UNIVERSIDADE DE CAXIAS DO SUL

ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

LEONARDO MACHADO INVERNIZZI

DESENVOLVIMENTO DE UMA VERSÃO PARALELA DO SOFTWARE MCMAILLE

CAXIAS DO SUL

2018

LEONARDO MACHADO INVERNIZZI

DESENVOLVIMENTO DE UMA VERSÃO PARALELA DO SOFTWARE MCMAILLE

Trabalho de Conclusão de Curso para obtenção do grau de Bacharel em Ciência da Computação da Universidade de Caxias do Sul.

Orientador Prof. Dr. André Luis Martinotto

CAXIAS DO SUL
2018

LEONARDO MACHADO INVERNIZZI

DESENVOLVIMENTO DE UMA VERSÃO PARALELA DO SOFTWARE MCMAILLE

Trabalho de Conclusão de Curso para obtenção do grau de Bacharel em Ciência da Computação da Universidade de Caxias do Sul.

Aprovado em ___/___/_____

Banca Examinadora

Prof. Dr. André Luis Martinotto
Universidade de Caxias do Sul - UCS

Prof. Dr. Ricardo Vargas Dorneles
Universidade de Caxias do Sul - UCS

Prof. Ms. Alexandre Erasmo Krohn Nascimento
Universidade de Caxias do Sul - UCS

RESUMO

Muitas propriedades dos sólidos podem ser compreendidas através da estrutura cristalina desses materiais. A estrutura cristalina relaciona-se à organização dos átomos que constituem o sólido, sendo que esta pode ser identificada através do uso da técnica de difração de raio X. Diversos softwares foram desenvolvidos para a identificação das estruturas cristalinas a partir de dados obtidos através de técnicas de difração de raio X. Dentre estes, destaca-se o McMaille, que é um programa desenvolvido na linguagem de programação FORTRAN, e que utiliza os métodos de Monte Carlo e *Grid Search* para a classificação da estrutura cristalina. No caso de estruturas cristalinas complexas, o McMaille apresenta um alto custo computacional, devido ao grande número de combinações a serem testadas. Desta forma, neste trabalho foi desenvolvida uma versão paralela do software McMaille, sendo que essa foi desenvolvida para ser executada em computadores com múltiplos processadores e com memória compartilhada, utilizando a biblioteca OpenMP. Inicialmente, foi realizada uma conversão do software McMaille de FORTRAN para a linguagem C utilizando o conversor *f2c*, sendo verificado que a conversão não introduziu erros no programa. Em seguida, foi realizado um perfilamento do código utilizando o programa *gprof*, identificando que a função *calcul1* possui o maior custo computacional do programa. Foram realizadas otimizações nesta função, que resultaram em um ganho de desempenho de até aproximadamente 32%. Após, foi realizada a paralelização dos métodos de Monte Carlo e *Grid Search*. Foi verificado que a paralelização não inseriu erros no programa e verificou-se uma diminuição no tempo de execução, atingindo em casos mais complexos um *speedup* de 2,34 e uma eficiência de 58%.

Palavras-chaves: McMaille. OpenMP.

ABSTRACT

Many properties of solids can be understood by the crystalline structure from these materials. Crystalline structure is related to the atoms organization that compose the solid, which can be identified by the use of the X-ray diffraction technique. Many softwares were developed in order to identify the crystalline structures from data obtained through X-ray diffraction techniques. Among these, is worth highlighting McMaille, which is a software developed using the FORTRAN programming language, and uses Monte Carlo and Grid Search methods to classify the crystalline structure. In the case of complex crystalline structures, McMaille shows a high computational cost, due to the large number of combinations to be tested. Therefore, in this work, a paralell version of the McMaille software was developed, which is executed in computers with multiple cores and with shared memory, using the OpenMP library. Initially, it was made a conversion of the McMaille software from FORTRAN to the C programming language using the *f2c* conversor, and it was verified that the conversion did not introduce errors in the program. Then, a code profiling was made using the *gprof* software, identifying that the function *calcul1* has the higher computational cost. Optimizations were made in this function, resulting in a performance gain of up to approximately 32%. Afterwards, The Monte Carlo and Grid Search methods were parallelized. It was verified that the parallelization did not introduce errors in the program and was verified a decrease in execution time, reaching a speedup of 2.34 and an efficiency of 58% in the more complex cases.

Key-words: McMaille, OpenMP

LISTA DE ILUSTRAÇÕES

LISTA DE TRECHOS DE CÓDIGO

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

OpenMP	*Open Multi-Processing*
f2c	*Fortran-to-C Converter*
gprof	*GNU profiler*
FORTRAN	*Formula Translation*
gcc	*GNU Compiler Collection*
Cs	Césio
Cl	Cloro
Na	Sódio
O	Oxigênio
Si	Silício

LISTA DE SÍMBOLOS

α	Letra grega Alfa
β	Letra grega Beta
γ	Letra grega Gama
θ	Letra grega Teta
λ	Letra grega Lambda
ρ	Letra grega Ró
$\sum$	Somatório
Å	Valor de medida Angström
CuK_α	Radiação de cobre K_α

SUMÁRIO

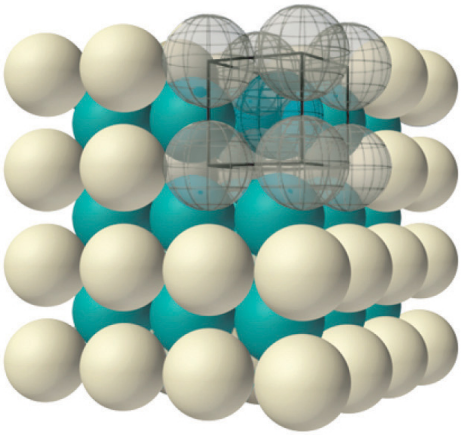
1 INTRODUÇÃO

Estrutura cristalina se refere à forma como os átomos estão agrupados dentro de um sólido, formando um padrão que se repete dentro deste. A forma de representação que se repete em uma estrutura cristalina é chamada célula unitária, sendo que de acordo com o formato da célula unitária, as estruturas cristalinas são classificadas em sete tipos de sistemas, que são: cúbico, hexagonal, tetragonal, romboédrico, ortorrômbico, monoclínico e triclínico, sendo que esses sistemas se diferenciam pela relação entre seus eixos e entre seus ângulos (CALLISTER, 2007).

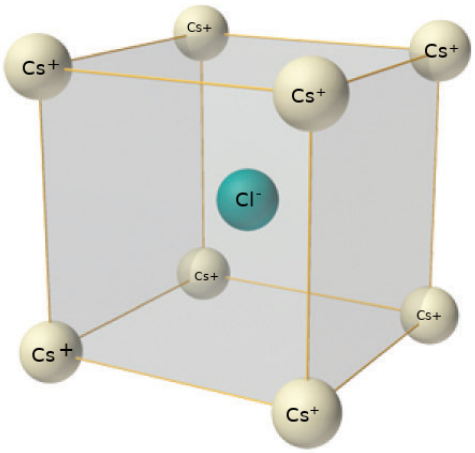
Na Figura 1.1a tem-se a estrutura cristalina do cloreto de césio. Esse possui uma estrutura cristalina cúbica, com os átomos de césio (Cs) organizados de forma que um átomo de césio sempre será seguido por outro de césio, e no centro de oito átomos de césio, sempre haverá um de cloro (Cl). Na Figura 1.1b tem-se a célula unitária do cloreto de césio, sendo que se esta estrutura for replicada é possível reconstruir a estrutura apresentada na Figura 1.1a.

Figura 1.1 – Estrutura cristalina do cloreto de césio

(a) Estrutura cristalina do cloreto de césio



(b) Célula unitária do cloreto de césio



A difração de raio X frequentemente é utilizada para a identificação das posições e a organização dos átomos em um sólido. Nesta técnica, são lançadas ondas de raios X na direção do cristal, e os átomos refletem os raios em outra direção. A partir disso, consegue-se verificar a intensidade dos raios refletidos em cada um dos ângulos e calcular a estrutura cristalina do sólido (CULLITY; STOCK, 2001). A identificação da estrutura cristalina é importante uma vez que possibilita compreender algumas propriedades dos materiais. O carbono, por exemplo, pode assumir diferentes formas, sendo que entre elas estão o grafite, que possui uma superfície mais flexível, e o diamante, que possui a superfície

mais rígida entre os materiais conhecidos. Essa diferença está relacionada à forma como os átomos se organizam nesses sólidos (CALLISTER, 2007).

O McMaille é um programa de código aberto, que foi criado por Armed Le Bail, e que utiliza os métodos de Monte Carlo (METROPOLIS; ULAM, 1949) e *Grid Search* (BERGSTRA; BENGIO, 2012) para a identificação da estrutura cristalina de um sólido. Este apresenta como entrada um arquivo texto com as informações resultantes da difração de raio X e apresenta como saída um arquivo texto com: o sistema cristalino encontrado; o tempo de execução; o volume da célula unitária; e seus prováveis parâmetros (parâmetros de rede e ângulos). Esse software apresenta um alto custo computacional, principalmente na identificação de estruturas cristalinas mais complexas, uma vez que neste caso são necessárias efetuar simulações com um grande conjunto de combinações (BAIL, 2004).

Uma das técnicas que pode ser utilizada para diminuir esse tempo de execução é a programação paralela, que consiste em dividir uma tarefa em partes menores, que podem ser executadas de forma simultânea em diferentes processadores (PACHECO, 1997). Esse tipo de abordagem tornou-se atrativa principalmente pelo fato dos processadores atuais possuírem múltiplos *cores* (PLLANA; XHAFA, 2017), permitindo assim a execução de diferentes processos ou *threads* de forma simultânea (NEGRI et al., 2001).

Dentro deste contexto, neste trabalho foi desenvolvida uma versão paralela do software McMaille. Essa versão foi desenvolvida para ser executada em computadores com memória compartilhada (computadores multiprocessados e computadores com processadores *multicore*) e utilizando múltiplas *threads*. Mais especificamente foi utilizada a biblioteca de *threads* OpenMP (CHAPMAN; JOST; PAS, 2008).

1.1 OBJETIVOS

O principal objetivo deste trabalho consiste no desenvolvimento de uma versão paralela do programa McMaille. Para que o objetivo principal seja atingido, os seguintes objetivos específicos devem ser realizados:

1. Conversão do programa McMaille da linguagem de programação Fortran para a linguagem de programação C.
2. Desenvolvimento de uma versão paralela do programa utilizando a biblioteca OpenMP.
3. Testes para verificar a corretude[1] e o desempenho da versão paralela.

[1] Corretude entende-se por verificar se o programa produz a saída correta para determinada entrada

1.2 ESTRUTURA DO TRABALHO

Este trabalho encontra-se dividido em 4 capítulos, onde:

- No Capítulo 2 é feita uma descrição das estruturas cristalinas de sólidos. Esse capítulo tem como objetivo familiarizar o leitor com o assunto de estudo deste trabalho. Neste será apresentado um exemplo simples para a identificação da estrutura cristalina de um sólido. Uma descrição mais aprofundada e completa pode ser obtida no livro *Elements of X-ray Diffraction* (CULLITY; STOCK, 2001).
- No Capítulo 3 é descrito o processo de conversão do software McMaille para a linguagem C. Além disso, são apresentados os resultados do processo de perfilação do mesmo.
- No Capítulo 4 é mostrado o processo de paralelização do McMaille. Ainda, é demonstrado o processo avaliação da mesma e os resultados obtidos.
- No Capítulo 5 são apresentas as conclusões do trabalho e sugestões de trabalhos futuros.

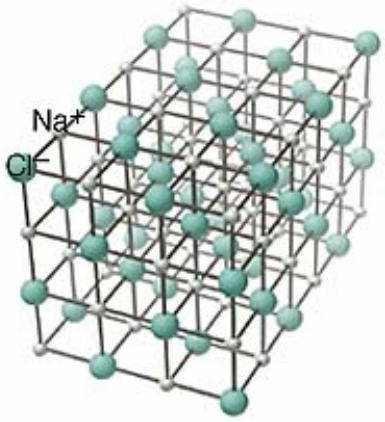
2 ESTRUTURAS CRISTALINAS

De acordo com a organização dos átomos, os materiais sólidos podem ser classificados como cristalinos e amorfos. Os sólidos cristalinos são aqueles em que os átomos estão organizados em um modelo de três dimensões, que se repete após longas distâncias atômicas. Já os amorfos são aqueles que não possuem essa repetição em longas distâncias (CALLISTER, 2007).

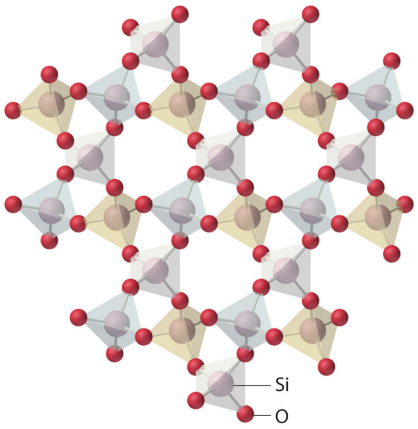
Na Figura 2.1a tem-se um exemplo da estrutura cristalina do cloreto de sódio. Como pode-se observar, os átomos de Sódio (Na) e de Cloro (Cl) estão arranjados de forma que um átomo de Sódio sempre será seguido de um átomo de Cloro, e um átomo de Cloro sempre será seguido por um átomo de Sódio, e assim por diante (BLEICHER; SASAKI, 2000). Já na Figura 2.1b tem-se a estrutura amorfa do dióxido de silício, onde cada átomo de Silício (Si) está conectado a quatro de Oxigênio (O), e cada átomo de Oxigênio está conectado a dois átomos de Silício. A diferença da estrutura apresentada na Figura 2.1a em relação a estrutura da Figura 2.1b está na sua disposição. Enquanto os átomos do cloreto de sódio sempre se repetem após uma certa distância, a estrutura do dióxido de silício apresenta uma forma desordenada, com uma distância diferente entre cada átomo (CALLISTER, 2007).

Figura 2.1 – Exemplo de estruturas cristalinas

(a) Estrutura do cloreto de sódio



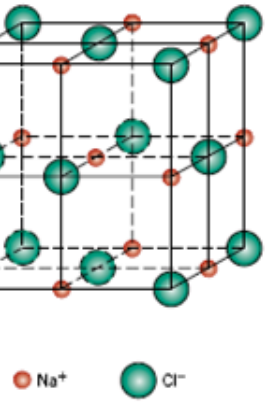
(b) Estrutura do dióxido de silício



As estruturas cristalinas podem assumir diferentes formas, que podem ser subdivididas em unidades menores, que são chamadas de células unitárias. Essa unidade apresenta a forma com que os átomos estão organizados em um cristal, e inclui todos os átomos que participam de sua composição. Deste modo, ao replicar as células unitárias, é possível recompor o sólido original (KOSEVICH, 2006). Na Figura 2.2 pode-se observar a célula

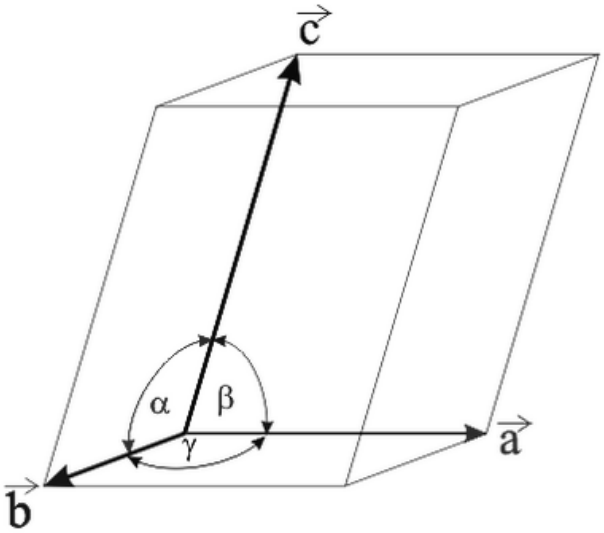
unitária do cloreto de sódio, sendo que a partir da repetição dessa, pode-se reconstruir o sólido formado na Figura 2.1a.

Figura 2.2 – Célula unitária do cloreto de sódio



Existem diferentes tipos de estruturas cristalinas, sendo que essas podem ser classificadas, de acordo com o arranjo dos átomos da célula unitária, em sete tipos, que são: cúbico, hexagonal, tetragonal, romboédrico (ou trigonal), ortorrômbico, monoclínico e triclínico. Esses sistemas cristalinos se diferenciam pelo comprimento de seus eixos (a, b e c), e pelos ângulos gerados entre eles (α, β e γ) (CALLISTER, 2007). Na Figura 2.3 tem-se um exemplo de célula unitária e seus parâmetros. Nela, é possível verificar os parâmetros de rede a, b, e c, e os ângulos α, β e γ.

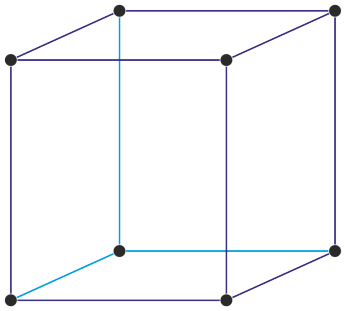
Figura 2.3 – Parâmetros de rede e ângulos da célula unitária



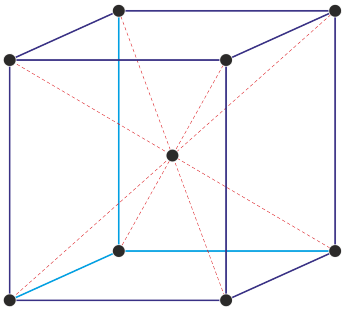
O sistema cristalino cúbico se caracteriza por apresentar os três eixos com o mesmo comprimento ($a = b = c$) e os ângulos entre eles serem iguais a 90º ($\beta = \gamma = \alpha = 90°$). As estruturas cúbicas podem ser encontradas em três formatos diferentes, que são: o sistema cúbico simples, o sistema cúbico de corpo centrado e o sistema cúbico de face centrada. Na Figura 2.4a tem-se a célula unitária de um sistema cúbico simples. Como pode ser observado, ela possui átomos somente nos cantos da mesma. Já na Figura 2.4b tem-se um sistema cúbico de corpo centrado, que possui átomos nos cantos e um átomo no centro da célula unitária. Por fim, na Figura 2.4c tem-se um sistema cúbico de face centrada, que dispõe de átomos nos cantos e nos centros das faces (TISZA, 2001).

Figura 2.4 – Sistema cúbico

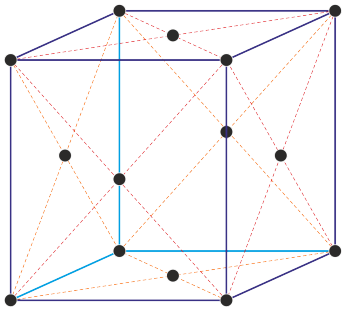
(a) Cúbico simples



(b) Cúbico de corpo centrado

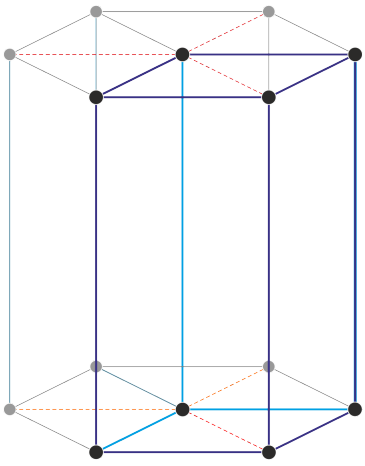


(c) Cúbico de face centrada



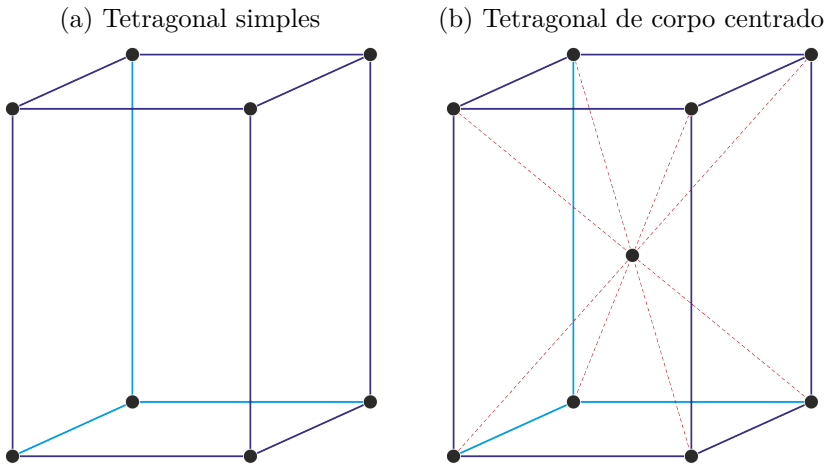
No sistema cristalino hexagonal, temos apenas dois eixos com o mesmo comprimento ($a = b \neq c$), sendo que o ângulo formado entre eles deve ser igual a 120º ($\alpha = 120°$), e o comprimento do terceiro eixo tem que ser diferente dos outros dois, gerando um ângulo de 90º com eles ($\beta = \gamma = 90°$) (TISZA, 2001). Na Figura 2.5 é mostrada a célula unitária de um sistema hexagonal, onde existem seis átomos nos cantos, sendo que cada um deles forma um ângulo de 120º com os seus vizinhos.

Figura 2.5 – Sistema hexagonal



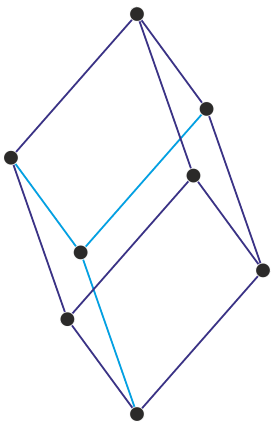
O sistema tetragonal possui todos os seus ângulos iguais a 90º e possui dois eixos iguais, e um terceiro que possui um valor diferente deles ($a = b \neq c$), apresentando assim um formato retangular. Dependendo da posição dos átomos em uma célula unitária tetragonal, esse pode ser classificado em simples ou de corpo centrado. Uma célula tetragonal simples dispõe de átomos somente nos cantos (Figura 2.6a). Já um sistema tetragonal de corpo centrado possui átomos nos cantos e no ponto em que as diagonais formadas pelos átomos se encontram (Figura 2.6b) (TISZA, 2001).

Figura 2.6 – Sistema tetragonal



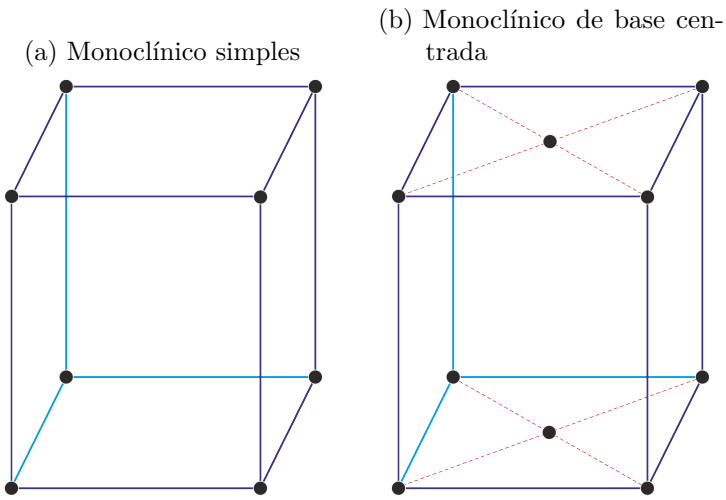
O sistema romboédrico, ou trigonal, apresenta todos os eixos de mesmo comprimento ($a = b = c$), e os ângulos são diferentes de 90º e menores que 120º ($\alpha \neq 90°, < 120°$). Destaca-se que todos os ângulos possuem o mesmo valor ($\alpha = \beta = \gamma$) (PATTERSON; BAILEY, 2007). Na Figura 2.7 tem-se um exemplo de sistema romboédrico, onde cada átomo está conectado a outros três átomos, sendo que o ângulo formando por esses três é sempre o mesmo, fazendo com que cada face da célula unitária seja formada por um losango.

Figura 2.7 – Sistema romboédrico



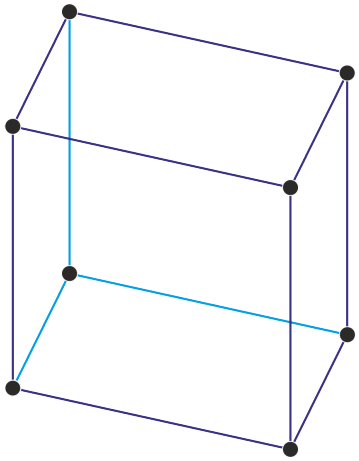
O sistema monoclínico dispõe de dois ângulos iguais a 90º, e um terceiro que é diferente ($\beta = \gamma = 90°$ e $\alpha \neq 90°$). Além disso, seus lados tem dimensões diferentes ($a \neq b \neq c$). Esse sistema apresenta duas formas, que são a simples e a de base centrada. Na Figura 2.8a tem-se o sistema monoclínico simples, que dispõe de átomos somente nos cantos (PATTERSON; BAILEY, 2007). Já na Figura 2.8b tem-se um sistema monoclínico de base centrada, que possui átomos nos cantos e no centro das duas bases.

Figura 2.8 – Sistema monoclínico



O sistema triclínico apresenta todos os lados diferentes ($a \neq b \neq c$) e todos os ângulos diferentes ($\alpha \neq \beta \neq \gamma$) (PATTERSON; BAILEY, 2007). Na Figura 2.9 tem-se um exemplo de um sistema triclínico, onde cada átomo está conectado a três outros átomos, sendo que cada um deles está a uma distância diferente e formando ângulos diferentes.

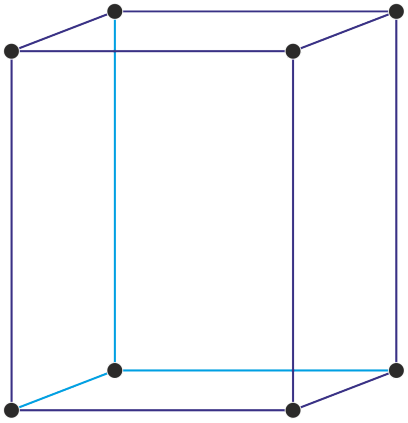
Figura 2.9 – Sistema triclínico



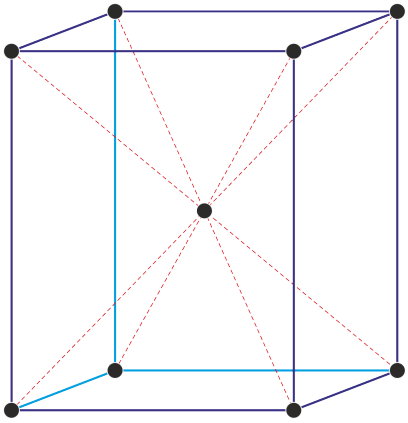
O sistema ortorrômbico possui cada lado com um comprimento diferente ($a \neq b \neq c$), mas todos os seus ângulos são iguais a 90º ($\alpha = \beta = \gamma = 90°$). Existem quatro formas diferentes para esse sistema, que são: o ortorrômbico simples, o ortorrômbico de corpo centrado, o ortorrômbico de base centrada e o ortorrômbico de face centrada. Na Figura 2.10a tem-se o ortorrômbico simples, que apresenta átomos somente nos cantos. Na Figura 2.10b tem-se o ortorrômbico de corpo centrado, que possui átomos nos cantos e um átomo no centro da célula unitária. Já na Figura 2.10c tem-se o ortorrômbico de base centrada, que possui átomos nos cantos e átomos no centro das duas bases. Por fim, na Figura 2.10d tem-se o ortorrômbico de face centrada que possui átomos nos cantos e nos centros de todas as faces (PATTERSON; BAILEY, 2007).

Figura 2.10 – Sistema ortorrômbico

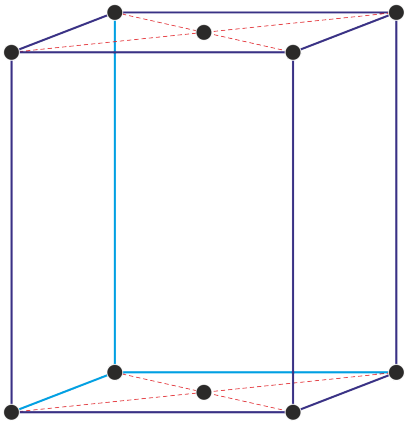
(a) Ortorrômbico simples



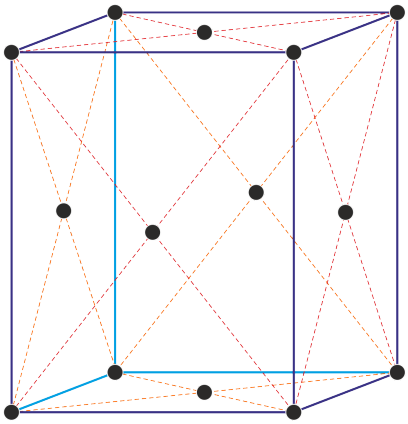
(b) Ortorrômbico de corpo centrado



(c) Ortorrômbico de base centrada

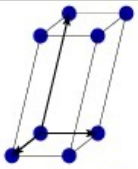
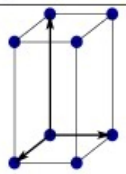
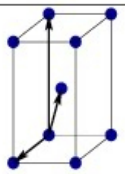
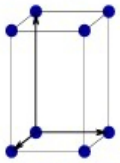
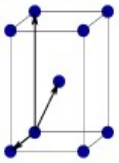
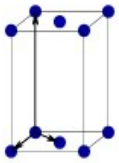
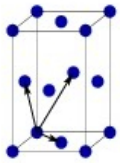
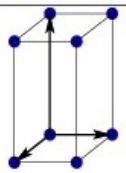
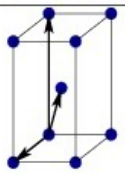
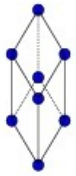
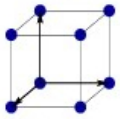
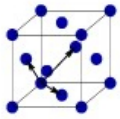
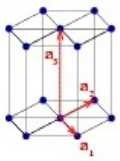


(d) Ortorrômbico de face centrada



Na Tabela 2.1 pode-se observar um resumo dos sistemas cristalinos e suas respectivas formas, totalizando os 14 possíveis tipos de organização das células unitárias. Essas formas são conhecidas como Redes de Bravais (SZWACKI; SZWACKA, 2016).

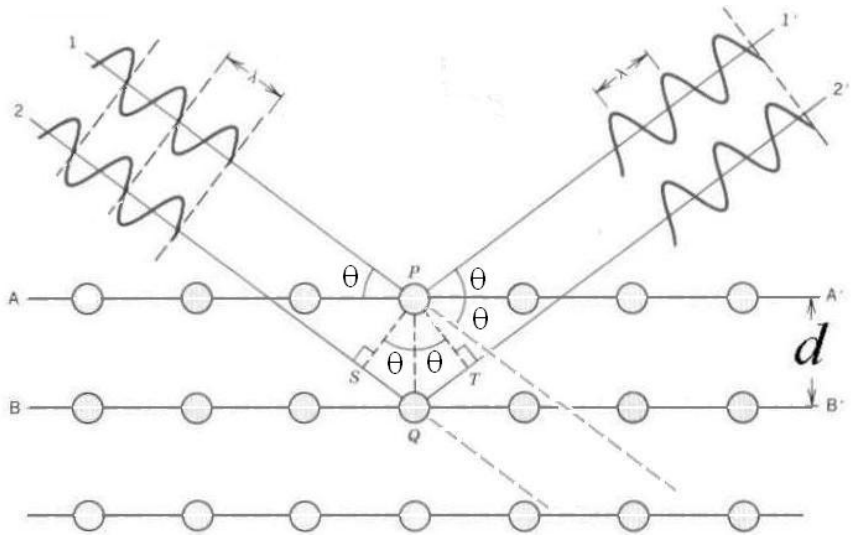
Tabela 2.1 – Sistemas cristalinos e redes Bravais

Sistemas Cristalinos	Parâmetros	Simples	Corpo Centrado	Base Centrada	Face Centrada
Triclínico	$a_1 \neq a_2 \neq a_3$ $\alpha_{12} \neq \alpha_{23} \neq \alpha_{31}$				
Monoclínico	$a_1 \neq a_2 \neq a_3$ $\alpha_{23} = \alpha_{31} = 90°$ $\alpha_{12} \neq 90°$				
Ortorrômbico	$a_1 \neq a_2 \neq a_3$ $\alpha_{12} = \alpha_{23} = \alpha_{31} = 90°$				
Tetragonal	$a_1 = a_2 \neq a_3$ $\alpha_{12} = \alpha_{23} = \alpha_{31} = 90°$				
Trigonal	$a_1 = a_2 = a_3$ $\alpha_{12} = \alpha_{23} = \alpha_{31} < 120°$				
Cúbico	$a_1 = a_2 = a_3$ $\alpha_{12} = \alpha_{23} = \alpha_{31} = 90°$				
Hexagonal	$a_1 = a_2 \neq a_3$ $\alpha_{12} = 120°$ $\alpha_{23} = \alpha_{31} = 90°$				

2.1 DIFRAÇÃO DE RAIO X

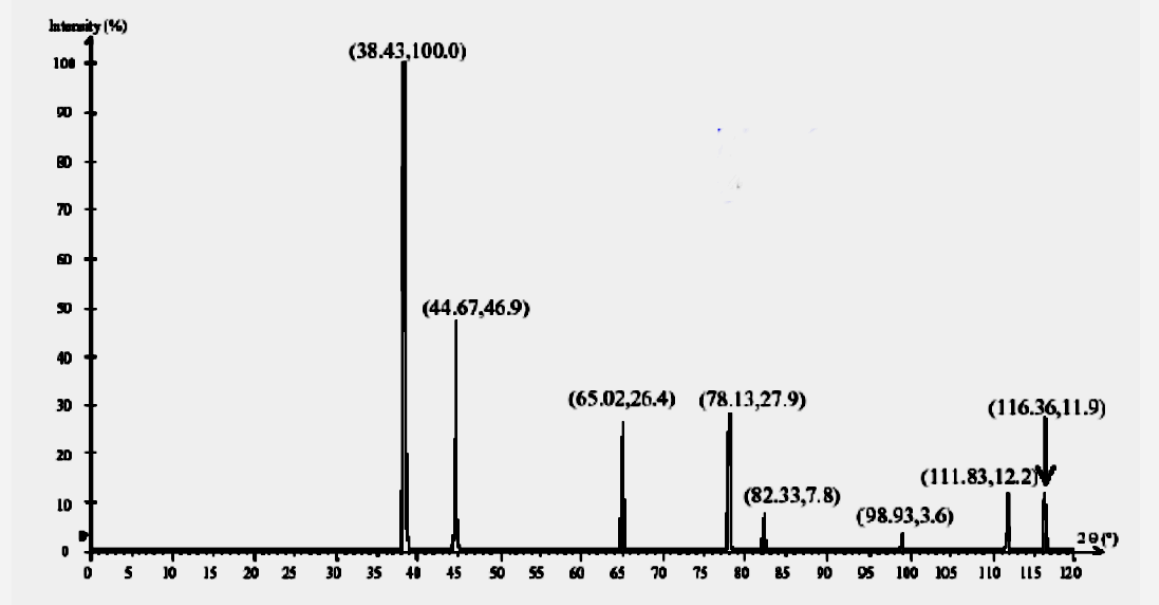
A difração de raio X é a técnica mais utilizada para a identificação da estrutura cristalina de um sólido. Esse fenômeno ocorre quando os raios X são enviados na direção do cristal, sendo que os átomos causam um espelhamento dos raios X em outras direções. A partir desse espelhamento pode-se medir o ângulo de difração dos raios e identificar como os átomos que compõem o cristal estão organizados (CULLITY; STOCK, 2001). Na Figura 2.11 tem-se uma ilustração dos átomos de um sólido recebendo as ondas de raio X, que são desviadas em outra direção, sendo que o ângulo entre o raio enviado e o raio refletido será de 2θ.

Figura 2.11 – Difração de raio X



Para a definição da estrutura de um sólido, primeiramente são lançados raios X com o comprimento de onda λ na direção do mesmo, sendo calculada a difração para cada ângulo de 2θ entre 0^o e 180^o, resultando em um gráfico com a intensidade da difração em relação a cada ângulo (Figura 2.12). Os ângulos onde são encontrados os átomos retornam uma intensidade da difração maior, sendo chamados de picos de intensidade (CULLITY; STOCK, 2001). Na Figura 2.12 tem-se um exemplo do resultado da difração de raio X do alumínio, no qual foram encontrados oito picos de intensidade. A informação de tipo de sistema cristalino pode ser obtido através dos picos de intensidade de cada um dos ângulos 2θ.

Figura 2.12 – Exemplo de resultado da difração de raio X do alumínio



De fato, para cada sistema cristalino tem-se uma relação diferente para os valores de $\text{sen}^2\theta$. Desta forma, é escolhido um dos sistemas cristalinos e é verificado se os dados da difração de raio X atendem essa relação. Se essa relação não for atendida, é escolhido outro sistema cristalino, sendo que esse processo é repetido até encontrar o sistema cristalino correto (CULLITY; STOCK, 2001).

Por exemplo, para um sistema ser considerado cúbico, ele deve satisfazer a relação

$$\frac{\text{sen}^2\theta}{s} = \frac{\lambda^2}{4a^2} \tag{2.1}$$

onde s é um valor inteiro, λ é o comprimento de onda do raio X e a é o parâmetro de rede do sistema cúbico. Uma vez que λ e a são valores constantes, o valor de $\lambda^2/4a^2$ também será um valor constante. Os valores de s possuem uma sequência específica para cada sistema cúbico. Na Tabela 2.2 pode-se ver os primeiros valores da sequência s para cada um dos sistemas cúbicos, sendo que cada um desses conjuntos é testado com a relação anterior (CULLITY; STOCK, 2001).

Tabela 2.2 – Sequência dos valores s para as formas cúbicas

Simples	Corpo Centrado	Face Centrada
1	2	3
2	4	4
3	6	8
4	8	11
5	10	12
6	12	16
8	14	19
9	16	20
10	18	24
11	20	27
12	22	32
13	24	35

Na Tabela 2.3 são apresentados os oito picos de intensidade resultantes da difração de raio X do alumínio (Figura 2.12). Na coluna 2 são mostrados os valores de $\text{sen}^2\theta$ para cada um dos picos de intensidade. Para as formas cúbicas simples, de corpo centrado e de face centrada, são mostrados os valores calculados de $\text{sen}^2\theta/s$ utilizando os valores de s na sequência que foram apresentadas na Tabela 2.2. Assim, pode-se ver que o sistema é cúbico de face centrada, pois o resultado de $\text{sen}^2\theta/s$ não varia significativamente, diferentemente das outras duas formas. O valor de $\text{sen}^2\theta/s$ deve ser igual a uma constante, uma vez que λ e a são constantes (ver Equação 2.1). Se nos três casos houvesse uma variação significativa nos valores de sen^2/s, poderia ser concluído que a estrutura não é cúbica e deveria ser testado outro sistema. As relações para os outros sistemas cristalinos podem ser encontrados no livro *Elements of X-ray Diffraction* (CULLITY; STOCK, 2001).

Tabela 2.3 – Exemplo de classificação do Alumínio

1	2	3	4	5	6	7	8
		Simples		Corpo Centrado		Face Centrada	
Linha	$\text{sen}^2\theta$	s	$\frac{\text{sen}^2\theta}{s}$	s	$\frac{\text{sen}^2\theta}{s}$	s	$\frac{\text{sen}^2\theta}{s}$
1	0.1083	1	0.1083	2	0.0541	3	0.0361
2	0.1444	2	0.0722	4	0.0361	4	0.0361
3	0.2888	3	0.0963	6	0.0481	8	0.0361
4	0.3972	4	0.0993	8	0.0496	11	0.0361
5	0.4333	5	0.0867	10	0.0433	12	0.0361
6	0.5776	6	0.0963	12	0.0481	16	0.0361
7	0.6859	8	0.0857	14	0.0490	19	0.0361
8	0.7220	9	0.0802	16	0.0451	20	0.0361

A partir do valor de $\text{sen}^2\theta/s$ calculado e conhecendo o valor do comprimento de onda λ utilizado na difração de raio X, é possível determinar o parâmetro de rede a, através da Equação 2.1 (CULLITY; STOCK, 2001). Por exemplo, na difração do alumínio, o valor de $\text{sen}^2\theta/s$ é de 0.0361. Considerando que foi utilizada a radiação CuK_α, com o comprimento de onda de 1.5405 Å[1], tem-se que

$$0.0361 = \frac{1.5405^2}{4a^2}, \tag{2.2}$$

uma vez que o valor do parâmetro de rede a deve ser um valor positivo, tem-se

$$a = \sqrt{\frac{1.5405^2}{(4)(0.0361)}} = 4.0539\text{Å} \tag{2.3}$$

Após ser calculado o parâmetro de rede a, pode ser calculado o número de átomos existentes na célula unitária. Para isso, primeiro é necessário calcular o volume da célula unitária. Para os sistemas cúbicos, o volume V é dado por $V = a^3$ (CULLITY; STOCK, 2001). Assim, considerando o exemplo do alumínio, o volume será dado por:

$$V = a^3 = 4.0539^3 = 66.6222\text{Å}^3 \tag{2.4}$$

O volume obtido é multiplicado pela densidade ρ da matéria (g/cm^3), resultando no peso atômico total da célula unitária (CULLITY; STOCK, 2001). Por exemplo, considerando que o volume da célula unitária do alumínio é igual a 66.6222 Å^3, esse resultado será multiplicado pela densidade do alumínio, que é de 2.7 g/cm^3, obtendo-se:

$$\sum A = \rho V = 2.7\frac{g}{cm^3} \times 66.6222 \times (10^{-8}cm)^3 = 179.880 \times 10^{-24}g \tag{2.5}$$

O peso atômico frequentemente é apresentado em g/mol, assim o resultado da Equação 2.5 ($179.880 \times 10^{-24}g$) pode ser multiplicado pela constante de Avogadro ($6.02257 \times 10^{23}mol^{-1}$), obtendo-se:

$$(179.880 \times 10^{-24}g) \times (6.02257 \times 10^{23}mol^{-1}) = 108.334g/mol \tag{2.6}$$

Uma vez que o peso total da célula unitária igual a 108.334 g/mol, e considerando que o peso atômico do alumínio é de 26.9815 g/mol, tem-se que o número de átomos na célula unitária é igual a quatro (considerando erro experimental):

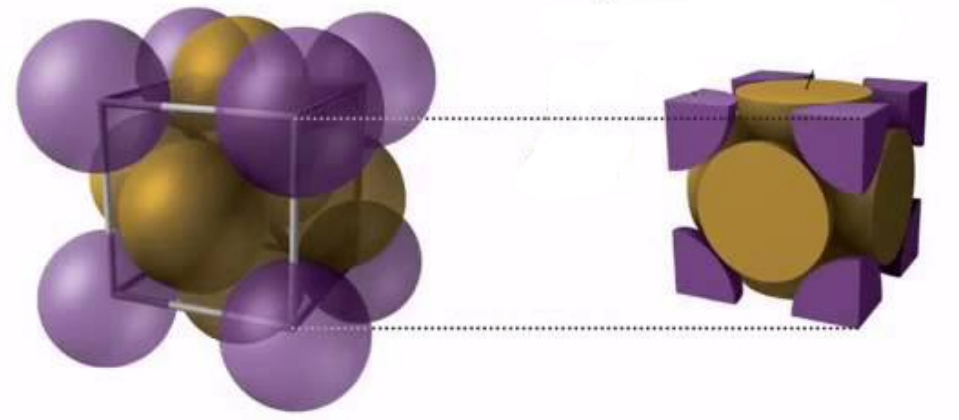
$$n_1 = \frac{108.334g/mol}{26.9815g/mol} = 4.0151 \tag{2.7}$$

[1] Valor medido em Angström

De fato, uma célula unitária cúbica de face centrada possui quatro átomos. Na Figura 2.13 tem-se uma ilustração da célula unitária do alumínio. Para verificar a quantidade de átomos presentes nesta célula unitária, é necessário considerar que um átomo pode pertencer a mais de uma célula. A célula unitária do alumínio tem oito átomos nos cantos da célula, e seis átomos nas faces, sendo que os átomos dos cantos são compartilhados com outras sete células unitárias, sendo que somente 1/8 do átomo pertence a esta célula unitária. Já cada átomo das faces são compartilhados com outra célula unitária, sendo que somente metade do átomo pertence a cada uma delas. Como pode-se observar na Equação 2.8, a célula unitária possui 1/8 de cada um dos oito átomos dos cantos, e 1/2 dos seis átomos das faces, resultando um total de quatro átomos na célula unitária (ASKELAND; FULAY, 2008).

$$(8 \times \frac{1}{8}) + (6 \times \frac{1}{2}) = 4 \tag{2.8}$$

Figura 2.13 – Célula unitária do alumínio



Por fim, a intensidade dos picos da difração é utilizada para descobrir a posição dos átomos da célula unitária. Porém, não existe uma equação ou método exato para isso, sendo necessário utilizar técnicas baseadas em estratégias de simulação e tentativa e erro. Assim, os cálculos das intensidades são feitos utilizando posições hipotéticas para os átomos, e o resultado é comparado com as intensidades reais. Esse processo é repetido até que sejam encontradas as posições que resultem na intensidade observada. Por esse motivo, estruturas cristalinas complexas tendem a levar um tempo longo para serem definidas, sendo que muitas delas ainda não tem suas posições estabelecidas (CULLITY; STOCK, 2001).

3 MCMAILLE

O McMaille é um programa desenvolvido por Armed Le Bail com o objetivo de identificar o sistema cristalino de um sólido. Ele foi escrito na linguagem de programação FORTRAN 77 utilizando o compilador Compaq Visual Fortran (LAWRENCE, 2002), e sua última atualização foi realizada em 2006 (BAIL, 2004).

Existem basicamente dois modos de execução do McMaille, que são: o manual e o *black box*. No Trecho de Código 3.1 tem-se um exemplo do formato do arquivo de entrada do modo manual. O arquivo é definido por sete seções, sendo que as seis primeiras seções apresentam os parâmetros de execução do software e a última seção possui os ângulos 2θ e os picos de intensidade resultantes da difração de raio X.

Trecho de Código 3.1 – Exemplo de arquivo de entrada no modo manual

```
1  ! <titulo>
2  NAC - Monte Carlo e Grid Search
3
4  ! <comprimento de onda> <zeropoint> <modo de execucao>
5  1.54056 0.0 2
6
7  ! Sistemas cristalinos que serao testados
8  ! <cub> <hex\romb> <tet> <ort> <mon> <tri>
9  1 0 0 0 0 0
10
11 ! <par. rede min> <par. rede max> <vol. min> <vol. max> <R3> <R2> <R1>
12 3.0 20.0 100.0 2000.0 0.10 0.20 0.40
13
14 ! Somente para Grid Search
15 ! <incremento parametro de rede> <incremento angulo>
16 0.005 0.01
17
18 ! Somente para Monte Carlo
19 ! <numero testes> <numero de execucoes>
20 10000  1
21
22 ! Informacoes da difracao
23 ! <2-theta> <Intensidade>
24 2.206        272.0
25 17.250       236.0
26 21.197      1473.0
27 24.549      1076.0
28 27.482      1362.0
29 ...
```

Na primeira seção do arquivo de entrada é definido o título de identificação do problema. Na segunda seção é apresentado o comprimento de onda utilizado nas medidas de difração de raio X e o código de qual método será utilizado para processar os dados da difração. Na Tabela 3.1 tem-se todos os métodos que podem ser utilizados, sendo que existem seis modos de execução, três em modo manual e três em modo *black box*.

Tabela 3.1 – Processos disponíveis no McMaille

Código	Modo	Método
0	Manual	Monte Carlo
1	Manual	*Grid Search*
2	Manual	Monte Carlo + *Grid Search*
3	Black box	Monte Carlo em todos os sistemas cristalinos
-3	Black box	Monte Carlo não testando para triclínico
4	Black box	Monte Carlo + *Grid Search* em todos os sistemas cristalinos

Na terceira seção deve ser informado quais os sistemas cristalinos que serão testados, sendo escritos seis números separados por espaço. Se for informado 0, o sistema não será testado. Já se o valor for 1, o sistema será testado, sendo que os sistemas deverão ser informados na seguinte ordem: cúbico, hexagonal ou romboédrico, tetragonal, ortorrômbico, monoclínico e trigonal. Na segunda posição, deve ser informado 1 para hexagonal e 2 para romboédrico.

Na quarta seção devem ser definidos os parâmetros de rede e o volume mínimo e máximo que a célula unitária poderá atingir. Além disso, é necessário definir três valores, que são: R_1, R_2 e R_3, sendo $R_1 > R_2 > R_3$. Esses valores representam a porcentagem de diferença entre os picos de intensidade calculadas e observados, e serão utilizados para verificar convergência do método. De fato, se o valor de R (diferença entre os picos calculados e observados) for maior que R_1, o método de Monte Carlo não será executado sobre a estrutura, ou seja, essa será descartada e será gerada uma outra célula unitária com parâmetros aleatórios (R_1 é utilizado somente pelo método de Monte Carlo). Se o R for menor que R_2, o valor é guardado em uma lista de possíveis soluções. E se a diferença R for menor que R_3, o programa considera esta solução como provável, e o McMaille encerra sua execução (BAIL, 2004).

A quinta seção somente é necessária caso o método de *Grid Search* seja utilizado. Nesta seção, serão informados os valores que serão utilizados para incrementar os parâmetros de rede e os ângulos da célula unitária a cada iteração do método *Grid Search*. Por exemplo, no Trecho de Código 3.1, os parâmetros de rede serão incrementados em 0.005 Å e o ângulo em 0.01°.

A sexta seção é utilizada somente pelo método de Monte Carlo. Nele é definida a quantidade de testes que serão realizados pelo método de Monte Carlo, e o número de vezes que o mesmo será executado. No Trecho de Código 3.1 tem-se um exemplo em que o método será executado uma única vez e no máximo 10.000 iterações. Ou seja, o método será abortado no caso de uma solução não ser encontrada em 10.000 iterações.

No modo *black-box* o usuário informa somente o comprimento de onda do raio X, o processo a ser utilizado (Tabela 3.1) e as posições dos picos de difração, bem como as intensidades das mesmas. Para os outros parâmetros são utilizados valores pré-definidos e são realizados testes com todos os sistemas cristalinos. No Trecho de Código 3.2 tem-se um exemplo de arquivo de entrada para o modo *black box*. O modo *black-box* é recomendado para ser utilizado em uma primeira tentativa. Caso a solução não seja encontrada, deve-se utilizar o modo manual, de modo a dar liberdade ao usuário buscar novas possibilidades com ajustes nos parâmetros de entrada (BAIL, 2004).

Trecho de Código 3.2 – Exemplo de arquivo de entrada no modo *black box*

```
1  ! <titulo>
2  NAC - Monte Carlo e Grid Search
3
4  ! <comprimento de onda> <zeropoint> <modo de execucao>
5  1.54056 0.0 3
6
7  ! Informacoes da difracao
8  ! <2-theta> <Intensidade>
9  2.206        272.0
10 17.250       236.0
11 21.197      1473.0
12 24.549      1076.0
13 27.482      1362.0
14 ...
```

Como saída, tem-se um arquivo de texto que apresenta uma lista de possíveis soluções, ordenadas pela diferença entre os picos de intensidade calculados e pelo volume. Além disso, ao final do arquivo tem-se a sugestão de uma célula unitária com maior probabilidade de ser a solução real (Trecho de Código 3.3). Também são gerados outros arquivos para serem utilizados pelos programas *Crysfire* (SHIRLEY, 2000), *Chekcell* (LAUGIER; BOCHU, 2004) e *Winplotr* (ROISNEL; RODRÍQUEZ-CARVAJAL, 2001).

O *Crysfire* é um programa para classificar estruturas cristalinas, resultando em uma lista de possíveis soluções (SHIRLEY, 2000). Já o *Chekcell* é um software auxiliar que oferece uma resposta mais detalhada de uma estrutura cristalina encontrada, sendo muitas vezes utilizado para detalhar os dados encontrados no *Crysfire* (LAUGIER; BOCHU, 2004). Por fim, o *Winplotr* é uma ferramenta utilizada para obter uma visualização gráfica dos resultados da difração de raio X (ROISNEL; RODRÍQUEZ-CARVAJAL, 2001).

Trecho de Código 3.3 – Exemplo de provável solução do arquivo de saída

```
================================================================================
          THE SELECTION OF THE "BEST" CELL
 smallest Rp ? F(20), M(20) ? V ? highest symmetry ?
            DEPENDS ON YOU, EXCLUSIVELY.

                      However...
   It is suggested that the correct cell could be :
================================================================================

IN F.o.M.  Volume   a        b        c        alpha   beta    gamma   Bravais latt
20 187.87 1186.607 11.1881 11.1881 9.4797 90.000 90.000 90.000  P  Tetra *
    Found            5  time(s) head of the best lists

================================================================================
```

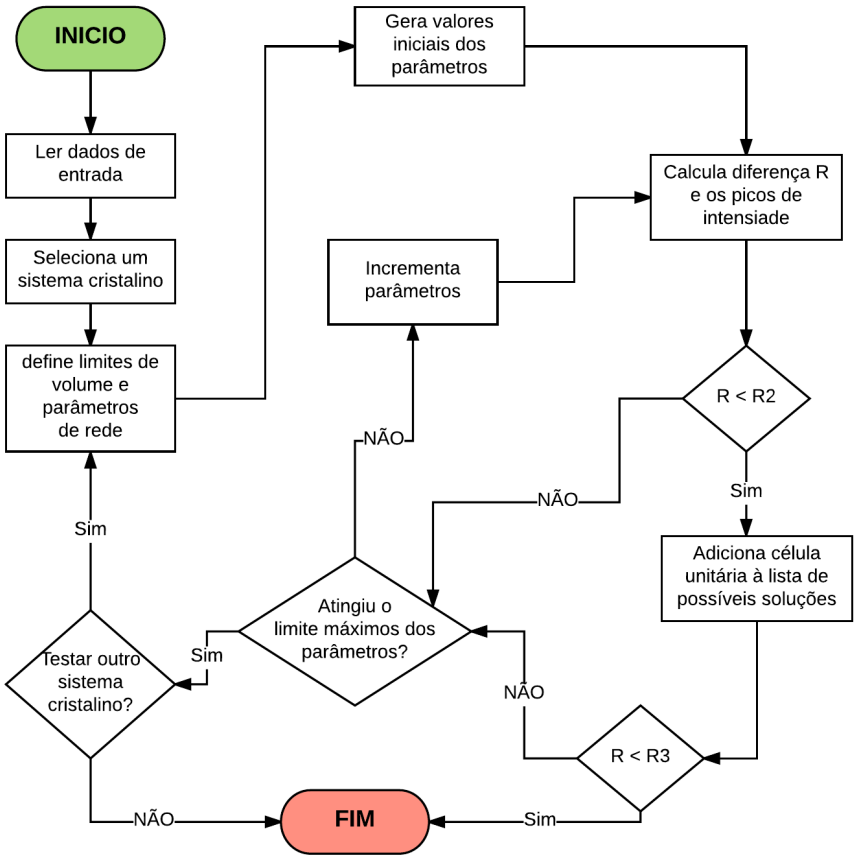
3.1 MÉTODO GRID SEARCH

O *Grid Search* é um algoritmo de "Força Bruta", ou seja, ele testa todas as soluções possíveis dentro de um conjunto de soluções de forma a obter a resposta correta. Este método apresenta uma maior chance de encontrar o resultado correto, porém apresenta um alto custo computacional, sendo inviável para estruturas mais complexas. Desta forma, sua utilização é recomendada para estruturas cristalinas menores e mais simples.

O McMaille utiliza a técnica de *Grid Search* alterando os parâmetros de rede e os ângulos da célula unitária. Para isso, torna-se necessário definir alguns parâmetros, que são: o intervalo do comprimento que os parâmetros de rede poderão assumir; o volume mínimo e máximo da célula unitária; um valor de incremento para os parâmetros de rede da célula unitária; e um de valor de incremento para os ângulos. Esses valores serão incrementados a cada iteração, sendo que se os valores de incremento forem muito baixos, o tempo de execução será muito alto. Mas se esses valores forem muitos altos, muitas configurações de células unitárias não serão testadas, diminuindo assim a possibilidade de encontrar a solução correta.

Desta forma, o *Grid Search* percorre todo o conjunto de possibilidades, calculando os picos de intensidade de cada célula unitária gerada, sendo computadas as diferenças entre o picos calculados e os picos informados no arquivo de entrada. A célula unitária com a menor diferença será escolhida como solução. Na Figura 3.1 tem-se o fluxograma com o funcionamento do método de *Grid Search* no McMaille.

Figura 3.1 – Fluxograma do *Grid Search* no McMaille



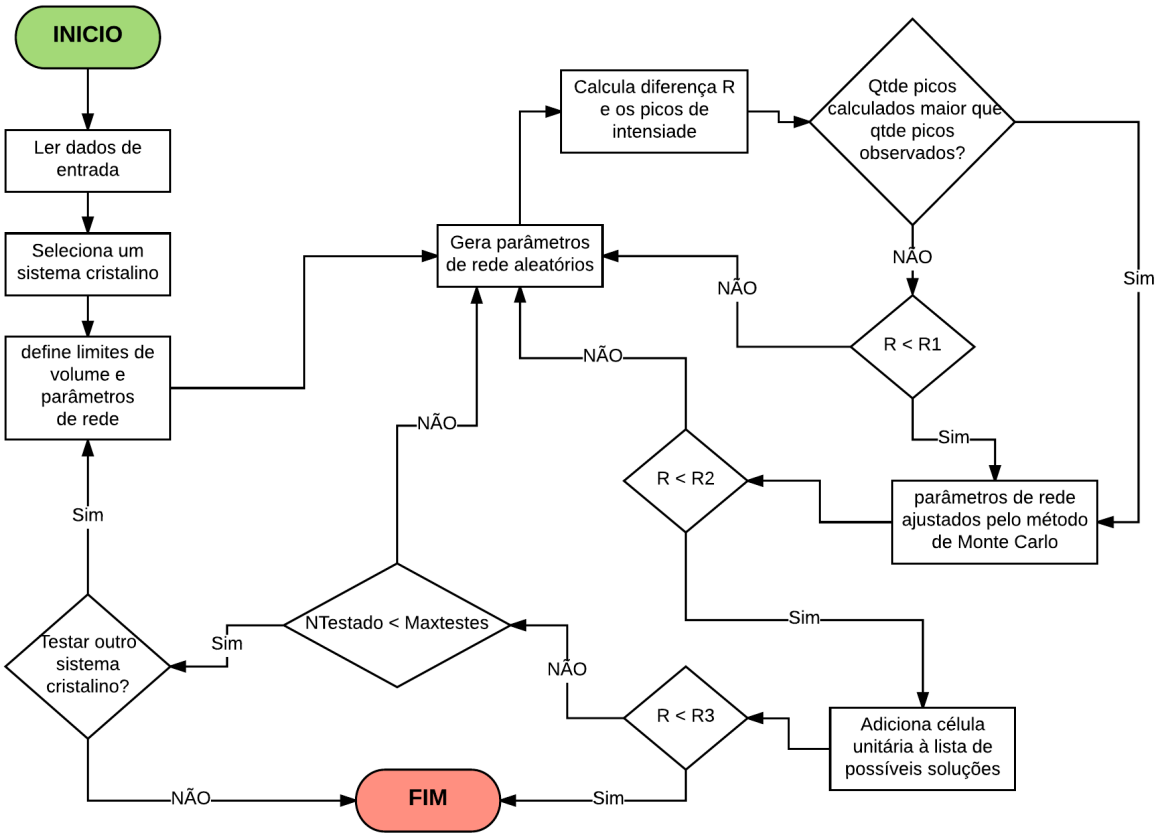
3.2 MÉTODO DE MONTE CARLO

A ideia geral de um método da classe de Monte Carlo é gerar várias amostras aleatórias e observar o comportamento de cada uma delas. Essa classe de métodos é utilizada em casos onde não existem soluções numéricas, ou são muito complexas de serem implementadas. Essa classe de métodos apresenta como resultado uma aproximação da solução real onde, quanto maior a quantidade de amostras utilizadas, mais próximo o resultado ficará do valor real (MOONEY, 1997).

No McMaille, o método de Monte Carlo é executado sobre os parâmetros de rede da célula unitária. Inicialmente é definido o conjunto de possíveis soluções para os mesmos. Para isso, são definidos o comprimento mínimo e máximo que os parâmetros de rede poderão ter, e o volume mínimo e máximo da célula unitária. Esses valores são definidos automaticamente pelo software, caso esse esteja sendo executado em modo *black-box*, ou podem ser definidos pelo usuário, caso esteja executando pelo modo manual. Também pode ser definido a quantidade de vezes que o processo será executado sobre cada célula unitária.

Para iniciar o processo, o algoritmo recebe uma célula unitária com seus parâmetros gerados aleatoriamente e são calculados os picos de intensidade que resultariam da difração de raio X destes parâmetros. Após isso, um parâmetro de rede é escolhido, também de forma aleatória, e é feita uma pequena mudança no mesmo, sendo que o volume da nova célula unitária deve ficar entre os valores mínimo e máximo definidos. Após, são calculados os picos de intensidade que resultariam da difração de raio X da nova célula unitária, sendo calculada a diferença desses picos de intensidade em relação aos que se encontram no arquivo de entrada. Se a diferença R desta célula unitária for menor que a diferença da célula anterior, a nova célula unitária está mais próxima do resultado real. Desta forma, essa é guardada e o processo de Monte Carlo é executado novamente sobre essa célula unitária. Caso contrário, o processo é repetido sobre a célula unitária antiga. Esse processo é repetido até o método encontrar a solução ($R < R_3$) ou o método atingir o número máximo de iterações. Na Figura 3.2 pode-se observar o fluxograma do funcionamento do método de Monte Carlo no McMaille.

Figura 3.2 – Fluxograma do Método de Monte Carlo no McMaille



3.3 CONVERSÃO DO MCMAILLE PARA A LINGUAGEM DE PROGRAMAÇÃO C

Para a conversão do software McMaille, da linguagem de programação FORTRAN para a linguagem de programação C foi utilizado o conversor de código *f2c* na versão 20100827 (FELDMAN et al., 1995). Após a conversão, foi necessário realizar mudanças no código fonte, bem como reescrever ou implementar algumas funções manualmente. O código fonte do programa McMaille convertido para a linguagem de programação C encontra-se em *<https://github.com/runei/McMaille5>*. As funções que tiveram que ser alteradas ou implementadas foram:

- A função *killk* foi implementada. Essa é usada para encerrar o programa caso a tecla *K* tenha sido pressionada. A função utiliza a biblioteca *DFLIB* e a função *peekcharqq*, que são específicas do compilador FORTRAN.
- A função *peekcharqq* que lê um caractere sem retirá-lo do buffer, teve que ser implementada.
- A função *esp_init*, que é utilizada para inicializar a semente do gerador de números aleatórios, foi implementada manualmente. Ela utiliza a biblioteca *ifport* que é específica do compilador Compaq Visual Fortran (INTEL, 2016).
- A função *mcmnam*, que é usada para ler o nome do arquivo de entrada da linha de comando, utilizando as funções *getarg* e *iarg* do compilador FORTRAN (INTEL, 2016), foi substituída pela utilização de argumentos em linhas de comando nativos da linguagem C (SCHILDT, 2000).
- A função *datn*, que grava a data atual formatada no arquivo de saída utilizando a função *date_and_time* (INTEL, 2016), foi reescrita de forma a utilizar a biblioteca *time.h* da linguagem C (SCHILDT, 2000).
- As funções *open*, *read*, *write* e *close* (INTEL, 2016), que são utilizadas para entrada e saída de dados em arquivos, foram substituídas para utilizar ponteiros do tipo *FILE* e as funções *fopen*, *fwrite*, *getline* e *remove* (SCHILDT, 2000).
- Para alguns processos nativos do FORTRAN, como os de entrada e saída e manipulação de *strings*, o conversor *f2c* utiliza funções da biblioteca *libf2c*, que é uma biblioteca criada especificamente para o mesmo. Essas funções foram substituídas por funções nativas da linguagem C, como *printf* e *scanf*, para que o programa não dependa de pacotes externos.

3.4 VALIDAÇÃO DA CONVERSÃO DO MCMAILLE PARA A LINGUAGEM C

Nesta seção serão apresentados os resultados das soluções encontradas pelo programa original (desenvolvido em FORTRAN), e os resultados gerados pelo programa após a conversão para a linguagem C. Esses testes foram realizados com o objetivo de verificar se o processo de conversão não introduziu erros no software McMaille. Os arquivos utilizados para os testes encontram-se em *<https://github.com/runei/McMaille5/blob/master/tests.zip>*. Para comparação serão utilizados os dados da célula unitária considerada como a solução mais provável, mostrando o volume, os parâmetros de rede e os ângulos da mesma. A Tabela 3.2 apresenta os resultados obtidos pelo programa em FORTRAN. Já a Tabela 3.3 mostra os resultados obtidos pelo programa após a conversão para C.

Tabela 3.2 – Resultados esperados

		Linguagem Fortran						
entrada	tipo	vol	a	b	c	α	β	γ
cim	monoclínico	1279,413	10,695	18,816	6,823	90,000	111,282	90,000
cimb	monoclínico	1279,489	10,695	18,815	6,822	90,000	111,275	90,000
nac	cúbico	1078,191	10,254	10,254	10,254	90,000	90,000	90,000
test1	ortorrômbico	378,219	10,027	11,028	3,420	90,000	90,000	90,000
test2	tetragonal	1186,909	11,189	11,189	9,480	90,000	90,000	90,000
test3	ortorrômbico	1154,693	11,032	11,335	9,234	90,000	90,000	90,000
test4	monoclínico	684,718	6,243	12,475	9,190	90,000	106,914	90,000
test5	monoclínico	582,817	10,222	6,496	8,807	90,000	94,697	90,000
test6	triclínico	182,280	7,466	5,117	5,511	90,207	105,500	64,895
test7	cúbico	13744,061	23,954	23,953	23,953	90,000	90,000	90,000
Test8	monoclínico	149,304	5,070	5,856	5,030	90,000	91,483	90,000
Test9	triclínico	984,681	16,791	8,790	7,084	85,866	109,137	89,389
y2o3	cúbico	1190,458	10,598	10,598	10,598	90,000	90,000	90,000
Al2O3	romboédrico	254,896	4,759	4,759	12,995	90,000	90,000	120,000
dicvol1	monoclínico	2597,067	13,392	13,809	14,048	90,000	91,540	90,000
dicvol2	triclínico	515,561	6,915	9,001	8,745	98,957	81,289	77,646

Tabela 3.3 – Resultados obtidos

		Linguagem C						
entrada	tipo	vol	a	b	c	α	β	γ
cim	monoclínico	1279,594	10,682	18,907	6,799	90,000	111,093	90,000
cimb	monoclínico	1280,077	10,697	18,818	6,823	90,000	111,281	90,000
nac	cúbico	1078,170	10,254	10,254	10,254	90,000	90,000	90,000
test1	ortorrômbico	378,165	10,025	11,030	3,419	90,000	90,000	90,000
test2	tetragonal	1187,477	11,185	11,185	9,491	90,000	90,000	90,000
test3	ortorrômbico	1150,511	11,019	11,319	9,223	90,000	90,000	90,000
test4	monoclínico	684,703	6,238	12,490	9,184	90,000	106,908	90,000
test5	monoclínico	586,372	10,232	6,512	8,827	90,000	94,550	90,000
test6	triclínico	182,361	7,997	5,118	5,513	89,852	113,091	63,975
test7	cúbico	13748,945	23,956	23,956	23,956	90,000	90,000	90,000
Test8	monoclínico	149,401	5,070	5,855	5,033	90,000	91,494	90,000
Test9	triclínico	984,681	16,790	8,986	7,239	77,304	109,277	92,261
y2o3	cúbico	1190,467	10,598	10,598	10,598	90,000	90,000	90,000
Al2O3	romboédrico	254,908	4,759	4,759	12,994	90,000	90,000	120,000
dicvol1	monoclínico	2596,917	13,321	13,736	13,965	90,000	90,638	90,000
dicvol2	triclínico	515,723	6,916	9,000	8,747	98,725	98,925	102,374

Como pode-se observar, as soluções obtidas pelo McMaille após a conversão atingiram resultados próximos aos resultados apresentados pelo software em FORTRAN. Deve-se ressaltar que, pelo fato de os métodos de Monte Carlo e *Grid Search* retornarem apenas uma aproximação da solução real, os resultados das duas versões não irão ser idênticos.

3.5 AVALIAÇÃO DO SOFTWARE MCMAILLE

Posteriormente, foram realizados testes de modo a verificar se o tempo de execução do software McMaille apresentou uma variação após a conversão. Para a realização dos testes, foi utilizado um computador com o processador Intel Core i7-5500U de 2.40 GHz com 4 *cores*. Este possui uma cache L1 de 32 KB por core, cache L2 de 256 KB por core e cache L3 de 4 MB compartilhada entre todos os *cores*. O computador possui ainda 8 GB de RAM do tipo DDR3. Já o sistema operacional utilizado foi o Ubuntu 17.10 64-bit com o Kernel 4.13.0-17.

Para a apresentação dos resultados foi considerado o tempo total de execução do programa. Foram testadas as etapas do método de Monte Carlo e *Grid Search* para picos de intensidade pertencentes a cada um dos sete sistemas cristalinos. Para o sistema triclínico não foi utilizado o *Grid Search*, pois o mesmo não está implementado para este tipo de sistema cristalino.

Na Tabela 3.4 tem-se os parâmetros utilizados no arquivo de entrada, sendo que estes parâmetros foram utilizados para todos os testes. Os arquivos de testes encontram-se em *<https://github.com/runei/McMaille5/blob/master/testesDesempenho.zip>*.

Tabela 3.4 – Parâmetros para testes de desempenho

Descrição	Valor
Parâmetro de rede mínimo	3
Parâmetro de rede máximo	20
Volume mínimo	100
Volume máximo	2000
Incremento para *Grid Search*	0,01
Máximo de testes Monte Carlo	100.000.000

A Tabela 3.5 apresenta um comparativo de tempo de execução das versões do McMaille em linguagem C e em linguagem FORTRAN. O tempo médio foi calculado após a realização de 50 execuções para os sistemas cristalinos cúbico, hexagonal, romboédrico e tetragonal, de 20 execuções para os sistemas ortorrômbico e monoclínico, e 5 para o triclínico. Foi considerado também que, caso seja encontrada uma solução com menos que 5% de diferença para os picos de intensidade inseridos pelo usuário, esta é considerada a solução correta e o programa é encerrado. Como pode-se observar na Tabela 3.5, os sistemas cristalinos mais simples (cúbico, hexagonal, romboédrico e tetragonal), por apresentarem uma quantidade menor de parâmetros, apresentam tempos consideravelmente menores que os sistemas cristalinos de maior complexidade (monoclínico e triclínico). Observa-se ainda que o tempo de execução da aplicação em linguagem C apresentou um desempenho superior, chegando a um tempo duas vezes menor que o tempo da aplicação em linguagem FORTRAN, como no caso do sistema ortorrômbico. Destaca-se que foi utilizada a diretiva de otimização *O3* do compilador *gcc* (STALLMAN, 2018).

Tabela 3.5 – Tempos de execução das versões em linguagem C e FORTRAN

	Tempo Médio (seg)			
	Linguagem C		Linguagem FORTRAN	
Sistema Cristalino	Monte Carlo	*Grid Search*	Monte Carlo	*Grid Search*
Cúbico	0,128	0,018	0,370	0,215
Hexagonal	1,072	22,74	1,801	35,790
Romboédrico	1,479	19,293	2,735	34,576
Tetragonal	0,302	9,355	0,436	27,545
Ortorrômbico	1,933	464,353	2,287	836,493
Monoclínico	84,384	-	136,378	-
Triclínico	822,44	-	1.144,308	-

3.6 PERFILAÇÃO DO CÓDIGO FONTE

Para a identificação dos trechos de código que apresentam um maior custo computacional, foi utilizada a ferramenta *gprof* (FENLASON; STALLMAN, 2009). A partir de seu uso, foi possível identificar o tempo de execução de cada uma das funções do McMaille. Para realizar o perfilamento, foram considerados os dois algoritmos implementados. No primeiro caso foi utilizado o modo *black box*, que testa para todos os sistemas cristalinos, com o método de Monte Carlo. Para esse teste foi utilizado o arquivo *cim* (Tabela 3.3) que resulta em um sistema cristalino monoclínico. No segundo caso foi executado o método *Grid Search* sobre o arquivo *test2* que resulta em um sistema cristalino tetragonal. Os arquivos de testes encontram-se disponíveis em *<https://github.com/runei/McMaille5/blob/master/tests.zip>*. No Trecho de Código 3.4 e no Trecho de Código 3.5 encontram-se os resultados gerados pelo *gprof*.

No primeiro caso (Trecho de Código 3.4) o tempo de execução total foi de 118,56 segundos, onde a função para calcular a diferença entre as intensidades calculadas e observadas (*calcul1*) ocupou 98,8% desse tempo. Já a função que calcula a célula recíproca (*dcell*) ocupou 0,8% do tempo, sendo 0,62% deste tempo foi ocupada pela função que transforma uma célula unitária em célula recíproca (*trcl*), sendo executada 23.862.305 vezes. Por fim, a função que gera números aleatórios (*randi*) ocupou 0,3% do tempo. As demais funções não apresentaram um impacto significativo no tempo total de execução. A perfilação do código completa pode ser encontrada no Apêndice A.

Trecho de Código 3.4 – Saída *gprof* para perfilamento do método de Monte Carlo

```
index % time    self  children    called     name
                                                 <spontaneous>
[1]     98.8   93.41     0.00                 calcul1 [1]
-----------------------------------------------
                                                 <spontaneous>
[2]      0.8    0.17     0.62                 dcell [2]
                0.62     0.00 23862305/23862305     trcl [3]
-----------------------------------------------
                0.62     0.00 23862305/23862305     dcell [2]
[3]      0.7    0.62     0.00 23862305          trcl [3]
-----------------------------------------------
                                                 <spontaneous>
[4]      0.3    0.32     0.00                 randi [4]
-----------------------------------------------
...
```

No segundo caso (Trecho de Código 3.5) o tempo de execução total foi de 38,77 segundos, sendo que o método *calcul1* (diferença entre as intensidades calculadas e observadas) ocupou 99,5% desse tempo. Novamente, as demais funções não apresentam impacto no tempo total de execução. A perfilação do código completa pode ser encontrada no Apêndice B.

Trecho de Código 3.5 – Saída *gprof* da perfilação do *Grid Search*

```
index % time    self  children    called     name
                                                 <spontaneous>
[1]     99.5   31.64    0.00                   calcul1 [1]
-----------------------------------------------
                                                 <spontaneous>
[2]      0.5    0.07    0.08                   dcell [2]
                0.08    0.00 11877196/11877196      trcl [3]
-----------------------------------------------
                0.08    0.00 11877196/11877196      dcell [2]
[3]      0.3    0.08    0.00 11877196          trcl [3]
-----------------------------------------------
...
```

Os resultados do *gprof* demonstraram que a função *calcul1* é a que apresenta um maior tempo de execução, tanto para o método de Monte Carlo como para o *Grid Search*. Desta forma, as otimizações no código do McMaille foram concentradas na função *calcul1*, que possui o maior custo computacional do programa. Verificou-se que um dos motivos desse alto custo computacional era o cálculo do arco seno, realizado diversas vezes durante a execução da função. Foi constatado que a maioria dos valores de entrada para esta função são entre 0 e 1. Considerando isso, foram realizados testes com valores pré-calculados do arco seno entre 0 e 1, variando a quantidade de valores pré-calculados. Assim, observou-se que os testes convergiram em 2^{15} valores, com um intervalo entre eles de aproximadamente 0,0000305, e essa foi a quantidade de valores pré-calculados utilizada. Se o número de entrada não estiver entre 0 e 1, é utilizada a função do arco seno padrão da biblioteca *math.h* da linguagem C.

Na Tabela 3.6 é apresentado o tempo de execução do McMaille antes e depois dessa otimização. Como pode ser observado, essa alteração proporcionou uma redução de até 32% em alguns casos.

Tabela 3.6 – Tempos de execução

	Tempo Médio (seg)			
	Antes das otimizações		Após as otimizações	
Sistema Cristalino	Monte Carlo	*Grid Search*	Monte Carlo	*Grid Search*
Cúbico	0,128	0,018	0,092	0,018
Hexagonal	1,072	22,740	0,841	15,352
Romboédrico	1,518	19,293	1,384	13,760
Tetragonal	0,302	9,355	0,260	7,518
Ortorrômbico	1,933	464,353	1,299	370,137
Monoclínico	84,384	-	69,651	-
Triclínico	822,44	-	692,499	-

4 PARALELIZAÇÃO DO MCMAILLE

Neste capítulo será descrito o processo de paralelização dos métodos de Monte Carlo e *Grid Search*. Além disso, será descrito o processo de validação da implementação paralela, bem como os resultados obtidos.

4.1 PARALELIZAÇÃO DO MÉTODO DE MONTE CARLO

O algoritmo de Monte Carlo possui como entrada uma célula unitária com os parâmetros gerados de forma aleatória e são calculados os picos de intensidade da difração de raio X para essa célula unitária. Após isso, o programa é dividido em N *threads*, sendo que cada *thread* é executada em um *core*. Em cada *thread*, um parâmetro de rede da célula unitária inicial é escolhido aleatoriamente, sendo feita uma pequena alteração neste. Destaca-se que o volume da nova célula unitária deve ficar entre os valores mínimo e máximo definidos.

Após, cada *thread* calcula os picos de intensidade resultantes da difração de raio X da nova célula unitária. Ao final desse processo, cada *thread* verifica se a diferença R calculada por ela é menor que a diferença R da célula unitária inicial. Caso a diferença R seja menor tem-se que essa nova célula encontra-se mais próxima do resultado real. Desta forma, essa é guardada e o processo de Monte Carlo de todas as *threads* serão executados a partir desta nova célula unitária. Caso contrário, o processo é repetido sobre a célula unitária antiga.

Para evitar as *threads* entrem em conflito, como, por exemplo, duas *threads* substituírem a melhor célula unitária ao mesmo tempo, a comparação entre a nova célula e a antiga e a substituição da nova é executada em uma área de exclusão mútua. Esse processo é repetido até alguma *thread* encontrar a solução ($R < R_3$) ou a soma das iterações de todas as *threads* atingir o máximo de iterações. Nas Figuras 4.1 e 4.2 pode-se observar o fluxograma do funcionamento do método de Monte Carlo no McMaille. Na Figura 4.1, tem-se o fluxograma da versão sequencial. Já na Figura 4.2 tem-se o fluxograma da versão paralela.

Figura 4.1 – Fluxograma do método de Monte Carlo sequencial

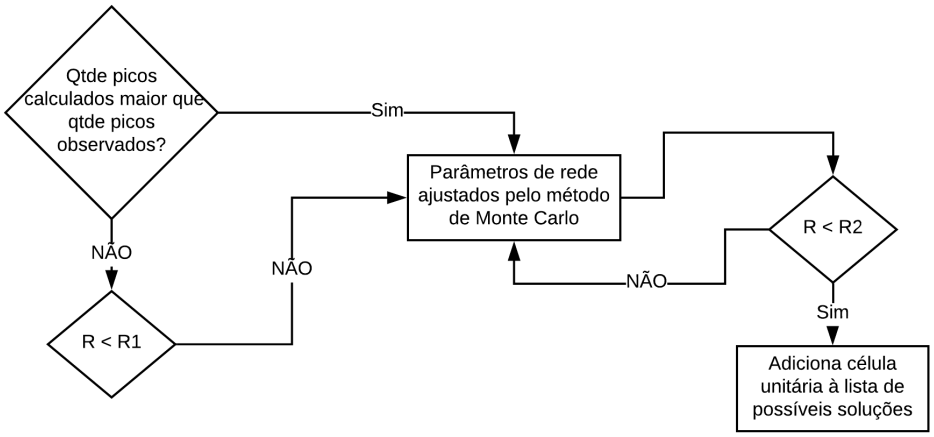
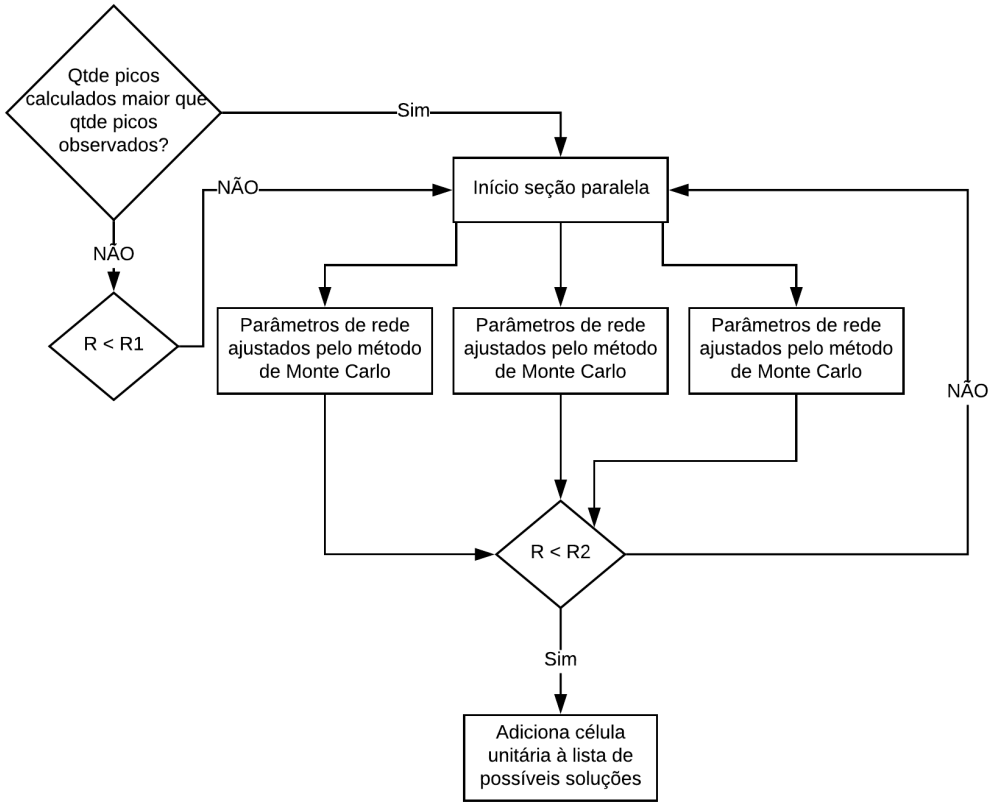


Figura 4.2 – Fluxograma do método de Monte Carlo paralelo



4.2 PARALELIZAÇÃO DO MÉTODO DE GRID SEARCH

A paralelização do método de *Grid Search* foi realizada dividindo-se o espaço de busca em partes menores, sendo que cada uma dessas partes é executada por uma *thread*. Assim, para cada *thread* é definido um subespaço de busca, onde em cada iteração, são calculados os picos de intensidade de cada célula unitária gerada e é calculada as diferenças entre o picos calculados e os picos informados no arquivo de entrada. A célula unitária com a menor diferença será escolhida como solução. Nas Figuras 4.3 e 4.4 tem-se o fluxograma com o funcionamento do método de *Grid Search* no McMaille. Na Figura 4.3 tem-se o fluxograma de execução da versão sequencial e na Figura 4.4 o fluxograma de execução da versão paralela.

Figura 4.3 – Fluxograma do método de *Grid Search* sequencial

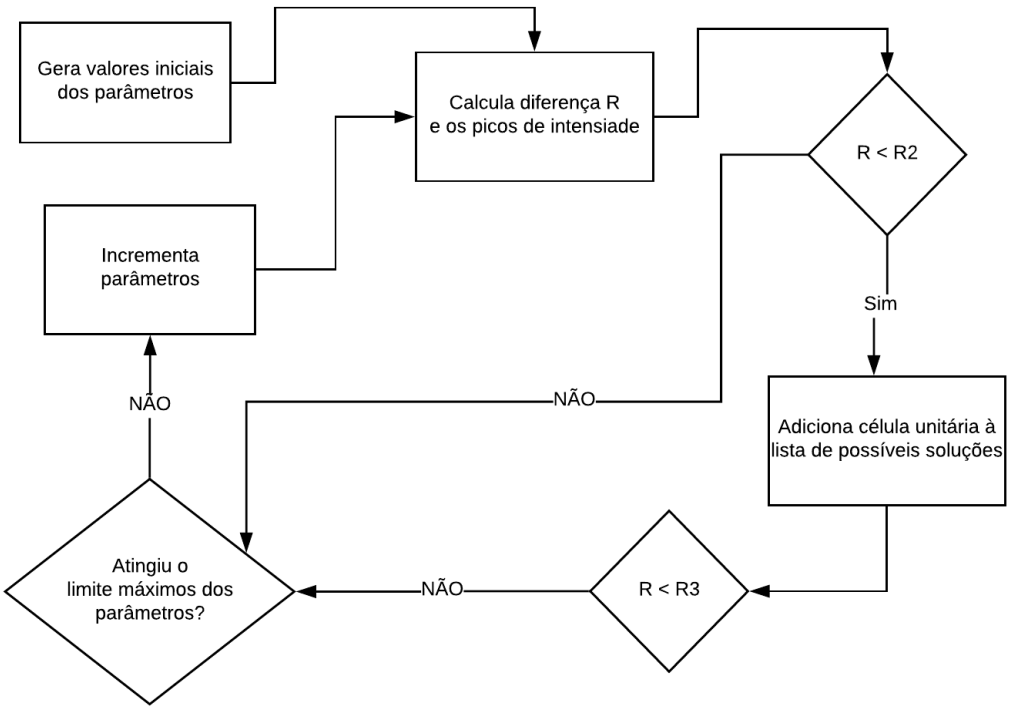
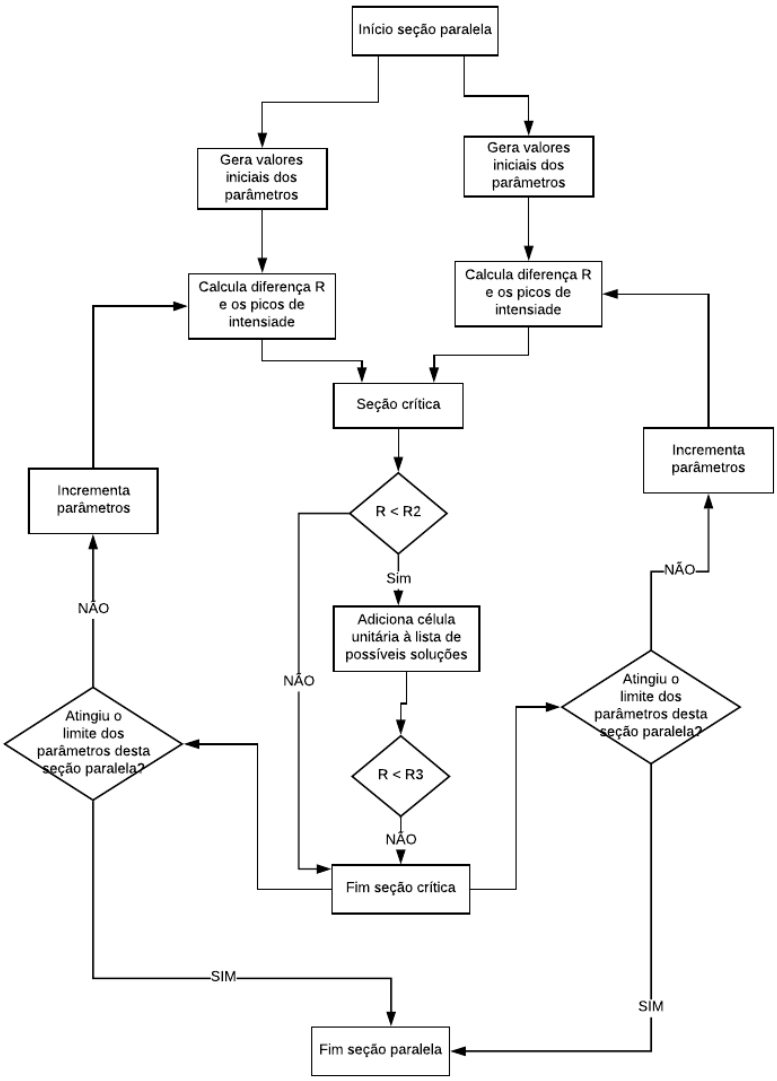


Figura 4.4 – Fluxograma do método de *Grid Search* paralelo



4.3 VALIDAÇÃO DA PARALELIZAÇÃO DO MCMAILLE

Nesta seção serão comparados os resultados obtidos pelo programa sequencial, e os resultados gerados pelo programa após a paralelização. Esses testes foram realizados com o objetivo de verificar se o processo de paralelização não introduziu erros no software McMaille. Os arquivos utilizados para os testes encontram-se em *<https://github.com/runei/McMaille5/blob/master/tests.zip>*. Para comparação serão utilizados o volume, os parâmetros de rede e os ângulos da célula unitária que foi considerada a solução mais provável. A Tabela 4.1 apresenta os resultados obtidos pela versão sequencial do McMaille e a Tabela 4.2 mostra os resultados obtidos pela versão paralela utilizando 4 *threads*.

Tabela 4.1 – Resultados esperados

		Versão Sequencial						
entrada	tipo	vol	a	b	c	α	β	γ
cim	monoclínico	1279,594	10,682	18,907	6,799	90,000	111,093	90,000
cimb	monoclínico	1280,077	10,697	18,818	6,823	90,000	111,281	90,000
nac	cúbico	1078,170	10,254	10,254	10,254	90,000	90,000	90,000
test1	ortorrômbico	378,165	10,025	11,030	3,419	90,000	90,000	90,000
test2	tetragonal	1187,477	11,185	11,185	9,491	90,000	90,000	90,000
test3	ortorrômbico	1150,511	11,019	11,319	9,223	90,000	90,000	90,000
test4	monoclínico	684,703	6,238	12,490	9,184	90,000	106,908	90,000
test5	monoclínico	586,372	10,232	6,512	8,827	90,000	94,550	90,000
test6	triclínico	182,361	7,997	5,118	5,513	89,852	113,091	63,975
test7	cúbico	13748,945	23,956	23,956	23,956	90,000	90,000	90,000
Test8	monoclínico	149,401	5,070	5,855	5,033	90,000	91,494	90,000
Test9	triclínico	984,681	16,790	8,986	7,239	77,304	109,277	92,261
y2o3	cúbico	1190,467	10,598	10,598	10,598	90,000	90,000	90,000
Al2O3	romboédrico	254,908	4,759	4,759	12,994	90,000	90,000	120,000
dicvol1	monoclínico	2596,917	13,321	13,736	13,965	90,000	90,638	90,000
dicvol2	triclínico	515,302	6,916	9,000	8,747	98,725	98,925	102,374

Tabela 4.2 – Resultados obtidos

		Versão Paralela						
entrada	tipo	vol	a	b	c	α	β	γ
cim	monoclínico	1279,295	10,696	18,813	6,823	90,000	111,288	90,000
cimb	monoclínico	1279,305	10,391	18,811	6,823	90,000	106,436	90,000
nac	cúbico	1078,170	10,254	10,254	10,254	90,000	90,000	90,000
test1	ortorrômbico	378,156	10,027	11,028	3,420	90,000	90,000	90,000
test2	tetragonal	1187,477	11,185	11,185	9,492	90,000	90,000	90,000
test3	ortorrômbico	1143,044	11,020	11,280	9,209	90,000	90,000	90,000
test4	monoclínico	684,412	6,241	12,473	9,188	90,000	106,925	90,000
test5	monoclínico	582,645	10,221	6,495	8,805	90,000	94,694	90,000
test6	triclínico	182,276	7,467	5,119	5,510	90,211	105,513	64,889
test7	cúbico	13744,619	23,952	23,952	23,952	90,000	90,000	90,000
Test8	monoclínico	149,330	5,070	5,858	5,031	90,000	91,503	90,000
Test9	triclínico	984,161	16,787	8,986	7,241	70,682	102,736	92,322
y2o3	cúbico	1190,681	10,599	10,599	10,599	90,000	90,000	90,000
Al2O3	romboédrico	254,875	4,759	4,759	12,994	90,000	90,000	120,000
dicvol1	monoclínico	2595,985	13,390	13,810	14,043	90,000	91,512	90,000
dicvol2	triclínico	515,723	8,869	9,459	8,796	79,517	97,437	102,425

Como pode ser observado, as soluções obtidas pelo McMaille após a paralelização atingiram resultados próximos aos resultados apresentados pela versão sequencial. Deve-se ressaltar que, pelo fato de os métodos de Monte Carlo e *Grid Search* retornarem apenas uma aproximação da solução real, os resultados das duas versões não irão ser idênticos.

4.4 COMPARATIVO DE DESEMPENHO DA VERSÃO SEQUENCIAL E PARALELA

De forma a avaliar a paralelização dos métodos de Monte Carlo e *Grid Search*, foi utilizado o tempo de execução total, o *speedup* e a eficiência dos dois algoritmos em diferentes sistemas cristalinos. O *speedup* indica quantas vezes o programa paralelo é mais rápido do que a versão sequencial. Ele é obtido através da Equação 4.1, onde o *speedup* é resultado da razão entre o tempo de execução utilizando apenas uma *thread* (T_1), e o tempo de execução utilizando N *threads* (T_N) (GALLIVAN et al., 1990).

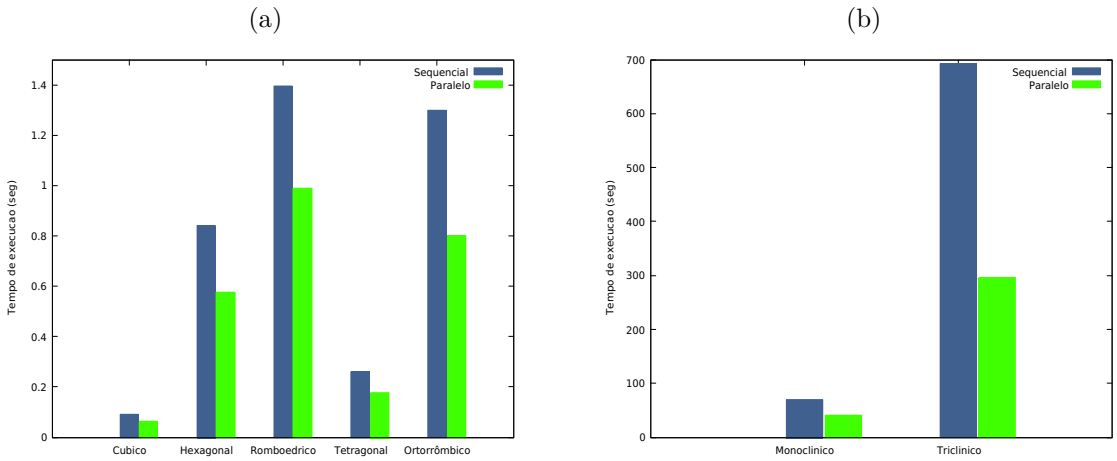
$$sp = \frac{T_1}{T_N} \tag{4.1}$$

A eficiência mostra o percentual de utilização dos recursos computacionais disponíveis, sendo que uma eficiência igual a 1 corresponde a um uso de 100% dos recursos computacionais. A eficiência é calculada pela divisão do *speedup* pelo número de processadores, como é apresentado na Equação 4.2 (GALLIVAN et al., 1990).

$$ef = \frac{sp}{P} \tag{4.2}$$

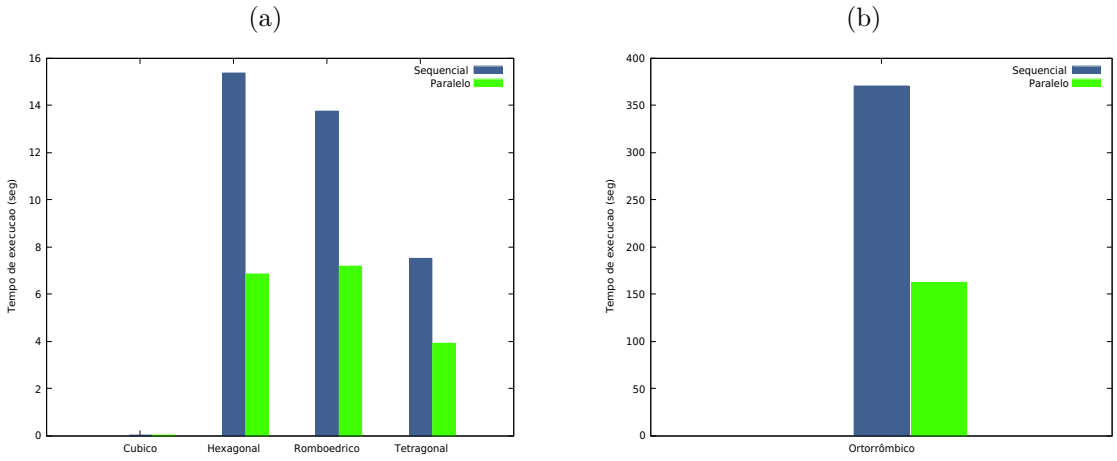
A Figura 4.5 apresenta um comparativo de desempenho entre a versão sequencial e a versão paralela do McMaille. O Apêndice C apresenta uma tabela com os tempos de execução das duas versões. O tempo médio foi calculado após a realização de 20 execuções para cada sistema cristalino. Para a versão paralela, a execução foi realizada utilizando 4 *threads*. Como pode ser observado, a paralelização resultou em um ganho de desempenho em todos os sistemas cristalinos. A Figura 4.5a apresenta os resultados para os sistemas cúbico, hexagonal, romboédrico, tetragonal e ortorrômbico. Já na Figura 4.5b é apresentado um comparativo dos sistemas monoclínico e triclínico. Observa-se que o sistema triclínico, que é o sistema que possui um maior custo computacional, apresentou a maior redução após a paralelização, possuindo o *speedup* de 2,34 e uma eficiência de 58,5%.

Figura 4.5 – Comparativo de desempenho do método de Monte Carlo



Na Figura 4.6 é apresentado o desempenho das versões sequencial e paralela do método *Grid Search*. Percebe-se que para este método também se obteve um tempo de execução inferior na versão paralela, chegando a atingir 2,28 de *speedup* e 57% de eficiência no sistema cristalino ortorrômbico.

Figura 4.6 – Comparativo de desempenho do método de *Grid Search*



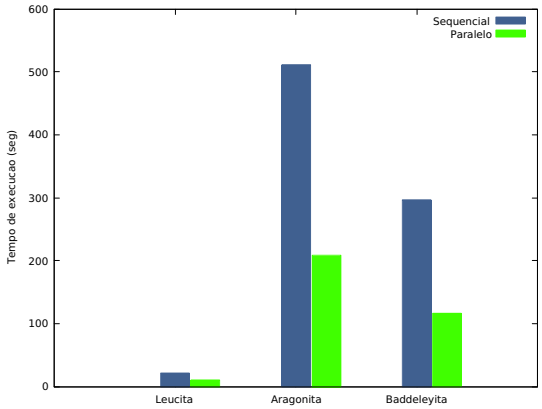
4.4.1 Comparativo de desempenho com os outros sistemas cristalinos

Nesta seção será apresentado um comparativo de desempenho entre as versões sequencial e paralela utilizando dados de difração que não são encontrados nos exemplos do McMaille. De fato, esses foram obtidos do *RRUFF Project* (LAFUENTE et al., 2015), que é um repositório de dados de difração de raio X. Para a realização dos testes, foram utilizadas as difrações de raio X de 3 minerais: a Leucita, que possui um sistema cristalino tetragonal; a Aragonita, que possui um sistema cristalino ortorrômbico; e a Baddeleyita,

que tem um sistema cristalino monoclínico. As difrações utilizadas podem ser encontradas em *<http://rruff.info/>*.

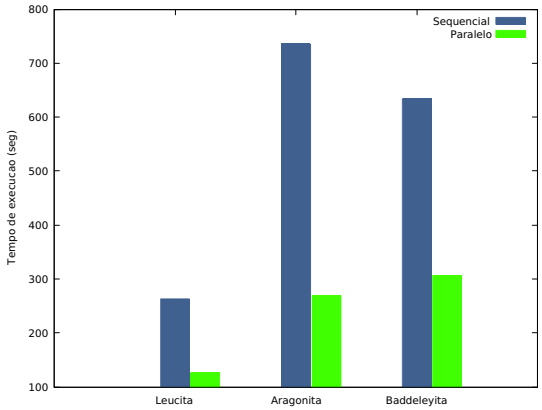
Na Figura 4.7 tem-se um comparativo dos tempos de execução das versões sequencial e paralela do método de Monte Carlo. Obteve-se um ganho de desempenho para os três, atingindo 2,53 de *speedup* e 61,4% de eficiência para a Baddeleyita.

Figura 4.7 – Desempenho do método de Monte Carlo



Na Figura 4.8 é apresentado o desempenho para o método *Grid Search* das duas versões. O melhor desempenho da versão paralela foi obtido com a Aragonita, com 2,74 de *speedup* e 68,4% de eficiência.

Figura 4.8 – Desempenho do método *Grid Search*

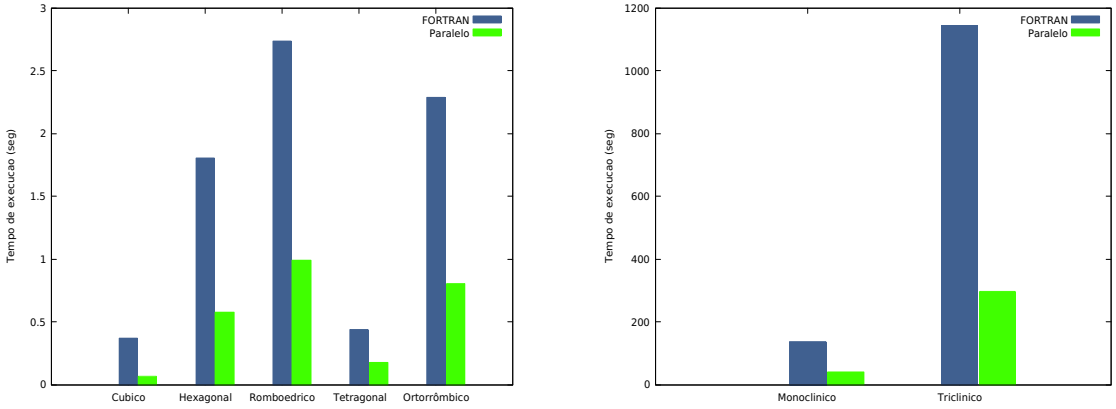


4.5 COMPARATIVO DE DESEMPENHO ENTRE A VERSÃO EM FORTRAN E A VERSÃO PARALELA

Nesta seção será apresentado um comparativo de desempenho entre a versão do McMaille desenvolvido na linguagem de programação FORTRAN e a versão paralela e otimizada em linguagem de programação C.

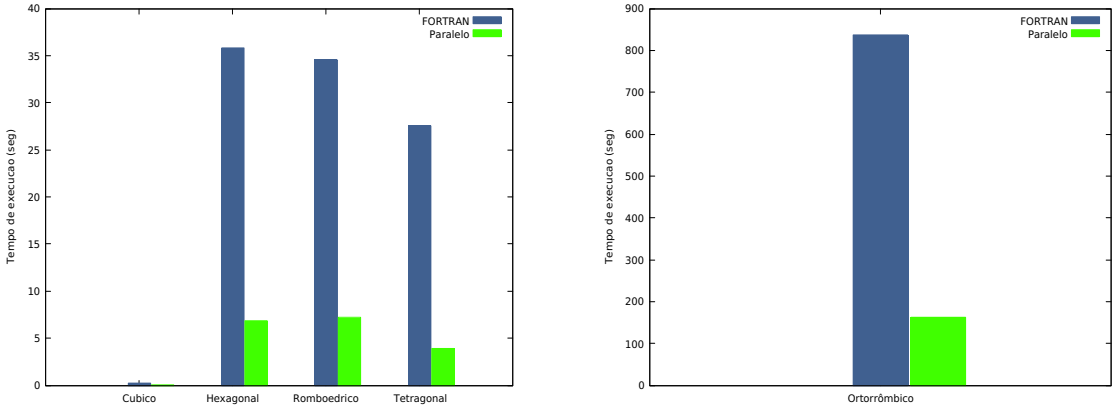
Na Figura 4.9 tem-se um comparativo dos tempos de execução das versões em FORTRAN e paralela do método de Monte Carlo. Obteve-se um ganho de desempenho de até 4 vezes na versão paralela em relação a versão em linguagem FORTRAN.

Figura 4.9 – Comparativo de desempenho do método de Monte Carlo



Na Figura 4.10 é apresentado o desempenho da versão em FORTRAN e da versão paralela do método *Grid Search*. Percebe-se que para este método se obteve um tempo de execução inferior na versão paralela, chegando a atingir um tempo 7 vezes menor que o tempo da versão em FORTRAN.

Figura 4.10 – Comparativo de desempenho do método de *Grid Search*



5 CONCLUSÃO

O objetivo deste trabalho foi desenvolver uma versão na linguagem C para o software McMaille e desenvolver uma versão paralela do mesmo. Para a conversão do software McMaille para a linguagem C, foi utilizado o conversor *f2c* (FELDMAN et al., 1995), sendo necessário reescrever ou implementar algumas funções. Foram realizados testes com o objetivo de verificar a corretude do programa após a conversão, bem como testes comparando o desempenho entre as duas versões. Os testes realizados mostraram que a conversão do código fonte não introduziu erros no software. Também se verificou que a versão em linguagem C apresentou um tempo de execução inferior ao da versão em FORTRAN. Destaca-se que a versão em linguagem C foi compilada utilizando diretivas de otimização *O3* do compilador gcc (STALLMAN, 2018).

Após, foi feita a perfilação do código convertido utilizando o software *gprof* (FENLASON; STALLMAN, 2009). A partir da perfilação do código fonte, observou-se que o maior custo computacional se encontra na função *calcul1*, que é responsável pelo cálculo dos picos de intensidade da célula unitária hipotética e o cálculo da diferença entre esses picos e os inseridos pelo usuário. Observou-se que um dos motivos deste alto custo computacional devia-se ao cálculo da função arco seno. Com isso, os valores do arco seno foram pré-calculados no início do programa, resultando em um ganho de até 32% de desempenho.

Após, foi desenvolvida uma versão paralela do McMaille utilizando a biblioteca OpenMP (CHAPMAN; JOST; PAS, 2008). A paralelização se concentrou nos métodos de Monte Carlo e *Grid Search*, que são utilizados para encontrar a célula unitária. Após a paralelização, foram realizados testes para verificar a corretude do programa, sendo constatado que nenhum erro foi introduzido.

Por fim, foram apresentados os resultados de testes de desempenho realizados entre a versão sequencial e paralela do McMaille. Destaca-se que para os testes foram utilizadas 4 *threads*. Com isso, observou-se uma diminuição no tempo de execução da versão paralela para os dois métodos utilizados pelo programa, chegando a um *speedup* de 2,3 e uma eficiência de 58% em sistemas cristalinos mais complexos.

5.1 TRABALHOS FUTUROS

Este trabalho cria algumas opções para trabalhos futuros. Entre eles, podemos citar:

- Implementação do método de *Grid Search* para o sistema cristalino triclínico, bem como sua paralelização.

- Testes com dados de difração cuja estrutura cristalina ainda não seja conhecida.

REFERÊNCIAS

ASKELAND, D.; FULAY, P. **Essentials of Materials Science & Engineering**. [S.l.]: Cengage Learning, 2008. ISBN 9780495244462.

BAIL, A. L. Monte carlo indexing with mcmaille. **Powder Diffraction**, Cambridge University Press, v. 19, n. 3, p. 249–254, 2004.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **Journal of Machine Learning Research**, v. 13, n. Feb, p. 281–305, 2012.

BLEICHER, L.; SASAKI, J. M. Introdução à difração de raios-x em cristais. **Universidade Federal do Ceará**, p. 1–20, 2000.

CALLISTER, W. D. **Materials Science and Engineering: An Introduction, 7th Edition**. [S.l.]: John Wiley & Sons, Limited, 2007. ISBN 9780470120323.

CHAPMAN, B.; JOST, G.; PAS, R. V. D. **Using OpenMP: portable shared memory parallel programming**. [S.l.]: MIT press, 2008. v. 10.

CULLITY, B.; STOCK, S. **Elements of X-ray Diffraction**. [S.l.]: Prentice Hall, 2001. (Pearson education). ISBN 9780201610918.

FELDMAN, S. I. et al. A fortran-to-c converter. 1995. Disponível em: <http://www.netlib.org/f2c/f2c.pdf>. Acesso em: 27 ago. 2017.

FENLASON, J.; STALLMAN, R. **GNU gprof**: The gnu profiler. [S.l.], 2009. Disponível em: <http://www.ecoscentric.com/ecospro/doc/html/gnutools/share/doc/gprof.pdf>. Acesso em: 27 ago. 2017.

GALLIVAN, K. et al. **Parallel Algorithms for Matrix Computations**. [S.l.]: Society for Industrial and Applied Mathematics, 1990. (Other Titles in Applied Mathematics). ISBN 9780898712605.

INTEL. **Intel® Fortran Compiler 17.0 Developer Guide and Reference**. [S.l.], 2016. Disponível em: <https://software.intel.com/sites/default/files/managed/93/88/PDF_Fortran_Compiler_UG_17_0.pdf>. Acesso em: 29 ago. 2017.

KOSEVICH, A. **The Crystal Lattice: Phonons, Solitons, Dislocations, Superlattices**. [S.l.]: Wiley, 2006. ISBN 9783527606931.

LAFUENTE, B. et al. Highlights in mineralogical crystallography. **W. De Gruyter, Berlin**, p. 1–30, 2015.

LAUGIER, J.; BOCHU, B. Lmgp-suite of programs for the interpretation of x-ray experiments. **ENSP/Laboratoire des Matériaux et du Génie Physique, BP**, aint Martin d'Hères France, v. 46, p. 38–42, 2004.

LAWRENCE, N. **Compaq Visual Fortran: A guide to creating windows applications**. [S.l.]: Digital Press, 2002.

METROPOLIS, N.; ULAM, S. The monte carlo method. **Journal of the American statistical association**, Taylor & Francis Group, v. 44, n. 247, p. 335–341, 1949.

MOONEY, C. **Monte Carlo Simulation**. [S.l.]: SAGE Publications, 1997. (Monte Carlo Simulation, Nº 116). ISBN 9780803959439.

NEGRI, A. et al. **Multi thread programming**. 2001.

PACHECO, P. S. **Parallel programming with MPI**. [S.l.]: Morgan Kaufmann, 1997.

PATTERSON, J.; BAILEY, B. **Solid-State Physics: Introduction to the Theory**. [S.l.]: Springer Berlin Heidelberg, 2007. (SpringerLink: Springer e-Books). ISBN 9783540349334.

PLLANA, S.; XHAFA, F. **Programming Multicore and Many-core Computing Systems**. [S.l.]: Wiley, 2017. (Wiley Series on Parallel and Distributed Computing). ISBN 9780470936900.

ROISNEL, T.; RODRÍQUEZ-CARVAJAL, J. Winplotr: a windows tool for powder diffraction pattern analysis. In: TRANSTEC PUBLICATIONS; 1999. **Materials Science Forum**. [S.l.], 2001. v. 378, n. 1, p. 118–123.

SCHILDT, H. **C: The Complete Reference**. [S.l.]: McGraw-Hill Education (India) Pvt Limited, 2000. ISBN 9780070411838.

SHIRLEY, R. The crysfire system for automatic powder indexing: user's manual. **The Lattice Press**, v. 41, p. 931–932, 2000.

STALLMAN, R. M. **Using the GNU Compiler Collection**: For gcc version 8.1.0. [S.l.], 2018. 114–172 p. Disponível em: <https://gcc.gnu.org/onlinedocs/gcc-8.1.0/gcc.pdf>. Acesso em: 12 mai. 2018.

SZWACKI, N.; SZWACKA, T. **Basic Elements of Crystallography, Second Edition**. [S.l.]: Pan Stanford Publishing, 2016. ISBN 9789814613583.

TISZA, M. **Physical Metallurgy for Engineers**. [S.l.]: ASM International, 2001. ISBN 9781615032419.

APÊNDICE A – PERFILAÇÃO DO MÉTODO DE MONTE CARLO

Trecho de Código A.1 – Saída *gprof* para perfilamento do método de Monte Carlo

```
index % time    self  children    called     name
                                                 <spontaneous>
[1]     98.8   93.41    0.00                 calcul1 [1]
-----------------------------------------------
                                                 <spontaneous>
[2]      0.8    0.17    0.62                 dcell [2]
                0.62    0.00 23862305/23862305     trcl [3]
-----------------------------------------------
                0.62    0.00 23862305/23862305     dcell [2]
[3]      0.7    0.62    0.00 23862305          trcl [3]
-----------------------------------------------
                                                 <spontaneous>
[4]      0.3    0.32    0.00                 randi [4]
-----------------------------------------------
                                                 <spontaneous>
[5]      0.0    0.02    0.00                 calcul2 [5]
-----------------------------------------------
                                                 <spontaneous>
[6]      0.0    0.02    0.00                 killk [6]
-----------------------------------------------
                0.00    0.00    2800/2800        mcrnl [11]
[7]      0.0    0.00    0.00    2800         calc [7]
-----------------------------------------------
                0.00    0.00    2800/2800        mcrnl [11]
[8]      0.0    0.00    0.00    2800         matinv [8]
-----------------------------------------------
                0.00    0.00       4/282         celref [19]
                0.00    0.00     278/282         celref2 [20]
[9]      0.0    0.00    0.00     282          inver [9]
-----------------------------------------------
                0.00    0.00       2/280         celref [19]
                0.00    0.00     278/280         celref2 [20]
[10]     0.0    0.00    0.00     280          fonc [10]
-----------------------------------------------
                0.00    0.00       2/280         celref [19]
                0.00    0.00     278/280         celref2 [20]
[11]     0.0    0.00    0.00     280          mcrnl [11]
                0.00    0.00    2800/2800        calc [7]
                0.00    0.00    2800/2800        matinv [8]
-----------------------------------------------
                0.00    0.00       1/1           createImpFile [21]
[12]     0.0    0.00    0.00       1          insertHeader [12]
```

APÊNDICE B – PERFILAÇÃO DO MÉTODO *GRID SEARCH*

Trecho de Código B.1 – Saída *gprof* para perfilamento do método de *Grid Search*

```
[1]     99.5   31.64    0.00                  calcul1 [1]
-----------------------------------------------
                                                <spontaneous>
[2]      0.5    0.07    0.08                  dcell [2]
                0.08    0.00 11877196/11877196     trcl [3]
-----------------------------------------------
                0.08    0.00 11877196/11877196     dcell [2]
[3]      0.3    0.08    0.00 11877196          trcl [3]
-----------------------------------------------
                                                <spontaneous>
[4]      0.0    0.01    0.00                  brav [4]
-----------------------------------------------
                                                <spontaneous>
[5]      0.0    0.01    0.00                  supcel [5]
-----------------------------------------------
                0.00    0.00       20/20          mcrnl [10]
[6]      0.0    0.00    0.00       20          calc [6]
-----------------------------------------------
                0.00    0.00       20/20          mcrnl [10]
[7]      0.0    0.00    0.00       20          matinv [7]
-----------------------------------------------
                0.00    0.00        4/4           celref [18]
[8]      0.0    0.00    0.00        4          inver [8]
-----------------------------------------------
                0.00    0.00        2/2           celref [18]
[9]      0.0    0.00    0.00        2          fonc [9]
-----------------------------------------------
                0.00    0.00        2/2           celref [18]
[10]     0.0    0.00    0.00        2          mcrnl [10]
                0.00    0.00       20/20          calc [6]
                0.00    0.00       20/20          matinv [7]
-----------------------------------------------
                0.00    0.00        1/1           createImpFile [20]
[11]     0.0    0.00    0.00        1          insertHeader [11]
-----------------------------------------------
```

APÊNDICE C – COMPARATIVO DE DESEMPENHO DA VERSÃO SEQUENCIAL E PARALELA

Tabela C.1 – Tempos de execução

	Tempo Médio (seg)			
	Sequencial		Paralelo	
Sistema Cristalino	Monte Carlo	*Grid Search*	Monte Carlo	*Grid Search*
Cúbico	0,092	0,018	0,064	0,021
Hexagonal	0,841	15,352	0,578	6,814
Romboédrico	1,394	13,760	0,991	7,195
Tetragonal	0,260	7,518	0,177	3,918
Ortorrômbico	1,299	370,137	0,801	162,319
Monoclínico	69,651		40,560	
Triclínico	692,499	-	295,816	-