

UNIVERSIDADE FEDERAL DO PARANÁ

MARCOS VINICIUS ANDRIOLO

MODELAGEM COMPUTACIONAL UTILIZANDO AUTÔMATOS CELULARES PARA
O CÁLCULO DO TOTAL DE GÁS DISSOLVIDO

CURITIBA - PR

2025

MARCOS VINICIUS ANDRIOLO

MODELAGEM COMPUTACIONAL UTILIZANDO AUTÔMATOS CELULARES PARA
O CÁLCULO DO TOTAL DE GÁS DISSOLVIDO

Tese apresentada ao curso de Pós-Graduação em Engenharia de Recursos Hídricos e Ambiental, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Doutor em Engenharia de Recursos Hídricos e Ambiental.

Orientador: Prof. Dr. Eloy Kaviski

CURITIBA - PR

2025

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Andriolo, Marcos Vinicius

Modelagem computacional utilizando autômatos celulares para o cálculo do total de gás dissolvido. / Marcos Vinicius Andriolo. – Curitiba, 2025.

1 recurso on-line : PDF.

Tese (Doutorado) – Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Ambiental.

Orientador: Prof. Dr. Eloy Kaviski

1. Autômato Celular. 2. Vertedouro. 3. Peixes – Doenças (Doença da bolha). I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Ambiental. III. Kaviski, Eloy. IV. Título.

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585



MINISTÉRIO DA EDUCAÇÃO
SETOR DE TECNOLOGIA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO ENGENHARIA DE
RECURSOS HÍDRICOS E AMBIENTAL - 40001016021P0

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação ENGENHARIA DE RECURSOS HÍDRICOS E AMBIENTAL da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **MARCOS VINICIUS ANDRIOLO**, intitulada: "**Modelagem Computacional Utilizando Autômatos Celulares para o Cálculo do Total de Gás Dissolvido**", sob orientação do Prof. Dr. ELOY KAVISKI, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 15 de Dezembro de 2025.

Assinatura Eletrônica

16/12/2025 23:29:16.0

ELOY KAVISKI

Presidente da Banca Examinadora

Assinatura Eletrônica

17/12/2025 13:58:23.0

VANDERLEI GALINA

Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO
PARANÁ)

Assinatura Eletrônica

19/12/2025 14:11:57.0

JORGE ANTONIO SILVA CENTENO

Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica

17/12/2025 09:47:33.0

WELLINGTON MAZER

Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO
PARANÁ)

Assinatura Eletrônica

17/12/2025 23:06:08.0

JOSÉ JUNJI OTA

Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Centro Politécnico, Bloco V - CURITIBA - Paraná - Brasil

CEP 81531-990 - Tel: (41) 3361-3210 - E-mail: ppgerha@ufpr.br

Documento assinado eletronicamente de acordo com o disposto na legislação federal Decreto 8539 de 08 de outubro de 2015.

Gerado e autenticado pelo SIGA-UFPR, com a seguinte identificação única: 504289

Para autenticar este documento/assinatura, acesse <https://siga.ufpr.br/siga/visitante/autenticacaoassinaturas.jsp>
e insira o código 504289

Julia, Helena e Andréa.

AGRADECIMENTOS

Aos meus pais Wânia e José, meu eterno agradecimento.

Ao meu orientador, Prof. Dr. Eloy Kaviski minha gratidão por todo o apoio, incentivo e orientação.

Aos professores da banca, por terem aceitado este convite e pelas valiosas contribuições, Prof. Dr. José Junji Ota, Prof. Dr. Jorge Antonio Silva Centeno, Prof. Dr. Wellington Mazer e Prof. Dr. Vanderlei Galina.

À Universidade Federal do Paraná (UFPR), ao Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Ambiental (PPGERHA), pela oportunidade e aos professores do PPGERHA que tanto contribuíram para minha formação acadêmica e profissional.

Meu agradecimento a equipe que participou do diagnóstico, projeto, construção e P&D sobre TDG, Anderson, André, Carolina, Cássia, Fernando, Ingrid, Rafael, Soraia, Marcela, Carlos, Marcelo, Paulo Cabral, Paulo Henrique, e aos professores José Ota, Eloy e Fernando.

A Copel GeT por acreditar nos estudos desenvolvidos, na solução proposta e na construção do defletor.

Esta tese apresenta partes dos resultados obtidos durante a execução do projeto de PDI PD-06491-0541/2019 de título "Metodologia para modelagem computacional de TDG na água em fluxos efluentes de vertedouros" executado pelo Instituto Lactec e patrocinado pela Copel Geração e Transmissão S.A. no âmbito do Programa de Pesquisa, Desenvolvimento e Inovação do Setor Elétrico Brasileiro, regulamentado pela Agência Nacional de Energia Elétrica (ANEEL).

RESUMO

A Saturação Gasosa, também conhecida como Total de Gás Dissolvido (TDG - *Total Dissolved Gas*), pode ocorrer em barragens, cachoeiras e corredeiras. Em usinas hidroelétricas, eventos de vertimento elevam significativamente os níveis de TDG, podendo provocar a doença da bolha (GBD – *Gas Bubble Disease*) na ictiofauna. Uma forma de reduzir a supersaturação de TDG em vertedouros é a instalação de defletores, que modificam a hidrodinâmica do escoamento, evitando que a alta velocidade do fluxo “empurre” “bolhas” de gás para regiões profundas da bacia de dissipação, onde a alta pressão favorece sua dissolução e, conseqüentemente, o aumento do TDG. Idealmente, o projeto desses defletores deve ser fundamentado em ensaios em modelo físico, complementados por estudos em modelos matemáticos ou computacionais. Entretanto, simulações numéricas demandam um grande esforço computacional. O objetivo desta tese foi desenvolver um modelo computacional baseado em autômatos celulares para o cálculo do TDG, capaz de estimar o valor máximo e o comportamento do TDG ao longo do vertedouro e da bacia de dissipação. O modelo de autômatos celulares, apresentado e implementado nesta tese, apresentou valores máximos de TDG ao longo do vertedouro e da bacia de dissipação, aderentes a outros estudos, sendo um modelo promissor para estimar esse parâmetro de qualidade da água. Devido ao menor esforço computacional, o modelo proposto baseado em autômato celular, pode ser aplicado nas etapas de Estudo de Viabilidade ou Projeto Básico de aproveitamentos hidroelétricos. Nesses estágios iniciais, a ferramenta permite estudar os valores máximos de TDG e propor, de forma simplificada, a geometria preliminar de um defletor em vertedouros, contribuindo para reduzir, desde o início do projeto, os riscos à ictiofauna.

Palavras-chave: Total de Gás Dissolvido; Defletor em Vertedouro; Doença da Bolha; Autômato Celular.

ABSTRACT

Gas supersaturation, also known as Total Dissolved Gas (TDG), may occur in dams, waterfalls, and rapids. In hydroelectric plants, spill events can significantly increase TDG levels, potentially causing gas bubble disease (GBD) in fish. One approach to reducing TDG supersaturation in spillways is the installation of deflectors, which modify the flow hydrodynamics and prevent the high-velocity jet from driving gas bubbles into deeper regions of the stilling basin. At greater depths, the elevated hydrostatic pressure promotes bubble dissolution, thereby increasing TDG. Ideally, the design of these deflectors should be supported by physical model tests, complemented by mathematical or computational modeling. However, numerical simulations require substantial computational resources. The objective of this thesis was to develop a cellular automata model for calculating TDG, capable of estimating both the maximum TDG value and its spatial distribution along the spillway and stilling basin. The cellular automata model presented and implemented in this thesis showed maximum TDG values along the spillway and stilling basin, with good agreement with other studies, demonstrating its potential as a reliable tool for estimating this water quality parameter. Due to its lower computational cost, the proposed cellular automata model can be applied during the Initial Studies or Basic Design phases of hydropower projects. At these early stages, the tool enables the assessment of maximum TDG levels and the simplified proposal of preliminary deflector geometries, thereby helping to mitigate risks to fish fauna during preliminary projects phases.

Keywords: Total Dissolved Gas; Spillway Deflector; Gas Bubble Disease; Cellular Automata.

LISTA DE FIGURAS

Figura 1 – Superfície do escoamento no vertedouro com e sem defletor.	24
Figura 2 – Diversidade das espécies de peixes e a localização de barragens na bacia Amazônica, Congo e Mekong.	27
Figura 3 – Relação entre % de TDG e profundidade.	29
Figura 4 – Indicação da doença da bolha em peixes a jusante de vertedouros.	32
Figura 5 – Tipo de escoamento.	35
Figura 6 – Regime de escoamento observado em modelo reduzido.	36
Figura 7 – Variáveis de interesse utilizadas na Curva de performance da UHE Colíder.	36
Figura 8 – Curva de performance para diferentes regimes de escoamento da UHE Colíder.	37
Figura 9 – Seção típica do vertedouro com defletor da UHE Colíder.	38
Figura 10 – Concentração de TDG na bacia de dissipação antes e após a construção do defletor.	38
Figura 11 – Concentração de TDG na bacia de dissipação após a construção do defletor.	39
Figura 12 – Local de instalação da sonda de TDG no vertedouro da UHE Colíder.	40
Figura 13 – Sonda HydroLab MS5 utilizada para medição de TDG em campo.	40
Figura 14 – Entrada de “bolhas” de ar em um ressalto hidráulico.	41
Figura 15 – Comparação entre a relação ar-água em três testes.	43
Figura 16 – Resumo das principais considerações computacionais da simulação de TDG.	46
Figura 17 – Três mecanismos de incorporação de ar.	48
Figura 18 – Sucessão de eventos de fragmentação das “bolhas”.	49
Figura 19 – Mecanismo de incorporação de ar proposto para impacto de jato em alta velocidade.	50
Figura 20 – Processos físicos que conduzem ao aprisionamento de ar no escoamento.	52
Figura 21 – Mecanismos de Coalescência.	55
Figura 22 – Modelagem hidrodinâmica.	58

Figura 23 – Modelagem hidrodinâmica - Classificação do escoamento.....	59
Figura 24 – Classificação dos tipos de simulação da turbulência.	60
Figura 25 – Comparação entre simulações DNS e LES.	61
Figura 26 – Comparação de simulações entre RANS, SRS (LES) e DNS.	64
Figura 27 – Comparação da superfície livre média.	65
Figura 28 - Escoamento multifásico: escoamento separado (estratificado) e escoamento disperso.	65
Figura 29 - Efeito de vórtice turbulento (linha contínua) em uma partícula (linha tracejada) para diferentes números de Stokes.	67
Figura 30 - Abordagens do escoamento multifásico: a) Euleriano-Lagrangiano e b) Euleriano- Euleriano.	68
Figura 31 – Modelo VOF (<i>Volume of Fluid</i>): a) interface real entre fluidos, b) fração de volume calculada pelo VOF e c) interpolação linear da superfície.	69
Figura 32 – Configuração da solução de escoamento utilizando a simulação DES. .	72
Figura 33 – Análise de sensibilidade do TDG.	74
Figura 34 – Perfil médio de concentração de ar.	75
Figura 35 – Escoamento a jusante de vertedouros.	78
Figura 36 – Análise de sensibilidade do % de TDG adotando $D_b = 0,8\text{mm}$ e α variando entre 0,01 a 0,03.	80
Figura 37 – Análise de sensibilidade do % de TDG adotando $\alpha = 0,03$ e D_b variando entre 0,1mm a 2mm.	81
Figura 38 – Regra 150.	87
Figura 39 – Evolução da Regra 150.	88
Figura 40 – As quatro classes do AC.	91
Figura 41 – Exemplo de regras da Classe I-Fixa do AC.	91
Figura 42 – Exemplo de regras da Classe II-Periódica do AC.	91
Figura 43 – Exemplo de regras da Classe IV-Complexo do AC.	92
Figura 44 – Exemplo de regras da Classe III-Caótico do AC.	92
Figura 45 – Exemplos de células com seu estado atualizado.	95
Figura 46 – Direções de movimento do reticulado D2Q5, com 4 velocidades não nulas.	96
Figura 47 - Exemplo de configuração de partículas no modelo HPP.	97
Figura 48 - A regra HPP: (a) Propagação, (b) e (c) Colisão.	98
Figura 49 – Modelo de reticulado D2Q7, com 6 velocidades não nulas.	99

Figura 50 – Divisão do mosaico em blocos de 3 x 3 células.	103
Figura 51 – Simulação da difusão e redistribuição dos elementos nas células.	104
Figura 52 – Mudanças que ocorrem com o modelo baseado em bloco CA.	104
Figura 53 – Arranjo Geral da UHE Colíder.	113
Figura 54 – Barragem da UHE Colíder.	115
Figura 55 – Vertedouro da UHE Colíder.	117
Figura 56 – Casa de força.	119
Figura 57 – Geometria do vertedouro com o defletor.	120
Figura 58 – Detalhe da geometria do defletor.	121
Figura 59 – Construção do defletor – Montagem da forma.	122
Figura 60 – Construção do defletor – Detalhe da armadura.	122
Figura 61 – Construção do defletor – Concretagem.	123
Figura 62 – Construção do defletor – Concretagem.	124
Figura 63 – Defletor após construção.	124
Figura 64 – Regimes de escoamento verificados no defletor.	125
Figura 65 – Escoamento a jusante do defletor.	127
Figura 66 – Comparação das linhas piezométricas e perfil de linha de água.	127
Figura 67 – Perfil médio da concentração de ar.	128
Figura 68 – Região de refinamento da malha computacional em dimensões de modelo físico (1:15).	132
Figura 69 - Escoamento multifásico: 3 fases.	134
Figura 70 – Domínio da malha computacional.	137
Figura 71 – TDG ao longo da bacia de dissipação.	141
Figura 72 – TDG ao longo da profundidade na seção $x = 60\text{m}$ (60m da crista do vertedouro).	141
Figura 73 – TDG ao longo da bacia de dissipação $\alpha_b = 3\%$, $D_b = 0,8\text{mm}$	142
Figura 74 – TDG na Elv 241 m ao longo da bacia de dissipação e tempo de simulação.	142
Figura 75 – TDG na Elv 246 m ao longo da bacia de dissipação e tempo de simulação.	143
Figura 76 – Concentração de TDG, $Q_{\text{vertida}} = 800 \text{ m}^3/\text{s}$	144
Figura 77 – Concentração de TDG, $Q_{\text{vertida}} = 800 \text{ m}^3/\text{s}$, Elv 241 $x = 161\text{m}$	145
Figura 78 – Concentração de ar ao longo da bacia de dissipação.	146

Figura 79 – Concentração de ar ao longo da bacia de dissipação, para $D_b = 0,6\text{mm}$, $0,8\text{mm}$ e $1,0\text{mm}$	147
Figura 80 – Distribuição do TDG ao longo da bacia $\alpha_b = 3\%$ $d_b = 0,8\text{mm}$	147
Figura 81 – Fluxograma das principais etapas para o cálculo do TDG.	148
Figura 82 – Fluxograma das principais etapas para o cálculo do TDG utilizando autômatos celulares.	156
Figura 83 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6\text{mm}$	158
Figura 84 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6\text{ mm}$; $0,8\text{ mm}$ e $1,0\text{ mm}$	160
Figura 85 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $\alpha_b = 5\%$ $d_b = 0,8\text{mm}$	161
Figura 86 – Distribuição do TDG conforme a profundidade.	162
Figura 87 – Distribuição do TDG ao longo da bacia $\alpha_b = 3\%$ $d_b = 0,8\text{mm}$	162
Figura 88 – Distribuição da velocidade da “bolha” no eixo z conforme a profundidade e distância da crista.	163
Figura 89 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6\text{mm}$	165
Figura 90 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6\text{ mm}$; $0,8\text{ mm}$ e $1,0\text{ mm}$	167
Figura 91 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $\alpha_b = 5\%$ $d_b = 0,8\text{mm}$	168
Figura 92 – Distribuição do TDG conforme a profundidade.	169
Figura 93 – Distribuição do TDG ao longo da bacia $\alpha_b = 3\%$ $d_b = 0,8\text{mm}$	169
Figura 94 – Distribuição da velocidade da “bolha” no eixo z conforme a profundidade e distância da crista.	170
Figura 95 – Comparação do TDG – Autômato Celular e Modelo Numérico.	171

LISTA DE TABELAS

Tabela 1 - Resistência ao TDG e tempo de exposição do <i>Atlantic Salmon</i>	32
Tabela 2 – Parâmetros dos Testes 1, 2 e 3.....	43
Tabela 3 – Quantidade de incorporação de ar no ressalto hidráulico.	51
Tabela 4 – Constantes e termos de fechamento na simulação DES.	72
Tabela 5 – Coeficientes D_i e K_i para cálculo da flutuação de velocidade.	109
Tabela 6 – Configurações hidráulicas e resultados – Com defletor.	126
Tabela 7 – Configurações hidráulicas e resultados – Sem defletor.....	126
Tabela 8 - Coeficientes e parte das relações de fechamento do modelo $k-\omega$ SST.....	130

LISTA DE QUADROS

Quadro 1 – Resumo dos modelos de turbulência.	62
Quadro 2 – Simulações de TDG	157
Quadro 3 – Simulações de TDG.	164

LISTA DE ABREVIATURAS OU SIGLAS

AC	- Autômato Celular
CACM	- <i>Cellular Automata for Complex Matter</i>
CFD	- <i>Computational Fluid Dynamics</i>
CNTP	- Condições Normais de Temperatura e Pressão)
CPU	- <i>Central Processing Unit</i>
D2Q5	- Modelo de reticulado bidimensional com cinco direções de movimento
D2Q7	- Modelo de reticulado bidimensional com sete direções de movimento
DES	- <i>Detached Eddy Simulation</i>
DNS	- <i>Direct Numerical Simulation</i>
FHP	- Modelo de Autômato Celular do tipo Gás em Rede desenvolvido por Frisch, Hasslacher e Pomeau
GBD	- <i>Gas Bubble Disease</i>
HPP	- Modelo de Autômato Celular do tipo Gás em Rede desenvolvido por Hardy, Pazziz e Pomeau
LBM	- <i>Lattice Boltzmann Method</i>
LES	- <i>Large-Eddy Simulation</i>
LGCA	- <i>Lattice Gas Cellular Automata</i>
MHR	- Modelo Hidráulico Reduzido
PIMPLE	- <i>Pressure-Implicit Method for Pressure-Linked Equations</i>
PISO	- <i>Pressure Implicit Splitting of Operators</i>
RANS	- <i>Reynolds-Averaged Navier Stokes</i>
RSM	- <i>Reynolds Stress Model</i>
SBG	- <i>Stochastic Bubble Generator</i>
SIMPLE	- <i>Semi-Implicit Method for Pressure-Linked Equations</i>
SPH	- <i>Smoothed Particle Hydrodynamics</i>
SRS	- <i>Scale Resolving Simulation</i>
TDG	- <i>Total Dissolved Gas</i> – Total de Gás Dissolvido
UHE	- Usina Hidroelétrica
VOF	- <i>Volume of Fluid</i>

LISTA DE SÍMBOLOS

© - copyright

@ - arroba

® - marca registrada

Σ - somatório de números

Π - produtório de números

∇ - operador nabla - operador diferencial vetorial

SUMÁRIO

1	INTRODUÇÃO	19
2	DOENÇA DA BOLHA E O TOTAL DE GÁS DISSOLVIDO.....	22
2.1	CÁLCULO DO TDG	24
2.2	ICTIOFAUNA DA BACIA AMAZÔNICA.....	27
2.3	EFEITOS DA SUPERSATURAÇÃO EM PEIXES	28
3	MODELO HIDRÁULICO REDUZIDO	34
3.1	CLASSIFICAÇÃO DO ESCOAMENTO E CURVA DE PERFORMANCE	35
3.1.1	Mensuração do TDG no vertedouro da UHE Colíder	39
3.2	MENSURAÇÃO DE AR EM MHR	40
4	MODELAGEM MATEMÁTICA DO TDG	44
4.1	MODELO PREDITIVO DO TDG	44
4.2	CLUSTERIZAÇÃO DE “BOLHAS”	53
4.3	HIDRODINÂMICA	57
4.4	SIMULAÇÃO COMPUTACIONAL DE ESCOAMENTO MULTIFÁSICOS	65
4.4.1	Caracterização do escoamento multifásico	66
4.4.2	Tipos de modelos computacionais multifásicos	67
4.4.3	Modelo <i>Volume of Fluid</i> (VOF)	68
4.4.4	Algoritmo simplificado do VOF.....	70
4.4.5	DES (<i>Detached-Eddy Simulation</i>).....	71
4.4.6	PISO (<i>Pressure Implicit Splitting of Operators</i>).....	73
4.4.7	SIMPLE (<i>Semi-Implicit Method for Pressure-Linked Equations</i>).....	73
4.5	MODELAGEM DA CONCENTRAÇÃO DE AR.....	73
4.6	MODELAGEM DA “BOLHA”	75
4.7	MODELAGEM DO TDG	77
5	AUTÔMATO CELULAR.....	83

5.1	INTRODUÇÃO	83
5.2	O AUTÔMATO CELULAR.....	87
5.2.1	Autômatos celulares do tipo gás em rede (LGCAs).....	95
5.2.2	Exemplo de modelo de autômato celular para águas rasas	99
5.2.3	Exemplo de modelo de autômato celular para simulação da série temporal acumulada de massa liberada de CO ₂	101
5.3	MODELOS MATEMÁTICOS PARA ESCOAMENTOS COM “BOLHAS” ...	105
5.4	GERADOR DE “BOLHAS” ESTOCÁSTICO.....	107
6	DESENVOLVIMENTO DO CÁLCULO DO TDG	110
6.1	ESTUDO DE CASO	110
6.1.1	UHE Colíder.....	110
6.1.2	Descrição geral do arranjo.....	111
6.1.3	Barragem	114
6.1.4	Vertedouro	116
6.1.5	Circuito de Geração.....	118
6.1.6	Geometria final do defletor	120
6.2	DADOS HIDRODINÂMICOS.....	126
6.3	CÁLCULO DO VOLUME DE AR.....	128
6.3.1	<i>Solver</i> InterFoam	129
6.3.2	Modelo de turbulência $k-\omega$	129
6.4	MALHA COMPUTACIONAL.....	131
7	IMPLEMENTAÇÃO DO MODELO MATEMÁTICO	133
7.1	MALHA COMPUTACIONAL E PREENCHIMENTO DE FALHAS	135
7.2	CÁLCULO DO TDG	138
7.3	RESULTADOS	140
8	IMPLEMENTAÇÃO DO AUTÔMATO CELULAR	149
8.1	MALHA COMPUTACIONAL E PREENCHIMENTO DE FALHAS	149

8.2	PARÂMETROS UTILIZADOS	150
8.3	CÁLCULO DO TDG UTILIZANDO AUTÔMATOS CELULARES	150
8.4	RESULTADOS PARA $N_t = 1.000.000$	157
8.5	RESULTADOS PARA $N_t = 5.000.000$	163
8.6	CONCLUSÃO SOBRE O CÁLCULO DE TDG COM AUTÔMATOS CELULARES.....	171
9	CONCLUSÕES E RECOMENDAÇÕES.....	174
9.1	CONCLUSÕES	174
9.2	RECOMENDAÇÕES.....	175
10	REFERÊNCIAS.....	176
	APÊNDICE I – Leitura de arquivo (Ler_Velocidade)	187
	APÊNDICE II – Modelo de autômato celular.....	225

1 INTRODUÇÃO

A Saturação Gasosa, também conhecida como Total de Gás Dissolvido ou TDG (*Total Dissolved Gas*), é a soma das pressões parciais de todos os gases dissolvidos na água (principalmente: Oxigênio, Nitrogênio, Argônio e Dióxido de Carbono) e é muito influenciado pela pressão hidrostática (Weitkamp, 2008). Águas supersaturadas tendem a perder gás para atmosfera, com taxa de decaimento lenta e com a transferência de gás ocorrendo na interface entre a água e o ar (Colt, 2012). A supersaturação pode ser produzida em barragens, cachoeiras e corredeiras. Eventos de vertimento em usinas hidroelétricas elevam os níveis de TDG, podendo provocar a doença da bolha (*Gas Bubble Disease* - GBD) na ictiofauna. O efeito da supersaturação de TDG sobre a ictiofauna depende dos níveis de TDG, do tempo de exposição e da profundidade (Politano *et al.*, 2012). A tolerância da ictiofauna à elevados níveis de TDG variam conforme a espécie, temperatura, frequência e intensidade da exposição.

No Brasil não existe legislação específica sobre a concentração admissível de TDG nos corpos de água, sendo que o monitoramento rotineiro de reservatórios não contempla o monitoramento do total de gases dissolvidos, sendo que este parâmetro de qualidade de água não consta nas resoluções do CONAMA (Conselho Nacional de Meio Ambiente) (Agostinho *et al.*, 2021). No estado de Washington, EUA, o limite de TDG é de até 110%, sendo permitido que a concentração exceda os 120% no caso de cheias excepcionais (Politano *et al.*, 2012). A Agência de Proteção Ambiental Americana, US EPA, estabelece como critério o valor de 115% (USEPA, 2020).

Li *et al.* (2022) apresenta uma lista com mais de 30 usinas hidroelétricas com problemas de supersaturação, localizadas nos Estados Unidos, Noruega e China, com TDG de até 205%.

Uma forma de minimizar o aumento da supersaturação de TDG é instalar defletores no vertedouro, modificando a hidrodinâmica do escoamento, evitando que a alta velocidade do fluxo “empurre” as “bolhas” de gás para regiões profundas da bacia de dissipação, onde a alta pressão favorece a dissolução destas “bolhas” e, conseqüentemente, o aumento do TDG. Na literatura encontram-se avaliações de defletores construídos na UHE Colíder (Andriolo *et al.*, 2023), UHE Hells Canyon (Politano e Laughery, 2016); UHE McNary (Politano *et al.*, 2015), (Wang, Politano e

Weber, 2018); UHE Wanapum (Dotson, 2015); UHE Brownlee (Myers e Parkinson, 2003) e UHE Yacyretá (Bacchiega e Fattor, 2004).

Idealmente, o projeto desses defletores deve ser fundamentado em ensaios em modelo físico, complementados por estudos em modelos matemáticos ou computacionais. Entretanto, simulações numéricas demandam elevado esforço computacional.

No Brasil, durante a fase de Estudos Básicos e de Estudos de Viabilidade de usinas hidroelétricas, fase anterior ao processo de licitação de um empreendimento, os recursos financeiros disponíveis são mais escassos, dificultando o projeto de um defletor com estudos em modelo físico e matemático.

Considerando a experiência adquirida pelo autor nos estudos desenvolvidos para construir o defletor do vertedouro da UHE Colíder (Andriolo *et al.*, 2023), a hipótese adotada nesta tese foi de que seria possível, utilizando um modelo de autômato celular, calcular o valor de TDG utilizando como dados de entrada os parâmetros hidrodinâmicos calculados pelo software OpenFOAM. Assim, reduzindo a necessidade de recursos computacionais mais sofisticados, seria possível estudar possíveis geometrias de defletor, de forma preliminar, já nas fases prévias ao projeto executivo.

O objetivo desta tese foi desenvolver um modelo computacional baseado em autômatos celulares para o cálculo do TDG, capaz de estimar o valor máximo e o comportamento espacial do TDG ao longo do vertedouro e da bacia de dissipação.

Devido ao baixo custo computacional, o modelo proposto pode ser aplicado nas etapas de Estudo de Viabilidade ou Projeto Básico de aproveitamentos hidroelétricos. Nesses estágios iniciais, o modelo desenvolvido permite estudar os valores máximos de TDG e propor, de forma simplificada, a geometria preliminar de um defletor em vertedouros, contribuindo para reduzir, desde o início do projeto, os riscos à ictiofauna.

Cabe destacar que no Brasil, a doença da bolha induzida por supersaturação de TDG em vertimentos, é um assunto em fase inicial de pesquisa, sendo o primeiro projeto do tipo proposto ao órgão licenciador ambiental no ano de 2020 pela UHE Colíder, cuja construção do defletor foi concluída em novembro de 2021 (Andriolo M. V. *et al.*, 2022).

Os objetivos específicos desta tese são:

- a) Desenvolver um modelo computacional baseado em autômatos celulares para a estimar o total de gás dissolvido (TDG) em vertedouros durante a fase de projeto conceitual utilizando como dado de entrada apenas o campo de velocidade do fluxo obtido a partir de simulações hidrodinâmicas.
- b) Implementar no modelo de autômato celular um método de cálculo da velocidade relativa entre a fase água e a fase “bolha” do escoamento;

Esta tese está organizada em 9 capítulos. A presente introdução apresentada no capítulo 1. O capítulo 2, intitulado *Doença da Bolha e o Total de Gás Dissolvido* apresenta como o total de gás dissolvido e a supersaturação pode provocar a doença da bolha. O capítulo 3, intitulado *Modelo Hidráulico Reduzido* apresenta os estudos desenvolvidos em MHR para avaliar a influência do defletor no comportamento hidrodinâmico do jato. O capítulo 4, intitulado *Modelagem Matemática do TDG* apresenta a formulação matemática para o cálculo do TDG. O capítulo 5, intitulado *Autômato Celular* apresenta a origem dos autômatos celulares, exemplos de aplicação e sua formulação matemática. O capítulo 6, intitulado *Desenvolvimento do Cálculo do TDG*, descreve o estudo de caso, as simulações hidrodinâmicas realizadas para este caso e os resultados de outros estudos. O capítulo 7, intitulado *Implementação do Modelo Matemático*, descreve como o modelo matemático desenvolvido por Wang et al. (2019) foi implementado em linguagem Go, utilizando os resultados do modelo hidrodinâmico para “alimentar” o modelo desenvolvido para o cálculo do TDG, e os resultados obtidos com esse modelo. O capítulo 8, intitulado *Implementação do Autômato Celular*, descreve a formulação computacional para o cálculo do TDG utilizando os autômatos celulares e os resultados obtidos com esse modelo. O capítulo 9, intitulado *Conclusões e Recomendações*, apresenta as principais conclusões desta tese e as sugestões para futuros estudos.

2 DOENÇA DA BOLHA E O TOTAL DE GÁS DISSOLVIDO

O total de gás dissolvido (TDG – *Total Dissolved Gas*) é a medida da soma das pressões parciais de todos os gases dissolvidos na água, incluindo vapor de água. Tipicamente, na maioria dos corpos naturais de água, o TDG é uma medida da quantidade de nitrogênio, oxigênio, argônio, dióxido de carbono e vapor de água que é dissolvido em uma determinada quantidade de água. A solubilidade dos gases atmosféricos na água é principalmente afetada pela temperatura e pressão (Myers e Parkinson, 2003).

A solubilidade do oxigênio é quase o dobro da do nitrogênio e, portanto, na água, as porcentagens de nitrogênio e oxigênio são de aproximadamente 64% e 36%, respectivamente. No entanto, as medições de oxigênio sozinhas não podem ser usadas para prever a concentração de TDG, pois o oxigênio é um gás biologicamente ativo. Em contraste, o nitrogênio é biologicamente inerte e, portanto, a proporção entre esses gases é normalmente desconhecida. Em um estudo experimental elaborado por Li et al. (2018), correlacionando o TDG com o oxigênio, foi demonstrado que esta correlação resultou em erros significativos.

A “Doença da bolha” (*bubble disease*) pode aparecer com valores de TDG próximos de 105-110%, podendo afetar a vida dos peixes (Bacchiaga e Fattor, 2004), resultando na formação de “bolhas” de gases no sangue e tecidos de animais aquáticos (Colt, 2012), depende além dos níveis de TDG, do tempo de exposição e da profundidade (Politano *et al.*, 2012).

A capacidade da água de suportar gases dissolvidos é tanto maior quanto mais elevada for a pressão e menor for a temperatura. A saturação de gases medida na superfície de um corpo d’água tende a cair cerca de 10% a cada metro, pois o aumento da pressão permite a água suportar mais gases nela dissolvidos. Quando a capacidade de suportar gases diminui, seja pelo aumento de temperatura ou redução na pressão, estes passam a escapar para a superfície, formando “bolhas” em tecidos internos de animais aquáticos. Os sintomas deste fenômeno compõem a já mencionada “doença da bolha”, podendo resultar em morte por embolia gasosa.

Em vertedouros a capacidade da “bolha” mergulhar a maiores profundidades depende do regime do jato, da geometria do vertedouro e da elevação do nível de água de jusante, sendo a concentração de TDG governada pela turbulência e hidrodinâmica (Politano *et al.*, 2012). A supersaturação do gás ocorre quando as

“bolhas” de ar são carregadas para regiões mais profundas (maior pressão) e a solubilidade é aumentada pela alta pressão (Politano *et al.*, 2016).

A transferência de oxigênio das “bolhas” para a água é medida a partir da evolução temporal da concentração de oxigênio dissolvido na água, obtida pelo método de desgaseificação. Em relação à dinâmica das “bolhas”, observa-se que a velocidade vertical média diminui com α (fração volumétrica de gás) em concordância com ensaios experimentais e numéricas anteriores, enquanto a agitação das “bolhas” se mostra pouco dependente de α . Em relação à transferência de massa, o número de Sherwood é muito próximo ao de uma única “bolha” subindo com o mesmo número de Reynolds, desde que este último seja baseado na velocidade vertical média da “bolha”, que leva em conta o efeito da fração volumétrica de gás na velocidade de ascensão da “bolha”. Essa conclusão é válida para situações em que o coeficiente de difusão do gás no líquido é muito baixo (número de Péclet alto) e o gás dissolvido está bem misturado na escala da “bolha” (Colombet *et al.*, 2014).

A transferência de massa ar-água através da interface da “bolha” de ar é significativa, pois a área de superfície líquida de milhares de pequenas “bolhas” é muito maior do que a área de superfície acima das nuvens de “bolhas” (Chanson, Aoki e Maruyama, 2002). A desgaseificação na superfície livre após as “bolhas” deixarem a região de produção de TDG é um processo ineficiente, a menos que a transferência de massa seja intensificada pelo vento ou turbulência.

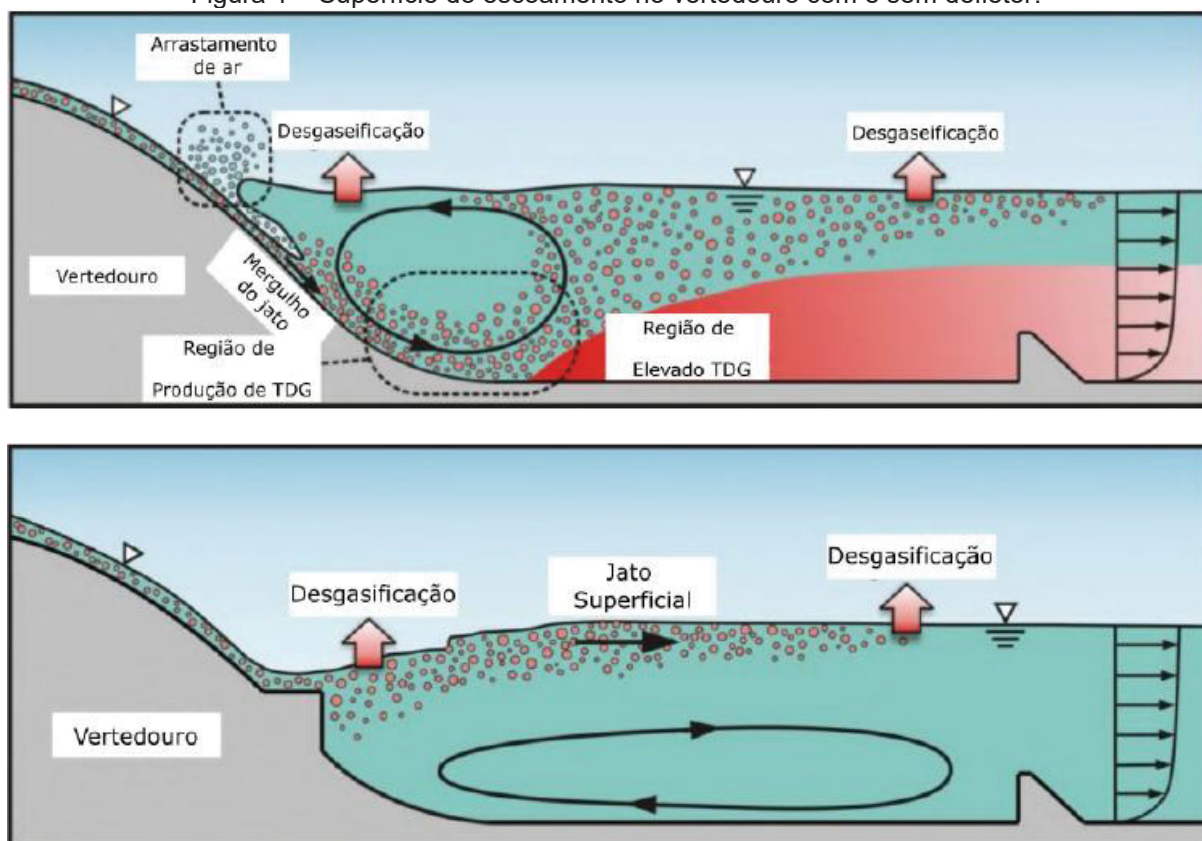
No Brasil não existe legislação específica sobre a concentração de gás dissolvido nos corpos de água. Por exemplo, no estado de Washington, EUA, o limite de TDG é limitado a 110%. Entretanto, é permitido que a concentração exceda os 120% nos seguintes casos: a) Cheias maiores que a $Q_{7,10}$ (Cheia com recorrência de 10 anos e sete dias de duração) ou b) Vertimentos com objetivo de ajudar a migração de salmões juvenis (Politano *et al.*, 2012).

A revisão da literatura nacional, demonstra que o fenômeno da supersaturação gasosa em vertedouros e seus efeitos biológicos, é muito pouco conhecida ou estudada no Brasil. O levantamento recente apresentado por Agostinho *et al.* (2021), identificou 251 relatos de mortandade de peixes no Brasil entre os anos de 2010 e 2020, tendo como indicativo que 0,8% dessas mortes foram atribuídas a supersaturação gasosa. Entretanto, o mesmo trabalho afirma que, alguns eventos foram preliminarmente atribuídos à anoxia ou alta turbidez e assim publicados, sendo

posteriormente reclassificados como supersaturação gasosa e barotrauma, ficando evidente a dificuldade do diagnóstico da mortandade causada por TDG.

Para minimizar a supersaturação de gases dissolvidos foram instalados defletores nos vertedouros de diversas usinas (Politano, Carrica e Weber, 2009), como: Hells Canyon Dam (Myers e Parkinson, 2003) (Politano e Laughery, 2016); McNary Dam (Politano *et al.*, 2015) (Wang, Politano e Weber, 2018); Wanapum Dam (Hadjeriousa, Politano, Bender, *et al.*, 2014); Brownlee Dam (Hadjeriousa, Politano, DeNeale, *et al.*, 2014), UHE Yacyretá (Bacchiega e Fattor, 2004) e UHE Colíder (Andriolo M. V. *et al.*, 2022). O projeto destes defletores busca alterar o fluxo de água de forma que sua trajetória seja direcionada à superfície, fazendo com que as “bolhas” sejam transportadas em uma fina camada superficial, evitando que essas “bolhas” mergulhem a profundidades maiores, minimizando a dissolução dos gases, conforme apresentado na Figura 1 (Politano *et al.*, 2016).

Figura 1 – Superfície do escoamento no vertedouro com e sem defletor.



Fonte: Politano et al. (2016).

2.1 CÁLCULO DO TDG

No ar seco, a fração molar, ou porcentagem em volume, de nitrogênio, oxigênio, argônio e dióxido de carbono é de 78,084%, 20,946%, 0,934% e 0,0320%,

respectivamente (Glueckauf, 1951 citado por (Colt, 1983)), sendo que os 0,004% restantes consistem em gases nobres, hidrocarbonetos e traços de gases.

No equilíbrio, a pressão de cada gás na fase líquida é a mesma que a pressão na fase gasosa (Glasstone, 1946 citado por (Colt, 1983)).

Colt (1983) apresenta a Equação (2.1) para calcular a pressão parcial em mmHg de cada gás na fase gasosa:

$$P_i^g = Z_i(BP - RH P_{H_2O}/100), \quad (2.1)$$

sendo Z_i a fração molar do i-ésimo gás; BP a pressão barométrica (mmHg); P_{H_2O} a pressão do vapor de água (mmHg); RH a umidade relativa do ar.

No método de difusão por membrana, a pressão diferencial do gás (ΔP) é lida diretamente. A pressão total do gás na fase líquida e na fase gasosa é, portanto, igual a $BP + \Delta P$ e BP , respectivamente. A pressão total do gás (TGP – *Total Pressure Gas ou Total Dissolved Gas Pressure*) é importante porque seu valor deve exceder todas as pressões compensatórias antes que uma “bolha” possa existir em um líquido. Se a pressão total do gás da fase líquida for menor ou igual à pressão atmosférica, “bolhas” de gás não podem se formar, independentemente dos níveis de nitrogênio, argônio e dióxido de carbono (Weitkamp e Katz, 1980). A pressão total do gás (percentual de saturação em relação à superfície) pode ser calculada de três maneiras, dependendo da inclusão ou exclusão de vapor d'água nas duas fases (Schneider & D'Aoust, 1976 in (Colt, 1983)), conforme apresentado nas Equações (2.2), (2.3) e (2.4):

$$TDG (\%) = \left[\frac{BP + \Delta P - P_{H_2O}}{BP - P_{H_2O}} \right] 100, \quad (2.2)$$

$$TDG (\%) = \left[\frac{BP + \Delta P - P_{H_2O}}{BP} \right] 100, \quad (2.3)$$

$$TDG (\%) = \left[\frac{BP + \Delta P}{BP} \right] 100, \quad (2.4)$$

onde ΔP é a pressão diferencial do gás.

O método preferencial para tratar da pressão total dos gases é em termos de ΔP . Este é o excesso de pressão total dos gases dissolvidos em relação à pressão barométrica e é medido diretamente pelo método de difusão por membrana (Colt, 2012).

O coeficiente de Bunsen (β) representa historicamente a solubilidade de um gás real em litros nas CNTP (condições normais de temperatura e pressão) quando a pressão parcial na fase gasosa é igual a 1 atmosfera padrão (1 atm). A pressão parcial do gás *i*-ésimo nas atmosferas é apresentada na Equação (2.5):

$$Pressão_{parcial}(atm) = \chi_i(BP - P_{wv}), \quad (2.5)$$

em que χ_i é a fração molar do gás *i*-ésimo (atm); BP a pressão barométrica (atm); P_{wv} a pressão do vapor de água (atm).

A concentração de saturação ($C_{p,\chi}$) em unidades de gás L nas CNTP/L de água é apresentada na Equação (2.6):

$$C_{p,\chi} = \beta_i \chi_i (BP - P_{wv}). \quad (2.6)$$

Os valores de β_i variam conforme a temperatura e salinidade, sendo possível encontrar valores tabelados em Colt (2012).

Para um gás puro, $BP = 1 + P_{wv}$, $C_{p,\chi} = \beta_i$, a Equação (2.6) pode ser usada para calcular a solubilidade de um gás a uma pressão, temperatura e composição gasosa arbitrária. Embora o coeficiente de Bunsen tenha sido historicamente expresso em termos de L/(L atm), esta não é a unidade mais conveniente para gases atmosféricos. A Equação (2.6) pode ser convertida para mg/L multiplicando o lado direito por 1.000mL/L x K_i e a pressão em kPa, resultando na Equação (2.7):

$$C_{p,\chi} = \left[\frac{1.000 K_i \beta_i}{101.325} \right] \chi_i (BP - P_{wv}). \quad (2.7)$$

Definindo o termo entre colchetes da Equação (2.7) como β_i'' com unidade de mg L⁻¹ kPa⁻¹, a Equação (2.6) pode ser reescrita na forma apresentada na Equação (2.8):

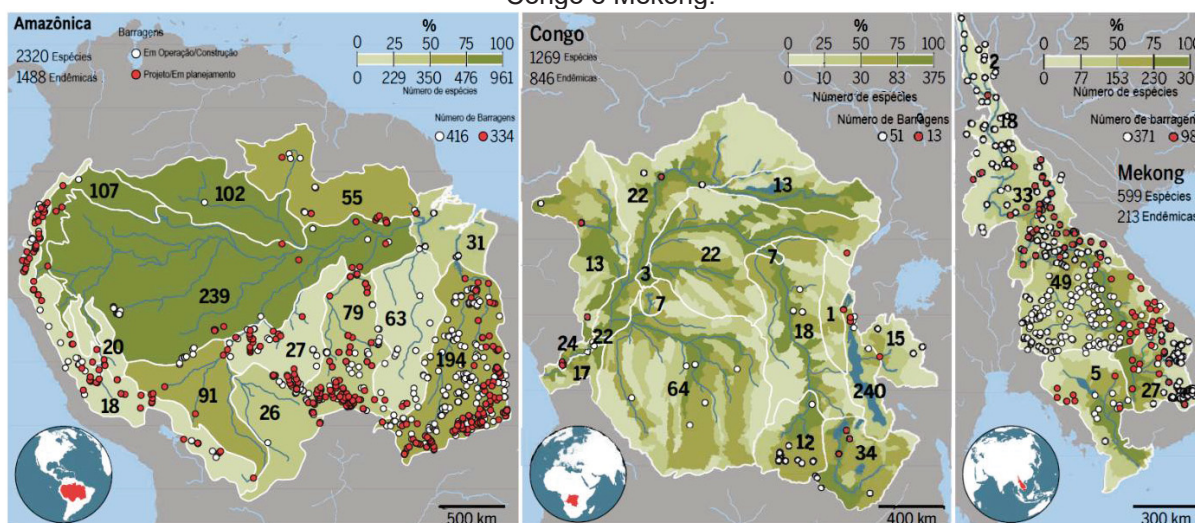
$$C_{p,\chi} = \beta_i'' \chi_i (BP - P_{wv}), \quad (2.8)$$

sendo BP e P_{wv} com unidade de kPa.

2.2 ICTIOFAUNA DA BACIA AMAZÔNICA

Segundo Winemiller et al. (2016), um terço das espécies de peixes de água doce do mundo está localizada nas bacias do rio Amazonas, Congo e Mekong, sendo que a bacia amazônica detém mais de 2.300 espécies, a bacia do Congo 1.270 espécies e a bacia do Mekong quase 600 espécies diferentes. A Figura 2 ilustra a localização de barragens construídas e em projeto nestas bacias bem como o número de espécies em cada uma delas.

Figura 2 – Diversidade das espécies de peixes e a localização de barragens na bacia Amazônica, Congo e Mekong.



Fonte: Winemiller et al. (2016).

A região biogeográfica neotropical, que corresponde à América Latina, concentra provavelmente a maior biodiversidade do planeta e abriga a ictiofauna de água doce mais diversificada e rica do mundo, com 71 famílias, centenas de gêneros e seis mil espécies de peixes ((Menezes, 1996), (Lundberg *et al.*, 2000), (Albert e Reis, 2011)). Embora as estimativas sejam variáveis, por representar uma grande parcela da região neotropical, o Brasil apresenta a maior riqueza de espécies de peixes de água doce do mundo (Buckup, Menezes e Ghazzi, 2007).

De acordo com Lowe-McConnell (1999), as espécies de peixes da região amazônica são representadas principalmente pelos *Ostariophysi*, sendo 43% da ordem *Characiformes*, 39% da ordem *Siluriformes* e 3% da ordem *Gymnotiformes*. Muitas dessas espécies estão amplamente distribuídas, enquanto outras são restritas a ambientes específicos em função de barreiras ambientais ou em função de características químicas da água (Junk, Bayley e Sparks, 1989).

O rio Tapajós, com cerca de 360 espécies de peixes registradas (Ferreira e J. A. S. Zuanon, 1998), drena uma área de aproximadamente 492 mil km², tem como principais afluentes os rios Juruena e Teles Pires (ou São Manuel).

Na bacia do rio Teles Pires, onde está localizada a UHE Colíder, um dos rios amazônicos com a maior quantidade de espécies, um estudo recente identificou 355 espécies de peixes de 11 ordens e 42 famílias (Andriolo M. V. *et al.*, 2022).

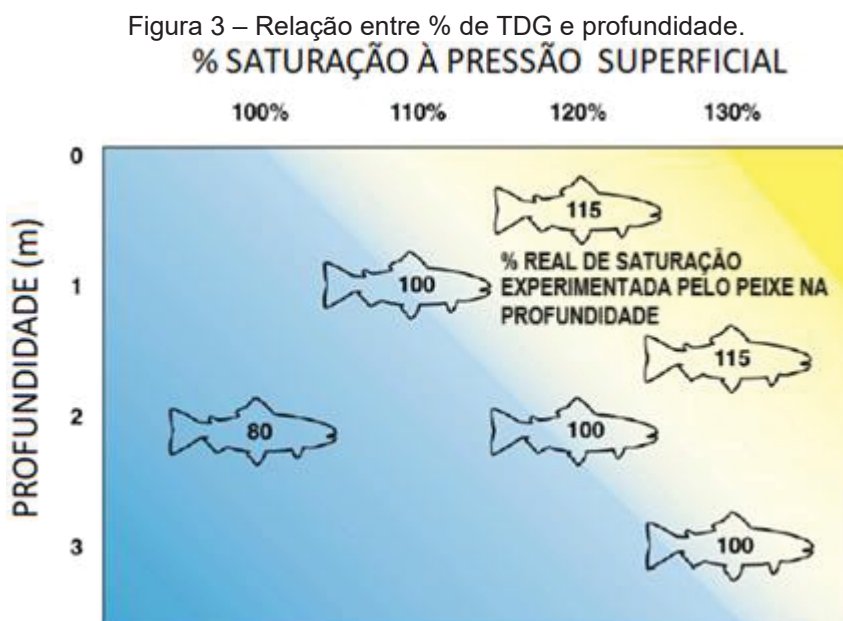
2.3 EFEITOS DA SUPERSATURAÇÃO EM PEIXES

A água tem sua capacidade de dissolver gases aumentada com o aumento da pressão e/ou a diminuição da temperatura. A condição de saturação de um volume de água aumenta cerca de 10% a cada metro adicional de profundidade. Quando a capacidade de suportar gases diminui, por aumento de temperatura ou queda na pressão (redução da profundidade), os gases escapam para a superfície ou formam “bolhas” em tecidos internos dos animais aquáticos. Este fenômeno produz a já mencionada “doença da bolha”, podendo resultar em morte dos peixes por embolia gasosa.

O fenômeno da supersaturação e seus efeitos biológicos têm sido muito pouco estudado no Brasil. Weitkamp (2008) realizou uma revisão da literatura sobre o assunto entre 1980 e 2007. Segundo o autor, a supersaturação é um fenômeno que ocorre naturalmente em cachoeiras, sendo que a cachoeira do Niágara apresenta TDG em níveis de 120% a 130%. Inversões térmicas em lagos podem, eventualmente, provocar embolia gasosa em peixes, pois a massa de água que ascende perde pressão e ganha calor, diminuindo sua capacidade de suportar gases. Há evidências ainda limitadas de aversão ativa por parte de peixes a altos níveis de TDG (Weitkamp, 2008). O comportamento observado dos peixes é o de nadar para locais mais profundos. A tolerância a elevados níveis de TDG varia conforme a espécie, temperatura ambiente, frequência e intensidade da exposição. Muitos estudos sobre esse aspecto no hemisfério norte se deram em ambiente controlado, onde a profundidade é pequena, o que impede o peixe de fugir para locais mais profundos.

Weitkamp (2008) comenta que casos de mortandade severa são atribuídos à supersaturação quando as profundidades disponíveis são menores que 1 m, e as concentrações de TDG são superiores a 130%. Em profundidades maiores a resistência do peixe à supersaturação de TDG aumenta. A Figura 3 ilustra de forma aproximada, o efeito da pressão hidrostática sobre os gases dissolvidos e os

potenciais efeitos biológicos na ictiofauna (a resistência do peixe a supersaturação aumenta em aproximadamente 10 pontos percentuais a cada metro adicional de profundidade). Ilustrando, o número apresentado dentro da figura do peixe (Figura 3), representa o efeito da profundidade na resistência biológica do peixe à supersaturação gasosa. Por exemplo, quando o peixe está aproximadamente a 3 m de profundidade e a supersaturação da água está em 130%, o efeito biológico da supersaturação (130% a 3 m) seria equivalente a 100% (na superfície).



Fonte: Weitkamp (2008).

A patologia inicial da “doença da bolha” em trutas arco-íris começa com “bolhas” nos capilares da retina, progredindo para exoftalmia (olho saltado) quando o peixe é exposto a TDG entre 114% e 118% (Machado *et al.*, 1987). Fidler (1988) desenvolveu um modelo teórico para a formação e crescimento de “bolhas”, identificando que a diferença de pressão necessária para surgimento de “bolhas” no sistema cardiovascular de peixes no Hemisfério Norte é de 1,15 a 1,18 atmosferas, o que corresponde à 115% a 118% de TDG. Antcliffe; Fidler e Birtwell (2003) fez experimento em laboratório com trutas em exposição crônica a diferentes níveis de TDG, que apresentou taxa de letalidade (LT50 (50% dos exemplares morreram)) de 55 h de exposição a 122% e 5,1 h a 140% de TDG. Estes estudos foram realizados com uma espécie típica do hemisfério norte, para as condições encontradas nos rios daquela região.

A realidade para os peixes neotropicais é outra. A resistência de cada espécie e indivíduo à supersaturação gasosa pode ser diferente do observado em espécies

da América do Norte. Fatores como o tamanho dos poros da linha lateral e a estrutura da bexiga natatória podem influenciar na presença de sinais de “doença de bolha” e nos efeitos da supersaturação (Weitkamp, 2008). Nebeker, Kent Hauck e Baker (1979) estudaram diferentes espécies de salmonídeos expostas a supersaturações da ordem de 115% e 120% em temperaturas entre 8°C e 20°C. Estes autores concluíram que a elevação da temperatura aumentava significativamente a mortalidade em salmão real e truta arco-íris. Os rios de regiões temperadas da América do Norte têm águas mais frias que os rios neotropicais. As temperaturas encontradas no rio Teles Pires giram em torno de 27°C e 29°C no período chuvoso.

O sinal característico da embolia gasosa é a presença de “bolhas” em membranas finas, notadamente nos olhos, brânquias, nadadeiras e na linha lateral. Em alguns casos pode ocorrer também exoftalmia e, mais raramente, ruptura da bexiga natatória e eversão de parte do trato digestório. Internamente, é possível observar “bolhas” nos vasos sanguíneos. Porém, a morte por embolia gasosa é de difícil detecção. A mortalidade de peixes por embolia parece estar fracamente relacionada com os sinais visíveis da “doença da bolha” (Weitkamp, 2008), assim a grande maioria das mortes por embolia não são identificáveis. Isto ocorre porque seus sinais característicos, as “bolhas”, tendem a desaparecer pouco tempo após o incidente, ou ainda a formação de “bolhas” fica restrita a tecidos internos, gerando óbito sem nenhum sinal externo. Weitkamp (2008) relatou que os sinais de embolia gasosa desaparecem de brânquias seccionadas em 2 e 15 minutos, e em cerca de 30 minutos para brânquias intactas. Maule et al. (1999) verificaram que a presença de “bolhas” em filamentos branquiais cai de 70% para 10% quando o peixe é transferido de águas com 120% e 139% de TDG para águas com menos de 104% de TDG.

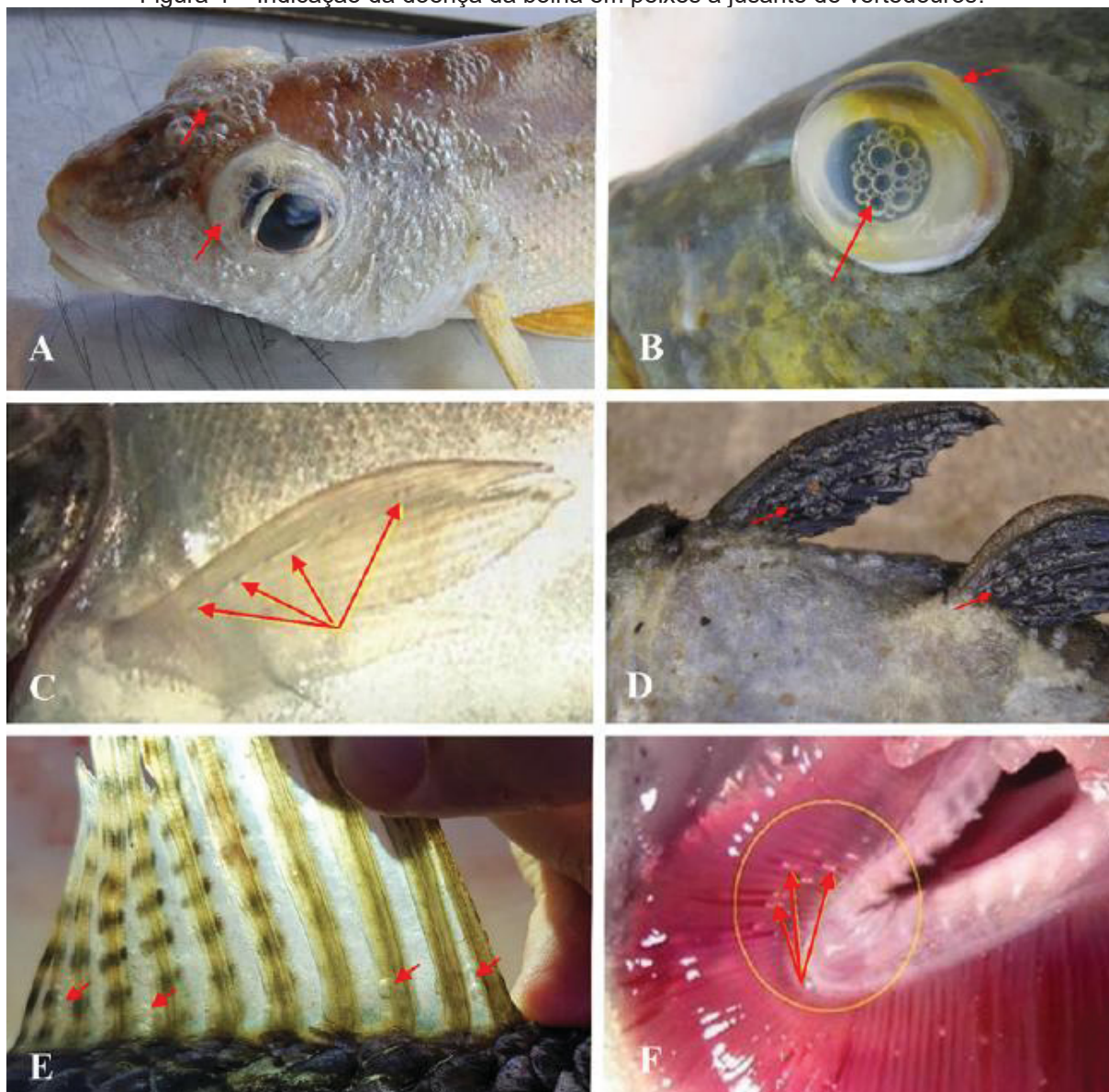
VanderKooi et al. (2003 apud (Weitkamp, 2008)) descobriram que sinais de “doença da bolha” eram mais evidentes em exposições crônicas à supersaturação (115% de TDG) quando comparadas a exposições agudas (125% e 135% de TDG). Segundo Weitkamp (2008), isto demonstra que, em exposições agudas, a formação de “bolhas” internas causa óbito antes do desenvolvimento de “bolhas” na superfície do animal, logo, antes do surgimento de sinais visíveis. Os resultados dos estudos sobre este tema no Hemisfério Norte sugerem que, enquanto a presença dos sinais acima citados deixa clara a ocorrência de embolia gasosa, não é possível através dessa detecção quantificar com segurança a quantidade de peixes que de fato podem ter sofrido embolia gasosa em determinado evento.

Uma consequência natural da embolia gasosa é a perda de coordenação motora apropriada, por infarto em órgãos e dor. Peixes que sofrem embolia podem perder a capacidade de se manter na coluna de água, podendo assim ser levados a se chocar em estruturas e sofrer as consequências da turbulência e forças de cisalhamento.

Na UHE Yacyretá (Domitrovic *et al.*, 1993), o maior número de exemplares afetados pelo TDG correspondeu a espécies de fundo ou frequentadoras de fundo de rio, como armados (*Pterodoras granulosus*, *Oxydoras kneri* e *Rhinodoras d'orbignyi*), bagres (*Pimelodus spp.*), cascudos (*Loricariichthys spp.*, *Hypostomus spp.*, *Rhineloricaria spp.* e *Megalancistrus gigas*), raias (*Potamotrygon sp.*), muçum (*Synbranchus marmoratus*), entre outras. Também se detectou, em menor número, a mortalidade de outras espécies como a piracanjuba (*Brycon orbignyanus*), curimbatá (*Prochilodus Scrofa*), piaus (*Leporinus spp* e *Schizodon spp.*), lambaris (*Astyanax spp.*), surubins (*Pseudoplatystoma spp.*, *Sorubim lima* e *Hemisorubim platyrhynchos*), dourados (*Salminus maxillosus*), entre outras.

Alguns exemplos de lesões devido a supersaturação de gases são ilustrados na Figura 4.

Figura 4 – Indicação da doença da bolha em peixes a jusante de vertedouros.



Sendo: Indicação da doença das bolhas gasosas em peixes a jusante de vertedouros, com bolhas no tegumento (A), globo ocular (B), exoftalmia (A, B), juntamente com os raios da nadadeira peitoral (C, D) e nadadeira dorsal (E) e no filamento branquial (F). **Fonte:** Agostinho et al. (2021).

Os estudos realizados sobre a mortalidade de peixes submetidos ao TDG variam conforme o percentual de supersaturação, tempo de exposição e profundidade, sendo o exemplo do Atlantic Salmon apresentado na Tabela 1.

Tabela 1 - Resistência ao TDG e tempo de exposição do *Atlantic Salmon*.

Espécie	Tempo de exposição (dias)	% TDG	Profundidade (m)	Probabilidade de mortandade (%)
<i>Atlantic salmon</i>	0,5	111,6+-1,1	0,1-0,55	81
	3	110,2+-0,6	0,1-0,55	55
	8	107,1+-2,2	0,1-0,55	25
	14	107,2+-0,1	0-1,15	0

Fonte: Pulg et al. (2022).

Na Tabela 1 é possível constatar o aumento acentuado da probabilidade de mortandade do peixe, com um pequeno aumento no percentual do TDG.

3 MODELO HIDRÁULICO REDUZIDO

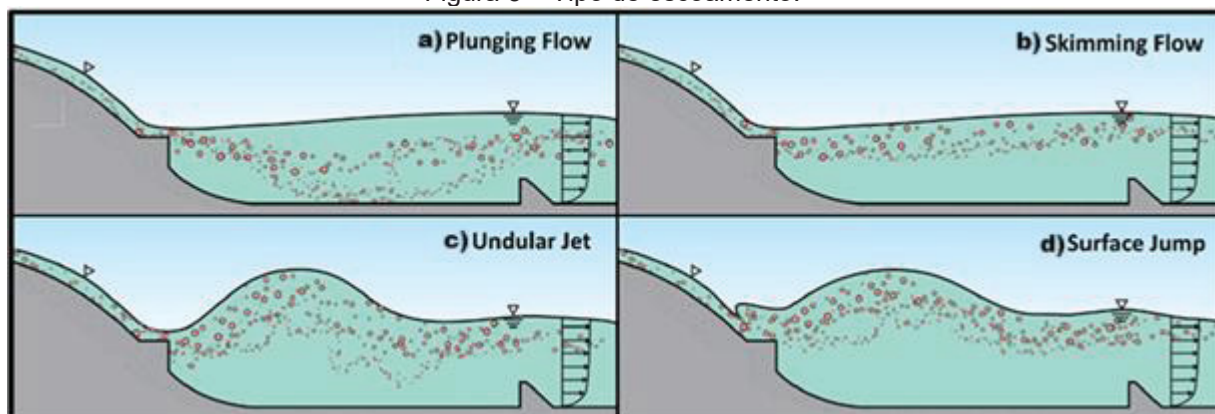
O estudo do comportamento do escoamento efluente do vertedouro em modelo hidráulico reduzido (MHR) enfocando a problemática da concentração de gás dissolvido é uma ferramenta importante para entender a hidrodinâmica do fluxo e, principalmente, subsidiar o projeto e a construção de defletores. Estudo em MHR para subsidiar o projeto de defletores foram utilizados no projeto dos defletores da UHE Yacyretá na Argentina (Bacchiega e Fattor, 2004), Hells Canyon Dam nos EUA (Politano e Laughery, 2016) e UHE Colíder no Brasil (Andriolo M. V. *et al.*, 2022).

Em escoamentos com superfície livre, as forças preponderantes possíveis de atuar sobre o fluido são as forças de gravidade e de inércia, caracterizadas pelo número de Froude, que representa de forma adimensional a relação entre as forças de inércia e de gravidade que atuam no escoamento. Resulta, dessa consideração, o critério de semelhança de Froude.

O escoamento em vertedouros com defletor pode ser classificado em quatro tipos principais: *plunging flow*, *skimming flow*, *undular jet* e *surface jump* (Politano e Laughery, 2016). As principais características desses escoamentos são descritas abaixo e ilustradas na Figura 5.

- a) *Plunging flow*: Caracteriza-se pelo mergulho do jato em direção ao fundo da bacia de dissipação, que devido à alta pressão hidrostática resulta em um escoamento com alta produção de TDG.
- b) *Skimming Flow*: Ocorre quando o fluxo do jato é praticamente tangente à superfície de água, com ângulo do escoamento na saída do defletor próximo a 5° em relação a horizontal. Dessa forma, as “bolhas” de água não são carregadas até o fundo da bacia, reduzindo a dissolução dos gases.
- c) *Undulating Surface Jet*: Ocorre quando o ângulo de lançamento é da ordem de 20°, com jato superficial. É um regime de transição entre o *skimming flow* e o *surface jump*, no qual observa-se uma sobre-elevação do escoamento logo após o defletor.
- d) *Surface Jump*: Alta submergência, formação de um “rolo” sobre o defletor com arraste de “bolhas” para o fundo mais a jusante, possui características similares a um ressalto hidráulico convencional.

Figura 5 – Tipo de escoamento.



Fonte: Politano e Laughery (2016).

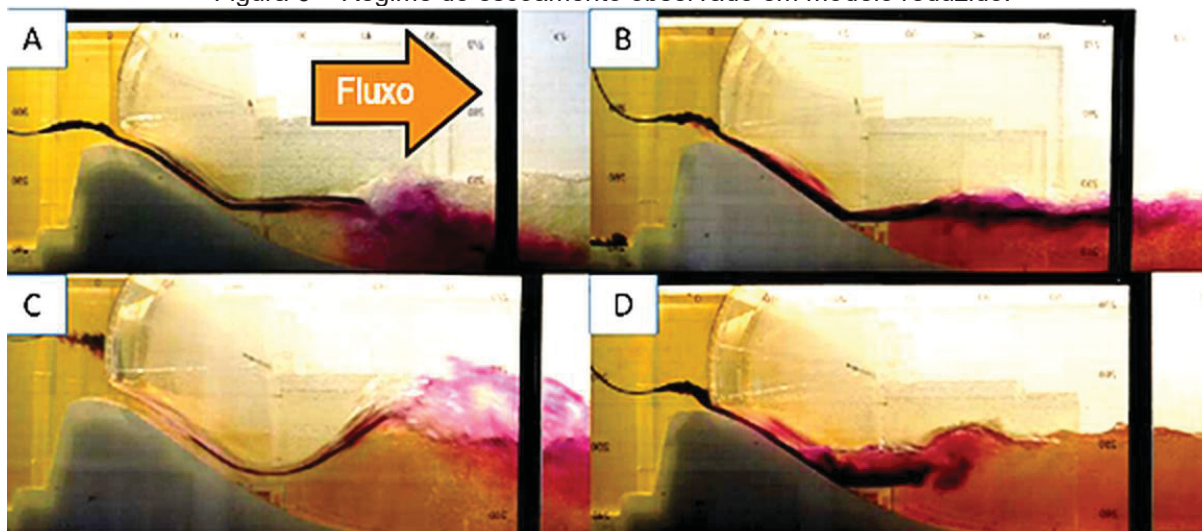
Segundo Orlins e Gulliver (2000) o MHR baseado no número Froude reproduz muito bem o fluxo de água na estrutura do vertedouro. Entretanto, não consegue representar o arraste de ar na bacia de dissipação e a transferência de gás através da interface bolha-água, não sendo representativo para simular a distribuição de “bolhas” de ar (incluindo o seu tamanho e quantidade de ar arrastado) bem como o tempo de permanência das “bolhas” no fluxo. Politano et al. (2012) também comenta que o MHR é limitado na representação da concentração do TDG.

3.1 CLASSIFICAÇÃO DO ESCOAMENTO E CURVA DE PERFORMANCE

A função dos defletores é mudar a trajetória do jato, direcionando-o de forma tangente à superfície da água. Ao evitar o mergulho em direção ao fundo da bacia de dissipação, diminui-se consideravelmente o transporte de “bolhas” para o fundo, reduzindo a dissolução dos gases incorporados ao escoamento devido as maiores pressões devido à profundidade. O comprimento e elevação dos defletores dependem das condições hidráulicas do empreendimento (características do vertedouro, níveis de jusante, regime hidrológico, entre outros), sendo que o projeto do defletor deve considerar sua elevação, comprimento, raio de transição e ângulo de lançamento.

A Figura 6 apresenta os regimes de escoamento observados em MHR para diferentes vazões vertidas e condições operativas da casa de força. O defletor da UHE Colíder foi projetado para ter o melhor desempenho para vazões defluentes de $3.200 \text{ m}^3/\text{s}$, sendo $1.600 \text{ m}^3/\text{s}$ vertidos e $1.600 \text{ m}^3/\text{s}$ turbinados (Andriolo M. V. et al., 2022), equivalente a um tempo de retorno de 10 anos, e o regime “B” ilustrado na Figura 6.

Figura 6 – Regime de escoamento observado em modelo reduzido.

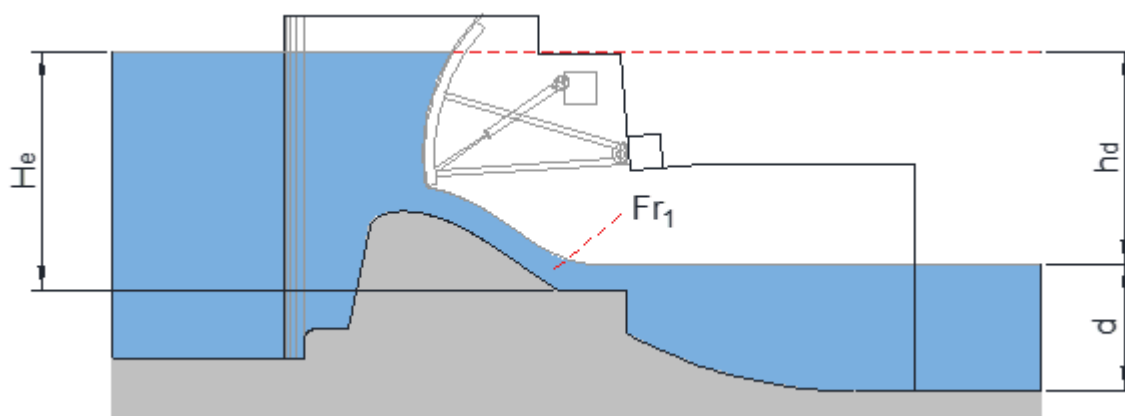


Legenda: (A) *Plunging flow* (B) *Skimming flow* (C) *Undular Jet* (D) *Surface Jump*. **Fonte:** Dettmer (2020).

Os estudos em MHR possibilitam a construção da curva de performance do defletor (Weber e Haug, 1999), (Andriolo M. V. *et al.*, 2022), mostrando o regime do escoamento para diferentes vazões vertidas e níveis de jusante, possibilitando a comparação de diferentes configurações de defletores, auxiliando na escolha dos defletores com melhor desempenho.

A Figura 8 apresenta a curva de performance do defletor da UHE Colíder, baseado em 131 ensaios em MHR (Andriolo M. V. *et al.*, 2022), sendo as variáveis utilizadas ilustradas na Figura 7.

Figura 7 – Variáveis de interesse utilizadas na Curva de performance da UHE Colíder.



Fonte: Andriolo M. V. *et al.* (2022).

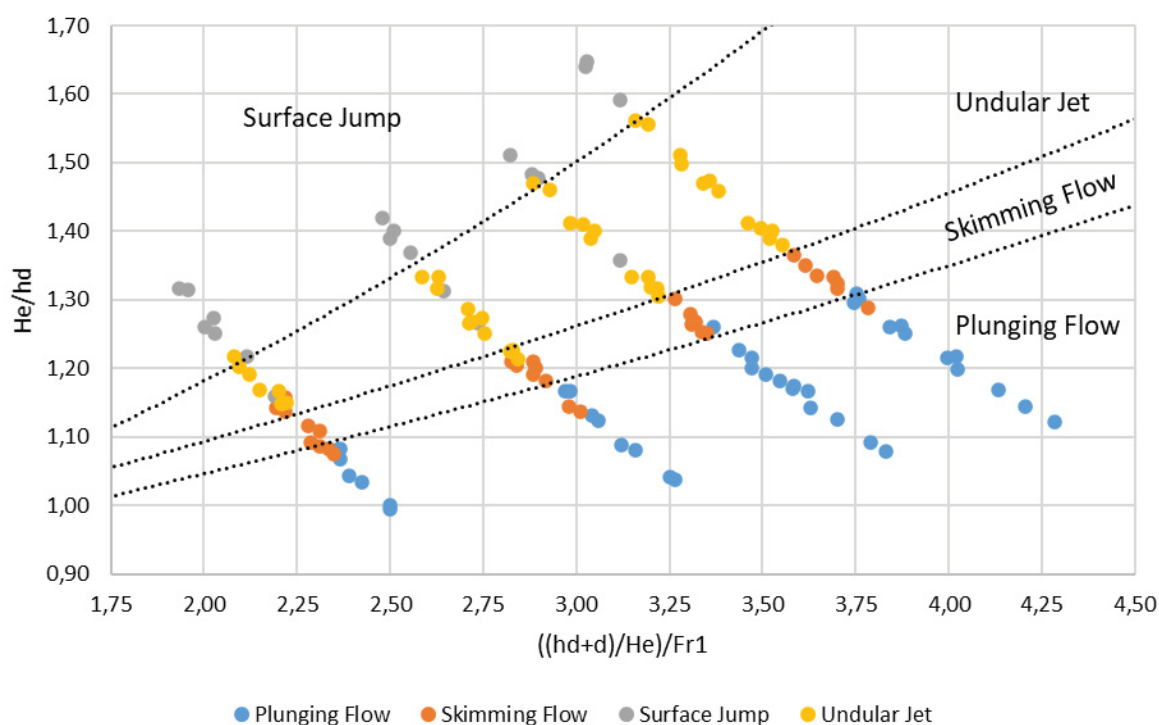
Onde H_e é a distância vertical entre o nível do reservatório e o defletor; Fr_1 é o número de Froude calculado sobre o defletor; hd é a distância vertical entre o nível do

reservatório e o nível de água a jusante e, d a distância entre o fundo da bacia de dissipação e o nível de água de jusante.

Os eixos x e y da Figura 8, adimensionais, utilizam dois termos para classificar o tipo do regime de escoamento. O primeiro termo, no eixo y , He/hd , mede a altura da água disponível em relação à elevação do defletor. O segundo termo, no eixo x , $[(hd+d)/He]/Fr_1$, mede a posição do defletor em relação ao nível a jusante, à abertura da comporta e à vazão. Quando os valores de He/hd se aproximam de 1, o defletor está acima da elevação da água a jusante e espera-se que o escoamento seja do tipo *plunging flow*.

O termo $(hd+d)/He$ mede a posição do defletor em relação ao nível da água a jusante; quanto mais próximo de 1, mais submerso o defletor se encontra. O termo Fr_1 é usado para considerar a vazão através das comportas radiais.

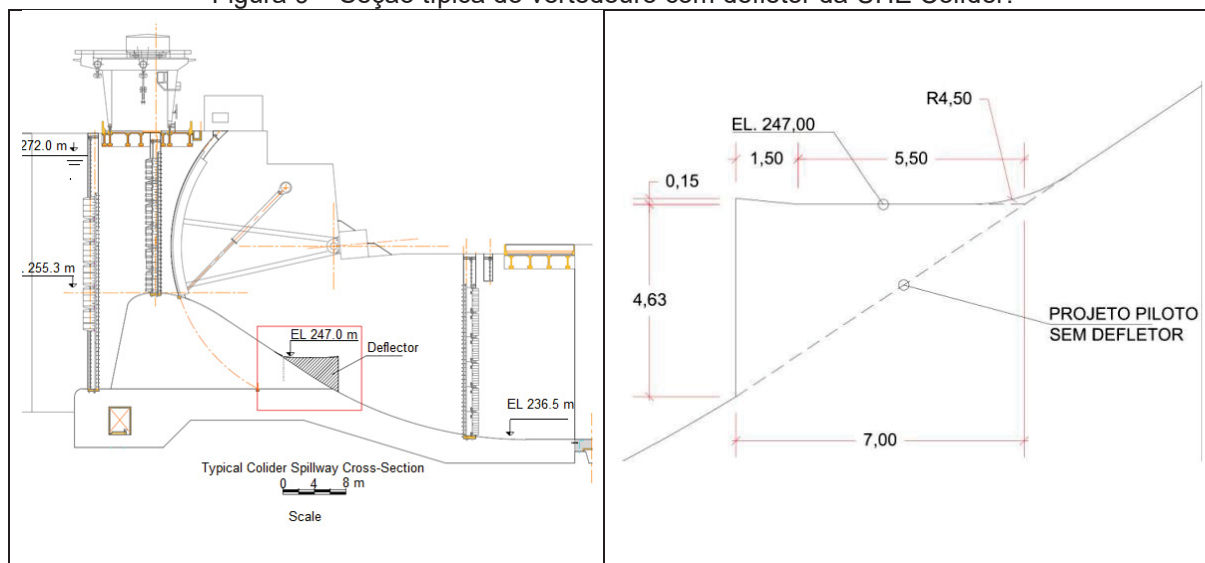
Figura 8 – Curva de performance para diferentes regimes de escoamento da UHE Colíder.



Fonte: Andriolo M. V. et al. (2022).

No caso da UHE Colíder, após avaliação do resultado dos ensaios em MHR e da curva de performance, foram selecionadas duas configurações de defletores para estudo em modelo computacional (Andriolo M. V. et al., 2022), (Politano et al., 2024), sendo a geometria selecionada apresentada na Figura 9.

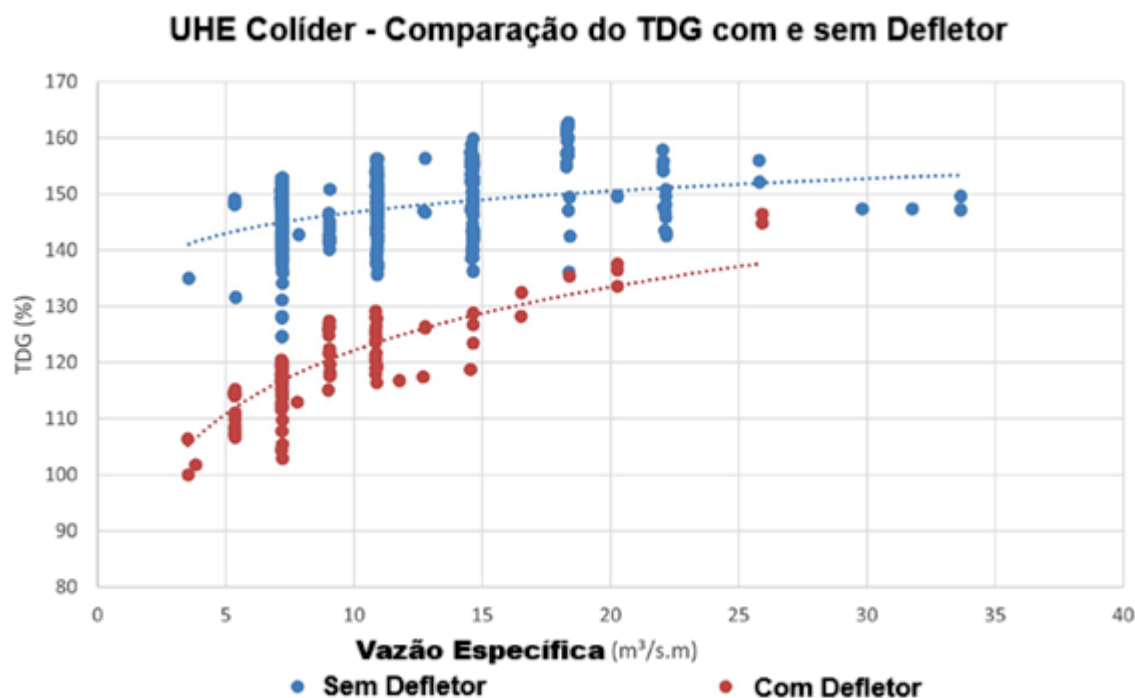
Figura 9 – Seção típica do vertedouro com defletor da UHE Colíder.



Fonte: Andriolo et al. (2023).

Andriolo et al. (2023) apresenta a performance do defletor da UHE Colíder, reproduzida na Figura 10, comparando o percentual de TDG antes e após a construção do defletor. Para escoamentos com vazão específica inferior a $5 \text{ m}^3/\text{s.m}$, equivalente a $240 \text{ m}^3/\text{s}$, a redução no percentual de TDG chega a 30 pontos percentuais, e para vertimentos superiores a $1.000 \text{ m}^3/\text{s}$, a redução do percentual de TDG é de aproximadamente 10 pontos percentuais.

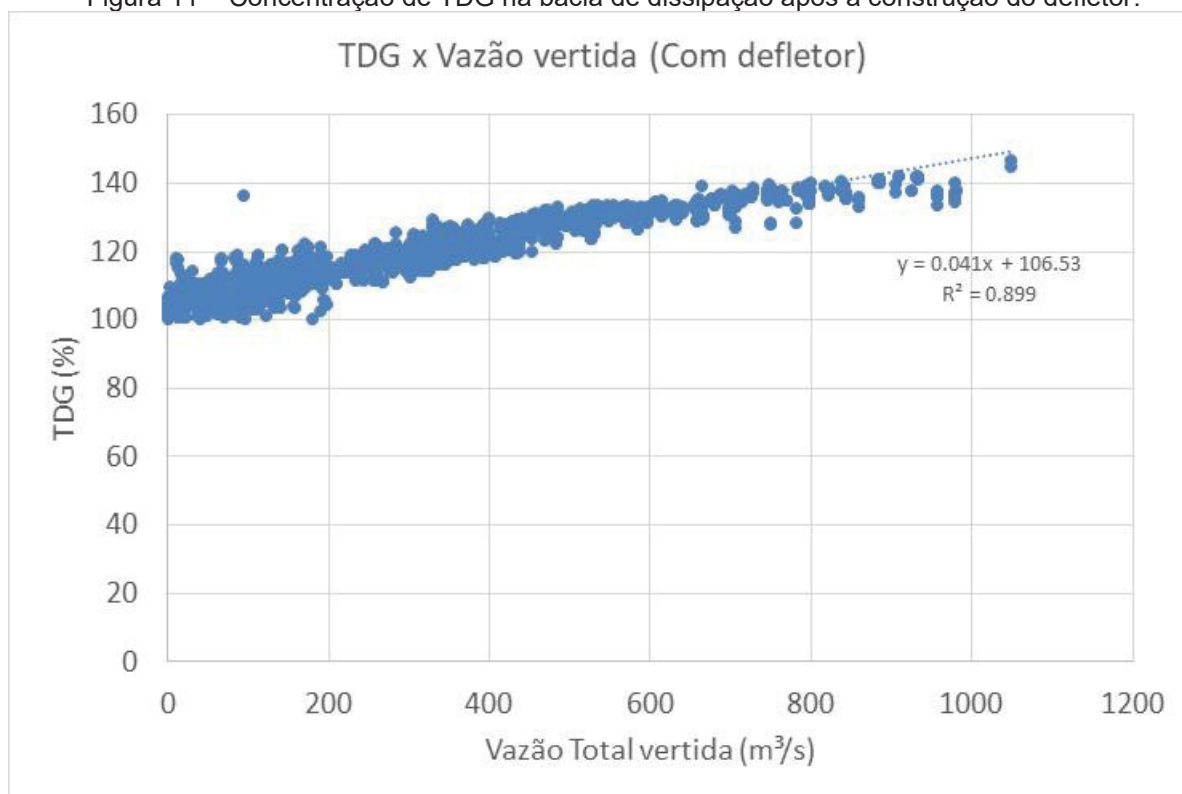
Figura 10 – Concentração de TDG na bacia de dissipação antes e após a construção do defletor.



Fonte: Andriolo et al. (2023).

A Figura 11 atualiza a Figura 10 com dados verificados até maio de 2025, considerando o percentual de TDG após a conclusão da construção do defletor no vertedouro da UHE Colíder. Essa figura permite comparar os valores de TDG previsto no modelo proposto nesta tese, e nos outros estudos apresentados, com os valores verificados em campo.

Figura 11 – Concentração de TDG na bacia de dissipação após a construção do defletor.

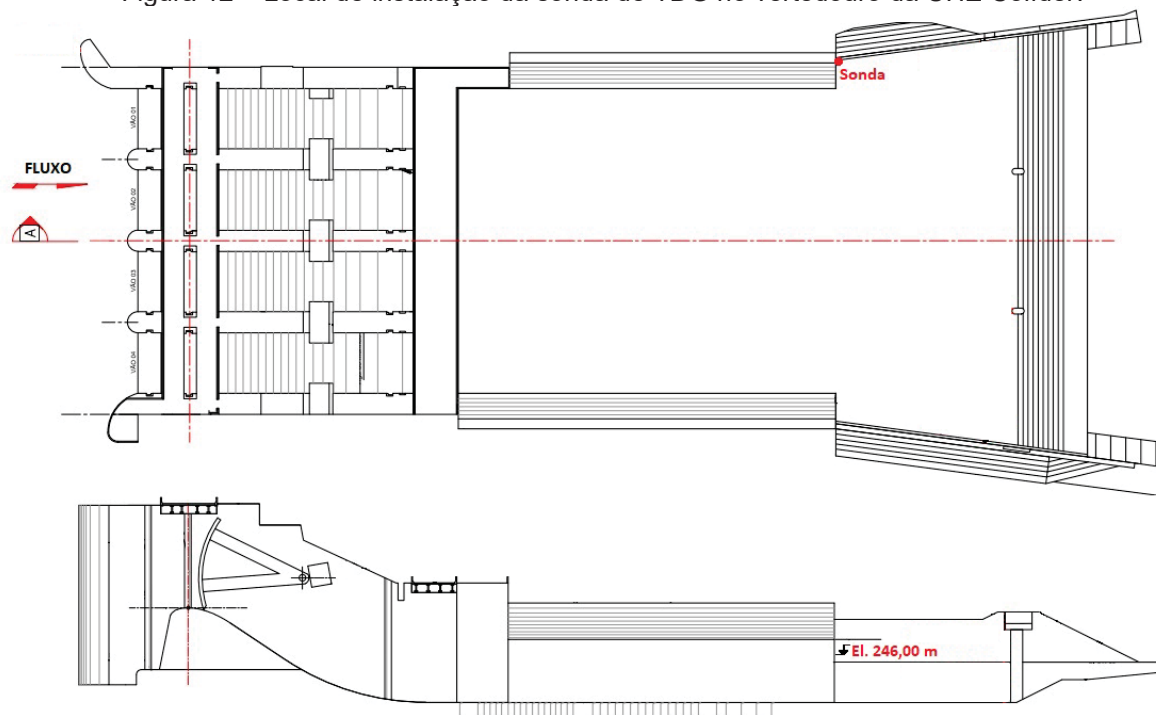


Fonte: Autor (2025).

3.1.1 Mensuração do TDG no vertedouro da UHE Colíder

Para mensurar a concentração de TDG no vertedouro da UHE Colíder foi instalada uma sonda multiparâmetro na bacia de dissipação, localizada conforme apresentada na Figura 12, sendo os dados lidos a partir de outubro de 2019. A sonda instalada, ilustrada na Figura 13, é da fabricante Hydrolab, modelo MS5, com faixa de medição de 533 mbar a 1.866 mbar, com precisão de $\pm 1,9$ mbar e resolução de 1,3 mbar. Os valores de TDG medidos pelas sondas são fornecidos em pressão e depois transformados em porcentagem, relativo à pressão atmosfera local.

Figura 12 – Local de instalação da sonda de TDG no vertedouro da UHE Colíder.



Fonte: Autor (2025).

Figura 13 – Sonda HydroLab MS5 utilizada para medição de TDG em campo.



Fonte: Fabricante HydroLab (2022).

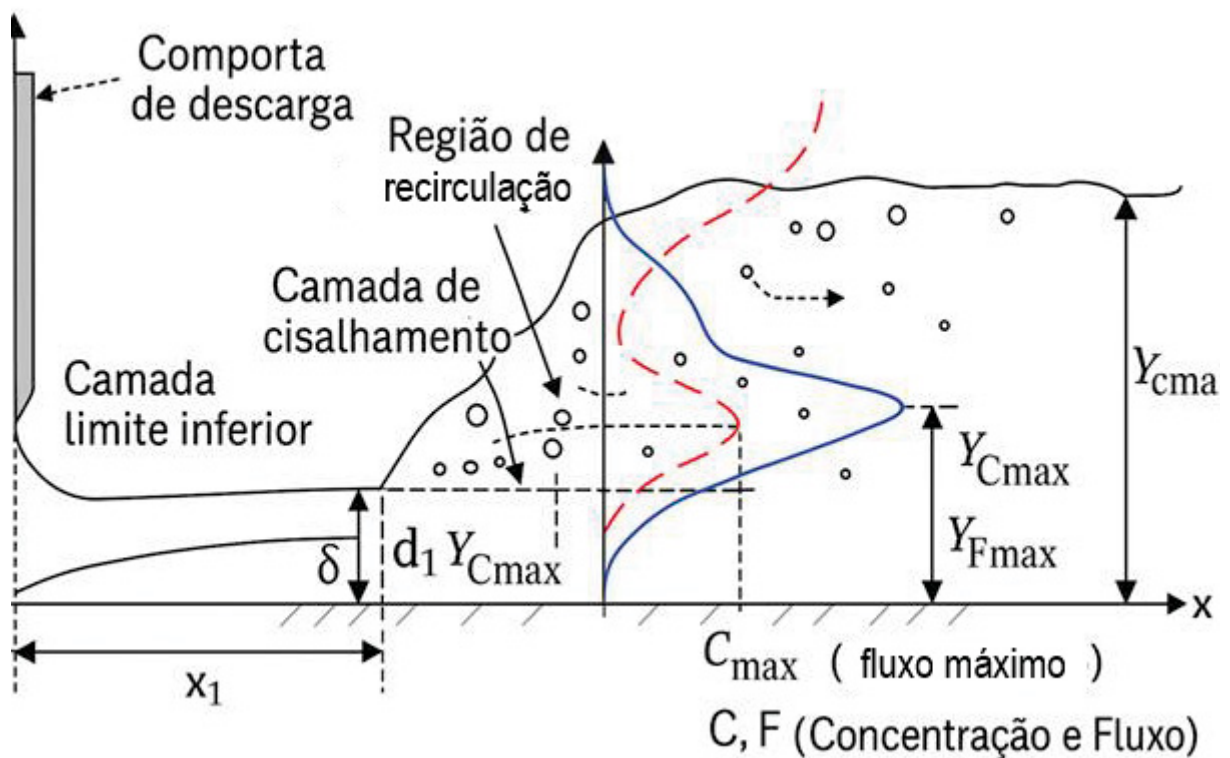
3.2 MENSURAÇÃO DE AR EM MHR

O mecanismo de arraste e transporte das “bolhas” de ar no ressalto hidráulico são processos não completamente compreendidos, dominados pelos efeitos de tensão superficial, cujos estudos analíticos e numéricos são complexos devido ao grande número de equações envolvidas (número de Froude, Reynolds e Morton). O fluxo aerado do ressalto hidráulico é dividido em três regiões: a camada de cisalhamento turbulento com formação de pequenas “bolhas” e alto teor de ar, a região “*boiling flow*” caracterizada pelo desenvolvimento de vórtices de grande escala e

coalescência de “bolhas” e a camada de espuma (*foam layer*) na superfície livre (Chanson, 2006).

Chanson (2007) estudou a entrada de ar no ressalto hidráulico e os efeitos de escala, utilizando dois modelos hidráulico reduzidos, com comprimento de 3,2 m e largura de 0,25 m e 0,5 m, instrumentado com uma sonda de condutividade de ponta única. A Figura 14 ilustra o ressalto hidráulico parcialmente desenvolvido, com pico de concentração de ar se encontrando no início do ressalto hidráulico e diminuindo conforme aumenta a distância ($x-x_1$).

Figura 14 – Entrada de “bolhas” de ar em um ressalto hidráulico.



Fonte: Chanson (2007).

Sendo Y_{Cmax} a altura da concentração máxima; Y_{Fmax} a altura do fluxo máximo; C_{max} a concentração máxima.

Na região da camada limite, a fração de ar no escoamento é expressa pela Equação (3.1) (Chanson, 2007), (Wang, Felder e Chanson, 2014):

$$C = C_{max} \exp \left(-\frac{1}{4D^{\#}} \frac{\left(\frac{y}{d_1} - \frac{Y_{Cmax}}{d_1} \right)^2}{\frac{x-x_1}{d_1}} \right), \quad (3.1)$$

sendo C_{max} (Equação (3.2)) a máxima concentração de ar na região da camada limite turbulenta; $D^\#$ a difusividade; $D^\# = D_t/D_d d_1$; D_t a difusividade turbulenta:

$$C_{max} = A \exp \left[-B \left(\frac{x - x_1}{d_1} \right) \right], \quad (3.2)$$

$$\begin{aligned} A &= 0,436 + 0,014 Fr_1, \\ B &= 0,911 \exp(-0,335 Fr_1). \end{aligned} \quad (3.3)$$

O tamanho das “bolhas” de ar pode ser mensurado a partir das análises de sinais de sondas de condutividade de duas pontas, que fornecem também informações quanto à concentração de ar e à velocidade média do fluxo aerado (Chanson, 1996).

Nesse estudo as sondas de condutividade de ponta dupla mediram uma característica de comprimento da “bolha” de ar e não o seu diâmetro, pois as “bolhas” de ar não são necessariamente esféricas e não cruzam a sonda obrigatoriamente ao longo do diâmetro. Fotografias de vídeo de alta velocidade podem mostrar que as “bolhas” têm uma grande variedade de formas, principalmente as “bolhas” grandes. Considerando “bolhas” pequenas, é possível estimar a distribuição de probabilidade do diâmetro da “bolha” a partir dessa característica de comprimento da “bolha” obtida pela sonda dupla de condutividade. Essa probabilidade assume que todas as “bolhas” são esféricas.

De acordo com Chanson (1996) para o ressalto hidráulico com jato de água vertical e jato de água turbulento lançado na atmosfera, a distribuição máxima do tamanho da “bolha” pode ser expressa pela fórmula analítica apresentada na Equação (3.4):

$$d_m \approx \sqrt[3]{\frac{2\sigma(We)_c}{\rho_w} \frac{\pi}{K^2} \left\{ \frac{x^2}{(V_0)^2 \exp \left[-2 \left(\frac{K(y - y_{50})}{x} \right) \right]^2} \right\}}, \quad (3.4)$$

onde d_m (m) é o tamanho máximo da “bolha” na camada de cisalhamento turbulenta; σ (N/m) a tensão superficial; $(We)_c$ (adimensional) o número crítico de Weber; K (adimensional) a constante determinada pela configuração do fluxo; y_{50} a posição onde $V = V_0/2$.

No projeto de pesquisa intitulado “Metodologia para Modelagem Computacional de TDG na Água em Fluxos Efluentes de Vertedouros” (LACTEC, 2020), foi possível mensurar em modelo físico (escala 1:15) a concentração de ar ao longo do escoamento, sendo utilizada sondas de escoamento bifásico por condutividade construídas pelo próprio laboratório.

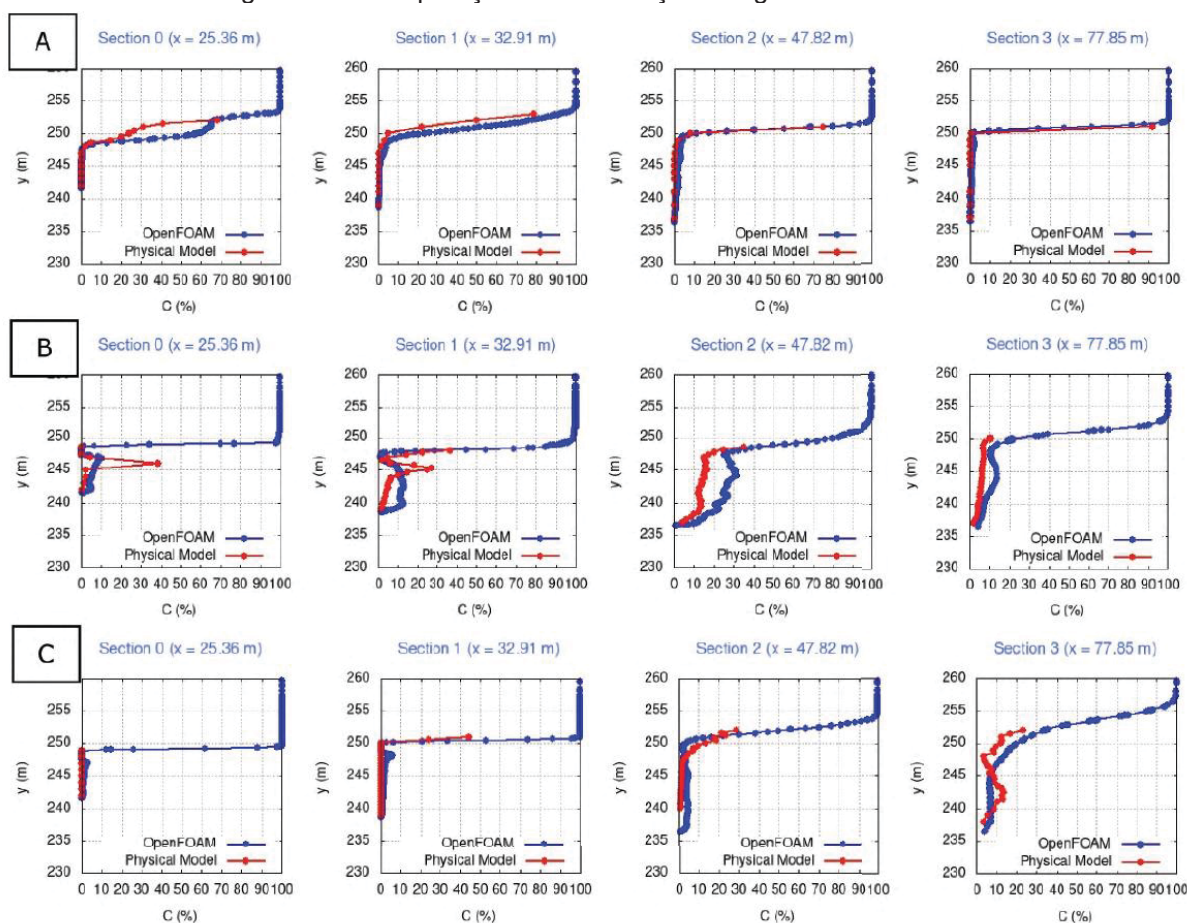
Takenobu et al. (2022) apresenta um comparativo entre a relação ar-água mensurado em modelo físico com o calculado em modelo computacional (CFD) considerando a média do parâmetro *alpha.water* (Figura 15), extraído dos estudos do Lactec (Dettmer, 2022). A Tabela 2 apresenta os parâmetros de vazão específica (q), nível de jusante (TWE), e regime de escoamento, para os três ensaios realizados.

Tabela 2 – Parâmetros dos Testes 1, 2 e 3.

Teste	q (m ³ /(s.m))	TWE (m)	Regime do Jato
1	6,25	250,88	<i>Surface Jump</i>
2	33,33	250,19	<i>Plunging Flow</i>
3	33,33	253,21	<i>Undular Jet</i>

Fonte: Dettmer (2022).

Figura 15 – Comparação entre a relação ar-água em três testes.



Fonte: Takenobu et al. (2022).

4 MODELAGEM MATEMÁTICA DO TDG

Segundo Orlins e Gulliver (2000), o MHR baseado no número de Froude reproduz muito bem o fluxo de água na estrutura do vertedouro. Entretanto, não consegue representar o arraste de ar para a bacia de dissipação e a transferência de gás através da interface bolha-água, não sendo representativo para simular a distribuição de “bolhas” de ar (incluindo o seu tamanho e quantidade de ar arrastado) bem como o tempo de permanência das “bolhas” no fluxo. Politano et al. (2012) também comenta que o MHR é limitado na representação da concentração do TDG.

A performance do projeto de um defletor normalmente é avaliada em modelo físico (MHR) e com métodos computacionais (CFD - *Computational Fluid Dynamics*). Entretanto, estes dois modelos não são capazes de verificar o efeito das “bolhas” no escoamento, sendo que a modelagem matemática utilizando um modelo em duas fases (*two-phase*) vem sendo empregada (Politano *et al.*, 2016).

Desta forma, a modelagem matemática computacional do TDG se torna uma ferramenta muito útil para entender os fenômenos que levam à supersaturação do gás e à alta concentração de TDG a jusante de aproveitamentos hidráulicos. Conforme Politano; Carrica; Weber (2009), a concentração de TDG depende de processos extremamente complexos, como o arraste do ar pelo fluxo de água no vertedouro (pré-arraste), que ocorre quando o jato impacta a lâmina de água (na bacia de dissipação ou no próprio rio).

A concentração de gás dissolvido resultante de um vertimento depende da dispersão da “bolha” no fluxo, da distribuição do diâmetro da “bolha”, profundidade de penetração, a capacidade de retenção do gás, do tempo de permanência e da transferência de massa (Bin, 1993).

4.1 MODELO PREDITIVO DO TDG

De acordo com Geldert, Gulliver e Wilhelms (1998), o primeiro modelo preditivo das concentrações de gases dissolvidos a jusante de vertedouros foi desenvolvido por Roesner e Norton em 1971. O modelo foi desenvolvido com base em dados de vertedouros de baixa carga e com controle por comportas. A forma geral da equação de concentração é dada pela Equação (4.1):

$$C_t = C_s + (C_i - C_s) \exp (-Kt), \quad (4.1)$$

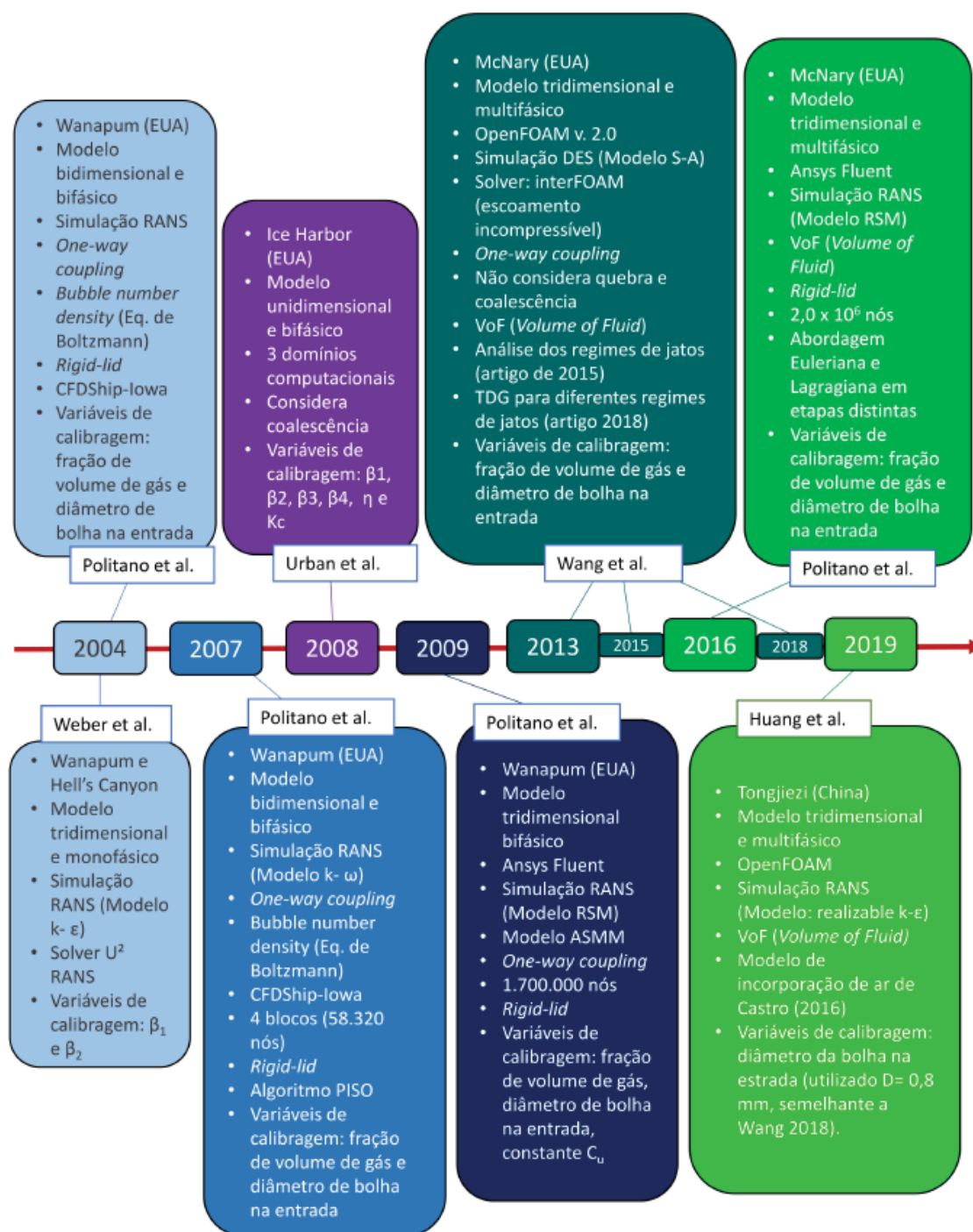
sendo C_t (%) a concentração total de saturação de gás a jusante; C_s (%) a concentração de saturação à profundidade considerada; C_i (%) a concentração de gás dissolvido no reservatório; t (s) o tempo de residência no qual os gases são dissolvidos; K (constante) o coeficiente empírico de transferência e arrastamento de ar.

Segundo Wang, Politano e Weber (2018) os modelos desenvolvidos por Ead and Rajaratnam em 2002, (Urban, Gulliver e Johnson, 2008) a distribuição de “bolhas” na bacia de dissipação não é totalmente simulada, mas sim estimada com base em correlações empíricas.

Os primeiros estudos com abordagem bifásica foram desenvolvidos por Politano et al. (2007) utilizando a simulação RANS, modelo $k-\omega$, e seu próprio solver CFDShip-Iowa para avaliação de TDG a jusante da bacia de dissipação da UHE Wanapum.

A Figura 16 apresenta os principais trabalhos sobre a modelagem matemática do TDG a jusante de vertedouros, desenvolvidos por diversos autores a partir do ano de 2004 (LACTEC, 2020).

Figura 16 – Resumo das principais considerações computacionais da simulação de TDG.



Fonte: LACTEC (2020).

O modelo matemático proposto por Wang, Politano e Weber (2018) considera o escoamento multifásico composto por três fases: Fase ar, fase água e fase “bolha”. As fases ar e água são consideradas um escoamento multifásico estratificado e normalmente utiliza a modelagem VOF (*Volume of Fluid*) para o cálculo dessas duas fases. Já a fase “bolha” (transporte e dissolução) é calculada com uma abordagem Euleriana, ou seja, é assumida a simplificação de que a presença de “bolhas” não

influi no escoamento ar/água circundante. Neste modelo, a produção de TDG é dada pela transferência de massa da fase “bolhas” para o líquido. A fase “bolha” é introduzida manualmente através das condições de contorno do domínio computacional. Além da fase “bolha”, deve ser indicar o diâmetro das “bolhas” que serão introduzidas no escoamento pelas condições de contorno.

Este modelo de fluxo bifásico foi implementado no código-fonte aberto OpenFOAM (*Open Source Field Operation and Manipulation*) utilizado para prever regimes do escoamento do vertedouro e do total de gás total dissolvido a jusante de uma barragem. Um método de volume de fluido foi utilizado para rastrear a interface entre ar e água. Uma abordagem Euleriana foi utilizada para prever a distribuição de “bolhas” no escoamento. Um modelo de simulação de DES (*Detached Eddy Simulation*) foi utilizado para simular a turbulência. O tamanho da “bolha” foi simulado utilizando a equação de transporte de densidade de número de “bolhas”. Uma equação de transporte escalar que inclui a transferência de massa entre “bolhas” e água foi usada para prever a concentração total de gás dissolvido.

O modelo computacional proposto por Huang et al. (2019) para modelar o TDG em escoamentos a jusante de vertedouros é composto por três fases: água, ar e “bolhas” de ar na água. A interface entre ar e a mistura ar-água é solucionada com o modelo VOF, a interface entre a formação de “bolha” e a mistura ar-água é solucionada com o modelo de fluxo bifásico unidirecional Euleriano onde as “bolhas” não afetam o fluxo do líquido. Este modelo utiliza um termo “sumidouro” para minimizar o comportamento não físico da entrada de grandes “bolhas” com tamanho próximo da malha.

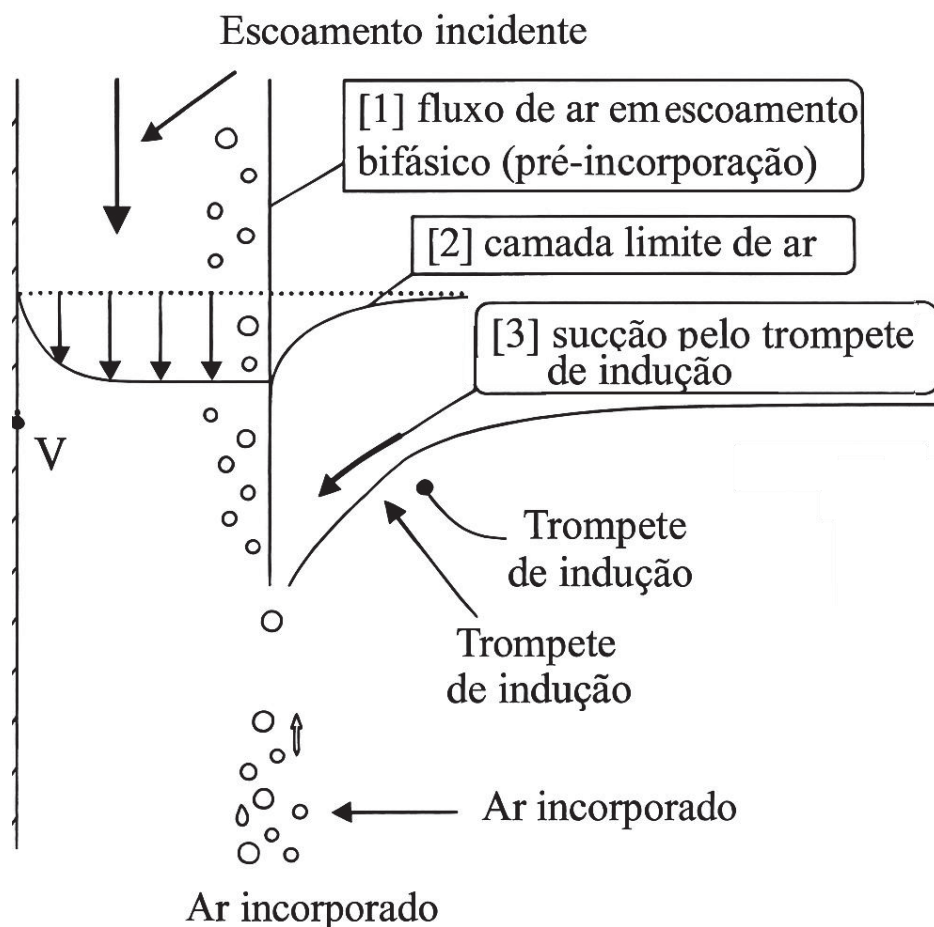
Huang et al. (2019) em seu trabalho adicionou ao modelo de Wang, Politano e Weber (2018), um método para o cálculo da fonte de ar arrastado e fração volume de gás da fase “bolha”. Desta forma, é necessário apenas a calibração do diâmetro da “bolha”.

O arraste de ar é um processo complexo muito controlado pela velocidade do fluxo (Bin, 1993), cuja presença do ar não influencia muito a distribuição de velocidade (Chanson, 1996). Entretanto a turbulência influencia o tamanho das “bolhas” que é um importante parâmetro na previsão da transferência de gás entre a água e o ar.

Segundo Chanson (1996) existem três mecanismos para incorporação de ar no escoamento, ilustrados na Figura 17: 1) Fluxo de ar em escoamento bifásico (*two-*

phase flow air flux); 2) incorporação de ar na camada limite (*air boundary layer*); 3) sucção (aspiração) de ar induzida (*suction by induction trumpet*).

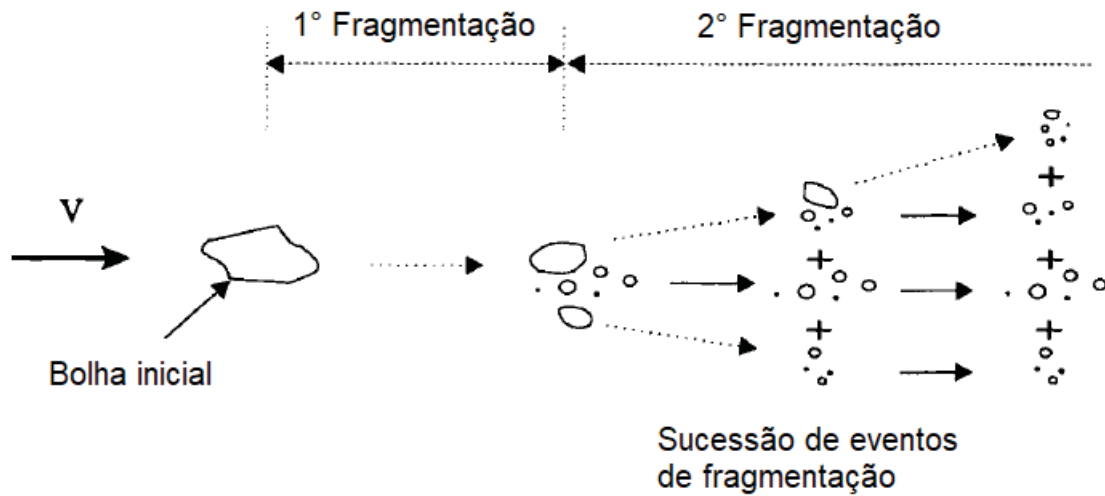
Figura 17 – Três mecanismos de incorporação de ar.



Fonte: Chanson (1996).

Em um escoamento turbulento uma “bolha” com diâmetro maior que o diâmetro máximo $(d_{ab})_{max}$, é fragmentada em “bolhas” menores, levando a uma distribuição de diâmetro de “bolhas” com diâmetros diferentes, cujo processo termina quando todos os fragmentos possuem diâmetro menor que o $(d_{ab})_{max}$, ou as condições do escoamento se alteram, conforme ilustrado na Figura 18.

Figura 18 – Sucessão de eventos de fragmentação das “bolhas”.



Fonte: Chanson (1996).

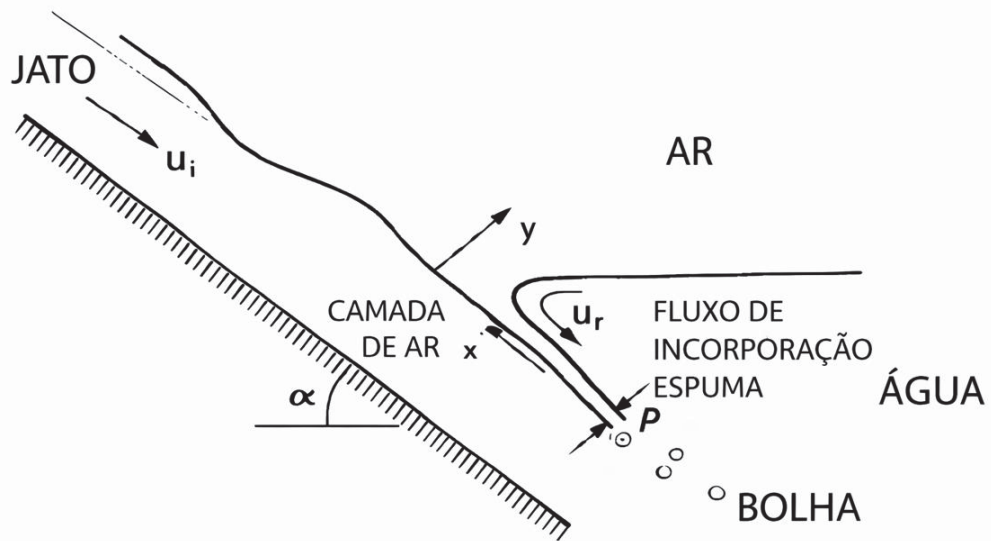
Segundo Chanson (1996), o diâmetro máximo da “bolha” é dado pela Equação (4.2):

$$(d_{ab})_{max} \sim \sqrt[3]{\frac{2\alpha(We)_c}{\rho_w \left(\frac{\partial V}{\partial y}\right)^2}} \quad (4.2)$$

onde We é o número de Weber ($We = \frac{\rho v^2 y}{\sigma}$); σ a tensão superficial; ρ_w a densidade da água; y a direção perpendicular ao fluxo.

Sene (1988) propõe o mecanismo de incorporação de ar para jatos de alta velocidade, ilustrado na Figura 19. Neste modelo, a camada de ar entra em movimento devido ao impacto do jato de água, que produz uma sucessão de “bolhas” a jusante devido à instabilidade gerada na superfície da água, sendo provável que a camada de ar primeiro se estabeleça em regiões localizadas ao longo da interface entre o jato e os fluxos de incorporação de espuma. Em velocidades de jato muito altas, essas regiões se unem para formar uma camada contínua ao longo da interface.

Figura 19 – Mecanismo de incorporação de ar proposto para impacto de jato em alta velocidade.



Fonte: Sene (1988).

A taxa de incorporação de ar é dada pela Equação (4.3):

$$q_a = \int_0^{\beta_M} \mu_e y dy = \frac{1}{3} \sqrt{\frac{2\mu_a}{\rho g \sin \alpha}} U_i^{3/2}, \quad (4.3)$$

$$\beta_M^2 = \frac{2\mu_a}{\rho g \sin \alpha} \quad \text{para } \frac{u_r}{U_i} \ll 1, \quad (4.4)$$

em que β_M é a espessura da camada de ar entre o jato e a espuma; μ_e é a viscosidade efetiva na região aerada; μ_a a viscosidade do ar; y a coordenada vertical adimensional; ρ a densidade da água; g a aceleração da gravidade; α o ângulo de impacto do jato; U_i a velocidade de impacto do jato; u_r a escala de velocidade de recirculação do escoamento.

Sene (1988) apresenta a Equação (4.5) que calcula a relação entre o volume de ar e volume de água em relação ao número de Froude:

$$q_a/q_w \sim 0,05 Fr_i, \quad (4.5)$$

sendo q_a a vazão específica de entrada de ar; q_w a vazão específica de água; Fr_i o número de Froude ($Fr_i = \frac{v}{\sqrt{gY}}$).

Em estruturas hidráulicas, a concentração de ar é um parâmetro de projeto importante que, por exemplo, pode determinar a altura necessária das paredes

laterais de uma bacia de dissipação. A análise dimensional sugere que a quantidade de incorporação de ar no ressalto hidráulico pode ser expressa pela Equação (4.6) (Wood, 1991 *in* (Chanson, 2006)):

$$\frac{Q_a}{Q_w} = K_1 \frac{(V_1 - V_e)^2}{gY_1}, \quad (4.6)$$

sendo Q_a a vazão de ar (m³/s); Q_w a vazão de água (m³/s); K_1 o coeficiente empírico de arraste de ar (adimensional); V_1 a velocidade do escoamento na seção 1 (regime supercrítico) (m/s); V_e a velocidade incipiente de incorporação de ar no ressalto hidráulico (m/s).

A Tabela 3 elaborada por Chanson (2006) apresenta um resumo dos estudos desenvolvidos de incorporação de ar no ressalto hidráulico.

Tabela 3 – Quantidade de incorporação de ar no ressalto hidráulico.

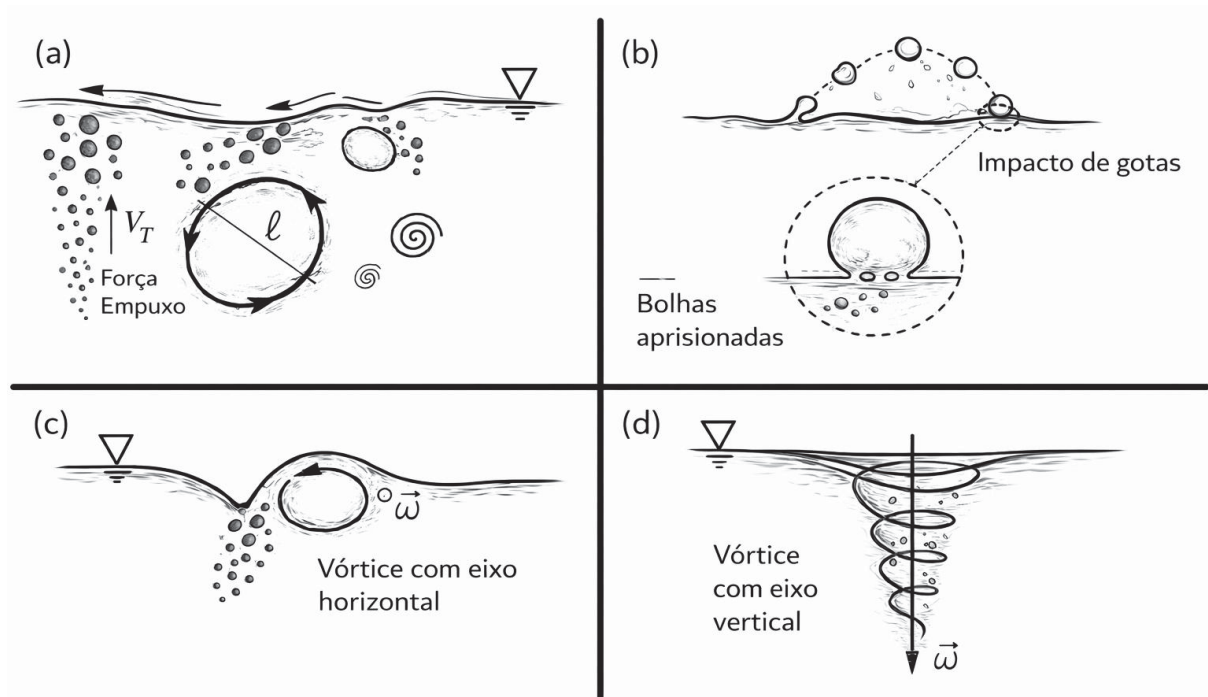
Autor	Ano	Geometria	Q_a/Q_w	Comentários
Kalinski e Robertson	1943	Ressalto hidráulico em tubo circular horizontal	$0,0066(Fr_1 - 1)^{1,4}$	Dados de modelo. $2 < Fr_1 < 25$
Wisner	1965	Ressalto hidráulico em conduto retangular	$0,014(Fr_1 - 1)^{1,4}$	Dados de protótipo. $5 < Fr_1 < 25$
Rajaratnam	1967	Ressalto hidráulico em canal retangular	$0,018(Fr_1 - 1)^{1,245}$	Dados de modelo. $2,4 < Fr_1 < 8,7$
Rabben et al.	1983	Ressalto hidráulico em tubos retangulares horizontais	$0,03(Fr_1 - 1)^{0,76}$	Dados de modelo. Razão de abertura da comporta: 0,25

Fonte: Chanson (2006).

A modelagem de arrastamento de ar proposta por Castro, Li e Carrica (2016), considera que o arrastamento de ar para o escoamento pode ocorrer de forma direta, com impactos de gotas ou quebra de ondas, ou pelos vórtices próximos à superfície livre. A Figura 20 apresenta exemplos desses processos, sendo que: a) representa a incorporação de “bolhas” pela ação de vórtices que deformam a superfície livre acima de um limite e a atuação da força de empuxo nas “bolhas” filtrando o tamanho de

“bolhas”; b) representa o impacto por gotas que ao cair novamente no escoamento aprisionam “bolhas”; c) e d) apresentam os vórtices com eixo horizontal e vertical, respectivamente que deformam a superfície livre e arrastam “bolhas” para o interior do escoamento.

Figura 20 – Processos físicos que conduzem ao aprisionamento de ar no escoamento.



Fonte: Castro, Li e Carrica (2016).

A formação das “bolhas” foi considerada em três eventos: 1) a formação da “bolha” na superfície; 2) o arraste de “bolhas” para o escoamento contra o empuxo; e 3) a quebra ou coalescência das “bolhas”. O modelo de Castro, Li e Carrica (2016) calcula a probabilidade da ocorrência de cada um desses eventos e os combina em uma única fonte volumétrica de ar arrastado.

O fluxo de massa total de ar arrastado pela área superficial é calculado pela Equação (4.7):

$$J = \int_{l_{min}}^{l_{max}} J_l(l) dl, \quad (4.7)$$

onde $J_l(l)dl$ (kg/m²) é a contribuição do fluxo dos vórtices dentro de dl com tamanho l para o fluxo total:

$$J_l(l) = 0,8413 \frac{f_l}{l^4} P_B \frac{u_l^3 l^3}{g z_e} I_Q, \quad (4.8)$$

$$f_l = \frac{KL}{((KL)^2 + c_L)^2}, \quad (4.9)$$

sendo f_l o fator de correção da fase líquida (adimensional); P_B (adimensional) a probabilidade de encontrar um vórtice $V(l)$ de tamanho l com energia suficiente para formar uma “bolha” de diâmetro D_b pela quebra da superfície; u_l (m/s) a raiz do valor quadrático médio (RMS) da velocidade dos vórtices de tamanho l ; z_e (m) o comprimento inicial para o arrastamento de um único vórtice; I_Q (adimensional) a fonte de arrastamento de um único vórtice; K (1/m) o número da onda dado por $K = 2\pi/l$; c_L (adimensional) a constante espectral turbulenta, adotado valor de 6,78; L (m) a escala de comprimento característica dado por $L = k3/2/\varepsilon$.

4.2 CLUSTERIZAÇÃO DE “BOLHAS”

Os fenômenos de transferência de massa ocorrem através da interface das “bolhas”, sendo sua estimativa dependente da compreensão do movimento das “bolhas” na fase líquida e do movimento da fase líquida ao redor das “bolhas”. Nesse sentido, a área interfacial e o formato da interface são os parâmetros fundamentais para a modelagem do fenômeno. De modo geral, a turbulência interfacial promove intensa transferência de massa (nesse sentido, a interação de micro redemoinhos e partículas leva a um aumento na intensidade da turbulência na fase de massa (Besagni, Inzoli e Ziegenhein, 2018).

A clusterização de “bolhas”, ou agrupamento, é a formação de regiões mais densas em “bolhas” dentro da fase líquida, ou seja, as “bolhas” não se distribuem uniformemente. Esses agrupamentos são dinâmicos, com “bolhas” entrando e saindo de clusters conforme as flutuações turbulentas do escoamento.

Segundo Risso (2018), a velocidade média das bolhas dentro de um cluster de “bolhas” é menor do que a de uma bolha isolada e diminui à medida que a fração volumétrica de gás aumenta. Este fenômeno é atribuído a um mecanismo de impedimento causado pelo fluxo de retorno necessário para equilibrar o arrastamento do fluxo pelas “bolhas”.

Em simulações DNS (*Direct Numerical Simulation*), a interação entre duas “bolhas” esféricas ascendentes, indica que a configuração estável ocorre quando

ambas permanecem lado a lado (Hallez & Legendre 2011), para número de Reynolds entre 50 e 500. A formação de aglomerados horizontais de “bolhas” também foi observada tanto em simulações de escoamento potencial (Smereka 1993, Yurkovetsky & Brady 1996, Sangani & Didwania 2006) quanto em DNS envolvendo “bolhas” esféricas com $Re = 91,5$ (Esmaeeli & Tryggvason 2005). Além disso, verificou-se que os intervalos de tempo entre a passagem consecutiva de duas “bolhas” por um mesmo ponto seguem uma distribuição de Poisson em escoamentos com clusters homogêneos de “bolhas”, para número de Reynolds entre 600 e 800 e frações volumétricas de ar entre 0,5% e 13% (Risso & Ellingsen 2002, Almeras *et al.* 2015), indicando que as posições das “bolhas” são estatisticamente independentes.

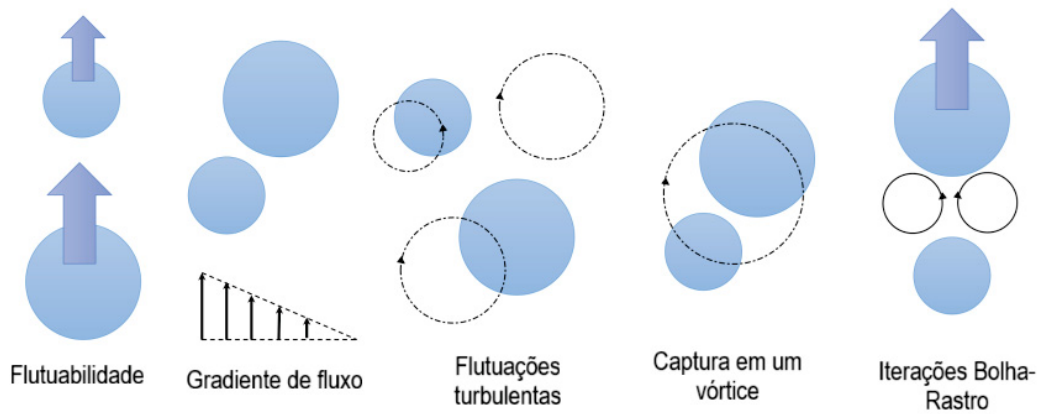
A coalescência de “bolhas” ocorre quando duas bolhas colidem e se fundem. Em regimes de fluxo homogêneo, a análise pode ser restrita a colisões binárias, já que interações envolvendo três ou mais “bolhas” têm probabilidade muito menor. Para que duas “bolhas” colidam, deve existir diferença de velocidade relativa entre elas.

As colisões podem ocorrer principalmente devido a cinco mecanismos (Figura 21):

- i) diferentes velocidades de ascensão causadas pela flutuabilidade (*buoyancy*);
- ii) gradientes de velocidade média no escoamento (*shear stresses*);
- iii) flutuações turbulentas da fase contínua (*turbulent fluctuation*);
- iv) captura de “bolhas” por estruturas de um vórtice (*eddy capture*); e
- v) interações entre “bolhas” e o rastro deixado por outra “bolha” (*wake capture*).

Uma colisão pode resultar em coalescência ou rebote. Assim, ao modelar a taxa de coalescência, é necessário considerar tanto a frequência de colisões quanto a eficiência com que essas colisões levam à fusão das “bolhas” (Besagni, Inzoli e Ziegenhein, 2018).

Figura 21 – Mecanismos de Coalescência.



Fonte: Besagni, Inzoli e Ziegenhein (2018).

Jones e Zuber (1975), demonstram que a função densidade de probabilidade (PDF) das flutuações na fração de vazios pode ser utilizada como um critério objetivo e quantitativo para discriminar os três padrões dominantes de escoamento bifásico: borbulhante (*bubbly*), lento (*slug*) e anular (*annular*). Em uma descrição conceitual, os autores ilustraram que, para um escoamento em água, a PDF de fração de vazios tende a seguir uma distribuição normal gaussiana.

Herrero, Guido Lavallo e Clausse (1995) apresentam a seguinte expressão para a probabilidade de colisão entre “bolhas”:

$$T(u, v)dudv = \frac{\pi v_r(u, v)[R(u)R(v)]^2}{1 - \chi} f(u)f(v)dudv, \quad (4.10)$$

em que $T(u, v)$ é a taxa de colisão entre partículas de velocidades u e v ; n a densidade numérica; $v_r(u, v)$ a velocidade relativa; $R(u)$ e $R(v)$ os raios das partículas ou “bolhas”; χ o fator de correção do volume ocupado e $f(u)$ a função distribuição de probabilidade.

Considerando a distribuição isotrópica de velocidade com “bolhas” semelhantes ($R(u) \approx R(v)$), meio diluído ($1 - \chi \approx 1$) e a função de distribuição normalizada, é possível simplificar a Equação (4.10) resultando na Equação (4.11):

$$T(u, v) = 4nR^2\langle u_r \rangle f(u)f(v), \quad (4.11)$$

onde $f(u)$ e $f(v)$ são distribuições Gaussianas.

Supondo uma distribuição normal gaussiana tem-se que a taxa média de colisão por unidade de volume $\langle T \rangle$ (1/s):

$$\langle T \rangle = \frac{8nR^2\sigma_u}{\sqrt{\pi}}, \quad (4.12)$$

sendo n a densidade de “bolhas” (1/m³); R o raio médio das “bolhas” (m) e σ_u o desvio padrão de velocidade.

A fração de vazios α_g representa a fração volumétrica de ar V_{ar} em uma célula da malha $V_{célula}$:

$$\alpha_g = \frac{V_{ar}}{V_{célula}}. \quad (4.13)$$

A variação da fração de vazios pode ser interpretada como:

$$\alpha'_g(t) = \alpha_g(t) - \overline{\alpha_g}, \quad (4.14)$$

$$\alpha'_g(x, y, z) = \alpha_g(x, y, z) - \overline{\alpha_g}, \quad (4.15)$$

onde $\alpha'_g(t)$ é a flutuação local temporal; $\alpha'_g(x, y, z)$ a flutuação espacial entre células.

Considerado a descrição conceitual da distribuição de vazios apresentada por Jones e Zuber (1975), e assumindo que os valores de α_g em diferentes células do escoamento seguem aproximadamente uma distribuição normal $N(\mu, \sigma^2)$ obtém-se:

$$\overline{\alpha_g} = \mu, \quad (4.16)$$

$$\sigma_{\alpha_g} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (\alpha_{g,i} - \mu)^2}, \quad (4.17)$$

sendo $\overline{\alpha_g}$ a fração média de ar no volume considerado; μ a média global da fração de vazios; σ_{α_g} o desvio padrão (variância) e N o número de células.

O coeficiente de clusterização pode ser dado por:

$$C_v = \frac{\sigma_{\alpha_g}}{\overline{\alpha_g}}. \quad (4.18)$$

Para estimar como a variação de α_g afeta a relação entre ar e água $R = \alpha_g/(1-\alpha_g)$ pode-se utilizar a expansão de Taylor:

$$\bar{R} = \frac{\bar{\alpha}_g}{1-\bar{\alpha}_g} + \frac{\sigma_{\alpha_g}^2}{(1-\bar{\alpha}_g)^3}. \quad (4.19)$$

4.3 HIDRODINÂMICA

As simulações computacionais de fluidos em programas CFD baseiam-se em equações governantes utilizando métodos numéricos para o cálculo de cada grandeza do domínio definido ao longo do tempo.

O conjunto das equações de Navier-Stokes, baseadas em leis de conservação, descreve o comportamento de fluídos em um domínio definido considerando efeitos de campos vetoriais (como velocidade), grandezas escalares (pressão, temperatura) e propriedades da matéria (viscosidade e densidade).

A equação de conservação da massa e da quantidade de movimento para fluídos incompressíveis é dada pelas equações (4.20) e (4.21), respectivamente:

$$\nabla \cdot \mathbf{U} = 0, \quad (4.20)$$

$$\rho \left(\frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U} \cdot \nabla) \mathbf{U} \right) = -\nabla p + \mu \nabla^2 \mathbf{U} + \rho \mathbf{g}, \quad (4.21)$$

sendo $\mathbf{U}$ o vetor de velocidade; ρ e μ a densidade e viscosidade; p a pressão; $\mathbf{g}$ a aceleração da gravidade.

No modelo VOF a equação de transporte utiliza uma função indicativa que representa a superfície livre dada pela Equação (4.22):

$$\frac{\partial \gamma}{\partial t} + \nabla \cdot (\mathbf{U} \gamma) = 0, \quad (4.22)$$

onde γ é a fração do volume de gás.

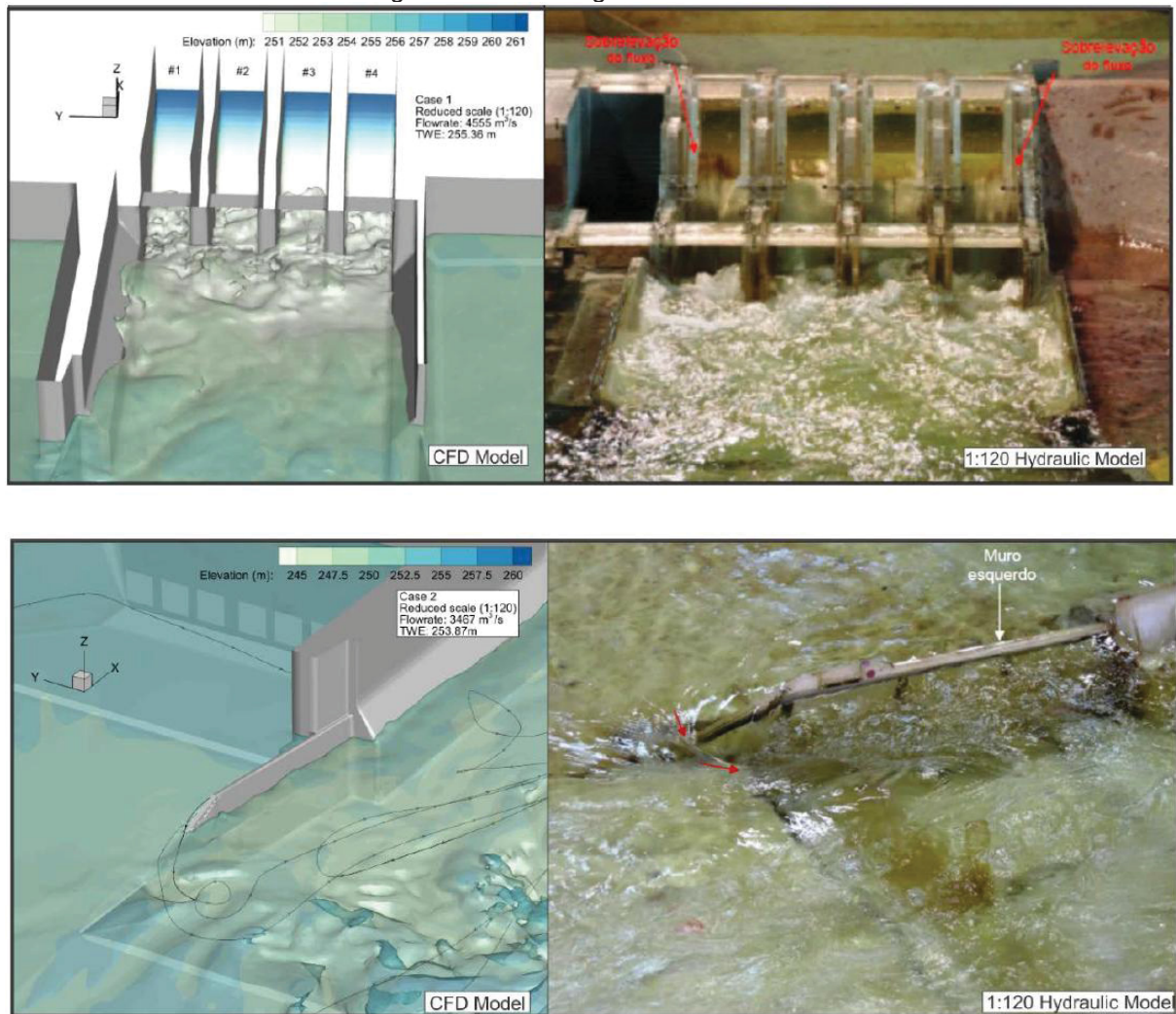
As propriedades físicas do fluido em qualquer ponto no domínio computacional são calculadas como médias ponderadas com base na distribuição da fração de volume pela Equação (4.23) e Equação (4.24):

$$\rho = \rho_l \gamma + \rho_g(1 - \gamma), \quad (4.23)$$

$$\mu = \mu_l \gamma + \mu_g(1 - \gamma). \quad (4.24)$$

A Figura 22 ilustra um comparativo da superfície livre prevista no modelo computacional e verificada no MHR, apresentado no estudo desenvolvido por Politano et al. (2022) para a UHE Colíder.

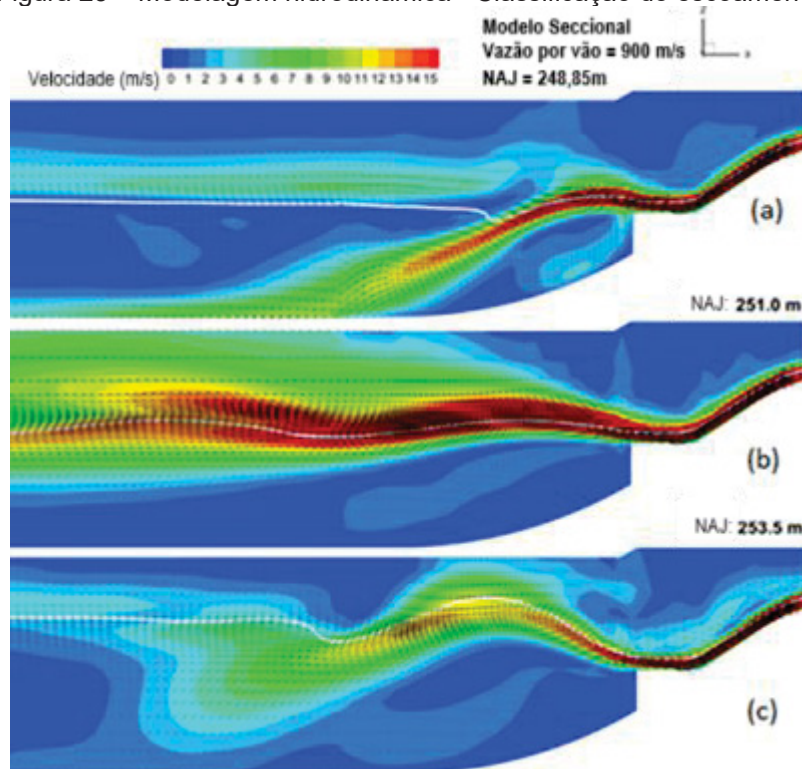
Figura 22 – Modelagem hidrodinâmica.



Fonte: Politano et al. (2022).

A Figura 23 ilustra a simulação hidrodinâmica do escoamento classificando o tipo do escoamento em: a) *plunging flow*; b) *skimming flow* e c) *surface jump*, respectivamente, sendo estas simulações compatíveis com os tipos de escoamentos verificados em modelo hidráulico reduzido e apresentado na Figura 6.

Figura 23 – Modelagem hidrodinâmica - Classificação do escoamento.



Fonte: Politano et al. (2022).

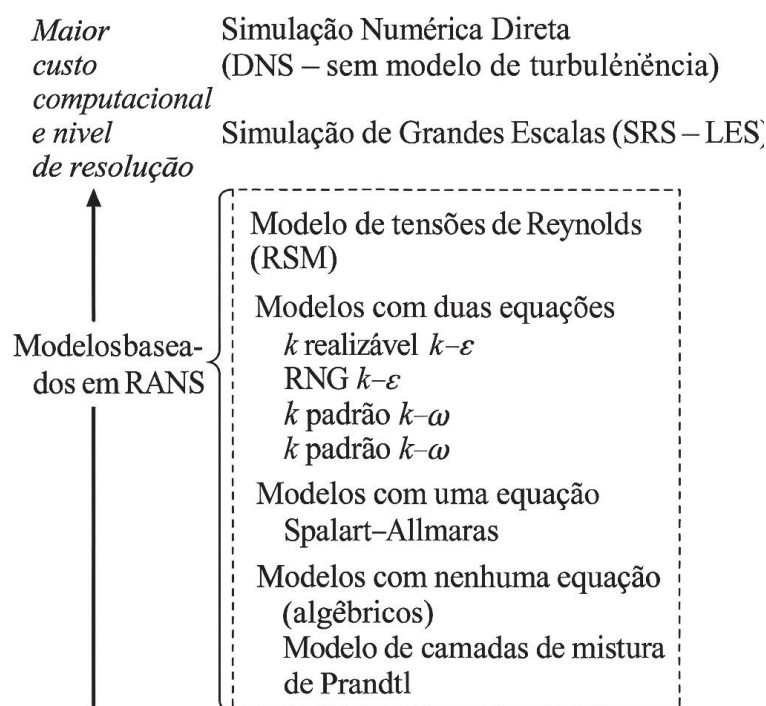
De acordo com Hinze (1975), conforme citado por Bengt Andersson et al. (2011) o movimento turbulento é definido como uma condição irregular do escoamento na qual diversas quantidades apresentam variações aleatórias em função das coordenadas de espaço e tempo, de modo que seus valores médios podem ser estatisticamente discernidos. Essas variações instantâneas, também chamadas de flutuações, ocorrem no campo de velocidades, temperatura, pressão e outros escalares, geradas por diferentes dimensões de vórtices no espaço e tempo que são preponderantes no transporte de quantidade de movimento e energia comparadas à difusão a nível molecular (Souza *et al.*, 2011).

O processo de transferência contínua de energia no interior do escoamento turbulento foi descrito por Kolmogorov em 1941. Os maiores vórtices absorvem a energia do escoamento médio, transferindo-a para outros menores sucessivamente até a ocorrência da dissipação de energia pela viscosidade. Esse processo é conhecido por “cascata de energia”.

A resolução direta das equações governantes de escoamento com geometrias complexas é inviável levando à necessidade do uso de modelos que diminuam o esforço computacional requerido e sejam capazes de representar a turbulência. Os principais modelos de turbulência utilizados em modelos CFDs são o DNS (*Direct*

Numerical Simulation), LES (*Large-Eddy Simulation*), RSM (*Reynolds Stress Model*), RANS (*Reynolds-Averaged Navier Stokes*) e DES (*Detached Eddy Simulation*). A Figura 24 relaciona o esforço computacional com os principais modelos de turbulência utilizados em modelos CFDs.

Figura 24 – Classificação dos tipos de simulação da turbulência.



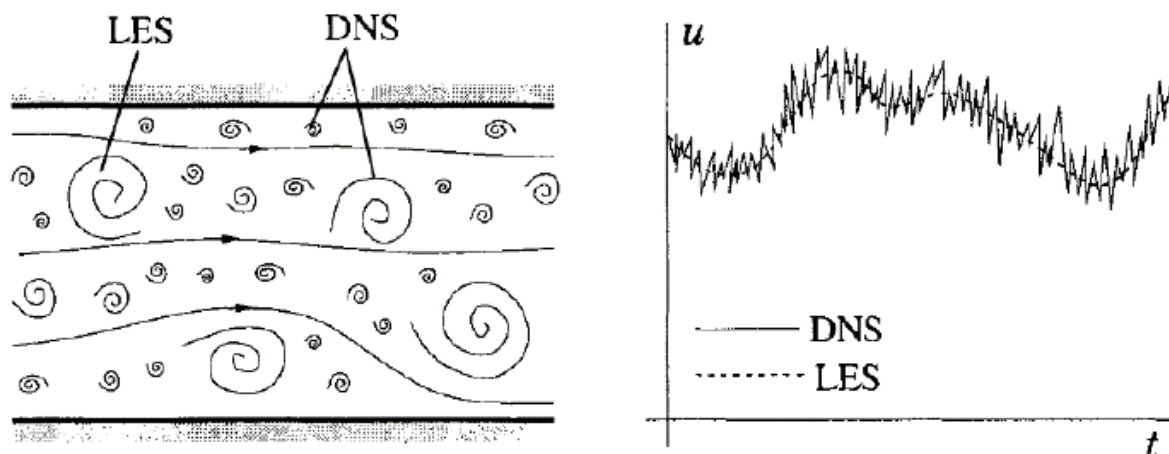
Fonte: Ferziger e Peric (2002).

A simulação DNS resolve adequadamente as equações de Navier-Stokes por completo, não sendo necessário o fechamento por modelos de turbulência. Segundo Ferziger e Peric (2002), os resultados obtidos em simulações DNS são equivalentes a dados experimentais. Contudo, seu uso em projetos de engenharia é impraticável, devido à grande infraestrutura computacional demandada.

O método de simulação de turbulência em grande escala, LES, também conhecido como SRS (*Scale Resolving Simulation*), é uma simulação numérica intermediária aos modelos RANS e DNS, e calcula numericamente o movimento das maiores escalas de comprimento e tempo do escoamento, semelhante ao DNS (Jasak, 2009). As menores escalas, de comportamento isotrópico, são representadas de maneira simplificada por modelos submalha, aumentando sua aplicação em escoamentos com valores de Reynolds mais elevados. A Figura 25 apresenta um

esquema comparativo dos vórtices calculados por LES e DNS (à esquerda) e o tratamento da flutuação de velocidade ao longo do tempo em cada modelo (à direita).

Figura 25 – Comparação entre simulações DNS e LES.



Fonte: Ferziger e Peric (2002).

O RANS também é conhecido como modelos de fechamento em um ponto ou fechamento local, e representa a decomposição das variáveis das equações de Navier-Stokes em uma parcela média e outra flutuante (Souza *et al.*, 2011). As demais nomenclaturas se devem ao fato da variável turbulenta ser parametrizada com base no gradiente do valor médio da variável no mesmo ponto ou local analisado.

As equações RANS são obtidas pelo processo de decomposição de Reynolds. Isto é, as variáveis do escoamento (i.e., componentes do vetor velocidade, pressão e campos escalares) são decompostas em um valor médio somado ao valor flutuante em torno da média. As variáveis definidas dessa forma são substituídas nas equações originais de Navier-Stokes e uma média temporal é extraída em seguida. Esse processo resulta nas equações médias de conservação de massa e quantidade de movimento.

O DES é um modelo híbrido, que une características das simulações LES e RANS, com uma discretização espacial e temporal variável de acordo com a região do domínio. A união dos dois modelos tem como objetivo reduzir o esforço computacional nas regiões próximas às paredes geradas pelo detalhamento necessário na malha do LES, mantendo a representação física de escoamentos com valores elevados de Reynolds das aplicações industriais (Wilcox, 2006).

Para resolver a turbulência a abordagem DES foi utilizada por Politano *et al.* (2012). O modelo DES combinado com o RANS perto de paredes e com o LES no

fluxo principal, captura o comportamento anisotrópico da turbulência perto da superfície livre, o que é importante para prever com precisão os padrões de fluxo de fuga e os regimes de fluxo de jato do vertedouro com defletores instalados.

Os modelos RSM (*Reynolds Stress Models*) não são baseados no conceito de viscosidade turbulenta. Esses modelos são fundamentados em equações deduzidas para o transporte das componentes do tensor de Reynolds, os quais tendem a produzir resultados melhores para escoamentos turbulentos heterogêneos com forte anisotropia. Isto acontece porque a difusão turbulenta não é mais baseada em uma única propriedade escalar, como acontece nos modelos de viscosidade turbulenta, mas tratada separadamente nas equações de transporte das componentes do tensor de Reynolds. A desvantagem dessa abordagem é a complexidade das equações resultantes, as quais acabam necessitando de submodelos para possibilitar o fechamento do sistema (LACTEC, 2020).

O Quadro 1 apresenta um resumo dos principais modelos de turbulência das simulações RANS apresentado em LACTEC (2020).

Quadro 1 – Resumo dos modelos de turbulência.

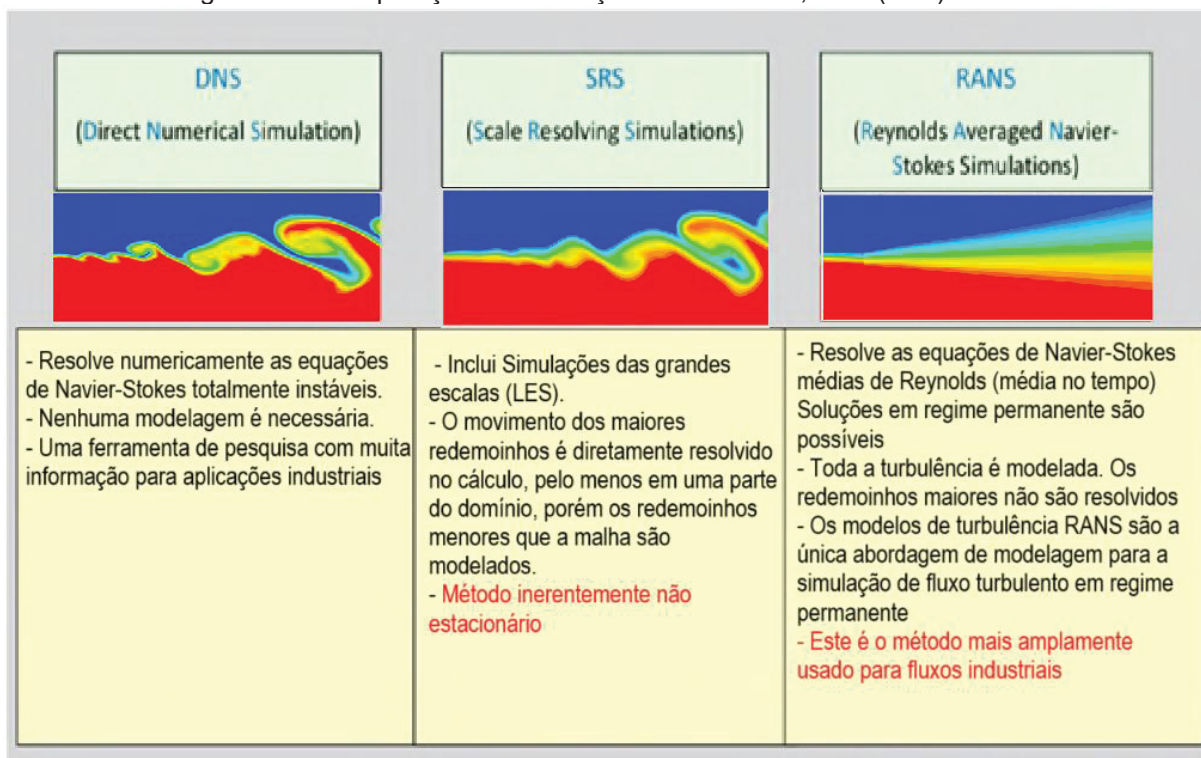
Modelo de Turbulência	Vantagens	Desvantagens / Limitações
Modelos de zero-equação, ex.: modelo de comprimento de mistura.	Custo baixo, aplicável para um número limitado de fluxos.	Quando $\partial U / \partial y = 0 \rightarrow \nu_t = 0$. Falta transporte de escalas turbulentas. A estimativa do comprimento de mistura é difícil. Não pode ser usado como modelo geral de turbulência.
Modelos de uma equação, ex.: modelo algébrico k.	Custo baixo, aplicável para um número limitado de fluxos.	O uso de uma equação algébrica para a escala de comprimento é muito restritivo. O transporte da escala de comprimento não é considerado.
Modelos de duas equações, k- ϵ , k- ω , RNG, Realizable e SST.	Modelos completos nos quais a velocidade e escalas de comprimento da turbulência são previstas. Bons resultados para muitas aplicações de engenharia. Robusto, econômico e fácil de aplicar.	Limitado pela suposição de viscosidade turbulenta. A viscosidade turbulenta é assumida como isotrópica. Convecção e difusão das tensões de cisalhamento são negligenciadas.
k- ϵ padrão.	O modelo mais utilizado e validado.	Não é bom para jatos redondos e fluxos com forte curvatura, rotação, aceleração súbita, separação e baixos números de Reynolds.
RNG k- ϵ .	Modificação do k- ϵ padrão com melhor desempenho para fluxos rotacionais e com separação.	Menos estável que o k- ϵ padrão. Não adequado para jatos redondos.

k- ω	Funciona bem para baixos números de Reynolds. Não precisa de funções de parede. Bom com gradientes de pressão adversos e fluxos separados.	Necessita malha refinada próxima à parede, primeiro ponto na parede em $y^+ < 5$.
SST	Usa k- ϵ no escoamento livre e k- ω na região junto à parede. Funciona bem com gradientes de pressão adversos e fluxos separados. Muitos autores recomendam substituir o k- ϵ por SST.	Necessita malha refinada junto à parede. Superestima a turbulência em regiões de alta deformação normal, como regiões de estagnação e aceleração forte. Melhor que k- ϵ .
Modelos de tensão de Reynolds (RSM)	Aplicável para fluxos complexos onde modelos de viscosidade turbulenta falham. Considera anisotropia. Bom desempenho em fluxos complexos como rotação, separação e jatos planos.	Computacionalmente caro com 11 equações de transporte. Muitos termos precisam ser fechados. Desempenho ruim para alguns fluxos devido às aproximações dos fechamentos.
Simulação de Grandes Escalas (LES)	Aplicável para fluxos complexos. Dá informações sobre estruturas em fluxos turbulentos. Fornece muitos dados que não podem ser obtidos de outra forma.	Alto custo computacional. Grande volume de dados para armazenamento e pós-processamento. Difícil encontrar condições de contorno dependentes do tempo.
Simulação Numérica Direta (DNS)	Nenhum modelo de turbulência é usado. Útil para desenvolver e validar modelos de turbulência. Alta precisão.	Custo computacional extremo para simulações de engenharia. Grande quantidade de dados.

Fonte: LACTEC (2020).

A Figura 26 ilustra uma comparação entre os três modelos de comparação (DNS, SRS e RANS).

Figura 26 – Comparação de simulações entre RANS, SRS (LES) e DNS.

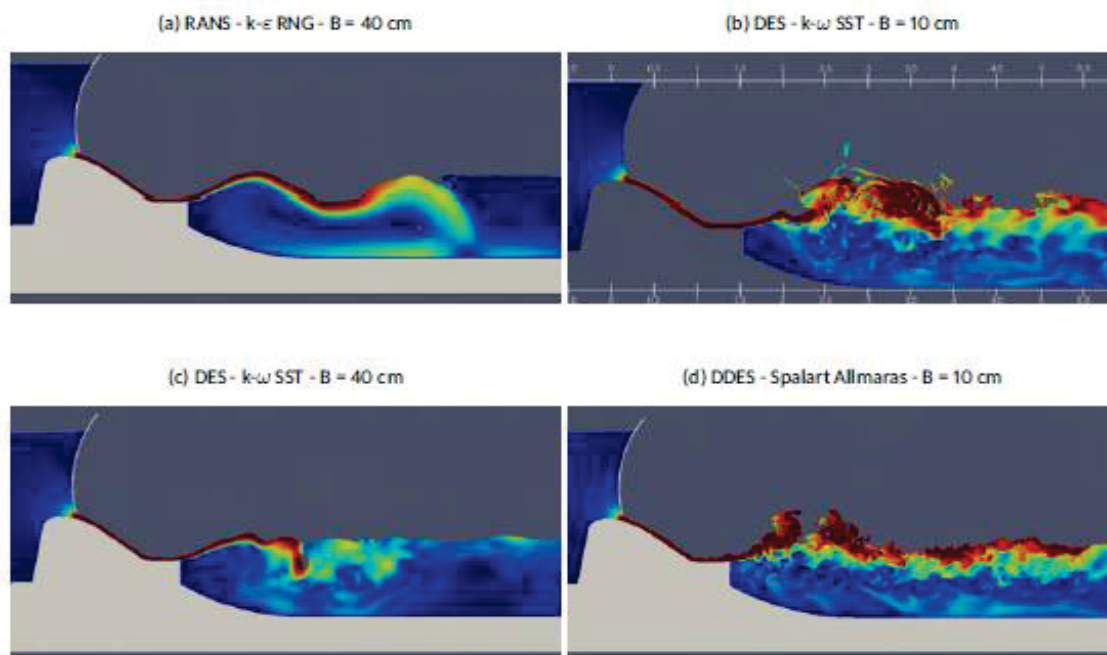


Fonte: Ovelar e Takenobu (2023).

No trabalho de Pesquisa desenvolvido pelo Lactec sobre a temática do TDG (LACTEC, 2020), o estudo do comportamento hidrodinâmico do escoamento e das concentrações de ar no ressalto hidráulico, foi avaliado utilizando o software OpenFOAM, com o uso do solver InterFoam, sendo avaliadas as simulações RANS (modelo de turbulência: $k-\epsilon$ padrão, realizable $k-\epsilon$, RNG $k-\epsilon$ e $k-\omega$ SST), DES (modelo de turbulência: $k-\omega$ SST) e LES (modelo de turbulência: Smagorinsky-Lily).

A Figura 27 apresenta a diferença na aparência do ressalto hidráulico formado para cada modelo de simulação. A simulação DES mistura a simulação RANS (em regiões de pouco refinamento) e a simulação LES (em regiões de alto refinamento). A natureza caótica da superfície livre do ressalto hidráulico é melhor representada nos modelos DES com maior grau de refinamento da malha.

Figura 27 – Comparação da superfície livre média.

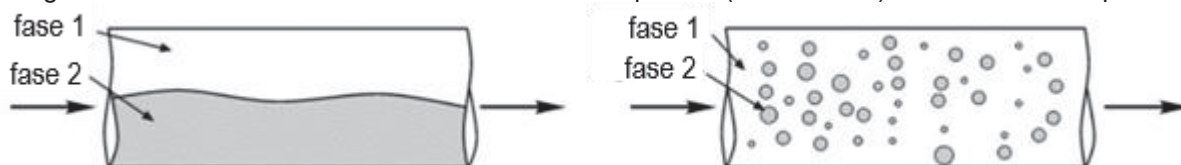


Fonte: Ovelar e Takenobu (2023).

4.4 SIMULAÇÃO COMPUTACIONAL DE ESCOAMENTO MULTIFÁSICOS

Escoamentos multifásicos existem quando mais de uma fase escoar simultaneamente. Por fase, entende-se uma região do espaço delimitada por uma interface de espessura infinitesimal que encerra em seu interior um material com composição química homogênea, propriedades de transporte e de estado definíveis e que seja separável por processos mecânicos. Os escoamentos multifásicos são classificados principalmente em dois grupos: escoamentos separados (estratificados) e escoamentos dispersos (CEHPAR - LACTEC, 2020). A Figura 28 ilustra o escoamento multifásico estratificado (à esquerda) e o escoamento disperso (à direita).

Figura 28 - Escoamento multifásico: escoamento separado (estratificado) e escoamento disperso.



Fonte: Andersson et al. (2012).

Andersson et al. (2012) classificam o escoamento disperso em dois tipos: densos ou diluídos, conforme o grau de separação das partículas da fase dispersa. Em escoamentos diluídos, o comportamento dominante é da fase contínua. Já em escoamentos densos, as interações entre partículas são dominantes.

4.4.1 Caracterização do escoamento multifásico

Em um escoamento multifásico, a quantidade de volume ocupado pelas fases é descrita pela variável fração de volume (*volume fraction*). A fração de volume da fase dispersa se caracteriza pela soma dos volumes das partículas dispersas em um volume unitário, conforme Equação (4.25):

$$\alpha_d = \frac{\sum_{i=1}^{N_d} V^i}{V}, \quad (4.25)$$

sendo α_d a fração de volume da fase dispersa; V^i a volume ocupado por uma partícula ou gota; N_d a quantidade de partículas ou gotas no volume V .

Tendo em vista que a soma das frações de volume das fases dispersa e contínua deve ser igual a 1, a fração de volume da fase contínua α_f resulta na Equação (4.26):

$$\alpha_f = 1 - \alpha_d. \quad (4.26)$$

A fração de volume é calculada ao longo do tempo com uma equação de transporte adicional. Como a soma de todas as frações de volume devem ser iguais a 1, em um sistema multifásico com dois fluidos, é necessária somente uma equação de transporte para o cálculo da fração de volume de ambas as fases. Se o sistema multifásico possuir três fluidos (ex. água, ar e óleo), são necessárias duas equações de transporte para o cálculo da fração de volume. A Equação (4.27) expressão o transporte da fração de volume da fase dispersa:

$$\frac{\partial \alpha_d}{\partial t} + \nabla \cdot (\alpha_d \mathbf{u}_d) = 0, \quad (4.27)$$

em que $\mathbf{u}_d$ é a velocidade da fase dispersa.

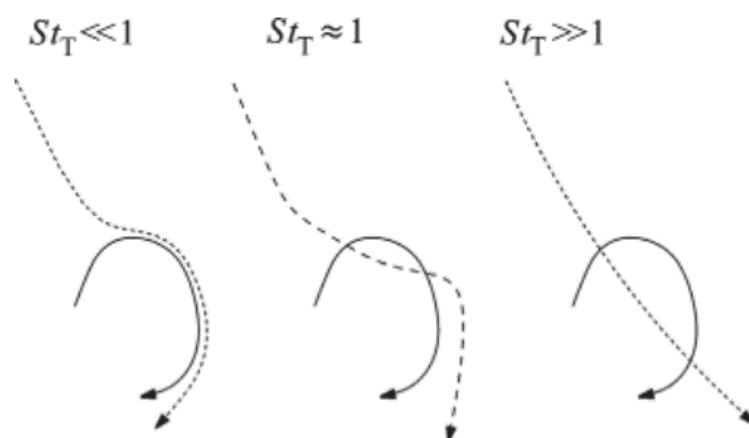
A inércia das fases é expressa em termos de densidade aparente (*bulk density*), representado por $\alpha_d \mathbf{u}_d$. A densidade da mistura é representada pelas somas das densidades das fases dispersa e contínua.

O número de Stokes é um número adimensional que caracteriza o comportamento de partículas no escoamento, relacionando a escala de tempo da fase dispersa (partículas) com a da fase contínua (escoamento). Valores de número de

Stokes muito maiores que a unidade indica um escoamento em que as partículas são independentes do comportamento da fase continua.

Pode-se citar como exemplos de número de Stokes os números de Stokes de Colisão e de Turbulência. O número de Stokes de Colisão pode ser utilizado para categorizar o escoamento disperso em diluído (número de Stokes de colisão maior que 1) ou denso. Já o número de Stokes de Turbulência se relaciona ao comportamento de partículas em relação às turbulências do escoamento, conforme apresentado na Figura 29 (CEHPAR - LACTEC, 2020).

Figura 29 - Efeito de vórtice turbulento (linha contínua) em uma partícula (linha tracejada) para diferentes números de Stokes.

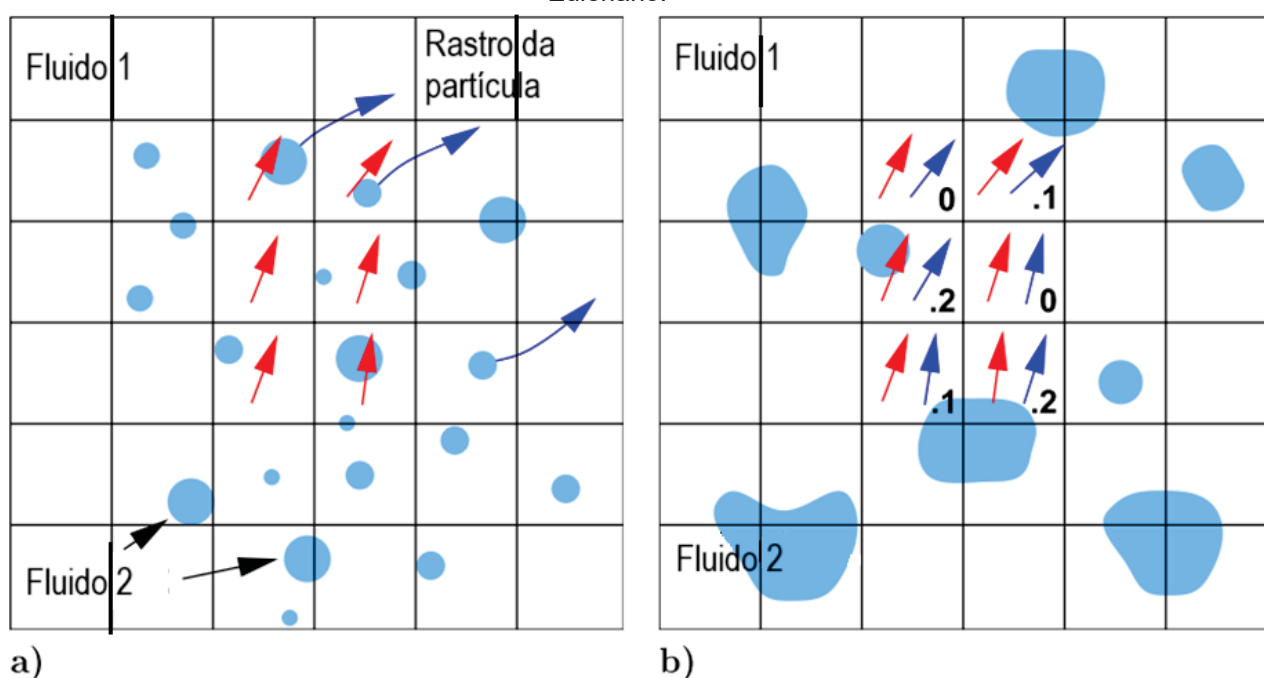


Fonte: Andersson et al. (2012).

4.4.2 Tipos de modelos computacionais multifásicos

As duas principais abordagens para o cálculo de escoamentos multifásicos são os modelos Euleriano-Lagrangiano e Euleriano-Euleriano, ilustrados na Figura 30. O primeiro é utilizado para escoamentos dispersos, onde a fase fluida é modelada como continua e as partículas da fase dispersa têm suas trajetórias modeladas individualmente. O segundo modelo, Euleriano-Euleriano, aborda as fases como fluidos, por isso também é chamado de modelo de dois fluidos. O modelo Euleriano-Euleriano pode ser simplificado para reduzir o esforço computacional dos cálculos dando origem a dois modelos: o modelo de mistura (escoamento algébrico) e o modelo *Volume of Fluid* (VOF).

Figura 30 - Abordagens do escoamento multifásico: a) Euleriano-Lagrangiano e b) Euleriano-Euleriano.



Fonte: Andersson et al. (2012).

O modelo Euleriano-Euleriano também é chamado de modelo de dois fluidos, pois considera as fases como dois (ou mais) quase-fluidos totalmente Interpenetrantes. Os modelos de dois fluidos baseiam-se na média de conjuntos ou na média de volumes, assim, sendo as propriedades tomadas por valores médios, uma importante quantidade considerada no equacionamento é a fração de volume α_k — fração de volume de uma fase k qualquer (Andersson *et al.*, 2012). A fração de volume é a base da distribuição das fases e do tamanho do volume computacional. As equações locais são tomadas com médias dos volumes considerando a densidade aparente de cada fase.

4.4.3 Modelo *Volume of Fluid* (VOF)

A modelagem *Volume of Fluid* (VOF) é utilizada em escoamentos multifásicos estratificados. Segundo Andersson et al. (2012), um dos principais objetivos da simulação de escoamentos estratificados é a correta caracterização da interface entre fluidos.

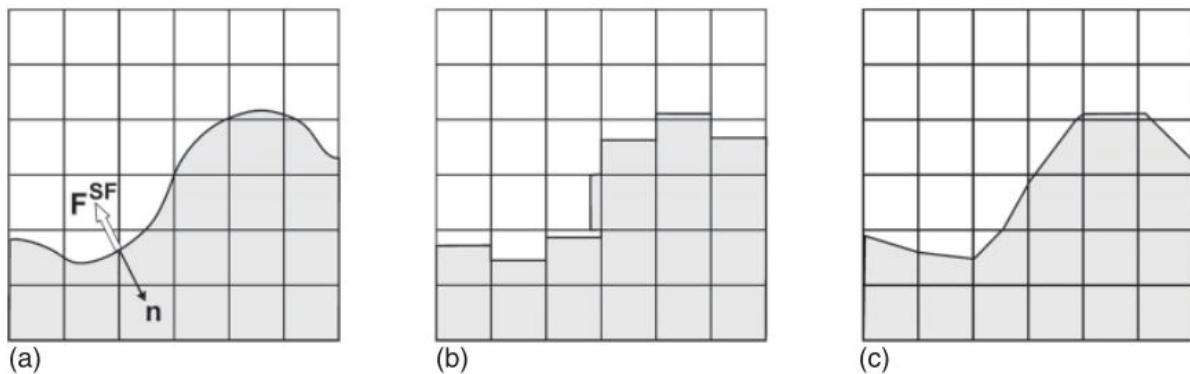
Este método assume a condição de não deslizamento (*no slip*) na interface entre fluidos, e com isso devem ser resolvidos os vórtices na escala de comprimento de Kolmogorov. O método VOF utiliza uma equação de transporte para representar a

fração de volume ocupada por um fluido ao longo do domínio computacional, dada pela Equação (4.28):

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\alpha) = 0. \quad (4.28)$$

A fração de volume também é utilizada para descrever a posição da interface entre fluidos, conforme ilustrada na Figura 31. Por esse método ser considerado de precisão de primeira ordem, uma malha refinada é necessária para garantir precisão dos resultados. Andersson et al. (2012) exemplificam que, para se obter boa precisão na caracterização da interface de uma “bolha”, uma malha de cerca de 20 células por diâmetro da “bolha” é necessária.

Figura 31 – Modelo VOF (*Volume of Fluid*): a) interface real entre fluidos, b) fração de volume calculada pelo VOF e c) interpolação linear da superfície.



Fonte: Andersson et al. (2012).

Em modelos de escoamento estratificado, geralmente adiciona-se a força de tensão superficial $\mathbf{S}_{i,s}$ na equação de quantidade de movimento, uma vez que essa força se torna importante quando a interface entre fluidos deixa de ser reta. A Equação (4.29) ilustra a equação de quantidade de movimento adicionando-se a tensão superficial:

$$\rho \left(\frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U} \cdot \nabla) \mathbf{U} \right) = -\nabla p + \mu \nabla^2 \mathbf{U} + \rho \mathbf{g} + \mathbf{S}_{i,s}. \quad (4.29)$$

Andersson et al. (2012) salientam que, em grandes interfaces tais como as encontradas em escoamentos estratificados sem grandes ondulações, ou em “bolhas” ou gotas com diâmetros maiores que alguns centímetros, a força de tensão superficial pode ser negligenciada. Todavia, em escoamentos com “bolhas” de alguns milímetros,

a força de tensão superficial pode ser dominante e ser responsável por manter as “bolhas” com formato esférico.

4.4.4 Algoritmo simplificado do VOF

Simplificadamente um algoritmo VOF segue as seguintes etapas:

1. Inicialização:

- Inicializa-se o campo de fração de fase (α) (por exemplo, ($\alpha = 1$) para ar, ($\alpha = 0$) para água);
- Inicializa-se o campo de velocidade ($\mathbf{U}$);
- Inicializa-se o campo de pressão p .

2. Loop de tempo: para cada passo de tempo (t) até o final da simulação:

a) Resolve-se a equação de transporte de fração de fase:

- Calcula-se a velocidade relativa ($\mathbf{U}_r$) para compressão de interface;
- Resolve-se a Equação (4.30) de transporte para α :

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{U}) + \nabla \cdot (\alpha(1 - \alpha) \mathbf{U}_r) = 0. \quad (4.30)$$

- Aplique-se condições de contorno apropriadas para α .

3. Atualiza-se as Propriedades do Fluido:

Calcula-se a densidade da mistura ρ e a viscosidade μ com base em α , conforme apresentado na Equação (4.30) e Equação (4.31), respectivamente:

$$\rho = \alpha \rho_{ar} + (1 - \alpha) \rho_{\acute{a}gua}, \quad (4.31)$$

$$\mu = \alpha \mu_{ar} + (1 - \alpha) \mu_{\acute{a}gua}. \quad (4.32)$$

4. Resolve-se a equação do momento:

- Resolve-se as equações de Navier-Stokes para o campo de velocidade $\mathbf{U}$ e campo de pressão p , conforme apresentado pela Equação (4.33):

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U} \mathbf{U}) = -\nabla p + \nabla \cdot (\mu \nabla \mathbf{U}) + \mathbf{F}_c = 0. \quad (4.33)$$

- Aplique-se condições de contorno apropriadas para $\mathbf{U}$ e p .

5. Acoplamento Pressão-Velocidade:

- Utiliza-se um algoritmo de acoplamento pressão-velocidade, como o PISO (*Pressure-Implicit with Splitting of Operators*) ou SIMPLE (*Semi-Implicit Method for Pressure-Linked Equations*), conforme Equação (4.34):

$$\nabla \cdot \mathbf{U} = 0. \quad (4.34)$$

6. Atualizar quantidades de turbulência (se estiver usando um modelo de turbulência):

- Calcular a viscosidade turbulenta ν_t usando o modelo $k-\omega$ SST, Spalart-Allmaras ou outro modelo de turbulência.
- Atualizar a viscosidade efetiva $\mu_{ef} = \mu + \nu_t$.

7. Pós Processamento

Armazena-se os campos calculados (α , $\mathbf{U}$, ρ) para pós-processamento e visualização.

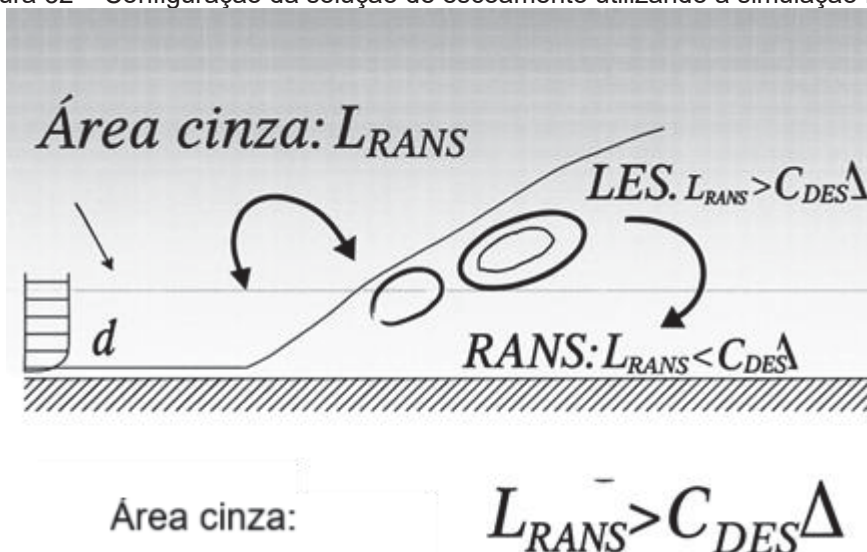
8. Fim do loop.

4.4.5 DES (*Detached-Eddy Simulation*)

A metodologia de simulação DES, inicialmente proposta por Spalart et al. (1997), envolve uma aplicação híbrida das metodologias LES e RANS. Segundo Shur et al. (1999), o DES é uma solução numérica tridimensional transiente utilizando um modelo de turbulência, que funciona como um modelo submalha (SGS) em regiões suficientemente detalhadas da malha para realizar o LES e, nas demais regiões, como um modelo RANS. A união dos dois modelos tem como objetivo reduzir o esforço computacional nas regiões próximas às paredes geradas pelo refinamento necessário na malha do LES, mantendo a representação física de escoamentos com valores elevados de Re das aplicações industriais (Wilcox, 2006).

A separação do domínio para o uso das diferentes metodologias de simulação (LES/RANS) no escoamento está ilustrado na Figura 32, com a espessura da camada limite representada pela letra δ .

Figura 32 – Configuração da solução de escoamento utilizando a simulação DES.



Fonte: CEHPAR - LACTEC (2020).

Nota-se que, na Figura 32, a região inferior da camada limite ($\delta < C_{DES} \Delta$) é modelada utilizando o RANS devido ao elevado grau de anisotropia da malha computacional, comum nesta região, e a flutuabilidade do campo de pressões naquele domínio. Nas regiões mais distantes do fundo do canal ($L_{RANS} > C_{DES} \Delta$), utiliza-se o modelo LES. A região intermediária entre o uso das duas soluções é chamada de zona cinza (*gray area*), onde $L_{RANS} \approx C_{DES} \Delta$. Estudos sobre as zonas cinzas são mais complexos, envolvendo mutuamente os dois modelos de turbulência.

Os valores de L_{RANS} e C_{DES} dependem do modelo de turbulência RANS escolhido. As formulações utilizadas em DES possuem variáveis e termos semelhantes a seus respectivos modelos de turbulência originais, como o de Spalart-Allmaras e $k-\omega$. Na Tabela 4 estão apresentados alguns dos valores e termos mais utilizados na modelagem DES.

Tabela 4 – Constantes e termos de fechamento na simulação DES.

Constante/Tempo	Valor/Equação
C_{DES}^{SA}	0,65
$C_{DES}^{k-\omega}$	0,78
$C_{DES}^{k-\varepsilon}$	0,61
$L_{RANS}^{k-\omega}$	$\frac{k^{1/2}}{C_{\mu\omega}}$

Fonte: CEHPAR - LACTEC (2020).

4.4.6 PISO (*Pressure Implicit Splitting of Operators*)

O algoritmo PISO (*Pressure Implicit Splitting of Operators*) é um método utilizado em CFD para resolver as equações de Navier Stokes, aplicado à fluxos compressíveis e incompressíveis. O método é baseado no uso da pressão e velocidade como variáveis dependentes, sendo concebido como uma técnica não iterativa para solução das equações de transporte. A principal característica da técnica é a divisão do processo de solução em uma série de etapas nas quais as operações de pressão são desacopladas daquelas de velocidade em cada etapa, com os conjuntos de equações divididos sendo passíveis de solução por técnicas simplificadas (Issa, 1986).

O método PISO resolve as equações do momento utilizando a pressão como variável dependente no passo de tempo precedente. Embora o momento seja conservado nesta etapa de previsão da velocidade, a conservação de massa tem que ser satisfeita. Isso é alcançado substituindo expressões de fluxo, derivadas do campo de velocidade previsto, na equação de continuidade de massa. A equação resultante é uma equação de correção de pressão, e a solução para ela produz o campo de pressão previsto e correções para as velocidades e fluxos (Bressloff, 2001).

4.4.7 SIMPLE (*Semi-Implicit Method for Pressure-Linked Equations*)

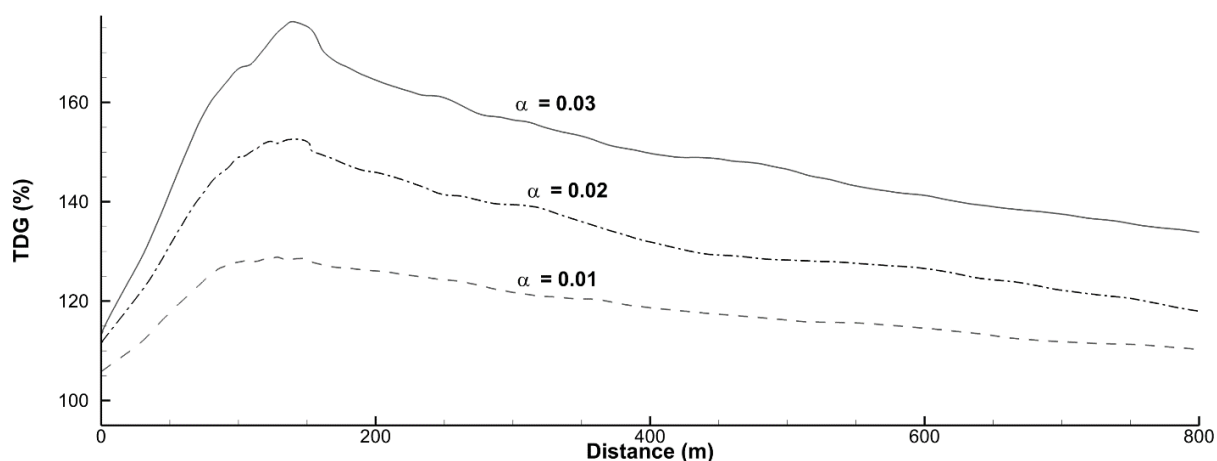
SIMPLE é um procedimento numérico amplamente usado em dinâmica dos fluídos computacional (CFD) para resolver as equações fundamentais que governam a mecânica dos fluídos. As equações da continuidade e de Navier-Stokes são discretizadas e resolvidas de forma semi-implícita.

Os problemas numéricos são discretizados em nós e elementos, sendo que os nós representam a posição no espaço onde os espaçamentos serão calculados (por exemplo: pressão, velocidade), e os elementos definem as equações relacionadas aos parâmetros (por exemplo: continuidade, Navier-Stokes). A geometria do elemento define a extensão dos cálculos relativos às dimensões espaciais, sendo unidimensionais, bidimensionais e tridimensionais (Khawaja e Moatamedi, 2018).

4.5 MODELAGEM DA CONCENTRAÇÃO DE AR

O volume de concentração de ar é um dos parâmetros utilizados na modelagem do TDG. No estudo desenvolvido para a UHE Colíder (Politano *et al.*, 2022) foi avaliada a sensibilidade da relação entre gás e água, entre 1% e 3%, sendo a variação no percentual de TDG ilustrada na Figura 33.

Figura 33 – Análise de sensibilidade do TDG.



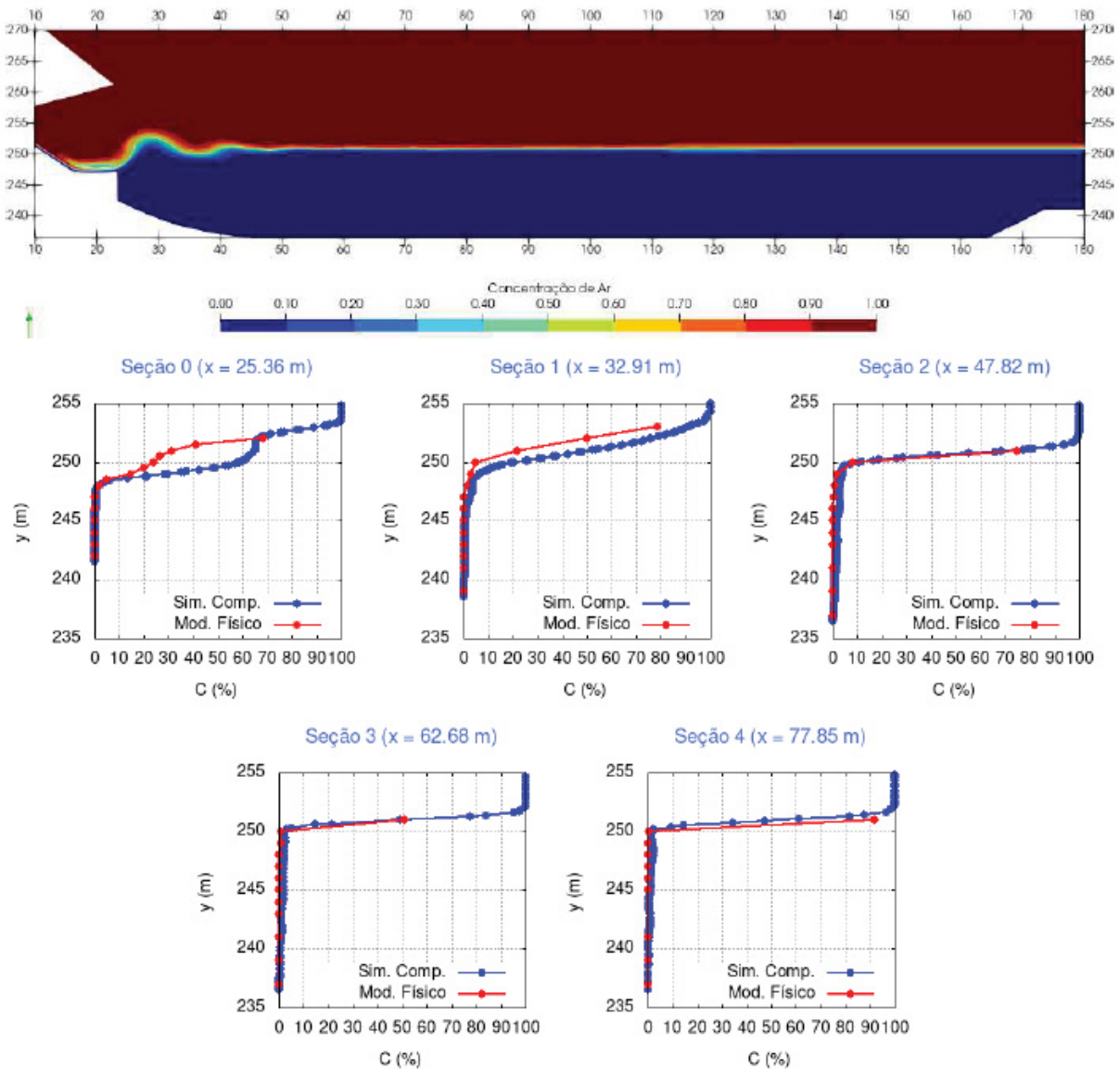
Fonte: Politano et al. (2022).

Em outros estudos desenvolvidos por Politano et al. (2022), a relação entre gás e água foi de 4% para a Wanapum Dam e McNary Dam e 3% para Wells Dam, Hells Canyon Dam e Brownlee Dam.

No estudo das concentrações de ar no ressalto hidráulico, Ovelar e Takenobu (2023) utilizou o software OpenFOAM e o solver interFoam, com as simulações RANS, DES e LES. Os modelos de turbulência (k - ϵ padrão, k - ϵ RNG, k - ϵ realizable e k - ω SST), e malhas de 2,5 mm a 10 cm.

A Figura 34 apresenta a simulação da concentração de ar para $Q = 300 \text{ m}^3/\text{s}$, NAM (nível de água de montante) = 272 m e NAJ (nível de água de jusante) = 250,89 m, e compara as concentrações simuladas com os valores mensurados em modelo físico.

Figura 34 – Perfil médio de concentração de ar.



Fonte: Takenobu et al. (2022).

4.6 MODELAGEM DA “BOLHA”

A equação de transporte de TDG é obtida ao se realizar o balanço de massa do gás dissolvido na fase líquida dentro de um volume de controle. Politano, Carrica e Weber (2009) apresentam a equação de continuidade e momento da fase “bolha” dada pelas Equações (4.35) e (4.36), respectivamente:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{U}_b \alpha) = -S, \quad (4.35)$$

$$0 = -\alpha \nabla p + \alpha \rho_b \mathbf{g} + M_b, \quad (4.36)$$

sendo α (adimensional) a fração de volume de gás na fase “bolha”; S (1/s) a termo fonte de TDG, representa a transferência de massa entre a fase “bolha” e a mistura ar-água circundante, devido à turbulência e pressão; ρ_b (kg/m³) a massa específica do gás da fase “bolha”; M_b (kg/m²s²) o termo fonte de quantidade de movimento, representa a transferência de quantidade de movimento entre as “bolhas” e o líquido, devido à turbulência e pressão.

A massa específica do gás da fase “bolha” é dada pela Equação (4.37):

$$\rho_b = \frac{Mp}{RT}, \quad (4.37)$$

onde M é a massa molar do ar (kg/mol); R a constante universal do gás (kg·m²/(s²·mol·K)); T a temperatura (K).

O termo fonte da quantidade de movimento é dado pela Equação (4.38):

$$M_b = -\frac{3}{8}\rho\alpha_b \frac{C^D}{Re_b} \mathbf{U}_{br}[\mathbf{U}_{br}], \quad (4.38)$$

em que $\mathbf{U}_{br}$ é a velocidade relativa da “bolha” em relação a fase líquida; C^D o coeficiente de arrasto dado pela Equação (4.39):

$$C^D = \begin{cases} \frac{24(1+0,15Re_b^{0,687})}{Re_b} & \text{para } D_b \geq 2e^{-4} \\ \frac{24}{Re_b} & \text{para } D_b < 2e^{-4} \end{cases}, \quad (4.39)$$

sendo D_b o diâmetro da “bolha”; Re_b o número de Reynolds da “bolha” dado pela Equação (4.40):

$$Re_b = \frac{\rho_l D_b |\mathbf{U}_{br}|}{\mu_l}. \quad (4.40)$$

Nos estudos desenvolvidos por Politano et al. (2007) (Politano, Carrica e Weber, 2009), foi considerado que as “bolhas” eram adicionadas ao sistema pelo arrastamento de ar no vertedouro e que não havia nenhuma outra fonte volumétrica de “bolhas” ao longo do domínio, sendo desconsiderada a geração de “bolhas” a partir

de regiões supersaturadas de TDG, pois, segundo os autores, na região a jusante de usinas não há locais que propiciem a nucleação de “bolhas”. Assim, a equação de transporte de densidade de número de bolhas pode ser expressa pela Equação (4.41):

$$\frac{\partial N}{\partial t} + \nabla \cdot (\mathbf{U}_b N) = 0, \quad (4.41)$$

onde N é o número de bolhas por unidade de volume.

O raio da “bolha” é calculado pela Equação (4.42):

$$R = \left(\frac{3\alpha}{4\pi N} \right)^{\frac{1}{3}}. \quad (4.42)$$

Reagrupando as Equações (4.20) e (4.22) e isolando o valor da velocidade relativa das “bolhas” (α) tem-se a Equação (4.43):

$$|\mathbf{U}_{br}| = \frac{Re_b \mu_l}{2\rho_l} \left(\frac{4\pi N}{3\alpha} \right)^{1/3}. \quad (4.43)$$

4.7 MODELAGEM DO TDG

Na modelagem proposta por Wang, Politano e Weber (2018), para o correto cálculo do TDG é necessário ajustar e calibrar dois parâmetros: o tamanho das “bolhas” (raio R_b ou diâmetro D_b) e a fração volume de gás da fase bolha (α_b). Esses dois parâmetros são ajustados e os resultados de TDG são confrontados com dados medidos em campo.

Os trabalhos de Wang, Politano e Weber (2018) e Huang et al. (2019) obtiveram êxito na modelagem do TDG em usinas hidroelétricas, sendo a produção de TDG dada pela transferência de massa da fase bolha para a fase líquida.

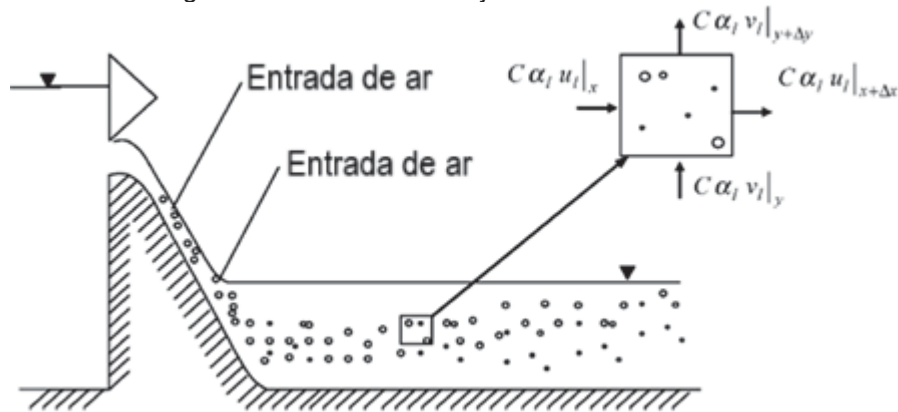
No trabalho de Wang, Politano e Weber (2018), as equações de transporte de “bolhas” e TDG foram implantadas no código fonte do software OpenFOAM, utilizando o solver interFoam, com modelagem da turbulência DES e acoplamento de uma via para a fase bolha. Este modelo necessita de dois parâmetros de entrada: fração volume de gás da fase bolha (α) e o tamanho das “bolhas” (D_b). Neste trabalho foi

apresentada a análise de sensibilidade destes parâmetros (α e D_b) aplicado à UHE McNary, sendo os resultados reproduzidos na Figura 36 e na Figura 37.

Wang, Politano e Weber (2018) concluíram que a modelagem utilizada foi capaz de representar de maneira adequada o transporte e produção de TDG, sendo os resultados da modelagem comparados com os dados medidos em campo, na usina hidrelétrica McNary, no rio Columbia, EUA.

A equação de transporte de TDG é obtida ao se realizar o balanço de massa do gás dissolvido na fase líquida dentro de um volume de controle. A Figura 35 ilustra o ar arrastado na face do vertedouro e o ar aprisionado no início do ressalto hidráulico, além do detalhe do volume de controle.

Figura 35 – Escoamento a jusante de vertedouros.



Fonte: Politano et al. (2007).

Politano, Carrica e Weber (2009) apresenta o cálculo da concentração de TDG utilizando a Equação (4.44):

$$\frac{\partial \alpha_l C}{\partial t} + \nabla \cdot (\mathbf{U} \alpha_l C) + \nabla \cdot (\gamma(1 - \gamma) \mathbf{U}_r) = \nabla \cdot \left(\left(v + \frac{v_t}{Sc} \right) \alpha_l \nabla C \right) + S, \quad (4.44)$$

onde α_l (adimensional) é a fração volume de líquido; C (adimensional) a concentração de TDG em relação ao equilíbrio; $\mathbf{U}_r$ (m/s) a velocidade relativa entre fases; Sc (adimensional) o número de Schmidt; v (m²/s) a viscosidade cinemática do líquido; v_t (m²/s) a viscosidade turbulenta do líquido; S (1/s) o termo fonte de TDG, corresponde à transferência de massa entre a fase bolha para a fase líquida.

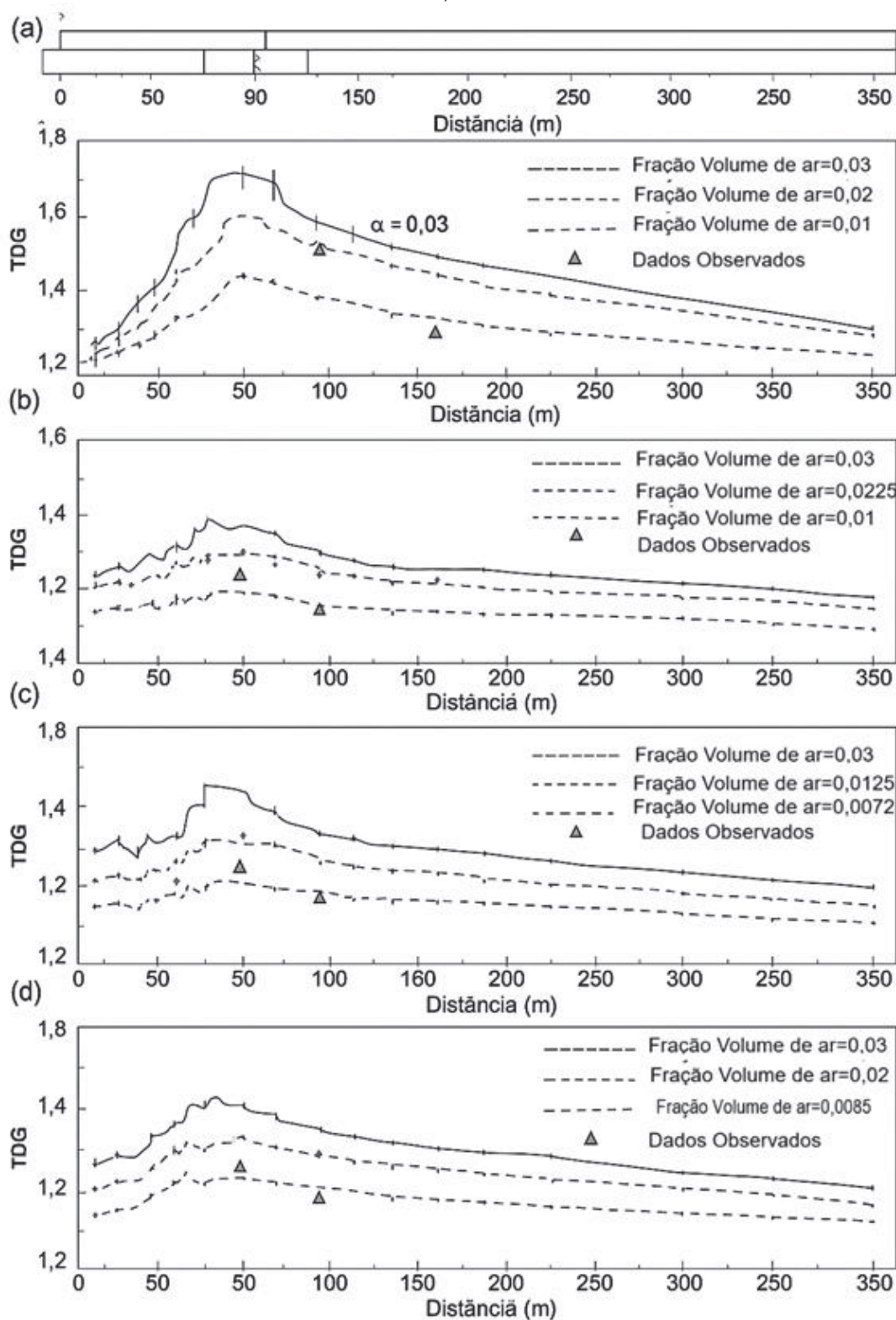
O coeficiente de transferência de massa devido à turbulência externa entre a fase bolha para a fase líquida, dada pela Equação (4.45):

$$S = 4\pi NR^2 K_l \left(\frac{p + \frac{\sigma}{R}}{H} - C \right), \quad (4.45)$$

em que K_l (adimensional) é o coeficiente de transferência de massa devido à turbulência externa; p (N/m²) a pressão; σ (N/m) a tensão interfacial; H (adimensional) a constante de Henry.

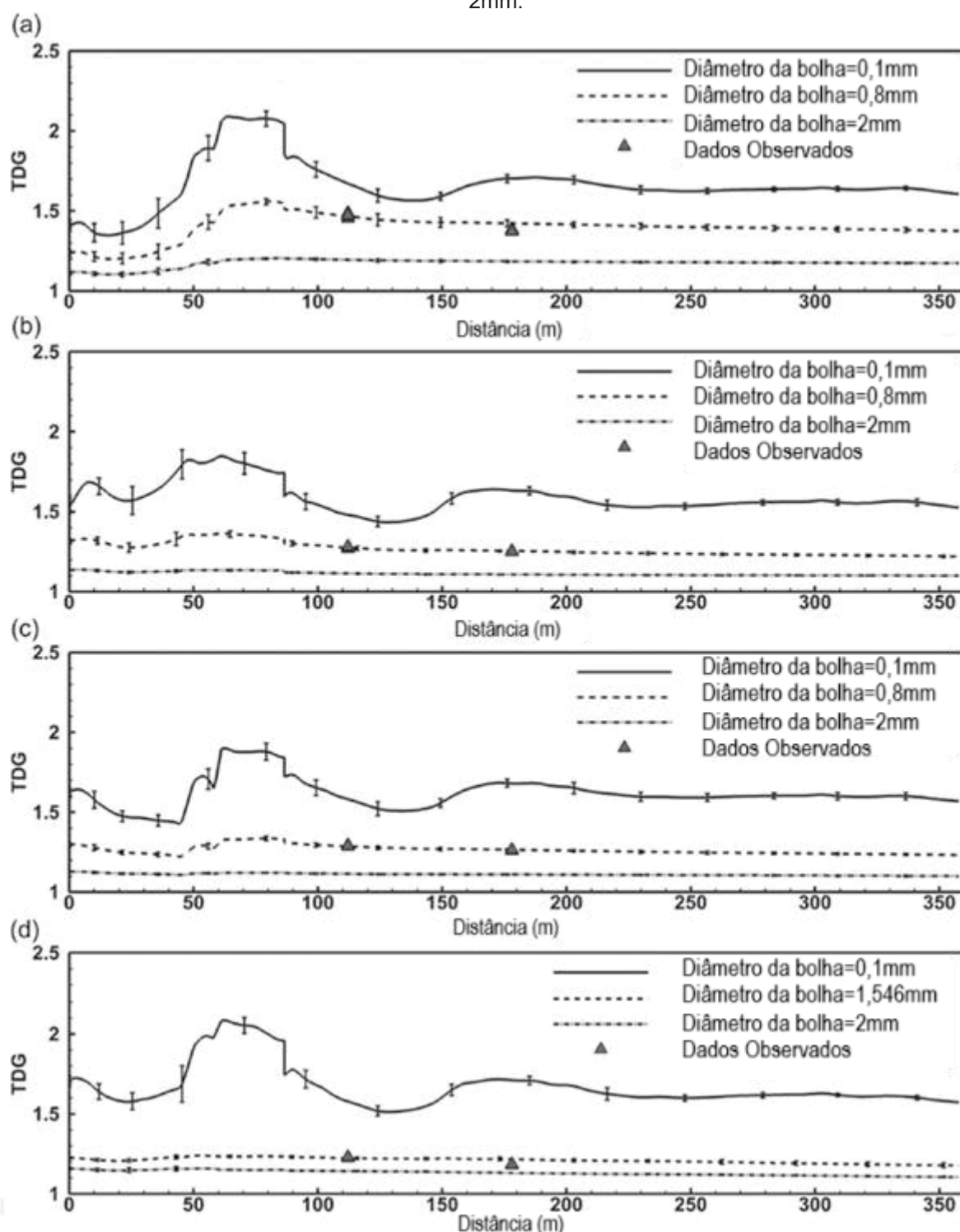
No modelo proposto por Wang, Politano e Weber (2018) a transferência de massa na superfície livre foi negligenciada, uma vez que na maioria das condições a dissolução da “bolha” é o processo dominante na zona aerada perto do vertedouro.

Figura 36 – Análise de sensibilidade do % de TDG adotando $D_b = 0,8\text{mm}$ e α variando entre 0,01 a 0,03.



Fonte: Wang, Politano e Weber (2018).

Figura 37 – Análise de sensibilidade do % de TDG adotando $\alpha = 0,03$ e D_b variando entre 0,1mm a 2mm.



Fonte: Wang, Politano e Weber (2018).

Os resultados apresentados na Figura 36 e na Figura 37 indicam que quanto maior a fração volume de gás da fase bolha e menor o diâmetro da “bolha” mais elevado é o percentual de TDG.

O estudo desenvolvido por Politano et al. (2024), para auxiliar no projeto do defletor da UHE Colíder, utilizou a modelagem CFD, baseada em ferramentas do

OpenFOAM. Foi utilizado o modelo RANS e o modelo de turbulência DES. Foi utilizado o modelo interFoam TDG, modelo trifásico (ar, água e “bolhas” de ar) de superfície livre para prever a distribuição hidrodinâmica do TDG no canal de fuga.

Ao todo foram realizadas 28 simulações utilizando o interFoam TDG para avaliar o efeito do comprimento, raio de curvatura e elevação do defletor no percentual de TDG na bacia de dissipação.

5 AUTÔMATO CELULAR

5.1 INTRODUÇÃO

Modelos computacionais representam todos os meios computacionais que são usados para realizar o estudo de fenômenos, por meio da imitação básica do processo num computador. Autômatos celulares são métodos usados para desenvolver modelos computacionais descritos através de regras simples, geralmente formuladas com a aplicação de física elementar para a realização da transição de estados (Agrawal, 1999).

Exemplos de modelos computacionais: a) uso do modelo de simulação direta de Monte Carlo para analisar sistemas complexos onde os modelos matemáticos que podem ser aplicados são muito aproximados, como no movimento de gases rarefeitos, onde o escoamento é descontínuo (Kaviski E., 2006); b) uso de autômatos celulares através de regras formuladas com a aplicação de física elementar para a realização da transição de estados (Silva, Gramani e Kaviski, 2011); e c) uso do método do reticulado de Boltzmann (Cargnelutti *et al.*, 2017) que é uma categoria de autômato celular com uso específico em dinâmica dos fluidos.

Com modelos matemáticos o fenômeno físico é descrito pelas expressões que representam as leis de conservação de massa, da quantidade de movimento linear, quantidade de movimento angular, conservação de energia, conservação de massa parcial em misturas, conservação da densidade de cargas elétricas e de corrente elétrica, etc. Podem ser aplicados quando o meio é contínuo, regiões de escoamento com número de Knudsen menor do que 0,2 (Kaviski E., 2006).

Modelos computacionais de autômatos celulares e LBM (*Lattice Boltzmann Method*) são inerentemente paralelos, devido a sua extensa rede regular e a forma uniforme da sua dinâmica. O uso de processamento paralelo pode reduzir consideravelmente o tempo total de processamento (Wylie, 1990). Em qualquer sistema de computação paralelo, CPUs (*central processing unit*) que trabalham em partes diferentes do mesmo serviço devem se comunicar umas com as outras para trocar informações. Tanenbaum e Austin (2013), detalharam as principais formas de paralelismo, que são: no circuito integrado (*chip-silicon wafer microprocessor*); com coprocessador (por exemplo, na unidade de processamento gráfico de uso geral); com multiprocessador; com multicomputador; e com computação em grade. A forma do

paralelismo influi diretamente no projeto e no desenvolvimento do método computacional.

A solução numérica de sistemas de equações diferenciais parciais que descrevem fenômenos físicos, tradicionalmente é realizada por meio de métodos baseados em malhas. Na dinâmica dos fluidos a descrição euleriana geralmente é realizada através de uma grade fixa discretizando toda a região onde ocorre a simulação. Na formulação lagrangeana considera-se uma malha móvel ou de um sistema de partículas para discretizar somente o fluido em movimento. Em 1977, foi proposto por Gingold, Monaghan e Lucy (Paiva *et al.*, 2009) o uso de um método sem malha para solucionar numericamente problemas de astrofísica. O método proposto é denominado de SPH (*Smoothed Particle Hydrodynamics*). Com o método SPH a região do escoamento é representada por um conjunto de partículas sem o uso de qualquer malha que forneça a conectividade entre essas partículas. A discretização numérica é obtida usando-se as informações que caracterizam as propriedades das partículas através de médias locais por meio de integrações. A simulação avança no tempo calculando-se as novas posições e respectivas propriedades de cada partícula por meio da integração das equações governantes do escoamento.

Durante o ano de 1986 um grupo de pesquisa do Departamento de Energia americano, produziu um relatório sobre a utilização de autômatos celulares do tipo gás de rede (LGCA - *Lattice Gas Cellular Automata*) em ciências físicas, especialmente em dinâmica dos fluidos (Abarbanel *et al.*, 1990). As análises realizadas tiveram como objetivos: a) comparar as técnicas de autômatos celulares com os métodos convencionais usados para solucionar as equações diferenciais da dinâmica dos fluidos; e b) comparar o uso de programação paralela ou máquinas de programação concorrente usando autômatos celulares e os métodos convencionais.

Foi concluído que: autômatos celulares do tipo gás de rede (LGCA) são apropriados e competitivos com o método dos elementos finitos para solucionar problemas que são modelados pelas equações de Navier-Stokes, principalmente para baixos números de Reynolds e números de Mach; usando-se uma representação simplificada das partículas individuais do movimento com os autômatos celulares do tipo gás de rede (LGCA) consegue-se obter uma boa representação macroscópica das equações de Navier-Stokes.

Wylie (1990) em sua tese de doutorado comprovou a viabilidade de solucionar problemas de hidrodinâmica bidimensional por meio de autômatos celulares do tipo

gás de rede (LGCA). O autor aplicou o método a escoamentos em reservatórios, canais, canais com obstáculos (com barreiras ortogonais, inclinadas, cilíndricas, prismáticas e mistas; com turbulência; em regime não permanente) e jatos (em canais e cruzados). O método computacional foi desenvolvido com processamento paralelo em arquitetura de multicomputadores.

Chen (2004) em sua tese de doutorado comprovou a viabilidade da aplicação de autômatos celulares usando regras definidas com lógica difusa para solucionar problemas de ecologia hidráulica (evolução de diversos tipos de algas).

No livro escrito por Wolf-Gladrow (2000) são apresentadas aplicações de autômatos celulares em várias áreas da física. Com LBM os seguintes assuntos foram tratados: hidrodinâmica em 3 dimensões; hidrodinâmica com conservação de energia; simulação da circulação de águas em oceanos; e difusão e convecção.

Yang e Young (2010) pesquisaram sobre o uso de autômatos celulares probabilísticos na formação de padrões em biologia e engenharia. Foram obtidos resultados em sistemas de escoamentos com reação e difusão (Prud'homme, 2010). Os métodos investigados por Yang e Young (2010) oferecem grande potencial para serem adaptados para investigar a formação de “bolhas” em mecânica dos fluidos.

Chen (2009) usando LBM simulou em 2 dimensões o crescimento de “bolhas” em processos de cavitação natural. O autor concluiu que é viável o uso de LBM para simulação de “bolhas” em escoamentos com cavitação.

Wolf, Santos e Philippi (2006), com LBM e usando um modelo para o potencial de interação entre partículas simularam a interface líquido-vapor. O modelo faz uso de uma equação de estado que possui o comportamento semelhante à equação de Van der Waals. No trabalho são apresentadas soluções para simulações de sistemas simples, mas os autores concluem que o método é capaz de simular uma grande variedade de fenômenos complexos.

Zhang e Yan (2008) (Zhang e Yan, 2010) aplicaram LBM para solucionar a equação de Burgers em uma e duas dimensões. A equação de Burgers representa processos de difusão e convecção em dinâmica dos fluidos. É uma simplificação da equação de Navier-Stokes. Os autores concluíram que LBM pode ser usado para simular processos governados pela equação de Burgers.

Désérable et al. (2011) investigaram o uso de autômatos celulares e LBM para modelar escoamentos com matérias complexas, como em forma granular, fluido-fluido ou sólido-fluido. Os autores propõem o método CACM (*Cellular Automata for Complex*

Matter) que pode ser visto como uma extrema simplificação da dinâmica de partículas, mas que obtém bons resultados para as soluções macroscópicas.

Wu e Shu (2011) demonstraram a viabilidade da aplicação de um esquema de LBM para simular escoamentos em regime permanente e não-permanente sobre um cilindro circular. Os resultados obtidos foram comparados com dados da literatura.

Li et al. (2015) apresentaram uma revisão do LBM para fluidos térmicos com aplicações em estudos energéticos, focalizando em escoamentos multifásicos, escoamentos térmicos e escoamentos térmicos multifásicos, com mudança de fase. A revisão teórica do método permitiu revelar algumas inconsistências, defeitos e aspectos comuns entre modelos matemáticos para escoamentos multifásicos e térmicos. Algumas representações do LBM permitem obter resultados consistentes com a termodinâmica e hidrodinâmica, reduzindo correntes espúrias e garantindo estabilidades numéricas. Com o LBM pode-se propor modelos com forte fundamentação teórica para investigar escoamentos com altos valores para a razão entre a densidade da fase líquida e fase gasosa, altos números de Reynolds e com superfícies das interfaces (Rabóczkay, 2016) com grande variabilidade.

Zhang et al. (2016) examinaram a viabilidade e a eficiência de um esquema de LBM para simular escoamentos em canais em duas dimensões. O método computacional foi desenvolvido numa unidade de processamento gráfico de uso geral (Tanenbaum e Austin, 2013). Foram definidos fatores relacionados com acurácia e a eficiência. Os resultados obtidos foram estáveis se ajustaram bem aos dados observados, podendo ser usados como referências em projetos.

Douillet-Grellier et al. (2019) compararam resultados obtidos com SPH e LBM para simular escoamentos intermitentes multifásicos. O trabalho teve por objetivo avaliar quantitativamente o SPH e o LBM, para decidir critérios para estabelecer em que contextos determinam os melhores resultados. O problema investigado é de interesse em aplicações petrolíferas sobre a formação de regiões com escoamentos com alta densidade e lentos, que podem ocorrer em emissários submarinos causando graves consequências.

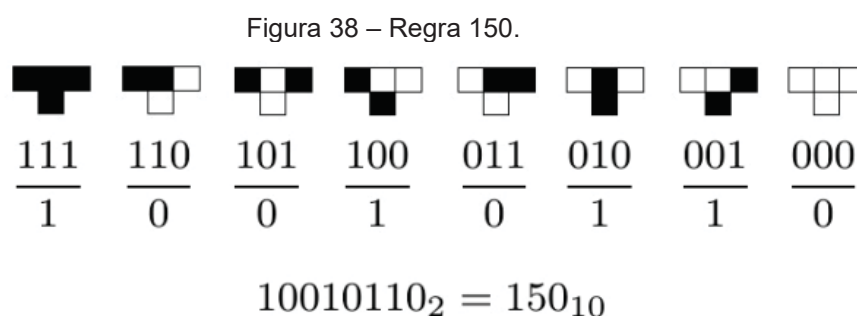
Guzella et al. (2019) usando LBM simularam a ascensão de uma “bolha” e da junção de duas “bolhas”. No problema analisado considerou-se o líquido estacionário e isotérmico com a “bolha” movimentando-se pela ação da força de empuxo. Os resultados obtidos foram coerentes com os previstos por modelos de física teórica.

5.2 O AUTÔMATO CELULAR

Os modelos computacionais representam todos os meios que são usados para realizar o estudo de fenômenos, através da imitação básica do processo num computador. Autômatos celulares (CAs – *Cellular Automata*) são métodos usados para desenvolver modelos computacionais descritos através de regras simples, geralmente formuladas com a aplicação de física elementar para a realização da transição de estados, sendo que os autômatos celulares "elementares" consistem em uma sequência de células com valores "0" ou "1", com cada célula evoluindo de forma determinista, em intervalos de tempo discretos de acordo com regras definidas, envolvendo os valores de seus vizinhos mais próximos (Wolfram, 1994). A base do CAs é que o espaço é dividido em células e o tempo é discreto. Cada uma das células tem um estado inicial o qual será atualizado de acordo com regras simples (Galina, W, 2017).

Wolfram (1994), estudou e classificou regras para CAs elementares, sendo que existem 256 CAs elementares distintos. Cada regra elementar é determinada por uma sequência de oito dígitos binários para especificar um CA. As regras locais podem ser consideradas como uma função booleana das células dentro da vizinhança.

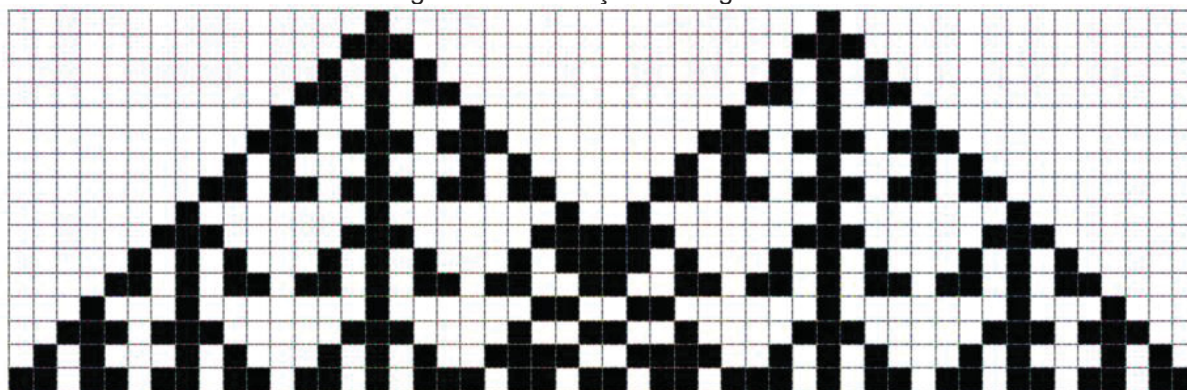
A Figura 38 ilustra a identificação do autômato elementar chamado de Regra 150.



Fonte: Wolfram (1994).

Como exemplo, na Regra 150, o estado das células no próximo passo de tempo é encontrado pela função de atualização $s^+ = s^- \oplus s \oplus s^+$, onde s^+ é o novo estado, s o estado anterior, s^- o estado da célula à esquerda, s^+ o estado da célula à direita e $\oplus$ indica a adição módulo 2. A evolução da Regra 150 com 51 células e 16 iterações, porém com apenas duas células com estado 1 é mostrada na Figura 39 (Galina, W, 2017).

Figura 39 – Evolução da Regra 150.



Fonte: Galina, W (2017).

Wolfram acredita que existe uma teoria fundamental da física que estabelece que programas simples, processando modelos computacionais, são capazes de reproduzir em detalhe tudo o que acontece no Universo. Com os autômatos celulares consegue-se simular sistemas, definidos com regras muito simples, que apresentam comportamentos muito complexos (Wolfram, 2002). Os autômatos celulares (ACs) constituem um dos tipos mais simples de sistemas dinâmicos discretos no tempo, no espaço e nos valores, tendo sido introduzidos inicialmente por John von Neumann. A partir dessa concepção, os ACs passaram a ser analisados sob diferentes perspectivas, incluindo propriedades estruturais, algébricas, dinâmicas e comportamentais (Kantic *et al.*, 2024). Em função dessas características, sua aplicação tornou-se ampla e multidisciplinar, sendo utilizado na área médica, biológica, econômica, química, entre outras (Sanz, Urquia e Leva, 2014), (Kantic *et al.*, 2024).

Autômatos celulares são simples idealizações matemáticas de sistemas naturais, e constituem métodos de uso geral com propósito para simular sistemas complexos (Wolfram, 2002). Um sistema complexo é aquele cujas propriedades não podem ser completamente explicadas pelo entendimento das suas partes constituintes. A ciência dos sistemas complexos aborda as maneiras pelas quais os seus componentes mais elementares dão origem ao comportamento coletivo emergente do seu todo (Magalhães e Souza Cruz, 2021).

Os autômatos celulares consistem em uma rede de células discretas e idênticas que são caracterizadas localmente por um conjunto de variáveis. Em função das regras que caracterizam o fenômeno analisado, em intervalos regulares de tempo, todas as variáveis são atualizadas simultaneamente. Autômatos celulares podem ser considerados como idealizações de equações diferenciais parciais usadas para

descrever sistemas naturais (Kier, Seybold e Cheng, 2005), formado por um número finito de elementos, chamado células, caracterizado por determinadas propriedades (como: cor, forma, etc.), que evolui com o tempo, mudando o estado das células simultaneamente conforme uma função de estado particular.

Autômatos celulares probabilísticos são processos markovianos em tempo discreto descritos por um conjunto de variáveis estocásticas que residem em células de um reticulado. Em intervalos de tempo regulares, todas as variáveis são atualizadas simultaneamente de acordo com regras probabilísticas (Tomé e Oliveira, 2014). Um processo é markoviano quando a probabilidade condicional de uma variável tomar um valor no próximo intervalo de tempo depende somente do valor que tenha assumido no instante anterior.

Em 1986, Frisch, Hasslacher e Pomeau (Galina, W., 2017) demonstraram que um simples autômato celular que satisfaz apenas as leis de conservação em nível celular é hábil para reproduzir a complexidade dos escoamentos de fluidos reais. Com um autômato celular definido desta forma e usando a equação de Boltzmann para determinar a evolução temporal da rede de células define-se o método do reticulado de Boltzmann (LBM). A equação de Boltzmann (Kremer, 2005) é uma equação íntegro-diferencial para a função de distribuição de um conjunto de partículas com um determinado operador de colisão (termo da integral). O LBM é uma categoria de autômato celular usado especificamente para simular escoamentos de fluidos.

O método do reticulado de Boltzmann é um esquema numérico baseado em equações cinéticas formuladas em uma escala mesoscópica que simula a dinâmica do fluido em uma escala macroscópica. A dinâmica deste método é dada pela equação do reticulado de Boltzmann, a qual mostra como ocorrem as distribuições das partículas. Ao comparar as equações governantes do LBM e do LGCA, percebe-se a familiaridade entre elas. Isto porque a equação do LBM é oriunda da equação governante do esquema LGCA, fazendo a substituição das variáveis Booleanas que representam as distribuições de partículas n_α , presentes na equação do LGCA, pelas funções reais de distribuição de partículas $f_\alpha = \langle n_\alpha \rangle$, onde $\langle . \rangle$ representa uma média por amostra, negligenciando-se o movimento individual das partículas (Cargnelutti, 2017).

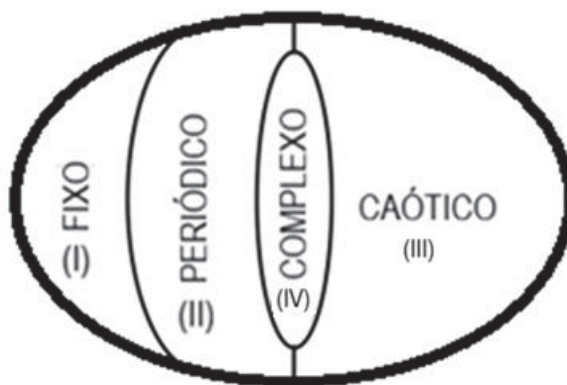
Na literatura existem comprovações sobre a aplicação de autômatos celulares para simular uma grande variedade de fenômenos, como por exemplo: fluxo de tráfego de pedestres e veículos; evacuação de ambientes; mecânica estatística; ecologia

hidráulica (evolução de diversos tipos de algas), escoamentos unidimensionais em tubos e canais; hidrodinâmica em três dimensões (simulação da circulação de águas em oceanos); na formação de padrões em biologia; escoamentos com matérias complexas (em forma granular, fluido-fluido ou sólido-fluido); escoamentos térmicos e escoamentos térmicos multifásicos, com mudança de fase; erosão por cavitação em instalações industriais; processos com reações químicas (Kantic *et al.*, 2024), (Sanz, Urquia e Leva, 2014).

A história do autômato celular é relativamente recente, sendo desenvolvido por John von Neumann e Stanislaw Ulam no início dos anos de 1950. Com o Jogo da Vida de John Conway, apresentado na *Scientific American* em outubro de 1970, o assunto se popularizou (Schiff, 2008).

Stephen Wolfram (1959-) é parcialmente responsável pelo ressurgimento do interesse nos autômatos celulares, realizando modelagens em mecânica estatística, após a conclusão do seu doutorado, em 1979. Testando o comportamento dos autômatos celulares, Wolfram concluiu que os autômatos celulares podem ser usados para a modelagem de sistemas adaptativos complexos, organizando o comportamento dos autômatos celulares em quatro classes: 1) o sistema converge para um atrator pontual independentemente da configuração inicial; 2) o sistema estabiliza numa estrutura periódica simples, como se fosse atraído por um atrator periódico; 3) o sistema torna-se caótico, mudando de forma continuamente e de maneira não-periódica; e 4) são produzidas estruturas localizadas, que se propagam, crescem, separam-se e se recombina de forma complexa. Christopher Gale Langton (1948-), concluiu que do ponto de vista de sistemas dinâmicos as classes estariam na seguinte ordem: $(1) \rightarrow (2) \rightarrow (4) \rightarrow (3)$, com comportamento seguindo o padrão: ordem $\rightarrow$ complexidade $\rightarrow$ caos. Langton denominou este comportamento de "margem do caos", com os autômatos celulares produzindo um comportamento entre instabilidade e estabilidade (Lewin, 1993). A Figura 40 ilustra as quatro classes do AC e sua ordem.

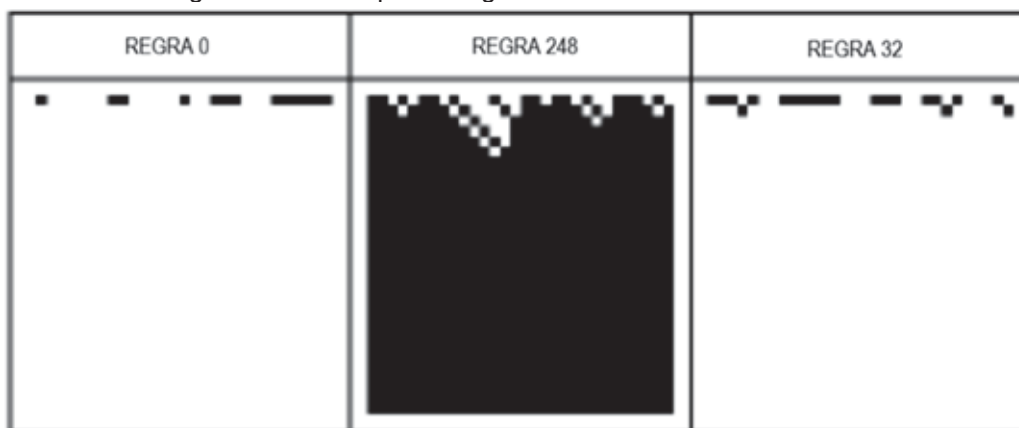
Figura 40 – As quatro classes do AC.



Fonte: Adaptado Carvalho D. de S. (2011).

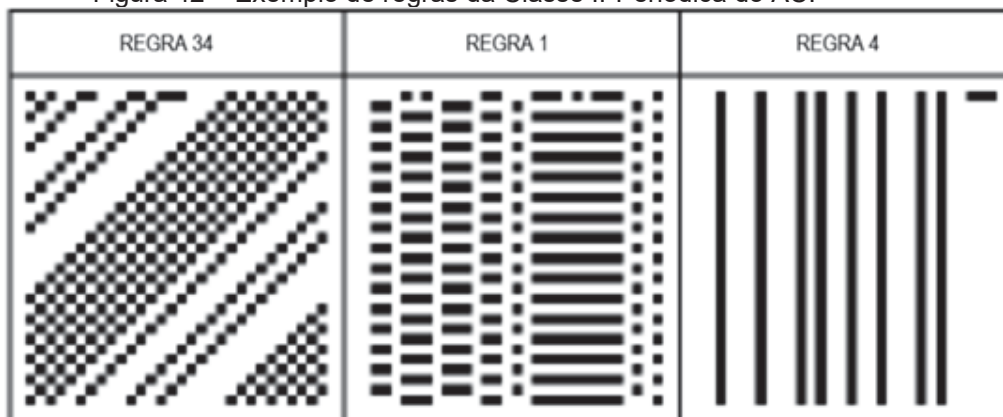
A Figura 41 ilustra três exemplos de regras da Classe I do AC, a Figura 42 três exemplos de regras da Classe II do AC, Figura 43 três exemplos de regras da Classe IV do AC e a Figura 44 três exemplos de regras da Classe III do AC.

Figura 41 – Exemplo de regras da Classe I-Fixa do AC.



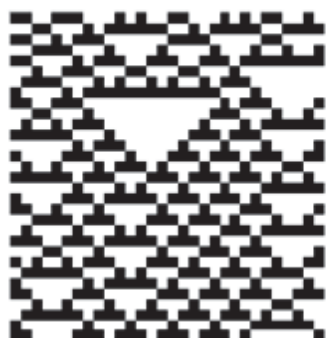
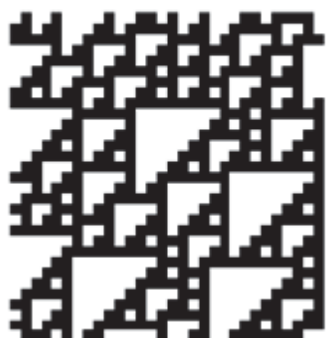
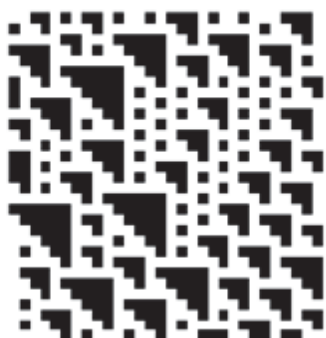
Fonte: Carvalho D. de S. (2011).

Figura 42 – Exemplo de regras da Classe II-Periódica do AC.



Fonte: Carvalho D. de S. (2011).

Figura 43 – Exemplo de regras da Classe IV-Complexo do AC.

REGRA 54	REGRA 110	REGRA 193
		

Fonte: Carvalho D. de S. (2011).

Figura 44 – Exemplo de regras da Classe III-Caótico do AC.

REGRA 90	REGRA 30	REGRA 165
		

Fonte: Carvalho D. de S. (2011).

Uma métrica λ foi definida para medir como regras diferentes de caóticas se comportam. Lambda é uma medida da fração das possíveis configurações iniciais que resultem num estado diferente da imobilidade. O estado de imobilidade pode ser escolhido como algum estado simples e válido para o autômato celular. A medida $\lambda = 0$ caracteriza um autômato celular que permanece na imobilidade não dependendo da configuração inicial, enquanto a medida $\lambda = 1$ identifica um autômato celular completamente caótico (Bye, 2016).

Sukop e Thorne-Jr. (2006) resumem a ideia geral de um autômato celular, sendo que um autômato celular ocupa uma posição sobre um ponto do reticulado de formato especificado no espaço e interage com seus vizinhos. Para cada passo de tempo, o autômato celular examina seu próprio estado e os estados dos seus vizinhos mais próximos. Em seguida, no próximo passo de tempo e de acordo com as regras utilizadas, ocorre a redefinição do seu próprio estado. Assim, as regras e as condições iniciais e de contorno impostas ao grupo de autômatos celulares determinam unicamente a sua evolução no tempo. Os autômatos celulares podem ser n -

dimensionais, sendo os mais comuns os unidimensionais e os bidimensionais. Os autômatos unidimensionais são mais simples e podem ser representados por um vetor de células, e cada geração posterior pode ser armazenada em uma matriz, onde cada linha (ou coluna) representa a atualização do vetor em um passo de tempo.

Os autômatos celulares são construções matemáticas nas quais os domínios de espaço, estado e tempo são discretos, contrastando com as equações diferenciais parciais em que esses três domínios são contínuos. Assim, em problemas mais complexos, os modelos baseados em AC podem contornar os problemas encontrados em métodos mais clássicos de modelagem (Silva, Da e Freire, 2015).

O autômato celular é composto por um número de células dispostas em uma malha regular, sendo que cada uma das células usa um conjunto de estado finito, intitulado S . O estado de uma célula se altera conforme o estado das células vizinhas e as regras estabelecidas (Subrata, 2022).

Com o estudo sobre autômatos celulares, Wolfram concluiu que com programas simples consegue-se explicar sistemas naturais complexos e entender como a natureza cria esses comportamentos complexos. Por exemplo, uma das consequências de um fluido quando ele escoar rapidamente é a turbulência. A turbulência é aparentemente aleatória, não seguindo nenhum padrão. Wolfram sugere que é possível responder por que fluidos produzem essa aparente aleatoriedade, usando autômatos celulares. Alguns estudiosos da teoria do caos, dizem que essa aparente aleatoriedade na turbulência dos fluidos é uma consequência de detalhes nas condições iniciais. Segundo Wolfram, para que esta afirmação seja verdadeira é necessário explicar por que as condições iniciais são aparentemente aleatórias. Wolfram descobriu que há um mecanismo básico que mostra que fluidos podem produzir aleatoriedade sozinhos, sem a necessidade de nenhum fator externo que os crie. Desenvolveu programas que, apesar de terem condições iniciais muito simples, acabam apresentando comportamentos tão complicados que podem parecer aleatórios. A aparente aleatoriedade encontrada nesses programas simples inspirou Wolfram a acreditar que esse efeito, também pode ocorrer em um fluido.

Formalmente um autômato celular pode ser definido por uma quintupla (Weenen, Van Der, 2014): (T, S, s, N, F) , sendo T um reticulado (ou mosaico) num espaço Euclidiano $\mathbb{R}^d$, constituído por células c_i , com $i, d \in \mathbb{N}$; S é um conjunto finito com h estados, com $S \in \mathbb{N}$; A função de saída $s: T \times \mathbb{N} \rightarrow S$, relaciona o valor do estado na célula c_i no instante de tempo discreto t , ou seja $s(c_i, t)$; N é uma função de

aproximação (ou de vizinhança), $N: T \rightarrow \bigcup_{q=1}^{\infty} T^q$, mapeando cada célula c_i numa sequência finita conforme Equação (5.1):

$$N(c_i) = (c_i)_{j=1}^{|N(c_i)|}, \quad (5.1)$$

consistindo $N(c_i)$ o conjunto de vizinhos da célula c_i ; $|N(c_i)|$ representando o número vizinhos da células c_i ; e $F = (f_i)_{i \in \mathbb{N}}$ uma família de funções, $f_i: S^{|N(c_i)|} \rightarrow S$, com cada f_i governando a dinâmica da célula c_i , conforme Equação (5.2):

$$s(c_i, t + 1) = f_i[\tilde{s}(N(c_i, t))]. \quad (5.2)$$

Com:

$$\tilde{s}(N(c_i, t)) = (s(c_i, t))_{j=1}^{|N(c_i)|}. \quad (5.3)$$

O reticulado T é uma coleção de entidades espaciais definidas num espaço Euclidiano. Em uma dimensão os elementos são definidos em segmentos sobre linhas retas ou curvas, com comprimentos iguais ou distintos. Em duas dimensões os elementos podem ser definidos com retângulos, triângulos, hexágonos ou com formatos irregulares. Em três dimensões os elementos podem ser definidos com prismas regulares ou irregulares. Cada um dos mosaicos de T consiste em uma célula c_i .

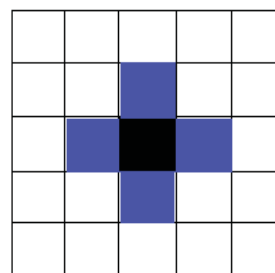
As funções de aproximação, $N(c_i, t)$, são usadas nas funções f_i , para modificar o estado de cada célula, para o próximo instante de tempo discreto $t + 1$. As funções de aproximação definem um conjunto de células consideradas próximas da célula que deverá ter o seu estado atualizado. Na Figura 45 são apresentados alguns exemplos das células que são consideradas próximas (em azul) da célula central (em preto), que deverá ter o seu estado atualizado.

Figura 45 – Exemplos de células com seu estado atualizado.

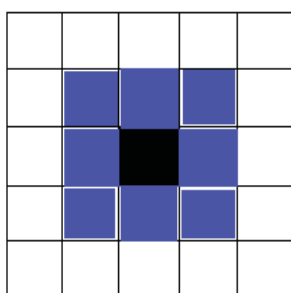
a) Unidimensional



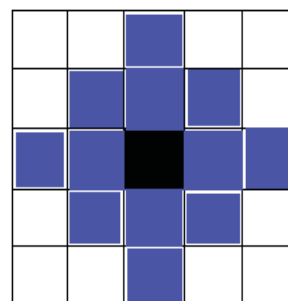
b) Bidimensional (função de Von Neumann)



c) Bidimensional (função de Moore)



d) Bidimensional (função de Moore estendida)



Fonte: Autor (2025).

5.2.1 Autômatos celulares do tipo gás em rede (LGCAs)

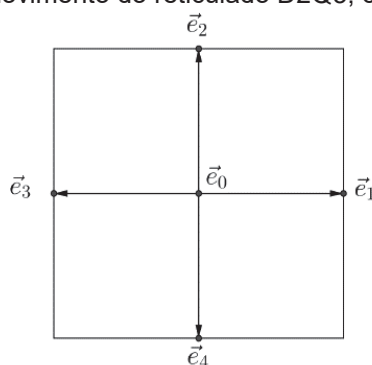
Os CAs que fazem a simulação do comportamento de um gás são chamados de autômatos celulares do tipo gás em rede (LGCAs - *Lattice Gas Cellular Automata*). Os LGCAs formam uma classe particular dos CAs, propostos para modelar fluídos, considerando duas características essenciais que são, as leis de conservação da quantidade de movimento e conservação da massa (Cargnelutti, 2017). São desenvolvidos como um modelo microscópico simples, completamente independente, para um fluido baseado em partículas fictícias dispostas em um reticulado regular. Essas partículas movem-se de um nó (ou célula) para outro da malha com velocidades dadas pelas direções do reticulado. A colisão acontece quando mais de uma partícula chega no mesmo local. Os LGCAs têm a importante característica da conservação explícita da massa e da quantidade de movimento, algo muito desejável na simulação de problemas reais. Essa característica não ocorre nos CAs (Zhou, 2004).

A dinâmica das partículas no LGCA é desenvolvida como um modelo microscópico simples, totalmente discreto para um fluido a base de partículas fictícias

localizadas nos nós de um reticulado uniforme. Ou seja, o tempo, o espaço e as partículas são discretos e caracterizados pelo reticulado uniforme. Tais partículas movem-se uma unidade da malha, em cada passo de tempo, nas direções de suas velocidades. Duas ou mais partículas que chegam ao mesmo nó podem colidir. A característica importante dos autômatos gás em rede é que a massa e a quantidade de movimento são explicitamente conservadas. Este é um recurso desejável na simulação de problemas físicos reais. E mais, pode-se mostrar que é possível recuperar as equações de Navier-Stokes assintoticamente, utilizando modelos bem simples de LGCA (Zhou, 2004).

Foi proposto por Hardy, Pazzis e Pomeau (1976) o primeiro modelo discreto para simulação de escoamentos de fluidos, baseado em uma rede quadrada com quatro possibilidade de movimento (direções cardeais). A velocidade de uma partícula é um dos quatro vetores unitários $(1, 0)$, $(0, 1)$, $(-1, 0)$, $(0, -1)$, conforme pode ser observado na Figura 46. O modelo é bidimensional com velocidades e posições discretas, ou seja, uma partícula em um determinado nó se desloca para outro nó mais próximo em uma das quatro direções do reticulado. Neste momento, ocorre a colisão de forma que o número total de partículas e o momento total são conservados. Este modelo é chamado de HPP. Cada célula contém quatro bits de informação, representando a existência ou não de uma partícula viajando em cada uma das direções cardeais (Cargnelutti *et al.*, 2017).

Figura 46 – Direções de movimento do reticulado D2Q5, com 4 velocidades não nulas.

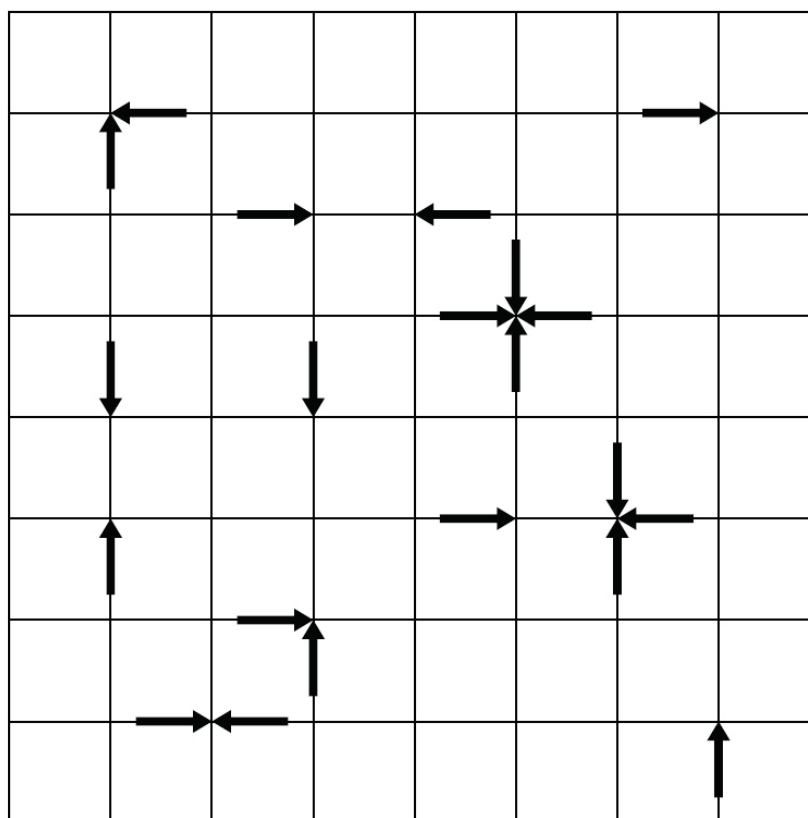


Fonte: Cargnelutti, J. (2017).

A configuração das partículas no modelo HPP, como observado na Figura 47, é condicionada por regras apropriadas de propagação e colisão (Figura 48). O conjunto $S = \{0, 1\}$ representa os estados possíveis das partículas, sendo “0” para ausência e “1” para presença.

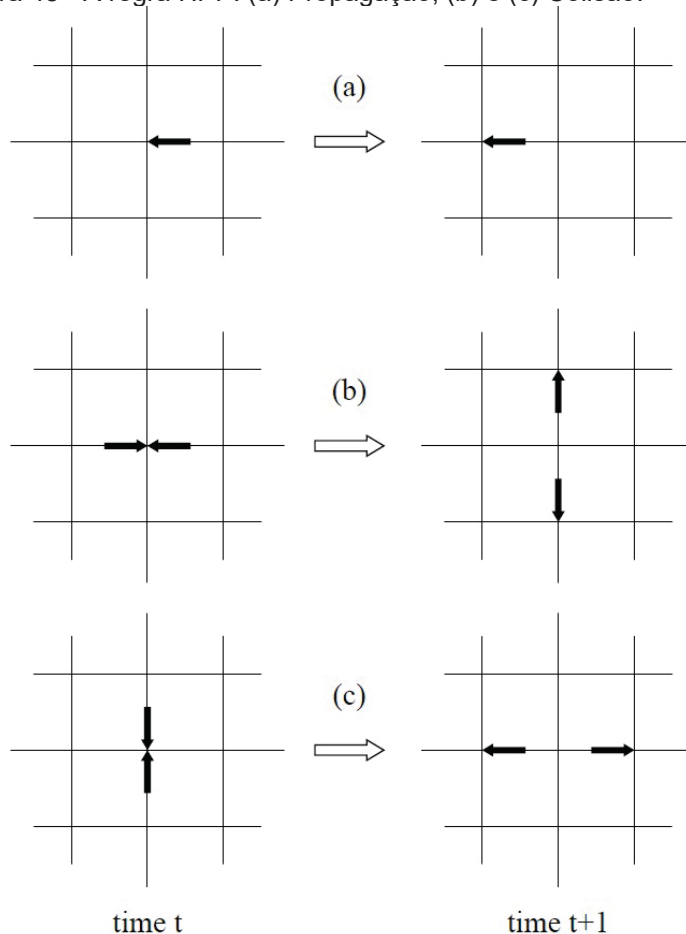
Segundo Chopard e Masselot (1999), a evolução do sistema é frequentemente dividida nas etapas de colisão e propagação. Na etapa de colisão é especificado como as partículas que estão chegando em um mesmo nó do reticulado interagem e mudam suas trajetórias. O princípio da exclusão, limita a uma única partícula chegando em um local por uma dada direção de movimento, e em um determinado instante de tempo. Na etapa de propagação, cada partícula move-se para a célula vizinha mais próxima seguindo a sua direção de movimento. A Figura 48 mostra as etapas de colisão e propagação no HPP.

Figura 47 - Exemplo de configuração de partículas no modelo HPP.



Fonte: Galina, W (2017).

Figura 48 - A regra HPP: (a) Propagação, (b) e (c) Colisão.



Fonte: Galina, W (2017).

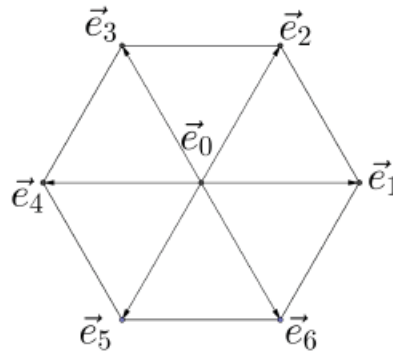
Além do fato de que o HPP não representa as equações de Navier-Stokes, devido à sua anisotropia (comparando as direções cardiais com as colaterais), o modelo HPP não representa de forma precisa o comportamento de um gás, como por exemplo, sua difusão. Ratificando a importância dos modelos baseados em autômatos celulares do tipo gás em rede e com o objetivo de resolver os problemas do modelo HPP, anos depois Frisch, Hasslacher e Pomeau (1986) descobriram as simetrias que permitem representar as equações de Navier-Stokes. Com base em uma estrutura hexagonal de seis velocidades o novo modelo é chamado de FHP (Cargnelutti, 2017).

Frisch, Hasslacher e Pomeau (1986) mostraram que um fluido poderia ser representado por partículas imaginárias, todas elas contendo a mesma massa e movendo-se com velocidade. Estas velocidades ficariam restritas a um conjunto finito (número finito de direções de movimento para uma partícula). A representação através de partículas fictícias faz com que o movimento no interior do fluido não precise ser detalhado e mesmo assim reproduz a dinâmica dos fluidos de forma real, demonstrando as simetrias que permitem representar as equações de Navier-Stokes, com base em

uma estrutura hexagonal de seis velocidades (Figura 49), com direções de movimento descritas pela Equação (5.4):

$$\vec{e}_\alpha = \begin{cases} (0,0) & \text{se } \alpha = 0 \\ e \left(\cos \left[\frac{\pi}{2} (\alpha - 1) \right], \sin \left[\frac{\pi}{3} (\alpha - 1) \right] \right) & \text{se } \alpha = 1, \dots, 6 \end{cases} \quad (5.4)$$

Figura 49 – Modelo de reticulado D2Q7, com 6 velocidades não nulas.



Fonte: Frisch, Hasslacher e Pomeau (1986).

As partículas fictícias ficam distribuídas em um reticulado regular. A movimentação dessas partículas ocorre de acordo com uma descrição simples. De acordo com Frisch, Hasslacher e Pomeau (1986), primeiro ocorre a preparação do reticulado de tal maneira que não haja mais de uma partícula movendo-se na mesma direção em uma determinada célula. Em um segundo momento ocorre a transmissão, ou deslocamento, das partículas por uma unidade do reticulado até a célula vizinha mais próxima, de acordo com a direção da sua velocidade. Por último, ocorre a colisão entre as partículas que chegarem no mesmo passo de tempo à sua célula de destino. As velocidades e as regras de colisão fazem com que as velocidades das partículas sejam alteradas. A forma como a partícula segue para uma nova célula depende das regras de colisão. Uma característica importante é que o número de partículas não é modificado e nem o vetor soma das velocidades, ou seja, a colisão preserva a massa e a quantidade de movimento.

5.2.2 Exemplo de modelo de autômato celular para águas rasas

Os modelos computacionais são de uso geral em dinâmica dos fluídos e teoria cinética dos gases, pois podem ser aplicados em regiões do escoamento em que a hipótese de continuidade não é verdadeira para uma ampla faixa do número de Knudsen.

O autômato celular do tipo gás de rede pode ser utilizado para descrever fenômenos físicos regulados apenas por leis locais, sendo aplicado com êxito em fenômenos fluidodinâmicos (Dottori e Todini, 2010).

As três características principais do autômato celular são:

- a) uniformidade: todos os estados das células são atualizados pelo mesmo conjunto de regras;
- b) sincronicidade: todos os estados das células são atualizados simultaneamente;
- c) localidade: as regras são de natureza local.

Vários esquemas computacionais para modelagem hidráulica 2D foram propostos na literatura. Os modelos de inundação são geralmente baseados em diferenças finitas, elementos finitos ou pelo método de volumes finitos.

Quando aplicada para reproduzir fluxos bidimensionais de água, a abordagem com autômato celular do tipo gás de rede origina um esquema computacional equivalente ao método de volumes finitos: a área de estudo é esquematizada usando uma grade regular de elementos poligonais (normalmente quadrados), enquanto as funções de estado que descrevem a evolução do sistema são as equações do momento e da continuidade, escritas em versão completa ou simplificada.

Dottori e Todini (2010) propôs um modelo baseado em uma versão do método de volumes finitos, baseado na abordagem de autômato celular, proposto para representar fenômenos dominados pela difusão nas equações de escoamento, utilizando uma malha poligonal com formato variável (regular ou irregular).

O modelo hidráulico proposto por Dottori e Todini (2010) foi baseado nas equações de águas rasas, escritas da forma difusiva, com a área de estudo esquematizada com uma malha de elementos poligonais, considerando duas células adjacentes, a vazão passa através da face de contato e é calculada pela Equação (5.5). Esse modelo utiliza o esquema explícito de Euler. Primeiro a vazão entre as células é calculada com a Equação (5.5):

$$Q_{i,j} = \frac{bh_m^{5/3}}{n} \left[\frac{H_i - H_j}{\Delta x} \right]^{1/2}, \quad (5.5)$$

onde H_i e H_j são o estágio da água nas duas células; Δx a distância entre o centroide de duas células; b a largura da face de contato; n o coeficiente de rugosidade de Manning; h_m a média de profundidade de água entre duas células.

Somando as vazões na entrada e na saída do i -th nó para m nós conectados, a vazão total Q é calculada pela Equação (5.6), sendo os volumes V e as profundidades da água nas células são atualizadas pela Equação (5.7):

$$Q = (V_t) = \frac{\Delta V_t}{\Delta t} = \sum_{j=1}^m Q_{i,j}, \quad (5.6)$$

$$\Delta V_{t+\Delta t} = \Delta t Q(V_t) + O(Q^2), \quad (5.7)$$

sendo $O(Q^2)$ o erro truncado do esquema.

Cada etapa no cálculo da vazão inclui o controle de volume, evitando que o volume de água que saia de uma célula seja maior que o volume armazenado na própria célula e os volumes que entram das células adjacentes.

Em Andriolo e Kaviski (2023) foi desenvolvido um modelo de autômato celular, baseado no trabalho de Dottori e Todini (2010), para simular a propagação de um hidrograma ao longo de um canal de grande largura e com pequena declividade. O hidrograma obtido utilizando o autômato celular foi comparado com o hidrograma obtido utilizando o tradicional modelo HEC-RAS, sendo os resultados entre ambos os modelos compatíveis, demonstrando a aplicabilidade do autômato celular.

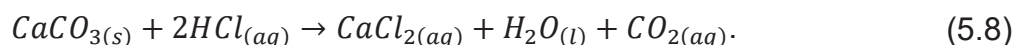
5.2.3 Exemplo de modelo de autômato celular para simulação da série temporal acumulada de massa liberada de CO₂

As duas questões fundamentais da engenharia de reações químicas, quando é dado um conjunto de espécies reativas e as condições para que ocorra a reação, são: a) determinação da composição final da mistura reacional; e b) determinação da evolução temporal e o tempo necessário para atingir o estado de equilíbrio. A primeira questão é solucionada com termodinâmica do equilíbrio químico e a segunda é resolvida com cinética química (Felder e Rousseaud, 2005).

Em geral, os problemas de cinética química podem ser solucionados realizando-se técnicas experimentais em laboratório e simulando-se com modelos matemáticos (solução de sistemas de equações diferenciais parciais) e com modelos computacionais (autômatos celulares). Em sistemas reacionais com algum componente gasoso o avanço da reação pode provocar modificações de pressão se o sistema operar a volume constante. A reação também poderá ser acompanhada pela modificação da pressão em função do tempo.

A propriedade que o carbonado de cálcio possui de se dissolver em ácido é responsável pela formação das cavernas de calcário, e também pelos efeitos destrutivos que a chuva ácida provoca em estátuas de mármore e materiais de construção (Brady J.E. e J.R., 2002).

Em Andriolo e Kaviski (2024) foi desenvolvido um método computacional para determinar, utilizando um modelo bidimensional de autômatos celulares, a série temporal acumulada da massa de dióxido de carbono CO_2 que é liberado para o ambiente durante a reação de carbonato de cálcio (CaCO_3) com ácido hidrocloreto (HCl) baseado no trabalho elaborado por Weeen, Van Der (2014) cuja reação química é apresentada na Equação (5.8):



Em sua tese Van Der Weeen (2014), apresenta a série temporal acumulada da massa de dióxido de carbono (CO_2) liberada para o ambiente, da reação apresentada na equação acima, que foi ensaiada em laboratório, à temperatura e pressão ambientes. A perda de peso de CO_2 foi medida durante 565 segundos, sendo no início feita uma leitura a cada 5 ou 10 segundos e no final do experimento a cada 100 segundos. A perda de peso máxima estequiométrica teórica devido à formação de CO_2 é de 3,3 g.

Van Der Weeen (2014) também resolveu a reação apresentada na Equação (5.8) com modelagem matemática e modelagem computacional. O autor diz que durante a execução do experimento, percebeu-se que outras fontes de perda de peso ocorreram como a evaporação ao ar livre, bem como a adesão do fluído ao béquer plástico. Essas fontes adicionais de perda de peso foram quantificadas por meio de um experimento em branco, nas mesmas condições, e somadas até 0,80 g ao final do experimento. Assim, Van Der Weeen (2014) diz que foram feitas correções na saída do modelo computacional desenvolvido por ele, utilizando a perda de peso adicional quantificada (0,8 g), adicionando esta perda de peso adicional à perda de peso simulada pelo modelo para obter uma perda de peso total simulada.

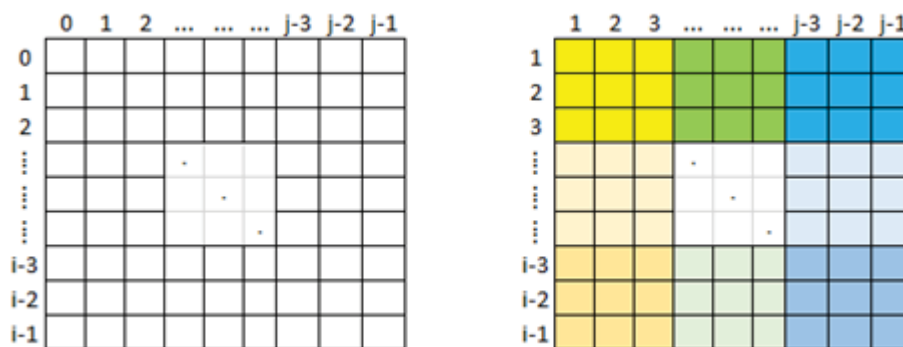
Em Andriolo e Kaviski (2024) o problema apresentado foi solucionado com um autômato celular bidimensional, programado em linguagem Go (www.golang.org), e os resultados comparados com o ensaio descrito por Van Der Weeen (2014). Sendo a mistura composta pelos seguintes componentes: (a) 7,5 g de $\text{CaCO}_{3(s)}$; (b) 0,0356 L

de $H_2O_{(l)}$; e (c) 0,0144 L com uma mistura (massa específica igual a 1100 g/L) à 23%, em termos mássicos de $HCl_{(aq)}$ (com 3,64 g de HCl). Considerando-se que a massa específica do H_2O é igual a 1000 g/L, calculou-se a massa total de H_2O na mistura, referente a quantidade de H_2O adicionada e de H_2O na solução com HCl , resultando em 47,68 g.

Em Andriolo e Kaviski (2024), foi utilizado um reticulado com dimensões de 99 x 99, totalizando 9.801 células, e outro com 999 x 999, totalizando 998.001 células, com a finalidade de calcular a série temporal de massa de dióxido de carbono (CO_2), sendo adotada as seguintes etapas para solucionar esse problema:

- Os estados das células são inicializados com quantidades molares iguais, distribuídos aleatoriamente, com um dos reagentes, $CaCO_3$, HCl ou H_2O . A quantidade de células para cada um dos reagentes é definida em função da massa total de cada um dos reagentes, que são adicionados na mistura. Os estados de cada célula, contém quantidades molares iguais, para somente um dos reagentes.
- O reticulado é subdividido em blocos de 3 x 3 células, conforme apresentado na Figura 50.

Figura 50 – Divisão do mosaico em blocos de 3 x 3 células.



Fonte: Andriolo e Kaviski (2024).

Em cada um dos blocos, com 3 x 3 células, a função de aproximação de Moore é aplicada quando um dos reagentes, $CaCO_3$ ou HCl , for o estado presente na célula central.

- Para simular o processo difusivo que ocorre durante a reação química, os estados das células, em cada bloco, deverão ser embaralhados aleatoriamente, como ilustrado na Figura 51.

Figura 51 – Simulação da difusão e redistribuição dos elementos nas células.

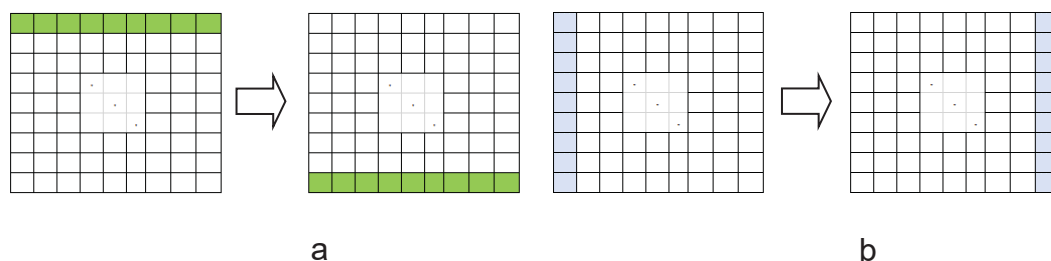


Fonte: Andriolo e Kaviski (2024).

Nesta figura, o quadrado em branco representa o estado de uma célula com H_2O ; em azul representa o estado de uma célula com $CaCO_3$; e em verde representa o estado de uma célula com HCl .

- d) Recompôr o reticulado para o próximo instante de tempo simulando a realização da reação química, em cada bloco. Se no bloco analisado, o estado das outras células completarem a reação (se na célula central o estado atual é $CaCO_3$, nas proximidades deverão existir pelo menos duas células com o estado HCl ; ou se na célula central o estado atual é HCl , nas proximidades deverão existir pelo menos uma célula com o estado HCl e pelo menos uma célula com o estado $CaCO_3$), o estado da célula central e das outras células, contendo os reagentes para ocorrer a reação, terão os seus estados atualizados para o próximo instante, com H_2O . Quando a reação química ocorrer num dos blocos, deve-se computar a quantidade em mol, de CO_2 (g), para compor a série temporal acumulada deste produto.
- e) Como as reações químicas são simuladas somente em função do estado da célula central de cada bloco, as células deverão ser agitadas, usando-se alguma regra de mobilização, como esquematizado na Figura 52.

Figura 52 – Mudanças que ocorrem com o modelo baseado em bloco CA.



Legenda: (a) mudança da linha superior para a inferior e (b) mudança da coluna totalmente da esquerda para a direita. Fonte: Andriolo e Kaviski (2024).

Nestas figuras representam-se os deslocamentos que as linhas (figura (a)) e as colunas (figura (b)) do reticulado, deverão ser submetidos, de $n-2$ linhas e de $n-2$ colunas, respectivamente, sendo n o número total de linhas ou colunas.

Andriolo e Kaviski (2024) concluíram que a utilização de autômatos celulares para simular sistemas complexos é uma ferramenta promissora, sendo utilizado para solucionar diversos tipos de problemas em uma e duas dimensões. Neste trabalho, a série temporal acumulada de massa de dióxido de carbono (CO_2) liberada para o ambiente, durante a reação de carbonato de cálcio (CaCO_3) com ácido hidrocloreto (HCl), foi simulada utilizando o autômato celular proposto, sendo os resultados obtidos compatíveis com os resultados do experimento em laboratório utilizado como referência.

5.3 MODELOS MATEMÁTICOS PARA ESCOAMENTOS COM “BOLHAS”

O livro preparado por Brennen (1995) pode ser considerado como um texto de referência para a realização de estudos de modelagem matemática sobre cavitação, dinâmica de “bolhas” e escoamentos multifásicos. Fazem parte do livro os seguintes assuntos: mudança de fase, nucleação e cavitação; dinâmica de “bolhas” esféricas; colapso de “bolhas” esféricas; dinâmica da oscilação de “bolhas”; translação de “bolhas”; escoamento de “bolhas” homogêneas; escoamentos com cavitação; e escoamentos com superfície livre.

Um dos capítulos do livro editado por Oertel (2004) trata de escoamentos multifásicos. Neste texto enfatiza-se a necessidade do estudo de “bolhas” e gotas no movimento de líquidos e gases na representação da velocidade macroscópica do escoamento. Por meio de análises heurísticas concluem que a velocidade relativa das “bolhas” e gotas depende da interação entre as fases, da interação com os contornos e com o efeito de forças externas (gravidade, resistência ao escoamento e tensão superficial). Desta forma, definem-se uma relação funcional para a velocidade relativa das “bolhas” e gotas que pode ser usada, por meio de analogias (Incropera *et al.*, 2007), do suporte das equações de conservação e de resultados obtidos experimentalmente, para estimar parâmetros para o dimensionamento de obras hidráulicas. Oertel (2004) apresenta também uma figura proposta por Clift, Grace e Weber em 1978, que classifica as formas das “bolhas” que ascendem nos líquidos em função dos números de Reynolds, Eötvös e de Morton.

Schmidt (2015) propôs um método computacional para prever o risco de erosão por cavitação em instalações industriais. O método é fundamentado num esquema de volumes finitos aplicado em 3 dimensões numa malha formada por hexaedros. No modelo adotado considera como variável a massa específica da

mistura que é definida como uma combinação linear entre a massa específica da fase líquida e da massa específica da fase gasosa. As equações governantes (conservação de massa, quantidade de movimento e energia total) são em 3 dimensões espaciais e dependentes do tempo. Considera os efeitos inerciais e devido à resistência ao escoamento, negligenciando a transferência de calor. Os resultados obtidos foram validados comparando com dados experimentais. O método pode ser aplicado em todos os regimes de ocorrência de cavidades.

Papoulias e Gavaises (2015) investigaram os efeitos no comportamento térmico e de pressão na solução de diferentes modelos de Rayleigh-Plesset (Fuster, 2019) na simulação da dinâmica da expansão e colapso de “bolhas”. Os resultados numéricos foram validados com observações obtidas experimentalmente.

Ho (2016) pesquisou sobre métodos para a modelagem da representação e discretização da geometria de escoamentos multifásicos usando um método superficial (*interface tracking*) e um método volumétrico (*interface capturing*). Avaliou o desempenho do método interface tracking na simulação de escoamentos bifásicos, contendo “bolhas”, através da comparação dos resultados obtidos com dados experimentais.

Niedzwiedzka (2017) analisou as formulações matemáticas usadas para representar o comportamento físico dos termos fonte nas equações de transporte nos modelos que descrevem o fenômeno da cavitação. Avaliou os seguintes modelos: Schnerr e Sauer de 2001; Simghal, Athavale e Jiang de 2002; e de Zwart, Gerber e Belamri de 2004.

Fuster (2019) apresenta uma revisão de modelos para representar o escoamento de grupos de “bolhas” com o objetivo de estudar o fenômeno da cavitação. A seguir apresenta-se um resumo das questões analisadas e avaliadas: a) descrição qualitativa do fenômeno físico: a pressão interna relativa e a velocidade de expansão da “bolha” decresce, enquanto que a velocidade térmica e os efeitos da difusividade mássica aumentam devido a agitação induzida durante os estágios iniciais do processo; b) equações básicas: além das equações de conservação (massa, quantidade de movimento e energia) para cada fase do escoamento, são consideradas uma equação para representar a convecção da posição da interface, equações de balanço locais para estabelecer as propriedades na interface e uma relação empírica (modelo de Hertz-Knudsen-Langmuir) para representar o fluxo de vaporização; c) dinâmica de uma “bolha” solitária: considerando simetria esférica e os

efeitos dos fluxos de transferência de calor e de massa calor através da interface pode ser usada a equação modificada de Rayleigh-Plesset para modelar a evolução temporal do raio da “bolha”; d) modelagem de grupos de “bolhas”: a representação do escoamento multifásico é dificultada pela impossibilidade da captura das escalas do escoamento; sendo baixa a concentração de “bolhas” a equação modificada de Rayleigh-Plesset pode ser usada; no contexto de escoamentos turbulentos grupos de “bolhas” podem ser modeladas por meio de funções probabilísticas, aplicando o conceito de ensemble para definir quantidades médias, e considerar o movimento relativo entre “bolhas” e o líquido, como por exemplo a formulação de Van Wijngaarden’s, que relaxa em algumas hipóteses; e) modelagem de grupos de “bolhas” contendo alta concentração de gás ou vapor: o modelo modificado de Rayleigh-Plesset não é adequado; a evolução temporal da fração gás/vapor pode ser obtida pela solução de uma equação difusiva-advectiva que considera a existência de “bolhas” localizadas em alguns pontos do escoamento por meio da função delta de Dirac; e f) modelos para escoamentos turbulentos multifásicos com cavitação: existem dificuldades para extrapolar o uso dos modelos clássicos que representam os escoamentos de turbulência de uma fase para os escoamentos multifásicos; as principais dificuldades estão relacionadas com a geração de turbulência devido a mudança de volume produzida pela compressibilidade do gás/vapor e a mudança de fase, e também pela turbulência induzida pela presença de interfaces; um modelo que considera a geração de vorticidade causada pela interface foi proposto por Lundgren e Koumoutsakos, em 1999.

5.4 GERADOR DE “BOLHAS” ESTOCÁSTICO

Valero et al. (2019) expandiu para três dimensões o modelo estocástico de geração de “bolhas” (*Stochastic Bubble Generator* – SBG) desenvolvido por Kramer et al. (2019) e Bung D. e Valero D. (2017). Neste artigo, o autor apresenta os resultados da quantidade de bolhas geradas, em torno de 2%, e comenta que o modelo computacional foi desenvolvido em MATLAB.

Para simular a cinemática de “bolhas” sujeitas a um campo de velocidade turbulento, a flutuação de velocidade foi modelada como um movimento browniano seguindo a Equação (5.9) de Langevin (1908) citado por Valero et al. (2019):

$$du_i(t) = u_i(t) \frac{dt}{T_i} + \left(\frac{2\sigma_i^2}{T_i} \right)^{\frac{1}{2}} dW(t), \quad (5.9)$$

sendo σ_i a escala da flutuação de velocidade; T_i a escala de tempo característico do processo estocástico, definida pela função de autocorrelação de velocidade ($\rho_i(s) = \exp(-s/T_i)$), sendo: $T_i = \int_0^\infty \rho_i(s) ds$.

Para reproduzir a dinâmica das “bolhas” utilizando o SBG, algumas estimativas de u_i , σ_i e T_i devem ser desenvolvidas, com base nos escoamentos da camada limite. Quando um fluido viscoso encontra uma superfície sólida, a condição de não deslizamento leva a um gradiente de velocidade, resultando em um escoamento na camada limite. O coeficiente de atrito da superfície em escoamento de vertedouros pode ser calculado utilizando a Equação (5.10) apresentada por (Castro-Orgaz, 2010):

$$C_f^{-1/2} = 3,967 \log \left(\frac{\delta}{K_s} \right) + 6,7, \quad (5.10)$$

em que K_s é o coeficiente de rugosidade do concreto (1 a 1,5mm); δ a espessura da camada limite dada pela Equação (5.11) (Pinto, 1987):

$$\frac{\delta}{x} = 0,0212 \left(\frac{x}{H_s} \right)^{0,11} \left(\frac{x}{K_s} \right)^{-0,1}, \quad (5.11)$$

onde H_s é a carga em relação ao ponto x .

Para estimar a escala das flutuações de velocidade longitudinal, vertical e transversal: σ_x , σ_y e σ_z , respectivamente, no intervalo $0,10 < y/\delta < 1$, Valero et al. (2019) utilizou a expressão semiempírica (5.12) proposta por Nezu (1977):

$$\frac{\sigma_i}{u_*} = D_i \exp \left(-K_i \frac{y}{\delta} \right), \quad (5.12)$$

sendo D_i e K_i os coeficientes apresentados na Tabela 5; u_* a velocidade de cisalhamento; y a profundidade; δ a espessura da camada limite.

Tabela 5 – Coeficientes D_i e K_i para cálculo da flutuação de velocidade.

	D_x	K_x	D_y	K_y	D_z	K_z
Melhor Ajuste	2,22	0,84	1,11	0,66	1,35	0,69
Mínimo	2,18	0,8	1,06	0,58	1,33	0,65
Máximo	2,26	0,88	1,15	0,75	1,38	0,73
R2	0,983		0,978		0,898	

Fonte: Valero et al. (2019).

6 DESENVOLVIMENTO DO CÁLCULO DO TDG

6.1 ESTUDO DE CASO

Os dados utilizados nesta tese têm como base os estudos desenvolvidos para construção do defletor da UHE Colíder e o projeto de P&D (Takenobu, 2022), intitulado “Metodologia para Modelagem Computacional de TDG na Água em Fluxos Efluentes de Vertedouros”.

6.1.1 UHE Colíder

A UHE Colíder, com potência instalada de 300 MW, situa-se no rio Teles Pires, a 680 km da foz, entre os municípios de Nova Canaã do Norte e Itaúba, no Estado de Mato Grosso. Localizada nas coordenadas geográficas 10° 59' 5,40" latitude Sul e 55° 45' 57,60" longitude Oeste, ou 635.070 L e 8.785.362 S no sistema de coordenadas UTM. O início do enchimento do reservatório ocorreu em fevereiro de 2017 e no ano de 2019 iniciou-se a operação comercial.

As características principais do arranjo desenvolvido para o Projeto Básico da UHE Colíder estão apresentadas na sequência:

Vazões

Série Histórica de Vazões Médias Mensais:	Jan/1931 a Dez/2008
Vazão Média de Longo Termo:	1.013 m³/s
Vazão de Projeto do Vertedouro – Cheia Máxima Provável:	6.935 m³/s
Vazão de Desvio do Rio - TR = 50 anos:	4.108 m³/s

Níveis de água de montante

N.A. Máximo <i>Maximorum</i> de Montante:	272,00 m
N.A. Máximo Normal de Montante:	272,00 m
N.A. Mínimo Normal de Montante:	272,00 m

Níveis de água de jusante circuito principal

N.A. Máximo <i>Maximorum</i> de Jusante:	258,72 m
N.A. Máximo Normal de jusante:	250,32 m
N.A. Mínimo de jusante:	247,54 m

Circuito de Geração Principal

Potência Instalada Casa de Força:	300 MW
Número de Unidades Geradoras:	3
Tipo de Turbina:	Kaplan, eixo vertical
Queda Bruta:	21,68 m
Queda Líquida de Referência:	20,22 m
Queda Líquida de Projeto:	21,14 m
Vazão Nominal Unitária:	538,10 m³/s

6.1.2 Descrição geral do arranjo

O arranjo geral da UHE Colíder consiste em estruturas dispostas ao longo de um eixo retilíneo, com extensão total de 1.525,30 m, que pode ser definido a partir de dois pontos, situados em cada uma das margens, cujas coordenadas são N 8.785.170,882; E 635.223,717 (margem esquerda) e N 8.786.516,202, E 634.504,929 (margem direita). A topografia da região do eixo selecionado condiciona a distribuição das estruturas, que estão concentradas na margem direita, significativamente mais extensa e suave que a da margem esquerda.

Com crista na elevação 276,00 m, o barramento propicia a formação de um reservatório com 182,8 km² de área e volume total de 1.525 hm³ no seu nível máximo normal, correspondente à elevação 272,00 m (conforme dimensionamento do Vertedouro), igual ao nível máximo *maximorum* e abrange os municípios de Itaúba, Nova Canaã do Norte, Colíder e Cláudia, no Estado do Mato Grosso.

A partir da margem esquerda, onde a topografia mais íngreme aproxima a ombreira da calha do rio, o arranjo das estruturas apresenta a barragem do leito do rio, que avança por cerca de 1.126,00 m até o muro de encosto situado na margem oposta e adjacente à estrutura vertente.

O vertedouro possui 4 vãos com 16,70 m de altura e 12,00 m de largura cada, com controle de vazão formado por comportas tipo segmento, extensão total de 69,00 m e crista na elevação 255,30 m.

Separadas do vertedouro por um muro divisor, a tomada de água e a casa de força formam uma estrutura única, composta por 3 blocos estruturalmente independentes de 28,20 m cada, totalizando comprimento de 84,60 m por 76,51 m de largura, que abrigam 3 conjuntos de máquinas do tipo Kaplan. Cada conjunto é servido por 3 vãos de tomada de água com 6,60 m de largura e 15,01 m de altura, possui 3

vãos no tubo de sucção com 6,73 m de largura por 13,61 m de altura. À direita dos blocos, cujas máquinas somam 300 MW de potência instalada, encontra-se a área de montagem, com dimensões iguais a 56,40 m por 51,16 m.

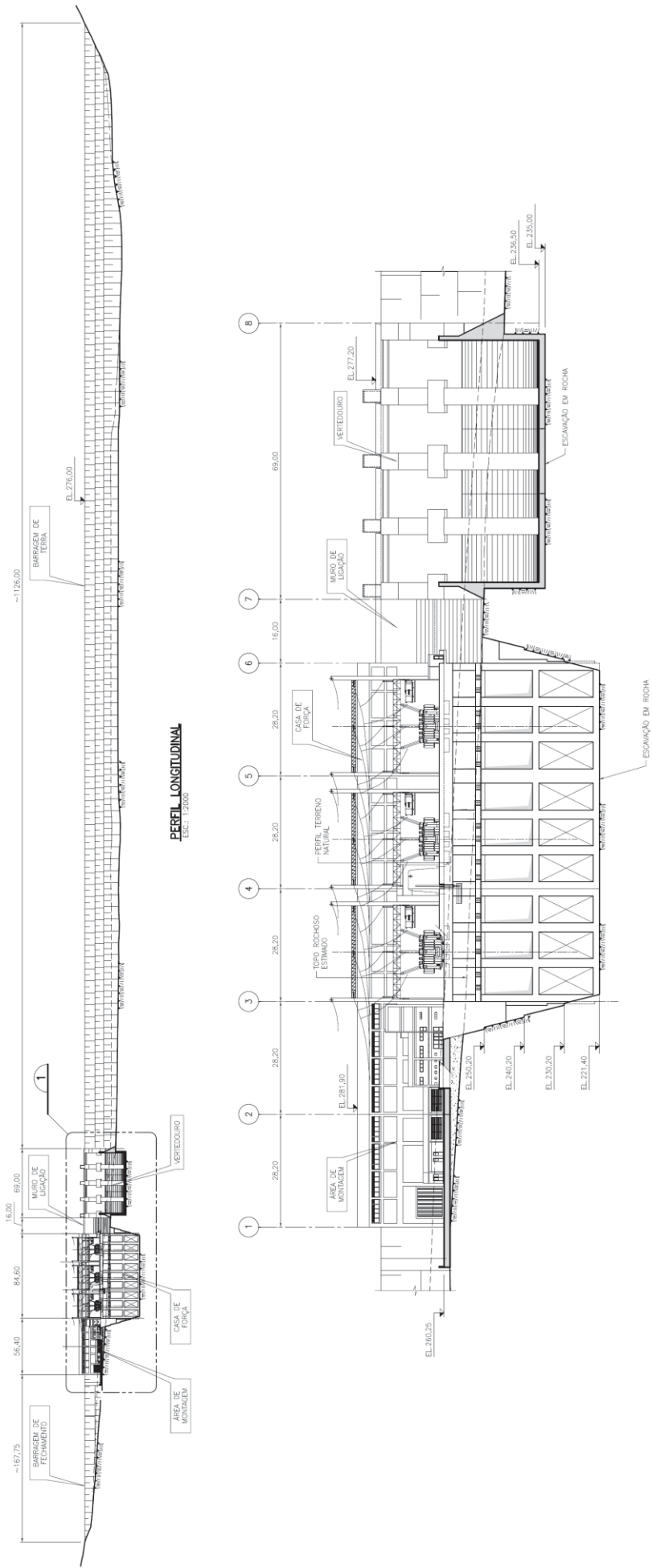
Um muro de encosto com uma extensão de 51,16 m, adjacente à área de montagem, recebe a barragem da margem direita que, com uma extensão de 174,19 m, proporciona o fechamento daquela ombreira. A Figura 53 apresenta o arranjo geral da UHE Colíder.

6.1.3 Barragem

O arranjo da UHE Colíder apresenta, na região do barramento, solução composta de barragem de terra e muros de encosto em concreto. O barramento tem coroamento na elevação 276,00 m, estando ilustrado na Figura 54. Esta cota é suficiente para que a barragem acomode a cheia de projeto (cheia máxima provável - CMP), que atinge o nível máximo *maximorum* na elevação 272,00 m.

Dessa maneira, para a condição de cheia de projeto, tem-se uma borda livre de 4,00 m, para o acomodamento de ondas no reservatório.

Figura 54 – Barragem da UHE Colider.



NETAL DE 1

Fonte: Projeto Executivo.

6.1.4 Vertedouro

O vertedouro está localizado na margem direita entre a barragem de terra no leito do rio e as estruturas da casa de força e tomada de água.

O vertedouro tem quatro vãos, cada um com 12 m de largura livre e crista na elevação 255,30 m, controlado por comportas tipo segmento com 16,70 m de altura. Os vãos estão separados por pilares com 4,20 m de espessura resultando numa largura total da estrutura de controle de 69,00 m. No sentido do fluxo o vertedouro tem um comprimento de 63,70 m. O comprimento total do vertedouro (incluindo bacia de dissipação) é de 200,15 m. O canal de aproximação do vertedouro fica na elevação 246,00 m, sendo rebaixado nas proximidades da estrutura para a elevação 240,00 m, enquanto o canal de restituição está na elevação 245,00 m, conforme ilustrado na Figura 55.

6.1.5 Circuito de Geração

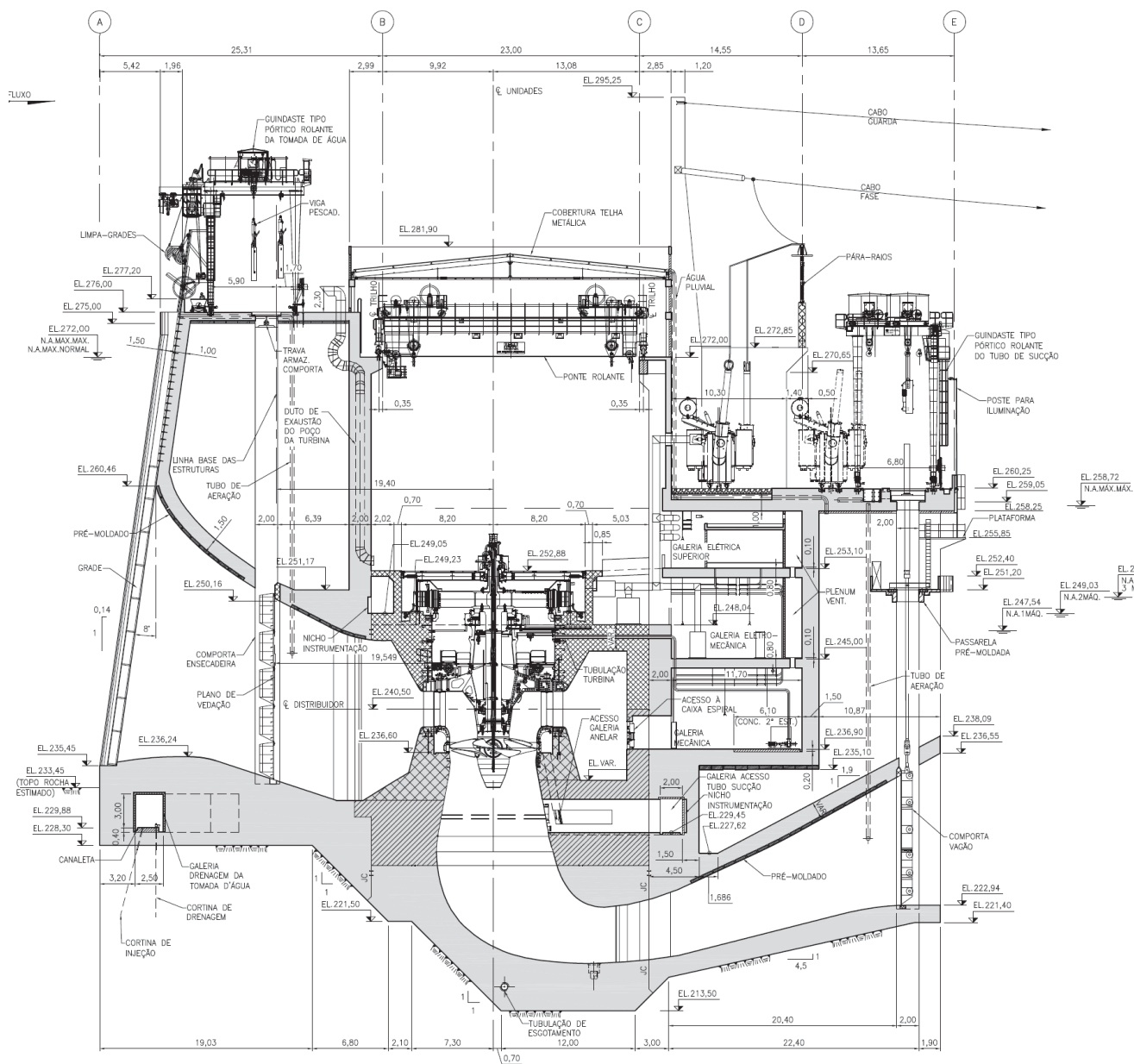
O circuito de geração é composto pelas estruturas da tomada de água, da casa de força e da área de montagem. O circuito fica na margem direita, localizado entre a barragem de terra de fechamento da margem direita e o vertedouro.

O conjunto da tomada de água / casa de força abriga três unidades geradoras com potência nominal de 100 MW cada e tem uma extensão, no sentido transversal ao fluxo, de 84,60 m, dividido em 3 blocos de unidades com 28,20 m de largura. A altura máxima dos blocos das unidades é de 63,70 m.

O mencionado conjunto da tomada de água / casa de força é de concreto convencional armado e, para fins de descrição, está subdividido em três blocos, de montante para jusante, estruturalmente interligados: o bloco de montante constituído pela tomada de água e os dois blocos de jusante formando a estrutura da casa de força.

A Figura 56 apresenta o corte transversal da Casa de Força.

Figura 56 – Casa de força.



CORTE TRANSVERSAL PELO EIXO DAS UNIDADES 1, 2 E 3

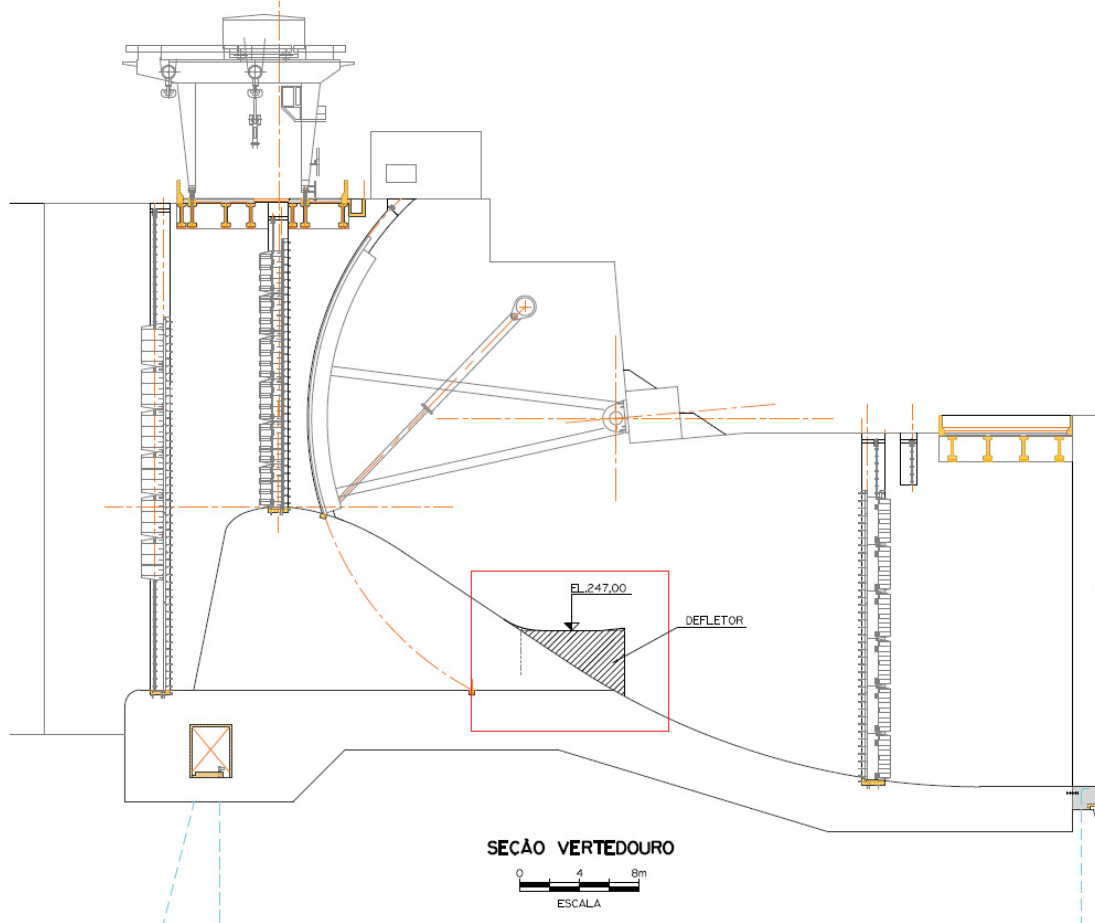
ESC, 1:200

Fonte: Projeto Executivo.

6.1.6 Geometria final do defletor

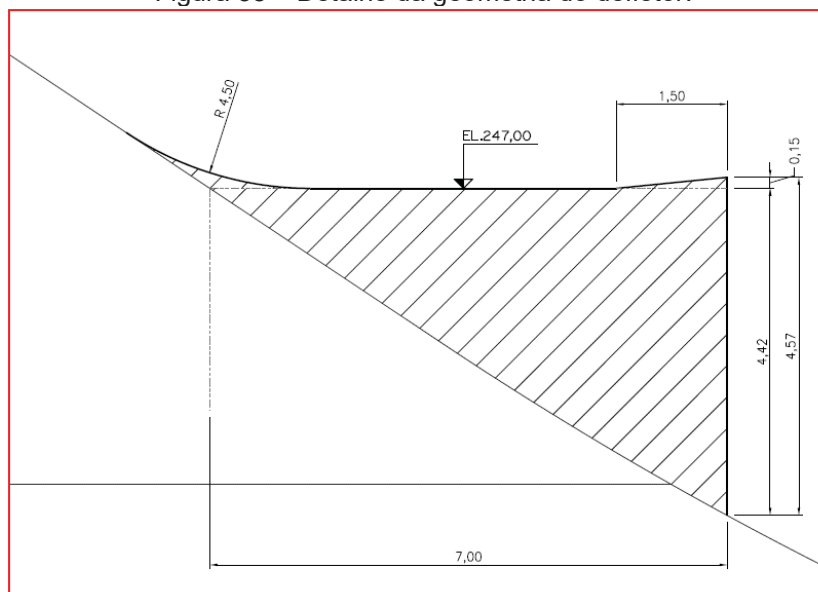
Concluída a modelagem computacional e os ensaios finais em MHR, o defletor com 7 m de comprimento e crista na elevação 247,00 m foi definido como a melhor geometria para mitigar os riscos à ictiofauna. A Figura 55 apresenta o projeto do vertedouro da UHE Colíder, sendo apresentada a geometria do defletor na Figura 57 e o detalhe do defletor na Figura 58.

Figura 57 – Geometria do vertedouro com o defletor.



Fonte: Projeto Executivo.

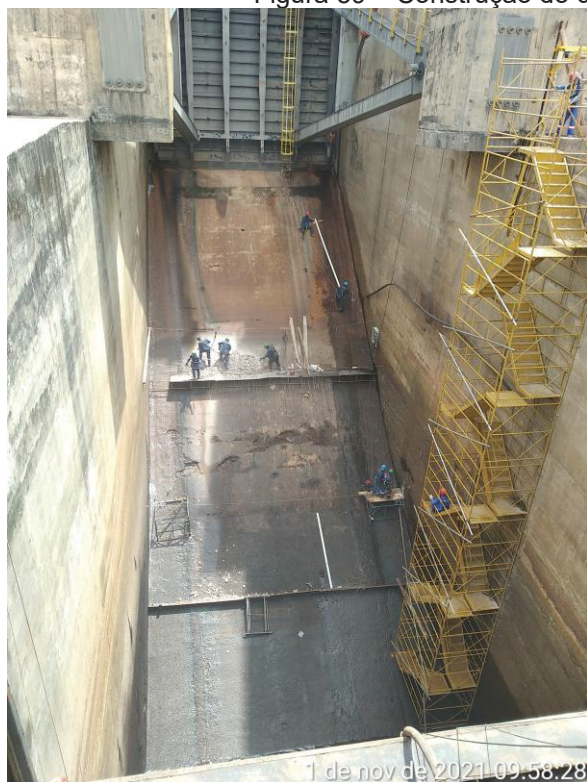
Figura 58 – Detalhe da geometria do defletor.

**DETALHE DEFLETOR**

Fonte: Projeto Executivo.

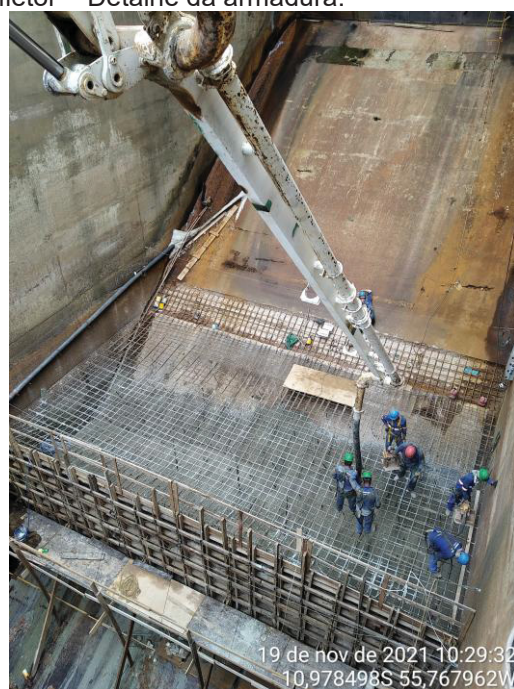
Entre a Figura 59 e a Figura 62 são ilustradas algumas das etapas de construção do defletor da UHE Colíder, com preparação da calha do defletor, montagem das formas, lançamento do concreto e cura. A Figura 63 ilustra o defletor após construção.

Figura 59 – Construção do defletor – Montagem da forma.



Fonte: Autor (2025).

Figura 60 – Construção do defletor – Detalhe da armadura.



Fonte: Autor (2025).

Figura 61 – Construção do defletor – Concretagem.



Fonte: Autor (2025).

Figura 62 – Construção do defletor – Concretagem.



Fonte: Autor (2025).

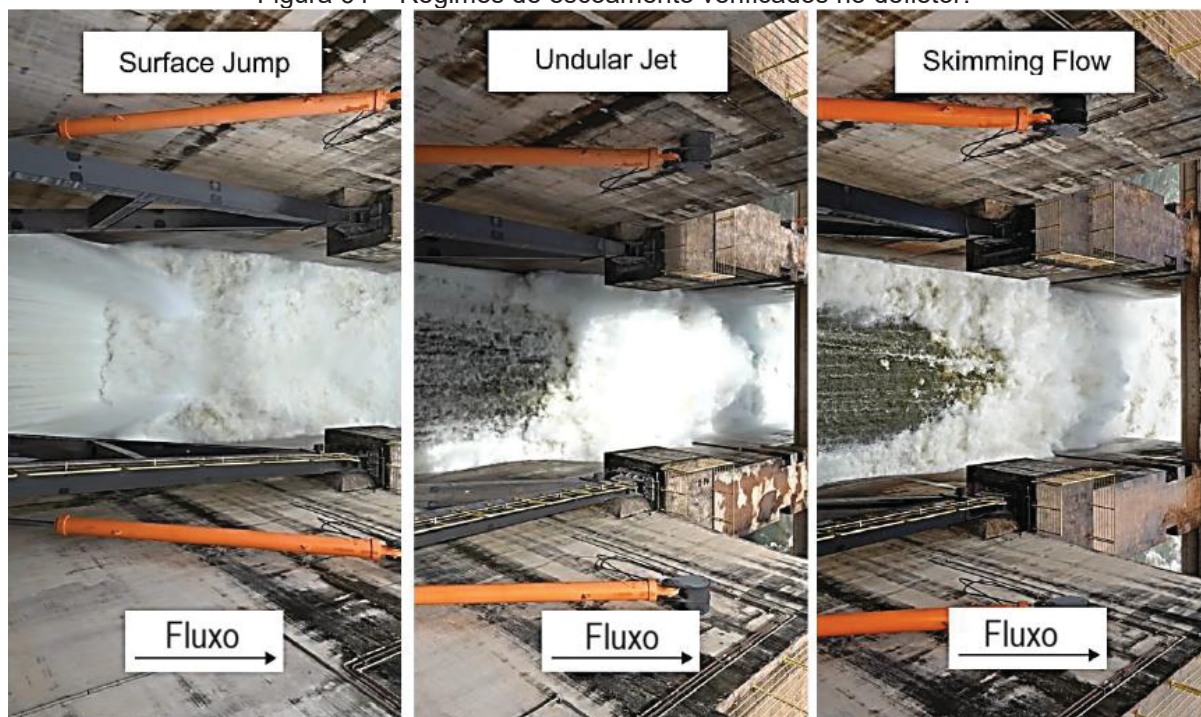
Figura 63 – Defletor após construção.



Fonte: Autor (2025).

A Figura 64 ilustra três tipos de regimes de escoamentos verificados no vertedouro da UHE Colíder após a construção do defletor.

Figura 64 – Regimes de escoamento verificados no defletor.



Fonte: Ovelar et al. (2022).

6.2 DADOS HIDRODINÂMICOS

O estudo de caso utilizado no projeto de P&D (Takenobu, 2022) intitulado “Metodologia para Modelagem Computacional de TDG na Água em Fluxos Efluentes de Vertedouros”, cujo dados de velocidade do fluxo, pressão do escoamento e volume de ar do escoamento foram utilizados nesta tese, consistem na simulação do fluxo no vertedouro com as configurações hidráulicas apresentadas na Tabela 6, para o caso do vertedouro com o defletor e na Tabela 7 para o caso do vertedouro sem o defletor.

Tabela 6 – Configurações hidráulicas e resultados – Com defletor.

Vazão (m/s)	N.E.R (m)		N.A.J. (m)		C _d	
	CFD	Físico	CFD	Físico	CFD	Físico
300	272,36	272,13	247,22	247,13	0,7241	0,7289
300	272,34	272,05	250,97	250,89	0,7244	0,7306
1600	271,73	271,97	250,11	250,21	0,7009	0,6953
1600*	271,79	271,98	253,14	253,22	0,6993	0,6950

Sendo: N.E.R. – nível da linha de energia; N.A.J. – nível de água de jusante; C_d – Coeficiente de descarga. * caso utilizado. **Fonte:** Takenobu (2022).

Tabela 7 – Configurações hidráulicas e resultados – Sem defletor.

Vazão (m/s)	N.E.R (m)		N.A.J. (m)		C _d	
	CFD	Físico	CFD	Físico	CFD	Físico
281	271,71	271,76	250,70	250,76	0,7326	0,7315
1600*	271,78	271,98	253,15	253,21	0,6995	0,695

Sendo: N.E.R. – nível da linha de energia; N.A.J. – nível de água de jusante; C_d – Coeficiente de descarga. * caso utilizado. **Fonte:** Takenobu, (2022).

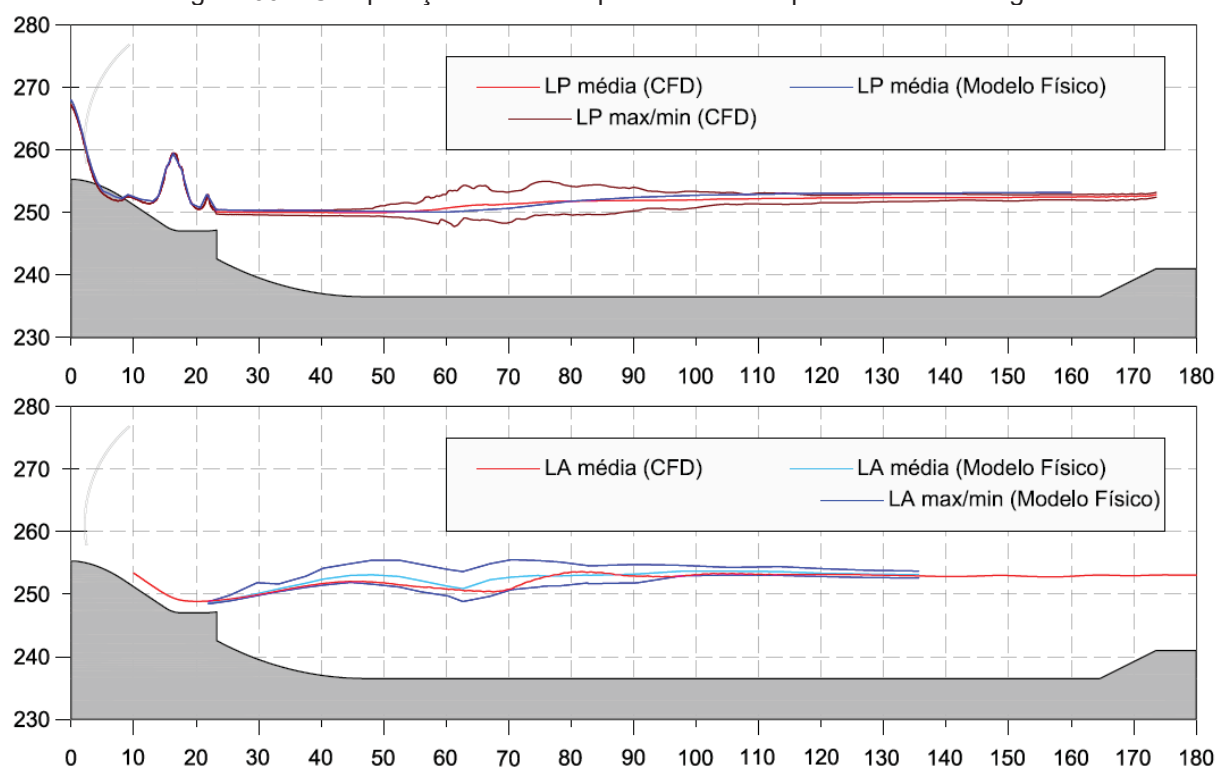
A Figura 65 apresenta o escoamento a jusante do defletor registrado no modelo físico. A Figura 66 apresenta o comparativo das linhas piezométricas e perfil de linha de água obtida na simulação computacional apresentada em Takenobu (2022) e as compara com os resultados obtidos em modelo hidráulico reduzido (escala 1:15). A Figura 67 apresenta o perfil médio da concentração de ar calculada utilizando o software OpenFOAM e compara com os resultados obtidos em modelo físico. Essas figuras apresentam os resultados para a simulação da vazão de 1.600 m³/s e NAJ = 253,21 m.

Figura 65 – Escoamento a jusante do defletor.



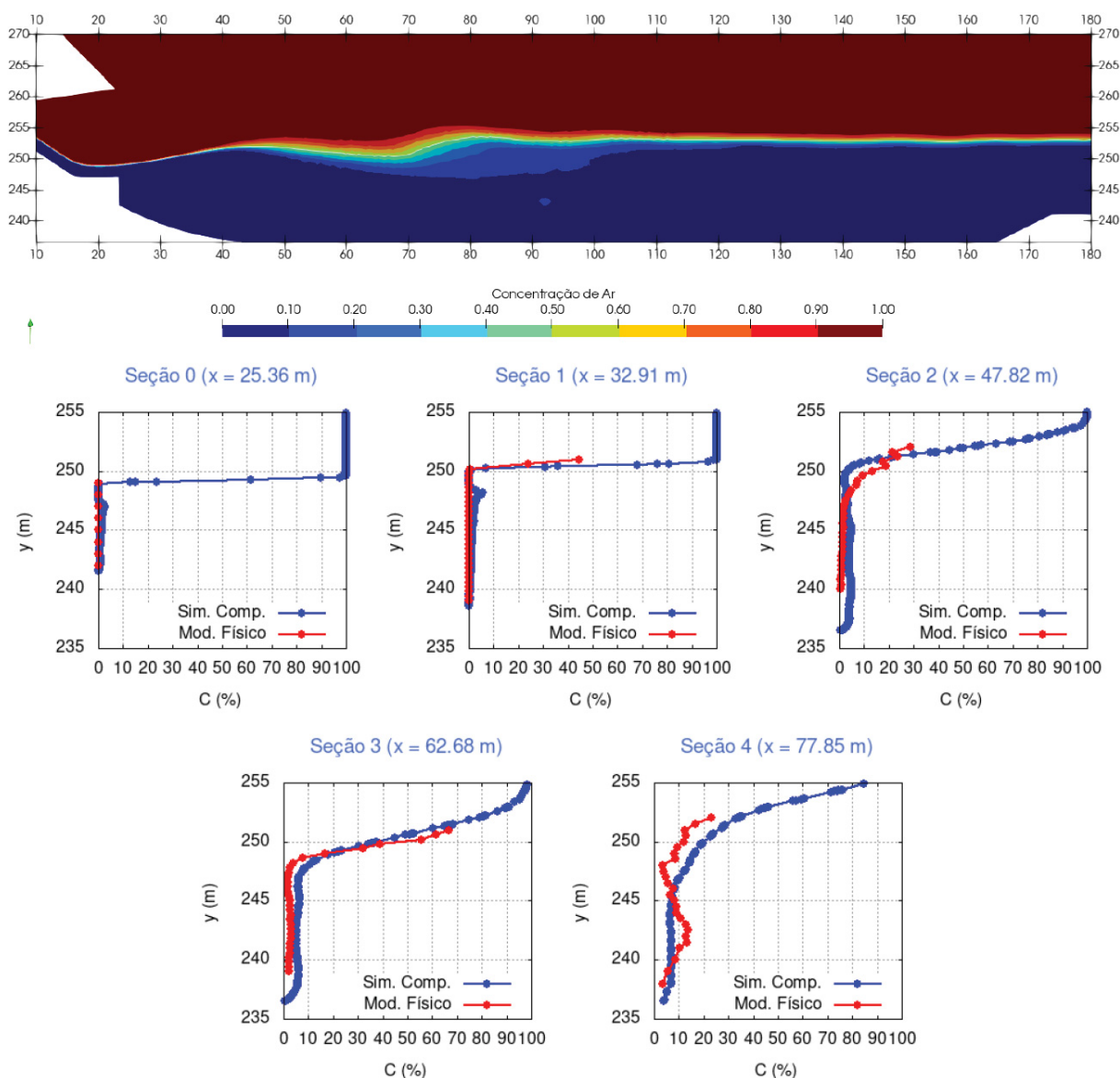
Fonte: Takenobu (2022).

Figura 66 – Comparação das linhas piezométricas e perfil de linha de água.



Fonte: Takenobu (2022).

Figura 67 – Perfil médio da concentração de ar.



Fonte: Takenobu (2022).

6.3 CÁLCULO DO VOLUME DE AR

Nesta tese, foi adotado o volume de ar na mistura ar-água, conforme os resultados obtidos no estudo de P&D (Takenobu, 2022).

O volume de ar no escoamento (Takenobu, 2022) foi obtido utilizando o software OpenFOAM, com o solver interFoam, simulação DES e o modelo de turbulência $k-\omega$ SST, sendo adotado o valor máximo do número de Courant igual a 0,5 nas simulações. O solver InterFoam utiliza o método VOF (*Volume of Fluid*) para simular a incorporação de ar no escoamento.

6.3.1 Solver InterFoam

O algoritmo de duas fases (*solver*) InterFoam é baseado no método VOF (*Volume of Fluid*), no qual uma equação de transporte é utilizada para determinar a relação entre a fração de volume relativa das duas fases (ar e água), em cada uma das células computacionais, resolvendo o escoamento de dois fluidos incompressíveis, imiscíveis e isotérmicos. As propriedades físicas são calculadas como médias ponderadas com base nessa fração. A natureza do método VOF significa que uma interface entre os fluidos não é explicitamente computada, mas sim surge como uma propriedade do campo de fração de fase. Como a fração de fase pode ter qualquer valor no intervalo entre 0 e 1, a interface nunca é nitidamente definida, mas ocupa um volume ao redor da região onde uma interface nítida deve existir. O InterFoam utiliza o algoritmo PIMPLE (*Pressure-Implicit Method for Pressure-Linked Equations*) e uma forma modificada para o termo de pressão (Greenshields, 2019). O PIMPLE combina as vantagens do PISO (*Pressure Implicit Splitting of Operators*) para os cálculos transientes e do SIMPLE (*Semi-Implicit Method for Pressure-Linked Equations*) para o estado estacionário.

6.3.2 Modelo de turbulência $k-\omega$

O modelo de turbulência $k-\omega$, com duas equações de transporte adicionais, utiliza k e a dissipação específica ω para o cálculo da turbulência. A variável ω deve ser interpretada como o inverso da escala de tempo na qual o processo de dissipação ocorre (Andersson *et al.*, 2012).

Considerando os modelos RANS, existem dois tipos de $k-\omega$: o $k-\omega$ padrão e o $k-\omega$ SST. Nesta seção são detalhadas as características do modelo $k-\omega$ SST (*Shear Stress Transport*), apresentado por Menter (1993). A Equação (6.1) apresenta o cálculo da viscosidade cinemática turbulenta:

$$\mu_t = \frac{\rho \alpha_1 k}{\max(\alpha_1 \omega, |S| F_2)}, \quad (6.1)$$

sendo α_1 (adimensional) uma constante igual a 0,31; $|S|$ a relação auxiliar dada pela Equação (6.2) ; F_2 a relação auxiliar; ω (1/kg/s) a dissipação específica dada pela Equação (6.3):

$$|S| = \sqrt{2 \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right)}, \quad (6.2)$$

$$F_2 = \tanh \left[\left[\max \left(\frac{2\sqrt{k}}{\beta \omega y}, \frac{500\nu}{y^2 \omega} \right) \right]^2 \right]. \quad (6.3)$$

As equações de transporte de k e ω são, respectivamente as equações (6.4) e (6.5):

$$\frac{D(\rho k)}{Dt} = \tau_{ij} \frac{\partial \bar{u}_i}{\partial x_j} - \beta^* \rho k \omega + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_k \mu_t) \frac{\partial k}{\partial x_j} \right], \quad (6.4)$$

$$\frac{D\rho\omega}{Dt} = \frac{\gamma}{\nu_t} \tau_{ij} \frac{\partial \bar{u}_i}{\partial x_j} - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + 2\rho(1 - F_1) \sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, \quad (6.5)$$

sendo τ_{ij} a tensão turbulenta (Reynolds).

A Tabela 8 apresenta as relações auxiliares e coeficientes de fechamento.

Tabela 8 - Coeficientes e parte das relações de fechamento do modelo $k-\omega$ SST.

Constante/Relação	Valor/Equação
$CD_{k\omega}$	$\max \left(2\rho\sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_j}; 10^{-20} \right)$
F_1	$\tanh \left\{ \min \left[\max \left(\frac{\sqrt{k}}{0,09\omega y}; \frac{500\nu}{y^2 \omega} \right) \frac{4\rho\omega_2 k}{CD_{k\omega} y^2} \right]^4 \right\}$
β	$\beta_1=0,075; \beta_2=0,0828$
β^*	0,09
σ_k	$\sigma_{k1}=0,075; \sigma_{k2}=0,0828$
σ_ω	$\left(\frac{\beta}{\beta^*} - \gamma \right) \sqrt{\beta^*} / k^2$

Fonte: Menter (1993).

Andersson et al. (2012) compararam os modelos padrões $k-\varepsilon$ e $k-\omega$ nas regiões próximas a contornos sólidos. Nessas regiões, notou-se um melhor desempenho do modelo $k-\omega$ padrão em comparação com o modelo $k-\varepsilon$ padrão. Contudo, segundo Versteeg e Malalasekera (2007), os valores de turbulência (k e ω) na entrada do domínio influenciam consideravelmente a representação de escoamentos livres no modelo $k-\omega$ padrão. Por isso, sugere-se o uso do modelo $k-\omega$ SST, o qual se baseia nos dois modelos padrões, aplicando o modelo $k-\omega$ padrão nas

regiões próximas aos contornos, e $k-\varepsilon$ padrão no restante do escoamento, sendo pouco influenciável pelos valores de entrada aplicados na simulação.

6.4 MALHA COMPUTACIONAL

No estudo apresentado por Takenobu (2022) cujos dados de velocidade do fluido, relação ar-água e pressão foram utilizados nesta tese, as malhas computacionais foram elaboradas utilizando o gerador de malha, *blockMesh*.

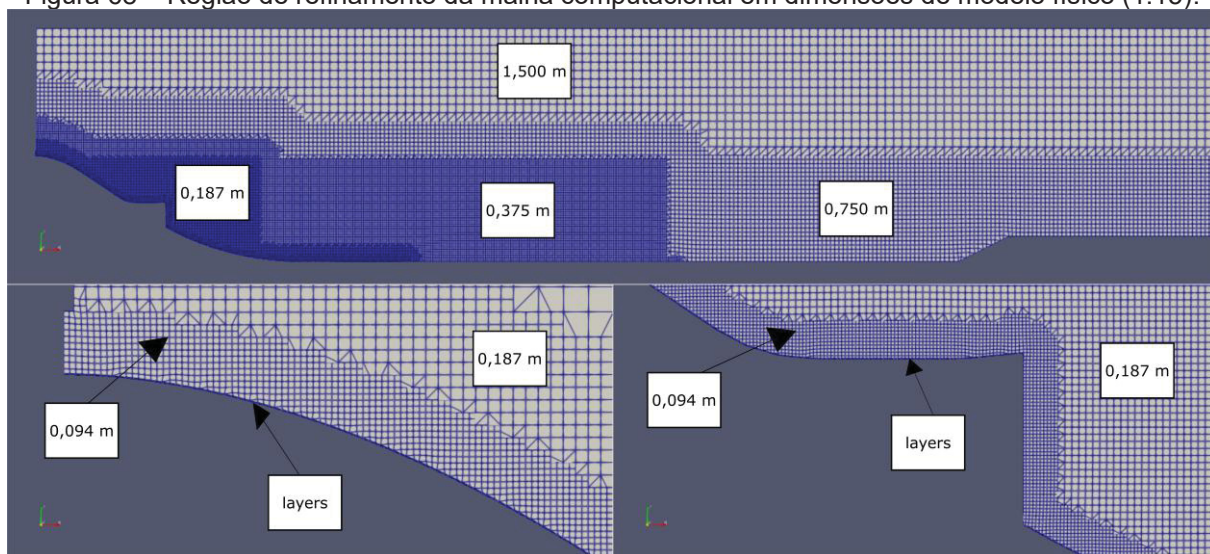
As dimensões de comprimento, largura e altura dos domínios computacionais em dimensões de protótipo foram: 690,00 m x 8,10 m x 45,00 m, respectivamente. Os domínios em dimensão do modelo hidráulico reduzido (1:15) seguiram o critério de semelhança por número de Froude (46,00 m x 0,54 m x 3,00 m).

A quantidade e o tamanho dos elementos das malhas computacionais variaram entre as diferentes configurações hidráulicas. O refinamento da malha foi realizado em cinco regiões específicas do escoamento: na abertura da comporta, na região próxima à calha do vertedouro, no defletor, na bacia de dissipação e na região próxima ao meio pilar esquerdo. Também foram adicionadas layers (elementos ortogonais e paralelos ao contorno sólido) para representar adequadamente o gradiente de velocidades do escoamento no vertedouro.

Em dimensões de modelo físico, o tamanho dos elementos variou entre 6,125 mm e 10 cm no sentido do escoamento (eixo x), 3 mm a 10 cm na altura (eixo z) e 2 cm a 14 cm na largura (eixo y). Salienta-se que somente para as configurações com vazão de 281 m³/s e 300 m³/s, foi utilizada o tamanho de 6,125 mm. Nas demais configurações, a menor dimensão em x utilizada foi de 1,25 cm. A Figura 68 apresenta as principais características da malha e o refinamento adotado em escala de protótipo.

Os tamanhos dos elementos em protótipo seguiram a escala geométrica conforme o critério de semelhança de Froude, com elementos entre 9 cm e 1,5 m no eixo x, 4,5 cm a 1,5 m no eixo z e 30 cm a 2,0 m no eixo y. Nas configurações com vazão de 281 m³/s e 300 m³/s também foram mantidos os menores tamanhos de células no eixo x, semelhante ao modelo em escala 1:15.

Figura 68 – Região de refinamento da malha computacional em dimensões de modelo físico (1:15).



Fonte: Takenobu (2022).

7 IMPLEMENTAÇÃO DO MODELO MATEMÁTICO

O objetivo deste capítulo é apresentar as etapas do modelo matemático desenvolvido em linguagem Go, para calcular o Total de Gás Dissolvido (TDG), baseado no modelo matemático apresentado em Huang et al. (2019) e descrito no capítulo 4, e nos dados hidrodinâmicos: velocidade do fluido, relação ar-água e pressão, variáveis calculadas por Takenobu (2022) e descrito na seção 6.4.

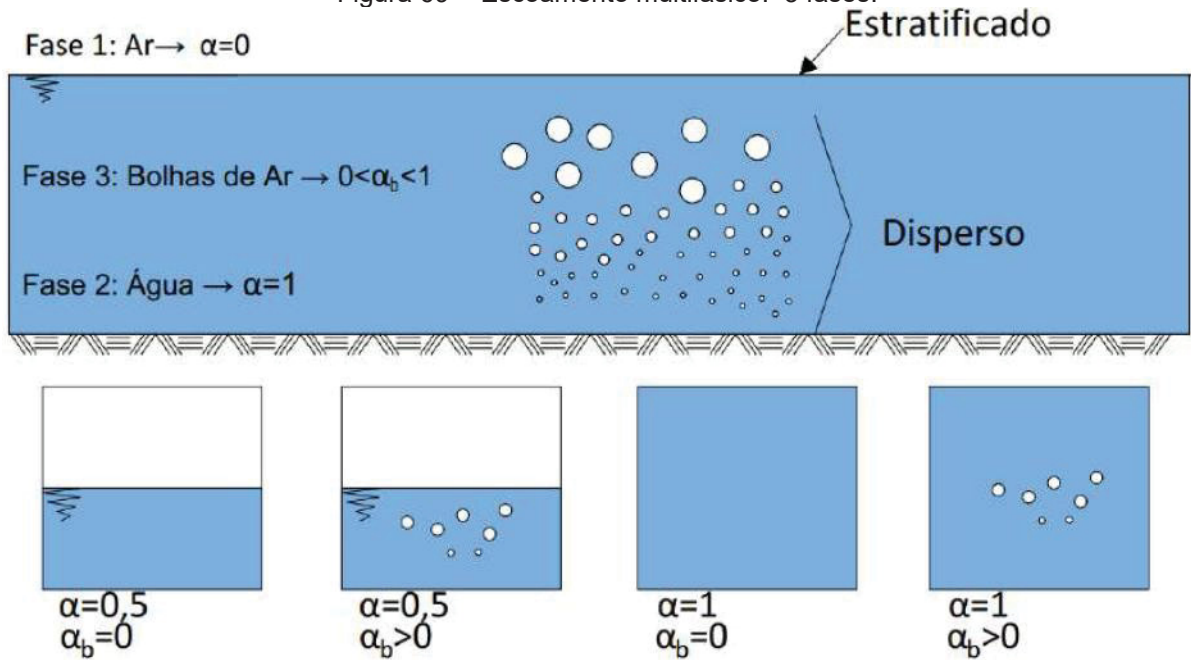
A linguagem de programação Go (também conhecida como Golang) foi desenvolvida, em 2007 (lançada em 2009), num projeto interno no Google, por Robert Griesemer, Rob Pike e Ken Thompson. Foi lançada em 2009, sendo uma linguagem de programação compilada, fortemente tipada e de código aberto (Donavan A. A. A. e Kernighan B. W., 2015).

Os programas em Go podem ser escritos em qualquer editor de textos com suporte para codificação em UTF-8. Todos os programas em Go estão contidos em pacotes. Os programas iniciam a partir do pacote *main*. Todo código executável deve estar no interior de funções e todo pacote *main* deve conter uma função *main* usada como ponto de início do programa.

Durante o desenvolvimento desta tese, especificamente na implementação de um modelo de autômato celular do tipo gás de rede, descrito na seção 5.2.2 e apresentado por Andriolo e Kaviski (2023), o programa de resolução do problema foi desenvolvido tanto na linguagem Go quanto na linguagem Python, utilizando-se a mesma sequência lógica de códigos. Nesse experimento, o problema foi resolvido em aproximadamente 30 minutos com o programa escrito em Go e em cerca de 8 horas com o programa escrito em Python. Em razão desse desempenho computacional, a linguagem Go foi escolhida para o desenvolvimento desta tese.

Os estudos de Huang et al. (2019) e Politano et al. (2007), consideram o escoamento multifásico composto por três fases: fase ar, fase líquida (α) e fase bolha (α_b), sendo um escoamento típico com estas fases ilustradas na Figura 69.

Figura 69 - Escoamento multifásico: 3 fases.



Fonte: Ovelar (2022).

O modelo desenvolvido calcula a concentração de TDG em função da concentração de equilíbrio obtida pela equação de transporte de TDG apresentada na Equação (7.1), sendo: 2º termo da equação o termo convectivo, 3º termo da equação o termo difusivo e o 4º termo da equação o termo fonte de TDG:

$$\frac{\partial(\alpha_l C)}{\partial t} + \nabla \cdot (\alpha_l \mathbf{u}_l C) = \nabla \cdot \left(\left(\nu + \frac{\nu_t}{SCc} \right) \alpha_l \nabla C \right) + S, \quad (7.1)$$

sendo α_l (adimensional) a fração volume de líquido, complementar à fase bolha; C (adimensional) a concentração de TDG em relação ao equilíbrio; $\mathbf{u}_l$ (m/s) a velocidade do líquido; SCc (adimensional) o número de Schmidt, (igual a 0,7 conforme (Wang, Politano e Weber, 2018)); ν (m²/s) a viscosidade cinemática do líquido; ν_t (m²/s) a viscosidade turbulenta do líquido; S (1/s) o termo fonte de TDG, corresponde à transferência de massa entre a fase bolha para a fase líquida. Considerando a temperatura média da água no projeto piloto (+/- 28°C) e a pressão atmosférica local (Latitude: -10.980920, Longitude: -55.770636), a Concentração de Equilíbrio adotada foi igual a 0,016915 kg/m³ (Ovelar e Takenobu, 2023).

Para estimar o valor de TDG nesta tese, foram utilizados como dados de entrada as variáveis do modelo hidrodinâmico: fração volume de líquido (α_l), velocidade do líquido ($\mathbf{u}_l$) e pressão (p), calculados no estudo de caso do projeto de

P&D (Takenobu, 2022) intitulado “Metodologia para Modelagem Computacional de TDG na Água em Fluxos Efluentes de Vertedouros”, que consistem na simulação do fluxo no vertedouro com as configurações hidráulicas apresentadas na seção 6.1.

7.1 MALHA COMPUTACIONAL E PREENCHIMENTO DE FALHAS

Os parâmetros de fração de volume de líquido, velocidade do líquido e pressão, calculados a partir do modelo hidrodinâmico desenvolvido no projeto de P&D (Takenobu, 2022), foram utilizados como parâmetros de entrada desta tese. A malha utilizada no modelo OpenFOAM por Takenobu (2022), foi ilustrada na Figura 68 e descrita na seção 6.4. Desta forma, a disponibilidade de dados varia ao longo da malha com espaçamentos de 9,4 cm até 1,5 m.

Nesta tese o domínio computacional abrange o vertedouro desde a crista, a bacia de dissipação até o término do *end sill*, totalizando 174,20 m em escala de protótipo. Adicionalmente, foi incluído um trecho de 25,80 m correspondente ao canal de restituição, resultando em um comprimento total de 200,00 m a partir da crista do vertedouro. A Figura 70 apresenta o domínio computacional adotado, o qual é composto por aproximadamente 230.000 células, formado por uma malha retangular com resolução espacial de 10 cm nos eixos x e z . O nível de água adotado foi na elevação 253,20, correspondente a uma vazão defluente total de 3.200 m³/s.

A malha bidimensional adotada nesta tese é composta por células discretizadas no eixo z , representando a profundidade da bacia, e no eixo x correspondente ao comprimento da bacia de dissipação. O eixo y foi desconsiderado, uma vez que a simulação hidrodinâmica utilizada considerou um vertedouro confinado lateralmente e uma bacia de dissipação com restrição nas bordas. Nesta condição, as componentes de velocidade lateral (eixo y) podem ser consideradas desprezíveis devido à sua baixa variabilidade ao longo da largura do escoamento.

Para compatibilizar os dados disponíveis com a malha utilizada para simular o TDG desta tese, foi necessário estimar os dados conforme a dimensão de malha escolhida para fazer as simulações.

Os dados disponíveis foram alocados nas respectivas células i , z , sendo as células vazias preenchidas adotando os seguintes critérios:

- a) Existindo dados disponíveis nas células próximas, $i-1$ e $i+1$, ou $z-1$ e $z+1$, foi adotado o valor médio para a célula, i,z , dando-se preferência pelos dados do eixo i , e caso não disponível aos dados do eixo z .

- b) Não existindo dados nas células imediatamente adjacentes, foram utilizados 6 dados das células mais próximas, respeitando o limite de 5 células no eixo i e 1 célula no eixo z ($i-5$, $i+5$, $z-1$ e $z+1$). Sempre que possível, foram utilizados os 6 dados no eixo i , caso não existam, se utiliza os dados do eixo $z-1$ e $z+1$. A partir desses dados, ajustou-se um plano pelo método dos mínimos quadrados, empregando o algoritmo de eliminação de Gauss com pivotamento total.
- c) Na ausência de dados próximos à célula a ser preenchida, todos os pontos disponíveis na malha são considerados. Nesse caso, o valor da célula é estimado pelo método do inverso do quadrado das distâncias, conforme Equação (7.2):

$$v_{i,z} = \frac{\sum_{i=1}^N w_i(x) v_i}{\sum_{i=1}^N w_i(x)}, \quad (7.2)$$

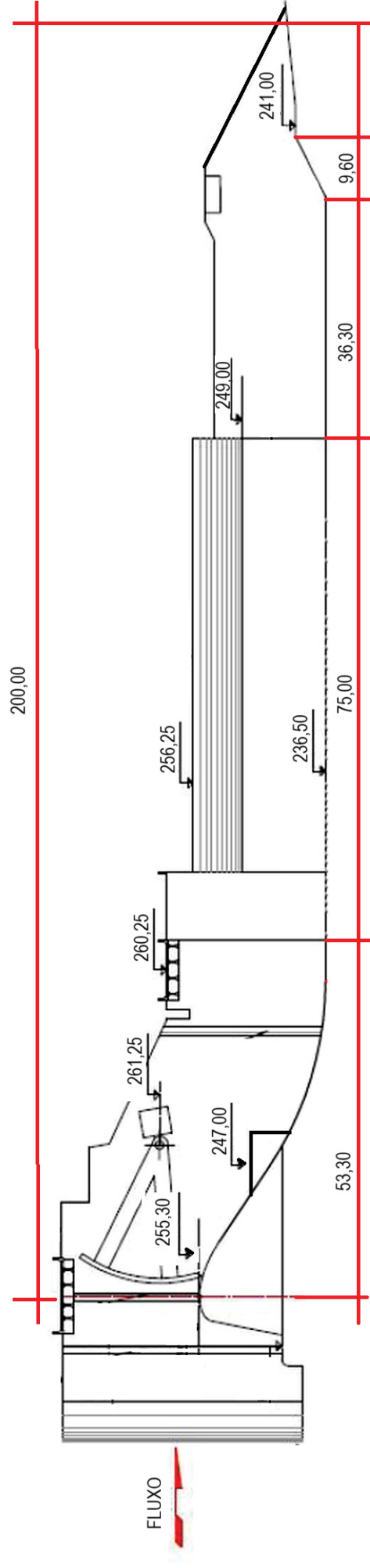
onde $v_{i,z}$ é a variável interpolada na célula i,z ; v_i os valores conhecidos; $w_i(x) = 1/d_i^2$ peso associado ao ponto i , com base na distância (d) do ponto x ao ponto $v_{i,z}$.

Com o preenchimento dos dados de velocidade do escoamento, fração do volume de líquido (relação ar/água) e a pressão em cada ponto da malha, a velocidade da “bolha” é calculada utilizando a equação (5.12) apresentada em Valero et al. (2019) e brevemente descrita na seção 5.4.

O código fonte do programa em linguagem Go, desenvolvido para leitura dos parâmetros hidrodinâmicos simulados no software OpenFOAM, bem como o algoritmo desenvolvido para fazer o preenchimento das variáveis hidrodinâmicas na malha selecionada estão descritos no APÊNDICE I – Leitura de arquivo (Ler_Velocidade).

O código fonte do programa em linguagem Go, desenvolvido para calcular o TDG descrito neste capítulo está apresentado no APÊNDICE II – Modelo de autômato celular.

Figura 70 – Domínio da malha computacional.



Fonte: Autor (2025).

7.2 CÁLCULO DO TDG

Com a escolha da dimensão da malha, a equação de transporte de TDG Equação (7.1) é resolvida com o método das diferenças finitas explícito.

Para resolver o termo convectivo, o termo difusivo e o termo fonte de TDG, segundo, terceiro e quarto termos da Equação (7.1), respectivamente, foram adotados os procedimentos descritos nas Equações (7.3) a (7.8).

Adotou-se como condição inicial:

- a) C (Concentração inicial) é igual a concentração de equilíbrio = 100% ou 0,016915 Kg/m³, em todo o domínio. Valor igual ao adotado no projeto de P&D (Ovelar e Takenobu, 2023);
- b) O diâmetro inicial da “bolha” (D_b) adotado foi igual a 0,8mm, conforme utilizado nos estudos desenvolvidos por Politano et al. (2024) para a UHE Colíder;
- c) A entrada de ar na superfície livre foi considerada como incorporada pelo valor de α_l , obtida na simulação hidrodinâmica (Takenobu, 2022), e adotado como parâmetro de entrada desta tese.

O termo convectivo da Equação (7.1) ($\nabla \cdot (\alpha_l \mathbf{u}_l C)$) foi aproximado utilizando o método de diferenças finitas, conforme apresentado na Equação (7.3):

$$conv(i, z) = \frac{\alpha_{i+1,z} u_{i+1,z} C_{i+1,z} - \alpha_{i-1,z} u_{i-1,z} C_{i-1,z}}{2\Delta_x} + \frac{\alpha_{i,z+1} u_{i,z+1} C_{i,z+1} - \alpha_{i,z-1} u_{i,z-1} C_{i,z-1}}{2\Delta_z}. \quad (7.3)$$

O termo difusivo da Equação (7.1) ($\nabla \cdot \left(\left(v + \frac{v_t}{SCc} \right) \alpha_l \nabla C \right)$) foi aproximado utilizando o método de diferenças finitas, conforme apresentado na Equação (7.4):

$$dif(i, z) = \frac{Visc \alpha_{i+1,z} C_{i+1,z} - 2 Visc \alpha_{i,z} C_{i,z} + Visc \alpha_{i-1,z} C_{i-1,z}}{\Delta_x^2} + \frac{Visc \alpha_{i,z+1} C_{i,z+1} - 2 Visc \alpha_{i,z} C_{i,z} + Visc \alpha_{i,z-1} C_{i,z-1}}{\Delta_z^2}, \quad (7.4)$$

onde $Visc = v + \frac{v_t}{SCc}$; v a viscosidade cinemática = $1,004 \times 10^{-6}$; v_t a viscosidade turbulenta = $1,002 \times 10^{-5}$; SCc o número de Schmidt = 0,7.

Para calcular o termo fonte de TDG da Equação (7.1) (S), foi utilizada a Equação (7.5), sendo os detalhes da equação apresentados no capítulo 4:

$$S = 4\pi NR^2 K_l \left(\frac{p + \frac{\sigma}{R}}{H} - C \right). \quad (7.5)$$

Sendo o coeficiente de transferência de massa devido à turbulência apresentado na Equação (7.6):

$$K_l = 0,4Sc^{-1/2}(v\varepsilon)^{1/4}. \quad (7.6)$$

O valor de N (densidade de número de “bolhas”), definido na Equação (4.41) é recalculado a cada intervalo de tempo de simulação conforme Equação (7.7), sendo o diâmetro da “bolha” (D , definido na Equação (4.42)) também recalculado para cada intervalo de tempo:

$$N_{i,z}^t = N_{i,z}^{t-1} - \Delta_t \left(\frac{(ul_{i+1,z} + ub_{i+1,z})N_{i+1,z}^{t-1} - (ul_{i-1,z} + ub_{i-1,z})N_{i-1,z}^{t-1}}{2\Delta_x} \right), \quad (7.7)$$

sendo $ul_{i,z}$ a velocidade do líquido e $ub_{i,z}$ a velocidade relativa da “bolha” no eixo i e z .

A taxa de dispersão da energia cinética turbulenta (ε (m²/s³)) é resolvida conforme apresentado na Equação (7.8):

$$\varepsilon = v \left(\frac{ul_{i+1,z} + ub_{i+1,z} - ul_{i,z} - ub_{i,z}}{\Delta_z} + \frac{ul_{i,z} + ub_{i,z} - ul_{i,z-1} - ub_{i,z-1}}{\Delta_x} \right)^2. \quad (7.8)$$

Nesta tese a velocidade relativa da “bolha” (ub) foi calculada utilizando a equação apresentada em Valero et al. (2019) e brevemente descrita na seção 5.4, divergindo da velocidade relativa da “bolha” calculada conforme Huang et al. (2019) e descrito no capítulo 4.

Para resolver o 1º termo da Equação (7.1) ($\frac{\partial(\alpha_l C)}{\partial t}$) foi utilizado o método das diferenças finitas, conforme ilustrado na Equação (7.9), sendo a Equação (7.10) resolvida conforme o passo de tempo selecionado:

$$\partial(\alpha_l C)^t = (Equação (7.4) + Equação (7.5) - Equação (7.3))/\alpha_l, \quad (7.9)$$

$$\frac{\partial(\alpha_l C)}{\partial t} = \sum_{t=1}^{t=N} (\alpha_l C)^t \Delta t. \quad (7.10)$$

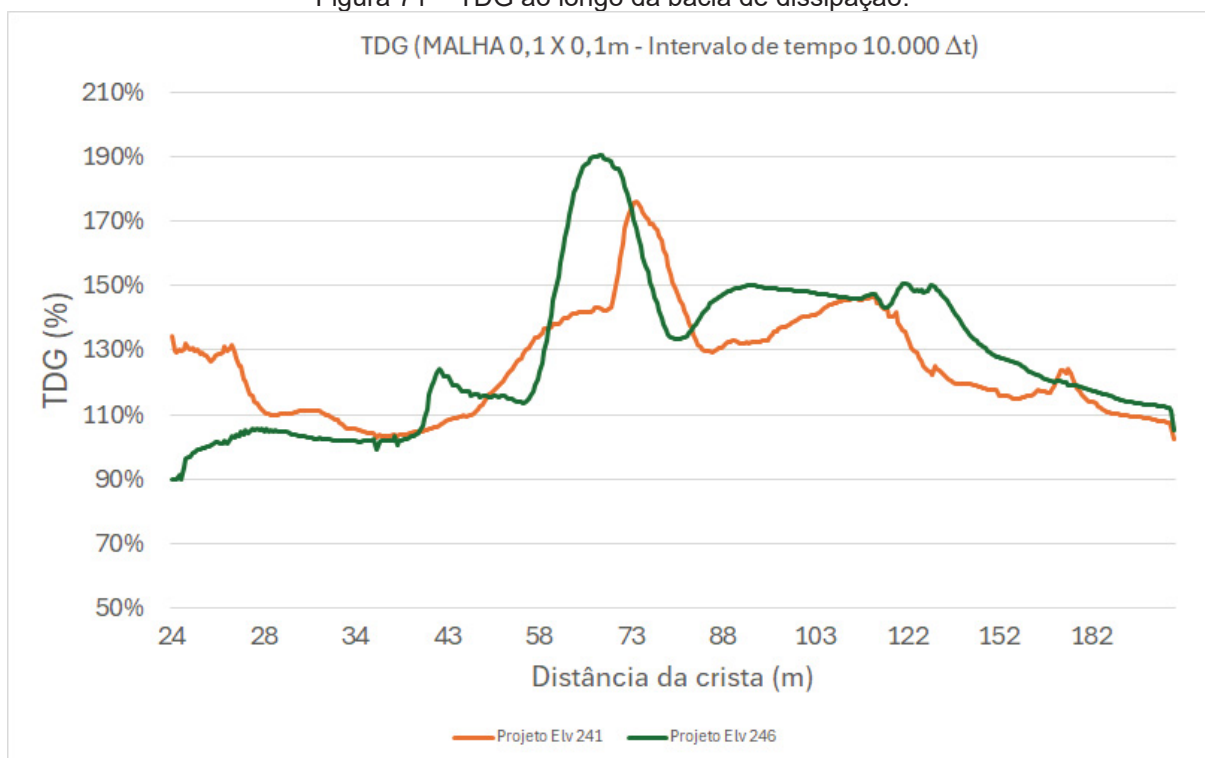
7.3 RESULTADOS

O modelo apresentado nas seções 7.1 e 7.2 foi simulado considerando uma malha de 10 cm x 10 cm, 10.000 intervalos de tempo, com passo de tempo igual a 0,01 s, totalizando um tempo simulado em protótipo igual a 100 segundos.

A Figura 71 apresenta a simulação do TDG ao longo da bacia de dissipação nas elevações 241 m e 246 m, considerando a vazão vertida de 1.600 m³/s (400 m³/s por vão e vazão específica de 33,3 m³/s.m), nível de jusante na elevação 253,22 m, com defletor construído na elevação 247 m. Na elevação 241 m o valor máximo obtido foi de aproximadamente 176% e para a elevação 246 m foi de 190%. A Figura 72 apresenta o TDG simulado ao longo da seção de escoamento na seção igual a 60 m (a 60 m da crista da barragem).

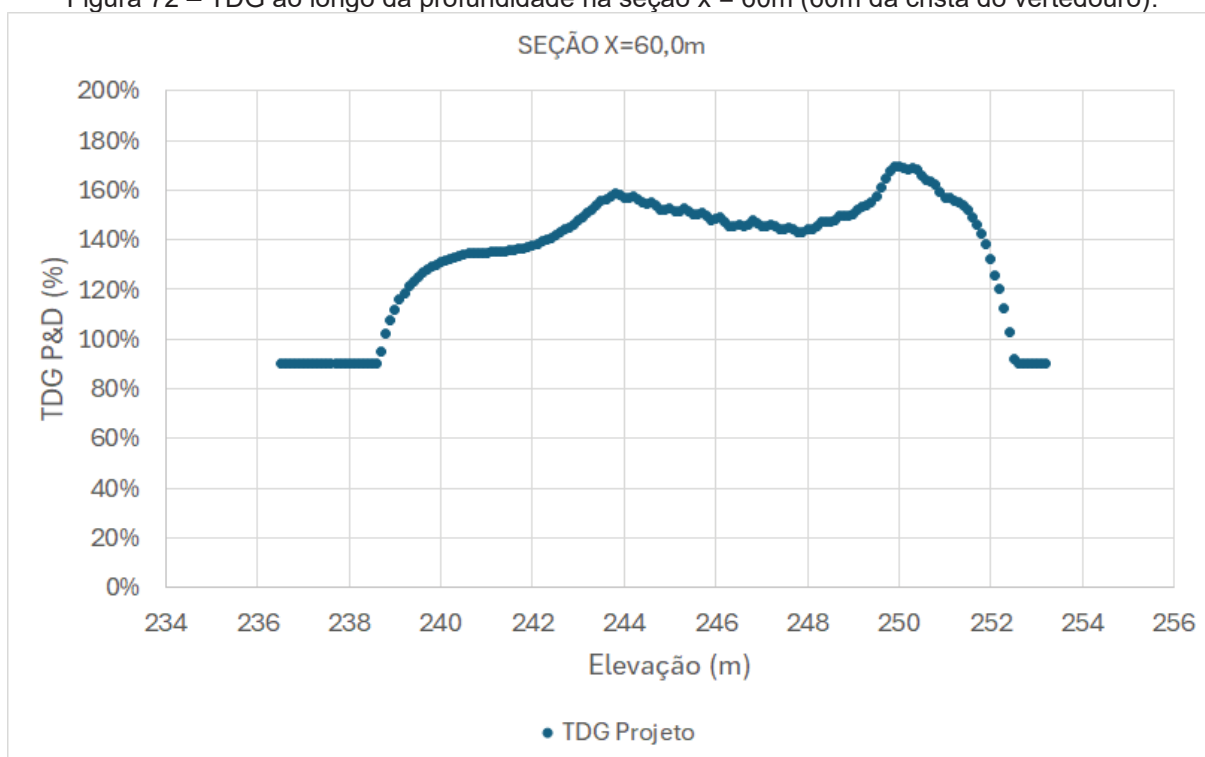
A Figura 73 apresenta a distribuição do TDG ao longo da bacia de dissipação para $\alpha_b = 3\%$, $D_b = 0,8\text{mm}$, destacando a maior concentração entre a elevação 242 m e 250 m, principalmente após a distância de 60 m da crista do vertedouro até o final da região simulada.

Figura 71 – TDG ao longo da bacia de dissipação.



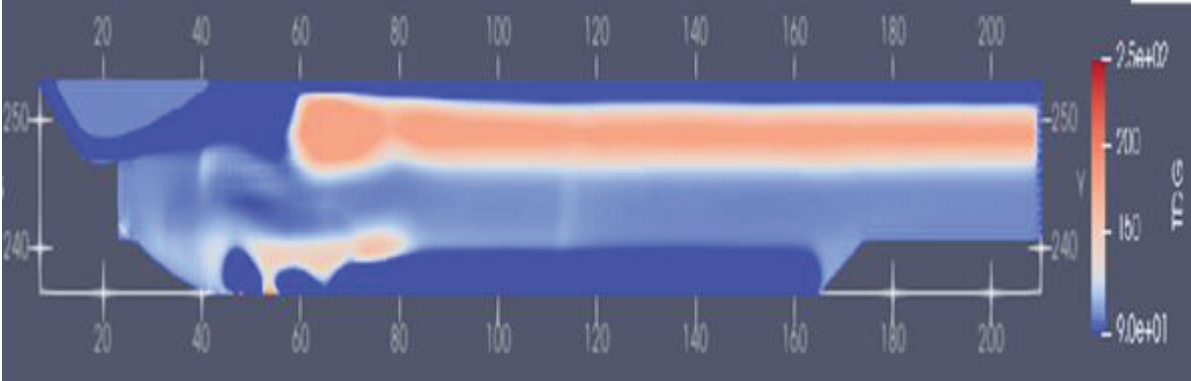
Fonte: Autor (2025).

Figura 72 – TDG ao longo da profundidade na seção x = 60m (60m da crista do vertedouro).



Fonte: Autor (2025).

Figura 73 – TDG ao longo da bacia de dissipação $\alpha_b = 3\%$, $D_b = 0,8\text{mm}$.

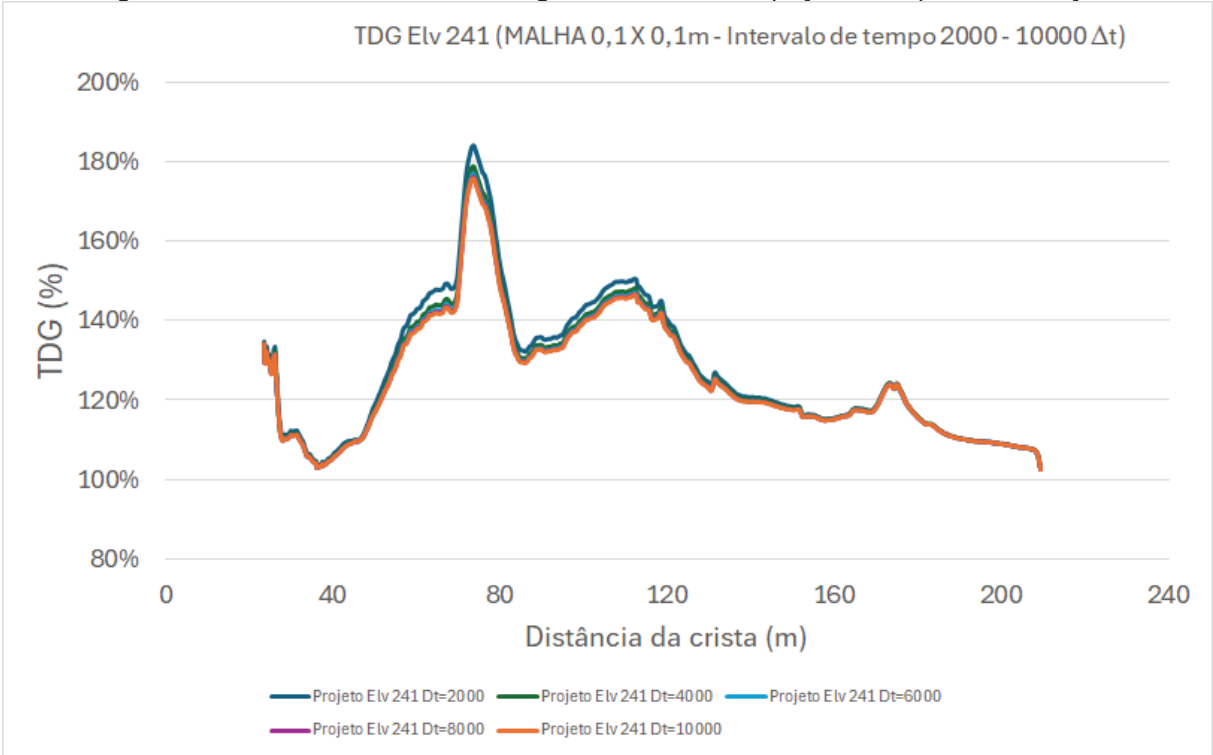


Fonte: Autor (2025).

A Figura 74 e a Figura 75 apresentam a simulação do TDG ao longo da bacia de dissipação, nas elevações 241,00 m e 246,00 m, respectivamente, considerando a vazão vertida de 1.600 m³/s (400 m³/s por vão e vazão específica de 33,3 m³/s.m) e nível de jusante na elevação 253,22 m, com defletor construído na elevação 247 m, para os tempos de simulação de 2.000 Δt , 4.000 Δt , 6.000 Δt , 8.000 Δt e 10.000 Δt .

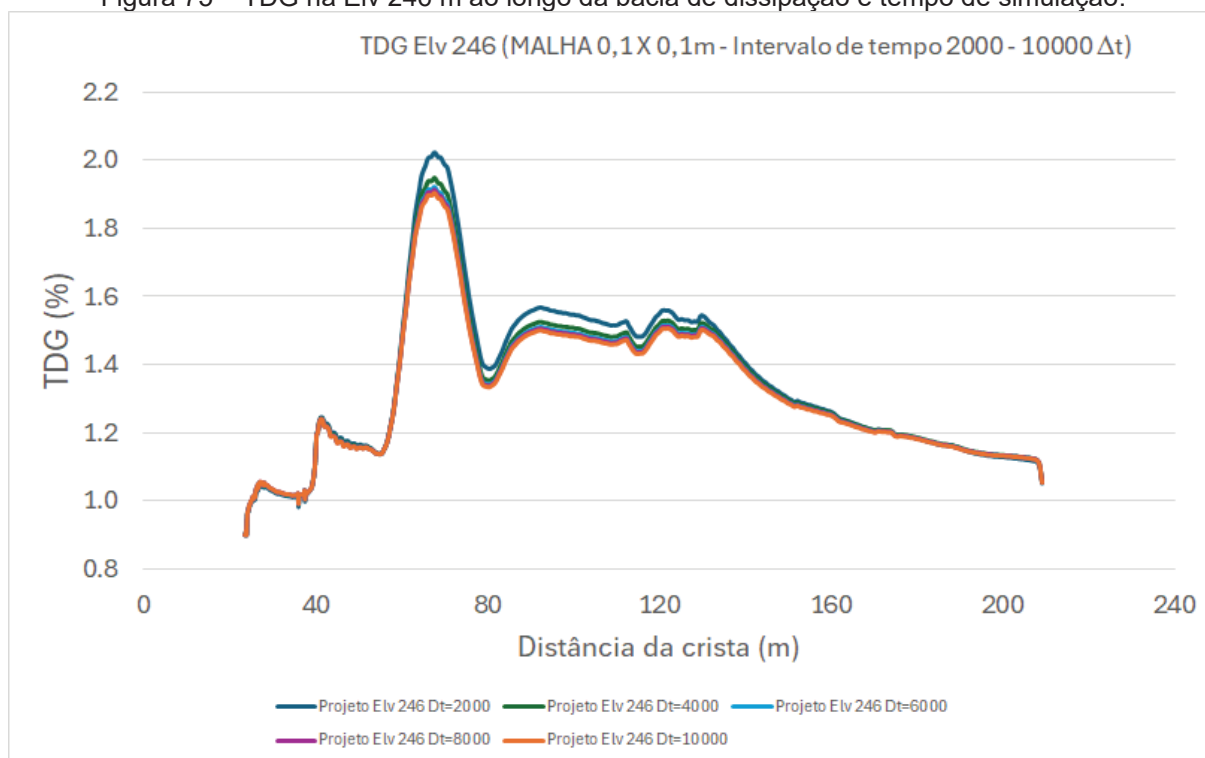
Para a elevação 241 m o valor máximo obtido foi de aproximadamente 170% e para a elevação 246 m foi de 190%.

Figura 74 – TDG na Elv 241 m ao longo da bacia de dissipação e tempo de simulação.



Fonte: Autor (2025).

Figura 75 – TDG na Elv 246 m ao longo da bacia de dissipação e tempo de simulação.



Fonte: Autor (2025).

Os resultados obtidos no modelo matemático desenvolvido nesta tese e apresentados entre a Figura 71 a Figura 75, podem ser comparados com a curva de performance do vertedouro da UHE Colíder, apresentados em Andriolo et al. (2023) e reproduzidos na Figura 10. A sonda de TDG que originou os dados apresentados em Andriolo et al. (2023) está instalada na bacia de dissipação, na elevação 246,00 m e a aproximadamente 54 m da crista do vertedouro. Para a vazão de 1.600 m³/s (33,3 m³/s.m) a curva de performance indica um TDG de aproximadamente 140%. Considerando os dados de TDG e vazão vertida atualizado até maio de 2025, e apresentados na Figura 11, o valor extrapolado de TDG seria de 172% para a vazão vertida de 1.600 m³/s. Cabe destacar que a máxima vazão vertida após a construção do defletor foi de 1.050 m³/s, com TDG medido de 145%.

Comparando o valor do TDG de aproximadamente 150%, para a elevação 246 m, apresentado na Figura 71 e na Figura 72 e simulado pelo modelo matemático, com o valor de TDG de 140% da curva de performance, apresentada na Figura 10, os resultados obtidos nesta tese, indicam que o modelo matemático desenvolvido são aderentes (os dados lidos em campo apresentam variações de aproximadamente 10 pontos percentuais para a mesma vazão) com os valores medidos em campo.

Comparando os valores máximos de 170% e 190% simulados e apresentados na Figura 71, com as estimativas baseadas nas medições de campo apresentadas na Figura 11, pode-se afirmar que os dados são aderentes, visto a variabilidade que se verifica no modelo matemático e nas medições de TDG em campo são próximas a 10 pontos percentuais.

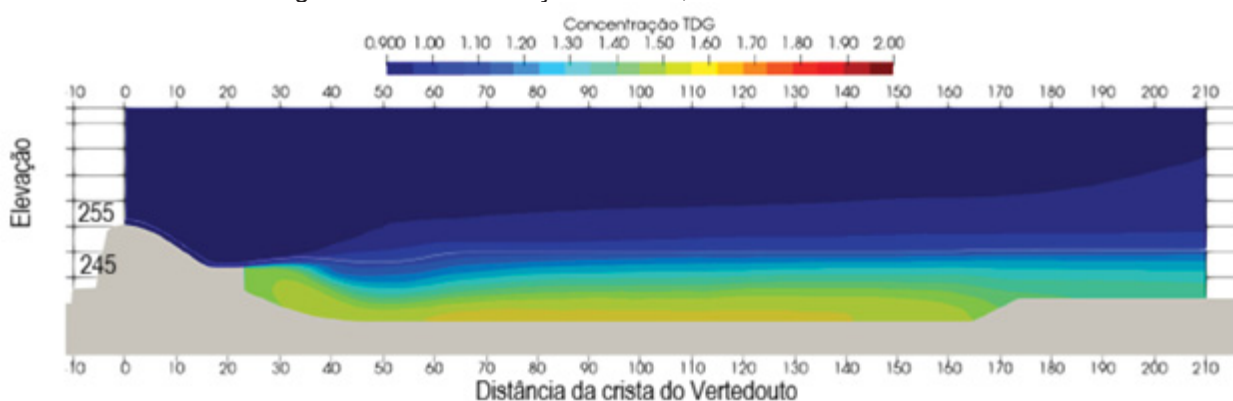
A ascendência do TDG entre a distância de 50 m e 80 m da crista do vertedouro, apresentada nas Figura 71, Figura 74 e Figura 75 se deve ao aumento da relação ar-água nesta região, apresentado na Figura 67 e a equação de transporte de TDG apresentada na Equação (7.1).

O maior percentual de TDG verificado na elevação 246 m é explicado pela maior disponibilidade de ar nesta região, igual a 3,13%, comparado ao máximo de 2,83% na elevação 241 m, conforme é possível verificar na Figura 78.

A Figura 71 apresenta uma desgaseificação (queda do TDG) ao longo da bacia após atingir o valor máximo, sendo que em outros estudos essa queda é menos acentuada. Como exemplo, a Figura 76 reproduzida do estudo de Ovelar (2022), apresenta o valor de TDG ao longo da bacia de dissipação, para uma vazão vertida igual a 800 m³/s, sendo possível verificar certa estabilidade do TDG ao longo da bacia após atingir o seu ápice.

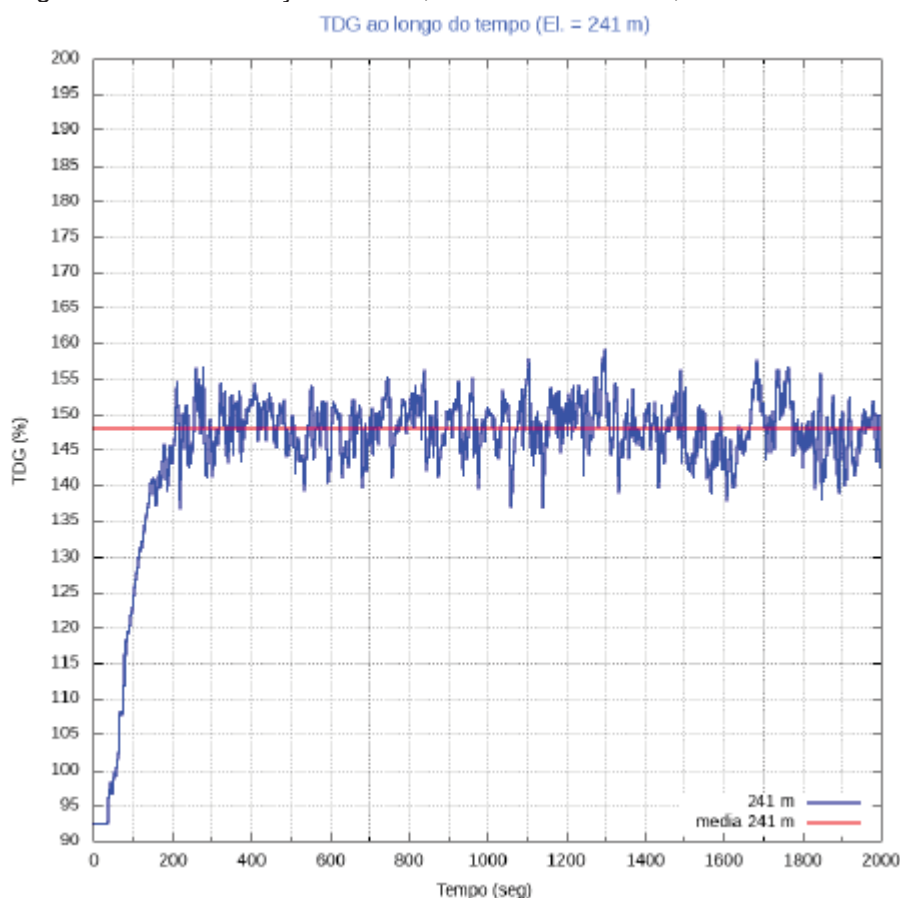
A Figura 77 apresenta o cálculo do TDG na elevação 241 m na seção x = 161 m (a 161 m de distância da crista do vertedouro).

Figura 76 – Concentração de TDG, Qvertida = 800 m³/s.



Fonte: Ovelar (2022).

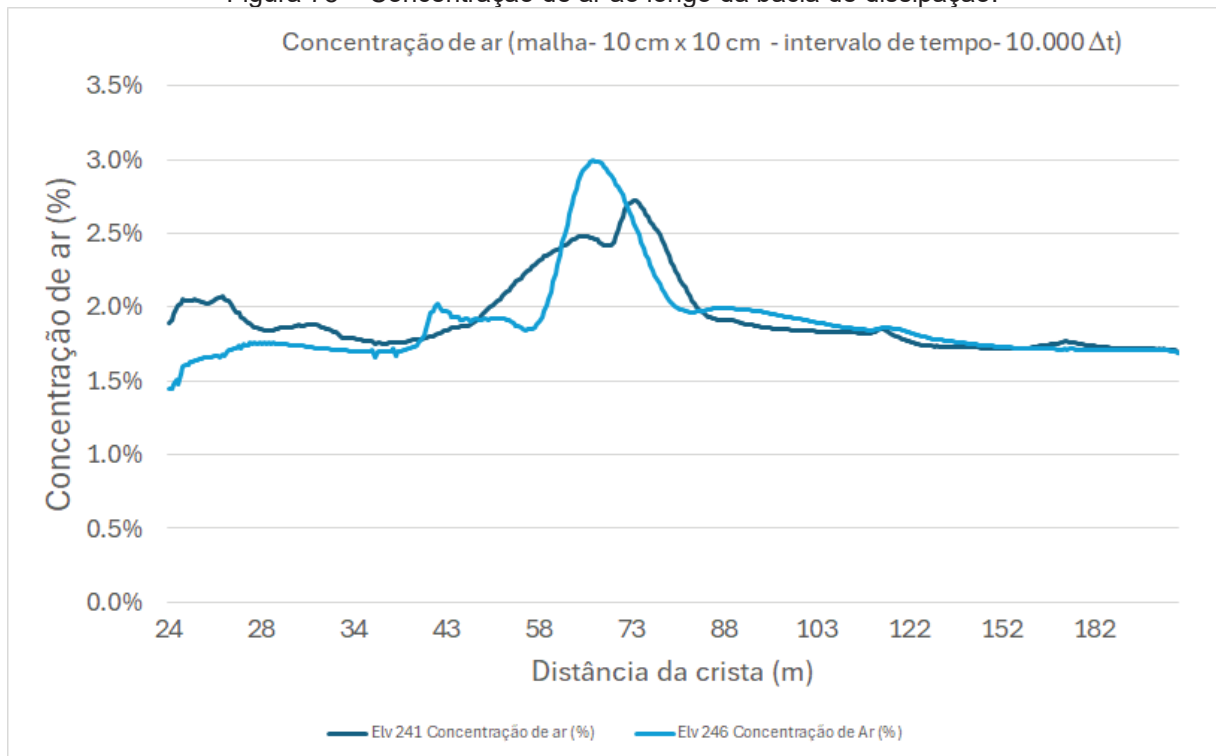
Figura 77 – Concentração de TDG, Qvertida = 800 m³/s, Elv 241 x = 161m.



Fonte: Ovelar (2022).

A diferença na redução do TDG ao longo da bacia de dissipação simulada nesta tese pelo modelo matemático quando comparada ao TDG calculado por Ovelar (2022), é explicada pela forma como o TDG é calculado. Nesta tese, o TDG calculado utiliza o equacionamento apresentado no capítulo 4 e os dados de concentração de ar, ilustrados na Figura 71 como parâmetro de entrada. O TDG calculado por Ovelar (2022) e apresentado na Figura 77, utilizam a concentração de ar calculada através da Equação (4.35). Assim, a maneira que a concentração de ar (“bolhas”) é transportada ao longo da bacia de dissipação não é o mesmo, gerando resultados diferentes.

Figura 78 – Concentração de ar ao longo da bacia de dissipação.



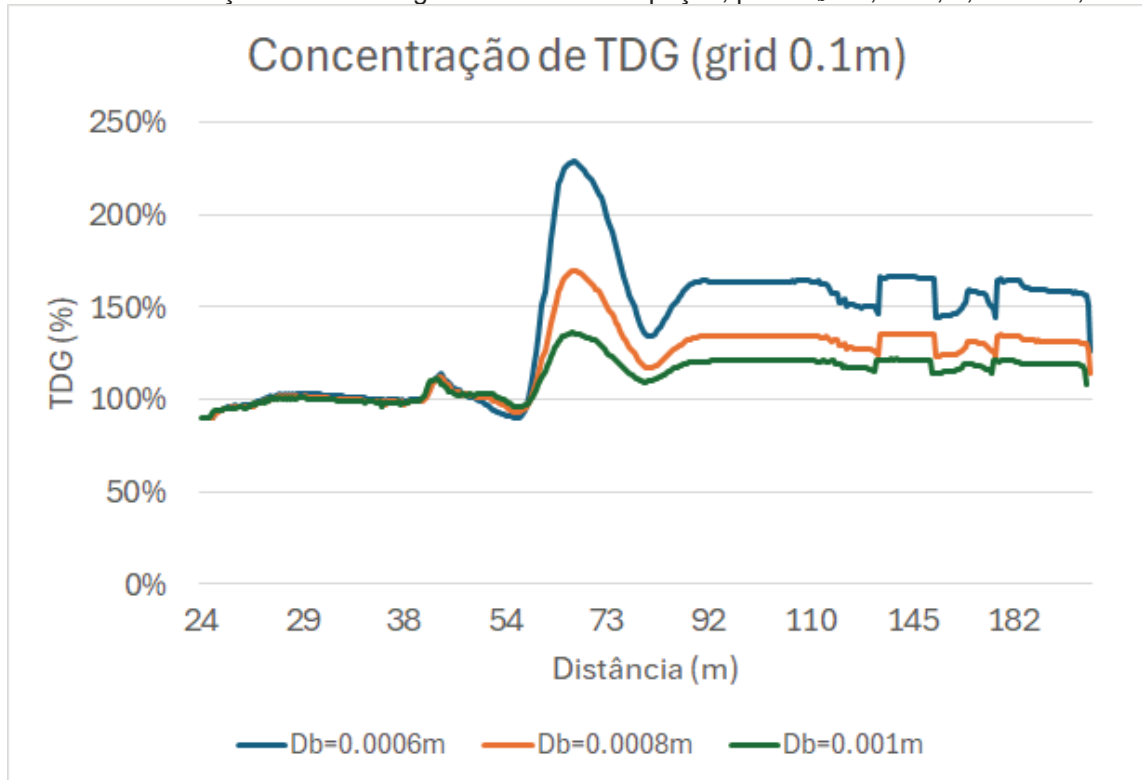
Fonte: Autor (2025).

Uma variável muito importante para estimar o TDG e que impacta diretamente nos resultados é o diâmetro da “bolha”, parâmetro inserido como dado de entrada.

Nas simulações apresentadas entre a Figura 72 e a Figura 75 foi adotado o diâmetro da “bolha” igual a 0,8 mm, em consonância com os estudos desenvolvidos por Politano et al. (2024) para a UHE Colíder. Para verificar a influência do diâmetro da bolha no presente estudo, considerando as equações e metodologia apresentada no capítulo 4 desta tese, simulou-se o TDG para o diâmetro de “bolha” igual a 0,6 mm, 0,8 mm e 1,00 mm, sendo os valores máximos obtidos iguais a 229%, 170% e 136%, respectivamente, sendo os resultados ilustrados na Figura 79.

O impacto da variação do diâmetro a “bolha” no valor simulado do TDG ao longo da bacia de dissipação é similar ao apresentado na Figura 37.

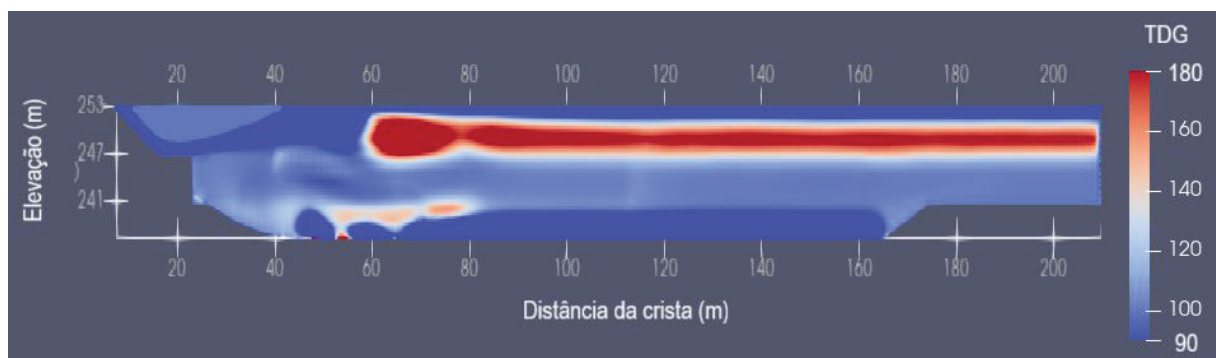
Figura 79 – Concentração de ar ao longo da bacia de dissipação, para $D_b = 0,6\text{mm}$, $0,8\text{mm}$ e $1,0\text{mm}$.



Fonte: Autor (2025).

A Figura 80 apresenta a distribuição do TDG ao longo da bacia de dissipação para $\alpha_b = 3\%$ e $d_b = 0,8\text{mm}$.

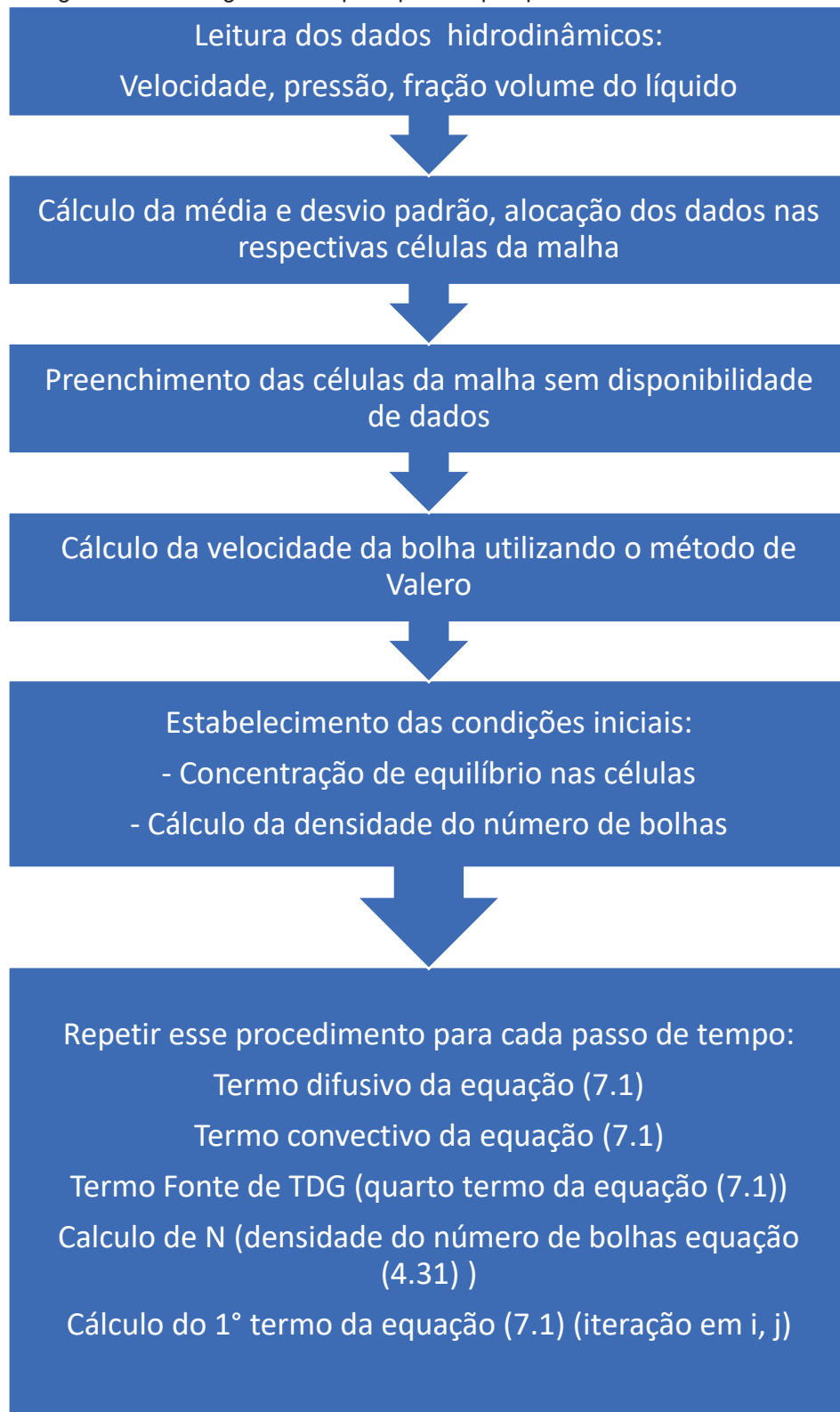
Figura 80 – Distribuição do TDG ao longo da bacia $\alpha_b = 3\%$ $d_b = 0,8\text{mm}$.



Fonte: Autor (2025).

A Figura 81 apresenta o fluxograma das etapas para o cálculo do TDG desenvolvidos nesta tese, cuja equações estão apresentadas no capítulo 4.

Figura 81 – Fluxograma das principais etapas para o cálculo do TDG.



Fonte: Autor (2025).

8 IMPLEMENTAÇÃO DO AUTÔMATO CELULAR

O objetivo deste capítulo é apresentar o modelo computacional desenvolvido com autômatos celulares em linguagem Go, para calcular o Total de Gás Dissolvido (TDG), cuja teoria é descrita no capítulo 5.

Nesta tese foi adotada uma malha bidimensional, composta por células discretizadas no eixo z , representando a profundidade da bacia, e no eixo x correspondente ao comprimento da bacia de dissipação. O eixo y foi desconsiderado, uma vez que a simulação hidrodinâmica utilizada considerou um vertedouro confinado lateralmente e uma bacia de dissipação com restrição nas bordas. Nesta condição, as componentes de velocidade lateral (eixo y) podem ser consideradas desprezíveis devido à sua baixa variabilidade ao longo da largura do escoamento.

Para simular o valor de TDG, foi utilizado como dado de entrada a velocidade do líquido (u_l), variável simulada no modelo hidrodinâmico do projeto de P&D (Takenobu, 2022), intitulado “Metodologia para Modelagem Computacional de TDG na Água em Fluxos Efluentes de Vertedouros”, que abrangem a simulação do fluxo no vertedouro com as configurações hidráulicas apresentadas na seção 6.1.

8.1 MALHA COMPUTACIONAL E PREENCHIMENTO DE FALHAS

O modelo hidrodinâmico desenvolvido no projeto de P&D (Takenobu, 2022), cujo dado de velocidade do líquido foi utilizado como parâmetro de entrada do modelo de autômato celular, utilizou a malha apresentada na Figura 68 e foi descrita na seção 6.4. Desta forma, a disponibilidade de dados varia ao longo da malha com espaçamentos de 9,4 cm até 1,5 m.

Nesta tese o domínio computacional abrange o vertedouro desde a crista, a bacia de dissipação até o término do *end sill*, totalizando 174,20 m em escala de protótipo. Adicionalmente, foi incluído um trecho de 25,80 m correspondente ao canal de restituição, resultando em um comprimento total de 200,00 m a partir da crista do vertedouro. A Figura 70 apresenta o domínio computacional adotado, o qual é composto por aproximadamente 230.000 células, formado por uma malha retangular com resolução espacial de 10 cm nos eixos x e z . O nível de água adotado foi na elevação 253,20, correspondente a uma vazão defluente total de 3.200 m³/s.

Para compatibilizar os dados disponíveis com a malha utilizada para simular o TDG desta tese, foi necessário estimar os dados conforme a dimensão de malha escolhida para fazer as simulações, sendo o procedimento adotado descrito na seção 7.1.

Com o preenchimento dos dados de velocidade do escoamento, a velocidade da “bolha” é calculada utilizando a equação (5.12) apresentada em Valero et al. (2019) e brevemente descrita na seção 5.4.

O código fonte do programa em linguagem Go, desenvolvido para leitura dos parâmetros hidrodinâmicos simulados no software OpenFOAM, bem como o algoritmo desenvolvido para fazer o preenchimento das variáveis hidrodinâmicas na malha selecionada estão descritos no APÊNDICE I – Leitura de arquivo (Ler_Velocidade).

O código fonte do programa em linguagem Go, desenvolvido para calcular o TDG descrito neste capítulo está apresentado no APÊNDICE II – Modelo de autômato celular.

8.2 PARÂMETROS UTILIZADOS

Para o cálculo do TDG utilizando autômatos celulares, foram inicialmente definidos como parâmetros as dimensões da célula, o passo de tempo e o tempo total de simulação.

Foi adotada uma malha com células de 10 cm nos eixos x e z , totalizando aproximadamente 230.000 células.

O passo de tempo (Δ_t) adotado foi de 0,0001 segundos.

O número de passos de tempo simulado (N_t) foi de 1.000.000 e 5.000.000, equivalente a 100 segundos e 500 segundos de tempo em protótipo, respectivamente.

Considerou-se que no início da simulação ($t = 0s$) todas as células da malha apresentavam concentração de oxigênio em equilíbrio, ou seja, o TDG inicial em cada célula correspondia a uma saturação de 100%, considerando concentração total do gás igual a $0,021 \text{ kg m}^{-3}$, para $T = 27,8^\circ\text{C}$, $P = 97,2 \text{ kPa}$, sendo $N_2 \approx 0,0122 \text{ kg m}^{-3}$, $O_2 \approx 0,0078 \text{ kg m}^{-3}$, $Ar \approx 0,00052 \text{ kg m}^{-3}$, $CO_2 \approx 0,00052 \text{ kg m}^{-3}$ (Colt, 2012).

8.3 CÁLCULO DO TDG UTILIZANDO AUTÔMATOS CELULARES

Este capítulo descreve como os autômatos celulares foram estruturados para calcular o total de gases dissolvidos, cuja teoria é apresentada no capítulo 5.

Para calcular a concentração de equilíbrio do O_2 calcula-se a pressão total na célula pela equação:

$$P_{total} = (P_{atm} + \rho_{H_2O}h - P_{vapor})/P_{atm}, \quad (8.1)$$

onde P_{atm} é a pressão atmosférica = 98,130 KPa; ρ_{H_2O} a densidade da água = 9.810 (N/m³); h (m) a profundidade; P_{vapor} a pressão de vapor = 3,7376 KPa, conforme Colt (2012).

A concentração molar de O_2 é dada pela equação:

$$C_{O_2} = 1000M_{O_2}K_H X_{O_2} P_{total}, \quad (8.2)$$

em que M_{O_2} é a massa molar do O_2 = 32 g/mol; K_H a constante de Henry = 10^{-3} mol/(L.atm); X_{O_2} a fração molar de oxigênio no ar (adimensional).

Foi considerada como área de entrada (injeção) de O_2 a região do defletor, compreendendo a sua crista na elevação 247,00 m até o nível de jusante do escoamento (253,20 m) e seu comprimento total, equivalente a 7 m acrescido de 3 m, totalizando um comprimento de admissão de ar igual a 10 m.

A entrada (admissão) de oxigênio no escoamento considera a concentração de equilíbrio conforme a profundidade da célula considerada. A concentração em mg/l é dada pela Equação (8.2).

Para cada passo de tempo o volume da célula é atualizado considerando os volumes de entrada e saída advindos das células adjacentes, considerando as velocidades do fluxo no eixo x e eixo z.

A concentração de oxigênio na célula é atualizada a cada passo de tempo, considerando a concentração existente na célula (i,z) e a concentração das células adjacentes. Nessa etapa, a equação da continuidade é satisfeita, uma vez que o fluido é considerado incompressível, sendo a fração molar de O_2 na célula (i,z) dada pela equação:

$$F_{O_2}(i,z)_t = \frac{F_{O_2}(i,z)_{t-1}}{V_{célula}} (V_{célula} - V_{entrada}) + \frac{C_{O_2}\alpha_{bolha}}{V_{célula}} V_{entrada} + \frac{F_{O_2}(i,z)_{t-1}(1-\alpha_{bolha})}{V_{célula}} V_{entrada}, \quad (8.3)$$

sendo $F_{O_2}(i, z)_t$ a fração molar de O_2 na célula (i, z) no instante t ; $V_{célula}$ (m^3/s) o volume de água na célula; $V_{entrada}$ (m^3/s) o volume de água entrando na célula advindo das células adjacentes; α_{bolha} a relação de “bolha” no início da mistura; C_{O_2} a concentração molar de O_2 considerando a concentração de equilíbrio conforme a profundidade.

Em seguida é calculada a difusão considerando o escoamento bidimensional:

$$dzdx = \frac{F_{O_2}(i+1, z)_{t-1} - 2F_{O_2}(i, z)_{t-1} + F_{O_2}(i-1, z)_{t-1}}{dx dz}, \quad (8.4)$$

$$dxdz = \frac{F_{O_2}(i, z+1)_{t-1} - 2F_{O_2}(i, z)_{t-1} + F_{O_2}(i, z-1)_{t-1}}{dz dx}, \quad (8.5)$$

onde Δ_x (m) é a dimensão da célula no eixo x (comprimento da bacia); Δ_z a dimensão da célula no eixo z (profundidade).

A advecção é calculada utilizando as equações abaixo:

$$adv_x = \frac{F_{O_2}(i, z)_{t-1}(V_{xbolha}(i, z) + V_x(i, z)) - F_{O_2}(i-1, z)_{t-1}(V_{xbolha}(i-1, z) + V_x(i-1, z))}{\Delta_x^2}, \quad (8.6)$$

$$adv_z = \frac{F_{O_2}(i, z)_{t-1}(V_{zbolha}(i, z) + V_z(i, z)) - F_{O_2}(i, z-1)_{t-1}(V_{zbolha}(i, z-1) + V_z(i, z-1))}{\Delta_z^2}, \quad (8.7)$$

sendo V_{zbolha} (m/s) a velocidade da “bolha” no eixo z; V_z a velocidade do escoamento no eixo z.

A perda de concentração de O_2 na superfície livre é dada pela equação:

$$C(t) = \frac{\Delta_t K_L}{86400} (C_{eq} - F_{O_2}(i, z)_{t-1}), \quad (8.8)$$

em que Δ_t é o passo de tempo (s); K_L o coeficiente de transferência de massa no líquido (m/s) = 2,0; C_{eq} a concentração de equilíbrio de O_2 na superfície livre.

Assim, a concentração de O_2 no instante t é dado pela equação:

$$F_{O_2}(i, z)_t = F_{O_2}(i, z)_{t-1} + \Delta_t (-adv_x - adv_z + DO_2H_2O(dzdx + dxdz)), \quad (8.9)$$

onde DO_2H_2O (m^2/s) é a difusividade molecular do soluto no líquido = $2,4 \cdot 10^{-9}$ (Cengel e Cimbala, 2017).

Caso a célula a ser calculada esteja adjacente à superfície livre, a Equação (8.9) deve incluir as perdas de concentração de O_2 na interface, conforme descrito pela Equação (8.8). Assim, para as células localizadas na fronteira da superfície livre, o cálculo de $F_{O_2}(i, z)_t$ é realizado de acordo com a Equação (8.10):

$$F_{O_2}(i, z)_t = F_{O_2}(i, z)_{t-1} + \Delta_t(-adv_x - adv_z + DO_2H_2O(dzdx + dxdz)) + C(t). \quad (8.10)$$

As Equações (8.1) a (8.10) são solucionadas em todas as células da malha a cada passo de tempo (t).

Para o cálculo da fração molar de O_2 no passo de tempo seguinte ($t+1$), determina-se a vazão de entrada em cada célula (i, z), proveniente das células adjacentes, considerando o vetor velocidade associado a cada célula.

Em seguida calcula-se a variação da concentração de O_2 decorrente das diferenças de pressão (associadas à profundidade), e do volume de água que entra na célula (i, z) proveniente das células ($i, z-1$) ou ($i, z+1$), as quais possuem profundidades distintas, e, conseqüentemente, diferentes concentrações de equilíbrio de O_2 (em mg/l):

$$\Delta mol_{O_2} = K_H(P_{O_2(i, z+1)} - P_{O_2(i, z)}), \quad (8.11)$$

sendo K_H a constante de Henry = 10^{-3} (mol/L.atm); $P_{O_2(i, z+1)}$ a pressão parcial de O_2 na célula ($i, z+1$), dada pela equação:

$$P_{O_2(i, z+1)} = (P_{atm} + \rho_{H_2O}H + \rho_{H_2O}\Delta_z - P_{vapor})/P_{atm}. \quad (8.12)$$

Para calcular a variação do diâmetro da “bolha” em função da pressão que ela está submetida, foi utilizado um modelo simples, que desconsidera a tensão superficial e a pressão do vapor, sendo o diâmetro da “bolha” dado pela equação:

$$d_{b1} = d_b \left(\frac{P_1}{P_2} \right)^{\frac{1}{3n}}, \quad (8.13)$$

onde d_{b1} é o diâmetro da “bolha” na profundidade da célula (i, z); d_b o diâmetro da “bolha” no início da mistura; $P_1 = P_{atm}$ a pressão atmosférica; $P_2 = P_{atm} + \rho_{H_2O} H$ a pressão no líquido ao redor da “bolha” conforme a profundidade da célula (i, z); $n = 1,4$ o coeficiente para escoamento adiabático.

Na sequência a transferência de massa de O_2 é calculada pela equação:

$$N_{mol} = S_h \frac{DO_2 H_2 O}{d_{b1}} \pi d_{b1}^2 \frac{\Delta z}{V_z} (F_{O_2}(i, z+1)_t - 0,5(K_H(P_{O_2(i,z+1)} + P_{O_2(i,z)}))), \quad (8.14)$$

sendo S_h o número de Sherwood; d_b o diâmetro da “bolha” conforme a profundidade da mistura (m); $DO_2 H_2 O$ (m^2/s) a difusividade molecular do soluto no líquido = $2,4 \cdot 10^{-9}$ (Cengel e Cimbala, 2017):

$$S_h = 2 + 0,6 R_e^{1/2} S_c^{1/3}, \quad (8.15)$$

sendo o número de Reynolds e o número de Schmidt dado pelas equações (8.16) e (8.17):

$$R_e = V_z d_{b1} / \nu_c, \quad (8.16)$$

$$S_c = \frac{\nu_c}{DO_2 H_2 O}, \quad (8.17)$$

onde ν_c é a viscosidade cinemática = $1,004 \cdot 10^{-6}$ (m^2/s).

Calculada a transferência de massa, a concentração de O_2 para o passo de tempo seguinte é calculada utilizando a Equação (8.18):

$$F_{O_2}(i, z)_t = \frac{F_{O_2}(i, z)_{t-1}}{V_{célula}} (V_{célula} - V_{entrada_z}) + \frac{(F_{O_2}(i, z+1)_{t-1} + N_{mol})}{V_{célula}} V_{entrada_z}, \quad (8.18)$$

em que $V_{entrada_z}$ é o volume de entrada na célula advindo da célula ($i, z-1$) ou ($i, z+1$).

Após o cálculo da concentração de O_2 para o passo de tempo seguinte, a razão de “bolha” no início da mistura (α_{bolha}) é atualizada considerando o processo

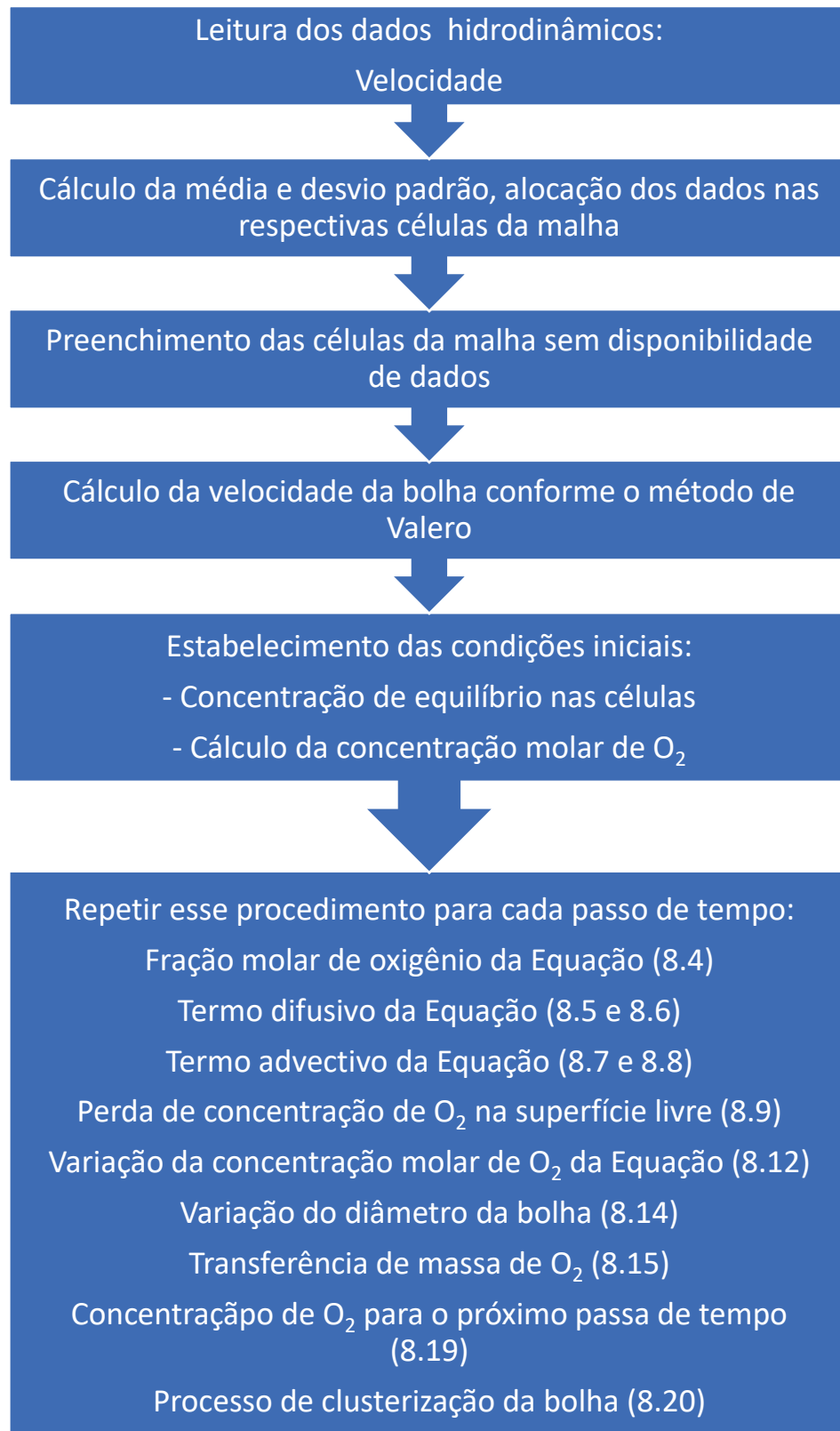
de clusterização dos vazios da mistura, conforme descrito na seção 4.2 utilizando a seguinte equação:

$$(\alpha_{bolha})_t = (\alpha_{bolha})_{t-1} + \left[\frac{(\alpha_{bolha})_{t-1}}{1 - (\alpha_{bolha})_{t-1}} + \frac{((\alpha_{bolha})_{t-1}\sigma)^2}{(1 - (\alpha_{bolha})_{t-1})^3} - (\alpha_{bolha})_{t-1} \right] / N_t, \quad (8.19)$$

sendo $(\alpha_{bolha})_t$ a relação de “bolha” no próximo passo de tempo; $(\alpha_{bolha})_{t-1}$ a relação de “bolha” no passo de tempo que calculou a transferência de massa do próximo passo de tempo $F_{O_2}(i, z)_t$; α_{bolha} a relação de “bolha” no início da mistura; σ um número aleatório uniforme (variando de -1 a +1, com média 0); $(\alpha_{bolha}\sigma)^2$ representa o desvio padrão (variância) apresentado na Equação (4.17).

A Figura 82 apresenta o fluxograma das etapas de cálculo do TDG utilizando o modelo de autômato celular, descrito na seção 8.3, conforme as equações (8.1) a (8.20).

Figura 82 – Fluxograma das principais etapas para o cálculo do TDG utilizando autômatos celulares.



Fonte: Autor (2025).

8.4 RESULTADOS PARA $N_t = 1.000.000$

Nesta seção são apresentados os resultados da simulação do TDG utilizando o modelo de autômatos celulares, considerando o número de passos de tempo igual a 1.000.000, equivalente a 100 s de simulação. Para cada um dos 6 casos analisados, os parâmetros de diâmetro da “bolha” e relação ar-água (α_{bolha}) foram variados. O tempo de processamento foi de aproximadamente 24 horas, utilizando um computador pessoal (Intel i7-10510U 1,8GHz).

O modelo de cálculo de TDG utilizando autômatos celulares descrito na seção 8.3, considerou como região de injeção de oxigênio o comprimento de 10 m, entre as células localizadas na distância de 17,7 m a 27,7 m da crista do vertedouro, e entre as profundidades 247 m a 253,2 m (crista do defletor e nível de água a jusante). O defletor está localizado entre as células 17,7 m e 24,7 m, na elevação 247 m.

O TDG foi simulado variando os seguintes parâmetros: a) diâmetro inicial da “bolha” considerando (d_b) igual a 0,6 mm, 0,8 mm e 1,0 mm e b) relação ar-água (α_{bolha}), igual a 3% e 5%. A simulação considerou vazão vertida de 1.600 m³/s (400 m³/s por vão e vazão específica de 33,3 m³/s.m), nível de jusante na elevação 253,22 m, que equivale a uma vazão defluente total de 3.200 m³/s (1.600 m³/s de vazão vertida e 1.600 m³/s de vazão turbinada), que caracteriza um escoamento a jusante do tipo *skimming flow*.

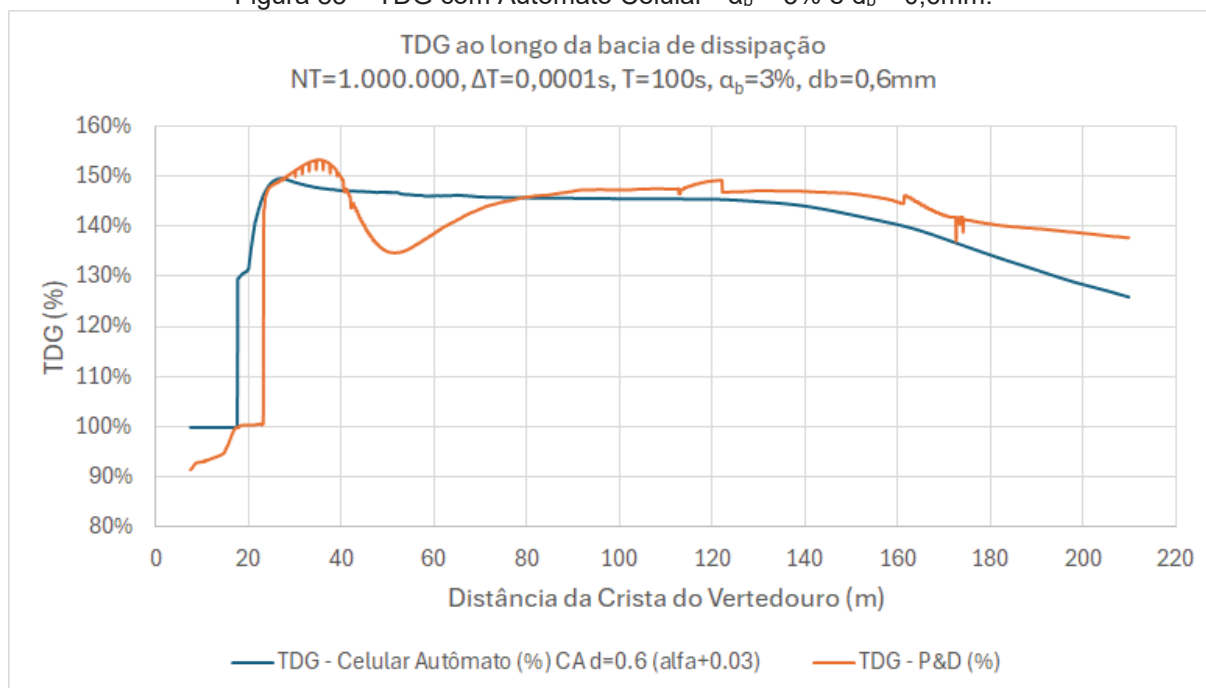
O Quadro 2 apresenta as seguintes colunas: 1) número do ensaio; 2) relação ar-água (α_{bolha}); 3) Diâmetro da “Bolha” (mm); 4) “TDG Máximo”, que representa o maior valor de TDG obtido em uma célula ao longo de toda a simulação; 5) “TDG Média Máxima”, que corresponde à média dos valores máximos de TDG em cada célula ao longo do eixo x e, 6) “TDG Médio em toda a Bacia”, que considera o valor médio em todas as células da malha, aproximadamente 230.000 células.

Quadro 2 – Simulações de TDG.

Nº Ensaio	α_{bolha} (%)	Diâmetro da “Bolha” (mm)	TDG Máximo (%)	TDG Média Máxima (%)	TDG Médio em toda a Bacia (%)
1	3	0,60	150	140	117
2	3	0,80	150	140	117
3	3	1,00	150	140	117
4	5	0,60	153	143	119
5	5	0,80	153	143	119
6	5	1,00	153	143	119

Os resultados obtidos pelo modelo de autômatos celulares foram comparados com os resultados do projeto de P&D apresentados por Ovelar (2022). A Figura 83 apresenta este comparativo, considerando os parâmetros do ensaio n° 1, apresentado no Quadro 2, com $\alpha_b = 3\%$ e $d_b = 0,6$ mm. Nesta figura o valor máximo do TDG simulado pelo modelo de autômato celular foi de 150%, com ápice na distância de 27,1 m da crista do vertedouro. O valor máximo de TDG simulado por Ovelar (2022) foi de 153% a distância de 35,3 m da crista do vertedouro.

Figura 83 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6$ mm.



Fonte: Autor (2025).

Os resultados ilustrados na Figura 83 podem ser comparados à curva de performance do vertedouro da UHE Colíder, apresentados por Andriolo et al. (2023) e reproduzidos na Figura 10. A sonda de TDG responsável pelos dados foi instalada na bacia de dissipação, na elevação 246,00 m, a aproximadamente 54 m da crista do vertedouro. Para a vazão de 1.600 m³/s (33,3 m³/s.m), a curva de performance indica um TDG de aproximadamente 140%. Considerando os dados de TDG e vazão vertida atualizado até maio de 2025, e apresentados na Figura 11, o valor extrapolado de TDG seria de aproximadamente 172% para a mesma vazão vertida de 1.600 m³/s. É importante destacar que, como a maior vazão vertida registrada após a construção do defletor foi de 1.050 m³/s, com TDG medido de 145%, não é possível afirmar que o TDG atingiria 172%, uma vez que a dinâmica de dissolução de gases pode variar

conforme maior seja o “colchão” de água na bacia de dissipação e as condições de vazão turbinada.

O maior valor de TDG obtido pela da simulação utilizando CA, igual a 153%, indica que o modelo desenvolvido tem aderência com os valores verificados em campo e com os valores simulados por Ovelar (2022).

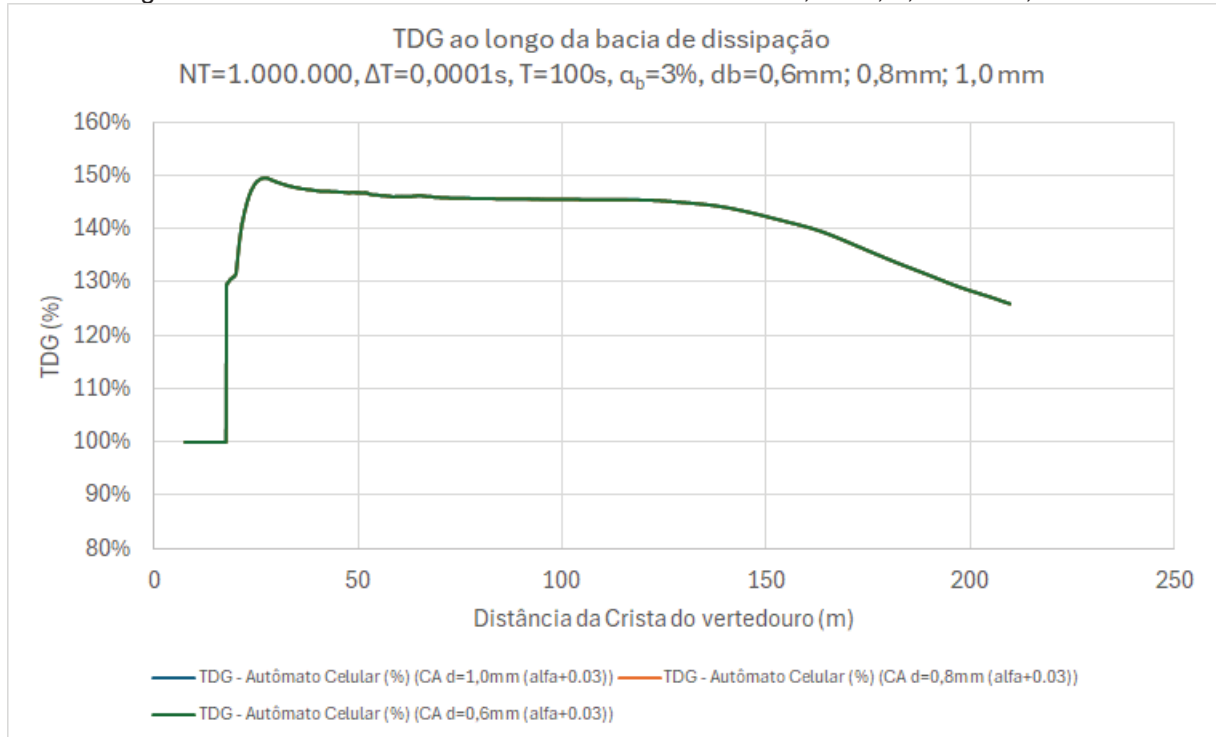
O maior valor do TDG, igual a 153%, foi verificado a 23,8 m da crista do vertedouro, praticamente ao final do defletor localizado a 24,7 m da crista. O decaimento do valor do TDG ao longo da bacia, é compatível com o decaimento apresentado no estudo de Politano et al. (2022), que está reproduzido na Figura 33. Em relação ao decaimento do TDG ao longo da bacia de dissipação os resultados obtidos com autômatos celulares, são mais consistentes que os resultados obtidos com o modelo matemático descrito no capítulo 4.

Comparando os resultados de TDG do modelo de autômato celular proposto nesta tese com os resultados apresentados por Ovelar (2022), o aumento do TDG na região de entrada das “bolhas” apresenta um padrão muito semelhante, com valores máximos quase coincidentes. Aproximadamente a 50 m a jusante da crista do vertedouro, o modelo de Ovelar mostra uma rápida redução do TDG seguida por um subsequente incremento, que pode ser atribuído a uma maior concentração de “bolhas” de ar arrastadas ainda não dissolvidos (ver Figura 78). Como o modelo de autômato celular não incluiu essa informação como dado de entrada (relação ar-água simulado no OpenFOAM), esse comportamento localizado não foi reproduzido. No entanto, a cerca de 70 m da crista, os valores de TDG obtidos por ambas as abordagens são muito próximos, demonstrando a robustez do modelo de autômato celular. A 130 m da crista do vertedouro a redução do TDG simulada pelo modelo de autômato celular torna-se mais pronunciada do que a observada nos resultados de Ovelar. Em contraste, a simulação realizada por Politano *et al.* (Figura 33) apresenta uma redução mais lenta do TDG ao longo da bacia de dissipação, embora a tendência geral e os valores máximos de TDG permaneçam próximos (diferença inferior a 10 pontos percentuais) entre os dois modelos.

A Figura 84 apresenta o cálculo de TDG com autômato celular ao longo da bacia de dissipação, considerando relação ar-água (α_{bolha}) igual a 3% e com diâmetro da “bolha” de 0,6 mm, 0,8 mm e 1,0 mm. As simulações do modelo de AC mostram que o diâmetro da “bolha” não é um parâmetro sensível para o resultado dos valores máximos de TDG ao longo da bacia, mesmo o diâmetro da “bolha” sendo um

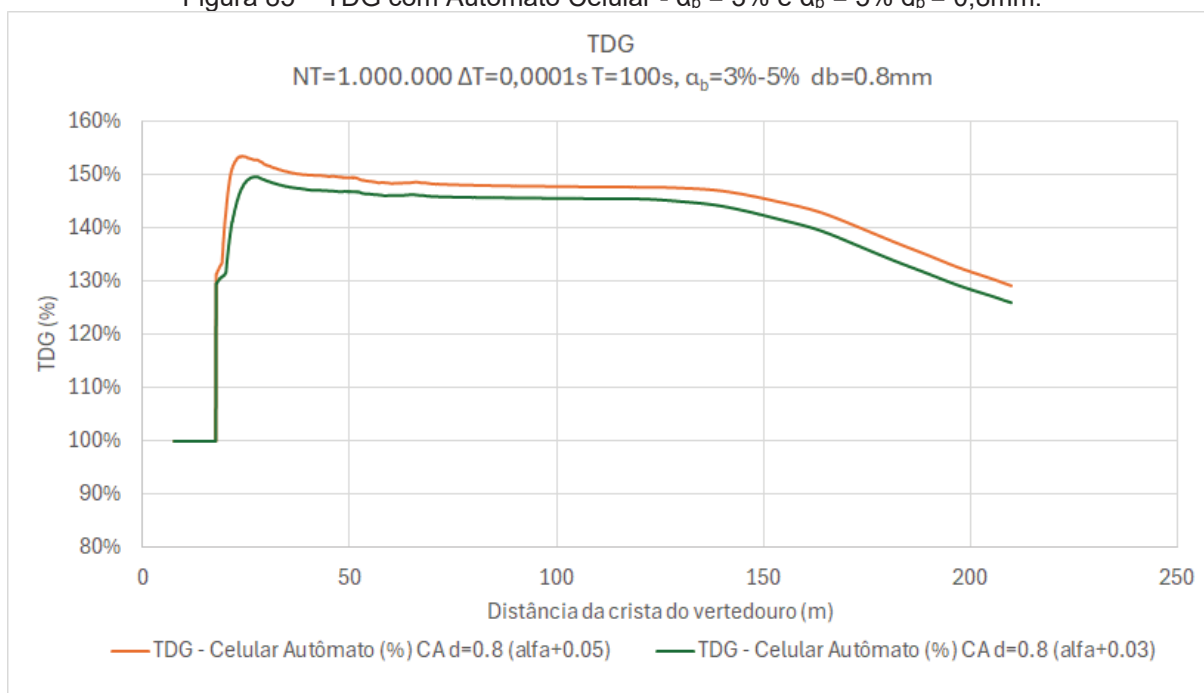
parâmetro de entrada das equações (8.13), (8.14) e (8.16). Isso indica que este parâmetro não precisaria ser calibrado como no modelo matemático descrito no capítulo 7, cuja influência do diâmetro da “bolha” no resultado do TDG é muito importante.

Figura 84 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6 \text{ mm}$; $0,8 \text{ mm}$ e $1,0 \text{ mm}$.



Fonte: Autor (2025).

Na Figura 85 é apresentado o cálculo do TDG ao longo da bacia de dissipação utilizando o modelo de autômato celular, com a relação ar-água (α_{bolha}) igual a 3% e 5% e diâmetro da “bolha” igual a 0,8 mm.

Figura 85 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $\alpha_b = 5\%$ $d_b = 0,8\text{mm}$.

Fonte: Autor (2025).

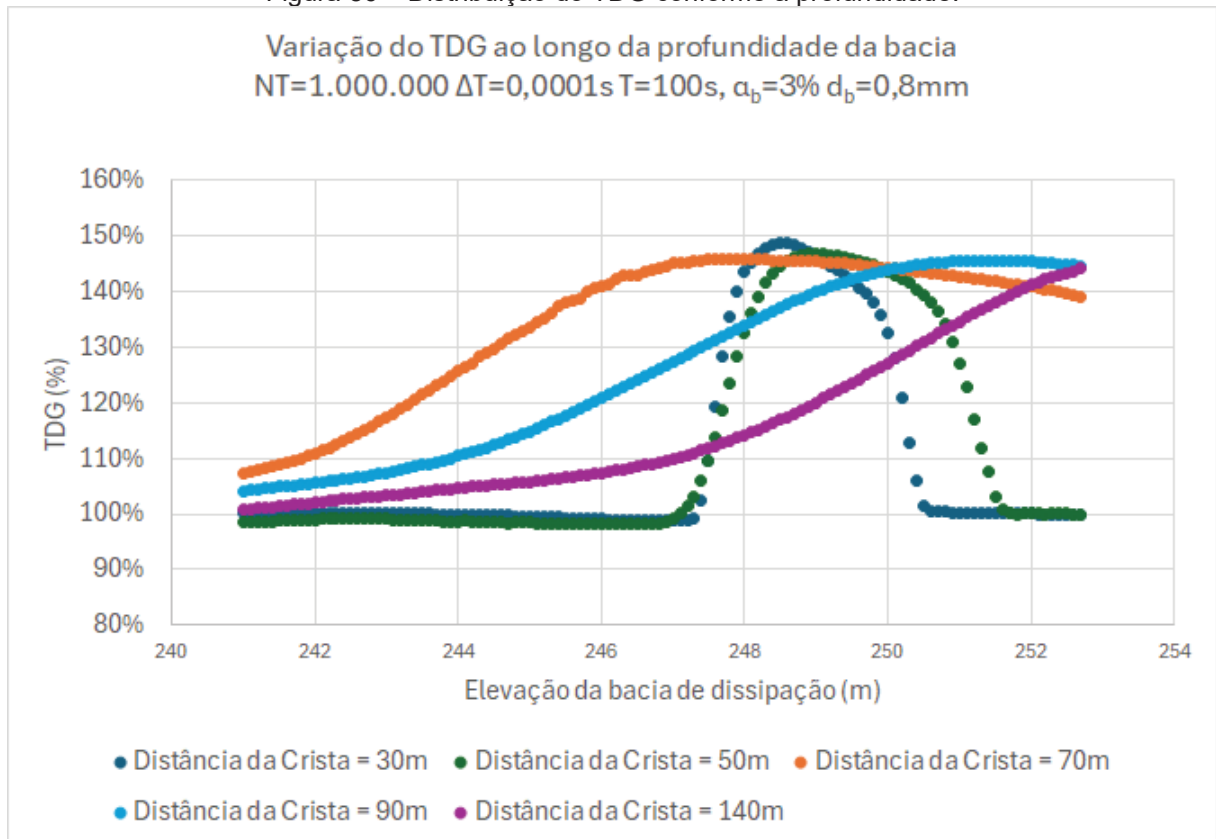
O Quadro 2 apresenta as diferenças nos valores de TDG máximos obtidos na bacia de dissipação em função da variação da relação ar-água entre 3% e 5%. Observa-se que a diferença entre os valores máximos foi de 3 pontos percentuais (150% vs 153%, respectivamente). Considerando a média dos valores máximos de TDG a diferença também foi de 3 pontos percentuais (140% vs 143%, respectivamente). Por fim, ao se analisar a média dos valores de TDG em todas as células da malha, aproximadamente 230.000 células, a diferença foi de 2 pontos percentuais (117% vs 119%).

Na Figura 85 é possível verificar uma diferença relativamente pequena e constante do TDG máximo ao longo da bacia de dissipação, considerando a relação ar-água de 3% e 5%. Nos estudos desenvolvidos por Wang, Politano e Weber (2018), a sensibilidade do TDG ao percentual da relação ar-água é mais acentuada do que a obtida com o modelo com autômato celular. Na Figura 36 é possível verificar que a diferença do TDG é de aproximadamente 30 pontos percentuais, entre a simulação de TDG considerando a relação ar-água de 1% e 3%. Essa variação é muito superior à obtida pelo modelo de autômato celular.

A Figura 86 apresenta a distribuição do TDG ao longo do eixo z (profundidade) para diversas distâncias da crista do vertedouro (30 m, 50 m, 70 m, 90 m e 140 m). Nela é possível verificar que para as distâncias de 30 m e 50 m a distribuição do TDG

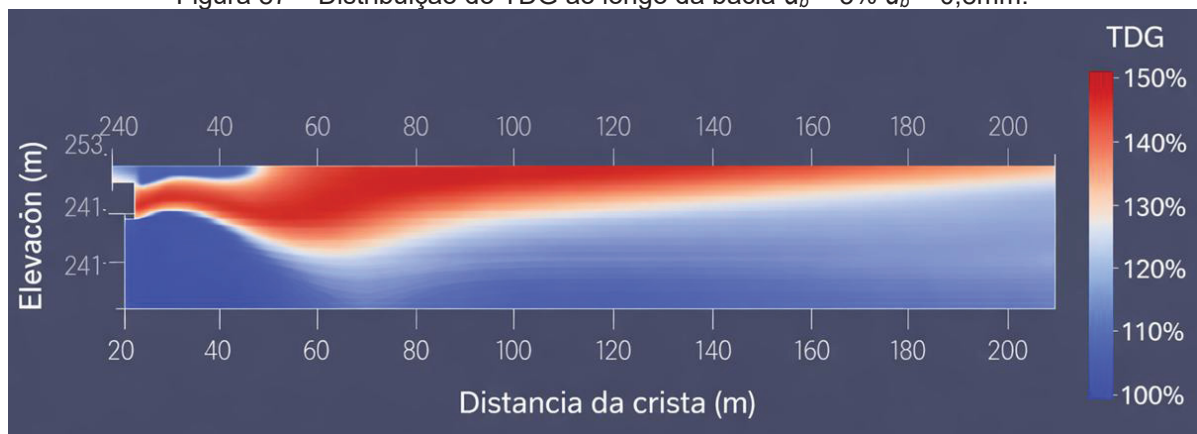
é muito influenciada pela área de admissão das “bolhas”, que se distribui entre a elevação do defletor 247 m e o nível de água de jusante. Já para as maiores distâncias da crista do vertedouro (distâncias de 70 m a 140 m) a distribuição do TDG é influenciada pela velocidade do escoamento no eixo z, ilustrada na Figura 88, e pelas equações de advecção/difusão da concentração de TDG. A Figura 87 apresenta a distribuição do TDG ao longo da bacia de dissipação para $\alpha_b = 3\%$ e $d_b = 0,8\text{mm}$.

Figura 86 – Distribuição do TDG conforme a profundidade.



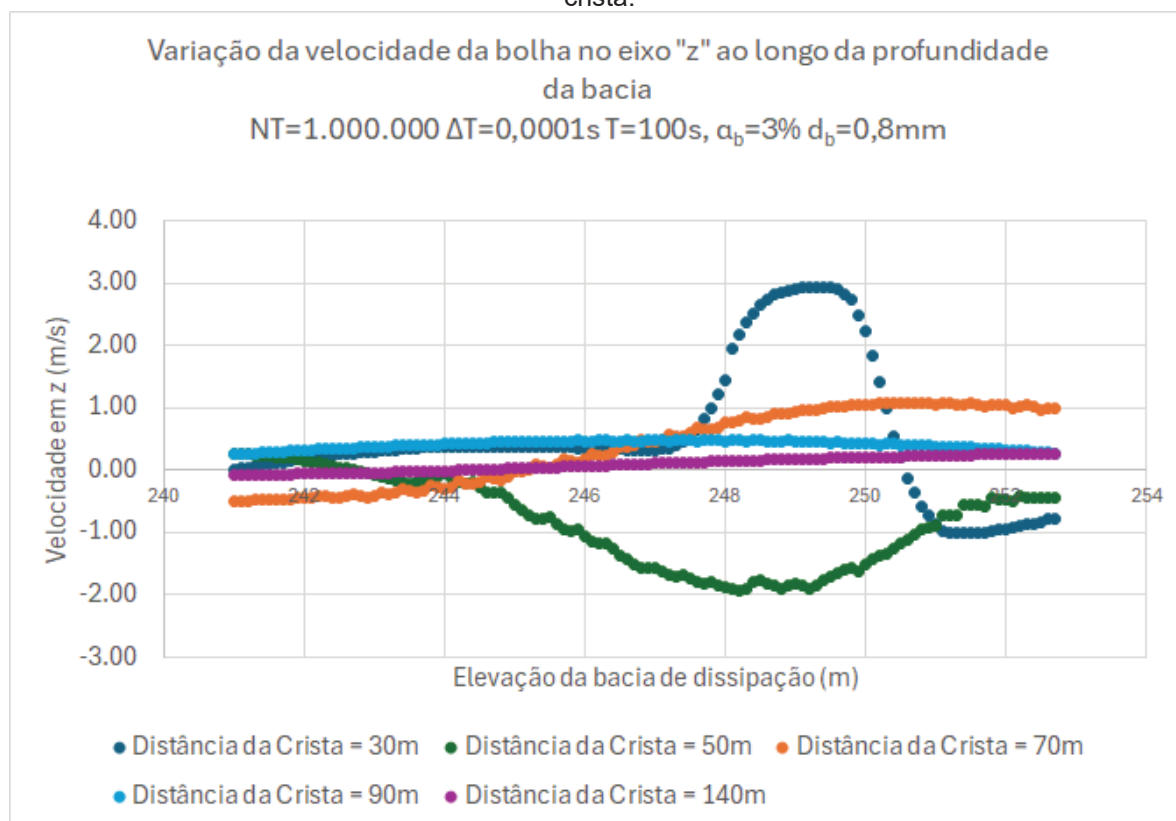
Fonte: Autor (2025).

Figura 87 – Distribuição do TDG ao longo da bacia $\alpha_b = 3\%$ $d_b = 0,8\text{mm}$.



Fonte: Autor (2025).

Figura 88 – Distribuição da velocidade da “bolha” no eixo z conforme a profundidade e distância da crista.



Fonte: Autor (2025).

Os resultados de TDG obtidos utilizando o modelo de autômato celular se mostraram promissores, apresentando valores máximos e distribuição ao longo da bacia de dissipação, comparáveis com os estudos desenvolvidos por Politano et al. (2022) e por Ovelar (2022).

8.5 RESULTADOS PARA $N_t = 5.000.000$

Nesta seção são apresentados os resultados do cálculo do TDG obtidos por meio de autômatos celulares, considerando um total de 5.000.000 de passos de tempo, equivalente a 500 s de simulação. Para cada um dos 6 casos analisados, os parâmetros de diâmetro da “bolha” e relação ar-água (α_{bolha}) foram variados. O tempo de processamento foi de aproximadamente 60 horas, utilizando um computador pessoal (Intel i7-13850HX 2,1GHz).

O modelo de cálculo de TDG utilizando autômatos celulares descrito na seção 8.3, considerou como região de injeção de oxigênio o comprimento de 10 m, entre as células localizadas na distância de 17,7 m a 27,7 m da crista do vertedouro, e entre

as profundidades 247 m a 253,2 m (crista do defletor e nível de água a jusante). O defletor está localizado entre as células 17,7 m e 24,7 m, na elevação 247 m.

O TDG foi simulado variando os seguintes parâmetros: a) diâmetro inicial da “bolha” considerando (d_b) igual a 0,6 mm, 0,8 mm e 1,0 mm e b) relação ar-água (α_{bolha}), igual a 3% e 5%. A simulação considerou vazão vertida de 1.600 m³/s (400 m³/s por vão e vazão específica de 33,3 m³/s.m), nível de jusante na elevação 253,22 m, que equivale a uma vazão defluente de 3.200 m³/s (1.600 m³/s de vazão vertida e 1.600 m³/s de vazão turbinada), que caracteriza um escoamento a jusante do tipo *skimming flow*.

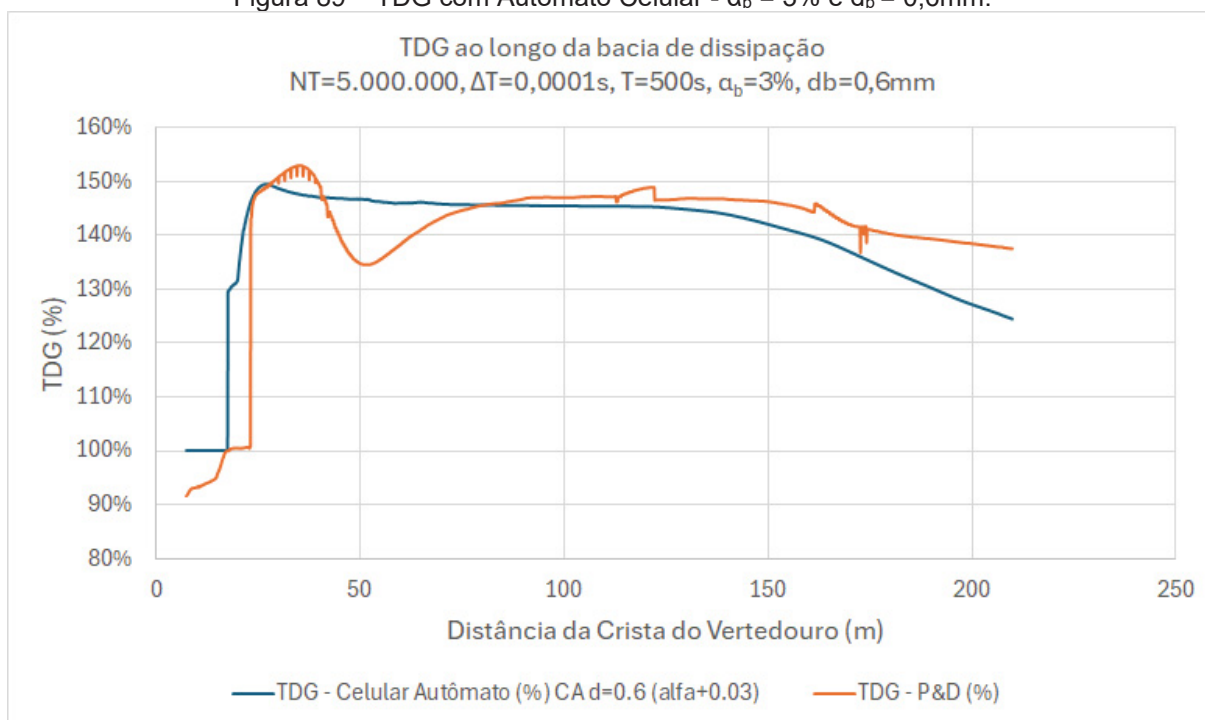
O Quadro 3 apresenta as seguintes colunas: 1) número do ensaio; 2) relação ar-água (α_{bolha}); 3) Diâmetro da “Bolha” (mm); 4) “TDG Máximo”, que representa o maior valor de TDG obtido em uma célula ao longo de toda a simulação; 5) “TDG Média Máxima”, que corresponde à média dos valores máximos de TDG em cada célula ao longo do eixo x e, 6) “TDG Médio em toda a Bacia”, que considera o valor médio em todas as células da malha, aproximadamente 230.000 células.

Quadro 3 – Simulações de TDG.

Nº Ensaio	α_{bolha} (%)	Diâmetro da “Bolha” (mm)	TDG Máximo (%)	TDG Média Máxima (%)	TDG Médio em toda a Bacia (%)
1	3	0,60	150	140	115
2	3	0,80	150	140	115
3	3	1,00	150	140	115
4	5	0,60	153	142	117
5	5	0,80	153	142	117
6	5	1,00	153	142	117

Os resultados obtidos utilizando autômatos celulares foram comparados com os resultados obtidos no projeto de P&D e apresentados em Ovelar (2022). A Figura 89 apresenta este comparativo, considerando os parâmetros do ensaio nº 1, apresentado no Quadro 3, com $\alpha_b = 3\%$ e $d_b = 0,6$ mm.

Figura 89 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6\text{mm}$.



Fonte: Autor (2025).

Os resultados apresentados na Figura 89 podem ser comparados com a curva de performance do vertedouro da UHE Colíder, apresentados em Andriolo et al. (2023) e reproduzidos na Figura 10. A sonda de TDG responsável pelos dados foi instalada na bacia de dissipação, na elevação 246 m, aproximadamente 54 m da crista do vertedouro. Para a vazão de $1.600\text{ m}^3/\text{s}$ ($33,3\text{ m}^3/\text{s.m}$), a curva de performance indica um TDG de aproximadamente 140%. Considerando os dados de TDG e vazão vertida atualizado até maio de 2025, apresentados na Figura 11, o valor extrapolado de TDG seria de aproximadamente 172% para a mesma vazão vertida de $1.600\text{ m}^3/\text{s}$. É importante destacar que, como a maior vazão vertida registrada após a construção do defletor foi de $1.050\text{ m}^3/\text{s}$, com TDG medido de 145%, não é possível afirmar que o TDG atingiria 172%, uma vez que a dinâmica de dissolução de gases pode variar conforme maior seja o “colchão” de água na bacia de dissipação e as condições de vazão turbinada.

Os valores máximos de TDG simulados utilizando o modelo de AC resultou em TDG de 153%, indicando aderência com os valores medidos em campo e com os valores simulados por Ovelar (2022).

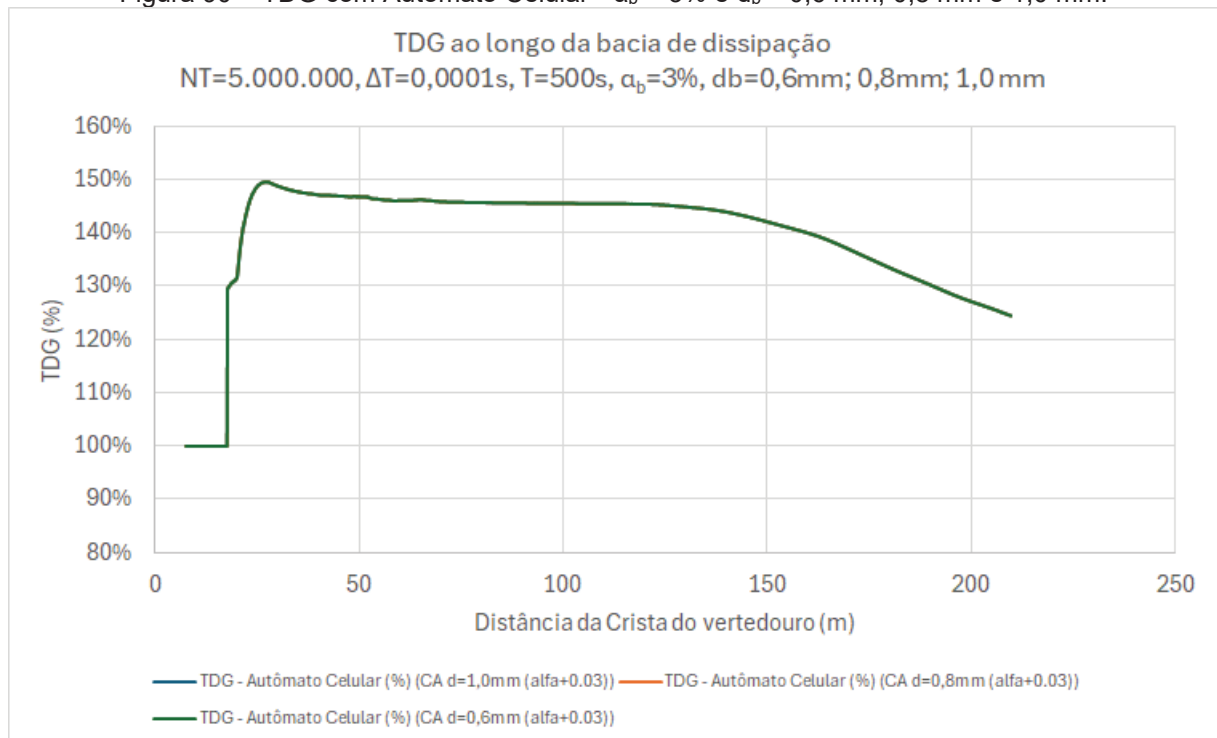
O valor máximo do TDG, igual a 153%, foi verificado a 23,9 m da crista do vertedouro, praticamente ao final do defletor que está localizado a 24,7 m. O decaimento do valor do TDG ao longo da bacia, é compatível com o decaimento

apresentado no estudo de Politano et al. (2022) e reproduzido na Figura 33. Em relação ao decaimento do TDG ao longo da bacia de dissipação os resultados obtidos com autômatos celulares, apresentam um comportamento mais próximos aos outros estudos (decaimento lento) do que os resultados obtidos com o modelo matemático descrito no capítulo 7.

Comparando os resultados de TDG do modelo de autômato celular proposto nesta tese com os resultados simulados por Ovelar (2022), o aumento do TDG na região de entrada das “bolhas” apresenta um padrão muito semelhante, com valores máximos quase coincidentes. Aproximadamente a 50 m a jusante da crista do vertedouro, o modelo de Ovelar mostra uma rápida redução do TDG seguida por um subsequente incremento, que pode ser atribuído a uma maior concentração de “bolhas” de ar arrastadas ainda não dissolvidas (ver Figura 78). Como o modelo de autômato celular não incluiu essa informação como dado de entrada (relação ar-água simulado no OpenFOAM), esse comportamento localizado não foi reproduzido. No entanto, a cerca de 70 m da crista, os valores de TDG obtidos por ambas as abordagens são muito próximos, demonstrando a robustez do modelo de autômato celular. A 130 m da crista do vertedouro a redução do TDG simulada pelo modelo de autômato celular torna-se mais pronunciada do que a observada nos resultados de Ovelar. Em contraste, a simulação realizada por Politano et al. (2022) (Figura 33) apresenta uma redução mais lenta do TDG ao longo da bacia de dissipação, embora a tendência geral e os valores máximos de TDG permaneçam consistentes entre os dois modelos.

A Figura 90 apresenta o cálculo de TDG com autômato celular ao longo da bacia de dissipação, considerando relação ar-água (α_{bolha}) igual a 3% e com diâmetro da “bolha” de 0,6 mm, 0,8 mm e 1,0 mm. As simulações do modelo de AC mostram que o diâmetro da “bolha” não é um parâmetro sensível para o resultado dos valores máximos de TDG ao longo da bacia, mesmo com o diâmetro da “bolha” sendo um parâmetro de entrada das equações (8.13), (8.14) e (8.16). Isso indica que este parâmetro não precisaria ser calibrado como no modelo matemático apresentado no capítulo 7, cuja influência do diâmetro da “bolha” no resultado do TDG é muito importante.

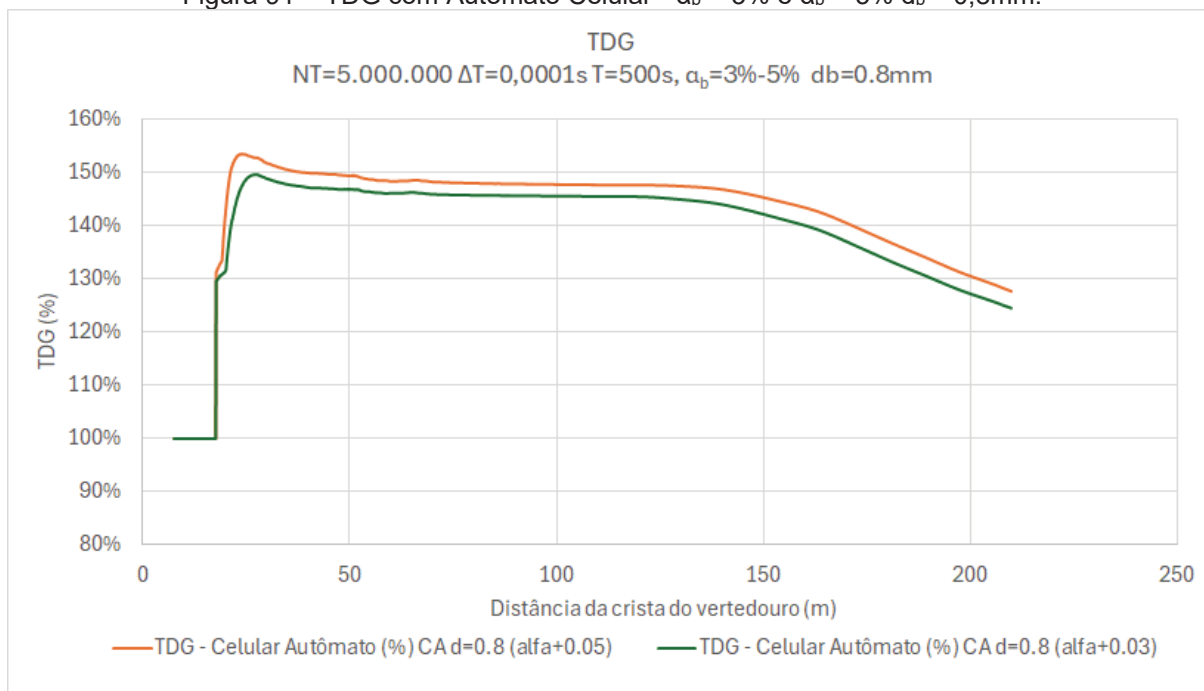
Figura 90 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $d_b = 0,6 \text{ mm}$; $0,8 \text{ mm}$ e $1,0 \text{ mm}$.



Fonte: Autor (2025).

Na Figura 91 é apresentado o cálculo do TDG ao longo da bacia de dissipação utilizando o modelo de autômato celular, com a relação ar-água (α_{bolha}) igual a 3% e 5% e diâmetro da “bolha” igual a 0,8 mm.

Figura 91 – TDG com Autômato Celular - $\alpha_b = 3\%$ e $\alpha_b = 5\%$ $d_b = 0,8\text{mm}$.



Fonte: Autor (2025).

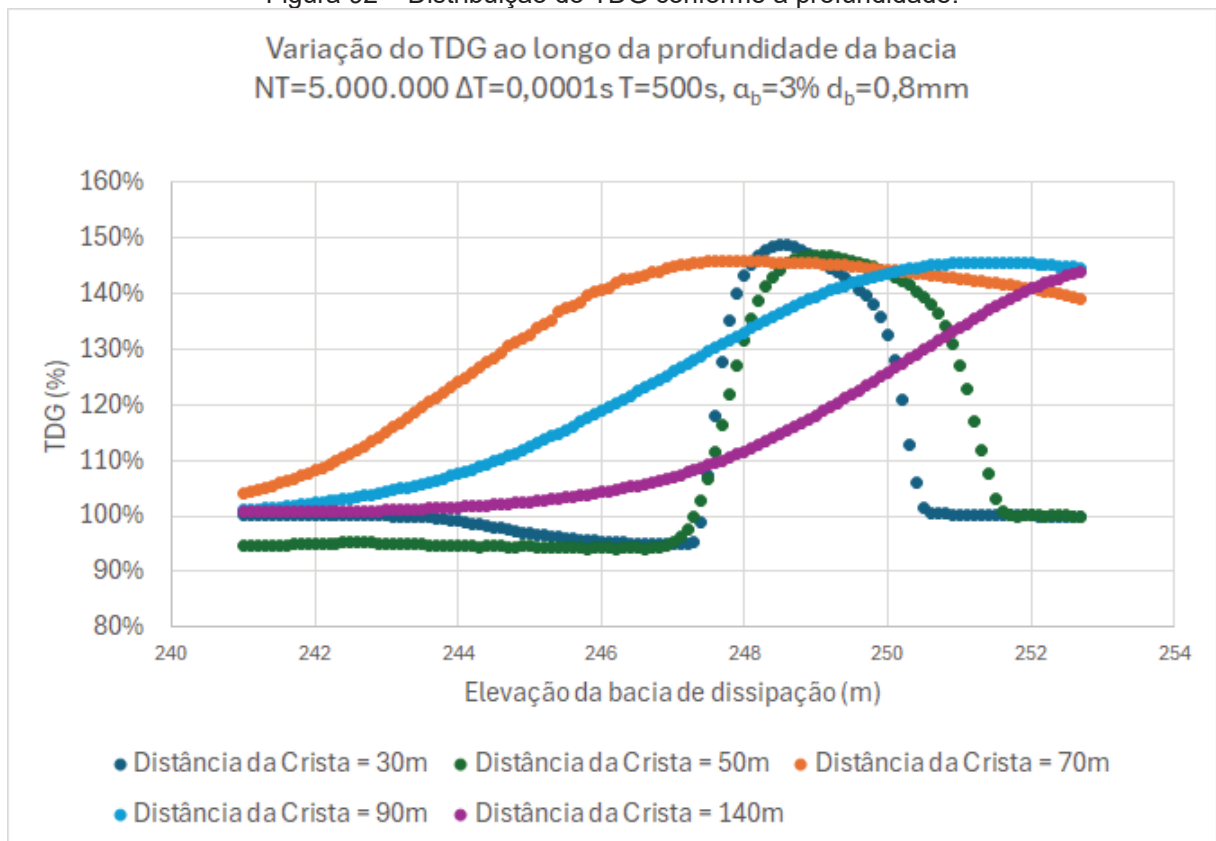
Conforme apresentado no Quadro 3 a diferença entre o máximo valor de TDG verificado na bacia de dissipação, considerando a variação na relação ar-água entre 3% e 5% foi de 3 pontos percentuais (150% vs 153%, respectivamente). Considerando a média dos valores máximos do TDG a diferença foi de 2 pontos percentuais (140% vs 142%, respectivamente). Considerando a média dos valores de TDG em todas as células da malha, aproximadamente 230.000 células, a diferença é de 2 pontos percentuais (115% vs 117%, respectivamente).

Na Figura 91 é possível verificar uma diferença relativamente pequena e constante do TDG máximo ao longo da bacia de dissipação, para a relação ar-água de 3% e 5%. Nos estudos desenvolvidos por Wang, Politano e Weber (2018), a sensibilidade do TDG ao percentual da relação ar-água é mais acentuada do que a obtida com o modelo com autômato celular. Na Figura 36 é possível verificar que a diferença do TDG é de aproximadamente 30 pontos percentuais, entre o TDG simulado considerando a relação ar-água de 1% e 3%. Essa variação é muito superior à obtida pelo modelo de autômato celular.

A Figura 92 apresenta a distribuição do TDG ao longo do eixo z (profundidade) para diversas distâncias da crista do vertedouro (30 m, 50 m, 70 m, 90 m e 140 m). Nela é possível verificar que nas distâncias de 30 m e 50 m a distribuição do TDG é

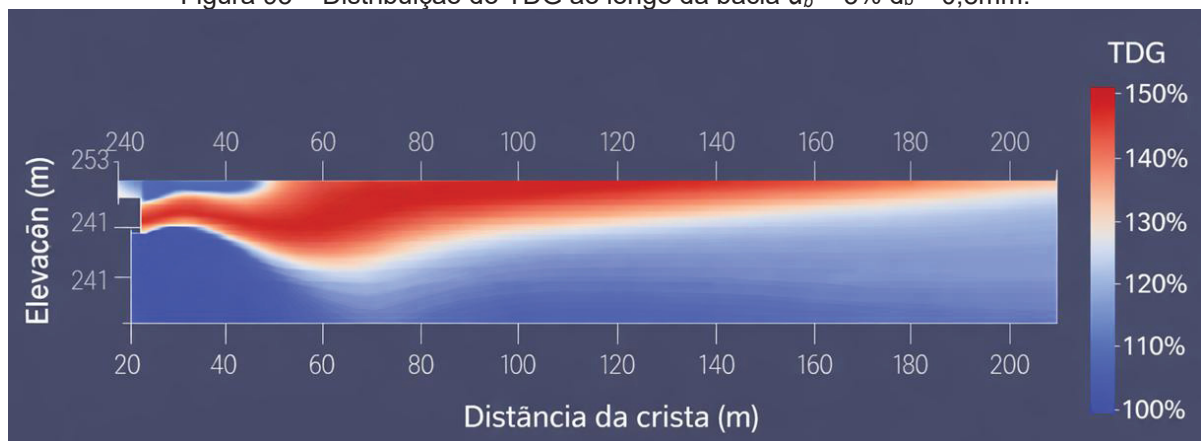
muito influenciada pela região de admissão das “bolhas”, que se distribui entre a elevação do defletor 247 m e o nível de água de jusante. Já para as maiores distâncias da crista do vertedouro (distâncias de 70 m a 140m) a distribuição do TDG é influenciada pela velocidade do escoamento no eixo z, mostrada na Figura 94, e pelas equações de advecção/difusão da concentração de TDG. A Figura 93 apresenta a distribuição do TDG ao longo da bacia de dissipação para $\alpha_b = 3\%$ e $d_b = 0,8\text{mm}$.

Figura 92 – Distribuição do TDG conforme a profundidade.



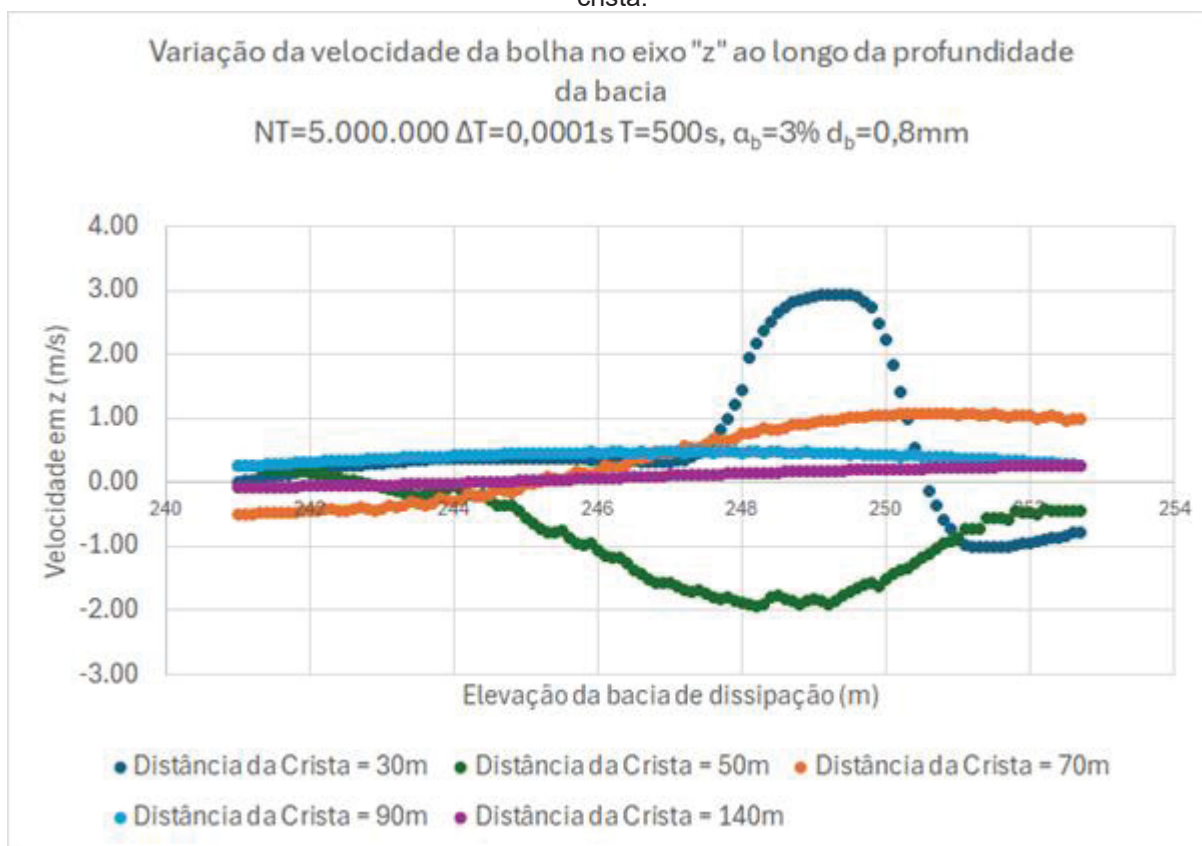
Fonte: Autor (2025).

Figura 93 – Distribuição do TDG ao longo da bacia $\alpha_b = 3\%$ $d_b = 0,8\text{mm}$.



Fonte: Autor (2025).

Figura 94 – Distribuição da velocidade da “bolha” no eixo z conforme a profundidade e distância da crista.

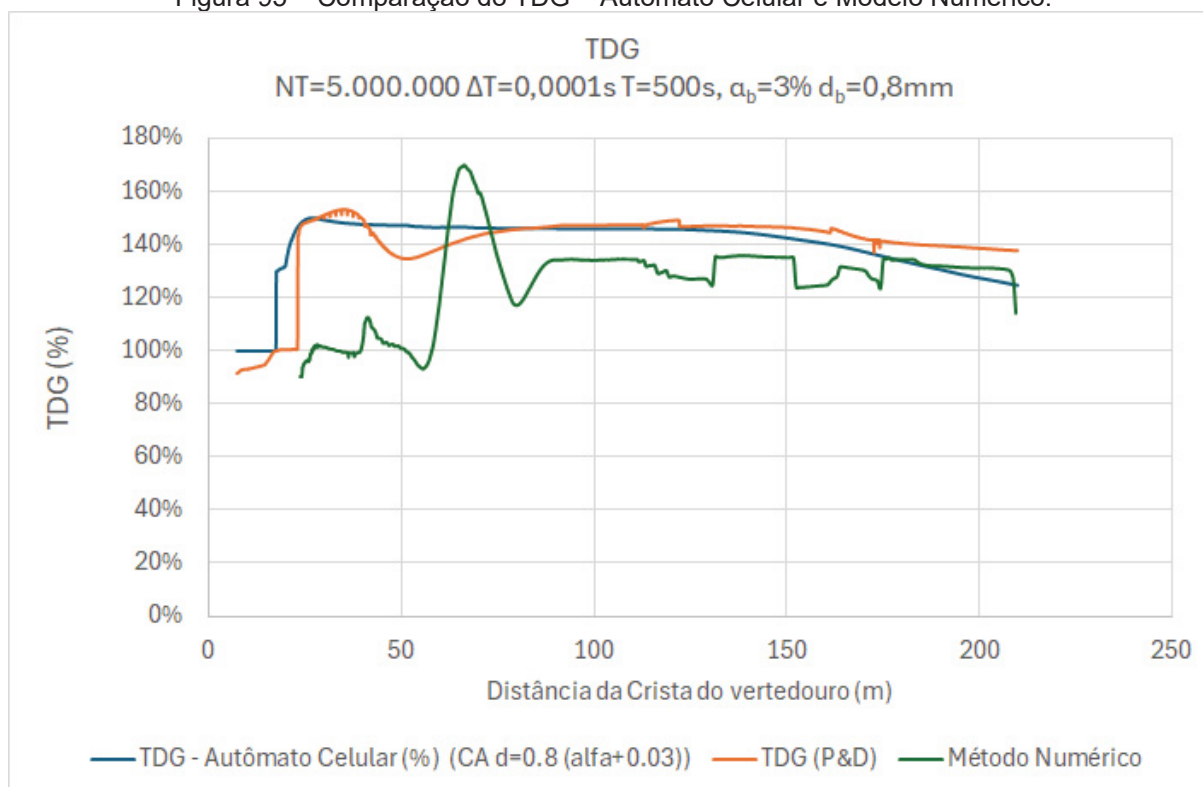


Fonte: Autor (2025).

Os resultados de TDG simulados com o modelo de autômato celular mostraram ser promissores, apresentando valores máximos e distribuição ao longo da bacia de dissipação, comparáveis com os estudos desenvolvidos por Politano et al. (2022) e por Ovelar (2022).

A Figura 95 apresenta um comparativo da simulação do TDG ao longo da bacia de dissipação, considerando os dois modelos descritos nesta tese, o modelo matemático, apresentado no capítulo 7, o modelo de autômato celular, apresentado no capítulo 8 e o modelo desenvolvido por Ovelar (2022). O TDG calculado pelos três modelos utilizou como parâmetro inicial a relação ar-água (α_b) igual a 3% e o diâmetro da “bolha” (d_b) igual a 0,8 mm, sendo os valores máximos simulados de TDG de 150%, 153% e 170% (autômato celular, modelo proposto por Ovelar e modelo matemático, respectivamente).

Figura 95 – Comparação do TDG – Autômato Celular e Modelo Numérico.



Fonte: Autor (2025).

Em relação ao comportamento do TDG ao longo da bacia, o modelo matemático apresentado no capítulo 7, baseado nas equações utilizadas no modelo apresentado por Huang et al. (2019) e descrito no capítulo 4, apresenta uma influência mais pronunciada da relação ar-água, parâmetro que foi calculado pelo OpenFOAM.

8.6 CONCLUSÃO SOBRE O CÁLCULO DE TDG COM AUTÔMATOS CELULARES

Neste capítulo foi apresentada a formulação matemática e computacional para o cálculo do total de gases dissolvidos utilizando o modelo com autômatos celulares. Nas 12 simulações realizadas os seguintes parâmetros foram avaliados: a) diâmetro inicial da “bolha” considerando (d_b) igual a 0,6 mm, 0,8 mm e 1,0 mm; b) relação ar-água (α_{bolha}), igual a 3% e 5% e c) Número de passos de tempo (N_t) igual a 1.000.000 e 5.000.000, equivalente ao tempo de simulação de 100 segundos e 500 segundos, respectivamente.

O diâmetro da “bolha” não mostrou ser um parâmetro relevante para o valor máximo do TDG simulado pelo modelo de autômatos celulares, e o parâmetro ar-água interfere pouco no resultado.

Os valores máximos de TDG simulados com autômatos celulares apresentam valores próximos (diferença inferior a 10 pontos percentuais) aos estudos desenvolvidos por Politano et al. (2022) e por Ovelar (2022). O comportamento do decaimento do TDG ao longo da bacia de dissipação também apresenta comportamento similar com ambos os estudos.

A distribuição dos valores de TDG calculadas pelo modelo de autômatos celulares, ao longo da profundidade (eixo z), principalmente no fundo da bacia de dissipação, ilustrada na Figura 87 e na Figura 93, apresentou diferença em relação aos estudos desenvolvidos por Politano et al. (2022) e por Ovelar (2022) (ver Figura 76). Esse fato pode ser explicado pelo deslocamento da fração de ar (“bolhas”), que é “arrastada” pelo escoamento ao longo da bacia de dissipação, desde o ponto de injeção de ar, através do vetor velocidade do fluxo no eixo da profundidade (eixo z). Na condição “estabilizada” do modelo hidrodinâmico, o vetor velocidade no eixo z , apresenta velocidades próximas e até iguais a zero, assim, não há arraste de “bolhas” até o fundo da bacia de dissipação. Essa constatação pode ser corroborada pelo perfil médio da concentração de ar apresentada na Figura 67 (estudo Ovelar (2022)). Como o TDG calculado no modelo de autômato celular considera o deslocamento da fração de ar (“bolhas”), deste o ponto de injeção, não existindo arraste do ar, não há alteração do valor de TDG devido ao processo advectivo, apenas a variação devido ao processo difusivo. Assim, como as variações dinâmicas que ocorrem na distribuição da velocidade ao longo do tempo (de forma instantânea), principalmente no eixo z , não estão representadas na condição estabilizada do modelo hidrodinâmico, o modelo de autômato celular também não representa essas possíveis variações.

Em relação ao resultado de TDG apresentado na Figura 80, cuja formulação está baseada no modelo matemático apresentado em Huang et al. (2019), descrito no capítulo 7, e o modelo de autômato celular, as diferenças podem ser atribuídas principalmente:

- a) o modelo de autômato celular não usa a fração de ar-água (α_{bolha}) simulada no modelo hidrodinâmico, e sim o vetor velocidade do fluxo e como ele transporta a fração molar de O_2 entre as células que compõem a malha;
- b) no modelo de autômato celular o diâmetro da bolha demonstrou que a diferença dos resultados pode ser considerada desprezível;
- c) no modelo de autômato celular a relação ar-água demonstrou que a diferença nos resultados é pequena.

A grande vantagem do modelo de autômato celular é que ele requer apenas um computador pessoal para fazer as estimativas de TDG, ao invés de um cluster computacional como os outros modelos. Outra vantagem é em relação aos parâmetros de relação ar-água e diâmetro da “bolha” que mostraram ser parâmetros com uma pequena interferência no resultado do TDG, facilitando o processo de calibração dos modelos.

9 CONCLUSÕES E RECOMENDAÇÕES

A supersaturação do total de gás dissolvido (TDG) na água pode provocar a doença da bolha na ictiofauna, e considerando a crescente preocupação com a mitigação dos impactos ambientais, o estudo e quantificação do TDG pode se tornar um importante parâmetro a ser avaliado nos novos estudos de Viabilidade, Projeto Básico e Projeto Executivo de usinas hidroelétricas.

O modelo de autômato celular desenvolvido nesta tese, pode contribuir nos estudos iniciais do total de gás dissolvido nos fluxos efluentes de usinas hidroelétricas, bem como no comportamento do TDG ao longo da bacia de dissipação, canal de restituição e rio, permitindo uma avaliação preliminar deste parâmetro de qualidade da água.

9.1 CONCLUSÕES

O modelo de autômatos celulares desenvolvido e implementado nesta tese, calculou os valores de TDG ao longo da bacia de dissipação, sendo que os valores máximos do TDG apresentaram diferença inferior a dez pontos percentuais quando comparados com outros estudos. Cabe destacar que diferenças de 10 pontos percentuais são verificadas em campo para a mesma vazão vertida, devido principalmente a flutuações nas vazões turbinadas e na temperatura da água.

A distribuição dos valores de TDG calculadas pelo modelo de autômato celulares, ao longo da profundidade (eixo z), principalmente no fundo da bacia de dissipação, apresentou diferença em relação aos outros estudos apresentados nesta tese. Esse fato pode ser explicado pelo deslocamento da fração de ar (“bolhas”), que é “arrastada” pelo escoamento ao longo da bacia de dissipação, desde o ponto de injeção de ar, através do vetor velocidade perpendicular ao fluxo principal (eixo z). Na condição “estabilizada” do modelo hidrodinâmico, o vetor velocidade perpendicular ao fluxo (eixo z), apresenta velocidades próximas e até iguais a zero, assim, não há arraste de “bolhas” até o fundo da bacia de dissipação. Como o TDG calculado no modelo de autômato celular considera o deslocamento da fração de ar (“bolhas”), deste o ponto de injeção, não existindo arraste do ar, não há alteração do valor de TDG devido ao processo advectivo, apenas a variação devido ao processo difusivo. Assim, como as variações dinâmicas que ocorrem na distribuição da velocidade ao longo do tempo (de forma instantânea), principalmente perpendicular ao fluxo (eixo z),

não estão representadas na condição estabilizada do modelo hidrodinâmico, o modelo de autômato celular também não representa essas variações.

O menor esforço computacional requerido pelo modelo de autômato celular, pode facilitar estudos iniciais com o objetivo de compreender o comportamento do TDG ao longo da bacia, seus valores máximos e avaliar as diversas condições de operação de uma usina hidroelétrica.

Outra vantagem do modelo de autômato celular é que os resultados demonstraram que os parâmetros de entrada: a) fração de ar e b) diâmetro da bolha, influenciam pouco nos valores máximos de TDG, com diferença de 3 pontos percentuais, o que facilita o processo de calibração do modelo, principalmente em situações em que não há dados observados em campo.

9.2 RECOMENDAÇÕES

Recomenda-se avaliar o modelo de autômato celular para outras configurações de vazões vertidas e níveis de água de jusante, de forma a refletir as variações no funcionamento das turbinas da usina (com alteração nas vazões turbinadas), abrangendo uma gama maior de condições operativas.

A aplicação do modelo de autômato celular nos projetos de outras usinas hidroelétricas pode contribuir na validação do modelo para outras geometrias de vertedouro e condições de jusante.

Outro ponto a se avaliar é em relação a influência dos vetores de velocidade do escoamento na distribuição do TDG utilizando o modelo autômatos celulares, pois, como este modelo considera o vetor velocidade na transferência do TDG de uma célula para outra, e o tempo de simulação adotado considera que o percentual de TDG esteja em condição estável, talvez, seja possível reduzir o refinamento da malha utilizada no OpenFOAM para calcular essa variável, reduzindo assim o esforço computacional desta etapa.

Recomenda-se avaliar a possibilidade de utilizar o modelo de autômato celular acoplado ao método do reticulado de Boltzmann para a modelagem computacional dos parâmetros hidrodinâmicos do escoamento, permitindo a determinação dos campos de velocidade e pressão. Dessa forma, seria possível substituir as simulações hidrodinâmicas realizadas no OpenFOAM, simplificando o processo de obtenção dos valores de TDG e reduzindo o esforço computacional em todo o processo.

10 REFERÊNCIAS

- ABARBANEL, H.; CASE, K.; DESPAIN, A.; DYSON, F.; FREEMAN, M. **Cellular Automata and Parallel Processing for Practical Fluid-Dynamics Problems**. [s.l: s.n.].
- AGOSTINHO, A. A.; ALVES, D. C.; GOMES, L. C.; DIAS, R. M.; PETRERE JR, M.; PELICICE, F. M. Fish die-off in river and reservoir: A review on anoxia and gas supersaturation. **Neotropical Ichthyology**, v. 19, n. 3, 2021.
- AGRAWAL, H. Construction of Molecular Dynamics Like Cellular Automata Models for Simulation of Compressible Fluid Dynamic Systems. **arXiv: Cellular Automata and Lattice Gases**, 1999.
- ALBERT, J.; REIS, R. **Historical Biogeography of Neotropical Freshwater Fishes**. [s.l: s.n.].
- ANDERSSON, B.; ANDERSSON, R.; HAKANSSON, L.; MORTENSEN, M.; SUDIYO, R.; WACHEM, B. VAN. **Computational fluid dynamics for engineers**. [s.l: s.n.].
- ANDRIOLO, M. V.; ARAÚJO, A. N. DE; FONTANELLI, C.; PARANHOS, C. S. A.; POLITANO, M.; DETTMER, P. H. C. Installation of deflector in Colider Dam to minimize the percentual of total dissolved gases. **Revista Brasileira de Recursos Hídricos**, v. 28, 2023.
- ANDRIOLO M. V.; ARAÚJO, A. N.; FONTANELLI, C.; PARANHOS, C. S. A.; DETTMER, P. H. C. **Hydraulic Performance Curves to Support the Design of Spillway Deflectors in Colíder Dam** (Hydrovision, Ed.)Denver: Hydrovision, 2022
- ANDRIOLO, M. V; KAVISKI, E. Modelo de Autômato Celular para Simulação da Propagação de um Hidrograma ao Longo de um Canal. **XXV Simpósio Brasileiro de Recursos Hídricos, Aracaju, SE**, 2023.
- ANDRIOLO, M. V.; KAVISKI, E. **Simulação da série temporal de massa liberada de CO2 utilizando autômato celular** (Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, Ed.)Proceeding Series of the Brazilian Society of Computational and Applied Mathematics. **Anais...**2024Disponível em: <https://www.researchgate.net/publication/384310782_Simulacao_da_Serie_Temporal_de_Massa_Liberada_de_CO2_Utilizando_Automato_Celular>. Acesso em: 25 out. 2025

ANTCLIFFE, B. L.; FIDLER, L. E.; BIRTWELL, I. K. **Effect of prior exposure to hydrostatic pressure on rainbow trout (*Oncorhynchus mykiss*) survival in air-supersaturated water.** [s.l: s.n.].

BACCHIEGA, J.; FATTOR, C. Yacyretá Dam: spillways modification to reduce total dissolved gases concentrations downstream of the dam. *Em*: [s.l: s.n.]. p. 77–84.

BENGT ANDERSSON; RONNIE ANDERSSON; MIKAEL MORTENSEN; LOVE HÅKANSSON; RAHMAN SUDIYO. **Computational Fluid Dynamics for Engineers.** 1st edition ed. [s.l.] Cambridge University Press, 2011.

BESAGNI, G.; INZOLI, F.; ZIEGENHEIN, T. Two-Phase Bubble Columns: A Comprehensive Review. **ChemEngineering**, v. 2, p. 13, out. 2018.

BIN, A. Gas Entrainment by Plunging Liquid Jet. **Chemical Engineering Science**, v. 48, p. 3585–3630, abr. 1993.

BRADY J.E., R. J. W.; J.R., H. **Química: A Matéria e suas Transformações Químicas.** [s.l.] LTC, 2002.

BRENNEN, C. Cavitation and Bubble Dynamics. **Cavitation and Bubble Dynamics**, v. 44, abr. 1995.

BRESSLOFF, N. W. A parallel pressure implicit splitting of operators algorithm applied to flows at all speeds. **International Journal for Numerical Methods in Fluids**, v. 36, n. 5, p. 497–518, 2001.

BUCKUP, P.; MENEZES, N.; GHAZZI, M. **Catálogo das Espécies de Peixes de Água Doce do Brasil.** [s.l: s.n.]. v. 1

BUNG D.; VALERO D. **FLOWCV - AN OPEN - SOURCE TOOLBOX FOR COMPUTER VISION APPLICATIONS IN TURBULENT FLOWS** 37th IAHR World Congress. **Anais...** Kuala Lumpur, Malaysia: IAHR, 13 ago. 2017

BYE, E. T. **Investigation of Elementary Cellular Automata for Reservoir Computing.** [s.l.] Norwegian University of Science and Technology, 2016.

CARGNELUTTI, J. **Simulação do Escoamento em Canais com Obstáculos Submersos por Meio do Método do Reticulado de Boltzmann.** Curitiba: UFPR, 2017.

CARGNELUTTI, J.; GALINA, V.; KAVISKI, E.; GRAMANI, L.; LOBEIRO, A. Two-dimensional numerical simulation of channel flow with submerged obstacles using the lattice Boltzmann method. **Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería**, v. 34, abr. 2017.

- CARVALHO D. DE S. **Classifying the Complexity and Information of Cellular Automata**. [s.l.] Wolfram Demonstrations Project, 2011.
- CASTRO, A. M.; LI, J.; CARRICA, P. M. A mechanistic model of bubble entrainment in turbulent free surface flows. **International Journal of Multiphase Flow**, v. 86, p. 35–55, 2016.
- CASTRO-ORGAZ, O. Velocity Profile and Flow Resistance Models for Developing Chute Flow. **Journal of Hydraulic Engineering**, v. 136, n. 7, p. 447–452, jul. 2010.
- CEHPAR - LACTEC. Relatório Técnico 1: Revisão bibliográfica. 2020.
- CENGEL, Y. A.; CIMBALA, J. M. **Fluid mechanics: Fundamentals and applications**. 4th ed. ed. Columbus, OH: McGraw-Hill Education, 2017.
- CHANSON, H. **Air Bubble Entrainment in Free-Surface Turbulent Shear Flows**. [s.l.] Elsevier, 1996.
- CHANSON, H. **Air Bubble Entrainment in Hydraulic Jumps: Similitude and Scale Effects**. [s.l.: s.n.].
- CHANSON, H. **Air Bubble Entrainment in Hydraulic Jumps: Physical Modeling and Scale Effects** (Proceedings of the 32nd IAHR World Congress, Ed.)Chanson, H. **Anais...**Venice: 2007
- CHANSON, H.; AOKI, S.; MARUYAMA, M. Unsteady air bubble entrainment and detrainment at a plunging breaker: dominant time scales and similarity of water level variations. **Coastal Engineering**, v. 46, n. 2, p. 139–157, 2002.
- CHEN, Q. **Cellular Automata and Artificial Intelligence in Ecohydraulics Modelling**. [s.l.] CRC Press, 2004.
- CHEN, X.-P. Simulation of 2D Cavitation Bubble Growth Under Shear Flow by Lattice Boltzmann Model. **Communications in Computational Physics - COMMUN COMPUT PHYS**, v. 7, abr. 2009.
- CHOPARD, B.; MASSELOT, A. Cellular automata and lattice Boltzmann methods: A new approach to computational fluid dynamics and particle transport. **Future Generation Computer Systems**, v. 16, p. 249–257, jan. 1999.
- COLOMBET, D.; LEGENDRE, D.; RISSO, F.; COCKX, A.; GUIRAUD, P. Dynamics and mass transfer of rising bubbles in a homogenous swarm at large gas volume fraction. **Journal of Fluid Mechanics**, v. 763, p. 254–285, out. 2014.
- COLT, J. **Dissolved Gas Concentration in Water**. [s.l.] Elsevier, 2012.
- COLT, J. E. The computation and reporting of dissolved gas levels. **Water Research**, v. 17, n. 8, p. 841–849, 1983.

- DÉSÉRABLE, D.; DUPONT, P.; HELLOU, M.; KAMALI-BERNARD, S. Cellular Automata in Complex Matter. **Complex Systems**, v. 20, abr. 2011.
- DETTMER, P. H. C. **Estudos Hidráulicos em Modelo Reduzido do Vertedouro da UHE Colíder**. Curitiba: [s.n.].
- _____. **PD-06491-0541/2019 - Metodologia para modelagem computacional de TDG na água em fluxos efluentes de vertedouros - Relatório da Etapa 4: Estudos em Modelo Físico**. Curitiba: [s.n.].
- DOMITROVIC, H. A.; BECHARA, J. A.; JACOBO, W. R.; FLORES QUINTANA, C. I.; ROUX, J. P. Mortandad de peces en el río Paraná provocada por sobresaturación de gases: Causas y lesiones. **Revista de Ictiología 2/3 (1/2):** , p. 49–54, 1993.
- DONAVAN A. A. A.; KERNIGHAN B. W. **The Go Programming Language**. Indiana: [s.n.].
- DOTSON, C. **Wanapum and Priest Rapids Dams** 145th Annual Meeting of the American Fisheries Society. **Anais...**2015
- DOTTORI, F.; TODINI, E. A 2D flood inundation model based on cellular automata approach. abr. 2010.
- DOUILLET-GRELLIER, T.; LECLAIRE, S.; VIDAL, D.; BERTRAND, F.; VUYST, F. DE. **Comparison of multiphase SPH and LBM approaches for the simulation of intermittent flows**, abr. 2019.
- FELDER, R. M.; ROUSSEAUD, R. W. **Princípios Elementares dos Processos Químicos**. [s.l.] LTC, 2005.
- FERREIRA, E. J. G.; J. A. S. ZUANON. **Peixes Comerciais Do Médio Amazonas : Região de Santarém, Pará**. Edições IBAMA ed. Brasília: [s.n.].
- FERZIGER, J. H.; PERIC, M. **Computational methods for fluid dynamics**. [s.l.: s.n.]. v. 34
- FIDLER, L. E. **Gas bubble trauma in fish**. [s.l.: s.n.].
- FRISCH, U.; HASSLACHER, B.; POMEAU, Y. Lattice-Gas Automata for the Navier-Stokes Equation. **Physical Review Letters**, v. 56, n. 14, p. 1505–1508, 7 abr. 1986.
- FUSTER, D. A Review of Models for Bubble Clusters in Cavitating Flows. **Flow, Turbulence and Combustion**, v. 102, 1 mar. 2019.
- GALINA, W. Simulação do Escoamento em Bacias Hidrográficas Utilizando o Método do Reticulado de Boltzmann. **Tese de Doutorado, UFPR, Curitiba**, 2017.
- GALINA, W. **Simulação do Escoamento em Bacias Hidrográficas Utilizando o Método do Reticulado de Boltzmann**. Curitiba: UFPR, 2017.

- GELDERT, D. A.; GULLIVER, J. S.; WILHELMS, S. C. Modeling Dissolved Gas Supersaturation Below Spillway Plunge Pools. **Journal of Hydraulic Engineering**, v. 124, p. 513–521, maio 1998.
- GREENSHIELDS, C. J. **OpenFOAM User's Guide**. [s.l.: s.n.].
- GUZELLA, M.; CZELUSNIAK, L.; PESSOA MAPELLI, V.; RIBEIRO, G.; CABEZAS-GÓMEZ, L. Simulação numérica tridimensional da ascensão de uma bolha e da coalescência vertical de duas bolhas em um meio líquido. **CQD Revista Eletrônica Paulista de Matemática**, v. 14, p. 34–46, 1 fev. 2019.
- HADJERIOUA, B.; POLITANO, M. S.; BENDER, M. D.; STEWART, K. Prediction of Total Dissolved Gas Exchange at Hydropower Dams: Mid-Columbia River System. **US Department of Energy**, n. September, p. 81, 2014.
- HADJERIOUA, B.; POLITANO, M. S.; DENEALE, S.; BENDER, M. D.; CASTRO, A. Predicting Total Dissolved Gas (TDG) for the Mid-Columbia River System. **USBR Publications**, p. 1–32, 2014.
- HERRERO, V.; GUIDO LAVALLE, G.; CLAUSSE, A. Non-equilibrium effects in void fraction distribution. **Nuclear Engineering and Design**, v. 154, p. 183–192, out. 1995.
- HO, M. **Modelling of Bubbly Flows in CFD Using the Intersection Marker Front-Tracking Method**. [s.l.] UNSW Sydney, 2016.
- HUANG, J.; LI, J.; POLITANO, M.; LI, R.; CARRICA, P. **Modeling Air Entrainment Downstream of Spillways** 38th IAHR World Congress. **Anais...** 1 set. 2019 Disponível em: <https://www.iahr.org/paper/detail?paper_id=3451>
- INCROPERA, F. P.; DEWITT, D. P.; BERGMAN, T. L.; LAVINE, A. S. **Fundamentals of Heat and Mass Transfer**. [s.l.] J. Wiley, 2007.
- ISSA, R. I. Solution of the implicitly discretised fluid flow equations by operator-splitting. **Journal of Computational Physics**, v. 62, n. 1, p. 40–65, 1986.
- JASAK, H. Dynamic Mesh Handling in OpenFOAM. 5 jan. 2009.
- JONES, O. C.; ZUBER, N. The interrelation between void fraction fluctuations and flow patterns in two-phase flow. **International Journal of Multiphase Flow**, v. 2, n. 3, p. 273–306, 1975.
- JUNK, W.; BAYLEY, P.; SPARKS, R. **The Flood Pulse Concept in River-Floodplain Systems** Can. Spec. Public Fish. Aquat. Sci. **Anais...** out. 1989

- KANTIC, J.; LEGL, F. C.; STECHELE, W.; HERMANN, J. ReLiCADA: Reservoir Computing Using Linear Cellular Automata design algorithm. **Complex & Intelligent Systems**, 2024.
- KAVISKI E. **Solução de problemas de fenômenos de transporte pelo método de Monte Carlo**. Curitiba: UFPR, 2006.
- KHAWAJA, H.; MOATAMED, M. Semi-implicit method for pressure-linked equations (SIMPLE) ↓ solution in MATLAB®. **The International Journal of Multiphysics**, v. 12, p. 313, out. 2018.
- KIER, L. B.; SEYBOLD, P. G.; CHENG, C. K. **Modeling Chemical Systems Using Cellular Automata**. [s.l.: s.n.].
- KRAMER, M.; VALERO, D.; CHANSON, H.; BUNG, D. B. Towards reliable turbulence estimations with phase-detection probes: an adaptive window cross-correlation technique. **Experiments in Fluids**, v. 60, n. 1, p. 2, 26 jan. 2019.
- KREMER, G. M. **Uma Introdução à Equação de Boltzmann**. São Paulo: EDUSP, 2005.
- LACTEC. **PD-06491-0541/2019 - Metodologia para modelagem computacional de TDG na água em fluxos efluentes de vertedouros - Relatório da Etapa 1**. Curitiba: [s.n.].
- LEWIN, R. **Complexity: Life and Edge of Chaos**. Macmillian ed. [s.l.] Rocco, 1993.
- LI, P.; ZHU, D. Z.; LI, R.; WANG, Y.; CROSSMAN, J. A.; KUHN, W. L. Production of total dissolved gas supersaturation at hydropower facilities and its transport: A review. **Water Research**, v. 223, p. 119012, set. 2022.
- LI, Q.; LUO, K.; KANG, Q.; HE, Y.; CHEN, Q.; LIU, Q. Lattice Boltzmann methods for multiphase flow and phase-change heat transfer. **Progress in Energy and Combustion Science**, v. 52, 4 ago. 2015.
- LI, R.; MA, Q.; FENG, J.; WANG, L. RELATIONSHIPS BETWEEN TOTAL TDG AND DISSOLVED OXYGEN IN WATER. n. January 2013, 2018.
- LOWE-MCCONNELL, R. H. Estudos Ecológicos de Comunidades de Peixes Tropicais. **São Paulo: EDUSP. 514p.**, 1999.
- LUNDBERG, J. G.; KOTTELAT, M.; SMITH, G. R.; STIASSNY, M. L. J.; GILL, A. C. So Many Fishes, So Little Time: An Overview of Recent Ichthyological Discovery in Continental Waters. **Annals of the Missouri Botanical Garden**, v. 87, n. 1, p. 26–62, 2000.

- MACHADO, J. P.; GARLING JR., D. L.; KEVERN, N. R.; TRAPP, A. L.; BELL, T. G. Histopathology and the Pathogenesis of Embolism (Gas Bubble Disease) in Rainbow Trout (*Salmo gairdneri*). **Canadian Journal of Fisheries and Aquatic Sciences**, v. 44, n. 11, p. 1985–1994, 1 nov. 1987.
- MAGALHÃES, D. A.; SOUZA CRUZ, F. F. Sistema climático e corpo humano: conceitos da física. **Revista Brasileira de Ensino de Física**, 2021.
- MAULE, A. G.; BEEMAN, J. W.; MESA, M. G.; HANS, K. M.; ROBINSON, T. C.; VANDERKOOI, S. P.; HANER, P. V.; WEILAND, L.; ADMINISTRATION, B. P. **Gas bubble trauma monitoring and research of juvenile salmonids**. [s.l: s.n.]. Disponível em: <<https://pubs.usgs.gov/publication/70180169>>.
- MENEZES, N. Methods for assessing freshwater fish diversity. *Em*: [s.l: s.n.]. p. 289–295.
- MENTER, F. R. Zonal Two Equation k-w , Turbulence Models for Aerodynamic Flows. **24th Fluid Dynamics Conference July 6-9, 1993 / Orlando, Florida**, 1993.
- MYERS, R.; PARKINSON, S. E. **Hells Canyon Complex Total Dissolved Gas Study**. Idaho: [s.n.].
- NEBEKER, A. V.; KENT HAUCK, A.; BAKER, F. D. Temperature and oxygen-nitrogen gas ratios affect fish survival in air-supersaturated water. **Water Research**, v. 13, n. 3, p. 299–303, 1979.
- NIEDZWIEDZKA, A. HOMOGENEOUS CAVITATION MODELING-ANALYSIS OF BASICS OF MATHEMATICAL FORMULATION OF SOURCE TERMS. **Modelowanie Inżynierskie**, 1 dez. 2017.
- OERTEL, H. **Prandtl's Essentials of Fluid Mechanics**. New York: Springer, 2004.
- ORLINS, J.; GULLIVER, J. Dissolved gas supersaturation downstream of a spillway II: Computational model. **Journal of Hydraulic Research - J HYDRAUL RES**, v. 38, p. 151–159, 1 mar. 2000.
- VELAR, C. O. S. **Relatório Técnico Final: Etapa 6**, 2022.
- VELAR, C. O. S.; TAKENOBU, M. Y. M. **Projeto PD-06491-0541/2019 metodologia para modelagem computacional de TDG na água em fluxos efluentes de vertedouros. Relatório Técnico Final**. Curitiba: [s.n.].
- VELAR, C. O. S.; TAKENOBU, M. Y. M.; DETTMER, P. H. C.; ANDRIOLO, M. V.; PARANHOS, C. S. A.; OTA, J. J. Total Dissolved Gases (TDG) Supersaturation Reduction at Colíder Dam: First Spill with Gorizonta Deflectors Verification. **REVISTA BRASILEIRA DE ENGENHARIA DE BARRAGENS**, p. 23–29, dez. 2022.

- PAIVA, A.; PETRONETTO, F.; LEWINER, T.; TAVARES, G. Particle-based viscoplastic fluid/solid simulation. **Computer-Aided Design**, v. 41, p. 306–314, 1 abr. 2009.
- PAPOULIAS, D.; GAVAISES, M. Modelling of single bubble-dynamics and thermal effects. **Journal of Physics: Conference Series**, v. 656, n. 1, p. 12098, nov. 2015.
- PINTO, N. L. DE S. **Hidráulica Aplicada**. Curitiba: Centro de Hidráulica e Hidrologia Professor Parigot de Souza - Universidade Federal do Paraná, 1987.
- POLITANO, M.; HUANG, J.; ANDRIOLO, M. V.; ARAÚJO, A. N.; LYONS, T. O. **A Numerical Modeling to Support the Design of Spillway Deflectors in Colider Dam** Denver: Hydrovision, 2022
- POLITANO, M.; HUANG, J.; LYONS, T.; VINICIUS ANDRIOLO, M.; NASCIMENTO DE ARAUJO, A. Performance Evaluation of Spillway Deflectors with a Multiphase Flow Model. **Journal of Hydraulic Engineering**, v. 150, n. 2, 2024.
- POLITANO, M. S.; ARENAS AMADO, A.; BICKFORD, S.; MURAUSKAS, J.; HAY, D. Evaluation of operational strategies to minimize gas supersaturation downstream of a dam. **Computers and Fluids**, v. 68, p. 168–185, 2012.
- POLITANO, M. S.; CARRICA, P. M.; TURAN, C.; WEBER, L. A multidimensional two-phase flow model for the total dissolved gas downstream of spillways. **Journal of Hydraulic Research**, v. 45, n. 2, p. 165–177, 2007.
- POLITANO, M. S.; CARRICA, P.; WEBER, L. A multiphase model for the hydrodynamics and total dissolved gas in tailraces. **International Journal of Multiphase Flow**, v. 35, n. 11, p. 1036–1050, 2009.
- POLITANO, M. S.; LAUGHERY, R. **A two-phase flow model to evaluate the effect of fish passage structures on the hydrodynamics and TDG downstream of McNary Dam** (G. Constantinescu, M. Garcia, & D. Hanes, Eds.) Proceedings of International Conference on Fluvial Hydraulics - River Flow 2016. **Anais...** St. Louis: 2016
- POLITANO, M. S.; LYONS, T.; ANDERSON, K.; PARKINSON, S. Spillway Deflector Design Using Physical and Numerical Models. **6th International Symposium on Hydraulic Structures**, n. June, p. 27–30, 2016.
- POLITANO, M. S.; WANG, Y.; LAUGHERY, R.; WEBER, L. A Numerical Model for Spillway Jet Regimes and Total Dissolved Gas. **Hydro Research Foundation**, v. 357, n. July, p. 339–357, 2015.
- PRUD'HOMME, R. **Flows of Reactive Fluids**. [s.l: s.n.]. v. 94

PULG, U.; VELLE, G.; KVALSVIK STENBERG, S.; SKOGLUND, H. Acute effects of gas supersaturation on Atlantic salmon smolt in two Norwegian rivers.

Hydrobiologia, v. 1, 1 jan. 2022.

RABÓCZKAY, T. **Físico-Química de Interfaces**. São Paulo: EDUSP, 2016.

RISSE, F. Agitation, Mixing, and Transfers Induced by Bubbles. **Annual Review of Fluid Mechanics**, v. 50, n. Volume 50, 2018, p. 25–48, 2018.

SANZ, V.; URQUIA, A.; LEVA, A. 1D/2D Cellular Automata Modeling with Modelica. **Proceedings of the 10th International Modelica Conference; Lund; Sweden**, 2014.

SCHIFF, J. L. **Cellular Automata: A Discrete View of the World**. [s.l.] John Wiley e Sons, Inc, 2008.

SCHMIDT, S. J. **A Low Mach Number Consistent Compressible Approach for Simulation of Cavitating Flows**. München: Technische Universität München, 2015.

SENE, K. J. Air entrainment by plunging jets. **Chemical Engineering Science**, v. 43, n. 10, p. 2615–2623, 1988.

SHUR, M.; SPALART, P. R.; STRELETS, M.; TRAVIN, A. Detached-eddy simulation of an airfoil at high angle of attack. **Engineering Turbulence Modelling and Experiments** 4, p. 669–678, 1999.

SILVA, P. L. DA; FREIRE, I. L. On the group analysis of a modified Novikov equation. *Em*: COJOCARU, M.; KOTSIREAS, I.; MAKAROV, R.; MELNIK, R.; SHODIEV, H. (Eds.). **Interdisciplinary Topics in Applied Mathematics, Modeling and Computational Science, Springer Proceedings in Mathematics and Statistics**. [s.l.] Springer, 2015. v. 117p. 161–166.

SILVA, T. C.; GRAMANI, L. M.; KAVISKI, E. **Análise comparativa da lotação e da disposição das saídas no processo de evacuação de um cinema: simulação via autômatos celulares** Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE), 2011

SOUZA, J. F. A.; OLIVEIRA, L. R. DE; AZEVEDO, J. L. L. DE; SOARES, I. D.; MATA, M. M. Uma revisão sobre a turbulência e sua modelagem. **Revista Brasileira de Geofísica**, v. 29, n. 1, p. 21–41, 2011.

SPALART, P. R.; JOU, W. H.; STRELETS, M.; ALLMARAS, S. R. Comments on the feasibility of LES for wings and on a hybrid RANS/LES approach. **Advances in DNS/LES**, v. 1, n. JANUARY, p. 4–8, 1997.

- SUBRATA, P. **Cellular Automata: Temporal Stochasticity and Computability**. [s.l.] Indian Institute of Engineering Science and Technology, 2022.
- SUKOP, M. C.; THORNE-JR., D. T. **Lattice Boltzmann Mod Model: An introduction for geoscientists and engineers**. [s.l.] Springer, 2006.
- TAKENOBU, M. Y. M. **Relatório Técnico Final: Etapa 5**, 2022.
- TAKENOBU, M. Y. M.; OVELAR, C. O. S.; DETTMER, P. H. C.; ANDRIOLO, M. V.; FONTANELLI, C. **Physical and Numerical Hydraulic Modeling of a Spillway with Horizontal Deflectors** Proceedings of the IAHR World Congress. **Anais...**2022
- TANENBAUM, A. S.; AUSTIN, T. **Organização Estruturada de Computadores**. 6ª Edição ed. São Paulo: Pearson, 2013.
- TOMÉ, T.; OLIVEIRA, M. J. **Dinâmica Estocástica e Irreversibilidade**. São Paulo: EDUSP, 2014.
- URBAN, A. L.; GULLIVER, J. S.; JOHNSON, D. W. Modeling Total Dissolved Gas Concentration Downstream of Spillways. **Journal of hydraulic Engineering - ASCE**, v. 134, n. 5, p. 550–561, 2008.
- USEPA. **EPA's Action on Revisions to the Washington State Department of Ecology's Surface Water Quality Standards for the Site-Specific Total Dissolved Gas Criteria in the Columbia and Snake Rivers, and Other Water Quality Standards Revisions**. Washington: [s.n.].
- VALERO, D.; KRAMER, M.; BUNG, D. B.; CHANSON, H. **A Stochastic Bubble Generator for Air-Water Flow Research**1 set. 2019
- VERSTEEG, H. K.; MALALASEKERA, W. **An Introduction to Parallel Computational Fluid Dynamics**. [s.l.: s.n.]. v. 6
- WANG, H.; FELDER, S.; CHANSON, H. An experimental study of turbulent two-phase flow in hydraulic jumps and application of a triple decomposition technique. **Experiments in Fluids**, v. 55, n. 7, 2014.
- WANG, Y.; POLITANO, M.; WEBER, L. Spillway jet regime and total dissolved gas prediction with a multiphase flow model. **Journal of Hydraulic Research**, v. 57, p. 1–13, 18 abr. 2018.
- WEBER, L. J.; HAUG, P. E. **Performance Curves for Wanapum Dam Spillway Deflectors**Seattle: Water Resources Convention - ASCE, 1999
- WEEEN, P. VAN DER. **Meso-scale modeling of reaction-diffusion processes using cellular automata**. [s.l.] Ghent University, 2014.

- WEITKAMP, D. E. Total Dissolved Gas Supersaturation Biological Effects , Review of Literature 1980-2007. **Review Literature And Arts Of The Americas**, p. 60, 2008.
- WEITKAMP, D. E.; KATZ, M. A Review of Dissolved Gas Supersaturation Literature. **Transactions of the American Fisheries Society**, v. 109, n. 6, p. 659–702, 1980.
- WILCOX, D. **Turbulence Modeling for CFD (Third Edition) (Hardcover)**. [s.l: s.n.].
- WINEMILLER, K. *et al.* Balancing Hydropower and Biodiversity in the Amazon, Congo, and Mekong. **Science**, v. 351, p. 128–129, 14 jan. 2016.
- WOLF, F. G.; SANTOS, L. O. E. DOS; PHILIPPI, P. C. Formação e dinâmica da interface líquido-vapor simulada pelo método Lattice-Boltzmann. **Revista Brasileira de Ensino de Física**, v. 28, n. 2, p. 167–175, jun. 2006.
- WOLF-GLADROW, D. A. **Lattice Gas Cellular Automata and Lattice Boltzmann Models**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. v. 1725
- WOLFRAM, S. A. **Cellular Automata and Complexity**. [s.l: s.n.].
- _____. **New Kind of Science**. [s.l.] Wolfram Media, 2002.
- WU, J.; SHU, C. A solution-adaptive lattice Boltzmann method for two-dimensional incompressible viscous flows. **Journal of Computational Physics**, v. 230, n. 6, p. 2246–2269, 2011.
- WYLIE, B. J. N. **Application of Two-dimensional Cellular Automaton Lattice-Gas Models to the Simulation of Hydrodynamics. Doctor dissertation**. Edinburgh: University of Edinburgh, 1990.
- YANG, X.-S.; YOUNG, Y. **Cellular Automata, PDEs, and Pattern Formation**, 2010.
- ZHANG, C.; CHENG, Y.; WU, J.; DIAO, W. Lattice Boltzmann simulation of the open channel flow connecting two cascaded hydropower stations. **Journal of Hydrodynamics, Ser. B**, v. 28, p. 400–410, 1 jun. 2016.
- ZHANG, J.; YAN, G. Lattice Boltzmann method for one and two-dimensional Burgers equation. **Physica A: Statistical Mechanics and its Applications**, v. 387, n. 19, p. 4771–4786, 2008.
- _____. A lattice Boltzmann model for the Burgers-Fisher equation. **Chaos (Woodbury, N.Y.)**, v. 20, p. 023129, 1 jun. 2010.
- ZHOU, J. G. **Lattice Boltzmann Method for Shallow Water Flows**. Queensland: The University of Queensland, 2004.

APÊNDICE I – Leitura de arquivo (Ler_Velocidade)

Programa desenvolvido em Go para realizar a leitura dos parâmetros de saída do OpenFOAM, após os mesmos serem salvos em arquivo txt pelo software ParaFoam.

```
package main
/*
    Leitura dos vetores de velocidade, pressão e coordenadas dos arquivos
    de saída do OpenFOAM, arquivos U e p.
*/

import (//"encoding/binary"
        "os"
        "fmt"
        "math"
        "bufio"
        "strings"
        "strconv"
        "math/rand"
        "time"
        "path/filepath"
        "runtime"
        "ler_U_OpenFOAM/velocidade_bolha/velocidade_bolha"
    )

var (arq,
    ark *os.File)
var narq string
var err error
var numero_dados int
var numero_dados_tdg int
var Fundo_bacia int // calcula o fundo da bacia e relaciona com o ponto de inicio do array utilizando a função
velocidade_bolha.Inicializa_fundo_bacia (x float64) float64

const elv_na_jusante = velocidade_bolha.Elv_na_jusante // Elevação do nível de água de jusante
const lim_na_jus = velocidade_bolha.Lim_na_jus // Elevação do nível de água de jusante (inteiro)
const elv_reservatorio = velocidade_bolha.Elv_reservatorio // Elevação do nível de água de montante

func ler_arq_aci () {

    // esta função lê o arquivo de saída do Parafoam, contendo dados de : coordenadas (x, y, z), velocidade (x,y,z) , velocidade
    média (x,y,z), alpha.water,alphaMean.water,p
    // Lê os arquivos da Etapa ue contém apenas as simulações hidrodinâmicas
    var i int

    // diretório de localização dos dados do paraFoam

    //narq := "C:/Users/mvand/OneDrive/PED_TDG/PD-6491-0541-
    2019/SIMULACOES_COMPUTACIONAIS/ETAPA_5/SIMULACOES_COLIDER/PROTOTOPI/COM_DEFLETOR/Q1600_253.2
    1_malha_final"

    // este aruivo contém os dados com defletor
    //narq := "/mnt/c/Users/mvand/OneDrive/PED_TDG/PD-6491-0541-
    2019/SIMULACOES_COMPUTACIONAIS/ETAPA_5/SIMULACOES_COLIDER/PROTOTOPI/COM_DEFLETOR/Q1600_253.2
    1_malha_final" substituido pasta do arquivo 25/10/25
    //narq := "/mnt/c/Users/mvand/OneDrive/PED_TDG/PD-6491-0541-
    2019/SIMULACOES_COMPUTACIONAIS/ETAPA_5/SIMULACOES_COLIDER/PROTOTOPI/COM_DEFLETOR/Q300_247.11"
    // narq := "/mnt/c/Users/mvand/OneDrive/PED_TDG/PD-6491-0541-
    2019/SIMULACOES_COMPUTACIONAIS/ETAPA_6/WANG2018_MODIFICADO/EVENTO4_Q800_q200_250.75_0.8mm"
    // Obtém a função que chamou esta função (neste caso, main)
    _, filename, _, ok := runtime.Caller(0)
    if !ok {
        panic("Erro ao obter informações do arquivo")
    }
}
```

```

// Obtém o diretório onde o arquivo está instalado
dir := filepath.Dir(filename)
fmt.Println("O diretório de instalação é:", dir)

// nome do arquivo utilizado
// filePath := narq + "/tempo_2060.csv" // Q_1600 Q_300 - 24/10/25 arquivo original colocado nesta pasta:
filePath := dir + "/dados_OpenFOAM/Q1600_25321_Defletor.csv"
//filePath := narq + "/tempo_Q_800_250.csv"

// Open the file
file, err := os.Open(filePath)
if err != nil {
    fmt.Println("Error opening file:", err)
    return
}
defer file.Close()

// Create a new scanner to read the file line by line
scanner := bufio.NewScanner(file)

i=0
// Read each line
for scanner.Scan() {
    if i == 0 {
        scanner.Scan()
        fmt.Println("Descartada a primeira linha")
    } // descarta o cabeçalho do arquivo

    line := scanner.Text()
    //fmt.Println(line)

    // Split the line into individual values
    values := strings.Split(line, ",")

    // Convert each data for float64
    velocidade_bolha.Leitura_vetor_parafoam.T, err = strconv.ParseFloat(values[0], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.X, err = strconv.ParseFloat(values[1], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Y, err = strconv.ParseFloat(values[2], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Z, err = strconv.ParseFloat(values[3], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Ux, err = strconv.ParseFloat(values[4], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Uy, err = strconv.ParseFloat(values[5], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Uz, err = strconv.ParseFloat(values[6], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Umx, err = strconv.ParseFloat(values[7], 64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
    }
}

```

```

        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Umy, err = strconv.ParseFloat(values[8],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.U mz, err = strconv.ParseFloat(values[9],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Alpha_water, err = strconv.ParseFloat(values[10],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.AlphaMean_water, err = strconv.ParseFloat(values[11],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Pressao, err = strconv.ParseFloat(values[12],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam.Pressao_mean, err = strconv.ParseFloat(values[13],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Dado_parafoam =
append(velocidade_bolha.Dado_parafoam,velocidade_bolha.Leitura_vetor_parafoam) // armazena os dados de leitura em um
array
    i = i+1

} // fim do for de leitura do arquivo

numero_dados = i

} // fim da função ler_arq_aci

func ler_arq_tdg () { // leitura do arquivo de saída do parafoam para os dados de TDG do P&D

    // esta função lê o arquivo de saída do Parafoam, contendo dados de : coordenadas (x, y, z), velocidade (x,y,z) ,
    velocidade média (x,y,z), alpha.water,alphaMean.water,p

    var i int

    // diretório de localização dos dados do paraFoam

    // lê os dados da Etapa 6 contendo as simulações do TDG
    // narq := "C:/Users/mvand/OneDrive/PED_TDG/PD-6491-0541-
2019/SIMULACOES_COMPUTACIONAIS/SIMULACOES_COMPUTACIONAIS/ETAPA_6/WANG2018_MODIFICADO/EVENT
O4_Q800_q200_250.75_0.8mm"
    //narq := "/mnt/c/Users/mvand/OneDrive/PED_TDG/PD-6491-0541-
2019/SIMULACOES_COMPUTACIONAIS/ETAPA_6/WANG2018_MODIFICADO/EVENTO4_Q800_q200_250.75_0.8mm"
    25/10/25 substituido pasta do arquivo

    // Obtém a função que chamou esta função (neste caso, main)
    _, filename, _, ok := runtime.Caller(0)
    if !ok {
        panic("Erro ao obter informações do arquivo")
    }

    // Obtém o diretório onde o arquivo está instalado
    dir := filepath.Dir(filename)
    fmt.Println("O diretório de instalação é:", dir)

    // nome do arquivo utilizado
    // filePath := narq + "/EVENTO4_Q800_q200_250.75_0.8mm.csv" - 25/10/25 arquivo inserido na pasta abaixo
    filePath := dir + "/dados_OpenFÔAM/EVENTO4_Q800_q200_250.75_0.8mm.csv"

    // Open the file
    file, err := os.Open(filePath)

```

```

if err != nil {
    fmt.Println("Error opening file:", err)
    return
}
defer file.Close()

// Create a new scanner to read the file line by line
scanner := bufio.NewScanner(file)

i=0
// Read each line
for scanner.Scan() {
    if i == 0 {
        scanner.Scan()
        fmt.Println ("Descartada a primeira linha")

        // descarta o cabeçalho do arquivo

        line := scanner.Text()
        //fmt.Println (line)

        // Split the line into individual values
        values := strings.Split(line, ",")

        // Convert each data for float64

/* velocidade_bolha.Leitura_vetor_parafoam_tdg.t, err = strconv.ParseFloat(values[0],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.X, err = strconv.ParseFloat(values[0],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Y, err = strconv.ParseFloat(values[1],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Z, err = strconv.ParseFloat(values[2],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_c , err = strconv.ParseFloat(values[3],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_mc , err = strconv.ParseFloat(values[4],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_c_ent , err = strconv.ParseFloat(values[5],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_c_Source , err =
strconv.ParseFloat(values[6],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_c_m_Source , err =
strconv.ParseFloat(values[7],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_Db , err = strconv.ParseFloat(values[8],64)
    if err != nil {

```

```

        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_n, err = strconv.ParseFloat(values[9],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_ubx, err = strconv.ParseFloat(values[10],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_uby, err = strconv.ParseFloat(values[11],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_ubz, err = strconv.ParseFloat(values[12],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_ubrx, err = strconv.ParseFloat(values[13],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_ubry, err = strconv.ParseFloat(values[14],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_ubrz, err = strconv.ParseFloat(values[15],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tgd_alpha_bolha, err =
    strconv.ParseFloat(values[16],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tgd_alpha_bolha_m, err =
    strconv.ParseFloat(values[17],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tgd_alpha_bolha_entr, err =
    strconv.ParseFloat(values[18],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tgd_alpha_bolha_source, err =
    strconv.ParseFloat(values[19],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Tdg_rhob, err = strconv.ParseFloat(values[20],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Ux, err = strconv.ParseFloat(values[21],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Uy, err = strconv.ParseFloat(values[22],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }

```

```

    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Uz, err = strconv.ParseFloat(values[23],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Alpha_water, err = strconv.ParseFloat(values[24],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.AlphaMean_water, err = strconv.ParseFloat(values[25],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.K_tdg, err = strconv.ParseFloat(values[26],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Nut, err = strconv.ParseFloat(values[27],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Omega, err = strconv.ParseFloat(values[28],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Pressao, err = strconv.ParseFloat(values[29],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Leitura_vetor_parafoam_tdg.Pressao_mean, err = strconv.ParseFloat(values[30],64)
    if err != nil {
        fmt.Println("Erro ao converter para float64:", err)
        return
    }
    velocidade_bolha.Dado_parafoam_tdg =
append(velocidade_bolha.Dado_parafoam_tdg,velocidade_bolha.Leitura_vetor_parafoam_tdg) // armazena os dados de leitura
em um array
    i = i+1

} // fim do for de leitura do arquivo

numero_dados_tdg = i

} // fim da função ler_arq_tdg

func media_desv () {
// Esta função tem a função de calcular a média e desvio padrão das velocidades

var i,j,a,b int
var coord_x, coord_z int
var diferenca float64
var num_dados float64
var arq *os.File
var err error

// tem a função de zerar o array
for i = 0; i<velocidade_bolha.Array_Comprimento_x; i++){
    for j = 0; j<velocidade_bolha.Array_Comprimento_z; j++){

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z =0
    }
}

```

```

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_x =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_y =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_z =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_x =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_y =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_z =0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alpha_water=0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alphaMean_water=0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao=0
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao_mean=0

        velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_x=0
        velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_y=0
        velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_z=0
        velocidade_bolha.Estatistica_bolha[i][j].Velocidade_x=0
        velocidade_bolha.Estatistica_bolha[i][j].Velocidade_y=0
        velocidade_bolha.Estatistica_bolha[i][j].Velocidade_z=0
        velocidade_bolha.Estatistica_bolha[i][j].Velocidade_inst_x=0
        velocidade_bolha.Estatistica_bolha[i][j].Velocidade_ux = 0
        velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_x_PED = 0

    }
}
b = int(len (velocidade_bolha.Dado_parafoam))
fmt.Printf ("O número total de dados lidos é %d\n",b)
a = 0
    // Faz o somatório dos valores para depois calcular a média
    for a=0; a < b; a++ {
        coord_x = int (velocidade_bolha.Dado_parafoam[a].X/ velocidade_bolha.Dimensao_malha_x)
        coord_z = int ((velocidade_bolha.Dado_parafoam[a].Z-
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z)
        if coord_x > 0 && coord_z >0 {

            // novo vetor descartar o de cima
            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Vel_media_x
+=velocidade_bolha.Dado_parafoam[a].Umx
            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Vel_media_y
+=velocidade_bolha.Dado_parafoam[a].Umy
            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Vel_media_z
+=velocidade_bolha.Dado_parafoam[a].Umz
            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Numero_dados_lido +=1
            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Media_flutuacao_x
+=velocidade_bolha.Dado_parafoam[a].Ux
            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Media_flutuacao_y
+=velocidade_bolha.Dado_parafoam[a].Uy
            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Media_flutuacao_z
+=velocidade_bolha.Dado_parafoam[a].Uz

            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Alpha_water+=velocidade_bolha.Dado_parafoam[a].
Alpha_water

            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].AlphaMean_water+=velocidade_bolha.Dado_parafo
am[a].AlphaMean_water

            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Pressao+=velocidade_bolha.Dado_parafoam[a].Pres
sao

            velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Pressao_mean+=velocidade_bolha.Dado_parafoam[
a].Pressao_mean
        }
    }

    // calcula a media e atribui código de falha caso o número de dados seja igual a zero
    for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
        for j = 0; j<=velocidade_bolha.Array_Comprimento_z;j++){

            num_dados = float64(velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido)
            // if num_dados == 0 {
            if num_dados == 0 &&
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].AlphaMean_water > 1 {

```

```

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_x = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_y = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_z = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_x
=999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_y
=999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_z
=999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alpha_water = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alphaMean_water = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao = 999999
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao_mean = 999999

    } else {
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x/num_dados
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y/num_dados
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z/num_dados
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x/num_dados
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y/num_dados
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z/num_dados

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water = velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha
a_water/num_dados

        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water = velocidade_bolha.Estatistica_dado_parafoam[i][j]
].AlphaMean_water/num_dados

        //velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao =
(float64(velocidade_bolha.Lim_na_jus) - float64(velocidade_bolha.Min_na_jus) -
float64(j) * velocidade_bolha.Dimensao_malha_z) * 9810 // teste
        //velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean =
(float64(velocidade_bolha.Lim_na_jus) - float64(velocidade_bolha.Min_na_jus) -
float64(j) * velocidade_bolha.Dimensao_malha_z) * 9810 // teste

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao = velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao/
num_dados

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean = velocidade_bolha.Estatistica_dado_parafoam[i][j].P
ressao_mean/num_dados
    }

}

fmt.Println("Terminou de calcular a média")

// calcula o desvio padrão
for a = 0; a < numero_dados; a++ {
    coord_x = int(velocidade_bolha.Dado_parafoam[a].X / velocidade_bolha.Dimensao_malha_x)
    coord_z = int((velocidade_bolha.Dado_parafoam[a].Z -
velocidade_bolha.Min_na_jus) / velocidade_bolha.Dimensao_malha_z)
    if coord_x >= 0 && coord_z >= 0 {

        if velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Numero_dados_lido == 0 {
        } else {

            diferenca = velocidade_bolha.Dado_parafoam[a].Umx -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Vel_media_x

```

```

        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_vel_media_x
+=diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Umy -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Vel_media_y
        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_vel_media_y
+=diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Umz -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Vel_media_z
        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_vel_media_z
+=diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Ux -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Media_flutuacao_x

        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_vel_media_flutuacao_x +=
diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Uy -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Media_flutuacao_y

        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_vel_media_flutuacao_y +=
diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Uz -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Media_flutuacao_z

        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_vel_media_flutuacao_z +=
diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Alpha_water -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Alpha_water
        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_alpha_water +=
diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].AlphaMean_water -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].AlphaMean_water
        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_alphaMean_water
+= diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Pressao -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Pressao
        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_pressao +=
diferenca*diferenca

        diferenca = velocidade_bolha.Dado_parafoam[a].Pressao_mean -
velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Pressao_mean
        velocidade_bolha.Estatistica_dado_parafoam[coord_x][coord_z].Desv_padrao_pressao_mean +=
diferenca*diferenca
    }
    } // fim do if ue verifica se a posição é negativa

}
fmt.Println("Terminou de calcular o desvio padrão")

for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
    for j = 0; j<=velocidade_bolha.Array_Comprimento_z;j++){

        num_dados = float64(velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido)
        if velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido == 0 {
        } else {

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_x
=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_x/num_dados)
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_y
=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_y/num_dados)
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_z
=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_z/num_dados)
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_x
=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_x/num_dados)
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_y
=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_y/num_dados)
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_z
=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_z/num_dados)

```

```

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alpha_water=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alpha_water/num_dados)

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alphaMean_water=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alphaMean_water/num_dados)

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao/num_dados)

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao_mean=math.Sqrt(velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao_mean/num_dados)
    }
} // fim do for j

} // fim do for i

// imprime os dados da função media_desvi

arq, err = os.Create("estatistica_grade.txt")

if err != nil {
    fmt.Println("erro parametros - ", err)
    return
}

// imprime o cabeçalho do arquivo

cabecalho := []string{
    "distância x", "Profundidade z", "vel_media_x", "vel_media_y", "vel_media_z", "numero_dados_lido",
    "media_flutuacao_x", "media_flutuacao_y", "media_flutuacao_z", "alpha_water", "alphaMean_water", "pressao",
    "pressao_mean", "desv_padrao_vel_media_x", "desv_padrao_vel_media_y", "desv_padrao_vel_media_z",
    "desv_padrao_vel_media_flutuacao_x", "desv_padrao_vel_media_flutuacao_y", "desv_padrao_vel_media_flutuacao_z",
    "desv_padrao_alpha_water", "desv_padrao_alphaMean_water", "desv_padrao_pressao", "desv_padrao_pressao_mean",
}

for _, header := range cabecalho {
    fmt.Fprintf(arq,"%40s", header)
}

fmt.Fprintf(arq,"\n")
for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
    Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
    for j = Fundo_bacia; j<=velocidade_bolha.Array_Comprimento_z;j++){

        fmt.Fprintf(arq,"\n %40.2f %40.2f %40.6f %40.6f %40.6f %40d %40.6f %40.6f %40.6f %40.6f",
        %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f",
        float64(i)*velocidade_bolha.Dimensao_malha_x,
        (float64(j)*velocidade_bolha.Dimensao_malha_z+velocidade_bolha.Min_na_jus),
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_x,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_y,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_z,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_x,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_y,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_z,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alpha_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alphaMean_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao_mean)
    }
}

arq.Close()

```

```

} //fim da função media_desv

func media_desv_TDG () {
    // Esta função tem a função de calcular a média e desvio padrão dos parâmetros dos dados lidos de TDG da saída do
    ParaFoam

    var i,j,a,b int
    var coord_x, coord_z int
    //var diferenca float64
    var num_dados float64
    var arq      *os.File
    var err      error

    // tem a função de zerar o array
    for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
        for j = 0; j<=velocidade_bolha.Array_Comprimento_z-1;j++){

            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Numero_dados_lido=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_mc=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_c_ent=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_c_Source=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_c_m_Source=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_Db=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_n=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubx=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_uby=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubz=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubrx=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubry=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubrz=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tgd_alpha_bolha=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tgd_alpha_bolha_m=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tgd_alpha_bolha_entr=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tgd_alpha_bolha_source=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_rhob=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Ux=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Uy=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Uz=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Alpha_water=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].AlphaMean_water=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].K_tdg=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Nut=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Omega=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Pressao=0
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Pressao_mean=0

            /*velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_x=0 Já é zerado na
            leitura inicial dos dados

            velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_y=0
            velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_z=0
            velocidade_bolha.Estatistica_bolha[i][j].Velocidade_x=0
            velocidade_bolha.Estatistica_bolha[i][j].Velocidade_y=0
            velocidade_bolha.Estatistica_bolha[i][j].Velocidade_z=0
            velocidade_bolha.Estatistica_bolha[i][j].Velocidade_inst_x=0
            velocidade_bolha.Estatistica_bolha[i][j].Velocidade_ux=0
            velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_x_PED=0 */

        }
    }

    b = int(len (velocidade_bolha.Dado_parafoam_tdg))
    fmt.Printf ("O número total de dados lidos de tdg é %d\n",b)
    a = 0

    // Faz o somatório dos valores para depois calcular a média
    for a=0; a < b; a++ {

        coord_x = int (velocidade_bolha.Dado_parafoam_tdg[a].X/velocidade_bolha.Dimensao_malha_x)
        coord_z = int ((velocidade_bolha.Dado_parafoam_tdg[a].Z -
        velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z)
        if coord_x > 0 && coord_z > 0 {
            velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Numero_dados_lido +=1
            velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_c
            +=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_c
            velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_mc
            +=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_mc
        }
    }
}

```

```

        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_c_ent
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_c_ent
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_c_Source
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_c_Source
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_c_m_Source
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_c_m_Source
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_Db
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_Db
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_n
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_n
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_ubx
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_ubx
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_uby
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_uby
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_ubz
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_ubz
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_ubrx
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_ubrx
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_ubry
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_ubry
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_ubrz
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_ubrz
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tgd_alpha_bolha
+=velocidade_bolha.Dado_parafoam_tdg[a].Tgd_alpha_bolha
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tgd_alpha_bolha_m
+=velocidade_bolha.Dado_parafoam_tdg[a].Tgd_alpha_bolha_m
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tgd_alpha_bolha_entr
+=velocidade_bolha.Dado_parafoam_tdg[a].Tgd_alpha_bolha_entr
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tgd_alpha_bolha_source
+=velocidade_bolha.Dado_parafoam_tdg[a].Tgd_alpha_bolha_source
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Tdg_rhob
+=velocidade_bolha.Dado_parafoam_tdg[a].Tdg_rhob
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Ux
+=velocidade_bolha.Dado_parafoam_tdg[a].Ux
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Uy
+=velocidade_bolha.Dado_parafoam_tdg[a].Uy
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Uz
+=velocidade_bolha.Dado_parafoam_tdg[a].Uz
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Alpha_water
+=velocidade_bolha.Dado_parafoam_tdg[a].Alpha_water
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].AlphaMean_water
+=velocidade_bolha.Dado_parafoam_tdg[a].AlphaMean_water
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].K_tdg
+=velocidade_bolha.Dado_parafoam_tdg[a].K_tdg
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Nut
+=velocidade_bolha.Dado_parafoam_tdg[a].Nut
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Omega
+=velocidade_bolha.Dado_parafoam_tdg[a].Omega
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Pressao
+=velocidade_bolha.Dado_parafoam_tdg[a].Pressao
        velocidade_bolha.Estatistica_dado_parafoam_tdg[coord_x][coord_z].Pressao_mean
+=velocidade_bolha.Dado_parafoam_tdg[a].Pressao_mean
    }
}

// calcula a media e atribui código de falha caso o número de dados seja igual a zero
for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
    for j = 0; j<=velocidade_bolha.Array_Comprimento_z;j++){

        num_dados =
float64(velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Numero_dados_lido)
        if num_dados == 0 {
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_c =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_mc =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_c_ent =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_c_Source
=999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_c_m_Source
=999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_Db =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_n =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubx =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_uby =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubz =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubrx =999999
            velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Tdg_ubry =999999

```

[illegible]

[illegible]

```

    arq.Close()

} //fim da função media_desv_TDG

// Função para calcular a distância euclidiana entre dois pontos
func euclideanDistance(x1, y1, x2, y2 float64) float64 {
    return math.Sqrt(float64((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1)))
} // fim da função euclideanDistance

// função para preencher todos os dados da malha x e z com todos os dados disponíveis utilizando o inverso da distância
func Inverso_distancia () {
    var i, j, m, n, num_dados_m_n int
    var soma_peso float64

    num_dados_m_n = 0

    // Utiliza todos os dados
    for i = 0; i < velocidade_bolha.Array_Comprimento_x; i++ {
        for j = 0; j <= int((velocidade_bolha.Array_Comprimento_z)); j++ {
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_x = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_y = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_z = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_x = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_y = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_z = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Alpha_water = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].AlphaMean_water = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao = 0
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao_mean = 0
        }
    }

    for i = 0; i < velocidade_bolha.Array_Comprimento_x; i++ {
        Fundo_bacia =
        int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
        velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
        //fmt.Printf ("Iniciando o preenchido o dado na célula inverso da distância %5d\n", i,)
        for j = Fundo_bacia; j <= int((velocidade_bolha.Array_Comprimento_z)); j++ {

            soma_peso = 0
            num_dados_m_n = 0

            for m = 0; m < velocidade_bolha.Array_Comprimento_x; m++ {
                Fundo_bacia =
                int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(m)*velocidade_bolha.Dimensao_malha_x) -
                velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
                for n = Fundo_bacia; n <= int((velocidade_bolha.Array_Comprimento_z)); n++ {

                    if
                    float64(velocidade_bolha.Estatistica_dado_parafoam[m][n].Numero_dados_lido) > 0 { // verifica se existe dado nessa célula
                    para fazer a interpolação

                        num_dados_m_n ++ // conta uando dados foram interpolados
                        dist:=
                        euclideanDistance(float64(i)*velocidade_bolha.Dimensao_malha_x, float64(j)*velocidade_bolha.Dimensao_malha_z,
                        float64(m)*velocidade_bolha.Dimensao_malha_x, float64(n)*velocidade_bolha.Dimensao_malha_z)

                        if dist > 0 {
                            peso:=1/math.Pow(dist, 2) // Ponderação pelo inverso da
                            distância

                            soma_peso += peso
                            //fmt.Printf ("Iniciando o preenchido o dado na
                            célula peso %5d %5d %5.8f\n", i, j, num_dados_m_n)

                            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_x
                            +=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_x

                            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_y
                            +=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_y

                            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_z
                            +=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_z

                            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_x
                            +=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_x

```

```

        velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_y
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_y

        velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_z
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_z

        velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Alpha_water+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Alpha_water

        velocidade_bolha.Estimar_malha_dado_parafoam[i][j].AlphaMean_water+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].AlphaMean_water

        velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Pressao

        velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao_mean+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Pressao_mean

    } // fim do if dist
} // fim do if Numero_dados_lido != 0

} //fim do for n
} //fim do for m

// verificado que mão havia dados nessa célula é calculado o valor interpolado para
esta célula vazia em relação a i, j
if soma_peso > 0 {
    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_x
=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_x/soma_peso
    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_y
=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_y/soma_peso
    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_z
=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_z/soma_peso
    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_x
=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_x/soma_peso
    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_y
=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_y/soma_peso
    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_z
=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_z/soma_peso

    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Alpha_water=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Alpha_water/soma_peso

    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].AlphaMean_water=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].AlphaMean_water/soma_peso

    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao/soma_peso

    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao_mean=velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao_mean/soma_peso

    velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Numero_dados_lido = 1

    //fmt.Printf ("Foi preenchido o dado na célula %5d %5d sendo interpolado
%5.2f com peso %5.5f\n", i, j, num_dados_m_n, soma_peso )
} // fim do if soma_peso > 0

} // fim do for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++
} //fim do for i=0; i<velocidade_bolha.Array_Comprimento_x; i++

} // fim da função func Inverso_distancia ()

// Essa função tem o objetivo de inserir os dados estimados para calcular o TDG
func Inverte_Dado () {
    var i, j, Fundo_bacia int

    for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
        Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
        for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++{

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
            velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_x

```

```

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_y
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Vel_media_z
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_x
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_y
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Media_flutuacao_z
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Alpha_water
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].AlphaMean_water
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao
                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean =
velocidade_bolha.Estimar_malha_dado_parafoam[i][j].Pressao_mean

                                }
        }
    } // fim da func Inverte_Dado ()

```

func Ajusteplano (x, z , c2 [6] float64) ([10] float64){ // entrada - distância x, y e variável a ser interpolada saída - coeficientes
 // função com o objetivo de preencher os vazios na malha utilizando o método de ajuste do plano por mínimos
 quadrados

```

    var vv [90]float64 // d - armazena a coordenada y; r - armazena a coordenada x; v - armazena o valor da célula
    var xx [3][90]float64
    var p int
    var a [10][10]float64
    var r [10] float64

    valores := func () {
        var i int
        p = 6

        for i=0; i<p; i++ {
            xx[0][i] = 1.0
            xx[1][i] = x[i]
            xx[2][i] = z[i]
            vv[i] = c2[i]
            //fmt.Printf("Iniciando o cálculo de v1 %10.5f %10.5f %10.5f %10d \n",x[1][i],x[2][i],y[i],p)
        }
    }

    } // fim da função func valores ()

```

```

    selgau := func (n int, aa [10][10]float64) ([10]float64) {
        //fmt.Printf("Inicio da função segau %5.5f\n",aa[1][1])

        // metodo de eliminacao de Gauss - pivotamento total

        var (i,j,k, n1,l,c int)
        var (max, aux,s float64)
        var x,y [10]float64
        var p [10]int

        n1 = n + 1

        for i=0; i<n; i++ {
            p[i] = i
        }

        for i=0; i<n; i++ {

            c = i
            l = i

            max = math.Abs(aa[i][i])

            for k=i; k<n; k++ {
                for j=i; j<n; j++ {

                    if max < math.Abs(aa[k][j]) {

```

```

        max = math.Abs(aa[k][j])
        c = j;
        l = k;
    }
}

if max < 1.0e-30 {
    return x;
}

if l != i {
    for j=i; j<n1; j++ {
        aa[i][j],aa[l][j] = aa[l][j],aa[i][j]
    }
}

if c != i {
    p[i],p[c] = p[c],p[i]
    for k=0; k<n; k++ {
        aa[k][i],aa[k][c] = aa[k][c],aa[k][i]
    }
}

for j=i+1; j<n; j++ {

    aux = aa[j][i]/aa[i][i]
    for k=i+1; k<n1; k++ {

        aa[j][k] -= aux*aa[i][k]

    }
}

if math.Abs(aa[i][i]) < 1.0e-30 {
    return x
}
}

for i=n-1; i>=0; i-- {
    s = aa[i][n];
    for j=i+1; j<n; j++ {
        s -= aa[i][j]*x[j]
    }
    x[i] = s/aa[i][i]
}

for i=0; i<n; i++ {
    y[p[i]] = x[i]
}

return y
} // fim da função func selgau(n int8, a [10][10]float64) (int8, [10]float64)

valores()

for i:=0; i<3; i++ {
    for j:=0; j<4; j++ {
        a[i][j] = 0.0
    }
}

for k:=0; k<p; k++ {
    for i:=0; i<3; i++ {
        a[i][3] += xx[i][k]*vv[k]
        for j:=0; j<3; j++ {
            a[i][j] += xx[i][k]*xx[j][k]
        }
    }
}

r = selgau(3,a)

/*for i:=0; i<p; i++ { // esse loop pode ser retirado
    v1:= 0.0
    for j:=0; j<3; j++ {

```

```

        v1 += r[j]*x[j][i]
    }
    v1 = math.Exp(v1)/c
}*/
//fmt.Printf("Iniciando o cálculo de v1 %10.5f %10.5f %10.5f\n",r[0],r[1],r[2])

/*for i:=0; i<6; i++{ // apenas testa
    fmt.Printf ("Finalizada a função AjustePlano %5d %5.6f %5.6f %5.6f %5.6f %5.6f %5.6f %5d \n", i, dt[i], rt[i], c2[i],
r[0], r[1], r[2], rc)
}*/

return r
} // fim da função func Ajusteplano ()

/* Função para preenchimento de falhas baseado no item 13.8 do Numerical Recipes-The Art of Scientific Computing 3rd
Edition (Press et al)
*/
/*func Lomb () {

    const TWOPI=6.283185307179586476
    var i, j int
    var ave,c,cc,cwtau,effm,expy,pnow,pymax,s,ss,sumc,sumcy,sums,sumsh,
sumsy,swtau, wtau,xave,xdif,xmax,xmin,yy,arg,wtemp float64

} */
// fim da função func Lomb ()

func preenche_falha () {
// Essa função tem como objetivo preencher as falhas nos dados da malha usando o método do inverso do quadrado
var soma_peso float64
var num_dados, num_dados_m_n float64
var i, j, m, n int
var arq *os.File
var err error
var numero_falha int = 0

for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
    //for j= 0; j<=int((float64(velocidade_bolha.Lim_na_jus)-
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z);j++{
        Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
        for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++{
            num_dados = float64(velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido)
            //fmt.Printf ("Iniciando o preenchido o dado na célula %5d %5d %5.2f/n", i, j, num_dados)
            num_dados_m_n = 0

            // inicia o teste e preenchimento de falhas - caso exista dados nas células vizinhas no eixo x é
            // preenchido pela média, caso não é usado o método do inverso da distância

            if i>0 && j>0 && num_dados ==0 && float64(velocidade_bolha.Estatistica_dado_parafoam[i-
1][j].Numero_dados_lido)>0 && float64(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Numero_dados_lido) > 0 {

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Vel_media_x+velocidade_bolha.Estatistica_dado_parafoam[i-
1][j].Vel_media_x)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Vel_media_y+velocidade_bolha.Estatistica_dado_parafoam[i-
1][j].Vel_media_y)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Vel_media_z+velocidade_bolha.Estatistica_dado_parafoam[i-
1][j].Vel_media_z)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Media_flutuacao_x+velocidade_bolha.Estatistica_dado_parafoam[i-
1][j].Media_flutuacao_x)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Media_flutuacao_y+velocidade_bolha.Estatistica_dado_parafoam[i-
1][j].Media_flutuacao_y)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Media_flutuacao_z+velocidade_bolha.Estatistica_dado_parafoam[i-
1][j].Media_flutuacao_z)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].
Alpha_water+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Alpha_water)/2

```

```

        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].AlphaMean_water+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].AlphaMean_water)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Pressao+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Pressao_mean+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao_mean)/2
        num_dados = 1
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1
    } // fim do if

    // continua o teste e preenchimento de falhas - caso exista dados nas células vizinhas no eixo z
    é preenchido pela média, caso não é usado o método do inverso da distância
    if i>0 && j>0 && j!= velocidade_bolha.Array_Comprimento_z && num_dados ==0 &&
float64(velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Numero_dados_lido)>0 &&
float64(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Numero_dados_lido) > 0 {

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Vel_media_x+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Vel_media_x)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Vel_media_y+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Vel_media_y)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Vel_media_z+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Vel_media_z)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Media_flutuacao_x+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Media_flutuacao_x)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Media_flutuacao_y+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Media_flutuacao_y)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Media_flutuacao_z+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Media_flutuacao_z)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Alpha_water+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Alpha_water)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].AlphaMean_water+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].AlphaMean_water)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Pressao+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Pressao)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=(velocidade_bolha.Estatistica_dado_parafoam[i][j+1].Pressao_mean+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Pressao_mean)/2
        num_dados = 1
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1
    } // fim do if

    if j== velocidade_bolha.Array_Comprimento_z && num_dados ==0 &&
float64(velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Numero_dados_lido)>0 &&
float64(velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido) > 0 {
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
(velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Vel_media_x)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Vel_media_y)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Vel_media_z)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Media_flutuacao_x)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Media_flutuacao_y)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Media_flutuacao_z)/2

```

```

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Alpha_water)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=(velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].AlphaMean_water)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Pressao)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=(velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean+velocidade_bolha.Estatistica_dado_parafoam[i][j-1].Pressao_mean)/2
        num_dados = 1
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1
    } // fim do if

    } // fim j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++{
} // fim for i=0; i<velocidade_bolha.Array_Comprimento_x; i++

// O Objetivo de rodar o for i e j 2 vezes é fazer que seja aproveitada uma interpolação em "j" em x-1 e reduzir os pontos interpolados mais abruptos
for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
    //for j= 0; j<=int((float64(velocidade_bolha.Lim_na_jus)-
    velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z);j++{
        Fundo_bacia =
        int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
        velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
        for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++{
            num_dados = float64(velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido)
            //fmt.Printf ("Iniciando o preenchimento o dado na célula %5d %5d %5.2f/n", i, j,
num_dados)

            num_dados_m_n = 0

            // inicia o teste e preenchimento de falhas - caso exista dados nas células vizinhas no
            eixo x é preenchido pela média, caso não é usado o método do inverso da distância

            if i>0 && j>0 && num_dados ==0 && float64(velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Numero_dados_lido)>0 && float64(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Numero_dados_lido) > 0 {

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
                (velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Vel_media_x+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_x)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
                =(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Vel_media_y+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_y)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
                =(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Vel_media_z+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_z)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
                =(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Media_flutuacao_x+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_x)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
                =(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Media_flutuacao_y+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_y)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
                =(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Media_flutuacao_z+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_z)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Alpha_water+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Alpha_water)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].AlphaMean_water+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].AlphaMean_water)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Pressao+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao)/2

                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=(velocidade_bolha.Estatistica_dado_parafoam[i+1][j].Pressao_mean+velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao_mean)/2
                num_dados = 1
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1
            } // fim do if

            // continua o teste e preenchimento de falhas - caso exista dados nas células vizinhas
            no eixo z é preenchido pela média, caso não é usado o método do inverso da distância

```

[illegible]

```

if num_dados == 0

Inserir a função preenche falha

Ajusteplano () func Ajusteplano (dt, rt, c2 [6] float64) ([3] float64) {

*/

if num_dados == 0 {
    var dt, rt [6] float64 // representa a variavel x e y
    var c2[6][10]float64 // representa os 6 valores mais próximos, 10 - as 10 variáveis à
serem interpoladas

    var c1 [6] float64
    var conta_dado int = 0
    //var marcador_z int = 0
    var marcador_zz int = 0
    var marcador_x int = 0
    var Resultado_Ajuste float64 = 0
    var conta_falha int = 0

    for ii:=0; ii<6; ii++ { // zera as células
        for jj:=0; jj<10; jj++ {
            c2[ii][jj] = 0.0
        }

        c1[ii] = 0.0
    }

    //marcador_z = j
    marcador_zz = 0

    // utiliza os dados na mesma elevação da bacia e de uma profundidade acima e abaixo,
e de 5 células à montante e a jusante
    for n= 1; n<=5; n++ { // verifica os 2 pontos em y mais próximos

        if n== 2 {
            if j+1 < velocidade_bolha.Array_Comprimento_z {marcador_zz = 1}
else {marcador_zz = -1} // procura linha acima
        }
        if n== 3 {
            if marcador_zz == 1 {
                if j-1> Fundo_bacia {marcador_zz = -1} else{marcador_zz
=2} // será 2 de chegou no fundo da bacia
            } else {marcador_zz=-2} // marcador_zz = -2 - limite
        }
        if n==4 {
            if marcador_zz ==-1 {
                if j+2 <=velocidade_bolha.Array_Comprimento_z
{marcador_zz = 2} else {
                    marcador_zz =-2}
            }
            if marcador_zz ==-2 {marcador_zz = -3 }
            if marcador_zz ==2 {marcador_zz = 3}
        }
        if n == 5{
            if marcador_zz == 3 {marcador_zz = 4}
            if marcador_zz == -3 {marcador_zz = -4}
            if marcador_zz == 2 {
                if j-2> Fundo_bacia {marcador_zz = -2} else
{marcador_zz=3}
            }
            if marcador_zz ==-2 {marcador_zz=-3}
        }

        marcador_x=0
        for m=0; m <= 7; m++ { // verifica os 10 pontos em x mais próximos
            if m==1 {
                if j+1 <= velocidade_bolha.Array_Comprimento_x
{marcador_x=1} else {marcador_x=-1}
            }
            if m==2 {
                if marcador_x<0 {marcador_x=marcador_x-1}
                if marcador_x==1 {
                    if j-1 > 1 {marcador_x=-1} else {marcador_x=2}
                }
            }
        }
    }
}

```

```

    }
    if m==3 {
        if marcador_x== -1 {
            if j+2 <=
velocidade_bolha.Array_Comprimento_x {marcador_x=2} else {marcador_x=marcador_x-1}
        }
    }
    if m==4 {
        if marcador_x==2 {
            if j-2 > 1 {marcador_x=-2} else{marcador_x=3}
        } else {marcador_x=marcador_x-1}
    }
    if m==5 {
        if marcador_x>=3 {marcador_x=marcador_x+1}
        if marcador_x== -2 {
            if j+3 <=
velocidade_bolha.Array_Comprimento_x {marcador_x=3} else {marcador_x=marcador_x-1}
        }
        if marcador_x < -2 {marcador_x=marcador_x-1}
    }
    if m==6 {
        if marcador_x == 3 {
            if j-3 > 1 {marcador_x=-3} else{marcador_x=4}
        } else {marcador_x=marcador_x-1}
    }
    if m==7 {
        if marcador_x== -3 {
            if j+4 <=
velocidade_bolha.Array_Comprimento_x {marcador_x=4} else {marcador_x=marcador_x-1}
        } else {marcador_x=marcador_x-1}
        if j+4 <= velocidade_bolha.Array_Comprimento_x
{marcador_x=4}
    }
    if m==8 {
        if marcador_x == 4 {
            if j-4 > 1 {marcador_x=-4} else{marcador_x=5}
        } else {marcador_x=marcador_x-1}
    }
    /*if m==9 {
        if j+5 <= velocidade_bolha.Array_Comprimento_x
{marcador_x=5}
    }
    if m==10 {
        if j-5 > 1 {marcador_x=-5}
    } */

    if conta_dado <=5 {
        if
float64(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Numero_dados_lido)>0 &&
        !math.IsNaN(
velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Vel_media_x ) &&

        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Vel_media_y) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Vel_media_z) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Media_flutuacao_x) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Media_flutuacao_y) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Media_flutuacao_z) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Alpha_water) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].AlphaMean_water) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Pressao) &&
        !math.IsNaN(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Pressao_mean) &&

        float64(velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Vel_media_x)<50 {

```

```

float64(i+marcador_x)*velocidade_bolha.Dimensao_malha_x
+marcador_zz)*velocidade_bolha.Dimensao_malha_z

[0]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Vel_media_x
[1]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Vel_media_y
[2]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Vel_media_z
[3]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Media_flutuacao_x
[4]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Media_flutuacao_y
[5]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Media_flutuacao_z
[6]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Alpha_water
[7]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].AlphaMean_water
[8]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Pressao
[9]=velocidade_bolha.Estatistica_dado_parafoam[i+marcador_x][j +marcador_zz].Pressao_mean
    conta_dado++
} // fim do if verifica se dados estão ok

} // fim do for conta_dado:=0; conta_dado<=5; conta_dado ++

if conta_dado>5 {
    for ii:=0; ii<10; ii++ {
        for kk:=0;kk<6; kk++{
            c1 [kk]= c2[kk][ii]
        }
        parametros_plano := Ajusteplano (dt, rt , c1) //
        log_x:=
        log_z:=

        Resultado_Ajuste =
        (parametros_plano[0]+parametros_plano[1]*log_x+parametros_plano[2]*log_z)
        //Resultado_Ajuste = Resultado_Ajuste
        /math.Sqrt(0.019614*float64(i)*velocidade_bolha.Dimensao_malha_x*float64(j)*velocidade_bolha.Dimensao_malha_z)

        if !math.IsNaN(Resultado_Ajuste) &&
        Resultado_Ajuste < math.Abs(10*(velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_x)) &&
        !math.IsInf(Resultado_Ajuste,0) {

            num_dados = 1

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao = Resultado_Ajuste }
            {velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean = Resultado_Ajuste }
        } else {
            num_dados = 0

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 0

```

```

//fmt.Printf ("falha encontrada em %d
%d\n", i, j)

conta_falha ++
}

if ii == 1 {
//fmt.Printf ("Iniciando o preenchido o
dado na célula %5d %5d marcador zz = %5d m = %5d %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f\n", i, j, marcador_zz, m, c1
[0], c1 [1], c1 [2], c1 [3], c1 [4], c1 [5], velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x)
}
}

n = 6
m = marcador_x+10

//fmt.Printf ("Iniciando o preenchido o dado na célula %5d
%5d %5.2f %5.2f %5.2f %5.2f %5.2f %5.2f %5.2f\n", i, j, c2 [0] [0], c2 [1] [0], c2 [2] [0], c2 [3] [0], c2 [4] [0], c2 [5] [0],
velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x)

} // fim do if conta_dado>5 {
} // fim do for m

} // fim do for n

if conta_falha > 0 {
velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 0
//fmt.Printf ("falha encontrada em %d %d\n", i, j)
num_dados = 0
}

} // fim do if que aciona a Ajusteplano ()
// utiliza os dados na mesma elevação da bacia e de uma profundidade acima e abaixo, e de 5
células à montante e a jusante

if num_dados == 0 { // verifica se nessa célula não existe dados e preenche usando o método do
inverso do quadrado da distância
var marcador_z int = 0
var marcador_x int = 0
if i < 5 {marcador_x=0}
if i >= 5 && j< velocidade_bolha.Array_Comprimento_x-5 {marcador_x=i-5}
if i > velocidade_bolha.Array_Comprimento_x-5 {marcador_x=i-10} // verifica se não é a primeira
célula

soma_peso = 0

//for m=0; m<velocidade_bolha.Array_Comprimento_x; m++ {
for m=marcador_x; m <= marcador_x+10; m++ {
Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(m)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
if j == 0 {marcador_z=0}
if j >= 1 && j< velocidade_bolha.Array_Comprimento_z {marcador_z=j-1}
if j == velocidade_bolha.Array_Comprimento_z {marcador_z=j-2} // verifica se
não é a primeira célula

for n= marcador_z; n<=marcador_z+2;n++{
//for n=0 ; n<=int((float64(velocidade_bolha.Lim_na_jus)-
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z);n++{
//num_dados_m_n =
float64(velocidade_bolha.Estatistica_dado_parafoam[m][n].Numero_dados_lido)

if
float64(velocidade_bolha.Estatistica_dado_parafoam[m][n].Numero_dados_lido) > 0 { // verifica se existe dado nessa célula
para fazer a interpolação

//fmt.Printf ("Iniciando o preenchido o dado na célula %5d
%5d %5.2f\n", i, j, num_dados)

num_dados_m_n =num_dados_m_n +1 // conta uando
dados foram interpolados

```

```

                                dist:=
euclideanDistance(float64(i)*velocidade_bolha.Dimensao_malha_x, float64(j)*velocidade_bolha.Dimensao_malha_z,
float64(m)*velocidade_bolha.Dimensao_malha_x, float64(n)*velocidade_bolha.Dimensao_malha_z)
                                if dist > 0 {
inverso da distância                                peso:=1/math.Pow(dist, 2) // Ponderação pelo

                                soma_peso += peso
                                //fmt.Printf ("Iniciando o preenchido o dado na
célula peso %5d %5d %5.8f\n", i, j, num_dados_m_n)                                if num_dados_m_n == 1{ // verifica se é o
primeiro dado

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x
=velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_x

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_y

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_z

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_x

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_y

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_z

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=velocidade_bolha.Estatistica_dado_parafoam[m
][n].Alpha_water

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=velocidade_bolha.Estatistica_dado_parafo
am[m][n].AlphaMean_water

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=velocidade_bolha.Estatistica_dado_parafoam[m][n].
Pressao

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=velocidade_bolha.Estatistica_dado_parafoam
[m][n].Pressao_mean

                                } else {

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x
+=velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_x

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
+=velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_y

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
+=velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_z

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
+=velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_x

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
+=velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_y

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
+=velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_z

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water+=velocidade_bolha.Estatistica_dado_parafoam[
m][n].Alpha_water

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water+=velocidade_bolha.Estatistica_dado_parafo
oam[m][n].AlphaMean_water

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao+=velocidade_bolha.Estatistica_dado_parafoam[m][n]
.Pressao

                                velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean+=velocidade_bolha.Estatistica_dado_parafoa
m[m][n].Pressao_mean

                                } // fim do else que testa se está com erro
                                } // fim do if dist

```

```

    } // fim do if Numero_dados_lido != 0

    } //fim do for n
} //fim do for m

// verificado que não havia dados nessa célula é calculado o valor interpolado para esta
célula vazia em relação a i, j

    if soma_peso > 0 {
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x/soma_peso
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y/soma_peso
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z/soma_peso
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x/soma_peso
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y/soma_peso
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z/soma_peso

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=velocidade_bolha.Estatistica_dado_parafoam[i][j].Alph
a_water/soma_peso

        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=velocidade_bolha.Estatistica_dado_parafoam[i][j
].AlphaMean_water/soma_peso

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao/
soma_peso

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=velocidade_bolha.Estatistica_dado_parafoam[i][j].P
ressao_mean/soma_peso

        //fmt.Printf ("Foi preenchido o dado na célula %5d %5d sendo interpolado
%5.2f com peso %5.5f\n", i, j, num_dados_m_n, soma_peso )

        num_dados = 1
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1
    } // fim do if soma_peso > 0
} //if num_dados == 0

// Utiliza todos os dados
if num_dados == 0 { // verifica se nessa célula não existe dados e preenche usando o método do
inverso do quadrado da distância

    soma_peso = 0
    numero_falha +=1

    for m=0; m<velocidade_bolha.Array_Comprimento_x; m++ {
        Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(m)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
        for n= Fundo_bacia; n<=int((velocidade_bolha.Array_Comprimento_z));n++{
            //for n=0 ; n<=int((float64(velocidade_bolha.Lim_na_jus)-
            velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z);n++{
                //num_dados_m_n =
float64(velocidade_bolha.Estatistica_dado_parafoam[m][n].Numero_dados_lido)

                if
float64(velocidade_bolha.Estatistica_dado_parafoam[m][n].Numero_dados_lido) > 0 { // verifica se existe dado nessa célula
para fazer a interpolação

                    //fmt.Printf ("Iniciando o preenchido o dado na
célula %5d %5d %5.2f\n", i, j, num_dados)

                    num_dados_m_n =num_dados_m_n +1 // conta uando
dados foram interpolados

                    dist:=
euclideanDistance(float64(i)*velocidade_bolha.Dimensao_malha_x, float64(j)*velocidade_bolha.Dimensao_malha_z,
float64(m)*velocidade_bolha.Dimensao_malha_x, float64(n)*velocidade_bolha.Dimensao_malha_z)
                    if dist > 0 {
                        peso:=1/math.Pow(dist, 2) // Ponderação pelo
inverso da distância

                        soma_peso += peso
                        //fmt.Printf ("Iniciando o preenchido o
dado na célula peso %5d %5d %5.8f\n", i, j, num_dados_m_n)

```

```

if num_dados_m_n == 1 { // verifica se é o
primeiro dado

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x
=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_x

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_y

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_z

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_x

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_y

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_z

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Alpha_water

    velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].AlphaMean_water

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Pressao

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Pressao_mean

} else {

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_x

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_y

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Vel_media_z

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_x

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_y

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Media_flutuacao_z

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Alpha_water

    velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].AlphaMean_water

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Pressao

    velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean+=peso*velocidade_bolha.Estatistica_dado_parafoam[m][n].Pressao_mean

} // if num_dados_m_n == 1 fim do

else que testa se está com erro

} // fim do if dist
} // fim do if Numero_dados_lido != 0
} //fim do for n
} //fim do for m

// verificado que mão havia dados nessa célula é calculado o valor
interpolado para esta célula vazia em relação a i, j

```

```

        if soma_peso > 0 {
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x/soma_peso
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y/soma_peso
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z/soma_peso
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x/soma_peso
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y/soma_peso
            velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z/soma_peso

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water/soma_peso

            velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water/soma_peso

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao/soma_peso

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean/soma_peso

            velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1

            //fmt.Printf ("Foi preenchido o dado na célula %5d %5d sendo
interpolado %5.2f com peso %5.5f\n", i, j, num_dados_m_n, soma_peso )
        } // fim do if soma_peso > 0

    } // fim do if [i][j].Numero_dados_lido = 0

    //fmt.Printf ("Rodando %5d %5d %5.2f\n", i, j, num_dados)
    //fmt.Printf ("Foi preenchido o dado na célula %5d %5d sendo a veloc média x %5.2f com pressão
%5.5f\n", i, j, velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x,
velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao )
    } // fim do for j

    // preenche o último dado de x

} // fim do for i

// Verifica se existe falha no último dado no eixo x

for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++){
    i = velocidade_bolha.Array_Comprimento_x
    num_dados = float64(velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido)

    if num_dados ==0 {
        //fmt.Printf ("Foi preenchido o dado na célula %5d %5d sendo interpolado %5.2f com peso
%5.5f\n", i, j, velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x,
velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water )

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_x + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_x-velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Vel_media_x)/2
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_y + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_y-velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Vel_media_y)/2
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_z + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Vel_media_z-velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Vel_media_z)/2
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x =
velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_x + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_x+velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Media_flutuacao_x)/2
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y
=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_y + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_y+velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Media_flutuacao_y)/2
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z
=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_z + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Media_flutuacao_z+velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Media_flutuacao_z)/2
    }
}

```

```

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Alpha_water + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Alpha_water+velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Alpha_water)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].AlphaMean_water + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].AlphaMean_water+velocidade_bolha.Estatistica_dado_parafoam[i-2][j].AlphaMean_water)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao+velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Pressao)/2

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean=velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao_mean + (velocidade_bolha.Estatistica_dado_parafoam[i-1][j].Pressao_mean+velocidade_bolha.Estatistica_dado_parafoam[i-2][j].Pressao_mean)/2
        num_dados = 1
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido = 1
    } // fim if i== velocidade_bolha.Array_Comprimento_x
} // Fim do for j

// verifica se os valores estão dentro dos limites definidos

for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
    //for j= 0; j<=int((float64(velocidade_bolha.Lim_na_jus)-
    velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z);j++{
        Fundo_bacia =
        int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
        velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
        for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++{

            if math.Abs(velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x) >
            velocidade_bolha.Vel_maxima {
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x =
            math.Copysign(velocidade_bolha.Vel_maxima, velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x)
            }
            if math.Abs(velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y) >
            velocidade_bolha.Vel_maxima {
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y =
            math.Copysign(velocidade_bolha.Vel_maxima, velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y)
            }
            if math.Abs(velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z) >
            velocidade_bolha.Vel_maxima {
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z =
            math.Copysign(velocidade_bolha.Vel_maxima, velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z)
            }
            if math.Abs(velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x) >
            velocidade_bolha.Vel_maxima {
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x =
            math.Copysign(velocidade_bolha.Vel_maxima, velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x)
            }
            if math.Abs(velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y) >
            velocidade_bolha.Vel_maxima {
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y =
            math.Copysign(velocidade_bolha.Vel_maxima, velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y)
            }
            if math.Abs(velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z) >
            velocidade_bolha.Vel_maxima {
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z =
            math.Copysign(velocidade_bolha.Vel_maxima, velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z)
            }
        }
    }
}

// imprime os dados da função preenche_falha no arquivo estatistica_grade_preenchido.txt
arq, err = os.Create("estatistica_grade_preenchido.txt")

if err != nil {
    fmt.Println("erro parametros - ", err)
    return
}

// imprime o cabeçalho do arquivo

```

```

cabecalho := []string{
    "distância x", "Profundidade z", "vel_media_x", "vel_media_y", "vel_media_z", "numero_dados_lido",
    "media_flutuacao_x", "media_flutuacao_y", "media_flutuacao_z", "alpha_water", "alphaMean_water", "pressao",
    "pressao_mean", "desv_padrao_vel_media_x", "desv_padrao_vel_media_y", "desv_padrao_vel_media_z",
    "desv_padrao_vel_media_flutuacao_x", "desv_padrao_vel_media_flutuacao_y", "desv_padrao_vel_media_flutuacao_z",
    "desv_padrao_alpha_water", "desv_padrao_alphaMean_water", "desv_padrao_pressao", "desv_padrao_pressao_mean",
}

for _, header := range cabecalho {
    fmt.Fprintf(arq,"%40s", header)
}

fmt.Fprintf(arq,"\n\n")
for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
    Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
    for j = Fundo_bacia; j<=velocidade_bolha.Array_Comprimento_z;j++){

        fmt.Fprintf(arq,"\n\n %40.2f %40.2f %40.6f %40.6f %40.6f %40d %40.6f %40.6f
%40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f",
float64(i)*velocidade_bolha.Dimensao_malha_x,
(float64(j)*velocidade_bolha.Dimensao_malha_z+velocidade_bolha.Min_na_jus),
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_x,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_y,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Media_flutuacao_z,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Alpha_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].AlphaMean_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Pressao_mean,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_x,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_y,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_z,

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_x,

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_y,

        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_vel_media_flutuacao_z,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alpha_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_alphaMean_water,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao,
        velocidade_bolha.Estatistica_dado_parafoam[i][j].Desv_padrao_pressao_mean)

    }

}

arq.Close()
fmt.Printf ("Fim da função de preenchimento de falhas na malha %5d\n", numero_falha)

} // fim da função preenche_falha

// Função de autocorrelação de velocidade

func rho_i(s, Ti float64) float64 {
    return math.Exp(-s / Ti)
}

// Função para integrar numericamente usando o método dos trapézios

func integrateRho(Ti float64, sMax float64, numSteps int) float64 {

    stepSize := sMax / float64(numSteps)
    integral := 0.0

    for i := 0; i < numSteps; i++ {
        s1 := float64(i) * stepSize
        s2 := float64(i+1) * stepSize
        integral += 0.5 * (rho_i(s1, Ti) + rho_i(s2, Ti)) * stepSize
    }
    //fmt.Printf ("Entrando na função Integrate Rho %5.2f %5.2f %d %5.2f/n", Ti, sMax, numSteps, integral) só para teste

```

```

    return integral
}

func eq_Langevin (pos_x, pos_z, n_dados int, v_fluido float64) {
/*movimento browniano seguindo a equação de Langevin foi utilizado o método de Euler para simular o movimento de uma
partícula sob o movimento browniano


$$m \cdot dv/dt = -\gamma \cdot v + \sqrt{(2 \cdot k \cdot T \cdot \gamma)} \cdot \eta(t)$$


Onde:
m: massa da partícula
v: velocidade da partícula
t: tempo
γ: coeficiente de atrito
k: Constante de Boltzmann
T: temperatura do fluido
η(t): Ruído branco gaussiano com média zero e variância unitária.

*/

/*

Prandtl's Method (for turbulent flow):
 $\delta \approx Re_x^{0.37} \cdot x$ 
Where:
•  $\delta$  is the boundary layer thickness.
•  $x$  is the distance along the plate from the leading edge.
•  $Re_x$  is the Reynolds number based on the distance  $x$ .

*/

const (
    m = 1.0 // mass of the particle
    γ = 0.1 // friction coefficient
    k = 1.0 // Boltzmann constant
    T = 300.0 // temperature of the fluid (Kelvin)
    dt = 0.01 // time step
)
// constantes do artigo de Valero, 2019
const (
    sigma = 0.5
    Dx = 2.22 //Valero et al (2019)
    Dy = 1.11 //Valero et al (2019)
    Dz = 1.35 //Valero et al (2019)
    Kx = 0.84 //Valero et al (2019)
    Ky = 0.66 //Valero et al (2019)
    Kz = 0.69 //Valero et al (2019)
    Delta_x = 1 // a malha utilizada é de 1 x 1 m
    Delta_y = 1 // a malha utilizada é de 1 x 1 m
    y = 0.0005 // Ly = 0.5 * Lx (Valero et al 2019)
    delta = 0.3
    pagua = 1000 // densidade da água 1000 kg/m3
    visc_cine = 0.001002 // viscosidade cinemática da água mPa*s a 20°C
    Ks = 0.0015 // ks the sand roughness coefficient 1.5 mm em Valero 2016 = 1.0 mm
    von_Karman = 0.41 // constante de von Kármán Valero et al (2019)
    B_canal_turbulento = 8.5 // constante do fluxo turbulento Valero et al (2019)
    pi_vertedouro = 0.2 // parâmetro de escoamento no vertedouro Valero et al (2019)
    sMax = 100 // utilizada na integração de Ti
)

//var v,x,
var exponentialTerm, Gaussian float64 //Reynolds (variável não usada)
var Cf,Hs, Ti, integral_Ti, delta_Ti float64 // profundidade (variável não usada)
var numSteps int
//var i, j int

//fmt.Println("dados de entrada para cálculo da estatística da bolha - %d %d %d %.2f\n ", pos_x, pos_z, n_dados , v_fluido)
// Seed the random number generator
rand.Seed(time.Now().UnixNano())
Gaussian = rand.NormFloat64() // era o número gaussiano para utilizar na equação de Valero, 2019

// Number of time steps to simulate
numSteps = 1000

```

```

//Ti = 0 // apenas para inicializar

/*
// Initial velocity and position of the particle
v = 0.0
x = 0.0

// Number of time steps to simulate
numSteps = 1000

for step := 0; step < numSteps; step++ {
    // Compute the random noise
    η := rand.NormFloat64()

    // Update the velocity and position using the Euler method
    v = v + (-γ * v) + math.Sqrt(2*k*T*γ)*η/m*dt
    x = x + v*dt
}

*/

//fmt.Println("Movimento aleatório da bolha, velocidade = ", v, " x = ", x, " γ = ", γ, " k = ", k, " T = ", T, " m = ", m)
//profundidade = (elv_na_jusante-float64(pos_z))

if n_dados == 0 {
} else {
    Hs = elv_reservatorio-elv_na_jusante

    //v_fluido := velocidade_bolha.Estatistica_dado_parafoam[pos_x][pos_z].vel_media_x // velocidade no eixo
x
viscosidade) //Reynolds = pagua * v_fluido * (profundidade)/visc_cine // Re (pagua x velocidade x profundidade /
Nelson //layer_thickness:=0.37*(profundidade)/(math.Sqrt(Reynolds)) // ver equação I-42 da apostila do prof.
layer_thickness:= float64(pos_x)*velocidade_bolha.Dimensao_malha_x * (0.0212 *
math.Pow(float64(pos_x)*velocidade_bolha.Dimensao_malha_x/Hs,0.11)*math.Pow(float64(pos_x)*velocidade_bolha.Dimensao_malha_x/Ks,-0.10)) // eq I-42 apostila do prof. Nelson, mesma que Valero 2016
Cf = 3.967*math.Log10(layer_thickness/Ks) + 6.7 // Equação 8
Cf = 1/(Cf * Cf)
vel_cis := v_fluido * math.Sqrt (Cf/2) // Equação 9

exponentialTerm = math.Exp(-Kx * 0.5 / layer_thickness)
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_x = (Dx * exponentialTerm) *
(vel_cis) // Equação 10 para x

exponentialTerm = math.Exp(-Kz * 0.5 / layer_thickness)
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_y = (Dy * exponentialTerm) *
(vel_cis) // Equação 10 para y

exponentialTerm = math.Exp(-Kz * 0.5 / layer_thickness)
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_z = (Dz * exponentialTerm) *
(vel_cis) // Equação 10 para z

// velocidade_bolha.Estatistica_bolha[pos_x][pos_z].velocidade_x = vel_cis *
((1/von_Karman)*math.Log(profundidade/Ks)+B_canal_turbulento+(1/von_Karman)*(1+6*pi_vertedouro)*math.Pow(profundad
e/layer_thickness,2)-(1/von_Karman)*(1+4*pi_vertedouro)*math.Pow(profundidade/layer_thickness,3))
// velocidade_bolha.Estatistica_bolha[pos_x][pos_z].velocidade_x =
1/von_Karman*math.Log(profundidade)+B_canal_turbulento+2*pi_vertedouro*math.Pow(math.Sin((math.Pi*profundidade)/(2*la
yer_thickness)), 2) // equação 2 Johnson 2016
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_x = v_fluido // As equações acima calculam a
velocidade do escoamento, como são conhecidas, melhor utilizar diretamente

//fmt.Printf("O valor de HS é %5.2f\n", Hs) apenas para teste
//fmt.Printf("O valor da posição x e z é %d %d\n", pos_x, pos_z) apenas para teste

if pos_x == 0 { //Ti = 0 {
    velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_ux =
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_x
} else {

    Ti = velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_x
    delta_Ti = Ti/float64(numSteps)
    //fmt.Printf("O valor da posição x e z é %d %d %d %5.2f %d\n", len(estadistica_bolha), len
(velocidade_bolha.Estatistica_bolha[0]), numSteps, Ti, sMax) apenas para teste

```

```

integral_Ti = integrateRho (Ti, sMax, numSteps) //calcula o valor de Ti utilizado na equação 3 de
Valero, 2019
Ti = integral_Ti
// equação 5 de Valero, 2019
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_ux =
velocidade_bolha.Estatistica_bolha[pos_x-1][pos_z].Velocidade_ux - velocidade_bolha.Estatistica_bolha[pos_x-
1][pos_z].Velocidade_ux
*delta_Ti/Ti+math.Pow(2*math.Pow(velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_x,2)*delta_Ti/Ti,0.
5)*Gaussian
}
if math.IsNaN (velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_ux) {
    velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_ux =
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_x
}
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_inst_x =
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_x +
velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Velocidade_ux
//fmt.Println (" pos_x = ", pos_x, " pos_z = ", pos_z, "profundidade ", profundidade, "Vel flut_x=
", velocidade_bolha.Estatistica_bolha[pos_x][pos_z].flutuacao_velocidade_x, "Vel flut_z=
", velocidade_bolha.Estatistica_bolha[pos_x][pos_z].flutuacao_velocidade_z, "velocidade_ux =
", velocidade_bolha.Estatistica_bolha[pos_x][pos_z].velocidade_ux, "v_fluido = ", v_fluido, " Reynolds = ", Reynolds, "
layer_thickness = ", layer_thickness, " Cf = ", Cf, " velocidade de cisalhamento = ", vel_cis, " exponentialTerm = ",
exponentialTerm, "velocidade instantânea em ", velocidade_bolha.Estatistica_bolha[pos_x][pos_z].velocidade_x, "variável
Gaussina", Gaussian)
if pos_x == 0 {
    velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_x_PED =
velocidade_bolha.Calcula_Ubrx (velocidade_bolha.Estatistica_dado_parafoam [pos_x][pos_z].Pressao,
velocidade_bolha.Estatistica_dado_parafoam [pos_x][pos_z].Pressao, velocidade_bolha.Estatistica_dado_parafoam
[pos_x+2][pos_z].Pressao) // usado 2 como condição de contorno
} else{
    velocidade_bolha.Estatistica_bolha[pos_x][pos_z].Flutuacao_velocidade_x_PED =
velocidade_bolha.Calcula_Ubrx (velocidade_bolha.Estatistica_dado_parafoam [pos_x-1][pos_z].Pressao,
velocidade_bolha.Estatistica_dado_parafoam [pos_x][pos_z].Pressao, velocidade_bolha.Estatistica_dado_parafoam
[pos_x+1][pos_z].Pressao)
}
}

} // fim da função func eq_Langevin

func main () {
    var i, j int
    var arq_bolha *os.File
    //var nx, ny int
    //var texto string
    //var concentracao_tdg, ent_vel_media_x, ent_vel_media_y, ent_alphaMean_water, ent_vel_inst_x, ent_pressao_mean []
float64 // carrega os dados da resolução da equação 40

    fmt.Printf("O número de i e j é %d %d\n", len(velocidade_bolha.Estatistica_dado_parafoam),
len(velocidade_bolha.Estatistica_dado_parafoam [0]))

    numCPUs := runtime.NumCPU()
    runtime.GOMAXPROCS(numCPUs)

    ler_arq_aci () // lê o arquivo de saída do Parafoam para os dados de escoamento
    media_desv () // calcula a média e o desvio padrão da velocidade
    //Inverso_distancia () // constroi a malha x e z com todos os dados disponíveis do arquivo ler_arq_aci

    preenche_falha () // preenche os dados vazios na área de interesse
    //Inverte_Dado () // tem a função de atribuir os dados da malha preenchida na malha do ler_arq_aci para calcular TDG
suavizado

    ler_arq_tdg () // lê o arquivo de saída do Parafoam para os dados de TDG - Desativado pra economizar processamento
    media_desv_TDG () // calcula a média dos dados de TDG - Desativado pra economizar processamento

    // essas duas linhas servem para preencher os dados e usar o hidrodinâmico do TDG 800 m³/s
    //velocidade_bolha.Atribui_dado_parafoam_TDG () // converte os dados hidrodinâmicos do TDG para o parafoam e aproveitas
o Q
    //preenche_falha () // preenche os dados vazios na área de interesse

    fmt.Printf("INICIO DO CALCULO DA VELOCIDADE DA BOLHA UTILIZANDO A EQUAÇÃO DE VALERO")
    // calcula a velocidade da bolha usando a equação de Valero (2019) - arquivo Etapa5
    for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {

```

```

        Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))
        for j = Fundo_bacia; j<=velocidade_bolha.Array_Comprimento_z;j++){
            eq_Langevin (i,j,velocidade_bolha.Estatistica_dado_parafoam[i][j].Numero_dados_lido,
velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x) // apenas exemplo de posição
        }
        //fmt.Printf("Rodou a linha %d de Langevin %d %d\n", i, j, lim_na_jus) //apenas para teste
    }
    fmt.Printf("FIM DO CALCULO DA VELOCIDADE DA BOLHA UTILIZANDO A EQUAÇÃO DE VALERO")

    // imprime os dados da estatística da bolha

    arq_bolha, err = os.Create("estatistica_bolha.txt")

    if err != nil {
        fmt.Println("erro parametros - ", err)
        return
    }

    // imprime o cabeçalho do arquivo

    cabecalho := []string{
        "distância x", "Profundidade z", "flutuação velocidade x", "flutuação velocidade y", "flutuação velocidade z", "velocidade
x", "velocidade ux", "velocidade_inst_x (bolha no fluxo)", "flutuacao_velocidade_x_PED",
    }

    for _, header := range cabecalho {
        fmt.Fprintf(arq_bolha,"%40s", header)
    }

    fmt.Fprintf(arq_bolha,"\n")

    for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
        Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))

        for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++){
            //for j = 0; j<=int((float64(velocidade_bolha.Lim_na_jus)-
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z);j++){
                fmt.Fprintf(arq_bolha,"\n %40.2f %40.2f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f ",
float64(i)*velocidade_bolha.Dimensao_malha_x,
(float64(j)*velocidade_bolha.Dimensao_malha_z+velocidade_bolha.Min_na_jus),
velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_x,velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_
y,velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_z,velocidade_bolha.Estatistica_bolha[i][j].Velocidade_x,velocid
ade_bolha.Estatistica_bolha[i][j].Velocidade_ux,velocidade_bolha.Estatistica_bolha[i][j].Velocidade_inst_x,
velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_x_PED)
            }
        }
        arq_bolha.Close()

        // calcula a velocidade da bolha usando a equação de Valero (2019) - arquivo Etapa6 - desativado porque usa a
etapa 5
        /*for i = 0; i<500; i++ {
            for j = 236; j<lim_na_jus; j++ {
                eq_Langevin (i,j,velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].numero_dados_lido,
velocidade_bolha.Estatistica_dado_parafoam_tdg[i][j].Ux)
            }
            if i % 50 == 0 {
                fmt.Printf("finalizado cálculo no ponto: %5d ", i)
            }
        }

    */

    // calculo da concentração de TDG (equação 40 e 41)

    fmt.Println("Concluída a atribuição de valores da equação 40")

    fmt.Println("Inicio da função velocidade_bolha")

    velocidade_bolha.Calcula_Campo_Concentracao ()

    for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
        Fundo_bacia =
int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))

```

```

        for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++){
            //for j = 0; j<=int((float64(velocidade_bolha.Lim_na_jus)-
            velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z);j++){
                if i % 50 == 0 {
                    fmt.Printf("finalizado cálculo no ponto: %5.2f %5.2f %5.2f\n",
                    float64(i)*velocidade_bolha.Dimensao_malha_x,
                    (float64(j)*velocidade_bolha.Dimensao_malha_z+velocidade_bolha.Min_na_jus), velocidade_bolha.Concentracao_tdg[i][j])
                }
            }
        }
        fmt.Println("Concluído o cálculo da concentração de TDG")

        for i=0; i<velocidade_bolha.Array_Comprimento_x; i++ {
            Fundo_bacia =
            int(math.Ceil((velocidade_bolha.Inicializa_fundo_bacia(float64(i)*velocidade_bolha.Dimensao_malha_x) -
            velocidade_bolha.Min_na_jus)/velocidade_bolha.Dimensao_malha_z))

            for j= Fundo_bacia; j<=int((velocidade_bolha.Array_Comprimento_z));j++){

                velocidade_bolha.Estatistica_dado_Automato[i][j].Vx_bolha =
                velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_x
                velocidade_bolha.Estatistica_dado_Automato[i][j].Vy_bolha =
                velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_y
                velocidade_bolha.Estatistica_dado_Automato[i][j].Vz_bolha =
                velocidade_bolha.Estatistica_bolha[i][j].Flutuacao_velocidade_z
                velocidade_bolha.Estatistica_dado_Automato[i][j].Vx =
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_x
                velocidade_bolha.Estatistica_dado_Automato[i][j].Vy =
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_y
                velocidade_bolha.Estatistica_dado_Automato[i][j].Vz =
                velocidade_bolha.Estatistica_dado_parafoam[i][j].Vel_media_z
            }

            velocidade_bolha.Fick_Difusao_Adveccao() // Calcula o TDG Utilizando Autômatos Celulares
        } // fim da função principal

```


APÊNDICE II – Modelo de autômato celular

Programa desenvolvido em Go para calcular o TDG utilizando o modelo de autômato celular.

```
package velocidade_bolha
/*
Declaração das variáveis das equações básicas de TDG e suas equações conforme apresentado nos relatórios do P&D TDG
*/
import (
    "fmt"
    "math"
    "os"
    //"math/rand"
    //"ler_U_OpenFOAM/main"
    "math/rand"
    "time"
)

type Estrutura_dado_parafoam struct {
    T float64 // tempo de simulação do dado
    X float64 // coordenada em x (m) - no comprimento do vertedouro
    Y float64 // coordenada em y (m) - na largura do vertedouro
    Z float64 // coordenda em z (m) - elevação do ponto
    Ux float64 // velocidade em m/s no ponto x
    Uy float64 // velocidade em m/s no ponto y
    Uz float64 // velocidade em m/s no ponto z
    Umx float64 // velocidade média em m/s no ponto x
    Umy float64 // velocidade média em m/s no ponto y
    Umz float64 // velocidade média em m/s no ponto Z
    Alpha_water float64 // leitura do alpha_water
    AlphaMean_water float64 // leitura da alpha_water médio
    Pressao float64 //pressão em bar
    Pressao_mean float64 // leitura da pressão média em bar
} // estrutura para leitura de todos os dados de saída de uma linha do OpenFoam para a etapa 5 - hidrodinâmico

type Estrutura_dado_parafoam_tdg struct {
    // t float64 // tempo de simulação do dado (não tem)
    Numero_dados_lido int // conta o número de dados lidos
    X float64 // coordenada em x (m) - no comprimento do vertedouro
    Y float64 // coordenada em y (m) - na largura do vertedouro
    Z float64 // coordenda em z (m) - elevação do ponto
    Tdg_c float64 // TDG instantâneo
    Tdg_mc float64 // TDG médio
    Tdg_c_ent float64 // TDG instantâneo na entrada
    Tdg_c_Source float64 // Fonte de TDG instantâneo
    Tdg_c_m_Source float64 // Fonte de TDG média
    Tdg_Db float64 // diâmetro da bolha
    Tdg_n float64 // milhões de bolha
    Tdg_ubx float64 // velocidade da bolha em x
    Tdg_uby float64 // velocidade da bolha em y
    Tdg_ubz float64 // velocidade da bolha em z
    Tdg_ubrx float64 // velocidade relativa da bolha em x
    Tdg_ubry float64 // velocidade relativa da bolha em y
    Tdg_ubrz float64 // velocidade relativa da bolha em z
    Tgd_alpha_bolha float64 // relação ar
    Tgd_alpha_bolha_m float64 // relação ar média
    Tgd_alpha_bolha_ent float64 // relação ar entr
    Tgd_alpha_bolha_source float64 // relação ar da fonte
    Tdg_rhob float64
    Ux float64 // velocidade do fluxo em m/s no ponto x
    Uy float64 // velocidade do fluxo em m/s no ponto y
    Uz float64 // velocidade do fluxo em m/s no ponto z
    Alpha_water float64 // leitura do alpha_water
    AlphaMean_water float64 // leitura da alpha_water médio
    K_tdg float64
}
```

```

    Nut float64
    Omega float64
    Pressao float64 // pressão em bar
    Pressao_mean float64 // leitura da pressão média em bar
} // estrutura para leitura de todos os dados de saída de uma linha do OpenFoam para a etapa 6 - TDG e Bolha

type Estrutura_estadistica_dados_lidos struct {
    Vel_media_x float64 // velocidade média no eixo x
    Vel_media_y float64 // velocidade média no eixo y
    Vel_media_z float64 // velocidade média no eixo z
    Numero_dados_lido int // número de dados lidos
    Media_flutuacao_x float64 // média da flutuação das velocidades o eixo x
    Media_flutuacao_y float64 // média da flutuação das velocidades o eixo y
    Media_flutuacao_z float64 // média da flutuação das velocidades o eixo z
    Alpha_water float64 // leitura do alpha_water
    AlphaMean_water float64 // leitura da alpha_water médio
    Pressao float64 // pressão em bar
    Pressao_mean float64 // leitura da pressão média em bar
    Desv_padrao_vel_media_x float64 // cálculo do desvio padrão da velocidade média no eixo x
    Desv_padrao_vel_media_y float64 // cálculo do desvio padrão da velocidade média no eixo y
    Desv_padrao_vel_media_z float64 // cálculo do desvio padrão da velocidade média no eixo z
    Desv_padrao_vel_media_flutuacao_x float64 // cálculo do desvio padrão da flutuação de velocidade no eixo x
    Desv_padrao_vel_media_flutuacao_y float64 // cálculo do desvio padrão da flutuação de velocidade no eixo y
    Desv_padrao_vel_media_flutuacao_z float64 // cálculo do desvio padrão da flutuação de velocidade no eixo z
    Desv_padrao_alpha_water float64 // leitura do alpha_water
    Desv_padrao_alphaMean_water float64 // leitura da alpha_water médio
    Desv_padrao_pressao float64 // pressão em bar
    Desv_padrao_pressao_mean float64 // leitura da pressão média em bar
} // variáveis para calcular as estatísticas dos dados lidos

// estrutura para calcular as características da bolha
type Estrutura_estadistica_bolha struct {
    Flutuacao_velocidade_x float64
    Flutuacao_velocidade_y float64
    Flutuacao_velocidade_z float64
    Velocidade_x float64
    Velocidade_y float64
    Velocidade_z float64
    Velocidade_ux float64
    Velocidade_inst_x float64
    Flutuacao_velocidade_x_PED float64
}

// estrutura dos dados para calcular o TDG utilizando autômato celulares
type Estrutura_dado_Automato struct {
    Vx_bolha, // velocidade relativa da bolha em x
    Vy_bolha, // velocidade relativa da bolha em y
    Vz_bolha, // velocidade relativa da bolha em z
    Vx, // velocidade do escoamento em x
    Vy, // velocidade do escoamento em y
    Vz, // velocidade do escoamento em z
    Pressao, // pressão
    Concentracao_O2, // concentração de o2 ma mistura = 36,8% fração molar do O2 no dissolvido no ar
    Concentracao_O2_percentual, // concentração de o2 ma mistura = 36,8% fração molar do O2 no dissolvido no ar em % de
    superdaturação
    Concentracao_N2, // concentração de N2 ma mistura = 63,2 % do ar em massa molar
    Concentracao_H2O,
    fra_molar_O2_old,
    fra_molar_O2_new,
    fra_molar_O2, // fração molar de O2
    alfa_bolha_cluster, // valor intermediário da clusterização conforme a distribuição normal
    alfa_bolha_cluster_final float64 // simula a alteração do alfa bolha devido a clusterização conforme a distribuição
    normal
}

const (
    Db_entrainment = 0.0008 // 0.0008 // diâmetro da bolha no início da mistura (m)
    s10 = 0.026 // (adimensional)
    Henry = 5.79e6 // (m2/s2)
    Const_Henry = 74800 // T300.8K em bars Tab 14.6 Çengel
    C_equilibrio = 0.016915 // 0.021 // 0.016915 // 97.202,70 Pa / 5,79e6 (m2s2) = 0,016788 kg/m3 >>>
    Politano(2009): 0,0152 (8,5°C) e 0,0154 (9,8°C) - conforme P&D

```

```

// 0.021 kg m-3 considering T = 27.8°C, P=97.2 kPa, with N2 ≈ 0.0122 kg m-3, O2 ≈ 0.0078 kg m-3, Ar ≈ 0.00052 kg m-3, CO2
≈ 0.00052 kg m-3;
DT = 1e-8 // (m²/s) coeficiente de difusão de massa (difusividade de massa)
nuagua = 1e-06 // (m²/s)
Sigma = 0.07 // (kg/m²)
T = 300.83 // temp em Kelvin (K) 273 + 27.83°C
P_Vapor = 3.7376 // Table 1.22 Vapor Pressure of Freshwater as a Function of Temperature in KPa (Colt) t=27.8°C
R = 287.05 // const universal dos gases (m²/s²/K)
DdtOn = 1 // (adimensional)
// limites de calculo - necessarios para nao explodirem os resultados
Reb_min = 0 // (adimensional)
Reb_max = 1e+21 // [Ishii (1979)] (adimensional)
Cd_min = 0.44 // 0.44 = solid limit [Ishii (1979)] (adimensional)
Cd_max = 1000000 // 1.15 = cap limit e 10.0 = drop limit [Ishii (1979)] (adimensional)
Ubr_max = 1.00 // máxima velocidade relativa entre bolhas e água
C_min = 0 // (adimensional)
C_max = 2.5 // (adimensional)
N_min = 1 // milhoes de bolhas por metro cubico (1/m³) original N_min = 0
N_max = 1e+21 // milhoes de bolhas por metro cubico (1/m³)
Db_min = 1e-21 // campo de diâmetro das bolhas (m)
Db_max = 0.0009 // 1e+21 // campo de diâmetro das bolhas (m)
ro_agua = 9810 // densidade da água (N/m³)
ro_ar = 11.772 // densidade do ar (N/m³)
alpha_bolha_max = 1 // limite maximo do valor das bolhas (adimensional)
alpha_bolha_filtro = 0 // filtro para ativar a opcao de corte do calculo de alpha_bolha (adimensional)
alpha_bolha_inicial = 0.05 // relação de bolha no inicio da mistura (adimensional)
alpha_bolha_distribuicao = 0.03 // relação de bolha no inicio da mistura (adimensional)
N_filtro = 0 // filtro para ativar a opcao de corte do calculo de N (adimensional)
adjust_rho_agua = 0 // (adimensional)
alphanMin = 0.00001
visc_din_agua = 1.002e-5 // (viscosidade dinâmica da água a 20°C em pascal)
visc_cine = 1.004e-6 // viscosidade cinemática da água a temperatura de 20 °C
Schmidt = 0.7 // conforme Wan, Politano e Weber (2018)
Patm = 98130 // KPa p(h)=p0 (1-2,25577*10-5*h)^5, 2,25588 po=101325 h = 272m(Elv NA reservatório) considerando
coordenadas de Colíder
Gravidade = 9.81 // Aceleração da gravidade (m/s²)
Rel_O2_ar = 0.368 // fração molar do O2 no dissolvido no ar
Dab_Air_O2 = 2.552e-5 // Binary diffusion coefficients of dilute gas mixtures at 1 atm Tabela 14.4 Çengel temp 300,8K m²/s
eq. 14-15
Dab_O2_H2O = 2.4e-9 // Binary diffusion coefficients of dilute gas mixtures at 1 atm Tabela 14.3 Çengel temp 298K
Vel_maxima = 30 // Velocidade máxima nos cálculos

) // valor maximo de alpha.water para o qual o termo sumidouro de alpha.bolha calculado (adimensional)

// const Elv_fundo_bacia = 236.50
const Elv_na_jusante = 253.21 // 250.5 // El Q=800 // 247.11 Elv Q=300 // 253.21 // Elevação do nível de água de jusante = 1600
m³/s
const Lim_na_jus int = 253 // 250 // 247 // 253 // Elevação do nível de água de jusante (inteiro)
const Min_na_jus float64 = 236.5 // Elevação mínima do NA de jusante
const Elv_reservatorio = 272 // Elevação do nível de água de montante
const Comprimento_x = 210 // comprimento da bacia e laje de proteção - No Open FOam foram simulados dados até x = 500m
const Comprimento_y = 260
const Dimensao_malha = 0.1
const Dimensao_malha_x = 0.1
const Dimensao_malha_z = 0.1
const Array_Comprimento_x = int(Comprimento_x/Dimensao_malha_x)
const Array_Comprimento_z = int(((float64(Lim_na_jus) - Min_na_jus)/Dimensao_malha_z)) // int(((float64(253.5) -
Min_na_jus)/Dimensao_malha_z))

type TEstadistica_dado_parafoam[5200][500] Estrutura_estadistica_dados_lidos
type TEstadistica_bolha[5200][500] Estrutura_estadistica_bolha
type TEstadistica_dado_parafoam_tdg[5200][500] Estrutura_dado_parafoam_tdg
type TEstadistica_dado_Automato[5200][500] Estrutura_dado_Automato

// Variáveis Globais
var Estadistica_dado_parafoam TEstadistica_dado_parafoam
var Estimar_malha_dado_parafoam TEstadistica_dado_parafoam // para preencher os dados da malha utilizando os dados
disponíveis
var Estadistica_bolha TEstadistica_bolha
var Estadistica_dado_parafoam_tdg TEstadistica_dado_parafoam_tdg
var Estadistica_dado_Automato TEstadistica_dado_Automato

```

```

var Concentracao_tdg_1, Concentracao_tdg [5200][500] float64 // carrega os dados da resolução da equação 40, considera o
tempo t-1 e o tempo t (atual)
var Leitura_vetor_parafoam Estrutura_dado_parafoam // linha com os dados de saída do paraFoam - transforma de string para
float p/ array
var Leitura_vetor_parafoam_tdg Estrutura_dado_parafoam_tdg // linha com os dados de saída do paraFoam arquivo TDG -
transforma de string para float p/ array
var Dado_parafoam [] Estrutura_dado_parafoam // array com todos os dados de saída do paraFoam
var Dado_parafoam_tdg [] Estrutura_dado_parafoam_tdg // array com todos os dados de saída do paraFoam TDG
var Elv_fundo_bacia float64 // A elevação do fundo da bacia é calculado conforme a posição x
var Num_bolha, Num_bolha_1 [5200][500] float64 // Densidade do número da bolha no tempo i e i+1
var Tempo_percorer_bacia float64 // cálculo do tempo médio para uma partícula percorrer toda a bacia calculado na func
Calcula_Campo_Concentracao ()

// Variáveis usadas em todo o pacote
var concentracao, concentracao_1 [][] float64 // resultado da derivada da concentração de TDG
var tempo_maximo int = 10000 // número de passos de tempo simulados TDG Wang
var tempo_maximo_automato int = 1000000 // número de passos de tempo simulados TDG automato celular
// variáveis que subdivide os parâmetros iniciais em um matriz para cálculo
var vel_media_x_t_sim, vel_media_y_t_sim, vel_media_z_t_sim, alpha_mean_t_sim, vel_inst_ubr_t_sim, pressao_mean_t_sim
[][]float64
var Result_Divergence_al_ul_C, Result_Termo_Fonte_TDG, Result_Divergence_visc_al_C,
Result_Divergence_al_ul_C_1, Result_Termo_Fonte_TDG_1,
Result_Divergence_visc_al_C_1 [Array_Comprimento_x][Array_Comprimento_z]float64 // dados com o passo de concentração

var visc_turb float64 = 0.09 *(1*1/0.1) // constante do modelo turbulento * energia cinética turbulenta (m²/s²) / taxa de
dissipação de energia (m²/s³)

var grad_xz, soma_grad_xz float64 // terceiro termo da equação 40
var div_a, soma_div_a float64 // segundo termo da equação 40
var div_a_1 float64 // terceiro termo da equação 24

// Função para o calculo da velocidade da bolha

func Atribui_dado_parafoam_TDG () {
// atribui os valores da leitura dos dados de TDG ao Estatistica_dado_parafoam para os cálculos do TDG
for i := 1; i < Array_Comprimento_x-1; i++ {
for j := Array_Comprimento_z-1; j >= 1; j-- { // considera que o fluxo inicia de montante para jusante e de cima para baixo

Estatistica_dado_parafoam[i][j].Vel_media_x = Estatistica_dado_parafoam_tdg [i][j].Ux // velocidade média no eixo x
Estatistica_dado_parafoam[i][j].Vel_media_y = Estatistica_dado_parafoam_tdg [i][j].Uy // velocidade média no eixo y
Estatistica_dado_parafoam[i][j].Vel_media_z = Estatistica_dado_parafoam_tdg [i][j].Uz // velocidade média no eixo z
Estatistica_dado_parafoam[i][j].Numero_dados_lido = Estatistica_dado_parafoam_tdg [i][j].Numero_dados_lido // número de
dados lidos
Estatistica_dado_parafoam[i][j].Media_flutuacao_x = Estatistica_dado_parafoam_tdg [i][j].Tdg_ubrx // média da flutuação
das velocidades o eixo x
Estatistica_dado_parafoam[i][j].Media_flutuacao_y = Estatistica_dado_parafoam_tdg [i][j].Tdg_ubry // média da flutuação
das velocidades o eixo y
Estatistica_dado_parafoam[i][j].Media_flutuacao_z = Estatistica_dado_parafoam_tdg [i][j].Tdg_ubrz // média da flutuação
das velocidades o eixo z
Estatistica_dado_parafoam[i][j].Alpha_water = Estatistica_dado_parafoam_tdg [i][j].Alpha_water // leitura do alpha_water
Estatistica_dado_parafoam[i][j].AlphaMean_water = Estatistica_dado_parafoam_tdg [i][j].AlphaMean_water // leitura da
alpha_water médio
Estatistica_dado_parafoam[i][j].Pressao = Estatistica_dado_parafoam_tdg [i][j].Pressao // pressão em bar
Estatistica_dado_parafoam[i][j].Pressao_mean = Estatistica_dado_parafoam_tdg [i][j].Pressao_mean // leitura da pressão
média em bar
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_vel_media_x // cálculo do desvio padrão da velocidade média no eixo
x
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_vel_media_y // cálculo do desvio padrão da velocidade média no eixo y
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_vel_media_z // cálculo do desvio padrão da velocidade média no eixo
z
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_vel_media_flutuacao_x // cálculo do desvio padrão da flutuação de
velocidade no eixo x
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_vel_media_flutuacao_y // cálculo do desvio padrão da flutuação de
velocidade no eixo y
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_vel_media_flutuacao_z // cálculo do desvio padrão da flutuação de
velocidade no eixo z
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_alpha_water // leitura do alpha_water
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_alphaMean_water // leitura da alpha_water médio
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_pressao // pressão em bar
// Estrutura_estatistica_dados_lidos[i][j].Desv_padrao_pressao_mean // leitura da pressão média em bar
}
}
} // fim da func Atribui_dado_parafoam_TDG ()

```

```

func Inicializa_fundo_bacia (x float64) float64 { // retorna o valor da elevação do fundo da bacia dada a distância x (do eixo da
crista do vertedouro)
//var Fundo_bacia [2][31] float64 // com os dados do fundo da bacia
var Fundo_bacia_defletor [2][36] float64 // com os dados do fundo da bacia com o defletor
var n int
var z float64
// Com defletor
Fundo_bacia_defletor [0][0] = 0
Fundo_bacia_defletor [0][1] = 0.6
Fundo_bacia_defletor [0][2] = 1.26
Fundo_bacia_defletor [0][3] = 1.86
Fundo_bacia_defletor [0][4] = 2.46
Fundo_bacia_defletor [0][5] = 3.66
Fundo_bacia_defletor [0][6] = 4.88
Fundo_bacia_defletor [0][7] = 6.06
Fundo_bacia_defletor [0][8] = 7.31
Fundo_bacia_defletor [0][9] = 8.51
Fundo_bacia_defletor [0][10] = 10.15
Fundo_bacia_defletor [0][11] = 11.96
Fundo_bacia_defletor [0][12] = 13.97
Fundo_bacia_defletor [0][13] = 16.35
Fundo_bacia_defletor [0][14] = 17.66
Fundo_bacia_defletor [0][15] = 21.67
Fundo_bacia_defletor [0][16] = 23.17
Fundo_bacia_defletor [0][17] = 23.18
Fundo_bacia_defletor [1][18] = 23.29
Fundo_bacia_defletor [1][19] = 23.30
Fundo_bacia_defletor [1][20] = 25.10
Fundo_bacia_defletor [0][21] = 26.68
Fundo_bacia_defletor [0][22] = 30.33
Fundo_bacia_defletor [0][23] = 33.99
Fundo_bacia_defletor [0][24] = 38.18
Fundo_bacia_defletor [0][25] = 39.76
Fundo_bacia_defletor [0][26] = 40.96
Fundo_bacia_defletor [0][27] = 44.24
Fundo_bacia_defletor [0][28] = 47.84
Fundo_bacia_defletor [0][29] = 164.6
Fundo_bacia_defletor [0][30] = 166.52
Fundo_bacia_defletor [0][31] = 168.44
Fundo_bacia_defletor [0][32] = 170.36
Fundo_bacia_defletor [0][33] = 172.28
Fundo_bacia_defletor [0][34] = 174.2
Fundo_bacia_defletor [0][35] = 500 // continua na mesma elevação
Fundo_bacia_defletor [1][0] = 255.30
Fundo_bacia_defletor [1][1] = 255.28
Fundo_bacia_defletor [1][2] = 255.21
Fundo_bacia_defletor [1][3] = 255.11
Fundo_bacia_defletor [1][4] = 254.98
Fundo_bacia_defletor [1][5] = 254.64
Fundo_bacia_defletor [1][6] = 254.19
Fundo_bacia_defletor [1][7] = 253.63
Fundo_bacia_defletor [1][8] = 252.94
Fundo_bacia_defletor [1][9] = 252.19
Fundo_bacia_defletor [1][10] = 251.09
Fundo_bacia_defletor [1][11] = 249.89
Fundo_bacia_defletor [1][12] = 248.55
Fundo_bacia_defletor [1][13] = 247.19
Fundo_bacia_defletor [1][14] = 247.00
Fundo_bacia_defletor [1][15] = 247.00
Fundo_bacia_defletor [1][16] = 247.14
Fundo_bacia_defletor [1][17] = 245.28
Fundo_bacia_defletor [1][18] = 244.79
Fundo_bacia_defletor [1][19] = 242.70
Fundo_bacia_defletor [1][20] = 241.64
Fundo_bacia_defletor [1][21] = 240.89
Fundo_bacia_defletor [1][22] = 239.42
Fundo_bacia_defletor [1][23] = 238.27
Fundo_bacia_defletor [1][24] = 237.31
Fundo_bacia_defletor [1][25] = 237.05
Fundo_bacia_defletor [1][26] = 236.89
Fundo_bacia_defletor [1][27] = 236.59
Fundo_bacia_defletor [1][28] = 236.5
Fundo_bacia_defletor [1][29] = 236.5
Fundo_bacia_defletor [1][30] = 237.4
Fundo_bacia_defletor [1][31] = 238.3
Fundo_bacia_defletor [1][32] = 239.2

```

```
Fundo_bacia_defletor [1][33] = 240.1
Fundo_bacia_defletor [1][34] = 241
Fundo_bacia_defletor [1][35] = 241
n = len(Fundo_bacia_defletor[0])
```

```
/* Sem defletor
```

```
Fundo_bacia [0][0] = 0
Fundo_bacia [0][1] = 0.6
Fundo_bacia [0][2] = 1.26
Fundo_bacia [0][3] = 1.86
Fundo_bacia [0][4] = 2.46
Fundo_bacia [0][5] = 3.66
Fundo_bacia [0][6] = 4.88
Fundo_bacia [0][7] = 6.06
Fundo_bacia [0][8] = 7.31
Fundo_bacia [0][9] = 8.51
Fundo_bacia [0][10] = 10.15
Fundo_bacia [0][11] = 11.96
Fundo_bacia [0][12] = 13.97
Fundo_bacia [0][13] = 17.62
Fundo_bacia [0][14] = 20.02
Fundo_bacia [0][15] = 23.08
Fundo_bacia [0][16] = 26.68
Fundo_bacia [0][17] = 30.33
Fundo_bacia [0][18] = 33.99
Fundo_bacia [0][19] = 38.18
Fundo_bacia [0][20] = 39.76
Fundo_bacia [0][21] = 40.96
Fundo_bacia [0][22] = 44.24
Fundo_bacia [0][23] = 47.84
Fundo_bacia [0][24] = 164.6
Fundo_bacia [0][25] = 166.52
Fundo_bacia [0][26] = 168.44
Fundo_bacia [0][27] = 170.36
Fundo_bacia [0][28] = 172.28
Fundo_bacia [0][29] = 174.2
Fundo_bacia [0][30] = 500 // continua na mesma elevação
Fundo_bacia [1][0] = 255.30
Fundo_bacia [1][1] = 255.28
Fundo_bacia [1][2] = 255.21
Fundo_bacia [1][3] = 255.11
Fundo_bacia [1][4] = 254.98
Fundo_bacia [1][5] = 254.64
Fundo_bacia [1][6] = 254.19
Fundo_bacia [1][7] = 253.63
Fundo_bacia [1][8] = 252.94
Fundo_bacia [1][9] = 252.19
Fundo_bacia [1][10] = 251.09
Fundo_bacia [1][11] = 249.89
Fundo_bacia [1][12] = 248.55
Fundo_bacia [1][13] = 246.11
Fundo_bacia [1][14] = 244.51
Fundo_bacia [1][15] = 242.69
Fundo_bacia [1][16] = 240.89
Fundo_bacia [1][17] = 239.42
Fundo_bacia [1][18] = 238.27
Fundo_bacia [1][19] = 237.31
Fundo_bacia [1][20] = 237.05
Fundo_bacia [1][21] = 236.89
Fundo_bacia [1][22] = 236.59
Fundo_bacia [1][23] = 236.5
Fundo_bacia [1][24] = 236.5
Fundo_bacia [1][25] = 237.4
Fundo_bacia [1][26] = 238.3
Fundo_bacia [1][27] = 239.2
Fundo_bacia [1][28] = 240.1
Fundo_bacia [1][29] = 241
Fundo_bacia [1][30] = 241
n = len(Fundo_bacia[0]) */
```

```
for i:= 0; i< n-1; i++ {
    if x >= Fundo_bacia_defletor [0][i] && x < Fundo_bacia_defletor [0][i+1]{
        z = Fundo_bacia_defletor [1][i] + (Fundo_bacia_defletor [1][i+1] - Fundo_bacia_defletor [1][i]) / (Fundo_bacia_defletor
[0][i+1] - Fundo_bacia_defletor [0][i]) * (x-Fundo_bacia_defletor [0][i])
        return z
    }
}
```

```

}

return z

} // fim da função Inicializa_fundo_bacia

func Fracao_Molar_ar (profundidade float64, i, j int) float64 { // calcula a fração molar do ar dissolvido na água a determinada
profundidade
  P_total:=Patm + ro_agua*profundidade - P_Vapor
  C_ar :=1/Const_Henry*P_total // concentraçã molar do ar dissolvido na água (mol/m3)
  C_h2o := ro_agua/Gravidade/0.018 // Concentração molar da água pura
  fra_molar_ar:=C_ar/(C_ar+C_h2o) // fração molar do ar dissolvido em água a x metro de profundidade [adimensional]
  fra_molar_O2:=fra_molar_ar*Rel_O2_ar
  return fra_molar_O2 // mg/L

} // fim da func Fracao_Molar

func O2_Equilibrio_Profundidade (profundidade float64, i, j int) (float64, float64) {
// função que calcula a concentração de equilíbrio de O2 dissolvido na água (solubilidade do oxigênio) conforme a
profundidade
var P_total float64
xO2 := 0.2095 // fração molar O2 no ar
K_H := 1.0e-3 // mol/(L·atm) para 30°C
M_O2 := 32.0 // g/mol

// Pressão total (atm)

if Estatistica_dado_parafoam[i][j].AlphaMean_water <1.01 && i!=0 && j!=0{
  P_total=Estatistica_dado_parafoam[i][j].Pressao_mean/Patm
}
if i == 0 && j==0 {
  P_total=(Patm + ro_agua*profundidade - P_Vapor)/Patm
}
if Estatistica_dado_parafoam[i][j].AlphaMean_water >=1.01{
  P_total=(Patm + ro_agua*profundidade - P_Vapor)/Patm
}

// Pressão parcial de O2
P_O2 := xO2 * P_total
// Concentração de equilíbrio (mol/L)
Ceq_molL := K_H * P_O2 // c
// Converter para mg/L
Ceq_mgL := Ceq_molL * M_O2 * 1000.0 // concentração em mg/L

return Ceq_molL, Ceq_mgL
} // fim da func O2_Equilibrio_Profundidade (z, t float64) float64

func sherwood (db, ub, D, nu float64) float64 {
  Re := ub * db / nu
  Sc := nu / D
  return 2.0 + 0.6*math.Sqrt(Re)*math.Pow(Sc, 1.0/3.0)
}

func bubbleMassTransfer (db, ub, D, nu, Cliq, Cstar float64) (N_mol float64) {
  Sh := sherwood(db, ub, D, nu)
  kL := Sh * D / db // m/s
  A := math.Pi * db * db
  t_s := Dimensao_malha_z / ub // s (tempo de residência no trecho)
  N_mol = kL * A * (Cliq - Cstar) * t_s // mol (sinal indica direção)
  return
}

func f_deltaC_mg_L (profundidade float64, i, j int) (deltaC_mol_m3, Ceq_molL, Ceq_molL_1 float64) {
// função tem a finalidade calcular o aumento da concentração do O2 em mmg/l devido ao aumento da profundidade (quando
o fluxo empurra para profundidades maiores)
var P_total, P_total_1 float64
xO2 := 0.2095 // fração molar O2 no ar
K_H := 1.0e-3 // mol/(L·atm) para 30°C
M_O2 := 32.0 // g/mol

// Pressão total (atm)

if Estatistica_dado_parafoam[i][j].AlphaMean_water <1.01 && i!=0 && j!=0{
  P_total=Estatistica_dado_parafoam[i][j].Pressao_mean/Patm
  if (j-1) > 0 {

```

```

    P_total_1 = Estatistica_dado_parafoam[i][j-1].Pressao_mean/Patm
  } else {
    P_total_1 = (Estatistica_dado_parafoam[i][j].Pressao_mean+ ro_agua*Dimensao_malha_z)/Patm
  }
  if i == 0 && j==0 {
    P_total=(Patm + ro_agua*profundidade - P_Vapor)/Patm
    P_total_1=(Patm + ro_agua*profundidade +ro_agua*Dimensao_malha_z - P_Vapor)/Patm
  }
  if Estatistica_dado_parafoam[i][j].AlphaMean_water >=1.01{
    P_total=(Patm + ro_agua*profundidade - P_Vapor)/Patm
    P_total_1=(Patm + ro_agua*profundidade +ro_agua*Dimensao_malha_z - P_Vapor)/Patm
  }

  // Pressão parcial de O2
  P_O2 := xO2 * P_total
  P_O2_1 := xO2 * P_total_1
  // Concentração de equilíbrio (mol/L)
  Ceq_molL = K_H * P_O2 // c
  Ceq_molL_1 = K_H * P_O2_1 // c
  // Converter para mg/L

  // Diferença de concentração (mol/m³ e mg/L)
  deltaC_mol_m3 =Ceq_molL_1/Ceq_molL

  //deltaC_Ceq_mgL := deltaC_mol_m3 * M_O2 * 1000.0 // concentração em mg/L

  return
}

func Diametro_bolha_d2 (d1, p1, p2 float64 ) float64 {
  // função para o cálculo do diâmetro da bolha de forma simples

  const n = 1.4 //(adiabático, variações muito rápidas)

  d2 := d1*math.Pow((p1/p2),1/(3*n))

  return d2
} // fim da func Diametro_bolha

func Fick_Difusao_Adveccao () {

  var arq_automato*.os.File
  var err error
  var j_fundo int
  var C_O2 float64 // concentração de equilíbrio para profundidade em mol/L
  var C_O2_mg float64 // concentração de equilíbrio para profundidade em mg/L
  //var dt float64

  var adv_x, adv_z float64
  var dyA_dx2, dyA_dz2 float64
  var soma_dyA_dx2, soma_dyA_dz2 float64
  var passo_x float64
  var passo_z float64
  var perda_superficie float64 // perda := kLa * (C_eq - C_O2) * deltaT // mg/L perdidos
  var i, j, k, l int

  // inicializa o gerador com uma semente variável (hora atual)
  rand.Seed(time.Now().UnixNano())

  // função que calcula o fluxo total de massa usando a lei di Fick, conforme equação 14-50 Çengel

  //ya:=alpha_bolha_inicial // % de ar na mistura
  //yb:=1-ya // % de água na mistura

  // Conversão fração mássica (ar é traço)
  //wa_entrada := (ya * ro_ar) / (ya * ro_ar + yb * ro_agua)
  //C_H2O := 55.56 // mol/L água
  //C_total:=55.5 // mol/L para água pura
  //M_O2 := 32.0 // g/mol
  C_O2, C_O2_mg = O2_Equilibrio_Profundidade(0, 0, 0)
  C_O2_equilibrio_h_0:=C_O2 // concentração de equilíbrio para profundidade 0 em mol/(L.atm)

  profundidade:=Elv_na_jusante - Min_na_jus
  C_O2, C_O2_mg = O2_Equilibrio_Profundidade(profundidade, 0, 0)

```

```

//C_O2 = C_O2/(M_O2*1000) // concentração de equilíbrio para profundidade 0 em mol/(L.atm)
//C_O2_equilibrio_h_maximo:=C_O2/(C_H2O+C_O2) // concentração de equilíbrio para profundidade máxima em mol/(L.atm)
C_O2_equilibrio_h_maximo:=C_O2 // concentração de equilíbrio para profundidade máxima em mol/(L.atm)
// xO2 := 0.2095 // fração molar O2 no ar
KLa := 2.0 // coeficiente volumétrico de transferência de oxigênio (1/dia), depende de turbulência, vento, etc. coef. de
transferência de massa por convecção

delta_t:= 0.0001 // passo de tempo 0.01s

//tempo_Auto:=100

//dt = 1/float64(tempo_maximo)

soma_dyA_dx2 = 0
soma_dyA_dz2 = 0 // apenas para teste

arq_automato, err = os.Create("Concentracao_Automato.txt")
if err != nil {
    fmt.Println("Erro ao criar o arquivo:", err)
}
defer arq_automato.Close()

cabecalho := []string{ // cabeçalho do arquivo arq_automato
    "Tempo", "Posição x", "Posição y", "AlfaBolha", "fracao_molar_O2", "Concentracao_O2", "Concentracao_O2_%",
    "Vx_bolha", "Vz_bolha", "TDG_P&D",
}

for _, header := range cabecalho {
    fmt.Fprintf(arq_automato, "%40s", header)
}
fmt.Fprintf(arq_automato, "\n\n")

// Inicializa matriz de fração molar de O2 (yA) considerando todos os pontos em equilíbrio
for i = 0; i < Array_Comprimento_x; i++ {
    for j = 0; j < Array_Comprimento_z; j++ {
        Estatistica_dado_Automato[i][j].fra_molar_O2 = C_O2_equilibrio_h_0 // fração molar de equilíbrio conversão de mg/L p/
mol/L
        Estatistica_dado_Automato[i][j].fra_molar_O2_new = Estatistica_dado_Automato[i][j].fra_molar_O2
        Estatistica_dado_Automato[i][j].Concentracao_O2_percentual =
Estatistica_dado_Automato[i][j].fra_molar_O2/C_O2_equilibrio_h_0
        Estatistica_dado_Automato[i][j].alfa_bolha_cluster = alfa_bolha_inicial
        // Gera um número com distribuição normal padrão (média=0, desvio=1)
        Estatistica_dado_Automato[i][j].alfa_bolha_cluster_final = alfa_bolha_inicial/(1-alfa_bolha_inicial) +
(rand.Float64()*math.Pow(alfa_bolha_distribuiçao,2))/(math.Pow(1-alfa_bolha_inicial,3))
        if Estatistica_dado_Automato[i][j].alfa_bolha_cluster_final < alfa_bolha_inicial-alfa_bolha_distribuiçao {
            Estatistica_dado_Automato[i][j].alfa_bolha_cluster_final = alfa_bolha_inicial-alfa_bolha_distribuiçao
        }
        if Estatistica_dado_Automato[i][j].alfa_bolha_cluster_final > alfa_bolha_inicial+alfa_bolha_distribuiçao {
            Estatistica_dado_Automato[i][j].alfa_bolha_cluster_final = alfa_bolha_inicial+alfa_bolha_distribuiçao
        }
    }
}

// início do cálculo para o tempo maior que 2 - no tempo igual a 1 tudo está em equilíbrio
for tempo := 0; tempo <= tempo_maximo_automato; tempo++ {

    // Subdivisão da malha de 1m x 1m dos dados fonte em sub-malha (n° div_malha)
    for i=177; i < Array_Comprimento_x-1; i++ { // inicia em 177 porque é o ponto de inserção do O2
        Elv_fundo_bacia = Inicializa_fundo_bacia(float64(i)/Dimensao_malha_x)
        j_fundo = int(math.Ceil((Inicializa_fundo_bacia(float64(i)*Dimensao_malha_x) - Min_na_jus)/Dimensao_malha_z))
        for j=j_fundo+1; j < Array_Comprimento_z-2; j++ {

            // insere a concentração de O2 [mg/l] no final do defletor entre a elevação do defletor [247] e o NA de jusante
            if (float64(i)*Dimensao_malha_x) >= 17.7 && (float64(i)*Dimensao_malha_x) <= 27.7 { //24.7{
                if (Min_na_jus+float64(j)*Dimensao_malha_z) >= 247 { //&& (Min_na_jus+float64(j)*Dimensao_malha_z)
                    <= float64(Lim_na_jus) { //float64(Array_Comprimento_z-2) { // (Min_na_jus+float64(j)*Dimensao_malha_z) <= Elv_na_jusante {
                        profundidade:=Elv_na_jusante - Min_na_jus-float64(j)*Dimensao_malha_z // sendo H o ponto da malha
                        // Fração molar na entrada (O2)
                        C_O2, C_O2_mg = O2_Equilibrio_Profundidade(profundidade, 0, 0)
                    }
                }
            }
        }
    }
}

```

```

    if C_O2 < C_O2_equilibrio_h_0 {
        C_O2 = C_O2_equilibrio_h_0
    }
    //yA_entrada := C_O2
    //yA_entrada := (C_O2-
C_O2_equilibrio_h_0)/((delta_t/Dimensao_malha_x)/float64(tempo_maximo_automato)*(Estatistica_dado_Automato[i][j].Vx_bol
ha))+ C_O2_equilibrio_h_0
    //Estatistica_dado_Automato[i][j].fra_molar_O2 = (1-alpha_bolha_inicial*xO2) * Estatistica_dado_Automato[i-
1][j].fra_molar_O2 + alpha_bolha_inicial*xO2*yA_entrada
    yA_entrada := C_O2
    vazao_entrada:=Dimensao_malha_x*Dimensao_malha_z*Estatistica_dado_Automato[i][j].Vx*delta_t // vazão/volume
que está entrando na célula
    volume_celula:=Dimensao_malha_x*Dimensao_malha_z*Dimensao_malha // volume da célula
    vazao_entrada_z:=Dimensao_malha_x*Dimensao_malha_z*Estatistica_dado_Automato[i][j].Vz*delta_t // vazão/volume
que está entrando na célula
    // verifica se a velocidade da bolha é positiva ou negativa e considera de onde vem a vazão molar

    // Estatistica_dado_Automato[i][j].fra_molar_O2_new = C_O2_equilibrio_h_0/volume_celula*(volume_celula-
math.Abs(vazao_entrada)-math.Abs(vazao_entrada_z))+yA_entrada*(1-
Estatistica_dado_parafoam[i][j].AlphaMean_water)/volume_celula*(math.Abs(vazao_entrada)+math.Abs(vazao_entrada_z))

    // 1° trecho reflete o O2 que permaneceu na célula, 2° reflete o incremento do O2 que entra na célula e o 3° reflete o O2
de equilibrio que entra na célula
    Estatistica_dado_Automato[i][j].fra_molar_O2_new =
Estatistica_dado_Automato[i][j].fra_molar_O2/volume_celula*(volume_celula-math.Abs(vazao_entrada)-
math.Abs(vazao_entrada_z))+yA_entrada*Estatistica_dado_Automato[i][j].alfa_bolha_cluster/volume_celula*(math.Abs(vazao_
entrada)+math.Abs(vazao_entrada_z))+
    Estatistica_dado_Automato[i][j].fra_molar_O2*(1-
Estatistica_dado_Automato[i][j].alfa_bolha_cluster)/volume_celula*(math.Abs(vazao_entrada)+math.Abs(vazao_entrada_z))

    Estatistica_dado_Automato[i][j].fra_molar_O2 = Estatistica_dado_Automato[i][j].fra_molar_O2_new

    //Estatistica_dado_Automato[i][j].fra_molar_O2/volume_celula*(volume_celula-
vazao_entrada)+yA_entrada*alpha_bolha_inicial/volume_celula*(vazao_entrada)
    } else{
    //Estatistica_dado_Automato[i][j].fra_molar_O2 = (1-alpha_bolha_inicial) *C_O2_equilibrio_h_0 +
alpha_bolha_inicial*Estatistica_dado_Automato[i][j].fra_molar_O2
    Estatistica_dado_Automato[i][j].fra_molar_O2 = C_O2_equilibrio_h_0
    }
}

// Difusão 2D (centrada)
dyA_dx2 = 0
dyA_dz2 = 0
adv_x = 0
adv_z = 0
perda_superficie = 0

if Estatistica_dado_parafoam[i][j].AlphaMean_water <1.01 && Estatistica_dado_parafoam[i-1][j].AlphaMean_water <1.01
&& Estatistica_dado_parafoam[i][j-1].AlphaMean_water <1.01{
    if i > 0 && i < Array_Comprimento_x-1 && (float64(i)*Dimensao_malha_x) != 17.7{
        dyA_dx2 = (Estatistica_dado_Automato[i+1][j].fra_molar_O2 - 2*Estatistica_dado_Automato[i][j].fra_molar_O2 +
Estatistica_dado_Automato[i-1][j].fra_molar_O2) / (Dimensao_malha_x * Dimensao_malha_x)
        soma_dyA_dx2 = soma_dyA_dx2+ dyA_dx2
    }
    if j > 0 && j < Array_Comprimento_z-1 {
        dyA_dz2 = (Estatistica_dado_Automato[i][j+1].fra_molar_O2 - 2*Estatistica_dado_Automato[i][j].fra_molar_O2 +
Estatistica_dado_Automato[i][j-1].fra_molar_O2) / (Dimensao_malha_z * Dimensao_malha_z)
        soma_dyA_dz2 = soma_dyA_dz2 + dyA_dz2
    }
}

// Advecção (upwind)

if i > 0 {
    adv_x =
(Estatistica_dado_Automato[i][j].fra_molar_O2*(Estatistica_dado_Automato[i][j].Vx_bolha+Estatistica_dado_Automato[i][j].Vx) -
Estatistica_dado_Automato[i-1][j].fra_molar_O2*(Estatistica_dado_Automato[i-1][j].Vx_bolha+Estatistica_dado_Automato[i-
1][j].Vx)) / Dimensao_malha_x
}
if j > 0 {
    adv_z =
(Estatistica_dado_Automato[i][j].fra_molar_O2*(Estatistica_dado_Automato[i][j].Vz_bolha+Estatistica_dado_Automato[i][j].Vz) -
Estatistica_dado_Automato[i][j-1].fra_molar_O2*(Estatistica_dado_Automato[i][j-1].Vz_bolha+Estatistica_dado_Automato[i][j-
1].Vz)) / Dimensao_malha_z
}
}

```

```

// perda de concentração na superfície livre
Estatistica_dado_Automato[i][j].fra_molar_O2_new = Estatistica_dado_Automato[i][j].fra_molar_O2 + delta_t*(-adv_x -
adv_z + Dab_O2_H2O*(dyA_dx2 + dyA_dz2))
if j == (Array_Comprimento_z-2) {
    if Estatistica_dado_Automato[i][j].fra_molar_O2 > C_O2_equilibrio_h_0 {
        perda_superficie = delta_t*KLa/86400 * (C_O2_equilibrio_h_0 - Estatistica_dado_Automato[i][j].fra_molar_O2) // em
Mol/L
        Estatistica_dado_Automato[i][j].fra_molar_O2_new = Estatistica_dado_Automato[i][j].fra_molar_O2_new +
perda_superficie
    }
} else {
    Estatistica_dado_Automato[i][j].fra_molar_O2_new = Estatistica_dado_Automato[i][j].fra_molar_O2
}

// Atualização explícita
//if Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 {
// Estatistica_dado_Automato[i][j].fra_molar_O2_new = Estatistica_dado_Automato[i][j].fra_molar_O2 + delta_t*( +
Dab_O2_H2O*(dyA_dx2 + dyA_dz2))

if (Min_na_jus+float64(i)*Dimensao_malha_z) == 246 && tempo%2000 == 0 {
    fmt.Printf("O Valor de yANew é %8d %8.2f %10.10f %10.10f %10.10f %10.10f %10.10f %10.10f %10.10f %10.10f\n", tempo, float64(i)*Dimensao_malha_x, Estatistica_dado_Automato[i][j].fra_molar_O2_new,
Estatistica_dado_Automato[i][j].fra_molar_O2, dyA_dx2, dyA_dz2, adv_x, adv_z, Estatistica_dado_Automato[i][j].Vx,
Estatistica_dado_Automato[i][j].Vz, perda_superficie )
}
//}

} // fim do for j:=j_fundo ; j<Array_Comprimento_z; j++
} // fim do for i:=2; i< int(Array_Comprimento_x)-1; i++

// atualiza a concentração de O2 para o próximo instante de tempo
for k=177; k< Array_Comprimento_x-1; k++ {
    Elv_fundo_bacia = Inicializa_fundo_bacia(float64(k)/Dimensao_malha_x)
    j_fundo = int(math.Ceil((Inicializa_fundo_bacia(float64(k)*Dimensao_malha_x) - Min_na_jus)/Dimensao_malha_z))

    for l=j_fundo+1 ; l<Array_Comprimento_z-2; l++{
        profundidade:=Elv_na_jusante - Min_na_jus-float64(l)*Dimensao_malha_z // sendo H o ponto da malha
        // Fração molar na entrada (O2)
        C_O2, C_O2_mg = O2_Equilibrio_Profundidade(profundidade, 0, 0)

        vazao_entrada:=Dimensao_malha_x*Dimensao_malha_z*(Estatistica_dado_Automato[k][l].Vx+Estatistica_dado_Automato[k][l].
Vx_bolha)*delta_t // vazão/volume que está entrando na célula
        volume_celula:=Dimensao_malha_x*Dimensao_malha_z*Dimensao_malha // volume da célula

        vazao_entrada_z:=Dimensao_malha_x*Dimensao_malha_z*(Estatistica_dado_Automato[k][l].Vz+Estatistica_dado_Automato[k]
[l].Vz_bolha)*delta_t // vazão/volume que está entrando na célula

        Estatistica_dado_Automato[k][l].fra_molar_O2_old =
Estatistica_dado_Automato[k][l].fra_molar_O2/volume_celula*(volume_celula-math.Abs(vazao_entrada)-
math.Abs(vazao_entrada_z))

        delta_fra_molar_O2, C_O2_0, C_O2_1 := f_deltaC_mg_L (profundidade,0,0)

        // C* médio ao longo do trecho (aproximação simples)
        CstarAvg := 0.5 * (C_O2_0 + C_O2_1)
        p1 := float64(Patm) // pressão na superfície livre
        p2:= p1 + ro_agua*profundidade // pressão na profundidade
        d1:=Db_entrainment // diâmetro da bolha na superfície livre
        d2:= Diametro_bolha_d2 (d1, p1, p2) // diâmetro da bolha na profundidade da célula
        Nmole := bubbleMassTransfer(d2,Estatistica_dado_Automato[k][l].Vz, Dab_O2_H2O, visc_cine,
Estatistica_dado_Automato[k][l-1].fra_molar_O2_new, CstarAvg)
        if math.IsNaN (Nmole) {
            Nmole=0
        }

        // verifica se a velocidade da bolha é positiva ou negativa e considera de onde vem a vazão molar
        if vazao_entrada > 0 {
            Estatistica_dado_Automato[k][l].fra_molar_O2 = Estatistica_dado_Automato[k][l].fra_molar_O2_old +
Estatistica_dado_Automato[k-1][l].fra_molar_O2_new/volume_celula*(math.Abs(vazao_entrada))

```

```

    } else {
        Estatistica_dado_Automato[k][l].fra_molar_O2 = Estatistica_dado_Automato[k][l].fra_molar_O2_old +
        Estatistica_dado_Automato[k+1][l].fra_molar_O2_new/volume_celula*(math.Abs(vazao_entrada))
    }

    if vazao_entrada_z > 0 { // transferência de O2 da célula inferior para a superior quando a velocidade "Z" é positiva
        // utilizando o método da cinética por bolha
        Estatistica_dado_Automato[k][l].fra_molar_O2 = Estatistica_dado_Automato[k][l].fra_molar_O2 +
        (Estatistica_dado_Automato[k][l-1].fra_molar_O2_new+Nmol)/volume_celula*(math.Abs(vazao_entrada_z))

        // fmt.Printf("O Valor de yANew é %8.6f %8.6f\n",Estatistica_dado_Automato[k][l-1].fra_molar_O2_new, Nmol)
    } else {
        Estatistica_dado_Automato[k][l].fra_molar_O2 = Estatistica_dado_Automato[k][l].fra_molar_O2 +
        (Estatistica_dado_Automato[k][l+1].fra_molar_O2_new+Nmol)/volume_celula*(math.Abs(vazao_entrada_z))
    }

    passo_x = (delta_t/Dimensao_malha_x)*(Estatistica_dado_Automato[k][l].Vx_bolha)
    passo_z = (delta_t/Dimensao_malha_z)*(Estatistica_dado_Automato[k][l].Vz_bolha)

    if passo_x < 1 && passo_z < 1 && passo_x > 0 && passo_z > 0{
    } else {
        // Estatistica_dado_Automato[k][l].fra_molar_O2 = Estatistica_dado_Automato[k-1][l].fra_molar_O2_new
    }

    if passo_z < 1 && passo_z > 0 {
        // Estatistica_dado_Automato[k][l].fra_molar_O2 = (1-passo_z)*Estatistica_dado_Automato[k][l].fra_molar_O2 +
        passo_z*Estatistica_dado_Automato[k][l-1].fra_molar_O2
        delta_fra_molar_O2 = C_O2_1/C_O2_0*delta_fra_molar_O2 // apens para testar
    } else if passo_z<=0 {
        // Estatistica_dado_Automato[k][l].fra_molar_O2 = (1+passo_z)*Estatistica_dado_Automato[k][l].fra_molar_O2 -
        passo_z*Estatistica_dado_Automato[k][l+1].fra_molar_O2
    } else {
        //Estatistica_dado_Automato[k][l].fra_molar_O2 = Estatistica_dado_Automato[k][l-1].fra_molar_O2_new
    }

    // }
    if Estatistica_dado_Automato[k][l].fra_molar_O2 < C_O2_equilibrio_h_0*0.8 {
        Estatistica_dado_Automato[k][l].fra_molar_O2 = C_O2_equilibrio_h_0*0.8
    }
    if Estatistica_dado_Automato[k][l].fra_molar_O2 > C_O2_equilibrio_h_maximo {
        Estatistica_dado_Automato[k][l].fra_molar_O2 = C_O2_equilibrio_h_maximo
    }
}

Estatistica_dado_Automato[k][l].Concentracao_O2 = Estatistica_dado_Automato[k][l].fra_molar_O2/C_O2*C_O2_mg // em
mg/L
Estatistica_dado_Automato[k][l].Concentracao_O2_percentual =
Estatistica_dado_Automato[k][l].fra_molar_O2/C_O2_equilibrio_h_0 // em %

//}
}
}

//if tempo== tempo_maximo_automato {
if tempo%1000000 == 0 && tempo == tempo_maximo_automato {
    // imprime os dados para o tempo final da simulação

    for i=1; i< Array_Comprimento_x-1; i++ {
        for j=j_fundo ; j<Array_Comprimento_z-2; j++){
            if Estatistica_dado_parafoam[i][j].AlphaMean_water <1.01 {
                fmt.Fprintf(arq_automato,"\r\n %40d %40.2f %40.2f %40.4f %40.10f %40.10f %40.6f %40.6f %40.6f %40.4f ",
                    tempo, float64(i)*Dimensao_malha_x, (float64(j)*Dimensao_malha_z+Min_na_jus),
                    Estatistica_dado_Automato[i][j].alfa_bolha_cluster, Estatistica_dado_Automato[i][j].fra_molar_O2,
                    Estatistica_dado_Automato[i][j].Concentracao_O2, Estatistica_dado_Automato[i][j].Concentracao_O2_percentual,
                    Estatistica_dado_Automato[i][j].Vx_bolha+Estatistica_dado_Automato[i][j].Vx,
                    Estatistica_dado_Automato[i][j].Vz_bolha+Estatistica_dado_Automato[i][j].Vz, Estatistica_dado_parafoam_tdg[i][j].Tdg_mc )
                //}
            }
        }
    }
} // fim do if tempo%2000 == 0

```

```

for i = 0; i<Array_Comprimento_x; i++){
    for j = 0; j<Array_Comprimento_z; j++){
        // calcula o alfa_bolha_custer intermediário Gera um número com distribuição normal padrão (média=0, desvio=1)

        Estatistica_dado_Automato[i][j].alfa_bolha_cluster = Estatistica_dado_Automato[i][j].alfa_bolha_cluster +
        (Estatistica_dado_Automato[i][j].alfa_bolha_cluster_final - alfa_bolha_inicial)/float64(tempo_maximo_automato)
        if Estatistica_dado_Automato[i][j].alfa_bolha_cluster < alfa_bolha_inicial-alfa_bolha_distribuiçao {
            Estatistica_dado_Automato[i][j].alfa_bolha_cluster = alfa_bolha_inicial-alfa_bolha_distribuiçao
        }
        if Estatistica_dado_Automato[i][j].alfa_bolha_cluster > alfa_bolha_inicial+alfa_bolha_distribuiçao {
            Estatistica_dado_Automato[i][j].alfa_bolha_cluster = alfa_bolha_inicial+alfa_bolha_distribuiçao
        }
    }
}
} // fim do for tempo := 0; tempo<= tempo_maximo; tempo ++

arq_automato.Close()
fmt.Println("Fim do calculo da concentração de O2 por autômatos celulares ")

} // fim da função func Fick_Difusão_Adveccao

func Calcula_Ubrx (pressao_0,pressao,pressao_1 float64) float64 {
// Função que calcula a velocidade relativa entre a fase bolha e a fase líquida eq. 38

var (Ubr_Ini,
    //Ubr, // Velocidade da bolha (m/s)
    //N_Entrada, // número de bolhas na entrada
    Ubr float64 // Velocidade relativa entre a fase bolha e a fase líquida (m/s)
)
var n_iteracao int

n_iteracao = 0
Ubr_Ini = 0.20

/*if pressao <=0 { //if pressao ==0 {
    pressao = 9810
} */ //9810 - pressão na profundidade de 1 m - a pressão é retirada do arquivo da simulação

F_Ub := func (Ubr_F_ub float64) float64 {
// Calcula a velocidade da bolha
var a, ro_b, grad_p, CD, Reynolds_b float64

Ubr = Ubr_F_ub
grad_p = (pressao_1-pressao_0)/(2*Dimensao_malha)
ro_b = math.Abs(pressao)/(R * T) // equação 35
Reynolds_b = Db_entrainment*math.Abs(float64(Ubr))/nuagua // equação 34 Algoritmo1
if Reynolds_b < Reb_min {Reynolds_b = Reb_min}
if Reynolds_b > Reb_max {Reynolds_b = Reb_max}
CD = 24*(1+0.15*math.Pow(Reynolds_b,0.687))/Reynolds_b // equação 33
if CD < Cd_min {CD = Cd_min}
if CD > Cd_max {CD=Cd_max}
a = ((ro_b*9.81-grad_p) * (4*Db_entrainment))/(3*ro_agua*CD) // equação 37
Ubr = a/math.Sqrt((math.Abs(float64(a)))) // equação 38
if Ubr > Ubr_max {Ubr=Ubr_max}
if Ubr < -Ubr_max {Ubr = -Ubr_max}

return Ubr
} //fim da função func (Ubr_F_ub float64)

Ubr = F_Ub (Ubr_Ini)
Ubr_Ini = Ubr
for {
    Ubr = F_Ub (Ubr)
    if (Ubr_Ini/Ubr < 0.99) || (Ubr_Ini/Ubr > 1.01) || (n_iteracao >100) {
        //fmt.Println("Dentro do IF Velocidade de flutuação da bolha %.2f %.2f %.2f", Ubr, pressao, Ubr_Ini)
        return Ubr
        break // sai do loop for quando atinge a precisão de 0,01
    } else{
        Ubr = F_Ub (Ubr)
        n_iteracao =n_iteracao+1
        //fmt.Println("Fora dados de entrada para cálculo da estatística da bolha %.5.2f %.5.2f %.5.2f", Ubr, pressao, Ubr_Ini)
        Ubr_Ini = Ubr
    }
}
}

```

```

return Ubr

} // fim da função func Calcula_Ubrx (pressao_0,pressao,pressao_1 float64)

func Pressao_efetiva (h, alpha, ux, uz float64) float64 {

// Calcular densidade efetiva
rhoEfetiva := alpha * ro_agua

// Calcular pressão hidrostática estática
pressaoEstatica := Patm + rhoEfetiva*h

// Calcular pressão dinâmica
pressaoDinamica := 0.5 * rhoEfetiva * (math.Pow(ux, 2) + math.Pow(uz, 2))

// Pressão total
pressaoTotal := pressaoEstatica + pressaoDinamica

return pressaoTotal
}

func Termo_Fonte_TDG (tempo, i, j int, vel_media_bacia float64) {
var Schmidt, Kit, //Coeficiente de transferência de massa devido à turbulência
epsilon, // taxa de dispersão da energia cinética turbulenta
alpha_bolha_Num_bolha, // calcula o alpha bolha
Db_bolha float64 // diâmetro da bolha após o instante t+1
var N_bolha_minimo, N_bolha_maximo float64 // cálculo do número de bolha mínimo e máximo conforme o diâmetro da bolha
mínimo e máximo
//var pressao_profundidade float64 = (Elv_na_jusante - (float64(j))) * ro_agua
//var nx, ny int
//var nx int

Schmidt = visc_cine/DT

//nx = Comprimento_x
//ny = Comprimento_y

epsilon=0

if j < Lim_na_jus && Estatistica_dado_parafoam[i][j].AlphaMean_water <1.01 &&
Estatistica_dado_parafoam[i+1][j].AlphaMean_water <1.01 && Estatistica_dado_parafoam[i][j+1].AlphaMean_water <1.01 {

epsilon = visc_cine *
math.Pow(((Estatistica_dado_parafoam[i+1][j].Vel_media_x+Estatistica_bolha[i+1][j].Flutuacao_velocidade_x-
Estatistica_dado_parafoam[i][j].Vel_media_x-Estatistica_bolha[i][j].Flutuacao_velocidade_x)/Dimensao_malha_z)+
((Estatistica_dado_parafoam[i][j+1].Vel_media_z+Estatistica_bolha[i][j+1].Flutuacao_velocidade_z-
Estatistica_dado_parafoam[i][j].Vel_media_z-Estatistica_bolha[i][j].Flutuacao_velocidade_z)/Dimensao_malha_x),2)
}
if j >= Lim_na_jus && Estatistica_dado_parafoam[i][j].AlphaMean_water <1.01 &&
Estatistica_dado_parafoam[i+1][j].AlphaMean_water <1.01 && Estatistica_dado_parafoam[i][j-1].AlphaMean_water <1.01 { //
para fazer a condição de contorno na superfície da água
epsilon = visc_cine *
math.Pow(((Estatistica_dado_parafoam[i+1][j].Vel_media_x+Estatistica_bolha[i+1][j].Flutuacao_velocidade_x-
Estatistica_dado_parafoam[i][j].Vel_media_x-Estatistica_bolha[i][j].Flutuacao_velocidade_x)/Dimensao_malha_z)+
((Estatistica_dado_parafoam[i][j].Vel_media_z+Estatistica_bolha[i][j].Flutuacao_velocidade_z-
Estatistica_dado_parafoam[i][j-1].Vel_media_z-Estatistica_bolha[i][j-1].Flutuacao_velocidade_z)/Dimensao_malha_x),2)
}
if math.IsNaN(epsilon) {epsilon=0}

alpha_bolha_Num_bolha = 1-Estatistica_dado_parafoam[i][j].AlphaMean_water

if alpha_bolha_Num_bolha > 1.0 {alpha_bolha_Num_bolha= alpha_bolha_max}
if alpha_bolha_Num_bolha < 0.0 {alpha_bolha_Num_bolha= alpha_bolha_filtro}

N_bolha_minimo = (6*alpha_bolha_Num_bolha)/(math.Pi*math.Pow(Db_max,3))
N_bolha_maximo = (6*alpha_bolha_Num_bolha)/(math.Pi*math.Pow(Db_min,3))

if tempo == 1 {
Num_bolha_1[i][j] = (6*alpha_bolha_Num_bolha)/(math.Pi*math.Pow(Db_entrainment,3))
Num_bolha [i][j] = Num_bolha_1[i][j]
}

}

if i>1 {
Num_bolha_1[i][j]= Num_bolha[i][j]-
1/float64(tempo_maximo)*((Estatistica_dado_parafoam[i+1][j].Vel_media_x+Estatistica_bolha[i+1][j].Flutuacao_velocidade_x)*N

```

```

um_bolha[i+1][j]-(Estatistica_dado_parafoam[i-1][j].Vel_media_x+Estatistica_bolha[i-
1][j].Flutuacao_velocidade_x)*Num_bolha[i-1][j])/(2*Dimensao_malha_x)

    } else {
        Num_bolha_1[i][j]= Num_bolha[i][j]-
1/float64(tempo_maximo)*((Estatistica_dado_parafoam[i+1][j].Vel_media_x+Estatistica_bolha[i+1][j].Flutuacao_velocidade_x)*N
um_bolha[i+1][j]-
(Estatistica_dado_parafoam[i][j].Vel_media_x+Estatistica_bolha[i][j].Flutuacao_velocidade_x)*Num_bolha[i][j])/(Dimensao_malh
a_x)
    }
    //Num_bolha[i][j]= Num_bolha_1[i][j]
    if Num_bolha_1[i][j]< N_bolha_minimo {Num_bolha_1[i][j]=N_bolha_minimo}
    if Num_bolha_1[i][j]> N_bolha_maximo {Num_bolha_1[i][j]=N_bolha_maximo}
    Db_bolha = math.Pow(((6*alpha_bolha_Num_bolha)/(math.Pi*Num_bolha_1[i][j])),(1/3))

    Klt = 0.4*math.Pow(Schmidt,-0.5)*math.Pow(visc_cine*epsilon,0.25) // Equação 42
    if Estatistica_dado_parafoam[i][j].AlphaMean_water <1.01 {
        Result_Termo_Fonte_TDG[i][j]=
math.Pi*Num_bolha_1[i][j]*Db_entrainment*Db_entrainment*Klt*(((Estatistica_dado_parafoam[i][j].Pressao_mean+Sigma/(Db_b
olha/2))/Henry)/C_equilibrio-Concentracao_tdg_1[i][j]) // Equação 41 1000000 conforme equação 41

    } else {Result_Termo_Fonte_TDG[i][j]=0}

    if math.IsNaN (Result_Termo_Fonte_TDG[i][j]) {Result_Termo_Fonte_TDG[i][j]=0} // verifica se é um número válido, se
não for retorna 0

} // fim da função de cálculo do campo de concentração de TDG, conforme Equa 40, 41 e 42

// resolução da equação 40, os dados de entrada são: Estatistica_dado_parafoam{vel_media.x, alphaMean_water,
pressaomean}, estatisticabolha{velocidade_inst_x}

//func Calcula_Campo_Concentracao (estatistica_dado_parafoam, estatistica_bolha ) [] [] float64 {
func Calcula_Campo_Concentracao () {
    // func Calcula_Campo_Concentracao (estatistica_dado_parafoam TEstatistica_dado_parafoam, estatistica_bolha
TEstatistica_bolha) {
        var calc_concentracao float64 // distância/velocidade da bolha
        var i_inic, j_inic int // identificação da posição i e j com escoamento
        var arq_concentracao*os.File
        var arq_divergente*os.File
        var arq_Elv_241_246*os.File
        var err error
        var vel_media_bacia float64 = 0 // calcula a velocidade média do escoamento em toda a malha na direção x
        var vel_media_maxima_bacia [Array Comprimento_x] float64 // calcula a velocidade média das velocidades máximas em
cada célula do escoamento na direção x
        var soma_vel_maxima float64 = 0 // soma as velocidades máximas em no eio x para o cálculo da média
        var tempo_percorer_bacia float64 // tempo necessário para que uma partícula percorra toda a bacia
        var cont_vel_valida int = 0 // número de pontos com velocidade em x
        var cont_max_valido int = 0 // conta o número de velocidades máximas em x
        var inicio_dado_alfa_bacia int = 9999
        var fim_dado_alfa_bacia int = 0 // verifica o ponto em x que inicia dados com alfa e finaliza (inicio e fim dos dados em x da
bacia)
        var dt_bacia float64 // espaço de tempo da simulação
        var conta_variacao_1 int = 0 // conta quantas vezes houve variação do TDG maior que 1% entre uma célula e outra
        var variacao_TDG_TDG1 float64 // verifica a variação do TDG entre uma célula e outra
        var j_fundo int // posição em j do fundo da bacia/vertedouro/defletor

        fmt.Println("INICIO DA FUNÇÃO CALCULO_CAMPO_CONCENTRAÇÃO %5d %5d\n", Comprimento_x, Comprimento_y)

        i_inic = 9999
        j_inic = 9999

        diverg_al_ul_C_1 := func (i, j int) { // segundo termo da equação 40
            div_a = 0

            if j> int(math.Ceil(Elv_fundo_bacia)) && j < (Lim_na_jus) && Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j+1].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 &&
                Estatistica_dado_parafoam[i][j-1].AlphaMean_water < 1.01{
                    div_a = (Estatistica_dado_parafoam[i+1][j].AlphaMean_water *
Estatistica_dado_parafoam[i+1][j].Vel_media_x*Concentracao_tdg_1[i+1][j] -

```

```

    Estatistica_dado_parafoam[i-1][j].AlphaMean_water * Estatistica_dado_parafoam[i-
1][j].Vel_media_x*Concentracao_tdg_1[i-1][j])*(0.5/float64(Dimensao_malha))+
    (Estatistica_dado_parafoam[i][j+1].AlphaMean_water *
Estatistica_dado_parafoam[i][j+1].Vel_media_z*Concentracao_tdg_1[i][j+1] -
    Estatistica_dado_parafoam[i][j-1].AlphaMean_water * Estatistica_dado_parafoam[i][j-
1].Vel_media_z*Concentracao_tdg_1[i][j-1])*(0.5/float64(Dimensao_malha))

}

if j == int(math.Ceil(Elv_fundo_bacia)) && Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j+1].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 {

    div_a = (Estatistica_dado_parafoam[i+1][j].AlphaMean_water *
Estatistica_dado_parafoam[i+1][j].Vel_media_x*Concentracao_tdg_1[i+1][j] -
    Estatistica_dado_parafoam[i-1][j].AlphaMean_water * Estatistica_dado_parafoam[i-
1][j].Vel_media_x*Concentracao_tdg_1[i-1][j])*(0.5/float64(Dimensao_malha))+
    (Estatistica_dado_parafoam[i][j+1].AlphaMean_water *
Estatistica_dado_parafoam[i][j+1].Vel_media_z*Concentracao_tdg_1[i][j+1] -
    Estatistica_dado_parafoam[i][j].AlphaMean_water *
Estatistica_dado_parafoam[i][j].Vel_media_z*Concentracao_tdg_1[i][j])*(1/float64(Dimensao_malha))

}

if j >= (Lim_na_jus) && Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j-1].AlphaMean_water < 1.01{
    div_a = (Estatistica_dado_parafoam[i+1][j].AlphaMean_water *
Estatistica_dado_parafoam[i+1][j].Vel_media_x*Concentracao_tdg_1[i+1][j] -
    Estatistica_dado_parafoam[i-1][j].AlphaMean_water * Estatistica_dado_parafoam[i-
1][j].Vel_media_x*Concentracao_tdg_1[i-1][j])*(0.5/float64(Dimensao_malha))+
    (Estatistica_dado_parafoam[i][j].AlphaMean_water * Estatistica_dado_parafoam[i][j].Vel_media_z*Concentracao_tdg_1[i][j]
-
    Estatistica_dado_parafoam[i][j-1].AlphaMean_water * Estatistica_dado_parafoam[i][j-
1].Vel_media_z*Concentracao_tdg_1[i][j-1])*(1/float64(Dimensao_malha))
    // não há fluxo no sentido z na superfície livre
}

if math.IsNaN(div_a) {div_a = 0}
} // fim da função diverg_al_ul_C_1 := func (i, j int)

diverg_al_1_u := func (i, j int) { // terceiro termo da equação 24
    div_a_1 = 0

    if j > int(math.Ceil(Elv_fundo_bacia)) && j < (Lim_na_jus) && Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j+1].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 &&
    Estatistica_dado_parafoam[i][j-1].AlphaMean_water < 1.01{
        div_a_1 = (Estatistica_dado_parafoam[i+1][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i+1][j].AlphaMean_water)
*Estatistica_bolha[i+1][j].Flutuacao_velocidade_x-
        Estatistica_dado_parafoam[i-1][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i-1][j].AlphaMean_water)
*Estatistica_bolha[i-1][j].Flutuacao_velocidade_x)*(0.5/float64(Dimensao_malha))+
        (Estatistica_dado_parafoam[i][j+1].AlphaMean_water * (1-Estatistica_dado_parafoam[i][j+1].AlphaMean_water)
*Estatistica_bolha[i][j+1].Flutuacao_velocidade_x-
        Estatistica_dado_parafoam[i][j-1].AlphaMean_water * (1-Estatistica_dado_parafoam[i][j-1].AlphaMean_water)
*Estatistica_bolha[i][j-1].Flutuacao_velocidade_x)*(0.5/float64(Dimensao_malha))

    }

    if j == int(math.Ceil(Elv_fundo_bacia)) && Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j+1].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 {

        div_a_1 = (Estatistica_dado_parafoam[i+1][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i+1][j].AlphaMean_water)
*Estatistica_bolha[i+1][j].Flutuacao_velocidade_x-
        Estatistica_dado_parafoam[i-1][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i-1][j].AlphaMean_water)
*Estatistica_bolha[i-1][j].Flutuacao_velocidade_x)*(0.5/float64(Dimensao_malha))+
        (Estatistica_dado_parafoam[i][j+1].AlphaMean_water * (1-Estatistica_dado_parafoam[i][j+1].AlphaMean_water)
*Estatistica_bolha[i][j+1].Flutuacao_velocidade_x-
        Estatistica_dado_parafoam[i][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i][j].AlphaMean_water)
*Estatistica_bolha[i][j].Flutuacao_velocidade_x)*(1/float64(Dimensao_malha))

    }
}

```

```

if j >= (Lim_na_jus) && Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j-1].AlphaMean_water < 1.01{

    div_a_1 = (Estatistica_dado_parafoam[i+1][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i+1][j].AlphaMean_water)
*Estatistica_bolha[i+1][j].Flutuacao_velocidade_x-
    Estatistica_dado_parafoam[i-1][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i-1][j].AlphaMean_water)
*Estatistica_bolha[i-1][j].Flutuacao_velocidade_x)*(0.5/float64(Dimensao_malha))+
    (Estatistica_dado_parafoam[i][j].AlphaMean_water * (1-Estatistica_dado_parafoam[i][j].AlphaMean_water)
*Estatistica_bolha[i][j].Flutuacao_velocidade_x-
    Estatistica_dado_parafoam[i][j-1].AlphaMean_water * (1-Estatistica_dado_parafoam[i][j-1].AlphaMean_water)
*Estatistica_bolha[i][j-1].Flutuacao_velocidade_x)*(1/float64(Dimensao_malha))

    // não há fluxo no sentido z na superfície livre
}

if math.IsNaN(div_a_1) {div_a_1 = 0}
} // fim da função diverg_al_1_u := func (i, j int)

grad_conc_1 := func (i, j int) { // terceiro termo da equação 40
var grad_x float64 = 0
var grad_z float64 = 0
var visc_Scc float64 = (visc_cine + visc_turb/Schmidt)
//var cont int = 0

    grad_xz = 0

    if j > int(math.Ceil(Elv_fundo_bacia)) && j < (Lim_na_jus) && Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01
&& Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j+1].AlphaMean_water < 1.01
&&
    Estatistica_dado_parafoam[i][j-1].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 {
        grad_x = 1/float64(Dimensao_malha*Dimensao_malha) * visc_Scc *
(Estatistica_dado_parafoam[i+1][j].AlphaMean_water * Concentracao_tdg_1[i+1][j] -
2*Estatistica_dado_parafoam[i][j].AlphaMean_water * Concentracao_tdg_1[i][j] + Estatistica_dado_parafoam[i-
1][j].AlphaMean_water * Concentracao_tdg_1[i-1][j] )
        grad_z = 1/float64(Dimensao_malha*Dimensao_malha) * visc_Scc *
(Estatistica_dado_parafoam[i][j+1].AlphaMean_water * Concentracao_tdg_1[i][j+1] -
2*Estatistica_dado_parafoam[i][j].AlphaMean_water * Concentracao_tdg_1[i][j] + Estatistica_dado_parafoam[i][j-
1].AlphaMean_water * Concentracao_tdg_1[i][j-1] )
        grad_xz = (grad_x + grad_z)
    }

    if j == int(math.Ceil(Elv_fundo_bacia)) && Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i-1][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j+2].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j+1].AlphaMean_water < 1.01 {
        grad_x = 1/float64(Dimensao_malha*Dimensao_malha) * visc_Scc *
(Estatistica_dado_parafoam[i+1][j].AlphaMean_water * Concentracao_tdg_1[i+1][j] -
2*Estatistica_dado_parafoam[i][j].AlphaMean_water * Concentracao_tdg_1[i][j] + Estatistica_dado_parafoam[i-
1][j].AlphaMean_water * Concentracao_tdg_1[i-1][j] )
        grad_z = 1/float64(Dimensao_malha*Dimensao_malha) * visc_Scc *
(Estatistica_dado_parafoam[i][j+2].AlphaMean_water * Concentracao_tdg_1[i][j+2] -
2*Estatistica_dado_parafoam[i][j+1].AlphaMean_water * Concentracao_tdg_1[i][j+1] +
Estatistica_dado_parafoam[i][j].AlphaMean_water * Concentracao_tdg_1[i][j] ) // repete a última célula
        grad_xz = (grad_x + grad_z) // não há fluxo no sentido vertical - Condição de contorno para a fundo da bacia
    }

    if j >= (Lim_na_jus) && Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i+1][j].AlphaMean_water < 1.01 && Estatistica_dado_parafoam[i][j-1].AlphaMean_water < 1.01 &&
Estatistica_dado_parafoam[i][j-2].AlphaMean_water < 1.01{
        grad_x = 1/float64(Dimensao_malha*Dimensao_malha) * visc_Scc *
(Estatistica_dado_parafoam[i+1][j].AlphaMean_water * Concentracao_tdg_1[i+1][j] -
2*Estatistica_dado_parafoam[i][j].AlphaMean_water * Concentracao_tdg_1[i][j] + Estatistica_dado_parafoam[i-
1][j].AlphaMean_water * Concentracao_tdg_1[i-1][j] )
        grad_z = 1/float64(Dimensao_malha*Dimensao_malha) * visc_Scc * (Estatistica_dado_parafoam[i][j].AlphaMean_water
* Concentracao_tdg_1[i][j] - 2*Estatistica_dado_parafoam[i][j-1].AlphaMean_water * Concentracao_tdg_1[i][j-1] +
Estatistica_dado_parafoam[i][j-2].AlphaMean_water * Concentracao_tdg_1[i][j-2] )
        grad_xz = (grad_x + grad_z) // não há fluxo no sentido vertical - Condição de contorno para a superfície de água
    }

    if math.IsNaN(grad_xz) {grad_xz=0}

    // Condição de contorno para a fundo da bacia
} // fim da função grad_conc_1

```

```

// Condição inicial - Equilíbrio
for i := 1; i < Array_Comprimento_x-1; i++ {
    Elv_fundo_bacia = Inicializa_fundo_bacia (float64(i))
    vel_media_maxima_bacia[i] = 0
    for j := Array_Comprimento_z-1; j >= 1; j-- { // considera que o fluxo inicia de montante para jusante e de cima para baixo
        // condição inicial para o número de bolhas
        Num_bolha[i][j] = (6*(1-Estatistica_dado_parafoam[i][j].AlphaMean_water))/(math.Pi*math.Pow(Db_entrainment,3)) //
        equação 45
        Num_bolha_1[i][j] = Num_bolha[i][j]

        if Estatistica_dado_parafoam[i][j].AlphaMean_water <= 1.01 && Estatistica_dado_parafoam[i][j].AlphaMean_water > 0.6 {
            Concentracao_tdg[i][j] = C_equilibrio // considera que o primeiro ponto do escoamento está em equilíbrio
            Concentracao_tdg_1[i][j] = C_equilibrio // considera que o primeiro ponto do escoamento está em equilíbrio
            Result_Divergence_al_ul_C[i][j] = 0
            Result_Divergence_visc_al_C[i][j] = 0
            Result_Termo_Fonte_TDG[i][j] = 0
            vel_media_bacia += Estatistica_dado_parafoam[i][j].Vel_media_x
            cont_vel_valida += 1

            if vel_media_maxima_bacia[i] < Estatistica_dado_parafoam[i][j].Vel_media_x {
                vel_media_maxima_bacia[i] = Estatistica_dado_parafoam[i][j].Vel_media_x
            }

            if inicio_dado_alfa_bacia > i {inicio_dado_alfa_bacia = i}
            if fim_dado_alfa_bacia < i {fim_dado_alfa_bacia = i}
        } else {
            Concentracao_tdg[i][j] = C_equilibrio
            Concentracao_tdg_1[i][j] = C_equilibrio
            Result_Divergence_al_ul_C[i][j] = 0
            Result_Divergence_visc_al_C[i][j] = 0
            Result_Termo_Fonte_TDG[i][j] = 0
        }

        if Estatistica_dado_parafoam[i][j].AlphaMean_water != 0 && Estatistica_dado_parafoam[i][j].AlphaMean_water < 1.01 {
            if i < i_inic {i_inic = i}
            if j < j_inic {j_inic = j}
        }
    } // fim do for j
    if vel_media_maxima_bacia[i] > 0 {
        soma_vel_maxima += vel_media_maxima_bacia[i]
        cont_max_valido += 1

        if i > 1 && vel_media_maxima_bacia[i-1] > 0 {
            vel_media_maxima_bacia[i] = (vel_media_maxima_bacia[i-1]*float64(cont_max_valido-1) + vel_media_maxima_bacia[i])/float64(cont_max_valido)
        }
    }
} // fim do for i

vel_media_bacia = vel_media_bacia/float64(cont_vel_valida)
tempo_percorer_bacia = float64(fim_dado_alfa_bacia-inicio_dado_alfa_bacia)*Dimensao_malha_x/vel_media_bacia
Tempo_percorer_bacia = tempo_percorer_bacia
dt_bacia = tempo_percorer_bacia/float64(tempo_maximo)
//vel_media_maxima_bacia = soma_vel_maxima/float64(cont_max_valido)

// impressão apenas para testes
fmt.Printf("O primeiro ponto com escoamento é %d %d\n", i_inic, j_inic) // apenas para teste
fmt.Printf("A velocidade média do escoamento é %2.5f O tempo para percorrer a bacia é %2.5f dt bacia = %2.5f\n", velmedia_maxima, vel_media_bacia, tempo_percorer_bacia, dt_bacia, soma_vel_maxima/float64(cont_max_valido))
// apenas para teste

// imprime o arquivo com os dados de TDG
arq_concentracao, err = os.Create("concentracao_TDG_test.txt")
if err != nil {
    fmt.Println("Erro ao criar o arquivo:", err)
}
defer arq_concentracao.Close()

arq_divergente, err = os.Create("Divergente.txt")
if err != nil {
    fmt.Println("Erro ao criar o arquivo:", err)
}

```

```

}
defer arq_divergente.Close()

arq_Elv_241_246, err = os.Create("TDG_Elv_241_246.txt")
if err != nil {
    fmt.Println("Erro ao criar o arquivo:", err)
}
defer arq_concentracao.Close()

cabecalho := []string{ // cabeçalho do arquivo arc_concentração
    "Tempo", "Posição x", "Posição y", "concentracao", "alpha_mean", "vel_media_x", "vel_media_z", "pressao_mean", "TDG_P&D",
}

for _, header := range cabecalho {
    fmt.Fprintf(arq_concentracao, "%40s", header)
}
fmt.Fprintf(arq_concentracao, "\r\n")

cabecalho_divergente := []string{
    "Tempo", "Posição x", "Posição y", "Divergence_al_ul_C", "Termo_Fonte_TDG", "Divergence_visc_al_C",
    "Divergence_al_ul_C_1", "Termo_Fonte_TDG_1", "Divergence_visc_al_C_1", "calc_concentracao", "Concentração",
}

for _, header := range cabecalho_divergente {
    fmt.Fprintf(arq_divergente, "%40s", header)
}
fmt.Fprintf(arq_divergente, "\r\n")

cabecalho_Elv_241_246 := []string{ // cabeçalho arq_Elv_241_246
    "Tempo", "Posição x", "concentracao z=241", "concentracao z=246", "alpha_mean z=241", "alpha_mean z=246",
    "vel_media_x z=241",
    "vel_media_x z=246", "vel_media_z z=241", "vel_media_z z=246", "pressao_mean z=241", "pressao_mean z=246", "TDG_P&D
z=241", "TDG_P&D z=246",
}

for _, header := range cabecalho_Elv_241_246 {
    fmt.Fprintf(arq_Elv_241_246, "%40s", header)
}
fmt.Fprintf(arq_Elv_241_246, "\r\n")

// inicio do cálculo para o tempo maior que 2 - no tempo igual a 1 tudo está em equilíbrio
for tempo := 2; tempo <= tempo_maximo; tempo ++ {

    // Subdivisão da malha de 1m x 1m dos dados fonte em sub-malha (n° div_malha)
    for i:=2; i<int(Array_Comprimento_x)-1; i++ {
        Elv_fundo_bacia = Inicializa_fundo_bacia(float64(i)/Dimensao_malha_x)
        j_fundo = int(math.Ceil((Inicializa_fundo_bacia(float64(i)*Dimensao_malha_x) - Min_na_jus)/Dimensao_malha_z))
        //var j_fundo int
        //j_fundo = int(math.Ceil(Elv_fundo_bacia))
        //for j:=1; j<int((float64(Lim_na_jus)-Min_na_jus)/Dimensao_malha_z); j++{
        for j:=j_fundo; j<Array_Comprimento_z; j++{

            // verifica se existe dado ou é vazio e atribui valores aos limites da submalha
            if Estatistica_dado_parafoam[i][j].AlphaMean_water <= 1.01 {

                grad_xz = 0 // resultado de grad_conc
                div_a = 0 // resultado de diverg_al_ul_C
                soma_div_a = 0
                soma_grad_xz = 0
                calc_concentracao = 0
                grad_conc_1(i, j) // Result_Divergence_visc_al_C[i][j]
                diverg_al_ul_C_1(i, j) // Result_Divergence_al_ul_C[i][j]
                diverg_al_1_u(i, j) // div_a_1
                Termo_Fonte_TDG(tempo, i, j, vel_media_maxima_bacia[i]) // calcula para todos os pontos da malha - depende apenas
na Concentracao_tdg_1[i][j]

                Result_Divergence_al_ul_C[i][j] = div_a
                Result_Divergence_visc_al_C[i][j] = grad_xz

                calc_concentracao = (Result_Divergence_visc_al_C[i][j] + Result_Termo_Fonte_TDG[i][j] -
Result_Divergence_al_ul_C[i][j])/Estatistica_dado_parafoam[i][j].AlphaMean_water

                if Estatistica_dado_parafoam[i][j].AlphaMean_water < 0.05 { // filtro para alfa < 0.5 - No P&D era 0.5
                    calc_concentracao = 0
                }
                if math.IsNaN(calc_concentracao) { calc_concentracao = 0 } // verifica se é um número válido, se não for retorna 0
            }
        }
    }
}

```

```

Concentracao_tdg [i][j] = Concentracao_tdg_1[i][j] + calc_concentracao*(1/float64(tempo_maximo))

if tempo>3 && i> 1{

    variacao_TDG_TDG1 = (Concentracao_tdg[i][j]-Concentracao_tdg[i-1][j])
    // Regras para suavização das variações de TDG entre uma célula e outra

    if math.Abs(variacao_TDG_TDG1)> 0.1*C_equilibrio {
        variacao_TDG_TDG1 = math.Copysign(0.1*C_equilibrio ,variacao_TDG_TDG1)
    }

    conta_variacao_1 +=1

    Concentracao_tdg_1[i][j] = Concentracao_tdg[i][j] // utilizado no cálculo do próximo instante de tempo

} // fim do if tempo>50 {

    if math.IsNaN (Concentracao_tdg [i][j]) {Concentracao_tdg [i][j]=C_equilibrio} // verifica se é um número válido, se não for
    retorna 0
    //if Concentracao_tdg [i][j] < 0 {Concentracao_tdg[i][j] = C_equilibrio}/0.9*C_equilibrio}
    if Concentracao_tdg [i][j] < 0.9*C_equilibrio {Concentracao_tdg[i][j] = 0.9*C_equilibrio}/0.9*C_equilibrio}
    if Concentracao_tdg [i][j]> (C_max*C_equilibrio) {Concentracao_tdg [i][j] = C_max*C_equilibrio}

} else {
    Concentracao_tdg[i][j] = C_equilibrio
    if Concentracao_tdg[i-1][j]>0 && Concentracao_tdg[i-1][j]<C_max*C_equilibrio
    {Concentracao_tdg[i][j]=Concentracao_tdg[i-1][j]}
    Concentracao_tdg_1[i][j] = Concentracao_tdg[i][j] // utilizado no cálculo do próximo instante de tempo
}

// imprime em arquivo os resultados do programa

if i%50 == 0 && j%10 ==0{
    fmt.Fprintf(arq_divergente,"\r\n %40d %40.2f %40.2f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f ",
    tempo, float64(i)*Dimensao_malha_x, float64(j)*Dimensao_malha_z+Min_na_jus, Result_Divergence_al_ul_C[i][j],
    Result_Termo_Fonte_TDG[i][j],Result_Divergence_visc_al_C[i][j],
    Result_Divergence_al_ul_C_1[i][j], Result_Termo_Fonte_TDG_1[i][j],Result_Divergence_visc_al_C_1[i][j],
    calc_concentracao, Concentracao_tdg [i][j])
}

//if i%20 == 0 && tempo%50 == 0 { // imprime os dados no arquivo
if tempo == tempo_maximo {
    if Estatistica_dado_parafoam[i][j].AlphaMean_water<1.01 && Estatistica_dado_parafoam[i][j].AlphaMean_water>0.0001 {
        fmt.Fprintf(arq_concentracao,"\r\n %40d %40.2f %40.2f %40.6f %40.6f %40.6f %40.6f %40.6f %40.6f ", tempo,
        float64(i)*Dimensao_malha_x, (float64(j)*Dimensao_malha_z+Min_na_jus),
        Concentracao_tdg[i][j], Estatistica_dado_parafoam[i][j].AlphaMean_water,
        Estatistica_dado_parafoam[i][j].Vel_media_x,
        Estatistica_dado_parafoam[i][j].Vel_media_z,
        Estatistica_dado_parafoam[i][j].Pressao_mean,Estatistica_dado_parafoam_tdg[i][j].Tdg_mc)
    }
}

Result_Divergence_al_ul_C_1[i][j] = Result_Divergence_al_ul_C[i][j]
Result_Termo_Fonte_TDG_1[i][j] = Result_Termo_Fonte_TDG[i][j]
Result_Divergence_visc_al_C_1[i][j] = Result_Divergence_visc_al_C[i][j]

} // fim do loop j

// Imprime os dados na Elv 241 e 246 para comparativo ao longo de x
//if tempo == tempo_maximo {
if tempo%2000 == 0 {

    var j_241, j_246, num_valido int

    j_241 = int (math.Ceil(float64(241)- Min_na_jus)/Dimensao_malha_z)-1 // -1 é para localizar o array que inicia em 0
    j_246 = int (math.Ceil(float64(246)- Min_na_jus)/Dimensao_malha_z)-1

    var concentracao_241, concentracao_246, alpha_mean_241,alpha_mean_246, vel_media_x_241,
    vel_media_x_246,vel_media_z_241,vel_media_z_246,pressao_mean_241,pressao_mean_246, TDG_PD_241,
    TDG_PD_246 float64

    concentracao_241 = 0
    concentracao_246 = 0

```

```

alpha_mean_241 = 0
alpha_mean_246 = 0
vel_media_x_241 = 0
vel_media_x_246 = 0
vel_media_z_241 = 0
vel_media_z_246 = 0
pressao_mean_241 = 0
pressao_mean_246 = 0
TDG_PD_241 = 0
TDG_PD_246 = 0
num_valido = 0

for k := j_241; k <= j_241+int(1.0 /Dimensao_malha_z); k++ {
    z := (j_246-j_241)

    if Estatistica_dado_parafoam_tdg[i][k].Tdg_mc <=3 {
        concentracao_241 += Concentracao_tdg[i][k]
        concentracao_246 += Concentracao_tdg[i][k+z]
        alpha_mean_241 += Estatistica_dado_parafoam[i][k].AlphaMean_water
        alpha_mean_246 += Estatistica_dado_parafoam[i][k+z].AlphaMean_water
        vel_media_x_241 += Estatistica_dado_parafoam[i][k].Vel_media_x
        vel_media_x_246 += Estatistica_dado_parafoam[i][k+z].Vel_media_x
        vel_media_z_241 += Estatistica_dado_parafoam[i][k].Vel_media_z
        vel_media_z_246 += Estatistica_dado_parafoam[i][k+z].Vel_media_z
        pressao_mean_241 += Estatistica_dado_parafoam[i][k].Pressao_mean
        pressao_mean_246 += Estatistica_dado_parafoam[i][k+z].Pressao_mean
        TDG_PD_241 += Estatistica_dado_parafoam_tdg[i][k].Tdg_mc
        TDG_PD_246 += Estatistica_dado_parafoam_tdg[i][k+z].Tdg_mc
        num_valido +=1
    } // fim do if if Estatistica_dado_parafoam_tdg[i][j_241+k].Tdg_mc <=3
} //fim do for k

if num_valido > 0 {
    concentracao_241 = concentracao_241/float64(num_valido)
    concentracao_246 = concentracao_246/float64(num_valido)
    alpha_mean_241 = alpha_mean_241/float64(num_valido)
    alpha_mean_246 = alpha_mean_246/float64(num_valido)
    vel_media_x_241 = vel_media_x_241/float64(num_valido)
    vel_media_x_246 = vel_media_x_246/float64(num_valido)
    vel_media_z_241 = vel_media_z_241/float64(num_valido)
    vel_media_z_246 = vel_media_z_246/float64(num_valido)
    pressao_mean_241 = pressao_mean_241/float64(num_valido)
    pressao_mean_246 = pressao_mean_246/float64(num_valido)
    TDG_PD_241 = TDG_PD_241/float64(num_valido)
    TDG_PD_246 = TDG_PD_246/float64(num_valido)

    fmt.Fprintf(arq_Elv_241_246,"\r\n %40d %40.2f %40.6f %40.6f %40.6f %40.6f %40.4f %40.4f %40.4f %40.4f %40.1f
%40.1f %40.6f %40.6f ", tempo, float64(i)*Dimensao_malha_x,
    concentracao_241, concentracao_246, alpha_mean_241, alpha_mean_246, vel_media_x_241, vel_media_x_246,
    vel_media_z_241, vel_media_z_246, pressao_mean_241,
    pressao_mean_246, TDG_PD_241, TDG_PD_246)

    } // fim do if num_valido > 0

} // fim fo if tempo == tempo_maximo

} // fim fo loop i

// imprime o status da simulação em tela
if tempo %100==0 {
    fmt.Printf("Calculo do tempo: %d de %d com implementação de pace máximos de variação de TDG %5d\n", tempo,
    tempo_maximo, conta_variacao_1 )
}

for i:=2; i< int(Array_Comprimento_x)-1; i++ {
    for j:=j_fundo ; j<Array_Comprimento_z; j++){
        Num_bolha[i][j] = Num_bolha_1[i-1][j]
    }
}
} // fim fo for do tempo simulado
arq_concentracao.Close()
arq_divergente.Close()
arq_Elv_241_246.Close()

} // fim da função func Calcula_Campo_Concentracao ()

```