,q-1 \tag{2.15}$$

Em modelos de *lattice* Boltzmann convencionais, vetores de velocidade discretas sempre conectam dois pontos de um *lattice* uniforme, isto é, o acréscimo $(\mathbf{r}+\delta_t\mathbf{c}_i)$ será um ponto da rede, desde que $\mathbf{r}$ o seja. O intervalo espacial de um *lattice* uniforme e o passo de tempo discreto, são respectivamente h_r e δ_t. O quociente entre os dois parâmetros define a velocidade de referência do *lattice* $c_r = h_r/\delta_t$, que adimensionaliza as velocidades discretas, de modo que $\mathbf{c}_i = c_r\mathbf{c}_i^\star = c_T\mathbf{v}_i$.

Quando o operador de colisão não linear é substituído por $\Omega_i = -f_i^{\,\mathrm{neq}}/\tau$ é especificada a equação de *lattice*-BGK com tempo de relaxação singular τ. A função de não equilíbrio é definida como $f_i^{\,\mathrm{neq}} := f_i - f_i^{\mathrm{eq}}$ e tempo de relação é intrínseco ao retorno do sistema ao equilíbrio após a perturbação (MATTILA; HEGELE JR.; PHILIPPI, 2014).

Os três primeiros momentos da função de distribuição no espaço com b velocidades, retornam grandezas macroscópicas tais como densidade local do fluido ρ, velocidade $\mathbf{u}$ e energia interna específica ε:

$$\rho = \sum_i^b f_i, \quad \rho\mathbf{u} = \sum_i^b f_i\mathbf{c}_i, \quad \rho\varepsilon = \sum_i^b f_i(\mathbf{c}_i-\mathbf{u})^2 \tag{2.16}$$

Segundo Mattila, Hegele Jr. e Philippi (2014), uma função de equilíbrio discreta pode ser especificada através da expansão da função de distribuição de equilíbrio de Maxwell-Boltzmann. Se o modelo é isotérmico, a função de distribuição de equilíbrio discreto é:

$$f_i^{\mathrm{eq}}(\rho,\mathbf{u}) = \rho(\mathscr{K}_i^{(0)} + \mathscr{K}_{i,\alpha}^{(1)}u_\alpha + \mathscr{K}_{i,\alpha\beta}^{(2)}u_\alpha u_\beta + \mathscr{K}_{i,\alpha\beta\gamma}^{(3)}u_\alpha u_\beta u_\gamma + \cdots + \mathscr{K}_{i,\alpha_1\cdots\alpha_n}^{(n)}\prod_i^n u_{\alpha_i}) \tag{2.17}$$

em que:

$$\mathscr{K}_{i,\alpha_1\cdots\alpha_n}^{(n)} \equiv \mathbf{K}_i^{(n)} := \frac{w_i}{n!c_T^{2n}}\mathbf{H}^{(n)}(\mathbf{c}_i) \tag{2.18}$$

representa a projeção cinética, que identifica as variáveis hidrodinâmicas dentro de um espaço de descrição cinética.

2.3.1 Discretização do Espaço de Velocidades

O passo inicial para derivação do método de *lattice*-Boltzmann é a discretização do espaço microscópico de velocidades $\mathbf{c} = c_T \mathbf{v}$, no qual $\mathbf{v}$ é uma velocidade adimensional e c_T é a velocidade térmica de referência e T_0 a temperatura de referência.

Mattila, Hegele Jr. e Philippi (2014) exemplifica o método das abcissas prescritas para a criação de *stencils* de *lattice*-Boltzmann. Nesse método, um conjunto de velocidades discretas adimensionais $\mathbf{c}_i$ é prescrito, de forma que $\mathbf{v}_i = a_s \mathbf{c}_i^*$. Cada vetor do conjunto deve apresentar uma contraparte oposta $\mathbf{c}_i = -\mathbf{c}_i$ e correspondentes que garantam a simetria radial do *stencil* para modelos regulares não deslocados.

Os pesos w_i e a constante a_s da escala de velocidades do *lattice*, são calculados a partir de um problema de quadratura até a N-ésima ordem.

2.3.2 Conservação da Isotropia

Nem toda rede é apropriada para uso, apenas as que conduzem a aproximações de derivadas isotrópicas, isto é, que não dependem de viés direcional. Isso implica nos vetores de *lattice* ($\mathbf{c}_i$), atenderem algumas condições a fim de que se permita a recuperação das equações de Navier-Stokes.

De acordo Mattila, Hegele Jr. e Philippi (2014), os pesos e velocidades que preenchem as condições de isotropia, detêm correspondência entre os momentos contínuos e discretos da função de distribuição de equilíbrio.

2.3.3 Operadores de Colisão

O operador de colisão Ω, indicado em (2.14) representa a taxa de mudança de f, dada colisões moleculares binárias. Esse operador é uma função integral bilinear de f, que garante a conservação de massa momento e energia:

$$\int \psi_i(\xi) f(x,\xi,t) d\xi = 0 \tag{2.19}$$

Para a equação de Boltzmann, é possível mostrar que a função H, definida por $H(t) = \int f \ln f dx d\xi$, sempre decresce com o tempo (teorema-H),

$$\frac{dH}{dt} \leq 0 \tag{2.20}$$

Que tem igualdade, quando o sistema atinge o estado de equilíbrio, no qual a função de distribuição é a de Maxwell-Boltzmann:

$$f^{(eq)}(x,\xi,t) = \frac{\rho}{(2\pi RT)^{3/2}} \exp\left[-\frac{(\xi - u)^2}{2RT}\right] \tag{2.21}$$

Na qual $R = k_B/m$ é a constante do gás e k_b é a constante de Boltzmann e m a massa molecular. No equilíbrio, a colisão não tem efeito líquido $\Omega\left(f^{(eq)}, f^{(eq)}\right) = 0$. As

observações anteriores são condicionantes para a dedução dos operadores (GUO; SHU, 2013). Uma abordagem mais detalhada sobre a dedução da equação de Boltzmann a partir da hierarquia de BBGKY consta no apêndice B.5 de Bazarin (2018).

2.3.4 Modelo de Colisão BGK

O modelo BGK (Bhatnagar-Gross-Kroos), conforme Andrade (2016), representa uma mudança no fluxo de probabilidades futuras devido à interação – colisões - entre duas partículas e o intercâmbio mútuo de momento. Sendo operador de colisão BGK:

$$\Omega_{BGK} = -\frac{1}{\tau_i}\left[f - f^{(eq)}\right] \tag{2.22}$$

Em que τ_i é o tempo de relaxação. Guo e Shu (2013) entende que o modelo reflete o efeito geral de colisões intermoleculares, isto é, a função de distribuição relaxa para o estado de equilíbrio com colisões.

Esse modelo conserva as grandezas de massa, momento e energia do sistema, e cumpre o teorema-H.

Capítulo 3

Metodologia

Algumas convenções de notação serão estabelecidas, junto com uma explanação sobre os polinômios de Hermite tensoriais e tensores monomiais e a resolução manual do *stencil* `D2Q9` em vista a exemplificar a metodologia aplicada neste projeto de pesquisa.

3.1 Considerações Prévias

A teoria de *lattice*-Boltzmann demanda conhecimento prévio sobre polinômios de Hermite tensoriais, monômios e suas propriedades. A atual sessão exibirá a fundamentação teórica pertinente as operações e notação dos operadores empregados no decorrer desta exposição. Uma nova notação, que difere daquela presente em (MATTILA; HEGELE JR.; PHILIPPI, 2014) é introduzida.

3.1.1 Convenções de Notação

A operação não padrão ${}^{n}_{k}\mathbb{C}_{\alpha}[\mathbf{AB}...]$ denota a soma de produtos tensoriais $[\mathbf{AB}...]$ sobre todas as possíveis combinações $\binom{n}{k}$ de índices livres de operadores, fixado um conjunto $\boldsymbol{\alpha}$ de a índices, por exemplo, $a > 0$, ou seja, $\boldsymbol{\alpha}$ pode ser o conjunto vazio.

Assim, a soma $A_{\alpha\beta}B_{\gamma\delta} + A_{\alpha\gamma}B_{\beta\delta} + A_{\alpha\delta}B_{\beta\gamma}$, na qual o índice $\boldsymbol{\alpha}$ permanece fixo como o 1º índice de $\mathbf{A}$ e o arranjo de índices nas $\binom{3}{1} = \binom{3}{2} = 3$ termos seguem combinações de 3 tomadas em 1 e 3 tomadas em 2 sobre os índices livres $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\delta}$ — com os índices de $\mathbf{B}$ sempre em ordem crescente — é representada por ${}^{3}_{1}\mathbb{C}_{\alpha}[A_{\alpha\beta}B_{\gamma\delta}]$ ou, equivalentemente, por ${}^{3}_{2}\mathbb{C}_{\alpha}[A_{\alpha\beta}B_{\gamma\delta}]$:

$$ {}^{3}_{2}\mathbb{C}_{\alpha}[A_{\alpha\beta}B_{\gamma\delta}] = A_{\alpha\beta}B_{\gamma\delta} + A_{\alpha\gamma}B_{\beta\delta} + A_{\alpha\delta}B_{\beta\gamma} \tag{3.1} $$

Ressalta-se que o argumento de ${}^{k}_{n}\mathbb{C}_{\alpha}$ é sempre igual ao primeiro termo da soma, e um *template* para os demais termos da mesma, como também é bom destacar que ${}^{k}_{n}\mathbb{C}_{\alpha} = {}^{k}_{n}\mathbb{C}_{\alpha}$, $k = a = n - a$.

Dessa forma, ${}^{3}_{1}\mathbb{C}_{\alpha}[A_{\alpha\beta}B_{\gamma\delta}]$ é uma adição $A_{\alpha\beta}B_{\gamma\delta}+...$ de termos, nas quais escolhe-se, dentre os 3 índices livres β,γ,δ 1 para a posição livre em $\mathbf{A}_{\alpha\_}$, ou, equivalentemente, 2 para as posições livres em $\mathbf{B}_{\_\_}$ em ${}^{3}_{2}\mathbb{C}_{\alpha}[A_{\alpha\beta}B_{\gamma\delta}]$.

Mattila, Hegele Jr. e Philippi (2014) expressa a mesma operação na notação menos explicita de $\mathbf{A}^{(k)} \circledast \mathbf{B}^{(n-k)}$:

$$\begin{aligned} \mathbf{A}^{(k)} \circledast \mathbf{B}^{(n-k)} &\equiv A^{(k)}_{\alpha_1\cdots\alpha_k} \circledast B^{(n-k)}_{\beta_1\cdots\beta_{n-k}} && (3.2)\\ &\equiv {}^{n}_{k}\mathbb{C}_{\alpha}[A_{\alpha_1\cdots\alpha_k}B_{\beta_1\cdots\beta_k}] \equiv^{n}_{k} \mathbb{C}_{\alpha}[\mathbf{AB}] && (3.3)\end{aligned}$$

Além disso, ainda conforme Mattila, Hegele Jr. e Philippi (2014) subscritos tipografados em negrito, $\mathbf{p}$ e $\mathbf{q}$, indicam conjuntos preenchidos de índices, ou seja, $\mathbf{pq}$ representa a união dos conjuntos, determinada pelo contexto. Assim, $\mathbf{p}=\{\alpha\}$ e $\mathbf{q}=\{\beta\gamma\delta\}$, implica em:

Outro padrão de notação empregado será ${}^{n}\mathbb{P}_{\alpha}$ para simbolizar o somatório dos $n!$ produtos tensoriais sobre todas as possíveis permutações de índices remanescentes ao conjunto α, que denota $\alpha_1\cdots\alpha_k$. Assim:

$${}^{2}\mathbb{P}_{\alpha}[A_{\alpha_1\beta_1}B_{\alpha_2\beta_2}] = A_{\alpha_1\beta_1}B_{\alpha_2\beta_2}+A_{\alpha_1\beta_2}B_{\alpha_2\beta_1} \tag{3.4}$$

Seja o delta de Kronecker δ_{ij}, o tensor de duas variáveis inteiras não negativas com lei de formação expressa como:

$$\delta_{ij} = \begin{cases} 1, \text{ se } i=j \\ 0, \text{ se } i\neq j \end{cases} \tag{3.5}$$

Em sua versão generalizada, ele é denotado por: $\Delta^{(n)} \equiv \Delta^{(n)}_{\alpha_1\cdots\alpha_{2n}}$. Esse tensor isotrópico é simétrico nos seus $2n$ índices. A recorrência entre os primeiros deltas de Kronecker generalizados é:

$$\begin{aligned} \Delta^{(0)} &= 1 && (3.6)\\ \Delta^{(1)}_{\alpha\beta} &= \delta_{\alpha\beta} && (3.7)\\ \Delta^{(2)}_{\alpha\beta\gamma\delta} &= \left(\delta_{\alpha\beta}\delta_{\gamma\delta}+\delta_{\alpha\gamma}\delta_{\beta\delta}+\delta_{\alpha\delta}\delta_{\beta\gamma}\right) && (3.8)\\ \Delta^{(3)}_{\alpha\beta\gamma\delta\varepsilon\zeta} &= (\delta_{\alpha\beta}\Delta^{(2)}_{\gamma\delta\varepsilon\zeta}+\delta_{\alpha\gamma}\Delta^{(2)}_{\beta\delta\varepsilon\zeta} && (3.9)\\ &+ \delta_{\alpha\delta}\Delta^{(2)}_{\beta\gamma\varepsilon\zeta}+\delta_{\alpha\varepsilon}\Delta^{(2)}_{\beta\gamma\delta\zeta}+\delta_{\alpha\zeta}\Delta^{(2)}_{\beta\gamma\delta\varepsilon}) && \\ \Delta^{(4)}_{\alpha\beta\gamma\delta\varepsilon\eta\theta} &= (\delta_{\alpha\beta}\Delta^{(3)}_{\gamma\delta\varepsilon\eta\theta}+\delta_{\alpha\gamma}\Delta^{(3)}_{\beta\delta\varepsilon\eta\theta}+\delta_{\alpha\delta}\Delta^{(3)}_{\beta\gamma\varepsilon\eta\theta} && (3.10)\\ &+ \delta_{\alpha\varepsilon}\Delta^{(3)}_{\beta\gamma\delta\zeta\eta\theta}+\delta_{\alpha\zeta}\Delta^{(3)}_{\beta\gamma\delta\varepsilon\eta\theta}+\delta_{\alpha\eta}\Delta^{(3)}_{\beta\gamma\delta\varepsilon\zeta\theta} && \\ &+ \delta_{\alpha\theta}\Delta^{(3)}_{\beta\gamma\delta\varepsilon\zeta\eta}) && \end{aligned}$$

A aplicação da notação do combinador de índices ${}^{n}_{k}\mathbb{C}_{\alpha}$ reduz (3.7) a (3.10) para:

$$\Delta^{(0)} = 1 \tag{3.11}$$
$$\Delta^{(1)}_{\alpha\beta} = \delta_{\alpha\beta} \tag{3.12}$$
$$\Delta^{(2)}_{\alpha\beta\gamma\delta} = {}^{3}_{1}\mathbb{C}_{\alpha}[\delta_{\alpha\beta}\delta_{\gamma\delta}] \tag{3.13}$$
$$\Delta^{(3)}_{\alpha\beta\gamma\delta\varepsilon\zeta} = {}^{5}_{1}\mathbb{C}_{\alpha}[\delta_{\alpha\beta}\Delta^{(2)}_{\gamma\delta\eta\zeta}] \tag{3.14}$$
$$= {}^{5}_{1}\mathbb{C}_{\alpha}[\delta_{\alpha\beta}{}^{3}_{1}\mathbb{C}_{\gamma}[\delta_{\gamma\delta}\delta_{\eta\zeta}]]$$
$$\Delta^{(4)}_{\alpha\beta\gamma\delta\varepsilon\eta\theta} = {}^{7}_{1}\mathbb{C}_{\alpha}[\delta_{\alpha\beta}\Delta^{(3)}_{\gamma\delta\eta\zeta\eta\theta}] \tag{3.15}$$
$$= {}^{7}_{1}\mathbb{C}_{\alpha}[\delta_{\alpha\beta}{}^{5}_{1}\mathbb{C}_{\gamma}[\delta_{\alpha\beta}{}^{3}_{1}\mathbb{C}_{\varepsilon}[\delta_{\eta\zeta}\delta_{\eta\theta}]]$$
$$\Delta^{(n)}_{\alpha_{i,i=1,\cdots,2n}} = {}^{2n-1}_{1}\mathbb{C}_{\alpha_1}[\delta_{\alpha_1\alpha_2}\Delta^{(n-1)}_{\alpha_{i,i=3,\cdots,2n}}] \tag{3.16}$$

O número de parcelas individuais a serem adicionadas é $\prod_{i=0}^{k}(n-2i) = (n)(n-2)\cdots$, no qual $k = \lceil n/2 \rceil - 1$, ou o duplo fatorial $(n)!! = (2n)!/(2^n n!)$.

Também denota-se como $\Lambda^{(n)} \equiv \Lambda^{(n)}_{\alpha_1\cdots\alpha_n\beta_1\cdots\beta_n}$ os tensores de ortogonalidade de ordem $2n$. Os primeiros são:

$$\Lambda^{(0)} = 1 \tag{3.17}$$
$$\Lambda^{(1)}_{\alpha_1\beta_1} = \delta_{\alpha_1\beta_1} \tag{3.18}$$
$$\Lambda^{(2)}_{\alpha_1\alpha_2\beta_1\beta_2} = \left(\delta_{\alpha_1\beta_1}\delta_{\alpha_2\beta_2} + \delta_{\alpha_1\beta_2}\delta_{\alpha_2\beta_1}\right) \tag{3.19}$$
$$\Lambda^{(3)}_{\alpha_1\alpha_2\alpha_3\beta_1\beta_2\beta_3} = (\delta_{\alpha_1\beta_1}\delta_{\alpha_2\beta_2}\delta_{\alpha_3\beta_3} + \delta_{\alpha_1\beta_1}\delta_{\alpha_2\beta_3}\delta_{\alpha_3\beta_2} \tag{3.20}$$
$$+ \delta_{\alpha_1\beta_2}\delta_{\alpha_2\beta_1}\delta_{\alpha_3\beta_3} + \delta_{\alpha_1\beta_2}\delta_{a_2\beta_3}\delta_{\alpha_3\beta_1} \tag{3.21}$$
$$+ \delta_{\alpha_1\beta_3}\delta_{\alpha_2\beta_1}\delta_{a_3\beta_2} + \delta_{\alpha_1\beta_3}\delta_{a_2\beta_2}\delta_{\alpha_3\beta_1}) \tag{3.22}$$

Na nova notação com o permutador de índices:

$$\Lambda^{(0)} = 1 \tag{3.23}$$
$$\Lambda^{(1)}_{\alpha_1\beta_1} = \delta_{\alpha_1\beta_1} \tag{3.24}$$
$$\Lambda^{(2)}_{\alpha_1\alpha_2\beta_1\beta_2} = {}^{2}\mathbb{P}_{\alpha}[\delta_{\alpha_1\beta_1}\delta_{\alpha_2\beta_2}] \tag{3.25}$$
$$\Lambda^{(3)}_{\alpha_1\alpha_2\alpha_3\beta_1\beta_2\beta_3} = {}^{3}\mathbb{P}_{\alpha}[\delta_{\alpha_1\beta_1}\delta_{\alpha_2\beta_2}\delta_{\alpha_3\beta_3}] \tag{3.26}$$
$$\Lambda^{(n)}_{\alpha_1\cdots\alpha_n\beta_1\cdots\beta_n} = {}^{n}\mathbb{P}_{\alpha}[\delta_{\alpha_1\beta_1}\delta_{\alpha_2\beta_2}\cdots\delta_{\alpha_n\beta_n}] \tag{3.27}$$

não há necessidade de relação recursiva.

Seja $\alpha \subseteq \alpha_{\mathbf{Ei}}$ um subconjunto tal que $\alpha_{\mathbf{Ei}}$ armazena os i-enésimos elementos de α que são iguais entre si. Uma nova formulação para os termos do tensor de ortogonalidade, que dispensa o uso de permutações, é:

$$\Lambda^{(n)}_{\alpha_1\cdots\alpha_n\beta_1\cdots\beta_n} = \begin{cases} 0, & \text{se } \exists\, i \text{ tal que: } n(\alpha_{\mathbf{Ei}}) \neq n(\beta_{\mathbf{Ei}}) \\ P_{n(\alpha_{\mathbf{E1}})}P_{n(\alpha_{\mathbf{E2}})}\cdots P_{n(\alpha_{\mathbf{En}})}, & \text{caso contrário} \end{cases} \tag{3.28}$$

em que $n(\alpha_{\mathbf{Ei}})$ denota o número de elementos de $\alpha_{\mathbf{Ei}}$ e $P_{n(\alpha_{\mathbf{E1}})}P_{n(\alpha_{\mathbf{E2}})}\cdots P_{n(\alpha_{\mathbf{En}})} = n(\alpha_{\mathbf{E1}})!n(\alpha_{\mathbf{E2}})!\cdots n(\alpha_{\mathbf{En}})!$ é o produto das permutações dos elementos desses subconjuntos. São exemplos não triviais para a Equação 3.28:

$$\Lambda_{00} = 2! = 2 \tag{3.29}$$

$$\Lambda_{0011} = 2!2! = 4 \tag{3.30}$$

$$\Lambda_{000111} = 3!3! = 36 \tag{3.31}$$

$$\Lambda_{001122} = 2!2!2! = 8 \tag{3.32}$$

3.1.2 Distinção entre *lattice* físico e conceitual

Discutir estratégias para a padronização dos *stencils* é importante em termos de comparação com parâmetros conhecidos. Em particular, o espaço discreto de velocidades pode ser adimensionalizado a partir de: $\mathbf{c} = c_T\mathbf{v}$, em que

$$c_T = \sqrt{k_b T_0/m} \tag{3.33}$$

indica a velocidade térmica de referência (MATTILA; HEGELE JR.; PHILIPPI, 2014) à temperatura T_0 em gás ideal, k_b a constante de Boltzmann e m a massa molecular do fluido. $\mathbf{v}$ é o espaço vetorial adimensional de velocidades. Assim, será padronizada a notação c_i para referir-se a um vetor de $\mathbf{c}$ e v_i para um vetor pertencente à $\mathbf{v}$.

O *lattice* físico uniforme é aquele no qual o espaçamento espacial h_r está vinculado às velocidades discretizadas. Com base no mencionado na seção 2.3, vetores de velocidades discretas sempre conectarão dois pontos de um lattice uniforme, ou seja, $(\mathbf{r} + \delta_t\mathbf{c}_i)$ será parte do conjunto vetorial espacial, sempre que r também for. A Figura 3.1 ilustra uma situação na qual o vetor posição r cumpre tal condição.

Se:

$$|\mathbf{c_i}|\delta_t = h_r \rightarrow |\mathbf{c_i}| = h_r/\delta_t \equiv c_r, \tag{3.34}$$

logo $c_r = |\mathbf{c_i}|$ e $\mathbf{c_i} = c_r\mathbf{c_i}^*$, implica em $\mathbf{c_i}^*$ ser uma velocidade adimensional discreta unitária ou $c_r = 1$ e $\mathbf{c_i} = \mathbf{c_i}^*$.

O *lattice* conceitual é derivado do espaço de velocidades adimensionais discretas $\mathbf{c_i} = c_r\mathbf{c_i}^*$ e nele o espaçamento espacial e o passo de tempo discreto da evolução temporal coincidem, definindo um espaço de velocidade unitário. A constante c_r é chamada velocidade de referência do *lattice*.

Assim:

$$\mathbf{c_i} = c_r\mathbf{c_i}^* = c_T\mathbf{v_i} \tag{3.35}$$

$$\mathbf{c_i}^* = \frac{c_T}{c_r}\mathbf{v_i} = \frac{1}{a_s}\mathbf{v_i} \tag{3.36}$$

E:

$$a_s = \frac{c_r}{c_T} = \frac{h_r/\delta_t}{\sqrt{k_b T_0/m}} \tag{3.37}$$

Como:

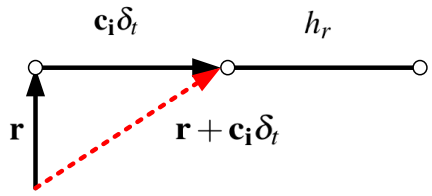
$$\mathbf{v_i} = a_s \mathbf{c_i}^* \rightarrow \mathbf{M_a} = a_s \mathbf{c_i}^* \tag{3.38}$$

Conclui-se que:

$$\mathbf{c_i}^* = \frac{1}{a_s} = \frac{\mathbf{M_a}}{a_s} \tag{3.39}$$

na qual $\mathbf{M_a}$ é o número de Mach vetorial, feita a consideração de que a velocidade térmica de referência coincide com a velocidade do som para gás ideal.

Figura 3.1 – Representação do *lattice* Físico. O vetor de velocidade discreta c_i sempre conectará dois pontos desse *lattice* uniforme de forma que $(\mathbf{r} + \delta_t \mathbf{c}_i)$. h_r e δ_t denotam o espaçamento espacial e o passo de tempo discreto da evolução temporal, respectivamente, e $c_r = h_r/\delta_t$ a velocidade de referência.



Fonte: Autor

3.1.3 Polinômios de Hermite Tensoriais e Tensores Monomiais

Um tensor genérico de ordem n e índices α_i terá representação: $\mathbf{T}^{(n)} \equiv \mathscr{T}^{(n)}_{\alpha_1 \cdots \alpha_n}$. Um tensor monomial, nessa notação, é definido como $\mathbf{M}^{(n)}(\mathbf{v}) \equiv \mathscr{M}^{(n)}_{\alpha_1 \cdots \alpha_n} := v_{\alpha_1} \cdots v_{\alpha_n}$ e sempre implica um argumento v. Ou seja, o seu termo geral tem formulação: $\prod_i^n v_{\alpha_i}, v_{\alpha_i} \in \mathbf{v}$. E os tensores polinomiais de Hermite $\mathbf{H}^{(n)} \equiv \mathscr{H}^{(n)}_{\alpha_1, \cdots \alpha_n}$, desenvolvidos por Grad, são fornecidos por

$$\mathbf{H}^{(n)}(\mathbf{v}) = \frac{(-1)^n}{\omega(\mathbf{v})} \nabla_v^{(n)} \omega(\mathbf{v}) \tag{3.40}$$

$$\omega(\mathbf{v}) = \frac{1}{(2\pi)^{D/2}} e^{-\left(v^2/2\right)}, \quad \nabla_v^{(n)} \equiv \partial_{\alpha_1} \cdots \partial_{\alpha_n} \tag{3.41}$$

Na qual ω é a função peso, D a dimensão espacial Euclideana e $v^2 = v_\alpha v_\alpha$. Lembrando que a repetição de um índice em um termo representará uma soma com respeito a esse índice no seu intervalo de variação, segundo convenção de somatório de Einstein (1916). Além disso, um tensor de Hermite polinomial de grau n, também será um polinômio de mesmo grau. Os tensores de Hermite com dimensão 2, isto é, nos quais $\mathbf{v} = (x, y)$ são:

$$\mathbf{H}^{(0)}(\mathbf{v}) = 1 \tag{3.42}$$

$$\mathbf{H}^{(1)}(\mathbf{v}) = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.43}$$

$$\mathbf{H}^{(2)}(\mathbf{v}) = \begin{bmatrix} x^2 - 1/2 & xy \\ xy & y^2 - 1/2 \end{bmatrix} \tag{3.44}$$

$$\mathbf{H}^{(3)}(\mathbf{v}) = \begin{bmatrix} \left[2x^3 - 3x, \quad 2x^2y - y\right] & \left[2x^2y - y, \quad 2xy^2 - x\right] \\ \left[2x^2y - y, \quad 2xy^2 - x\right] & \left[2xy^2 - x, \quad 2y^3 - 3y\right] \end{bmatrix} \tag{3.45}$$

3.1.4 Método das Abcissas Prescritas

De acordo com a seção há o intuito de assegurar a correspondência entre os momentos contínuos e discretos da função de distribuição de equilíbrio, que é uma condição para isotropia. No caso de um termo de colisão BGK isso se reduz a um problema de quadratura, tal que:

$$\sum_i^b w_{j(i)} \mathbf{M}_i^{(2n)} = \Delta^{(n)} = \int \omega(\mathbf{v}) \mathbf{M}^{(2n)} d\mathbf{v} \tag{3.46}$$

ou

$$\sum_i^b w_{j(i)} \mathbf{H}_i^{(n)} \mathbf{H}_i^{(n)} = \Lambda^{(n)} = \int \omega(\mathbf{v}) \mathbf{H}^{(n)} \mathbf{H}^{(n)} d\mathbf{v} \tag{3.47}$$

fornece os pesos $w_j(i)$ em função de abcissas prescritas c_i.

Além disso, combinações de produtos externos entre dos polinômios de Hermite Tensoriais até uma ordem N definida, atestará a ortogonalidade dos polinômios de Hermite Tensoriais no espaço discreto (MATTILA; HEGELE JR.; PHILIPPI, 2014). Segue que:

$$\sum_i^b w_{j(i)} \mathbf{H}_i^{(n)} \mathbf{H}_i^{(m)} = \begin{cases} 0^{(m+n)}, & n \neq m \\ \Lambda^{(n)}, & n = m, \end{cases} \qquad n + m \leq 2N + 1 \tag{3.48}$$

Philippi et al. (2006) demonstrou, em apêndice, que o produto interno ponderado entre dois polinômios de Hermite tensoriais, não necessariamente da mesma ordem, coincidirá tanto no espaço discreto, quanto contínuo. Esse resultado é indicado pela equação (3.47).

A Equação 3.48 codifica um número enorme de equações polinomiais escalares, a nível de componente, quando expandida, e não fornece nenhum guia sobre quais equações polinomiais escalares escolher nesse nível — isto é, sobre os pares n, m e componentes do tensor $\mathbf{H}^{(n)}\mathbf{H}^{(m)}$ —, nem mesmo se existem soluções 'ótimas', que minimizem o esforço computacional, reiterando a necessidade deste estudo, conforme previsto em 1.2.

A Tabela 3.1 ilustra a complexidade computacional de algoritmos de inspeção, que percorrem todos os casos possíveis no interior do produto tensorial. Seja **COMP** a quantidade total de componentes de todos os produtos $\mathbf{H}^{(n)}\mathbf{H}^{(m)}$, com $n, m \geq 0$ e $m + n \leq 2N + 1$ para um espaço Euclideano bidimensional ($D = 2$), então $COMP = N2^{2N+2} + 1$

(NAAKTGEBOREN, 2019), tal que $log_2(COMP-1) = 2N + log_2 N + 2$. Portanto, para um *lattice* de b velocidades prescritas e ordem N, a Equação 3.48 representará $b(N2^{2N+2}+1)$ polinômios em a .

A sequência descrita na terceira coluna da Tabela 3.1 corresponde à A014105 em The OEIS Foundation Inc. (2019), contabilizando o total de combinações entre n e m inspecionáveis.

Tabela 3.1 – Complexidade computacional da Equação 3.48, na qual **COMP** é a quantidade total de componentes de todos os produtos $\mathbf{H}^{(n)}\mathbf{H}^{(m)}$, com $n,m \geq 0$ e $m+n \leq 2N+1$ para um espaço Euclideano bidimensional ($D=2$); $COMP = N2^{2N+2}+1$ (NAAKTGEBOREN, 2019).

N	$2N+1$	$\mathbf{H}^{(n)}\mathbf{H}^{(m)}$	**COMP**	$log_2(COMP-1)$
0	1	3	1	-
1	3	10	17	4
2	5	21	129	7
3	7	36	769	9.585
4	9	55	4097	12
6	13	78	98305	16.585
8	17	105	2097153	21
10	21	136	41943041	25.322
12	25	171	805306369	29.585
14	29	210	15032385537	33.807
16	33	253	274877906945	38

3.2 Resolução dos *Stencils*

A solução implica em escolher combinações de ordens (m,n) de polinômios tensoriais de Hermite, de modo a calcular 3.48 até que a quantidade mínima de equações seja obtida, respeitando-se $m+n \leqslant 2N+1$. Ou seja, o caminho ótimo através um conjunto pontos m e $n \in \mathbb{Z}^+$ deve ser encontrado em um plano de combinações de $M \times N$.

E por caminho ótimo, entende-se aquele que demanda de menor complexidade computacional total nas etapas de identificação das equações L.I. e na resolução do sistema de equações lineares produzido. O diagrama, que representa o Plano de Produto Tensorial, está disposto na Figura 4.2.

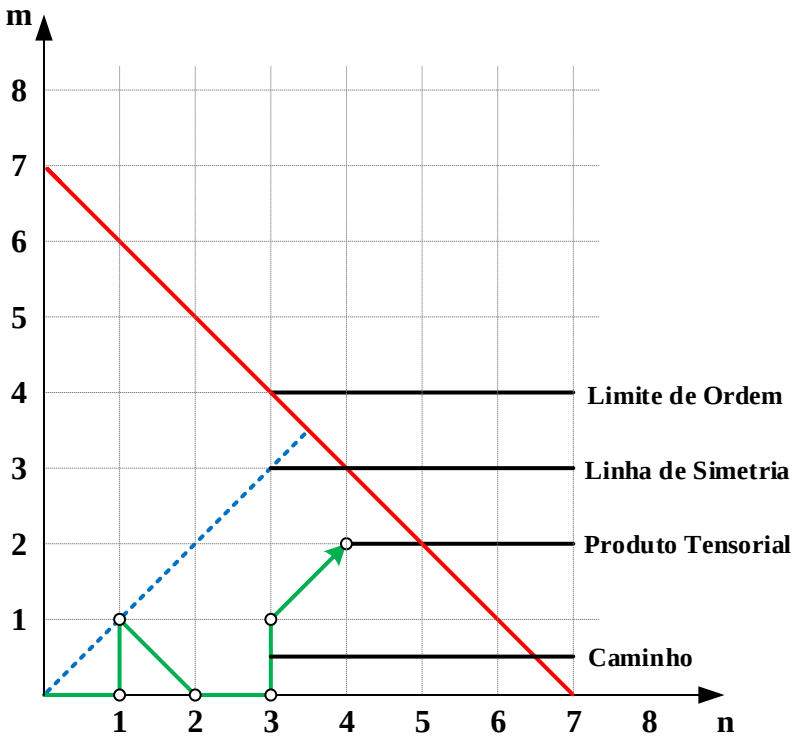
Inclue-se no cerne da presente atividade de pesquisa, buscar padrões que restrinjam a quantidade de pontos possíveis e, portanto, as combinações de trajetórias sobre este plano, de forma a agilizar a montagem do conjunto L.I. de equações.

A primeira vista, o rastreamento de padrões acontece com a experimentação de vários caminhos distintos sobre o plano, à medida que há incrementação da ordem dos *stencils* estudados. Resultados provenientes de investigação matemática são igualmente oportunos.

Algumas inferências já são iminentes, como a Linha de Ordem, consequência da desigualdade $n+m \leq 2N+1$, embutida na Equação (3.48), e uma Linha de Simetria, decorrente da simetria do produto tensorial face ao tensor de ortogonalidade, que torna desnecessária a investigação de produtos tensoriais de ambos os lados desta linha.

De início, procedem-se investigações sobre *stencils* de baixa ordem, manualmente, como o `D2Q9` e `D2Q17` — de 2ª e 3ª ordens, respectivamente —, a fim de um conhecimento preliminar sobre as expressões explicitas que compõem os sistemas, bem como para

Figura 3.2 – Plano de produto tensorial. As intersecções na grade quadriculada determinam coordenadas inteiras aplicáveis para n e m em 3.48. A linha escura indica o limite de ordem imposto por $m+n \leq 2N+1$ e a tracejada a linha de simetria na qual $m = n$. A região triangular limitada pelas linhas marca o conjunto de pontos inspecionáveis, nos quais é descrito um caminho.



Fonte: Autor

identificação de estratégias para aceleração dos algoritmos de solução.

A etapa seguinte é a de providenciar um código, em `SymPy` para indicar o caráter da equação - L.I., L.D. ou nula - e permitir o resgate de dados atrelados às coordenadas no plano $M \times N$. Uma modificação plausível é a de conseguir o output gráfico da indicação do caminho, destacando os pontos de interesse. Uma alternativa seria escrever um algoritmo em uma linguagem com melhor Benchmark, como `Julia` (JULIALANG, 2018).

3.3 Recursos Computacionais Apropriados

Realizar operações com *stencils* de alta ordem demanda o uso de linguagens de programação com suporte a pacotes de computação simbólica, de maneira que uma abordagem sem esse recurso seria não seria viável. Duas linguagens de programação foram cogitadas a serem utilizadas: `Python`^{TM} (PYTHON SOFTWARE FOUNDATION, 2018) v3.6.5 e `Julia` v1.0.1.

`Python` 3.6.5 apresenta a biblioteca simbólica `SymPy`, que possuí todos os recursos

necessários para a realização das operações com os tensores outrora definidos. Um programa não publicado, denominado `lbeST` (*lattice Boltzmann Equation Stencil*), feito para determinar conjuntos de pesos e escala de *stencils* já fora elaborado nessa linguagem.

`Julia` é uma linguagem de programação dinâmica de alto nível projetada para atender os requisitos da computação de alto desempenho numérico e científico. Contém um compilador just-in-time (JIT) baseado em LLVM, permitindo a ela se equiparar a C em performance nos testes de micro-benchmark. Outra vantagem dela, é a possibilidade de chamada para funções em outras linguagens, inclusive `Python`, com o uso do pacote `PyCall`.

O desenvolvimento em `Julia` sucede-se com a combinação de dois pacotes: o `SymEngine.jl` (SYMENGINE, 2018) e o `abstractAlgebra.jl` (NEMO, 2018). O primeiro, serve como complementação, encarregado de processar as rotinas inexistentes no segundo, com funcionalidade similar a do `SymPy`.

`SymEngine` é uma biblioteca de manipulação simbólica capaz de desenvolver operações de expanção, substituição e multiplicação matricial e, inclusive, diferenciação. O `abstractAlgebra`, por sua vez, tem ênfase em algebra abstrata e não suportando funções de diferenciação, nem números irracionais.

O atrativo para a utilização do `abstractAlgebra` está na rapidez em se trabalhar com operações de escalonamento matricial e resolução de sistemas lineares, atingindo ordens de grandeza temporais inferiores a do pacote `Sympy` e do próprio `SymEngine`.

Atualmente, esse pacote provê funcionalidades que atendem tópicos considerados pertinentes no âmbito desse projeto que são: aneis de polinômios genéricos espaços matriciais, operações precisas com números inteiros ou racionais e permutações.

3.4 Iteradores

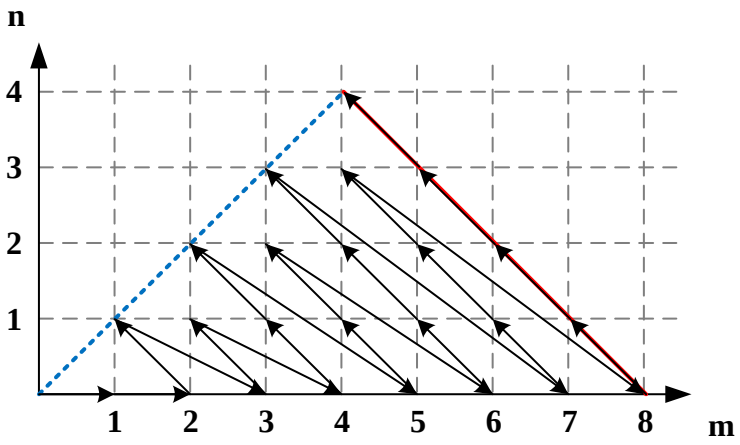
A inspeção do plano de produto tensorial é uma das etapas da resolução de *stencils* que possibilita grande economia de tempo, principalmente no estudo de *lattices* acima da sexta ordem. A fim de se delimitar os caminhos a serem percorridos no plano, são usados algoritmos de iteração, responsáveis por fornecer a informação do próximo ponto a ser inspecionado.

De início, serão analisados os desempenhos de iteradores pré-programados, que descrevem trajetórias comuns, e a partir desses exemplos, limitar formas mais eficientes que consideram a localização efetiva das equações linearmente independentes e descartem pontos com densidade muito baixa ou nula de sua incidência.

As coordenadas são descritas por caracteres alfa-numéricos, dentro da zona limitada por ordem e simetria do plano, com indicação maiúscula para pontos cartesianos com soma de coordenadas $n+m$ pares e minúscula para pontos com soma $n+m$ ímpar. Compõem o conjunto de iteradores pré-programados:

- *Bottom Up* (`'bottomUp'`): De baixo para cima ao longo das diagonais de valores constantes $n+m$, ou seja, sob uma mesma ordem N, percorrendo tanto pares coordenados de soma par quanto ímpar. O valor de $n+m$ é aumentado em um a cada vez que a diagonal principal $n=m$ é alcançada, até que a ordem N, relacionada a m e n por $n+m <= 2N+1$ atinja um valor máximo permitido. Este modo iterador é apropriado para verificações da ordem de precisão de determinado *stencil* LBE em relação a correspondência de produtos de tensores polinomiais de Hermite em espaços contínuos e discretos. É o único modo que passa por pontos $n+m$ ímpares do PPT, produzindo assim a sequência para N limite de 2 em:

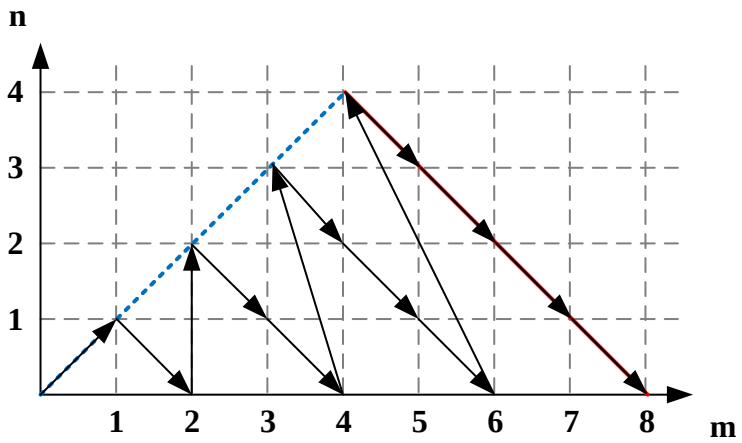
Figura 3.3 – `bottomUp`



Fonte: Autor

- *Successive Ranks* (`'succRank'`): Da diagonal principal $n=m$ para baixo, cobrindo níveis sucessivos de produto tensorial de valores constantes $n+m$, assumindo apenas valores pares de $n+m$ até que a ordem N atinja o valor máximo permitido para N. Este modo iterador pode ser usado para verificações de ordem de precisão de um determinado *stencil* LBE com respeito à correspondência de produtos tensoriais de Hermite no espaços discretos e contínuos, com a direta substituição no conjunto de pesos e escala na 3.48, sob o pressuposto de que os produtos classificados como ímpar são atendidos. A sequência para $N_{Max}=3$ é:

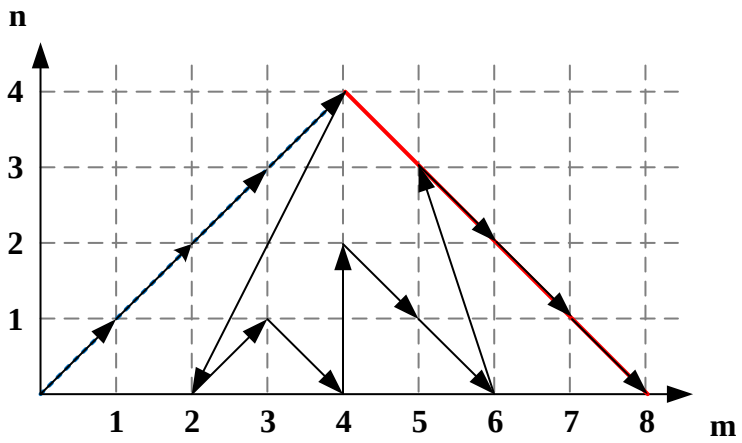
Figura 3.4 – `succRank`



Fonte: Autor

- *Diagonal Down* (`'diagDown'`): Viaja ao longo da diagonal principal $n = m$ até o valor máximo permitido para N e depois é direcionado para baixo, a partir da diagonal principal, cobrindo sucessivos ranks de produto tensorial de $n+m$ constante, de maneira similar à `'succRank'`. Descreve para $N_{Max} = 3$, a sequência: cobrindo primeiro uma diversidade de ranks de produtos tensoriais até o N_{Max} e então retorna para os ranks de produto tensorial inferiores, exaustivamente, e inicia o incremento de ordem novamente.

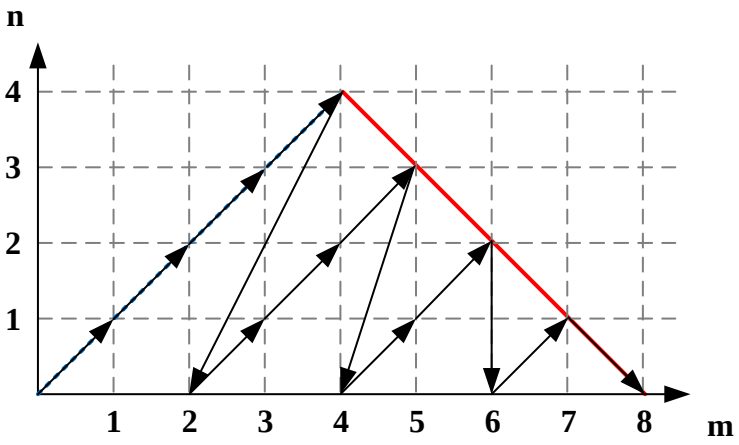
Figura 3.5 – `diagDown`



Fonte: Autor

- *Successive Diagonals* (`'succDiag'`): Itera ao longo de diagonais sucessivas, começando com a principal, $n = m$, subindo até o valor máximo assentido para N. Uma diversidade de ranks de produtos tensoriais é coberta, começando na diagonal principal $n = m$ em direção as diagonais inferiores. O percurso para $N_{Max} = 4$ fica:

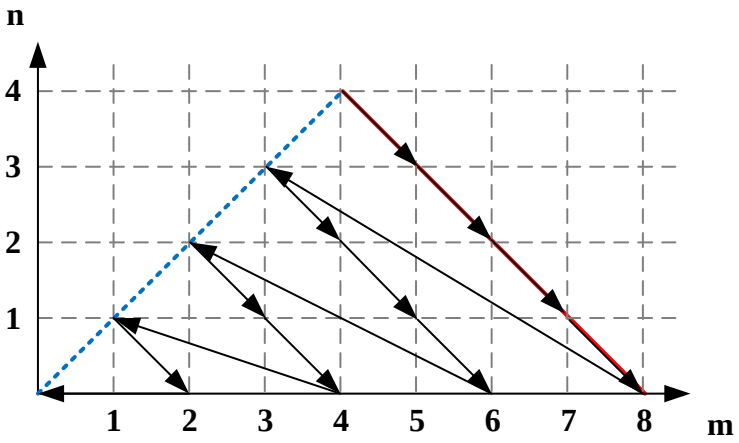
Figura 3.6 – `succDiag`



Fonte: Autor

- *Highest Order Down* (`'highOrdD'`): Caminha para baixo da diagonal principal ao longo de um nível, do mais alto ao mais baixo, assim compondo a sequência para $N_{Max} = 4$:

Figura 3.7 – `highOrdD`

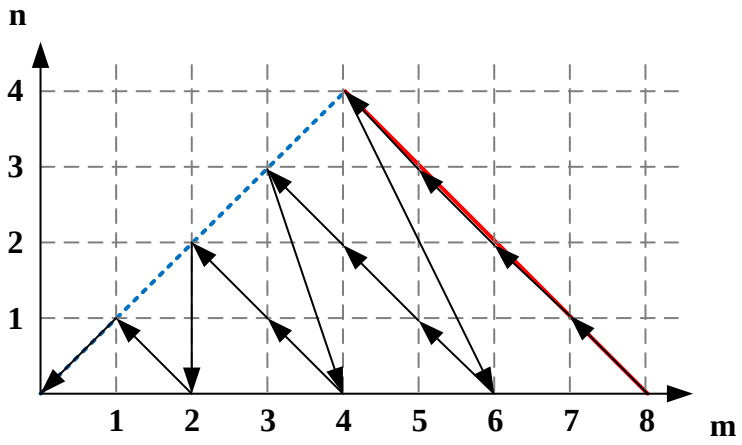


Fonte: Autor

Esse iterador primeiro perfaz da mais alta ordem em direção à mais baixa ordem.

- *Highest Order Up* (`'highOrdU'`): Desloca-se para a diagonal principal $n = m$ ao longo de um nível, do nível mais alto para o nível mais baixo, gerando assim a sequência para $N_{Max} = 4$:

Figura 3.8 – `highOrdU`



Fonte: Autor

Avança, assim como `'highOrdD'`, da mais alta em direção à mais baixa ordem.

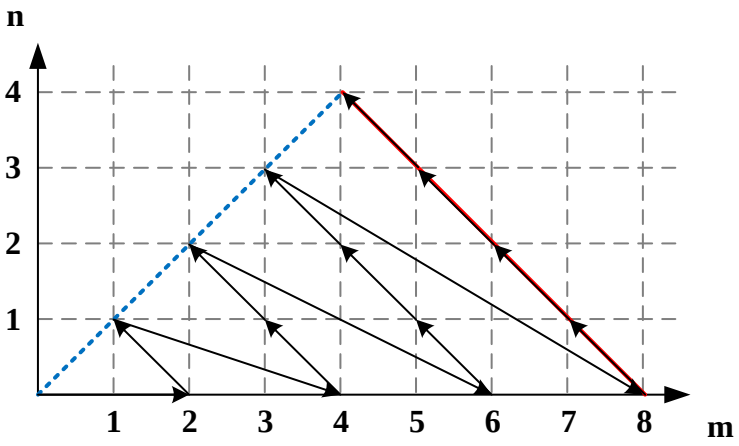
Os resultados apresentados na Sessão 4.3 revelam pistas para o traçado de caminhos que sejam melhores, ou seja, que descartem as coordenadas completamente indeterminadas, indicadas pelas entradas nulas das matrizes em `'wMNP'`, nas quais não há equações a acrescentar ao sistema. A mesma estratégia é usada na determinação do trajeto de busca para 'a' com a análise de `'aMNP'`. É esperado para o segundo caso que as candidatas tenham a localização definida sem influência da combinação de equações que compõe o sistema e, portanto, dos modos iteradores para w.

Esse resultado ajuda a comprovar a hipótese de que o subespaço gerado por vetores em 'a' formados pelos coeficientes do sistema linear é sempre o mesmo, independente da escolha feita para os seus elementos.

A fim de melhor classificar os modos, convenções de nomenclatura e atribuição de variáveis foram adotadas, a saber: a hipótese de $p = 0$, isto é, a trajetória sempre avançará por pontos com soma $n+m$ par a menos que versado sobre isso. Também as coordenadas padrão de início e término das iterações respectivamente serão $n = 0$ e $m = 0$, $n = 0$ e $N = N_{Max}$, sempre que implícitas as descrições. E por fim, a numeração de dois ou superior ao final da nomenclatura indica as variações fatoriais do modo. Dessa forma, os novos iteradores são:

- *Even Bottom Up* (`'evenBU'`): Descreve tragetória similar a *Bottom Up Mode*, percorrendo apenas coordenadas com soma $n+m$ par.

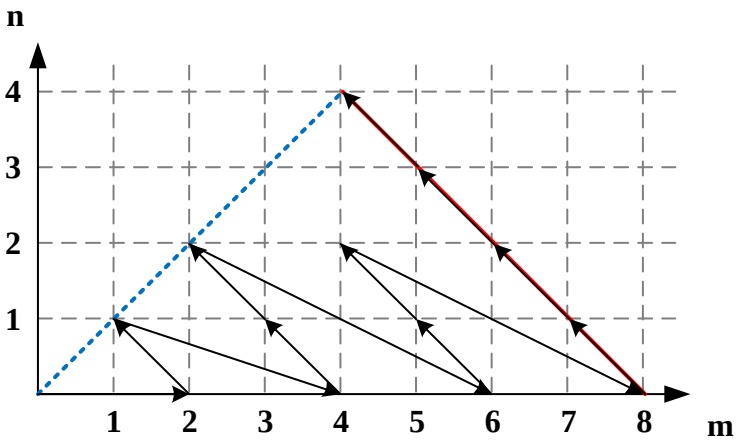
Figura 3.9 – `evenBU`



Fonte: Autor

- *Even Limited Bottom Up* (`'evenLimBU'`): Iterador com n limitado a um valor n_{Max} tal que $n_{Max} < N$, percorrendo o caminho de maneira idêntica a `'evenBU'`, até a condição, $n = n_{Max}$ ser cumprida, momento em que ocorre o incremento de ordem $N = N + 1$. Ao chegar em $N = N_{Max}$, a restrição é cancelada e todos os pontos até $n = N$ são inspecionados. Sempre que n_{Max} não for mencionado, será convencionado que $n_{Max} = 2$.

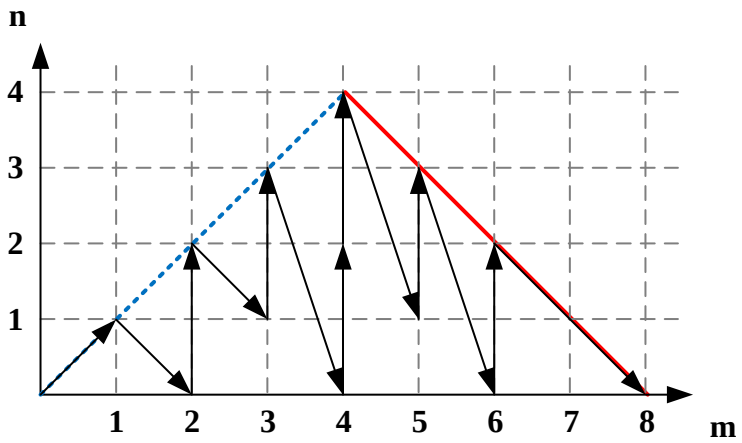
Figura 3.10 – `evenLimBU`



Fonte: Autor

- *The Sky* (`theSky`): Atua verticalmente sobre o PPT, fixado um valor para m, com incrementos em n até $N = N_{Max}$, quando $m_i = m_{i+1} + 1$. O critério de parada é a $N = N_{Max}$ e $n = 0$.

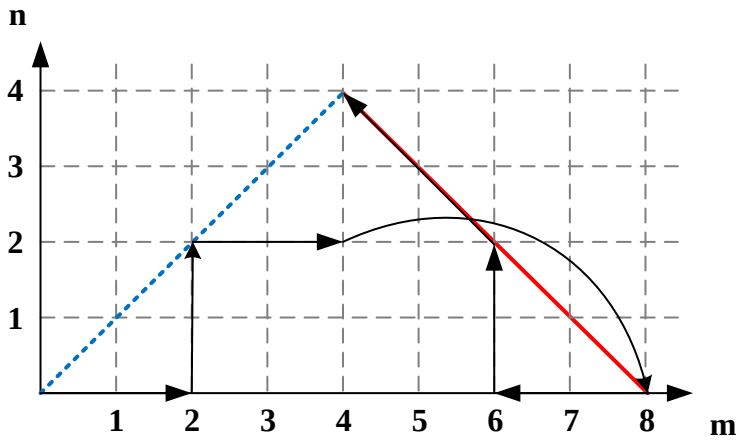
Figura 3.11 – `theSky`



Fonte: Autor

- *Limited by Triangles* (`'limTri'`): Avança sobre a diagonal principal e aquela em que $m = 2$ está disposta de maneira a performar um aumento gradativo de N, assim formando triângulos retângulos cujos catetos são marcados pela trajetória percorrida e as hipotenusas pela linha de simetria. Nessa modalidade, o ponto $n = N_{Max} - 2$, $N_{Max} = N_{Max} - 2$ é o último a limitar de triângulos antes do algoritmo começar a percorrer um caminho semelhante, o qual é projetado no limite de ordem após um espelhamento em relação a $m = N_{Max}/2$, a linha vertical média do PPT.

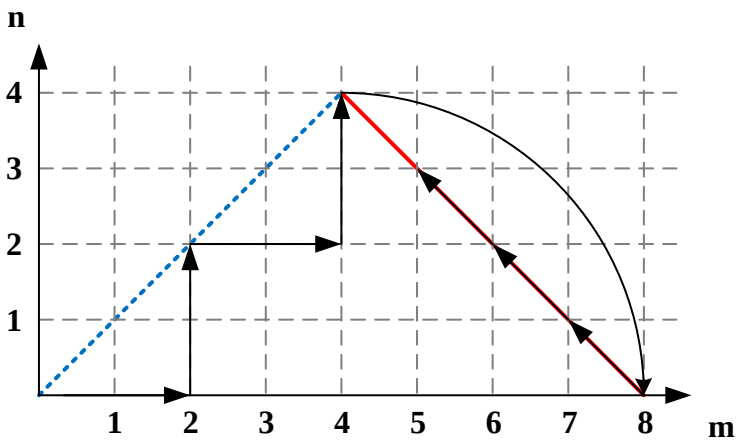
Figura 3.12 – `limTri`



Fonte: Autor

- *Limited by Triangles 2* (`'limTri2'`): Nessa modalidade, os triângulos são limitados até $n = N_{Max}/2$, $N_{Max} = N_{Max}/2$ e então continua percorrendo inteiramente o limite de ordem de baixo para cima.

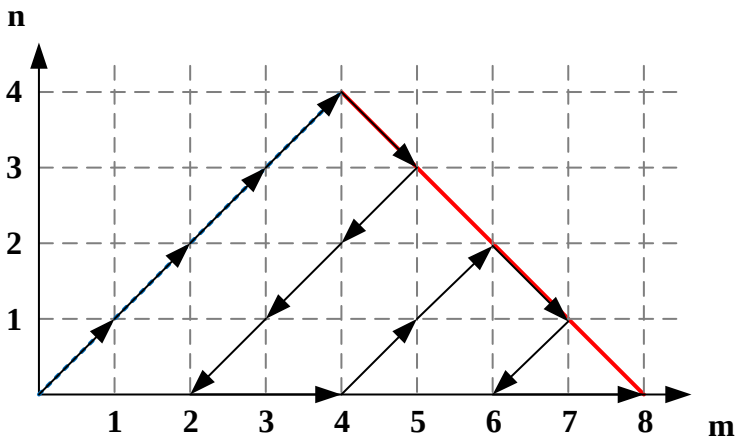
Figura 3.13 – `limTri2`



Fonte: Autor

- *Alternate Diagonals* (`alterDiag`): Variante de *Successive Diagonals* que faz as diagonais serem trilhadas de forma alternada. Ao invés de iniciar a diagonal após a diagonal principal em $n = 0$, o iterador avança para seu ponto em N_{Max} e decresce n, reiniciando a sequência.

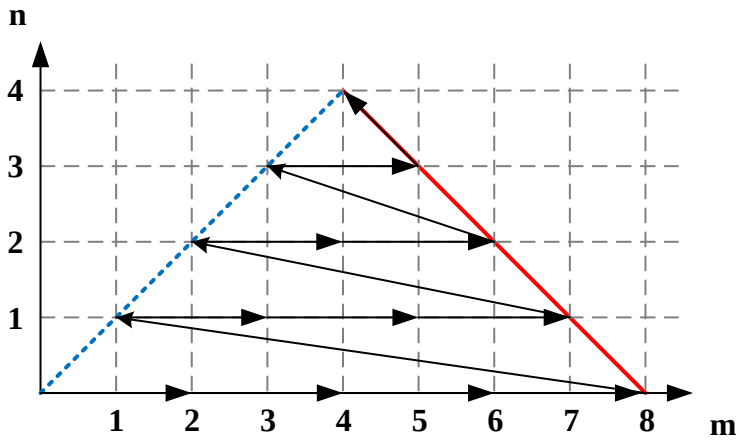
Figura 3.14 – `alterDiag`



Fonte: Autor

- *Horizontal Up* (`horizonUp`): Avança horizontalmente sobre o PPT com incrementos unitários em N, até $n = N_{Max}$.

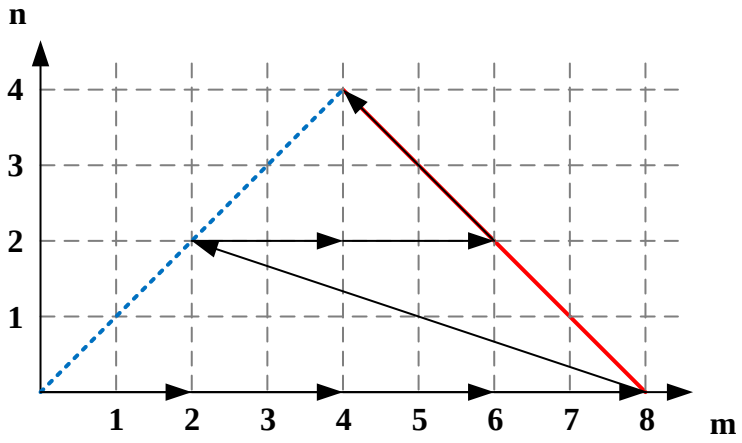
Figura 3.15 – `horizonUp`



Fonte: Autor

- *Horizontal Up 2* (`horizonUp2`): Progride horizontalmente sobre linhas as quais n é par até que $n = N_{Max}$ seja observado.

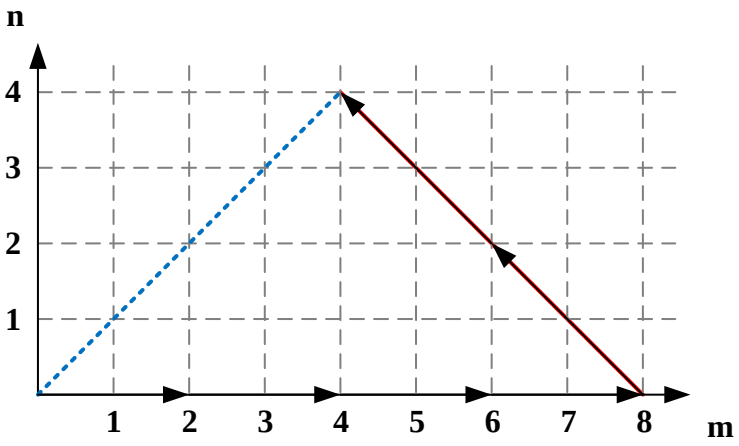
Figura 3.16 – `horizonUp2`



Fonte: Autor

- *Horizontal Up 3* (`horizonUp3`): Descreve um passo horizontal apenas na linha de base e quando chega em $n = 0$, $N = N_{Max}$, sobe através do limite de ordem, abstendo-se de coordenadas com n ímpar até $n = N = N_{Max}$.

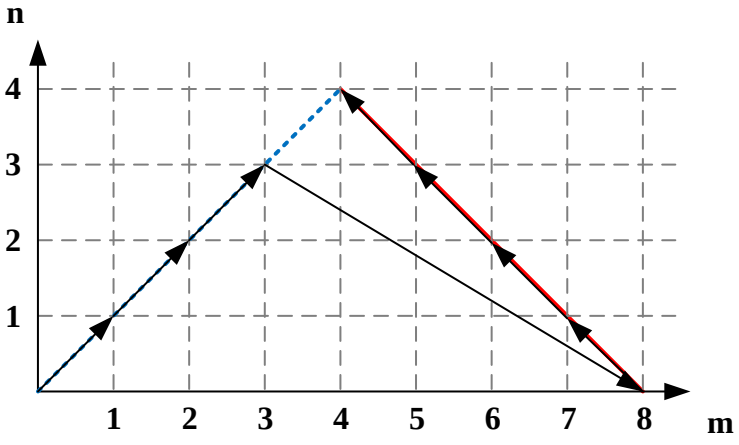
Figura 3.17 – `horizonUp3`



Fonte: Autor

- *In Line* (`inLine`): Itera sobre a diagonal principal ($n = m$) até $N = N_{Max} - 1$ e depois parte de $n = 0$ e $N = N_{Max}$ sobre a secundária em direção a diagonal principal. Esse modo tende a detectar o conjunto L.I., antes da exaustão. Se uma trajetória maior for requisitada, ele progredirá sobre as diagonais subsequentes interiores, partindo dos pontos $m = 2$, $n = 0$ e $m = 2(N_{Max} - 1)$, $n = 0$, conservando a mesma configuração.

Figura 3.18 – `inLine`



Fonte: Autor

3.5 Coleta de Dados e Parâmetros Analisados

A coleta de dados foi realizada por um código auxiliar, responsável por armazená-los em arquivo de extensão `.pickle`. Dentro dele, os dados estão organizados na forma de uma cadeia de dicionários iniciada com a variável declarada `DATA[...]`, no qual as informações das simulações são vinculadas aos modos iteradores através de um conjunto de chaves. O código auxiliar foi montado junto à um programa solução de *stencils* LBE no ambiente computacional interativo Jupyter Notebook, formalmente conhecido conhecido como IPython Notebook, capaz de combinar a execução de código a saídas de texto, equações matemáticas, gráficos e demais conteúdos de mídia do gênero. A esta combinação na extensão *.ipynb* será concedido o nome de plataforma de testes de iteradores LBE.

Dois *stencils* de 5ª e 6ª ordem, outrora apresentados na literatura de Mattila, Hegele Jr. e Philippi (2014) como D2V49 e D2V81, respectivamente, e com seu conjunto de pesos e escala previamente calculados, serão usados na plataforma. A preferência por *lattices* de ordem superior é justificada pela maior quantidade de pesos intrínsecos ao conjunto velocidades discretas, ao qual a quantidade de equações L.I. é numericamente equivalente a fim de se compor um sistema linear possível e determinado. Se um número maior de equações L.I. é requisitado, mais fácil fica a identificação de padrões sobre plano de produto de produto tensorial.

Ambos os *stencils* foram solucionados pelo programa, ao menos uma única vez, para apuração da compatibilidade com os resultados de Mattila, Hegele Jr. e Philippi (2014) e para realização de teste de verificação de ordem de precisão com um limite tolerado de até 10^{-9} na diferença, em módulo, entre os valores obtidos para os dois lados da igualdade em 3.48. Os resultados pertinentes foram alocados na Tabela 4.7.

A verificação de ordem também é função do programa, o qual será mantido em suas configuração padrão, que define `'bottomUp'` para delimitação do caminho. Vale ressaltar que, após a escolha de um ponto do PPT, a Equação 3.48 é computada para todas as combinações possíveis entre os componentes dos tensores $\boldsymbol{H}^{(m)}$ e $\boldsymbol{H}^{(n)}$ polinomiais de Hermite, excluso os elementos simétricos e portanto, iguais, em um mesmo tensor. Também será conservado invariável durante o procedimento de solução os parâmetros de entrada:

- `mode: (default = 'full')`: Aceita os valores válidos `['full','simp','modl']`, controlando quando a expressão da solução para os pesos, que se torna incondicionalmente extensa em quantidade de caracteres e grau polinomial com o acréscimo da ordem do *lattice*, deve ser adotada em sua forma cheia `full` (não simplificada), simplificada - pelo módulo `sympy.simplify()` do pacote de computação simbólica *sympy*, algo moroso, em especial para exemplares de alta ordem - ou modelados como polinômios de alto grau em intervalos para o fator de escala em que todos os pesos são simultaneamente positivos.

- `meth['w']`: Abrange os valores válidos `['LU', 'gen']` e determina a estratégia do programa usada para resolver o sistema de equações lineares para os pesos, com o fator de escala interpretado como uma contante. A opção `'LU'` habilita a escolha do método `LUsolve()` do `SymPy`, que é mais robusta e mais rápida indicação para sistemas maiores de equações lineares, embora gere expressões de complicada simplificação.

- `meth['a']`: Estabelece a estratégia de solução usada para a equação do fator de escala do *lattice* com os pesos, anteriormente definidos como expressões de 'a', aplicados à $a_E q$, dentre as opções `['num','ana']`. A seleção da opção `['num']` faz o programa escolher um método numérico para a equação. Em *lattices* de baixa ordem é usualmente rápido e assim permanece conforme a ordem aumenta, superando abordagens analíticas. Usualmente, é a única escolha viável para *lattices* de média e alta ordem e também a opção mais razoável para expressões de pesos solucionados por modelagem polinomial ou sem simplificação.

Apesar da solução numérica ser destinada para cálculos em alta ordem, a função de empacotamento (*wrapper*) simbólico, que converte uma expressão simbólica de uma variável a em uma função voltada à substituição de valores numéricos, tem sua ação limitada a modelos abaixo da oitava ordem. A restrição é uma possível consequência da atuação coligada dos módulos `symExpr.subs(var: x).evalf()` na substituição das instâncias e posterior avaliação numérica no interior da função. Durante a simulação de D2V141 `'f(x)'` não chegou a ser encaminhada ao solver numérico, demandando a interrupção forçada do Kernel.

Em acordo com as recomendações previstas na própria documentação do programa, sumariamente descrita acima, serão estipulados os parâmetros de entrada como:

```
(meth = {'w': 'LU', 'a': 'num'}, mode = 'full')
```

3.5.1 Cache

O programa contêm uma opção de cachê, encarregada de armazenar alguns objetos selecionados durante o processo de solução a fim de que o acesso ao bloco de memória temporário agilize as operações, reduzindo a recomputação de saídas. O cachê é integrado a uma base de dados `sqlite3` com uma interface de dicionário integrada. Os objetos são serializados e codificados em texto em base64 para serem armazenados nessa base.

Uma das sincronizações efetuadas no cachê afeta a duração do processo de iteração sobre o PPT, proporcionando um decréscimo no tempo da operação em casos de média e alta ordem estimada em 50%. Em situações de baixa ordem, computar os polinômios tensoriais de Hermite é, em geral, mais rápido do que resgatar a informação do BD. Como

a atuação do cachê atua sobre o desempenho dos modos iteradores ensaiados, duas coletas de dados serão empreendidas, compreendendo-o ligado e desligado no decorrer do processo.

Atualmente, são submetidos ao cachê funções encarregadas do somatório em **??** para um *lattice* e tensor de Hermite simbólico em uma combinação de índices especificados. As funções que retornam componentes únicos, índice identificado, de um tensor de Hermite polinomial a uma dimensão espacial `DIM` são armazenadas da mesma forma. A sincronização de cachê sobre essas funções é o que promove a otimização.

3.5.2 Verificador de Independência Linear

O mecanismo padrão de verificação de equações linearmente independentes é incremental, ou seja, tomadas em relação as equações anteriormente armazenadas para o sistema, de maneira a ser compatível com uma estratégia de varredura. No programa-padrão a checagem ocorre a partir da função `.areRowsLI(M)`, que retorna quando as linhas de uma `sympy.Matrix(M)`, com coeficientes polinomiais em a, cumprem a condição de independência. Dentro do módulo, acontecem substituições sucessivas da escala 'a' por inversos dos primos sucessivos, correspondentes aos dois últimos integrantes do 38º quadruplet, números k tais que k, $k+2$, $k+4$, $k+6$, $k+8$ são todos primos, conforme a sequência ID: A007530 (THE OEIS FOUNDATION INC., 2019). Essa é uma operação necessária, à proporção que testes simbólicos sobre a matriz de coeficientes usualmente falham, visto a incapacidade do `sympy` reportar divisões por zero, quando esse zero ocorre como uma função complicada de 'a'.

Infelizmente o uso dos inversos dos primos produzem coeficientes muito pequenos, à medida que o grau das expressões polinomiais em 'a' cresce, evoluindo a valores próximos à 10^{-25}, que são indistinguíveis de zero em métodos de decomposição matricial para revelação de rank. No caso da biblioteca `numpy.linalg.matrix_rank`, a deficiência de rank é detectada em um teste de magnitude de valores singulares de M após a decomposição $M = U\Sigma V^*$, descartando, por padrão, valores inferiores a $S.max() * max(M.shape) * \varepsilon$, em que $S.max()$ é o máximo valor singular de S, $max(M.shape)$ a maior dimensão da matriz e ε uma margem de tolerância.

Outras estratégias, tais como depomposição LU, QR, determinantes de submatrizes de M e decomposição interpolativa (ID) foram utilizadas com o intuito de evitar executar o pivoteamento sobre tipos racionais, que são computacionalmente mais desvantajosos do que números de ponto flutuantes. No entanto, nenhuma das alternativas se mostrou qualificada a revelar o ranking de M numericamente.

A única estratégia efetiva é a atualmente incorporada no programa padrão e consiste justamente no pivoteamento de uma matriz com coeficientes racionais para a forma escalonada reduzida por linhas, a partir da função `sympy.rref()`. A aprovação de uma equação conforme fica condicionada a presença de elemento não-nulo remanescente sobre a diagonal principal. Se a resultante é a matriz identidade, então a equação é

aproveitável. Embora seja uma abordagem válida de inspeção, ela desperdiça equações linearmente independentes, caso o elemento não nulo apareça fora da diagonal principal, destacando erroneamente a contagem de equações L.I. por nó do PPT.

O exercício de um critério que considera todas as linhas da matriz escalonada com elementos não nulos abre espaço para uma nova investigação sobre os iteradores. Então serão coletados ao todo, quatro séries para os dados padronizados na Subseção 3.5.4, que contemplem os cenários previstos na presente Subseção e em 3.5.1, denominados:

- `'nc-nct'`: Cachê desligado e verificação de elemento não nulo apenas na diagonal principal;
- `'c-nct'`: Cachê ligado e verificação de elemento não nulo apenas na diagonal principal;
- `'nc-ct'`: Cachê desligado e verificação por linha não nula;
- `'c-ct'`: Cachê ligado e verificação por linha não nula.

3.5.3 Máquina e Sistema Operacional

As simulações serão conduzidas em um computador com as seguintes características:

- Sistema operacional Windows 10 Pro versão 10.0.17134 64-bit, processador com base em 64x.
- Processador Intel(R) Core(TM) i5-2450M CPU @ 2.50 GHz com Turbo Boost até 3.1 GHz
- Memória RAM Instalada 4.00 GB (3.82 GB Aproveitável) E com implementação em python CPython versão 2.7.13 em um ambiente virtual para python gerenciado por Anaconda versão 4.5.12.

3.5.4 Dados Coletados

Foram coletados e armazenados no dicionário `DATA[key]` para vir a compor uma série de critérios avaliação e de auxílio na produção de novos algoritmos de iteração, sendo a listagem das chaves: `'wTime'`, `'MNP[w]'`, `'KY'`, `'AK'`, `'MNP[a]'`, `'normT'`, `'getCOL'`, `'dEq'`, `'Time'` endereçadoras do seguinte conteúdo:

- `'wTime'`: Arquiva o tempo de execução total da rotina de iteração até a montagem do sistema linear com coeficientes recebidos de um dicionário de equações `ED`, medido pelo módulo `time.time()` em segundos, como um número de ponto flutuante e com precisão mínima de 1 s.

- `'MNP[w]'`: Arquiva em `numpy.array()` de dimensões $(2ORD+1, 2ORD+1)$, sendo ORD a última ordem investigada ou a aquela que fora atribuída entre os argumentos do solucionador, uma representação matricial do plano de produto tensorial. Os índices de um respectivo elemento da matriz equivalem as n x m coordenadas de um ponto do plano e sua entrada indica a soma de expressões LI coletadas para o sistema linear dos pesos naquela localização.

- `'MNP[a]'`: Arquiva em estrutura matricial semelhante à `MNP[w]`, entradas que indicam a quantidade de candidatos para equação de a em cada coordenada do plano.

- `'KY'`: Guarda a lista de chaves binárias que identificam os termos dos dois tensores envolvidos no produto que determinou um componente linearmente do sistema para os w's. Em um ponto especificado do plano, que equivale à 3.48 para n e m, aplicada a um conjunto discreto de velocidades, existem $(m/2+1)*(n/2+1)$ possibilidades a serem verificadas para m e n par e $((m-1)/2+1)*((n-1)/2+1)$ se são ímpares, cada uma atrelada à chave com formato exemplificado por `'0011'.'000111'` produto entre tensores de ordem $n=4$ e $m=5$, ao qual o primeiro componente é gerado por duas derivações parciais em x: `'11'` e duas em y: `'00'` da Fórmula de Rodrigues.

- `'AK'`: Listagem de chaves associadas a contagem registrada em `'MNP[a]'`, idêntica à `'KY'`.

- `'normT'`: Armazena o cálculo do tempo normalizado em relação a menor média aritmética dos tempos de simulação de um dos iteradores. A normalização de tempo é tal que o gerador de caminho `PATH` mais rápido recebe `DATA['normT']['PATH'] = 1.0`.

- `'getCOL'`: Captura em tuplas a contagem parcial de $(LI, T_Eq)_P$, após varredura de 12 itens. Os dados presentes aqui serão utilizados na plotagem de um gráfico LI_Eq x T_Eq para cada trajetória testada.

- `'dEq'`: Reporta a densidade final de equações LI sobre o caminho percorrido, que é o quociente da última tupla presente em `'getCOL'`.

- `'Time'`: Registra o tempo de execução no qual cada representante L.I. foi descoberto e os insere em uma lista. `'Time'` em reunião com o primeiro índice das tuplas em `'getCOL'` resultarão em gráficos LI x t para as amostras, exteriorizando a densidade de expressões válidas relativas ao tempo.

3.5.5 Parametros Analisados

A razão LI/T_{Eq} entre o número de equações LI e T_{Eq} total de equações inspecionadas, junto com o *Benchmark*, medido no tempo de execução, do início até o final do processo de

descoberta de equações e formação do sistema linear são tidos como critérios de avaliação de performance de um modo iterador.

Dessa forma, um caminho é avaliado ótimo em relação a esses parâmetros quando tem potencial assimilar equações com LI/T_{Eq} o mais próximo de 1.0 e no menor tempo possível.

Em tese, 100% de densidade de equações linearmente independentes não é capaz de ser alcançada nas condições atuais do programa, devido a inspeção completa de todas as combinações possíveis entre os componentes originados do produto entre os dois polinômios tensoriais Hermite.

Investigações conduzidas sobre o PPT, que constam como resultado desse trabalho, revelam que a quantidade média de equações a serem extraídas em cada ponto do plano é de cerca de 3, para nós distribuídos em cada uma das $N-1$ ordens, que precedem a N$^{\underline{a}}$ ordem do *lattice* , na qual LI/T_{Eq} por nó tem um considerável aumento. No entanto, é errôneo pensar que iniciar trajetórias a partir da mais alta ordem conduzirá melhor desempenho, haja vista que o tempo de execução sobe com o acréscimo de complexidade das expressões, tornando alguns caminhos desferidos completamente em N para 6$^{\underline{a}}$ e 8$^{\underline{a}}$ ordens praticamente impossíveis.

3.6 Lattices Testados

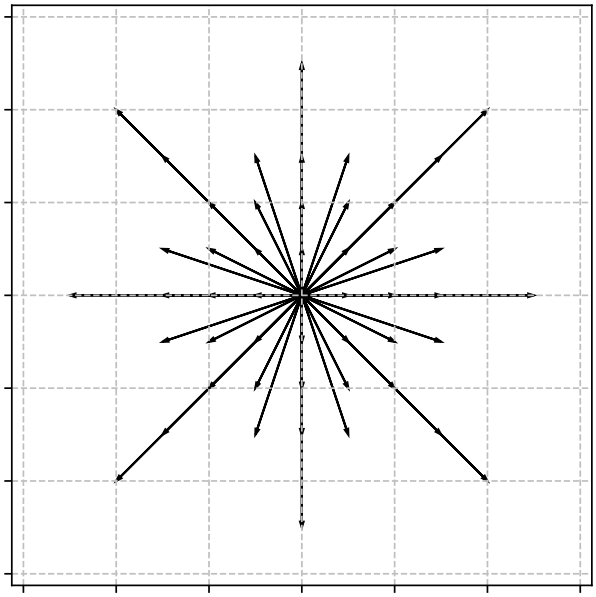
No total três *lattices*, representados junto aos seus conjuntos de pesos e escala derivados de Mattila, Hegele Jr. e Philippi (2014), e um inédito formulado pelo autor serão objeto de simulação no programa para a documentação do desempenho dos modos iteradores elaborados. São, respectivamente, os exemplares: `D2V49`, `D2V81`, `D2V141` e `D2V169`.

A necessidade de implementar rotas em mais de um *stencil* LBE, com ordens consecutivas, atende ao objetivo de comprovar o fato de que se os modos iteradores forem ranqueados pelo seu desempenho, então esse ranking será quase monótono em relação a distribuição vetorial das velocidades discretas e a ordem. `D2V141`, de 8$^{\underline{a}}$ ordem, por sua vez, tem a finalidade de examinar se as disparidades de desempenho entre os integrantes desse ranking, aumentará proporcionalmente a N.

Em termos geométricos, `D2V49` exibe vetores com módulos maiores sobre os eixos cartesianos, como também sobre as linhas inclinadas à 45$^{\underline{o}}$, que são as diagonais do plano. Esse comportamento não é presenciado nas amostras `D2V81` e `D2V141`, nos quais os vetores de grande magnitude estão dispostos sobre os eixo ou próximos ao centro dos octantes. Os grupos $(5a,0)$ e $(4a,4a)$, despontam duas maiores normas para `D2V49`, enquanto $(5a,0)$ e $(5a,2a)$ para `D2V81`, mais $(7a,0)$ e $(7a,2a)$ para `D2V141`.

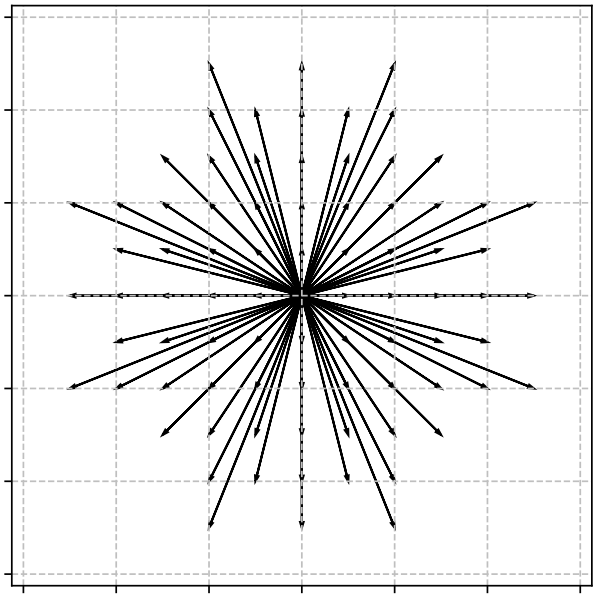
As Figuras 3.19, 3.20, 3.21 e 3.22 expõem a representação gráfica dos *stencils*, nos quais a provém da literatura de Mattila, Hegele Jr. e Philippi (2014):

Figura 3.19 – *Stencil* `D2V49`. Conjunto de 5ª ordem com $b = 49$ velocidades discretas simetricamente distribuídas sobre grade com dimensão de $2a = 2.297464497677078$ unidade e com velocidades recorrentes por octante.



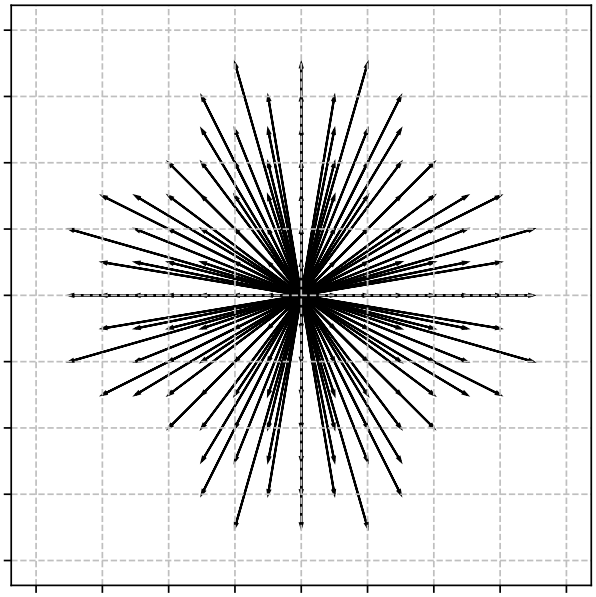
Fonte: Autor. Concebido pelo programa `lbeST`.

Figura 3.20 – *Stencil* `D2V81`. Conjunto de 6ª ordem com $b = 81$ velocidades discretas simetricamente distribuídas sobre grade com dimensão de $2a = 1.94001699747879$ unidade e com velocidades recorrentes por octante.



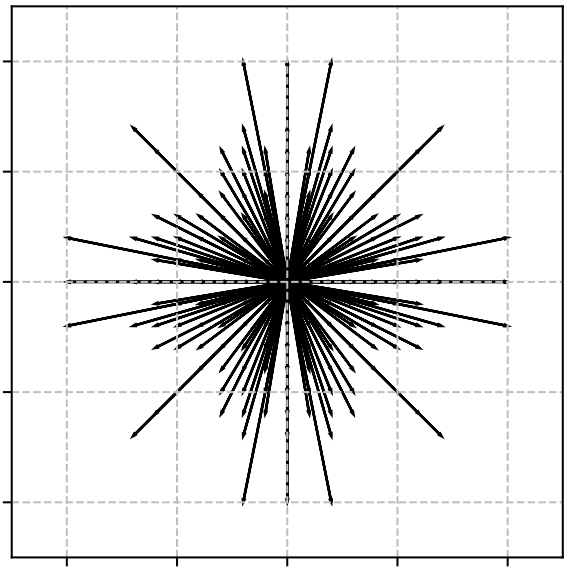
Fonte: Autor. Concebido pelo programa `lbeST`.

Figura 3.21 – *Stencil* `D2V141`. Conjunto de 8ª ordem com $b = 141$ velocidades discretas simetricamente distribuídas sobre grade com dimensão de $2a = 1.673840810860705$ unidade e com velocidades recorrentes por octante.



Fonte: Autor. Concebido pelo programa `lbeST`.

Figura 3.22 – *Stencil* `D2V169`. Conjunto de 9ª ordem com $b = 169$ velocidades discretas simetricamente distribuídas sobre grade com dimensão de $2a = 1.641561263219186$ unidade e com velocidades recorrentes por octante.



Fonte: Autor. Concebido pelo programa `lbeST`.

Capítulo 4

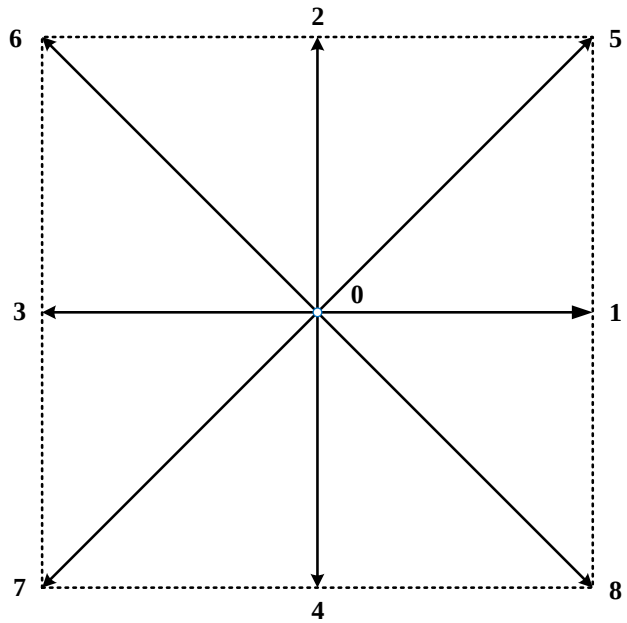
Resultados e Discussões

4.1 Resolução Manual do *Stencil* D2Q9

O *stencil* D2Q9 será resolvido a partir de (2.11) e validado através de um conjunto de pesos $w_i(j)$ e escala a_s documentados em Philippi et al. (2006). Por conveniência, admite-se um vetor velocidade $\mathbf{c_i} = (x_i, y_i)$ na geração dos polinômios e posterior aplicação nos vetores de velocidade discreta do *lattice*. Os polinômios de Hermite serão logrados por meio do código disposto no apêndice A.1 de (ANDRADE, 2016).

Seja D2Q9 um *stencil* com nove velocidades e abcissas prescritas de acordo com a Figura 4.1.

Figura 4.1 – *Stencil* D2Q9



Fonte: Autor.

Três grupos de velocidades simétricas radialmente estão presentes de maneira que

cada grupo possuí um peso de quadratura similar. Isto é, $w_i = w_{j(i)}$, j como função de i. No caso considerado $j = 0,1,2$. As velocidades discretas unitárias correspondentes e seus índices de peso estão inseridos na Tabela 4.2.

Tabela 4.1 – Numeração das velocidades discretas unitárias e seus respectivos índices de peso.

i	$\mathbf{c}_i^*$	j
0	$(0,0)$	0
1	$(1,0)$	1
2	$(0,1)$	1
3	$(-1,0)$	1
4	$(0,-1)$	1
5	$(1,1)$	2
6	$(-1,1)$	2
7	$(-1,-1)$	2
8	$(1,-1)$	2

Fonte: Autor.

O caminho selecionado atravessa as coordenadas: $(0,0),(0,1),(1,1)$ e $(0,2)$. Observa-se que $(m,n) \equiv (n,m)$ na relação de ortogonalidade (2.11) pela condição de simetria dos polinômios de Hermite. O procedimento de cálculo acompanha a progressão assinalada na Figura 4.2.

Assim, o número de equações linearmente independentes requisitadas para solução é $l = 4$ com o acréscimo de a_s.

Quando $m = 0$ e $n = 0$:

$$\sum_i w_i H^{(0)} H^{(0)} = \Lambda^{(0)} \tag{4.1}$$

$$\sum_i w_i \cdot 1 = 1$$

A equação 4.2 é identificada:

$$w_0 + 4w_1 + 4w_2 = 1 \tag{4.2}$$

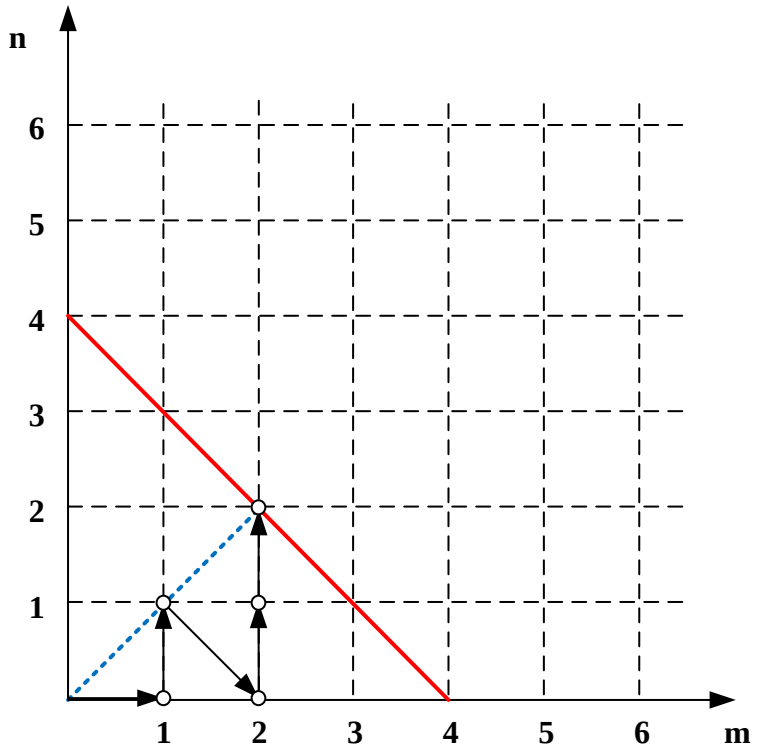
Quando $m = 1$ e $n = 0$:

$$\sum_i w_i H^{(1)} H^{(0)} = 0^{(1)}$$

Após o produto tensorial:

$$\sum_i w_i \begin{bmatrix} x_i \\ y_i \end{bmatrix} \cdot 1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Figura 4.2 – Trajetória percorrida no plano de produto tensorial.



Fonte: Autor

E com a expansão dos termos:

$$w_0 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + w_1 \begin{bmatrix} a \\ 0 \end{bmatrix} + w_1 \begin{bmatrix} 0 \\ a \end{bmatrix} + w_1 \begin{bmatrix} -a \\ 0 \end{bmatrix} + w_1 \begin{bmatrix} 0 \\ -a \end{bmatrix} + \tag{4.3}$$
$$w_2 \begin{bmatrix} a \\ a \end{bmatrix} + w_2 \begin{bmatrix} -a \\ a \end{bmatrix} + w_2 \begin{bmatrix} -a \\ -a \end{bmatrix} + w_2 \begin{bmatrix} a \\ -a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Chega-se a uma equação indeterminada:

$$w_0 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 4w_1 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 4w_2 \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$0 = 0$$

Quando m = 1 e n = 1:

$$\sum_i w_i H^{(1)} H^{(1)} = \Lambda^{(2)}$$

$$\sum_i w_i \begin{bmatrix} x_i \\ y_i \end{bmatrix} \cdot \begin{bmatrix} x_i \\ x_i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

O produto tensorial é:

$$\sum_i w_i \begin{bmatrix} x_i^2 & x_i y_i \\ x_i y_i & y_i^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Novamente a expansão dos termos, que agora são matriciais:

$$w_0 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + w_1 \begin{bmatrix} a^2 & 0 \\ 0 & 0 \end{bmatrix} + w_1 \begin{bmatrix} 0 & 0 \\ 0 & a^2 \end{bmatrix} + w_1 \begin{bmatrix} a^2 & 0 \\ 0 & 0 \end{bmatrix} + w_1 \begin{bmatrix} 0 & 0 \\ 0 & a^2 \end{bmatrix} + w_2 \begin{bmatrix} a^2 & a \\ a & a^2 \end{bmatrix} + w_2 \begin{bmatrix} a^2 & -a \\ -a & a^2 \end{bmatrix} + w_2 \begin{bmatrix} a^2 & a \\ a & a^2 \end{bmatrix} + w_2 \begin{bmatrix} a^2 & -a \\ -a & a^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Em evidência:

$$w_0 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + w_1 \begin{bmatrix} 2a^2 & 0 \\ 0 & 2a^2 \end{bmatrix} + w_2 \begin{bmatrix} 4a^2 & 0 \\ 0 & 4a^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Constata-se somente uma possibilidade de equação:

$$2a^2 w_1 + 4a^2 w_2 = 1 \tag{4.4}$$

Quando $m = 2$ e $n = 0$:

$$\sum_i w_i H^{(2)} H^{(0)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

A operação recaí no próprio polinômio na segunda ordem:

$$\sum_i w_i \begin{bmatrix} x_i^2 - 1 & x_i y_i \\ x_i y_i & y_i^2 - 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Abre-se o somatório:

$$w_0 \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} + w_1 \begin{bmatrix} a^2-1 & 0 \\ 0 & -1 \end{bmatrix} + w_1 \begin{bmatrix} -1 & 0 \\ 0 & a^2-1 \end{bmatrix} + w_1 \begin{bmatrix} a^2-1 & 0 \\ 0 & -1 \end{bmatrix} + w_1 \begin{bmatrix} -1 & 0 \\ 0 & a^2-1 \end{bmatrix} + w_2 \begin{bmatrix} a^2-1 & a^2 \\ a^2 & a^2-1 \end{bmatrix} + w_2 \begin{bmatrix} a^2-1 & -a^2 \\ -a^2 & a^2-1 \end{bmatrix} + w_2 \begin{bmatrix} a^2-1 & a^2 \\ a^2 & a^2-1 \end{bmatrix} + w_2 \begin{bmatrix} a^2-1 & -a^2 \\ -a^2 & a^2-1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Que se simplifica em:

$$w_0 \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} + w_1 \begin{bmatrix} 2a^2-4 & 0 \\ 0 & 2a^2-4 \end{bmatrix} + w_2 \begin{bmatrix} 4a^2-4 & 0 \\ 0 & 4a^2-4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Fornecendo a terceira equação:

$$\left(2a^2-4\right)w_1+\left(4a^2-4\right)w_2-w_0=0 \tag{4.5}$$

O produto tensorial $H^{(2)}H^{(1)}$ produz funções ímpares em relação a uma das variáveis para todos os índices. Isso ocasiona a indeterminação $0^{(2)}=0^{(2)}$. Efetuou-se a combinação $m=2$ e $n=2$ em apenas dois termos individuais do tensor: $\mathscr{H}^{(2)}_{01}\mathscr{H}^{(2)}_{01}$ e $\mathscr{H}^{(2)}_{00}\mathscr{H}^{(2)}_{00}$.

O primeiro caso:

$$\begin{aligned}\sum_i w_i\mathscr{H}^{(2)}_{01}\mathscr{H}^{(2)}_{01} &= \Lambda^{(2)}_{1010} &(4.6)\\ \sum_i w_i(x_iy_i)^2 &= \delta_{00}\delta_{11}+\delta_{10}\delta_{01} &(4.7)\\ 4w_2a^4 &= 1 &(4.8)\end{aligned}$$

resulta em:

$$w_2=a^4/4 \tag{4.9}$$

E o segundo caso:

$$\begin{aligned}\sum_i w_i\mathscr{H}^{(2)}_{00}\mathscr{H}^{(2)}_{00} &= \Lambda^{(2)}_{0000} &(4.10)\\ \sum_i w_i(x_i^2-1)^2 &= \delta_{11}\delta_{11}+\delta_{00}\delta_{00} &(4.11)\end{aligned}$$

conduz à equação:

$$w_0+2[(a^2-1)^2+2]w_1+4(a^2-1)^2w_2=2 \tag{4.12}$$

Em decorrência disso, o sistema conterá três equações linearmente independentes e uma equação linearmente dependente - terceira equação - provenientes de polinômios físicos de Hermite:

$$\begin{cases} w_0+4w_1+4w_2=1\\ 2a^2w_1+4a^2w_2=1\\ w_0+2[(a^2-1)^2+1]w_1+4(a^2-1)^2w_2=2\\ 4a^2w_4=1\end{cases} \tag{4.13}$$

O determinante não nulo:

$$\begin{vmatrix}1 & 4 & 4\\ 0 & 2a^2 & 4a^2\\ 0 & 0 & 4a^4\end{vmatrix}=8a^6 \tag{4.14}$$

garante isso.

A matriz inversa de coeficientes na sua forma inversa escalonada é:

$$M^{-1}=\begin{bmatrix}1 & -\frac{2}{a^2} & \frac{1}{a^4}\\ 0 & \frac{1}{2a^2} & -\frac{1}{2a^4}\\ 0 & 0 & \frac{1}{4a^4}\end{bmatrix} \tag{4.15}$$

Os coeficientes em função da escala são:

$$W_a = \begin{bmatrix} 1 - \frac{1}{2a^2} + \frac{1}{a^4} \\ \frac{1}{2a^2} - \frac{1}{2a^4} \\ \frac{1}{4a^4} \end{bmatrix} \tag{4.16}$$

A aplicação deles na última equação do sistema gera:

$$M_a * W_a = \left(\frac{1}{8a^2} - \frac{1}{32a^4}\right)\left(8a^2 - 8\right) - 2 + \frac{1}{a^2} + \frac{16a^2 - 8}{64a^4} - \frac{1}{8a^4} = 2 \tag{4.17}$$

em que M_a são os coeficientes em (4.12).

Simplificado, depreende-se que:

$$a^2 = 3 \rightarrow a = \sqrt{3} \tag{4.18}$$

Substituindo-se a em (4.16), recolhe-se os valores dos pesos, que estão alocados na Tabela 4.2.

Tabela 4.2 – Pesos para o *stencil* `D2Q9`

$\mathbf{c}_i^*$	p	w_i
$(0,0)$	1	$4/9$
$(\pm 1,0)$	4	$1/9$
$(\pm 1,\pm 1)$	4	$1/36$

Fonte: Autor.

4.2 Resolução Computacional dos *Stencils*

Os *lattices* `D2Q49` e `D2Q81`, que dão suporte às experimentações com os iteradores, foram ensaiados no programa `lbeST`, a fim de garantir a compatibilidade com os resultados disponíveis na literatura de Mattila, Hegele Jr. e Philippi (2014), retornando os conjuntos *stencils* presentes nas Tabelas 4.3 e 4.4.

A discrepância, em módulo, com os valores w_{iREF} em Mattila, Hegele Jr. e Philippi (2014), não ultrapassou 10^{-15}. Os erros decresceram regularmente junto com os pesos, conforme houve acréscimo da norma dos vetores de velocidade discreta.

Os testes de verificação foram feitos por um módulo do programa `lbeST` nos casos `D2V49` e `D2V81` e efetuados por uma função do programa `SimpleStencil` para `D2V141` e `D2V169`, atestando a prestabilidade dos valores em relação as demais combinações de produtos tensoriais até a máxima ordem ao qual cada *stencil* foi designado. Como já era esperado, houve um aumento do erro com a ordem. Os reportados na Tabela 4.7 condizem ao máximo valor absoluto da diferença entre aqueles resultantes de operações entre tensores sob a mesma ordem.

Tabela 4.3 – Pesos w_i de `D2V49`. A escala $a = 1.148732248838539$, difere da literatura $a_{REF} = 1.148732248838539$ em um erro de $|a - a_{REF}| = 0.0$.

$\mathbf{c}_i^\star$	w_i	w_{iREF}	$\lvert w_i - w_{iREF}\rvert$
(0, 0)	2.1486215695621577e-01	2.1486215695621588e-01	1.1102e-16
(1, 0)	1.0466324937773652e-01	1.0466324937773665e-01	1.2490e-16
(1, 1)	5.9279902836032169e-02	5.9279902836032114e-02	5.5511e-17
(2, 0)	1.6329411241508079e-02	1.6329411241508003e-02	7.6328e-17
(2, 1)	6.6560728515645019e-03	6.6560728515645175e-03	1.5613e-17
(2, 2)	1.4236189699711446e-03	1.4236189699711051e-03	3.9465e-17
(3, 0)	3.8641376938725795e-04	3.8641376938724456e-04	1.3390e-17
(3, 1)	4.4284732290059489e-04	4.4284732290059494e-04	5.4210e-20
(3, 3)	2.9169039140653727e-06	2.9169039140666428e-06	1.2701e-18
(5, 0)	8.8685055032561960e-07	8.8685055032561949e-07	1.0588e-22
(4, 4)	2.2046291628744430e-07	2.2046291628739507e-07	4.9234e-20

Tabela 4.4 – Pesos w_i de `D2V81`. A escala $a = 0.97000849873938888$, difere da literatura $a_{REF} = 0.970008498739395$ em um erro de $|a - a_{REF}| = 6.1062e-15$.

$\mathbf{c}_i^\star$	w_i	w_{iREF}	$\lvert w_i - w_{iREF}\rvert$
(0, 0)	1.5381368561993133e-01	1.5381368561993594e-01	4.6074e-15
(1, 0)	9.0384577224029344e-02	9.0384577224028734e-02	6.1062e-16
(1, 1)	6.0915016936039602e-02	6.0915016936039915e-02	3.1225e-16
(2, 0)	2.4220944974511015e-02	2.4220944974511140e-02	1.2490e-16
(2, 1)	1.3146486542344191e-02	1.3146486542343966e-02	2.2551e-16
(2, 2)	3.9625069477692305e-03	3.9625069477690709e-03	1.5959e-16
(3, 0)	1.8542634450575606e-03	1.8542634450574055e-03	1.5504e-16
(3, 1)	1.6035095499281192e-03	1.6035095499280431e-03	7.6111e-17
(3, 2)	2.2249142091157587e-04	2.2249142091156890e-04	6.9660e-18
(4, 0)	1.0005397521769422e-04	1.0005397521769796e-04	3.7405e-18
(4, 1)	3.4398327658315626e-05	3.4398327658312326e-05	3.3000e-18
(3, 3)	5.4494804327701634e-05	5.4494804327695976e-05	5.6582e-18
(4, 2)	1.8837067344238397e-05	1.8837067344235765e-05	2.6326e-18
(5, 0)	2.4044498210979649e-06	2.4044498210968849e-06	1.0800e-18
(5, 2)	4.3501093550345762e-07	4.3501093550344725e-07	1.0376e-20

Tabela 4.5 – Pesos w_i de `D2V169` e escala $a = 8.2078063160959303e-01$. Menções sobre esse *lattice* na literatura não são de conhecimento do autor.

$\mathbf{c}_i^\star$	w_i	w_{iREF}	$\lvert w_i - w_{iREF}\rvert$
(0, 0)	1.1040226513084352e-01	-	-
(1, 0)	7.3935350380450918e-02	-	-
(1, 1)	5.6821936412125483e-02	-	-
(2, 0)	2.9315452959156427e-02	-	-
(2, 1)	1.8711612190505957e-02	-	-
(2, 2)	7.8915706122873602e-03	-	-
(3, 0)	4.6592015086096448e-03	-	-
(3, 1)	4.1122880356028076e-03	-	-
(3, 2)	1.1197307545239795e-03	-	-
(4, 0)	5.9638330530080560e-04	-	-
(4, 1)	2.6285322600543324e-04	-	-
(3, 3)	3.2478017778134438e-04	-	-
(4, 2)	1.7210791758125554e-04	-	-
(4, 3)	9.5738950871639860e-06	-	-
(5, 0)	1.2866145699512254e-05	-	-
(5, 1)	2.5399196143602410e-05	-	-
(5, 2)	2.0157339711218683e-06	-	-
(4, 4)	4.4807247976089051e-06	-	-
(5, 3)	2.2328502171400653e-06	-	-
(6, 0)	8.4529042204351258e-07	-	-
(6, 1)	2.1190772989162247e-07	-	-
(6, 2)	2.2427099652280496e-07	-	-
(6, 3)	2.0934296169047109e-08	-	-
(7, 0)	1.5261525037644502e-08	-	-
(7, 2)	4.4315975674570706e-09	-	-
(6, 6)	2.4618623520954060e-10	-	-
(7, 7)	3.7622351406778712e-12	-	-
(10, 0)	4.1963051739530126e-13	-	-
(10, 2)	1.2380501798222870e-13	-	-

Tabela 4.6 – Pesos w_i de `D2V141`. A escala $a = 0.83692040543017843$, difere da literatura $a_{REF} = 0.8369204054303525$ em um erro de $|a - a_{REF}| = 1.7408297026122455e-13$.

$\mathbf{c}_i^\star$	w_i	w_{iREF}	$\lvert w_i - w_{iREF}\rvert$
(0, 0)	1.1508949125645836e-01	1.1508949125706189e-01	6.0353e-13
(1, 0)	7.5595334899980299e-02	7.5595334899625166e-02	3.5513e-13
(1, 1)	5.7734363121691910e-02	5.7734363121950370e-02	2.5846e-13
(2, 0)	2.9033095241433594e-02	2.9033095241565582e-02	1.3199e-13
(2, 1)	1.8074672747709450e-02	1.8074672747613353e-02	9.6097e-14
(2, 2)	7.4435479269468969e-03	7.4435479269746455e-03	2.7749e-14
(3, 0)	4.2549928887467073e-03	4.2549928887228462e-03	2.3861e-14
(3, 1)	3.7759145451449695e-03	3.7759145451527918e-03	7.8223e-15
(3, 2)	9.5510661516599296e-04	9.5510661516062627e-04	5.3667e-15
(4, 0)	4.9926973701088070e-04	4.9926973701248283e-04	1.6021e-15
(4, 1)	2.1766901272397883e-04	2.1766901272228135e-04	1.6975e-15
(3, 3)	2.7427360033272245e-04	2.7427360033189315e-04	8.2931e-16
(4, 2)	1.3852638332685974e-04	1.3852638332696371e-04	1.0397e-16
(5, 0)	1.2905073342702279e-05	1.2905073342509215e-05	1.9306e-16
(4, 3)	5.9323648903242820e-06	5.9323648903820310e-06	5.7749e-17
(5, 1)	1.6161185887346246e-05	1.6161185887309810e-05	3.6436e-17
(5, 2)	2.4159493338204858e-06	2.4159493337948245e-06	2.5661e-17
(4, 4)	3.3101853528026843e-06	3.3101853527875648e-06	1.5120e-17
(5, 3)	1.3086410702127725e-06	1.3086410701892049e-06	2.3568e-17
(6, 0)	1.1118416570979004e-07	1.1118416570950374e-07	2.8630e-19
(6, 1)	4.6121305137312854e-07	4.6121305137932601e-07	6.1975e-18
(6, 3)	3.6530518725909026e-08	3.6530518727364592e-08	1.4556e-18
(7, 0)	8.3273853758250555e-09	8.3273853753395782e-09	4.8548e-19
(7, 2)	2.3109247817687987e-09	2.3109247814347261e-09	3.3407e-19

Tabela 4.7 – Verificação de Ordem, de acordo com a Seção 3.5.

	$\lvert ERROR\rvert$			
ORDER	`D2V49`	`D2V81`	`D2V141`	`D2V169`
0	0.0000e+00	0.0000e+00	4.5770e-16	1.0869e-16
1	1.3396e-16	2.2204e-16	4.6129e-16	1.5116e-15
2	1.4821e-15	7.9450e-16	3.6815e-15	1.0599e-14
3	1.0401e-14	1.7736e-14	2.9517e-14	1.1815e-13
4	5.9619e-14	1.9962e-13	3.9179e-13	2.6672e-12
5	5.9697e-13	1.9744e-12	4.8401e-12	5.8754e-11
6		3.3324e-11	6.1737e-11	1.0617e-09
7			8.4356e-10	1.7607e-08
8			2.0690e-08	2.6274e-07
9				3.2092e-06

4.3 Simulação dos Iteradores

Aspectos inerentes a cada iterador simulado serão discutidos, sendo subdivididos em dois grupos principais: o que engloba o tempo de simulação e o que diz respeito a sequência de descoberta de equações L.I. e sua densidade relativa ao número total de equações. O último grupo não é afetado por alterações no *catching*, portanto as experimentações são concentradas apenas em duas das séries em 3.5.2.

4.3.1 Estudo da Variável `MNP[w]`

A variável `MNP[w]` sinaliza no Plano de Produto Tensorial o padrão de reconhecimento de equações para cada modo iterador testado, à esquerda em `nc/c-nct` e à direita em `nc/c-ct`, conforme exibido nas Figuras 4.3 até 4.16.

O `bottomUp` é o iterador de verificação padrão no `lbeST`, razão pela qual percorre coordenadas de soma ímpar $p = 1$ — a constante p é usada na programação de algoritmos de iteração, assumindo valor binário para atestar a paridade da soma $m+n$. O novo modo `evenBU` recebeu um ajuste para restringir as varreduras em $p = 0$, eliminando pontos intrinsecamente improdutivos, assim tornando-o um modo de captura. Quando o verificador de linha não nula é acionado LI_{Max} é atingido inteiramente em $n = 0$ com o acréscimo de LI_{Eq} por nó que acompanha a ordem.

Programou-se o `evenLimBU` com a finalidade de descartar a varredura sobre entradas nulas que ocorrem sempre que $n > 1$, se $N < N_{Max}$. Ao todo o número de expressões válidas por ordem é inferior a 3 para `bottomUp`.

`alterDiag` é uma alteração com base no desempenho de `diagDown`, para que o ponto $n = N_{Max} - 1$, $N = N_{Max}$ fosse inspecionado primeiro, evitando uma passagem *upward* na próxima diagonal. Em `nc/c-ct`, tanto `alterDiag`, quanto `diagDown` completam LI_{Max} na linha de simetria. A coleta de `succDiag` equivale a de `diagDown` e também usufrui dessa compensação, relembrando que dois iteradores são ditos equivalentes quando mapeiam expressões nas mesmas coordenadas do Plano de Produto Tensorial.

`limTri` e `limTri2` reagem a `succRank`, de modo a condicionar a busca de forma crescente sobre N nas duas primeiras diagonais não nulas e diferem de acordo com o rastreamento após o término em $N = N_{Max} - 1$, $n = N_{Max} - 2$.

E `inLine` por se assemelhar à `diagDown` e `succDiag` têm mesmo LI_{Eq} na diagonal principal, fora o último ponto, no qual é deslocado até a linha de ordem $N = N_{Max}$, bastando avançar apenas mais uma coordenada para completar o sistema com o verificador de linha não nula.

A classe horizontal, que compreende as modificações de `horizonUp`, atua do mesma maneira sobre a linha $n = 0$, no entanto `horizonUp` identificou apenas uma equação ao término da linha $n = 1$, requisitando avanço sobre $n = 3$. Se o incremento é de $n_{i+1} = n_{i+2}$, como em `horizonUp2`, as três expressões restantes ficam concentradas em $N = N_{Max}$, $n = 2$,

local também explorado por `horizonUp3`, mais rapidamente. Uma quarta versão poderia ser programada para escapar ($N = N_{Max}$, $n = 1$), um ponto da máxima ordem que é algebricamente custoso. Todas atingiram LI_{Max} em $n = 0$ para `nc/c-ct`.

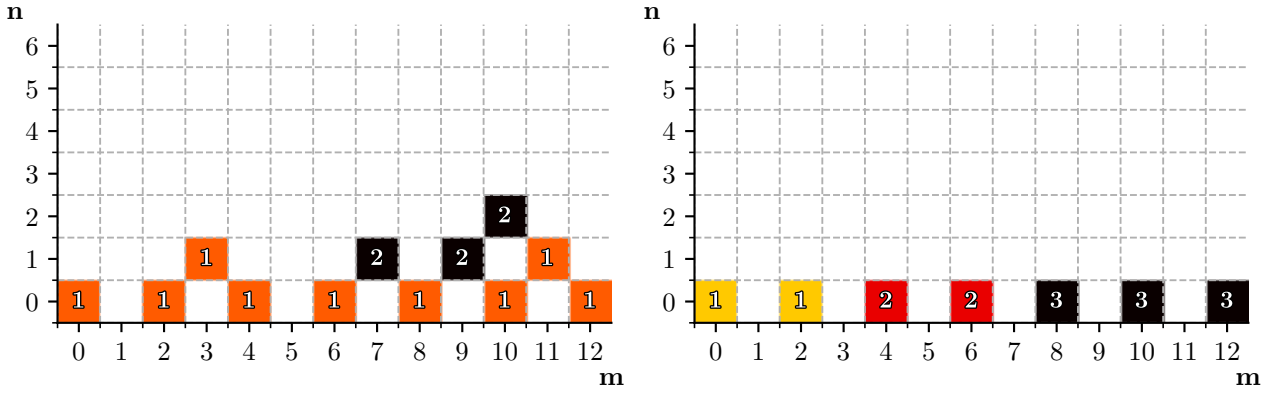
O `theSky` exibiu uma disposição sobre o PPT mais particular em relação aos demais iteradores, devido a presença de um passo de iteração não derivado. Na forma matricial à esquerda, o modo concentra LI_{Eq} em uma menor quantidade de coordenadas, auferindo até três equações nos pontos com maior N, então alcançando LI_{Max} em apenas um na linha de ordem.

Os modos `highOrdU` e `highOrdD` foram criados para a seleção de a_{Eq}, baseado em um conjunto L.I. condicionado a um traçado de caminho. Em sua função usual, ele depende desse traçado. Ao serem atribuídos para captura, apresentam desempenho pífio em comparação aos outros em termos de *Benchmark*, sendo portanto incompatíveis para tarefa, apesar de conservarem, em geral, uma boa d_{Eq}. Ambos não foram simulados no programa-padrão, que se mostrou incapaz de retornar o resultado em tempo hábil para essa ordem, mas pelo `SimpleStencil`, no qual se manteve o mesmo código para checagem L.I.. Entretanto, sua experimentação é interessante para avaliar, posteriormente, o plano no reconhecimento de candidatos à a_{Eq} — Seção 4.3.2.

`highOrdD` é o iterador que possui maior concentração de LI_{Eq} em um único nó, revelando $n = N = N_{Max}$ como um candidato a inspeção de chave, também é aquele que chega a LI_{Max} com o menor número de coordenadas averiguadas. Na ativação de `nc/c-ct` essa quantia é reduzida ainda mais, de quatro para três nós.

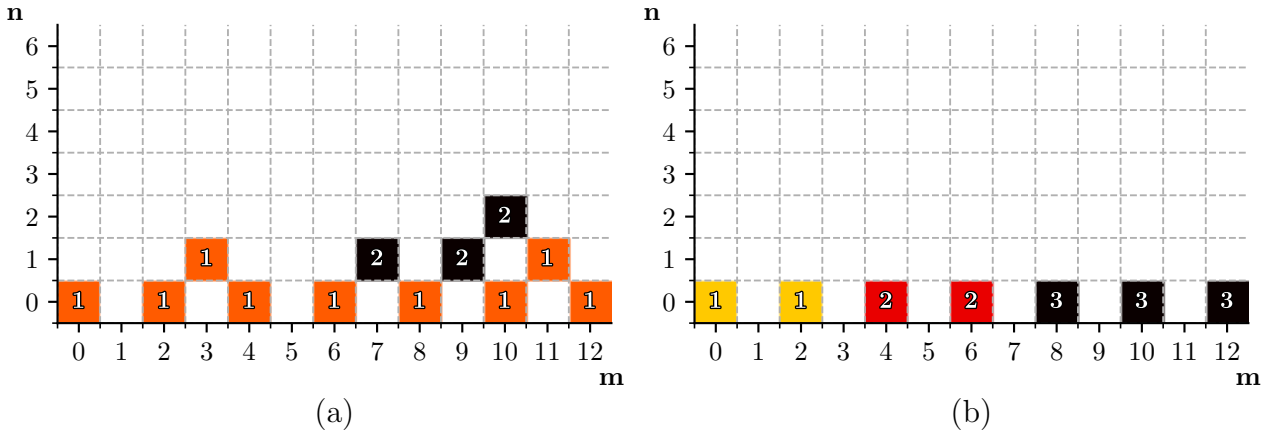
`highOrdU` praticamente é efetivo nos pontos complementares a `highOrdD` na máxima ordem, mas não é tão compacto quanto o primeiro. Foi o único modo homólogo no PPT em relação ao verificador de independência linear, assim figurando outro candidato a inspeção de chave.

Figura 4.3 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `evenLimBU`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda, (a) para os verificador não corrigido `nc/c-nct` e, à direita, (b) para o verificador corrigido `nc/c-ct`.



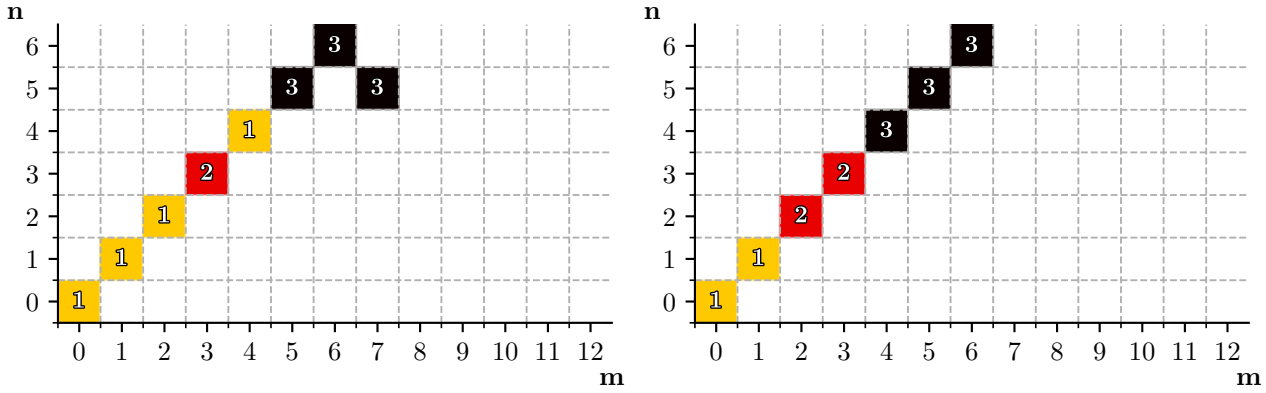
Fonte: Autor

Figura 4.4 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `bottomUp`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



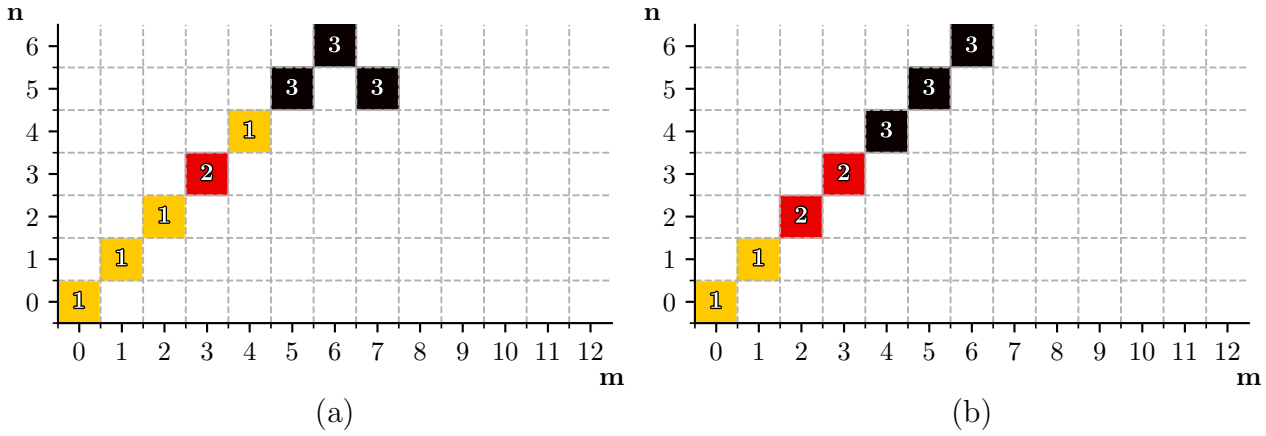
Fonte: Autor

Figura 4.5 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `alterDiag`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



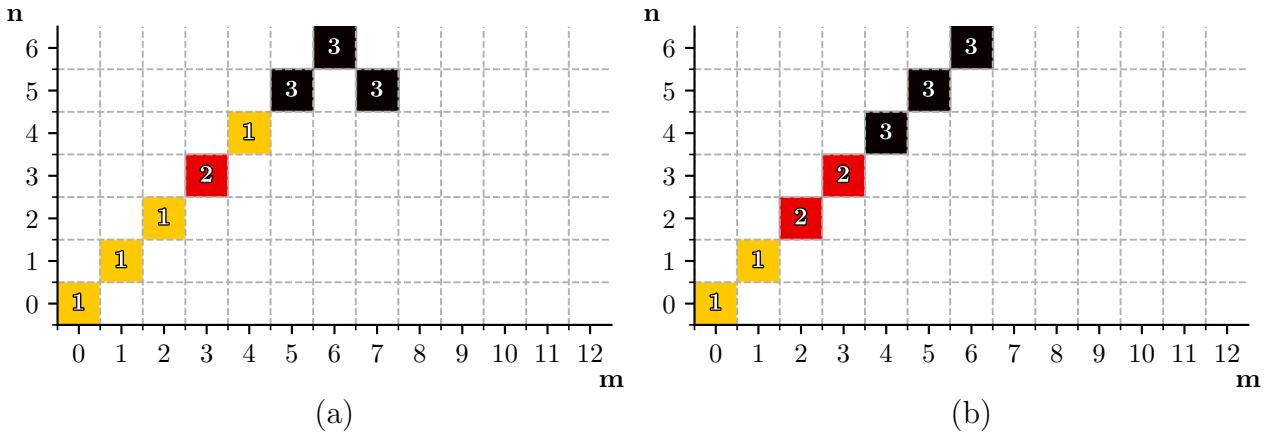
Fonte: Autor

Figura 4.6 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `diagDown`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



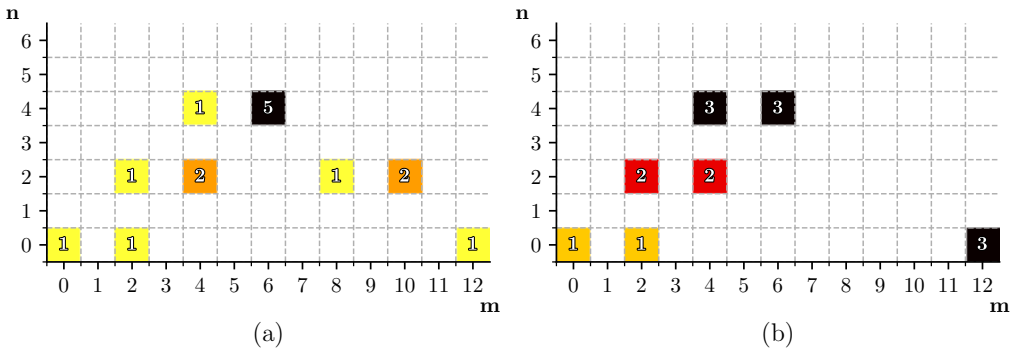
Fonte: Autor

Figura 4.7 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `succDiag`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



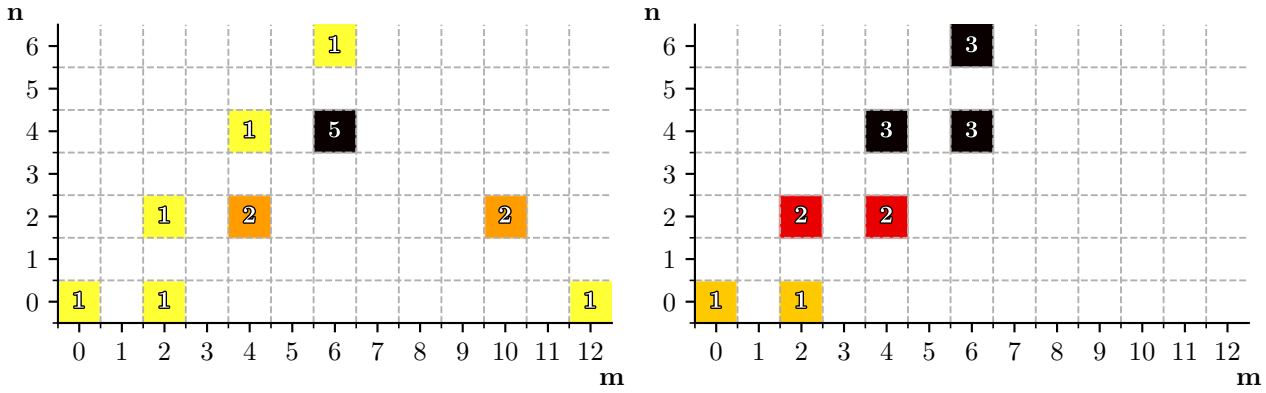
Fonte: Autor

Figura 4.8 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `limTri`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



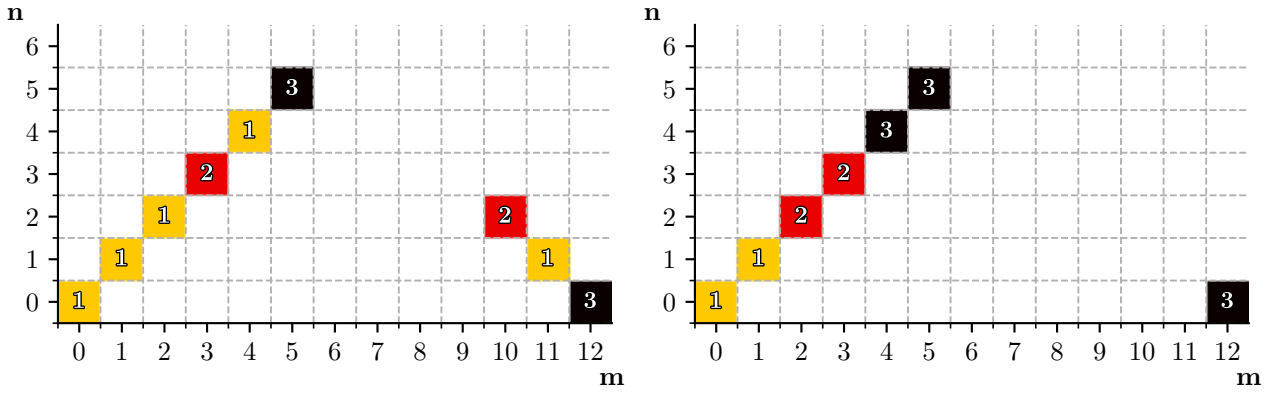
Fonte: Autor

Figura 4.9 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `limTri2`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



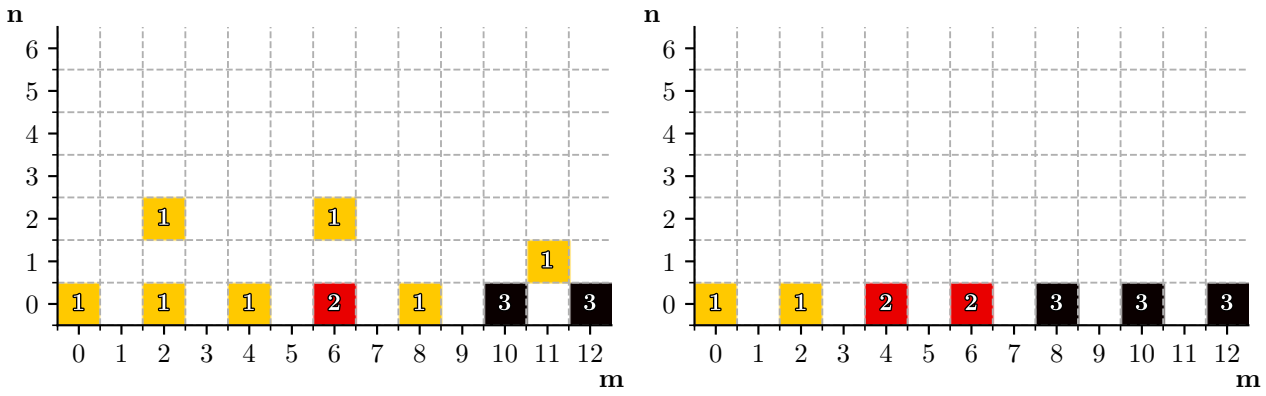
Fonte: Autor

Figura 4.10 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `inLine`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



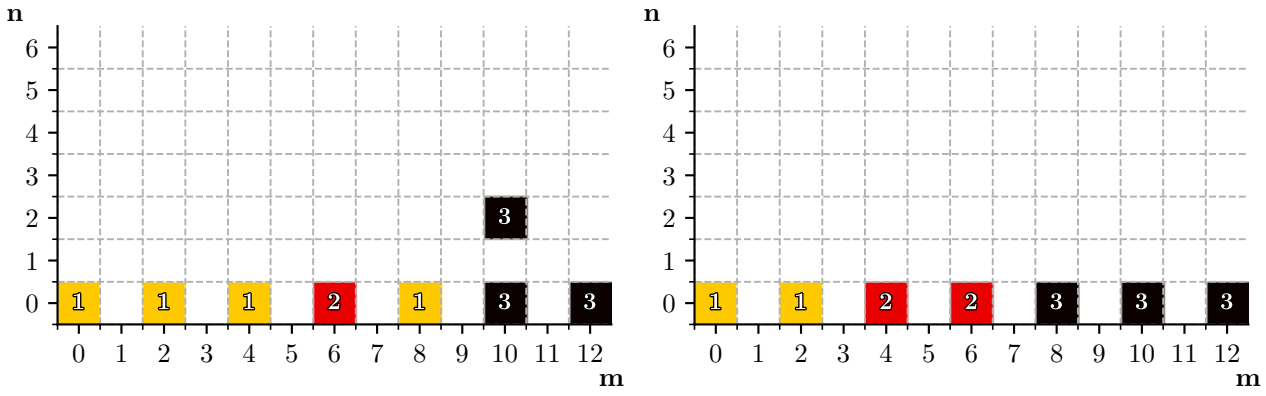
Fonte: Autor

Figura 4.11 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `horizonUp`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



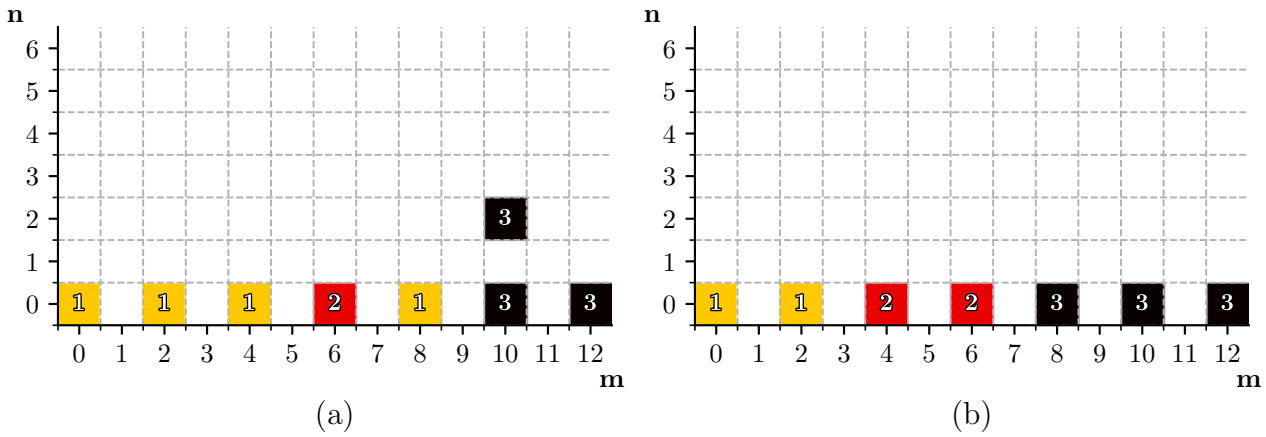
Fonte: Autor

Figura 4.12 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `horizonUp2`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



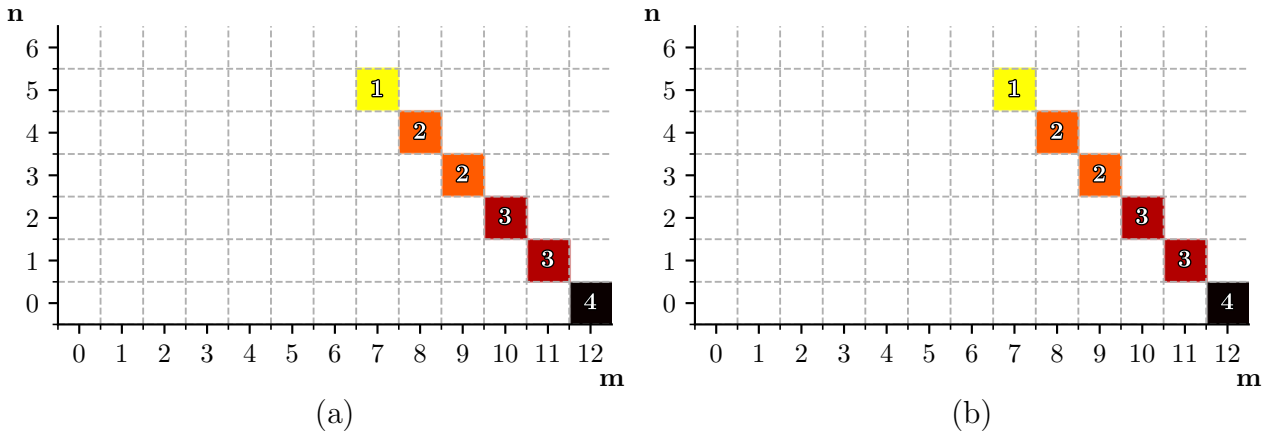
Fonte: Autor

Figura 4.13 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `horizonUp3`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



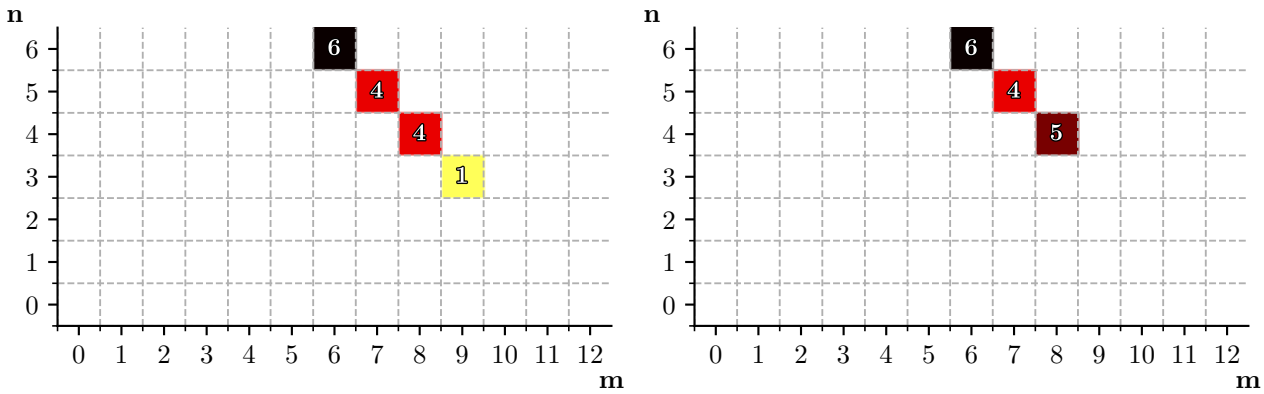
Fonte: Autor

Figura 4.14 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `highOrdU`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



Fonte: Autor

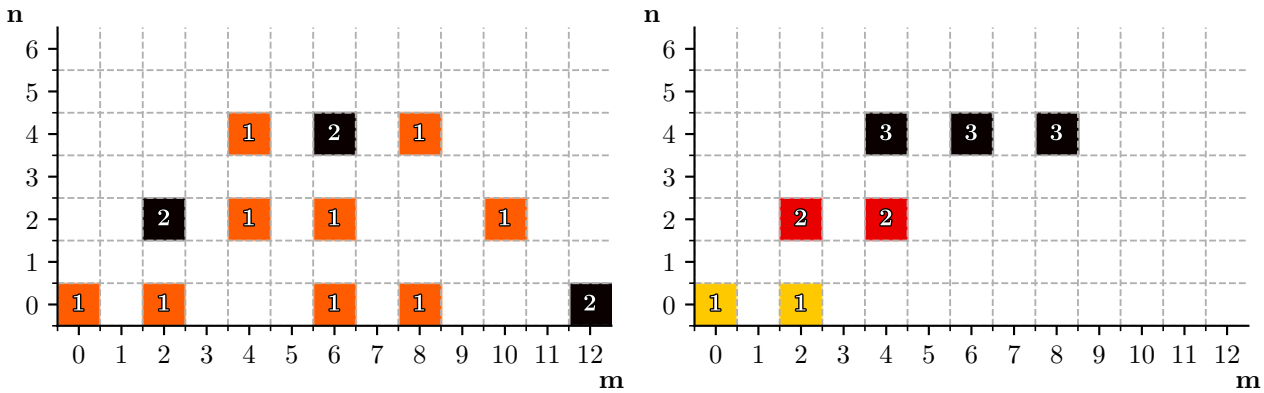
Figura 4.15 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `highOrdD`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



(a) (b)

Fonte: Autor

Figura 4.16 – Representação da variável `MNP[w]` — Seção 3.5.4 — para o modo iterador `theSky`. Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de equações L.I. assimiladas quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`. À esquerda (a) para os verificador não corrigido `nc/c-nct` e à direita (b) para o verificador corrigido `nc/c-ct`.



(a) (b)

Fonte: Autor

4.3.2 Estudo da Variável `MNP[a]`

A seleção da equação para a cabe ao método `lbeST.meth.PA.compEq.selectAEq`, que é encarregado da exclusão das equações já incorporadas ao sistema e daquelas consideradas redundantes ($0 = 0$), quanto a substituição da solução para os pesos. A condicionante `sympy.Eq(aExpr, vl) == True` na função, não detectava apropriadamente as indeterminações, se a substituição ocorresse com expressões não simplificadas para os w_i's, por conta de um *bug* dentro do pacote simbólico.

Uma nova tática foi incorporada, a fim de checar numericamente a discrepância em valores absolutos entre os lados da igualdade para a, partindo da substituição com o primeiro primo do 38º *quadruplet*: 99131. Ressalta-se o risco desse número coincidir com a raiz de algum polinômio em a, mesmo que o `lbeST` tenha operado sem inconformidades após a adequação.

A posição das candidatas a_{Eq} no PPT, inspecionadas por `selectAEq`, não dependem do iterador configurado dentro do método, mas do modo despendido na busca das equações L.I., como foi anteriormente mencionado. Assim, o confeccionador de caminho será fixado como `highOrdU`, por começar a pesquisa pelo ponto de processamento simbólico menos custoso da última ordem, no qual há menor chance de rastreamento de elementos redundantes. Depois do esgotamento de todos os arranjos possíveis com as variações dos iteradores, um novo algoritmo de caminho será proposto, se necessário, contemplando a captura de a_{Eq} no mínimo tempo permitido em todas as circunstâncias. Os diagramas derivados das formas matriciais do PPT armazenadas em `MNP[a]`, destacando as candidatas a a_{Eq} para cada cenário de captura L.I., em `nc/c-nct`, estão registradas nas Figuras 4.17 a 4.24.

A observação das matrizes revela diferentes níveis de dispersão sobre o plano. `alterDiag`, `diagDown`, `succDiag` e `inLine` têm candidatas em quase todos os nós a partir da segunda ordem, entretanto o último exterioriza uma concentração nos três pontos superiores de N_{Max}, exibindo o maior acúmulo por coordenada entre as ocorrências em $n = N_{Max}$. O `bottomUp` e o `evenLimBU` revelam expressões a partir da terceira ordem com maior $n(a)$ também em $n = N_{Max}$.

O `horizonUp` dispõe as equações em um retângulo com vértices em $(n,N) : [(N_{Max}, N_{Max}), (N_{Max}-2, N_{Max}-2), (0, N_{Max}-2), (2, N_{Max})]$. Como implicação dos últimos componentes L.I. serem incorporados na última ordem, `horizonUp2` e `horizonUp3` apontam um retângulo similar espelhado na coluna média do plano, embora $n(a)$ por nó continue incrementando no sentido da máxima ordem.

`limTri2` revela entradas não nulas da 4ª ordem em diante e `theSky` a partir da 5ª. A distribuição `limTri` difere da seu primeiro fatorial ao direcionar os candidatos à máxima ordem, portando cada nó exatamente um proveitoso. Além desse modo, `succRank` também compartilha o mesmo comportamento, que se destaca por minimizar $n(a)$. Então, para a amostra estudada, $n(a)_{Min} = N_{Max}$.

Tabela 4.8 – Chaves referentes aos elementos de `MNP[a]` — Seção 3.5.4 —, componentes dos produtos tensoriais que produzem equações candidatas à a_{Eq}, para os iteradores `limTri` e `succRank` e para todos os modos em `nc/c-ct`.

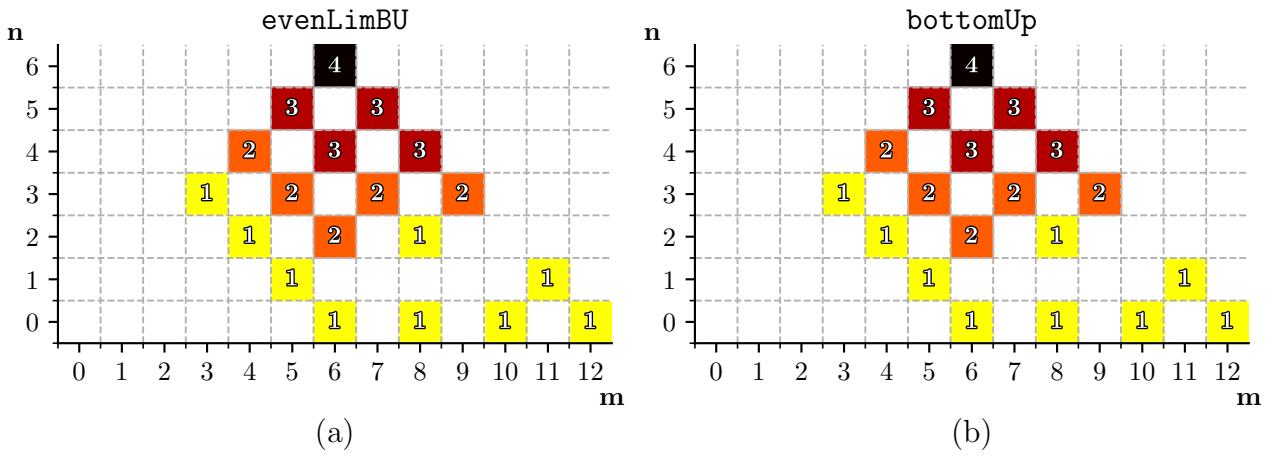
n	$\mathscr{H}_i^{(n)}\,\mathscr{H}_j^{(m)}$	Chaves
0	$\mathscr{H}^{(0)}\,\mathscr{H}^{(12)}_{000000000000}$	.000000000000
1	$\mathscr{H}^{(1)}_{0}\,\mathscr{H}^{(11)}_{00000000000}$	0.00000000000
2	$\mathscr{H}^{(2)}_{00}\,\mathscr{H}^{(10)}_{0000000000}$	00.0000000000
3	$\mathscr{H}^{(3)}_{000}\,\mathscr{H}^{(9)}_{000000000}$	000.000000000
4	$\mathscr{H}^{(4)}_{0000}\,\mathscr{H}^{(8)}_{00000000}$	0000.00000000
5	$\mathscr{H}^{(5)}_{00000}\,\mathscr{H}^{(7)}_{0000000}$	00000.0000000
6	$\mathscr{H}^{(6)}_{000000}\,\mathscr{H}^{(6)}_{000000}$	000000.000000

As chaves correspondentes aos elementos de `MNP[a]` dois dos últimos modos coincidem em serem inteiramente nulas, isto é, com derivadas parciais em um termos de uma única variável, em N_{Max}, vide a Tabela 4.8. Logo, a programação de um caminho ideal para `a` deve obrigatoriamente contemplar o conjunto mínimo, uma vez que os percursos ligados aos demais iteradores contêm pelo menos um membro desse grupo, com exceção de `highOrdU`. Uma solução simples e eficaz foi inserida no programa `SimpleStencil` , que é a reserva prévia da uma das chaves: '`.000000000000`' ou de '`000000.000000`', genericamente '`.0*`$2N_{Max}$' ou '`0*(`N_{Max}`).0*(`N_{Max}`)`', excluindo-a da inspeção feita para a montagem do SPD. Maiores detalhes na Seção 4.4. A equação especificada por '`.0*`N_{Max}' é preferível as demais de mesma categoria anteriormente tratadas, em virtude de economizar derivadas parciais na atual condição do `lbeST`, sem o cachê, e do `SimpleStencil`. E em comparação com '`0*(`$N_{Max}/2$`).0*(`$N_{Max}/2$`)`' descarta a demanda de permutações na determinação do termo vindo do tensor de ortogonalidade.

`highOrdD` e `highOrdU` constituiram casos especiais na análise de `MNP[a]` por resgatarem todo o LI_{Max} na máxima ordem, no entanto houve um contraste entre ambos no quesito de dispersão. O primeiro exibiu possibilidades em todo o plano, exceto nas coordenadas em que as equações L.I. foram apanhadas, enquanto o segundo as deteve sobre a diagonal principal, com $n(a)$ crescente com a ordem.

As formas matriciais em `MNP[a]`, com `nc/c-ct` são iguais ao resultado apresentado para `succRank` e `limTri` de `nc/c-nct` para todos os iteradores, fora aqueles com coleta inteiramente na máxima ordem, permitindo afirmar que se a verificação L.I. é efetuada com base em linha não nula, então as candidatas a a_{Eq} estarão dispostas em N_{Max}, como chaves escritas apenas em zero, estas entendidas como marcações de polinômios tensoriais de Hermite formados por derivação parcial em relação a uma única variável.

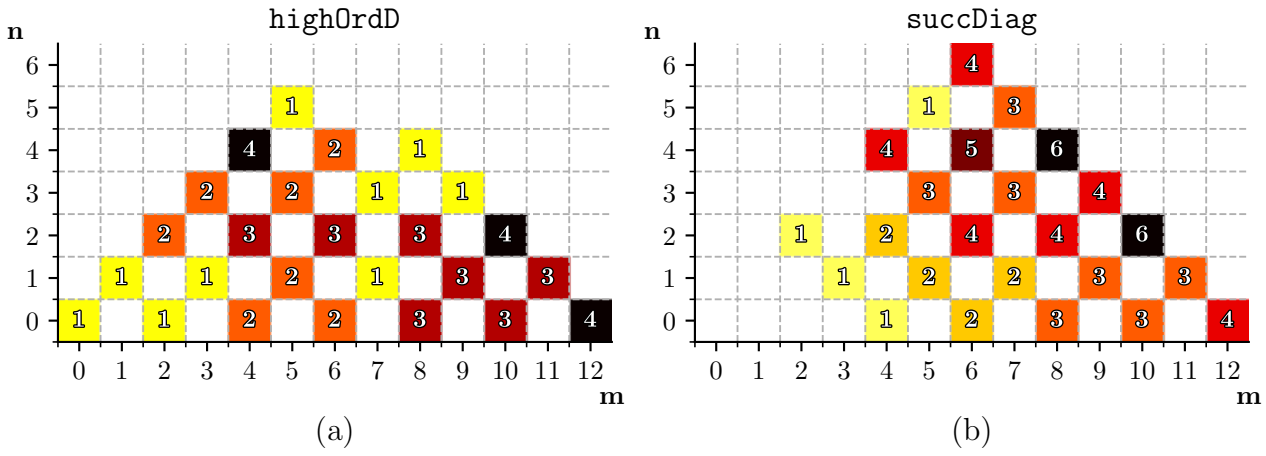
Figura 4.17 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `evenLimBU` (a) e `bottomUp` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

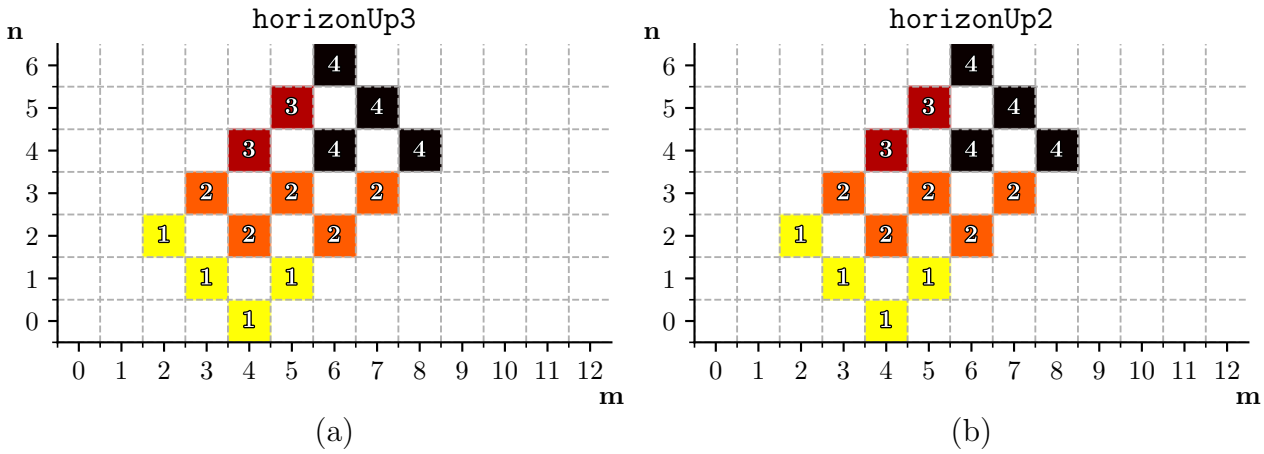
Figura 4.18 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `highOrdD` (a) e `succDiag` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

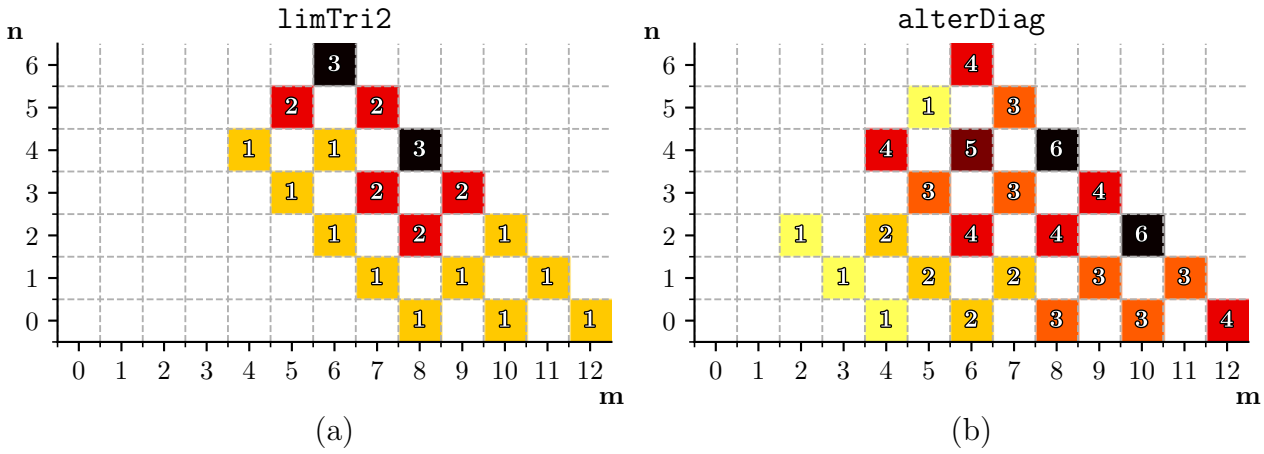
Figura 4.19 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `horizonUp3` (a) e `horizonUp2` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

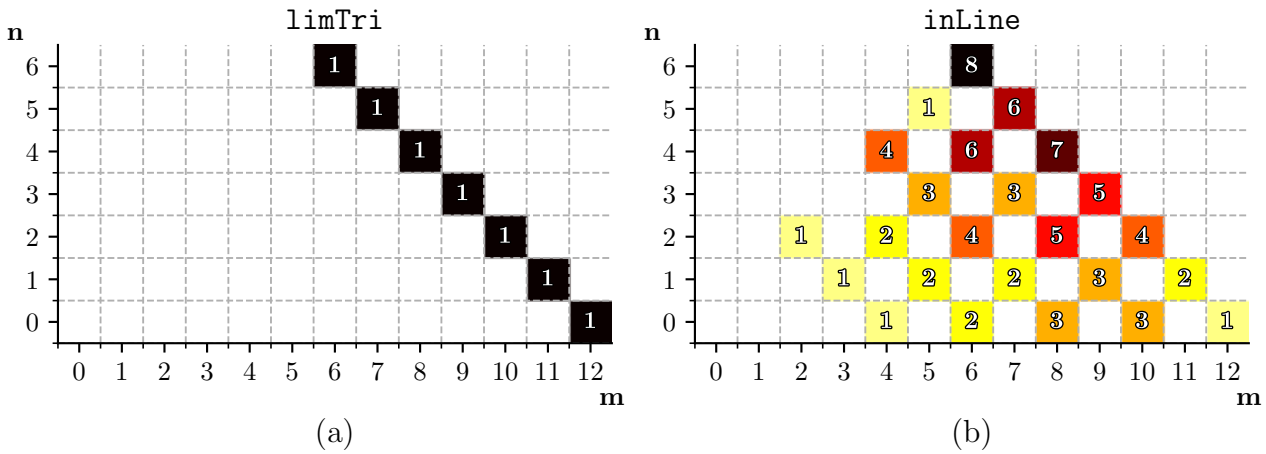
Figura 4.20 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `limTri2` (a) e `alterDiag` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

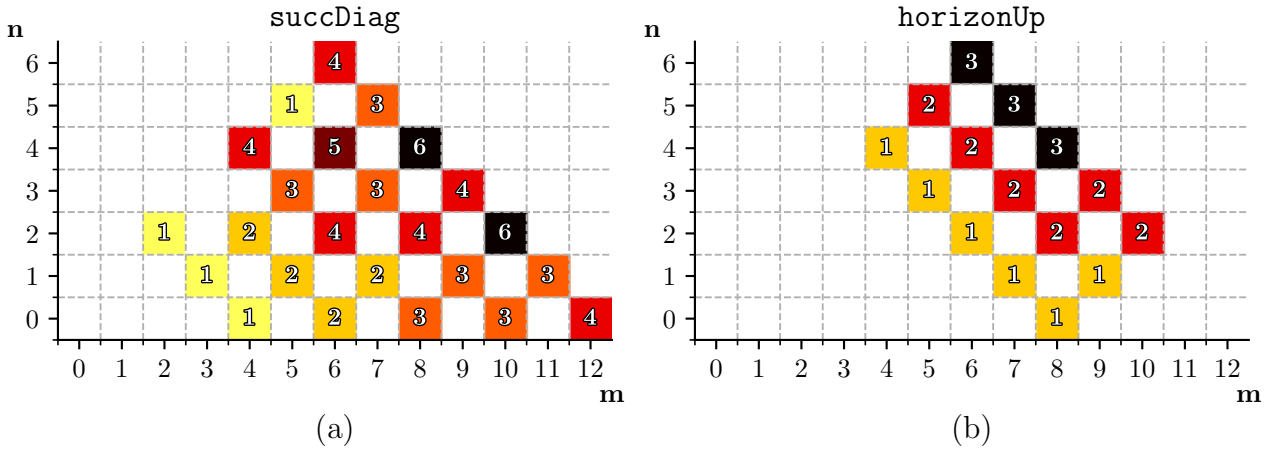
Figura 4.21 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `limTri` (a) e `inLine` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

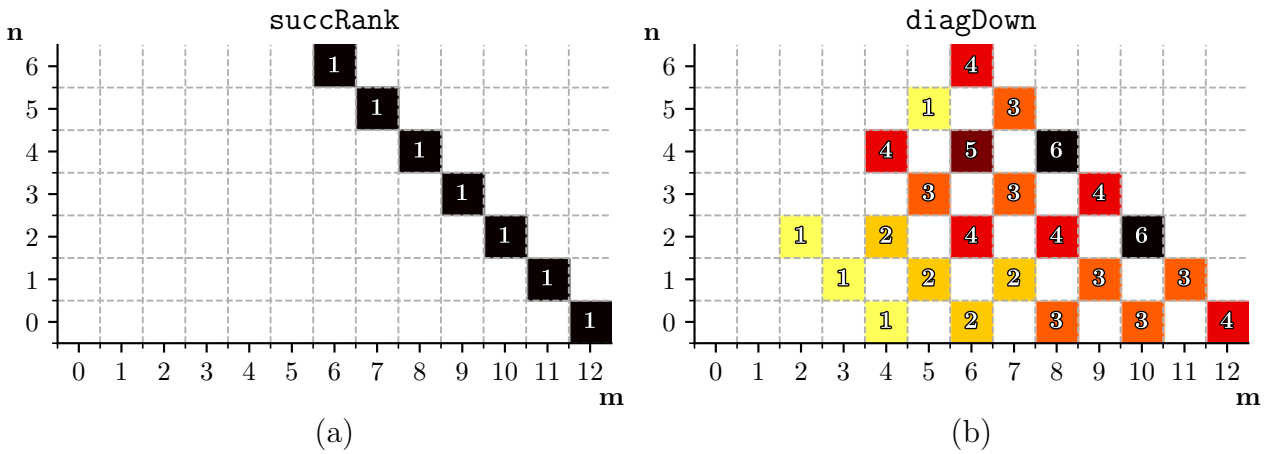
Figura 4.22 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `succDiag` (a) e `horizonUp` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

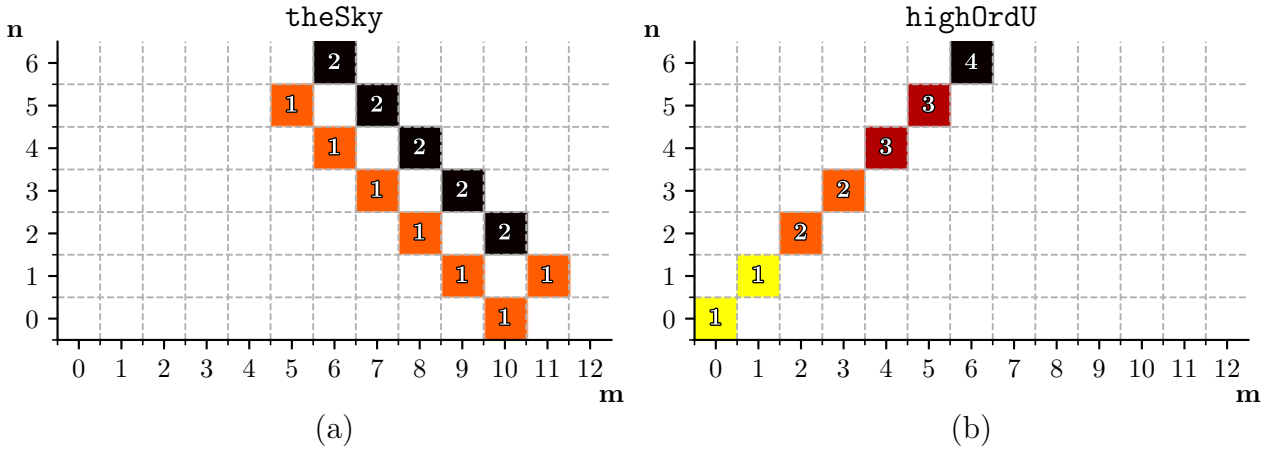
Figura 4.23 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `succRank` (a) e `diagDown` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

Figura 4.24 – Representação da variável `MNP[a]` — Seção 3.5.4 — para o modos iteradores `theSky` (a) e `highOrdU` (b). Cada quadrado é orientado no plano por uma tupla (n,m), indicando a quantidade de a_{Eq} possíveis quando essa combinação é inserida na Equação 3.48 para o *lattice* `D2V81`.



(a) (b)

Fonte: Autor

4.3.3 Estudo do Gráfico LI_{Eq} x T_{Eq} e da Variável d_{Eq}

Os Gráficos 4.27 e 4.27 para `nc/c-nct` e `nc/c-ct`, respectivamente, registram a razão entre o total T_{Eq} de equações inspecionadas e o número LI_{Eq} de L.I. armazenadas, exteriorizando os dados contidos em `DATA['getCOL']`. Essas plotagens permitem avaliar a progressão da densidade de equações nos momentos de captura.

Conforme o Gráfico 4.25, é possível averiguar o acréscimo performance dos novos iteradores, excetuando-se `theSky`, que alcançam LI_{Max}, percorrendo menos de 70 nós do plano e com uma d_{Eq} superior a quase todos os modos pré-programados durante o processo inteiro. Em geral, o conjunto de resposta mostrou densidade de L.I. muito similar até o resgate da 11ª equação, na qual se acentuaram as divergências entre as amostras.

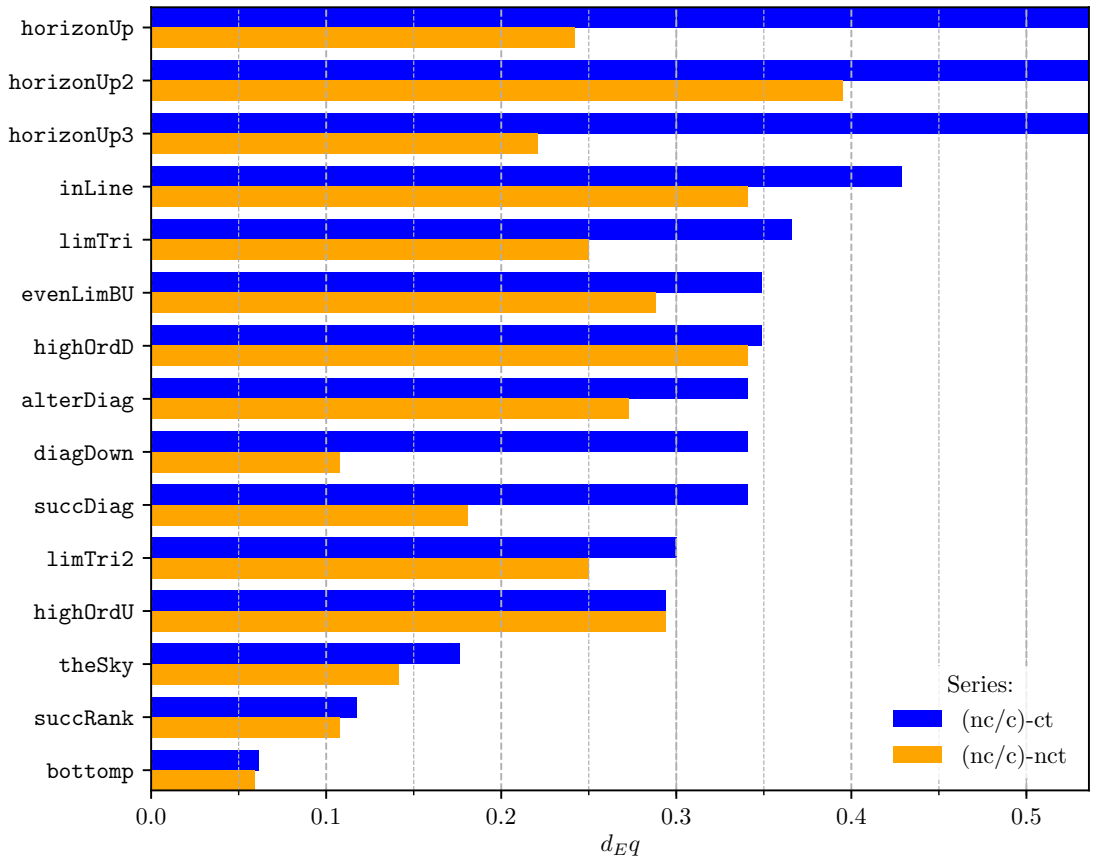
Os modos horizontais foram excelentes nesse teste, porque identificam na linha $m = 0$ até a 12ª equação, com destaque para `horizonUp3`, que conservou a inclinação até o término da operação, nos instantes em que os fatoriais diferiram entre si. Uma grande gama de iteradores progride em densidade até $LI_{Eq} = 11$ com $T_{Eq} \leq 35$, depois dos modos horizontais, sendo o grupo com o segundo melhor desempenho. Nele, estão inclusos além dos novos iteradores, `succDiag` e `diagDown`.

A d_{Eq} para os pré-programados, após a 12ª equação ser adicionada, foi conservada entre 80 e 255 T_{Eq}, com `bottomUp` integrante, e entre 80 e 140 sem `bottomUp`, que mostra os maiores *gap's* entre expressões válidas. A existência de intervalos entre coleções de LI_{Eq} não necessariamente indica um aumento severo do tempo de simulação, visto que o `lbeST` considera as indeterminações $(0 = 0)$ na contagem das equações totais, que têm influência irrisória sobre ele, ou seja, em casos de redundância, nenhuma expressão é construída. Os interstícios em `bottomUp` apontam a passagem por diagonais de soma $n + m$ ímpar, nas quais todos os componentes são redundantes.

Além disso, os saltos tendem a intensificar-se com o crescimento de N, haja vista tensores maiores, com mais pontos inconformes a serem descartados. Um exemplo repentino de progressão de T_{Eq} ocorreu para `diagDown`, depois do 12ª termo, o que é justificável pela sua trajetória e o efeito assente em `MNP[w]`.

O Gráfico 4.27 incorpora a LI_{Eq} x T_{Eq} para `nc/c-ct`, registrando uma regularidade mais proeminente na inclinação das linhas e uma maior aglutinação de modos iteradores, senão `bottomUp`, `succRank` e `theSky`, assim descobrindo termos a uma taxa similar. Há uma equiparação entre o desempenho dos modos horizontais, os quais assimilam LI_{Max} a uma T_{Eq} inferior a 30, constituindo o resultado mais satisfatório no tocante a densidade de L.I.. Ademais, os 'saltos' na captação acima da 12ª equação, que se relacionam as refusões em N_{Max}, são corrigidos, portanto sua causa é atribuída ao mecanismo de verificação de independência linear. Apesar disso, os três modos com menor d_{Eq} mantiveram configuração parecida com a do Gráfico 4.27, exteriorizando acréscimos de três em LI_{Eq} entre os *gap*'s, que é a quantia média a ser conseguida por ordem N. `succRank` e `theSky` têm interrupções mais acentuadas devido à insistência em se percorrer caminhos, em $p = 0$, mesmo após o

Figura 4.25 – Comparação entre densidades finais d_{Eq} para os modos iteradores testados nas séries `nc/c-nct` e `nc/c-ct`, visto que o cachê não afeta o valor da variável. Uma maior d_{Eq} é resultado de uma menor quantidade de equações descartadas, que pode culminar em menor dispêndio de tempo na varredura.

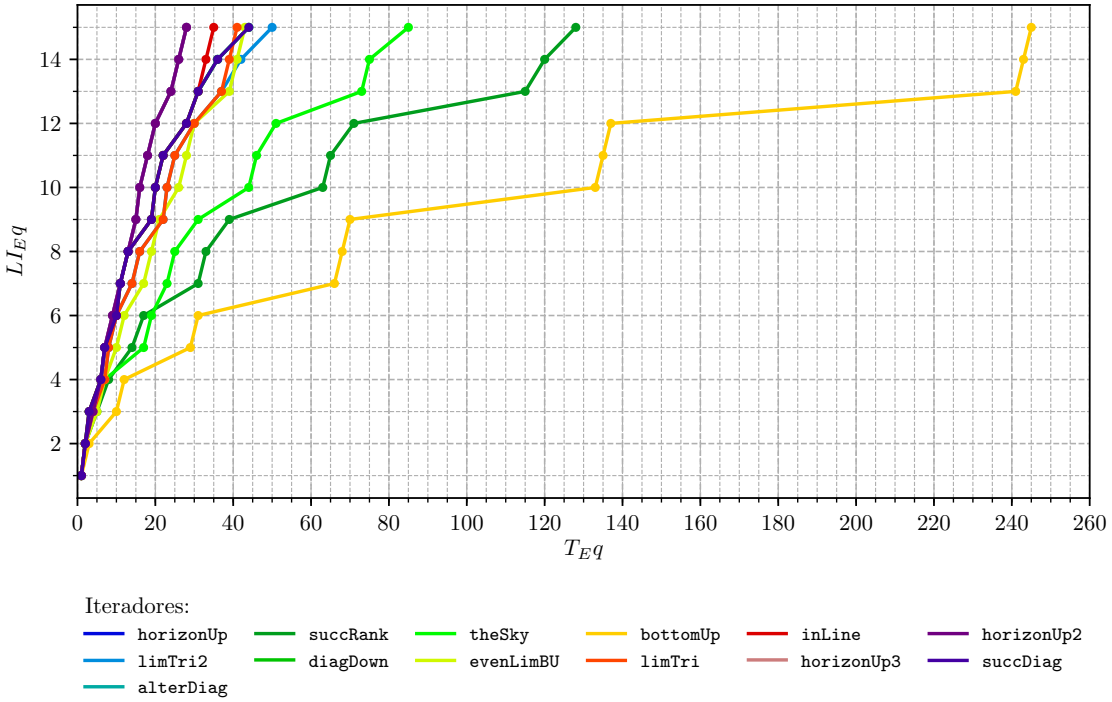


Fonte: Autor

esgotamento da ordem. `bottomUp`, além de acumular esse efeito, avança sobre diagonais em $p = 1$, conforme anteriormente comentado. No geral, todos os iteradores expressaram uma redução final de T_{Eq}, por exemplo para `bottomUp`, esta foi de 10 em comparação à `nc/c-nct`, de acordo com a Seção 3.5.2.

O `nc/c-ct` supera `nc/c-nct` no que concerne a d_{Eq} no momento da captura do 15º item, assim como prevê o Gráfico 4.25. O único modo a conservar mesma densidade nas duas séries foi `highOrdU`. A quantidade de traçados com mesma d_{Eq} é maior para `nc/c-ct`, como listado na Tabela 4.9. A Tabela 4.10 organiza de forma crescente os iteradores de acordo o ganho de d_{Eq} promovido pela checagem por linha não nula, medidos em relação à `nc/c-nct`.

Figura 4.26 – Razão L_{Eq} x T_{Eq} para as séries corrigidas `nc/c-ct` em simulações efetuadas sobre o *lattice* de sexta ordem `D2V81`. Os iteradores com melhor desempenho foram `horizonUp3` e `inLine`. Variável não afetada pelo cachê.

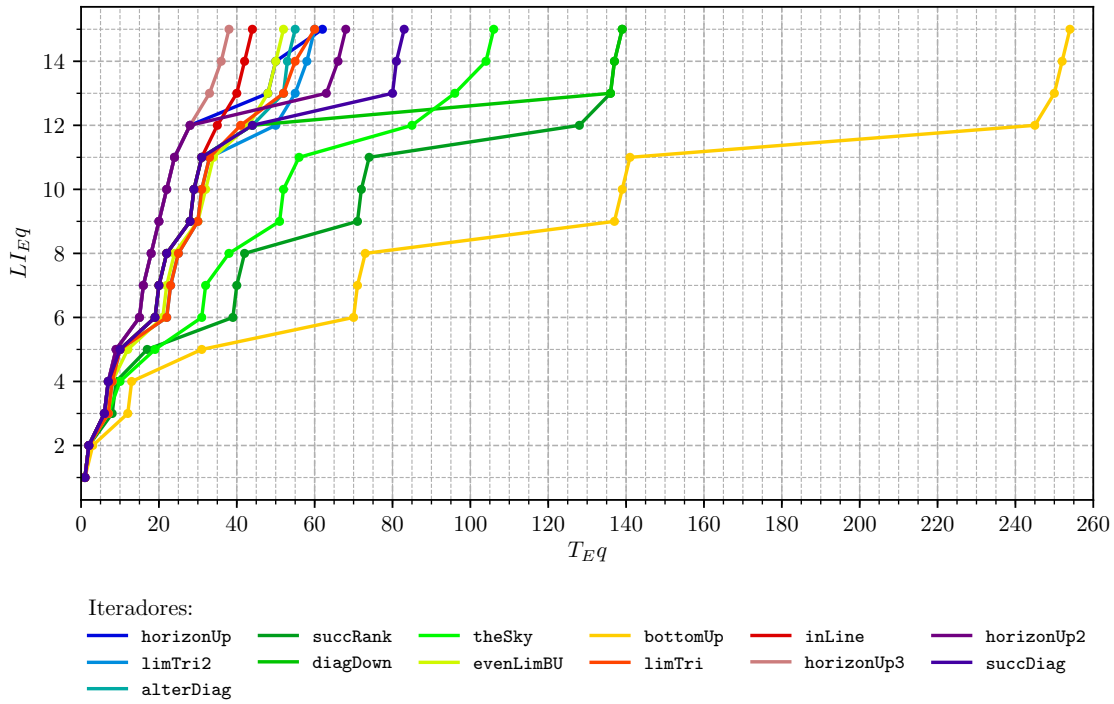


Fonte: Autor

Tabela 4.9 – Ranking de iteradores em função da d_Eq para as séries `nc/c-ct`, correspondendo a densidade de equações quando $LI_Eq = 15$ para o programa `lbeST`. O tempo normalizado em `c-ct` foi adicionado para comparação de performance.

Iterador	T. Normalizado	d_Eq
`horizonUp3`	1.00	0.54
`horizonUp2`	1.05	0.54
`horizonUp`	1.05	0.54
`inLine`	1.21	0.43
`limTri`	1.38	0.37
`evenLimBU`	1.20	0.35
`succDiag`	1.30	0.34
`alterDiag`	1.31	0.34
`diagDown`	1.41	0.34
`limTri2`	1.49	0.30
`theSky`	2.38	0.18
`succRank`	3.40	0.12
`bottomUp`	4.72	0.06

Figura 4.27 – Razão L_{Eq} x T_{Eq} para as séries não corrigidas `nc/c-nct` em simulações efetuadas sobre o *lattice* de sexta ordem `D2V81`. Destacam-se os desempenhos de `horizonUp3`, `horizonUp2` e `horizonUp`, descrevendo perfis idênticos. Variável não afetada pelo cachê.



Fonte: Autor

Tabela 4.10 – Ranking de iteradores em função da d_Eq para as séries `nc/c-nct`, correspondendo a densidade de equações quando $LI_Eq = 15$ para o programa `lbeST`. O tempo normalizado em `c-nct` foi adicionado para comparação de performance.

Iterador	T. Normalizado	d_Eq
`horizonUp3`	1.46	0.39
`inLine`	1.38	0.34
`evenLimBU`	1.48	0.29
`alterDiag`	1.83	0.27
`limTri2`	2.37	0.25
`limTri`	2.84	0.25
`horizonUp`	2.65	0.24
`horizonUp2`	2.91	0.22
`succDiag`	3.07	0.18
`theSky`	2.55	0.14
`succRank`	3.63	0.11
`diagDown`	5.62	0.11
`bottomUp`	4.94	0.06

4.3.4 Estudo dos *Benchmarks*

Os testes de *Benchmark* são afetados diretamente pela localização e densidade de equações linearmente independentes no plano de produto tensorial, devido a causas múltiplas, as quais figuram aumento do tempo, na perspectiva da construção dos polinômios tensoriais de Hermite, quando:

1. A ordem N aumenta, pois há um aumento generalizado da quantidade de derivadas parciais.

2. A ordem n e m individual de cada componente aumenta, caso haja expansão do polinômio em a após o produto dos termos.

Sobre a perspectiva da obtenção do termo proveniente do tensor de ortogonalidade, que compõe o lado direito das equações, há acréscimo de duração quando a ordem N e as chaves são simétricas, ou seja, são escolhidos componentes idênticos de dois tensores de Hermite de ordens iguais ($n = m$). Nesse caso, a operação no lado direito de 3.48 é realizada conforme 3.27, demandando de $n!$ permutações entre índices de operadores delta de Kronecker. Procedimento que é computacionalmente custoso, uma vez que a quantidade de permutações exequíveis em `Python` é limitada, ainda que com o módulo `itertools.permutations(iterable[, r])`. A construção de tensores de ortogonalidade por 3.27 foi completamente inviabilizada para $N > 11$ na sua implementação no programa `SimpleStencil`, devendo ser substituída pela Formulação **??**.

Também na perspectiva da verificação de independência linear, se:

1. A ordem N aumenta, porque é requisitado o armazenamento, substituição numérica e escalonamento matricial com tipo racionais, partindo de expressões maiores.

2. A d_Eq diminui, acarretando atraso pela verificação de maior quantidade de pontos inconformes, que era acentuado ainda mais pela remoção de equações L.D. não redundantes da matriz parcial do sistema na versão não modificada do programa-padrão.

3. O verificador de diagonal não nula, por desconsiderar exemplares válidos, demandando mais iterações.

E no que diz respeito às configurações iniciais do solver cachê inativo, é clara a existência de parâmetros que motivam morosidade na rotina de solução como um todo, como por exemplo, a simplificação de expressões simbólicas, estratégias de solução algébrica dos pesos, solução não numérica de a_{Eq}, *wrappers* númericos ou persistência de dados, mas que não atuam sobre essa amostragem de *Benchmark*, que examina somente o intervalo de tempo entre o início da exploração do plano até a formação completa do sistema.

O tempo real de captura para cada LI_{Eq} da série `nc-nct` está assinalado no Gráfico 4.31, revelando a distância de `bottomUp` em relação aos demais traçadores em termos de tempo de percurso, ocasionado majoritariamente pelo intervalo entre a $11^{\underline{a}}$ e a $12^{\underline{a}}$ chave, de $5^{\underline{a}}$ e $6^{\underline{a}}$ respectivas ordens.

A influência do número total T_{Eq} de inspeções feitas para aquisição de LI_{Max} foi prevista antecipadamente em 4.27. Com exceção dos modos `theSky`, `bottomUp` e `succRank`, os demais evoluem de forma parecida até $LI_{Eq} = 11$, com divergências que se acentuam após $LI_{Eq} = 12$. Conforme também observado nos gráficos do tipo LI_{Eq} x T_{Eq}, a inspeção por pontos localizados nas duas últimas ordens, determinam as maiores disparidades entre os iteradores. Assim a detecção dos quatro membros LI faltantes dos sistemas consumiu, em média, 65.00% da duração total das simulações executadas, alçando o pico de 86.92% para `diagDown`.

O `inLine` mostrou o melhor desempenho para essa série, destacando-se dos outros algoritmos em todo o trajeto, em especial quando $LI_{Eq} > 12$, também sendo aquele com distribuição mais balanceada no que condiz ao tempo gasto na $6^{\underline{a}}$ ordem. Isso contradiz a previsão no Gráfico 4.27 , favorável à `horizonUp`, corroborando a atuação de outros fatores além de d_{Eq}. Nessa mesma linha reside a equivalência, em *Benchmark*, entre `horizonUp2` e `succDiag`, apesar do segundo demandar de uma T_{Eq} maior.

No caso de cachê ativo `c-nct`, surgem algumas alterações, dentre as quais o fato de `diagDown` ultrapassar `bottomUp` em $LI_{Eq} > 12$. O comportamento atípico é justificado pela tendência em se realizar cerca de 90 verificações, com equações de $6^{\underline{a}}$ ordem já armazenadas, tardando o processo, uma vez que as checagens de L.I. não são inseridas no cache. Isso é diferente do que acontece com `bottomUp`, que efetua testes incrementando progressivamente a ordem dos elementos coletados no mesmo intervalo e faz poucas, e bem sucedidas, inspeções com equações da $6^{\underline{a}}$ ordem. Enfim, na circunstância do Gráfico **??** o aproveitamento pelo resgate de equações prontas, não compensou o tempo demandado pelo escalonamento matricial. Além disso em `c-nct`, `horizonUp3` obteve uma sensível vantagem em comparação com `inLine`, assim como `horizonUp2` face à `succDiag`.

Nas séries corrigidas para verificação por linha não nula, a disparidade entre as durações para as escolhas de LI_{Eq} finais reduziram significativamente, implicando na menor frequência de LD_{Eq}, portanto quase todos os caminhos foram finalizados em menos de 27 s para `nc-ct`. Os exemplares mais demorados, extrapoladores dos 27 s, exibiram pouca mudança no perfil temporal de captura, quando comparado com `c/nc-nct`. A diferença predominante reside no L.I. ao qual esses modos começam a se desvincular da ramificação da tendência de tempo principal, implicando na diminuição de uma LI_{Eq} coletada em N_{Max}.

Embora atenuada, as variações entre modos ainda são acentuadas progressivamente com a elevação de ordem, revelando a importância, outrora citada, de experimentações com altos valores de N, em especial para `c/nc-ct`. O `horizonUp3` desponta com a melhor performance, em limiar quase indistinguível de `inLine`. Dos elementos da ramificação

temporal principal, `evenLimBU` sustentou um desempenho menor, comparado aos outros, desvinculando-se antecipadamente em $LI_{Eq} = 6$. Apesar disso, quem encerrou com o resultado mais importuno foi `limTri2` com 33.30 s. Em geral *Benchmarks* `nc-ct`, excluídos os três mais lentos, exibiram tempo similar à `c-nct`, comprovando que as alterações na estratégia de checagem L.I. compensaram a ativação de cachê.

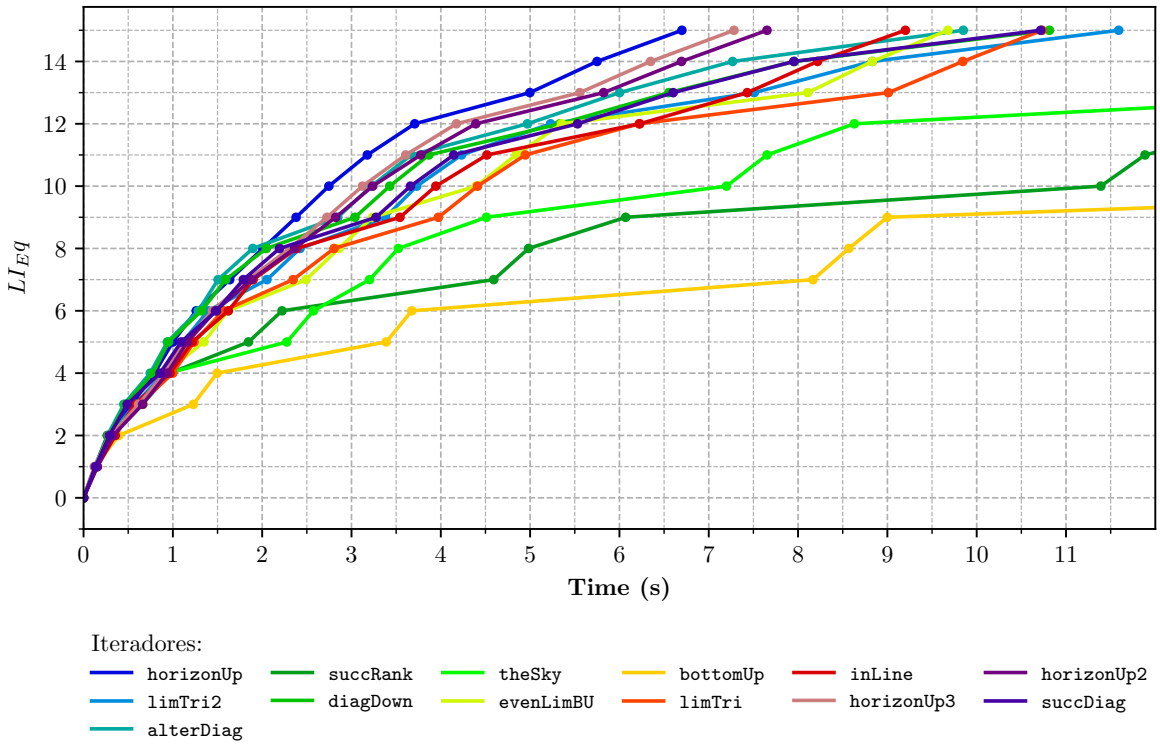
Como era previsto, `c-ct` foi a série mais rápida, com `bottomUp` concluído em 44.4 s e grande fração dos iteradores encerrada antes dos 12.5 s. `horizonUp` mostrou destaque a partir de $LI_{Eq} = 8$, com apenas 9.87 s de simulação, 462.17 % mais veloz que o mesmo modo na série mais lenta e 1542.0 % de `bottomUp`, membro mais demorado dessa série. Não houve atraso em `evenLimBU`.

Os *Benchmarks* correspondentes às simulações completas para todas as séries estão registrados no Gráfico 4.32, no qual é possível visualizar as reduções devido ao cachê ou correção do verificador de independência linear. As linhas vermelhas dispostas na extremidade de cada barra representa o desvio-padrão para essa amostra. Os valores de σ_{Max} e o σ_{Med} para todos os testes em ponto LI_{Eq} de captura, ambos para cada configuração de experimental, estão inseridos na Tabela .

Todos os iteradores com exceção de `diagDown`, cumpriram a tendência: `wTime[nc-nct]` $\leq$ `wTime[nc-ct]` $\leq$ `wTime[c-ct]` $\leq$ `wTime[c-nct]` na avaliação dos valores médios, desconsiderando os extremos do desvio padrão. Como era previsto, os modos com menor ganho de d_Eq foram aqueles com a mais baixa diminuição em `'wTime'`, abrangendo os casos mais lentos desenvolvidos por `bottomUp`, `succRank` e `theSky`. O último mostrou melhora mais acentuada nos casos de comparações entre séries com cachê inativo `nc-ct/nct`.

Diminuições percentuais do tempos de simulação de cachê ligado para desligado, fixada a estratégia de checagem L.I., registraram resultados em faixa que varia entre 50 a 72 %, com uma diferença pequena entre as séries de apurador diagonal e de linha não nula. O `theSky` foi o que experimentou redução maior em 12.02 % na comparação entre as duas séries `nct's` do que entre as `ct's`.

Figura 4.28 – Razão L_{Eq} x $t(s)$ para a série corrigida e com cachê ligado (`c-ct`) em simulações efetuadas sobre o *lattice* de sexta ordem `D2V81`. Os iteradores com melhor desempenho foram `horizonUp3` e `horizonUp2`. Os valores de tempo para $L_Eq = 15$ estão compreendidos no intervalo de $\pm 2\sigma_T$ sobre a média temporal para o modo registrada na Tabela 4.11.

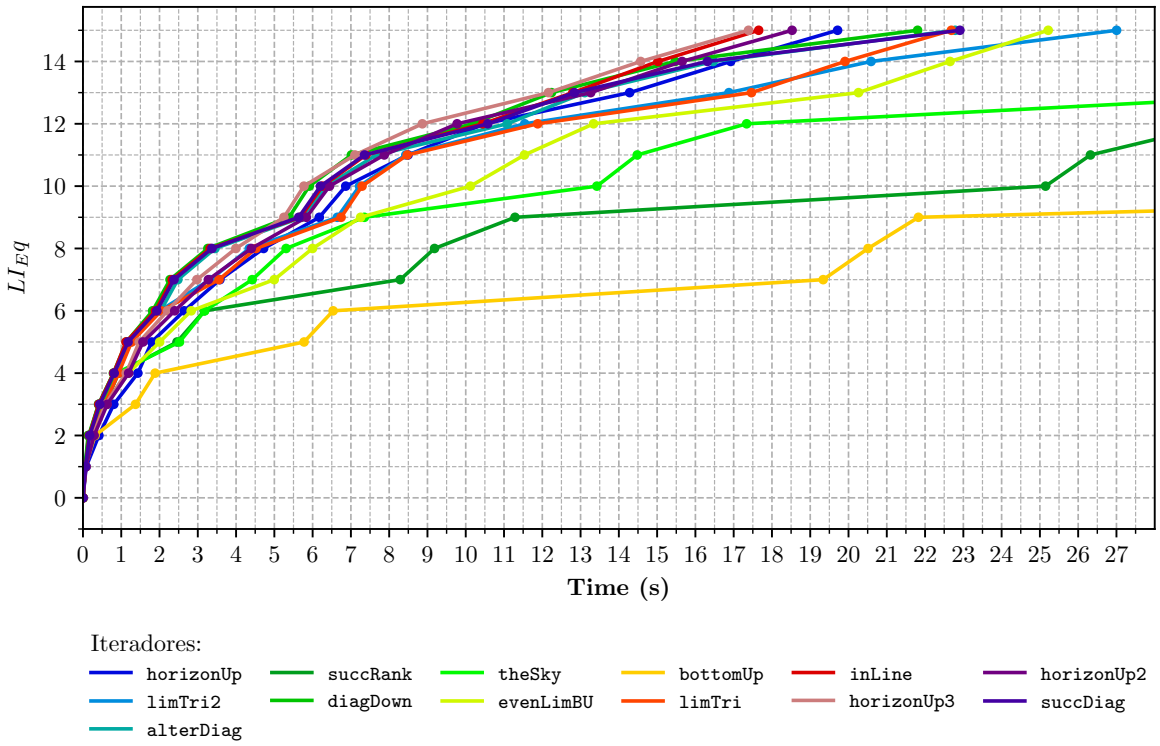


Fonte: Autor

Tabela 4.11 – Ranking de iteradores em função do tempo normalizado para a série `c-ct`, correspondendo aos melhores desempenhos de *benchmark* para o programa `lbeST`.

Iterador	Tempo (s)	σ_T	T. Normalizado	d_Eq
`horizonUp3`	9.41	1.316	1.00	0.54
`horizonUp2`	9.84	1.815	1.05	0.54
`horizonUp`	9.87	2.699	1.05	0.54
`evenLimBU`	11.31	1.194	1.20	0.35
`inLine`	11.41	1.980	1.21	0.43
`succDiag`	12.26	2.200	1.30	0.34
`alterDiag`	12.35	2.390	1.31	0.34
`limTri`	13.02	2.104	1.38	0.37
`diagDown`	13.24	2.757	1.41	0.34
`limTri2`	14.05	2.432	1.49	0.30
`theSky`	22.43	3.528	2.38	0.18
`succRank`	32.03	4.948	3.40	0.12
`bottomUp`	44.40	4.867	4.72	0.06

Figura 4.29 – Razão L_{Eq} x $t(s)$ para a série corrigida e com cachê desligado (`nc-ct`) em simulações efetuadas sobre o *lattice* de sexta ordem `D2V81`. Os iteradores com melhor desempenho foram `inLine` e `horizonUp3`. Os valores de tempo para $L_Eq = 15$ estão compreendidos no intervalo de $\pm 2\sigma_T$ sobre a média temporal para o modo registrada na Tabela 4.12.

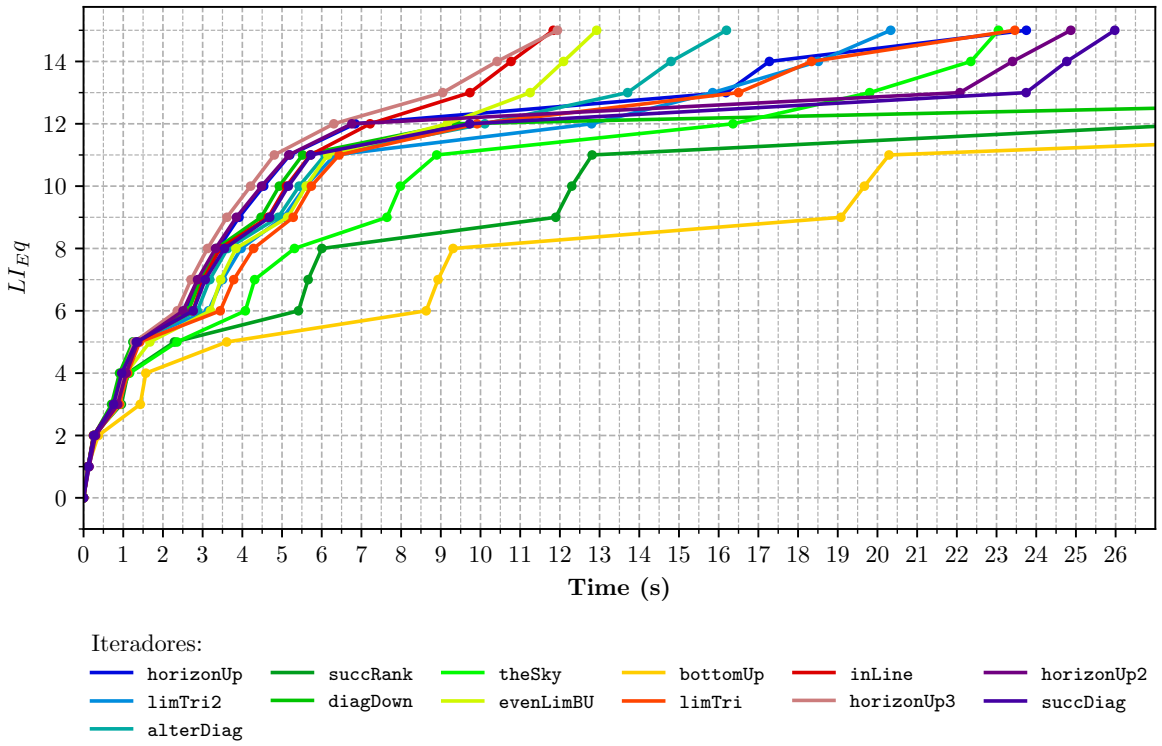


Fonte: Autor

Tabela 4.12 – Ranking de iteradores em função do tempo normalizado para a série `nc-ct`, correspondendo aos melhores desempenhos de *benchmark* para o programa `lbeST`.

Iterador	Tempo (s)	σ_T	T. Normalizado	d_Eq
`inLine`	27.10	8.587	2.88	0.43
`horizonUp3`	27.77	7.384	2.95	0.54
`alterDiag`	27.78	6.088	2.95	0.34
`horizonUp`	27.89	6.731	2.96	0.54
`horizonUp2`	27.96	7.463	2.97	0.54
`diagDown`	30.50	7.424	3.24	0.34
`succDiag`	30.91	6.274	3.28	0.34
`limTri`	32.49	8.333	3.45	0.37
`limTri2`	33.30	6.271	3.54	0.30
`evenLimBU`	35.59	7.303	3.78	0.35
`theSky`	51.89	8.374	5.51	0.18
`succRank`	78.70	8.699	8.36	0.12
`bottomUp`	133.65	9.867	14.20	0.06

Figura 4.30 – Razão L_{Eq} x $t(s)$ para a série não corrigida e com cachê ligado (`c-nct`) em simulações efetuadas sobre o *lattice* de sexta ordem `D2V81`. Os iteradores com melhor desempenho foram `inLine` e `horizonUp3`. Os valores de tempo para $L_Eq = 15$ estão compreendidos no intervalo de $\pm 2\sigma_T$ sobre a média temporal para o modo registrada na Tabela 4.13.

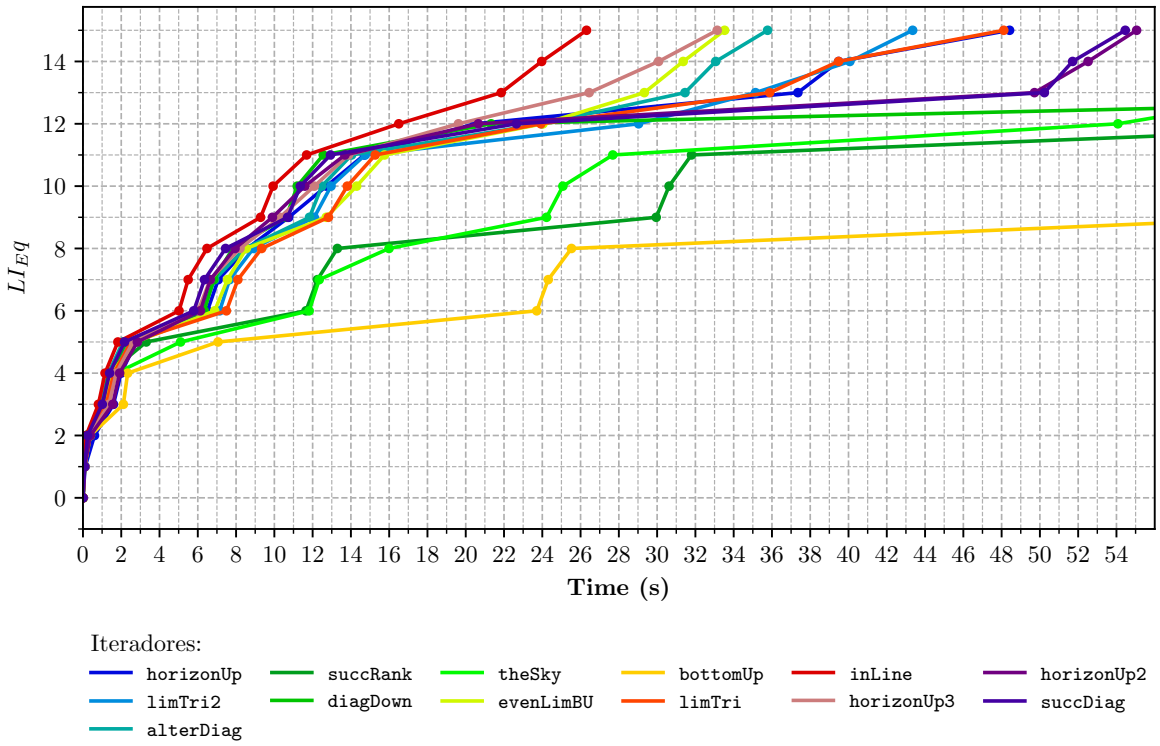


Fonte: Autor

Tabela 4.13 – Ranking de iteradores em função do tempo normalizado para a série `c-nct`, correspondendo aos melhores desempenhos de *benchmark* para o programa `lbeST`.

Iterador	Tempo (s)	σ_T	T. Normalizado	d_Eq
`inLine`	13.04	2.011	1.38	0.34
`horizonUp3`	13.77	2.079	1.46	0.39
`evenLimBU`	13.92	2.262	1.48	0.29
`alterDiag`	17.23	3.022	1.83	0.27
`limTri2`	22.34	3.474	2.37	0.25
`theSky`	23.98	3.015	2.55	0.14
`horizonUp`	24.91	3.935	2.65	0.24
`limTri`	26.71	6.316	2.84	0.25
`horizonUp2`	27.39	5.705	2.91	0.22
`succDiag`	28.91	6.451	3.07	0.18
`succRank`	34.16	6.669	3.63	0.11
`bottomUp`	46.46	7.437	4.94	0.06
`diagDown`	52.87	8.611	5.62	0.11

Figura 4.31 – Razão L_{Eq} x $t(s)$ para a série não corrigida e com cachê desligado (`nc-nct`) em simulações efetuadas sobre o *lattice* de sexta ordem `D2V81`. Os iteradores com melhor desempenho foram `inLine` e `horizonUp3`. Os valores de tempo para $L_Eq = 15$ estão compreendidos no intervalo de $\pm 2\sigma_T$ sobre a média temporal para o modo registrada na Tabela 4.14.

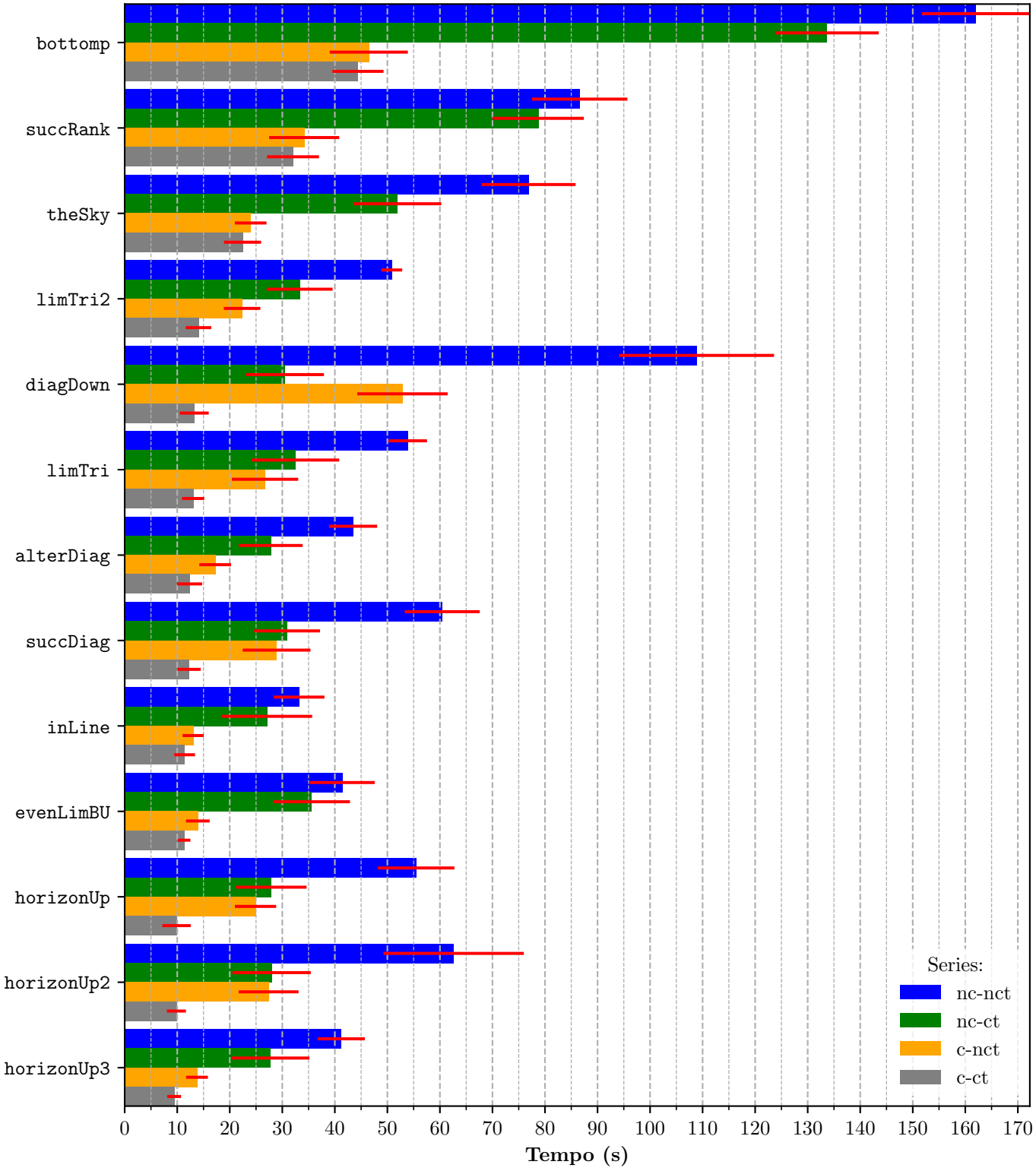


Fonte: Autor

Tabela 4.14 – Ranking de iteradores em função do tempo normalizado para a série `nc-nct`, correspondendo aos melhores desempenhos de *benchmark* para o programa `lbeST`.

Iterador	Tempo (s)	σ_T	T. Normalizado	d_Eq
`inLine`	33.19	4.834	3.53	0.34
`horizonUp3`	41.22	4.489	4.38	0.39
`evenLimBU`	41.40	6.192	4.40	0.29
`alterDiag`	43.48	4.582	4.62	0.27
`limTri2`	50.83	1.994	5.40	0.25
`limTri`	53.88	3.663	5.72	0.25
`horizonUp`	55.47	7.289	5.89	0.24
`succDiag`	60.43	7.154	6.42	0.18
`horizonUp2`	62.62	13.357	6.65	0.22
`theSky`	76.88	8.944	8.17	0.14
`succRank`	86.60	9.084	9.20	0.11
`diagDown`	108.85	14.750	11.56	0.11
`bottomUp`	162.02	10.264	17.21	0.06

Figura 4.32 – *Benchmark* total (s) dos modos iteradores simulados sobre o *lattice* `D2V81`, rankeados conforme ordem decrescente de desempenho na série `c-ct`. Destaca-se a performance dos modos `inLine` e `limTri2`. As barras vermelhas correspondem a um $\pm\sigma_T$ das amostras.



Fonte: Autor

4.4 Correlações entre chaves

A notação binária representa as possibilidades de derivação parcial nos eixos cartesianos no tocante as variáveis 'x' e 'y', respectivamente `'0'` e `'1'`, na construção do polinômio tensorial de Hermite descrito por 4.21 Assim uma composição da forma `'000111'`, equivale a:

$$H(\mathbf{v})_{xxxyyy} = 9xy - 3xy^3 - 3x^3y + x^3y^3 \tag{4.19}$$

Cada combinação retrata um componente de um tensor de ordem numericamente igual a quantidade de caracteres da chave. A saber, um tensor de Hermite na n-sésima ordem contém 2^n componentes em espaços de dimensão 2. Esse tensor é simétrico, ou seja, satisfaz:

$$A_{i_1 i_2 \cdots i_m} = A_{i_{\pi(1)} i_{\pi(2)} \cdots i_{\pi(m)}} \tag{4.20}$$

para qualquer permutação π de índices, devido ao teorema de Clairaut-Schwarz 4.21, que reduz o número de elementos examinados para $2^{(n/2)}$ se n par ou $2^{(n-1)/2}$ se n impar:

$$\nabla_n (\nabla_m f) = \nabla_m (\nabla_n f) \frac{\partial^2 f}{\partial x \partial y}(x,y) = \frac{\partial^2 f}{\partial y \partial x}(x,y) \tag{4.21}$$

Os componentes do tensor resultante da operação de produto tensorial são apontados como concatenações chaves, vinculadas aos dois tensores polinomiais participantes, por um ponto '.'. A operação é, por definição, comutativa e o referido tensor também é simétrico.

A identificação de otimizações, quanto a detecção de equações linearmente independentes não é restrita a alterações de caminhos, que sumariamente organizam agrupamentos de chaves, como também é influenciada por mudanças na sequência em que elas são inspecionadas, levando a uma melhor pormenorização das correlações, inclusive como justificativa da performance dos iteradores, a esse nível.

O programa-padrão monta, de acordo com uma entrada (n,m) as combinações de chaves, optando ou não por assumir hipóteses de simetria de tensores de Hermite, iniciando com a chave 'nula', que contêm apenas `0`'s, e intercambiando `1`'s até o ponto de parada. Se a hipótese `'hermite'` está acionada, esse ponto é $n/2$ para par e $(n-1)/2$ para impar, sendo n o comprimento delimitado para a string. Se $n = 6$:

```
   mdx = ndx = ['000000', '000001', '000011', '000111', '001111', '011111', '111111']
 'hermite' = False
```

Ou:

```
mdx = ndx = ['000000', '000001', '000011', '000111'], 'hermite' = True
```

A sequência dos produtos é determinada a partir de um loop duplo sobre duas listas de chaves, 'mdx' e 'ndx', com o primeiro indicando o termo à direita. Em um caso de ordem 6, no qual `len('ndx') = len('mdx') = 6` e com a hipótese '`hermite`' ativada o conjunto composto será:

Tabela 4.15 – Combinação de índices `'ndx'*'mdx'` que compõe a sequência de chaves a serem inspecionadas na coordenada $n = 6$ e $m = 6$. A tupla $(\partial y, \partial x)$ contabiliza a total de derivadas parciais realizadas em cada variável.

$(\partial y, \partial x)$	KY	$(\partial y, \partial x)$	KY
$(0,12)$	000000.000000	$(2,10)$	000011.000000
$(1,11)$	000000.000001	$(3,9)$	000011.000001
$(2,10)$	000000.000011	$(4,8)$	000011.000011
$(3,9)$	000000.000111	$(5,7)$	000011.000111
$(1,11)$	000001.000000	$(3,9)$	000111.000000
$(2,10)$	000001.000001	$(4,8)$	000111.000001
$(3,9)$	000001.000011	$(5,7)$	000111.000011
$(4,8)$	000001.000111	$(6,6)$	000111.000111

com destaque em vermelho para a chave que resultou em uma equação L.I. no modo '`succRank`'. Um conjunto `ndx`, mantidas as condições anteriores, mas em arranjo reverso, `ndxList = ndxList[::-1]`,

Tabela 4.16 – Combinação de índices `('ndx')⁻1*'mdx'` que compõe a sequência de chaves a serem inspecionadas na coordenada $n = 6$ e $m = 6$. `('ndx')⁻1` denota a reversão da lista `('ndx')`. A tupla $(\partial y, \partial x)$ contabiliza a total de derivadas parciais realizadas em cada variável.

$(\partial y, \partial x)$	KY	$(\partial y, \partial x)$	KY
$(3,9)$	000111.000000	$(4,8)$	000011.000011
$(4,8)$	000111.000001	$(5,7)$	000011.000111
$(5,7)$	000111.000011	$(1,11)$	000001.000000
$(6,6)$	000111.000111	$(2,10)$	000001.000001
$(2,10)$	000011.000000	$(3,9)$	000001.000011
$(3,9)$	000011.000001	$(4,8)$	000001.000111

As selecionadas para `ndx` não reverso e reverso em comparação são:

Tabela 4.17 – Chaves selecionadas com iterador `succRank'`, sem a reversão das listas `ndxList` no loop externo.

$(\partial y, \partial x)$	KY	$(\partial y, \partial x)$	KY
$(0,0)$	.	$(4,6)$	00011.00011
$(0,2)$	0.0	$(0,10)$	0000.000000
$(2,2)$	01.01	$(2,8)$	0000.000011
$(0,4)$	0.000	$(6,6)$	000111.000111
$(2,4)$	001.001	$(4,8)$	00001.0000111
$(4,4)$	0011.0011	$(2,10)$	00011.0000000
$(0,8)$	000.00000	$(4,8)$	00011.0000011
$(2,6)$	000.00011	-	

para o reverso na Tabela 4.17, e:

Tabela 4.18 – Chaves selecionadas com iterador `succRank'`, com a reversão `ndxList = ndxList[::-1]` das listas no loop externo.

$(\partial y, \partial x)$	KY	$(\partial y, \partial x)$	KY
$(0,0)$	.	$(4,6)$	0000.0011
$(0,2)$	0.0	$(2,8)$	00001.00001
$(2,2)$	01.01	$(0,10)$	00000.00000
$(0,4)$	00.00	$(6,6)$	000111.000111
$(2,4)$	001.001	$(4,8)$	000011.000011
$(4,4)$	0011.0011	$(2,10)$	000001.000001
$(0,8)$	0000.0000	$(4,8)$	000001.000111
$(2,6)$	0000.0011	-	

para o não reverso na Tabela 4.18.

O expoente $^{-1}$ denota reversão da lista '`ndx`'. A frequência de equações destacadas a partir de produtos entre dois termos iguais, tais que, '`ndx`' = '`mdx`' e '`mndx`' = '`ndx`'^{2} é maior para o segundo caso. Os *benchmarks*, contabilizados os demais modos iteradores, na circunstância de lista de chaves '`ndx`' reversa em `'c-nct'` não foram coletados por não serem relevantes na análise.

Uma importante conclusão inferida dessa experimentação é a de que equações frutos de produtos entre termos iguais procedentes de dois tensores polinomiais de Hermite tem grande chance de serem linearmente independentes entre si. Se uma lista `L` é pré-configurada com exemplares que atendam essa condição e inserida no programa-padrão para um teste de checagem L.I.:

Tabela 4.19 – Chaves da lista `L`. Todas as que satisfazem a condição `'ndx'*'mdx'='mdx'*'ndx'`.

$(\partial y, \partial x)$	KY	$(\partial y, \partial x)$	KY
$(0,0)$	.	$(0,8)$	0000.0000
$(0,2)$	0.0	$(4,6)$	00011.00011
$(2,2)$	01.01	$(2,8)$	00001.00001
$(0,4)$	00.00	$(6,6)$	000111.000111
$(2,4)$	001.001	$(4,8)$	000011.000011
$(4,4)$	0011.0011	$(2,10)$	000001.000001
$(2,6)$	0001.0001	$(0,12)$	000000.000000

Retorna 14 equações das 15 demandadas para o *lattice* de 6ª ordem `D2V81`, não integrando a coleção aquelas concernentes a `'000.000'` e `'0001.0001'`. Em uma averiguação realizada com a inserção de igualdades associadas a elementos de `L` como linhas de uma matriz M, cujas as colunas organizam os coeficientes dos w's, exibe exatamente `len(L-1)` linhas não nulas após ser escalonada e reduzida pela função `sympy.Matrix.rref`, sendo o correspondente a `'000000.000000'` a amostra linearmente dependente, que é um candidato a equação para 'a'.

4.4.1 Contagem de derivadas parciais

Definida uma função responsável por contabilizar para cada item de uma lista de chaves a frequência de '`0`''s e '`1`''s, isto é, das derivadas parciais executadas e retornar uma lista `testK` de tuplas `((n('1''s) ,n('0''s))` com as quantias armazenadas. A lista `KY` de `D2V81`, solucionado com o iterador `'evenLimBU'` em `'nc-nct'`, apresentará:

Tabela 4.20 – Lista `KY` de `D2V81`, solucionada com o iterador `'evenLimBU'` em `'nc-nct'`, conjugada a contagem da lista de tuplas de frequência de derivadas parciais.

$(\partial y, \partial x)$	KY	$(\partial y, \partial x)$	KY
$(0,0)$	.	$(4,6)$	.0000001111
$(0,2)$	.00	$(0,10)$	0.000000000
$(2,2)$	.0011	$(2,8)$	0.000000011
$(0,4)$	0.000	$(6,6)$	.000000111111
$(2,4)$	.000011	$(4,8)$	0.00000001111
$(4,4)$	.00001111	$(0,12)$	00.0000000000
$(0,8)$	0.0000000	$(2,10)$	00.0000000011
$(2,6)$	0.0000011	-	

e em `'nc-ct'`:

Tabela 4.21 – Lista `KY` de `D2V81`, solucionada com o iterador `'evenLimBU'` em `'nc-ct'`, conjugada a contagem da lista de tuplas de frequência de derivadas parciais.

$(\partial y, \partial x)$	KY	$(\partial y, \partial x)$	KY
$(0,0)$	.	$(4,6)$	.0000001111
$(0,2)$	.00	$(0,10)$	0.000000000
$(2,2)$	.0011	$(2,8)$	0.000000011
$(0,4)$	0.000	$(6,6)$	.000000111111
$(2,4)$	.000011	$(4,8)$	0.00000001111
$(4,4)$	.00001111	$(0,12)$	00.0000000000
$(0,8)$	0.0000000	$(2,10)$	00.0000000011
$(2,6)$	0.0000011	-	

e `testK = Counter((0, 0): 1, (2, 0): 1, (2, 2): 1, (4, 0): 1, (4, 2): 1, (4, 4): 1, (6, 0): 1, (6, 2): 1, (6, 4): 1, (6, 6): 1, (8, 0): 1, (8, 2): 1, (8, 4): 1, (10, 0): 1, (10, 2): 1)`

As coordenadas não se repetem até que $N = 6$ seja alcançado, revelando uma importante característica das equações linearmente independentes. Suponha que essa condição seja violada intencionalmente tal que `t[0] + t[1] != max(ORD)`, ou seja, que a soma dos valores presentes na tupla seja diferente da ordem máxima de inspeção do problema, então a matriz $\boldsymbol{M}$ de coeficientes do sistema linear será rank deficiente, o que é passível de ser demonstrado com `rref()`. Esse mesma análise desenvolvida em $\boldsymbol{L}$:

`testL = Counter((0, 0): 1, (2, 0): 1, (2, 2): 1, (4, 0): 1, (4, 2): 1, (4, 4): 1, (6, 0): 1, (6, 2): 1, (6, 4): 1, (6, 6): 1, (8, 0): 1, (8, 2): 1, (8, 4): 1, (10, 0): 1, (10, 2): 1, (12, 0): 1)`

evidencia a não repetição de quantidades de derivadas parciais. Esse resultado pode ser transformado em uma hipótese ótima generalizada, tal que:

Conjectura 1. Se um conjunto de n-chaves, associadas a n-pesos, formar uma lista de tuplas — de contagem de derivadas parciais — não repetidas, então a esse conjunto estarão associadas equações L.I. e a equação para 'a' será a tupla vinculada a uma chave inteiramente nula de comprimento igual ao valor da máxima ordem.

E por chave inteiramente nula convenciona-se aquela que é preenchida completamente por '0's, detendo apenas um tipo de derivação parcial. Em outras palavras, é condição suficiente para solucionar o problema, selecionar componentes de tensores que possuam somas de derivadas parciais, em cada variável, distintas entre si. A essa hipótese é concedida a denominação de "relação de chave de não repetição".

Disso é possível inferir o uso de produtos tensoriais como sendo facultativo, e que todas as equações são capazes de serem obtidas de um único polinômio tensorial de Hermite, fixando $n = 0$ e $H^n(x,y) = 1$. A validade desse resultado foi comprovada pelo escalonamento

concebido por `rref()` e vários casos foram simulados no programa `SimpleStencil` com sucesso na determinação dos conjuntos de peso e escala.

4.5 O Programa SimpleStencil

O `lbeST` exibiu limitações na resolução de *stencils* acima da 6ª ordem e restrições quando a velocidade no processamento de expressões simbólicas, devido a particularidades do próprio pacote `SymPy` e da linguagem de programação. A fim de que testes fossem elaborados acima dessa ordem e com melhor *Benchmark*, um programa novo foi concebido em `Python` 3.6, integrado a um *wrapper* de `symengine` e com uma abordagem de solução numérica diferente da apresentada no programa-padrão, denominado `SimpleStencil`. Além disso, sugestões de modificações no código do programa-padrão, que conduziriam a otimização de performance ou de ferramentas que permitiriam melhor estudo do problema foram incorporadas ao novo programa, contribuindo para os *Benchmarks* reportados na Tabela 4.22.

A aplicação do *wrapper* tem por finalidade agilizar a computação algébrica dos polinômios tensoriais, que anteriormente era executada por `SymPy`, escrito em `Python` puro, com a transição para um núcleo em `C++` das operações `Symengine` integrado ao núcleo do `SymPy`. A migração de objetos entre as bibliotecas é feita partir do módulo `.sympify()`. Embora algumas operações de expansão e simplificação sejam funcionais em objetos `SymPy`, no desenvolvimento do `SimpleStencil` concedeu-se preferência na manutenção da computação simbólica em `Symengine`, usufruindo do pacote mais lento apenas quando necessário, o que se resume ao módulo `sympy.Matrix.rref` no escalonamento reduzido de matrizes com coeficientes racionais.

O novo pacote não dispõe da mesma gama de funcionalidades que o antigo, tal qual um impasse ocorreu na manipulação de forma de objetos matriciais, no sentido da exclusão linhas ou colunas e na construção de submatrizes com entradas selecionadas. Funções foram inseridas ao programa a fim de suprir essa demanda. No âmbito numérico, houve predileção por uma estratégia que contemplasse a solução de um sistema de equações não lineares, que apesar de contraintuitiva é mais eficiente para o caso em estudo, em razão de restrições de performance inerente ao CAS. A matriz de coeficientes simbólicos do sistema, expressas como polinômios unidimensionais de a é convertida para uma função anônima lambda, com retorno de substituição numérica mais rápido, através do `symengine.Lamdify`.

Conforme experimentado no `lbeST`, existe uma quantidade máxima de caracteres ao qual a construção de uma função lambda se mostra temporalmente viável. Em geral, não é possível converter as soluções analíticas não simplificadas, produzidas da solução do sistema linear com coeficientes em a, pelo método de decomposição LU, acima da 5ª ordem. Um função definida com mecanismo de substituição numérica do próprio pacote atua até a 6ª ordem, a partir da qual fica incapacitado de detectar indeterminações, retornando "NaN"(*Not a Number*).

O módulo `scipy.optimize.fsolve()` consiste em um *wrapper* em torno de dos algoritmos `hydrd` e `hydrj` da MINPACK na linguagem FORTRAN. Na circunstância desse código, o algoritmo `hydrd` é utilizado e a raiz em n-variáveis é encontrada pelo método *powell hybrid* modicado, sendo o jacobiano então calculado por aproximação em diferenças finitas ou fornecido analiticamente quando possível.

A função alvo corresponde a $f(args) = M*W-B = 0$, em que `args = [w1,w2,w3... wn, a]` e `W` é o vetor dos pesos `W = [w1,w2,w3.., wn]`, partindo de uma estimativa inicial, que é convencionada como o ponto central de um intervalo em a, no qual um sistema linear a-aplicado, tem solução $wi > 0 \ \forall \ w_i \in W$. A tolerância `xtol` em termos de erro relativo entre duas iterações consecutivas é tomada como 10^{-12}.

Em suma, a nova abordagem numérica apresentou resultados compatíveis e pouco dissonantes da literatura, como registrado nas Tabelas 4.3, 4.4 e 4.6. A vantagem desse método é o retorno de resultado da última iteração, mesmo se as últimas forem chamadas dissidentes, permitindo inspeções sobre a convergência da escala do *lattice*.

Ao todo, três módulos foram programados na construção do sistema não linear: `lbSysWB`, `lbSysAll` e `LIASys`, cujas as atribuições se resumem em: compor um sistema $M*W-B=0$ com a equação de a inclusa, em que M é a matriz de coeficientes em a e B são os componentes selecionados do tensor de ortogonalidade, que foram originados via somatório de permutações de deltas de Kronecker, em conformidade com 3.48 e 3.27. E em sequência, fornecer a versão simbólica `Sys` desse sistema e a versão lambda de M, a segunda sem a equação linearmente dependente. `lbSysAll` recebe `Sys` e armazena na lista `lbSys` uma função lambda equivalente para cada linha dessa matriz. `LIASys`, por fim, compreende a função definida que provê o resultado da substituição numérica dos argumentos do sistema, por meio de um *loop* sobre os elementos de `lbSys`.

Tabela 4.22 – Comparação dos *benchmarks* totais em s para os *lattices* simulados nos programas `SimpleStencil` e `lbeST`, programado com iteradores `evenLimBU` para w_i's e `highOrdU` para a_{Eq}. O `lbeST` calculou exemplares inteiramente até a 6ª ordem, retornando somente o *set* L.I. para `D2V141` no tempo indicado por *.

N		`SimpleStencil` (s)	`lbeST` (s) `c-nct`
4	`D2V37`	1.1968	27.05
5	`D2V49`	1.6553	44.45
6	`D2V81`	2.8136	100.56
8	`D2V141`	5.9978	785.59*
9	`D2V169`	10.7228	-

Capítulo 5

Conclusão

Novas estratégias de combinações de índices na Equação de Hermite (3.48), culminaram no desenvolvimento de uma nova abordagem sobre o estudo da determinação de equações linearmente independentes, demandadas para o cálculo dos peso e escala para o conjunto de velocidades discretas inerente ao método das abcissas prescritas.

A concepção de um plano de produto tensorial permitiu uma projeção visual do problema, elucidando-o. Assim, formas de coletas de dados, estruturadas sobre variáveis do tipo *array* bidimensional, favoreceram análises para prever e registrar o comportamento de modos iteradores e orientar a programação de novos modos, de maneira a economizar tempo de simulação. A complexidade computacional vinculada à procedimento de varredura e checagem de independência linear forçou a reformulação de partes do programa `lbeST`, a fim de se reduzir ao máximo a quantidade de passos analíticos para a solução do problema, especialmente os que levavam a grandes e onerosas expressões para os pesos e a_{Eq} em função da escala `a`.

Com a concepção de um novo programa em Python$^{\text{TM}}$ 3.6.5 (PYTHON SOFTWARE FOUNDATION, 2018) a limitação de solução até a sexta ordem presente no programa-padrão pode ser superada e os valores inerentes ao lattice `D2V141` puderam ser calculados, mostrando-se compatíveis com a literatura. Não obstante, um novo lattice `D2V169` (9ª ordem) foi proposto, sendo aprovado nos testes de verificação, que foram executados pelo próprio programa `SimpleStencil`. O uso do `Symengine` com *wrapper* em C dispensou a necessidade de elaboração de um programa na linguagem `Julia` (JULIALANG, 2018), mostrando-se suficiente para atender os propósitos aqui demandados.

As duas maiores alterações do processo de obtenção dos *Stencils* em termos de programação foram: a adoção de um pacote de computação algébrica mais eficiente e a mudança do método de solução do sistema de linear com decomposição LU para inteiramente não linear com auxílio de pacote e método numérico pré-programado.

Também houve contribuição em se propor um nova conjectura para o cálculo dos componentes do tensor de ortogonalidade (Seção 3.1.1), pautada em análise combinatória, resultando em fatorial de índices que dispensa a necessidade das permutações de produtos

de delta de Kroenecker e de notação prévia.

As simulações foram separadas em quatro séries, considerando a ativação e desativação do cachê do `lbeST` e duas abordagens de verificação de independência linear: a que analisa apenas a diagonal principal da matriz de coeficientes em a escalonada — forma anterior do programa — e a que releva todas as linhas não-nulas — forma proposta. Por razão não apurada, o `lbeST` não é capaz de completar o processo de solução, quando é programada com a segunda estratégia.

Os modos iteradores com melhor *benchmark* correspondem a série com cachê ativo e verificador corrigido são: `horizonUp3` e `horizonUp2`. Quando o cachê é desligado, estes são `inLine` e `horizonUp3`. Os caminhos com melhor densidade de equações selecionados por ponto são: `horizonUp3`, `horizonUp2` e `horizonUp`.

No estudo da distribuição de candidatos à a_{Eq} no plano de produto tensorial foi possível descobrir que chaves tidas como 'nulas' — derivadas parciais em uma única variável — são preponderantemente sinalizadas para quase todos os modos de captura LI. Se o verificação corrigida é acionada, o padrão é inteiramente composto por tais chaves, que são distribuídas singularmente para cada ponto na diagonal de limite de ordem.

De acordo com uma prospecção mais detalhada das chaves uma hipótese de relação de chaves de 'não repetitividade' — selecionar todos os produtos de componentes de tensores, cuja as somas das quantidade de derivação parcial em cada variável sejam distintas entre si — e outra de 'paridade' — para *lattices* simétricos, $m+n$ impar retornará uma indeterminação — foram propostas, reduzindo de forma eficaz a complexidade computacional do problema.

Ainda há muito o que ser feito no âmbito de pesquisa sobre aperfeiçoamentos de métodos de discretização dos espaços de velocidade, principalmente no que diz respeito a migração para linguagens com maior capacidade de armazenamento de dígitos de ponto flutuante e com melhor tempo de execução de operações simbólicas.

Os estudos sobre contagem de derivadas parciais a partir das chaves foram desenvolvidos majoritariamente por experimentação numérica, carecendo de demonstração matemática apropriada, que poderá, por sugestão, ser efetuada considerando as relações de recorrência existentes entre polinômios de Hermite tensorial e que está documentada na literatura de Mattila, Hegele Jr. e Philippi (2014).

Além disso, nenhum *Stencil* aqui simulado — ou descoberto — foi validado em uma situação prática de aplicação do método LBE, persistindo dúvidas quanto ao uso da velocidade térmica de referência na escalabilidade de *lattices* designados a fenômenos isotérmicos em detrimento ao uso da velocidade do som. À saber, para gases ideais: $c/c_T = \gamma^{-1/2}$, γ é o coeficiente de expansão isentrópica

Também cabe a próxima etapa de desenvolvimento, pensar em formas de derivação de novos *Stencils* de mais alta ordem, de forma a adaptar o `SimpleStencil` com a instrumentação necessária para que ele cumpra a função maior a qual fora designado.

Referências

ANDRADE, F. N. d. Trabalho de Conclusão de Curso, *Sistemática de obtenção de métodos de lattice Boltzmann pelo método das abcissas prescritas*. 2016. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/8774>. Citado 2 vezes nas páginas 31 e 61.

BAZARIN, R. L. M. *Estudo numérico do processo de deslocamento de fluidos em meio poroso heterogêneo usando o método de Lattice Boltzmann*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná. Curitiba, 2018. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/3422>. Citado 3 vezes nas páginas 21, 28 e 31.

EINSTEIN, A. Die grundlage der allgemeinen relativitätstheorie. *Annalen der Physik*, Wiley Online Library, v. 354, n. 7, p. 769–822, 1916. Citado na página 37.

GUO, Z.; SHU, C. *Lattice Boltzmann method and its applications in engineering*. [S.l.]: World Scientific, 2013. v. 3. Citado na página 31.

HARRIS, S. *An introduction to the theory of the Boltzmann equation*. [S.l.]: Courier Corporation, 2004. Citado na página 25.

HE, X.; LUO, L.-S. Theory of the lattice boltzmann method: From the boltzmann equation to the lattice boltzmann equation. *Physical Review E*, APS, v. 56, n. 6, p. 6811, 1997. Citado na página 29.

JULIALANG. *The Julia Programming Language*. 2018. Disponível em: <https://julialang.org/>. Acesso em: 01 nov. 2018. Citado 2 vezes nas páginas 41 e 105.

KREMER, G. M. *Introdução à Equação de Boltzmann, Uma*. [S.l.]: Edusp, 2005. Citado na página 28.

MATTILA, K. K.; HEGELE JR., L. A.; PHILIPPI, P. C. High-accuracy approximation of high-rank derivatives: Isotropic finite differences based on lattice-boltzmann stencils. *The Scientific World Journal*, v. 2014, p. 1–16, January 2014. Article ID 142907. Disponível em: <http://dx.doi.org/10.1155/2014/142907>. Citado 12 vezes nas páginas 21, 22, 29, 30, 33, 34, 36, 38, 52, 57, 66 e 106.

NAAKTGEBOREN, C. Deduções do orientador. 2019. Citado 3 vezes nas páginas 13, 39 e 40.

NEMO. *AbstractAlgebra.jl*. 2018. Disponível em: <https://nemocas.github.io/AbstractAlgebra.jl/>. Acesso em: 01 nov. 2018. Citado na página 42.

PHILIPPI, P. C. et al. From the continuous to the lattice boltzmann equation: The discretization problem and thermal models. *Phys. Rev. E*, American Physical Society, v. 73, p. 056702, May 2006. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevE.73.056702>. Citado 2 vezes nas páginas 38 e 61.

PYTHON SOFTWARE FOUNDATION. *Python*^{TM}. 2018. Disponível em: <https://nemocas.github.io/AbstractAlgebra.jl/>. Acesso em: 01 nov. 2018. Citado 2 vezes nas páginas 41 e 105.

SHAN XUE-FENG YUAN, H. C. X. Kinetic theory representation of hydrodynamics: a way beyond the navier–stokes equation. *Journal of Fluid Mechanics*, v. 550, p. 413–441, March 2006. Disponível em: <http://dx.doi.org/10.1017/S0022112005008153>. Citado na página 22.

SYMENGINE. *Julia Wrappers for SymEngine, a fast symbolic manipulation library, written in C++*. 2018. Disponível em: <https://github.com/symengine/SymEngine.jl>. Acesso em: 01 nov. 2018. Citado na página 42.

THE OEIS FOUNDATION INC. *The on-line encyclopedia of integer sequences*$^{(R)}$. 2019. Disponível em: <https://oeis.org/A007530>. Acesso em: 12 abr. 2019. Citado 2 vezes nas páginas 39 e 54.