COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE MÉTODOS NUMÉRICOS USANDO PLACAS GRÁFICAS PARA A SOLUÇÃO DA EQUAÇÃO DE BALANÇO POPULACIONAL

Fabio Pereira dos Santos

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Química, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Química.

Orientadores: Paulo Laranjeira da Cunha Lage
Inanc Senocak

Rio de Janeiro
Março de 2014

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE MÉTODOS NUMÉRICOS USANDO PLACAS GRÁFICAS PARA A SOLUÇÃO DA EQUAÇÃO DE BALANÇO POPULACIONAL

Fabio Pereira dos Santos

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA QUÍMICA.

Examinada por:

Prof. Paulo Laranjeira da Cunha Lage, D.Sc.

Prof. Argimiro Resende Secchi, D.Sc.

Prof. Reinaldo Giudici, D.Sc.

Prof. Alvaro Luiz Gayoso de Azeredo Coutinho, D.Sc.

Prof. Luiz Fernando Lopes Rodrigues Silva, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2014

Santos, Fabio Pereira dos

Desenvolvimento e Implementação de métodos numéricos usando placas gráficas para a solução da equação de balanço populacional/Fabio Pereira dos Santos. – Rio de Janeiro: UFRJ/COPPE, 2014.

XIX, 257 p.: il.; 29, 7cm.

Orientadores: Paulo Laranjeira da Cunha Lage

Inanc Senocak

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Química, 2014.

Referências Bibliográficas: p. 110 – 124.

1. Cubatura Adaptativa. 2. Balanço Populacional. 3. GPUs. 4. CUDA. 5. Solução da Equação de Balanço Populacional. 6. OpenFOAM®. I. Lage, Paulo Laranjeira da Cunha *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Química. III. Título.

À minha família, Edson, Luana
e Regina

Agradecimentos

Primeiramente gostaria de agradecer à minha família, por ser meu porto seguro, que acredita e apoia meus sonhos. Ao meu pai, Edson, pelos conselhos, à minha mãe, Regina, pela fé e à minha irmã, Luana, pela paciência e amizade. Agradeço pelo suporte e compreensão quanto à minha escolha profissional.

Ao Prof. Dr. Paulo Laranjeira da Cunha Lage, pela sua valiosa orientação. Obrigado também pela amizade e pelos conselhos, advertências e correções que foram fundamentais para decisão do meu futuro acadêmico.

I would like to thank Prof. Inanc Senocak for his contribution to this present work. It was a pleasure meeting you. Thank you for your friendship and warnings which are very important to my academic future.

Agradeço aos integrantes do LTFD: João Mitre, Gabriel, Livia, Jovani, Antonio Neto, Diogo, Luiz Felipe, Thaína, Tatiana Pitchon, Samel e Thais.

Agradeço também aos meus amigos de forrozinho, muito importantes nos momentos de descontração: Luis Rivera (peruano), Marina (queridona), Terreirada Cearense, Bianca Pires (Bibi), Élida e Jana.

I would like to thank my friends from Boise, Idaho. They really received me with open arms. I would like to especially thank my friends from Boise Casino Rueda: Jonas, Jessie and Fil. Thanks my good friends: Molly and Maurício. Also, my Persian-Tajik friend Anis.

Agradeço aos amigos que cultivei em todos os lugares que passei, pelas piadas e amizade. Aos amigos da Escola de Química: Rafael e Fernanda Bertges, Alexandre França, Livia Moreira, Thiago Judson e Tânia Klein. Aos amigos que mesmo distantes, estão sempre presentes: Rita, Ricardo Damian e Ammar. Muito importantes para construção do meu caráter, pelos momentos de descontração e felicidade.

Aos meus amigos de boêmia e carnaval: Diogo Coelho, Luis Rivera (peruano), Priscilla Caldellas, Jovani e Hyago (primo).

Agradeço aos membros da banca, Prof. Argimiro Secchi, Prof. Luiz Fernando L. R Silva, Prof. Reinaldo Giudici e Prof. Alvaro Luiz Gayoso de Azeredo Coutinho, pelo apoio e sugestões para o desenvolvimento desse trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE MÉTODOS NUMÉRICOS USANDO PLACAS GRÁFICAS PARA A SOLUÇÃO DA EQUAÇÃO DE BALANÇO POPULACIONAL

Fabio Pereira dos Santos

Março/2014

Orientadores: Paulo Laranjeira da Cunha Lage
Inanc Senocak

Programa: Engenharia Química

Escoamentos multifásicos polidispersos estão presentes em várias processos industriais. Hoje em dia, é consenso de que o acoplamento entre balanço populacional e dinâmica de fluidos computacional pode ser usado para prever este tipo de escoamento. Neste trabalho, o método D^{2}uQMoGeM foi formulado visando a solução da equação de populacional populacional (PBE). Neste método, os pesos e as abscissas são calculados diretamente como no DQMoM e os erros de quadratura são controlados por quadratura adaptativa como no DuQMoGeM. O D^{2}uQMoGeM foi implementado e testado para vários problemas diferentes com soluções analíticas bem estabelecidas. Este se mostrou ser mais preciso do que DQMoM mas gerou um aumento razoável no tempo computacional.

Como no DuQMoGeM, a desvantagem do D^{2}uQMo GeM é seu alto custo computacional associado às integrações numéricas dos termos integrais da PBE. Felizmente, cada integrando pode ser integrado de forma independente e, por isso, passíveis de paralelização em GPUs. Dois algoritmos de cubatura adaptativa e o algoritmo SVD de solução de sistemas lineares foram implementados em GPUs visando acelerar os chamados métodos dos momentos com quadratura dupla (DQBMMs). Finalmente, a solução da PBE foi conduzida com os DQBMMs, em sua forma totalmente conservativa, juntamente com o modelo multi-fluido no OpenFOAM®. Simulações numéricas considerando quebra e agregação foram conduzidas visando avaliar o acoplamento. A fim de superar sua limitação computacional, os DQBMMs fizeram uso dos algoritmos paralelos em GPUs aqui implementados, o que tornou as simulações factíveis em um tempo computacional tratável.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

DEVELOPMENT AND IMPLEMENTATION OF NUMERICAL METHODS ON GPUS FOR SOLVING THE POPULATION BALANCE PROBLEMS

Fabio Pereira dos Santos

March/2014

Advisors: Paulo Laranjeira da Cunha Lage
Inanc Senocak

Department: Chemical Engineering

Polydisperse multiphase flows are present in numerous industrial technologies. Nowadays, it is consensus that the population balance coupled to computational fluid dynamics can be used to predict this sort of flow. In this work, the D^{2}uQMoGeM was formulated for the solution of the population balance equation. In this method, the weights and abscissas are tracked directly as in DQMoM and the quadrature errors are controlled by an adaptive quadrature as in DuQMoGeM. The D^{2}uQMoGeM was implemented and tested for several different problems with analytical solutions. It was shown to be more accurate than the DQMoM but with a reasonable increase in computational time.

As in DuQMoGeM, the drawback of D^{2}uQMoGeM is the high computational cost associated with numerical integrations of the PBE integral terms. Fortunately, each integrand can be integrated independently and, therefore, amenable to parallelization on GPUs. Two parallel adaptive cubature algorithms and a SVD algorithm were implemented on a hybrid architecture to accelerate the DuQMoGeM and the D^{2}uQMoGeM that are called as dual-quadrature-based moment methods (DQBMMs). Finally, the solution of the population balance equation was performed with the DQBMMs in their fully conservative forms together with the multi-fluid model in OpenFOAM®. Numerical simulations considering breakage and aggregation were conducted. Again, the main disadvantage of these methods is their high computational cost. In order to address this computational limitation, the dual-quadrature-based moment methods were parallelized on graphics processing units what resulted in a significant acceleration, and made the simulation feasible in a tractable computational time.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

$\mathbf{A}$	Vetor de integrais do termo de agregação do DuQMoGeM
a	Frequência de coalescência
b	Frequência de quebra
$blockDim.x$	Dimensão do block na direção x
$blockIdx.x$	Identificador do block em um *grid*
c_i	Coeficientes de uma expansão funcional
d	Diâmetro de partícula
$\mathbf{D_r}$	Coeficiente de difusão anisotrópico no espaço de variáveis internas
$\mathbf{D_x}$	Coeficiente de difusão anisotrópico no espaço de variáveis externas
E	Métrica de eficiência
f	Função de distribuição densidade numérica
$\mathbf{g}$	Aceleração gravitacional
G	Termo de crescimento
H	Termo fonte de quebra e coalescência
Idx	Identificador do thread em um *grid*
J	Termo de nucleação
$\mathbf{L}$	Vetor de integrais do termo de quebra do DuQMoGeM
$\mathbf{m}$	Variável interna aditiva
$M_{\alpha,\beta}$	Transferência de quantidade de movimento entre as fases α e as fases β;
N_T	Número médio total de partículas no domínio

p Pressão

$\mathbf{r}$ Vetor de variáveis internas

r Fração de fase

S Aceleração da aplicação serial (*Speed up*)

t Tempo

T Tempo de execução

$threadIdx.x$ Identificador do thread em um block

$\mathbf{u}$ Velocidade

$\mathbf{x}$ Vetor de variáveis externas

$\mathbf{Y}$ Vetor das variáveis da fase contínua

$\dot{\mathbf{R}}$ Velocidade no espaço de variáveis internas

$\dot{\mathbf{X}}$ Velocidade no espaço de variáveis externas

Letras Gegras

δ Função delta de Dirac

Γ Termo fonte mássico

γ Fração volumétrica de partículas

μ Momento de uma distribuição

ν Número de partículas formadas na quebra

Ω Domínio das variáveis externas e internas

ω Peso da quadratura de Gauss-Christoffel

ϕ Base de polinômios ortogonais

ρ Massa específica

θ Fração serial de uma aplicação paralela

υ Viscosidade cinemática

ς Abscissa ponderada

Sobrescritos

ϕ	Relativo a momento generalizado
a	Relativo a solução analítica

Subscritos

a	Relativo a coalescência
b	Relativo a quebra
eff	Relativo a propriedade efetiva turbulenta
k	Ordem do momento
max	Valor máximo
min	Valor mínimo
p	Número de processos
s	Relativo a aplicação serial

Lista de Abreviaturas

D^2uQMoGeM	*Direct Dual-Quadrature Method of Generalized Moments*, p. 3
BFS	*Backward Facing Step*, p. 97
CFD	*Computational Fluid Dynamics*, p. 3
CNMC	*Constant Number Monte Carlo*, p. 18
CPU	*Central Processing Unit*, p. 4
CUDA	*Compute Unified Device Architecture*, p. 4
DSMC	*Direct Simulation Monte Carlo*, p. 18
DWMC	*Differentially weighted Monte-Carlo*, p. 18
DuQMoGeM	*Dual-Quadrature Method of Generalized Moments*, p. 3
EADs	Equações algébrico diferenciais, p. 80
FCMoM	*Finite size domain Complete set of trial functions Method Of Moments*, p. 34
FDN	Função Densidade Numérica, p. 23
FDN	Função Densidade Numérica, p. 7
GPU	*Graphics Processing Unit*, p. 4
LAG	Polinômio de Laguerre, p. 33
MC	Método das Classes, p. 19
MIMD	*Multiple Instruction Multiple Data*, p. 51
MISD	*Multiple Instruction Single Data*, p. 51
MMC	*Multi-Monte Carlos*, p. 18
MRP	Método dos Resíduos Ponderados, p. 23

MUSIG	*Multiple Size Group*, p. 3
MoM	*Method of Moments*, p. 23
OpenFOAM	*Open source Field Operation And Manipulation*, p. 3
PEMC	*Point Ensemble Monte Carlo*, p. 18
PIFL	Parallelism at the Integral Formula Level, p. 72
PIL	Parallelism at the Integrand Level, p. 72
PPDC	*Parallel Parent Daughter Classes*, p. 3
QMoM	*Quadrature Method of Moments*, p. 3
SCVMC	*Stepwise Constant Volume Monte Carlo*, p. 18
SIMD	*Single Instruction Multiple Data*, p. 51
SISD	*Single Instruction Single Data*, p. 51
SQMoM	*Sectional Quadrature Method of Moments*, p. 33
TMJ	Transformação da Matriz Jacobiana, p. 29

Capítulo 1

Introdução e Objetivos

1.1 Contexto

Escoamentos multifásicos referem-se a sistemas que consistem de pelo menos duas fases distintas. Como esses sistemas estão presentes na maioria dos processos químicos, a comunidade científica vem desenvolvendo formas de prever seu comportamento [3–9]. Não obstante, descrever esses sistemas ainda é um desafio para nós cientistas e pesquisadores, tanto no desenvolvimento de métodos numéricos quanto na modelagem física.

1.1.1 Escoamentos multifásicos Polidispersos

Sistemas multifásicos podem ser classificados de acordo com a morfologia de sua interface [10]. Baseado nessa forma de classificação, os processos multifásicos dispersos são caracterizados pela presença de, pelo menos, uma fase dispersa, tais como gotas em um gás ou bolhas em um líquido [11, 12]. Eles são dito polidispersos quando essas partículas tem uma ou mais propriedades que diferem entre si e afetam o escoamento [11].

Muitas tecnologias se fundamentam nas propriedades de uma polidispersão, fato que impulsiona o desenvolvimento de técnicas numéricas capazes de descrever o comportamento dinâmico da população de partículas. Técnicas que visam auxiliar engenheiros a prever e controlar o comportamento de partículas em um dado equipamento. Na indústria química, pode-se citar os seguintes exemplos de equipamentos:

- Separadores ciclônicos;
- *Spray drying;*[1]
- Leitos fluidizados;

[1] Pulverização de uma solução líquida em uma corrente de ar quente para a vaporização do solvente.

- Flotadores;
- Sedimentadores;
- Colunas de borbulhamento;
- Reatores químicos multifásicos;
- Sistemas de extração.

Em sua maioria, os fenômenos associados aos processos de engenharia química dependem significativamente das propriedades das partículas de um sistema polidisperso. Um ótimo exemplo de equipamento multifásico polidisperso é reator de coluna de borbulhamento, onde sua fase dispersa, formada por bolhas, afeta sua eficiência, já que os fenômenos de transferência associados a área interfacial da fase dispersa é um dos parâmetros chaves para projeto desse equipamento. Neste exemplo, a distribuição de tamanho de partícula controla os fenômenos de transferência de massa, energia e quantidade de movimento, que por sua vez, afetam a temperatura e a concentração dos componentes nesse sistema [13].

Visando descrever o comportamento de uma distribuição de partículas, surge a teoria de balanço populacional (PB) [14]. Devido a sua vasta aplicabilidade, o estudo dessa teoria e dos métodos de solução da equação de balanço populacional vem sendo realizado com grande afinco pela comunidade acadêmica[2].

É importante realçar que, muitas vezes, o tratamento da equação de balanço populacional está conectado com a solução do escoamento do sistema multifásico. Deste modo, o acoplamento entre PB e CFD (*Computational Fluid Dynamics*) é considerado de vital importância para os problemas reais de engenharia.

1.2 Relevância no Grupo de Pesquisa

A principal linha de pesquisa do Laboratório de Termofluidodinâmica (LTFD) do Programa de Engenharia Química da COPPE/UFRJ é o desenvolvimento de métodos numéricos e de modelos fenomenológicos para a análise teórico-experimental de escoamentos multifásicos polidispersos.

O LTFD tem focado na análise experimental e modelagem de escoamentos gás-líquido [15–17], evaporadores por contato direto em processos de borbulhamento [18–20], [21, 22], escoamentos multifásicos estratificados [23], escoamento em meios porosos [24] e escoamentos multifásicos polidispersos [3, 20, 21, 25–28].

[2] 478 artigos nos últimos 5 anos apenas na *Chemical Engineering of Science* (levantamento da ISI Web of Knowledge com palavra chave a *Population Balance*)

Entre os trabalhos mais recentes estão o desenvolvimento de modelos de quebra e agregação [28], os métodos numéricos híbridos denominados DuQMoGeM (*Dual-Quadrature Method of Generalized Moments*) [29, 30] e Direct DuQMoGeM [31], e o acoplamento multi-fluido incompressível PB-CFD no OpenFOAM® [3].

Nessa linha, atualmente, o LTFD vem desenvolvendo um *Solver*, denominação para os códigos desenvolvidos no OpenFOAM® (código aberto de CFD), multifásico polidisperso com fases incompressíveis para simular o escoamento [32]. Nesse contexto, o presente trabalho almeja desenvolver metodologias numéricas para simulação de escoamentos multifásicos polidispersos, se inserindo assim no cenário atual de pesquisa do Laboratório de Termofluidodinâmica.

1.3 Motivação e Objetivo

1.3.1 Motivação

Nos últimos anos, a indústria do petróleo vem crescendo e alavancando a economia brasileira. Para que esse crescimento seja continuado, existe a necessidade de constante renovação tecnológica. Por esta razão, as companhias de petróleo, em especial a PETROBRAS, Petróleo Brasileiro S/A, têm se dedicado à pesquisa e ao desenvolvimento de técnicas de obtenção e beneficiamento de petróleo. Em especial, pela necessidade de exploração de poços de petróleo localizados no pré-sal. Outro aspecto relevante, que é destaque e preocupa a sociedade, é o meio ambiente. Tal preocupação tem sido constantemente notícia na mídia mundial, levando à tona também aspectos políticos e econômicos. Nesse contexto, as empresas petrolíferas devem se adequar e desenvolver novas tecnologias que sigam as novas tendências mundiais.

O emprego de CFD vem se tornando mais popular na indústria, extrapolando os limites dos laboratórios de pesquisa acadêmica e ganhando vida nos projetos de engenharia. A expansão da computação de alto desempenho e o desenvolvimento de técnicas numéricas rápidas, precisas e robustas são os principais motivos pelos quais CFD se tornou uma ferramenta de projeto. Por isto, hoje, é factível simular escoamentos complexos com um grau de detalhamento inviável há algumas décadas atrás.

Na PETROBRAS, o uso de CFD crescia em torno de 40% ao ano em 2007 (DAMIAN [27]), dado que evidenciou o interesse por essa tecnologia. Atualmente, o volume de aplicações de CFD nas empresas é vasto, atuando desde bioengenharia até engenharia básica.

Neste sentido, algumas técnicas como acoplamento entre balanço populacional e CFD vem ganhando espaço, exigindo assim o desenvolvimento de técnicas numéricas

mais precisas, robustas e velozes.

1.3.2 Novas Perspectivas em computação científica

O desenvolvimento da ciência é uma das forças motrizes para o rápido crescimento do poder computacional. As arquiteturas computacionais vêm se desenvolvendo em duas direções. Sendo uma direção focada no aumento da frequência com que as operações são realizadas (*clocks*) e a outra no aumento do número de núcleos de computação. Infelizmente, a frequência nos processadores tem se mantido constante devido a relação cúbica entre o aumento do consumo de energia e o *clock* [33]. Em suma, o custo de energia associado ao aumento da frequência não é justificável. Nesse sentido, uma alternativa seria reduzir essa frequência e aumentar o número de processadores. Isso implica não apenas em uma redução do consumo de energia mas também em um aumento no poder computacional. Todavia, para usufruir desse poder computacional, as aplicações (*software*) devem ser reestruturadas para computadores *multicores*, em particular, desenvolvendo novas formas de paralelismo numérico para problemas computacionais mais complexos [33].

Esse cenário demonstra parte da tendência de desenvolvimento na computação científica hoje. O paradigma da computação paralela, como é chamada, aparece como mais um desafio para vários campos da ciência. Em contra partida a essa tendência, os atuais processadores se limitam a um número pequeno de núcleos, o que limita a aceleração ao número de *cores* disponíveis. Assim, essa limitação levou ao surgimento de outros paradigmas em termos de *hardwares* paralelizáveis.

Nos últimos anos, os *hardwares* de aceleração gráfica se desenvolveram vertiginosamente motivados pela indústria de jogos [34]. Por volta de 2001, vislumbrando a oportunidade de aplicar essa nova tecnologia na computação científica, a NVIDIA deu o primeiro passo no desenvolvimento de placas gráficas programáveis voltadas à computação científica. Neste contexto, as GPUs (Unidade de Processamento Gráfico), antes usadas apenas para processamento gráfico passaram a ser usadas para outros propósitos, encontrando aplicações em vários ramos da ciência. O uso dessas novas arquiteturas também necessitou de uma nova forma de programação, chamada CUDA, a fim de usufruir das centenas de *cores* disponíveis em uma única GPU. As GPUs, hoje, estão na vanguarda do desenvolvendo da computação de alto desempenho. Essa fato é evidenciado pelo número de supercomputadores entre os *TOP500* (lista dos 500 supercomputadores mais poderosos do mundo) que possuem GPUs [35].

Novas tecnologias denominadas "não convencionais" estão sendo criadas nessa direção [33]. Essas ideias não convencionais normalmente usufruem de conceitos da física, da química e da biologia, para criação de novas abstrações e *hardwares* de

processamento paralelo. Hoje pode se citar dois possíveis novos paradigmas que talvez sejam viáveis: a computação baseada em **DNA** e a computação **quântica** [33].

Enquanto que os computadores tradicionais se baseiam em silício (transistores), os computadores de DNA se baseiam em fitas de DNA. Ao invés de tratar as informações de forma binária, as mesmas são transcritas em termos das quatro bases de nucleotídeos. Assim, aplicando a mesma lógica da máquina de Turing [36] só que traduzida de forma semelhante aos códigos genéticos. Em outras palavras, esse computador guarda, transmite e processa informação a partir da codificação de organismos vivos. Já os computadores quânticos, fortes candidatos para próxima geração de computadores, manipulam as informações se baseando em conceitos da mecânica quântica, como superposição. Diferente dos computadores tradicionais, que se fundamentam na forma binária de representar uma informação, os computadores quânticos adicionam o conceito de superposição de informação dos binários '0" e "1" [33]. Tais computadores podem ser construídos utilizando átomos ao invés de nano-cristais de silício. Isso introduz uma certa vantagem relativa ao tamanho e a quantidade de informações traduzidas por essa nova tecnologia.

Infelizmente, essas novas tecnologias ainda estão em fase de maturação, e portanto, o trabalho aqui desenvolvido será focado nas tecnologias de paralelismo que estão bem sedimentadas e consolidadas, como o paralelismo em CPUs e GPUs.

1.3.3 Objetivo

O presente trabalho tem como objetivo desenvolver e implementar ferramentas numéricas em arquiteturas híbridas, placas gráficas (GPUs) e processadores (CPUs), com o intuito de simular escoamentos multifásicos polidispersos de forma rápida e acurada. Para atingir este objetivo, o trabalho aqui desenvolvido foi dividido em três partes:

- Desenvolvimento de um método numérico para solução da equação de balanço populacional com controle de acurácia. Mais detalhes são descritos ao longo do texto.
- Paralelizar em GPUs o método numérico desenvolvido neste trabalho.
- Realizar o acoplamento PB-CFD no OpenFOAM® utilizando o método numérico paralelo em GPUs.

1.4 Organização do Texto

O presente texto é dividido em 8 capítulos. Os fundamentos da teoria de balanço populacional são expostos no Capítulo 2. Este capítulo abrange desde a formulação básica até o equacionamento geral da equação de balanço populacional e suas possíveis simplificações. Posteriormente, os diversos métodos de solução da equação de balanço populacional são elucidado no Capítulo 3. Neste trecho, os métodos híbridos receberam maior atenção, pois são, atualmente, os mais relevantes para o acoplamento PB-CFD. Em seguida no Capítulo 4 realizaremos uma revisão bibliográfica sobre a modelagem multi-fluido.

O Capítulo 5 analisa os diferentes modelos de programação paralela, e os motivos pelos quais o modelo de programação paralela CUDA foi usado. No Capítulo 6, a metodologia numérica e os algoritmos paralelos em CUDA propostos são formulados e discutidos. Por fim, o método paralelizado é então estendido para sua aplicação em CFD.

No sétimo capítulo, os resultados relativos ao paralelismo e ao método numérico desenvolvido são esmiuçados e discutidos, sendo relacionados com os artigos desenvolvidos neste trabalho. Ainda neste capítulo, o conceito da metodologia de simulação PB-CFD proposta é testada exaustivamente a fim de provar sua validade. Finalmente, no Capítulo 8, discutiremos as conclusões obtidas nesse trabalho e sugestões para trabalhos futuros.

Capítulo 2

Teoria de Balanço Populacional

Em muitos problemas práticos, o comportamento de partículas é crucial para a análise do desempenho de equipamentos onde ocorrem escoamentos multifásicos polidispersos. Deste modo, torna-se necessário realizar estudos que caracterizem melhor o comportamento espacial e dinâmico de uma população de partículas. Neste contexto, surge em 1916 [37] a teoria de balanço populacional [14].

O balanço populacional estuda a conservação da distribuição numérica de uma população de partículas. Matematicamente, as variáveis que afetam essa distribuição podem ser de dois tipos: variáveis externas e internas. As variáveis externas, $\mathbf{x}$ $\{\mathbf{x} \in \Omega_x\}$, referem-se à localização espacial de cada partícula, ou seja, suas coordenadas no $\mathbb{R}^3$. Por sua vez, as variáveis internas, $\mathbf{r}$ $\{\mathbf{r} \in \Omega_r\}$, referem-se às propriedades das partículas, tais como: diâmetro, área superficial, entre outros (RAMKRISHNA [14]). A Figura 2.1 representa um escoamento gás-líquido e ilustra a diferença entre variável interna e externa.

Pode-se considerar que o desenvolvimento dessa teoria teve início no século XIX, sendo a equação de Boltzmann a primeira aparição da PBE (YEOH e TU [10]), embora tenha sido expressa em termos de uma distribuição de moléculas, utilizando apenas as velocidades das moléculas como variáveis internas. HULBURT e KATZ [38] foram os pioneiros na aplicação desse equacionamento a um problema de engenharia química.

Desde então, a técnica começou a se difundir na engenharia química, sendo um marco importante para o surgimento de uma série de livros textos sobre balanço populacional aplicado a aero-colóides (PANDIS e SEINFELD [39], HIDY e BROCK [40], FRIEDLANDER [41]). Não obstante, toda a flexibilidade e capacidade da teoria só foi sedimentada como ferramenta básica a ser aplicada a processos de engenharia com o livro publicado por RAMKRISHNA [14] e mais recentemente no livro de MARCHISIO e FOX [12].

Mesmo sendo uma teoria antiga, a implementação e a aplicação da mesma contínua em desenvolvimento. Ainda existem muitas barreiras a serem ultrapassadas,

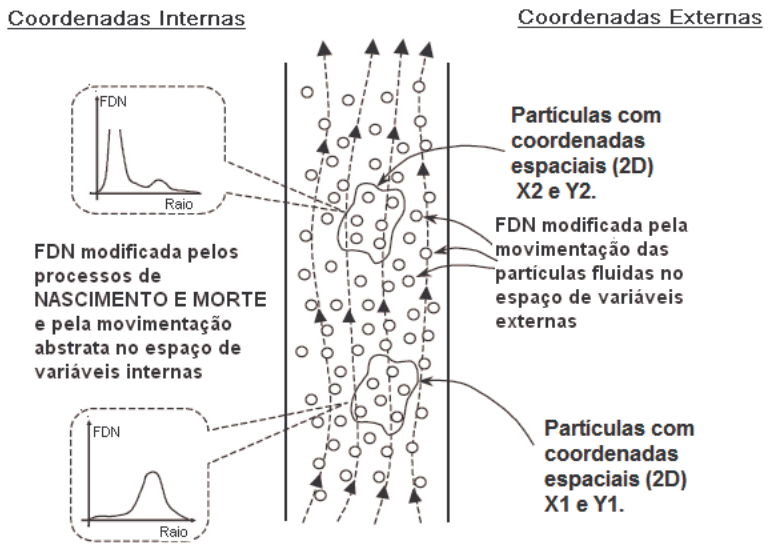


Figura 2.1: Exemplo ilustrativo dos tipos de variáveis interna, raio da partícula, e externas, coordenadas espaciais 2D (adaptado de YEOH e TU [10]).

como por exemplo: a modelagem dos fenômenos de quebra, nucleação, crescimento e coalescência.

Neste capítulo são apresentados os conceitos da teoria de balanço populacional, desde o seu equacionamento com as suas principais simplificações e hipóteses.

2.1 Função Densidade Numérica

A origem da formulação de mesoescala proposta pela teoria de balanço populacional está no conceito de Função Densidade Numérica (FDN). Embora o comportamento de uma única partícula possa ser considerado aleatório, prever o comportamento das partículas usando balanço populacional nada mais é que representar um sistema em microescala pelo comportamento médio amostral de sua população de partículas.

Partindo desse princípio, a função de densidade numérica média de partículas, $f(\mathbf{x}, \mathbf{r}, t)$, no espaço de variáveis internas e externas, sob a influência de propriedades da fase contínua, as quais são agrupadas em um único vetor de variáveis, $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_c\}$, de dimensão c, é definida pela Equação (2.1).

$$E[n(\mathbf{x}, \mathbf{r}, t)] = f(\mathbf{x}, \mathbf{r}, t), \tag{2.1}$$

$$\mathbf{x} \in \Omega_x, \quad \mathbf{r} \in \Omega_r$$

Na Equação (2.1), $f(\mathbf{x}, \mathbf{r}, t)$ é a expectativa (definida pelo operador E), para o conjunto de realizações do processo, da densidade numérica instantânea local de cada realização, n, quando o número de realizações tende ao infinito. Interessante frisar que a função densidade numérica média é suave, o que permite que seja diferenciada

em qualquer variável quantas vezes for necessário, além de permitir calcular o número médio de partículas em relação à qualquer região no espaço de variáveis de estado. Assim, o número médio total de partículas, N_T, pode ser calculado pela Equação (2.2).

$$N_T(t) = \int_{\Omega_{\mathbf{x}}} d\mathbf{V}_{\mathbf{x}} \int_{\Omega_{\mathbf{r}}} f(\mathbf{x}, \mathbf{r}, t) d\mathbf{V}_{\mathbf{r}}, \tag{2.2}$$

onde $d\mathbf{V}_{\mathbf{r}}$ e $d\mathbf{V}_{\mathbf{x}}$ são os volumes infinitesimais no espaço de coordenadas internas e externas, respectivamente. Da mesma forma, pode-se calcular a densidade numérica de partículas no espaço físico (número de partículas por unidade de volume físico):

$$N_x(\mathbf{x}, t) = \int_{\Omega_{\mathbf{r}}} d\mathbf{V}_{\mathbf{r}} f(\mathbf{x}, \mathbf{r}, t). \tag{2.3}$$

Outras formas de distribuição de densidade podem ser definidas para uma população. Por exemplo, sendo $v(\mathbf{r})$ o volume da partícula no espaço de variáveis internas $\mathbf{r}$, define-se $v(\mathbf{r})f(\mathbf{x}, \mathbf{r}, t)$ como a distribuição de densidade volumétrica. Desta forma, é possível escrever a fração volumétrica local e instantânea da fase dispersa por:

$$\gamma(\mathbf{x}, t) = \int_{\Omega_{\mathbf{r}}} v(\mathbf{r}) f(\mathbf{x}, \mathbf{r}, t) d\mathbf{V}_{\mathbf{r}}. \tag{2.4}$$

Vale ressaltar que as propriedades da população de partículas são usualmente mais relevantes que a densidade numérica para as aplicações práticas. Porém, representar a PBE baseada na densidade numérica é mais vantajoso, pois facilita a modelagem dos termos de quebra e coalescência.

As mudanças temporais das variáveis externas são definidas como velocidade no espaço físico, enquanto que as variáveis internas são definidas como uma movimentação abstrata da propriedade no espaço das variáveis internas. Por exemplo, a transferência de massa de um componente da fase contínua para a dispersa pode ser considerada uma movimentação no espaço de concentração. Assim, diz-se que $\dot{\mathbf{R}}(\mathbf{x}, \mathbf{r}, \mathbf{Y}, \mathbf{t})$ é a velocidade no espaço de variáveis internas e $\dot{\mathbf{X}}(\mathbf{x}, \mathbf{r}, \mathbf{Y}, \mathbf{t})$ é a velocidade no espaço físico.

Depois de definida as velocidades em ambos os espaços, agora é possível identificar o escoamento de partículas[7]. Portanto, $f(\mathbf{x}, \mathbf{r}, t)\dot{\mathbf{R}}(\mathbf{x}, \mathbf{r}, \mathbf{Y}, \mathbf{t})$ representa o fluxo de partículas no espaço de variáveis internas e $f(\mathbf{x}, \mathbf{r}, t)\dot{\mathbf{X}}(\mathbf{x}, \mathbf{r}, \mathbf{Y}, \mathbf{t})$, o fluxo no espaço físico.

[7]Número de partículas transportadas por unidade de tempo por unidade de área normal a direção da velocidade.

2.2 Equação de Balanço Populacional

Usando as definições acima, a equação de balanço populacional pode ser escrita como (RAMKRISHNA [14]):

$$\underbrace{\frac{\partial f(\mathbf{x},\mathbf{r},t)}{\partial t}}_{Transiente} = \underbrace{-\nabla_{\mathbf{x}}[\dot{\mathbf{X}} f(\mathbf{x},\mathbf{r},t)]}_{Advectivo} + \underbrace{\nabla_{\mathbf{x}} \cdot \left(\mathbf{D}_{\mathbf{x}} \cdot [\nabla_x \cdot (\mathbf{D}_{\mathbf{x}}{}^T f(\mathbf{x},\mathbf{r},t)]\right)}_{Difusivo} + \underbrace{H(\mathbf{x},\mathbf{r},\mathbf{Y},t)}_{Fonte} \tag{2.5}$$

O primeiro termo da Equação (2.5) representa o acúmulo de partículas no espaço das variáveis $\Omega_{\mathbf{x}} \times \Omega_{\mathbf{r}}$. Os segundo e terceiro termos representam o transporte advectivo e o dispersivo de partículas no espaço físico, respectivamente.

$\dot{\mathbf{X}}(\mathbf{x},\mathbf{r},\mathbf{Y},\mathbf{t})$ é a taxa de variação da variável externa (velocidade no espaço físico), $\mathbf{D}_{\mathbf{x}}$ é o coeficiente de dispersão anisotrópico. $H(\mathbf{x},\mathbf{r},\mathbf{Y},t)$ é o termo fonte, que pode ser dividido entre a taxa de nucleação, $J(\mathbf{x},\mathbf{r},\mathbf{Y},t)$, na taxa de crescimento, $G(\mathbf{x},\mathbf{r},\mathbf{Y},t)$, no termo fonte adicional, $s(\mathbf{x},\mathbf{r},\mathbf{Y},t)$, e nas taxas líquidas de nascimento e morte, $B(\mathbf{x},\mathbf{r},\mathbf{Y},t)$ e $D(\mathbf{x},\mathbf{r},\mathbf{Y},t)$, pelos fenômenos de coalescência (subscrito a) e de quebra (subscrito b), conforme a Equação 2.6.

$$\begin{aligned} H(\mathbf{x},\mathbf{r},\mathbf{Y},t) = B_a(\mathbf{x},\mathbf{r},\mathbf{Y},t) - D_a(\mathbf{x},\mathbf{r},\mathbf{Y},t) + B_b(\mathbf{x},\mathbf{r},\mathbf{Y},t) - D_b(\mathbf{x},\mathbf{r},\mathbf{Y},t) \\ + J(\mathbf{x},\mathbf{r},\mathbf{Y},t) + G(\mathbf{x},\mathbf{r},\mathbf{Y},t) + s(\mathbf{x},\mathbf{r},\mathbf{Y},t) \end{aligned} \tag{2.6}$$

O termo de crescimento $G(\mathbf{x},\mathbf{r},\mathbf{Y},t)$ é o divergente do fluxo total das variáveis internas. A equação geral para o termo de crescimento é dada por:

$$G(\mathbf{r},\mathbf{x},\mathbf{Y},t) = \underbrace{-\bigtriangledown_{\mathbf{r}} \cdot [\dot{\mathbf{R}} f(\mathbf{r},\mathbf{x},t)]}_{Advectivo} + \underbrace{\bigtriangledown_{\mathbf{r}} \cdot [\mathbf{D}_{\mathbf{r}} . (\bigtriangledown_r \cdot \mathbf{D}_{\mathbf{r}}{}^T f(\mathbf{r},\mathbf{x},t)]}_{Difusivo} \tag{2.7}$$

onde $\dot{\mathbf{R}}(\mathbf{x},\mathbf{r},\mathbf{Y},\mathbf{t})$ é taxa de variação determinística da variável interna e $\mathbf{D}_{\mathbf{r}}$ é o coeficiente de dispersão anisotrópico no espaço de variáveis internas. Geralmente, os termos de nucleação podem ser modelados como uma condição de fluxo no contorno $\partial\Omega_{\mathbf{r}}$, sendo neste caso o termo J retirado da PBE.

Para modelar uma polidispersão, o conhecimento dos processos físicos associados aos termos fontes são de vital importância para o sucesso da modelagem. Nas próximas seções, serão expostos detalhes sobre a modelagem dos termos de quebra e coalescência.

Notem também que a velocidade das partículas pode ser considerada uma variáveis interna da PBE. Neste caso, diz-se Equação de Balanço Populacional é Generalizada (GPBE) [12] ou definida como um "modelo cinético". Alguns autores defendem

o uso da PBE em sua forma generalizada, no entanto, essa é geralmente utilizada para aplicações onde os efeitos de quebra e coalescência não são significativos [4], [42]. Observe também que o uso da GPBE implica no aparecimento de pelo menos três novas variáveis internas, o que aumenta bastante o seu custo computacional.

2.3 Fenômeno de Coalescência

Coalescência é o processo onde duas ou mais partículas se unem, após se encontrarem, para formar uma nova partícula "filha". Em uma dispersão, as partículas se movimentam e podem colidir continuamente uma com as outras. Contudo, nem toda colisão acarretará em coalescência. É necessário que ocorram mais dois fenômenos: a drenagem e a ruptura do filme gerado entre as partículas em colisão. Desta forma, a coalescência só ocorre quando esse três fenômenos acontecem sequencialmente.

Para a modelagem do processo de coalescência binária define-se a frequência de coalescência, que representa a fração de pares de partículas com estado $(\tilde{\mathbf{x}}, \tilde{\mathbf{r}})$ e $(\mathbf{x}', \mathbf{r}')$, que sob o efeito da fase contínua representada por $\mathbf{Y}$, se agrega em um período de tempo entre t e $t + dt$.

$$a(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, \mathbf{r}', \mathbf{x}', \mathbf{Y}, t)dt \tag{2.8}$$

Podemos reconhecer que $a(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, \mathbf{r}', \mathbf{x}', \mathbf{Y}, t)$ também é a probabilidade de um par de partículas de estado $(\tilde{\mathbf{x}}, \tilde{\mathbf{r}})$ e $(\mathbf{x}', \mathbf{r}')$ se agregar na unidade de tempo t. A Figura (2.2) apresenta a possibilidade da colisão não ser efetiva, ou em outras palavras, de duas partículas colidirem e não coalescerem. Portanto, para modelar a frequência de coalescência, é necessário não só obter a frequência de colisão, $\bar{w}$, mas também a eficiência de coalescência, η,

frequência de coalescência = frequência de colisão x eficiência de coalescência

$$a(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, \mathbf{r}, \mathbf{x}, \mathbf{Y}, t) = \bar{w}(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, \mathbf{r}, \mathbf{x}, \mathbf{Y}, t)\eta(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, \mathbf{r}, \mathbf{x}, \mathbf{Y}, t) \tag{2.9}$$

Com a frequência de coalescência, pode-se definir os termos de nascimento e morte por coalescência, dado o estado $(\mathbf{x}, \mathbf{r})$ de uma nova partícula, formada da coalescência de uma partícula de estado $(\mathbf{x}', \mathbf{r}')$ com outra de estado $[\tilde{\mathbf{x}}(\mathbf{x}', \mathbf{r}'|\mathbf{x}, \mathbf{r}), \mathbf{r}(\mathbf{x}', \mathbf{r}'|\mathbf{x}, \mathbf{r})]$. É necessário especificar a distribuição de densidade numérica de pares de partículas, $f_2(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, \mathbf{x}', \mathbf{r}', t)$, que colidem com estado $(\tilde{\mathbf{x}}, \tilde{\mathbf{r}})$ e $(\mathbf{x}', \mathbf{r}')$ no instante t por unidade de volume dos espaços de estados. O termo fonte de nascimento por coalescência, B_a, descrito na Equação 2.10, produz uma partícula com estado $(\mathbf{x}, \mathbf{r})$.

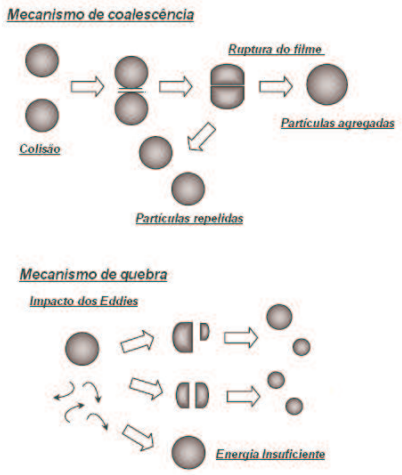


Figura 2.2: Mecanismos de quebra e coalescência (YEOH e TU [10]).

$$B_a(\mathbf{r},\mathbf{x},\mathbf{Y},t) = \frac{1}{\xi}\int_{\Omega_{\mathbf{x}'}} d\mathbf{V}_{\mathbf{x}'}\int_{\Omega_{\mathbf{r}'}} d\mathbf{V}_{\mathbf{r}'} a(\tilde{\mathbf{x}},\tilde{\mathbf{r}},\mathbf{r}',\mathbf{x}',\mathbf{Y},t) f_2(\tilde{\mathbf{x}},\tilde{\mathbf{r}},\mathbf{x}',\mathbf{r}',t)\frac{\partial(\tilde{\mathbf{x}},\tilde{\mathbf{r}})}{\partial(\mathbf{x},\mathbf{r})} \tag{2.10}$$

onde ξ controla o número de vezes que um par idêntico de partículas foi computado na integração, assim, o fator $\frac{1}{\xi}$ corrige possíveis erros de redundância. Sabendo que $f_2(\tilde{\mathbf{x}},\tilde{\mathbf{r}},\mathbf{x}',\mathbf{r}',t)$ é dada por unidade de volume das coordenadas $(\tilde{\mathbf{x}},\tilde{\mathbf{r}},\mathbf{x}',\mathbf{r}')$, deve-se fazer a transformação de coordenadas de partículas que colidem com estado $(\tilde{\mathbf{x}},\tilde{\mathbf{r}})$ para a partícula gerada com estado $(\mathbf{x},\mathbf{r})$. Por isso, a existência do determinante $\frac{\partial(\tilde{\mathbf{x}},\tilde{\mathbf{r}})}{\partial(\mathbf{x},\mathbf{r})}$, que é dado por:

$$\frac{\partial(\tilde{\mathbf{x}},\tilde{\mathbf{r}})}{\partial(\mathbf{x},\mathbf{r})} = \begin{vmatrix} \frac{\partial\tilde{\mathbf{x}}_1}{\partial\tilde{\mathbf{x}}_1} & \frac{\partial\tilde{\mathbf{x}}_1}{\partial\tilde{\mathbf{x}}_2} & \frac{\partial\tilde{\mathbf{x}}_1}{\partial\tilde{\mathbf{x}}_3} & \frac{\partial\tilde{\mathbf{x}}_1}{\partial\tilde{\mathbf{r}}_1} & \cdots & \frac{\partial\tilde{\mathbf{x}}_1}{\partial\tilde{\mathbf{r}}_n} \\ \frac{\partial\tilde{\mathbf{x}}_2}{\partial\tilde{\mathbf{x}}_1} & \frac{\partial\tilde{\mathbf{x}}_2}{\partial\tilde{\mathbf{x}}_2} & \frac{\partial\tilde{\mathbf{x}}_2}{\partial\tilde{\mathbf{x}}_3} & \frac{\partial\tilde{\mathbf{x}}_2}{\partial\tilde{\mathbf{r}}_1} & \cdots & \frac{\partial\tilde{\mathbf{x}}_2}{\partial\tilde{\mathbf{r}}_n} \\ \frac{\partial\tilde{\mathbf{x}}_3}{\partial\tilde{\mathbf{x}}_1} & \frac{\partial\tilde{\mathbf{x}}_3}{\partial\tilde{\mathbf{x}}_2} & \frac{\partial\tilde{\mathbf{x}}_3}{\partial\tilde{\mathbf{x}}_3} & \frac{\partial\tilde{\mathbf{x}}_3}{\partial\tilde{\mathbf{r}}_1} & \cdots & \frac{\partial\tilde{\mathbf{x}}_3}{\partial\tilde{\mathbf{r}}_n} \\ \frac{\partial\tilde{\mathbf{r}}_1}{\partial\tilde{\mathbf{x}}_1} & \frac{\partial\tilde{\mathbf{r}}_1}{\partial\tilde{\mathbf{x}}_2} & \frac{\partial\tilde{\mathbf{r}}_1}{\partial\tilde{\mathbf{x}}_3} & \frac{\partial\tilde{\mathbf{r}}_1}{\partial\tilde{\mathbf{r}}_1} & \cdots & \frac{\partial\tilde{\mathbf{r}}_1}{\partial\tilde{\mathbf{r}}_n} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial\tilde{\mathbf{r}}_n}{\partial\tilde{\mathbf{x}}_1} & \frac{\partial\tilde{\mathbf{r}}_n}{\partial\tilde{\mathbf{x}}_2} & \frac{\partial\tilde{\mathbf{r}}_n}{\partial\tilde{\mathbf{x}}_3} & \frac{\partial\tilde{\mathbf{r}}_n}{\partial\tilde{\mathbf{r}}_1} & \cdots & \frac{\partial\tilde{\mathbf{r}}_n}{\partial\tilde{\mathbf{r}}_n} \end{vmatrix} \tag{2.11}$$

O termo fonte de morte por coalescência é descrito por:

$$D_a(\mathbf{r},\mathbf{x},\mathbf{Y},t) = \int_{\Omega_{\mathbf{x}'}} d\mathbf{V}_{\mathbf{x}'}\int_{\Omega_{\mathbf{r}'}} d\mathbf{V}_{\mathbf{r}'} a(\mathbf{x},\mathbf{r},\mathbf{r}',\mathbf{x}',\mathbf{Y},t) f_2(\mathbf{x},\mathbf{r},\mathbf{x}',\mathbf{r}',t) \tag{2.12}$$

O termo de coalescência da PBE não está fechado, pois $f_2(\tilde{\mathbf{x}},\tilde{\mathbf{r}},\mathbf{x}',\mathbf{r}',t)$ ainda é desconhecido. Geralmente, nas análises que usam balanço populacional, pode-se fazer a seguinte aproximação:

$$f_2(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, \mathbf{x}', \mathbf{r}', t) = f(\tilde{\mathbf{x}}, \tilde{\mathbf{r}}, t) f(\mathbf{x}', \mathbf{r}', t) \tag{2.13}$$

Essa suposição implica que não existe correlação estatística entre as partículas com estados $(\tilde{\mathbf{x}}, \tilde{\mathbf{r}})$ e $(\mathbf{x}, \mathbf{r})$ no instante t. Esta suposição é plausível quando se trata de uma ampla população de partículas [14].

2.4 Fenômeno de Quebra

O processo de quebra é a ruptura de uma partícula em, pelo menos, duas outras [25]. A quebra pode ser ocasionada da colisão da partícula contra uma superfície sólida ou pela interação da partícula com campos cisalhantes ou pela ação da turbulência. O primeiro mecanismo é mais evidente em partículas sólidas, onde a partícula é lascada ou atritada com a superfície de outra partícula, parede ou impelidor. Os outros dois mecanismos dependem da interação da partícula com a fase contínua. Neste caso, o impacto não é mais com uma superfície rígida, mas sim com turbilhões (*eddies*). De fato, o contato da fase dispersa com o campo cisalhante do meio contínuo pode também provocar oscilações de formato na partícula fluida pela ação das forças inerciais e coesivas, e como consequência, induzir a quebra. Deste modo, os modelos de quebra se apoiam em três teorias básicas: colisão entre partículas e superfície rígida, colisão entre partículas fluidas e *eddies* ou ruptura devido à deformação da partícula por influência da fase contínua.

Se o processo de quebra de partículas ocorrer independentemente uma da outra, pode-se definir uma taxa específica de quebra, $b(\mathbf{x}, \mathbf{r}, \mathbf{Y}, t)$, interpretada como a fração de partículas com estado $(\mathbf{x}, \mathbf{r})$ que quebram por unidade de tempo, no instante t, em um ambiente descrito por $\mathbf{Y}$. Então o número médio de partículas "perdidas" por unidade de tempo e volume do espaço é dado por:

$$D_b(\mathbf{x}, \mathbf{r}, \mathbf{Y}, t) = b(\mathbf{x}, \mathbf{r}, \mathbf{Y}, t) f(\mathbf{x}, \mathbf{r}, t) \tag{2.14}$$

Antes de caracterizar o termo fonte de nascimento por quebra, deve-se definir as seguintes variáveis:

- $\nu(\mathbf{x}', \mathbf{r}', \mathbf{Y}, t)$ é número médio de partículas formado pela quebra da partícula com estado $(\mathbf{x}', \mathbf{r}')$ $(\nu \geq 2)$.
- $P(\mathbf{x}, \mathbf{r} | \mathbf{x}', \mathbf{r}', \mathbf{Y}, t)$ é a função de densidade de probabilidade das partículas formadas pela quebra de uma partícula de estado $(\mathbf{x}', \mathbf{r}')$ em um ambiente $\mathbf{Y}$ no instante t, de terem o estado $(\mathbf{x}, \mathbf{r})$.

Assim o termo de nascimento por quebra é dado por (RAMKRISHNA [14]):

$$B_b(\mathbf{r},\mathbf{x},\mathbf{Y},t) = \int_{\Omega_{\mathbf{x}'}}\int_{\Omega_{\mathbf{r}'}} \nu(\mathbf{x}',\mathbf{r}',\mathbf{Y},t)P(\mathbf{x},\mathbf{r}|\mathbf{x}',\mathbf{r}',\mathbf{Y},t)b(\mathbf{x}',\mathbf{r}',\mathbf{Y},t) \\ f(\mathbf{x}',\mathbf{r}',\mathbf{Y},t)d\mathbf{V}_{\mathbf{r}'}d\mathbf{V}_{\mathbf{x}'} \qquad (2.15)$$

2.5 Fenômeno de Nucleação

É o processo onde partículas surgem no domínio por efeitos de mudança de fase ou reação química [43]. Apesar da definição ser simples, hoje, ainda não se sabe exatamente como a formação dos núcleos acontece. No entanto, a nucleação que aparece por trás de fenômenos como cristalização e condensação, geralmente, descrita pela teoria de *clusters*, onde um núcleo estável é formado pela aglutinação de solutos ou moléculas para formar uma partícula com tamanho superior a um dado tamanho crítico. O leitor pode encontrar mais detalhes sobre essa teoria em MULLIN [43].

Basicamente, a taxa com que as partículas são formadas é definida pelo número de núcleos gerados por unidade de volume no espaço de variáveis e unidade de tempo. Como mencionado na Subseção 2.2, o termo de nucleação pode ser modelado como condição de contorno, $\dot{n}_0 = \dot{\mathbf{R}}f(\mathbf{x},\mathbf{r},\mathbf{Y},t)$ onde $\mathbf{r} \in \partial\Omega_{r_0}$ (contorno da variável interna em $\mathbf{r_0}$), sendo assim omitido da PBE. Se mantido na equação de balanço populacional, o termo de nucleação é modelado por $J(\mathbf{x},\mathbf{r},\mathbf{Y},t) = \dot{n}_0\delta(\mathbf{r}-\mathbf{r_0})$, onde $\mathbf{r_0}$ é o tamanho crítico de formação e $\dot{n}_0$ é a taxa de nucleação.

2.6 Simplificações e Hipóteses Usuais

Por conta da complexidade de modelar os termos da PBE é vital para sua solução considerar hipóteses que simplifiquem o seu equacionamento original sem perder as características físicas intrínsecas a cada problema. As hipóteses físicas e de modelagem apresentadas nesta seção são amplamente usadas na solução da PBE e no desenvolvimento de funções de quebra e coalescência [14].

A morte e o nascimento por coalescência envolvem pelo menos três posições, duas partículas com posições $\mathbf{x}'$ e $\tilde{\mathbf{x}}$ e uma gerada com posição $\mathbf{x}$. Contudo, para problemas práticos de engenharia, que são no geral casos cujas as partículas são muito pequenas se comparadas com o tamanho do equacionamento, formular a PBE levando em conta estas três posições não é necessário. Então, é comum considerar que $\mathbf{x}' \approx \tilde{\mathbf{x}} \approx \mathbf{x}$ nas equações anteriores. Em outras palavras, a coalescência é local e instantânea.

Assumir que as diferentes posições são próximas resulta nas Equações (2.16) e (2.17).

$$B_a(\mathbf{r},\mathbf{x},\mathbf{Y},t) = \frac{1}{\xi}\int_{\Omega_{\mathbf{r}'}} d\mathbf{V}_{\mathbf{r}'} a(\mathbf{x},\tilde{\mathbf{r}},\mathbf{r}',\mathbf{Y}) f(\mathbf{x},\tilde{\mathbf{r}},t) f(\mathbf{x},\mathbf{r}',t) \frac{\partial(\tilde{\mathbf{r}})}{\partial(\mathbf{r})} \tag{2.16}$$

$$D_a(\mathbf{r},\mathbf{x},\mathbf{Y},t) = \int_{\Omega_{\mathbf{r}'}} a(\mathbf{x},\mathbf{r},\mathbf{r}',\mathbf{Y}) f(\mathbf{x},\mathbf{r},t) f(\mathbf{x},\mathbf{r}',t) d\mathbf{V}_{\mathbf{r}'} \tag{2.17}$$

Considerando o problema multivariado, a jacobiana da transformação de coordenadas passa a ser:

$$\frac{\partial(\tilde{\mathbf{r}})}{\partial(\mathbf{r})} = \begin{vmatrix} \frac{\partial \tilde{r}_1}{\partial r_1} & \cdots & \cdots & \frac{\partial \tilde{r}_1}{\partial r_n} \\ \frac{\partial \tilde{r}_2}{\partial r_1} & \cdots & \cdots & \frac{\partial \tilde{r}_2}{\partial r_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \tilde{r}_n}{\partial r_1} & \cdots & \cdots & \frac{\partial \tilde{r}_n}{\partial r_n} \end{vmatrix}$$

Se o problema for monovariado e a propriedade interna for aditiva na coalescência, o jacobiano da transformação se torna unitário. O exemplo abaixo mostra o jacobiano para a massa, m, de partícula.

$$m' + \tilde{m} = m, \tag{2.18}$$

$$\frac{\partial(\tilde{m})}{\partial(m)} = 1 \tag{2.19}$$

Hipótese semelhante pode ser estendida para o fenômeno de quebra. O processo de quebra ocorre em uma escala de tempo muito menor que a evolução da população. Desta forma, a posição das partículas mãe e filhas podem ser consideradas iguais. Assim, o termo de nascimento por quebra se torna:

$$B_b(\mathbf{r},\mathbf{x},\mathbf{Y},t) = \int_{\Omega_{\mathbf{r}'}} \nu(\mathbf{x},\mathbf{r}',\mathbf{Y},t) P(\mathbf{x},\mathbf{r}|\mathbf{r}',\mathbf{Y},t) b(\mathbf{x},\mathbf{r}',\mathbf{Y},t) f(\mathbf{x},\mathbf{r}',t) d\mathbf{V}_{\mathbf{r}'} \tag{2.20}$$

A equação de balanço populacional escrita em termos de uma distribuição mássica, considerando os *Kernels* (núcleos), a, b e P independentes do tempo explícitamente, e levando em conta as simplificações citadas nessa seção, é ilustrada na Equação 2.21.

$$\frac{\partial f(m,\mathbf{x},t)}{\partial t} + \bigtriangledown_x[\dot{\mathbf{X}} f(m,\mathbf{x},t)] - \nabla_x \cdot [D_x \nabla_x f(m,\mathbf{x},t)]$$
$$= \frac{1}{2}\int_0^{m'} a(\tilde{m}, m', \mathbf{x}, \mathbf{Y}, t) f(m', \mathbf{x}, t) f(\tilde{m}, \mathbf{x}, t) dm'$$
$$- \int_0^{\infty} a(m', m, \mathbf{x}, \mathbf{Y}, t) f(m, \mathbf{x}, t) f(m', \mathbf{x}, t) dm'$$
$$+ \int_m^{\infty} \nu(m', \mathbf{x}, \mathbf{Y}, t) b(m', \mathbf{x}, \mathbf{Y}, t) P(m|m', \mathbf{x}, \mathbf{Y}, t) f(m', \mathbf{x}, t) dm'$$
$$- b(m, \mathbf{x}, \mathbf{Y}, t) f(m, \mathbf{x}, t) + \dot{n}_0 \delta(m - m_0) + G(m, \mathbf{x}, \mathbf{Y}, t) + S(m, \mathbf{x}, \mathbf{Y}, t) \quad (2.21)$$

Contudo, além das simplificações anteriores, a equação de balanço populacional usada no presente trabalho despreza o termo difusivo para todos as análises. Para a avaliação aqui proposta, esse termo não é necessário e sua análise foge do escopo desse trabalho. Neste capítulo foram vistos os fundamentos necessários para compreensão da teoria do balanço populacional. Já no próximo capítulo, serão apresentados os principais métodos de solução da PBE.

Capítulo 3

Métodos de Solução da PBE

A PBE é uma equação integro-diferencial com dependência temporal e espacial (coordenadas internas e externas) que quase nunca pode ser resolvida analiticamente. Sua solução analítica só é viável em casos muito simples, que geralmente não são aplicáveis a situações reais (PATIL e ANDREWS [44], LAGE [45], MCCOY e MADRAS [46], LIOU *et al.* [47]). No entanto, os métodos de solução analítica não são dispensáveis, visto que podem ser utilizados para testar e validar novas técnicas de soluções numéricas.

Entre as técnicas analíticas mais comuns estão o método de aproximações sucessivas, o método das características, o método das gerações sucessivas e o método de transformada de Laplace. Já entre os métodos numéricos, os mais comuns são os métodos estocásticos (Monte Carlo), o método das classes, o método dos resíduos ponderados, o método dos momentos e os métodos híbridos.

3.1 Métodos Estocásticos

São métodos baseados em realizações artificiais do comportamento de um sistema particulado. Com esse procedimento, é possível prever sua função de densidade numérica [14]. Portanto, o equacionamento integro-diferencial, que é determinístico[9], é substituído pela simulação estocástica do sistema.

A simulação estocástica de um sistema particulado foi introduzida por KENDALL [48] para um processo simples de quebra e agregação. Todavia, sua generalização só foi realizada em 1977 por SHAH *et al.* [49]. Neste caso, as taxas de quebra e agregação eram proporcionais ao número de partículas do sistema e função do tamanho das partículas. Posteriormente, DAS [50] criou uma metodologia considerada semi-analítica para o processo de quebra utilizando o método de Monte Carlo. O autor sugere que, na ausência de solução analítica, seu método pode ser

[9]Cálculo da expectativa da função distribuição.

usado sem perda de generalidade para problemas de quebra pura. GOODSON e KRAFT [51] avaliaram dois algoritmos estocásticos, o *Direct Simulation Algorithm* (DSA) de EIBECK e WAGNER [52] e o *Mass Flow Algorithm* (MFA) desenvolvido por DEBRY *et al.* [53], frente à problemas de quebra e coalescência. Neste trabalho, observaram que o método MFA prediz com maior precisão os momentos de ordem maiores que dois. No entanto, esse método falha ao tentar prever o momento de ordem zero, o que não ocorre com o DSA.

IMMANUEL e DOYLE [54] simularam a dinâmica de uma população utilizando o método de Monte Carlo com três variáveis internas, avaliando a influência dos diferentes termos de agregação. Concluíram que o método é eficiente para problemas multi-dimensionais (multivariado). ZHAO *et al.* [55] compararam quatro métodos mais usuais de simulação estocástica: *Constant Number Monte Carlo* (CNMC), *Stepwise Constant Volume Monte Carlo* (SCVMC), *Direct Simulation Monte Carlo* (DSMC) e *Multi-Monte Carlo* (MMC). Concluíram que os métodos SCVMC e CNMC são os mais eficientes, mas a escolha de um deles depende do comportamento da distribuição de partículas. Se o número de partículas aumentar com o tempo, o método CNVC é o mais indicado, não obstante se a concentração diminuir, o DSMC é o recomendado.

IRIZARRY [56] desenvolveu o *Point Ensemble Monte Carlo* (PEMC), que promoveu a aceleração o método de Monte Carlo tradicional. Esse método consiste em criar subdomínios no espaço de variáveis internas com suas próprias realizações artificiais. Cada um desses subdomínios se comunica por meio de uma rede de reações chamadas de *Random Product Channels* (RPC). Essas RPCs são definidas como representações aproximadas do processo de quebra, agregação e crescimento. A construção da rede de reações, nada mais é que a interpretação estocástica da equação de balanço populacional. Os autores atribuem a essa forma de representação a velocidade desse método, uma vez que ele reduz o número de partículas e realizações necessárias. IRIZARRY [57] alterou o método PEMC para o método τ-PEMC, cujas avaliações temporais de cada realização são defasadas de um parâmetro τ (intervalo de quiescência). Essa metodologia permitiu acelerar significativamente o tempo de computação do método PEMC sem reduzir a acurácia do método. Em seguida, IRIZARRY [58] adaptou seu método para problemas com múltiplas escalas de tempo, criando uma metodologia híbrida de Monte Carlo. Neste método, IRIZARRY [58] aplicou o método τ-PEMC para processos rápidos e o método Monte Carlo tradicional para processos lentos, Concluindo que os métodos híbrido-PEMC, PEMC e Monte Carlo Tradicional são equivalentes em termos de acurácia. Contudo, o método híbrido se mostrou mais rápido devido a eliminação de realizações desnecessárias.

Apesar dos métodos estocásticos serem extremamente flexíveis, é necessário um

número muito alto de realizações, fato que eleva muito o custo computacional. Ademais, incorporar esses métodos a simulações CFD não é uma tarefa simples (YEOH e TU [10]). Atualmente, existe algum investimento na tentativa de viabilizar seu uso em CFD. Nesse sentido, ZHAO *et al.* [59] desenvolveram o método de Monte Carlo com ponderação diferenciada (DWMC) que, segundo os autores, combina acurácia e custo computacional razoável devido a sua capacidade de decisão entre os métodos com volume e numérico constante. Posteriormente, ZHAO e ZHENG [60] criaram uma abordagem que mescla a abordagem Eulerian-Lagrangeana-PB utilizando o DWMC. Embora tenha obtido bons resultados, esse método ainda é extremamente custoso. ZHAO e ZHENG [60] sugerem que, no futuro, essa abordagem pode ser viável se o uso da computação paralela, em especial GPUs, forem utilizadas. Interessante ressaltar que tais métodos são altamente paralelizáveis. Portanto, a paralelização do método de Monte Carlo pode viabilizar seu uso em problemas complexos que envolvem balanço populacional. WEI e KRUIS [61] utilizaram placas gráficas com intuito de acelerar esse tipo de método. Eles usufruíram da independência de cada realização, característica que torna o método de Monte Carlo extremamente paralelizável. WEI e KRUIS [61] investigaram a escalabilidade e aceleração da método Monte Carlo em GPU e obtiveram aceleração de até 100 vezes. Nesse sentido, WEI e KRUIS [61] afirmam que o método pode ser aplicado em simulações CFD se paralelizado. De fato existe essa possibilidade, contudo essa abordagem só se torna vantajosa computacionalmente se for aplicada a problemas com alta dimensionalidade, uma vez que os outros métodos disponíveis se limitam a baixas dimensionalidades.

3.2 Método das Classes

O método das classes (MC) se fundamenta em um processo de discretização da distribuição em um número finito de classes, sendo que cada uma é representada por uma partícula característica, o pivô. Assim, a equação integro-diferencial se transforma em um sistema de equações diferenciais com solução numérica bem estabelecida. KUMAR e RAMKRISHNA [62] propuseram o MC internamente consistente com respeito a dois momentos da distribuição. Para um método ser internamente consistente, é necessário que a discretização da PBE após a aplicação de um operador matemático seja igual à forma discreta desse operador aplicado à forma da PBE [14]. Essa técnica é razoavelmente eficiente, rápida e acurada, fato que torna o MC apropriado para o acoplamento PB-CFD. Para um problema monovariado, a variável interna, neste caso tamanho de partícula, é subdividida em i subintervalos ($P_n = \{m_1 = 0, m_2, \ldots, m_{n+1} = m_{max}\}$), sendo N_i a densidade numérica de partículas no subintervalo i,

BLECK [63] sugeriu que essa discretização na variável interna seguisse uma progressão geométrica, $m_{i+1} = 2m_i$. No entanto, essa abordagem ainda não era internamente consistente. HOUNSLOW *et al.* [64] estudaram a coalescência de gotas em tanques agitados usando o volume de partícula como variável interna seguindo a malha geométrica proposta por BLECK [63]. O autor desenvolveu um equacionamento conservativo em massa e número para Equação 2.21, considerando apenas os efeitos de agregação, como na Equação (3.1), onde $a_{i,j}$ é a frequência de agregação aplicada à partículas nos nós i e j da discretização e i_{max} é o número de classes da malha discretizada.

$$\frac{\partial N_i(\mathbf{x},t)}{\partial t} = N_{i-1}\sum_{j=1}^{i-2} 2^{j-i+1} a_{i-1,j} N_i + \frac{1}{2} a_{i-1,i-1} N_{i-1}^2 - N_i \sum_{j=1}^{i-1} 2^{j-i} a_{i,j} N_j - N_i \sum_{j=1}^{i_max} a_{i,j} N_j \tag{3.1}$$

A técnica proposta por HOUNSLOW *et al.* [64] tem duas desvantagens: é inflexível quanto a discretização da malha no espaço de variáveis internas, e não é genérica, pois em sua metodologia é necessário deduzir novas equações para conservação de propriedades diferentes da massa e do número de partículas. LISTER *et al.* [65] modificaram o método de HOUNSLOW *et al.* [64] dando maior flexibilidade à discretização, estendendo a progressão geométrica ($m_{i+1} = 2^{1/q} m_i$), o que permitiu o refinamento da malha utilizando um fator q.

A partir de então, surgiu uma série de trabalhos de extensão do método de HOUNSLOW *et al.* [64]. Contudo, o método de KUMAR e RAMKRISHNA [62] foi o que realmente superou o problema de inflexibilidade da malha. Essa abordagem concentra todas as informações de um intervalo $I_i = [m_i, m_{i+1}]$ em torno de um único ponto, denominado pivô ou abscissa ($\bar{m}_i$), tal que $m_i < \bar{m}_i < m_{i+1}$. Infelizmente, neste caso, os termos de agregação e quebra podem gerar partículas muito diferentes dos pivôs que são fixos, criando assim problemas de acurácia. Para superar esse problema é necessário refinamento de malha, o que ocasiona aumento do custo computacional.

Representando a função densidade numérica em um número finito de funções delta de Dirac, tem-se:

$$f(\mathbf{x}, m, t) \approx \sum_{i=1}^{n} N_i \delta(m - \bar{m}_i) \tag{3.2}$$

As Equações (3.3), (3.4) e (3.5) descrevem o MC dado por KUMAR e RAMKRISHNA [62] com pivô fixo e conservação dos momentos de ordem zero e um,

onde $\delta^*_{j,k}$ representa o delta de Kronecker,

$$\frac{\partial N_i(\mathbf{x},t)}{\partial t} = \sum_{\bar{m}_{i-1}\leq \bar{m}_j+\bar{m}_k\leq \bar{m}_{i+1}}^{j\geq k} \left[1-\frac{1}{2}\delta^*_{j,k}\right]\eta_{i,j,k}a(\bar{m}_j,\bar{m}_k)N_jN_k$$
$$-N_i\sum_{k}^{i_{max}} a(\bar{m}_i,\bar{m}_k)N_k + \sum_{j\geq k}^{i_{max}} \kappa_{i,k}b(\bar{m}_i)N_j - b(\bar{m}_i)N_i \tag{3.3}$$

em que,

$$\eta_{i,j,k} = \begin{cases} \frac{\bar{m}_{i+1}-(\bar{m}_j+\bar{m}_k)}{\bar{m}_{i+1}-\bar{m}_i} & \text{se} \quad \bar{m}_i \leq \bar{m}_j+\bar{m}_k \leq \bar{m}_{i+1} \\ \\ \frac{(\bar{m}_j+\bar{m}_k)-\bar{m}_{i-1}}{\bar{m}_i-\bar{m}_{i-1}} & \text{se} \quad \bar{m}_{i-1} \leq \bar{m}_j+\bar{m}_k \leq \bar{m}_i \end{cases} \tag{3.4}$$

e

$$\kappa_{i,k} = \int_{\bar{m}_i}^{\bar{m}_{i+1}} \frac{\bar{m}_{i+1}-m}{\bar{m}_{i+1}-\bar{m}_i}\bar{P}(m|\bar{m}_k)dm + \int_{\bar{m}_{i-1}}^{\bar{m}_i} \frac{m-\bar{m}_{i-1}}{\bar{m}_i-\bar{m}_{i-1}}\bar{P}(m|\bar{m}_k)dm \tag{3.5}$$

KUMAR e RAMKRISHNA [66] desenvolveram a técnica do pivô móvel que permite a flexibilização dos pivôs de acordo com os efeitos de quebra e agregação, ou seja, os pivôs se movem para manter a consistência dos momentos escolhidos. O novo equacionamento proposto por KUMAR e RAMKRISHNA [66] está apresentado abaixo.

$$\frac{\partial N_i(\mathbf{x},t)}{\partial t} = \sum_{m_i\leq \bar{m}_j+\bar{m}_k\leq m_{i+1}}^{j\geq k} \left[1-\frac{1}{2}\delta^*_{j,k}\right] a(\bar{m}_j,\bar{m}_k)N_jN_k$$
$$-N_i\sum_{k}^{i_{max}} a(\bar{m}_i,\bar{m}_k)N_k + \sum_{j\geq k}^{i_{max}} b(\bar{m}_i)N_jB^*_{i,j} - b(\bar{m}_i)N_i \tag{3.6}$$

$$\frac{\partial \bar{m}_i(\mathbf{x},t)}{\partial t} = \frac{1}{N_i}\sum_{m_i\leq \bar{m}_j+\bar{m}_k\leq m_{i+1}}^{j\geq k} [1-\frac{1}{2}\delta^*_{j,k}][(\bar{m}_j+\bar{m}_k)-\bar{m}_i]a(\bar{m}_j,\bar{m}_k)N_jN_k$$
$$-\frac{1}{N_i}\sum_{j\geq i}^{i_{max}} b(\bar{m}_i)N_j[B^{(r)}_{i,j}-\bar{m}_iB^*_{i,j}] \tag{3.7}$$

onde,

$$B^{(*)}_{i,j} = \int_{m_i}^{m_{i+1}} \bar{P}(m|\bar{m}_j)dm \tag{3.8}$$

$$B_{i,j}^{(m)} = \int_{m_i}^{m_{i+1}} m\bar{P}(m|\bar{m}_j)dm \tag{3.9}$$

NOPENS *et al.* [67] testaram os métodos com pivô fixo e pivô móvel, avaliando a velocidade de processamento dos métodos. A conclusão dos autores foi que o método de pivô fixo atinge mais rapidamente o estado estacionário, enquanto que o de pivô móvel é mais lento, apresentando maior custo computacional.

O MC com pivô fixo foi primeiramente desenvolvido para distribuições monovariadas. VALE e MCKENNA [7] e ALEXOPOULOS e KIPARISSIDES [68] estenderam essa metodologia de uma dimensão para *ndim* dimensões. Quando o MC é expandido para *ndim* dimensões, é necessário que o método seja internamente consistente com 2^{ndim} momentos. CHAKRABORTY e KUMAR [69] desenvolveram uma variação do MC usando abscissas fixas denominado *Minimal Internal Consistency MC*, que para *ndim* variáveis internas apenas $ndim + 1$ momentos devem ser preservados. A estratégia proposta foi utilizar malhas triangulares para problemas 2d, ao invés de quadradas, e malhas tetraédricas ao invés de hexa-estruturadas.

Sem dúvida o método MC é promissor em aplicações de CFD. Infelizmente, só garantem conservação dos momentos de ordem zero e um, que implica, em não garantir conservação de propriedades também importantes para simulações CFD, como por exemplo, diâmetro de partícula. Além disso, para que o MC tenha solução precisa é necessário um número excessivo de classes, fato que aumenta significativamente o custo computacional.

3.3 Método dos Momentos

Esse método faz uso dos momentos de distribuição para representar propriedades de um sistema particulado. Desta maneira, as propriedades relevantes da distribuição numérica são, então, calculadas a partir do conjunto de momentos de baixa ordem, uma vez que são suficientes para computar as principais propriedades físicas de uma dispersão (FRIEDLANDER [41]). O método dos momentos (MoM) foi introduzido por HULBURT e KATZ [38] e, na época, era considerada a mais promissora técnica de solução da PBE (YEOH e TU [10]). A ideia básica do método é transformar a PBE em termos dos momentos que darão importantes informações estatísticas da distribuição. O momento de ordem zero, por exemplo, representa a densidade numérica da população, sendo o volume a variável interna, o momento de primeira ordem é a fração da fase dispersa e os momentos fracionais, como os de $k = \frac{1}{3}$ e $k = \frac{2}{3}$, fornecem as informações associadas ao diâmetro médio e à área superficial média, respectivamente.

A principal vantagem do MoM está na sua simplicidade de implementação e

baixo custo computacional, que condensa as informações da distribuição em um pequeno grupo de momentos. Contudo, este método não tem uma forma geral fechada pois, para muitos casos, a equação referente ao momento μ_k envolve momentos de ordem superior, surgindo, assim, um problema de fechamento (RAMKRISHNA [14]). BARRET [70] idealizaram uma forma funcional para os momentos e aplicaram para um caso de agregação com condensação de partículas. Assumiram que os momentos podiam ser expressos como a exponencial de um polinômio de ordem p em k, ou seja, $ln(\mu_k) = c_0 + c_1 k^1 + \ldots + c_n k^n$. Seus coeficientes ($c_k$) são calculados ao resolver $p + 1$ equações diferenciais numericamente. Uma limitação desse método está na dificuldade de determinar momentos de ordem fracionada. Alguns autores (MARKATOU *et al.* [71], FRENKLACH e WANG [72]) desenvolveram uma variação do MoM utilizando funções de interpolação (MoMIC) para obter os momentos de ordem fracionada. Infelizmente, a necessidade de fechamento é uma restrição severa deste método, e portanto, é o motivo pelo qual o mesmo não ganhou força junto à comunidade acadêmica.

3.4 Método dos Resíduos Ponderados

O método dos resíduos ponderados (MRP) é uma técnica geral de solução de equações diferenciais parciais. A ideia dessa abordagem é fazer com que uma função aproxime da melhor forma possível a solução procurada, minimizando os resíduos da equação que se almeja resolver. A solução é definida pela expansão funcional da equação em coeficientes e funções base conhecidas. Os coeficientes da expansão são obtidos a partir da minimização dos resíduos, que formam sistemas de equações algébricas ou diferenciais ordinárias. Os variados MRP diferem entre si basicamente no uso de funções locais ou globais, ou pela forma com que os resíduos são minimizados (PINTO e LAGE [73]). Geralmente, técnicas de natureza local requerem um grande número de pontos de discretização. Isto leva a um número muito grande de equações algébricas, mas é um método simples e pode ser discretizado facilmente, além de ser eficiente para problemas dinâmicos. Em comparação, os métodos de natureza global tem derivação mais complexa e são geralmente aplicados a problemas estacionários. De forma geral, as variantes dos MRP estão associadas as suas funções peso escolhidas. Por exemplo, escolhas para funções peso que incluem as funções delta de Dirac, caracterizam os métodos de colocação, ou por exemplo, a função peso como a própria base da expansão funcional, caracterizam os métodos de Galerkin.

GELBARD e SEINFELD [74] utilizaram a técnica de colocação ortogonal para solução da PBE. Incorporaram polinômios cúbicos para o método de colocação ortogonal. Em seguida, EYRE *et al.* [75] resolveram o mesmo problema com o método

de colocação, só que a função base utilizada foi *B-spline*. CHEN *et al.* [76] desenvolveram o método wavelet-Galerkin para resolver a PBE monovariada em termos de tamanhos de partículas. NICMAINS e HOUNSLOW [77] aplicaram elementos finitos para resolver a PBE estacionária e concluíram que essa metodologia é muito sensível ao refino de malha.

LIU e CAMERON [78] propuseram resolver a PBE dinamicamente utilizando como base funções wavelet para tratar um problema com crescimento, nucleação e agregação. MAHONEY e RAMKRISHNA [79] propuseram o uso do método Galerkin em elementos finitos para sistemas com precipitação e verificaram dificuldades associadas a descontinuidades nos integrandos de quebra e agregação. RIGOPOULOS e JONES [80] resolveram a PBE dinâmica unidimensional usando colocação ortogonal com elementos finitos.

DORAO e JAKOBSEN [81] compararam as formulações MoM-MRP e QMoM (será descrito na Seção (3.5)) para um problema de quebra pura com domínio limitado ([0,1]). O MoM-MRP consiste em utilizar polinômios interpoladores de Lagrange como função teste, $\pi_i(m)$. Expressando a função densidade numérica como a combinação linear dessas funções teste, onde c_i são os coeficientes da expansão, chega-se a:

$$f(\mathbf{x}, m, t) = \sum_{i=1}^{n} c_i(t)\pi_i(m) \tag{3.10}$$

DORAO e JAKOBSEN [81] desenvolveram aplicações do método dos mínimos quadrados à solução da PBE. O mesmo consiste em minimizar o quadrado do resíduo em um domínio computacional. Eles resolveram equações integro-diferenciais em um problema de quebra pura. Neste trabalho, compararam problemas transiente e estacionário frente as suas respectivas soluções analíticas, e verificaram que o método tem ótimas características, como alta taxa de convergência e controle da acurácia pelo número de coeficientes da expansão funcional. Eles não só compararam os métodos MoM-MWR e QMoM, mas também destacaram que esse métodos são idênticos, fato que é errôneo [29]. Essa informação é errada porque DORAO e JAKOBSEN [81] se basearam em um polinômio interpolador de Lagrange como função peso no MoM-MWR. Seu uso implica em não calcular exatamente os $2N$ primeiros momentos a partir de polinômios interpoladores de Lagrange até ordem N. Fato que a princípio ocorre no QMoM.

ZHU *et al.* [82] testaram uma abordagem semelhante, contudo para um problema com agregação pura, e propuseram um procedimento iterativo que combina o método de Picard e Newton para reduzir a sensibilidade do chute inicial e acelerar a taxa de convergência. DORAO e JAKOBSEN [83] utilizaram o método dos mínimos quadrados com elementos finitos com malha adaptativa para modelar um reator químico cilíndrico unidimensional. O método de mínimos quadrados com

adaptatividade *hp* utiliza os erros locais dos elementos da malha como critério de refinamento. O refinamento nesse método pode ser feito de duas formas, ora pelo refinamento da malha (h), ora pelo aumento da ordem do polinômio utilizado na expansão da FDN (p).

Em casos onde há quebra pura, a solução da PBE pelo MRP pode não conservar a massa de partículas. Por esse motivo, ZHU *et al.* [84] aplicaram o método dos mínimos quadrados à PBE, colocando a equação de conservação de massa da fase dispersa como uma restrição de igualdade. Examinaram essa metodologia frente a um escoamento 2D com o modelo de quebra de Martinez-Bazán [85] e obtiveram melhorias na conservação de massa das partículas em relação aos MRP tradicionais. No entanto, nessa abordagem existem alguns inconvenientes como o aparecimento dos multiplicadores de Lagrange e o parâmetro adicional de compatibilidade para obtenção de uma única solução.

O MRP é um método muito robusto, não obstante, na presença dos termos de agregação, ele é extremamente custoso. O MRP necessita de uma solução iterativa para obtenção dos coeficientes da expansão funcional da f, o que inviabiliza seu uso em aplicações que envolvem simulações CFD.

3.5 MoM Fechados por Quadratura

Baseado no conceito do MoM, MCGRAW [86] propôs o método intitulado *Quadrature Method of Moments* (QMoM), onde as equações de evolução de momentos são aproximadas por uma quadratura de Gauss-Christoffel, cuja máxima exatidão é atingida quando o integrando, neste caso representados pelos termos fontes da PBE, é um polinômio de ordem $2N-1$ [87], sendo N o número de pontos de quadratura. Essa metodologia soluciona o problema de fechamento para uma ampla variedade de problemas, sem a necessidade de assumir formas matemáticas especiais para a densidade numérica. A ideia desse método é determinar os pesos (ω_α) e abscissas (m_α) a cada instante, a partir dos momentos de menor ordem de uma distribuição. Assim, as integrais da função densidade numérica podem ser representadas usando uma discretização da função de distribuição em termos dos pontos de quadratura. Assim no QMoM, as abscissas se movimentam livremente pelo domínio conforme os momentos variam.

A formulação proposta por MCGRAW [86] foi inicialmente desenvolvida para problemas monovariados sem dependência com o espaço físico. A aproximação por quadratura dos momentos é descrita na Equação (3.11), onde tem-se disponível n pontos de quadratura para $2n$ momentos.

$$\mu_k = \int_{-\infty}^{\infty} m^k f(\mathbf{x}, m, t) dm = \sum_{\alpha=1}^{n} m_\alpha^k \omega_\alpha \tag{3.11}$$

Mesmo não eliminando os erros associados à aproximação por quadratura, esse método tem a vantagem de não depender de nenhum tipo de interpolação para fechar o sistema de equações para os momentos. No QMOM, os $2n$ primeiros momentos podem ser obtidos por n pesos e n abscissas, que são calculados a partir dos momentos, já que são eles as variáveis calculadas. Se utilizarmos a Equação (3.11) para determinar os pesos e abscissas, será necessário um método de solução de sistemas de equações não lineares. Essa abordagem é computacionalmente custosa ou mesmo inviável. Outra forma de obter esse pesos e abscissas no caso monovariado, é utilizando o algoritmo Produto Diferença (PD) [88] ou o algoritmo de Chebyshev modificado (CM) [89]. No entanto, esses métodos tem robustez muito limitada devido o mal-condicionamento da matriz de geração dos pesos e abscissas. Essa limitação também restringe o número de pontos quadratura possíveis em sua aplicação, o que afeta diretamente o fechamento desse termo por quadratura [90]. Em outras palavras, o método QMoM evolui os momentos da função densidade numérica, calculando os pesos e abscissas da quadratura a partir ou do algoritmo PD [88] ou do algoritmo CM [89]. Como por exemplo a Equação 3.12 que representa a evolução dos momentos gerados a partir da Equação 2.21 considerando apenas os efeitos de quebra e agregação.

$$\begin{aligned}\frac{\partial \mu_k(m, \mathbf{Y}, t)}{\partial t} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} [(m_i - m_j)^k - m_i^k] a(m_i, m_j) \omega_i \omega_j \\ + \sum_{i=1}^{n} b(m_i) \omega_i [\nu(m_i) \bar{P}_k(m_i) - m_i^k]\end{aligned} \tag{3.12}$$

onde

$$\bar{P}_k(m_i) = \int_0^{m_i} m^k P(m|m_i) dm \tag{3.13}$$

MCGRAW [86] testou seu método com um problema monovariado de crescimento de aerosóis e comparou seus resultados com o trabalho de HULBURT e KATZ [38]. Neste trabalho, o método QMoM avançava os 6 primeiros momentos acompanhando a solução exata, diferentemente do método MoM proposto por HULBURT e KATZ [38] que só acompanhava os dois primeiros momentos.

WRIGHT *et al.* [91] foram os primeiros a aplicar o QMoM para um problema multivariado. Neste trabalho estudaram a dinâmica de uma população de nanopartículas inorgânicas, considerando apenas o efeito de agregação. Os autores utilizaram volume de partículas e área superficial como variáveis internas, e obtiveram erros

menores que 1% frente a solução utilizando o método de Monte Carlo.

MCGRAW e WRIGHT [92] derivaram um novo método de fechamento baseado no QMoM, denominado de Transformação da Matriz Jacobiana (TMJ). Essa nova abordagem permite obter diretamente os pesos e abscissas de quadratura evitando as dificuldades numéricas do algoritmo PD.

YOON e MCGRAW [93] desenvolveram uma formulação multivariada do QMoM usando o algoritmo de análise dos componentes principais (PCA) e a aplicaram a um problema de agregação pura. O PCA é um método que usa os momentos de até segunda ordem para determinar as direções de maior variança da variável interna [94]. O PCA obtém as combinações lineares das variáveis de distribuição que têm correlação nula. Neste método, as coordenadas principais são escolhidas para obter os pesos e abscissas multidimensionais, usando o produto cartesiano de quadratura unidimensional. YOON e MCGRAW [93] confrontaram seus resultados com os de WRIGHT *et al.* [91] e obtiveram resultados concordantes.

FAVERO *et al.* [95] testaram os métodos de inversão DPCM, CQMoM e PCA. Os autores também introduziram o uso do ICA e a combinação entre vários métodos de inversão buscando maior robustez e acurácia. Para os casos multivariados, várias técnicas de inversão foram proposta, tais como: otimização direta, DPCM [96] (*Direct Cartesian Product Method*), CQMoM [97](*Conditional Quadrature Method of Moments*), ICA [98] (*Independent Component Analysis*) e PCA (*Principal Component Analysis*). Não obstante, até então nenhuma metodologia está bem consolidada. Basicamente, otimização direta é determinar os pesos e abscissas a partir de uma função objetivo gerada pela Equação (3.11). Cabe salientar que para o caso multivariado o fechamento do sistema de equações não lineares pode não existir, portanto, faz-se necessário o uso de otimização para inversão dos momentos multivariados. O DPCM [96] separa a função densidade numérica em funções marginais. Desta forma, os momentos mistos são representados pelo produto cartesiano dos momentos "puros", e portanto, também representados pelo produto cartesiano dos pontos de quadratura unidimensionais. No caso do CQMoM [97], a FDN é separada no produto de funções condicionais e marginal, $f(m_1, m_2) = f(m_1)f(m_1|m_2)$, onde $f(m_1)$ é sua distribuição marginal e $f(m_1|m_2)$ é a sua distribuição condicional. Assim, a FDN tem os momentos mistos calculados a partir do produto dos seus momentos condicionais e momentos marginais. No processo de inversão, os pontos de quadratura associados a função marginal são calculados por um algoritmo de inversão unidimensional. Os momentos condicionais são obtidos por soluções de sistema lineares gerados pela relação entre os momentos mistos, condicionais e marginais. A partir dos momentos condicionais, uma inversão unidimensional determina os pontos de quadratura na direção de m_2. A desvantagem dessa metodologia está na possibilidade de geração de um conjunto de momentos não realizáveis. Por fim o ICA que

nada mais é que uma extensão do PCA, sendo que, ao invés de achar direções descorrelacionadas, procura-se a transformação linear que maximize a independência das novas coordenadas. Note que, essa transformação viabiliza o uso do DPCM sem a preocupação com a dependência entre as variáveis internas. FAVERO *et al.* [95] concluíram que o método DPCM não é apropriado quando os momentos mistos tem direções correlacionadas. Observaram também que o PCA e o ICA são os mais robustos e acurados até momentos de segunda ordem, no entanto, o ICA se mostrou de alguma forma um pouco mais acurado. Apesar do método CQMoM ter acurácia até terceira ordem, o mesmo pode gerar momentos não realizáveis. Visando melhorar a acurácia e a robustez, FAVERO *et al.* [95] conduziram a combinação PCA-CQMoM e ICA-CQMoM. Com essa mescla, a inversão pode atingir acurácia de até terceira ordem sem produzir momentos não realizáveis.

Seguindo na linha desenvolvimento dos métodos fechados por quadratura, MARCHISIO *et al.* [99] aprimoraram o método QMoM considerando efeitos de quebra e agregação. Os autores investigaram a performance do QMoM com 10 casos testes, combinando diferentes núcleos de quebra e agregação, funções de densidade de probabilidade e condições iniciais. As predições foram comparadas com a solução da PBE fornecida por VANNI [100], que utilizou cerca de 2000 classes para solucionar a PBE. Concluíram que o método QMoM necessitou de poucos momentos para obter a mesma acurácia que o método das classes. A formulação QMoM proposta por MARCHISIO *et al.* [99] para o avanço dos $2n$ momentos é apresentada na Equação 3.12.

MARCHISIO *et al.* [101] implementaram o método QMOM no *software* comercial de fluidodinâmica computacional FLUENT via UDF[11] usando seu próprio integrador para calcular as equações evolutivas dos momentos. Os resultados obtidos foram confrontados com resultados experimentais e verificou-se uma concordância apenas qualitativa. WAN *et al.* [102] avaliaram o método QMoM implementado no ANSYS FLUENT, previamente implementado por MARCHISIO *et al.* [101], em um tanque de mistura 2D, e obtiveram erros menores que 1% frente a solução analítica do problema.

SU *et al.* [103] desenvolveram o método QMoM com o fator de ajuste p, redefinido assim os momentos para a Equação 3.14, e por consequência, modificando também a equação de evolução dos momentos.

$$\mu_k = \int_{-\infty}^{\infty} m^{k/p} f(\mathbf{x}, m, t) dm \tag{3.14}$$

Estes autores mostraram ainda em seu trabalho que a acurácia varia em função do valor p. Contudo, esse método não parece promissor, visto que para cada caso o

[11] *User Define Function* são funções escritas em C executadas pelo *software ANSYS FLUENT*.

fator p deve ser estipulado.

O QMoM possui dois fatores limitantes: a aplicação desse método a problemas multivariados não é simples e nem eficiente. Além disso, no caso monovariado, o número de pontos de quadratura pode não ser suficiente para calcular os termos integrais da PBE, acumulando, assim, o erro ao longo do tempo [104].

A grande evolução do QMoM foi o surgimento do *Direct Quadrature Method of Moments* (DQMoM), desenvolvido por MARCHISIO e FOX [104]. O DQMoM se fundamenta na ideia de acompanhar as variáveis primitivas de quadratura Gaussiana. Em outras palavras, as equações de evolução avançam os pesos e abscissas presentes na aproximação de quadratura ao invés de acompanhar os momentos da distribuição. A função distribuição é representada por um número finito de funções delta de Dirac multidimensionais:

$$f(\mathbf{x}, \mathbf{m}, t) \approx \sum_{\alpha=1}^{n} \omega_\alpha \delta[\mathbf{m} - \mathbf{m}_\alpha] \tag{3.15}$$

$$\delta[\mathbf{m} - \mathbf{m}_\alpha] = \prod_{j=1}^{Ns} \delta[m_j - m_{j,\alpha}] \tag{3.16}$$

em que, n é o número de pesos e abscissas, e N_s é o número de dimensões da variável interna m_j. Esta forma funcional pode ser interpretada como um conjunto de n fases dispersas, sendo cada fase caracterizada pela densidade numérica (ω_α). Por isso, esse método é considerado interessante em problemas de CFD. Para derivar as equações do método DQMoM, deve-se substituir a aproximação apresentada na Equação 3.15 na Equação (2.21) que, para caso monovariado com $\mathbf{D_x} = \sqrt{D_x}\mathbf{I}$, sendo D_x uma constante, leva a:

$$\begin{aligned}
&\sum_{\alpha=1}^{n} \delta[m - m_\alpha][\frac{\partial \omega_\alpha}{\partial t} + \bigtriangledown_x.(\mathbf{u}_\alpha \omega_\alpha) - \bigtriangledown_x.[D_x \bigtriangledown_x \omega_\alpha]] \\
&- \sum_{\alpha=1}^{n} \delta'[m - m_\alpha][\frac{\partial \varsigma_\alpha}{\partial t} + \bigtriangledown_x.(\mathbf{u}_\alpha \varsigma_\alpha) - \bigtriangledown_x.[D_x \bigtriangledown_x \varsigma_\alpha]] \\
&+ \sum_{\alpha=1}^{n} \delta'[m - m_\alpha] m_\alpha [\frac{\partial \omega_\alpha}{\partial t} + \bigtriangledown_x.(\mathbf{u}_\alpha \omega_\alpha) - \bigtriangledown_x.[D_x \bigtriangledown_x \omega_\alpha]] \\
&- \sum_{\alpha=1}^{n} \delta''[m - m_\alpha][D_x \omega_\alpha (\bigtriangledown_x m_\alpha).(\bigtriangledown_x m_\alpha)] = H(\mathbf{x}, m, t)
\end{aligned} \tag{3.17}$$

onde $\varsigma_\alpha = \omega_\alpha m_\alpha$ é a abscissa ponderada, u_α é a velocidade da fase α, δ' e δ'' são as derivadas primeira e segunda da função delta de Dirac e $H(\mathbf{x}, m, t)$ é a taxa líquida de nascimento e morte. Define-se os termos fontes da equação de transporte do

DQMoM por:

$$S_{\omega_\alpha} \doteq [\frac{\partial \omega_\alpha}{\partial t} + \bigtriangledown_x.(\mathbf{u}_\alpha \omega_\alpha) - \bigtriangledown_x.[D_x \bigtriangledown_x \omega_\alpha]] \tag{3.18}$$

$$S_{e_\alpha} \doteq [\frac{\partial \varsigma_\alpha}{\partial t} + \bigtriangledown_x.(\mathbf{u}_\alpha \varsigma_\alpha) - \bigtriangledown_x.[D_x \bigtriangledown_x \varsigma_\alpha]] \tag{3.19}$$

e, também:

$$C_\alpha \doteq D_x \omega_\alpha (\bigtriangledown_x m_\alpha).(\bigtriangledown_x m_\alpha) \tag{3.20}$$

Aplicando-se a integral $\int_{-\infty}^{\infty} m^k(.)dm$ à Equação (3.17) e usando as definições (3.18), (3.19) e (3.20), obtém-se a Equação (3.21).

$$(1-k)\sum_{\alpha=1}^{n} m_\alpha^k S_{\omega_\alpha} + k\sum_{\alpha=1}^{n} m_\alpha^{k-1} S_{e_\alpha} = \bar{H}_k(\mathbf{x}, m, t) + \bar{C}_k \tag{3.21}$$

onde,

$$\bar{C}_k = k(k-1)\sum_{\alpha=1}^{n} m_\alpha^{k-2} C_\alpha \tag{3.22}$$

e

$$\bar{H}_k = \int_{-\infty}^{\infty} m^k H(\mathbf{x}, m, t)dm \tag{3.23}$$

MARCHISIO e FOX [104] desenvolveram e testaram sua formulação para os problemas de crescimento, dispersão, nucleação e agregação homogêneos, difusão pura monovariada, além de analisar problemas bivariados de agregação homogênea (não depende de variáveis externas) e de crescimento.

3.6 Métodos Híbridos

Em 2005, BOVE *et al.* [105] desenvolveram o método PPDC, *Parallel Parent Daughter Classes*, que consiste em dividir a discretização da PBE em várias malhas para as partículas. Essa métodologia decompõe os operadores de discretização de acordo com os efeitos de nascimento e morte. BOVE *et al.* [105] representaram a função densidade numérica em um número finito de funções delta representando os pivôs como no MC. Para tanto, usaram três classes de malhas simultaneamente. A primeira é a malha usada para as classes de partículas mãe, a segunda engloba as M malhas usadas para as partículas filhas geradas por quebra e o terceiro tipo, com $M(M+1)/2$ malhas, são utilizadas para as filhas geradas por agregação. Essas malhas são paralelas no espaço de variáveis internas. Desta forma, é possível que os processos de quebra e agregação sejam calculados separadamente e posteriormente sobrepostos. Os autores discretizaram o sistema de equações dos momentos seccionais de ordem zero usando o método de Euler explícito e obtiveram as Equações (3.24), (3.25) e (3.26).

$$\frac{N_i(t^{n+1}) - N_i(t^n)}{\Delta t} = -N_i(t^n) \sum_{j=0}^{M} a(\bar{m}_i, \bar{m}_j) N_j(t^n) - b(\bar{m}_i) N_i(t^n),$$
$$i = 1, 2, \ldots, M \qquad (3.24)$$

$$\frac{A_{i,j}(t^{n+1}) - A_{i,j}(t^n)}{\Delta t} = (1 - \frac{1}{2}\delta_{i,j}) a(\bar{m}_i, \bar{m}_j) N_i(t^n),$$
$$i, j = 1, 2, \ldots, M, \qquad j \geq i \qquad (3.25)$$

$$\frac{B_k^{(i)}(t^{n+1}) - B_k^{(i)}(t^n)}{\Delta t} = \nu_i b(\bar{m}_i) N_i(t^n) \int_{m_k}^{m_{k+1}} P(m|\bar{m}_i) dm,$$
$$i = 1, 2, \ldots, M, \qquad k = 1, \ldots, NB(i) \qquad (3.26)$$

onde N_i, $A_{i,j}$, $B_k^{(i)}$ são, respectivamente, os pesos que representam as densidades numéricas das partículas mãe i, filhas ij geradas por agregação de mães i e j, e para partículas k geradas por quebra de uma partícula mãe i. Em seguida, soma-se as distribuições obtidas no instante t^{n+1}.

$$f_E(m, t^{n+1}) \approx \sum_{i=1}^{M} N_i(t^{n+1}) \delta(m - \bar{m}_i) + \sum_{i=1}^{M} \sum_{j=1}^{i} A_{i,j}(t^{n+1}) \delta(m - y_{i,j})$$
$$+ \sum_{i=1}^{M} \sum_{k=1}^{NB(i)} B_k^{(i)}(t^{n+1}) \delta(m - z_k^{(i)}) \qquad (3.27)$$

em que, m_i, y_{ij} e $z_k^{(i)}$ são os pivôs das partículas mãe, das filhas geradas por agregação e por quebra, respectivamente. Ao final do intervalo de integração, deve-se recaracterizar as novas partículas mães. BOVE *et al.* [105] recomendaram o uso dos momentos de f_E obtidos usando a Equação (3.27) para obter a quadratura Gaussiana empregando o algoritmo PD proposto por GORDON [88]. Em BOVE *et al.* [105], a técnica PPDC foi validada frente a solução analítica proposta por SCOTT [106], a qual só leva em consideração os efeitos de agregação. Foi também confrontado com a solução analítica proposta por MCCOY e MADRAS [46]. Além de comparar casos com solução analítica, BOVE *et al.* [105] confrontaram seu método com o método das classes com pivôs fixos de KUMAR e RAMKRISHNA [62]. Para os casos com solução analítica, o PPDC se mostrou concordante, mesmo com um número pequeno de classes. Contudo, para o caso simulado por KUMAR e RAMKRISHNA [66], os resultados foram divergentes.

3.7 Comparação entre os Métodos

SILVA *et al.* [107] compararam os métodos PPDC, QMoM, DQMoM e CM quanto a acurácia e eficiência. Para avaliar essas técnicas, eles confrontaram seus resultados com a solução analítica proposta por MCCOY e MADRAS [46], testando três casos: um com quebra dominante, outro com agregação dominante e por último o caso invariante, onde os efeitos de nascimento e quebra são equivalentes. Os autores observaram que o método PPDC tem uma baixa taxa de convergência e para o mesmo número de pontos de quadratura, os método QMoM e DQMoM são mais acurados, sendo que os últimos têm a mesma acurácia. No entanto, o método DQMoM se mostrou mais eficiente computacionalmente.

Objetivando reduzir os erros de quadratura dos métodos QMoM e DQMoM e aumentar a acurácia desses métodos, surge a ideia de utilizar momentos generalizados ao invés de momentos padrões na derivação desses dois métodos. Nessa linha de desenvolvimento, PISKUNOV e GOLUBEV [108] utilizaram momentos gerados a partir de monômios com potências fracionárias para derivar o QMoM. Todavia, apenas concluíram que seu método era numericamente equivalente ao QMoM tradicional.

GROSCH *et al.* [109] revitalizam a ideia de utilizar formas funcionais diferentes dos monômios para derivar o método QMoM. Neste trabalho, os autores reformularam o método QMoM utilizando os polinômios de Laguerre para gerar os momentos generalizados. Para avaliar essa nova metodologia os autores comparam seu resultado com um problema de dissolução solucionado pelo método de Garlekin com malha adaptativa implementado para o *software* Parsival (*Particle Size Evolution*). GROSCH *et al.* [109] compararam a performance desses métodos, e observaram que o método DQMOM é mais rápido e robusto utilizando momentos padrões. Nesse artigo os métodos que utilizaram polinômios de Laguerre não foram eficientes, então os autores recomendam o uso do método DQMoM para resolver a equação de balanço populacional.

SU *et al.* [110] reviveram a ideia do fator de ajuste, e o aplicaram ao método DQMoM (ADQMoM). O ADQMoM consiste na redefinição do DQMoM a partir da Equação (3.14), gerando assim um sistema linear dependente do fator de ajuste p. Neste trabalho, SU *et al.* [110] criaram um mecanismo de obtenção desse fator a partir da minimização do número de condicionamento do sistema linear gerado. Obtiveram resultados interessantes, contudo apesar deste método ser relativamente mais robusto que o DQMoM tradicional, o mesmo ainda sofre com os problemas do DQMoM, como por exemplo, os erros associados a quadratura.

Ainda na linha de desenvolvimento de métodos baseados em momentos generalizados, SANTOS *et al.* [111] analisaram o método DQMoGeM (*Direct Quadrature*

Method of Generalized Moments) utilizando os polinômios de Laguerre e Legendre para geração dos momentos generalizados. SANTOS *et al.* [111] observaram que o DQMoGeM se mostrou mais adequado para problemas com domínio finito utilizando polinômio de Legendre, o qual difere do resultado obtido por GROSCH *et al.* [109] utilizando polinômios de Laguerre. Portanto, SANTOS *et al.* [111] concluíram que para domínio finito utilizar base de Legendre aumenta robustez do método, uma vez que altera o sistema linear de tal forma que reduz o seu número de condicionamento. Contudo, o DQMoGeM ainda carrega alguns dos problemas presentes no DQMoM tradicional, como por exemplo os erros de quadratura provenientes dos termos de quebra e agregação.

ATTARAKIH *et al.* [112] criaram um método híbrido chamado SQMoM (*Sectional Quadrature Method of Moments*) que mescla as características do método das classes e do QMoM. Neste método o domínio da variável interna é discretizado em M_s intervalos, e cada intervalo é representado por uma partícula denominada primária com abscissas e pesos, $\bar{d}^{<i>}$ e $\bar{\omega}^{<i>}$, respectivamente. Em cada intervalo são definidas partículas secundárias $d_j^{<i>}$ e $\omega_j^{<i>}$, $i = 0, \ldots, M_s$, $j = 0, \ldots, M_p$, onde M_p é o número de pontos de quadratura dentro do intervalo i.

Essas partículas primárias, que representam um intervalo, são obtidas a partir das médias ponderadas das partículas secundárias na suas respectivas seções, como nas Equações (3.28) e (3.29).

$$\bar{\omega}^{<i>} = \frac{\sum_{j=0}^{M_p} \omega_j^{<i>}}{\Delta d_i} \tag{3.28}$$

$$\bar{d}^{<i>} = \frac{\sum_{j=0}^{M_p} \omega_j^{<i>} d_j^{<i>}}{\sum_{j=0}^{M_p} \omega_j^{<i>}} \tag{3.29}$$

$$\Delta d_i = d_{i+1/2} - d_{i-1/2} \tag{3.30}$$

Já as partículas secundárias são obtidas pela aplicação do QMoM em cada intervalo ($[d_{i-1/2}, d_{i+1/2}]$) usando momentos seccionais definidos pela Equação (3.31). Note que o número M_p de pontos de quadratura permite a conservação de até $2M_p$ momentos seccionais.

$$\mu_k^{<i>} = \int_{d_{i-1/2}}^{d_{i+1/2}} d^k f(d) \partial d \tag{3.31}$$

$$f^{<i>}(d,t) \approx \sum_{j=1}^{M_p} \omega_j^{<i>} \delta[d - d_j^{<i>}] \tag{3.32}$$

Por fim, pode-se definir a função densidade numérica global pela Equação 3.33.

$$f(d,t) \approx \sum_{i=1}^{M_s} \bar{\omega}^{<i>} \delta[d - \bar{d}^{<i>}] \tag{3.33}$$

$$\mu_k^{<i>} = \sum_{j=0}^{M_p} \omega_j^{<i>} (d_j^{<i>})^k \tag{3.34}$$

Almejando evitar os problemas numéricos relacionados a inversão dos momentos seccionais em pesos e abscissas, ATTARAKIH *et al.* [112] decidiram utilizar apenas dois pontos de quadratura em cada intervalo, uma vez que com apenas dois pontos a inversão é analítica. Ademais, propuseram a utilização de pesos iguais visando evitar problemas numéricos associados a possibilidade do processo físico se tornar monodisperso.

ATTARAKIH *et al.* [112] testaram o SQMoM com cinco casos de quebra e agregação. Esse método se mostrou bem eficiente em termos de acurácia. Além disso, este é um método bem flexível, uma vez que com apenas uma partícula primária o mesmo se torna no método QMoM tradicional. Outro ponto importante é a facilidade de acoplar o método SQMoM à CFD utilizando a metodologia MUSIG [113]. ATTARAKIH *et al.* [112] demostraram, que com o aumento das seções, a acurácia do método aumenta. Portanto, o número de intervalos é um parâmetro útil para o controle da precisão.

Apesar desse método ser mais eficiente que o QMoM, ele ainda possui algumas limitações herdadas do mesmo: os erros de aproximação por quadratura ainda existem e a incapacidade de convectar as partículas com velocidades diferentes no espaço de variáveis externas. Além disso, o SQMoM não prevê o fluxo convectivo entre as seções, ou seja, falha em modelar o crescimento de partículas. BRUNS e EZEKOYE [114] estenderam o SQMoM para o DQMoM Sectional (SDQMoM). A ideia é semelhante ao SQMoM, no entanto, ao invés de equações de momentos seccionais, tem-se pesos e abscissas seccionais sendo transportados em cada seção.

STRUMENDO e ARASTOOPOUR [115] derivaram o método FCMoM (*Finite size domain Complete set of trial functions Method Of Moments*). Para esse método, propuseram trabalhar com a PBE no domínio finito $[d_{min}, d_{max}]$, redefinindo a PBE para este intervalo:

$$\frac{\partial f(\bar{d},t)}{\partial t} - \frac{\partial f(\bar{d},t)}{\partial \bar{d}} \frac{1}{(d_{max} - d_{min})} \left(\left[\frac{\partial d_{min}}{\partial t} + \frac{\partial d_{max}}{\partial t} \right] + \bar{d} \left[\frac{\frac{\partial d_{max}}{\partial t} - \partial d_{min}}{\partial t} \right] \right)$$

$$+ \frac{2}{(d_{max} - d_{min})} \left[\frac{\partial G(\bar{d},t) f(\bar{d},t)}{\partial \bar{d}} \right] = B(\bar{d},t) - D(\bar{d},t) \tag{3.35}$$

A Equação 3.35 representa a PBE homogênea monovariada no domínio finito $[d_{min}, d_{max}]$, em função da variável transformada $\bar{d}$, dada pela Equação 3.36, onde $G(\bar{d},t)$, $B(\bar{d},t)$ e $D(\bar{d},t)$ são os termos de crescimento, nascimento e morte (quebra e agregação) em termos da variável interna transformada $\bar{d}$.

$$\bar{d} = \frac{d - (d_{max} + d_{min})/2}{(d_{max} - d_{min})/2} \tag{3.36}$$

Semelhante ao MRP, a função densidade numérica é aproximada por uma expansão em termos de uma base ortonormal $\phi_i(\bar{d})$. Se a base ortonormal for os polinômios de Legendre, os coeficientes da expansão são determinados a partir dos momentos da FDN, de acordo com a Equação 3.38.

$$f(\bar{d},t) \approx \sum_{i=1}^{n} c_i(t)\phi_i(\bar{d}) \tag{3.37}$$

$$c_i = \sqrt{\frac{2n+1}{2}} \frac{1}{2^n} \sum_{j=0}^{i} (-1)^{i-j} \frac{2j!}{(2j-i)!} \left[\frac{1}{(i-j)!j!}\right] \mu_{2j-i} \tag{3.38}$$

Por fim, as Equações dos momentos (3.39) são deduzidas a partir da equação transformada (3.35). As equações dos momentos são computadas concomitante a equação de movimentação do contorno, geralmente calculada a partir da taxa de crescimento.

$$\begin{aligned}
&\frac{\partial \mu_i}{\partial t} - \left(\left[f_1 - (-1)^i f_{-1}\right] - i\mu_{i-1}\right) \frac{1}{(d_{max} - d_{min})} \left(\left[\frac{\partial d_{min}}{\partial t} + \frac{\partial d_{max}}{\partial t}\right]\right) \\
&- \left(\left[f_1 - (-1)^{i+1} f_{-1}\right] - (i+1)\mu_i\right) \frac{1}{(d_{max} - d_{min})} \left(\left[\frac{\partial d_{max}}{\partial t} - \frac{\partial d_{min}}{\partial t}\right]\right) \\
&+ \frac{2}{(d_{max} - d_{min})} \left(\left[G_1 f_1 - (-1)^i G_{-1} f_{-1}\right] - i \int_{-1}^{1} G(\bar{d}) f(\bar{d}) \bar{d}^{i-1} \partial \bar{d}\right) = \\
&= \int_{-1}^{1} \bar{d}^i \left[B(\bar{d},t) - D(\bar{d},t)\right] \partial \bar{d}
\end{aligned} \tag{3.39}$$

onde, os termos f_1 e f_{-1} são os valores da função densidade numérica transformada nos limites superior e inferior e os termos G_1 e G_{-1} são os valores dos termos de crescimento nos limites superior e inferior do domínio.

STRUMENDO e ARASTOOPOUR [115] testaram casos com crescimento (constante, linear e difusão controlada), agregação e crescimento simultâneos, nucleação e dissolução, e obtiveram bons resultados. Contudo, o mais importante dessa técnica

é a possibilidade de movimentar o contorno e a reconstrução da função distribuição a cada instante de tempo de forma razoavelmente acurada. STRUMENDO e ARASTOOPOUR [116] ampliaram o FCMoM para escoamento não homogêneo, e corroboraram a validade do método para aplicações que envolvem CFD. Note que o uso de base polinômiais não monotônicas pode gerar momentos não realizáveis, e desta forma, não representar corretamente a FDN do sistema polidisperso.

Em 2011, surge o método CQMoM (*Cumulative Quadrature Method Of Moments*), que utiliza os momentos cumulativos da distribuição, definidos na Equação (3.40) [117–119].

$$\mu_k(x) = \int_0^x m^k f(m) dm, \qquad k = 0, \cdots, 2Np - 1 \tag{3.40}$$

De forma semelhante aos outros métodos fechados por quadratura a função densidade numérica é representada por uma soma de distribuições delta de Dirac baseadas nos pesos e abscissas, que são obtidos também a partir do próprio algoritmo PD, CM ou pela inversão analítica dos momentos (apenas com dois pontos de quadratura).

$$f(x) = \sum_{i=1}^{Np} \omega_i(x) \delta \left[m - m_i(x) \right] \tag{3.41}$$

$$\mu_k(x) = \sum_{i=1}^{Np} \omega_i(x) \left[m_i(x) \right]^k \tag{3.42}$$

A dedução desse método é semelhante ao QMoM, contudo ao invés de utilizar momentos padrões, aplica-se o operador dos momentos cumulativos na equação de balanço populacional. Obtém-se então a equação evolutiva dos momentos cumulativos em sua forma Lagrangeana descrita na equação (3.43).

$$\frac{D\mu_k(x)}{Dt} = k \sum_{i=0}^{Np} \left[m_i(x)^{k-1} G(m_i(x)) \omega_i(x) \right] + S_k(x) \tag{3.43}$$

$$\frac{Dx}{Dt} = G(x) \tag{3.44}$$

$$\frac{D\mu_k(x)}{Dt} = G(x) \frac{\partial \mu_k(x)}{\partial x} + \frac{\partial \mu_k(x)}{\partial t} \tag{3.45}$$

onde, o termo fonte $S_k(m_i(x), \omega_i(x), t)$ representa os termos de quebra e agregação. Diferente dos outros métodos dos momentos, a Equação 3.40 é resolvida pelo método das características em uma malha Lagrangeana móvel, seguida da interpolação dos momentos cumulativos em uma malha fixa Euleriana. Já a etapa de reconstrução da função densidade numérica é feita pela solução da Equação 3.46.

$$\frac{\partial \mu_0(x)}{\partial x} = f(x) \tag{3.46}$$

Para obter a solução da Equação 3.46 um esquema de discretização de segunda ordem é utilizado. Neste caso, para evitar oscilações numéricas a discretização é feita pelo método MinMod [120].

O método CQMoM é capaz prever tanto os momentos da distribuição quanto a própria distribuição com grande precisão e acurácia. A acurácia da solução da PBE é controlada com o número de divisões no domínio de variáveis internas e número de momentos cumulativos utilizados. Contudo, o aumento do número de divisões e momentos pode aumentar o custo computacional, e portanto, tornar o método proibitivo para aplicações que envolvem CFD. Uma possível solução para amenizar o custo computacional associado ao número de divisões seria paralelizar a etapa de inversão dos momentos e a etapa de interpolação da malha Lagrangeana para a Euleriana. Interessante lembrar que no trabalho de ATTARAKIH *et al.* [118] não foi descrita a possibilidade de aplicar o CQMoM para problemas multivariados. No entanto, o desenvolvimento feito por ATTARAKIH *et al.* [118] pode ser extensível para problemas multivariados.

Um dos problemas sempre presente nos métodos dos momentos fechados por quadratura são os erros associados à aproximação do termo fonte. Várias tentativas de minimizar a propagação desse erro foram propostas, mas apenas LAGE [29] conseguiu propor um método baseado em momentos capaz de controlar os erros associados aos termos fontes.

LAGE [29] deduziu o método DuQMoGeM (*Dual-Quadrature Weighted-Residual Method of Generalized Moments*) pela aproximação funcional da função densidade numérica em termos de uma base ortogonal, $\phi_i(m)$.

$$f(m) = w(m)\sum_{i=0}^{\infty} c_i\phi_i(m) \quad \Rightarrow f(m) \approx w(m)\sum_{i=0}^{2n-1} c_i\phi_i(m). \tag{3.47}$$

Se o método DuQMoGeM for do tipo Galerkin, a mesma base $\phi_i(m)$ também é usada no cálculo dos momentos generalizados, e desta forma, as constantes c_i se relacionam com os momentos generalizados da FDN de acordo com a expressão (3.48). Portanto, calcular momentos generalizados também significa representar a FDN. Todavia, o foco desse método não é representar corretamente a FDN mas sim calcular os momentos generalizados do termo fonte com seus erros de quadratura controlados.

$$c_i = \frac{\mu_i^{\phi}}{\|\phi_i\|^2_{d\bar{\xi}}} \tag{3.48}$$

Após algumas manipulações, a equação de balanço populacional é transformada em equação de momentos generalizados (3.49) para o método DuQMoGeM Galerkin.

$$\frac{\partial \mu_k^{\phi}}{\partial t} + \sum_{i=0}^{2N-1} \sum_{k=0}^{2N-1} A_{jik} \frac{\mu_i^{\phi}}{\|\phi_i\|_{d\bar{\varsigma}}^2} \frac{\mu_k^{\phi}}{\|\phi_k\|_{d\bar{\varsigma}}^2} + \sum_{i=0}^{2N-1} (L_{ji} + G_{ji}) \frac{\mu_i^{\phi}}{\|\phi_i\|_{d\bar{\varsigma}}^2} = \int_{\Omega_m} \phi_j(m) R(m,t) dx \tag{3.49}$$

onde,

$$A_{jik} = \int_{\Omega_m} \int_{\Omega_m} \left[\frac{1}{2} \phi_j(m+m') - \phi_j(m) \right] a(m,m',t) \phi_i(m) \phi_k(m') w(m) w(m') dm dm' \tag{3.50}$$

$$L_{ji} = \int_{\Omega_m} b(m,t) \left[2\Pi_j^{\phi}(m,t) - \phi_j(m) \right] \phi_i(m) w(m) dm \tag{3.51}$$

$$G_{ji} = [G(m)\phi_i(m)\phi_j(m)w(m)]_0^{\infty} - \int_{\Omega_m} \frac{\partial \phi_i(m)}{\partial m} G(m) \phi_i(m) w(m) dm \tag{3.52}$$

Os termos A_{jik}, L_{ji} e G_{ji} são calculados por quadratura multidimensional adaptativa com uma tolerância especificada dentro do domínio Ω_m no espaço da variável interna m. A integração desses termos por cubatura adaptativa pode ser feita de forma paralela em arquitetura híbrida, portanto esse método é promissor, uma vez que controla os erros associados aos termos fontes e pode ser acelerado em arquiteturas híbridas. Ainda em 2011, FAVERO e LAGE [30] estenderam esse método para problemas bivariados mas utilizando cubaturas adaptativas para obter os termos **A**, **L** e **G**. Posteriormente, como ordem natural de desenvolvimento, SANTOS *et al.* [31], como parte do presente trabalho, desenvolveram o método Direct DuQMoGeM. Os autores transformaram o método DuQMoGeM, que transporta momentos generalizados, em sua forma direta evoluindo os pesos e abscissas da quadratura de Gauss-Christoffel. Mais detalhes sobre o DuQMoGeM e o Direct DuQMoGeM serão descritos no capítulo de desenvolvimento.

YUAN *et al.* [121] propuseram recentemente o *Extended* QMoM (EQMoM), cujo objetivo é eliminar a deficiência do QMoM de não ter uma representação funcional para a FDN. Os autores focaram em resolver o problema de aproximação da FDN, propondo o uso de KDFs (*kernel density functions*), cuja a vantagem é garantir a positividade de FDN. Tal qual o DuQMoGeM, esse método também utiliza uma quadratura dupla. Portanto além de uma aproximação realizável da FDN, o mesmo pode controlar os erros de quadratura presentes no QMoM.

3.8 Considerações Finais

Dentre os métodos passíveis de paralelismo em GPUs podemos citar o método de Monte Carlo e os métodos híbridos de quadratura dupla. No caso do método de Monte Carlo, cada realização é independente, o que confere ao método sua capacidade de ser paralelizável [61]. A proposta paralela para esse método baseia-se em calcular cada realização por um *thread* da GPU. No entanto, como o método de Monte Carlo pode não ser acurado com um número pequeno de realizações, sua utilização pode ser proibitiva ainda que a computação paralela seja usada nas aplicações CFD. Como segunda opção, os métodos dos momentos baseados em quadratura dupla são ótimos candidatos devido ao seu controle de acurácia por cubatura adaptativa ser paralelizável. Esse paralelismo pode viabilizar seu uso em aplicações que envolvem o acoplamento PB-CFD com baixa dimensionalidade na PBE. Além disso, os métodos de quadratura dupla são flexíveis, uma vez que o controle de acurácia, seu limitante em termos de tempo computacional, é opcional. Assim, o presente trabalho está focado em acelerar os métodos DuQMoGeM e Direct DuQMoGeM.

Capítulo 4

Modelagem de Escoamentos Dispersos

Existem duas formas principais de modelar escoamentos multifásicos polidispersos, uma em sua forma Euleriana-Euleriana e outra na forma Euleriana-Lagrangeana. Em ambos os casos, a fase contínua é descrita por equações de transporte Eulerianas. No entanto, ao contrário da fase contínua, a dispersa pode ser resolvida tanto na sua forma Lagrangeana quanto Euleriana.

A essência da abordagem Euleriana-Lagrangeana (E-L) reside em acoplar a solução Euleriana da fase contínua com o tratamento Lagrangeano da fase dispersa, onde cada partícula (ou conjunto de partículas) são interpretadas como pontuais [10]. Sendo assim, nessa formulação a trajetória das partículas são determinadas pelas equações 4.1 e 4.2.

$$m_i \frac{d\mathbf{u}_i}{dt} = \sum \mathbf{Fp}_i \tag{4.1}$$

$$\frac{d\mathbf{z}_i}{dt} = \mathbf{u}_i \tag{4.2}$$

onde, $\mathbf{Fp}_i$ representa as forças que agem sobre a partícula i (forças de arraste massa virtual, dispersão turbulenta e etc), $\mathbf{z}_i$ é a posição, m_i é a massa e $\mathbf{u}_i$ é a velocidade da partícula i. O procedimento de cálculo se inicia com a solução dos campos da fase contínua, seguido pela evolução temporal e espacial das equações diferenciais ordinárias 4.1 e 4.2. Finalmente, os efeitos introduzidos pelos campos da fase dispersa são transferidos para fase contínua. Esse processo se repete de forma iterativa até a convergência. Infelizmente, nos casos em que o número de partículas é suficientemente grande, o custo computacional aumenta drasticamente, e portanto, essa abordagem é normalmente utilizada para escoamentos diluídos [11].

Na abordagem Euleriana-Euleriana (E-E), as diferentes fases são matematicamente tratadas como interpenetrantes, o que implica na coexistência simultânea das fases na mesma posição do espaço e no mesmo instante de tempo. Note que isso só

é possível porque as equações de transporte são equações promediadas, originadas pelo processo de média volumétrica ou amostral, cuja fração de fase (r_α) equivale a probabilidade de uma fase existir em um dado ponto no espaço e no tempo [122].

Usualmente, são utilizados três tipos de metodologias E-E: a modelagem *Volume of fluid* (VOF), o modelo de mistura e a modelagem multi-fluido. Na modelagem VOF, a interface que define as fases é rastreada por uma função topológica que é resolvida em conjunto com equações conservação para uma mistura. No caso do modelo de mistura, as equações de conservação para uma mistura são resolvidas concomitantemente com modelos algébricos de deslizamento entre a fase dispersa e a contínua. Já na modelagem multi-fluido, cada fase tem suas próprias equações de conservação de quantidade de movimento e de massa [10]. Como usaremos essa última abordagem, ela será descrita em mais detalhes nas seções subsequentes.

4.1 Modelagem Multifásica Euleriana-Euleriana

Como mencionado anteriormente, as equações de transporte multifásicas E-E são deduzidas a partir do processo de média amostral ou volumétrica [122]. Nesta derivação, a função indicadora de fase (χ_α), cuja média é a fração da fase α, é fundamental. Basicamente, χ_α identifica a presença ou não de uma determinada fase α, de acordo com a equação abaixo,

$$\chi_\alpha(\mathbf{x}, t; k) = \begin{cases} 1 & \text{se existe a fase } \alpha \text{ no ponto } \mathbf{x} \text{ e no tempo } t \text{ da realização } k \\ 0 & \text{se não existe a fase } \alpha \text{ no ponto } \mathbf{x} \text{ e no tempo } t \text{ da realização } k. \end{cases}$$

Multiplicando a função indicadora de fase nas equações de conservação, aplicando as operações de média, utilizando as regras de Reynolds, Leibnitz e Gauss, seguido de algumas manipulações, obtém-se as equações de conservação E-E de massa e quantidade movimento abaixo, respectivamente [122].

$$\frac{\partial(r_\alpha \rho_\alpha)}{\partial t} + \nabla.(r_\alpha \rho_\alpha \mathbf{u}_\alpha) = \Gamma_\alpha, \tag{4.3}$$

$$\frac{\partial(r_\alpha \rho_\alpha \mathbf{u}_\alpha)}{\partial t} + \nabla.(r_\alpha \rho_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha) = -r_\alpha \nabla p_\alpha + \nabla.(r_\alpha \rho_\alpha \boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{eff}}) + r_\alpha \rho_\alpha \mathbf{g} + \Gamma_\alpha \mathbf{u}_{\alpha,\mathbf{I}} + \sum_{\substack{\beta=0 \\ \beta \neq \alpha}}^{\beta=N} \mathbf{M}_{\alpha,\beta}, \tag{4.4}$$

onde ρ_α é a massa específica, Γ_α é o termo fonte mássico, $\mathbf{u}_\alpha$ é a velocidade da fase α e $\mathbf{u}_{\alpha,\mathbf{I}}$ é a velocidade da fase α na interface. $\boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{eff}}$ é o tensor tensão efetivo (equação 4.5) que geralmente é decomposto em efeitos viscosos ($\boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{lam}}$) e turbulentos ($\boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{turb}}$). Nos casos onde o escoamento é turbulento, a tensão turbulenta pode ser aproximada por um modelo linear usando uma viscosidade turbulenta (μ^{turb}), que geralmente é

calculada por modelos de fechamento tipo RANS (*Reynolds-averaged Navier–Stokes equations*) ou LES (*Large Eddy Simulation*) [123].

$$\boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{eff}} = (\mu^{lam} + \mu^{turb}) \left[\nabla \mathbf{u}_\alpha - \frac{2}{3}(\nabla \cdot \mathbf{u}_\alpha)\mathbf{I} \right] + \frac{2}{3}\kappa_\alpha \mathbf{I} \tag{4.5}$$

Na Equação 4.4, o termo fonte $\mathbf{M}_{\alpha,\beta}$ representa a transferência de quantidade de movimento na interface entre as fases α e β. Essas forças são geralmente desmembradas em forças de arraste, sustentação, massa virtual, pressão interfacial e tensão de cisalhamento na interface, respectivamente representadas na Equação 4.6.

$$\mathbf{M}_{\alpha,\beta} = \mathbf{M}_{\alpha,\beta}^{\mathbf{D}} + \mathbf{M}_{\alpha,\beta}^{\mathbf{L}} + \mathbf{M}_{\alpha,\beta}^{\mathbf{VM}} + p_{\alpha,\beta}\nabla r_\alpha - \boldsymbol{\tau}_{\boldsymbol{\alpha},\boldsymbol{\beta}} \nabla r_\alpha. \tag{4.6}$$

Apesar de relativamente geral, algumas simplificações são muito comuns nessa abordagem. No presente trabalho, por exemplo, todas as fases dividem o mesmo campo de pressão ($p_\alpha = p$) e as fases dispersas não transferem quantidade de movimento entre si. Portanto, as simulações executadas neste trabalho só consideram as forças de arraste, sustentação e massa virtual, entre cada fase dispersa e a fase contínua, que são dadas, respectivamente, por:

$$\mathbf{M}_{\alpha,\mathbf{0}}^{\mathbf{D}} = \frac{1}{2} r_\alpha A_\alpha \rho_0 C_{D,\alpha} |\mathbf{u}_{\mathbf{r},\alpha}| \mathbf{u}_{\mathbf{r},\alpha}, \tag{4.7}$$

$$\mathbf{M}_{\alpha,\mathbf{0}}^{\mathbf{L}} = r_\alpha \rho_0 C_{L,\alpha} \mathbf{u}_{\mathbf{r},\alpha} \times (\nabla \times \mathbf{u_0}), \tag{4.8}$$

e

$$\mathbf{M}_{\alpha,\mathbf{0}}^{\mathbf{VM}} = r_\alpha \rho_0 C_{VM,\alpha} \left[\frac{D\mathbf{u_0}}{Dt} - \frac{D\mathbf{u}_\alpha}{Dt} \right], \tag{4.9}$$

onde ρ_0 é a massa específica da fase contínua, $\mathbf{u}_{\mathbf{r},\alpha}$ é a velocidade relativa entre as fases contínua e dispersa α ($\mathbf{u}_{\mathbf{r},\alpha} = \mathbf{u_0} - \mathbf{u}_\alpha$). Para o cálculo das forças de massa virtual e sustentação, os coeficientes $C_{L,\alpha}$ e $C_{VM,\alpha}$ são aqui iguais a 0,5. Para a força de arraste, o coeficiente $C_{D,\alpha}$ é calculado pela correlação de SCHILLER e NAUMANN [124]. Por fim, A_α é a razão entre a área projetada da partícula e seu volume. Admitindo partículas esféricas, temos que A_α pode se calculado por,

$$A_\alpha = \frac{\pi d_\alpha^2/4}{\pi d_\alpha^3/6} = \frac{3}{2d_\alpha}. \tag{4.10}$$

Cabe ressaltar que os leitores podem encontrar uma dedução rigorosa das equações E-E multifásicas em YEOH e TU [10], DREW e PASSMAN [122], ISHII e HIBIKI [125]. Além disso, os igualmente interessados sobre a modelagem da turbulência podem encontrar mais detalhes em POPE [123].

4.2 Acoplamento PB-CFD multifásico

Na maioria dos casos, as características dos sistemas particulados variam de ponto para ponto dentro do domínio, isto é, são não homogêneas. Isso implica que o campo de velocidade das partículas, bem como os termos fonte (relacionados aos fenômenos de quebra e agregação), dependem da fluidodinâmica da fase contínua, da mesma forma que a velocidade da fase contínua depende do comportamento da fase dispersa.

Esse efeito pode ser capturado combinando a abordagem multifásica E-E com a equação de balanço populacional. A primeira tentativa de acoplar a PBE com CFD foi a proposta por LO [113]. Ele criou o método MUSIG (*Multiple Size Group*), onde uma das fases representa a fase contínua e a outra é representada por todas as classes[5] da fase dispersa. Essa metodologia assume que as velocidades de todas as partículas são iguais, o que não ocorre na maioria dos sistemas polidispersos.

SILVA *et al.* [126] realizaram testes comparativos entre o acoplamento PB-CFD numa formulação bifásica pelo método MUSIG utilizando dois *softwares*: ANSYS CFX (*software* comercial) e OpenFOAM® (código aberto), confirmando a eficiência do método DQMoM em realizar simulações acopladas. SILVA e LAGE [3] estenderam sua implementação para uma modelagem E-E multi-fluido utilizando o método DQMoM no OpenFOAM® . Neste trabalho, apresentaram o procedimento de discretização e acoplamento pressão velocidade para o escoamento multifásico multi-fluido. Obtiveram resultados satisfatórios, no entanto, ainda havia problemas associados à solução da equação de balanço populacional. Problemas relacionados com o acúmulo de erros de quadratura e com a não preservação dos momentos.

BUFFO *et al.* [9] compararam os métodos *Conditional* QMoM e o DQMoM frente ao método de monte Carlo DSCM. Depois da verificação dos métodos com casos testes homogêneos, os mesmos foram implementados usando a abordagem E-E multi-fluido no pacote CFD ANSYS FLUENT. Nesse trabalho, também propuseram uma metodologia que supera o problema de conservação dos momentos nos métodos diretos. Esse problema ocorre porquê as variáveis que são convectadas não são as variáveis conservadas (momentos), mas sim variáveis secundárias (pesos e abscissas). Nessa abordagem denominada DQMoM *Fully Conservative*, ou simplesmente DQMoM-FC, cada célula da malha CFD é tratada como um sistema homogêneo aberto, e os fluxos de momentos nas faces são calculados explicitamente, usando esquemas de discretização realizáveis [127]. Apesar de nenhum formalismo matemático que prove sua eficiência, essa pequena modificação propiciou a preservação dos momentos para o DQMoM. Note que essa metodologia pode ser aplicada em qualquer método direto que conserve momentos. Os resultados obtidos por BUFFO

[5]Metodologia implementada no software comercial ANSYS CFX.

et al. [9] corroboraram a equivalência entre os métodos CQMoM e DQMoM, exceto nos casos que o DQMoM não estava na sua forma conservativa (DQMoM-FC). BUFFO *et al.* [9] concluíram que a forma explícita no cálculo de momentos são necessárias para preservação dos momentos nos métodos diretos como o DQMoM.

Como já mencionado, o FCMoM é um método que permite calcular os momentos e ao mesmo tempo obter a FDN. No entanto, não é genérico o suficiente para qualquer aplicação, devido a sua especifidade na movimentação do contorno. ABBASI e ARASTOOPOUR [128] retestaram o FCMoM e avaliaram sua eficiência. Neste trabalho, corroboraram a sua aplicação para problemas homogêneos. Além disso, ABBASI e ARASTOOPOUR [128] ampliaram a sua aplicação para problema não homogêneos, implementando o método no código comercial ANSYS FLUENT. Analisaram sua implementação com o mesmo caso teste não realístico proposto por SILVA e LAGE [3] e compararam seus resultados com o método QMoM. Notaram que, para um número de pontos de quadratura menor que 3, os métodos são equivalentes em termos de acurácia. No entanto, para $N > 3$, o método FCMoM pode produzir momentos não realizáveis mesmo para funções de interpolação de primeira ordem, que a princípio garante a realizabilidade DQMoM de acordo com VIKAS *et al.* [127].

Em suma, os métodos de solução e acoplamento PB-CFD ainda não estão consolidados e estudo de desenvolvimento de novos métodos são necessários. Vários autores propuseram técnicas de acoplamento com diferentes métodos de solução da PBE. Todavia, ainda existem muitas dúvidas envolvidas, não só apenas nos métodos de solução, mas também na modelagem do escoamento multifásico polidisperso.

4.2.1 Solução do modelo Multi-fluido

Nesta seção, o modelo multi-fluido na sua forma semi-discretizada é apresentado, de acordo com a implementação de SILVA e LAGE [3]. Basicamente, para gerar a equação de conservação de quantidade de movimento em sua forma intensiva incompressível ("*phase intensive*") [129], divide-se a Equação 4.4 pela massa específica (ρ_α) e pela fração de fase (r_α), desconsiderando o termo $\mathbf{u}_{\alpha,I}$. Após as devidas manipulações usando a Equação 4.5, obtém-se a Equação 4.11. O leitor pode encontrar os passos detalhados para dedução da equação abaixo em SILVA e LAGE [3].

$$\begin{aligned}\frac{\partial \mathbf{u}_\alpha}{\partial t} + \nabla\cdot(\mathbf{u}_\alpha\mathbf{u}_\alpha) - \mathbf{u}_\alpha(\nabla\cdot\mathbf{u}_\alpha) - \nabla\cdot\left(\upsilon_\alpha^{eff}\frac{\nabla r_\alpha}{r_\alpha}\mathbf{u}_\alpha\right) + \mathbf{u}_\alpha\nabla\cdot\left(\upsilon_\alpha^{eff}\frac{\nabla r_\alpha}{r_\alpha}\right) \\ -\upsilon_\alpha^{eff}\nabla^2\mathbf{u}_\alpha + \nabla\cdot\left(\frac{\boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{eff}}}{\rho_\alpha}\right) + \frac{\nabla r_\alpha}{r_\alpha}\cdot\left(\frac{\boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{eff}}}{\rho_\alpha}\right) = -\frac{1}{\rho_\alpha}\nabla p + \frac{\mathbf{M}_\alpha}{\rho_\alpha r_\alpha} + \mathbf{g}.\end{aligned} \tag{4.11}$$

Reescrevendo a Equação 4.11 em sua forma semi-discretizada, temos que:

$$\boldsymbol{\Upsilon}_\alpha = -\frac{\nabla p}{\rho_\alpha} + \frac{\mathbf{M}_\alpha}{\rho_\alpha r_\alpha} + \mathbf{g} \quad \alpha = 0 \cdots N, \tag{4.12}$$

sendo que, M_α contém das forças de arraste, sustentação e massa virtual, respectivamente, citadas abaixo:

$$\mathbf{M}_\alpha = r_\alpha K_\alpha^D \mathbf{u}_{\mathbf{r},\alpha} + r_\alpha K_\alpha^L \mathbf{u}_{\mathbf{r},\alpha} \times (\nabla \times \mathbf{u_0}) + r_\alpha K_\alpha^{VM} \left[\frac{D\mathbf{u_0}}{Dt} - \frac{D\mathbf{u}_\alpha}{Dt} \right], \tag{4.13}$$

em que os multiplicadores associados as forças interfaciais de arraste, sustentação e massa virtual são, respectivamente:

$$K_\alpha^D = \frac{1}{2} A_\alpha \rho_0 C_{D,\alpha} |\mathbf{u}_{\mathbf{r},\alpha}|, \quad K_\alpha^L = \rho_0 C_{L,\alpha} \quad e \quad K_\alpha^{VM} = \rho_0 C_{VM,\alpha} \tag{4.14}$$

Como supracitado, não há nenhuma troca de quantidade de movimento entre as fases dispersas. Isso pode ser claramente observado na Equação 4.13, onde apenas a fase contínua afeta fase dispersa α. Já no caso da fase contínua, todas as fases dispersas podem transferir quantidade de momento para a mesma, de acordo com o Equação 4.15.

$$\mathbf{M_0} = -\sum_{i=1}^{N} r_\alpha K_\alpha^D \mathbf{u}_{\mathbf{r},\alpha} - \sum_{i=1}^{N} r_\alpha K_\alpha^L \mathbf{u}_{\mathbf{r},\alpha} \times (\nabla \times \mathbf{u_0}) + \sum_{i=1}^{N} r_\alpha K_\alpha^{VM} \left[\frac{D\mathbf{u}_\alpha}{Dt} - \frac{D\mathbf{u_0}}{Dt} \right] \tag{4.15}$$

Voltando a Equação 4.12, o termo $\boldsymbol{\Upsilon}_\alpha$ se refere a discretização do lado esquerdo da Equação 4.11. No entanto, de acordo com a implementação de RUSCHE [129], o termo fonte de troca de quantidade de movimento entre as fases é também discretizado de forma semi-implícita. Neste sentido, a velocidade na força de arraste e a derivada substantiva na força de massa virtual são discretizadas de forma implícita para a fase considerada. Todavia, na força de sustentação, a velocidade e suas derivadas são discretizadas explicitamente. Esses termos discretizados podem ser incluídos na parte discretizada da Equação 4.12, levando assim às seguintes equações para as fases contínua e dispersas:

$$\boldsymbol{\Upsilon}_0^C = -\frac{\nabla p}{\rho_0} + \mathbf{g} + \frac{1}{\rho_0 r_0} \sum_{\alpha=1}^{N} r_\alpha K_\alpha^D \mathbf{u}_\alpha \tag{4.16}$$

e

$$\boldsymbol{\Upsilon}_\alpha^C = -\frac{\nabla p}{\rho_\alpha} + \mathbf{g} + \frac{K_\alpha^D}{\rho_\alpha} \mathbf{u_0}. \tag{4.17}$$

A discretização do termo $\boldsymbol{\Upsilon}_\alpha^C$ (ver Anexo A para mais detalhes) gera a matriz $\mathbf{C}_\alpha$. Essa matriz é decomposta em termos de uma matriz diagonal "$(\mathbf{C_0})_D$" e

não diagonal "$(\mathbf{C_0})_N$". Definindo o operador linear "$(\mathbf{C_0})_H$", como sendo $(\mathbf{C_0})_H = (\mathbf{C_0})_S - (\mathbf{C_0})_N\Phi$, onde o índice "$S$" denota o termo fonte associado ao sistema linear gerado pela discretização da Equação 4.12 e Φ é a solução aproximada desse sistema linear. Após o rearranjo das equações 4.16 e 4.17, obtém-se as equações abaixo para as **velocidades das fases contínua e dispersas**:

$$\mathbf{u_0} = \frac{(\mathbf{C_0})_H}{(\mathbf{C_0})_\mathbf{D}} - \frac{\nabla p}{\rho_0(\mathbf{C_0})_D} + \frac{\mathbf{g}}{(\mathbf{C_0})_D} + \frac{1}{\rho_0 r_0(\mathbf{C_0})_D}\sum_{i=\alpha}^{N} K_\alpha^D \mathbf{u}_\alpha \tag{4.18}$$

$$\mathbf{u}_\alpha = \frac{(\mathbf{C}_\alpha)_H}{(\mathbf{C}_\alpha)_D} - \frac{\nabla p}{\rho_\alpha(\mathbf{C}_\alpha)_D} + \frac{\mathbf{g}}{(\mathbf{C}_\alpha)_D} + \frac{K_\alpha^D}{r_\alpha(\mathbf{C}_\alpha)_D}\mathbf{u_0}. \tag{4.19}$$

Supondo que todas as fases dispersas têm a mesma massa específica, ρ_α, usando as identidades $\sum_{\alpha=0}^{N} r_\alpha = 1$ e $\sum_{\alpha=0}^{N} \Gamma_\alpha = 0$, obtém-se a Equação 4.22, manipulando as Equações 4.20 e 4.21,

$$\frac{\partial r_0}{\partial t} + \nabla \cdot (r_0 \mathbf{u_0}) = 0 \tag{4.20}$$

$$\frac{\partial r_\alpha}{\partial t} + \nabla \cdot (r_\alpha \mathbf{u}_\alpha) = \frac{\Gamma_\alpha}{\rho_\alpha}, \quad \alpha = 1 \cdots N \tag{4.21}$$

$$\nabla_D \cdot \left(\sum_{\alpha=0}^{N} r_{\alpha,f}\psi_\alpha \right) = 0 \tag{4.22}$$

onde ∇_D é a forma discretizada do operador divergente, f se refere ao valor interpolado sobre a face do volume de controle e $\psi_\alpha = \mathbf{S}_f \cdot (\mathbf{u}_\alpha)_f$, em que $\mathbf{S}_f$ é o vetor de área da face. Usando as Equações 4.19 e 4.18, o fluxo volumétrico, ψ_α, pode ser escrito de acordo com a equação abaixo.

$$\psi_\alpha = \psi_\alpha^* - \left(\frac{1}{\rho_\alpha(\mathbf{C}_\alpha)_H)} \right)_f \mathbf{S} \cdot (\nabla p)_f, \quad \alpha = 0 \cdots N \tag{4.23}$$

onde,

$$\psi_\alpha^* = \left(\frac{(\mathbf{C}_\alpha)_H}{(\mathbf{C}_\alpha)_D} \right) \cdot \mathbf{S} + \left(\frac{1}{(\mathbf{C}_\alpha)_D} \right) \mathbf{g} \cdot \mathbf{S} + \left(\frac{K_\alpha^D}{r_\alpha(\mathbf{C}_\alpha)_D}\psi_\mathbf{0} \right)_f, \tag{4.24}$$

$$\psi_0^* = \left(\frac{(\mathbf{C_0})_H}{(\mathbf{C}_\alpha)_D} \right) \cdot \mathbf{S} + \left(\frac{1}{(\mathbf{C}_\alpha)_D} \right) \mathbf{g} \cdot \mathbf{S} + \left(\frac{1}{\rho_0 r_0(\mathbf{C_0})_D} \sum_{\alpha=1}^{N} K_\alpha^D \psi_\alpha \right)_f, \tag{4.25}$$

A **equação da pressão** resultante das equações 4.23 e 4.22 pode ser escrita como:

$$\nabla_D \cdot (D_p(\nabla p)_f) = \nabla_D \cdot \left(r_0\psi_0^* + \sum_{\alpha=0}^{N} r_\alpha \psi_\alpha^* \right)_f \tag{4.26}$$

onde,

$$D_p = r_{0,f} \left(\frac{1}{\rho_0(\mathbf{C}_\alpha)_D} \right)_f + \sum_{\alpha=1}^{N} r_{\alpha,f} \left(\frac{1}{\rho_\alpha(\mathbf{C}_\alpha)_D} \right)_f \tag{4.27}$$

Para uma modelagem multi-fluido, a velocidade média do sistema disperso é definido por

$$\overline{\mathbf{u}} = \sum_{\alpha=0}^{N} r_\alpha \mathbf{u}_\alpha \tag{4.28}$$

ou,

$$\mathbf{u}_\alpha = \overline{\mathbf{u}} + r_0 \mathbf{u}_{r,0} + \sum_{\substack{i=0 \\ i\neq\alpha}}^{i=N} r_i \mathbf{u}_{r,i} \tag{4.29}$$

Substituindo a Equação 4.29 na Equação 4.21, a Equação para fase dispersa passa a ser:

$$\frac{\partial r_\alpha}{\partial t} + \nabla\cdot(\overline{\mathbf{u}} r_\alpha) + \nabla\cdot(r_0 \mathbf{u}_{r,0} r_\alpha) + \nabla\cdot\left(\sum_{\substack{i=1 \\ i\neq\alpha}}^{i=N} r_i \mathbf{u}_{r,i} r_\alpha\right) = \frac{\Gamma_\alpha}{\rho_\alpha} \tag{4.30}$$

Para o acoplamento PB-CFD, cada ponto de quadratura de Gauss-Christoffel é atribuído a uma fase dispersa α e, quando suas abscissas são o volume de partículas, as abscissas ponderadas ζ_α são equivalentes a fração de fase em um escoamento incompressível. Assim, a **equação para fração de fase** se torna:

$$\frac{\partial \varsigma_\alpha}{\partial t} + \nabla\cdot(\overline{\mathbf{u}}\varsigma_\alpha) + \nabla\cdot(\mathbf{u}_{r,0}\varsigma_\alpha) - \nabla\cdot\left(\sum_{i=1}^{i=N} \varsigma_i \mathbf{u}_{r,0}\varsigma_\alpha\right) + \nabla\cdot\left(\sum_{\substack{i=1 \\ i\neq\alpha}}^{i=N} \varsigma_i \mathbf{u}_{r,i}\varsigma_\alpha\right) = \frac{\Gamma_\alpha}{\rho_\alpha} \tag{4.31}$$

Além disso, se as partículas forem consideradas esféricas, o volume de partícula pode ser convertido em seu diâmetro, e portanto, usado no cômputo das forças de interação entre as fases dispersa e contínua. Note que o termo $\frac{\Gamma_\alpha}{\rho_\alpha}$ corresponde ao ganho ou perda de volume das partículas da fase dispersa α, incluindo os efeitos de quebra e agregação. Observe também que o momento de primeira ordem da FDN representa a fração volumétrica da fase dispersa. Portanto, a Equação 4.31 está associada com a equação do momento de primeira ordem gerado pela equação de balanço populacional. Fundamentalmente, com o conjunto de equações descritos nesta seção é possível caracterizar escoamentos multifásicos polidispersos com fases incompressíveis considerando os efeitos de interação entre as partículas.

4.2.2 Pacote CFD OpenFOAM®

O pacote OpenFOAM® enquanto ferramenta para o desenvolvimento de novas tecnologias é, de fato, interessante por inúmeras razões. Nesse sentido, vale discutir sobre sua filosofia e as razões pelos quais o mesmo vem sendo utilizado como ferramenta para pesquisa, aplicação e desenvolvimento.

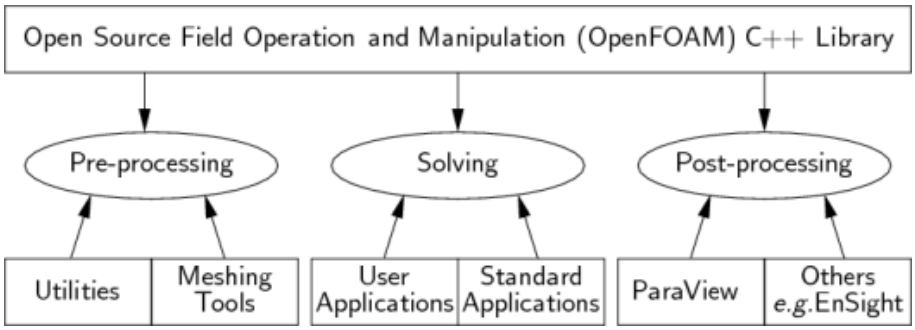


Figura 4.1: Estrutura global do OpenFOAM®(OPENFOAM [1]).

Os pacotes CFD em geral, tanto comerciais quanto livres, são divididos em três partes principais: O pré-processamento, o *solver* e o pós-processamento. No pré-processamento, a malha computacional, as condições de contorno e as propriedades físicas e químicas são especificadas. Na etapa do *solver*, as equações de transporte são discretizadas e resolvidas. Finalmente, o pós-processamento onde os resultados simulados são tratados e visualizados com um visualizador adequado. Nesse aspecto, o OpenFOAM® não é diferente, segue as formas tradicionais de uma simulador CFD.

Basicamente, o OpenFOAM® é um conjunto de bibliotecas escritas em C++ para desenvolvimento e customização de códigos que visam solucionar problemas de mecânica do contínuo por volumes finitos com variáveis colocalizadas. Seu desenvolvimento original se iniciou na década de 80 no *Imperial College*, Londres, com intuito de criar uma plataforma geral de simulação. A escolha do C++ ocorreu devido a sua modulabilidade e abstração orientada a objeto, tais como: encapsulamento de dados, herança e polimorfismo. Essas abstrações permitem a expansão do *software* de forma relativamente simplificada.

Embora seu antecessor, denominado FOAM, tenha sido vendido inicialmente pela empresa inglesa Nabla Ltd, o OpenFOAM®, como o nome já diz, foi liberado como código aberto sobre a licença GPL (GNU *General Public License*), em dezembro de 2004. Como consequência, a partir de então, o OpenFOAM® passou a ser visto com muito interesse pela comunidade científica. Contudo hoje, felizmente, novas tendências vem sendo observadas quando ao seu uso, uma vez que grandes empresas, tais como Honda, Shell, PETROBRAS e Airbus, vem utilizando e reconhecendo o potencial do OpenFOAM®.

Como citado anteriormente, a linguagem orientada a objeto permite maior legibilidade, o que viabiliza e flexibiliza ao usuário implementar seus próprios códigos, a fim de adaptá-los ao seu caso específico.

Como exemplo ilustrativo, a equação de transporte:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\Psi \mathbf{u}) - \nabla \cdot (\mu \nabla \mathbf{u}) - \nabla p, \tag{4.32}$$

é interpretada pelo seguinte trecho de código em C++ no OpenFOAM®

```
solve
(
   fvm::ddt(rho,U)
+ fvm::div(phi,U)
- fvm::laplacian(mu,U)
==
- fvc::grad(p)
);
```

Código 4.1: código ilustrativo do OpenFOAM

onde, os operadores de campos vetoriais associados a Equação 4.32 são relacionados com linguagem C++ de acordo com a Tabela 4.1. Ainda nesse trecho de código, a variável **phi**, nada mais é que a representação do fluxo convectivo Ψ definido pela Equação 4.33.

$$\Psi = \oint \mathbf{U} \cdot \mathbf{n} dA \tag{4.33}$$

Os termos **fvm** e **fvc** são as formas de discretização por volumes finitos implícitas e explícitas, respectivamente, utilizadas na montagem do sistema linear a ser resolvido. E, finalmente, a instrução **solve** indica que o sistema linear gerado pela discretização da Equação 4.32 será resolvido por algum algoritmo disponível ou acoplado ao OpenFOAM®.

Tabela 4.1: Operadores de campos vetoriais disponíveis no OpenFOAM® .

Operadores vetoriais	OpenFOAM®
$\frac{\partial}{\partial t}()$	fvm::ddt()
$\nabla \cdot ()$	fvm::div()
$\nabla^2()$	fvm::laplacian()
$\nabla()$	fvc::grad()

Notem que o uso do conceito de *classes* e *templates* permite que a programação seja semelhante à notação matemática. Isso se estende para os conceitos físicos, unidades, métodos de discretização e etc. Por exemplo, um campo escalar, como temperatura, pode ter os seus valores armazenados nos centros de um volume de controle da malha, o que define o **volScalarField**. Esse conceito é aplicado para outros tipos de variável, como por exemplo, a velocidade que é declarada como um **volVectorField**, em outras palavras, um campo de vetores armazenados nos centros dos volumes.

Importante destacar também que além de uma certa facilidade de implementação, o OpenFOAM® permite o uso de tecnologias avançadas de simulação CFD, tais como: computação em paralelo, malhas poliédricas não-estruturadas, o uso de

esquemas de interpolação de alta resolução e um conjunto de avançados *solvers* para resolução de sistemas lineares.

Em resumo, o OpenFOAM® tem várias vantagens: uma sintaxe amigável para solução de equações diferenciais parciais, permite o uso de malhas não estruturadas, tem ferramentas de paralelização, contém um conjunto amplo de modelos já disponíveis e não existe custo com licença. Obviamente, existem algumas desvantagens inerentes a maioria dos códigos livres. A principal delas está associada à longa curva de aprendizado devido à falta de uma documentação detalhada. Apesar de resumida, para obter maiores informações sobre a ferramenta OpenFOAM®, o autor recomenda como leitura inicial a sua documentação básica, disponível no *User's Guide* e no *Programmer's Guide*.

Capítulo 5

Computação Paralela

A solução de problemas de engenharia muitas vezes exige um esforço computacional que os supercomputadores não são capazes de realizar em tempo hábil de forma sequencial. Neste sentido, os supercomputadores paralelos tem sido a solução para problemas de custo elevado.

As técnicas de paralelismo vem sendo empregadas e desenvolvidas há anos. Contudo, as soluções que envolvem paralelismo têm sido limitadas por restrições físicas e consumo de energia, que são, hoje, os gargalos ao aumento da velocidade de computação. Concomitantemente, a indústria de jogos necessitava de processamento de imagem em tempo real e de alta definição, que por sua vez, se tornou a força motriz para a evolução das placas gráficas. Essa evolução levou a características massivamente paralelas que, devido a isso, ganharam espaço na computação científica. Ademais, essa tecnologia possui menor custo relativo ao consumo de energia e o espaço físico necessário, que ratificaram sua vantagem frente as CPUs [34].

Neste capítulo são abordados alguns conceitos básicos sobre computação paralela. Conceitos como classificação da computação, métricas de desempenho e modelos de programação em paralelo.

5.1 Classificação da Computação Paralela

Embora não haja um consenso sobre a taxonomia definida por Flynn [130], ela é amplamente utilizada na classificação dos processos paralelos. Nesta classificação, as formas de computação são organizadas de acordo com a quantidade de operações realizadas e o fluxo de dados processados na arquitetura paralela. As formas de classificação são: *Multiple Instruction Single Data* (MISD), *Single Instruction Single Data* (SISD), *Single Instruction Multiple Data* (SIMD) e *Multiple Instruction Multiple Data* (MIMD).

As máquinas SISD são as arquiteturas convencionais, onde as operações são tratadas de forma sequencial. Já nas máquinas MISD, um número p de processadores

executam instruções diferentes sobre o mesmo conjuntos de dados. Por outro lado, as máquinas paralelas classificadas como SIMD executam o mesmo fluxo de instrução com diferentes conjuntos de dados. Por fim, as arquiteturas MIMD operam múltiplas instruções em vários conjuntos de dados diferentes.

Geralmente, as máquinas tipo SIMD são utilizadas na operação de vetores e matrizes de grande porte, por isso, muitas vezes são chamadas de máquinas vetoriais e matriciais [131]. Como as máquinas paralelas MIMD eram mais eficientes e baratas, essa tecnologia foi deixada de lado por um longo tempo [34]. Todavia, recentemente com o desenvolvimento das placas gráficas programáveis, o interesse pela utilização desse tipo de arquitetura vem crescendo devido ao seu baixo custo e alto desempenho [132].

As máquinas paralelas também podem ser classificadas de acordo com a distribuição de memória. Essa classificação se divide em máquinas que utilizam memória compartilhada, distribuída e híbrida. Na arquitetura com memória compartilhada, cada processador tem acesso a mesma memória. Se a memória for acessada com a mesma velocidade, as máquinas com memória compartilhada são denominadas máquinas UMA (*Uniform Memory Access*), caso contrário são chamadas de NUMA (*Non Uniform Memory Access*). A principal vantagem dessa arquitetura está na proximidade entre a memória compartilhada e as CPUs, o que garante maior velocidade no acesso à memória. Contudo, a escalabilidade é limitada por questões físicas e pelo fato do acesso à memória ser prejudicado pelo tráfego de dados em uma única memória compartilhada [133].

As máquinas paralelas com memória distribuída são formadas por interconexões entre os processadores através de uma rede, sendo que cada processador ou conjunto de processadores tem sua memória local. Neste tipo de arquitetura cada processador tem acesso livre a sua memória local, no entanto, para acessar a memória do nó vizinho é necessário acesso via rede. A principal vantagem dessa arquitetura está na facilidade de escalabilidade, contudo a perda de eficiência ocorre devido ao elevado tempo despendido na comunicação entre os nós [133].

Nesse sentido, qualquer técnica de paralelismo está sujeita a perdas de eficiência devido aos limites algorítmicos e arquiteturais. Portanto, é natural que haja uma medida da "qualidade" do paralelismo. Na próxima seção é apresentada uma breve descrição das métricas comumente utilizadas na análise de códigos paralelos.

5.2 Métricas na computação paralela

O desempenho de uma aplicação paralela esta sujeita à perda de eficiência por vários fatores: latência e largura de banda, frequência de comunicação, frequência de sincronismo, escalonamento deficiente e impossibilidade de paralelismo de uma

parte da aplicação paralela [133].

Neste contexto, existem duas classes de métricas de desempenho: uma para os processadores e outra para as aplicações. As métricas utilizadas para medir o desempenho de processadores são as que avaliam a quantidade de FLOPs (*Floating point Operations Per Seconds*). Contudo, nosso interesse é identificar a "qualidade" do paralelismo de uma aplicação. Logo as métricas de desempenho destinadas às aplicações paralelas são o foco desta seção.

Speed up, ou simplesmente aceleração, é a métrica de desempenho mais usual para uma aplicação paralela. Ela é a razão entre o tempo de execução serial e o tempo de execução paralela, como descrita na Equação 5.1.

$$S = \frac{T_s}{T_p} \tag{5.1}$$

onde, T_s é o tempo de execução serial e T_p é o tempo de execução paralelo com p processadores. Todavia, essa métrica deve ser associada à métrica de eficiência de paralelismo, uma vez que a eficiência está associada com o grau de aproveitamento dos recursos computacionais. A eficiência em uma dada aplicação paralela é definida pela equação 5.2.

$$E_p = \frac{S}{p} \tag{5.2}$$

Os algoritmos em geral não são totalmente paralelizáveis, portanto, na maioria dos algoritmos paralelos existe um fração sequencial que limita a possibilidade de aumento do *speed up*. Desta forma, a influência da fração sequencial deve ser levado em consideração no cálculo de *speed up*.

O custo de uma aplicação paralela é dividido entre o tempo de computação serial, paralelo e comunicação. Então, o tempo total estimado para uma aplicação paralela, desprezando o tempo de comunicação, pode ser estimado pela lei de Amdahl [134]. A lei de Amdahl diz que um determinado problema A com um dado algoritmo paralelo B com p processadores tem tempo de computação total de:

$$T_p = \theta T_s + (1 - \theta)\frac{T_s}{p} \tag{5.3}$$

onde θ é a fração serial inerente a aplicação. Se essa fração for considerada constante o *speed up* é limitado pela Equação 5.4.

$$S_p = \frac{p}{1 + (p - 1)\theta} \tag{5.4}$$

Portanto, para um número infinito de processadores o *speed up* de uma aplicação pode atingir $S_\infty = 1/\theta$. Logo, o *speed up* máximo varia rapidamente com o aumento da fração serial de uma aplicação.

5.3 Modelos de Programação Paralela

Modelos de programação paralela são conjuntos de instruções que permitem a criação de aplicações paralelas compatíveis com uma dada arquitetura [132]. Teoricamente, qualquer modelo pode ser aplicado com qualquer tipo de arquitetura. Entretanto, alguns são mais eficientes em arquiteturas específicas. As formas de programação paralela podem ser baseadas em *threads* (abstração utilizada na divisão de tarefas entre processadores), em transferências de mensagens ou na forma híbrida. Para programação paralela em máquinas tipo MIMD, predominante nos dias de hoje, os modelos de programação mais usuais são os modelos OpenMP e MPI. Por conta disso, nas próximas seções esses dois modelos de programação mais comuns são apresentados.

5.3.1 OpenMP (*Open Multi-Processing*)

O modelo de programação OpenMP é utilizado para dividir tarefas entre processadores que compartilhem a mesma memória [135]. No padrão de programa OpenMP as tarefas são divididas, seguindo a ideia de que um *master thread* bifurca ou divide a tarefa entre os processadores *slaves*. Nessa forma de programação, os *threads* são gerados e executam tarefas de forma concorrente. Neste caso, seus dados privados são acessíveis a todos os *threads* que compartilham essa memória, ou seja, a latência de comunicação é pequena. No entanto, a desvantagem dessa forma de paralelismo está na possibilidade de haver dados corrompidos, caso haja falta de sincronismo entre as operações. Além disso, essa abordagem sofre por conta da dificuldade de escalabilidade da memória compartilhada.

A programação por OpenMP é feita por meio de diretivas de compilação que criam os *threads*, e formam as seções na memória antes delas serem usadas. Nesta forma de programação, cada *thread* é responsável por uma operação executada independentemente. Isso implica que o programador precisa identificar as etapas do código serial que podem ser executadas concorrentemente adicionando as diretivas de programação antes do trecho a ser paralelizado como no Exemplo 5.1.

```
1  int main(int argc, char **argv) {
       const int N = 100000;
3      int i, a[N];

5      #pragma omp parallel for
       for (i = 0; i < N; i++)
7          a[i] = 2 * i;

9      return 0;
   }
```

Código 5.1: código em OpenMP

Na linha 5 do exemplo 5.1 a diretiva *#pragma omp* avisa ao compilador que a execução deve ser realizada pelo OpenMP. A diretiva seguida da instrução *parallel for* permite que o *loop* seja bifurcado entre os processadores (*threads*), e desta forma a tarefa serial é divida entre os processadores. Mesmo não estando explícito no código acima, após o termino do *loop*, existe uma barreira que sincroniza todos os *threads*, e só assim o resto do código é executado por apenas um único *thread*.

5.3.2 MPI (*Message Passing Interface*)

MPI (*Message Passing Interface*) é um modelo de programação paralela que permite a troca de informação entre processadores por meio de um protocolo de rede. Esse tipo de paralelismo é mais flexível e depende mais do programador para identificar as funções MPI que devem ser utilizadas em uma dada aplicação. Na programação MPI cada processo recebe uma cópia do código, que é executado de forma diferenciada ou sobre diferentes conjuntos de dados [136].

Visando organizar a transferência de dados entre processadores, o MPI cria uma hierarquia de envio de dados. Essa hierarquia é dividida em comunicadores, grupos e *ranks*. Um grupo é o conjunto de processadores que são identificados pelo seus *ranks*. Portanto, cada operação de envio ou recebimento de mensagem, ponto a ponto (um processo envia mensagem para outro processo específico) ou coletiva (um processo envia mensagem para vários outros processos em um mesmo grupo), sempre se refere aos *ranks* de um grupo identificado com um dado comunicador.

O código 5.2 ilustra um exemplo simples de como programar em C usando MPI. O programa começa com a inclusão do cabeçalho *#include "mpi.h"*, seguido da declaração das variáveis de comunicação e inicialização do ambiente MPI na linha 15. Na linha 25, todos os processos enviam uma mensagem ponto a ponto de valor inteiro para o processador de *rank* zero.

```
1  #include "mpi.h"
2
3  main(int argc, char* argv[])
4  {
5
6  int rank;
7  int p;
8  int fonte;
9  int destino;
10 int tag = 0;
11 int res = 0;
12 int send;
13
14 MPI_Status status;
15 MPI_Init ( &argc, &argv);
16 MPI_Comm_rank ( MPI_COMM_WORLD, &rank);
17 MPI_Comm_size(MPI_COMM_WORLD, &p);
18
19 int *rec= new int[p];
20
21  if ( rank != 0 )
22   {
23     send=rank;
24     destino = 0;
25     MPI_Send(send,1, MPI_INT,destino, tag, MPI_COMM_WORLD);
26   }
27  else
28  {
29   for (fonte= 1; fonte < p; fonte++)
30    {
31     MPI_Recv(rec[fonte], 1, MPI_INT, fonte, tag,MPI_COMM_WORLD, &status
       );
32    }
33
34  }
35
36 MPI_Finalize();
37
38 }
```

Código 5.2: código em MPI

Na linha 31, o processador de *rank* zero recebe as mensagens dos outros processadores, em seguida na linha 36 o ambiente MPI é finalizado.

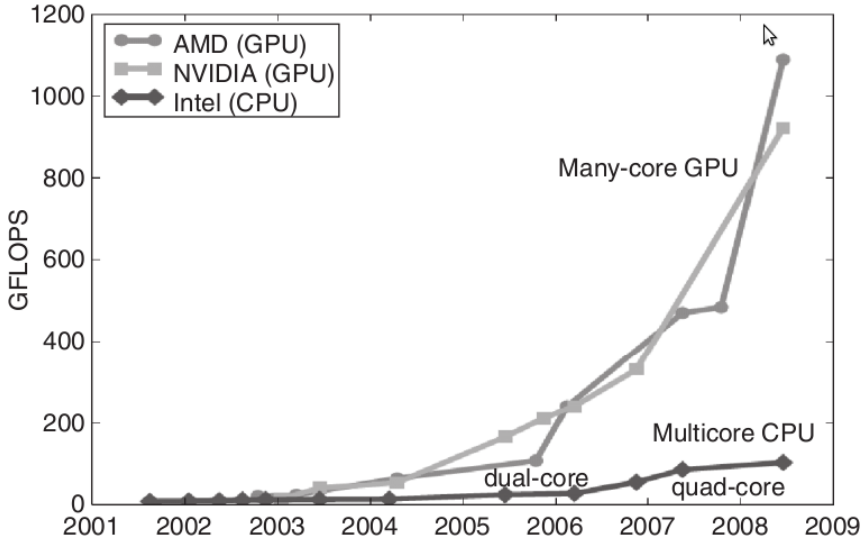


Figura 5.1: Crescimento em GFLOPs das placas gráficas ao longo dos anos (NVIDIA CORPORATION [2]).

5.4 GPU (*Graphics Processing Unit*)

Nos últimos anos, os *hardwares* de aceleração gráfica se desenvolveram vertiginosamente motivados pela indústria de jogos [34]. As placas gráficas aceleradoras surgiram na década de 70 com a família de jogos Atari. Contudo, apenas em 2001 a NVIDIA deu o primeiro passo no desenvolvimento de placas gráficas programáveis voltadas à computação científica. Esse fato ocorreu mais uma vez motivado pela insaciável necessidade de alta definição dos gráficos 3D que, nessa época já requeria um alto grau paralelismo e poder computacional em apenas uma única placa de vídeo [137]. Neste contexto, as GPUs (Unidade de Processamento Gráfico), antes usadas apenas para processamento gráfico passaram a ser usadas para outros propósito, encontrando aplicações em vários ramos da ciência. Note que, mesmo com o desenvolvimento de CPUs mais potentes, o poder computacional das GPUs cresceu em uma taxa muito maior, como pode ser visto na Figura 5.1.

A razão pela qual as GPUs são mais poderosas computacionalmente está na forma com que sua arquitetura foi desenvolvida. Ao contrário das CPUs, as GPUs dedicam um número muito maior de transistores para processamento que para armazenamento e controle do fluxo de dados [2], como mostra a Figura 5.2. Outro aspecto de diferenciação está na quantidade de memória *cache* disponíveis na GPU. Nas GPUs, existe um número muito maior desse tipo de memória, o que possibilita maior largura de banda e execução de maior quantidade de *threads* simultaneamente. Por isso, enquanto que em uma CPU o número de *threads* é muito limitado nas GPUs o número de *threads* é de até 1024 por multiprocessador [2].

Por exemplo, a arquitetura Fermi desenvolvido pela NVIDIA, consiste de pelo

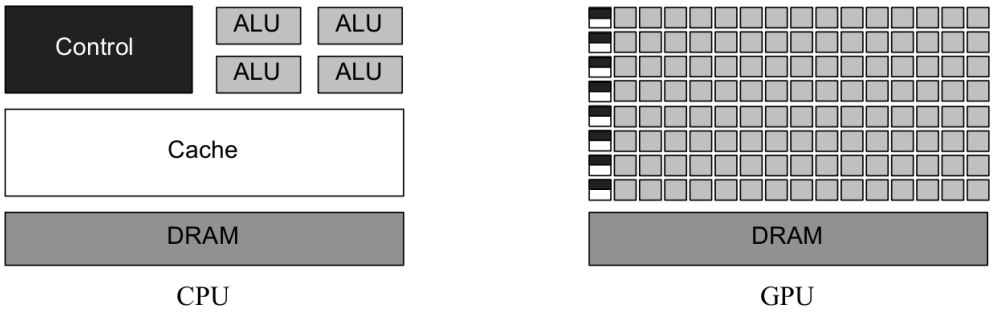


Figura 5.2: Diferenças entre a arquitetura das GPUs e CPUs. Em cinza claro estão as ALUs (unidades lógicas e aritméticas de processamento), o controlador de fluxos de dados está em preto e em branco e cinza escuro estão representadas as memórias *caches* e RAM, respectivamente (NVIDIA CORPORATION [2]).

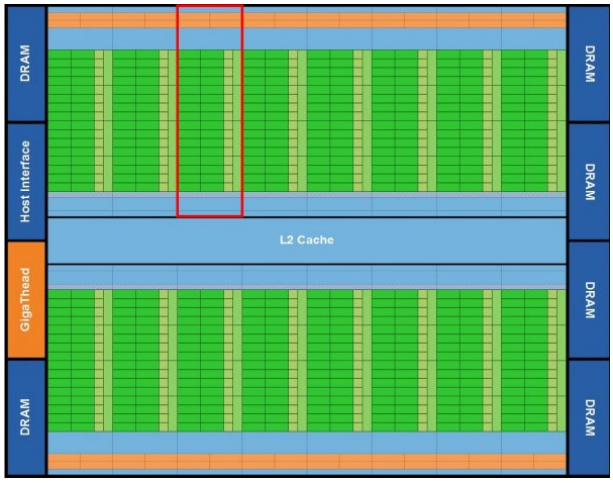


Figura 5.3: Visão geral da arquitetura Fermi (NVIDIA CORPORATION [2]).

menos 15 multiprocessadores. A Figura 5.3 mostra um exemplo dessa arquitetura com 512 CUDA *cores*, organizados em 16 multiprocessadores, ou seja, 32 CUDA *cores* por multiprocessador. Esses multiprocessadores dividem uma memória *cache* L2, seis memórias RAMs e uma interface de comunicação entre a GPU e CPU. Além disso, há um escalonador que divide os *threads* entre os *cores* dos multiprocessadores. Na Figura 5.4, é apresentada uma visão detalhada de um multiprocessador (SM-*streaming multiprocessor*) CUDA. Em cada SM existe em seu *chip* dois tipos de memória, os registradores e a memória compartilhada. Há também unidades para operar funções especiais na arquitetura (SFU), como por exemplo funções transcendentais [136].

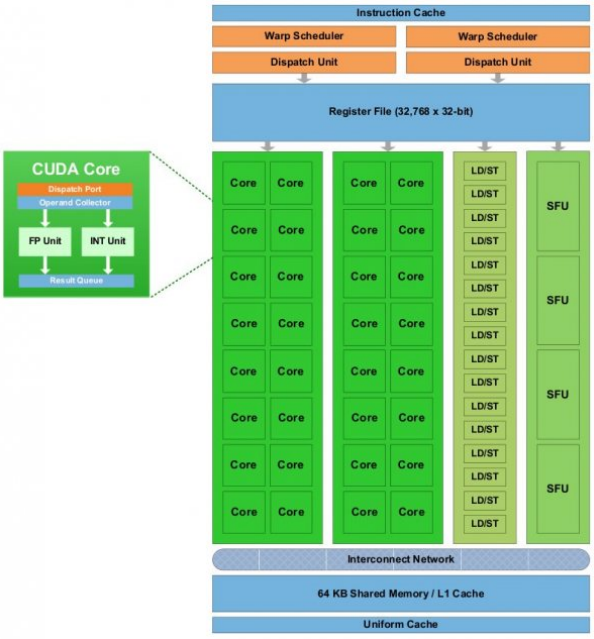


Figura 5.4: Multiprocessador da arquitetura Fermi (NVIDIA CORPORATION [2]). FP Unit = Unidade de Operação em ponto flutuante. INT Unit = Lógica aritmética inteira. LD/ST= Leitura e armazenamento de dados.

5.4.1 Modelo de Programação CUDA

CUDA (*Compute Unified Device Architecture*) é o modelo de programação paralela que permite desenvolver aplicações paralelas de propósito geral em placas gráficas da NVIDIA [136]. Para entender o funcionamento dessa forma de programação, cabe compreender como funciona o fluxo de dados no processamento paralelo das GPUs. As GPUs (*devices*) são vistas como co-processadores, que executam múltiplos *threads* simultaneamente, e necessitam da CPU (*host*) para realizar o controle de fluxo de dados da aplicação paralela.

Em CUDA, o programa se inicia com as funções chamadas de CUDA *Kernels*, onde cada uma delas especifica um código a ser executado N vezes por diferentes N *threads*. Note que, como cada *thread* executa a mesma instrução, esse modelo de programação é incluído na categoria SIMD ou SPMD (*Single Program Multiple Data*). A estrutura de organização dos *Kernels* é divida em *grid*, *blocks* e *threads*. Cada *Kernel* é representado por um *grid*, que por vez contém um conjunto de *blocks*, e cada *blocks* possuí um conjunto de *threads*. Tanto os *blocks* como os *threads* possuem uma identificação única, que podem ter até 3 dimensões. Para identificá-los em um *grid*, existem identificadores, que são *blockIdx.x*, *blockIdx.y* e *blockIdx.z* para os *blocks* e *threadIdx.x*, *threadIdx.y* e *threadIdx.z* para os *threads* em seus respectivos *blocks*. Além disso, cada *thread* possui um único identificador no *grid* que podem

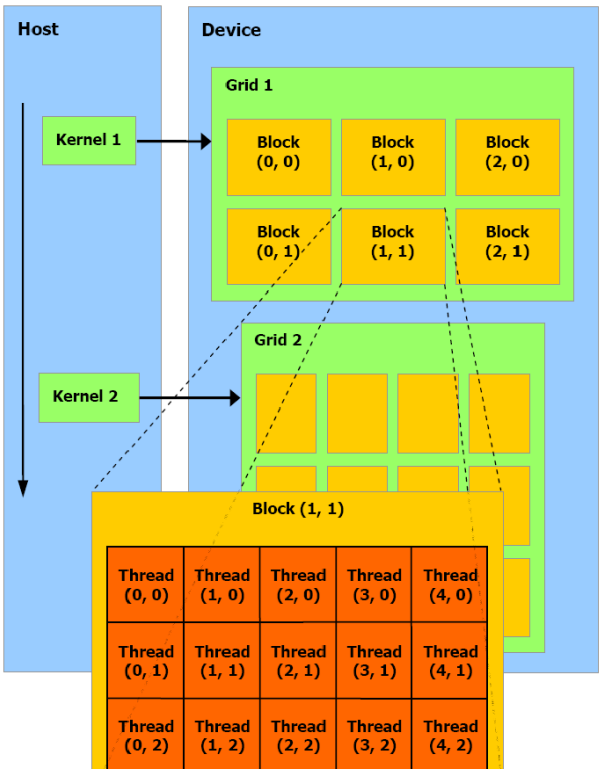


Figura 5.5: Fluxo de dados no paralelismo em cuda (NVIDIA CORPORATION [2]).

ser determinada a partir da Equação 5.5.

$$Idx = blockDim.x \times blockIdx.x + threadIdx.x \tag{5.5}$$

onde, *blockDim.x* representa a dimensão do *block* na direção x e a variável Idx indica a posição do *thread* no *grid*.

A Figura 5.5 ilustra a organização dos *blocks* e *threads* em um *grid*. Neste caso o *host* executa o *Kernel* em 6 *blocks* organizados bidimensionalmente (2×3) e cada *block* contém 15 *threads* (3×5). Importante destacar que os *Kernels* podem ser executados de forma concorrente nas arquiteturas Fermi, apesar de parecer que isso não é possível de acordo com a Figura 5.5.

Outro aspecto importante na programa em CUDA é a eficiência no acesso das memórias disponíveis nas GPUs, uma vez que otimizar aplicações paralelas em CUDA requer o uso apropriado de diferentes tipos de memória. As memórias mais velozes estão localizadas dentro dos *chips* dos multiprocessadores, sendo os registradores e as memórias compartilhadas. Os registradores são memórias acessíveis a cada *threads* individualmente, enquanto que as memórias compartilhadas são acessíveis a todos os *threads* de um mesmo bloco. Portanto, os dados armazenados na memória compartilhada e nos registradores possuem o mesmo tempo de vida dos *blocks*. As memórias localizadas fora dos multiprocessadores são cerca de 100 vezes mais lentas que as memórias internas aos multiprocessadores, e por isso, o gerenciamento de

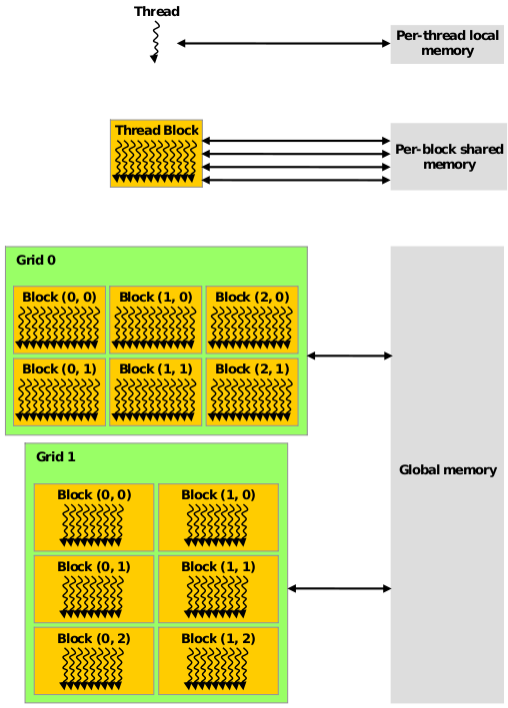


Figura 5.6: Tipos de memórias diferentes em GPUs (NVIDIA CORPORATION [2]).

memória é muito importante para eficiência da aplicação. As memórias globais são as que permitem a comunicação entre a CPU e a GPU, elas são acessíveis a todos os *threads* existentes no *Kernel*. A memória local a cada thread é uma abstração, e não um componente de *hardware*, localizado fisicamente na memória global, e portanto, as duas têm a mesma velocidade. Essa memória é de escopo local de cada *thread*, ou seja, quando alguma variável é declarada na memória local por um dado *thread*, apenas o mesmo terá acesso a esta variável. Logo, o uso otimizado das memórias compartilhada e dos registradores pode garantir uma eficiência maior a uma dada aplicação paralela.

As memórias de textura e constante são acessíveis a todos os *threads*. Essas duas últimas são apenas memórias para leitura e são utilizadas para aplicações específicas. Por exemplo as memórias de textura são especializadas em interpolações lineares.

Abaixo um exemplo de programação CUDA. Na linha 2 do código 5.3, o especificador __global__ cria a função *Kernel* que é executada na GPU. Dentro dessa função qualquer variável declarada sem nenhum qualificador será declarada na memória local, como por exemplo a variável *Aux*. Na linha 4, a variável i percorre todos os *threads* e soma as variáveis dos vetores **A** e **B** simultaneamente em cada *thread* i.

Já na função *main*, que é executada na CPU, das linhas 19 a 23, as variáveis são alocadas na memória global da GPU. Nas linhas 30 e 31, as variáveis declaradas na

CPU são copiadas para a memória das GPUs, seguido da chamada do *Kernel* na linha 35. Na linha 37, o resultado computado pela GPU é copiado da sua memória global para CPU, e logo depois a memória global da GPU é liberada.

```
/* Parte GPU */
__global__ void VecAdd(int N,float* A, float* B, float* C)
{
int i = blockDim.x * blockIdx.x + threadIdx.x;
double Aux;
 if (i < N)
  {
    Aux = A[i] + B[i];
    C[i]=__sqrt(Aux);
  }
}
/* Parte CPU */
int main()
{

int N=...
size_t size = N * sizeof(float);
float* d_A;
cudaMalloc((void**)&d_A, size);
float* d_B;
cudaMalloc((void**)&d_B, size);
float* d_C;
cudaMalloc((void**)&d_C, size);

float* h_A=malloc(size);
float* h_B=malloc(size);
float* h_C= ...
       ...;

cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
/* Chama o Kernel da GPU */
int threadsPerBlock = 256;
int BlocksPerGrid =(N + threadsPerBlock - 1) / threadsPerBlock;
VecAdd<<<threadsPerGrid, threadsPerBlock>>>(N,d_A, d_B, d_C);

cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
cudaFree(d_A);
cudaFree(d_B);
cudaFree(d_C);
}
```

Código 5.3: Exemplo de código em CUDA

Capítulo 6

Desenvolvimento da Solução PB-CFD usando GPUs

Neste capítulo, o desenvolvimento realizado neste trabalho é apresentado. Primeiramente, é apresentada uma breve dedução do D^{2}uQMoGeM em sua forma *Fully-Conservative*, seguida da descrição dos algoritmos paralelos de cubatura adaptativa e de solução de sistemas lineares. Finalmente, é descrito o acoplamento do D^{2}uQMoGeM-FC (*Fully-Conservative*) com o OpenFOAM® para a solução da PBE utilizando os algoritmos paralelos aqui desenvolvidos.

6.1 Métodos de Quadratura Dupla Baseados em Momentos

Estes métodos surgiram com a proposta de solucionar o problema relativo aos erros de quadratura nos QBMMs (*Quadrature-Based Moment Methods*). A essência desse tipo de método está na utilização de uma segunda quadratura, além da de Gauss-Christoffel, para o controle de erro associado às integrais dos termos fonte. Os resultados obtidos tanto por LAGE [29] quanto por FAVERO e LAGE [30] indicam que os métodos de quadratura dupla, como o DuQMoGeM, podem ser ótimos candidatos para o acoplamento PB-CFD. No presente trabalho, a evolução natural do DuQMoGeM para sua forma direta foi formulado, devido a possibilidade de advectar os pesos e abscissas como um conjunto de N fases dispersas. Nesse contexto, nas duas subseções subsequentes, é feita uma breve descrição dos métodos de quadratura dupla, DuQMoGeM [29] e D^{2}uQMoGeM [138] na sua forma totalmente conservativa.

6.1.1 DuQMoGeM-FC

O método DuQMoGeM é o ponto de partida para o D^{2}uQMoGeM. Portanto, para compreensão do método D^{2}uQMoGeM é interessante descrever o seu antecessor, uma vez que suas formulações tem origens semelhantes.

Para dedução do DuQMoGeM, uma representação contínua da FDN com uma base ortogonal polinomial conveniente é utilizada [29, 30] de acordo com a Equação 6.1.

$$f(m,\mathbf{x},t) = w(m)\sum_{i=0}^{2N-1} c_i(\mathbf{x},t)\phi_i(m), \tag{6.1}$$

onde, $\phi_i(m)$ é a base, $w(m)$ é a função peso de $\phi_i(m)$, e c_i são as constantes da aproximação funcional da Equação 6.1. Estas constantes, podem ser determinadas a partir da Equação 6.2.

$$\langle \phi_i, \phi_j \rangle_{d\bar{\lambda}(m)} = \int_0^{m_{max}} \phi_i(m)\phi_j(m)w(m)\,\mathrm{d}m = \delta_{ij}\|\phi_i\|^2_{d\bar{\lambda}}, \tag{6.2}$$

em que, $d\bar{\lambda}(m) = w(m)dm$, δ_{ij} é a função Delta de Kronecker e $\|\phi_i\|$ é a norma da base polinomial ϕ_i. Portanto, $c_i(\mathbf{x},t)$ são calculados por:

$$c_i(\mathbf{x},t) = \frac{\mu_i^{\phi}(\mathbf{x},t)}{\|\phi_i\|^2_{d\bar{\lambda}(m)}}, \qquad i = 0,\ldots,2N-1, \tag{6.3}$$

onde μ_i^{ϕ} são os momentos generalizados da FDN, dados pela equação abaixo,

$$\mu_i^{\phi}(\mathbf{x},t) = \int_0^{m_{max}} \phi_i(m)f(m,\mathbf{x},t)\,\mathrm{d}m, \qquad i = 0,\ldots,2N-1. \tag{6.4}$$

Finalmente, aplicando o operador momento generalizado, $\int_0^{m_{max}} \phi_j(m)(.)\,\mathrm{d}m$, na PBE da Equação 2.21, considerando apenas o termo convectivo, os efeitos de quebra e agregação e um termo fonte adicional, substituindo a aproximação da Equação 6.1 na equação de momentos da PBE, temos que após algumas manipulações (ver Anexos C e D), a equação de momentos se torna,

$$\begin{aligned}\frac{d\mu_j^{\phi}}{dt} + \nabla_x \cdot \left[\int_0^{m_{max}} \mathbf{u}_p(m_\alpha,\mathbf{x},\mathbf{Y},t)f(m,\mathbf{x},t)\phi_j(m)\,\mathrm{d}m\right] + \sum_{i=0}^{2N-1}\sum_{k=0}^{2N-1} A_{jik}\frac{\mu_i^{\phi}}{\|\phi_i\|^2_{d\bar{\lambda}(m)}}\frac{\mu_k^{\phi}}{\|\phi_k\|^2_{d\bar{\lambda}(m)}} \\ + \sum_{i=0}^{2N-1} L_{ji}\frac{\mu_i^{\phi}}{\|\phi_i\|^2_{d\bar{\lambda}(m)}} = s_j\end{aligned} \tag{6.5}$$

onde,

$$A_{jik} = \int_0^{m_{max}} \int_0^{m_{max}} a(m, m') \left[\phi_j(m) - \frac{1}{2}\phi_j(m + m') \right] \phi_i(m)\phi_k(m')w(m)w(m')dmdm', \tag{6.6}$$

$$L_{ji} = \int_0^{m_{max}} b(m, t) \left[\phi_j(m) - 2\Pi_j^{\phi}(m) \right] \phi_i(m)w(m)dm, \tag{6.7}$$

$$\Pi_j^{\phi}(m) = \int_0^{m_{max}} \phi_j(m')P(m'|m)dm', \tag{6.8}$$

$$s_j = \int_0^{m_{max}} \phi_j(m)S(m, t)dm. \tag{6.9}$$

O termo advectivo na Equação (6.5) deve ser discretizado usando a quadratura de Gauss-Christoffel da FDN, como na Equação 6.10,

$$f(m, \mathbf{x}, t) \backsimeq \sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{x}, t)\delta(m - m_\alpha(\mathbf{x}, t)), \tag{6.10}$$

onde $\delta(m - m_\alpha)$ é a função Delta de Dirac, e ω_α e m_α são, respectivamente, os pesos e abscissas de quadratura. Usando a Equação (6.10), o termo advectivo é dado por:

$$\nabla_x \cdot \left[\int_0^{m_{max}} \mathbf{u}_p(m_\alpha, \mathbf{x}, \mathbf{Y}, t)f(m, \mathbf{x}, t)\phi_j(m)\,\mathrm{d}m \right] \backsimeq \overline{F}_j^N = \sum_{\alpha=1}^{N} \nabla_x \cdot \left[\omega_\alpha \mathbf{u}_\alpha \phi_j(m_\alpha) \right], \tag{6.11}$$

em que, $\mathbf{u}_p(m_\alpha, \mathbf{x}, \mathbf{Y}, t) = \mathbf{u}_\alpha(\mathbf{x}, t)$ é a velocidade da fase dispersa α. Definindo então, $\overline{R}_j^{(N)}$, como,

$$\overline{R}_j^{(N)} = \sum_{i=0}^{2N-1} \sum_{k=0}^{2N-1} A_{jik} \frac{\mu_i^{\phi}}{\|\phi_i\|_{d\bar{\lambda}}^2} \frac{\mu_k^{\phi}}{\|\phi_k\|_{d\bar{\lambda}}^2} + \sum_{i=0}^{2N-1} L_{ji} \frac{\mu_i^{\phi}}{\|\phi_i\|_{d\bar{\lambda}}^2} \tag{6.12}$$

e usando a Equação (6.11), a equação de momento para o DuQMoGeM, Equação (6.5), pode ser escrita como:

$$\frac{\partial \mu_j^{\phi}}{\partial t} = -\overline{R}_j^{(N)} - \overline{F}_j^N + s_j. \tag{6.13}$$

Após a descrição do DuQMoGeM-FC, a sua extensão para forma a direta é facilmente compreendida. Na próxima subseção, a dedução do D^{2}uQMoGeM-FC é apresentada de forma condensada.

6.1.2 Direct DuQMoGeM-FC (D^{2}uQMoGeM)

Como no DQMoM, D^{2}uQMoGeM [138] usa a quadratura de Gauss-Christoffel para discretização da FDN presente na Equação 6.10 [104]. Devido a acurácia dos N-pontos de quadratura de Gauss-Christoffel, os $2N$ primeiros momentos podem ser calculados exatamente por:

$$\mu_i^{\phi}(\mathbf{x},t) = \sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{x},t)\phi_i(m_\alpha(\mathbf{x},t)) \tag{6.14}$$

Portanto, combinando as equações 6.14, 6.1 e 6.3, a aproximação da FDN para o D^{2}uQMoGeM é dado por,

$$f(m,\mathbf{x},t) = \frac{w(m)}{\|\phi_i\|^2_{d\bar{\lambda}}} \sum_{i=0}^{2N-1} \left[\sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{x},t)\phi_i(m_\alpha(\mathbf{x},t)) \right] \phi_i(m) \tag{6.15}$$

O D^{2}uQMoGeM é deduzido pela substituição da Equação 6.15 no termo $H(m,\mathbf{x},t)$ e da equação 6.10 no termo advectivo e transiente da equação de balanço populacional 2.21, porém apenas considerando os efeitos de quebra e agregação. Essas operações levam a seguinte equação de momentos em termos dos pesos e abscissas da quadratura de Gauss-Christoffel:

$$\begin{aligned} &\sum_{\alpha=1}^{N} \left[\phi_j(m_\alpha) - m_\alpha \phi_j'(m_\alpha)\right] \gamma_\alpha + \sum_{\alpha=1}^{N} \phi_j'(m_\alpha)\theta_\alpha = -\sum_{i=0}^{2N-1} L_{ji} \sum_{\alpha=1}^{N} \frac{\omega_\alpha \phi_i(m_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}} \\ &-\sum_{i=0}^{2N-1}\sum_{k=0}^{2N-1} A_{jik} \sum_{\alpha=1}^{N} \frac{\omega_\alpha \phi_i(m_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}} \sum_{l=1}^{N} \frac{\omega_l \phi_k(m_l)}{\|\phi_k\|^2_{d\bar{\lambda}}} = -\overline{R}_j^{(N)} + s_j, \end{aligned} \tag{6.16}$$

onde $\phi_j' = d\phi_j/dm$ e γ_α e θ_α são definidos como termos fonte da equação de conservação para pesos e abscissas ponderadas, $\zeta_\alpha = \omega_\alpha m_\alpha$:

$$\frac{\partial \omega_\alpha}{\partial t} + \nabla.(\omega_\alpha u_\alpha) = \gamma_\alpha, \tag{6.17}$$

$$\frac{\partial \zeta_\alpha}{\partial t} + \nabla.(\zeta_\alpha u_\alpha) = \theta_\alpha. \tag{6.18}$$

As Equações (6.16), (6.17) e (6.18) estão na forma original do D^{2}uQMoGeM [31]. Para sua versão totalmente conservativa, o termo advectivo da equação deve fazer parte do termo fonte do sistema linear gerado para o D^{2}uQMoGeM. Além disso, a formulação proposta para o D^{2}uQMoGeM-FC usa equações de transporte para pesos e abscissas. Portanto, as equações para o D^{2}uQMoGeM-FC são dados por:

$$\frac{\partial w_\alpha}{\partial t} = A_\alpha, \quad \frac{\partial m_\alpha}{\partial t} = B_\alpha, \tag{6.19}$$

onde A_α e B_α são obtidos pela solução do seguinte sistema linear:

$$\sum_{\alpha=1}^{N} \phi_j(m_\alpha)A_\alpha + \sum_{\alpha=1}^{N} w_\alpha \phi'_j(m_\alpha)B_\alpha = -\overline{R}_j^{(N)} - \overline{F}_j^N + s_j, \tag{6.20}$$

sendo que $\overline{F}_j^N$ e $\overline{R}_j^{(N)}$ são dados pelas Eqs. (6.11) e (6.12), respectivamente. Quando necessário, μ_i^ϕ é calculado pela Equação (6.14).

O leitor interessado em detalhes de dedução desse método pode encontrá-lo no apêndice C referente a sua publicação associada [31].

6.2 Cubatura Adaptativa

Integrais multidimensionais, como da Equação 6.21, estão presentes em vários ramos da ciência. Entretanto, em muitos dos casos não há solução analítica fechada, e por isso, pesquisadores vem tentando desenvolver técnicas numéricas para computar integrais multidimensionais com precisão e acurácia (SCHURER [139], D'APUZZO *et al.* [140], LI e DAGNINO [141] entre outros).

$$\mathbf{F} = \int_{C_s} \mathbf{h}(\mathbf{x})d\mathbf{x}, \quad \mathbf{h} : C_s \longrightarrow \mathbb{R}^{Ndim}, \quad C_s \in \mathbb{R}^s. \tag{6.21}$$

Entre as técnicas de integração numérica mais usadas estão o método de Monte Carlo, Quasi-Monte Carlo, que são tipicamente usados para problemas com alta dimensionalidade. Outras técnicas que figuram entre as mais usadas também estão as cubaturas e o produto das quadraturas de Gauss, sendo esses mais acuradas para problemas de baixa dimensionalidade (dimensão < 7) [142], como por exemplo, as integrais presentes na equação de balanço populacional. Desta forma, nesta seção serão descritos os conceitos básicos de integração por cubatura.

Maz [143] compararam as cubaturas e o produto das quadraturas de Gauss. Eles concluíram que as cubaturas podem obter a mesma acurácia que o produto das quadraturas de Gauss, só que com menos pontos de quadratura, o que confere uma vantagem às cubaturas. Fundamentalmente, uma cubatura é definida da seguinte forma [144]:

$$\mathbf{F}_M \approx \sum_{i=1}^{M} w_i \mathbf{h}(\mathbf{x_i}) \tag{6.22}$$

onde M é o número de pontos de cubatura, $\mathbf{x_i} \in C_s$ são as M abscissas e w_i seus pesos correspondentes. Como a Equação 6.22 nos fornece apenas uma aproximação da integral [145], um algoritmo adaptativo de subdivisão do subdomínio pode ser usado para atingir a acurácia desejada. Note que no processo de divisão, uma nova sub-região precisa ser gerada. Para tal, uma direção para divisão deve ser

Algoritmo 1: Algoritmo de cubatura adaptativa

Coloque as regiões iniciais em uma coleção de regiões.
para todas(os) *regiões da coleção* ***faça***
  Aplicar as regras de cubatura.
  Selecionar a direção de subdivisão.
  Computar a estimativa de erro.
fim do percorre
Computar a estimativa do erro global.
enquanto *a tolerância não é atingida e o número máximo de iterações não for atingido* ***faça***
  Escolha a região com maior erro abscoluto.
  Divida a região em pelo menos duas novas regiões.
  para todas(os) *novas regiões* ***faça***
    Aplicar as regras de cubaturas.
    Selecione a direção de subdivisão.
    Compute a estimativa de erro.
  fim do percorre
  Armazene estas novas regiões na coleção de regiões.
  Compute a estimativa do erro global.
fim do enquanto

selecionada. A escolha dessa direção afeta significativamente a taxa de convergência do processo adaptativo. Normalmente, a direção de divisão é escolhida de acordo com a maior variação no valor do integrando na dada sub-região [145]. Por fim, o algoritmo de cubatura adaptativa, dado abaixo no Algoritmo 1, termina quando o erro global, o qual é soma dos erros absolutos de todas as sub-regiões, satisfaz um tolerância especificada ou quando o número máximo de sub-regiões for atingido [145].

Visando acelerar esse algoritmo, pesquisadores propuseram várias formas de paralelismo. De acordo com GENZ [146], existem 6 formas de paralelismo para o cálculo de cubaturas adaptativas. Essas formas são diferenciadas pelo trecho ou nível que o algoritmo é paralelizado. Dentre os níveis de paralelismo mais importantes estão: o paralelismo ao nível do problema de integração, ao nível da subdivisão do domínio, ao nível das regiões, ao nível do cálculo na fórmula de cubatura (IFL) e ao nível do integrando (IL).

O paralelismo ao nível do problema de integração é aplicado quando o número de integrandos é elevado, pois nesse nível de paralelismo, cada integrando ou conjunto de integrandos é calculado independentemente por um processo usando um algoritmo sequencial, sem intercomunicação entre cada processo. Sua vantagem está no fato de não necessitar de comunicação constante entre os núcleos. No entanto, se uma das integrais for significativamente mais complicada, a aceleração do algoritmo é muito prejudicada, uma vez que a integral limitante é integrada de forma serial em um único núcleo.

No paralelismo ao nível da subdivisão do domínio, cada processo calcula uma parcela do domínio total. Essa técnica é bem eficiente. No entanto, pode ocorrer desbalanceamento de carga entre os nós, o que reduz a eficiência do paralelismo. Por exemplo se a região limitante estiver localizada totalmente em apenas um nó o *speed up* da aplicação fica limitado a 1.

O paralelismo ao nível das regiões utiliza o conceito de processos *Master* e *Slaves*. Nessa forma de paralelismo, o processo de divisão do domínio é feito de forma serial no processo *Master* e as regiões geradas são distribuídas entre os processos *Slaves*. Esse algoritmo é muito eficiente, porém esse método amarra o número de divisões ao número de nós disponíveis.

No caso do paralelismo ao nível de cálculo da fórmula de cubatura, a regra de cubatura é paralelizada, em outras palavras, cada processo é responsável por um conjunto de abscissas nos quais os integrandos são avaliados. O paralelismo ao nível de integrando é usualmente aplicado quando o número de integrandos é elevado, pois, nessa técnica, a integral de cada integrando é calculada por um processo. Essa técnica é bem semelhante ao algoritmo ao nível do problema de integração. Entretanto, no paralelismo ao nível do integrando, todas as etapas adaptativas são executadas de forma serial (no processo *Master*), e apenas a avaliação da integral na região é feita de forma paralela. Essa técnica é recomendada para máquinas paralelas do tipo SIMD ou SPMD como nas GPUs [139]. A grande desvantagem desse procedimento de paralelismo é a quantidade de comunicação necessária, e a restrição de todas as integrais terem o mesmo domínio de integração, uma vez que etapa adaptativa é feita de forma serial.

Atualmente, a maioria dos algoritmos implementados são ao nível da subdivisão do domínio [139, 147] porque essa estratégia é facilmente aplicável e adequada para arquiteturas tipo MIMD (*multiple-instruction multiple-data*). Existem vários pacotes que computam o algoritmo de cubatura adaptativa [139, 142, 148]. No entanto, nenhum deles usufruem do paralelismo em GPU. Como já mencionado, os algoritmos paralelos de cubatura adaptativas são geralmente implementados em arquiteturas MIMD e restritos ao nível da subdivisão do domínio. No entanto, no presente trabalho, este algoritmo foi implementado em GPUs usando o paralelismo ao nível do integrando e ao nível do cálculo na fórmula de cubatura. Essas formas de paralelismo foram escolhidas devido a sua melhor aplicabilidade para arquiteturas SIMD ou SIMT (*single-instruction multiple threads*). Isso ocorre porque, para tirar maior vantagem dessa arquitetura, cada integrando pode ser computado independentemente [139]. Em cada região, a integral e seu erro são computados usando as regras entrelaçadas de quinta e sétima ordem desenvolvida por GENZ e MALIK [149]. A direção com o maior valor do operador diferença de quarta ordem (FDDO) definido em GENZ e MALIK [149], expressa a direção com a qual a sub-região é

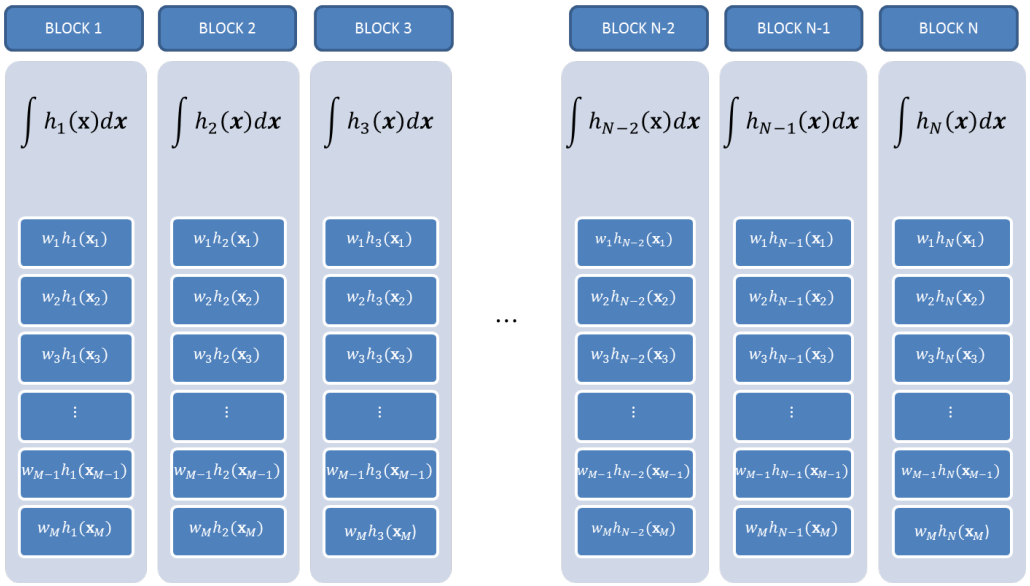


Figura 6.1: Paralelismo PIFL com M pontos de cubatura e N integrandos.

dividida.

Para o critério de parada, o erro global é dado pela soma da magnitude do erro absoluto local estimado de todas as sub-regiões. Observe que essa medida de erro global é conservadora, pois não permite a possibilidade de cancelamento entre erros negativos e positivos de diferentes sub-regiões. Assim, o algoritmo adaptativo termina quando o número máximo de iterações é atingindo ou esse erro global é menor que a tolerância mista ε, definida abaixo:

$$\varepsilon = \max\left(\varepsilon_{abs}, \varepsilon_{rel}\,|\mathbf{F}_M|\right) \tag{6.23}$$

onde, os valores para as tolerâncias relativa, ε_{rel}, e absoluta, ε_{abs}, são especificados, e $\mathbf{F}_M$ é o valor numérico da integral sobre todo domínio.

6.3 Paralelismo IFL (PIFL)

Nesta forma de paralelismo, o *Kernel* CUDA é responsável por computar integrais de um vetor de integrantes sobre uma dada sub-região. Isso significa que cada *block* de um *grid* calcula a integral por cubatura de um componente do vetor de integrandos, enquanto que cada *thread* computa um ponto de cubatura da integral correspondente ao *block*.

A Figura 6.1 mostra o PIFL aplicado a um problema com N integrandos e M pontos de cubatura, na qual existem pelo menos M *threads* por *block* e N *blocks*. Além disso, como o FDDO usa os pontos de cubatura, cada *block* também calcula o valor desse operador com intuito de encontrar a direção de divisão do subdomínio.

Visando sobrepor o cálculo de diferentes sub-regiões, os pontos de cubatura e o

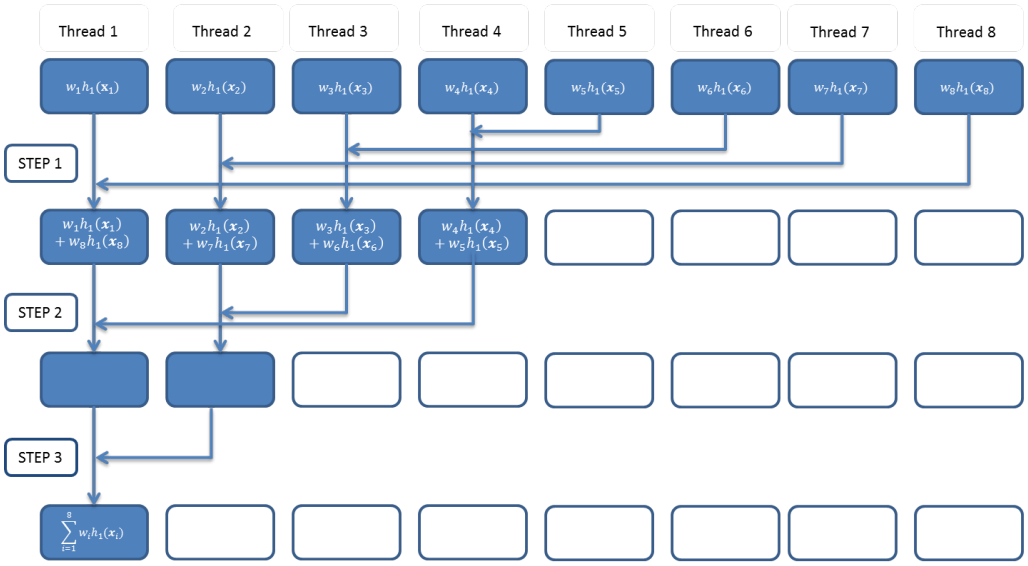


Figura 6.2: Algoritmo de soma em cada *block*.

intervalo de integração são copiados da CPU para a GPU usando cópias assíncronas. Essa estratégia também foi usada na cópia do vetor de integrais, do erro absoluto das sub-regiões e do vetor FDDO da GPU para a CPU. Essa particularidade é possível nas arquiteturas de GPUs iguais ou superiores a Fermi onde cada GPU pode executar múltiplos *kernels* baseado na disponibilidade da GPU.

Para cada integrando, os pontos de cubatura calculados pelos *threads* devem ser adicionados para obter o valor da integral associado a um *block*. Este é realizado pelo algoritmo chamado redução paralela que faz uso da memória compartilhada pelos *threads* de cada *block* [150]. O principal conceito desse algoritmo é usar cada *thread* na adição de dois valores de ponto de cubatura, $w_j h_i(\mathbf{x_j})$, que está disponíveis na memória compartilhada da GPU, sendo em seguida armazenado nesta mesma memória. Esse procedimento é realizado em $log_2(M)$, onde M é o número de pontos de cubatura. Ao fim de todos as etapas, o resultado final da soma dos pontos de cubatura é armazenado em uma variável associada ao índice do *block*.

Por exemplo, a Figura 6.2 ilustra o algoritmo paralelo de redução (soma) para $M = 8$, onde o ponto de cubatura calculado pelo *thread 1* é adicionado ao ponto de cubatura calculado pelo *thread 8*, e então, o resultado da soma é armazenado na memória compartilha com o índice correspondendo ao *thread 1*. Isso é realizado para todos os pares de *threads* com índices j e $M - j + 1$ nesta etapa do algoritmo. Após 3 etapas concluídas, a soma, $\sum_{j=1}^{8} w_j h_1(\mathbf{x_j})$, é armazenada na memória compartilha da GPU, que em seguida é copiada para memória global da GPU, usando o índice correspondente ao bloco.

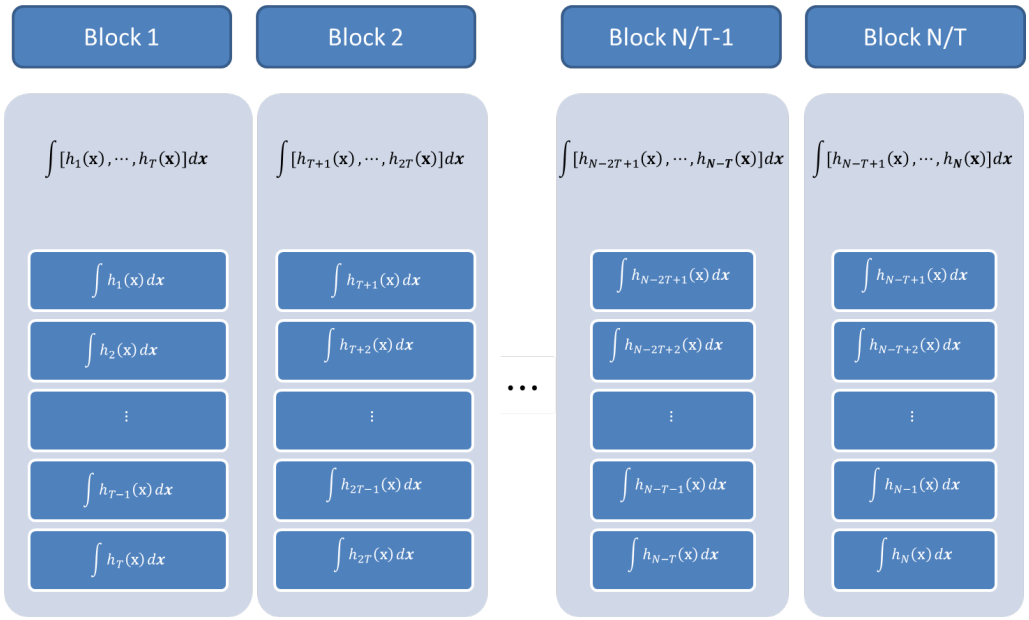


Figura 6.3: Paralelismo PIL com T integrandos e *threads* por *block* e um total de N integrandos.

6.4 Paralelismo IL (PIL)

Neste nível de paralelismo, cada *CUDA thread* calcula uma integral do vetor de integrandos. Neste caso, cada *block* computa T integrandos, e portanto, obtém T integrais. A Figura 6.3 descreve este algoritmo. O operador FDDO também é calculado no nível dos *threads*. Novamente, a técnica de cópias assíncronas é utilizada. Então, os pontos de cubatura e intervalos de integração são copiados da CPU para a GPU, e o resultado do vetor de integrais, os seus erros absolutos e os valores do operador FDDO são copiados da GPU para a CPU de forma assíncrona.

Para ser eficiente, esse algoritmo paralelo necessita de um alto número de integrandos, algo que é típico nos métodos tipos DQBMMs, especialmente, quando estes são aplicados em simulações CFD. Por exemplo, em uma PBE bivariada que inclui os efeitos de quebra e agregação, se o DuQMoGeM em sua solução utilizar $N = 3$ pontos de quadratura de Gauss-Christoffel, 810 integrandos são gerados. Note que, se a solução desse PBE bivariada for acoplada com uma simulação CFD, o número de integrandos é igual a 810 multiplicado ao número de volumes de controle de sua malha CFD. Importante realçar que a baixa dimensionalidade das integrais presentes na PBE torna mais vantajoso o PIL, uma vez que o custo computacional do cálculo das integrais não é alto.

6.5 Solução do Sistema Linear

Como visto nos capítulos anteriores, os métodos de quadratura em sua forma direta geram sistemas lineares $2N$ por $2N$, onde N é o número de pontos de quadratura. Tradicionalmente, esses sistemas lineares são solucionados pelo método direto de decomposição LU. No entanto, foram identificadas algumas vantagens se o método de decomposição em valores singulares [151] for utilizado. Portanto, nessa seção uma descrição do método, e a justificativa pela qual a mesma pode melhorar a robustez da solução do sistem linear gerado para o DQMoM e o D^2uQMoGeM. Além disso, neste trabalho, foi implementado o paralelismo na solução dos sistemas lineares usando métodos diretos em GPU.

6.5.1 Decomposição em valores singulares em GPU

Fundamentalmente, o método de decomposição em valores singulares (SVD) é uma técnica de fatoração de uma matriz real ou complexa, que fornece desde o cálculo de uma pseudo-inversa ao ajuste de dados por mínimos quadrados. O SVD é a decomposição de uma matriz A com dimensionalidade $M \times N$ em duas matrizes reais ou complexas U, $M \times N$, e V, $N \times N$, que são ortogonais por coluna, isto é, $U^T U = VV^T = I$, e uma matriz retangular diagonal Σ, $M \times N$, com números reais não-negativos na diagonal, denominados valores singulares (Σ_i). Note que para essa decomposição não existe nenhuma restrição para o tamanho M e N. Neste sentido, não importa o tamanho da matriz, ou até mesmo se ela é singular, já que sempre existirá a decomposição definida na Equação 6.24 [152].

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{6.24}$$

De acordo com a Equação 6.24, se a matriz A for quadrada, as matrizes de decomposição $\mathbf{U}$, $\mathbf{V}$ e $\mathbf{\Sigma}$ também são quadradas, e portanto, a inversão da matriz A é trivialmente calculada pela Equação 6.25.

$$\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T. \tag{6.25}$$

Pode-se observar que na Equação 6.25, quando Σ_i é zero ou numericamente zero, a matriz $\mathbf{A}$ é singular. Então, a primeira vantagem dessa decomposição está na possibilidade de identificar possíveis problemas associados a singularidade da matriz.

O SVD em sua decomposição constrói explicitamente uma base ortogonal para o espaço nulo e o *range* da matriz A. Isso significa que sendo A uma matriz singular, a solução de um sistema linear $\mathbf{Ax} = \mathbf{0}$ é dada imediatamente pelo SVD, uma vez que qualquer coluna V correspondente a $\Sigma_i = 0$ é solução do sistema $\mathbf{Ax} = \mathbf{0}$ [151].

Nos casos onde $\mathbf{Ax} = \mathbf{b}$, sendo $\mathbf{b} \neq 0$, em que $\mathbf{b}$ está no range de A, sabemos que pode existir mais de uma solução para o sistema linear da matriz singular $\mathbf{A}$. Como se deseja apenas uma solução, geralmente, a solução escolhida é a que tem o menor valor do comprimento $\mid \mathbf{x} \mid^2$. Para tal, basta substituir o termo $1/\Sigma_i$ por zero quando $\Sigma_i = 0$ na Equação 6.26. Assim, a solução de $\mathbf{Ax} = \mathbf{b}$ é dada por,

$$\mathbf{x} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}. \tag{6.26}$$

Nos casos onde $\mathbf{b}$ não está no range de uma matriz singular, não existe solução à princípio. No entanto, a Equação 6.27 obtém o vetor $\mathbf{x}$ que minimiza,

$$r \equiv \| \mathbf{Ax} - \mathbf{b} \|, \tag{6.27}$$

que é o resíduo do sistema linear.

Como já mencionado, quando a matriz $\mathbf{A}$ não é singular, não seria necessário o uso do SVD, uma vez que métodos como o LU e eliminação Gaussiana seriam satisfatórios. No entanto, nos casos em que os valores singulares são numericamente próximos de zero, o sistema linear é mal-condicionado. Desta forma, a vantagem em usar o SVD é obtida com a eliminação de uma linha ao especificar como zero os Σ_is, cujo os valores são considerados numericamente zero. PRESS *et al.* [152] recomenda que Σ_i deva ser eliminado quando $\mid \Sigma_i/\Sigma_{max} \mid < 10^{-8}$, onde Σ_{max} é o maior valor singular de $\mathbf{\Sigma}$.

O sistema linear gerado no DQMoM e D^{2}uQMoGeM pode ser extremamente mal-condicionado [111] (Apêndice B). Nesse sentido, usar o SVD pode ser uma vantagem no sentido de ganho de robustez. Além disso, para casos multivariados, o uso do SVD permite que o sistema linear seja retangular, i.e, tenha mais equações que incógnitas, e mesmo assim, o SVD determina pela Equação 6.27 a melhor solução que minimiza o resíduo do sistema linear.

Pouco trabalho vem sendo desenvolvido no sentido de paralelização do algoritmo SVD em GPUs. Existem algumas iniciativas como a encontrada na biblioteca CULA (*CUDA linear algebra*) [153]. No entanto, não se trata de um código aberto e livre. Nessa linha, ainda se pode encontrar o trabalho de LAHABAR e NARAYANAN [154]. Neste trabalho, eles utilizaram a biblioteca CUBLAS (*CUDA Basic Linear Algebra Subroutines*) [155] para realizar as operações matriciais presentes na decomposição de $\mathbf{A}$. Obtiveram valores de aceleração entre 3 e 8. No entanto, só se observa aceleração com matrizes com tamanho de pelo menos 256×256. Note que essa abordagem não é útil para nossos problemas, uma vez que trabalhamos com sistema lineares de pequeno porte. Desta forma, o paralelismo proposto no presente trabalho é simples. Cada *thread* da GPU está associado a um volume da malha em CFD, e portanto, responsável por um sistema linear de pequeno porte. Assim,

os sistemas lineares são resolvidos paralelamente entres os *threads* de uma ou mais GPUs. A decomposição SVD aqui desenvolvida por *thread* se baseia no algoritmo disponíveis na biblioteca numérica *Numerical Recipes* [152]. Esse algoritmo é dividido em duas etapas: bidiagonalização e diagonalização. Na primeira, o algoritmo de *Householder* [156] é utilizado, e na segunda o algoritmo QR [151]. A partir dessas operações as matrizes $\mathbf{\Sigma}$, $\mathbf{V}$ e $\mathbf{U}$ são obtidas, e por consequência a solução de um sistema linear.

6.6 Formulação do D^{2}uQMoGeM multi-fluido no OpenFOAM®

Nesta seção, a formulação do método D^{2}uQMoGeM-FC para solução da PBE acoplada com a modelagem multi-fluido é esmiuçada. Aqui, não é apenas discutido o equacionamento *Fully-Conservative* do D^{2}uQMoGeM, mas também o seu algoritmo de acoplamento com OpenFOAM® usando um *plugin CUDA*. Além disso, também é apresentado como o paralelismo entre processos MPI pode ser utilizado na manipulação de múltiplas GPUs.

6.6.1 D^{2}uQMoGeM-FC (Fully-Conservative)

O acoplamento PB-CFD com os métodos tipo DQBMMs são essencialmente semelhantes aos com os QBMMs. Novamente, cada ponto de quadratura de Gauss-Christoffel é atribuído a uma fase dispersa α. Quando suas abscissas representam o volume das partículas, as abscissas ponderadas, ζ_α, são equivalentes a fração de fase em um escoamento incompressível. Além disso, se as partículas forem consideradas esféricas, a mesma pode ser convertida em diâmetro de partícula, e portanto, usada para calcular as forças interfaciais entre a fase contínua e as fases dispersas. A Figura 6.4 e os Algoritmos 2 e 3 resumem o acoplamento DQBMM-PB-CFD para o DuQMoGeM e o Direct DuQMoGeM.

Note que a principal diferença entre as metodologias com os DQBMMs e os QBMMs está no controle de acurácia pela cubatura adaptativa (AC). Infelizmente, a AC aumenta substancialmente o custo dos DQBMMs [30], especialmente, quando está acoplado a simulações CFD. Então, visando atenuar esse problema, o algoritmo de AC descrito na seção 6.2 é utilizado para computar os termos A_{jik} e L_{ji}. Ademais, os DQBMMs aqui utilizados são implementados em sua forma *Fully-Conservative* com intuito de evitar problemas na preservação dos momentos [9].

Algoritmo 2: Acoplamento DQBMM-PB-CFD

1 - Solução do Método D^2*uQMoGeM-FC;*
2 - Cálculo das forças associadas a troca de quantidade de movimento da fase contínua, Eq.4.15, e dispersa, Eq. 4.13;
3 - Discretização dos termos Υ_{α}^{C} *e* Υ_{0}^{C}*.*
4 - Cálculo dos Operadores Matriciais $(\mathbf{C}_{\alpha})_{D}$ *e* $(\mathbf{C}_{\alpha})_{H}$*.*
enquanto *LOOP de correção do PISO* ***faça***
  Solução da Pressão pela Eq. 4.26;
  Correção do Fluxo por meio das Eqs. 4.23, 4.24 e 4.25;
  Novo cálculo para campo de velocidades, Eq.4.18 e Eq.4.19;
fim do enquanto

Algoritmo 3: Algoritmo de Solução do D^2uQMoGeM-FC

para todas(os) *Células CFD* ***faça***
  Transforma ω_{α} *e* m_{α} *do OpenFOAM® para tipos acessíveis na GPU;*
fim do percorre
Copia pesos e abscissas da CPU para GPU;
Computa os valores de A_{jik} *e* L_{ji} *na GPU por cubatura adaptativa;*
Computa os valores $\overline{F}_{j}^{N}$ *e* $\overline{R}_{j}^{(N)}$ *na CPU;*
para todas(os) *Células CFD* ***faça***
  Transforma $\overline{F}_{j}^{N}$ *e* $\overline{R}_{j}^{(N)}$ *do OpenFOAM® para tipos acessíveis na GPU;*
fim do percorre
Copia $\overline{F}_{j}^{N}$ *e* $\overline{R}_{j}^{(N)}$ *da CPU para GPU;*
Construir os sistemas lineares, Eq. 6.20, na GPU;
Solução do sistemas lineares, Eq. 6.20, na GPU;
Copiar A_{α} *e* B_{α} *da GPU para CPU;*
Resolver as equações de transporte para os pesos e as abscissas na CPU, Eq. 6.19;

Determinar $r_{\alpha} = \varsigma_{\alpha}$ *e* d_{α}*.*

6.6.2 Implementação MPI-CUDA no OpenFOAM®

Atualmente, existem algumas implementações bem sucedidas que visam o acoplamento entre o OpenFOAM® e os códigos CUDA [157–159]. O primeiro projeto surgiu com empresa Vratis SpeedIT [157], que criou um *plugin* que combina a biblioteca de solução de sistemas lineares, também desenvolvida pela Vratis SpeedIT, com a biblioteca de solução de sistemas lineares do OpenFOAM® por meio de uma classe *container*. Basicamente, esta classe *container* converte os tipos definidos no OpenFOAM® para tipos acessíveis dentro da GPU. Assim, o sistema linear de equações gerado no OpenFOAM® é transferido da CPU para GPU visando sua solução de forma paralela.

Seguindo essa mesma ideia, SYMSCAPE [158] desenvolveu um *plugin* que acopla os métodos de solução de sistemas lineares PCG e PBiCG disponíveis na biblioteca CUSP com o OpenFOAM®. Neste caso, para transferir os dados foi utilizada a

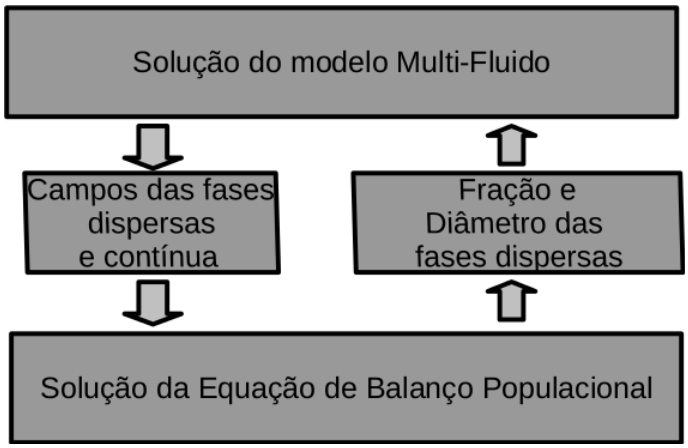


Figura 6.4: Acoplamento DQBMM-PB-CFD para o DuQMoGeM e o Direct DuQMoGeM.

biblioteca *Thrust* (biblioteca em CUDA que mimetiza a biblioteca em C++ STL [160]) com o intuito de transferir os dados da CPU para a GPU dentro de uma classe *container*.

Posteriormente, COMBEST e DAY [159] desenvolveram a biblioteca **Cufflink**. À princípio, essa biblioteca seria capaz de acoplar qualquer método numérico desenvolvido em CUDA com o OpenFOAM®. No entanto, a mesma só realiza o acoplamento entre a biblioteca CUSP, também por meio de uma classe *container* usando a biblioteca *Thrust*, e o OpenFOAM®. Diferente dos outros *plugins*, este último está disponível como livre e aberto. Além disso, este código aberto é capaz de usufruir multi-GPUs. Para tal, o mesmo utiliza o próprio paralelismo tipo *course-grained* do OpenFOAM®. Neste caso, a malha CFD é, essencialmente, dividida entre os nós de computação, e cada processo da CPU é atribuído a uma ou mais GPUs.

Baseado no *plugin* desenvolvido por COMBEST e DAY [159], o procedimento de acoplamento CUDA-OpenFOAM® aqui proposto é apresentado no diagrama 6.5 para o método D^2uQMoGeM-FC. Nesta implementação, a biblioteca *Thrust* também foi utilizada para transferir dados da CPU para GPU em uma classe *container*. Primeiro, esta classe transfere os pesos e abscissas para o cálculo dos termos de fontes de quebra e agregação ($\overline{R}_j^{(N)}$). Em uma próxima etapa, os termos $\overline{R}_j^{(N)}$ e $\overline{F}_j^N$, os pesos e as abscissas são novamente transferidos para GPU visando construir os sistemas lineares do Direct DuQMoGeM em todo volume de controle. Finalmente, os sistemas lineares gerados são resolvidos pelo *solver* SVD-CUDA aqui implementado.

Como mencionado anteriormente, o paralelismo tipo *course-grained* do OpenFOAM® pode ser usado para manipular múltiplas GPUs. Na decomposição do domínio entre CPUs, a malha computacional é dividida em subdomínios. Cada subdomínio está associado com um nó de CPU, que pode ser associado com

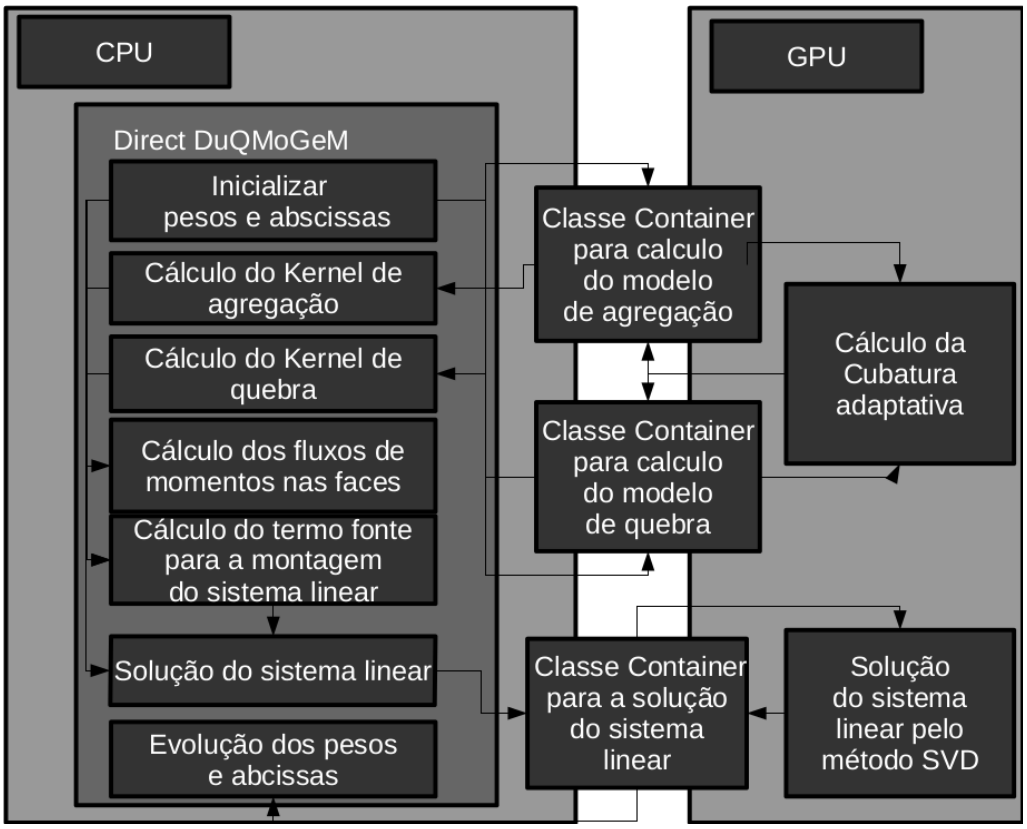


Figura 6.5: Fluxograma paralelo do Direct DuQMoGeM.

uma ou mais GPUs. Por exemplo, o nó de computação com índice 1 pode ser atribuído a GPU com índice 1. Neste caso, o sistema linear e as integrais do Direct DuQMoGeM presentes no subdomínio do nó de computação 1 são resolvidos pela GPU de índice 1. Nos casos, onde existem mais CPUs que GPUs, as tarefas são divididas entre as GPUs disponíveis para computação. Por exemplo, se existem X nós de CPU e W GPUs, onde $X > W$, os nós de 1 até X/W, são atribuídos a GPU 1, os nós de $X/W + 1$ até $2X/W$ são atribuídos a GPU 2, e assim por diante.

Capítulo 7

Resultados e Discussões

Neste capítulo, são apresentados e discutidos os resultados relativos ao desenvolvimento deste trabalho. Na primeira seção, será vista a avaliação comparativa em termos de acurácia e tempo computacional entre os métodos DQMoM e D^{2}uQMoGeM. Na seção seguinte, mostra-se uma avaliação do algoritmo de cubatura adaptativa paralelo desenvolvido com duas formas de paralelismo. A penúltima seção descreve a análise do algoritmo SVD paralelo em CUDA. Já na última seção, os resultados das simulações multifásicas polidispersas com o D^{2}uQMoGeM-FC no OpenFOAM® são apresentados e comparados com os obtidos no DQMoM-FC, incluindo a avaliação da aceleração de código para as simulações com o método D^{2}uQMoGeM-FC. Vale destacar que todos os resultados presentes nesse capítulo foram gerados em dupla precisão e podem ser encontrados com mais detalhados nos manuscritos anexados a este documento.

7.1 Comparação entre DQMoM e D^{2}uQMoGeM

Para essa análise, os casos testes foram implementados nas linguagens de programação C++ e FORTRAN. O código fonte foi compilado com os compiladores gfortran e g++ usando a *flag* de otimização -03 e variáveis em dupla precisão. Os testes nesta seção foram realizados com o processador Intel(R) Core(TM) i5-2400 3.1GHz com o sistema operacional FEDORA 15 Linux. Note que em nenhum dos testes a computação paralela em GPUs foi utilizada. Como o método aqui proposto é baseado no método DQMoM, é necessário um algoritmo para inicialização que converte os momentos iniciais da FDN em pesos e abscissas. Então, para obtê-los foi utilizado o pacote ORTHOPOL desenvolvido por GAUTSCHI [161]. Esse pacote contém o algoritmo de Chebyshev modificado. Além disso, há algoritmos que fornecem os termos de recorrência das séries de polinômios ortogonais clássicos. Em todos os casos simulados a integração temporal foi realizada com o integrador DASSLC versão 3.2 desenvolvido por SECCHI [162], bastante reconhecido no meio científico pela sua

eficiência e precisão na solução de EADs implícitas de índice até 1. Visando uma solução virtualmente livre de erros de integração temporal, a tolerância absoluta e relativa foi especificada a 5×10^{-13}. Para determinar o tempo de computação, a rotina intrínseca do g++ *clock* foi usado para obter o tempo CPU em segundos com acurácia de $0,01\ s$. Para maior precisão na determinação do tempo computacional, o mesmo é reportado como a média dos tempos computacionais de 10 simulações sucessivas. Para avaliar a acurácia dos métodos o erro relativo definido pela Equação 7.1 foi usado.

$$\epsilon_k = \frac{|\mu_k - \mu_k^a|}{\mu_k^a}, \tag{7.1}$$

onde μ_k e μ_k^a são os valores numéricos e analíticos dos momentos padrões, respectivamente.

Como supracitado, o método proposto foi investigado frente aos aspectos de acurácia e custo computacional. SANTOS *et al.* [138] (manuscrito no Anexo C) testaram 13 casos homogêneos com variáveis internas aditivas e soluções analíticas conhecidas. Dentre esses casos estão os casos de quebra e agregação simultâneas (Casos 1, 2 e 3), de quebra pura (Casos 4 e 5), de agregação pura (Casos 6 e 7), de crescimento puro (Casos 8, 9 e 10), de quebra e nucleação simultâneos e de quebra, agregação, crescimento e nucleação simultâneos. Embora, vários casos testes tenham sido avaliados, apenas seis casos são ilustrados nesta seção. Para maiores detalhes, o leitor deve consultar o Anexo C, onde a publicação associada a esse trabalho é apresentada. Vale lembrar que para os casos onde o domínio da variável interna são semi-infinito e finito, as bases polinomiais de Laguerre e Legendre foram usadas, respectivamente. Embora qualquer família de base polinomiais ortogonais possa ser usada, uma análise rigorosa das diferentes famílias de bases foge do escopo desse trabalho.

7.1.1 Quebra e Agregação Simultâneas

O problema simples de quebra e agregação simultâneas, com condição inicial dada pela Equação 7.2 e *Kernels* pelas Equações 7.3, 7.4 e 7.5, foi proposto por MCCOY e MADRAS [46].

$$f(x,0) = e^{-x}, \tag{7.2}$$

em que, $x \in [0, \infty)$, e

$$b(x) = Cx, \quad C = constante, \tag{7.3}$$

$$a(\tilde{x}, x) = K, \qquad K = 1, \tag{7.4}$$

e

$$P(x|x') = \frac{H(x'-x)}{x'}. \tag{7.5}$$

Neste trabalho, MCCOY e MADRAS [46] propuseram a sua solução analítica abaixo:

$$f(t,x) = \Phi^2(t)e^{-x\Phi(t)} \tag{7.6}$$

onde,

$$\Phi(t) = \Phi(\infty)\left[\frac{1+\Phi(\infty)\tanh(\Phi(\infty)\frac{t}{2})}{\Phi(\infty)+\tanh(\Phi(\infty)\frac{t}{2})}\right], \tag{7.7}$$

sendo que, $\Phi(\infty) = \sqrt{\frac{2C}{K}}\frac{\sqrt{\mu_1(0)}}{\mu_0(0)}$ é o valor $\Phi(t)$ no seu estado estacionário. E assim, os seus momentos regulares são dados por:

$$\mu_k^a(t) = \left[\frac{\Phi(\infty)+\tanh(\Phi(\infty)\frac{t}{2})}{\Phi(\infty)[1+\Phi(\infty)]}\right]^{k-1}\Gamma(1+k) \tag{7.8}$$

tal que, para $t = 0$, essa equação é simplificada para,

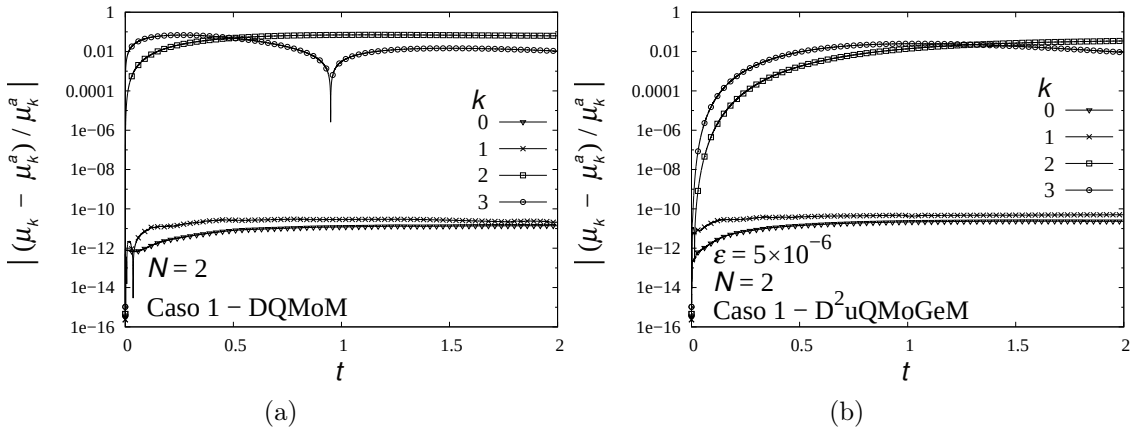
$$\mu_k^a(0) = \Gamma(1+k). \tag{7.9}$$

O parâmetro $\Phi(\infty)$ traduz informações importantes do estado estacionário. Se $\Phi(\infty) > 1$, o problema é de quebra dominante enquanto que se $\Phi(\infty) < 1$ é de agregação dominante. Sendo assim para 3 diferentes efeitos, 3 casos testes foram testados:

- Caso 1: $\Phi(\infty) = 0,5$,
- Caso 2: $\Phi(\infty) = 1,0$,
- Caso 3: $\Phi(\infty) = 2,0$.

Nas simulações os erros absoluto e relativo para a cubatura adaptativa foram de 5×10^{-6} para os Casos 1 e 3, e de 5×10^{-8} para o caso 2. A base polinomial de Laguerre com $w(x) = \exp(-x)$ foi utilizada na aproximação da FDN e na geração dos momentos generalizados, já que os problemas analisados estão em domínio semi-infinito.

Os erros relativos dos momentos para os métodos DQMoM e D^{2}uQMoGeM com $N = 2$ nos Casos 1 e 3 são mostrados nas Figuras 7.1 e 7.2. A Figura 7.1 mostra que D^{2}uQMoGeM é superior em termos de acurácia, especialmente, no início da simulação, contudo, sua diferença acaba não sendo tão significativa ao final da simulação. Na Figura 7.2 pode se observar que o D^{2}uQMoGeM é superior em pelo menos uma ordem de magnitude. Essa diferença se justifica, pois a fonte de erro é maior no



(a) (b)

Figura 7.1: Erro relativo dos momentos padrões para o caso 1 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.

termo de quebra, já que o modelo é linear. Assim, se a influência da quebra é maior, o erro também é mais pronunciado.

A Figura 7.3 ilustra a solução para os momentos com $N = 2$ com ambos os métodos para o caso invariante 2. Essa figura mostra que o momento de terceira ordem é fortemente afetado pelo erro de quadratura do DQMoM, fato que não ocorre no D^{2}uQMoGeM. Neste caso, os métodos diferem significadamente em termos de acurácia como mostrada na Figura 7.4. Assim pode-se observar que os erros dos momentos de terceira e quarta ordem são substancialmente afetados no DQMoM, sendo que seus valores chegam a cerca de 5%. Relativo aos tempos computacionais, para caso 1, 2 e 3 os tempos do DQMoM foram de $0,03$ segundos, e para o D^{2}uQMoGeM foram $0,07$, $0,10$ e $0,07$ segundos, respectivamente.

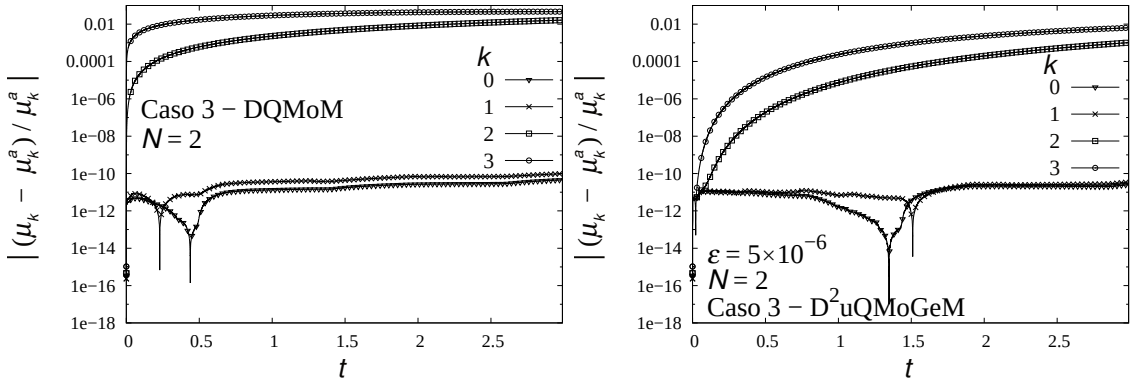
Por fim, deve-se apontar que o gargalo das simulações com o D^{2}uQMoGeM é o custo computational associado a integração numérica adaptativa. Esse problema é minimizado neste caso devido a invariabilidade com o tempo dos termos de quebra e agregação. Caso contrário, o tempo computacional muito elevado.

7.1.2 Quebra Pura

LAGE [29] definiram o problema de quebra pura binária em domínio de $x \in [0, 1]$ com:

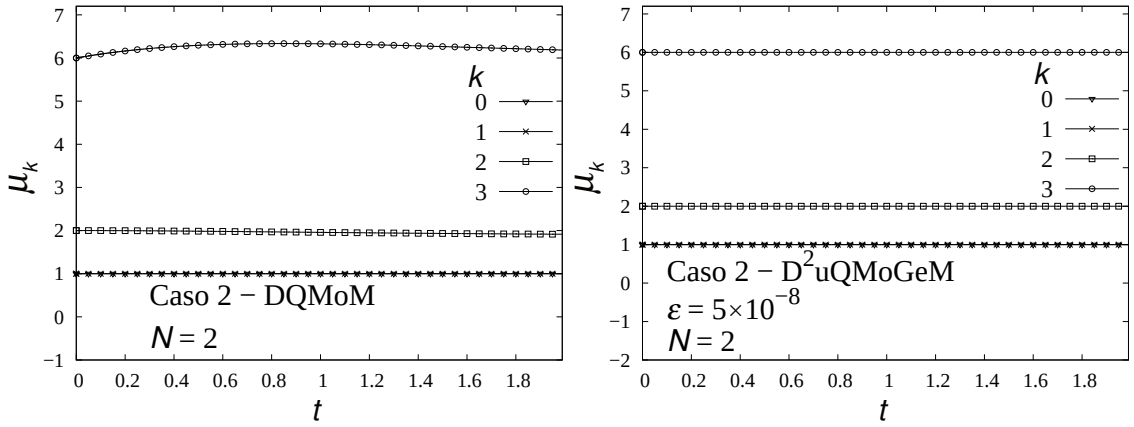
$$b(x) = x^p, \qquad P(x|x') = H(x' - x)/x' \tag{7.10}$$

Se p é um número inteiro, o frequência de quebra pertence ao espaço de polinômios ortogonais. Como é um problema fictício de modo a existir uma solução analítica,



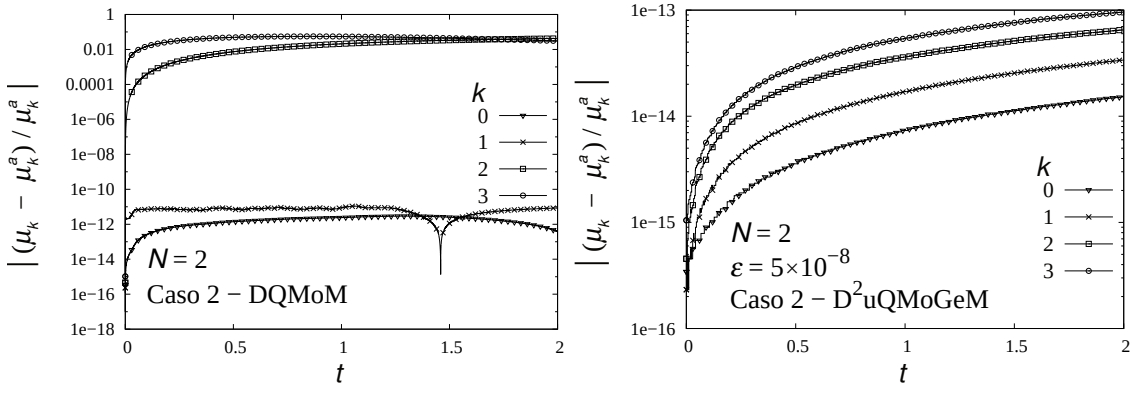
(a) (b)

Figura 7.2: Erro relativo dos momentos padrões para o caso 3 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.



(a) (b)

Figura 7.3: Evolução dos momentos padrões para o caso 2 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.



(a) (b)

Figura 7.4: Erro relativo dos momentos padrão para o caso 2 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.

um termo fonte adicional S foi definido para a validade da solução abaixo:

$$f(x,t) = 2 - e^{-t}, \tag{7.11}$$

cujo os momentos são:

$$\mu_k^a = \frac{2 - e^{-t}}{k+1}. \tag{7.12}$$

Para avaliar os métodos, os dois casos abaixo foram usados:

- caso 4: $p = 2$, $S(x,t) = 2x^2(2 - e^{-t}) - 2(1 - e^{-t})$,
- caso 5: $p = \frac{1}{3}$, $S(x,t) = 7e^{-t} - 12 + 7(2 - e^{-t})x^{1/3}$.

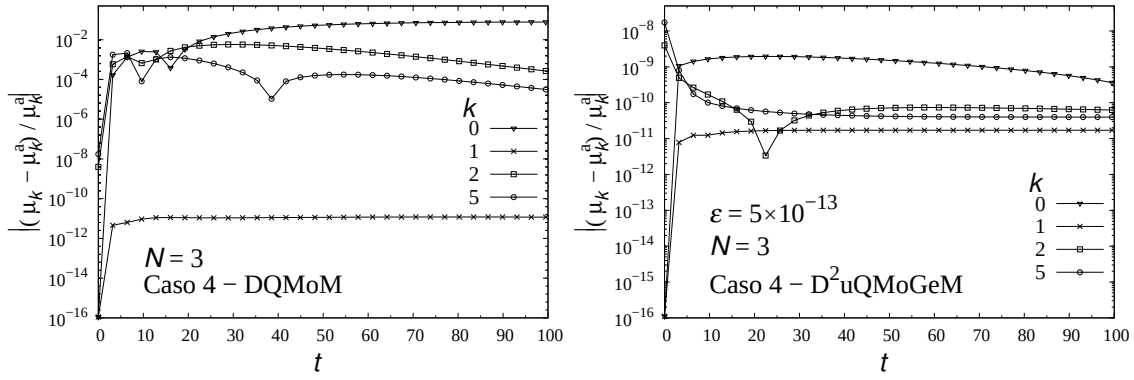
Para esse problema, as tolerâncias relativas e absolutas para integração foram de 5×10^{-13}, uma vez que o termo de integração L_{ji} é mais facilmente computado devido sua menor dimensionalidade. Outra diferença é o uso da base polinomial de Legendre *shifted*, já que neste caso se trata de um domínio entre $[0, 1]$.

A Figura 7.5 ilustra os erros relativos dos momentos para o caso 4 com ambos os métodos, claramente, ela mostra que o erro de solução do DQMoM é sempre superior ao do D^2uQMoGeM. Esse resultado já era esperado, uma vez que os erros associados às integrais são controlados [30].

A Figura 7.6 mostra a evolução de alguns momentos para solução de PBE com $N = 3$ para os dois métodos em estudo para o caso 4. Pode-se observar que μ_0 obtido pelo o DQMoM desvia do estado estacionário devido o erro de quadratura associado ao cálculo do termo de quebra. A Figura 7.7 mostra os erros relativos para os momentos, confirmando que o D^{2}uQMoGeM é substancialmente superior em termos de acurácia.

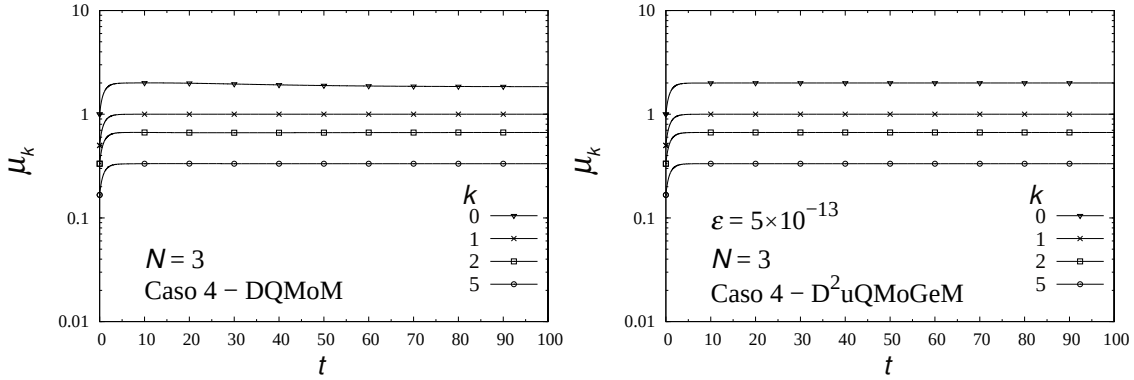
As Figuras 7.7 e 7.8 mostram, respectivamente, os momentos e seus erros relativos com $N = 3$ para o caso 5 com ambos os métodos. Elas mostram que o μ_0 obtido para o DQMoM é extremamente afetado pelos erros de quadratura, enquanto que no D^{2}uQMoGeM, os erros dos momentos são razoavelmente controlados até $t = 80$ segundos. Obviamente, existem outras fontes de erro, como por exemplo, o da aproximação funcional de f. Como já mencionado anteriormente, o trabalho aqui desenvolvido não está focado no estudo de qual base polinomial é mais adequada para aproximação de f.

Finalmente, relativo aos tempos computacionais, para caso 4 e 5 os tempos do DQMoM foram de $0,23$ e $0,32$ segundos, e para o D^{2}uQMoGeM foram de $0,10$ e $0,11$, respectivamente. Para esses casos, pode-se observar vantagem tanto em tempo computacional quanto acurácia. No entanto, vale lembrar que as cubaturas só são calculadas uma única vez, já que os termos integrais não tem dependência temporal. Para casos onde existe o efeito temporal, o D^2uQMoGeM será certamente mais lento.



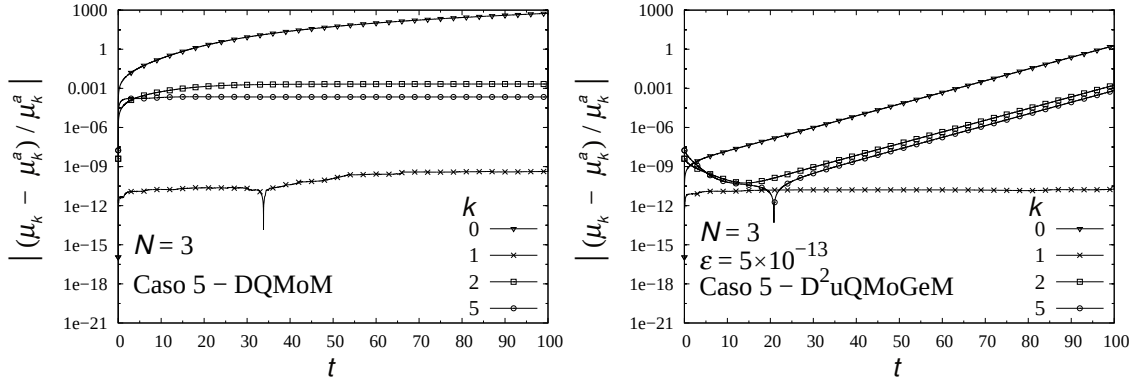
(a) (b)

Figura 7.5: Erro relativo dos momentos padrão para o caso 4 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.



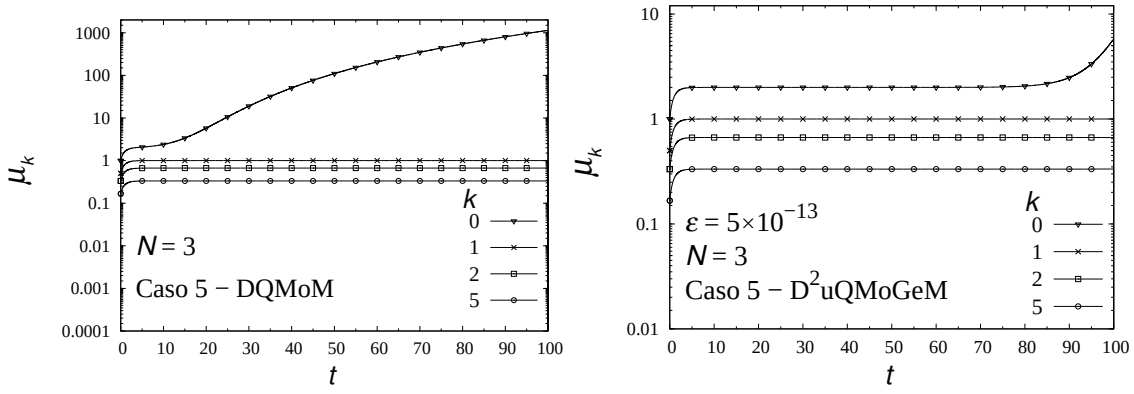
(a) (b)

Figura 7.6: Evolução dos momentos padrões para o caso 4 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.



(a) (b)

Figura 7.7: Erro relativo dos momentos padrão para o caso 5 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.



(a) (b)

Figura 7.8: Evolução dos momentos padrões para o caso 5 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.

7.1.3 Quebra, Agregação, Crescimento e Nucleação Simultâneos

Como análise final, um caso extremamente complexo porém fictício. Este incluí os fenômenos de quebra, agregação, crescimento e nucleação em um domínio semi-infinito, $x \in [0, \infty)$. As funções que definem o caso são: $b(x) = x^{1/2}$ (quebra), $r(x) = \delta(x-1)$ (nucleação), $a(x, x') = x+x'$ (agregação) e $g(x) = x^{1/2}$ (crescimento). A solução é dada pela Equação (7.11), cujos os momentos são dados pela Equação (7.12). Note que um termo fonte adicional é novamente imposto, sendo determinado pela Equação abaixo:

$$\begin{aligned} S(x,t) = {} & e^{-x-t} - (x+t)e^{-x-t} + (-2\sqrt{x} + \sqrt{\pi}\mathrm{erf}(\sqrt{\pi})e^{x} + 2t\sqrt{\pi}\mathrm{erf}(\sqrt{x})e^{x} \\ & - \sqrt{\pi}e^{x} - 2t\sqrt{\pi}e^{x})e^{-x-t} + \sqrt{\pi}(x+t)e^{-x-t}\frac{x^2}{12}(6t + 6t^2 + x^2)e^{-x-2t} \\ & + (x+t)e^{-x-t}(tx + x + t + 2)e^{-t} + \frac{(x+t)e^{-x-t}}{2\sqrt{x}} + \sqrt{x}e^{-x-t} \\ & - \sqrt{x}(x+t)e^{-x-t} - \delta(x-1) \end{aligned} \tag{7.13}$$

onde erf é a função erro.

A Figura 7.9 mostra que o erro de quadratura do DQMoM é novamente alto, levando a solução para momentos não conservados. Por outro lado, o D^{2}uQMoGeM fornece uma solução mais acurada. Porém ele tem tempo computacional muito superior ao DQMoM. Neste caso, os tempos computacionais do DQMoM e D^{2}uQMoGeM são de $0,04$ e $2,20$ segundos, respectivamente.

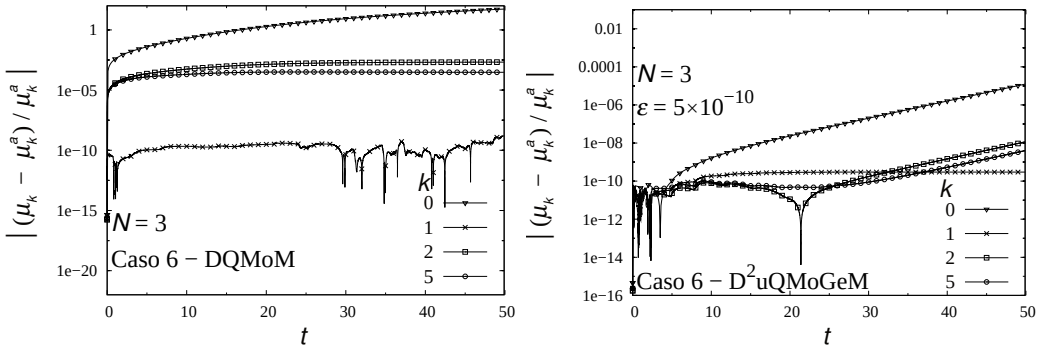


Figura 7.9: Erro relativo dos momentos padrões para o caso 4 ao longo do tempo: (a) DQMoM (b) Direct DuQMoGeM.

7.1.4 Conclusão

No presente trabalho, o método tipo Galerkin D^{2}uQMoGeM foi formulado. Vários casos homogêneos monovariados foram utilizados para teste. Todas as análises podem ser encontradas em mais detalhes no manuscrito SANTOS *et al.* [138] anexado. Em resumo, em todos os casos, o D^{2}uQMoGeM foi, pelo menos, equivalente em termos de acurácia se comparado com o DQMoM. Quando o erro de fechamento por quadratura no DQMoM é significante, o D^{2}uQMoGeM é superior devido ao seu controle de erro pela cubatura adaptativa.

De forma geral, o custo computacional das simulações com o D^{2}uQMoGeM foi ligeiramente superior ao do DQMoM. No entanto, o baixo custo computacional está associado a independência temporal dos *Kernels* de quebra e agregação, uma vez que as cubaturas são calculadas apenas no tempo $t = 0$. Portanto, se os *Kernels* da PBE forem dependentes com o tempo, o custo computacional será significadamente maior. Assim, o D^{2}uQMoGeM pode ser um ótimo candidato para o acoplamento PB-CFD, se seu custo computational for de alguma forma reduzido.

7.2 Avaliação da Cubatura Adaptativa Paralela

Nesta seção, os algoritmos paralelos de cubatura adaptativa são avaliados de acordo com sua aceleração. A métrica que representa a aceleração, *speed up*, é a razão entre o tempo de execução serial na versão em CPU e o tempo de execução da versão para GPU, definido a seguir:

$$S = \frac{T_s}{T_{gpu}}, \tag{7.14}$$

onde T_s é o tempo de execução serial e T_{gpu} da versão para GPUs.

O código fonte foi compilado com o compilador *g++*, versão 4.1.2, usando a *flag*

-03 de otimização. Para a compilação do código em CUDA, o compilador *nvcc* para *Toolkit 4.1* usando variáveis em dupla precisão foi usado. Os código para GPU e CPU foram executados em um Intel(R) Xeon(R) CPU X5570 2.93GHz e na placa gráfica da NVIDIA GTX480, respectivamente.

Para a nossa análise, primeiramente, foram realizadas análises de escalabilidade e aceleração para quatro integrais multi-dimensionais com soluções analíticas conhecidas. Posteriormente, o algoritmo de cubatura adaptativa paralelo é aplicado a 3 casos bivariados de PBE com solução analítica: dois problemas de agregação pura e um considerando os efeitos simultâneos de quebra e agregação.

7.2.1 Análise de Performance

GENZ e MALIK [149] identificaram alguns grupos de funções que são úteis na análise de rotinas de integração multidimensional, definindo famílias de integrais características como: oscilatória, função produto, função de canto e Gaussiana. Quatro diferentes integrandos pertencentes a essas famílias com 4 dimensões foram usados no presente trabalho com intuito de avaliar a aceleração a escalabilidade com os algoritmos paralelos PIFL e PIL. Abaixo as funções utilizadas neste análise:

1. Integrando oscilatório

$$f_1(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 cos(2\pi + \sum_{i=1}^{4} x_i)\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{7.15}$$

2. Integrando produto

$$f_2(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \frac{1}{\prod_{i=1}^{4} \left(1 + [x_i - 1]^2\right)}\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{7.16}$$

3. Integrando de canto

$$f_3(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \frac{1}{\left(1 + \sum_{i=1}^{4} x_i\right)^6}\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{7.17}$$

4. Integrando Gaussiano

$$f_4(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 e^{\sum_{i=1}^{4} (x_i - 1/2)^2}\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{7.18}$$

Para a análise de aceleração e escalabilidade com os modelos de famílias acima, duas estratégias foram usadas para mimetizar problemas com alto custo computacional. Primeiramente, visando analisar simultaneamente um número grande de

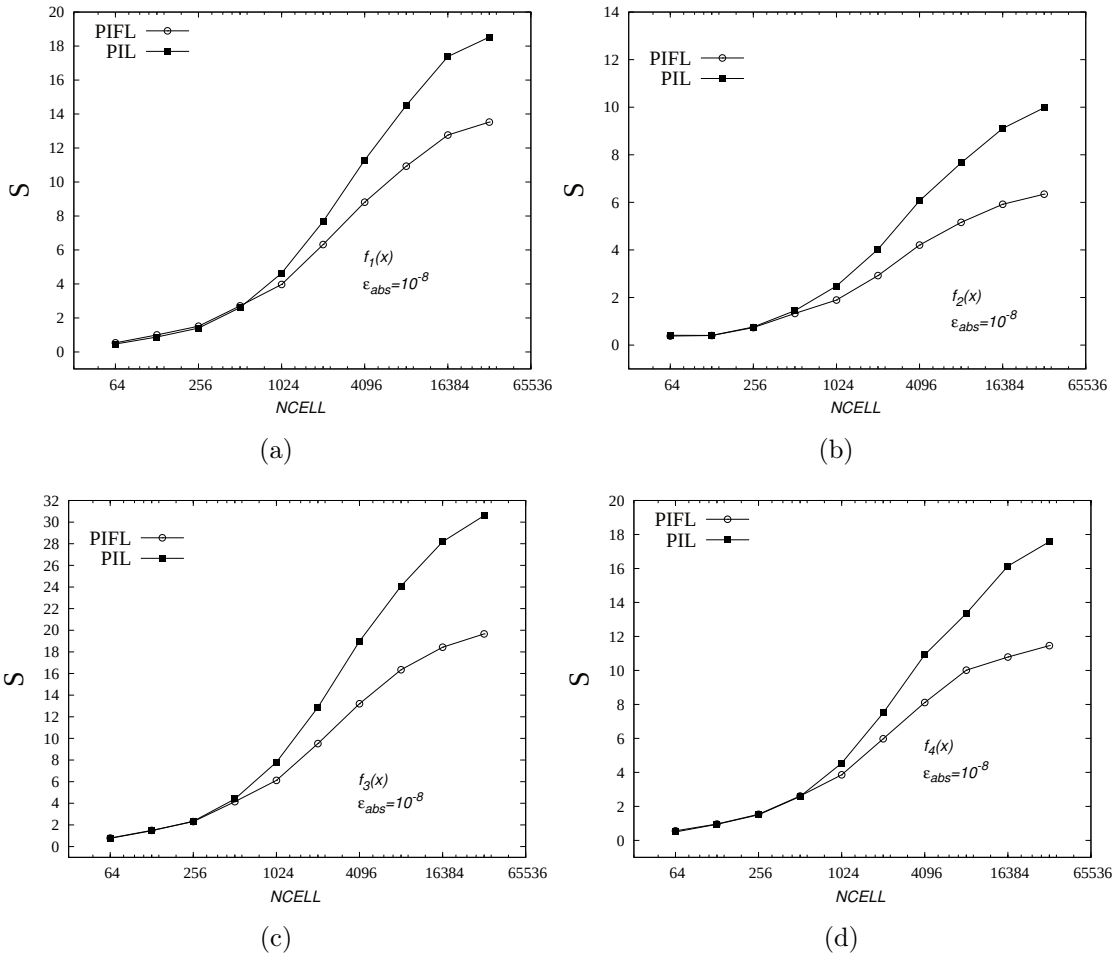


Figura 7.10: Análise de aceleração para $NCELLs$ integrandos com $\varepsilon_{abs} = \varepsilon_{rel} = 10^{-8}$: (a) $f_1(x)$, (b) $f_2(x)$, (c) $f_3(x)$ e (d) $f_4(x)$.

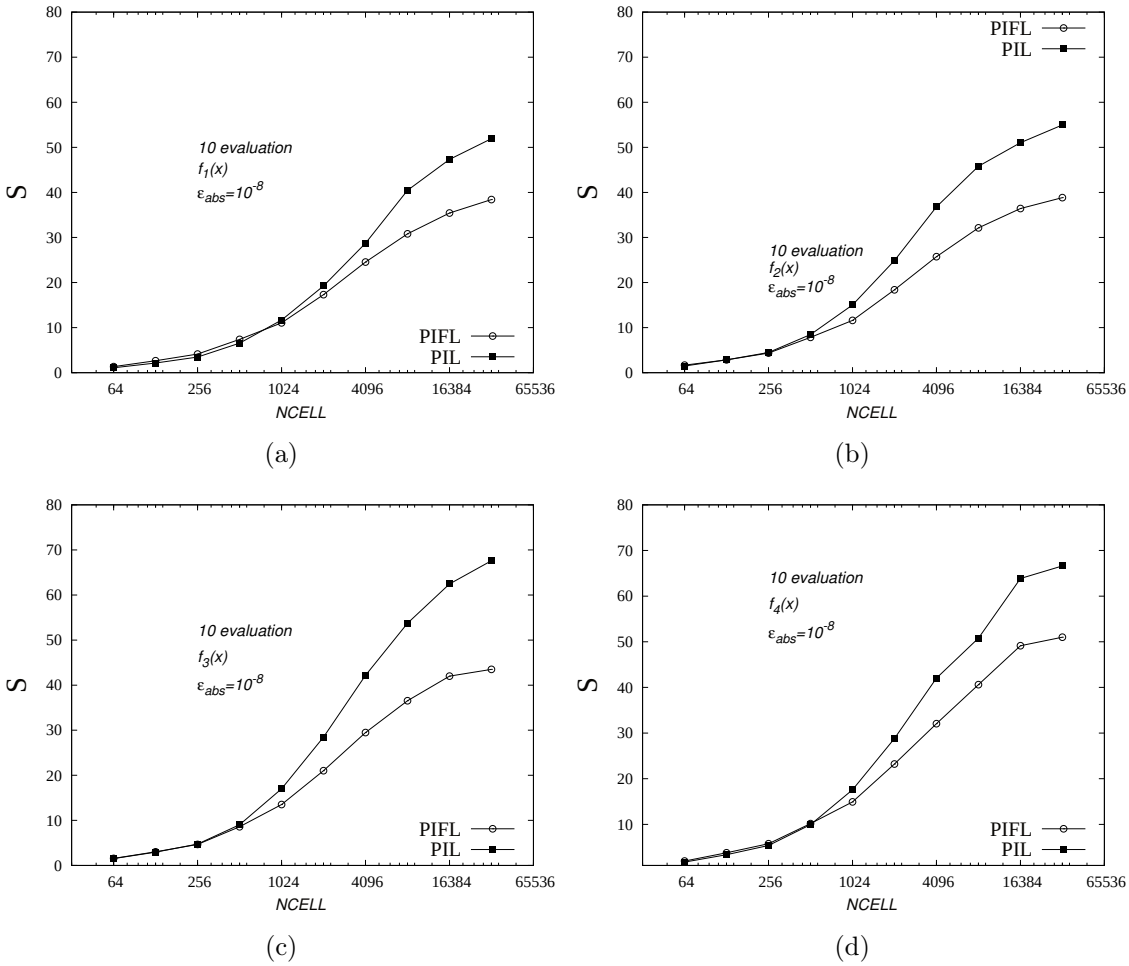


Figura 7.11: Análise de aceleração para $NCELLs$ integrandos, cada um avaliado 10 vezes em cada ponto de quadratura, com $\varepsilon_{abs} = \varepsilon_{rel} = 10^{-8}$: (a) $f_1(x)$, (b) $f_2(x)$, (c) $f_3(x)$ e (d) $f_4(x)$.

integrandos, cada integrando foi replicado para forma um vetor de integrandos com dimensão de $NCELL$. Na segunda estratégia, como as integrais multi-dimensionais de GENZ [163] foram criadas para ser simples, a carga de computação de cada integrando foi aumentada a partir da repetição do cálculo dos pontos de quadratura em 10 vezes. Em outras palavras, a estratégia mimetiza uma função 10 vezes mais complexa.

A Figura 7.10 compara as acelerações dos algoritmos PIFL e PIL para vários valores de $NCELL$, para os quatro integrandos com erros relativos e absolutos de 10^{-8}. A Figura 7.11 faz uma comparação similar porém os integrandos são 10 vezes mais "complexos". Para ambas as figuras, pode-se observar que a aceleração é menor que 1 para valores pequenos de $NCELL$, ou seja, houve uma desaceleração. Isso ocorre, porque a carga de trabalho é muito pequena frente a transferência de dados entre a CPU e a GPU. Porém, quando $NCELL$ aumenta, a aceleração aumenta, sendo que com características diferentes dependendo do algoritmo. Para

$NCELL \leq 512$, pode-se observar que os algoritmos PIL e PIFL são praticamente equivalentes, porém, quando $NCELL$ aumenta, é clara a superioridade do algoritmo PIL. Isso ocorre devido a sua maior independência entre os *threads*. Note também que com o aumento de $NCELL$, a aceleração tende a valores constantes. Isso ocorre porque quando o número máximo de *threads* é atingido o processo paralelo reduz sua escalabilidade como nas Figuras 7.10 e 7.11.

Esses resultados também demonstram que a aceleração é fortemente dependente da complexidade do integrando. Comparando a aceleração máxima entre os valores das Figuras 7.10 e 7.11, pode-se observar que o aumento da complexidade dos integrandos leva um aumento de 2-6 vezes no valor da aceleração. Além disso, a aceleração se torna menos dependente do tipo de integrando.

Visando avaliar a influência do número de sub-regiões geradas no processo de subdivisões de domínio na aceleração, o procedimento acima foi repetido, porém com tolerâncias relativas de absolutas de 10^{-5}. A partir dessa análise, pode-se concluir que a aceleração foi pouco afetada pela variação da tolerância. Esse resultado era esperado uma vez que cada região computa o mesmo número de integrandos. Em outras palavras, o aumento do número de regiões aumenta linearmente o custo computacional tanto no código serial quanto no paralelo, e assim, a aceleração não é afetada [31].

Importante destacar que as acelerações aqui atingidas para ambos os algoritmos são significativas, o que talvez viabilize o uso do DuQMoGeM e do D^{2}uQMoGeM no acoplamento PB-CFD. Contudo, o algoritmo PIL é mais atrativo para simulações PB-CFD por duas razões. Primeiramente, ele é mais rápido para um número grande de integrandos. Além disso, ele permite computar um maior número de integrandos simultaneamente devido à limitação da arquitetura Fermi de permitir apenas 65635 blocks [2]. Como o limite de *threads* é maior que o número total de blocks, o algoritmo PIL pode suportar um número maior de integrandos simultaneamente.

7.2.2 Aplicação

Para finalizar os testes, os algoritmos PIFL e PIL foram usados no DuQMoGeM para resolver uma PBE bivariada homogênea. O programa desenvolvido para solução da PBE foi escrito em CUDA/C++. A integração temporal foi realizada pela DASSLC [162] com tolerâncias relativa e absoluta de 10^{-10}.

O DuQMoGeM foi aplicado a PBE em domínio semi-finito, e por isso, a base polinomial de Laguerre foi utilizada. Nesse sentido, o domínio utilizado na cubatura adaptativa foi $x \in [0, 100]$ e $y \in [0, 100]$, cujo o intervalo é suficientemente grande para não afetar os valores das integrais.

Para análise de aceleração, o tempo total de simulação é o objeto de estudo.

Porém, o tempo de integração temporal (cerca de 1% do tempo total) é infinitamente inferior ao tempo de cálculo das integrais.

Visando analisar a aceleração nos testes acima, foi definida uma métrica que mede a complexidade dos *Kernels* da PBE. Esta métrica é dada pela equação abaixo:

$$E = \frac{T_c}{T_{ave}}, \tag{7.19}$$

onde T_c é o tempo total de computo dos *Kernels* da PBE e T_{ave} é o tempo médio das integrais de Genz usadas na seção anterior com o mesmo número de integrandos. Essa razão foi cerca de 4.7 para todos os casos estudados em SANTOS *et al.* [31] (documento no Anexo D). Isso significa que as acelerações resultantes da integração de Genz com 10 avaliações devem ser comparáveis com os obtidos na solução da PBE bivariada. Em SANTOS *et al.* [31] foram realizados 3 testes, e seus resultados foram semelhantes, e portanto, apenas um caso será descrito nessa seção. Para mais informações sobre os outros testes, o leitor deve consultar o Anexo D.

Problema de Agregação Pura

O problema de agregação pura bivariado proposto por GELBARD e SEINFELD [164] foi simulado com intuito de avaliar a aceleração dos algoritmos PIFL e PIL. Para esse caso, a frequência agregação é constante, $a(x, x', y, y') = 1.0$. Isso parece contraditório, já que para os métodos tipo DQMoM não haveria erro de quadratura. No entanto, à princípio, a idéia aqui é avaliar apenas a aceleração do DuQMoGeM em conjunto com os códigos paralelos e não compará-lo em termos de acurácia a outros métodos. Embora GELBARD e SEINFELD [164] tenham estudado vários casos com solução analítica, a solução analítica abaixo foi utilizada em nossa análise:

$$f(x, y, 0) = e^{-(x+y)}, \tag{7.20}$$

que leva a seguinte solução analítica:

$$f(x, y, t) = \frac{4}{(t+2)^2} e^{-(x+y)} I_0 \left(2\sqrt{\frac{t}{t+2}} xy \right) \tag{7.21}$$

onde I_0 é a função de Bessel do primeiro tipo de ordem zero.

Este caso simulação foi realizado com $N = 3$, o que produz um total de 729 integrandos. Suas tolerâncias absolutas e relativas foram de 5×10^{-5}, seguindo o trabalho de FAVERO e LAGE [30]. A Figura 7.12 mostra a evolução dos momentos e seus respectivos erros relativos. Neste gráfico, observa-se que os resultados estão em concordância com a solução analítica, em outras palavras, a cubatura adaptativa paralela reproduz os resultados de obtidos em FAVERO e LAGE [30]. Então, pode-

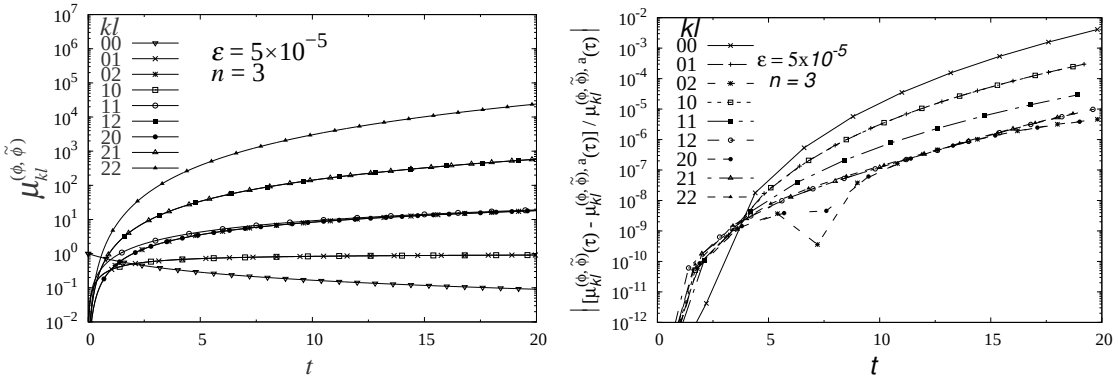


Figura 7.12: Solução numérica do problema de agregação pura com a cubatura adaptativa com $\varepsilon = \varepsilon_{abs} = \varepsilon_{rel} = 5 \times 10^{-5}$: evolução no tempo (a) dos momentos generalizados e (b) seus erros relativos.

se dizer que existe vantagem em usar o código paralelo, uma vez que a aceleração é de 12 para o algoritmo PIL e 11 para o algoritmo PIFL. Note que a diferença entre as acelerações não é significativa. Isso ocorre poque o número de integrandos não é significativamente alto para que haja influência como vimos na análise da Subseção 7.2.1.

7.2.3 Conclusão

Dois algoritmos de cubatura adaptativa foram propostos e implementados em GPU visando acelerar a solução da PBE pelo DuQMoGeM e Direct DuQMoGeM. Esses algoritmos estão em diferentes níveis, sendo que o primeiro (PIFL) é paralelizado ao nível do cálculo na fórmula de cubatura e o segundo ao nível do integrando (PIL).

A partir da análise de performance, pode-se observar que a aceleração dos dois métodos é pouco afetada pela tolerância. Mas, por outro lado, sua aceleração aumenta com a complexidade dos integrandos. Ambos os métodos obtiveram aceleração significativa, todavia, o algoritmo paralelo PIL é superior. Para os casos testados, a aceleração atingida foi cerca de 70 para integrais mais "complexas" e entre 6 e 30 para integrais computacionalmente mais simples. Entre as duas formas de paralelismo o algoritmo PIL é o recomendado para aplicações que envolvam CFD, uma vez que o mesmo tem maior aceleração e é mais adequado para um número maior de integrandos.

Os algoritmos propostos aqui foram aplicados a 3 problemas envolvendo balanço populacional com o DuQMoGeM. Todos os casos foram de acordo com as suas respectivas soluções analíticas e as suas acelerações foram entre 11 e 15, sendo todos os casos equivalentes em termos de aceleração, e acordo com SANTOS *et al.* [31] (ver anexo D). Finalmente, deve-se destacar que o algoritmo de cubatura adapta-

tiva desenvolvido neste trabalho pode ser usado para acelerar outras aplicações que envolvam vetores de integrais multidimensionais, não se limitando apenas a solução da equação de balanço populacional.

7.3 Avaliação da Solução do Sistema Linear Paralelo

Normalmente, os *solvers* lineares paralelos são projetados para resolver sistemas lineares de grande porte. No entanto, para o caso específico onde pequenos sistemas lineares são resolvidos por célula, os algoritmos paralelos convencionais não são úteis. Nesse sentido, o método de solução SVD foi implementado em GPU, onde cada *thread* calcula um sistema linear de pequeno porte Assim, cada *thread* executa o algoritmo de bidiagonalização de HOUSEHOLDER [156], seguido do algoritmo de diagonalização QR iterativo [151], por fim, calculando a matriz inversa ou pseudo-inversa, para o cálculo da solução do sistema linear de cada thread para CPU.

Para avaliar essa implementação foram propostos testes bem simples. Primeiramente, para avaliar a sua aceleração, a construção dos sistemas lineares com tamanhos entre 4×4 e 20×20 é feita em cada *thread* da mesma forma que sua solução. Visando mimetizar uma malha computacional de CFD, o mesmo sistema linear foi replicado de acordo ao número de células CFD. Em outras palavras, no caso CUDA cada thread computa o mesmo sistema linear, e no caso serial o mesmo sistema linear e repetidamente computado num *loop* correspondente ao número de *threads* na GPU. Assim, avaliarmos a aceleração definida pela Equação 7.14.

A Figura 7.13 ilustra os resultados obtidos para aceleração do SVD implementado por *thread* em CUDA. Observa-se que a aceleração aumenta com o aumento do número de sistemas lineares. Isso ocorre devido a alta capacidade de escalabilidade das GPUs. Esse efeito é causado porque enquanto no código serial o tempo computacional aumenta linearmente com o aumento do número de sistemas lineares, no código paralelo o aumento é muito pequeno, uma vez que os sistemas são distribuídos entre os núcleos da GPU até a sua "saturação". As GPUs não executam todos os *threads* simultaneamente, e por isso, existe uma queda brusca na aceleração quando o número de *threads* atinge o número máximo que pode ser calculado simultaneamente na GPU. Verifica-se também que a aceleração não varia significativamente com o aumento do tamanho dos sistemas lineares de pequeno porte. Esse resultado pode ser atribuído a pequena complexidade dos sistemas lineares estudados. Como nas aplicações desejadas, o número de pontos de quadratura quase nunca é superior a 10, limitamos nossa análise para nossas possíveis aplicações.

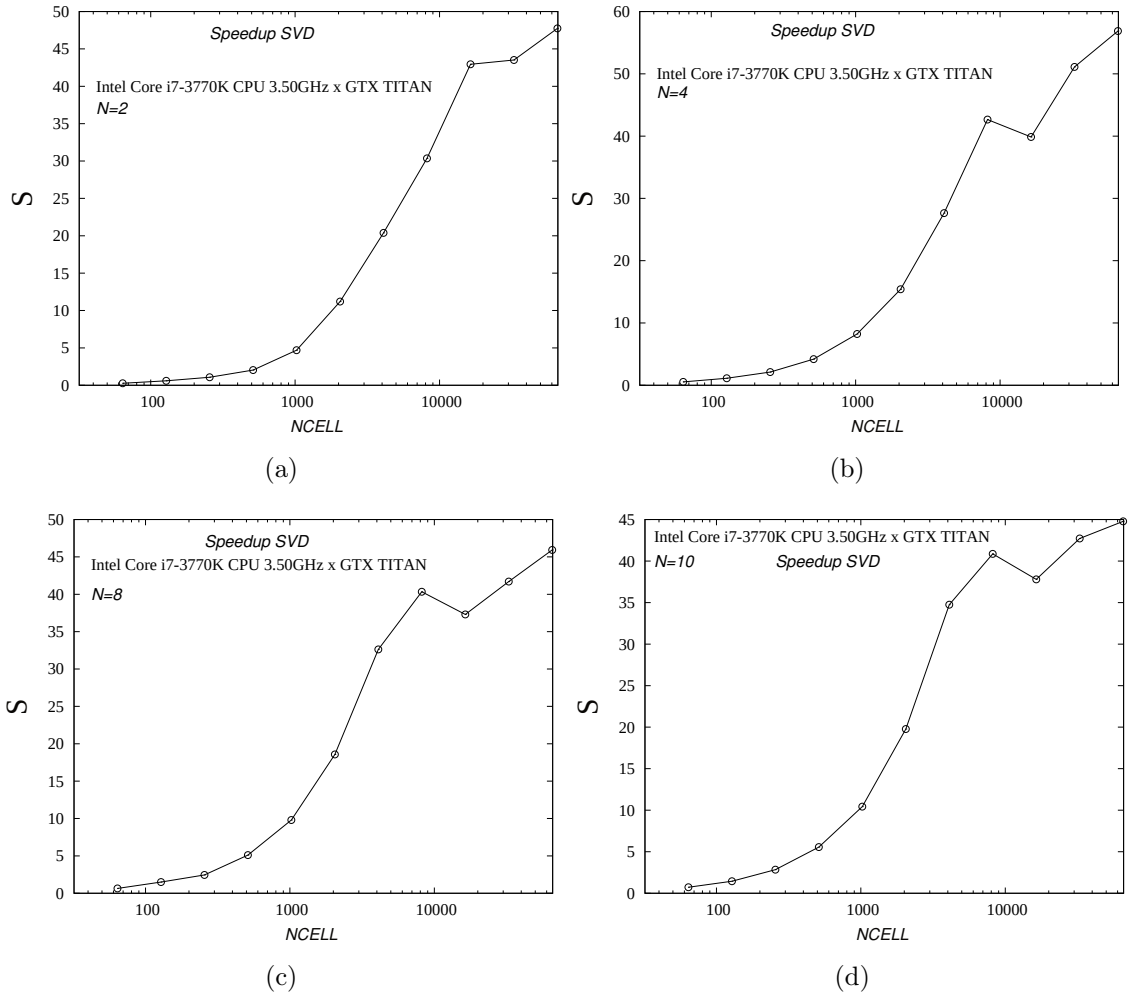


Figura 7.13: Análise de aceleração para $NCELLs$ sistemas lineares onde N é o número de pontos de quadratura no método direto: gerando matrizes de tamanho (a) 4×4, (b) 8×8, (c) 16×16 e (d) 20×20.

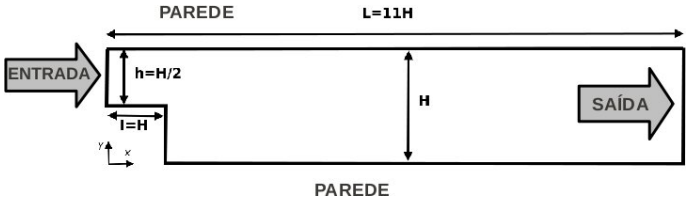


Figura 7.14: Geometria BFS com suas dimensões e condições de contorno.

7.3.1 Conclusão

O algoritmo paralelo SVD desenvolvido em CUDA usa a ideia principal das arquiteturas tipo SIMD ou SPMD (*Single Program Multiple Data*). Isso propiciou uma aceleração extremamente significativa do método SVD com o aumento do número de sistemas lineares a serem resolvidos. Assim, concluí-se que se a implementação for aplicada ao método D^{2}uQMoGeM-FC em uma malha de CFD, o tempo de computação associado ao sistema linear será extremamente menor. Vale ressaltar que o custo computacional de solução do sistema linear é muito menor que o custo da cubatura adaptativa. Portanto, a influência da aceleração adquirida na implementação SVD, não é significante se comparado ao tempo total de simulação. Note que essa implementação pode ser utilizada para outras aplicações, como por exemplo, solução de sistemas lineares para o fluxo de massa de Maxwell–Stefan [165].

7.4 Simulação de emulsão usando o D^{2}uQMoGeM no OpenFOAM®

A verificação do código DQBMM-PB-CFD foi realizada na geometria *Backward Facing Step* (BFS) em duas dimensões, devido sua simplicidade e zonas de recirculação bem definidas, onde os efeitos de interação entre as partículas são dominantes. A geometria parametrizada está de acordo com a Figura 7.14, onde $L = 11H$, $l = H$ e $h = H/2$, com $H = 0,01$ metros.

Para todas as simulações teste, uma mistura de duas fases, água (fase dispersa) e óleo, é alimentada na entrada (*inlet*) sob as condições e propriedades descritas na Tabela 7.1. Em relação aos métodos numéricos utilizados nas simulações, os termos integrais dos DQBMMs são computados pela AC paralela em GPU [31], onde cada tolerância varia de caso para caso. Nos casos onde o sistema linear de equações precisa ser resolvido por nó da malha de CFD, o código SVD-CUDA desenvolvido foi utilizado. Na discretização temporal, o método de Euler implícito foi usado com um passo de tempo fixo de 10^{-4} até o tempo total de $0,3$ segundos. O esquema *upwind* é utilizado para o termo advectivo com intuito de evitar momentos corrompidos [127, 166].

Tabela 7.1: Condições de contorno e propriedades das fases.

Propriedades	Óleo	Água
ρ (kg/m^3)	900	1000
ν (m^2/s)	10^{-5}	10^{-6}
Condições de contorno		
Velocidade de entrada (m/s)	1,0	
Fração de fase de entrada	0,926216	0,073784
Pressão de saída (N/m^2)	0,0	
Parede	Não deslizamento	Não deslizamento
Condições iniciais		
Fração de fase	0,926	0,074

As simulações testes são divididas em 3 partes. Primeiramente, o código DQBMM-PB-CFD é usado para simular 3 casos com os *Kernels* de quebra e agregação não realísticos propostos por LAGE [29] e MCCOY e MADRAS [46]. LAGE [29] e SANTOS *et al.* [138] já comparam os DQBMMs e QBMMs em um simulação homogênea para esses casos. Portanto, espera-se que os DQBMMs sejam superiores também em simulações não homogêneas. Posterioramente, a aceleração e a escalabilidade foram analisadas aumentando tanto o número de integrandos calculados pela AC paralela, quanto o número de células da malha CFD. Nesta etapa, todas as análises de aceleração foram executadas em uma CPU Intel Xeon I7-3770K 3.50Hz e nas placas gráficas NVIDIA GTX 680 e NVIDIA GTX TITAN para malhas 2D composta de quadriláteros com 1400 (malha 1), 2800 (malha 2) e 5600 (malha 3) nós. Finalmente, as análises MPI-CUDA foram realizadas com a malha mais fina, visando avaliar como a decomposição da malha de CFD pode aumentar o *speedup* da implementação.

Para avaliar a aceleração foram definidas duas métricas de aceleração. S_s é a aceleração descrita pela razão entre o tempo total de simulação da versão serial , T_s, e da versão CPU-GPU, T_g,

$$S_s = \frac{T_s}{T_g}, \tag{7.22}$$

Já para simulações MPI-CUDA uma outra definição foi necessária. S_p é a aceleração relativa ao número de processos em CPU e o número de GPUs utilizadas. S_p é definido pela razão entre o tempo total de execução de 1 CPU e 1 GPU sobre o tempo de Q processos em CPUs, usando M GPUs em uma versão de código multi CPU-GPU. A Equação 7.23 descreve S_p, onde T_g é o tempo total de uma simulação de uma única CPU-GPU, e T_{cg} é o tempo total usando M GPUs e Q processos em

CPUs.

$$S_p = \frac{T_g}{T_{cg}}. \tag{7.23}$$

Quando duas GPUs diferentes são utilizadas, T_g é uma média entre seus tempos. Por exemplo, se T_{g1} é o tempo computacional de uma única CPU-GPU usando a GTX TITAN e T_{g2} é o tempo computacional usando GTX 680, então o cálculo de tempo computacional médio de T_g é obtido da seguinte forma,

$$T_g = (T_{g1} + T_{g2})/2. \tag{7.24}$$

Para avaliar a relação do tempo computacional entre os métodos DQMoM-FC e D^{2}uQMoGeM-FC paralelo, a seguinte métrica foi definida:

$$\Upsilon = T_{du}/T_d \tag{7.25}$$

onde T_d e T_{du} são os tempos de computação da simulação PB-CFD usando os métodos DQMoM-FC e D^{2}uQMoGeM-FC paralelo, respectivamente.

7.4.1 Quebra e Agregação Simultâneas

Para os casos teste, os *Kernels* de MCCOY e MADRAS [46] foram utilizadas pra comparação entre o DQMoM, DuQMoGeM e Direct DuQMoGeM usando 4 momentos. A condição de entrada e inicial para distribuição é dada pela Equação 7.29, considerando $x \in [0, \infty)$, que representa neste caso uma variável adimensional.

A distribuição de tamanho inicial foi definida em termos do volume de gota, $v \in [0, \infty)$, e $F(v, 0)$, de acordo com momentos de ordem zero e um:

$$N_T(0) = \int_0^\infty F(v,0)\, dv, \qquad r_d(0) = \int_0^\infty vF(v,0)\, dv, \tag{7.26}$$

Definindo as seguintes variáveis adimensionais:

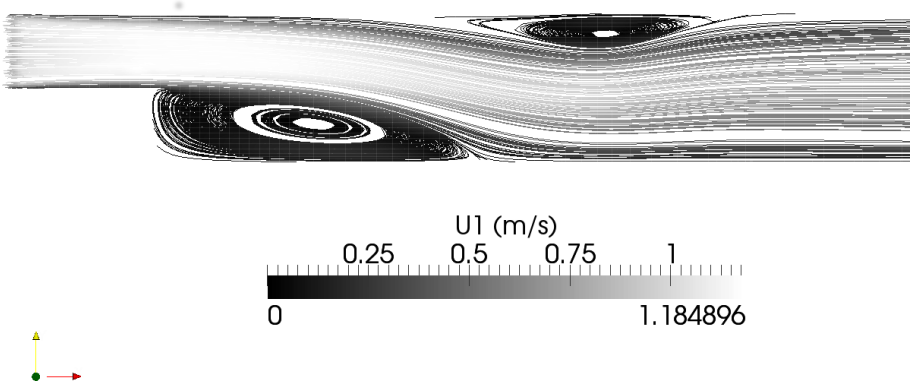
$$x = \frac{N_T(0)}{r_d(0)}v, \qquad f(x,0) = \frac{r_d(0)}{[N_T(0)]^2}F(v,0) \tag{7.27}$$

podemos escrever:

$$\int_0^\infty f(x,0)\, dx = 1, \qquad \int_0^\infty xf(x,0)\, dx = 1 \tag{7.28}$$

Portanto, para simulações PB-CFD, as condições de entrada iniciais para a distribuição foram dadas pela equação:

$$f(x,0) = e^{-x}, \quad x \in [0, \infty) \tag{7.29}$$



(a)

Figura 7.15: Magnitude da velocidade (m/s) da fase dispersa 1 para o caso 2 com a malha 3 para tolerâncias absoluta e relativa de 10^{-6} no tempo de 0.3 segundos para o método D^{2}uQMoGeM-FC.

Na qual satisfaz a Equação (7.28). O seguintes valores foram usados na transformação adimensional: $N_T(0) = 3,38 \times 10^{12} m^{-3}$ e $r_d(0) = 0,073784$.

Para os testes, foram usados dois casos. Caso 1 agregação dominante e caso 2 de quebra dominante, de acordo com o parâmetro $\Phi(\infty)$ abaixo:

- Caso 1: $\Phi(\infty) = 0,5$
- Caso 2: $\Phi(\infty) = 2,0$

Para o acoplamento PB-CFD, a variável adimensional x deve ser dimensionalizada, seguindo as equações 7.30 e 7.31, que consideram as partículas esféricas.

$$d_\alpha = \left(\frac{6 r_d(0)}{\pi N_T(0)} \right)^{1/3} (x_\alpha)^{1/3}, \tag{7.30}$$

$$r_\alpha = r_d(0) \zeta_\alpha; \tag{7.31}$$

Note que sendo $x \in [0, \infty)$, a base polinomial de Laguerre foi utilizada para o cálculo dos momentos generalizados e para a representação funcional da FDN.

Observa-se que pela Figura 7.15, que a região de recirculação no BFS é evidenciada logo após ao degrau. Nesta região, o tempo de residência da fase dispersa é maior e, portanto, os efeitos de quebra e agregação são maiores. Pode-se verificar também que o diâmetro das partículas nesta região aumenta na Figura 7.16(a) e diminui na Figura 7.16(b). Esse resultado era esperado já que os casos 1 e 2 são casos de agregação e quebra dominante, respectivamente. Portanto, os resultados são fisicamente consistentes com o esperado.

A Figura 7.17 mostra que os resultados do DuQMoGeM e Direct DuQMoGeM são equivalentes, o que era esperado uma vez que eles são equivalentes também em

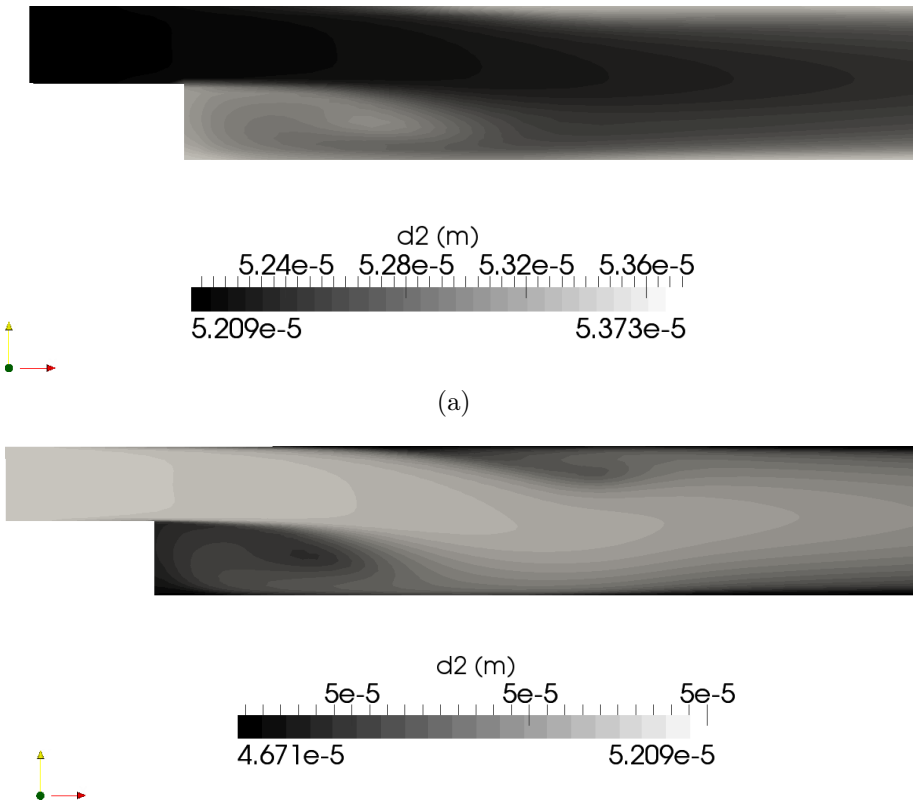


Figura 7.16: Diâmetro (m) da fase dispersa 2 usando a malha 3 e tolerâncias absoluta e relativa de 10^{-6} no tempo final de $0,3$ segundos para o método D^{2}uQMoGeM-FC: (a) caso 1 (b) caso 2.

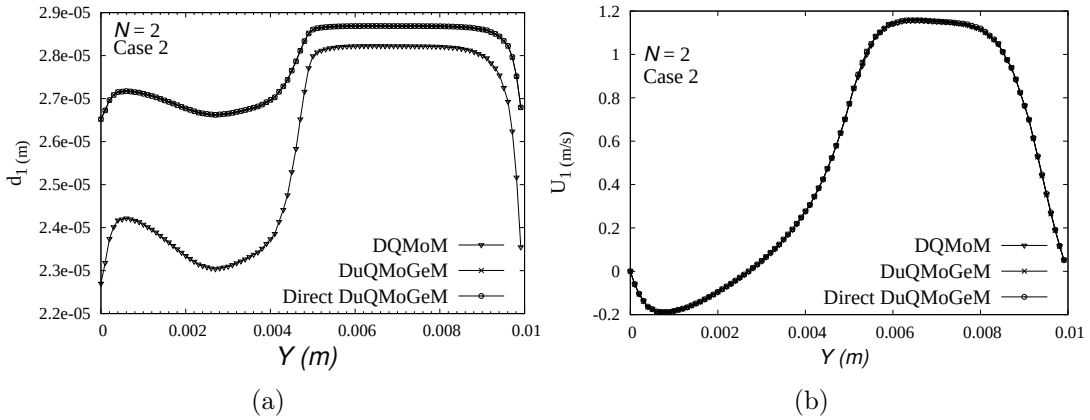


Figura 7.17: Perfil em $X = 2 \times H$ comparando os métodos D^{2}uQMoGeM-FC, DQMoM-FC e DuQMoGeM-FC para o caso 2 no tempo final de $0,3$ segundos para a malha 3. (a) Evolução do diâmetro da fase dispersa 1 (m) (b) Magnitude da velocidade (m/s) da fase dispersa 1.

Tabela 7.2: Relação do tempo computacional entre os métodos DQMoM-FC e D^{2}uQMoGeM-FC paralelo utilizado GTX TITAN para o caso 1.

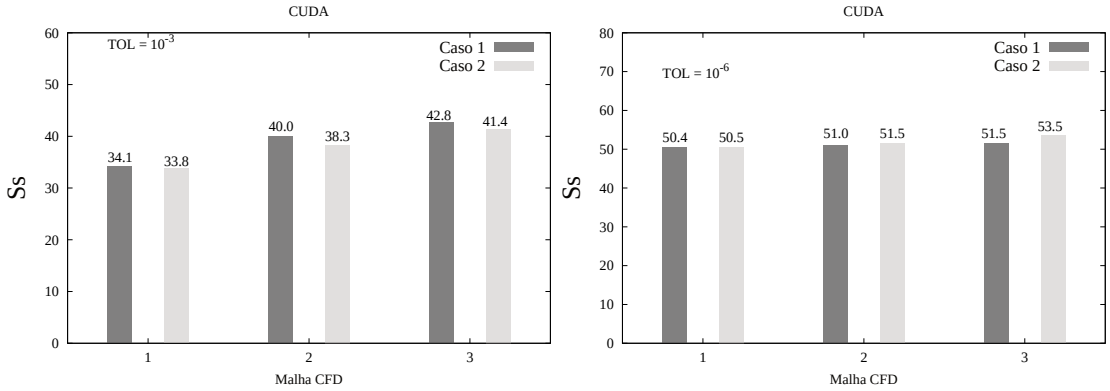
Malha	Υ com $TOL = 10^{-1}$	Υ com $TOL = 10^{-3}$	Υ com $TOL = 10^{-6}$
1	$16,0$	34	120
2	$16,3$	35	120
3	$15,5$	38	121

termos de acurácia. No entanto, os valores de diâmetro obtidos pelo DQMoM são diferentes dos obtidos pelos métodos anteriores em até 15%. Isso ocorre devido ao seu erro de quadratura. Nesse sentido, pode-se dizer que os métodos DuQMoGeM e Direct DuQMoGeM são provavelmente superiores em termos de acurácia para os casos não homogêneos.

A Figura 7.18 mostra a aceleração obtida pela uso da GPU. Pode-se concluir que a aceleração, S_s, depende da tolerância. Isso ocorre porque quando a tolerância é reduzida, a contribuição da AC para o tempo total de simulação é menor. Por outro lado, pode-se observar também que S_s não varia com o aumento da malha. Isso porque o número de integrandos é alto o suficiente para "saturar" a GPU, e portanto, a aceleração se mantém praticamente constante. Além disso, a Figura 7.18 mostra que S_s é pelo menos 34 para os casos com tolerância de 1×10^{-3} e é cerca de 50 para os casos com tolerância de 1×10^{-6}, resultados considerados bem significativos até mesmo para computação em GPUs. Já na Figura 7.19, a aceleração S_p é avaliada. Note que para 2 processos em CPU, S_p é próximo de 1.8 quando duas GPUs diferentes são utilizadas. Esse resultado acontece devido a redução da competição da transferência de dados quando os processos em CPU dividem uma única GPU. Por fim, o tempo total alcançado por 4 processos em CPU usando 2 GPUs está por volta de 100 vezes se comparado ao teste serial, o que pode tornar razoável o custo computacional das simulações usando os DQBMMs razoáveis computacionalmente. A Tabela 7.2 mostra a relação entre os tempos computacionais do DQMoM-FC e do D^{2}uQMoGeM-FC paralelo para Caso 1. Para os casos 1 e 2, a razão entre esses tempos são semelhantes, uma vez que seus custos computacionais e acelerações também o são. Para o caso mais crítico, o D^{2}uQMoGeM-FC serial é cerca de 6413 vezes mais lento que o DQMoM-FC para malha 3 com tolerância de 10^{-6}. Se os recursos de computação paralela forem utilizados essa discrepância cai para 55.

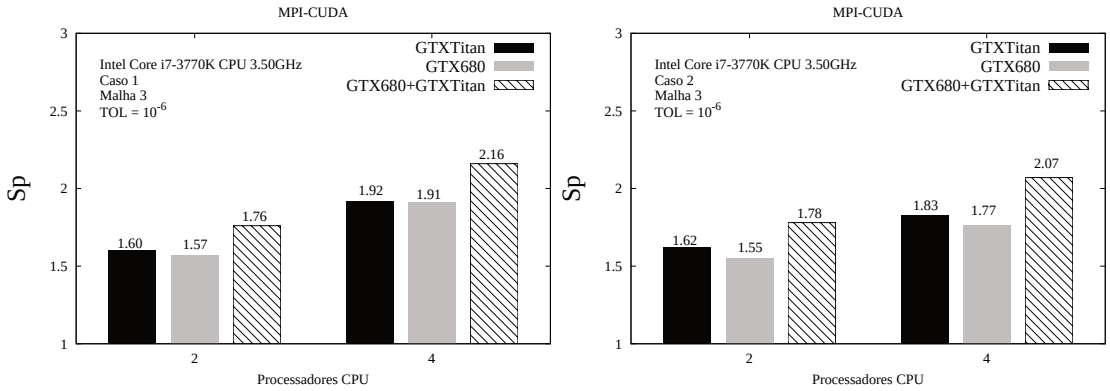
7.4.2 Conclusão

No presente trabalho, a modelagem de simulação multifásica polidispersa por balanço populacional foi implementada no OpenFOAM® utilizando os métodos de



(a) (b)

Figura 7.18: Análise de aceleração do código em CUDA para diferentes malhas CFD e tolerâncias para os casos 1 e 2 usando GTX TITAN (a) Tolerância de 10^{-3} (b) Tolerância de 10^{-6}.



(a) (b)

Figura 7.19: Análise de aceleração MPI-CUDA para diferentes GPUs e número de processos em CPU (a) caso 1 (b) caso 2.

quadratura dupla baseados em momentos. Essa metodologia se mostrou extremamente custosa computacionalmente e, por essa razão, foi implementada também de forma paralela em GPUs.

Visando verificar a metodologia proposta, o acoplamento DQBMM-PB-CFD foi aplicado a uma emulsão água-óleo e seu resultado foi comparado com o mesmo código usando o DQMoM. Os resultados foram similares. Contudo, pode-se considerar que os resultados obtidos com o DQBMM são superiores em termos de acurácia, um vez que eles controlam os erros de quadratura presentes do DQMoM. Além disso, os casos testes já haviam sido analisados e comparados com soluções analíticas em dimensão zero, onde os DQBMMs se mostraram superiores aos QBMMs.

Como as cubaturas adaptativas (AC) devem ser calculadas em cada célula CFD e em cada passo de tempo, o seu tempo computacional é muito elevado. A solução para esse desafio foi paralelizar esse trecho extremamente caro utilizando GPUs. Aqui, o algoritmo paralelo de AC proposto por SANTOS *et al.* [31] foi utilizado. A partir do resultado da análise de performance, pode-se concluir que a aceleração depende da tolerância especificada, o que pode ser explicada com o aumento da contribuição da AC para o tempo total, quando tolerâncias mais restritas são utilizadas. Concluí-se também que a aceleração não aumenta com o tamanho da malha devido ao limite físico das GPUs. Em outras palavras, existe um número máximo de *threads* a partir do qual a computação passa a ser serializada, quando a aceleração não é mais alterada.

Para os casos 1 e 2, a aceleração foi pelo menos 34 para tolerância de 1×10^{-3} e 50 para tolerância de 1×10^{-6}. Na análise MPI-CUDA, quando a malha de CFD é decomposta e usada com duas GPUs diferentes, o trabalho é divido entre as GPUs e a competição na etapa de transferência é menor. Por isso, quando são utilizados 4 processos em CPU e 2 GPUs, a aceleração é maior, e atinge cerca de 100 vezes com tolerância de 1×10^{-6} para a malha 3. Já para o caso 3, disponível no anexo, a aceleração é cerca de 10. Isso ocorre porque a custo da AC é muito menos importante se comparado ao tempo total de simulação. No entanto, quando duas GPUs são usadas, a operação é efetivamente dividida entre elas, e alguma aceleração é obtida. Com dois processos e quatro processos, obtém-se acelerações de 1.4 e 1.75, respectivamente.

Embora as simulações tenham sido realizadas com sucesso, sem os algoritmos paralelos elas seriam impraticáveis, e portanto, o uso de computação paralela é essencial. Vale mencionar também que existem algumas lagunas nessa abordagem que devem ser investigadas, como por exemplo, o tipo de base a ser utilizada na expansão funcional ou até mesmo quais funções de interpolação são ideias para a conservação dos momentos generalizados. Essas perguntas ficam em aberto e ficam como sugestão para o aprimoramento desse tipo método. Outra fato importante,

é que apesar do algoritmo paralelo viabilizar o uso do DQBMM, o mesmo ainda é significativamente mais caro que os QBMMs.

Capítulo 8

Conclusões e Sugestões

8.1 Conclusões

O presente trabalho teve como objetivo desenvolver e implementar ferramentas numéricas em arquiteturas híbridas, placas gráficas (GPUs) e processadores (CPUs), com o intuito de simular escoamentos multifásicos polidispersos de forma rápida e acurada. Para tal, o presente projeto foi, basicamente, dividido em três etapas ou metas:

1. Desenvolvimento de um método numérico para solução da equação de balanço populacional com controle de acurácia;

2. Paralelização usando GPUs o método numérico desenvolvido neste trabalho;

3. Implementação do acoplamento PB-CFD no OpenFOAM® utilizando o método numérico paralelo desenvolvido em GPUs.

Primeiramente, foi desenvolvido, formulado, implementado e testado o método *Direct Dual Quadrature Method of Generalized Moments* (D^2uQMoGeM), visando a solução da equação de balanço populacional. Esse método, fundamentalmente, mistura as características dos métodos DQMoM e DuQMoGeM. No D^2uQMoGeM, os pesos e abscissas são avaliados de forma direta como no DQMoM e os erros de quadratura são controlados por uma cubatura adaptativa como no DuQMoGeM. Cabe salientar que o DuQMoGeM não possibilitava advectar diferentes velocidades para cada fase. E por isso, a ideia inicial de desenvolver sua forma direta.

Vários casos monovariados foram usados para comparar os métodos D^2uQMoGeM e DQMoM com relação à acurácia e ao custo computacional. Em todos os casos, o D^2uQMoGeM obteve pelo menos o mesmo grau de acurácia que o DQMoM. Contudo, na maioria dos casos, foi mais custoso computacionalmente. A acurácia se deve ao fato do controle de erro. No entanto, nos casos onde esse controle não se faz necessário, como por exemplo os com *Kernels* constantes, o DQMoM leva

vantagem, devido ao menor custo computacional. No entanto, vale lembrar, que os *Kernels* mais realísticos, em sua maioria, não são constantes, e portanto, o controle de erro pode ser necessário.

Apesar do custo computacional do D^{2}uQMoGeM ser ligeiramente superior para os casos testados, isso só ocorre porque os *Kernels* tanto de quebra quanto agregação não dependem do tempo e, por isso, a avaliação desses termos pela algoritmo de cubatura adaptativa é apenas realizada no $t = 0$. Portanto, quando aplicado a simulações CFD, esse custo aumenta e o método se torna impraticável.

A desvantagem do *Direct DuQMoGeM* é o custo computacional associado à integração numérica das integrais presentes na PBE. Essas integrais são calculadas independentemente e, por isso, passível de paralelização em GPUs. Nesse sentido, aqui foram propostos dois algoritmos paralelos de cubatura adaptativa em GPUs, um considerando o paralelismo ao nível do cálculo na fórmula de cubatura (PIFL) e outro ao nível do integrando (PIL). Esses algoritmos foram testados exaustivamente com relação a escalabilidade e aceleração. A partir dessa avaliação, pode-se verificar que as acelerações variavam de acordo número de integrandos e a complexidade dos integrandos. Nos casos testes estudados, a máxima aceleração variou de 6 até 70 vezes de acordo com a complexidade da função. Entre os dois métodos, recomenda-se o método PIL para aplicações que envolvam CFD. Isso porque além de ser mais rápido, o algoritmo PIL permite computar maior número de integrais simultaneamente.

SANTOS *et al.* [111] analisaram o sistema linear gerado para solução dos métodos tipo DQMoM e concluíram que os mesmo podem ser mal-condicionados ou próximos da singularidade. Em outras palavras, os métodos de eliminação Gaussiana e de decomposição LU podem falhar na solução desse sistema linear. Nesse sentido a técnica de decomposição em valores singulares (SVD) auxilia não apenas em resolver sistemas lineares singulares ou quase singulares, mas também diagnostica a origem da singularidade, e por isso, o mesmo foi implementado em GPUs utilizando o conceito de SIMT (*Single-Instruction Multiple Threads*). Nessa implementação cada *thread* soluciona um pequeno sistema linear. Essa técnica propicia uma aceleração significativa ao método SVD resolvido em cada nó de uma malha CFD. No entanto, a aceleração depende do número de nós da malha CFD. Os resultados obtidos nesse trabalho mostram que só é válido seu uso quando o número de células da malha CFD é superior a 256.

Note que o custo computacional do método D^{2}uQMoGeM-FC (*Fully Conservative*) associado ao sistema linear é menor que o associado a cubatura adaptativa. Portanto, a influência da aceleração adquirida na implementação SVD, não é significante se comparado ao tempo total de simulação, não influenciando, assim, na aceleração global da aplicação.

Finalmente, a solução da equação de balanço populacional foi realizada com o D^{2}uQMoGeM em sua forma totalmente conservativa acoplada com a formulação multi-fluido no OpenFOAM®. Com intuito de verificar o código, simulações numéricas considerando os efeitos de quebra e agregação foram executadas, para o escoamento de uma emulsão de óleo e água. Para comparação, os mesmos casos foram simulados com o método DQMoM clássico. A partir dessa comparação, concluíu-se que os métodos são semelhantes nas regiões onde os efeitos de quebra e agregação não são predominantes, contudo, nas regiões onde esse efeito é pronunciado, as variáveis resultantes da equação de balanço populacional são ligeiramente diferentes. Isso pode ser atribuído ao erro de quadratura presente no DQMoM. Então, pode-se concluir que as simulações realizadas com o D^{2}uQMoGeM-FC são mais acuradas, devido ao seu controle de erro pela cubatura adaptativa.

De fato a cubatura adaptativa é o limitante computacional do D^{2}uQMoGeM-FC. A paralelização desse gargalo computacional pelo algoritmo PIL [31] viabilizou o uso dessa metodologia em aplicações que envolvam CFD. As acelerações obtidas variaram de 10 à 50, dependendo da tolerância especificada e do caso simulado, para os casos em que apenas uma GPU for utilizada. Quando a malha CFD é também decomposta e dividida entre duas GPUs, essa aceleração pode aumentar, atingindo 100, se 4 processos de CPU forem utilizados. Ficou claro que as simulações são impraticáveis se as GPUs não forem usadas.

8.2 Sugestões para trabalhos futuros

Para simulações multifásicas mais realísticas, alguns avanços na metodologia proposta devem ser incluídos. Entre elas pode-se citar:

- Implementação de modelos mais complexos de quebra e agregação em sua forma paralela.
- Implementação de classes *containers* para os termos de nucleação e crescimento no OpenFOAM® visando caraterizar efeitos de mudança de fase.
- Extensão da formulação para aplicações multivariadas.

Além disso, ainda existem alguns avanços em termos numéricos a serem explorados. Por fim, sugere-se os seguintes desenvolvimentos numéricos:

- Avaliar o D^{2}uQMoGeM com outras bases polinomiais.
- Avaliar o método EQMoM com o algoritmo de cubatura adaptativa paralelo. Note que esse método garante algumas propriedades da FDN, como por exemplo sua positividade, e portanto, pode fornecer alguma vantagem em termos de acurácia e custo computacional frente ao D^{2}uQMoGeM.

- Com relação a cubatura adaptativa, existem outros níveis de paralelismo que podem ser explorados, como por exemplo, o paralelismo no nível das subdivisões. Esse paralelismo pode ser feito utilizando a ferramenta *dynamic parallelism* disponível na versão cuda 5.5 e nas novas arquiteturas Kepler. Essa ferramenta permite que uma função global execute outra função global. Sua vantagem está na possibilidade de paralelizar o algoritmo de subdivisões dentro da própria GPU, dividindo entre os núcleos da GPU a etapa adaptativa do algoritmo.

Referências Bibliográficas

[1] OPENFOAM. *OpenFOAM. The Open Source CFD Toolbox. User Guide.* Free Software Foundation, Inc., 2009.

[2] NVIDIA CORPORATION. "NVIDIA CUDA C Programming Guide". 2010. Version 3.2.

[3] SILVA, L. F. L. R., LAGE, P. L. C. "Development and implementation of a polydispersed multiphase flow model in OpenFOAM", *Computers & Chemical Engineering*, v. 35, n. 12, pp. 2653 – 2666, 2011.

[4] FOX, R., LAURENT, F., MASSOT, M. "Numerical simulation of spray coalescence in an Eulerian framework: Direct quadrature method of moments and multi-fluid method", *Journal of Computational Physics*, v. 227, n. 6, pp. 3058 – 3088, 2008.

[5] FOX, R. "Bivariate direct quadrature method of moments for coagulation and sintering of particle populations", *Journal of Aerosol Science*, v. 37, n. 11, pp. 1562 – 1580, 2006.

[6] PETITTI, M., NASUTI, A., MARCHISIO, D. L., VANNI, M., BALDI, G., etal. "Bubble size distribution modeling in stirred gas–liquid reactors with QMOM augmented by a new correction algorithm", *AIChE Journal*, v. 56, n. 1, pp. 36–53, 2010.

[7] VALE, H., MCKENNA, T. "Modeling particle size distribution in emulsion polymerization reactors", *Progress in Polymer Science*, v. 30, n. 10, pp. 1019 – 1048, 2005.

[8] WULKOW, M., GERSTLAUER, A., NIEKEN, U. "Modeling and simulation of crystallization processes using Parsival", *Chemical Engineering Science*, v. 56, n. 7, pp. 2575 – 2588, 2001.

[9] BUFFO, A., VANNI, M., MARCHISIO, D., FOX, R. "Multivariate Quadrature-Based Moments Methods for turbulent polydisperse gas–liquid systems", *International Journal of Multiphase Flow*, v. 50, pp. 41 – 57, 2013.

[10] YEOH, G. H., TU, J. *Computational Techniques for Multiphase Flows*. Oxford, Butterworth-Heinemann, 2010.

[11] CROWE, C., SOMMERFELD, M., TSUJI, Y. *Multiphase Flows with Droplets and Particles*. CRC Press, 1998.

[12] MARCHISIO, D., FOX, R. *Computational Models for Polydisperse Particulate and Multiphase Systems*. Cambridge Series in Chemical Engineering. Cambridge University Press, 2013.

[13] SILVA, L. F. L. R. *Desenvolvimento de Metodologias para Simulação de Escoamentos Polidispersos usando Código Livre*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, PEQ/COPPE, Brasil, RJ., 2008.

[14] RAMKRISHNA, D. *Population Balance - Theory and Applications to Particulate Systems in Engineering*. Academic Press, San Diego, 2000.

[15] LAGE, P. L. C., ESPÓSITO, R. O. "Experimental determination of bubble size distributions in bubble columns: Prediction of mean bubble diameter and gas hold up", *Powder Technology*, v. 101, n. 2, pp. 142 – 150, 1999.

[16] LAGE, P. L. C., RIBEIRO JR, C. P. "Experimental study on bubble size distributions in a direct-contact evaporator", *Brazilian Journal of Chemical*, v. 21, n. 1, pp. 69 – 81, 2004.

[17] LAGE, P. L. C., RIBEIRO JR, C. P. "Direct-contact evaporation in the homogeneous and heterogeneous bubbling regimes. Part I: experimental analysis", *International Journal of Heat and Mass Transfer*, v. 47, n. 17 - 18, pp. 3825 – 3840, 2004.

[18] RIBEIRO JR., C. P., BORGES, C. P., LAGE, P. L. C. "Modelling of direct-contact evaporation using a simultaneous heat and multicomponent mass-transfer model for superheated bubbles", *Chemical Engineering Science*, v. 60, n. 6, pp. 1761 – 1772, 2005.

[19] LAGE, P. L. C., RIBEIRO JR, C. P. "Direct-contact evaporation in the homogeneous and heterogeneous bubbling regimes. Part II: dynamic simulation", *International Journal of Heat and Mass Transfer*, v. 47, n. 17-18, pp. 3841 – 3854, 2004.

[20] CAMPOS, F. B., LAGE, P. L. C. "Simultaneous heat and mass transfer during the ascension of superheated bubbles", *International Journal of Heat and Mass Transfer*, v. 43, n. 2, pp. 179 – 189, 2000.

[21] LAGE, P. L. C., RIBEIRO JR, C. P. "Gas-Liquid Direct-Contact Evaporation: A Review", *Chemical Engineering and Technology*, v. 28, n. 10, pp. 1081 – 1107, 2005.

[22] LAGE, P. L. C., RIBEIRO JR, C. P. "Sparger effects during the concentration of synthetic fruit juices by direct-contact evaporation", *Journal of Food Engineering*, v. 79, n. 3, pp. 979 – 988, 2007.

[23] SILVA, L. F. L. R., FONTES, C. E., LAGE, P. L. C. "Front tracking in recirculating flows: a comparison between the TVD and RCM methods in solving the VOF equation", *Brazilian Journal of Chemical Engineering*, v. 22, n. 1, pp. 105 – 1016, 2005.

[24] MONTEAGUDO, J. E. P., SILVA, L. F. L. R., LAGE, P. L. C. "Scaling laws for network model permeability: lation with solid deposition", *Chemical Engineering Science*, v. 58, n. 9, pp. 1815 – 1829, 2005.

[25] ARAÚJO, J. M. *Estudo dos modelos de quebra e coalescência para escoamentos polidispersos.* Tese de Mestrado, Universidade Federal do Rio de Janeiro, PEQ/COPPE, Brasil, RJ., 2006.

[26] MITRE, J. F., TAKAHASHI, R. S. M., RIBEIRO JR., C. P., LAGE, P. L. C. "Analysis of breakage and coalescence models for bubble columns", *Chemical Engineering Science*, v. 65, pp. 6089–6100, 2010.

[27] DAMIAN, R. *Acoplamento de Balanço Populacional à simulação computacional de escoamentos polidispersos.* Tese de Mestrado, Universidade Federal do Rio de Janeiro, COPPE/UFRJ, Brasil, RJ, 2007.

[28] ARAUJO, J. M. *Modelo de Quebra e Coalescência de Gotas Para o Escoamento de Emulsões.* Tese de Doutorado, Universidade Federal do Rio de Janeiro, PEQ/COPPE, Brasil, RJ., 2010.

[29] LAGE, P. L. C. "On the representation of QMOM as a weighted-residual method — The dual-quadrature method of generalized moments", *Computers & Chemical Engineering*, v. 35, n. 11, pp. 2186 – 2203, 2011.

[30] FAVERO, J. L., LAGE, P. L. C. "The dual-quadrature method of generalized moments using automatic integration packages", *Computers & Chemical Engineering*, v. 38, pp. 1 – 10, 2012.

[31] SANTOS, F. P., FAVERO, J. L., LAGE, P. L. C. "Solution of the population balance equation by the direct dual quadrature method of generalized moments", *Chemical Engineering Science*, v. 101, pp. 663 – 673, 2013.

[32] FAVERO, J. L. *Simulação de Escoamentos Multifásicos Polidispersos Multivariados*. Tese de Doutorado, Programa de Engenharia Química, COPPE/UFRJ, 2014.

[33] TROBEC, R., VAJTERŠIC, M., ZINTERHOF, P. *Parallel Computing: Numerics, Applications, and Trends*. Springer, 2009.

[34] KIRK, D. B., MEI W. HWU, W. *Programming Massively Parallel Processors: A Hands-on Approach*. Elsevier, 2010.

[35] "Projeto TOP500 dos 500 supercomputadores mais poderosos do mundo." `http://www.top500.org/`.

[36] TURING, A. M. "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, v. 2, n. 42, pp. 230–265, 1936.

[37] SMOLUCHOWSKI, M. V. "Drei Vortrage uber Diffusion, Brownsche Bewegung und Koagulation von Kolloidteilchen", *Zeitschrift fur Physik*, v. 17, pp. 557–585, 1916.

[38] HULBURT, H., KATZ, S. "Some problems in particle technology: A statistical mechanical formulation", *Chemical Engineering Science*, v. 19, n. 8, pp. 555 – 574, 1964.

[39] PANDIS, S. N., SEINFELD, J. H. *Atmospheric chemistry and physics: from air pollution to climate change.* New York: John Wiley and Sons., 1998.

[40] HIDY, G. M., BROCK, J. R. *The dynamics of aerocolloidal systems.* Oxford: Pergamon., 1970.

[41] FRIEDLANDER, S. K. *Smoke, dust and haze: Fundamentals of aerosol dynamics (2nd ed.).* Oxford University Press., 2000.

[42] MASSOT, M., LAURENT, F., KAH, D., CHAISEMARTIN, S. D. "A robust moment method for evalutation of the disappearance of evaporating sprays", *SIAM Journal on Applied Mathematics*, v. 70, n. 8, pp. 3203–3234, 2010.

[43] MULLIN, J. W. *Crystallization.* Butterworth-Heinemann, 2001.

[44] PATIL, D. P., ANDREWS, J. R. G. "An analytical solution to continuous population balance model describing floc coalescence and breakage – A special case", *Chemical Engineering Science*, v. 53, n. 3, pp. 599 – 601, 1998.

[45] LAGE, P. L. C. "Comments on the by D.P. Patil and J.R.G. Andrews [Chemical Engineering Science 53(3) 599-601]", *Chemical Engineering Science*, v. 57, n. 19, pp. 4253 – 4254, 2002.

[46] MCCOY, B. J., MADRAS, G. "Analytical solution for a population balance equation with aggregation and fragmentation", *Chemical Engineering Science*, v. 58, n. 13, pp. 3049 – 3051, 2003.

[47] LIOU, J.-J., SRIENC, F., FREDRICKSON, A. G. "Solutions of population balance models based on a successive generations approach", *Chemical Engineering Science*, v. 52, n. 9, pp. 1529 – 1540, 1997.

[48] KENDALL, D. "An Artificial Realization of a SImple Birth-and-Death Process", *Royal Statistical Society, Serie B*, v. 12, pp. 116 – 119, 1950.

[49] SHAH, B. H., RAMKRISHNA, D., BORWANKER, J. D. "Simulation of Particulate Systems Using teh Concepts of the Interval of Quiescence", *AICHE Journal*, v. 53, pp. 897 – 904, 1977.

[50] DAS, P. K. "Monte Carlo simulation of drop breakage on the basis of drop volume", *Computers & Chemical Engineering*, v. 20, n. 3, pp. 307 – 313, 1996.

[51] GOODSON, M., KRAFT, M. "Simulation of coalescence and breakage: an assessment of two stochastic methods suitable for simulating liquid-liquid extraction", *Chemical Engineering Science*, v. 59, n. 18, pp. 3865 – 3881, 2004.

[52] EIBECK, A., WAGNER, W. "Stochastic Particle Approximations for Smoluchoski's Coagulation Equation", *Annals of Applied Probability*, v. 11, pp. 1137 – 1165, 2001.

[53] DEBRY, E., SPORTISSE, B., JOURDAIN, B. "A stochastic approach for the numerical simulation of the general dynamics equation for aerosols", *Journal of Computational Physics*, v. 184, n. 2, pp. 649 – 669, 2003.

[54] IMMANUEL, C. D., DOYLE, F. J. "Solution technique for a multi-dimensional population balance model describing granulation processes", *Powder Technology*, v. 156, n. 2-3, pp. 213 – 225, 2005. Particle Technology Forum Special Issue - Papers presented in the Particle Technology Forum sessions at the 2003 Annual AIChE meeting in San Francisco (November, 2003).

[55] ZHAO, H., MAISELS, A., MATSOUKAS, T., ZHENG, C. "Analysis of four Monte Carlo methods for the solution of population balances in dispersed systems", *Powder Technology*, v. 173, n. 1, pp. 38 – 50, 2007.

[56] IRIZARRY, R. "Fast Monte Carlo methodology for multivariate particulate systems—I: Point ensemble Monte Carlo", *Chemical Engineering Science*, v. 63, n. 1, pp. 95 – 110, 2008.

[57] IRIZARRY, R. "Fast Monte Carlo methodology for multivariate particulate systems-II: tau-PEMC", *Chemical Engineering Science*, v. 63, n. 1, pp. 111 – 121, 2008.

[58] IRIZARRY, R. "Stochastic simulation of population balance models with disparate time scales: Hybrid strategies", *Chemical Engineering Science*, v. 66, n. 18, pp. 4059 – 4069, 2011.

[59] ZHAO, H., KRUIS, F. E., ZHENG, C. "A differentially weighted Monte Carlo method for two-component coagulation", *Journal of Computational Physics*, v. 229, n. 19, pp. 6931 – 6945, 2010.

[60] ZHAO, H., ZHENG, C. "A population balance-Monte Carlo method for particle coagulation in spatially inhomogeneous systems", *Computers & Fluids*, v. 71, pp. 196 – 207, 2013.

[61] WEI, J., KRUIS, F. "A GPU-based parallelized Monte-Carlo method for particle coagulation using an acceptance–rejection strategy", *Chemical Engineering Science*, v. 104, pp. 451 – 459, 2013.

[62] KUMAR, S., RAMKRISHNA, D. "On the solution of population balance equations by discretization–I. A fixed pivot technique", *Chemical Engineering Science*, v. 51, n. 8, pp. 1311 – 1332, 1996.

[63] BLECK, R. "A Fast Approximate Method for Integrating the Stochastic Coalescence Equation", *Geophysical Research Oceans*, v. 49, pp. 77–80, 1970.

[64] HOUNSLOW, M., RYALL, R., MARSHALL, V. "A Discretized Population Balance for Nucleation, Growth adn Aggregation", *AICHE Journal*, v. 34, pp. 1821 – 1832, 1988.

[65] LISTER, J., SMIT, D., HOUNSLOW, M. "Adjustable discretized population balance for growth and aggregation", *AICHE*, v. 41, pp. 591 – 603, 1995.

[66] KUMAR, S., RAMKRISHNA, D. "On the solution of population balance equations by discretization–II. A moving pivot technique", *Chemical Engineering Science*, v. 51, n. 8, pp. 1333 – 1342, 1996.

[67] NOPENS, I., BEHEYDT, D., VANROLLEGHEM, P. A. "Comparison and pitfalls of different discretised solution methods for population balance models: a simulation study", *Computers & Chemical Engineering*, v. 29, n. 2, pp. 367 – 377, 2005.

[68] ALEXOPOULOS, A., KIPARISSIDES, C. "Solution of the bivariate dynamic population balance equation in batch particulate systems: Combined aggregation and breakage", *Chemical Engineering Science*, v. 62, n. 18-20, pp. 5048 – 5053, 2007. 19th International Symposium on Chemical Reaction Engineering - From Science to Innovative Engineering - ISCRE-19.

[69] CHAKRABORTY, J., KUMAR, S. "A new framework for solution of multidimensional population balance equations", *Chemical Engineering Science*, v. 62, n. 15, pp. 4112 – 4125, 2007.

[70] BARRET, J. E JHEETA, J. "Improving the accuracy of the moments method for solving the aerosol dynamic equation", *Aerosol Science*, v. 27, pp. 1135 – 1142, 1996.

[71] MARKATOU, P., WANG, H., FRENKLACH, M. "A computational study of sooting limits in laminar premixed flames of ethane, ethylene, and acetylene", *Combustion and Flame*, v. 93, n. 4, pp. 467 – 482, 1993.

[72] FRENKLACH, M., WANG, H. "Detailed modeling of soot particle nucleation and growth", *Symposium (International) on Combustion*, v. 23, n. 1, pp. 1559 – 1566, 1991.

[73] PINTO, J., LAGE, P. L. C. *Métodos Numéricos em Problemas de Engenharia Química*. E-papers, 2001.

[74] GELBARD, F., SEINFELD, J. H. "Numerical solution of the dynamic equation for particulate systems", *Computational Physics*, v. 28, pp. 357 – 375, 1978.

[75] EYRE, D., WRIGHT, C., REUTER, G. "Spline-Collocation wit adaptative mesh grading for solving the sthocastic colecation equation", *Computational Physics*, v. 78, pp. 288 – 304, 1988.

[76] CHEN, M.-Q., HWANG, C., SHIH, Y.-P. "A wavelet-Galerkin method for solving population balance equations", *Computers & Chemical Engineering*, v. 20, n. 2, pp. 131 – 145, 1996.

[77] NICMAINS, M., HOUNSLOW, M. "Finite element method for steady-state population balance equation", *AICHE Journal*, v. 44, pp. 2258 – 2272, 1998.

[78] LIU, Y., CAMERON, I. "A new wavelet method for the solution of the population balance equation", *Chemical Engineering Science*, v. 56, pp. 131 – 145, 2001.

[79] MAHONEY, A. W., RAMKRISHNA, D. "Efficient solution of population balance equations with discontinuities by finite elements", *Chemical Engineering Science*, v. 57, n. 7, pp. 1107 – 1119, 2002.

[80] RIGOPOULOS, S., JONES, A. "Finite-element scheme for solution of dynamics populaion balance equation", *AICHE*, v. 49, pp. 1127–1139, 2003.

[81] DORAO, C., JAKOBSEN, H. "The quadrature method of moments and its relationship with the method of weighted residuals", *Chemical Engineering Science*, v. 61, n. 23, pp. 7795 – 7804, 2006.

[82] ZHU, Z., DORAO, C., JAKOBSEN, H. "A least-squares method with direct minimization for the solution of the breakage-coalescence population balance equation", *Mathematics and Computers in Simulation*, v. 79, n. 3, pp. 716 – 727, 2008.

[83] DORAO, C., JAKOBSEN, H. "hp-adaptive least squares spectral element method for population balance equations", *Applied Numerical Mathematics*, v. 58, n. 5, pp. 563 – 576, 2008.

[84] ZHU, Z., DORAO, C. A., JAKOBSEN, H. A. "Mass Conservative Solution of the Population Balance Equation Using the Least-Squares Spectral Element Method", *Industrial & Engineering Chemistry Research*, v. 49, n. 1, pp. 6204 – 6214, 2010.

[85] MARTÍNEZ-BAZÁN, C., ASHERAS, J. L., EASTWOOD, C. E., MONTAÑES, J. "A review of statistical models for the break-up of an immiscible fluid immersed into a fully developed turbulent flow", *International Journal of Multiphase Flow*, v. 28, n. 2, pp. 247 – 278, 2002.

[86] MCGRAW, R. "Description of aerosol dynamics by the quadrature method of moments", *Aerosol Science and Technology*, v. 27, pp. 255 – 265, 1997.

[87] GAUTSCHI, W. "Construction of Gauss-Christoffel Quadrature Formulas", *Mathematics of Computation*, v. 22, n. 102, pp. pp. 251–270.

[88] GORDON, R. "Error Bounds in Equilibrium Statistical Mechanics", *Journal of Mathematical Physics*, v. 9, pp. 655 – 663, 1968.

[89] WHEELER, J. "Modified moments and Gaussian quadrature", *Rocky Mountain Journal of Mathematics*, v. 4, n. 2, pp. 287 – 296, 1974.

[90] JOHN, V., THEIN, F. "On the efficiency and robustness of the core routine of the quadrature method of moments (QMOM)", *Chemical Engineering Science*, v. 75, pp. 327 – 333, 2012.

[91] WRIGHT, MCGRAW, R., ROSNER, D. E. "Bivariate Extension of the Quadrature Method of Moments for Modeling Simultaneous Coagulation and Sintering of Particle Populations", *Journal of Colloid and Interface Science*, v. 236, n. 2, pp. 242 – 251, 2001.

[92] MCGRAW, R., WRIGHT, D. L. "Chemically resolved aerosol dynamics for internal mixtures by the quadrature method of moments", *Journal of Aerosol Science*, v. 34, n. 2, pp. 189 – 209, 2003.

[93] YOON, C., MCGRAW, R. "Representation of generally mixed multivariate aerosols by the quadrature method of moments: I. Statistical foundation", *Journal of Aerosol Science*, v. 35, n. 5, pp. 561 – 576, 2004.

[94] JOHNSON, R., WICHERN, D. *Applied multivariate statistical analysis*. Prentice-Hall, 1992.

[95] FAVERO, J. L., SILVA, L. F. L. R., LAGE, P. L. C. "Comparison of methods for multivariate moment inversion—Introducing the independent component analysis", *Computers & Chemical Engineering*, v. 60, pp. 41 – 56, 2014.

[96] MARCHISIO, D. "On the use of bi-variate population balance equations for modelling barium titanate nanoparticle precipitation", *Chemical Engineering Science*, v. 64, n. 4, pp. 697–708, 2009.

[97] CHENG, J. C., FOX, R. O. "Kinetic Modeling of Nanoprecipitation using CFD Coupled with a Population Balance", *Industrial & Engineering Chemistry Research*, v. 49, n. 21, pp. 10651–10662, 2010.

[98] COMON, P. "Independent component analysis, A new concept?" *Signal Processing*, v. 36, n. 3, pp. 287 – 314, 1994.

[99] MARCHISIO, D. L., VIGIL, R. D., FOX, R. O. "Quadrature method of moments for aggregation-breakage processes", *Journal of Colloid and Interface Science*, v. 258, n. 2, pp. 322 – 334, 2003.

[100] VANNI, M. "Approximate Population Balance Equations for Aggregation-Breakage Processes", *Journal of Colloid and Interface Science*, v. 221, n. 2, pp. 143 – 160, 2000.

[101] MARCHISIO, D. L., VIGIL, R. D., FOX, R. O. "Implementation of the quadrature method of moments in CFD codes for aggregation-breakage problems", *Chemical Engineering Science*, v. 58, n. 15, pp. 3337 – 3351, 2003.

[102] WAN, B., RING, T. A., DHANASEKHARAN, K. M., SANYAL, J. "Comparison of analytical solutions for cmsmpr crystallizer with QMOM population balance modeling in fluent", *China Particuology*, v. 3, n. 4, pp. 213 – 218, 2005.

[103] SU, J., GU, Z., LI, Y., FENG, S., XU, X. Y. "Solution of population balance equation using quadrature method of moments with an adjustable factor", *Chemical Engineering Science*, v. 62, n. 21, pp. 5897 – 5911, 2007.

[104] MARCHISIO, D. L., FOX, . R. O. "Solution of population balance equations using the direct quadrature method of moments", *Journal of Aerosol Science*, v. 36, n. 1, pp. 43 – 73, 2005.

[105] BOVE, S., SOLBERG, T., HJERTAGER, B. H. "A novel algorithm for solving population balance equations: the parallel parent and daughter classes. Derivation, analysis and testing", *Chemical Engineering Science*, v. 60, n. 5, pp. 1449 – 1464, 2005.

[106] SCOTT, W. "Analytic studies of cloud droplet coalescence", *Atmosphere Science*, v. 25, pp. 54–65, 1968.

[107] SILVA, L. F. L. R., RODRIGUES, R. C., MITRE, J. F., LAGE, P. L. C. "Comparison of the accuracy and performance of quadrature-based methods for population balance problems with simultaneous breakage and aggregation", *Computers & Chemical Engineering*, v. 34, n. 3, pp. 286 – 297, 2010.

[108] PISKUNOV, V. N., GOLUBEV, A. I. "The generalized approximation method for modeling coagulation kinetics–Part 1: justification and implementation of the method", *Journal of Aerosol Science*, v. 33, n. 1, pp. 51 – 63, 2002.

[109] GROSCH, R., MARQUARDT, W., BRIESEN, H., WULKOW, M. "Generalization and Numerical Investigation of QMOM", *AICHE Journal*, v. 53, pp. 207 – 227, 2006.

[110] SU, J., GU, Z., LI, Y., FENG, S., XU, X. Y. "An Adaptive Direct Quadrature Method of Moment for Population Balance Equations", *AICHE Journal*, v. 54, pp. 2872 – 2887, 2008.

[111] SANTOS, F. P., LAGE, P. L. C., FONTES, C. E. "Numerical aspects of direct quadrature-based moment methods for solving the population balance equation", *Brazilian Journal of Chemical Engineering*, v. 30, pp. 643 – 656, 09 2013.

[112] ATTARAKIH, M. M., DRUMM, C., BART, H.-J. "Solution of the population balance equation using the sectional quadrature method of moments (SQMOM)", *Chemical Engineering Science*, v. 64, n. 4, pp. 742 – 752, 2009.

[113] LO, S. "Applications of population balance to CFD modelling of bubbly flow via the MUSIG model", *Rel. Tec. AEAT-1096, AEA Technology*, 1996.

[114] BRUNS, M. C., EZEKOYE, O. A. "Development of a hybrid sectional quadrature-based moment method for solving population balance equations", *Journal of Aerosol Science*, v. 54, pp. 88 – 102, 2012.

[115] STRUMENDO, M., ARASTOOPOUR, H. "Solution of PBE by MOM in finite size domains", *Chemical Engineering Science*, v. 63, n. 10, pp. 2624 – 2640, 2008.

[116] STRUMENDO, M., ARASTOOPOUR, H. "Solution of Population Balance Equations by the Finite Size Domain Complete Set of Trial Functions Method of Moments (FCMOM) for Inhomogeneous Systems", *Industrial & Engineering Chemistry Research*, v. 49, n. 11, pp. 5222 – 5230, 2010.

[117] ATTARAKIH, M. "Integral formulation of the population balance equation: Application to particulate systems with particle growth", *Computers & Chemical Engineering*, v. 48, pp. 1 – 13, 2013.

[118] ATTARAKIH, M., JARADAT, M., HLAWITSCHKA, M., BART, H.-J., KUHNERT, J. "Integral Formulation of the Population Balance Equation using the Cumulative QMOM". In: E.N. Pistikopoulos, M. G., Kokossis, A. (Eds.), *21st European Symposium on Computer Aided Process Engineering*, v. 29, *Computer Aided Chemical Engineering*, Elsevier, pp. 81 – 85, 2011.

[119] ATTARAKIH, M., BART, H.-J. "Integral Formulation of the Smoluchowski Coagulation Equation using the Cumulative Quadrature Method of Mo-

ments (CQMOM)". In: Karimi, I. A., Srinivasan, R. (Eds.), *11th International Symposium on Process Systems Engineering*, v. 31, *Computer Aided Chemical Engineering*, Elsevier, pp. 1130 – 1134, 2012.

[120] KURGANOV, A., TADMOR, E. "New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection–Diffusion Equations", *Journal of Computational Physics*, v. 160, n. 1, pp. 241 – 282, 2000.

[121] YUAN, C., LAURENT, F., FOX, R. "An extended quadrature method of moments for population balance equations", *Journal of Aerosol Science*, v. 51, pp. 1 – 23, 2012.

[122] DREW, D. A., PASSMAN, S. L. *Theory of multicomponent fluids*. Applied mathematical sciences. New York, Springer, 1999.

[123] POPE, S. *Turbulent Flows*. Cambridge University Press, 2000. ISBN: 9780521598866.

[124] SCHILLER, L., NAUMANN, A. Z. "Über die grundlegenden Berechnungen bei der Schwerkraftaufbereitung", *Ver. Deut. Ing.*, v. 77, pp. 318–320, 1933.

[125] ISHII, M., HIBIKI, T. *Thermo-fluid dynamics of two-phase flow*. Birkhäuser, 2006.

[126] SILVA, L. F. L. R., DAMIAN, R. B., LAGE, P. L. C. "Implementation and analysis of numerical solution of the population balance equation in CFD packages", *Computers & Chemical Engineering*, v. 32, n. 12, pp. 2933 – 2945, 2008.

[127] VIKAS, V., WANG, Z., PASSALACQUA, A., FOX, R. "Realizable high-order finite-volume schemes for quadrature-based moment methods", *Journal of Computational Physics*, v. 230, n. 13, pp. 5328 – 5352, 2011.

[128] ABBASI, E., ARASTOOPOUR, H. "Numerical analysis and implementation of finite domain complete trial functions method of moments (FCMOM) in CFD codes", *Chemical Engineering Science*, v. 102, pp. 432 – 441, 2013.

[129] RUSCHE, H. *Computational fluid dynamics of dispersed two-phase flows at high phase fractions*. Tese de Doutorado, Imperial College of Science, Technology and Medicine, UK., 2002.

[130] FLYNN, M. J. "Very high speed computing systems", *Proceedings of IEEE*, v. 54, n. -, pp. 1901–1909, 1966.

[131] SECCHI, A. R. *Simulação Dinâmica de Processos Químicos pelo Método de Relaxação em Forma de Onda em Computadores Paralelos*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, PEQ/COPPE, Brasil, RJ., 1992.

[132] SIMAS, J. J. C. *Simulation of fluid flows with Lattice Boltzmann on GPUs*. Tese de Mestrado, Universidade Técnica de Lisboa, 2010.

[133] SINGH, J. P., CULLER, D. E. *Parallel Computer Architecture, A hardware/-software approach*. Morgan-Kaufmann, 1999.

[134] AMDAHL, G. "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities", *AFIPS Conference Proceedings (30)*, v. 30, pp. 483 –485, 1967.

[135] CHAPMAN, B., JOST, G., VAN DER PAS, R. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007.

[136] GROPP, L., HUSS-LEDERMAN. *MPI - The complete reference*. MIT Press, 1998.

[137] IKEDA, P. A. *Um estudo do uso eficiente de programas em placas gráficas*. Tese de Mestrado, Universidade de São Paulo, 2010.

[138] SANTOS, F. P., SENOCAK, I., FAVERO, J. L., LAGE, P. L. "Solution of the population balance equation using parallel adaptive cubature on GPUs", *Computers & Chemical Engineering*, v. 55, pp. 61 – 70, 2013.

[139] SCHURER, R. *High-Dimensional Numerical Integration on Parallel Computers*. Tese de Mestrado, University of Salzburg, Austria, 2001.

[140] D'APUZZO, M., LAPEGNA, M., MURLI, A. "Scalability and load balancing in adaptive algorithms for multidimensional integration", *Parallel Computing*, v. 23, n. 8, pp. 1199 – 1210, 1997.

[141] LI, C.-J., DAGNINO, C. "An adaptive numerical integration algorithm for polygons", *Applied Numerical Mathematics*, v. 60, n. 3, pp. 165 – 175, 2010.

[142] JOHNSON, S. G. *Cubature (Multi-dimensional integration)*, 2008. http://ab-initio.mit.edu/wiki/index.php/Cubature.

[143] "Product Gauss quadrature rules vs. cubature rules in the meshless local Petrov-Galerkin method", *Journal of Complexity*, v. 26, n. 1, pp. 82 – 101, 2010.

[144] STROUD, A. H. *Approximate calculation of multiple integrals*. Englewood Chiffs, New Jersey: Prentice-Hall, 1971.

[145] SCHÜRER, R. "A comparison between (quasi-)Monte Carlo and cubature rule based methods for solving high-dimensional integration problems", *Mathematics and Computers in Simulation*, v. 62, n. 3-6, pp. 509 – 517, 2003.

[146] GENZ, A. C. "Parallel adaptive algorithms for multiple integrals. In Mathematics for Large Scale Computing", *in Lecture Notes in Pure and Applied Mathematics*, pp. 35 – 47, 1989.

[147] BULL, J., FREEMAN, T. "Parallel globally adaptive algorithms for multidimensional integration", *Applied Numerical Mathematics*, v. 19, n. 1-2, pp. 3 – 16, 1995.

[148] HAHN, T. "Cuba-a library for multidimensional numerical integration", *Computer Physics Communications*, v. 168, n. 2, pp. 78 – 95, 2005.

[149] GENZ, A. C., MALIK, A. A. "An imbedded family of fully symmetric numerical integration rules", *SIAM J. Numer. Anal*, v. 20, n. 3, pp. 580–588, 1983.

[150] SANDERS, J., KANDROT, E. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley, 2010.

[151] GOLUB, G., VAN LOAN, C. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[152] PRESS, W., TEUKOLSKY, S., VETTERLING, W., FLANNERY, B. *Numerical Recipes in C*. 2nd ed. Cambridge, UK, Cambridge University Press, 1992.

[153] HUMPHREY, J. R., PRICE, D. K., SPAGNOLI, K. E., PAOLINI, A. L., KELMELIS, E. J. "CULA: Hybrid GPU Accelerated Linear Algebra Routines", *SPIE Defense and Security Symposium (DSS)*.

[154] LAHABAR, S., NARAYANAN, P. J. "Singular value decomposition on GPU using CUDA". In: *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1–10, 2009.

[155] NVIDIA. *CUBLAS Library User Guide*, v5.0 ed. NVIDIA, 2012. Disponível em: <http://docs.nvidia.com/cublas/index.html>.

[156] HOUSEHOLDER, A. S. "Unitary Triangularization of a Nonsymmetric Matrix", *J. ACM*, v. 5, n. 4, pp. 339–342, 1958.

[157] SPEEDIT, V. "Vratis SpeedIT plugin". 2010. Disponível em: <http://speedit.vratis.com/>.

[158] SYMSCAPE. "GPU v1.0 Linear Solver Library for OpenFOAM". 2011. Disponível em: <http://www.symscape.com/gpu-1-0-openfoam>. Version 1.0.

[159] COMBEST, D. P., DAY, J. "Cufflink: a library for linking numerical methods based on CUDA C/C++ with OpenFOAM." 2011. Disponível em: <http://cufflink-library.googlecode.com>. Version 0.1.0.

[160] HOBEROCK, J., BELL, N. "Thrust: A Parallel Template Library". 2010. Disponível em: <http://thrust.github.io/>. Version 1.7.0.

[161] GAUTSCHI, W. *Orthogonal Polynomials - Computation and Approximation*. Oxford Science, 2004.

[162] SECCHI, A. *DASSLC: User's Manual, a Differential-Algebraic System Solver*. Relatório técnico, UFRGS, Porto Alegre, RS/Brazil, 2007. http://www.enq.ufrgs.br/enqlib/numeric/DASSLC.

[163] GENZ, A. C. "Testing multidimensional integration routines", *Tools, Methods and Languages for Scientific and Engineering Computation*, pp. 81 – 94, 184.

[164] GELBARD, F., SEINFELD, J. "Coagulation and growth of a multicomponent aerosol", *Journal of Colloid and Interface Science*, v. 63, n. 3, pp. 472 – 479, 1978.

[165] JATOBÁ, L. F. C. *Desenvolvimento de métodos numéricos para a simulação da difusão de Maxwell-Stefan no escoamento de misturas multicomponentes semi-contínuas com caracterização adaptativa*. Tese de Doutorado, Programa de Engenharia Química, COPPE/UFRJ (em andamento), 2014.

[166] DESJARDINS, O., FOX, R., VILLEDIEU, P. "A quadrature-based moment method for dilute fluid-particle flows", *Journal of Computational Physics*, v. 227, n. 4, pp. 2514 – 2539, 2008.

ANEXO A

Procedimento de Discretização

Neste anexo, o procedimento de discretização das equações do Capítulo 4 é descrito com mais detalhes. Como mencionado no Capítulo 4, o OpenFOAM® discretiza os operadores de campo, e.g., ∇, a partir das funções estáticas **fvc** e **fvm**. A função **fvm** realiza a discretização pelo método dos volumes finitos, que por sua vez gera a matriz $\mathbf{C}_\alpha$. Já a função **fvc** avalia os operadores de campo explicitamente gerando o termo não homogêneo do sistema linear para a matriz $\mathbf{C}_\alpha$. Da mesma forma que em SILVA e LAGE [3], no presente trabalho, as discretizações implícita e explícita são definidas, respectivamente, por $\lfloor \bullet [\varphi] \rfloor$ e $\underline{\bullet}\,[\varphi]$, onde $\bullet$ representa a discretização do operador diferencial em termos da variável φ.

A.1 Equação multifásica

No Capítulo 4, foi apresentada a Equação 4.12 cuja o lado esquerdo da igualdade é a forma discretizada do lado esquerdo da Equação 4.11. Nesta parte discreta, enquanto que os termos temporal, advectivo e difusivo são tratados implicitamente, o tensor de Reynolds é discretizado explicitamente. Assim o lado esquerdo da Equação 4.12 é dado por:

$$\begin{aligned}\boldsymbol{\Upsilon}_\alpha &= \left\lfloor \frac{\partial[\mathbf{u}_\alpha]}{\partial t} \right\rfloor + \lfloor \nabla \cdot (\psi_\alpha [\mathbf{u}_\alpha]) \rfloor - \lfloor (\nabla \cdot \psi_\alpha) [\mathbf{u}_\alpha] \rfloor - \left\lfloor \nabla \cdot \left(\upsilon_{\alpha,f}^{eff} \frac{\overline{\nabla}_f r_\alpha}{r_{\alpha,f}} [\mathbf{u}_\alpha] \right) \right\rfloor \\ &+ \left\lfloor \upsilon_{\alpha,f}^{eff} \frac{\overline{\nabla}_f r_\alpha}{r_{\alpha,f}} [\mathbf{u}_\alpha] \right\rfloor - \left\lfloor \upsilon_{\alpha,f}^{eff} \nabla^2 ([\mathbf{u}_\alpha]) \right\rfloor + \underline{\nabla} \cdot \boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{eff}} + \frac{\underline{\nabla} r_\alpha}{\langle \overline{r}_\alpha \rangle} \boldsymbol{\tau}_{\boldsymbol{\alpha}}^{\boldsymbol{eff}} \end{aligned} \tag{A.1}$$

em que, ψ_α é o fluxo volumétrico através da superfície do volume de controle, $\psi_\alpha = \mathbf{S}_f \cdot (\mathbf{u}_\alpha)_f$, sendo que $\mathbf{S}_f$ é o vetor de área da face. Nesta equação, o subscrito f e a variável $\langle \overline{r}_\alpha \rangle$ referem-se à variável na face e à fração de fase média na área do volume de controle, respectivamente.

A.2 Termos de transferência na interface

Os termos de tranferência na interface são discretizados de acordo com o trabalho de SILVA e LAGE [3]. A equação semi-discretizada da contribuição da fase dispersa é dada pela Equação A.2.

$$\begin{aligned}\mathbf{M}_\alpha &= r_\alpha K_\alpha^D \mathbf{u_0} - \left\lfloor r_{\alpha,f} K_{\alpha,f}^D \left[\mathbf{u}_\alpha\right]\right\rfloor + r_{\alpha,f} K_{\alpha,f}^L \psi_{r,\alpha} \underline{\times} \left(\underline{\nabla \times} \psi_0\right) \\ &+ r_{\alpha,f} K_{\alpha,f}^{VM} \underline{\frac{D\mathbf{u_0}}{Dt}} - \left\lfloor r_{\alpha,f} K_{\alpha,f}^{VM} \left[\frac{D\mathbf{u}_\alpha}{Dt}\right]\right\rfloor \end{aligned} \tag{A.2}$$

em que, a velocidade da fase dispersa e sua derivada substativa são termos discretizados de forma implícita, diferente dos termos de força de sustentação e derivada substativa da fase contínua. Assim, a discretização das derivadas substativas são dadas pela decomposição abaixo:

$$\left[\frac{D\mathbf{u}_\alpha}{Dt}\right] = \left[\frac{\partial[\mathbf{u}_\alpha]}{\partial t}\right] + \left\lfloor \nabla \cdot \left(\psi_\alpha \left[\mathbf{u}_\alpha\right]\right)\right\rfloor - \left\lfloor \left(\nabla \cdot \psi_\alpha\right)\left[\mathbf{u}_\alpha\right]\right\rfloor \tag{A.3}$$

$$\underline{\frac{D\mathbf{u_0}}{Dt}} = \underline{\frac{\partial[\mathbf{u}_0]}{\partial t}} + \underline{\nabla} \cdot \left(\psi_0 \left[\mathbf{u}_0\right]\right) - \left(\underline{\nabla} \cdot \psi_0\right)\left[\mathbf{u}_0\right] \tag{A.4}$$

Já o termo de transferência de quantidade de movimento para fase contínua é discretizado de acordo com a Equação A.5. Essa última equação contém os efeitos das αs fases dispersas sobre a fase contínua, e portanto, sua discretização é bem semelhante a realizada na Equação A.2. Neste caso, enquanto que a velocidade da fase contínua e sua derivada substativa são discretizados implicitamente, a força de sustentação e a derivada substativa da fase contínua são discretizadas explicitamente.

$$\begin{aligned}\mathbf{M_0} &= \sum_{\alpha=1}^{N} r_\alpha K_\alpha^D \mathbf{u}_\alpha - \left[\sum_{\alpha=1}^{N} r_{\alpha,f} K_{\alpha,f}^D \left[\mathbf{u}_0\right]\right] - \sum_{\alpha=1}^{N} r_{\alpha,f} K_{\alpha,f}^L \psi_{r,\alpha} \underline{\times} \left(\underline{\nabla \times} \psi_0\right) \\ &+ \sum_{\alpha=1}^{N} r_{\alpha,f} K_{\alpha,f}^{VM} \underline{\frac{D\mathbf{u}_\alpha}{Dt}} - \left[\sum_{\alpha=1}^{N} r_{\alpha,f} K_{\alpha,f}^{VM} \left[\frac{D\mathbf{u_0}}{Dt}\right]\right] \end{aligned} \tag{A.5}$$

A.3 Equação de pressão

A equação de pressão (equação de poisson) é feita totalmente implícita de acordo com a equação abaixo:

$$\left\lfloor \nabla^2 \left(D_p[p]\right)\right\rfloor = \underline{\nabla} \cdot \left(r_{0,f} \psi_0^* + \sum_{\alpha=1}^{N} r_{\alpha,f} \psi_\alpha^*\right) \tag{A.6}$$

MANUSCRITOS

Os manuscritos relativos ao presente trabalho de doutoramento se encontram neste anexo. No primeiro documento, o método DQMoM utilizando momentos generalizados é testado em termos de acurácia e robustez. Para avaliar a robustez, o número de condicionalmento do sistema linear do DQMoM foi analisado com intuito de usá-lo para aprimorar a robustez do mesmo. Neste trabalho, algumas propostas baseadas em otimização foram feitas. No entanto, o custo computacional associado não justifica seu uso. No segundo manuscrito a formulação do D^2uQMoGeM foi derivada. Como já supracitado, vários casos foram analisados e comparados com o método DQMoM tradicional. O D^2uQMoGeM se mostrou mais acurado, porém mais custoso na maioria dos casos. No terceiro documento, dois algoritmos de cubatura adaptativa paralelizados em GPUs foram introduzidos. Os métodos obtiveram aceleração significativa, e podem ser aplicados no controle de erro do D^2uQMoGeM. Finalmente, o manuscrito em preparação, que apresenta a formulação do D^2uQMoGeM-FC, usando os métodos implementados em GPU acoplados com o OpenFOAM® com a modelagem multi-fluido. Abaixo suas respectivas publicações:

- SANTOS, F. P., LAGE, P. L. C., FONTES, C. E. "Numerical aspects of direct quadrature-based moment methods for solving the population balance equation", Brazilian Journal of Chemical Engineering, v. 30, pp. 643 – 656, 09 2013.

- SANTOS, F. P., FAVERO, J. L., LAGE, P. L. C. "Solution of the population balance equation by the direct dual quadrature method of generalized moments", Chemical Engineering Science, v. 101, pp. 663 – 673, 2013.

- SANTOS, F. P., SENOCAK, I., FAVERO, J. L., LAGE, P. L. "Solution of the population balance equation using parallel adaptive cubature on GPUs", Computers & Chemical Engineering, v. 55, pp. 61 – 70, 2013.

- SANTOS, F. P., SENOCAK, I., FAVERO, J. L., LAGE, P. L.SILVA, L. F. L. R." Simulation of Polydispersed Multiphase Flows using Dual-Quadrature-Based Moment Methods on GPUs", submitted to Computers & Chemical Engineering.

ANEXO B

Manuscrito I

Aspectos numéricos associados à robustez do método DQMoM baseado em momentos generalizados.

Numerical aspects of direct quadrature-based moment methods for solving the population balance equation

Abstract

Direct-quadrature generalized moment based methods were analysed in terms of accuracy, computational cost and robustness for the solution of the population balance problems in $[0, \infty]$ and $[0, 1]$ domains. The minimum condition number of the coefficient matrix of their linear system of equations was obtained by global optimization. An heuristic scaling rule from the literature was also evaluated. The results indicate that the methods based on Legendre generalized moments are the most robust for the finite domain problems while the DQMoM formulation that solves for the abscissas and weights using the heuristic scaling rule is the best for the infinite domain problems.

Keywords: Population balance, Numerical methods, DQMoM, Generalized moments, condition number

1. Introduction

There is a need to develop accurate and robust techniques for analysing the dynamics of particle systems. A proper way to obtain the evolution of the particle size distribution involves the solution of the population balance equation (PBE). The PBE is the conservation equation for the number of particles, represented by the mean number density distribution, which is a function of some particle properties, the internal variables, the position in the physical space, the external variables, and time (Ramkrishna, 2000).

There are several numerical methods for solving the PBE. Among them, moment based methods are been continuously developed. The method of moments (MoM) (Hulburt and Katz, 1964) solves the moments of the number density distribution. Considering a mono-variate distribution $f(z, t)$ in a semi-infinite domain, $z \in [0, \infty)$, the essence of this method is to operate the PBE with the moment integral operator, $\int_0^\infty z^k \cdot dz,\ k = 0, 1, \ldots, \infty$, to generate differential equations for the moments of k-order, μ_k. Despite of being simple this method has a closure problem because higher-order moments are usually present in the lower-order moment equations (Friedlander, 2000). McGraw (1997) proposed a methodology to solve this closure problem by approximating the integral terms in the moments equations using a N-point Gauss-Christoffel quadrature rule, that is, the Gaussian quadrature whose weight function is the particle number density function. This quadrature rule can be calculated from the first $2N$ moments (Gordon, 1968; Gautschi, 1968, 2004) and it can exactly integrate polynomials to the $2N - 1$ order. Therefore, it can be used to calculate exactly the first $2N$ moments:

$$\mu_k = \int_0^\infty z^k f(z, t) dz = \sum_{\alpha=1}^{N} \varepsilon_\alpha^k \omega_\alpha, \quad k = 0, 1, \ldots \tag{1}$$

which form a non-linear system of equations that determines the quadrature weights, ω_α, and abscissas, ε_α. The method was called QMoM (Quadrature Method of Moments).

Since the integral approximations derived from the Gauss-Christoffel quadrature can also be obtained by assuming the discrete representation of the distribution function given by

$$f(z,t) \approx \sum_{\alpha=1}^{N} \omega_\alpha(t)\delta[z-\varepsilon_\alpha(t)], \tag{2}$$

Marchisio and Fox (2005) developed the idea of solving directly for the quadrature by substituting the equation (2) into the PBE and then applying the moment integral operator. They chose the solve for the weights and weighted abscissas, $\lambda_\alpha = \omega_\alpha\varepsilon_\alpha$. This method, called DQMoM (Direct QMOM), does not calculate the Gauss-Christoffel quadrature in each time step but it needs an initial discretization of the distribution function, that can be given by the Gauss-Christoffel quadrature.

Several variants of QMoM and DQMoM have been developed. In the following, those that numerically analysed these methods are reviewed.

Alopaeus et al. (2006) proposed the usage of quadrature rules with fixed quadrature points (Fixed-Quadrature MoM, or FQMoM). This approach avoids some robustness and accuracy problems associated to the Gauss-Christoffel quadrature computation. By analysing several problems, they concluded that FQMoM was better than QMoM in accuracy and computational cost.

Fox (2006) applied the DQMoM to bivariate problems including coagulation and sintering. Due to coagulation, very large particle sizes can be produced and he recommended an heuristic rule to reduce the condition number of the DQMoM system of linear equations. The condition number of the matrix of a linear system is the ratio of the largest to the smallest singular value in the singular value decomposition of this matrix. It measures the loss of accuracy in the computation of the linear system solution. If it is low, the system is said to be well-conditioned and if it is large, the system is ill-conditioned. Its reduction by a linear transformation or operator can reflect positively in the stability of the DQMoM solution due to the increased accuracy in its linear system solution. This rule proposed by Fox (2006) basically scales each equation of the linear system by dividing it by an adequate power of the largest abscissa.

Su et al. (2007) proposed the usage of an adjustable factor, s, in QMoM, whose purpose is to improve the robustness in the the Gauss-Christoffel quadrature calculation. Basically, they used fractional moments given by

$$\mu_{k/s} = \int_0^\infty z^{k/s} f(z,t)dz = \sum_{\alpha=1}^{N} \varepsilon_\alpha^{k/s}\omega_\alpha \tag{3}$$

and defined $\tilde{z} = z^{1/s}$ as equivalent abscissas which were then calculated by the product-difference algorithm (Gordon, 1968).

Afterwards, Su et al. (2008) applied the same idea to DQMoM but using an adaptive procedure to choose the value of the adjustable factor, calling the method as Adaptive DQMoM (ADQMOM). The value of the adjustable factor was determined by a search procedure based on the condition number of the ADQMoM system of linear equations.

Attarakih et al. (2009) proposed the sectional QMoM (SQMoM) focusing in reconstructing the distribution function. The domain is divided in sections whose sectional moments are then used to determine a quadrature for each section, as in QMOM. Although the Gauss-Christoffel quadrature was also used, they recommended an equal-weight quadrature with better numerical properties. An interesting advantage of this method is that integral properties are not very sensitive to the reconstructed distribution.

All the direct quadrature moment methods described above used the monomial moment integral operator and the formulation based on the weights and weighted abscissas.

On the other hand, Grosch et al. (2006) presented a generalized framework for the quadrature method of moments that is based on the concepts of generalized moments and coordinate transformations. In analogy to the QMoM, this framework transforms the PBE using the generalized moment integral operator, $\int_0^\infty \phi_k(z) \cdot dz$, where $\phi_k(z)$ may be an orthogonal polynomial (QMoGeM). They formulated the QMOM equations as a differential algebraic system of equations (QMoM-DAE) by solving the QMoM moment equations simultaneously with

$$\mu_k^\phi = \int_0^\infty \phi_k(z) f(z,t) dz = \sum_{\alpha=1}^{N} \phi_k(\varepsilon_\alpha)\omega_\alpha \tag{4}$$

They also derived a method by applying an index reduction procedure to QMoM-DAE, which produced the DQMoM in terms of weights and abscissas when the monomial polynomial basis were used, which is called here as DQMoMa. The framework developed by Grosch et al. (2006) also includes the DQMoM developed by Marchisio and Fox (2005), the QMoM developed by McGraw (1997) and their variations with generalized moments (DQMoGeM). Grosch et al. (2006) tested these variants of QMoM by comparing their solutions of mono-variate PBE problems in semi-infinite domain to those obtained by the Parsival solver (Wulkow et al., 2001), which ensures a specified tolerance by using a hp-adaptive discontinuous finite elements method. Grosch et al. (2006) concluded that the methods based on Laguerre generalized moments were only marginally more robust than the corresponding methods using the monomial moments. They recommended the usage of the DQMoM formulation derived using the monomial moment integral operator and solved in terms of the quadrature weights and abscissas because of its good robustness and low computational cost.

Although Grosch et al. (2006) did analyse the numerical behavior of quadrature moment methods using generalized moments, the analysis was limited to Laguerre generalized moments due to the choice of population balance problems in semi-infinite domains. However, there is an increasing interested in moment methods for problems in finite domains (Strumendo and Arastoopour, 2008; Lage, 2011). Therefore, the main goal of the present work is to analyse the numerical properties of direct quadrature moment methods using generalized moments both in semi-infinite and finite domains with Laguerre and Legendre polynomials, respectively. The formulations that solve either for the abscissas or for the weighted abscissas were also compared.

2. Population balance modeling

Consider the population balance equation with an unique additive internal variable, z, represented by the following form of the PBE:

$$\begin{aligned}\frac{df(z,t)}{dt} = \nu &\int_z^{z_{max}} b(z)P(z|z')f(z',t)dz' - b(z)f(z,t) \\ &+\frac{1}{2}\int_0^z a(z-z',z')f(z-z',t)f(z',t)dz' \\ &-\int_0^{z_{max}} a(z,z')f(z,t)f(z',t)dz' + R(z,t)\end{aligned} \tag{5}$$

where $a(z,z')$ is the aggregation frequency of the particles with property z and z', $b(z)$ is the breakage frequency of particle with the property z and $P(z|z')$ is the daughter probability

distribution function for the breakage of a particle with property z', ν is the mean number of particles formed in the breakage process and R is an additional source term, which can represent, for instance, nucleation or growth. The first and the second terms are, respectively, the birth and death terms for breakage, and the third and the fourth are corresponding terms for aggregation.

3. Direct quadrature methods

Following Marchisio and Fox (2005), the DQMoGeM is derived by substituting equation (2) into the PBE and then applying the generalized moment integral operator. It is then possible to define the following formulations.

3.1. Formulation using weights and weighted-abscissas (DQMoGeM)

$$\frac{d\omega_\alpha}{dt} = \theta_\alpha, \quad \frac{d\lambda_\alpha}{dt} = \varrho_\alpha, \quad \alpha = 1, \cdots, N \tag{6}$$

which must be solved together with the following system of linear equations:

$$\sum_{\alpha=1}^{N} (\phi_k(\varepsilon_\alpha) - \varepsilon_\alpha \phi_k'(\varepsilon_\alpha))\theta_\alpha + \sum_{\alpha=1}^{N} \phi_k'(\varepsilon_\alpha)\varrho_\alpha = S_k, \quad k = 0, \cdots, 2N-1 \tag{7}$$

where S_k is defined below. If $\phi_k(z) = z^k$, the DQMoGeM is reduced to the standard form of DQMoM. Fox (2006) heuristic rule consists of dividing the k equation of equation (7) by ε_{max}^k, where $\varepsilon_{max} = \max_{\forall j} \varepsilon_j$.

3.2. Formulation using weights and abscissas (DQMoGeMa)

$$\frac{d\omega_\alpha}{dt} = \theta_\alpha, \quad \frac{d\varepsilon_\alpha}{dt} = \upsilon_\alpha, \quad \alpha = 1, \cdots, N \tag{8}$$

which must be solved together with the following system of linear equations:

$$\sum_{\alpha=1}^{N} \phi_k(\varepsilon_\alpha)\theta_\alpha + \sum_{\alpha=1}^{N} \omega_\alpha \phi_k'(\varepsilon_\alpha)\upsilon_\alpha = S_k, \quad k = 0, \cdots, 2N-1 \tag{9}$$

If $\phi_k(z) = z^k$, the DQMoGeMa is reduced to what is called here as the DQMoMa.

3.3. Source terms due to aggregation and breakage

Using the Gauss-Christoffel quadrature approximation, the sources terms in the system of linear equations are given by

$$S_k(t) = \int_0^{z_{max}} \phi_k(z) S(z,t) dz = B_k^a(t) - D_k^a(t) + B_k^b(t) - D_k^b(t) + R_k(t) \tag{10}$$

where

$$B_k^a(t) = \int_D B^a(z,t)\phi(z)dz = \frac{1}{2}\sum_{\beta=1}^{N}\sum_{\alpha=1}^{N} \phi_k(\varepsilon_\alpha + \varepsilon_\beta) a_{\alpha\beta}\omega_\alpha\omega_\beta \tag{11}$$

$$D_k^a(t) = \int_D D^a(z,t)\phi(z)dz = \sum_{\beta=1}^{N}\sum_{\alpha=1}^{N}\phi_k(\varepsilon_\alpha)a_{\alpha\beta}\omega_\alpha\omega_\beta \tag{12}$$

$$B_k^b(t) = \int_D B^b(z,t)\phi(z)dz = \sum_{\alpha=1}^{N} b_\alpha\phi_k(\varepsilon_\alpha)\omega_\alpha \tag{13}$$

$$D_k^b(t) = \int_D D^b(z,t)\phi(z)dz = \sum_{\alpha=1}^{n} b_\alpha\nu_\alpha\omega_\alpha\Pi_\alpha^k \tag{14}$$

$$R_k(t) = \int_D R(z,t)\phi(z)dz \tag{15}$$

$$\Pi_\alpha^k = \int_0^{\varepsilon_\alpha}\phi_k(z)P(z|\varepsilon_\alpha)dz, \quad \alpha = 1,\cdots,N \tag{16}$$

If $\phi_k(z) = z^k$ these terms became those of DQMoM and QMoM (Marchisio and Fox, 2005). It is important to point out that equations (15) and (16) are calculated analytically or by an adaptive quadrature with error control.

4. Numerical Procedure

The implementation of the PBE solution was performed in C and FORTRAN. The source codes were compiled with gfortran and g++ using the -03 high optimization flag with double precision variables. The work was carried out in an Intel(R) Core(TM)2 Quad CPU Q6600 2.40GHz processor with Ubuntu 10.04 Linux operating system.

The integration of equations (15) and (16) were performed using the adaptive quadrature AUTOQUAD routine (Lage, 1992). The ORTHOPOL package (Gautschi, 1994) was used to calculate the recursion coefficients of the orthogonal polynomials. For the initial conditions, the modified Chebyshev algorithm was used to obtain the Gauss-Christoffel weights and abscissas from the initial values of the generalized moments (Gautschi, 1994).

The time integration was performed using the DASSLC package (Secchi, 2007), an adaptive time step and variable order integrator based on backward differentiation formulas, with required relative and absolute tolerances equal to 10^{-10}. This made the time integration error much smaller than the quadrature error embodied in these quadrature moment methods.

An evaluation of the computation time for all methods was performed for 1.0 second of simulation. The *clock* intrinsic subroutine of g++ was used to obtain the elapsed CPU time in seconds with $0.01s$ of accuracy.

The computation of the condition number of the coefficient matrix, $\mathbf{A}$, of the system of linear equations given by equation (7) or equation (9) was carried out by DGTRF and DGECON routines of the LAPACK package (Anderson et al., 1999).

In order to evaluate if the robustness of the solution of the linear system of equations can be improved, a diagonal pre-conditioner, $\mathbf{P} = diag(\mathbf{p})$ was applied to the linear system given by equation (7) or (9) and the minimum condition number (Marechal and Ye, 2009) of the resulting coefficient matrix was obtained by defining the following minimization problem:

$$CN_{min} = \min_{\mathbf{p}\in\Re^{2N}} \left[CN\left(\mathbf{PA}\right)\right] \tag{17}$$

This minimization was performed by NLOPT package (Johnson, 2008). The results obtained for the minimum condition number of DQMoM were compared against the condition number obtained using the heuristic rule proposed by Fox (2006).

5. Results and discussion

The methods present in section 3 were applied to four test cases for which analytical solutions are known using the two formulations derived using the monomial (DQMoM and DQMoMa) and generalized moment (DQMoGeM and DQMoGeMa) integral operators.

The first two test cases used the population balance problem in a semi-infinite domain described in Appendix A and their solution may use the monic Laguerre generalized moments. Case I is aggregation dominant with $\Phi(\infty) = 0.5$ and case II is breakage dominant with $\Phi(\infty) = 2$. Both cases were solved up to $t = 1$ where steady-state is basically reached. The last two test cases are for a pure breakage problem in a finite domain as given in Appendix B and they may be solved using monic shifted Legendre generalized moments. Case III defines $\gamma = 2$ and it is easier to be solved than case IV with $\gamma = 1/3$. Cases III and IV were solved up to $t = 5$.

These problems are sufficient simple to allow all methods to convergence for relatively small N values. In other words, if a relatively small N were used, the quadrature errors are not too large to generate two abscissas close to each other during integration. This is a well-known problem that would render the matrix of the system of linear equations singular which would make the methods comparison difficult. It should be pointed out that the breakage and aggregation kernels do not affect directly the coefficient matrix of the linear system of equations in these methods.

5.1. Convergence analysis

Since the analytical solutions are known, the relative error of each regular moment can always be calculated by

$$X_k = \left| \frac{\mu_k^{(e)} - \mu_k}{\mu_k^{(e)}} \right| \tag{18}$$

where $\mu_k^{(e)}$ are the analytical moments and μ_k are the moments reconstructed from the numerical solution using equation (1). Since the first six moments are used in the solution for $N \geq 3$, the convergence of the quadrature moment methods was evaluated using their mean square relative error:

$$X_{RMS,6} = \sqrt{\frac{1}{6} \sum_{k=0}^{5} X_k^2} \tag{19}$$

The regular moments were used to evaluate the convergence even for the methods that use generalized moments.

Figure 1 shows the time evolution of $X_{RMS,6}$ for cases I, II, III and IV solved with $N = 5$. All cases show initial $X_{RMS,6}$ values of the order of the required integration tolerances but they increase during the integration due to the accumulation of quadrature error along the time integration. Nevertheless, at the end of the simulations, although the final $X_{RMS,6}$ values vary from case to case, the errors are still acceptable. As expected, the largest error is for case IV, due to the fact that the breakage frequency in this case does not belong to the space of polynomials, which increases the quadrature error.

The $X_{RMS,6}$ values in the solution of all cases by all methods are compared in Figure 2 for several N values at the end of the simulations. Two kinds of convergence behavior can be seen in Figure 2 that are exemplified for the simulations of case III. The first behavior can be seen in DQMoGeM and DQMoGeMa simulations, which converge with a almost constant

rate up to a $X_{RMS,6}$ value that is about 1-2 orders of magnitude larger than the tolerance in time integration used in DASSLC. Then, for large N values, the $X_{RMS,6}$ value remains almost constant indicating that the accumulated time integration error becomes more important than the quadrature error. The second kind of behavior can be seen in the DQMoM and DQMoMa simulations for case III, which converges with a almost constant rate up a N value where the convergence rate decreases and, sometimes, the $X_{RMS,6}$ value even increases. For even larger N values, the solution could not be completed because the DASSLC routine could not perform the integration with the required local accuracy due to the numerical error in the solution of the linear system of the moment method. In other words, both behaviors can be explained by the following reasoning. If the linear system of the direct quadrature moment method is well-conditioned, the convergence rate is almost constant until N increases up to a point when the time integration error became larger than the quadrature error, when $X_{RMS,6}$ becomes almost constant. However, if the the linear system becomes ill-conditioned, the numerical error deteriorates the convergence or even make the DASSLC routine to give up trying to perform the integration. This lack of robustness caused by the numerical error in the linear system solution is analysed in section 5.3.

5.2. Computational cost

Figure 3 shows the computational time for several values of N for all methods and for all test cases. Again, two behaviors can be perceived and they are again exemplified for case III. The first one can be seen in the DQMoGeMa solution, for which the computation time of the simulation steadily increases with N. The second behavior occurs for all other methods for case III, but the DQMoM and DQMoMa simulations are better examples. In this behavior, the computation time versus N value curve suffers a strong slope change at some N value. Again, this abnormal change of slope are related to the loss of robustness, which makes the DASSLC routine spend much more time to achieved the required tolerance.

It is clear from Figure 3 that, before losing robustness, the DQMoM and DQMoMa are faster than the generalized moment methods. In infinite domain problems, they are 10 times faster but, for the finite domain problems, the generalized moment methods are only 3-4 times slower.

5.3. Robustness analysis

It is usually considered that the loss of robustness in these methods comes from the numerical error embodied in the solution of the linear system of equations, equations (7) and (9), as its matrix becomes singular. For some problems, this happens when two abscissas become very close to each other due to error accumulation during the solution. For this case, there are some solutions, one of these is the usage of fractional moment operator with an adapted parameter to create a different linear system with a better condition number (Su et al., 2008). Other solution is to keep the same moment integral operator and scale the equations (Fox, 2006). Here we analysed this last solution in a systematic way.

The condition number of the coefficient matrix of the unscaled linear system of equations, equations (7) and (9), was evaluated for each simulation that could be integrated up to its final time value and exactly at this instant. The minimum condition number using a diagonal pre-conditioner was also obtained at this time to verify if the solution can be improved by scaling.

Figure 4 shows the condition number obtained for all test cases and methods without any scaling and for different N values. Regarding the two formulations of the methods, this figure

clearly shows that there is no significant difference in the condition number values for both formulations of the same method. For case I, there is not much difference among the condition numbers for all methods. On the other hand, for all other cases, especially for the finite domain problems, the generalized moment methods have smaller condition numbers than those for the monomial moment methods. For case I, although the values of the condition numbers are close for all methods, DQMoMa has shown be be more robust than the other methods. For case II, it was as robust as DQMoGeMa, although its condition number is 10^8 times larger than that for DQMoGeMa at $N = 11$. This behavior was also found by Grosch et al. (2006) that compared methods using monomial and Laguerre generalized moments. For the finite domain problems, the condition number values are $10^2 - 10^3$ times smaller for the Legendre generalized moment methods. This fact is reflected in the method robustness, as cases III and IV could be solved by DQMoGeM and DQMoGeMa using N values above 30 while DQMoM and DQMoMa could not be employed above 10 quadrature points. This was not observed by Grosch et al. (2006) because they did not solved problems in finite domains.

In order to verify the possible robustness gain by the scaling of the coefficient matrix, all test cases were solved by all methods using a diagonal pre-conditioner obtained at time zero by an optimization procedure. Figure 5 shows the ratio between the minimum condition number and the condition number for all test cases and methods and for different N values at the end of the simulations. For the problems in semi-infinite domain, the reduction of the matrix condition number by scaling increases almost exponentially, and a $10^{10} - 10^{15}$ reduction is achieved for $N = 10$. For the finite domain problems, the condition number was only sligthly affected by the optimization for DQMoM and DQMoMa. However, the values of the minimum condition number were much smaller than the original CN values for both DQMoGeM and DQMoGeMa, which are related to the use of the Legendre polynomial moments. The ratio between the minimum and the original condition numbers drops very fast with N. The robustness gain was large, as the maximum N value in the solution of cases III and IV raised from around 30 to above 50. Therefore, the optimization procedure applied at $t = 0$ was a very effective way of improving the robustness of the quadrature methods based on Legendre polynomial moments applied to finite domain problems.

A sensitive analysis was carried out to verify if the CN_{min}/CN behavior depends on the breakage kernel parameters. The parameters $\Phi(\infty)$ and γ of cases I and III were modified to $\Phi(\infty) = 0.1$ and $\gamma = 3$, generating cases V and VI, respectively. The corresponding results for CN_{min}/CN are shown in Figure 6. It can be seen that these perturbations in the parameters do not change the CN_{min}/CN behavior observed in Figures 5(a) and 5(c). Besides, the robustness improvement was the same. It seems to be independent on the dynamics of the problem.

The heuristic rule of Fox (2006) was extended for all methods and for cases I and III. Figure 7 shows the ratio between the minimum condition number and the condition number calculated using this heuristic rule evaluated at the end of the simulations, using all methods and for different N values.

For case I, Figure 7(a) shows that the heuristic rule proposed by Fox (2006) reduces significantly the condition number of the original linear system of equation of DQMoM and DQMoMa as the CN_{min}/CN_{Fox} value is almost always between 0.1 and 1 for these methods. The rule also worked reasonably well for the Laguerre generalized moment methods ($10^{-5} \lesssim CN_{min}/CN_{Fox} \lesssim 0.3$), probably because monic polynomials were used and the semi-infinite domain generates large abscissa values that make the largest order polynomial term much larger than the others. In these problems in semi-finite domains, the robustness gain obtained was not too impressive, as both scaling rules allowed to obtain the solution for only

one more quadrature point for case I solved by DQMoM and DQMoMa.

As can be seen in Figure 7(b), Fox's rule did not work at all for case III using Legendre polynomial moments. The behavior of CN_{min}/CN_{Fox} for case III was basically equal to that of CN_{min}/CN shown in Figure 6(b). This was expected as the largest abscissa is not related to the value of $\phi_k(\epsilon_\alpha)$ for the Legendre polynomials in the $[0, 1]$ domain. Besides, no robustness gain was obtained for the finite domain problems for all methods using Fox's heuristic scaling rule. On the other hand, as commented above, the optimization procedure applied to this case increased the robustnesses of DQMoGeM and DQMoGeMa solutions. This behavior was also observed for cases V and VI.

Figure 8 shows the ratio between the computational time of the simulations using the minimum condition number and the Fox's heuristic scaling rule for cases I and III and for several N values. This ratio decreases with N because the simulation time increases faster with N than the optimization time. For case I, both methods have the same robustness but the Fox's heuristic scaling rule is computationally much cheaper. This is expected to happen for all semi-infinite domain problems for which the computational cost of the optimization procedure used to calculate CN_{min} at $t = 0$ is a large fraction of the total computational time. For case III, the scaling using the optimization procedure for CN_{min} determination is more robust than the Fox's heuristic scaling rule and Figure 8(b) shows that its additional computational cost becomes less important as N increases. The strong slope change in the curves for DQMoM and DQMoMa are related to the loss of robustness, which makes the simulation time becomes larger than the optimization time. Therefore, for Legendre moment methods applied to problems in finite domains, it is expected that the computation of CN_{min} is justified due to the increased robustness, specially when the simulation time is quite large, as in PB-CFD simulations.

6. Conclusions

Direct quadrature methods based on regular and orthogonal polynomial moments were analysed using the weighted-abscissa and abscissa formulations by solving population balance problems in semi-infinite and finite domains. The methods were compared regarding their accuracy, computational cost and robustness.

The methods based on regular moments were always the fastest. However, they were the least robust for the finite domain test cases. For these problems, the Legendre generalized moment methods were the most robust. The DQMoM formulation based on the abscissas (DQMoMa) showed to be more robust than the weighted-abscissa formulation (DQMoM) for the problems in the semi-infinite domain.

The best scaling of the linear system of equations of all methods were obtained by global minimization of the condition number of the transformed coefficient matrix using a diagonal preconditioning matrix. The results showed only a small improvement over Fox's heuristic rule for the methods based on regular moments applied to problems in the semi-infinite domain. In this case, both scaling methods allowed the PBE to be solved with only one extra quadrature point. For the methods based on regular moments, the optimization procedure did not give a robustness gain when applied to problems in the finite domain. On the other hand, for the methods based on Legendre polynomial moments, the scaling by optimization yielded a large reduction in the condition number, allowing the solution for a much larger number of quadrature points.

The heuristic rule of Fox (2006) had almost the same performance as the scaling obtained by optimization for the problems in the semi-infinite domain, but with less computational cost. However, for problems in the finite domain, the scaling by minimizing the condition number was much better, inproving the method robustness. A sensitive analysis has shown that the results seem to be independent of the problem dynamics.

Therefore, it can be concluded that, for problems in the semi-infinite domain, DQMoMa should be used with Fox (2006) scaling rule. This conclusion agrees with Grosch et al. (2006), who found that DQMoMa is better than other QMoM variations studied by them for semi-finite domain problems. However, for problems in finite domains, the generalized Legendre moment methods should be used with the scaling obtained by the minimization of the coefficient matrix condition number, specially if a large number of quadrature points were required or when the simulation time is large.

7. Acknowledgments

Paulo L. C. Lage and Fabio P. Santos acknowledge the financial support of CNPq, grants nos. 302963/2011-1, 476268/2009-5 and 140794/2010-7.

Nomenclature

$\mathbf{A}$	coefficient matrix
a	aggregation frequency
B	birth term
b	breakage frequency
CN	condition number
CT	computational time in seconds
D	death term
f	number density distribution function
H	Heaviside step function
N	number of quadrature points
P	probability density function of breakage of particles
$\mathbf{P}$	preconditioning matrix
$\mathbf{p}$	pre-conditioner diagonal elements
R	source term
s	adjustable factor
S_k	moment of k order of source term

t	time
X	relative error
z	additive internal variable

Greek letters

γ	breakage kernel parameter in the finite domain problem
δ	Dirac delta function
ε	abscissa
λ	weighted abscissa
μ_k	moment of k order
ν	number of particle formed in breakage
ω	weight function
Φ	the ratio $\mu_0(t)/\mu_0(0)$
$\Phi(\infty)$	value of $\Phi(t)$ when $t \to \infty$
ϕ_k	k order polynomial
Π	moment of k order of P
θ	source in weight equation
υ	source in abscissas equation
ϱ	source in weighted abscissa

Superscripts

a	aggregation
b	breakage
e	exact solution of standard moment

Subscripts

α	quadrature point
β	abscissa
ϕ	generalized moment
Fox	using Fox's heuristic scaling rule
OPT	using the optimization procedure

max	maximum
min	minimum
$RMS, 6$	mean square relative error of the first six monomial moments

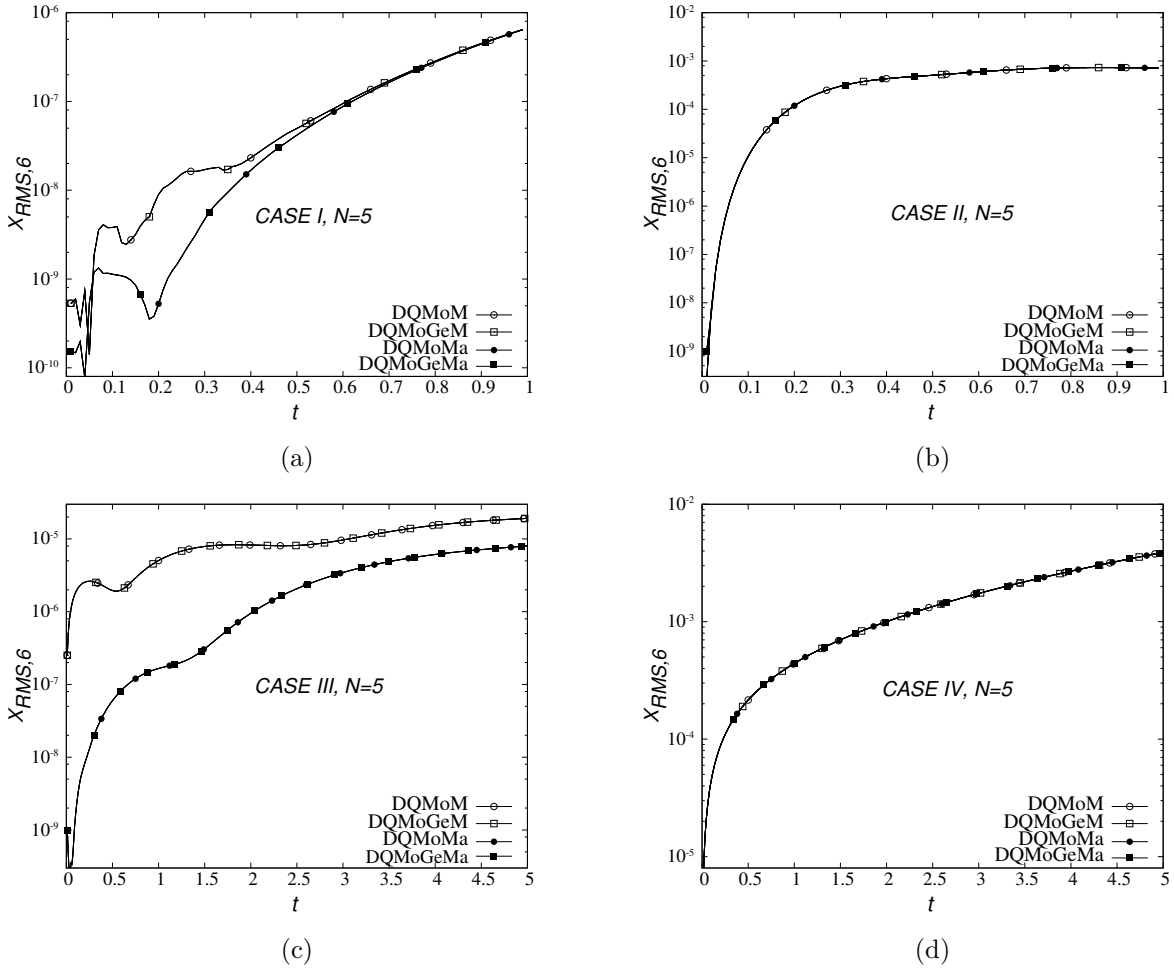


Figure 1: Behavior of $X_{RMS,6}$ during simulation using $N = 5$ for: (a) case I, (b) case II, (c) case III, and (d) case IV.

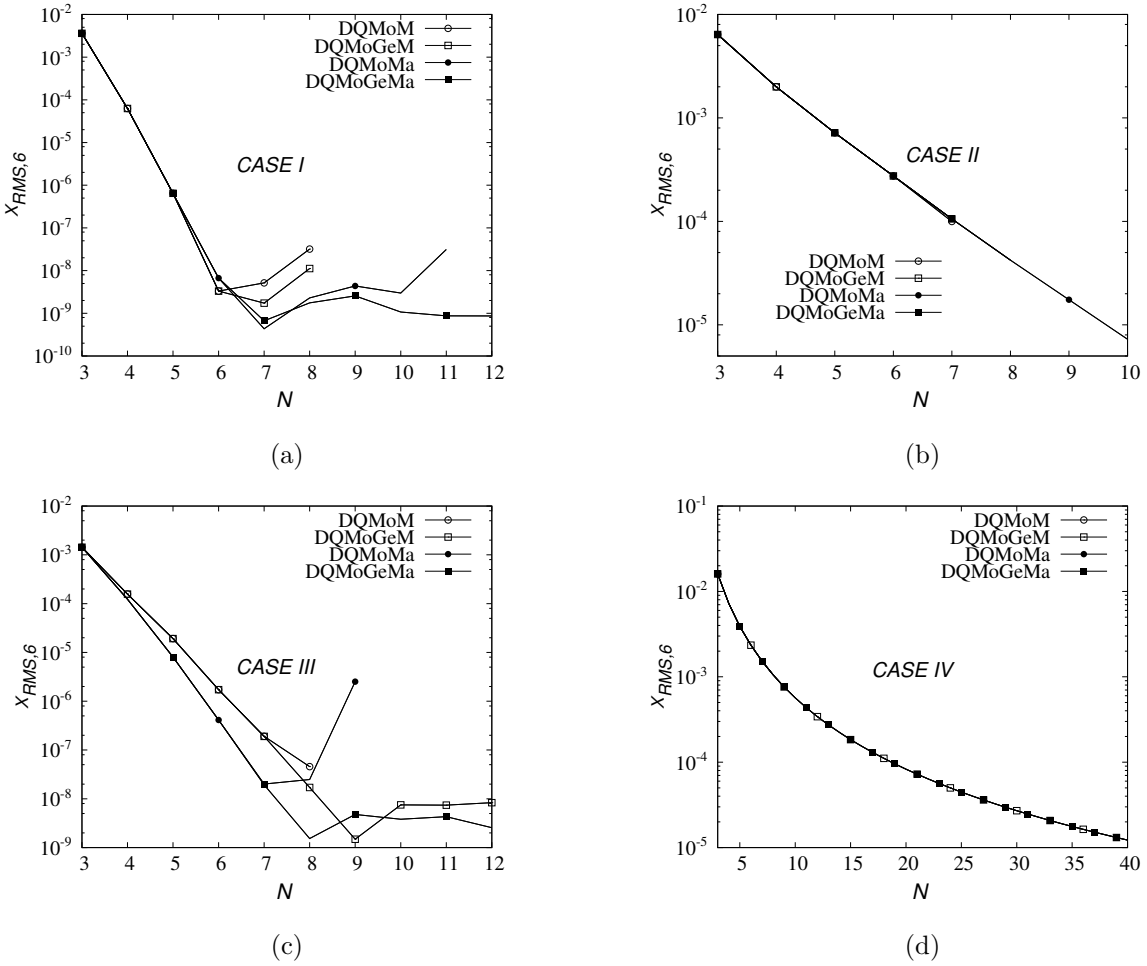


Figure 2: Values of $X_{RMS,6}$ at the end of simulation using several N values for: (a) case I, (b) case II, (c) case III and (d) case IV.

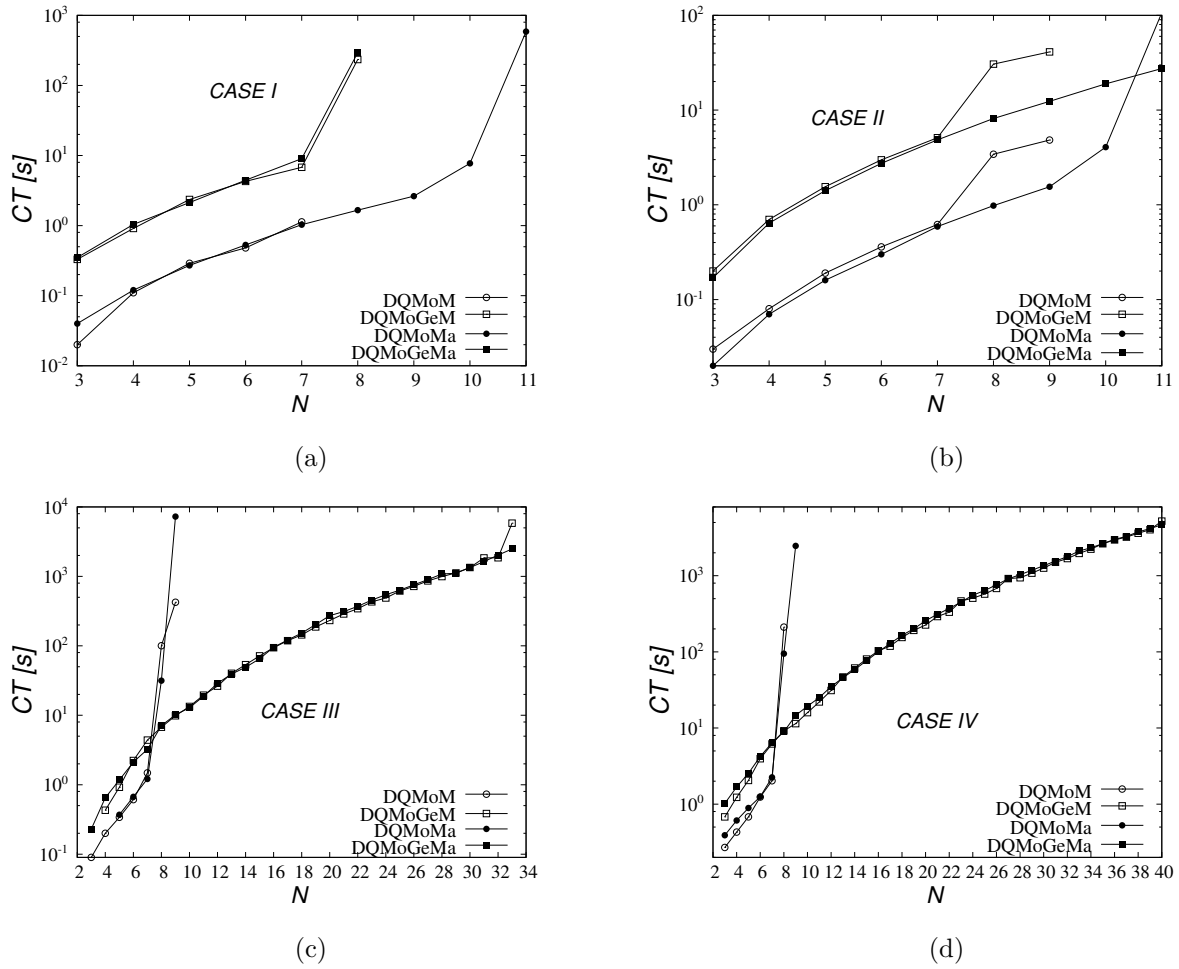


Figure 3: Computational time using several N values for: (a) case I, (b) case II (c) case III and (d) case IV.

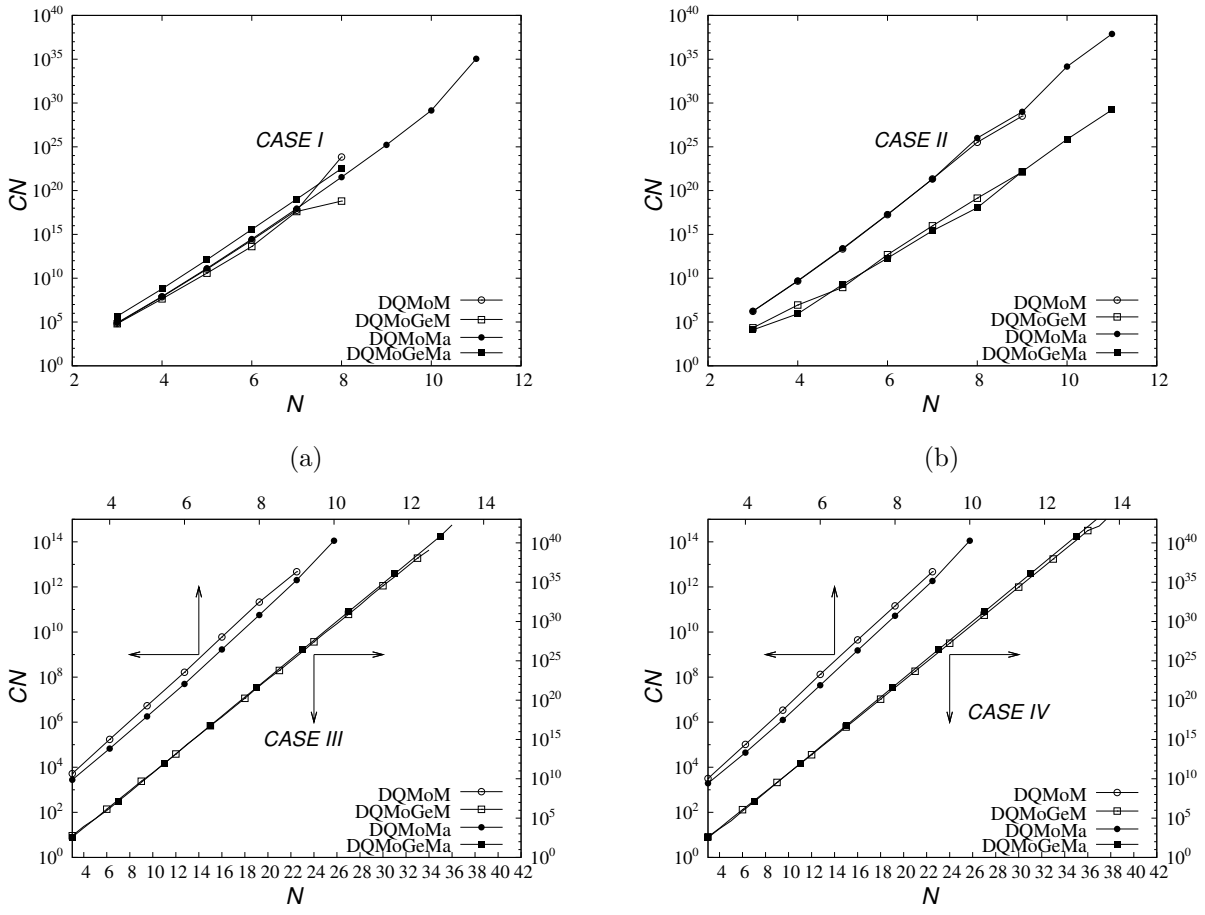


Figure 4: Condition number of the coefficient matrix of all methods using several N values for: (a) case I, (b) case II (c) case III and (d) case IV.

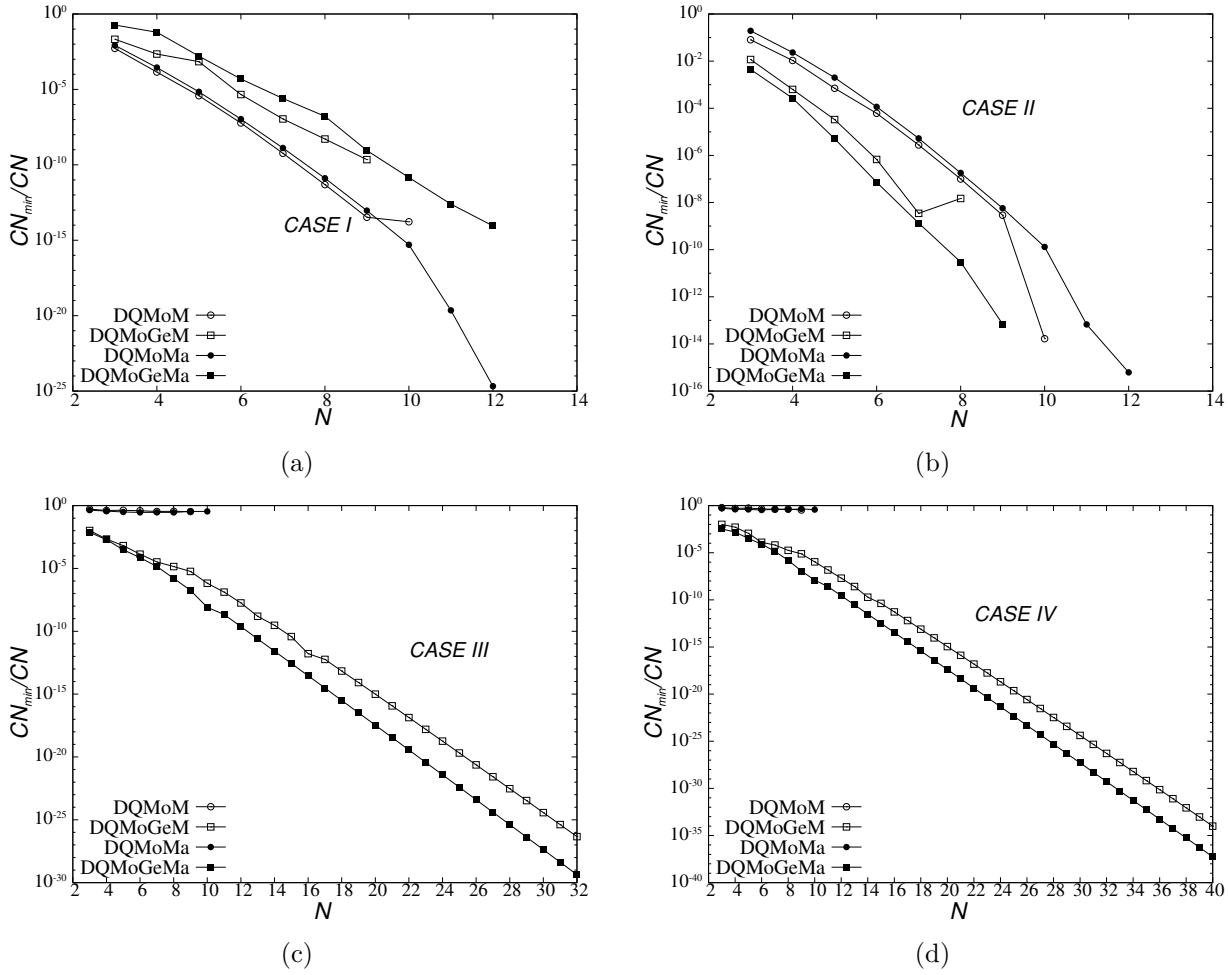
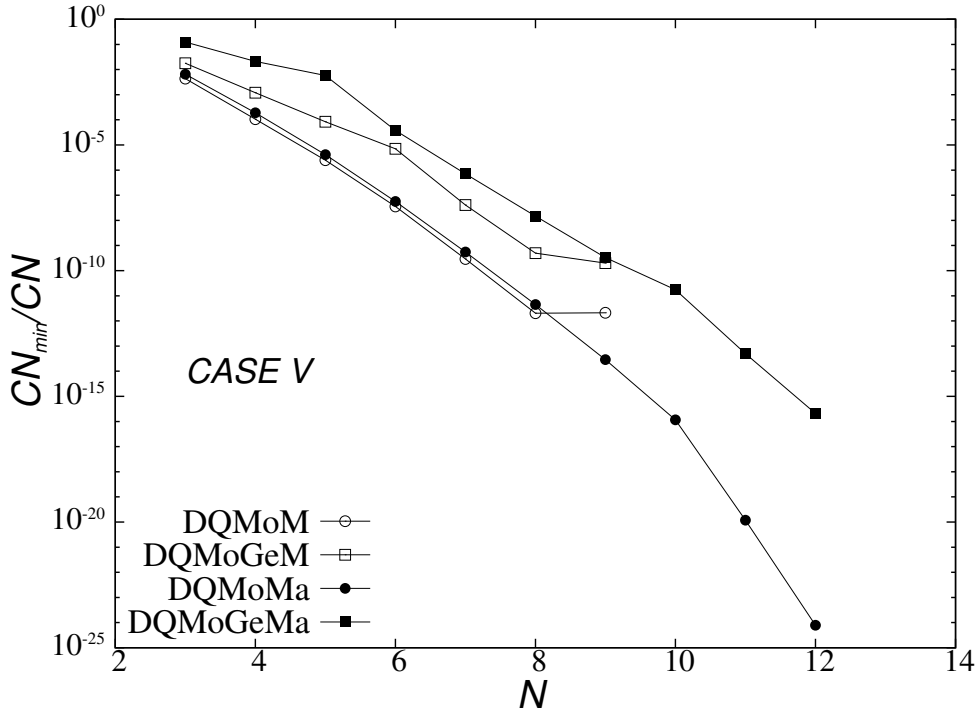
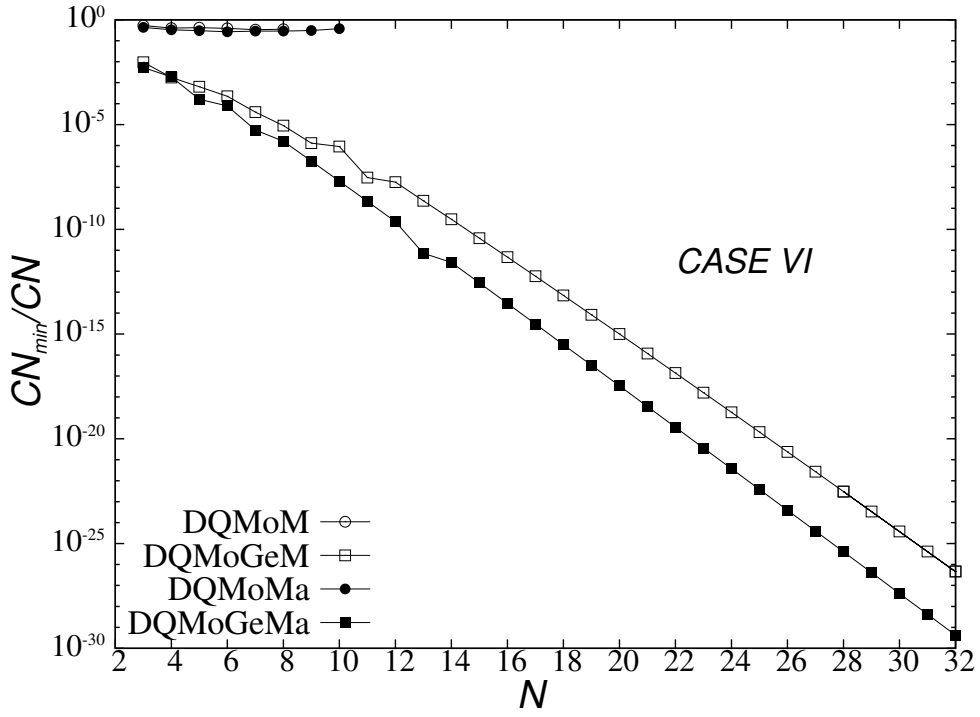


Figure 5: Reduction of the condition number of the coefficient matrix of all methods by minimization using several N values for: (a) case I, (b) case II (c) case III and (d) case IV.

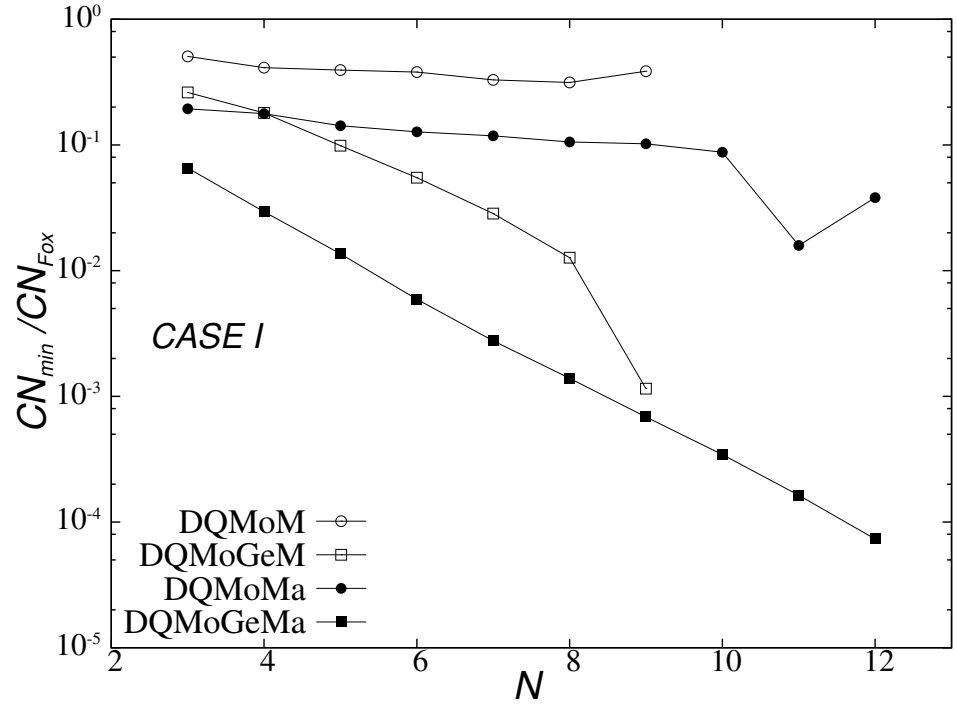


(a)

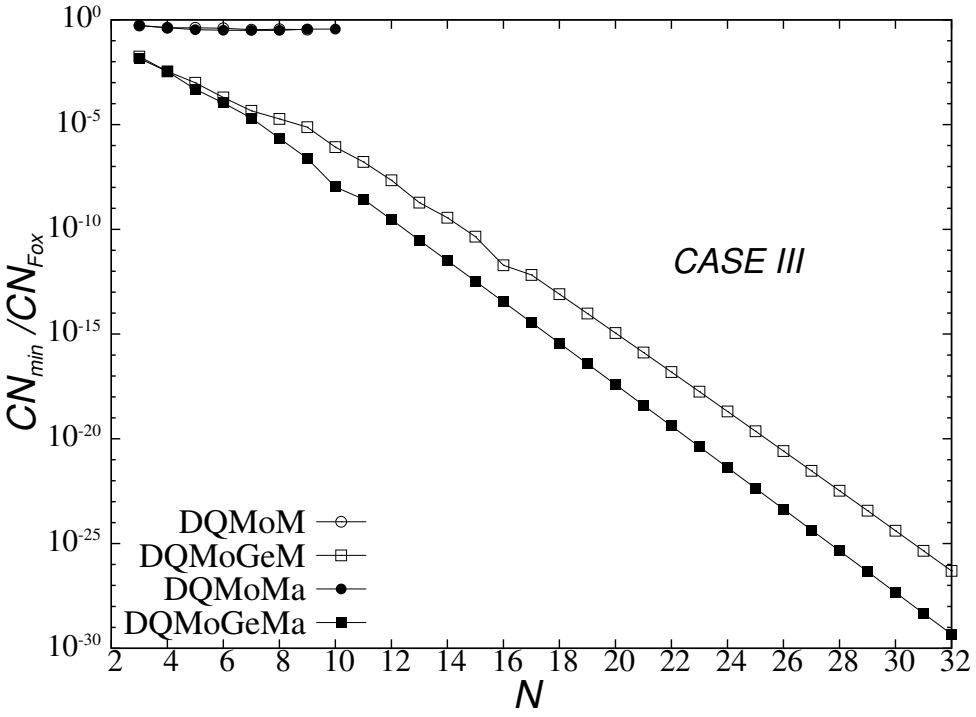


(b)

Figure 6: Reduction of the condition number of the coefficient matrix of all methods by minimization using several N values for: (a) case V and (b) case VI (sensitivite analysis).

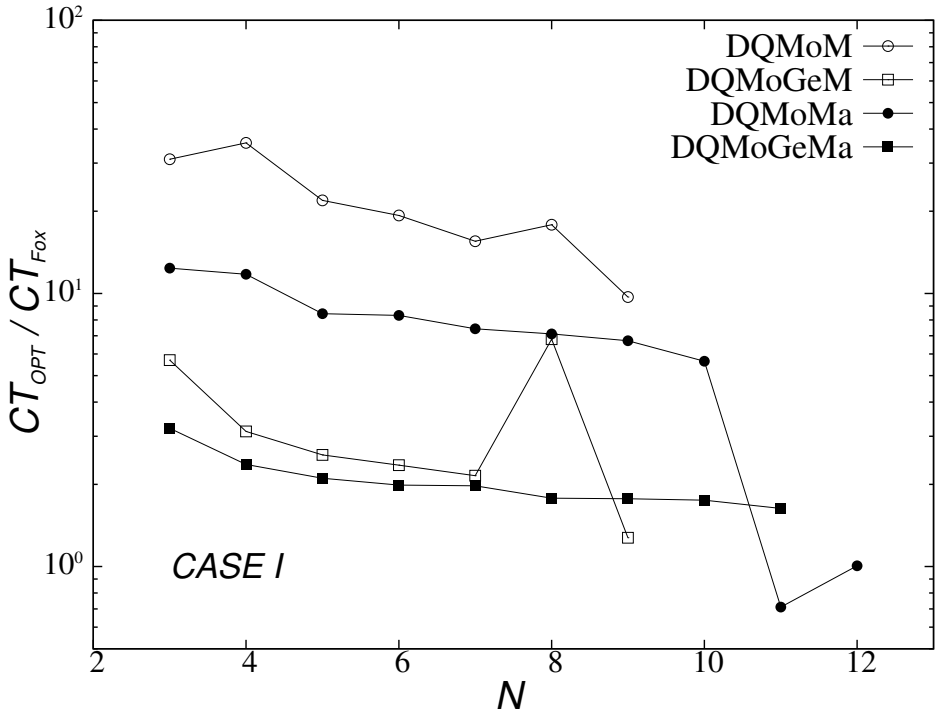


(a)

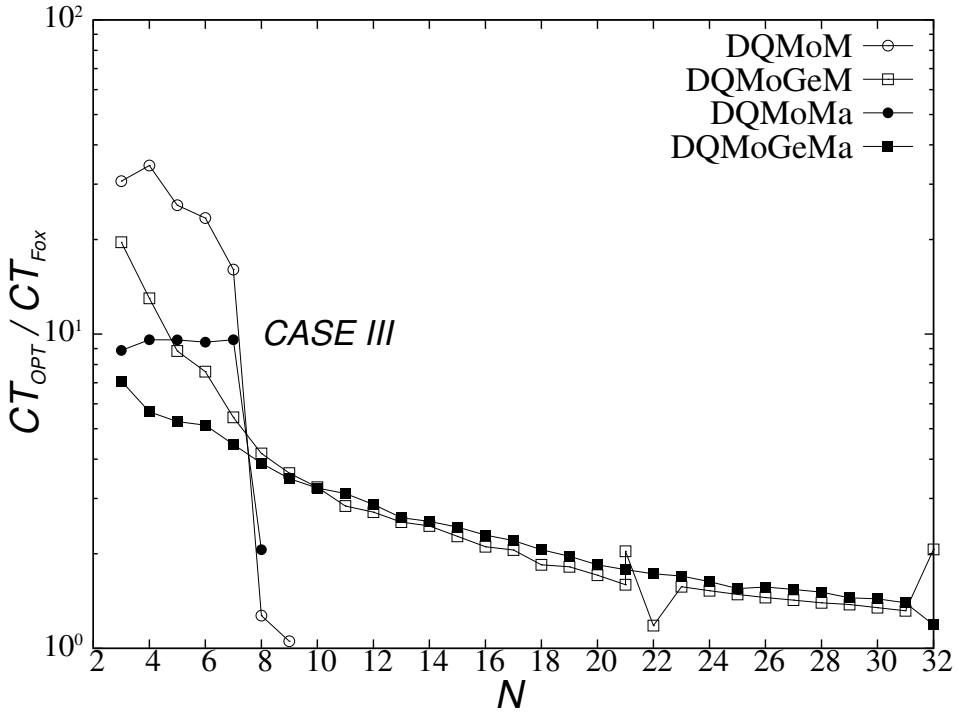


(b)

Figure 7: Reduction of the condition number of the coefficient matrix of all methods by minimization using several N values for: (a) case I and (b) case III.



(a)



(b)

Figure 8: Ratio between the computational time of the simulations using the minimum condition number and the condition number obtained by Fox (2006) rule for all methods using several N values for: (a) case I and (b) case III.

Appendix A. Population balance problem in semi-infinite domain

The problem proposed by Patil and Andrews (1998) for which McCoy and Madras (2003) developed an analytical solution was chosen to analyse the methods. The analytical solution is known for the initial condition:

$$f(z,0) = e^{-z} \tag{A.1}$$

and for the aggregation frequency, breakage frequency and daughter distribution function defined as:

$$b(z) = Lz, \quad a(z',z) = K \quad P(z|z') = \frac{H(z'-z)}{z'} \tag{A.2}$$

where K was assumed to be 1 and L is a constant calculated from a given $\Phi(\infty)$, defined below. The analytical solution is given by (McCoy and Madras, 2003):

$$f(t,z) = \Phi^2(t)e^{-z\Phi(t)}, \quad \Phi(t) = \Phi(\infty)\left[\frac{1+\Phi(\infty)\tanh(\Phi(\infty)\frac{t}{2})}{\Phi(\infty)+\tanh(\Phi(\infty)\frac{t}{2})}\right] \tag{A.3}$$

where $\Phi(\infty)$ is the value of $\Phi(t)$ in the steady-state which is given by

$$\Phi(\infty) = \sqrt{\frac{2L}{K}}\frac{\sqrt{\mu_1(0)}}{\mu_0(0)} \tag{A.4}$$

The regular moments are given by

$$\mu_k(t) = \left[\frac{\Phi(\infty)+\tanh(\Phi(\infty)\frac{t}{2})}{\Phi(\infty)[1+\Phi(\infty)]}\right]^{k-1}\Gamma(1+k) \tag{A.5}$$

which, for $t = 0$, simplifies to

$$\mu_k(0) = \Gamma(1+k) \tag{A.6}$$

Appendix B. Population balance problem in finite domain

Lage (2011) defined a pure binary breakage problem using

$$b(z) = z^\gamma, \quad P(z|z') = H(z'-z)/z' \tag{B.1}$$

For a given value of γ, a source term can be define in order to the following analytical solution be valid

$$f(z,t) = 2 - e^{-t}, z \in [0,1] \tag{B.2}$$

whose moments are given by

$$\mu_k = \frac{2-e^{-t}}{k+1} \tag{B.3}$$

For an integer value of γ, the breakage frequency $b(z)$ belongs to a finite dimensional polynomial space. Two cases were analysed with their corresponding PBE source terms:

- $\gamma = 2 \Longrightarrow R(z,t) = 2z^2(2-e^{-t}) - 2(1-e^{-t})$, and
- $\gamma = \frac{1}{3} \Longrightarrow R(z,t) = 7e^{-t} - 12 + 7(2-e^{-t})z^{1/3}$.

References

Alopaeus, V., Laakkonen, M., and Aittamaa, J. Numerical solution of moment-transformed population balance equation with fixed quadrature points. *Chemical Engineering Science*, *61*, 4919 – 4929 (2006).

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. *LAPACK Users' Guide*. (3rd ed.). Philadelphia, PA: Society for Industrial and Applied Mathematics (1999).

Attarakih, M. M., Drumm, C., and Bart, H.-J. Solution of the population balance equation using the sectional quadrature method of moments (SQMOM). *Chemical Engineering Science*, *64*, 742 – 752 (2009).

Fox, R. Bivariate direct quadrature method of moments for coagulation and sintering of particle populations. *Journal of Aerosol Science*, *37*, 1562 – 1580 (2006).

Friedlander, S. K. *Smoke, dust and haze: Fundamentals of aerosol dynamics (2nd ed.)*. Oxford University Press. (2000).

Gautschi, W. Construction of Gauss-Christoffel quadrature formulas. *Mathematics of Computation*, *22*, 251–270 (1968).

Gautschi, W. Algorithm 726: ORTHPOL: a package of routines for generating orthogonal polynomials and gauss-type quadrature rules. *ACM Transactions on Mathematical Software*, *20*, 21–62 (1994).

Gautschi, W. *Orthogonal Polynomials - Computation and Approximation*. Oxford Science (2004).

Gordon, R. Error bounds in equilibrium statistical mechanics. *Journal od Mathematical Physics*, *9*, 655 – 663 (1968).

Grosch, R., Marquardt, W., Briesen, H., and Wulkow, M. Generalization and numerical investigation of qmom. *AICHE J*, *53*, 207 – 227 (2006).

Hulburt, H., and Katz, S. Some problems in particle technology: A statistical mechanical formulation. *Chemical Engineering Science*, *19*, 555 – 574 (1964).

Johnson, S. G. The NLopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt (2008).

Lage, P. L. C. *Vaporização de Gotas Multicomponentes em Campos Convectivos e Radiantes*. Ph.D. thesis Programa de Engenharia Química, COPPE/UFRJ (1992).

Lage, P. L. C. On the representation of QMOM as a weighted-residual method the dual-quadrature method of generalized moments. *Computers & Chemical Engineering*, *35*, 2186 – 2203 (2011).

Marchisio, D. L., and Fox, . R. O. Solution of population balance equations using the direct quadrature method of moments. *Journal of Aerosol Science*, *36*, 43 – 73 (2005).

Marechal, P., and Ye, J. J. Optimizing condition number. *SIAM journal on optimization*, *20*, 935 – 947 (2009).

McCoy, B. J., and Madras, G. Analytical solution for a population balance equation with aggregation and fragmentation. *Chemical Engineering Science*, *58*, 3049 – 3051 (2003).

McGraw, R. Description of aerosol dynamics by the quadrature method of moments. *Aerosol Science and Technology*, *27*, 255 – 265 (1997).

Patil, D. P., and Andrews, J. R. G. An analytical solution to continuous population balance model describing floc coalescence and breakage – a special case. *Chemical Engineering Science*, *53*, 599 – 601 (1998).

Ramkrishna, D. *Population Balance - Theory and Applications to Particulate Systems in Engineering*. Academic Press, San Diego (2000).

Secchi, A. *DASSLC: User's Manual, a Differential-Algebraic System Solver*. Technical Report UFRGS, Porto Alegre, RS/Brazil (2007). http://www.enq.ufrgs.br/enqlib/numeric/DASSLC.

Strumendo, M., and Arastoopour, H. Solution of PBE by MOM in finite size domains. *Chemical Engineering Science*, *63*, 2624 – 2640 (2008).

Su, J., Gu, Z., Li, Y., Feng, S., and Xu, X. Y. Solution of population balance equation using quadrature method of moments with an adjustable factor. *Chemical Engineering Science*, *62*, 5897 – 5911 (2007).

Su, J., Gu, Z., Li, Y., Feng, S., and Xu, X. Y. An adaptive direct quadrature method of moment for population balance equations. *AICHE J*, *54*, 2872 – 2887 (2008).

Wulkow, M., Gerstlauer, A., and Nieken, U. Modeling and simulation of crystallization processes using Parsival. *Chemical Engineering Science*, *56*, 2575 – 2588 (2001).

ANEXO C

Manuscrito II

Formulação, implementação e avaliação do método D^2uQMoGeM.

Solution of the Population Balance Equation by the Direct Dual Quadrature Method of Generalized Moments[☆]

F. P. Santos, J. L. Favero, P. L. C. Lage*

Programa de Engenharia Química COPPE, Universidade Federal do Rio de Janeiro, PO Box 68502, Rio de Janeiro, RJ, 21941-972, Brazil

Abstract

The Direct Dual Quadrature Method of Generalized Moments (D^2uQMoGeM) was formulated for the solution of the population balance equation. It mixes the properties of the Direct Quadrature Method of Moments (DQMoM) and the Dual Quadrature Method of Generalized Moments (DuQMoGeM). The weights and weighted abscissas are tracking directly as in DQMoM and the quadrature errors are controlled by an adaptive quadrature as in DuQMoGeM. The D^2uQMoGeM was implemented and tested for several different problems with analytical solutions. It was shown to be more accurate than DQMoM with a reasonable increase in computational time.

Keywords: Population balance, Numerical methods, Direct Quadrature Methods of Moments, Generalized moments, Multiphase flow, Particulate systems

1. Introduction

Polydisperse multiphase flows are present in several industry processes. These flows can be modeled by a mesoscopic approach called Population

[☆]This article is dedicated to Professor Alberto Luiz Coimbra, in the 50th anniversary of COPPE (1963-2013), the Graduate School of Engineering of the Federal University of Rio de Janeiro.

*Corresponding author. Tel.: +55 21 2562 8346; fax: +55 21 25628300.
Email address: paulo@peq.coppe.ufrj.br (P. L. C. Lage)
URL: www.peq.coppe.ufrj.br/pesquisa/tfd (P. L. C. Lage)

Balance (PB) combined with the Eulerian multifluid flow formulation (Silva and Lage, 2011), which may be called PB-CFD simulations. The population balance equation (PBE) is the conservation equation for the number of particles, represented by the mean number density function, which depends on the particle properties, the physical space and time (Ramkrishna, 2000). This mesoscopic framework has a large range of applicability and there is a considerable impetus for the development of numerical methods for solving the PBE (Bove et al. (2005), Strumendo and Arastoopour (2008), Fox et al. (2008), Massot et al. (2010), Attarakih et al. (2009), Lage (2011), Yuan et al. (2012) among others). Despite these efforts, there is still a lack of accurate and robust techniques for analyzing the dynamics of particle systems.

Basically, there exist four classes of well established methods to solve the PBE: Monte-Carlo methods, discretization methods, moment methods closed by quadrature and weighted residual methods. In the following, we focused on the quadrature-based moment methods (QBMM) and their variants.

The first QBMM was the quadrature method of moments (QMoM). The QMoM solves the moments of the number density function (NDF) and the integral terms of the PBE are approximated by a N-point Gauss-Christoffel quadrature rule, that is, a Gaussian quadrature whose weight function is the particle number density function (McGraw, 1997). In this method, the N-point Gauss-Christoffel needs to be calculated from the first $2N$ moments using the Product-Difference algorithm (PDA) (Gordon, 1968) or the Modified Chebyshev method (MCM) (Wheeler, 1974). Afterwards, Marchisio and Fox (2005) developed the Direct QMoM (DQMoM), in which the weights and abscissas are tracked in time and space instead of the moments of the NDF. Therefore, DQMoM solves the PBE without calculating the Gauss-Christoffel quadrature during the solution, which reduces the computational cost. The Gauss-Christoffel quadrature provides a discretization in the internal variable, which yields N particle phases in an Eulerian multifluid approach for polydispersed multiphase flows, being the reason why theses methods are considered well-suited for PB-CFD simulations (Silva and Lage, 2011).

Nonetheless, QMoM and DQMoM have some limitations. Both are unable to reconstruct the distribution function, whose values must be calculated at the lower boundary of the particle size space for some problems with a negative growth rate (Massot et al., 2010). Furthermore, the N-point Gauss-Christoffel calculation is an ill-conditioned problem what may limit the number of quadrature points that can be used in the QBMM (Gautschi, 2004). Since the number of quadrature points also controls the QBMM accuracy, a

small number of quadrature points are not usually enough to represent the integral terms of the integrated PBE (Dorao and Jakobsen, 2006b,a). Hence, this might yield a poor representation of the physical behavior of the moments and even a loss of the positiveness of the measure defined by them (Petitti et al., 2010).

Several variants of QMoM and DQMoM have been developed. In the following, these variants are shortly reviewed.

Alopaeus et al. (2006) proposed the usage of fixed-point quadrature rules (FQMoM). This approach avoids some robustness and accuracy problems associated to the Gauss-Christoffel quadrature computation. By analyzing several problems, they concluded that FQMoM was better than QMoM in accuracy and computational cost.

Grosch et al. (2006) solved the moment equations and the quadrature moment approximation simultaneously as a differential algebraic system of equations (DAE). However, they did not obtain a significant improvement compared to the standard QMoM. Gimbun et al. (2009) applied a similar methodology for several mechanisms using an analytical Jacobian matrix in the DAE system. The authors observed an enhancement of robustness and accuracy for pure growth problems but not for cases with breakage and aggregation. This occurs due to the intrinsic quadrature errors of QMoM. Later on, Kariwala et al. (2012) used automatic differentiation (AD) to compute the Jacobian matrix of the DAE-QMoM. This method, termed AD-QMoM, was more robust and at least 2 times faster than DAE-QMoM for the same level of accuracy. Nevertheless, it still presents the intrinsic quadrature error of QMoM.

Su et al. (2007) proposed the usage of an adjustable factor, s, in QMoM, whose purpose is to improve the robustness in the the Gauss-Christoffel quadrature calculation. Basically, they used fractional moments given by

$$\mu_{k/s} = \int_0^\infty x^{k/s} f(x,t) dx = \sum_{\alpha=1}^{N} x_\alpha^{k/s} \omega_\alpha \tag{1}$$

and defined $\tilde{x} = x^{1/s}$ as equivalent abscissas which were then calculated by the product-difference algorithm (Gordon, 1968).

Afterwards, Su et al. (2008) applied the same idea to DQMoM but using an adaptive procedure to chose the value of the adjustable factor, calling the method as Adaptive DQMoM (ADQMOM). The value of the adjustable

factor was determined by a search procedure based on the conditional number of the ADQMoM system of linear equations.

Attarakih et al. (2009) proposed the sectional QMoM (SQMoM), focusing on reconstructing the NDF. The domain is divided in sections whose sectional moments are then used to determine a quadrature for each section, as in QMoM. Although the Gauss-Christoffel quadrature was also used, Attarakih et al. (2009) recommended an equal-weight two-point quadrature with better numerical properties.

Qamar et al. (2011) applied QMoM for solving an univariate PBE using the moments of polynomials to compute the Gauss-Christoffel quadrature for closure. These polynomials are orthogonal in relation to the measure defined by the particle number distribution function and their generalized moments were used to obtain the coefficients in the three-term recurrence relation and then the quadrature points. They stated that this determination of the quadrature rule avoids the ill-conditioned issue present in PDA and MCM. However, they applied the method with just three quadrature points which usually does not lead to an ill-conditioned problem (John and Thein, 2012). In essence, the quadrature rule computation is similar to that used in DuQMoGeM (Lage, 2011), where the modified Chebyshev method was applied to the generalized moments of known families of orthogonal polynomials (for instance Legendre, Laguerre) to compute the coefficients in the recurrence relation. However, the later procedure is better conditioned (Gautschi, 1994). Later on, the method extension for bivariate population balance equation was proposed by Qamar et al. (2010), using an arbitrary transformation of the two internal variables into the independent variable of the polynomials. This method leads to a three-point quadrature whose determination involves a specific set of generalized mixed moments with orders as large as ten.

Massot et al. (2010) introduced a modified sectional DQMoM combined with the method of characteristics. Since they were interested in droplet evaporation problems, the entropy maximization (EM) reconstruction technique was used to rebuild the NDF in order to evaluate the particle flux at the lower boundary of the particle size space. They simulated some evaporation problems, showing that the modified sectional DQMoM is accurate and stable for describing the dynamics of the moments. However, the EM numerical complexity increases significantly when it is applied to a multi-dimensional NDF (Yuan et al., 2012).

An interesting technique that is able to reconstruct the NDF from a finite number of its moments is the kernel density element method (KDEM)

(Athanassoulis and Gavriliadis, 2002). The KDEM expresses the NDF in terms of the superposition of Kernel Density Functions (KDF), which has special features that can ensure the positivity of the reconstructed NDF. In essence, this method is similar to EM. It also uses functions (KDF) whose unknown parameters are obtained by solving a minimization problem based on the moments of the NDF. Based on this idea, Yuan et al. (2012) developed a method called EQMoM, which mixes the properties of QMoM and KDEM. EQMoM is a dual-Gaussian quadrature method which uses a unique parameter to determine the KDFs, which is obtained from an additional moment equation. The results obtained by Yuan et al. (2012) were very good for all studied cases.

As it has already cited, the QBMM usually suffers from error accumulation due to the quadrature approximations, which can eventually degenerate the PBE solution. In order to overcome this inherent problem, Lage (2011) developed the Dual Quadrature Method of Generalized Moments (DuQMoGeM). In this method, the quadrature errors can be controlled using adaptive numerical integration (Favero and Lage, 2012). For this reason, DuQMoGeM attained better results than QMoM for all cases studied by Lage (2011). However, this methodology has some shortcomings. It also tracks the moments of the distribution and, therefore, cannot be used to simulate a polydisperse multiphase flow when the particle velocity depends on the internal variables.

In this work, the Direct Dual Quadrature Method of Generalized Moments (D^2uQMoGeM) was formulated and tested. D^2uQMoGeM uses the same idea behind DQMoM to make DuQMoGeM a direct method. The weights and weighted abscissas are tracking directly as in DQMoM and the quadrature errors are controlled by an adaptive quadrature as in DuQMoGeM. A comparative study between conventional DQMoM and Direct DuQMoGeM (D^2uQMoGeM) was conducted using ten different univariate cases with known analytical solutions.

2. Population balance modeling

An inhomogeneous univariate population balance model with an additive internal variable, x, was considered in this work, leading to the following the PBE in tensorial index notation:

$$\frac{\partial f(x,\mathbf{z},t)}{\partial t} + \frac{\partial \left[u_n(x,\mathbf{z},t) f(x,\mathbf{z},t)\right]}{\partial z_n} - \frac{\partial}{\partial z_n}\left[D_z \frac{\partial f(x,\mathbf{z},t)}{\partial z_n}\right] - S(x,\mathbf{z},t) + R(x,\mathbf{z},t) - r(x,\mathbf{z},t) = 0 \tag{2}$$

where $S(x, \mathbf{z}, t)$ is an additional source term, $\mathbf{z}$ is the physical space coordinates, $r(x, \mathbf{z}, t)$ is the nucleation term and $R(x, \mathbf{z}, t)$ is given by the equation below:

$$R(x, \mathbf{z}, t) = \mathcal{L}_b f(x, \mathbf{z}, t) + \mathcal{L}_a f(x, \mathbf{z}, t) + \frac{\partial [g(x,t) f(x, \mathbf{z}, t)]}{\partial x} \tag{3}$$

where $\mathcal{L}_b$ and $\mathcal{L}_a$ are, respectively, the breakage and aggregation operators, which are given by:

$$\mathcal{L}_b f(x, \mathbf{z}, t) = b(x, \mathbf{y}) f(x, \mathbf{z}, t) - \int_x^{x_{max}} \vartheta(x', \mathbf{y}) b(x', \mathbf{y}) P(x \mid x'; \mathbf{y}) f(x', \mathbf{z}, t)\, \mathrm{d}x' \tag{4}$$

$$\begin{aligned} \mathcal{L}_a f(x, \mathbf{z}, t) = & \int_0^{x_{max}} a(x, x', \mathbf{y}) f(x, \mathbf{z}, t) f(x', \mathbf{z}, t)\, \mathrm{d}x' \\ & - \frac{1}{2} \int_0^{x} a(x - x', x', \mathbf{y}) f(x - x', \mathbf{z}, t) f(x', \mathbf{z}, t)\, \mathrm{d}x' \end{aligned} \tag{5}$$

where the breakage and aggregation functions were assumed to depend on $\mathbf{z}$ and t only through the vector of the continuous phase properties, $\mathbf{y}(\mathbf{z}, t)$, and x_{max} is the upper limit of the internal variable domain, which may be finite or infinite. The daughter probability distribution function has the following properties (Ramkrishna, 2000):

$$\begin{aligned} & \int_0^{x'} P(x \mid x', \mathbf{y})\, \mathrm{d}x = 1; \quad P(x \mid x', \mathbf{y}) = 0 \quad \forall\, (x > x'\,) \\ & \int_0^{x'} x P(x \mid x', \mathbf{y})\, \mathrm{d}x = \frac{x'}{\vartheta(x', \mathbf{y})}. \end{aligned} \tag{6}$$

The other symbols are defined in the nomenclature.

3. Discretization and Approximation of the NDF

The Gauss-Christoffel quadrature rule used in DQMoM is equivalent to the following discretization of the number density function (Marchisio and Fox, 2005):

$$f(x, \mathbf{z}, t) = \sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{z}, t) \delta(x - x_\alpha(\mathbf{z}, t)). \tag{7}$$

where $\delta(x - x_\alpha)$ are Dirac delta functions and ω_α and x_α are, respectively, the quadrature weights and abscissas.

The DuQMoGeM (Lage, 2011) introduced a functional approximation for the NFD in terms of a series of polynomials $\phi_i(x), i = 0, 1, \ldots$ that are orthogonal in terms of the following inner product:

$$\langle \phi_i, \phi_j \rangle_{d\bar{\lambda}(x)} = \int_0^{x_{max}} \phi_i(x)\phi_j(x)w(x)\,\mathrm{d}x = \delta_{ij}\|\phi_i\|^2_{d\bar{\lambda}} \tag{8}$$

where $d\bar{\lambda}(x) = w(x)dx$ is a known measure. The NDF approximation is written as:

$$f(x, \mathbf{z}, t) = w(x) \sum_{i=0}^{2N-1} c_i(\mathbf{z}, t)\phi_i(x) \tag{9}$$

where the c_i functions can be obtained using the orthogonality property as:

$$c_i(\mathbf{z}, t) = \frac{\mu_i^{\phi}(\mathbf{z}, t)}{\|\phi_i\|^2_{d\bar{\lambda}(x)}}, \qquad i = 0, \ldots, 2N-1 \tag{10}$$

where μ_i^{ϕ} are the generalized moments of the NDF defined by

$$\mu_i^{\phi}(\mathbf{z}, t) = \int_0^{x_{max}} \phi_i(x)f(x, \mathbf{z}, t)\,\mathrm{d}x, \qquad i = 0, \ldots, 2N-1. \tag{11}$$

Due to the accuracy of the N-point Gauss-Christoffel quadrature, the first $2N$ moments can be calculated exactly by:

$$\mu_i^{\phi}(\mathbf{z}, t) = \sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{z}, t)\phi_i(x_\alpha(\mathbf{z}, t)). \tag{12}$$

Therefore, the NDF approximation given by equation (9) can be written as:

$$f(x, \mathbf{z}, t) = w(x) \sum_{i=0}^{2N-1} \left[\sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{z}, t)\phi_i(x_\alpha(\mathbf{z}, t)) \right] \frac{\phi_i(x)}{\|\phi_i\|^2_{d\bar{\lambda}(x)}}. \tag{13}$$

4. The Direct Dual Quadrature Method of Generalized Moments

In this section, the D^{2}uQMoGeM is derived for the univariate case. This deduction can be easily extended to multivariate problems similarly to those of DQMoM (Marchisio and Fox, 2005) and DuQMoGeM (Favero and Lage, 2012).

The substitution of equation (7) the first three terms of the equation (2) gives, respectively:

$$\frac{\partial}{\partial t}\left[\sum_{\alpha=1}^{N}\omega_\alpha\delta(x-x_\alpha)\right]=\sum_{\alpha=1}^{N}\left[\delta(x-x_\alpha)\frac{\partial\omega_\alpha}{\partial t}-\delta'(x-x_\alpha)\omega_\alpha\frac{\partial x_\alpha}{\partial t}\right] \tag{14}$$

$$\frac{\partial}{\partial z_n}\left[u_n\sum_{\alpha=1}^{N}\omega_\alpha\delta(x-x_\alpha)\right]=\sum_{\alpha=1}^{N}\left[\delta(x-x_\alpha)\frac{\partial(u_{\alpha,n}\omega_\alpha)}{\partial z_n}-\delta'(x-x_\alpha)u_{\alpha,n}\omega_\alpha\frac{\partial x_\alpha}{\partial z_n}\right] \tag{15}$$

$$\begin{aligned}\frac{\partial}{\partial z_n}\left[D_z\frac{\partial}{\partial z_n}\left(\sum_{\alpha=1}^{N}\omega_\alpha\delta(x-x_\alpha)\right)\right] &= \sum_{\alpha=1}^{N}D_z\left\{\delta(x-x_\alpha)\frac{\partial^2\omega_\alpha}{\partial z_n^2}\right.\\ &- \delta'(x-x_\alpha)\left[2\frac{\partial x_\alpha}{\partial z_n}\frac{\partial\omega_\alpha}{\partial z_n}+\omega_\alpha\frac{\partial^2 x_\alpha}{\partial z_n^2}\right]\\ &+ \left.\delta''(x-x_\alpha)\omega_\alpha\frac{\partial x_\alpha}{\partial z_n}\frac{\partial x_\alpha}{\partial z_n}\right\}\end{aligned} \tag{16}$$

where D_z was assumed to be constant and δ' and δ'' represent the first and second derivatives of the Dirac delta function, respectively.

Substituting equations (14), (15) and (16) in equation (2), we have:

$$\begin{aligned}&\sum_{\alpha=1}^{N}\left[\delta(x-x_\alpha)+x_\alpha\delta'(x-x_\alpha)\right]\left[\frac{\partial\omega_\alpha}{\partial t}+\frac{\partial(\omega_\alpha u_{\alpha,n})}{\partial z_n}-D_z\frac{\partial^2\omega_\alpha}{\partial z_n^2}\right]\\ &\quad-\sum_{\alpha=1}^{N}\delta'(x-x_\alpha)\left[\frac{\partial(x_\alpha\omega_\alpha)}{\partial t}+\frac{\partial(x_\alpha\omega_\alpha u_{\alpha,n})}{\partial z_n}-D_z\frac{\partial^2(x_\alpha\omega_\alpha)}{\partial z_n^2}\right]\\ &=\sum_{\alpha=1}^{N}\delta''(x-x_\alpha)D_z\omega_\alpha\frac{\partial x_\alpha}{\partial z_n}\frac{\partial x_\alpha}{\partial z_n}+S(x,\mathbf{z},t)-R(x,\mathbf{z},t)+r(x,\mathbf{z},t).\end{aligned} \tag{17}$$

The functions γ_α and β_α are defined as the sources of the transport equations for the weights and weighted abscissas, $\theta_\alpha=\omega_\alpha x_\alpha$, respectively:

$$\frac{\partial\omega_\alpha}{\partial t}+\frac{\partial(\omega_\alpha u_{\alpha,n})}{\partial z_n}-D_z\frac{\partial^2\omega_\alpha}{\partial z_n^2}=\gamma_\alpha, \tag{18}$$

$$\frac{\partial\theta_\alpha}{\partial t}+\frac{\partial(\theta_\alpha u_{\alpha,n})}{\partial z_n}-D_z\frac{\partial^2\theta_\alpha}{\partial z_n^2}=\beta_\alpha, \tag{19}$$

Using equations (18) and (19), equation (17) becomes:

$$\begin{aligned}\sum_{\alpha=1}^{N}\left[\delta(x-x_\alpha)+\delta'(x-x_\alpha)x_\alpha\right]\gamma_\alpha-\sum_{\alpha=1}^{N}\delta'(x-x_\alpha)\beta_\alpha=\\ +S(x,\mathbf{z},,t)-R(x,\mathbf{z},t)+r(x,\mathbf{z},t)+\sum_{\alpha=1}^{N}c_\alpha\delta''(x-x_\alpha)\end{aligned} \tag{20}$$

where

$$c_\alpha = D_z\omega_\alpha\frac{\partial x_\alpha}{\partial z_n}\frac{\partial x_\alpha}{\partial z_n}. \tag{21}$$

Applying the generalized moment operator, $\int_0^{x_{max}}\phi_j(x)(.)\,\mathrm{d}x$, to equation (20), we obtain:

$$\begin{aligned}\sum_{\alpha=1}^{N}\left[\phi_j(x_\alpha)-x_\alpha\phi_j'(x_\alpha)\right]\gamma_\alpha+\sum_{\alpha=1}^{N}\phi_j'(x_\alpha)\beta_\alpha=\sum_{\alpha=1}^{N}c_\alpha\phi_j''(x_\alpha)\\ +\int_0^{x_{max}}\phi_j(x)S(x,\mathbf{z},t)dx+\int_0^{x_{max}}\phi_j(x)r(x,\mathbf{z},t)dx\\ -\int_0^{x_{max}}\phi_j(x)R(x,\mathbf{z},t)dx\end{aligned} \tag{22}$$

where $\phi_j' = d\phi_j/dx$.

The approximation defined in equation (13) has the same accuracy of equation (7). Therefore, there is no loss of accuracy if equation (13) is used in the approximation of the aggregation and breakage terms. Hence, these terms become:

$$\begin{aligned}\int_0^{x_{max}}\phi_j(x)\left[\mathcal{L}_a f(x,\mathbf{z},t)+\mathcal{L}_b f(x,\mathbf{z},t)\right]\,\mathrm{d}x=\sum_{i=0}^{2N-1}L_{ji}\sum_{\alpha=1}^{N}\frac{\omega_\alpha\phi_i(x_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}}\\ +\sum_{i=0}^{2N-1}\sum_{k=0}^{2N-1}A_{jik}\sum_{\alpha=1}^{N}\frac{\omega_\alpha\phi_i(x_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}}\sum_{l=1}^{N}\frac{\omega_l\phi_k(x_l)}{\|\phi_k\|^2_{d\bar{\lambda}}}\end{aligned} \tag{23}$$

where

$$A_{jik}=\int_{\Re}\int_{\Re}\left[\phi_j(x)-\frac{1}{2}\phi_j(x+x')\right]a(x,x',t)\phi_i(x)\phi_k(x')w(x)w(x')dxdx' \tag{24}$$

$$L_{ji} = \int_{\Re} b(x,t) \left[\phi_j(x) - 2\Pi_j^{\phi}(x,t)\right] \phi_i(x) w(x) dx \tag{25}$$

$$\Pi_j^{\phi}(x,t) = \int_{\Re} \phi_j(x') P(x'/x) dx' \tag{26}$$

The $\Pi_j^{\phi}(x,t)$, A_{jik} and L_{ji} can be calculated by an adaptive cubature with error control as in DuQMoGeM (Favero and Lage, 2012).

Based on the results of Lage (2011), we decided to approximate the growth integral term by the N-point Gauss-Christoffel quadrature for the semi-infinite domain problems:

$$\int_0^{x_{max}} g(x,t) f(x,\mathbf{z},t) \phi_j'(x)\, \mathrm{d}x = \sum_{\alpha=1}^{N} g(x_\alpha,t) \omega_\alpha \phi_j'(x_\alpha) \tag{27}$$

while, for the finite domain problems, the calculation of this term was carried out using equation (13), which gives

$$\int_0^{x_{max}} g(x,t) f(x,\mathbf{z},t) \phi_j'(x)\, \mathrm{d}x = \sum_{i=0}^{2N-1} G_{ji} \sum_{\alpha=1}^{N} \frac{\omega_\alpha \phi_i(x_\alpha)}{\|\phi_i\|_{d\bar{\lambda}}^2} \tag{28}$$

where

$$G_{ji} = \int_0^{x_{max}} g(x,t) w(x) \phi_j'(x) \phi_i(x)\, \mathrm{d}x. \tag{29}$$

Therefore, the following sets of equations represent the solution of equation (2) by the D^{2}uQMoGeM for semi-infinite and finite domain problems, respectively:

$$\begin{aligned}
&\sum_{\alpha=1}^{N} \left[\phi_j(x_\alpha) - x_\alpha \phi_j'(x_\alpha)\right] \gamma_\alpha + \sum_{\alpha=1}^{N} \phi_j'(x_\alpha) \beta_\alpha + \sum_{i=0}^{2N-1} L_{ji} \sum_{\alpha=1}^{N} \frac{\omega_\alpha \phi_i(x_\alpha)}{\|\phi_i\|_{d\bar{\lambda}}^2} \\
&+ \sum_{i=0}^{2N-1} \sum_{k=0}^{2N-1} A_{jik} \sum_{\alpha=1}^{N} \frac{\omega_\alpha \phi_i(x_\alpha)}{\|\phi_i\|_{d\bar{\lambda}}^2} \sum_{l=1}^{N} \frac{\omega_l \phi_k(x_l)}{\|\phi_k\|_{d\bar{\lambda}}^2} + \left[g(x,t) f(x,\mathbf{z},t) \phi_j(x)\right]_0^{x_{max}} \\
&- \sum_{\alpha=1}^{N} g(x_\alpha,t) \omega_\alpha \phi_j'(x_\alpha) - \int_0^{x_{max}} \phi_j(x) \left[S(x,\mathbf{z},t) + r(x,\mathbf{z},t)\right] dx
\end{aligned} \tag{30}$$

and

$$\sum_{\alpha=1}^{N}\left[\phi_j(x_\alpha)-x_\alpha\phi_j'(x_\alpha)\right]\gamma_\alpha+\sum_{\alpha=1}^{N}\phi_j'(x_\alpha)\beta_\alpha+\sum_{i=0}^{2N-1}L_{ji}\sum_{\alpha=1}^{N}\frac{\omega_\alpha\phi_i(x_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}}$$
$$+\sum_{i=0}^{2N-1}\sum_{k=0}^{2N-1}A_{jik}\sum_{\alpha=1}^{N}\frac{\omega_\alpha\phi_i(x_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}}\sum_{l=1}^{N}\frac{\omega_l\phi_k(x_l)}{\|\phi_k\|^2_{d\bar{\lambda}}}+\left[g(x,t)f(x,\mathbf{z},t)\phi_j(x)\right]_0^{x_{max}}$$
$$-\sum_{i=0}^{2N-1}G_{ji}\sum_{\alpha=1}^{N}\frac{\omega_\alpha\phi_i(x_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}}-\int_0^{x_{max}}\phi_j(x)\left[S(x,\mathbf{z},t)+r(x,\mathbf{z},t)\right]dx. \tag{31}$$

The linear system of equations given by equations (30) or (31) are solved for γ_α and β_α, which are the sources of the transport equations (18) and (19). It should also be noted that the value of the NDF at zero size is required in both equations (30) and (31). If it is different from zero, it can be calculated by equation (13), which may produce unrealistic values due to the lack of positiveness of the f expansion.

5. Numerical Procedure

The implementation of the PBE solution was performed in C++ and FORTRAN. The source codes were compiled with gfortran and g++ using the -03 high optimization flag with double precision variables. The work was carried out in an Intel(R) Core(TM) i5-2400 3.1GHz processor with FEDORA 15 Linux operating system.

The integrations in the evaluation of $\Pi_j^\phi(x,t)$, A_{jik} and L_{ji} were performed using the adaptive CUBATURE routine (Johnson, 2008). The ORTHOPOL package (Gautschi, 1994) was used to calculate the recursion coefficients of the orthogonal polynomials. For the initial conditions, the modified Chebyshev algorithm was used to obtain the Gauss-Christoffel weights and abscissas from the initial values of the generalized moments (Gautschi, 1994).

The time integration was performed using the DASSLC package (Secchi, 2007), an adaptive time step and variable order integrator based on backward differentiation formulas, with required relative and absolute tolerances equal to 5×10^{-13}. This made the time integration error much smaller than the quadrature error embodied in the quadrature moment methods.

In order to evaluate the computation time, the *clock* intrinsic subroutine of g++ was used to obtain the elapsed CPU time in seconds with $0.01s$ of

accuracy. The reported computational time is the average of the CPU time of 10 successive computations.

6. Results and discussion

The D^{2}uQMoGeM was compared against DQMoM in respect of accuracy and computational time. Thirteen test cases considering an univariate homogeneous PBE with additive internal variable with known analytical solutions were selected. These include problems with simultaneous breakage and aggregation (Cases 1, 2 and 3), pure breakage (Cases 4 and 5), pure aggregation (Cases 6 and 7), pure growth (Cases 8, 9 and 10), simultaneous growth and nucleation (Case 11), simultaneous breakage and nucleation (Case 12) and simultaneous breakage, aggregation, growth and nucleation (Case 13). Legendre and Laguerre orthogonal polynomials were used for the problems whose internal variable is defined in finite and semi-finite domains, respectively. Although any family of orthogonal polynomials can be used, an analysis of different families of orthogonal polynomials is out of the scope of this work.

Table 1 summarizes the results of accuracy and computational time for D^{2}uQMoGeM and DQMoM at the end of each simulations, that is, at $t_f = 1$ for cases 6, 7, 8, 9 and 10, $t_f = 2$ for cases 1 and 2, $t_f = 3$ for case 3, $t_f = 5$ for case 13, $t_f = 50$ for case 12 and $t_f = 100$ for cases 4, 5 and 11. In order to evaluate the accuracy, Table 1 shows the relative errors of the moments defined by:

$$\epsilon_k = \frac{|\mu_k - \mu_k^a|}{\mu_k^a} \tag{32}$$

where μ_k and μ_k^a are the numerical and analytical values of the moments, respectively.

6.1. Simultaneous breakage and aggregation

This problem was proposed by Patil and Andrews (1998) and analytically solved by McCoy and Madras (2003) for the initial condition:

$$f(x, 0) = e^{-x} \tag{33}$$

considering $x \in [0, \infty)$ and

$$b(x) = Cx, \quad C = constant, \tag{34}$$

$$a(\tilde{x}, x) = K, \qquad K = 1, \tag{35}$$

$$P(x|x') = \frac{H(x' - x)}{x'}. \tag{36}$$

The analytical solution is given by (McCoy and Madras, 2003):

$$f(t, x) = \Phi^2(t) e^{-x\Phi(t)} \tag{37}$$

where

$$\Phi(t) = \Phi(\infty) \left[\frac{1 + \Phi(\infty) \tanh(\Phi(\infty)\frac{t}{2})}{\Phi(\infty) + \tanh(\Phi(\infty)\frac{t}{2})} \right] \tag{38}$$

being

$$\Phi(\infty) = \sqrt{\frac{2C}{K}} \frac{\sqrt{\mu_1(0)}}{\mu_0(0)} \tag{39}$$

the value of $\Phi(t)$ in the steady state. The regular moments are given:

$$\mu_k(t) = \left[\frac{\Phi(\infty) + \tanh(\Phi(\infty)\frac{t}{2})}{\Phi(\infty)[1 + \Phi(\infty)]} \right]^{k-1} \Gamma(1 + k) \tag{40}$$

which, for $t = 0$, simplifies to

$$\mu_k(0) = \Gamma(1 + k). \tag{41}$$

The parameter $\Phi(\infty)$ determines the value of the particle number density at the steady state. If $\Phi(\infty) > 1$ breakage is dominant while if $\Phi(\infty) < 1$ the aggregation is dominant. The following three cases were considered:

- Case 1: $\Phi(\infty) = 0.5$,
- Case 2: $\Phi(\infty) = 1.0$,
- Case 3: $\Phi(\infty) = 2.0$.

The absolute and relative error used in the CUBATURE package to calculate the A_{jik} and L_{ji} terms was 5×10^{-6} for cases one and three, and 5×10^{-8} for case two. The Laguerre polynomial basis for $w(x) = \exp(-x)$ was used to approximate the NDF and provide the generalized moments.

The differences between the moment values calculated by both methods are small for cases 1 and 3. Therefore, only the moment relative errors of both solutions for these cases with $N = 2$ are shown in Figures 1 and 2. The

former shows that D^{2}uQMoGeM is superior, especially in the beginning of the simulation, but the latter shows that D^{2}uQMoGeM is at least one order of magnitude more accurate than DQMoM.

Figures 3 illustrates the solutions for the moments with $N = 2$ using both methods for case 2, which has an invariant solution. This figure shows that the third moment is affected by the quadrature error in the DQMoM solution but not in the D^{2}uQMoGeM solution. In this case, the methods differs largely in accuracy as shown in Figure 4, which shows the moment relative errors. It can be observed that the errors in the third and fourth moments are substantially larger for the DQMoM solution, being around 5%.

It should be point out that the bottleneck of the D^{2}uQMoGeM simulations is the cost of the adaptive numerical integrations used to compute A_{jik} and L_{ji}. This problem is minimized for these test cases due to the time invariance of the PBE kernels. As shown in Table 1 for these cases, D^{2}uQMoGeM is 2-3 times slower than DQMoM.

For semi-infinite domain problems, it was verified that the adaptive cubature package could not calculate the integrals in a reasonable computational time if the required integration tolerance was too small. Therefore, the adaptive cubature algorithm needs improvement. This issue is not too significant because most engineering problems be be solved in a finite domain due to physical constraints.

6.2. Pure breakage

Lage (2011) defined a pure binary breakage problem in the $x \in [0, 1]$ domain with:

$$b(x) = x^d, \qquad P(x|x') = H(x' - x)/x' \tag{42}$$

If d is an integer number, the breakage frequency belongs to a finite-dimensional polynomial space. A additional source term, S, was defined in order to the following solution be valid:

$$f(x, t) = 2 - e^{-t} \tag{43}$$

whose moments are

$$\mu_k = \frac{2 - e^{-t}}{k + 1}. \tag{44}$$

Two test cases were defined:

- Case 4: $d = 2$, $S(x,t) = 2x^2(2 - e^{-t}) - 2(1 - e^{-t})$,
- Case 5: $d = \frac{1}{3}$, $S(x,t) = 7e^{-t} - 12 + 7(2 - e^{-t})x^{1/3}$.

The absolute and relative tolerances used in the CUBATURE package to integrate the L_{ji} term were 5×10^{-13}, since the breakage integral term is easier to compute. The Legendre polynomials shifted to the $[0, 1]$ interval were used ($w = 1$).

Figure 5 shows the evolution of some of the moments for the solutions with $N = 3$ using both methods for case 4. It can be seen that the μ_0 obtained by the DQMoM deviates from the steady-state solution due to its quadrature error in computing the breakage terms. Figure 6 shows the corresponding moment relative errors, confirming that the D^{2}uQMoGeM is substantially more accurate than DQMoM.

Figures 7 and 8 show, respectively, some of the moments and their relative errors for the solutions with $N = 3$ using both methods for case 5. They show that μ_0 obtained by the DQMoM was largely affected by quadrature errors, while the D^{2}uQMoGeM solution could obtain this moment with a small error up to $t = 80$. Figure 8 clearly shows that the errors in the DQMoM solution are larger for all moments. This result was expected, since the adaptive cubature can control the errors associated by the integral terms (Favero and Lage, 2012).

From Table 1 it can be seen that the D^{2}uQMoGeM was more accurate and faster than the DQMoM for all pure breakage cases.

6.3. Pure growth

Three growth problems were analyzed: two with a negative growth rate in a finite domain, $x \in [0, 1]$, and one considering a positive growth rate in a semi-infinite domain, $x \in [0, \infty)$. In the first two cases, shifted Legendre polynomials were employed, while the Laguerre polynomials were used in the latter. These cases are:

- Case 6: $g(x) = -0.5x$, $f(0,t) = 0.3\delta(x - 0.3) + 0.7\delta(x - 0.7)$,
- Case 7: $g(x) = -0.5$, $f(0,t) = 1.0$,
- Case 8: $g(x) = 0.5x^{0.5}$, $f(0,t) = 0.3\delta(x - 0.3) + 0.7\delta(x - 0.7)$.

In case 6, the particle flux at the boundary $x = 0$ is always zero. Therefore, DQMoM and D^{2}uQMoGeM were able to solve this problem. Figure

9 represents the moment relative errors for the DQMoM and D^{2}uQMoGeM solutions with $N = 2$ for this case, showing that both solutions have similar errors. On the other hand, case 7 can only be solved by D^{2}uQMoGeM, because the non-zero flux at $x = 0$ cannot be calculated by DQMoM. However, the D^{2}uQMoGeM approximation is not too accurate as shown in Figure 10 where the moment relative errors are represented. Both methods provided solutions for case 8, whose moment relative errors are shown in Figure 11, where it can observed that both methods have good accuracy. Table 1 shows that both methods have similar computational costs for these cases.

6.4. Pure aggregation

Two pure aggregation problems in a semi-infinite domain were analyzed. Both considered a sum kernel, $a(x, x') = x + x'$, but with different initial conditions:

- Case 9: $f(x, 0) = e^{-x}$

whose analytical solution solution is (Gelbard and Seinfeld, 1978):

$$f(x,t) = \frac{\exp\left(-t - 2x + xe^{-t}\right) I_1\left(2x\sqrt{1-e^{-t}}\right)}{x\sqrt{1-e^{-t}}} \tag{45}$$

- Case 10: $f(0, t) = \delta(x - 1.0)$

whose analytical solution is given by (Ramkrishna, 2000):

$$f(x,t) = \exp\left(-t - 2x + xe^{-t}\right) \sum_{i=0}^{\infty} \left[x\left(1 - e^{-t}\right)\right]^{i-1} \frac{\delta(x-i)}{i!}. \tag{46}$$

Both cases were solved using Laguerre polynomials. For case 9, both the D^{2}uQMoGeM and DQMoM numerical solutions agree with the analytical solution as can be seen in Figure 12, which shows the moment relative errors.

Case 10 cannot be initialized from the f moments because the ORTHOPOL routine was not able to obtain the quadrature weights and abscissas. However, using a perturbation of 10^{-6} in the 3^{rd} moment, it was possible to initialize both methods for $N = 2$ with initial errors of 10^{-6} in the 3^{rd} and 4^{th} moments. This problem is not present in the DuQMoGeM but must be overcome for the direct methods. Figure 13 shows the moment relative errors for both solutions, which are in good agreement with the analytical solution. Table 1 shows that the DQMoM is 2-3 times faster than D^{2}uQMoGeM for cases 9 and 10.

6.5. Simultaneous growth and nucleation

The combined effects of nucleation and growth was investigated in this section using a semi-infinite domain problem, $x \in [0, \infty)$, named case 11. The corresponding functions that define this case are: $g(x) = 1.0$, $r(x) = \delta(x)$, $f(0,t) = 0$ and $f(x,0) = e^{-x}$. The analytical solution for the moments are (Gimbun et al., 2009):

$$\mu_0(t) = r(x)t + \mu_0(0) \tag{47}$$

$$\mu_k(t) = kg(x) \int_0^t \mu_{k-1}(t)dt + \mu_k(0) \qquad k > 0. \tag{48}$$

The moment relative errors in the solution using both methods are shown in Figure 14. From it and Table 1, we can observe that the D^{2}uQMoGeM seems to be equivalent to or slightly better than DQMoM but it is somewhat slower than the DQMoM for this case. The difference in the computational times is associated to the adaptive integration of the nucleation term in Equation (31) of D^{2}uQMoGeM.

6.6. Simultaneous breakage and nucleation

A new test problem (case 12) was formulated in order to investigate the simultaneous effects of breakage and nucleation in finite domain, $x \in [0, 1]$. For this case, the breakage frequency was $b(x) = x^{1/3}$ and the nucleation rate was $r(x) = \delta(x - 1)$. The solution given by equation (43) is imposed by defining the following source term:

$$S(x,t) = 7e^{-t} - 12 + 7(2 - e^{-t})x^{1/3} - r(x). \tag{49}$$

The moments of the solution are given by equation (44).

The Legendre polynomials shifted to the $[0, 1]$ interval were used to calculate the generalized moments. Figure 15 and Table 1 show that D^{2}uQMoGeM is much more accurate than DQMoM due to its quadrature error control. From this figure, it can be seen that DQMoM could not conserve the μ_0, while D^{2}uQMoGeM calculated it with an relative error smaller than 10^{-5}. Additionally, the D^{2}uQMoGeM was faster than DQMoM in this case.

6.7. Simultaneous breakage, aggregation, growth and nucleation

Finally, a very complex problem (case 13) was constructed including breakage, aggregation, growth and nucleation in the semi-infinite domain, $x \in [0, \infty)$. The functions that define this case are: $b(x) = x^{1/2}$, $r(x) =$

$\delta(x-1)$, $a(x,x') = x + x'$ and $g(x) = x^{1/2}$. The solution given by equation (43), whose moments are given by equation (44), is again imposed by defining the source term:

$$\begin{aligned} S(x,t) = {} & e^{-x-t} - (x+t)e^{-x-t} + (-2\sqrt{x} + \sqrt{\pi}\mathrm{erf}(\sqrt{\pi})e^{x} + 2t\sqrt{\pi}\mathrm{erf}(\sqrt{x})e^{x} \\ & - \sqrt{\pi}e^{x} - 2t\sqrt{\pi}e^{x})e^{-x-t} + \sqrt{\pi}(x+t)e^{-x-t}\frac{x^2}{12}(6t + 6t^2 + x^2)e^{-x-2t} \\ & + (x+t)e^{-x-t}(tx + x + t + 2)e^{-t} + \frac{(x+t)e^{-x-t}}{2\sqrt{x}} + \sqrt{x}e^{-x-t} \\ & - \sqrt{x}(x+t)e^{-x-t} - \delta(x-1) \end{aligned} \tag{50}$$

where erf is the error function. The analytic moments of $S(x,t)$ were used in both D^{2}uQMoGeM and DQMoM.

Figure 16 and Table 1 show that, for case 13, the quadrature errors in the DQMoM are very large, leading to a solution that does not conserve any of its moments. On the other hand, the D^{2}uQMoGeM gave an accurate solution of all moments in a reasonable computational time.

7. Conclusions

In this work, the Galerkin direct DuQMoGeM was formulated. Several homogeneous univariate population balance problems were used to compare the D^{2}uQMoGeM and DQMoM regarding accuracy and computational cost.

For all cases studied, the D^{2}uQMoGeM had, at least, the same accuracy of the DQMoM. When the quadrature closure errors in the DQMoM are significant, the D^{2}uQMoGeM accuracy is much better due to the error control of the adaptive cubature algorithm.

In general, the computational cost of the D^{2}uQMoGeM simulation was somewhat larger than the corresponding DQMoM solution. However, the low number of integrands for adaptive integration in the D^{2}uQMoGeM solution and the time independence of the PBE kernels had contributed to lower its computational costs. Therefore, if the PBE kernels were dependent of the time, the computational time would be very large. Likewise, in order to D^{2}uQMoGeM be considered a candidate for CFD-PBE coupling, the computational cost of the cubature should be somehow reduced.

8. Acknowledgments

Paulo L. C. Lage and Fabio P. Santos acknowledge the financial support from CNPq, grants nos. 302963/2011-1, 476268/2009-5 and 140794/2010-7, and from FAPERJ, grant no. E-26/111.361/2010.

Nomenclature

a	aggregation frequency
A	three-dimensional aggregation matrix
b	breakage frequency
c	coefficient in polynomial approximation of f
f	number density distribution function
g	growth rate
G	growth matrix
H	Heaviside step function
L	breakage matrix
N	number of quadrature points
P	daughter probability density function for particle breakage
S	additional source term in PBE
t	time
$u_{\alpha,n}$	n Cartesian component of the velocity of the α disperse phase, $\alpha = 1, \ldots, N$
w	weight function of inner product
x	internal variable
x_α	quadrature abscissas, $\alpha = 1, \ldots, N$
x_{max}	upper limit of internal variable domain

$\mathbf{z}$	position vector

Greek letters

β	source term in weighted abscissa equation
γ	source term in weight equation
δ	Dirac delta function
ε	value required for the absolute and relative tolerances in adaptive cubature
θ_α	quadrature weighted abscissas, $\alpha = 1, \ldots, N$
ϑ	number of daughter particles formed by breakage
μ_k	k-order monomial moment of f
μ_k	k-order generalized moment of f
Π_k	k-order moment of P
ϕ_k	orthogonal polynomial of k order
ω_α	quadrature weights, $\alpha = 1, \ldots, N$

Superscripts

a	exact solution

Subscripts

a	aggregation
b	breakage
α	related to quadrature point

References

Alopaeus, V., Laakkonen, M., Aittamaa, J., 2006. Numerical solution of moment-transformed population balance equation with fixed quadrature points. Chemical Engineering Science 61, 4919 – 4929.

Athanassoulis, G., Gavriliadis, P., 2002. The truncated Hausdorff moment problem solved by using kernel density functions. Probabilistic Engineering Mechanics 17, 273 – 291.

Attarakih, M.M., Drumm, C., Bart, H.J., 2009. Solution of the population balance equation using the sectional quadrature method of moments (SQMOM). Chemical Engineering Science 64, 742 – 752.

Bove, S., Solberg, T., Hjertager, B.H., 2005. A novel algorithm for solving population balance equations: the parallel parent and daughter classes. derivation, analysis and testing. Chemical Engineering Science 60, 1449 – 1464.

Dorao, C., Jakobsen, H., 2006a. Numerical calculation of the moments of the population balance equation. Journal of Computational and Applied Mathematics 196, 619 – 633.

Dorao, C., Jakobsen, H., 2006b. The quadrature method of moments and its relationship with the method of weighted residuals. Chemical Engineering Science 61, 7795 – 7804.

Favero, J., Lage, P., 2012. The dual-quadrature method of generalized moments using automatic integration packages. Computers & Chemical Engineering 38, 1 – 10.

Fox, R., Laurent, F., Massot, M., 2008. Numerical simulation of spray coalescence in an eulerian framework: Direct quadrature method of moments and multi-fluid method. Journal of Computational Physics 227, 3058 – 3088.

Gautschi, W., 1994. Algorithm 726: ORTHPOL: a package of routines for generating orthogonal polynomials and gauss-type quadrature rules. ACM Transactions on Mathematical Software 20, 21–62.

Gautschi, W., 2004. Orthogonal Polynomials - Computation and Approximation. Oxford Science.

Gelbard, F., Seinfeld, J.H., 1978. Numerical solution of the dynamic equation for particulate systems. Computational Physics 28, 357 – 375.

Gimbun, J., Nagy, Z.K., Rielly, C.D., 2009. Simultaneous quadrature method of moments for the solution of population balance equations, using a differential algebraic equation framework. Industrial & Engineering Chemistry Research 48, 7798–7812.

Gordon, R., 1968. Error bounds in equilibrium statistical mechanics. Journal of Mathematical Physics 9, 655 – 663.

Grosch, R., Marquardt, W., Briesen, H., Wulkow, M., 2006. Generalization and numerical investigation of qmom. AICHE J 53, 207 – 227.

John, V., Thein, F., 2012. On the efficiency and robustness of the core routine of the quadrature method of moments (qmom). Chemical Engineering Science 75, 327 – 333.

Johnson, S.G., 2008. Cubature (Multi-dimensional integration). http://ab-initio.mit.edu/wiki/index.php/Cubature.

Kariwala, V., Cao, Y., K. Nagy, Z., 2012. Automatic differentiation-based quadrature method of moments for solving population balance equations. AIChE Journal 58, 842–854.

Lage, P.L.C., 2011. On the representation of QMOM as a weighted-residual method the dual-quadrature method of generalized moments. Computers & Chemical Engineering 35, 2186 – 2203.

Marchisio, D.L., Fox, .R.O., 2005. Solution of population balance equations using the direct quadrature method of moments. Journal of Aerosol Science 36, 43 – 73.

Massot, M., Laurent, F., Kah, D., Chaisemartin, S.D., 2010. A robust moment method for evalutation of the disappearance of evaporating sprays. SIAM Journal on Applied Mathematics 70, 3203–3234.

McCoy, B.J., Madras, G., 2003. Analytical solution for a population balance equation with aggregation and fragmentation. Chemical Engineering Science 58, 3049 – 3051.

McGraw, R., 1997. Description of aerosol dynamics by the quadrature method of moments. Aerosol Science and Technology 27, 255 – 265.

Patil, D.P., Andrews, J.R.G., 1998. An analytical solution to continuous population balance model describing floc coalescence and breakage – a special case. Chemical Engineering Science 53, 599 – 601.

Petitti, M., Nasuti, A., Marchisio, D.L., Vanni, M., Baldi, G., Mancini, N., Podenzani, F., 2010. Bubble size distribution modeling in stirred gasliquid reactors with qmom augmented by a new correction algorithm. AIChE Journal 56, 36–53.

Qamar, S., Mukhtar, S., Ali, Q., Seidel-Morgenstern, A., 2011. A gaussian quadrature method for solving batch crystallization models. AIChE Journal 57, 149–159.

Qamar, S., Noor, S., Ain, Q.u., Seidel-Morgenstern, A., 2010. Bivariate extension of the quadrature method of moments for batch crystallization models. Industrial & Engineering Chemistry Research 49, 11633–11644.

Ramkrishna, D., 2000. Population Balance - Theory and Applications to Particulate Systems in Engineering. Academic Press, San Diego.

Secchi, A., 2007. DASSLC: User's Manual, a Differential-Algebraic System Solver. Technical Report. UFRGS,Porto Alegre, RS/Brazil. http://www.enq.ufrgs.br/enqlib/numeric/DASSLC.

Silva, L., Lage, P., 2011. Development and implementation of a polydispersed multiphase flow model in openfoam. Computers & Chemical Engineering 35, 2653 – 2666.

Strumendo, M., Arastoopour, H., 2008. Solution of PBE by MOM in finite size domains. Chemical Engineering Science 63, 2624 – 2640.

Su, J., Gu, Z., Li, Y., Feng, S., Xu, X.Y., 2007. Solution of population balance equation using quadrature method of moments with an adjustable factor. Chemical Engineering Science 62, 5897 – 5911.

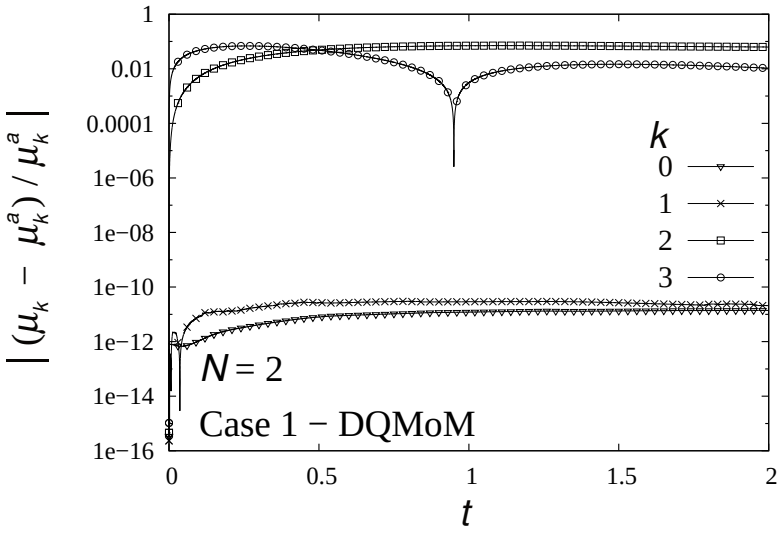
Su, J., Gu, Z., Li, Y., Feng, S., Xu, X.Y., 2008. An adaptive direct quadrature method of moment for population balance equations. AICHE J 54, 2872 – 2887.

Wheeler, J., 1974. Modified moments and gaussian quadrature. Rocky Mountain Journal of Mathematics 4, 287 – 296.

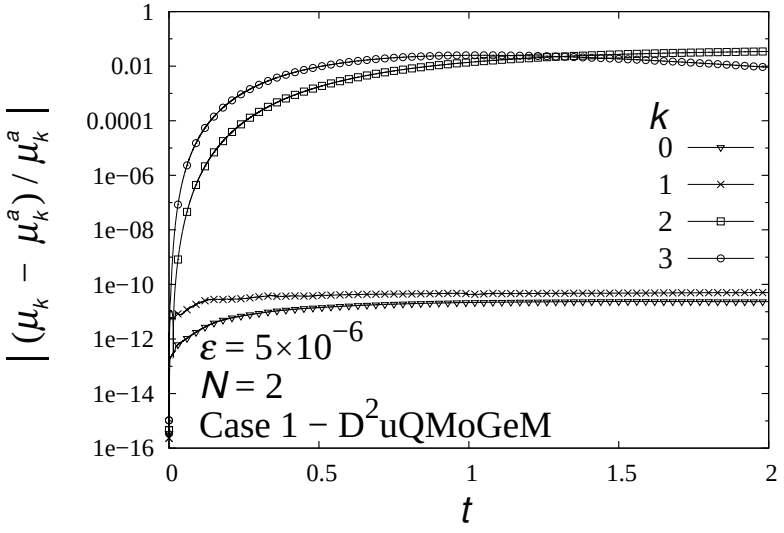
Yuan, C., Laurent, F., Fox, R., 2012. An extended quadrature method of moments for population balance equations. Journal of Aerosol Science 51, 1 – 23.

Table 1: Accuracy versus CPU time for D^{2}uQMoGeM and DQMoM at the end of simulation time.

Cases	Method	$-\log_{10}(\epsilon_k)$ at $t = t_f$ for k equal to						CPU time (s)
		0	1	2	3	4	5	
1	DQMoM	10.6	10.2	2.1	1.4	−−	−−	0.03
	D^{2}uQMoGeM	10.7	10.6	3.7	2.6	−−	−−	0.07
2	DQMoM	12.4	11.1	1.4	1.5	−−	−−	0.03
	D^{2}uQMoGeM	15.7	15.1	14.6	14.4	−−	−−	0.10
3	DQMoM	10.9	10.7	1.2	2.0	−−	−−	0.03
	D^{2}uQMoGeM	10.6	10.2	1.5	2.0	−−	−−	0.07
4	DQMoM	1.1	10.3	3.6	3.3	4.1	4.5	0.23
	D^{2}uQMoGeM	9.5	10.8	10.2	10.6	10.9	10.4	0.10
5	DQMoM	−2.8	8.9	2.7	3.5	3.8	3.7	0.32
	D^{2}uQMoGeM	−0.2	10.1	2.8	3.3	3.4	3.2	0.11
6	DQMoM	15.1	11.0	10.9	11.0	−−	−−	0.01
	D^{2}uQMoGeM	20.0	11.3	11.0	10.9	−−	−−	0.01
7	DQMoM	failed to achieve a solution						−−
	D^{2}uQMoGeM	2.1	1.8	2.4	2.0	−−	−−	0.01
8	DQMoM	15.3	14.9	14.8	14.6	−−	−−	0.01
	D^{2}uQMoGeM	15.6	14.8	14.7	14.5	−−	−−	0.01
9	DQMoM	10.3	10.0	9.3	9.1	−−	−−	0.03
	D^{2}uQMoGeM	10.7	11.9	9.5	9.2	−−	−−	0.10
10	DQMoM	6.3	6.3	5.6	5.3	−−	−−	0.06
	D^{2}uQMoGeM	6.3	6.3	5.6	5.3	−−	−−	0.10
11	DQMoM	14.8	12.3	11.7	11.4	−−	−−	1.19
	D^{2}uQMoGeM	20.0	12.8	11.9	11.5	−−	−−	1.95
12	DQMoM	−1.73	8.8	2.68	3.52	3.82	3.66	0.59
	D^{2}uQMoGeM	4.90	9.53	7.93	8.39	8.57	8.34	0.51
13	DQMoM	−0.95	−1.37	−2.02	−3.04	−−	−−	0.04
	D^{2}uQMoGeM	6.58	6.53	6.49	6.41	−−	−−	2.20

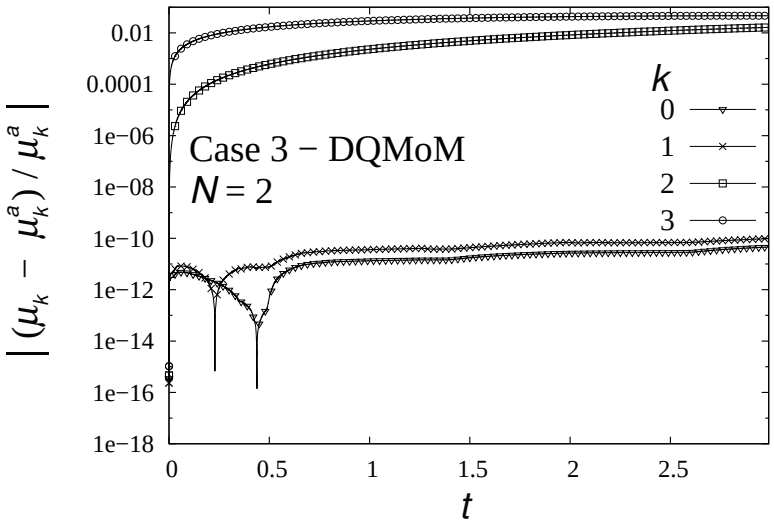


(a)

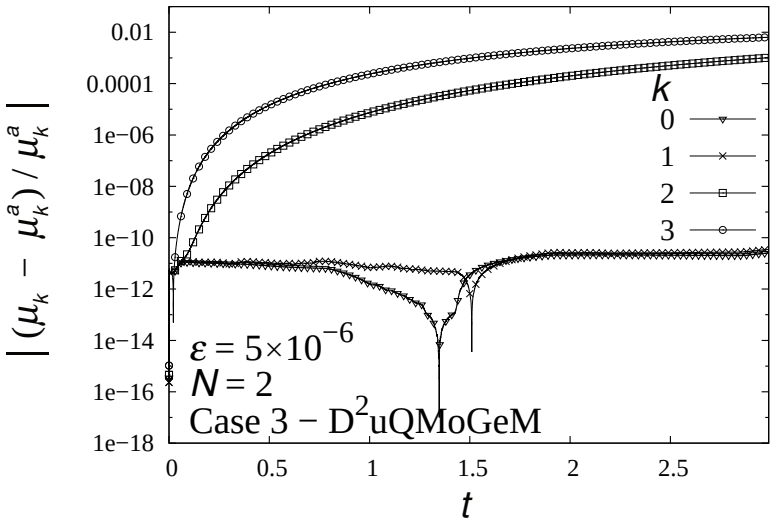


(b)

Figure 1: Moment relative error along time for case 1 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

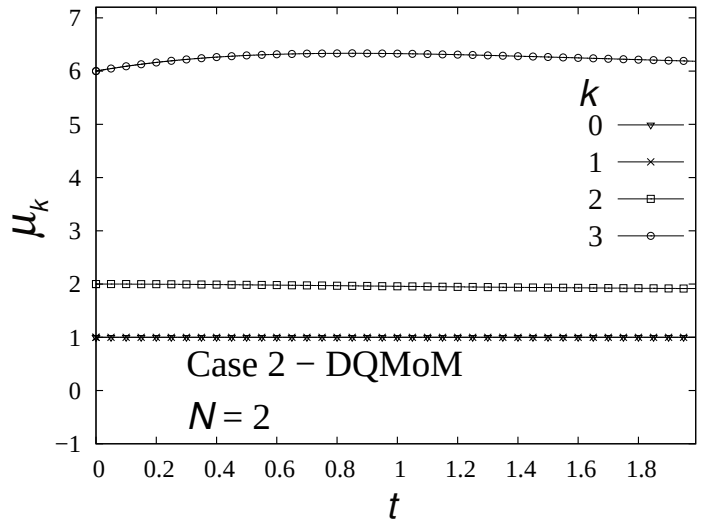


(a)

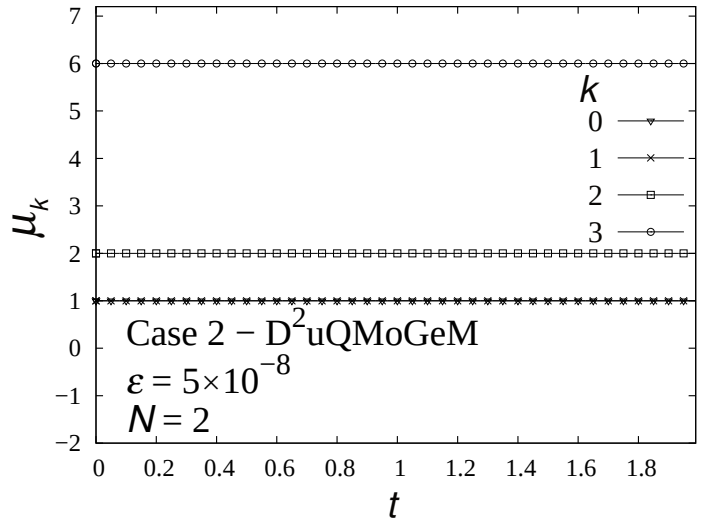


(b)

Figure 2: Moment relative error along time for case 3 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

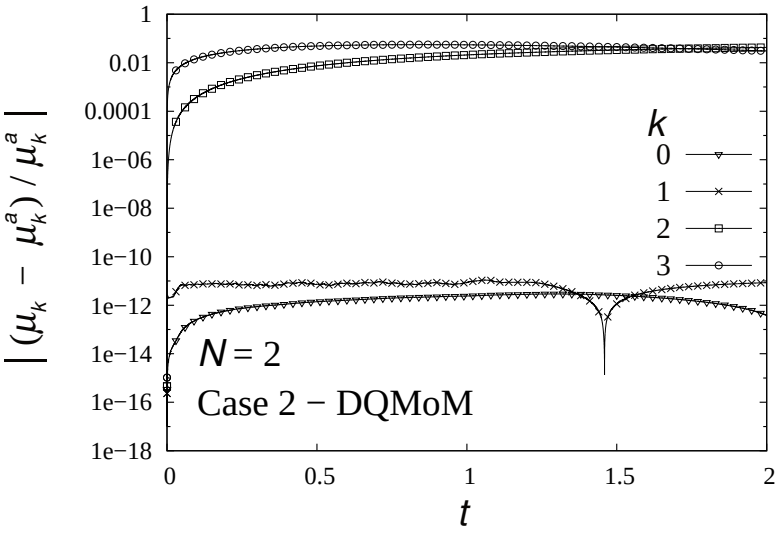


(a)

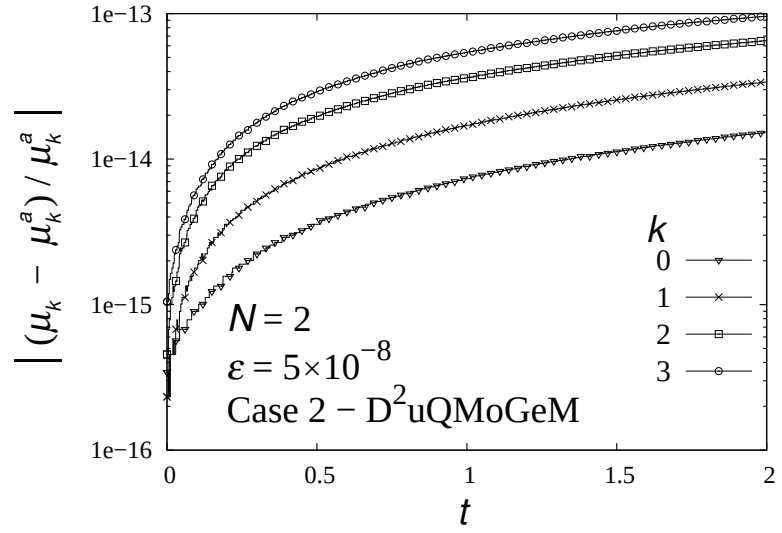


(b)

Figure 3: Moment evolution in time for case 2 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

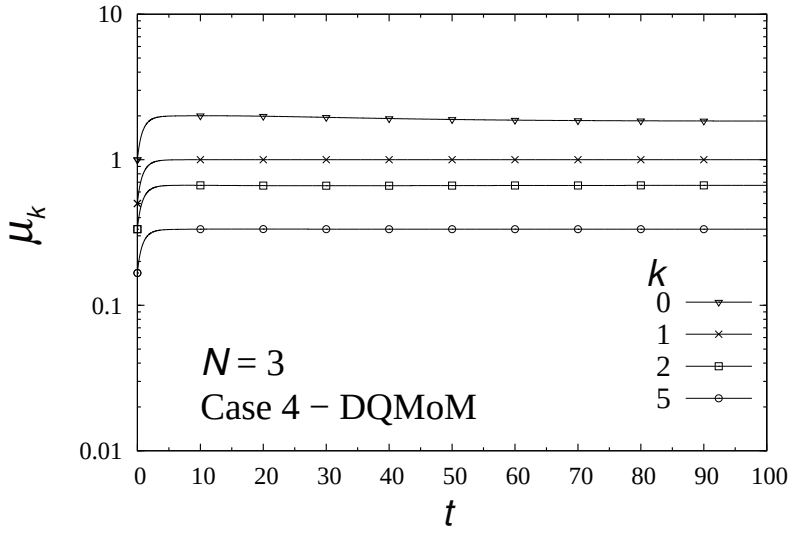


(a)

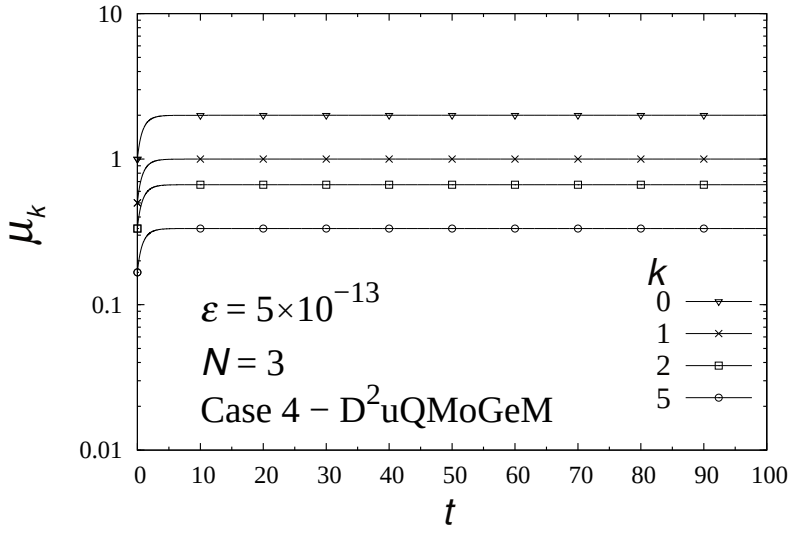


(b)

Figure 4: Moment relative error along time for case 2 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

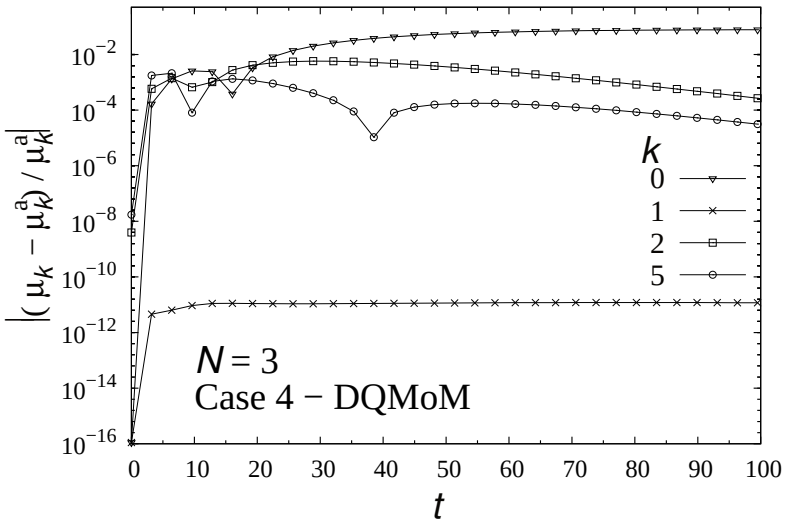


(a)

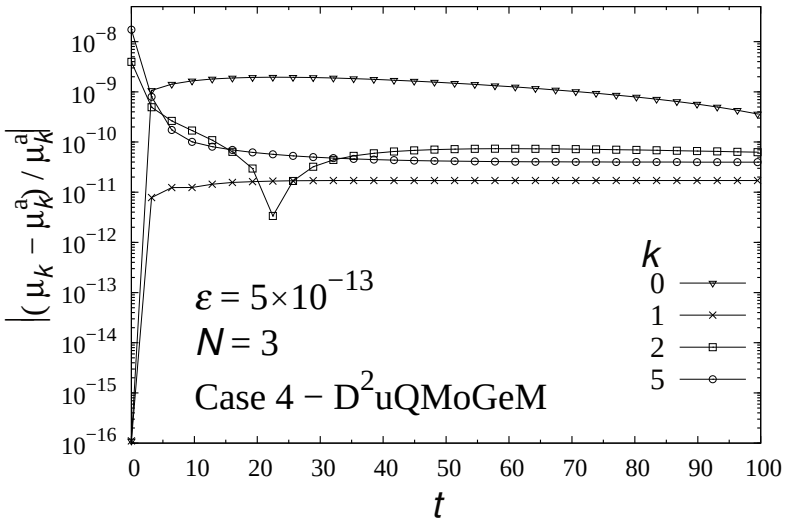


(b)

Figure 5: Moment evolution in time for case 4 with 3 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

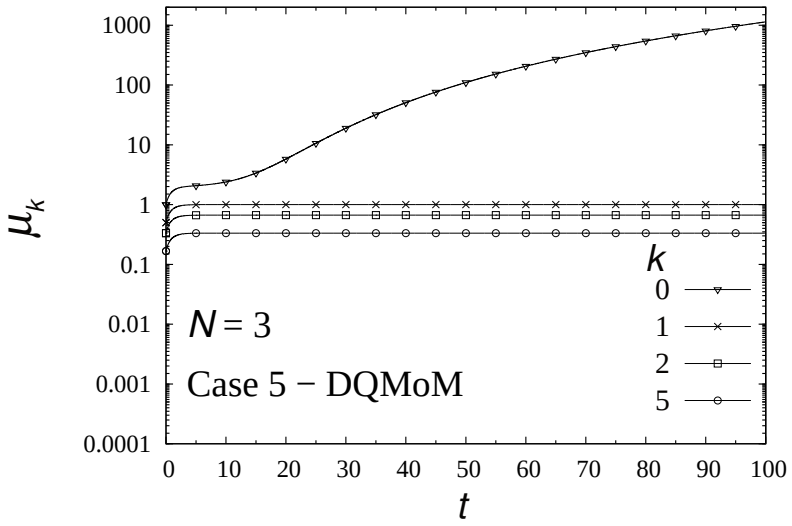


(a)

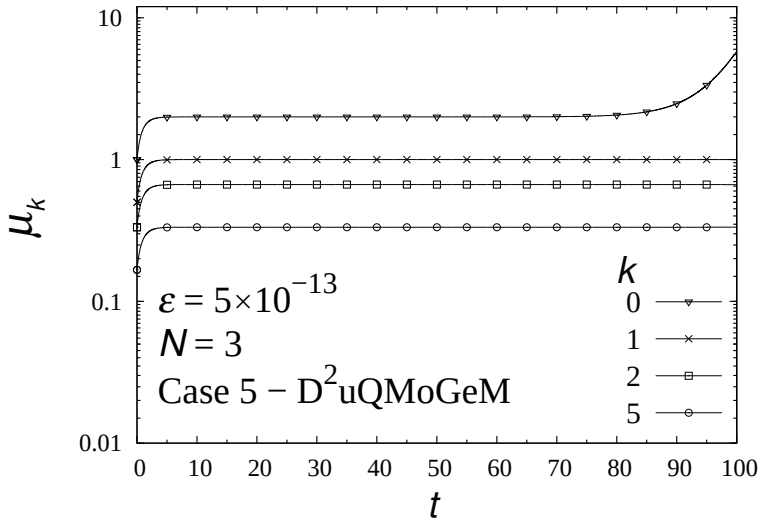


(b)

Figure 6: Moment relative error along time for case 4 with 3 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

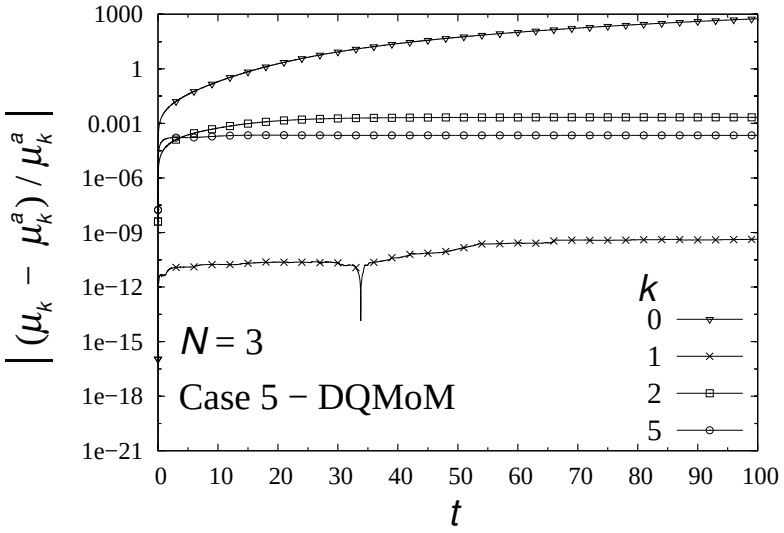


(a)

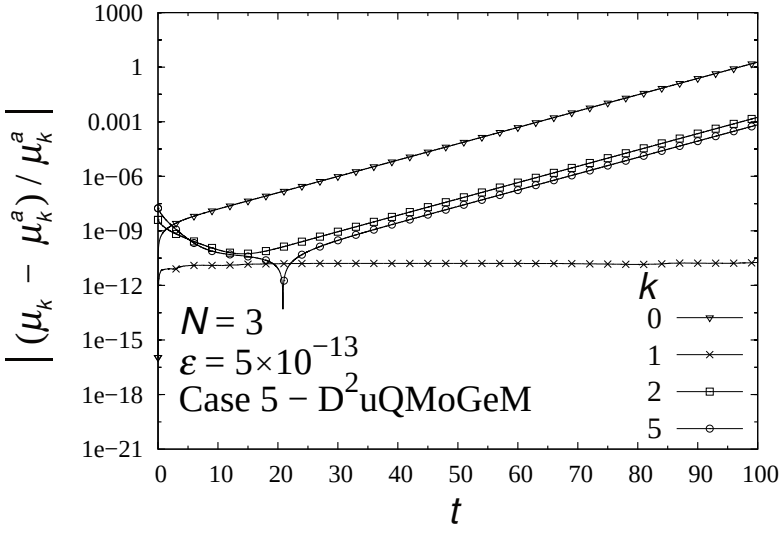


(b)

Figure 7: Moment evolution in time for case 5 with 3 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

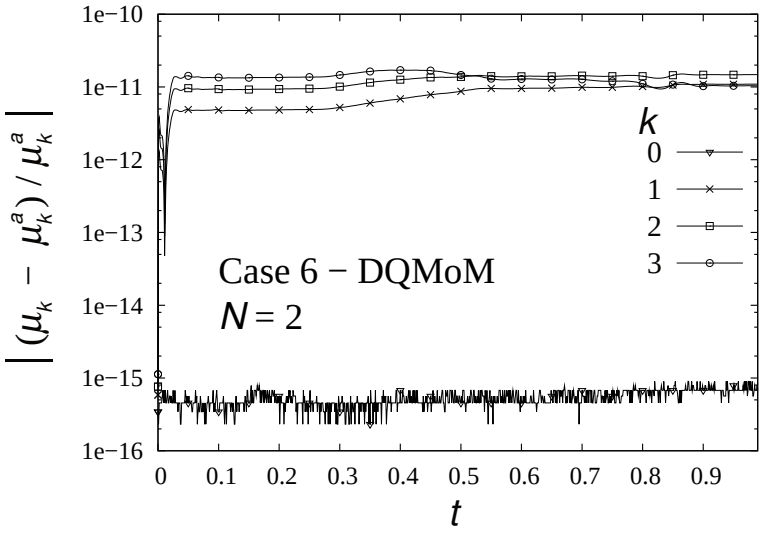


(a)

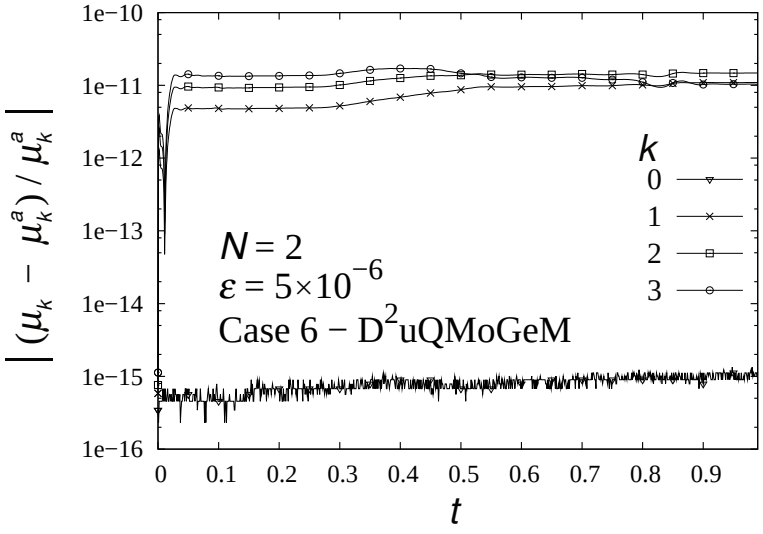


(b)

Figure 8: Moment relative error along time for case 5 with 3 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.



(a)



(b)

Figure 9: Moment relative error along time for case 6 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

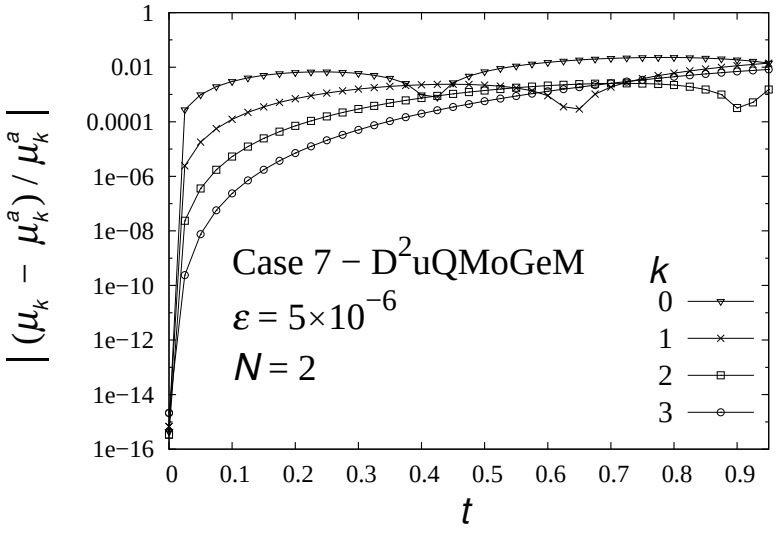
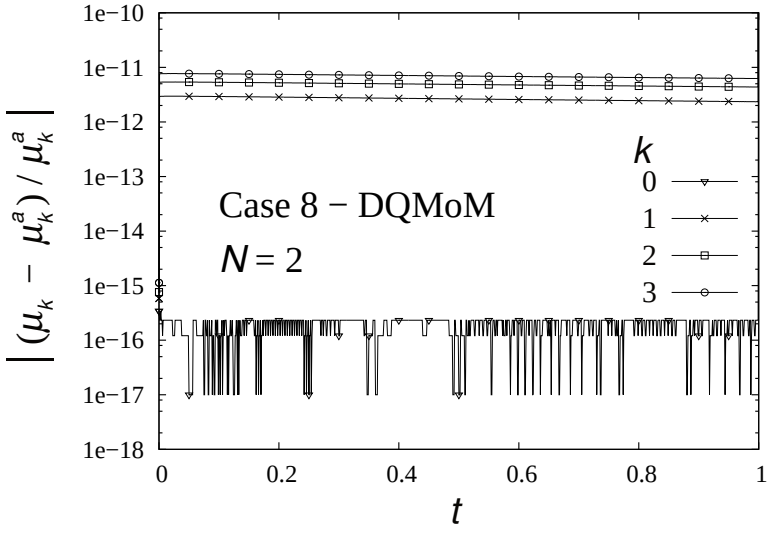
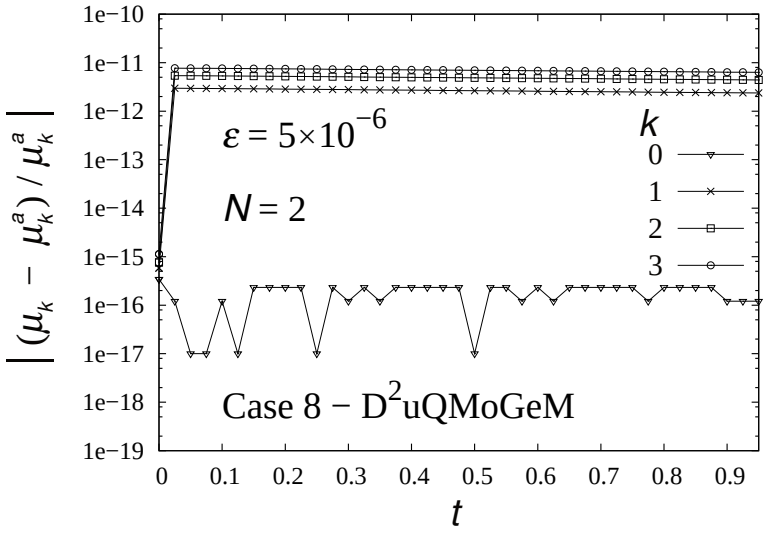


Figure 10: Moment relative error along time for case 7 with 2 quadrature points for the Direct DuQMoGeM.

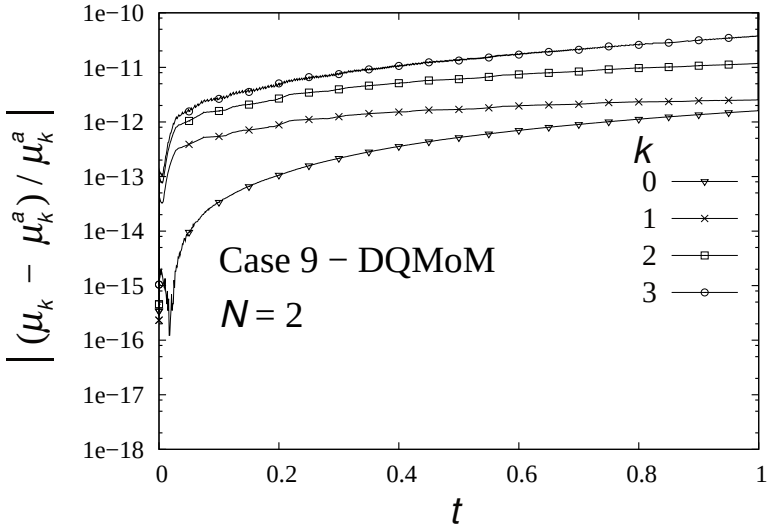


(a)

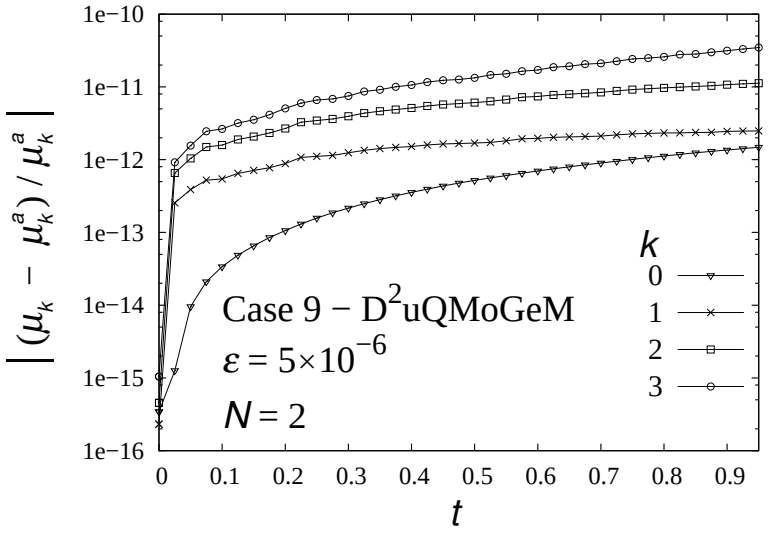


(b)

Figure 11: Moment relative error along time for case 8 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

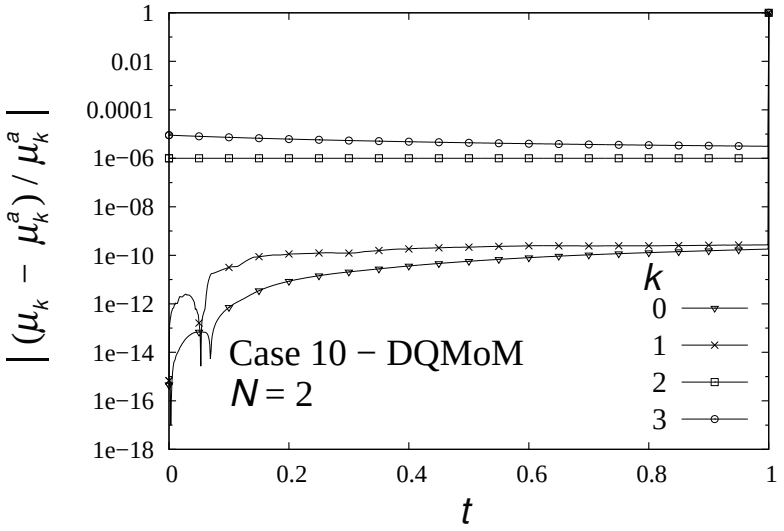


(a)

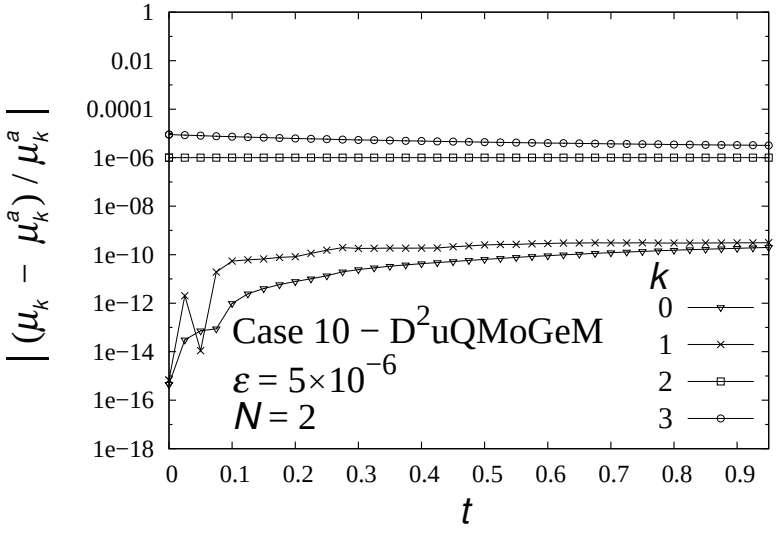


(b)

Figure 12: Moment relative error along time for case 9 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

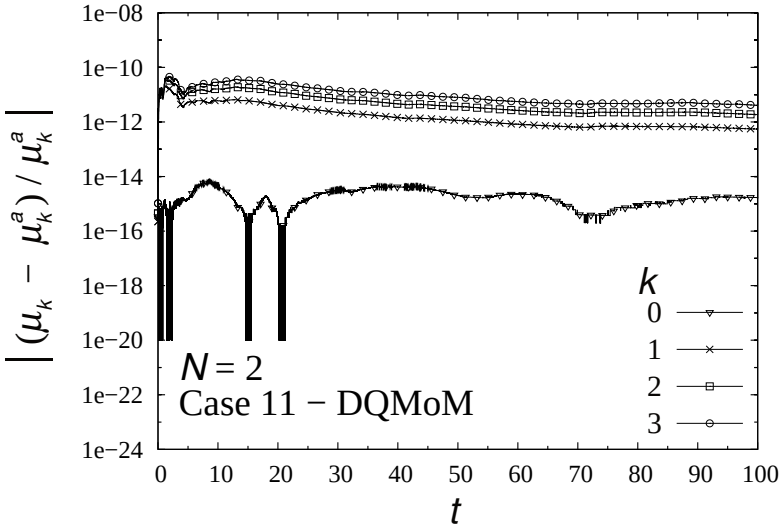


(a)

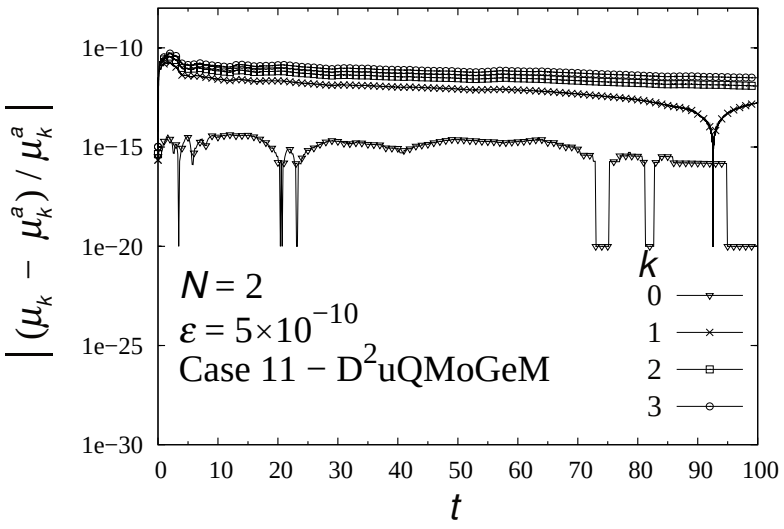


(b)

Figure 13: Moment relative error along time for case 10 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

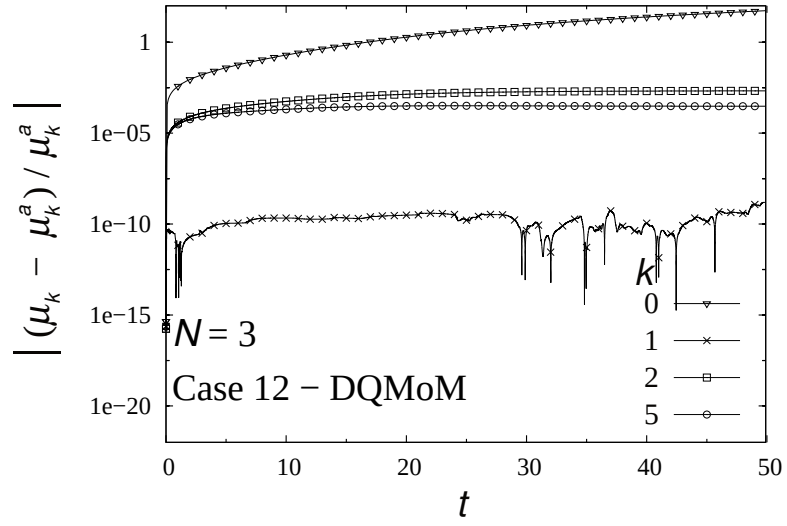


(a)

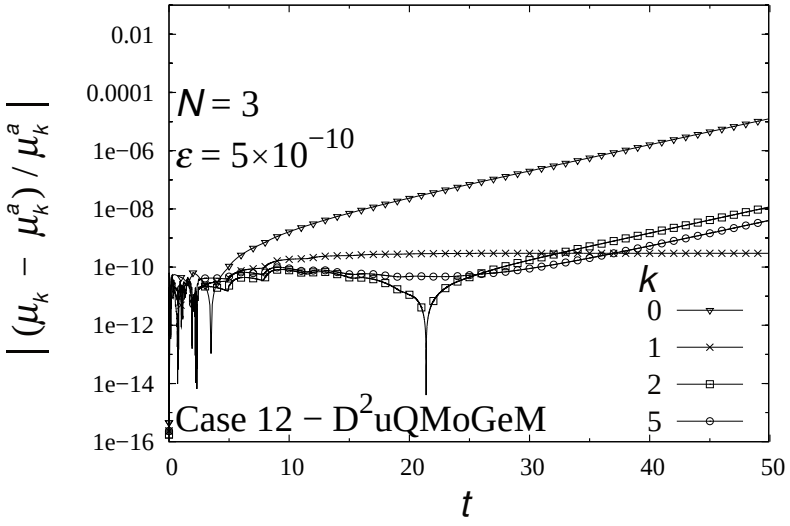


(b)

Figure 14: Moment relative error along time for case 11 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

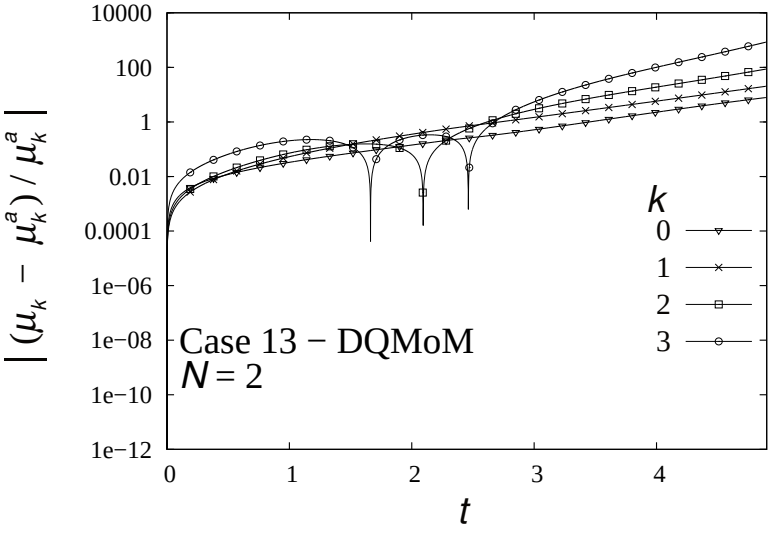


(a)

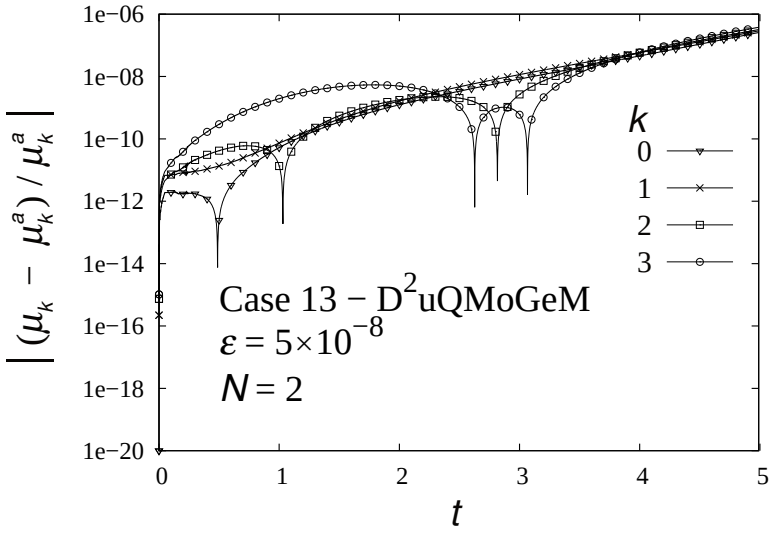


(b)

Figure 15: Moment relative error along time for case 12 with 3 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.



(a)



(b)

Figure 16: Moment relative error along time for case 13 with 2 quadrature points: (a) DQMoM (b) Direct DuQMoGeM.

ANEXO D

Manuscrito III

Desenvolvimento de algoritmos numéricos de cubatura adaptativa em GPUs.

Solution of the Population Balance Equation using Parallel Adaptive Cubature on GPUs[☆]

Fabio P. Santos[a], Inanc Senocak[b], Jovani L. Favero[a], Paulo L. C. Lage[a,1,*]

[a] *Programa de Engenharia Química COPPE, Universidade Federal do Rio de Janeiro, PO Box 68502, Rio de Janeiro, RJ, 21941-972, Brazil*
[b] *Department of Mechanical and Biomedical Engineering, Boise State University, Boise, ID 83725, USA*

Abstract

The Dual Quadrature Method of Generalized Moments (DuQMoGeM) is an accurate moment method for solving the Population Balance Equation (PBE). The drawback of DuQMoGeM is the high computational cost associated with numerical integrations of the PBE integral terms in which each integrand can be integrated independently and, therefore, amenable to parallelization on GPUs. In this work, two parallel adaptive cubature algorithms were implemented on a hybrid architecture (CPU-GPU) to accelerate the DuQMoGeM. The speedup and scalability of these parallel algorithms were studied with different types of Genz's test functions. Then, we applied these parallel numerical integration algorithms in the DuQMoGeM solution of the PBE for three bivariate cases, obtaining speedups between 11 and 15.

Keywords: Population balance modeling, Quadrature Methods of Generalized Moments, Particulate Systems, Adaptive Cubature, GPU

[☆]This article is dedicated to Professor Alberto Luiz Coimbra, in the 50th anniversary of COPPE (1963-2013), the Graduate School of Engineering of the Federal University of Rio de Janeiro.

[*]Corresponding author. Tel.: +55 21 2562 8346; fax: +55 21 25628300.
Email address: paulo@peq.coppe.ufrj.br (Paulo L. C. Lage)
URL: www.peq.coppe.ufrj.br/pesquisa/tfd (Paulo L. C. Lage)

1. Introduction

The modeling and simulation of polydispersed multiphase flow is critically important, because they are present in numerous natural and industrial processes. The precipitation reactor is a good example of an industrial application in which a solid phase with specific features is crystallized from a liquid phase. The final market value of the crystallized product depends on its particle size distribution. Another example is the bubble column chemical reactor whose efficiency depends on the interfacial area density which is determined from the bubble size distribution (Yeoh & Tu, 2010).

A mesoscale framework called Population Balance Equation (PBE) combined with Eulerian multifluid flow formulation (Silva & Lage, 2011) is a convenient way to model the particle dynamics in the above mentioned chemical engineering applications. But development of robust and accurate methods for solving the PBE (Kumar & Ramkrishna, 1996; Bove et al., 2005; Strumendo & Arastoopour, 2008; Fox et al., 2008; Massot et al., 2010; Attarakih et al., 2009; Lage, 2011; Yuan et al., 2012) has been proved to be challenging and remains to be an active research area.

Among the existing methods, the quadrature-based moment methods (QBMM) and their variants are suitable to be coupled to CFD models (Silva & Lage, 2011). This characteristic originates from the fact that these methods provide a discretization in the internal variable that yields n particles phases in an Eulerian multifluid approach for simulating polydispersed multiphase flows. The first QBMM was the Quadrature Method of Moments (QMoM) (McGraw, 1997). QMoM solves for the first $2n$ moments of the particle number density function (NDF) using the n-point Gauss-Christoffel quadrature formula, obtained by the the Product Difference (PD) algorithm (Gordon, 1968), to overcome the so-called closure problem. Later, Marchisio & Fox (2005) proposed the Direct Quadrature Method of Moments (DQMoM) as an alternative to the QMoM. Instead of calculating the NDF moments, Marchisio & Fox (2005) evaluated directly the weights and abscissas of the n-point Gauss-Christoffel quadrature. Fundamentally, DQMoM does not need to compute the Gauss-Christoffel quadrature except for the discretization of the NDF at the initial or boundary conditions.

However, DQMoM and QMoM have limitations. The n-point Gauss-Christoffel calculation is an ill-conditioned problem that limits the number of quadrature points that can be used (Gautschi, 2004). Since a small number of quadrature points may not be enough to accurately calculate the integral

source terms of the PBE moments, these methods accumulate errors due to the quadrature approximation of these terms that may eventually degenerate the solution.

In order to overcome the error accumulation problem, Lage (2011) introduced the Dual Quadrature Method of Generalized Moments (DuQMoGeM). In this method, a quadrature rule of high accuracy can be used to compute the source terms of the PBE, while the NDF is still discretized by the Gauss-Christoffel quadrature formula, having also an approximate polynomial expansion. Lage (2011) showed that, if the PBE kernels do not belong to any polynomial space, a fixed point quadrature cannot be enough to guarantee the accuracy of the method.

Afterwards, Yuan et al. (2012) developed the Extended QMoM (EQMoM), which is also a dual quadrature method. The advantage of this method is that it approximates the NDF using non-negative functions, the kernel density functions (KDFs), which guarantees the positivity of the NDF. The EQMoM also uses the Gauss-Christoffel quadrature based on the first $2n$ moments for the KDF definition, which is completed by minimizing the difference between the values of the $2n$-order moment calculated from the PBE moment and from the Gauss-Christoffel quadrature formula.

Favero & Lage (2012) used an adaptive cubature (Johnson, 2008) to compute the integral terms of the DuQMoGeM, which was able to control the quadrature error within a specified tolerance. They also extended the DuQMoGeM to bivariate population balance problems also using the adaptive cubature to solve homogeneous bivariate problems with error control. However, the computational time using the adaptive cubature can be quite large, make it extremely expensive for CFD applications. But recent developments in programmable GPUs promise to broader the adoption of this technique.

Graphics Processing Units (GPU) were initially developed to accelerate specialized graphics applications. GPUs have emerged as a massively parallel co-processor to CPU for high-performance scientific computing. GPUs have become a new paradigm in the supercomputing field due to its computational power, low cost and high performance per watt (Kirk & Hwu, 2010).

In order to exploit GPU power, we implemented an adaptive cubature method in two different parallel algorithms in GPUs in order to accelerate the DuQMoGeM solution. These algorithms differ in the level of the task and data parallelism. One is parallelized in the integrand level and other in the integral formula level (Schurer, 2001). The parallel adaptive cubature algorithms were extensively tested with different types of Genz's test functions

(Genz & Malik, 1983) and these algorithms were applied for solving three bivariate PBE problems.

2. Population balance modeling, PBM

The PBE is the conservation equation for the number of particles, represented by the mean number density distribution function $f(x, y, t)$, which is a function of particle properties, space and time (Ramkrishna, 2000). Here we consider the bivariate PBE for a homogeneous problem written in terms of additive particle properties, (x, y):

$$\frac{\partial f(x,y,t)}{\partial t} + \mathcal{L}_a f(x,y,t) + \mathcal{L}_b f(x,y,t) = S(x,y,t) \tag{1}$$

where $S(x, y, t)$ is a generic source term and the aggregation and breakage operators are defined by:

$$\begin{aligned} \mathcal{L}_a f(x,y,t) = &\int_0^{x_{max}} \int_0^{y_{max}} a(x,x',y,y',t) f(x,y,t) f(x',y',t)\,\mathrm{d}x'\,\mathrm{d}y' \\ &- \frac{1}{2}\int_0^{x}\int_0^{y} a(x-x',x',y-y',y',t) \\ &f(x-x',y-y',t) f(x',y',t)\,\mathrm{d}x'\,\mathrm{d}y', \end{aligned} \tag{2}$$

$$\begin{aligned} \mathcal{L}_b f(x,y,t) = b(x,y,t) f(x,y,t) - &\int_x^{x_{max}} \int_y^{y_{max}} \vartheta(x',y',t) b(x',y',t) \\ &P(x,y \mid x',y',t) f(x',y',t)\,\mathrm{d}x'\,\mathrm{d}y', \end{aligned} \tag{3}$$

where $b(x, y)$ is the breakage frequency, $\vartheta(x', y', t)$ is the number of particles originated by the breakage of a particle with property x' and y', $P(x, x|x', y')$ is their probability distribution function and $a(x, x', y, y')$ is the aggregation frequency (Ramkrishna, 2000).

The daughter probability distribution function has the following properties:

$$\begin{aligned} &\int_0^{x'}\int_0^{y'} P(x,y \mid x',y',t)\,\mathrm{d}x = 1; \\ &P(x,y \mid x',y',t) = 0, \quad \forall x > x' \text{ or } \forall y > y' \\ &\int_0^{x'} xP(x,y \mid x',y',t)\,\mathrm{d}x = \frac{x'}{\vartheta(x',y',t)}; \\ &\int_0^{x'} yP(x,y \mid x',y',t)\,\mathrm{d}y = \frac{y'}{\vartheta(x',y',t)}; \end{aligned} \tag{4}$$

2.1. Dual quadrature method of generalized moments, DuQMoGeM

In DuQMoGeM, the NDF may be represented by an expansion using a convenient orthogonal polynomial basis (Favero & Lage, 2012):

$$f(x,y,t) = \omega(x)\tilde{\omega}(y)\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} c_{ij}\phi_i(x)\tilde{\phi}_j(y), \tag{5}$$

where c_{ij} are the coefficients of the expansion, $\phi_i(x)$ and $\tilde{\phi}_j(y)$ are the orthogonal polynomial bases in both variables and $\omega(x)$ and $\tilde{\omega}(y)$ are the corresponding weight functions of the inner products that define the orthogonality property of these polynomials:

$$\langle \phi_j, \phi_i \rangle_{d\bar{\lambda}(x)} = \int_0^{x_{max}} \phi_j(x)\phi_i(x)\omega(x)\,\mathrm{d}x = \delta_{ij}\|\phi_i\|^2_{d\bar{\lambda}(x)}, \tag{6}$$

$$\langle \tilde{\phi}_j, \tilde{\phi}_i \rangle_{d\tilde{\lambda}(y)} = \int_0^{y_{max}} \tilde{\phi}_j(y)\tilde{\phi}_i(y)\tilde{\omega}(y)\,\mathrm{d}y = \delta_{ij}\|\tilde{\phi}_i\|^2_{d\tilde{\lambda}(y)}, \tag{7}$$

and the corresponding measures $d\bar{\lambda}(x) = w(x)dx$ and $d\tilde{\lambda}(y) = \tilde{w}(y)dy$.

The coefficients c_{ij} can be obtained by the orthogonality properties as

$$c_{ij} = \frac{\mu_{ij}^{(\phi,\tilde{\phi})}}{\|\phi_i\|^2_{\mathrm{d}\bar{\lambda}(x)}\|\tilde{\phi}_j\|^2_{\mathrm{d}\tilde{\lambda}(y)}}, \tag{8}$$

where the generalized mixed moment is given by

$$\mu_{ij}^{(\phi,\tilde{\phi})} = \int_0^{x_{max}}\int_0^{y_{max}} \phi_i(x)\tilde{\phi}_j(y)f(x,y,t)\,\mathrm{d}x\,\mathrm{d}y. \tag{9}$$

The substitution of the approximation given by Equation (5) into Equation (1) gives (Favero & Lage, 2012):

$$\|\phi_k\|^2_{\mathrm{d}\bar{\lambda}(x)}\|\tilde{\phi}_l\|^2_{\mathrm{d}\tilde{\lambda}(y)}\frac{dc_{kl}}{dt} + \sum_{i,j,p,q=0}^{n-1} A_{klijpq}c_{ij}c_{pq} + \sum_{i,j=0}^{n-1} L_{klij}c_{ij} = S_{kl}, \tag{10}$$

where

$$\begin{aligned} A_{klijpq} = &\int_0^{x_{max}}\int_0^{x_{max}}\int_0^{y_{max}}\int_0^{y_{max}} \left[\phi_k(x)\tilde{\phi}_l(y) - \frac{1}{2}\phi_k(x+x')\tilde{\phi}_l(y+y')\right] \\ &a(x,x',y,y')\omega(x)\tilde{\omega}(y)\phi_i(x)\tilde{\phi}_j(y)\omega(x)\tilde{\omega}(y) \\ &\times \phi_p(x)\tilde{\phi}_q(y)\,\mathrm{d}x\,\mathrm{d}x'\,\mathrm{d}y\,\mathrm{d}y', \end{aligned} \tag{11}$$

$$L_{klij} = \int_0^{x_{max}} \int_0^{y_{max}} b(x,y) \left[\phi_k(x)\tilde{\phi}_l(y) - \vartheta(x,y,t)\Pi^{\phi}_{kl}(x,y) \right] \times \omega(x)\tilde{\omega}(y)\phi_i(x)\tilde{\phi}_j(y)\,\mathrm{d}x\,\mathrm{d}y, \tag{12}$$

$$\Pi^{\phi}_{kl}(x,y) = \int_0^{x} \int_0^{y} \phi_k(x')\tilde{\phi}_l(y')P(x,y \mid x',y',t)\,\mathrm{d}x'\,\mathrm{d}y', \tag{13}$$

$$S_{kl} = \int_0^{x_{max}} \int_0^{y_{max}} \phi_k(x)\tilde{\phi}_l(y)S(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{14}$$

The accuracy of this method depends on the terms A_{klijpq} and L_{klij}, $k,l,i,j,p,q = 0 \cdots n-1$, which can be calculated by an adaptive cubature. It is important to point out that the calculation of these terms is the bottleneck of the DuQMoGeM. Therefore, these terms were computed using the parallel adaptive cubature algorithms implemented on the CPU-GPU heterogeneous architecture in the present study. These integral terms are good candidates for acceleration on GPUs because they are highly parallelizable.

3. Basics of GPU and CUDA

The GPU is a dedicated and specialized device which was originally designed to accelerate graphics rendering. Presently, the GPU is not only being used for graphics applications, but also to accelerate scientific computation (Sanders & Kandrot, 2010).

The CPUs are optimized for sequential code performance and require a sophisticated logic control, while GPUs are highly parallel computing devices composed of hundreds of cores which dedicate more of their resources to computation. This difference of architecture provides a significant disparity between the computational power of the GPUs and CPUs. The GPUs are co-processors that have many scalar processors (SP) distributed over multiple streaming multiprocessors (SM). GPUs programming model are based on the single-instruction multiple threads (SIMT), in which each computational thread executes the same instruction on numerous data elements simultaneously (NVIDIA Corporation, 2010).

The CUDA (Compute Unified Device Architecture) is the Application Programming Interface (API) provided by NVIDIA that allows the development of parallel applications on its GPUs. The CUDA is implemented as an extension of C/C++ or Fortran. In CUDA programming, there exist three main abstractions: the hierarchical thread, which is divided in grids, blocks

and threads, the memory management and the synchronization of threads (NVIDIA Corporation, 2010). A CUDA kernel is a parallel portion of an application that is executed on the GPU, one at a time. Each CUDA kernel executes on one grid that is composed of several blocks, each of them formed by a set of threads. This grid-block-thread hierarchy provides the fine-grained data-level and thread-level parallelism carried out by the threads of each block combined with coarse-grained data-level and the task-level parallelism carried out by blocks. For instance, Frezzotti et al. (2011) presented an algorithm for solving a kinetic equation model onto GPUs, where each GPU block was responsible for one group of particles with a given velocity which was decomposed in a grid of threads, with each thread being associated with one cell of a physical space.

The threads within a thread-block can communicate through the shared memory which can be synchronized using barriers. It means that shared memory is only accessible by threads within a thread-block. Likewise, threads of different blocks can not communicate directly through the shared memory. In addition to shared memory, there are other types of memory: global, texture and constant. These memory spaces are accessible for all threads, but can not be used for communication among them (NVIDIA Corporation, 2010). In this way, the correct usage of these three main abstractions can ensure the performance of CUDA-enabled GPU applications.

4. Adaptive Cubature Algorithm

Multidimensional integrals are present in various branches of science. However, in many cases there is no analytical closed solution (Schurer, 2001) and, therefore, a numerical solution is required.

Let us assume the multidimensional integral to be written as

$$\mathbf{F} = \int_{C_s} \mathbf{h}(\mathbf{x}) d\mathbf{x} \tag{15}$$

where $\mathbf{h} : C_s \longrightarrow \Re^N$, being C_s a hypercube of dimension s. There are four important numerical integration methods. Among them are the Monte Carlo and Quasi-Monte Carlo, that are typically used for large N, the interpolatory cubature and the product of Gauss quadrature rules, being both more suitable for small N (Rudolf & Schrer, 2003). Therefore, the latter two methods are suitable for population balance models due to the low dimensionality of the existing integrals.

Mazzia & Pini (2010) compared the interpolatory cubature and the product of Gauss quadrature rules in the integration over a circle. They concluded that the interpolatory cubature can obtain the same accuracy of the product of Gaussian quadrature rules with only a few quadrature points for two dimensional integrals.

Fundamentally, an interpolatory cubature rule is defined in the following form (Stroud, 1971):

$$\mathbf{F}_M \approx \sum_{i=1}^{M} w_i \mathbf{h}(\mathbf{x_i}) \tag{16}$$

where M is the number of cubature points, $\mathbf{x_i} \in C_s$ are the M abscissas and w_i are the corresponding weights. As Equation (16) gives only an approximation to the integral (Rudolf & Schrer, 2003), an adaptive algorithm can be used to achieve the desired accuracy.

The adaptive algorithm consists of applying the cubature rule to subregions of the integration domain, which are generated by subdivision accordingly to their integral error estimates. The integration error over a subregion can be estimated by using two cubature rules with different orders. Note that a split direction should be selected every time a region is divided whose choice affects the convergence rate. Usually, the split direction is chosen to be the direction with the largest integrand variation in the region. The adaptive cubature algorithm, given below, stops when the global error, which is computed from the local errors, satisfies a desired tolerance or the maximum number of function evaluation is reached (Rudolf & Schrer, 2003).

In order to accelerate the automatic cubature, researchers have proposed several parallel adaptive cubature algorithms. According to Schurer (2001), there are six forms of parallelizing an adaptive cubature algorithm. These forms differ in terms of their levels of parallelism (Genz, 1989). Among these, the most common are the parallelism at the integral formula, subdivision and integrand levels. In the integral formula level, the basic formula is parallelized, and each processor or thread computes a subset of the cubature points. The parallelization at the integrand level means that one or more integrands are calculated by one processor or thread. In the subdivision level, each processor calculates a subdomain of the entire domain (Schurer, 2001).

The majority of the implemented algorithms are parallelized at the subdivision level (D'Apuzzo et al., 1997; Bull & Freeman, 1995; Schurer, 2001), because this strategy is well-suited for multiple-instruction multiple-data architectures (MIMD). Therefore, the parallelization of the adaptive cubature

Adaptive Cubature Algorithm

Put the initial regions into a collection of regions.
for all *regions in the collection* ***do***
  Apply the cubature rules.
  Select the subdivision direction.
  Compute the error estimate.
end for
Compute the global error estimate.
while *the tolerance is not achieved and the maximum number of iterations is not reached* ***do***
  Choose a region with the largest absolute error.
  Split this region into new regions.
  for all *new regions* ***do***
    Apply the cubature rules.
    Select the subdivision direction.
    Compute the error estimate.
  end for
  Store this new regions in the collection of regions.
  Compute the global error estimate.
end while

on the GPU programming model, which is based on single-instruction multiple threads, is a novel and challenging task.

5. GPU Implementation

There are several packages that implemented an adaptive cubature algorithm (Hahn, 2005; Johnson, 2008; Schurer, 2001). However, none of those takes advantage of CUDA-enabled GPU devices. Generally, the parallel adaptive cubature implementations in MIMD architectures are restricted to the parallelism at the subdivision level. However, in the present study we implemented this algorithm on GPUs using parallelism also at the integrand and integral formula levels. These parallelism levels were chosen because they are better suited for the single-instruction multiple threads programming model, because each integrand can be computed independently (Schurer, 2001).

In each region, the integral and its error were computed using the embedded cubature rules of fifth and seventh-order developed by Genz & Malik

(1983). Likewise, the direction with the largest value of the fourth divided difference operator (FDDO) was selected for the region subdivision (Genz & Malik, 1983). It is important to note that this operator does not contribute appreciably to the computational cost because it uses the integrand values at the cubature points of the Genz & Malik (1983) rules.

The error control strategy was based on a measure of the integration error over the whole domain, which was given by the sum of the magnitudes of the estimated local errors in all subregions. Clearly, this is a conservative measure of the global error, because it does not allow the possible cancellation of positive and negative errors of different subregions. Given values for the relative, ε_{rel}, and absolute, ε_{abs}, tolerances were used to define a mixed tolerance given by:

$$\varepsilon = \max\left(\varepsilon_{abs}, \varepsilon_{rel}\left|\mathbf{F}_M\right|\right) \tag{17}$$

where $\mathbf{F}_M$ is the integral over the entire domain. The adaptive algorithm stops when the maximum number of iterations is achieved or the measure of the global error is less than ε.

5.1. Parallelism at the Integral Formula Level (PIFL)

At the integral formula level, we adopt both task- and data-level parallelism. A CUDA kernel is responsible for computing the integral of a vector of integrands over a sub-region. The blocks perform the task-level parallelism and the threads the data-level parallelism. It means that each block of a grid calculates the integral of one integrand of the vector of integrands, while each thread of this block computes one cubature point to obtain this integral. Figure 1 shows the PIFL applied to a problem with N integrands and M cubature points in which there are at least M threads per block and N blocks. Furthermore, as FDDO uses the cubature points, each block also calculates the value of this operator in all directions for its integrand.

In order to overlap the computation of different sub-regions, the cubature points and the integral interval are copied from CPU to the GPU using asynchronous copy functions. This strategy is also used to copy the vector of integrals, the sub-region absolute errors and the FDDO values from GPU to GPU. This feature is particularly related to the Fermi architecture where a GPU can execute multiple kernels based on the resource availability.

For each integrand, the cubature points calculated by the threads must be added for obtaining the integral in each block. This is performed by the so-called parallel reduction algorithm that uses the shared memory for

thread collaboration inside a block (Sanders & Kandrot, 2010). The main concept of this algorithm is that each thread add two values of the vector of function evaluations at the cubature points, $w_j h_i(\mathbf{x_j})$, which are available in the shared memory, being the result saved back in the shared memory. This procedure is performed for $log_2(M)$ steps, where M is the number of cubature points. After all steps, the final sum of the cubature points is stored at a variable with the corresponding block index.

For instance, Figure 2 shows the parallel reduction algorithm for $M = 8$, where the cubature point calculated by *thread one* is added to the cubature point calculated by the *thread eight*, and the resulting value is stored in the shared memory with the index corresponding to thread one. This is carried out for all pair of threads with indexes j and $M - j + 1$ in this step. After 3 steps, the $\sum_{j=1}^{8} w_j h_1(\mathbf{x_j})$ is stored in the shared memory, which is then copied to the global memory using the index of the corresponding block and integrand. The reader can find more details of the reduction algorithm in Sanders & Kandrot (2010).

5.2. Parallelism at the Integrand Level (PIL)

In this parallelism level, each CUDA thread calculates one integral of the vector of integrands. Each block has T integrands and, therefore, calculates the integrals of T integrands. Figure 3 describes this algorithm. The FDDO is also calculated at the thread level. Again, the asynchronous copy is used to overlap the data copy and the computations. The cubature points and the intervals of integration are copied from the CPU to GPU and the resulting vectors of integrals, absolute errors and FDDO values are copied from GPU to GPU. In order to be efficient, this parallel algorithm requires a large number of integrands, which is typical of the solution of DuQMoGeM, especially when it is coupled to CFD simulations. For instance, a simultaneous aggregation and breakage bivariate PBE using $n = 3$ produces a total of 810 integrands that correspond to the n^6 integrals that define A_{klijpq}, Equation (11), and the n^4 integrals in L_{klij} definition, Equation (12). It should be noted that, if the same polynomial basis is used for both variables ($\phi = \tilde{\phi}$), there are several symmetries that can be used to reduce the total number of integrands. However, this is just a particular case and we preferred not to use such symmetries. When the PBE solution is coupled to a CFD simulation, the number of integrands is equal to 810 times the number of CFD control volumes. Equally important is the low dimensionality of the PBE integrals, which makes their cubature calculation at the thread level not too expensive.

6. Results and Discussion

In this section, parallel algorithms are assessed regarding their computational speedup. This metric represents the ratio between the execution time of the CPU serial version of the code and the GPU version, as follows,

$$S = \frac{T_s}{T_{gpu}}, \tag{18}$$

The source codes were compiled with *g++* compiler, version 4.1.2, using the -03 high optimization flag and with *nvcc* compiler using CUDA compilation Toolkit 4.1 with double precision variables. The CPU and GPU codes were executed on an Intel(R) Xeon(R) CPU X5570 2.93GHz and NVIDIA GTX480 graphic card, respectively.

First, we carried out a speedup and scalability analysis using four multidimensional integrals with known analytical solutions. Then, the implemented parallel automatic cubature algorithms were applied to three bivariate population balance problems: two pure aggregation problems and one with simultaneous aggregation and breakage.

6.1. Perfomance Analysis

Genz & Malik (1983) identified some groups of functions which are useful to evaluate multidimensional integrations routines, defining integrand families as the oscillatory, product peak, corner peak and Gaussian families. Four different integrands of these families with four dimensions were used in this work for the speedup and scalability tests of the implemented PIFL and PIL algorithms. The test integrands were:

1. Oscillatory integrand

$$f_1(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 cos(2\pi + \sum_{i=1}^{4} x_i)\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{19}$$

2. Product Peak integrand

$$f_2(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \frac{1}{\prod_{i=1}^{4} \left(1 + [x_i - 1]^2\right)}\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{20}$$

3. Corner Peak integrand

$$f_3(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \frac{1}{\left(1 + \sum_{i=1}^{4} x_i\right)^6}\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{21}$$

4. Gaussian integrand

$$f_4(\mathbf{x}) = \int_0^1 \int_0^1 \int_0^1 \int_0^1 e^{\sum_{i=1}^{4}(x_i-0.5)^2}\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4 \tag{22}$$

For the speedup and scalability analysis using the model integrands given above, two strategies were used to mimic problems with large computational costs. First, in order to analyzed the simultaneous integration of a large number of integrands, each of the above integrand was replicated to form an N dimension integrand vector. Second, as the Genz (1984) multi-dimensional integrand families were designed to be simple, the arithmetic load of each integrand was increased by calculating each quadrature point ten times.

Figure 4 shows the speedup of the two CUDA implementations for several values of the number of integrands, N, for the four chosen integrands with absolute and relative tolerances of 10^{-8}. Figure 5 shows similar results that were obtained with all integrands evaluated 10 times at each quadrature point, simulating a tenfold increase in the computational complexity of the integrand. From both figures, it can be seen that the speedup can be less than one for very low N values because the workload is too low to justify the GPU usage. However, when N increases, the speedup increases in a behavior that depends on the integrand and algorithm. For $N \leq 512$, it can be seen that the PIL and PIFL algorithms are equivalent but, as N further increases, it is clear that the PIL algorithm becomes superior due to its larger independence among threads.

These results also demonstrate that the speedup is strongly dependent on the complexity of the integrand functions. Comparing the maximum speedup values in Figures 4 and 5 for the PIL algorithm, it can be verified that the tenfold increase in the computational cost of the integrand led to 2-6 times larger speedup values. Moreover, the speedup became less dependent on the integrand, being in the 50-70 range for all test integrands.

In order to verify the influence of the number of sub-regions generated by the domain division algorithm divisions on the speedup, we decided to perform a similar speedup analysis to that shown in Figure 5 but with the less strict value of 10^{-5} for both relative and absolute tolerances. The corresponding results are shown in Figure 6. Comparing Figures 5 and 6, it is clear that the speedup curves for both algorithms were very little affected by the value chosen for the tolerances. Therefore, the speedup of each algorithm is basically related to their parallel CUDA implementation.

It should be pointed out that the speedup achieved by both algorithms with a large number of integrands was considerable, which is important for PB-CFD simulations using DuQMoGeM. However, the PIL algorithm is more attractive for PB-CFD simulations for two reasons. First, it is faster than the PIFL algorithm for a large number of integrands. Second, it can compute more integrands simultaneously than the PIFL algorithm due to the Fermi architecture limitation of 65635 blocks (NVIDIA Corporation, 2010). In other words, as the limit of the total number of threads is larger than the total number of blocks, the PIL algorithm can support a larger number of integrands.

6.2. Application to DuQMoGeM solutions

The PIFL and PIL algorithms were used in the DuQMoGeM solution for some homogeneous bivariate population balance problems. The simulation program to solve the PBE was written in CUDA/C++. The time integration was carried out by DASSLC (Secchi, 2007) with required relative and absolute tolerances equal to 10^{-10}. The DuQMoGeM was applied to the PBE in a semi-finite domain using the Laguerre polynomial basis. The adaptive cubature domain was defined as $x \in [0, 100]$ and $y \in [0, 100]$, which is large enough not to affect the integral values. The speedup of the PBE solution includes the computational costs for both the adaptive cubature and the time integration. However, the latter is negligible and the speedup can be considered to be the speedup of the former.

Additionally, in order to evaluate the computational complexity of the PBE kernel integrands, we defined the ratio

$$E = \frac{T_c}{T_{ave}} \tag{23}$$

where T_c is the total computational time of evaluating all PBE kernel integrands and T_{ave} is the average computational time of all the Genz's integrands used in the previous analysis but with the same number of integrands. This ratio was about 4.7 for all the cases described below, which means that the speedup results for the Genz's integrands with 10 evaluations and same N value should be comparable to the speedup obtained in the PBE solutions.

6.2.1. Pure Aggregation Problem (Case I)

The pure bivariate aggregation problem proposed by Gelbard & Seinfeld (1978) was simulated with the parallel algorithms in order to verify their

speedup. For this case, the aggregation frequency is constant, $a(x, x', y, y') = 1.0$. Although Gelbard & Seinfeld (1978) have studied many analytical solutions for this type of problem with different initial conditions, the initial condition

$$f(x, y, 0) = e^{-(x+y)} \tag{24}$$

was used here, which leads to the analytical solution:

$$f(x, y, t) = \frac{4}{(t+2)^2} e^{-(x+y)} I_0 \left(2\sqrt{\frac{t}{t+2}} xy \right) \tag{25}$$

where I_0 is the zeroth order Bessel function of the first kind. This simulation was carried out with $n = 3$, which yields a total of 729 integrands, using absolute and relative tolerances of 5×10^{-5}.

Figure 7 shows the time evolution of moments and their relative errors. It can be seen from Figure 7 that the results are in agreement with the analytical solution. Both the serial and the parallel versions of the code produced the same numerical results. The parallel version is about 12 times faster for the PIL algorithm and 11 times for PIFL algorithm.

6.2.2. Simultaneous Aggregation and Breakage Problem (Case II)

This subsection presents the results for a simultaneous aggregation and breakage population balance model described in Favero & Lage (2012). Equation 1 was considered with two additive internal variables which can represent the total particle mass, x, and the mass of a component, y, in a two-component particle. Binary breakage with linear breakage frequency and constant aggregation frequency were assumed:

$$\vartheta(x', y', t) = 2, \qquad b(x, y) = x, \qquad a(x, x', y, y') = 0.1. \tag{26}$$

The probability distribution function of daughter particles was considered to be uniform for the particle mass and a Dirac delta distribution for the daughter particle concentration, y/x, which have to be equal to the mother particle concentration, y'/x'. Therefore, it is written as:

$$P(x, x|x', y') = \frac{1}{x'} H(x' - x) \delta \left(y - \frac{y'x}{x'} \right). \tag{27}$$

If the analytical solution given by

$$f(x, y, t) = \left(2 - e^{-t} \right) xy e^{-(x+y)} \tag{28}$$

is imposed, the additional source term necessary to make it valid is (Favero & Lage, 2012):

$$S(x,y,t) = xye^{-(x+y)}\Big(e^{-t} - \Big(2 - e^{-t}\Big) \Big[\tfrac{1}{10}\Big(2 - e^{-t}\Big)\Big(\tfrac{x^2y^2}{72} - 1\Big) + 2x\Big(\tfrac{x^2+y^2+2xy+2x+2y+2}{(x+y)^3} - \tfrac{1}{2})\Big]\Big). \quad (29)$$

For this case, the absolute and relative tolerances were specified as 10^{-3} for the serial and parallel adaptive cubature codes. The simulation was performed with $n = 3$, which produces a total of 810 integrands. The results was in agreement with the analytical solution and the parallel version is about 15 times faster than the serial CPU code for both algorithms.

6.2.3. Pure Aggregation Problem with an additive frequency (Case III)

This aggregation problem is similar to that given above, being somewhat more complex due to the additive aggregation frequency given by $a(x,x',y,y') = (x+y) + (x'+y')$. Fernández-Díaz & Gómez-García (2010) developed an analytical solution given by

$$f(x,y,t) = exp\left(-x - y - \theta t - \frac{(x+y)\chi}{2}\right) \times \sum_{k=0}^{\infty} \frac{[xy(x+y)\chi]^k}{(k+1)!(k!)^2}, \quad (30)$$

where $\chi = 1 - e^{-\theta t}$ and $\theta = \mu_{01} + \mu_{10}$, with the initial condition given by

$$f(x,y,t) = \Big(2 - e^{-t}\Big)xye^{-(x+y)}. \quad (31)$$

The parameters for this case were $n = 3$ and $\varepsilon_{abs} = \varepsilon_{rel} = 10^{-3}$. For this case, the speedup was 13 for the PIL algorithm and 12.3 for PIFL algorithm.

Figure 8 summarizes the speedup for the population balance simulations presented here. As expected, these speedups of 12-15 are all comparable to those shown in Figure 6 for $N \sim 700 - 800$. At first, these results might indicate that both parallel cubature algorithm can be utilized in PB-CFD simulations. However, in a PB-CFD simulation, N would be larger than 64k, which is its upper limit shown in Figure 6. Therefore, the PIL algorithm is more adequate for PB-CFD simulations because it would obtain speedups of 50-70, even for small meshes.

7. Conclusions

Two parallel adaptive cubature codes were implemented on GPU in order to accelerate a dual quadrature method for solving the PBE. One was parallelized at the integrand level and other at the integral formula level.

From the performance results, it can be concluded that the speedups of both algorithms are little affected by the required tolerance in the integral values. On the other hand, their speedups increase with the computational cost of the integrand evaluation. Both methods have significant speedups, but the parallel algorithm at the integrand level is superior. In the test cases studied, the maximum speedup achieved was about 70 for compute-intensive integrands and between 6 and 30 for computationally cheap integrands. Furthermore, as the parallelism at the integrand level enables the integration of a larger number of integrand at one instance, it is considered more adequate for CFD simulations.

The parallel adaptive numerical integration algorithms were applied for solving three population balance problems with DuQMoGeM. All results were in excellent agreement with their respective analytical solutions and the speedups factors were between 11 and 15, being both algorithms equivalent in terms of speedup. Theses results were similar to the speedup values obtained with the compute-intensive test integrands for the same number of integrands. Therefore, it can be inferred that a PB-CFD simulation with DuQMoGeM in a 100-volume mesh should achieve a speedup around 70.

Finally, it should be pointed out that the parallel adaptive cubature codes developed in this work can be used to accelerate the computation of multidimensional integrals of a vector of integrands in other applications.

8. Acknowledgments

Paulo L. C. Lage and Fabio P. Santos acknowledge the financial support from CNPq, grants nos. 302963/2011-1, 476268/2009-5 and 140794/2010-7, CAPES, grant no. 0090/12-3, and FAPERJ, grant no. E-26/111.361/2010.

Nomenclature

a	aggregation frequency
A	aggregation matrix

B	number of blocks per grid
b	breakage frequency
c	coefficient in polynomial approximation of f
f	number density distribution function
H	Heaviside step function
L	breakage matrix
M	number of cubature points
N	number of integrands
n	moment order in PBE solution
P	probability density function of particle daughters
T	number of threads per blocks
t	time
x	first particle propert
y	second particle property

Greek letters

δ	Dirac delta function
ε	tolerance
$d\lambda$	measure with a positive definite inner product
μ_k	moment of k order
ϑ	number of particle formed in breakage
ω	weight function
ϕ_k	k order polynomial
Π	moment of k order of P

Subscripts

a	aggregation
b	breakage
max	related to maximum value

Superscripts

ϕ	generalized moment
a	exact solution of standard moment
abs	related to absolute tolerance
rel	related to relative tolerance

References

Attarakih, M. M., Drumm, C., & Bart, H.-J. (2009). Solution of the population balance equation using the sectional quadrature method of moments (SQMOM). *Chemical Engineering Science*, *64*, 742 –52.

Bove, S., Solberg, T., & Hjertager, B. H. (2005). A novel algorithm for solving population balance equations: the parallel parent and daughter classes. Derivation, analysis and testing. *Chemical Engineering Science*, *60*, 1449 –64.

Bull, J., & Freeman, T. (1995). Parallel globally adaptive algorithms for multi-dimensional integration. *Applied Numerical Mathematics*, *19*, 3 – 16.

D'Apuzzo, M., Lapegna, M., & Murli, A. (1997). Scalability and load balancing in adaptive algorithms for multidimensional integration. *Parallel Computing*, *23*, 1199 –210.

Favero, J., & Lage, P. (2012). The dual-quadrature method of generalized moments using automatic integration packages. *Computers & Chemical Engineering*, *38*, 1 – 10.

Fernández-Díaz, J. M., & Gómez-García, G. J. (2010). Exact solution of a coagulation equation with a product kernel in the multicomponent case. *Physica D: Nonlinear Phenomena*, *239*, 279 –90.

Fox, R., Laurent, F., & Massot, M. (2008). Numerical simulation of spray coalescence in an Eulerian framework: Direct quadrature method of moments and multi-fluid method. *Journal of Computational Physics*, *227*, 3058 –88.

Frezzotti, A., Ghiroldi, G., & Gibelli, L. (2011). Solving model kinetic equations on gpus. *Computers & Fluids*, *50*, 136 –46.

Gautschi, W. (2004). *Orthogonal Polynomials - Computation and Approximation*. Oxford Science.

Gelbard, F., & Seinfeld, J. (1978). Coagulation and growth of a multicomponent aerosol. *Journal of Colloid and Interface Science*, *63*, 472 –9.

Genz, A. C. (1984). Testing multidimensional integration routines. *Tools, Methods and Languages for Scientific and Engineering Computation*, (pp. 81 – 94).

Genz, A. C. (1989). Parallel adaptive algorithms for multiple integrals. In Mathematics for large scale computing. *in Lecture Notes in Pure and Applied Mathematics*, (pp. 35 – 47).

Genz, A. C., & Malik, A. A. (1983). An imbedded family of fully symmetric numerical integration rules. *SIAM J. Numer. Anal*, *20*, 580–8.

Gordon, R. (1968). Error bounds in equilibrium statistical mechanics. *Journal of Mathematical Physics*, *9*, 655 –63.

Hahn, T. (2005). Cuba-a library for multidimensional numerical integration. *Computer Physics Communications*, *168*, 78 – 95.

Johnson, S. G. (2008). *Cubature (Multi-dimensional integration)*. http://abinitio.mit.edu/wiki/index.php/Cubature.

Kirk, D. B., & Hwu, W. W. (2010). *Programming Massively Parallel Processors: A Hands-on Approach*. Elsevier.

Kumar, S., & Ramkrishna, D. (1996). On the solution of population balance equations by discretization–I. a fixed pivot technique. *Chemical Engineering Science*, *51*, 1311 –32.

Lage, P. L. C. (2011). On the representation of QMOM as a weighted-residual method the dual-quadrature method of generalized moments. *Computers & Chemical Engineering*, *35*, 2186 –203.

Marchisio, D. L., & Fox, . R. O. (2005). Solution of population balance equations using the direct quadrature method of moments. *Journal of Aerosol Science*, *36*, 43 – 73.

Massot, M., Laurent, F., Kah, D., & Chaisemartin, S. D. (2010). A robust moment method for evalutation of the disappearance of evaporating sprays. *SIAM Journal on Applied Mathematics*, *70*, 3203–34.

Mazzia, A., & Pini, G. (2010). Product Gauss quadrature rules vs. cubature rules in the meshless local Petrov-Galerkin method. *Journal of Complexity*, *26*, 82 – 101.

McGraw, R. (1997). Description of aerosol dynamics by the quadrature method of moments. *Aerosol Science and Technology*, *27*, 255 –65.

NVIDIA Corporation (2010). *NVIDIA CUDA C Programming Guide*. Version 3.2.

Ramkrishna, D. (2000). *Population Balance - Theory and Applications to Particulate Systems in Engineering*. Academic Press, San Diego.

Rudolf, & Schrer (2003). A comparison between (quasi-)Monte Carlo and cubature rule based methods for solving high-dimensional integration problems. *Mathematics and Computers in Simulation*, *62*, 509 –17.

Sanders, J., & Kandrot, E. (2010). *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley.

Schurer, R. (2001). *High-Dimensional Numerical Integration on Parallel Computers*. Master's thesis University of Salzburg, Austria.

Secchi, A. (2007). *DASSLC: User's Manual, a Differential-Algebraic System Solver*. Technical Report UFRGS,Porto Alegre, RS/Brazil. Http://www.enq.ufrgs.br/enqlib/numeric/DASSLC.

Silva, L., & Lage, P. (2011). Development and implementation of a polydispersed multiphase flow model in OpenFOAM. *Computers & Chemical Engineering*, *35*, 2653 –66.

Stroud, A. H. (1971). *Approximate calculation of multiple integrals*. Englewood Chiffs, New Jersey: Prentice-Hall.

Strumendo, M., & Arastoopour, H. (2008). Solution of PBE by MOM in finite size domains. *Chemical Engineering Science*, *63*, 2624 –40.

Yeoh, G. H., & Tu, J. (2010). *Computational Techniques for Multiphase Flows*. Butterworth-Heinemann.

Yuan, C., Laurent, F., & Fox, R. (2012). An extended quadrature method of moments for population balance equations. *Journal of Aerosol Science*, *51*, 1 – 23.

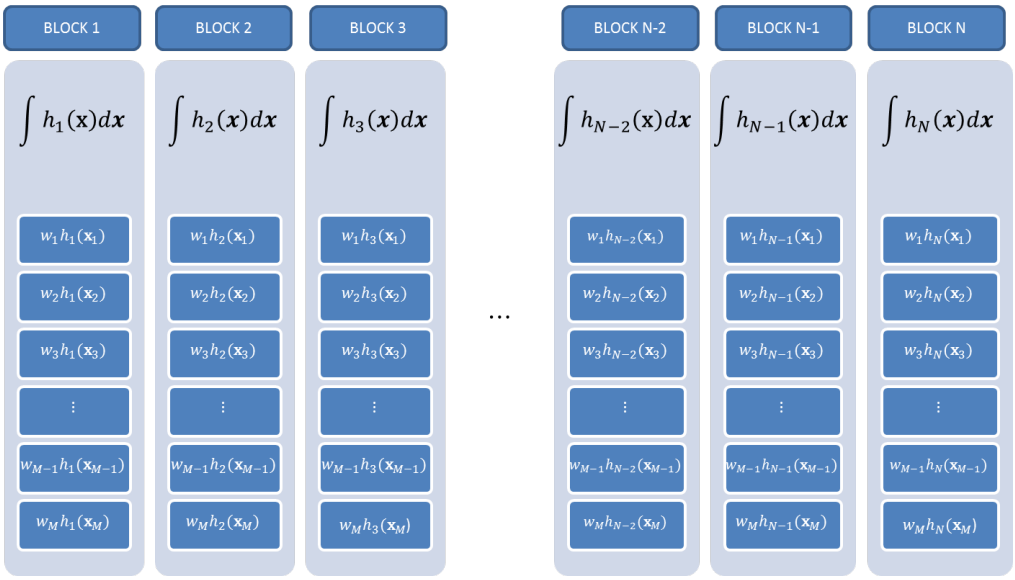


Figure 1: Parallelism at the Integral Formula Level (PIFL) with M cubature points and N integrands.

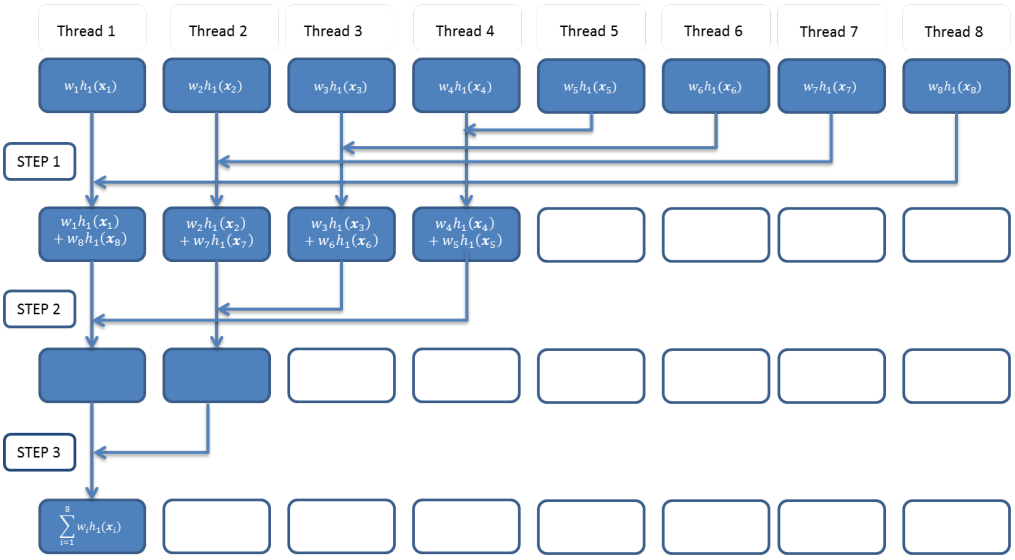


Figure 2: Reduction algorithm for the sum of the cubature points in each block.

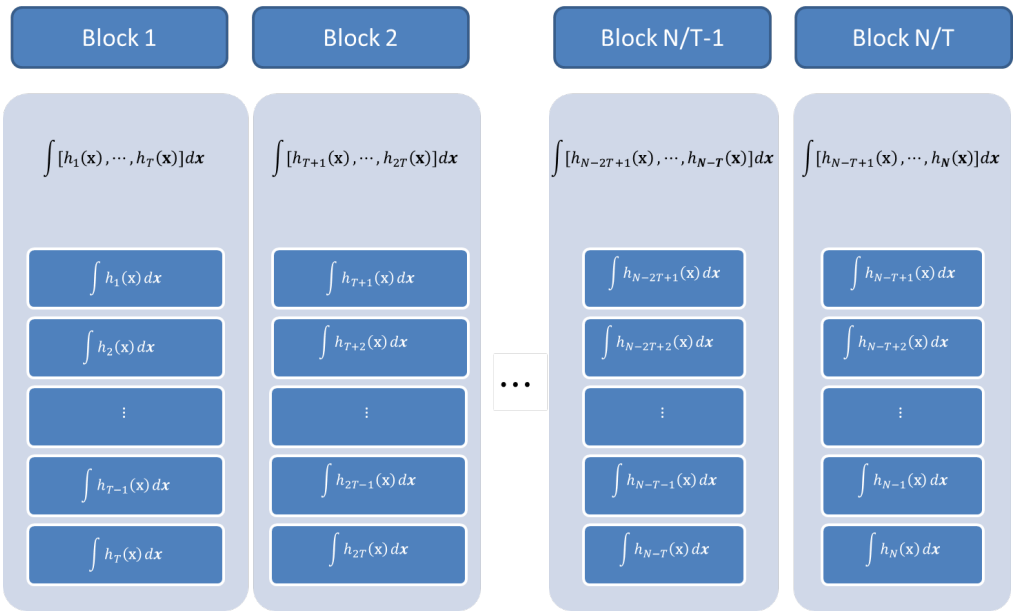


Figure 3: Parallelism at the Integrand Level (PIL) with T integrands and threads per block and a total of N integrands.

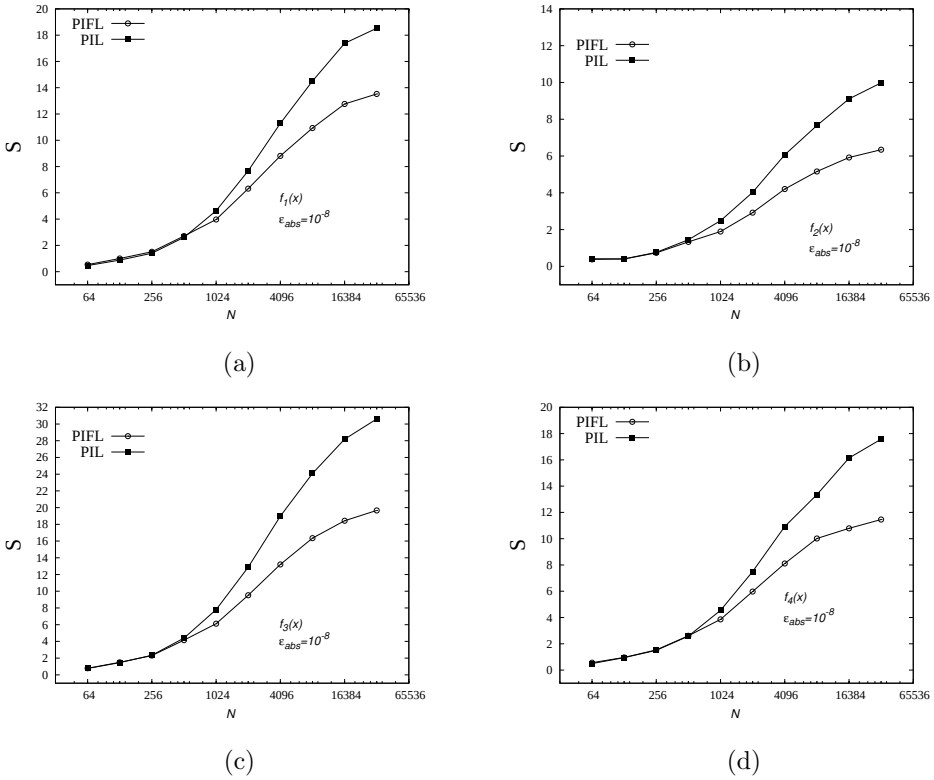


Figure 4: Speedup analysis for N integrands with $\varepsilon_{abs} = \varepsilon_{rel} = 10^{-8}$: (a) $f_1(x)$, (b) $f_2(x)$, (c) $f_3(x)$ and (d) $f_4(x)$.

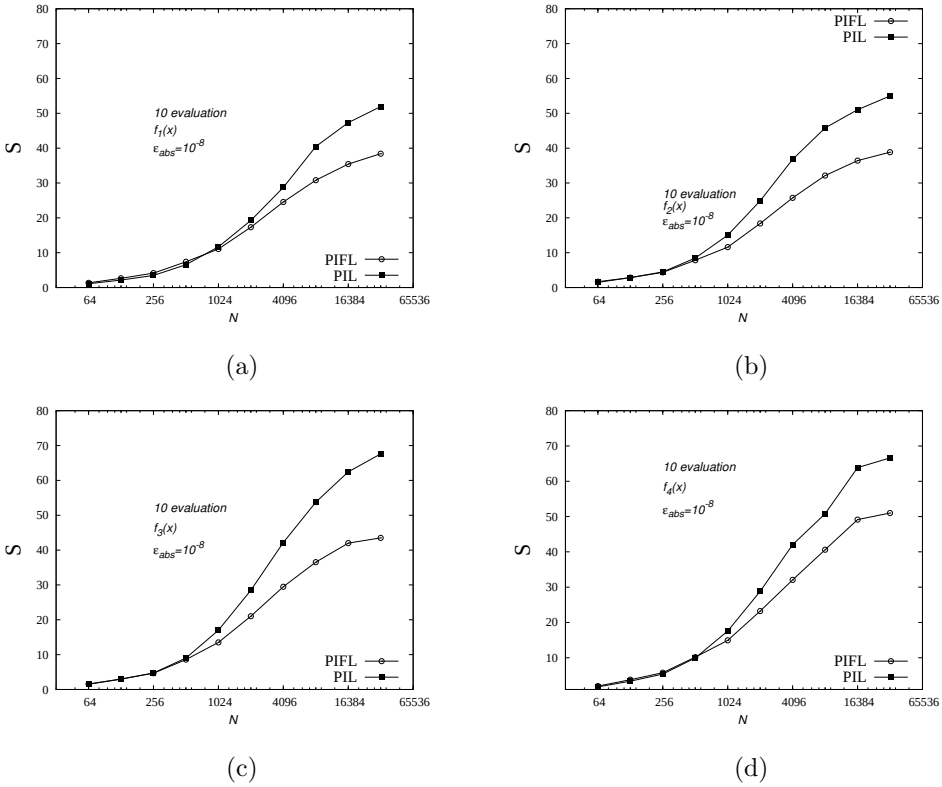


Figure 5: Speedup analysis for N integrands, each one evaluated 10 times at each cubature point, with $\varepsilon_{abs} = \varepsilon_{rel} = 10^{-8}$: (a) $f_1(x)$, (b) $f_2(x)$, (c) $f_3(x)$ and (d) $f_4(x)$.

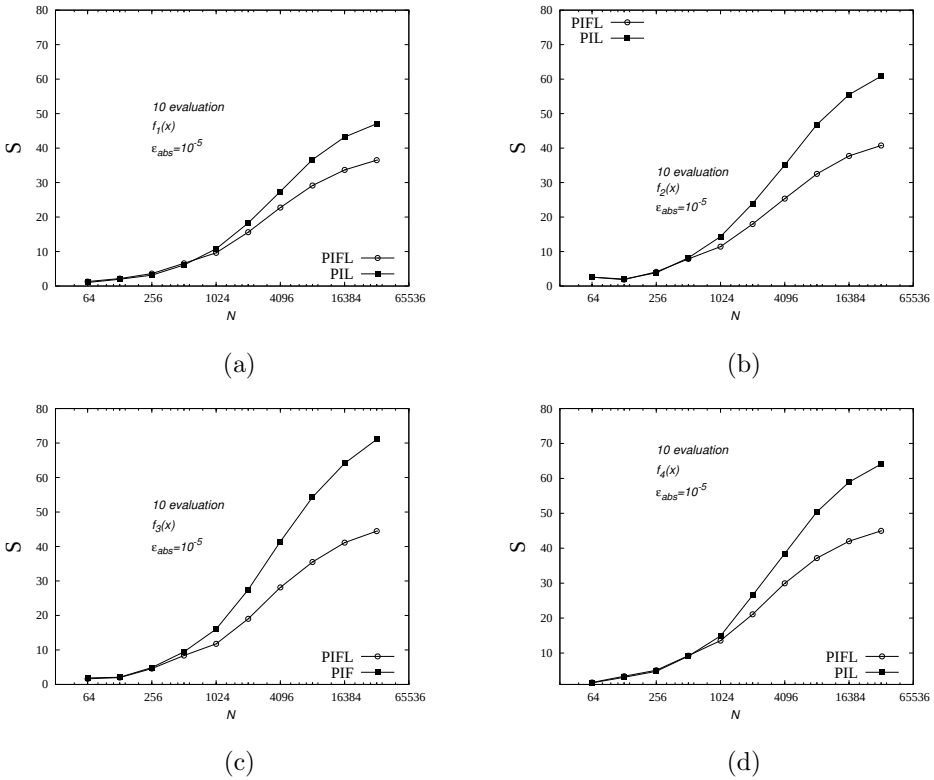
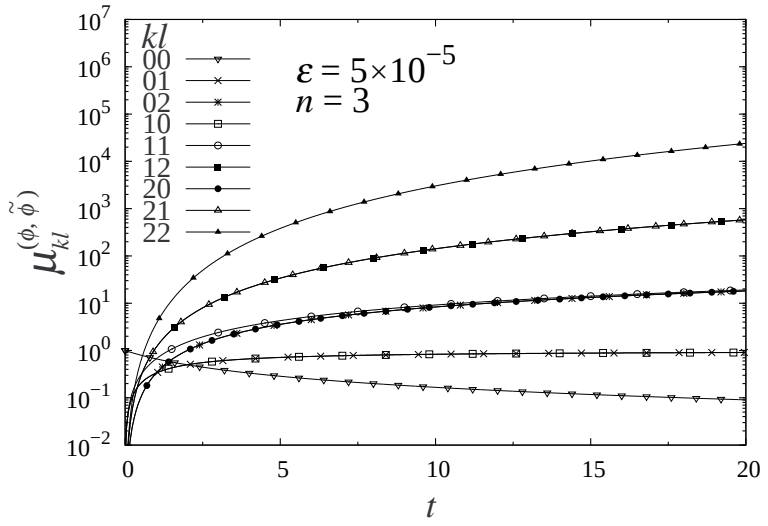
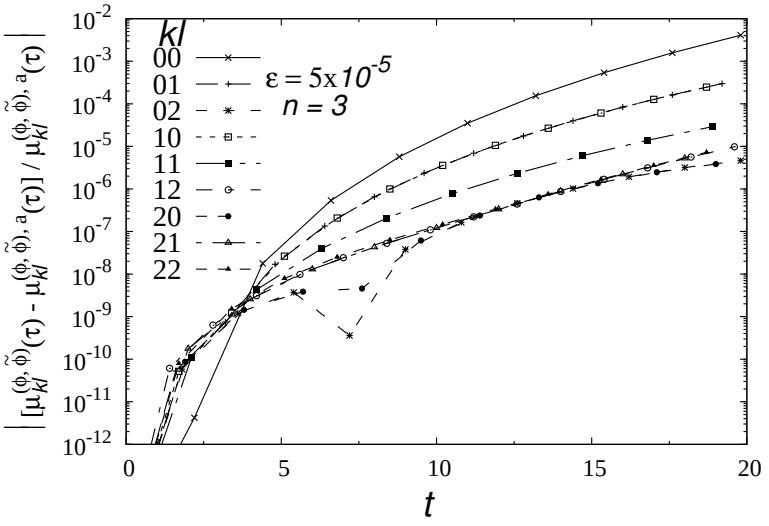


Figure 6: Speedup analysis for N integrands, each one evaluated 10 times at each cubature point, with $\varepsilon_{abs} = \varepsilon_{rel} = 10^{-5}$: (a) $f_1(x)$, (b) $f_2(x)$, (c) $f_3(x)$ and (d) $f_4(x)$.



(a)



(b)

Figure 7: Numerical solution of Case I with the parallel adaptive cubature algorithm: time evolution of (a) the moments and (b) their relative errors.

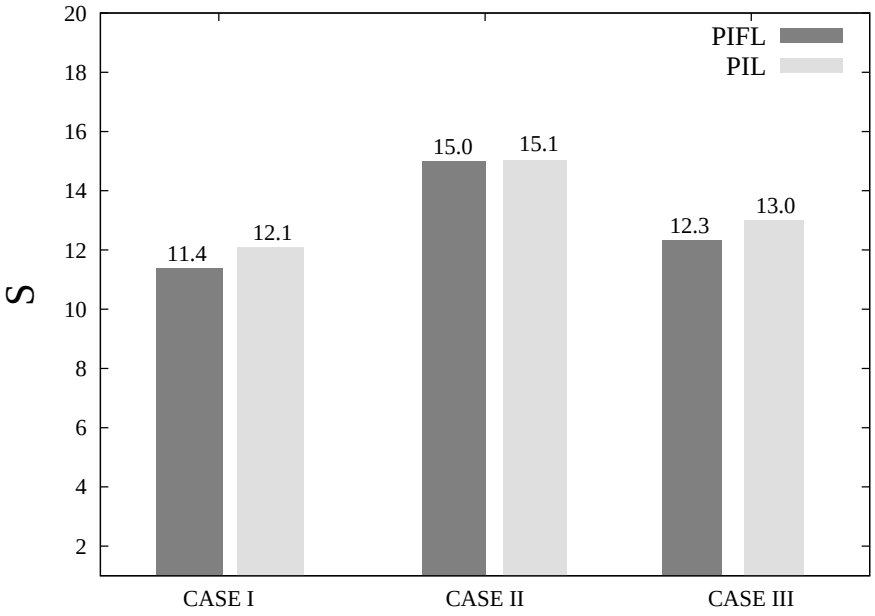


Figure 8: Speedup values for the population balance simulations with DuQ-MoGeM using the PIFL and PIL algorithms.

ANEXO E

Manuscrito IV

Formulação, implementação e avaliação do D^{2}uQMoGeM-FC no OpenFOAM usando algoritmos em GPUs.

Simulation of Polydispersed Multiphase Flows using Dual-Quadrature-Based Moment Methods on GPUs

F. P. Santos[a], J. L. Favero[a], P. L. C. Lage[a,*], I. Senocak[b], L. F. L. R. Silva[c]

[a]*Programa de Engenharia Química COPPE, Universidade Federal do Rio de Janeiro, PO Box 68502, Rio de Janeiro, RJ, 21941-972, Brazil*
[b]*Department of Mechanical and Biomedical Engineering, Boise State University, Boise, ID 83725, USA*
[c]*Escola de Química, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, 21941-909, Brazil.*

Abstract

Polydisperse multiphase flows are present in numerous industrial technologies. Nowadays, it is consensus that the population balance coupled to computational fluid dynamics can be used to predict this sort of flow. In this work, the solution of the population balance equation was performed with the dual-quadrature-based moment methods in their fully conservative forms together with the multi-fluid model. Numerical simulations considering breakage and aggregation were conducted. The main disadvantage of these methods is their high computational cost. In order to address this computational limitation, the dual-quadrature-based moment methods were parallelized on graphics processing units what resulted in a significant acceleration.

Keywords: population balance, OpenFOAM®, dual-quadrature-based moment methods, multiphase flow, particulate systems, CFD

1. Introduction

Multiphase polydisperse flows are present in a plethora of industrial technologies, such as: fluidized beds, flame reactors, precipitation processes and bubble reactors (Ramkrishna, 2000). Modeling them is an important subject

*Corresponding author. Tel.: +55 21 2562 8346; fax: +55 21 25628300.
Email address: paulo@peq.coppe.ufrj.br (P. L. C. Lage)
URL: www.peq.coppe.ufrj.br/pesquisa/tfd (P. L. C. Lage)

that has been developed with considerable impetus by the scientific community (Silva and Lage, 2011; Fox et al., 2008; Fox, 2006; Petitti et al., 2010; Vale and McKenna, 2005; Wulkow et al., 2001; Buffo et al., 2013). Although intensive research efforts have been devoted on this topic, several numerical and modeling shortcomings remain to be overcome.

Three main approaches can be used to model polydisperse multiphase flows: the fully-resolved, the Lagrangian point-particle and the Eulerian-Eulerian models (Yeoh and Tu, 2010). In the fully-resolved approach, each element of the dispersed phase is solved for individually, regarding its shape and interactions with the fluid and the other particles. It produces a very accurate solution but with a very high computational cost.

The Lagrangian point-particle methods consider each element of the dispersed phase individually but they neglect the details of its shape and interactions with other particles (Yeoh and Tu, 2010). Therefore, it can be very inaccurate in some cases. This approach is usually applied for low Stokes number simulations where the particle-particle interactions are not pronounced. Likewise, this limits the range of applications where they can be used.

In the Eulerian-Eulerian (E-E) models, conservation equations are derived for the mean variables of each phase, which are closed by constitutive relations obtained from empirical data (Ishii and Hibiki, 2006). This average procedure yields equations with a low computational cost and reasonable accuracy for these complex problems, being able to solve them in large scales. Nonetheless, these models by themselves cannot capture the particle-particle interactions, such as the aggregation and breakage phenomena. In order to overcome this limitation, a mesoscale framework called population balance model (PBM), which is based on the conservation of the particle number density function (NDF) (Ramkrishna, 2000), can be combined with a multi-fluid flow formulation (Silva and Lage, 2011) to predict both the particle-particle and particle-fluid interactions. Recently, Marchisio and Fox (2013) showed that this mesoscale formulation can be deduced directly from the so-called generalized population balance equation (GPBE) that is basically the conservation equation for the particle number density function including its dependence on all particle and fluid variables that may affect the particulate system behavior.

In spite of the PBM being an interesting framework, there is still a lack of robust and accurate numerical methods for solving the Population Balance Equation (PBE) (Kumar and Ramkrishna, 1996; Bove et al., 2005;

Strumendo and Arastoopour, 2008; Fox et al., 2008; Massot et al., 2010; Attarakih et al., 2009; Lage, 2011; Yuan et al., 2012). Generally, the existing methods can be classified in four main categories: Monte-Carlo methods, discretization methods, quadrature-based moment methods (QBMM) and weighted residual methods. However, not all of them are suitable for solving the so-called PB-CFD coupling, either by efficiency and accuracy or by computational cost. Among these methods, the QBMMs and their variants have been shown to be those best suited for PB-CFD simulations (Marchisio et al., 2003; Fan et al., 2004; Silva et al., 2008; Petitti et al., 2010; Silva and Lage, 2011; Buffo et al., 2013; Petitti et al., 2013). Therefore, this section is focused on the QBMMs and their variants.

All QBMMs calculate the moments of the NDF directly or indirectly. It means that the moment evolution equations can track either the moments of the NDF or the weights and abscissas of its quadrature approximation. This quadrature is an N-point Gauss-Christoffel quadrature rule, that is, a Gaussian quadrature whose weight function is the particle number density function, which is used to approximate the integral terms of the PBE (McGraw, 1997; Marchisio and Fox, 2005; Marchisio et al., 2003; Attarakih, 2013; Yuan and Fox, 2011; Lage, 2011; Yuan et al., 2012; Santos et al., 2013b). This quadrature provides a discretization in the internal variable space, which yields N particle phases in an Eulerian multi-fluid model for polydispersed multiphase flows, being the reason why these methods are considered well-suited for PB-CFD simulations (Silva and Lage, 2011).

The QBMMs that solve for the weights and abscissas do not require to compute the N-point Gauss-Christoffel quadrature except for the discretization of the NDF for the initial or boundary conditions (Marchisio and Fox, 2005). However, for those QBMMs that solve for the NDF moments, this quadrature must be calculated from the computed moments using an adequate moment inversion algorithm at every time step and point of discretization of the physical space. For a univariate NDF, moment inversion can be performed by the Product-Difference algorithm (Gordon, 1968) or the Modified Chebyshev method (Wheeler, 1974). However, there is no general moment inversion method for multivariate NDFs (Favero et al., 2014).

These moment inversion algorithms are ill-conditioned what limits the number of quadrature points that can be used (John and Thein, 2012). This implies that the number of quadrature points may not be enough to accurately calculate the integral source terms of the PBE moments. In order to overcome the quadrature error accumulation, Lage (2011) derived the Dual

Quadrature Method of Generalized Moments (DuQMoGeM). In this method, a quadrature rule of high accuracy can be used to compute the source terms of the PBE and the NDF is still discretized by the Gauss-Christoffel quadrature formula. Later on, Yuan et al. (2012) developed the Extended QMoM (EQMoM), which is also a dual-quadrature-based moment method. The advantage of this method is that it approximates the NDF using the kernel density functions (KDFs), which guarantees the positivity of the NDF.

Favero and Lage (2012) extended the DuQMoGeM to solve homogeneous bivariate population balance problems with error control using an adaptive cubature (AC) (Johnson, 2008), which was very expensive computationally. Santos et al. (2013a) parallelized the AC on GPUs in order to overcome the computational limitation. Based on their results, they stated that GPUs can speedup the Dual Quadrature-Based Moments Methods (DQBMM), making them applicable to PB-CFD simulations. Later on, Santos et al. (2013b) derived the Direct Dual Quadrature Method of Generalized Moments (D^2uQMoGeM). In D^2uQMoGeM, the weights and weighted abscissas are tracked directly as in DQMoM and the quadrature errors are controlled by an adaptive quadrature as in DuQMoGeM. Although it was more accurate than DQMoM, its computational time was higher.

In the PB-CFD coupling using QBMMs, some issues related to the corruption of moments arise (Lage, 2011; Desjardins et al., 2008; Vikas et al., 2011). In order words, numerical errors in the quadrature approximation of the breakage and aggregation terms (Lage, 2011) or the usage of high-order discretization schemes for the advection term (Vikas et al., 2011) can produce a non-realizable set of moments. According to Desjardins et al. (2008), the realizability of the moments is only guaranteed when a first-order discretization scheme is used for the moment advection terms. Recently, Vikas et al. (2011, 2013) derived some high-order numerical schemes for the advection and diffusion terms that are able to guarantee the realizability of the moment set.

Aiming to simulate a polydispersed multiphase flow, Silva and Lage (2011) implemented an Eulerian multi-fluid model coupled with the PBE solution in OpenFOAM® (Weller et al., 1998). However, the formulation was based on the DQMoM (Marchisio and Fox, 2005), which is unable to preserve the moments in some cases. In order to overcome this deficiency, Buffo et al. (2013) developed the DQMoM-FC (*Fully-conservative*), where the advection terms are incorporated in the source term of the DQMoM linear system of equations and discretized using realizable schemes. It is worthwhile to men-

tion that the fully-conservative formulation can also be used in other direct methods like D^{2}uQMoGeM.

In the present work, for the first time, the DuQMoGeM and the *fully-conservative* version of its direct version (D^{2}uQMoGeM-FC) were used to simulate polydisperse multiphase flows using the Eulerian multi-fluid model proposed by Silva and Lage (2011). Both methods were implemented in OpenFOAM® and an adaptive cubature algorithm on GPUs previously developed (Santos et al., 2013b) was used to accelerate the simulations.

The numerical tests were performed in the well-known Backward Facing Step (BFS) geometry using simple, unphysical and computational tractable breakage and aggregation models. Besides, speedup and scalability tests with three different mesh sizes were carried out to evaluate the advantage of using the GPUs co-processors to accelerate the turnaround time of the simulations.

2. Mathematical Model

2.1. Eulerian Multiphase Model

As aforementioned, the Eulerian-Eulerian multiphase equations are derived by averaging the conservation equations. The resulting mass conservation equation for the each phase is written as (Yeoh and Tu, 2010; Drew and Passman, 1999):

$$\frac{\partial(r_\alpha \rho_\alpha)}{\partial t} + \nabla.(r_\alpha \rho_\alpha \mathbf{u}_\alpha) = \Gamma_\alpha, \tag{1}$$

where r_α is the phase fraction, ρ_α is the density, Γ_α is a mass source term and $\mathbf{u}_\alpha$ is the velocity of the phase α, where $\alpha = 0, \ldots, N$, being the continuous phase represented by $\alpha = 0$.

Extending the same procedure to the momentum conservation equation, the following averaged equation can be obtained for each phase:

$$\frac{\partial(r_\alpha \rho_\alpha \mathbf{u}_\alpha)}{\partial t} + \nabla.(r_\alpha \rho_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha) = -r_\alpha \nabla p_\alpha + \nabla\cdot\left(r_\alpha \boldsymbol{\tau}_\alpha^{eff}\right) + r_\alpha \rho_\alpha \mathbf{g} + \Gamma_\alpha \mathbf{u}_{\alpha,I} + \sum_{\substack{\beta=0 \\ \beta\neq\alpha}}^{\beta=N} \mathbf{M}_{\alpha,\beta} \tag{2}$$

where $\boldsymbol{\tau}_\alpha^{eff}$ is the effective stress tensor, which includes both viscous and turbulent effects, and $\Gamma_\alpha \mathbf{u}_{\alpha,I}$ is the momentum source due to the mass source. $\mathbf{M}_{\alpha,\beta}$ represents the interfacial momentum net exchanged from phase β to phase α, which is usually broken into drag, lift, virtual mass forces and the

average interfacial pressure and shear stress at the interface:

$$\mathbf{M}_{\alpha,\beta} = \mathbf{M}^{\mathbf{D}}_{\alpha,\beta} + \mathbf{M}^{\mathbf{L}}_{\alpha,\beta} + \mathbf{M}^{\mathbf{VM}}_{\alpha,\beta} + p_{\alpha,\beta}\nabla r_\alpha - \tau_{\alpha,\beta}\nabla r_\alpha. \tag{3}$$

In the present work, we assumed that all phases have the same pressure field ($p_\alpha = p$) and the effects of the interfacial pressure and shear stress, as well as the momentum source, were neglected. Besides, we considered only the momentum exchange between the continuous and each dispersed phase by drag, lift and virtual mass forces, which are given by Eqs. (4), (5) and (6), respectively:

$$\mathbf{M}^{\mathbf{D}}_{\alpha,\mathbf{0}} = \frac{1}{2} r_\alpha A_\alpha \rho_0 C_{D,\alpha} |\mathbf{u_r}_{,\alpha}| \mathbf{u_r}_{,\alpha}, \tag{4}$$

$$\mathbf{M}^{\mathbf{L}}_{\alpha,\mathbf{0}} = r_\alpha \rho_0 C_{L,\alpha} \mathbf{u_r}_{,\alpha} \times (\nabla \times \mathbf{u_0}), \tag{5}$$

and

$$\mathbf{M}^{\mathbf{VM}}_{\alpha,\mathbf{0}} = r_\alpha \rho_0 C_{VM,\alpha} \left[\frac{D_0 \mathbf{u_0}}{Dt} - \frac{D_\alpha \mathbf{u}_\alpha}{Dt} \right] \tag{6}$$

where A_α is the particle projected area normal to $\mathbf{u_r}_{,\alpha} = \mathbf{u_0} - \mathbf{u}_\alpha$, and $C_{D,\alpha}$, $C_{L,\alpha}$ and $C_{VM,\alpha}$ are the drag, lift and virtual mass coefficients, respectively. We used the Schiller and Naumann's correlation (Schiller and Naumann, 1933) for $C_{D,\alpha}$ and set $C_{L,\alpha} = C_{VM,\alpha} = 0.5$. More details regarding the Eulerian-Eulerian multiphase equations can be found elsewhere (Drew and Passman, 1999; Ishii and Hibiki, 2006; Yeoh and Tu, 2010).

2.2. Population Balance Modeling

The PB-CFD coupling has proved to be well-suited to predict polydispersed multiphase flows (Silva and Lage, 2011; Buffo et al., 2013; Doisneau et al., 2013; Petitti et al., 2013). In this work, the following inhomogeneous univariate PBE with an additive internal variable, x, was considered (Ramkrishna, 2000):

$$\frac{\partial f(x,\mathbf{z},t)}{\partial t} + \nabla_z . \left[\mathbf{u}_d(x,\mathbf{z},\mathbf{y},t) f(x,\mathbf{z},t)\right] + R(x,\mathbf{z},\mathbf{y},t) = S(x,\mathbf{z},\mathbf{y},t), \tag{7}$$

where $\mathbf{z}$ is the physical space coordinates, $\mathbf{y}$ is the vector with the continuous phase properties, $\mathbf{u}_d$ is the particle velocity, $S(x,\mathbf{z},t)$ is an additional source term and $R(x,\mathbf{z},t)$ is the source term given by breakage and aggregation:

$$R(x,\mathbf{z},t) = \mathcal{L}_b f(x,\mathbf{z},t) + \mathcal{L}_a f(x,\mathbf{z},t). \tag{8}$$

The $\mathcal{L}_b$ and $\mathcal{L}_a$ are, respectively, the breakage and aggregation operators, defined by:

$$\mathcal{L}_b f(x,\mathbf{z},t) = b(x,\mathbf{y})f(x,\mathbf{z},t) - \int_x^{x_{max}} \vartheta(x',\mathbf{y})b(x',\mathbf{y})P(x \mid x';\mathbf{y})f(x',\mathbf{z},t)\,\mathrm{d}x', \tag{9}$$

$$\begin{aligned}\mathcal{L}_a f(x,\mathbf{z},t) = &\int_0^{x_{max}} a(x,x',\mathbf{y})f(x,\mathbf{z},t)f(x',\mathbf{z},t)\,\mathrm{d}x' \\ &- \frac{1}{2}\int_0^{x} a(x-x',x',\mathbf{y})f(x-x',\mathbf{z},t)f(x',\mathbf{z},t)\,\mathrm{d}x',\end{aligned} \tag{10}$$

where $b(x,\mathbf{y})$ is the breakage frequency, $\vartheta(x',\mathbf{y})$ is the mean number of particles originated by the breakage of one particle with property x', $P(x|x')$ is the daughter probability distribution function, $a(x,x',\mathbf{y})$ is the aggregation frequency and x_{max} is the upper limit of the internal variable domain, which may be finite or infinite. The daughter probability distribution function has the following properties:

$$\begin{aligned}&\int_0^{x'} P(x \mid x',\mathbf{y})\,\mathrm{d}x = 1; \quad P(x \mid x',\mathbf{y}) = 0 \quad \forall\,(x > x'\,), \\ &\int_0^{x'} xP(x \mid x',\mathbf{y})\,\mathrm{d}x = \frac{x'}{\vartheta(x',\mathbf{y})}.\end{aligned} \tag{11}$$

As represented above, the dependence of a, b, ϑ and P on $(\mathbf{z},t)$ was assumed to occur only through the continuous phase properties, $\mathbf{y}(\mathbf{z},t)$. It is also worthwhile to mention that the diffusive and growth terms of the PBE were not considered in this work. Hereinafter, in order to simplify the notation, these functional dependencies are not explicitly given.

3. Dual Quadrature-Based Moments Methods (DQBMMs)

The class of DQBMMs allows the representation of the NDF by continuous functions and, at the same time, provides a quadrature error control using a second quadrature. Furthermore, the Gauss-Christoffel quadrature yields the discretization for N particle phases for the Eulerian multi-fluid model, being adequate for PB-CFD simulations. There exist three DQBMMs: the EQMoM (Yuan and Fox, 2011), the DuQMoGeM (Lage, 2011) and the Direct DuQMoGeM (Santos et al., 2013a). In this work, we focused on the development of the PB-CFD implementations of the DuQMoGeM and its direct version.

3.1. Dual Quadrature Method of Generalized Moments (DuQMoGeM)

The DuQMoGeM uses an orthogonal polynomial basis for the continuous representation of the NDF (Lage, 2011; Favero and Lage, 2012):

$$f(x,\mathbf{z},t) \backsimeq w(x)\sum_{i=0}^{2N-1} c_i(\mathbf{z},t)\phi_i(x), \tag{12}$$

where ϕ_i is the basis of orthogonal polynomials, w is the weight function and the c_i are the expansion coefficients of the approximation. They can be obtained using the following orthogonality property:

$$\langle \phi_i,\phi_j\rangle_{d\bar{\lambda}(x)} = \int_0^{x_{max}} \phi_i(x)\phi_j(x)w(x)\,\mathrm{d}x = \delta_{ij}\|\phi_i\|^2_{d\bar{\lambda}}, \tag{13}$$

where $d\bar{\lambda}(x) = w(x)dx$ is a known measure. Thus, c_i are calculated by:

$$c_i(\mathbf{z},t) = \frac{\mu_i^{\phi}(\mathbf{z},t)}{\|\phi_i\|^2_{d\bar{\lambda}(x)}}, \qquad i=0,\ldots,2N-1, \tag{14}$$

where μ_i^{ϕ} are the generalized moments of the NDF defined by

$$\mu_i^{\phi}(\mathbf{z},t) = \int_0^{x_{max}} \phi_i(x)f(x,\mathbf{z},t)\,\mathrm{d}x, \qquad i=0,\ldots,2N-1. \tag{15}$$

Applying the generalized moment operator, $\int_0^{x_{max}} \phi_j(x)(.)\,\mathrm{d}x$, to Eq. (7), substituting the approximation given by Eq. (12) into Eq. (7) and using Eq. (14), we can obtain the following equation after some manipulations:

$$\frac{d\mu_j^{\phi}}{dt} + \nabla_z\cdot\left[\int_0^{x_{max}} \mathbf{u}_d f(x,\mathbf{z},t)\phi_j(x)\,\mathrm{d}x\right] + \sum_{i=0}^{2N-1}\sum_{k=0}^{2N-1} A_{jik}\frac{\mu_i^{\phi}}{\|\phi_i\|^2_{d\bar{\lambda}(x)}}\frac{\mu_k^{\phi}}{\|\phi_k\|^2_{d\bar{\lambda}(x)}} + \sum_{i=0}^{2N-1} L_{ji}\frac{\mu_i^{\phi}}{\|\phi_i\|^2_{d\bar{\lambda}(x)}} = s_j \tag{16}$$

where

$$A_{jik} = \int_0^{x_{max}}\int_0^{x_{max}} a(x,x')\left[\phi_j(x) - \frac{1}{2}\phi_j(x+x')\right]\phi_i(x)\phi_k(x')w(x)w(x')dxdx', \tag{17}$$

$$L_{ji} = \int_0^{x_{max}} b(x,t) \left[\phi_j(x) - 2\Pi_j^{\phi}(x)\right] \phi_i(x) w(x) dx, \tag{18}$$

$$\Pi_j^{\phi}(x) = \int_0^{x_{max}} \phi_j(x') P(x'|x) dx', \tag{19}$$

$$s_j = \int_0^{x_{max}} \phi_j(x) S(x,t) dx \tag{20}$$

and $\vartheta(x') = 2$. The computation of A_{jik} and L_{ji} are detailed in section 3.3, where the PB-CFD coupling is discussed.

The advection term in Eq. (16) must be discretized using the Gauss-Christoffel quadrature discretization of the NDF:

$$f(x, \mathbf{z}, t) \backsimeq \sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{z}, t) \delta(x - x_\alpha(\mathbf{z}, t)), \tag{21}$$

where $\delta(x - x_\alpha)$ are Dirac delta functions, and ω_α and x_α are, respectively, the quadrature weights and abscissas. Using Eq. (21), this advection term may be written as:

$$\nabla_z \cdot \left[\int_0^{x_{max}} \mathbf{u}_d f(x, \mathbf{z}, t) \phi_j(x) \, \mathrm{d}x\right] \backsimeq \overline{F}_j^N = \sum_{\alpha=1}^{N} \nabla_z \cdot \left[\omega_\alpha \mathbf{u}_\alpha \phi_j(x_\alpha)\right], \tag{22}$$

where $\mathbf{u}_d(x_\alpha, \mathbf{z}, \mathbf{y}, t) = \mathbf{u}_\alpha(\mathbf{z}, t)$ is the velocity of dispersed phase α.

Defining,

$$\overline{R}_j^{(N)} = \sum_{i=0}^{2N-1} \sum_{k=0}^{2N-1} A_{jik} \frac{\mu_i^{\phi}}{\|\phi_i\|_{d\bar{\lambda}}^2} \frac{\mu_k^{\phi}}{\|\phi_k\|_{d\bar{\lambda}}^2} + \sum_{i=0}^{2N-1} L_{ji} \frac{\mu_i^{\phi}}{\|\phi_i\|_{d\bar{\lambda}}^2} \tag{23}$$

and using Eq. (22), the moment equations in DuQMoGeM, Eq. (16), can be written as:

$$\frac{\partial \mu_j^{\phi}}{\partial t} = -\overline{R}_j^{(N)} - \overline{F}_j^N + s_j. \tag{24}$$

3.2. Direct Dual Quadrature Method of Generalized Moments (D^2uQMoGeM)

Due to the accuracy of the N-point Gauss-Christoffel quadrature, Eq. (21), it is used in D^2uQMoGeM to exactly calculate the first $2N$ moments of the NDF, leading to (Santos et al., 2013b):

$$\mu_i^{\phi}(\mathbf{z},t) = \sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{z},t)\phi_i(x_\alpha(\mathbf{z},t)) \tag{25}$$

Therefore, combining Eqs. (12), (14) and (21), the NDF approximation can be written as:

$$f(x,\mathbf{z},t) = \frac{w(x)}{\|\phi_i\|^2_{d\bar{\lambda}}} \sum_{i=0}^{2N-1} \left[\sum_{\alpha=1}^{N} \omega_\alpha(\mathbf{z},t)\phi_i(x_\alpha(\mathbf{z},t))\right] \phi_i(x) \tag{26}$$

The D^2uQMoGeM is derived by substituting Eq. (26) into Eq. (7), except into the advective term, which is represented using the Gauss-Christoffel quadrature, Eq. (21). These substitutions lead to the following equation:

$$\begin{aligned} &\sum_{\alpha=1}^{N} \left[\phi_j(x_\alpha) - x_\alpha \phi_j'(x_\alpha)\right] \gamma_\alpha + \sum_{\alpha=1}^{N} \phi_j'(x_\alpha)\theta_\alpha = -\sum_{i=0}^{2N-1} L_{ji} \sum_{\alpha=1}^{N} \frac{\omega_\alpha \phi_i(x_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}} \\ &- \sum_{i=0}^{2N-1}\sum_{k=0}^{2N-1} A_{jik} \sum_{\alpha=1}^{N} \frac{\omega_\alpha \phi_i(x_\alpha)}{\|\phi_i\|^2_{d\bar{\lambda}}} \sum_{l=1}^{N} \frac{\omega_l \phi_k(x_l)}{\|\phi_k\|^2_{d\bar{\lambda}}} = -\overline{R}_j^{(N)} + s_j, \end{aligned} \tag{27}$$

where $\phi_j' = d\phi_j/dx$ and γ_α and θ_α were defined as the source terms of the conservation equations for the weights and weighted abscissas, $\zeta_\alpha = \omega_\alpha x_\alpha$:

$$\frac{\partial \omega_\alpha}{\partial t} + \nabla.(\omega_\alpha u_\alpha) = \gamma_\alpha, \tag{28}$$

$$\frac{\partial \zeta_\alpha}{\partial t} + \nabla.(\zeta_\alpha u_\alpha) = \theta_\alpha. \tag{29}$$

Equations (27), (28) and (29) are the original form of the D^2uQMoGeM (Santos et al., 2013a). Its fully conservative version consists of making the advective term of the moment equations to be part of the source term of the linear system of equations. Besides, the proposed D^2uQMoGeM-FC implementation uses weights and abscissas as the tracked variables. Therefore, the D^2uQMoGeM-FC equations are given by:

$$\frac{\partial w_\alpha}{\partial t} = A_\alpha, \quad \frac{\partial x_\alpha}{\partial t} = B_\alpha, \tag{30}$$

where A_α and B_α are given by the solution of the following linear system of equations:

$$\sum_{\alpha=1}^{N} \phi_j(x_\alpha) A_\alpha + \sum_{\alpha=1}^{N} w_\alpha \phi'_j(x_\alpha) B_\alpha = -\overline{R}_j^{(N)} - \overline{F}_j^N + s_j, \tag{31}$$

where $\overline{F}_j^N$ and $\overline{R}_j^{(N)}$ are given by Eqs. (22) and (23), respectively. When needed, μ_i^ϕ is calculated by Eq. (25).

3.3. PB-CFD coupling with DQBMMs

The PB-CFD coupling using a DQBMM is very similar to that using a QBMM. Again, each Gauss-Christoffel quadrature node is assigned to one dispersed phase α. When x is the particle volume, the weighted abscissa, $\zeta_\alpha = \omega_\alpha x_\alpha$, is the phase fraction. Furthermore, if the particle is assumed to be spherical, the particle diameter can be evaluated from x_α and used to calculate the interfacial force between the continuous and the α dispersed phase in each CFD cell. Figure 1 illustrates briefly how the PB-CFD coupling is done for the DuQMoGeM and Direct DuQMoGeM.

Note that the main feature of this PB-CFD coupling method is its accuracy control by an AC. Unfortunately, the AC increases substantially the DQBMMs computational cost (Favero and Lage, 2012), specially, when it is coupled to CFD simulations. For instance, for the solution of an univariate population balance model including simultaneous aggregation and breakage using four moments, DuQMoGeM needs to compute a total of 80 integrals. When the PBE solution is coupled to a CFD simulation, the number of integrands is equal to 80 times the number of mesh cells.

It should be noted that the computation of A_{jik} and L_{ji} are the bottleneck for both the DuQMoGeM and its direct version. Therefore, in order to accelerated the simulations, the computation of A_{jik} and L_{ji} were performed on GPUs using the parallel algorithm at the Integrand Level proposed by Santos et al. (2013a).

In the present work, the PB-CFD coupling is implemented in OpenFOAM® using the same methodology proposed by Silva and Lage (2011), where more details can be found.

4. MPI and CUDA programing in OpenFOAM®

There exist some successful projects that have linked CUDA® GPU routines to the OpenFOAM® code. The first project that arose was Vratis

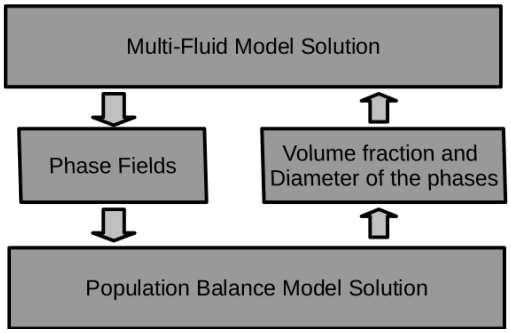


Figure 1: PB-CFD coupling diagram.

SpeedIT plugin (SpeedIT, 2010). It links the Vratis SpeedIT's diagonal preconditioned conjugate gradient (PCG) dynamic library to OpenFOAM® library though a container class. Basically, this C++ container class converts OpenFOAM types to C++ types, such as fields to arrays. Thus, the linear system of discretized equations can be transfer from the CPU to GPUs to be solved in parallel.

Following the same idea, Symscape (2011) provided an OpenFOAM-CUDA plugin with the PCG and PBiCG solvers that are available at the CUSP library (Bell and Garland, 2012). The difference in this case is that the plugin uses the *Thrust* library, which is a CUDA library that mimics the C++ Standard Template Library (Hoberock and Bell, 2010), to transfer data from CPU to GPUs in its container class.

Later on, Combest and Day (2011) released the Cufflink Library, an open-source library for linking numerical methods based on CUDA and OpenFOAM. It also utilizes the *Thrust* and CUSP libraries to solve the linear system derived from OpenFOAM's *lduMatrix* class (Weller et al., 1998). This library is capable to utilize multi-GPUs making use of the coarse-grained parallelism of OpenFOAM. The CFD mesh is, essencially, broken up between computational nodes and each CPU process works together with one or more GPUs.

5. Numerical Procedure

Based on the plugin designed by Combest and Day (2011), we proposed the procedure shown in Figure 2 for D^2uQMoGeM-FC. In our implementation, we used the *Thrust* library to transfer data from the CPU to GPU

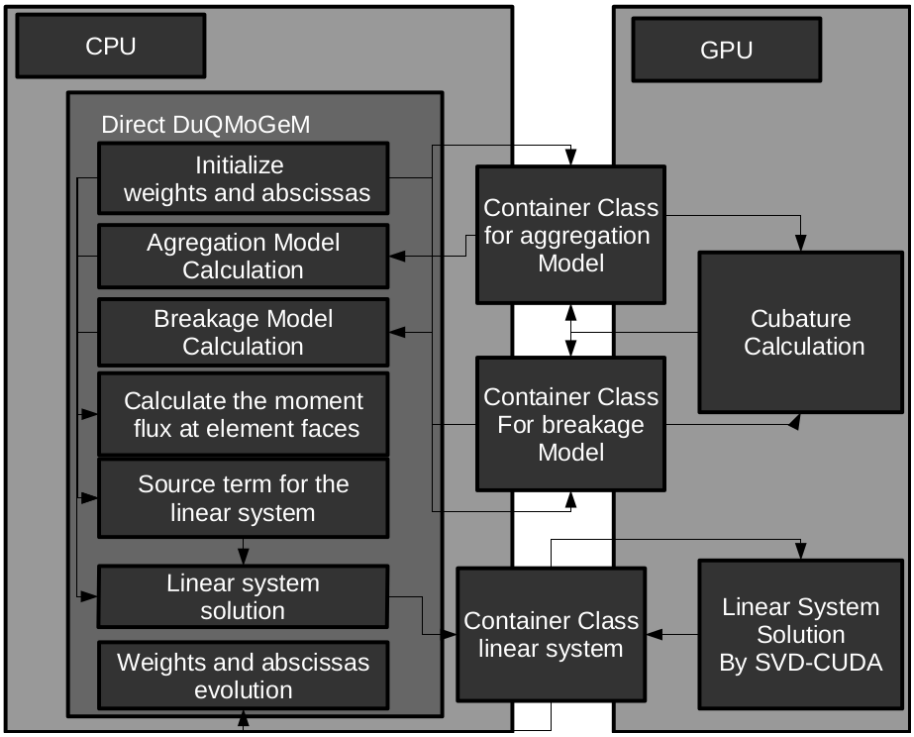


Figure 2: Parallel Direct DuQMoGeM fluxogram.

in our container class. This class transfers the weights and abscissas for computation of the breakage and aggregation term, $\overline{R}_k^{(N)}$, by the parallel AC. Afterwards, $\overline{R}_k^{(N)}$, $\overline{F}_k^N$, weights and abscissas are transfered again to the GPUs in order to construct the systems of linear equations, Eq. (31). Finally, the systems of linear equations are solved using our GPU singular value decomposition (SVD) solver (Santos, 2014).

As mentioned before, the coarse-grained parallelism of OpenFOAM® can be used to allow the multi-GPUs parallelism. OpenFOAM® parallelism is achieved by the decomposition of the computational domain (mesh), which is divided into subdomains. Each subdomain is associated with one computational node, which is a core in a host CPU, that can be assigned to work with one or more GPUs of the same host. For example, if node 1 is assigned to GPU 1, the direct DuQMoGeM systems of linear equations and integrals that belong to its subdomain are computed by GPU 1.

When there are fewer GPUs than computational nodes, the computa-

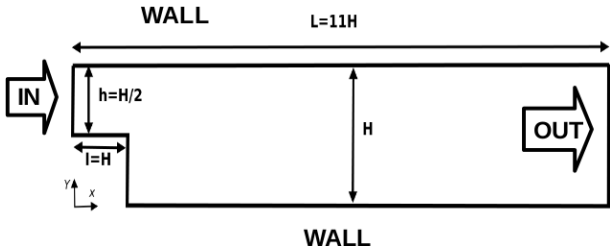


Figure 3: Backward facing step geometry with dimensions and boundaries.

tional work is divided between the available GPUs. For instance, if there are available X nodes and W GPUs, where $X > W$, nodes from 1 to X/W are assigned to work with GPU 1, nodes from $X/W + 1$ to $2X/W$ use GPU 2 and so on.

The PB-CFD code verification with the DQBMM was carried out using a two dimesional Backward Facing Step (BFS) geometry, due to its simplicity and well-defined recirculation zones, where the effects of particle interaction are significant. The BFS is parameterized according to Figure 3, where $L = 11H$, $l = H$ and $h = H/2$, with $H = 0.01$ meters.

For all simulations, a two phase flow mixture of water droplets dispersed in oil was fed at the inlet under the conditions and properties described in Table 1.

Regarding the numerical methods for the simulations, the integral terms of DQBMMs were computed by AC on GPUs (Santos et al., 2013a) using the tolerance defined by

$$\varepsilon = \max\left(\varepsilon_{abs}, \varepsilon_{rel}\left|\mathbf{F}_M\right|\right) \tag{32}$$

where $\mathbf{F}_M$ is the integral over the entire domain. The ε_{rel} and ε_{abs} are the relative and absolute tolerances, whose values were different for each simulation case.

In cases where a linear system of equations has to be solved per node, a GPU parallel SVD solver was employed (Santos, 2014). For temporal discretization, the implicit Euler method was used with a fixed time step of $10^{-4}\,s$. The total time of simulation was 0.3 seconds. The upwind scheme was utilized in the discretization of the moment advective terms to avoid non-realizable moment sets (Desjardins et al., 2008; Vikas et al., 2011).

Table 1: Boundary conditions and phases properties.

Physical Properties	oil	water
ρ (kg/m^3)	900	1000
ν (m^2/s)	10^{-5}	10^{-6}
Boundary Conditions		
Inlet Velocity (m/s)	1.0	
Inlet volume fraction	0.926216	0.073784
Outlet Pressure (N/m^2)	0.0	
Wall	No Slip	No Slip
Initial Conditions		
Volume fraction	0.926216	0.073784

6. Results and Discussions

The simulations were divided into two parts. First, the PB-CFD code with the DQBMM was used to simulate three unrealistic problems with the breakage and aggregation kernels used by McCoy and Madras (2003) and Lage (2011) for homogeneous problems. Lage (2011) and Santos et al. (2013b) have already compared DQBMMs and QBMMs for homogeneous PB models for these cases, concluding that DQBMMs are superior in terms of accuracy. These simulations were aimed at comparing the DQMoM, DuQMoGeM and its direct version. The direct methods were used in their fully-conservative versions. These breakage and aggregation kernels do not depend on $(\mathbf{z}, t)$, but they were computed at all cells of the CFD meshes for every time step to make fair the evaluation of the the computational times.

Note that when D^{2}uQMoGeM-FC uses the regular moments and the FDN is only approximated by Eq. 21, it turns into the DQMoM-FC. Likewise, the DQMoM-FC can be considered a particular case of the D^{2}uQMoGeM-FC.

In the second part, speedup and scalability analysis were carried out by increasing the number of integrands calculated by AC due to the increase in the cells of the CFD mesh. In this step, all simulations analysis were executed on an Intel Xeon I7-3770K CPU at 3.50Hz using a NVIDIA GTX 680 and a NVIDIA GTX TITAN graphic cards using hexaedral two-dimensional meshes with 1400 (mesh 1), 2800 (mesh 2) and 5600 (mesh 3) cells. Finally, some MPI-CUDA simulations were carried out for the finest mesh in order to assess

how the mesh decomposition can increase the speedup.

We defined two different metrics to measure the speedup. S_s is the speedup defined by the ratio between the total execution times of the serial version of the code, T_s, and of the CPU-GPU version, T_g:

$$S_s = \frac{T_s}{T_g}. \tag{33}$$

For the MPI-CUDA simulations, S_p is the acceleration related to the number of CPU processors and GPU co-processors. Using the multi CPU-GPU version of the code, it was defined as the ratio between the total execution times on the configuration with one CPU core and one GPU, T_g, and on a configuration with Q CPU cores and M GPUs co-processors, T_{cg}:

$$S_p = \frac{T_g}{T_{cg}}, \tag{34}$$

When two different GPUs were use in the multi-GPU simulation, T_g was the average computational time. For instance, if T_{g1} is the computational time of the single CPU core-single GPU using the GTX TITAN GPU and T_{g2} is the corresponding time but using the GTX 680 GPU, the average computational time was obtained by:

$$T_g = (T_{g1} + T_{g2})/2. \tag{35}$$

In order to evaluate the relationship between the computational time of the DQMoM-FC and the parallel D^{2}uQMoGeM-FC, the following metric was defined:

$$\Upsilon = T_{du}/T_d \tag{36}$$

where T_d and T_{du} are the computational time of PB-CFD simulations that uses DQMoM-FC and parallel D^{2}uQMoGeM-FC, respectively.

6.1. Cases with simultaneous breakage and aggregation

The breakage and aggregation kernels proposed by McCoy and Madras (2003) were used in PB-CFD simulations to compare the DQMoM, DuQMoGeM and Direct DuQMoGeM results, all of them using the conservation of 4 moments.

The initial size distribution function was defined in terms of droplet volume, $v \in [0, \infty)$, as $F(v, 0)$, with the following zeroth and first order moments:

$$N_T(0) = \int_0^\infty F(v,0)\, dv, \qquad r_d(0) = \int_0^\infty vF(v,0)\, dv, \tag{37}$$

Defining the following dimensionless variables:

$$x = \frac{N_T(0)}{r_d(0)} v, \qquad f(x,0) = \frac{r_d(0)}{[N_T(0)]^2} F(v,0) \tag{38}$$

we can write:

$$\int_0^\infty f(x,0)\, dx = 1, \qquad \int_0^\infty x f(x,0)\, dx = 1 \tag{39}$$

Therefore, for the PB-CFD simulations, the inlet and initial dimensionless distributions were equal and given by:

$$f(x,0) = e^{-x}, \quad x \in [0,\infty) \tag{40}$$

which satisfies Eq. (39). The following values were used in the dimensionless transformation: $N_T(0) = 3.38 \times 10^{12} m^{-3}$ and $r_d(0) = 0.073784$.

The breakage and aggregation frequencies and daughter probability density function were:

$$b(x) = Cx, \quad C = constant, \tag{41}$$

$$a(x) = K, \qquad K = 1, \tag{42}$$

$$P(x|x') = \frac{H(x'-x)}{x'}. \tag{43}$$

For these functions, the dimensionless particle number density at the steady state is given by:

$$\Phi(\infty) = \sqrt{\frac{2C}{K}} \frac{\sqrt{\mu_1(0)}}{\mu_0(0)}. \tag{44}$$

whose value determines if breakage is dominant ($\Phi(\infty) > 1$) or if aggregation is dominant ($\Phi(\infty) < 1$). Once $\Phi(\infty)$ is chosen, C is easily calculated form Eqs. (44) and (40).

Two simulation cases were considered:

- Case 1: $\Phi(\infty) = 0.5$, where aggregation is dominant, and
- Case 2: $\Phi(\infty) = 2.0$, where breakage is dominant.

Assuming spherical particles and noting that the abscissas and weights are also dimensionless variables, the transformations given below were used to obtain the disperse phase fractions and particle diameters:

$$d_\alpha = \left[\frac{6 r_d(0)}{\pi N_T(0)}\right]^{1/3} (x_\alpha)^{1/3}, \tag{45}$$

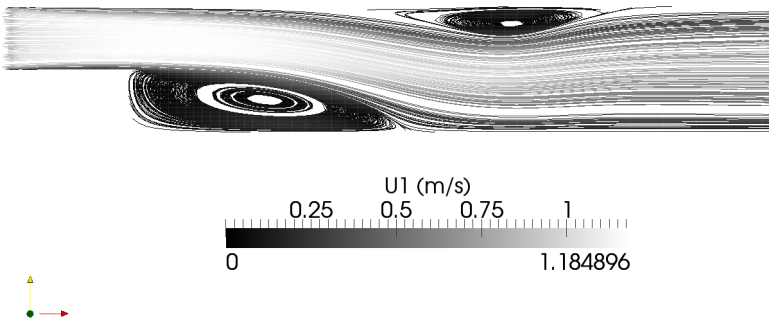


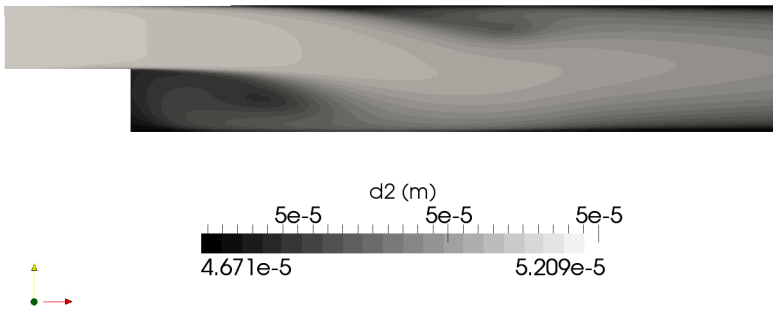
Figure 4: Streamlines colored by the velocity magnitude of the dispersed phase 1 for Case 2 with mesh 3 and $\varepsilon = 10^{-6}$ at $t = 0.3s$ for the D^{2}uQMoGeM-FC.

$$r_\alpha = r_d(0)\zeta_\alpha; \tag{46}$$

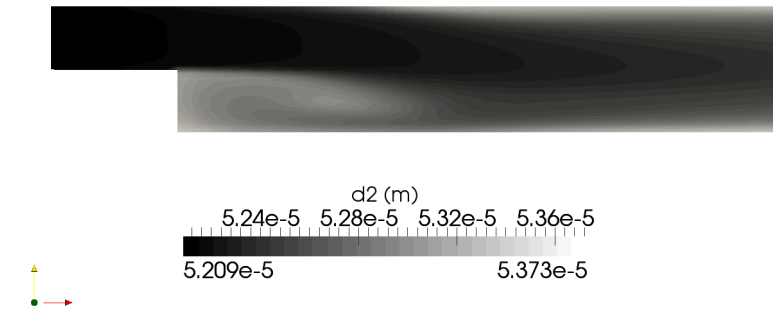
Since $x \in [0, \infty)$, we used the Laguerre polynomial basis to approximate the NDF in the DQBMMs, as in Lage (2011) and Santos et al. (2013b).

Figure 4 shows the streamlines of the dispersed phase 1 at steady state, as obtained by the D^{2}uQMoGeM-FC simulation. It clearly shows the present of recirculation zones in the BFS. In these regions, the residence times of the dispersed phases are longer and the aggregation and breakage effects are higher. Figure 5 demonstrate that particle interactions effects are predominant in the recirculation zones. It can be observed that the particle diameter of the largest class increases in Fig. 5(a) and decreases in the Fig. 5(b). This was expected because cases 1 and 2 are, respectively, aggregation and breakage dominant. Thereby, the present results are physically consistent.

Figure 6 shows the transversal profiles of d_2 and U_2 at the longitudinal position of $2H$ from the inlet obtained using DQMoM-FC, DuQMoGeM and D^{2}DuQMoGeM-FC. It can be seen that the results of DuQMoGeM and Direct DuQMoGeM are equivalent, what was expected because they are equivalent in terms of accuracy. However, the DQMoM results for d_2 are about 15% smaller due to its intrinsic quadrature error. In this sense, one can say



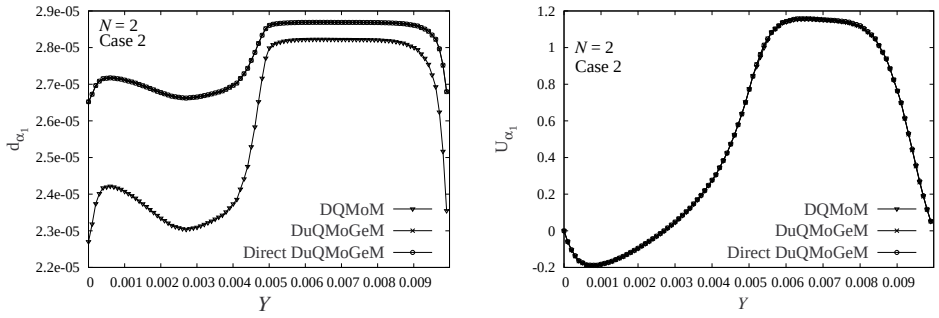
(a)



(b)

Figure 5: Diameter of dispersed phase 2 using mesh 3 and $\varepsilon = 10^{-6}$ at $t = 0.3s$ for the D^{2}uQMoGeM-FC: (a) Case 1 and (b) Case 2.

423 that DuQMoGeM and D^{2}DuQMoGeM-FC are superior in term of accuracy



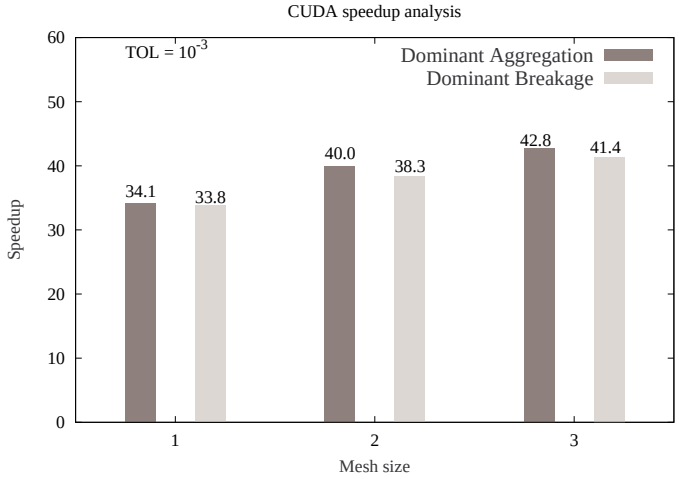
(a) (b)

Figure 6: Profiles on line at longitudinal position $2H$ from the inlet ($X = 2H$) obtained with the DuQMoGeM, DQMoM-FC and D^{2}DuQMoGeM-FC for case 2 using mesh 3 at $t = 0.3s$: (a) d_1 (b) the magnitude of phase velocity u_1.

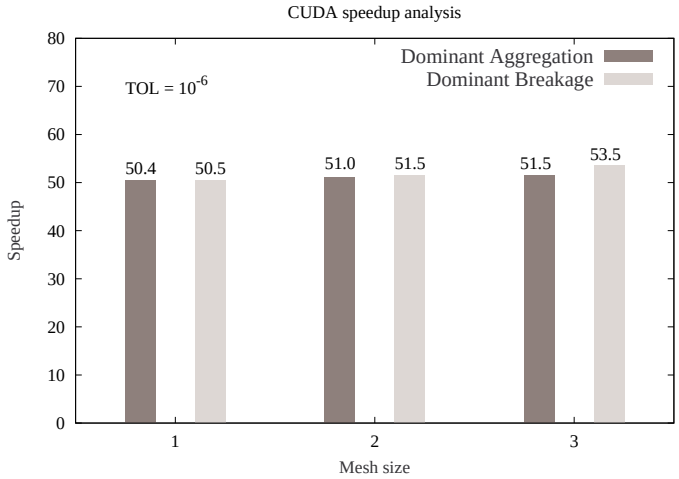
for inhomogeneous PB simulations, because their quadrature error can be controlled.

Figure 7 shows the values of S_s for the PB-CFD simulations with Direct DuQMoGeM-FC using the configuration with one CPU core and one GPU. From this figure, it can be seen that the speedup (S_s) depends on the value of the AC tolerance. This occurs because the AC contribution to the total simulation time increases when the tolerance value is reduced. On the other hand, we can see that S_s depends only slightly on the mesh size, being almost independent of it for the smallest tolerance value. This happens because even for the coarse mesh, the number of integrands is high enough to "saturate" the GPU, giving practically the largest possible speedup. Besides, Figure 7 shows that S_s is at least 34 for $\varepsilon = 10^{-3}$, being about 50 for $\varepsilon = 10^{-6}$, which are quite significant accelerations.

Figure 8 shows the values of S_p for the PB-CFD simulations with Direct DuQMoGeM-FC using the multi-core and multi-GPU configuration. Note that for 2 CPU processors, S_p is closer to 1.8 only when two different GPUs were used. This occurs due to the competition of data transfer between two MPI processes when they share only one GPU. Finally, we can say that the total speedup achieved by 4 CPU processors and 2 GPUs is larger than 100, what enabled this simulation to be performed in a reasonable computational time. Although the parallel implementation had an expressive speed up, we can observe from Table 2 that DQMoM-FC is still faster than Direct DuQMoGeM-FC due to the error control that increases the computational



(a)



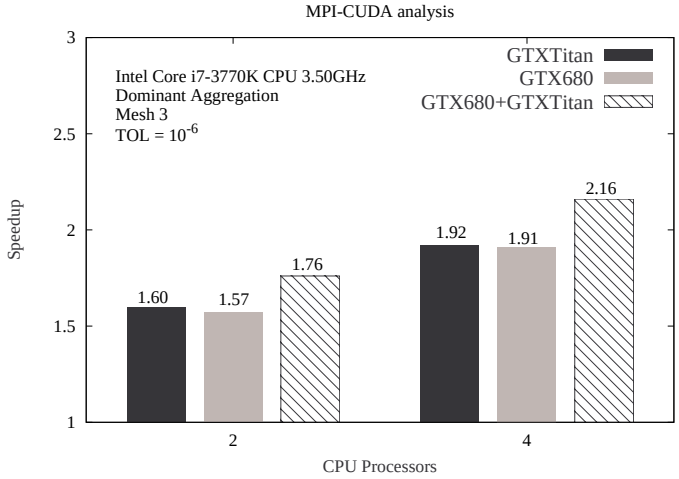
(b)

Figure 7: CUDA speedup analysis for different CFD meshes and AC tolerances for Cases 1 and 2 using GTX TITAN (a) $\varepsilon = 10^{-3}$ (b) $\varepsilon = 10^{-6}$.

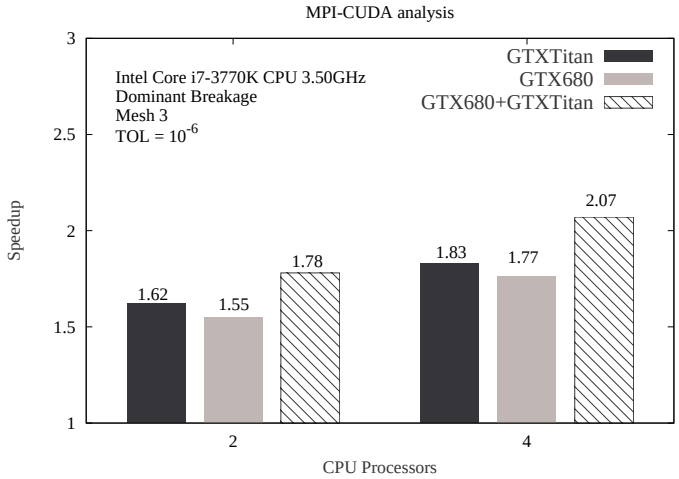
time.

6.2. Pure breakage case

Lage (2011) defined a pure binary breakage problem in the $x \in [0, 1]$



(a)



(b)

Figure 8: MPI-CUDA speedup analysis for different GPUs and number CPU processors: (a) Case 1 (b) Case 2.

domain with:

$$b(x) = x^p, \qquad P(x|x') = H(x' - x)/x' \tag{47}$$

If p is not an integer number, the breakage frequency does not belong to a finite-dimensional polynomial space that yields a higher quadrature error.

Table 2: Computational time relationship, Υ, between DQMoM-FC e D^{2}uQMoGeM-FC CUDA for case 1 with GTX TITAN.

Mesh	$TOL = 10^{-1}$	$TOL = 10^{-3}$	$TOL = 10^{-6}$
1	16	34	120
2	16.3	35	120
3	15.5	38	121

Thus, we have chosen $p = 1/3$ in order to produce this effect in our simulation. As in Lage (2011), a additional source term S was defined as:

- Case 3: $p = \frac{1}{3}$, $S(x,t) = 7e^{-t} - 12 + 7(2 - e^{-t})x^{1/3}$.

and the inlet and initial distribution were equal to:

$$f(x,0) = 2.0 \tag{48}$$

For these simulations, 6 moments were used to solve the PBE. Again, Eqs. (45) and (46) were used to obtain de dispersed phase fractions and particle diameters. However, as $x \in [0,1]$ for this case, the shifted Legendre polynomial basis was utilized to approximate the NDF in the DQBMMs.

As D^{2}uQMoGeM and DuQMoGeM were equivalent, they are not compared for this case. Figures 9(a), 9(b), 9(c) and 9(d) show the results obtained by DQMoM and D^{2}uQMoGeM for d_1, velocity magnitude of the phase 1, u_1, r_1 and r_0 for case 3 and $t = 0.3s$. These figures show clearly that DQMoM and D^{2}uQMoGeM results are slightly different. A expected, the results of D^{2}uQMoGeM are less affected by any inaccuracy in the quadrature approximation of the breakage term, due to the AC error control. Note that the quadrature error in DQMoM did not affect the velocity profile. This means that the quadrature errors were not high enough to damage the multiphase flow behavior. However, the differences between the results obtained from both methods are pronounced in the recirculation zones, where the effects of the quadrature error might be greater or more evident.

Case 3 has a number of integrals smaller than the other two cases. In addition, the AC work is smaller because all of them are single integrals. Therefore, the AC contribution to the total computational time is smaller for this case. This is clearly seen in Figs. 10(a) and 10(b) that show the values of S_s and S_p, respectively, for case 3. The S_s values are around 10 and

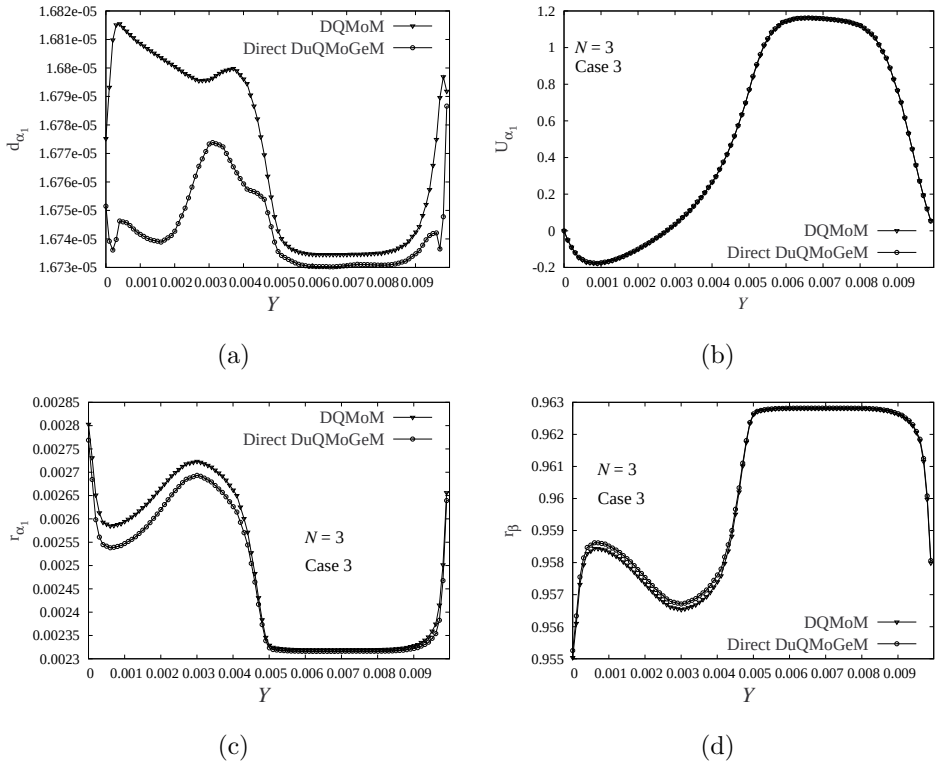
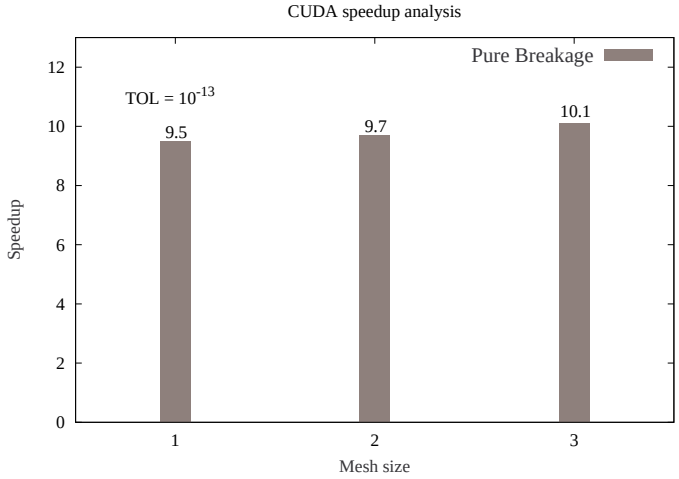


Figure 9: Profiles on line at longitudinal position $2H$ from the inlet ($X = 2H$) obtained with the DQMoM-FC and D^{2}DuQMoGeM-FC for case 3 using mesh 3 at $t = 0.3s$: (a) d_1, (b) the magnitude of phase velocity u_1, (c) r_1 and (d) r_0.

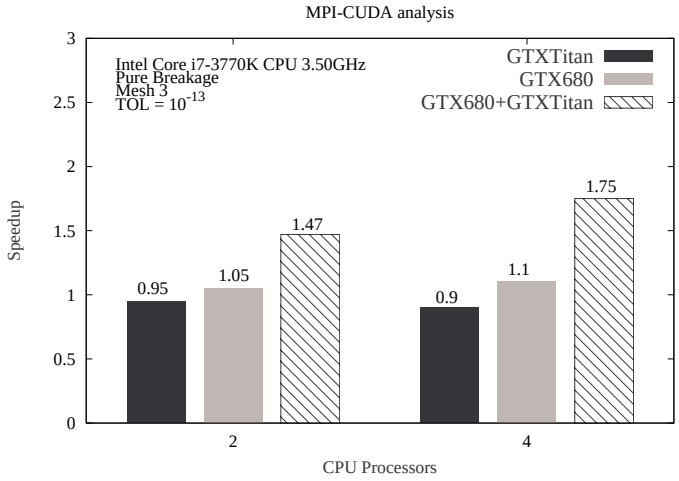
and the S_p values are about 1 when only one GPU is used together with 2 or more CPU processors. When two GPUs were used, some speedup up to 1.75 was observed. Again, DQMoM-FC is faster than Direct DuQMoGeM as described in Table 3. However, in this case, the difference is lower because the aggregation integrals are not included.

7. Conclusions

In this work, a polydispersed multiphase flow model using dual-quadrature-based moments methods were implemented in OpenFOAM® and parallelized on GPUs. In order to verify the code, the PB-CFD coupling with DQBMM were applied to a water-in-oil emulsion flow and their results were compared to those obtained with the same code but using DQMoM-FC. The



(a)



(b)

Figure 10: CUDA speedup analysis for different CFD meshes and tolerances for Cases 3 (a) tolerance of 10^{-13} (b) MPI-CUDA performance analysis.

results were similar. However, it was possible to see the effect of the error control in the automatic quadrature used in the dual-quadrature-based moments methods, which are considered to be more accurate than DQMoM-FC.

Since the adaptive cubature (AC) has to be employed for several integrals

Table 3: Computational time relationship, Υ, between DQMoM-FC e D^{2}uQMoGeM-FC CUDA for case 3.

Mesh	$TOL = 10^{-3}$	$TOL = 10^{-6}$	$TOL = 10^{-13}$
1	5	8.2	20
2	4.7	8.3	23
3	4.3	8.1	21.6

in each CFD cell and at each time step, the computational costs of our simulations are large. The solution for this challenge was to parallelize the computations on GPUs. Herein, the parallel adaptive cubature algorithm developed by Santos et al. (2013a) was used to speedup our simulations. From the performance analysis, it was concluded that the speedup depend on the specified tolerance what can be explained by the increase of AC contribution to the total time of computation. Also, the results showed that the speedup does not increase with the mesh size. This happens because, even for the coarsest mesh used, there were enough integrals to "saturate " the GPU.

For cases including aggregation, the speedup varied from 34 to 50 depending on the value of the AC tolerance. When the CFD mesh was decomposed between 4 CPU processors and 2 GPUs, the speedup for these cases were larger than 100 what makes the PB-CFD simulations with D^{2}DuQMoGeM-FC feasible.

For the pure breakage case, the speedup was just about 10. This occurs because there much fewer integrals to be computed by the AC, whose computational cost became less important when compared to the total computational time. Due to this smaller number of integrals and the coarse meshes used, there were basically no speedup when more than one CPU processor were used with just one GPU. With two GPUs, modest speedup values were obtained.

Finally, we can conclude that the PB-CFD simulations with dual-quadrature-based moments methods would be impractical without GPU parallelization. However, it is worthwhile to mention that the parallel DQBMMs are still more expensive than the QBMMs.

Acknowledgements

The authors acknowledge the financial support of CNPq, grants nos.

302963/2011-1, 140794/2010-7 and 478589/2011-5, and FAPERJ, grant no. E-26/111361/2010. F. P. Santos also acknowledges the support from CAPES for his stay at Boise State University. This work was partially supported by PETROBRAS (SAP no. 4600310999).

Nomenclature

a	aggregation frequency
A	three-dimensional aggregation matrix
b	breakage frequency
c	coefficient in polynomial approximation of f
d	dispersed phase diameter
f	number density distribution function
F	advective flux
H	Heaviside step function
L	breakage matrix
N	number of quadrature points
P	daughter probability density function for particle breakage
S	additional source term in PBE
t	time
u_α	phase velocity, $\alpha = 0, \ldots, N$
w	weight function of inner product
x	internal variable
x_α	quadrature abscissas, $\alpha = 1, \ldots, N$
x_{max}	upper limit of internal variable domain
$\mathbf{z}$	position vector

Greek letters

β	source term in weighted abscissa equation
γ	source term in weight equation
δ	Dirac delta function
ε	value required for the absolute and relative tolerances in adaptive cubature
θ_α	quadrature weighted abscissas, $\alpha = 1, \ldots, N$
ϑ	number of daughter particles formed by breakage
ρ	density
μ_k	k-order monomial moment of f
μ_k^ϕ	k-order generalized moment of f
Π_k	k-order moment of P
ϕ_k	orthogonal polynomial of k order
ω_α	quadrature weights, $\alpha = 1, \ldots, N$
Γ_α	mass source term

Superscripts

a	exact solution
ϕ	generalized moment

Subscripts

a	aggregation
b	breakage
d	related to dispersed phase
α	related to quadrature point associated to a dispersed phase

References

Attarakih, M., 2013. Integral formulation of the population balance equation: Application to particulate systems with particle growth. Computers & Chemical Engineering 48, 1 – 13.

Attarakih, M.M., Drumm, C., Bart, H.J., 2009. Solution of the population balance equation using the sectional quadrature method of moments (SQMOM). Chemical Engineering Science 64, 742 – 752.

Bell, N., Garland, M., 2012. Cusp: Generic parallel algorithms for sparse matrix and graph computations. Version 0.3.0.

Bove, S., Solberg, T., Hjertager, B.H., 2005. A novel algorithm for solving population balance equations: the parallel parent and daughter classes. derivation, analysis and testing. Chemical Engineering Science 60, 1449 – 1464.

Buffo, A., Vanni, M., Marchisio, D., Fox, R., 2013. Multivariate quadrature-based moments methods for turbulent polydisperse gasliquid systems. International Journal of Multiphase Flow 50, 41 – 57.

Combest, D.P., Day, J., 2011. Cufflink: a library for linking numerical methods based on cuda c/c++ with openfoam. Version 0.1.0.

Desjardins, O., Fox, R., Villedieu, P., 2008. A quadrature-based moment method for dilute fluid-particle flows. Journal of Computational Physics 227, 2514 – 2539.

Doisneau, F., Laurent, F., Murrone, A., Dupays, J., Massot, M., 2013. Eulerian multi-fluid models for the simulation of dynamics and coalescence of particles in solid propellant combustion. Journal of Computational Physics 234, 230 – 262.

Drew, D.A., Passman, S.L., 1999. Theory of multicomponent fluids. Applied mathematical sciences, Springer, New York.

Fan, R., Marchisio, D.L., Fox, R.O., 2004. Application of the direct quadrature method of moments to polydisperse gassolid fluidized beds. Powder Technology 139, 7 – 20.

Favero, J., Lage, P., 2012. The dual-quadrature method of generalized moments using automatic integration packages. Computers & Chemical Engineering 38, 1 – 10.

Favero, J.L., Silva, L.F.L., Lage, P.L., 2014. Comparison of methods for multivariate moment inversionintroducing the independent component analysis. Computers & Chemical Engineering 60, 41 – 56.

Fox, R., 2006. Bivariate direct quadrature method of moments for coagulation and sintering of particle populations. Journal of Aerosol Science 37, 1562 – 1580.

Fox, R., Laurent, F., Massot, M., 2008. Numerical simulation of spray coalescence in an eulerian framework: Direct quadrature method of moments and multi-fluid method. Journal of Computational Physics 227, 3058 – 3088.

Gordon, R., 1968. Error bounds in equilibrium statistical mechanics. Journal of Mathematical Physics 9, 655 – 663.

Hoberock, J., Bell, N., 2010. Thrust: A parallel template library. Version 1.7.0.

Ishii, M., Hibiki, T., 2006. Thermo-fluid dynamics of two-phase flow. Birkhäuser.

John, V., Thein, F., 2012. On the efficiency and robustness of the core routine of the quadrature method of moments (QMOM). Chemical Engineering Science 75, 327 – 333.

Johnson, S.G., 2008. Cubature (Multi-dimensional integration). http://abinitio.mit.edu/wiki/index.php/Cubature.

Kumar, S., Ramkrishna, D., 1996. On the solution of population balance equations by discretization–i. a fixed pivot technique. Chemical Engineering Science 51, 1311 – 1332.

Lage, P.L.C., 2011. On the representation of QMOM as a weighted-residual method the dual-quadrature method of generalized moments. Computers & Chemical Engineering 35, 2186 – 2203.

Marchisio, D., Fox, R., 2013. Computational Models for Polydisperse Particulate and Multiphase Systems. Cambridge University Press, Cambridge, United Kingdom.

Marchisio, D.L., Fox, .R.O., 2005. Solution of population balance equations using the direct quadrature method of moments. Journal of Aerosol Science 36, 43 – 73.

Marchisio, D.L., Vigil, R.D., Fox, R.O., 2003. Quadrature method of moments for aggregation-breakage processes. Journal of Colloid and Interface Science 258, 322 – 334.

Massot, M., Laurent, F., Kah, D., Chaisemartin, S.D., 2010. A robust moment method for evalutation of the disappearance of evaporating sprays. SIAM Journal on Applied Mathematics 70, 3203–3234.

Santos, F., 2014. Desenvolvimento de implementação computacional multiplataforma usando GPUs para a solução de problemas envolvendo balano populacional. Ph.D. thesis. Universidade Federal do Rio de Janeiro, PEQ/COPPE, Brasil, RJ.

McCoy, B.J., Madras, G., 2003. Analytical solution for a population balance equation with aggregation and fragmentation. Chemical Engineering Science 58, 3049 – 3051.

McGraw, R., 1997. Description of aerosol dynamics by the quadrature method of moments. Aerosol Science and Technology 27, 255 – 265.

Petitti, M., Nasuti, A., Marchisio, D.L., Vanni, M., Baldi, G., Mancini, N., Podenzani, F., 2010. Bubble size distribution modeling in stirred gasliquid reactors with QMOM augmented by a new correction algorithm. AIChE Journal 56, 36–53.

Petitti, M., Vanni, M., Marchisio, D.L., Buffo, A., Podenzani, F., 2013. Simulation of coalescence, break-up and mass transfer in a gasliquid stirred tank with CQMOM. Chemical Engineering Journal 228, 1182 – 1194.

Ramkrishna, D., 2000. Population Balance - Theory and Applications to Particulate Systems in Engineering. Academic Press, San Diego.

Santos, F., Favero, J., Lage, P., 2013a. Solution of the population balance equation by the direct dual quadrature method of generalized moments. Chemical Engineering Science 101, 663 – 673.

Santos, F.P., Senocak, I., Favero, J.L., Lage, P.L., 2013b. Solution of the population balance equation using parallel adaptive cubature on GPUs. Computers & Chemical Engineering 55, 61 – 70.

Schiller, L., Naumann, A.Z., 1933. Über die grundlegenden Berechnungen bei der Schwerkraftaufbereitung. Ver. Deut. Ing. 77, 318–320.

Silva, L., Lage, P., 2011. Development and implementation of a polydispersed multiphase flow model in OpenFOAM. Computers & Chemical Engineering 35, 2653 – 2666.

Silva, L.F.L.R., Damian, R.B., da Cunha Lage, P.L., 2008. Implementation and analysis of numerical solution of the population balance equation in CFD packages. Computers & Chemical Engineering 32, 2933 – 2945.

SpeedIT, V., 2010. Vratis speedit plugin.

Strumendo, M., Arastoopour, H., 2008. Solution of PBE by MOM in finite size domains. Chemical Engineering Science 63, 2624 – 2640.

Symscape, 2011. Gpu v1.0 linear solver library for openfoam. Version 1.0.

Vale, H., McKenna, T., 2005. Modeling particle size distribution in emulsion polymerization reactors. Progress in Polymer Science 30, 1019 – 1048.

Vikas, V., Wang, Z., Passalacqua, A., Fox, R., 2011. Realizable high-order finite-volume schemes for quadrature-based moment methods. Journal of Computational Physics 230, 5328 – 5352.

Vikas, V., Wang, Z.J., Fox, R.O., 2013. Realizable high-order finite-volume schemes for quadrature-based moment methods applied to diffusion population balance equations. Journal of Computational Physics 249, 162–179.

Weller, H.G., Tabor, G., Jasak, H., Fureby, C., 1998. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in Physics 12, 620–631.

686 Wheeler, J., 1974. Modified moments and gaussian quadrature. Rocky Moun-
687 tain Journal of Mathematics 4, 287 – 296.

688 Wulkow, M., Gerstlauer, A., Nieken, U., 2001. Modeling and simulation of
689 crystallization processes using Parsival. Chemical Engineering Science 56,
690 2575 – 2588.

691 Yeoh, G.H., Tu, J., 2010. Computational Techniques for Multiphase Flows.
692 Butterworth-Heinemann, Oxford.

693 Yuan, C., Fox, R., 2011. Conditional quadrature method of moments for
694 kinetic equations. Journal of Computational Physics 230, 8216 – 8246.

695 Yuan, C., Laurent, F., Fox, R., 2012. An extended quadrature method of
696 moments for population balance equations. Journal of Aerosol Science 51,
697 1 – 23.