UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

Fernando Henrique e Paula da Luz

Implementação do software MILC no estudo da
QCD completa

São Carlos
2010

FERNANDO HENRIQUE E PAULA DA LUZ

Implementação do software MILC no estudo da QCD completa.

Dissertação Apresentada ao Programa de Pós-Graduação em
Física do Instituto de Física de São Carlos da Universidade de
São Paulo para a obtenção do Título de Mestre em Ciências.

Área de Concentração: Física Computacional

Orientadora: Profa. Dra. Tereza Cristina da Rocha Mendes

São Carlos
2010

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pelo Serviço de Biblioteca e Informação IFSC/USP

Luz, Fernando Henrique e Paula da
Implementação do software MILC no estudo da QCD completa /
Fernando Henrique e Paula da Luz; orientador Tereza Cristina da
Rocha Mendes.-- São Carlos, 2009.
133 p.

Dissertação (Mestrado em Ciência - Área de concentração: Física
Computacional) – Instituto de Física de São Carlos da Universidade
de São Paulo.

1. MILC. 2. Programação paralela. 3. QCD. 4. Quarks. 5.
Formulação de rede I. Título.

A todos os membros da minha família, que sempre estiveram comigo em todos os momentos, e em especial à Júlia, por todo o carinho, dedicação, amor e compreensão gastos comigo desde o começo da nossa história.

AGRADECIMENTOS

À pesquisadora, amiga e orientadora Tereza Mendes, que durante todos estes anos colaborou de forma positiva com a minha formação científica e intelectual, sempre acreditando em mim em todos os momentos.

Ao pesquisador, amigo e co-orientador Gonzalo Travieso, pelos seus ensinamentos que se transformaram em valiosas ferramentas, na minha busca pelo conhecimento.

Ao Instituto de Física de São Carlos, por me acolher de maneira receptiva e calorosa durante toda a minha formação.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela concessão da bolsa de estudos.

Aos grandes amigos, *Flash* (Caio), *Kureck* (Eric), *Sombra* (Guto), que foram como irmãos durante todo o meu percurso na faculdade, ao *Chuck* (Henrique), pelo grande incentivo que me deu neste trabalho, ao *Adão* (Adalberto) pelas discussões construtivas sobre a ciência, ao He-Man (Diogo) pelo comentários sugeridos neste trabalho e ao Ph (Paulo) pela auxílio na finalização deste projeto.

A todos os amigos da CBF e da Al-Jazeera, que sempre contribuíram e acreditaram na realização deste projeto.

À minha Mãe e ao meu Pai por tudo que fizeram por mim. À minha família por todo o suporte.

Ao meu irmão Roberto, presente em todos os momentos, principalmente nos mais difíceis.

À minha amiga, namorada, noiva e pré-esposa *Linda* (Júlia), *por me fazer respirar, por me fazer existir...*

E a todos que, de alguma maneira ou outra, contribuíram para a elaboração deste trabalho.

RESUMO

LUZ, F. H. P. **Implementação do software MILC no estudo da QCD completa.** 2010. 133 p. Dissertação (Mestrado) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2010.

A CromoDinâmica Quântica (QCD) é a teoria quântica de campos que descreve as interações fortes entre quarks, que são os constituintes fundamentais das partículas do núcleo atômico. Devido ao caráter peculiar destas interações, o estudo da QCD não pode ser realizado pelos métodos usuais em teorias quânticas de campos, baseados em expansões perturbativas. O estudo não-perturbativo da QCD a partir de primeiros princípios torna-se possível através da formulação de rede da teoria, que equivale a um modelo de mecânica estatística clássica, para o qual podem ser realizadas simulações numéricas através de métodos de Monte Carlo. A área de simulações numéricas da QCD representa uma das maiores aplicações atuais da computação de alto desempenho, sendo realizada nos principais centros computacionais do mundo. As grandes exigências do trabalho de pesquisa nesta área contribuíram inclusive para o desenvolvimento de novas arquiteturas computacionais. O uso de processamento paralelo é vital nessas simulações, principalmente nos casos em que está envolvida a simulação da chamada QCD completa, onde se consideram os efeitos dos quarks dinâmicos. Vários pacotes contendo implementações de algoritmos para o estudo da QCD começam a ser disponibilizados por grupos de pesquisa na área. Nosso foco neste trabalho é voltado para o pacote MILC. Além de fazer uma descrição detalhada da forma de utilização deste pacote, realizamos aqui um acompanhamento da evolução dos métodos empregados, desde o Método de Monte Carlo aplicado no algoritmo de Metropolis até a elaboração do algoritmo RHMC, introduzido recentemente. Fazemos uma comparação de eficiência entre o RHMC e o algoritmo R, que foi o mais utilizado por décadas.

Palavras Chave: computação de alto desempenho, programação paralela, pacote MILC, QCD, formulação de rede

ABSTRACT

LUZ, F. H. P. **Implementation of the MILC package in the study of full QCD.** 2010. 133 p. Dissertation (Master) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Quantum ChromoDynamics (QCD) is the quantum field theory that describes the strong interactions between quarks, which are the fundamental constituents of particles in the atomic nucleus. Due to the peculiar characteristic of these interactions, the study of QCD cannot be carried out by usual methods in quantum field theory, which are based on perturbative expansions. The non-perturbative study of QCD from first principles becomes possible through the lattice formulation of the theory, which is equivalent to a classical statistical mechanics model, which in turn can be carried out by numerical simulations using Monte Carlo methods. The field of numerical simulations of QCD is one of the main applications of high performance computing, and is performed in most major computational centers around the world. The demanding requirements needed in this field led also to the development of new computational architectures. The use of parallel processing is vital in these types of simulations, especially in cases that involve what is known as full QCD, where the effects of dynamic quarks are taken into account. Several packages with algorithms implemented for the study of QCD have been recently made available by research groups in this field. The focus of this work is the MILC package. Here we make a detailed description of how to use this package and a follow up of the used methods, from the Monte Carlo method applied in the Metropolis algorithm up to the development of the RHMC algorithm, recently introduced. Comparisons are made between the efficiency of RHMC and the R algorithm, which was the most used in the past decades.

Keywords: high-performance computing, parallel programming, MILC package, QCD, lattice formulation

LISTA DE FIGURAS

Figura 1.1 – Representação do método científico por Popper	26
Figura 1.2 – Relação entre Teoria-Experimento-Simulação	32
Figura 2.1 – Representação de um nucleon	44
Figura 3.1 – Fluxograma do algoritmo Híbrido	70
Figura 3.2 – Fluxograma do algoritmo de Monte Carlo Híbrido	72
Figura 3.3 – Fluxograma do algoritmo de Monte Carlo Híbrido com Pseudo-Férmions	83
Figura 5.1 – Estrutura da instalação do pacote MILC	102

LISTA DE GRÁFICOS

Gráfico 6.1 – Análise dos tempos de execução dos algoritmos R e RMHC.	115
Gráfico 6.2 – Análise refinada do Gráfico 6.1.	115

LISTA DE TABELAS

Tabela 6.1 – Medida do tempo de execução por número de processos.	114
Tabela 6.2 – Análise dos valores médios para diferentes discretizações.	116

LISTA DE SIGLAS

QCD	Cromodinâmica Quântica(Quantum ChromoDynamics)
MILC	MIMD Lattice Computation
MIMD	Multiple Instruction Multiple Data
RHMC	Rational Hybrid Monte Carlo
HMC	Hybrid Monte Carlo (Monte Carlo Híbrido)
GNU	Gnu is Not Unix
QED	Quantum ElectroDynamics (EletroDinâmica Quântica)
KS	Kogut-Susskind
PC	Personal Computer (Computador Pessoal)
CPS	Columbia Physics System
XML	eXtensible Markup Language
CVS	Concurrent Version System
UKQCD	United Kingdom QCD
EPCC	Edinburgh Parallel Computing Centre
PHMC	Polinomial Hybrid Monte Carlo
SIMD	Single Instruction Multiple Data
HPF	High Performance Fortran
SMP	Symmetric MultiProcessing
NUMA	Non-Uniform Memory Access

MPI Message Passing Interface

POSIX Portable Operating System Interface

FSF Free Software Foundation

GMP GNU Multiple Precision Arithmetic Library

MPFR Multiple Precision Floating point with correct Rounding

DDHMC Domain-Decomposed Hybrid Monte Carlo

SUMÁRIO

1	Introdução	25
1.1	Simulações Numéricas e a Pesquisa Científica	25
1.2	Exemplos de Uso de Simulações Numéricas	30
1.3	Programação Paralela	31
1.4	Motivação	33
1.5	Apresentação	33
1.6	Organização do Texto	34
2	Revisão Conceitual	35
2.1	Ação na Mecânica Clássica	35
2.2	Ação na Mecânica Quântica	36
2.3	Rotação de Wick	39
2.4	QCD	43
2.5	QCD na Rede	44
2.6	Cromodinâmica Quântica (QCD) pura	46
2.7	QCD completa	47
2.8	Máquinas Utilizadas	48
2.9	Outros Softwares	49
2.10	Chroma	49
2.11	FermiQCD	49
2.12	Columbia Physics System	50
3	O Método de Monte Carlo	51
3.1	Introdução	51
3.2	O Princípio do Balanço Detalhado	55
3.3	Ergodicidade	57
3.4	Método de Metropolis	57
3.5	Algoritmo de Langevin	59

3.6	Método da Dinâmica Molecular	61
3.7	Algoritmo Híbrido	66
3.8	Algoritmo de Monte Carlo Híbrido	71
3.9	Método do Pseudo-Férmion	75
3.10	Aplicações do Método de Monte Carlo Híbrido	80
3.11	Algoritmo R	83
3.12	Algoritmo PHMC	87
3.13	Algoritmo RHMC	90
4	Computação Paralela	93
4.1	Modelos Computacionais Paralelos	93
4.2	Paralelismo de Dados	94
4.3	Memória Compartilhada	94
4.4	Threads	95
4.5	Passagem de Mensagem	96
4.6	Vantagens	96
4.7	Operações de Memória Remota	98
4.8	Combinação de Modelos	98
4.9	MPI	99
5	O pacote MILC	101
5.1	Descrição Geral	101
5.2	Forma de Utilização	103
6	Metodologia e Resultados	107
6.1	Geração dos Arquivos de Entrada	107
6.2	Aplicativo su3_hmd	108
6.3	Aplicativo su3_rhmc	109
6.4	Aplicativo Remez	109
6.5	Execução dos Casos	112
6.6	Resultados e Análises	114

7	Considerações Finais	119
7.1	Contribuições	119
A	Instalando o MILC	128
A.1	Procedimentos para a instalação	128
B	Instalando as Aplicações Necessárias	132
B.1	MPI	132
B.2	GMP	133
B.3	MPFR	133

Introdução

1.1 Simulações Numéricas e a Pesquisa Científica

Qual o papel dos métodos computacionais na realização da ciência?

Não há dúvida de que os computadores estão cada vez mais presentes em todos os níveis da produção científica atual (desde a análise de dados experimentais até a divulgação de novos resultados na forma de artigos científicos). É necessário, talvez, refletir um pouco mais até reconhecer que hoje em dia a simulação computacional tornou-se uma terceira abordagem para a ciência, ao lado da teoria e do experimento. A seguir, fazemos algumas considerações sobre a natureza da ciência do ponto de vista histórico, procurando enfatizar a evolução da metodologia científica até os dias de hoje.

Começamos essa nossa consideração no século passado, onde acirrou-se um debate sobre as características que determinavam um conhecimento como científico. Uma escola que remonta a longa data, desde os tempos de Francis Bacon, no século XVI, defendia o critério de demarcação que seria capaz de dividir o conhecimento entre *científico* e *não científico*. Era justamente o **método científico**. Esse método era conhecido como método indutivo, e foi adaptado e divulgado pelos positivistas, principalmente no século XIX. Em suma, ele consistia na repetição exaustiva de observações ou experimentos, com pequenas alterações em características básicas na tentativa de garantir a veracidade das

conclusões. Podemos citar como exemplo, que após realizar muitos experimentos variando pressão, temperatura, etc, concluímos que uma lei da natureza seria: *“a água sempre ferve a 100 °C ao nível do mar”*. Ou seja, da observação extenuante da natureza induzimos uma lei; dessa forma, a natureza acabaria sendo a origem do conhecimento.

Contudo, desde as críticas de David Hume, por volta de 1700, tal método já mostrava não ser capaz de garantir a veracidade das – assim chamadas – leis naturais.

Para mostrar alguns episódios ocorridos no século passado, explicitaremos sucintamente duas concepções distintas de ciência, bastante exemplares de escolas e em até certo ponto pensamentos concorrentes: as ideias de Karl Popper (1902-1994) e as de Thomas Kuhn (1922-1996).

Karl Popper acreditava na existência de um método científico. Tal método serviria para guiar o trabalho dos cientistas, sendo representado pelas seguintes etapas (1)

Figura 1.1 – Representação do método científico por Popper

1. Existência de um fenômeno a ser estudado;
2. Procura de modelos para a explicação do fenômeno através da elaboração de várias hipóteses tentativas, e a escolha de uma delas através do critério de aceitar aquela que apresenta maior grau de possibilidades de refutação;
3. Dedução de consequências dessa hipótese;

4. Critério de refutabilidade em ação: a hipótese é testada, isto é, procura-se refutá-la buscando contra-exemplos significativos;
5. Passando por esse teste, isto é, na ausência de refutação, a hipótese se transforma na nova teoria;
6. No caso de uma descoberta refutadora ou de uma dedução não confirmada, retorna-se ao estágio inicial.

Para Popper, o critério de demarcação entre ciência e não-ciência é demonstrado desta forma. Tal método, conhecido como método dedutivo, se diferencia do método indutivo em alguns pontos.

Em primeiro lugar, ao contrário do que ocorre com o cientista na concepção indutivista (que se submete à natureza), o cientista na concepção dedutivista possui a liberdade de criação. Ele pode criar quantas hipóteses quiser, desde que tais hipóteses permitissem experimentos capazes de corroborar ou refutar a teoria. Desta forma, verificamos uma outra diferença: enquanto para o método indutivo a natureza seria a fonte do conhecimento, no método dedutivo a natureza seria a juíza das hipóteses elaboradas pelo cientista. Porém, ainda que as ideias de Popper solucionassem alguns problemas irremediáveis do indutivismo, outros filósofos discordavam de sua teoria.

De fato, Gaston Bachelard (1884-1962) afirmava que um mesmo cientista utilizaria diferentes métodos de acordo com seu grau de conhecimento do assunto por ele tratado, ou seja, no início do estudo de uma nova área, os cientistas utilizariam um método indutivo; com o amadurecimento desta área do conhecimento, o método dedutivo ganharia terreno, e assim sucessivamente, passando por outros modelos de métodos científicos.

Após verificar estas ideias de metodologia científica, podemos constatar que existe um pressuposto nas mesmas: os cientistas seguiriam determinados passos durante seus trabalhos. Dessa maneira, podemos afirmar que tais métodos são prescritivos pois, assim como uma bula de remédio, prescrevem os passos futuros dos usuários.

Por outro lado, seguindo uma direção ortogonal, estão as ideias de Thomas Kuhn.

Estudando história da ciência, ele verificou que os cientistas nem sempre seguiam determinados métodos, ou seja, eles não trabalhavam de acordo com etapas existentes em alguma tabuleta prescritiva pregada na parede do laboratório. Podemos assim definir as ideias de Kuhn como descritivas, pois descrevem aquilo que ele vislumbrou ao estudar o passado.

As ideias de Kuhn se aproximam muito mais de um estudo sociológico ou antropológico. Para ele, a comunidade científica se portaria de maneira distinta em dois momentos diferentes da ciência: A *Ciência Normal* e as *Revoluções Científicas*.

A Ciência Normal determina o período onde a comunidade científica trabalha para reafirmar um paradigma em vigor. Tal paradigma abrange a teoria dominante na explicação de um determinado conjunto de fenômenos, além de todo o aparato técnico, a lógica, a matemática e tudo que fosse utilizado na tentativa de mostrar a eficácia do paradigma em vigor. Dessa forma, um cientista que não conseguisse estabelecer um paralelo entre as teorias e os experimentos, nunca duvidaria do paradigma e sim da sua própria capacidade de manusear o mesmo. Com o tempo, o paradigma seria articulado e melhorado permitindo certo progresso das ideias defendidas.

Contudo, eventuais falhas podem se mostrar resistentes a tentativas de articulação do paradigma. Com a persistência dessas falhas, elas poderiam levar a uma crise do paradigma, permitindo que alguns cientistas (geralmente novos no estudo da área em questão) comessem a duvidar do paradigma em vigor. Neste período, se instalaria uma crise. Porém, o paradigma em vigor seria mantido até que outro candidato a paradigma aparecesse, mostrando-se capaz de resolver os problemas que o paradigma anterior possuía dificuldades em resolver.

Pode-se considerar que ocorre uma mudança do paradigma quando os cientistas da geração anterior gradativamente começam a perder destaque no cenário científico atual para uma nova geração que gradativamente começa a assumir a produção do conhecimento científico. Essa mudança de paradigma seria conhecida como Revolução Científica.

Voltemos, agora, a nossa questão inicial: qual o papel dos métodos computacionais na

realização da ciência? Verificamos que essa questão pode receber respostas diferentes de acordo com a concepção de ciência adotada.

Um popperiano poderia afirmar que os métodos computacionais seriam muito eficazes no momento de deduzir as consequências de uma hipótese, facilitando encontrar as conclusões a serem comparadas com os experimentos a cargo de corroboração ou refutação das hipóteses.

Já um kuhniano poderia comparar os métodos computacionais aos experimentos mentais, famosos na produção científica de Galileu e Einstein. Para tanto, poderíamos imaginar diferentes graus de complexidade desses experimentos mentais, pois um computador permite a simulação de situações muito complexas, como por exemplo a modelagem de uma nebulosa planetária utilizando mais de 10 equações diferenciais relativas a hidrodinâmica, a atração gravitacional, etc, resolvendo todas as equações concomitantemente.

De qualquer modo, a possibilidade de utilizar o computador não apenas como um instrumento para fazer contas complicadas, mas principalmente para produzir simulações numéricas, nos induz a repensar a maneira como a ciência é produzida. Em particular, em todas as visões descritas acima, a condição final para se aceitar ou recusar uma teoria é dada pelo experimento, ou seja, pela natureza. Por outro lado, a simulação numérica se apresenta como “experimento teórico”, que pode servir para vários fins, como por exemplo:

1. “Simular” a teoria, ou seja, permitir que aspectos inacessíveis de uma teoria sejam explorados a partir da solução explícita de suas equações, ou de técnicas estatísticas. Este é o caso da simulação da QCD abordada nesta dissertação.
2. “Simular” o experimento, ou seja, processar possíveis resultados de experimentos (tipicamente em física de partículas, onde é gerada uma grande quantidade de dados) utilizando resultados teóricos esperados, para auxiliar na análise dos dados. Esta foi a aplicação original do Método de Monte Carlo. Desta forma são testadas novas teorias utilizando resultados teóricos já estabelecidos.
3. Simulação de sistemas para os quais não há uma teoria aceita ou conhecida. É o

caso dos chamados Sistemas Complexos, em que características universais (como a invariância de escala ou auto-similaridade) encontradas no comportamento de diversos sistemas físicos são reproduzidas a partir de modelos simples, como autômatos celulares. Exemplos são o padrão de conchas, terremotos, avalanches, organização de redes complexas como a internet, etc.

4. Visualização para melhor entendimento dos resultados produzidos, e.g. simulações em linhas de amarração em projetos na engenharia naval, simuladores de voo e simulações de experimentos com intuito didático. Neste caso a teoria usada é já aceita e estabelecida e são criadas “animações” que fazem o papel de experimentos.

Nos casos de 1 e 2, cientistas que utilizam os métodos computacionais não pretendem refutar o paradigma com o qual trabalham, e sim ampliar os horizontes de aplicação das teorias em vigor. Claramente será especialmente importante este trabalho quando já se sabe que a teoria não é completa, como as investigações atuais sobre o Modelo Padrão das Partículas Elementares no acelerador LHC no CERN. Para encontrar sinais da chamada “nova física” (i.e. física além do Modelo Padrão) é preciso ter controle sobre as previsões do Modelo Padrão, que inclui a QCD. No caso 3, temos a simulação como guia para a possível introdução de um novo paradigma (2). Em 4 temos um exemplo de laboratório virtual, em que a simulação é utilizada como substituto do experimento.

Podemos portanto concluir que a simulação numérica ocupa a posição de terceiro modo de fazer ciência (além de teoria e experimento), conforme é argumentado em (3).

1.2 Exemplos de Uso de Simulações Numéricas

O uso das simulações numéricas está presente nas mais diversas áreas. Complementando os exemplos acima, podemos citar

- simular a propagação de fogo em uma floresta através de um modelo de autômato

celular,

- realizar previsões meteorológicas com alguns dias de antecedência,
- verificar a condição necessária para uma síntese correta de um aminoácido presente em um fármaco,
- prever as formas com que ocorrem as interações entre as Partículas Elementares,
- elaborar a forma correta de fixar uma plataforma em alto mar para a extração de petróleo,
- reproduzir fenômenos físicos com caráter didático como forma de facilitar a divulgação científica, que assim requer menos recursos que os laboratórios didáticos usuais, permitindo o acesso a experimentos sem a necessidade de aparatos físicos para qualquer pessoa com acesso a um computador.

Vale lembrar que alguns dos motivos para a utilização de simulações computacionais ao invés de experimentos são o custo elevado para a realização do experimento, a falta de instrumentos que permitam a reprodução de certos fenômenos, ou a possibilidade de evitar a realização de experimentos perigosos, entre outros.

1.3 Programação Paralela

Apesar de nos dias atuais os computadores serem centenas de vezes mais rápidos do que há uma década, as diversas áreas de pesquisa precisam continuamente de recursos computacionais, pois em máquinas mais rápidas os resultados podem ser obtidos com uma maior precisão (gerando uma reprodução dos resultados mais precisa ao modelo a ser simulado) e/ou em um tempo menor de simulação.

Esperar um determinado tempo para ter um ganho na capacidade de cálculo dos computadores (pela lei de Moore, o número de transistores inseridos na pastilha dobraria a cada 1,5 à 2 anos) acaba sendo inviável, pois o desejável é obter resultados para os experimentos o mais breve possível. Portanto, para conseguir esse ganho, a opção é a utilização

Figura 1.2 – Relação entre Teoria, Experimento e Simulação – adaptação de Landau e Binder (3), p. 5.

da computação paralela (4), que permite obter de imediato esse poder computacional para a resolução de problemas. Outro fato a ressaltar sobre a Lei de Moore, é que as atuais pastilhas os transistores adicionais estão sendo utilizados na implementação de tecnologia multicore, sendo mais um motivo para a aplicação de técnicas de computação paralela.

A computação paralela é utilizada nas mais diversas áreas, que necessitam de grande poder computacional, tais como as simulações de clima para previsões do tempo, a análise de dados coletados de telescópios para o estudo de questões cosmológicas, a elaboração de circuitos e componentes eletrônicos para serem utilizados na fabricação de computadores mais rápidos, a redistribuição de rotas mais coerentes para os pacotes da internet, o enovelamento de proteínas e também o estudo não-perturbativo das teorias de gauge que descrevem as interações fundamentais entre as Partículas Elementares, através do estudo das propriedades da QCD através de uma formulação na rede.

Vale ressaltar que a programação paralela possui diversos aspectos diferentes da programação sequencial, tais como a distribuição da carga de cálculos e dados, a comunicação e sincronização efetuada entre os nós, entre outros fatores que influenciam significativamente o tempo de cálculo do programa.

1.4 Motivação

Como já foi mencionado, os principais motivos para a utilização da computação paralela são a diminuição do tempo gasto nas simulações e a inclusão de um maior detalhamento resultando em uma melhor representação do fenômeno físico modelado.

A QCD é uma teoria que não pode em geral ser estudada pelos métodos analíticos usuais de teorias de campo, pois estes são baseados em teoria de perturbação. Dessa forma, o estudo se torna viável somente através de uma formulação de rede da teoria, ainda que necessite de grande recurso computacional.

Um pacote de aplicações de simulações em paralelo para a QCD foi desenvolvido pela MIMD Lattice Computation (MILC). Tais aplicações estão disponíveis através de licença GNU embora não seja um pacote comercial ou de uso simples como uma caixa preta, são utilizadas como aplicações padrão por vários grupos de pesquisa da área. Fazer uma utilização deste pacote e verificar alguns detalhes da sua implementação é o foco principal desta dissertação.

1.5 Apresentação

Esta pesquisa foi desenvolvida no Instituto de Física de São Carlos, um dos institutos pioneiros na utilização de simulações numéricas em larga escala no país e onde recentemente foi criado o primeiro curso brasileiro de Bacharelado em Física Computacional. O trabalho foi desenvolvido em conjunto com uma especialista em QCD na Rede, a Profa. Dra. Tereza Mendes, e com um especialista na área de Programação Paralela, o Prof. Dr. Gonzalo Travieso.

1.6 Organização do Texto

A dissertação foi elaborada da seguinte forma: O Capítulo 2 apresenta uma breve revisão conceitual da QCD para um entendimento dos fenômenos nas simulações dos algoritmos que foram estudados e estão descritos no Capítulo 3. Uma descrição sucinta das diversas implementações paralelas possíveis é fornecida no Capítulo 4. O funcionamento do pacote MILC e a forma utilizada para estudar as implementações do Algoritmo R e Rational Hybrid Monte Carlo (RHMC) serão apresentados no Capítulo 5. A metodologia e os resultados obtidos são mostrados no Capítulo 6 e as conclusões e trabalhos futuros no Capítulo 7.

Revisão Conceitual

2.1 Ação na Mecânica Clássica

As equações de Lagrange na Mecânica Clássica são utilizadas para representar as equações de movimento de um forma mais conveniente, onde ao invés de se trabalhar com vetores (forças e acelerações) utilizamos duas funções escalares (Energia Cinética e Potencial). Estas equações podem ser obtidas pelo princípio de D’Alambert (conhecido como “princípio diferencial”) ou pelo princípio de Hamilton (também denominado “princípio variacional”). Utilizamos como referências (5, 6).

Vamos imaginar o seguinte caso onde queremos considerar o movimento de um sistema entre o tempo t_1 e t_2 . A configuração instantânea de um sistema é descrita pelos valores das suas n coordenadas generalizadas $(q_1, \dots, q_n)$ que correspondem a um particular ponto em um hiperespaço cartesiano que possui n eixos de coordenadas (conhecido como espaço de configurações) e o movimento citado acima refere-se à curva que é gerada pelo ponto a cada instante de tempo, desde t_1 até t_2 .

O princípio integral de Hamilton que descreve o movimento de sistemas mecânicos para todas as suas forças (exceto as de vínculo) é derivado de um potencial escalar generalizado que pode ser uma função das coordenadas, velocidades e tempo.

Podemos enunciá-lo como sendo o movimento de um sistema do tempo t_1 ao tempo

t_2 determinado pela integral de linha (chamada de ação ou integral de ação),

$$I = \int_{t_1}^{t_2} L dt, \quad (2.1)$$

onde $L = T - V$ (diferença entre as energias Cinética e Potencial).

É claro que existem diversas trajetórias possíveis para o sistema evoluir de um ponto a outro, mas a representação (clássica) do movimento físico acontece apenas no caminho chamado estacionário, onde a integral de linha possui o mesmo valor mesmo quando aplicamos variações de primeira ordem em toda a vizinhança da sua trajetória. Pode-se resumir o princípio de Hamilton dizendo que o movimento é tal que a “variação” na integral de linha I para um t_1 e t_2 fixos é zero,

$$\delta I = \delta \int_{t_1}^{t_2} L(q_1, \dots, q_n, \dot{q}_1, \dots, \dot{q}_n, t) dt = 0. \quad (2.2)$$

Tal formulação foi desenvolvida no âmbito do tratamento de sistemas com um número discreto de graus de liberdade, porém existem inúmeros sistemas físicos que são elaborados em meios contínuos. Felizmente, a formulação hamiltoniana é facilmente generalizada se lidarmos com integrais por todos os graus de liberdade do sistema, analogamente à soma sob os graus de liberdade do caso discreto. O tratamento para campos clássicos foi de extrema importância na formulação para o tratamento de sistemas quânticos.

2.2 Ação na Mecânica Quântica

Além da equação de Schrödinger e da mecânica de matrizes de Heisenberg (7), a ação na teoria quântica surgiu como uma terceira formulação para a descrição de forma não-relativística para a mecânica quântica, sendo inicialmente proposta por Dirac, que tentou estabelecer a relação da ação clássica para a mecânica quântica. Posteriormente, esta tentativa ganhou um embasamento matemático por Feynman, através do artigo (8). Em nossa descrição a seguir utilizaremos as fontes (9, 7).

Na formulação de Hilbert da Mecânica Quântica, os estados de um sistema são descritos por vetores no chamado espaço de Hilbert, e os observáveis são representados por operadores hermitianos agindo dentro desse espaço. A evolução temporal do sistema quântico é dada pela equação de Schrödinger, ou equivalentemente por

$$|\psi(t')\rangle = e^{-i \frac{H(t'-t)}{\hbar}} |\psi(t)\rangle, \quad (2.3)$$

onde H é o Hamiltoniano. Portanto se soubermos o estado de um sistema no tempo t , podemos determinar o estado em um tempo posterior t' . Vamos tomar $q = q_\alpha$ para representar coletivamente os graus de liberdade de um sistema e $|q\rangle$ os autovetores correspondentes aos operadores Q_α , isto é,

$$Q_\alpha |q\rangle = q_\alpha |q\rangle \quad \alpha = 1, \dots, n. \quad (2.4)$$

A partir de 2.3, pode-se encontrar a função de onda $\psi(q', t')$

$$\psi(q', t') = \langle q' | \psi(t') \rangle. \quad (2.5)$$

De fato, substituindo 2.5 em 2.3 e inserindo a relação de completeza

$$\int Dq |q\rangle \langle q| = \mathbf{1} \quad (2.6)$$

entre $e^{-i \frac{H(t'-t)}{\hbar}}$ e $|\psi(t)\rangle$, nós obtemos

$$\begin{aligned} \psi(q', t') &= \int Dq \langle q' | e^{-i \frac{H(t'-t)}{\hbar}} |q\rangle \langle q | \psi(t) \rangle \\ &= \int Dq G(q', t'; q, t) \psi(q, t) \end{aligned} \quad (2.7)$$

onde

$$G(q', t'; q, t) \equiv \langle q' | e^{-i \frac{H(t'-t)}{\hbar}} |q\rangle \quad (2.8)$$

é a função de Green que descreve a propagação do estado $|\psi(t)\rangle$, e onde a medida da integração usada é dada por

$$Dq = \prod_{\alpha=1}^n dq_\alpha, \quad (2.9)$$

sendo n o número de graus de liberdade do sistema.

Uma importante propriedade da função de Green em 2.8 é o fato de que ela satisfaz a seguinte lei de composição

$$G(t', q'; q, t) = \int dq'' G(t', q'; q'', t'') G(q'', t''; q, t), \quad (2.10)$$

obtida através de

$$\exp\left(-i\frac{H(t' - t)}{\hbar}\right) = \exp\left(-i\frac{H(t' - t'')}{\hbar}\right) \exp\left(-i\frac{H(t'' - t)}{\hbar}\right), \quad (2.11)$$

e introduzindo um conjunto completo de estados intermediários $|q''\rangle$ entre as duas exponenciais. Utilizando a propriedade 2.11, Feynman obteve uma representação para a integral de trajetória para um elemento de matriz 2.10, que mostra de uma forma bem transparente a conexão entre a teoria clássica e quântica. Na física clássica a evolução temporal de um sistema é dado pelas equações de movimento de Lagrange que seguem o princípio de mínima ação. Para quantizar o sistema, devemos construir o Hamiltoniano e escrever a equação de movimento em função dos colchetes de Poisson. Isto providencia o ponto de início para a quantização canônica da teoria. A representação da integral de trajetória de Feynman estabelece a conexão com o princípio clássico de ação.

Vamos considerar que o Hamiltoniano H não depende explicitamente do tempo e chamaremos $\langle\phi_n|$ e E_n como os seus autoestados e autovalores, respectivamente. Assim temos

$$H|\phi_n\rangle = E_n|\phi_n\rangle \quad (2.12)$$

e será usada a forma discreta da relação de completeza,

$$\sum_n |\phi_n\rangle \langle\phi_n| = \mathbb{1}, \quad (2.13)$$

o que nos permite escrever

$$\langle q', t' | q, t \rangle = \langle q' | e^{-i\frac{H(t' - t)}{\hbar}} | q \rangle \quad (2.14)$$

da seguinte forma

$$\langle q', t' | q, t \rangle = \langle q' | e^{-i\frac{H(t' - t)}{\hbar}} \sum_n |\phi_n\rangle \langle\phi_n| q \rangle. \quad (2.15)$$

Utilizando 2.12 obtemos

$$\langle q', t' | q, t \rangle = \sum_n e^{-i\frac{E_n(t' - t)}{\hbar}} \langle q' | \phi_n \rangle \langle \phi_n | q \rangle, \quad (2.16)$$

onde

$$\langle q' | \phi_n \rangle = \phi_n(q') \quad (2.17)$$

$$\langle \phi_n | q \rangle = \phi_n^*(q). \quad (2.18)$$

Desta forma podemos re-escrever 2.16 como

$$\langle q', t' | q, t \rangle = \sum_n e^{-i \frac{E_n(t'-t)}{\hbar}} \phi_n(q') \phi_n^*(q). \quad (2.19)$$

Como já foi dito, a somatória sobre n pode ser estendida para o espectro contínuo do Hamiltoniano. Para prosseguir com os cálculos, faremos uso da rotação de Wick.

2.3 Rotação de Wick

A rotação de Wick é uma técnica que considera a energia ou o tempo imaginário, transformando o espaço de Minkowski em um espaço euclidiano, facilitando dessa forma os cálculos, como veremos a seguir. Temos que

$$ds^2 = -(dt^2) + dx^2 + dy^2 + dz^2 \quad (\text{Espaço de Minkowsky}) \quad (2.20)$$

e

$$ds^2 = d\tau^2 + dx^2 + dy^2 + dz^2 \quad (\text{Espaço Euclidiano}) \quad (2.21)$$

são equivalentes se $it = \tau$, onde após realizar os cálculos nesta nova coordenada pode-se retomar o valor correto dos cálculos realizando uma substituição reversa. A rotação de Wick permite executar um tratamento estatístico clássico para a Mecânica Quântica.

Iremos continuar com a expressão 2.19, fazendo a substituição de $t \rightarrow -i\tau$ e $t' \rightarrow -i\tau'$. Teremos assim

$$\langle q', t' | q, t \rangle_{\substack{t = -i\tau \\ t' = -i\tau'}} = \sum_n e^{\frac{E_n(\tau' - \tau)}{\hbar}} \psi_n(q') \psi_n^*(q). \quad (2.22)$$

O lado direito da equação é apenas o elemento de matriz $\langle q' | \exp(-\frac{H(\tau' - \tau)}{\hbar}) | q \rangle$. Então, como é esperado de 2.14, a função de Green em termos do tempo imaginário é dada por

$$\begin{aligned} \langle q', t' | q, t \rangle_{t = -i\tau}^{t' = -i\tau'} &= \langle q' | e^{-\frac{H(\tau' - \tau)}{\hbar}} | q \rangle. \end{aligned} \quad (2.23)$$

Para se chegar à representação da integral de trajetória para o lado direito de 2.23, nós dividimos o intervalo temporal $[\tau, \tau']$ em N segmentos infinitesimais de comprimento $\epsilon = (\tau' - \tau)/N$. Denotaremos por $\tau_1, \tau_2, \dots, \tau_{N-1}$ os tempos intermediários. Então o tempo imaginário da Função de Green pode ser obtido pela sequência de passos temporais intermediários como se segue

$$\begin{aligned} \langle q' | e^{-\frac{H(\tau' - \tau)}{\hbar}} | q \rangle &= \langle q' | e^{-\frac{H(\tau' - \tau_{N-1})}{\hbar}} e^{-\frac{H(\tau_{N-1} - \tau_{N-2})}{\hbar}} \dots e^{-\frac{H(\tau_1 - \tau)}{\hbar}} | q \rangle \\ &= \int \prod_{l=1}^{N-1} Dq^l \langle q' | e^{-\frac{H\epsilon}{\hbar}} | q^{N-1} \rangle \langle q^{N-1} | e^{-\frac{H\epsilon}{\hbar}} | q^{N-2} \rangle \dots \langle q^1 | e^{-\frac{H\epsilon}{\hbar}} | q \rangle, \end{aligned} \quad (2.24)$$

onde

$$Dq^l = \prod_{\alpha} dq_{\alpha}^{(l)}, \quad (2.25)$$

sendo que $|q^{(l)}\rangle$ representa o conjunto completo de autoestados que são introduzidos em um l -ésimo passo temporal.

Para solucionar os elementos de matriz em 2.24, vamos especificar uma estrutura do Hamiltoniano. Vamos supor a forma

$$H = \frac{1}{2} \sum_{\alpha=1}^n P_{\alpha}^2 + V(Q), \quad (2.26)$$

onde P_{α} são os momentos canonicamente conjugados a Q_{α} . temos que $\exp(\frac{-H\epsilon}{\hbar})$ pode ser aproximado (considerando ϵ pequeno) como

$$e^{-\frac{H\epsilon}{\hbar}} \approx e^{-\epsilon \frac{1}{2} \sum_{\alpha} \frac{P_{\alpha}^2}{\hbar}} e^{-\epsilon \frac{V(Q)}{\hbar}}. \quad (2.27)$$

Isto nos leva a

$$\langle q^{(l+1)} | e^{-\frac{H\epsilon}{\hbar}} | q^{(l)} \rangle \approx \langle q^{(l+1)} | e^{-\epsilon \frac{1}{2} \sum_{\alpha} \frac{P_{\alpha}^2}{\hbar}} | q^{(l)} \rangle e^{-\epsilon \frac{V(q^{(l)})}{\hbar}}. \quad (2.28)$$

Para resolver os elementos de matriz que sobraram, nós introduzimos um conjunto completo de autoestados de momento do lado direito e esquerdo de $\exp(-\epsilon \sum_{\alpha} P_{\alpha}^2)$. Com

$$\langle p | q \rangle = \prod_{\alpha=1}^n \frac{1}{\sqrt{2\pi}} e^{ip_{\alpha} q_{\alpha}}, \quad (2.29)$$

nós temos que

$$\langle q^{(l+1)} | e^{-\frac{H\epsilon}{\hbar}} | q^{(l)} \rangle \approx e^{-\frac{V(q^{(l)})\epsilon}{\hbar}} \int \prod_{\beta=1}^n \frac{dp_{\beta}^{(l)}}{2\pi} \prod_{\alpha=1}^n \exp \left\{ -\epsilon \left[\frac{1}{2} p_{\alpha}^{(l)2} - i p_{\alpha}^{(l)} \left(\frac{q_{\alpha}^{(l+1)} - q_{\alpha}^{(l)}}{\epsilon} \right) \right] \right\} \quad (2.30)$$

A integral gaussiana pode ser resolvida e obtemos

$$\langle q^{(l+1)} | e^{-\frac{H\epsilon}{\hbar}} | q^{(l)} \rangle \approx \left(\frac{1}{\sqrt{2\pi\epsilon}} \right)^n \exp \left\{ -\epsilon \left[\sum_{\alpha=1}^n \frac{1}{2} \left(\frac{q_{\alpha}^{(l+1)} - q_{\alpha}^{(l)}}{\epsilon} \right)^2 \right] \right\}. \quad (2.31)$$

Substituindo esse resultado em 2.24, nós aproximamos para a seguinte fórmula válida para ϵ pequeno,

$$\langle q' | e^{-\frac{H(\tau' - \tau)}{\hbar}} | q \rangle \approx \left(\frac{1}{\sqrt{2\pi\epsilon}} \right)^{nN} \int \prod_{l'=1}^{N-1} dq^{(l')} e^{-\sum_{l=0}^{N-1} \epsilon L_E(q^{(l)}, \dot{q}^{(l)})}, \quad (2.32)$$

onde

$$L_E(q^{(l)}, \dot{q}^{(l)}) = \sum_{\alpha} \frac{1}{2} \dot{q}_{\alpha}^{(l)2} + V(q^{(l)}), \quad (2.33)$$

$$\dot{q}_{\alpha}^{(l)} \equiv \frac{q_{\alpha}^{(l+1)} - q_{\alpha}^{(l)}}{\epsilon}, \quad (2.34)$$

e $q^{(0)} \equiv q$, e $q^{(N)} \equiv q'$. O índice E em L_E serve pra evidenciar que estamos usando na função de Green a formulação euclidiana. Vamos agora interpretar os resultados obtidos em 2.32. Consideremos um caminho arbitrário conectando os pontos espaço-temporais (q, τ) e (q', τ') , consistindo de segmentos de linhas em todos os intervalos temporais infinitesimais. Seja $q^{(l)}$ o conjunto de coordenadas do sistema no tempo τ_l , e para enfatizar vamos fazer

$$q_{\alpha}^{(l)} = q_{\alpha}(\tau_l), \quad (2.35)$$

onde $\tau_0 \equiv \tau$, $\tau_N \equiv \tau'$. Então 2.34 é a “velocidade euclidiana” em um intervalo de tempo $[\tau_l, \tau_{l+1}]$ de uma partícula se movendo em um espaço de configurações n -dimensional, e L_E é a versão discretizada da Lagrangiana clássica na formulação euclidiana. A ação associada à trajetória é dada por

$$S_E[q] = \sum_{l=0}^{N-1} \epsilon \left[\sum_{\alpha} \frac{1}{2} (\dot{q}_{\alpha}(\tau_l))^2 + V(q(\tau_l)) \right]. \quad (2.36)$$

Essa expressão aparece no argumento da exponencial em 2.32. Temos assim o seguinte algoritmo para a resolução da função de Green para um tempo imaginário.

1. Divida o intervalo de tempo $[\tau, \tau']$ em segmentos de comprimento $\epsilon = (\tau' - \tau)/N$.
2. Considere todas as possíveis trajetórias iniciando do ponto q no tempo τ e que termine no ponto q' no tempo τ' . Aproxime essas trajetórias por segmentos retos, e calcule a ação 2.36 em cada caminho.
3. Pesando cada trajetória com $\exp(-S_E[q])$, some essas exponenciais sobre todos os possíveis caminhos, integrando sobre todas as trajetórias possíveis das coordenadas nos tempos intermediários.
4. Multiplique a expressão resultante por $(1/\sqrt{2\pi\epsilon})^{nN}$, onde n é o número de graus de liberdade e tome o limite $\epsilon \rightarrow 0$, $N \rightarrow \infty$, mantendo o produto $N\epsilon = (\tau' - \tau)$ fixo.

O resultado do algoritmo acima é descrito por

$$(q', \tau' | q, \tau) = \int_q^{q'} Dq e^{-\frac{S_E[q]}{\hbar}}, \quad (2.37)$$

onde

$$S_E[q] = \int_{\tau}^{\tau'} d\tau'' L_E(q(\tau''), \dot{q}(\tau'')). \quad (2.38)$$

Por conveniência, podemos escrever 2.37 da seguinte forma

$$(q', \tau' | q, \tau) \equiv \langle q' | e^{-\frac{H(\tau' - \tau)}{\hbar}} | q \rangle, \quad (2.39)$$

que é uma analogia com 2.14. Esta acaba se tornando a expressão para a integral de trajetória que desejamos obter. Podemos perceber que, entre os caminhos a serem medidos por $\exp(-S_E)$, contribuições importantes em 2.37 esperamos quando $S_E[q]$ obtém valores próximos do mínimo, ou seja

$$\delta S_E[q] = 0. \quad (2.40)$$

Este é, no limite de $\hbar \rightarrow 0$, o princípio de mínima ação que nos remete as equações do movimento euclidianas clássicas. Então, dentro da estrutura da integral de trajetória, a quantização de um sistema clássico nos leva a considerar flutuações em torno da trajetória clássica. Na formulação euclidiana essas flutuações são suprimidas exponencialmente se $S_E \geq 0$. Por outro lado, na formulação com o tempo real, um procedimento análogo nos

leva à representação da integral de trajetória de 2.14

$$\langle q' | e^{-\frac{H(t'-t)}{\hbar}} | q \rangle = \int_q^{q'} Dq e^{\frac{iS[q]}{\hbar}}, \quad (2.41)$$

onde $S[q]$ é a ação no tempo real. A integral de trajetória em 2.41 é definida da mesma forma que a expressão 2.37 acima, mas agora as trajetórias são “pesadas” por uma função oscilante. Por esta razão esta representação de integral de trajetória não é utilizada para cálculos numéricos.

Uma solução exata da integral de trajetória 2.41 ou 2.39 só é possível para poucos casos.

Por outro lado, o tratamento descrito aqui para o caso da mecânica quântica não-relativística pode ser estendido de maneira simples para quantização de teorias quânticas de campo, como a QCD, descrita a seguir. Em particular a ação da QCD é considerada em detalhe no Capítulo 3 (3.9

2.4 QCD

A Cromodinâmica Quântica (QCD, do inglês Cromodinâmica Quântica) é a teoria quântica de campos utilizada na descrição da interação forte, que explica desde o motivo do agrupamento dos constituintes do núcleo dos átomos até a forma como ocorre a queima do combustível das estrelas (9).

Esta teoria é formulada como um modelo de Partículas Elementares – os quarks, que possuem “carga de cor” e interagem por troca de campos de gauge – os glúons. A QCD é baseada em uma teoria quântica de campos, com simetria local de gauge $SU(3)$, correspondendo a 3 possíveis escolhas para a chamada “carga de cor”, (r - red), (g - green) e (b - blue). A forma da descrição da QCD é simples e elegante. Seus parâmetros são as massas dos vários tipos de quarks considerados, que recebem o nome de “sabores” (up, down, charm, strange, bottom e top) e o valor da constante de acoplamento forte.

Pode-se observar na figura 2.1 como ocorrem essas interações dentro de nucleon. Se não

Figura 2.1 – Representação de um nucleon sob o olhar da QCD. Fonte: Lawrence Berkeley National Laboratory. Disponível em: <http://www.lbl.gov/Science-Articles/Archive/sabl/2006/Oct/3.html>. Acesso em: 14 Out 2008.

considerarmos a simetria sob o grupo de gauge $SU(3)$, a forma da lagrangiana da QCD é a mesma da Quantum ElectroDynamics (QED), onde o grupo de simetria na QED é o $U(1)$, e os quarks correspondem aos elétrons e os glúons aos fótons. (Sendo os dois primeiros férmions de spin $1/2$ e os dois últimos bósons vetoriais sem massa.) De maneira análoga, a constante de acoplamento forte α_S tem correspondência com a constante de estrutura fina $\alpha \approx 1/137$. O ponto central é que o grupo na QCD não é abeliano, o que leva a diferenças qualitativas entre a QCD e a QED, que descrevem respectivamente as interações fortes e as interações eletromagnéticas. A principal diferença é que os glúons possuem carga de cor, e dessa forma interagem entre si, ao contrário dos fótons.

2.5 QCD na Rede

A liberdade assintótica é a característica segundo a qual **constante de acoplamento forte** α_S vai se tornando desprezível no limite de pequenas distâncias (altas energias).

Em distâncias maiores há um aumento na intensidade da interação e espera-se que a força de atração entre os quarks seja constante, o que acabaria determinando o confinamento dos quarks e dos glúons dentro dos hádrons. Isto acaba fazendo com que os estudos de fenômenos importantes, tais como o cálculo da transição do desconfinamento de quarks na QCD a temperatura finita, o mecanismo de confinamento e o espectro de massas dos hádrons não sejam viáveis por teoria de perturbação, que é baseada em expansões de acoplamentos fracos, justamente pelo fato de α_S não ser desprezível a baixas energias. Para tratar o problema de maneira não-perturbativa, pode ser considerada uma formulação de rede da teoria, que equivale a um modelo de mecânica estatística clássica, utilizando técnicas numéricas de Monte Carlo. No entanto, as simulações neste caso são consideravelmente mais complexas do que em Mecânica Estatística devido ao grande número de graus de liberdade e a forma da interação.

A formulação da QCD na rede (lattice QCD), introduzida por K. Wilson em 1974 (10), é baseada em três ingredientes:

- a quantização por meio de integrais de trajetória,
- a continuação para tempos imaginários ou euclidianos,
- a regularização da rede, que é obtida pela discretização do espaço-tempo, com os Férmions (quarks) ocupando os sítios e os campos de gauge (glúons) os elos de ligação.

O limite do contínuo, onde são obtidos os resultados físicos, é fornecido pelo ponto crítico deste modelo.

Dessa forma, as grandezas de interesse, escritas como integrais de trajetória ou soma sobre configurações do sistema (equivalentemente à equação 2.37 acima) podem ser calculadas por métodos usuais em mecânica estatística, como a simulação numérica de Monte Carlo.

Como exemplo do grau de dificuldade do cálculo da QCD, consideremos (9) o procedimento para se obter o resultado do seguinte valor médio de ensemble (espaço de

configurações)

$$\langle O \rangle = \frac{\int d\phi O[\phi] e^{-S[\phi]}}{\int d\phi e^{-S[\phi]}}, \quad (2.42)$$

onde $S[\phi]$ é um funcional real de variáveis de ligação (que nada mais é do que o valor médio de um operador que depende dos graus de liberdade ϕ , onde a probabilidade do sistema estar no estado ϕ é dada por $\exp^{-S[\phi]} / \int d\phi \exp^{-S[\phi]}$). Por simplicidade, vamos considerar apenas os valores possíveis para os campos quânticos ϕ , associados dos elos da rede, como dito acima. Usando uma rede espaço-temporal de 10^4 pontos, o número de variáveis de ligação é aproximadamente 4×10^4 . Se for considerado o caso do grupo de gauge $SU(3)$, cada uma das variáveis de ligação é uma função com 8 parâmetros reais, o que nos obriga a resolver cerca de 320000 integrações. Se for utilizada uma malha de apenas 10 pontos por integração, as integrais múltiplas geram uma somatória na ordem de 10^{320000} termos. Para ter uma ideia do que isto significa, é interessante estimar o tempo de processamento necessário no caso bem mais simples do conhecido modelo de Ising (3). Tomando uma rede de 10^3 pontos temos uma soma com $2^{1000} \approx 10^{300}$ termos. Realizar este cálculo levaria da ordem de 10^{270} vezes a idade do universo em uma máquina de 1 Teraflops!

Não há, portanto, esperança de realizar a soma de forma direta.

Este exemplo demonstra o motivo de utilizar ferramentas estatísticas para tentar resolver essa média de ensemble. No cálculo, podem se utilizar as simulações numéricas baseadas no método de Monte Carlo (descrito no Capítulo 3), que realiza uma descrição estocástica do sistema.

2.6 QCD pura

Mesmo utilizando métodos estocásticos, o cálculo da QCD na sua forma plena é extremamente pesado. Em particular, não é possível realizar a soma sobre configurações como em 2.42 acima no caso dos campos fermiônicos (i.e. os campos de quarks), já que o peso

estatístico não é positivo neste caso, tornando-se necessária a utilização de algoritmos especiais (ver Seção 3.10), bastante lentos. Uma forma de diminuir o peso dos cálculos é fazer simplificações no nível de detalhamento das simulações. O procedimento deste tipo mais comum é chamada uma aproximação *Quenched* (ou QCD pura), que consiste em desprezar os efeitos dos férmions dinâmicos. Isto equivale a produzir configurações considerando os quarks com massa infinita. Vale ressaltar que os observáveis calculados para cada configuração ainda podem incluir quarks com as massas desejadas, que neste caso são chamados de quarks de valência. Mesmo sendo uma aproximação grosseira, há diversos casos em que a QCD pura apresenta poucas correções em relação à QCD completa, indicando que a influência dos férmions dinâmicos é pequena.

2.7 QCD completa

A consideração dos férmions dinâmicos faz com que a simulação da QCD completa seja bem mais lenta, pois na discretização da rede dentro da simulação tem-se a presença de um termo não-local, fornecido pelo determinante fermiônico, o que complica a paralelização da simulação. O ponto principal na simulação de QCD na rede completa é a escolha da formulação para a discretização dos campos fermiônicos. Escolhas comuns são os férmions de Wilson e os de Kogut-Susskind (KS), também conhecidos como férmions staggered, onde o primeiro tem problemas com quebra da simetria quiral por utilizar um espaçamento de rede a finito, já o segundo consegue manter as propriedades quirais, mas acaba produzindo quatro sabores de quarks, o que obriga a ter que tirar raízes do $\det K$. Outra escolha muito importante é o algoritmo para a geração das trajetórias, sendo os mais aconselhados os baseados no Método de Monte Carlo Híbrido, Hybrid Monte Carlo (HMC). O termo híbrido vem do uso de dinâmica molecular combinada com uma amostragem de Monte Carlo. A partir do desenvolvimento do HMC surgiram duas formas mais utilizadas, o algoritmo R (que nos últimos tempos vem caindo em desuso) e

o RHMC (uma evolução das ideias utilizadas no HMC).

Uma descrição detalhada destes algoritmos é dada no Capítulo 3.

2.8 Máquinas Utilizadas

Para a realização de simulações numéricas da QCD, é necessário fazer o uso de um grande poder computacional para poder extrair resultados mais rápidos e com uma maior riqueza de detalhes. Alguns dos super computadores atuais foram projetados especificamente para realizar estudos da QCD na rede, tais como o APE-Next na Europa e o QCDOC (colaboração entre EUA, Japão e Reino Unido). Todos eles apresentam desempenhos na ordem dos multi-Teraflops. Somente no começo desta década, foi possível executar simulações da QCD em sistemas de computadores de pequeno porte, os chamados **clusters de PCs**, pois a montagem destas máquinas um custo muito menor. De fato, mesmo sendo inferiores às máquinas paralelas atuais os clusters de PCs, possuem poder computacional de grandes computadores de uma década atrás, o que já viabiliza diversos tipos de cálculos e estudos. Na verdade, já há uma tendência atual em favor dos clusters para simulação da QCD, como o cluster JUGENE, na Alemanha, embora existam ainda projetos de desenvolvimento de máquinas específicas, como o QPACE, em parceria com a IBM, que esta na primeira posição dos Green500 (11), que são os 500 supercomputadores com maior eficiência de energia no mundo. Vale ressaltar a utilização de novas arquiteturas (como por exemplo, o uso de GPU's) no estudo da QCD que podem significar grandes mudanças em um futuro próximo (12).

2.9 Outros Softwares

A presente dissertação enfoca o uso do pacote MILC, descrito no Capítulo 5, que é o pacote de uso mais difundido na área.

Entretanto, existem outras propostas de softwares para o estudo da QCD na rede disponíveis no meio acadêmico. Podemos citar como as principais o Chroma, o FermiQCD e o Columbia Physics System (CPS).

2.10 Chroma

O pacote Chroma esta na versão 3.28.3 e é gerenciado pelo Jefferson Lab, nos EUA. Ele é escrito em C++ e utiliza diversos aspectos recentes de técnicas de programação, como o uso de Designer Patterns, eXtensible Markup Language (XML), Concurrent Version System (CVS) e Doxygen. Assim como o MILC, o pacote fornece aplicações para a QCD na rede para serem executadas com suporte bastante amplo para arquiteturas diversas. Tem como dependência a instalação do pacote QDP++ para poder realizar os cálculos. Podem ser obtidas mais informações na seguinte referência (13).

2.11 FermiQCD

O FermiQCD seria uma biblioteca padrão de algoritmos para aplicações paralelas da QCD na rede, escrito em C++. Ela foi elaborada para ser de fácil utilização, com uso de sintaxe bem similar a notação matemática comum, e ao mesmo tempo otimizada para ser executado em Clusters de PCs. Todos os algoritmos utilizados no FermiQCD são

paralelos, mas ficam ocultos para programadores de nível mais alto. Mais informações pode ser encontradas na fonte (14).

2.12 Columbia Physics System

O Columbia Physics System, também chamado de CPS, é um conjunto de códigos para o estudo da QCD na rede, o pacote utiliza programação orientada à objeto para representar diversos tipo de ações na rede, além de possuir rotinas otimizadas em assembler para as arquiteturas QCDSF e QCDOC. O projeto esta em funcionamento e é gerenciado conjuntamente pela Universidade de Columbia, Laboratório Nacional de Brookhaven, United Kingdom QCD (UKQCD) e Edinburgh Parallel Computing Centre (EPCC)(15).

O Método de Monte Carlo

Neste capítulo introduzimos o método de Monte Carlo (3, 16) e descrevemos os principais algoritmos que fazem parte do pacote MILC. Parte da nossa discussão é baseada no livro do Rothe (9), assim como em vários artigos de pesquisa. Vamos inicialmente introduzir os aspectos gerais do método em 3.1. Nas seções 3.4, 3.5, 3.6 e 3.7 discutiremos métodos alternativos para seguir a evolução temporal de um sistema. Na seção 3.8 descreveremos o Método de Monte Carlo Híbrido, que será a base da simulação da QCD completa. (Nas seções 3.9 e 3.10 apresentaremos alternativas e aplicações do método.). Finalmente, nas seções 3.11, 3.12, 3.13 e discutimos os algoritmos R, PHMC, RHMC e DDHMC, que são utilizados nas simulações atuais. O método RHMC é o mais recente e apresenta o melhor desempenho. Foi introduzido na versão 7 do pacote MILC.

3.1 Introdução

A ideia básica do Método de Monte Carlo é a identificação matemática de probabilidades com o cálculo de integrais. Pode-se resolver uma integral utilizando amostras do integrando, obtidos através de sorteios de configurações aleatórias das variáveis com distribuição de probabilidade proporcional à medida de integração. Essa técnica é chamada de “amostragem por importância”.

É claro que existem métodos mais eficientes para o cálculo de integrais na forma numérica, mas eles são eficientes apenas em dimensões baixas, tornando-se impraticáveis em dimensões maiores, tais como na seção 2.5.

Desta forma, o método de Monte Carlo é uma alternativa para resolver o problema da QCD na rede de forma eficiente, consistindo na geração de uma sequência de configurações de variáveis de ligação com uma distribuição de probabilidades definida pelo fator de Boltzmann $\exp(-S[\phi])$. Retoma-se a equação 2.42

$$\langle O \rangle = \frac{\int d\phi O[\phi] e^{-S[\phi]}}{\int d\phi e^{-S[\phi]}} ,$$

onde o operador O depende do campo ϕ , a ação é $S[\phi]$, a medida é $d\phi$ e o valor no denominador é a condição de normalização.

No cálculo do valor esperado, é gerada uma sequência de configurações de campo $\phi_i (i = 1, \dots, N)$ escolhidas a partir da distribuição de probabilidades

$$P(\phi_i) d\phi_i = \frac{e^{-S[\phi_i]}}{\int d\phi_i e^{-S[\phi_i]}} d\phi_i , \quad (3.1)$$

como será explicado mais adiante.

Se a sequência gerada consiste de um conjunto representativo de configurações, então a chamada “média de ensemble” $\overline{O}$ é dada por,

$$\overline{O} = \frac{1}{N} \sum_{i=1}^N O(\phi_i) . \quad (3.2)$$

Devido à lei dos grandes números, temos que $\overline{O}$ aproxima-se do valor esperado $\langle O \rangle$ à medida que N tende a infinito

$$\langle O \rangle = \lim_{N \rightarrow \infty} \overline{O} . \quad (3.3)$$

O procedimento para o cálculo da média de ensemble é baseado no uso de cadeias de Markov (definida mais adiante dentro desta seção) e consiste em estipular uma regra para a geração da sequência de configurações, desconsiderando um número apropriado de configurações inicialmente geradas, de forma a garantir a termalização. O número de passos para se obter a termalização, assim como o número de passos entre duas configurações termalizadas independentes, depende dos seguintes fatores

1. Algoritmo utilizado.
2. Observável estudado.
3. Parâmetros associados ao problema.

Isso acaba criando uma dificuldade quando são geradas amostras termalizadas para a QCD na rede perto do limite do contínuo, já que este limite corresponde ao ponto crítico do modelo de mecânica estatística associado (ver Seção 2.5). Nas proximidades deste ponto, com o aparecimento de comportamento crítico (i.e. divergências dos comprimentos de correlação do sistema), o número de passos requeridos aumenta com a potência do tamanho do sistema. Este fenômeno é chamado de *critical slowing down*.

Os princípios gerais para a construção de algoritmos garantindo que a sequência gerada de configurações constitua uma amostra representativa e que possa ser utilizada para a medida de observáveis são satisfeitos por um processo markoviano, isto é, tomando-se as configurações mencionadas como elementos de uma cadeia de Markov. Na descrição abaixo considera-se que as configurações sejam discretas (por simplicidade nas demonstrações). Se as mesmas forem contínuas, basta substituir as somas abaixo por integrais, e a probabilidade de ocorrência de uma dada configuração pela respectiva densidade de probabilidade.

Vamos começar definindo as cadeias de Markov. Seja $C_1, C_2, \dots$ um conjunto enumerável de estados do sistema e consideremos um processo estocástico onde um conjunto finito de configurações é gerado sequencialmente de acordo com a probabilidade de transição $P(C_i \rightarrow C_j) \equiv P_{ij}$, sendo P_{ij} a probabilidade para uma configuração ir de C_i para C_j , e sejam $C_{r_1}, C_{r_2}, \dots$ as configurações geradas desta forma. Devido ao elemento probabilístico construído neste procedimento, o estado do sistema em qualquer tempo (de simulação) é dado por uma variável aleatória, onde a distribuição irá depender apenas do estado precedente a ela. Isto é, a probabilidade de transição é independente de todos os estados exceto o imediatamente anterior a ela (isto é, ela não carrega a história dos passos anteriores). Uma cadeia de Markov é definida quando seus elementos sucessivos são as variáveis aleatórias mencionadas acima. Agora, dado um conjunto de configurações $\{C_{r_i}\}$

geradas por um processo markoviano e um observável O calculado sobre este conjunto de estados, pode-se definir uma média temporal de O obtida sobre N elementos da cadeia de Markov por

$$\langle O \rangle = \frac{1}{N} \sum_{i=1}^N O(C_{\tau_i}) . \quad (3.4)$$

Essa é a quantidade obtida na prática e é tomada como igual ao valor procurado como média de ensemble correspondente a uma dada distribuição de Boltzmann, utilizando-se uma probabilidade de transição P_{ij} apropriada. Com isso em vista, a discussão será focada para as cadeias de Markov chamadas irredutíveis, aperiódicas, nas quais os estados são positivos. A seguir temos uma breve explicação sobre estes tipos de cadeias de Markov.

1. Uma cadeia é chamada de irredutível se, ao começar de uma configuração arbitrária C_i , existir uma probabilidade finita de alcançar qualquer outra configuração C_j após um número finito de passos de Markov. Em outras palavras, existe um número finito N tal que

$$P_{ij}^{(N)} = \sum_{i_k} P_{ii_1} P_{i_1 i_2} \dots P_{i_{N-1} j} \neq 0 . \quad (3.5)$$

2. Uma cadeia de Markov é chamada de aperiódica se $P_{ii}^{(N)} \neq 0$ para qualquer N .
3. Um estado é chamado de positivo se a média do chamado tempo de recorrência for finita. Se $p_{ii}^{(n)}$ é a probabilidade de sair de C_i e chegar até C_i em n -passos da cadeia de Markov sem passar por esta configuração em qualquer passo intermediário, então a média temporal de recorrência de C_i é dada por

$$\tau_i = \sum_{n=1}^{\infty} n p_{ii}^{(n)} . \quad (3.6)$$

Teorema 1

Se a cadeia é irredutível e os estados são positivos e aperiódicos, tem-se que o limite $N \rightarrow \infty$ em 3.5 existe, e é único. Temos que

$$\lim_{N \rightarrow \infty} P_{ij}^{(N)} = \pi_j , \quad (3.7)$$

onde $\{\pi_j\}$ são números que satisfazem as equações

$$\pi_j > 0, \quad \sum_j \pi_j = 1 , \quad (3.8)$$

$$\pi_j = \sum_i \pi_i P_{ij} . \quad (3.9)$$

Interpretando esses resultados, a equação 3.7 mostra que no limite de número de passos infinito $P_{ij}^{(N)}$ é independente da configuração usada para iniciar o processo markoviano. Além disso, as equações em 3.1 indicam que o conjunto $\{\pi_i\}$ não é modificado durante a atualização desses números com a probabilidade de transição $\{P_{ij}\}$. Mas isto, juntamente com 3.8, é apenas a condição para que o sistema esteja em equilíbrio, com π_i sendo a probabilidade de encontrar a configuração C_i . No nosso caso, temos que escolher P_{ij} de forma que π_i seja a distribuição de Boltzmann desejada, como descrito na próxima seção.

Teorema 2

Se a cadeia é irredutível e os seus estados são positivos, e se

$$\tau_i^{(2)} \equiv \sum_{n=1}^{\infty} n^2 p_{ii}^{(n)} < \infty , \quad (3.10)$$

então o tempo médio de 3.4 aproxima da média de ensemble por

$$\langle O \rangle = \sum_i \pi_i O(C_i) , \quad (3.11)$$

com uma incerteza estatística da ordem $O(1/\sqrt{N})$.

Os teoremas acima providenciam as bases para a utilização de um processo markoviano para se calcular a média de ensemble em 2.42.

3.2 O Princípio do Balanço Detalhado

O princípio do Balanço Detalhado relaciona as probabilidades de transição com a distribuição de equilíbrio. Determinaremos a forma da probabilidade de transição para a geração das configurações da cadeia de Markov, eventualmente distribuída por uma dada probabilidade de equilíbrio.

Para englobar o caso onde as configurações são rotuladas como um conjunto de variáveis contínuas, desconsideraremos o índice discreto em C_i e as configurações serão deno-

tadas por letras maiúsculas para representar uma configuração arbitrária. Desta forma, as equações 3.9 e 3.11 serão reescritas assim

$$P_{eq}(C) = \sum_{C'} P_{eq}(C') P(C' \rightarrow C) , \quad (3.12)$$

$$\langle O \rangle = \sum_C P_{eq}(C) O(C) , \quad (3.13)$$

onde $P_{eq}(C)$ agora descreve a densidade de probabilidade para encontrar a configuração C no equilíbrio, e $P(C \rightarrow C')$ é a densidade de probabilidade de transição para ir de C até C' . Para um processo markoviano, é requisito suficiente para a amostragem da distribuição de equilíbrio $P_{eq}(C)$ que a probabilidade de transição (densidade) satisfaça o balanço detalhado

$$P_{eq}(C) P(C \rightarrow C') = P_{eq}(C') P(C' \rightarrow C) , \quad (3.14)$$

para qualquer par C e C' . Note que a probabilidade de transição $P(C \rightarrow C')$ não é uma função de variáveis rotulando as configurações C e C' , mas define uma regra de como selecionar a próxima configuração em um cadeia de Markov dada a configuração imediatamente anterior.

Satisfeita a condição acima, a evolução da cadeia [utilizando $P(C \rightarrow C')$] por um tempo suficientemente longo determinará que as configurações estejam distribuídas de acordo com $P_{eq}(C)$, como desejado.

De fato, considere por exemplo a geração da seguinte densidade de probabilidade

$$P_{eq}(C) = \frac{\exp^{-S(C)}}{\sum_c \exp^{-S(C)}} . \quad (3.15)$$

Usando a condição de normalização

$$\sum_{C'} P(C \rightarrow C') = 1 \quad (3.16)$$

e supondo que $P(C \rightarrow C')$ satisfaça 3.14, tem-se que 3.12 também é satisfeita. Então, se as condições do teorema 1 forem satisfeitas, 3.15 é a única distribuição em equilíbrio gerada pelo processo markoviano, relacionando a probabilidade de transição à distribuição de equilíbrio desejada.

O requisito do balanço detalhado não é determinado apenas pela probabilidade de transição, e pode ser utilizado livremente para elaborar algoritmos eficientes adaptados para um dado problema a ser estudado.

3.3 Ergodicidade

Existem diversas definições sobre o que é ser um sistema ergódico. Nesta dissertação, a definição que é utilizada para o termo ergódico é a mesma usada na referência (17), onde um sistema é chamado de ergódico se um determinado microestado do sistema pode ser atingido partindo de qualquer outro microestado, após um número N de transições. A consequência central desta propriedade é que uma média de ensemble possui o mesmo valor de uma média temporal, após uma quantidade elevada de passos temporais.

3.4 Método de Metropolis

Este método foi proposto originalmente por Metropolis et al. (18) e em princípio pode ser aplicado a qualquer sistema. A proposta deste algoritmo é encontrar o macroestado de equilíbrio físico para uma dada temperatura T . Apresentaremos regras para a geração das sequências e será provado que as mesmas satisfazem o balanço detalhado.

Uma configuração C qualquer é atualizada, onde sugerindo-se uma nova configuração C' com a probabilidade de transição $P_0(C \rightarrow C')$ que satisfaça o seguinte requerimento de micro-reversibilidade

$$P_0(C \rightarrow C') = P_0(C' \rightarrow C) . \quad (3.17)$$

Em seguida verifica-se a mesma é aceita. A resposta logicamente depende das ações $S(C)$ e $S(C')$. A decisão proposta pelo método ocorre da seguinte forma: Se $\exp[-S(C')] >$

$\exp[-S(C)]$, ou seja se a ação diminui (equivale dizer que a energia do sistema diminui), então a configuração C' é aceita. Por outro lado se a ação é aumentada (a energia do sistema aumenta), a configuração ainda tem chance de ser aceita de acordo com a probabilidade $\exp[-S(C')] / \exp[-S(C)]$. Na prática é gerado um número aleatório R no intervalo entre $[0,1]$ e a configuração C' somente será aceita se

$$R \leq \frac{e^{-S(C')}}{e^{-S(C)}} . \quad (3.18)$$

Qualquer outro resultado faz a configuração C' ser rejeitada, e com isso a configuração anterior C é mantida. O detalhe deste sorteio do número aleatório é que o mesmo permite que ocorra um aumento na ação (equivalentemente na energia). O algoritmo desta forma permite a criação de flutuações quânticas.

Provaremos agora que o algoritmo satisfaz o balanço detalhado, utilizando o fato de que a probabilidade para a transição de $C \rightarrow C'$ é o produto da probabilidade $P_0(C \rightarrow C')$ para um C' ser sugerido como nova configuração a probabilidade de aceitação, dada no algoritmo como descrito acima. Podemos então determinar a probabilidade de transição $P(C \rightarrow C')$ da seguinte forma. Se $\exp[-S(C')] > \exp[-S(C)]$, então

$$P(C \rightarrow C') = P_0(C \rightarrow C') ,$$

e

$$P(C' \rightarrow C) = P_0(C' \rightarrow C) \frac{e^{-S(C)}}{e^{-S(C')}} .$$

Desde que $P_0(C' \rightarrow C) = P_0(C \rightarrow C')$, tem-se que 3.14 é satisfeita. Por outro lado, se $\exp[-S(C')] < \exp[-S(C)]$, então

$$P(C \rightarrow C') = P_0(C \rightarrow C') \frac{e^{-S(C')}}{e^{-S(C)}} ,$$

e

$$P(C' \rightarrow C) = P_0(C' \rightarrow C) ,$$

e da mesma forma é provado o balanço detalhado.

O algoritmo de Metropolis pode ser utilizado para fazer atualização em um número X de variáveis, mas esta escolha deve ser feita de forma inteligente, pois o sistema corre

o risco de ficar parado em um estado, não simulando o fenômeno que estamos associando a ele. Vale ressaltar que podemos também variar as configurações em um único ponto, mas aí a quantidade de atualizações que devemos realizar é da ordem do volume do sistema. De qualquer forma, este não é o principal problema com atualizações locais, mas sim o fenômeno de critical slowing down, discutido anteriormente. Serão mostradas mais adiante neste capítulo formas melhores para a atualização de um grande número de pontos e ainda assim ser possível utilizar o teste de Metropolis.

3.5 Algoritmo de Langevin

Apresentamos agora o algoritmo de Langevin, inicialmente elaborado por Parisi e Wu (19) e proposto como método de atualização na QCD completa por Fukugita e Ukawa (20) e Batrouni et al. (21).

Denotamos por q_i ($i = 1, \dots, N$) as coordenadas de um sistema com ação $S[q]$. No caso da QCD pura essas são as variáveis de ligação da rede espaço-tempo. Depois de fazer as coordenadas serem dependentes de uma nova variável temporal τ que, quando discretizada, rotula os elementos de uma cadeia de Markov, consideremos a seguinte equação diferencial que relaciona as variáveis com os “tempos” $\tau_{n+1} = (n+1)\epsilon_L$ e $\tau_n = n\epsilon_L$, sendo ϵ_L chamado de “passo temporal de Langevin”

$$q_i(\tau_{n+1}) = q_i(\tau_n) + \epsilon_L \left[-\frac{\partial S[q]}{\partial q_i(\tau_n)} + \eta_i(\tau_n) \right], \quad (3.19)$$

onde $\{\eta_i(\tau_n)\}$ é um conjunto variáveis aleatórias de distribuições gaussianas independentes, com distribuição de probabilidade dada por

$$P(\{\eta_i(\tau_n)\}) = \prod_i \sqrt{\frac{\epsilon_L}{4\pi}} \exp \left[-\sum_i \frac{\epsilon_L}{4} \eta_i(\tau_n)^2 \right]. \quad (3.20)$$

No limite $\epsilon_L \rightarrow 0$, a equação 3.19 se torna a equação de Langevin

$$\frac{dq_i}{d\tau} = -\frac{\partial S[q]}{\partial q_i} + \eta_i(\tau). \quad (3.21)$$

Desde que a largura da distribuição 3.20 seja da ordem de $1/\sqrt{\epsilon_L}$, o termo do ruído gaussiano em 3.19 é da ordem de $\sqrt{\epsilon_L}$. Podemos reescrever a equação com a introdução de uma nova variável aleatória $\tilde{\eta}_i(\tau_n) = (\sqrt{\epsilon_L/2}) \eta_i(\tau_n)$. Então 3.19 toma a forma

$$q_i(\tau_{n+1}) = q_i(\tau_n) - (2\epsilon_L) \frac{1}{2} \frac{\partial S[q]}{\partial q_i(\tau_n)} + \sqrt{2\epsilon_L} \tilde{\eta}_i(\tau_n) , \quad (3.22)$$

onde $\{\tilde{\eta}_i(\tau_n)\}$ é agora a distribuição da variável aleatória de acordo com

$$\tilde{P}(\tilde{\eta}(\tau_n)) = \left(\prod_i \frac{1}{\sqrt{2\pi}} \right) e^{-\sum_i \frac{1}{2} \tilde{\eta}_i(\tau_n)^2} \quad (3.23)$$

com a variância

$$\langle \tilde{\eta}_i(\tau_n) \tilde{\eta}_j(\tau_m) \rangle = \delta_{ij} \delta_{nm} .$$

Essa forma da equação diferencial será mais útil na discussão do algoritmo híbrido de dinâmica molecular, discutido na Seção 3.6.

Considere a equação 3.22. Nesta evolução tem-se que a probabilidade de fazer uma transição de q_i no “tempo” τ_n para q'_i no “tempo” τ_{n+1} é a probabilidade do ruído $\tilde{\eta}_i$ ($i = 1, 2, \dots$) ser igual a $\frac{\sqrt{2\epsilon_L}}{2} \frac{\partial S[q]}{\partial q_i} + \frac{(q'_i - q_i)}{\sqrt{2\epsilon_L}}$. Novamente no limite de $\epsilon_L \rightarrow 0$ podemos observar a seguinte evolução

$$\begin{aligned} \frac{P(q \rightarrow q')}{P(q' \rightarrow q)} &\xrightarrow{\epsilon_L \rightarrow 0} \frac{\prod_i P\left(\tilde{\eta}_i = \frac{\sqrt{2\epsilon_L}}{2} \frac{\partial S[q]}{\partial q_i} + \frac{(q'_i - q_i)}{\sqrt{2\epsilon_L}}\right)}{\prod_i P\left(\tilde{\eta}_i = \frac{\sqrt{2\epsilon_L}}{2} \frac{\partial S[q]}{\partial q_i} + \frac{(q_i - q'_i)}{\sqrt{2\epsilon_L}}\right)} \\ &= \frac{\exp\left(-\frac{1}{2} \sum_i \left(\frac{\sqrt{2\epsilon_L}}{2} \frac{\partial S[q]}{\partial q_i} + \frac{(q'_i - q_i)}{\sqrt{2\epsilon_L}}\right)^2\right)}{\exp\left(-\frac{1}{2} \sum_i \left(\frac{\sqrt{2\epsilon_L}}{2} \frac{\partial S[q]}{\partial q_i} + \frac{(q_i - q'_i)}{\sqrt{2\epsilon_L}}\right)^2\right)} \\ &= e^{-\sum_i (q'_i - q_i) \frac{\partial S}{\partial q_i}} \\ &\xrightarrow{\epsilon_L \rightarrow 0} e^{-[S(q') - S(q)]} . \end{aligned} \quad (3.24)$$

Desta forma, provamos que a probabilidade de transição satisfaz o balanço detalhado

$$e^{-S(q)} P(q \rightarrow q') = e^{-S(q')} P(q' \rightarrow q) . \quad (3.25)$$

Fica demonstrado portanto que o algoritmo leva à distribuição de equilíbrio desejada, proporcional a $e^{-S(q)}$. Em passos finitos do tempo de Langevin o algoritmo leva a erros sistemáticos que precisam ser controlados. Isto significa que os dados obtidos usando este

algoritmo precisam ser extrapolados para $\epsilon_L \rightarrow 0$. Este requerimento acaba forçando a geração de configurações para diversos passos do tempo de Langevin, o que consome muito tempo de processamento. Por outro lado, ele é um algoritmo simples, que pode ser utilizado para atualizar todas as variáveis de uma vez, sem ter que se preocupar com as taxas de aceitação, desde que não sejam utilizados os testes de aceitação de Metropolis. Em particular, o algoritmo de Langevin é muito útil no tratamento de sistemas que possuem ação não localizada, como o caso da QCD quando consideramos férmions dinâmicos.

3.6 Método da Dinâmica Molecular

A ideia central do método da dinâmica molecular é que a integral de caminho euclidiana associada à teoria quântica pode ser escrita na forma de uma função de partição para um sistema de mecânica estatística clássica em quatro dimensões, com um Hamiltoniano canônico que governa a dinâmica em uma nova variável temporal, o tempo de simulação. Invoca-se a ergodicidade e usa-se o fato de que, no limite termodinâmico, a média de ensemble canônica pode ser obtida através da média de ensemble do micro-canônico com uma energia determinada pelos parâmetros do sistema. Assim podem ser calculados os valores esperados dos observáveis numa média temporal sobre as trajetórias “clássicas”. Na aproximação da dinâmica molecular, as configurações são geradas de forma determinística, e a complexidade do movimento em 4 dimensões nos fornece as flutuações quânticas presentes na teoria original.

Por simplicidade, consideramos o caso de uma teoria de campo escalar, com uma ação $S[\phi; \beta]$ dependendo apenas do campo escalar ϕ e de um parâmetro β (uma constante de acoplamento, por exemplo). O valor esperado de um observável é considerado como

$$\langle O \rangle = \frac{1}{Z} \int D\phi O[\phi] e^{-S[\phi; \beta]} , \quad (3.26)$$

onde

$$Z = \int D\phi e^{-S[\phi; \beta]} \quad (3.27)$$

e

$$D\phi = \prod_i d\phi_i . \quad (3.28)$$

É introduzida uma rede no espaço-tempo para definir a integral 3.6. Os graus de liberdade do sistema são rotulados pelas coordenadas dos sítios da rede, denotados por i . Apesar de a integral 3.6 ter semelhanças com a encontrada em mecânica estatística, Z não é a função de partição de um sistema hamiltoniano clássico. A mesma pode ser obtida através da introdução de um conjunto de momentos conjugados π_i . Então, 3.6 pode ser escrita da forma

$$\langle O \rangle = \frac{1}{\bar{Z}} \int D\phi D\pi O[\phi] e^{-H[\phi, \pi; \beta]} , \quad (3.29)$$

onde

$$H[\phi, \pi; \beta] = \sum_i \frac{1}{2} \pi_i^2 + S[\phi; \beta] , \quad (3.30)$$

$$\bar{Z} = \int D\phi D\pi e^{-H[\phi, \pi; \beta]} \quad (3.31)$$

e a medida de integração é definida como

$$D\phi D\pi = \prod_i d\phi_i d\pi_i .$$

Claramente 3.29 – 3.31 coincidem com as expressões em 3.6, desde que O não dependa do momento. Vale ressaltar que a teoria quântica em quatro dimensões no espaço-tempo euclidiano relembra uma amostragem canônica clássica em quatro dimensões em contato com um reservatório térmico. Note que 3.6 não pode ser confundida com a representação usual do espaço de fase da integral de trajetória 3.6, onde a estrutura é completamente diferente e envolve o hamiltoniano do sistema em três dimensões espaciais (veja por exemplo a discussão na Seção 2.3).

No limite termodinâmico, a média de ensemble canônica pode ser substituída pela média de ensemble do micro-canônico que evolui na superfície de “energia” determinada pelo sistema. Essa “energia” é de um sistema na mecânica estatística em quatro dimensões, e dessa forma, a ação do sistema na mecânica quântica permite que ocorram flutuações, e que as mesmas sejam obtidas no cálculo. Além disso, a equação 3.30 fornece uma conexão entre o parâmetro β e a “energia” que a média de ensemble do micro-canônico é calculada.

Consideremos que o valor esperado em 3.6 possa ser escrito da forma

$$\langle O \rangle_{can}(\beta) = \frac{\int D\phi D\pi O[\phi] \int dE \delta(H[\phi, \pi; \beta] - E) e^{-E}}{\int D\phi D\pi \int dE \delta(H[\phi, \pi; \beta] - E) e^{-E}} , \quad (3.32)$$

onde o índice *can* serve para indicar que é calculado o valor da média de ensemble canônica, quantidade esta que depende de β .

Por outro lado, a média de ensemble micro-canônica $\langle O \rangle_{mic}$, resolvida para um determinado β na superfície de energia $H = E$, é dada por

$$\langle O \rangle_{mic}(E, \beta) = \frac{1}{Z_{mic}(E, \beta)} \int D\phi D\pi O[\phi] \delta(H[\phi, \pi; \beta] - E) , \quad (3.33)$$

onde

$$Z_{mic}(E, \beta) = \int D\phi D\pi \delta(H[\phi, \pi; \beta] - E) \quad (3.34)$$

é o volume no espaço de fase com energia E . Estabelecemos uma conexão entre estas duas médias de ensemble no limite termodinâmico. Vamos definir a entropia de um sistema por

$$s(E, \beta) = \ln [Z_{mic}(E, \beta)] . \quad (3.35)$$

Introduzindo as definições 3.6 e 3.35 em 3.32, a conexão será escrita na seguinte forma

$$\langle O \rangle_{can}(\beta) = \frac{\int dE \langle O \rangle_{mic}(E, \beta) e^{-(E-s(E, \beta))}}{\int dE e^{-(E-s(E, \beta))}} . \quad (3.36)$$

Agora vem o argumento central: à medida que os graus de liberdade do sistema vão se tornando grandes, as exponenciais dentro dos integrandos de 3.36 são fortemente suprimidas por uma energia $\bar{E}$ dada implicitamente por

$$\left(\frac{\partial s(E, \beta)}{\partial E} \right)_{E=\bar{E}} = 1 . \quad (3.37)$$

No limite termodinâmico isso nos leva à seguinte conexão

$$\langle O \rangle_{can}(\beta) = [\langle O \rangle_{mic}]_{E=\bar{E}(\beta)} , \quad (3.38)$$

onde β e $\bar{E}$ são relacionados por 3.37. Esta relação, entretanto, não é útil quando estamos usando cálculo numérico. Uma expressão mais adequada pode ser obtida fazendo uso do teorema de equipartição (22), que nos diz que a seguinte equação existe para cada momento π_i

$$\left\langle \pi_i \frac{\partial H}{\partial \pi_i} \right\rangle_{mic} = \frac{1}{\partial s / \partial E} . \quad (3.39)$$

Inserindo o H da expressão 3.30 e fazendo uso de 3.37 temos

$$[\langle K \rangle_{mic}]_{E=\bar{E}(\beta)} = \frac{N}{2} , \quad (3.40)$$

onde $K = \sum_i \pi_i^2/2$ é a energia cinética e N é o número de graus de liberdade do sistema.

Uma forma mais familiar da equação 3.40 é obtida quando a dependência do parâmetro β é multiplicativa, isto é, se

$$S[\phi, \beta] = \beta V[\phi] . \quad (3.41)$$

Neste caso o valor esperado 3.6 pode ser escrito como se segue

$$\langle O \rangle = \frac{1}{\tilde{Z}} \int D\phi D\pi O[\phi] e^{-\beta H[\phi, \pi]} ,$$

onde

$$H[\phi, \pi] = \frac{1}{2} \sum_i \pi_i^2 + V[\phi]$$

e

$$\tilde{Z} = \int D\phi D\pi e^{-\beta H[\phi, \pi]} .$$

Pelos argumentos apresentados anteriormente, chega-se à conclusão de que a conexão entre as médias de ensemble do canônico e do micro-canônico é

$$\langle O \rangle_{can}(\beta) = \langle O \rangle_{mic}(\bar{E}) ,$$

onde

$$\langle O \rangle_{mic}(E) = \frac{1}{Z_{mic}(E)} \int D\phi D\pi O[\phi] \delta(H[\phi, \pi] - E)$$

e

$$Z_{mic}(E) = \int D\phi D\pi \delta(H[\phi, \pi] - E) .$$

A relação entre β e $\bar{E}$ é dada por

$$\frac{N}{2\beta} = \langle K \rangle_{mic}(\bar{E}) .$$

Esta é a forma familiar do teorema de equipartição, com β fazendo o papel do inverso da temperatura.

Com essas informações, retomaremos a expressão 3.38, e reescreveremos o lado direito de uma forma conveniente para cálculos numéricos. Supondo que o movimento gerado pelo Hamiltoniano 3.30 seja ergódico, serão utilizadas as equações de Hamilton do movimento

$$\dot{\phi}_i = \frac{\partial H[\phi, \pi]}{\partial \pi_i} , \quad (3.42)$$

$$\dot{\pi}_i = -\frac{\partial H[\phi, \pi]}{\partial \phi_i} \quad (3.43)$$

para gerar uma amostra representativa de configurações do espaço de fase com energia constante. Desde que os observáveis (que são os valores esperados que desejamos calcular) dependam apenas das coordenadas, a média de ensemble do micro-canônico pode ser substituída por uma média temporal sobre $\phi(\tau)$, levando à seguinte cadeia de equações válidas para sistemas ergódicos dentro do limite termodinâmico,

$$\langle O \rangle_{can}(\beta) \xrightarrow{\text{lim. termod.}} [\langle O \rangle_{mic}]_{E=\overline{E}(\beta)} \xrightarrow{\text{ergodic.}} \lim \frac{1}{T} \int_0^T d\tau O(\phi_i(\tau)) , \quad (3.44)$$

onde a trajetória $\{\phi_i(\tau)\}$ é uma solução para

$$\frac{d^2 \phi_i}{d\tau^2} = -\frac{\partial S[\phi]}{\partial \phi_i} , \quad (3.45)$$

sujeita às condições iniciais da energia $\overline{E}$. Vale ressaltar que, supondo que o movimento seja ergódico, podemos utilizar qualquer configuração inicial com essa energia para realizar a integração.

Perceba a diferença na estrutura de 3.45 e a equação de Langevin 3.21. Enquanto 3.45 é um conjunto determinístico de equações diferenciais de segunda ordem acopladas, a equação de Langevin é de primeira ordem, onde se tem uma variável estocástica. Na formulação de Langevin, as equações diferenciais correspondentes são dadas por 3.22. Considerando a discretização simples de 3.45 e usando a versão simétrica para a discretização da derivada segunda, $\ddot{\phi}_i(\tau_n) = [\phi_i(\tau_{n+1}) + \phi_i(\tau_{n-1}) - 2\phi_i(\tau_n)] / (\Delta\tau)^2$, obteremos que

$$\phi_i(\tau_{n+1}) = \phi_i(\tau_n) - \epsilon^2 \frac{\partial S[\phi]}{2 \partial \phi_i(\tau_n)} + \epsilon \pi_i(\tau_n) , \quad (3.46)$$

onde $\epsilon = \Delta\tau = \tau_{n+1} - \tau_n$ é o passo temporal do micro-canônico, e

$$\pi_i(\tau_n) = \frac{1}{2\epsilon} [\phi_i(\tau_{n+1}) - \phi_i(\tau_{n-1})] . \quad (3.47)$$

A versão discretizada obtida da equação de Langevin obtida em 3.22 pode ser gerada da equação 3.46 se utilizarmos os momentos pi_i como distribuições de variáveis aleatórias de acordo com a equação 3.23 e fazendo a identificação de ϵ_L com $\epsilon^2/2$.

Comparemos as “distâncias” percorridas no espaço de configurações pelos algoritmos para $\epsilon_L = \epsilon^2/2$. No caso da dinâmica molecular, a distância coberta após n passos é de $O(n\epsilon)$, enquanto na aproximação de Langevin é de apenas $O(\sqrt{n}\epsilon)$, devido à natureza estocástica. Assim, o algoritmo clássico move-se no sistema mais rapidamente no espaço de configurações. Porém, vale ressaltar, a equação 3.38 funciona apenas para sistemas com um número infinito de graus de liberdade. Isto claramente não é o caso em cálculos com pequenos tamanhos de redes. Além disso, o último passo na cadeia 3.44 requer que o movimento seja ergódico. Entretanto, em contraste com a aproximação de Langevin, a ergodicidade não é explicitamente construída dentro do algoritmo de dinâmica molecular. Isto acaba sugerindo um novo algoritmo, que combine as melhores partes destes dois métodos. Este algoritmo será descrito na próxima seção.

3.7 Algoritmo Híbrido

Um forma possível de combinar a aproximação estocástica de Langevin e a simulação micro-canônica em um novo algoritmo que leve em conta a ergodicidade foi proposta por Duane (23). Este algoritmo foi chamado de híbrido, já que esta baseado em utilizar o algoritmo de dinâmica molecular na forma 3.46, que se torna a equação de Langevin discretizada quando o passo temporal é igual $\epsilon_L = \epsilon^2/2$, se o momento $\{\pi_i\}$ for substituído por uma variável aleatória com densidade de probabilidade

$$P(\{\pi_i\}) = \left(\prod_i \frac{1}{\sqrt{2\pi}} \right) e^{-\sum_i \frac{1}{2} \pi_i^2} . \quad (3.48)$$

Note que isto é precisamente a distribuição do momento que aparece em 3.6, onde se o movimento gerado pelo hamiltoniano 3.30 é ergódico, então o momento pode ser eventualmente distribuído de acordo com 3.6.

Desta forma, ao invés de integrar as equações de movimento de Hamilton 3.6 ao longo de uma única trajetória com uma dada energia $\bar{E}$, nós interrompemos o processo de integração por um instante e escolhemos um novo conjunto de momentos com densidade de probabilidade dada por 3.48. Isso corresponde a executar um passo de Langevin, que repetido muitas vezes garante a ergodicidade. Ao mesmo tempo, tiramos vantagem do fato de os passos de dinâmica molecular moverem o sistema mais rapidamente no espaço de configurações do que a aproximação de Langevin.

Note que assim o algoritmo híbrido contém um novo parâmetro: A frequência com que o momento é renovado (ou “refrescado”). Este parâmetro deve ser sintonizado para dar melhores resultados. De fato, Duane e Kogut (23) sugeriram que a cada passo do processo de atualização uma escolha aleatória seja feita entre a atualização da dinâmica molecular com probabilidade α [que corresponde a relacionar o momento em 3.46 com ϕ de acordo com 3.47] e uma atualização de Langevin com probabilidade $1 - \alpha$ [que corresponde a substituir $\{\pi_i\}$ por números aleatórios com distribuição gaussiana]. Se α é próximo de 1, o sistema se move mais rápido no espaço de configurações, agindo de uma forma bem próxima ao caso de dinâmica molecular, mas não contempla a ergodicidade. Por outro lado, se α é próximo a zero, obtemos a ergodicidade, mas o sistema se move mais lentamente dentro do espaço de configurações. A escolha ideal para α fica entre estes dois extremos e é determinada numericamente.

Vejamos agora como este algoritmo é implementado na prática, considerando por simplicidade o caso da teoria de campo escalar.

Tomemos as equações de movimento na forma hamiltoniana 3.6 e discretizemos no tempo com passo temporal ϵ . Para tanto, vamos expandir $\phi_i(\tau + \epsilon)$ e $\pi_i(\tau + \epsilon)$ em série de Taylor até ordem ϵ^2 :

$$\phi_i(\tau + \epsilon) = \phi_i(\tau) + \epsilon \dot{\phi}_i(\tau) + \frac{\epsilon^2}{2} \ddot{\phi}_i(\tau) + O(\epsilon^3) \quad (3.49)$$

$$\pi_i(\tau + \epsilon) = \pi_i(\tau) + \epsilon \dot{\pi}_i(\tau) + \frac{\epsilon^2}{2} \ddot{\pi}_i(\tau) + O(\epsilon^3) \quad (3.50)$$

Ao mesmo tempo, das equações de movimento de Hamilton obtém-se que $\dot{\phi}_i = \pi_i(\tau)$ e

$\ddot{\phi}_i(\tau) = \dot{\pi}_i(\tau) = -\partial S / \partial \phi_i(\tau)$. Então segue que

$$\ddot{\pi}_i(\tau) = - \sum_j \frac{\partial^2 S}{\partial \phi_i(\tau) \partial \phi_j(\tau)} \pi_j(\tau) . \quad (3.51)$$

Expandindo a 3.51 temos

$$\sum_j \frac{\partial^2 S}{\partial \phi_i(\tau) \partial \phi_j(\tau)} \pi_j(\tau) = \frac{1}{\epsilon} \left(\frac{\partial S}{\partial \phi_i(\tau + \epsilon)} - \frac{\partial S}{\partial \phi_i(\tau)} \right) + O(\epsilon) .$$

Introduzindo as expressões acima em 3.48

$$\phi_i(\tau + \epsilon) = \phi_i(\tau) + \epsilon \pi_i(\tau) - \frac{\epsilon^2}{2} \frac{\partial S}{\partial \phi_i(\tau)} + O(\epsilon^3) \quad (3.52)$$

$$\begin{aligned} \pi_i(\tau + \epsilon) &= \pi_i(\tau) - \epsilon \frac{\partial S}{\partial \phi_i(\tau)} + \frac{\epsilon^2}{2} \left[-\frac{1}{\epsilon} \left(\frac{\partial S}{\partial \phi_i(\tau + \epsilon)} - \frac{\partial S}{\partial \phi_i(\tau)} \right) \right] + O(\epsilon^3) \\ \pi_i(\tau + \epsilon) &= \pi_i(\tau) - \epsilon \frac{\partial S}{\partial \phi_i(\tau)} + \frac{\epsilon}{2} \frac{\partial S}{\partial \phi_i(\tau)} - \frac{\epsilon}{2} \frac{\partial S}{\partial \phi_i(\tau + \epsilon)} + O(\epsilon^3) \\ \pi_i(\tau + \epsilon) + \frac{\epsilon}{2} \frac{\partial S}{\partial \phi_i(\tau + \epsilon)} &= \pi_i(\tau) - \epsilon \frac{\partial S}{\partial \phi_i(\tau)} + \frac{\epsilon}{2} \frac{\partial S}{\partial \phi_i(\tau)} + O(\epsilon^3) \end{aligned} \quad (3.53)$$

após alguns rearranjos obtemos

$$\begin{aligned} \phi_i(\tau + \epsilon) &= \phi_i(\tau) + \epsilon \left[\pi_i(\tau) - \frac{\epsilon}{2} \frac{\partial S}{\partial \phi_i(\tau)} \right] + O(\epsilon^3) , \\ \left[\pi_i(\tau + \epsilon) + \frac{\epsilon}{2} \frac{\partial S}{\partial \phi_i(\tau + \epsilon)} \right] &= \left[\pi_i(\tau) - \frac{\epsilon}{2} \frac{\partial S}{\partial \phi_i(\tau)} \right] + O(\epsilon^3) . \end{aligned} \quad (3.54)$$

Mas a quantidade que aparece dentro dos parênteses é o momento resolvido no ponto médio dos intervalos de tempo. Portanto, até $O(\epsilon^3)$, as equações 3.54 são equivalentes ao seguinte par de equações

$$\phi_i(\tau + \epsilon) = \phi_i(\tau) + \epsilon \pi_i(\tau + \epsilon/2) , \quad (3.55)$$

$$\pi_i \left(\tau + \frac{3}{2} \epsilon \right) = \pi_i \left(\tau - \frac{\epsilon}{2} \right) , \quad (3.56)$$

onde a cada iteração a integração das equações de movimento de Hamilton 3.6 é feita de acordo com o método de “leapfrog”: enquanto as coordenadas são solucionadas no tempo $\tau_n = n\epsilon$, os momentos (isto é, as derivadas de ϕ_i) são calculados no ponto médio dos intervalos de tempo.

No algoritmo híbrido, os momentos são atualizados no começo de toda iteração de dinâmica molecular. Supondo que iniciamos a integração pelo método de leapfrog no tempo

τ e utilizando $\{\phi_i(\tau)\}$ como as coordenadas neste tempo, obtemos os momentos $\{\pi_i(\tau)\}$ que são escolhidos através de uma amostra gaussiana conforme foi descrito anteriormente. Entretanto, para começar o processo de iteração em 3.7, é preciso saber $\pi_i(\tau - \epsilon/2)$, que pode ser obtido através da execução da metade de um passo temporal da forma descrita abaixo

$$\pi_i\left(\tau - \frac{\epsilon}{2}\right) = \pi_i(\tau) - \frac{\epsilon}{2} \frac{\partial S[\phi]}{\partial \phi_i(\tau)} + O(\epsilon^2) , \quad (3.57)$$

o que leva uma margem de erro $O(\epsilon^2)$. Perceba que este erro é feito apenas uma vez para cada nova geração de trajetória da dinâmica molecular, e que o erro acumulado na trajetória após $1/\epsilon$ passos é da ordem de ϵ^2 . Então, os cálculos dos observáveis estarão corretos até esta ordem.

Antes de também explicitar os passos requeridos para atualizar as configurações de $\{\phi_i\}$ de acordo com o algoritmo híbrido, as equações 3.7 e 3.57 são reescritas de uma forma apropriada para os cálculos numéricos. Denotando por $\{\phi_i(n)\}$ e $\{\pi_i(n)\}$ o momento no passo temporal $n\epsilon$ e por $\{\tilde{\pi}_i(n)\}$ o momento no tempo $(n - 1/2)\epsilon$, as equações 3.7 e 3.57 se tornam

$$\phi_i(n+1) = \phi_i(n) + \epsilon \tilde{\pi}_i(n) , \quad (3.58)$$

$$\tilde{\pi}_i(n+1) = \tilde{\pi}_i(n) - \epsilon \frac{\partial S[\phi]}{\partial \phi_i(n+1)} , \quad (3.59)$$

onde

$$\tilde{\pi}_i(n) = \pi_i(n) - \frac{\epsilon}{2} \frac{\partial S[\phi]}{\partial \phi_i(n)} . \quad (3.60)$$

Na formulação da rede, todas as quantidades que aparecem em 3.7 são adimensionais. Podemos definir a implementação do algoritmo híbrido através dos seguintes passos (ver fluxograma na Figura 3.1):

1. Escolha a coordenada $\{\phi_i\}$ em algum caminho arbitrário.
2. Escolha um conjunto de momentos $\{\pi_i\}$ da amostra gaussiana 3.48.
3. Execute o meio passo 3.60 para iniciar o processo de integração.
4. Itere as equações 3.58 e 3.59 por vários passos temporais, e armazene as configurações geradas neste caminho.

5. Volte ao item 1 e repita os passos de 2 à 4.

Figura 3.1 – Fluxograma sobre o procedimento adotado no algoritmo Híbrido.

Como foi mostrado por Duane e Kogut (23), este algoritmo gera uma sequência de configurações $\{\phi_i\}$ que são eventualmente distribuídas de acordo com $\exp(-S[\phi])$.

Note que se ao invés de atualizar o momento apenas uma vez no passo 4 ele for atualizado em cada passo temporal, a equação 3.59 não será mais utilizada para a geração das coordenadas $\{\phi_i\}$, que agora serão atualizadas de acordo com

$$\phi_i(n+1) = \phi_i(n) - \frac{\epsilon^2}{2} \frac{\partial S[\phi]}{\partial \phi_i(n)} + \epsilon \eta_i(n) , \quad (3.61)$$

onde $\{\eta_i(n)\}$ são números aleatórios com distribuições gaussianas independentes. Mas isso nada mais é do que além a equação de Langevin com passo temporal igual a $\epsilon^2/2$. Portanto, o método híbrido difere da aproximação de Langevin apenas se o número de passos da dinâmica molecular for maior do que um.

Apesar de o algoritmo híbrido apresentar uma melhoria em relação aos algoritmos anteriores, ele apresenta alguns problemas no controle dos erros sistemáticos introduzidos ao utilizar um passo temporal finito. De fato, de todos os algoritmos mencionados até aqui, apenas o método de Metropolis era totalmente livre de erros sistemáticos. O problema é que o algoritmo de Metropolis se torna muito lento quando se atualizam as configurações com uma ação que depende dos campos de forma não-local. Por outro lado, o algoritmo

híbrido permite uma atualização de todas as variáveis de uma só vez, e acaba sendo o melhor para o estudo de sistemas que possuem ação não-local (como no caso da QCD na inclusão dos férmions). Nossa ideia central é portanto tentar eliminar os erros sistemáticos no algoritmo híbrido com a combinação de testes de aceitação de Metropolis. Isto é alcançado no algoritmo de Monte Carlo Híbrido descrito a seguir.

3.8 Algoritmo de Monte Carlo Híbrido

O método foi proposto inicialmente por Scalettar, Scalapino e Sugar (24) para o algoritmo de Langevin, e aplicado depois à QCD Completa por Gottlieb et al. (25). Seguindo a ideia Duane, Kennedy, Pendleton e Roweth (26) sugeriram modificações similares no algoritmo Híbrido, introduzindo um teste de aceitação de Metropolis entre o passo 4 e 5 do método híbrido discutido na seção anterior. Ou seja, a configuração do espaço de fase gerada no final de cada passo de dinâmica molecular é utilizada como amostra de configuração em um teste de Metropolis com a Hamiltoniana 3.30, agindo como discutido na seção 3.4. Por esta razão este algoritmo é chamado de Monte Carlo Híbrido. As regras para a atualização das configurações de $\{\phi_i\}$ serão apresentadas de forma que elas possam gerar uma cadeia de Markov com balanço detalhado (ver fluxograma na figura 3.2).

1. Escolha as coordenadas $\{\phi_i\}$ em alguma direção arbitrária.
2. Escolha o momento $\{\pi_i\}$ de uma amostra gaussiana 3.48.
3. Calcule $\tilde{\pi}_i$ de acordo com 3.60
4. Itere as equações 3.58 e 3.59 por alguns passos temporais. Seja $\{\phi'_i, \pi'_i\}$ a última configuração gerada na cadeia da dinâmica molecular, onde $\tilde{\pi}'_i$ é gerado de acordo com 3.60.

5. Aceite $\{\phi'_i, \pi'_i\}$ como nova configuração com probabilidade

$$p = \min \left(1, \frac{e^{-H[\phi', \pi']}}{e^{-H[\phi, \pi]}} \right) ,$$

onde H é o hamiltoniano 3.30.

6. Se a configuração $\{\phi'_i, \pi'_i\}$ não for aceita, mantenha a configuração anterior $\{\phi_i, \pi_i\}$, e repita os passos, começando de 2. Caso contrário, use as coordenadas $\{\phi'_i\}$ para gerar uma nova configuração inicial e reinicie o processo a partir do segundo passo.

Figura 3.2 – Fluxograma do comportamento do algoritmo do Método de Monte Carlo Híbrido.

Note que se as equações de movimento de Hamilton pudessem ser integradas de forma exata, então H seria uma constante ao longo da trajetória de dinâmica molecular e a configuração sempre seria aceita no teste de Metropolis. Entretanto, para um determinado ϵ finito, temos $\delta H \neq 0$ e o teste de Metropolis acaba eliminando os erros sistemáticos introduzidos pelo passo temporal finito. Este algoritmo leva a uma probabilidade de transição $P(\phi \rightarrow \phi')$ que satisfaz a equação de balanço detalhado 3.14 para um passo temporal arbitrário ϵ . A prova vem logo a seguir.

Vamos definir $\{\phi_i\}$ como as coordenadas iniciais da cadeia de dinâmica molecular e considerar como uma rodada os vários passos requeridos para a atualização da configuração de acordo com o algoritmo híbrido.

Passo 1

Escolha um conjunto de momentos $\{\pi\}$ de uma amostra gaussiana com valor médio zero e variância unitária. A densidade de probabilidade correspondente é dada por

$$P_G(\pi) = N_0 e^{-\frac{1}{2} \Sigma \pi^2} , \quad (3.62)$$

onde N_0 é a constante de normalização.

Passo 2

Desenvolva a configuração no espaço de fase (ϕ, π) (por simplicidade não será escrito o índice i em ϕ_i e π_i), deterministicamente para cada N passos temporais de acordo com 3.7, incluindo a metade do passo temporal feito para a inicialização e no final da cadeia da dinâmica molecular. Denote a configuração gerada desta forma por $(\phi^{(N)}, \pi^{(N)})$. Aqui $(\phi^{(N)}$ e $\pi^{(N)})$ são funções de inicialização da configuração (ϕ, π) . Note que, dado que 3.45 envolve uma derivada segunda no tempo, o movimento de $\phi \rightarrow \phi^{(N)}$ pode ser revertido pela mudança de sinal de todos os momentos no início e no final da configuração no espaço de fase. Então, a probabilidade para uma transição $(\phi, \pi) \rightarrow (\phi', \pi')$ é a mesma que teremos se formos de $(\phi', -\pi') \rightarrow (\phi, -\pi)$, isto é

$$P_{DM}[(\phi, \pi) \rightarrow (\phi', \pi')] = P_{DM}[(\phi', -\pi') \rightarrow (\phi, -\pi)] , \quad (3.63)$$

onde $_{DM}$ indica o uso da dinâmica molecular. Desde que o sistema evolua deterministicamente, o estado (ϕ', π') gerado pela cadeia de dinâmica molecular irá ocorrer com probabilidade unitária. Então, P_{DM} é uma função δ , assegurando que (ϕ', π') seja obtida a partir de (ϕ, π) .

Passo 3

Aceitando a configuração (ϕ', π') com probabilidade

$$P_A[(\phi, \pi) \rightarrow (\phi', \pi')] = \min \left(1, \frac{e^{-H[\phi', \pi']}}{e^{-H[\phi, \pi]}} \right) , \quad (3.64)$$

obtendo o produto da densidade de probabilidade para estes três passos e integrando sobre os momentos, temos que a transição obedece a densidade de probabilidade

$$P(\phi \rightarrow \phi') = \int D\pi D\pi' P_G(\pi) P_{DM}[(\phi, \pi) \rightarrow (\phi', \pi')] P_A[(\phi, \pi) \rightarrow (\phi', \pi')] . \quad (3.65)$$

Consideremos agora o produto $e^{-S(\phi)} P(\phi \rightarrow \phi')$, correspondente ao lado esquerdo da equação 3.14. Introduzindo a expansão 3.62 dentro da equação 3.65, o produto pode ser escrito da forma

$$e^{-S(\phi)} P(\phi \rightarrow \phi') = \int D\pi D\pi' e^{-H[\phi', \pi']} P_{DM}[(\phi, \pi) \rightarrow (\phi', \pi')] P_A[(\phi, \pi) \rightarrow (\phi', \pi')] . \quad (3.66)$$

Da definição em 3.64 segue que

$$e^{-H[\phi, \pi]} P_A[(\phi, \pi) \rightarrow (\phi', \pi')] = e^{-H[\phi', \pi']} P_A[(\phi', \pi') \rightarrow (\phi, \pi)] , \quad (3.67)$$

que é a prova do balanço detalhado no espaço de fase para o teste de aceitação do passo de Metropolis. Inserindo esta expressão em 3.66, e fazendo uso de 3.63 e do fato que $H[\phi, \pi] = H[\phi, -\pi]$, encontramos

$$\begin{aligned} e^{-S(\phi)} P(\phi \rightarrow \phi') &= e^{-S(\phi')} \int D\pi D\pi' \{P_G(\pi') P_{DM}[(\phi', \pi') \rightarrow (\phi, \pi)] \times P_A[(\phi', \pi') \rightarrow (\phi, \pi)]\} \\ &= e^{-S(\phi')} P(\phi' \rightarrow \phi) . \end{aligned} \quad (3.68)$$

O algoritmo Monte Carlo Híbrido então gera uma configuração da cadeia de Markov $\{\phi_i\}$ distribuída eventualmente de acordo com $\exp[-S(\phi)]$.

Podemos perceber que toda a configuração do campo é atualizada de uma só vez. No caso de o número de passos da dinâmica molecular não ser muito grande ou no caso de ϵ não ser pequeno o suficiente, pode-se ter o problema da aceitação, conforme mencionado na Seção 3.5. Em geral escolhemos a probabilidade de aceitação 3.64 pequena, o que faz o sistema se mover lentamente pelo espaço de configurações. De qualquer modo, experimentos numéricos mostram que mesmo para ϵ pequeno o sistema avança no espaço de configurações pelo menos tão rapidamente quanto o algoritmo híbrido. A grande vantagem do Monte Carlo híbrido é que ele é livre de erros sistemáticos devidos a passos temporais finito, que elimina a não necessidade de extrapolação de dados, que por sua vez tem custo de tempo e é uma fonte adicional de erros.

3.9 Método do Pseudo-Férmion

Os algoritmos discutidos nas subseções anteriores podem ser utilizados para calcular médias de ensemble de funcionais que dependam de variáveis numéricas, mas eles não podem ser aplicados diretamente a sistemas com campos de Grassmann (ou seja, variáveis com a propriedade de anti-comutação), utilizados no estudo da QCD para a representação dos férmions (i.e. os quarks). Desta forma, precisamos integrar os graus de liberdade correspondentes à parte fermiônica, para poder executar um cálculo de Monte Carlo na QCD. Isso pode ser feito sempre que tivermos uma função de correlação envolvendo o produto de um número arbitrário de campos de férmions, contanto que a ação seja quadrática nas variáveis de Grassman. Para facilitarmos os cálculos, será considerada a formulação de Wilson (ou seja, a discretização mais simples) para os férmions. A média de ensemble de um observável $O[U, \psi, \bar{\psi}]$ é dada por

$$\langle O \rangle = \frac{\int DU \langle O \rangle_{S_F} e^{-S_{ef}[U]}}{\int DU e^{-S_{ef}[U]}} , \quad (3.69)$$

onde

$$\langle O \rangle_{S_F} = \frac{\int D\bar{\psi} D\psi O[U, \psi, \bar{\psi}] e^{-S_F^{(W)}[U, \psi, \bar{\psi}]}}{\int D\bar{\psi} D\psi e^{-S_F^{(W)}[U, \psi, \bar{\psi}]}} \quad (3.70)$$

é escrita em termos da função de Green calculada em um campo de fundo $U = \{U_\mu(n)\}$ e onde

$$S_{ef}[U] = S_G[U] - \ln [\det \tilde{K}[U]] . \quad (3.71)$$

Temos que S_G é a ação na teoria pura de Gauge e $\tilde{K}[U]$ é a matriz que aparece com a contribuição fermiônica na ação após a integração formal dos graus de liberdade fermiônicos, usando propriedades das variáveis de Grassmann. No caso da QCD com N_f sabores de massas de quarks degenerados, os elementos de matriz de $\tilde{K}[U]$ são dados por

$$\tilde{K}_{n\alpha f', m\beta b}[U] = K_{n\alpha a, m\beta b}[U] \delta_{ff'} , \quad (3.72)$$

onde

$$\begin{aligned}
K_{n\alpha a, m\beta b}[U] &= (\hat{M}_0 + 4r) \delta_{nm} \delta_{\alpha\beta} \delta_{ab} \\
&- \frac{1}{2} \sum_{\mu} \{ (r - \gamma_{\mu})_{\alpha\beta} [U_{\mu}(n)]_{ab} \delta_{n+\hat{\mu}, m} \\
&+ (r + \gamma_{\mu})_{\alpha\beta} [U_{\mu}^{\dagger}(m)]_{ab} \delta_{n-\hat{\mu}, m} \} .
\end{aligned} \tag{3.73}$$

e δ_{ij} é a função delta de Kronecker. A formulação de Wilson é definida (9, cap. 4) pelos parâmetros $\hat{M}_0$ (que define a massa em unidades de rede) e r (ou pelo parâmetro de hopping κ). O índice $f(f')$ na equação 3.72 indica os graus de liberdade do sabor, os índices $n, \alpha, a(m, \beta, b)$ denotam o ponto da rede e os graus de liberdade de Dirac e de cor, respectivamente. Vamos denotar este tripleto por um único simbolo coletivo i . Desta forma, a contribuição fermiônica na ação leva a

$$S_F^{(W)} = \sum_{i,j} \bar{\psi}_i^{f'} K_{ij}[U] \delta_{f'f} \psi_j^f ,$$

onde $K_{ij}[U]$ é independente do sabor. Segue que

$$\det \tilde{K}[U] = (\det K[U])^{N_f} . \tag{3.74}$$

Uma importante propriedade da matriz $K[U]$ é que seu determinante é real, isto é, $\det K[U] = \det K^{\dagger}[U]$. Além disso, $\det K[U] > 0$ para valores do parâmetro de hopping $\kappa = 1/(8r + 2\hat{M}_0)$ menores do que $1/8$. A ideia do método de pseudo-férmions é que o $\det K$ pode ser escrito na forma

$$\det K[U] = \sqrt{\det Q[U]} , \tag{3.75}$$

onde

$$Q[U] = K^{\dagger}[U] K[U] \tag{3.76}$$

é agora uma matriz hermitiana positiva. Seguindo as equações 3.74 e 3.75, vemos que 3.69 é dada pela média ensemble de $\langle O \rangle_{S_F}$, definida em 3.70, calculada com o fator de Boltzmann $\exp(-S_{ef})$ onde

$$S_{ef}[U] = S_G[U] - \ln [\sqrt{\det Q[U]}]^{N_f} = S_G[U] - \frac{N_f}{2} \ln [\det Q[U]] . \tag{3.77}$$

Agora consideraremos que será utilizado o algoritmo de Metropolis para fazer a atualização da configuração das variáveis de ligação U . Então, o teste de aceitação/rejeição

de Metropolis requer que seja calculada a diferença $S_{ef}[U'] - S_{ef}[U]$, onde U denota a configuração original a ser atualizada e U' é a sugestão da nova configuração. Será preciso calcular a fração

$$\rho(U, U') = \frac{\det Q[U']}{\det Q[U]} . \quad (3.78)$$

Embora $Q[U]$ seja uma (grande) matriz esparsa, construída a partir de $K[U]$ com apenas poucos pares de vizinhos próximos, o cálculo exato de $\rho(U, U')$ é muito lento para ser utilizado como atualização das configurações de ligação em redes grandes, e será preciso encontrar uma forma mais rápida para o cálculo dos dados 3.78.

Uma proposta feita inicialmente por Fucito et al. (27) consiste em formar δU como sendo a diferença entre a configuração da variável de ligação e a antiga configuração, isto é, $\delta U \equiv \{\delta U_\mu(n)\}$, onde $\delta U_\mu = U'_\mu(n) - U_\mu(n)$. A mudança correspondente na matriz $Q[U]$ é chamada por δQ : $\delta Q = Q[U'] - Q[U]$. Então

$$\rho(U, U') = \frac{\det(Q + \delta Q)}{\det Q} = \det(1 + Q^{-1} \delta Q) . \quad (3.79)$$

Para $Q^{-1}\delta Q$ pequeno (i.e. mudanças pequenas), a expressão acima pode ser linearizada em δQ :

$$\rho(U, U') \approx 1 + \text{Tr} Q^{-1} \delta Q . \quad (3.80)$$

Isto acaba sendo uma aproximação razoável se as variáveis de ligação são atualizadas uma a uma e se apenas pequenas mudanças em δU são permitidas. Dessa maneira, a computação do traço fica facilitada, mas Q^{-1} precisa ser resolvida em cada atualização, o que continua consumindo muito tempo. Uma forma de resolver seria utilizar a mesma matriz $Q^{-1}[U]$ para toda varredura da rede. Isto é consistente com a aproximação linearizada em 3.80, mas pode resultar na propagação de grandes erros sistemáticos, que seriam introduzidos a cada passo de atualização e que se acumulariam durante a varredura. A aproximação para a resolução exata de Q^{-1} nas configurações das variáveis de ligação geradas após cada varredura da rede não é fácil de se realizar, mas temos uma forma de determinar a matriz Q^{-1} aproximadamente, notando que seus elementos são dados pela seguinte média de ensemble

$$Q_{ij}^{-1} = \langle \phi_i \phi_j^* \rangle = \frac{\int D\phi^* D\phi \phi_i \phi_j^* e^{-\sum_{i,j} \phi_i^* Q_{ij} \phi_j}}{\int D\phi^* D\phi e^{-\sum_{i,j} \phi_i^* Q_{ij} \phi_j}} . \quad (3.81)$$

Aqui ϕ e ϕ^* são chamados de variáveis pseudo-fermiônicas (onde para cada grau de liberdade dos férmions temos uma correspondência com um par de variáveis complexas). Por Q ser uma matriz hermitiana positiva, a exponencial que aparece em 3.81 pode ser interpretada como um fator de Boltzmann. Isto explica o motivo de expressamos $\det K[U]$ na forma 3.75.

De acordo com 3.81, pode-se calcular a matriz inversa Q^{-1} através

1. da geração de um conjunto complexo de variáveis pseudo-fermiônicas com densidade de probabilidade $\exp(-\phi^* Q \phi)$.
2. do cálculo dos produtos $\phi_i \phi_j^*$ para cada configuração gerada.
3. da média de ensemble.

Isso mostra que o maior trabalho consiste na geração destas configurações.

Um procedimento alternativo foi proposto por Petcher e Weingarten (28), onde o determinante de Q é obtido através da seguinte integral de trajetória sobre variáveis pseudo-fermiônicas

$$\det Q = \frac{1}{\det Q^{-1}} = \int D\phi^* D\phi e^{-\sum_{i,j} \phi_i^* Q_{ij}^{-1} \phi_j} .$$

Como pode ser visto em 3.77, o método é relevante no caso de dois sabores de quarks. A função de partição correspondente pode ser escrita da seguinte forma

$$\begin{aligned} Z &= \int DU \det Q[U] e^{-S_G[U]} \\ &= \int DU D\phi^* D\phi e^{-S_G[U] - \sum_{i,j} \phi_i^* Q_{ij}^{-1}[U] \phi_j} . \end{aligned} \quad (3.82)$$

Isto nos mostra como calcular as médias de ensemble, uma vez que é preciso gerar configurações com uma ação não-local. Então a eficácia do método depende da eficiência do algoritmo para o cálculo da matriz inversa de $Q[U]$.

Uma alteração bastante interessante foi proposta por Bhanot, Heller e Stamatescu (29), que não apresenta a sobrecarga de erros sistemáticos, por desprezar as correções de

ordens mais altas em δU . A sugestão feita é de que apenas um cálculo da fração 3.78 deva ser realizado utilizando diretamente o método pseudo-fermiônico. Já que

$$\frac{1}{\det Q[U]} = \int D\phi^* D\phi e^{-\sum_{i,j} \phi_i^* Q_{ij}[U] \phi_j} , \quad (3.83)$$

a razão em 3.79 é dada pela seguinte expressão

$$\rho(U, U') = \frac{\int D\phi^* D\phi e^{-\phi^\dagger Q \phi}}{\int D\phi^* D\phi e^{-\phi^\dagger \delta Q \phi} e^{-\phi^\dagger Q \phi}} , \quad (3.84)$$

onde $\delta Q = Q' - Q$, com $Q' = Q[U']$, $Q = Q[U]$, além de ter sido utilizada a notação matricial para simplificar a expressão. De acordo com 3.84, ρ é dado pelo inverso da média de ensemble de $\exp(-\phi^\dagger \delta Q \phi)$, calculada com o fator de Boltzmann $\exp(-S_Q)$, onde $S_Q = \phi^\dagger Q \phi$ é uma expressão local para as variáveis de ligação

$$\rho = \frac{1}{\langle \exp(-\phi^\dagger \delta Q \phi) \rangle_{S_Q}} . \quad (3.85)$$

Alternativamente, ρ pode ser dado também por

$$\rho = \langle \exp(\phi^\dagger \delta Q \phi) \rangle_{S_{Q'}} , \quad (3.86)$$

onde $S_{Q'}$ é a ação pseudo-férmionica para a variável de ligação proposta U' . Para argumentos pequenos dentro da exponencial, nós podemos substituir ρ por

$$\rho \approx 1 + \sum_{i,j} \langle \phi_i^* \phi_j \rangle_{S_Q} \delta Q_{ij} .$$

Fazendo uso de 3.81 temos

$$\rho \approx 1 + \text{Tr} Q^{-1} \delta Q ,$$

que coincide com 3.80

As vantagens de fazer essa modificação são que não ficamos limitados a pequenas mudanças nas variáveis de ligação (os erros introduzidos são apenas os estatísticos) e que a precisão do método pode ser estimada pelo cálculo da fração do determinante em 3.85 ou em 3.86. Embora a simulação direta dos campos pseudo-fermiônicos possa ser feita como descrito acima, na prática será mais eficiente aplicar os métodos descritos no início do capítulo, utilizando-se a discussão acima para integração formal dos campos de férmions. Nas seções abaixo descreveremos aplicações do método de Monte Carlo Híbrido, que é o mais amplamente usado em simulações de QCD na rede, discutindo suas versões mais recentes.

3.10 Aplicações do Método de Monte Carlo Híbrido

Vamos agora discutir a implementação do Monte Carlo Híbrido no caso da QED (ver introdução da Seção 2.4) por simplicidade.

Antes de aplicar o algoritmo para calcular a função de correlação, nós primeiramente devemos escrever a integral de trajetória na forma de uma média de ensemble calculada sobre variáveis bosônicas, onde temos que a função de partição relevante é dada por 3.82, sendo Q a matriz hermitiana positiva definida em 3.76.

O próximo passo consiste em reescrever a função de partição como uma integral de trajetória no espaço de fase. Dessa forma, são introduzidos os momentos π_i^*, π_i e P_l canonicamente conjugados com ϕ_i, ϕ_i^* e A_l , onde A_l é a variável angular que parametriza a ligação indexada por l , onde l relaciona a coordenada de um site da rede n e a direção μ de uma determinada variável de ligação, e U_l e A_l são relacionados por $U_l = \exp(iA_l)$. Define-se o “Hamiltoniano” como

$$H = \frac{1}{2} \sum_l P_l^2 + \sum_i \pi_i^* \pi_i + S_G[U] + S_{PF}[U, \phi, \phi^*] , \quad (3.87)$$

onde S_{PF} é a ação pseudo-fermiônica

$$S_{PF} = \sum_{i,j} \phi_i^* Q_{ij}^{-1}[U] \phi_j , \quad (3.88)$$

e S_G é a ação da teoria pura de gauge $U(1)$. Assim a função de partição pode ser escrita no espaço de fase como

$$Z = \int DU D\phi D\phi^* DP D\pi D\pi^* e^{-H} .$$

Na equação de Hamilton do movimento derivada de 3.87, é descrita a dinâmica do sistema em um novo tempo τ , que é identificado como o tempo de simulação no cálculo de Monte Carlo. Essas equações são dadas por

$$\dot{\phi}_i(\tau) = \pi_i(\tau) , \quad (3.89)$$

$$\dot{\pi}_i(\tau) = - \sum_j Q_{ij}^{-1}[U] \phi_j(\tau) , \quad (3.90)$$

$$\dot{A}_l(\tau) = P_l(\tau) , \quad (3.91)$$

$$\dot{P}_l(\tau) = -\frac{\partial S_G}{\partial A_l(\tau)} - \sum_{i,j} \phi_i^*(\tau) \frac{\partial Q_{ij}^{-1}[U]}{\partial A_l(\tau)} \phi_j(\tau) . \quad (3.92)$$

Em termos das variáveis de ligação, a equação 3.91 se torna

$$\dot{U}(\tau) = i P_l(\tau) U(\tau) .$$

A derivada de Q^{-1} que aparece em 3.92 pode ser escrita em uma forma mais conveniente usando o fato que Q é dado pelo produto em 3.76, e que $\partial K^{-1}K/\partial A_l = \partial(K^{\dagger-1}K^{\dagger})/\partial A_l = 0$. Temos portanto

$$\frac{\partial Q^{-1}}{\partial A_l} = Q^{-1} \frac{\partial Q}{\partial A_l} Q^{-1}.$$

Introduzindo essa expressão em 3.92, observa-se que as equações em 3.10 são equivalentes ao seguinte conjunto

$$\eta_i = \sum_j Q_{ij}^{-1}[U] \phi_j , \quad (3.93)$$

$$\dot{\phi}_i = \pi_i , \quad (3.94)$$

$$\dot{\pi}_i = -\eta_i , \quad (3.95)$$

$$\dot{U}_l = i P_l U_l , \quad (3.96)$$

$$\dot{P}_l = -\frac{\partial S_G}{\partial A_l} - \sum_{i,j} \eta_i^* \frac{\partial Q_{ij}}{\partial A_l} \eta_j . \quad (3.97)$$

Em princípio essas equações seriam utilizadas na parte dos passos de dinâmica molecular do algoritmo de Monte Carlo Híbrido, mas isso não é feito na prática pois existe uma forma mais rápida de se gerar um conjunto de configurações com variáveis de ligação e os pseudo-férmions. Na ação pseudo-fermiônica, o inverso da matriz Q é dado pelo produto $K^{-1}(K^{\dagger})^{-1}$. Portanto podemos escrever a ação na forma $S_{PF} = \sum_i \xi_i^{\dagger} \xi_i$, onde o vetor $\xi = (\xi_1, \xi_2, \dots)$ é construído de $\phi = (\phi_1, \phi_2, \dots)$ como se segue

$$\xi = (K^{\dagger} [U]^{-1}) \phi .$$

Agora pode-se facilmente gerar as configurações com estas novas variáveis que são distribuídas de acordo com $\exp(\xi^{\dagger} \xi)$. Calculando $\phi = K^{\dagger}[U] \xi$, obtém-se para um $U = \{U_l\}$ fixo uma amostra de configurações sobre variáveis pseudo-fermiônicas distribuídas de acordo

com $\exp(-\phi^\dagger Q^{-1} \phi)$. Cada uma destas configurações pode ser usada como um campo de fundo no processo da dinâmica molecular nas variáveis de ligação. Temos então a seguinte descrição do procedimento para a implementação do método de Monte Carlo Híbrido com férmions.

1. Escolher uma configuração inicial das variáveis de ligação.
2. Escolher P_l de uma amostra gaussiana com fator de Boltzmann $\exp(-\frac{1}{2} \sum_l P_l^2)$.
3. Escolher ξ como um campo de ruído gaussiano.
4. Calcular

$$\phi = K^\dagger[U] \xi .$$

5. Permitir que as variáveis de ligação e o momento canônico P_l entre si (evolução determinística) interajam como é descrito em 3.96 e 3.97, onde $\eta = Q^{-1}[U] \phi$, com ϕ mantido fixo.
6. Aceitar a nova configuração (A', P') gerada pelo processo da dinâmica molecular com a probabilidade

$$p = \min \left(1, \frac{e^{-\tilde{H}[A', P']}}{e^{-\tilde{H}[A, P]}} \right) ,$$

onde $\tilde{H}$ é o hamiltoniano governado pela dinâmica das variáveis de ligação na presença do campo de fundo ϕ :

$$\tilde{H}[A, P_l] = \frac{1}{2} \sum_l P_l^2 + S_G[U] + \sum_{i,j} \phi_i^* Q_{ij}^{-1}[U] \phi_j .$$

7. Armazenar a nova configuração gerada, ou a antiga, utilizando o teste de Metropolis.
8. Retornar ao passo 2.

Este é o princípio de funcionamento das versões atuais do método, sendo que as variáveis de ligação devem pertencer ao grupo $SU(3)$ no caso da QCD. Descreveremos a seguir algumas versões do método.

fermiônico Φ é dada por

$$P(U, \Phi) = \frac{1}{Z} e^{-[S_G(U) + \Phi^* (M^\dagger M)^{-1} \Phi]} = e^{-S_{ef}} , \quad (3.98)$$

onde M é o operador Dirac staggered com o campo pseudo-fermiônico definido apenas nos sítios pares. O algoritmo Φ utiliza o conceito do Monte Carlo Híbrido 3.8 para quatro “sabores” de staggered férmions, onde nós introduzimos o momento conjugado π e definimos o Hamiltoniano H ,

$$P(U, \Phi, \pi) = \frac{1}{Z'} e^{-[\frac{1}{2}\pi^2 + S_{ef}]} = \frac{1}{Z'} e^{-H(U, \Phi, \pi)} . \quad (3.99)$$

A ação S_{ef} utiliza a regra do potencial no Hamiltoniano. O campo de gauge U pode ser equacionado através do tempo τ pela integração das equações de Hamilton, utilizando o esquema de integração da Dinâmica Molecular (visto em 3.6). Como descrito anteriormente, cada trajetória da Dinâmica Molecular envolve em uma atualização do momento pelo método do banho térmico (3) através do uso de um ruído gaussiano. Em seguida o pseudo-férmion é atualizado utilizando novamente o banho térmico com um ruído gaussiano, e a trajetória de dinâmica molecular é calculada por meio de $\tau/\delta\tau$ passos.

Note que não temos a conservação da energia, pois $\delta H = O(\delta\tau^2)$ para qualquer comprimento de trajetória.

Para $\delta\tau > 0$, o ponto fixo da distribuição no passo de dinâmica molecular e a atualização do momento através do banho térmico não coincidem. O que é preciso encontrar é a distribuição de equilíbrio atual que surge da composição desses dois passos. Como são descartados os novos momentos e pseudo-férmion em cada passo, somente é considerado como o passo completo de Markov a atualização do campo de gauge U , integrando com os campos auxiliares π e Φ . Vamos fazer $V(\tau)$ representar o operador de evolução para o passo de dinâmica molecular.

Seja $V(\tau)$ tal que $(U, \pi) \mapsto (U'', \pi'')$ e denota-se por $e^{-(S+\Delta S)}$ o ponto fixo da distribuição para o passo completo de Markov, onde ΔS mede o desvio da distribuição desejada,

desde que satisfaça

$$\begin{aligned}
e^{-[S(U') + \Delta S(U')]} &= \int DU D\pi e^{-H(U, \pi) - \Delta S(U)} \delta(U' - U) \\
&= \int DU'' D\pi'' e^{-(H + \Delta S) \circ V^{-1}} \delta(U' - U'') \\
&= \int DU'' D\pi'' e^{-(H + \Delta S)} e^{-\delta(H + \Delta S)} \\
&= e^{-[S(U') + \Delta S(U')]} \langle e^{-\delta(H + \Delta S)} \rangle_\pi
\end{aligned}$$

com δ tal que $\Omega \mapsto \Omega \circ [V(r) - 1]$, que mede a mudança de Ω sobre a trajetória (isto é, δH é a extensão onde a energia não é conservada). É suposta a reversibilidade $V^{-1} = F \circ V \circ F$ onde F é tal que $(U, \pi) \mapsto (U, -\pi)$ e a preservação da área no Jacobiano é unitária. Com isso, obtém-se a condição

$$\langle e^{-\delta(H + \Delta S)} \rangle_\pi = 1 . \quad (3.100)$$

Executando uma expansão assintótica para esta condição em potências de $\delta\tau$, sabendo que $\delta H = O(\delta\tau^2)$ para qualquer comprimento de trajetória τ , deduz-se que $\delta\Delta S \sim O(\delta\tau^2)$. Dessa forma, temos que $\Delta S \sim O(\delta\tau^2)$ para $\tau \gg \delta\tau$, onde a precisão de Φ é de ordem $O(\delta\tau^2)$. Esta é fundamentação do algoritmo Φ .

Ainda antes de descrever o algoritmo R , será mostrada uma versão anterior do mesmo chamada de R_0 , que começa de um ponto de vista totalmente diferente. Ao invés de introduzir pseudo-férmions para substituir o determinante fermiônico, inclui-se o determinante na ação. Isto resulta em

$$S_F = -n \operatorname{tr} \ln (M^\dagger M) , \quad (3.101)$$

onde n é o número múltiplos de férmions ($n = N_f/4$ para férmions staggered). Quando é calculada a contribuição da força fermiônica, um campo estimador de ruído é utilizado no traço.

A composição ergódica do passo de Markov consiste de uma atualização do momento pelo método do banho térmico e uma trajetória de dinâmica molecular é definida por $N \equiv \tau/\delta\tau$ passos com η ruídos independentes utilizados para cada passo.

Para calcular o erro em δH temos que incluir o efeito de se utilizar o estimador de ruído, $\langle \Sigma' \rangle_\eta = S'$, para a força fermiônica. Temos que para um único ruído no passo da

dinâmica molecular

$$\langle e^{-\delta H} \rangle_\eta = \frac{1}{2} \langle (\Sigma' - S')^2 \rangle_\eta (1 - \pi^2) \delta \tau^2 + O(\delta \tau^3) . \quad (3.102)$$

Aqui o coeficiente de $(1 - \pi^2) \delta \tau^2$ é proporcional a variância da força estimada e irá desaparecer apenas se a força calculada for exata. Já que a média do momento não é Gaussiana após vários passos de leapfrog, o termo principal não pode ser cancelado se executarmos ao menos um passo de dinâmica molecular na trajetória. Então, para uma dada trajetória, $\delta H \sim O(\delta \tau)$ e o erro do algoritmo R_0 é portanto da ordem de $\Delta S \sim O(\delta \tau)$.

Se no algoritmo Φ alterarmos a execução do banho térmico sobre o pseudo-férmion para antes de cada passo de dinâmica molecular e compararmos essa nova implementação com a do algoritmo R_0 , temos que a única diferença entre eles é a forma do campo pseudo-fermiônico (feito pelo campo de ruído) no R_0 é calculado no meio de cada passo de dinâmica molecular. Temos que o algoritmo Φ possui erros de ordem $O(\delta \tau^2)$ para $n = 1$ múltiplos, e o R_0 possui erros $O(\delta \tau)$. Entretanto, para $n = 0$ (ou seja, sem nenhum férmion) ambos são idênticos e possuem erros na ordem de $O(\delta \tau^2)$. Esperamos que o erro principal possua uma dependência linear com o tempo em que os pseudo-férmions são atualizados e com o número de múltiplos, e se nós atualizarmos o campo pseudo-férmion no tempo $t = (1 - n) \delta \tau / 2$, um algoritmo da ordem $O(\delta \tau^2)$ para $0 \leq n \leq 1$ múltiplos de férmions pode ser obtido. Para dois “sabores” de férmions staggered, isto significa resolver um quarto do campo pseudo-fermiônico para cada iteração da dinâmica molecular. Obtemos dessa forma um algoritmo não plenamente reversível e que não é executado de forma exata (ao contrário dos algoritmos mencionados nesta seção, que são exatos por causa da inclusão do passo de aceitação e rejeição).

O argumento da equação 3.100 pode ser generalizado para

$$\left\langle e^{(\delta + \bar{\delta})(H + \Delta S) - tr \ln V_0} \right\rangle_\pi = 1 , \quad (3.103)$$

onde δ mede a ausência da conservação de energia, e $\bar{\delta}$ mede a ausência de reversibilidade ($\bar{\delta} : \Omega \rightarrow \Omega \circ [V(\tau)^{-1} - F \circ V \circ F]$) e $tr \ln V_* \equiv \ln \det \frac{\partial(U'', \pi'')}{\partial(U, \pi)}$ mede a ausência de preservação da área. Vamos considerar um único passo do algoritmo R, onde o campo

auxiliar χ é calculado no tempo $t = (1-\alpha)\delta\tau/2$ (α é algum parâmetro a ser determinado), e expandindo a equação 3.103 em $\delta\tau$ será encontrado

$$\left\langle e^{-(\delta+\bar{\delta})(H+\Delta S)-tr \ln V_0} \right\rangle_{\pi} = 1 - A\delta\tau^2 + O(\delta\tau^3) , \quad (3.104)$$

onde A é proporcional a $(n-\alpha)$. Se $\alpha = n$ o termo principal é cancelado, e isto leva ao erro principal na ordem $O(\delta\tau^2)$ para toda a trajetória.

O problema com este método é que por causa da utilização do estimador de ruído para a força fermiônica, um novo erro da ordem $O(\delta\tau)$ é introduzido. Pode ser obtido por meio de uma integração que não preserva a área um algoritmo com erro na ordem de $O(\delta\tau^2)$. Entretanto, isto remove o balanço detalhado e o algoritmo não pode ser executado de forma exata com a inclusão do teste de aceitação de Metropolis e qualquer resultado obtido do mesmo irá ter dependência com o passo de integração.

O custo computacional do algoritmo R é equivalente ao do HMC simples, pois apenas uma equação linear é requerida para o calculo de uma força, mas uma extrapolação para passo zero é requirida, isto acaba tornando o processo custoso e o faz consumir tempo. Por muitos anos, nos cálculos utilizando férmions staggered com algoritmo R tem se utilizado a regra $\delta\tau^R \approx 2/3m$ onde a extrapolação para o passo zero raramente era executada, o que aumentou o questionamento sobre os resultados gerados por esse algoritmo desde que o erro por passo é desconhecido.

3.12 Algoritmo PHMC

Será feito um breve comentário sobre o método de Monte Carlo Híbrido Polinomial (Polynomial Hybrid Monte Carlo (PHMC), do inglês Polynomial Hybrid Monte Carlo), que baseamos no artigo original de R. Frezzotti e K. Jansen (32) e em Clark (31).

Iniciamos pela representação de uma integral de trajetória para férmions de Wilson

na rede,

$$\begin{aligned} Z &= \int DU \exp -S_G \det(Q^2) \\ &= \int DU D\phi^\dagger D\phi e^{-(S_g + \phi^\dagger Q^{-2} \phi)} , \end{aligned} \quad (3.105)$$

onde o termo S_g na exponencial é a ação pura da rede, dada por

$$S_G = -\frac{\beta}{6} \sum_P \text{tr}(U_p + U_p^\dagger) . \quad (3.106)$$

U_p representa o termo usual da plaqueta na rede com ligação dada por elementos do grupo $SU(3)$. O fator do determinante $\det(Q^2)$ representa a contribuição fermiônica na integral de trajetória. O campo bosônico ϕ esta associado aos índices de sabor, spinor e cor. Vamos supor que em 3.105 estamos com um caso em que temos 2 sabores de massas degeneradas. A matriz Q que aparece no determinante é esparsa e hermitiana, definida por

$$Q(U) = c_0 \gamma_5 \left[\delta_{x,y} - \kappa \sum_\mu (1 - \gamma_{\mu\nu}) U_{x,\mu} \delta_{x+\mu,y} + (1 + \gamma_\mu) U_{x-\mu,\mu}^\dagger \delta_{x-\mu,y} \right] , \quad (3.107)$$

onde κ é o parâmetro de hopping, que se relaciona com a massa de quark m_0 por $\kappa = (8 - 2m_0)^{-1}$, e $c_0 = [c_m(1 + 8\kappa)]^{-1}$, onde c_m é uma escolha que leva os autovalores λ de Q a satisfazer $|\lambda| < 1$.

Vamos supor que nós construímos um polinômio P_n de grau n que satisfaz

$$\det [Q^2 P_n(Q^2)] \rightarrow 1 \quad \text{para} \quad n \rightarrow \infty , \quad (3.108)$$

com $P_n(\lambda(Q^2)) > 0$ para todos os autovalores $\lambda(Q^2)$ de Q^2 na faixa $0 \leq \lambda(Q^2) < 1$.

Então podemos reescrever o determinante como

$$\det(Q^2) = \frac{\det[Q^2 P_n(Q^2)]}{\det[P_n(Q^2)]} . \quad (3.109)$$

Cada um dos determinantes do lado direito da equação pode ser representado por uma integral Gaussiana com a ajuda de campos bosônicos ϕ e η , respectivamente. Então a função de partição se torna

$$Z = \int DU D\phi^\dagger D\phi D\eta^\dagger D\eta W e^{-[S_g + \phi^\dagger P_n(Q^2) \phi + \eta^\dagger \eta]} , \quad (3.110)$$

onde W é o “fator de correção” dado por

$$W = \exp \left\{ \eta^\dagger \left[1 - (Q^2 P_n(Q^2))^{-1} \right] \eta \right\}. \quad (3.111)$$

Perceba que a 3.110 é uma forma reescrita da função de partição 3.105.

Agora com a introdução do fator de correção W , o valor esperado de um observável O é computado por

$$\langle O \rangle = \frac{\langle OW \rangle_p}{\langle W \rangle_p}, \quad (3.112)$$

sendo as médias $\langle \dots \rangle_p$ obtidas com respeito às medidas definidas através da aproximação da ação fermiônica $\phi^\dagger P_n(Q^2) \phi$. Alternativamente nós podemos incorporar o fator W no passo de aceitação/rejeição.

Vamos especificar a forma do polinômio que vamos utilizar. Nós escolhemos um polinômio de Chebyshev para aproximar Q^{-2} . Quando o escrevemos na forma fatorizada, temos

$$P_{[n,\epsilon]}(Q^2) = c_N \prod_{k=1}^n [Q^2 - z_k] = c_N \prod_{k=1}^n [(Q - \sqrt{z_k})(Q + \sqrt{z_k})], \quad (3.113)$$

sendo caracterizado pelas raízes (para $k = 1, 2, \dots, n$),

$$z_k = \frac{1}{2}(1 + \epsilon) - \frac{1}{2}(1 + \epsilon) \cos \left(\frac{2\pi k}{n+1} \right) - i\sqrt{\epsilon} \sin \left(\frac{2\pi k}{n+1} \right), \quad (3.114)$$

onde c_N é um fator de normalização. O polinômio $P_{[n,\epsilon]}(s)$ aproxima a função $1/s$ (onde s pode corresponder a qualquer dos autovalores de Q^2) uniformemente no intervalo $\epsilon \leq s \leq 1$. O erro relativo é

$$R_{[n,\epsilon]}(s) = [P_{[n,\epsilon]}(s) - 1/s]s, \quad (3.115)$$

e nesse intervalo ele é exponencialmente pequeno

$$|R_{[n,\epsilon]}(s)| \leq 2 \left(\frac{1 - \sqrt{\epsilon}}{1 + \sqrt{\epsilon}} \right)^{n+1}. \quad (3.116)$$

Finalmente introduzimos o parâmetro de precisão δ , que define qual é o máximo erro permitido em uma aproximação polinomial

$$\delta = 2 \left(\frac{1 - \sqrt{\epsilon}}{1 + \sqrt{\epsilon}} \right)^{n+1}, \quad (3.117)$$

fornecendo uma forma fácil de medir o quão bem a escolha polinomial se aproxima de $1/s$ no intervalo $\epsilon \leq s \leq 1$.

3.13 Algoritmo RHMC

Por fim chegamos ao algoritmo Monte Carlo Híbrido Racional (RHMC - Rational Hybrid Monte Carlo), que foi desenvolvido mais recentemente. Sua apresentação será baseada novamente no artigo de Clark (31) e no artigo de revisão de Kennedy (33).

Utilizar um núcleo racional é uma alternativa ao núcleo polinomial descrito acima, pois aproximações racionais possuem propriedades superiores de convergência. De fato, sobre a faixa espectral em que estamos interessados, nós podemos obter uma aproximação para a função desejada com um erro relativo máximo bom para a precisão das máquinas atuais ($\approx 10^{-15}$) com um grau de aproximação da ordem de $O(20)$ ou menos [comparando com o método polinomial tínhamos $O(1000)$].

Aproximações racionais otimizadas são geradas utilizando o algoritmo de Remez, as raízes e polos em geral são reais. Além disso, as polos são sempre positivos para $|\alpha| < 1$ que é a faixa de interesse para as nossas funções. Quando escrevemos na forma de fração parcial ($r(x) = \sum_{k=1}^m \frac{\alpha_k}{x+\beta_k}$) eles podem ser calculados utilizando um solver. Então o custo para resolver uma função racional é essencialmente o mesmo para uma matriz de inversão, e que a primeira ordem de precisão é independente do custo. Isso sem mencionar que os sinais dos coeficientes α_k são em geral todos de mesmo sinal (para $|\alpha| < 1$), então a resolução de funções racionais utilizando frações parciais é numericamente estável.

Assim como é feito no algoritmo PHMC, nós reescrevemos o determinante em termos dos pseudo-férmions, mas agora substituímos o núcleo por uma aproximação racional

$$\det M^\alpha = \int D\Phi^\dagger D\Phi e^{-\Phi^\dagger M^{-\alpha} \Phi} \approx \int D\Phi^\dagger D\Phi e^{-\Phi^\dagger r^2(M) \Phi} , \quad (3.118)$$

com $r(x) = x^{-\alpha/2}$. Diferentemente do caso PHMC, nós temos que impor a condição de degenerescência com um kernel racional para permitir que o banho térmico possa ser solucionado. A grande vantagem é por que nós podemos incluir uma aproximação racional para uma precisão arbitrária onde não é requerido recalculer o teste de aceitação. Então podemos proceder exatamente como se fosse um algoritmo de Monte Carlo Híbrido,

conforme foi discutido na seção anterior 3.7. Sendo assim, o algoritmo RHMC consiste de

- Trajetória de Dinâmica Molecular Híbrida
 - Atualização do momento através do banho térmico [$P(\pi) \propto e^{-\pi^* \pi/2}$]
 - Atualização do Pseudo-Férmion através do banho térmico [$\Phi \propto r(M)^{-1} \xi$, onde $P(\xi) \propto e^{-\xi^* \xi}$]
 - Trajetória de dinâmica molecular com $\tau/\delta\tau$ passos
- Teste de aceitação de Metropolis $P_{aceita} = \min(1, e^{-\delta H})$.

Resolvendo a força para uma ação fermiônica pode resultar em uma inversão dupla devido à presença do quadrado na função racional. Uma forma alternativa de se obter a aproximação racional é usada no passo de dinâmica molecular, onde ($\bar{r} \approx M^{-\alpha} \approx r^2$). Dessa forma, a força pseudo-fermiônica é apenas uma soma de termos de HMC

$$S'_{pf} = - \sum_{i=1}^{\bar{m}} \bar{\alpha}_i \Phi^\dagger (M + \bar{\beta}_i)^{-1} M' (M + \bar{\beta}_i)^{-1} \Phi . \quad (3.119)$$

O custo deste algoritmo é quase idêntico ao do HMC simples, onde temos uma inversão de férmion por força calculada (também temos uma inversão extra para cada solução de banho térmico no começo de cada trajetória).

O algoritmo RHMC é hoje o mais utilizado com os férmions do tipo “staggered”, que serão escolhidos por nós para os testes com o pacote MILC.

Vale lembrar que as duas formulações mais utilizadas para a discretização dos campos fermiônicos são a formulação de Wilson e os férmions staggered. Esses últimos têm sido empregados pela colaboração MILC (dos EUA) em seus resultados, devido à maior eficiência de simulação em relação aos férmions de Wilson. Por sua vez, os férmions de Wilson são os escolhidos pela maioria das colaborações da QCD na rede da Europa. Recentemente a simulação utilizando férmions de Wilson tornou-se mais eficiente e competitiva devido à introdução do método Domain-Decomposed Hybrid Monte Carlo (DDHMC) (34). Naturalmente é importante uma produção de resultados com as duas formulações de

férmions acima para comparação, já que as duas formulações devem conduzir a resultados idênticos no limite do contínuo.

É preciso também notar que ambas as formulações acima apresentam limitações, e há um acirrado debate sobre o limite de validade das simulações com férmions staggered, devido ao processo de extração da raiz quarta do determinante fermiônico descrito no presente capítulo (35).

Vale também ressaltar que nos últimos anos começaram a ser produzidos resultados utilizando as formulações de férmions quirais, que não apresentam as limitações mencionadas acima (36), mas requerem ainda mais recursos computacionais para a sua simulação.

Computação Paralela

Podemos definir a Computação Paralela como a execução de diversas tarefas simultaneamente, onde o problema a ser calculado é segmentado em pedaços menores (chamados de tarefas). Computadores Paralelos oferecem o potencial de concentrar grandes recursos computacionais, na forma de processadores, memória e banda de entrada e saída, o que proporciona uma forma de obter resultados mais rápidos além de possibilitar um aumento nos detalhes da simulação.

4.1 Modelos Computacionais Paralelos

A definição de modelo computacional é a abstração das ações do computador para realizar as suas execuções, ou seja, na especificação de um modelo estamos apenas interessados na ideia de como fazer a execução dos programas e não na implementação do modelo. Qualquer um dos modelos discutidos abaixo pode ser implementado em qualquer computador paralelo, com um pouco de auxílio do sistema operacional. A eficiência de uma dada implementação depende, entretanto, da relação entre o modelo e a máquina.

Modelos Computacionais paralelos formam uma estrutura complicada. Eles podem ser diferenciados por vários ângulos: se a memória é fisicamente compartilhada ou distribuída,

o quanto de comunicação é realizada via software ou hardware, qual é a unidade de execução, entre outros. Apresentamos a seguir alguns desses modelos.

4.2 Paralelismo de Dados

O modelo de paralelismo de dados envolve a execução de uma computação similar em cima de muitos objetos de dados de forma simultanea. Uma das primeiras implementações deste tipo de modelo foi nos processadores vetoriais, que foram as máquinas que iniciaram a era da supercomputação. A ideia central de uma máquina vetorial é a aplicação de uma mesma instrução em uma estrutura de dados, tal como a definição de Single Instruction Multiple Data (SIMD). O paralelismo não precisa necessariamente ser executado instrução por instrução em todos os passos para ser classificado como um paralelismo de dados, sendo melhor definido como um estilo de programação do que uma arquitetura computacional.

Em qualquer nível, o modelo permanece o mesmo: O paralelismo é aplicado sobre os dados, e o programa é muito semelhante a um programa sequencial. O particionamento dos dados realizado no modelo pode ser feito pelo compilador, onde podemos citar o caso do High Performance Fortran (HPF) que são usadas diretivas fornecidas pelo programador que especificando um conjunto de otimizações, no Fortran, que ajuda um compilador no processo de computação dos dados (37).

4.3 Memória Compartilhada

O paralelismo que não é determinado implicitamente pela independência de dados, mas é explicitamente especificado pelo programador é o paralelismo de controle. Um

modelo simples do paralelismo de controle é o modelo de memória compartilhada, onde cada processo pode acessar todo o espaço endereçado de memória. Para permitir que os processadores acessem essa região de memória é implementada alguma espécie de trava. Em linguagens de alto nível pode ser escondido o uso explícito destas travas.

Outro exemplo que podemos citar que utiliza memória compartilhada é o OpenMP (38), que por meio de diretivas de compilação permite uma maneira de o programador fornecer informações para que o compilador saiba onde ele pode encontrar paralelismo, tais como em loops codificados sequencialmente, utilização em alto nível de threads, etc.

As máquinas de memória compartilhada em escala menor são chamadas de Symmetric MultiProcessings (SMPs), que nada mais é que a capacidade de dois ou mais processadores, receberem tarefas para executar por meio do sistema operacional.

O uso do paralelismo oriundo de técnicas de memória compartilhada tem ganhado bastante força, devido ao fato já comentado sobre o crescimento de transistores dentro da pastilha 1.3.

Outro exemplo que também podemos citar é as Non-Uniform Memory Access (NUMA), onde são um conjunto de processadores, cada um ligado a uma memória local (que é de acesso rápido), mas permitindo acessar as memórias que estão ligadas a outros processadores, onde todos possuem acesso a todas as memórias (39).

4.4 Threads

Formas iniciais do modelo de memória compartilhada tinham processos com espaço de endereçamento separados. A memória compartilhada era obtida através de operações explícitas, tais como formas especiais como por exemplo o malloc em C. A versão mais comum do modelo de memória compartilhada especifica que toda a memória tem que ser compartilhada. Isto permite que o modelo seja aplicado a sistemas com multithread,

onde um processo unitário (no espaço de endereços) pode ser associado com diversas pilhas de execução. Já que o modelo permite trocas rápidas de uma thread para outra e não necessita explicitamente de operações de memória, ele pode ser utilizado em programas Fortran. A dificuldade imposta pelo modelo de thread é que qualquer estado do programa é definido pelo valor das variáveis do programa compartilhadas por todas as threads simultaneamente. Entretanto em muitos sistemas de threads é permitido alocar na memória local da thread. O modelo de thread mais comumente utilizado é especificado pelo padrão Portable Operating System Interface (POSIX) (40). Uma implementação em alto nível para programar com threads é oferecida pelo OpenMP (38).

4.5 Passagem de Mensagem

O modelo de passagem de mensagem estabelece que um conjunto de processadores possuem apenas uma memória local, mas é permitido que eles se comuniquem com outros processos através do envio e recebimento de mensagens. A definição de uma das características do modelo de passagem de mensagem é que a transferência de dados da memória local de um dos processos para a memória local de outro processo requer operações sendo executadas por ambos os processos. O Message Passing Interface (MPI) é uma interface de programação para o modelo de passagem de mensagem.

4.6 Vantagens

Universalidade

O modelo de passagem de mensagem se baseia no princípio de que os processadores estão separados e são conectados por uma rede. Este princípio é satisfeito desde computadores

vetoriais até clusters de PCs, diferentemente do que ocorre caso desejamos utilizar um modelo de memória compartilhada que necessita de aquisição extra de hardware para ser implementado.

Expressividade

O uso de passagem de mensagens é considerado um método completo e útil para se expressar algoritmos paralelos. Ele fornece o controle que falta nos modelos de paralelismo de dados e compiladores baseados na divisão da localidade de dados. Isso é perfeito para os programas serem adaptativos, na auto-sincronização de algoritmos e para programas que podem ser feitos tolerantes ao desbalanceamento na velocidade dos processos encontrada em redes compartilhadas.

Depuração

Depurar programas paralelos continua sendo uma área de desafio. Enquanto depuradores para programas paralelos são fáceis de escrever para o modelo de memória compartilhada, vale ressaltar que o processo de depuração em si é fácil no paradigma de passagem de mensagem. Isto ocorre porque um dos casos de erro mais comuns é a sobrescrita de memória de forma não esperada. No caso do modelo de passagem de mensagem, já que o controle da referência da memória é mais explícito que em outros modelos (apenas um processo tem acesso a uma parte qualquer da memória), torna-se mais fácil localizar erros de leitura e escrita. Alguns depuradores paralelos podem mostrar mensagens de informações, que normalmente são invisíveis ao programador.

Desempenho

Os computadores atuais tiveram um grande ganho de desempenho devido ao controle da hierarquia de memória e otimização da memória cache. Ao utilizar passagem de mensagem, temos uma forma de fornecer explicitamente os dados aos processos, que permite que o compilador e o hardware de gerenciamento de cache funcionem plenamente, ou seja, não tenham que se preocupar com detalhes da execução paralela.

4.7 Operações de Memória Remota

Considere o modelo de memória compartilhada, onde se permite que os processos acessem qualquer ponto da memória independentemente da forma como essa chamada é realizada via hardware, e o modelo de passagem de mensagem, em que os processos local e remoto precisam participar. Temos que na mescla destas duas ideias vem a concepção do modelo de operações de memória remota. Este modelo utiliza operações do tipo put e get, onde um determinado processo pode acessar a memória de outro, sem a participação do processo que está sendo acessado, sendo que esse acesso é feito de forma explícita, não da mesma forma que é realizado quando acessa a memória local. Um tipo de operação que pode ser citado é a “mensagem ativa”, onde uma mensagem causa a execução de uma sub-rotina no espaço de endereço de outro processo. Mensagens ativas são usadas para facilitar cópias de memórias remotas. Assim operações de cópia de memória remota são executadas por apenas um lado, ao invés das execuções dos dois lados necessárias no modelo de passagem de mensagem.

4.8 Combinação de Modelos

A combinação entre os modelos apresentados acima é possível, onde em alguns clusters de processos temos o uso de memória compartilhada, mas pode haver comunicação entre clusters via passagem de mensagem, ou podemos ter um processo único que pode ser multithread mas que não compartilha memória com outro.

4.9 MPI

Conforme já foi mencionado anteriormente, o MPI é a uma especificação de interface para o modelo de passagem de mensagem, sendo que existem diversas implementações para o MPI e todas as suas operações são expressas em forma de funções, subrotinas e métodos, de acordo com linguagem de programação a ser utilizada. O seu padrão é definido através de um processo aberto entre fabricantes de computadores paralelos, cientistas e desenvolvedores de aplicações.

Os padrões de documentos ficam localizados no MPI Forum (41), e a especificação atual é a versão 2.2 lançada em setembro de 2009.

O pacote MILC para ser executados em ambientes paralelos utiliza qualquer uma das implementações disponíveis de MPI, veremos maiores detalhes no capítulo a seguir.

O pacote MILC

O pacote é produzido pela colaboração MILC, que tem como alvo a construção de um conjunto de aplicações de pesquisa de alto desempenho escritas em C para simulações da QCD, mais especificamente de teorias de gauge na rede com grupo de simetria $SU(3)$, para uma variedade de formulações fermiônicas de rede incluindo vários algoritmos de simulação distribuídos em diferentes implementações de máquinas paralelas Multiple Instruction Multiple Data (MIMD). Quando implementada em modo escalar, a ferramenta roda em uma variedade de workstations, sendo extremamente versátil para produção e exploração de suas aplicações. Um dos objetivos da colaboração MILC é escrever códigos portáteis que funcionam em diferentes arquiteturas de máquinas. Ele é escrito sobre os termos da General Public License GNU redigido pela Fundação do Software Livre Free Software Foundation (FSF).

5.1 Descrição Geral

O pacote MILC é distribuído na forma de um arquivo comprimido, disponibilizado nos sites de alguns dos membros da colaboração (42). Os procedimentos para a instalação são comentados no apêndice A.

Após a instalação do MILC, no arquivo raiz de instalação estão dispostas seis categorias diferentes de diretórios: “applications”, “generic”, “include”, “libraries”, “doc” e “file utilities”.

Nos diretórios de aplicações (“applications”) ficam as simulações dos fenômenos físicos, ou seja, o objeto principal de estudo que se deseja reproduzir. Todas as aplicações dividem entre si o diretório das bibliotecas (“libraries”), que possui rotinas de baixo nível utilizadas nas simulações. O diretório de cabeçalhos (“include”) possui os arquivos header (*.h). O diretório genérico (“generic”) possui rotinas de alto nível que são mais ou menos independentes da física. O diretório de utilitários de arquivos (“file utilities”) é usado internamente pelos programas, para rotinas de teste dos arquivos. O diretório de amostras binárias (“binary_samples”) é um agrupado de dados salvos em formato binário, utilizado para entrada de dados em exemplos e para teste. Por final, há os diretórios sse, que implementam rotinas SSE nos códigos MILC.

Figura 5.1 – Estrutura da instalação do pacote MILC

5.2 Forma de Utilização

Depois de instalado e configurado o MILC para o ambiente de trabalho, existem duas formas de serem executadas as aplicações MILC.

A primeira forma é o usuário executar o aplicativo desejado sem informar previamente nenhum parâmetro, dessa forma

```
$ cd /usr/local/share/milc/ks_imp_dyn  
$ ./su3_hmd
```

Após a inicialização do aplicativo, são solicitados os valores iniciais para começar a simulação, um à um. No momento em que é enviado o último parâmetro de entrada, é iniciada a simulação, fornecendo os resultados no terminal. Uma outra forma de inicializar a simulação é através de um arquivo de entrada que contenha as informações que se deseja executar.

Lembremos que cada um dos aplicativos possui um arquivo de entrada próprio para executar a sua simulação.

Temos como proposta a execução de dois tipos de aplicativos, o *su3_hmd* e o *su3_rhmc*. Portanto, a seguir serão apresentados os parâmetros que são comuns aos seus respectivos arquivos de entradas, e no capítulo a seguir descreveremos apenas as variáveis que não foram contempladas nesta seção.

O primeiro parâmetro solicitado é

```
prompt [1 | 0]
```

onde

- 1 = Cria legendas indicando qual é o nome do parâmetro esperado pelo aplicativo no momento da execução. É utilizado basicamente no caso de o usuário estar inserindo os valores diretamente no aplicativo.

- 0 = Não apresenta as legendas. O uso deste valor na variável esta associado à execução do aplicativo utilizando um arquivo de entrada.

As dimensões da rede são definidas através das seguintes variáveis

```
nx [int]
ny [int]
nz [int]
nt [int]
```

Pode-se definir a semente geradora dos números aleatórios através do parâmetro

```
iseed [int]
```

A variável responsável pelo número de trajetórias a serem executadas é fornecida por

```
trajecs [int]
```

O número de trajetórias entre as medidas informa quantas trajetórias devem ser realizadas antes que se inicie uma medida da rede, e é informado através de

```
traj_between_meas [int]
```

O parâmetro de acoplamento de gauge é informado por meio de

```
beta [double]
```

O fator de tadpole (para melhoria da ação utilizada) é fornecido por

```
u0 [double]
```

O tamanho do passo temporal de integração no processo de dinâmica molecular é informado pelo parâmetro

```
microcanonical_time_step [double]
```

e o número de passos por trajetória é informado por

```
steps_per_trajectory [int]
```

A definição do máximo de iterações para o cálculo do gradiente conjugado é fornecido por

```
max_cg_iterations 300
```

O número máximo de vezes que podem ser reiniciadas as medidas do gradiente conjugado na simulação é definido por

```
max_cg_restarts 5
```

assim como a tolerância no cálculo do gradiente conjugado

```
error_for_propagator .000002
```

A lista de possíveis valores para a variável de entrada da rede a ser utilizada é dada por

```
[ continue | fresh | reload_ascii | reload_serial ]
```

sendo as opções definidas por

- `continue` = continua a execução da simulação com a rede atual
- `fresh` = inicializa uma nova rede para os cálculos
- `reload_ascii` = inicializa a simulação com uma rede salva em formato ascii
- `reload_serial` = inicializa a simulação com uma rede salva em formato binário

O formato de saída do estado atual da rede é dado através de

```
[ forget | save_ascii | save_serial ]
```

- `forget` = Não salva o estado atual da rede.
- `save_ascii` = Salva o estado atual da rede no formato ascii.
- `save_serial` = Salva o estado atual da rede no formato binário.

Vale ressaltar que não necessariamente os valores são requeridos nesta ordem e que será preciso informar alguns outros parâmetros de entrada, como apresentado no capítulo seguinte (Capítulo 6), para cada um dos aplicativos a serem utilizados.

Metodologia e Resultados

Neste capítulo são apresentados os parâmetros de entrada dos aplicativos **su3_hmd** e **su3_rhmc**, explicando os parâmetros ainda não descritos no Capítulo 5. Em seguida, será elaborada uma comparação entre os algoritmos R e RHMC.

6.1 Geração dos Arquivos de Entrada

Para realizar as medidas comparativas dos aplicativos, foi inicializada uma rede totalmente aleatória com o aplicativo R (*su3_hmd*) e a configuração da rede foi salva, para ser utilizada como rede de entrada para ambos aplicativos.

Serão primeiramente descritos os arquivos de entrada para o aplicativo R, realizando a composição com os parâmetros já descritos em conjunto com as variáveis específicas para esta execução. Na sequência, será executado o mesmo procedimento para a geração do aplicativo RHMC.

6.2 Aplicativo su3_hmd

Podemos definir um arquivo genérico para a execução do aplicativo **su3_hmd** conforme descrito a seguir

```
prompt 0
nflavors1 4
nflavors2 4
nx 4
ny 4
nz 4
nt 4
iseed 5682304

warms 0
trajecs 2
traj_between_meas 1
beta 7.00
mass1 0.1
mass2 0.1
u0 0.8441
microcanonical_time_step 0.02
steps_per_trajectory 4
max_cg_iterations 300
max_cg_restarts 5
error_per_site .000005
error_for_propagator .000002
reload_serial ../binary_samples/lat.sample.14444
forget
```

Os primeiros parâmetros que não são descritos anteriormente são

```
nflavors1
nflavors2
```

utilizados para definir o número de sabores dos quarks.

As massas dos quarks são definidas pelos parâmetros

```
mass1
mass2
```

e a precisão por sítio é dada por

```
error_per_site .000005
```

6.3 Aplicativo su3_rhmc

Uma das características na execução dos aplicativos que utilizam o método RHMC é a necessidade de utilizar um arquivo extra de entrada, que especifica a função racional utilizada nos cálculos. Este arquivo é gerado baseado nos valores de entrada e o pacote MILC possui um aplicativo de apoio para gerar este arquivo de input. Os detalhes de como utilizá-lo estão descritos na seção a seguir.

6.4 Aplicativo Remez

O local contendo os arquivos para se gerar este aplicativo fica dentro da pasta

```
$ /usr/local/share/milc/ks_imp_rhmc/remez-milc/
```

Mas, antes de executar a instalação deste aplicativo é necessário instalar o pacote **GNU Multiple Precision Arithmetic Library (GMP)** com suporte a **Multiple Precision Floating point with correct Rounding (MPFR)**. Os passos para instalação podem ser vistos no apêndice B.

Após instalar as dependências, podemos compilar o aplicativo executando o seguinte comando

```
$ make poly4
```

que gera o programa poly4, cujos dados de entrada são descritos a seguir

```
$ ./poly4 n_pseudo naik_term n1 m1 n2 m2 n3 m3 n4 m4 ordem_dm  
ordem_acao min_eig max_eig digitos
```

onde

- *n_pseudo* = Define o número de campos de pseudoférmions.

- *naik_term* = Define o epsilon para o termo *naik*, mas não tem uso nenhum atualmente no código, servindo apenas para acertar a entrada de parâmetros.
- *n1* = O número de sabores para o primeiro tipo de quark.
- *m1* = Massa do primeiro tipo de quark.

Temos a repetição destes parâmetros mais 3 vezes (de dois até quatro, representando outros tipos de quarks), e em seguida temos

- *ordem_dm* = Ordem de aproximação para a integração da dinâmica molecular.
- *ordem_acao* = Ordem de aproximação para a resolução da ação.
- *min_eig* = Mínimo autovalor de $\mathcal{D}^2$. É necessário escolher um valor bem próximo de nulo, por exemplo $1e^{-15}$
- *max_eig* = Máximo autovalor de $\mathcal{D}^2$.
- *digitos* = Define a precisão dos dígitos utilizados internamente.

A saída do arquivo pode ser redirecionada para um arquivo texto, o qual deverá ser utilizado como valor de entrada para a execução do aplicativo *su3_rhmc*, o parâmetro responsável por receber este valor é *load_rhmc_params*.

Após realizar a montagem do arquivo de entrada para a função racional, vamos continuar a apresentação do arquivo de entrada genérico do aplicativo **su3_rhmc**.

```
prompt 0
nx 6
ny 6
nz 6
nt 6
iseed 3546789
n_pseudo 2
load_rhmc_params rationals.sample.su3_leapfrog
beta 6.50
n_dyn_masses 2
dyn_mass 0.01 0.05
dyn_flavors 2 1
u0 0.862
```

```

warms 0
trajecs 2
traj_between_meas 1
microcanonical_time_step 0.01
steps_per_trajectory 12
cgresid_md_fa_gr .000002 .000002 .000002
max_multicg_md_fa_gr 1750 1750 1750
cgprec_md_fa_gr 1 2 2
cgresid_md_fa_gr .000002 .000002 .000002
max_multicg_md_fa_gr 1750 1750 1750
cgprec_md_fa_gr 1 2 2
prec_ff 2
error_for_propagator .000001
max_cg_prop 500
max_cg_prop_restarts 5
reload_serial ../binary_samples/lat.sample.l6666b650m010m050
forget

```

O número de pseudo-férmions é dado pela variável

```
n_pseudo
```

lembrando que este valor precisa estar de acordo com o valor fornecido na geração da função racional e que, conforme foi dito, o parâmetro que fornece o arquivo de entrada contendo as informações da função racional gerada é fornecido por

```
load_rhmc_params
```

O número de massas dinâmicas a serem usadas no cálculo é dado por

```
n_dyn_mass
```

e os respectivos valores para cada massa são fornecidos pela variável

```
dyn_mass
```

A degenerescência dos sabores é dada por

```
dyn_flavors
```

Os próximos três parâmetros são fornecidos para cada pseudoférmion, e são calculados para o uso do gradiente conjugado na evolução da dinâmica molecular (md), no cálculo

da ação fermiônica (fa) e na seleção da fonte gaussiana aleatória para o pseudo-férmion (gr). O erro residual para a solução do gradiente conjugado é fornecido em

```
cgresid_md_fa_gr
```

Já o número máximo de iterações para o gradiente conjugado para cada instante é dado por

```
max_multicg_md_fa_gr
```

e a precisão da solução do gradiente conjugado para cada instante é

```
cgprec_md_fa_gr
```

A precisão do cálculo da força fermiônica é representada por

```
prec_ff
```

Pode-se definir o número máximo de execuções do gradiente para o cálculo do propagador

```
max_cg_prop
```

e a informação sobre o número máximo de vezes para reiniciar o cálculo do gradiente conjugado é dada por

```
max_cg_prop_restarts
```

De posse dessas informações podemos realizar a geração dos arquivos de entrada para a execução dos casos de comparação.

6.5 Execução dos Casos

Para realizar esta comparação, a base dos arquivos de entrada utilizados para os aplicativos R e RHMC, respectivamente são apresentados a seguir.

```

prompt 0
nflavors1 2
nflavors2 0
nx 4
ny 4
nz 4
nt 4
iseed 5682304
warms 0
trajecs 20000
traj_between_meas 100
beta 6.50
mass1 0.05
mass2 0.99
u0 0.862
microcanonical_time_step 0.01
steps_per_trajectory 4
max_cg_iterations 1750
max_cg_restarts 5
error_per_site .000005
error_for_propagator .000001
reload_serial ../redes/lat.l4
save_serial ../redes/r/lat.l4

```

```

prompt 0
nx 4
ny 4
nz 4
nt 4
iseed 3546789
n_pseudo 2
load_rhmc_params input_rhmc_poly4.in
beta 6.50
n_dyn_masses 2
dyn_mass 0.05 0.99
dyn_flavors 2 0
u0 0.862
warms 0
trajecs 20000
traj_between_meas 100
microcanonical_time_step 0.01
steps_per_trajectory 4
cgresid_md_fa_gr .000002 .000002 .000002
max_multicg_md_fa_gr 1750 1750 1750
cgprec_md_fa_gr 1 1 1
prec_ff 1
error_for_propagator .000001
max_cg_prop 500
max_cg_prop_restarts 5
reload_serial ../redes/lat.l4

```

```
save_serial ../redes/rhmc/lat.14
```

6.6 Resultados e Análises

Na realização desta análise, utilizamos uma rede com tamanho $V = 4^4$. Realizamos cada caso com 20000 trajetórias, e executando uma medida a cada 100 trajetórias, o que nos fornece cerca de 200 medidas.

Primeiramente foi verificado o ganho de tempo na execução paralela das simulações. A Tabela 6.1 e o Gráfico 1 mostram os resultados obtidos.

Tabela 6.1 – Medida do tempo de execução dos algoritmos R e RHMC, variando a quantidade de nós para processamento.

Alg.	steps_per_ trajectory	microcanonical_ time_step	Tempo de 1 Processo	Tempo de 4 Processos	Tempo de 8 Processos
R	4	0.010	6221.36	1912.63	1640.40
R	8	0.005	12438.26	3885.12	3355.53
R	40	0.001	119547.00	18504.65	15385.77
RHMC	4	0.010	10657.73	3112.97	2484.72

Podemos comprovar nestas medidas que o comportamento da curva é bastante semelhante em todos os casos, e temos um ganho grande no tempo de cálculo quando saímos da execução sequencial para a execução paralela utilizando quatro processos, sendo que execução roda ≈ 3.2 vezes mais rápido. A melhora obtida entre quatro processos e oito processos é de ≈ 1.2 , que pode passar a sensação de não ser um ganho tão grande quanto o anterior, mas mesmo assim ele é bastante significativo, pois se tivermos uma simulação que executa em 30 dias com quatro processos, ao executar a mesma simulação com oito processos ela demora cerca de ≈ 25 dias, um ganho de cinco dias.

Outro ponto que é interessante ressaltar é que o ganho temporal deve melhorar ao aumentar a rede, pois teríamos mais elementos a serem distribuídos pelos processos.

Agora será feita a análise de comparação entre o algoritmo R e o algoritmo RHMC.

Gráfico 6.1 – Análise dos tempos de execução dos algoritmos R e RMHC.

Gráfico 6.2 – Análise refinada do Gráfico 6.1.

Esta comparação será feita através do uso da medida *GACTION*, que informa o estado atual de energia da rede. Para verificar quanto o resultado está próximo do esperado,

foram calculados a média e o desvio padrão de cada uma das medidas. Podemos observar os resultados na Tabela 6.2.

Tabela 6.2 – Análise dos valores médios para diferentes discretizações.

Alg.	steps_per_ trajectory	microcanonical_ time_step	Média	Desvio Padrão
R	4	0.010	6.40	0.13
R	8	0.005	6.45	0.18
R	40	0.001	6.34	0.16
RHMC	4	0.010	6.38	0.08

Ao utilizar a média e o desvio padrão como instrumento de medida para verificar a execução dos aplicativos, foi suposto que cada uma das medidas obtidas fosse independente da anterior. Consideramos que ao realizar o cálculo das medidas a cada 100 trajetórias, esta independência da medida anterior seja alcançada. Desta forma, o desvio padrão ilustra o erro da medida efetuada.

Ao analisar os números obtidos, todos são equivalentes e possuem um erro na ordem de $\approx 3\%$ dos valores das medidas, o que é esperado se tratando de uma medida obtida por Método de Monte Carlo.

Outra análise que podemos realizar é a comparação dos erros (dados pelo desvio padrão) entre o algoritmo RHMC e o algoritmo R. Podemos ver que, embora o algoritmo tenha um tempo de execução maior análise dos valores de média e desvio padrão, ele possui uma precisão melhor. Desta forma, temos um ganho na execução, pois conseguimos obter um resultado mais preciso, utilizando uma discretização mais grosseira.

Podemos explicar a diferença de desempenho observado acima pelo fato que o algoritmo RHMC é exato, diferentemente do algoritmo R (devido ao ruído adicionando). Fica demonstrado, portanto, que é preciso cuidado na escolha de parâmetros na utilização do algoritmo R, o que nem sempre foi observado nas décadas anteriores. De fato, conforme já foi mencionado, ao utilizar os férmios staggered com o algoritmo R, a regra $\delta\tau^R \approx 2m/3$ era amplamente utilizada e raramente era executada a extrapolação para o passo zero, pois o erro por passo é desconhecido. O algoritmo RHMC não possui este problema.

Resumindo, o algoritmo RHMC é melhor conceitualmente por ser exato e é mais

eficiente, por conseguir obter resultados com uma discretização temporal menos refinada.

Considerações Finais

Em todas as áreas de pesquisa, o uso da computação científica é crescente, pois nos últimos 40 anos o acesso a recursos computacionais foi bastante facilitado, devido a sua produção em larga escala. Graças a este tipo de avanço, áreas como a simulação numérica da cromodinâmica quântica (QCD), que até pouco tempo atrás estavam limitadas apenas a super-computadores, podem ser estudadas em máquinas de pequeno porte, que formam os clusters de PCs, oferecendo uma ótima relação custo/benefício.

Dessa forma, a criação de algoritmos mais eficazes e definidos em uma aplicação paralela, capaz de utilizar o arranjo de PCs conectados, é crucial para o entendimento da QCD.

7.1 Contribuições

Este trabalho foi responsável por apresentar diversas visões sobre o papel das simulações numéricas na ciência nos dias atuais, não sendo defendida uma só visão como correta, mas enfatizando que isso é uma decisão pessoal. Qualquer que seja a opinião sobre a simulação numérica, percebemos que ela cada vez ocupa um papel de maior destaque na ciência, seja na realização da pesquisa, seja na área de divulgação dos conhecimentos, etc.

Após a apresentação deste ponto, foi percorrido um caminho teórico e histórico para a introdução dos métodos numéricos utilizados para o estudo da QCD com uso dos férmions dinâmicos. Analisamos a princípio a formulação da ação na Mecânica Clássica, e fizemos a extrapolação para a ação na Mecânica Quântica. A partir deste ponto, fizemos uma descrição dos principais aspectos envolvidos no estudo da QCD, e discutimos por que motivo precisamos realizar uma abordagem que evite o uso da teoria de perturbação.

A seguir verificamos as fundamentações dos métodos numéricos a serem utilizados, tais como a compreensão das cadeias de Markov, e a definição de ergodicidade para o nosso estudo. Fizemos então um estudo evolutivo das técnicas utilizadas no estudo da QCD na rede, começando pelo uso do método de Monte Carlo na sua forma mais simples, e apresentando as evoluções necessárias no método para se obter ganhos tanto em performance como em resultado. O método final nesta escala evolutiva foi o Método RHMC, que recebeu diversas influências dos métodos anteriores.

Após esta etapa, realizamos uma breve explicação sobre o uso de programação paralela, com uma ênfase principalmente no modelo de passagem de mensagem, pois é uma das metodologias de paralelização mais utilizadas, principalmente pelo advento dos clusters de PCs citado anteriormente.

Enfim, o objetivo principal foi apresentar o Pacote MILC, que utiliza toda a teoria descrita nos capítulos anteriores, e permitir a instalação e a manutenção do pacote em um ambiente de produção de pesquisa. Como forma de utilização do Pacote, estudamos a comparação do algoritmo R e do algoritmo RHMC, comprovando que temos uma implementação melhor tanto no aspecto de performance como no aspecto de precisão do segundo método sobre o primeiro.

Acreditamos que o presente trabalho será bastante útil à comunidade de simulações de QCD na rede no Brasil, como guia para a introdução às simulações com férmions dinâmicos, tanto do ponto de vista da utilização do pacote MILC, quanto do entendimento dos algoritmos nele envolvidos. De fato, nas simulações realizadas até o momento tem sido considerado o caso da QCD pura (ou aproximação *quenched*), isto é, sem consi-

derar os efeitos de fermions dinamicos. Embora estudos da QCD quenched permitam a compreenso de aspectos qualitativos da QCD, como o mecanismo de confinamento de quarks (43, 44), ha casos em que crucial poder realizar simulaes da QCD completa. Entre estes, esta o estudo da transio de fase de desconfinamento de quarks na QCD a temperatura finita (45).

Como continuao do estudo aqui apresentado, poderiam ser utilizadas ferramentas de anlise de eficincia de paralelizao, tais como a ferramenta HyeraAnalysis, desenvolvida por pesquisadores do IFSC-USP (46), o que nos permitiria verificar gargalos na implementao paralela do software MILC para as diversas arquiteturas utilizadas.

REFERÊNCIAS

- 1 ZANETIC, J. *Textos de evolução dos conceitos da física: notas de aula*. São Paulo: IF, 2006.
- 2 WOLFRAM, S. *A new kind of science*. Champaign: Wolfram Media, 2002.
- 3 LANDAU, D. P.; BINDER, K. *A guide to monte carlo simulations in statistical physics*. Cambridge: Cambridge University Press, 2000.
- 4 QUINN, M. J. *Parallel programming in c with mpi and openmp*. Dubuque: McGraw-Hill, 2004.
- 5 GOLDSTEIN, H.; POOLE, C.; SAFKO, J. *Classical mechanics*. 3rd ed. San Francisco: Addison-Wesley, 2001.
- 6 LANCZOS, C. *The variational principles of mechanics*. 4th ed. Mineola: Courier Dover Publications, 1986.
- 7 COHEN-TANNOUDJI, C.; DIU, B.; LALOË, F. *Quantum mechanics*. 2nd ed. Toronto: Wiley-Interscience Publication, 1977. v. 1.
- 8 FEYNMAN, R. P. Space-time approach to non-relativistic quantum mechanics. *Reviews of Modern Physics*, Ridge, v. 20, n. 2, p. 367–387, 1948.
- 9 ROTHE, H. J. *Lattice gauge theories: an introduction*. 3rd ed. Hackensack: World Scientific Publishing, 2005.
- 10 WILSON, K. G. Confinement of quarks. *Physical Review D*, College Park, v. 10, p. 2445–2459, 1974.
- 11 The Green 500. The green500 list - environmentally responsible supercomputing. Disponível em: <<http://www.green500.org/>>. Acesso em: 15 Dez. 2009.
- 12 CLARK, M. A.; BABICH, R.; BARROS, K.; BROWER, R. C.; REBBI, C. Solving lattice qcd systems of equations using mixed precision solvers on gpus. *Computer Physics Communications*, Amsterdam, 2009. in press.
- 13 EDWARDS, R. G.; JOÓ, B. The chroma software system for lattice qcd. In: INTERNATIONAL SYMPOSIUM ON LATTICE FIELD THEORY, 22., 2004, Batavia, Illinois. *Proceedings...* Amsterdam, The Netherlands: Elsevier, 2005. p. 832–834.

14 DI PIERRO, M.; FLYNN, J. M. Lattice qft with fermiqcd. In: INTERNATIONAL SYMPOSIUM ON LATTICE FIELD THEORY, 23., 2005, Dublin. *Proceedings...* Trieste: IOP Publishing, 2005. v. 104. p. 1–6.

15 COLUMBIA UNIVERSITY. CPS user manual. Disponível em: <<http://qcdoc.phys.columbia.edu/doxygen/usr/html/>>. Acesso em: 12 Out. 2008.

16 NEWMAN, M. E.; BARKEMA, G. T. *Monte carlo methods in statistical physics*. Oxford: Oxford University Press, 1999.

17 SCHERER, C. *Métodos computacionais da física*. São Paulo: Editora Livraria da Física, 2005.

18 METROPOLIS, N.; ROSEBLUTH, A. W.; ROSEBLUTH, M. N.; TELLER, A.; TELLER, E. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, Melville, v. 21, n. 6, p. 1087–1092, 1953.

19 PARISI, G.; WU, Y. Perturbation theory without gauge fixing. *Scientia Sinica*, Beijing, v. 24, p. 483–496, 1981.

20 UKAWA, A.; FUKUGITA, M. Langevin simulation including dynamical quark loops. *Physical Review Letters*, College Park, v. 55, n. 18, p. 1854–1857, 1985.

21 BATROUNI, G. G.; KATZ, G. R.; KRONFELD, A. S.; LEPAGE, G. P.; SVETITSKY, B.; WILSON, K. G. Langevin simulations of lattice field theories. *Physical Review D*, College Park, v. 32, n. 10, p. 2736–2747, 1985.

22 SALINAS, S. *Introdução à física estatística*. São Paulo: Editora Universidade de São Paulo, 2005.

23 DUANE, S. Stochastic quantization versus the microcanonical ensemble: getting the best of both worlds. *Nuclear Physics B*, Amsterdam, v. 257, n. 1, p. 652–662, 1985.

24 SCALETTAR, R. T.; SCALAPINO, D. J.; SUGAR, R. L. New algorithm for the numerical simulation of fermions. *Physical Review B*, College Park, v. 34, n. 11, p. 7911–7917, 1986.

25 GOTTLIEB, S.; LIU, W.; TOUSSAINT, D.; RENKEN, R. L.; SUGAR, R. L. Hybrid-molecular-dynamics algorithms for the numerical simulation of quantum chromodynamics. *Physical Review D*, College Park, v. 35, n. 8, p. 2531–2542, 1987.

26 DUANE, S.; KOGUT, J. B. The theory of hybrid stochastic algorithms. *Nuclear Physics B*, Amsterdam, v. 275, n. 3, p. 398–420, 1986.

- 27 FUCITO, M.; MARINARI, E.; PARISI, G.; REBBI, C. A proposal for monte carlo simulations of fermionic systems. *Nuclear Physics B*, Amsterdam, v. 180, n. 3, p. 369–377, 1981.
- 28 PETCHER, D. N.; WEINGARTEN, D. H. Monte carlo integration for lattice gauge theories with fermions. *Physics Letters B*, Amsterdam, v. 99, n. 4, p. 333–338, 1981.
- 29 BHANOT, G.; HELLER, U.; STAMATESCU, I. A new method for fermion monte carlo. *Physical Letters B*, Amsterdam, v. 129, n. 6, p. 440–444, 1983.
- 30 CLARK, M.; JOÓ, B.; KENNEDY, A. Comparing the r algorithm and rhmc for staggered fermions. *Nuclear Physics B: proceedings supplements*, Amsterdam, v. 119, n. 1, p. 1015–1017, 2003.
- 31 CLARK, M. The rational hybrid monte carlo algorithm. In: INTERNATIONAL SYMPOSIUM ON LATTICE FIELD THEORY, 24., 2006, Tucson. *Proceedings...* Trieste: IOP Publishing, 2006. p. 1–15.
- 32 FREZZOTTI, R.; JANSEN, K. A polynomial hybrid monte carlo algorithm. *Physics Letters B*, Amsterdam, v. 402, n. 3-4, p. 328–334, 1997.
- 33 KENNEDY, A. D. Algorithms for dynamical fermions. Disponível em: <<http://arXiv.org/abs/hep-lat/0607038v1>>. Acesso em: 14 Out. 2009.
- 34 DEL DEBBIO, L.; GIUSTI, L.; LUSCHER, M.; PETRONZIO, R.; TANTALO, N. Qcd with light wilson quarks on fine lattices (i): first experiences and physics results. *Journal of High Energy Physics*, Trieste, v. 2, p. 056, 2007.
- 35 SHARPE, S. R. Rooted staggered fermions: Good, bad or ugly? In: INTERNATIONAL SYMPOSIUM ON LATTICE FIELD THEORY, 24., 2006, Tucson. *Proceedings...* Trieste: IOP Publishing, 2006. p. 1–42.
- 36 LUSCHER, M. Exact chiral symmetry on the lattice and the ginsparg-wilson relation. *Physics Letters B*, Amsterdam, v. 428, n. 3-4, p. 342–345, 1998.
- 37 Rice University. High performance fortran forum. Disponível em: <<http://hpff.rice.edu/>>. Acesso em: 12 Dez. 2009.
- 38 OpenMP Architecture Review Board. Openmp.org. Disponível em: <<http://openmp.org/wp/>>. Acesso em: 14 Dez. 2009.
- 39 Linux Scalability Effort. Numa group homepage. Disponível em: <<http://lse.sourceforge.net/numa/>>. Acesso em: 12 Dez. 2009.

- 40 The Open Group. Posix faq's. Disponível em: http://www.opengroup.org/austin/papers/posix_faq.html/>. Acesso em: 15 Dez. 2009.
- 41 Message Passing Interface Forum. Message passing interface (mpi) forum home page. Disponível em: <http://www.mpi-forum.org/>>. Acesso em: 12 Set. 2009.
- 42 MILC. The mimd lattice computation collaboration. Disponível em: <http://www.physics.utah.edu/~detar/milc/>>. Acesso em: 16 Nov. 2008.
- 43 CUCCHIERI, A.; MENDES, T. Constraints on the IR behavior of the gluon propagator in Yang-Mills theories. *Physical Review Letters*, College Park, v. 100, p. 241601, 2008.
- 44 CUCCHIERI, A.; MENDES, T. Constraints on the IR behavior of the ghost propagator in Yang-Mills theories. *Physical Review D*, College Park, v. 78, p. 094503, 2008.
- 45 MENDES, T. Universality and Scaling at the chiral transition in two- flavor QCD at finite temperature. *PoS*, v. LAT2007, p. 208, 2007.
- 46 PIOLA, T. *Tracing de aplicações paralelas com informações de alto nível de abstração*. 2007. 123 p. Tese (Doutorado em Física) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2007.

Instalando o MILC

A.1 Procedimentos para a instalação

A instalação foi realizada em uma distribuição Ubuntu, em um ambiente Linux no kernel 2-6-24-23. O diretório base para a instalação foi `/usr/local/share`. Após obter a versão atual do package MILC , foi realizada a descompressão do arquivo no diretório recém-criado `milc`.

```
$ cd /usr/local/share
$ sudo mkdir milc
$ sudo chown <seu_usuario>:<seu_grupo> milc
$ cd milc
$ wget http://www.physics.utah.edu/~detar/milc/milc_qcd-7.6.2.tar.gz
$ tar -xvzf milc_qcd-7.6.2.tar.gz
```

Após estes passos, é feita a instalação dos pacotes que se deseja estudar através do makefile `Make_unpack`. No caso do estudo desta dissertação, estas aplicações são `ks_imp_dyn` e `ks_imp_rhmc`.

```
$ make -f Make_unpack ks_imp_dyn
$ make -f Make_unpack ks_imp_rhmc
```

Lembrando que pode ser realizada a instalação completa do pacote, utiliza-se a seguinte sintaxe

```
$ make -f Make_unpack all
```

Depois de instalar as aplicações desejadas, basta realizar a configuração do makefile com informações sobre a máquina onde vai ser executado o pacote MILC. No diretório raiz de instalação (`/usr/local/share/milc`), edite o arquivo `Makefile` com o seu editor preferido.

```
$ emacs Makefile
```

O arquivo possui diversas informações para a sua configuração, sendo as mais importantes para nosso estudo descritas a seguir:

```
#1. Define a shell que esta sendo utilizada
SHELL = /bin/bash

#2. Define se vai rodar em um computador paralelo
MPP = true

#3. Define a precisao dos calculos
PRECISION = 1

#4. Define o local onde esta o compilador mpi
CC = /usr/bin/mpicc

#5. Define o nivel de otimizacao de compilacao
OPT = -O2

#6. Define as outras opcoes de compilacao
OCFLAGS = -Wall

#7. Define o suporte a uso de arquivos grandes
CLFS = -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOURCE

#8. Define o local especifico das bibliotecas e includes do MPI
(nao e necessario selecionar caso esteja usando mpicc ou gcc)
IMPI = -I/usr/local/include
LMPI = -L/usr/local/lib/shared -L/usr/local/lib -lmpich

#9. Define as rotinas de entrada e saida
MACHINE_DEP_IO = io_ansi.o

#13. Define o linkador
LD = ${CC}

#14. Defines flags extras para o linkador
```

```
#LDFLAGS =

#16. Define bibliotecas extras
LIBADD =

#17. Define opcao inline
INLINEOPT = -DC_GLOBAL_INLINE #-DSSE_GLOBAL_INLINE

#18. Define macros para controle de performance e de metrica
CTIME = -DCGTIME -DFFTIME -DLLTIME -DGFTIME -DREMAP

#19. Escolha do arquivo make para fazer as bibliotecas dentro da pasta
#de libraries

MAKELIBRARIES = Make_vanilla
```

Após editar o arquivo Makefile, deve-se alterar a configuração do arquivo `include/config.h`, e do arquivo `libraries/Make_vanilla`. Passada esta fase de configuração do pacote, faça uma cópia do Makefile configurado para dentro do diretório em que você deseja executar a aplicação.

```
$ cd /usr/local/share/milc
$ cp Makefile ks_imp_rhmc/
```

Agora para finalizar o processo, basta você compilar a aplicação desejada e pode ser feita uma verificação do programa criado.

```
$ make su3_rmd
$ make check --PROJS=su3_rmd --PRECLIST=1
```

Para executar um caso, basta inserir os parâmetros de entrada de dados e saída dos resultados da seguinte forma.

```
$ su3_rmd < arquivo_de_entrada > arquivo_de_saida
```


Instalando as Aplicações Necessárias

B.1 MPI

A instalação foi realizada em uma distribuição Ubuntu, ambiente Linux sobre o kernel 2-6-24-23. O diretório base para a instalação foi `/usr/local` e a distribuição do MPI utilizada foi a MPICH2, na versão 1.0.8. Iniciamos o procedimento baixando o pacote e realizamos a sua descompressão.

```
$ wget http://www.mcs.anl.gov/research/projects/mpich2\  
> /downloads/tarballs/1.0.8/mpich2-1.0.8.tar.gz  
$ tar -xvzf mpich2-1.0.8.tar.gz
```

entramos no diretório criado e executamos os seguintes passos

```
$ cd mpich2-1.0.8.tar.gz  
$ ./configure --enable-cxx --prefix=/usr/local/share --bindir=/usr/bin  
$ sudo make  
$ sudo make install
```

B.2 GMP

A instalação foi realizada em uma distribuição Ubuntu, ambiente Linux sobre o kernel 2-6-24-23. O diretório base para a instalação foi `/usr/local` e a versão instalada do GMP foi a 4.2.4. Iniciamos o procedimento baixando o pacote e realizamos a sua descompressão.

```
$ wget ftp://ftp.gmplib.org/pub/gmp-4.2.4.tar.gz
$ tar -xvzf gmp-4.2.4.tar.gz
```

entramos no diretório criado e executamos os seguintes passos

```
$ cd gmp-4.2.4
$ ./configure --enable-cxx --prefix=/usr/local
$ sudo make
$ sudo make install
$ sudo check
```

B.3 MPFR

A instalação foi realizada em uma distribuição Ubuntu, ambiente Linux sobre o kernel 2-6-24-23. O diretório base para a instalação foi `/usr/local` e a versão instalada do MPFR foi a 2.4.1. Iniciamos o procedimento baixando o pacote e realizamos a sua descompressão.

```
$ wget http://www.mpfr.org/mpfr-current/mpfr-2.4.1.tar.gz
$ tar -xvzf mpfr-2.4.1.tar.gz
```

entramos no diretório criado e executamos os seguintes passos

```
$ cd mpfr-2.4.1
$ ./configure --prefix=/usr/local --with-gmp-build=/usr/local
```

```
$ sudo make
$ sudo make install
$ make check
```

Pode ocorrer de o aplicativo que dependa do MPFR não localizar a biblioteca, sendo necessário adicionar o caminho da mesma na variável de ambiente correspondente.

```
$ LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
$ make check
```