



**UNIVERSIDADE FEDERAL DE JUIZ DE FORA
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA**

Marcos Vinícius Teixeira

**ESTUDOS SOBRE A IMPLEMENTAÇÃO ONLINE DE UMA TÉCNICA DE
ESTIMAÇÃO DE ENERGIA NO CALORÍMETRO HADRÔNICO DO ATLAS
EM CENÁRIOS DE ALTA LUMINOSIDADE**

Dissertação de Mestrado

Juiz de Fora

2015

Marcos Vinícius Teixeira

**ESTUDOS SOBRE A IMPLEMENTAÇÃO ONLINE DE UMA TÉCNICA DE
ESTIMAÇÃO DE ENERGIA NO CALORÍMETRO HADRÔNICO DO ATLAS
EM CENÁRIOS DE ALTA LUMINOSIDADE**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, PPEE, da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos.

Orientador: D.Sc. Augusto Santiago Cerqueira.

Co-Orientador: D.Sc. Luciano Manhães de A. Filho.

Juiz de Fora

2015

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Teixeira, Marcos Vinícius .
Estudos sobre a Implementação Online de uma Técnica de
Estimação de Energia no Calorímetro Hadrônico do ATLAS em
Cenários de Alta Luminosidade / Marcos Vinícius Teixeira. --
2015.
99 p. : il.

Orientador: Augusto Santiago Cerqueira
Coorientador: Luciano Manhães de Andrade Filho
Dissertação (mestrado acadêmico) - Universidade Federal de
Juiz de Fora, Faculdade de Engenharia. Programa de Pós-
Graduação em Engenharia Elétrica, 2015.

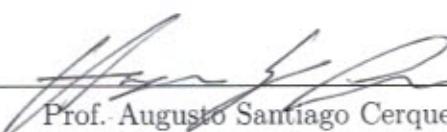
1. Estimação de Energia. 2. Melhor Estimador Linear
Imparcial. 3. Filtragem Ótima. 4. Empilhamento de Sinais. 5.
Métodos Iterativos. I. Cerqueira, Augusto Santiago, orient.
II. Filho, Luciano Manhães de Andrade, coorient. III. Título.

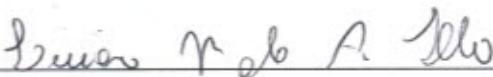
ESTUDOS SOBRE A IMPLEMENTAÇÃO ONLINE DE UMA TÉCNICA DE
ESTIMAÇÃO DE ENERGIA NO CALORÍMETRO HADRÔNICO DO ATLAS
EM CENÁRIOS DE ALTA LUMINOSIDADE

Marcos Vinícius Teixeira

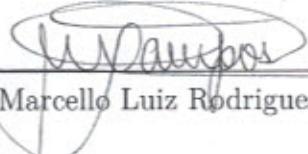
DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA
DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA (PPEE) DA
UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE
EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:


Prof. Augusto Santiago Cerqueira, D.Sc.


Prof. Luciano Manhães de A. Filho, D.Sc.


Prof. Rafael Antunes Nóbrega, D.Sc.


Prof. Marcello Luiz Rodrigues de Campos, Ph.D.

JUIZ DE FORA, MG – BRASIL
AGOSTO DE 2015

*Dedico esta conquista aos meus
pais e a minha esposa.*

Agradecimentos

Primeiramente gostaria de agradecer a Deus por ter me dado saúde ao longo da vida e por permitir a conclusão desta fase do trabalho. Em especial, agradeço a minha esposa Juliana, por sempre compartilhar dos meus sonhos e por me apoiar com muito amor e dedicação. Aos meus pais por tudo. Sem vocês seria muito difícil chegar até aqui!

Aos meus orientadores prof. Augusto Santiago Cerqueira e prof. Luciano Manhães de Andrade Filho pela confiança no meu trabalho. Ao prof. Augusto, agradeço a oportunidade no PPEE e por me permitir trabalhar neste ambiente de colaboração com CERN. Ao prof. Luciano que participou ativamente deste projeto junto ao prof. Augusto, agradeço a dedicação e aos grandes ensinamentos dentro do ambiente de pesquisa. À toda colaboração ATLAS no CERN, em especial ao TileCal que de forma direta ou indireta contribuíram para o desenvolvimento deste trabalho. À Coordenadoria de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio dado a este trabalho. Um agradecimento especial, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), através dos programas RHAE Pesquisador na Empresa e StartUP Barsil, que ao longo do desenvolvimento deste trabalho selecionou dois projetos pessoais, apoiando financeiramente minhas pesquisas para desenvolvimento de tecnologias que visam contribuir significativamente para o desenvolvimento científico e tecnológico e a inovação no País. Obrigado pelo apoio aos meus projetos e por permitir coordenar, ainda como aluno de mestrado e nos próximos anos como doutorando, profissionais nos níveis de graduação, mestres e doutores. Ainda sim que em áreas distintas, durante esse período do mestrado pude concluir que fiz as melhores escolhas. Com o apoio de todos, sinto-me extremamente motivado a continuar minha vida acadêmica sabendo que meu crescimento profissional está caminhando no mesmo sentido.

Por fim, agradeço aos colegas e amigos do LAPTEL, e a outras pessoas que, de alguma forma, colaboraram com este projeto.

Resumo da Dissertação apresentada à PPEE como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESTUDOS SOBRE A IMPLEMENTAÇÃO ONLINE DE UMA TÉCNICA DE ESTIMAÇÃO DE ENERGIA NO CALORÍMETRO HADRÔNICO DO ATLAS EM CENÁRIOS DE ALTA LUMINOSIDADE

Marcos Vinícius Teixeira

Agosto/2015

Orientadores: Augusto Santiago Cerqueira
Luciano Manhães de A. Filho

Programa: Engenharia Elétrica

Este trabalho tem como objetivo o estudo de técnicas para a estimação da amplitude de sinais no calorímetro de telhas (TileCal) do ATLAS no LHC em cenários de alta luminosidade. Em alta luminosidade, sinais provenientes de colisões adjacentes são observados, ocasionando o efeito de empilhamento de sinais. Neste ambiente, o método **COF** (do inglês, *Constrained Optimal Filter*), apresenta desempenho superior ao algoritmo atualmente implementado no sistema. Entretanto, o **COF** requer a inversão de matrizes para o cálculo da pseudo-inversa de uma matriz de convolução, dificultando sua implementação *online*. Para evitar a inversão de matrizes, este trabalho apresenta métodos iterativos, para adaptação do **COF**, que resultam em operações matemáticas simples. Baseados no Gradiente Descendente, os resultados demonstraram que os algoritmos são capazes de estimar a amplitude de sinais empilhados, além do sinal de interesse com eficiência similar ao **COF**. Visando a implementação *online*, este trabalho apresenta estudos sobre a complexidade dos métodos iterativos e propõe uma arquitetura de processamento em FPGA. Baseado em uma estrutura sequencial e utilizando lógica aritmética em ponto fixo, os resultados demonstraram que a arquitetura desenvolvida é capaz executar o método iterativo, atendendo os requisitos de tempo de processamento exigidos no TileCal.

Palavras-chave: Estimação de Energia, Melhor Estimador Linear Imparcial, Filtragem Ótima, Empilhamento de Sinais, Métodos Iterativos.

Abstract of Dissertation presented to PPEE/UFJF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

STUDIES ABOUT ONLINE IMPLEMENTATION OF A TECHNIQUE OF
ENERGY ESTIMATION IN THE ATLAS HADRON CALORIMETER FOR
HIGH LUMINOSITY ENVIRONMENT

Marcos Vinícius Teixeira

August/2015

Advisors: Augusto Santiago Cerqueira
Luciano Manhães de A. Filho

Department: Electrical Engineering

This work aims at the study of techniques for online energy estimation in the ATLAS hadronic Calorimeter (TileCal) on the LHC collider. During further periods of the LHC operation, signals coming from adjacent collisions will be observed within the same window, producing a signal superposition. In this environment, the energy reconstruction method **COF** (Constrained Optimal Filter) outperforms the algorithm currently implemented in the system. However, the **COF** method requires an inversion of matrices and its online implementation is not feasible. To avoid such inversion of matrices, this work presents iterative methods to implement the **COF**, resulting in simple mathematical operations. Based on the Gradient Descent, the results demonstrate that the algorithms are capable of estimating the amplitude of the superimposed signals with efficiency similar to **COF**. In addition, a processing architecture for FPGA implementation is proposed. The analysis has shown that the algorithms can be implemented in the new TilaCal electronics, reaching the processing time requirements.

Keywords: Energy Estimation, Best Linear Unbiased Estimator, Optimal Filter, Signal PileUp, Iterative optimization.

Sumário

Lista de Figuras	x
Lista de Tabelas	xii
1 Introdução	1
1.1 Motivação	1
1.2 O que foi Feito	3
1.3 Visita Guiada	4
2 CERN, LHC e o ATLAS	5
2.1 O CERN e o LHC	5
2.2 O ATLAS	7
2.2.1 Calorimetria	9
2.2.2 Calorímetro Hadrônico de Telhas (TileCal)	10
2.3 Os Efeitos da Alta Luminosidade e o Plano de Atualização	17
2.3.1 Fases do Plano de Atualização	18
2.3.2 Super Read Out Driver	19
3 Revisão Bibliográfica	21
3.1 Melhor Estimador Linear Imparcial - BLUE	21
3.1.1 Determinando os pesos do BLUE	22
3.1.2 BLUE para um vetor de parâmetros	25
3.2 Método OF	28
3.3 Método COF	30
4 Propostas para Implementação <i>Online</i> do COF	34
4.1 Adaptação do COF através de Métodos Iterativos	34
4.1.1 Gradiente Descendente - GD	35
4.1.2 Gradiente Descendente com Convergência Dinâmica - GDD .	37
4.1.3 Gradiente Conjugado - GC	39
4.2 Simulação dos Métodos Iterativos	48
4.2.1 Ambiente de Simulação	48

4.2.2	Simulação do método GD	49
4.2.3	Simulação do método GDD	53
4.2.4	Simulação do método GC	55
5	Implementação em FPGA	61
5.1	Análise de Complexidade	61
5.2	Arquitetura do Processador em FPGA	68
5.2.1	Controle e Processamento	71
5.2.2	Aritmética de Ponto Fixo	72
5.2.3	Memórias de Armazenamento	73
5.3	Resultados do Processador em FPGA	74
6	Conclusões e Trabalhos Futuros	77
6.1	Trabalhos Futuros	78
	Referências Bibliográficas	80
	A Coordenadas do ATLAS	84

Listas de Figuras

2.1	Vista aérea do Complexo do CERN. [Extraído de [5]].	6
2.2	O anel do LHC e seus quatro detectores principais [extraído de [10]]. . .	7
2.3	ATLAS e seus principais detectores e dimensões [extraído de [11]]. . .	8
2.4	Coordenadas do ATLAS [extraído de [13]].	8
2.5	Sistema de calorimetria do ATLAS [extraído de [19]].	10
2.6	Estrutura de partição do TileCal [extraído de [21]].	11
2.7	Visão tridimensional de um módulo do TileCal [extraído de [9]]. . . .	12
2.8	Segmentação do TileCal, para os módulos do barril e barril estendido do TileCal [extraído de [9]].	13
2.9	Diagrama de blocos da cadeia eletrônica TileCal [extraído de [10]]. . .	13
2.10	Esquema de um bloco de PMT [extraído de [9]].	14
2.11	Pulso característico do TileCal.	15
2.12	TileCal <i>Read-Out Motherboard</i> [extraído de [10]].	16
2.13	Fenômeno de empilhamento de sinais no TileCal [extraído de [26]]. . .	18
2.14	Diagrama de Blocos da sROD [extraído de [27]].	19
3.1	Identificação de sinais empilhados e a reconstrução da energia. . . .	31
3.2	Diagrama Geral do método COF.	32
3.3	Histograma de Reconstrução da Energia [extraído de [35]].	32
3.4	Correlação entre o COF e o OF [extraído de [35]].	33
4.1	Identificação dos valores de μ em que há divergência.	50
4.2	Convergência do algoritmo GD	50
4.3	Desvio do Erro para ($0 < \text{it} < 20$, $0.01 < \mu < 0.5$ e ocupância de 10%).	51
4.4	Desvio do Erro para ($20 < \text{it} < 100$, $0.26 < \mu < 0.5$ e ocupância de 10%).	52
4.5	Desvio do Erro para ($20 < \text{it} < 100$, $0.26 < \mu < 0.5$ e ocupância de 20%).	53
4.6	Desvio do Erro para ($13 < \text{it} < 100$, $0.4 < \mu < 0.5$ e ocupância de 10%).	54
4.7	Desvio do Erro para ($20 < \text{it} < 100$, $0.4 < \mu < 0.5$ e ocupância de 20%).	54

4.8	Comparação entre os métodos GD , GDD e COF e a identificação do valor mínimo de iterações para o método GDD , considerando o desvio de 1% e ocupância de 10 e 20%.	55
4.9	Comparação entre os métodos GD , GDD , GC e COF e a identificação do valor mínimo de iterações para o método GC , considerando o desvio de 1% e ocupância de 10 e 20%.	56
4.10	Curva de Convergência dos Métodos GD, GDD e GC	57
4.11	Curva de Convergência dos Métodos para BC-2 ($a_{-2} = 29$)	58
4.12	Curva de Convergência dos Métodos para BC central ($a_0 = 25$)	58
4.13	Curva de Convergência dos Métodos para BC+3 ($a_{+3} = 27$)	59
5.1	Arquitetura do Processador em FPGA	70
5.2	Formato para Ponto Fixo.	72
5.3	Formato para Ponto Fixo.	72
5.4	Soma e subtração em ponto fixo.	73
5.5	Comparação entre o GD FPGA e o GD MATLAB.	75
A.1	O sistema de coordenadas do ATLAS.	84

Listas de Tabelas

3.1	Tabela Comparativa entre BLUE Escalar e a um Vetor de Parâmetros.	28
4.1	Comparativo entre os Métodos (COF, GD, GDD e GC) a partir da amplitude real de sinais sobrepostos.	59
5.1	Análise de complexidade para células do Barril (ocupância de 10%), considerando o desenvolvimento de uma arquitetura sequêncial.	66
5.2	Análise de complexidade para células do Barril Estendido (ocupância de 20%), considerando o desenvolvimento de uma arquitetura sequêncial.	67
5.3	Comparativo entre os Métodos (COF, GD e GC) a partir da amplitude real de sinais sobrepostos.	68
5.4	Tabela com valores de convergência do algoritmo GD	75

Lista de Abreviaturas e Siglas

ATLAS *A Toroidal LHC ApparatuS*

LHC *Large Hadron Collider*

OF *Optimal Filter*

COF *Constrained Optimal Filter*

FPGA *Field-Programmable Gate Array*

TileCal *Tile Calorimeter*

CERN *Conseil Européen pour la Recherche Nucléaire*

ALICE *A Large Ion Collider Experiment*

LHCb *Large Hadron Collider beauty*

CMS *Compact Muon Solenoid*

LVL1 *Level 1*

LVL2 *Level 2*

EF *Event Filter*

PMT *Photo Multiplier Tube*

CIS *Charge Injection System*

LAS *Laser System*

CS *Cesium System*

TRT *Transition Radiation Tracker*

SCT *Semi-Conductor Tracker*

ROC *Receiver Operating Curve*

ROD *Read Out Driver*

FIR *Finite Impulse Response*

ADC *Analog-to-Digital Converter*

DSP *Digital Signal Processor*)

TDAQ *Trigger and Data Acquisition*

LAR *Liquid Argonic*

EM Eletro-magnético

CTP *Central Trigger Processor*

HLT *High Level Trigger*

TT *Trigger Tower*

PU *Processing Unit*

BLUE *Best Linear Unbiased Estimator*

MSE *Mean Squared Error*

GD *Gradiente Descendente*

GDD *Gradiente Descendente Adaptativo*

GC *Gradiente Conjugado*

BC *Bunch Crossing*

Capítulo 1

Introdução

No campo da engenharia, mais especificamente da instrumentação eletrônica, constantemente surgem novos requisitos e restrições que impactam diretamente nos sistemas eletrônicos, sensores e principalmente nas técnicas de processamento de sinais. Junto com o aumento da capacidade de processamento e do armazenamento nos hardwares atuais, aumenta-se o número de sinais e serem processados e consequentemente a quantidade de dados a serem processados, analisados e armazenados.

Dentro do contexto da instrumentação eletrônica avançada, a física experimental de altas energias é um ambiente extremamente desafiador, envolvendo o processamento e aquisição de milhares de sinais a taxas extremamente altas. Neste contexto, a energia das partículas subatômicas geradas a partir de um fenômeno físico é absorvida e amostrada por materiais especiais e posteriormente convertida em sinais elétricos. Estes sinais são condicionados por circuitos elétricos e enviados para conversores analógico-digitais. Uma vez digitalizado, os mesmos podem ser lidos e processados. Muitas das vezes, devido à grande quantidades de dados gerados, vários níveis de filtragem podem ser requeridos, reduzindo a taxa de eventos e consequentemente a quantidade de dados a serem armazenados. O armazenamento total da informação é muitas vezes inviável devido à restrição na quantidade de dados que podem ser armazenados e, adicionalmente, a maior parte dos dados pode não conter informações relevantes para análises.

1.1 Motivação

Experimentos de física de altas energias apoiam-se em complexos sistemas que visam à medição de parâmetros e propriedades de partículas subatômicas. Nesse contexto, destacam-se o Grande Colisionador de Prótons (**LHC**, do inglês Large Hadron Collider) e um de seus principais detectores, o **ATLAS**, localizados na Organização Europeia para Pesquisas Nucleares (**CERN**, do francês Centre Européen pour La Recherche Nucléaire). Estes experimentos apoiam-se fortemente em

seus sistemas de calorimetria. Um calorímetro é um bloco de matéria que deve ser suficientemente espesso para que toda a energia das partículas seja depositada em seu volume. Uma parte da energia inicial é dissipada na forma de calor, mas uma fração é amostrada de forma mais prática para o processamento, sendo proporcional a energia inicial.

O calorímetro hadrônico do **ATLAS**, o Calorímetro de Telhas (TileCal, do inglês Tile Calorimeter), é um dos principais detectores do **LHC**, que está passando por um processo de atualização devido ao aumento do número de interações por colisão no **LHC** para os próximos 10 anos. Com este aumento, mais partículas atingirão os calorímetros do **ATLAS**, o que pode gerar sinais empilhados dificultando o processo de estimativa da energia. Desta forma, o algoritmo atualmente utilizado no TileCal, para estimativa da energia depositada em cada um de seus canais, sofrerá uma redução de desempenho. Portanto, um novo algoritmo para estimativa deve ser desenvolvido.

Devido à eletrônica comumente utilizada nos calorímetros, a estimativa de energia pode ser feita através da estimativa da amplitude do pulso resultante da deposição de energia em um determinado canal, ou seja, a amplitude do pulso é proporcional a energia depositada no canal. O método de estimativa online da amplitude do pulso utilizado atualmente no TileCal, chamado de **OF** (do inglês, *Optimal Filter*), baseia-se em um processo de minimização de variância para a estimativa da amplitude de um pulso de referência. O método é ótimo para ambientes onde o ruído de fundo pode ser modelado por uma distribuição gaussiana e o sinal não varia o seu formato e a sua fase. Porém, em cenários com empilhamento de eventos o **OF** deixa de ser ótimo.

Proposto pela colaboração brasileira, um algoritmo mais eficiente foi implementado *offline*. Tal algoritmo, denominado **COF** (do inglês, *Constrained Optimal Filter*), vem obtendo excelentes resultados no cenário de alta luminosidade, estimando, além da amplitude do pulso de interesse, a amplitude de cada componente sobreposta. Em um contexto global, os novos requisitos e restrições previstos no plano de atualização exigem o desenvolvimento de sistemas eletrônicos mais modernos capaz de atender os problemas decorrentes da alta luminosidade. Neste sentido, a eletrônica atual será substituída, incluindo novos processadores, possibilitando a implementação de métodos de reconstrução *online* de energia mais complexos computacionalmente mas que possam ter melhor desempenho em cenários de empilhamento de eventos. No entanto, o **COF** apresenta elevada complexidade o que dificulta sua implementação dentro dos requisitos de tempo de processamento exigidos nos níveis de filtragem de eventos.

Tendo em vista a importância da estimativa da energia para calorimetria e da relevância desta informação para o **ATLAS**, este trabalho tem como objetivo avaliar

métodos de implementação do **COF**, viabilizando sua utilização para processamento *online*.

1.2 O que foi Feito

Neste trabalho, foram realizados estudos que visam implementar o **COF** para processamento *online*. O **COF** requer a inversão de matrizes para o cálculo da pseudo-inversa de uma matriz de convolução. Isso permite deconvoluir os sinais sobrepostos, possibilitando recuperar a amplitude dos mesmos. A necessidade de realizar um procedimento de inversão de matrizes ou a possibilidade de utilização de bancos de filtros para cada combinação de sinais empilhados, apresenta elevado custo para implementação *online*. Nesse sentido, o objeto de contribuição deste trabalho é realizar esta inversão através da solução de um sistema de equações lineares, a ser resolvido de forma iterativa, sequencial, viabilizando a implementação em FPGA para processamento *online* e, permitindo assim, a reconstrução de energia em ambiente de alta luminosidade dentro dos requisitos de tempo de processamento dado pelo sistema.

A obtenção da pseudo-inversa pode ser um processo de minimização do erro médio quadrático entre a solução do problema e o modelo linear adotado para estimação de sinais. São utilizados então, métodos iterativos para se chegar ao custo mínimo, empregando variações do método de Gradiente Descendente. O número de iterações e taxa de convergência são analisadas permitindo a comparação dos métodos quanto a eficiência na estimação de múltiplas amplitudes. Além disso, os algoritmos são analisados quanto a complexidade computacional e quanto ao número de ciclos de *clock* necessários para execução do algoritmo em FPGA. Desta forma, no presente trabalho foi proposto um modelo de processador em FPGA para execução dos algoritmos.

Para o teste da arquitetura em FPGA, o algoritmo Gradiente Descendente foi implementado no processador. Baseado em um arquitetura sequencial e utilizando lógica aritmética em ponto fixo, o processador é capaz de executar o método iterativo e consequentemente estimar a amplitude de sinais sobrepostos com desempenho similar ao método **COF**, viabilizando sua implementação em FPGA para processamento *online*. As análises dos algoritmos considerando a quantidade de operações realizadas, por iteração, e a quantidade de ciclos utilizados em cada uma das etapas de cálculo, permitiram definir a frequencia de processamento para arquitetura proposta, em cada caso, viabilizando a implementação do **COF** dentro dos requisitos de tempo de processamento dado pelo sistema de trigger do **ATLAS**.

1.3 Visita Guiada

Visando a contextualização do ambiente de pesquisa deste trabalho, o Capítulo 2 descreve sucintamente o **CERN**, o **LHC** e o TileCal. Ainda neste capítulo, é apresentada uma descrição detalhada dos efeitos da alta luminosidade e os problemas decorrentes do empilhamento de sinais. Neste contexto, o programa de atualização da eletrônica do TileCal é descrito com intuito de demonstrar ao leitor que o algoritmo eficiente em cenário de alta luminosidade será implementado na nova eletrônica do TileCal, mais especificamente em FPGAs modernas que substituirão os DSPs.

O Capítulo 3 é dedicado à revisão bibliográfica das técnicas para estimação da amplitude de sinais. Inicialmente apresenta-se uma revisão da teoria da estimação, no contexto do problema da calorimetria, tendo um papel importante neste trabalho, pois serão utilizadas para descrever os métodos de reconstrução de energia no TileCal. Após demonstrado que o pulso característico do TileCal pode ser modelado através de um sinal determinístico, e que o parâmetro de amplitude é proporcional a energia, o leitor será capaz de associar a teoria da estimação com a formulação do método de reconstrução online de energia atualmente implementando em DSPs, o **OF**. Posteriormente, o leitor poderá comparar o **OF** com um método que apresenta melhor desempenho em cenário de alta luminosidade (**COF**). Desta forma, o **COF** é utilizado como base para a formulação matemática de métodos iterativos, apresentados no Capítulo 4.

O Capítulo 4 é dedicado a descrever uma das principais contribuições deste trabalho. Neste capítulo, são propostas técnicas para adaptação do **COF** através de métodos iterativos. Os métodos permitem estimar a amplitude do pulso de interesse e de cada componente sobreposta, resultando em simples operações matemáticas que viabilizam a implementação em dispositivos FPGA para processamento *online*. Ainda neste capítulo, os métodos iterativos são simulados e comparados.

No Capítulo 5 são apresentadas as análises de complexidade dos métodos iterativos e a proposta de um modelo de processador em FPGA para execução dos algoritmos. Finalmente, no Capítulo 6, serão apresentadas as conclusões finais do trabalho realizado e as expectativas sobre os planos futuros.

Capítulo 2

CERN, LHC e o ATLAS

Este capítulo descreve o ambiente de desenvolvimento do trabalho. Para tal, inicialmente é apresentado o maior centro de pesquisas de física de partículas do mundo, o **CERN** (do francês, *Conseil Européenne pour la Recherche Nucleaire*) e o grande colisionador de protons, o **LHC** (do inglês, *Large Hadron Collider*). Em seguida, a Seção 2.2 apresenta o detector **ATLAS**, com foco no Calorímetro Hadrônico de Telhas, chamado de **TileCal**. Por fim, a Seção 2.3 detalha os efeitos do aumento da luminosidade do **LHC** e o plano de atualização do TileCal.

2.1 O CERN e o LHC

Fundada em 1954 e situada na fronteira Franco-Suíça, próximo a cidade de Genebra [1], a Organização Européia para Pesquisa Nuclear, o **CERN**, foi idealizado para impulsionar a ciência na Europa. Atualmente o **CERN** é o maior centro de pesquisa de física de partículas do mundo e é formado por países membros, europeus, e países associados, dos demais continentes, chamados não membros. Neste contexto de colaboração internacional, o objetivo primário do laboratório é estudar e explorar duas questões da física: i) a composição da matéria; ii) as forças que regem as interações eletromagnéticas, fortes e fracas.

O **CERN** desenvolveu e construiu o maior colisionador de partículas do mundo, o **LHC** [2]. O Grande Colisionador de Hâdrons, como é chamado, encontra-se atualmente em funcionamento, sendo o maior e mais energético colisionador de partículas já construído. Do ponto de vista tecnológico e científico, o **LHC** representou um enorme avanço em diversas áreas do conhecimento, estando entre elas as Engenharias. No campo científico, o estudo das colisões de partículas no **LHC** possibilitou a comprovação da existência do Bóson de Higgs [3] [4]. O Bóson de Higgs é uma partícula elementar predita pelo Modelo Padrão da física de partículas e possibilita explicar a origem da massa das outras partículas elementares. Apesar de ter sido descoberta apenas em julho de 2012, sua teoria foi anunciada pela primeira vez em

1964, pelo físico inglês Peter Higgs, dentre outros pesquisadores.



Figura 2.1: Vista aérea do Complexo do CERN. [Extraído de [5]].

No **LHC**, os feixes de partículas, formado por milhares de pacotes de prótons, são acelerados no interior de um anel em sentidos contrários e colidem em pontos específicos em intervalos de 25 ns. Desta colisão, resultam em partículas elementares que são emitidas em várias direções. Devido à grande quantidade de diferentes fenômenos físicos a serem observados, quatro grandes experimentos estão instalados ao longo do túnel no **LHC**. São eles:

- O **ALICE** (do inglês, *A Large Ion Collider Experiment*) é um experimento ligado à física de colisões nucleares. Seu objetivo é estudar uma fase da matéria, onde é esperado a observação do plasma de quarks e glúons [6].
- O **CMS** (do inglês, *Compact Muon Solenoid*), é um experimento de propósito geral para estudo das colisões próton-próton do LHC, buscando principalmente comprovar o modelo padrão, como por exemplo a existência do Bóson do Higgs [7].
- O **LHCb** (do inglês, *Large Hadron Collider Beauty Experiment*) é um experimento dedicado ao estudo da quebra de simetria entre matéria e anti-matéria e outros fenômenos raros que podem ser observados do decaimento do méson-B [8].
- O **ATLAS** (do inglês, *A Toroidal LHC AparatuS*) assim como o **CMS**, é um experimento que visa estudar uma ampla gama de fenômenos físicos, principalmente a existência do Bóson do Higgs [9].

Estes experimentos realizam a absorção, detecção e identificação das partículas resultantes das colisões. A alta taxa de eventos, a energia da colisão e o grande número de prótons em cada feixe geram uma enorme quantidade de dados, sendo em sua maioria ruído de fundo do experimento. Desta forma, complexos sistemas de filtragem de eventos *online* devem ser utilizados.

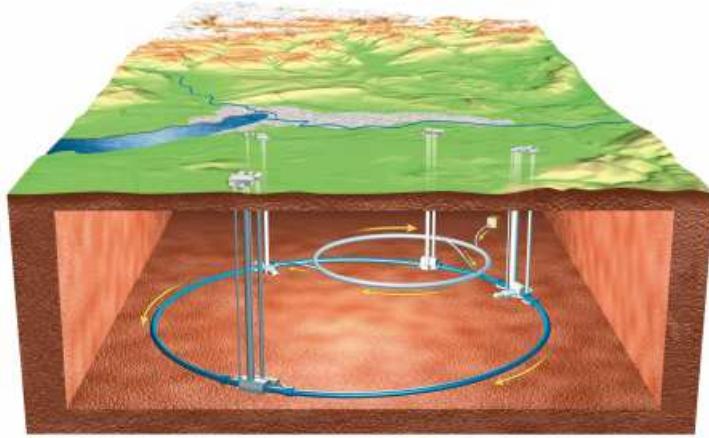


Figura 2.2: O anel do LHC e seus quatro detectores principais [extraído de [10]].

Na próxima seção será apresentado, com maiores detalhes, o detector **ATLAS**.

2.2 O ATLAS

O detector **ATLAS**, apresentado na Figura 2.3, é um dos principais experimentos do **LHC**. Possui forma cilíndrica com diâmetro de 22 metros, comprimento de 42 metros e peso igual a 7 mil toneladas. Desenvolvido por uma colaboração entre 37 países, o **ATLAS** é atualmente o maior detector para colisionadores e de partículas.

Em sua representação em coordenadas cilíndricas, como é apresentado através da Figura 2.4, o eixo-Z representa a direção do feixe no túnel do acelerador **LHC**. O eixo-X aponta para o centro do anel do **LHC** e o eixo-Y é perpendicular aos eixos X e Z, sendo O o ponto de colisão do detector. O termo ρ é o raio e φ é o ângulo ao redor do eixo do feixe de partículas. O termo θ define o ângulo de incidência de uma partícula em relação a direção do feixe do **LHC**. Para caracterizar a rapidez de uma partícula, é necessário medir sua energia e sua força longitudinal. Em várias experiências, só é possível medir o ângulo da partícula detectada em relação ao eixo do feixe. Nesse caso, é conveniente utilizar essas informações usando a variável pseudo-rapidez η para localizar a partícula detectada [12], que é definida como:

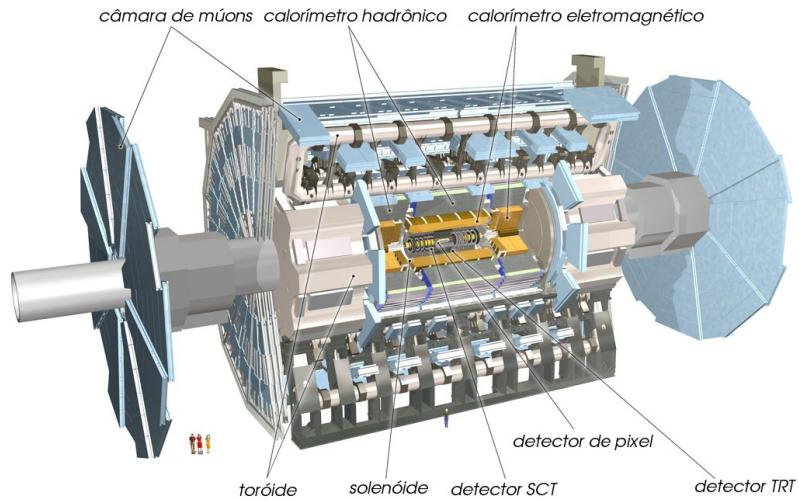


Figura 2.3: ATLAS e seus principais detectores e dimensões [extraído de [11]].

$$\eta = -\ln \left[\tan \left(\frac{\theta}{2} \right) \right] \text{ (ver Apêndice A)} \quad (2.1)$$

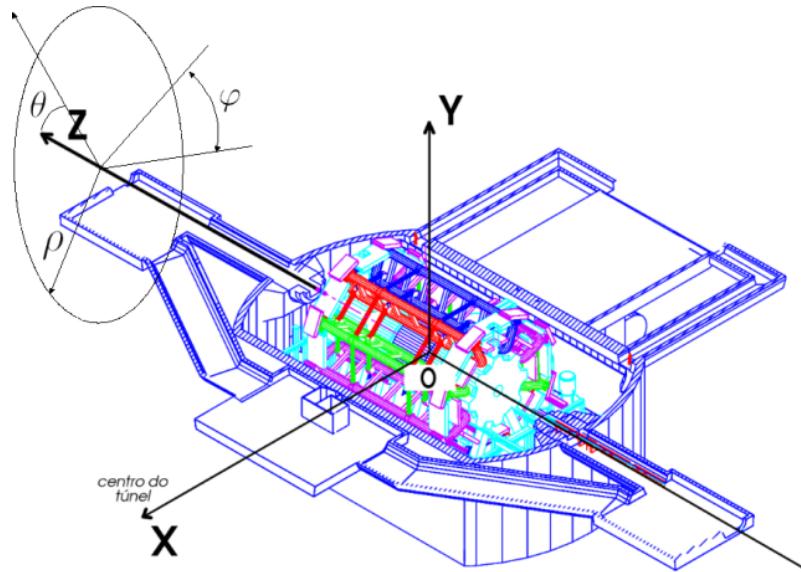


Figura 2.4: Coordenadas do ATLAS [extraído de [13]].

O detector **ATLAS** é composto por diversos sub-detectores, onde cada um se dedica a medir propriedades específicas das partículas incidentes. Os sub-detectores que compõem o **ATLAS** envolvem o feixe de partículas em um ponto de colisão [14] por camadas cilíndricas. Além dos Magnetos (Solenoid e Toroid Magnets), desenvolvidos para contribuir na medição do momento das partículas carregadas, outros três sub-detectores compõem o **ATLAS**. São eles:

- Detector de Traço (do inglês, *Inner Detector*): Sub-detector mais interno com-

posto por outros três sub-detectores (Pixel Detector, Semiconductor Tracker (*SCT*) e Transition Radiation Tracker (*TRT*)) responsáveis por determinar os traços das partículas e medir o momento das mesmas [15].

- Espectrômetro de Múons (do inglês, *Muon Spectrometer*): Sub-detector desenvolvido para identificar e medir o momento dos múons [16]. Este detector é localizado na parte mais externa do **ATLAS**, pois os múons são as únicas partículas detectáveis que alcançam esta seção do detector.
- Calorímetro Eletromagnético e Hadrônico: O Calorímetro Eletromagnético e o Calorímetro Hadrônico (*TileCal*) do **ATLAS** são subdetectores desenvolvidos para medir a energia das partículas que interagem de forma eletromagnética e hadrônica respectivamente [17, 18].

A seguir, o sistema de calorimetria é apresentado. Por se tratar do ambiente de pesquisa deste trabalho, o Calorímetro Hadrônico do **ATLAS** será detalhado. As próximas seções visam a contextualização do ambiente de pesquisa, pelo qual é possível absorver, detectar e especialmente estimar a energia das partículas resultantes das colisões através da estimação de amplitude de um pulso com forma bem definida (eletronicamente condicionado).

2.2.1 Calorimetria

Os calorímetros são responsáveis por absorver e medir a energia das partículas nele incidentes. As partículas incidentes geram um chuveiro de partículas ao entrarem em contato com material denso dos calorímetros [15]. Neste processo, a energia da partícula é depositada, coletada e medida com precisão. A etapa de absorção se dá através da composição do material dos calorímetros, que se utilizam de materiais pesados para formar barreiras, fazendo com que as partículas sejam completamente absorvida. Ao penetrar nos calorímetros, inicia-se o processo de chuveiros onde as partículas sofrem decaimentos que resultam na produção de outras partículas de menor energia. Este processo ocorre até que a energia da partícula incidente seja totalmente absorvida no interior da estrutura dos calorímetros.

Tipicamente, os calorímetro são transversalmente segmentados para obter informação da direção das partículas, bem como da energia depositada. Uma segmentação longitudinal obtém a informação da identidade da partícula, baseada na forma do chuveiro que ela produz. Dentre as razões pelas quais os calorímetros emergiram como detectores-chave em praticamente todos os experimentos em física de partículas, podemos citar:

- Calorímetros podem ser sensíveis tanto a partículas neutras quanto a carregadas.

- Devido a diferenças na forma de deposição de energia, a identificação de partículas pode ser feita com alta eficiência.
- Para conter o desenvolvimento de cascatas dos objetos a serem medidos, a profundidade dos calorímetro aumenta logaritmicamente com a energia, o que permite o projeto de detetores mais compactos.
- Podem ser segmentados, o que permite tanto medida da energia quanto de trajetória das partículas.
- Resposta rápida (menor que 50 ns) pode ser atingida, o que é importante num ambiente com alta taxa de eventos.
- A informação de energia pode ser usada para filtrar eventos interessantes com alta seletividade.

O sistema de calorimetria do **ATLAS**, mostrado na Figura 2.5, é composto pelos calorímetros Eletromagnético e Hadrônico. O primeiro absorve energia de partículas que interagem com maior probabilidade de forma eletromagnética, elétrons e fótons, permitindo medidas de alta precisão, tanto em energia quanto em posição. O calorímetro hadrônico absorve energia de partículas que interagem através da interação forte, principalmente hâdrons. Este último é descrito com maiores detalhes, a seguir.

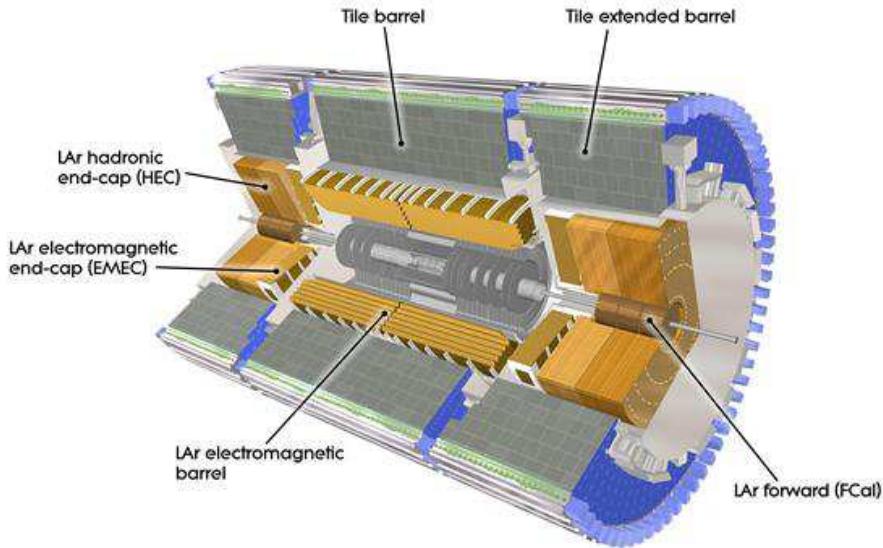


Figura 2.5: Sistema de calorimetria do ATLAS [extraído de [19]].

2.2.2 Calorímetro Hadrônico de Telhas (TileCal)

O calorímetro hadrônico tem como finalidade medir a energia e a direção das partículas hadrônicas, que não são absorvidas pelo calorímetro eletromagnético. O

sistema de calorimetria hadrônica é formado pelo Calorímetro de Telhas (TileCal) (do inglês *Tile Calorimeter*) [20], principal calorímetro hadrônico do **ATLAS**, que cobre uma região de $0 < |\eta| < 1.7$. É formado, também, pela tampa hadrônica que estende-se em uma região de $1.5 < |\eta| < 3.2$ [9]. Em forma cilíndrica, o TileCal possui raio inferior e externo de 2,3 m e 4,3 m, respectivamente. Assim como o calorímetro elétromagnético, o mesmo divide-se em 4 partições longitudinais, tendo a parte central o barril (BL, do inglês *Long Barrel*) com 5,6 m e as duas extremidades, denominada de barril estendido (EB, do inglês *Extended Barrel*), com 2,9 m cada uma. O barril central também divide-se em duas partes resultando assim nas partições EBC, LBC, LBA e EBA, em que as letras A e C identificam as partições, se positivo ou negativo em relação a η . A Figura 2.6 ilustra a estrutura de partição do TileCal.

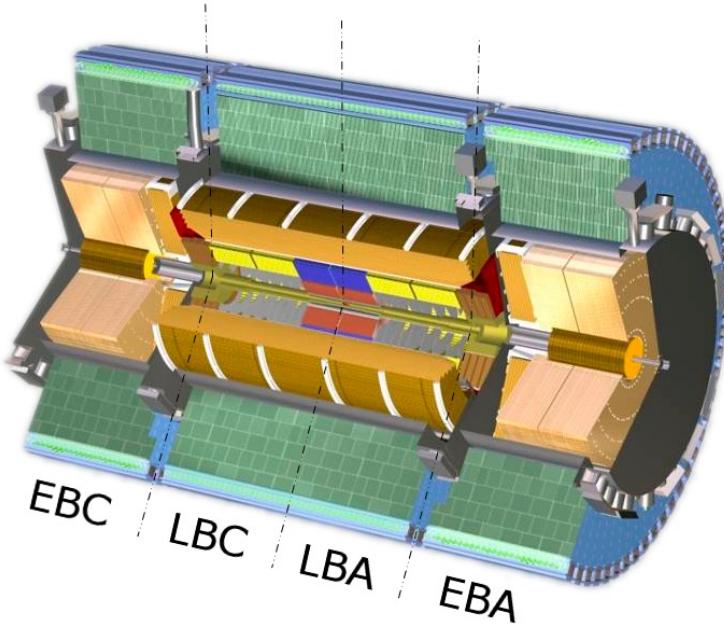


Figura 2.6: Estrutura de partição do TileCal [extraído de [21]].

O Calorímetro de Telhas utiliza o aço como material absorvedor e telhas cintilantes como amostradores de energia. Com a passagem das partículas carregadas, as telhas se excitam, produzindo fótons, que são coletados por células fotomultiplicadoras (PMT, do inglês *PhotoMultiplier Tube*), cuja função é converter a luz em um sinal elétrico. A luz gerada nos cintiladores é coletada por fibras ópticas em suas duas extremidades, permitindo assim uma redundância na coleta dos dados. Estas fibras são estrategicamente agrupadas e acopladas às fotomultiplicadoras, de modo a formarem células de leitura, para se obter uma segmentação radial tridimensional. O sinal gerado na PMT também passa por um circuito conformador de pulsos, visando tornar o sinal um pulso padrão (pulso de referência do TileCal), cuja amplitude é diretamente proporcional à energia depositada pela partícula. Cada PMT

recebe sinais luminosos de certo número de telhas cintilantes formando as células do calorímetro, sendo que cada célula é lida por duas PMTs (visando redundância e maior uniformidade na leitura). Cada PMT corresponde a um canal de leitura e a combinação de dois canais referentes às fibras de um mesmo cintilador corresponde à uma célula de leitura. A Figura 2.7 apresenta a visão tridimensional de um módulo do TileCal.

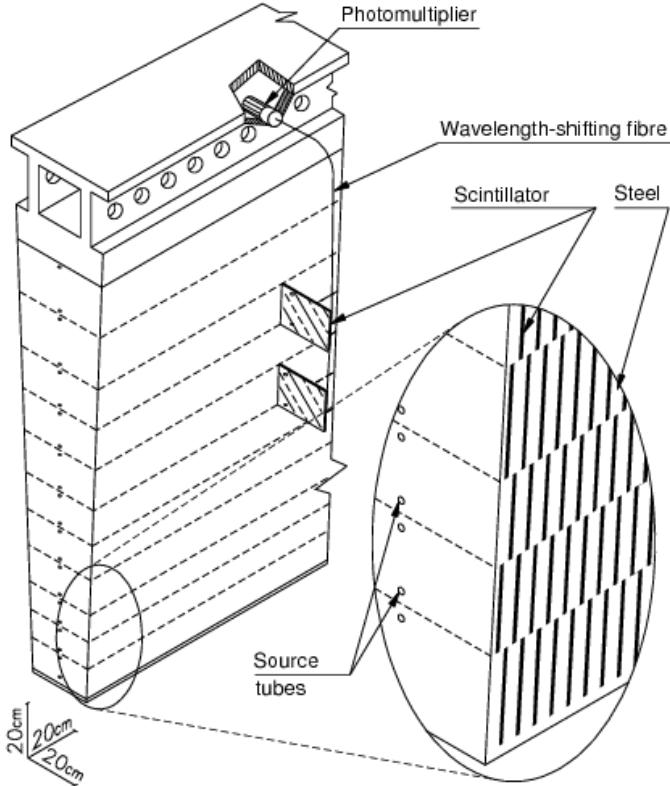


Figura 2.7: Visão tridimensional de um módulo do TileCal [extraído de [9]].

Ao longo da profundidade do detector, o TileCal, é segmentado em três camadas, chamadas de A, BC e D. Também é segmentado em células, ao longo de cada camada. As duas primeiras camadas, A e BC, possuem a mesma granularidade, tanto para o barril como para o barril estendido, enquanto que a terceira camada, a camada D, possui células maiores. Através da Figura 2.8, pode-se observar que a granularidade no sentido de η , que é mantida constante em $\Delta\eta = 0,1$, tanto para o barril quanto para o barril estendido. A camada D possui células maiores com $\Delta\eta = 0,2$.

O TileCal possui aproximadamente 10.000 canais de leitura. Sendo assim, uma enorme quantidade de dados devem ser processados. Entretanto, devido à alta taxa de eventos e ao fluxo de dados resultante, o armazenamento total da informação de cada subdetector torna-se inviável, adicionalmente a maior parte destes dados não contém informações relevantes para análises, no contexto da física de interesse [22]. Para conseguir selecionar apenas os eventos relevantes, o sistema de *trigger* do **ATLAS** foi desenvolvido.

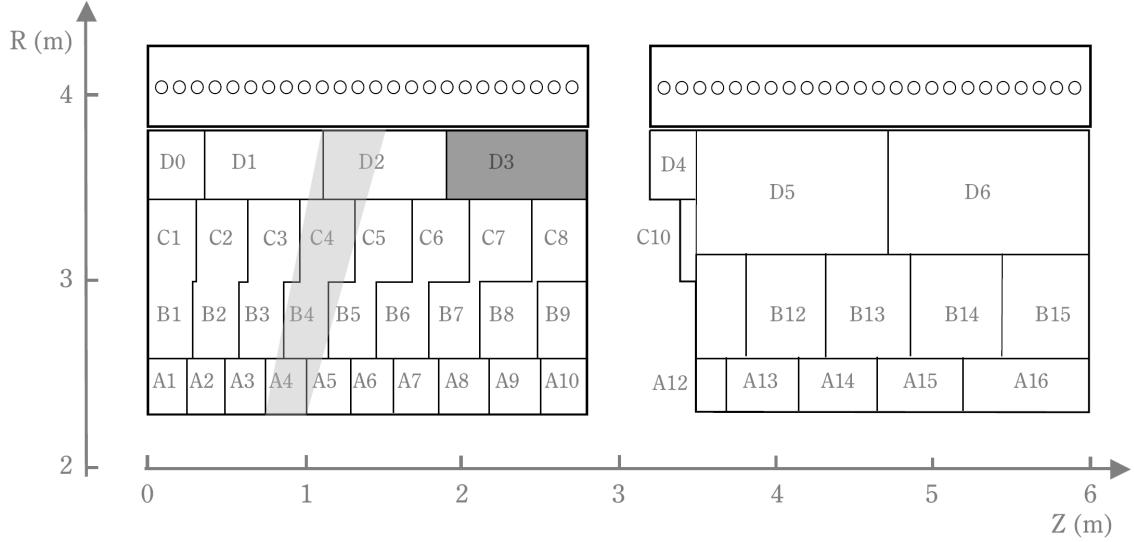


Figura 2.8: Segmentação do TileCal, para os módulos do barril e barril estendido do TileCal [extraído de [9]].

2.2.2.1 Eletrônica de Leitura do TileCal

A Figura 2.9 apresenta uma visão geral da eletrônica de leitura para o TileCal. Ele é dividido em eletrônica de *front-end*, que são instalados no detector, e de *back-end* que são instaladas nas salas de controle do **ATLAS**, em ambiente de baixa radiação.

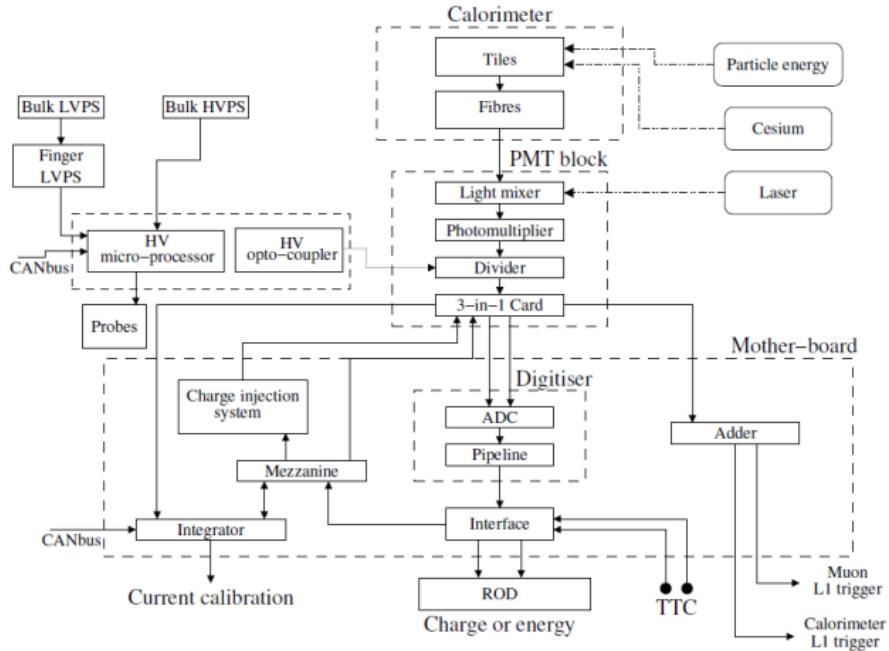


Figura 2.9: Diagrama de blocos da cadeia eletrônica TileCal [extraído de [10]].

2.2.2.2 Eletrônica de Front-End

A eletrônica de *front-end* está alojada dentro do detector em uma estrutura sólida feita de aço, conhecida como gaveta (*Drawer*). Estas gavetas são móveis, permitindo o acesso ao sistema. Tal estrutura está localizada na região externa de cada módulo, na parte de trás do calorímetro. Duas gavetas formam uma super-gaveta, que é composta por vários subsistemas: Blocos PMTs, *motherboards*, placas digitalizadoras e placas de interface óptica [10] [9].

Os fotomultiplicadores, ou PMTs, são elementos-chave de leitura. Estão localizados dentro de uma estrutura mecânica que forma um escudo cilíndrico de aço para blindagem magnética. Um misturador de luz (do inglês, *light mixer*) está localizado entre a fibra de saída do calorímetro e a PMT. Sua função é misturar a luz proveniente de todas as fibras de leitura em um feixe, para garantir uma iluminação uniforme, descorrelacionando a posição da fibra e a área da PMT a qual recebe a luz. A Figura 2.10 mostra o esquema de um bloco de PMT.

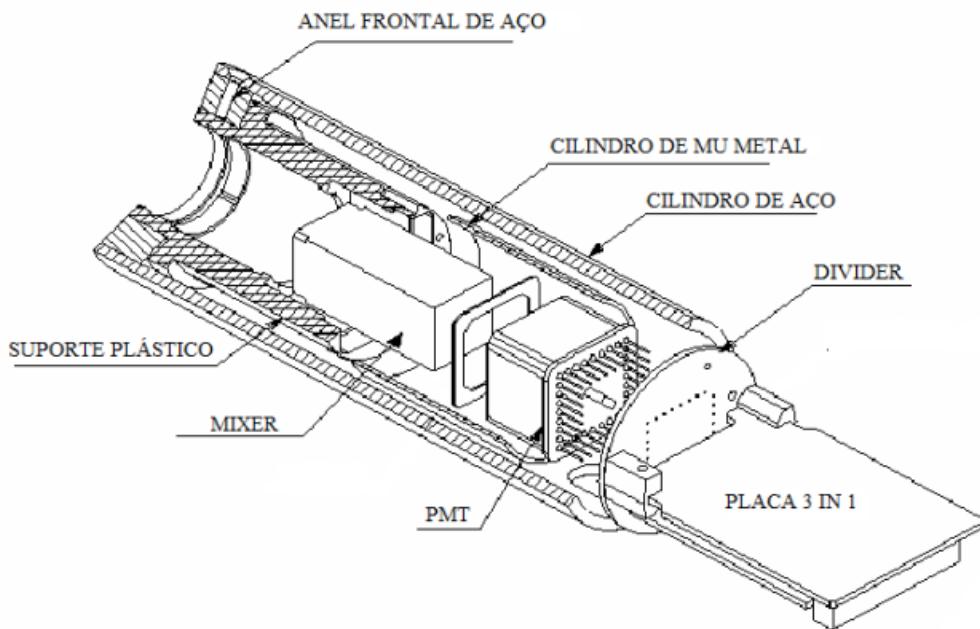


Figura 2.10: Esquema de um bloco de PMT [extraído de [9]].

Eles são capazes de multiplicar o sinal produzido pela luz incidente em milhões de vezes, permitindo que pequenos fluxos de fótons sejam detectados. No TileCal, o fotomultiplicador utilizado é o Hamamatsu R7877 [23]. O número de canais de leitura (PMTs) são de aproximadamente 10.000.

O divisor de alta tensão (do inglês, *high voltage divider*) é um circuito impresso com dispositivos de montagem em superfície (SMD, do inglês, *Surface Mount Device*) que estão ligados às placas 3-em-1. O divisor (*Divider*) de alta tensão tem como função principal a distribuição de energia entre as PMTs. As placas 3-em-1,

por sua vez, tem como função principal o condicionamento dos pulsos elétricos por um circuito de *shaper* [24] visando tornar estes sinais um pulso padrão (pulso de referência do TileCal), de modo que a energia total da partícula que é medida na PMT seja proporcional a amplitude do pulso. Antes dos pulsos serem aplicados às placas digitalizadoras, os mesmos são amplificados por circuitos operacionais produzindo dois sinais com ganho relativo de 64, correspondendo a uma faixa de ganho alto (*High Gain*) e baixo (*Low Gain*). Além disso, sinais de baixo ganho são enviados para as placas somadoras (*Adder*) as quais somam os sinais para serem enviados ao nível 1 de filtragem do **ATLAS** [25], o qual recebe os mesmos [10].

A placa-mãe é o elemento básico que une todos os componentes eletrônicos em uma gaveta. Há duas placas-mãe em uma super-gaveta. Ela fornece energia, sincronismo e controle para as placas 3-em-1, e abriga até 4 placas digitalizadoras e uma placa de interface. Cada digitalizador (do inglês digitizer) lê a informação de até seis blocos de PMT. O digitalizador recebe os sinais de ganho alto e baixo provenientes de seis placas 3-em-1, para digitalização dos eventos a cada 25 ns, utilizando ADCs comerciais de 10 bits. Cada pulso é amostrado por um ADC na taxa de 40 MHz resultando em amostras discretas separadas 25 ns uma da outra. A Figura 2.11 ilustra o pulso característico de um canal do TileCal com seus parâmetros de amplitude, largura, pedestal e fase.

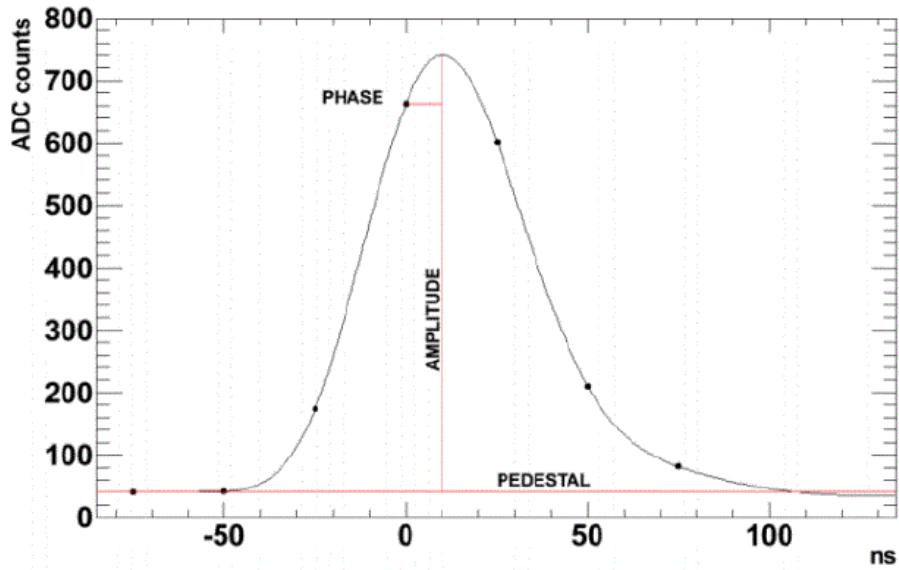


Figura 2.11: Pulso característico do TileCal.

O pedestal é definido como sendo a linha de base do sinal e a amplitude é a altura do sinal medido a partir do pedestal. A fase do sinal pode ser medida como a diferença de tempo entre a amostra central (quarta amostra) e o pico do pulso. Uma janela de 150 ns, ou seja, as sete amostras enviadas através de fibras óticas, são enviadas para os Read Out Drivers (RODs) na eletrônica de back-end, onde a

energia é estimada para os eventos aceitos pelo nível 1 de filtragem do ATLAS. O pulso da saída de *High Gain* é utilizado a menos que alguma de suas amostras tenha saturado o ADC. No caso de saturação, a leitura do pulso da saída de *Low Gain* é utilizada.

Até aqui, pode ser visto que os sinais utilizados para determinar a energia são transmitidos por circuito de condicionamento do pulso. Este pré-processamento fornece um pulso estável e de forma fixa o qual tem sua amplitude proporcional à energia do sinal (Figura 2.11). Com isso, através da estimativa da amplitude do pulso, o sinal e sua energia podem ser estimadas. Idealmente, os parâmetros do pulso são constantes exceto a amplitude, a qual depende da energia depositada. Uma vez que este trabalho está relacionado ao processo de estimação da amplitude do pulso do TileCal, realizado nas RODs, então a eletrônica de *back-end* é detalhada a seguir.

2.2.2.3 Eletrônica de Back-End

A placa *Read-Out Driver*, Figura 2.12, é o elemento central de processamento e de ligação com a eletrônica de *front-end*. A ROD recebe dados das super gavetas, através de oito ligações ópticas, cada um correspondendo a uma super gaveta, sendo necessárias 32 RODs para cobrir todo o calorímetro.

A ROD é equipada por uma série de componentes em que se destaca a unidade de processamento (PU), cuja função primária é a de computar a energia e fase de cada canal de leitura a partir das amostras digitalizadas.

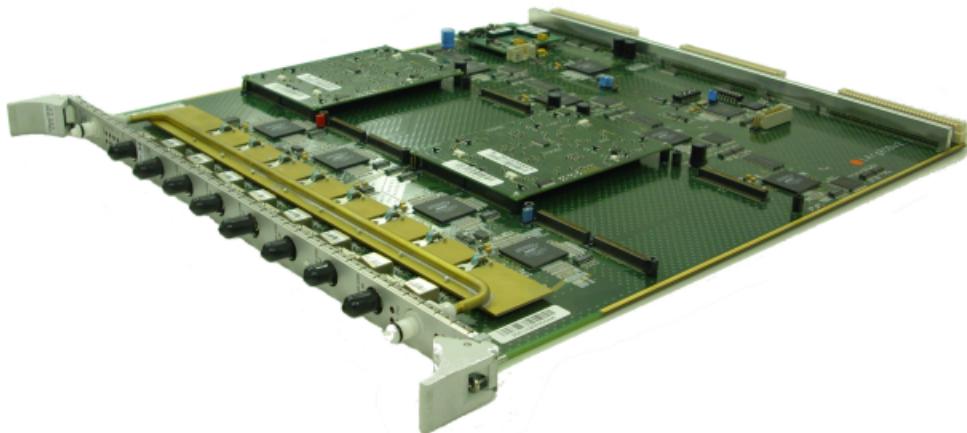


Figura 2.12: TileCal *Read-Out Motherboard* [extraído de [10]].

O TileCal possui duas unidades de processamento (PUs) por ROD. Cada PU é equipada com dois dispositivos DSPs (do inglês, *Digital Signal Processor*) e três FPGAs (do inglês, *Field-Programmable Gate Array*), sendo duas FPGAs para entrada, recebendo dados de dois links de front-end e uma para a saída, fornecendo uma interface para o resto da ROD. O DSP executa o algoritmo de estimação *online* de

energia, garantindo a reconstrução do sinal respeitando a taxa máxima de eventos permitida pelo sistema de filtragem do ATLAS.

Devido ao programa de atualização do **LHC**, atualmente a colaboração **ATLAS** está trabalhando em mudanças necessárias para atualização de vários componentes do TileCal, como por exemplo a eletrônica de *front-end* e *back-end*. A obrigatoriedade da atualização contempla as novas exigências de alta luminosidade proposta no programa do **LHC**.

A luminosidade é compreendida como a medida do número de colisões que podem ser produzidas em um detector por cm^2 e por segundo. O aumento da luminosidade significa que o feixe será mais denso e, assim sendo, mais interações próton-próton ocorrerão quando os feixes se cruzarem (colisão). A luminosidade tem unidade $cm^2 s^{-1}$ e pode ser calculada através da Equação (2.2),

$$L \sim \frac{N^2}{t \cdot S}, \quad (2.2)$$

onde N corresponde ao número de prótons. O parâmetro t representa o tempo entre colisões e S , a seção transversal do feixe.

Dentro deste contexto, o algoritmo de estimativa online de energia passará a ser processado por FPGAs modernas, substituindo os antigos dispositivos DSPs presentes na ROD. Além disso, o método de estimativa de energia deve garantir que a reconstrução do sinal seja realizada de forma apropriada.

A seguir são apresentados com maiores detalhes os efeitos da alta luminosidade para a estimativa de energia no TileCal (efeito de empilhamento de sinais), as fases do plano de atualização e a atualização da eletrônica do TileCal (que contempla o desenvolvimento da nova ROD), sendo o TileCal o ambiente de pesquisa deste trabalho, cujo o objetivo é estimar, em FPGAs modernas e baseado em um método eficiente (**COF**), a amplitude de sinais empilhados independentemente das informações de luminosidade, recuperando, além do sinal de interesse, os sinais sobrepostos.

2.3 Os Efeitos da Alta Luminosidade e o Plano de Atualização

Como a resposta do TileCal é mais lenta que o período de colisão, com o aumento da luminosidade, sinais provenientes de colisões adjacentes serão observados ocasionando o efeito de empilhamento de sinais. A Figura 2.13 ilustra o empilhamento de sinais em uma mesma janela de 150 ns.

Na figura, as amostras digitalizadas são representadas por pontos e o sinal de interesse, representado pela cor preta, está no cruzamento do feixe (BC, do inglês

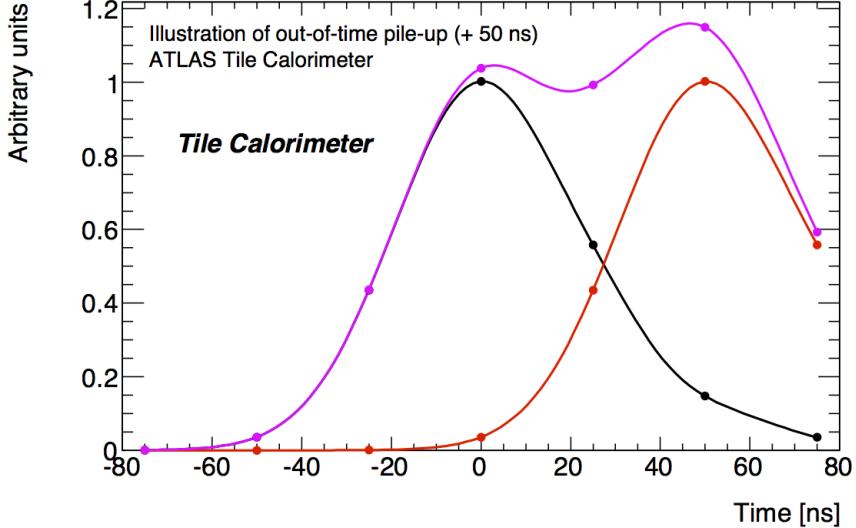


Figura 2.13: Fenômeno de empilhamento de sinais no TileCal [extraído de [26]].

Bunch Crossing) central, ou seja, BC=0. Com os efeitos da alta luminosidade, outro sinal, de outra colisão (50 ns atrasado) é observado distorcendo o pulso final.

O objetivo do algoritmo de reconstrução do TileCal é estimar com a maior precisão possível a energia do evento que ocorreu no centro da janela. O efeito de empilhamento de sinais tem impacto direto nos algoritmos, pois deteriora o desempenho da reconstrução dos sinais, refletindo diretamente na performance do sistema de calorimetria. Os detalhes da atualização do TileCal são apresentados a seguir.

2.3.1 Fases do Plano de Atualização

O LHC prevê três fases de atualização, com três longas paradas do colisor. A primeira iniciou-se no final de 2012 e se estendeu até meados de 2015, com planos de atingir o pico de luminosidade nominal de $10^{34} \text{cm}^2 \text{s}^{-1}$ (FASE-0). A segunda está prevista para começar em meados de 2019 e tem duração de dois anos, e os planos concentram-se em se preparar para atingir o pico de luminosidade em torno de $2 - 3 \times 10^{34} \text{cm}^2 \text{s}^{-1}$, correspondente de 50 a 80 interações médias por BC com intervalo de 25ns (FASE-I). A terceira longa parada terá início em 2024, com duração prevista de dois anos e meio, se preparando para pico de luminosidade em torno de $5 - 7 \times 10^{34} \text{cm}^2 \text{s}^{-1}$, correspondente a aproximadamente 200 interações por BC (FASE-II).

Nestas fases, o programa de atualização da eletrônica do TileCal visa a substituição do sistema eletrônico atual por componentes mais robustos à radiação, com menos conectores e de melhor precisão, fornecendo uma informação com maior granularidade e resolução para o nível 1 de filtragem de eventos do **ATLAS**. A nova

arquitetura eletrônica está sendo testada de forma incremental, onde três novas opções de projeto para a nova placa de *front-end* estão sob avaliação. Incluem também o desenvolvimento de uma nova eletrônica para os componentes do bloco PMT, considerando a exigência de alta tensão e corrente. Já na eletrônica de *back-end*, a ROD será substituída pela "super" *Read-Out Driver*, ou sROD, com a finalidade de reconstruir também a energia das células na taxa de colisão (40 MHz).

2.3.2 Super Read Out Driver

A nova arquitetura da ROD, a sROD, é constituída de uma FPGA Xilinx Virtex 7 e uma Kintex 7 como núcleos de processamento. A sROD executa várias funções que incluem recepção e processamento dos dados da *Daughter Board*; *Timing*, *Trigger* e Controle (TTC); Sistema de Controle de Detecção (DCS), Gestão e Transmissão para a eletrônica de front-end; reconstrução e transmissão de dados para a *ReadOut Subsystem* (ROS); bem como o pré-processamento e transmissão de dados para o nível 1 [27]. A Virtex 7 está ligado a 4 QSFP que fornecem uma comunicação de alta velocidade com a eletrônica de front-end em uma taxa de dados máxima de 160 Gb/s. A Kintex 7 permite uma compatibilidade com a eletrônica atual (ROD), recebe informações TCC e faz a interface com o sistema de *trigger*. A Figura 2.14 apresenta diagrama de blocos da sROD.

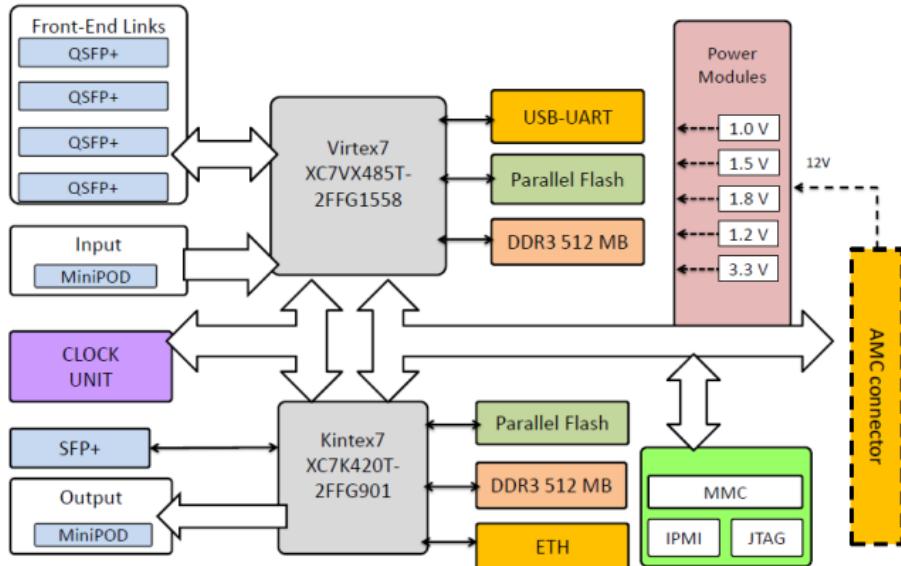


Figura 2.14: Diagrama de Blocos da sROD [extraído de [27]].

Através da sROD, a estimativa de energia que é feita a partir da estimativa da amplitude do pulso do TileCal, poderá ser realizada em dispositivos modernos, permitindo a implementação de novas técnicas para processamento *online*. Desta forma, este trabalho propõe a adaptação de um algoritmo de reconstrução mais

eficiente através de processos iterativos, estimando a amplitude de sinais empilhados independentemente das informações de luminosidade, recuperando, além do sinal de interesse, os sinais sobrepostos. Para rodar em ambiente com empilhamento de sinais é proposta uma arquitetura de processamento sequencial em FPGA para ser implementada na nova ROD. Sendo assim, já conhecido o ambiente de pesquisa e os efeitos da alta luminosidade para a estimação de energia no TileCal, apresenta-se no Capítulo 3 a revisão bibliográfica de métodos para a estimação de amplitude de sinais.

Capítulo 3

Revisão Bibliográfica

Neste capítulo, será realizado uma revisão bibliográfica sobre técnicas de estimação de energia relacionadas a calorimetria. A Seção 3.1 apresenta a formulação matemática baseada no melhor estimador linear imparcial, chamado de **BLUE** (do inglês, *Best Linear Unbiased Estimator*). Ainda nesta seção, será apresentada a generalização do **BLUE** para um vetor de parâmetros, que é a base para os métodos implementados no TileCal e outros calorímetros no **LHC**. A Seção 3.2 descreve o algoritmo *online* de reconstrução de energia do TileCal, chamado de **OF**. Finalizando este capítulo, a Seção 3.3 apresenta o método **OF** com restrições adicionais, chamado de **COF** (do inglês, *Constrained Optimal Filter*). O **COF** foi proposto pela colaboração brasileira [28], possuindo maior eficiência em altas luminosidades. Porém, na forma como foi apresentado até então, o **COF** não é indicado para aplicação *online*, devido à sua alta complexidade computacional. O foco deste trabalho está relacionado a implementação *online* do **COF** em FPGAs modernas.

3.1 Melhor Estimador Linear Imparcial - BLUE

Em problemas relacionados com a estimação de amplitude de processos aleatórios com restrições relacionadas a complexidade computacional do método, técnicas lineares de estimação apresentam-se como melhores alternativas devido a sua simplicidade de implementação. No TileCal, devido à sua cadeia eletrônica, a estimativa de energia depositada em um determinado canal pode ser realizada através da estimativa da amplitude de um pulso de forma fixa imerso em ruído. Para este caso particular, o método **BLUE** restringe-se a um estimador linear, de modo a determinar os pesos ótimos de um filtro que minimiza a variância da estimativa [29]. O método resulta em uma equação matemática simples, baseada na soma de produtos, facilitando a implementação em co-processadores para reconstrução *online* de sinais.

Para compreensão da equação geral do método **BLUE**, um modelo para estimativa da amplitude de um sinal determinístico [30] é apresentado. Neste, as amostras

$r[n]$ do sinal de entrada digitalizado são modeladas através da Equação (3.1),

$$r[n] = ag[n] + w[n], \text{ para } 1 < n < N, \quad (3.1)$$

onde a é a amplitude do sinal e $g[n]$ representa as amostras do sinal de referência, como por exemplo, o sinal de referência do TileCal (Figura 2.11). A componente $w[n]$ trata-se de um ruído aditivo com média zero. Sendo assim, para a presente aplicação, a equação geral do método **BLUE** pode ser definida como o produto interno entre os pesos do filtro e as N amostras do sinal de entrada, conforme a Equação (3.2),

$$\hat{a} = \sum_{n=1}^N h_n r[n], \quad (3.2)$$

onde \hat{a} é a estimativa da amplitude do sinal. Os coeficientes h_n , chamados de pesos do **BLUE**, são os parâmetros a serem determinados.

As características do ruído e das amostras do sinal de entrada $r[n]$ para o TileCal serão melhor detalhadas em seções posteriores. O objetivo a seguir é apresentar a formulação matemática para determinar os pesos do **BLUE**.

3.1.1 Determinando os pesos do BLUE

Para que os pesos do **BLUE** sejam encontrados, é imposta a imparcialidade na estimativa de a . Um estimador é dito imparcial, ou não-tendencioso, se o valor esperado da estimativa for igual ao valor real. Portanto, para um estimador linear não-tendencioso, temos que

$$\begin{aligned} E\{\hat{a}\} &= a \\ E\{\hat{a}\} &= \sum_{n=1}^N h_n E\{r[n]\} = a, \end{aligned} \quad (3.3)$$

onde $E\{\cdot\}$ representa o valor esperado. Logo, substituindo a Equação (3.1) na Equação (3.3), o valor esperado da estimativa da amplitude $E\{\hat{a}\}$ poderá ser reescrita como

$$E\{\hat{a}\} = \sum_{n=1}^N h_n E\{ag[n] + w[n]\} = a. \quad (3.4)$$

Seja $w[n]$ um ruído aditivo com média zero, então a Equação (3.4) para o valor esperado da estimativa da amplitude $E\{\hat{a}\}$ pode ser reescrita como mostra a Equação (3.5),

$$E\{\hat{a}\} = a \sum_{n=1}^N h_n g[n] = a. \quad (3.5)$$

Portanto, para manter a imparcialidade, a Equação (3.5) apresenta uma restrição a ser inserida no processo de minimização. Essa restrição impõe que o produto interno entre os pesos do **BLUE** e as amostras do sinal de referência $g[n]$ seja igual a unidade, conforme a Equação (3.6),

$$\sum_{n=1}^N h_n g[n] = 1. \quad (3.6)$$

Desta forma, o valor esperado da estimativa da amplitude $E\{\hat{a}\}$ será igual a própria constante a e, dado que a constante a representa a amplitude real das amostras do sinal de entrada $r[n]$, a propriedade de imparcialidade apresentada pela Equação (3.3) é atendida. Uma vez determinada a restrição para os pesos do **BLUE**, o objetivo passa a ser minimizar a variância da estimativa de a levando em conta esta restrição. A Equação (3.7) apresenta a variância \hat{a} .

$$\begin{aligned} var(\hat{a}) &= E\{(\hat{a} - E\{\hat{a}\})^2\} \\ &= E\left\{\left(\sum_{n=1}^N h_n r[n] - E\left\{\sum_{n=1}^N h_n r[n]\right\}\right)^2\right\} \end{aligned} \quad (3.7)$$

Buscando facilitar a compreensão dos procedimentos de minimização da variância da estimativa da amplitude \hat{a} , os passos seguintes terão seus elementos definidos em notação vetorial, representados por termos em negrito e utilizando letras minúsculas seguindo a convenção de vetor coluna. Para representação de um termo na forma matricial, letras maiúsculas e em negrito foram utilizadas. Denotando h_n e $r[n]$ em sua forma vetorial, encontra-se a variância como mostra a Equação (3.8),

$$\begin{aligned}
var(\hat{a}) &= E \left\{ (\mathbf{h}^T \mathbf{r} - \mathbf{h}^T \bar{\mathbf{r}})^2 \right\} \\
&= E \left\{ (\mathbf{h}^T (\mathbf{r} - \bar{\mathbf{r}}))^2 \right\} \\
&= E \left\{ \mathbf{h}^T (\mathbf{r} - \bar{\mathbf{r}}) (\mathbf{r} - \bar{\mathbf{r}})^T \mathbf{h} \right\} \\
&= \mathbf{h}^T \mathbf{C} \mathbf{h},
\end{aligned} \tag{3.8}$$

onde \mathbf{C} é a matriz de covariância do ruido. A etapa seguinte consiste em aplicar o método dos multiplicadores de Lagrange [31], dado a restrição (3.6). Como apenas uma restrição é aplicada, apenas um multiplicador é utilizado no método, como mostra a Equação (3.9),

$$J = \mathbf{h}^T \mathbf{C} \mathbf{h} + \lambda (\mathbf{h}^T \mathbf{g} - 1), \tag{3.9}$$

onde λ é multiplicador de Lagrange e \mathbf{g} é a forma vetorial de $g[n]$ (sinal de referência).

O método de Lagrange, formado pela combinação linear dada pela Equação (3.9), determina o ponto mínimo da variância da estimativa da amplitude $var(\hat{a})$ condicionada à restrição imposta pela Equação (3.6). Isto é feito derivando a Equação (3.9) e igualando a zero, como mostra a Equação (3.10),

$$\frac{\partial J}{\partial \mathbf{h}} = \frac{\partial}{\partial \mathbf{h}} [\mathbf{h}^T \mathbf{C} \mathbf{h} + \lambda (\mathbf{h}^T \mathbf{g} - 1)] = 0. \tag{3.10}$$

As próximas etapas da formulação matemática consistem em isolar variáveis e substituí-las nas equações apresentadas anteriormente para determinar os pesos do **BLUE** e, consequentemente, a estimativa ótima da amplitude do sinal. Inicialmente, derivando-se a Equação (3.10) e isolando a variável que corresponde aos pesos do BLUE, encontra-se a Equação (3.11),

$$\begin{aligned}
2\mathbf{C}\mathbf{h} + \lambda\mathbf{g} &= \mathbf{0} \\
\mathbf{h} &= -\frac{1}{2} \mathbf{C}^{-1} \mathbf{g} \lambda.
\end{aligned} \tag{3.11}$$

Em seguida, utilizando a restrição dada pela Equação (3.6) em sua forma vetorial, o multiplicador de lagrange pode ser encontrado isolando-o e reescrevendo a Equação (3.11) condicionada à restrição, como mostra a Equação (3.12),

$$\begin{aligned}
\mathbf{h}^T \mathbf{g} &= -\frac{\lambda}{2} \mathbf{g}^T \mathbf{C}^{-1} \mathbf{g} = 1 \\
\frac{1}{\lambda} &= -\frac{1}{2} \mathbf{g}^T \mathbf{C}^{-1} \mathbf{g} \\
\lambda &= -2 (\mathbf{g}^T \mathbf{C}^{-1} \mathbf{g})^{-1}.
\end{aligned} \tag{3.12}$$

Substituindo a Equação (3.12) na Equação (3.11), os pesos ótimos do **BLUE** são encontrados, conforme a Equação (3.13),

$$\begin{aligned}
\mathbf{h}_{otimo} &= \left[-\frac{1}{2} (\mathbf{C}^{-1} \mathbf{g}) \right] \left[-2 (\mathbf{g}^T \mathbf{C}^{-1} \mathbf{g})^{-1} \right] \\
&= \frac{\mathbf{C}^{-1} \mathbf{g}}{\mathbf{g}^T \mathbf{C}^{-1} \mathbf{g}}.
\end{aligned} \tag{3.13}$$

que também é conhecido como *linearly Constrained Minimum Variance* (LCMV) [32].

Finalmente, substituindo a Equação (3.13) na Equação (3.2), a estimativa ótima da amplitude do sinal é determinada por

$$\hat{a} = \frac{\mathbf{h}^T \mathbf{r}}{\mathbf{g}^T \mathbf{C}^{-1} \mathbf{g}} \tag{3.14}$$

Todos os procedimentos descritos, visam determinar os pesos ótimos de um filtro linear que estima a amplitude de um sinal determinístico. Entretanto, o método **BLUE** pode ser generalizado para estimar um vetor de parâmetros linearmente combinados. A seguir apresenta-se a formulação matemática da extensão do **BLUE** para um caso onde se deseja estimar as amplitudes de sinais determinísticos sobrepostos.

3.1.2 BLUE para um vetor de parâmetros

Anteriormente, o modelo para estimação da amplitude de um sinal determinístico foi apresentado. Aqui, reescrevemos a Equação (3.1) para o caso de sinais determinísticos sobrepostos, de acordo com a Equação (3.15),

$$\mathbf{r} = \mathbf{G} \mathbf{a} + \mathbf{w}, \tag{3.15}$$

onde \mathbf{a} é o vetor coluna que contém as amplitudes dos sinais sobrepostos, como por exemplo, sinais do calorímetro defasados no tempo (ver Figura 2.13). As colunas da matriz \mathbf{G} representam as amostras dos sinais de referência, cuja as amplitudes estão contidas no vetor \mathbf{a} . Em seguida, reescrevendo a Equação (3.2), definimos a estimativa resultante do vetor de amplitudes $\hat{\mathbf{a}}$, conforme é apresentado pela Equação (3.16),

$$\hat{\mathbf{a}} = \mathbf{H}^T \mathbf{r}, \quad (3.16)$$

sendo que \mathbf{H} representa a matriz de coeficientes do banco de filtros a ser determinado. Para que os pesos do **BLUE** generalizado para um vetor de parâmetros sejam encontrados, é imposta a imparcialidade na estimativa de \mathbf{a} . Logo, reescrevendo a Equação (3.16) para um estimador não-tendencioso, temos que o valor esperado $E\{\hat{\mathbf{a}}\}$ da estimativa da amplitude é dado por

$$E\{\hat{\mathbf{a}}\} = \mathbf{H}^T E\{\mathbf{r}\} = \mathbf{a}. \quad (3.17)$$

Assim como apresentado para as Equações (3.3) e (3.4), aqui, substituindo a Equação (3.15) na Equação (3.17), o valor esperado da estimativa da amplitude $E\{\hat{\mathbf{a}}\}$ poderá ser obtido, conforme a Equação (3.18),

$$E\{\hat{\mathbf{a}}\} = \mathbf{H}^T E\{\mathbf{G}\mathbf{a} + \mathbf{w}\} = \mathbf{a}. \quad (3.18)$$

Seja o vetor \mathbf{w} um ruido aditivo com média zero, então a Equação (3.18) para o valor esperado da estimativa da amplitude $E\{\hat{\mathbf{a}}\}$ pode ser reescrita, como mostra a Equação (3.19),

$$E\{\hat{\mathbf{a}}\} = \mathbf{H}^T \mathbf{G}\mathbf{a} = \mathbf{a}. \quad (3.19)$$

Na Seção 3.1.1 mostrou-se que, para atender a propriedade de imparcialidade, o valor esperado da estimativa da amplitude $E\{\hat{\mathbf{a}}\}$ deve ser igual a amplitude real do sinal de entrada \mathbf{r} . A partir dessa suposição, apenas uma restrição foi inserida no processo de minimização da variância. Diferentemente, para o método **BLUE** generalizado para um vetor de parâmetros, p restrições são inseridas no processo de minimização de cada componente, onde p é o número de sinais sobrepostos. Então, no presente método, para que $E\{\hat{\mathbf{a}}\} = \mathbf{a}$, o produto entre a matriz dos pesos do

BLUE \mathbf{H} e a matriz do pulso de referência \mathbf{G} deve ser igual a matriz identidade \mathbf{I} , conforme é apresentado pela Equação (3.20),

$$\mathbf{H}^T \mathbf{G} = \mathbf{I}. \quad (3.20)$$

Reescrevendo a Equação (3.8) em sua forma matricial, temos a Equação (3.21),

$$cov(\hat{\mathbf{a}}) = \mathbf{H}^T \mathbf{C} \mathbf{H}, \quad (3.21)$$

onde $cov(\hat{\mathbf{a}})$ é a matriz de covariância dos estimadores. Esta matriz porém, apresenta elementos não nulos apenas em sua diagonal, pois a Equação (3.20) força a ortogonalidade entre os estimadores. Os elementos da diagonal dessa matriz correspondem às variâncias dos estimadores, sendo necessário, portanto, uma função custo para cada um desses elementos, como mostra a Equação (3.22)

$$J(a_i) = \mathbf{h}_i^T \mathbf{C} \mathbf{h}_i + \lambda_0 (\mathbf{h}_i^T \mathbf{g}_i - 1) + \sum_{j=1}^p \lambda_j \mathbf{h}_j \mathbf{g}_i, \text{ para } j \neq i, \quad (3.22)$$

onde a_i corresponde à amplitude estimada pelo i -ésimo vetor de pesos \mathbf{h}_i . Nesta função custo são necessários p multiplicadores de Lagrange, onde p corresponde ao número de sinais empilhados. Desses multiplicadores, o primeiro (λ_0), como no caso escalar, é necessário para manter a imparcialidade do estimador. Os demais, $p - 1$ multiplicadores, são restrições adicionais (ver Equação (3.20)) para forçar a ortogonalidade entre os estimadores, representados pelo somatório dado pela Equação (3.22).

Após manipulação matemática, para eliminar os multiplicadores de Lagrange, o resultado final do processo de minimização pode ser agrupado em uma única equação matricial [29]

$$\mathbf{H}_{\text{ótimo}} = \mathbf{C}^{-1} \mathbf{G} \left(\mathbf{G}^T \mathbf{C}^{-1} \mathbf{G} \right)^{-1}. \quad (3.23)$$

Substituindo a Equação (3.23) na Equação (3.16), a estimativa ótima da amplitude do sinal é determinada por meio da Equação (3.24),

$$\begin{aligned} \hat{\mathbf{a}} &= \mathbf{H}^T \mathbf{r} \\ &= \left(\mathbf{G}^T \mathbf{C}^{-1} \mathbf{G} \right)^{-1} \mathbf{G}^T \mathbf{C}^{-1} \mathbf{r} \end{aligned} \quad (3.24)$$

que também é conhecido como *Minimum Variance Distortionless Response* (MVDR) [33].

Devido à similaridade na formulação matemática dos métodos, uma tabela Comparativa entre BLUE Escalar e a um Vetor de Parâmetros é apresentada.

Tabela 3.1: Tabela Comparativa entre BLUE Escalar e a um Vetor de Parâmetros.

BLUE ESCALAR	BLUE VETOR de PARÂMETROS
$\mathbf{r} = \mathbf{ag} + \mathbf{w}$	$\mathbf{r} = \mathbf{Ga} + \mathbf{w}$
$\hat{\mathbf{a}} = \mathbf{h}^T \mathbf{r}$	$\hat{\mathbf{a}} = \mathbf{H}^T \mathbf{r}$
$E\{\hat{\mathbf{a}}\} = \mathbf{h}^T E\{\mathbf{r}\} = \mathbf{a}$	$E\{\hat{\mathbf{a}}\} = \mathbf{H}^T E\{\mathbf{r}\} = \mathbf{a}$
$E\{\hat{\mathbf{a}}\} = \mathbf{h}^T E\{\mathbf{ag} + \mathbf{w}\} = \mathbf{a}$	$E\{\hat{\mathbf{a}}\} = \mathbf{H}^T E\{\mathbf{Ga} + \mathbf{w}\} = \mathbf{a}$
$E\{\hat{\mathbf{a}}\} = \mathbf{a} \mathbf{h}^T \mathbf{g} = \mathbf{a}$	$E\{\hat{\mathbf{a}}\} = \mathbf{H}^T \mathbf{G} \mathbf{a} = \mathbf{a}$
$\mathbf{h}^T \mathbf{g} = 1$	$\mathbf{H}^T \mathbf{G} = \mathbf{I}$
$\text{var}(\hat{\mathbf{a}}) = \mathbf{h}^T \mathbf{C} \mathbf{h}$	$\text{cov}(\hat{\mathbf{a}}) = \mathbf{H}^T \mathbf{C} \mathbf{H}$
$\mathbf{h}_{óptimo} = \mathbf{C}^{-1} \mathbf{g} (\mathbf{g}^T \mathbf{C}^{-1} \mathbf{g})^{-1}$	$\mathbf{H}_{óptimo} = \mathbf{C}^{-1} \mathbf{G} (\mathbf{G}^T \mathbf{C}^{-1} \mathbf{G})^{-1}$
$\hat{\mathbf{a}} = (\mathbf{g}^T \mathbf{C}^{-1} \mathbf{g})^{-1} \mathbf{g}^T \mathbf{C}^{-1} \mathbf{r}$	$\hat{\mathbf{a}} = (\mathbf{G}^T \mathbf{C}^{-1} \mathbf{G})^{-1} \mathbf{G}^T \mathbf{C}^{-1} \mathbf{r}$

3.2 Método OF

O algoritmo *online* de reconstrução de energia do TileCal, reconstrói a amplitude, a fase e o pedestal, de um sinal analógico a partir de suas amostras digitalizadas [34]. A formulação matemática do método **OF** consiste em determinar os pesos ótimos de um filtro linear que minimiza a contribuição das fontes de ruido para reconstrução dos parâmetros (amplitude, tempo e pedestal) do sinal de interesse. O método resulta em uma combinação linear entre os pesos do filtro e as 7 amostras discretas disponibilizadas pela eletrônica após a conversão analógico-digital.

Assim como descrito no método **BLUE**, para a compreensão do algoritmo **OF**, inicialmente se faz necessário apresentar o modelo linear adotado para estimativa de sinais. Neste, o modelo do sinal do TileCal é expresso pela Equação (3.25),

$$r(t) = ag(t - \tau) + ped + w(t), \quad (3.25)$$

onde a é a amplitude real do sinal e $g(t)$ representa a forma do pulso característico do TileCal, chamado de pulso de referência e apresentado na Figura 2.11. O termo ped é uma constante representando a linha de base e $w(t)$ é o ruído característico do ambiente do TileCal correspondendo a soma do ruído eletrônico e a contribuição

de sinais adjacentes empilhados. O conjunto de amostras que são retiradas a partir de $r(t)$ em intervalos regulares n é dadas pela Equação (3.26),

$$r[n] = ag[n] - a\tau g'[n] + ped + w[n], \quad (3.26)$$

onde τ é a fase entre o sinal digitalizado e o pulso de referência. Os termos τ , a e ped são os parâmetros a serem determinados pelo **OF**. Neste caso, a fim de linearizar a dependência de r com o parâmetro τ , aplicou-se a expansão em série de Taylor de primeira ordem.

A partir da expansão em série de Taylor, o modelo para estimação da amplitude representado pela Equação (3.26) pode ser generalizado pela extensão do **BLUE** para o caso onde se deseja estimar as amplitudes de três sinais sobrepostos. Desta forma, a Equação (3.26) é equivalente a Equação (3.15), onde \mathbf{a} é o vetor que contém as amplitudes a , $-a\tau$ e ped , dos sinais sobrepostos $g[n]$, $g'[n]$ e $\mathbf{1}$ (vetor cujo os elementos são iguais a um), conforme é apresentado pela Equação (3.27),

$$\mathbf{a} = \begin{bmatrix} a \\ -a\tau \\ ped \end{bmatrix}. \quad (3.27)$$

A primeira coluna da matriz \mathbf{G} contém as amostras do pulso de referência g . Os elementos da segunda coluna representam as amostras da derivada do pulso de referência g , logo g' . Já a última coluna da matriz contém apenas valores unitários, visto que está relacionada a linha de base. Todas as amostras dos sinais sobrepostos contidos na matriz \mathbf{G} são apresentadas na Equação (3.28),

$$\mathbf{G} = \begin{bmatrix} g_1 & g'_1 & 1 \\ g_2 & g'_2 & 1 \\ g_3 & g'_3 & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ g_7 & g'_7 & 1 \end{bmatrix}. \quad (3.28)$$

Baseado no método **BLUE** para um vetor de parâmetros, os pesos do método **OF** são encontrados de acordo com a Equação (3.23), onde \mathbf{C} é a matriz de covariância do ruído contendo a estatística de segunda ordem do ruído eletrônico mais o ruído decoorente do empilhamento de sinais. Logo a estimativa da amplitude do algoritmo

OF é determinada por meio da Equação (3.24).

Após a obtenção dos parâmetros, o valor de fase τ é obtido, fazendo,

$$\hat{\tau} = -\frac{\hat{a}_2}{\hat{a}_1}, \quad (3.29)$$

onde \hat{a}_1 e \hat{a}_2 são o primeiro e segundo elementos, respectivamente, do vetor $\hat{\mathbf{a}}$ da Equação (3.24).

Por ser baseado no **BLUE**, o método **OF** [34] é otimizado para estimar a amplitude de um sinal determinístico em ruído gaussiano. Entretanto, devido ao aumento do número de colisões prótons-prótons, ocorre o efeito de empilhamento de sinais [30] (ver Figura 2.13). Consequentemente, o empilhamento de sinais em condição de alta luminosidade degrada o pulso de referência comprometendo a estimativa da energia. No método **OF**, o empilhamento de sinais é tratado como ruído e a estatística de segunda ordem do mesmo é usada no projeto do filtro. Porém, o ruído deixa de ser Gaussiano nestas condições, o que torna o filtro sub-ótimo. Além disso, o projeto do filtro torna-se dependente da luminosidade.

3.3 Método COF

Tendo em vista os problemas do método **OF** na presença de sinais empilhados e no intuito de solucioná-los, a colaboração brasileira propôs o método **OF** com restrições adicionais, chamado de **COF** [28]. A proposta visa estimar além da amplitude do pulso de interesse, a amplitude de cada componente sobreposta. Nestas condições, o ruído é caracterizado somente pelo ruído eletrônico que é Gaussiano, mantendo a característica ótima do estimador, além de manter seu projeto independente da luminosidade. No método **COF** a fase é desprezada e o pedestal é calibrado e previamente retirado.

Para que o **COF** funcione corretamente é necessário um pré-processamento para identificação dos BCs (*Bunch Crossing*) com ocorrência de sinais empilhados. No **COF**, a matriz \mathbf{G} contém os sinais defasados que foram detectados. Para o caso em que somente esteja presente a informação relevante para o sistema de *trigger* do TileCal, ou seja, apenas o pulso central da janela, os pesos de um filtro linear seriam dados pela Equação (3.23), substituindo a matriz \mathbf{G} pelo vetor \mathbf{g} com as amostras do sinal de referência do TileCal. Portanto, na ausência de sinais empilhados, e assumindo que o ruído eletrônico é gaussiano branco (WG), a Equação (3.23) pode ser reescrita conforme a Equação (3.30),

$$\mathbf{h}_{\text{ótimo}} = \frac{\mathbf{g}}{\|\mathbf{g}\|^2}. \quad (3.30)$$

Para o caso particular em que se observa sinal em algum BC e assumindo que o ruído eletrônico é WG, a matriz de covariância do ruído \mathbf{C} é diagonal e, portanto, a Equação (3.23) pode ser simplificada, conforme é apresentado pela Equação (3.31),

$$\mathbf{H}_{\text{ótimo}} = \mathbf{G}(\mathbf{G}^T \mathbf{G})^{-1}. \quad (3.31)$$

Assim, a Equação (3.24) também pode ser reescrita, conforme a Equação (3.32),

$$\hat{\mathbf{a}} = (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T \mathbf{r}, \quad (3.32)$$

onde \mathbf{G} contém os sinais de referência relacionados aos BCs onde sinais empilhados foram observado.

O processo de detecção de sinais empilhados é um pré-processamento importante para o **COF** e também pode ser executado com a ajuda da Equação (3.32), onde \mathbf{G} contém os sete sinais mais centrais. No caso especial em que se deseja estimar as 7 componentes sobrepostas, por ser uma matriz quadrada, podemos reescrever a Equação (3.31) reduzindo-a, como mostra a Equação (3.33). Desta forma, a inversa da matriz \mathbf{G}^T é, portanto, chamada de matriz de deconvolução (**DM**),

$$\mathbf{H}_{\text{ótimo}} = (\mathbf{G}^T)^{-1}. \quad (3.33)$$

O processo de deconvolução indica a existência de informações de sinais empilhados. Estas informações são então comparadas com um limiar. Se as mesmas forem maiores do que este limiar, então o **COF** será capaz de reconstruir a energia de cada componente detectada com o mínimo de restrições. As figuras a seguir ilustram o diagrama do método **COF**.

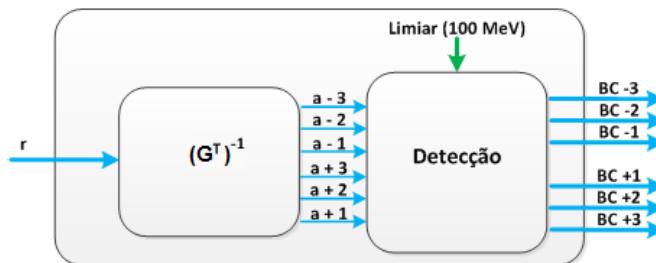


Figura 3.1: Identificação de sinais empilhados e a reconstrução da energia.

Além de reconstruir a amplitude de todos os sinais sobrepostos e manter as características ótimas do estimador, o método **COF** é independente da luminosidade.

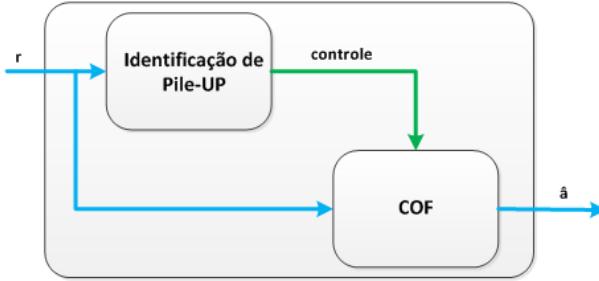


Figura 3.2: Diagrama Geral do método COF.

A Figura (3.3) apresenta o histograma de reconstrução de energia, comparando a eficiência entre o **OF** e o **COF**. O **OF2** é equivalente ao algoritmo **OF** apresentado neste trabalho. Estas análises referem-se a dados reais tomados em 2012, com uma energia de colisão de 8 TeV e uma luminosidade de $\mu = 11.3$. Nestas condições de luminosidade o efeito de empilhamento de sinais já pode ser percebido. Neste gráfico é dado evidência apenas na região do ruído (± 200 MeV), onde se pode notar uma melhor resolução em energia para o **COF** (maior concentração em torno do zero).

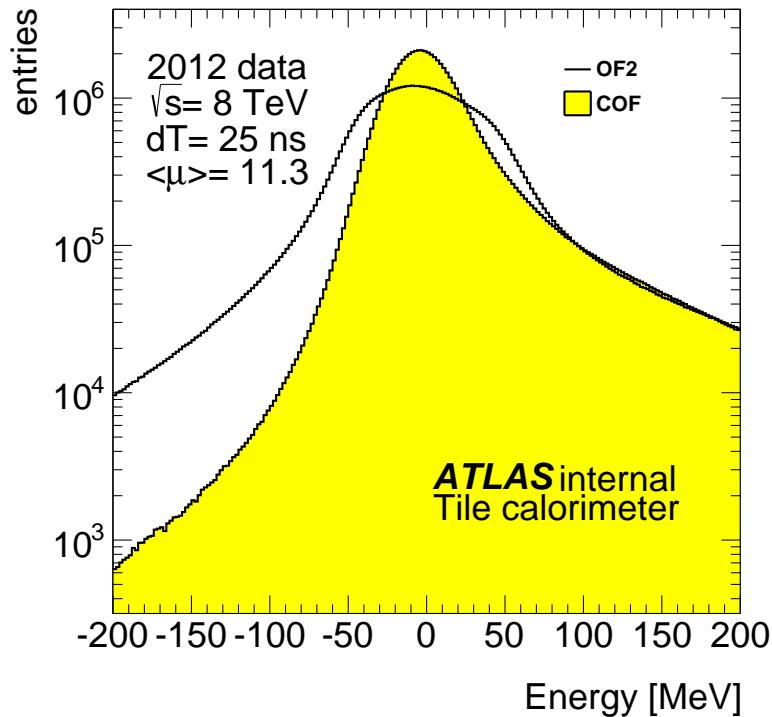


Figura 3.3: Histograma de Reconstrução da Energia [extraído de [35]].

A Figura (3.4) apresenta um gráfico de correlação entre as reconstruções de energia feitas pelo **OF** e o **COF** para a mesma tomada de dados da figura anterior. Nesta figura são evidenciadas situações onde ocorrem apenas empilhamento de sinais (OOT Signal $|BC| > 3$, OOT Signal $\pm 1, 2 \text{ e } 3$). Para os sinais dentro da janela tratada

pelo **COF** (até ± 3 *Bunch Crossing*) o mesmo se mostra imune na presença de sinais empilhados, pois estima a energia dentro da faixa de ruído ± 200 MeV. Em contrapartida o **OF** espalha esta energia de -2 a 2 GeV.

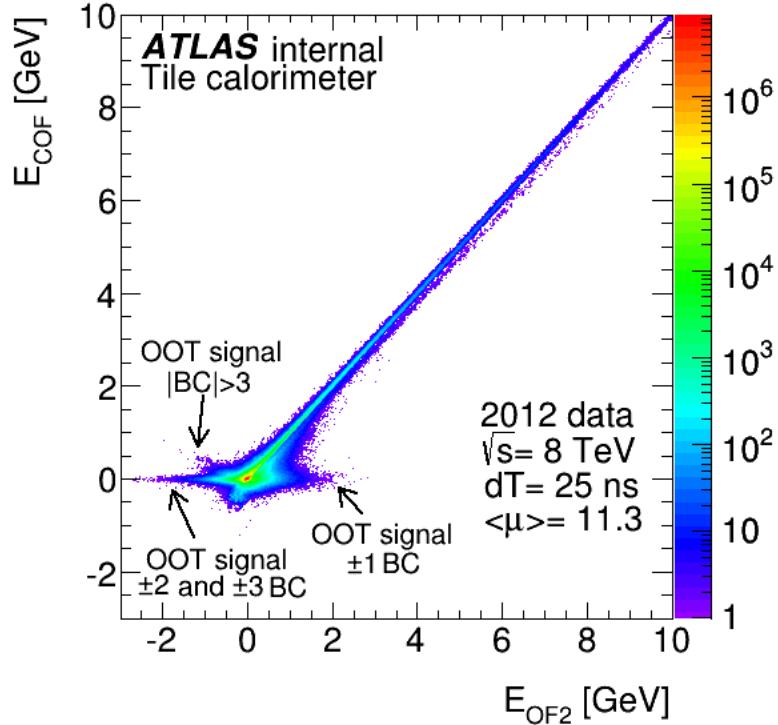


Figura 3.4: Correlação entre o **COF** e o **OF** [extraído de [35]].

O método **COF** resulta em uma formulação com alta complexidade para processamento *online*. Os pesos dos filtros são dinâmicos, dependentes das posições onde os sinais sobrepostos foram identificados. Portanto, a Equação (3.32) deve ser computada para cada evento. Neste processo, um procedimento de inversão de matrizes deve ser executado, dificultando a implementação *online* **COF**. Sendo assim, este trabalho propõe técnicas para evitar a complexidade matemática do **COF**, viabilizando seu processamento *online*, como mostra o Capítulo (4).

Capítulo 4

Propostas para Implementação *Online* do COF

Previsto o aumento da luminosidade no programa do **LHC**, o TileCal estará sujeito ao efeito do empilhamento de sinais. Como pôde-se observar, na Seção 3.2, o algoritmo **OF** não trata corretamente este efeito, pois considera o empilhamento de sinais como ruído, o que o torna sub-ótimo devido a não gaussianidade do ruído de sinais empilhados. Na Seção 3.3, pôde-se observar também, que o método **COF** foi formulado visando atender o efeito de empilhamento de sinais, através do qual estima-se a amplitude do pulso de interesse e a amplitude de cada componente sobreposta, sendo estas componentes, tratadas como ruído pelo método **OF**. O **COF** vem sendo utilizado para estimação *offline* no TileCal, e tem obtido excelentes resultados, porém o método resulta em inversão de matrizes e sua implementação para processamento *online* não é otimizada, posto que se faz necessário um procedimento de inversão de matrizes.

Tendo em vista o cenário de alta luminosidade do **LHC**, e de forma a implementar o **COF** iterativamente, a seguir são apresentados os métodos para adaptação do **COF**. Assim, a Seção 4.1 apresenta a formulação matemática para os algoritmos Gradiente Descendente, Gradiente Descendente com Taxa de Convergência Dinâmica e o algoritmo Gradiente Conjugado. Por fim, a Seção 4.2 apresenta a simulação dos métodos iterativos.

4.1 Adaptação do COF através de Métodos Iterativos

As presentes técnicas propõem a estimação da amplitude do pulso de interesse e de cada componente sobreposta através de métodos iterativos. Baseado na adaptação do **COF**, métodos resultam em simples operações matemáticas que viabilizarão

a implementação do **COF** em dispositivos FPGA para a estimação de energia.

4.1.1 Gradiente Descendente - GD

Partindo-se da função custo para a minimização do erro médio quadrático entre o sinal recebido \mathbf{r} e o modelo linear adotado para estimativa de sinais, dado pela Equação (3.15), temos a Equação (4.1),

$$J(\hat{\mathbf{a}}) = \|\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}\|^2, \quad (4.1)$$

onde \mathbf{r} é o sinal de entrada digitalizado, $\hat{\mathbf{a}}$ é o vetor coluna que contém as amplitudes dos sinais sobrepostos. As colunas da matriz \mathbf{G} representam as amostras dos sinais de referência, cuja as amplitudes estão contidas no vetor \mathbf{a} .

Derivando a Equação (4.1) em relação a $\hat{\mathbf{a}}$ e igualando a zero, encontra-se a Equação (4.2),

$$-\mathbf{2G}^T(\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}) = \mathbf{0}. \quad (4.2)$$

Isolando $\hat{\mathbf{a}}$ na Equação (4.2), encontramos a Equação (4.3) que corresponde à formulação do **COF** para o caso em que se observa o empilhamento de sinais em algum BC (Equação (3.32)),

$$\hat{\mathbf{a}} = (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T \mathbf{r}. \quad (4.3)$$

Sendo assim, utilizando-se de uma abordagem iterativa para minimizar (4.1) sem a necessidade de inversão de matrizes, devemos atualizar os valores de $\hat{\mathbf{a}}$ na direção contrária ao gradiente de J , em pequenos passos indexado por k , para aproximarmos dos estimadores de amplitude $\hat{\mathbf{a}}$ que minimizam o erro médio quadrático a cada iteração. Este procedimento é conhecido na literatura como Gradiente Descendente [36], e sua formulação geral é dado por,

$$\hat{\mathbf{a}}[k+1] = \hat{\mathbf{a}}[k] - \mu \frac{\partial J}{\partial \hat{\mathbf{a}}[k]}, \quad (4.4)$$

onde μ é a taxa de convergência do processo iterativo. Para este caso em particular, a amplitude de cada componente sobreposta é então reconstruída através do processo iterativo dado pela Equação (4.5),

$$\hat{\mathbf{a}}[k+1] = \hat{\mathbf{a}}[k] + \mu \mathbf{G}^T (\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k]), \quad (4.5)$$

onde o fator 2 é absorvido pela taxa de convergência μ (ver Equação (4.2)). O valor inicial $\hat{\mathbf{a}}[0]$ pode ser obtido com o processo de deconvolução dado pela Equação (3.33), para os sinais acima do limiar.

O algoritmo funcional do **GD** é apresentado no quadro a seguir. Inicialmente, o algoritmo recebe os valores das amplitudes estimadas pelo **DM** quando o empilhamento de sinais é observado em algum BC. Em seguida, define-se a taxa de aprendizagem para convergência do algoritmo. No passo seguinte, o algoritmo computa passo-a-passo os valores de cada componente sobreposta que convergem para as estimativas dos sinais acumulados.

Algorithm 1: ALGORITMO GRADIENTE DESCENDENTE

Data:

- \mathbf{r} : Sinal de entrada digitalizado
- \mathbf{G} : Sinais de referência selecionado pelo DM
- $\mathbf{a}[0] = \mathbf{G}^{-1} \mathbf{r}$: Amp. Estimada DM

Result:

- $\hat{\mathbf{a}}[k+1]$: Vetor de Amp. Estimadas, onde k é o índice de interação.

```

1  $\mu \leftarrow$  taxa de evolução;
2  $it \leftarrow$  máxima iteração;
3  $k \leftarrow$  valor da iteração inicial para convergência;
4 for  $k \leftarrow 1$  to  $it$  do
5   |  $\hat{\mathbf{a}}[k+1] \leftarrow \hat{\mathbf{a}}[k] + \mu \mathbf{G}^T (\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k]);$ 
6 end
```

Um detalhe importante para o algoritmo é encontrar μ que seja suficientemente pequeno para maior precisão e suficientemente grande para viabilizar a implementação *online* (necessidade de poucas iterações para convergência). O método é extremamente simples do ponto vista de complexidade computacional, pois resulta em simples operações de produto e soma. No entanto, para uma conversão rápida para o **COF**, deve-se garantir um número mínimo de iterações.

É sabido que, como a matriz $\mathbf{G}^T \mathbf{G}$ é simétrica, a mesma possui autovalores reais e tem-se que a maior taxa de convergência possível é $\mu = 2/\lambda_{max}$, onde λ_{max} é o

maior autovalor da matriz resultante $\mathbf{G}^T \mathbf{G}$ [37]. No entanto, no presente método, a matriz \mathbf{G} é dinâmica, pois a cada tomada de dados esta matriz poderá apresentar quantidades diferentes de sinais empilhados. Deste modo, não é possível encontrar um limite analítico para μ . Logo, o limite para μ deste método será encontrado a partir de dados simulados.

A próxima seção detalha o método do gradiente descendente com taxa de convergência (μ) dinâmica, onde um valor ótimo de μ é computado para cada iteração.

4.1.2 Gradiente Descendente com Convergência Dinâmica - GDD

No método **GDD** [38], o processo iterativo na determinação de $\hat{\mathbf{a}}$ é implementado de forma análoga à Equação (4.5). Entretanto, no presente método, determina-se o valor de μ capaz de obter o menor valor possível de J na direção de propagação a cada iteração. Assim, partindo da Equação (4.5) e assumindo o vetor direção \mathbf{d} equivalente à $\mathbf{G}^T (\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k])$, determina-se a equação geral do presente método,

$$\hat{\mathbf{a}}[k + 1] = \hat{\mathbf{a}}[k] + \mu \mathbf{d}. \quad (4.6)$$

Definida a Equação (4.6), os passos seguintes da formulação matemática consistem em determinar a taxa de convergência dinâmica μ . Para tal, inicialmente reescrevemos a Equação (4.1), conforme apresenta-se a Equação (4.7),

$$J(\hat{\mathbf{a}}) = (\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k + 1])^T (\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k + 1]), \quad (4.7)$$

onde $\hat{\mathbf{a}}[k + 1]$ representa o vetor de amplitudes da próxima iteração na direção de propagação. Em seguida, substitui-se a Equação (4.6) em $\hat{\mathbf{a}}[k + 1]$ fazendo com que o termo μ a ser determinado esteja presente na Equação (4.7), como mostra a Equação (4.8),

$$J(\mu) = \{\mathbf{r} - \mathbf{G}(\hat{\mathbf{a}}[k] + \mu \mathbf{d})\}^T \{\mathbf{r} - \mathbf{G}(\hat{\mathbf{a}}[k] + \mu \mathbf{d})\}. \quad (4.8)$$

Determinado J em termos de μ , os próximos passos consistem em minimizar a Equação (4.8) em relação a μ . Logo, derivando a Equação (4.8) e igualado a zero,

$$\frac{\partial J}{\partial \mu} = \frac{\partial}{\partial \mu} \left[\{\mathbf{r} - \mathbf{G}(\hat{\mathbf{a}}[k] + \mu \mathbf{d})\}^T \{\mathbf{r} - \mathbf{G}(\hat{\mathbf{a}}[k] + \mu \mathbf{d})\} \right] = 0, \quad (4.9)$$

encontramos a Equação (4.10),

$$\frac{\partial J}{\partial \mu} = -2(\mathbf{G}\mathbf{d})^T \{\mathbf{r} - \mathbf{G}(\hat{\mathbf{a}}[k] + \mu \mathbf{d})\} = 0. \quad (4.10)$$

Em seguida, resolvendo a Equação (4.10), encontramos a Equação (4.11),

$$\begin{aligned} \frac{\partial J}{\partial \mu} &= -2(\mathbf{G}\mathbf{d})^T \mathbf{r} + 2(\mathbf{G}\mathbf{d})^T G(\hat{\mathbf{a}}[k] + \mu \mathbf{d}) = 0 \\ &= -2\mathbf{d}^T \mathbf{G}^T \mathbf{r} + 2\mathbf{d}^T \mathbf{G}^T G(\hat{\mathbf{a}}[k] + \mu \mathbf{d}) = 0 \\ &= -2\mathbf{d}^T \mathbf{G}^T \mathbf{r} + 2\mathbf{d}^T \mathbf{G}^T \mathbf{G}\hat{\mathbf{a}}[k] + 2\mu\mathbf{d}^T \mathbf{G}^T \mathbf{G}\mathbf{d} = 0 \\ &= -2\mathbf{d}^T \mathbf{G}^T (\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k]) + 2\mu\mathbf{d}^T \mathbf{G}^T \mathbf{G}\mathbf{d} = 0. \end{aligned} \quad (4.11)$$

Sabendo que, $\mathbf{G}^T (\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k])$ é equivalente a \mathbf{d} , então ao isolarmos os demais termos da Equação (4.11) em função da taxa de convergência μ dinâmica, encontramos a Equação (4.12),

$$\mu = \frac{\mathbf{d}^T \mathbf{d}}{\mathbf{d}^T (\mathbf{G}^T \mathbf{G}) \mathbf{d}}. \quad (4.12)$$

Logo, substituindo a Equação (4.12) na Equação (4.6), encontramos a formulação para estimativa de sinais, expressa por,

$$\hat{\mathbf{a}}[k+1] = \hat{\mathbf{a}}[k] + \frac{\mathbf{d}^T \mathbf{d}}{\mathbf{d}^T (\mathbf{G}^T \mathbf{G}) \mathbf{d}} \mathbf{d}[k]. \quad (4.13)$$

O algoritmo funcional do **GDD** é apresentado no quadro a seguir. O valor inicial $\hat{\mathbf{a}}[0]$ do presente método, assim como no método **GC**, é obtido pelo processo de deconvolução (3.33). O algoritmo então, computa adaptativamente a direção de busca e a taxa de convergência para determinar passo-a-passo a estimativa da amplitude dos sinais.

Assim como o algoritmo **GD**, o presente método é dependente de uma boa escolha do ponto de partida ($\mathbf{a}[0]$) para convergência. O **GDD** busca o ponto de mínimo na direção atual $\mathbf{d}[k]$, onde a próxima direção de busca $\mathbf{d}[k+1]$ é sempre ortogonal a direção atual. O critério da ortogonalidade pode ser provado se nos restringirmos à direção, tal que $\mathbf{d}^T[k] \mathbf{d}[k+1] = 0$. Neste caso, ao ser imposta a restrição, é possível encontrar a taxa de convergência dinâmica, dada pela Equação (4.12). Consequentemente, o presente método localiza o ponto de mínimo com menor número iterações

Algorithm 2: ALGORITMO GRADIENTE D. ADAPTATIVO

Data:

- \mathbf{r} : Sinal de entrada digitalizado
- \mathbf{G} : Sinais de referência
- $\mathbf{a}[0] = \mathbf{G}^{-1} \mathbf{r}$: Amp. Estimada DM

Result:

- $\hat{\mathbf{a}}[k + 1]$: Vetor de Amp. Estimadas, onde k é o índice de interação.

```
1  $\mu \leftarrow$  taxa de evolução;
2  $it \leftarrow$  máxima iteração;
3  $k \leftarrow$  valor inicial para convergência;
4 for  $k \leftarrow 1$  to  $it$  do
5    $d[k] \leftarrow \mathbf{G}^T(\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k]);$ 
6    $\mu \leftarrow \frac{d^T[k]d[k]}{d^T[k](\mathbf{G}^T\mathbf{G})d[k]};$ 
7    $\hat{\mathbf{a}}[k + 1] \leftarrow \hat{\mathbf{a}}[k] + \mu d[k];$ 
8 end
```

se comparado com o **GD**. No entanto, para estes métodos, ainda pode ser necessário um número muito grande de iterações antes de localizar o ponto de mínimo com boa precisão. Em vista destas características, um outro método conhecido na literatura é generalizado para o caso particular de estimação de sinais no ambiente do TileCal. Este algoritmo é detalhado a seguir.

4.1.3 Gradiente Conjugado - GC

O método do Gradiente Conjugado [38] é um método iterativo para resolver um sistema específico de equações lineares,

$$\mathbf{A} \mathbf{x} = \mathbf{b}, \quad (4.14)$$

onde \mathbf{A} é uma matriz quadrada $p \times p$, simétrica e definida positiva, \mathbf{x} e \mathbf{b} são vetores coluna. Neste sentido, uma solução encontrada para resolver o sistema seria simplesmente considerar a matriz \mathbf{A} inversa, ou seja,

$$\mathbf{x} = \mathbf{A}^{-1} \mathbf{b}. \quad (4.15)$$

Entretanto, o cálculo da inversa retornaria ao problema apresentado pelo método

COF, dado que sua implementação para processamento *online* não é otimizada. Uma outra solução é propor \mathbf{x} como sendo uma combinação linear de uma base vetorial \mathbf{p}_i pré-determinada, como mostra a Equação (4.16), e encontrar uma solução para determinar α_i ,

$$\mathbf{x} = \sum_{i=1}^p \alpha_i \mathbf{p}_i. \quad (4.16)$$

Neste caso, substituindo a Equação (4.16) na Equação (4.14), encontramos a Equação (4.17) para \mathbf{A} simétrica e definida positiva,

$$\mathbf{b} = \mathbf{Ax} = \sum_{i=1}^p \alpha_i \mathbf{A} \mathbf{p}_i. \quad (4.17)$$

Multiplicando ambos os lados da Equação (4.14) por \mathbf{p}_k , então podemos reescrever a Equação (4.17), conforme apresenta a Equação (4.18),

$$\mathbf{p}_k^T \mathbf{b} = \mathbf{p}_k^T \sum_{i=1}^p \alpha_i \mathbf{A} \mathbf{p}_i. \quad (4.18)$$

Novamente, sabendo que \mathbf{A} é uma matriz simétrica e definida positiva, então podemos reescrever a Equação (4.18), conforme apresenta a Equação (4.19),

$$\mathbf{p}_k^T \mathbf{b} = \sum_{i=1}^p \alpha_i \mathbf{p}_k^T \mathbf{A} \mathbf{p}_i. \quad (4.19)$$

A Equação (4.19) pode ser extremamente simplificada se nos restringirmos a uma base tal que $\mathbf{p}_k^T \mathbf{A} \mathbf{p}_i = 0$, para $k \neq i$. Logo, através desse produto, os vetores \mathbf{p}_k e \mathbf{p}_i dizem-se ser A-ortogonais ou conjugados. Assim, podemos reescrever a Equação (4.19) conforme a Equação (4.20),

$$\mathbf{p}_k^T \mathbf{b} = \alpha_k \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k. \quad (4.20)$$

Finalmente, para que o ponderador α_k do sistema linear seja encontrado a partir da propriedade imposta, basta resolver a Equação (4.20) isolando-o, como mostra a Equação (4.21). Uma outra vantagem em restringir as componentes da base a serem A-ortogonais ($\mathbf{p}_k^T \mathbf{A} \mathbf{p}_i = 0$) é que os mesmos podem ser encontrados iterativamente usando de decomposição de Gram-Schmidt, como será visto adiante. Este fato reflete

uma estreita relação desse método com o **GDD**.

$$\alpha_k = \frac{\mathbf{p}_k^T \mathbf{b}}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} \quad (4.21)$$

4.1.3.1 Adaptação do GC para o COF

Em nosso caso particular, o sistema linear a ser otimizado é dado pela Equação (4.22),

$$\mathbf{G} \mathbf{a} = \mathbf{r}, \quad (4.22)$$

onde \mathbf{G} contém os sinais de referência relacionados aos BCs onde o empilhamento de sinais foram observados. Neste caso, a matriz \mathbf{G} não é quadrada nem simétrica já que o empilhamento de sinais ocorre aleatoriamente. Porém, multiplicando à direita ambos os lados da Equação (4.22) por \mathbf{G}^T , como mostra a Equação (4.23),

$$\mathbf{G}^T \mathbf{G} \mathbf{a} = \mathbf{G}^T \mathbf{r}, \quad (4.23)$$

e relacionando a Equação (4.23) com a Equação (4.14), temos que $\mathbf{A} = \mathbf{G}^T \mathbf{G}$ e $\mathbf{b} = \mathbf{G}^T \mathbf{r}$, de modo que a Equação (4.21) pode ser reescrita para este caso em particular, como mostra a Equação (4.24),

$$\alpha_k = \frac{\mathbf{p}_k^T \mathbf{G}^T \mathbf{r}}{\mathbf{p}_k^T \mathbf{G}^T \mathbf{G} \mathbf{p}_k}. \quad (4.24)$$

Assim, de acordo com a Equação (4.24), para usar o método **GC** no mesmo caso, devemos primeiro projetar o vetor de entrada \mathbf{r} na matriz \mathbf{G} e considerar $\mathbf{A} = \mathbf{G}^T \mathbf{G}$. Desta forma, conclui-se que os sistemas de equações lineares, (4.14) e (4.22), podem ser resolvidos através da Equação (4.16). Neste caso, reescrevendo a Equação (4.16), temos

$$\mathbf{x} = \alpha_1 \mathbf{p}_1 + \alpha_2 \mathbf{p}_2 + \dots + \alpha_k \mathbf{p}_p, \quad (4.25)$$

onde p é o número de sinais empilhados.

Pode-se traçar um paralelo entre esta equação e a Equação (4.6) do método **GD**. Para tal, nota-se que o resultado obtido com a Equação (4.6), após k iterações, é um valor muito próximo de $\hat{\mathbf{a}}$ dado pelo método **COF**, e pode ser escrito, por extenso,

como,

$$\hat{\mathbf{a}} \cong \hat{\mathbf{a}}[0] + \mu_1 \mathbf{d}_1 + \mu_2 \mathbf{d}_2 + \dots + \mu_k \mathbf{d}_k, \quad (4.26)$$

onde $\hat{\mathbf{a}}[0]$ é o valor inicial obtido pelo **DM** (3.33). Porém inicializando-se com um vetor nulo $\hat{\mathbf{a}}[0] = 0$ e comparando-se a Equação (4.25) e (4.26) termo a termo, chegamos as seguintes conclusões: i) Se forçarmos o resíduo \mathbf{d} a ser A-ortogonal a cada iteração, ou seja, $\mathbf{d}_k^T \mathbf{A} \mathbf{d}_{k+1}^T = 0$, o método convergirá perfeitamente para o **COF**, após p iterações, onde p é o número de sinais de referência contribuindo com o empilhamento de sinais (número de colunas de \mathbf{G}). ii) Nesse caso, os p fatores de convergência dinâmica μ_k são dados diretamente pela Equação (4.24). iii) Não se faz necessário utilizar o valor do **DM** como ponto de partida. iv) O processo de se obter direções A-ortogonais a cada iteração é computado com a decomposição de Gram-Smidt.

Uma vez que se provou ser possível encontrar a solução em p iterações, minimizando em cada passo ao longo das direções, só é necessário encontrar essas direções. Para tal, nessa primeira etapa, é possível gerar vetores conjugados a partir de vetores \mathbf{v}_k linearmente independentes através do processo de conjugação de Gram-Schmidt. Assim, fazendo \mathbf{p}_1 igual a \mathbf{d}_1 , podemos calcular as demais direções conjugadas \mathbf{p}_k através da combinação linear representada pela Equação (4.27),

$$\mathbf{p}_k = \mathbf{d}_k + \sum_{j=0}^{k-1} \beta_{k,j} \mathbf{p}_j \text{ para } k > 2, \quad (4.27)$$

onde $\beta_{k,j}$ é um escalar a ser determinado. Tal escalar tem a função de ajustar a direção de \mathbf{d}_k de modo a torná-lo conjugado às direções anteriores. Para isso, transponemos a Equação (4.27) e multiplicamos à direita por $\mathbf{A}\mathbf{p}_i$, em ambos os lados, para encontrarmos a Equação (4.28),

$$\mathbf{p}_k^T \mathbf{A} \mathbf{p}_i = \mathbf{d}_k^T \mathbf{A} \mathbf{p}_i + \sum_{j=0}^{k-1} \beta_{k,j} \mathbf{p}_j^T \mathbf{A} \mathbf{p}_i. \quad (4.28)$$

Impondo a condição $\mathbf{p}_k^T \mathbf{A} \mathbf{p}_i = 0$, para $i = k + 1$, podemos reescrever a Equação (4.28) como segue,

$$0 = \mathbf{d}_k^T \mathbf{A} \mathbf{p}_j + \beta_{k,j} \mathbf{p}_j^T \mathbf{A} \mathbf{p}_j. \quad (4.29)$$

Finalmente, para que $\beta_{k,j}$ seja determinado, basta isolarmos o termo na Equação (4.29). Desta forma, encontramos a Equação (4.30),

$$\beta_{k,j} = - \frac{\mathbf{d}_k^T \mathbf{A} \mathbf{p}_j}{\mathbf{p}_j^T \mathbf{A} \mathbf{p}_j}. \quad (4.30)$$

Até esta etapa, podemos sumarizar o processo de adaptação do algoritmo **GC** para a determinação das amplitudes dadas pelo **COF**, como; i) Inicializando com todas as amplitudes igual a zero, encontramos a direção da primeira propagação \mathbf{d}_1 e seu passo μ_1 de acordo com a Equação (4.12). ii) A direção da próxima iteração \mathbf{d}_k é calculada normalmente como no algoritmo **GD**, porém um ajuste fino nesta direção, usando a Equação (4.27) é necessário para garantir que a direção atual seja conjugada às direções anteriores em relação a $\mathbf{G}^T \mathbf{G}$. iii) Após p iterações, obtém-se o valor exato dado pelo **COF**.

A computação dos vetores conjugados dado pela Equação (4.27) pode ser simplificada usando algumas manipulações matemáticas. Visando otimizar a formulação matemática do algoritmo **GC**, tais simplificações são apresentadas a seguir.

4.1.3.2 Simplificação do método GC

O cálculo de $\beta_{k,j}$ necessita do armazenamento dos vetores \mathbf{d}_k das direções anteriores o que gera operações adicionais a serem executadas pelo presente método. No entanto, ao se gerar um conjunto de vetores conjugados, pode-se calcular um novo vetor \mathbf{p}_k usando apenas o vetor anterior \mathbf{p}_{k-1} , o que exige pouco armazenamento e cálculo. Desta forma, podemos reescrever a Equação (4.27), em que cada direção \mathbf{p}_k é escolhida para ser uma combinação linear da direção do gradiente \mathbf{d}_k , que é equivalente ao resíduo negativo $-\mathbf{e}_k$, e da direção anterior \mathbf{p}_{k-1} ,

$$\mathbf{p}_k = -\mathbf{e}_k + \beta_k \mathbf{p}_{k-1}, \quad (4.31)$$

onde \mathbf{p}_k e \mathbf{p}_{k-1} devem ser conjugados a respeito de \mathbf{A} ($\mathbf{G}^T \mathbf{G}$ no caso particular do **COF**), e o resíduo \mathbf{e}_k , equivalente a $\mathbf{e}_k = \mathbf{b} - \mathbf{Ax}_k$. Em seguida os mesmos procedimentos descritos em (4.28) e em (4.29) devem ser realizados, porém, inicialmente e diferentemente para este caso, devemos multiplicar a direita por $\mathbf{p}_{k-1}^T \mathbf{A}$, em ambos os lados, para encontrarmos a Equação (4.32),

$$\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_k = -\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{e}_k + \beta_k \mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1}. \quad (4.32)$$

Logo, impondo a condição $\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_k = 0$, podemos reescrever a Equação (4.32) como segue,

$$0 = -\mathbf{e}_k^T \mathbf{A} \mathbf{p}_{k-1} + \beta_k \mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1}. \quad (4.33)$$

Finalmente, determinamos β_k isolando o termo na Equação (4.33), como mostra a Equação (4.34),

$$\beta_k = \frac{\mathbf{e}_k^T \mathbf{A} \mathbf{p}_{k-1}}{\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1}}. \quad (4.34)$$

As formulações apresentadas através das equações (4.34) e (4.30) demonstram as propriedades essenciais do método **GC**, no entanto uma forma ainda mais eficiente do método pode ser determinada. Para tal, devemos inicialmente determinar a equação do resíduo \mathbf{e}_{k+1} para a próxima iteração, ou seja,

$$\mathbf{e}_{k+1} = \mathbf{b} - \mathbf{A} \mathbf{x}_{k+1}. \quad (4.35)$$

Vimos anteriormente que o resíduo para o passo atual é equivalente a $\mathbf{e}_k = \mathbf{b} - \mathbf{A} \mathbf{x}_k$. Então, ao isolarmos o vetor coluna \mathbf{b} na Equação (4.35), podemos reescrever a equação do resíduo \mathbf{e}_k ,

$$\mathbf{e}_k = \mathbf{e}_{k+1} + \mathbf{A} \mathbf{x}_{k+1} - \mathbf{A} \mathbf{x}_k, \quad (4.36)$$

para que em seguida a Equação (4.35) possa ser reescrita, como mostra a Equação (4.37),

$$\begin{aligned} \mathbf{e}_{k+1} &= \mathbf{e}_k - \mathbf{A} \mathbf{x}_{k+1} + \mathbf{A} \mathbf{x}_k \\ \mathbf{e}_{k+1} &= \mathbf{e}_k + \mathbf{A} (\mathbf{x}_k - \mathbf{x}_{k+1}). \end{aligned} \quad (4.37)$$

Ainda, considerando a iteração $k + 1$ podemos então reescrever a Equação (4.16) como segue,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{p}_k \\ \mathbf{x}_k - \mathbf{x}_{k+1} &= -\alpha_k \mathbf{p}_k, \end{aligned} \quad (4.38)$$

para que em seguida a Equação (4.35) esteja em termos de α_k e das direções conjugadas, substituindo a Equação (4.38) em (4.37), como apresenta a Equação (4.39),

$$\begin{aligned}\mathbf{e}_{k+1} &= \mathbf{e}_k - \alpha_k \mathbf{A} \mathbf{p}_k \\ \mathbf{A} \mathbf{p}_k &= \frac{\mathbf{e}_k - \mathbf{e}_{k+1}}{\alpha_k}.\end{aligned}\quad (4.39)$$

A partir da Equação (4.39), podemos retornar ao objetivo principal que é encontrar a forma mais eficiente do método **GC**. Para tal, analisa-se a Equação (4.34) em duas etapas. A primeira etapa visa a análise do numerador, como é apresentado: Sabendo que o numerador é dado por $\mathbf{e}_k^T \mathbf{A} \mathbf{p}_{k-1}$ e que este pode ser reescrito em termos de $\mathbf{e}_{k+1}^T \mathbf{A} \mathbf{p}$, então utilizando a Equação (4.39) podemos reescrever o numerador com sendo,

$$\begin{aligned}\mathbf{e}_{k+1}^T \mathbf{A} \mathbf{p}_k &= \frac{\mathbf{e}_{k+1}^T (\mathbf{e}_k - \mathbf{e}_{k+1})}{\alpha_k} \\ \mathbf{e}_{k+1}^T \mathbf{A} \mathbf{p}_k &= -\frac{\mathbf{e}_{k+1}^T \mathbf{e}_{k+1}}{\alpha_k}.\end{aligned}\quad (4.40)$$

Dado que \mathbf{e}_{k+1}^T e \mathbf{e}_k são ortogonais, então o numerador da Equação (4.34) é dado por $\mathbf{e}_{k+1}^T \mathbf{e}_{k+1}$.

Já a segunda etapa visa a análise do denominador. Neste caso, sabendo que o denominador é dado por $\mathbf{p}_{k-1}^T \mathbf{A} \mathbf{p}_{k-1}$ e que este pode ser reescrito em termos de $\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k$, e sabendo que $\mathbf{p}_k^T = \mathbf{e}_k + \beta_{k+1} \mathbf{p}_{k+1}$, então podemos reescrever o denominador com sendo,

$$\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k = (\mathbf{e}_k + \beta_{k+1} \mathbf{p}_{k+1}) \mathbf{A} \mathbf{p}_k. \quad (4.41)$$

Em seguida, substituindo a Equação (4.39) em (4.41) temos que,

$$\begin{aligned}\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k &= \frac{\mathbf{e}_k^T (\mathbf{e}_{k+1} - \mathbf{e}_{k-1})}{\alpha_k} \\ \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k &= \frac{\mathbf{e}_k^T \mathbf{e}_k}{\alpha_k}.\end{aligned}\quad (4.42)$$

Finalmente, após analisado o numerador e o denominador da Equação (4.34), determinamos a forma mais eficiente para o cálculo de β do presente método, como segue,

$$\beta_k = \frac{\mathbf{e}_{k+1}^T \mathbf{e}_{k+1}}{\mathbf{e}_k^T \mathbf{e}_k}. \quad (4.43)$$

A partir dessa análise, podemos também encontrar a forma mais eficiente de α apenas isolando o termo na Equação (4.42), como é apresentado na Equação (4.44),

$$\alpha_k = \frac{\mathbf{e}_k^T \mathbf{e}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}. \quad (4.44)$$

Determinado a forma mais eficiente de α e β partindo do resíduo para o passo atual, podemos finalmente reescrever as Equações (4.35) e (4.42) para o nosso caso particular. Inicialmente, reescrevendo a Equação (4.35) nos termos da Equação 4.26, encontramos a Equação (4.45) do resíduo para um novo passo,

$$\begin{aligned} \mathbf{e}_{k+1} &= \mathbf{b} - \mathbf{A}\mathbf{x}_{k+1} \\ \mathbf{e}_{k+1} &= \mathbf{G}^T \mathbf{r} - \mathbf{G}^T \mathbf{G} \mathbf{x}_{k+1} \\ \mathbf{e}_{k+1} &= \mathbf{G}^T (\mathbf{r} - \mathbf{G} \mathbf{a}_{k+1}), \end{aligned} \quad (4.45)$$

em que o resíduo inicial agora é dado por ($\mathbf{e} = \mathbf{G}^T(\mathbf{r} - \mathbf{G}\mathbf{a})$). Finalmente reescrevendo a Equação (4.42), encontramos a Equação (4.46) para o cálculo de α .

$$\begin{aligned} \mu_k &= \frac{\mathbf{e}_k^T \mathbf{e}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} \\ \mu_k &= \frac{\mathbf{e}_k^T \mathbf{e}_k}{\mathbf{p}_k^T \mathbf{G}^T \mathbf{G} \mathbf{p}_k} \end{aligned} \quad (4.46)$$

O algoritmo funcional do método **GC** é apresentado no quadro a seguir. Inicialmente defini-se o resíduo \mathbf{e} que também pode ser visto como a direção máxima de descida \mathbf{p} . Em seguida, são computados os parâmetros para convergência do método. O primeiro parâmetro a ser encontrado é a taxa de evolução μ , que neste método é compreendido como o comprimento do passo. O melhor valor de μ é encontrado através do processo de minimização ao longo da direção \mathbf{p} . Logo, podemos calcular o primeiro ponto em que o valor estimado esteja suficientemente próximo do sinal real. O resíduo \mathbf{e} é então atualizado para o cálculo da direção A-ortogonal. Neste caso, a direção resultante é linearmente independente da direção anterior. Todo o processo é repetido até que a p -ésima iteração seja atingida.

O algoritmo **GC** converge com poucas iterações sendo capaz de encontrar a am-

Algorithm 3: ALGORITMO GRADIENTE CONJUGADO

Data:

- \mathbf{r} : Sinal de entrada digitalizado
- \mathbf{G} : Sinais de referência

Result:

- $\hat{\mathbf{a}}(k+1)$: Vetor de Amp. Estimadas

```
1  $\mu \leftarrow$  taxa de evolução;
2  $it \leftarrow$  máxima iteração;
3  $k \leftarrow$  valor inicial para convergência;
4  $\mathbf{e} \leftarrow \mathbf{G}^T(\mathbf{r} - \mathbf{G}\mathbf{a})$ ;
5  $\mathbf{p} \leftarrow \mathbf{e}$ ;
6 for  $k \leftarrow 1$  to  $it$  do
7    $\mu[k] \leftarrow \frac{\mathbf{e}^T[k]\mathbf{e}[k]}{\mathbf{p}^T[k]\mathbf{G}^T\mathbf{G}\mathbf{p}[k]}$ ;
8    $\hat{\mathbf{a}}[k+1] \leftarrow \hat{\mathbf{a}}[k] + \mu\mathbf{p}[k]$ ;
9    $\mathbf{e}[k+1] \leftarrow \mathbf{G}^T(\mathbf{r} - \mathbf{G}\hat{\mathbf{a}}[k+1])$ ;
10   $\beta[k+1] \leftarrow \frac{\mathbf{e}^T[k+1]\mathbf{e}[k+1]}{\mathbf{e}^T[k]\mathbf{e}[k]}$ ;
11   $\hat{\mathbf{p}}[k+1] \leftarrow -\mathbf{e}[k+1] + \beta[k+1]\mathbf{p}[k]$ ;
12 end
```

plitude de sinais com elevada precisão, superando os demais algoritmos apresentados para otimização do método **COF**. Esse fato pode ser comprovado neste capítulo, onde chegamos as seguintes conclusões: i) Na Seção 4.1.1, foi possível comprovar matematicamente que o algoritmo **GD** evita a inversão de matrizes resultando em simples operações de soma e produto, o que facilita sua implementação em dispositivos FPGA para processamento *on-line*. No entanto, o método **GD** necessita de um número elevado de iterações e é dependente de uma boa escolha do ponto de partida ($\mathbf{a}[0]$) para convergência. Além disso, se faz necessário uma análise do critério de parada para maior precisão com o menor número de interações. ii) Na Seção 4.1.2, comprova-se através das formulações matemáticas que o algoritmo **GDD** localiza o ponto de mínimo com menor número iterações se comparado com o **GD**, haja visto que a taxa de evolução é calculada adaptativamente. Neste caso, o **GDD** localiza o ponto de mínimo com maior precisão reduzindo assim o número de iterações a serem executadas. Do ponto de vista de implementação, o método **GDD** possui complexidade computacional superior ao **GD** pois executa uma operação de divisão entre escalares. iii) Na Seção (4.1.3), foi possível comprovar que o algoritmo **GC**, assim como os demais, evita a inversão de matrizes do método **COF**. Matematicamente, vimos que o **GC** converge para um número finito de iterações (p iterações), que correspondem exatamente a quantidades de sinais empilhados dentro de uma

mesma janela. Em sua forma otimizada, o método necessita apenas da direção anterior para o cálculo da nova direção. Além disso, o **GC** não se utiliza do resultado do **DM** como ponto de partida para execução. Do ponto de vista de implementação, o presente método possui complexidade computacional superior aos métodos **GD** e **GDD**, pois executa duas operações de divisão entre escalares. Maiores detalhes sobre análise de complexidades, serão vistos no Capítulo 5.

A seguir, os métodos propostos são comparados utilizando um ambiente de simulação no TileCal em alta luminosidade.

4.2 Simulação dos Métodos Iterativos

Nesta seção, são apresentadas as análises para as propostas de otimização do **COF** através de métodos iterativos. Inicialmente, é apresentado o ambiente de simulação da resposta do sistema de calorimetria após as colisões ocorridas no **LHC**. Em seguida, são apresentados os resultados das simulações dos métodos Gradiente Descendente, Gradiente Descendente com Taxa de Convergência Dinâmica e, por fim, são apresentadas as simulações para o Gradiente Cunjugado.

4.2.1 Ambiente de Simulação

Os sinais de entrada que simulam a resposta do sistema de calorimetria foram gerados em ambiente MatLab a fim de permitir a comparação dos métodos iterativos com o **COF**. Esta comparação, objetivo principal deste capítulo, se dá fundamentalmente através do erro médio quadrático, o **MSE** (do inglês, *Mean Squared Error*) entre os métodos propostos e o **COF**, uma vez que o **COF** tem apresentado excelentes resultados dentro do processo de estimativa da amplitude de sinais no TileCal, operando em condições de alta luminosidade.

Inicialmente foi gerado um vetor linha com aproximadamente 100.000 amostras, onde cada uma das amostras é equivalente a energia absorvida para uma determinada célula do calorímetro, após o evento de colisão. Neste caso, o vetor de amostras é uma sequência de colisões (*Bunch Crossing*) para um canal do TileCal.

A sequência de colisões é preenchida aleatoriamente para um determinado fator de ocupância, que representa a porcentagem de colisões que realmente atingiram a célula em questão. A maioria dos prótons não colidem frontalmente, gerando somente um desvio da trajetória horizontal (eixo z). Portanto, regiões de $|\eta|$ alto tendem a ter uma maior ocupância. Desta forma, espera-se uma maior ocupância nas células de barril estendido. Estima-se em média, uma ocupância de 10% e 20% para o TileCal [39], na região do barril e barril estendido, respectivamente. Sendo assim, para o fator de ocupância de 10%, as comparações dos métodos propostos

com o COF estão relacionadas há eventos de colisão em células do barril. Já para o fator de ocupância de 20%, as comparações estão relacionadas há eventos de colisão para células do barril estendido. Em amostras do vetor com valores não-nulos, são propostos sinais de referência do TileCal com uma distribuição de amplitude dada por uma exponencial e com o valor médio igual a 30 contagens de ADC para simular a energia depositada. A utilização desta distribuição apresenta maior probabilidade de ocorrência de deposição de eventos de menor energia, assim como a distribuição da energia de um sistema de calorimetria. Já o valor médio de 30 contagens de ADC é utilizado para se obter uma largura de distribuição mais realista quando comparada à distribuição da energia do ruído de sinais empilhados do TileCal.

Por fim, a sequência de colisões é então dividida em janelas de 7 amostras, gerando vetores de entrada para a simulação. Uma vez que trata-se de um processo aleatório, em algumas situações o vetor de entrada não irá conter sinal ou sobreposição. Em outros casos, há existência de sinal no BC central com influência do empilhamento de sinais de seus BCs adjacentes imediatos. Após determinado os vetores de entrada um ruído WG de 1 contagem de ADC é adicionado, representando o ruído eletrônico.

A seções a seguir apresentam as análises dos métodos iterativos Gradiente Descendente, Gradiente Descendente Adaptativo e Gradiente Conjugado.

4.2.2 Simulação do método GD

No presente método, busca-se inicialmente determinar a faixa de valores de μ que permite a convergência do algoritmo para a estimativa da amplitude de sinais sobrepostos.

No processo de identificação de sinais empilhados, inicialmente calcula-se o **DM** para identificar a existência de sinal em um determinado BC. O valor utilizado para o limiar é o mesmo definido no método **COF**. Após obter as informações do **DM** o algoritmo **COF** é executado para comparação. Através destes resultados, o **MSE** (do inglês, *Mean Squared Error*) entre o **GD** e **COF** é calculado. Tal cálculo é utilizado para definir o quanto longe a estimativa do **GD** estará da estimativa **COF**, em porcentagem. Para tal, o algoritmo **GD** é então executado, tendo como valores iniciais de convergência os dados calculados pelo **DM**. Durante o processo iterativo, os resultados do presente método são encontrados. Por fim, o desvio entre o **GD** e **COF** é calculado, em porcentagem, tendo como base de cálculo o **MSE** entre o **DM** e **COF**.

A Figura (4.1) mostra a simulação do **GD** considerando 20 valores para μ dentro faixa $0.01 < \mu < 0.6$. Para o número de iterações também foi considerado 20 valores dentro da faixa de $0 < \text{iterações} < 2000$. A figura apresenta o resultado da

simulação para uma ocupância de 10% de modo a identificar os valores de μ que divergem. O eixo na vertical demonstra o **MSE** entre o **GD** e o **COF**.

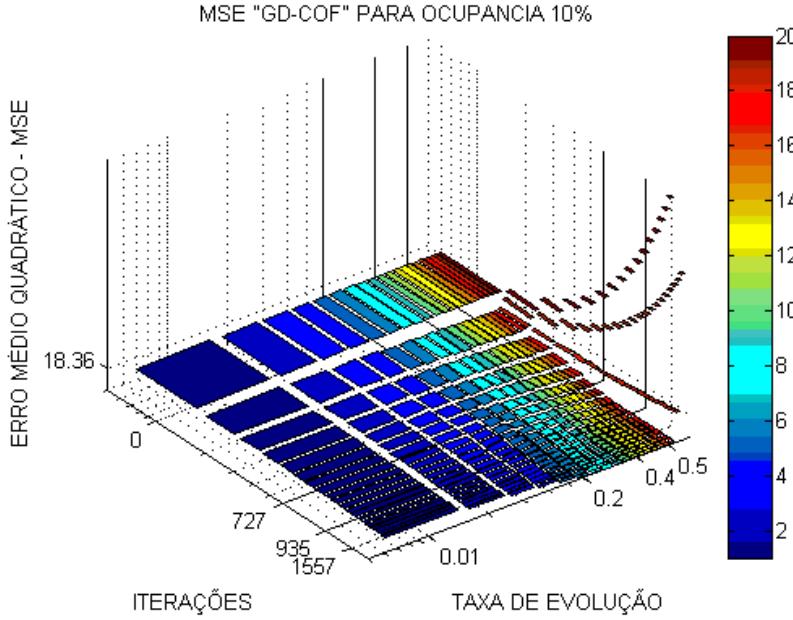


Figura 4.1: Identificação dos valores de μ em que há divergência.

O resultado dessa simulação indica que o algoritmo **GD** diverge para valores de μ acima de 0.54. A partir desta análise, inicialmente considera-se que o algoritmo será executado para μ dentro da faixa de $0.01 < \mu < 0.5$, respeitando a margem de convergência do algoritmo. Neste caso, considerando a nova faixa para μ , obtém-se o gráfico de convergência do algoritmo, como mostra a Figura (4.2).

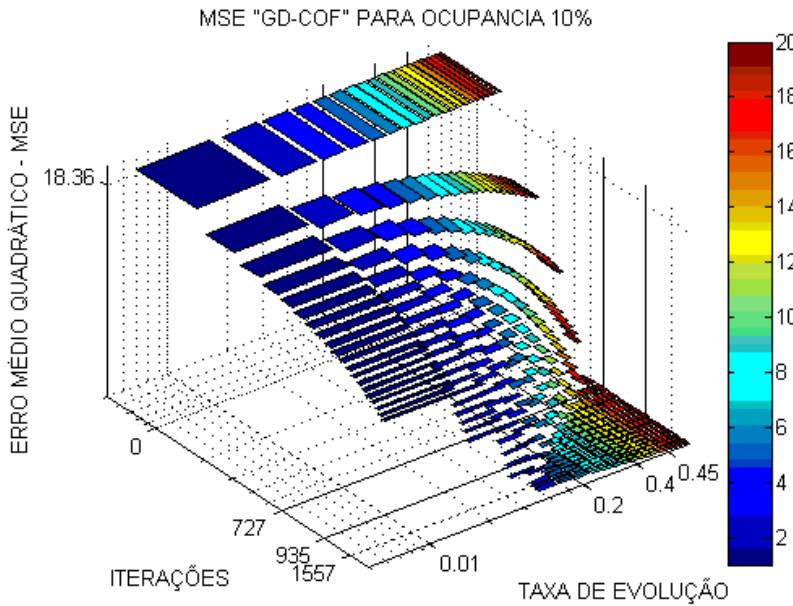


Figura 4.2: Convergência do algoritmo **GD**.

A partir do resultado obtido através da Figura (4.2), pode-se notar que, quando o valor do número de iterações é igual a zero, os valores de **MSE** são constantes, independentemente da variação de μ , como esperado. Este fato ocorre quando o método Gradiente Descendente não executa nenhuma iteração. Sendo assim, tendo em vista que o mesmo inicializa com valores calculados pelo **DM**, então o seu **MSE** será idêntico ao **MSE** calculado entre o **DM** e o **COF**. Percebe-se também que, a medida em que o número de iterações aumenta com o aumento da taxa de evolução, o **MSE** entre o **GD** e o **COF** diminui, como era esperado.

Determinado a faixa inicial de valores para μ , busca-se então, encontrar o número mínimo de iterações em que o desvio entre o **COF** e o **GD** esteja próximo de valores nulos. A Figura (4.3) mostra o resultado da simulação considerando 20 valores dentro das faixas de $0 < \text{iterações} < 20$ e $0.01 < \mu < 0.5$. Neste gráfico, para o número de iterações igual a zero, o desvio entre o **COF** e o **GD** é de 100%, o que equivale ao erro entre o **DM** e o **COF**. Ou seja, novamente, podemos notar que quando o valor do número de iterações é igual a zero, e independentemente da variação de μ , o desvio entre o **COF** e o **GD** é máximo, já que o **GD** assume como ponto de partida o valor calculado pelo **DM**. Quanto maior o desvio, maior será a distância entre a estimativa do **GD** e a estimativa do **COF**. Quanto menor o desvio, o **GD** estará estimando a amplitude dos sinais com valores próximos aos valores calculados pelo **COF**.

Nas figuras em que são apresentados os desvios, o eixo na vertical demonstra o **MSE** entre o **(DM – COF)** sobre, o **(DM – GD)**.

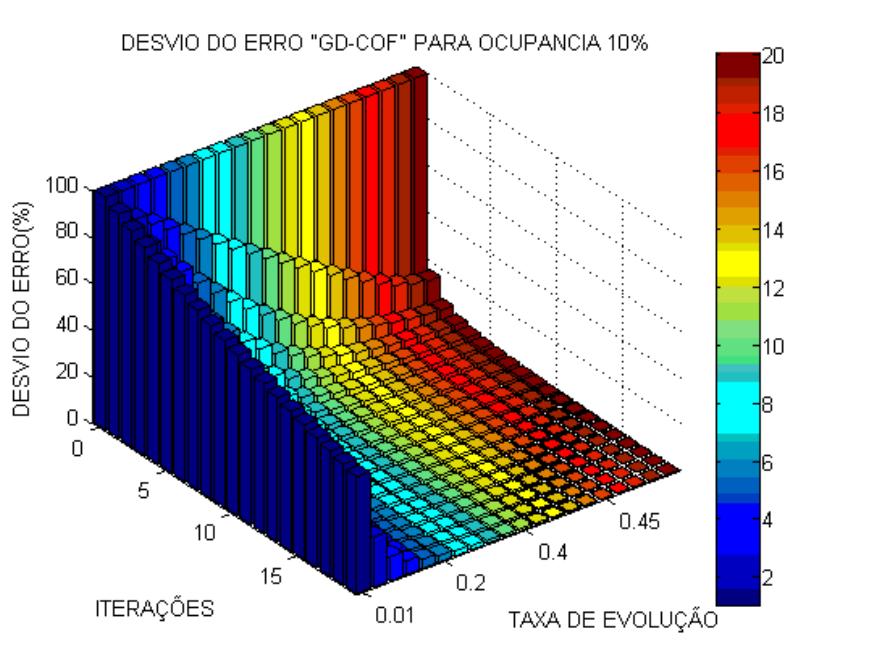


Figura 4.3: Desvio do Erro para ($0 < \text{it} < 20$, $0.01 < \mu < 0.5$ e ocupância de 10%).

Analizando a menor taxa de evolução ($\mu = 0.01$) para toda a faixa do número de iteração, é possível perceber que o algoritmo não atinge um desvio próximo de valores nulos, mesmo com a diminuição do desvio ao longo de toda a faixa. No entanto, a medida em que o número de iterações aumenta, com o aumento de μ , o desvio diminui. Neste caso, se considerarmos os valores de μ acima de 0.2 e o número de iteração acima de 20 iterações, encontraremos a faixa de valores para um desvio próximo de valores nulos. Sendo assim, buscando a melhor visualização da Figura (4.3), uma nova simulação é realizada (ver Figura (4.4)), considerando 20 valores entre faixas $20 < \text{iterações} < 100$ e $0.26 < \mu < 0.5$.

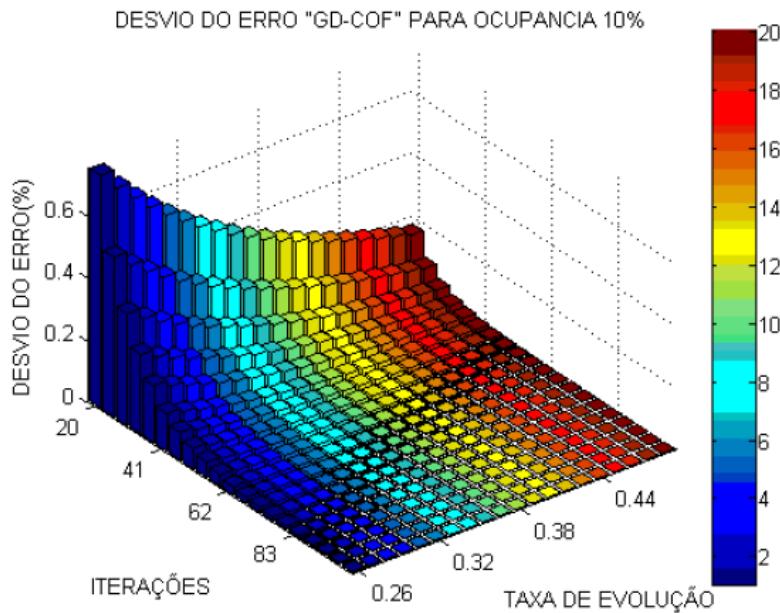


Figura 4.4: Desvio do Erro para ($20 < \text{it} < 100$, $0.26 < \mu < 0.5$ e ocupância de 10%).

Através do resultado apresentado pela Figura (4.4) é possível verificar que com 20 iterações obtém-se um desvio inferior a 1%, considerando a menor taxa de evolução (0,26) e ocupância de 10%. Seguindo esta mesma análise, porém agora para uma simulação considerando uma ocupância de 20% (Figura (4.5)), são necessárias no mínimo 32 iterações, para se obter desvio inferior a 1% no menor valor de μ . Ou seja, para uma população aumentando de 10.000 para 20.000 eventos de colisão em 100.000 amostras, o número de iterações aumenta de 20 para 32, se for considerada a menor taxa de evolução ($\mu = 0,26$). A medida em que o número de iterações aumenta, com o aumento da taxa de evolução, o desvio aproxima-se de valores nulos.

Como visto anteriormente, um detalhe importante para análise deste algoritmo é encontrar um critério de parada, onde μ seja suficientemente pequeno para maior precisão e suficientemente grande para viabilizar a implementação *online* (necessi-

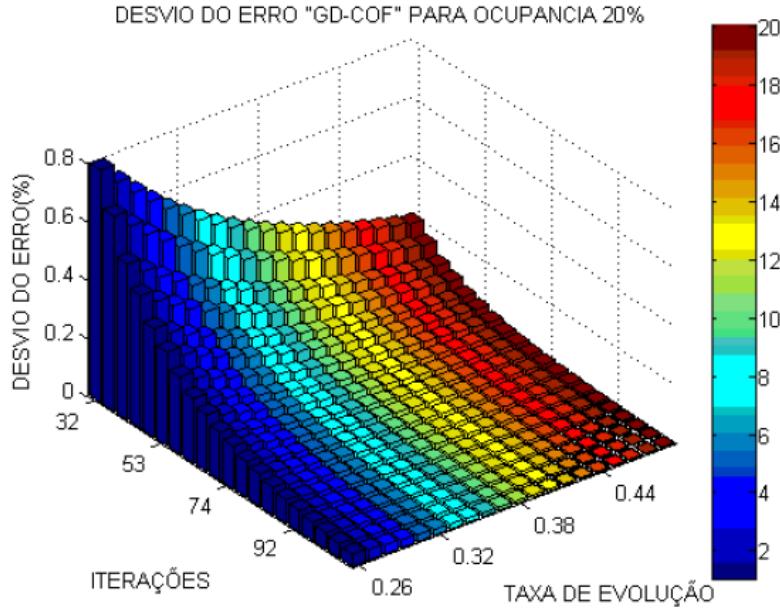


Figura 4.5: Desvio do Erro para ($20 < \text{it} < 100$, $0.26 < \mu < 0.5$ e ocupância de 20%).

dade de poucas iterações para convergência). No método **GD**, uma aproximação aos valores nulos do desvio acarreta no aumento do número de iterações e consequentemente há uma necessidade de um valor maior para μ . Desta forma, com o intuito de determinar o número mínimo de iterações e a faixa de μ com valores suficientemente pequenos para maior precisão, uma nova simulação é realizada. Para a análise, considera-se a faixa de valores entre $0.4 < \mu < 0.5$, como mostra a Figuras (4.6) e (4.7).

Através dos resultados apresentados pelas Figuras (4.6) e (4.7), considerando os eventos de colisão em células do barril e barril estendido, pode-se notar que são necessárias 13 e 20 iterações, para se obter desvio entre o **GD** e o **COF** inferior a 1%, considerando μ igual a 0.4. Este valor para μ e os valores para o número de iterações (13 e 20) serão utilizados para implementação *online* deste algoritmo. No entanto, como o critério de parada e a taxa de evolução são fixados, valores estes, resultados de simulação obtidos na média, pode ocorrer para uma ocupância de 10%, por exemplo, de um evento necessitar que o algoritmo encontre o ponto de mínimo com um valor superior a 13 iterações. Sendo assim, a seguir, são apresentadas análises do método Gradiente Descendente com Taxa de Convergência Dinâmica.

4.2.3 Simulação do método GDD

Diferentemente do método anterior, para o presente método busca-se, apenas, o menor valor para o número de iteração, já que para este algoritmo os valores

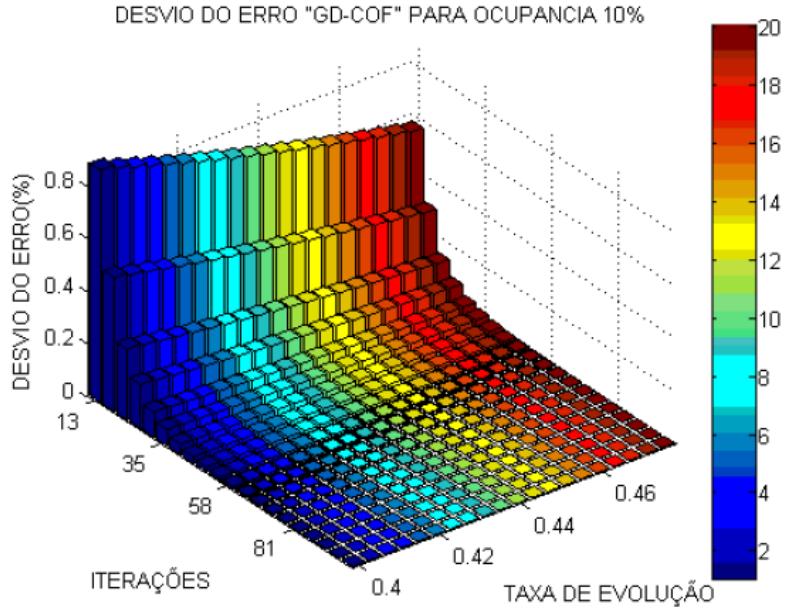


Figura 4.6: Desvio do Erro para ($13 < it < 100$, $0.4 < \mu < 0.5$ e ocupância de 10%).

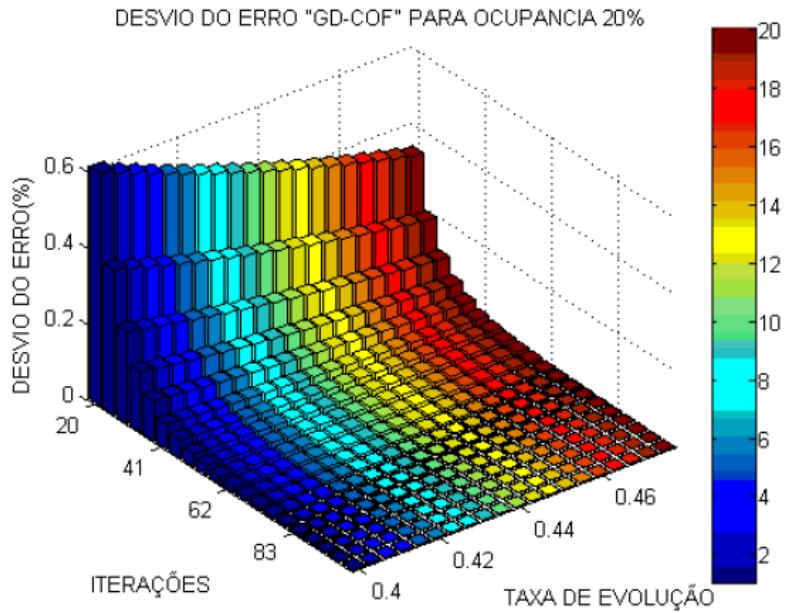


Figura 4.7: Desvio do Erro para ($20 < it < 100$, $0.4 < \mu < 0.5$ e ocupância de 20%).

de μ são determinados adaptativamente. No processo de simulação, manteve-se as considerações relacionadas ao desvio inferior a 1%, uma vez que conhecemos a priori o algoritmo mais eficiente. A ocupância de 10% e 20% também são consideradas nessa seção.

A Figura (4.8) demonstra os resultados obtidos para eventos de colisão para as células do barril e barril estendido e a comparação do presente algoritmo com o

COF e com o método Gradiente Descendente.

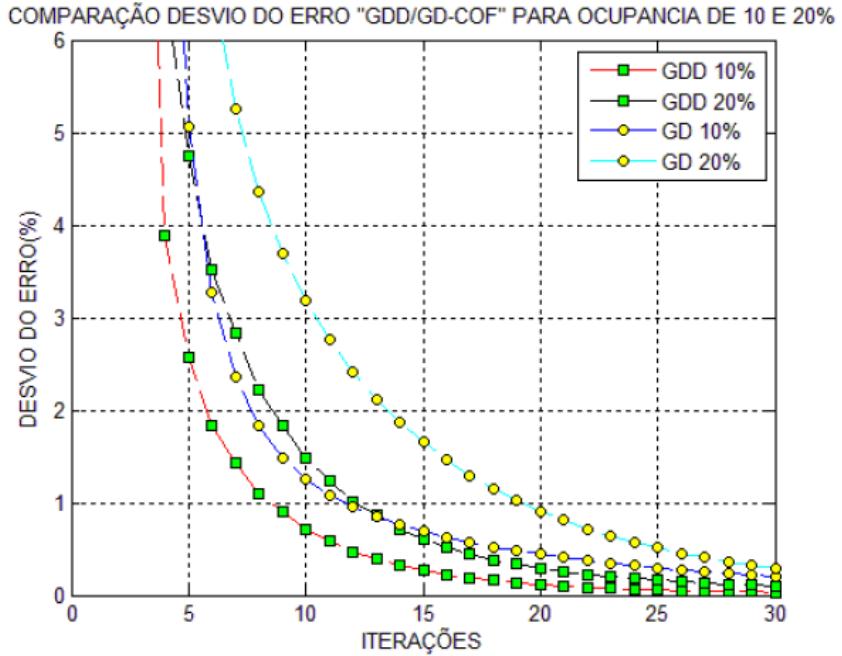


Figura 4.8: Comparação entre os métodos **GD**, **GDD** e **COF** e a identificação do valor mínimo de iterações para o método **GDD**, considerando o desvio de 1% e ocupância de 10 e 20%.

De posse dos resultados, é possível concluir que o presente método necessita, em média, de 9 e 13 iterações, no mínimo, para um desvio inferior a 1% do **COF** padrão, com ocupância de 10% e 20%, respectivamente. Este resultado comprova uma eficiência superior do **GDD** quando o método é comparado ao algoritmo **GD**, já que para este último são necessárias 13 e 20 iterações, com ocupância de 10% e 20%, respectivamente, e uma taxa de evolução fixa $\mu = 0,4$.

Os resultado do método **GDD** também não podem ser generalizados para todo e qualquer evento, pois, assim como o método **GD**, para o presente algoritmo também pode ser necessário um número infinito de iterações antes de se localizar o ponto de mínimo. Diante as características, o método **GC** é avaliado, como segue.

4.2.4 Simulação do método **GC**

Para o método Gradiente Conjugado, também foi gerado um vetor linha com aproximadamente 100.000 amostras, considerando as mesmas ocupâncias de 10% ou 20%. Diferentemente dos outros métodos, para o **GC**, busca-se apenas confirmar que, o algoritmo irá convergir para o **COF** com um número finito de iterações. Em nosso caso particular, este número finito tem seu valor máximo dado pela quantidade máxima de sinais empilhados detectados pelo **DM** na janela de 150 ns ($p=7$). A Figura (4.9) apresenta o **MSE** para um desvio inferior a 1% para ocupância de

10% e 20%, respectivamente, e sua comparação do presente método com os demais algoritmos.

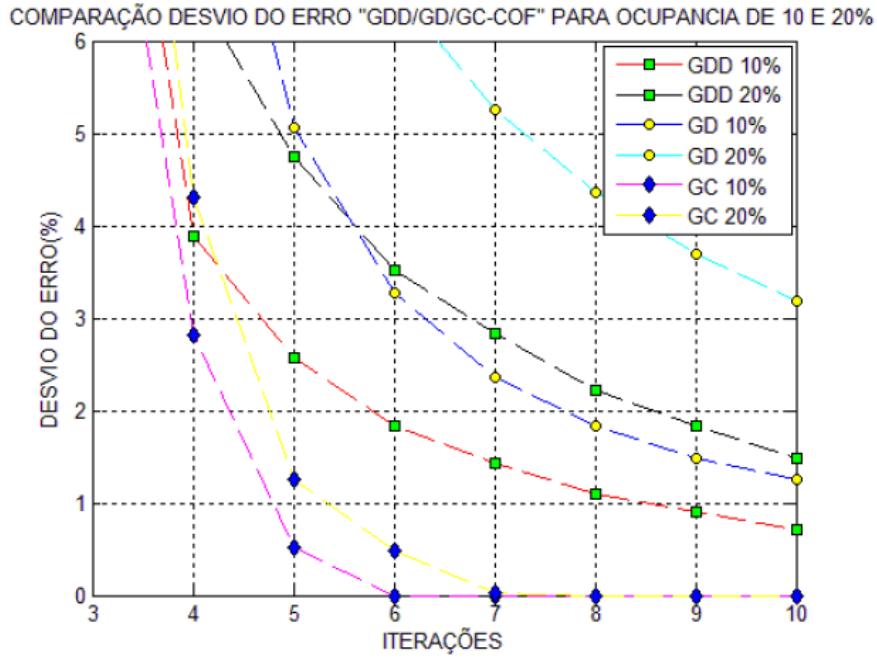


Figura 4.9: Comparaçāo entre os mētods **GD**, **GDD**, **GC** e **COF** e a identificaçāo do valor mēnimo de iteraçōes para o mētod **GC**, considerando o desvio de 1% e ocupância de 10 e 20%.

Através destas figuras, pode-se comprovar que o **MSE** diminui com o aumento do númer de iteraçōes, por mēr diferentemente dos outros algoritmos, o **GC** converge com um mēnimo de 6 iteraçōes (ocupância de 10%) atingindo um desvio nulo. Para ocupância de 20% sāo necessárias 6 iteraçōes, no mēnimo, por mēr o desvio nulo so é obtido para um total de 7 iteraçōes, no māximo, como era esperado.

Para a análise do desvio entre o valor real da amplitude de um sinal e o valor estimado pelos mētods propostos, uma nova simulaçāo foi realizada. Para tal, foi considerado a existênci de sinal no BC central com influênci dos BCs a_{-2} e a_{+3} . Desta forma, para um vetor de amplitudes $\mathbf{a} = [a_{-3} \ a_{-2} \ a_{-1} \ a_0 \ a_{+1} \ a_{+2} \ a_{+3}]$ foram atribuídos os valoress $[0 \ 29 \ 0 \ 25 \ 0 \ 0 \ 27]$. Apôs o empilhamento dos sinais e a adiçō de ruído, os valoress de simulaçāo para as amostras de entrada \mathbf{r} sāo obtidos, $[14.23 \ 29 \ 28.69 \ 30.02 \ 17.37 \ 16.89 \ 25.38]$. No processo de identificaçāo de sinal empilhados o **DM** é calculado, resultando em um vetor com valoress $[0 \ 23.78 \ 0 \ 29.18 \ 0 \ 0 \ 22.75]$. Para um limiar de 5 ADC *counts*, o **DM** identifica corretamente o sinal no BC central e nos BCs a_{-2} e a_{+3} . O resultado do **DM** é entanto utilizad como ponto de partida apenas para o mētod **GD**. No entanto, é igualmente utilizad pelos tréis mētods para a determinaçāo das dimensões da matriz **G** (condiçāo para estimaçāo da amplitude de forma ótima). Sendo assim, foi

determinado para μ , o valor igual a 0.4. Esse valor, é resultado do melhor caso para as simulações em 4.2.2, pois, em média, o algoritmo pode convergir para o ponto de mínimo realizando 13 iterações.

As figuras a seguir demonstram os resultados para comparação dos métodos apresentados em 4.1.1, 4.1.1 e 4.1.3. A Figura (4.10) demonstra o resultado da simulação onde são apresentadas as curvas de convergência para os três métodos. Nesta visão geral, é possível perceber que o algoritmo **GC** converge para o ponto de mínimo com apenas três iterações, como era esperado, já que foi considerado a existência de sinal em a_{-2} , a_0 e a_{+3} . Ou seja, para $p=3$ o algoritmo foi capaz de convergir sem desvio para o valor do **COF**.

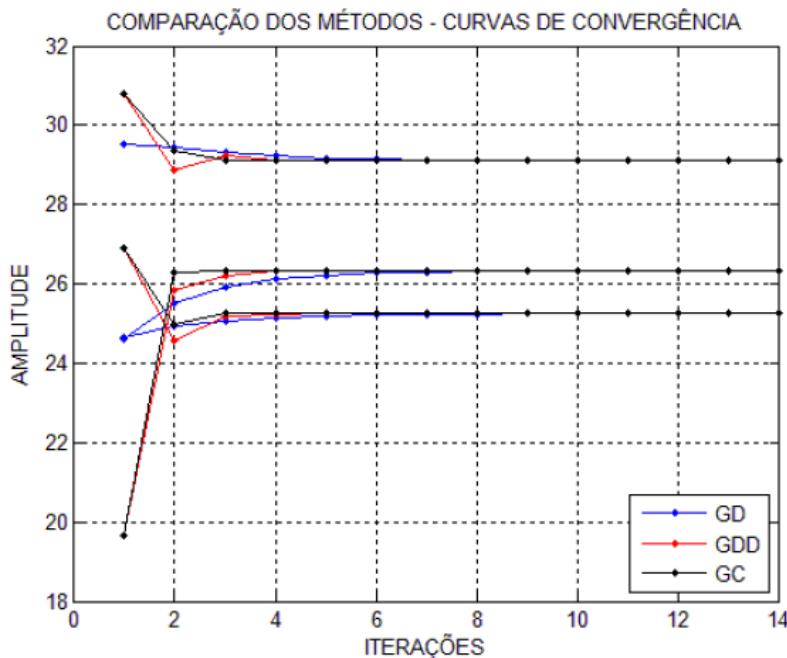


Figura 4.10: Curva de Convergência dos Métodos GD, GDD e GC

As Figuras (4.11), (4.12) e (4.13) tratam-se de um visão ampliada da Figura (4.10) que demonstra a curva de convergência para cada BC (a_{-2} , a_0 e a_{+3}), respectivamente. Nestas simulações é possível notar que os algoritmos **GDD** e **GC** partem de forma equivalente no ponto inicial de convergência, diferentemente do **GD**. Tal diferença, caracteriza o **GD** como o algoritmo de melhor eficiência, apenas, no ponto de partida, pois se utiliza dos valores calculados pelo **DM**. Entretanto, por ser menos eficiente ao longo da convergência, o **GD** é rapidamente superado.

Através da Figura (4.11) é possível notar que a convergência dos algoritmos atinge os pontos de mínimo em valores próximos a 29 ADC *counts*, sendo este o valor real definido para o BC a_{-2} . A mesma análise pode ser realizada para as Figuras (4.12) e (4.13), comparando os pontos de mínimo com os valores de **a**. Neste caso, para os BCs a_0 e a_{+3} , a curva de convergência dos algoritmos atinge os

pontos de mínimo em valores próximos a 25 e a 27 ADCs *counts*, respectivamente.

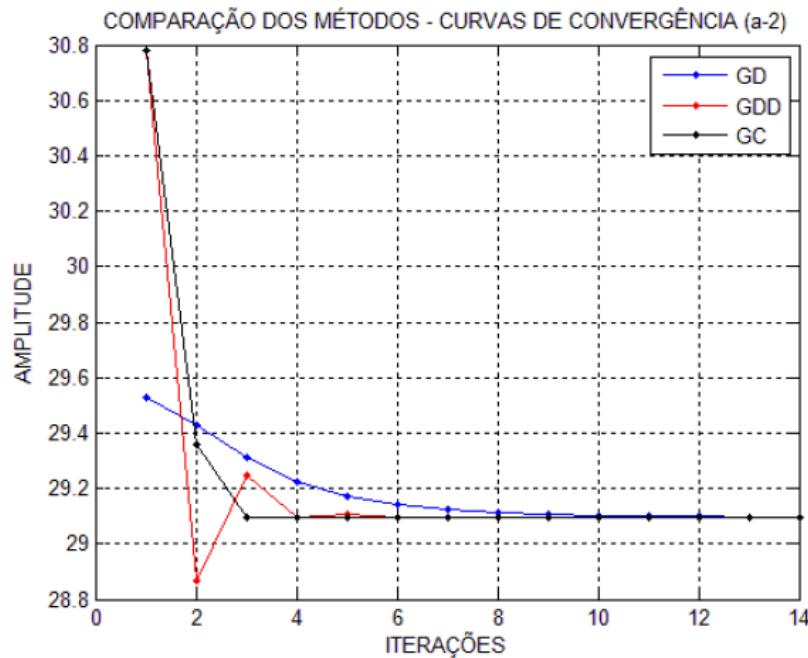


Figura 4.11: Curva de Convergência dos Métodos para BC-2 ($a_{-2} = 29$)

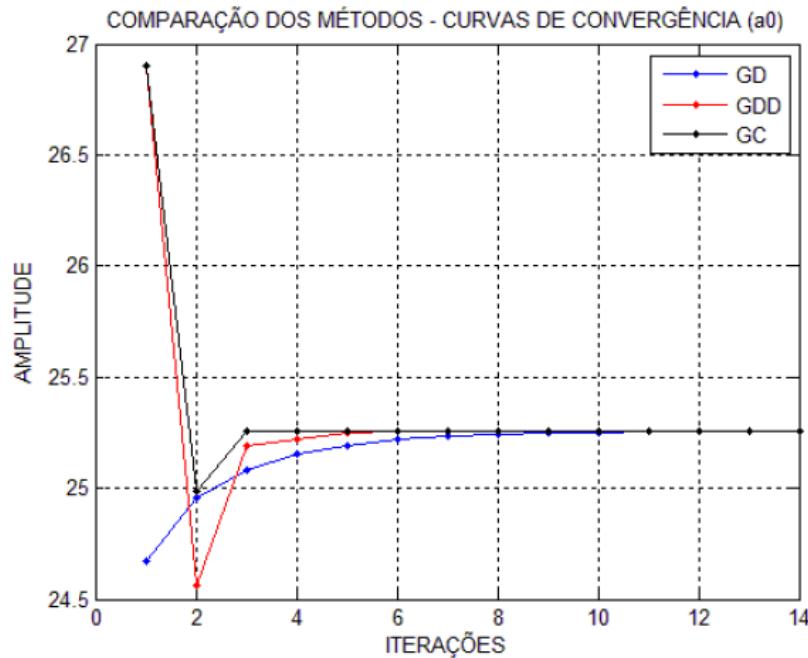


Figura 4.12: Curva de Convergência dos Métodos para BC central ($a_0 = 25$)

A análise do desvio entre o valor real da amplitude de um sinal e o valor estimado pelos métodos propostos podem ser melhor avaliados através da Tabela (4.2.4). Ela demonstra o comparativo entre resultados extraídos dos algoritmos **GD**, **GDD** e

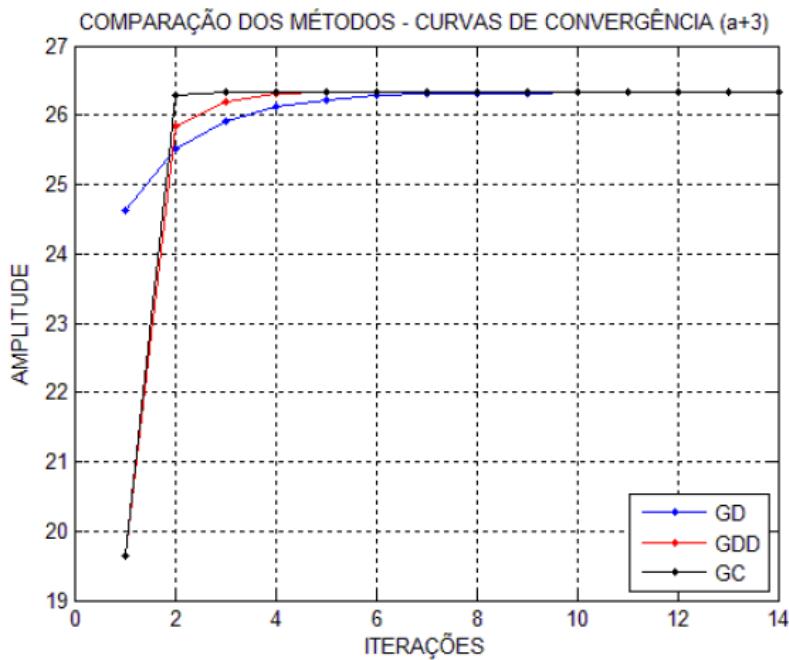


Figura 4.13: Curva de Convergência dos Métodos para BC+3 ($a_{+3} = 27$)

GC com os valores de amplitudes reais, determinados para a simulação, bem como, com os resultados extraídos pelo **COF**.

	a_{-2}	a_0	a_{+3}	Iterações
REAL	25.00	29.00	27.00	-
COF	25.25	29.09	26.32	-
GD	24.95	29.52	24.63	3
GDD	25.19	29.24	26.20	3
GC	25.25	29.09	26.32	3

Tabela 4.1: Comparativo entre os Métodos (**COF**, **GD**, **GDD** e **GC**) a partir da amplitude real de sinais sobrepostos.

Os resultados da Tabela (4.2.4) confirmam que o método **GC** tem eficiência superior aos demais métodos, já que este determina com apenas 3 iterações a amplitude dos 3 sinais sobrepostos definidos para a simulação. Pode-se notar, também, que o método **GC** encontra, no ponto de mínimo, valores idênticos ao método COF, diferentemente dos métodos **GD** e **GDD**, que com apenas 3 iterações, não são capazes de estimar os sinais com a mesma precisão. Assim, pode-se concluir que os métodos **GD** e **GDD** necessitam de um número maior de iterações para obter uma eficiência similar ao **GC**, quando estes são comparados com o método **COF**, objetivo final deste capítulo.

O método **GC** possui eficiência superior aos demais métodos como pode ser verificado, no entanto este apresenta custo computacional superior aos demais algoritmos. Dentre os métodos, o **GD** é o algoritmo que apresenta menor custo com-

putacional. A partir das simulações deste capítulo podemos inferir que apesar dos demais métodos não apresentarem eficiência similar ao método **GC**, eles apresentam um desvio muito pequeno entre o valor real e o valor estimado, conforme os dados da Tabela (4.2.4). Neste caso, sabendo que o objetivo final do presente trabalho é a implementação em FPGA, considerando uma arquitetura em ponto fixo, uma nova análise deverá ser realizada, no capítulo a seguir, com intuito de definir o algoritmo a ser testado para uma arquitetura de processamento em FPGA para reconstrução *online* de energia em ambiente de alta luminosidade.

Capítulo 5

Implementação em FPGA

Neste capítulo são apresentadas as análises de complexidade dos algoritmos de implementação *online* do **COF** e a proposta de arquitetura de processamento em FPGA para reconstrução *online* de energia em ambiente de alta luminosidade. A Seção 5.1 apresenta a análise de complexidade dos algoritmos. Na Seção 5.2 será apresentada a arquitetura de processamento desenvolvida em FPGA, composta por uma unidade de controle e processamento, por uma unidade lógica aritmética e por memórias de armazenamento. Por fim, na Seção 5.3 são apresentados os resultados extraídos da arquitetura de processamento desenvolvida.

5.1 Análise de Complexidade

Analizando cada uma das etapas de cálculo dos algoritmos, podemos fazer um levantamento, a priori, da complexidade computacional bem como do número de ciclos de clock necessários para a execução dos algoritmos (complexidade temporal). A quantidade de operações realizadas por iteração e a quantidade de ciclos utilizados em cada etapa das operações permitem a comparação entre os algoritmos. Serão consideradas as somas, multiplicações e divisões entre escalares, para o detalhamento dos algoritmos em relação a sua complexidade e a sua eficiência. Com o objetivo de encontrar o número máximo de ciclos executados pelos algoritmos para o desenvolvimento de uma arquitetura sequencial, iremos considerar no levantamento o seu pior caso. No pior caso, para cada algoritmo, considera-se que o **DM** foi capaz de identificar a existência de sinais empilhados em todos os BCs.

Como visto anteriormente, o algoritmo **GD** é dependente de uma boa escolha do ponto de partida para sua convergência. Além disso, utiliza o **DM** para selecionar a matriz de desconvolução, visando estimar a amplitude de sinais de forma ótima, ou seja, com o mínimo de restrição. Os métodos **GDD** e **GC**, apesar de não utilizarem o **DM** como ponto de partida, é dependente de seu processamento para a seleção das informações dos BCs identificados. Logo, as operações do **DM** são comuns aos três

algoritmos. Sendo assim, para o caso particular em que se observa sinais empilhados em todos os BCs, podemos reescrever a Equação (3.31) como segue:

$$\mathbf{H}_{\text{ótimo}} = \begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} & H_{1,4} & H_{1,5} & H_{1,6} & H_{1,7} \\ H_{2,1} & H_{2,2} & H_{2,3} & H_{2,4} & H_{2,5} & H_{2,6} & H_{2,7} \\ H_{3,1} & H_{3,2} & H_{3,3} & H_{3,4} & H_{3,5} & H_{3,6} & H_{3,7} \\ H_{4,1} & H_{4,2} & H_{4,3} & H_{4,4} & H_{4,5} & H_{4,6} & H_{4,7} \\ H_{5,1} & H_{5,2} & H_{5,3} & H_{5,4} & H_{5,5} & H_{5,6} & H_{5,7} \\ H_{6,1} & H_{6,2} & H_{6,3} & H_{6,4} & H_{6,5} & H_{6,6} & H_{6,7} \\ H_{7,1} & H_{7,2} & H_{7,3} & H_{7,4} & H_{7,5} & H_{7,6} & H_{7,7} \end{pmatrix} \quad (5.1)$$

Sabendo que as amostras do sinal de entrada digitalizado são dadas pela Equação (5.2),

$$\mathbf{r} = \begin{pmatrix} r_{1,1} \\ r_{2,1} \\ r_{3,1} \\ r_{4,1} \\ r_{5,1} \\ r_{6,1} \\ r_{7,1} \end{pmatrix} \quad (5.2)$$

então a amplitude estimada por cada BC é encontrada após as operações de produto e soma, conforme apresentado através da Equação (5.3):

$$\hat{\mathbf{a}} = \begin{pmatrix} \hat{a}_{1,1} \\ \hat{a}_{2,1} \\ \hat{a}_{3,1} \\ \hat{a}_{4,1} \\ \hat{a}_{5,1} \\ \hat{a}_{6,1} \\ \hat{a}_{7,1} \end{pmatrix} = \begin{pmatrix} H_{1,1} * r_{1,1} + \dots + H_{1,4} * r_{4,1} + \dots + H_{1,7} * r_{7,1} \\ H_{2,1} * r_{1,1} + \dots + H_{2,4} * r_{4,1} + \dots + H_{2,7} * r_{7,1} \\ H_{3,1} * r_{1,1} + \dots + H_{3,4} * r_{4,1} + \dots + H_{3,7} * r_{7,1} \\ H_{4,1} * r_{1,1} + \dots + H_{4,4} * r_{4,1} + \dots + H_{4,7} * r_{7,1} \\ H_{5,1} * r_{1,1} + \dots + H_{5,4} * r_{4,1} + \dots + H_{5,7} * r_{7,1} \\ H_{6,1} * r_{1,1} + \dots + H_{6,4} * r_{4,1} + \dots + H_{6,7} * r_{7,1} \\ H_{7,1} * r_{1,1} + \dots + H_{7,4} * r_{4,1} + \dots + H_{7,7} * r_{7,1} \end{pmatrix} \quad (5.3)$$

Podemos notar que, no processo de deconvolução, o **DM** estima a amplitude de cada sinal sobreposto com sete operações de produto e seis operações de soma. No entanto, considerando o desenvolvimento de uma arquitetura para processamento sequencial, podemos utilizar apenas uma operação de produto e uma operação de soma, acumulando o resultado. Desta forma, é possível inferir que serão necessários 49 ciclos de *clock* para realizar todas as etapas de cálculo do **DM**, em que cada BC é estimado em sete ciclos de *clock*.

Após a análise do número de ciclos de *clock* e a quantidade de operações ne-

cessárias para o cálculo da deconvolução, passamos a analisar o complexidade dos métodos propostos. Analisando inicialmente o método **GD**, podemos reescrever a Equação (4.5), com sendo:

$$\hat{\mathbf{a}}[k+1] = \hat{\mathbf{a}}[k] + \mu \mathbf{G}^T \hat{\mathbf{e}}[k] \quad (5.4)$$

em que $\hat{\mathbf{e}}[k] = (\mathbf{G}\hat{\mathbf{a}}[k] - \mathbf{r})$. Na primeira etapa do cálculo de $\hat{\mathbf{e}}[k]$, realizamos a operação de multiplicação entre a matriz \mathbf{G} , que representa as amostras dos sinais de referência, com a amplitudes dos sinais sobrepostos determinados pelo processo de deconvolução, como mostra a Equação (5.5).

$$\mathbf{G}\hat{\mathbf{a}} = \begin{pmatrix} G_{1,1} & G_{1,2} & G_{1,3} & G_{1,4} & G_{1,5} & G_{1,6} & G_{1,7} \\ G_{2,1} & G_{2,2} & G_{2,3} & G_{2,4} & G_{2,5} & G_{2,6} & G_{2,7} \\ G_{3,1} & G_{3,2} & G_{3,3} & G_{3,4} & G_{3,5} & G_{3,6} & G_{3,7} \\ G_{4,1} & G_{4,2} & G_{4,3} & G_{4,4} & G_{4,5} & G_{4,6} & G_{4,7} \\ G_{5,1} & G_{5,2} & G_{5,3} & G_{5,4} & G_{5,5} & G_{5,6} & G_{5,7} \\ G_{6,1} & G_{6,2} & G_{6,3} & G_{6,4} & G_{6,5} & G_{6,6} & G_{6,7} \\ G_{7,1} & G_{7,2} & G_{7,3} & G_{7,4} & G_{7,5} & G_{7,6} & G_{7,7} \end{pmatrix} * \begin{pmatrix} \hat{a}_{1,1} \\ \hat{a}_{2,1} \\ \hat{a}_{3,1} \\ \hat{a}_{4,1} \\ \hat{a}_{5,1} \\ \hat{a}_{6,1} \\ \hat{a}_{7,1} \end{pmatrix} \quad (5.5)$$

No processo de deconvolução, é realizada a multiplicação entre a matriz $\mathbf{H}_{\text{ótimo}}$ com as amostras do sinal de entrada digitalizado \mathbf{r} , onde a complexidade computacional é dada por uma operação de produto e uma operação de soma (acumulando o resultado) e a complexidade temporal é inferida para 49 ciclos de *clock*. Como na primeira etapa de cálculo do $\hat{\mathbf{e}}[k]$, o processamento executado é idêntico ao **DM**, então podemos inferir que também serão necessários 49 ciclos de *clock*. Por se tratar do desenvolvimento de uma arquitetura sequencial, apenas uma unidade de lógica aritmética é utilizada para o produto e soma entre escalares (Equação (5.6)).

$$\mathbf{G}\hat{\mathbf{a}} = \begin{pmatrix} G_{1,1} * a_{1,1} + \dots + G_{1,4} * a_{4,1} + \dots + G_{1,7} * a_{7,1} \\ G_{2,1} * a_{1,1} + \dots + G_{2,4} * a_{4,1} + \dots + G_{2,7} * a_{7,1} \\ G_{3,1} * a_{1,1} + \dots + G_{3,4} * a_{4,1} + \dots + G_{3,7} * a_{7,1} \\ G_{4,1} * a_{1,1} + \dots + G_{4,4} * a_{4,1} + \dots + G_{4,7} * a_{7,1} \\ G_{5,1} * a_{1,1} + \dots + G_{5,4} * a_{4,1} + \dots + G_{5,7} * a_{7,1} \\ G_{6,1} * a_{1,1} + \dots + G_{6,4} * a_{4,1} + \dots + G_{6,7} * a_{7,1} \\ G_{7,1} * a_{1,1} + \dots + G_{7,4} * a_{4,1} + \dots + G_{7,7} * a_{7,1} \end{pmatrix} \quad (5.6)$$

Por fim, na segunda etapa do cálculo de $\hat{\mathbf{e}}[k]$ é necessário realizar a operação de subtração entre o resultado da Equação (5.6) com amostras do sinal de entrada digitalizado, conforme mostra a Equação (5.7). Nesta segunda etapa, podemos

inferir que serão necessários 7 ciclos de *clock*, sendo um ciclo para cada operação de subtração entre escalares. Novamente por tratar-se de operação sequencial, apenas uma estrutura de subtração é necessária para o cálculo.

$$\hat{\mathbf{e}} = \begin{pmatrix} \hat{e}_{1,1} \\ \hat{e}_{2,1} \\ \hat{e}_{3,1} \\ \hat{e}_{4,1} \\ \hat{e}_{5,1} \\ \hat{e}_{6,1} \\ \hat{e}_{7,1} \end{pmatrix} = \begin{pmatrix} G_{1,1} * a_{1,1} + \dots + G_{1,4} * a_{4,1} + \dots + G_{1,7} * a_{7,1} \\ G_{2,1} * a_{1,1} + \dots + G_{2,4} * a_{4,1} + \dots + G_{2,7} * a_{7,1} \\ G_{3,1} * a_{1,1} + \dots + G_{3,4} * a_{4,1} + \dots + G_{3,7} * a_{7,1} \\ G_{4,1} * a_{1,1} + \dots + G_{4,4} * a_{4,1} + \dots + G_{4,7} * a_{7,1} \\ G_{5,1} * a_{1,1} + \dots + G_{5,4} * a_{4,1} + \dots + G_{5,7} * a_{7,1} \\ G_{6,1} * a_{1,1} + \dots + G_{6,4} * a_{4,1} + \dots + G_{6,7} * a_{7,1} \\ G_{7,1} * a_{1,1} + \dots + G_{7,4} * a_{4,1} + \dots + G_{7,7} * a_{7,1} \end{pmatrix} - \begin{pmatrix} r_{1,1} \\ r_{2,1} \\ r_{3,1} \\ r_{4,1} \\ r_{5,1} \\ r_{6,1} \\ r_{7,1} \end{pmatrix} \quad (5.7)$$

Segundo as operações da Equação (4.4), é necessário multiplicar a matriz **G** pelo resultado de $\hat{\mathbf{e}}[k]$, como mostra a Equação (5.8). Uma vez que trata-se de uma multiplicação entre uma matriz 7×7 com vetor 7×1 , então novamente são necessários 49 ciclos de *clock* para se obter o resultado nesta etapa. Este resultado é determinado utilizando a mesma unidade lógica aritmética.

$$\mathbf{G}\hat{\mathbf{e}} = \begin{pmatrix} G\hat{e}_{1,1} \\ G\hat{e}_{2,1} \\ G\hat{e}_{3,1} \\ G\hat{e}_{4,1} \\ G\hat{e}_{5,1} \\ G\hat{e}_{6,1} \\ G\hat{e}_{7,1} \end{pmatrix} = \begin{pmatrix} G_{1,1} & G_{1,2} & G_{1,3} & G_{1,4} & G_{1,5} & G_{1,6} & G_{1,7} \\ G_{2,1} & G_{2,2} & G_{2,3} & G_{2,4} & G_{2,5} & G_{2,6} & G_{2,7} \\ G_{3,1} & G_{3,2} & G_{3,3} & G_{3,4} & G_{3,5} & G_{3,6} & G_{3,7} \\ G_{4,1} & G_{4,2} & G_{4,3} & G_{4,4} & G_{4,5} & G_{4,6} & G_{4,7} \\ G_{5,1} & G_{5,2} & G_{5,3} & G_{5,4} & G_{5,5} & G_{5,6} & G_{5,7} \\ G_{6,1} & G_{6,2} & G_{6,3} & G_{6,4} & G_{6,5} & G_{6,6} & G_{6,7} \\ G_{7,1} & G_{7,2} & G_{7,3} & G_{7,4} & G_{7,5} & G_{7,6} & G_{7,7} \end{pmatrix} * \begin{pmatrix} \hat{e}_{1,1} \\ \hat{e}_{2,1} \\ \hat{e}_{3,1} \\ \hat{e}_{4,1} \\ \hat{e}_{5,1} \\ \hat{e}_{6,1} \\ \hat{e}_{7,1} \end{pmatrix} \quad (5.8)$$

Em seguida, o resultado obtido através da Equação (5.8) deve ser multiplicado pelo valor de μ determinado em 4.2. Neste caso, para cada operação de multiplicação é necessário 1 ciclo de *clock*. Como trata-se apenas de uma operação de multiplicação, e não de uma operação seguida de produto e soma, então não se utiliza a unidade lógica aritmética para determinar o resultado. Ou seja, nesta etapa mais

um multiplicador é utilizado para se obter o resultado.

$$\mu \mathbf{G}\hat{\mathbf{e}} = \begin{pmatrix} \mu G\hat{e}_{1,1} \\ \mu G\hat{e}_{2,1} \\ \mu G\hat{e}_{3,1} \\ \mu G\hat{e}_{4,1} \\ \mu G\hat{e}_{5,1} \\ \mu G\hat{e}_{6,1} \\ \mu G\hat{e}_{7,1} \end{pmatrix} = (0.4) * \begin{pmatrix} G\hat{e}_{1,1} \\ G\hat{e}_{2,1} \\ G\hat{e}_{3,1} \\ G\hat{e}_{4,1} \\ G\hat{e}_{5,1} \\ G\hat{e}_{6,1} \\ G\hat{e}_{7,1} \end{pmatrix} \quad (5.9)$$

Por fim, para completar o cálculo de todas as etapas do método **GD**, para a primeira iteração é necessário realizar a operação de soma entre as amplitudes dos sinais sobrepostos determinados pelo processo de deconvolução (Equação (5.3)) com o resultado obtido através da Equação (5.9), conforme mostra a Equação (5.10). Neste caso, a mesma unidade utilizada para realizar a subtração da Equação (5.7) é utilizada para efetuar a operação de soma, alterando o *bit* de sinal de cada escalar da Equação (5.9). Do ponto de vista da complexidade temporal, são necessários 7 ciclos de *clock* para se obter o resultado.

$$\hat{\mathbf{a}}[k+1] = \begin{pmatrix} \hat{a}_{1,1} \\ \hat{a}_{2,1} \\ \hat{a}_{3,1} \\ \hat{a}_{4,1} \\ \hat{a}_{5,1} \\ \hat{a}_{6,1} \\ \hat{a}_{7,1} \end{pmatrix} + \begin{pmatrix} \mu G\hat{e}_{1,1} \\ \mu G\hat{e}_{2,1} \\ \mu G\hat{e}_{3,1} \\ \mu G\hat{e}_{4,1} \\ \mu G\hat{e}_{5,1} \\ \mu G\hat{e}_{6,1} \\ \mu G\hat{e}_{7,1} \end{pmatrix} \quad (5.10)$$

Através destas análises, podemos concluir que, para o método **GD** são necessários 168 ciclos de *clock* para realizar uma iteração do algoritmo, sendo 49 destes ciclos utilizados para o cálculo do **DM**. Do ponto de vista da complexidade computacional são necessários dois multiplicadores e dois somadores, sendo um multiplicador e um somador os responsáveis pelas operações entre matrizes 7x7 e vetores 7x1 (unidade lógica aritmética).

Já conhecido as etapas de análise de complexidade do método **GD**, podemos então analisar diretamente os demais métodos a partir dos Algoritmos (2) e (3). Analisando o método **GDD**, podemos inferir que para a linha cinco do Algoritmo (2), serão necessários 105 ciclos de *clock* para as operações entre matrizes 7x7 e vetores 7x1. Observando a sexta linha do mesmo algoritmo é possível inferir que serão necessários 77 ciclos de *clock* para as operações de multiplicação, soma e divisão. Por fim, analisando as etapas de cálculo para a linha sete, podemos observar que

serão necessários 14 ciclos de *clock* para que o resultado seja encontrado. Levando em conta as operações realizadas pelo **DM**, podemos concluir que a complexidade temporal para o método **GDD** é de 245 ciclos de *clock*, considerando apenas uma iteração. Do ponto de vista de recursos de hardware, o algoritmo **GDD** utiliza-se de uma unidade lógica aritmética (um multiplicador e um somador), um somador, um multiplicador e um divisor isolado para efetuar as operações de soma, subtração, multiplicação e divisão entre vetores 7x1.

Através dos mesmos procedimentos de análise de complexidade até agora apresentado, podemos inferir que para a linha quatro do Algoritmo (3) (método **GC**) são necessários 105 ciclos de *clock*. Observando a sétima linha do mesmo algoritmo é possível inferir que serão necessários 77 ciclos de *clock* para as operações de multiplicação, soma e divisão. Já para a linha oito, é possível inferir que a complexidade temporal é de 14 ciclos de *clock*. Sabendo que as etapas de operação da linha 9 são idênticas as etapas da linha quatro, então sua complexidade computacional é de 105 ciclos. Como a linha 10 realiza duas operações de produto e uma divisão entre vetores, é possível inferir que são necessários 21 ciclos de *clock*. Por fim, analisando os cálculos realizados pela linha 11, podemos observar que serão necessários 14 ciclos de *clock*. Logo, do ponto de vista da complexidade temporal, podemos concluir que são necessários 364 ciclos de *clock* para que todas as operações sejam realizadas. Do ponto de vista de recursos de *hardware*, e levando em consideração o desenvolvimento de uma arquitetura sequêncial, podemos concluir que o método **GC** é equivalente ao método **GDD**, pois são utilizados também um somador, um multiplicador e um divisor isolado para efetuar todas as operações de soma, subtração, multiplicação e divisão entre vetores 7x1, além de uma unidade lógica aritmética para operações de produto e soma acumulada.

Após concluído o levantamento de complexidade para os três métodos, analisando apenas uma iteração, e considerando o desenvolvimento de uma arquitetura sequêncial, podemos então avaliar a melhor opção de implementação confrontando as conclusões desta seção com as conclusões obtidas através da simulação das técnicas propostas. Para tal, são apresentadas duas tabelas demonstrando a complexidade dos algoritmos para células do Barril e Barril Estendido, respectivamente.

Algoritmos	Recursos de Hardware			Ciclos de Clock	Iterações
	Adição	Multiplicação	Divisão		
GD	2	2	-	1596	13
GDD	2	2	1	1813	9
GC	2	2	2	1939	6

Tabela 5.1: Análise de complexidade para células do Barril (ocupância de 10%), considerando o desenvolvimento de uma arquitetura sequêncial.

Através da Tabela (5.1) podemos observar que o melhor método para implementação em FPGA é o algoritmo Gradiente Descendente, dado que sua complexidade computacional é inferior aos demais métodos, assim como sua complexidade temporal. A Tabela (5.1) mostra que o **GD** e o **DM**, juntos, podem ser processados com 1596 ciclos de *clock*, utilizando-se de uma arquitetura de processamento sequêncial. Neste caso, sabendo que o segundo nível de filtragem possui tempo de processamento máximo de 10 us e visto que 13 iterações são suficientes para as células do Barril, então este algoritmo pode ser processado a uma frequencia aproximada de 160 MHz, considerando a ocupância de 10%.

Algoritmos	Recursos de Hardware			Ciclos de Clock	Iterações
	Adição	Multiplicação	Divisão		
GD	2	2	-	2429	20
GDD	2	2	1	2597	13
GC	2	2	2	2254	7

Tabela 5.2: Análise de complexidade para células do Barril Estendido (ocupância de 20%), considerando o desenvolvimento de uma arquitetura sequêncial.

Na Seção 4.2 do Capítulo 4 comprovamos que, para as células do Barril Estendido, há uma necessidade de mais iterações devido a sua maior ocupância. Ao compararmos as Tabelas (5.1) e (5.2) podemos notar que apenas o **GC** mantém o número de iterações, pois em seu pior caso, o máximo de iterações a serem executadas é idêntico ao número de sinais empilhados observados. Sendo o **GC** o algoritmo mais eficiente do ponto de vista do número de iterações, neste caso o método é também o mais eficiente do ponto de vista da complexidade temporal.

Continuando a análise da Tabela (5.2), onde identifica-se o pior caso de simulação (ocupância de 20%), podemos notar que o algoritmo menos eficiente é o **GDD**, pois apresenta maior complexidade temporal. Sua complexidade computacional, considerando o desenvolvimento de uma arquitetura sequêncial, é equivalente ao **GC**, que consequentemente é superior ao **GD**. Podemos observar também, através da Tabela (5.2), que o método **GD** além de ser mais eficiente do ponto de vista da complexidade computacional, possui eficiência aproximada ao **GC**.

No levantamento de complexidade é possível notar que o **GC** apresenta mais etapas de cálculo do que os demais algoritmos. Neste caso, apesar do **GC** exigir poucas iterações, o número elevado de etapas de cálculo exigem um maior número de ciclos de *clock* para seu processamento. Este fato pode ser melhor compreendido observando apenas o número de ciclos de *clock* necessário para o processamento do **GC** (2254) em comparação com o **GD** (2429). Desta forma, é possível compará-los e concluir que para o desenvolvimento de uma arquitetura sequencial o **GC** possui uma complexidade temporal que equivale a aproximadamente 19 iterações

do método **GD**.

Considerando o critério de parada do método **GD** em 20 iterações, podemos reescrever a Tabela (4.2.4) para comparar a eficiência dos métodos **GD** e **GC** na estimativa de sinais.

	a_{-2}	a_0	a_{+3}	<i>Iterações</i>
REAL	29.0000	25.0000	27.0000	-
COF	25,2570	29,0967	26,3277	-
GD	25,2570	29,0967	26,3277	20
GC	25,2570	29,0967	26,3277	3

Tabela 5.3: Comparativo entre os Métodos (**COF**, **GD** e **GC**) a partir da amplitude real de sinais sobrepostos.

Nesta simulação funcional dos algoritmos **GD** e **GC**, podemos concluir que os métodos possuem eficiência similar na estimação da amplitude de sinais sobrepostos. Sabendo que para 2429 ciclos de *clock* a frequência de processamento do **GD** é equivalente a 242,9 MHz e para 2254 ciclos de *clock* a frequência do **GC** é equivalente a 225,4 MHz, então por apresentarem uma complexidade temporal aproximada e sendo o **GD** o método mais simples do ponto de vista da complexidade computacional, podemos então definir o método **GD** como o algoritmo a ser testado na proposta de arquitetura de processamento em FPGA para reconstrução *online* de energia em ambiente de alta luminosidade.

As características da arquitetura de processamento do **GD** são detalhadas na Seção 5.2, como segue.

5.2 Arquitetura do Processador em FPGA

Diante das características da arquitetura de um dispositivo FPGA [40], cuja função de sua estrutura de matriz de células ou de blocos lógicos permite o desenvolvimento de arquiteturas com fluxo paralelo de dados ou sequencial, a seguir descreve-se o *Top level* da arquitetura do processador para execução seqüencial do algoritmo **GD**. Posteriormente, através da Seção 5.3, são apresentados os detalhes da simulação do processador em FPGA.

A Figura (5.1) mostra o *Top level* do processador e suas características. O bloco PROCESSOR tem como função o controle e processamento de todas as etapas de cálculo do algoritmo **GD**. Através de uma máquina de estados, o bloco recebe e armazena os dados do segundo nível de *trigger*. A primeira janela de 7 amostras é processada, enquanto as demais aguardam a execução do algoritmo. Ao seu término, as demais são processadas sucessivamente, assim como a primeira.

Separando o controle e processamento em etapas de cálculo, assim como demonstrado na Seção 5.1, inicialmente o bloco PROCESSOR controla as etapas de cálculo do processo de deconvolução, o **DM**.

Neste caso, a máquina de estados realiza a leitura das janelas de sinais empilhados armazenadas no bloco PROCESSOR e a leitura da matriz de referência (Equação (5.1)) armazenada no bloco RAM1. Por apresentar uma estrutura simples de cálculos através de produto e soma acumulada (filtro FIR), o **DM** é então calculado por uma unidade lógica aritmética, aqui chamado de bloco ULA. Uma vez que o **GD** apresenta etapas de cálculo, tanto com a matriz de referência como com sua inversa, então a máquina de estados também tem como função o controle dos blocos RAM2 e RAM1, sendo este último bloco a memória de armazenamento da matriz de referência inversa. O controle destas matrizes é realizada por meio do bloco MUX1. Para cada amplitude estimada pelo **DM**, o bloco PROCESSOR controla a escrita dos resultados de cada BC nas primeiras sete posições de memória (0 a 6) do bloco RAM3. Neste caso, a máquina de estados se encarrega de selecionar a saída de dados do bloco ULA para entrada de dados do bloco RAM3, através do MUX2. No cálculo do **DM**, o próprio bloco ULA, controlado pelo bloco PROCESSOR, se encarrega de verificar se as informações (sinais empilhados) são maiores do que o limiar de 5 ADC *counts*.

Após finalizado as etapas de cálculo do processo de deconvolução, tendo as primeiras sete posições de memória do bloco RAM3 os pontos iniciais de convergência do algoritmo **GD**, o bloco PROCESSOR, então, inicializa a primeira iteração do método. Nesta etapa, o bloco PROCESSOR controla as operações realizadas pela Equação (5.6). Por apresentar uma estrutura de filtro FIR, o cálculo é realizado pelo bloco ULA. Para tal, o bloco PROCESSOR se encarrega de ler os dados da memória RAM2, selecionando-os por meio do MUX1, e também se encarrega do controle para o armazenamento do resultado no bloco RAM3, antes da leitura das primeiras sete posições de memória que contém as amplitudes estimadas pelo **DM**.

O resultado da Equação (5.6) é armazenado nas posições de 7 a 13 do bloco RAM3. Em seguida, as etapas de cálculo da Equação (5.7) são executadas. Neste caso, por se tratar de uma operação isolada, o próprio bloco PROCESSOR se encarrega de realizar a operação de subtração e controlar o armazenamento do resultado, através do MUX2, sobrepondo as posições de 7 a 13 do bloco RAM3. Continuando as etapas de cálculos ainda na primeira iteração, o bloco PROCESSOR se encarrega do controle para o processamento da Equação (5.8). Novamente, por apresentar uma estrutura de filtro FIR, o cálculo é realizado pelo bloco ULA. O resultado da Equação (5.8) é armazenado nas posições de 14 a 20 do bloco RAM3. Por conseguinte, o bloco PROCESSOR realiza a leitura das posições de 14 a 20 do bloco RAM3 e multiplica cada dado pelo valor de μ determinado nas simulações,

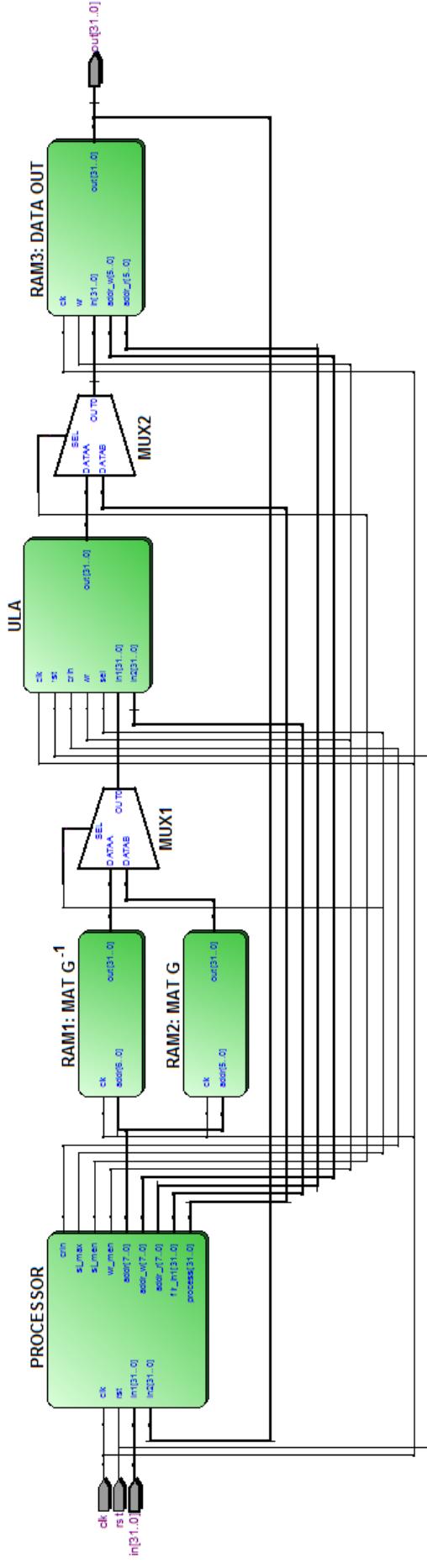


Figura 5.1: Arquitetura do Processador em FPGA

conforme a Equação (5.8). Por se tratar de uma operação isolada de multiplicação, o próprio bloco PROCESSOR se encarrega de realizar a operação. O resultado da Equação (5.8) é armazenado no bloco RAM3 nas posições de 21 a 27.

Por fim, o bloco PROCESSOR realiza a leitura das posições de 0 a 6, correspondente ao resultado do **DM**, e de 21 a 27 para realizar a última etapa de cálculo, dado pela Equação (5.10). A operação de soma é então executada e o resultado é então armazenado nas posições de 0 a 6 do bloco RAM3, sobrepondo o resultado do resultado do **DM** e permitindo a execução das demais iterações do **GD**. Ao término de todas as iterações necessárias, definidas neste trabalho, a máquina de estados retorna ao estado inicial para o cálculo do **DM**, considerando a segunda janela de sinais empilhados. Novamente, e para cada janela de sinais empilhados lida, todas as etapas de cálculo são executadas para a estimativa dos sinais sobrepostos através do método iterativo Gradiente Descendente.

Visando o melhor entendimento do processador desenvolvido, a seguir são apresentados os diagramas dos principais blocos da arquitetura em **FPGA**.

5.2.1 Controle e Processamento

A Figura (5.2) apresenta o fluxograma da máquina de estados, responsável pelo controle e processamento do algoritmo **GD**. Na proposta de arquitetura de processamento a máquina de estados pode ser alterada para a execução dos algoritmos **GDD** e **GC**.

A *state-machine* possui 7 estados principais que permitem basicamente o armazenamento dos dados de entrada para posterior processamento (IDLE), a inicialização das etapas de cálculo (START), o cálculo das etapas compostas por produto e soma acumulada (ULA), o controle de troca entre estados (CTRL), o cálculo das etapas correspondente ao erro (ERRO), o cálculo relacionado à multiplicação com μ (MU) e por fim, o cálculo das etapas em que a amplitude dos sinais são estimadas (GD).

Notadamente, por vezes a *state-machine* fecha o laço entre o segundo e último estado com intuito de realizar todas as 20 iterações necessárias para o cálculo da amplitude dos sinais com valores próximos ao **COF**. Considerando apenas uma iteração, podemos notar que o estado CTRL será capaz de inicializar o estado START por três vezes, realizando as etapas de cálculo das Equações (5.3), (5.6) e (5.8), que totalizam 147 ciclos de *clock*. Seguindo a mesma análise, podemos notar que o estado CTRL será capaz de controlar e executar as etapas de cálculo dos estados ERRO, MU e **GD** para as Equações (5.7), (5.9) e (5.10) respectivamente, totalizando 21 ciclos de *clock*. Logo para uma iteração as etapas são executadas em 168 ciclos de *clock* e para vinte iterações o **GD** converge para valores próximos ao **COF** com 2429 ciclos de *clock*.

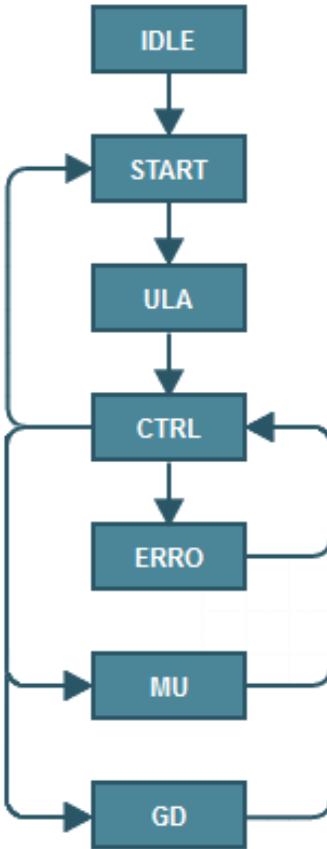


Figura 5.2: Formato para Ponto Fixo.

5.2.2 Aritmética de Ponto Fixo

A implementação das etapas de cálculo do método **GD** foram baseadas em aritmética de ponto fixo para a precisão na estimativa dos sinais sobrepostos. Através da aritmética de ponto fixo foi possível representar a parte inteira e fracionária dos algarismos, predeterminado a posição da vírgula. A Figura (5.3) demonstra o formato para ponto fixo:

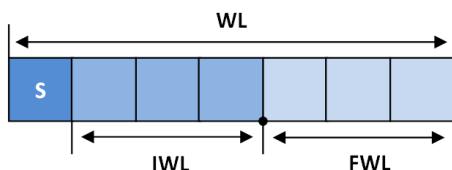


Figura 5.3: Formato para Ponto Fixo.

Na Figura (5.3) a sigla **WL** representa a quantidade de bits total, ou comprimento da palavra, **IWL** o comprimento da porção inteira, **FWL** o comprimento da porção fracionária e **S** o bit de sinal.

No desenvolvimento da arquitetura foi utilizado uma palavra de 32 bits, representação em complemento de 2, dada por $WL = 1 + IWL + FWL$, sendo 1 bit de sinal, 16 bits para representar valores inteiros e 15 bits para representar valores

fracionários. A faixa de números dessa representação é definida por $\{-2^{IWL}, 2^{IWL}\}$ e a resolução dada por 2^{-FWL} [41].

A Figura (5.4) exemplifica a arquitetura para soma e subtração em ponto fixo. Para efetuar as operações de soma ($SEL = 1$), a saída do MUX será sempre o próprio valor do bit de sinal. Para efetuar as operações de subtração ($SEL = 0$), o bit de sinal do número a direita é sempre verificado. Caso este bit seja igual a zero, a saída do MUX é sempre alterada para 1. Desta forma, a mesma lógica utilizada para para soma é também utilizada para subtração.

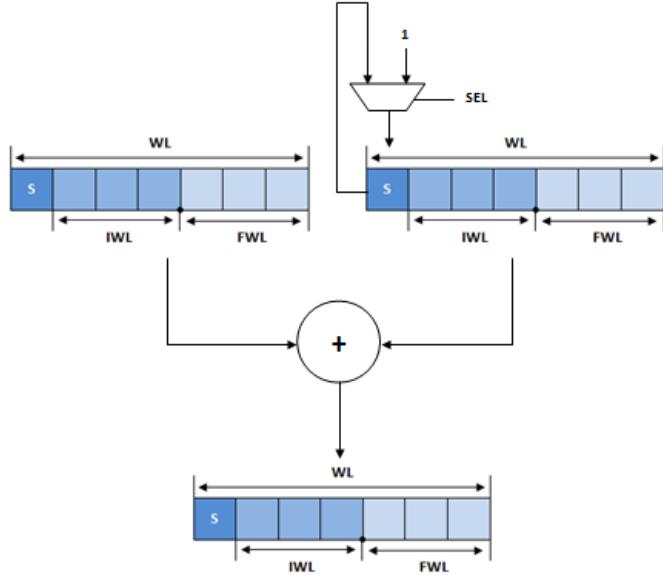


Figura 5.4: Soma e subtração em ponto fixo.

As operações de multiplicação isolada seguem a mesma arquitetura da Figura (5.4), porém não há controle do bit de sinal e, ao invés da estrutura de um somador, é utilizado um multiplicador. Já para as operações de produto e soma acumulada, a arquitetura é compreendida por um multiplicador em série com um somador, sendo que a saída do somador é registrada para realimentá-lo. As operações de divisão isolada correspondentes os algoritmos **GDD** e **GC** foram implementados, e podem utilizados após a alteração da máquina de estados para atender tais algoritmos.

5.2.3 Memórias de Armazenamento

Conforme abordado, a Figura (5.4) se utiliza de três memórias internas da FPGA, além memória de armazenamento dos dados de entrada. As memórias RAM2 e a RAM1 são memórias idênticas com 64 posições e palavra de 32 bits. Por se tratar da matriz de referência e de sua inversa, estas memórias são somente de leitura. As amostras dos sinais de referência são então carregadas a priori nas memórias para permitir a execução das etapas de cálculo do **GD**. Já a RAM3 é uma memória de

32 posições e palavra de 32 bits. Conforme apresentado, a RAM3 é uma memória de leitura e escrita de dados. Sua principal função é armazenar o resultado de cada uma das etapas de cálculo permitindo a execução seqüencial do algoritmo.

5.3 Resultados do Processador em FPGA

Para a análise do processador desenvolvido em FPGA, foi realizada a simulação em FPGA para a comparação com os resultados simulados em ambiente MatLab. Novamente foi considerado, como exemplo, um vetor de amplitudes [0 29 0 25 0 0 27]. Após a adição de ruído, os valores de simulação para as amostras de entrada \mathbf{r} é dado por [14.2392 29.0015 28.6983 30.0245 17.3710 16.8980 25.3826]. Neste caso, novamente temos a presença de sinais empilhados no BC central com influência dos BCs a_{-2} e a_{+3} .

No processo de identificação de sinais empilhados, o **DM** simulado em MatLab resultou em um vetor com valores [0 23.7825 0 29.1809 0 0 22.7578]. Com eficiência similar, o processador desenvolvido executou as etapas de cálculo do **DM** e armazenou como resultado os valores [0 23.7827 0 29.1809 0 0 22.7579]. Assim, o resultado do processador, prova que para um limiar de 5 ADC *counts*, o **DM** executado em FPGA é capaz de identificar o empilhamento no BC central e nos BCs a_{-2} e a_{+3} .

O resultado do **DM**, é então utilizado como ponto de partida para o método **GD**. Nas iterações seguintes, o processador executa todas as etapas de cálculo do **GD** para encontrar a estimava da amplitude dos sinais sobrepostos. O resultado de cada iteração do **GD** é armazenado nas mesmas posições de memória do **DM**. Cada um destes valores foram lidos da memória RAM3 com intuito de gerar a curva de convergência. Sendo assim, a Figura (5.5) demonstra a curva de convergência do algoritmo **GD** para a comparação com os resultados obtidos em MatLab.

Através da curva de convergência podemos concluir o correto funcionamento do processador desenvolvido em FPGA, bem como a precisão dos resultados para os valores em estrutura de ponto fixo. Nos pontos de mínimo, ou seja em 20 iterações, o processador estima, através do **GD**, a amplitude dos sinais com valores próximos aos valores simulados em MatLab, conforme apresentado pelas Tabelas (4.2.4) e (5.3). Consequentemente, o resultado obtido com o processador é aproximadamente igual ao resultado obtido através do **COF**. A Tabela (5.3) apresenta os valores em ponto fixo para a convergência da Figura (5.5).

Para o pior caso a frequência de operação é de 250 MHz, porém na média, teremos muito menos sinais empilhados do que foi visto (10% para o barril e 20% para o barril estendido), de modo que podemos diminuir o *clock* para 20% deste valor. Além disso, é possível alterar facilmente a arquitetura desenvolvida para realizar o Filtro FIR

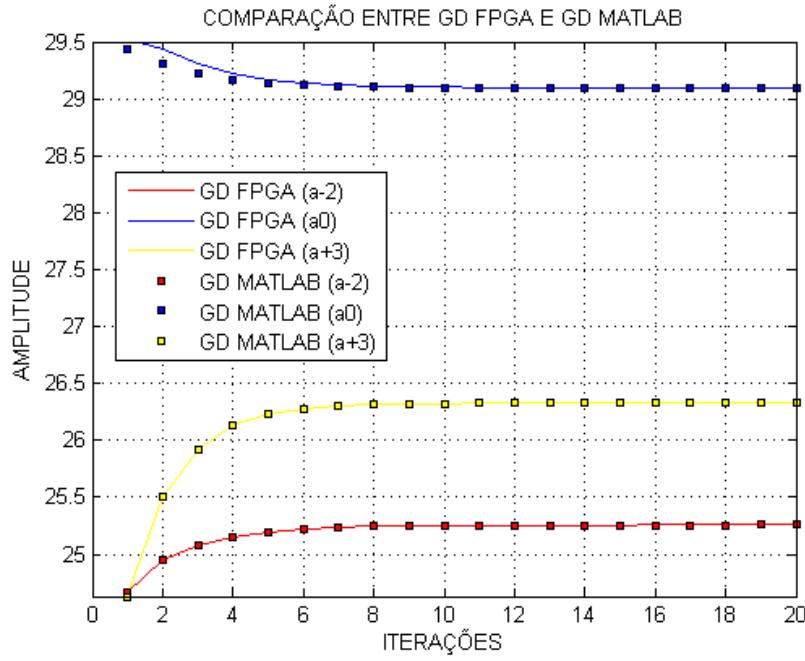


Figura 5.5: Comparaçāo entre o GD FPGA e o GD MATLAB.

Iterações	GD FPGA			GD MATLAB		
	a_{-2}	a_0	a_{+3}	a_{-2}	a_0	a_{+3}
1	24.6717	29.5284	24.6312	24.6714	29.4309	24.6303
2	24.9552	29.4308	25.5055	24.9549	29.3092	25.5049
3	25.0819	29.3093	25.9243	25.0817	29.2241	25.9239
4	25.1522	29.2242	26.1280	25.1520	29.1713	26.1275
5	25.1942	29.1715	26.2280	25.1940	29.1400	26.2276
6	25.2197	29.1402	26.2776	25.2195	29.1216	26.2772
7	25.2350	29.1219	26.3023	25.2348	29.1110	26.3020
8	25.2441	29.1112	26.3148	25.2440	29.1048	26.3145
9	25.2495	29.1051	26.3212	25.2494	29.1013	26.3208
10	25.2527	29.1016	26.3245	25.2525	29.0993	26.3241
11	25.2545	29.0995	26.3261	25.2544	29.0981	26.3258
12	25.2556	29.0984	26.3270	25.2555	29.0975	26.3267
13	25.2562	29.0977	26.3275	25.2561	29.0971	26.3271
14	25.2566	29.0973	26.3278	25.2564	29.0969	26.3274
15	25.2568	29.0971	26.3279	25.2566	29.0968	26.3275
16	25.2569	29.0970	26.3280	25.2567	29.0967	26.3276
17	25.2570	29.0969	26.3280	25.2568	29.0966	26.3276
18	25.2570	29.0969	26.3280	25.2568	29.0966	26.3276
19	25.2570	29.0968	26.3280	25.2569	29.0967	26.3276
20	25.2570	29.0968	26.3280	25.2570	29.0967	26.3277

Tabela 5.4: Tabela com valores de convergência do algoritmo **GD**.

(sete produtos e seis somas) em único ciclo de *clock*, diminuindo consideravelmente a frequência de operação, porém aumenta os recursos utilizados em FPGA. Como

consequência deste aumento, menos canais em paralelo poderão ser implementados em um mesma FPGA.

Capítulo 6

Conclusões e Trabalhos Futuros

Esta dissertação apresentou os estudos sobre a implementação *online* em FPGA do método **COF** para a estimativa da amplitude de sinais no Calorímetro Hadrônico do **ATLAS** em cenários de alta luminosidade. Neste cenário, o **COF** apresenta desempenho superior ao algoritmo *online* de reconstrução de energia (**OF**). No entanto, o **COF** possui elevada complexidade o que dificulta sua implementação para dentro dos requisitos de tempo de processamento exigidos no TileCal (10 us). Portanto, técnicas para reduzir a complexidade **COF** são cruciais.

Dada a complexidade do **COF**, este trabalho mostrou que a obtenção de pseudo-inversa pôde ser realizada através do processo de minimização do erro médio quadrático entre a solução do problema e o modelo linear adotado para estimativa de sinais empilhados. Sendo este o principal ponto de contribuição deste trabalho, foi possível reduzir a complexidade do **COF** utilizando métodos iterativos.

O trabalho apresentou três métodos iterativos que possibilitam a adaptação do **COF** para implementação *online*. Foi demonstrado que o método iterativo **GC** (Gradiente Conjugado) converge para um número finito de iterações (p iterações), que correspondem exatamente a quantidades de sinais empilhados observados (máximo 7 iterações). O método evita a inversão de matrizes com operações de produto, soma e duas divisões. Já o método **GD** (Gradiente Descendente) é dependente de uma boa escolha dos parâmetros de convergência (taxa de convergência e ponto de partida) para estimar com eficiência a amplitude de sinais empilhados. Este método evita a inversão de matrizes com simples operações de produto e soma entre escalares, apenas. O método **GDD** (Gradiente Descendente com Taxa de Convergência Dinâmica) localiza o ponto de mínimo com menor número iterações se comparado com o **GD**, no entanto, o método evita a inversão de matrizes com operações de produto, soma e uma divisão entre escalares.

Simulando a resposta do sistema de calorimetria no MatLab foi possível comparar os métodos iterativos com o **COF**. Neste ambiente de simulação, o cálculo do erro médio quadrático (**MSE**) possibilitou verificar o desvio entre os métodos ite-

rativos e o **COF** na estimativa de sinais empilhados. Com isso, foi possível concluir que o **GD** necessita, em média, de apenas 20 iterações para se chegar a convergência considerando seu pior caso (ocupância de 20%). Da mesma forma, concluiu-se que, em média, o **GDD** é capaz de convergir com apenas 13 iterações. Já para o **GC** foi possível confirmar que o número máximo de iterações para convergência é dependente da quantidade de sinais empilhados observados.

Como última etapa de análise, e visando a implementação em FPGA, este trabalho apresentou estudos sobre a complexidade dos métodos iterativos. Analisando cada uma das etapas de cálculo dos algoritmos, bem como do número de ciclos de *clock* necessários para a execução dos mesmos, foi possível compara-los. Assim, concluiu-se que o **GC**, apesar de exigir poucas iterações, o número elevado de etapas de cálculo demanda maior número de ciclos de *clock* para seu processamento, 2254 ciclos. Neste caso, para atender aos requisitos de tempo de processamento exigidos no TileCal, o mesmo deve ser processado a um *clock* 225,4 MHz. Para o método **GD**, verificou-se a necessidade de 2429 ciclos de *clock* e uma frequência de processamento (242,9 MHz), considerando uma arquitetura sequencial.

Baseado em um arquitetura sequencial e utilizando lógica aritmética em ponto fixo, este trabalho mostrou que a arquitetura desenvolvida é capaz executar o método iterativo e consequentemente estimar a amplitude de sinais empilhados com desempenho similar ao método **COF**, o que viabiliza a implementação nas sRODs para processamento *online*. Para a arquitetura desenvolvida, a alteração da máquina de estados possibilita a implementação dos demais algoritmos, além do **GD**. Além disso, é possível alterar facilmente a arquitetura desenvolvida para realizar o Filtro FIR (sete produtos e seis somas) em único ciclo de *clock*, diminuindo consideravelmente a frequência de operação, porém aumentando consideravelmente os recursos utilizados em FPGA. No caso particular do algoritmo **GD**, em ambiente de simulação da FPGA, comprovou-se que a arquitetura de processamento foi capaz de estimar a amplitude de sinais empilhados com valores similares aos valores simulados no MatLab para o **COF**.

6.1 Trabalhos Futuros

Motivado pelos resultados positivos deste trabalho para a estimação da amplitude de sinais no Calorímetro Hadrônico do **ATLAS** em cenários de alta luminosidade, as propostas de trabalhos futuros concentram-se inicialmente nos testes com dados reais do **ATLAS** para posterior implementação e testes dos algoritmos nas sRODs. Novas propostas de arquiteturas paralelas de processamento também serão consideradas futuramente. Neste caso, devido a quantidade de canais a serem observados e o aumentando considerável de recursos utilizados em FPGA, um

estudo mais detalhado sobre a quantidade de elementos lógicos necessários para o processamento de sinais por célula, deverá ser realizado. Também como proposta de trabalhos futuros, está o estudo e desenvolvimento de novos métodos de estimação para calorimetria em cenários de alta luminosidade. Além disso, será realizada uma avaliação do impacto da utilização destes métodos na reconstrução da energia de uma determinada partícula resultante da colisão, já que as análises realizadas até o momento só avaliam o impacto a nível de canal e não olhando para todo o detector.

Referências Bibliográficas

- [1] CERN. “The history of CERN”. , 2014. Disponível em: <<http://timeline.web.cern.ch/timelines/The-history-of-CERN>>.
- [2] EVANS, L., BRYANT, P. “LHC Machine”, *Journal of Instrumentation, JINST* 3 S08001, 2008.
- [3] THE ATLAS COLLABORATION. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”, *Physics Letters B*, v. 716, pp. 1–29, 2012.
- [4] WIKIPEDIA. “Higgs Boson”. , 2014. Disponível em: <http://en.wikipedia.org/wiki/Higgs_boson>.
- [5] CERN. “CERN Document Server”. , 2014. Disponível em: <http://cds.cern.ch>.
- [6] CERN. “ALICE Collaboration”. , 2014. Disponível em: <http://aliweb.cern.ch>.
- [7] CERN. “CMS Experiment”. , 2014. Disponível em: <http://cms.web.cern.ch>.
- [8] CERN. “LHCb Collaboration”. , 2014. Disponível em: <https://lhcb.web.cern.ch/lhcb>.
- [9] CERN. “ATLAS”. , 2014. Disponível em: <http://www.atlas.ch>.
- [10] SÁNCHEZ, C. A. S. *Implementation of the ROD Crate DAQ Software for the ATLAS Tile Calorimeter and a Search for a MSSM Higgs Boson Decaying into Tau Pairs*. Tese de Ph.D., (<https://cds.cern.ch/record/1309926>) Universitat de Valéncia - CSIC, Valência, Espanha, 2010.
- [11] CERN. “CERN Document Server”. , 2013. Disponível em: <<https://cds.cern.ch/record/1095924>>.

- [12] WONG, C. *Introduction to High-Energy Heavy-Ion Collisions*. World Scientific, 1994.
- [13] WIGMANS, R. *Calorimetry: Energy Measurement in Particle Physics*. Oxford University Press, 2000.
- [14] COLLABORATION, T. A. “The ATLAS Experiment at the CERN Large Hadron Collider”, *Journal of Instrumentation*, v. 3, n. 08003, 2008.
- [15] ROS, E. “ATLAS inner detector”, *Nuclear Physics B - Proceedings Supplements*, v. 120, pp. 235–238, 2003.
- [16] PALESTINI, S. “The Muon Spectrometer of the ATLAS Experiment”, *Nuclear Physics B*, v. 125, pp. 337–345, 2003.
- [17] ATLAS GROUP. *ATLAS Tile Calorimeter Technical Design Report*. Relatório técnico, CERN/LHCC/96–42, 1996.
- [18]ADRAGNA, P. E. A. “The ATLAS hadronic tile calorimeter: from construction toward physics”, *Nuclear Science*, v. 53, pp. 1275 – 1281, June 2006.
- [19] <https://cds.cern.ch/record/1095927> (acessado em dezembro de 2013).
- [20] TILECAL COLLABORATION. *ATLAS Tile Calorimeter Technical Design Report*. Atlas tdr 3, cern/lhcc/96-42, CERN, Geneva, 1996.
- [21] ATLAS. “The ATLAS Experiment at CERN”. , maio 2012. Disponível em: <<http://www.atlas.ch>>.
- [22] KORDAS, K., ABOLINS, M. “The ATLAS Data Acquisition and Trigger: concept, design and status”, *Nuclear Physics B - Proceedings Supplements*, v. 172, pp. 178–182.
- [23] BIOT, J. A. V. *TileCal Read-Out Drivers Production and Firmware Developments*. Relatório Técnico. Relatório técnico, IFIC - Universitat de Valencia.
- [24] ANDERSON, K. “Design of the front-end analog electronics for the ATLAS tile calorimeter”, *Nuclear Instruments and Methods in Physics Research A*, v. 551, pp. 469–476, 2005.
- [25] ATLAS. *Level-1 Trigger*. Technical Design Report, ATLAS TDR-12, 1998.
- [26] KLIMEK, P. “ATLAS Tile Calorimeter Data Quality Assessment with Commissioning Data”, *15th International Conference on Calorimetry in High Energy Physics*, link: <https://cdsweb.cern.ch/record/1473499>, 2012.

- [27] CARRIÓN, F., CASTILLOA, V., FERRERA, A., et al. “The sROD module for the ATLAS Tile Calorimeter Phase-II Upgrade Demonstrator”, *Topical Workshop on Electronics for Particle Physics 2013 (TWEPP-13)*, pp. 1–13, 2013.
- [28] DE A. FILHO, L. M. “Calorimeter Response Deconvolution for Energy Estimation in High-Luminosity Conditions”, 2014.
- [29] KAY, S. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Pearson, 1993.
- [30] CLELAND, W., STERN, E. “Signal processing considerations for liquid ionization calorimeters in a high rate environment”, *Nuclear Instruments and Methods in Physics Research*, v. A338, pp. 467–497, 1994.
- [31] BERTSEKAS, D. P. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1996. ISBN: 1886529043.
- [32] JENSEN, J. R., GLENTIS, G., CHRISTENSEN, M. G. “Fast LCMV-Based Methods for Fundamental Frequency Estimation”, *Signal Processing, IEEE Transactions*, v. 61, 2013.
- [33] CAPON, J. “High Resolution Frequency-Wavenumber Spectrum Analysis”, *Signal Processing, IEEE Transactions*, 1969.
- [34] FULLANA, E. “Optimal Filtering in the ATLAS Hadronic Tile Calorimeter”, *IEEE Transactions On Nuclear Science*, v. 53, n. 4, 2006.
- [35] CERN. “Public Tile Calorimeter Plots for Collision Data”. , 2015. Disponível em: <https://twiki.cern.ch/twiki/bin/view/AtlasPublic/TileCaloPublicResults#Energy_Reconstruction>.
- [36] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, segunda edição, 1998.
- [37] HAYKIN, S. *Adaptive Filter Theory*. Prentice Hall, terceira edição, 1996.
- [38] LUENBERGER, D. G., Y. *Linear and Nonlinear Programming*. Reading, MA, Addison-Wesley, 2008. ISBN: 978-0-387-74502-2.
- [39] AAD, G., OTHERS. “The ATLAS Experiment at the CERN Large Hadron Collider.” *JINST*, v. 3, pp. S08003, 2008.
- [40] MEYER-BAESE, U. *Digital Signal Processing with Field Programmable Gate Arrays*. Springer, 2007.

- [41] YATES R., RANDY. *Fixed-Point Arithmetic: An Introduction*. Relatório técnico, Digital Signal Labs, 2 2013.

Apêndice A

Coordenadas do ATLAS

O sistema de coordenadas usados em experimentos com feixes não é o sistema polar. É um sistema adequado ao formato cilíndrico dos detectores dispostos ao redor do ponto de impacto, ou seja, um sistema que acompanha a direção dos feixes de partículas provenientes da colisão. As coordenadas empregadas são η , ϕ e z em contraposição a x , y e z . Os termos η e ϕ seguem a uma transformação não-linear de x e y .

A Figura A.1 pode ser explicativa quanto ao sistema. Em sua parte superior, é possível ver um esquema do barril e da tampa de um detector, mostrando como se comportam as coordenadas tomando por referência as coordenadas cartesianas x , y e z (marcadas em pontilhado).

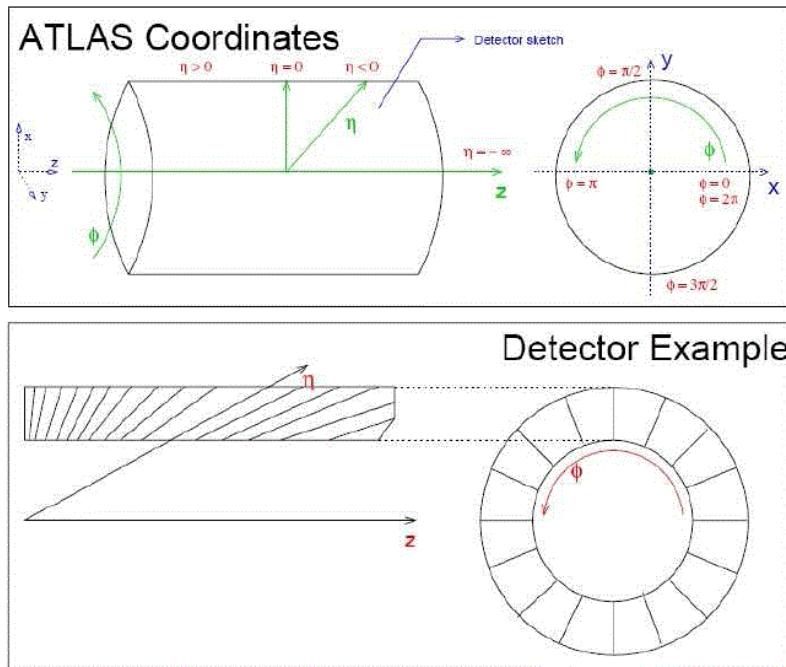


Figura A.1: O sistema de coordenadas do ATLAS.

Nota-se que a variável ϕ representa a rotação e a variável η (também chamada

de pseudo-rapidez) representa a direção de projeção das partículas, após a colisão.

Os valores dados das variáveis η e ϕ são apenas para referência do leitor. A variável ϕ , como é possível ver no canto direito da parte superior da figura, possui uma região em que dois valores são possíveis: 0 e 2π . Esta área é chamada de região *wrap-around*. Cálculos utilizando esta variável devem atentar para este fato.

Os detectores são simétricos, com relação ao eixo ϕ . A construção dos dispositivos é feita em gomos. Repara-se que quando alcança o eixo z , $\eta = 1$, isto significa que objetos com valores grandes em η representam colisões onde as partículas do feixe apenas se desviaram, não havendo, usualmente informações interessantes de análise pois representam choques elásticos. É comum utilizar-se detectores com baixa resolução quando $\eta > 3$.

Na parte inferior da Figura B.1, é possível ver um exemplo de como um detector genérico é segmentado, acompanhando as coordenadas η e ϕ , tanto para o barril, quanto para uma tampa.