UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

INTERAÇÕES FLUIDO-SÓLIDO VIA MÉTODO LATTICE BOLTZMANN E ELEMENTOS DISCRETOS

JOAQUIM ARAÚJO COSTA NETO

ORIENTADOR: MÁRCIO MUNIZ DE FARIAS, Ph.D.

DISSERTAÇÃO DE MESTRADO EM GEOTECNIA

PUBLICAÇÃO: G.DM-343/20

BRASÍLIA/DF: FEVEREIRO DE 2020

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

INTERAÇÕES FLUIDO-SÓLIDO VIA MÉTODO LATTICE BOLTZMANN E ELEMENTOS DISCRETOS

JOAQUIM ARAÚJO COSTA NETO

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA CIVIL DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRÁU DE MESTRE.

APROVADA POR:

_____

MÁRCIO MUNIZ DE FARIAS, PhD (ENC/UnB)

(ORIENTADOR)

_____

LEANDRO LIMA RASMUSSEM (UnB)

(EXAMINADOR INTERNO)

_____

CARLOS ALEXANDER RECARREY MORFA, Dr. Ing. (UCLV)

(EXAMINADOR EXTERNO)

DATA: BRASÍLIA/DF, 18 DE FEVEREIRO DE 2020

FICHA CATALOGRÁFICA

COSTA NETO, JOAQUIM ARAÚJO

Interações fluido-sólido via método lattice Boltzmann e elementos discretos . [Distrito Federal] 2020

xviii, 101 p., 210x297 mm (ENC/FT/UnB, Mestre, Geotecnia, 2020)

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia

Departamento de Engenharia Civil e Ambiental

Palavras chaves:

1. Interação Fluido-Solido
2. Método Lattice Boltzmann
3. Método dos Elementos Discretos

I. ENC/FT/UnB II. Mestre

REFERÊNCIA BIBLIOGRÁFICA

COSTA NETO, J. A. (2020). Interações Fluido-Sólido via Método Lattice Boltzmann e Elementos Discretos. Dissertação de Mestrado, Publicação G.DM-343/20, Departamento de Engenharia Civil e Ambiental, Universidade de Brasília, Brasília, DF, 101 p.

CESSÃO DE CRÉDITOS

NOME DO AUTOR: Joaquim Araújo Costa Neto
TÍTULO DA DISSERTAÇÃO DE MESTRADO: Interações Fluido-Sólido via Método Lattice Boltzmann e Elementos Discretos
GRAU/ANO: Mestre/2020

_____

Joaquim Araújo Costa Neto

Universidade de Brasília, Secretaria da Coordenação de Pós-Graduação em Geotecnia
Campus Darcy Ribeiro, Departamento de Engenharia Civil e Ambiental. Prédio SG-12, Universidade de Brasília. CEP 70910-900 – Brasília, DF – Brasil

engcjoaquim@gmail.com

DEDICATÓRIA

Dedico esta dissertação

ao meu pai Evaristo e minha mãe Maria.

AGRADECIMENTOS

À Deus e aos bons espíritos que me guiaram durante essa caminhada. Foi um trabalho muito desafiador e sem o apoio deles durantes os momentos de dificuldades com certeza não teria conseguido.

Aos meus pais, Maria e Evaristo, que foram minha fundação durante essa caminhada, sempre me dando o apoio necessário e me aconselhando nos momentos de desespero.

Ao meu orientador, Professor Márcio Muniz, que sempre me aconselhou e me guiou para o desenvolvimento desta pesquisa e que acima de tudo se tornou um grande amigo.

Ao Professor Leandro Lima que sempre me aconselhou e me engrandeceu com grandes conversas, além de aturar todas as minhas dúvidas sobre programação.

A todos os funcionários e amigos feitos no INFRALAB durante este período, muito obrigado pelas conversas e pelo o apoio durante o desenvolvimento desta pesquisa.

A todos os meus amigos do mestrado da UnB muito obrigado pelas conversas, pelas risadas e pelos conselhos, com certeza minha vida ficou um pouco mais fácil com a presença de vocês.

A todos os meus amigos de infância da minha querida São Bento muito obrigado por sempre estarem do meu lado mesmo distante fisicamente e por sempre me levantar com uma palavra amiga nos momentos mais difíceis.

À CAPES e ao CNPq por terem me concedido essa bolsa de estudo que sem a qual eu jamais conseguiria terminar esse mestrado, muito obrigado por acreditarem na educação como o fator transformador desse mundo.

Interações Fluido-Sólido via Método Lattice Boltzmann e Elementos Discretos

RESUMO

A pesquisa tem como objetivo o estudo bidimensional da interação entre fluido e partículas sólidas por meio de um código computacional dos métodos lattice Boltzmann e elementos discretos. O código foi desenvolvido utilizando a linguagem C++ devido a sua maior rapidez e por ser uma linguagem orientada a objetos. O método lattice Boltzmann implementado faz uso do operador de colisão BGK com um único tempo de relaxação e discretização do domínio feita por meio de células lattice tipo D2Q9. O método dos elementos discretos (MED) utilizado é o mesmo proposto por Cundall & Strack com a adoção de partículas circulares ou discos para simular o meio discreto. Ambos os métodos foram validados por meio de problemas de referência amplamente estudados e de solução acurada. No método lattice Boltzmann foram feitas simulações de fluxo Poiseuille e fluxo ao redor de um cilindro onde foi verificada a capacidade geral do método de simular fluxos incompressíveis e capturar os diferentes tipos de regimes desenvolvidos em um canal em função do número de Reynolds. Já o método dos elementos discretos foi validado e teve parâmetros calibrados por meio da análise da energia de uma partícula em queda livre e da conservação de momento linear na colisão entre partículas, se mostrando bastante eficaz em ambos os casos. O acoplamento dos métodos foi feito por meio do método das fronteiras móveis devido a sua facilidade de implementação e conservação da localidade das operações e teve sua validação por meio da análise do coeficiente de arrasto desenvolvido na partícula. Os valores do coeficiente de arrasto em função do número de Reynolds se mostraram totalmente em conformidade com a solução analítica para números de Reynolds maiores que 60 e apresentaram valores subestimados para números de Reynolds menores, porém o método ainda se mostrou eficaz. Por fim, o código desenvolvido é uma ferramenta com um grande potencial para simulação de meios particulados em contato com um fluido e avaliar as nuances de suas interações.

Fluid-Solid Interactions via Lattice Boltzmann and Discrete Element Methods

ABSTRACT

The research aims at the two-dimensional study of the fluid-solid interaction by means of a computational code of the lattice Boltzmann and the discrete element methods. The code was developed using the C++ language due to its greater speed and for being an object-oriented language. The lattice Boltzmann method implemented makes use of the BGK collision operator with a single relaxation time and the domain discretization was done with D2Q9 lattice cells. The discrete element method (DEM) implemented was the same proposed by Cundall & Strack with the adoption of circular particles or discs to simulate the discrete medium. Both methods were validated by means of widely studied reference cases with accurate analytical solution. For the lattice Boltzmann method, simulations of Poiseuille flow and flow around a cylinder were performed, in which the general capacity of the method to simulate incompressible flows and capture the different types of regimes developed in a channel as a function of the Reynolds number were evaluated. The discrete element method was validated and had parameters calibrated by analyzing the energy of a particle in free fall and the conservation of linear momentum in the collision between particles, proving to be quite effective in both cases. The coupling of these methods was done using the immersed moving boundary method due to its ease of implementation and conservation of the locality of the operations and had its validation done with an analysis of the drag coefficient developed by the particle. The values of drag coefficient as a function of Reynolds number proved to be totally in accordance with the analytical solution for Reynolds number greater than 60 and showed underestimated values for smaller Reynolds numbers, however the method still proved to be effective. Finally, the developed code is a tool with great potential for simulating particulate media in contact with a fluid and assessing the nuances of their interactions.

ÍNDICE

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

LISTA NOMENCLATURA E ABREVIAÇÕES

D2Q9	Malha lattice bidimensional com nove velocidade discretas
et al.	E outros
IDE	Integrated development environment
IMB	Immersed moving boundary
kg	Quilograma
LBB	Linked bounce-back
m	Metros
MDF	Método das diferenças finitas
MED	Método dos elementos discretos
MEF	Método dos elementos finitos
MFM	Método das fronteiras móveis
MLB	Método lattice Boltzmann
MVF	Método dos volumes finitos
Pa	Pascal
s	Segundos
VTK	Visualization toolkit
YADE	Yet Another Dynamic Engine
2D	Bidimensional
3D	Tridimensional

LISTA DE SÍMBOLOS

a	Metade do canal de estudo do fluxo Poiseuille
Bn	Função de peso da fração sólida
c	Menor coesão do contato partícula-partícula
C	Velocidade lattice
C_d	Coeficiente de arrasto
$\mathbf{c_i}$	Vetor de velocidades discretas
c_s	Velocidade do som no método lattice Boltzmann
D	Distância entre o contato partícula-partícula ou partícula parede
d	Diâmetro da partícula sólida
$\mathbf{F}$	Vetor de força
$\mathbf{F_c}$	Vetor de força do contato partícula-partícula
$\mathbf{F_f}$	Vetor de força hidrodinâmica atuante na partícula
f_i	Função de distribuição de partículas do fluido
f_i,	Função de distribuição de partículas na direção oposta a f_i
f_i^{eq}	Função de distribuição no equilíbrio
F_n	Força normal aplicada na partícula
$F_{s,max}$	Máxima força cisalhante do contato partícula-partícula
$\mathbf{F_v}$	Vetor de força de amortecimento local não-viscoso
Fx	Componente horizontal da força hidrodinâmica
G	Gradiente de pressão linear ou gravitacional
$\mathbf{G}$	Força da gravidade
I	Momento de inércia da partícula
k	Rigidez do sistema de partículas
k_n	Rigidez normal do contato partícula-partícula
k_s	Rigidez cisalhante do contato partícula-partícula
m	Massa da partícula
Ma	Número Mach
n_{sub}	Número de subciclos do método dos elementos discretos
p	Pressão exercida pelo fluido
R	Raio da partícula
Re	Número de Reynolds
S_i	Termo de força externa do método lattice Boltzmann
t	Tempo
t^+	Tempo após aplicação do processo de colisão de partículas
$\mathbf{T_c}$	Vetor de torque do contato partícula-partícula
$\mathbf{T_f}$	Vetor de torque hidrodinâmico atuante na partícula
$\mathbf{T_v}$	Vetor de torque de amortecimento local não-viscoso
$\mathbf{u}$	Vetor de velocidade macroscópica do fluido
$\mathbf{u_0}$	Vetor de velocidade inicial do fluido
U_n	Interpenetração do contato partícula-partícula ou partícula-parede
$\mathbf{u^{eq}}$	Vetor de velocidade do fluido no equilíbrio
u_{max}	Máxima velocidade no canal
$u(y)$	Perfil de velocidade do fluxo Poiseuille
$\mathbf{v}$	Vetor de velocidade translacional da partícula
$\mathbf{v_b}$	Vetor de velocidade do contorno
v_{max}	Máxima velocidade do fluido

$\mathbf{v_t}$	Vetor de velocidade translacional
$\mathbf{x}$	Vetor de posição
$\mathbf{x_c}$	Vetor de posição do centro da partícula
$\mathbf{x_i}$	Vetor de posição do centro da célula lattice
y	Posição vertical do canal de estudo do fluxo Poiseuille
α	Coeficiente de amortecimento local
ΔF_n	Incremento de força normal
ΔF_s	Incremento de força cisalhante
ΔU_n	Incremento de deslocamento relativo normal
ΔU_s	Incremento de deslocamento relativo cisalhante
Δt	Passo de tempo
Δt_{crit}	Passo de tempo crítico do método dos elementos discretos
Δt_{MED}	Passo de tempo do método dos elementos discretos
Δt_{MLB}	Passo de tempo do método lattice Boltzmann
ε	Fração sólida
$\boldsymbol{\theta}$	Vetor de rotação
λ	Fator de segurança do passo de tempo crítico
N	Viscosidade cinemática do fluido
ρ	Densidade do fluido
ρ_0	Densidade inicial do fluido
τ	Tempo de relaxação
φ_u	Menor ângulo de atrito do contato partícula-partícula
$\boldsymbol{\omega}$	Vetor de velocidade angular da partícula
ω_i	Peso correspondente ao nó da malha lattice
Ω_i	Operador de colisão BGK
Ω_i^s	Operador de colisão do método das fronteiras móveis

CAPÍTULO I

1 INTRODUÇÃO

O fluxo de fluidos em meios porosos é um fenômeno muito comum na natureza e pode ser encontrado em diversas atividades ligadas à engenharia. É de fundamental importância entender como esse fenômeno físico se desenvolve para poder, de forma eficaz, desenvolver estruturas com maior durabilidade e segurança. Nas aplicações da engenharia e particularmente na engenharia geotécnica, este tema é relevante em várias frentes, por exemplo, na avaliação de volumes de infiltração, análise da erosão, previsão e evolução de plumas de contaminantes, análise da sucção, dentre muitos outros (Ocampo-Gómez, 2013).

Para compreender este fenômeno é necessário entender como se dá a interação entre as partículas sólidas e o fluido. Assim, é necessário fazer estudos com base numa escala adequada. Uma abordagem microscópica é complexa de ser simulada devido à complexidade da geometria e à impossibilidade de descrever o comportamento de cada partícula de fluido. Tradicionalmente, estas interações são estudadas por meio de uma abordagem macroscópica utilizando os métodos elementos finitos (MEF), volumes finitos (MVF) ou diferenças finitas (MDF), porém o grande custo computacional em função da complexidade da geometria e condições de contorno acaba por dificultar o uso de uma abordagem discreta.

Uma abordagem mesoscópica, ou seja, escala dos grãos parece ser adequada para analisar a interação fluido-sólido. Um dos métodos baseados nessa abordagem que vem sendo amplamente utilizado é o Método Lattice Boltzmann (MLB). Este método consegue recuperar a solução das equações de Navier-Stokes para fluidos incompressíveis, descrevendo o comportamento mesoscópico do fluido a partir da distribuição das partículas ao invés de considerar a velocidade e a densidade macroscópica ou o comportamento individual das partículas (Gardner & Sitar, 2019). A superioridade deste método decorre da sua simplicidade, o que possibilita a simulação de geometrias complexas.

Além da interação fluido-sólido, deve ser avaliada também a interação entre as partículas sólidas para assim prever, de forma mais realista, o comportamento do fluxo de fluidos em meios porosos. Um dos métodos que têm sido aplicado no estudo de matrizes porosas é o Método dos Elementos Discretos (MED), proposto por Cundall (Cundall,1971;

Cundall & Strack, 1979). A formulação do MED considera o movimento de cada partícula individualmente, incorporando a interação entre as partículas por meio das forças de contato. Uma vez que a força de contato entre as partículas é estabelecida, o movimento de cada partícula pode ser atualizado independentemente (Gardner & Sitar, 2019).

1.1 IDENTIFICAÇÃO DO PROBLEMA

O problema científico a ser investigado por este trabalho se resume no seguinte questionamento: É possível simular fluxo de fluidos em meios porosos para entender como se dá a interação fluido-sólido e obter parâmetros intrínsecos da matriz porosa por meio de uma metodologia híbrida dos métodos lattice Boltzmann e elementos discretos?

1.2 JUSTIFICATIVA

O presente trabalho se insere no estudo e compreensão da interação entre fluido e partículas sólidas durante o fluxo de fluidos em meios porosos. Este fenômeno é um problema complexo devido às particularidades geométricas do meio, assim, um melhor entendimento numérico destas interações pode propiciar a obtenção de parâmetros intrínsecos de matrizes porosas de forma mais rápida, além de poder prever o comportamento macroscópico dos arranjos sólidos analisados sob diversas condições.

1.3 HIPÓTESE

A principal hipótese deste trabalho é de que uma metodologia híbrida dos métodos lattice Boltzmann e Elementos Discretos pode produzir uma ferramenta poderosa capaz de simular problemas de fluxo de fluidos em meios porosos e obter parâmetros intrínsecos dos arranjos sólidos analisados.

1.4 OBJETIVOS

O objetivo deste trabalho é implementar uma metodologia híbrida da combinação dos métodos lattice Boltzmann e elementos discretos por meio de um código computacional para ser utilizado em simulações de problemas de fluxo de fluidos em meios porosos e estudar a interação fluido-sólido.

Entre os demais objetivos desse trabalho, constam:

- Desenvolver e validar um código computacional dos métodos lattice Boltzmann e Elementos Discretos para ser utilizado em simulações de fluxo de fluidos em meios porosos;
- Simular problemas de transporte e deposição de sedimentos por arraste e fluxo de areia para estudar a interação fluido-sólido;
- Analisar os parâmetros calculados e verificar a eficiência geral do código computacional.

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado em sete capítulos, conforme segue:

- Capítulo 1: Trata-se de uma introdução ao tema tratado, além de apresentar o problema o qual a pesquisa pretende solucionar, a hipótese levantada e os objetivos da mesma;
- Capítulo 2: Neste capítulo, apresenta-se a revisão bibliográfica suporte ao trabalho realizado, contemplando o funcionamento dos Métodos Lattice Boltzmann e Elementos Discretos, além de apresentar diferentes modelos de acoplamento hidromecânico;
- Capítulo 3: Apresenta-se a metodologia do estudo com foco na divisão das fases deste trabalho e detalhamento dos cenários de simulação para calibração e validação do código computacional. Além disso, é feito uma análise de possíveis casos de interesse da engenharia geotécnica para ser simulado utilizando a metodologia híbrida lattice Boltzmann-Elementos Discretos;
- Capítulo 4: Apresentam-se os resultados obtidos, sua análise e comparação com os resultados esperados.
- Capítulo 5: As conclusões da pesquisa são apontadas e sugestões para trabalhos futuros são feitas.
- Apêndices: Apresentam-se os cabeçalhos das classes implementadas no código computacional e os scripts de cenários para cada simulação necessários para o desenvolvimento da dissertação.

CAPÍTULO II

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentada a revisão bibliográfica necessária para prover o conhecimento teórico para o desenvolvimento deste trabalho. São apresentadas as bases teórica e matemática dos métodos lattice Boltzmann e elementos discretos e são discutidos alguns modelos de acoplamento hidromecânico essenciais para o desenvolvimento da metodologia híbrida MLB-MED.

Todas as variáveis em **negrito** apresentadas neste capítulo representam vetores.

2.1 MÉTODO LATTICE BOLTZMANN

O Método lattice Boltzmann (MLB) foi originado a partir do método Lattice Gás Autômata Celular (Frisch *et al.*, 1986) e pode ser visto como uma discretização especial da equação de Boltzmann com velocidades discretas. Neste método, o fluido é modelado por uma única função de distribuição de partículas (Guo & Zhao, 2002) e sua evolução é dada pela seguinte equação:

$$f_i(\boldsymbol{x} + \boldsymbol{c_i}\Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = \Omega_i(\boldsymbol{x}, t) \tag{2.1}$$

onde f_i é a função escalar de distribuição de partículas na posição $\boldsymbol{x}$ e tempo t, $\boldsymbol{c_i}$ é o vetor de velocidade discreta e Ω_i é o operador de colisão que controla a taxa de relaxação da função de distribuição das partículas. A Eq. 2.1 representa os processos de colisão e propagação de partículas em cada nó (ponto de cálculo) e passo de tempo. Na colisão, as funções de distribuição que chegam em um nó são redistribuídas e na propagação elas são realocadas para os nós mais próximos (Owen *et al.,* 2011). Um esquema do processo de propagação das partículas é ilustrado na Figura 2.1.

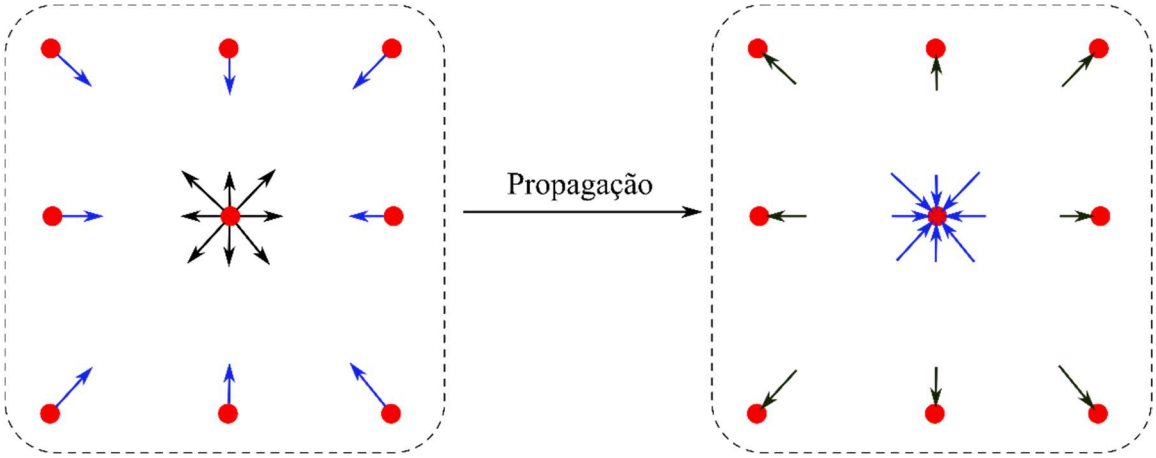

Figura 2.1: Partículas fluindo do nó central para seus vizinhos, das quais as partículas são transmitidas de volta (modificado – Krüger *et al.,* 2017).

As funções de distribuição das partículas são expressões que definem a probabilidade de encontrar um conjunto de partículas no espaço em determinado passo de tempo. A partir dessas funções e do vetor de velocidade discretas é possível definir as variáveis macroscópicas do fluido, densidade (ρ) e o vetor de velocidade ($\boldsymbol{u}$), sendo elas calculadas a partir das seguintes equações:

$$\rho = \sum_{i=0}^{8} f_i \tag{2.2}$$

$$\boldsymbol{u} = \frac{1}{\rho}\sum_{i=0}^{8} \boldsymbol{c_i} f_i \tag{2.3}$$

É interessante notar que a Eq. 2.3 permite obter vetores de velocidade em qualquer direção como uma combinação linear das velocidades discretas, tendo como pesos as frequências de distribuição das partículas em cada direção.

2.1.1 ESTRUTURA DA MALHA

No MLB, o domínio é discretizado em malhas (*lattice*) com *n* direções e definida num espaço com *d* dimensões, comumente identificadas por D*d*Q*n* (Qian *et al.,* 1992). A Figura 2.2 ilustra a discretização do domínio em malhas lattice e o esquema da estrutura da malha D2Q9 comumente utilizada em simulações bidimensionais.

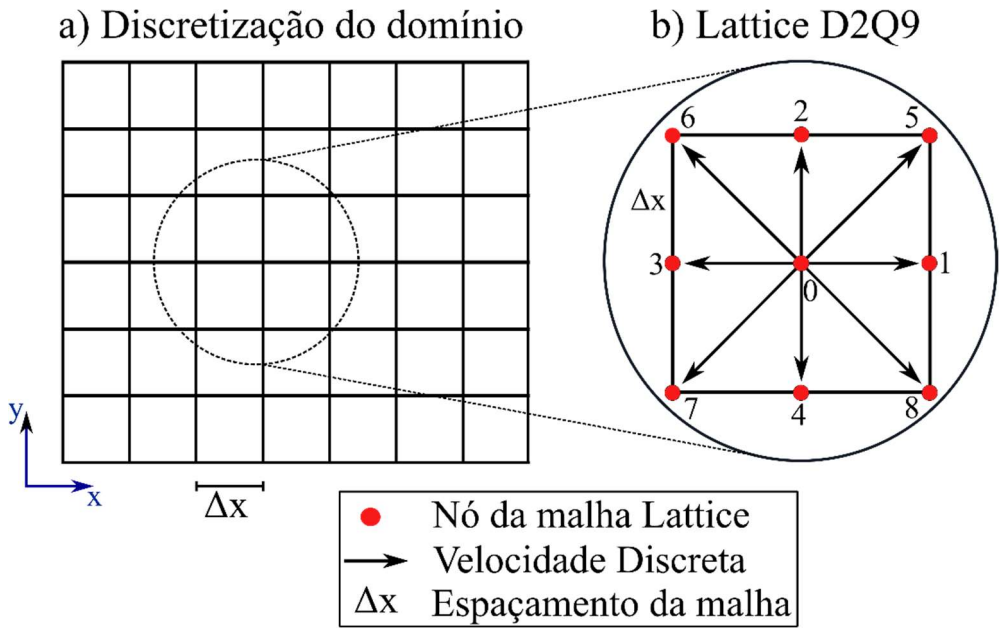

Figura 2.2: a) discretização do domínio em malhas lattice; b) estrutura da malha D2Q9.

Cada arranjo de malhas lattice apresenta algumas constantes como os vetores de velocidade discreta, os pesos referentes aos nós da malha e a velocidade do som. Em fluxos isotérmicos os vetores de velocidade discretas apontam para os nós na vizinhança imediata e apresentam três tipos de pesos: o peso correspondente ao vetor de velocidade zero, pesos para baixas velocidades e pesos para altas velocidades (Ocampo-Gómez, 2013). As velocidades discretas, os pesos e a velocidade do som para uma malha lattice D2Q9 são definidos, respectivamente, por:

$$\boldsymbol{c_i} = \begin{cases} 0 & \text{i} = 0 \\ C\left(\cos\left(\frac{\pi(i-1)}{2}\right), \sin\left(\frac{\pi(i-1)}{2}\right)\right) & \text{i} = 1-4 \\ \sqrt{2}C\left(\cos\left(\frac{\pi(i-5)}{2}+\frac{\pi}{4}\right), \sin\left(\frac{\pi(i-5)}{2}+\frac{\pi}{4}\right)\right) & \text{i} = 5-8 \end{cases} \tag{2.4}$$

$$w_0 = \frac{4.0}{9.0}; w_{1-4} = \frac{1.0}{9.0}; w_{5-8} = \frac{1.0}{36.0} \tag{2.5}$$

$$c_s = \frac{C}{\sqrt{3}} \tag{2.6}$$

onde C é a velocidade lattice definida pela seguinte equação:

$$C = \frac{\Delta x}{\Delta t} \tag{2.7}$$

onde Δx é o espaçamento da malha e Δt é o passo de tempo adotado.

Em fluxos isotérmicos, a velocidade do som é utilizada para determinar a equação de estado do modelo, definida por:

$$p = c_s^2 \rho \tag{2.8}$$

onde p é a pressão e ρ é a densidade do fluido. Além disso, a velocidade do som define efetivamente um limite superior para máxima velocidade de simulação do fluido (Gardner, 2018). Para simular as equações de Navier-Stokes para fluidos incompressíveis utilizando o MLB, é necessário que o número Mach, *Ma*, se limite no máximo a 0.1, para evitar instabilidades na simulação (Abdelhamid & Shamy, 2014). O número Mach pode ser dado pela seguinte expressão:

$$Ma = \frac{v_{max}}{C} \tag{2.9}$$

onde v_{max} é a máxima velocidade do fluido no domínio simulado. Tais limitações são importantes na determinação do passo de tempo e escolha do tamanho da malha lattice, ou seja, discretização do domínio para cada simulação.

2.1.2 OPERADOR DE COLISÃO

O operador de colisão modela choques entre as partículas do fluido, redistribuindo-as entre as diferentes populações (f_i) em cada nó da malha lattice. Embora o operador de colisão não consiga descrever toda a física microscópica subjacente, ele ainda respeita a conservação de massa e momento, além de recuperar, de forma correta, o comportamento macroscópico (Gardner, 2018). Um dos operadores de colisão mais utilizados devido a sua simplicidade e melhor eficiência computacional (Wang *et al.*, 2019) é o modelo Bhatnagar-Gross-Krook (BGK*)* (Bhatnagar *et al*., 1954), descrito por:

$$\Omega_i = -\frac{\Delta t}{\tau}\left[f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)\right] \tag{2.10}$$

onde τ é o tempo de relaxação e f_i^{eq} é a função de distribuição no equilíbrio. Pode-se perceber da Eq. 2.10 que as funções de distribuição são relaxadas para o equilíbrio local na mesma taxa em função do tempo de relaxação, sendo este o parâmetro mais importante em simulações utilizando o MLB (Feng *et al*., 2007).

He & Luo (1997) mostraram que a equação de Navier-Stokes para fluxo de fluidos incompressíveis é recuperada quando a função de distribuição no equilíbrio e a viscosidade cinemática do fluido forem definidas pelas seguintes equações:

$$f_i^{eq} = \omega_i \rho \left[1 + 3\,\frac{\boldsymbol{c_i} \bullet \boldsymbol{u}}{C^2} + \frac{9}{2}\frac{(\boldsymbol{c_i} \bullet \boldsymbol{u})^2}{C^4} - \frac{3}{2}\frac{\boldsymbol{u} \bullet \boldsymbol{u}}{C^2}\right] \tag{2.11}$$

$$\nu = c_s^2 \left(\tau - \frac{1}{2}\right)\frac{\Delta x^2}{\Delta t} \tag{2.12}$$

onde ρ e $\boldsymbol{u}$ são respectivamente a densidade e o vetor de velocidade do fluido, w_i corresponde aos pesos em cada nó da malha (Eq. 2.5) e $\bullet$ simboliza produto escalar. Convém ressaltar da Eq. 2.12 que τ deve ser maior que 0.5 para evitar que a viscosidade cinemática do fluido simulado seja nula. Assim, o tempo de relaxação tem papel fundamental na estabilidade numérica de simulações utilizando o MLB e quanto maior o seu valor mais estável é a simulação (El Shamy & Abdelhamid, 2014).

2.1.3 FORÇAS DE CORPO

A inclusão de forças de corpo no MLB pode ser feita por meio de um termo adicional à Eq. 2.1, resultando em:

$$f_i(\boldsymbol{x} + \boldsymbol{c_i}\Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = \Omega_i(\boldsymbol{x}, t) + S_i(\boldsymbol{x}, t) \tag{2.13}$$

onde S_i é o termo de fonte para acréscimo de forças externas ao modelo. Guo *et al*. (2002) demonstraram que para recuperar corretamente a equação de Navier-Stokes, este termo de força deve ser descrito da seguinte maneira:

$$S_i = \left(1 - \frac{1}{2\tau}\right)\omega_i \left[\frac{\boldsymbol{c_i} - \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{c_i} \bullet \boldsymbol{u})}{c_s^4}\boldsymbol{c_i}\right] \bullet \boldsymbol{F} \tag{2.14}$$

onde $\boldsymbol{F}$ é o vetor de força, no caso de forças de corpo $\boldsymbol{F} = \rho \boldsymbol{g}$, sendo $\boldsymbol{g}$ a aceleração da gravidade e ρ a densidade do fluido. Além disso, os vetores de velocidade no equilíbrio macroscópico do fluido devem ser modificados devido à adição da força, sendo calculados pela seguinte equação:

$$\boldsymbol{u_{macro}} = \boldsymbol{u^{eq}} = \frac{1}{\rho}\sum_{\mathrm{i}=0}^{8} f_i \boldsymbol{c_i} + \frac{\boldsymbol{F}\Delta t}{2\rho} \tag{2.15}$$

2.1.4 CONDIÇÕES INICIAIS

A condição inicial das funções de distribuição em uma simulação MLB depende do tipo de fluxo, estacionário ou transiente, do problema simulado. Em situações onde o fluxo é estacionário, a metodologia mais utilizada é fazer a inicialização das funções coincidir com a função de distribuição no equilíbrio, demonstrado a seguir:

$$f_i(\boldsymbol{x}, t = 0) = f_i^{eq}(\rho_0, \boldsymbol{u_0}) \tag{2.16}$$

onde ρ_0 e $\boldsymbol{u_0}$ são as condições iniciais impostas para densidade e vetor de velocidade do fluido, respectivamente. Em fluxo estacionário, o resultado final não depende da condição inicial e, por isso, esta metodologia se apresenta como uma alternativa viável para determinação da condição inicial das funções de distribuição das partículas.

Em fluxos transientes ou altamente não lineares esta metodologia pode acarretar em grandes erros devido à sensibilidade destes tipos de fluxo às condições iniciais. Nestes casos, uma abordagem mais rigorosa para determinação da condição inicial deve ser adotada. Algumas metodologias iterativas (Skordos, 1993; Caiazzo, 2005; Mei *et al*., 2006) foram propostas, porém como se deseja determinar parâmetros é desejável chegar a um estado estacionário, assim essas metodologias fogem do escopo deste trabalho.

2.1.5 CONDIÇÕES DE CONTORNO

Nos contornos do domínio discretizado, algumas funções de distribuição são desconhecidas sendo necessário impor algumas condições para que estas funções sejam determinadas, assim as condições de contorno desempenham papel fundamental para o desenvolvimento de uma simulação MLB. Geralmente, as condições de contorno mais aplicadas são as condições periódicas e o *bounce-back* ou reflexão. Além destas, Zou & He (1997) propuseram condições para calcular as funções de distribuição a partir de velocidades ou densidades prescritas também conhecidas como condições de Von Neumann e Dirichlet, respectivamente.

As condições de contorno periódicas são as mais simples, sendo o domínio tratado como um sistema fechado com as bordas opostas conectadas entre si, assim, a função de distribuição que sai por uma extremidade entra pelo outro contorno e vice-versa. Quando esta condição é aplicada em uma única direção a topologia do domínio computacional é um cilindro representado na Figura 2.3.

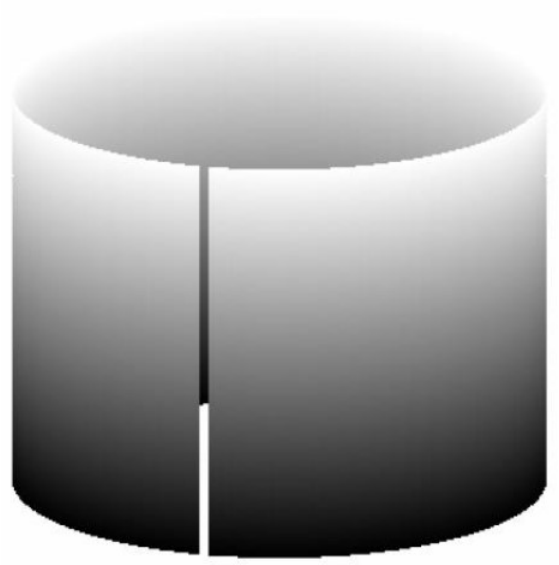

Figura 2.3: Representação da topologia do domínio computacional para uma condição de contorno periódico aplicada em uma única direção (Sukop & Thorne, 2006).

A condição de contorno *bounce-back* consiste numa condição de não escorregamento aplicada nas paredes ou em qualquer outro contorno sólido da simulação. Neste tipo de condição, os nós da malha são divididos em nós fluidos e sólidos, assim, quando uma população parte de um nó fluido para um nó sólido ela é refletida de volta ao nó fluido com a mesma direção de entrada (Ocampo-Gómez, 2013). Devido a sua simplicidade de implementação computacional, esta técnica permite que geometrias complexas possam ser facilmente simuladas. Um esquema da condição de contorno *bounce-back* é ilustrado na Figura 2.4.

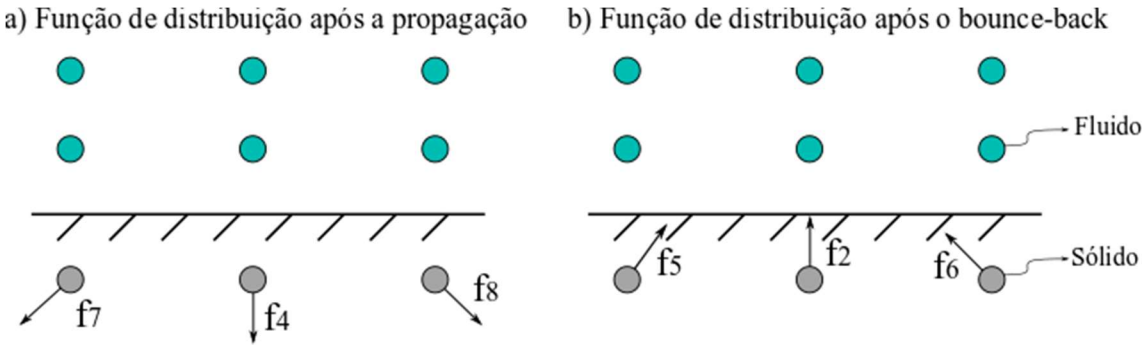

Figura 2.4: Representação da condição de contorno bounce-back.

As condições de contornos propostas por Zou & He (1997) podem ser divididas em dois tipos: condição de Von Neumann ou de velocidade e condição de Dirichlet ou de densidade. Para ambos os casos, velocidades ou densidades prescritas são impostas ao contorno do domínio e as funções de distribuição das partículas são calculadas de tal forma que estas grandezas permaneçam prescritas.

2.1.6 ESTRUTURA DO ALGORITMO

O algoritmo de uma simulação no MLB pode ser dividido em três fases. Na primeira fase é preparado o cenário de simulação, ou seja, o domínio é discretizado em nós fluidos e sólidos, as funções de distribuição das partículas são inicializadas (Eq. 2.16) e são prescritas as condições de velocidades e/ou densidades para aplicação das condições de contorno proposta por Zou & He (1997). Na segunda fase são aplicados os processos de colisão (Eq. 2.10), propagação (Eq. 2.1), condições de contorno e é feito o cálculo das variáveis macroscópicas (Eq. 2.2 e 2.3). A ordem de aplicação do processo colisão-propagação pode ser feita de forma inversa. Por fim, é feito o pós-processamento e visualização da simulação. O resumo da estrutura de um algoritmo para uma simulação utilizando o MLB é ilustrado na Figura 2.5.

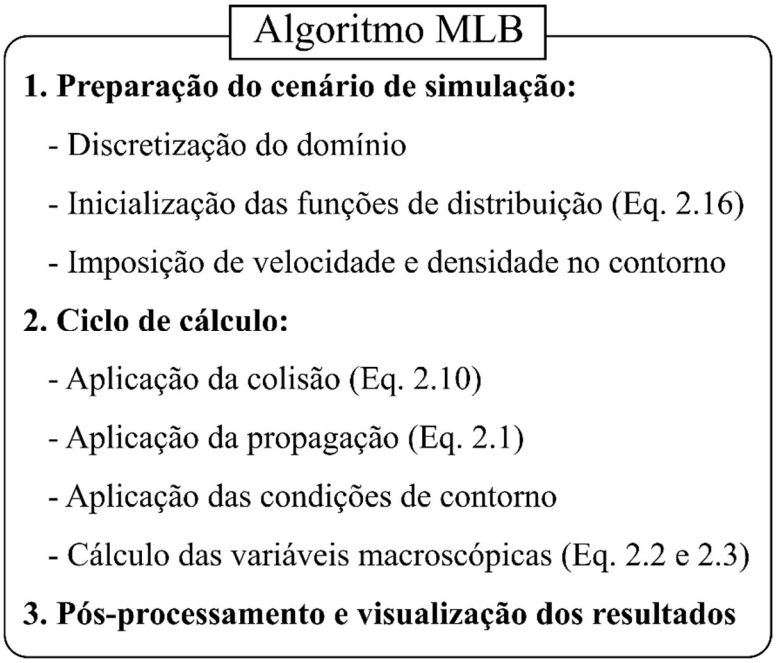

Figura 2.5: Estrutura do algoritmo em uma simulação utilizando o MLB.

2.2 MÉTODO DOS ELEMENTOS DISCRETOS

O método dos elementos discretos (MED) trata a interação entre partículas de acordo com um conjunto de equações e parâmetros físicos para simular os contatos que ocorrem em meios porosos (Brumby *et al.,* 2015). Neste método, as forças de contato e os deslocamentos de um conjunto de elementos discretos são determinados a partir de uma série de cálculos para mapear o movimento de cada partícula (Cundall & Strack, 1979).

O MED é um procedimento adequado para o estudo da dinâmica de meios particulados, já que considera explicitamente sua natureza discreta. Além disso, ele permite avaliar o comportamento físico e mecânico de materiais granulares, mediante o entendimento das propriedades mecânicas microscópicas das partículas e da interação entre as mesmas (Zuluaga, 2016).

Segundo Zuluaga (2016) as principais hipóteses usadas na formulação do MED são:

1. As partículas são elementos rígidos, que possuem inércia finita, descritas analiticamente;
2. As partículas podem se mover livremente uma da outra, podendo girar e deslocar;
3. O contato acontece unicamente entre duas partículas, sobre uma área infinitesimal;
4. As partículas podem se sobrepor levemente nos contatos, sendo esta interpenetração uma analogia com as deformações superficiais da partícula (esmagamento);
5. As forças de compressão entre partículas podem ser calculadas da interpenetração e as de tração da separação das partículas, no caso de partículas cimentadas.

Uma simulação utilizando o MED pode ser dividida em 4 estágios. No primeiro estágio é determinado para cada partícula uma lista de elementos que possam interagir com ela. No segundo estágio, a partir da colisão entre as partículas, é calculada a força nos contatos por meio de uma lei de interação. No terceiro estágio é calculada a aceleração de cada partícula devido às forças atuantes, utilizando a segunda lei de Newton. Por fim, a velocidade e a posição das partículas são atualizadas por meio de um método de integração no tempo.

2.2.1 FORMULAÇÃO GERAL

As forças de contato desenvolvidas entre os corpos durante a colisão são obtidas por meio de uma lei que relaciona força e o deslocamento relativo do corpo em contato com outra partícula ou uma parede. A Figura 2.6 ilustra o desenvolvimento do contato de uma partícula com outra partícula e com uma parede.

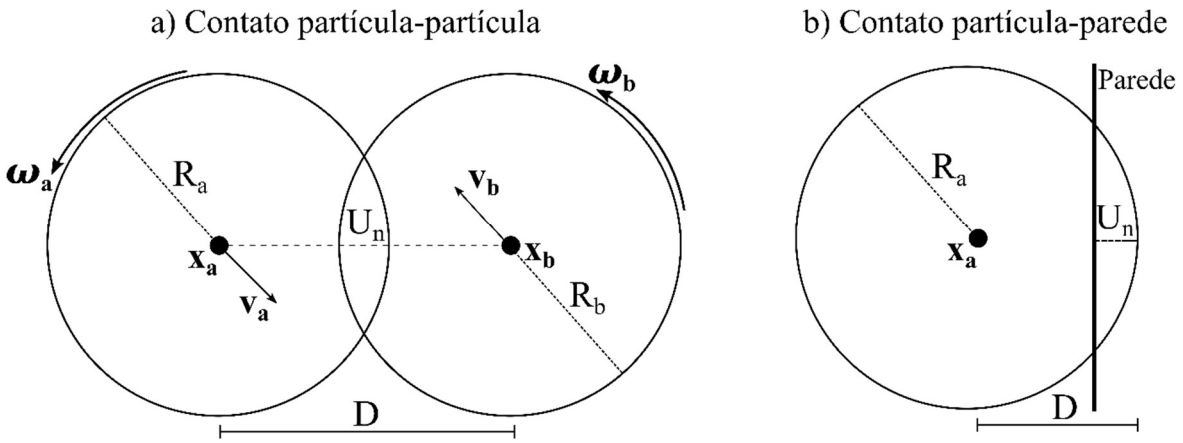

Figura 2.6: Desenvolvimento de contato entre partículas (a) e paredes (b).

As equações dos incrementos de força de contato normal e cisalhante, para uma lei de interação linear, podem ser definidas, respectivamente, por:

$$\Delta F_n = k_n \Delta U_n \tag{2.17}$$

$$\Delta F_s = k_s \Delta U_s \tag{2.18}$$

onde k_n e k_s são as rigidezes normal e tangencial; U_n e U_s são os deslocamentos normais e tangenciais. As forças de contato finais são então determinadas somando os incrementos às forças do passo de tempo anterior, conforme mostrado a seguir:

$$(\mathrm{F_n})_N = (F_n)_{N-1} + \Delta F_n \tag{2.19}$$

$$(\mathrm{F_s})_N = (F_s)_{N-1} + \Delta F_s \tag{2.20}$$

Além da lei de interação, o modelo deve obedecer a uma lei de atrito. Em simulações discretas nas quais o objeto de estudo é o solo, a lei de atrito de Mohr-Coulomb é geralmente utilizada, sendo expressa da seguinte maneira:

$$(F_s)_{max} = F_n tan\phi_u + c \tag{2.21}$$

onde F_n é a força normal aplicada no corpo, φ_u e c correspondem ao menor ângulo de atrito e a menor coesão do contato entre os corpos, e $F_{s,max}$ é a máxima força cisalhante do contato entre duas partículas.

No MED, a força de cisalhamento calculada pela Eq. 2.20 é limitada ao valor máximo definido pela lei de atrito de Mohr-Coulomb (Eq. 2.21). Caso $|F_s| > |F_{s,max}|$ o escorregamento pode ocorrer, fazendo:

$$(F_s)_N = (F_s)_N \frac{(F_s)_{max}}{|(F_s)_N|} \tag{2.22}$$

O movimento de cada partícula é determinado pela segunda Lei de Newton em função das forças atuantes no corpo. As equações que governam o movimento translacional e rotacional de cada partícula são expressas, respectivamente, por:

$$m\frac{d\boldsymbol{v}}{dt} = \boldsymbol{F_c} + \boldsymbol{F_f} + \boldsymbol{G} + \boldsymbol{F_v} \tag{2.23}$$

$$I\frac{d\boldsymbol{\omega}}{dt} = \boldsymbol{T_c} + \boldsymbol{T_f} + \boldsymbol{T_v} \tag{2.24}$$

onde $\boldsymbol{v}$ e $\boldsymbol{\omega}$ são os vetores de velocidade translacional e rotacional da partícula, respectivamente; m é a massa e I é o momento de inércia; $\boldsymbol{G}$ é a força da gravidade; $\boldsymbol{F_c}$ e $\boldsymbol{T_c}$ são a força e o torque resultante do contato entre as partículas; $\boldsymbol{F_f}$ e $\boldsymbol{T_f}$ correspondem à força e ao torque devido a um fluido atuantes numa partícula; e $\boldsymbol{F_v}$ e $\boldsymbol{T_v}$ correspondem à força e ao torque de amortecimento local não-viscoso aplicados nas partículas para evitar que o sistema oscile indefinidamente. As equações de força e torque não-viscosos são definidas como:

$$\boldsymbol{F_v} = -\alpha|\boldsymbol{F}|sign(\boldsymbol{v}) \tag{2.25}$$

$$\boldsymbol{T_v} = -\alpha|\boldsymbol{T}|sign(\boldsymbol{\omega}) \tag{2.26}$$

onde α é o coeficiente de amortecimento local, $|\boldsymbol{F}|$ e $|\boldsymbol{T}|$ são a força e o torque atuantes na partícula e *sign(x)* é uma função definida por:

$$sign(x) = \begin{cases} +1, & x > 0 \\ -1, & x < 0 \end{cases} \tag{2.27}$$

2.2.2 MÉTODO DE INTEGRAÇÃO EXPLÍCITO

As equações de movimento das partículas (Eq. 2.23 e 2.24) são geralmente solucionadas utilizando um esquema de integração explícito baseado em um esquema de diferentes finitas centrado. O método de integração explícito *LeapFrog* foi utilizado nesta pesquisa por ser um método que apresenta precisão de segunda ordem, fazer conservação exata de momento e apresentar estabilidade global. Neste método, a velocidade do corpo é conhecida a meio passo de tempo defasado da sua posição, e seus cálculos intercalam-se ao longo da simulação (Rasmussen et al., 2018). A Figura 2.7 ilustra o esquema do processo de cálculo do método *LeapFrog*.

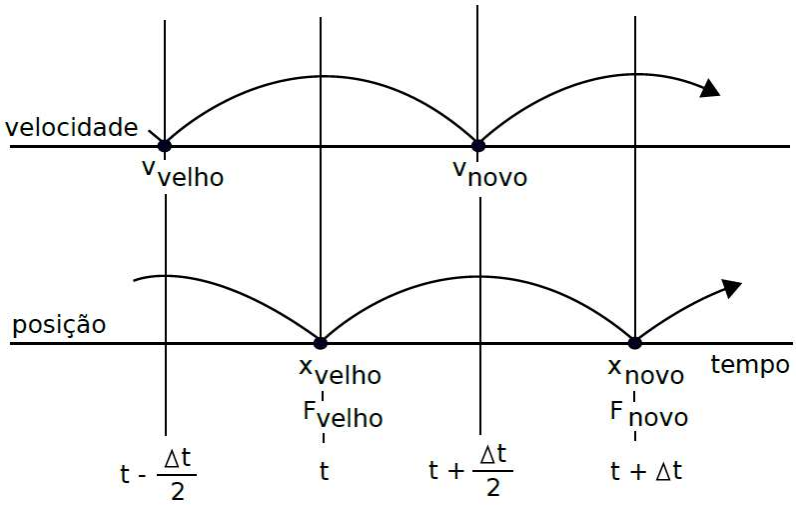

Figura 2.7: Esquema do processo de cálculo do método *LeapFrog* (Rasmussen, 2018).

Neste método, o vetor de velocidade $\boldsymbol{v}$ e o vetor de velocidade angular $\boldsymbol{\omega}$ da partícula são atualizados a cada instante de tempo t por meio das seguintes equações:

$$\boldsymbol{v}^{(t+\Delta t/2)} = \boldsymbol{v}^{(t-\Delta t/2)} + \frac{\boldsymbol{F}^{(t)}}{m}\Delta t \tag{2.28}$$

$$\boldsymbol{\omega}^{(t+\Delta t/2)} = \boldsymbol{\omega}^{(t-\Delta t/2)} + \frac{\boldsymbol{T}^{(t)}}{I}\Delta t \tag{2.29}$$

onde $\boldsymbol{F}$ e $\boldsymbol{T}$ são as forças e os torques atuantes na partícula. Após o cálculo dos vetores de velocidades, os vetores de posição e rotação são atualizados com as seguintes equações:

$$\boldsymbol{x}^{(t+\Delta t)} = \boldsymbol{x}^{(t)} + \boldsymbol{v}^{(t+\Delta t/2)}\Delta t \tag{2.30}$$

$$\boldsymbol{\theta}^{(t+\Delta t)} = \boldsymbol{\theta}^{(t)} + \boldsymbol{\omega}^{(t+\Delta t/2)}\Delta t \tag{2.31}$$

Uma interação de meio passo de tempo é feita inicialmente no método *LeapFrog* para determinar os vetores de velocidade translacional e rotacional por meio das seguintes equações:

$$\boldsymbol{v}^{(\Delta t/2)} = \boldsymbol{v}^{(0)} + \frac{1}{2}\frac{\mathbf{F}^{(0)}}{\mathrm{m}}\Delta t \tag{2.32}$$

$$\boldsymbol{\omega}^{(\Delta t/2)} = \boldsymbol{\omega}^{(0)} + \frac{1}{2}\frac{\mathbf{T}^{(0)}}{\mathrm{I}}\Delta t \tag{2.33}$$

No desenvolvimento de uma simulação utilizando o MED é necessário escolher um passo de tempo de tal forma que perturbações não possam ser propagadas para além das partículas vizinhas, causando instabilidades na simulação. Para que isto não ocorra, é

recomendado que o passo de tempo escolhido seja menor que o passo de tempo crítico estimado por:

$$\Delta t_{crit} = 2\lambda\sqrt{\frac{m}{k}} \tag{2.34}$$

onde m é a menor massa e k é a rigidez do sistema de partículas e λ é um fator de segurança variando de 0 a 1, sendo aconselhado utilizar valores próximos a 0.1 para garantir estabilidade e acurácia da solução (Feng *et al*., 2007).

2.2.3 ESTRUTURA DO ALGORITMO

Uma simulação utilizando o MED pode ser dividida em 3 fases. Na primeira fase é feita a preparação do cenário de simulação, sendo definidos o tamanho do domínio no qual os corpos serão inseridos, é feita a adição dos corpos e, por fim, o passo de tempo é determinado. Na segunda fase é feito o ciclo de cálculo do método, assim, as forças atuantes no corpo são resetadas, é verificado se o corpo está em colisão com algum outro corpo ou parede, são aplicadas as leis de interação e atrito, as forças atuantes para a determinação da aceleração do corpo são calculadas, por fim, a velocidade e a posição dos corpos são atualizados. Na terceira fase são feitos o pós-processamento e a visualização dos resultados. A estrutura do algoritmo de uma simulação utilizando o MED é descrita na Figura 2.8.

Algoritmo MED

1. Preparação do cenário de simulação:
- Delimitação do domínio (*Bounding box*)
- Adição dos corpos
- Definição do passo de tempo (Eq. 2.34)

2. Ciclo de cálculo:
- Resete das forças e momentos atuantes
- Detecção de colisão
- Aplicação da lei de atrito (Eq. 2.22)
- Aplicação da lei constitutiva (Eq. 2.17 e 2.18)
- Aplicação das forças atuantes no corpo
- Determinação da aceleração (Eq. 2.23 e 2.24)
- Atualização da velocidade e posição (Eq. 2.28, 2.29, 2.30 e 2.31)

3. Pós-processamento e visualização dos resultados

Figura 2.8: Algoritmo detalhado de uma simulação utilizando o MED.

2.3 MÉTODOS DE ACOPLAMENTO HIDROMECÂNICO

A interação entre o fluxo de fluidos com partículas sólidas em suspensão é parte integrante para acoplagem dos métodos lattice Boltzmann e elementos discretos. Uma série de técnicas que variam em complexidade, precisão e custo computacional foram desenvolvidas para lidar com essa interação (Owen *et al*., 2011), podendo ser citados os métodos *Linked Bounce-Back* (LBB) ou Reflexão Conectada (Ladd, 1994) e o *Immersed Moving Boundary* (IMB) ou Método das Fronteiras Móveis (Noble & Torczynski, 1998).

2.3.1 LINKED BOUNCE-BACK

O método *linked bounce-back* foi proposto por Ladd (1994) como uma forma de melhorar a técnica de *bounce-back* para a modelagem da interação fluido-sólido por meio de condições de contornos móveis. Como visto anteriormente, a técnica *bounce-back* é utilizada como condição de contorno, porém este método impõe esta técnica na ligação entre os contornos de nós fluidos e contornos de nós sólidos, assumindo que a interface sólido-fluido existe no meio de cada ligação (Owen et al., 2011). A Figura 2.9 ilustra um esquema do método *linked bounce-back.*

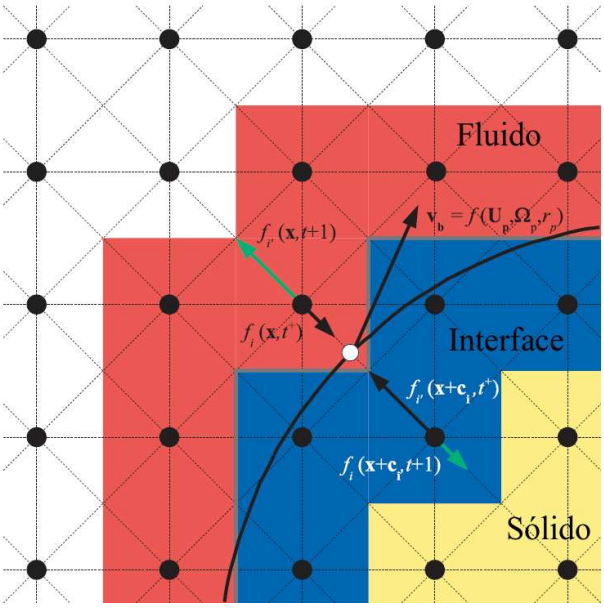

Figura 2.9: Esquema do método *linked bounce-back* (modificado – Owen et al., 2011).

A velocidade do contorno ($\boldsymbol{v_b}$) próximo a interface é calculada a partir da velocidade translacional ($\boldsymbol{v_t}$), velocidade rotacional ($\boldsymbol{\omega}$) e do centroide ($\boldsymbol{x_c}$) das partículas sólidas por meio da seguinte expressão:

$$\boldsymbol{v_b} = \boldsymbol{v_t} + \boldsymbol{\omega} x \left[\left(\boldsymbol{x} + \frac{1}{2}\boldsymbol{c_i}\Delta t\right) - \boldsymbol{x_c}\right] \tag{2.35}$$

A partir da velocidade do contorno pode-se aplicar a condição de *bounce-back* para os nós fluidos e sólidos, respectivamente, como a seguir:

$$f_{i'}(\boldsymbol{x}, t + \Delta t) = f_i(\boldsymbol{x}, t^+) - 2w_i\rho\boldsymbol{v_b} \bullet \boldsymbol{c_i} \tag{2.36}$$

$$f_i(\boldsymbol{x} + \boldsymbol{c_i}\Delta t, t + \Delta t) = f_{i'}(\boldsymbol{x} + \boldsymbol{c_i}, t^+) + 2w_i\rho\boldsymbol{v_b} \bullet \boldsymbol{c_i} \tag{2.37}$$

onde t^+ corresponde ao tempo após a aplicação do processo de colisão das partículas de fluido e $f_{i'}$ é a função de distribuição de partículas na direção oposta a f_i.

A definição das funções de distribuição é essencial para determinar a força hidrodinâmica exercida sobre a partícula sólida. Neste método, a força hidrodinâmica é definida pela seguinte expressão:

$$\boldsymbol{F_{i'}}\left(\boldsymbol{x} + \frac{1}{2}\boldsymbol{c_i}\Delta t, t + \frac{1}{2}\Delta t\right) = 2[f_i(\boldsymbol{x}, t^+) - f_{i'}(\boldsymbol{x} + \boldsymbol{c_i}\Delta t, t^+) - 2w_i\rho\boldsymbol{v_b} \bullet \boldsymbol{c_i}]\boldsymbol{c_i} \tag{2.38}$$

A Eq. 2.38 corresponde à força hidrodinâmica devido a uma única célula lattice do fluido. Assim, para determinar a força total deve-se somar a contribuição de todas as outras células que estão em contato com a partícula sólida.

Este é um método fácil de ser implementado computacionalmente, porém apresenta algumas desvantagens significativas como a disparidade entre a forma física e simulada da partícula sólida e a ocorrência de flutuações na força hidrodinâmica induzida (Owen *et al*., 2011) devido à necessidade da constante atualização do tipo de nó, fluido ou sólido, em função da passagem da partícula sólida pelas células lattice.

2.3.2 MÉTODO DAS FRONTEIRAS MÓVEIS

O método das fronteiras móveis (MFM) foi desenvolvido por Noble & Torczynski (1998). Este método apresenta grandes vantagens como a conservação da localidade das operações de colisão e adição de um operador de propagação linear simples (Owen *et al*., 2011), o que o torna ideal para ser utilizando junto com o MLB. No MFM, a equação do MLB é modificada para incluir um novo termo de colisão que leva em consideração a interação do fluido com os obstáculos sólidos. As modificações feitas na equação do MLB são mostradas a seguir:

$$f_i(\boldsymbol{x} + c_i\Delta t, t + \Delta t) = f_i - (1 - B_n)\left[\frac{\Delta t}{\tau}\left(f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)\right)\right] + B_n\Omega_i^S \tag{2.39}$$

onde B_n é uma função de peso para o operador de colisão adicional variando de 0 a 1, correspondendo os valores extremos às condições de fluido ou sólido puro, respectivamente. Esta função pode ser calculada como:

$$B_n(\varepsilon, \tau) = \frac{\varepsilon(\tau - 1/2)}{(1 - \varepsilon) + (\tau - 1/2)} \tag{2.40}$$

onde ε corresponde à fração do volume ocupado pelas partículas sólidas que interceptam os nós da malha lattice. A correta determinação do volume ocupado pela partícula sólida não é uma tarefa trivial. Algumas técnicas variando em precisão e desempenho computacional foram propostas, podendo ser citados os métodos Monte Carlo, *Sphero-Polyhedra* (Galindo-Torres, 2013), Aproximação Linear (Jones & Williams, 2017) e o método proposto por Wang *et al*. (2019). Dentro os métodos citados, o método *Sphero-Polyhedra* se apresenta com uma técnica eficaz, rápida e simples de ser implementada computacionalmente e foi o método escolhido para a definição da fração sólida nesta dissertação.

O novo operador de colisão (Ω_i^S) é baseado no conceito de *bounce-back* da parte não equilibrada da função de distribuição e pode ser calculado como:

$$\Omega_i^S = f_{i'}(\boldsymbol{x}, t) - f_i(\boldsymbol{x}, t) + f_i^{eq}(\rho, \boldsymbol{v}) - f_i^{eq}(\rho, \boldsymbol{u}) \tag{2.41}$$

onde $\boldsymbol{v}$ é a velocidade da partícula sólida. O cálculo de força e momento da interação do fluido com as partículas sólidas é dado, respectivamente, pelas seguintes equações:

$$\boldsymbol{F_f} = \frac{(\Delta x)^D}{\Delta t}\sum_n B_n \sum_i \Omega_i^S \boldsymbol{c_i} \tag{2.42}$$

$$\boldsymbol{M_f} = \frac{(\Delta x)^D}{\Delta t}\sum_n (\boldsymbol{x_i} - \boldsymbol{x_c}) \text{ x } \left(B_n \sum_i \Omega_i^S \boldsymbol{c_i}\right) \tag{2.43}$$

onde *D* representa o domínio da análise simulada e x representa o produto vetorial.

2.3.3 COMPATIBILIDADE DO PASSO DE TEMPO

O MLB e o MED são esquemas explícitos que utilizam diferentes passos de tempo na execução de suas simulações. O acoplamento desses métodos requer a escolha de um passo de

tempo único que obedeça às limitações de propagação de perturbação impostas pelo MED para evitar que a simulação apresente instabilidades.

Owen *et al*. (2011) propuseram dois cenários para definição eficaz do passo de tempo de acoplamento. Em simulações onde $\Delta t_{MED} > \Delta t_{MLB}$, o passo de tempo do acoplamento deve ser igual ao Δt_{MLB}. Esta abordagem reduz de maneira conservadora o passo de tempo do MED, garantindo que a simulação possa ocorrer sem apresentar instabilidades. Por outro lado, em simulações onde $\Delta t_{MLB} > \Delta t_{MED}$, deve se utilizar uma abordagem de subciclos, ou seja, durante um ciclo do MLB são realizado alguns ciclos do MED. A definição do número de subciclos é feita relacionando os passos de tempo do MLB e da fração do passo de tempo crítico do MED, conforme mostrado a seguir:

$$n_{sub} = \frac{\Delta t_{MLB}}{\Delta t_{MED}} + 1 \tag{2.44}$$

onde n_{sub} é o número de subciclos da simulação MED. Além da definição do número de subciclos, o passo de tempo da simulação MED é reduzido para garantir ainda mais estabilidade no acoplamento dos modelos, assim, o novo passo de tempo é definido por:

$$\Delta t_{MED} = \frac{\Delta t_{MLB}}{n_{sub}} \tag{2.45}$$

É importante notar que durante o subciclo do MED o mapeamento das partículas, a força e o torque hidrodinâmicos não são atualizados. Isso significa que a força e o torque hidrodinâmico aplicados aos elementos discretos devido à interação com o fluido são constantes sobre o número de subciclos, e que a interface fluido-sólido não se move. Os nós do contorno e os valores de carga hidrodinâmica do primeiro subciclo são utilizados para todos os subciclos. Por estas razões, é necessário um limite prático no número de subciclos, para garantir que o limite do elemento discreto seja remapeado antes de cruzar outras células e que todas as variações das cargas sejam capturadas (Owen *et al*., 2011).

Em simulações práticas onde se conhece a viscosidade cinemática do fluido é mais conveniente fazer adequações no espaçamento da malha lattice e no tempo de relaxação de tal forma que o passo de tempo do MLB seja o menor.

2.3.4 ESTRUTURA DO ALGORITMO

A estrutura do algoritmo de uma simulação utilizando qualquer um dos métodos de acoplamento pode ser dividido em três fases. Na primeira fase é delimitado o domínio, os passos de tempo são determinados conforme Item 2.3.3, os parâmetros do fluido e das partículas são definidos e é feita a inicialização das células lattice e a adição dos corpos. Na segunda fase a força e o torque hidrodinâmico são determinados por meio do método de acoplamento utilizado e é aplicado os ciclos de cálculo do MED e MLB conforme os Itens 2.2.3 e 2.1.6, respectivamente. Na terceira fase são feitos o pós-processamento e a visualização dos resultados. A Figura 2.10 ilustra o algoritmo detalhado de uma simulação utilizando os métodos lattice Boltzmann e elementos discretos acoplados por meio do método das fronteiras móveis.

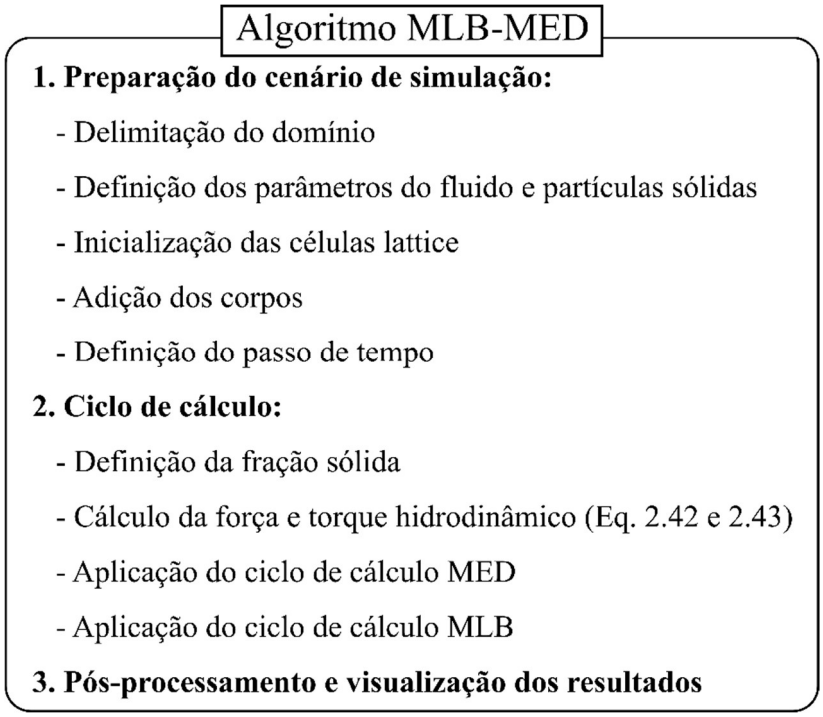

Figura 2.10: Estrutura do algoritmo de uma simulação utilizando o MLB e MED acoplados por meio do MFM.

2.3.5 APLICAÇÕES DO ACOPLAMENTO MLB-MED

A acoplagem dos métodos lattice Boltzmann e elementos discretos tem começado a ganhar mais relevância no estudo da interação entre partículas sólidas e fluidos devido à facilidade de implementação e à fácil paralelização destes modelos, possibilitando a simulação de problemas complexos aliados a um baixo custo computacional. Desta forma, diversas

pesquisas foram e estão sendo desenvolvidas para avaliar a eficácia desta acoplagem e entender melhor as nuances da interação fluido-sólido.

O desenvolvimento do método das fronteiras móveis desenvolvido por Noble & Torczynski (1998) possibilitou um grande avanço na simulação de fluxo de fluidos em meios particulados. A partir deste modelo, diversos trabalhos foram desenvolvidos para estudar problemas de interesse da engenharia geotécnica (Galindo-Torres, 2013; Zhang *et al*., 2017).

Han & Cundall (2013) fizeram simulações utilizando este método para validar a equação de Ergun para um conjunto de partículas fixas, verificar o fluxo ascendente em uma coluna de solo e entender o fenômeno de bombeamento de areia, problema de interesse da engenharia de petróleo. Mansouri *et al*. (2009), El Shamy & Abdelhamid (2014) e Youssoufi *et al*. (2016) estudaram o fenômeno de liquefação e os fatores que acarretam este tipo de problema de relevada importância na estabilidade e funcionamento de barragens de terra.

Habte & Wu (2017) & Rettinger e Rüde (2018) realizaram simulações para entender o comportamento de partículas em suspensão no fluido e verificar o processo de sedimentação de partículas imersas em um fluido. Por fim, pode-se citar os trabalhos de El Shamy & Abdelhamid (2014) e Harshani *et al*. (2015) que fizeram simulações de processos erosivos para avaliar quais as variáveis críticas que influenciam diretamente no desenvolvimento deste tipo de transporte de partículas sólidas.

Todas essas pesquisas demonstram que o acoplamento dos métodos lattice Boltzmann e elementos discretos é uma ferramenta de alto potencial para a simulação de processos que envolvam a interação entre partículas sólidas e fluido, possibilitando assim uma maior compreensão desses fenômenos.

CAPÍTULO III

3 METODOLOGIA

A metodologia usada para o desenvolvimento da dissertação pode ser dividida em duas fases. Na primeira fase foi implementado um código computacional dos métodos lattice Boltzmann e elementos discretos, incluindo ainda a validação do código por meio de problemas de referência (*benchmarks*). Na segunda fase serão feitas simulações de problemas de fluxo de fluidos em meios porosos para estudar a interação fluido-sólido e sólido-sólido. O algoritmo descrito no fluxograma na Figura 3.1 descreve a metodologia que será desenvolvida neste estudo.

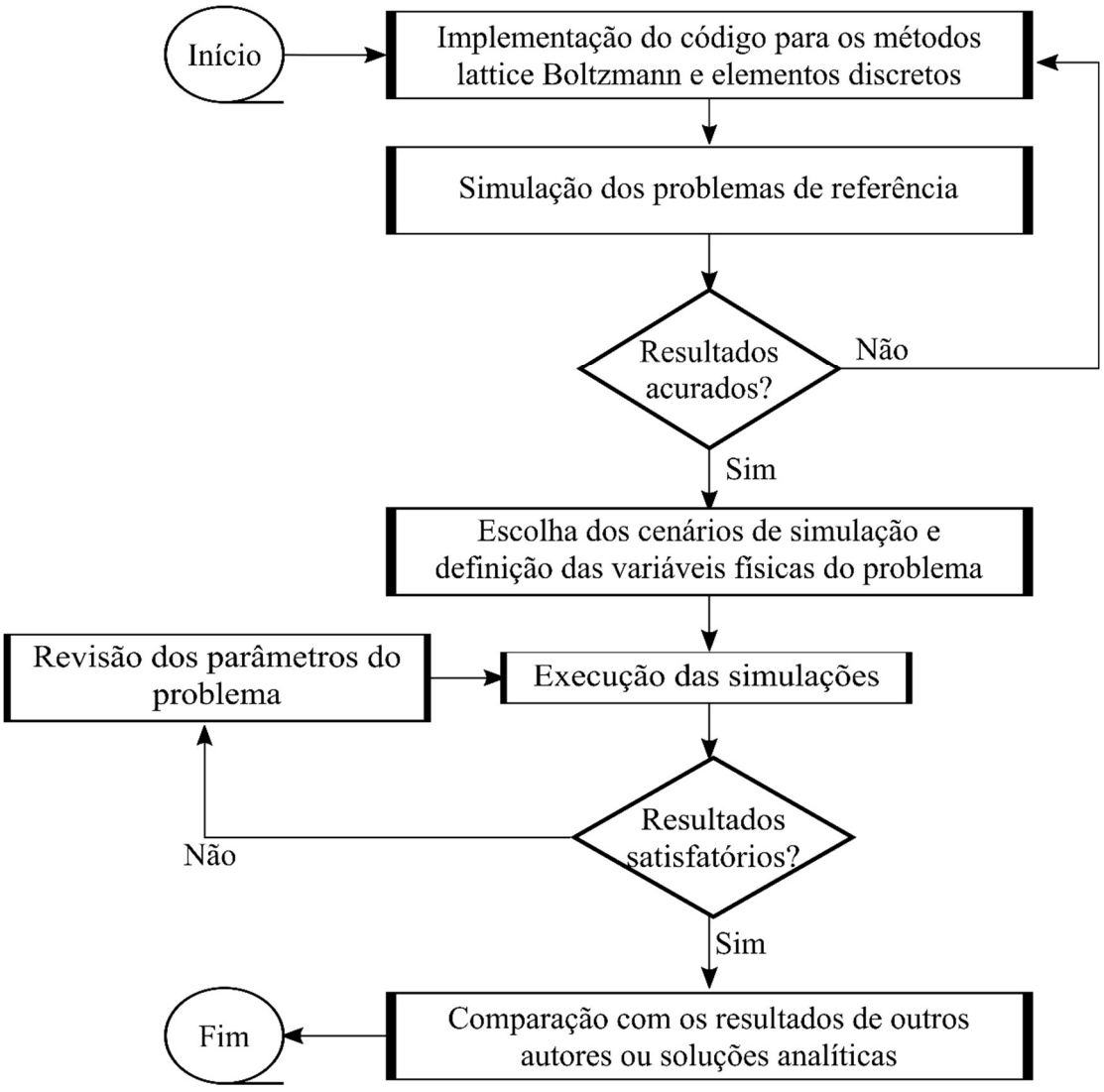

Figura 3.1: Metodologia adotada no desenvolvimento da pesquisa.

3.1 NOVO CÓDIGO COMPUTACIONAL: HYBRID2D

Almejando alcançar os objetivos propostos para esta pesquisa, foi desenvolvido um novo programa para análises bidimensionais de fluxo de fluidos em meios porosos baseado nos métodos lattice Boltzmann e elementos discretos. O programa recebeu o nome de Hybrid2D, estando na versão 0.0.5 até o momento em que esta dissertação foi escrita.

O Hybrid2D foi desenvolvido utilizando a IDE (*Integrated Development Environment*) do *Visual Studio,* e compilado pelo GNU *Compiler Collection* (GCC) em um ambiente *Windows*. A linguagem de programação utilizada para o desenvolvimento do código do programa foi o C++, devido a sua maior velocidade de processamento de dados, quando comparada com outras linguagens (Zapalowski, 2011) e por ser uma linguagem orientada a objeto, o que possibilita a maior reutilização de códigos e uma melhor estruturação do mesmo. O pós-processamento e visualização dos resultados foram feitos utilizando o programa Paraview que faz uso de arquivos com formatação VTK (*visualization toolkit*). Os códigos-fontes de todos os *Headers* ou cabeçalhos do programa estão listados no Apêndice A. O computador utilizado para gerar as simulações possui as características listadas na Tabela 3.1.

Tabela 3.1: Configuração do computador utilizado para geração das simulações.

Processador	Intel® Core™ i5-7300HQ CPU @ 2.50GHz
Memória RAM	8 GB
Tipo de sistema	64bits

No Hybrid2D foram implementados os métodos lattice Boltzmann e elementos discretos, com acoplamento feito por meio do método das fronteiras móveis. No MLB foi o utilizado o operador de colisão BGK com um único tempo de relaxação e discretização do domínio feita por meio de malhas lattice tipo D2Q9, por ser mais indicada para simulações bidimensionais de fluxo de fluidos (Mohammad, 2011). No MED, as interações entre os corpos foram limitadas para partículas circulares (discos em simulações bidimensionais) de diversos tamanhos para simular a matriz porosa. Todas essas características fazem do Hybrid2D um programa com um grande potencial para análises bidimensionais da interação entre partículas sólidas e fluidos.

3.1.1 ORGANIZAÇÃO DO PROGRAMA

O código do programa Hybrid2D foi organizado em classes e a sua estrutura é descrita na Figura 3.2.

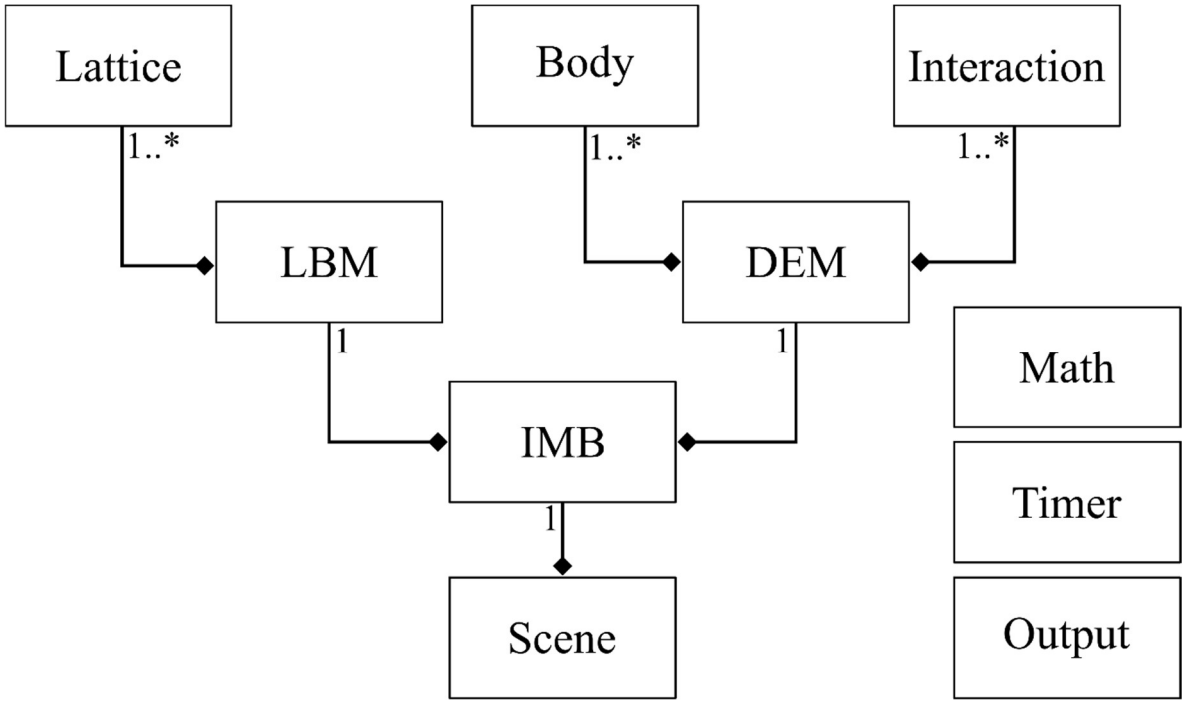

Figura 3.2: Estrutura do código do programa Hybrid2D.

Cada classe foi desenvolvida com o seguinte propósito:

- Lattice: responsável por guardar as informações de posição, condições de contorno, função de distribuição e os valores das variáveis macroscópicas do fluido de cada célula lattice utilizada para discretizar o domínio;
- LBM: responsável pelo armazenamento do contêiner de ponteiros para objetos do tipo lattice e apresenta todos os mecanismos de máquina necessários para o desenvolvimento de uma simulação utilizando o MLB, tais como, colisão, propagação e condições de contorno;
- *Body*: responsável por guardar as informações de massa, momento de inércia, geometria, posição e velocidade de cada partícula sólida;
- *Interaction*: responsável por guardar informações do contato entre dois corpos e aplicação das leis de interação e atrito durante as colisões;

- DEM: responsável pelo armazenamento dos contêineres de ponteiros para objetos do tipo *body* e *interaction* e apresenta todos os mecanismos de máquina necessários para o desenvolvimento de uma simulação utilizando o MED, tais como, verificação de colisão, cálculo das forças atuantes e atualização da velocidade e posição dos corpos;
- IMB: apresenta todos os mecanismos de máquina necessários para o acoplamento do MLB e MED, tais como, determinação da fração sólida do contato entre partícula sólida e a célula lattice e o cálculo da força e torque hidrodinâmico;
- *Math*: classe auxiliar para geração de templates de vetores bidimensionais e funções matemáticas;
- *Timer*: classe auxiliar responsável pela determinação do tempo de uma simulação utilizando o Hybrid2D;
- *Output*: classe auxiliar utilizada para obter os arquivos com extensão .vtk e .csv necessários para o pós-processamento e visualização das simulações;
- *Scene*: responsável pela definição de informações do domínio, do fluido e das partículas e pela geração dos cenários de simulação.

3.2 SIMULAÇÃO DOS PROBLEMAS DE REFERÊNCIA

A validação dos métodos implementados e a calibração de certos parâmetros do fluido e das partículas sólidas no Hybrid2D foi feita de forma separada por meio da análise de problemas com soluções acuradas ou amplamente estudadas. Todos os códigos dos cenários de referência estão apresentados no apêndice B.

Os cenários escolhidos para a calibração e validação do MLB foram o desenvolvimento de fluxo Poiseuille em um cano e fluxo de fluidos ao redor de um cilindro circular para diversos valores de número de Reynolds. No MED os cenários foram escolhidos para avaliar a interação de um corpo com o meio (paredes) e com outros corpos. Desta forma, os cenários escolhidos foram a análise de uma partícula em queda livre e a verificação de colisões elásticas e inelásticas entre partículas. Para todos esses casos foi feita uma variação do amortecimento local. Além disso, os resultados obtidos com o Hybrid2D foram comparados com resultados determinados de forma analítica.

3.2.1 FLUXO POISEUILLE

O fluxo Poiseuille é um tipo de fluxo simples que ocorre no interior de tubulações ou no espaço entre duas placas paralelas. O perfil de velocidades encontrado para este tipo de fluxo é parabólico com velocidade nula nas paredes (condição de contorno não deslizante) e velocidade máxima no ponto médio. O perfil de velocidades pode ser descrito analiticamente pela seguinte equação:

$$u(y) = \frac{G}{2\mu}(a^2 - y^2) \tag{3.1}$$

Onde *u* é a componente de velocidade horizontal do fluido, *y* é a coordenada vertical medida a partir do centro do canal, *a* corresponde à metade da espessura do canal, μ é a viscosidade dinâmica do fluido e *G* é um gradiente de pressão linear ou gravitacional para dutos horizontais e verticais, respectivamente. A Figura 3.3 ilustra a geometria do problema, as condições de contorno impostas e o perfil teórico de velocidades desenvolvida neste tipo de fluxo para um canal com dimensões Nx x Ny.

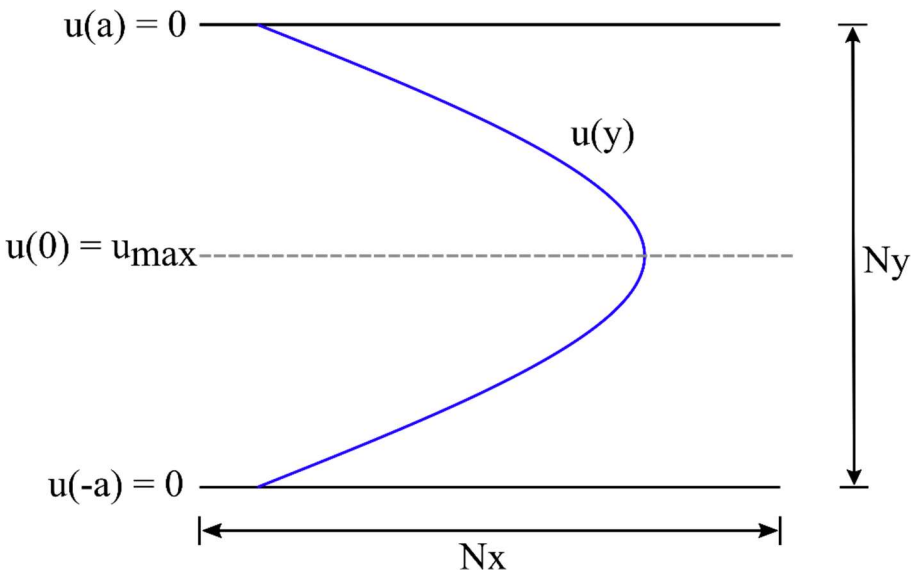

Figura 3.3: Geometria, condições de contorno e perfil de velocidades de um fluxo Poiseuille.

Para este tipo de fluxo foi feito uma análise do fluxo Poiseuille para um canal na horizontal para verificar a eficácia geral do MLB sem a adição de forças de corpos. Para este cenário, a condição de contorno proposta por Zou & He (1997) foi utilizada, considerando um perfil de velocidade descrito pela Eq. 4.1 nas células lattice de entrada do canal e uma condição de pressão nas células lattice de saída do canal.

Para este caso foi considerado as condições de contorno *bounce-back* nos contornos sólidos e condições periódicas. Além disso, as funções de distribuição de partículas das células lattice foram inicializadas considerando densidade igual a 1.0 kg/m^3 e vetor de velocidade (0.08, 0.0) m/s, respectivamente. Um resumo de todos os parâmetros e condições de contornos adotados nas simulações do fluxo Poiseuille são apresentados na Tabela 3.2.

Tabela 3.2: Resumo dos parâmetros e condições de contorno considerados na simulação do fluxo Poiseuille.

Parâmetros da simulação	
Velocidade máxima no canal (m/s)	0.1
Geometria do canal - Nx x Ny - (m)	500x100
Número de células lattice	50000
Densidade inicial das células (kg/m^3)	1.0
Vetor de velocidade inicial das células (m/s)	(0.08, 0.0)
Viscosidade Cinemática do fluido (m^2/s)	0.24
Tempo de Relaxação (s)	1.22
Espaçamento da malha lattice (m)	1
Passo de tempo da simulação (s)	1
Velocidade lattice (m/s)	1
Aplicação de forças de corpo	Não
Condição de contorno *bounce-back*	Sim
Condição de contorno periódica	Sim
Condição de contorno de Von Neumann	Sim
Condição de contorno de Dirichlet	Sim

3.2.2 FLUXO PASSANDO AO REDOR DE UM CILINDRO CIRCULAR

O fluxo ao redor de um cilindro é um problema que vem sendo estudado há muito tempo na mecânica de fluidos computacional (Breuer *et al*., 2000; Owen, 2011; Galindo-Torres, 2013). Uma das características mais interessantes deste problema é que para uma geometria simples e fixa é possível observar diferentes fenômenos em função das condições de fluxo (Ocampo-Gómez, 2013). A Figura 3.4 ilustra diversos perfis de fluxo ao redor de um cilindro em função da variação do número de Reynolds.

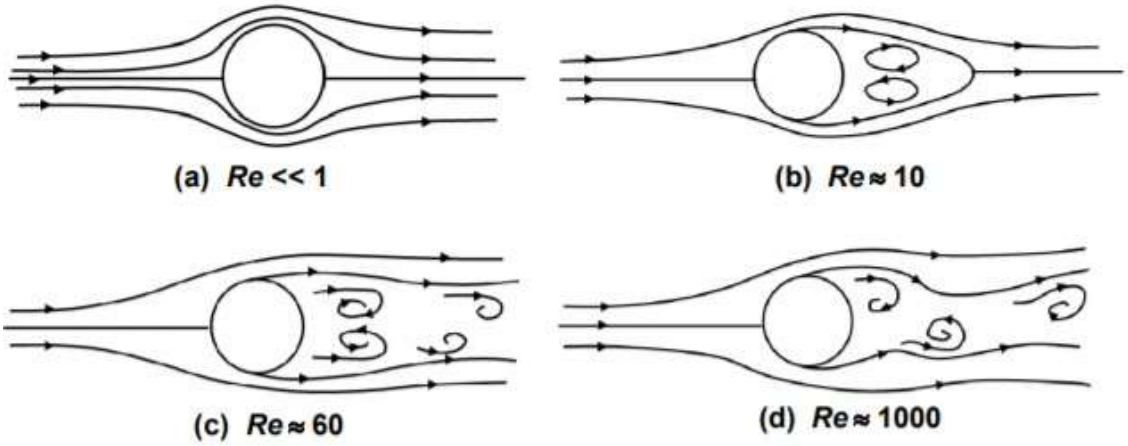

Figura 3.4: Perfis de fluxo ao redor de um cilindro em função do número de Reynolds (Sato & Kobayashi, 2012).

Este cenário consistiu em simular um canal retangular com adição de um cilindro circular para observar a influência do número de Reynolds no tipo de fluxo, laminar ou turbulento, desenvolvido no canal. A Figura 3.5 ilustra a geometria escolhida para realização da simulação.

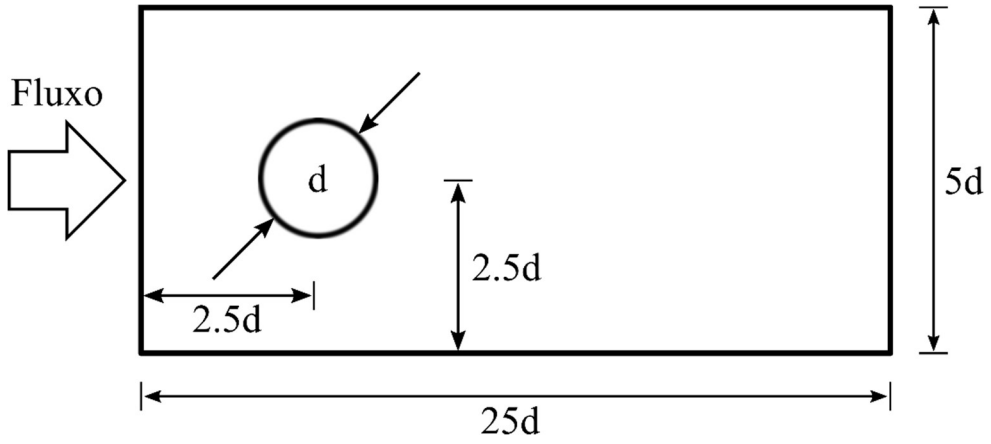

Figura 3.5: Geometria da simulação do fluxo de fluido ao redor de um cilindro circular.

Foram feitas 4 simulações variando o número de Reynolds em 5, 25, 45 e 100 e os seus resultados foram comparados com os resultados propostos por Taneda (1956). A variação do número de Reynolds tem influência direta na viscosidade cinemática do modelo e, por consequência, o tempo de relaxação de cada simulação será diferente. A influência do número de Reynolds na viscosidade cinemática é descrita por:

$$\nu = \frac{u_{max} d}{Re} \tag{3.2}$$

onde ν é a viscosidade cinemática do fluido, d é o diâmetro do cilindro, u_{max} é a máxima velocidade do fluido e Re é o número de Reynolds.

Assim como na simulação do fluxo Poiseuille foi considerado condições de contorno *bounce-back* nos contornos sólidos (paredes e ao redor do cilindro), condição de contorno periódica e as condições de Zou & He (1997). Além disso, as funções de distribuição das partículas foram inicializadas considerando valores iniciais de densidade e vetor de velocidade iguais a 1.0 kg/m^3 e (0.08, 0.0) m/s, respectivamente. Um resumo de todos os parâmetros e condições de contornos adotadas nessa simulação são apresentados na Tabela 3.3.

Tabela 3.3: Resumo dos parâmetros e condições de contorno considerados na simulação de fluxo de fluido ao redor de um cilindro circular.

Parâmetros da simulação	**Fluxo ao redor de cilindro circular**			
	Re = 5	Re = 25	Re = 45	Re = 100
Velocidade máxima no canal (m/s)	0.1	0.1	0.1	0.1
Geometria do canal - Nx x Ny - (m)	500x100	500x100	500x100	500x100
Número de células lattice	50000	50000	50000	50000
Densidade inicial das células (kg/m^3)	1.0	1.0	1.0	1.0
Vetor de velocidade inicial (m/s)	(0.08,0.0)	(0.08,0.0)	(0.08,0.0)	(0.08,0.0)
Viscosidade Cinemática do Fluido (m^2/s)	0.4	0.08	0.04	0.02
Tempo de Relaxação (s)	1.7	0.74	0.63	0.56
Espaçamento da malha lattice (m)	1	1	1	1
Passo de tempo da simulação (s)	1	1	1	1
Velocidade lattice (m/s)	1	1	1	1
Condição de contorno *bounce-back*	sim	sim	sim	sim
Condição de contorno periódica	sim	sim	sim	sim
Condição de contorno de Von Neumann	sim	sim	sim	sim
Condição de contorno de Dirichlet	sim	sim	sim	sim

3.2.3 PARTÍCULA SÓLIDA EM QUEDA LIVRE

Neste cenário, foi feita a simulação de uma partícula sólida em queda livre para avaliar a interação do corpo com o meio (paredes) e analisar a conservação de energia do sistema. A geometria da simulação é descrita na Figura 3.6.

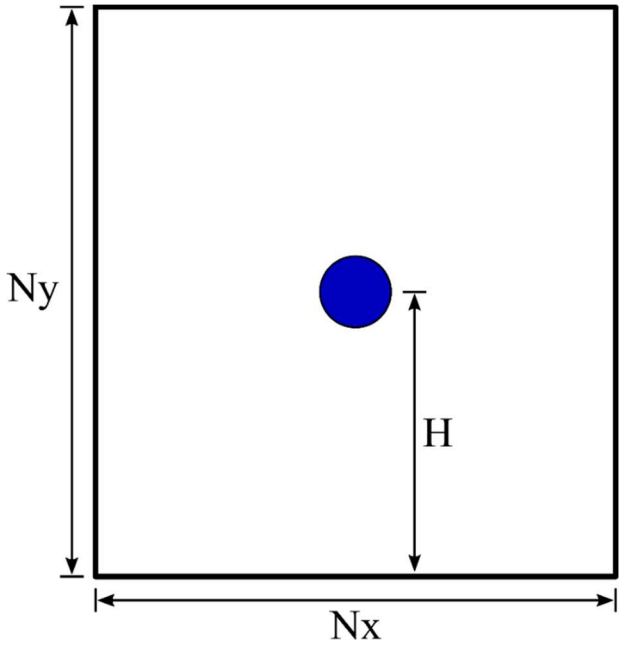

Figura 3.6: Geometria considerada na simulação de uma partícula em queda livre.

Primeiramente, foi feita uma simulação com coeficiente de amortecimento nulo para avaliar a conservação de energia de uma partícula com massa unitária e raio 0.5 m. Após essa primeira análise, verificou-se a influência do coeficiente de amortecimento no comportamento geral do sistema. Desta forma, foram feitas simulações com valores de coeficiente de amortecimento iguais a 0.1, 0.3, 0.5 e 0.7. O resumo de todos os parâmetros considerados nesta simulação é descrito na Tabela 3.4.

Tabela 3.4: Resumo dos parâmetros da partícula e do meio considerados na simulação de um corpo em queda livre.

Parâmetros da Simulação	**Partícula sólida em queda livre**				
	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$
Geometria - Nx x Ny - (m)	10x10	10x10	10x10	10x10	10x10
Massa (kg)	1.0	1.0	1.0	1.0	1.0
Raio (m)	0.5	0.5	0.5	0.5	0.5
H (m)	5	5	5	5	5
Fator de Segurança	0.1	0.1	0.1	0.1	0.1
Ângulo de atrito (graus)	30	30	30	30	30
Rigidez normal (Pa)	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$
Rigidez cisalhante (Pa)	$0.5x10^6$	$0.5x10^6$	$0.5x10^6$	$0.5x10^6$	$0.5x10^6$
Rigidez da parede (Pa)	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$

3.2.4 COLISÃO ENTRE PARTÍCULAS

Neste cenário, foram simuladas duas situações. No primeiro cenário foi feita a simulação de dois corpos, sendo um móvel e outro fixo, para avaliar as condições de contato entre os corpos. No segundo cenário foi feito uma análise de colisões entre corpos para avaliar as condições de conservação de momento linear entre os mesmos. A Figura 3.7 ilustra a geometria dos casos simulados.

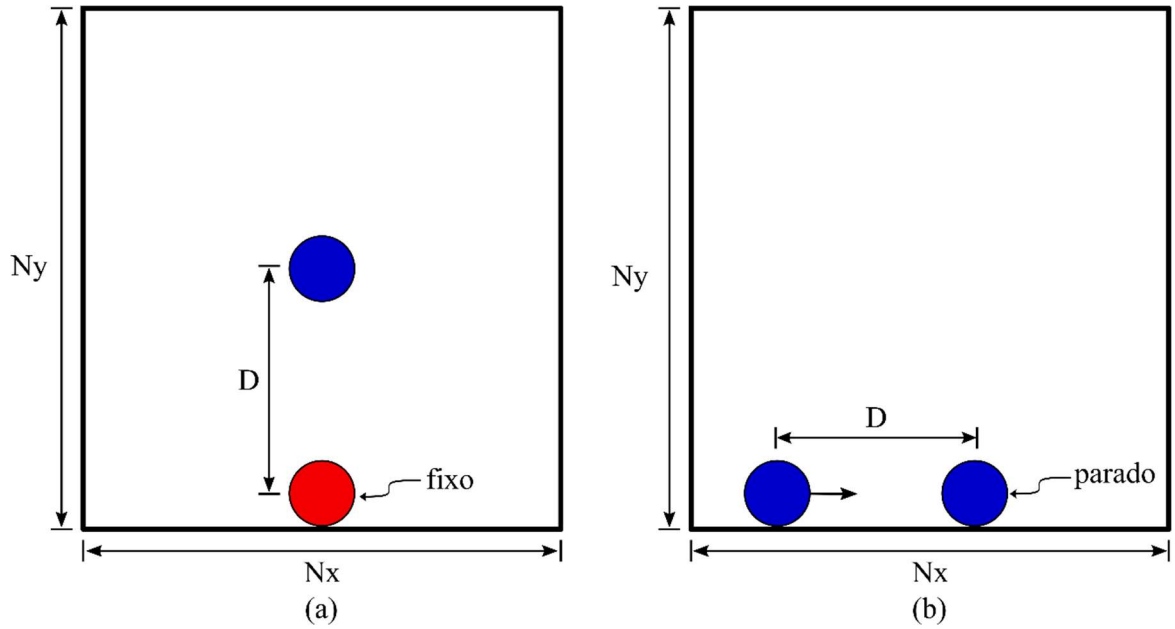

Figura 3.7: Geometria adotada para os casos a) disco quicando sobre outro; b) colisão entre dois discos.

Em ambos os casos, foi considerado que todos os corpos têm a mesma massa e raio e estão separados por uma distância D = 5 m. No segundo caso de simulação, a velocidade da partícula foi considerada de 4 m/s. Além disso, foram feitas diversas simulações alterando o valor do coeficiente de amortecimento para avaliar como o comportamento das partículas em colisão é afetado por esse parâmetro. O resumo de todos os parâmetros definidos para a realização dessas simulações é descrito na Tabela 3.5.

Tabela 3.5: Resumo dos parâmetros definidos para simulação de colisão entre partículas.

Parâmetros da Simulação	**Interação Partícula-Partícula**				
	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$
Geometria - Nx x Ny - (m)	10x10	10x10	10x10	10x10	10x10
Massa (kg)	1.0	1.0	1.0	1.0	1.0
Raio (m)	0.5	0.5	0.5	0.5	0.5
D (m)	5	5	5	5	5
Fator de Segurança	0.1	0.1	0.1	0.1	0.1

Ângulo de atrito (graus)	30	30	30	30	30
Rigidez normal (Pa)	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$
Rigidez cisalhante (Pa)	$0.5x10^6$	$0.5x10^6$	$0.5x10^6$	$0.5x10^6$	$0.5x10^6$
Rigidez da parede (Pa)	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$	$1.0x10^6$

3.3 SIMULAÇÃO DA INTERAÇÃO FLUIDO-SÓLIDO

Neste cenário foram feitas simulações para validar o acoplamento dos métodos implementados e analisar o comportamento da interação fluido-sólido. A geometria para este cenário consistiu na adição de uma partícula sólida na base de um canal e aplicação de um perfil de velocidade na entrada do canal descritos na Figura 3.8. O código para este cenário é apresentado no Apêndice C.

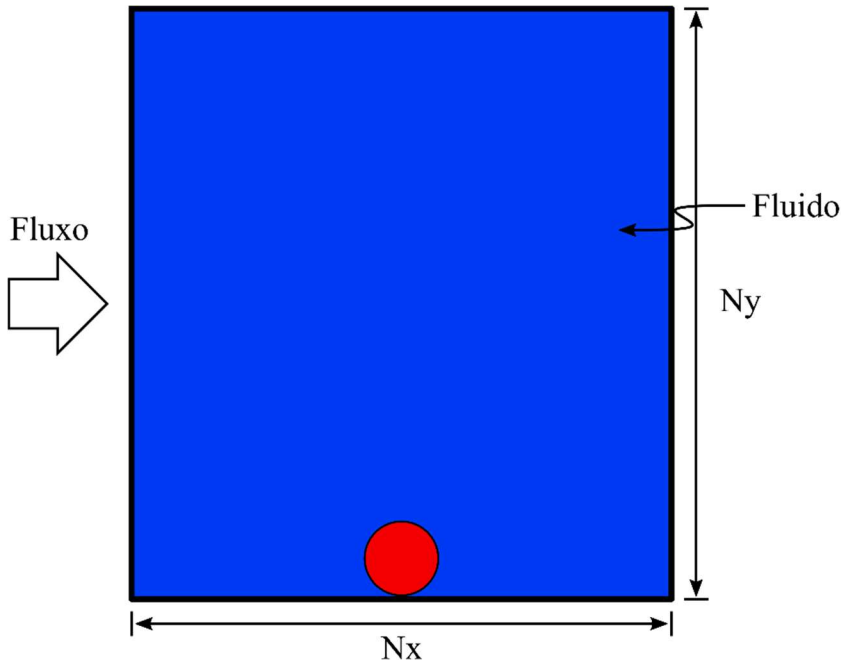

Figura 3.8: Geometria considerada na simulação da interação fluido-sólido e sólido-sólido.

Foram desenvolvidos dois tipos de cenários de simulação para este caso. No primeiro caso, foi feita a simulação para uma partícula sólida fixa de modo a avaliar o coeficiente de arrasto e validar se o cálculo da força e torque hidrodinâmicos estão coerentes. O coeficiente de arrasto pode ser determinado da seguinte maneira:

$$C_d = \frac{8F_x}{\rho u_{max}^2 \pi d^2} \tag{3.3}$$

onde C_d é o coeficiente de arrasto, ρ é a densidade do fluido, F_x é a componente horizontal da força hidrodinâmica atuante na partícula e d é o diâmetro da partícula. Os resultados calculados

pela Eq. 3.3 foram comparados com a solução analítica (White, 1991) em função do número de Reynolds descrita pela seguinte equação:

$$C_d = \frac{24}{Re} + \frac{6}{1+\sqrt{Re}} + 0.4 \tag{3.4}$$

No segundo caso, a partícula foi deixa interagir com o fluido e foi analisado o perfil de energia desenvolvido na simulação. A condição de contorno adotada para o fluido em ambos os cenários foi a condição periódica e as funções de distribuição foram inicializadas com densidade e vetor de velocidade iguais a 1.0 kg/m^3 e (0.1, 0.0) m/s, respectivamente. A partícula sólida adicionada tem massa unitária e raio 0.5 m. O resumo de todos os parâmetros considerados para as simulações é apresentado na Tabela 3.6.

Tabela 3.6: Resumo dos parâmetros definidos para a simulação da interação fluido-sólido e sólido-sólido.

Parâmetros da simulação	
Geometria do canal - Nx x Ny - (m)	500x100
Velocidade máxima no canal (m/s)	0.1
Espaçamento da malha lattice (m)	1
Passo de tempo da simulação (s)	1
Velocidade lattice (m/s)	1
Condição de contorno *bounce-back*	não
Condição de contorno periódica	sim
Condição de contorno de Von Neumann	não
Condição de contorno de Dirichlet	não
Massa da partícula (kg)	1.0
Raio (m)	0.5
Coeficiente de amortecimento	0.1
Fator de Segurança	0.1
Ângulo de atrito (graus)	30
Rigidez normal do contato partícula-partícula (Pa)	1.0×10^6
Rigidez cisalhante do contato partícula-partícula (Pa)	0.5×10^6
Rigidez do contato parede-partícula (Pa)	1.0×10^6

CAPÍTULO IV

4 ANÁLISE DOS RESULTADOS

O presente capítulo apresenta a visualização e processamento dos dados dos cenários de simulação apresentados anteriormente. Para todos os casos é mostrado a geometria do domínio discretizada e são apresentas informações de densidade e perfis de velocidade para o fluido e a avaliação de energia e momento linear para as partículas sólidas.

4.1 FLUXO POISEUILLE

4.1.1 GEOMETRIA

A geometria do canal foi discretizada em células lattice divididas em dois tipos: células lattice que representam o fluido e células lattice que representam contornos sólidos. No código, a classe Lattice apresenta a variável *node* responsável por guardar informações de cada tipo de célula, recebendo o valor 0 quando a célula representa um fluido e 1 quando esta representa um contorno sólido. A Figura 4.1 ilustra a geometria discretizada do canal em células lattice. As células em azul representam o fluido e as células em vermelho representam os contornos sólidos.

Figura 4.1: Geometria discretizada do canal para simulação de fluxo Poiseuille.

4.1.2 DENSIDADE

A densidade é um fator importante a ser analisado em uma simulação utilizando o MLB. Como se está trabalhando com fluxo incompressível, espera-se que a densidade não varie muito durante a simulação. Foram observadas pequenas oscilações do valor de densidade entre 1.005 e 1.015 kg/m^3, porém como esse valor está muito próximo do valor inicial de 1.0 kg/m^3, e considerou-se que estas variações não influenciariam resultando em um fluido pouco compressível. Desta forma, o método lattice Boltzmann é capaz de simular fluxos incompressíveis do ponto de vista prático. A Figura 4.2 ilustra o perfil de densidade desenvolvido durante a simulação de fluxo Poiseuille em um canal. O valor da densidade para os contornos sólidos foi considerado como 0, representadas pela cor azul escuro.

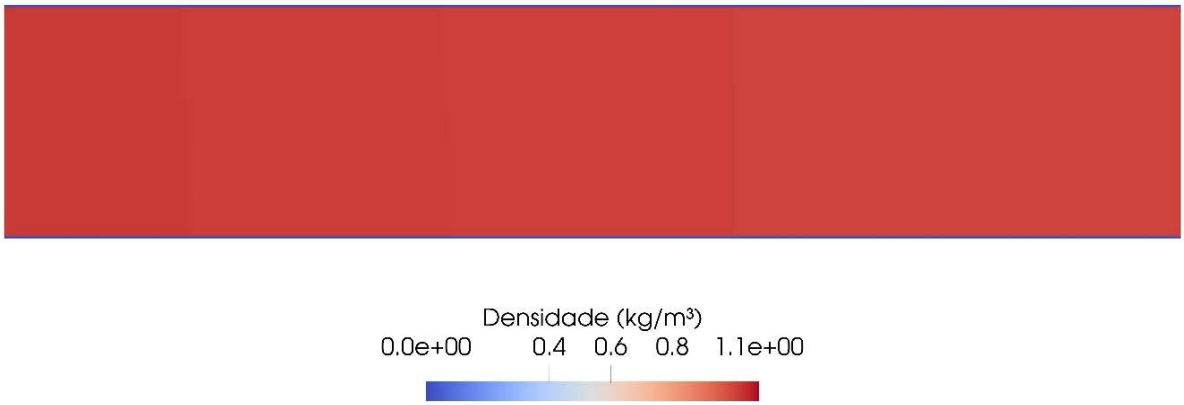

Figura 4.2: Distribuição da densidade do fluido durante a simulação de fluxo Poiseuille.

4.1.3 PERFIL DE VELOCIDADE

A aplicação das condições de contorno de velocidade no início do canal gerou um perfil de fluxo Poiseuille por todo o canal no decorrer de toda a simulação. Foram observadas algumas oscilações nos valores de velocidade, porém em média o perfil é coerente com aquele esperado para este tipo de fluxo. A Figura 4.3 ilustra o desenvolvimento do fluxo Poiseuille em um canal.

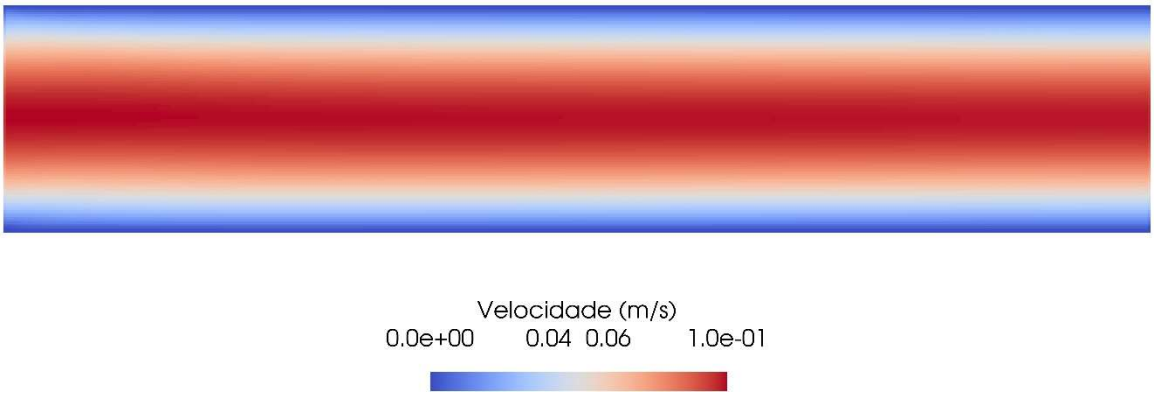

Figura 4.3: Perfil de velocidade de um fluxo Poiseuille em um canal.

Além da visualização do perfil de velocidade, foram feitas medidas dos valores de velocidade na entrada, no meio e na saída do canal para fazer comparações com a solução analítica para este tipo de problema, como ilustrado na Figura 4.4. Pode-se perceber que os valores de entrada, meio e saída do canal se aproximam muito dos valores apresentados pela solução analítica demonstrando, assim, a eficácia geral do MLB.

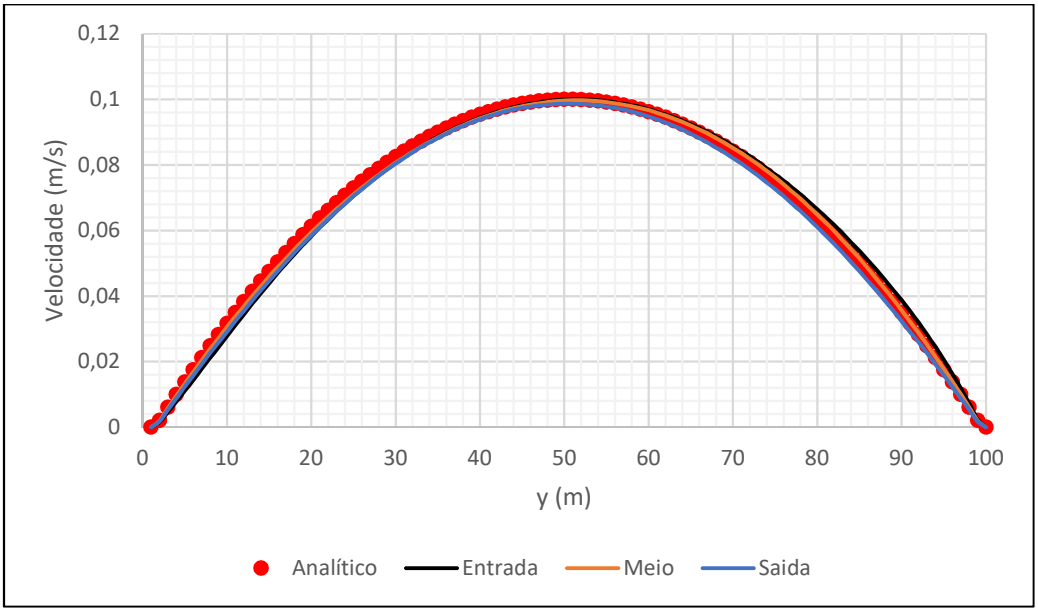

Figura 4.4: Perfil de velocidade da simulação de fluxo Poiseuille.

4.2 FLUXO PASANDO AO REDOR DE UM CILINDRO CIRCULAR

4.2.1 GEOMETRIA

Assim como no cenário de simulação do fluxo Poiseuille, as células lattice foram divididas em células de fluido em azul e as que representa os obstáculos, paredes e o cilindro,

da simulação em vermelho. A geometria escolhida para a realização das simulações nesse cenário é descrita na Figura 4.5.

Figura 4.5: Geometria definida para o fluxo de fluido ao redor de um cilindro circular.

A adição do cilindro foi feita por meio de células que respeitassem o critério de uma célula estar contida em um círculo. Assim, a discretização do cilindro apresenta algumas "irregularidades" no seu contorno, porém este aspecto não influenciou os resultados da simulação.

4.2.2 DENSIDADE

O perfil de densidade desenvolvido em todas as simulações apresentou algumas oscilações entre 1.04 e 1.1 kg/m^3. A formulação do problema, com a utilização da equação de estado (Eq. 2.8), de fato considera o fluido levemente compressível. Desta forma, faz sentido que o obstáculo represe o fluxo, fazendo com que a densidade seja levemente superior a montante e sofra um leve alívio à jusante, porém foi considerado que essas variações não influenciaram o comportamento (quase) incompressível do fluido e isto foi percebido para todas as simulações onde o número de Reynolds foi variado. Assim como no exemplo de fluxo Poiseuille a densidade dos contornos sólidos foi considerada 0 kg/m^3. O perfil de densidade desenvolvido para uma simulação com número de Reynolds igual a 5 é apresentado na Figura 4.6.

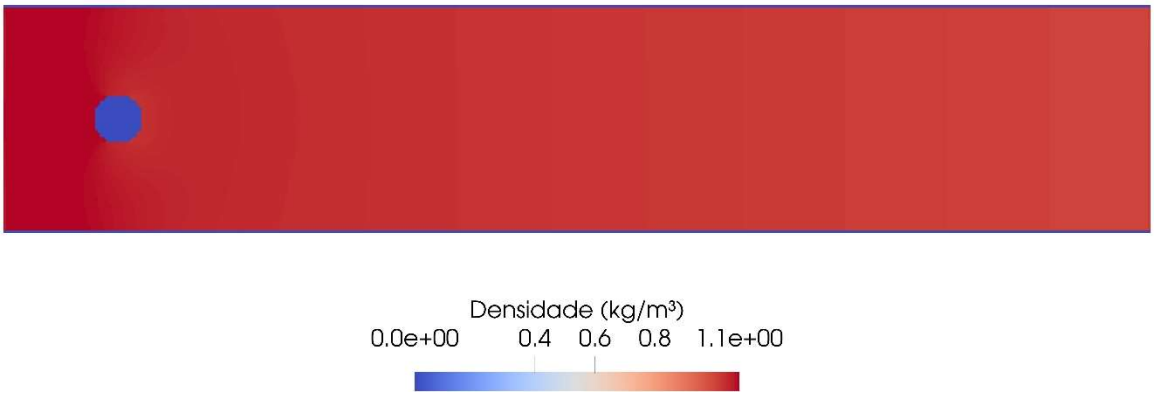

Figura 4.6: Perfil de densidade desenvolvido durante uma simulação de fluxo de fluido ao redor de um cilindro com *Re* = 5.

4.2.3 PERFIL DE VELOCIDADE

O perfil de velocidade desenvolvido para este cenário é altamente influenciado pelo número de Reynolds escolhido para a simulação. Taneda (1956) analisou o comportamento das linhas de fluxo por trás de um cilindro circular em função do número de Reynolds e os seus resultados são ilustrados na Figura 4.7.

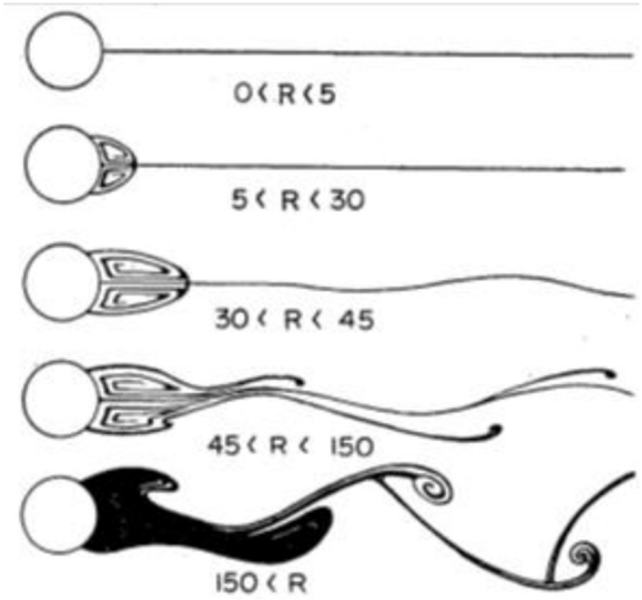

Figura 4.7: Análise de linhas de fluxo por trás de um cilindro circular (Taneda, 1956).

Em todas as simulações foi aplicado o perfil de velocidade Poiseuille nas células de entrada e foi verificado o perfil de velocidade em três locais: na entrada, na metade do cilindro e na saída do canal.

A primeira simulação foi feita para número de Reynolds igual 5. Neste cenário, o regime do fluxo de fluido observado foi laminar e foi o mesmo observado por Taneda em seu estudo. O perfil de velocidade mostrou uma pequena variação dos perfil Poiseuille mostrado na Figura 4.4, porém verificou-se que essa variação não influenciou o resultado geral da simulação. As linhas de fluxo e o perfil de velocidade desenvolvidas nesta simulação são apresentadas nas Figuras 4.8 e 4.9, respectivamente.

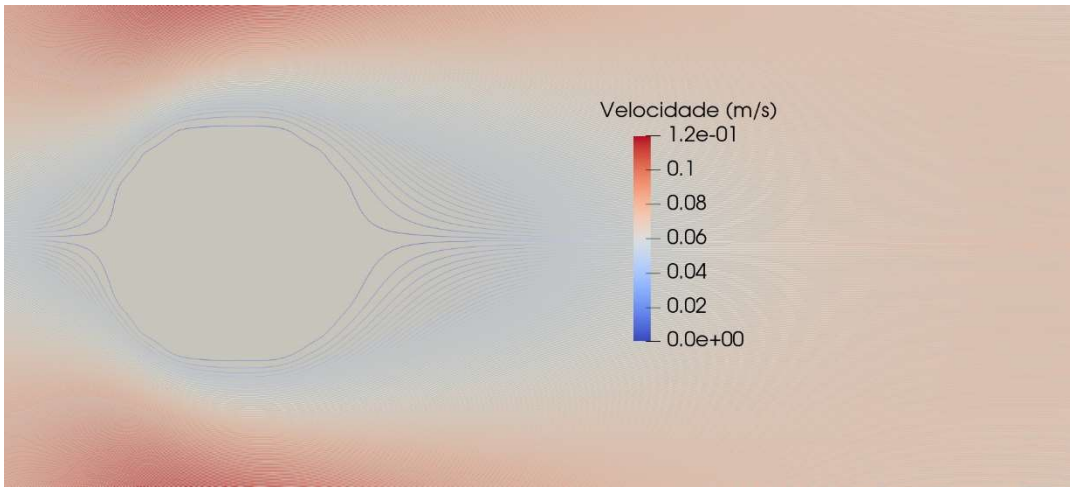

Figura 4.8: Linhas de fluxo desenvolvidas na simulação de fluxo passando por um cilindro circular e $Re = 5$.

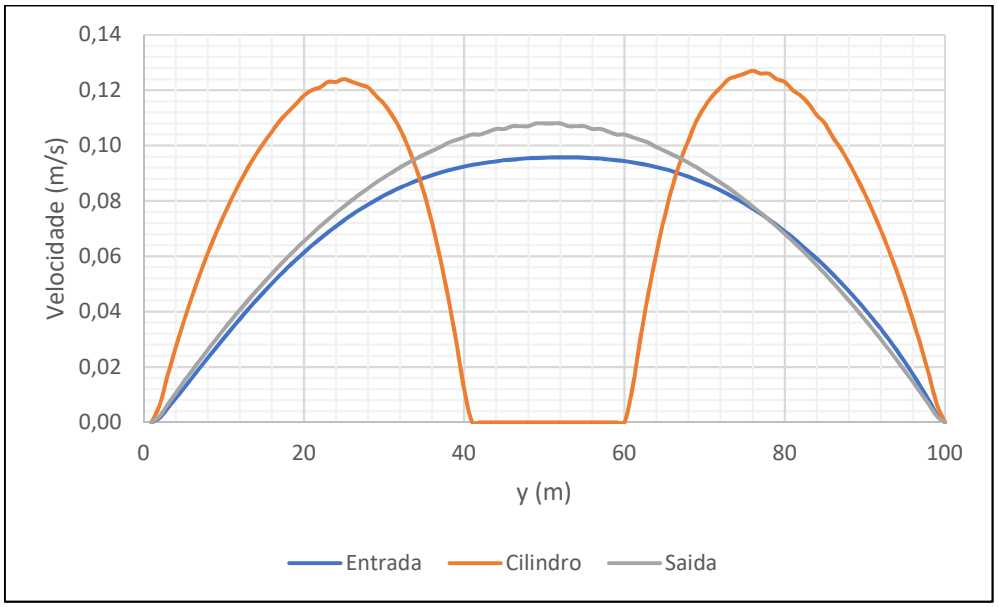

Figura 4.9: Perfil de velocidade desenvolvido na simulação em três seções: entrada, no meio do cilindro e na saída do canal.

Nesta análise foi verificado se o MLB atende às condições de conservação de massa. Para tal, foi verificado se o fluxo de massa ($\int_{-a}^{+a} \rho . u_x . dy$) se mantinha (quase) constante ao

longo do canal. Assim, as áreas dos perfis de velocidades ilustrados na Figura 4.9 foram calculadas, sendo encontrados os valores de 6.1, 6.1 e 6.0 kg.m/s para as seções de entrada, no meio do cilindro e saída, respectivamente.

Para as outras simulações, verificou-se que ao aumentar o valor do número de Reynolds, o regime desenvolvido no canal passava a formar vórtices, indicando o aparecimento de um regime turbulento no canal. A formação de vórtices também foi estudada por Taneda (1956), que fez uma relação entre o comprimento dos vórtices formados atrás do cilindro, o diâmetro do cilindro e o número de Reynolds do meio. Ocampo-Gómez (2013) compilou os dados apresentados por Taneda (1956) e formulou um gráfico destas relações ilustrado na Figura 4.10.

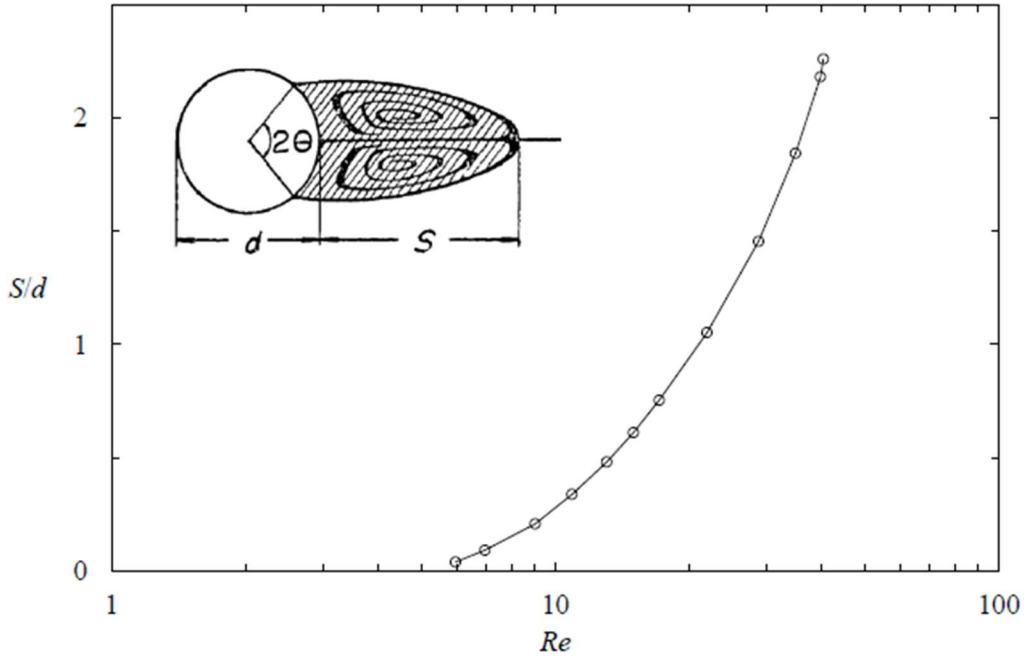

Figura 4.10: Relação entre o diâmetro do cilindro, o comprimento do vórtice e o número de Reynolds (Ocampo-Gómez, 2013).

A simulação com *Re* = 25 apresentou perfil de linhas de fluxo muito próximas as encontradas por Taneda. Além disso, a relação S/d encontrada foi de 1.4, valor muito próximo ao sugerido pelo gráfico da Figura 4.10 com S/d = 1.3. A Figura 4.11 ilustra as linhas de fluxo para esta simulação.

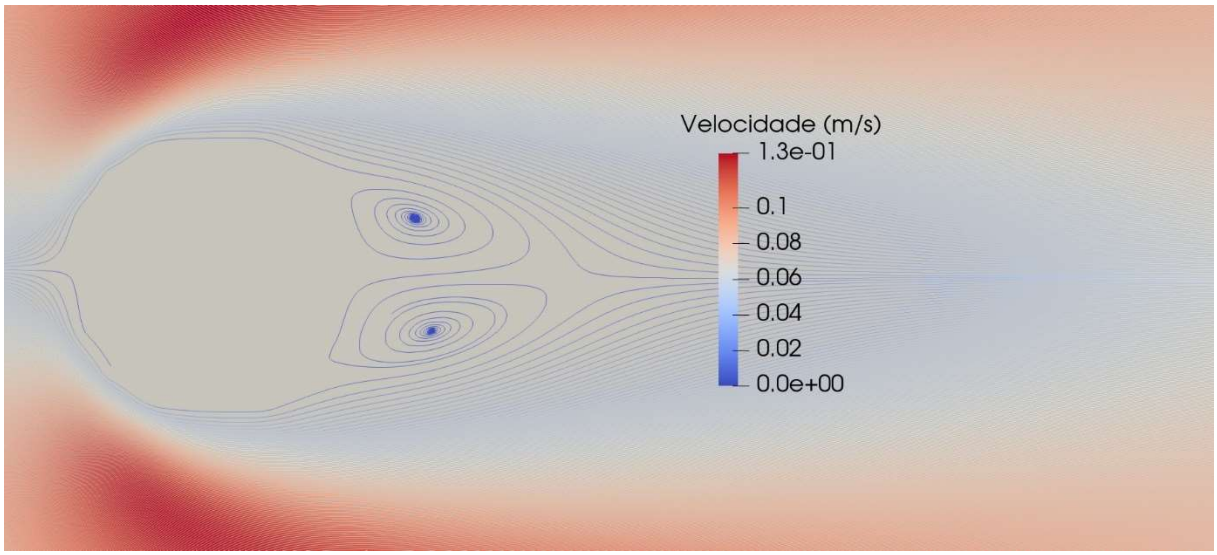

Figura 4.11: Linhas de fluxo desenvolvidas na simulação de fluxo passando por um cilindro circular e *Re* = 25.

A simulação com *Re* = 45 também apresentou um perfil de linhas de fluxo muito parecidas as encontradas por Taneda e apresentou relação S/d = 2.2, resultado muito próximo ao encontrado pela Figura 4.10 de 2.1. Acredita-se que para esses dois casos, aproximações feitas pelo código possam ter causado essas oscilações. A Figura 4.12 ilustra as linhas de fluxo para esta simulação.

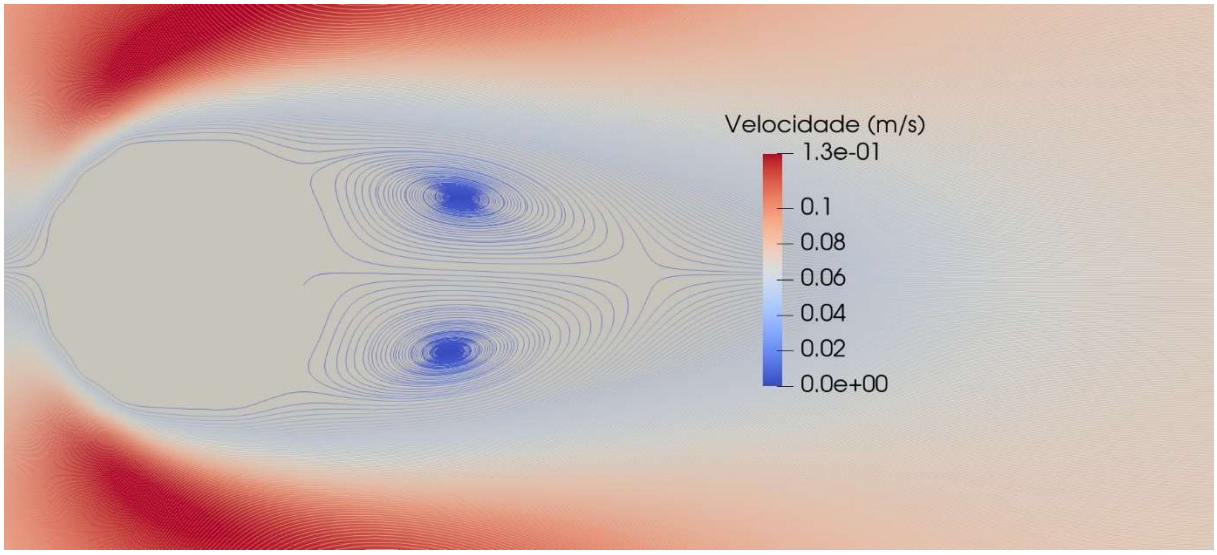

Figura 4.12: Linhas de fluxo desenvolvidas na simulação de fluxo passando por um cilindro circular e *Re* = 45.

A última simulação foi feita para *Re* = 100. Neste cenário, não foi possível fazer a avaliação da relação S/d devido à turbulência desenvolvida no canal. Além disso, foi verificado uma oscilação do perfil de velocidade Poiseuille na saída do canal, acredita-se que para este

cenário a formação de vórtices não é estabilizada no comprimento total do canal, devendo este comprimento ser aumentado para garantir tal estabilidade. As figuras 4.13 e 4.14 ilustram o perfil de velocidade total no canal e o perfil de velocidade para três seções na entrada, na metade do cilindro e na saída do canal, respectivamente.

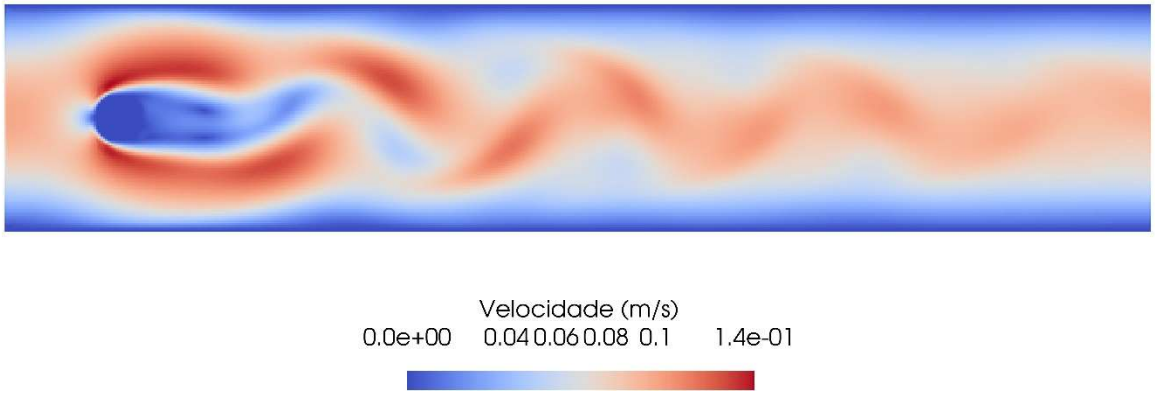

Figura 4.13: Perfil de velocidade desenvolvido na simulação de fluxo passando por um cilindro com *Re* = 100.

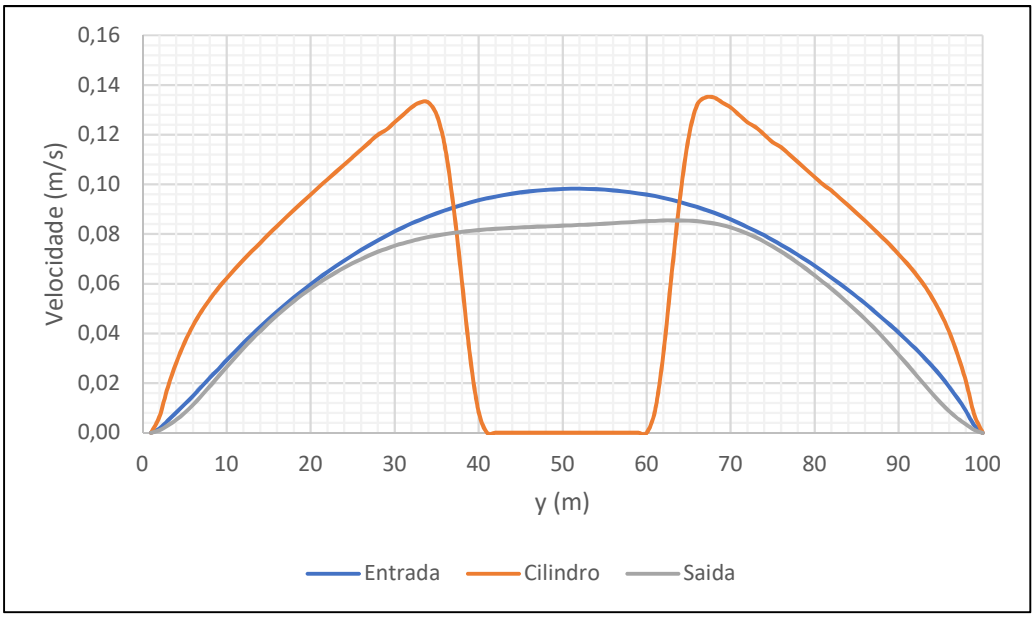

Figura 4.14: Perfil de velocidade desenvolvido na simulação com *Re* = 100 em três seções: entrada, no meio do cilindro e na saída do canal.

É possível perceber na Figura 4.13 a formação de redemoinhos durante a simulação, o que pode ter sido descrito pela grande formação de vórtices por trás do cilindro ilustrados na Figura 4.15.

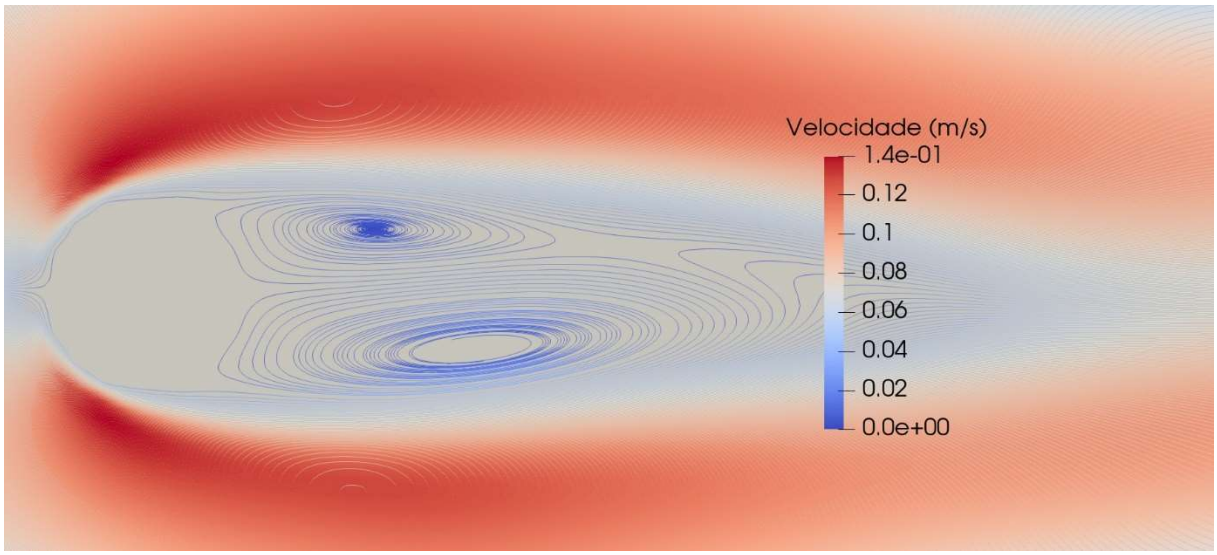

Figura 4.15: Linhas de fluxo desenvolvidas na simulação de fluxo passando por um cilindro circular e *Re* = 100.

A mesma análise de conservação de massa foi feita para este cenário. Foram avaliados os valores de fluxo de massa para as seções de entrada, no cilindro e saída e os seus valores são 6.0, 6.0 e 6.0 kg.m/s, respectivamente. Percebe-se que para os cenários de *Re* = 5 e *Re* = 100 os valores de fluxo são bem próximos, o que garante que o modelo respeita a conservação de massa para fluxos incompressíveis.

Por fim, percebeu-se que para o método lattice Boltzmann implementado nesta dissertação, esse cenário não consegue ultrapassar números de Reynolds maiores que 100, o que acarreta no aparecimento de instabilidades na simulação. Acredita-se que ao aumentar o número de Reynolds a condição limitante do número *Mach* seja violada, o que é caracterizado pelas instabilidades. Além disso, pôde-se perceber que o método lattice Boltzmann é capaz de simular problemas de fluxo com uma precisão significativa, o que justifica o seu uso em problemas de fluxo de fluidos em meios porosos.

4.3 PARTÍCULA EM QUEDA LIVRE

4.3.1 GEOMETRIA

A geometria escolhida para esta simulação consistiu numa caixa para simbolizar as paredes e uma partícula sólida foi deixada cair do centro desta caixa. A Figura 4.16 ilustra a geometria definida para todas as simulações envolvendo a queda de uma partícula.

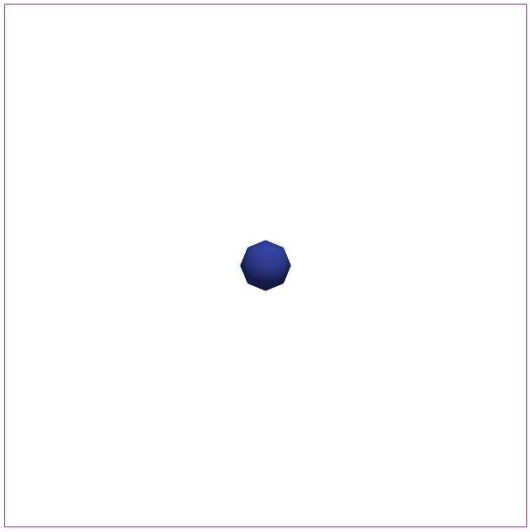

Figura 4.16: Geometria definida para a simulação da queda de uma partícula.

4.3.2 PERFIS DE ENERGIA

Foi verificado o perfil de energia cinética e potencial gravitacional para os casos onde o coeficiente de amortecimento foi variado nos seguintes valores: 0, 0.1, 0.3, 0.5 e 0.7. Esta variação é necessária para que este parâmetro seja calibrado para ser utilizado em outras simulações. Neste primeiro caso, pretende-se avaliar a sua interferência quando uma partícula interage com o meio. Para todos os casos, a energia foi calculada tendo como referência a quina inferior à esquerda, assim, mesmo que a partícula esteja parada sobre a parede, ela ainda terá energia potencial.

No primeiro caso, foi feita a avaliação da partícula em queda livre, ou seja, sem amortecimento. Percebeu-se que os perfis de energia potencial gravitacional e cinética determinados pelo Hybrid2D estão de acordo com a solução analítica do problema. Os perfis de energia cinética e potencial gravitacional são ilustrados na Figura 4.17.

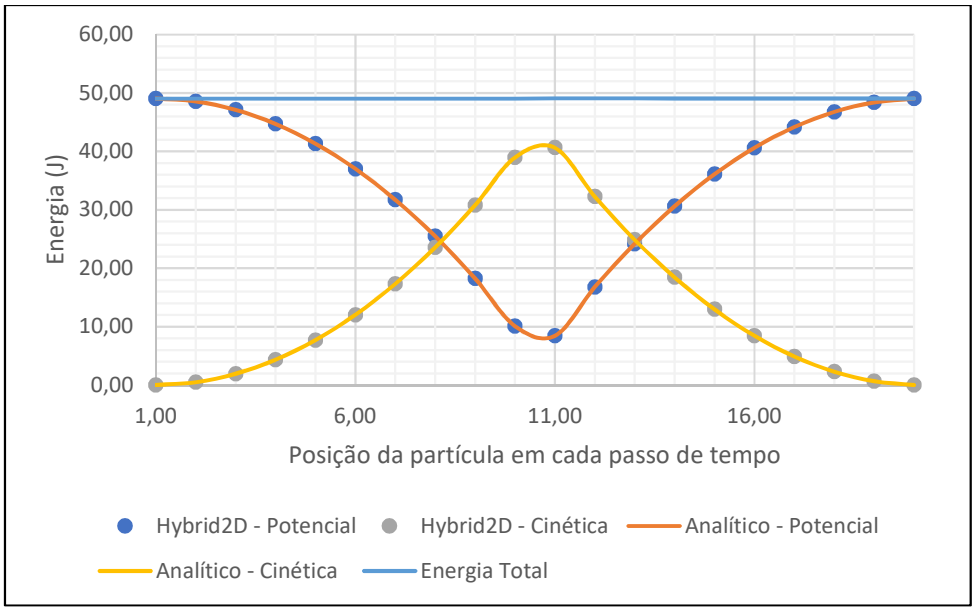

Figura 4.17: Perfil de energia cinética e potencial gravitacional para uma partícula em queda livre (coeficiente de amortecimento = 0.0).

Nas outras simulações, foi feita a variação do coeficiente de amortecimento. As figuras 4.18, 4.19, 4.20 e 4.21 ilustram o perfil de energia cinética e potencial gravitacional para os casos de coeficiente de amortecimento de 0.1, 0.3, 0.5 e 0.7, respectivamente.

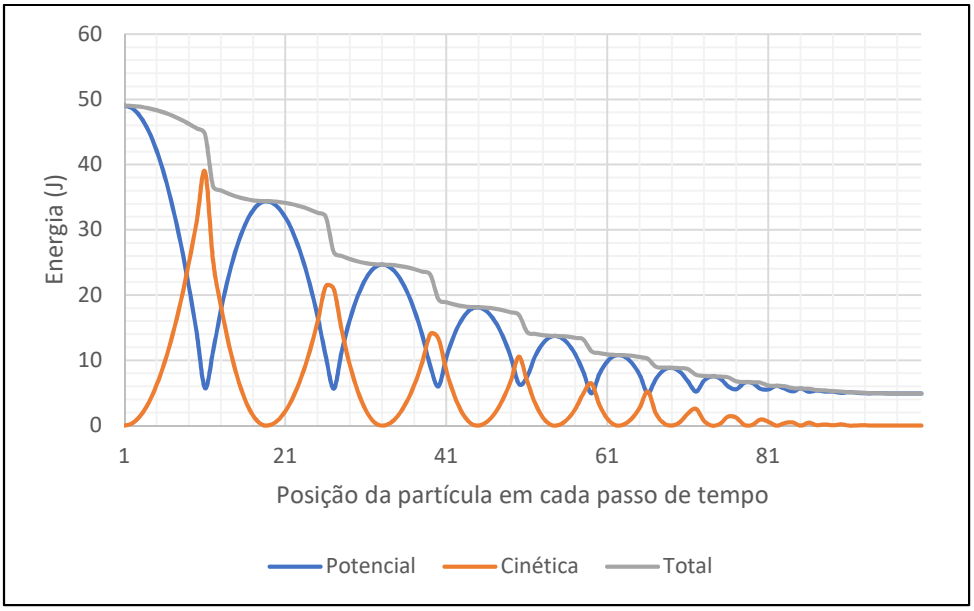

Figura 4.18: Perfil de energia potencial gravitacional e cinética para uma partícula caindo com coeficiente de amortecimento de 0.1.

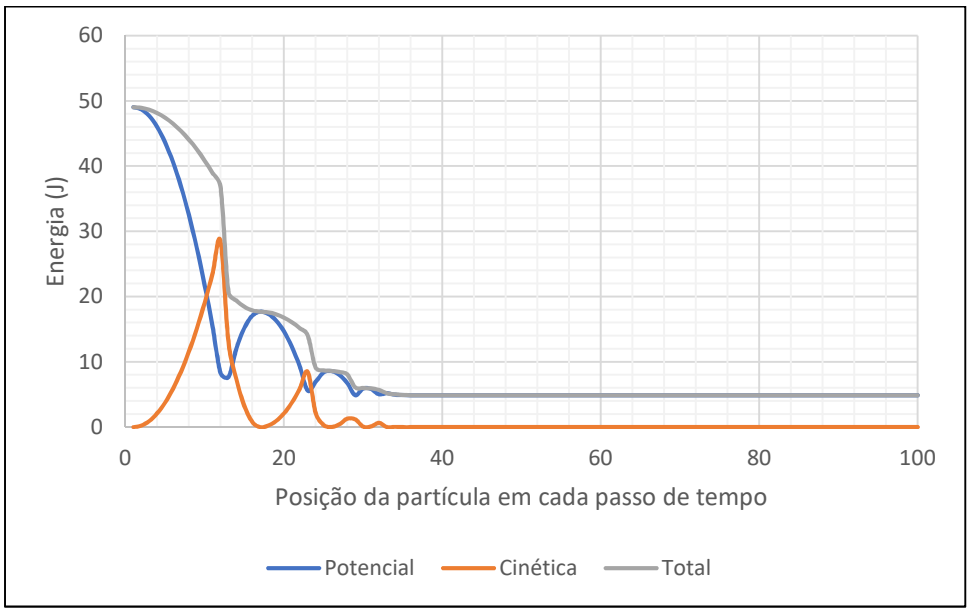

Figura 4.19: Perfil de energia potencial gravitacional e cinética para uma partícula caindo com coeficiente de amortecimento de 0.3.

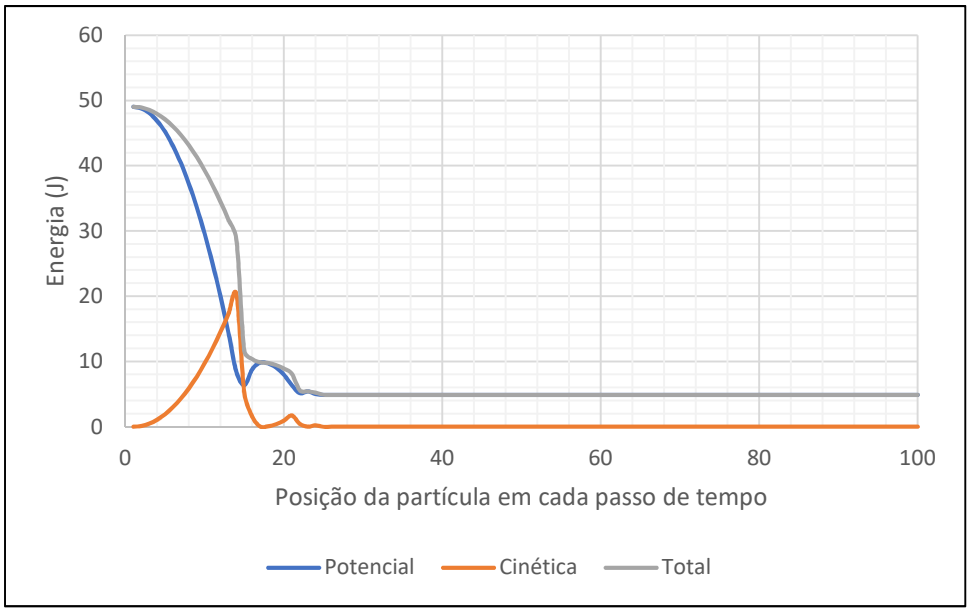

Figura 4.20: Perfil de energia potencial gravitacional e cinética para uma partícula caindo com coeficiente de amortecimento de 0.5.

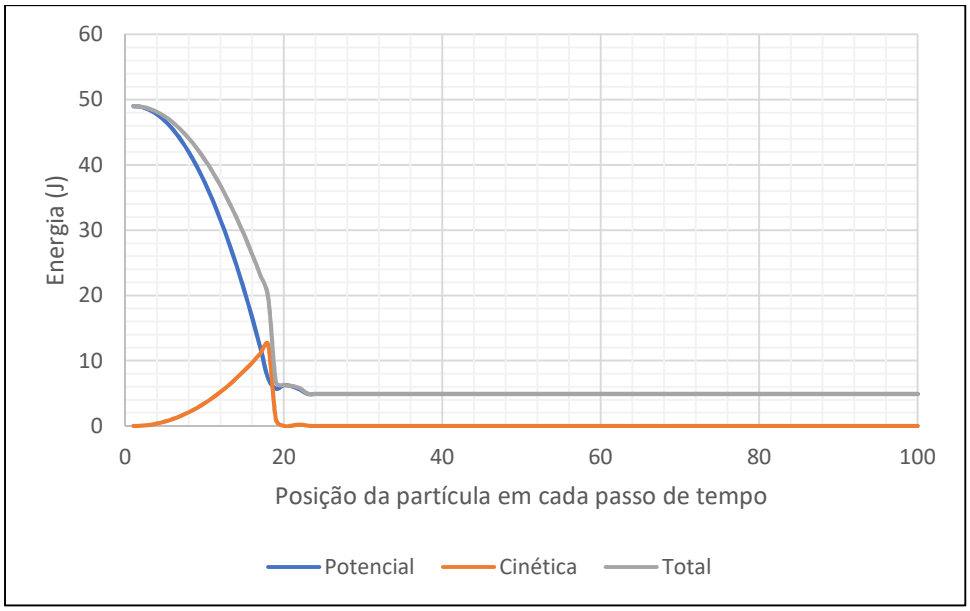

Figura 4.21: Perfil de energia potencial gravitacional e cinética para uma partícula caindo com coeficiente de amortecimento de 0.7.

Pode-se analisar que para os coeficientes de amortecimento de 0.3, 0.5 e 0.7 a partícula chega ao equilíbrio de energia numa taxa muito rápida e que talvez não reflita a realidade para este caso de simulação. Por outro lado, para o coeficiente de amortecimento de 0.1 a partícula é levada ao equilíbrio, mas ainda apresenta uma oscilação dos valores de energia para então chegar ao equilíbrio geral. Desta forma, nessa primeira análise o coeficiente de amortecimento ideal para a simulação da interação de uma partícula com o meio é de 0.1.

4.4 COLISÃO ENTRE PARTÍCULAS

4.4.1 GEOMETRIA

Essa simulação foi dividida em dois cenários. No primeiro, fez-se a simulação de um caso típico com o programa Yade (Šmilauer *et al*., 2015*)*) que é um disco quicando sobre o outro, com a finalidade de avaliar a eficiência do Hybrid2D em comparação com outro programa. O outro cenário consistiu na colisão entre partículas. As figuras 4.22 e 4.23 ilustram a geometria de cada cenário avaliado nessas simulações.

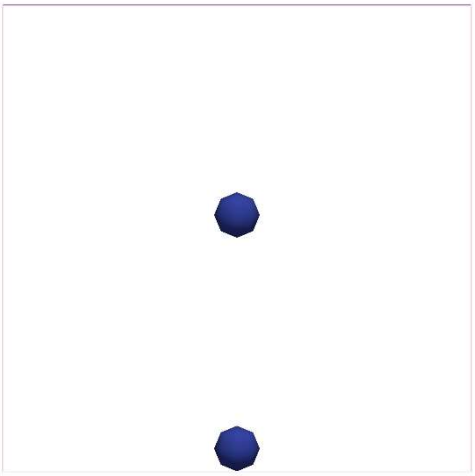

Figura 4.22: Geometria definida para a simulação de um disco quicando sobre o outro.

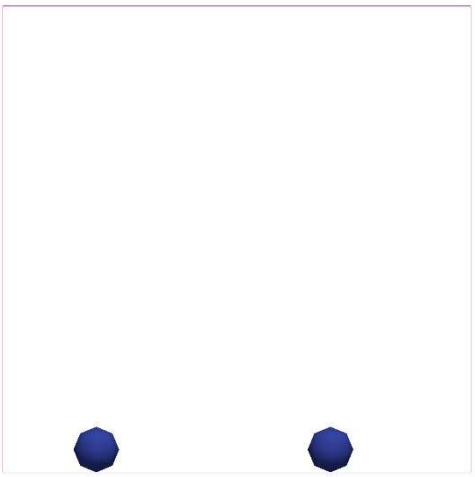

Figura 4.23: Geometria definida para a simulação da interação partícula-partícula.

4.4.2 DISCO QUICANDO SOBRE OUTRO

Neste cenário, o objetivo é verificar se o programa Hybrid2D e o Yade apresentam resultados semelhantes. Assim, não foram feitas variações no coeficiente de amortecimento das partículas, sendo considerado um coeficiente de amortecimento de 0.1, e foi verificado a energia potencial da partícula. Pode-se perceber que houve uma concordância muito significante dos valores de energia calculados pelos programas Hybrid2D e Yade, o que leva a crer que o programa Hybrid2D tem um ótimo potencial para simulação de meios discretos. A Figura 4.24 ilustra o perfil de energia potencial gravitacional da partícula sólida determinado pelos programas Hybrid2D e Yade para um coeficiente de amortecimento de 0.1.

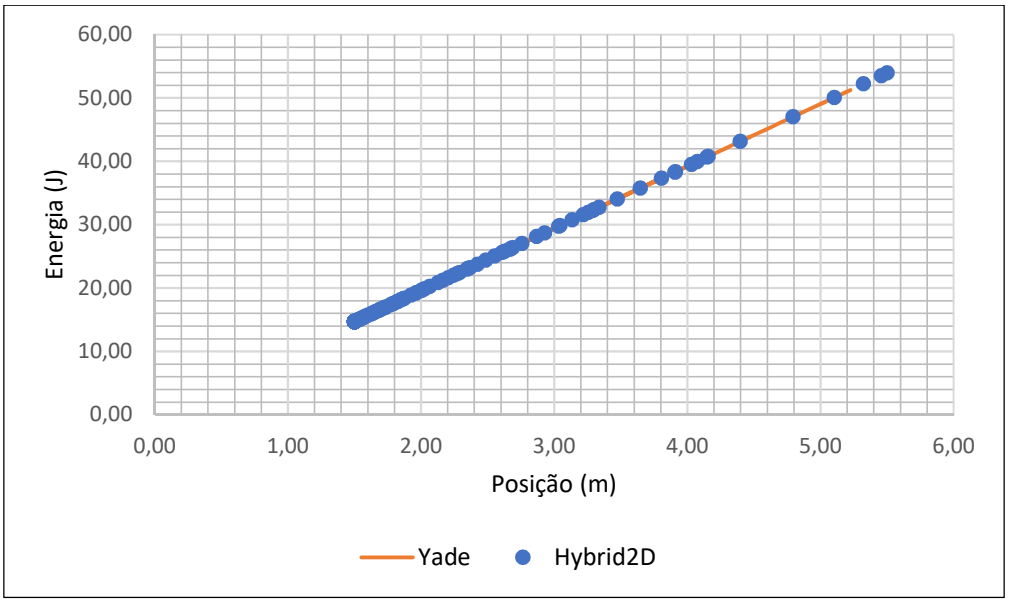

Figura 4.24: Perfil de energia potencial determinados com os programas Hybrid2D e Yade para um coeficiente de amortecimento de 0.1.

4.4.3 COLISÕES ELÁSTICAS E INELÁSTICAS

O primeiro teste feito nesse cenário de simulação foi considerando o valor do coeficiente de amortecimento igual a 0, ou seja, uma colisão elástica entre partículas. Neste tipo de colisão, o momento linear das partículas é conservado, então o momento linear de uma partícula deve ser totalmente transferido para outra partícula sem que haja perda pela colisão partícula-partícula ou partícula-parede. A Figura 4.25 ilustra o perfil de momento linear das partículas para um coeficiente de amortecimento nulo.

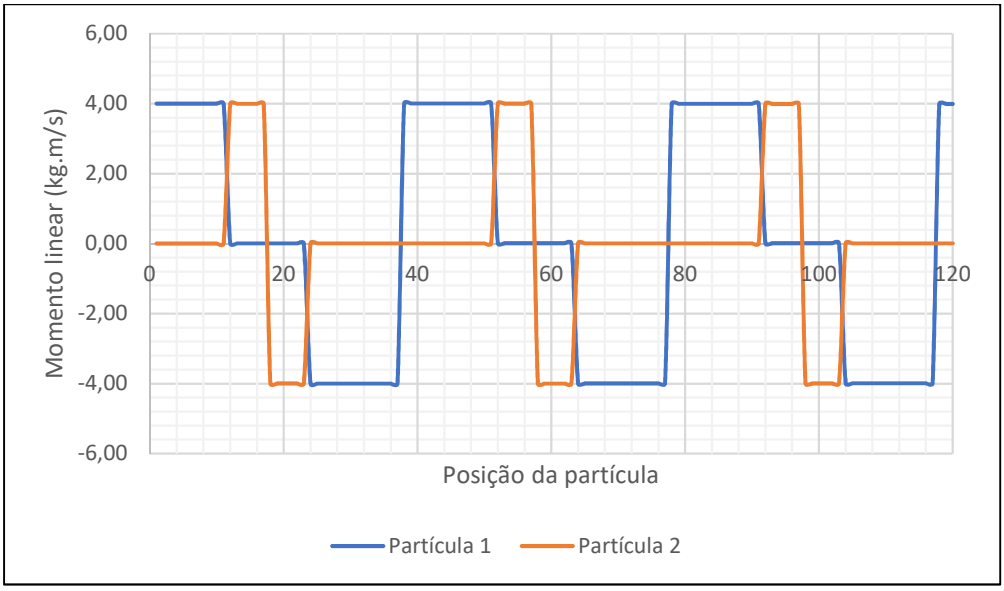

Figura 4.25: Perfil de momento linear das partículas com coeficiente de amortecimento nulo.

Pode ser visto que o momento linear das partículas se conservou mesmo havendo interação entre partículas e o meio. Esse cenário mostra a capacidade de interação entre duas partículas o que pode ser estendido para um conjunto delas, sendo assim possível simular meios particulados com o programa Hybrid2D. Prosseguiu-se com as outras simulações para avaliar o papel do coeficiente de amortecimento na simulação e poder calibrar esse parâmetro para situações em que ocorram interações entre partículas. As figuras 4.26, 4.27, 4.28 e 4.29 ilustram o perfil de momento linear desenvolvendo na colisão entre duas partículas para coeficientes de amortecimento iguais a 0.1, 0.3, 0.5 e 0.7, respectivamente.

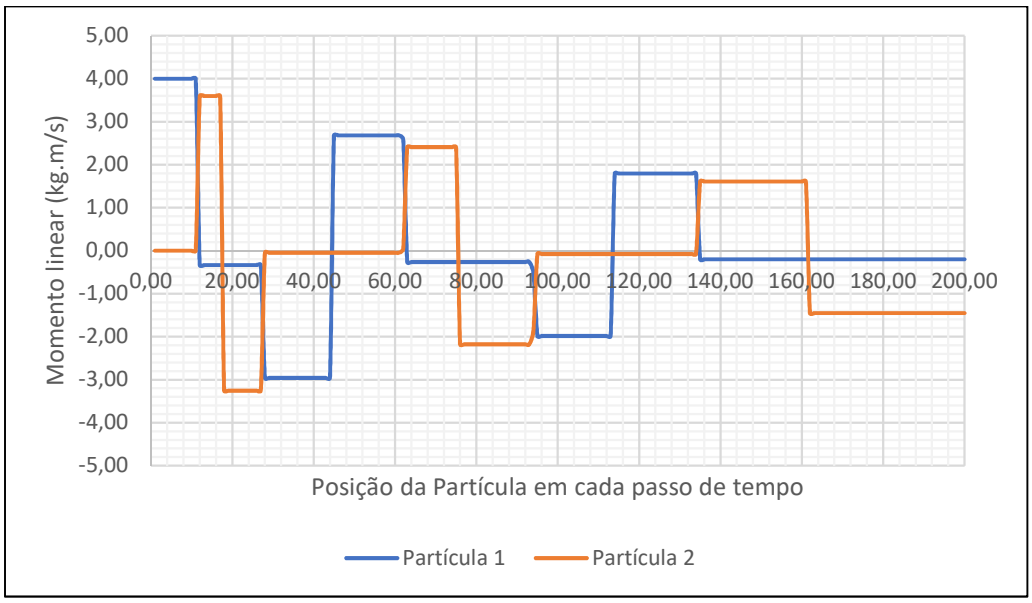

Figura 4.26: Perfil de momento linear das partículas com coeficiente de amortecimento 0.1.

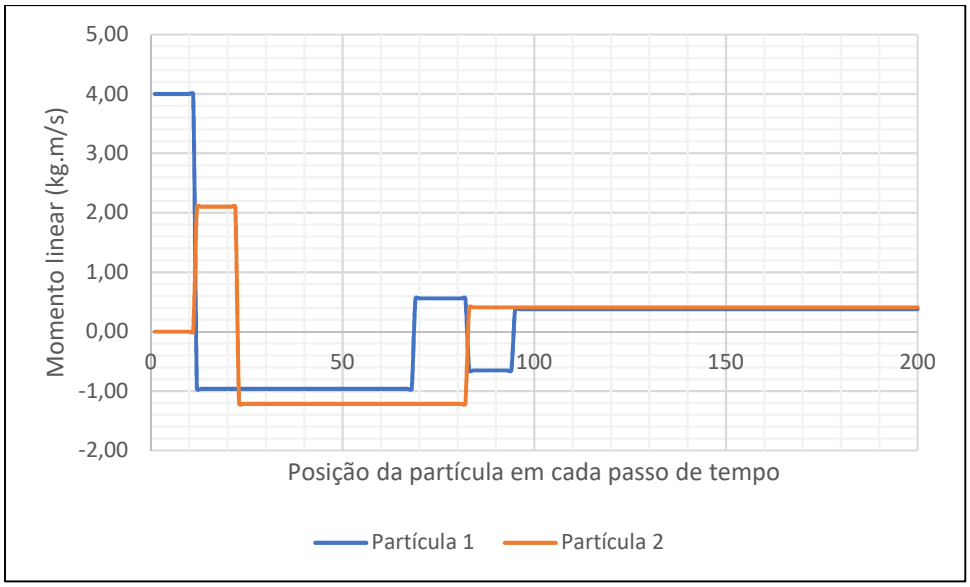

Figura 4.27: Perfil de momento linear das partículas com coeficiente de amortecimento 0.3.

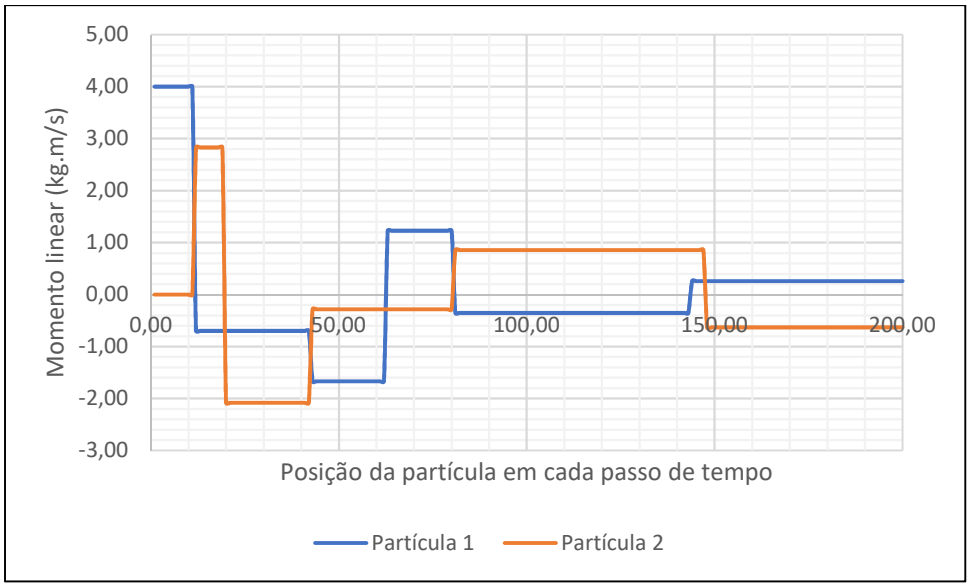

Figura 4.28: Perfil de momento linear das partículas com coeficiente de amortecimento 0.5.

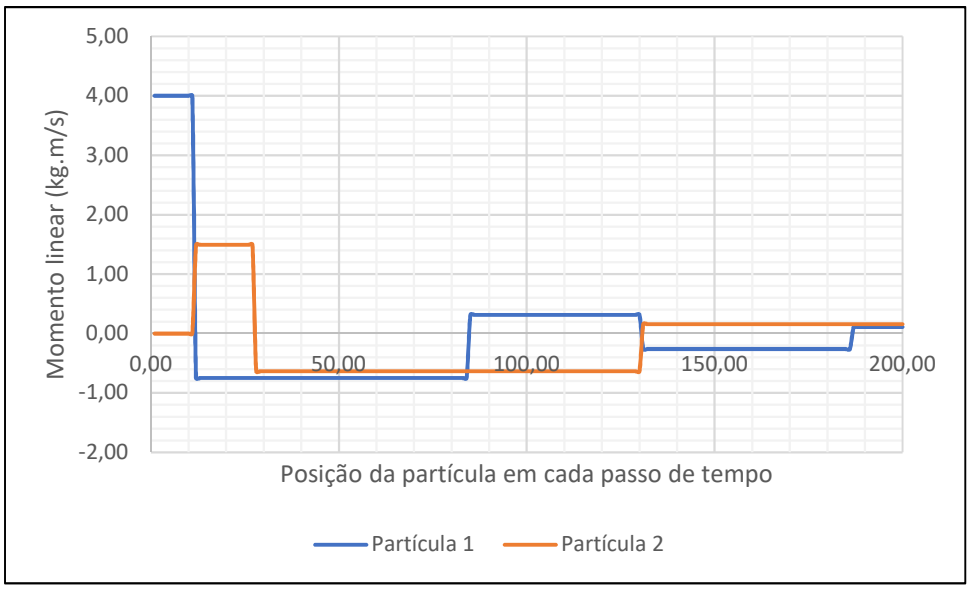

Figura 4.29: Perfil de momento linear das partículas com coeficiente de amortecimento 0.7.

Pelas análises os coeficientes de amortecimento 0.3, 0.5 e 0.7 levam ao equilíbrio de momento linear numa taxa muito rápida. Assim, o coeficiente de amortecimento 0.1 apresentou os valores mais coerentes, assim como na interação de uma partícula com o meio. Desta forma, o coeficiente de amortecimento calibrado para a simulação envolvendo os métodos MLB e MED foi definido como 0.1.

4.5 INTERAÇÃO FLUIDO-SÓLIDO

4.5.1 GEOMETRIA

A geometria da simulação para todos os casos simulados consistiu na adição de uma partícula logo no início do canal e discretização das células lattice como feito anteriormente, neste caso, porém, não foi utilizado paredes, sendo considerado contornos sólidos somente a partícula, assim, todas as células em azul representam o fluido naquela seção do canal. A geometria para este cenário de simulação é descrita na Figura 4.30.

Figura 4.30: Geometria definida para a simulação da interação fluido-sólido.

4.5.2 DENSIDADE

A densidade em todas as simulações com variação do número de Reynolds apresentou algumas oscilações no seu valor como nos casos anteriores, variando entre 0.98 e 1.02 kg/m^3. Essa oscilação foi considerada muito sutil o que garantiu que o fluido simulado era praticamente incompressível. O perfil de densidades desenvolvido no canal durante as simulações é ilustrado na Figura 4.31.

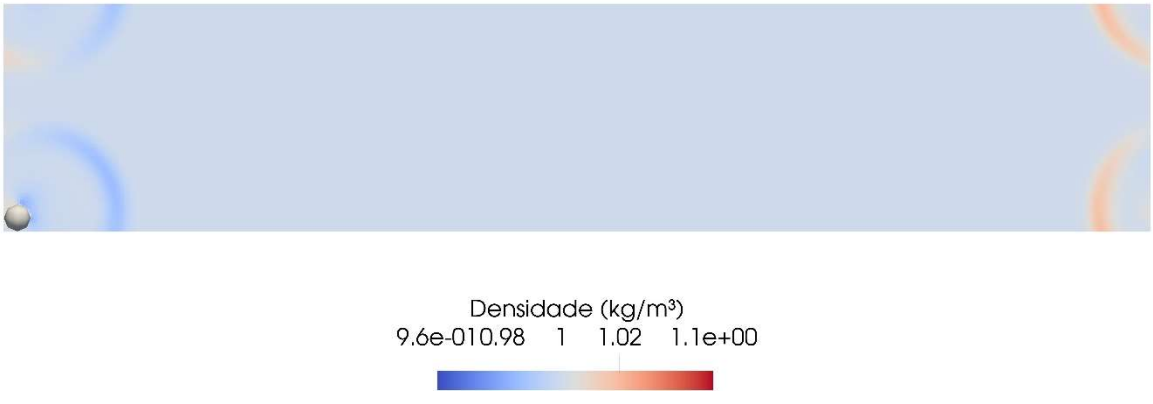

Figura 4.31: Perfil de densidade desenvolvido durante a simulação da interação fluido-sólido.

4.5.3 COEFICIENTE DE ARRASTO

A análise do coeficiente de arrasto é essencial para verificar se o cálculo da força hidrodinâmica está sendo feita de forma coerente e, assim, possibilitar que o fluido possa transmitir momento para as partículas. Para este caso, a partícula foi fixada, então, mesmo que a sua velocidade seja atualizada os seus graus de liberdade foram fixados, o que possibilitou a análise deste parâmetro. A Figura 4.32 ilustra os resultados determinados para o coeficiente de arrasto em função do número de Reynolds pelo programa Hybrid2D e pela solução analítica.

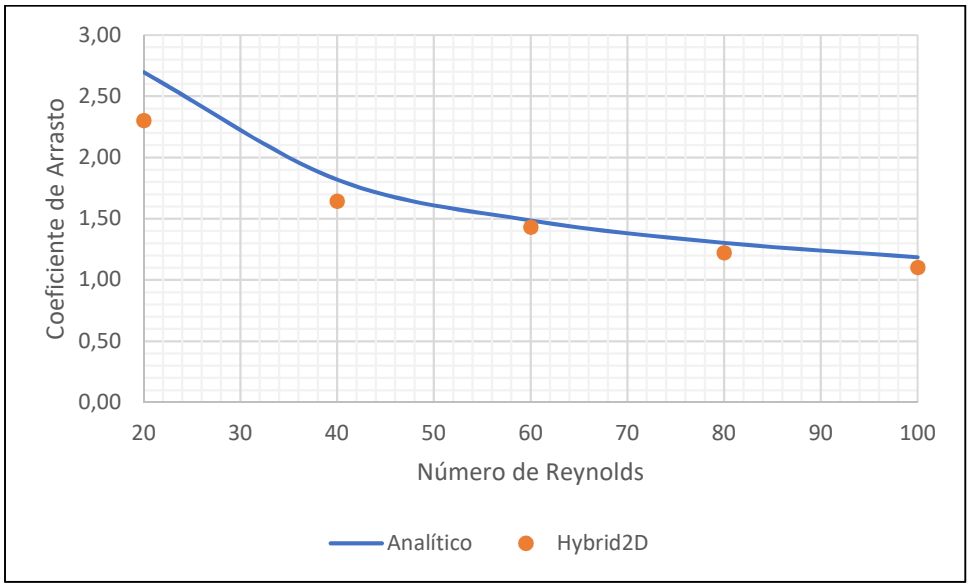

Figura 4.32: Valores de coeficiente de arrasto determinados pelo programa Hybrid2D e de forma analítica.

Os valores de coeficiente de arrasto determinados pelo programa Hybrid2D se aproximam muito da solução analítica para valores de número de Reynolds de 60, 80 e 100. Por outro lado, o programa Hybrid2D subestimou os valores de coeficiente de arrasto para números de Reynolds menores que 40, porém os valores ainda se aproximam da solução real do problema. Acredita-se que a formulação do MLB e arredondamentos feitos pelo próprio código possam ter gerado essas inconsistências, porém os valores determinados ainda se encontram bem próximos da solução analítica. Desta forma, foi considerado que o método das fronteiras móveis se apresentou como um método viável para o acoplamento dos modelos implementados, tornando o Hybrid2D uma ferramenta com um elevado potencial para simulação de interações entre fluidos e meios particulados.

4.6 PARTÍCULA SÓLIDA EM ARRASTO

Este cenário de simulação considerou a mesma geometria ilustrada na Figura 4.30 e consistiu em deixar a partícula sólida livre para avaliar a interação do fluido com a partícula e o correspondente arrasto transmitido para ela. É importante destacar que a partícula sofre tanto translação quanto rotação, porém só os efeitos de translação foram observados nesta simulação. Acredita-se que neste caso específico não houve rotação, pois, a força hidrodinâmica aplicada se concentrou no centro de gravidade da partícula sólida. A Figura 4.33 ilustra o arrasto de uma partícula sólida sujeita a um campo de velocidades para diversos passos de tempo da simulação.

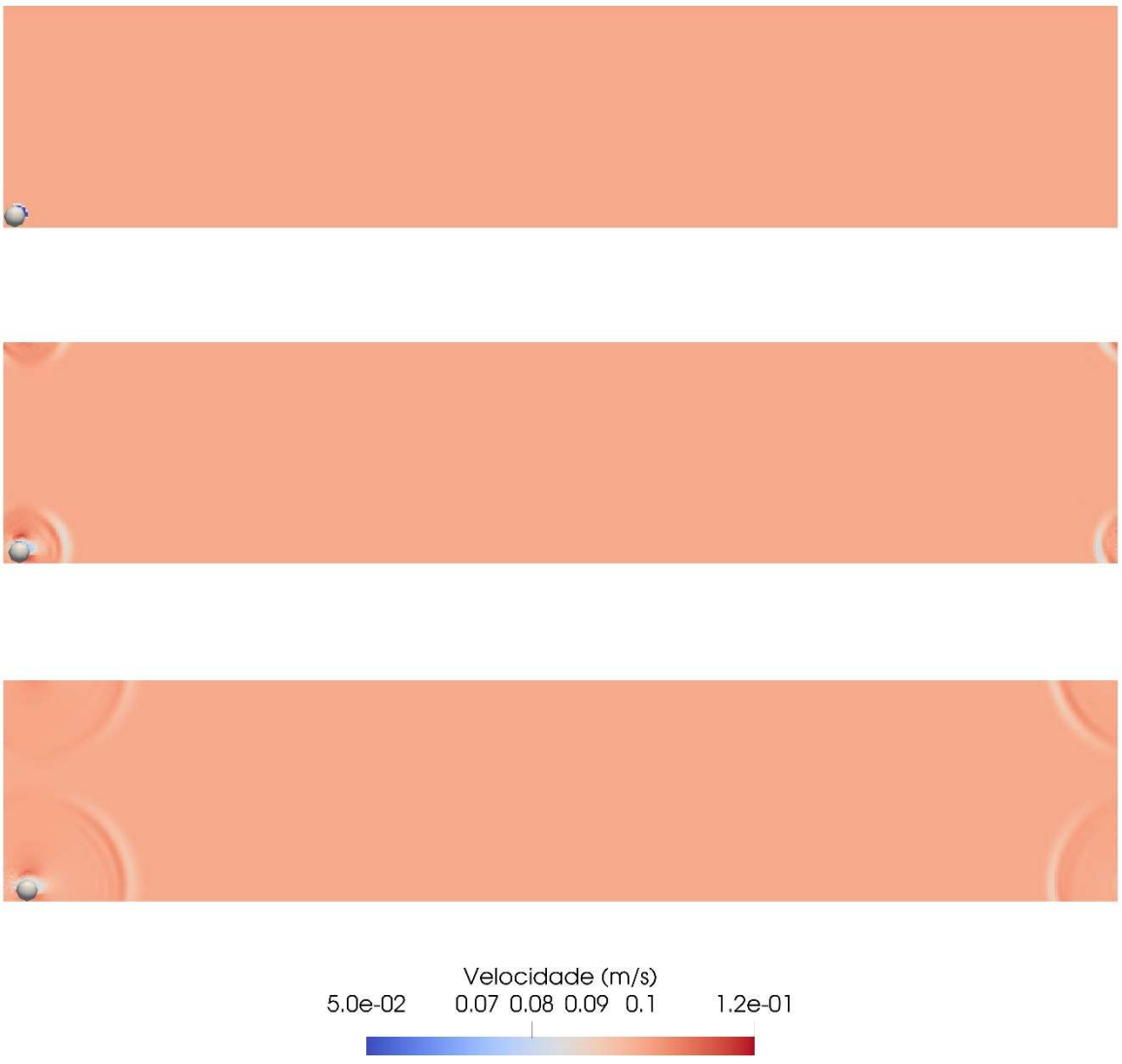

Figura 4.33: Arrasto de uma partícula sólida sujeita a um campo de velocidade para diversos passos de tempo da simulação.

Pode-se perceber que a interação entre o fluido e o sólido está ocorrendo já que não há nenhuma outra força horizontal atuando na partícula, além da força hidrodinâmica. Esta interação é traduzida no fluido com o desenvolvimento de ondas de choque e na partícula com o seu arrasto o que demonstra o potencial do programa Hybrid2D.

Durante a simulação, pode ser verificado o desenvolvimento de ondas de choque em partes do fluido que não estão em contato com a partícula, isto ocorre devido à condição de contorno periódica imposta. Além disso, percebe-se que a simulação apresenta alguns pontos instáveis, acredita-se que pela utilização de um modelo com um único tempo de relaxação e sem a utilização de modelos mais robustos no MLB possam ter causado essas inconsistências.

Além dessa análise visual, foi feita também uma verificação do comportamento cinemático da partícula sujeita a um campo de velocidade constantes. Percebeu-se que a partícula inicialmente sofre uma pequena aceleração que é caracterizado por uma zona de instabilidade no gráfico devido à inércia da partícula, porém, ao longo do tempo a sua velocidade se estabiliza para valores muito próximos ao do fluido. A Figura 4.34 ilustra o perfil de velocidades desenvolvido pela partícula sólida.

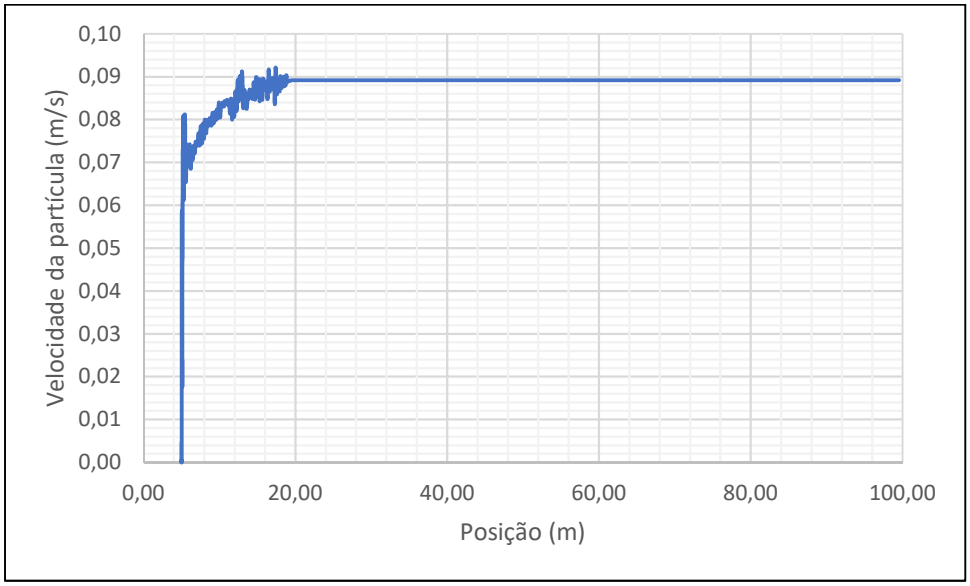

Figura 4.34: Perfil de velocidade desenvolvido pela partícula sólida em arrasto.

Era esperado que, ao longo do tempo, as velocidades do fluido e partícula se igualassem, porém, devido algumas pequenas instabilidades decorrentes de arredondamentos feitos pelo código a velocidade da partícula foi subestimada para um valor muito próximo ao esperado.

CAPÍTULO V

5 CONSIDERAÇÕES FINAIS

O presente capítulo tem por objetivo enunciar as conclusões alcançadas por esta pesquisa, de forma a sintetizar o conhecimento obtido. São feitas, ainda, sugestões para pesquisas futuras a fim de complementar e elucidar pontos deste trabalho.

5.1 CONCLUSÕES

Neste trabalho foi desenvolvido um código computacional para os métodos lattice Boltzmann e elementos discretos com acoplamento feito por meio do método de fronteiras móveis com o objetivo de avaliar a interação entre partículas sólidas e fluidos. O código foi desenvolvido na linguagem C++ devido a sua maior rapidez e aproveita de diversas características dessa linguagem como a organização do código por meio de classes. O nome do código desenvolvido ficou conhecido como Hybrid2D e está na sua versão 0.0.5.

O método lattice Boltzmann implementado utiliza do operador de colisão BGK com um único tempo de relaxação e a discretização do domínio foi feita por meio de células *lattice* conhecidas como D2Q9, comumente utilizadas para simulação bidimensional de fluxo de fluidos. No método dos elementos discretos foi implementado o método desenvolvido por Cundall & Strack no qual se fez da utilização de discos para a simulação bidimensional de meios particulados.

O código do método lattice Boltzmann foi validado por meio de casos de referência amplamente estudados e com solução analítica, sendo estes casos o fluxo Poiseuille e o fluxo ao redor de um cilindro circular, ambos em um canal horizontal. No fluxo Poiseuille foi verificado para seções na entrada, no meio e na saída do canal que o perfil de velocidade parabólico comumente desenvolvido por este tipo de fluxo foi respeitado, apresentando algumas pequenas oscilações devido aos arredondamentos feitos pelo código o que não influenciaram no resultado geral da simulação. Já para o fluxo ao redor de um cilindro circular, percebeu-se a variação do regime de fluxo entre laminar e turbulento e a formação de vórtices por trás do cilindro em função do número de Reynolds da simulação. Em ambos os casos a densidade do fluido apresentou algumas pequenas variações que não influenciaram no

comportamento quase incompressível do fluido. Além disso, foi verificado que o método implementado por não ser um modelo com turbulência apresentou limitações para número de Reynolds maiores que 100.

O código do método dos elementos discretos foi validado e teve seus parâmetros calibrados por meio da análise de casos como queda de uma partícula e análise de colisões entre partículas. No primeiro caso, foi avaliada a conservação de energia de uma partícula em queda livre e foi percebido que os valores determinados pelo Hybrid2D reproduzem perfeitamente a solução analítica do problema. No segundo caso, foi avaliada a conservação de momento linear na colisão entre partículas sem a presença de fatores de amortecimento e foi verificado que o momento linear de uma partícula era todo transferido sem que houvesse a perda de nenhuma parcela. Além disso para esses dois casos foi variado o coeficiente de amortecimento com a finalidade de verificar a sua influência no comportamento geral da simulação e poder calibrar esse parâmetro para as simulações de interação fluido-sólido. Por fim, foi feita uma análise de um disco quicando sobre o outro no Hybrid2D e em outro software muito utilizado na simulação de meios discretos que é o Yade e foi verificado que a energia e posição da partícula nos dois casos tiveram o mesmo resultado.

O método utilizado para fazer o acoplamento dos métodos lattice Boltzmann e elementos discretos foi o método das fronteiras móveis devido a sua maior simplicidade e facilidade em paralelização. A validação deste método foi feita por meio da análise da interação fluido-sólido por meio da avaliação do coeficiente de arrasto da partícula. Os valores de coeficiente de arrasto calculados numericamente apresentam pequenas diferenças em relação aos encontrados por solução analítica devido aos arredondamentos feitos pelo código, porém acredita-se que esta variação não influencie o comportamento geral da simulação.

Por fim, verificou-se que o acoplamento dos métodos lattice Boltzmann e elementos discretos pode ser uma ferramenta poderosa para avaliar a interação entre fluido e meios particulados e o método das fronteiras imersas como um método simples e acurado para determinar as forças de arrasto das partículas, fator extremamente importante para fazer o acoplamento de forma adequada.

5.2 SUGESTÕES PARA PESQUISAS FUTURAS

Incialmente, sugere-se fazer a utilização de um operador de colisão com mais de um tempo de relaxação e avaliar a sua influência no comportamento geral do fluido.

Implementar um modelo lattice Boltzmann turbulento que possibilite a simulação de números de Reynolds maiores que 100 e avaliar o comportamento desse tipo de fluido.

Para a construção de um modelo mais realista, sugere-se que se faça a implementação dos modelos para situações tridimensionais.

Implementar o método Hertz-Mindlin ao método dos elementos discretos para garantir maior confiabilidade nos resultados do meio discreto e avaliar o comportamento do meio submetido a outras solicitações.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDELHAMID, Y. & EL SHAMY, U. (2014). Pore-scale Modeling of Surface Erosion in a Particle Bed. International Journal for Numerical and Analytical Methods in Geomechanics, 38, 142-166.

BHATNAGAR, P. L., GROSS, E. P. & KROOK, M. (1954). A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. Physical Review, 94, 511-525.

BREUER, M., BERNSDORF, J., ZEISER, T. & DURST, F. (2000). Accurate Computations of the Laminar Flow Past a Square Cylinder Based on Two Different Methods: Lattice-Boltzmann and Finite-Volume. International Journal of Heat and Fluid Flow, 21, 186-196.

BRUMBY, P. E., SATO, T., NAGAO, J., TENMA, N. & NARITA, H. (2015). Coupled LBM-DEM Micro-Scale Simulations of Cohesive Particle Erosion Due to Shear Flows. Transport in Porous Media, 109, 43-60.

CAIAZZO, A. (2005). Analysis of Lattice Boltzmann Initialization Routines. Journal of Statistical Physics, 121, 37-48.

CUNDALL, P. A. (1971). A Computer Model for Simulating Progressive Large Scale Movements in Blocky Rock Systems. Proceedings of the Symposium of the International Society for Rock Mechanics, Society for Rock Mechanics (ISRM), France, II-8.

CUNDALL, P. A. & STRACK, O. D. L. (1979). A Discrete Numerical Model for Granular Assemblies. Géotechnique, 29, 47-65.

EL SHAMY, U. & ABDELHAMID, Y. (2014). Modeling Granular Soils Liquefaction Using Coupled Lattice Boltzmann Method and Discrete Element Method. Soil Dynamics and Earthquake Engineering, 67, 119-132.

EL SHAMY, U. & ABDELHAMID, Y. (2014). Pore-scale Modeling of Surface Erosion in a Particle Bed. International Journal for Numerical and Analytical Methods in Geomechanics, 38, 142-166.

FENG, Y. T., HAN, K. & OWEN, D. R. J. (2007). Coupled Lattice Boltzmann Method and Discrete Element Modelling of Particle Transport in Turbulent Fluid Flows: Computational Issues. International Journal for Numerical Methods in Engineering, 72, 1111-1134.

FRISCH U., HASSLACHER, B. & POMEAU, Y. (1986). Lattice-gas automata for the Navier-Stokes equation. Physical Review Letters, 56.14, p. 1505.

GALINDO-TORRES, S. A. (2013). A Coupled Discrete Element Lattice Boltzmann Method for the Simulation of Fluid-Solid Interaction with Particles of General Shapes. Computer Methods in Applied Mathematics and Engineering, 265, 107-119.

GARDNER, M. (2018). Development of a Coupled 3-D DEM-LBM Model for Simulation of Dynamic Rock-Fluid Interactions. Tese de Doutorado, Departamento de Engenharia Civil e Ambiental, Universidade de Berkeley, 95 p.

GARDNER, M. & SITAR, N. (2019). Modelling of Dynamic Rock-Fluid Interaction Using Coupled 3-D Discrete Element and Lattice Boltzmann Methods. Rock Mechanics and Rock Engineering, 1-10.

GUO, Z & ZHAO, T. S. (2002). Lattice Boltzmann Model for Incompressible Flows Through Immersed Boundary Method Scheme. Powder Technology, 315, 486-498.

GUO, Z., ZHENG, C. & SHI, B. (2002). Discrete Lattice Effects on the Forcing Term in the Lattice Boltzmann Method. Physical Review E, 65.

HABTE, M. A. & WU, C. (2017). Particle Sedimentation using Hybrid Lattice-Boltzmann-Immersed Boundary Method Scheme. Powder Technology, 315, 486-498.

HAN, Y & CUNDALL, P. A. (2013). LBM-DEM Modeling of Fluid-Solid Interaction in Porous Media. International Journal for Numerical and Analytical Methods in Geomechanics, 37, 1391-1407.

HARSHANI, H. M. D., GALINDO-TORRES, S. A., SCHEUERMANN, A. & MUHLHAUS, H. B. (2015). Micro-mechanical Analysis on the Onset Erosion in Granular Materials. Philosofical Magazine, 92, 28-30.

HE, X. & LUO, L. (1997). Lattice Boltzmann Model for the Incompressible Navier-Stokes Equation. Journal of Statistical Physics, 88, 927-944.

JONES, B. D. & WILLIAMS, J. R. (2017). Fast Computation of Accurate Sphere-Cube Intersection Volume. Engineering Computations, 34, 1204-1216.

KRÜGER, T., KUSUMAATMAJA, H., KUZMIN, A., SHARDT, O., SILVA, G. & VIGGEN, E. M. (2017). The Lattice Boltzmann Method. Springer, 694 p.

LADD, A. J. C. (1994). Numerical Simulations of Particulate Suspensions via a Discretized Boltzmann Equation. Part 1. Theoretical Foundation. Journal of Fluid Mechanics, 271, 284-309.

LADD, A. J. C. (1994). Numerical Simulations of Particulate Suspensions via a Discretized Boltzmann Equation. Part 2. Numerical Results. Journal of Fluid Mechanics, 271, 311-339.

LIU, W. & WU C-Y. (2019). A Hybrid LBM-DEM Numerical Approach with an Improved Immersed Moving Boundary Method for Complex Particle-Liquid Flows Involving Adhesive Particles. ArXiv e-Prints.

MANSOURI, M., DELENE, J. Y., EL YOUSSOUFI, M. & SERIDI, A. (2009). A 3D-DEM-LBM Approach for the Assessment of the Quick Condition for Sands. Comptes Rendus Mécanique, 337, 675-681.

MEI, R., LUO, L. S., LALLEMAND, P. & D'HUMIÈRES, D. (2006). Consistent Initial Conditions for the Lattice Boltzmann Simulations. Computers and Fluids, 35, 855-862.

MOHAMAD, A. A. (2011) Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes. Springer, Londres, 178 p.

NOBLE, D. R. & TORCZYNSKI, J. R. (1998). A Lattice-Boltzmann Method for Partially Saturated Computational Cells. International Journal of Modern Physics, 8, 1189-1201.

OCAMPO-GÓMEZ, D. A. (2013). Modelagem de Problemas de Fluxo na Escala Granular Usando o Método Lattice Boltzmann. Dissertação de Mestrado, Departamento de Engenharia Civil, Universidade de Brasília, 118 p.

OWEN, D. R. J., LEONARDI, C. R. & FENG, Y. T. (2011). An Efficient Framework for Fluid-Structure Interaction Using the Lattice Boltzmann Method and Immersed Moving Boundaries. International Journal for Numerical Methods in Engineering, 87, 66-95.

QIAN, Y. H., D'HUMIÈRES, D. & LALLEMAND, P. (1992). Lattice BGK Models for the Navier-Stokes Equation. Europhysics Letter, 17, 479-484.

RASMUSSEN, L. L. (2018). Modelos Lattice na Engenharia de Rochas. Tese de Doutorado, Departamento de Engenharia Civil, Universidade de Brasília, 200 p.

RASMUSSEN, L. L., DE FARIAS, M. M. & ASSIS, A. P. (2018). Extended Rigid Body Spring Network Method for the Simulation of Brittle Rocks. Computers and Geotechnics, 99, 31-41.

RETTINGER, C. & RÜDE, U. (2018). A Coupled Lattice Boltzmann Method and Discrete Element Method for Discrete Particle Simulation of Particulate Flows. Computer and Fluids, 172, 706-719.

SATO, M. & KOBAYASHI, T. (2012). A Fundamental Study of the Flow Past a Circular Cylinder Using Abaqus/CFD. Simulia Community Conference, 15 p.

ŠMILAUER, V., CATALANO, E., CHAREYRE, B., DOROFEENKO, S., DURIEZ, J., DYCK, N., ELIÁŠ, J. ER, B., EULITZ, A., GLADKY, A., GUO, N., JAKOB, C., KNEIB, F., KOZICKI, J., MARZOUGUI, D., MAURIN, R., MODENESE, C., SCHOLTÈS, L., SIBILLE, L., STRÁNKÝ, J., SWEIJEN, T., THOENI, K. & YUAN, C. (2015). Yade Documentation, 2nd edition.

SKORDOS, P. A. (1993). Initial and Boundary Conditions for the Lattice Boltzmann Method. Physical Review E, 48, 4823-4842.

SUKOP, M. C. & THORNE, D. T. Jr. (2006). Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers. Spring, Nova York, 173 p.

TANEDA, S. (1956). Experimental Investigations of the Wakes Behind Cylinders and Plates at Low Reynolds Numbers. J. Phys. Soc. Japan 11.

WANG, M., FENG, Y. T., OWEN, D. R. J. & QU, T. M. (2019). A Novel Algorithm of Immersed Moving Boundary Scheme for Fluid-Particle Interaction in DEM-LBM. Computer Methods in Applied Mechanics and Engineering, 346, 109-125.

WHITE, F. (1991). Viscous Fluid Flow. McGraw-Hill, New York, 66.

YOUSSOUFI, M. S., MANSOURI, M. & NICOT, F. (2016). Numerical Simulation of the Quicksand Phenomenon by a 3D Couple Discrete Element – Lattice Boltzmann Hydromechanical Model. International Journal for Numerical and Analytical Methods in Geomechanics.

ZAPALOWSKI, V. (2011). Análise Quantitativa e Comparativa de Linguagens de Programação. Trabalho de Graduação. Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, 45 p.

ZHANG, P., GALINDO-TORRES, S. A., TANG, H., JIN, G., SCHEUERMANN, A. & LI, L. (2017). Na Efficient Discrete Element Lattice Boltzmann Model for Simulation of Particle-Fluid, Particle-Particle Interactions. Computers and Fluids, 147, 63-71.

ZOU, Q. & HE, X. (1997). On Pressure and Velocity Boundary Conditions for the Lattice Boltzmann BGK Model. Physics of Fluids, 9, 1591-1598.

ZULUAGA, R. A. G. (2016). Relação Entre Características Microestruturais e o Comportamento Macroscópico de Solos Granulares. Tese de Doutorado, Departamento de Engenharia Civil, Universidade de Brasília, 171 p.

APÊNDICE A: CABEÇALHOS DO PROGRAMA HYBRID2D

Neste apêndice é apresentado a listagem completa do código-fonte de todos os cabeçalhos do programa Hybrid2D.

LATTICE.H

```
#ifndef LATTICE_H
#define LATTICE_H

//Standard Libray
#include <vector>

//Hybrid Library
#include "Math.h"

class Lattice {
public:
	//Constructor:
	Lattice(int _ID, double _dx, double _dt, double _tau, Vector2r
_dim, Vector2r _cellPos) {
		ID			= _ID;
		dx			= _dx;
		dt			= _dt;
		tau			= _tau;
		latticeSpeed	= _dx / _dt;
		dim			= _dim;
		cellPos		= _cellPos;

		//Set neighbor node
		for (int k = 0; k < Q; k++) {
			aux = cellPos + discreteVelocity[k];

			if (aux[0] == -1)	aux[0] = dim[0] - 1;
			if (aux[1] == -1)	aux[1] = dim[1] - 1;

			if (aux[0] == (int)dim[0]) aux[0] = 0;
			if (aux[1] == (int)dim[1]) aux[1] = 0;

			nCell.push_back(aux[0] + dim[0] * aux[1]);
		}
	}

	//Methods:
	double setEqFun(double _rho, Vector2r _vel, int k);
	double setSourceTerm(double& _tau, double _dt, int k);
	double getDensityAndVelocity(Vector2r& _vel);
	void   updateMacro();
```

```
    friend class LBM;
    friend class IMB;

    //Cell variables:
    int    ID;
    double dx;
    double dt;
    double tau;
    double latticeSpeed;
    double rhoBC;
    Vector2r velBC;
    Vector2r aux;
    Vector2r dim;
    Vector2r cellPos;

    Vector2r sourceForce           = Vector2r::Zero();
    Vector2r vel                   = Vector2r::Zero();
    double   solidFraction         = 0.0;
    double   previousSolidFraction = 0.0;
    double   rho                   = 0.0;
    int      particleFluidID       = 0;
    int      node                  = 0;
    int      isSolid               = 1;
    int      isFluid               = 0;

private:
    //D2Q9 Variables:
    const int Q = 9;
    const std::vector<int>      opNode           = { 0, 3, 4, 1, 2,
7, 8, 5, 6 };
    const std::vector<double>   weight           = { 4.0 / 9.0, 1.0
/ 9.0, 1.0 / 9.0, 1.0 / 9.0, 1.0 / 9.0, 1.0 / 36.0, 1.0 / 36.0, 1.0
/ 36.0, 1.0 / 36.0 };
    const std::vector<Vector2r> discreteVelocity = { {0,0}, {1,0},
{0,1}, {-1,0}, {0,-1}, {1,1}, {-1,1}, {-1,-1}, {1,-1} };

    std::vector<double> f     = { 0,0,0,0,0,0,0,0,0 };
    std::vector<double> fTemp = { 0,0,0,0,0,0,0,0,0 };
    std::vector<double> omega = { 0,0,0,0,0,0,0,0,0 };
    std::vector<double> nCell;
};
    #endif // !LATTICE_H
```

LBM.H

```
#ifndef LBM_H
#define LBM_H

//Standard Library
#include <memory>
#include <vector>

//Hybrid Library
#include "Lattice.h"
#include "Math.h"

class LBM {
public:

	//Getters:
	int getCell(int i, int j);

	//Boundary conditions:
	void setVelBC(int i, int j, Vector2r _vel);
	void setDenBC(int i, int j, double _rho);
	void setZouBC();
	void setBounceBack();

	//LBM Engine:
	void initializeCells();
	void setInitCond(double _rhoInit, Vector2r _vel);
	void resetSolidFraction();
	void applyBodyForce();
	void collision();
	void collisionNT();
	void stream();

	bool setVelNorth = false;
	bool setVelEast  = false;
	bool setVelWest  = false;
	bool setVelSouth = false;

	bool setDenNorth = false;
	bool setDenEast  = false;
	bool setDenWest  = false;
	bool setDenSouth = false;

	std::vector<std::shared_ptr<Lattice>> cells;

	Vector2r gravity = { 0.0, -9.81 };
	Vector2r domainSize = Vector2r::Zero();
	double   dx = 1.0;
```

```
	double	dtLBM = 1.0;
	double	tau = 1.0;
	double	latticeSpeed = 1.0;
	double	kinViscosity = 1.0;
};
#endif // !LBM_H
```

BODY.H

```
#ifndef BODY_H
#define BODY_H

//Standard Library
#define _USE_MATH_DEFINES
#include <math.h>
#include <memory>
#include <vector>

//Hybrid Library
#include "Math.h"

class Interaction;

class Body {
public:

	Body(double _mass, double _radius, Vector2r _pos, Vector2r
_vel, int _id, bool _fixed) {
		mass		= _mass;
		radius		= _radius;
		inertiaMoment = _mass * _radius * _radius * 0.5;
		pos		= _pos;
		vel		= _vel;
		id		= _id;
		if (_fixed) { blockedDOFs = Vector2r::Zero();
blockedMDOFs = 0; }
	}

	bool checkInteraction(int _bodyId);
	void calculateEnergy();
	bool fluidInteraction(Vector2r _cellPos, double _dx);

	//Body variables:
	double radius;
	double mass;
	double inertiaMoment;
	int	id;
```

```
	//Smart pointers:
	std::vector<std::weak_ptr<Interaction>> inter;
	std::vector<int> fluidSolidInteraction;

	Vector2r pos          = Vector2r::Zero();
	Vector2r vel          = Vector2r::Zero();
	Vector2r force        = Vector2r::Zero();
	Vector2r forceLBM     = Vector2r::Zero();
	Vector2r torqueLBM    = Vector2r::Zero();
	Vector2r blockedDOFs  = { 1,1 };
	double   rot          = 0.0;
	double   rotVel       = 0.0;
	double   moment       = 0.0;
	double   blockedMDOFs = 1;
	double   kinEnergy    = 0.0;
	double   potEnergy    = 0.0;

	~Body() {};
};

#endif //BODY_H
```

INTERACTION.H

```
#ifndef INTERACTION_H
#define INTERACTION_H

//Hybrid Libary
#define _USE_MATH_DEFINES
#include <math.h>
#include <memory>

//Hybrid Libary
#include "Math.h"

class Body;

class Interaction {
public:
	Interaction(std::weak_ptr<Body> _body1, std::weak_ptr<Body>
_body2) : body1(_body1), body2(_body2) {};

	//Methods:
	bool checkContact();
	void calculateUnitVectorandContact();
```

```
        void calculateForceAndShearIncrements(double _dt, double _kn,
double _ks);
        void applyFrictionLaw(double _phi);

        //Smart pointers:
        std::weak_ptr<Body> body1;
        std::weak_ptr<Body> body2;

        //Variables:
        Vector2r unitNormal  = Vector2r::Zero();
        Vector2r unitShear   = Vector2r::Zero();
        Vector2r contact     = Vector2r::Zero();
        double   normalForce = 0.0;
        double   shearForce  = 0.0;
};

#endif //INTERACTION_H
```

DEM.H

```
#ifndef DEM_H
#define DEM_H

//Standard Library
#include <vector>
#include <memory>

//Hybrid Library
#include "Math.h"
#include "Interaction.h"
#include "Body.h"

class DEM {
public:

        //Engine:
        void     calculateParticleTimeStep();
        void     contactVerification();
        void     forceCalculation();
        void     updateVelPos();
        void     updateContact();
        Vector2r applyBorderForce(std::shared_ptr<Body> _body);
        void     calculateEnergy();

        //Smart pointers to classes:
        std::vector<std::shared_ptr<Body>> bodies;
        std::vector<std::shared_ptr<Interaction>> interactions;
```

```
	//Vector:
	std::vector<double> kinEnergy;
	std::vector<double> potEnergy;

	//Model variables:
	Vector2r domainReference = Vector2r::Zero();
	Vector2r domainSize      = Vector2r::Zero();
	Vector2r gravity         = { 0.0, -9.81 };
	double   normalStiffness = 1e6;
	double   shearStiffness  = 0.5e6;
	double   localDamping    = 0.8;
	double   frictionAngle   = 30;
	double   borderStifness  = 1e6;
	double   dtDEM           = 0.0;
	double factorOfSafety = 0.1;
	int    nIter          = 0;
};

#endif //DEM_H
```

IMB.H

```
#ifndef IMB_H
#define IMB_H

//Hybrid Libary
#include "Math.h"
#include "LBM.h"
#include "DEM.h"

class IMB {
public:
	//Constructor:
	IMB() : eLBM(), eDEM() {};

	//Engine
	void   defineLinkedCells();
	double calculateSolidFraction(Vector2r& _particlePos, Vector2r&
_cellPos, double _particleRadius, double _dx);
	void   calculateForceAndTorque();

	LBM eLBM;  //LBM engine
	DEM eDEM;  //DEM engine

};
```

```
#endif // !SCENE_H
```

SCENE.H

```
#ifndef SCENE_H
#define SCENE_H

//Standard Libary

//Hybrid Libary
#include "IMB.h"
#include "Math.h"
#include "Timer.h"

class Scene {
public:

	//Scene constructor:
	Scene() : eIMB() {};

	//Classes
	IMB eIMB;

	//Scenario prepartion:
	void addCircle(double _mass, double _radius, Vector2r _pos,
Vector2r _vel, bool _fixed);
	void setDomain();
	void prepareScenario();

	//Solver:
	void LBMEngine();
	void DEMEngine();

	//Method to acess the class Scene
	Scene& getScene();

	//Geometry parameters:
	Vector2r domainSize = Vector2r::Zero();
	bool topIsSolid     = false;
	bool botIsSolid     = false;
	bool leftIsSolid    = false;
	bool rightIsSolid   = false;
	bool bodiesAreSolid = false;

	//Fluid parameters:
	double   latticeSpacing = 1.0;
	double   kinViscosity   = 1e-6;
```

```
	double   relaxationTime = 1.0;
	double   rhoInit        = 1.0;
	Vector2r velInit        = { 0.08, 0.0 };

	//Particle parameters:
	double frictionAngle   = 30;
	double localDamping    = 0.0;
	double factorOfSafety  = 0.3;
	double normalStiffness = 1e6;
	double shearStiffness  = 0.5e6;

	double Time           = 0.0;
	int    simDuration    = 10000;
	int    subCycleNumber = 0;
};

#endif // !SCENE_H
```

OUTPUT.H

```
#ifndef OUTPUT_H
#define OUTPUT_H

//Standard Library
#include <iomanip>
#include <fstream>
#include <string>
#include <memory>

//Hybrid Library
#include "Scene.h"
#include "Lattice.h"
#include "Body.h"

class Output {
public:
	void displaySimulationInfo();
	void fluidVTK(std::string _fileName);
	void solidVTK(std::string _fileName);
	void fluidVelocityProfile(std::string _fileName, int _cellId);
	void particleEnergy(std::string _fileName, int _bodyId);

	int fluidVtkCounter = 0;
	int solidVtkCounter = 0;
};

#endif // !OUTPUT_H
```

MATH.H

```
#ifndef MATH_H
#define MATH_H

//Standard Libary
#include <iostream>
#include <string>
#include <Eigen/Core>

using real = double;

//Templates:
template<typename Scalar> using Vector2 = Eigen::Matrix<Scalar, 2,
1, Eigen::DontAlign>;
using Vector2r = Vector2<real>;
using Vector2i = Vector2<int>;

//Defines:
#define ASSERT_FLAG 1
#if ASSERT_FLAG == 1
#define ASSERT(x) \
                 if (! (x)) \
                { \
                   std::cout << "ERROR!! Assert " << #x << "
failed\n"; \
                   std::cout << " on line " << __LINE__  << "\n"; \
                   std::cout << " in file " << __FILE__ << "\n";  \
                   std::string pause; \
                   std::cin >> pause; \
                }
#else
#define ASSERT(x)
#endif


#endif //MATH_H
```

TIMER.H

```
#ifndef TIMER_H
#define TIMER_H

//Standard Libary
```

```
#include <chrono>
#include <iostream>

class Timer {
public:
	Timer() {
		startTimePoint =
std::chrono::high_resolution_clock::now();
	}

	~Timer() {
		Stop();
	}

	void Stop() {
		auto endTimePoint =
std::chrono::high_resolution_clock::now();
		auto start =
std::chrono::time_point_cast<std::chrono::minutes>(startTimePoint).t
ime_since_epoch().count();
		auto end =
std::chrono::time_point_cast<std::chrono::minutes>(endTimePoint).tim
e_since_epoch().count();
		auto duration = end - start;
		std::cout << "----------------------------------------
\n";
		std::cout << "Simulation Time: " << duration << "
minutes\n";
	}
private:
	std::chrono::time_point<std::chrono::high_resolution_clock>
startTimePoint;
};

#endif //TIMER_H
```

APÊNDICE B: SCRIPTS DOS CENÁRIOS DE SIMULAÇÃO DE REFERÊNCIA

Neste apêndice é apresentado os códigos utilizados para a geração dos cenários de simulação utilizados para validar o programa Hybrid2D.

POISEUILLE.CPP

```
#include "Scene.h"
#include "Output.h"
#include "Timer.h"

int main() {

	Timer Time;
	Scene& S = S.getScene();
	Output Out;

	//Max velocity
	double   uMax           = 0.1;

	//Geometry
	S.domainSize  = { 500, 100 };
	S.simDuration = 15000.0;
	S.topIsSolid  = true;
	S.botIsSolid  = true;

	//Zou&He Boundary Conditions
	S.eIMB.eLBM.setVelWest = true;
	S.eIMB.eLBM.setDenEast = true;

	//Fluid Properties:
	S.kinViscosity = 1 / 6;
	S.relaxationTime = 1.0;
	S.velInit        = { 0.08, 0.0 };
	S.rhoInit        = 1.0;

	//Prepare Scenario
	S.prepareScenario();

	//Apply vel and density boundary conditions (intlet and outlet
cells)
	for (int j = 0; j < domainSize[1]; ++j) {
		double L = domainSize[1] - 2;
		double yp = j - 1.5;
		Vector2r Vel = Vector2r::Zero();
		Vel[0] = uMax * 4 / (L * L) * (L * yp - yp * yp);
		Vel[1] = 0.0;
		S.eIMB.eLBM.setVelBC(0, j, Vel);
```

```
            S.eIMB.eLBM.setDenBC(domainSize[0] - 1, j, 1.0);
      }

      int ignore = system("mkdir VTK_Fluid");
      int i = 0;
      while (S.Time < S.simDuration) {
            if (i % 1000 == 0) {
                  Out.displaySimulationInfo();

            }

            S.LBMEngine();
            if (i % 100 == 0) {
                  for (int j = 0; j < domainSize[1]-1; ++j) {
                        Out.fluidVelocityProfile("inlet.csv",
      S.eIMB.eLBM.getCell(10, j));
                        Out.fluidVelocityProfile("mid.csv",
S.eIMB.eLBM.getCell(249.5, j));
                        Out.fluidVelocityProfile("Outlet.csv",
S.eIMB.eLBM.getCell(domainSize[0]-10, j));

                  }
                  Out.fluidVTK("LBM");
            }

            S.Time += S.eIMB.eLBM.dtLBM;
            ++i;
      }

      return 0;
}
```

CYLINDER.CPP

```
#include "Scene.h"
#include "Output.h"
#include "Timer.h"

int main() {

      Timer Time;
      Scene& S = S.getScene();
      Output Out;

      //General Information
      Vector2r domainSize     = { 500, 100 };
```

```
	Vector2r cylinderCoord  = { domainSize[1] / 2., domainSize[1] /
2. };
	double   particleRadius = domainSize[1] / 10;
	double   uMax           = 0.1;
	double   Re             = 100;

	//Geometry
	S.domainSize     = domainSize;
	S.simDuration    = 15000.0;
	S.topIsSolid     = true;
	S.botIsSolid     = true;
	S.bodiesAreSolid = true;

	//Zou&He Boundary Conditions
	S.eIMB.eLBM.setVelWest = true;
	S.eIMB.eLBM.setDenEast = true;

	//Bodies:
	S.addCircle(1, particleRadius, cylinderCoord, { 0.0, 0.0 },
false);

	//Fluid Properties:
	S.kinViscosity   = uMax * (2 * particleRadius) / Re;
	S.relaxationTime = 3.0 * S.kinViscosity + 0.5;
	S.velInit        = { 0.08, 0.0 };
	S.rhoInit        = 1.0;

	//Prepare Scenario
	S.prepareScenario();

	//Apply vel and density boundary conditions (inlet and outlet
cells)
	for (int j = 0; j < domainSize[1]; ++j) {
		double L = domainSize[1] - 2;
		double yp = j - 1.5;
		Vector2r Vel = Vector2r::Zero();
		Vel[0] = uMax * 4 / (L * L) * (L * yp - yp * yp);
		Vel[1] = 0.0;
		S.eIMB.eLBM.setVelBC(0, j, Vel);
		S.eIMB.eLBM.setDenBC(domainSize[0] - 1, j, 1.0);
	}

	int ignore = system("mkdir VTK_Fluid");
	int i = 0;
	while (S.Time < S.simDuration) {
		if (i % 1000 == 0) {
			Out.displaySimulationInfo();

		}
```

```
            S.LBMEngine();
            if (i % 100 == 0) {
                  for (int j = 0; j < domainSize[1]-1; ++j) {
                        Out.fluidVelocityProfile("inlet.csv",
S.eIMB.eLBM.getCell(10, j));
                        Out.fluidVelocityProfile("mid.csv",
S.eIMB.eLBM.getCell(70, j));
                        Out.fluidVelocityProfile("Outlet.csv",
S.eIMB.eLBM.getCell(domainSize[0]-10, j));
                        Out.fluidVelocityProfile("Cilindro.csv",
S.eIMB.eLBM.getCell(cylinderCoord[0], j));
                  }
                  Out.fluidVTK("LBM");
            }

            S.Time += S.eIMB.eLBM.dtLBM;
            ++i;
      }

      return 0;
}
```

DROP.CPP

```
#include "Scene.h"
#include "Output.h"

int main() {
      Timer Time;
      Scene& S = S.getScene();
      Output Out;

      //Geometry
      S.domainSize = { 10, 10 };
      S.addCircle(1.0, 0.5, { 5, 5 }, { 0.0, 0.0 }, false);

      //Solid Properties:
      S.frictionAngle   = 30;
      S.localDamping    = 0.7;
      S.factorOfSafety  = 0.1;
      S.normalStiffness = 1.0e6;
      S.shearStiffness  = 0.5e6;
      S.simDuration     = 100000;

      //Solver
      S.prepareScenario();
```

```
	int ignore = system("mkdir VTK_Solid");
	for (int i = 0; i != S.simDuration; ++i) {
		if (i % 1000 == 0) {
			Out.displaySimulationInfo();
			Out.solidVTK("DEM");
			Out.particleEnergy("Body1.csv", 0);
		}
		S.DEMEngine();
	}

	return 0;
}
```

BOUNCINGBALL.CPP

```
#include "Scene.h"
#include "Output.h"

int main() {
	Scene& S = S.getScene();
	Output Out;

	//Geometry
	S.domainSize = { 10, 10 };
	S.addCircle(1.0, 0.5, { 5, 5.5 }, { 0.0, 0.0 }, false);
	S.addCircle(1.0, 0.5, { 5, 0.5 }, { 0.0, 0.0 }, true);

	//Solid Properties:
	S.frictionAngle = 30;
	S.localDamping = 0.7;
	S.factorOfSafety = 0.1;
	S.normalStiffness = 1.0e6;
	S.shearStiffness = 0.5e6;
	S.simDuration = 100000;

	//Solver
	S.prepareScenario();
	int ignore = system("mkdir VTK_Solid");
	for (int i = 0; i != S.simDuration; ++i) {
		if (i % 1000 == 0) {
			Out.displaySimulationInfo();
			Out.solidVTK("DEM");
			Out.particleEnergy("Body1.csv", 0);
			Out.particleEnergy("Body2.csv", 1);
		}
		S.DEMEngine();
	}
```

```
	return 0;
}
```

COLLISION.CPP

```
#include "Scene.h"
#include "Output.h"

int main() {
	Scene& S = S.getScene();
	Output Out;

	//Geometry
	S.domainSize = { 10, 10 };
	S.addCircle(1.0, 0.5, { 2, 0.5 }, { 4.0, 0.0 }, false);
	S.addCircle(1.0, 0.5, { 7, 0.5 }, { 0.0, 0.0 }, false);

	//Solid Properties:
	S.frictionAngle = 30;
	S.localDamping = 0.7;
	S.factorOfSafety = 0.1;
	S.normalStiffness = 1.0e6;
	S.shearStiffness = 0.5e6;
	S.simDuration = 100000;

	//Solver
	S.prepareScenario();
	int ignore = system("mkdir VTK_Solid");
	for (int i = 0; i != S.simDuration; ++i) {
		if (i % 1000 == 0) {
			Out.displaySimulationInfo();
			Out.solidVTK("DEM");
			Out.particleEnergy("Body1.csv", 0);
			Out.particleEnergy("Body2.csv", 1);
		}
		S.DEMEngine();
	}

	return 0;
}
```

APENDICE C: SCRIPT DA SIMULAÇÃO DA INTERAÇÃO FLUIDO-SÓLIDO

Neste apêndice é apresentado o código utilizado para validar o acoplamento dos métodos lattice Boltzmann e elementos discretos por meio do método das fronteiras móveis e avaliar a interação fluido-sólido.

EROSION.CPP

```
#include "Scene.h"
#include "Output.h"

int main() {

	Timer Time;
	Scene& S = S.getScene();
	Output Out;

	//General Information
	Vector2r  domainSize = { 500, 100 };
	double particleRadius = domainSize[1] / 20 + 1;
	Vector2r  cylinderCoord = { particleRadius, particleRadius };
	double uMax = 0.1;
	double Re = 100;

	S.domainSize = domainSize;

	//Bodies:
	S.addCircle(1, particleRadius, cylinderCoord, Vector2r::Zero(),
true);

	//Fluid Properties:
	S.kinViscosity = uMax * (2 * particleRadius) / Re;
	S.relaxationTime = 3.0 * S.kinViscosity + 0.5;
	S.velInit = { 0.1, 0.0 };
	S.rhoInit = 1.0;

	//Particle Properties:
	S.frictionAngle = 30;
	S.localDamping = 0.1;
	S.factorOfSafety = 0.1;
	S.normalStiffness = 1.0e6;
	S.shearStiffness = 0.5e6;
	S.simDuration = 10000;

	//Prepare Scenario
	S.prepareScenario();
```

```
	int ignore = system("mkdir VTK_Fluid");
	    ignore = system("mkdir VTK_Solid");

	double tlbm = 0.0;
	int i = 0;
	while (S.Time < S.simDuration) {
		if (i % 10000 == 0)   Out.displaySimulationInfo();
		S.eIMB.eLBM.resetSolidFraction();
		S.eIMB.defineLinkedCells();
		S.eIMB.calculateForceAndTorque();
		S.DEMEngine();
		if (S.Time >= tlbm) {
			S.eIMB.eLBM.setZouBC();
			S.eIMB.eLBM.collisionNT();
			S.eIMB.eLBM.setBounceBack();
			S.eIMB.eLBM.stream();
			tlbm += S.eIMB.eLBM.dtLBM;
		}
		if (i % 10000 == 0) {
			Out.fluidVTK("LBM");
			Out.solidVTK("DEM");
			Out.particleInfo("bodyInfo.csv", 0);
		}
		S.Time += S.eIMB.eDEM.dtDEM;
		++i;
	}

	return 0;
}
```