ALGORITMOS GENÉTICOS ACOPLADOS A CÁLCULOS QUÂNTICOS PARA REFINAMENTO DE CAMPO DE FORÇA

RODRIGO BENTES KATO

ALGORITMOS GENÉTICOS ACOPLADOS A CÁLCULOS QUÂNTICOS PARA REFINAMENTO DE CAMPO DE FORÇA

Tese apresentada ao Programa de Pós-Graduação em Bioinformática do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Bioinformática.

Orientador: Jadson Cláudio Belchior
Coorientadora: Gisele Lobo Pappa

Belo Horizonte
Setembro de 2014

Kato, Rodrigo B

D1234p Algoritmos Genéticos acoplados a Cálculos Quânticos para Refinamento de Campo de Força / Rodrigo Bentes Kato. — Belo Horizonte, 2014
xxiii, 478 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais
Orientador: Jadson Cláudio Belchior.
Coorientadora: Gisele Lobo Pappa.

1. Algoritmo Genético.
2. Campo de Força.
3. Cálculos Quânticos.
I. Tï£¡tulo.

CDU 519.6*82.10

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Se a imagem da folha de aprovação precisar ser ajustada, use:
`approval=[`*ajuste*`][`*escala*`]{`*nome do arquivo*`}`
onde *ajuste* ï£¡ uma distância para deslocar a imagem para baixo
e *escala* é um fator de escala para a imagem. Por exemplo:
`approval=[-2cm][0.9]{`*nome do arquivo*`}`
desloca a imagem 2cm para cima e a escala em 90%.

Agradecimentos

Gostaria de agradecer a todas as pessoas que estiveram ao meu lado, que me auxiliam e ajudam no meu crescimento diário. Desta forma, agradeço à minha família que de forma direta ou indiretamente esteve presente em minha vida, incentivando-me em meus estudos e possibilitando que eu chegasse até aqui. Em especial a minha mãe Marly, ao meu pai Otávio, a minha irmã Rayla, aos meus irmãos Rogério e Ricardo, e sem esquecer é claro, de minhas avós Eneide e Misako. E meus avôs Moacyr e Arlindo (*in memorian*).

Agradeço a minha noiva Camila Alves pelo incentivo, dedicação, compreensão e apoio, em mais uma etapa de minha vida.

Agradeço também ao Laboratório de Química (Departamento de Química, UFMG) e ao Laboratório de Bioinformática e Sistemas (Departamento de Ciências da Computação, UFMG), Raquel, Valdete, Sabrina, Elisa, Nilma, Sandro, Douglas, Wellisson, Pedro, Laerte, João, Alexandre, Ferré, Mateus, Frederico, Breno.

Aos meus companheiros de doutorado e grandes amigos Ohashi, Artur Gustavo, Fábio, Vinícius, Rodrigo, Mariana, Mainá, Michele, Kátia, Tetsu, Izinara, Moema. Aos mestres da Universidade Federal do Minas Gerais (UFMG), Universidade Federal do Pará (UFPA) e do Centro Universitário do Pará (CESUPA), pelos ensinamentos.

Agradeço ao Programa de Doutorado em Bioinformática pela estrutura, Natália, Ana e Sheila, aos docentes pelos ensinamentos; à FUNDEP, FAPEMIG, CAPES e CNPq pelos fomentos diretos e indiretos ao grupo de pesquisa, ao trabalho e à minha formação.

Gostaria de fazer um agradecimento especial ao Prof. Marcelo Santoro (*in memorian*) pela grande contribuição, de maneira direta, dada no desenvolvimento deste trabalho até próximo de sua reta final, e de maneira indireta, pois depois do seu falecimento tive que manter o foco, apesar da dor, e continuar o trabalho para não deixar suas idéias serem perdidas.

Agradeço toda a atenção a mim dispensada pelo meu orientador, Jadson Cláudio Belchior, e minha co-orientadora, Gisele Lobo Pappa, e também pelo incentivo e ami-

zade. Espero que todos continuem a acreditar em nós alunos, mostrando que somos capazes.

"Ele (Deus) é o dono de tudo. Devo a Ele a oportunidade que tive de chegar aonde cheguei. Muitas pessoas têm essa capacidade, mas não têm a oportunidade. Ele a deu pra mim, não sei por quê. Só sei que não posso desperdiçá-la."

(Ayrton Senna)

Resumo

Neste trabalho é apresentado a aplicação do Algoritmo Genético (AG) como uma metodologia para refinar os parâmetros do campo de força para determinar as energias dos ácidos nucleicos (adenina, guanina, citosina e uracila). São realizados cálculos quânticos para esses ácidos nucleicos, e os seus resultados (energias e estruturas) vão servir como dados de referência para o AG. Para essa abordagem teste (AG acoplado com cálculos quânticos) nós trabalhamos, mais especificamente, com a reparametrização dos termos torcionais e eletrostático do campo de força. Foi realizada uma comparação usando o RMSE (*Root Means Squared Error*) com resultados recentemente publicados para descrever ácidos nucleicos e mostraram uma melhora, em média, de 31%. Finalmente, o termo potencial reparametrizado foi utilizado para determinar a estrutura de uma molécula de PDB (1r4h) que não foi utilizado para o processo de parametrização. A estrutura foi melhorada em 81% comparada com os resultados publicados anteriomente.

Abstract

This work has dealt with the use of Genetic Algorithms (GA) as a method to refine force field parameters in order to determine nucleic acids energies (adenine, guanine, cytosine and uracil). Quantum calculations are carried out for these nucleic acids that are taken as reference data. In this particular study torsion and electrostatic energies are reparametrized in order to test the proposed approach, i.e, GA coupled with quantum mechanics calculations. Overall, RMSE comparison with recent published results for describing nucleic acids energies showed an improvement, on average, of 31%. Finally, the new reparametrized potential energy was used to determine energy and structure of a PDB molecule (1r4h) that was not used to parametrization process. The structure was improved about 81% compared with previous published results.

Keywords: Genetic Algorithm, Force Field, Quantum Calculations.

Lista de Figuras

Lista de Tabelas

Lista de expressões latinas

- *a priori*: admitido como evidente, independe da experiência
- *in memoriam*: em memória de
- *ab initio*: desde o início

Lista de abreviaturas

- DNA Ácido Desoxirribonucleico
- RNA Ácido Ribonucléico
- RNAi Ácido Ribonucléico de interferêcia
- HPV Papilomavírus Humano
- HIV Vírus da Imunodeficiência Humana
- AG Algoritmo Genético
- PDB (Protein Data Bank) Banco de dados de Proteínas
- RMSD (Root Mean Deviation Square) Raiz do Desvio Médio Quadrático
- QM (Quantum Mechanic) Mecânica Quântica
- DFT (Density Functional Theory) Teoria do Funcional de Densidade
- CE Computação Evolutiva
- DM Dinâmica Molecular
- MM Mecânica Molecular
- OLPS (Optimized Potentials for Liquid Simulations) Potenciais Otimizados por Simulações em Líquido
- CENAPAD Centro Regional do Sistema Nacional de Processamento de Alto Desempenho
- LCC Laboratório de Computação Científica

Sumário

Capítulo 1

Introdução

Nestes últimos anos, o uso de computadores para simulações tornou-se de grande importância para o desenvolvimento das diversas áreas de conhecimento. Particularmente em simulações moleculares essas simulações utilizam os princípios gerais das dinâmicas clássica e quântica, dando origem à Dinâmica Molecular (DM) [1]. Na Dinâmica Molecular os átomos são capazes de permanecer em equilíbrio mecânico por causa das interações entre eles, que mesmo recebendo uma perturbação externa ele volta a encontrar um novo equilíbrio [2].

O crescimento desta área de pesquisa ocorre devido ao fato de que as simulações computacionais permitem diminuir a interface entre a teoria e o experimento. A literatura especializada nos mostra que a criação e a proposição de novos materiais têm crescido muito após o surgimento das técnicas de modelagem computacional [3]. O uso destas técnicas estimula o aparecimento de novos materiais, fármacos e estudos de estruturas biomoleculares [3, 4].

Existem diversos modelos teóricos aplicados para estudar as propriedades e estruturas de compostos, por exemplo, dinâmica molecular, mecânica molecular, mecânica quântica, entre outros. Um motivo importante para o desenvolvimento desses modelos é a limitação computacional que nos tempos atuais tenta resolver problemas envolvendo alto número de graus de liberdade (por ex., coordenadas atômicas, nucleares e eletrônicas) que impossibilita o tratamento destes sistemas. Técnicas de dinâmica molecular são muito conhecidas e fornecem dados precisos para análise de propriedades de sistemas [5]. Vários programas que realizam simulações de DM usam campos de força baseados em mecânica molecular. No entanto, a análise referente ao sistema depende da precisão dos parâmetros do campo de força para descrever o comportamento da molécula. Por isso, devemos ter uma atenção maior nessa parametrização.

Para comprovar a teoria de interações importantes somos obrigados a refletir so-

bre os resultados experimentais e previsão de estruturas tridimensionais de moléculas de RNA [6]. Por esse motivo este trabalho tem como principal objetivo refinar parâmetros de uma determinada função, chamado campo de força (mais tarde será detalhado). Nós iremos trabalhar com campo de força baseado em ácidos nucleicos, que são compostos químicos contendo: ácido fosfórico, açúcares, bases purínicas e pirimidínicas (Figura 1.1). Na natureza existem dois tipos: o ácido desoxirribonucleico (DNA) e o ácido ribonucleico (RNA).

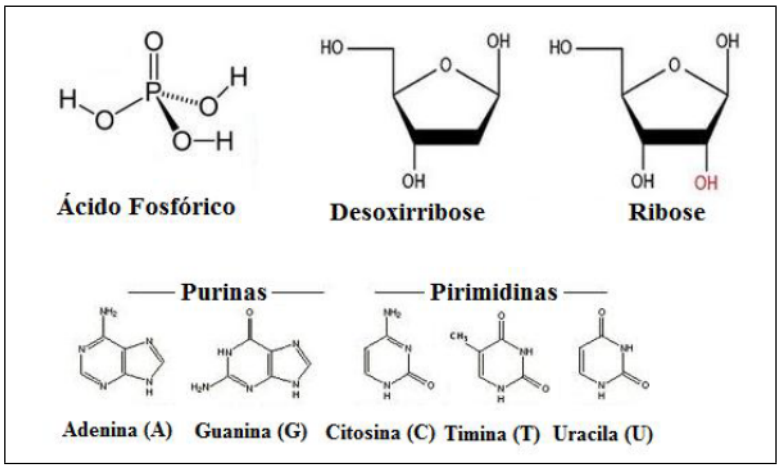


Figura 1.1. Compostos químicos dos ácidos nucleicos.

O foco deste trabalho é o RNA que é um polímero linear, não ramificado composto de quatro unidades de nucleotídeos: Adenosina (A), Citosina (C), Guanosina (G) e Uridina (U). Os ácidos nucleicos são macromoléculas que desempenham um papel vital na vida celular. O primeiro e mais essencial é o ácido desoxirribonucleico (DNA) que tem como função preservar a informação genética na célula. Por outro lado, os ácidos ribonucleicos são responsáveis por uma grande variedade de funções e ainda não foram totalmente explorados [7].

Os RNAs estruturais apresentam uma grande variedade de conformações. Elas são necessárias para garantir o desempenho de suas diversas funções além de simplesmente levar informação genética [8]. Algumas das principais funções do RNA são:

- Participam na síntese de proteínas;
- Catalisam reações [9, 10, 11];
- Regulam a expressão gênica [12];
- Além disso, existem muitas sequências de RNA que são transcritos em genoma, mas suas funções ainda não são conhecidas [13].

Alguns RNAs são considerados potenciais alvos terapêuticos, pois estão incluídos em estruturas de vírus humanos, como: o papilomavírus humano (HPV), o vírus da imunodeficiência humana (HIV), a varíola e os vírus da gripe. Outro fator importante é, por exemplo, o RNA de interferência (RNAi) que possui uma grande importância na regulação da expressão genética, pois participa da defesa do organismo contra infecções virais.

O uso do RNAi como uma ferramenta de terapia gênica tem sido foco de muitas pesquisas, especialmente em infecções virais, câncer, desordens genéticas herdadas, doenças cardiovasculares e mesmo em doenças reumáticas [14]. Portanto, a compreensão do mecanismo de enovelamento do RNA é essencial para o entendimento das várias funções que a molécula executa [7]. Entretanto, a estrutura da molécula do RNA pode permitir reconhecer outras através da ligação seletiva com ele, e até mesmo, em certos casos, para catalisar alterações químicas nas moléculas a qual estão ligados [15].

A utilização da dinâmica molecular para estudos de proteínas e ácidos nucleicos vem sendo amplamente usada, há mais de 10 anos, para nos ajudar a investigar melhor a estrutura e dinâmica sob uma ampla variedade de condições dessas moléculas [16]. Uma das limitações da dinâmica é a precisão no campo de força que como consequência a representação dos sistemas apresenta alguns erros. Trabalhos recentes mostram que ainda existem alguns erros em campos de força para ácidos nucleicos [17, 18, 19]. A fim de resolver esse problema a literatura está trabalhando com cálculos quânticos para ajudar no refinamento dos parâmetros dos campos de força [17, 18, 19, 20, 21, 22, 23].

Métodos Quânticos A maneira mais eficaz para se obter a energia de um sistema é usando cálculos quânticos, que possuem vários métodos com diferentes níveis de precisão [24, 25]. Apesar da complexidade na implementação e utilização dos métodos quânticos, esses cálculos são de fundamental importância na parametrização dos campos de força clássicos. Esta parametrização é feita aplicando os métodos quânticos em pequenos sistemas moleculares no sentido de caracterizar propriedades moleculares, as quais serão estendidas aos sistemas maiores [5].

O uso desses cálculos para estudar as propriedades moleculares, com base em métodos quânticos, tais como Hartree-Fock, Funcional de Densidade, entre outros, têm certos impedimentos computacionais. Nestes métodos, a precisão é inversamente proporcional ao tamanho do sistema [7]. O problema ocorre quando se possui um alto número de graus de liberdade nas coordenadas nucleares e eletrônicas que tornaria o tratamento destes sistemas usando cálculos *ab initio* muito demorados. O tempo computacional envolvido em uma abordagem puramente quântica destes problemas são absolutamente proibitivos. Isto impossibilita, na maioria dos casos, o estudo de

propriedades físico-química e termodinâmicas de sistemas moleculares complexos como, por exemplo:

- Estudo do processo de *docking* de fármacos em proteínas;
- Estudo do enovelamento de proteínas (*protein folding*);
- Desenvolvimento e proposição de novos fármacos e estruturas de biomoléculas.

Particularmente, para sistemas grandes, tais como RNA ou proteínas a tarefa da construção do campo de força é ainda mais difícil devido as várias razões, por exemplo, um grande número de elétrons, devido ao tamanho do sistema, bem como vários termos que contribuem para o total de energia. Portanto, o desenvolvimento de modelos empíricos para descrever o sistema é uma alternativa eficiente para proporcionar uma forma de construção de campos de força.

Uma alternativa para solucionar esse problema é utilizar moléculas básicas de sistemas complexos, como foi realizado nas referências [17] e [19]. Esses cálculos servirão de referência para os diversos métodos de otimização. Dessa forma temos uma maneira eficaz para refinar os parâmetros dos campos de força já existentes ou desenvolver outros modelos empíricos (campos de força) para descrever os sistemas em questão. E assim, a partir de uma estrutura inicial e fazendo uso de um campo de força, as propriedades, tais como estabilidade e dinâmica podem ser previstas para as configurações de RNA complicadas [6].

Mecânica Molecular O procedimento realizado na Mecânica Molecular (MM) é geralmente baseado em otimização de funções matemáticas com várias contribuições físicas, tais como os termos de atração e repulsão (forças intermoleculares) e outros para determinar a contribuição de energia interna (forças intermoleculares).

A maior vantagem da mecânica molecular em comparação aos métodos quânticos é, sem dúvida, sua velocidade. No modelo teórico denominado Mecânica Molecular as ligações químicas são representadas por potenciais harmônicos, potencial de Lennard-Jones e de Coulomb. O conjunto desses potenciais é denominado campo de força. Neste caso, é necessário encontrar um conjunto de parâmetros para que o modelo de energia potencial possa descrever adequadamente as interações do sistema com suficiente precisão.

Estes métodos não levam em consideração os elétrons do sistema, e sim as interações entre núcleos. Apesar dos efeitos eletrônicos já estarem implicitamente incluídos nos campos de força através de parametrizações, e assim reduzindo o custo computacional.

O uso de campos de força pode ser comparado com vários tipos de resultados experimentais [26]. Baseado nisso, se o modelamento de um campo de força for fisicamente adequado e bem parametrizado, a simulação da Dinâmica Molecular (DM) pode, em princípio, nos permitir compreender a estrutura energética e a dinâmica dos ácidos nucleicos em nível atômico [27] e determinar propriedades de sistemas maiores como macromoléculas.

Os campos de força mais usados em simulações de Dinâmica Molecular para ácidos nucleicos, proteínas e carboidratos são: Amber, CHARMm e OPLS, respectivamente [28]. O uso da Dinâmica Molecular com o auxílio do campo de força para estudo de estruturas em nível atômico é uma área bem estudada [29, 30]. Porém, esta ainda possui algumas limitações; por exemplo, quando temos sistemas muito grandes com muitas dimensões, o poder computacional requerido cresce criando problemas proibitivos. Outro problema da dinâmica é quando necessitamos obter uma molécula com sua representação exata, muito próximo da natureza. Essa limitação pode ser resolvida melhorando a qualidade do campo de força, tema abordado neste trabalho.

Métodos de otimização são muito utilizados para otimização de campos de força. Existem diversos tipos de métodos, alguns baseados em método gradiente, tal como, quasi-Newton [31] e Levenberg-Maquardt [31, 32]. Por exemplo, Yildirim *et. al.* [19] usaram o método *steep descent* e os resultados mostraram-se bem acurados comparados com os resultados quânticos. Outro método usado recentemente é o heurístico que tem sido usado e testado para refinar os parâmetros dos campos de força. Por exemplo, Sakae and Okamoto [33] aplicaram o método de Monte Carlo e *Simulated Annealing* para otimizar os parâmetros do campo de força para proteínas.

Algoritmos Genéticos Uma abordagem alternativa para o processo de otimização do campo de força está sendo usada em trabalhos recentes para refinamento de parâmetros para campos de força [34]. Similarmente, algoritmos genéticos (AG) acoplados a cálculos quânticos têm sido aplicados para se determinar energias e estruturas, porém em sistemas pequenos com destaque em cluster de sódio-potássio [35], cobre-ouro [36]. O uso de AGs aplicados para otimização de estruturas (atômicas e moleculares) e energias são comuns [37, 38].

Com o objetivo de melhorar a parametrização na torção glicosídica, pois é umas das ligações mais importantes dos ácidos nucleicos [17], este trabalho propõe uma solução heurística baseada em algoritmo genético [39, 40, 41] para refinar os parâmetros de torção da ligação glicosídica do campo de força para as energias de torções baseadas em princípios de evolução de Darwin e sobrevivência do mais apto, detalhado no Capítulo 3.1. A fim de fornecer dados precisos como referência (já que não temos

os dados experimentais) para testar a presente abordagem, o AG irá utilizar cálculos quânticos de ácidos nucleicos (Adenina, Guanina, Citosina e Uracila). Este processo é iterativo, terminando apenas quando atinge seu critério de parada. Como um método de validação dos nossos resultados obtidos será feita uma comparação entre os nossos novos parâmetros encontrados pelo AG com os resultados recém publicados e com dado do *Protein Data Bank* (PDB), utilizando como medida de comparação o RMSD (que será detalhado mais tarde).

Uma das vantagens de utilizar os algoritmos genéticos é que eles são muito utilizados na literatura como otimizadores de função e para este caso podemos tratar o problema do refinamento dos parâmetros como um problema de otimização da função de energia. Essa função costuma ser multimodal (contém vários mínimos locais) que, combinados com o enorme tamanho do espaço de busca conformacional, dificulta o uso de técnicas de otimização clássicas [3]. A natureza probabilística de algoritmos genéticos pode ser explorada para escapar de mínimos locais em um esforço para encontrar o mínimo global.

No algoritmo genético os possíveis mínimos são obtidos através de comparações das energias moleculares de duas ou mais combinações de parâmetros do campo de força, gerados de maneira estocástica. Isto é, a rotina do AG gera aleatoriamente os novos parâmetros e calcula a energia molecular. A evolução ocorre através da troca de materiais genéticos entre as possíveis soluções (esse processo será detalhado mais tarde na seção 3.1). O processo termina quando a convergência é atingida para um dado critério escolhido *a priori*.

Através de pesquisas feitas nas bases de dados da literatura (*web of science*) percebemos que o uso de AGs para refinar os parâmetros dos campos de força é raro. Por isso, este trabalho irá estudar o seu comportamento, a fim de reparametrizar o campo de força Amber (ff99χOL). Deste modo, nós iremos testar esse procedimento considerando apenas o potencial de torção.

1.1 Objetivo

O presente trabalho tem como foco principal o desenvolvimento de uma metodologia baseada em algoritmo genético acoplado com métodos de cálculos quânticos que possa ser usada como ferramenta para a parametrização de qualquer campo de força baseado em ácidos nucleicos de RNAs.

1.1.1 Objetivos específicos

- Efetuar revisão bibliográfica das áreas de computação evolucionária, e parametrização de campos de força;
- Isolar cada nucleotídeo do RNA (vácuo);
- Estabelecer as combinações de nucleotídeos de RNA que serão testadas;
- Refinar as diversas estruturas de cada nucleotídeo que serão utilizadas usando cálculos quânticos;
- Desenvolver um AG para o refinamento dos parâmetros do termo de torção do campo de força escolhido;
- Validar a nova parametrização usando dados do PDB;
- Avaliar a relevância da nova parametrização.

O trabalho realizado obteve inovações na parametrização do campo de força Amber, cuja possíveis contribuições serão apresentados a seguir.

1.2 Contribuições

Os resultados que serão apresentados possibilitarão descobertas, cujas contribuições estão pontuadas a seguir.

Determinação de propriedades dos RNAs Obtenção de diferenças de energias conformacionais e de barreiras de energias rotacionais, estudo de estabilidade relativa de isômeros, pressão de vapor, cálculo de propriedades termodinâmicas de gases e densidade.

Determinação ou Refinamento de estruturas dos RNAs Com uma boa parametrização espera-se conseguir obter estruturas bem próximas dos dados experimentais, facilitando assim o estudo dos sistemas e desenvolvimentos de novas ligas, compostos e materiais.

Determinação de funções Obtenção ou determinação de novas funções de moléculas de RNA através de comparação de estruturas já conhecidas.

1.3 Organização do texto

Essa introdução acima teve como objetivo mostrar a importância deste trabalho tanto para o meio acadêmico quanto o empresarial. Este trabalho foi dividido em quatro capítulos da seguinte forma.

No referencial teórico contido no capítulo 2, destacamos as três principais áreas deste trabalho, dando uma breve introdução em cada tema. No primeiro tópico apresentamos ao leitor uma base sobre cálculos quânticos, dando apenas detalhes do que vamos precisar para compreender os próximos capítulos. No segundo tópico destacamos o campo de força que será usado em nossa metodologia e explicamos cada termo. Por último, são apresentados os conceitos de algoritmo genético, seu mecanismo de funcionamento e seus principais componentes e operadores genéticos.

O capítulo 4 apresenta a metodologia aplicada neste trabalho com base em artigos recentemente publicados na literatura. Nesta seção são detalhados os modelos de sistemas que foram usados para a reparametrização, como foram obtidas as variações desses modelos, de que forma foram realizados os cálculos quânticos e os de mecânica molecular. Por fim mostramos como foram encontrados os novos parâmetros usando o algoritmo genético.

No capítulo 5 temos os resultados da aplicação dessa metodologia comparando com os resultados da literatura e testes usando uma determinada molécula da literatura para validação dos resultados.

No último capítulo são apresentadas as conclusões e possíveis trabalhos futuros que ainda podem ser realizados para melhorar o refinamento dos parâmetros do campo de força desenvolvido neste trabalho.

Capítulo 2

Referencial teórico

Neste capítulo iremos apresentar os conhecimentos básicos necessários para o leitor poder entender o projeto. Nele temos diversas referências caso o leitor queira se aprofundar no assunto. O referencial teórico abrange as três áreas principais que foram usadas para a execução desse trabalho, cada uma em sua respectiva seção.

2.1 Cálculos Quânticos

No início do século XX, sugiram várias formulações sugeridas para representar a estrutura da matéria, com o objetivo de solucionar problemas físicos em menor escala, a atômica. Uma dessas possíveis abordagens é a que utiliza os termos da função de onda representada por Schrödinger. A função de Schrödinger é uma das equações mais conhecidas da Física Quântica [42] e foi a responsável por descrever corretamente o movimento de sistemas quânticos.

A existência de estados estacionários quantizados para sistemas atômicos foi um conceito fundamental para o desenvolvimento da Mecânica Quântica (QM), postulado por Niels Bohr no início do século XX.

Os métodos da Química Computacional têm como objetivo resolver a equação de Schrödinger, de maneira que descreva a dinâmica de um conjunto de M núcleos e N elétrons em sistemas moleculares [43]. No entanto, o custo da resolução exata dessa equação para sistemas arbitrários ainda é proibitivo, mas é possível conseguir boas aproximações para um conjunto limitado de sistemas. A qualidade do método necessita basicamente de duas aproximações: no operador hamiltoniano e na função de onda [44, 45].

Vários métodos de aproximação foram propostos com o objetivo de encontrar os orbitais eletrônicos para átomos com muitos elétrons. Entre eles temos os métodos

de Hartree-Fock e a Teoria do Funcional da Densidade (DFT). Esses métodos estão sempre sendo melhorados e implementados no estudo da modelagem de sistemas.

Neste trabalho daremos destaque ao DFT, que foi o método utilizado em nossa metodologia. A escolha foi feita devido ao seu grande uso na literatura, não possuir tantos erros quanto o Hartree-Fock e também não ser tão demorado quanto os outros métodos mais sofisticados.

2.1.1 Teoria do Funcional de Densidade - DFT

A precisão dos cálculos usando o método da Teoria do Funcional de Densidade (DFT), em química quântica, varia de acordo com o conjunto de funções de base usado (detalhes na seção 2.1.2). Caso o número de funções de base seja infinito, o espaço de Hilbert é inteiramente descrito e a base é considerada completa. Porém, o conjunto completo não obrigatoriamente é infinito. O uso de um conjunto de base completo nos dá condições de obter valores limites, dentro da aproximação usada, das propriedades físicas e químicas do sistema que está sendo estudado. Entretanto, na prática é impossível trabalhar o sistema com uma base infinita. Contudo, quando vamos efetuar um cálculo atômico ou molecular em química quântica, dois pontos essenciais devem ser considerados:

- O tamanho do conjunto de bases, pois dependendo do tamanho do conjunto, o tempo computacional se torna custoso e o cálculo impraticável;
- A escolha do conjunto de bases, que deve ser feita de maneira que consiga descrever de forma precisa a propriedade física ou química que se está procurada.

A Teoria do Funcional de Densidade tem como base principal a energia de um sistema eletrônico que pode ser escrito em termos da densidade eletrônica total (ρ) [46, 47]. A energia eletrônica E é um funcional de densidade eletrônica, representada como E[ρ], no sentido que para uma dada função $\rho(r)$, existe uma única energia correspondente.

O conceito do funcional de densidade para energia foi utilizado como embasamento para alguns modelos importantes, tais como o método de Thomas-Fermi (que surgiu em 1920, do trabalho de E. Fermi e L. H. Thomas) e o método de HF-Slater, surgido do trabalho de J. C. Slater em 1950. Todavia, foi após 1964 que uma prova formal foi obtida [48] para que a energia e todas as outras propriedades eletrônicas do estado fundamental fossem determinadas pela densidade eletrônica. Infelizmente, o teorema de Hohenberg-Kohn não nos mostra como o funcional de energia varia com a densidade. Ele somente afirma que tal funcional existe.

Outro passo importante no desenvolvimento da Teoria do Funcional de Densidade veio com a derivação de um conjunto de equações de um elétron, em teoria, a partir do qual se pode obter a densidade eletrônica ρ [49].

Como apresentado por Kohn e Sham, a energia E eletrônica do estado fundamental exata de um sistema de n elétrons pode ser escrita como:

$$E = -\frac{h^2}{2m_e}\sum_{i=1}^{n}\int \psi_i^*(r_1)\nabla_1^2\psi_i(r_1)dr_1 - \sum_{I=1}^{N}\int \frac{Z_1e^2}{4\pi\epsilon_0 r_{I1}}\rho r_1 dr_1 + \frac{1}{2}\int \frac{\rho(r_1)\rho(r_2)e^2}{4\pi\epsilon_0 r_{I2}}\rho r_1 dr_2 + E_{XC}[\rho], \tag{2.1}$$

em que os orbitais espaciais de um elétron ψ_i (i = 1, 2, ..., n) são os orbitais de Kohn-Sham (KS), soluções das equações dadas abaixo. A densidade de carga ρ do estado fundamental numa posição r é dada por:

$$\rho(r) = \sum_{r=1}^{n} |\psi_i(r)|^2, \tag{2.2}$$

onde a soma sobre todos os orbitais KS ocupados é conhecida, visto que esses orbitais tenham sido calculados. O primeiro termo na Equação 2.1 representa a energia cinética dos elétrons, o segundo representa a atração núcleo-elétron com a soma sobre todos os N núcleos com índice I e número atômico Z_I, o terceiro termo representa a interação de Coulomb entre a distribuição de carga total (somada sobre todos os orbitais) em r_1 e r_2 e o quarto é a energia de troca-correlação do sistema, que é um funcional de densidade e leva em consideração todas as interações elétron-elétron não clássicas. O último é o único que não se tem conhecimento de como determiná-lo exatamente. Embora o teorema de Hohenberg-Kohn diga que E e E_{XC} devem ser funcionais da densidade eletrônica, não se conhece a forma analítica exata de E_{XC}, por isso se utilizam formas aproximadas [50].

Os orbitais KS são obtidos ao se resolver as equações de KS, que podem ser deduzidas utilizando o princípio variacional para a energia dos elétrons E[ρ] com a densidade de carga (Equação 2.2). As equações de KS para os orbitais de um elétron são dadas por:

$$\Psi_i(r_1) = -\frac{h^2}{2m_e}\nabla_1^2 - \sum\frac{Z_1e^2}{4\pi\epsilon_0 r_{I1}} + \int \frac{\rho(r_2)e^2}{4\pi\epsilon_0 r_{I2}}dr_2 + V_{XC}(r_1)\Psi_i(r_1) = \epsilon_i\Psi_i(r_1), \tag{2.3}$$

onde ϵ_i são as energias orbitais de KS e o potencial de troca-correlação (V_{XC}) é a derivada do funcional da energia de troca-correlação:

$$V_{XC}(\rho) = \frac{\delta E_{XC}[\rho]}{\delta \rho} \tag{2.4}$$

Se for conhecido E_{XC}, V_{XC} pode ser obtido. KS é que permite que se calcule a densidade ρ da Equação 2.4.

As equações KS são resolvidas de uma maneira autoconsistente. Primeiramente, se escolhe a densidade de carga ρ (na maioria das vezes é utilizada uma superposição de densidades atômicas para sistemas moleculares). Usando alguma forma aproximada (que é fixada em todas as iterações) para a dependência funcional de E_{XC} com a densidade ρ, se encontra V_{XC} como uma função de r. Soluciona-se o conjunto de equações KS para conseguir um conjunto inicial de orbitais KS. Este conjunto é usado para calcular uma melhor densidade a partir da Equação 2.2. O processo se repete até a densidade e a energia de troca-correlação não variem da antecedente.

Os orbitais podem ser expressos em termos de um conjunto de funções de base. Nesse caso, resolvem-se as equações KS para achar os coeficientes da expansão do conjunto de bases. Como acontece nos métodos Hartree-Fock, uma grande quantidade de conjuntos de funções de base pode ser utilizada e a vasta experiência conseguida em cálculos Hartree-Fock pode ser vantajosa na escolha de conjuntos de bases DFT.

2.1.2 Funções de Base

Uma função de base é o conjunto de funções utilizadas para produzir os orbitais moleculares, que são expressos como uma combinação linear de tais funções com coeficientes a serem determinados [45]. Dois tipos de funções de bases têm sido utilizadas, funções gaussianas e de Slater. Funções do tipo exponencial, chamadas de funções de Slater, foram introduzidas na década de 30 [51]. Em 1950, foram propostas na referência [52] o uso de funções gaussianas como alternativas às funções de Slater para representar orbitais atômicos. Porém, a princípio, pode-se utilizar qualquer função de onda.

As funções gaussianas são as mais usadas hoje em dia para aproximação de orbitais atômicos [53]. Uma função gaussiana não fornece uma boa representação de um orbital atômico, mas um conjunto de funções gaussianas representa bem. Na prática pega-se uma combinação linear normalizada de um conjunto de gaussianas (criando uma gaussiana contraída) [42], como:

$$\chi_r = \sum_u d_{ur} g_u, \tag{2.5}$$

onde os g_us (gaussianas primitivas) são as gaussianas normalizadas centradas no mesmo átomo, d_{ur} (coeficientes de contração) são constantes mantidas fixas durante os cálculos. Com as gaussianas contraídas reduz-se o número de coeficientes a serem otimizadas pelos métodos, reduzindo assim o tempo computacional. Funções gaussianas se tornaram populares por reduzir cálculos necessários na implementação de métodos da Química Quântica [53].

Para alguns sistemas são necessárias funções para melhorar a descrição dos orbitais, como as funções de polarização e as funções difusas. Com as funções de polarização num conjunto de bases, a descrição dos elétrons melhora, pois com estas funções consegue-se uma descrição das modificações sofridas pelos orbitais que fazem parte da ligação química. Por outro lado, as funções difusas permitem mostrar uma região maior do espaço dos orbitais ocupados. Estas funções são usadas para melhorar a representação de sistemas que possuem densidades eletrônicas mais afastadas do núcleo.

Há algum tempo atrás, devido a limitação computacional, se usavam funcionais e bases mais simples [20, 54]. Como exemplo, a base 6-31G* que significa a base 6-31G combinada com duas funções polarizadas. A 6-31G é uma contração de seis gaussianas polarizadas representando o núcleo e que o orbital de valência possui duas contrações (uma com três e outra com uma Gaussiana isolada) representando a camada de valência para cada átomo.

Hoje em dia com máquinas mais avanças podemos trabalhar com bases maiores e assim podendo tornar nossos cálculos mais precisos. Um exemplo de uma base grande usada neste trabalho é a 6-311++G(3df, 3dp), onde os sinais de mais (+) indicam a função difusa em todos os átomos; as letras nos parênteses indicam três conjuntos de cinco funções gaussianas polarizadas do tipo d, um conjunto de sete gaussianas do tipo f, e três conjunto de gaussianas do tipo p e um conjunto de cinco gaussianas do tipo d [42]. Essa base possui 264 funções de bases, 200 funções a mais se comparadas com a 6-31G e 6-31G* que possuem 48 e 72, respectivamente.

2.2 Campo de força

O campo de força é descrito como uma função matemática usada para descrever a energia potencial de um sistema de partículas (átomos e moléculas). As funções e os parâmetros são derivados tanto de dados experimentais como de dados de alto nível (cálculos de mecânica quântica).

O campo de força ou Mecânica Molecular (MM) foi apresentado separadamente

por Hill e por Westheimer in 1949 [55], sendo aplicado para moléculas orgânicas com a finalidade de encontrar propriedades como energia, entre outras. Em 1960, Lifson aplicou o campo de força em sistemas biológicos. Os campos de força vêm sendo utilizados há muito tempo, fornecendo informações importantes como estrutura e funções das moléculas.

Existem diversas modelagens de campos de força na literatura, tais como: Amber (PARM99) [20, 54], CHARMm22 [56], GROMOS [57], OPLS/AA [58], entre outros. Esses campos de força usam um modelo clássico que nos permite prever:

- Estados de transição e estruturas de equilíbrio,
- Energias relativas entre isômeros conformacionais ou entre moléculas diferentes.

2.2.1 Termos do campo de força

Os termos para o cálculo do campo de força estão divididos basicamente em dois tipos: diretamente ligados (interações fortes) e os não diretamente ligados (interações mais fracas). Os termos diretamente ligados são descritos pelo cálculo do potencial entre as distâncias dos átomos, dos ângulos de ligação e ângulos torcionais. Os não diretamente ligados se dividem em dois tipos: interações de van der Waals e eletrostática. Estas interações são cruciais em determinados campos de força de uma molécula, e se não forem adequadamente descritas pelo modelo do campo de força, o resultado não irá reproduzir adequadamente as condições reais (físicas) do sistema [59].

A seguir será explicado alguns desses termos que compõem a maioria dos campos de força. As interações mais fortes são determinadas pelos átomos diretamente ligados por ligações covalentes, e calculadas pelas energias de ligação (E_{lig}), de ângulo (E_{ang}) e de torção (E_{tor}), mostradas na Figura 2.1.

O termo que representa a modelagem da energia das interações de estiramento e compressão entre dois átomos (Figura 2.1A) é composto pelo somatório de todas as interações entre os átomos diretamente ligados. No presente estudo está sendo usada a aproximação harmônica dada por:

$$E_{lig} = \sum_{todas\ lig.} K_r(r_i - r_{eq})^2, \tag{2.6}$$

onde r_i é a distância entre dois átomos ligados. Ela é derivada da lei de Hooke e r_{eq} representa a distância de equilíbrio. A constante K_r pode receber diferentes valores dependendo de que átomos estão envolvidos na ligação.

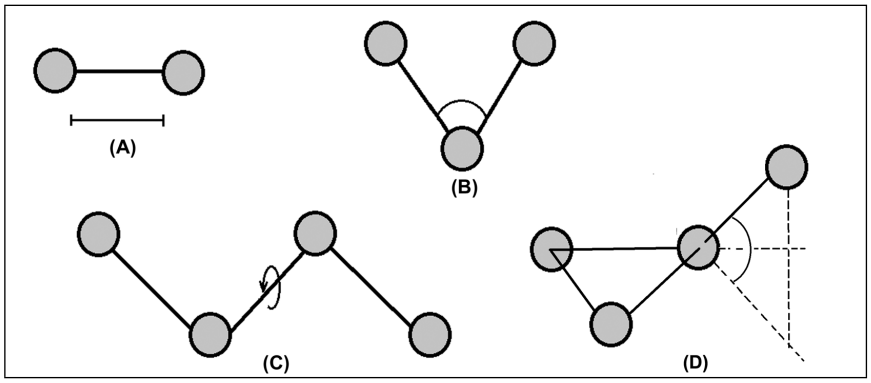


Figura 2.1. Representação entre átomos diretamente ligados: (A) Ligação covalente entre pares de átomos; (B) Ângulo entre os átomos e suas respectivas ligações covalentes; (C) Ângulo de torção entre os átomos e suas ligações covalentes e (D) Ângulo de torção impróprio entre os átomos e suas ligações covalentes.

A função escolhida para descrever a energia de deformação do ângulo de ligação (Figura 2.1B) é modelado pelo somatório das variações dos ângulos entre os átomos ligados. Similarmente ao caso dos estiramentos próximos à posição de equilíbrio, para o movimento angular em torno de uma dada ligação será também usada a aproximação harmônica dada por:

$$E_{ang} = \sum_{todos\ ang.} K_\theta(\theta_i - \theta_{eq})^2, \tag{2.7}$$

onde θ_i é o ângulo de equilíbrio entre três átomos ligados, θ_{eq} indica o ângulo em que os três átomos estão em equilíbrio e K_θ é a constante de força de deformação.

O próximo termo apresentado é onde temos a energia das interações dos ângulos diedros (Figura 2.1C) e é modelado pelo somatório dos ângulos torcionais, sendo descrita por uma função periódica mostrada a seguir:

$$E_{tor} = \sum_{todas\ tor.} \sum_{n}^{n_{max}} \frac{V_n}{2}[1 + \cos(n\varphi - \gamma)], \tag{2.8}$$

onde φ é o ângulo formado entre dois planos (quatro átomos diretamente ligados), utilizando quatro átomos consecutivos, n é a periodicidade desse ângulo, n_{max} é a priodicidade máxima de cada torção, γ é a fase e V_n é a constante de força.

A outra função, também utilizado no campo de força GROMACS [60], é a que calcula a energia da deformação dos ângulos torcionais impróprios, esses ângulos geram deformações fora do plano, dado pela Equação 2.9. Ele é um ângulo formado por quatro

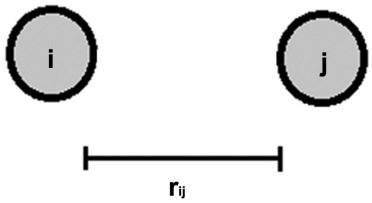
átomos não diretamente ligados (Figura 2.1D).

$$E_{imp} = \sum_{todas\ tor.} \frac{K_\varphi}{2}(\varphi - \varphi_0)^2, \tag{2.9}$$

onde K_φ é a constante de força da equção e φ_0 é o ângulo de equilíbrio da torção entre os quatro átomos.

Os outros três termos são cálculos feitos entre átomos não diretamente ligados. Esses termos são:

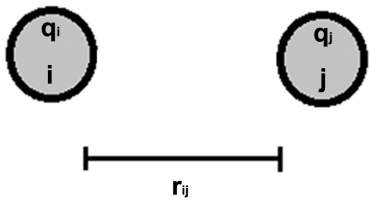
$$E_{vdW} = \sum_{i<j}^{N} 4\epsilon_{ij}[(\frac{\sigma_{ij}}{r_{ij})^{12}} - (\frac{\sigma_{ij}}{r_{ij})^{6}}], \tag{2.10}$$



A Eq. 2.10 representa o modelo da energia das interações de van der Waals (E_{vdW}), e inclui tanto as forças de atração quanto as de repulsão entre dois átomos, sendo N o número total das interações. Valores para as constantes ϵ e r, que estão implícitas nas variáveis A e B, dependem dos tipos de átomos que estão interagindo e a variável r_{ij} define a distância entre os átomos.

Já o termo E_{ele} define a energia das interações eletrostáticas repulsivas existentes devido às cargas entre dois átomos, e é representada por:

$$E_{ele} = \sum_{i=1}^{N-1} \sum_{j>i}^{N} \frac{q_1 q_2}{\epsilon r_{12}}, \tag{2.11}$$



onde as variáveis q_i e q_j são as cargas dos átomos que estão interagindo separados por uma distância r_{ij}.

Temos também termo que são muito utilizados no campo de força CHARMm. O termo chamado de energia Urey-Brandley, onde faz o cálculo entre pares de átimos separados por uma distância de duas ligações atômicas [61], dada por:

$$E_{urey} = \sum_{todas\ lig.} \frac{K_{urey}}{2}(s - s_0)^2, \tag{2.12}$$

onde K_{urey} é a constante de força da interação Urey-Brandley e s_0 é a distância entre os dois átomos.

É importante relembrar que o campo de força não é absoluto podendo possuir os mais diversos termos para descrever as condições reais do sistema em questão. A função escolhida neste estudo para o cálculo da energia da molécula é o campo de força Amber, largamente utilizado entre as famílias de campo de força, com 4.419 citações (http://pubs.acs.org/journal/jacsat). Além desse motivo, a escolha também foi feita porque o trabalho em aqui desenvolvido foi comparado com um trabalho recente da literatura que usou essa função [17]. A função Amber (ff99χOL) é descrita na Equação 2.13:

$$E_{tot} = E_{lig} + E_{ang} + E_{tor} + E_{vdW} + E_{ele} \tag{2.13}$$

Todos os termos do campo de força possuem uma constante de força. Esses valores são, teoricamente, retirados dos cálculos de mecânica quântica. Entretanto, na prática, eles são obtidos empiricamente. Ou seja, as constantes são ajustadas baseadas em estruturas de uma série de compostos já conhecidos. A precisão dessas constantes é de extrema importância, pois elas serão usadas para se calcular estruturas de novos compostos. Essa determinação dos parâmetros é crítica na mecânica molecular, pois não há um único conjunto de constantes de força disponível por causa da diversidade de tipos de compostos.

2.2.2 Tipos de átomos e seus parâmetros

O primeiro passo para iniciar um cálculo é o de identificar os diferentes tipos de átomos na molécula. Em alguns programas criados para o cálculo do campo de força isso deve ser feito manualmente pelo usuário. Em muitos programas existe uma rotina que realiza este passo automaticamente. No entanto, a atribuição de tipos de átomos automáticos pode ser incorreta, e por isso o usuário deve verificar para se certificar se os tipos de átomos foram atribuídos corretamente.

Todo campo de força possui uma tabela de tipos de átomos com o objetivo de padronizá-los, pois a escolha desses átomos é a base do campo de força. Na Tabela 2.1 são apresentados os tipos e as características de alguns átomos. Os átomos do tipo sp^3 são relativamente comuns, porém foram inseridos os sp^2 para garantir maior precisão na geometria de sistemas com anéis, como purinas, pirimidinas, entre outros. A tabela completa está no apêndice A deste trabalho.

A seguir são apresentadas as tabelas referentes aos termos do campo de força apresentado pela Eq. 2.13 para facilitar o entendimento do leitor quando for explicado

Tabela 2.1. Lista dos tipos de átomos

Átomo	Tipo	Descrição
Carbono	CT	qualquer carbono sp^3
	C	qualquer carbonila aromática sp^2
	CK	carbono sp^2 em anel aromático com 5 membros e entre nitrogênios
	CM	qualquer carbono sp^2, com ligação dupla
Nitrogênio	N*	nitrogênio sp^2 com 5 membros no anel e carbono subsequente
	NA	nitrogênio sp^2 em anéis aromáticos com hidrogênio
	N3	nitrogênio sp^3
Oxigênio	OS	qualquer oxigênio sp^3 em éteres
	O	qualquer oxigênio sp^2 em amidas
Hidrogênio	H	hidrogênios ligados a átomos de carbono alifático
Fósforo	P	fósforo em fosfatos

a metodologia do trabalho. As tabelas completas estão no apêndice A. Todas foram retiradas do programa Tinker [62].

A Tabela 2.2 nos mostra alguns dos parâmetros do termo que calcula a energia de ligação entre os átomos do sistema. Nela temos as constantes de força e a distâncias de equilíbrio de cada par de átomos diretamente ligados.

Tabela 2.2. Parâmetros do termo de ligação

Átomos	K_r^a	r_{eq}^b
CT-N*	337,0	1,475
CK-NB	529,0	1,304
CK-N*	440,0	1,371
CM-CM	549,0	1,350
CM-CT	317,0	1,510
OS-P	230,0	1,610
CT-CT	310,0	1,526
CT-OH	320,0	1,410
CT-OS	320,0	1,410
CM-HA	367,0	1,080
CM-N*	448,0	1,365

[a] Constante em kcal/mol.
[b] Distância entre átomos em Å.

Na Tabela 2.3 são apresentados alguns parâmetros do termo da deformação angular do campo de força. Aqui temos as constantes de força e os ângulos de equilíbrio entre três átomos diretamente ligados.

A Tabela 2.4 apresenta alguns dos parâmetros relacionados ao termo de torção do campo de força: as constantes de força (barreira torcional), ângulos torcionais, fases e periodicidades das torções entre quatro átomos diretamente ligados.

Tabela 2.3. Parâmetros do termo angular

Átomos	K_θ^c	θ_{eq}^d
C-CM-CM	63,0	120,7
C-CM-CT	70,0	119,7
CK-N*-H	30,0	128,8
CT-CT-OH	50,0	109,5
CT-CT-OS	50,0	109,5
CT-OS-CT	60,0	109,5
CT-OS-P	100,0	120,5
CT-OH-HO	55,0	108,5
OH-P-OS	45,0	102,6
H1-CT-OH	50,0	109,5
H1-CT-OS	50,0	109,5
N*-CK-NB	70,0	113,9
N*-CT-OS	50,0	109,5

[a] Constante em kcal/mol.
[b] Ângulo entre átomos em graus.

Tabela 2.4. Parâmetros do termo de torção

Átomos	$V_n/2^a$	φ^b	n
CT-CT-N-C	0,5	180	4
CT-CT-N-C	0,15	180	3
CT-CT-N-C	0	180	2
CT-CT-N-C	0,53	0	1
CT-CT-OS-CT	0,383	0	3
CT-CT-OS-CT	0,1	180	2
OH-P-OS-CT	0,25	0	3
OH-P-OS-CT	1,2	0	2
OS-CT-CT-OH	0,144	0	3
OS-CT-CT-OH	1,00	0	2
OS-CT-CT-OS	0,144	0	3
OS-CT-CT-OS	1,00	0	2
OS-CT-N*-CK	0,5	180	2
OS-CT-N*-CK	2,5	0	1

[a] Constante em kcal/mol.
[b] Periodicidade em graus.

Na Table 2.5 temos alguns valores que são usados como parâmetros do termo de van der Waals (Eq. 2.10) do campo de força para calcular a energia de interação entre os átomos i e j, onde $r_{ij} = r_i + r_j$ e $\epsilon_{ij} = (\epsilon_i \epsilon_j)^{1/2}$.

Por fim, alguns dos parâmetros das cargas que foram obtidos através de cálculos quânticos (Tabela 2.6).

Tabela 2.5. Parâmetros de van der Waals

Átomo	r^a	ϵ^b
CT	1,9080	0,1094
C	1,9080	0,0860
N*	1,8240	0,1700
CM	1,9080	0,0860
OS	1,6837	0,1700
P	2,1000	0,2000
OH	1,7210	0,2104
HA	1,4590	0,0150
H1	1,3870	0,0157

[a] Raio de van der Waals em Å.
[b] Profundidade do poço de van der Waals em kcal/mol.

Tabela 2.6. Parâmetros de cargas

Átomo	Cargas
CT	0,1851
NA	-0,4684
N*	-0,4770
CM	-0,6013
OS	-0,6411
P	1,3461
OH	-0,6712
HA	-0,0877
H1	0,0048

Com auxílio dessas tabelas de parâmetros podemos realizar o cálculo de energia usando a Eq. 2.13 para qualquer sistema que possua os átomos que já foram parametrizados pelo campo de força Amber.

Capítulo 3

Computação Evolutiva

Os Algoritmos de computação evolutiva (CE) são aplicados em diversas áreas como: ciências naturais, engenharia, biologia e ciência da computação. Na CE não se necessita conhecer previamente como encontrar a solução e nem de uma modelagem matemática formal para o problema [63].

Os primeiros trabalhos na área de algoritmos de computação evolutiva (CE) surgiram aproximadamente há 60 anos [64, 65]. Eles baseiam-se nos princípios de Darwin, que simulavam os processos da Teoria da Evolução [66]. Esses trabalhos se inspiram nesses princípios para solucionar problemas, principalmente na área de otimização. Esses algoritmos têm uma área de abrangência ampla, pois os problemas de otimização podem ser encontrados em diversas áreas como: ciência da computação, engenharia, biologia, entre outros [67].

Uma das vantagens do uso da CE em relação a outros métodos de otimização é a capacidade desses algoritmos de trabalhar com problemas que possuem um espaço de busca muito grande (muitas variáveis), e por dispensarem uma descrição detalhada do problema. Por exemplo, para se utilizar um algoritmo de busca que utiliza métodos de gradiente necessita-se que a função a ser otimizada (função *fitness*) seja diferenciável e sua derivada calculada com baixo custo computacional [68]; por outro lado, para o método de programação linear é necessário que a função seja linear [68]. Os algoritmos de CE aparecem quando não temos nenhuma destas características. Dessa forma eles se tornam uma das poucas maneiras para se encontrar uma solução para o problema em questão [67].

Outra justificativa para se utilizar a CE é que este tipo de método gera soluções relativamente aceitáveis em um tempo computacional pequeno quando se trata de problemas de alta complexidade. Uma vez que utilizando técnicas clássicas (tradicionais) para resolver problemas desse tipo com uma solução ótima, seria necessário uma

quantidade muito alta de recursos computacionais para tratar problemas totalmente proibitivos (intratáveis), com o auxílio da CE temos soluções aproximadas, ou ótimas, necessitando de poucos recursos computacionais.

3.1 Algoritmo Genético

Um dos algoritmos que fazem parte da CE é o algoritmo genético (AG), que surgiu na década de 1960 e foi criado por John Holland, na Universidade de Michigan, e posteriormente aperfeiçoado por seus alunos. O elemento essencial de um AG é a população baseada em variações e seleções randômicas de indivíduos [69].

O AG é um método que consiste em uma população inicial (conjunto de indivíduos que representam as possíveis soluções para o problema) que será modificada em uma nova população utilizando a teoria de Darwin, onde os mais aptos sobrevivem (seleção natural), e os operadores de cruzamento (ou *crossover*) e mutação são utilizados em um processo iterativo de busca da melhor solução para o problema.

O primeiro passo em um AG é a codificação das variáveis de busca em um cromossomo. A maneira como as soluções são representadas é de fundamental importância para o sucesso dos algoritmos genéticos. A forma de representação pode variar de muitas maneiras e basicamente depende do tipo de problema tratado. De modo geral, as formas de representação que têm recebido mais atenção recentemente são aquelas que consideram características específicas do problema. A representação pode dar-se por uma *string* binária (0s e 1s) ou um vetor de números reais.

A cada nova iteração é gerada uma nova população que aliado aos parâmetros apropriados, apresentam novas e melhores soluções para o problema em questão, culminando com a sua convergência.

O processo de evolução dos indivíduos num AG é um procedimento de busca num espaço de possíveis soluções do problema. Porém, segundo Michalewicz (1996), necessitamos equilibrar dois objetivos conflitantes: o aproveitamento das melhores soluções e a exploração do espaço de busca [67]. Caso isso não aconteça provavelmente ficaremos presos em ótimos locais. Portanto, o processo de busca é multidirecional, mantendo uma variabilidade entre os indivíduos (várias partes do espaço de busca) e com troca de informações entre eles.

A cada geração os indivíduos relativamente "bons" trocam material genético, enquanto que soluções relativamente "ruins" tendem a ser extintas da população. Como método para avaliação de cada indivíduo é empregada a função de avaliação (*fitness*), que indica se a solução é boa ou não para o problema.

Tabela 3.1. Exemplo *fitness*

Indivíduo	Informação	Fitness
1	010110010	3
2	111110001	6
3	101011010	5
4	000000011	2

Segundo David Fogel [69] os algoritmos genéticos são na maioria das vezes representados pela forma canônica como pode ser visto a seguir:

- Representação do indivíduo (etapa de codificação). Os candidatos à população são iniciados com algumas restrições. Cada candidato é geralmente codificado em um vetor x, chamado de cromossomo, com elementos chamados genes e suas posições, chamadas alelos;
- Será criada uma população inicial aleatória;
- Função de *fitness* (como resolver o problema). É definida como a habilidade de um indivíduo sobreviver e reproduzir-se em um ambiente;
- Escolha dos operadores genéticos;
- Seleção dos valores para os diversos parâmetros do AG tais como, tamanho da população, probabilidades de aplicação dos operadores genéticos, entre outros.

3.1.1 Função de *Fitness*

A *fitness* ou função objetivo nada mais é do que a uma forma de avaliação indicando o quanto um indivíduo (possível solução) é "bom" ou não para o problema em questão. Dessa forma, ela se torna essencial para o algoritmo, pois caso seja mal planejada, dificilmente o método conseguirá sucesso. Caso um indivíduo potencialmente "bom" para o problema seja avaliado de forma incorreta, provavelmente, ele será perdido durante a evolução do AG.

O exemplo que a Tabela 3.1 apresenta, mostra a importância dessa *fitness*, que seria: encontrar o maior número de 1s que um indivíduo possa ter.

Imagine se ao invés da *fitness* contar o número de 1s ela contasse o número de 0s. Esse algoritmo nunca iria chegar numa resposta aceitável para o usuário. Por isso devemos tomar muita atenção na hora de se determinar a *fitness* para o AG.

Outro ponto importante é a escolha apropriada dos melhores indivíduos para a exploração do seu espaço de busca. Na próxima seção são apresentados alguns métodos de seleção.

3.1.2 Métodos de Seleção

A função dos métodos de seleção é escolher os indivíduos de uma população que participarão do processo de reprodução (troca de material genético). Esta escolha é feita de maneira que a probabilidade dos indivíduos selecionados da população sejam os mais aptos, ou seja, aqueles que tenham o melhor valor da função *fitness*.

Nesta seção iremos fazer uma breve revisão dos três principais métodos de seleção. Cada método de seleção possui uma pressão seletiva diferente [41]. A pressão seletiva nada mais é do que a velocidade de convergência da população (indivíduos com valores de *fitness* próximos e ótimos). Quanto maior é sua pressão seletiva, mais rápido o algoritmo converge.

3.1.2.1 Roleta

A técnica denominada seleção por roleta foi formulada por John Holland em 1975. Este método é um dos mais simples, onde a probabilidade de um indivíduo ser selecionado é proporcional a sua *fitness*. Neste método, os pais (indivíduos selecionados da população) são selecionados de acordo com sua *fitness*, eles não podem ter valores nulos (0) e nem negativos. Quanto melhores são os indivíduos, maior as chances deles serem escolhidos. Por exemplo, imagine uma roleta onde são colocados todos os indivíduos da população. O espaço ocupado na roleta é proporcional ao valor da *fitness* de cada indivíduo: quanto maior for esse valor, maior é essa a área que eles ocupam. O número de vezes que a roleta gira é proporcional ao tamanho da população. Veja a Figura 3.1 para exemplificar melhor.

Indivíduo	Informação	Fitness	%
1	000110010	6	50
2	101110001	3	20
3	100011010	1,5	15
4	010110011	1,5	15

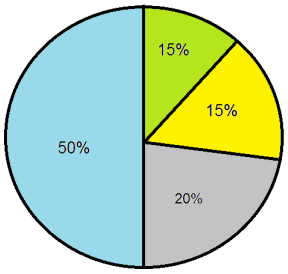


Figura 3.1. Representação do método de seleção por Roleta.

Na figura são observados quatro cromossomos com suas *fitness* correspondentes. A porcentagem que está apresentada na coluna ao lado serve para indicar o quanto de espaço que esse cromossomo vai ocupar na roleta, como ilustrado no gráfico ao lado. A fórmula para o cálculo do espaço que cada indivíduo irá ocupar é dado pela Equação 3.1. E assim, pode-se verificar que o cromossomo de número 1 tem maior probabilidade de ser escolhido entre os outros três.

$$Fitness = \sum_{i=1}^{N} \frac{f_i}{N}, \tag{3.1}$$

onde i é o número de indivíduos e f_i é o valor da *fitness* de cada indivíduo.

3.1.2.2 Torneio

Na seleção por torneio, apresentada por David Goldberg em 1991, duas ou mais estruturas são escolhidas aleatoriamente na população e de acordo com sua probabilidade, o indivíduo mais adaptado será selecionado para reprodução.

Esse é um dos métodos que nos permite fazer ajustes na pressão seletiva, através do número de indivíduos pertencentes ao torneio [41]. Neste método, k indivíduos são aleatoriamente selecionados da população e competem um com o outro por uma chance de passar para os operadores genéticos. O indivíduo com a melhor *fitness* é selecionado, como mostrado no Figura 3.2.

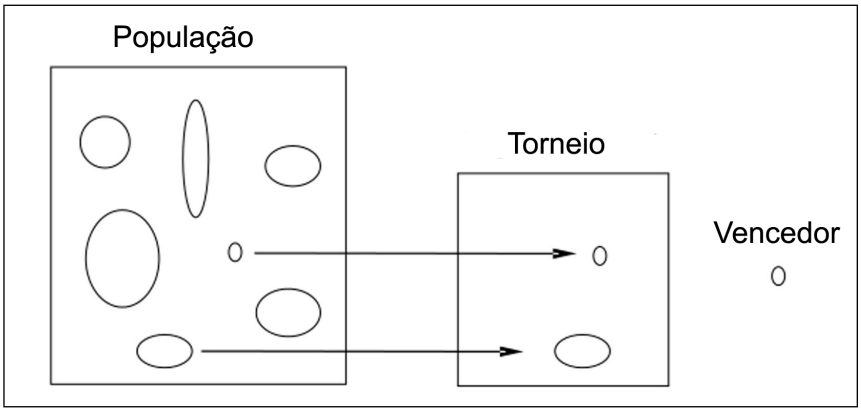


Figura 3.2. Representação do método de seleção por Torneio.

3.1.2.3 Ranking

Nesse método todos os indivíduos recebem uma classificação de acordo com o valor de sua *fitness*. Este método é semelhante ao da Roleta, onde os indivíduos são ordenados

de acordo com sua *fitness*. Quanto maior o *ranking* maior a chance de o indivíduo ser escolhido.

Outros tipos de seleção e mais detalhes sobre os citados aqui podem ser encontrados em Back *et. al.* 2000 [40].

3.1.3 Operadores Genéticos

Descreveremos aqui os operadores genéticos que junto com os métodos de seleção são os pontos mais importantes do algoritmo, pois são responsáveis pela busca das melhores soluções no AG. Eles são os responsáveis pela evolução dos indivíduos da população.

3.1.3.1 Operador de Cruzamento

O operador de cruzamento tem como função fazer a troca de material genético entre dois indivíduos (pais), para gerar dois novos indivíduos (filhos). Existem vários tipos de operadores de cruzamento, uns desenvolvidos para serem mais genéricos e outros mais adequados a um tipo de codificação de indivíduos. Existem vários operadores de cruzamento [41].

Ponto de Cruzamento Um único ponto de cruzamento é escolhido aleatoriamente. Dados dois cromossomos de comprimento l que são chamados de pais, sorteia-se um número p (ponto de cruzamento) tal que $0 < p < l$ para produzirem dois filhos. O primeiro filho receberá os genes do primeiro pai de zero até p e todos os genes do segundo pai de p até l, e o segundo filho receberá os genes de segundo pai de zero até p e do primeiro pai de p até l [70]. Como mostrado na Figura 3.3.

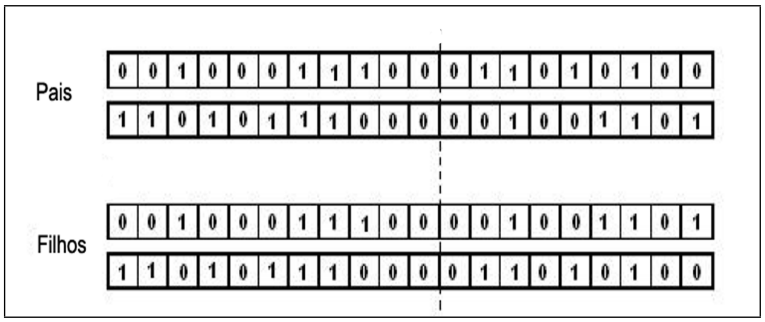


Figura 3.3. Representação do operador do cruzamento de 1 ponto (1PX).

Pontos de Cruzamento Semelhante ao cruzamento de um ponto, também é comumente utilizado o cruzamento de dois pontos (2PX), onde são escolhidos dessa vez

dois pontos de corte. A informação do início do cromossomo até o primeiro ponto de cruzamento é copiada do primeiro pai, a partir do primeiro ponto de cruzamento até o segundo ponto é copiada do outro pai e a outra parte do cromossomo é copiado do primeiro pai novamente. Como mostrado na Figura 3.4.

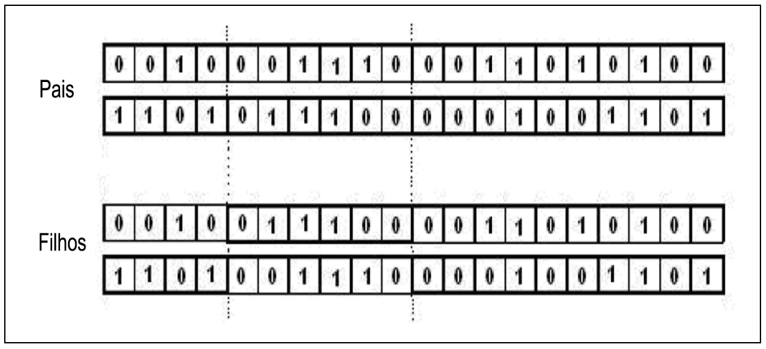


Figura 3.4. Representação do operador do cruzamento de 2 pontos (2PX).

Cruzamento Uniforme Nesse tipo de cruzamento é usada uma máscara aleatória entre os indivíduos. Cada posição do gene dos cromossomos dos pais são trocados numa posição escolhida pela máscara formando assim seus novos descendentes [41]. A Figura 3.5 ilustra o procedimento. Nesse exemplo foram escolhidos aleatoriamente, os genes 2 e 4 (círculos em cinza) para o cruzamento. Em seguida é verificada a mudança em suas respectivas posições.

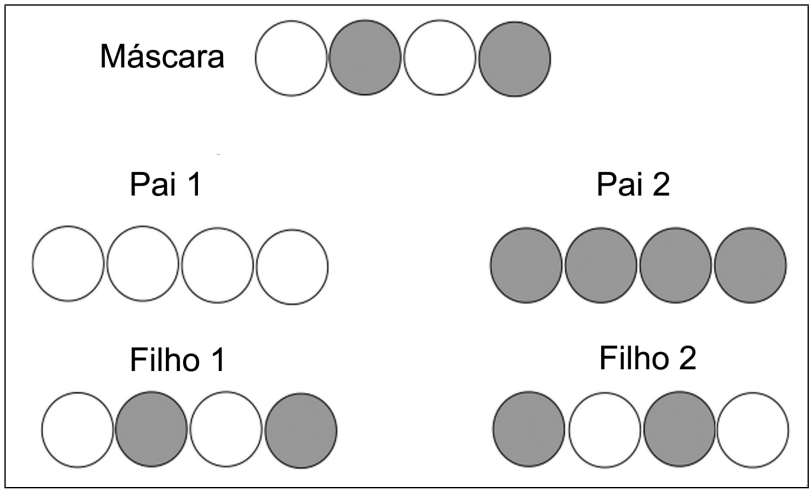


Figura 3.5. Representação do operador do cruzamento Uniforme.

3.1.3.2 Operador de Mutação

O operador de mutação tem como função inserir diversidade na população com o objetivo de tentar encontrar um indivíduo melhor (ótimo). Seu funcionamento se dá com a troca de partes do cromossomo (genes) de um indivíduo com valores gerados aleatoriamente.

Bit a bit O operador do tipo bit a bit é um dos mais fáceis para se usar, podendo ser utilizado em qualquer forma de representação de indivíduos. A mutação que atua em um único indivíduo consiste em selecionar um indivíduo e mudar certas características do mesmo, dessa maneira produzindo um novo indivíduo para a população. Este método gera aleatoriamente valores de probabilidade para cada gene do cromossomo, caso a mesma esteja dentro da faixa determinada pelo usuário o gene sofrerá a mutação estabelecida.

A Figura 3.6 exemplifica esse tipo de mutação, com um indivíduo de comprimento 4 e com a representação binária. Esse indivíduo foi selecionado aleatoriamente de uma população de tamanho X. Cada bit do indivíduo possui um valor para a probabilidade de mutação (gerado aleatoriamente), então o algoritmo verifica um por um se esses valores são menores ou igual à taxa de mutação (parâmetro do AG determinado pelo usuário), caso positivo ele troca o valor 0 por 1 ou vice versa. Para este exemplo foi determinado que os valores menores que 5% serão mutados.

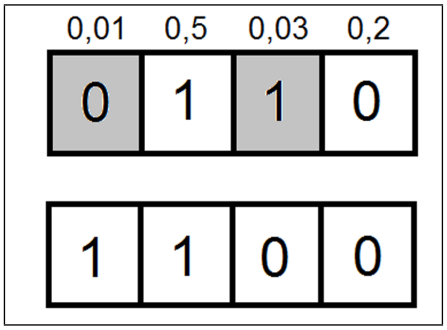


Figura 3.6. Representação do operador da mutação bit a bit.

As diferentes etapas do AG podem ser aplicadas de diferentes maneiras que dependem do tipo de seu problema. Por exemplo, há várias técnicas de cruzamento, seleção e de mutação a serem usados. Com isso o AG busca possíveis soluções para o problema a partir de uma busca em sua população inicial aplicando esses três métodos (seleção, cruzamento e mutação), a fim de evoluir essa população e encontrar melhores soluções de forma semelhante ao da evolução natural.

Neste trabalho foi desenvolvido um algoritmo genético em linguagem Java e usando a ferramenta Eclipse (ambiente de desenvolvimento em Java), devido à grande portabilidade e robustez que o Java oferece. Mais detalhes sobre o algoritmo no capítulo 4.

Capítulo 4

Metodologias aplicadas para o refinamento das ligações glicosídica e fósforo diéster

Este capítulo detalha como o trabalho foi implementado, mostrando passo a passo afim de facilitar o entendimento do leitor. Ele foi dividido em seções para para facilitar a compreensão e a leitura. No apêndice D é apresentado o código em Java do AG implementado.

A metodologia apresentada neste capítulo (Figura 4.1) que tem o objetivo de otimizar os parâmetros do termo de torção das ligações glicosídicas, pois é uma das ligações mais importantes dos nucleotídeos [71], denominado sistema I, e fósforo diéster do campo de força, denominado sistema II, a fim de descrever os nucleotídeos do RNA que serão detalhados ao longo deste capítulo. Os procedimentos foram realizados usando diversos cálculos quânticos para obter dados de referência (energia e estrutura). Estes dados serviram de referência para a parametrização do campo de força usando outros métodos [7], neste trabalho, um algoritmo genético.

Na Figura 4.1 temos a representação esquemática, passo a passo, indicando como serão implementadas as abordagens a seguir. Para efeito de visualização, fica implícita em todas as setas da figura a transferência das coordenadas em cada passo. Como a mesma metodologia foi aplicada para os dois sistemas, então nas seções em que os métodos forem similares será informado que no próximo sistema a implementação ocorrerá do mesmo jeito, evitando assim ter seções repetidas.

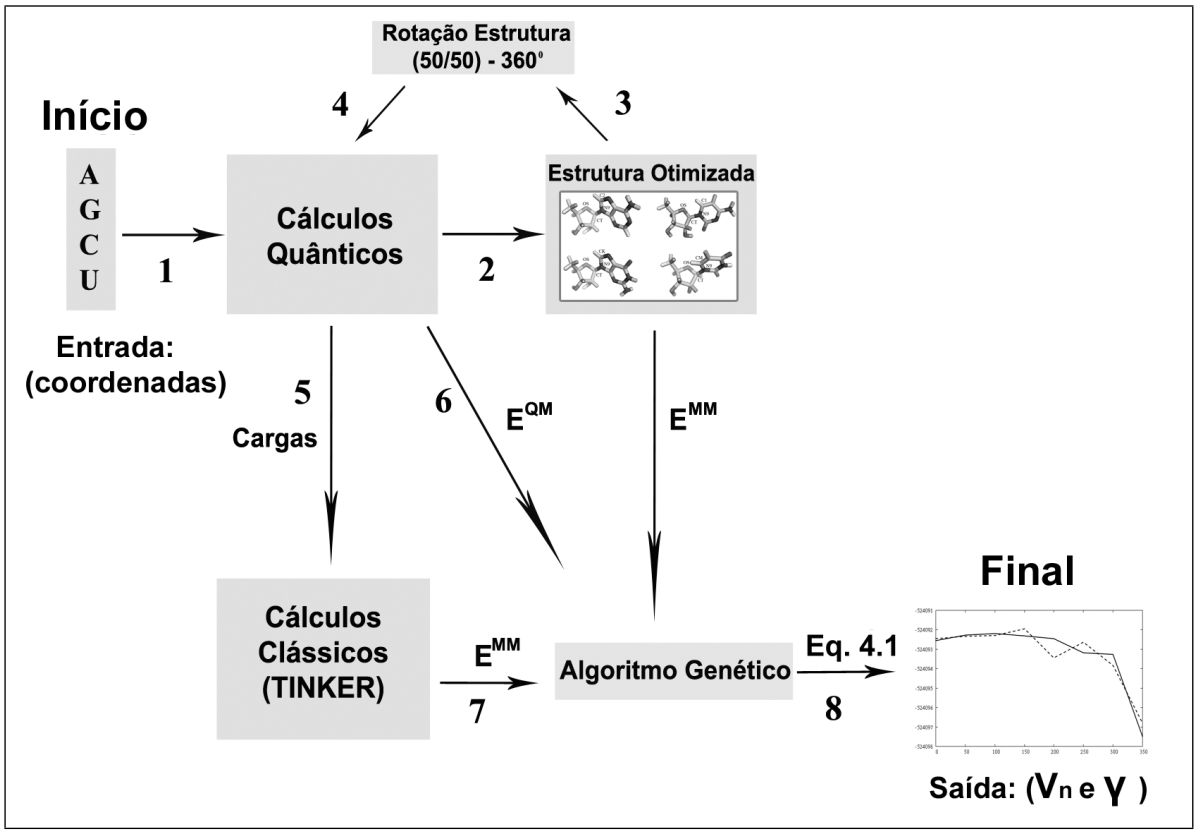


Figura 4.1. Apresentação esquemática das metodologias propostas (AGN e AGW), onde A é a adenina, U é a uracila, C é a citosina e G é a guanina que são definidos pelas suas coordenadas.

4.1 Modelo do Sistema I

Na primeira parametrização, foram desenvolvidas e aplicadas duas abordagens para otimizar os parâmetros dos termos torcional e eletrostático do campo de força para RNA, como já foi explicado. Para ambas as abordagens são realizados cálculos quânticos para a obtenção dos dados de referência (energias e estruturas) para o algoritmo genético no refinamento do campo de força.

Na primeira abordagem (AGN) o refinamento dos parâmetros do termo torcional da ligação glicosídica é realizado utilizando os mesmos parâmetros da literatura, ou seja, todos os termos do campo de força ficam constantes e apenas o termo torcional (E_{tor}) varia. Na representação esquemática (Figura 4.1) essa abordagem (AGN) é representada seguindo os passo 1, 2, 3, 4, 6, 7 e 8 (exceto o 5). Já na segunda abordagem (AGW) é realizado um ajuste no termo eletrostático (explicado mais tarde) que ajudará o AG no refinamento dos parâmetros do termo torcional da ligação glicosídica para cada nucleosídeo. Para este caso, seguimos todos os passos da representação es-

quemática (Figura 4.1). O ajuste no termo eletrostático foi realizado, pois este termo é de fundamental importância para o campo de força [72, 73]. Devido a essa importância foram utilizadas as cargas dos cálculos quânticos obtidos na otimização das energias de torção.

Preparação da base de dados. As estruturas dos nucleotídeos foram retiradas de um repositório de estruturas tridimensionais, chamado *Nucleic Acid Database* (NDB). Este foi criado em 1992, usa formatos ndb e pdb, é financiado pela *National Science Foundation e Departament of Energy* dos Estados Unidos, e conta atualmente com mais de 7 (seis) mil estruturas [74]. Inicialmente foram utilizados dois dinucleotideos (Adenina e Uracila; Guanina e Citosina), como nesse modelo iremos trabalhar apenas com as ligações glicosídicas, então a base de cada dinucleotideo foi isolada (Figure 4.2) e inseridos os hidrogênios usando o programa openBabel [75], para em seguida servir de dados de entrada para os cálculos quânticos.

Depois da preparação, as estruturas são otimizadas através de cálculos quânticos realizados pelo programa GAMESS [76]. As saídas são: as estruturas otimizadas das bases de entrada pelo AGN e AGW, o conjunto de cargas de cada átomo envolvido nos cálculos (somente para o AGW) e as energias quânticas (para o AGN e AGW), que serão usadas no próximo passo dessa abordagem. Esses resultados são usados no refinamento dos parâmetros do campo de força (AGN: somente os parâmetros torcionais e AGW: os parâmetros eletrostático e torcional). No passo seguinte foi usado o algoritmo genético para realizar a otimização da torção glicosídica. O resultado final é o conjunto de novos parâmetros de torção ajustados com base nos dados quânticos dado por:

$$\min_{w} \parallel E_i^{QM} - E_i^{MM}(w) \parallel^2, \tag{4.1}$$

onde E^{QM} e E^{MM} são as energias quânticas e moleculares, respectivamente; w é definido como:

$$\begin{aligned} w_{AGN} &= \{V_n, \gamma\} \\ w_{AGW} &= \{V_n, \gamma; q_i^Q; q_j^Q\}, \end{aligned} \tag{4.2}$$

onde para w_{AGW} as cargas q_i^Q e q_j^Q são obtidas através dos cálculos quânticos da estrutura torcionada e serão usadas na Eq. 2.11.

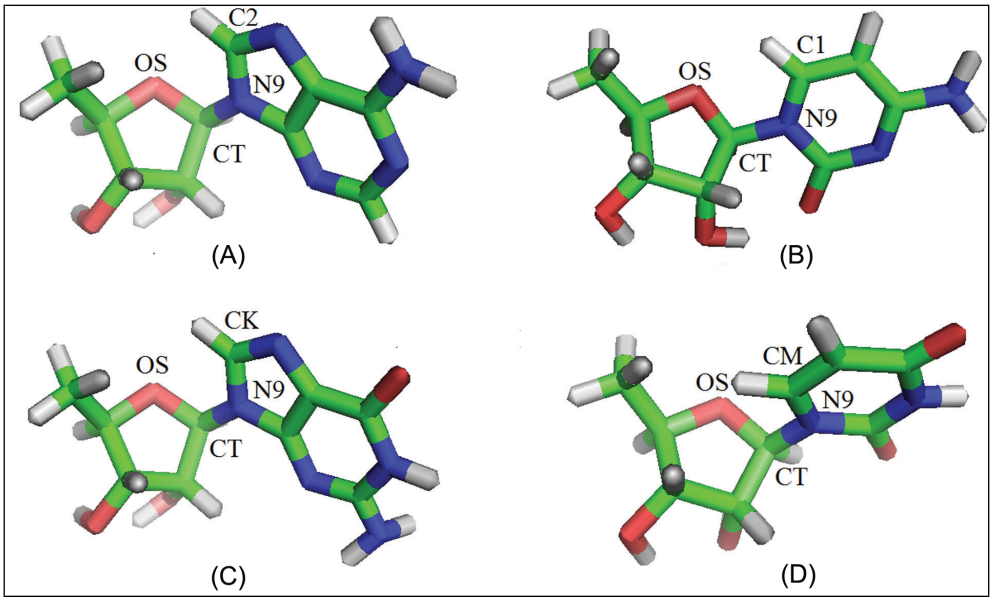


Figura 4.2. Moléculas utilizadas como entrada (A) Adenosina, (B) Citosina, (C) Guanosina e (D) Uridina. O ângulo torcional da adenosina é definido pelos átomos OS-CT-N9-C2, OS-CT-N9-C1 para a citosina, OS-CT-N9-CK para a guanosina e OS-CT-N9-CM para a uridina (definição usada neste trabalho).

4.1.1 Obtendo os Perfis de Torção

Os perfis de torção foram obtidos a fim de gerar testes preliminares para a metodologia proposta (AG). Consideramos um número reduzido de cálculos quânticos para parametrizar a energia de torção devido à necessidade de uma grande demanda computacional. Para este trabalho foi mapeado um subconjunto de possíveis ângulos diedros da molécula fazendo rotações com o passo de 50 graus na torção glicosídica, em um intervalo de 0 a 360 graus. Caso a proposta obtenha um bom resultado, então pode-se esperar resultados similares para os outros ângulos. Outros trabalhos na literatura realizam o mesmo processo, porém com rotações com passos de 10 graus [17]. No entanto, a principal finalidade desta análise está relacionada com a eficácia da abordagem. Para cada uma das sete novas estruturas geradas com a rotação (360 graus em passo de 50 graus) foram realizados cálculos quânticos para encontrar uma melhor estrutura.

Na primeira parametrização, o AG aplicado nas abordagens AGN e AGW recebem como dados de referência as sete estruturas otimizadas com a rotação e a outra sem rotação para cada nucleosídeo. O algoritmo usa a mesma estrutura para ambos os cálculos tanto de mecânica molecular quanto de quântica. Este procedimento é similar ao realizado pelo trabalho de Zgarbová [17] que aplicou o método de Monte Carlo para o refinamento destes parâmetros. Como medida de comparação, a análise é realizada usando o RMSE (*Root Mean Square Error*) que é a diferença das energias para cada

conformação, descrita como:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}[(E_{i-1}^{QM} - E_{i}^{QM}) - (E_{i-1}^{MM} - E_{i}^{MM})]^2}{n}}, \tag{4.3}$$

onde i é a rotação, n é o número total das rotações, E^{QM} é a energia quântica e E^{MM} é a energia da mecânica molecular.

A energia (E^{MM}) é calculada usando o campo de força Amber (ff99χOL), que usa os parâmetros retornados pelo AG. A diferença entre as energias de rotação (0 à 360 graus com passo de 50 graus) são comparados com as diferenças da energia quântica (E^{QM}) para cada molécula (Adenosina, Guanosina, Citosina e Uridina) calculados pelo GAMESS.

Essa mesma metodologia se estende para o sistema II, apenas mudando a molécula e o ângulo de torção que nesta caso será feito na ligação fósforo diéster.

4.1.2 Cálculos Quânticos

Para a realização dos cálculos quânticos foi instalado o programa GAMESS no cluster, disponível no Laboratório de Computação Científica (LCC), devido sua grande capacidade de processamento, e desse modo podemos dispor do módulo em paralelo do programa utilizando até 8 (oito) processadores, tornando o cálculo mais rápido do que executá-lo em um computador de mesa.

A escolha dos métodos quânticos foi baseada em trabalhos recentemente publicados na literatura [17]. Poucas parametrizações do campo de força Amber (ff94 [20], ff98 [21] e ff99 [22]) foram realizadas usando o método Hartree-Fock. Porém, neste método os elétrons interagem sem levar em conta as correlações eletrônicas. Este fato gera grandes consequências na interpretação das funções de onda [77]. Assim, estudos recentes focaram em métodos mais eficazes. Por exemplo, o último aperfeiçoamento do campo de força CHARMm otimiza as geometrias com base em cálculos MP2/6-31+G* e RI-MP2/cc-pVTZ usando a teoria do funcional de densidade. Por outro lado, os cálculos baseados em DFT são utilizados para melhorar o potencial de torção para o campo de força Amber [17, 23].

Para o sistema I usaremos o nível de teoria (*Density Functional Theory* - DFT), mais especificamente o funcional PBE, e a base 6-311++G (3df, 3pd) como, recentemente, também foi utilizado na referência [17] e mostrou bons resultados comparados com dados quânticos. Outro motivo que foi levado em consideração é sua maior precisão comparados com Hartree-Fock e mais rápido do que os métodos MP2, MP3, entre outros.

Para o sistema II a escolha do segundo cálculo quântico também foi baseada em trabalhos publicados na literatura. Também foi realizada usando o método DFT, porém devido à mudança da molécula (inserção do grupo fosfato) o funcional foi trocado para o TPSS, mas a base (6-311++G (3df, 3pd)) permaneceu a mesma. Essa escolha foi baseada na referência [7] que utilizou esse mesmo funcional e base para uma molécula semelhante à molécula que estamos usando nessa segunda reparametrização.

4.1.3 Cálculos de Mecânica Molecular

Como mencionado anteriormente, os parâmetros refinados pelo AG são testados com o campo de força Amber, definidos pela Eq. 2.13 [17]. A equação é representada pela soma de várias contribuições de energia, ou seja, a ligação de estiramento (E_{lig}), ângulo de flexão (E_{ang}), torção (E_{tor}), eletrostática (E_{ele}) e termo de van der Waals (E_{vdW}), como já foi dito.

O AG evolui apenas os parametros do termo torcional (veja a Eq. 2.8 para mais detalhes), pois de acordo com Lankas *et. al.* [71] a ligação glicosídica é uma das mais importantes dos ácidos nucleicos. Na abordagem AGN todos os parâmetros foram retirados da referência [17] e os parâmetros do termo torcional são otimizados. Na abordagem AGW, as cargas do termo eletrostático foram retiradas dos cálculos quânticos realizados pelo GAMESS. Como existem diversas rotações, então foi realizada uma média das cargas para cada átomo, que é calculado como:

$$\bar{\delta_i} = \frac{1}{N} \sum_{0}^{360} \delta_i, \tag{4.4}$$

onde N define o número de passos dos ângulos (neste trabalho N é igual a 8) e δi é a densidade de carga de cada átomo i (q_i). Em seguida, os valores da média dessas cargas são usados no termo eletrostático (Eq. 2.11) do campo de força.

No sistema II os parâmetros encontrados para a torção glicosídica são mantidos e os novos parâmetros, relacionados à ligação fósforo diéster, são evoluídos apenas pelo AGW, pois foi a melhor abordagem se comparada com o AGN.

Todos os cálculos de mecânica molecular foram realizados pelo programa Tinker. Este programa é amplamente utilizado na literatura, e os exemplos desses estudos são descritos em outros trabalhos [78, 79, 80].

4.1.4 A Otimização do Termo de Torcional I usando o AG

Os Algoritmos genéticos são muito conhecidos na realização de uma pesquisa global ao analisar várias soluções do espaço de busca em cada iteração, como já foi detalhado na seção 3.1. Estes algoritmos trabalham com a representação de indivíduos, onde cada indivíduo representa uma solução para o problema a ser resolvido. O indivíduo é representado por um vetor de 8 posições, como ilustrado na Figura 4.3, representando os termos V_n e γ da Equação 2.8.

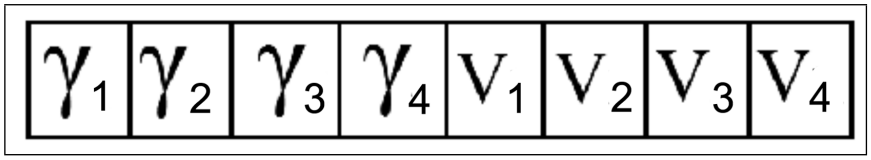


Figura 4.3. Representação do indivíduo do Algoritmo Genético da primeira parametrização.

A escolha dessa representação foi baseada na tabela de torção [20] que informa que para a torção da ligação glicosídica existem 4 periodicidades. Dessa forma precisaríamos de 4 barreiras torcionais (V_n) e 4 fases (γ), totalizando 8 variáveis para o cálculo da energia. Como os valores que elas assumem são valores reais achamos mais coerente trabalhar com a representação numérica ao invés da binária.

O procedimento de busca padrão do AG é mostrado na Figura 4.4. No primeiro passo, uma população inicial de indivíduos é gerada aleatoriamente, seguindo uma distribuição normal com média 0 e desvio padrão definido como 1. Em seguida, os indivíduos são avaliados de acordo com o quão bem eles resolvem o problema em questão. Como os nossos indivíduos são parâmetros de torção, são avaliados de acordo com o quanto distante a energia da mecânica molecular E^{MM} (obtida por Tinker com os parâmetros otimizados AG) está da energia quântica E^{QM} (obtido por GAMESS). O RMSE (já definido na Eq. 4.3) será a função de *fitness* definida pelo AG. Quanto menor o RMSE, melhor é a aproximação do campo de força. Entretanto, o AG irá otimizar os parâmetros (para o AGN e AGW) a fim de minimizar o RMSE.

Na estrutura do AG mostrada na Figura 4.4, após a avaliação os indivíduos são selecionados para realizar operações de cruzamento e mutação de acordo com as suas probabilidades chamadas de p_c (probabilidade de cruzamento) e p_m (probabilidade de mutação). O método de seleção torneio é utilizado no AG implementado neste trabalho [40]. Este procedimento foi detalhado na seção 3.1.2.2 do Capítulo 3. O indivíduo com a melhor *fitness* (neste caso, menor RMSE), é selecionado. Note que os indivíduos com maior aptidão têm maior probabilidade de sobreviverem, e essa probabilidade varia de

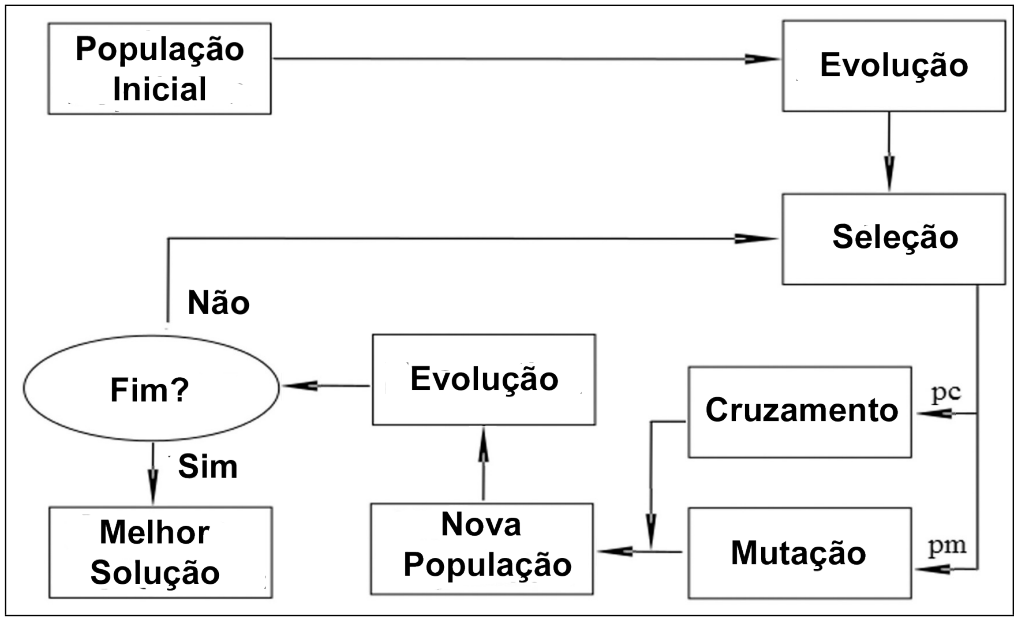


Figura 4.4. Fluxograma do Algoritmo Genético, onde p_c e p_m são as probabilidades de cruzamento e mutação, respectivamente.

acordo com o tamanho de indivíduos (k) do torneio.

Os indivíduos selecionados através de vários torneios consecutivos poderão ser modificados usando o operador de cruzamento ou de mutação, antes de serem inseridos na nova população. O cruzamento utiliza dois indivíduos (pais), e seu principal objetivo é a troca de material entre eles, levando a criação de dois filhos.

A mutação por sua vez, tem como objetivo causar uma pequena modificação aleatória no indivíduo, a fim de deixar uma estagnação local na região de busca (ótimo local). A mutação usada aqui muda o valor de cada parâmetro selecionado aleatoriamente por um valor de +(-) 0,02.

Todos os indivíduos gerados pela Figura 4.4 são inseridos em uma nova população, juntamente com o melhor de todos os indivíduos, o que é preservado de uma geração para a seguinte (procedimento conhecido como elitismo). Este processo continua até que um critério de parada seja atendido, o que geralmente é um erro mínimo ou o número máximo de gerações.

É importante observar que o algoritmo usa um conjunto de parâmetros, que são: número de indivíduos, número de gerações, probabilidades de cruzamento e de mutação e tamanho do torneio. Uma escolha adequada destes parâmetros é fundamental para uma boa cobertura do espaço de busca. O número de indivíduos e gerações são parâmetros altamente dependentes do tamanho do espaço de busca e determinados por problemas ainda pouco estudados. Foi testado o tamanho das populações de 50,

100, 150 e 200, e gerações de 200, 500 e 800. Muitas combinações foram testadas, e os melhores resultados obtidos são com 200 indivíduos e 800 gerações, pois atingiram a convergência esperada.

As probabilidades de cruzamento e mutação têm valores padrões, com altas taxas de cruzamentos e taxas de mutações baixas. Foram testadas probabilidades de cruzamentos 0,9 e 0,95, e as taxas de mutações de 0,01, 0,1, 0,2, 0,25 e 0,3. Todas as combinações possíveis foram testadas, e os melhores resultados obtidos foram 0,95 e 0,3 para cruzamento e mutação, respectivamente. Para o tamanho do torneio, os valores de 8, 10, 20 e 30 foram testados, com 8 apresentaram os melhores resultados. Todos os parâmetros testados foram executados cinco vezes, pois o algoritmo genético é um método heurístico. No final os melhores resultados foram selecionados assumindo os mesmos critérios do parágrafo anterior.

4.2 Modelo do Sistema II

A mesma metodologia da segunda abordagem (AGW) foi utilizada nessa segunda reparametrização realizada na ligação fósforo diéster com o objetivo de melhorar ainda mais a precisão do campo de força. Dessa forma, essa nova parametrização irá focar na eficácia dos parâmetros do termo torcional na torção fósforo diéster para representar o RNA. Devido a metodologia aplicada nessa segunda parametrização ser muito parecida com a primeira, então será destacado apenas nas diferenças entre as duas parametrizações.

Preparação da base de dados Os dados desse sistema foram obtidos no mesmo banco de dados do sistema I, porém ao invés de isolar as bases nós a retiramos e inserimos dois grupos metoxi no lugar delas, pois a ligação glicosídica já teria sido parametrizada (primeira reparametrização), e com isso os cálculos se tornam mais rápidos do que se fossem realizados com as bases do RNA. A inserção dos hidrogênios ocorreu também da mesma forma que no sistema I, porém foi necessário adição de mais hidrogênios com o objetivo de neutralizar a molécula.

Depois da preparação o segundo modelo que servirá de dado de entrada, na Figura 4.1, para os cálculos quânticos é composto por duas unidades de açúcares ligados pelo grupo fosfato (ligação fósforo diéster) e dois grupos metoxi ligados via ligação glicosídica (ver Figura 4.5), simulando um dinucleotídeo.

Como se pode observar na Figura 4.5 a molécula possui seis combinações de ângulos rotacionais (α, β, γ, δ, ε e ζ). Porém, para os nossos testes preliminares apenas o ângulo α foi o escolhido [6]. A diferença em relação à primeira molécula é que

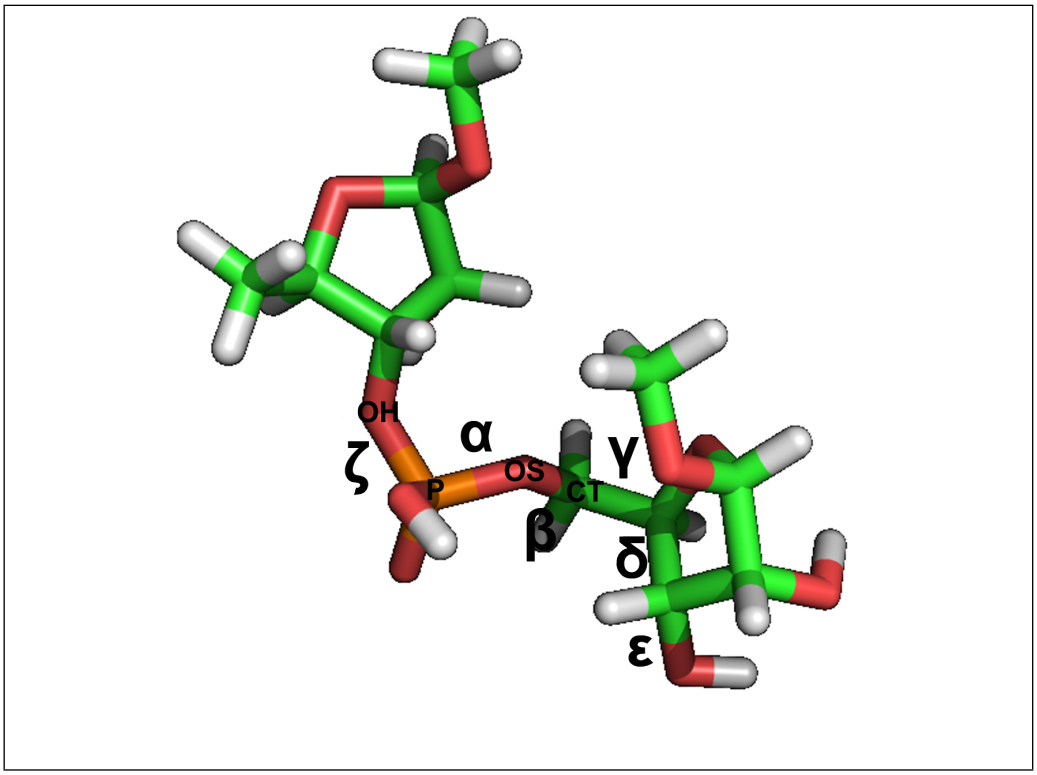


Figura 4.5. Molécula utilizada como entrada. O ângulo torcional parametrizado é definido pelos átomos OH-P-OS-CT (ângulo de torção α da ligação fósforo diéster).

agora vamos reparametrizar dois nucleotídeos ligados por um grupo fosfato, além da ausência dos quatro nucleotídeos, pois nessa nova reparametrização não necessitamos otimizar a ligação glicosídica, visto que já a temos otimizada.

Os métodos para se obter os perfis de torção, realizar os cálculos quânticos e de mecânica molecular são os mesmo realizados para o sistema I, como já foi citado acima.

4.2.1 A Otimização do Termo de Torcional II usando o AG

Para a segunda parametrização o indivíduo do AG é representado por um vetor de 4 posições, como ilustrado na Figura 4.6, representando os termos V_n e γ, como definido na Eq. 2.8, para os dois diferentes períodos da função cosseno.

O mesmo critério usado na primeira otimização foi usada na escolha dessa representação. A tabela de torção [20] informando que para a torção da ligação fósforo diéster existem 2 periodicidades. Dessa forma precisaríamos de 2 barreiras torcionais (V_n) e 2 fases (γ), totalizando 4 variáveis para o cálculo da energia. Como os valores também assumem valores reais achamos mais coerente trabalhar com a representação numérica ao invés da binária.

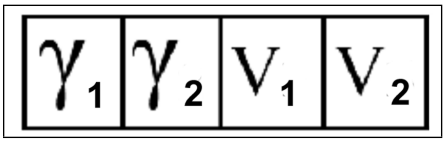


Figura 4.6. Representação do indivíduo do Algoritmo Genético da segunda parametrização.

O mesmo fluxograma do AG foi aplicado para essa molécula com os mesmos métodos e parâmetros de seleção, cruzamento e mutação. Os testes para a sua escolha também foram realizados de maneira similar.

Capítulo 5

Resultados

Nas duas primeiras seções deste capítulo são apresentados os resultados do primeiro modelo e do segundo modelo, descritos no capítulo anterior, cada modelo sendo detalhado em subseções mostrando a otimização dos parâmetros do termo de torção realizado pelo AG nos dois sistemas, as comparações dos resultados obtidos com os novos parâmetros em relação aos dados da literatura e por último os testes dos dois conjuntos de parâmetros encontrados contra um dado experimental (1r4h) obtido do PDB.

5.1 Resultados do Sistema I

5.1.1 A Otimização I

A otimização dos parâmetros de energia de torção é feita através de um algoritmo genético, que utiliza como referências as estruturas e as energias dos nucleosídeos fornecido pelos cálculos quânticos. O AG evolui os parâmetros do termo torcional (ver Eq. 2.8 para mais detalhes) de acordo com a sua função de *fitness* (ver Eq. 4.3).

Na primeira parametrização os valores dos parâmetros torcionais encontrados pelo AG, no AGN e AGW, para as torções glicosídicas (definidas na Figura 4.2) são comparados com aqueles relatados na referência [17] e apresentados na Tabela 5.1.

Como o AG é um método heurístico, ele foi executado 5 vezes (Apêndice B) a fim de mostrar que o algoritmo não encontrou uma solução adequada por acaso. Como a variância dos resultados é pequena, relatamos os parâmetros obtidos com a execução que resultou no menor RMSE entre as diferenças da energia da mecânica quântica (E^{QM}) e da energia da mecânica molecular (E^{MM}) calculadas utilizando as diferentes rotações dos nucleosídeos (Eq. 4.3). O tempo de processamento de cada geração do AG

Tabela 5.1. Parâmetros do termo de torção encontrados pelo Algoritmo Genético

Nucleotídeo	Torção	n^a	AGN		AGW^d		Referência [17]	
			$Vn/2^b$	γ^c	$Vn/2^b$	γ^c	$Vn/2^b$	γ^c
Adenina	OS-CT-N9-C2*	1	0,9310	88,44	0,9660	69,00	0,6956	68,79
		2	1,0650	6,00	0,9465	15,89	1,0740	15,64
		3	2,9778	301,18	0,0597	24,95	0,4575	171,68
		4	1,7601	147,11	4,0000	308,85	0,3092	19,09
Guanina	OS-CT-N9-CK*	1	0,9660	69,00	0,5327	50,60	0,7051	74,76
		2	1,0037	95,35	0,0002	0,80	1,0655	6,23
		3	3,1155	301,80	0,0001	1,96	0,4427	168,65
		4	3,6411	308,70	0,3919	3,91	0,2560	3,97
Citosina	OS-CT-N9-C1*	1	1,6639	158,00	0,0092	0,87	1,2251	146,99
		2	3,1230	301,24	2,9787	277,70	1,6346	16,48
		3	3,5517	308,40	3,5213	278,05	0,9375	185,88
		4	0,1332	12,90	0,1253	10,03	0,3103	32,16
Uracila	OS-CT-N9-CM*	1	1,3911	132,10	1,9634	186,52	1,0251	149,88
		2	1,5847	150,50	1,4182	137,21	1,7488	16,76
		3	1,9126	175,30	1,0670	69,31	0,5815	179,35
		4	1,1374	85,60	1,4675	96,34	0,3515	16,00

[a] Periodicidade da torção.
[b] Constante de força em kcal/mol.
[c] Fase em graus.
* Ver Figura 4.5 para detalhes.
[d] Novas cargas usadas como parâmetros do termo eletrostático serão apresentadas na Tabela 5.2 e no Apêndice A.

é, em média, de 60 segundos para a primeira parametrização. Cada geração avaliou um conjunto de 200 indivíduos (1600 parâmetros diferentes), e 800 gerações são executados para cada nucleosídeo. Assim, um experimento completo é executado por cerca de 13 horas na primeira parametrização usando um processador i7 com 2GB de RAM. Para este primeiro refinamento foram necessários, para a parametrização do AG, realizar 140 testes (28 testes de parâmetros x 5 sementes). Isto correspondeu a 76 dias de cálculos. Enquanto que para realizar um cálculo quântico foi necessário 5 dias usando o CENAPAD-MG (Centro Regional do Sistema Nacional de Processamento de Alto Desempenho) usando um nó com dois processadores quadcore e 32GB de RAM para o processamento. Neste caso, foram realizados 32 cálculos, que corresponderam à aproximadamente 60 dias usando 12 nós para o processamento.

A Figura 5.1 mostra a evolução dos indivíduos ao longo das gerações em termos de RMSE referente à primeira abordagem sem o refinamento dos parâmetros do termo eletrostático (AGN). As linhas cheias mostradas nos gráficos da figura representam o melhor indivíduo e a evolução indicando que houve uma melhora no RMSE ao longo das gerações. Nota-se através da média dos valores do RMSE a convergência do algoritmo, que ocorre próximo da geração de número 500.

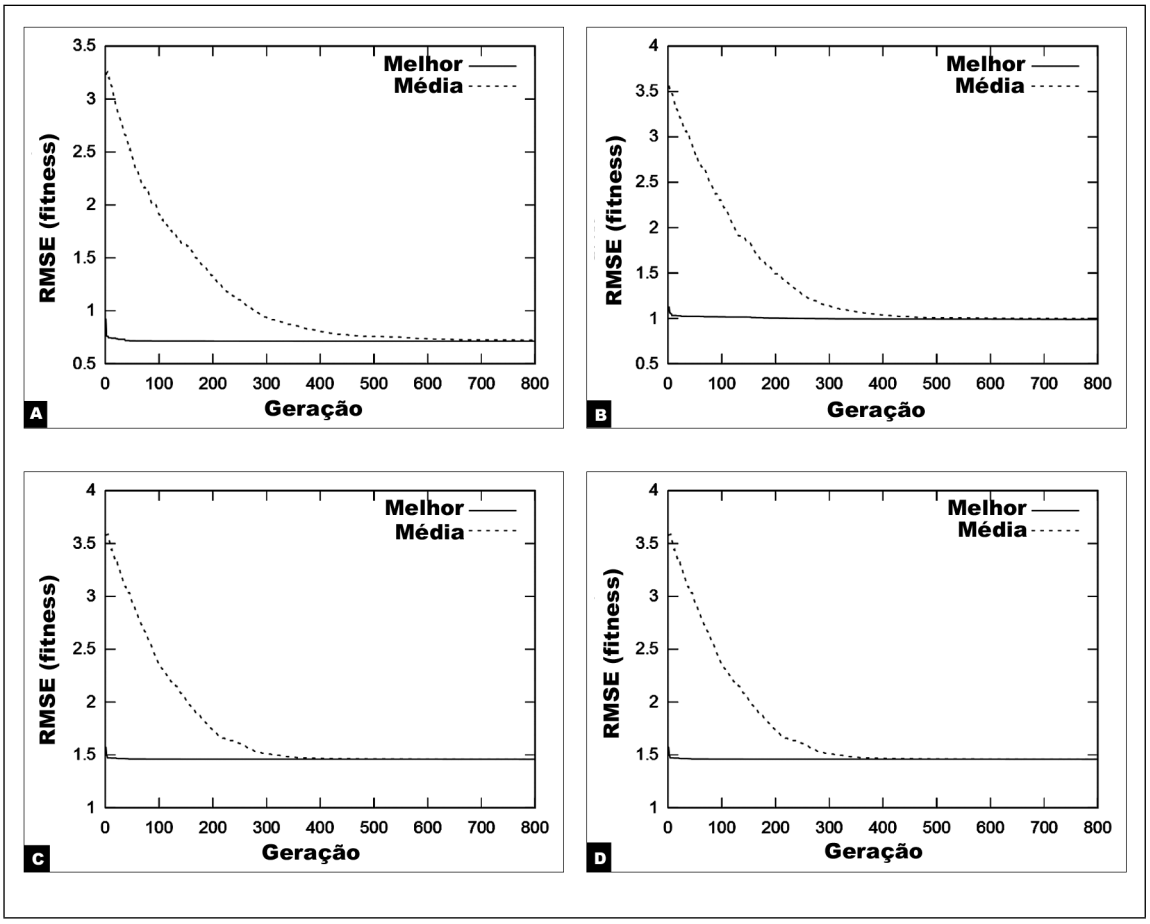


Figura 5.1. RMSE dos indivíduos evoluídos pelo AGN, considerando o melhor dos indivíduos e a média da população: (A) Adenina, (B) Guanina, (C) Citosina e (D) Uracila.

A importância da escolha dos parâmetros pode ser notada na Figura 5.2. Ela mostra os resultados com o parâmetros da seguinte maneira: 200 indivíduos, 600 gerações, probabilidade de cruzamento de 0.95 e mutação de 0.3.

A Figura 5.2 indica que os indivíduos ainda estão evoluindo, podemos notar isso através da linha pontilhada, que indica a média da população. Vemos que ela ainda não estabilizou indicando para o usuário a necessidade de aumentar o número de gerações.

Porém, na Figura 5.3 com os parâmetros corretos (indicados na seção 4.1.4) observa-se a evolução dos indivíduos ao longo das gerações em termos de RMSE referente à primeira parametrização usando o refinamento dos parâmetros do termo eletrostático. As linhas cheias dos gráficos da figura representam o melhor indivíduo e a evolução mostrando que o RMSE melhora ao longo das gerações.

Note que para guanina, a curva pontilhada estabiliza rapidamente, enquanto que para a citosina as melhorias continuam até ao final do processo de pesquisa. O RMSE

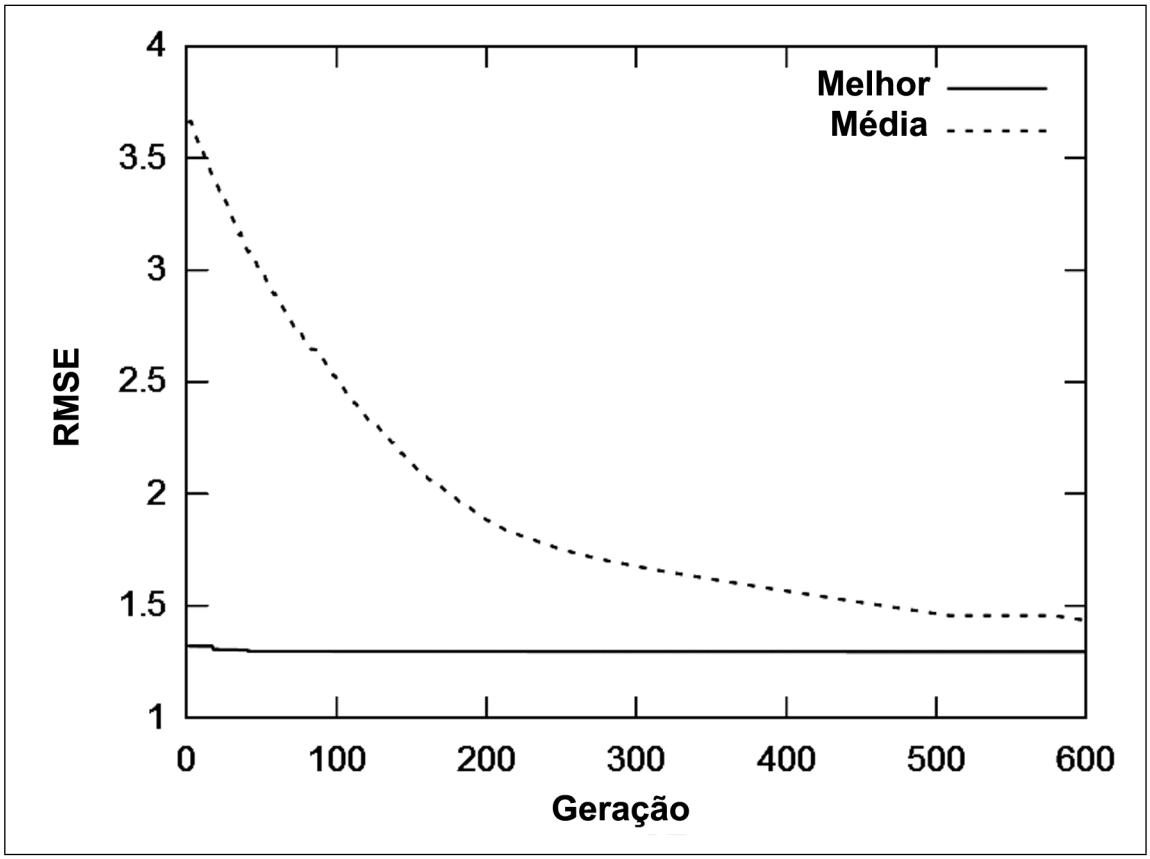


Figura 5.2. RMSE dos indivíduos evoluídos pelo AGW, considerando o melhor dos indivíduos e a média da população para a Adenina

médio é um indicativo de quando a busca converge, pois neste momento todos os indivíduos (que representam o conjunto de parâmetros que estão sendo otimizados) terão valores muito semelhantes à *fitness*. Na presente análise a convergência dos indivíduos começa perto das 300 gerações.

A reparametrização das cargas dos átomos da uracila para o campo de força Amber (ff99χOL) são mostrados na Tabela 5.2 para comparação com as cargas obtidas pelos cálculos quânticos. As tabelas para os outros nucleosídeos serão apresentadas no Apêndice A.

Podemos perceber através da Tabela 5.2 que a maioria dos valores teve uma diferença considerável em relação à nova parametrização. Mais tarde, quando os parâmetros forem validados através da molécula escolhida para este trabalho veremos sua influência.

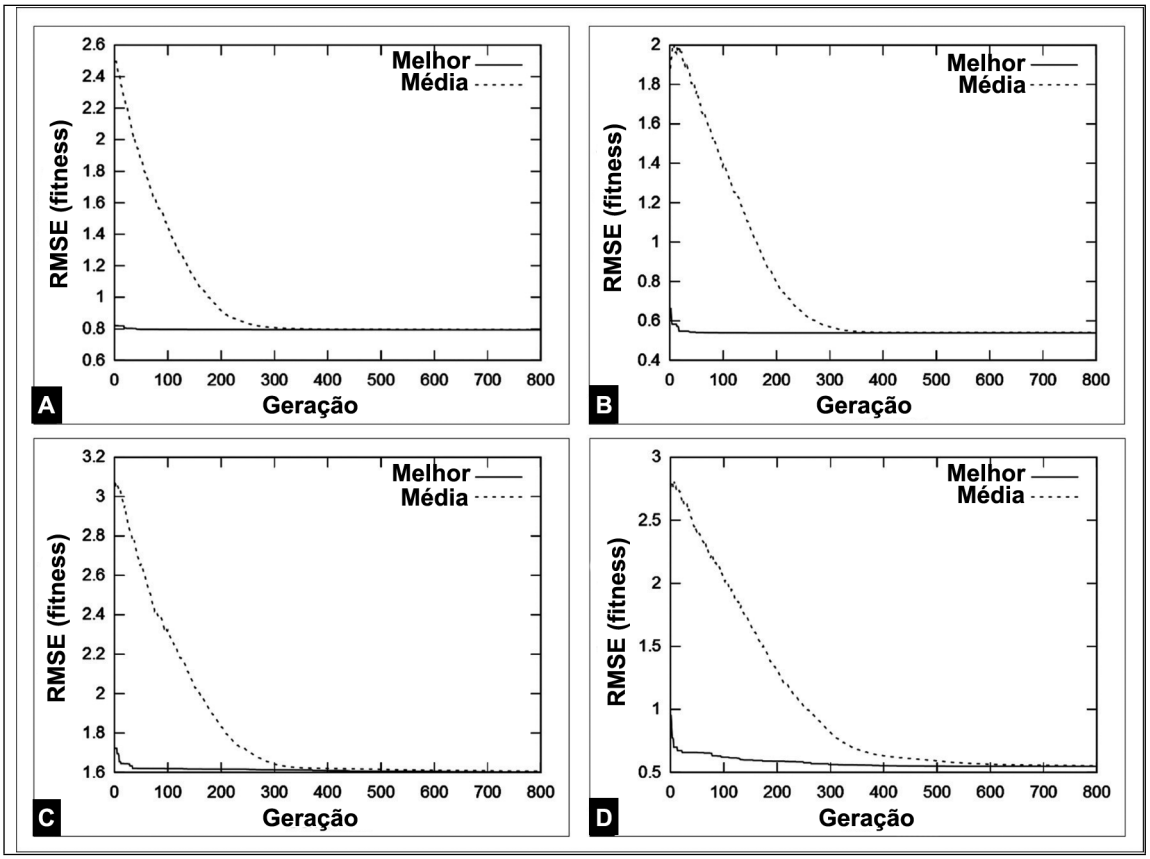


Figura 5.3. RMSE dos indivíduos evoluídos pelo AGW, considerando o melhor dos indivíduos e a média da população: (A) Adenina, (B) Guanina, (C) Citosina e (D) Uracila.

5.1.2 Comparação com a Literatura I

Na Figura 5.4 são analisados os RMSE da primeira parametrização utilizada neste trabalho (AGN e AGW) comparando com os dados da referência [17].

Na Figura 5.4 percebe-se que com o AG fomos capazes de melhorar o RMSE, em aproximadamente, 10% (AGN) e 37% (AGW) na adenina, 27% (AGN) e 67% (AGW) na guanina, 4% (AGN) e 21% (AGW) na citosina e 6% (AGN) e 72% (AGW) na uracila, quando comparado com os resultados obtidos na referência [17]. Estes números mostram que o método de parametrização realizada pelo AGW é mais eficaz, especialmente para a uracila.

A Figura 5.5 mostra que a energia potencial total para todos os nucleosídeos usando o AGW é mais próxima dos resultados quânticos se comparados com a abordagem AGN ou a referência [17]. O estudo atual mostra uma melhoria no cálculo de energia com base na otimização do AG. Nas três comparações foi adotado um procedimento de re-escala, anteriormente usado por Guvench e MacKerell [81] para este tipo

Tabela 5.2. Parâmetros encontrados pelos cálculos quânticos para a energia eletrostática da uracila

Átomo	Cargas Novas	Cargas Antigas	Erro(%)
CT5	0,1405	0,0558	60,28
H12	0,0145	0,0679	78,64
H13	0,0435	0,0679	35,93
CT4	0,1793	0,1065	40,60
H14	0,0165	0,1174	85,94
OS	-0,6410	-0,3548	44,65
CT1	0,6526	0,0674	89,67
H2	0,0024	0,1824	98,68
CT3	0,4826	0,2022	58,10
H15	0,0310	0,0615	49,59
CT2	0,0874	0,0670	23,34
H16	-0,0002	0,0972	100,20
OH2	-0,7209	-0,6139	14,84
HO2	0,3327	0,4186	20,52
N9	-0,6850	0,0418	106,10
CC1	0,7428	0,4687	36,90
NA	-0,4510	-0,3549	21,31
CC2	1,0673	0,5952	99,99
CM1	-0,5025	-0,3635	27,66
CM2	0,4332	-0,1126	125,99
OO1	-0,529	-0,5477	3,41
HH	0,2065	0,3154	34,53
OO2	-0,5428	-0,5761	5,78
HA	-0,0699	0,1811	138,60
H4	0,0182	0,2188	91,68
Erro Médio			7,83

de análise, a fim de comparar dados da literatura e os cálculos realizados neste trabalho contra os resultados quânticos que são os dados de referência.

Como já foi discutido, os melhores resultados obtidos pela metodologia (AGW) são os da uracila. Observando as curvas da Figura 5.5 (D), o comportamento da função AGW é mais próximo da E^{QM}. A parametrização mostrada na referência [17] tem demonstrado um alto nível de precisão. No entanto, a nossa nova abordagem demonstra uma melhora de tal parametrização. A consequência dos pequenos desvios geralmente observadas em campo de força é apontada por Banás *et. al.* [18] como justificativa para a subestimação das interações entre o pareamento das bases. Como mostrado na Figuras 5.5 a metodologia proposta reduziu este problema, tornando a estrutura dos nucleosídeos mais próxima dos dados assumidos como referência (cálculos quânticos).

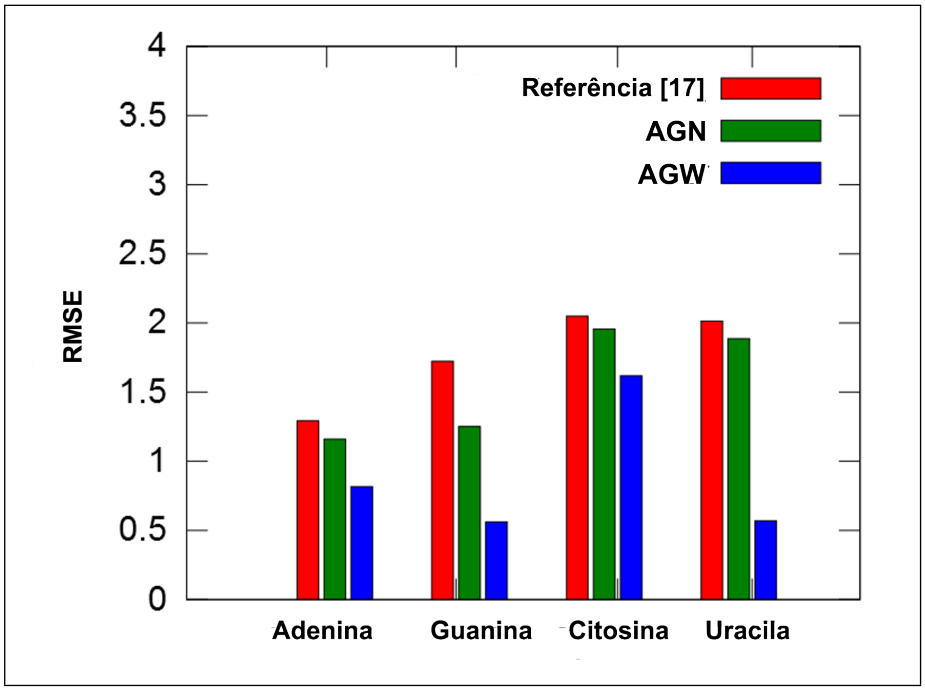


Figura 5.4. Análise do RMSE da referência [17] vs. RMSE encontrados pelas abordagens propostas (AGN e AGW).

Na Tabela 5.3 é apresentada a energia clássica dos termos do campo de força (Amber), com os parâmetros da referência [17] vs. os parâmetros obtidos pelo AGN. Como os valores de alguns termos do campo de força se repetem, foi atribuída uma variável (S) que irá assumir a soma desses valores. Nela podemos perceber que apesar de apresentarmos um RMSE menor (objetivo do AG), na energia total não houve uma diminuição. Isso pode ser melhorado com uma parametrização melhor usando outros termos junto com o de torção. As energias dos outros nucleosídeos sem a parametrização do termo eletrostático podem ser vistas no Apêndice C.

A Tabela 5.4 apresenta a energia clássica usando os parâmetros obtidos pelo AGW dos termos do campo de força (ff99χOL) comparados com a energia obtida com os parâmetros da referência [17]. Como os valores de alguns termos do campo de força se repetem, foi atribuída uma variável (S) que irá assumir a soma desses valores. Nela podemos perceber a diminuição do valor total da energia de cada rotação. Isso pode ser uma indicação de que a molécula se apresenta numa conformação mais estável do que a anterior. Isso se deve graças a boa parametrização do termo eletrostático. As energias dos outros nucleosídeos podem ser vistas no Apêndice C.

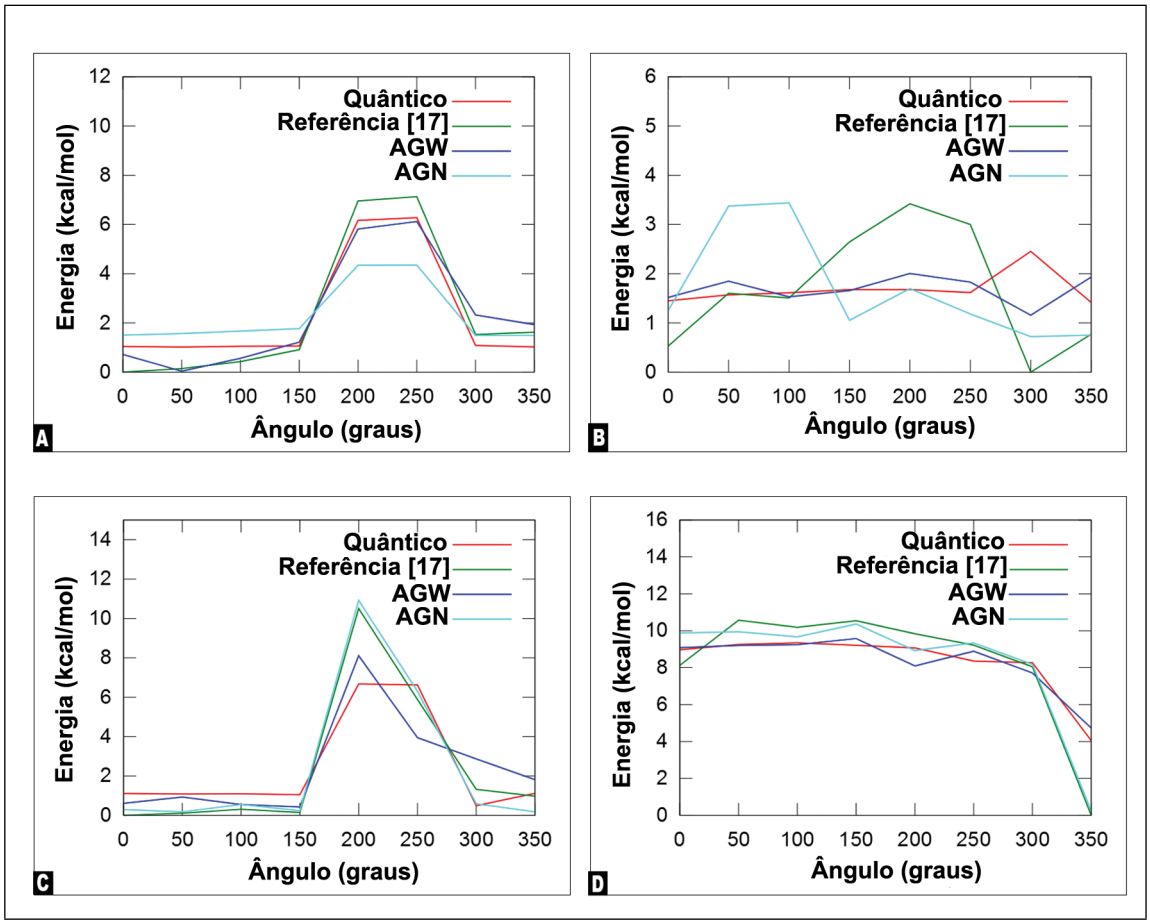


Figura 5.5. Comparação da E^{QM} com a E^{MM} previamente reportado na referência [17] e a E^{MM} obtida pelas abordagens (AGN e AGW). Energias da (A) Adenina, (B) Guanina, (C) Citosina, (D) Uracila.

5.1.3 Simulação do RNA usando a Primeira Parametrização

Para melhor validar o processo proposto neste trabalho considerando o processo de refinamento de parâmetros de campo de força, nós selecionamos o RNA 1r4h do PDB (Protein Data Bank) [82] para mostrar que os parâmetros podem ser aplicados para otimizar a representação da estrutura do RNA. O RNA 1r4h foi obtido utilizando ressonância magnética nuclear (RMN) e contém dez nucleotídeos. Foi usado na análise estrutural do domínio IIIc de vírus humano GB. Devido ao seu tamanho relativamente pequeno, esta estrutura fornece uma simulação de computador razoável sem dificuldades para o teste. Além disso, se o método proposto da utilização do AG para refinar os parâmetros do potencial de torção é eficaz (por exemplo, uma molécula diferente que não foi considerada no processo de refinamento), então acreditamos que a aplicação deste método a outros termos (Eq. 2.13) pode melhorar ainda mais o campo de força, pelo menos para os ácidos nucleicos dos RNAs.

Tabela 5.3. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGN) para a uracila

	Rotação*	S	E_{tor}	Total
Ref.[17]	0	-33,4077	18,9235	-14,4842
	50	-31,8592	19,7931	-12,0661
	100	-32,5304	20,0767	-12,4537
	150	-31,4593	19,3550	-12,1043
	200	-32,4361	19,6634	-12,7727
	250	-33,4946	20,1066	-13,3880
	300	-34,0675	19,5077	-14,5598
	350	-40,8726	18,2655	-22,6071
AGN	0	-33,4077	21,3822	-12,0255
	50	-31,8592	19,8735	-11,9857
	100	-32,5304	20,2668	-12,2636
	150	-31,4593	19,8827	-11,5766
	200	-32,4361	19,4589	-12,9772
	250	-33,4946	20,9347	-12,5599
	300	-34,0675	20,3468	-13,7207
	350	-40,8726	19,1983	-21,6743

* Ângulo em graus.
Todas as energias em kcal/mol.

Foi comparada a estrutura obtida usando os parâmetros relatados na referência [17] e os resultados calculados com dados experimentais. O Tinker foi utilizado para realizar os cálculos de mecânica molecular para ambos os conjuntos de parâmetros.

Preparação da molécula 1r4h Para simular condições semelhantes às aplicadas no experimento 1r4h, uma caixa de água foi usada para solvatar molécula de dimensões 36 x 36 x 36 Å com 1000 moléculas de água para um 1r4h. A simulação foi realizada utilizando o software DICE [83], e o total de cargas foi neutralizada com íons Na+ [84]. Foram determinados 300 mil passos para o sistema chegar ao equilibrio, as constantes de carga (E_{ele}), ϵ e σ (E_{wdv}) que são usadas no DICE foram retiradas do campo de força reparametrizado apresentado neste seção. Esses mesmos passos foram adotados para a segunda parametrização que será mostrada na secção 5.2.3. A configuração inicial da caixa é representada na Figura 5.6.

Em seguida, foram realizadas pequenas rotações aleatórias (menor que 5 graus) em partes também aleatórias do RNA. Isso foi feito apenas para retirá-lo de sua posição de equilíbrio, com o objetivo de ter uma nova estrutura que será otimizada pelo TINKER usando os parâmetros da referência [17] e os encontrados neste trabalho para

Tabela 5.4. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGW) para a uracila

	Rotação*	S	E_{tor}	E_{ele}	Total
Ref.[17]	0	13,6030	18,9235	-47,0107	-14,4842
	50	13,5837	19,7931	-45,4429	-12,0661
	100	13,3924	20,0767	-45,9228	-12,4537
	150	13,9389	19,3550	-45,3982	-12,1043
	200	13,7800	19,6634	-46,2161	-12,7727
	250	14,2668	20,1066	-47,7614	-13,3880
	300	14,1993	19,5077	-48,2668	-14,5598
	350	17,4758	18,2655	-58,3484	-22,6071
AGW	0	13,6030	20,2535	-197,0068	-163,1503
	50	13,5837	22,2428	-198,8853	-163,0588
	100	13,3924	22,6244	-199,0380	-163,0212
	150	13,9389	22,1233	-198,7557	-162,6935
	200	13,7800	18,3129	-196,2379	-164,1450
	250	14,2668	20,3679	-197,9765	-163,3418
	300	14,1993	19,6024	-198,3236	-164,5219
	350	17,4758	19,2520	-204,2356	-167,5078

* Ângulo em graus.
Todas as energias em kcal/mol.

o campo de força Amber (ff99χOL).

Um ponto fundamental para fazer as comparações descritas acima é calcular uma medida adequada de semelhança entre as diferentes estruturas [85]. Nesse trabalho foi escolhido o RMSD (Root Mean Deviation Square), uma medida amplamente utilizada na literatura para representar o desvio quadrático médio dos átomos. O RMSD é escrito como:

$$RMSD = \sqrt{\sum_{n}^{i=1} \frac{(x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2}{n}}, \tag{5.1}$$

onde x, y e z são as coordenadas dos dados experimentais, x', y' e z' são as coordenadas encontradas pelo Tinker e n define o número de átomos na molécula. Como apontado na referência [86], RSMD é um bom indicador da precisão das coordenadas atômicas. Quanto menor o valor, melhor é a precisão.

Os resultados de RMSD considerando a estrutura gerada com os parâmetros do AG (AGN e AGW) e da referência [17] são comparados na Tabela 5.5. Nota-se que houve uma pequena melhora no RMSD. Usando o AGW observa-se que o RMSD é reduzido em relação à publicação anterior [17]. Embora, Zgarbová *et. al.* [17] tenham

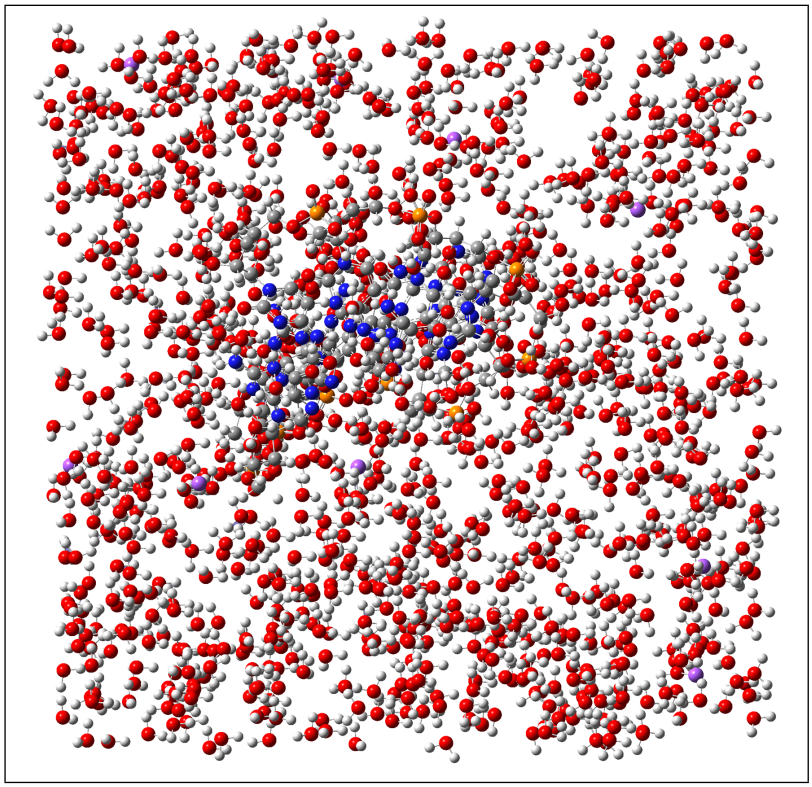


Figura 5.6. Representação da caixa de água para a molécula 1r4h.

um bom valor de RMSD, fomos capazes de melhorar este valor em 13% com o AGN e 78% com o AGW.

Tabela 5.5. Comparação usando o RMSD entre as metodologias aplicadas (AGN e AGW) e o dado experimental

	$RMSD^a$
Referência [17]	1,50
AGN	1,30
AGW	0,33

[a] Valores em Angstrom.

Observa-se na Figura 5.7 onde o termo eletrostático não é levado em consideração no processo de parametrização (AGN - em vermelho), se obtém estrutura da molécula 1r4h próxima dos dados experimentais obtidos do PDB (verde), porém com alguns erros. Isto pode estar relacionado, principalmente, devido às ligações de hidrogênio e interações hidrofóbicas, que são fundamentais para a estabilidade da hélice [18]. Como esperado, as estruturas são bastante afetadas pelas interações entre o emparelhamento das bases.

Porém, na Figura 5.8 que quando o termo eletrostático é levado em consideração

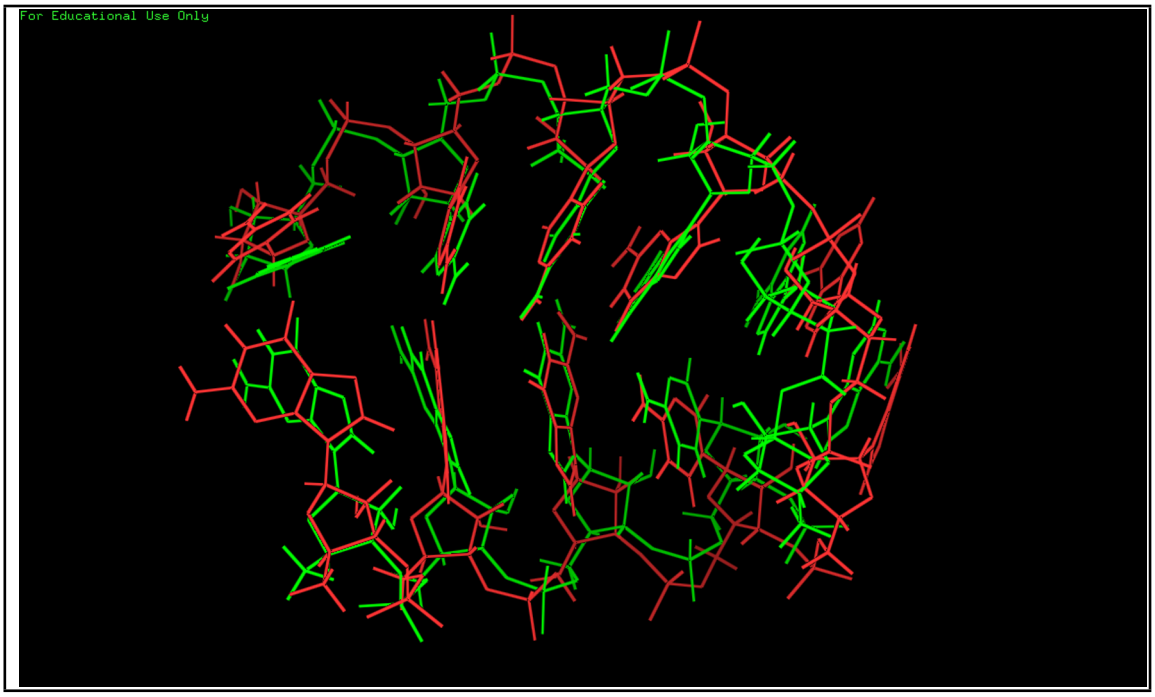


Figura 5.7. Comparação entre a estrutura do RNA 1r4h encontrados pela metodologia proposta AGN (vermelho) e o dado experimental (verde).

no processo de parametrização (AGW - em vermelho), se obtém estrutura da molécula 1r4h próxima dos dados experimentais obtidos do PDB (verde), nota-se que a estrutura está mais precisa se comparada com o dado experimental do PDB. Isto acontece, pois o bom ajuste que ocorreu nas cargas do termo eletrostático ajudou num melhor refinamento do termo torcional e numa melhor representação das ligações de hidrogênio e interações hidrofóbicas mostrando assim a importância delas na estabilização da molécula, como já citado acima.

5.1.3.1 O Refinamento Final

Como observado na Figura 5.8, quase toda a estrutura está bem predita quando comparado com o RNA 1r4h do PDB. Entretanto, o nucleotídeo da extremidade esquerda da molécula é aquele que apresenta o maior erro nesta fase da parametrização, em especial o oxigênio (OS) do açúcar do nucleotídeo guanina (marcadas com um quadrado na Figura 5.8). A ligação dele com os carbonos não foi bem otimizada, obtendo um erro total de 0,33 Åem comparação ao dado experimental do PDB. Então surge um questionamento: Por que somente a superposição do açúcar não é boa o bastante?

Para responder essa pergunta precisamos analisar mais profundamente a natureza do erro na conformação de oxigênio. A Figura 5.9 mostra em linhas pontilhada as

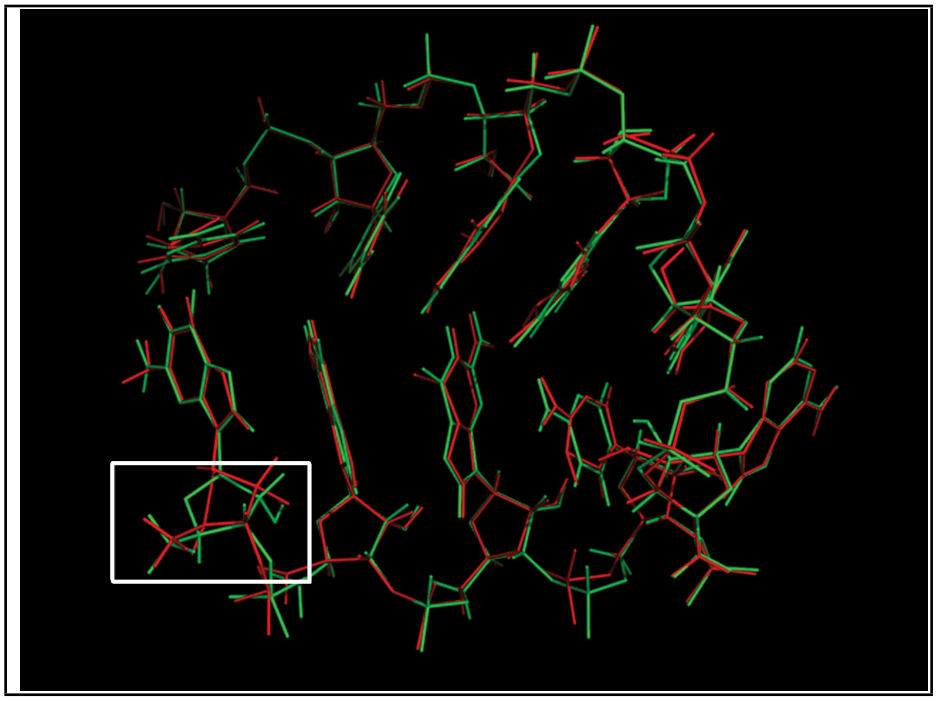


Figura 5.8. Comparação entre a estrutura do RNA 1r4h encontrada pela metodologia proposta AGW (vermelho) e o dado experimental (verde).

ligações de hidrogênio para a molécula em questão e percebe-se que o nucleotídeo isolado (Figura 5.10) não possui a ligação de hidrogênio que é o que aparentemente está estabilizando o oxigênio do açúcar das outras guaninas. Porém, foi observado ainda que para os outros nucleotídeos que não tem a ligação de hidrogênio isso não acontece, ex. a citosina (do outro lado da molécula, que é o par da guanina. Concluímos que o problema encontra-se na barreira torcional (V_n, parâmetro dado pelo AG) para a guanina, que está muito baixo (em média 0,2312 kcal/mol), abaixo da literatura (em média 0,6173 kcal/mol) (ver Tabela 5.1). A maior diferença nos nossos cálculos estão nos V_{n_2} e V_{n_3}. Para manter o átomo de oxigênio em sua posição correta precisamos de um valor maior. Este problema pode ser resolvido através da alteração desse parâmetro no termo de torção do campo de força Amber.

Em vez de executar o algoritmo genético novamente para melhorar os valores deste parâmetro específico, aproveitamos o fato de que o AG trabalha com múltiplas soluções simultaneamente, e buscamos um novo indivíduo na população final levando em consideração o balanceamento entre o RMSE e o V_n, pois o RMSE cresce junto com o valor de V_n. O objetivo é encontrar um indivíduo com um RMSE próximo do primeiro, porém com um V_n maior.

Baseado nos critérios citados acima foi usado o terceiro indivíduo, que em termos de RMSE em comparação com a população final oferece um RMSE de cerca de 0,02

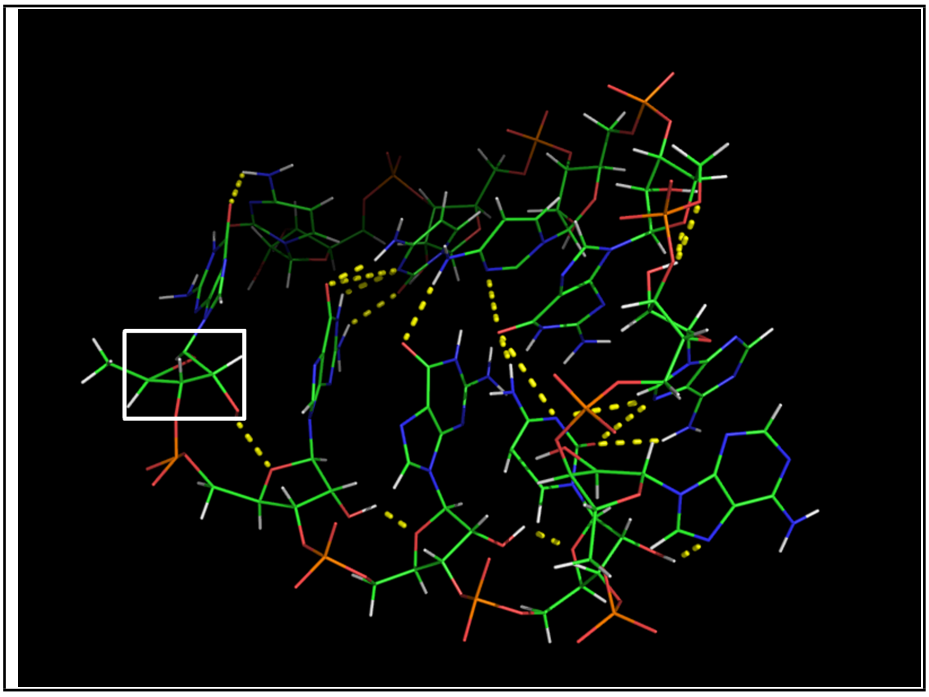


Figura 5.9. A ligação de Hidrogênio da molécula 1r4h representada pelas linhas pontilhadas.

kcal/mol maior do que o conjunto de parâmetros originais e os valores de V_n são, em média, 0,33 kcal/mol maior. Estes novos parâmetros são apresentados na Tabela 5.6. Como observado, com esses novos dados (todos os V_n mudaram) e a nova média dos V_{ns} (0,5042 kcal/mol) é mais próxima dos dados da literatura [17] (0,6173 kcal/mol).

Tabela 5.6. Novos parâmetros selecionados pelo AG para a guanina

Nucleotídeo	Torção	n^a	AGW-Novo		AGW-Velho	
			$V_n/2^b$	γ^c	$V_n/2^b$	γ^c
Guanina	OS-CT-N9-CK*	1	1,1954	113,57	0,5327	50,60
		2	0,0996	5,76	0,0002	0,80
		3	0,0004	13,11	0,0001	1,96
		4	0,7215	4,27	0,3919	3,91

[a] Periodicidade da torção.
[b] Valor da torção em kcal/mol.
[c] Fase em graus.
* Ver Figura 4.2 para mais detalhes.

Como se pode notar com mais detalhes nas Figuras 5.10 que a guanina isolada é mais precisa se comparada com o dado experimental e que o átomo de oxigênio está agora em sua posição esperada, independentemente da ligação de hidrogênio.

A alta precisão no alinhamento se reflete nos valores de RMSD mostrados na

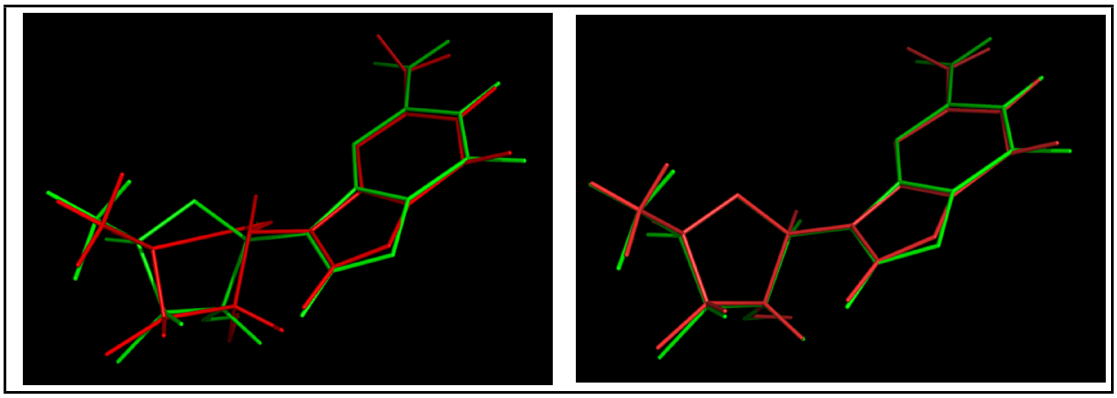


Figura 5.10. Comparação das guaninas isoladas após o refinamento final. (A) Alinhamento da Guanina destacada da molécula do RNA 1r4h (Figure 5.8) e (B) Alinhamento da Guanina obtida com o AGW-Novo.

Tabela 5.7. Note que os valores foram melhorados ainda mais. Este resultado final apresenta uma melhoria de cerca de 82% (mais de 1 Å) usando AGW-Novo em comparação com os resultados obtidos na referência [17].

Tabela 5.7. Análise usando o RMSD para a molécula 1r4h com os novos parâmetros da guanina

	$RMSD^a$
Referência [17]	1,50
AG-Velho	0,33
AG-Novo	0,27

[a] Valores em Angstrom.

A nova estrutura da molécula 1r4h comparada com dados experimentais (verde) é mostrada na Figura 5.11. Ela compara a estrutura obtida com os parâmetros da referência [17] contra o dado do PDB (Figura 5.11A) e a estrutura obtida com parâmetros encontrados pelo algoritmo genético contra o dado do PDB (Figura 5.11B). Dessa forma observamos que o alinhamento usando a estrutura obtida com os dados do AG está mais próxima dos dados experimentais que a estrutura obtida com os dados da referência [17]. Observa-se ainda que os anéis das bases de nucleotídeos são melhores representados do que o esqueleto estrutura considerando o refinamento final. Isto acontece porque o algoritmo foca a sua otimização nas ligações glicosídicas, melhorando o valor RMSD no que diz respeito aos dados experimentais com estruturas geradas por computador e reduzindo o valor do RMSD de 1,5 à 0,27 Å. A partir de agora a abordagem AGW-Novo será chamada apenas de AGW.

Podemos notar na figura que obtivemos maior sucesso na superposição das extremidades da molécula, comparando a metodologia proposta (em vermelho) e os dados

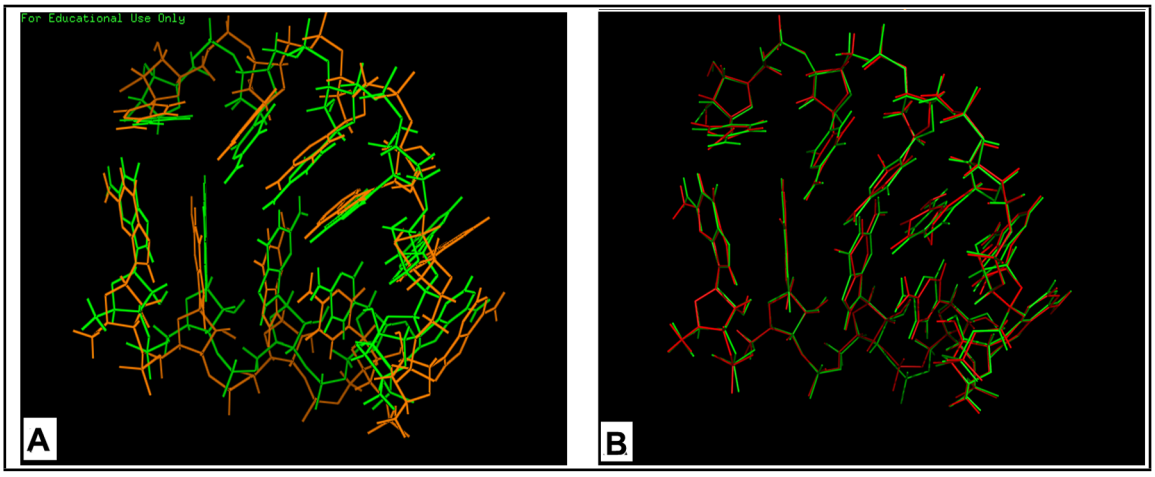


Figura 5.11. Comparação das estruturas encontradas pelos dois refinamentos do campo de força e o dado experimental. (A) Otimização gerada com os parâmetros encontrados na referência [17] (laranja) e o dado experimental (verde). (B) Otimização gerada com os parâmetros encontrados pelo AGW (vermelho) e o dado experimental (verde).

da referência (em laranja) contra o dados experimental (em verde). Esse resultado pode até ser comparado como resultado muito próximo do exato (dado experimental). Percebe-se que o local onde a metodologia proposta apresenta maior contradição com o dado experimental é no dinucleotídeo GC próximo do "grampo", porém se comparado com a referência [17] a metodologia ainda se encontra bem mais próxima do dado experimental, como podemos ver na Figura 5.12 destacado com um quadrado.

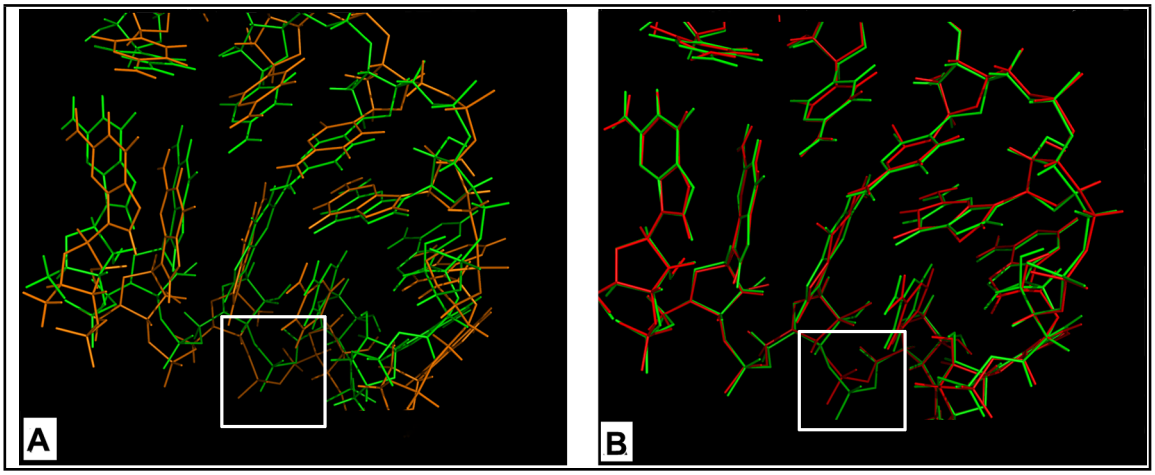


Figura 5.12. Comparação entre a metodologia e a referência [17] apenas da estrutura próxima do grampo da molécula (Figura 5.11 com zoom), onde verde é o dado experimental, laranja é a referência (A) e vermelho é a metodologia AGW (B).

O erro mostrado na região citada acima indica que ainda podemos melhorar esse refinamento. Sendo nescessário um estudo mais detalhado para saber o por que e como diminuir essa imprecisão.

5.2 Resultados do Sistema II

5.2.1 A Otimização II

Na segunda parametrização, o AG otimiza os dois diferentes períodos da função cosseno que descreve a energia de torção para a ligação fósforo diéster, otimizando a barreira rotacional V_n e o ângulo de fase γ (ver Eq. 2.8 para mais detalhes).

Os parâmetros são refinados usando a metodologia da segunda abordagem (AGW) desenvolvida na primeira otimização (seção 5.1.1), onde a avaliação é feita de acordo com o RMSE (veja Eq. 4.3 para mais detalhes). A diferença para a otimização I é que o campo de força Amber (ff99χOL) utiliza os parâmetros da parametrização anterior e os parâmetros obtidos pela AG para a nova molécula (Figura 4.5).

A Tabela 5.8 mostra os parâmetros encontrados na segunda parametrização comparados com os da referência [20].

Tabela 5.8. Parâmetros do termo de torção encontrados pela metodologia AGW

Torção	n^a	AGW-Novo		Referência [20]	
		$V_n/2^b$	γ^c	$V_n/2^b$	γ^c
OS-P-OS-CT*	2	0,0136	0,46	0,25	0,0
	3	0,0316	0,08	1,20	0,0

[a] Periodicidade da torção.
[b] Constante de força em kcal/mol.
[c] Fase em graus.
* Ver Figura 4.5 para detalhes

A reparametrização das cargas dos átomos da molécula para o campo de força Amber (ff99χOL) são mostrados na Tabela 5.9 para comparação. Esses dados nos revelam que apenas algumas cargas foram reparametrizadas. Essa escolha foi feita depois de se realizar comparações entre a parametrização de todos os átomos e a parametrização de apenas alguns. A melhor opção foi manter as cargas da primeira parametrização e reparametrizar apenas o grupo fosfato, pois a primeira parametrização estava mais completa (açúcar e base do nucleosídeo).

Na Tabela 5.9 percebemos que não houve uma grande mudança nos valores. Porém, como optamos por trocar a base do nucleotídeo por um grupo metoxi, foi necessário parametrizar dez novos átomos, pois o grupo metoxi não é parametrizado pelo

Tabela 5.9. Parâmetros encontrados pelo cálculos quânticos para a energia eletrostática

Átomo	Cargas Novas	Cargas Antigas	Erro(%)
OH	-0,6835	-0,6223	8,95
HO	0,4194	0,4295	2,35
OSL	-0,574	-0,6328	9,29
P	1,3461	0,9000	33,14
O2	-0,5979	-0,8200	27,08
OSM1	-0,6205	Não parametrizado	
OSM2	-0,6098	Não parametrizado	
CTM1	0,4516	Não parametrizado	
CTM2	0,4202	Não parametrizado	
H1M1	-0,0406	Não parametrizado	
H1M2	-0,0258	Não parametrizado	
H1M3	-0,0330	Não parametrizado	
H1M4	-0,0385	Não parametrizado	
H1M5	-0,0105	Não parametrizado	
H1M6	-0,0195	Não parametrizado	
hline Erro Médio			16,16

Amber e precisamos desse grupo para fazer a comparação da nova reparametrização e da antiga contra os cálculos quânticos.

A execução do algoritmo também funciona da mesma forma da primeira otimização, onde ele é executado 5 vezes (Apêndice B), pois é um método heurístico. Nessa otimização também buscou-se o menor RMSE entre as diferenças de energia quântica (E^{QM}) e energia mecânica molecular (E^{MM}) calculadas através das diferentes rotações dos nucleosídeos (Eq. 4.3). O tempo de processamento de cada geração do AG é, aproximadamente, igual ao da primeira parametrização. Cada geração avaliou um conjunto de 200 indivíduos (800 parâmetros diferentes), e 800 gerações são executadas.

O tempo de processamento de cada geração do AG é, em média, de 60 segundos para a primeira parametrização. Cada geração avaliou um conjunto de 200 indivíduos (cerca de 800 parâmetros diferentes), e 800 gerações são executados para cada nucleosídeo. Assim, um experimento completo é executado por cerca de 10 horas na primeira parametrização com um processador i7 com 2GB de RAM. Para este primeiro refinamento foram necessários, para a parametrização do AG, realizar 140 testes (28 testes de parâmetros x 5 sementes) correspondendo a 58 dias de cálculos. O tempo de duração de apenas um cálculo quântico no CENAPAD-RS (Centro Regional do Sistema Nacional de Processamento de Alto Desempenho) foi de aproximadamente 15 dias usando apenas um nó com dois processadores dodeca-core de 42GB de RAM. Tanto para o primeiro quanto para o segundo modelo foram necessários 8 cálculos correspondendo a

120 dias.

Na Figura 5.13 temos uma demostração de como a escolha errada dos parâmetros faz com que o AG não encontra a melhor solução.

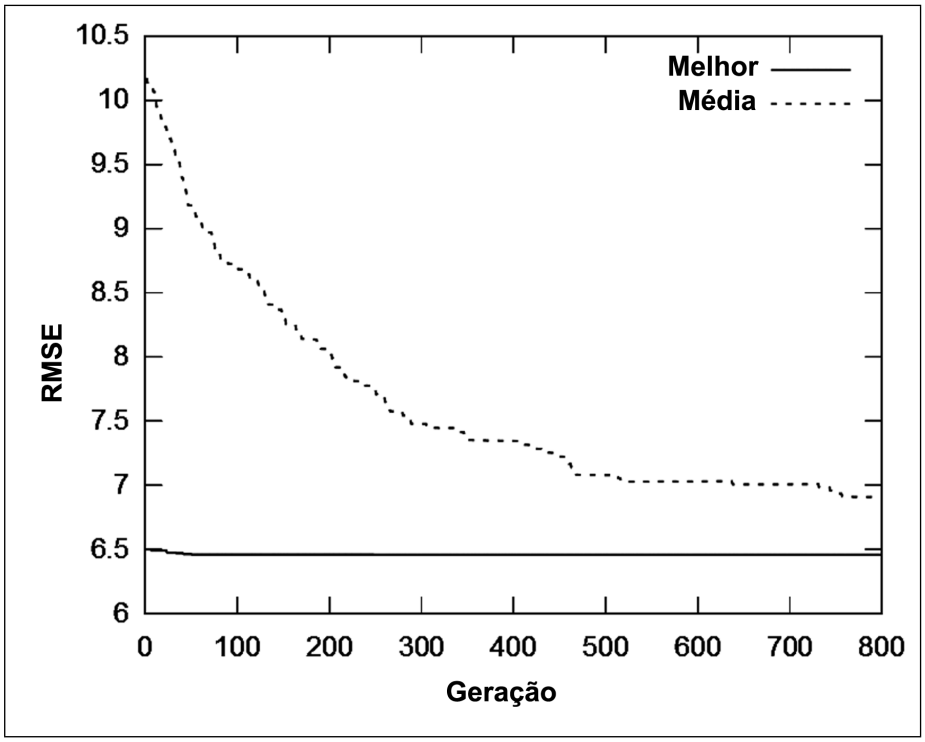


Figura 5.13. Análise do RMSE (*fitness* do AG) considerando o melhor indivíduo e a média da população com taxa de mutação 0,01 para a molécula em estudo.

A Figura 5.13 mostra a evolução dos indivíduos ao longo das gerações usando o AG com os seguintes parâmetros: 200 indivíduos, 800 gerações, torneio de 8 indivíduos, probabilidade de cruzamento de 0.95 e mutação de 0.01. Podemos perceber que a linha pontilhada que representa a média da população não esta próxima do melhor o indivíduo estabilizando antes, isso indica que provavelmente a convergência ficou presa em um ótimo local. Isso pode ser explicado pela baixa taxa de mutação.

Agora na Figura 5.14 mostra a evolução dos indivíduos ao longo das gerações usando o AG com os seguintes parâmetros: 200 indivíduos, 800 gerações, torneiro de 8 indivíduos, probabilidade de curzamento de 0.95 e mutação de 0.3, como foram especificados na seção 4.1.4. As linhas cheias mostradas nos gráficos da figura representam o melhor indivíduo e a evolução mostra que o RMSE melhorou ao longo das gerações.

Na segunda parametrização percebemos que a convergência é mais lenta em relação à primeira. Através do RMSE médio pode-se notar a convergência da pesquisa, visto que neste momento todos os indivíduos terão valores muito semelhantes das *fitness*. Na presente análise a convergência dos indivíduos começa perto das 500 gerações.

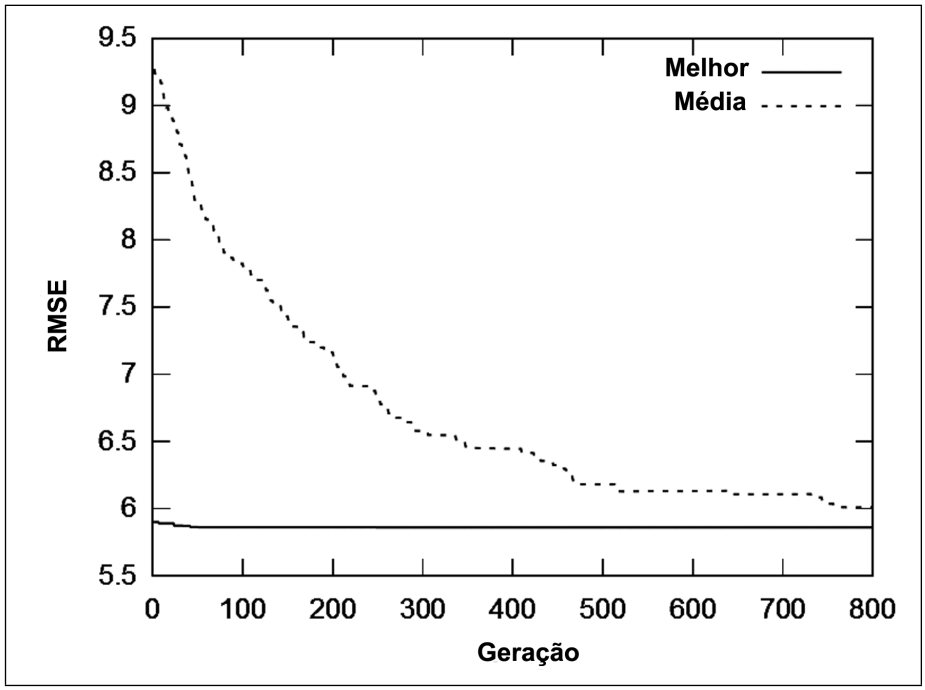


Figura 5.14. Análise do RMSE (*fitness* do AG) considerando o melhor indivíduo e a média da população com taxa de mutação 0,3 para a molécula em estudo.

Na geração 750 ainda houve uma mudança, porém muito pequena e não influenciou na evolução do melhor indivíduo.

5.2.2 Comparação com a Literatura II

A Figura 5.15 mostra o RMSE obtido com a segunda parametrização comparado com o da referência [20]. Fomos capazes de melhorar o RMSE, em relação à molécula em questão de 69%, quando comparado com os resultados obtidos na referência [20]. Estes números mostram que o método de parametrização realizada pelo AGW também pode ser considerado eficaz para este caso.

Podemos perceber através da Figura 5.16 que a energia potencial total para a molécula contendo o grupo fosfato (segunda abordagem - AGW) está mais próxima dos resultados quânticos. A mesma re-escala utilizada na seção 5.1.2 é usada aqui, com a finalidade de comparar dados da literatura e os cálculos realizados neste trabalho contra os resultados quânticos (dados usados como referência).

Observando as curvas da Figura 5.16, o comportamento da função AGW é mais próximo da E^{QM}. A parametrização obtida com referência [17] tem demonstrado um alto nível de precisão. No entanto, a nossa nova abordagem demonstra uma melhora de tal parametrização. Como mostrado na Figura 5.16 a metodologia proposta nesta seção

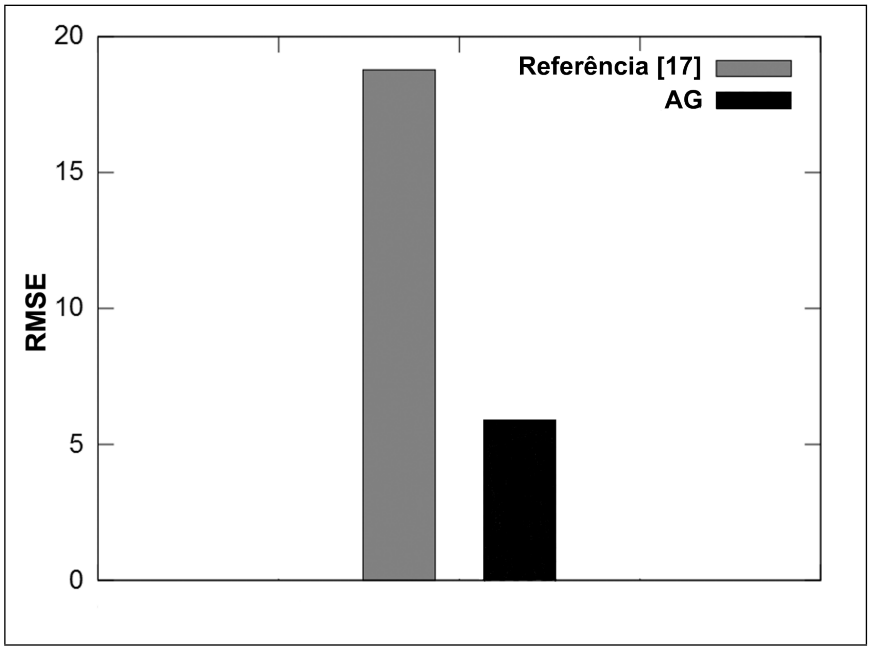


Figura 5.15. Análise do RMSE da referência [17] vs. RMSE encontrado pelo AGW.

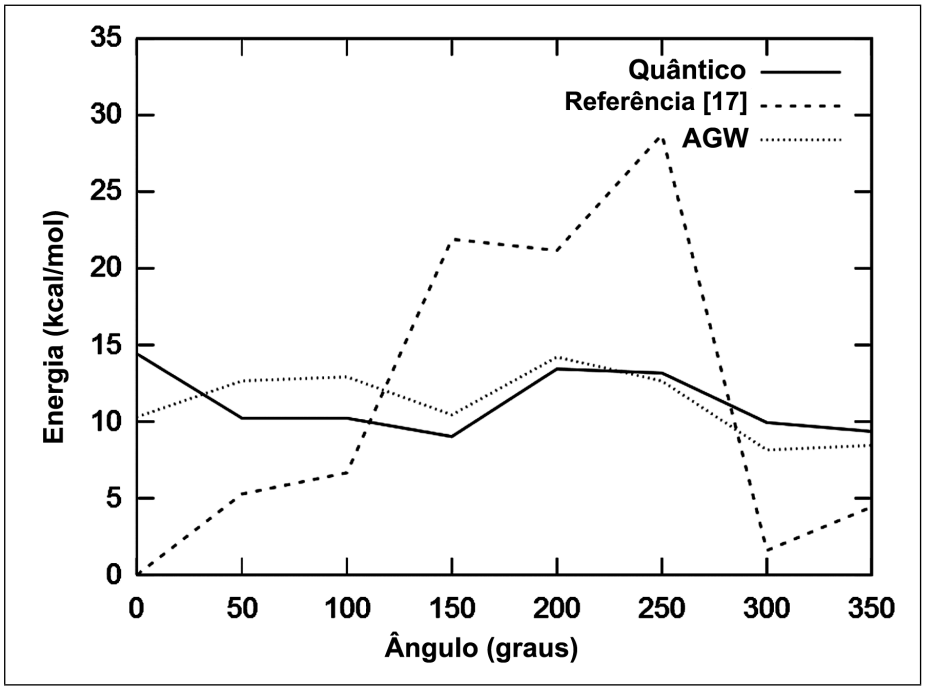


Figura 5.16. Comparação da E^{QM} com a E^{MM} obtida da referência [17] e a E^{MM} obtida pela metodologia AGW proposta.

também reduziu este problema, tornando a estrutura dos nucleosídeos mais próxima dos dados considerados como referência (cálculos quânticos).

Através da Tabela 5.10 podemos avaliar a energia clássica (ff99χOL) dos termos separadamente. Como os valores de alguns termos do campo de força se repetem, foi atribuída uma variável (S) que irá assumir a soma desses valores. Nela podemos também perceber a diminuição do valor total da energia de cada rotação. Isso pode ser indicativo de que a molécula reparametrizada está numa conformação mais estável do que a anterior, semelhante como aconteceu na primeira parametrização (seção 5.1.2). Isso se deve graças à boa parametrização tanto do termo de torção quanto do eletrostático. As energias dos outros nucleosídeos podem ser vistas no Apêndice C.

Tabela 5.10. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGW)

	Rotação*	S	E_{tor}	E_{ele}	Total
Ref.[17]	0	24,2132	45,0072	-49,1651	20,0553
	50	28,342	46,8137	-49,8259	25,3298
	100	28,7408	47,5273	-49,5510	26,7171
	150	27,7493	51,6653	-37,4597	41,9549
	200	26,4749	49,7344	-34,9874	41,2219
	250	26,0564	51,6124	-28,8912	48,7776
	300	26,5462	54,8574	-59,7636	21,6400
	350	26,3776	56,5018	-58,3877	24,4917
AGW	0	24,2132	41,3852	-51,4451	14,1533
	50	28,342	40,7867	-52,6215	16,5072
	100	28,7408	40,7802	-52,7469	16,7741
	150	27,7493	44,3855	-57,8397	14,2951
	200	26,4749	41,8455	-50,2544	18,066
	250	26,0564	41,0021	-50,5569	16,5016
	300	26,5462	42,0268	-56,5746	11,9984
	350	26,3776	43,1481	-57,2237	12,302

* Ângulos em graus.
Energias em kcal/mol.

5.2.3 Simulação do RNA usando a Segunda Parametrização

Toda a metodologia para validação dos parâmetros usada na seção 5.1.3 foi repetida nesta seção, utilizando os dados da segunda fase junto com os dados da primeira fase da parametrização obtivemos resultados muito semelhantes. Porém, se comparados com os dados da literatura obtivemos, melhores resultados que aqueles já reportados como podemos ver na Tabela 5.11. A redução do RMSD foi de 81%.

As duas estruturas otimizadas das seções 5.1.3 e 5.2.3 foram alinhadas usando o software PyMOL [87]. A fim de facilitar a visualização, as moléculas de água que

Tabela 5.11. Comparação usando o RMSD entre a metodologia aplicada (AGW) e o dado experimental

	RMSD
Referência [17]	1,50
AGW	0,28

[a] Valores em Angstrom.

fazem a solvatação foram removidas. Os resultados mostram a melhoria da estrutura usando a nova parametrização, quando comparados com os dados experimentais.

Podemos ver na Figura 5.17 a parametrização completa usando o refinamento tanto da ligação glicosídica quanto da ligação fósforo diéster. Nela temos a estrutura otimizada usando os parâmetros refinados (vermelho) e a estrutura obtida com dados experimentais (verde).

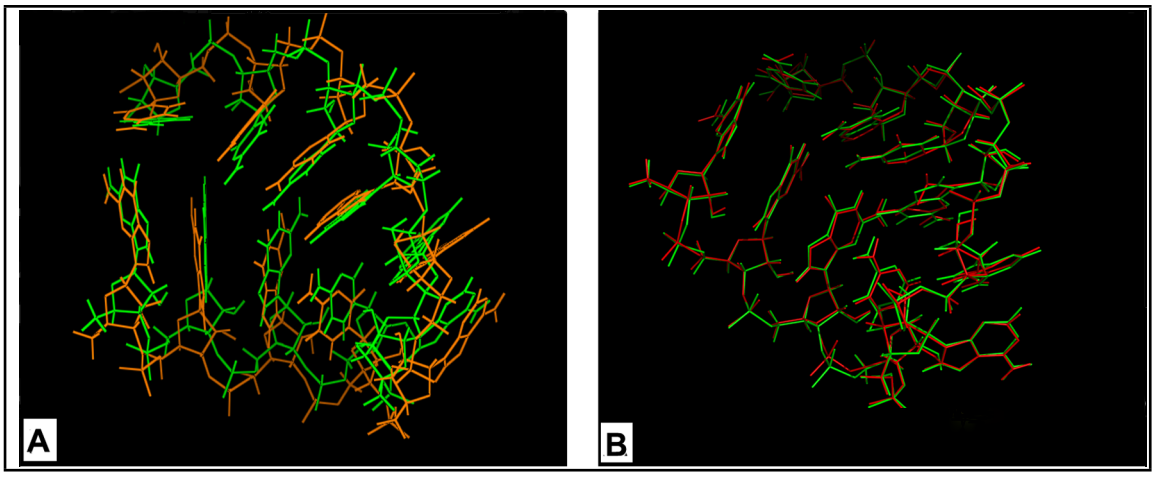


Figura 5.17. Comparação das estruturas encontradas pelos dois refinamentos do campo de força e o dado experimental. (A) Otimização gerada com os parâmetros encontrados na referência [17] (laranja) e o dado experimental (verde). (B) Otimização gerada com os parâmetros encontrados pelo AG (vermelho) e o dado experimental (verde).

Capítulo 6

Conclusões e considerações finais

O presente estudo propôs o uso de cálculos quânticos como referência para o refinamento dos parâmetros do termo torcional da torção glicosídica e a torção α da ligação fósforo diéster usando um modelo matemático chamado algoritmo genético, a fim de melhorar a precisão na representação de RNA. De acordo com Lankas *et. al.* [71] e Yildirim *et.al.* [6] estes são os ângulos mais relevante em ácidos nucleicos para ser refinado do campo de força.

A otimização usando a abordagem AGW (refinamento dos parâmetros torcionais e eletrostático) produziu melhores resultados demonstrando a importância das interações entre os pares de base. Esta interação é fundamental para a estabilização das hélices do RNA [18]. Portanto uma boa parametrização nos termos eletrostático e torcionais é muito importante para se poder conseguir uma representação quantitativa da molécula em questão (próxima do dado experimental).

O método do AGW pode ser utilizado para obter e otimizar os parâmetros para diversos termos do campo de força para ser usado, por exemplo, nas simulações de DM ajustando os parâmetros de acordo com estruturas e energias de um dado de referência (cálculo quântico). O AGW resultou em um refinamento dos parametros do potencial de torção do campo de força Amber capaz de: encontrar localizações próximas do mínimo global em uma função que tem múltiplos mínimos e com a capacidade de reproduzir energias de acordo com os seus correspondentes (valores de DFT) a fim de representar a estrutura de moléculas que não foram incluídos na simulação de DFT, como um caso de teste.

O objetivo da metodologia foi de reduzir ainda mais o erro dos cálculos computacionais envolvidos na parametrização do campo de força, com particular atenção a termo de torção do campo de força para ácidos nucleicos, especificamente o RNA. A nova parametrização reduziu, em média, 50% o valor do RMSE. Tal como demonstrado

no presente trabalho, os parâmetros encontrados levaram a uma melhor representação estrutural da molécula 1r4h quando comparado com parametrizações previamente relatadas na literatura [17]. É importante ressaltar que a abordagem atual foi capaz de ir mais longe e prever com um nível razoável de precisão uma estrutura experimental (RMN), que não foi tida em conta no processo de refinamento.

Esta nova forma de otimizar os parâmetros produzirá melhorias no estudos das moléculas (por exemplo, nos comportamentos e nas propriedades), dado que se mostrou mais preciso do que a metodologia anteriormente proposta. Além disso, o método pode ser aplicado para refinar os parâmetros de outros termos do campo de força, permitindo no futuro ter um campo de força totalmente parametrizado e mais preciso.

Usando esta metodologia apresentadas neste trabalho como um teste, a técnica AGW mostrou para ser capaz de lidar com o potencial das torções glicosídicas bem como a topologia multimodal da energia do sistema. Como o refinamento dos parâmetros está limitado pelos seus dados de referência, então esse refinamento pode ser melhorado se tivermos dados experimentais mais refinados ou cálculos quânticos mais refinados.

Além do mais, o sucesso do método AGW se deu graças ao seu custo computacional relativamente baixo, capaz de refletir a qualidade da parametrização. Do ponto de vista do AGW, o conjunto de dados de referência é nada mais do que uma lista de configurações de estruturas e suas respectivas energias.

Existem muitas deficiências que são conhecidas já citadas na introdução deste trabalho. Além do mais, os parâmetros em diferentes campos de força podem ser muito diferentes, pois são parametrizados de formas diferentes. Apesar de suas deficiências, ele continuará a ser amplamente utilizado, devido à sua eficiência computacional, enquanto que a sua confiabilidade irá continuar a ser melhorada.

6.1 Trabalhos Futuros

Como sugestão para possíveis trabalhos futuros, a fim de melhorar a metodologia proposta aqui, pode-se apontar:

- Fazer rotações menores para tornar a parametrização mais refinada (ao invés de 50 em 50 fazer de 10 em 10 graus).

- Fazer as parametrizações tanto da ligação glicosídica quanto a ligação fósforo diéster juntas. Tendo assim que refazer os cálculos quânticos com a molécula contendo o grupo fosfato e a base do nucleotídeo.

- Fazer parametrizações para outras moléculas;
- Desenvolver novos operadores ou usar algoritmos híbridos para melhorar a otimização do algoritmo genético.
- Reparametrizar todas as ligações do RNA e todos os termos do campo de força. Fazer isso também para o DNA para generalizar o campo de força para todos os ácidos nucleicos.

Referências

[1] T. R. C. Guizado, *PhD thesis*, Pontifícia Universidade Católica do Rio de Janeiro, 2008.

[2] J. P. Rino and N. Studart, *Química Nova*, 2011, **24**, 838–845.

[3] G. H. J. Gates, *PhD thesis*, Faculty of the School of Engineering and Management of the Air Force Institute of Technology Air University, 1994.

[4] B. Ramos, *PhD thesis*, Escola Politécnica da Universidade de São Paulo, 2009.

[5] M. A. Ditzler, M. Otyepka, J. Sponer and N. G. Walter, *Accounts of Chemical Research*, 2010, **43**, 40–47.

[6] I. Yildirim, S. D. Kennedy, H. A. Stern, J. M. Hart, R. Kierzek and D. H. Turner, *Journal of Chemical Theory and Computation*, 2012, **8**, 172–181.

[7] J. Sponer, A. Mládek, J. E. Sponer, D. Svozil, M. Zgarbová, P. Banás, P. Jurecka and M. Otyepka, *Physical Chemistry Chemical Physics*, 2012, **14**, 15257–15277.

[8] A. Spasic, J. Serafini and D. H. Mathews, *Journal Chemistry Theory Comput*, 2012, **8**, 2497–2505.

[9] J. A. Doudna and T. R. Cech, *Nature*, 2002, **418**, 222–228.

[10] P. Nissen, J. Hansen, N. Ban, P. B. Moore and T. A. Steitz, *Science*, 2000, **289**, 920–930.

[11] W. G. Scott, *Curr. Opin. Struct. Biol*, 2007, **17**, 280–286.

[12] B. J. Tucker and R. R. Breaker, *Curr. Opin. Struct. Biol*, 2005, **15**, 342–348.

[13] L. David, W. Huber, M. Granovskaia, J. Toedling, C. J. Palm, L. Bofkin, T. Jones, R. W. Davis and L. M. Steinmetz, *Proc. Natl. Acad. Science*, 2006, **103**, 5320–5325.

[14] N. R. França, D. J. Mesquita, A. B. Lima, F. V. Pucci, L. E. Andrade and N. P. Silva, *Rev Bras Reumatologia*, 2010, **50**, 695–709.

[15] B. Albert, A. Johnson, J. Lewis, M. Raff, K. Roberts and P. Walter, *Molecular Biology of the Cell*, Fifth Edition, New York, 2008.

[16] J. W. Ponder and D. A. Case, *Advances in Protein Chemistry*, 2003, **66**, 27–85.

[17] M. Zgarbová, M. Otyepka, J. Sponer, A. Mládek, P. Banás, T. E. Cheatham, III and P. Jurecka, *J. Chem. Theory Comput.*, 2011, **7**, 2886–2902.

[18] P. Banás, A. Mladék, M. Otyepka, M. Zgarbová, P. Jurecka, D. Svozil, F. Lankas and J. Sponer, *J. Chem. Theory Comput.*, 2012, **8**, 2448–2460.

[19] I. Yildirim, H. A. Stern, S. D. Kennedy, J. D. Tubbs and D. H. Turner, *Journal of Chemical Theory and Computation*, 2010, **6**, 1520–1531.

[20] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell and P. A. Kollman, *J. Am. Chem. Soc.*, 1995, **117**, 5179–5197.

[21] T. E. Cheatham, P. Cieplak and P. A. Kollman, *Journal Biomol. Struct. Dyn.*, 1999, **16**, 845–862.

[22] J. M. Wang, P. Cieplak and P. A. Kollman, *Journal of Computational Chemistry*, 2000, **21**, 1049–1074.

[23] P.-P. Zhou and W.-Y. Qiu, *Journal Physical Chemistry*, 2009, **113**, 10306–10320.

[24] A. Perez, I. Marchan, D. Svozil, J. Sponer, T. E. Cheatham, C. A. Laughton and M. Orozco, *Biophysical Journal*, 2007, **92**, 3817–3829.

[25] P. Auffinger and E. Westhof, *Current Opinion in Structural Biology*, 1998, **8**, 227–236.

[26] I. Yildirim and D. H. Turner, *Biochemistry*, 2005, **44**, 13225–13234.

[27] H. Ode, Y. Matsuo, S. Neya and T. Hoshino, *Journal of Computational Chemistry*, 2008, **29**, 15–22.

[28] K. Murzyn, M. Bratek and M. Pasenkiewicz-Gierula, *Journal of Physical Chemistry B*, 2013, **117**, 16388–16396.

[29] S. A. Adcock and J. A. McCammon, *Chemical Reviews*, 2006, **106**, 1589–1605.

[30] W. F. van Gunsteren, D. Bakowies, R. Baron, I. Chandrasekhar, M. Christen, X. Daura, P. Gee, D. P. Geerke, A. Glattli, P. H. Hunenberger, M. A. Kastenholz, C. Ostenbrink, M. Schenk, D. Trzesniak, N. F. A. Vegt and H. B. Yu, *Angew. Chem. Int. Edition*, 2006, **45**, 4064–4092.

[31] Y. B. Cheng and J. K. Sykulski, *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 1996, **9**, 59–69.

[32] M. Buchvarova and P. I. Y. Velinov, *Advances in Space Research*, 2010, **45**, 1026–1034.

[33] Y. Sakae and Y. Okamoto, *Journal of Theoretical and Computational Chemistry*, 2004, **3**, 339–358.

[34] J. Kästner, J. M. Carr, T. W. Keal, W. Thiel, A. Wander and P. Sherwood, *Journal Physical Chemistry*, 2009, **113**, 11856–11865.

[35] M. X. Silva, B. R. L. Galvão and J. C. Belchior, *Physical Chemistry Chemical Physics*, 2014, **16**, 8895–8904.

[36] D. D. C. Rodrigues, A. M. Nascimento, H. A. Duarte and J. C. Belchior, *Chemical Physics*, 2008, **349**, 91–97.

[37] F. F. Guimarães, J. C. Belchior, R. L. Johnston and C. Roberts, *Physical Chemistry Chemical Physics*, 2002, **19**, 8327–8333.

[38] M. Böyükata, E. Borges, J. C. Belchior and J. P. Braga, *Physical Chemistry Chemical Physics*, 2002, **19**, 8327–8333.

[39] A. H. Wright, Foundations of Genetic Algorithms, 1991, pp. 205–218.

[40] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, Oxford, UK, 1996.

[41] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer, Gernany, 2002.

[42] I. N. Levine, *Quantum Chemistry*, Prentice Hall, New Jersey, 1991.

[43] C. Caetano, *PhD thesis*, Instituto Tecnológico de Aeronáutica, 2003.

[44] B. Alder, S. Fernbach and M. Rotenberg, *The Gaussian Function in Calculations of Statistical Mechanics and Quantum Mechanics*, Academic Press, New York, 1963.

[45] M. G. Santos, *PhD thesis*, UNICAMP, 1992.

[46] S. Borman, *Chemical and Engineering New*, 1990, **67**, 28–32.

[47] T. Ziegler, *Chem. Rev.*, 1991, **91**, 651–667.

[48] P. Hohenberg and W. Konh, *Physycal Review*, 1964, **136**, 864–871.

[49] W. Kohn and L. J. Sham, *Physycal Review*, 1965, **140**, 1133–1138.

[50] P. J. P. d. Oliveira, *PhD thesis*, Universidade Federal do Espírito Santo, 2010.

[51] J. C. Slater, *Physycal Review*, 1930, **36**, 51–62.

[52] P. O. Lowdin, *Physycal Review*, 1953, **90**, 120–127.

[53] I. S. O. Bezerra, *PhD thesis*, Pontifícia Universidade Católica do Rio de Janeiro, 2009.

[54] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, Jr and P. W. Weiner, *Journal America Chemistry Society*, 1984, **106**, 765–784.

[55] S. Patel and C. L. Brooks, *Journal Computational Chemical*, 2004, **25**, 1504–1514.

[56] A. D. Mackerell, Jr, D. Bashford, M. Bellott, R. L. Dunbrack, Jr, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin and M. Karplus, *Journal Physical Chemistry*, 1998, **102**, 3586–3616.

[57] W. F. v. Gunsteren, S. R. Billeter, A. A. Eising, P. H. Hünenberger, P. Krüger, A. E. Mark, W. R. P. Scott and I. G. Tironi, *Vdf Hochschulverlag an der ETH Zürich*, 1996.

[58] W. L. Jorgensen and J. Tirado-Rives, *Journal American Chemical Society*, 1988, **110**, 6–18.

[59] N. L. Allinger, K.-H. Chen, J.-H. Lii and K. A. Durkin, *J Comput Chem*, 2003, **24**, 1447–1472.

[60] D. van der Spoel, E. Lindahl, B. Hess, A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, K. A. Feenstra, R. van Drunen and H. J. C. Berendsen, *Gromacs User Manual version 3.2*, Groningen, Netherlands, 2004.

[61] T. Lima, *PhD thesis*, Universidade de São Paulo, 2006.

[62] S. Rubenstein, C. Kundrot, S. Huston, M. Dudek, Y. Kong, R. Hart, M. Hodsdon, R. Pappu, W. Mooij, G. Loeffler, M. Vorobieva, N. Sokolova, P. Bagossi, P. Ren, A. Carlsson, A. Kutepov, A. Grossfield, M. Schnieders, D. Gohara and T. Darden, 2014.

[63] D. Goldberg and J. Holland, *Mach. Learn*, 1988, **3**, 95–99.

[64] D. Campbell, *Behavior Science*, 1956.

[65] G. Friedman, *PhD thesis*, UCLA, 1956.

[66] C. Silva Junior and S. Sasson, *Biologia 1*, Saraiva, São Paulo, 2011.

[67] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*, Springer-Verlag, New York, 2004.

[68] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, Spring, California, 2008.

[69] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, New York, 2000.

[70] D. C. Lucas, *PhD thesis*, UFPel-RS, 2000.

[71] F. Lankas, N. Spackova, M. Moakher, P. Enkhbayar and J. Sponer, *Nucleic Acids Research*, 2010, **38**, 3414–3422.

[72] H. Margenau and N. R. Kestner, *Int. Series of Mono. In Natural Phy. : Theory of Intermolecular Forces*, Pergamon Press, New York, 1971.

[73] X. X. Yao, C. G. Ji, D. Q. Xie and J. Z. H. Zhang, *Journal of Computational Chemistry*, 2013, **34**, 1136–1142.

[74] *A Portal for Three-dimensional Structural Information about Nucleic Acids*, 2014, `http://ndbserver.rutgers.edu/`.

[75] N. M. OBoyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch and G. R. Hutchison, 2006.

[76] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. J. Su, T. L. Windus, M. Dupuis and J. A. Montgomery, *J. Comput. Chem.*, 1993, **14**, 1347–1363.

[77] C. Froese-Fischer, *The Hartree-Fock Method for Atoms*, Wiley, New York, 1997.

[78] P. H. Shah and R. C. Batra, *Computational Materials Science*, 2014, **83**, 349–361.

[79] A. Zhong, X. Jiang, Y. Hu and C. Du, *Journal of the Society of Leather Technologists and Chemists*, 2013, **97**, 121–124.

[80] S. Saito, K. Ohno, T. Suzuki and H. Sakuraba, *Molecular Genetics and Metabolism*, 2012, **105**, 244–248.

[81] O. Guvench and A. D. MacKerell, *Journal Molecular Modeling*, 2008, **14**, 667–679.

[82] *An Information Portal to Biological Macromolecular Structures*, 2014, `http://www.rcsb.org/pdb/home/home.do`.

[83] K. Coutinho, Symposium in Memory of Michael C. Zerner, 2000.

[84] J. Aqvist, *Journal Physical Chemistry*, 1990, **94**, 8021–8024.

[85] A. M. Lesk, *Introdução à Bioinformática*, Artmed, São Paulo, 2005.

[86] P. E. Bourne and H. Weissig, *Structural bioinformatics*, John Wiley & Sons, New Jersey, 2003.

[87] L. Schrödinger, PyMOL The PyMOL Molecular Graphics System, Version 1.3, Schrödinger, LLC.

Apêndice A

Parâmetros do Campo de Força Amber

Tabela A.1. TINKER - Número da classe para os tipos de átomos do Amber

Clásse	Átomo
1	CT
2	C6
3	CA
4	CM
5	CC
6	CV
7	CW
8	CR
9	CB
10	C*
11	CN
12	CK
13	CQ
14	N
15	NA
16	NB
17	NC

continua na próxima página

continuação da página anterior

Clásse	Átomo
18	N*
19	N2
20	N3
21	OW
22	OH
23	OS
24	O
25	O2
26	S
27	SH
28	P
29	H
30	HW
31	HO
32	HS
33	HA
34	HC
35	H1
36	H2
37	H3
38	HP
39	H4
40	H5
41	C2
42	C1

Tabela A.2. TINKER - Parâmetros de ligação do Amber (f99)

Clásse 1	Clásse 2	K_r	r_{eq}
1	1	310	1.526
1	2	317	1.522
1	3	317	1.51
1	4	317	1.51
1	5	317	1.504
1	10	317	1.495
1	14	337	1.449
1	18	337	1.475
1	19	337	1.463
1	20	367	1.471
1	22	320	1.41
1	23	320	1.41
1	26	227	1.81
1	27	237	1.81
1	34	340	1.09
1	35	340	1.09
1	36	340	1.09
1	37	340	1.09
1	38	340	1.09
2	3	469	1.409
2	4	410	1.444
2	9	447	1.419
2	14	490	1.335
2	15	418	1.388
2	17	457	1.358
2	18	424	1.383
2	24	570	1.229
2	25	656	1.25
3	3	469	1.4
3	4	427	1.433

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	K_r	r_{eq}
3	9	469	1.404
3	11	469	1.4
3	15	427	1.381
3	17	483	1.339
3	19	481	1.34
3	22	450	1.364
3	33	367	1.08
3	39	367	1.08
4	4	549	1.35
4	18	448	1.365
4	33	367	1.08
4	39	367	1.08
4	40	367	1.08
5	6	512	1.375
5	7	518	1.371
5	15	422	1.385
5	16	410	1.394
6	16	410	1.394
6	39	367	1.08
7	10	546	1.352
7	15	427	1.381
7	39	367	1.08
8	15	477	1.343
8	16	488	1.335
8	40	367	1.08
9	9	520	1.37
9	10	388	1.459
9	11	447	1.419
9	16	414	1.391
9	17	461	1.354
9	18	436	1.374
10	34	367	1.08

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	K_r	r_{eq}
11	15	428	1.38
12	16	529	1.304
12	18	440	1.371
12	40	367	1.08
13	17	502	1.324
13	40	367	1.08
14	29	434	1.01
15	29	434	1.01
18	29	434	1.01
19	29	434	1.01
20	29	434	1.01
21	30	553	0.9572
22	28	230	1.61
22	31	553	0.96
23	28	230	1.61
23	31	553	0.96
25	28	525	1.48
26	26	166	2.038
27	32	274	1.336
30	30	553	1.5136
41	42	367	1.08
41	18	440	1.371
41	16	529	1.304
41	40	367	1.08
2	42	410	1.444
3	42	427	1.433
42	42	549	1.35
42	1	317	1.51
42	33	367	1.08
42	39	367	1.08
42	18	448	1.365
4	42	549	1.35

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	K_r	r_{eq}

Tabela A.3. TINKER - Parâmetros angular do Amber (f99)

Clásse 1	Clásse 2	Clásse 3	K_r	r_{eq}
1	1	1	40	109.5
1	1	2	63	111.1
1	1	3	63	114
1	1	5	63	113.1
1	1	10	63	115.6
1	1	14	80	109.7
1	1	18	50	109.5
1	1	19	80	111.2
1	1	20	80	111.2
1	1	22	50	109.5
1	1	23	50	109.5
1	1	26	50	114.7
1	1	27	50	108.6
1	1	34	50	109.5
1	1	35	50	109.5
1	1	36	50	109.5
1	1	38	50	109.5
2	1	14	63	110.1
2	1	20	80	111.2
2	1	34	50	109.5
2	1	35	50	109.5
2	1	38	50	109.5
3	1	34	50	109.5
4	1	34	50	109.5
5	1	34	50	109.5
10	1	34	50	109.5
14	1	35	50	109.5
18	1	23	50	109.5
18	1	35	50	109.5
18	1	36	50	109.5

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	K_θ	θ_{eq}
19	1	35	50	109.5
20	1	35	50	109.5
20	1	38	50	109.5
22	1	35	50	109.5
23	1	35	50	109.5
23	1	36	50	109.5
26	1	35	50	109.5
27	1	35	50	109.5
34	1	34	35	109.5
35	1	35	35	109.5
36	1	36	35	109.5
38	1	38	35	109.5
1	2	14	70	116.6
1	2	24	80	120.4
1	2	25	70	117
3	2	3	63	120
4	2	15	70	114.1
4	2	24	80	125.3
9	2	15	70	111.3
9	2	24	80	128.8
14	2	24	80	122.9
15	2	18	70	115.4
15	2	24	80	120.6
17	2	18	70	118.6
17	2	24	80	122.5
18	2	24	80	120.9
24	2	24	80	126
25	2	25	80	126
1	3	3	70	120
2	3	3	63	120
2	3	33	50	120
3	3	3	63	120

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	K_θ	θ_{eq}
3	3	9	63	120
3	3	11	63	120
3	3	22	70	120
3	3	33	50	120
3	3	39	50	120
4	3	17	70	121.5
4	3	19	70	120.1
9	3	17	70	117.3
9	3	19	70	123.5
9	3	33	50	120
9	3	39	50	120
11	3	33	50	120
15	3	17	70	123.3
15	3	19	70	116
17	3	19	70	119.3
19	3	19	70	120
1	4	2	70	119.7
1	4	4	70	119.7
2	4	4	63	120.7
2	4	33	50	119.7
2	4	39	50	119.7
3	4	4	63	117
3	4	33	50	123.3
3	4	39	50	123.3
4	4	18	70	121.2
4	4	33	50	119.7
4	4	39	50	119.7
18	4	39	50	119.1
1	5	6	70	120
1	5	7	70	120
1	5	15	70	120
1	5	16	70	120

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	K_θ	θ_{eq}
6	5	15	70	120
7	5	15	70	120
7	5	16	70	120
5	6	16	70	120
5	6	39	50	120
16	6	39	50	120
5	7	15	70	120
5	7	39	50	120
10	7	15	70	108.7
10	7	39	50	120
15	7	39	50	120
15	8	15	70	120
15	8	16	70	120
15	8	40	50	120
16	8	40	50	120
2	9	9	63	119.2
2	9	16	70	130
3	9	9	63	117.3
3	9	10	63	134.9
3	9	11	63	116.2
3	9	16	70	132.4
9	9	16	70	110.4
9	9	17	70	127.7
9	9	18	70	106.2
10	9	11	63	108.8
17	9	18	70	126.2
1	10	7	70	125
1	10	9	70	128.6
7	10	9	63	106.4
3	11	9	63	122.7
3	11	15	70	132.8
9	11	15	70	104.4

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	K_θ	θ_{eq}
16	12	18	70	113.9
16	12	40	50	123.05
18	12	40	50	123.05
17	13	17	70	129.1
17	13	40	50	115.45
1	14	1	50	118
1	14	2	50	121.9
1	14	29	50	118.04
2	14	29	50	120
29	14	29	35	120
2	15	2	70	126.4
2	15	3	70	125.2
2	15	29	50	116.8
3	15	29	50	118
5	15	8	70	120
5	15	29	50	120
7	15	8	70	120
7	15	11	70	111.6
7	15	29	50	120
8	15	29	50	120
11	15	29	50	123.1
5	16	8	70	117
6	16	8	70	117
9	16	12	70	103.8
2	17	3	70	120.5
3	17	9	70	112.2
3	17	13	70	118.6
9	17	13	70	111
1	18	2	70	117.6
1	18	4	70	121.2
1	18	9	70	125.8
1	18	12	70	128.8

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	K_θ	θ_{eq}
2	18	4	70	121.6
2	18	29	50	119.2
4	18	29	50	121.2
9	18	12	70	105.4
9	18	29	50	125.8
12	18	29	50	128.8
1	19	3	50	123.2
1	19	29	50	118.4
3	19	29	50	120
29	19	29	35	120
1	20	1	50	109.5
1	20	29	50	109.5
29	20	29	35	109.5
30	21	30	100	104.52
1	22	31	55	108.5
3	22	31	50	113
28	22	31	45	108.5
1	23	1	60	109.5
1	23	28	100	120.5
28	23	28	100	120.5
1	26	1	62	98.9
1	26	26	68	103.7
1	27	32	43	96
32	27	32	35	92.07
22	28	23	45	102.6
22	28	25	45	108.23
23	28	23	45	102.6
23	28	25	100	108.23
25	28	25	140	119.9
21	30	30	0	127.74
40	41	18	50	123.05
40	41	16	50	123.05

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	K_θ	θ_{eq}
18	41	16	70	113.9
9	18	41	70	105.4
41	18	1	70	128.8
9	16	41	70	103.8
42	2	15	70	114.1
42	2	24	80	125.3
42	3	19	70	120.1
42	3	17	70	121.5
2	42	42	63	120.7
2	42	1	70	119.7
2	42	33	50	119.7
2	42	39	50	119.7
3	42	42	63	117
3	42	33	50	123.3
3	42	39	50	123.3
42	42	1	70	129.7
42	42	33	50	129.7
42	42	39	50	129.7
42	42	18	70	121.2
39	42	18	50	119.1
35	1	42	50	109.5
34	1	42	50	109.5
2	18	42	70	121.6
42	18	1	70	121.2
3	4	42	63	117
42	4	33	50	123.3
18	42	4	70	121.2
4	42	39	50	119.7
23	1	23	80	126

Tabela A.4. TINKER - Parâmetros torcional de Zgarbová *et. al.*[17]

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
1	1	1	1	0.2	180	1
1	1	1	1	0.25	180	2
1	1	1	1	0.18	0	3
1	1	1	2	0.156	0	3
1	1	1	14	0.156	0	3
1	1	1	18	0.156	0	3
1	1	1	19	0.156	0	3
1	1	1	20	0.156	0	3
1	1	1	22	0.156	0	3
1	1	1	23	0.156	0	3
1	1	1	26	0.156	0	3
1	1	1	34	0.16	0	3
1	1	1	35	0.156	0	3
1	1	1	36	0.156	0	3
1	1	1	38	0.156	0	3
2	1	1	2	0.156	0	3
2	1	1	3	0.156	0	3
2	1	1	5	0.156	0	3
2	1	1	10	0.156	0	3
2	1	1	14	0.156	0	3
2	1	1	20	0.156	0	3
2	1	1	22	0.156	0	3
2	1	1	26	0.156	0	3
2	1	1	27	0.156	0	3
2	1	1	34	0.156	0	3
2	1	1	35	0.156	0	3
2	1	1	38	0.156	0	3
3	1	1	14	0.156	0	3
3	1	1	20	0.156	0	3
3	1	1	35	0.156	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
3	1	1	38	0.156	0	3
5	1	1	14	0.156	0	3
5	1	1	20	0.156	0	3
5	1	1	35	0.156	0	3
3	1	1	38	0.156	0	3
5	1	1	14	0.156	0	3
5	1	1	20	0.156	0	3
5	1	1	35	0.156	0	3
5	1	1	38	0.156	0	3
10	1	1	14	0.156	0	3
10	1	1	20	0.156	0	3
10	1	1	35	0.156	0	3
10	1	1	38	0.156	0	3
14	1	1	22	0.156	0	3
14	1	1	26	0.156	0	3
14	1	1	27	0.156	0	3
14	1	1	34	0.156	0	3
14	1	1	35	0.156	0	3
18	1	1	22	0.156	0	3
18	1	1	34	0.156	0	3
18	1	1	35	0.156	0	3
19	1	1	34	0.156	0	3
20	1	1	22	0.156	0	3
20	1	1	26	0.156	0	3
20	1	1	27	0.156	0	3
20	1	1	34	0.156	0	3
20	1	1	35	0.156	0	3
22	1	1	18	0.156	0	3
22	1	1	22	0.144	0	3
22	1	1	22	1.175	0	2
22	1	1	22	0.144	0	3
22	1	1	22	1.175	0	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
22	1	1	34	0.156	0	3
22	1	1	34	0.25	0	1
22	1	1	35	0.156	0	3
22	1	1	36	0.156	0	3
22	1	1	38	0.156	0	3
23	1	1	23	0.144	0	3
23	1	1	23	1.175	0	2
23	1	1	34	0.156	0	3
23	1	1	34	0.25	0	1
22	1	1	35	0.156	0	3
22	1	1	36	0.156	0	3
22	1	1	38	0.156	0	3
23	1	1	23	0.144	0	3
23	1	1	23	1.175	0	2
23	1	1	23	0.156	0	3
23	1	1	23	0.25	0	1
23	1	1	35	0.156	0	3
26	1	1	34	0.156	0	3
26	1	1	35	0.156	0	3
26	1	1	38	0.156	0	3
27	1	1	35	0.156	0	3
27	1	1	38	0.156	0	3
34	1	1	34	0.15	0	3
34	1	1	35	0.156	0	3
34	1	1	36	0.156	0	3
34	1	1	38	0.156	0	3
35	1	1	35	0.156	0	3
35	1	1	36	0.156	0	3
35	1	1	38	0.156	0	3
1	1	2	14	0.07	0	2
1	1	2	14	0.1	0	4
1	1	2	24	0	0	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
1	1	2	25	0	0	2
14	1	2	14	1.7	180	1
14	1	2	14	2	180	2
14	1	2	24	0	0	2
14	1	2	25	0	0	2
20	1	2	14	0	0	2
20	1	2	24	0	0	2
20	1	2	25	0	0	2
34	1	2	14	0	0	2
34	1	2	24	0.8	0	1
34	1	2	24	0.08	180	3
34	1	2	25	0	0	2
35	1	2	14	0	0	2
35	1	2	24	0.8	0	1
35	1	2	24	0.08	180	3
35	1	2	25	0	0	2
1	1	3	3	0	0	2
34	1	3	3	0	0	2
34	1	4	2	0	0	3
34	1	4	4	1.15	0	1
34	1	4	4	0.38	180	3
1	1	5	6	0	0	2
1	1	5	7	0	0	2
1	1	5	15	0	0	2
1	1	5	16	0	0	2
34	1	5	6	0	0	2
34	1	5	7	0	0	2
34	1	5	15	0	0	2
34	1	5	16	0	0	2
1	1	10	7	0	0	2
1	1	10	9	0	0	2
34	1	10	7	0	0	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
34	1	10	9	0	0	2
1	1	14	1	0	0	2
1	1	14	2	0.53	0	1
1	1	14	2	0.15	180	3
1	1	14	2	0.5	180	4
1	1	14	29	0	0	2
2	1	14	1	0	0	2
2	1	14	2	0.8	0	1
2	1	14	2	0.85	180	2
2	1	14	29	0	0	2
35	1	14	1	0	0	2
35	1	14	2	0	0	2
35	1	14	29	0	0	2
1	1	18	2	0	0	2
1	1	18	4	0	0	2
1	1	18	42	0	0	2
1	1	18	9	0	0	2
1	1	18	12	0	0	2
1	1	18	41	0	0	2
23	1	18	2	0	0	2
23	1	18	4	1.0251	149.88	1
23	1	18	4	1.7488	16.76	2
23	1	18	4	0.5815	179.35	3
23	1	18	4	0.3515	16	4
23	1	18	9	0	0	2
23	1	18	12	0.7051	74.76	1
23	1	18	12	1.0655	6.23	2
23	1	18	12	0.4427	168.65	3
23	1	18	12	0.256	3.97	4
36	1	18	2	0	0	2
36	1	18	4	0	0	2
36	1	18	42	0	0	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
36	1	18	9	0	0	2
36	1	18	12	0	0	2
36	1	18	41	0	0	2
1	1	19	3	0	0	3
1	1	19	29	0	0	3
35	1	19	3	0	0	3
35	1	19	29	0	0	3
1	1	20	1	0.156	0	3
1	1	20	29	0.156	0	3
2	1	20	1	0.156	0	3
2	1	20	29	0.156	0	3
35	1	20	1	0.156	0	3
35	1	20	29	0.156	0	3
38	1	20	1	0.156	0	3
38	1	20	29	0.156	0	3
1	1	22	31	0.025	0	1
1	1	22	31	0.16	0	3
35	1	22	31	0.167	0	3
1	1	23	1	0.383	0	3
1	1	23	1	0.1	180	2
1	1	23	2	0.383	0	3
1	1	23	2	0.8	180	1
1	1	23	28	0.383	0	3
18	1	23	1	0.383	0	3
18	1	23	1	0.65	0	2
23	1	23	1	1.35	180	1
23	1	23	1	0.85	180	2
23	1	23	1	0.1	0	3
35	1	23	1	0.383	0	3
35	1	23	28	0.383	0	3
36	1	23	1	0.383	0	3
1	1	26	1	0.333	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
1	1	26	26	0.333	0	3
35	1	26	1	0.333	0	3
35	1	26	26	0.333	0	3
1	1	27	32	0.25	0	3
35	1	27	32	0.25	0	3
15	2	4	1	2.175	180	2
15	2	4	4	2.175	180	2
15	2	4	42	2.175	180	2
15	2	4	33	2.175	180	2
24	2	4	1	2.175	180	2
24	2	4	4	2.175	180	2
24	2	4	4	0.3	0	3
24	2	4	42	2.175	180	2
24	2	4	42	0.3	0	3
24	2	4	33	2.175	180	2
15	2	9	9	3	180	2
15	2	9	16	3	180	2
24	2	9	9	3	180	2
24	2	9	16	3	180	2
1	2	14	1	2.5	180	2
1	2	14	29	2.5	180	2
24	2	14	1	2.5	180	2
24	2	14	29	2	0	1
24	2	14	29	2.5	180	2
4	2	15	2	1.35	180	2
4	2	15	29	1.35	180	2
9	2	15	3	1.35	180	2
9	2	15	29	1.35	180	2
18	2	15	2	1.35	180	2
18	2	15	29	1.35	180	2
24	2	15	2	1.35	180	2
24	2	15	3	1.35	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
24	2	15	29	1.35	180	2
18	2	17	3	4	180	2
24	2	17	3	4	180	2
15	2	18	1	1.45	180	2
15	2	18	4	1.45	180	2
15	2	18	42	1.45	180	2
17	2	18	1	1.45	180	2
17	2	18	4	1.45	180	2
17	2	18	42	1.45	180	2
24	2	18	1	1.45	180	2
24	2	18	4	1.45	180	2
24	2	18	42	1.45	180	2
24	2	22	31	1.9	0	1
24	2	22	31	2.3	180	2
24	2	23	1	1.4	180	1
24	2	23	1	2.7	180	2
1	3	3	3	3.625	180	2
1	3	3	33	3.625	180	2
3	3	3	3	3.625	180	2
3	3	3	9	3.625	180	2
3	3	3	11	3.625	180	2
3	3	3	22	3.625	180	2
3	3	3	33	3.625	180	2
9	3	3	33	3.625	180	2
11	3	3	33	3.625	180	2
22	3	3	33	3.625	180	2
33	3	3	33	3.625	180	2
17	3	4	4	2.55	180	2
17	3	4	42	2.55	180	2
17	3	4	33	2.55	180	2
19	3	4	4	2.55	180	2
19	3	4	42	2.55	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
19	3	4	33	2.55	180	2
3	3	9	10	3.5	180	2
3	3	9	11	3.5	180	2
17	3	9	9	3.5	180	2
17	3	9	16	3.5	180	2
19	3	9	9	3.5	180	2
19	3	9	16	3.5	180	2
33	3	9	10	3.5	180	2
33	3	9	11	3.5	180	2
3	3	11	9	3.625	180	2
3	3	11	15	3.625	180	2
33	3	11	9	3.625	180	2
33	3	11	15	3.625	180	2
17	3	15	2	1.5	180	2
17	3	15	29	1.5	180	2
19	3	15	2	1.5	180	2
19	3	15	29	1.5	180	2
4	3	17	2	4.8	180	2
9	3	17	13	4.8	180	2
15	3	17	9	4.8	180	2
19	3	17	2	4.8	180	2
19	3	17	9	4.8	180	2
19	3	17	13	4.8	180	2
4	3	19	29	2.4	180	2
9	3	19	29	2.4	180	2
15	3	19	29	2.4	180	2
17	3	19	29	2.4	180	2
19	3	19	1	2.4	180	2
19	3	19	29	2.4	180	2
3	3	22	31	0.9	180	2
1	4	4	18	6.65	180	2
1	4	42	18	6.65	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
1	4	4	39	6.65	180	2
1	4	42	39	6.65	180	2
2	4	4	18	6.65	180	2
2	4	42	18	6.65	180	2
2	4	4	39	6.65	180	2
2	4	42	39	6.65	180	2
3	4	4	18	6.65	180	2
3	4	42	18	6.65	180	2
3	4	4	39	6.65	180	2
3	4	42	39	6.65	180	2
33	4	4	18	6.65	180	2
33	4	42	18	6.65	180	2
33	4	4	39	6.65	180	2
33	4	42	39	6.65	180	2
4	4	18	1	1.85	180	2
4	42	18	1	1.85	180	2
4	4	18	2	1.85	180	2
4	42	18	2	1.85	180	2
39	4	18	1	1.85	180	2
39	42	18	1	1.85	180	2
39	4	18	2	1.85	180	2
39	42	18	2	1.85	180	2
1	5	6	16	5.15	180	2
1	5	6	39	5.15	180	2
15	5	6	16	5.15	180	2
15	5	6	39	5.15	180	2
1	5	7	15	5.375	180	2
1	5	7	39	5.375	180	2
15	5	7	15	5.375	180	2
15	5	7	39	5.375	180	2
16	5	7	15	5.375	180	2
16	5	7	39	5.375	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
1	5	15	8	1.4	180	2
1	5	15	29	1.4	180	2
6	5	15	8	1.4	180	2
6	5	15	29	1.4	180	2
7	5	15	8	1.4	180	2
7	5	15	29	1.4	180	2
1	5	16	8	2.4	180	2
7	5	16	8	2.4	180	2
5	6	16	8	2.4	180	2
39	6	16	8	2.4	180	2
15	7	10	1	6.525	180	2
15	7	10	9	6.525	180	2
39	7	10	1	6.525	180	2
39	7	10	9	6.525	180	2
5	7	15	8	1.5	180	2
5	7	15	29	1.5	180	2
10	7	15	11	1.5	180	2
10	7	15	29	1.5	180	2
39	7	15	8	1.5	180	2
39	7	15	11	1.5	180	2
39	7	15	29	1.5	180	2
15	8	15	5	2.325	180	2
15	8	15	7	2.325	180	2
15	8	15	29	2.325	180	2
16	8	15	5	2.325	180	2
16	8	15	7	2.325	180	2
16	8	15	29	2.325	180	2
40	8	15	5	2.325	180	2
40	8	15	7	2.325	180	2
40	8	15	29	2.325	180	2
15	8	16	5	5	180	2
15	8	16	6	5	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
40	8	16	5	5	180	2
40	8	16	6	5	180	2
2	9	9	17	5.45	180	2
2	9	9	18	5.45	180	2
3	9	9	17	5.45	180	2
3	9	9	18	5.45	180	2
16	9	9	17	5.45	180	2
16	9	9	18	5.45	180	2
3	9	10	1	1.675	180	2
3	9	10	7	1.675	180	2
11	9	10	1	1.675	180	2
11	9	10	7	1.675	180	2
3	9	11	3	3	180	2
3	9	11	15	3	180	2
10	9	11	3	3	180	2
10	9	11	15	3	180	2
2	9	16	12	2.55	180	2
3	9	16	12	2.55	180	2
9	9	16	12	2.55	180	2
9	9	16	41	2.55	180	2
9	9	17	3	4.15	180	2
9	9	17	13	4.15	180	2
18	9	17	3	4.15	180	2
18	9	17	13	4.15	180	2
9	9	18	1	1.65	180	2
9	9	18	12	1.65	180	2
9	9	18	41	1.65	180	2
17	9	18	1	1.65	180	2
17	9	18	12	1.65	180	2
17	9	18	41	1.65	180	2
3	11	15	7	1.525	180	2
3	11	15	29	1.525	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
9	11	15	7	1.525	180	2
9	11	15	29	1.525	180	2
18	12	16	9	10	180	2
18	41	16	9	10	180	2
40	12	16	9	10	180	2
40	41	16	9	10	180	2
16	12	18	1	1.7	180	2
16	41	18	1	1.7	180	2
16	12	18	9	1.7	180	2
16	41	18	9	1.7	180	2
40	12	18	1	1.7	180	2
40	41	18	1	1.7	180	2
40	12	18	9	1.7	180	2
40	41	18	9	1.7	180	2
17	13	17	3	6.8	180	2
17	13	17	9	6.8	180	2
40	13	17	3	6.8	180	2
40	13	17	9	6.8	180	2
31	22	28	23	0.25	0	3
31	22	28	25	0.25	0	3
1	23	28	22	0.25	0	3
1	23	28	22	1.2	0	2
1	23	28	23	0.25	0	3
1	23	28	23	1.2	0	2
1	23	28	25	0.25	0	3
1	26	26	1	3.5	0	2
1	26	26	1	0.6	0	3
23	1	18	41	0.9656	68.79	1
23	1	18	41	1.074	15.64	2
23	1	18	41	0.4575	171.58	3
23	1	18	41	0.3092	19.09	4
23	1	18	42	1.2251	146.99	1

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
23	1	18	42	1.6346	16.48	2
23	1	18	42	0.9375	185.88	3
23	1	18	42	0.3103	32.16	4
2	9	16	41	2.55	180	2
3	9	16	41	2.55	180	2

Tabela A.5. TINKER - Parâmetros torcional da abordagem AGN

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
1	1	1	1	0.2	180	1
1	1	1	1	0.25	180	2
1	1	1	1	0.18	0	3
1	1	1	2	0.156	0	3
1	1	1	14	0.156	0	3
1	1	1	18	0.156	0	3
1	1	1	19	0.156	0	3
1	1	1	20	0.156	0	3
1	1	1	22	0.156	0	3
1	1	1	23	0.156	0	3
1	1	1	26	0.156	0	3
1	1	1	34	0.16	0	3
1	1	1	35	0.156	0	3
1	1	1	36	0.156	0	3
1	1	1	38	0.156	0	3
2	1	1	2	0.156	0	3
2	1	1	3	0.156	0	3
2	1	1	5	0.156	0	3
2	1	1	10	0.156	0	3
2	1	1	14	0.156	0	3
2	1	1	20	0.156	0	3
2	1	1	22	0.156	0	3
2	1	1	26	0.156	0	3
2	1	1	27	0.156	0	3
2	1	1	34	0.156	0	3
2	1	1	35	0.156	0	3
2	1	1	38	0.156	0	3
3	1	1	14	0.156	0	3
3	1	1	20	0.156	0	3
3	1	1	35	0.156	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
3	1	1	38	0.156	0	3
5	1	1	14	0.156	0	3
5	1	1	20	0.156	0	3
5	1	1	35	0.156	0	3
5	1	1	38	0.156	0	3
10	1	1	14	0.156	0	3
10	1	1	20	0.156	0	3
10	1	1	35	0.156	0	3
10	1	1	38	0.156	0	3
14	1	1	22	0.156	0	3
14	1	1	26	0.156	0	3
14	1	1	27	0.156	0	3
14	1	1	34	0.156	0	3
14	1	1	35	0.156	0	3
18	1	1	22	0.156	0	3
18	1	1	34	0.156	0	3
18	1	1	35	0.156	0	3
19	1	1	34	0.156	0	3
20	1	1	22	0.156	0	3
20	1	1	26	0.156	0	3
20	1	1	27	0.156	0	3
20	1	1	34	0.156	0	3
20	1	1	35	0.156	0	3
22	1	1	18	0.156	0	3
22	1	1	22	0.144	0	3
22	1	1	22	1.175	0	2
22	1	1	23	0.144	0	3
22	1	1	23	1.175	0	2
22	1	1	34	0.156	0	3
22	1	1	34	0.25	0	1
22	1	1	35	0.156	0	3
22	1	1	36	0.156	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
22	1	1	38	0.156	0	3
23	1	1	23	0.144	0	3
23	1	1	23	1.175	0	2
23	1	1	34	0.156	0	3
23	1	1	34	0.25	0	1
23	1	1	35	0.156	0	3
26	1	1	34	0.156	0	3
26	1	1	35	0.156	0	3
26	1	1	38	0.156	0	3
27	1	1	35	0.156	0	3
27	1	1	38	0.156	0	3
34	1	1	34	0.15	0	3
34	1	1	35	0.156	0	3
34	1	1	36	0.156	0	3
34	1	1	38	0.156	0	3
35	1	1	35	0.156	0	3
35	1	1	36	0.156	0	3
35	1	1	38	0.156	0	3
1	1	2	14	0.07	0	2
1	1	2	14	0.1	0	4
1	1	2	24	0	0	2
1	1	2	25	0	0	2
14	1	2	14	1.7	180	1
14	1	2	14	2	180	2
14	1	2	24	0	0	2
14	1	2	25	0	0	2
20	1	2	14	0	0	2
20	1	2	24	0	0	2
20	1	2	25	0	0	2
34	1	2	14	0	0	2
34	1	2	24	0.8	0	1
34	1	2	24	0.08	180	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
34	1	2	25	0	0	2
35	1	2	14	0	0	2
35	1	2	24	0.8	0	1
35	1	2	24	0.08	180	3
35	1	2	25	0	0	2
1	1	3	3	0	0	2
34	1	3	3	0	0	2
34	1	4	2	0	0	3
34	1	4	4	1.15	0	1
34	1	4	4	0.38	180	3
1	1	5	6	0	0	2
1	1	5	7	0	0	2
1	1	5	15	0	0	2
1	1	5	16	0	0	2
1	1	5	16	0	0	2
34	1	5	6	0	0	2
34	1	5	7	0	0	2
34	1	5	15	0	0	2
34	1	5	16	0	0	2
1	1	10	7	0	0	2
1	1	10	9	0	0	2
34	1	10	7	0	0	2
34	1	10	9	0	0	2
1	1	14	1	0	0	2
1	1	14	2	0.53	0	1
1	1	14	2	0.15	180	3
1	1	14	2	0.5	180	4
1	1	14	29	0	0	2
2	1	14	1	0	0	2
2	1	14	2	0.8	0	1
2	1	14	2	0.85	180	2
2	1	14	29	0	0	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
35	1	14	1	0	0	2
35	1	14	2	0	0	2
35	1	14	29	0	0	2
1	1	18	2	0	0	2
1	1	18	4	0	0	2
1	1	18	9	0	0	2
1	1	18	12	0	0	2
23	1	18	2	0	0	2
23	1	18	4	1.3911	132.1	1
23	1	18	4	1.5847	150.5	2
23	1	18	4	1.9126	175.3	3
23	1	18	4	1.1374	85.6	4
23	1	18	9	0	0	2
23	1	18	12	0.966	69	1
23	1	18	12	1.0037	95.35	2
23	1	18	12	3.1155	301.8	3
23	1	18	12	3.6411	308.7	4
36	1	18	2	0	0	2
36	1	18	4	0	0	2
36	1	18	9	0	0	2
36	1	18	12	0	0	2
1	1	19	3	0	0	3
1	1	19	29	0	0	3
35	1	19	3	0	0	3
35	1	19	29	0	0	3
1	1	20	1	0.156	0	3
1	1	20	29	0.156	0	3
2	1	20	1	0.156	0	3
2	1	20	29	0.156	0	3
35	1	20	1	0.156	0	3
35	1	20	29	0.156	0	3
38	1	20	1	0.156	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
38	1	20	29	0.156	0	3
1	1	22	31	0.025	0	1
1	1	22	31	0.16	0	3
35	1	22	31	0.167	0	3
1	1	23	1	0.383	0	3
1	1	23	1	0.1	180	2
1	1	23	2	0.383	0	3
1	1	23	2	0.8	180	1
1	1	23	28	0.383	0	3
18	1	23	1	0.383	0	3
18	1	23	1	0.65	0	2
23	1	23	1	1.35	180	1
23	1	23	1	0.85	180	2
23	1	23	1	0.1	0	3
35	1	23	1	0.383	0	3
35	1	23	28	0.383	0	3
36	1	23	1	0.383	0	3
1	1	26	1	0.333	0	3
1	1	26	26	0.333	0	3
35	1	26	1	0.333	0	3
35	1	26	26	0.333	0	3
1	1	27	32	0.25	0	3
35	1	27	32	0.25	0	3
15	2	4	1	2.175	180	2
15	2	4	4	2.175	180	2
15	2	4	33	2.175	180	2
24	2	4	1	2.175	180	2
24	2	4	4	2.175	180	2
24	2	4	4	0.3	0	3
24	2	4	33	2.175	180	2
15	2	9	9	3	180	2
15	2	9	16	3	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
24	2	9	9	3	180	2
24	2	9	16	3	180	2
1	2	14	1	2.5	180	2
1	2	14	29	2.5	180	2
24	2	14	1	2.5	180	2
24	2	14	29	2	0	1
24	2	14	29	2.5	180	2
4	2	15	2	1.35	180	2
4	2	15	29	1.35	180	2
9	2	15	3	1.35	180	2
9	2	15	29	1.35	180	2
18	2	15	2	1.35	180	2
18	2	15	29	1.35	180	2
24	2	15	2	1.35	180	2
24	2	15	3	1.35	180	2
24	2	15	29	1.35	180	2
18	2	17	3	4	180	2
24	2	17	3	4	180	2
15	2	18	1	1.45	180	2
15	2	18	4	1.45	180	2
17	2	18	1	1.45	180	2
17	2	18	4	1.45	180	2
24	2	18	1	1.45	180	2
24	2	18	4	1.45	180	2
24	2	22	31	1.9	0	1
24	2	22	31	2.3	180	2
24	2	23	1	1.4	180	1
24	2	23	1	2.7	180	2
1	3	3	3	3.625	180	2
1	3	3	33	3.625	180	2
3	3	3	3	3.625	180	2
3	3	3	9	3.625	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
3	3	3	11	3.625	180	2
3	3	3	22	3.625	180	2
3	3	3	33	3.625	180	2
9	3	3	33	3.625	180	2
11	3	3	33	3.625	180	2
22	3	3	33	3.625	180	2
33	3	3	33	3.625	180	2
17	3	4	4	2.55	180	2
17	3	4	33	2.55	180	2
19	3	4	4	2.55	180	2
19	3	4	33	2.55	180	2
3	3	9	10	3.5	180	2
3	3	9	11	3.5	180	2
17	3	9	9	3.5	180	2
17	3	9	16	3.5	180	2
19	3	9	9	3.5	180	2
19	3	9	16	3.5	180	2
33	3	9	10	3.5	180	2
33	3	9	11	3.5	180	2
3	3	11	9	3.625	180	2
3	3	11	15	3.625	180	2
33	3	11	9	3.625	180	2
33	3	11	15	3.625	180	2
17	3	15	2	1.5	180	2
17	3	15	29	1.5	180	2
19	3	15	2	1.5	180	2
19	3	15	29	1.5	180	2
4	3	17	2	4.8	180	2
9	3	17	13	4.8	180	2
15	3	17	9	4.8	180	2
19	3	17	2	4.8	180	2
19	3	17	9	4.8	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
19	3	17	13	4.8	180	2
4	3	19	29	2.4	180	2
9	3	19	29	2.4	180	2
15	3	19	29	2.4	180	2
17	3	19	29	2.4	180	2
19	3	19	1	2.4	180	2
19	3	19	29	2.4	180	2
3	3	22	31	0.9	180	2
1	4	4	18	6.65	180	2
1	4	4	39	6.65	180	2
2	4	4	18	6.65	180	2
2	4	4	39	6.65	180	2
3	4	4	18	6.65	180	2
3	4	4	39	6.65	180	2
33	4	4	18	6.65	180	2
33	4	4	39	6.65	180	2
4	4	18	1	1.85	180	2
4	4	18	2	1.85	180	2
39	4	18	1	1.85	180	2
39	4	18	2	1.85	180	2
1	5	6	16	5.15	180	2
1	5	6	39	5.15	180	2
15	5	6	16	5.15	180	2
15	5	6	39	5.15	180	2
1	5	7	15	5.375	180	2
1	5	7	39	5.375	180	2
15	5	7	15	5.375	180	2
15	5	7	39	5.375	180	2
16	5	7	15	5.375	180	2
16	5	7	39	5.375	180	2
1	5	15	8	1.4	180	2
1	5	15	29	1.4	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
6	5	15	8	1.4	180	2
6	5	15	29	1.4	180	2
7	5	15	8	1.4	180	2
7	5	15	29	1.4	180	2
1	5	16	8	2.4	180	2
7	5	16	8	2.4	180	2
5	6	16	8	2.4	180	2
39	6	16	8	2.4	180	2
15	7	10	1	6.525	180	2
15	7	10	9	6.525	180	2
39	7	10	1	6.525	180	2
39	7	10	9	6.525	180	2
5	7	15	8	1.5	180	2
5	7	15	29	1.5	180	2
10	7	15	11	1.5	180	2
10	7	15	29	1.5	180	2
39	7	15	8	1.5	180	2
39	7	15	11	1.5	180	2
39	7	15	29	1.5	180	2
15	8	15	5	2.325	180	2
15	8	15	7	2.325	180	2
15	8	15	29	2.325	180	2
16	8	15	5	2.325	180	2
16	8	15	7	2.325	180	2
16	8	15	29	2.325	180	2
40	8	15	5	2.325	180	2
40	8	15	7	2.325	180	2
40	8	15	29	2.325	180	2
15	8	16	5	5	180	2
15	8	16	6	5	180	2
40	8	16	5	5	180	2
40	8	16	6	5	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
2	9	9	17	5.45	180	2
2	9	9	18	5.45	180	2
3	9	9	17	5.45	180	2
3	9	9	18	5.45	180	2
16	9	9	17	5.45	180	2
16	9	9	18	5.45	180	2
3	9	10	1	1.675	180	2
3	9	10	7	1.675	180	2
11	9	10	1	1.675	180	2
11	9	10	7	1.675	180	2
3	9	11	3	3	180	2
3	9	11	15	3	180	2
10	9	11	3	3	180	2
10	9	11	15	3	180	2
2	9	16	12	2.55	180	2
3	9	16	12	2.55	180	2
9	9	16	12	2.55	180	2
9	9	17	3	4.15	180	2
9	9	17	13	4.15	180	2
18	9	17	3	4.15	180	2
18	9	17	13	4.15	180	2
9	9	18	1	1.65	180	2
9	9	18	12	1.65	180	2
17	9	18	1	1.65	180	2
17	9	18	12	1.65	180	2
3	11	15	7	1.525	180	2
3	11	15	29	1.525	180	2
9	11	15	7	1.525	180	2
9	11	15	29	1.525	180	2
18	12	16	9	10	180	2
40	12	16	9	10	180	2
16	12	18	1	1.7	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
16	12	18	9	1.7	180	2
40	12	18	1	1.7	180	2
40	12	18	9	1.7	180	2
17	13	17	3	6.8	180	2
17	13	17	9	6.8	180	2
40	13	17	3	6.8	180	2
40	13	17	9	6.8	180	2
31	22	28	23	0.25	0	3
31	22	28	25	0.25	0	3
1	23	28	22	0.25	0	3
1	23	28	22	1.2	0	2
1	23	28	23	0.25	0	3
1	23	28	23	1.2	0	2
1	23	28	25	0.25	0	3
1	26	26	1	3.5	0	2
1	26	26	1	0.6	0	3
23	1	18	41	0.931	88.44	1
23	1	18	41	1.065	6	2
23	1	18	41	2.9778	301.18	3
23	1	18	41	1.7601	147.11	4
23	1	18	42	1.6639	158	1
23	1	18	42	3.123	301.24	2
23	1	18	42	3.5517	308.4	3
23	1	18	42	0.1332	12.9	4
2	9	16	41	2.55	180	2
3	9	16	41	2.55	180	2

Tabela A.6. TINKER - Parâmetros torcional da abordagem AGW

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
1	1	1	1	0.2	180	1
1	1	1	1	0.25	180	2
1	1	1	1	0.180	0	3
1	1	1	2	0.156	0	3
1	1	1	14	0.156	0	3
1	1	1	18	0.156	0	3
1	1	1	19	0.156	0	3
1	1	1	20	0.156	0	3
1	1	1	22	0.156	0	3
1	1	1	23	0.156	0	3
1	1	1	26	0.156	0	3
1	1	1	34	0.16	0	3
1	1	1	35	0.156	0	3
1	1	1	36	0.156	0	3
1	1	1	38	0.156	0	3
2	1	1	2	0.156	0	3
2	1	1	3	0.156	0	3
2	1	1	5	0.156	0	3
2	1	1	10	0.156	0	3
2	1	1	14	0.156	0	3
2	1	1	20	0.156	0	3
2	1	1	22	0.156	0	3
2	1	1	26	0.156	0	3
2	1	1	27	0.156	0	3
2	1	1	34	0.156	0	3
2	1	1	35	0.156	0	3
2	1	1	38	0.156	0	3
3	1	1	14	0.156	0	3
3	1	1	20	0.156	0	3
3	1	1	35	0.156	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
3	1	1	38	0.156	0	3
5	1	1	14	0.156	0	3
5	1	1	20	0.156	0	3
5	1	1	35	0.156	0	3
5	1	1	38	0.156	0	3
10	1	1	14	0.156	0	3
10	1	1	20	0.156	0	3
10	1	1	35	0.156	0	3
10	1	1	38	0.156	0	3
14	1	1	22	0.156	0	3
14	1	1	26	0.156	0	3
14	1	1	27	0.156	0	3
14	1	1	34	0.156	0	3
14	1	1	35	0.156	0	3
18	1	1	22	0.156	0	3
18	1	1	34	0.156	0	3
18	1	1	35	0.156	0	3
19	1	1	34	0.156	0	3
20	1	1	22	0.156	0	3
20	1	1	26	0.156	0	3
20	1	1	27	0.156	0	3
20	1	1	34	0.156	0	3
20	1	1	35	0.156	0	3
22	1	1	18	0.156	0	3
22	1	1	22	0.144	0	3
22	1	1	22	1.175	0	2
22	1	1	23	0.144	0	3
22	1	1	23	1.175	0	2
22	1	1	34	0.156	0	3
22	1	1	34	0.25	0	1
22	1	1	35	0.156	0	3
22	1	1	36	0.156	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
22	1	1	38	0.156	0	3
23	1	1	23	0.144	0	3
23	1	1	23	1.175	0	2
23	1	1	34	0.156	0	3
23	1	1	34	0.25	0	1
23	1	1	35	0.156	0	3
26	1	1	34	0.156	0	3
26	1	1	35	0.156	0	3
26	1	1	38	0.156	0	3
27	1	1	35	0.156	0	3
27	1	1	38	0.156	0	3
34	1	1	34	0.15	0	3
34	1	1	35	0.156	0	3
34	1	1	36	0.156	0	3
34	1	1	38	0.156	0	3
35	1	1	35	0.156	0	3
35	1	1	36	0.156	0	3
35	1	1	38	0.156	0	3
1	1	2	14	0.07	0	2
1	1	2	14	0.1	0	4
1	1	2	24	0	0	2
1	1	2	25	0	0	2
14	1	2	14	1.7	180	1
14	1	2	14	2	180	2
14	1	2	24	0	0	2
14	1	2	25	0	0	2
20	1	2	14	0	0	2
20	1	2	24	0	0	2
20	1	2	25	0	0	2
34	1	2	14	0	0	2
34	1	2	24	0.8	0	1
34	1	2	24	0.08	180	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
34	1	2	25	0	0	2
35	1	2	14	0	0	2
35	1	2	24	0.8	0	1
35	1	2	24	0.08	180	3
35	1	2	25	0	0	2
1	1	3	3	0	0	2
34	1	3	3	0	0	2
34	1	4	2	0	0	3
34	1	4	4	1.15	0	1
34	1	4	4	0.38	180	3
1	1	5	6	0	0	2
1	1	5	7	0	0	2
1	1	5	15	0	0	2
1	1	5	16	0	0	2
34	1	5	6	0	0	2
34	1	5	7	0	0	2
34	1	5	15	0	0	2
34	1	5	16	0	0	2
1	1	10	7	0	0	2
1	1	10	9	0	0	2
34	1	10	7	0	0	2
34	1	10	9	0	0	2
1	1	14	1	0	0	2
1	1	14	2	0.53	0	1
1	1	14	2	0.15	180	3
1	1	14	2	0.5	180	4
1	1	14	29	0	0	2
2	1	14	1	0	0	2
2	1	14	2	0.8	0	1
2	1	14	2	0.85	180	2
2	1	14	29	0	0	2
35	1	14	1	0	0	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
35	1	14	2	0	0	2
35	1	14	29	0	0	2
1	1	18	2	0	0	2
1	1	18	4	0	0	2
1	1	18	42	0	0	2
1	1	18	9	0	0	2
1	1	18	12	0	0	2
1	1	18	41	0	0	2
23	1	18	2	0	0	2
23	1	18	4	1.9223	182.2	1
23	1	18	4	1.5465	134.09	2
23	1	18	4	1.0151	60.87	3
23	1	18	4	1.3818	97.16	4
23	1	18	9	0	0	2
23	1	18	12	1.1954	113.57	1
23	1	18	12	0.0996	5.76	2
23	1	18	12	0.0004	13.1	3
23	1	18	12	0.7215	4.27	4
36	1	18	2	0	0	2
36	1	18	4	0	0	2
36	1	18	42	0	0	2
36	1	18	9	0	0	2
36	1	18	12	0	0	2
36	1	18	41	0	0	2
1	1	19	3	0	0	3
1	1	19	29	0	0	3
35	1	19	3	0	0	3
35	1	19	29	0	0	3
1	1	20	1	0.156	0	3
1	1	20	29	0.156	0	3
2	1	20	1	0.156	0	3
2	1	20	29	0.156	0	3

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
35	1	20	1	0.156	0	3
35	1	20	29	0.156	0	3
38	1	20	1	0.156	0	3
38	1	20	29	0.156	0	3
1	1	22	31	0.025	0	1
1	1	22	31	0.16	0	3
35	1	22	31	0.167	0	3
1	1	23	1	0.383	0	3
1	1	23	1	0.1	180	2
1	1	23	2	0.383	0	3
1	1	23	2	0.8	180	1
1	1	23	28	0.383	0	3
18	1	23	1	0.383	0	3
18	1	23	1	0.65	0	2
23	1	23	1	1.35	180	1
23	1	23	1	0.85	180	2
23	1	23	1	0.1	0	3
35	1	23	1	0.383	0	3
35	1	23	28	0.383	0	3
36	1	23	1	0.383	0	3
1	1	26	1	0.333	0	3
1	1	26	26	0.333	0	3
35	1	26	1	0.333	0	3
35	1	26	26	0.333	0	3
1	1	27	32	0.25	0	3
35	1	27	32	0.25	0	3
15	2	4	1	2.175	180	2
15	2	4	4	2.175	180	2
15	2	4	42	2.175	180	2
15	2	4	33	2.175	180	2
24	2	4	1	2.175	180	2
24	2	4	4	2.175	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
24	2	4	4	0.3	0	3
24	2	4	42	2.175	180	2
24	2	4	42	0.3	0	3
24	2	4	33	2.175	180	2
15	2	9	9	3	180	2
15	2	9	16	3	180	2
24	2	9	9	3	180	2
24	2	9	16	3	180	2
1	2	14	1	2.5	180	2
1	2	14	29	2.5	180	2
24	2	14	1	2.5	180	2
24	2	14	29	2	0	1
24	2	14	29	2.5	180	2
4	2	15	2	1.35	180	2
4	2	15	29	1.35	180	2
9	2	15	3	1.35	180	2
9	2	15	29	1.35	180	2
18	2	15	2	1.35	180	2
18	2	15	29	1.35	180	2
24	2	15	2	1.35	180	2
24	2	15	3	1.35	180	2
24	2	15	29	1.35	180	2
18	2	17	3	4	180	2
24	2	17	3	4	180	2
15	2	18	1	1.45	180	2
15	2	18	4	1.45	180	2
15	2	18	42	1.45	180	2
17	2	18	1	1.45	180	2
17	2	18	4	1.45	180	2
17	2	18	42	1.45	180	2
24	2	18	1	1.45	180	2
24	2	18	4	1.45	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
24	2	18	42	1.45	180	2
24	2	22	31	1.9	0	1
24	2	22	31	2.3	180	2
24	2	23	1	1.4	180	1
24	2	23	1	2.7	180	2
1	3	3	3	3.625	180	2
1	3	3	33	3.625	180	2
3	3	3	3	3.625	180	2
3	3	3	9	3.625	180	2
3	3	3	11	3.625	180	2
3	3	3	22	3.625	180	2
3	3	3	33	3.625	180	2
9	3	3	33	3.625	180	2
11	3	3	33	3.625	180	2
22	3	3	33	3.625	180	2
33	3	3	33	3.625	180	2
17	3	4	4	2.55	180	2
17	3	4	42	2.55	180	2
17	3	4	33	2.55	180	2
19	3	4	4	2.55	180	2
19	3	4	42	2.55	180	2
19	3	4	33	2.55	180	2
3	3	9	10	3.5	180	2
3	3	9	11	3.5	180	2
17	3	9	9	3.5	180	2
17	3	9	16	3.5	180	2
19	3	9	9	3.5	180	2
19	3	9	16	3.5	180	2
33	3	9	10	3.5	180	2
33	3	9	11	3.5	180	2
3	3	11	9	3.625	180	2
3	3	11	15	3.625	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
33	3	11	9	3.625	180	2
33	3	11	15	3.625	180	2
17	3	15	2	1.5	180	2
17	3	15	29	1.5	180	2
19	3	15	2	1.5	180	2
19	3	15	29	1.5	180	2
4	3	17	2	4.8	180	2
9	3	17	13	4.8	180	2
15	3	17	9	4.8	180	2
19	3	17	2	4.8	180	2
19	3	17	9	4.8	180	2
19	3	17	13	4.8	180	2
4	3	19	29	2.4	180	2
9	3	19	29	2.4	180	2
15	3	19	29	2.4	180	2
17	3	19	29	2.4	180	2
19	3	19	1	2.4	180	2
19	3	19	29	2.4	180	2
3	3	22	31	0.9	180	2
1	4	4	1	1.9	180	1
1	4	4	1	6.65	180	2
1	4	4	18	6.65	180	2
1	4	42	18	6.65	180	2
1	4	4	39	6.65	180	2
1	4	42	39	6.65	180	2
2	4	4	18	6.65	180	2
2	4	42	18	6.65	180	2
2	4	4	39	6.65	180	2
2	4	42	39	6.65	180	2
3	4	4	18	6.65	180	2
3	4	42	18	6.65	180	2
3	4	4	39	6.65	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
3	4	42	39	6.65	180	2
33	4	4	18	6.65	180	2
33	4	42	18	6.65	180	2
33	4	4	39	6.65	180	2
33	4	42	39	6.65	180	2
4	4	18	1	1.85	180	2
4	42	18	1	1.85	180	2
4	4	18	2	1.85	180	2
4	42	18	2	1.85	180	2
39	4	18	1	1.85	180	2
39	42	18	1	1.85	180	2
39	4	18	2	1.85	180	2
39	42	18	2	1.85	180	2
1	5	6	16	5.15	180	2
1	5	6	39	5.15	180	2
15	5	6	16	5.15	180	2
15	5	6	39	5.15	180	2
1	5	7	15	5.375	180	2
1	5	7	39	5.375	180	2
15	5	7	15	5.375	180	2
15	5	7	39	5.375	180	2
16	5	7	15	5.375	180	2
16	5	7	39	5.375	180	2
1	5	15	8	1.4	180	2
1	5	15	29	1.4	180	2
6	5	15	8	1.4	180	2
6	5	15	29	1.4	180	2
7	5	15	8	1.4	180	2
7	5	15	29	1.4	180	2
1	5	16	8	2.4	180	2
7	5	16	8	2.4	180	2
5	6	16	8	2.4	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
39	6	16	8	2.4	180	2
15	7	10	1	6.525	180	2
15	7	10	9	6.525	180	2
39	7	10	1	6.525	180	2
39	7	10	9	6.525	180	2
5	7	15	8	1.5	180	2
5	7	15	29	1.5	180	2
10	7	15	11	1.5	180	2
10	7	15	29	1.5	180	2
39	7	15	8	1.5	180	2
39	7	15	11	1.5	180	2
39	7	15	29	1.5	180	2
15	8	15	5	2.325	180	2
15	8	15	7	2.325	180	2
15	8	15	29	2.325	180	2
16	8	15	5	2.325	180	2
16	8	15	7	2.325	180	2
16	8	15	29	2.325	180	2
40	8	15	5	2.325	180	2
40	8	15	7	2.325	180	2
40	8	15	29	2.325	180	2
15	8	16	5	5	180	2
15	8	16	6	5	180	2
40	8	16	5	5	180	2
40	8	16	6	5	180	2
2	9	9	17	5.45	180	2
2	9	9	18	5.45	180	2
3	9	9	17	5.45	180	2
3	9	9	18	5.45	180	2
16	9	9	17	5.45	180	2
16	9	9	18	5.45	180	2
3	9	10	1	1.675	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
3	9	10	7	1.675	180	2
11	9	10	1	1.675	180	2
11	9	10	7	1.675	180	2
3	9	11	3	3	180	2
3	9	11	15	3	180	2
10	9	11	3	3	180	2
10	9	11	15	3	180	2
2	9	16	12	2.55	180	2
3	9	16	12	2.55	180	2
9	9	16	12	2.55	180	2
9	9	16	41	2.55	180	2
9	9	17	3	4.15	180	2
9	9	17	13	4.15	180	2
18	9	17	3	4.15	180	2
18	9	17	13	4.15	180	2
9	9	18	1	1.65	180	2
9	9	18	12	1.65	180	2
9	9	18	41	1.65	180	2
17	9	18	1	1.65	180	2
17	9	18	12	1.65	180	2
17	9	18	41	1.65	180	2
3	11	15	7	1.525	180	2
3	11	15	29	1.525	180	2
9	11	15	7	1.525	180	2
9	11	15	29	1.525	180	2
18	12	16	9	10	180	2
18	41	16	9	10	180	2
40	12	16	9	10	180	2
40	41	16	9	10	180	2
16	12	18	1	1.7	180	2
16	41	18	1	1.7	180	2
16	12	18	9	1.7	180	2

continua na próxima página

continuação da página anterior

Clásse 1	Clásse 2	Clásse 3	Clásse 4	V_n	φ	n
16	41	18	9	1.7	180	2
40	12	18	1	1.7	180	2
40	41	18	1	1.7	180	2
40	12	18	9	1.7	180	2
40	41	18	9	1.7	180	2
17	13	17	3	6.8	180	2
17	13	17	9	6.8	180	2
40	13	17	3	6.8	180	2
40	13	17	9	6.8	180	2
31	22	28	23	0.25	0	3
31	22	28	25	0.25	0	3
1	23	28	22	0.25	0	3
1	23	28	22	1.2	0	2
1	23	28	23	0.25	0	3
1	23	28	23	1.2	0	2
1	23	28	25	0.25	0	3
1	26	26	1	3.5	0	2
1	26	26	1	0.6	0	3
23	1	18	41	0.966	69	1
23	1	18	41	1.3331	16.14	2
23	1	18	41	0.4315	36.54	3
23	1	18	41	0.9064	19.56	4
23	1	18	42	0.0092	0.87	1
23	1	18	42	2.9787	277.7	2
23	1	18	42	3.5213	278.05	3
23	1	18	42	0.1253	10.03	4
2	9	16	41	2.55	180	2
3	9	16	41	2.55	180	2

Tabela A.7. TINKER - Parâmetros de van der Waals para os tipos de átomos do Amber

Clásse	r	ϵ
1	1.908	0.1094
2	1.908	0.086
3	1.908	0.086
4	1.908	0.086
5	1.908	0.086
6	1.908	0.086
7	1.908	0.086
8	1.908	0.086
9	1.908	0.086
10	1.908	0.086
11	1.908	0.086
12	1.908	0.086
13	1.908	0.086
14	1.824	0.17
15	1.824	0.17
16	1.824	0.17
17	1.824	0.17
18	1.824	0.17
19	1.824	0.17
20	1.875	0.17
21	1.7683	0.152
22	1.721	0.2104
23	1.6837	0.17
24	1.6612	0.21
25	1.6612	0.21
26	2	0.25
27	2	0.25
28	2.1	0.2
29	0.6	0.0157

continua na próxima página

continuação da página anterior

Clásse	r	ϵ
30	0	0
31	0	0
32	0.6	0.0157
33	1.459	0.015
34	1.487	0.0157
35	1.387	0.0157
36	1.287	0.0157
37	1.187	0.0157
38	1.1	0.0157
39	1.409	0.015
40	1.359	0.015
41	1.137	0.0183
42	1.868	0.00277
43	2.658	0.000328
44	2.956	0.00017
45	3.395	0.0000806
46	0.7926	0.8947
47	1.7131	0.459789
48	1.1	0.0125
49	2.1241	0.047096
50	2.47	0.1

Tabela A.8. TINKER - Parâmetros do termo eletrostático de Zgarbová *et. al.*[17]

Átomo	Carga
Adenina	
CT5	0.0558
CT4	0.1065
OS1	-0.3548
CT3	0.2022
OS2	-0.5246
CT2	0.067
OH2	-0.6139
CT1	0.0394
N9	-0.0251
C2	0.1553
NB	-0.6073
CB	0.3053
CA	0.7009
N2	-0.9019
NC	-0.6997
CQ	0.5875
NC	-0.7615
CB	0.0515
H11	0.0679
H12	0.0679
H14	0.1174
H15	0.0615
H16	0.0972
HO2	0.4186
H2	0.2007
H5	0.0473
HH	0.4115
HH	0.4115

continua na próxima página

continuação da página anterior

Átomo	Carga
H5	0.1553
Guanina	
CT5	0.0558
CT4	0.1065
OS1	-0.3548
CT3	0.2022
OS2	-0.5246
CT2	0.067
OH2	-0.6139
CT1	0.0191
N9	0.0492
CK	0.1374
NB	-0.5709
CB2	0.1744
C6	0.477
OO	-0.5597
NA	-0.4787
CA	0.7657
NC	-0.6323
CB1	0.1222
H11	0.0679
H12	0.0679
H14	0.1174
H15	0.0615
H16	0.0972
HO2	0.4186
H2	0.2006
H5	0.164
N2	-0.9672
HH1	0.4364
HH2	0.4364
HH3	0.3424

continua na próxima página

continuação da página anterior

Átomo	Carga
Cistosina	
CT5	0.0558
CT4	0.1065
OS1	-0.3548
CT3	0.2022
OS2	-0.5246
CT2	0.067
OH2	-0.6139
CT1	0.0066
N9	-0.0484
C6	0.7538
OO	-0.6252
NC	-0.7584
CA	0.8185
N2	-0.953
CM	-0.5215
C1	0.0053
H11	0.0679
H12	0.0679
H14	0.1174
H15	0.0615
H16	0.0972
HO2	0.4186
H2	0.2029
HA	0.1928
H4	0.1958
HH1	0.4234
HH2	0.4234
Uracila	
hline CT5	0.0558
H11	0.0679
H12	0.0679

continua na próxima página

continuação da página anterior

Átomo	Carga
CT4	0.1065
H14	0.1174
OS	-0.3548
CT1	0.0674
H2	0.1824
CT3	0.2022
H15	0.0615
CT2	0.067
H16	0.0972
OH2	-0.6139
HO2	0.4186
OS2	-0.5246
N9	0.0418
CC1	0.4687
NA	-0.3549
CC2	0.5952
CM1	-0.3635
CM2	-0.1126
OO1	-0.5477
HH	0.3154
OO2	-0.5761
HA	0.1811
H4	0.2188

Tabela A.9. TINKER - Parâmetros do termo eletrostático da abordagem AGW

Átomo	Carga
Adenina	
CT5	0.107001
CT4	-0.154059
OS1	0.188446
CT3	-0.180004
OH1	0.275435
CT2	-0.174878
OH2	0.274654
CT1	-0.130778
N9	0.093658
C2	-0.037245
NB	0.091022
CB	-0.177932
CA	-0.202527
N2	0.334509
NC	0.073679
CQ	-0.105119
NC	0.104075
CB	-0.162167
H11	0.002958
H12	0.002071
H13	-0.005107
H14	0.019771
H15	0.021925
H16	0.024214
HO2	-0.181824
H2	0.021782
H5	0.018914
HH	-0.005625
HH	-0.018476

continua na próxima página

continuação da página anterior

Átomo	Carga
H5	0.021044
HO1	-0.139419
Guanina	
CT5	0.11670625
CT4	0.262314125
OS1	-0.601846375
CT3	0.1918935
OH1	-0.610297625
CT2	0.254527625
OH2	-0.679903875
CT1	0.534696875
N9	-0.47700125
CK	0.407974
NB	-0.5695065
CB2	0.082285625
C6	0.685142375
OO	-0.478649375
NA	-0.4684015
CA	0.87062825
NC	-0.534481875
CB1	0.30916775
H11	0.035539
H12	0.008545625
H13	0.017891625
H14	0.0288125
H15	0.04423875
H16	-0.007962
HO2	0.321754
H2	-0.000559875
H5	-0.05477825
N2	-0.604212625
HO1	0.31511425

continua na próxima página

continuação da página anterior

Átomo	Carga
HH1	0.190730875
HH2	0.2065485
HH3	0.20309025
Cistosina	
CT5	0.1760125
CT4	0.0741055
OS1	-0.685221375
CT3	0.390590125
OH1	-0.673261625
CT2	0.339592
OH2	-0.75178525
CT1	0.480994625
N9	-0.613533125
C6	0.732818875
OO	-0.5650825
NC	-0.603109875
CA	0.91149475
N2	-0.618010125
CM	-0.601289
C1	0.766095
H11	0.004822625
H12	0.037944375
H13	0.028458375
H14	0.02590375
H15	0.008225375
H16	0.011125
HO2	0.338275
HO1	0.39027775
H2	0.0406235
HA	-0.087719125
H4	0.01595275
HH1	0.218792375

continua na próxima página

continuação da página anterior

Átomo	Carga
HH2	0.206907125
Uracila	
CT5	0.1405
H12	0.0145
H13	0.0435
CT4	0.1793
H14	0.0165
OS	-0.641
CT1	0.6526
H2	0.0024
CT3	0.4826
H15	0.0310
CT2	0.0874
H16	-0.0002
OH2	-0.7209
HO2	0.3327
N9	-0.685
CC1	0.7428
NA	-0.451
CC2	1.0673
CM1	-0.5025
CM2	0.4332
OO1	-0.529
HH	0.2065
OO2	-0.5428
HA	-0.0699
H4	0.0182

Apêndice B

Resultados a Evolução do AG

Abaixo são mostradas as figuras com o melhor e a médias dos indivíduos do Algoritmo Genético com diferentes sementes. O resultado da quarta semente não é mostrado aqui, pois ela esta dentro da tese.

Modelo I - Ligação glicosídica

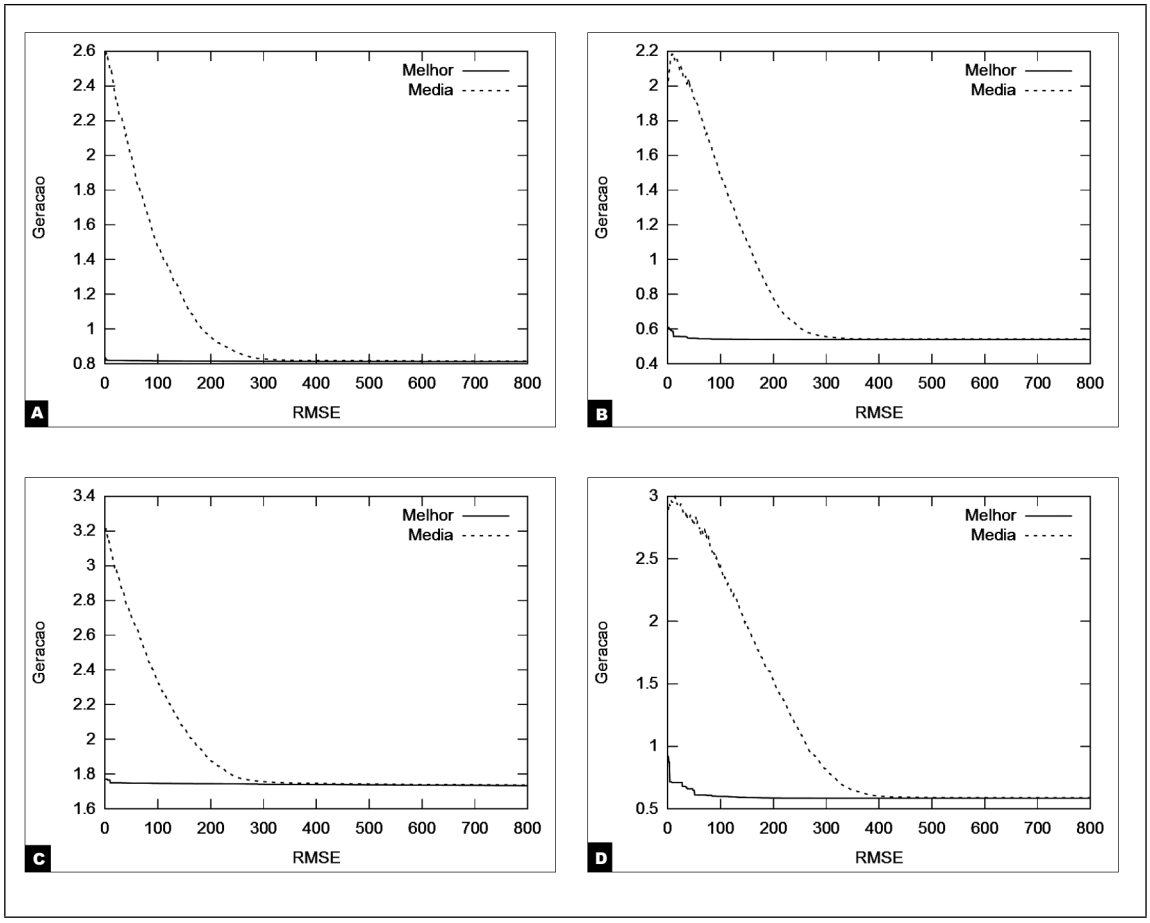


Figura B.1. RMSE dos indivíduos evoluídos pelo AGW, considerando o melhor dos indivíduos e a média da população para a primeira semente do AGW: (A) Adenina, (B) Guanina, (C) Citosina e (D) Uracila.

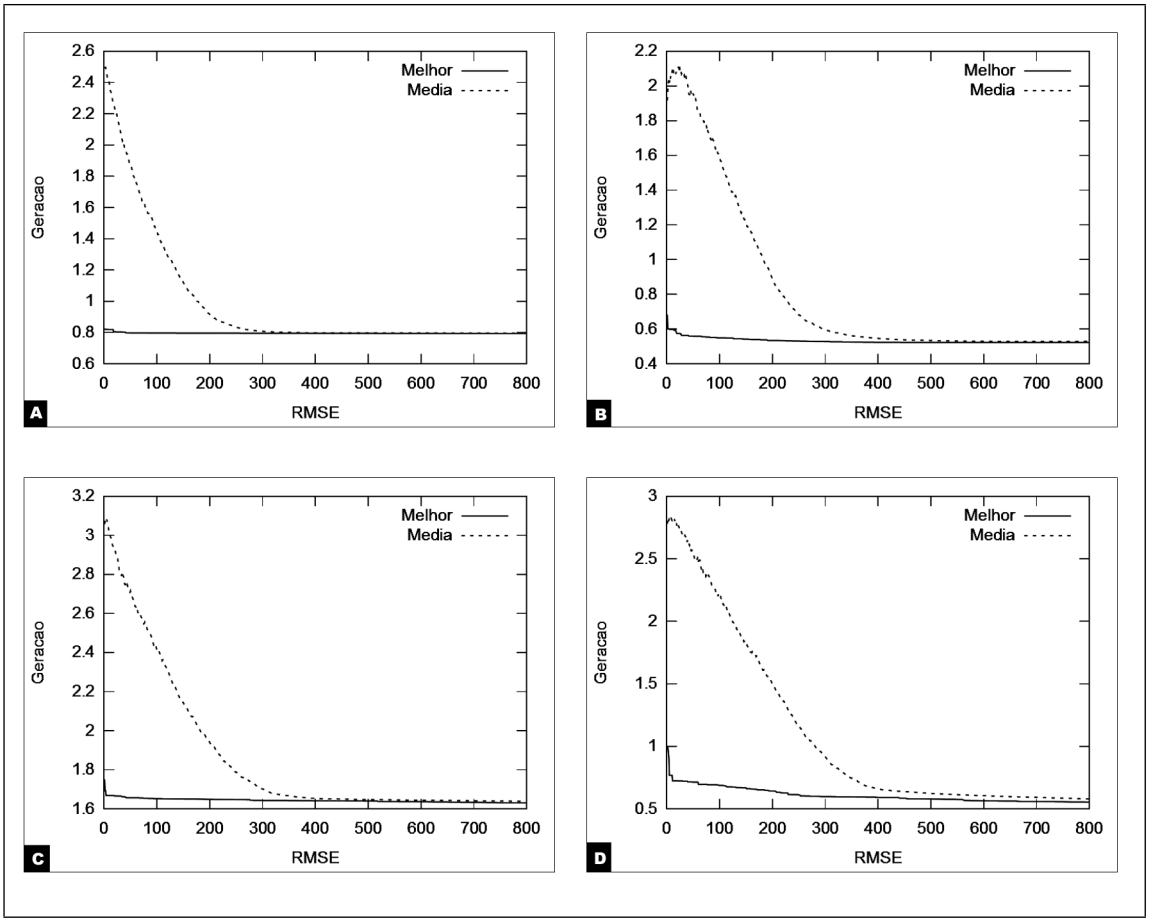


Figura B.2. RMSE dos indivíduos evoluídos pelo AGW, considerando o melhor dos indivíduos e a média da população para a segunda semente do AGW: (A) Adenina, (B) Guanina, (C) Citosina e (D) Uracila.

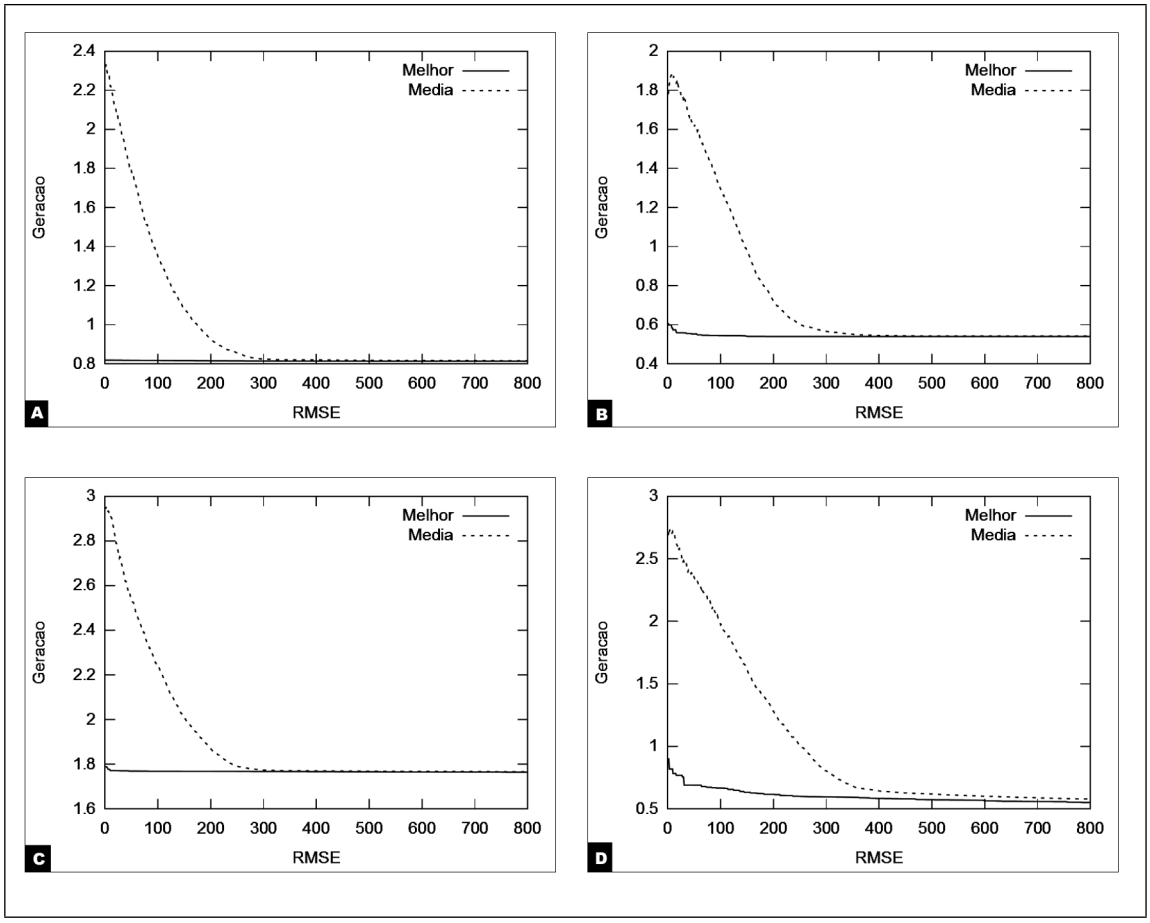


Figura B.3. RMSE dos indivíduos evoluídos pelo AGW, considerando o melhor dos indivíduos e a média da população para a terceira semente do AGW: (A) Adenina, (B) Guanina, (C) Citosina e (D) Uracila.

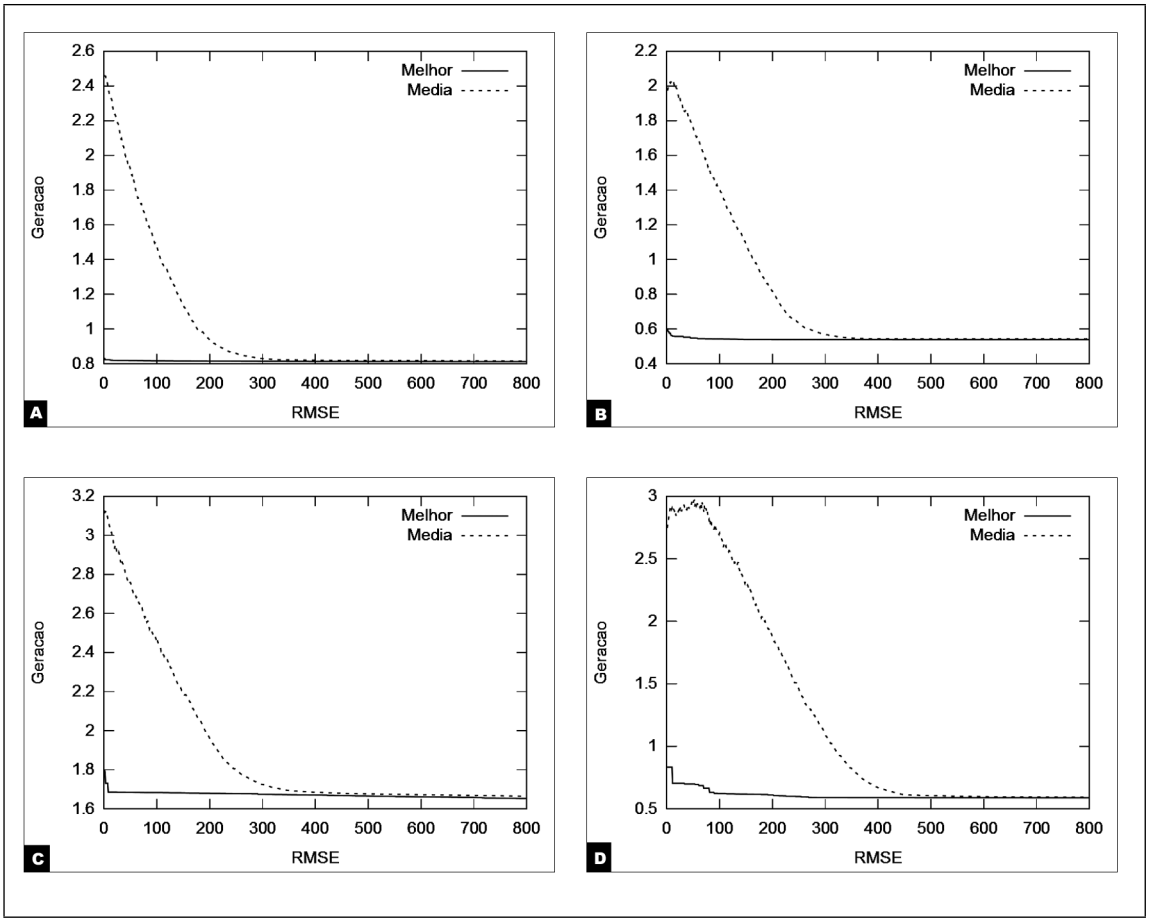


Figura B.4. RMSE dos indivíduos evoluídos pelo AGW, considerando o melhor dos indivíduos e a média da população para a quinta semente do AGW: (A) Adenina, (B) Guanina, (C) Citosina e (D) Uracila.

Modelo II - Ligação fósforo diéster

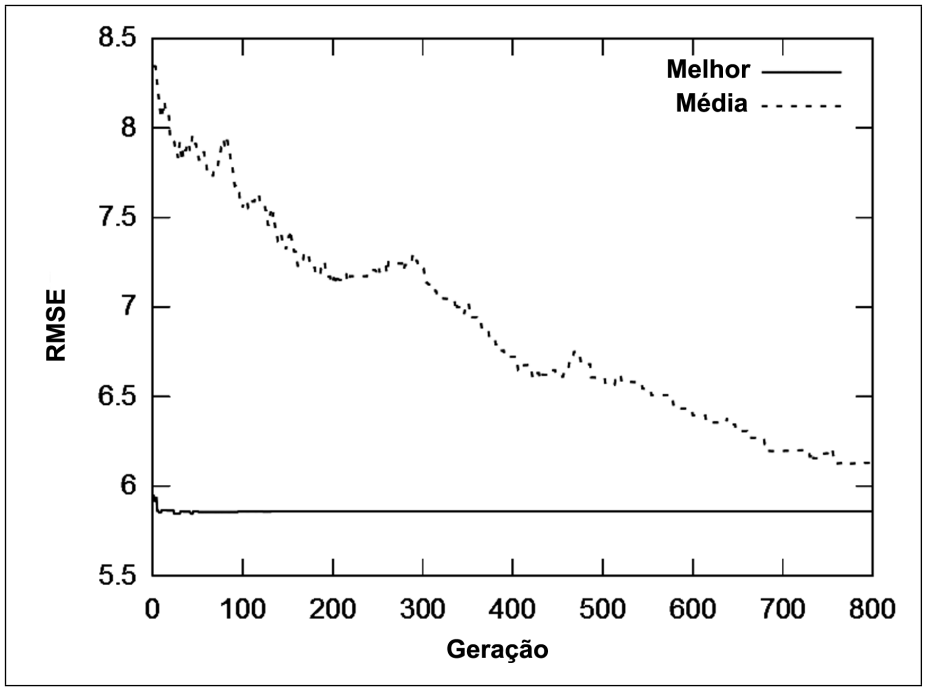


Figura B.5. Análise do RMSE (*fitness* do AG) considerando o melhor indivíduo e a média da população para a primeira semente do AG.

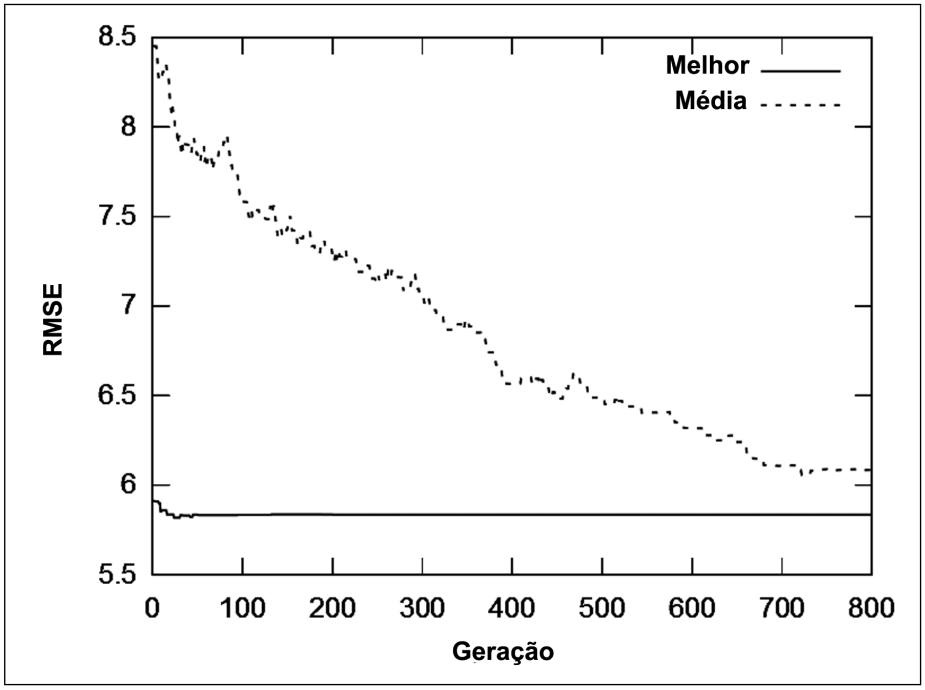


Figura B.6. Análise do RMSE (*fitness* do AG) considerando o melhor indivíduo e a média da população para a segunda semente do AG.

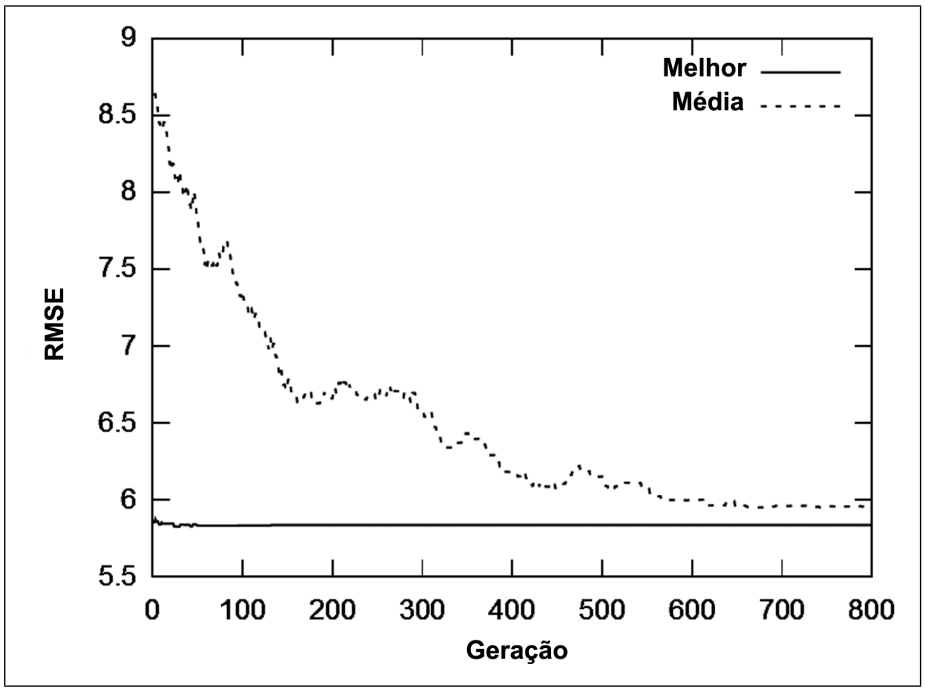


Figura B.7. Análise do RMSE (*fitness* do AG) considerando o melhor indivíduo e a média da população para a terceira semente do AG.

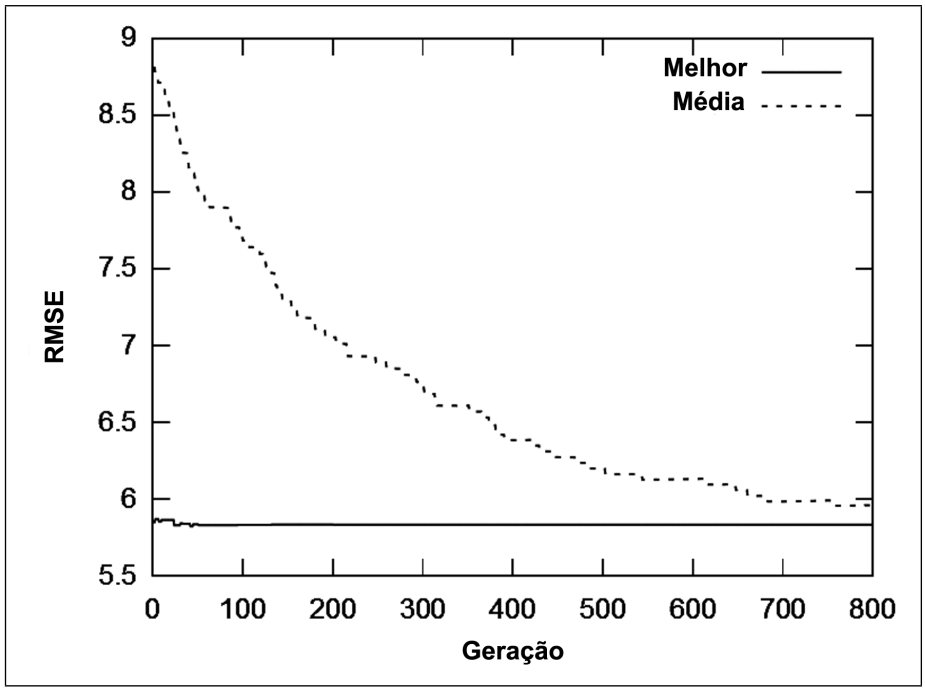


Figura B.8. Análise do RMSE (*fitness* do AG) considerando o melhor indivíduo e a média da população para a quinta semente do AG.

Apêndice C

Descrição das Energias dos Nucleosídeos

Abaixo são mostradas as tabelas referente a energia do campo de força usando as abordagens AGN e AGW contra a referência [17], onde S é a soma dos termo do campo de força que são constantes para todas as rotações.

AGN

Tabela C.1. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGN) para a adenina

	Rotação*	S	E_{tor}	Total
Ref.[17]	0	-55,9815	16,8152	-39,1663
	50	-56,0573	17,0622	-38,9951
	100	-55,7636	17,0521	-38,7115
	150	-55,6756	17,3953	-38,2803
	200	-49,1759	16,7800	-32,3959
	250	-49,5347	17,332	-32,2027
	300	-55,9779	18,3346	-37,6433
	350	-56,1933	18,6703	-37,523
AGN	0	-55,9815	19,7058	-36,2757
	50	-56,0573	19,8444	-36,2129
	100	-55,7636	19,6509	-36,1127
	150	-55,6756	19,6652	-36,0104
	200	-49,1759	15,7369	-33,4390
	250	-49,5347	16,1051	-33,4296
	300	-55,9779	19,6864	-36,2915
	350	-56,1933	19,9085	-36,2848

* Ângulo em graus.

Todas as energias em kcal/mol.

Tabela C.2. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGN) para a guanina

	Rotação*	S	E_{tor}	Total
Ref.[17]	0	-55,9815	19,6211	-76,4260
	50	-56,0573	19,4760	-75,3429
	100	-55,7636	19,2855	-75,4341
	150	-55,6756	18,0764	-74,2437
	200	-49,1759	17,9573	-73,4846
	250	-49,5347	18,3838	-73,8819
	300	-55,9779	19,1635	-76,9259
	350	-56,1933	19,8230	-76,1701
AGN	0	-55,9815	21,9353	-74,1118
	50	-56,0573	22,8405	-71,9784
	100	-55,7636	22,8118	-71,9078
	150	-55,6756	18,0771	-74,2430
	200	-49,1759	17,8263	-73,6156
	250	-49,5347	18,1601	-74,1056
	300	-55,9779	21,4785	-74,6109
	350	-56,1933	21,3981	-74,5950

* Ângulo em graus.

Todas as energias em kcal/mol.

Tabela C.3. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGN) para a citosina

	Rotação*	S	E_{tor}	Total
Ref.[17]	0	-55,9815	19,8249	-122,5886
	50	-56,0573	19,9907	-122,453
	100	-55,7636	20,0570	-122,2594
	150	-55,6756	20,3579	-122,3842
	200	-49,1759	21,3445	-112,1845
	250	-49,5347	17,5290	-116,8043
	300	-55,9779	20,7684	-121,2697
	350	-56,1933	20,8038	-121,6225
AGN	0	-55,9815	19,7029	-122,7106
	50	-56,0573	19,6477	-122,796
	100	-55,7636	19,8832	-122,4332
	150	-55,6756	20,0354	-122,7067
	200	-49,1759	21,3445	-112,1845
	250	-49,5347	17,5290	-116,8043
	300	-55,9779	19,6195	-122,4186
	350	-56,1933	19,5947	-122,8316

* Ângulo em graus.

Todas as energias em kcal/mol.

AGW

Tabela C.4. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGW) para a adenina

	Rotação*	S	E_{tor}	E_{ele}	Total
Ref.[17]	0	15,1903	16,8152	-71,1718	-39,1663
	50	16,0085	17,0622	-72,0658	-38,9951
	100	15,7098	17,0521	-71,4734	-38,7115
	150	15,6457	17,3953	-71,3213	-38,2803
	200	12,8441	16,7800	-62,0200	-32,3959
	250	12,7770	17,3320	-62,3117	-32,2027
	300	14,8969	18,3346	-70,8748	-37,6433
	350	15,7143	18,6730	-71,9076	-37,5203
AGW	0	15,1903	24,0153	-82,2443	-43,0387
	50	16,0085	24,0859	-83,7887	-43,6943
	100	15,7098	23,9017	-82,7812	-43,1697
	150	15,6457	24,3372	-82,5508	-42,5679
	200	12,8441	24,1780	-75,1526	-38,1305
	250	12,7770	24,6654	-75,2490	-37,8066
	300	14,8969	25,6585	-81,9949	-41,4395
	350	15,7143	25,7282	-83,2495	-41,8070

* Ângulo em graus.

Todas as energias em kcal/mol.

Tabela C.5. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGW) para a guanina

	Rotação*	S	E_{tor}	E_{ele}	Total
Ref.[17]	0	15,1133	19,6211	-111,1604	-76,4260
	50	15,4984	19,4760	-110,3173	-75,3429
	100	15,1394	19,2855	-109,8590	-75,4341
	150	14,9613	18,0764	-107,2814	-74,2437
	200	14,3124	17,9573	-105,7543	-73,4846
	250	14,4787	18,3838	-106,7444	-73,8819
	300	15,0863	19,1635	-111,1757	-76,9259
	350	15,0641	19,8230	-111,0572	-76,1701
AGW	0	15,1133	17,3796	-104,6459	-72,1530
	50	15,4984	17,2733	-104,5863	-71,8146
	100	15,1394	17,1923	-104,4589	-72,1272
	150	14,9613	17,9641	-104,8746	-71,9492
	200	14,3124	17,8301	-103,7593	-71,6168
	250	14,4787	18,2514	-104,4984	-71,7683
	300	15,0863	17,0689	-104,6407	-72,4855
	350	15,0641	17,7669	-104,5561	-71,7251

* Ângulo em graus.

Todas as energias em kcal/mol.

Tabela C.6. Energia dos termos do campo de força (ff99χOL) usando os parâmetros da referência [17] e os parâmetros encontrados pela metodologia proposta (AGW) para a citosina

	Rotação*	S	E_{tor}	E_{ele}	Total
Ref.[17]	0	17,9458	19,8249	-160,3593	-122,5886
	50	18,2952	19,9907	-160,7389	-122,4530
	100	18,2490	20,0570	-160,5654	-122,2594
	150	17,8946	20,3579	-160,6367	-122,3842
	200	14,4831	21,3445	-148,0121	-112,1845
	250	14,2803	17,5290	-148,6136	-116,8043
	300	17,8919	20,7684	-159,9300	-121,2697
	350	17,9947	20,8038	-160,4210	-121,6225
AGW	0	18,9235	21,9114	-237,9657	-198,1085
	50	18,2952	22,3146	-238,3698	-197,7600
	100	18,2490	21,6477	-238,0456	-198,1489
	150	17,8946	22,0244	-238,1472	-198,2282
	200	14,4831	21,9725	-227,1682	-190,7126
	250	14,2803	18,1699	-227,3220	-194,8718
	300	17,8919	22,0465	-235,7929	-195,8545
	350	17,9947	23,1412	-238,0511	-196,9152

* Ângulo em graus.

Todas as energias em kcal/mol.

Apêndice D

Código fonte do funcionamento do AG desenvolvido em Java

O código apresentado neste apêndice é para a Adenina, para os outros nucleotídeos é só efetuar a mudança deles.

Codigo principal

```
public class AG {

  public Populacao individuos;
  public String log = "";
  public String log2 = "";
  public Gravar gravar = new Gravar();
  public GravarBusca gravarB = new GravarBusca();

  public AG(String t, String t2){
    try {
      double j=1;
      int sem= 35;
      for (int teste=0; teste<= 7; teste++){
        for (int corre= 0; corre<= 4; corre++){
          if (teste == 0){
            //individuos, geracoes, cruzamento, mutacao, teste
                e corre e semente
```

```
        individuos = new Populacao(200,800,0.95,0.01,teste
            ,corre,sem,t,t2);
        for (int i=0; i< individuos.PopAngd.length; i++){
          log += individuos.PopAngd[i].getIndM();
          log += individuos.PopAngd[i].aux1[0];
          log += individuos.PopAngd[i].aux1[1];
          log += individuos.PopAngd[i].aux1[2];
          log += individuos.PopAngd[i].aux1[3];
          log += individuos.PopAngd[i].aux1[4];
          log += individuos.PopAngd[i].aux1[5];
          log += individuos.PopAngd[i].aux1[6];
          log += individuos.PopAngd[i].aux1[7];
        }
        log += "\n\n---------- FIM TESTE "+teste+"
            ----------\n";
        System.out.println(log);
        gravar.gravaLog(log, teste, corre,t2);
        log = "";
      }//fim if
      if (teste == 1){
        //individuos, geracoes, cruzament e mutacao
        individuos = new Populacao(200,800,0.95,0.1,teste,
            corre,sem,t,t2);
        for (int i=0; i< individuos.PopAngd.length; i++){
          log += individuos.PopAngd[i].getIndM();
          log += individuos.PopAngd[i].aux1[0];
          log += individuos.PopAngd[i].aux1[1];
          log += individuos.PopAngd[i].aux1[2];
          log += individuos.PopAngd[i].aux1[3];
          log += individuos.PopAngd[i].aux1[4];
          log += individuos.PopAngd[i].aux1[5];
          log += individuos.PopAngd[i].aux1[6];
          log += individuos.PopAngd[i].aux1[7];
        }
        log += "\n\n---------- FIM TESTE "+teste+"
            ----------\n";
```

```
        System.out.println(log);
        gravar.gravaLog(log, teste, corre,t2);
        log = "";
      }//fim if
      if (teste == 2){
        individuos = new Populacao(200,800,0.95,0.2,teste,
            corre,sem,t,t2);
        for (int i=0; i< individuos.PopAngd.length; i++){
          log += individuos.PopAngd[i].getIndM();
          log += individuos.PopAngd[i].aux1[0];
          log += individuos.PopAngd[i].aux1[1];
          log += individuos.PopAngd[i].aux1[2];
          log += individuos.PopAngd[i].aux1[3];
          log += individuos.PopAngd[i].aux1[4];
          log += individuos.PopAngd[i].aux1[5];
          log += individuos.PopAngd[i].aux1[6];
          log += individuos.PopAngd[i].aux1[7];
        }
        log += "\n\n---------- FIM TESTE "+teste+"
            ----------\n";
        System.out.println(log);
        gravar.gravaLog(log, teste, corre,t2);
        log = "";
      }//fim if
      if (teste == 3){
        //individuos, geracoes, cruzament e mutacao
        individuos = new Populacao(200,800,0.95,0.25,teste
            ,corre,sem,t,t2);
        for (int i=0; i< individuos.PopAngd.length; i++){
          log += individuos.PopAngd[i].getIndM();
          log += individuos.PopAngd[i].aux1[0];
          log += individuos.PopAngd[i].aux1[1];
          log += individuos.PopAngd[i].aux1[2];
          log += individuos.PopAngd[i].aux1[3];
          log += individuos.PopAngd[i].aux1[4];
          log += individuos.PopAngd[i].aux1[5];
```

```
            log += individuos.PopAngd[i].aux1[6];
            log += individuos.PopAngd[i].aux1[7];
          }
          log += "\n\n---------- FIM TESTE "+teste+"
              ----------\n";
          System.out.println(log);
          gravar.gravaLog(log, teste, corre,t2);
          log = "";
        }//fim if
        if (teste == 4){
          //individuos, geracoes, cruzament e mutacao
          individuos = new Populacao(200,800,0.9,0.01,teste,
              corre,sem,t,t2);
          for (int i=0; i< individuos.PopAngd.length; i++){
            log += individuos.PopAngd[i].getIndM();
            log += individuos.PopAngd[i].aux1[0];
            log += individuos.PopAngd[i].aux1[1];
            log += individuos.PopAngd[i].aux1[2];
            log += individuos.PopAngd[i].aux1[3];
            log += individuos.PopAngd[i].aux1[4];
            log += individuos.PopAngd[i].aux1[5];
            log += individuos.PopAngd[i].aux1[6];
            log += individuos.PopAngd[i].aux1[7];
          }
          log += "\n\n---------- FIM TESTE "+teste+"
              ----------\n";
          System.out.println(log);
          gravar.gravaLog(log, teste, corre,t2);
          log = "";
        }//fim if
        //testes com pc=.8 e pm=.2
        if (teste == 5){
          individuos = new Populacao(200,800,0.9,0.1,teste,
              corre,sem,t,t2);
          for (int i=0; i< individuos.PopAngd.length; i++){
          log += individuos.PopAngd[i].getIndM();
```

```
        log += individuos.PopAngd[i].aux1[0];
        log += individuos.PopAngd[i].aux1[1];
        log += individuos.PopAngd[i].aux1[2];
        log += individuos.PopAngd[i].aux1[3];
        log += individuos.PopAngd[i].aux1[4];
        log += individuos.PopAngd[i].aux1[5];
        log += individuos.PopAngd[i].aux1[6];
        log += individuos.PopAngd[i].aux1[7];
        }
        log += "\n\n---------- FIM TESTE "+teste+"
           ----------\n";
        System.out.println(log);
        gravar.gravaLog(log, teste, corre,t2);
        log = "";
      }//fim if
      if (teste == 6){
        individuos = new Populacao(200,800,0.9,0.2,teste,
           corre,sem,t,t2);
        for (int i=0; i< individuos.PopAngd.length; i++){
          log += individuos.PopAngd[i].getIndM();
          log += individuos.PopAngd[i].aux1[0];
          log += individuos.PopAngd[i].aux1[1];
          log += individuos.PopAngd[i].aux1[2];
          log += individuos.PopAngd[i].aux1[3];
          log += individuos.PopAngd[i].aux1[4];
          log += individuos.PopAngd[i].aux1[5];
          log += individuos.PopAngd[i].aux1[6];
          log += individuos.PopAngd[i].aux1[7];
        }
        log += "\n\n---------- FIM TESTE "+teste+"
           ----------\n";
        System.out.println(log);
        gravar.gravaLog(log, teste, corre,t2);
        log = "";
      }//fim if
      if (teste == 7){
```

```
            individuos = new Populacao(200,800,0.9,0.25,teste,
               corre,sem,t,t2);
            for (int i=0; i< individuos.PopAngd.length; i++){
              log += individuos.PopAngd[i].getIndM();
              log += individuos.PopAngd[i].aux1[0];
              log += individuos.PopAngd[i].aux1[1];
              log += individuos.PopAngd[i].aux1[2];
              log += individuos.PopAngd[i].aux1[3];
              log += individuos.PopAngd[i].aux1[4];
              log += individuos.PopAngd[i].aux1[5];
              log += individuos.PopAngd[i].aux1[6];
              log += individuos.PopAngd[i].aux1[7];
            }
            log += "\n\n---------- FIM TESTE "+teste+"
               ----------\n";
            System.out.println(log);
            gravar.gravaLog(log, teste, corre,t2);
            log = "";
          }//fim if
          sem= sem+5;
        }//fim for
      }//fim for
    } catch (Exception e) {
      e.printStackTrace();
    }

  }
  public static void main(String[] args) {
    String t1 = "";
    Scanner sc1 = new Scanner(System.in);
    String t2 = "";
    Scanner sc2 = new Scanner(System.in);

    System.out.println("Digite o endereco do arquivo:");
    t1= sc1.next();
    System.out.println("Digite o endereco de gravacao:");
```

```
    t2= sc2.next();
    AG teste = new AG(t1,t2);

  }
}
```

Codigo da População

```
public class Populacao implements Cloneable {
  Atomos objAtomos= new Atomos();
  IndividuoAngd[] PopAngd;
  IndividuoAngd[] filhosAngd;
  IndividuoAngd paradaAngd = new IndividuoAngd();

  int p = 0;
  int contDistC= 0;
  int contAngC= 0;
  int contAngdC= 0;
  int contCargaC= 0;
  int contLjC= 0;
  int contC= 0;
  int contM= 0;
  public Vector melhores = new Vector();
  public int[] selecionadosAngd;
  int nTor = 8;
  public double [] torneioAngd = new double [nTor];
  double pm = 0.0;
  double pc = 0.0;
  IndividuoAngd indAngd= new IndividuoAngd();
  public IndividuoAngd[] elitismoAngd;
  private int tamPop;
  public int gera = 0;
  public String log = "";
  public String log2 = "";
  public double[] mediaParada;
  public double mediaFitness=0;
```

```
int[] sementes;
public Gravar gravar = new Gravar();
long soma = 0;

public Populacao(int tamanho, int geracoes, double pc,
   double pm, int teste, int corre, int semente, String s,
   String s2) throws Exception{

  gera= geracoes;
  sementes = new int[tamanho];
  System.out.println("\nIniciado o processo de Geracao da
     Populacao...");
  setTamPop(tamanho);
  System.out.println("----------> Iniciando o processo de
     Geracao da Populacao Inicial...");
  gerarPopInicial(semente,s);
  objAtomos.getSemente(semente);
  System.out.println("gerado Pop inicial, OK!!!");
  elitismoAngd = new IndividuoAngd[PopAngd.length];
  int z=0;
  double mediaTeste2=0.0;

  int i =1;
  do{
    long start = System.currentTimeMillis();
    selecaoPop();
    reproducao(PopAngd,pc,semente,s);
    mutacaoPop(i, pm, semente,s);
    //elitismo alicado no metodo abaixo
    calcularFitPop(i);
    //aplicacao de elitismo
    elitismoPop(i,s);
    double m= PopAngd[0].getFit();
    double r= PopAngd[0].getFitD();
    log += "\n"+i+"\t"
      + "Melhor Fitness\t"+m+"\t"
```

```
      + "Melhor Erro\t"+r+"\t"
      + "Melhor Angd\t"+PopAngd[0].getFit();
    contC= contDistC+contAngC+contAngdC+contCargaC+contLjC;;
    double p= PopAngd[PopAngd.length-1].getFitD();
    double e= PopAngd[PopAngd.length-1].getFitD();
    log += "\tMedia\t"+mediaFitness+"\tPior Fitness\t"+p;
    log += "\tPior Erro\t"+e;
    log +="\tDesvio Padrao:\t"+desvioP();
    log +="\t Mutacao:\t"+contM+"\t Cruzamento:\t"+contC;
    long delay = System.currentTimeMillis() - start;
    log += "\t Demorou:\t"+delay+ " milissegundos";
    System.out.println(log);
    gravar.gravaLog(log,teste,corre,s2);
    log = "";
    log2 = "";
    soma =+ System.currentTimeMillis() + start;
    if (i>2){
      if ((mediaParada[0]-mediaParada[1]) < 0.5){
        z++;
      }else{
        z=0;
      }
    }else{
    }
    i++;
  }while ((PopAngd[0].getFitD()!=0)&&(i<geracoes));

  //}//fim for
  ordenarAngd(PopAngd);
  melhores.addElement(PopAngd[0]);
  melhores.addElement(PopAngd[1]);

}//fim Populacao
public void ordenarAngd(IndividuoAngd[] PopAuxAngd){

  IndividuoAngd tempAngd = new IndividuoAngd();
```

```
      for (int i = 0; i < PopAuxAngd.length; i++){
    for (int j = i+1; j < PopAuxAngd.length; j++){
      if ((PopAuxAngd[i]!=null)&&(PopAuxAngd[j]!=null)){
        if (PopAuxAngd[i].getFitD() >= PopAuxAngd[j].getFitD
          ()){
          tempAngd = PopAuxAngd[i];
          PopAuxAngd[i] = PopAuxAngd[j];
          PopAuxAngd[j] = tempAngd;
        }
      }
    }//fim for
  }//fim for
}//fim ordenar
public int ordenarTorAngd(int[] auxAngd){
  double temp = 0.0;
  int aux2 = 0;
  temp=0;
  aux2=0;
  for (int i = 0; i < torneioAngd.length; i++){
    for (int j = i+1; j < torneioAngd.length; j++){
      if (torneioAngd[i] >= torneioAngd[j]){
        temp = torneioAngd[i];
        torneioAngd[i] = torneioAngd[j];
        torneioAngd[j] = temp;
        aux2 = auxAngd[i];
        auxAngd[i] = auxAngd[j];
        auxAngd[j]= aux2;
      }
    }//fim for
  }//fim for
  return auxAngd[0];

}
public void elitismoPop(int i, String s) throws JXLException
   , IOException{
```

```
    //int aux = 0;
    double aux1 = 0.0;
    double aux = 0.0;

    ordenarAngd(PopAngd);
    //compara o elitismo da primeira geracao com o outro
    if (i>=2){
      aux1=0;
      aux=0;
      aux1= elitismoAngd[0].getFitD();
      aux= PopAngd[0].getFitD();
      if (aux != aux1){
        if (aux <= aux1){
          //ultimo elemento
          elitismoAngd[PopAngd.length-1]= PopAngd[0];
          ordenarAngd(elitismoAngd);
        }else{
          PopAngd[PopAngd.length-1]= elitismoAngd[0];
          ordenarAngd(PopAngd);
        }//fim else
      }
    //elitismo da primeira geracao
    }else{
      if ((i==1) && (elitismoAngd[0]== null)){
        //copia o melhor da populacao para o vetor elitismo,
            estava com problemas de referencia qnd mudava um o
            outro tbm mudava
        elitismoAngd[0]= setIndividuoAngd(PopAngd[0],s);
      }//fim if
    }
    ordenarAngd(elitismoAngd);

  }//fim elitismo
  public IndividuoAngd setIndividuoAngd(IndividuoAngd
     melhorPop, String s) throws JXLException, IOException{
    IndividuoAngd aux = new IndividuoAngd();
```

```
    double fit = 0.0;
    double erro = 0.0;
    int semente=0;

    //copiando a fitness
    fit= melhorPop.getFitD();
    aux.setFit(fit);
    erro= melhorPop.getFitD();
    aux.setErro(erro);
    //compiando a molecula
    aux.completarMol(s);
    aux.cruzaPeri= melhorPop.cruzaPeri;
    aux.cruzaPeriRep= melhorPop.cruzaPeriRep;
    aux.cruzaVn= melhorPop.cruzaVn;
    aux.cruzaVnRep= melhorPop.cruzaVnRep;
    aux.indMutacao1= melhorPop.indMutacao1;
    aux.indMutacao2= melhorPop.indMutacao2;
    aux.indMutacao3= melhorPop.indMutacao3;
    aux.indMutacao3Rep= melhorPop.indMutacao3Rep;
    aux.indMutacao4= melhorPop.indMutacao4;
    aux.indMutacao5= melhorPop.indMutacao5;
    for (int j=0; j< aux.vnDInd.length; j++){
      aux.vnDInd[j]= melhorPop.vnDInd[j];
      aux.periDInd[j]= melhorPop.periDInd[j];
    }
    for (int i=0; i< aux.vnD.length; i++){
      aux.vnD[i]= melhorPop.vnD[i];
      aux.periD[i]= melhorPop.periD[i];
    }//fim for
    return aux;
  }
  public void setTamPop(int tam) throws Exception {
    this.tamPop = tam;
    PopAngd = new IndividuoAngd[tamPop];
    mediaParada = new double[2];
```

```
    for (int i = 0; i < PopAngd.length; i++){
      PopAngd[i] = new IndividuoAngd();
    }//fim for i
    selecionadosAngd = new int[tam/2];
    filhosAngd = new IndividuoAngd[tam/2];

  }//fim setTamPop
  private void selecaoPop(){//int auxger) {
    int ind = 0;
    int [] aux = new int [nTor];
    int j=0;

    ind=0;
    j=0;
    while (j<= selecionadosAngd.length-1){
      //fazer o torneio entre o nTor
      for (int i=0; i< nTor; i++){
        aux[i]= (int) (objAtomos.random() * PopAngd.length);
        torneioAngd[i]= PopAngd[aux[i]].getFit();
       }
      ind = ordenarTorAngd(aux);
       selecionadosAngd[j]= ind;
      j=j+1;
    }//fim do while

  }//fim do metodo selecaoPop
  public void gerarPopInicial(int semente, String s) throws
     Exception {

    for (int i = 0; i < PopAngd.length; i++) {
      PopAngd[i].completarMol(s);
      if (i< (PopAngd.length/10)){
        PopAngd[i].gerarIndAngdCp(semente);
        PopAngd[i].calculaFit(semente);
//          PopAngd[i].calculaFitness(semente);
      }else{
```

```
        PopAngd[i].gerarIndAngd(semente);
        PopAngd[i].calculaFit(semente);
       }
      semente= semente+100;
    }//fim for

  }//fim do metodo gerarPopInicial
  public void mostraPop(int g,IndividuoAngd[] popAuxAngd, int
     teste, int corre) throws Exception {

    ordenarAngd(popAuxAngd);
    for (int i = 0; i < PopAngd.length; i++) {
      log +=popAuxAngd[i].getIndM();
    }//fim for

  }//fim do metodo mostrarPop
  public double desvioP(){
    int divAngd = PopAngd.length;
    int div = divAngd;//+divHbond;
    double d= 0.0;
    double[] vetorAngd= new double[PopAngd.length];

    for (int i= 0; i< PopAngd.length; i++) {
      //tem que ser modulo da fitness - a media
      vetorAngd[i]= Math.abs(PopAngd[i].getFit() -
         mediaFitness);
    }//fim for
    d = d/div;
    return d;

  }//fim devioP
  public void calcularFitPop(int indice) {
    double aux= 0.0;
    double mediaFit = 0;
    int divAngd = PopAngd.length;
    int div= divAngd;
```

```
    //como em tds os outros metodos jah em calculado a fitness
       , nesse metodo soh calcula a media
    if (div != 0){
      for (int i = 0; i < PopAngd.length; i++) {
        aux = PopAngd[i].getFitD();
        mediaFit= mediaFit + aux;
      }//fim for
    }//fim if
    if((indice % 2)==0){
      mediaParada[1] = (mediaFit/div);
    }else{
      mediaParada[0] = (mediaFit/div);
    }
    mediaFitness= mediaFit/div;

  }//fim do metodo calcularFitPop
  public void reproducao(IndividuoAngd[] PopAuxAngd, double pc
     , int semente, String s) throws Exception{
    int indice1= 0;
    int indice2= 0;
    double aux = 0.0;

    for (int j = 0; j < selecionadosAngd.length; j=j+2) {
      PopAuxAngd[j].getSemente(semente);
      aux= PopAuxAngd[j].random();
      //probabilidade de cruzar eh o pq passado no inicio do
         programa
      if (aux<= pc){
        IndividuoAngd a= new IndividuoAngd();
        IndividuoAngd b = new IndividuoAngd();
        if(j>selecionadosAngd.length-2){
          //problema de referencia de vetores
          filhosAngd[j]= setIndividuoAngd(PopAuxAngd[
             selecionadosAngd[j]],s);
          PopAngd[j]= setIndividuoAngd(filhosAngd[j],s);
        }else{
```

```
            filhosAngd[j]= setIndividuoAngd(PopAuxAngd[
                selecionadosAngd[j]],s);
            a= filhosAngd[j];
            indice1 = selecionadosAngd[j];
            filhosAngd[j+1]= setIndividuoAngd(PopAuxAngd[
                selecionadosAngd[j+1]],s);
            b= filhosAngd[j+1];
            indice2 = selecionadosAngd[j+1];
            crossoverAngd(filhosAngd[j],filhosAngd[j+1],semente)
                ;
            PopAngd[indice1]= setIndividuoAngd(filhosAngd[j],s);
            PopAngd[indice2]= setIndividuoAngd(filhosAngd[j+1],s
                );
          }//fim else
        }//fim if
        semente++;
      }//fim for
    }//fim reproducao

    private void crossoverAngd(IndividuoAngd filho1,
       IndividuoAngd filho2, int semente) throws JXLException,
       IOException{
      IndividuoAngd a= filho1;
      IndividuoAngd b = filho2;
      double ind= 0.0;
      //cruzamento
      filho1.getSemente(semente);
      int cortel = (int) (filho1.random()*3);
      double[] aux1= new double[filho1.vnD.length];
      int[] aux2= new int[filho1.vnD.length];

      for (int w =0; w<= cortel; w++){
      double auxRep= b.vnDInd[w];
          b.vnDInd[w] = a.vnDInd[w];
          a.vnDInd[w] = auxRep;
          auxRep = b.periDInd[w];
```

```
        b.periDInd[w] = a.periDInd[w];
        a.periDInd[w] = auxRep;
    }
        a.calculaFit(10);
        b.calculaFit(10);
    if ((a.getFitD()<=filho1.getFitD())||(b.getFitD()<=filho2.
        getFitD())){
      contAngdC++;
    }
    filho1= a;
    filho2= b;

}//fim do metodo Reproducao

private void mutacaoPop(int pos, double pm, int semente,
    String s) {
    IndividuoAngd auxAngd = new IndividuoAngd();
    double ind= 0.0;

    try {
      double auxM = 0.0;

      ordenarAngd(PopAngd);
      auxM=0;
      for (int i = 1; i < PopAngd.length; i++) {
        PopAngd[i].getSemente(semente);
        auxM = PopAngd[i].random();
        if (auxM<= pm){
          auxAngd= setIndividuoAngd(PopAngd[i],s);
          PopAngd[i].mutacao(pm,pos,gera,semente);
           PopAngd[i].calculaFit(10);
          if (PopAngd[i].getFitD()<=auxAngd.getFitD()){
            contM++;
          }
        }//fim if
        semente++;
```

```
    }//fim for
  } catch (Exception e) {
    e.printStackTrace();
  }
 }//fim do metodo mutacaoPop
}//fim classe Populacao
```

Código da Indivíduo

```
import java.io.IOException;
import java.util.Random;
import java.util.Vector;

import jxl.JXLException;

public class IndividuoAngd implements Comparable, Cloneable{
 Atomos objAtomos= new Atomos();
 private int qntdcol= 12;
 private int qntdlinha= 7;
 double[] nInd= new double [4];
 double[] periDInd= {-1,-1,-1,-1};
 double[] vnDInd= {-1,-1,-1,-1};
 double[] pathD=
    {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};

 //ADENINA
 double[] n=
    {3,3,3,2,3,3,3,1,2,3,2,3,3,3,2,3,3,1,3,3,2,3,3,3,3,3,3,3,
     1,2,3,4,
     0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2

};
```

```
  //ADENINA
   double[] periD=
       {0,0,0,180,0,0,0,180,180,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

       //AMBER
       69,16,172,19,
       0,0,0,0,0,180,180,180,180,180,180,180,180,180,180,180,180,180,1

       180,180,180,180,180,180,180,180,180,180,180,180,180,180,180
 };
  //Adenina
  double[] vnD=
     {0.156,0.156,0.156,0.1,0.383,0.383,0.156,0.2,0.25,0.18,1.175,0.144,0

       //AMBER
       0.966,1.074,0.458,0.309,
       0,0,0,0,0,1.7,1.7,1.7,1.7,1.65,1.65,1.65,1.65,10,10,2.55,2.55,3.5,3

       5.45,2.4,2.4,4.8,4.8,6.8,6.8,6.8,6.8,4.15,4.15
};
  //Adenina
  String[] vnS= {"H1–CT–CT–OS","H1–CT–CT–CT","H1–CT–CT–H1","CT
     –CT–OS–CT","CT–CT–OS–CT","H1–CT–OS–CT","CT–CT–CT–OH","CT–
     CT–CT–CT","CT–CT–CT–CT","CT–CT–CT–CT","OS–CT–CT–OH",
       "OS–CT–CT–OH","OS–CT–CT–CT","H1–CT–CT–OH","CT–OS–CT–N9
           ","CT–OS–CT–N9","CT–OS–CT–H2","CT–CT–OH–HO","CT–CT–OH
           –HO","H1–CT–OH–HO","OH–CT–CT–OH","OH–CT–CT–OH","CT–CT
           –CT–N9",
       "CT–CT–CT–H2","OH–CT–CT–N9","OH–CT–CT–H2","H1–CT–CT–N9
           ","H1–CT–CT–H2","OS–CT–N9–C2","OS–CT–N9–C2","OS–CT–N9
           –C2","OS–CT–N9–C2","OS–CT–N9–CB","CT–CT–N9–C2","CT–CT
           –N9–CB",
       "H2–CT–N9–C2","H2–CT–N9–CB","CT–N9–C2–NB","CT–N9–C2–H5
           ","CB–N9–C2–NB","CB–N9–C2–H5","CT–N9–CB–CB","CT–N9–CB
           –NC","C2–N9–CB–CB","C2–N9–CB–NC","N9–C2–NB–CB","H5–C2
           –NB–CB",
```

```
        "C2-NB-CB-CA","C2-NB-CB-CB","NB-CB-CA-N2","NB-CB-CA-NC
            ","CB-CB-CA-N2","CB-CB-CA-NC","NB-CB-CB-N9","NB-CB-CB
            -NC","CA-CB-CB-N9","CA-CB-CB-NC","CB-CA-N2-HH","NC-CA
            -N2-HH",
        "CB-CA-NC-CQ","N2-CA-NC-CQ","CA-NC-CQ-NC","CA-NC-CQ-H5
            ","NC-CQ-NC-CB","H5-CQ-NC-CB","CQ-NC-CB-N9","CQ-NC-CB
            -CB"
    };
    //vetores de string com os atomos da tabela da cte
    String[][] molS= new String[qntdlinha][qntdcol];
    double[][] molA= new double[4][11];
    public String[] vetorInd;
    int sem= 0;
    double fit= 0.0;
    double erro= 0.0;
    Random rand;
    int num = 1;
    int indice = 0;
    //indice que vao ser mutados
    int indMutacao1 =0;
    int indMutacao2 =0;
    int indMutacao3 =0;
    //esse 3 eh pq as estruturas se repetem no tamanho 4 da
       tabela de parametros
    int[] indMutacao3Rep = new int[4];
    int indMutacao4 =0;
    int indMutacao5 =0;
    int[] cruzaVnRep= new int[4];
    int[] cruzaPeriRep= new int[4];
    //soh tenho 5 diedros
    int[] cruzaVn= new int[5];
    int[] cruzaPeri= new int[5];
    double[] angulos= new double[7];
    String[] torS= {"0","1","2","3","4","5","6","7"};
    public GravarEnergDiedro garvar = new GravarEnergDiedro();
    //variavel que indica a repeticao do vetor estrutura vnS
```

```
int rep= 0;
int repPeri= 0;
double paramaux = 0.0;
//valores das energias das rotacoes
double[] aux1= new double[torS.length];
//valores das energias das rotacoes auxiliar fitness
double[] aux2= new double[torS.length];

public IndividuoAngd(){
  //colocar o numero de individuos entre parenteses
  vetorRad= "";
  vetorMol= "";
}//fim IndividuoConstrutor

public double random(){
  double r = 0.0;

  //-1<x<1
  r= rand.nextGaussian();
  if (r>1){
    r=r-1;
  }else if (r<-1){
    r=r+2;
  }else if (r<0){
    r=r+1;
  }
  return r;

}
public Random getSemente(int s1){
  long seed = s1;
  rand= new Random (s1);
  return rand;

}
//esse metodo esta no gerarPopInicial()
```

```
public void completarMol (String s) throws JXLException,
    IOException{
   //Adenina
     molS[0][0]="";
   molS[0][1]="";
   molS[0][2]="";
   molS[0][3]="";
   molS[0][4]="";
   molS[0][5]="";
   molS[0][6]="";
   molS[0][7]="";
   molS[0][8]="HHA000111000";
   molS[0][9]="N2A000111100";
   molS[0][10]="HHA000211000";
   molS[0][11]="";
   molS[1][0]="";
   molS[1][1]="";
   molS[1][2]="";
   molS[1][3]="";
   molS[1][4]="";
   molS[1][5]="";
   molS[1][6]="";
   molS[1][7]="NBA000110011";
   molS[1][8]="";
   molS[1][9]="CAA000110111";
   molS[1][10]="";
   molS[1][11]="";
   molS[2][0]="";
   molS[2][1]="H1R100300100";
   molS[2][2]="H1R100400100";
   molS[2][3]="OSR100100011";
   molS[2][4]="H2R100100100";
   molS[2][5]="H5A000111000";
   molS[2][6]="C2A000111010";
   molS[2][7]="";
   molS[2][8]="CBA000110111";
```

```
molS[2][9]="";
molS[2][10]="NCA000110101";
molS[2][11]="";
molS[3][0]="H1R100101000";
molS[3][1]="CTR100501100";
molS[3][2]="CTR100401110";
molS[3][3]="";
molS[3][4]="CTR100101101";
molS[3][5]="X00000001000";
molS[3][6]="X00000001000";
molS[3][7]="N9A000111001";
molS[3][8]="CBA000211110";
molS[3][9]="";
molS[3][10]="CQA000111101";
molS[3][11]="H5A000211000";
molS[4][0]="";
molS[4][1]="H1R100200100";
molS[4][2]="H1R100501000";
molS[4][3]="CTR100301101";
molS[4][4]="CTR100201100";
molS[4][5]="H1R100601000";
molS[4][6]="";
molS[4][7]="";
molS[4][8]="";
molS[4][9]="NCA000210011";
molS[4][10]="";
molS[4][11]="";
molS[5][0]="";
molS[5][1]="";
molS[5][2]="HOR100101000";
molS[5][3]="OHR100101100";
molS[5][4]="OHR100201100";
molS[5][5]="HOR100201000";
molS[5][6]="";
molS[5][7]="";
molS[5][8]="";
```

```
    molS[5][9]="";
    molS[5][10]="";
    molS[5][11]="";
    molS[6][0]="";
    molS[6][1]="";
    molS[6][2]="";
    molS[6][3]="";
    molS[6][4]="";
    molS[6][5]="";
    molS[6][6]="";
    molS[6][7]="";
    molS[6][8]="";
    molS[6][9]="";
    molS[6][10]="";
    molS[6][11]="";

    //le Arquivo para eu ter o atomos[i][j]
    objAtomos.leXls(s);
    //completa os x,y e z dos atomos
    for(int r= 0; r<torS.length; r++){
      for (int i= 0; i<qntdlinha; i++){
        for(int j= 0; j< qntdcol; j++){
          if (molS[i][j]!=""){
            //para Uracil e adenina
            objAtomos.getAtomoEle(molS[i][j],r);
            objAtomos.getAtomoEle2(molS[i][j],r);
            //para guanina e citosina
            //objAtomos.getAtomoEleCG(molS[i][j],r);
            //objAtomos.getAtomoEleCG2(molS[i][j],r);
          }
        }
      }
    }

  }//fim completarM
  public double[] angdU (int semente){
```

```
double[] pangd= new double[torS.length];
double pangdebb3,pangdeb3,pangdbbe,pangdbe,pangdd,pangdb3,
    pangdbb3,pangdl,pangdll, pangdc,pangdcc, pangdb,
   pangdbb, pangdc1,pangdcc1, pangdb1, pangdbb1,pangdc2,
   pangdcc2,pangdbb2,pangdb2;
double angdebb2,angdebb3,angdeb3,angdbe,angdbbe,angdb3,
   angdbb3,angdl,angdll,angdc,angdcc,angdb,angdbb,angdb1,
   angdbb1,angdb2,angdbb2, angdc1,angdcc1,angdcc2,angdc2;
double auxdebb2,auxdeb3, auxdebb3,auxdbe,auxdbbe,auxdb3,
   auxdbb3,auxdl,auxdll,auxdb,auxdbb,auxdc,auxdcc,auxdb1,
   auxdbb1,auxdb2,auxdbb2,auxdc1,auxdcc1,auxdcc2,auxdc2;
double auxeb3, auxebb3,auxbe,auxbbe,auxb3,auxbb3,auxl,
   auxll,auxb,auxbb,auxc,auxcc,auxb1,auxbb1,auxb2,auxbb2,
   auxc1,auxcc1,auxcc2,auxc2;
int indiceD= 0;
double a1, a2, b1, b2, c1, c2, v1, x1, z1, v2, x2, z2, v3,
    x3, z3, v4, x4, z4, x1s, y1s, z1s;//, x2s, y2s, z2s;
double aux = 0.0;
double[] aux1ebb2, aux2ebb2,aux3ebb2,aux4ebb2,aux1eb3,
   aux2eb3,aux3eb3,aux4eb3,aux1ebb3,aux2ebb3,aux3ebb3,
   aux4ebb3,aux1be, aux2be,aux3be,aux4be, aux1bbe,aux2bbe,
   aux3bbe,aux4bbe,aux1l, aux2l, aux3l, aux4l, aux1ll,
   aux2ll, aux3ll, aux4ll,
aux1b, aux2b, aux3b, aux4b, aux1bb, aux2bb, aux3bb, aux4bb
   , aux1cc, aux2cc, aux3cc, aux4cc, aux1c, aux2c, aux3c,
    aux4c, aux1c1, aux2c1, aux3c1, aux4c1, aux1cc1,
   aux2cc1, aux3cc1, aux4cc1, aux1b1, aux2b1, aux3b1,
   aux4b1,aux1bb1, aux2bb1, aux3bb1, aux4bb1, aux1c2,
   aux2c2, aux3c2, aux4c2, aux1cc2, aux2cc2, aux3cc2,
   aux4cc2, aux1b3, aux2b3, aux3b3, aux4b3, aux1bb3,
   aux2bb3, aux3bb3, aux4bb3, aux1bb2, aux2bb2, aux4bb2,
   aux3bb2;
double pmod, mod, mod2, prod;
int ind= 0;
//variaveis para ver se nao existe mais de uma forma para
   ser feito o calculo
```

```
num=1;
indice=0;
String log="";
double energia=0.0;
sem= semente;

for(int r= 0; r< torS.length; r++){
  aux1eb3= new double[3];
  aux2eb3= new double[3];
  aux3eb3= new double[3];
  aux4eb3= new double[3];
  aux1ebb3= new double[3];
  aux2ebb3= new double[3];
  aux3ebb3= new double[3];
  aux4ebb3= new double[3];
  aux1be= new double[3];
  aux2be= new double[3];
  aux3be= new double[3];
  aux4be= new double[3];
  aux1bbe= new double[3];
  aux2bbe= new double[3];
  aux3bbe= new double[3];
  aux4bbe= new double[3];
  aux1l= new double[3];
  aux2l= new double[3];
  aux3l= new double[3];
  aux4l= new double[3];
  aux1ll= new double[3];
  aux2ll= new double[3];
  aux3ll= new double[3];
  aux4ll= new double[3];
  aux1b= new double[3];
  aux2b= new double[3];
  aux3b= new double[3];
  aux4b= new double[3];
  aux1bb= new double[3];
```

```
aux2bb= new double[3];
aux3bb= new double[3];
aux4bb= new double[3];
aux1c= new double[3];
aux2c= new double[3];
aux3c= new double[3];
aux4c= new double[3];
aux1cc= new double[3];
aux2cc= new double[3];
aux3cc= new double[3];
aux4cc= new double[3];
aux1c1= new double[3];
aux2c1= new double[3];
aux3c1= new double[3];
aux4c1= new double[3];
aux1cc1= new double[3];
aux2cc1= new double[3];
aux3cc1= new double[3];
aux4cc1= new double[3];
aux1b1= new double[3];
aux2b1= new double[3];
aux3b1= new double[3];
aux4b1= new double[3];
aux1bb1= new double[3];
aux2bb1= new double[3];
aux3bb1= new double[3];
aux4bb1= new double[3];
aux1b3= new double[3];
aux2b3= new double[3];
aux3b3= new double[3];
aux4b3= new double[3];
aux1bb3= new double[3];
aux2bb3= new double[3];
aux3bb3= new double[3];
aux4bb3= new double[3];
aux1c2= new double[3];
```

```
aux2c2= new double[3];
aux3c2= new double[3];
aux4c2= new double[3];
aux1cc2= new double[3];
aux2cc2= new double[3];
aux3cc2= new double[3];
aux4cc2= new double[3];
aux1bb2= new double[3];
aux2bb2= new double[3];
aux3bb2= new double[3];
aux4bb2= new double[3];
x1s=0.0;
y1s=0.0;
z1s=0.0;
v1= 0.0;
v2= 0.0;
x1= 0.0;
x2= 0.0;
z1= 0.0;
z2= 0.0;
v3= 0.0;
v4= 0.0;
x3= 0.0;
x4= 0.0;
z3= 0.0;
z4= 0.0;
mod= 0.0;
mod2= 0.0;
pmod = 0.0;
prod = 0.0;
pangdebb3=0.0;
pangdeb3=0.0;
pangdbbe=0.0;
pangdbe= 0.0;
pangdd=0.0;
pangdb3=0.0;
```

```
pangdbb3=0.0;
pangdbb2=0.0;
pangdl=0.0;
pangdll=0.0;
pangdc=0.0;
pangdcc=0.0;
pangdb=0.0;
pangdbb=0.0;
pangdc1=0.0;
pangdcc1=0.0;
pangdb1=0.0;
pangdbb1=0.0;
pangdc2=0.0;
pangdcc2=0.0;
auxdeb3=0.0;
auxdebb3=0.0;
auxdebb2=0.0;
auxdbe=0.0;
auxdbbe=0.0;
auxeb3=0.0;
auxebb3=0.0;
auxdebb2=0.0;
auxbe=0.0;
auxbbe=0.0;
angdebb3=0.0;
angdeb3=0.0;
pangdb2= 0.0;
angdl=0.0;
angdll=0.0;
angdb3=0.0;
angdbb3=0.0;
angdbe= 0;
angdbbe= 0;
angdb=0.0;
angdbb=0.0;
angdc=0.0;
```

```
angdcc=0.0;
angdc1=0.0;
angdcc1=0.0;
angdc2=0.0;
angdcc2=0.0;
angdb1=0.0;
angdbb1=0.0;
angdb2=0.0;
auxdl=0.0;
auxdll=0.0;
auxdb3=0.0;
auxdbb3=0.0;
auxdb=0.0;
auxdbb=0.0;
auxdc=0.0;
auxdcc=0.0;
auxdc1=0.0;
auxdcc1=0.0;
auxdc2=0.0;
auxdcc2=0.0;
auxdb1=0.0;
auxdbb1=0.0;
auxdb2=0.0;
auxl=0.0;
auxll=0.0;
auxb3=0.0;
auxbb3=0.0;
auxb=0.0;
auxbb=0.0;
auxc=0.0;
auxcc=0.0;
auxc1=0.0;
auxcc1=0.0;
auxc2=0.0;
auxcc2=0.0;
auxb1=0.0;
```

```
    auxbb1=0.0;
    auxb2=0.0;
    auxbb2=0.0;
    a1=0.0;
    a2=0.0;
    b1=0.0;
    b2=0.0;
    c1=0.0;
  for (int i= 0; i<qntdlinha; i++){
  for (int j= 0; j< qntdcol; j++){
    aux=0.0;
    if (molS[i][j].equals("X00000001000")){
      //nao quero que comece com X
    }else{
      if((i<qntdlinha-1)&&(molS[i][j]!="") && (j<qntdcol-2)
         &&(molS[i][j+1]!="") && (molS[i][j+2]!="") && (molS
         [i+1][j+2]!="")){
        indice=0;
        //primeira parte, posicao(7,8), indica ligacoes
           horizontais
        if ((molS[i][j].substring(8, 9).equals("1"))&&(molS[
           i][j+1].substring(8, 9).equals("1"))&&(molS[i][j
           +1].substring(8, 9).equals("1"))&&(molS[i][j+2].
           substring(8, 9).equals("1"))&&
            (molS[i][j+2].substring(9, 10).equals("1"))&&(
               molS[i+1][j+2].substring(9, 10).equals("1")))
               {
          indice= 0;
          //isolar a b e c dos planos
          aux1l= objAtomos.setAtomoU(molS[i][j],torS[r]);
          aux2l= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
          aux3l= objAtomos.setAtomoU(molS[i][j+2],torS[0]);
          aux4l= objAtomos.setAtomoU(molS[i+1][j+2],torS[r])
             ;
          v1= objAtomos.getX(aux1l,torS[r])-objAtomos.getX(
             aux2l,torS[r]);
```

```
x1= objAtomos.getY(aux1l,torS[r])-objAtomos.getY(
    aux2l,torS[r]);
z1= objAtomos.getZ(aux1l,torS[r])-objAtomos.getZ(
    aux2l,torS[r]);
v2= objAtomos.getX(aux3l,torS[r])-objAtomos.getX(
    aux2l,torS[r]);
x2= objAtomos.getY(aux3l,torS[r])-objAtomos.getY(
    aux2l,torS[r]);
z2= objAtomos.getZ(aux3l,torS[r])-objAtomos.getZ(
    aux2l,torS[r]);
v3= objAtomos.getX(aux2l,torS[r])-objAtomos.getX(
    aux3l,torS[r]);
x3= objAtomos.getY(aux2l,torS[r])-objAtomos.getY(
    aux3l,torS[r]);
z3= objAtomos.getZ(aux2l,torS[r])-objAtomos.getZ(
    aux3l,torS[r]);
v4= objAtomos.getX(aux4l,torS[r])-objAtomos.getX(
    aux3l,torS[r]);
x4= objAtomos.getY(aux4l,torS[r])-objAtomos.getY(
    aux3l,torS[r]);
z4= objAtomos.getZ(aux4l,torS[r])-objAtomos.getZ(
    aux3l,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdl= (prod)/(pmod);
angdl= Math.acos(auxdl);
```

```
        double sinal= (v1*a2)+(x1*b2)+(z1*c2);
        if (sinal<=0.0){
          angdl= -angdl;
        }
        pangdl= (getVnU(molS[i][j],molS[i][j+1],molS[i][j
            +2],molS[i+1][j+2],sem,num,indice)/getPath(molS
            [i][j],molS[i][j+1],molS[i][j+2],molS[i+1][j
            +2],sem,num,indice)*(1+(Math.cos((getN(molS[i][
            j],molS[i][j+1],molS[i][j+2],molS[i+1][j+2],sem
            ,indice)*angdl)-getPeriU(molS[i][j],molS[i][j
            +1],molS[i][j+2],molS[i+1][j+2],sem,indice)))))
            ;
        //arredondamento 4 casas decimais TINKER
        pangdl= (Math.round((pangdl * 10000.0))) /
            10000.0;
        pangdl= pangdl+auxl;
        auxl= pangdl;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }//fim if
      while (num==0){
        //transforma pra graus o cos
        pangdl= (getVnU(molS[i][j],molS[i][j+1],molS[i][j
            +2],molS[i+1][j+2],sem,num,indice)/getPath(molS
            [i][j],molS[i][j+1],molS[i][j+2],molS[i+1][j
            +2],sem,num,indice)*(1+(Math.cos((getN(molS[i][
            j],molS[i][j+1],molS[i][j+2],molS[i+1][j+2],sem
            ,indice)*angdll)-getPeriU(molS[i][j],molS[i][j
            +1],molS[i][j+2],molS[i+1][j+2],sem,indice)))))
            ;
        //arredondamento TINKER
        pangdl= (Math.round((pangdl * 10000.0))) /
            10000.0;
        pangdl= pangdl+auxl;
        auxl= pangdl;
```

```
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim if
        if ((j<qntdcol-5)&&(molS[i][j]!="") && (molS[i][j
            +3]!="") && (molS[i][j+4]!="") && (molS[i][j
            +5]!="")&&((molS[i][j+1]=="X00000001000")&&(molS[i
            ][j+2]=="X00000001000"))){
          indice=0;
          int x=j+3;
          if ((molS[i][j].substring(8, 9).equals("1"))&&(molS[
              i][x].substring(8, 9).equals("1"))&&(molS[i][x].
              substring(8, 9).equals("1"))&&(molS[i][x+1].
              substring(8, 9).equals("1"))&&
             (molS[i][x+1].substring(8, 9).equals("1"))&&(molS[
                 i][x+2].substring(8, 9).equals("1"))){
            aux1ll= objAtomos.setAtomoU(molS[i][j],torS[r]);
            aux2ll= objAtomos.setAtomoU(molS[i][x],torS[r]);
            aux3ll= objAtomos.setAtomoU(molS[i][x+1],torS[r]);
            aux4ll= objAtomos.setAtomoU(molS[i][x+2],torS[r]);
            v1= objAtomos.getX(aux1ll,torS[r])-objAtomos.getX(
                aux2ll,torS[r]);
            x1= objAtomos.getY(aux1ll,torS[r])-objAtomos.getY(
                aux2ll,torS[r]);
            z1= objAtomos.getZ(aux1ll,torS[r])-objAtomos.getZ(
                aux2ll,torS[r]);
            v2= objAtomos.getX(aux3ll,torS[r])-objAtomos.getX(
                aux2ll,torS[r]);
            x2= objAtomos.getY(aux3ll,torS[r])-objAtomos.getY(
                aux2ll,torS[r]);
            z2= objAtomos.getZ(aux3ll,torS[r])-objAtomos.getZ(
                aux2ll,torS[r]);
            v3= objAtomos.getX(aux2ll,torS[r])-objAtomos.getX(
                aux3ll,torS[r]);
```

```
x3= objAtomos.getY(aux2ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z3= objAtomos.getZ(aux2ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
v4= objAtomos.getX(aux4ll,torS[r])-objAtomos.getX(
    aux3ll,torS[r]);
x4= objAtomos.getY(aux4ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z4= objAtomos.getZ(aux4ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdll= (prod)/(pmod);
angdll= Math.acos(auxdll);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdll= -angdll;
}
pangdll= (getVnU(molS[i][j],molS[i][x],molS[i][x
    +1],molS[i][x+2],sem,num,indice)/getPath(molS[i
    ][j],molS[i][x],molS[i][x+1],molS[i][x+2],sem,
    num,indice)*(1+(Math.cos((getN(molS[i][j],molS[
    i][x],molS[i][x+1],molS[i][x+2],sem,indice)*
    angdll)-getPeriU(molS[i][j],molS[i][x],molS[i][
    x+1],molS[i][x+2],sem,indice)))));
//arredondamento TINKER
```

```
            pangdll= (Math.round((pangdll * 10000.0))) /
               10000.0;
            pangdll= pangdll+auxll;
            auxll= pangdll;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdll= (getVnU(molS[i][j],molS[i][x],molS[i][x
                  +1],molS[i][x+2],sem,num,indice)/getPath(molS
                  [i][j],molS[i][x],molS[i][x+1],molS[i][x+2],
                  sem,num,indice)*(1+(Math.cos((getN(molS[i][j
                  ],molS[i][x],molS[i][x+1],molS[i][x+2],sem,
                  indice)*angdll)-getPeriU(molS[i][j],molS[i][x
                  ],molS[i][x+1],molS[i][x+2],sem,indice)))));
              //arredondamento TINKER
              pangdll= (Math.round((pangdll * 10000.0))) /
                  10000.0;
              pangdll= pangdll+auxll;
              auxll= pangdll;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }else if ((j<qntdcol-5)&&(molS[i][j]!="") && (molS[i][
            j+1]!="") && (molS[i][j+4]!="") && (molS[i][j
            +5]!="")&&((molS[i][j+2]=="X00000001000")&&(molS[i
            ][j+3]=="X00000001000"))){
          indice=0;
          int x=j+4;
          if ((molS[i][j].substring(8, 9).equals("1"))&&(molS[
              i][j+1].substring(8, 9).equals("1"))&&(molS[i][j
              +1].substring(8, 9).equals("1"))&&(molS[i][x].
              substring(8, 9).equals("1"))&&
```

```
(molS[i][x].substring(8, 9).equals("1"))&&(molS[i
    ][x+1].substring(8, 9).equals("1"))){
aux1ll= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2ll= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
aux3ll= objAtomos.setAtomoU(molS[i][x],torS[r]);
aux4ll= objAtomos.setAtomoU(molS[i][x+1],torS[r]);
v1= objAtomos.getX(aux1ll,torS[r])-objAtomos.getX(
    aux2ll,torS[r]);
x1= objAtomos.getY(aux1ll,torS[r])-objAtomos.getY(
    aux2ll,torS[r]);
z1= objAtomos.getZ(aux1ll,torS[r])-objAtomos.getZ(
    aux2ll,torS[r]);
v2= objAtomos.getX(aux3ll,torS[r])-objAtomos.getX(
    aux2ll,torS[r]);
x2= objAtomos.getY(aux3ll,torS[r])-objAtomos.getY(
    aux2ll,torS[r]);
z2= objAtomos.getZ(aux3ll,torS[r])-objAtomos.getZ(
    aux2ll,torS[r]);
v3= objAtomos.getX(aux2ll,torS[r])-objAtomos.getX(
    aux3ll,torS[r]);
x3= objAtomos.getY(aux2ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z3= objAtomos.getZ(aux2ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
v4= objAtomos.getX(aux4ll,torS[r])-objAtomos.getX(
    aux3ll,torS[r]);
x4= objAtomos.getY(aux4ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z4= objAtomos.getZ(aux4ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
```

```
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdll= (prod)/(pmod);
angdll= Math.acos(auxdll);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdll= -angdll;
}
pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[i][x
    ],molS[i][x+1],sem,num,indice)/getPath(molS[i][
    j],molS[i][j+1],molS[i][x],molS[i][x+1],sem,num
    ,indice)*(1+(Math.cos((getN(molS[i][j],molS[i][
    j+1],molS[i][x],molS[i][x+1],sem,indice)*angdll
    )-getPeriU(molS[i][j],molS[i][j+1],molS[i][x],
    molS[i][x+1],sem,indice)))));
//arredondamento TINKER
pangdll= (Math.round((pangdll * 10000.0))) /
    10000.0;
pangdll= pangdll+auxll;
auxll= pangdll;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[i
      ][x],molS[i][x+1],sem,num,indice)/getPath(
      molS[i][j],molS[i][j+1],molS[i][x],molS[i][x
      +1],sem,num,indice)*(1+(Math.cos((getN(molS[i
      ][j],molS[i][j+1],molS[i][x],molS[i][x+1],sem
      ,indice)*angdll)-getPeriU(molS[i][j],molS[i][
      j+1],molS[i][x],molS[i][x+1],sem,indice)))));
```

```
            //arredondamento TINKER
            pangdll= (Math.round((pangdll * 10000.0))) /
                10000.0;
            pangdll= pangdll+auxll;
            auxll= pangdll;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }else if ((j<qntdcol-5)&&(molS[i][j]!="") && (molS[i][
          j+1]!="") && (molS[i][j+2]!="") && (molS[i][j
          +5]!="")&&
        ((molS[i][j+3]=="X00000001000")&&(molS[i][j+4]=="
            X00000001000"))){
        indice=0;
        int x=j+5;
        //primeira parte, posicao(8,9), indica ligacoes a
            direita
        if ((molS[i][j].substring(8, 9).equals("1"))&&(molS[
            i][j+1].substring(8, 9).equals("1"))&&(molS[i][j
            +1].substring(8, 9).equals("1"))&&(molS[i][j+2].
            substring(8, 9).equals("1"))&&
          (molS[i][j+2].substring(8, 9).equals("1"))&&(molS[
              i][x].substring(8, 9).equals("1"))){
          aux1ll= objAtomos.setAtomoU(molS[i][j],torS[r]);
          aux2ll= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
          aux3ll= objAtomos.setAtomoU(molS[i][j+2],torS[r]);
          aux4ll= objAtomos.setAtomoU(molS[i][x],torS[r]);
          v1= objAtomos.getX(aux1ll,torS[r])-objAtomos.getX(
              aux2ll,torS[r]);
          x1= objAtomos.getY(aux1ll,torS[r])-objAtomos.getY(
              aux2ll,torS[r]);
          z1= objAtomos.getZ(aux1ll,torS[r])-objAtomos.getZ(
              aux2ll,torS[r]);
```

```
v2= objAtomos.getX(aux3ll,torS[r])-objAtomos.getX(
    aux2ll,torS[r]);
x2= objAtomos.getY(aux3ll,torS[r])-objAtomos.getY(
    aux2ll,torS[r]);
z2= objAtomos.getZ(aux3ll,torS[r])-objAtomos.getZ(
    aux2ll,torS[r]);
v3= objAtomos.getX(aux2ll,torS[r])-objAtomos.getX(
    aux3ll,torS[r]);
x3= objAtomos.getY(aux2ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z3= objAtomos.getZ(aux2ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
v4= objAtomos.getX(aux4ll,torS[r])-objAtomos.getX(
    aux3ll,torS[r]);
x4= objAtomos.getY(aux4ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z4= objAtomos.getZ(aux4ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdll= (prod)/(pmod);
angdll= Math.acos(auxdll);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdll= -angdll;
}
```

```
        pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[i][j
            +2],molS[i][x],sem,num,indice)/getPath(molS[i][
            j],molS[i][j+1],molS[i][j+2],molS[i][x],sem,num
            ,indice)*(1+(Math.cos((getN(molS[i][j],molS[i][
            j+1],molS[i][j+2],molS[i][x],sem,indice)*angdll
            )-getPeriU(molS[i][j],molS[i][j+1],molS[i][j
            +2],molS[i][x],sem,indice)))));
        //arredondamento TINKER
        pangdll= (Math.round((pangdll * 10000.0))) /
            10000.0;
        pangdll= pangdll+auxll;
        auxll= pangdll;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[i
              ][j+2],molS[i][x],sem,num,indice)/getPath(
              molS[i][j],molS[i][j+1],molS[i][j+2],molS[i][
              x],sem,num,indice)*(1+(Math.cos((getN(molS[i
              ][j],molS[i][j+1],molS[i][j+2],molS[i][x],sem
              ,indice)*angdll)-getPeriU(molS[i][j],molS[i][
              j+1],molS[i][j+2],molS[i][x],sem,indice)))));
          //arredondamento TINKER
          pangdll= (Math.round((pangdll * 10000.0))) /
              10000.0;
          pangdll= pangdll+auxll;
          auxll= pangdll;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }
    }else if ((j<qntdcol-3)&&(molS[i][j]!="")&&(molS[i][j
      +1]!="")&&(molS[i][j+2]!="")&&(molS[i][j+3]!="")){
```

```
if ((molS[i][j].substring(8, 9).equals("1"))&&(molS[
    i][j+1].substring(8, 9).equals("1"))&&(molS[i][j
    +1].substring(8, 9).equals("1"))&&(molS[i][j+2].
    substring(8, 9).equals("1"))&&
   (molS[i][j+2].substring(8, 9).equals("1"))&&(molS[
       i][j+3].substring(8, 9).equals("1"))&&((molS[i
       ][j+1]!="X00000001000")&&(molS[i][j+2]!="
       X00000001000")&&(molS[i][j+3]!="X00000001000"))
       ){
   indice=0;
   if ((molS[i][j+1].substring(0,7).equals("OHR1001")
       )&&(molS[i][j+2].substring(0,7).equals("OHR1002
       "))){

   }else{
     //isolar a b e c dos planos
     aux1ll= objAtomos.setAtomoU(molS[i][j],torS[r]);
     aux2ll= objAtomos.setAtomoU(molS[i][j+1],torS[r
         ]);
     aux3ll= objAtomos.setAtomoU(molS[i][j+2],torS[r
         ]);
     aux4ll= objAtomos.setAtomoU(molS[i][j+3],torS[r
         ]);
     v1= objAtomos.getX(aux1ll,torS[r])-objAtomos.
         getX(aux2ll,torS[r]);
     x1= objAtomos.getY(aux1ll,torS[r])-objAtomos.
         getY(aux2ll,torS[r]);
     z1= objAtomos.getZ(aux1ll,torS[r])-objAtomos.
         getZ(aux2ll,torS[r]);
     v2= objAtomos.getX(aux3ll,torS[r])-objAtomos.
         getX(aux2ll,torS[r]);
     x2= objAtomos.getY(aux3ll,torS[r])-objAtomos.
         getY(aux2ll,torS[r]);
     z2= objAtomos.getZ(aux3ll,torS[r])-objAtomos.
         getZ(aux2ll,torS[r]);
```

```
v3= objAtomos.getX(aux2ll,torS[r])-objAtomos.
    getX(aux3ll,torS[r]);
x3= objAtomos.getY(aux2ll,torS[r])-objAtomos.
    getY(aux3ll,torS[r]);
z3= objAtomos.getZ(aux2ll,torS[r])-objAtomos.
    getZ(aux3ll,torS[r]);
v4= objAtomos.getX(aux4ll,torS[r])-objAtomos.
    getX(aux3ll,torS[r]);
x4= objAtomos.getY(aux4ll,torS[r])-objAtomos.
    getY(aux3ll,torS[r]);
z4= objAtomos.getZ(aux4ll,torS[r])-objAtomos.
    getZ(aux3ll,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdll= (prod)/(pmod);
angdll= Math.acos(auxdll);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdll= -angdll;
}
pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[i
    ][j+2],molS[i][j+3],sem,num,indice)/getPath(
    molS[i][j],molS[i][j+1],molS[i][j+2],molS[i][
    j+3],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i][j+1],molS[i][j+2],molS[i][j+3],
    sem,indice)*angdll)-getPeriU(molS[i][j],molS[
```

```
                i][j+1],molS[i][j+2],molS[i][j+3],sem,indice)
                ))));
            //arredondamento TINKER
            pangdll= (Math.round((pangdll * 10000.0))) /
                10000.0;
            pangdll= pangdll+auxll;
            auxll= pangdll;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[
                  i][j+2],molS[i][j+3],sem,num,indice)/
                  getPath(molS[i][j],molS[i][j+1],molS[i][j
                  +2],molS[i][j+3],sem,num,indice)*(1+(Math.
                  cos((getN(molS[i][j],molS[i][j+1],molS[i][j
                  +2],molS[i][j+3],sem,indice)*angdll)-
                  getPeriU(molS[i][j],molS[i][j+1],molS[i][j
                  +2],molS[i][j+3],sem,indice)))));
              //arredondamento TINKER
              pangdll= (Math.round((pangdll * 10000.0))) /
                  10000.0;
              pangdll= pangdll+auxll;
              auxll= pangdll;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }//fim if
      }//fim if
      if((((i<qntdlinha-1)&&(j<qntdcol-3)&&(molS[i][j]!="")
          && (molS[i+1][j+2]!="") && (molS[i+1][j+3]!="") &&
          (molS[i][j+2]!=""))))){
```

```
if ((molS[i+1][j+1]=="X00000001000")&&(molS[i+1][j
    +2]=="X00000001000")){
  int x=j+3;
  if((x>=1)&&(molS[i][j]!="") && (molS[i+1][j]!="")
     && (molS[i+1][x]!="") && (molS[i][x-1]!="")){
    if ((((molS[i][j].substring(9, 10).equals("1"))
       &&(molS[i+1][j].substring(9, 10).equals("1"))
       ))&&(molS[i+1][j].substring(8, 9).equals("1")
       )&&(molS[i+1][x].substring(8, 9).equals("1"))
       &&(molS[i+1][x].substring(11, 12).equals("1")
       )&(molS[i][x-1].substring(10, 11).equals("1")
       )){
      indice=0;
      aux1eb3= objAtomos.setAtomoU(molS[i][j],torS[r
         ]);
      aux2eb3= objAtomos.setAtomoU(molS[i+1][j],torS
         [r]);
      aux3eb3= objAtomos.setAtomoU(molS[i+1][x],torS
         [r]);
      aux4eb3= objAtomos.setAtomoU(molS[i][x-1],torS
         [r]);
      v1= objAtomos.getX(aux1eb3,torS[r])-objAtomos.
         getX(aux2eb3,torS[r]);
      x1= objAtomos.getY(aux1eb3,torS[r])-objAtomos.
         getY(aux2eb3,torS[r]);
      z1= objAtomos.getZ(aux1eb3,torS[r])-objAtomos.
         getZ(aux2eb3,torS[r]);
      v2= objAtomos.getX(aux3eb3,torS[r])-objAtomos.
         getX(aux2eb3,torS[r]);
      x2= objAtomos.getY(aux3eb3,torS[r])-objAtomos.
         getY(aux2eb3,torS[r]);
      z2= objAtomos.getZ(aux3eb3,torS[r])-objAtomos.
         getZ(aux2eb3,torS[r]);
      v3= objAtomos.getX(aux2eb3,torS[r])-objAtomos.
         getX(aux3eb3,torS[r]);
```

```
x3= objAtomos.getY(aux2eb3,torS[r])-objAtomos.
    getY(aux3eb3,torS[r]);
z3= objAtomos.getZ(aux2eb3,torS[r])-objAtomos.
    getZ(aux3eb3,torS[r]);
v4= objAtomos.getX(aux4eb3,torS[r])-objAtomos.
    getX(aux3eb3,torS[r]);
x4= objAtomos.getY(aux4eb3,torS[r])-objAtomos.
    getY(aux3eb3,torS[r]);
z4= objAtomos.getZ(aux4eb3,torS[r])-objAtomos.
    getZ(aux3eb3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdeb3= (prod)/(pmod);
angdeb3= Math.acos(auxdeb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdeb3= -angdeb3;
}
//transforma pra graus o cos
pangdeb3= (getVnU(molS[i][j],molS[i+1][j],molS
    [i+1][x],molS[i][x-1],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j],molS[i+1][x
    ],molS[i][x-1],sem,num,indice)*(1+(Math.cos
    ((getN(molS[i][j],molS[i+1][j],molS[i+1][x
    ],molS[i][x-1],sem,indice)*angdeb3)-
    getPeriU(molS[i][j],molS[i+1][j],molS[i+1][
```

```
                x],molS[i][x-1],sem,indice)))));
            //arredondamento TINKER
            pangdeb3= (Math.round((pangdeb3 * 10000.0))) /
                10000.0;
            pangdeb3= pangdeb3+auxeb3;
            auxeb3= pangdeb3;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdeb3= (getVnU(molS[i][j],molS[i+1][j],
                  molS[i+1][x],molS[i][x-1],sem,num,indice)
                  /getPath(molS[i][j],molS[i+1][j],molS[i
                  +1][x],molS[i][x-1],sem,num,indice)*(1+(
                  Math.cos((getN(molS[i][j],molS[i+1][j],
                  molS[i+1][x],molS[i][x-1],sem,indice)*
                  angdeb3)-getPeriU(molS[i][j],molS[i+1][j
                  ],molS[i+1][x],molS[i][x-1],sem,indice)))
                  ));
              //arredondamento TINKER
              pangdeb3= (Math.round((pangdeb3 * 10000.0)))
                  / 10000.0;
              pangdeb3= pangdeb3+auxeb3;
              auxeb3= pangdeb3;
              indiceD= indiceD + 1;
              ind++;
              sem++;
             }
          }
        }
      }
    }
    if((((i<qntdlinha-1)&&(j<qntdcol-4)&&(molS[i][j]!="")
       && (molS[i+1][j]!="") && (molS[i+1][j+3]!="") && (
       molS[i][j+4]!=""))))&&((molS[i+1][j+1]=="
```

```
 X00000001000")&&(molS[i+1][j+2]=="X00000001000"))){
int x=j+3;
if((molS[i][j]!="") && (molS[i+1][j]!="") && (molS[i
   +1][x]!="") && (molS[i][x+1]!="")){
  if ((((molS[i][j].substring(9, 10).equals("1"))&&(
     molS[i+1][j].substring(9, 10).equals("1"))))&&(
     molS[i+1][j].substring(8, 9).equals("1"))&&(
     molS[i+1][x].substring(8, 9).equals("1"))&&(
     molS[i+1][x].substring(10, 11).equals("1"))&(
     molS[i][x+1].substring(11, 12).equals("1"))){
    indice=0;
    aux1ebb3= objAtomos.setAtomoU(molS[i][j],torS[r
       ]);
    aux2ebb3= objAtomos.setAtomoU(molS[i+1][j],torS[
       r]);
    aux3ebb3= objAtomos.setAtomoU(molS[i+1][x],torS[
       r]);
    aux4ebb3= objAtomos.setAtomoU(molS[i][x+1],torS[
       r]);
    v1= objAtomos.getX(aux1ebb3,torS[r])-objAtomos.
       getX(aux2ebb3,torS[r]);
    x1= objAtomos.getY(aux1ebb3,torS[r])-objAtomos.
       getY(aux2ebb3,torS[r]);
    z1= objAtomos.getZ(aux1ebb3,torS[r])-objAtomos.
       getZ(aux2ebb3,torS[r]);
    v2= objAtomos.getX(aux3ebb3,torS[r])-objAtomos.
       getX(aux2ebb3,torS[r]);
    x2= objAtomos.getY(aux3ebb3,torS[r])-objAtomos.
       getY(aux2ebb3,torS[r]);
    z2= objAtomos.getZ(aux3ebb3,torS[r])-objAtomos.
       getZ(aux2ebb3,torS[r]);
    v3= objAtomos.getX(aux2ebb3,torS[r])-objAtomos.
       getX(aux3ebb3,torS[r]);
    x3= objAtomos.getY(aux2ebb3,torS[r])-objAtomos.
       getY(aux3ebb3,torS[r]);
```

```
z3= objAtomos.getZ(aux2ebb3,torS[r])-objAtomos.
    getZ(aux3ebb3,torS[r]);
v4= objAtomos.getX(aux4ebb3,torS[r])-objAtomos.
    getX(aux3ebb3,torS[r]);
x4= objAtomos.getY(aux4ebb3,torS[r])-objAtomos.
    getY(aux3ebb3,torS[r]);
z4= objAtomos.getZ(aux4ebb3,torS[r])-objAtomos.
    getZ(aux3ebb3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdebb3= (prod)/(pmod);
angdebb3= Math.acos(auxdebb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdebb3= -angdebb3;
}
//transforma pra graus o cos
pangdebb3= (getVnU(molS[i][j],molS[i+1][j],molS[
    i+1][x],molS[i][x+1],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j],molS[i+1][x],molS[i][
    x+1],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i+1][j],molS[i+1][x],molS[i][x+1],
    sem,indice)*angdebb3)-getPeriU(molS[i][j],
    molS[i+1][j],molS[i+1][x],molS[i][x+1],sem,
    indice)))));
//arredondamento TINKER
```

```
            pangdebb3= (Math.round((pangdebb3 * 10000.0))) /
                10000.0;
            pangdebb3= pangdebb3+auxebb3;
            auxebb3= pangdebb3;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdebb3= (getVnU(molS[i][j],molS[i+1][j],
                  molS[i+1][x],molS[i][x+1],sem,num,indice)/
                  getPath(molS[i][j],molS[i+1][j],molS[i+1][x
                  ],molS[i][x+1],sem,num,indice)*(1+(Math.cos
                  ((getN(molS[i][j],molS[i+1][j],molS[i+1][x
                  ],molS[i][x+1],sem,indice)*angdebb3)-
                  getPeriU(molS[i][j],molS[i+1][j],molS[i+1][
                  x],molS[i][x+1],sem,indice)))));
              //arredondamento TINKER
              pangdebb3= (Math.round((pangdebb3 * 10000.0)))
                  / 10000.0;
              pangdebb3= pangdebb3+auxebb3;
              auxebb3= pangdebb3;
              indiceD= indiceD + 1;
              ind++;
              sem++;
             }
          }
        }
      }else if ((i<qntdlinha-1)&&(j<qntdcol-2)&&(molS[i][j
          ]!="") && (molS[i+1][j]!="") && (molS[i+1][j
          +1]!="") && (molS[i][j+2]!="")&&(((molS[i][j].
          substring(9, 10).equals("1"))&&(molS[i+1][j].
          substring(9, 10).equals("1"))))&&(molS[i+1][j].
          substring(8, 9).equals("1"))&&(molS[i+1][j+1].
          substring(8, 9).equals("1"))&&(molS[i+1][j+1].
          substring(10, 11).equals("1"))&(molS[i][j+2].
```

```
substring(11, 12).equals("1"))&& (molS[i+1][j+1]!="
X00000001000")){
 indice=0;
 aux1ebb3= objAtomos.setAtomoU(molS[i][j],torS[r]);
 aux2ebb3= objAtomos.setAtomoU(molS[i+1][j],torS[r
    ]);
 aux3ebb3= objAtomos.setAtomoU(molS[i+1][j+1],torS[
    r]);
 aux4ebb3= objAtomos.setAtomoU(molS[i][j+2],torS[r
    ]);
 v1= objAtomos.getX(aux1ebb3,torS[r])-objAtomos.
    getX(aux2ebb3,torS[r]);
 x1= objAtomos.getY(aux1ebb3,torS[r])-objAtomos.
    getY(aux2ebb3,torS[r]);
 z1= objAtomos.getZ(aux1ebb3,torS[r])-objAtomos.
    getZ(aux2ebb3,torS[r]);
 v2= objAtomos.getX(aux3ebb3,torS[r])-objAtomos.
    getX(aux2ebb3,torS[r]);
 x2= objAtomos.getY(aux3ebb3,torS[r])-objAtomos.
    getY(aux2ebb3,torS[r]);
 z2= objAtomos.getZ(aux3ebb3,torS[r])-objAtomos.
    getZ(aux2ebb3,torS[r]);
 v3= objAtomos.getX(aux2ebb3,torS[r])-objAtomos.
    getX(aux3ebb3,torS[r]);
 x3= objAtomos.getY(aux2ebb3,torS[r])-objAtomos.
    getY(aux3ebb3,torS[r]);
 z3= objAtomos.getZ(aux2ebb3,torS[r])-objAtomos.
    getZ(aux3ebb3,torS[r]);
 v4= objAtomos.getX(aux4ebb3,torS[r])-objAtomos.
    getX(aux3ebb3,torS[r]);
 x4= objAtomos.getY(aux4ebb3,torS[r])-objAtomos.
    getY(aux3ebb3,torS[r]);
 z4= objAtomos.getZ(aux4ebb3,torS[r])-objAtomos.
    getZ(aux3ebb3,torS[r]);
 a1= ((x1*z2)-(x2*z1));
 b1= ((z1*v2)-(v1*z2));
```

```
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdebb3= (prod)/(pmod);
angdebb3= Math.acos(auxdebb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdebb3= -angdebb3;
}
//transforma pra graus o cos
pangdebb3= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +1][j+1],molS[i][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j],molS[i+1][j+1],molS[i][
    j+2],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i+1][j],molS[i+1][j+1],molS[i][j+2],
    sem,indice)*angdebb3)-getPeriU(molS[i][j],molS[
    i+1][j],molS[i+1][j+1],molS[i][j+2],sem,indice)
    ))));
//arredondamento TINKER
pangdebb3= (Math.round((pangdebb3 * 10000.0))) /
    10000.0;
pangdebb3= pangdebb3+auxebb3;
auxebb3= pangdebb3;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
```

```
            pangdebb3= (getVnU(molS[i][j],molS[i+1][j],molS[
                i+1][j+1],molS[i][j+2],sem,num,indice)/
                getPath(molS[i][j],molS[i+1][j],molS[i+1][j
                +1],molS[i][j+2],sem,num,indice)*(1+(Math.cos
                ((getN(molS[i][j],molS[i+1][j],molS[i+1][j
                +1],molS[i][j+2],sem,indice)*angdebb3)-
                getPeriU(molS[i][j],molS[i+1][j],molS[i+1][j
                +1],molS[i][j+2],sem,indice)))));
            //arredondamento TINKER
            pangdebb3= (Math.round((pangdebb3 * 10000.0))) /
                 10000.0;
            pangdebb3= pangdebb3+auxebb3;
            auxebb3= pangdebb3;
            indiceD= indiceD + 1;
            ind++;
            sem++;
           }
}
if(((i>=1)&&(j<qntdcol-2)&&(molS[i][j]!="") && (molS[i
    -1][j]!="") && (molS[i-1][j+1]!="") && (molS[i][j
    +2]!="")){
  if ((((molS[i][j].substring(9, 10).equals("1"))&&(
      molS[i-1][j].substring(9, 10).equals("1"))))&&(
      molS[i-1][j].substring(8, 9).equals("1"))&&(molS[
      i-1][j+1].substring(8, 9).equals("1"))&&
        (molS[i-1][j+1].substring(10, 11).equals("1"))&(
            molS[i][j+2].substring(11, 12).equals("1"))){
      //isolar a b e c dos planos
      indice=0;
      aux1c1= objAtomos.setAtomoU(molS[i][j],torS[r]);
      aux2c1= objAtomos.setAtomoU(molS[i-1][j],torS[r]);
      aux3c1= objAtomos.setAtomoU(molS[i-1][j+1],torS[r
          ]);
      aux4c1= objAtomos.setAtomoU(molS[i][j+2],torS[r]);
      v1= objAtomos.getX(aux1c1,torS[r])-objAtomos.getX(
          aux2c1,torS[r]);
```

```
x1= objAtomos.getY(aux1c1,torS[r])-objAtomos.getY(
    aux2c1,torS[r]);
z1= objAtomos.getZ(aux1c1,torS[r])-objAtomos.getZ(
    aux2c1,torS[r]);
v2= objAtomos.getX(aux3c1,torS[r])-objAtomos.getX(
    aux2c1,torS[r]);
x2= objAtomos.getY(aux3c1,torS[r])-objAtomos.getY(
    aux2c1,torS[r]);
z2= objAtomos.getZ(aux3c1,torS[r])-objAtomos.getZ(
    aux2c1,torS[r]);
v3= objAtomos.getX(aux2c1,torS[r])-objAtomos.getX(
    aux3c1,torS[r]);
x3= objAtomos.getY(aux2c1,torS[r])-objAtomos.getY(
    aux3c1,torS[r]);
z3= objAtomos.getZ(aux2c1,torS[r])-objAtomos.getZ(
    aux3c1,torS[r]);
v4= objAtomos.getX(aux4c1,torS[r])-objAtomos.getX(
    aux3c1,torS[r]);
x4= objAtomos.getY(aux4c1,torS[r])-objAtomos.getY(
    aux3c1,torS[r]);
z4= objAtomos.getZ(aux4c1,torS[r])-objAtomos.getZ(
    aux3c1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdc1= (prod)/(pmod);
angdc1= Math.acos(auxdc1);
```

```
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc1= -angdc1;
}
//transforma pra graus o cos
pangdc1= (getVnU(molS[i][j],molS[i-1][j],molS[i
    -1][j+1],molS[i][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i-1][j],molS[i-1][j+1],molS[i][
    j+2],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i-1][j],molS[i-1][j+1],molS[i][j+2],
    sem,indice)*angdc1)-getPeriU(molS[i][j],molS[i
    -1][j],molS[i-1][j+1],molS[i][j+2],sem,indice))
    )));
//arredondamento TINKER
pangdc1= (Math.round((pangdc1 * 10000.0))) /
    10000.0;
pangdc1= pangdc1+auxc1;
auxc1= pangdc1;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdc1= (getVnU(molS[i][j],molS[i-1][j],molS[i
      -1][j+1],molS[i][j+2],sem,num,indice)/getPath
      (molS[i][j],molS[i-1][j],molS[i-1][j+1],molS[
      i][j+2],sem,num,indice)*(1+(Math.cos((getN(
      molS[i][j],molS[i-1][j],molS[i-1][j+1],molS[i
      ][j+2],sem,indice)*angdc1)-getPeriU(molS[i][j
      ],molS[i-1][j],molS[i-1][j+1],molS[i][j+2],
      sem,indice)))));
  //arredondamento TINKER
  pangdc1= (Math.round((pangdc1 * 10000.0))) /
      10000.0;
  pangdc1= pangdc1+auxc1;
  auxc1= pangdc1;
```

```
            indiceD= indiceD + 1;
            ind++;
            sem++;
           }
      }//fim if
    }
    if ((((i<qntdlinha-2)&&(j>=3)&&(molS[i][j]!="") && (
        molS[i][j-3]!="") && (molS[i+1][j-3]!="") && (molS[
        i+2][j-3]!="")))&&
      ((molS[i][j-1]=="X00000001000")&&(molS[i][j-2]=="
          X00000001000")))){
      int x=j-3;
      if((j>=1)&&(molS[i][j]!="") && (molS[i][x]!="") && (
          molS[i+1][x]!="") && (molS[i+2][x]!="")){
        if ((((molS[i][j].substring(8, 9).equals("1"))&&(
            molS[i][x].substring(8, 9).equals("1"))))&&(
            molS[i+1][x].substring(9, 10).equals("1"))&&(
            molS[i+1][x].substring(9, 10).equals("1"))&&
           (molS[i+1][x].substring(9, 10).equals("1"))&(
              molS[i][x].substring(9, 10).equals("1"))){
           indice=0;
           aux1bb= objAtomos.setAtomoU(molS[i][j],torS[r]);
           aux2bb= objAtomos.setAtomoU(molS[i][x],torS[r]);
           aux3bb= objAtomos.setAtomoU(molS[i+1][x],torS[r
               ]);
           aux4bb= objAtomos.setAtomoU(molS[i+2][x],torS[r
               ]);
           v1= objAtomos.getX(aux1bb,torS[r])-objAtomos.
               getX(aux2bb,torS[r]);
           x1= objAtomos.getY(aux1bb,torS[r])-objAtomos.
               getY(aux2bb,torS[r]);
           z1= objAtomos.getZ(aux1bb,torS[r])-objAtomos.
               getZ(aux2bb,torS[r]);
           v2= objAtomos.getX(aux3bb,torS[r])-objAtomos.
               getX(aux2bb,torS[r]);
```

```
x2= objAtomos.getY(aux3bb,torS[r])-objAtomos.
    getY(aux2bb,torS[r]);
z2= objAtomos.getZ(aux3bb,torS[r])-objAtomos.
    getZ(aux2bb,torS[r]);
v3= objAtomos.getX(aux2bb,torS[r])-objAtomos.
    getX(aux3bb,torS[r]);
x3= objAtomos.getY(aux2bb,torS[r])-objAtomos.
    getY(aux3bb,torS[r]);
z3= objAtomos.getZ(aux2bb,torS[r])-objAtomos.
    getZ(aux3bb,torS[r]);
v4= objAtomos.getX(aux4bb,torS[r])-objAtomos.
    getX(aux3bb,torS[r]);
x4= objAtomos.getY(aux4bb,torS[r])-objAtomos.
    getY(aux3bb,torS[r]);
z4= objAtomos.getZ(aux4bb,torS[r])-objAtomos.
    getZ(aux3bb,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb= (prod)/(pmod);
angdbb= Math.acos(auxdbb);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb= -angdbb;
}
pangdbb= (getVnU(molS[i][j],molS[i][x],molS[i
    +1][x],molS[i+2][x],sem,num,indice)/getPath(
```

```
                molS[i][j],molS[i][x],molS[i+1][x],molS[i+2][
                x],sem,num,indice)*(1+(Math.cos((getN(molS[i
                ][j],molS[i][x],molS[i+1][x],molS[i+2][x],sem
                ,indice)*angdbb)-getPeriU(molS[i][j],molS[i][
                x],molS[i+1][x],molS[i+2][x],sem,indice)))));
            //arredondamento TINKER
            pangdbb= (Math.round((pangdbb * 10000.0))) /
                10000.0;
            pangdbb= pangdbb+auxbb;
            auxbb= pangdbb;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdbb= (getVnU(molS[i][j],molS[i][x],molS[i
                  +1][x],molS[i+2][x],sem,num,indice)/getPath
                  (molS[i][x],molS[i][x],molS[i+1][x],molS[i
                  +2][x],sem,num,indice)*(1+(Math.cos((getN(
                  molS[i][x],molS[i][x],molS[i+1][x],molS[i
                  +2][x],sem,indice)*angdbb)-getPeriU(molS[i
                  ][x],molS[i][x],molS[i+1][x],molS[i+2][x],
                  sem,indice)))));
              //arredondamento TINKER
              pangdbb= (Math.round((pangdbb * 10000.0))) /
                  10000.0;
              pangdbb= pangdbb+auxbb;
              auxbb= pangdbb;
              indiceD= indiceD + 1;
              ind++;
              sem++;
             }
          }
        }
      }else if ((i<qntdlinha-2)&&(j>=1)&&(molS[i][j]!="") &&
          (molS[i][j-1]!="") && (molS[i+1][j-1]!="") && (
```

```
molS[i+2][j-1]!="")&&(((molS[i][j].substring(8, 9).
equals("1"))&&(molS[i][j-1].substring(8, 9).equals
("1"))))&&(molS[i][j-1].substring(9, 10).equals
("1"))&&(molS[i+1][j-1].substring(9, 10).equals
("1"))&&(molS[i+1][j-1].substring(9, 10).equals
("1"))&(molS[i+2][j-1].substring(9, 10).equals("1")
)&&((molS[i][j-1]!="X00000001000")&&(molS[i][j]!="
X00000001000"))){
  if (((molS[i][j-1].substring(0, 7).equals("OHR2001
      "))&&(molS[i+1][j-1].substring(0, 7).equals("
      P1P0001")))||((molS[i+1][j-1].substring(0, 7).
      equals("OHR2001"))&&(molS[i+2][j-1].substring
      (0, 7).equals("P1P0001")))){

  }else{
    //isolar a b e c dos planos
    indice=0;
    aux1bb= objAtomos.setAtomoU(molS[i][j],torS[r]);
    aux2bb= objAtomos.setAtomoU(molS[i][j-1],torS[r
        ]);
    aux3bb= objAtomos.setAtomoU(molS[i+1][j-1],torS[
        r]);
    aux4bb= objAtomos.setAtomoU(molS[i+2][j-1],torS[
        r]);
    v1= objAtomos.getX(aux1bb,torS[r])-objAtomos.
        getX(aux2bb,torS[r]);
    x1= objAtomos.getY(aux1bb,torS[r])-objAtomos.
        getY(aux2bb,torS[r]);
    z1= objAtomos.getZ(aux1bb,torS[r])-objAtomos.
        getZ(aux2bb,torS[r]);
    v2= objAtomos.getX(aux3bb,torS[r])-objAtomos.
        getX(aux2bb,torS[r]);
    x2= objAtomos.getY(aux3bb,torS[r])-objAtomos.
        getY(aux2bb,torS[r]);
    z2= objAtomos.getZ(aux3bb,torS[r])-objAtomos.
        getZ(aux2bb,torS[r]);
```

```
v3= objAtomos.getX(aux2bb,torS[r])-objAtomos.
    getX(aux3bb,torS[r]);
x3= objAtomos.getY(aux2bb,torS[r])-objAtomos.
    getY(aux3bb,torS[r]);
z3= objAtomos.getZ(aux2bb,torS[r])-objAtomos.
    getZ(aux3bb,torS[r]);
v4= objAtomos.getX(aux4bb,torS[r])-objAtomos.
    getX(aux3bb,torS[r]);
x4= objAtomos.getY(aux4bb,torS[r])-objAtomos.
    getY(aux3bb,torS[r]);
z4= objAtomos.getZ(aux4bb,torS[r])-objAtomos.
    getZ(aux3bb,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb= (prod)/(pmod);
angdbb= Math.acos(auxdbb);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb= -angdbb;
}
pangdbb= (getVnU(molS[i][j],molS[i][j-1],molS[i
    +1][j-1],molS[i+2][j-1],sem,num,indice)/
    getPath(molS[i][j],molS[i][j-1],molS[i+1][j
    -1],molS[i+2][j-1],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i][j-1],molS[i+1][j
    -1],molS[i+2][j-1],sem,indice)*angdbb)-
```

```
            getPeriU(molS[i][j],molS[i][j-1],molS[i+1][j
            -1],molS[i+2][j-1],sem,indice)))));
          //arredondamento TINKER
          pangdbb= (Math.round((pangdbb * 10000.0))) /
            10000.0;
          pangdbb= pangdbb+auxbb;
          auxbb= pangdbb;
          indiceD= indiceD + 1;
          ind++;
          sem++;
          while (num==0){
            //transforma pra graus o cos
            pangdbb= (getVnU(molS[i][j],molS[i][j-1],molS[
                i+1][j-1],molS[i+2][j-1],sem,num,indice)/
                getPath(molS[i][j],molS[i][j-1],molS[i+1][j
                -1],molS[i+2][j-1],sem,num,indice)*(1+(Math
                .cos((getN(molS[i][j],molS[i][j-1],molS[i
                +1][j-1],molS[i+2][j-1],sem,indice)*angdbb)
                -getPeriU(molS[i][j],molS[i][j-1],molS[i
                +1][j-1],molS[i+2][j-1],sem,indice)))));
            //arredondamento TINKER
            pangdbb= (Math.round((pangdbb * 10000.0))) /
                10000.0;
            pangdbb= pangdbb+auxbb;
            auxbb= pangdbb;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }//fim if
      if(((i<qntdlinha-2)&&(j<qntdcol-1)&&(molS[i][j]!="")
        && (molS[i][j+1]!="") && (molS[i+1][j+1]!="") && (
        molS[i+2][j+1]!=""))){
        if ((((molS[i][j].substring(8, 9).equals("1"))&&(
            molS[i][j+1].substring(8, 9).equals("1"))))&&(
```

```
molS[i][j+1].substring(9, 10).equals("1"))&&(molS
[i+1][j+1].substring(9, 10).equals("1"))&&(molS[i
+1][j+1].substring(9, 10).equals("1"))&(molS[i
+2][j+1].substring(9, 10).equals("1"))){
if (((molS[i][j+1].substring(0, 7).equals("OHR2001
"))&&(molS[i+1][j+1].substring(0, 7).equals("
P1P0001")))||((molS[i+1][j+1].substring(0, 7).
equals("OHR2001"))&&(molS[i+2][j+1].substring
(0, 7).equals("P1P0001")))){

}else{
  indice=0;
  //isolar a b e c dos planos
  aux1be= objAtomos.setAtomoU(molS[i][j],torS[r]);
  aux2be= objAtomos.setAtomoU(molS[i][j+1],torS[r
      ]);
  aux3be= objAtomos.setAtomoU(molS[i+1][j+1],torS[
      r]);
  aux4be= objAtomos.setAtomoU(molS[i+2][j+1],torS[
      r]);
  v1= objAtomos.getX(aux1be,torS[r])-objAtomos.
      getX(aux2be,torS[r]);
  x1= objAtomos.getY(aux1be,torS[r])-objAtomos.
      getY(aux2be,torS[r]);
  z1= objAtomos.getZ(aux1be,torS[r])-objAtomos.
      getZ(aux2be,torS[r]);
  v2= objAtomos.getX(aux3be,torS[r])-objAtomos.
      getX(aux2be,torS[r]);
  x2= objAtomos.getY(aux3be,torS[r])-objAtomos.
      getY(aux2be,torS[r]);
  z2= objAtomos.getZ(aux3be,torS[r])-objAtomos.
      getZ(aux2be,torS[r]);
  v3= objAtomos.getX(aux2be,torS[r])-objAtomos.
      getX(aux3be,torS[r]);
  x3= objAtomos.getY(aux2be,torS[r])-objAtomos.
      getY(aux3be,torS[r]);
```

```
z3= objAtomos.getZ(aux2be,torS[r])-objAtomos.
    getZ(aux3be,torS[r]);
v4= objAtomos.getX(aux4be,torS[r])-objAtomos.
    getX(aux3be,torS[r]);
x4= objAtomos.getY(aux4be,torS[r])-objAtomos.
    getY(aux3be,torS[r]);
z4= objAtomos.getZ(aux4be,torS[r])-objAtomos.
    getZ(aux3be,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbe= (prod)/(pmod);
angdbe= Math.acos(auxdbe);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbe= -angdbe;
}
pangdbe= (getVnU(molS[i][j],molS[i][j+1],molS[i
    +1][j+1],molS[i+2][j+1],sem,num,indice)/
    getPath(molS[i][j],molS[i][j+1],molS[i+1][j
    +1],molS[i+2][j+1],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i][j+1],molS[i+1][j
    +1],molS[i+2][j+1],sem,indice)*angdbe)-
    getPeriU(molS[i][j],molS[i][j+1],molS[i+1][j
    +1],molS[i+2][j+1],sem,indice)))));
//arredondamento TINKER
```

```
            pangdbe= (Math.round((pangdbe * 10000.0))) /
                10000.0;
            pangdbe= pangdbe+auxbe;
            auxbe= pangdbe;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdbe= (getVnU(molS[i][j],molS[i][j+1],molS[
                  i+1][j+1],molS[i+2][j+1],sem,num,indice)/
                  getPath(molS[i][j],molS[i][j+1],molS[i+1][j
                  +1],molS[i+2][j+1],sem,num,indice)*(1+(Math
                  .cos((getN(molS[i][j],molS[i][j+1],molS[i
                  +1][j+1],molS[i+2][j+1],sem,indice)*angdbe)
                  -getPeriU(molS[i][j],molS[i][j+1],molS[i
                  +1][j+1],molS[i+2][j+1],sem,indice)))));
              //arredondamento TINKER
              pangdbe= (Math.round((pangdbe * 10000.0))) /
                  10000.0;
              pangdbe= pangdbe+auxbe;
              auxbe= pangdbe;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }//fim if
      if(((i<qntdlinha-1)&&(j<qntdcol-4)&&(molS[i][j]!="")
          && (molS[i+1][j+1]!="") && (molS[i+1][j+2]!="") &&
          (molS[i][j+1]!=""))&&  ((molS[i+1][j+2]=="
          X00000001000")&&(molS[i+1][j+3]=="X00000001000"))){
        int x=j+4;
        if((molS[i][j]!="") && (molS[i+1][j+1]!="") && (molS
            [i+1][x]!="") && (molS[i][x-1]!="")){
```

```
if ((((molS[i][j].substring(10, 11).equals("1"))
   &&(molS[i+1][j+1].substring(11, 12).equals("1")
   )))&&(molS[i+1][j+1].substring(8, 9).equals
   ("1"))&&(molS[i+1][x].substring(8, 9).equals
   ("1"))&&(molS[i+1][x].substring(11, 12).equals
   ("1"))&(molS[i][x-1].substring(10, 11).equals
   ("1")))){
  indice=0;
  aux1b3= objAtomos.setAtomoU(molS[i][j],torS[r]);
  aux2b3= objAtomos.setAtomoU(molS[i+1][j+1],torS[
     r]);
  aux3b3= objAtomos.setAtomoU(molS[i+1][x],torS[r
     ]);
  aux4b3= objAtomos.setAtomoU(molS[i][x-1],torS[r
     ]);
  v1= objAtomos.getX(aux1b3,torS[r])-objAtomos.
     getX(aux2b3,torS[r]);
  x1= objAtomos.getY(aux1b3,torS[r])-objAtomos.
     getY(aux2b3,torS[r]);
  z1= objAtomos.getZ(aux1b3,torS[r])-objAtomos.
     getZ(aux2b3,torS[r]);
  v2= objAtomos.getX(aux3b3,torS[r])-objAtomos.
     getX(aux2b3,torS[r]);
  x2= objAtomos.getY(aux3b3,torS[r])-objAtomos.
     getY(aux2b3,torS[r]);
  z2= objAtomos.getZ(aux3b3,torS[r])-objAtomos.
     getZ(aux2b3,torS[r]);
  v3= objAtomos.getX(aux2b3,torS[r])-objAtomos.
     getX(aux3b3,torS[r]);
  x3= objAtomos.getY(aux2b3,torS[r])-objAtomos.
     getY(aux3b3,torS[r]);
  z3= objAtomos.getZ(aux2b3,torS[r])-objAtomos.
     getZ(aux3b3,torS[r]);
  v4= objAtomos.getX(aux4b3,torS[r])-objAtomos.
     getX(aux3b3,torS[r]);
```

```
x4= objAtomos.getY(aux4b3,torS[r])-objAtomos.
    getY(aux3b3,torS[r]);
z4= objAtomos.getZ(aux4b3,torS[r])-objAtomos.
    getZ(aux3b3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb3= (prod)/(pmod);
angdb3= Math.acos(auxdb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb3= -angdb3;
}
//transforma pra graus o cos
pangdb3= (getVnU(molS[i][j],molS[i+1][j+1],molS[
    i+1][x],molS[i][x-1],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j+1],molS[i+1][x],molS[i
    ][x-1],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j+1],molS[i+1][x],molS[i
    ][x-1],sem,indice)*angdb3)-getPeriU(molS[i][j
    ],molS[i+1][j+1],molS[i+1][x],molS[i][x-1],
    sem,indice)))));
//arredondamento TINKER
pangdb3= (Math.round((pangdb3 * 10000.0))) /
    10000.0;
pangdb3= pangdb3+auxb3;
auxb3= pangdb3;
```

```
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdb3= (getVnU(molS[i][j],molS[i+1][j+1],
                  molS[i+1][x],molS[i][x-1],sem,num,indice)/
                  getPath(molS[i][j],molS[i+1][j+1],molS[i
                  +1][x],molS[i][x-1],sem,num,indice)*(1+(
                  Math.cos((getN(molS[i][j],molS[i+1][j+1],
                  molS[i+1][x],molS[i][x-1],sem,indice)*
                  angdb3)-getPeriU(molS[i][j],molS[i+1][j+1],
                  molS[i+1][x],molS[i][x-1],sem,indice)))));
              //arredondamento TINKER
              pangdb3= (Math.round((pangdb3 * 10000.0))) /
                  10000.0;
              pangdb3= pangdb3+auxb3;
              auxb3= pangdb3;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }
      if(((i<qntdlinha-1)&&(j<qntdcol-3)&&(molS[i][j]!="")
         && (molS[i+1][j+1]!="") && (molS[i+1][j+2]!="") &&
         (molS[i][j+3]!=""))&&
          //posicao dois tem X
          //((i<qntdlinha-2)&&(j>=3)&&(molS[i][j]!="") && (
              molS[i][j-3]!="") && (molS[i+1][j-3]!="") && (
              molS[i+2][j-3]!="")) ||((molS[i+1][j+2]=="
              X00000001000")&&(molS[i+1][j+3]=="X00000001000
              "))){
        int x=j+4;
```

```
if((molS[i][j]!="") && (molS[i+1][j+1]!="") && (molS
    [i+1][x]!="") && (molS[i][x+1]!="")){
  if ((((molS[i][j].substring(10, 11).equals("1"))
     &&(molS[i+1][j+1].substring(11, 12).equals("1")
     )))&&(molS[i+1][j+1].substring(8, 9).equals
     ("1"))&&(molS[i+1][x].substring(8, 9).equals
     ("1"))&&(molS[i+1][x].substring(10, 11).equals
     ("1"))&(molS[i][x+1].substring(11, 12).equals
     ("1"))){
    indice=0;
    aux1bb3= objAtomos.setAtomoU(molS[i][j],torS[r])
        ;
    aux2bb3= objAtomos.setAtomoU(molS[i+1][j+1],torS
        [r]);
    aux3bb3= objAtomos.setAtomoU(molS[i+1][x],torS[r
        ]);
    aux4bb3= objAtomos.setAtomoU(molS[i][x+1],torS[r
        ]);
    v1= objAtomos.getX(aux1bb3,torS[r])-objAtomos.
        getX(aux2bb3,torS[r]);
    x1= objAtomos.getY(aux1bb3,torS[r])-objAtomos.
        getY(aux2bb3,torS[r]);
    z1= objAtomos.getZ(aux1bb3,torS[r])-objAtomos.
        getZ(aux2bb3,torS[r]);
    v2= objAtomos.getX(aux3bb3,torS[r])-objAtomos.
        getX(aux2bb3,torS[r]);
    x2= objAtomos.getY(aux3bb3,torS[r])-objAtomos.
        getY(aux2bb3,torS[r]);
    z2= objAtomos.getZ(aux3bb3,torS[r])-objAtomos.
        getZ(aux2bb3,torS[r]);
    v3= objAtomos.getX(aux2bb3,torS[r])-objAtomos.
        getX(aux3bb3,torS[r]);
    x3= objAtomos.getY(aux2bb3,torS[r])-objAtomos.
        getY(aux3bb3,torS[r]);
    z3= objAtomos.getZ(aux2bb3,torS[r])-objAtomos.
        getZ(aux3bb3,torS[r]);
```

```
v4= objAtomos.getX(aux4bb3,torS[r])-objAtomos.
    getX(aux3bb3,torS[r]);
x4= objAtomos.getY(aux4bb3,torS[r])-objAtomos.
    getY(aux3bb3,torS[r]);
z4= objAtomos.getZ(aux4bb3,torS[r])-objAtomos.
    getZ(aux3bb3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb3= (prod)/(pmod);
angdbb3= Math.acos(auxdbb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb3= -angdbb3;
}
//transforma pra graus o cos
pangdbb3= (getVnU(molS[i][j],molS[i+1][j+1],molS
    [i+1][x],molS[i][x+1],sem,num,indice)/getPath
    (molS[i][j],molS[i+1][j+1],molS[i+1][x],molS[
    i][x+1],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j+1],molS[i+1][x],molS[i
    ][x+1],sem,indice)*angdbb3)-getPeriU(molS[i][
    j],molS[i+1][j+1],molS[i+1][x],molS[i][x+1],
    sem,indice)))));
//arredondamento TINKER
pangdbb3= (Math.round((pangdbb3 * 10000.0))) /
    10000.0;
```

```
            pangdbb3= pangdbb3+auxbb3;
            auxbb3= pangdbb3;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdbb3= (getVnU(molS[i][j],molS[i+1][j+1],
                  molS[i+1][x],molS[i][x+1],sem,num,indice)/
                  getPath(molS[i][j],molS[i+1][j+1],molS[i
                  +1][x],molS[i][x+1],sem,num,indice)*(1+(
                  Math.cos((getN(molS[i][j],molS[i+1][j+1],
                  molS[i+1][x],molS[i][x+1],sem,indice)*
                  angdbb3)-getPeriU(molS[i][j],molS[i+1][j
                  +1],molS[i+1][x],molS[i][x+1],sem,indice)))
                  ));
              //arredondamento TINKER
              pangdbb3= (Math.round((pangdbb3 * 10000.0))) /
                   10000.0;
              pangdbb3= pangdbb3+auxbb3;
              auxbb3= pangdbb3;
              indiceD= indiceD + 1;
              ind++;
              sem++;
             }
          }
        }
      }
      if(((i<qntdlinha-3)&&(j<qntdcol-1)&&(molS[i][j]!="")
         && (molS[i+1][j+1]!="") && (molS[i+2][j+1]!="") &&
         (molS[i+3][j+1]!=""))){
          if (((((molS[i][j].substring(10, 11).equals("1"))
             &&(molS[i+1][j+1].substring(11, 12).equals("1")
             )))&&(molS[i+1][j+1].substring(9, 10).equals
             ("1"))&&(molS[i+2][j+1].substring(9, 10).equals
             ("1"))&&(molS[i+2][j+1].substring(9, 10).equals
```

```
    ("1"))&(molS[i+3][j+1].substring(9, 10).equals
    ("1"))){
//isolar a b e c dos planos
indice=0;
aux1bb2= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2bb2= objAtomos.setAtomoU(molS[i+1][j+1],torS[r
    ]);
aux3bb2= objAtomos.setAtomoU(molS[i+2][j+1],torS[r
    ]);
aux4bb2= objAtomos.setAtomoU(molS[i+3][j+1],torS[r
    ]);
v1= objAtomos.getX(aux1bb2,torS[r])-objAtomos.getX
    (aux2bb2,torS[r]);
x1= objAtomos.getY(aux1bb2,torS[r])-objAtomos.getY
    (aux2bb2,torS[r]);
z1= objAtomos.getZ(aux1bb2,torS[r])-objAtomos.getZ
    (aux2bb2,torS[r]);
v2= objAtomos.getX(aux3bb2,torS[r])-objAtomos.getX
    (aux2bb2,torS[r]);
x2= objAtomos.getY(aux3bb2,torS[r])-objAtomos.getY
    (aux2bb2,torS[r]);
z2= objAtomos.getZ(aux3bb2,torS[r])-objAtomos.getZ
    (aux2bb2,torS[r]);
v3= objAtomos.getX(aux2bb2,torS[r])-objAtomos.getX
    (aux3bb2,torS[r]);
x3= objAtomos.getY(aux2bb2,torS[r])-objAtomos.getY
    (aux3bb2,torS[r]);
z3= objAtomos.getZ(aux2bb2,torS[r])-objAtomos.getZ
    (aux3bb2,torS[r]);
v4= objAtomos.getX(aux4bb2,torS[r])-objAtomos.getX
    (aux3bb2,torS[r]);
x4= objAtomos.getY(aux4bb2,torS[r])-objAtomos.getY
    (aux3bb2,torS[r]);
z4= objAtomos.getZ(aux4bb2,torS[r])-objAtomos.getZ
    (aux3bb2,torS[r]);
a1= ((x1*z2)-(x2*z1));
```

```
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdbb2= (prod)/(pmod);
//angdl= java.lang.Math.cos(aux/(180*Math.PI));
angdbb2= Math.acos(auxdbb2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb2= -angdbb2;
}
pangdbb2= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
    +2][j+1],molS[i+3][j+1],sem,num,indice)/getPath
    (molS[i][j],molS[i+1][j+1],molS[i+2][j+1],molS[
    i+3][j+1],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j+1],molS[i+2][j+1],molS[i
    +3][j+1],sem,indice)*angdbb2)-getPeriU(molS[i][
    j],molS[i+1][j+1],molS[i+2][j+1],molS[i+3][j
    +1],sem,indice)))));
//arredondamento TINKER
pangdbb2= (Math.round((pangdbb2 * 10000.0))) /
    10000.0;
pangdbb2= pangdbb2+auxbb2;
auxbb2= pangdbb2;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
```

```
        pangdbb2= (getVnU(molS[i][j],molS[i+1][j+1],molS
            [i+2][j+1],molS[i+3][j+1],sem,num,indice)/
            getPath(molS[i][j],molS[i+1][j+1],molS[i+2][j
            +1],molS[i+3][j+1],sem,num,indice)*(1+(Math.
            cos((getN(molS[i][j],molS[i+1][j+1],molS[i
            +2][j+1],molS[i+3][j+1],sem,indice)*angdbb2)-
            getPeriU(molS[i][j],molS[i+1][j+1],molS[i+2][
            j+1],molS[i+3][j+1],sem,indice)))));
        //arredondamento TINKER
        pangdbb2= (Math.round((pangdbb2 * 10000.0))) /
            10000.0;
        pangdbb2= pangdbb2+auxbb2;
        auxbb2= pangdbb2;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }//fim if
  }//fim if
  if((i<qntdlinha-2)&&(j>=1)&&(molS[i][j]!="") && (molS[
     i+1][j]!="") && (molS[i+1][j-1]!="") && (molS[i+2][
     j-1]!="")){
    if ((molS[i][j].substring(9, 10).equals("1"))&&(molS
        [i+1][j].substring(9, 10).equals("1"))&&(molS[i
        +1][j].substring(8, 9).equals("1"))&&(molS[i+1][j
        -1].substring(8, 9).equals("1"))&&
         (molS[i+1][j-1].substring(9, 10).equals("1"))&&(
            molS[i+2][j-1].substring(9, 10).equals("1")))
            {
      indice=0;
      //isolar a b e c dos planos
      aux1c= objAtomos.setAtomoU(molS[i][j],torS[r]);
      aux2c= objAtomos.setAtomoU(molS[i+1][j],torS[r]);
      aux3c= objAtomos.setAtomoU(molS[i+1][j-1],torS[r])
          ;
```

```
aux4c= objAtomos.setAtomoU(molS[i+2][j-1],torS[r])
    ;
v1= objAtomos.getX(aux1c,torS[r])-objAtomos.getX(
    aux2c,torS[r]);
x1= objAtomos.getY(aux1c,torS[r])-objAtomos.getY(
    aux2c,torS[r]);
z1= objAtomos.getZ(aux1c,torS[r])-objAtomos.getZ(
    aux2c,torS[r]);
v2= objAtomos.getX(aux3c,torS[r])-objAtomos.getX(
    aux2c,torS[r]);
x2= objAtomos.getY(aux3c,torS[r])-objAtomos.getY(
    aux2c,torS[r]);
z2= objAtomos.getZ(aux3c,torS[r])-objAtomos.getZ(
    aux2c,torS[r]);
v3= objAtomos.getX(aux2c,torS[r])-objAtomos.getX(
    aux3c,torS[r]);
x3= objAtomos.getY(aux2c,torS[r])-objAtomos.getY(
    aux3c,torS[r]);
z3= objAtomos.getZ(aux2c,torS[r])-objAtomos.getZ(
    aux3c,torS[r]);
v4= objAtomos.getX(aux4c,torS[r])-objAtomos.getX(
    aux3c,torS[r]);
x4= objAtomos.getY(aux4c,torS[r])-objAtomos.getY(
    aux3c,torS[r]);
z4= objAtomos.getZ(aux4c,torS[r])-objAtomos.getZ(
    aux3c,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
```

```
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdc= (prod)/(pmod);
angdc= Math.acos(auxdc);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc= -angdc;
}
pangdc= (getVnU(molS[i][j],molS[i+1][j],molS[i+1][
    j-1],molS[i+2][j-1],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j],molS[i+1][j-1],molS[i
    +2][j-1],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j],molS[i+1][j-1],molS[i
    +2][j-1],sem,indice)*angdc)-getPeriU(molS[i][j
    ],molS[i+1][j],molS[i+1][j-1],molS[i+2][j-1],
    sem,indice)))));
//arredondamento TINKER
pangdc= (Math.round((pangdc * 10000.0))) /
    10000.0;
pangdc= pangdc+auxc;
auxc= pangdc;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdc= (getVnU(molS[i][j],molS[i+1][j],molS[i
      +1][j-1],molS[i+2][j-1],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j],molS[i+1][j
      -1],molS[i+2][j-1],sem,num,indice)*(1+(Math.
      cos((getN(molS[i][j],molS[i+1][j],molS[i+1][j
      -1],molS[i+2][j-1],sem,indice)*angdc)-
      getPeriU(molS[i][j],molS[i+1][j],molS[i+1][j
      -1],molS[i+2][j-1],sem,indice)))));
  //arredondamento TINKER
```

```
            pangdc= (Math.round((pangdc * 10000.0))) /
                10000.0;
            pangdc= pangdc+auxc;
            auxc= pangdc;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim else
      }//fim if
      if ((((j<qntdcol-5)&&(molS[i][j]!="") &&(i!=0)&& (molS[
          i-1][j+1]!="") && (molS[i][j+2]!="") && (molS[i][j
          +5]!="")))&&
        //esses if sao caso encontrem o valor XX, ai pula
            duas casas (verificar se esta certo)
        ((molS[i][j+3]=="X00000001000")&&(molS[i][j+4]=="
            X00000001000"))){
        int x=j+5;
        if((j<qntdcol-5)&&(molS[i][j]!="") &&(i!=0)&& (molS[
            i-1][j+1]!="") && (molS[i][j+2]!="") && (molS[i][
            x]!="")){
          if ((molS[i][j].substring(10, 11).equals("1"))&&(
              molS[i-1][j+1].substring(11, 12).equals("1"))
              &&(molS[i-1][j+1].substring(10, 11).equals("1")
              )&&(molS[i][j+2].substring(11, 12).equals("1"))
              &&(molS[i][j+2].substring(8, 9).equals("1"))&&(
              molS[i][x].substring(8, 9).equals("1"))){
            indice=0;
            //isolar a b e c dos planos
            aux1cc= objAtomos.setAtomoU(molS[i][j],torS[r]);
            aux2cc= objAtomos.setAtomoU(molS[i-1][j+1],torS[
                r]);
            aux3cc= objAtomos.setAtomoU(molS[i][j+2],torS[r
                ]);
            aux4cc= objAtomos.setAtomoU(molS[i][x],torS[r]);
```

```
v1= objAtomos.getX(aux1cc,torS[r])-objAtomos.
    getX(aux2cc,torS[r]);
x1= objAtomos.getY(aux1cc,torS[r])-objAtomos.
    getY(aux2cc,torS[r]);
z1= objAtomos.getZ(aux1cc,torS[r])-objAtomos.
    getZ(aux2cc,torS[r]);
v2= objAtomos.getX(aux3cc,torS[r])-objAtomos.
    getX(aux2cc,torS[r]);
x2= objAtomos.getY(aux3cc,torS[r])-objAtomos.
    getY(aux2cc,torS[r]);
z2= objAtomos.getZ(aux3cc,torS[r])-objAtomos.
    getZ(aux2cc,torS[r]);
v3= objAtomos.getX(aux2cc,torS[r])-objAtomos.
    getX(aux3cc,torS[r]);
x3= objAtomos.getY(aux2cc,torS[r])-objAtomos.
    getY(aux3cc,torS[r]);
z3= objAtomos.getZ(aux2cc,torS[r])-objAtomos.
    getZ(aux3cc,torS[r]);
v4= objAtomos.getX(aux4cc,torS[r])-objAtomos.
    getX(aux3cc,torS[r]);
x4= objAtomos.getY(aux4cc,torS[r])-objAtomos.
    getY(aux3cc,torS[r]);
z4= objAtomos.getZ(aux4cc,torS[r])-objAtomos.
    getZ(aux3cc,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
```

```
auxdcc= (prod)/(pmod);
angdcc= Math.acos(auxdcc);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdcc= -angdcc;
}
pangdcc= (getVnU(molS[i][j],molS[i-1][j+1],molS[
    i][j+2],molS[i][x],sem,num,indice)/getPath(
    molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i
    ][x],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i-1][j+1],molS[i][j+2],molS[i][x],
    sem,indice)*angdcc)-getPeriU(molS[i][j],molS[
    i-1][j+1],molS[i][j+2],molS[i][x],sem,indice)
    ))));
//arredondamento TINKER
pangdcc= (Math.round((pangdcc * 10000.0))) /
    10000.0;
pangdcc= pangdcc+auxcc;
auxcc= pangdcc;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdcc= (getVnU(molS[i][j],molS[i-1][j+1],
      molS[i][j+2],molS[i][x],sem,num,indice)/
      getPath(molS[i][j],molS[i-1][j+1],molS[i][j
      +2],molS[i][x],sem,num,indice)*(1+(Math.cos
      ((getN(molS[i][j],molS[i-1][j+1],molS[i][j
      +2],molS[i][x],sem,indice)*angdcc)-getPeriU
      (molS[i][j],molS[i-1][j+1],molS[i][j+2],
      molS[i][x],sem,indice)))));
  //arredondamento TINKER
  pangdcc= (Math.round((pangdcc * 10000.0))) /
      10000.0;
  pangdcc= pangdcc+auxcc;
```

```
                auxcc= pangdcc;
                indiceD= indiceD + 1;
                ind++;
                sem++;
              }
            }
          }
        }else if ((j<qntdcol-3)&&(molS[i][j]!="") &&(i!=0)&& (
            molS[i-1][j+1]!="") && (molS[i][j+2]!="") && (molS[
            i][j+3]!="")&&(molS[i][j].substring(10, 11).equals
            ("1"))&&(molS[i-1][j+1].substring(11, 12).equals
            ("1"))&&(molS[i-1][j+1].substring(10, 11).equals
            ("1"))&&(molS[i][j+2].substring(11, 12).equals("1")
            )&&(molS[i][j+2].substring(8, 9).equals("1"))&&(
            molS[i][j+3].substring(8, 9).equals("1"))&& (molS[i
            ][j+3]!="X00000001000")){
            indice=0;
            //isolar a b e c dos planos
            aux1cc= objAtomos.setAtomoU(molS[i][j],torS[r]);
            aux2cc= objAtomos.setAtomoU(molS[i-1][j+1],torS[r
                ]);
            aux3cc= objAtomos.setAtomoU(molS[i][j+2],torS[r]);
            aux4cc= objAtomos.setAtomoU(molS[i][j+3],torS[r]);
            v1= objAtomos.getX(aux1cc,torS[r])-objAtomos.getX(
                aux2cc,torS[r]);
            x1= objAtomos.getY(aux1cc,torS[r])-objAtomos.getY(
                aux2cc,torS[r]);
            z1= objAtomos.getZ(aux1cc,torS[r])-objAtomos.getZ(
                aux2cc,torS[r]);
            v2= objAtomos.getX(aux3cc,torS[r])-objAtomos.getX(
                aux2cc,torS[r]);
            x2= objAtomos.getY(aux3cc,torS[r])-objAtomos.getY(
                aux2cc,torS[r]);
            z2= objAtomos.getZ(aux3cc,torS[r])-objAtomos.getZ(
                aux2cc,torS[r]);
```

```
v3= objAtomos.getX(aux2cc,torS[r])-objAtomos.getX(
    aux3cc,torS[r]);
x3= objAtomos.getY(aux2cc,torS[r])-objAtomos.getY(
    aux3cc,torS[r]);
z3= objAtomos.getZ(aux2cc,torS[r])-objAtomos.getZ(
    aux3cc,torS[r]);
v4= objAtomos.getX(aux4cc,torS[r])-objAtomos.getX(
    aux3cc,torS[r]);
x4= objAtomos.getY(aux4cc,torS[r])-objAtomos.getY(
    aux3cc,torS[r]);
z4= objAtomos.getZ(aux4cc,torS[r])-objAtomos.getZ(
    aux3cc,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdcc= (prod)/(pmod);
angdcc= Math.acos(auxdcc);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdcc= -angdcc;
}
pangdcc= (getVnU(molS[i][j],molS[i-1][j+1],molS[i
    ][j+2],molS[i][j+3],sem,num,indice)/getPath(
    molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i][
    j+3],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i-1][j+1],molS[i][j+2],molS[i][j+3],
    sem,indice)*angdcc)-getPeriU(molS[i][j],molS[i
```

```
        -1][j+1],molS[i][j+2],molS[i][j+3],sem,indice))
        ))));
      //arredondamento TINKER
      pangdcc= (Math.round((pangdcc * 10000.0))) /
        10000.0;
      pangdcc= pangdcc+auxcc;
      auxcc= pangdcc;
      indiceD= indiceD + 1;
      ind++;
      sem++;
      while (num==0){
        //transforma pra graus o cos
        pangdcc= (getVnU(molS[i][j],molS[i-1][j+1],molS[
          i][j+2],molS[i][j+3],sem,num,indice)/getPath(
          molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i
          ][j+3],sem,num,indice)*(1+(Math.cos((getN(
          molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i
          ][j+3],sem,indice)*angdcc)-getPeriU(molS[i][j
          ],molS[i-1][j+1],molS[i][j+2],molS[i][j+3],
          sem,indice)))));
        //arredondamento TINKER
        pangdcc= (Math.round((pangdcc * 10000.0))) /
          10000.0;
        pangdcc= pangdcc+auxcc;
        auxcc= pangdcc;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }//fim if
    if(((j<qntdcol-2)&&(i<qntdlinha-1)&&(molS[i][j]!="")
      &&(i!=0)&& (molS[i-1][j+1]!="") && (molS[i][j
      +2]!="") && (molS[i+1][j+2]!=""))){
      if ((molS[i][j].substring(10, 11).equals("1"))&&(
        molS[i-1][j+1].substring(11, 12).equals("1"))&&(
        molS[i-1][j+1].substring(10, 11).equals("1"))&&(
```

```
 molS[i][j+2].substring(11, 12).equals("1"))&&(
 molS[i][j+2].substring(9, 10).equals("1"))&&(molS
 [i+1][j+2].substring(9, 10).equals("1"))){
indice=0;
//isolar a b e c dos planos
aux1c2= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2c2= objAtomos.setAtomoU(molS[i-1][j+1],torS[r
    ]);
aux3c2= objAtomos.setAtomoU(molS[i][j+2],torS[r]);
aux4c2= objAtomos.setAtomoU(molS[i+1][j+2],torS[r
    ]);
v1= objAtomos.getX(aux1c2,torS[r])-objAtomos.getX(
    aux2c2,torS[r]);
x1= objAtomos.getY(aux1c2,torS[r])-objAtomos.getY(
    aux2c2,torS[r]);
z1= objAtomos.getZ(aux1c2,torS[r])-objAtomos.getZ(
    aux2c2,torS[r]);
v2= objAtomos.getX(aux3c2,torS[r])-objAtomos.getX(
    aux2c2,torS[r]);
x2= objAtomos.getY(aux3c2,torS[r])-objAtomos.getY(
    aux2c2,torS[r]);
z2= objAtomos.getZ(aux3c2,torS[r])-objAtomos.getZ(
    aux2c2,torS[r]);
v3= objAtomos.getX(aux2c2,torS[r])-objAtomos.getX(
    aux3c2,torS[r]);
x3= objAtomos.getY(aux2c2,torS[r])-objAtomos.getY(
    aux3c2,torS[r]);
z3= objAtomos.getZ(aux2c2,torS[r])-objAtomos.getZ(
    aux3c2,torS[r]);
v4= objAtomos.getX(aux4c2,torS[r])-objAtomos.getX(
    aux3c2,torS[r]);
x4= objAtomos.getY(aux4c2,torS[r])-objAtomos.getY(
    aux3c2,torS[r]);
z4= objAtomos.getZ(aux4c2,torS[r])-objAtomos.getZ(
    aux3c2,torS[r]);
a1= ((x1*z2)-(x2*z1));
```

```
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdc2= (prod)/(pmod);
angdc2= Math.acos(auxdc2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc2= -angdc2;
}
pangdc2= (getVnU(molS[i][j],molS[i-1][j+1],molS[i
    ][j+2],molS[i+1][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i
    +1][j+2],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i
    +1][j+2],sem,indice)*angdc2)-getPeriU(molS[i][j
    ],molS[i-1][j+1],molS[i][j+2],molS[i+1][j+2],
    sem,indice)))));
//arredondamento TINKER
pangdc2= (Math.round((pangdc2 * 10000.0))) /
    10000.0;
pangdc2= pangdc2+auxc2;
auxc2= pangdc2;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
```

```
            pangdc2= (getVnU(molS[i][j],molS[i-1][j+1],molS[
                i][j+2],molS[i+1][j+2],sem,num,indice)/
                getPath(molS[i][j],molS[i-1][j+1],molS[i][j
                +2],molS[i+1][j+2],sem,num,indice)*(1+(Math.
                cos((getN(molS[i][j],molS[i-1][j+1],molS[i][j
                +2],molS[i+1][j+2],sem,indice)*angdc2)-
                getPeriU(molS[i][j],molS[i-1][j+1],molS[i][j
                +2],molS[i+1][j+2],sem,indice)))));
            //arredondamento TINKER
            pangdc2= (Math.round((pangdc2 * 10000.0))) /
                10000.0;
            pangdc2= pangdc2+auxc2;
            auxc2= pangdc2;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim else
      }//fim if
      if(((j<qntdcol-2)&&(molS[i][j]!="") &&(i<qntdlinha-1)
         && (molS[i+1][j]!="") && (molS[i][j+1]!="") && (
         molS[i+1][j+2]!="")))){
        if ((molS[i][j].substring(9, 10).equals("1"))&&(molS
            [i+1][j].substring(9, 10).equals("1"))&&(molS[i
            +1][j].substring(10, 11).equals("1"))&&(molS[i][j
            +1].substring(11, 12).equals("1"))&&(molS[i][j
            +1].substring(10, 11).equals("1"))&&(molS[i+1][j
            +2].substring(11, 12).equals("1"))){

          indice=0;
          //isolar a b e c dos planos
          aux1cc2= objAtomos.setAtomoU(molS[i][j],torS[r]);
          aux2cc2= objAtomos.setAtomoU(molS[i+1][j],torS[r])
              ;
          aux3cc2= objAtomos.setAtomoU(molS[i][j+1],torS[r])
              ;
```

```
aux4cc2= objAtomos.setAtomoU(molS[i+1][j+2],torS[r
    ]);
v1= objAtomos.getX(aux1cc2,torS[r])-objAtomos.getX
    (aux2cc2,torS[r]);
x1= objAtomos.getY(aux1cc2,torS[r])-objAtomos.getY
    (aux2cc2,torS[r]);
z1= objAtomos.getZ(aux1cc2,torS[r])-objAtomos.getZ
    (aux2cc2,torS[r]);
v2= objAtomos.getX(aux3cc2,torS[r])-objAtomos.getX
    (aux2cc2,torS[r]);
x2= objAtomos.getY(aux3cc2,torS[r])-objAtomos.getY
    (aux2cc2,torS[r]);
z2= objAtomos.getZ(aux3cc2,torS[r])-objAtomos.getZ
    (aux2cc2,torS[r]);
v3= objAtomos.getX(aux2cc2,torS[r])-objAtomos.getX
    (aux3cc2,torS[r]);
x3= objAtomos.getY(aux2cc2,torS[r])-objAtomos.getY
    (aux3cc2,torS[r]);
z3= objAtomos.getZ(aux2cc2,torS[r])-objAtomos.getZ
    (aux3cc2,torS[r]);
v4= objAtomos.getX(aux4cc2,torS[r])-objAtomos.getX
    (aux3cc2,torS[r]);
x4= objAtomos.getY(aux4cc2,torS[r])-objAtomos.getY
    (aux3cc2,torS[r]);
z4= objAtomos.getZ(aux4cc2,torS[r])-objAtomos.getZ
    (aux3cc2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
```

```
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdcc2= (prod)/(pmod);
angdcc2= Math.acos(auxdcc2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdcc2= -angdcc2;
}
pangdcc2= (getVnU(molS[i][j],molS[i+1][j],molS[i][
    j+1],molS[i+1][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j],molS[i][j+1],molS[i+1][
    j+2],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i+1][j],molS[i][j+1],molS[i+1][j+2],
    sem,indice)*angdcc2)-getPeriU(molS[i][j],molS[i
    +1][j],molS[i][j+1],molS[i+1][j+2],sem,indice))
    )));
//arredondamento TINKER
pangdcc2= (Math.round((pangdcc2 * 10000.0))) /
    10000.0;
pangdcc2= pangdcc2+auxcc2;
auxcc2= pangdcc2;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdcc2= (getVnU(molS[i][j],molS[i+1][j],molS[i
      ][j+1],molS[i+1][j+2],sem,num,indice)/getPath
      (molS[i][j],molS[i+1][j],molS[i][j+1],molS[i
      +1][j+2],sem,num,indice)*(1+(Math.cos((getN(
      molS[i][j],molS[i+1][j],molS[i][j+1],molS[i
      +1][j+2],sem,indice)*angdcc2)-getPeriU(molS[i
      ][j],molS[i+1][j],molS[i][j+1],molS[i+1][j
      +2],sem,indice)))));
  //arredondamento TINKER
```

```
            pangdcc2= (Math.round((pangdcc2 * 10000.0))) /
                10000.0;
            pangdcc2= pangdcc2+auxcc2;
            auxcc2= pangdcc2;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim else
      }//fim if
      if((((j<qntdcol-4)&&(i<qntdlinha-1)&&(molS[i][j]!="")
         && (molS[i+1][j]!="") && (molS[i+1][j+3]!="") && (
         molS[i+1][j+4]!="")))&&
        (molS[i+1][j+1]=="X00000001000")&&(molS[i+1][j+2]=="
            X00000001000")){
        int x=j+3;
        if((molS[i][j]!="") && (molS[i+1][j]!="") && (molS[i
            +1][x]!="") && (molS[i+1][x+1]!="")){
          if (((((molS[i][j].substring(9, 10).equals("1"))&&(
              molS[i+1][j].substring(9, 10).equals("1"))))&&(
              molS[i+1][j].substring(8, 9).equals("1"))&&(
              molS[i+1][x].substring(8, 9).equals("1")) &&
             (molS[i+1][x].substring(8, 9).equals("1"))&&(
                molS[i+1][x+1].substring(8, 9).equals("1"))){
             indice=0;
             aux1b1= objAtomos.setAtomoU(molS[i][j],torS[r]);
             aux2b1= objAtomos.setAtomoU(molS[i+1][j],torS[r
                ]);
             aux3b1= objAtomos.setAtomoU(molS[i+1][x],torS[r
                ]);
             aux4b1= objAtomos.setAtomoU(molS[i+1][x+1],torS[
                r]);
             v1= objAtomos.getX(aux1b1,torS[r])-objAtomos.
                getX(aux2b1,torS[r]);
             x1= objAtomos.getY(aux1b1,torS[r])-objAtomos.
                getY(aux2b1,torS[r]);
```

```
z1= objAtomos.getZ(aux1b1,torS[r])-objAtomos.
    getZ(aux2b1,torS[r]);
v2= objAtomos.getX(aux3b1,torS[r])-objAtomos.
    getX(aux2b1,torS[r]);
x2= objAtomos.getY(aux3b1,torS[r])-objAtomos.
    getY(aux2b1,torS[r]);
z2= objAtomos.getZ(aux3b1,torS[r])-objAtomos.
    getZ(aux2b1,torS[r]);
v3= objAtomos.getX(aux2b1,torS[r])-objAtomos.
    getX(aux3b1,torS[r]);
x3= objAtomos.getY(aux2b1,torS[r])-objAtomos.
    getY(aux3b1,torS[r]);
z3= objAtomos.getZ(aux2b1,torS[r])-objAtomos.
    getZ(aux3b1,torS[r]);
v4= objAtomos.getX(aux4b1,torS[r])-objAtomos.
    getX(aux3b1,torS[r]);
x4= objAtomos.getY(aux4b1,torS[r])-objAtomos.
    getY(aux3b1,torS[r]);
z4= objAtomos.getZ(aux4b1,torS[r])-objAtomos.
    getZ(aux3b1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb1= (prod)/(pmod);
angdb1= Math.acos(auxdb1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
```

```
    angdb1= -angdb1;
  }
  pangdb1= (getVnU(molS[i][j],molS[i+1][j],molS[i
      +1][x],molS[i+1][x+1],sem,num,indice)/getPath
      (molS[i][j],molS[i+1][j],molS[i+1][x],molS[i
      +1][j+1],sem,num,indice)*(1+(Math.cos((getN(
      molS[i][j],molS[i+1][j],molS[i+1][x],molS[i
      +1][x+1],sem,indice)*angdb1)-getPeriU(molS[i
      ][j],molS[i+1][j],molS[i+1][x],molS[i+1][x
      +1],sem,indice)))));
  //arredondamento TINKER
  pangdb1= (Math.round((pangdb1 * 10000.0))) /
      10000.0;
  pangdb1= pangdb1+auxb1;
  auxb1= pangdb1;
  indiceD= indiceD + 1;
  ind++;
  sem++;
  while (num==0){
    //transforma pra graus o cos
    pangdb1= (getVnU(molS[i][j],molS[i+1][j],molS[
        i+1][x],molS[i+1][x+1],sem,num,indice)/
        getPath(molS[i][j],molS[i+1][j],molS[i+1][x
        ],molS[i+1][j+1],sem,num,indice)*(1+(Math.
        cos((getN(molS[i][j],molS[i+1][j],molS[i
        +1][x],molS[i+1][x+1],sem,indice)*angdb1)-
        getPeriU(molS[i][j],molS[i+1][j],molS[i+1][
        x],molS[i+1][x+1],sem,indice)))));
    //arredondamento TINKER
    pangdb1= (Math.round((pangdb1 * 10000.0))) /
        10000.0;
    pangdb1= pangdb1+auxb1;
    auxb1= pangdb1;
    indiceD= indiceD + 1;
    ind++;
    sem++;
```

```
                }
            }
        }
    }else if(((j<qntdcol-2)&&(i<qntdlinha-1)&&(molS[i][j
        ]!="") && (molS[i+1][j]!="") && (molS[i+1][j
        +1]!="") && (molS[i+1][j+2]!=""))&&
        (((molS[i][j].substring(9, 10).equals("1"))&&(molS[i
            +1][j].substring(9, 10).equals("1"))))&&(molS[i
            +1][j].substring(8, 9).equals("1"))&&(molS[i+1][j
            +1].substring(8, 9).equals("1")) &&
        (molS[i+1][j+1].substring(7,8).equals("1"))&&(molS[i
            +1][j+2].substring(8, 9).equals("1"))){
        if (((molS[i][j].substring(0, 7).equals("OHR2001"))
            && (molS[i+1][j].substring(0, 7).equals("P1P0001
            "))))||
        ((molS[i+1][j].substring(0, 7).equals("OHR1001")) &&
            (molS[i+1][j+1].substring(0, 7).equals("OHR1002
            "))))){

        }else{
            indice=0;
            aux1cc1= objAtomos.setAtomoU(molS[i][j],torS[r]);
            aux2cc1= objAtomos.setAtomoU(molS[i+1][j],torS[r])
                ;
            aux3cc1= objAtomos.setAtomoU(molS[i+1][j+1],torS[r
                ]);
            aux4cc1= objAtomos.setAtomoU(molS[i+1][j+2],torS[r
                ]);
            v1= objAtomos.getX(aux1cc1,torS[r])-objAtomos.getX
                (aux2cc1,torS[r]);
            x1= objAtomos.getY(aux1cc1,torS[r])-objAtomos.getY
                (aux2cc1,torS[r]);
            z1= objAtomos.getZ(aux1cc1,torS[r])-objAtomos.getZ
                (aux2cc1,torS[r]);
            v2= objAtomos.getX(aux3cc1,torS[r])-objAtomos.getX
                (aux2cc1,torS[r]);
```

```
x2= objAtomos.getY(aux3cc1,torS[r])-objAtomos.getY
    (aux2cc1,torS[r]);
z2= objAtomos.getZ(aux3cc1,torS[r])-objAtomos.getZ
    (aux2cc1,torS[r]);
v3= objAtomos.getX(aux2cc1,torS[r])-objAtomos.getX
    (aux3cc1,torS[r]);
x3= objAtomos.getY(aux2cc1,torS[r])-objAtomos.getY
    (aux3cc1,torS[r]);
z3= objAtomos.getZ(aux2cc1,torS[r])-objAtomos.getZ
    (aux3cc1,torS[r]);
v4= objAtomos.getX(aux4cc1,torS[r])-objAtomos.getX
    (aux3cc1,torS[r]);
x4= objAtomos.getY(aux4cc1,torS[r])-objAtomos.getY
    (aux3cc1,torS[r]);
z4= objAtomos.getZ(aux4cc1,torS[r])-objAtomos.getZ
    (aux3cc1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdcc1= (prod)/(pmod);
angdcc1= Math.acos(auxdcc1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdcc1= -angdcc1;
}
pangdcc1= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +1][j+1],molS[i+1][j+2],sem,num,indice)/getPath
```

```
            (molS[i][j],molS[i+1][j],molS[i+1][j+1],molS[i
            +1][j+2],sem,num,indice)*(1+(Math.cos((getN(
            molS[i][j],molS[i+1][j],molS[i+1][j+1],molS[i
            +1][j+2],sem,indice)*angdcc1)-getPeriU(molS[i][
            j],molS[i+1][j],molS[i+1][j+1],molS[i+1][j+2],
            sem,indice)))));
          //arredondamento TINKER
          pangdcc1= (Math.round((pangdcc1 * 10000.0))) /
              10000.0;
          pangdcc1= pangdcc1+auxcc1;
          auxcc1= pangdcc1;
          indiceD= indiceD + 1;
          ind++;
          sem++;
          while (num==0){
            //transforma pra graus o cos
            pangdcc1= (getVnU(molS[i][j],molS[i+1][j],molS[i
                +1][j+1],molS[i+1][j+2],sem,num,indice)/
                getPath(molS[i][j],molS[i+1][j],molS[i+1][j
                +1],molS[i+1][j+2],sem,num,indice)*(1+(Math.
                cos((getN(molS[i][j],molS[i+1][j],molS[i+1][j
                +1],molS[i+1][j+2],sem,indice)*angdcc1)-
                getPeriU(molS[i][j],molS[i+1][j],molS[i+1][j
                +1],molS[i+1][j+2],sem,indice)))));
            //arredondamento TINKER
            pangdcc1= (Math.round((pangdcc1 * 10000.0))) /
                10000.0;
            pangdcc1= pangdcc1+auxcc1;
            auxcc1= pangdcc1;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }//fim if
```

```
if ((((j>=4)&&(i<qntdlinha-1)&&(molS[i][j]!="") && (
    molS[i][j-3]!="") && (molS[i+1][j-3]!="") && (molS[
    i+1][j-4]!="")))&&
   ((molS[i][j-1]=="X00000001000")&&(molS[i][j-2]=="
      X00000001000")))){
   int x=j-3;
   if((molS[i][j]!="") && (molS[i][x]!="") && (molS[i
      +1][x]!="") && (molS[i+1][x-1]!="")){
     if (((((molS[i][j].substring(8, 9).equals("1"))&&(
        molS[i][x].substring(8, 9).equals("1"))))&&(
        molS[i][x].substring(9, 10).equals("1"))&&(molS
        [i+1][x].substring(9, 10).equals("1")) &&
       (molS[i+1][x].substring(8, 9).equals("1"))&&(
          molS[i+1][x-1].substring(8, 9).equals("1"))){
       indice=0;
       aux1b1= objAtomos.setAtomoU(molS[i][j],torS[r]);
       aux2b1= objAtomos.setAtomoU(molS[i][x],torS[r]);
       aux3b1= objAtomos.setAtomoU(molS[i+1][x],torS[r
          ]);
       aux4b1= objAtomos.setAtomoU(molS[i+1][x-1],torS[
          r]);
       v1= objAtomos.getX(aux1b1,torS[r])-objAtomos.
          getX(aux2b1,torS[r]);
       x1= objAtomos.getY(aux1b1,torS[r])-objAtomos.
          getY(aux2b1,torS[r]);
       z1= objAtomos.getZ(aux1b1,torS[r])-objAtomos.
          getZ(aux2b1,torS[r]);
       v2= objAtomos.getX(aux3b1,torS[r])-objAtomos.
          getX(aux2b1,torS[r]);
       x2= objAtomos.getY(aux3b1,torS[r])-objAtomos.
          getY(aux2b1,torS[r]);
       z2= objAtomos.getZ(aux3b1,torS[r])-objAtomos.
          getZ(aux2b1,torS[r]);
       v3= objAtomos.getX(aux2b1,torS[r])-objAtomos.
          getX(aux3b1,torS[r]);
```

```
x3= objAtomos.getY(aux2b1,torS[r])-objAtomos.
    getY(aux3b1,torS[r]);
z3= objAtomos.getZ(aux2b1,torS[r])-objAtomos.
    getZ(aux3b1,torS[r]);
v4= objAtomos.getX(aux4b1,torS[r])-objAtomos.
    getX(aux3b1,torS[r]);
x4= objAtomos.getY(aux4b1,torS[r])-objAtomos.
    getY(aux3b1,torS[r]);
z4= objAtomos.getZ(aux4b1,torS[r])-objAtomos.
    getZ(aux3b1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb1= (prod)/(pmod);
angdb1= Math.acos(auxdb1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb1= -angdb1;
}
pangdb1= (getVnU(molS[i][j],molS[i][x],molS[i
    +1][x],molS[i+1][x-1],sem,num,indice)/getPath
    (molS[i][j],molS[i][x],molS[i+1][x],molS[i
    +1][x-1],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i][x],molS[i+1][x],molS[i+1][
    x-1],sem,indice)*angdb1)-getPeriU(molS[i][j],
    molS[i][x],molS[i+1][x],molS[i+1][x-1],sem,
    indice)))));
```

```
            //arredondamento TINKER
            pangdb1= (Math.round((pangdb1 * 10000.0))) /
                10000.0;
            pangdb1= pangdb1+auxb1;
            auxb1= pangdb1;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdb1= (getVnU(molS[i][j],molS[i][x],molS[i
                  +1][x],molS[i+1][x-1],sem,num,indice)/
                  getPath(molS[i][j],molS[i][x],molS[i+1][x],
                  molS[i+1][x-1],sem,num,indice)*(1+(Math.cos
                  ((getN(molS[i][j],molS[i][x],molS[i+1][x],
                  molS[i+1][x-1],sem,indice)*angdb1)-getPeriU
                  (molS[i][j],molS[i][x],molS[i+1][x],molS[i
                  +1][x-1],sem,indice)))));
              //arredondamento TINKER
              pangdb1= (Math.round((pangdb1 * 10000.0))) /
                  10000.0;
              pangdb1= pangdb1+auxb1;
              auxb1= pangdb1;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }else if ((j>=2)&&(i<qntdlinha-1)&&(molS[i][j]!="") &&
          (molS[i][j-1]!="") && (molS[i+1][j-1]!="") && (
          molS[i+1][j-2]!="")&&(molS[i][j].substring(8, 9).
          equals("1"))&&(molS[i][j-1].substring(8, 9).equals
          ("1"))&&(molS[i][j-1].substring(9, 10).equals("1"))
          &&(molS[i+1][j-1].substring(9, 10).equals("1"))&&
```

```
(molS[i+1][j-1].substring(8, 9).equals("1"))&&(
    molS[i+1][j-2].substring(8, 9).equals("1"))&& (
    molS[i][j-1]!="X00000001000")){
if (((molS[i][j-1].substring(0, 7).equals("OHR2001")
)&&(molS[i+1][j-1].substring(0,7).equals("P1P0001
"))))||
((molS[i+1][j-1].substring(0, 7).equals("OHR1002"))
&&(molS[i+1][j-2].substring(0, 7).equals("OHR1001
"))))){

}else{
indice=0;
aux1b1= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2b1= objAtomos.setAtomoU(molS[i][j-1],torS[r]);
aux3b1= objAtomos.setAtomoU(molS[i+1][j-1],torS[r
]);
aux4b1= objAtomos.setAtomoU(molS[i+1][j-2],torS[r
]);
v1= objAtomos.getX(aux1b1,torS[r])-objAtomos.getX(
aux2b1,torS[r]);
x1= objAtomos.getY(aux1b1,torS[r])-objAtomos.getY(
aux2b1,torS[r]);
z1= objAtomos.getZ(aux1b1,torS[r])-objAtomos.getZ(
aux2b1,torS[r]);
v2= objAtomos.getX(aux3b1,torS[r])-objAtomos.getX(
aux2b1,torS[r]);
x2= objAtomos.getY(aux3b1,torS[r])-objAtomos.getY(
aux2b1,torS[r]);
z2= objAtomos.getZ(aux3b1,torS[r])-objAtomos.getZ(
aux2b1,torS[r]);
v3= objAtomos.getX(aux2b1,torS[r])-objAtomos.getX(
aux3b1,torS[r]);
x3= objAtomos.getY(aux2b1,torS[r])-objAtomos.getY(
aux3b1,torS[r]);
z3= objAtomos.getZ(aux2b1,torS[r])-objAtomos.getZ(
aux3b1,torS[r]);
```

```
v4= objAtomos.getX(aux4b1,torS[r])-objAtomos.getX(
    aux3b1,torS[r]);
x4= objAtomos.getY(aux4b1,torS[r])-objAtomos.getY(
    aux3b1,torS[r]);
z4= objAtomos.getZ(aux4b1,torS[r])-objAtomos.getZ(
    aux3b1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdb1= (prod)/(pmod);
angdb1= Math.acos(auxdb1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb1= -angdb1;
}
pangdb1= (getVnU(molS[i][j],molS[i][j-1],molS[i
    +1][j-1],molS[i+1][j-2],sem,num,indice)/getPath
    (molS[i][j],molS[i][j-1],molS[i+1][j-1],molS[i
    +1][j-2],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i][j-1],molS[i+1][j-1],molS[i
    +1][j-2],sem,indice)*angdb1)-getPeriU(molS[i][j
    ],molS[i][j-1],molS[i+1][j-1],molS[i+1][j-2],
    sem,indice)))));
//arredondamento TINKER
pangdb1= (Math.round((pangdb1 * 10000.0))) /
    10000.0;
pangdb1= pangdb1+auxb1;
```

```
        auxb1= pangdb1;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdb1= (getVnU(molS[i][j],molS[i][j-1],molS[i
              +1][j-1],molS[i+1][j-2],sem,num,indice)/
              getPath(molS[i][j],molS[i][j-1],molS[i+1][j
              -1],molS[i+1][j-2],sem,num,indice)*(1+(Math.
              cos((getN(molS[i][j],molS[i][j-1],molS[i+1][j
              -1],molS[i+1][j-2],sem,indice)*angdb1)-
              getPeriU(molS[i][j],molS[i][j-1],molS[i+1][j
              -1],molS[i+1][j-2],sem,indice)))));
          //arredondamento TINKER
          pangdb1= (Math.round((pangdb1 * 10000.0))) /
              10000.0;
          pangdb1= pangdb1+auxb1;
          auxb1= pangdb1;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim else
    }//fim if
    if((((j>=4)&&(i<qntdlinha-1)&&(molS[i][j]!="") && (
        molS[i][j-1]!="") && (molS[i][j-4]!="") && (molS[i
        +1][j-4]!="")))&&  ((molS[i][j-2]=="X00000001000")
        &&(molS[i][j-3]=="X00000001000"))){
      int x=j-4;
      if((molS[i][j]!="") && (molS[i][j-1]!="") && (molS[i
          ][x]!="") && (molS[i+1][x]!="")){
        if ((molS[i][j].substring(8, 9).equals("1"))&&(
            molS[i][j-1].substring(8, 9).equals("1"))&&(
            molS[i][j-1].substring(8, 9).equals("1"))&&(
            molS[i][x].substring(8, 9).equals("1"))
```

```
  &&(molS[i][x].substring(9, 10).equals("1"))&&(
      molS[i+1][x].substring(9, 10).equals("1")))
      {
indice=0;
aux1bb1= objAtomos.setAtomoU(molS[i][j],torS[r])
    ;
aux2bb1= objAtomos.setAtomoU(molS[i][j-1],torS[r
    ]);
aux3bb1= objAtomos.setAtomoU(molS[i][x],torS[r])
    ;
aux4bb1= objAtomos.setAtomoU(molS[i+1][x],torS[r
    ]);
v1= objAtomos.getX(aux1bb1,torS[r])-objAtomos.
    getX(aux2bb1,torS[r]);
x1= objAtomos.getY(aux1bb1,torS[r])-objAtomos.
    getY(aux2bb1,torS[r]);
z1= objAtomos.getZ(aux1bb1,torS[r])-objAtomos.
    getZ(aux2bb1,torS[r]);
v2= objAtomos.getX(aux3bb1,torS[r])-objAtomos.
    getX(aux2bb1,torS[r]);
x2= objAtomos.getY(aux3bb1,torS[r])-objAtomos.
    getY(aux2bb1,torS[r]);
z2= objAtomos.getZ(aux3bb1,torS[r])-objAtomos.
    getZ(aux2bb1,torS[r]);
v3= objAtomos.getX(aux2bb1,torS[r])-objAtomos.
    getX(aux3bb1,torS[r]);
x3= objAtomos.getY(aux2bb1,torS[r])-objAtomos.
    getY(aux3bb1,torS[r]);
z3= objAtomos.getZ(aux2bb1,torS[r])-objAtomos.
    getZ(aux3bb1,torS[r]);
v4= objAtomos.getX(aux4bb1,torS[r])-objAtomos.
    getX(aux3bb1,torS[r]);
x4= objAtomos.getY(aux4bb1,torS[r])-objAtomos.
    getY(aux3bb1,torS[r]);
z4= objAtomos.getZ(aux4bb1,torS[r])-objAtomos.
    getZ(aux3bb1,torS[r]);
```

```
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb1= (prod)/(pmod);
angdbb1= Math.acos(auxdbb1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb1= -angdbb1;
}
pangdbb1= (getVnU(molS[i][j],molS[i][j-1],molS[i
    ][x],molS[i+1][x],sem,num,indice)/getPath(
    molS[i][j],molS[i][j-1],molS[i][x],molS[i+1][
    x],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i][j-1],molS[i][x],molS[i+1][x],sem
    ,indice)*angdbb1)-getPeriU(molS[i][j],molS[i
    ][j-1],molS[i][x],molS[i+1][x],sem,indice))))
    );
//arredondamento TINKER
pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
    10000.0;
pangdbb1= pangdbb1+auxbb1;
auxbb1= pangdbb1;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
```

```
            pangdbb1= (getVnU(molS[i][j],molS[i][j-1],molS
                [i][x],molS[i+1][x],sem,num,indice)/getPath
                (molS[i][j],molS[i][j-1],molS[i][x],molS[i
                +1][x],sem,num,indice)*(1+(Math.cos((getN(
                molS[i][j],molS[i][j-1],molS[i][x],molS[i
                +1][x],sem,indice)*angdbb1)-getPeriU(molS[i
                ][j],molS[i][j-1],molS[i][x],molS[i+1][x],
                sem,indice)))));
            //arredondamento TINKER
            pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
                 10000.0;
            pangdbb1= pangdbb1+auxbb1;
            auxbb1= pangdbb1;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }
    }else if ((j>=2)&&(i<qntdlinha-1)&&(molS[i][j]!="") &&
        (molS[i][j-1]!="") && (molS[i][j-2]!="") && (molS[
       i+1][j-2]!="")
        &&(molS[i][j].substring(8, 9).equals("1"))&&(molS[
            i][j-1].substring(8, 9).equals("1"))&&(molS[i][
            j-1].substring(8, 9).equals("1"))&&(molS[i][j
            -2].substring(8, 9).equals("1"))
        &&(molS[i][j-2].substring(9, 10).equals("1"))&&(
            molS[i+1][j-2].substring(9, 10).equals("1"))&&
            ((molS[i][j-1]!="X00000001000")&&(molS[i][j
            -2]!="X00000001000"))){
        indice=0;
        aux1bb1= objAtomos.setAtomoU(molS[i][j],torS[r]);
        aux2bb1= objAtomos.setAtomoU(molS[i][j-1],torS[r])
            ;
        aux3bb1= objAtomos.setAtomoU(molS[i][j-2],torS[r])
            ;
```

```
aux4bb1= objAtomos.setAtomoU(molS[i+1][j-2],torS[r
    ]);
v1= objAtomos.getX(aux1bb1,torS[r])-objAtomos.getX
    (aux2bb1,torS[r]);
x1= objAtomos.getY(aux1bb1,torS[r])-objAtomos.getY
    (aux2bb1,torS[r]);
z1= objAtomos.getZ(aux1bb1,torS[r])-objAtomos.getZ
    (aux2bb1,torS[r]);
v2= objAtomos.getX(aux3bb1,torS[r])-objAtomos.getX
    (aux2bb1,torS[r]);
x2= objAtomos.getY(aux3bb1,torS[r])-objAtomos.getY
    (aux2bb1,torS[r]);
z2= objAtomos.getZ(aux3bb1,torS[r])-objAtomos.getZ
    (aux2bb1,torS[r]);
v3= objAtomos.getX(aux2bb1,torS[r])-objAtomos.getX
    (aux3bb1,torS[r]);
x3= objAtomos.getY(aux2bb1,torS[r])-objAtomos.getY
    (aux3bb1,torS[r]);
z3= objAtomos.getZ(aux2bb1,torS[r])-objAtomos.getZ
    (aux3bb1,torS[r]);
v4= objAtomos.getX(aux4bb1,torS[r])-objAtomos.getX
    (aux3bb1,torS[r]);
x4= objAtomos.getY(aux4bb1,torS[r])-objAtomos.getY
    (aux3bb1,torS[r]);
z4= objAtomos.getZ(aux4bb1,torS[r])-objAtomos.getZ
    (aux3bb1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
```

```
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdbb1= (prod)/(pmod);
//angdl= java.lang.Math.cos(aux/(180*Math.PI));
angdbb1= Math.acos(auxdbb1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb1= -angdbb1;
}
pangdbb1= (getVnU(molS[i][j],molS[i][j-1],molS[i][
    j-2],molS[i+1][j-2],sem,num,indice)/getPath(
    molS[i][j],molS[i][j-1],molS[i][j-2],molS[i+1][
    j-2],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i][j-1],molS[i][j-2],molS[i+1][j-2],
    sem,indice)*angdbb1)-getPeriU(molS[i][j],molS[i
    ][j-1],molS[i][j-2],molS[i+1][j-2],sem,indice))
    ))));
//arredondamento TINKER
pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
    10000.0;
pangdbb1= pangdbb1+auxbb1;
auxbb1= pangdbb1;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdbb1= (getVnU(molS[i][j],molS[i][j-1],molS[i
      ][j-2],molS[i+1][j-2],sem,num,indice)/getPath
      (molS[i][j],molS[i][j-1],molS[i][j-2],molS[i
      +1][j-2],sem,num,indice)*(1+(Math.cos((getN(
      molS[i][j],molS[i][j-1],molS[i][j-2],molS[i
      +1][j-2],sem,indice)*angdbb1)-getPeriU(molS[i
      ][j],molS[i][j-1],molS[i][j-2],molS[i+1][j
      -2],sem,indice)))));
```

```
                //arredondamento TINKER
                pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
                    10000.0;
                pangdbb1= pangdbb1+auxbb1;
                auxbb1= pangdbb1;
                indiceD= indiceD + 1;
                ind++;
                sem++;
              }
          }//fim if
        }//fim else
    }//fim for
    }//fim for
    pangd[r]= pangdebb3+pangdeb3+pangdbbe+pangdbe+pangdd+
        pangdb3+ pangdbb3+pangdl+pangdll+ pangdc+pangdcc+
        pangdb+pangdbb+ pangdc1+pangdcc1+ pangdb1+ pangdbb1+
        pangdc2+ pangdcc2+pangdbb2+pangdb2;
    }//fim for r

    return pangd;

}//fim angulo diedral

//excedeu o limite de linhas do metodo
public double[] angd2U (int semente){
    double[] pangd= new double[torS.length];
    double pangdebb3,pangdeb3,pangdbbe,pangdbe,pangdd,pangdb3,
        pangdbb3,pangdl,pangdll, pangdc,pangdcc, pangdb,
        pangdbb, pangdc1,pangdcc1, pangdb1, pangdbb1,pangdc2,
    pangdcc2,pangdbb2,pangddir,pangddir1,pangdb2;
    double  angdl,angdll,angdeb3,angdbe,angdbbe,angdb3,angdbb3
        ,angdb2,angdbb2, angdc1,angdcc1,angdcc2,angdc2;
    double auxdebb2,auxdeb3, auxdebb3,auxdbe,auxdbbe,auxdb3,
        auxdbb3,auxdl,auxdll,auxdb,auxdbb,auxdc,auxdcc,auxdb1,
        auxdbb1,auxdb2,auxdbb2,auxdc1,auxdcc1,auxdcc2,auxdc2;
```

```
double auxeb3,auxebb3,auxbe,auxbbe,auxb3,auxbb3,auxl,auxll
    ,auxb,auxbb,auxc,auxcc,auxb1,auxbb1,auxb2,auxbb2,auxc1,
    auxcc1,auxcc2,auxc2;
int indiceD= 0;
double a1, a2, b1, b2, c1, c2, v1, x1, z1, v2, x2, z2, v3,
     x3, z3, v4, x4, z4, x1s, y1s, z1s;//, x2s, y2s, z2s;
double aux = 0.0;
double[] aux1ebb2, aux2ebb2,aux3ebb2,aux4ebb2,aux1eb3,
    aux2eb3,aux3eb3,aux4eb3,aux1ebb3,aux2ebb3,aux3ebb3,
    aux4ebb3,aux1be, aux2be,aux3be,aux4be,
aux1l, aux2l, aux3l, aux4l, aux1ll, aux2ll, aux3ll, aux4ll
    , aux1b, aux2b, aux3b, aux4b, aux1bb, aux2bb, aux3bb,
    aux4bb, aux1cc, aux2cc, aux3cc, aux4cc, aux1c, aux2c,
    aux3c, aux4c,
aux1c1, aux2c1, aux3c1, aux4c1, aux1cc1, aux2cc1, aux3cc1,
     aux4cc1, aux1b1, aux2b1, aux3b1, aux4b1,aux1bb1,
    aux2bb1, aux3bb1, aux4bb1, aux1c2, aux2c2, aux3c2,
    aux4c2,
aux1cc2, aux2cc2, aux3cc2, aux4cc2, aux1b3, aux2b3, aux3b3
    , aux4b3, aux1bb3, aux2bb3, aux3bb3, aux4bb3, aux1bb2,
    aux2bb2, aux4bb2, aux3bb2, aux1b2, aux2b2, aux4b2,
    aux3b2 = new double[3];
double pmod, mod, mod2, prod= 0.0;
int ind= 0;
double auxdir=0;
num = 1;
indice = 0;
String log2="";
sem= semente;

for(int r= 0; r< torS.length; r++){
  aux1eb3= new double[3];
  aux2eb3= new double[3];
  aux3eb3= new double[3];
  aux4eb3= new double[3];
  aux1ebb3= new double[3];
```

```
aux2ebb3= new double[3];
aux3ebb3= new double[3];
aux4ebb3= new double[3];
aux1be= new double[3];
aux2be= new double[3];
aux3be= new double[3];
aux4be= new double[3];
aux1l= new double[3];
aux2l= new double[3];
aux3l= new double[3];
aux4l= new double[3];
aux1ll= new double[3];
aux2ll= new double[3];
aux3ll= new double[3];
aux4ll= new double[3];
aux1b= new double[3];
aux2b= new double[3];
aux3b= new double[3];
aux4b= new double[3];
aux1bb= new double[3];
aux2bb= new double[3];
aux3bb= new double[3];
aux4bb= new double[3];
aux1c= new double[3];
aux2c= new double[3];
aux3c= new double[3];
aux4c= new double[3];
aux1cc= new double[3];
aux2cc= new double[3];
aux3cc= new double[3];
aux4cc= new double[3];
aux1c1= new double[3];
aux2c1= new double[3];
aux3c1= new double[3];
aux4c1= new double[3];
aux1cc1= new double[3];
```

```
aux2cc1= new double[3];
aux3cc1= new double[3];
aux4cc1= new double[3];
aux1b1= new double[3];
aux2b1= new double[3];
aux3b1= new double[3];
aux4b1= new double[3];
aux1bb1= new double[3];
aux2bb1= new double[3];
aux3bb1= new double[3];
aux4bb1= new double[3];
aux1b3= new double[3];
aux2b3= new double[3];
aux3b3= new double[3];
aux4b3= new double[3];
aux1bb3= new double[3];
aux2bb3= new double[3];
aux3bb3= new double[3];
aux4bb3= new double[3];
aux1c2= new double[3];
aux2c2= new double[3];
aux3c2= new double[3];
aux4c2= new double[3];
aux1cc2= new double[3];
aux2cc2= new double[3];
aux3cc2= new double[3];
aux4cc2= new double[3];
aux1b2= new double[3];
aux2b2= new double[3];
aux4b2= new double[3];
aux1bb2= new double[3];
aux2bb2= new double[3];
aux3bb2= new double[3];
aux4bb2= new double[3];
double angdbb= 0.0;
x1s=0.0;
```

```
y1s=0.0;
z1s=0.0;
v1= 0.0;
v2= 0.0;
x1= 0.0;
x2= 0.0;
z1= 0.0;
z2= 0.0;
v3= 0.0;
v4= 0.0;
x3= 0.0;
x4= 0.0;
z3= 0.0;
z4= 0.0;
mod= 0.0;
mod2= 0.0;
pmod = 0.0;
prod = 0.0;
pangddir1=0;
pangddir=0;
pangdebb3=0.0;
pangdeb3=0.0;
pangdbbe=0.0;
pangdbe= 0.0;
pangdd=0.0;
pangdb3=0.0;
pangdbb3=0.0;
pangdbb2=0.0;
pangdl=0.0;
pangdll=0.0;
pangdc=0.0;
pangdcc=0.0;
pangdb=0.0;
pangdbb=0.0;
pangdc1=0.0;
pangdcc1=0.0;
```

```
pangdb1=0.0;
pangdbb1=0.0;
pangdc2=0.0;
pangdcc2=0.0;
pangdb2= 0.0;
auxdeb3=0.0;
auxdebb3=0.0;
auxdebb2=0.0;
auxdbe=0.0;
auxdbbe=0.0;
auxeb3=0.0;
auxebb3=0.0;
auxdebb2=0.0;
auxbe=0.0;
auxbbe=0.0;
angdeb3=0.0;
angdb3=0.0;
angdbb3=0.0;
angdbe= 0;
angdbbe= 0;
angdc1=0.0;
angdcc1=0.0;
angdc2=0.0;
angdcc2=0.0;
angdb2=0.0;
auxdl=0.0;
auxdll=0.0;
auxdb3=0.0;
auxdbb3=0.0;
auxdb=0.0;
auxdbb=0.0;
auxdc=0.0;
auxdcc=0.0;
auxdc1=0.0;
auxdcc1=0.0;
auxdc2=0.0;
```

```
    auxdcc2=0.0;
    auxdb1=0.0;
    auxdbb1=0.0;
    auxdb2=0.0;
    auxl=0.0;
    auxll=0.0;
    auxb3=0.0;
    auxbb3=0.0;
    auxb=0.0;
    auxbb=0.0;
    auxc=0.0;
    auxcc=0.0;
    auxc1=0.0;
    auxcc1=0.0;
    auxc2=0.0;
    auxcc2=0.0;
    auxb1=0.0;
    auxbb1=0.0;
    auxb2=0.0;
    auxbb2=0.0;
    a1=0.0;
    a2=0.0;
    b1=0.0;
    b2=0.0;
    c1=0.0;
  for (int i=0; i<qntdlinha; i++){
  for (int j= 0; j< qntdcol; j++){
    aux=0.0;
    if (molS[i][j].equals("X00000001000")){
      //nao quero que comece com X
    }else{
      if((i<qntdlinha-3)&&(molS[i][j]!="") &&(j!=0)&& (molS[
          i+1][j]!="") && (molS[i+2][j]!="") && (molS[i+3][j
          ]!="")){
        //verificar se todo esta ligados
```

```
if ((molS[i][j].substring(9, 10).equals("1"))&&(molS
    [i+1][j].substring(9, 10).equals("1"))&&(molS[i
    +1][j].substring(9, 10).equals("1"))&&(molS[i+2][
    j].substring(9, 10).equals("1")&&
     (molS[i+2][j].substring(9, 10).equals("1"))&&(
        molS[i+3][j].substring(9, 10).equals("1")))){
  if (((molS[i+2][j].substring(0, 7).equals("OHR2001
     "))&&(molS[i+3][j].substring(0, 7).equals("
     P1P0001")))||
  ((molS[i+1][j].substring(0, 7).equals("OHR2001"))
     &&(molS[i+2][j].substring(0, 7).equals("P1P0001
     "))))){

  }else{
    indice=0;
    aux1b2= objAtomos.setAtomoU(molS[i][j],torS[r]);
    aux2b2= objAtomos.setAtomoU(molS[i+1][j],torS[r
       ]);
    aux3b2= objAtomos.setAtomoU(molS[i+2][j],torS[r
       ]);
    aux4b2= objAtomos.setAtomoU(molS[i+3][j],torS[r
       ]);
    v1= objAtomos.getX(aux1b2,torS[r])-objAtomos.
       getX(aux2b2,torS[r]);
    x1= objAtomos.getY(aux1b2,torS[r])-objAtomos.
       getY(aux2b2,torS[r]);
    z1= objAtomos.getZ(aux1b2,torS[r])-objAtomos.
       getZ(aux2b2,torS[r]);
    v2= objAtomos.getX(aux3b2,torS[r])-objAtomos.
       getX(aux2b2,torS[r]);
    x2= objAtomos.getY(aux3b2,torS[r])-objAtomos.
       getY(aux2b2,torS[r]);
    z2= objAtomos.getZ(aux3b2,torS[r])-objAtomos.
       getZ(aux2b2,torS[r]);
    v3= objAtomos.getX(aux2b2,torS[r])-objAtomos.
       getX(aux3b2,torS[r]);
```

```
x3= objAtomos.getY(aux2b2,torS[r])-objAtomos.
    getY(aux3b2,torS[r]);
z3= objAtomos.getZ(aux2b2,torS[r])-objAtomos.
    getZ(aux3b2,torS[r]);
v4= objAtomos.getX(aux4b2,torS[r])-objAtomos.
    getX(aux3b2,torS[r]);
x4= objAtomos.getY(aux4b2,torS[r])-objAtomos.
    getY(aux3b2,torS[r]);
z4= objAtomos.getZ(aux4b2,torS[r])-objAtomos.
    getZ(aux3b2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb2= (prod)/(pmod);
angdb2= Math.acos(auxdb2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb2= -angdb2;
}
pangdb2= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j],molS[i+3][j],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j],molS[i+2][j],molS[i
    +3][j],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j],molS[i+2][j],molS[i
    +3][j],sem,indice)*angdb2)-getPeriU(molS[i][j
    ],molS[i+1][j],molS[i+2][j],molS[i+3][j],sem,
    indice)))));
```

```
            //arredondamento TINKER
            pangdb2= (Math.round((pangdb2 * 10000.0))) /
                10000.0;
            pangdb2= pangdb2+auxb2;
            auxb2= pangdb2;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdb2= (getVnU(molS[i][j],molS[i+1][j],molS[
                  i+2][j],molS[i+3][j],sem,num,indice)/
                  getPath(molS[i][j],molS[i+1][j],molS[i+2][j
                  ],molS[i+3][j],sem,num,indice)*(1+(Math.cos
                  ((getN(molS[i][j],molS[i+1][j],molS[i+2][j
                  ],molS[i+3][j],sem,indice)*angdb2)-getPeriU
                  (molS[i][j],molS[i+1][j],molS[i+2][j],molS[
                  i+3][j],sem,indice)))));
              //arredondamento TINKER
              pangdb2= (Math.round((pangdb2 * 10000.0))) /
                  10000.0;
              pangdb2= pangdb2+auxb2;
              auxb2= pangdb2;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }//fim else
      }//fim if
      if (((j<qntdcol-5)&&(molS[i][j]!="") && (i!=0)&&(molS[
          i][j+3]!="") && (molS[i-1][j+4]!="") && (molS[i][j
          +5]!=""))&&
        (molS[i][j+1]=="X00000001000")&&(molS[i][j+2]==""
            X00000001000")){
          int x=j+3;
```

```
if((j<qntdcol-4)&&(molS[i][j]!="") && (molS[i][x
    ]!="") && (molS[i-1][x+1]!="") && (molS[i][x
    +2]!="")){
  if ((molS[i][j].substring(8, 9).equals("1"))&&(
      molS[i][x].substring(8, 9).equals("1")) && (
      molS[i][x].substring(10, 11).equals("1"))&&(
      molS[i-1][x+1].substring(11, 12).equals("1"))
      &&
        (molS[i-1][x+1].substring(10, 11).equals
            ("1"))&&(molS[i][x+2].substring(11, 12).
            equals("1"))){
    indice=0;
    aux1be= objAtomos.setAtomoU(molS[i][j],torS[r
        ]);
    aux2be= objAtomos.setAtomoU(molS[i][x],torS[r
        ]);
    aux3be= objAtomos.setAtomoU(molS[i-1][x+1],
        torS[r]);
    aux4be= objAtomos.setAtomoU(molS[i][x+2],torS[
        r]);
    v1= objAtomos.getX(aux1be,torS[r])-objAtomos.
        getX(aux2be,torS[r]);
    x1= objAtomos.getY(aux1be,torS[r])-objAtomos.
        getY(aux2be,torS[r]);
    z1= objAtomos.getZ(aux1be,torS[r])-objAtomos.
        getZ(aux2be,torS[r]);
    v2= objAtomos.getX(aux3be,torS[r])-objAtomos.
        getX(aux2be,torS[r]);
    x2= objAtomos.getY(aux3be,torS[r])-objAtomos.
        getY(aux2be,torS[r]);
    z2= objAtomos.getZ(aux3be,torS[r])-objAtomos.
        getZ(aux2be,torS[r]);
    v3= objAtomos.getX(aux2be,torS[r])-objAtomos.
        getX(aux3be,torS[r]);
    x3= objAtomos.getY(aux2be,torS[r])-objAtomos.
        getY(aux3be,torS[r]);
```

```
z3= objAtomos.getZ(aux2be,torS[r])-objAtomos.
    getZ(aux3be,torS[r]);
v4= objAtomos.getX(aux4be,torS[r])-objAtomos.
    getX(aux3be,torS[r]);
x4= objAtomos.getY(aux4be,torS[r])-objAtomos.
    getY(aux3be,torS[r]);
z4= objAtomos.getZ(aux4be,torS[r])-objAtomos.
    getZ(aux3be,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbe= (prod)/(pmod);
angdbe= Math.acos(auxdbe);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbe= -angdbe;
}
pangdbe= (getVnU(molS[i][j],molS[i][x],molS[i
    -1][x+1],molS[i][x+2],sem,num,indice)/
    getPath(molS[i][j],molS[i][x],molS[i-1][x
    +1],molS[i][x+2],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i][x],molS[i-1][x
    +1],molS[i][x+2],sem,indice)*angdbe)-
    getPeriU(molS[i][j],molS[i][x],molS[i-1][x
    +1],molS[i][x+2],sem,indice)))));
//arredondamento TINKER
```

```
                pangdbe= (Math.round((pangdbe * 10000.0))) /
                    10000.0;
                pangdbe= pangdbe+auxbe;
                auxbe= pangdbe;
                indiceD= indiceD + 1;
                ind++;
                sem++;
                while (num==0){
                  //transforma pra graus o cos
                  pangdbe= (getVnU(molS[i][j],molS[i][x],molS[
                      i-1][x+1],molS[i][x+2],sem,num,indice)/
                      getPath(molS[i][j],molS[i][x],molS[i-1][x
                      +1],molS[i][x+2],sem,num,indice)*(1+(Math
                      .cos((getN(molS[i][j],molS[i][x],molS[i
                      -1][x+1],molS[i][x+2],sem,indice)*angdbe)
                      -getPeriU(molS[i][j],molS[i][x],molS[i
                      -1][x+1],molS[i][x+2],sem,indice)))));
                  //arredondamento TINKER
                  pangdbe= (Math.round((pangdbe * 10000.0))) /
                      10000.0;
                  pangdbe= pangdbe+auxbe;
                  auxbe= pangdbe;
                  indiceD= indiceD + 1;
                  ind++;
                  sem++;
                }
              }
            }
          }else if (((j<qntdcol-3)&&(molS[i][j]!="") && (i!=0)
              &&(molS[i][j+1]!="") && (molS[i-1][j+2]!="") && (
              molS[i][j+3]!=""))&&(j<qntdcol-3)&&(molS[i][j]!="")
               && (i!=0)&&(molS[i][j+1]!="") && (molS[i-1][j
              +2]!="") && (molS[i][j+3]!="")&&(molS[i][j].
              substring(8, 9).equals("1"))&&(molS[i][j+1].
              substring(8, 9).equals("1")) && (molS[i][j+1].
              substring(10, 11).equals("1"))&&(molS[i-1][j+2].
```

```
substring(11, 12).equals("1"))&&
    (molS[i-1][j+2].substring(10, 11).equals("1"))
        &&(molS[i][j+3].substring(11, 12).equals("1")
        )&&(molS[i][j+1]!="X00000001000")){
 indice=0;
 aux1be= objAtomos.setAtomoU(molS[i][j],torS[r]);
 aux2be= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
 aux3be= objAtomos.setAtomoU(molS[i-1][j+2],torS[r
    ]);
 aux4be= objAtomos.setAtomoU(molS[i][j+3],torS[r]);
 v1= objAtomos.getX(aux1be,torS[r])-objAtomos.getX(
    aux2be,torS[r]);
 x1= objAtomos.getY(aux1be,torS[r])-objAtomos.getY(
    aux2be,torS[r]);
 z1= objAtomos.getZ(aux1be,torS[r])-objAtomos.getZ(
    aux2be,torS[r]);
 v2= objAtomos.getX(aux3be,torS[r])-objAtomos.getX(
    aux2be,torS[r]);
 x2= objAtomos.getY(aux3be,torS[r])-objAtomos.getY(
    aux2be,torS[r]);
 z2= objAtomos.getZ(aux3be,torS[r])-objAtomos.getZ(
    aux2be,torS[r]);
 v3= objAtomos.getX(aux2be,torS[r])-objAtomos.getX(
    aux3be,torS[r]);
 x3= objAtomos.getY(aux2be,torS[r])-objAtomos.getY(
    aux3be,torS[r]);
 z3= objAtomos.getZ(aux2be,torS[r])-objAtomos.getZ(
    aux3be,torS[r]);
 v4= objAtomos.getX(aux4be,torS[r])-objAtomos.getX(
    aux3be,torS[r]);
 x4= objAtomos.getY(aux4be,torS[r])-objAtomos.getY(
    aux3be,torS[r]);
 z4= objAtomos.getZ(aux4be,torS[r])-objAtomos.getZ(
    aux3be,torS[r]);
 a1= ((x1*z2)-(x2*z1));
 b1= ((z1*v2)-(v1*z2));
```

```
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdbe= (prod)/(pmod);
angdbe= Math.acos(auxdbe);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbe= -angdbe;
}
pangdbe= (getVnU(molS[i][j],molS[i][j+1],molS[i
    -1][j+2],molS[i][j+3],sem,num,indice)/getPath(
    molS[i][j],molS[i][j+1],molS[i-1][j+2],molS[i][
    j+3],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i][j+1],molS[i-1][j+2],molS[i][j+3],
    sem,indice)*angdbe)-getPeriU(molS[i][j],molS[i
    ][j+1],molS[i-1][j+2],molS[i][j+3],sem,indice))
    )));
//arredondamento TINKER
pangdbe= (Math.round((pangdbe * 10000.0))) /
    10000.0;
pangdbe= pangdbe+auxbe;
auxbe= pangdbe;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdbe= (getVnU(molS[i][j],molS[i][j+1],molS[i
      -1][j+2],molS[i][j+3],sem,num,indice)/getPath
```

```
            (molS[i][j],molS[i][j+1],molS[i-1][j+2],molS[
            i][j+3],sem,num,indice)*(1+(Math.cos((getN(
            molS[i][j],molS[i][j+1],molS[i-1][j+2],molS[i
            ][j+3],sem,indice)*angdbe)-getPeriU(molS[i][j
            ],molS[i][j+1],molS[i-1][j+2],molS[i][j+3],
            sem,indice)))));
          //arredondamento TINKER
          pangdbe= (Math.round((pangdbe * 10000.0))) /
              10000.0;
          pangdbe= pangdbe+auxbe;
          auxbe= pangdbe;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim if
      if(((i<qntdlinha-1)&&(j<qntdcol-5)&&(molS[i][j]!="")
         && (molS[i][j+3]!="") && (molS[i][j+4]!="") && (
         molS[i+1][j+5]!=""))&&
         ((molS[i][j+1]=="X00000001000")&&(molS[i][j+2]=="
            X00000001000"))){
        int x=j+3;
        if((j<qntdcol-4)&&(molS[i][j]!="") && (molS[i][x
            ]!="") && (molS[i][x+1]!="") && (molS[i+1][x
            +2]!="")){
          if ((molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][x].substring(8, 9).equals("1")) && (
              molS[i][x].substring(8, 9).equals("1"))&&(molS[
              i][x+1].substring(8, 9).equals("1"))&&
             (molS[i][x+1].substring(10, 11).equals("1"))&&(
                molS[i+1][x+2].substring(11, 12).equals("1"))
                ){
          indice=0;
          aux1cc2= objAtomos.setAtomoU(molS[i][j],torS[r])
              ;
```

```
aux2cc2= objAtomos.setAtomoU(molS[i][x],torS[r])
    ;
aux3cc2= objAtomos.setAtomoU(molS[i][x+1],torS[r
    ]);
aux4cc2= objAtomos.setAtomoU(molS[i+1][x+2],torS
    [r]);
v1= objAtomos.getX(aux1cc2,torS[r])-objAtomos.
    getX(aux2cc2,torS[r]);
x1= objAtomos.getY(aux1cc2,torS[r])-objAtomos.
    getY(aux2cc2,torS[r]);
z1= objAtomos.getZ(aux1cc2,torS[r])-objAtomos.
    getZ(aux2cc2,torS[r]);
v2= objAtomos.getX(aux3cc2,torS[r])-objAtomos.
    getX(aux2cc2,torS[r]);
x2= objAtomos.getY(aux3cc2,torS[r])-objAtomos.
    getY(aux2cc2,torS[r]);
z2= objAtomos.getZ(aux3cc2,torS[r])-objAtomos.
    getZ(aux2cc2,torS[r]);
v3= objAtomos.getX(aux2cc2,torS[r])-objAtomos.
    getX(aux3cc2,torS[r]);
x3= objAtomos.getY(aux2cc2,torS[r])-objAtomos.
    getY(aux3cc2,torS[r]);
z3= objAtomos.getZ(aux2cc2,torS[r])-objAtomos.
    getZ(aux3cc2,torS[r]);
v4= objAtomos.getX(aux4cc2,torS[r])-objAtomos.
    getX(aux3cc2,torS[r]);
x4= objAtomos.getY(aux4cc2,torS[r])-objAtomos.
    getY(aux3cc2,torS[r]);
z4= objAtomos.getZ(aux4cc2,torS[r])-objAtomos.
    getZ(aux3cc2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
```

```
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdcc2= (prod)/(pmod);
angdcc2= Math.acos(auxdcc2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdcc2= -angdcc2;
}
pangdcc2= (getVnU(molS[i][j],molS[i][x],molS[i][
    x+1],molS[i+1][x+2],sem,num,indice)/getPath(
    molS[i][j],molS[i][x],molS[i][x+1],molS[i+1][
    x+2],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i][x],molS[i][x+1],molS[i+1][x+2],
    sem,indice)*angdcc2)-getPeriU(molS[i][j],molS
    [i][x],molS[i][x+1],molS[i+1][x+2],sem,indice
    )))));
//arredondamento TINKER
pangdcc2= (Math.round((pangdcc2 * 10000.0))) /
    10000.0;
pangdcc2= pangdcc2+auxcc2;
auxcc2= pangdcc2;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
//transforma pra graus o cos
  pangdcc2= (getVnU(molS[i][j],molS[i][x],molS[i
      ][x+1],molS[i+1][x+2],sem,num,indice)/
      getPath(molS[i][j],molS[i][x],molS[i][x+1],
      molS[i+1][x+2],sem,num,indice)*(1+(Math.cos
      ((getN(molS[i][j],molS[i][x],molS[i][x+1],
      molS[i+1][x+2],sem,indice)*angdcc2)-
```

```
                getPeriU(molS[i][j],molS[i][x],molS[i][x
                +1],molS[i+1][x+2],sem,indice)))));
            //arredondamento TINKER
            pangdcc2= (Math.round((pangdcc2 * 10000.0))) /
                 10000.0;
            pangdcc2= pangdcc2+auxcc2;
            auxcc2= pangdcc2;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }
    }else if (((i<qntdlinha-1)&&(j<qntdcol-3)&&(molS[i][j
       ]!="") && (molS[i][j+1]!="") && (molS[i][j+2]!="")
       && (molS[i+1][j+3]!=""))
      &&(((molS[i][j].substring(8, 9).equals("1"))&&(molS[
         i][j+1].substring(8, 9).equals("1")))) && (molS[i
         ][j+1].substring(8, 9).equals("1"))&&(molS[i][j
         +2].substring(8, 9).equals("1"))&&
      (molS[i][j+2].substring(10, 11).equals("1"))&&(molS[
         i+1][j+3].substring(11, 12).equals("1"))&&(molS[i
         ][j+1]!="X00000001000")){
      indice=0;
      aux1cc2= objAtomos.setAtomoU(molS[i][j],torS[r]);
      aux2cc2= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
      aux3cc2= objAtomos.setAtomoU(molS[i][j+2],torS[r]);
      aux4cc2= objAtomos.setAtomoU(molS[i+1][j+3],torS[r])
         ;
      v1= objAtomos.getX(aux1cc2,torS[r])-objAtomos.getX(
         aux2cc2,torS[r]);
      x1= objAtomos.getY(aux1cc2,torS[r])-objAtomos.getY(
         aux2cc2,torS[r]);
      z1= objAtomos.getZ(aux1cc2,torS[r])-objAtomos.getZ(
         aux2cc2,torS[r]);
```

```
v2= objAtomos.getX(aux3cc2,torS[r])-objAtomos.getX(
    aux2cc2,torS[r]);
x2= objAtomos.getY(aux3cc2,torS[r])-objAtomos.getY(
    aux2cc2,torS[r]);
z2= objAtomos.getZ(aux3cc2,torS[r])-objAtomos.getZ(
    aux2cc2,torS[r]);
v3= objAtomos.getX(aux2cc2,torS[r])-objAtomos.getX(
    aux3cc2,torS[r]);
x3= objAtomos.getY(aux2cc2,torS[r])-objAtomos.getY(
    aux3cc2,torS[r]);
z3= objAtomos.getZ(aux2cc2,torS[r])-objAtomos.getZ(
    aux3cc2,torS[r]);
v4= objAtomos.getX(aux4cc2,torS[r])-objAtomos.getX(
    aux3cc2,torS[r]);
x4= objAtomos.getY(aux4cc2,torS[r])-objAtomos.getY(
    aux3cc2,torS[r]);
z4= objAtomos.getZ(aux4cc2,torS[r])-objAtomos.getZ(
    aux3cc2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
auxdcc2= (prod)/(pmod);
angdcc2= Math.acos(auxdcc2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdcc2= -angdcc2;
}
```

```
        pangdcc2= (getVnU(molS[i][j],molS[i][j+1],molS[i][j
            +2],molS[i+1][j+3],sem,num,indice)/getPath(molS[i
            ][j],molS[i][j+1],molS[i][j+2],molS[i+1][j+3],sem
            ,num,indice)*(1+(Math.cos((getN(molS[i][j],molS[i
            ][j+1],molS[i][j+2],molS[i+1][j+3],sem,indice)*
            angdcc2)-getPeriU(molS[i][j],molS[i][j+1],molS[i
            ][j+2],molS[i+1][j+3],sem,indice)))));
        //arredondamento TINKER
        pangdcc2= (Math.round((pangdcc2 * 10000.0))) /
            10000.0;
        pangdcc2= pangdcc2+auxcc2;
        auxcc2= pangdcc2;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdcc2= (getVnU(molS[i][j],molS[i][j+1],molS[i][
              j+2],molS[i+1][j+3],sem,num,indice)/getPath(
              molS[i][j],molS[i][j+1],molS[i][j+2],molS[i+1][
              j+3],sem,num,indice)*(1+(Math.cos((getN(molS[i
              ][j],molS[i][j+1],molS[i][j+2],molS[i+1][j+3],
              sem,indice)*angdcc2)-getPeriU(molS[i][j],molS[i
              ][j+1],molS[i][j+2],molS[i+1][j+3],sem,indice))
              ))));
          //arredondamento TINKER
          pangdcc2= (Math.round((pangdcc2 * 10000.0))) /
              10000.0;
          pangdcc2= pangdcc2+auxcc2;
          auxcc2= pangdcc2;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim if
```

```
if (((i!=0)&&(j<qntdcol-5)&&(molS[i][j]!="") && (molS[i
    ][j+3]!="") && (molS[i][j+4]!="") && (molS[i-1][j
    +5]!=""))&&
   ((molS[i][j+1]=="X00000001000")&&(molS[i][j+2]=="
      X00000001000"))){
     int x=j+3;
     if((i!=0)&&(j<qntdcol-5)&&(molS[i][j]!="") && (
        molS[i][x]!="") && (molS[i][x+1]!="") && (molS[
        i-1][x+2]!="")){
       if ((molS[i][j].substring(8, 9).equals("1"))&&(
          molS[i][x].substring(8, 9).equals("1")) && (
          molS[i][x].substring(8, 9).equals("1"))&&(
          molS[i][x+1].substring(8, 9).equals("1"))&&
            (molS[i][x+1].substring(10, 11).equals("1"))
              &&(molS[i-1][x+2].substring(11, 12).
                 equals("1"))){
         indice=0;
         aux1l= objAtomos.setAtomoU(molS[i][j],torS[r])
            ;
         aux2l= objAtomos.setAtomoU(molS[i][x],torS[r])
            ;
         aux3l= objAtomos.setAtomoU(molS[i][x+1],torS[r
            ]);
         aux4l= objAtomos.setAtomoU(molS[i-1][x+2],torS
            [r]);
         v1= objAtomos.getX(aux1l,torS[r])-objAtomos.
            getX(aux2l,torS[r]);
         x1= objAtomos.getY(aux1l,torS[r])-objAtomos.
            getY(aux2l,torS[r]);
         z1= objAtomos.getZ(aux1l,torS[r])-objAtomos.
            getZ(aux2l,torS[r]);
         v2= objAtomos.getX(aux3l,torS[r])-objAtomos.
            getX(aux2l,torS[r]);
         x2= objAtomos.getY(aux3l,torS[r])-objAtomos.
            getY(aux2l,torS[r]);
```

```
z2= objAtomos.getZ(aux3l,torS[r])-objAtomos.
    getZ(aux2l,torS[r]);
v3= objAtomos.getX(aux2l,torS[r])-objAtomos.
    getX(aux3l,torS[r]);
x3= objAtomos.getY(aux2l,torS[r])-objAtomos.
    getY(aux3l,torS[r]);
z3= objAtomos.getZ(aux2l,torS[r])-objAtomos.
    getZ(aux3l,torS[r]);
v4= objAtomos.getX(aux4l,torS[r])-objAtomos.
    getX(aux3l,torS[r]);
x4= objAtomos.getY(aux4l,torS[r])-objAtomos.
    getY(aux3l,torS[r]);
z4= objAtomos.getZ(aux4l,torS[r])-objAtomos.
    getZ(aux3l,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdl= (prod)/(pmod);
angdl= Math.acos(auxdl);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdl= -angdl;
}
pangdl= (getVnU(molS[i][j],molS[i][x],molS[i][
    x+1],molS[i-1][x+2],sem,num,indice)/getPath
    (molS[i][j],molS[i][x],molS[i][x+1],molS[i
    -1][x+2],sem,num,indice)*(1+(Math.cos((getN
```

```
                (molS[i][j],molS[i][x],molS[i][x+1],molS[i
                -1][x+2],sem,indice)*angdl)-getPeriU(molS[i
                ][j],molS[i][x],molS[i][x+1],molS[i-1][x
                +2],sem,indice)))));
            //arredondamento TINKER
            pangdl= (Math.round((pangdl * 10000.0))) /
                10000.0;
            pangdl= pangdl+auxl;
            auxl= pangdl;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdl= (getVnU(molS[i][j],molS[i][x],molS[i
                  ][x+1],molS[i-1][x+2],sem,num,indice)/
                  getPath(molS[i][j],molS[i][x],molS[i][x
                  +1],molS[i-1][x+2],sem,num,indice)*(1+(
                  Math.cos((getN(molS[i][j],molS[i][x],molS
                  [i][x+1],molS[i-1][x+2],sem,indice)*angdl
                  )-getPeriU(molS[i][j],molS[i][x],molS[i][
                  x+1],molS[i-1][x+2],sem,indice)))));
              //arredondamento TINKER
              pangdl= (Math.round((pangdl * 10000.0))) /
                  10000.0;
              pangdl= pangdl+auxl;
              auxl= pangdl;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }else if(((i!=0)&&(j<qntdcol-3)&&(molS[i][j]!="") && (
          molS[i][j+1]!="") && (molS[i][j+2]!="") && (molS[i
          -1][j+3]!="")))){
```

```
if (((molS[i][j].substring(8, 9).equals("1"))&&(molS
    [i][j+1].substring(8, 9).equals("1"))) && (molS[i
    ][j+1].substring(8, 9).equals("1"))&&(molS[i][j
    +2].substring(8, 9).equals("1"))&&
   (molS[i][j+2].substring(10, 11).equals("1"))&&(
      molS[i-1][j+3].substring(11, 12).equals("1"))
      &&(molS[i][j+1]!="X00000001000")){
   indice=0;
   aux1l= objAtomos.setAtomoU(molS[i][j],torS[r]);
   aux2l= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
   aux3l= objAtomos.setAtomoU(molS[i][j+2],torS[r]);
   aux4l= objAtomos.setAtomoU(molS[i-1][j+3],torS[r])
      ;
   v1= objAtomos.getX(aux1l,torS[r])-objAtomos.getX(
      aux2l,torS[r]);
   x1= objAtomos.getY(aux1l,torS[r])-objAtomos.getY(
      aux2l,torS[r]);
   z1= objAtomos.getZ(aux1l,torS[r])-objAtomos.getZ(
      aux2l,torS[r]);
   v2= objAtomos.getX(aux3l,torS[r])-objAtomos.getX(
      aux2l,torS[r]);
   x2= objAtomos.getY(aux3l,torS[r])-objAtomos.getY(
      aux2l,torS[r]);
   z2= objAtomos.getZ(aux3l,torS[r])-objAtomos.getZ(
      aux2l,torS[r]);
   v3= objAtomos.getX(aux2l,torS[r])-objAtomos.getX(
      aux3l,torS[r]);
   x3= objAtomos.getY(aux2l,torS[r])-objAtomos.getY(
      aux3l,torS[r]);
   z3= objAtomos.getZ(aux2l,torS[r])-objAtomos.getZ(
      aux3l,torS[r]);
   v4= objAtomos.getX(aux4l,torS[r])-objAtomos.getX(
      aux3l,torS[r]);
   x4= objAtomos.getY(aux4l,torS[r])-objAtomos.getY(
      aux3l,torS[r]);
```

```
z4= objAtomos.getZ(aux4l,torS[r])-objAtomos.getZ(
    aux3l,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdl= (prod)/(pmod);
angdl= Math.acos(auxdl);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdl= -angdl;
}
pangdl= (getVnU(molS[i][j],molS[i][j+1],molS[i][j
    +2],molS[i-1][j+3],sem,num,indice)/getPath(molS
    [i][j],molS[i][j+1],molS[i][j+2],molS[i-1][j
    +3],sem,num,indice)*(1+(Math.cos((getN(molS[i][
    j],molS[i][j+1],molS[i][j+2],molS[i-1][j+3],sem
    ,indice)*angdl)-getPeriU(molS[i][j],molS[i][j
    +1],molS[i][j+2],molS[i-1][j+3],sem,indice)))))
    ;
//arredondamento TINKER
pangdl= (Math.round((pangdl * 10000.0))) /
    10000.0;
pangdl= pangdl+auxl;
auxl= pangdl;
indiceD= indiceD + 1;
ind++;
sem++;
```

```
        while (num==0){
          //transforma pra graus o cos
          pangdl= (getVnU(molS[i][j],molS[i][j+1],molS[i][
              j+2],molS[i-1][j+3],sem,num,indice)/getPath(
              molS[i][j],molS[i][j+1],molS[i][j+2],molS[i
              -1][j+3],sem,num,indice)*(1+(Math.cos((getN(
              molS[i][j],molS[i][j+1],molS[i][j+2],molS[i
              -1][j+3],sem,indice)*angdl)-getPeriU(molS[i][
              j],molS[i][j+1],molS[i][j+2],molS[i-1][j+3],
              sem,indice)))));
          //arredondamento TINKER
          pangdl= (Math.round((pangdl * 10000.0))) /
              10000.0;
          pangdl= pangdl+auxl;
          auxl= pangdl;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }

      }
    }//fim if
    if(((i!=0)&&(j<qntdcol-5)&&(molS[i][j]!="") && (molS[i
        ][j+3]!="") && (molS[i-1][j+4]!="") && (molS[i-1][j
        +5]!=""))&&
      ((molS[i][j+1]=="X00000001000")&&(molS[i][j+2]=="
          X00000001000"))){
      int x=j+3;
      if((j<qntdcol-5)&&(molS[i][j]!="") && (molS[i][x
          ]!="") && (molS[i-1][x+1]!="") && (molS[i-1][x
          +2]!="")){
        if ((molS[i][j].substring(8, 9).equals("1"))&&(
            molS[i][x].substring(8, 9).equals("1")) && (
            molS[i][x].substring(10, 11).equals("1"))&&(
            molS[i-1][x+1].substring(11, 12).equals("1"))&&
```

```
    (molS[i-1][x].substring(8, 9).equals("1"))&&(
        molS[i-1][x+2].substring(8, 9).equals("1"))
        ){
indice=0;
aux1ll= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2ll= objAtomos.setAtomoU(molS[i][x],torS[r]);
aux3ll= objAtomos.setAtomoU(molS[i-1][x+1],torS[
    r]);
aux4ll= objAtomos.setAtomoU(molS[i-1][x+2],torS[
    r]);
v1= objAtomos.getX(aux1ll,torS[r])-objAtomos.
    getX(aux2ll,torS[r]);
x1= objAtomos.getY(aux1ll,torS[r])-objAtomos.
    getY(aux2ll,torS[r]);
z1= objAtomos.getZ(aux1ll,torS[r])-objAtomos.
    getZ(aux2ll,torS[r]);
v2= objAtomos.getX(aux3ll,torS[r])-objAtomos.
    getX(aux2ll,torS[r]);
x2= objAtomos.getY(aux3ll,torS[r])-objAtomos.
    getY(aux2ll,torS[r]);
z2= objAtomos.getZ(aux3ll,torS[r])-objAtomos.
    getZ(aux2ll,torS[r]);
v3= objAtomos.getX(aux2ll,torS[r])-objAtomos.
    getX(aux3ll,torS[r]);
x3= objAtomos.getY(aux2ll,torS[r])-objAtomos.
    getY(aux3ll,torS[r]);
z3= objAtomos.getZ(aux2ll,torS[r])-objAtomos.
    getZ(aux3ll,torS[r]);
v4= objAtomos.getX(aux4ll,torS[r])-objAtomos.
    getX(aux3ll,torS[r]);
x4= objAtomos.getY(aux4ll,torS[r])-objAtomos.
    getY(aux3ll,torS[r]);
z4= objAtomos.getZ(aux4ll,torS[r])-objAtomos.
    getZ(aux3ll,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
```

```
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdll= (prod)/(pmod);
angdll= Math.acos(auxdll);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdll= -angdll;
}
pangdll= (getVnU(molS[i][j],molS[i][x],molS[i
    -1][x+1],molS[i-1][x+2],sem,num,indice)/
    getPath(molS[i][j],molS[i][x],molS[i-1][x+1],
    molS[i-1][x+2],sem,num,indice)*(1+(Math.cos((
    getN(molS[i][j],molS[i][x],molS[i-1][x+1],
    molS[i-1][x+2],sem,indice)*angdll)-getPeriU(
    molS[i][j],molS[i][x],molS[i-1][x+1],molS[i
    -1][x+2],sem,indice)))));
//arredondamento TINKER
pangdll= (Math.round((pangdll * 10000.0))) /
    10000.0;
pangdll= pangdll+auxll;
auxll= pangdll;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdll= (getVnU(molS[i][j],molS[i][x],molS[i
      -1][x+1],molS[i-1][x+2],sem,num,indice)/
```

```
                getPath(molS[i][j],molS[i][x],molS[i-1][x
                +1],molS[i-1][x+2],sem,num,indice)*(1+(Math
                .cos((getN(molS[i][j],molS[i][x],molS[i-1][
                x+1],molS[i-1][x+2],sem,indice)*angdll)-
                getPeriU(molS[i][j],molS[i][x],molS[i-1][x
                +1],molS[i-1][x+2],sem,indice)))));
            //arredondamento TINKER
            pangdll= (Math.round((pangdll * 10000.0))) /
                10000.0;
            pangdll= pangdll+auxll;
            auxll= pangdll;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }
    }else if (((i!=0)&&(j<qntdcol-3)&&(molS[i][j]!="") &&
      (molS[i][j+1]!="") && (molS[i-1][j+2]!="") && (molS
      [i-1][j+3]!=""))&&(molS[i][j]!="") && (molS[i][j
      +1]!="") && (molS[i-1][j+2]!="") && (molS[i-1][j
      +3]!="")&&(((molS[i][j].substring(8, 9).equals("1")
      )&&(molS[i][j+1].substring(8, 9).equals("1")))) &&
      (molS[i][j+1].substring(10, 11).equals("1"))&&(molS
      [i-1][j+2].substring(11, 12).equals("1"))&&
      (molS[i-1][j+2].substring(8, 9).equals("1"))&&(molS[
          i-1][j+3].substring(8, 9).equals("1"))){
      indice=0;
      aux1ll= objAtomos.setAtomoU(molS[i][j],torS[r]);
      aux2ll= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
      aux3ll= objAtomos.setAtomoU(molS[i-1][j+2],torS[r]);
      aux4ll= objAtomos.setAtomoU(molS[i-1][j+3],torS[r]);
      v1= objAtomos.getX(aux1ll,torS[r])-objAtomos.getX(
          aux2ll,torS[r]);
      x1= objAtomos.getY(aux1ll,torS[r])-objAtomos.getY(
          aux2ll,torS[r]);
```

```
z1= objAtomos.getZ(aux1ll,torS[r])-objAtomos.getZ(
    aux2ll,torS[r]);
v2= objAtomos.getX(aux3ll,torS[r])-objAtomos.getX(
    aux2ll,torS[r]);
x2= objAtomos.getY(aux3ll,torS[r])-objAtomos.getY(
    aux2ll,torS[r]);
z2= objAtomos.getZ(aux3ll,torS[r])-objAtomos.getZ(
    aux2ll,torS[r]);
v3= objAtomos.getX(aux2ll,torS[r])-objAtomos.getX(
    aux3ll,torS[r]);
x3= objAtomos.getY(aux2ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z3= objAtomos.getZ(aux2ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
v4= objAtomos.getX(aux4ll,torS[r])-objAtomos.getX(
    aux3ll,torS[r]);
x4= objAtomos.getY(aux4ll,torS[r])-objAtomos.getY(
    aux3ll,torS[r]);
z4= objAtomos.getZ(aux4ll,torS[r])-objAtomos.getZ(
    aux3ll,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
auxdll= (prod)/(pmod);
angdll= Math.acos(auxdll);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
```

```
    angdll= -angdll;
  }
  pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[i-1][j
      +2],molS[i-1][j+3],sem,num,indice)/getPath(molS[i
      ][j],molS[i][j+1],molS[i-1][j+2],molS[i-1][j+3],
      sem,num,indice)*(1+(Math.cos((getN(molS[i][j],
      molS[i][j+1],molS[i-1][j+2],molS[i-1][j+3],sem,
      indice)*angdll)-getPeriU(molS[i][j],molS[i][j+1],
      molS[i-1][j+2],molS[i-1][j+3],sem,indice)))));
  //arredondamento TINKER
  pangdll= (Math.round((pangdll * 10000.0))) /
      10000.0;
  pangdll= pangdll+auxll;
  auxll= pangdll;
  indiceD= indiceD + 1;
  ind++;
  sem++;
  while (num==0){
    //transforma pra graus o cos
    pangdll= (getVnU(molS[i][j],molS[i][j+1],molS[i
        -1][j+2],molS[i-1][j+3],sem,num,indice)/getPath
        (molS[i][j],molS[i][j+1],molS[i-1][j+2],molS[i
        -1][j+3],sem,num,indice)*(1+(Math.cos((getN(
        molS[i][j],molS[i][j+1],molS[i-1][j+2],molS[i
        -1][j+3],sem,indice)*angdll)-getPeriU(molS[i][j
        ],molS[i][j+1],molS[i-1][j+2],molS[i-1][j+3],
        sem,indice)))));
    //arredondamento TINKER
    pangdll= (Math.round((pangdll * 10000.0))) /
        10000.0;
    pangdll= pangdll+auxll;
    auxll= pangdll;
    indiceD= indiceD + 1;
    ind++;
    sem++;
  }
```

```
        }//fim if
        if((molS[i][j]!="") &&(i<qntdlinha-2)&&(j<qntdcol-2)&&
            (molS[i][j+1]!="") && (molS[i+1][j+2]!="") && (
           molS[i+2][j+2]!="")){
          //verificar se todo esta ligados
          if (((molS[i][j].substring(8, 9).equals("1"))&&(molS
              [i][j+1].substring(8, 9).equals("1"))&&(molS[i][j
              +1].substring(10, 11).equals("1"))&&(molS[i+1][j
              +2].substring(11, 12).equals("1"))&&
               (molS[i+1][j+2].substring(9, 10).equals("1"))&&(
                  molS[i+2][j+2].substring(9, 10).equals("1")))
                  ){
            indice=0;
            aux1b= objAtomos.setAtomoU(molS[i][j],torS[r]);
            aux2b= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
            aux3b= objAtomos.setAtomoU(molS[i+1][j+2],torS[r])
                ;
            aux4b= objAtomos.setAtomoU(molS[i+2][j+2],torS[r])
                ;
            v1= objAtomos.getX(aux1b,torS[r])-objAtomos.getX(
                aux2b,torS[r]);
            x1= objAtomos.getY(aux1b,torS[r])-objAtomos.getY(
                aux2b,torS[r]);
            z1= objAtomos.getZ(aux1b,torS[r])-objAtomos.getZ(
                aux2b,torS[r]);
            v2= objAtomos.getX(aux3b,torS[r])-objAtomos.getX(
                aux2b,torS[r]);
            x2= objAtomos.getY(aux3b,torS[r])-objAtomos.getY(
                aux2b,torS[r]);
            z2= objAtomos.getZ(aux3b,torS[r])-objAtomos.getZ(
                aux2b,torS[r]);
            v3= objAtomos.getX(aux2b,torS[r])-objAtomos.getX(
                aux3b,torS[r]);
            x3= objAtomos.getY(aux2b,torS[r])-objAtomos.getY(
                aux3b,torS[r]);
```

```
z3= objAtomos.getZ(aux2b,torS[r])-objAtomos.getZ(
    aux3b,torS[r]);
v4= objAtomos.getX(aux4b,torS[r])-objAtomos.getX(
    aux3b,torS[r]);
x4= objAtomos.getY(aux4b,torS[r])-objAtomos.getY(
    aux3b,torS[r]);
z4= objAtomos.getZ(aux4b,torS[r])-objAtomos.getZ(
    aux3b,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdb= (prod)/(pmod);
double angdb = Math.acos(auxdb);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb= -angdb;
}
pangdb= (getVnU(molS[i][j],molS[i][j+1],molS[i+1][
    j+2],molS[i+2][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i][j+1],molS[i+1][j+2],molS[i
    +2][j+2],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i][j+1],molS[i+1][j+2],molS[i
    +2][j+2],sem,indice)*angdb)-getPeriU(molS[i][j
    ],molS[i][j+1],molS[i+1][j+2],molS[i+2][j+2],
    sem,indice)))));
//arredondamento TINKER
```

```
        pangdb= (Math.round((pangdb * 10000.0))) /
            10000.0;
        pangdb= pangdb+auxb;
        auxb= pangdb;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdb= (getVnU(molS[i][j],molS[i][j+1],molS[i
              +1][j+2],molS[i+2][j+2],sem,num,indice)/
              getPath(molS[i][j],molS[i][j+1],molS[i+1][j
              +2],molS[i+2][j+2],sem,num,indice)*(1+(Math.
              cos((getN(molS[i][j],molS[i][j+1],molS[i+1][j
              +2],molS[i+2][j+2],sem,indice)*angdb)-
              getPeriU(molS[i][j],molS[i][j+1],molS[i+1][j
              +2],molS[i+2][j+2],sem,indice)))));
          //arredondamento TINKER
          pangdb= (Math.round((pangdb * 10000.0))) /
              10000.0;
          pangdb= pangdb+auxb;
          auxb= pangdb;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim else
    }//fim if
    if((molS[i][j]!="") &&(i<qntdlinha-2)&&(j<qntdcol-1)&&
        (molS[i+1][j]!="") && (molS[i+2][j]!="") && (molS[
       i+2][j+1]!="")&&
      ((molS[i][j].substring(9, 10).equals("1"))&&(molS[i
          +1][j].substring(9, 10).equals("1"))&&(molS[i+1][
          j].substring(9, 10).equals("1"))&&(molS[i+2][j].
          substring(9, 10).equals("1"))&&
```

```
(molS[i+2][j].substring(8, 9).equals("1"))&&(molS[i
    +2][j+1].substring(8, 9).equals("1")))){
if ((molS[i+1][j].substring(0, 7).equals("OHR2001"))
    &&(molS[i+2][j].substring(0, 7).equals("P1P0001")
    )){

}else{
  indice=0;
  aux1bb= objAtomos.setAtomoU(molS[i][j],torS[r]);
  aux2bb= objAtomos.setAtomoU(molS[i+1][j],torS[r]);
  aux3bb= objAtomos.setAtomoU(molS[i+2][j],torS[r]);
  aux4bb= objAtomos.setAtomoU(molS[i+2][j+1],torS[r
      ]);
  v1= objAtomos.getX(aux1bb,torS[r])-objAtomos.getX(
      aux2bb,torS[r]);
  x1= objAtomos.getY(aux1bb,torS[r])-objAtomos.getY(
      aux2bb,torS[r]);
  z1= objAtomos.getZ(aux1bb,torS[r])-objAtomos.getZ(
      aux2bb,torS[r]);
  v2= objAtomos.getX(aux3bb,torS[r])-objAtomos.getX(
      aux2bb,torS[r]);
  x2= objAtomos.getY(aux3bb,torS[r])-objAtomos.getY(
      aux2bb,torS[r]);
  z2= objAtomos.getZ(aux3bb,torS[r])-objAtomos.getZ(
      aux2bb,torS[r]);
  v3= objAtomos.getX(aux2bb,torS[r])-objAtomos.getX(
      aux3bb,torS[r]);
  x3= objAtomos.getY(aux2bb,torS[r])-objAtomos.getY(
      aux3bb,torS[r]);
  z3= objAtomos.getZ(aux2bb,torS[r])-objAtomos.getZ(
      aux3bb,torS[r]);
  v4= objAtomos.getX(aux4bb,torS[r])-objAtomos.getX(
      aux3bb,torS[r]);
  x4= objAtomos.getY(aux4bb,torS[r])-objAtomos.getY(
      aux3bb,torS[r]);
```

```
z4= objAtomos.getZ(aux4bb,torS[r])-objAtomos.getZ(
    aux3bb,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdbb= (prod)/(pmod);
angdbb = Math.acos(auxdbb);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb= -angdbb;
}
pangdbb= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j],molS[i+2][j+1],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j],molS[i+2][j],molS[i+2][
    j+1],sem,num,indice)*(1+(Math.cos((getN(molS[i
    ][j],molS[i+1][j],molS[i+2][j],molS[i+2][j+1],
    sem,indice)*angdbb)-getPeriU(molS[i][j],molS[i
    +1][j],molS[i+2][j],molS[i+2][j+1],sem,indice))
    ))));
//arredondamento TINKER
pangdbb= (Math.round((pangdbb * 10000.0))) /
    10000.0;
pangdbb= pangdbb+auxbb;
auxbb= pangdbb;
indiceD= indiceD + 1;
ind++;
sem++;
```

```
        while (num==0){
          //transforma pra graus o cos
          pangdbb= (getVnU(molS[i][j],molS[i+1][j],molS[i
              +2][j],molS[i+2][j+1],sem,num,indice)/getPath
              (molS[i][j],molS[i+1][j],molS[i+2][j],molS[i
              +2][j+1],sem,num,indice)*(1+(Math.cos((getN(
              molS[i][j],molS[i+1][j],molS[i+2][j],molS[i
              +2][j+1],sem,indice)*angdbb)-getPeriU(molS[i
              ][j],molS[i+1][j],molS[i+2][j],molS[i+2][j
              +1],sem,indice)))));
          //arredondamento TINKER
          pangdbb= (Math.round((pangdbb * 10000.0))) /
              10000.0;
          pangdbb= pangdbb+auxbb;
          auxbb= pangdbb;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }
    }//fim if
    if((((i<qntdlinha-1)&&(j<qntdcol-5)&&(molS[i][j]!="")
       && (molS[i+1][j+1]!="") && (molS[i+1][j+4]!="") &&
       (molS[i+1][j+5]!="")))&&
      //posicao tres tem X
      ((molS[i+1][j+2]=="X00000001000")&&(molS[i+1][j
          +3]=="X00000001000"))){
      int x=j+4;
      if((j<qntdcol-5)&&(molS[i][j]!="") && (molS[i+1][j
          +1]!="") && (molS[i+1][x]!="") && (molS[i+1][x
          +1]!="")){
        if ((molS[i][j].substring(10, 11).equals("1"))&&(
            molS[i+1][j+1].substring(11, 12).equals("1"))
           && (molS[i+1][j+1].substring(8, 9).equals("1"))
           &&(molS[i+1][x].substring(8, 9).equals("1"))&&
```

```
    (molS[i+1][x].substring(8, 9).equals("1"))&&(
        molS[i+1][x+1].substring(8, 9).equals("1"))
        ){
indice=0;
aux1b3= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2b3= objAtomos.setAtomoU(molS[i+1][j+1],torS[
    r]);
aux3b3= objAtomos.setAtomoU(molS[i+1][x],torS[r
    ]);
aux4b3= objAtomos.setAtomoU(molS[i+1][x+1],torS[
    r]);
v1= objAtomos.getX(aux1b3,torS[r])-objAtomos.
    getX(aux2b3,torS[r]);
x1= objAtomos.getY(aux1b3,torS[r])-objAtomos.
    getY(aux2b3,torS[r]);
z1= objAtomos.getZ(aux1b3,torS[r])-objAtomos.
    getZ(aux2b3,torS[r]);
v2= objAtomos.getX(aux3b3,torS[r])-objAtomos.
    getX(aux2b3,torS[r]);
x2= objAtomos.getY(aux3b3,torS[r])-objAtomos.
    getY(aux2b3,torS[r]);
z2= objAtomos.getZ(aux3b3,torS[r])-objAtomos.
    getZ(aux2b3,torS[r]);
v3= objAtomos.getX(aux2b3,torS[r])-objAtomos.
    getX(aux3b3,torS[r]);
x3= objAtomos.getY(aux2b3,torS[r])-objAtomos.
    getY(aux3b3,torS[r]);
z3= objAtomos.getZ(aux2b3,torS[r])-objAtomos.
    getZ(aux3b3,torS[r]);
v4= objAtomos.getX(aux4b3,torS[r])-objAtomos.
    getX(aux3b3,torS[r]);
x4= objAtomos.getY(aux4b3,torS[r])-objAtomos.
    getY(aux3b3,torS[r]);
z4= objAtomos.getZ(aux4b3,torS[r])-objAtomos.
    getZ(aux3b3,torS[r]);
a1= ((x1*z2)-(x2*z1));
```

```
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb3= (prod)/(pmod);
angdb3= Math.acos(auxdb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb3= -angdb3;
}
pangdb3= (getVnU(molS[i][j],molS[i+1][j+1],molS[
    i+1][x],molS[i+1][x+1],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j+1],molS[i+1][x
    ],molS[i+1][x+1],sem,num,indice)*(1+(Math.cos
    ((getN(molS[i][j],molS[i+1][j+1],molS[i+1][x
    ],molS[i+1][x+1],sem,indice)*angdb3)-getPeriU
    (molS[i][j],molS[i+1][j+1],molS[i+1][x],molS[
    i+1][x+1],sem,indice)))));
//arredondamento TINKER
pangdb3= (Math.round((pangdb3 * 10000.0))) /
    10000.0;
pangdb3= pangdb3+auxb3;
auxb3= pangdb3;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
```

```
                pangdb3= (getVnU(molS[i][j],molS[i+1][j+1],
                    molS[i+1][x],molS[i+1][x+1],sem,num,indice)
                    /getPath(molS[i][j],molS[i+1][j+1],molS[i
                    +1][x],molS[i+1][x+1],sem,num,indice)*(1+(
                    Math.cos((getN(molS[i][j],molS[i+1][j+1],
                    molS[i+1][x],molS[i+1][x+1],sem,indice)*
                    angdb3)-getPeriU(molS[i][j],molS[i+1][j+1],
                    molS[i+1][x],molS[i+1][x+1],sem,indice)))))
                    ;
                //arredondamento TINKER
                pangdb3= (Math.round((pangdb3 * 10000.0))) /
                    10000.0;
                pangdb3= pangdb3+auxb3;
                auxb3= pangdb3;
                indiceD= indiceD + 1;
                ind++;
                sem++;
            }
        }
    }
}else if ((i<qntdlinha-1)&&(j<qntdcol-3)&&(molS[i][j
    ]!="") && (molS[i+1][j+1]!="") && (molS[i+1][j
    +2]!="") && (molS[i+1][j+3]!="")&&(molS[i][j].
    substring(10, 11).equals("1"))&&(molS[i+1][j+1].
    substring(11, 12).equals("1")) && (molS[i+1][j+1].
    substring(8, 9).equals("1"))&&(molS[i+1][j+2].
    substring(8, 9).equals("1"))&&
    (molS[i+1][j+2].substring(8, 9).equals("1"))&&(molS[
        i+1][j+3].substring(8, 9).equals("1"))&&
    (molS[i+1][j+2]!="X00000001000")){
    indice=0;
    aux1b3= objAtomos.setAtomoU(molS[i][j],torS[r]);
    aux2b3= objAtomos.setAtomoU(molS[i+1][j+1],torS[r]);
    aux3b3= objAtomos.setAtomoU(molS[i+1][j+2],torS[r]);
    aux4b3= objAtomos.setAtomoU(molS[i+1][j+3],torS[r]);
```

```
v1= objAtomos.getX(aux1b3,torS[r])-objAtomos.getX(
    aux2b3,torS[r]);
x1= objAtomos.getY(aux1b3,torS[r])-objAtomos.getY(
    aux2b3,torS[r]);
z1= objAtomos.getZ(aux1b3,torS[r])-objAtomos.getZ(
    aux2b3,torS[r]);
v2= objAtomos.getX(aux3b3,torS[r])-objAtomos.getX(
    aux2b3,torS[r]);
x2= objAtomos.getY(aux3b3,torS[r])-objAtomos.getY(
    aux2b3,torS[r]);
z2= objAtomos.getZ(aux3b3,torS[r])-objAtomos.getZ(
    aux2b3,torS[r]);
v3= objAtomos.getX(aux2b3,torS[r])-objAtomos.getX(
    aux3b3,torS[r]);
x3= objAtomos.getY(aux2b3,torS[r])-objAtomos.getY(
    aux3b3,torS[r]);
z3= objAtomos.getZ(aux2b3,torS[r])-objAtomos.getZ(
    aux3b3,torS[r]);
v4= objAtomos.getX(aux4b3,torS[r])-objAtomos.getX(
    aux3b3,torS[r]);
x4= objAtomos.getY(aux4b3,torS[r])-objAtomos.getY(
    aux3b3,torS[r]);
z4= objAtomos.getZ(aux4b3,torS[r])-objAtomos.getZ(
    aux3b3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
```

```
auxdb3= (prod)/(pmod);
angdb3= Math.acos(auxdb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb3= -angdb3;
}
pangdb3= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
    +1][j+2],molS[i+1][j+3],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i
    +1][j+3],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i+1][j
    +3],sem,indice)*angdb3)-getPeriU(molS[i][j],molS[
    i+1][j+1],molS[i+1][j+2],molS[i+1][j+3],sem,
    indice)))));
//arredondamento TINKER
pangdb3= (Math.round((pangdb3 * 10000.0))) /
    10000.0;
pangdb3= pangdb3+auxb3;
auxb3= pangdb3;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdb3= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
      +1][j+2],molS[i+1][j+3],sem,num,indice)/getPath
      (molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[
      i+1][j+3],sem,num,indice)*(1+(Math.cos((getN(
      molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i
      +1][j+3],sem,indice)*angdb3)-getPeriU(molS[i][j
      ],molS[i+1][j+1],molS[i+1][j+2],molS[i+1][j+3],
      sem,indice)))));
  //arredondamento TINKER
  pangdb3= (Math.round((pangdb3 * 10000.0))) /
      10000.0;
  pangdb3= pangdb3+auxb3;
```

```
        auxb3= pangdb3;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }//fim if
    if((molS[i][j]!="") &&(i<qntdlinha-2)&&(j>=1)&& (molS[
        i+1][j]!="") && (molS[i+2][j]!="") && (molS[i+2][j
        -1]!="")&&
      ((molS[i][j].substring(9, 10).equals("1"))&&(molS[i
          +1][j].substring(9, 10).equals("1"))&&(molS[i+1][
          j].substring(9, 10).equals("1"))&&(molS[i+2][j].
          substring(9, 10).equals("1"))&&
      (molS[i+2][j].substring(8, 9).equals("1"))&&(molS[i
          +2][j-1].substring(8, 9).equals("1")))){
      if (((molS[i+1][j].substring(0, 7).equals("OHR2001")
          )&&(molS[i+2][j].substring(0, 7).equals("P1P0001
          ")))||
            ((molS[i+2][j].substring(0, 7).equals("OHR1002")
                )&&(molS[i+2][j-1].substring(0, 7).equals("
                OHR1001")))){

      }else{
        indice=0;
        aux1c2= objAtomos.setAtomoU(molS[i][j],torS[r]);
        aux2c2= objAtomos.setAtomoU(molS[i+1][j],torS[r]);
        aux3c2= objAtomos.setAtomoU(molS[i+2][j],torS[r]);
        aux4c2= objAtomos.setAtomoU(molS[i+2][j-1],torS[r
            ]);
        v1= objAtomos.getX(aux1c2,torS[r])-objAtomos.getX(
            aux2c2,torS[r]);
        x1= objAtomos.getY(aux1c2,torS[r])-objAtomos.getY(
            aux2c2,torS[r]);
        z1= objAtomos.getZ(aux1c2,torS[r])-objAtomos.getZ(
            aux2c2,torS[r]);
```

```
v2= objAtomos.getX(aux3c2,torS[r])-objAtomos.getX(
    aux2c2,torS[r]);
x2= objAtomos.getY(aux3c2,torS[r])-objAtomos.getY(
    aux2c2,torS[r]);
z2= objAtomos.getZ(aux3c2,torS[r])-objAtomos.getZ(
    aux2c2,torS[r]);
v3= objAtomos.getX(aux2c2,torS[r])-objAtomos.getX(
    aux3c2,torS[r]);
x3= objAtomos.getY(aux2c2,torS[r])-objAtomos.getY(
    aux3c2,torS[r]);
z3= objAtomos.getZ(aux2c2,torS[r])-objAtomos.getZ(
    aux3c2,torS[r]);
v4= objAtomos.getX(aux4c2,torS[r])-objAtomos.getX(
    aux3c2,torS[r]);
x4= objAtomos.getY(aux4c2,torS[r])-objAtomos.getY(
    aux3c2,torS[r]);
z4= objAtomos.getZ(aux4c2,torS[r])-objAtomos.getZ(
    aux3c2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdc2= (prod)/(pmod);
angdc2= Math.acos(auxdc2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc2= -angdc2;
}
```

```
        pangdc2= (getVnU(molS[i][j],molS[i+1][j],molS[i
            +2][j],molS[i+2][j-1],sem,num,indice)/getPath(
            molS[i][j],molS[i+1][j],molS[i+2][j],molS[i+2][
            j-1],sem,num,indice)*(1+(Math.cos((getN(molS[i
            ][j],molS[i+1][j],molS[i+2][j],molS[i+2][j-1],
            sem,indice)*angdc2)-getPeriU(molS[i][j],molS[i
            +1][j],molS[i+2][j],molS[i+2][j-1],sem,indice))
            ))));
        //arredondamento TINKER
        pangdc2= (Math.round((pangdc2 * 10000.0))) /
            10000.0;
        pangdc2= pangdc2+auxc2;
        auxc2= pangdc2;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdc2= (getVnU(molS[i][j],molS[i+1][j],molS[i
              +2][j],molS[i+2][j-1],sem,num,indice)/getPath
              (molS[i][j],molS[i+1][j],molS[i+2][j],molS[i
              +2][j-1],sem,num,indice)*(1+(Math.cos((getN(
              molS[i][j],molS[i+1][j],molS[i+2][j],molS[i
              +2][j-1],sem,indice)*angdc2)-getPeriU(molS[i
              ][j],molS[i+1][j],molS[i+2][j],molS[i+2][j
              -1],sem,indice)))));
          //arredondamento TINKER
          pangdc2= (Math.round((pangdc2 * 10000.0))) /
              10000.0;
          pangdc2= pangdc2+auxc2;
          auxc2= pangdc2;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }
```

```
}//fim if
if(((i<qntdlinha-1)&&(molS[i][j]!="")&& (j<qntdcol-3)
   &&(molS[i][j+1]!="") && (molS[i+1][j+2]!="")&& (
   molS[i+1][j+3]!=""))&&
   ((molS[i][j].substring(8, 9).equals("1"))&&(molS[i][
      j+1].substring(8, 9).equals("1"))&&(molS[i][j+1].
      substring(10, 11).equals("1"))&&(molS[i+1][j+2].
      substring(11, 12).equals("1"))&&
   (molS[i+1][j+2].substring(8, 9).equals("1"))&&(molS[
      i+1][j+3].substring(8, 9).equals("1")))){
   indice=0;
   auxdbb3=0;
   auxbb3=0;
   aux1bb3= objAtomos.setAtomoU(molS[i][j],torS[r]);
   aux2bb3= objAtomos.setAtomoU(molS[i][j+1],torS[r]);
   aux3bb3= objAtomos.setAtomoU(molS[i+1][j+2],torS[r])
      ;
   aux4bb3= objAtomos.setAtomoU(molS[i+1][j+3],torS[r])
      ;
   v1= objAtomos.getX(aux1bb3,torS[r])-objAtomos.getX(
      aux2bb3,torS[r]);
   x1= objAtomos.getY(aux1bb3,torS[r])-objAtomos.getY(
      aux2bb3,torS[r]);
   z1= objAtomos.getZ(aux1bb3,torS[r])-objAtomos.getZ(
      aux2bb3,torS[r]);
   v2= objAtomos.getX(aux3bb3,torS[r])-objAtomos.getX(
      aux2bb3,torS[r]);
   x2= objAtomos.getY(aux3bb3,torS[r])-objAtomos.getY(
      aux2bb3,torS[r]);
   z2= objAtomos.getZ(aux3bb3,torS[r])-objAtomos.getZ(
      aux2bb3,torS[r]);
   v3= objAtomos.getX(aux2bb3,torS[r])-objAtomos.getX(
      aux3bb3,torS[r]);
   x3= objAtomos.getY(aux2bb3,torS[r])-objAtomos.getY(
      aux3bb3,torS[r]);
```

```
z3= objAtomos.getZ(aux2bb3,torS[r])-objAtomos.getZ(
    aux3bb3,torS[r]);
v4= objAtomos.getX(aux4bb3,torS[r])-objAtomos.getX(
    aux3bb3,torS[r]);
x4= objAtomos.getY(aux4bb3,torS[r])-objAtomos.getY(
    aux3bb3,torS[r]);
z4= objAtomos.getZ(aux4bb3,torS[r])-objAtomos.getZ(
    aux3bb3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
auxdbb3= (prod)/(pmod);
angdbb3= Math.acos(auxdbb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb3= -angdbb3;
}
pangdbb3= (getVnU(molS[i][j],molS[i][j+1],molS[i+1][
    j+2],molS[i+1][j+3],sem,num,indice)/getPath(molS[
    i][j],molS[i][j+1],molS[i+1][j+2],molS[i+1][j+3],
    sem,num,indice)*(1+(Math.cos((getN(molS[i][j],
    molS[i][j+1],molS[i+1][j+2],molS[i+1][j+3],sem,
    indice)*angdbb3)-getPeriU(molS[i][j],molS[i][j
    +1],molS[i+1][j+2],molS[i+1][j+3],sem,indice)))))
    ;
//arredondamento TINKER
```

```
        pangdbb3= (Math.round((pangdbb3 * 10000.0))) /
            10000.0;
        pangdbb3= pangdbb3+auxbb3;
        auxbb3= pangdbb3;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdbb3= (getVnU(molS[i][j],molS[i][j+1],molS[i
              +1][j+2],molS[i+1][j+3],sem,num,indice)/getPath
              (molS[i][j],molS[i][j+1],molS[i+1][j+2],molS[i
              +1][j+3],sem,num,indice)*(1+(Math.cos((getN(
              molS[i][j],molS[i][j+1],molS[i+1][j+2],molS[i
              +1][j+3],sem,indice)*angdbb3)-getPeriU(molS[i][
              j],molS[i][j+1],molS[i+1][j+2],molS[i+1][j+3],
              sem,indice)))));
          //arredondamento TINKER
          pangdbb3= (Math.round((pangdbb3 * 10000.0))) /
              10000.0;
          pangdbb3= pangdbb3+auxbb3;
          auxbb3= pangdbb3;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim if
      if((i<qntdlinha-2)&&(j<qntdcol-2)&&(molS[i][j]!="") &&
          (molS[i+1][j+1]!="") && (molS[i+1][j+2]!="")&& (
         molS[i+2][j+2]!="")&&
        ((molS[i][j].substring(10, 11).equals("1"))&&(molS[i
            +1][j+1].substring(11, 12).equals("1"))&&(molS[i
            +1][j+1].substring(8, 9).equals("1"))&&(molS[i
            +1][j+2].substring(8, 9).equals("1"))&&
        (molS[i+1][j+2].substring(9, 10).equals("1"))&&(molS
            [i+2][j+2].substring(9, 10).equals("1")))){
```

```
indice=0;
aux1eb3= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2eb3= objAtomos.setAtomoU(molS[i+1][j+1],torS[r])
    ;
aux3eb3= objAtomos.setAtomoU(molS[i+1][j+2],torS[r])
    ;
aux4eb3= objAtomos.setAtomoU(molS[i+2][j+2],torS[r])
    ;
v1= objAtomos.getX(aux1eb3,torS[r])-objAtomos.getX(
    aux2eb3,torS[r]);
x1= objAtomos.getY(aux1eb3,torS[r])-objAtomos.getY(
    aux2eb3,torS[r]);
z1= objAtomos.getZ(aux1eb3,torS[r])-objAtomos.getZ(
    aux2eb3,torS[r]);
v2= objAtomos.getX(aux3eb3,torS[r])-objAtomos.getX(
    aux2eb3,torS[r]);
x2= objAtomos.getY(aux3eb3,torS[r])-objAtomos.getY(
    aux2eb3,torS[r]);
z2= objAtomos.getZ(aux3eb3,torS[r])-objAtomos.getZ(
    aux2eb3,torS[r]);
v3= objAtomos.getX(aux2eb3,torS[r])-objAtomos.getX(
    aux3eb3,torS[r]);
x3= objAtomos.getY(aux2eb3,torS[r])-objAtomos.getY(
    aux3eb3,torS[r]);
z3= objAtomos.getZ(aux2eb3,torS[r])-objAtomos.getZ(
    aux3eb3,torS[r]);
v4= objAtomos.getX(aux4eb3,torS[r])-objAtomos.getX(
    aux3eb3,torS[r]);
x4= objAtomos.getY(aux4eb3,torS[r])-objAtomos.getY(
    aux3eb3,torS[r]);
z4= objAtomos.getZ(aux4eb3,torS[r])-objAtomos.getZ(
    aux3eb3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
```

```
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
auxdeb3= (prod)/(pmod);
angdeb3= Math.acos(auxdeb3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdeb3= -angdeb3;
}
pangdeb3= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
    +1][j+2],molS[i+2][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i
    +2][j+2],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i+2][j
    +2],sem,indice)*angdeb3)-getPeriU(molS[i][j],molS
    [i+1][j+1],molS[i+1][j+2],molS[i+2][j+2],sem,
    indice)))));
//arredondamento TINKER
pangdeb3= (Math.round((pangdeb3 * 10000.0))) /
    10000.0;
pangdeb3= pangdeb3+auxeb3;
auxeb3= pangdeb3;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdeb3= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
      +1][j+2],molS[i+2][j+2],sem,num,indice)/getPath
      (molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[
      i+2][j+2],sem,num,indice)*(1+(Math.cos((getN(
```

```
            molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i
            +2][j+2],sem,indice)*angdeb3)-getPeriU(molS[i][
            j],molS[i+1][j+1],molS[i+1][j+2],molS[i+2][j
            +2],sem,indice)))));
        //arredondamento TINKER
        pangdb3= (Math.round((pangdeb3 * 10000.0))) /
            10000.0;
        pangdeb3= pangdeb3+auxeb3;
        auxeb3= pangdeb3;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }//fim if
    if(((i<qntdlinha-2)&&(j>=4)&&(molS[i][j]!="") && (molS
        [i+1][j]!="") && (molS[i+2][j-1]!="")&& (molS[i+2][
        j-4]!=""))&&

      ((molS[i+2][j-2]=="X00000001000")&&(molS[i+2][j
          -3]=="X00000001000"))){
      int x=j-4;
      if((molS[i][j]!="") && (molS[i+1][j]!="") && (molS[i
          +2][j-1]!="") && (molS[i+1][x]!="")){
        if ((molS[i][j].substring(9, 10).equals("1"))&&(
            molS[i+1][j].substring(9, 10).equals("1")) && (
            molS[i+1][j].substring(11, 12).equals("1"))&&(
            molS[i+2][j-1].substring(10, 11).equals("1"))&&
           (molS[i+2][j-1].substring(8, 9).equals("1"))&&(
              molS[i+2][x].substring(8, 9).equals("1"))){
          indice=0;
          aux1bb1= objAtomos.setAtomoU(molS[i][j],torS[r])
              ;
          aux2bb1= objAtomos.setAtomoU(molS[i+1][j],torS[r
              ]);
          aux3bb1= objAtomos.setAtomoU(molS[i+2][j-1],torS
              [r]);
```

```
aux4bb1= objAtomos.setAtomoU(molS[i+2][x],torS[r
    ]);
v1= objAtomos.getX(aux1bb1,torS[r])-objAtomos.
    getX(aux2bb1,torS[r]);
x1= objAtomos.getY(aux1bb1,torS[r])-objAtomos.
    getY(aux2bb1,torS[r]);
z1= objAtomos.getZ(aux1bb1,torS[r])-objAtomos.
    getZ(aux2bb1,torS[r]);
v2= objAtomos.getX(aux3bb1,torS[r])-objAtomos.
    getX(aux2bb1,torS[r]);
x2= objAtomos.getY(aux3bb1,torS[r])-objAtomos.
    getY(aux2bb1,torS[r]);
z2= objAtomos.getZ(aux3bb1,torS[r])-objAtomos.
    getZ(aux2bb1,torS[r]);
v3= objAtomos.getX(aux2bb1,torS[r])-objAtomos.
    getX(aux3bb1,torS[r]);
x3= objAtomos.getY(aux2bb1,torS[r])-objAtomos.
    getY(aux3bb1,torS[r]);
z3= objAtomos.getZ(aux2bb1,torS[r])-objAtomos.
    getZ(aux3bb1,torS[r]);
v4= objAtomos.getX(aux4bb1,torS[r])-objAtomos.
    getX(aux3bb1,torS[r]);
x4= objAtomos.getY(aux4bb1,torS[r])-objAtomos.
    getY(aux3bb1,torS[r]);
z4= objAtomos.getZ(aux4bb1,torS[r])-objAtomos.
    getZ(aux3bb1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
```

```
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb1= (prod)/(pmod);
double angdbb1= Math.acos(auxdbb1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb1= -angdbb1;
}
pangdbb1= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j-1],molS[i+2][x],sem,num,indice)/getPath
    (molS[i][j],molS[i+1][j],molS[i+2][j-1],molS[
    i+2][x],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j],molS[i+2][j-1],molS[i
    +2][x],sem,indice)*angdbb1)-getPeriU(molS[i][
    j],molS[i+1][j],molS[i+2][j-1],molS[i+2][x],
    sem,indice)))));
//arredondamento TINKER
pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
    10000.0;
pangdbb1= pangdbb1+auxbb1;
auxbb1= pangdbb1;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdbb1= (getVnU(molS[i][j],molS[i+1][j],molS
      [i+2][j-1],molS[i+2][x],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j],molS[i+2][j
      -1],molS[i+2][x],sem,num,indice)*(1+(Math.
      cos((getN(molS[i][j],molS[i+1][j],molS[i
      +2][j-1],molS[i+2][x],sem,indice)*angdbb1)-
      getPeriU(molS[i][j],molS[i+1][j],molS[i+2][
      j-1],molS[i+2][x],sem,indice)))));
  //arredondamento TINKER
```

```
                pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
                    10000.0;
                pangdbb1= pangdbb1+auxbb1;
                auxbb1= pangdbb1;
                indiceD= indiceD + 1;
                ind++;
                sem++;
              }
            }
          }
        }else if(((i<qntdlinha-2)&&(j>=2)&&(molS[i][j]!="") &&
            (molS[i+1][j]!="") && (molS[i+2][j-1]!="")&& (molS
            [i+2][j-2]!=""))&&(molS[i][j].substring(9, 10).
            equals("1"))&&(molS[i+1][j].substring(9, 10).equals
            ("1"))&&(molS[i+1][j].substring(11, 12).equals("1")
            )&&(molS[i+2][j-1].substring(10, 11).equals("1"))&&
            (molS[i+2][j-1].substring(8, 9).equals("1"))&&(molS[
                i+2][j-2].substring(8, 9).equals("1"))&& (molS[i
                +2][j-2]!="")){
          indice=0;
          aux1bb1= objAtomos.setAtomoU(molS[i][j],torS[r]);
          aux2bb1= objAtomos.setAtomoU(molS[i+1][j],torS[r]);
          aux3bb1= objAtomos.setAtomoU(molS[i+2][j-1],torS[r])
              ;
          aux4bb1= objAtomos.setAtomoU(molS[i+2][j-2],torS[r])
              ;
          v1= objAtomos.getX(aux1bb1,torS[r])-objAtomos.getX(
              aux2bb1,torS[r]);
          x1= objAtomos.getY(aux1bb1,torS[r])-objAtomos.getY(
              aux2bb1,torS[r]);
          z1= objAtomos.getZ(aux1bb1,torS[r])-objAtomos.getZ(
              aux2bb1,torS[r]);
          v2= objAtomos.getX(aux3bb1,torS[r])-objAtomos.getX(
              aux2bb1,torS[r]);
          x2= objAtomos.getY(aux3bb1,torS[r])-objAtomos.getY(
              aux2bb1,torS[r]);
```

```
z2= objAtomos.getZ(aux3bb1,torS[r])-objAtomos.getZ(
    aux2bb1,torS[r]);
v3= objAtomos.getX(aux2bb1,torS[r])-objAtomos.getX(
    aux3bb1,torS[r]);
x3= objAtomos.getY(aux2bb1,torS[r])-objAtomos.getY(
    aux3bb1,torS[r]);
z3= objAtomos.getZ(aux2bb1,torS[r])-objAtomos.getZ(
    aux3bb1,torS[r]);
v4= objAtomos.getX(aux4bb1,torS[r])-objAtomos.getX(
    aux3bb1,torS[r]);
x4= objAtomos.getY(aux4bb1,torS[r])-objAtomos.getY(
    aux3bb1,torS[r]);
z4= objAtomos.getZ(aux4bb1,torS[r])-objAtomos.getZ(
    aux3bb1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
auxdbb1= (prod)/(pmod);
double angdbb1= Math.acos(auxdbb1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb1= -angdbb1;
}
pangdbb1= (getVnU(molS[i][j],molS[i+1][j],molS[i+2][
    j-1],molS[i+2][j-2],sem,num,indice)/getPath(molS[
    i][j],molS[i+1][j],molS[i+2][j-1],molS[i+2][j-2],
    sem,num,indice)*(1+(Math.cos((getN(molS[i][j],
```

```
            molS[i+1][j],molS[i+2][j-1],molS[i+2][j-2],sem,
            indice)*angdbb1)-getPeriU(molS[i][j],molS[i+1][j
            ],molS[i+2][j-1],molS[i+2][j-2],sem,indice)))));
        //arredondamento TINKER
        pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
            10000.0;
        pangdbb1= pangdbb1+auxbb1;
        auxbb1= pangdbb1;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdbb1= (getVnU(molS[i][j],molS[i+1][j],molS[i
              +2][j-1],molS[i+2][j-2],sem,num,indice)/getPath
              (molS[i][j],molS[i+1][j],molS[i+2][j-1],molS[i
              +2][j-2],sem,num,indice)*(1+(Math.cos((getN(
              molS[i][j],molS[i+1][j],molS[i+2][j-1],molS[i
              +2][j-2],sem,indice)*angdbb1)-getPeriU(molS[i][
              j],molS[i+1][j],molS[i+2][j-1],molS[i+2][j-2],
              sem,indice)))));
          //arredondamento TINKER
          pangdbb1= (Math.round((pangdbb1 * 10000.0))) /
              10000.0;
          pangdbb1= pangdbb1+auxbb1;
          auxbb1= pangdbb1;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim if
      if(((i<qntdlinha-2)&&(j<qntdcol-1)&&(j>=2)&&(molS[i][j
          ]!="") && (molS[i+1][j]!="") && (molS[i+2][j
          +1]!="")&& (molS[i+2][j-2]!=""))&&
        ((molS[i+2][j]=="X00000001000")&&(molS[i+2][j-1]=="
            X00000001000"))){
```

```
int x=j-2;
if((molS[i][j]!="") && (molS[i+1][j]!="") && (molS[i
    +2][j+1]!="") && (molS[i+2][x]!="")){
  if ((molS[i][j].substring(9, 10).equals("1"))&&(
      molS[i+1][j].substring(9, 10).equals("1")) && (
      molS[i+1][j].substring(10, 11).equals("1"))&&(
      molS[i+2][j+1].substring(11, 12).equals("1"))&&
     (molS[i+2][j+1].substring(8, 9).equals("1"))&&(
        molS[i+2][x].substring(8, 9).equals("1"))){
     indice=0;
     aux1c= objAtomos.setAtomoU(molS[i][j],torS[r]);
     aux2c= objAtomos.setAtomoU(molS[i+1][j],torS[r])
        ;
     aux3c= objAtomos.setAtomoU(molS[i+2][j+1],torS[r
        ]);
     aux4c= objAtomos.setAtomoU(molS[i+2][x],torS[r])
        ;
     v1= objAtomos.getX(aux1c,torS[r])-objAtomos.getX
        (aux2c,torS[r]);
     x1= objAtomos.getY(aux1c,torS[r])-objAtomos.getY
        (aux2c,torS[r]);
     z1= objAtomos.getZ(aux1c,torS[r])-objAtomos.getZ
        (aux2c,torS[r]);
     v2= objAtomos.getX(aux3c,torS[r])-objAtomos.getX
        (aux2c,torS[r]);
     x2= objAtomos.getY(aux3c,torS[r])-objAtomos.getY
        (aux2c,torS[r]);
     z2= objAtomos.getZ(aux3c,torS[r])-objAtomos.getZ
        (aux2c,torS[r]);
     v3= objAtomos.getX(aux2c,torS[r])-objAtomos.getX
        (aux3c,torS[r]);
     x3= objAtomos.getY(aux2c,torS[r])-objAtomos.getY
        (aux3c,torS[r]);
     z3= objAtomos.getZ(aux2c,torS[r])-objAtomos.getZ
        (aux3c,torS[r]);
```

```
v4= objAtomos.getX(aux4c,torS[r])-objAtomos.getX
    (aux3c,torS[r]);
x4= objAtomos.getY(aux4c,torS[r])-objAtomos.getY
    (aux3c,torS[r]);
z4= objAtomos.getZ(aux4c,torS[r])-objAtomos.getZ
    (aux3c,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdc= (prod)/(pmod);
double angdc= Math.acos(auxdc);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc= -angdc;
}
pangdc= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j+1],molS[i+2][x],sem,num,indice)/getPath
    (molS[i][j],molS[i+1][j],molS[i+2][j+1],molS[
    i+2][x],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j],molS[i+2][j+1],molS[i
    +2][x],sem,indice)*angdc)-getPeriU(molS[i][j
    ],molS[i+1][j],molS[i+2][j+1],molS[i+2][x],
    sem,indice)))));
//arredondamento TINKER
pangdc= (Math.round((pangdc * 10000.0))) /
    10000.0;
pangdc= pangdc+auxc;
```

```
            auxc= pangdc;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdc= (getVnU(molS[i][j],molS[i+1][j],molS[i
                  +2][j+1],molS[i+2][x],sem,num,indice)/
                  getPath(molS[i][j],molS[i+1][j],molS[i+2][j
                  +1],molS[i+2][x],sem,num,indice)*(1+(Math.
                  cos((getN(molS[i][j],molS[i+1][j],molS[i
                  +2][j+1],molS[i+2][x],sem,indice)*angdc)-
                  getPeriU(molS[i][j],molS[i+1][j],molS[i+2][
                  j+1],molS[i+2][x],sem,indice)))));
              //arredondamento TINKER
              pangdc= (Math.round((pangdc * 10000.0))) /
                  10000.0;
              pangdc= pangdc+auxc;
              auxc= pangdc;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }else if(((i<qntdlinha-2)&&(j<qntdcol-1)&&(molS[i][j
          ]!="") && (molS[i+1][j]!="") && (molS[i+2][j
          +1]!="")&& (molS[i+2][j]!=""))&&
            (molS[i][j].substring(9, 10).equals("1"))&&(molS[i
                +1][j].substring(9, 10).equals("1"))&&(molS[i
                +1][j].substring(10, 11).equals("1"))&&(molS[i
                +2][j+1].substring(11, 12).equals("1"))&&
          (molS[i+2][j+1].substring(8, 9).equals("1"))&&(molS[
              i+2][j].substring(8, 9).equals("1"))&& (molS[i
              +2][j]!="X00000001000")){
        indice=0;
```

```
aux1c= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2c= objAtomos.setAtomoU(molS[i+1][j],torS[r]);
aux3c= objAtomos.setAtomoU(molS[i+2][j+1],torS[r]);
aux4c= objAtomos.setAtomoU(molS[i+2][j],torS[r]);
v1= objAtomos.getX(aux1c,torS[r])-objAtomos.getX(
   aux2c,torS[r]);
x1= objAtomos.getY(aux1c,torS[r])-objAtomos.getY(
   aux2c,torS[r]);
z1= objAtomos.getZ(aux1c,torS[r])-objAtomos.getZ(
   aux2c,torS[r]);
v2= objAtomos.getX(aux3c,torS[r])-objAtomos.getX(
   aux2c,torS[r]);
x2= objAtomos.getY(aux3c,torS[r])-objAtomos.getY(
   aux2c,torS[r]);
z2= objAtomos.getZ(aux3c,torS[r])-objAtomos.getZ(
   aux2c,torS[r]);
v3= objAtomos.getX(aux2c,torS[r])-objAtomos.getX(
   aux3c,torS[r]);
x3= objAtomos.getY(aux2c,torS[r])-objAtomos.getY(
   aux3c,torS[r]);
z3= objAtomos.getZ(aux2c,torS[r])-objAtomos.getZ(
   aux3c,torS[r]);
v4= objAtomos.getX(aux4c,torS[r])-objAtomos.getX(
   aux3c,torS[r]);
x4= objAtomos.getY(aux4c,torS[r])-objAtomos.getY(
   aux3c,torS[r]);
z4= objAtomos.getZ(aux4c,torS[r])-objAtomos.getZ(
   aux3c,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
```

```
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
auxdc= (prod)/(pmod);
double angdc= Math.acos(auxdc);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc= -angdc;
}
pangdc= (getVnU(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+2][j],sem,num,indice)/getPath(molS[i][
    j],molS[i+1][j],molS[i+2][j+1],molS[i+2][j],sem,
    num,indice)*(1+(Math.cos((getN(molS[i][j],molS[i
    +1][j],molS[i+2][j+1],molS[i+2][j],sem,indice)*
    angdc)-getPeriU(molS[i][j],molS[i+1][j],molS[i
    +2][j+1],molS[i+2][j],sem,indice)))));
//arredondamento TINKER
pangdc= (Math.round((pangdc * 10000.0))) / 10000.0;
pangdc= pangdc+auxc;
auxc= pangdc;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdc= (getVnU(molS[i][j],molS[i+1][j],molS[i+2][
      j+1],molS[i+2][j],sem,num,indice)/getPath(molS[
      i][j],molS[i+1][j],molS[i+2][j+1],molS[i+2][j],
      sem,num,indice)*(1+(Math.cos((getN(molS[i][j],
      molS[i+1][j],molS[i+2][j+1],molS[i+2][j],sem,
      indice)*angdc)-getPeriU(molS[i][j],molS[i+1][j
      ],molS[i+2][j+1],molS[i+2][j],sem,indice)))));
  //arredondamento TINKER
```

```
            pangdc= (Math.round((pangdc * 10000.0))) /
                10000.0;
            pangdc= pangdc+auxc;
            auxc= pangdc;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim if
        if((molS[i][j]!="") &&(i!=0)&&(j<qntdcol-1)&& (molS[i
            -1][j]!="") && (molS[i-1][j+1]!="") && (molS[i][j
            +1]!="")){
          //verificar se todo esta ligados
          if ((molS[i][j].substring(9, 10).equals("1"))&&(molS
              [i-1][j].substring(9, 10).equals("1"))&&(molS[i
              -1][j].substring(8, 9).equals("1"))&&(molS[i-1][j
              +1].substring(8, 9).equals("1"))&&
            (molS[i-1][j+1].substring(9, 10).equals("1"))&&(
                molS[i][j+1].substring(9, 10).equals("1"))){
            if ((molS[i-1][j].substring(0, 7).equals("OHR1001
                "))&&(molS[i-1][j+1].substring(0, 7).equals("
                OHR1002"))){

            }else{
              indice=0;
              aux1cc= objAtomos.setAtomoU(molS[i][j],torS[r]);
              aux2cc= objAtomos.setAtomoU(molS[i-1][j],torS[r
                  ]);
              aux3cc= objAtomos.setAtomoU(molS[i-1][j+1],torS[
                  r]);
              aux4cc= objAtomos.setAtomoU(molS[i][j+1],torS[r
                  ]);
              v1= objAtomos.getX(aux1cc,torS[r])-objAtomos.
                  getX(aux2cc,torS[r]);
              x1= objAtomos.getY(aux1cc,torS[r])-objAtomos.
                  getY(aux2cc,torS[r]);
```

```
z1= objAtomos.getZ(aux1cc,torS[r])-objAtomos.
    getZ(aux2cc,torS[r]);
v2= objAtomos.getX(aux3cc,torS[r])-objAtomos.
    getX(aux2cc,torS[r]);
x2= objAtomos.getY(aux3cc,torS[r])-objAtomos.
    getY(aux2cc,torS[r]);
z2= objAtomos.getZ(aux3cc,torS[r])-objAtomos.
    getZ(aux2cc,torS[r]);
v3= objAtomos.getX(aux2cc,torS[r])-objAtomos.
    getX(aux3cc,torS[r]);
x3= objAtomos.getY(aux2cc,torS[r])-objAtomos.
    getY(aux3cc,torS[r]);
z3= objAtomos.getZ(aux2cc,torS[r])-objAtomos.
    getZ(aux3cc,torS[r]);
v4= objAtomos.getX(aux4cc,torS[r])-objAtomos.
    getX(aux3cc,torS[r]);
x4= objAtomos.getY(aux4cc,torS[r])-objAtomos.
    getY(aux3cc,torS[r]);
z4= objAtomos.getZ(aux4cc,torS[r])-objAtomos.
    getZ(aux3cc,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdcc= (prod)/(pmod);
double angdcc= Math.acos(auxdcc);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
```

```
    angdcc= -angdcc;
  }
  pangdcc= ((getVnU(molS[i][j],molS[i-1][j],molS[i
      -1][j+1],molS[i][j+1],sem,num,indice)/getPath
      (molS[i][j],molS[i-1][j],molS[i-1][j+1],molS[
      i][j+1],sem,num,indice))*(1+(Math.cos((getN(
      molS[i][j],molS[i-1][j],molS[i-1][j+1],molS[i
      ][j+1],sem,indice)*angdcc)-getPeriU(molS[i][j
      ],molS[i-1][j],molS[i-1][j+1],molS[i][j+1],
      sem,indice)))));
  //arredondamento TINKER
  pangdcc= (Math.round((pangdcc * 10000.0))) /
      10000.0;
  pangdcc= pangdcc+auxcc;
  auxcc= pangdcc;
  indiceD= indiceD + 1;
  ind++;
  sem++;
  while (num==0){
    //transforma pra graus o cos
    pangdcc= ((getVnU(molS[i][j],molS[i-1][j],molS
        [i-1][j+1],molS[i][j+1],sem,num,indice)/
        getPath(molS[i][j],molS[i-1][j],molS[i-1][j
        +1],molS[i][j+1],sem,num,indice))*(1+(Math.
        cos((getN(molS[i][j],molS[i-1][j],molS[i
        -1][j+1],molS[i][j+1],sem,indice)*angdcc)-
        getPeriU(molS[i][j],molS[i-1][j],molS[i-1][
        j+1],molS[i][j+1],sem,indice)))));
    //arredondamento TINKER
    pangdcc= (Math.round((pangdcc * 10000.0))) /
        10000.0;
    pangdcc= pangdcc+auxcc;
    auxcc= pangdcc;
    indiceD= indiceD + 1;
    ind++;
    sem++;
```

```
            }
          }
        }
      }//fim if
      if((molS[i][j]!="") &&(j>=2)&&(j<qntdcol-1)&& (i<
          qntdlinha-1)&&(molS[i][j+1]!="") && (molS[i+1][j
          +1]!="") && (molS[i+1][j-2]!="")&&
          ((molS[i+1][j]=="X00000001000")&&(molS[i+1][j-1]=="
              X00000001000")))){
        int x=j-2;
        if((molS[i][j]!="") && (molS[i+1][j]!="") && (molS[i
            +2][j+1]!="") && (molS[i+2][x]!="")){
          if ((molS[i][j].substring(9, 10).equals("1"))&&(
              molS[i+1][j].substring(9, 10).equals("1")) && (
              molS[i+1][j].substring(10, 11).equals("1"))&&(
              molS[i+1][j+1].substring(11, 12).equals("1"))&&
                (molS[i+1][j+1].substring(8, 9).equals("1"))
                    &&(molS[i+1][x].substring(8, 9).equals("1")
                    )){
            indice=0;
            aux1c1= objAtomos.setAtomoU(molS[i][j],torS[r]);
            aux2c1= objAtomos.setAtomoU(molS[i][j+1],torS[r
                ]);
            aux3c1= objAtomos.setAtomoU(molS[i+1][j+1],torS[
                r]);
            aux4c1= objAtomos.setAtomoU(molS[i+1][x],torS[r
                ]);
            v1= objAtomos.getX(aux1c1,torS[r])-objAtomos.
                getX(aux2c1,torS[r]);
            x1= objAtomos.getY(aux1c1,torS[r])-objAtomos.
                getY(aux2c1,torS[r]);
            z1= objAtomos.getZ(aux1c1,torS[r])-objAtomos.
                getZ(aux2c1,torS[r]);
            v2= objAtomos.getX(aux3c1,torS[r])-objAtomos.
                getX(aux2c1,torS[r]);
```

```
x2= objAtomos.getY(aux3c1,torS[r])-objAtomos.
    getY(aux2c1,torS[r]);
z2= objAtomos.getZ(aux3c1,torS[r])-objAtomos.
    getZ(aux2c1,torS[r]);
v3= objAtomos.getX(aux2c1,torS[r])-objAtomos.
    getX(aux3c1,torS[r]);
x3= objAtomos.getY(aux2c1,torS[r])-objAtomos.
    getY(aux3c1,torS[r]);
z3= objAtomos.getZ(aux2c1,torS[r])-objAtomos.
    getZ(aux3c1,torS[r]);
v4= objAtomos.getX(aux4c1,torS[r])-objAtomos.
    getX(aux3c1,torS[r]);
x4= objAtomos.getY(aux4c1,torS[r])-objAtomos.
    getY(aux3c1,torS[r]);
z4= objAtomos.getZ(aux4c1,torS[r])-objAtomos.
    getZ(aux3c1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdc1= (prod)/(pmod);
angdc1= Math.acos(auxdc1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc1= -angdc1;
}
pangdc1= (getVnU(molS[i][j],molS[i][j+1],molS[i
    +1][j+1],molS[i+1][x],sem,num,indice)/getPath
```

```
            (molS[i][j],molS[i][j+1],molS[i+1][j+1],molS[
            i+1][x],sem,num,indice)*(1+(Math.cos((getN(
            molS[i][j],molS[i][j+1],molS[i+1][j+1],molS[i
            +1][x],sem,indice)*angdc1)-getPeriU(molS[i][j
            ],molS[i][j+1],molS[i+1][j+1],molS[i+1][x],
            sem,indice)))));
          //arredondamento TINKER
          pangdc1= (Math.round((pangdc1 * 10000.0))) /
              10000.0;
          pangdc1= pangdc1+auxc1;
          auxc= pangdc;
          indiceD= indiceD + 1;
          ind++;
          sem++;
          while (num==0){
            //transforma pra graus o cos
            pangdc1= (getVnU(molS[i][j],molS[i][j+1],molS[
                i+1][j+1],molS[i+1][x],sem,num,indice)/
                getPath(molS[i][j],molS[i][j+1],molS[i+1][j
                +1],molS[i+1][x],sem,num,indice)*(1+(Math.
                cos((getN(molS[i][j],molS[i][j+1],molS[i
                +1][j+1],molS[i+1][x],sem,indice)*angdc1)-
                getPeriU(molS[i][j],molS[i][j+1],molS[i+1][
                j+1],molS[i+1][x],sem,indice)))));
            //arredondamento TINKER
            pangdc1= (Math.round((pangdc1 * 10000.0))) /
                10000.0;
            pangdc1= pangdc1+auxc1;
            auxc= pangdc;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }
```

```
}else if ((molS[i][j]!="") &&(j<qntdcol-1)&& (i<
    qntdlinha-1)&&(molS[i][j+1]!="") && (molS[i+1][j
    +1]!="") && (molS[i+1][j]!="")&&(molS[i][j].
    substring(8, 9).equals("1"))&&(molS[i][j+1].
    substring(8, 9).equals("1"))&&(molS[i][j+1].
    substring(9, 10).equals("1"))&&(molS[i+1][j+1].
    substring(9, 10).equals("1"))&&
     (molS[i+1][j+1].substring(8, 9).equals("1"))&&(
        molS[i+1][j].substring(8, 9).equals("1"))&& (
        molS[i+1][j]!="X00000001000")){
   if ((molS[i+1][j+1].substring(0,7).equals("OHR1002")
      )&&(molS[i+1][j].substring(0,7).equals("OHR1001")
      )){

   }else{
        indice=0;
        aux1c1= objAtomos.setAtomoU(molS[i][j],torS[r]);
        aux2c1= objAtomos.setAtomoU(molS[i][j+1],torS[r
            ]);
        aux3c1= objAtomos.setAtomoU(molS[i+1][j+1],torS[
            r]);
        aux4c1= objAtomos.setAtomoU(molS[i+1][j],torS[r
            ]);
        v1= objAtomos.getX(aux1c1,torS[r])-objAtomos.
            getX(aux2c1,torS[r]);
        x1= objAtomos.getY(aux1c1,torS[r])-objAtomos.
            getY(aux2c1,torS[r]);
        z1= objAtomos.getZ(aux1c1,torS[r])-objAtomos.
            getZ(aux2c1,torS[r]);
        v2= objAtomos.getX(aux3c1,torS[r])-objAtomos.
            getX(aux2c1,torS[r]);
        x2= objAtomos.getY(aux3c1,torS[r])-objAtomos.
            getY(aux2c1,torS[r]);
        z2= objAtomos.getZ(aux3c1,torS[r])-objAtomos.
            getZ(aux2c1,torS[r]);
```

```
v3= objAtomos.getX(aux2c1,torS[r])-objAtomos.
    getX(aux3c1,torS[r]);
x3= objAtomos.getY(aux2c1,torS[r])-objAtomos.
    getY(aux3c1,torS[r]);
z3= objAtomos.getZ(aux2c1,torS[r])-objAtomos.
    getZ(aux3c1,torS[r]);
v4= objAtomos.getX(aux4c1,torS[r])-objAtomos.
    getX(aux3c1,torS[r]);
x4= objAtomos.getY(aux4c1,torS[r])-objAtomos.
    getY(aux3c1,torS[r]);
z4= objAtomos.getZ(aux4c1,torS[r])-objAtomos.
    getZ(aux3c1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdc1= (prod)/(pmod);
angdc1= Math.acos(auxdc1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdc1= -angdc1;
}
pangdc1= ((getVnU(molS[i][j],molS[i][j+1],molS[i
    +1][j+1],molS[i+1][j],sem,num,indice)/getPath
    (molS[i][j],molS[i][j+1],molS[i+1][j+1],molS[
    i+1][j],sem,num,indice))*(1+(Math.cos((getN(
    molS[i][j],molS[i][j+1],molS[i+1][j+1],molS[i
    +1][j],sem,indice)*angdc1)-getPeriU(molS[i][j
```

```
                ],molS[i][j+1],molS[i+1][j+1],molS[i+1][j],
                sem,indice)))));
            //arredondamento TINKER
            pangdc1= (Math.round((pangdc1 * 10000.0))) /
                10000.0;
            pangdc1= pangdc1+auxc1;
            auxc1= pangdc1;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdc1= ((getVnU(molS[i][j],molS[i][j+1],molS
                  [i+1][j+1],molS[i+1][j],sem,num,indice)/
                  getPath(molS[i][j],molS[i][j+1],molS[i+1][j
                  +1],molS[i+1][j],sem,num,indice))*(1+(Math.
                  cos((getN(molS[i][j],molS[i][j+1],molS[i
                  +1][j+1],molS[i+1][j],sem,indice)*angdc1)-
                  getPeriU(molS[i][j],molS[i][j+1],molS[i+1][
                  j+1],molS[i+1][j],sem,indice)))));
              //arredondamento TINKER
              pangdc1= (Math.round((pangdc1 * 10000.0))) /
                  10000.0;
              pangdc1= pangdc1+auxc1;
              auxc1= pangdc1;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }//fim if
      if((i<qntdlinha-1)&&(j<qntdcol-2)&&(molS[i][j]!="") && (
          molS[i+1][j+1]!="") && (molS[i+1][j+2]!="")&& (molS[i
          ][j+2]!="")&&
            ((molS[i][j].substring(10, 11).equals("1"))&&(molS[i
                +1][j+1].substring(11, 12).equals("1"))&&(molS[i
```

```
    +1][j+1].substring(8, 9).equals("1"))&&(molS[i
    +1][j+2].substring(8, 9).equals("1"))&&
(molS[i+1][j+2].substring(9, 10).equals("1"))&&(molS
    [i][j+2].substring(9, 10).equals("1")))){
indice=0;
aux1cc1= objAtomos.setAtomoU(molS[i][j],torS[r]);
aux2cc1= objAtomos.setAtomoU(molS[i+1][j+1],torS[r])
    ;
aux3cc1= objAtomos.setAtomoU(molS[i+1][j+2],torS[r])
    ;
aux4cc1= objAtomos.setAtomoU(molS[i][j+2],torS[r]);
v1= objAtomos.getX(aux1cc1,torS[r])-objAtomos.getX(
    aux2cc1,torS[r]);
x1= objAtomos.getY(aux1cc1,torS[r])-objAtomos.getY(
    aux2cc1,torS[r]);
z1= objAtomos.getZ(aux1cc1,torS[r])-objAtomos.getZ(
    aux2cc1,torS[r]);
v2= objAtomos.getX(aux3cc1,torS[r])-objAtomos.getX(
    aux2cc1,torS[r]);
x2= objAtomos.getY(aux3cc1,torS[r])-objAtomos.getY(
    aux2cc1,torS[r]);
z2= objAtomos.getZ(aux3cc1,torS[r])-objAtomos.getZ(
    aux2cc1,torS[r]);
v3= objAtomos.getX(aux2cc1,torS[r])-objAtomos.getX(
    aux3cc1,torS[r]);
x3= objAtomos.getY(aux2cc1,torS[r])-objAtomos.getY(
    aux3cc1,torS[r]);
z3= objAtomos.getZ(aux2cc1,torS[r])-objAtomos.getZ(
    aux3cc1,torS[r]);
v4= objAtomos.getX(aux4cc1,torS[r])-objAtomos.getX(
    aux3cc1,torS[r]);
x4= objAtomos.getY(aux4cc1,torS[r])-objAtomos.getY(
    aux3cc1,torS[r]);
z4= objAtomos.getZ(aux4cc1,torS[r])-objAtomos.getZ(
    aux3cc1,torS[r]);
a1= ((x1*z2)-(x2*z1));
```

```
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
auxdcc1= (prod)/(pmod);
angdcc1= Math.acos(auxdcc1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdcc1= -angdcc1;
}
pangdcc1= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
    +1][j+2],molS[i][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i][
    j+2],sem,num,indice)*(1+(Math.cos((getN(molS[i][j
    ],molS[i+1][j+1],molS[i+1][j+2],molS[i][j+2],sem,
    indice)*angdcc1)-getPeriU(molS[i][j],molS[i+1][j
    +1],molS[i+1][j+2],molS[i][j+2],sem,indice)))));
//arredondamento TINKER
pangdcc1= (Math.round((pangdcc1 * 10000.0))) /
    10000.0;
pangdcc1= pangdcc1+auxcc1;
auxcc1= pangdcc1;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdcc1= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
      +1][j+2],molS[i][j+2],sem,num,indice)/getPath(
```

```
            molS[i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i
            ][j+2],sem,num,indice)*(1+(Math.cos((getN(molS[
            i][j],molS[i+1][j+1],molS[i+1][j+2],molS[i][j
            +2],sem,indice)*angdcc1)-getPeriU(molS[i][j],
            molS[i+1][j+1],molS[i+1][j+2],molS[i][j+2],sem,
            indice)))));
        //arredondamento TINKER
        pangdcc1= (Math.round((pangdcc1 * 10000.0))) /
            10000.0;
        pangdcc1= pangdcc1+auxcc1;
        auxcc1= pangdcc1;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }//fim if
    //Algumas regras a mais para os hidrogenios
    if((i<qntdlinha-2)&&(j<qntdcol-2)&&(molS[i][j]!="") &&
        (molS[i+1][j]!="") && (molS[i+2][j+1]!="")&& (molS
       [i+2][j+2]!="")&&
      ((molS[i][j].substring(9, 10).equals("1"))&&(molS[i
          +1][j].substring(9, 10).equals("1"))&&(molS[i+1][
          j].substring(10, 11).equals("1"))&&(molS[i+2][j
          +1].substring(11, 12).equals("1"))&&
      (molS[i+2][j+1].substring(8, 9).equals("1"))&&(molS[
          i+2][j+2].substring(8, 9).equals("1")))){
      indice=0;
      double[] aux1dir = objAtomos.setAtomoU(molS[i][j],
          torS[r]);
      double[] aux2dir = objAtomos.setAtomoU(molS[i+1][j],
          torS[r]);
      double[] aux3dir = objAtomos.setAtomoU(molS[i+2][j
          +1],torS[r]);
      double[] aux4dir = objAtomos.setAtomoU(molS[i+2][j
          +2],torS[r]);
```

```
v1= objAtomos.getX(aux1dir,torS[r])-objAtomos.getX(
    aux2dir,torS[r]);
x1= objAtomos.getY(aux1dir,torS[r])-objAtomos.getY(
    aux2dir,torS[r]);
z1= objAtomos.getZ(aux1dir,torS[r])-objAtomos.getZ(
    aux2dir,torS[r]);
v2= objAtomos.getX(aux3dir,torS[r])-objAtomos.getX(
    aux2dir,torS[r]);
x2= objAtomos.getY(aux3dir,torS[r])-objAtomos.getY(
    aux2dir,torS[r]);
z2= objAtomos.getZ(aux3dir,torS[r])-objAtomos.getZ(
    aux2dir,torS[r]);
v3= objAtomos.getX(aux2dir,torS[r])-objAtomos.getX(
    aux3dir,torS[r]);
x3= objAtomos.getY(aux2dir,torS[r])-objAtomos.getY(
    aux3dir,torS[r]);
z3= objAtomos.getZ(aux2dir,torS[r])-objAtomos.getZ(
    aux3dir,torS[r]);
v4= objAtomos.getX(aux4dir,torS[r])-objAtomos.getX(
    aux3dir,torS[r]);
x4= objAtomos.getY(aux4dir,torS[r])-objAtomos.getY(
    aux3dir,torS[r]);
z4= objAtomos.getZ(aux4dir,torS[r])-objAtomos.getZ(
    aux3dir,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math.
    pow(c2,2));
pmod= mod*mod2;
```

```
double auxddir = (prod)/(pmod);
double angddir = Math.acos(auxddir);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angddir= -angddir;
}
pangddir = (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j+1],molS[i+2][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j],molS[i+2][j+1],molS[i+2][
    j+2],sem,num,indice)*(1+(Math.cos((getN(molS[i][j
    ],molS[i+1][j],molS[i+2][j+1],molS[i+2][j+2],sem,
    indice)*angddir)-getPeriU(molS[i][j],molS[i+1][j
    ],molS[i+2][j+1],molS[i+2][j+2],sem,indice)))));
//arredondamento TINKER
pangddir= (Math.round((pangddir * 10000.0))) /
    10000.0;
pangddir= pangddir+auxdir;
auxdir= pangddir;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangddir= (getVnU(molS[i][j],molS[i+1][j],molS[i
      +2][j+1],molS[i+2][j+2],sem,num,indice)/getPath
      (molS[i][j],molS[i+1][j],molS[i+2][j+1],molS[i
      +2][j+2],sem,num,indice)*(1+(Math.cos((getN(
      molS[i][j],molS[i+1][j],molS[i+2][j+1],molS[i
      +2][j+2],sem,indice)*angddir)-getPeriU(molS[i][
      j],molS[i+1][j],molS[i+2][j+1],molS[i+2][j+2],
      sem,indice)))));
  //arredondamento TINKER
  pangddir= (Math.round((pangddir * 10000.0))) /
      10000.0;
  pangddir= pangddir+auxdir;
  auxdir= pangddir;
```

```
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim if
      }//fim else

    }//fim for
    }//fim for
    pangd[r]= pangdebb3+pangdeb3+pangdbbe+pangdbe+ pangdd+
        pangdb3+ pangdbb3+pangdl+pangdll+ pangdc+pangdcc+
        pangdb+pangdbb+ pangdc1+pangdcc1+ pangdb1+ pangdbb1+
        pangdc2+
    pangdcc2+pangdbb2+pangddir+pangddir1+pangdb2;
    }//fim for r

    return pangd;

  }//fim angulo diedral
  //excedeu o limite de linhas do metodo
  public double[] angd3U (int semente){
    double[] pangd= new double[torS.length];
    double pangdp,pangdz,pangdz1,pangdz2,pangdu,pangdu2,
        pangdu3,pangdu4,pangdu5,pangdz3,pangdz4,pangdz5,pangdb4
        ,pangdbb4,pangdb5,pangdbb5,pangdb6;
    double angdp,angdb4,angdbb4,angdb5,angdbb5,angdb6,angdz,
        angdz2,angdz3,angdz4,angdz5,angdu,angdu2,angdu3,angdu4,
        angdu5;
    double auxdp,auxdb4,auxdbb4,auxdb5,auxdbb5,auxdb6,auxdz,
        auxdz2,auxdz3,auxdz4,auxdz5,auxdu,auxdu2,auxdu3,auxdu4,
        auxdu5;
    double auxp,auxb4,auxbb4,auxb5,auxbb5,auxb6,auxz,auxz1,
        auxz2,auxz3,auxz4,auxz5,auxu,auxu2,auxu3,auxu4,auxu5;
    int indiceD= 0;
    double a1, a2, b1, b2, c1, c2, v1, x1, z1, v2, x2, z2, v3,
        x3, z3, v4, x4, z4, x1s, y1s, z1s;//, x2s, y2s, z2s;
```

```
double aux = 0.0;
double[] aux1b4, aux2b4, aux3b4, aux4b4, aux1bb4, aux2bb4,
    aux3bb4, aux4bb4, aux1b5, aux2b5, aux3b5, aux4b5,
   aux1bb5, aux2bb5, aux3bb5, aux4bb5, aux1b6, aux2b6,
   aux3b6, aux4b6;
double pmod, mod, mod2, prod;
int ind= 0;

sem= semente;
num = 1;
indice = 0;

String log3="";
double energia=0.0;
for(int r= 0; r< torS.length; r++){
  aux1b4= new double[3];
  aux2b4= new double[3];
  aux3b4= new double[3];
  aux4b4= new double[3];
  aux1bb4= new double[3];
  aux2bb4= new double[3];
  aux3bb4= new double[3];
  aux4bb4= new double[3];
  aux1b5= new double[3];
  aux2b5= new double[3];
  aux3b5= new double[3];
  aux4b5= new double[3];
  aux1b6= new double[3];
  aux2b6= new double[3];
  aux3b6= new double[3];
  aux4b6= new double[3];
  aux1bb5= new double[3];
  aux2bb5= new double[3];
  aux3bb5= new double[3];
  aux4bb5= new double[3];
  pmod= 0;
```

```
mod= 0;
mod2= 0;
prod=0;
auxz=0;
auxz2=0;
auxz3=0;
auxz4=0;
auxz5=0;
auxu=0;
auxp=0.0;
angdp=0.0;
auxdp=0.0;
pangdp=0;
auxu2=0;
auxu3=0;
auxu4=0;
auxu5=0;
pangdb4=0.0;
pangdbb4=0.0;
pangdb5=0.0;
pangdbb5=0.0;
pangdb6=0.0;
auxdb4=0.0;
auxdbb4=0.0;
auxdb5=0.0;
auxdbb5=0.0;
auxdb6=0.0;
angdb4=0.0;
angdbb4=0.0;
angdb5=0.0;
angdbb5=0.0;
angdb6=0.0;
auxb4=0.0;
auxbb4=0.0;
auxz1=0.0;
auxb5=0.0;
```

```
    auxbb5=0.0;
    auxb6=0.0;
    pangdz=0;
    pangdz1=0;
    pangdz2=0;
    pangdz3=0;
    pangdz4=0;
    pangdz5=0;
    pangdu=0;
    pangdu2=0;
    pangdu3=0;
    pangdu4=0;
    pangdu5=0;
  for (int i=0; i<qntdlinha; i++){
    for (int j= 0; j< qntdcol; j++){
      aux=0.0;
      if (molS[i][j].equals("X00000001000")){
        //nao quero que comece com X
      }else{
        if(((i<qntdlinha-3)&&(j<qntdcol-1)&&(molS[i][j]!="")
            && (molS[i+1][j]!="") && (molS[i+2][j+1]!="") &&
            (molS[i+3][j+1]!=""))){
            if ((((molS[i][j].substring(9, 10).equals("1"))
               &&(molS[i+1][j].substring(9, 10).equals("1"))
               ))&&(molS[i+1][j].substring(10, 11).equals
               ("1"))&&(molS[i+2][j+1].substring(11, 12).
               equals("1"))&&
                 (molS[i+2][j+1].substring(9, 10).equals("1")
                    )&(molS[i+3][j+1].substring(9, 10).equals
                    ("1"))){
            indice=0;
            //isolar a b e c dos planos
            aux1b4= objAtomos.setAtomoU(molS[i][j],torS[r]);
            aux2b4= objAtomos.setAtomoU(molS[i+1][j],torS[r
                ]);
```

```
aux3b4= objAtomos.setAtomoU(molS[i+2][j+1],torS[
    r]);
aux4b4= objAtomos.setAtomoU(molS[i+3][j+1],torS[
    r]);
v1= objAtomos.getX(aux1b4,torS[r])-objAtomos.
    getX(aux2b4,torS[r]);
x1= objAtomos.getY(aux1b4,torS[r])-objAtomos.
    getY(aux2b4,torS[r]);
z1= objAtomos.getZ(aux1b4,torS[r])-objAtomos.
    getZ(aux2b4,torS[r]);
v2= objAtomos.getX(aux3b4,torS[r])-objAtomos.
    getX(aux2b4,torS[r]);
x2= objAtomos.getY(aux3b4,torS[r])-objAtomos.
    getY(aux2b4,torS[r]);
z2= objAtomos.getZ(aux3b4,torS[r])-objAtomos.
    getZ(aux2b4,torS[r]);
v3= objAtomos.getX(aux2b4,torS[r])-objAtomos.
    getX(aux3b4,torS[r]);
x3= objAtomos.getY(aux2b4,torS[r])-objAtomos.
    getY(aux3b4,torS[r]);
z3= objAtomos.getZ(aux2b4,torS[r])-objAtomos.
    getZ(aux3b4,torS[r]);
v4= objAtomos.getX(aux4b4,torS[r])-objAtomos.
    getX(aux3b4,torS[r]);
x4= objAtomos.getY(aux4b4,torS[r])-objAtomos.
    getY(aux3b4,torS[r]);
z4= objAtomos.getZ(aux4b4,torS[r])-objAtomos.
    getZ(aux3b4,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
```

```
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb4= (prod)/(pmod);
angdb4= Math.acos(auxdb4);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb4= -angdb4;
}
pangdb4= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j+1],molS[i+3][j+1],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+3][j+1],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+3][j+1],sem,indice)*angdb4)-
    getPeriU(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+3][j+1],sem,indice)))));
//arredondamento TINKER
pangdb4= (Math.round((pangdb4 * 10000.0))) /
    10000.0;
pangdb4= pangdb4+auxb4;
auxb4= pangdb4;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdb4= (getVnU(molS[i][j],molS[i+1][j],molS[
      i+2][j+1],molS[i+3][j+1],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j],molS[i+2][j
      +1],molS[i+3][j+1],sem,num,indice)*(1+(Math
      .cos((getN(molS[i][j],molS[i+1][j],molS[i
      +2][j+1],molS[i+3][j+1],sem,indice)*angdb4)
      -getPeriU(molS[i][j],molS[i+1][j],molS[i
```

```
            +2][j+1],molS[i+3][j+1],sem,indice)))));
          //arredondamento TINKER
          pangdb4= (Math.round((pangdb4 * 10000.0))) /
              10000.0;
          pangdb4= pangdb4+auxb4;
          auxb4= pangdb4;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim if
    }//fim if
    if(((i<qntdlinha-2)&&(j<qntdcol-1)&&(molS[i][j]!="")
        && (molS[i+1][j+1]!="") && (molS[i+2][j+1]!="")
       && (molS[i+2][j]!=""))){
      if ((((molS[i][j].substring(10, 11).equals("1"))
          &&(molS[i+1][j+1].substring(11, 12).equals("1")
          )))&&(molS[i+1][j+1].substring(9, 10).equals
          ("1"))&&(molS[i+2][j+1].substring(9, 10).equals
          ("1"))&&
          (molS[i+2][j+1].substring(8, 9).equals("1"))&(
              molS[i+2][j].substring(8, 9).equals("1"))){
          indice=0;
          //isolar a b e c dos planos
          aux1bb4= objAtomos.setAtomoU(molS[i][j],torS[r])
              ;
          aux2bb4= objAtomos.setAtomoU(molS[i+1][j+1],torS
              [r]);
          aux3bb4= objAtomos.setAtomoU(molS[i+2][j+1],torS
              [r]);
          aux4bb4= objAtomos.setAtomoU(molS[i+2][j],torS[r
              ]);
          v1= objAtomos.getX(aux1bb4,torS[r])-objAtomos.
              getX(aux2bb4,torS[r]);
          x1= objAtomos.getY(aux1bb4,torS[r])-objAtomos.
              getY(aux2bb4,torS[r]);
```

```
z1= objAtomos.getZ(aux1bb4,torS[r])-objAtomos.
    getZ(aux2bb4,torS[r]);
v2= objAtomos.getX(aux3bb4,torS[r])-objAtomos.
    getX(aux2bb4,torS[r]);
x2= objAtomos.getY(aux3bb4,torS[r])-objAtomos.
    getY(aux2bb4,torS[r]);
z2= objAtomos.getZ(aux3bb4,torS[r])-objAtomos.
    getZ(aux2bb4,torS[r]);
v3= objAtomos.getX(aux2bb4,torS[r])-objAtomos.
    getX(aux3bb4,torS[r]);
x3= objAtomos.getY(aux2bb4,torS[r])-objAtomos.
    getY(aux3bb4,torS[r]);
z3= objAtomos.getZ(aux2bb4,torS[r])-objAtomos.
    getZ(aux3bb4,torS[r]);
v4= objAtomos.getX(aux4bb4,torS[r])-objAtomos.
    getX(aux3bb4,torS[r]);
x4= objAtomos.getY(aux4bb4,torS[r])-objAtomos.
    getY(aux3bb4,torS[r]);
z4= objAtomos.getZ(aux4bb4,torS[r])-objAtomos.
    getZ(aux3bb4,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb4= (prod)/(pmod);
angdbb4= Math.acos(auxdbb4);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
```

```
    angdbb4= -angdbb4;
  }
  pangdbb4= (getVnU(molS[i][j],molS[i+1][j+1],molS
      [i+2][j+1],molS[i+2][j],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j+1],molS[i+2][j
      +1],molS[i+2][j],sem,num,indice)*(1+(Math.cos
      ((getN(molS[i][j],molS[i+1][j+1],molS[i+2][j
      +1],molS[i+2][j],sem,indice)*angdbb4)-
      getPeriU(molS[i][j],molS[i+1][j+1],molS[i+2][
      j+1],molS[i+2][j],sem,indice)))));
  //arredondamento TINKER
  pangdbb4= (Math.round((pangdbb4 * 10000.0))) /
      10000.0;
  pangdbb4= pangdbb4+auxbb4;
  auxbb4= pangdbb4;
  indiceD= indiceD + 1;
  ind++;
  sem++;
  while (num==0){
    //transforma pra graus o cos
    pangdbb4= (getVnU(molS[i][j],molS[i+1][j+1],
        molS[i+2][j+1],molS[i+2][j],sem,num,indice)
        /getPath(molS[i][j],molS[i+1][j+1],molS[i
        +2][j+1],molS[i+2][j],sem,num,indice)*(1+(
        Math.cos((getN(molS[i][j],molS[i+1][j+1],
        molS[i+2][j+1],molS[i+2][j],sem,indice)*
        angdbb4)-getPeriU(molS[i][j],molS[i+1][j
        +1],molS[i+2][j+1],molS[i+2][j],sem,indice)
        ))));
    //arredondamento TINKER
    pangdbb4= (Math.round((pangdbb4 * 10000.0))) /
        10000.0;
    pangdbb4= pangdbb4+auxbb4;
    auxbb4= pangdbb4;
    indiceD= indiceD + 1;
    ind++;
```

```
            sem++;
          }
        }//fim if
      }//fim if
      if(((i<qntdlinha-2)&&(j<qntdcol-2)&&(molS[i][j]!="")
          && (molS[i+1][j+1]!="") && (molS[i+2][j+1]!="")
         && (molS[i+2][j+2]!=""))){
        if (((((molS[i][j].substring(10, 11).equals("1"))
           &&(molS[i+1][j+1].substring(11, 12).equals("1")
           )))&&(molS[i+1][j+1].substring(9, 10).equals
           ("1"))&&(molS[i+2][j+1].substring(9, 10).equals
           ("1"))&&
          (molS[i+2][j+1].substring(8, 9).equals("1"))&(
              molS[i+2][j+2].substring(8, 9).equals("1"))){
          if ((molS[i+2][j+1].substring(0, 7).equals("
              OHR1001"))&(molS[i+2][j+2].substring(0, 7).
              equals("OHR1002"))){

          }else{
            indice=0;
            //isolar a b e c dos planos
            aux1b5= objAtomos.setAtomoU(molS[i][j],torS[r
                ]);
            aux2b5= objAtomos.setAtomoU(molS[i+1][j+1],
                torS[r]);
            aux3b5= objAtomos.setAtomoU(molS[i+2][j+1],
                torS[r]);
            aux4b5= objAtomos.setAtomoU(molS[i+2][j+2],
                torS[r]);
            v1= objAtomos.getX(aux1b5,torS[r])-objAtomos.
                getX(aux2b5,torS[r]);
            x1= objAtomos.getY(aux1b5,torS[r])-objAtomos.
                getY(aux2b5,torS[r]);
            z1= objAtomos.getZ(aux1b5,torS[r])-objAtomos.
                getZ(aux2b5,torS[r]);
```

```
v2= objAtomos.getX(aux3b5,torS[r])-objAtomos.
    getX(aux2b5,torS[r]);
x2= objAtomos.getY(aux3b5,torS[r])-objAtomos.
    getY(aux2b5,torS[r]);
z2= objAtomos.getZ(aux3b5,torS[r])-objAtomos.
    getZ(aux2b5,torS[r]);
v3= objAtomos.getX(aux2b5,torS[r])-objAtomos.
    getX(aux3b5,torS[r]);
x3= objAtomos.getY(aux2b5,torS[r])-objAtomos.
    getY(aux3b5,torS[r]);
z3= objAtomos.getZ(aux2b5,torS[r])-objAtomos.
    getZ(aux3b5,torS[r]);
v4= objAtomos.getX(aux4b5,torS[r])-objAtomos.
    getX(aux3b5,torS[r]);
x4= objAtomos.getY(aux4b5,torS[r])-objAtomos.
    getY(aux3b5,torS[r]);
z4= objAtomos.getZ(aux4b5,torS[r])-objAtomos.
    getZ(aux3b5,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb5= (prod)/(pmod);
angdb5= Math.acos(auxdb5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb5= -angdb5;
}
```

```
pangdb5= (getVnU(molS[i][j],molS[i+1][j+1],
    molS[i+2][j+1],molS[i+2][j+2],sem,num,
    indice)/getPath(molS[i][j],molS[i+1][j+1],
    molS[i+2][j+1],molS[i+2][j+2],sem,num,
    indice)*(1+(Math.cos((getN(molS[i][j],molS[
    i+1][j+1],molS[i+2][j+1],molS[i+2][j+2],sem
    ,indice)*angdb5)-getPeriU(molS[i][j],molS[i
    +1][j+1],molS[i+2][j+1],molS[i+2][j+2],sem,
    indice)))));
//arredondamento TINKER
pangdb5= (Math.round((pangdb5 * 10000.0))) /
    10000.0;
pangdb5= pangdb5+auxb5;
auxb5= pangdb5;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdb5= (getVnU(molS[i][j],molS[i+1][j+1],
      molS[i+2][j+1],molS[i+2][j+2],sem,num,
      indice)/getPath(molS[i][j],molS[i+1][j
      +1],molS[i+2][j+1],molS[i+2][j+2],sem,num
      ,indice)*(1+(Math.cos((getN(molS[i][j],
      molS[i+1][j+1],molS[i+2][j+1],molS[i+2][j
      +2],sem,indice)*angdb5)-getPeriU(molS[i][
      j],molS[i+1][j+1],molS[i+2][j+1],molS[i
      +2][j+2],sem,indice)))));
  //arredondamento TINKER
  pangdb5= (Math.round((pangdb5 * 10000.0))) /
       10000.0;
  pangdb5= pangdb5+auxb5;
  auxb5= pangdb5;
  indiceD= indiceD + 1;
  ind++;
  sem++;
```

```
                }
              }//fim else
            }//fim if
          }//fim if
          if(((j<qntdcol-1)&&(i<qntdlinha-2)&&(molS[i][j]!="")
              && (j>=2)&&(molS[i+1][j+1]!="") && (molS[i+1][j
             -2]!="") && (molS[i+2][j-2]!=""))&&

            //esses if sao caso encontrem o valor XX, ai pula
               duas casas (verificar se esta certo)
            ((molS[i+1][j]=="X00000001000")&&(molS[i+1][j
               -1]=="X00000001000"))){
            int x=j-2;
            if((x>=2)&&(molS[i][j]!="") &&(molS[i+1][j+1]!="")
                && (molS[i+1][x]!="") && (molS[i+2][x]!="")){
              if ((molS[i][j].substring(10, 11).equals("1"))
                 &&(molS[i+1][j+1].substring(11, 12).equals
                 ("1"))&&(molS[i+1][j+1].substring(8, 9).
                 equals("1"))&&(molS[i+1][x].substring(8, 9).
                 equals("1")) &&(molS[i+1][x].substring(9, 10)
                 .equals("1"))&&(molS[i+2][x].substring(9, 10)
                 .equals("1"))){
                indice=0;
                aux1b5= objAtomos.setAtomoU(molS[i][j],torS[r
                   ]);
                aux2b5= objAtomos.setAtomoU(molS[i+1][j+1],
                   torS[r]);
                aux3b5= objAtomos.setAtomoU(molS[i+1][x],torS[
                   r]);
                aux4b5= objAtomos.setAtomoU(molS[i+2][x],torS[
                   r]);
                v1= objAtomos.getX(aux1b5,torS[r])-objAtomos.
                   getX(aux2b5,torS[r]);
                x1= objAtomos.getY(aux1b5,torS[r])-objAtomos.
                   getY(aux2b5,torS[r]);
```

```
z1= objAtomos.getZ(aux1b5,torS[r])-objAtomos.
    getZ(aux2b5,torS[r]);
v2= objAtomos.getX(aux3b5,torS[r])-objAtomos.
    getX(aux2b5,torS[r]);
x2= objAtomos.getY(aux3b5,torS[r])-objAtomos.
    getY(aux2b5,torS[r]);
z2= objAtomos.getZ(aux3b5,torS[r])-objAtomos.
    getZ(aux2b5,torS[r]);
v3= objAtomos.getX(aux2b5,torS[r])-objAtomos.
    getX(aux3b5,torS[r]);
x3= objAtomos.getY(aux2b5,torS[r])-objAtomos.
    getY(aux3b5,torS[r]);
z3= objAtomos.getZ(aux2b5,torS[r])-objAtomos.
    getZ(aux3b5,torS[r]);
v4= objAtomos.getX(aux4b5,torS[r])-objAtomos.
    getX(aux3b5,torS[r]);
x4= objAtomos.getY(aux4b5,torS[r])-objAtomos.
    getY(aux3b5,torS[r]);
z4= objAtomos.getZ(aux4b5,torS[r])-objAtomos.
    getZ(aux3b5,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb5= (prod)/(pmod);
angdb5= Math.acos(auxdb5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
```

```
    angdb5= -angdb5;
  }
  //transforma pra graus o cos
  pangdb5= (getVnU(molS[i][j],molS[i+1][j+1],
      molS[i+1][x],molS[i+2][x],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j+1],molS[i
      +1][x],molS[i+2][x],sem,num,indice)*(1+(
      Math.cos((getN(molS[i][j],molS[i+1][j+1],
      molS[i+1][x],molS[i+2][x],sem,indice)*
      angdb5)-getPeriU(molS[i][j],molS[i+1][j+1],
      molS[i+1][x],molS[i+2][x],sem,indice)))));
  //arredondamento TINKER
  pangdb5= (Math.round((pangdb5 * 10000.0))) /
      10000.0;
  pangdb5= pangdb5+auxb5;
  auxb5= pangdb5;
  indiceD= indiceD + 1;
  ind++;
  sem++;
  while (num==0){
  //transforma pra graus o cos
    pangdb5= (getVnU(molS[i][j],molS[i+1][j+1],
        molS[i+1][x],molS[i+2][x],sem,num,indice)
        /getPath(molS[i][j],molS[i+1][j+1],molS[i
        +1][x],molS[i+2][x],sem,num,indice)*(1+(
        Math.cos((getN(molS[i][j],molS[i+1][j+1],
        molS[i+1][x],molS[i+2][x],sem,indice)*
        angdb5)-getPeriU(molS[i][j],molS[i+1][j
        +1],molS[i+1][x],molS[i+2][x],sem,indice)
        ))));
    //arredondamento TINKER
    pangdb5= (Math.round((pangdb5 * 10000.0))) /
        10000.0;
    pangdb5= pangdb5+auxb5;
    auxb5= pangdb5;
    indiceD= indiceD + 1;
```

```
                ind++;
                sem++;
              }
            }
          }
        }//fim if
        if(((j<qntdcol-2)&&(i<qntdlinha-1)&&(molS[i][j]!="")
            && (molS[i][j+1]!="") && (molS[i+1][j+1]!="") &&
            (molS[i+1][j+2]!="")))){
          if ((molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(9, 10).equals("1"))&&(
              molS[i+1][j+1].substring(9, 10).equals("1"))&&(
              molS[i+1][j+1].substring(8, 9).equals("1"))&&(
              molS[i+1][j+2].substring(8, 9).equals("1"))){
            if (((molS[i][j+1].substring(0, 7).equals("
                OHR2001"))&&(molS[i+1][j+1].substring(0,7).
                equals("P1P0001")))||((molS[i+1][j+1].
                substring(0, 7).equals("OHR1001"))&&(molS[i
                +1][j+2].substring(0, 7).equals("OHR1002"))))
                {

            }else{
              indice=0;
              aux1bb5= objAtomos.setAtomoU(molS[i][j],torS[r
                  ]);
              aux2bb5= objAtomos.setAtomoU(molS[i][j+1],torS
                  [r]);
              aux3bb5= objAtomos.setAtomoU(molS[i+1][j+1],
                  torS[r]);
              aux4bb5= objAtomos.setAtomoU(molS[i+1][j+2],
                  torS[r]);
              v1= objAtomos.getX(aux1bb5,torS[r])-objAtomos.
                  getX(aux2bb5,torS[r]);
              x1= objAtomos.getY(aux1bb5,torS[r])-objAtomos.
                  getY(aux2bb5,torS[r]);
```

```
z1= objAtomos.getZ(aux1bb5,torS[r])-objAtomos.
    getZ(aux2bb5,torS[r]);
v2= objAtomos.getX(aux3bb5,torS[r])-objAtomos.
    getX(aux2bb5,torS[r]);
x2= objAtomos.getY(aux3bb5,torS[r])-objAtomos.
    getY(aux2bb5,torS[r]);
z2= objAtomos.getZ(aux3bb5,torS[r])-objAtomos.
    getZ(aux2bb5,torS[r]);
v3= objAtomos.getX(aux2bb5,torS[r])-objAtomos.
    getX(aux3bb5,torS[r]);
x3= objAtomos.getY(aux2bb5,torS[r])-objAtomos.
    getY(aux3bb5,torS[r]);
z3= objAtomos.getZ(aux2bb5,torS[r])-objAtomos.
    getZ(aux3bb5,torS[r]);
v4= objAtomos.getX(aux4bb5,torS[r])-objAtomos.
    getX(aux3bb5,torS[r]);
x4= objAtomos.getY(aux4bb5,torS[r])-objAtomos.
    getY(aux3bb5,torS[r]);
z4= objAtomos.getZ(aux4bb5,torS[r])-objAtomos.
    getZ(aux3bb5,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb5= (prod)/(pmod);
angdbb5= Math.acos(auxdbb5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
```

```
    angdbb5= -angdbb5;
  }
  pangdbb5= (getVnU(molS[i][j],molS[i][j+1],molS
      [i+1][j+1],molS[i+1][j+2],sem,num,indice)/
      getPath(molS[i][j],molS[i][j+1],molS[i+1][j
      +1],molS[i+1][j+2],sem,num,indice)*(1+(Math
      .cos((getN(molS[i][j],molS[i][j+1],molS[i
      +1][j+1],molS[i+1][j+2],sem,indice)*angdbb5
      )-getPeriU(molS[i][j],molS[i][j+1],molS[i
      +1][j+1],molS[i+1][j+2],sem,indice)))));
  //arredondamento TINKER
  pangdbb5= (Math.round((pangdbb5 * 10000.0))) /
       10000.0;
  pangdbb5= pangdbb5+auxbb5;
  auxbb5= pangdbb5;
  indiceD= indiceD + 1;
  ind++;
  sem++;
  while (num==0){
    //transforma pra graus o cos
    pangdbb5= (getVnU(molS[i][j],molS[i][j+1],
        molS[i+1][j+1],molS[i+1][j+2],sem,num,
        indice)/getPath(molS[i][j],molS[i][j+1],
        molS[i+1][j+1],molS[i+1][j+2],sem,num,
        indice)*(1+(Math.cos((getN(molS[i][j],
        molS[i][j+1],molS[i+1][j+1],molS[i+1][j
        +2],sem,indice)*angdbb5)-getPeriU(molS[i
        ][j],molS[i][j+1],molS[i+1][j+1],molS[i
        +1][j+2],sem,indice)))));
    //arredondamento TINKER
    pangdbb5= (Math.round((pangdbb5 * 10000.0)))
         / 10000.0;
    pangdbb5= pangdbb5+auxbb5;
    auxbb5= pangdbb5;
    indiceD= indiceD + 1;
    ind++;
```

```
                sem++;
              }
            }
          }//fim else
        }//fim if
        if ((j<qntdcol-2)&&(j>=1)&&(i<qntdlinha-1)&&(molS[i
            ][j]!="") && (molS[i][j+1]!="") && (molS[i+1][j
            +2]!="") && (molS[i+1][j-1]!="")&&((molS[i+1][j
            +1]=="X00000001000")&&(molS[i+1][j]=="
            X00000001000"))){
          int x=j-1;
          if ((molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(8, 9).equals("1")) && (
              molS[i][j+1].substring(10, 11).equals("1"))&&(
              molS[i+1][j+2].substring(11, 12).equals("1"))&&
              (molS[i+1][j+2].substring(8, 9).equals("1"))&&(
                  molS[i+1][x].substring(8, 9).equals("1"))){
              indice=0;
              aux1b6 = objAtomos.setAtomoU(molS[i][j],torS[r])
                  ;
              aux2b6= objAtomos.setAtomoU(molS[i][j+1],torS[r
                  ]);
              aux3b6= objAtomos.setAtomoU(molS[i+1][j+2],torS[
                  r]);
              aux4b6= objAtomos.setAtomoU(molS[i+1][x],torS[r
                  ]);
              v1= objAtomos.getX(aux1b6,torS[r])-objAtomos.
                  getX(aux2b6,torS[r]);
              x1= objAtomos.getY(aux1b6,torS[r])-objAtomos.
                  getY(aux2b6,torS[r]);
              z1= objAtomos.getZ(aux1b6,torS[r])-objAtomos.
                  getZ(aux2b6,torS[r]);
              v2= objAtomos.getX(aux3b6,torS[r])-objAtomos.
                  getX(aux2b6,torS[r]);
              x2= objAtomos.getY(aux3b6,torS[r])-objAtomos.
                  getY(aux2b6,torS[r]);
```

```
z2= objAtomos.getZ(aux3b6,torS[r])-objAtomos.
    getZ(aux2b6,torS[r]);
v3= objAtomos.getX(aux2b6,torS[r])-objAtomos.
    getX(aux3b6,torS[r]);
x3= objAtomos.getY(aux2b6,torS[r])-objAtomos.
    getY(aux3b6,torS[r]);
z3= objAtomos.getZ(aux2b6,torS[r])-objAtomos.
    getZ(aux3b6,torS[r]);
v4= objAtomos.getX(aux4b6,torS[r])-objAtomos.
    getX(aux3b6,torS[r]);
x4= objAtomos.getY(aux4b6,torS[r])-objAtomos.
    getY(aux3b6,torS[r]);
z4= objAtomos.getZ(aux4b6,torS[r])-objAtomos.
    getZ(aux3b6,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb6= (prod)/(pmod);
angdb6= Math.acos(auxdb6);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb6= -angdb6;
}
pangdb6= (getVnU(molS[i][j],molS[i][j+1],molS[i
    +1][j+2],molS[i+1][x],sem,num,indice)/getPath
    (molS[i][j],molS[i][j+1],molS[i+1][j+2],molS[
    i+1][x],sem,num,indice)*(1+(Math.cos((getN(
```

```
            molS[i][j],molS[i][j+1],molS[i+1][j+2],molS[i
            +1][x],sem,indice)*angdb6)-getPeriU(molS[i][j
            ],molS[i][j+1],molS[i+1][j+2],molS[i+1][x],
            sem,indice)))));
        //arredondamento TINKER
        pangdb6= (Math.round((pangdb6 * 10000.0))) /
            10000.0;
        pangdb6= pangdb6+auxb6;
        auxb6= pangdb6;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdb6= (getVnU(molS[i][j],molS[i][j+1],molS[
              i+1][j+2],molS[i+1][x],sem,num,indice)/
              getPath(molS[i][j],molS[i][j+1],molS[i+1][j
              +2],molS[i+1][x],sem,num,indice)*(1+(Math.
              cos((getN(molS[i][j],molS[i][j+1],molS[i
              +1][j+2],molS[i+1][x],sem,indice)*angdb6)-
              getPeriU(molS[i][j],molS[i][j+1],molS[i+1][
              j+2],molS[i+1][x],sem,indice)))));
          //arredondamento TINKER
          pangdb6= (Math.round((pangdb6 * 10000.0))) /
              10000.0;
          pangdb6= pangdb6+auxb6;
          auxb6= pangdb6;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }
    }else if(((j<qntdcol-2)&&(i<qntdlinha-1)&&(molS[i][j
        ]!="") && (molS[i][j+1]!="") && (molS[i+1][j
        +2]!="") && (molS[i+1][j+1]!="")){
```

```
if ((molS[i][j].substring(8, 9).equals("1"))&&(
   molS[i][j+1].substring(8, 9).equals("1"))&&(
   molS[i][j+1].substring(10, 11).equals("1"))
   &&(molS[i+1][j+2].substring(11, 12).equals
   ("1"))&&
      (molS[i+1][j+1].substring(8, 9).equals
         ("1"))&&(molS[i+1][j+2].substring(8, 9)
         .equals("1"))){
  indice=0;
  aux1b6= objAtomos.setAtomoU(molS[i][j],torS[r
     ]);
  aux2b6= objAtomos.setAtomoU(molS[i][j+1],torS[
     r]);
  aux3b6= objAtomos.setAtomoU(molS[i+1][j+2],
     torS[r]);
  aux4b6= objAtomos.setAtomoU(molS[i+1][j+1],
     torS[r]);
  v1= objAtomos.getX(aux1b6,torS[r])-objAtomos.
     getX(aux2b6,torS[r]);
  x1= objAtomos.getY(aux1b6,torS[r])-objAtomos.
     getY(aux2b6,torS[r]);
  z1= objAtomos.getZ(aux1b6,torS[r])-objAtomos.
     getZ(aux2b6,torS[r]);
  v2= objAtomos.getX(aux3b6,torS[r])-objAtomos.
     getX(aux2b6,torS[r]);
  x2= objAtomos.getY(aux3b6,torS[r])-objAtomos.
     getY(aux2b6,torS[r]);
  z2= objAtomos.getZ(aux3b6,torS[r])-objAtomos.
     getZ(aux2b6,torS[r]);
  v3= objAtomos.getX(aux2b6,torS[r])-objAtomos.
     getX(aux3b6,torS[r]);
  x3= objAtomos.getY(aux2b6,torS[r])-objAtomos.
     getY(aux3b6,torS[r]);
  z3= objAtomos.getZ(aux2b6,torS[r])-objAtomos.
     getZ(aux3b6,torS[r]);
```

```
v4= objAtomos.getX(aux4b6,torS[r])-objAtomos.
    getX(aux3b6,torS[r]);
x4= objAtomos.getY(aux4b6,torS[r])-objAtomos.
    getY(aux3b6,torS[r]);
z4= objAtomos.getZ(aux4b6,torS[r])-objAtomos.
    getZ(aux3b6,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb6= (prod)/(pmod);
angdb6= Math.acos(auxdb6);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb6= -angdb6;
}
pangdb6= (getVnU(molS[i][j],molS[i][j+1],molS[
    i+1][j+2],molS[i+1][j+1],sem,num,indice)/
    getPath(molS[i][j],molS[i][j+1],molS[i+1][j
    +2],molS[i+1][j+1],sem,num,indice)*(1+(Math
    .cos((getN(molS[i][j],molS[i][j+1],molS[i
    +1][j+2],molS[i+1][j+1],sem,indice)*angdb6)
    -getPeriU(molS[i][j],molS[i][j+1],molS[i
    +1][j+2],molS[i+1][j+1],sem,indice)))));
//arredondamento TINKER
pangdb6= (Math.round((pangdb6 * 10000.0))) /
    10000.0;
pangdb6= pangdb6+auxb6;
```

```
            auxb6= pangdb6;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdb6= (getVnU(molS[i][j],molS[i][j+1],
                  molS[i+1][j+2],molS[i+1][j+1],sem,num,
                  indice)/getPath(molS[i][j],molS[i][j+1],
                  molS[i+1][j+2],molS[i+1][j+1],sem,num,
                  indice)*(1+(Math.cos((getN(molS[i][j],
                  molS[i][j+1],molS[i+1][j+2],molS[i+1][j
                  +1],sem,indice)*angdb6)-getPeriU(molS[i][
                  j],molS[i][j+1],molS[i+1][j+2],molS[i+1][
                  j+1],sem,indice)))));
              //arredondamento TINKER
              pangdb6= (Math.round((pangdb6 * 10000.0))) /
                   10000.0;
              pangdb6= pangdb6+auxb6;
              auxb6= pangdb6;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }//fim if
        }//fim if
        if((i<qntdlinha-1)&&(j<qntdcol-3)&&(molS[i][j]!="")
           && (molS[i+1][j+1]!="") && (molS[i][j+2]!="")&& (
           molS[i+1][j+3]!="")&&((molS[i][j].substring(10,
           11).equals("1"))&&(molS[i+1][j+1].substring(11,
           12).equals("1"))&&(molS[i+1][j+1].substring(10,
           11).equals("1"))&&(molS[i][j+2].substring(11, 12)
           .equals("1"))&&(molS[i][j+2].substring(10, 11).
           equals("1"))&&(molS[i+1][j+3].substring(11, 12).
           equals("1")))){
          indice=0;
```

```
double[] aux1z = objAtomos.setAtomoU(molS[i][j],
    torS[r]);
double[] aux2z = objAtomos.setAtomoU(molS[i+1][j
    +1],torS[r]);
double[] aux3z = objAtomos.setAtomoU(molS[i][j+2],
    torS[r]);
double[] aux4z = objAtomos.setAtomoU(molS[i+1][j
    +3],torS[r]);
v1= objAtomos.getX(aux1z,torS[r])-objAtomos.getX(
    aux2z,torS[r]);
x1= objAtomos.getY(aux1z,torS[r])-objAtomos.getY(
    aux2z,torS[r]);
z1= objAtomos.getZ(aux1z,torS[r])-objAtomos.getZ(
    aux2z,torS[r]);
v2= objAtomos.getX(aux3z,torS[r])-objAtomos.getX(
    aux2z,torS[r]);
x2= objAtomos.getY(aux3z,torS[r])-objAtomos.getY(
    aux2z,torS[r]);
z2= objAtomos.getZ(aux3z,torS[r])-objAtomos.getZ(
    aux2z,torS[r]);
v3= objAtomos.getX(aux2z,torS[r])-objAtomos.getX(
    aux3z,torS[r]);
x3= objAtomos.getY(aux2z,torS[r])-objAtomos.getY(
    aux3z,torS[r]);
z3= objAtomos.getZ(aux2z,torS[r])-objAtomos.getZ(
    aux3z,torS[r]);
v4= objAtomos.getX(aux4z,torS[r])-objAtomos.getX(
    aux3z,torS[r]);
x4= objAtomos.getY(aux4z,torS[r])-objAtomos.getY(
    aux3z,torS[r]);
z4= objAtomos.getZ(aux4z,torS[r])-objAtomos.getZ(
    aux3z,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
```

```
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdz = (prod)/(pmod);
angdz = Math.acos(auxdz);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz= -angdz;
}
pangdz = (getVnU(molS[i][j],molS[i+1][j+1],molS[i
    ][j+2],molS[i+1][j+3],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j+1],molS[i][j+2],molS[i
    +1][j+3],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j+1],molS[i][j+2],molS[i
    +1][j+3],sem,indice)*angdz)-getPeriU(molS[i][j
    ],molS[i+1][j+1],molS[i][j+2],molS[i+1][j+3],
    sem,indice)))));
//arredondamento TINKER
pangdz= (Math.round((pangdz * 10000.0))) /
    10000.0;
pangdz= pangdz+auxz;
auxz= pangdz;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdz= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
      ][j+2],molS[i+1][j+3],sem,num,indice)/getPath
      (molS[i][j],molS[i+1][j+1],molS[i][j+2],molS[
      i+1][j+3],sem,num,indice)*(1+(Math.cos((getN(
```

```
            molS[i][j],molS[i+1][j+1],molS[i][j+2],molS[i
            +1][j+3],sem,indice)*angdz)-getPeriU(molS[i][
            j],molS[i+1][j+1],molS[i][j+2],molS[i+1][j
            +3],sem,indice)))));
          //arredondamento TINKER
          pangdz= (Math.round((pangdz * 10000.0))) /
            10000.0;
          pangdz= pangdz+auxz;
          auxz= pangdz;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }

      }//fim if
      if((i>=1)&&(j<qntdcol-2)&&(molS[i][j]!="") && (molS[
        i-1][j+1]!="") && (molS[i][j+2]!="")&& (molS[i
        -1][j+2]!="")&&((molS[i][j].substring(10, 11).
        equals("1"))&&(molS[i-1][j+1].substring(11, 12).
        equals("1"))&&(molS[i-1][j+1].substring(10, 11).
        equals("1"))&&(molS[i][j+2].substring(11, 12).
        equals("1"))&&(molS[i][j+2].substring(9, 10).
        equals("1"))&&(molS[i-1][j+2].substring(9, 10).
        equals("1")))){
        indice=0;
        double[] aux1z1 = objAtomos.setAtomoU(molS[i][j],
            torS[r]);
        double[] aux2z1 = objAtomos.setAtomoU(molS[i-1][j
            +1],torS[r]);
        double[] aux3z1 = objAtomos.setAtomoU(molS[i][j
            +2],torS[r]);
        double[] aux4z1 = objAtomos.setAtomoU(molS[i-1][j
            +2],torS[r]);
        v1= objAtomos.getX(aux1z1,torS[r])-objAtomos.getX(
            aux2z1,torS[r]);
```

```
x1= objAtomos.getY(aux1z1,torS[r])-objAtomos.getY(
    aux2z1,torS[r]);
z1= objAtomos.getZ(aux1z1,torS[r])-objAtomos.getZ(
    aux2z1,torS[r]);
v2= objAtomos.getX(aux3z1,torS[r])-objAtomos.getX(
    aux2z1,torS[r]);
x2= objAtomos.getY(aux3z1,torS[r])-objAtomos.getY(
    aux2z1,torS[r]);
z2= objAtomos.getZ(aux3z1,torS[r])-objAtomos.getZ(
    aux2z1,torS[r]);
v3= objAtomos.getX(aux2z1,torS[r])-objAtomos.getX(
    aux3z1,torS[r]);
x3= objAtomos.getY(aux2z1,torS[r])-objAtomos.getY(
    aux3z1,torS[r]);
z3= objAtomos.getZ(aux2z1,torS[r])-objAtomos.getZ(
    aux3z1,torS[r]);
v4= objAtomos.getX(aux4z1,torS[r])-objAtomos.getX(
    aux3z1,torS[r]);
x4= objAtomos.getY(aux4z1,torS[r])-objAtomos.getY(
    aux3z1,torS[r]);
z4= objAtomos.getZ(aux4z1,torS[r])-objAtomos.getZ(
    aux3z1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
double auxdz1 = (prod)/(pmod);
double angdz1= Math.acos(auxdz1);
```

```
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz1= -angdz1;
}
pangdz1= (getVnU(molS[i][j],molS[i-1][j+1],molS[i
    ][j+2],molS[i-1][j+2],sem,num,indice)/getPath(
    molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i
    -1][j+2],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i-1][j+1],molS[i][j+2],molS[i
    -1][j+2],sem,indice)*angdz1)-getPeriU(molS[i][j
    ],molS[i-1][j+1],molS[i][j+2],molS[i-1][j+2],
    sem,indice)))));
//arredondamento TINKER
pangdz1= (Math.round((pangdz1 * 10000.0))) /
    10000.0;
pangdz1= pangdz1+auxz1;
auxz1= pangdz1;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdz1= (getVnU(molS[i][j],molS[i-1][j+1],molS[
      i][j+2],molS[i-1][j+2],sem,num,indice)/
      getPath(molS[i][j],molS[i-1][j+1],molS[i][j
      +2],molS[i-1][j+2],sem,num,indice)*(1+(Math.
      cos((getN(molS[i][j],molS[i-1][j+1],molS[i][j
      +2],molS[i-1][j+2],sem,indice)*angdz1)-
      getPeriU(molS[i][j],molS[i-1][j+1],molS[i][j
      +2],molS[i-1][j-2],sem,indice)))));
  //arredondamento TINKER
  pangdz1= (Math.round((pangdz1 * 10000.0))) /
      10000.0;
  pangdz1= pangdz1+auxz1;
  auxz1= pangdz1;
  indiceD= indiceD + 1;
```

```
            ind++;
            sem++;
          }
        }//fim if
        if(((i<qntdlinha-2)&&(j>=4)&&(molS[i][j]!="") && (
            molS[i+1][j-1]!="") && (molS[i+1][j-4]!="")&& (
            molS[i+2][j-4]!=""))&&((molS[i+1][j-2]=="
            X00000001000")&&(molS[i+1][j-3]=="X00000001000"))
            ){
          int x=j-4;
          if((molS[i][j]!="") && (molS[i+1][j-1]!="") && (
              molS[i+1][x]!="") && (molS[i+2][x]!="")){
            if ((molS[i][j].substring(11, 12).equals("1"))
                &&(molS[i+1][j-1].substring(10, 11).equals
                ("1")) && (molS[i+1][j-1].substring(8, 9).
                equals("1"))&&(molS[i+1][x].substring(8, 9).
                equals("1"))&&(molS[i+1][x].substring(9, 10).
                equals("1"))&&(molS[i+2][x].substring(9, 10).
                equals("1"))){
              indice=0;
              double[] aux1z2 = objAtomos.setAtomoU(molS[i][
                  j],torS[r]);
              double[] aux2z2= objAtomos.setAtomoU(molS[i
                  +1][j-1],torS[r]);
              double[] aux3z2= objAtomos.setAtomoU(molS[i
                  +1][x],torS[r]);
              double[] aux4z2= objAtomos.setAtomoU(molS[i
                  +2][x],torS[r]);
              v1= objAtomos.getX(aux1z2,torS[r])-objAtomos.
                  getX(aux2z2,torS[r]);
              x1= objAtomos.getY(aux1z2,torS[r])-objAtomos.
                  getY(aux2z2,torS[r]);
              z1= objAtomos.getZ(aux1z2,torS[r])-objAtomos.
                  getZ(aux2z2,torS[r]);
              v2= objAtomos.getX(aux3z2,torS[r])-objAtomos.
                  getX(aux2z2,torS[r]);
```

```
x2= objAtomos.getY(aux3z2,torS[r])-objAtomos.
    getY(aux2z2,torS[r]);
z2= objAtomos.getZ(aux3z2,torS[r])-objAtomos.
    getZ(aux2z2,torS[r]);
v3= objAtomos.getX(aux2z2,torS[r])-objAtomos.
    getX(aux3z2,torS[r]);
x3= objAtomos.getY(aux2z2,torS[r])-objAtomos.
    getY(aux3z2,torS[r]);
z3= objAtomos.getZ(aux2z2,torS[r])-objAtomos.
    getZ(aux3z2,torS[r]);
v4= objAtomos.getX(aux4z2,torS[r])-objAtomos.
    getX(aux3z2,torS[r]);
x4= objAtomos.getY(aux4z2,torS[r])-objAtomos.
    getY(aux3z2,torS[r]);
z4= objAtomos.getZ(aux4z2,torS[r])-objAtomos.
    getZ(aux3z2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz2= (prod)/(pmod);
angdz2= Math.acos(auxdz2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz2= -angdz2;
}
pangdz2= (getVnU(molS[i][j],molS[i+1][j-1],
    molS[i+1][x],molS[i+2][x],sem,num,indice)/
```

```
            getPath(molS[i][j],molS[i+1][j-1],molS[i
            +1][x],molS[i+2][x],sem,num,indice)*(1+(
            Math.cos((getN(molS[i][j],molS[i+1][j-1],
            molS[i+1][x],molS[i+2][x],sem,indice)*
            angdz2)-getPeriU(molS[i][j],molS[i+1][j-1],
            molS[i+1][x],molS[i+2][x],sem,indice)))));
          //arredondamento TINKER
          pangdz2= (Math.round((pangdz2 * 10000.0))) /
              10000.0;
          pangdz2= pangdz2+auxz2;
          auxz2= pangdz2;
          indiceD= indiceD + 1;
          ind++;
          sem++;
          while (num==0){
            //transforma pra graus o cos
            pangdz2= (getVnU(molS[i][j],molS[i+1][j-1],
                molS[i+1][x],molS[i+2][x],sem,num,indice)
                /getPath(molS[i][j],molS[i+1][j-1],molS[i
                +1][x],molS[i+2][x],sem,num,indice)*(1+(
                Math.cos((getN(molS[i][j],molS[i+1][j-1],
                molS[i+1][x],molS[i+2][x],sem,indice)*
                angdz2)-getPeriU(molS[i][j],molS[i+1][j
                -1],molS[i+1][x],molS[i+2][x],sem,indice)
                ))));
            //arredondamento TINKER
            pangdz2= (Math.round((pangdz2 * 10000.0))) /
                10000.0;
            pangdz2= pangdz2+auxz2;
            auxz2= pangdz2;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }
```

```
}else if (((i<qntdlinha-2)&&(j>=2)&&(molS[i][j]!="")
    && (molS[i+1][j-1]!="") && (molS[i+1][j-2]!="")
   && (molS[i+2][j-2]!=""))
    &&(molS[i][j].substring(11, 12).equals("1"))&&(
        molS[i+1][j-1].substring(10, 11).equals("1"))
        &&(molS[i+1][j-1].substring(8, 9).equals("1")
        )&&(molS[i+1][j-2].substring(8, 9).equals
        ("1"))&&
       (molS[i+1][j-2].substring(9, 10).equals("1"))
          &&(molS[i+2][j-2].substring(9, 10).equals
          ("1"))&& (molS[i+1][j-2]!="X00000001000")){
    indice=0;
    double[] aux1z2 = objAtomos.setAtomoU(molS[i][j
        ],torS[r]);
    double[] aux2z2 = objAtomos.setAtomoU(molS[i+1][
        j-1],torS[r]);
    double[] aux3z2 = objAtomos.setAtomoU(molS[i+1][
        j-2],torS[r]);
    double[] aux4z2 = objAtomos.setAtomoU(molS[i+2][
        j-2],torS[r]);
    v1= objAtomos.getX(aux1z2,torS[r])-objAtomos.
        getX(aux2z2,torS[r]);
    x1= objAtomos.getY(aux1z2,torS[r])-objAtomos.
        getY(aux2z2,torS[r]);
    z1= objAtomos.getZ(aux1z2,torS[r])-objAtomos.
        getZ(aux2z2,torS[r]);
    v2= objAtomos.getX(aux3z2,torS[r])-objAtomos.
        getX(aux2z2,torS[r]);
    x2= objAtomos.getY(aux3z2,torS[r])-objAtomos.
        getY(aux2z2,torS[r]);
    z2= objAtomos.getZ(aux3z2,torS[r])-objAtomos.
        getZ(aux2z2,torS[r]);
    v3= objAtomos.getX(aux2z2,torS[r])-objAtomos.
        getX(aux3z2,torS[r]);
    x3= objAtomos.getY(aux2z2,torS[r])-objAtomos.
        getY(aux3z2,torS[r]);
```

```
z3= objAtomos.getZ(aux2z2,torS[r])-objAtomos.
    getZ(aux3z2,torS[r]);
v4= objAtomos.getX(aux4z2,torS[r])-objAtomos.
    getX(aux3z2,torS[r]);
x4= objAtomos.getY(aux4z2,torS[r])-objAtomos.
    getY(aux3z2,torS[r]);
z4= objAtomos.getZ(aux4z2,torS[r])-objAtomos.
    getZ(aux3z2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz2 = (prod)/(pmod);
angdz2 = Math.acos(auxdz2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz2= -angdz2;
}
pangdz2= (getVnU(molS[i][j],molS[i+1][j-1],molS[
    i+1][j-2],molS[i+2][j-2],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j-1],molS[i+1][j
    -2],molS[i+2][j-2],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i+1][j-1],molS[i
    +1][j-2],molS[i+2][j-2],sem,indice)*angdz2)-
    getPeriU(molS[i][j],molS[i+1][j-1],molS[i+1][
    j-2],molS[i+2][j-2],sem,indice)))));
//arredondamento TINKER
```

```
            pangdz2= (Math.round((pangdz2 * 10000.0))) /
                10000.0;
            pangdz2= pangdz2+auxz2;
            auxz2= pangdz2;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdz2= (getVnU(molS[i][j],molS[i+1][j-1],
                  molS[i+1][j-2],molS[i+2][j-2],sem,num,
                  indice)/getPath(molS[i][j],molS[i+1][j-1],
                  molS[i+1][j-2],molS[i+2][j-2],sem,num,
                  indice)*(1+(Math.cos((getN(molS[i][j],molS[
                  i+1][j-1],molS[i+1][j-2],molS[i+2][j-2],sem
                  ,indice)*angdz2)-getPeriU(molS[i][j],molS[i
                  +1][j-1],molS[i+1][j-2],molS[i+2][j-2],sem,
                  indice)))));
              //arredondamento TINKER
              pangdz2= (Math.round((pangdz2 * 10000.0))) /
                  10000.0;
              pangdz2= pangdz2+auxz2;
              auxz2= pangdz2;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }//fim if
          if(((i<qntdlinha-2)&&(j>=3)&&(molS[i][j]!="") && (
              molS[i+1][j-1]!="") && (molS[i+2][j]!="")&& (molS
              [i+2][j-3]!=""))&&((molS[i+2][j-1]=="X00000001000
              ")&&(molS[i+2][j-2]=="X00000001000"))){
            int x=j-3;
            if((molS[i][j]!="") && (molS[i+1][j-1]!="") && (
                molS[i+2][j]!="") && (molS[i+2][x]!="")){
```

```
if ((molS[i][j].substring(11, 12).equals("1"))
   &&(molS[i+1][j-1].substring(10, 11).equals
   ("1")) && (molS[i+1][j-1].substring(10, 11)
   .equals("1"))&&(molS[i+2][j].substring(11,
   12).equals("1"))&&(molS[i+2][j].substring
   (8, 9).equals("1"))&&(molS[i+2][x].
   substring(8, 9).equals("1"))){
  indice=0;
  double[] aux1z3 = objAtomos.setAtomoU(molS[i
     ][j],torS[r]);
  double[] aux2z3= objAtomos.setAtomoU(molS[i
     +1][j-1],torS[r]);
  double[] aux3z3= objAtomos.setAtomoU(molS[i
     +2][j],torS[r]);
  double[] aux4z3= objAtomos.setAtomoU(molS[i
     +2][x],torS[r]);
  v1= objAtomos.getX(aux1z3,torS[r])-objAtomos
     .getX(aux2z3,torS[r]);
  x1= objAtomos.getY(aux1z3,torS[r])-objAtomos
     .getY(aux2z3,torS[r]);
  z1= objAtomos.getZ(aux1z3,torS[r])-objAtomos
     .getZ(aux2z3,torS[r]);
  v2= objAtomos.getX(aux3z3,torS[r])-objAtomos
     .getX(aux2z3,torS[r]);
  x2= objAtomos.getY(aux3z3,torS[r])-objAtomos
     .getY(aux2z3,torS[r]);
  z2= objAtomos.getZ(aux3z3,torS[r])-objAtomos
     .getZ(aux2z3,torS[r]);
  v3= objAtomos.getX(aux2z3,torS[r])-objAtomos
     .getX(aux3z3,torS[r]);
  x3= objAtomos.getY(aux2z3,torS[r])-objAtomos
     .getY(aux3z3,torS[r]);
  z3= objAtomos.getZ(aux2z3,torS[r])-objAtomos
     .getZ(aux3z3,torS[r]);
  v4= objAtomos.getX(aux4z3,torS[r])-objAtomos
     .getX(aux3z3,torS[r]);
```

```
x4= objAtomos.getY(aux4z3,torS[r])-objAtomos
    .getY(aux3z3,torS[r]);
z4= objAtomos.getZ(aux4z3,torS[r])-objAtomos
    .getZ(aux3z3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)
   +Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2
    ,2)+Math.pow(c2,2));
pmod= mod*mod2;
auxdz3= (prod)/(pmod);
angdz3= Math.acos(auxdz3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz3= -angdz3;
}
pangdz3= (getVnU(molS[i][j],molS[i+1][j-1],
    molS[i+2][j],molS[i+2][x],sem,num,indice)
    /getPath(molS[i][j],molS[i+1][j-1],molS[i
    +2][j],molS[i+2][x],sem,num,indice)*(1+(
    Math.cos((getN(molS[i][j],molS[i+1][j-1],
    molS[i+2][j],molS[i+2][x],sem,indice)*
    angdz3)-getPeriU(molS[i][j],molS[i+1][j
    -1],molS[i+2][j],molS[i+2][x],sem,indice)
    ))));
//arredondamento TINKER
pangdz3= (Math.round((pangdz3 * 10000.0))) /
     10000.0;
pangdz3= pangdz3+auxz3;
auxz3= pangdz3;
```

```
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdz3= (getVnU(molS[i][j],molS[i+1][j
                  -1],molS[i+2][j],molS[i+2][x],sem,num,
                  indice)/getPath(molS[i][j],molS[i+1][j
                  -1],molS[i+2][j],molS[i+2][x],sem,num,
                  indice)*(1+(Math.cos((getN(molS[i][j],
                  molS[i+1][j-1],molS[i+2][j],molS[i+2][x
                  ],sem,indice)*angdz3)-getPeriU(molS[i][
                  j],molS[i+1][j-1],molS[i+2][j],molS[i
                  +2][x],sem,indice)))));
              //arredondamento TINKER
              pangdz3= (Math.round((pangdz3 * 10000.0)))
                   / 10000.0;
              pangdz3= pangdz3+auxz3;
              auxz3= pangdz3;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }else if(((i<qntdlinha-2)&&(j>=1)&&(molS[i][j]!="")
        && (molS[i+1][j-1]!="") && (molS[i+2][j]!="")&& (
        molS[i+2][j-1]!=""))&&
            (molS[i][j].substring(11, 12).equals("1"))&&(
              molS[i+1][j-1].substring(10, 11).equals
              ("1"))&&(molS[i+1][j-1].substring(10, 11).
              equals("1"))&&(molS[i+2][j].substring(11,
              12).equals("1"))&&
        (molS[i+2][j].substring(8, 9).equals("1"))&&(molS[
          i+2][j-1].substring(8, 9).equals("1"))&& (molS[
          i+2][j-1]!="X00000001000")){
```

```
indice=0;
double[] aux1z3 = objAtomos.setAtomoU(molS[i][j
    ],torS[r]);
double[] aux2z3 = objAtomos.setAtomoU(molS[i+1][
    j-1],torS[r]);
double[] aux3z3 = objAtomos.setAtomoU(molS[i+2][
    j],torS[r]);
double[] aux4z3 = objAtomos.setAtomoU(molS[i+2][
    j-1],torS[r]);
v1= objAtomos.getX(aux1z3,torS[r])-objAtomos.
    getX(aux2z3,torS[r]);
x1= objAtomos.getY(aux1z3,torS[r])-objAtomos.
    getY(aux2z3,torS[r]);
z1= objAtomos.getZ(aux1z3,torS[r])-objAtomos.
    getZ(aux2z3,torS[r]);
v2= objAtomos.getX(aux3z3,torS[r])-objAtomos.
    getX(aux2z3,torS[r]);
x2= objAtomos.getY(aux3z3,torS[r])-objAtomos.
    getY(aux2z3,torS[r]);
z2= objAtomos.getZ(aux3z3,torS[r])-objAtomos.
    getZ(aux2z3,torS[r]);
v3= objAtomos.getX(aux2z3,torS[r])-objAtomos.
    getX(aux3z3,torS[r]);
x3= objAtomos.getY(aux2z3,torS[r])-objAtomos.
    getY(aux3z3,torS[r]);
z3= objAtomos.getZ(aux2z3,torS[r])-objAtomos.
    getZ(aux3z3,torS[r]);
v4= objAtomos.getX(aux4z3,torS[r])-objAtomos.
    getX(aux3z3,torS[r]);
x4= objAtomos.getY(aux4z3,torS[r])-objAtomos.
    getY(aux3z3,torS[r]);
z4= objAtomos.getZ(aux4z3,torS[r])-objAtomos.
    getZ(aux3z3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
```

```
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz3 = (prod)/(pmod);
angdz3 = Math.acos(auxdz3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz3= -angdz3;
}
pangdz3= (getVnU(molS[i][j],molS[i+1][j-1],molS[
    i+2][j],molS[i+2][j-1],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j-1],molS[i+2][j
    ],molS[i+2][j-1],sem,num,indice)*(1+(Math.cos
    ((getN(molS[i][j],molS[i+1][j-1],molS[i+2][j
    ],molS[i+2][j-1],sem,indice)*angdz3)-getPeriU
    (molS[i][j],molS[i+1][j-1],molS[i+2][j],molS[
    i+2][j-1],sem,indice)))));
//arredondamento TINKER
pangdz3= (Math.round((pangdz3 * 10000.0))) /
    10000.0;
pangdz3= pangdz3+auxz3;
auxz3= pangdz3;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdz3= (getVnU(molS[i][j],molS[i+1][j-1],
      molS[i+2][j],molS[i+2][j-1],sem,num,indice)
      /getPath(molS[i][j],molS[i+1][j-1],molS[i
```

```
                +2][j],molS[i+2][j-1],sem,num,indice)*(1+(
                Math.cos((getN(molS[i][j],molS[i+1][j-1],
                molS[i+2][j],molS[i+2][j-1],sem,indice)*
                angdz3)-getPeriU(molS[i][j],molS[i+1][j-1],
                molS[i+2][j],molS[i+2][j-1],sem,indice)))))
                ;
            //arredondamento TINKER
            pangdz3= (Math.round((pangdz3 * 10000.0))) /
                10000.0;
            pangdz3= pangdz3+auxz3;
            auxz3= pangdz3;
            indiceD= indiceD + 1;
            ind++;
            sem++;
        }
    }//fim if
    if((i<qntdlinha-3)&&(j>=1)&&(molS[i][j]!="") && (
        molS[i+1][j-1]!="") && (molS[i+2][j]!="")&& (molS
        [i+3][j]!="")&&((molS[i][j].substring(11, 12).
        equals("1"))&&(molS[i+1][j-1].substring(10, 11).
        equals("1"))&&(molS[i+1][j-1].substring(10, 11).
        equals("1"))&&(molS[i+2][j].substring(11, 12).
        equals("1"))&&(molS[i+2][j].substring(9, 10).
        equals("1"))&&(molS[i+3][j].substring(9, 10).
        equals("1")))){
      if ((molS[i][j].substring(0, 7).equals("NCA0002"))
        &&(molS[i+1][j-1].substring(0, 7).equals("
        OSR1001"))){

      }else{
        indice=0;
        double[] aux1z4 = objAtomos.setAtomoU(molS[i][j
            ],torS[r]);
        double[] aux2z4 = objAtomos.setAtomoU(molS[i+1][
            j-1],torS[r]);
```

```
double[] aux3z4 = objAtomos.setAtomoU(molS[i+2][
    j],torS[r]);
double[] aux4z4 = objAtomos.setAtomoU(molS[i+3][
    j],torS[r]);
v1= objAtomos.getX(aux1z4,torS[r])-objAtomos.
    getX(aux2z4,torS[r]);
x1= objAtomos.getY(aux1z4,torS[r])-objAtomos.
    getY(aux2z4,torS[r]);
z1= objAtomos.getZ(aux1z4,torS[r])-objAtomos.
    getZ(aux2z4,torS[r]);
v2= objAtomos.getX(aux3z4,torS[r])-objAtomos.
    getX(aux2z4,torS[r]);
x2= objAtomos.getY(aux3z4,torS[r])-objAtomos.
    getY(aux2z4,torS[r]);
z2= objAtomos.getZ(aux3z4,torS[r])-objAtomos.
    getZ(aux2z4,torS[r]);
v3= objAtomos.getX(aux2z4,torS[r])-objAtomos.
    getX(aux3z4,torS[r]);
x3= objAtomos.getY(aux2z4,torS[r])-objAtomos.
    getY(aux3z4,torS[r]);
z3= objAtomos.getZ(aux2z4,torS[r])-objAtomos.
    getZ(aux3z4,torS[r]);
v4= objAtomos.getX(aux4z4,torS[r])-objAtomos.
    getX(aux3z4,torS[r]);
x4= objAtomos.getY(aux4z4,torS[r])-objAtomos.
    getY(aux3z4,torS[r]);
z4= objAtomos.getZ(aux4z4,torS[r])-objAtomos.
    getZ(aux3z4,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
```

```
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz4 = (prod)/(pmod);
angdz4 = Math.acos(auxdz4);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz4= -angdz4;
}
pangdz4= (getVnU(molS[i][j],molS[i+1][j-1],molS[
    i+2][j],molS[i+3][j],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j-1],molS[i+2][j],molS[i
    +3][j],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i+1][j-1],molS[i+2][j],molS[i
    +3][j],sem,indice)*angdz4)-getPeriU(molS[i][j
    ],molS[i+1][j-1],molS[i+2][j],molS[i+3][j],
    sem,indice)))));
//arredondamento TINKER
pangdz4= (Math.round((pangdz4 * 10000.0))) /
    10000.0;
pangdz4= pangdz4+auxz4;
auxz4= pangdz4;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdz4= (getVnU(molS[i][j],molS[i+1][j-1],
      molS[i+2][j],molS[i+3][j],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j-1],molS[i
      +2][j],molS[i+3][j],sem,num,indice)*(1+(
      Math.cos((getN(molS[i][j],molS[i+1][j-1],
      molS[i+2][j],molS[i+3][j],sem,indice)*
      angdz4)-getPeriU(molS[i][j],molS[i+1][j-1],
```

```
            molS[i+2][j],molS[i+3][j],sem,indice)))));
          //arredondamento TINKER
          pangdz4= (Math.round((pangdz4 * 10000.0))) /
              10000.0;
          pangdz4= pangdz4+auxz4;
          auxz4= pangdz4;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }
    }//fim if
    if((i>=1)&&(j>=2)&&(i<qntdlinha-1)&&(molS[i][j]!="")
        && (molS[i-1][j-1]!="") && (molS[i][j-2]!="")&&
       (molS[i+1][j-1]!="")&&((molS[i][j].substring(11,
       12).equals("1"))&&(molS[i-1][j-1].substring(10,
       11).equals("1"))&&(molS[i-1][j-1].substring(11,
       12).equals("1"))&&(molS[i][j-2].substring(10, 11)
       .equals("1"))&&(molS[i][j-2].substring(10, 11).
       equals("1"))&&(molS[i+1][j-1].substring(11, 12).
       equals("1")))){
        indice=0;
        double[] aux1z5 = objAtomos.setAtomoU(molS[i][j
            ],torS[r]);
        double[] aux2z5 = objAtomos.setAtomoU(molS[i-1][
            j-1],torS[r]);
        double[] aux3z5 = objAtomos.setAtomoU(molS[i][j
            -2],torS[r]);
        double[] aux4z5 = objAtomos.setAtomoU(molS[i+1][
            j-1],torS[r]);
        v1= objAtomos.getX(aux1z5,torS[r])-objAtomos.
            getX(aux2z5,torS[r]);
        x1= objAtomos.getY(aux1z5,torS[r])-objAtomos.
            getY(aux2z5,torS[r]);
        z1= objAtomos.getZ(aux1z5,torS[r])-objAtomos.
            getZ(aux2z5,torS[r]);
```

```
v2= objAtomos.getX(aux3z5,torS[r])-objAtomos.
    getX(aux2z5,torS[r]);
x2= objAtomos.getY(aux3z5,torS[r])-objAtomos.
    getY(aux2z5,torS[r]);
z2= objAtomos.getZ(aux3z5,torS[r])-objAtomos.
    getZ(aux2z5,torS[r]);
v3= objAtomos.getX(aux2z5,torS[r])-objAtomos.
    getX(aux3z5,torS[r]);
x3= objAtomos.getY(aux2z5,torS[r])-objAtomos.
    getY(aux3z5,torS[r]);
z3= objAtomos.getZ(aux2z5,torS[r])-objAtomos.
    getZ(aux3z5,torS[r]);
v4= objAtomos.getX(aux4z5,torS[r])-objAtomos.
    getX(aux3z5,torS[r]);
x4= objAtomos.getY(aux4z5,torS[r])-objAtomos.
    getY(aux3z5,torS[r]);
z4= objAtomos.getZ(aux4z5,torS[r])-objAtomos.
    getZ(aux3z5,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz5 = (prod)/(pmod);
angdz5 = Math.acos(auxdz5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz5= -angdz5;
}
```

```
pangdz5= (getVnU(molS[i][j],molS[i-1][j-1],molS[
    i][j-2],molS[i+1][j-1],sem,num,indice)/
    getPath(molS[i][j],molS[i-1][j-1],molS[i][j
    -2],molS[i+1][j-1],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i-1][j-1],molS[i][j
    -2],molS[i+1][j-1],sem,indice)*angdz5)-
    getPeriU(molS[i][j],molS[i-1][j-1],molS[i][j
    -2],molS[i+1][j-1],sem,indice)))));
//arredondamento TINKER
pangdz5= (Math.round((pangdz5 * 10000.0))) /
    10000.0;
pangdz5= pangdz5+auxz5;
auxz5= pangdz5;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdz5= (getVnU(molS[i][j],molS[i-1][j-1],
      molS[i][j-2],molS[i+1][j-1],sem,num,indice)
      /getPath(molS[i][j],molS[i-1][j-1],molS[i][
      j-2],molS[i+1][j-1],sem,num,indice)*(1+(
      Math.cos((getN(molS[i][j],molS[i-1][j-1],
      molS[i][j-2],molS[i+1][j-1],sem,indice)*
      angdz5)-getPeriU(molS[i][j],molS[i-1][j-1],
      molS[i][j-2],molS[i+1][j-1],sem,indice)))))
      ;
  //arredondamento TINKER
  pangdz5= (Math.round((pangdz5 * 10000.0))) /
      10000.0;
  pangdz5= pangdz5+auxz5;
  auxz5= pangdz5;
  indiceD= indiceD + 1;
  ind++;
  sem++;
}
```

```
        }//fim if
        if((molS[i][j]!="") &&(i<qntdlinha-2)&&(j<qntdcol-2)
           && (molS[i+1][j]!="") && (molS[i+1][j+1]!="") &&
           (molS[i+2][j+2]!="")){
          //verificar se todo esta ligados
          if ((molS[i][j].substring(9, 10).equals("1"))&&(
              molS[i+1][j].substring(9, 10).equals("1"))&&(
              molS[i+1][j].substring(8, 9).equals("1"))&&(
              molS[i+1][j+1].substring(8, 9).equals("1"))&&(
              molS[i+1][j+1].substring(10, 11).equals("1"))
              &&(molS[i+2][j+2].substring(11, 12).equals("1")
              )){
            indice=0;
            double[] aux1u5 = objAtomos.setAtomoU(molS[i][j
                ],torS[r]);
            double[] aux2u5 = objAtomos.setAtomoU(molS[i+1][
                j],torS[r]);
            double[] aux3u5 = objAtomos.setAtomoU(molS[i+1][
                j+1],torS[r]);
            double[] aux4u5 = objAtomos.setAtomoU(molS[i+2][
                j+2],torS[r]);
            v1= objAtomos.getX(aux1u5,torS[r])-objAtomos.
                getX(aux2u5,torS[r]);
            x1= objAtomos.getY(aux1u5,torS[r])-objAtomos.
                getY(aux2u5,torS[r]);
            z1= objAtomos.getZ(aux1u5,torS[r])-objAtomos.
                getZ(aux2u5,torS[r]);
            v2= objAtomos.getX(aux3u5,torS[r])-objAtomos.
                getX(aux2u5,torS[r]);
            x2= objAtomos.getY(aux3u5,torS[r])-objAtomos.
                getY(aux2u5,torS[r]);
            z2= objAtomos.getZ(aux3u5,torS[r])-objAtomos.
                getZ(aux2u5,torS[r]);
            v3= objAtomos.getX(aux2u5,torS[r])-objAtomos.
                getX(aux3u5,torS[r]);
```

```
x3= objAtomos.getY(aux2u5,torS[r])-objAtomos.
    getY(aux3u5,torS[r]);
z3= objAtomos.getZ(aux2u5,torS[r])-objAtomos.
    getZ(aux3u5,torS[r]);
v4= objAtomos.getX(aux4u5,torS[r])-objAtomos.
    getX(aux3u5,torS[r]);
x4= objAtomos.getY(aux4u5,torS[r])-objAtomos.
    getY(aux3u5,torS[r]);
z4= objAtomos.getZ(aux4u5,torS[r])-objAtomos.
    getZ(aux3u5,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu5 = (prod)/(pmod);
angdu5 = Math.acos(auxdu5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu5= -angdu5;
}
pangdu5= ((getVnU(molS[i][j],molS[i+1][j],molS[i
    +1][j+1],molS[i+2][j+2],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j],molS[i+1][j
    +1],molS[i+2][j+2],sem,num,indice))*(1+(Math.
    cos((getN(molS[i][j],molS[i+1][j],molS[i+1][j
    +1],molS[i+2][j+2],sem,indice)*angdu5)-
    getPeriU(molS[i][j],molS[i+1][j],molS[i+1][j
    +1],molS[i+2][j+2],sem,indice)))));
```

```
            //arredondamento TINKER
            pangdu5= (Math.round((pangdu5 * 10000.0))) /
                10000.0;
            pangdu5= pangdu5+auxu5;
            auxu5= pangdu5;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdu5= ((getVnU(molS[i][j],molS[i+1][j],molS
                  [i+1][j+1],molS[i+2][j+2],sem,num,indice)/
                  getPath(molS[i][j],molS[i+1][j],molS[i+1][j
                  +1],molS[i+2][j+2],sem,num,indice))*(1+(
                  Math.cos((getN(molS[i][j],molS[i+1][j],molS
                  [i+1][j+1],molS[i+2][j+2],sem,indice)*
                  angdu5)-getPeriU(molS[i][j],molS[i+1][j],
                  molS[i+1][j+1],molS[i+2][j+2],sem,indice)))
                  ));
              //arredondamento TINKER
              pangdu5= (Math.round((pangdu5 * 10000.0))) /
                  10000.0;
              pangdu5= pangdu5+auxu5;
              auxu5= pangdu5;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }//fim if
        if((molS[i][j]!="") &&(i<qntdlinha-1)&&(j<qntdcol-1)
            && (molS[i+1][j]!="") && (molS[i+1][j+1]!="") &&
            (molS[i][j+1]!="")){
          //verificar se todo esta ligados
          if ((molS[i][j].substring(9, 10).equals("1"))&&(
              molS[i+1][j].substring(9, 10).equals("1"))&&(
```

```
 molS[i+1][j].substring(8, 9).equals("1"))&&(
 molS[i+1][j+1].substring(8, 9).equals("1"))&&(
 molS[i+1][j+1].substring(9, 10).equals("1"))&&(
 molS[i][j+1].substring(9, 10).equals("1"))){
if ((molS[i+1][j].substring(0,7).equals("OHR1001
    "))&&(molS[i+1][j+1].substring(0,7).equals("
    OHR1002"))){

}else{
  indice=0;
  double[] aux1u = objAtomos.setAtomoU(molS[i][j
      ],torS[r]);
  double[] aux2u = objAtomos.setAtomoU(molS[i
      +1][j],torS[r]);
  double[] aux3u = objAtomos.setAtomoU(molS[i
      +1][j+1],torS[r]);
  double[] aux4u = objAtomos.setAtomoU(molS[i][j
      +1],torS[r]);
  v1= objAtomos.getX(aux1u,torS[r])-objAtomos.
      getX(aux2u,torS[r]);
  x1= objAtomos.getY(aux1u,torS[r])-objAtomos.
      getY(aux2u,torS[r]);
  z1= objAtomos.getZ(aux1u,torS[r])-objAtomos.
      getZ(aux2u,torS[r]);
  v2= objAtomos.getX(aux3u,torS[r])-objAtomos.
      getX(aux2u,torS[r]);
  x2= objAtomos.getY(aux3u,torS[r])-objAtomos.
      getY(aux2u,torS[r]);
  z2= objAtomos.getZ(aux3u,torS[r])-objAtomos.
      getZ(aux2u,torS[r]);
  v3= objAtomos.getX(aux2u,torS[r])-objAtomos.
      getX(aux3u,torS[r]);
  x3= objAtomos.getY(aux2u,torS[r])-objAtomos.
      getY(aux3u,torS[r]);
  z3= objAtomos.getZ(aux2u,torS[r])-objAtomos.
      getZ(aux3u,torS[r]);
```

```
v4= objAtomos.getX(aux4u,torS[r])-objAtomos.
    getX(aux3u,torS[r]);
x4= objAtomos.getY(aux4u,torS[r])-objAtomos.
    getY(aux3u,torS[r]);
z4= objAtomos.getZ(aux4u,torS[r])-objAtomos.
    getZ(aux3u,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu = (prod)/(pmod);
angdu = Math.acos(auxdu);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu= -angdu;
}
pangdu= ((getVnU(molS[i][j],molS[i+1][j],molS[
    i+1][j+1],molS[i][j+1],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j],molS[i+1][j
    +1],molS[i][j+1],sem,num,indice))*(1+(Math.
    cos((getN(molS[i][j],molS[i+1][j],molS[i
    +1][j+1],molS[i][j+1],sem,indice)*angdu)-
    getPeriU(molS[i][j],molS[i+1][j],molS[i+1][
    j+1],molS[i][j+1],sem,indice)))));
//arredondamento TINKER
pangdu= (Math.round((pangdu * 10000.0))) /
    10000.0;
pangdu= pangdu+auxu;
```

```
            auxu= pangdu;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdu= ((getVnU(molS[i][j],molS[i+1][j],
                  molS[i+1][j+1],molS[i][j+1],sem,num,
                  indice)/getPath(molS[i][j],molS[i+1][j],
                  molS[i+1][j+1],molS[i][j+1],sem,num,
                  indice))*(1+(Math.cos((getN(molS[i][j],
                  molS[i+1][j],molS[i+1][j+1],molS[i][j+1],
                  sem,indice)*angdu)-getPeriU(molS[i][j],
                  molS[i+1][j],molS[i+1][j+1],molS[i][j+1],
                  sem,indice)))));
              //arredondamento TINKER
              pangdu= (Math.round((pangdu * 10000.0))) /
                  10000.0;
              pangdu= pangdu+auxu;
              auxu= pangdu;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }//fim if
      if((molS[i][j]!="") &&(i<qntdlinha-1)&&(j>=3)&& (
          molS[i][j-3]!="") && (molS[i+1][j-3]!="") && (
          molS[i+1][j-2]!="")&&((molS[i][j-1]=="
          X00000001000")&&(molS[i][j-2]=="X00000001000"))){
        int x=j-3;
        if((molS[i][j]!="") && (molS[i][x]!="") && (molS[i
            +1][x]!="") && (molS[i+1][x+1]!="")){
          if ((molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][x].substring(8, 9).equals("1")) && (
```

```
 molS[i][x].substring(9, 10).equals("1"))&&(
 molS[i+1][x].substring(9, 10).equals("1"))&&(
 molS[i+1][x].substring(8, 9).equals("1"))&&(
 molS[i+1][x+1].substring(8, 9).equals("1"))){
indice=0;
double[] aux1u2 = objAtomos.setAtomoU(molS[i][
    j],torS[r]);
double[] aux2u2= objAtomos.setAtomoU(molS[i][x
    ],torS[r]);
double[] aux3u2= objAtomos.setAtomoU(molS[i
    +1][x],torS[r]);
double[] aux4u2= objAtomos.setAtomoU(molS[i
    +1][x+1],torS[r]);
v1= objAtomos.getX(aux1u2,torS[r])-objAtomos.
    getX(aux2u2,torS[r]);
x1= objAtomos.getY(aux1u2,torS[r])-objAtomos.
    getY(aux2u2,torS[r]);
z1= objAtomos.getZ(aux1u2,torS[r])-objAtomos.
    getZ(aux2u2,torS[r]);
v2= objAtomos.getX(aux3u2,torS[r])-objAtomos.
    getX(aux2u2,torS[r]);
x2= objAtomos.getY(aux3u2,torS[r])-objAtomos.
    getY(aux2u2,torS[r]);
z2= objAtomos.getZ(aux3u2,torS[r])-objAtomos.
    getZ(aux2u2,torS[r]);
v3= objAtomos.getX(aux2u2,torS[r])-objAtomos.
    getX(aux3u2,torS[r]);
x3= objAtomos.getY(aux2u2,torS[r])-objAtomos.
    getY(aux3u2,torS[r]);
z3= objAtomos.getZ(aux2u2,torS[r])-objAtomos.
    getZ(aux3u2,torS[r]);
v4= objAtomos.getX(aux4u2,torS[r])-objAtomos.
    getX(aux3u2,torS[r]);
x4= objAtomos.getY(aux4u2,torS[r])-objAtomos.
    getY(aux3u2,torS[r]);
```

```
z4= objAtomos.getZ(aux4u2,torS[r])-objAtomos.
    getZ(aux3u2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu2= (prod)/(pmod);
angdu2= Math.acos(auxdu2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu2= -angdu2;
}
pangdu2= (getVnU(molS[i][j],molS[i][x],molS[i
    +1][x],molS[i+1][x+1],sem,num,indice)/
    getPath(molS[i][j],molS[i][x],molS[i+1][x],
    molS[i+1][x+1],sem,num,indice)*(1+(Math.cos
    ((getN(molS[i][j],molS[i][x],molS[i+1][x],
    molS[i+1][x+1],sem,indice)*angdu2)-getPeriU
    (molS[i][j],molS[i][x],molS[i+1][x],molS[i
    +1][x+1],sem,indice)))));
//arredondamento TINKER
pangdu2= (Math.round((pangdu2 * 10000.0))) /
    10000.0;
pangdu2= pangdu2+auxu2;
auxu2= pangdu2;
indiceD= indiceD + 1;
ind++;
sem++;
```

```
            while (num==0){
              //transforma pra graus o cos
              pangdu2= (getVnU(molS[i][j],molS[i][x],molS[
                  i+1][x],molS[i+1][x+1],sem,num,indice)/
                  getPath(molS[i][j],molS[i][x],molS[i+1][x
                  ],molS[i+1][x+1],sem,num,indice)*(1+(Math
                  .cos((getN(molS[i][j],molS[i][x],molS[i
                  +1][x],molS[i+1][x+1],sem,indice)*angdu2)
                  -getPeriU(molS[i][j],molS[i][x],molS[i
                  +1][x],molS[i+1][x+1],sem,indice)))));
              //arredondamento TINKER
              pangdu2= (Math.round((pangdu2 * 10000.0))) /
                   10000.0;
              pangdu2= pangdu2+auxu2;
              auxu2= pangdu2;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }else if((molS[i][j]!="") &&(i<qntdlinha-1)&&(j>=1)
          && (molS[i][j-1]!="") && (molS[i+1][j-1]!="") &&
          (molS[i+1][j]!="")&&
            (molS[i][j-1]!="X00000001000")){
        if ((molS[i+1][j-1].substring(0,7).equals("OHR1001
            "))&&(molS[i+1][j].substring(0,7).equals("
            OHR1002"))){

        }else{
        //verificar se todo esta ligados
        if ((molS[i][j].substring(8, 9).equals("1"))&&(
            molS[i][j-1].substring(8, 9).equals("1"))&&(
            molS[i][j-1].substring(9, 10).equals("1"))&&(
            molS[i+1][j-1].substring(9, 10).equals("1"))&&
```

```
(molS[i+1][j-1].substring(8, 9).equals("1"))&&(
    molS[i+1][j].substring(8, 9).equals("1"))){
if ((molS[i][j-1].substring(0, 7).equals("
    OHR2001"))&&(molS[i+1][j-1].substring(0, 7).
    equals("P1P0001"))){

}else{
  indice=0;
  double[] aux1u2 = objAtomos.setAtomoU(molS[i][
      j],torS[r]);
  double[] aux2u2 = objAtomos.setAtomoU(molS[i][
      j-1],torS[r]);
  double[] aux3u2 = objAtomos.setAtomoU(molS[i
      +1][j-1],torS[r]);
  double[] aux4u2 = objAtomos.setAtomoU(molS[i
      +1][j],torS[r]);
  v1= objAtomos.getX(aux1u2,torS[r])-objAtomos.
      getX(aux2u2,torS[r]);
  x1= objAtomos.getY(aux1u2,torS[r])-objAtomos.
      getY(aux2u2,torS[r]);
  z1= objAtomos.getZ(aux1u2,torS[r])-objAtomos.
      getZ(aux2u2,torS[r]);
  v2= objAtomos.getX(aux3u2,torS[r])-objAtomos.
      getX(aux2u2,torS[r]);
  x2= objAtomos.getY(aux3u2,torS[r])-objAtomos.
      getY(aux2u2,torS[r]);
  z2= objAtomos.getZ(aux3u2,torS[r])-objAtomos.
      getZ(aux2u2,torS[r]);
  v3= objAtomos.getX(aux2u2,torS[r])-objAtomos.
      getX(aux3u2,torS[r]);
  x3= objAtomos.getY(aux2u2,torS[r])-objAtomos.
      getY(aux3u2,torS[r]);
  z3= objAtomos.getZ(aux2u2,torS[r])-objAtomos.
      getZ(aux3u2,torS[r]);
  v4= objAtomos.getX(aux4u2,torS[r])-objAtomos.
      getX(aux3u2,torS[r]);
```

```
x4= objAtomos.getY(aux4u2,torS[r])-objAtomos.
    getY(aux3u2,torS[r]);
z4= objAtomos.getZ(aux4u2,torS[r])-objAtomos.
    getZ(aux3u2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu2 = (prod)/(pmod);
angdu2 = Math.acos(auxdu2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu2= -angdu2;
}
pangdu2= ((getVnU(molS[i][j],molS[i][j-1],molS
    [i+1][j-1],molS[i+1][j],sem,num,indice)/
    getPath(molS[i][j],molS[i][j-1],molS[i+1][j
    -1],molS[i+1][j],sem,num,indice))*(1+(Math.
    cos((getN(molS[i][j],molS[i][j-1],molS[i
    +1][j-1],molS[i+1][j],sem,indice)*angdu2)-
    getPeriU(molS[i][j],molS[i][j-1],molS[i+1][
    j-1],molS[i+1][j],sem,indice)))));
//arredondamento TINKER
pangdu2= (Math.round((pangdu2 * 10000.0))) /
    10000.0;
pangdu2= pangdu2+auxu2;
auxu2= pangdu2;
indiceD= indiceD + 1;
```

```
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdu2= ((getVnU(molS[i][j],molS[i][j-1],
                  molS[i+1][j-1],molS[i+1][j],sem,num,
                  indice)/getPath(molS[i][j],molS[i][j-1],
                  molS[i+1][j-1],molS[i+1][j],sem,num,
                  indice))*(1+(Math.cos((getN(molS[i][j],
                  molS[i][j-1],molS[i+1][j-1],molS[i+1][j],
                  sem,indice)*angdu2)-getPeriU(molS[i][j],
                  molS[i][j-1],molS[i+1][j-1],molS[i+1][j],
                  sem,indice)))));
              //arredondamento TINKER
              pangdu2= (Math.round((pangdu2 * 10000.0))) /
                   10000.0;
              pangdu2= pangdu2+auxu2;
              auxu2= pangdu2;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
          }
        }
      }//fim if
      if(((j<qntdcol-4)&&(molS[i][j]!="") &&(i>=1)&& (molS
          [i][j+3]!="") && (molS[i][j+4]!="") && (molS[i
          -1][j+4]!=""))&&((molS[i][j+1]=="X00000001000")
          &&(molS[i][j+2]=="X00000001000"))){
        int x=j+3;
        if((i>=1)&&(x<qntdcol-4)&&(molS[i][j]!="") && (
            molS[i][x]!="") && (molS[i][x+1]!="") && (molS[
            i-1][x+1]!="")){
          if ((molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][x].substring(8, 9).equals("1")) && (
```

```
 molS[i][x].substring(8, 9).equals("1"))&&(
 molS[i][x+1].substring(8, 9).equals("1"))&&(
 molS[i][x+1].substring(9, 10).equals("1"))&&(
 molS[i-1][x+1].substring(9, 10).equals("1")))
 {
indice=0;
double[] aux1u3 = objAtomos.setAtomoU(molS[i][
    j],torS[r]);
double[] aux2u3= objAtomos.setAtomoU(molS[i][x
    ],torS[r]);
double[] aux3u3= objAtomos.setAtomoU(molS[i][x
    +1],torS[r]);
double[] aux4u3= objAtomos.setAtomoU(molS[i
    -1][x+1],torS[r]);
v1= objAtomos.getX(aux1u3,torS[r])-objAtomos.
    getX(aux2u3,torS[r]);
x1= objAtomos.getY(aux1u3,torS[r])-objAtomos.
    getY(aux2u3,torS[r]);
z1= objAtomos.getZ(aux1u3,torS[r])-objAtomos.
    getZ(aux2u3,torS[r]);
v2= objAtomos.getX(aux3u3,torS[r])-objAtomos.
    getX(aux2u3,torS[r]);
x2= objAtomos.getY(aux3u3,torS[r])-objAtomos.
    getY(aux2u3,torS[r]);
z2= objAtomos.getZ(aux3u3,torS[r])-objAtomos.
    getZ(aux2u3,torS[r]);
v3= objAtomos.getX(aux2u3,torS[r])-objAtomos.
    getX(aux3u3,torS[r]);
x3= objAtomos.getY(aux2u3,torS[r])-objAtomos.
    getY(aux3u3,torS[r]);
z3= objAtomos.getZ(aux2u3,torS[r])-objAtomos.
    getZ(aux3u3,torS[r]);
v4= objAtomos.getX(aux4u3,torS[r])-objAtomos.
    getX(aux3u3,torS[r]);
x4= objAtomos.getY(aux4u3,torS[r])-objAtomos.
    getY(aux3u3,torS[r]);
```

```
z4= objAtomos.getZ(aux4u3,torS[r])-objAtomos.
    getZ(aux3u3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu3= (prod)/(pmod);
angdu3= Math.acos(auxdu3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu3= -angdu3;
}
pangdu3= (getVnU(molS[i][j],molS[i][x],molS[i
    ][x+1],molS[i-1][x+1],sem,num,indice)/
    getPath(molS[i][j],molS[i][x],molS[i][x+1],
    molS[i-1][x+1],sem,num,indice)*(1+(Math.cos
    ((getN(molS[i][j],molS[i][x],molS[i][x+1],
    molS[i-1][x+1],sem,indice)*angdu3)-getPeriU
    (molS[i][j],molS[i][x],molS[i][x+1],molS[i
    -1][x+1],sem,indice)))));
//arredondamento TINKER
pangdu3= (Math.round((pangdu3 * 10000.0))) /
    10000.0;
pangdu3= pangdu3+auxu3;
auxu3= pangdu3;
indiceD= indiceD + 1;
ind++;
sem++;
```

```
            while (num==0){
              //transforma pra graus o cos
              pangdu3= (getVnU(molS[i][j],molS[i][x],molS[
                  i][x+1],molS[i-1][x+1],sem,num,indice)/
                  getPath(molS[i][j],molS[i][x],molS[i][x
                  +1],molS[i-1][x+1],sem,num,indice)*(1+(
                  Math.cos((getN(molS[i][j],molS[i][x],molS
                  [i][x+1],molS[i-1][x+1],sem,indice)*
                  angdu3)-getPeriU(molS[i][j],molS[i][x],
                  molS[i][x+1],molS[i-1][x+1],sem,indice)))
                  ));
              //arredondamento TINKER
              pangdu3= (Math.round((pangdu3 * 10000.0))) /
                   10000.0;
              pangdu3= pangdu3+auxu3;
              auxu3= pangdu3;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }
      }else if (((molS[i][j]!="") &&(i>=1)&&(j<qntdcol-2)
          && (molS[i][j+1]!="") && (molS[i][j+2]!="") && (
          molS[i-1][j+2]!=""))
          &&(molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(8, 9).equals("1"))&&(
              molS[i][j+2].substring(8, 9).equals("1"))&&
          (molS[i][j+2].substring(9, 10).equals("1"))&&(
              molS[i-1][j+2].substring(9, 10).equals("1"))
              &&(molS[i][j+1]!="X00000001000")){
          if ((molS[i][j+1].substring(0,7).equals("OHR1001
              "))&&(molS[i][j+2].substring(0,7).equals("
              OHR1002"))){
```

```
}else{
  indice=0;
  double[] aux1u3 = objAtomos.setAtomoU(molS[i][
     j],torS[r]);
  double[] aux2u3 = objAtomos.setAtomoU(molS[i][
     j+1],torS[r]);
  double[] aux3u3 = objAtomos.setAtomoU(molS[i][
     j+2],torS[r]);
  double[] aux4u3 = objAtomos.setAtomoU(molS[i
     -1][j+2],torS[r]);
  v1= objAtomos.getX(aux1u3,torS[r])-objAtomos.
     getX(aux2u3,torS[r]);
  x1= objAtomos.getY(aux1u3,torS[r])-objAtomos.
     getY(aux2u3,torS[r]);
  z1= objAtomos.getZ(aux1u3,torS[r])-objAtomos.
     getZ(aux2u3,torS[r]);
  v2= objAtomos.getX(aux3u3,torS[r])-objAtomos.
     getX(aux2u3,torS[r]);
  x2= objAtomos.getY(aux3u3,torS[r])-objAtomos.
     getY(aux2u3,torS[r]);
  z2= objAtomos.getZ(aux3u3,torS[r])-objAtomos.
     getZ(aux2u3,torS[r]);
  v3= objAtomos.getX(aux2u3,torS[r])-objAtomos.
     getX(aux3u3,torS[r]);
  x3= objAtomos.getY(aux2u3,torS[r])-objAtomos.
     getY(aux3u3,torS[r]);
  z3= objAtomos.getZ(aux2u3,torS[r])-objAtomos.
     getZ(aux3u3,torS[r]);
  v4= objAtomos.getX(aux4u3,torS[r])-objAtomos.
     getX(aux3u3,torS[r]);
  x4= objAtomos.getY(aux4u3,torS[r])-objAtomos.
     getY(aux3u3,torS[r]);
  z4= objAtomos.getZ(aux4u3,torS[r])-objAtomos.
     getZ(aux3u3,torS[r]);
  a1= ((x1*z2)-(x2*z1));
  b1= ((z1*v2)-(v1*z2));
```

```
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu3 = (prod)/(pmod);
angdu3 = Math.acos(auxdu3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu3= -angdu3;
}
pangdu3= ((getVnU(molS[i][j],molS[i][j+1],molS
    [i][j+2],molS[i-1][j+2],sem,num,indice)/
    getPath(molS[i][j],molS[i][j+1],molS[i][j
    +2],molS[i-1][j+2],sem,num,indice))*(1+(
    Math.cos((getN(molS[i][j],molS[i][j+1],molS
    [i][j+2],molS[i-1][j+2],sem,indice)*angdu3)
    -getPeriU(molS[i][j],molS[i][j+1],molS[i][j
    +2],molS[i-1][j+2],sem,indice)))));
//arredondamento TINKER
pangdu3= (Math.round((pangdu3 * 10000.0))) /
    10000.0;
pangdu3= pangdu3+auxu3;
auxu3= pangdu3;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdu3= ((getVnU(molS[i][j],molS[i][j+1],
      molS[i][j+2],molS[i-1][j+2],sem,num,
```

```
                indice)/getPath(molS[i][j],molS[i][j+1],
                molS[i][j+2],molS[i-1][j+2],sem,num,
                indice))*(1+(Math.cos((getN(molS[i][j],
                molS[i][j+1],molS[i][j+2],molS[i-1][j+2],
                sem,indice)*angdu3)-getPeriU(molS[i][j],
                molS[i][j+1],molS[i][j+2],molS[i-1][j+2],
                sem,indice)))));
            //arredondamento TINKER
            pangdu3= (Math.round((pangdu3 * 10000.0))) /
                 10000.0;
            pangdu3= pangdu3+auxu3;
            auxu3= pangdu3;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }

}//fim if
if((molS[i][j]!="") &&(i<qntdlinha-2)&&(j<qntdcol-2)
   && (molS[i][j+1]!="") && (molS[i+1][j+1]!="") &&
    (molS[i+2][j+2]!="")){
//verificar se todo esta ligados
if ((molS[i][j].substring(8, 9).equals("1"))&&(molS[
   i][j+1].substring(8, 9).equals("1"))&&(molS[i][j
   +1].substring(9, 10).equals("1"))&&(molS[i+1][j
   +1].substring(9, 10).equals("1"))&&(molS[i+1][j
   +1].substring(10, 11).equals("1"))&&(molS[i+2][j
   +2].substring(11, 12).equals("1"))){
  indice=0;
  double[] aux1p = objAtomos.setAtomoU(molS[i][j],
     torS[r]);
  double[] aux2p = objAtomos.setAtomoU(molS[i][j+1],
     torS[r]);
  double[] aux3p = objAtomos.setAtomoU(molS[i+1][j
     +1],torS[r]);
```

```
double[] aux4p = objAtomos.setAtomoU(molS[i+2][j
    +2],torS[r]);
v1= objAtomos.getX(aux1p,torS[r])-objAtomos.getX(
    aux2p,torS[r]);
x1= objAtomos.getY(aux1p,torS[r])-objAtomos.getY(
    aux2p,torS[r]);
z1= objAtomos.getZ(aux1p,torS[r])-objAtomos.getZ(
    aux2p,torS[r]);
v2= objAtomos.getX(aux3p,torS[r])-objAtomos.getX(
    aux2p,torS[r]);
x2= objAtomos.getY(aux3p,torS[r])-objAtomos.getY(
    aux2p,torS[r]);
z2= objAtomos.getZ(aux3p,torS[r])-objAtomos.getZ(
    aux2p,torS[r]);
v3= objAtomos.getX(aux2p,torS[r])-objAtomos.getX(
    aux3p,torS[r]);
x3= objAtomos.getY(aux2p,torS[r])-objAtomos.getY(
    aux3p,torS[r]);
z3= objAtomos.getZ(aux2p,torS[r])-objAtomos.getZ(
    aux3p,torS[r]);
v4= objAtomos.getX(aux4p,torS[r])-objAtomos.getX(
    aux3p,torS[r]);
x4= objAtomos.getY(aux4p,torS[r])-objAtomos.getY(
    aux3p,torS[r]);
z4= objAtomos.getZ(aux4p,torS[r])-objAtomos.getZ(
    aux3p,torS[r]);
a1= ((x1*z2)-(x2*z1));//<-- mudar nos outro
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
```

```
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdp = (prod)/(pmod);
angdp = Math.acos(auxdp);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdp= -angdp;
}
pangdp= ((getVnU(molS[i][j],molS[i][j+1],molS[i
    +1][j+1],molS[i+2][j+2],sem,num,indice)/getPath
    (molS[i][j],molS[i][j+1],molS[i+1][j+1],molS[i
    +2][j+2],sem,num,indice))*(1+(Math.cos((getN(
    molS[i][j],molS[i][j+1],molS[i+1][j+1],molS[i
    +2][j+2],sem,indice)*angdp)-getPeriU(molS[i][j
    ],molS[i][j+1],molS[i+1][j+1],molS[i+2][j+2],
    sem,indice)))));
//arredondamento TINKER
pangdp= (Math.round((pangdp * 10000.0))) /
    10000.0;
pangdp= pangdp+auxp;
auxp= pangdp;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdp= ((getVnU(molS[i][j],molS[i][j+1],molS[i
      +1][j+1],molS[i+2][j+2],sem,num,indice)/
      getPath(molS[i][j],molS[i][j+1],molS[i+1][j
      +1],molS[i+2][j+2],sem,num,indice))*(1+(Math.
      cos((getN(molS[i][j],molS[i][j+1],molS[i+1][j
      +1],molS[i+2][j+2],sem,indice)*angdp)-
      getPeriU(molS[i][j],molS[i][j+1],molS[i+1][j
      +1],molS[i+2][j+2],sem,indice)))));
  //arredondamento TINKER
```

```
                pangdp= (Math.round((pangdp * 10000.0))) /
                    10000.0;
                pangdp= pangdp+auxp;
                auxp= pangdp;
                indiceD= indiceD + 1;
                ind++;
                sem++;
              }
            }
            }//fim if
          }//fim else
        }//fim for
      }//fim for
      pangd[r]= pangdp+pangdz+pangdz1+pangdz2+pangdu+pangdu2+
          pangdu3+pangdu4+pangdu5+pangdz3+pangdz4+pangdz5+pangdb4
          +pangdbb4+pangdb5+pangdbb5+pangdb6;
      }//fim for r
      return pangd;
    }
    public double[] angdAnelU (int semente){
      double[] pangd= new double[torS.length];
      double pangdz,pangdz1,pangdz2,pangdu,pangdu2,pangdu3,
          pangdu4,pangdu5,pangdu6,pangdu7,pangdu8,pangdz3,pangdz4
          ,pangdz5,
      pangdz6,pangdz7,pangdz8,pangdb4,pangdbb4,pangdb5,pangdbb5,
          pangdb6;
      double angdb4,angdbb4,angdb5,angdbb5,angdb6,angdz,angdz2,
          angdz3,angdz4,angdz5,angdz6,angdz7,angdz8,angdu,angdu2,
          angdu3,angdu4,angdu5,angdu6,angdu7,angdu8;
      double auxdb4,auxdbb4,auxdb5,auxdbb5,auxdb6,auxdz,auxdz2,
          auxdz3,auxdz4,auxdz5,auxdz6,auxdz7,auxdz8,auxdu,auxdu2,
          auxdu3,auxdu4,auxdu5,auxdu6,auxdu7,auxdu8;
      double auxb4,auxbb4,auxb5,auxbb5,auxb6,auxz,auxz1,auxz2,
          auxz3,auxz4,auxz5,auxz6,auxz7,auxz8,auxu,auxu2,auxu3,
          auxu4,auxu5,auxu6,auxu7,auxu8;
      int indiceD= 0;
```

```
double a1, a2, b1, b2, c1, c2, v1, x1, z1, v2, x2, z2, v3,
    x3, z3, v4, x4, z4, x1s, y1s, z1s;
double aux = 0.0;
double[] aux1b4, aux2b4, aux3b4, aux4b4, aux1bb4, aux2bb4,
    aux3bb4, aux4bb4, aux1b5, aux2b5, aux3b5, aux4b5,
   aux1bb5, aux2bb5, aux3bb5, aux4bb5, aux1b6, aux2b6,
   aux3b6, aux4b6,
    aux1u7, aux2u7, aux3u7, aux4u7;
double pmod, mod, mod2, prod;
int ind= 0;
//numero de interacoes
int cont=0;


String log3="";
double energia=0.0;
for(int r= 0; r< torS.length; r++){
  sem= semente;
  num = 1;
  indice = 0;
  aux1b4= new double[3];
  aux2b4= new double[3];
  aux3b4= new double[3];
  aux4b4= new double[3];
  aux1bb4= new double[3];
  aux2bb4= new double[3];
  aux3bb4= new double[3];
  aux4bb4= new double[3];
  aux1b5= new double[3];
  aux2b5= new double[3];
  aux3b5= new double[3];
  aux4b5= new double[3];
  aux1b6= new double[3];
  aux2b6= new double[3];
  aux3b6= new double[3];
  aux4b6= new double[3];
```

```
aux1bb5= new double[3];
aux2bb5= new double[3];
aux3bb5= new double[3];
aux4bb5= new double[3];
aux1u7= new double[3];
aux2u7= new double[3];
aux3u7= new double[3];
aux4u7= new double[3];
pmod= 0;
mod= 0;
mod2= 0;
prod=0;
auxz=0;
auxz2=0;
auxz3=0;
auxz4=0;
auxz5=0;
auxz6=0;
auxz7=0;
auxz8=0;
auxu=0;
auxu2=0;
auxu3=0;
auxu4=0;
auxu5=0;
auxu6=0;
auxu7=0;
auxu8=0;
auxz=0.0;
auxdz=0.0;
auxdz2=0.0;
auxdz3=0.0;
auxdz4=0.0;
auxdz5=0.0;
auxdz6=0.0;
auxdz7=0.0;
```

```
auxdz8=0.0;
auxdu=0.0;
auxdu2=0.0;
auxdu3=0.0;
auxdu4=0.0;
auxdu5=0.0;
auxdu6=0.0;
auxdu7=0.0;
auxdu8=0.0;
pangdb4=0.0;
pangdbb4=0.0;
pangdb5=0.0;
pangdbb5=0.0;
pangdb6=0.0;
auxdb4=0.0;
auxdbb4=0.0;
auxdb5=0.0;
auxdbb5=0.0;
auxdb6=0.0;
angdb4=0.0;
angdbb4=0.0;
angdb5=0.0;
angdbb5=0.0;
angdb6=0.0;
auxb4=0.0;
auxbb4=0.0;
auxz1=0.0;
auxb5=0.0;
auxbb5=0.0;
auxb6=0.0;
pangdz=0;
pangdz1=0;
pangdz2=0;
pangdz3=0;
pangdz4=0;
pangdz5=0;
```

```
        pangdz6=0;
        pangdz7=0;
        pangdz8=0;
        pangdu=0;
        pangdu2=0;
        pangdu3=0;
        pangdu4=0;
        pangdu5=0;
        pangdu6=0;
        pangdu7=0;
        pangdu8=0;
      for (int i=0; i<qntdlinha; i++){
        for (int j= 0; j< qntdcol; j++){
          aux=0.0;
          if (molS[i][j].equals("X00000001000")){
          }else{
            if(((i<qntdlinha-2)&&(j<qntdcol-2)&&(molS[i][j]!="")
                && (molS[i+1][j]!="") && (molS[i+2][j+1]!="") &&
                (molS[i+1][j+2]!=""))){
                if ((((molS[i][j].substring(9, 10).equals("1"))
                    &&(molS[i+1][j].substring(9, 10).equals("1"))
                    ))&&(molS[i+1][j].substring(10, 11).equals
                    ("1"))&&(molS[i+2][j+1].substring(11, 12).
                    equals("1"))&&(molS[i+2][j+1].substring(10,
                    11).equals("1"))&(molS[i+1][j+2].substring
                    (11, 12).equals("1"))){
                indice=0;
                //isolar a b e c dos planos
                aux1b4= objAtomos.setAtomoU(molS[i][j],torS[r]);
                aux2b4= objAtomos.setAtomoU(molS[i+1][j],torS[r
                    ]);
                aux3b4= objAtomos.setAtomoU(molS[i+2][j+1],torS[
                    r]);
                aux4b4= objAtomos.setAtomoU(molS[i+1][j+2],torS[
                    r]);
```

```
v1= objAtomos.getX(aux1b4,torS[r])-objAtomos.
    getX(aux2b4,torS[r]);
x1= objAtomos.getY(aux1b4,torS[r])-objAtomos.
    getY(aux2b4,torS[r]);
z1= objAtomos.getZ(aux1b4,torS[r])-objAtomos.
    getZ(aux2b4,torS[r]);
v2= objAtomos.getX(aux3b4,torS[r])-objAtomos.
    getX(aux2b4,torS[r]);
x2= objAtomos.getY(aux3b4,torS[r])-objAtomos.
    getY(aux2b4,torS[r]);
z2= objAtomos.getZ(aux3b4,torS[r])-objAtomos.
    getZ(aux2b4,torS[r]);
v3= objAtomos.getX(aux2b4,torS[r])-objAtomos.
    getX(aux3b4,torS[r]);
x3= objAtomos.getY(aux2b4,torS[r])-objAtomos.
    getY(aux3b4,torS[r]);
z3= objAtomos.getZ(aux2b4,torS[r])-objAtomos.
    getZ(aux3b4,torS[r]);
v4= objAtomos.getX(aux4b4,torS[r])-objAtomos.
    getX(aux3b4,torS[r]);
x4= objAtomos.getY(aux4b4,torS[r])-objAtomos.
    getY(aux3b4,torS[r]);
z4= objAtomos.getZ(aux4b4,torS[r])-objAtomos.
    getZ(aux3b4,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
```

```
auxdb4= (prod)/(pmod);
angdb4= Math.acos(auxdb4);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb4= -angdb4;
}
pangdb4= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j+1],molS[i+1][j+2],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+1][j+2],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+1][j+2],sem,indice)*angdb4)-
    getPeriU(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+1][j+2],sem,indice)))));
//arredondamento TINKER
pangdb4= (Math.round((pangdb4 * 10000.0))) /
    10000.0;
pangdb4= pangdb4+auxb4;
auxb4= pangdb4;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdb4= (getVnU(molS[i][j],molS[i+1][j],molS[
      i+2][j+1],molS[i+1][j+2],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j],molS[i+2][j
      +1],molS[i+1][j+2],sem,num,indice)*(1+(Math
      .cos((getN(molS[i][j],molS[i+1][j],molS[i
      +2][j+1],molS[i+1][j+2],sem,indice)*angdb4)
      -getPeriU(molS[i][j],molS[i+1][j],molS[i
      +2][j+1],molS[i+1][j+2],sem,indice)))));
  //arredondamento TINKER
  pangdb4= (Math.round((pangdb4 * 10000.0))) /
      10000.0;
  pangdb4= pangdb4+auxb4;
```

```
            auxb4= pangdb4;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim if
      }//fim if
      if(((i<qntdlinha-1)&&(j<qntdcol-3)&&(molS[i][j]!="")
          && (molS[i+1][j+1]!="") && (molS[i][j+2]!="") &&
          (molS[i][j+3]!=""))){
          if ((((molS[i][j].substring(10, 11).equals("1"))
              &&(molS[i+1][j+1].substring(11, 12).equals
              ("1"))))&&(molS[i+1][j+1].substring(10, 11).
              equals("1"))&&(molS[i][j+2].substring(11, 12)
              .equals("1"))&&(molS[i][j+2].substring(8, 9).
              equals("1"))&(molS[i][j+3].substring(8, 9).
              equals("1"))){
          indice=0;
          //isolar a b e c dos planos
          aux1bb4= objAtomos.setAtomoU(molS[i][j],torS[r])
              ;
          aux2bb4= objAtomos.setAtomoU(molS[i+1][j+1],torS
              [r]);
          aux3bb4= objAtomos.setAtomoU(molS[i][j+2],torS[r
              ]);
          aux4bb4= objAtomos.setAtomoU(molS[i][j+3],torS[r
              ]);
          v1= objAtomos.getX(aux1bb4,torS[r])-objAtomos.
              getX(aux2bb4,torS[r]);
          x1= objAtomos.getY(aux1bb4,torS[r])-objAtomos.
              getY(aux2bb4,torS[r]);
          z1= objAtomos.getZ(aux1bb4,torS[r])-objAtomos.
              getZ(aux2bb4,torS[r]);
          v2= objAtomos.getX(aux3bb4,torS[r])-objAtomos.
              getX(aux2bb4,torS[r]);
```

```
x2= objAtomos.getY(aux3bb4,torS[r])-objAtomos.
    getY(aux2bb4,torS[r]);
z2= objAtomos.getZ(aux3bb4,torS[r])-objAtomos.
    getZ(aux2bb4,torS[r]);
v3= objAtomos.getX(aux2bb4,torS[r])-objAtomos.
    getX(aux3bb4,torS[r]);
x3= objAtomos.getY(aux2bb4,torS[r])-objAtomos.
    getY(aux3bb4,torS[r]);
z3= objAtomos.getZ(aux2bb4,torS[r])-objAtomos.
    getZ(aux3bb4,torS[r]);
v4= objAtomos.getX(aux4bb4,torS[r])-objAtomos.
    getX(aux3bb4,torS[r]);
x4= objAtomos.getY(aux4bb4,torS[r])-objAtomos.
    getY(aux3bb4,torS[r]);
z4= objAtomos.getZ(aux4bb4,torS[r])-objAtomos.
    getZ(aux3bb4,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdbb4= (prod)/(pmod);
angdbb4= Math.acos(auxdbb4);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb4= -angdbb4;
}
pangdbb4= (getVnU(molS[i][j],molS[i+1][j+1],molS
    [i][j+2],molS[i][j+3],sem,num,indice)/getPath
```

```
            (molS[i][j],molS[i+1][j+1],molS[i][j+2],molS[
            i][j+3],sem,num,indice)*(1+(Math.cos((getN(
            molS[i][j],molS[i+1][j+1],molS[i][j+2],molS[i
            ][j+3],sem,indice)*angdbb4)-getPeriU(molS[i][
            j],molS[i+1][j+1],molS[i][j+2],molS[i][j+3],
            sem,indice)))));
          //arredondamento TINKER
          pangdbb4= (Math.round((pangdbb4 * 10000.0))) /
              10000.0;
          pangdbb4= pangdbb4+auxbb4;
          auxbb4= pangdbb4;
          indiceD= indiceD + 1;
          ind++;
          sem++;
          while (num==0){
            //transforma pra graus o cos
            pangdbb4= (getVnU(molS[i][j],molS[i+1][j+1],
                molS[i][j+2],molS[i][j+3],sem,num,indice)/
                getPath(molS[i][j],molS[i+1][j+1],molS[i][j
                +2],molS[i][j+3],sem,num,indice)*(1+(Math.
                cos((getN(molS[i][j],molS[i+1][j+1],molS[i
                ][j+2],molS[i][j+3],sem,indice)*angdbb4)-
                getPeriU(molS[i][j],molS[i+1][j+1],molS[i][
                j+2],molS[i][j+3],sem,indice)))));
            //arredondamento TINKER
            pangdbb4= (Math.round((pangdbb4 * 10000.0))) /
                 10000.0;
            pangdbb4= pangdbb4+auxbb4;
            auxbb4= pangdbb4;
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }//fim if
      }//fim if
```

```
if (((i<qntdlinha-2)&&(j<qntdcol-3)&&(molS[i][j]!="")
    && (molS[i+1][j+1]!="") && (molS[i+1][j+2]!="")
   && (molS[i+2][j+3]!="")))){
  if (((((molS[i][j].substring(10, 11).equals("1"))
     &&(molS[i+1][j+1].substring(11, 12).equals("1")
     )))&&(molS[i+1][j+1].substring(8, 9).equals
     ("1"))&&(molS[i+1][j+2].substring(8, 9).equals
     ("1"))&&(molS[i+1][j+2].substring(10, 11).
     equals("1"))&(molS[i+2][j+3].substring(11, 12).
     equals("1"))){
    indice=0;
      //isolar a b e c dos planos
      aux1b5= objAtomos.setAtomoU(molS[i][j],torS[r
          ]);
      aux2b5= objAtomos.setAtomoU(molS[i+1][j+1],
          torS[r]);
      aux3b5= objAtomos.setAtomoU(molS[i+1][j+2],
          torS[r]);
      aux4b5= objAtomos.setAtomoU(molS[i+2][j+3],
          torS[r]);
      v1= objAtomos.getX(aux1b5,torS[r])-objAtomos.
          getX(aux2b5,torS[r]);
      x1= objAtomos.getY(aux1b5,torS[r])-objAtomos.
          getY(aux2b5,torS[r]);
      z1= objAtomos.getZ(aux1b5,torS[r])-objAtomos.
          getZ(aux2b5,torS[r]);
      v2= objAtomos.getX(aux3b5,torS[r])-objAtomos.
          getX(aux2b5,torS[r]);
      x2= objAtomos.getY(aux3b5,torS[r])-objAtomos.
          getY(aux2b5,torS[r]);
      z2= objAtomos.getZ(aux3b5,torS[r])-objAtomos.
          getZ(aux2b5,torS[r]);
      v3= objAtomos.getX(aux2b5,torS[r])-objAtomos.
          getX(aux3b5,torS[r]);
      x3= objAtomos.getY(aux2b5,torS[r])-objAtomos.
          getY(aux3b5,torS[r]);
```

```
z3= objAtomos.getZ(aux2b5,torS[r])-objAtomos.
    getZ(aux3b5,torS[r]);
v4= objAtomos.getX(aux4b5,torS[r])-objAtomos.
    getX(aux3b5,torS[r]);
x4= objAtomos.getY(aux4b5,torS[r])-objAtomos.
    getY(aux3b5,torS[r]);
z4= objAtomos.getZ(aux4b5,torS[r])-objAtomos.
    getZ(aux3b5,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb5= (prod)/(pmod);
angdb5= Math.acos(auxdb5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb5= -angdb5;
}
pangdb5= (getVnU(molS[i][j],molS[i+1][j+1],
    molS[i+1][j+2],molS[i+2][j+3],sem,num,
    indice)/getPath(molS[i][j],molS[i+1][j+1],
    molS[i+1][j+2],molS[i+2][j+3],sem,num,
    indice)*(1+(Math.cos((getN(molS[i][j],molS[
    i+1][j+1],molS[i+1][j+2],molS[i+2][j+3],sem
    ,indice)*angdb5)-getPeriU(molS[i][j],molS[i
    +1][j+1],molS[i+1][j+2],molS[i+2][j+3],sem,
    indice)))));
//arredondamento TINKER
```

```
            pangdb5= (Math.round((pangdb5 * 10000.0))) /
                10000.0;
            pangdb5= pangdb5+auxb5;
            auxb5= pangdb5;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdb5= (getVnU(molS[i][j],molS[i+1][j+1],
                  molS[i+1][j+2],molS[i+2][j+3],sem,num,
                  indice)/getPath(molS[i][j],molS[i+1][j
                  +1],molS[i+1][j+2],molS[i+2][j+3],sem,num
                  ,indice)*(1+(Math.cos((getN(molS[i][j],
                  molS[i+1][j+1],molS[i+1][j+2],molS[i+2][j
                  +3],sem,indice)*angdb5)-getPeriU(molS[i][
                  j],molS[i+1][j+1],molS[i+1][j+2],molS[i
                  +2][j+3],sem,indice)))));
              //arredondamento TINKER
              pangdb5= (Math.round((pangdb5 * 10000.0))) /
                  10000.0;
              pangdb5= pangdb5+auxb5;
              auxb5= pangdb5;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }//fim if
        }//fim if
        if(((i>=1)&&(molS[i][j]!="") && (j<qntdcol-3)&&(molS
            [i][j+1]!="") && (molS[i-1][j+2]!="") && (molS[i
            -1][j+3]!=""))){
          if ((molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(10, 11).equals("1"))&&(
              molS[i-1][j+2].substring(11, 12).equals("1"))
```

```
 &&(molS[i-1][j+2].substring(8, 9).equals("1"))
 &&(molS[i-1][j+3].substring(8, 9).equals("1")))
 {
indice=0;
aux1bb5= objAtomos.setAtomoU(molS[i][j],torS[r])
    ;
     aux2bb5= objAtomos.setAtomoU(molS[i][j+1],
         torS[r]);
     aux3bb5= objAtomos.setAtomoU(molS[i-1][j+2],
         torS[r]);
     aux4bb5= objAtomos.setAtomoU(molS[i-1][j+3],
         torS[r]);
     v1= objAtomos.getX(aux1bb5,torS[r])-
         objAtomos.getX(aux2bb5,torS[r]);
     x1= objAtomos.getY(aux1bb5,torS[r])-
         objAtomos.getY(aux2bb5,torS[r]);
     z1= objAtomos.getZ(aux1bb5,torS[r])-
         objAtomos.getZ(aux2bb5,torS[r]);
     v2= objAtomos.getX(aux3bb5,torS[r])-
         objAtomos.getX(aux2bb5,torS[r]);
     x2= objAtomos.getY(aux3bb5,torS[r])-
         objAtomos.getY(aux2bb5,torS[r]);
     z2= objAtomos.getZ(aux3bb5,torS[r])-
         objAtomos.getZ(aux2bb5,torS[r]);
     v3= objAtomos.getX(aux2bb5,torS[r])-
         objAtomos.getX(aux3bb5,torS[r]);
     x3= objAtomos.getY(aux2bb5,torS[r])-
         objAtomos.getY(aux3bb5,torS[r]);
     z3= objAtomos.getZ(aux2bb5,torS[r])-
         objAtomos.getZ(aux3bb5,torS[r]);
     v4= objAtomos.getX(aux4bb5,torS[r])-
         objAtomos.getX(aux3bb5,torS[r]);
     x4= objAtomos.getY(aux4bb5,torS[r])-
         objAtomos.getY(aux3bb5,torS[r]);
     z4= objAtomos.getZ(aux4bb5,torS[r])-
         objAtomos.getZ(aux3bb5,torS[r]);
```

```
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)
    +Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2
    ,2)+Math.pow(c2,2));
pmod= mod*mod2;
auxdbb5= (prod)/(pmod);
angdbb5= Math.acos(auxdbb5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdbb5= -angdbb5;
}
//transforma pra graus o cos
pangdbb5= (getVnU(molS[i][j],molS[i][j+1],
    molS[i-1][j+2],molS[i-1][j+3],sem,num,
    indice)/getPath(molS[i][j],molS[i][j+1],
    molS[i-1][j+2],molS[i-1][j+3],sem,num,
    indice)*(1+(Math.cos((getN(molS[i][j],
    molS[i][j+1],molS[i-1][j+2],molS[i-1][j
    +3],sem,indice)*angdbb5)-getPeriU(molS[i
    ][j],molS[i][j+1],molS[i-1][j+2],molS[i
    -1][j+3],sem,indice)))));
//arredondamento TINKER
pangdbb5= (Math.round((pangdbb5 * 10000.0)))
     / 10000.0;
pangdbb5= pangdbb5+auxbb5;
auxbb5= pangdbb5;
indiceD= indiceD + 1;
ind++;
sem++;
```

```
            while (num==0){
              //transforma pra graus o cos
              pangdbb5= (getVnU(molS[i][j],molS[i][j+1],
                  molS[i-1][j+2],molS[i-1][j+3],sem,num,
                  indice)/getPath(molS[i][j],molS[i][j
                  +1],molS[i-1][j+2],molS[i-1][j+3],sem,
                  num,indice)*(1+(Math.cos((getN(molS[i][
                  j],molS[i][j+1],molS[i-1][j+2],molS[i
                  -1][j+3],sem,indice)*angdbb5)-getPeriU(
                  molS[i][j],molS[i][j+1],molS[i-1][j+2],
                  molS[i-1][j+3],sem,indice)))));
              //arredondamento TINKER
              pangdbb5= (Math.round((pangdbb5 * 10000.0)
                  )) / 10000.0;
              pangdbb5= pangdbb5+auxbb5;
              auxbb5= pangdbb5;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
        }
      }//fim if
      if(((j<qntdcol-2)&&(i<qntdlinha-2)&&(molS[i][j]!="")
          && (molS[i][j+1]!="") && (molS[i+1][j+1]!="") &&
          (molS[i+2][j]!="")))){
        if ((molS[i][j].substring(8, 9).equals("1"))&&(
            molS[i][j+1].substring(8, 9).equals("1"))&&(
            molS[i][j+1].substring(9, 10).equals("1"))&&(
            molS[i+1][j+1].substring(9, 10).equals("1"))&&(
            molS[i+1][j+1].substring(11, 12).equals("1"))
            &&(molS[i+2][j].substring(10, 11).equals("1")))
            {
             indice=0;
             double[] aux1u= objAtomos.setAtomoU(molS[i][j
                 ],torS[r]);
```

```
double[] aux2u= objAtomos.setAtomoU(molS[i][j
    +1],torS[r]);
double[] aux3u= objAtomos.setAtomoU(molS[i+1][
    j+1],torS[r]);
double[] aux4u= objAtomos.setAtomoU(molS[i+2][
    j],torS[r]);
v1= objAtomos.getX(aux1u,torS[r])-objAtomos.
    getX(aux2u,torS[r]);
x1= objAtomos.getY(aux1u,torS[r])-objAtomos.
    getY(aux2u,torS[r]);
z1= objAtomos.getZ(aux1u,torS[r])-objAtomos.
    getZ(aux2u,torS[r]);
v2= objAtomos.getX(aux3u,torS[r])-objAtomos.
    getX(aux2u,torS[r]);
x2= objAtomos.getY(aux3u,torS[r])-objAtomos.
    getY(aux2u,torS[r]);
z2= objAtomos.getZ(aux3u,torS[r])-objAtomos.
    getZ(aux2u,torS[r]);
v3= objAtomos.getX(aux2u,torS[r])-objAtomos.
    getX(aux3u,torS[r]);
x3= objAtomos.getY(aux2u,torS[r])-objAtomos.
    getY(aux3u,torS[r]);
z3= objAtomos.getZ(aux2u,torS[r])-objAtomos.
    getZ(aux3u,torS[r]);
v4= objAtomos.getX(aux4u,torS[r])-objAtomos.
    getX(aux3u,torS[r]);
x4= objAtomos.getY(aux4u,torS[r])-objAtomos.
    getY(aux3u,torS[r]);
z4= objAtomos.getZ(aux4u,torS[r])-objAtomos.
    getZ(aux3u,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
```

```
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu= (prod)/(pmod);
angdu= Math.acos(auxdu);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu= -angdu;
}
pangdu= (getVnU(molS[i][j],molS[i][j+1],molS[i
    +1][j+1],molS[i+2][j],sem,num,indice)/
    getPath(molS[i][j],molS[i][j+1],molS[i+1][j
    +1],molS[i+2][j],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i][j+1],molS[i
    +1][j+1],molS[i+2][j],sem,indice)*angdu)-
    getPeriU(molS[i][j],molS[i][j+1],molS[i+1][
    j+1],molS[i+2][j],sem,indice)))));
//arredondamento TINKER
pangdu= (Math.round((pangdu * 10000.0))) /
    10000.0;
pangdu= pangdu+auxu;
auxbb5= pangdbb5;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdu= (getVnU(molS[i][j],molS[i][j+1],molS
      [i+1][j+1],molS[i+2][j],sem,num,indice)/
      getPath(molS[i][j],molS[i][j+1],molS[i
      +1][j+1],molS[i+2][j],sem,num,indice)
      *(1+(Math.cos((getN(molS[i][j],molS[i][j
      +1],molS[i+1][j+1],molS[i+2][j],sem,
```

```
                    indice)*angdu)-getPeriU(molS[i][j],molS[i
                    ][j+1],molS[i+1][j+1],molS[i+2][j],sem,
                    indice)))));
                //arredondamento TINKER
                pangdu= (Math.round((pangdu * 10000.0))) /
                    10000.0;
                pangdu= pangdu+auxu;
                auxbb5= pangdbb5;
                indiceD= indiceD + 1;
                ind++;
                sem++;
              }
            }
          }//fim if
          if(((j<qntdcol-3)&&(i<qntdlinha-1)&&(molS[i][j]!="")
              && (molS[i][j+1]!="") && (molS[i+1][j+2]!="") &&
              (molS[i][j+3]!=""))){
            if ((molS[i][j].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(8, 9).equals("1"))&&(
              molS[i][j+1].substring(10, 11).equals("1"))&&(
              molS[i+1][j+2].substring(11, 12).equals("1"))
              &&(molS[i+1][j+2].substring(10, 11).equals("1")
              )&&(molS[i][j+3].substring(11, 12).equals("1"))
              ){
                indice=0;
                aux1b6= objAtomos.setAtomoU(molS[i][j],torS[r
                    ]);
                aux2b6= objAtomos.setAtomoU(molS[i][j+1],torS[
                    r]);
                aux3b6= objAtomos.setAtomoU(molS[i+1][j+2],
                    torS[r]);
                aux4b6= objAtomos.setAtomoU(molS[i][j+3],torS[
                    r]);
                v1= objAtomos.getX(aux1b6,torS[r])-objAtomos.
                    getX(aux2b6,torS[r]);
```

```
x1= objAtomos.getY(aux1b6,torS[r])-objAtomos.
    getY(aux2b6,torS[r]);
z1= objAtomos.getZ(aux1b6,torS[r])-objAtomos.
    getZ(aux2b6,torS[r]);
v2= objAtomos.getX(aux3b6,torS[r])-objAtomos.
    getX(aux2b6,torS[r]);
x2= objAtomos.getY(aux3b6,torS[r])-objAtomos.
    getY(aux2b6,torS[r]);
z2= objAtomos.getZ(aux3b6,torS[r])-objAtomos.
    getZ(aux2b6,torS[r]);
v3= objAtomos.getX(aux2b6,torS[r])-objAtomos.
    getX(aux3b6,torS[r]);
x3= objAtomos.getY(aux2b6,torS[r])-objAtomos.
    getY(aux3b6,torS[r]);
z3= objAtomos.getZ(aux2b6,torS[r])-objAtomos.
    getZ(aux3b6,torS[r]);
v4= objAtomos.getX(aux4b6,torS[r])-objAtomos.
    getX(aux3b6,torS[r]);
x4= objAtomos.getY(aux4b6,torS[r])-objAtomos.
    getY(aux3b6,torS[r]);
z4= objAtomos.getZ(aux4b6,torS[r])-objAtomos.
    getZ(aux3b6,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdb6= (prod)/(pmod);
angdb6= Math.acos(auxdb6);
```

```
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdb6= -angdb6;
}
pangdb6= (getVnU(molS[i][j],molS[i][j+1],molS[
    i+1][j+2],molS[i][j+3],sem,num,indice)/
    getPath(molS[i][j],molS[i][j+1],molS[i+1][j
    +2],molS[i][j+3],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i][j+1],molS[i
    +1][j+2],molS[i][j+3],sem,indice)*angdb6)-
    getPeriU(molS[i][j],molS[i][j+1],molS[i+1][
    j+2],molS[i][j+3],sem,indice)))));
//arredondamento TINKER
pangdb6= (Math.round((pangdb6 * 10000.0))) /
    10000.0;
pangdb6= pangdb6+auxb6;
auxb6= pangdb6;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdb6= (getVnU(molS[i][j],molS[i][j+1],
      molS[i+1][j+2],molS[i][j+3],sem,num,
      indice)/getPath(molS[i][j],molS[i][j+1],
      molS[i+1][j+2],molS[i][j+3],sem,num,
      indice)*(1+(Math.cos((getN(molS[i][j],
      molS[i][j+1],molS[i+1][j+2],molS[i][j+3],
      sem,indice)*angdb6)-getPeriU(molS[i][j],
      molS[i][j+1],molS[i+1][j+2],molS[i][j+3],
      sem,indice)))));
  //arredondamento TINKER
  pangdb6= (Math.round((pangdb6 * 10000.0))) /
      10000.0;
  pangdb6= pangdb6+auxb6;
  auxb6= pangdb6;
```

```
                indiceD= indiceD + 1;
                ind++;
                sem++;
            }
        }//fim if
    }//fim if
    if((i>=2)&&(j<qntdcol-2)&&(molS[i][j]!="") && (molS[
        i-1][j]!="") && (molS[i-2][j+1]!="")&& (molS[i
        -1][j+2]!="")&&((molS[i][j].substring(9, 10).
        equals("1"))&&(molS[i-1][j].substring(9, 10).
        equals("1"))&&(molS[i-1][j].substring(10, 11).
        equals("1"))&&(molS[i-2][j+1].substring(11, 12).
        equals("1"))&&(molS[i-2][j+1].substring(10, 11).
        equals("1"))&&(molS[i-1][j+2].substring(11, 12).
        equals("1")))){
        indice=0;
        double[] aux1z = objAtomos.setAtomoU(molS[i][j],
            torS[r]);
        double[] aux2z = objAtomos.setAtomoU(molS[i-1][j],
            torS[r]);
        double[] aux3z = objAtomos.setAtomoU(molS[i-2][j
            +1],torS[r]);
        double[] aux4z = objAtomos.setAtomoU(molS[i-1][j
            +2],torS[r]);
        v1= objAtomos.getX(aux1z,torS[r])-objAtomos.getX(
            aux2z,torS[r]);
        x1= objAtomos.getY(aux1z,torS[r])-objAtomos.getY(
            aux2z,torS[r]);
        z1= objAtomos.getZ(aux1z,torS[r])-objAtomos.getZ(
            aux2z,torS[r]);
        v2= objAtomos.getX(aux3z,torS[r])-objAtomos.getX(
            aux2z,torS[r]);
        x2= objAtomos.getY(aux3z,torS[r])-objAtomos.getY(
            aux2z,torS[r]);
        z2= objAtomos.getZ(aux3z,torS[r])-objAtomos.getZ(
            aux2z,torS[r]);
```

```
v3= objAtomos.getX(aux2z,torS[r])-objAtomos.getX(
    aux3z,torS[r]);
x3= objAtomos.getY(aux2z,torS[r])-objAtomos.getY(
    aux3z,torS[r]);
z3= objAtomos.getZ(aux2z,torS[r])-objAtomos.getZ(
    aux3z,torS[r]);
v4= objAtomos.getX(aux4z,torS[r])-objAtomos.getX(
    aux3z,torS[r]);
x4= objAtomos.getY(aux4z,torS[r])-objAtomos.getY(
    aux3z,torS[r]);
z4= objAtomos.getZ(aux4z,torS[r])-objAtomos.getZ(
    aux3z,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdz = (prod)/(pmod);
angdz = Math.acos(auxdz);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz= -angdz;
}
pangdz = (getVnU(molS[i][j],molS[i-1][j],molS[i
    -2][j+1],molS[i-1][j+2],sem,num,indice)/getPath
    (molS[i][j],molS[i-1][j],molS[i-2][j+1],molS[i
    -1][j+2],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i-1][j],molS[i-2][j+1],molS[i
    -1][j+2],sem,indice)*angdz)-getPeriU(molS[i][j
```

```
        ],molS[i-1][j],molS[i-2][j+1],molS[i-1][j+2],
        sem,indice)))));
    //arredondamento TINKER
    pangdz= (Math.round((pangdz * 10000.0))) /
        10000.0;
    pangdz= pangdz+auxz;
    auxz= pangdz;
    indiceD= indiceD + 1;
    ind++;
    sem++;
    while (num==0){
      //transforma pra graus o cos
      pangdz = (getVnU(molS[i][j],molS[i-1][j],molS[i
          -2][j+1],molS[i-1][j+2],sem,num,indice)/
          getPath(molS[i][j],molS[i-1][j],molS[i-2][j
          +1],molS[i-1][j+2],sem,num,indice)*(1+(Math.
          cos((getN(molS[i][j],molS[i-1][j],molS[i-2][j
          +1],molS[i-1][j+2],sem,indice)*angdz)-
          getPeriU(molS[i][j],molS[i-1][j],molS[i-2][j
          +1],molS[i-1][j+2],sem,indice)))));
      //arredondamento TINKER
      pangdz= (Math.round((pangdz * 10000.0))) /
          10000.0;
      pangdz= pangdz+auxz;
      auxz= pangdz;
      indiceD= indiceD + 1;
      ind++;
      sem++;
    }

  }//fim if
  if((i>=2)&&(j<qntdcol-2)&&(molS[i][j]!="") && (molS[
      i-1][j+1]!="") && (molS[i-2][j+1]!="")&& (molS[i
      -2][j+2]!="")&&((molS[i][j].substring(10, 11).
      equals("1"))&&(molS[i-1][j+1].substring(11, 12).
      equals("1"))&&(molS[i-1][j+1].substring(9, 10).
```

```
 equals("1"))&&(molS[i-2][j+1].substring(9, 10).
 equals("1"))&&(molS[i-2][j+1].substring(8, 9).
 equals("1"))&&(molS[i-2][j+2].substring(8, 9).
 equals("1")))){
indice=0;
double[] aux1z1 = objAtomos.setAtomoU(molS[i][j],
    torS[r]);
double[] aux2z1 = objAtomos.setAtomoU(molS[i-1][j
    +1],torS[r]);
double[] aux3z1 = objAtomos.setAtomoU(molS[i-2][j
    +1],torS[r]);
double[] aux4z1 = objAtomos.setAtomoU(molS[i-2][j
    +2],torS[r]);
v1= objAtomos.getX(aux1z1,torS[r])-objAtomos.getX(
    aux2z1,torS[r]);
x1= objAtomos.getY(aux1z1,torS[r])-objAtomos.getY(
    aux2z1,torS[r]);
z1= objAtomos.getZ(aux1z1,torS[r])-objAtomos.getZ(
    aux2z1,torS[r]);
v2= objAtomos.getX(aux3z1,torS[r])-objAtomos.getX(
    aux2z1,torS[r]);
x2= objAtomos.getY(aux3z1,torS[r])-objAtomos.getY(
    aux2z1,torS[r]);
z2= objAtomos.getZ(aux3z1,torS[r])-objAtomos.getZ(
    aux2z1,torS[r]);
v3= objAtomos.getX(aux2z1,torS[r])-objAtomos.getX(
    aux3z1,torS[r]);
x3= objAtomos.getY(aux2z1,torS[r])-objAtomos.getY(
    aux3z1,torS[r]);
z3= objAtomos.getZ(aux2z1,torS[r])-objAtomos.getZ(
    aux3z1,torS[r]);
v4= objAtomos.getX(aux4z1,torS[r])-objAtomos.getX(
    aux3z1,torS[r]);
x4= objAtomos.getY(aux4z1,torS[r])-objAtomos.getY(
    aux3z1,torS[r]);
```

```
z4= objAtomos.getZ(aux4z1,torS[r])-objAtomos.getZ(
    aux3z1,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
double auxdz1 = (prod)/(pmod);
double angdz1= Math.acos(auxdz1);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz1= -angdz1;
}
pangdz1= (getVnU(molS[i][j],molS[i-1][j+1],molS[i
    -2][j+1],molS[i-2][j+2],sem,num,indice)/getPath
    (molS[i][j],molS[i-1][j+1],molS[i-2][j+1],molS[
    i-2][j+2],sem,num,indice)*(1+(Math.cos((getN(
    molS[i][j],molS[i-1][j+1],molS[i-2][j+1],molS[i
    -2][j+2],sem,indice)*angdz1)-getPeriU(molS[i][j
    ],molS[i-1][j+1],molS[i-2][j+1],molS[i-2][j+2],
    sem,indice)))));
//arredondamento TINKER
pangdz1= (Math.round((pangdz1 * 10000.0))) /
    10000.0;
pangdz1= pangdz1+auxz1;
auxz1= pangdz1;
indiceD= indiceD + 1;
ind++;
sem++;
```

```
        while (num==0){
          //transforma pra graus o cos
          pangdz1= (getVnU(molS[i][j],molS[i-1][j+1],molS[
              i-2][j+1],molS[i-2][j+2],sem,num,indice)/
              getPath(molS[i][j],molS[i-1][j+1],molS[i-2][j
              +1],molS[i-2][j+2],sem,num,indice)*(1+(Math.
              cos((getN(molS[i][j],molS[i-1][j+1],molS[i
              -2][j+1],molS[i-2][j+2],sem,indice)*angdz1)-
              getPeriU(molS[i][j],molS[i-1][j+1],molS[i-2][
              j+1],molS[i-2][j+2],sem,indice)))));
          //arredondamento TINKER
          pangdz1= (Math.round((pangdz1 * 10000.0))) /
              10000.0;
          pangdz1= pangdz1+auxz1;
          auxz1= pangdz1;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim if
      if((i>=2)&&(j>=1)&&(molS[i][j]!="") && (molS[i-1][j
          -1]!="") && (molS[i-2][j-1]!="")&& (molS[i-2][j
          ]!="")&&((molS[i][j].substring(11, 12).equals
          ("1"))&&(molS[i-1][j-1].substring(10, 11).equals
          ("1"))&&(molS[i-1][j-1].substring(9, 10).equals
          ("1"))&&(molS[i-2][j-1].substring(9, 10).equals
          ("1"))&&(molS[i-2][j-1].substring(8, 9).equals
          ("1"))&&(molS[i-2][j].substring(8, 9).equals("1")
          ))){
        indice=0;
        double[] aux1u3 = objAtomos.setAtomoU(molS[i][j],
            torS[r]);
        double[] aux2u3 = objAtomos.setAtomoU(molS[i-1][j
            -1],torS[r]);
        double[] aux3u3 = objAtomos.setAtomoU(molS[i-2][j
            -1],torS[r]);
```

```
double[] aux4u3 = objAtomos.setAtomoU(molS[i-2][j
    ],torS[r]);
v1= objAtomos.getX(aux1u3,torS[r])-objAtomos.getX(
    aux2u3,torS[r]);
x1= objAtomos.getY(aux1u3,torS[r])-objAtomos.getY(
    aux2u3,torS[r]);
z1= objAtomos.getZ(aux1u3,torS[r])-objAtomos.getZ(
    aux2u3,torS[r]);
v2= objAtomos.getX(aux3u3,torS[r])-objAtomos.getX(
    aux2u3,torS[r]);
x2= objAtomos.getY(aux3u3,torS[r])-objAtomos.getY(
    aux2u3,torS[r]);
z2= objAtomos.getZ(aux3u3,torS[r])-objAtomos.getZ(
    aux2u3,torS[r]);
v3= objAtomos.getX(aux2u3,torS[r])-objAtomos.getX(
    aux3u3,torS[r]);
x3= objAtomos.getY(aux2u3,torS[r])-objAtomos.getY(
    aux3u3,torS[r]);
z3= objAtomos.getZ(aux2u3,torS[r])-objAtomos.getZ(
    aux3u3,torS[r]);
v4= objAtomos.getX(aux4u3,torS[r])-objAtomos.getX(
    aux3u3,torS[r]);
x4= objAtomos.getY(aux4u3,torS[r])-objAtomos.getY(
    aux3u3,torS[r]);
z4= objAtomos.getZ(aux4u3,torS[r])-objAtomos.getZ(
    aux3u3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
```

```
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdu3 = (prod)/(pmod);
angdu3= Math.acos(auxdu3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu3= -angdu3;
}
pangdu3= (getVnU(molS[i][j],molS[i-1][j-1],molS[i
    -2][j-1],molS[i-2][j],sem,num,indice)/getPath(
    molS[i][j],molS[i-1][j-1],molS[i-2][j-1],molS[i
    -2][j],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i-1][j-1],molS[i-2][j-1],molS[i-2][j
    ],sem,indice)*angdu3)-getPeriU(molS[i][j],molS[
    i-1][j-1],molS[i-2][j-1],molS[i-2][j],sem,
    indice)))));
//arredondamento TINKER
pangdu3= (Math.round((pangdu3 * 10000.0))) /
    10000.0;
pangdu3= pangdu3+auxu3;
auxu3= pangdu3;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdu3= (getVnU(molS[i][j],molS[i-1][j-1],molS[
      i-2][j-1],molS[i-2][j],sem,num,indice)/
      getPath(molS[i][j],molS[i-1][j-1],molS[i-2][j
      -1],molS[i-2][j],sem,num,indice)*(1+(Math.cos
      ((getN(molS[i][j],molS[i-1][j-1],molS[i-2][j
      -1],molS[i-2][j],sem,indice)*angdu3)-getPeriU
      (molS[i][j],molS[i-1][j-1],molS[i-2][j-1],
      molS[i-2][j],sem,indice)))));
  //arredondamento TINKER
```

```
        pangdu3= (Math.round((pangdu3 * 10000.0))) /
            10000.0;
        pangdu3= pangdu3+auxu3;
        auxu3= pangdu3;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }//fim if
    if(((j<qntdcol-2)&&(i>=2)&&(molS[i][j]!="") && (molS
        [i][j+1]!="") && (molS[i-1][j+1]!="")&& (molS[i
        -2][j+2]!=""))&&((molS[i][j].substring(8, 9).
        equals("1"))&&(molS[i][j+1].substring(8, 9).
        equals("1"))&&(molS[i][j+1].substring(9, 10).
        equals("1"))&&(molS[i-1][j+1].substring(9, 10).
        equals("1"))&&(molS[i-1][j+1].substring(10, 11).
        equals("1"))&&(molS[i-2][j+2].substring(11, 12).
        equals("1")))){
      indice=0;
      double[] aux1u4 = objAtomos.setAtomoU(molS[i][j],
          torS[r]);
      double[] aux2u4 = objAtomos.setAtomoU(molS[i][j
          +1],torS[r]);
      double[] aux3u4 = objAtomos.setAtomoU(molS[i-1][j
          +1],torS[r]);
      double[] aux4u4 = objAtomos.setAtomoU(molS[i-2][j
          +2],torS[r]);
      v1= objAtomos.getX(aux1u4,torS[r])-objAtomos.getX(
          aux2u4,torS[r]);
      x1= objAtomos.getY(aux1u4,torS[r])-objAtomos.getY(
          aux2u4,torS[r]);
      z1= objAtomos.getZ(aux1u4,torS[r])-objAtomos.getZ(
          aux2u4,torS[r]);
      v2= objAtomos.getX(aux3u4,torS[r])-objAtomos.getX(
          aux2u4,torS[r]);
```

```
x2= objAtomos.getY(aux3u4,torS[r])-objAtomos.getY(
    aux2u4,torS[r]);
z2= objAtomos.getZ(aux3u4,torS[r])-objAtomos.getZ(
    aux2u4,torS[r]);
v3= objAtomos.getX(aux2u4,torS[r])-objAtomos.getX(
    aux3u4,torS[r]);
x3= objAtomos.getY(aux2u4,torS[r])-objAtomos.getY(
    aux3u4,torS[r]);
z3= objAtomos.getZ(aux2u4,torS[r])-objAtomos.getZ(
    aux3u4,torS[r]);
v4= objAtomos.getX(aux4u4,torS[r])-objAtomos.getX(
    aux3u4,torS[r]);
x4= objAtomos.getY(aux4u4,torS[r])-objAtomos.getY(
    aux3u4,torS[r]);
z4= objAtomos.getZ(aux4u4,torS[r])-objAtomos.getZ(
    aux3u4,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdu4 = (prod)/(pmod);
angdu4= Math.acos(auxdu4);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu4= -angdu4;
}
pangdu4= (getVnU(molS[i][j],molS[i][j+1],molS[i
    -1][j+1],molS[i-2][j+2],sem,num,indice)/getPath
```

```
        (molS[i][j],molS[i][j+1],molS[i-1][j+1],molS[i
        -2][j+2],sem,num,indice)*(1+(Math.cos((getN(
        molS[i][j],molS[i][j+1],molS[i-1][j+1],molS[i
        -2][j+2],sem,indice)*angdu4)-getPeriU(molS[i][j
        ],molS[i][j+1],molS[i-1][j+1],molS[i-2][j+2],
        sem,indice)))));
      //arredondamento TINKER
      pangdu4= (Math.round((pangdu4 * 10000.0))) /
          10000.0;
      pangdu4= pangdu4+auxu4;
      auxu4= pangdu4;
      indiceD= indiceD + 1;
      ind++;
      sem++;
      while (num==0){
        //transforma pra graus o cos
        pangdu4= (getVnU(molS[i][j],molS[i][j+1],molS[i
            -1][j+1],molS[i-2][j+2],sem,num,indice)/
            getPath(molS[i][j],molS[i][j+1],molS[i-1][j
            +1],molS[i-2][j+2],sem,num,indice)*(1+(Math.
            cos((getN(molS[i][j],molS[i][j+1],molS[i-1][j
            +1],molS[i-2][j+2],sem,indice)*angdu4)-
            getPeriU(molS[i][j],molS[i][j+1],molS[i-1][j
            +1],molS[i-2][j+2],sem,indice)))));
        //arredondamento TINKER
        pangdu4= (Math.round((pangdu4 * 10000.0))) /
            10000.0;
        pangdu4= pangdu4+auxu4;
        auxu4= pangdu4;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }//fim if
    if(((j>=1)&&(i<qntdlinha-3)&&(molS[i][j]!="") && (
        molS[i+1][j-1]!="") && (molS[i+2][j-1]!="")&& (
```

```
 molS[i+3][j]!="") )&&((molS[i][j].substring(11,
 12).equals("1"))&&(molS[i+1][j-1].substring(10,
 11).equals("1"))&&(molS[i+1][j-1].substring(9,
 10).equals("1"))&&(molS[i+2][j-1].substring(9,
 10).equals("1"))&&(molS[i+2][j-1].substring(10,
 11).equals("1"))&&(molS[i+3][j].substring(11, 12)
 .equals("1")))){
indice=0;
double[] aux1u8 = objAtomos.setAtomoU(molS[i][j],
    torS[r]);
double[] aux2u8 = objAtomos.setAtomoU(molS[i+1][j
    -1],torS[r]);
double[] aux3u8 = objAtomos.setAtomoU(molS[i+2][j
    -1],torS[r]);
double[] aux4u8 = objAtomos.setAtomoU(molS[i+3][j
    ],torS[r]);
v1= objAtomos.getX(aux1u8,torS[r])-objAtomos.getX(
    aux2u8,torS[r]);
x1= objAtomos.getY(aux1u8,torS[r])-objAtomos.getY(
    aux2u8,torS[r]);
z1= objAtomos.getZ(aux1u8,torS[r])-objAtomos.getZ(
    aux2u8,torS[r]);
v2= objAtomos.getX(aux3u8,torS[r])-objAtomos.getX(
    aux2u8,torS[r]);
x2= objAtomos.getY(aux3u8,torS[r])-objAtomos.getY(
    aux2u8,torS[r]);
z2= objAtomos.getZ(aux3u8,torS[r])-objAtomos.getZ(
    aux2u8,torS[r]);
v3= objAtomos.getX(aux2u8,torS[r])-objAtomos.getX(
    aux3u8,torS[r]);
x3= objAtomos.getY(aux2u8,torS[r])-objAtomos.getY(
    aux3u8,torS[r]);
z3= objAtomos.getZ(aux2u8,torS[r])-objAtomos.getZ(
    aux3u8,torS[r]);
v4= objAtomos.getX(aux4u8,torS[r])-objAtomos.getX(
    aux3u8,torS[r]);
```

```
x4= objAtomos.getY(aux4u8,torS[r])-objAtomos.getY(
    aux3u8,torS[r]);
z4= objAtomos.getZ(aux4u8,torS[r])-objAtomos.getZ(
    aux3u8,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdu8 = (prod)/(pmod);
angdu8= Math.acos(auxdu8);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu8= -angdu8;
}
pangdu8= (getVnU(molS[i][j],molS[i+1][j-1],molS[i
    +2][j-1],molS[i+3][j],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j-1],molS[i+2][j-1],molS[i
    +3][j],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i+1][j-1],molS[i+2][j-1],molS[i+3][j
    ],sem,indice)*angdu8)-getPeriU(molS[i][j],molS[
    i+1][j-1],molS[i+2][j-1],molS[i+3][j],sem,
    indice)))));
//arredondamento TINKER
pangdu8= (Math.round((pangdu8 * 10000.0))) /
    10000.0;
pangdu8= pangdu8+auxu8;
auxu8= pangdu8;
indiceD= indiceD + 1;
```

```
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdu8= (getVnU(molS[i][j],molS[i+1][j-1],molS[
              i+2][j-1],molS[i+3][j],sem,num,indice)/
              getPath(molS[i][j],molS[i+1][j-1],molS[i+2][j
              -1],molS[i+3][j],sem,num,indice)*(1+(Math.cos
              ((getN(molS[i][j],molS[i+1][j-1],molS[i+2][j
              -1],molS[i+3][j],sem,indice)*angdu8)-getPeriU
              (molS[i][j],molS[i+1][j-1],molS[i+2][j-1],
              molS[i+3][j],sem,indice)))));
          //arredondamento TINKER
          pangdu8= (Math.round((pangdu8 * 10000.0))) /
              10000.0;
          pangdu8= pangdu8+auxu8;
          auxu8= pangdu8;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }//fim if
      if(((j<qntdcol-1)&&(i<qntdlinha-3)&&(molS[i][j]!="")
          && (molS[i+1][j+1]!="") && (molS[i+2][j+1]!="")
         && (molS[i+3][j]!=""))&&((molS[i][j].substring
         (10, 11).equals("1"))&&(molS[i+1][j+1].substring
         (11, 12).equals("1"))&&(molS[i+1][j+1].substring
         (9, 10).equals("1"))&&(molS[i+2][j+1].substring
         (9, 10).equals("1"))&&(molS[i+2][j+1].substring
         (11, 12).equals("1"))&&(molS[i+3][j].substring
         (10, 11).equals("1")))){
        indice=0;
        double[] aux1z7 = objAtomos.setAtomoU(molS[i][j],
            torS[r]);
        double[] aux2z7 = objAtomos.setAtomoU(molS[i+1][j
            +1],torS[r]);
```

```
double[] aux3z7 = objAtomos.setAtomoU(molS[i+2][j
    +1],torS[r]);
double[] aux4z7 = objAtomos.setAtomoU(molS[i+3][j
    ],torS[r]);
v1= objAtomos.getX(aux1z7,torS[r])-objAtomos.getX(
    aux2z7,torS[r]);
x1= objAtomos.getY(aux1z7,torS[r])-objAtomos.getY(
    aux2z7,torS[r]);
z1= objAtomos.getZ(aux1z7,torS[r])-objAtomos.getZ(
    aux2z7,torS[r]);
v2= objAtomos.getX(aux3z7,torS[r])-objAtomos.getX(
    aux2z7,torS[r]);
x2= objAtomos.getY(aux3z7,torS[r])-objAtomos.getY(
    aux2z7,torS[r]);
z2= objAtomos.getZ(aux3z7,torS[r])-objAtomos.getZ(
    aux2z7,torS[r]);
v3= objAtomos.getX(aux2z7,torS[r])-objAtomos.getX(
    aux3z7,torS[r]);
x3= objAtomos.getY(aux2z7,torS[r])-objAtomos.getY(
    aux3z7,torS[r]);
z3= objAtomos.getZ(aux2z7,torS[r])-objAtomos.getZ(
    aux3z7,torS[r]);
v4= objAtomos.getX(aux4z7,torS[r])-objAtomos.getX(
    aux3z7,torS[r]);
x4= objAtomos.getY(aux4z7,torS[r])-objAtomos.getY(
    aux3z7,torS[r]);
z4= objAtomos.getZ(aux4z7,torS[r])-objAtomos.getZ(
    aux3z7,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
```

```
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+Math.
    pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+Math
    .pow(c2,2));
pmod= mod*mod2;
auxdz7 = (prod)/(pmod);
angdz7= Math.acos(auxdz7);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz7= -angdz7;
}
pangdz7= (getVnU(molS[i][j],molS[i+1][j+1],molS[i
    +2][j+1],molS[i+3][j],sem,num,indice)/getPath(
    molS[i][j],molS[i+1][j+1],molS[i+2][j+1],molS[i
    +3][j],sem,num,indice)*(1+(Math.cos((getN(molS[
    i][j],molS[i+1][j+1],molS[i+2][j+1],molS[i+3][j
    ],sem,indice)*angdz7)-getPeriU(molS[i][j],molS[
    i+1][j+1],molS[i+2][j+1],molS[i+3][j],sem,
    indice)))));
//arredondamento TINKER
pangdz7= (Math.round((pangdz7 * 10000.0))) /
    10000.0;
pangdz7= pangdz7+auxz7;
auxz7= pangdz7;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdz7= (getVnU(molS[i][j],molS[i+1][j+1],molS[
      i+2][j+1],molS[i+3][j],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j+1],molS[i+2][j
      +1],molS[i+3][j],sem,num,indice)*(1+(Math.cos
      ((getN(molS[i][j],molS[i+1][j+1],molS[i+2][j
      +1],molS[i+3][j],sem,indice)*angdz7)-getPeriU
      (molS[i][j],molS[i+1][j+1],molS[i+2][j+1],
```

```
            molS[i+3][j],sem,indice))))));
        //arredondamento TINKER
        pangdz7= (Math.round((pangdz7 * 10000.0))) /
            10000.0;
        pangdz7= pangdz7+auxz7;
        auxz7= pangdz7;
        indiceD= indiceD + 1;
        ind++;
        sem++;
      }
    }
    if(((i<qntdlinha-3)&&(j>=1)&&(molS[i][j]!="") && (
        molS[i+1][j]!="") && (molS[i+2][j-1]!="")&& (molS
        [i+3][j-1]!="")){
      if ((molS[i][j].substring(9, 10).equals("1"))&&(
          molS[i+1][j].substring(9, 10).equals("1")) && (
          molS[i+1][j].substring(11, 12).equals("1"))&&(
          molS[i+2][j-1].substring(10, 11).equals("1"))&&
        (molS[i+2][j-1].substring(9, 10).equals("1"))&&(
            molS[i+3][j-1].substring(9, 10).equals("1")))
            {
        indice=0;
        double[] aux1z2 = objAtomos.setAtomoU(molS[i][j
            ],torS[r]);
        double[] aux2z2= objAtomos.setAtomoU(molS[i+1][j
            ],torS[r]);
        double[] aux3z2= objAtomos.setAtomoU(molS[i+2][j
            -1],torS[r]);
        double[] aux4z2= objAtomos.setAtomoU(molS[i+3][j
            -1],torS[r]);
        v1= objAtomos.getX(aux1z2,torS[r])-objAtomos.
            getX(aux2z2,torS[r]);
        x1= objAtomos.getY(aux1z2,torS[r])-objAtomos.
            getY(aux2z2,torS[r]);
        z1= objAtomos.getZ(aux1z2,torS[r])-objAtomos.
            getZ(aux2z2,torS[r]);
```

```
v2= objAtomos.getX(aux3z2,torS[r])-objAtomos.
    getX(aux2z2,torS[r]);
x2= objAtomos.getY(aux3z2,torS[r])-objAtomos.
    getY(aux2z2,torS[r]);
z2= objAtomos.getZ(aux3z2,torS[r])-objAtomos.
    getZ(aux2z2,torS[r]);
v3= objAtomos.getX(aux2z2,torS[r])-objAtomos.
    getX(aux3z2,torS[r]);
x3= objAtomos.getY(aux2z2,torS[r])-objAtomos.
    getY(aux3z2,torS[r]);
z3= objAtomos.getZ(aux2z2,torS[r])-objAtomos.
    getZ(aux3z2,torS[r]);
v4= objAtomos.getX(aux4z2,torS[r])-objAtomos.
    getX(aux3z2,torS[r]);
x4= objAtomos.getY(aux4z2,torS[r])-objAtomos.
    getY(aux3z2,torS[r]);
z4= objAtomos.getZ(aux4z2,torS[r])-objAtomos.
    getZ(aux3z2,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz2= (prod)/(pmod);
angdz2= Math.acos(auxdz2);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz2= -angdz2;
}
```

```
        pangdz2= (getVnU(molS[i][j],molS[i+1][j],molS[i
            +2][j-1],molS[i+3][j-1],sem,num,indice)/
            getPath(molS[i][j],molS[i+1][j],molS[i+2][j
            -1],molS[i+3][j-1],sem,num,indice)*(1+(Math.
            cos((getN(molS[i][j],molS[i+1][j],molS[i+2][j
            -1],molS[i+3][j-1],sem,indice)*angdz2)-
            getPeriU(molS[i][j],molS[i+1][j],molS[i+2][j
            -1],molS[i+3][j-1],sem,indice)))));
        //arredondamento TINKER
        pangdz2= (Math.round((pangdz2 * 10000.0))) /
            10000.0;
        pangdz2= pangdz2+auxz2;
        auxz2= pangdz2;
        indiceD= indiceD + 1;
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdz2= (getVnU(molS[i][j],molS[i+1][j],molS[
              i+2][j-1],molS[i+3][j-1],sem,num,indice)/
              getPath(molS[i][j],molS[i+1][j],molS[i+2][j
              -1],molS[i+3][j-1],sem,num,indice)*(1+(Math
              .cos((getN(molS[i][j],molS[i+1][j],molS[i
              +2][j-1],molS[i+3][j-1],sem,indice)*angdz2)
              -getPeriU(molS[i][j],molS[i+1][j],molS[i
              +2][j-1],molS[i+3][j-1],sem,indice)))));
          //arredondamento TINKER
          pangdz2= (Math.round((pangdz2 * 10000.0))) /
              10000.0;
          pangdz2= pangdz2+auxz2;
          auxz2= pangdz2;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }
```

```
}//fim if
if(((i>=1)&&(j<qntdcol-3)&&(molS[i][j]!="") && (molS
    [i-1][j+1]!="") && (molS[i][j+2]!="")&& (molS[i
    -1][j+3]!="")){
  if ((molS[i][j].substring(10, 11).equals("1"))&&(
     molS[i-1][j+1].substring(11, 12).equals("1"))
     &&(molS[i-1][j+1].substring(10, 11).equals("1")
     )&&(molS[i][j+2].substring(11, 12).equals("1"))
     &&(molS[i][j+2].substring(10, 11).equals("1"))
     &&(molS[i-1][j+3].substring(11, 12).equals("1")
     )){
     indice=0;
     double[] aux1z3 = objAtomos.setAtomoU(molS[i][j
        ],torS[r]);
     double[] aux2z3 = objAtomos.setAtomoU(molS[i-1][
        j+1],torS[r]);
     double[] aux3z3 = objAtomos.setAtomoU(molS[i][j
        +2],torS[r]);
     double[] aux4z3 = objAtomos.setAtomoU(molS[i-1][
        j+3],torS[r]);
     v1= objAtomos.getX(aux1z3,torS[r])-objAtomos.
        getX(aux2z3,torS[r]);
     x1= objAtomos.getY(aux1z3,torS[r])-objAtomos.
        getY(aux2z3,torS[r]);
     z1= objAtomos.getZ(aux1z3,torS[r])-objAtomos.
        getZ(aux2z3,torS[r]);
     v2= objAtomos.getX(aux3z3,torS[r])-objAtomos.
        getX(aux2z3,torS[r]);
     x2= objAtomos.getY(aux3z3,torS[r])-objAtomos.
        getY(aux2z3,torS[r]);
     z2= objAtomos.getZ(aux3z3,torS[r])-objAtomos.
        getZ(aux2z3,torS[r]);
     v3= objAtomos.getX(aux2z3,torS[r])-objAtomos.
        getX(aux3z3,torS[r]);
     x3= objAtomos.getY(aux2z3,torS[r])-objAtomos.
        getY(aux3z3,torS[r]);
```

```
z3= objAtomos.getZ(aux2z3,torS[r])-objAtomos.
    getZ(aux3z3,torS[r]);
v4= objAtomos.getX(aux4z3,torS[r])-objAtomos.
    getX(aux3z3,torS[r]);
x4= objAtomos.getY(aux4z3,torS[r])-objAtomos.
    getY(aux3z3,torS[r]);
z4= objAtomos.getZ(aux4z3,torS[r])-objAtomos.
    getZ(aux3z3,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz3 = (prod)/(pmod);
angdz3 = Math.acos(auxdz3);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz3= -angdz3;
}
pangdz3= (getVnU(molS[i][j],molS[i-1][j+1],molS[
    i][j+2],molS[i-1][j+3],sem,num,indice)/
    getPath(molS[i][j],molS[i-1][j+1],molS[i][j
    +2],molS[i-1][j+3],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i-1][j+1],molS[i][j
    +2],molS[i-1][j+3],sem,indice)*angdz3)-
    getPeriU(molS[i][j],molS[i-1][j+1],molS[i][j
    +2],molS[i-1][j+3],sem,indice)))));
//arredondamento TINKER
```

```
            pangdz3= (Math.round((pangdz3 * 10000.0))) /
                10000.0;
            pangdz3= pangdz3+auxz3;
            auxz3= pangdz3;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdz3= (getVnU(molS[i][j],molS[i-1][j+1],
                  molS[i][j+2],molS[i-1][j+3],sem,num,indice)
                  /getPath(molS[i][j],molS[i-1][j+1],molS[i][
                  j+2],molS[i-1][j+3],sem,num,indice)*(1+(
                  Math.cos((getN(molS[i][j],molS[i-1][j+1],
                  molS[i][j+2],molS[i-1][j+3],sem,indice)*
                  angdz3)-getPeriU(molS[i][j],molS[i-1][j+1],
                  molS[i][j+2],molS[i-1][j+3],sem,indice)))))
                  ;
              //arredondamento TINKER
              pangdz3= (Math.round((pangdz3 * 10000.0))) /
                  10000.0;
              pangdz3= pangdz3+auxz3;
              auxz3= pangdz3;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }
        }//fim if
        if(((i<qntdlinha-2)&&(j<qntdcol-1)&&(j>=1)&&(molS[i
            ][j]!="") && (molS[i+1][j-1]!="") && (molS[i+2][j
            ]!="")&& (molS[i+2][j+1]!=""))&&((molS[i][j].
            substring(11, 12).equals("1"))&&(molS[i+1][j-1].
            substring(10, 11).equals("1"))&&(molS[i+1][j-1].
            substring(10, 11).equals("1"))&&(molS[i+2][j].
            substring(11, 12).equals("1"))&&(molS[i+2][j].
```

```
 substring(8, 9).equals("1"))&&(molS[i+2][j+1].
 substring(8, 9).equals("1")))){
if ((molS[i][j].substring(0, 7).equals("NCA0002"))
   &&(molS[i+1][j-1].substring(0, 7).equals("
   OSR1001"))){

}else{
  indice=0;
  double[] aux1z4 = objAtomos.setAtomoU(molS[i][j
     ],torS[r]);
  double[] aux2z4 = objAtomos.setAtomoU(molS[i+1][
     j-1],torS[r]);
  double[] aux3z4 = objAtomos.setAtomoU(molS[i+2][
     j],torS[r]);
  double[] aux4z4 = objAtomos.setAtomoU(molS[i+2][
     j+1],torS[r]);
  v1= objAtomos.getX(aux1z4,torS[r])-objAtomos.
     getX(aux2z4,torS[r]);
  x1= objAtomos.getY(aux1z4,torS[r])-objAtomos.
     getY(aux2z4,torS[r]);
  z1= objAtomos.getZ(aux1z4,torS[r])-objAtomos.
     getZ(aux2z4,torS[r]);
  v2= objAtomos.getX(aux3z4,torS[r])-objAtomos.
     getX(aux2z4,torS[r]);
  x2= objAtomos.getY(aux3z4,torS[r])-objAtomos.
     getY(aux2z4,torS[r]);
  z2= objAtomos.getZ(aux3z4,torS[r])-objAtomos.
     getZ(aux2z4,torS[r]);
  v3= objAtomos.getX(aux2z4,torS[r])-objAtomos.
     getX(aux3z4,torS[r]);
  x3= objAtomos.getY(aux2z4,torS[r])-objAtomos.
     getY(aux3z4,torS[r]);
  z3= objAtomos.getZ(aux2z4,torS[r])-objAtomos.
     getZ(aux3z4,torS[r]);
  v4= objAtomos.getX(aux4z4,torS[r])-objAtomos.
     getX(aux3z4,torS[r]);
```

```
x4= objAtomos.getY(aux4z4,torS[r])-objAtomos.
    getY(aux3z4,torS[r]);
z4= objAtomos.getZ(aux4z4,torS[r])-objAtomos.
    getZ(aux3z4,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz4 = (prod)/(pmod);
angdz4 = Math.acos(auxdz4);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz4= -angdz4;
}
pangdz4= (getVnU(molS[i][j],molS[i+1][j-1],molS[
    i+2][j],molS[i+2][j+1],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j-1],molS[i+2][j
    ],molS[i+2][j+1],sem,num,indice)*(1+(Math.cos
    ((getN(molS[i][j],molS[i+1][j-1],molS[i+2][j
    ],molS[i+2][j+1],sem,indice)*angdz4)-getPeriU
    (molS[i][j],molS[i+1][j-1],molS[i+2][j],molS[
    i+2][j+1],sem,indice)))));
//arredondamento TINKER
pangdz4= (Math.round((pangdz4 * 10000.0))) /
    10000.0;
pangdz4= pangdz4+auxz4;
auxz4= pangdz4;
indiceD= indiceD + 1;
```

```
        ind++;
        sem++;
        while (num==0){
          //transforma pra graus o cos
          pangdz4= (getVnU(molS[i][j],molS[i+1][j-1],
              molS[i+2][j],molS[i+2][j+1],sem,num,indice)
              /getPath(molS[i][j],molS[i+1][j-1],molS[i
              +2][j],molS[i+2][j+1],sem,num,indice)*(1+(
              Math.cos((getN(molS[i][j],molS[i+1][j-1],
              molS[i+2][j],molS[i+2][j+1],sem,indice)*
              angdz4)-getPeriU(molS[i][j],molS[i+1][j-1],
              molS[i+2][j],molS[i+2][j+1],sem,indice)))))
              ;
          //arredondamento TINKER
          pangdz4= (Math.round((pangdz4 * 10000.0))) /
              10000.0;
          pangdz4= pangdz4+auxz4;
          auxz4= pangdz4;
          indiceD= indiceD + 1;
          ind++;
          sem++;
        }
      }
    }//fim if
    if((i<qntdlinha-2)&&(j>=2)&&(molS[i][j]!="") && (
        molS[i+1][j]!="") && (molS[i+2][j-1]!="")&& (molS
        [i+1][j-2]!="")&&((molS[i][j].substring(9, 10).
        equals("1"))&&(molS[i+1][j].substring(9, 10).
        equals("1"))&&(molS[i+1][j].substring(11, 12).
        equals("1"))&&(molS[i+2][j-1].substring(10, 11).
        equals("1"))&&(molS[i+2][j-1].substring(11, 12).
        equals("1"))&&(molS[i+1][j-2].substring(10, 11).
        equals("1")))){
        indice=0;
        double[] aux1z5 = objAtomos.setAtomoU(molS[i][j
            ],torS[r]);
```

```
double[] aux2z5 = objAtomos.setAtomoU(molS[i+1][
    j], torS[r]);
double[] aux3z5 = objAtomos.setAtomoU(molS[i+2][
    j-1], torS[r]);
double[] aux4z5 = objAtomos.setAtomoU(molS[i+1][
    j-2], torS[r]);
v1= objAtomos.getX(aux1z5, torS[r])-objAtomos.
    getX(aux2z5, torS[r]);
x1= objAtomos.getY(aux1z5, torS[r])-objAtomos.
    getY(aux2z5, torS[r]);
z1= objAtomos.getZ(aux1z5, torS[r])-objAtomos.
    getZ(aux2z5, torS[r]);
v2= objAtomos.getX(aux3z5, torS[r])-objAtomos.
    getX(aux2z5, torS[r]);
x2= objAtomos.getY(aux3z5, torS[r])-objAtomos.
    getY(aux2z5, torS[r]);
z2= objAtomos.getZ(aux3z5, torS[r])-objAtomos.
    getZ(aux2z5, torS[r]);
v3= objAtomos.getX(aux2z5, torS[r])-objAtomos.
    getX(aux3z5, torS[r]);
x3= objAtomos.getY(aux2z5, torS[r])-objAtomos.
    getY(aux3z5, torS[r]);
z3= objAtomos.getZ(aux2z5, torS[r])-objAtomos.
    getZ(aux3z5, torS[r]);
v4= objAtomos.getX(aux4z5, torS[r])-objAtomos.
    getX(aux3z5, torS[r]);
x4= objAtomos.getY(aux4z5, torS[r])-objAtomos.
    getY(aux3z5, torS[r]);
z4= objAtomos.getZ(aux4z5, torS[r])-objAtomos.
    getZ(aux3z5, torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
```

```
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdz5 = (prod)/(pmod);
angdz5 = Math.acos(auxdz5);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdz5= -angdz5;
}
pangdz5= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j-1],molS[i+1][j-2],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j],molS[i+2][j
    -1],molS[i+1][j-2],sem,num,indice)*(1+(Math.
    cos((getN(molS[i][j],molS[i+1][j],molS[i+2][j
    -1],molS[i+1][j-2],sem,indice)*angdz5)-
    getPeriU(molS[i][j],molS[i+1][j],molS[i+2][j
    -1],molS[i+1][j-2],sem,indice)))));
//arredondamento TINKER
pangdz5= (Math.round((pangdz5 * 10000.0))) /
    10000.0;
pangdz5= pangdz5+auxz5;
auxz5= pangdz5;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdz5= (getVnU(molS[i][j],molS[i+1][j],molS[
      i+2][j-1],molS[i+1][j-2],sem,num,indice)/
      getPath(molS[i][j],molS[i+1][j],molS[i+2][j
      -1],molS[i+1][j-2],sem,num,indice)*(1+(Math
      .cos((getN(molS[i][j],molS[i+1][j],molS[i
      +2][j-1],molS[i+1][j-2],sem,indice)*angdz5)
```

```
                -getPeriU(molS[i][j],molS[i+1][j],molS[i
                +2][j-1],molS[i+1][j-2],sem,indice)))));
            //arredondamento TINKER
            pangdz5= (Math.round((pangdz5 * 10000.0))) /
                10000.0;
            pangdz5= pangdz5+auxz5;
            auxz5= pangdz5;
            indiceD= indiceD + 1;
            ind++;
            sem++;
        }
    }//fim if
    if((molS[i][j]!="") &&(i<qntdlinha-3)&&(j<qntdcol-2)
       && (molS[i+1][j+1]!="") && (molS[i+2][j+1]!="")
       && (molS[i+3][j+2]!="")){
      //verificar se todo esta ligados
      if ((molS[i][j].substring(10, 11).equals("1"))&&(
          molS[i+1][j+1].substring(11, 12).equals("1"))
          &&(molS[i+1][j+1].substring(9, 10).equals("1"))
          &&(molS[i+2][j+1].substring(9, 10).equals("1"))
          &&(molS[i+2][j+1].substring(10, 11).equals("1")
          )&&(molS[i+3][j+2].substring(11, 12).equals
          ("1"))){
          //nao posso torcer o anel de ligacao dupla
          indice=0;
          double[] aux1u5 = objAtomos.setAtomoU(molS[i][j
              ],torS[r]);
          double[] aux2u5 = objAtomos.setAtomoU(molS[i+1][
              j+1],torS[r]);
          double[] aux3u5 = objAtomos.setAtomoU(molS[i+2][
              j+1],torS[r]);
          double[] aux4u5 = objAtomos.setAtomoU(molS[i+3][
              j+2],torS[r]);
          v1= objAtomos.getX(aux1u5,torS[r])-objAtomos.
              getX(aux2u5,torS[r]);
```

```
x1= objAtomos.getY(aux1u5,torS[r])-objAtomos.
    getY(aux2u5,torS[r]);
z1= objAtomos.getZ(aux1u5,torS[r])-objAtomos.
    getZ(aux2u5,torS[r]);
v2= objAtomos.getX(aux3u5,torS[r])-objAtomos.
    getX(aux2u5,torS[r]);
x2= objAtomos.getY(aux3u5,torS[r])-objAtomos.
    getY(aux2u5,torS[r]);
z2= objAtomos.getZ(aux3u5,torS[r])-objAtomos.
    getZ(aux2u5,torS[r]);
v3= objAtomos.getX(aux2u5,torS[r])-objAtomos.
    getX(aux3u5,torS[r]);
x3= objAtomos.getY(aux2u5,torS[r])-objAtomos.
    getY(aux3u5,torS[r]);
z3= objAtomos.getZ(aux2u5,torS[r])-objAtomos.
    getZ(aux3u5,torS[r]);
v4= objAtomos.getX(aux4u5,torS[r])-objAtomos.
    getX(aux3u5,torS[r]);
x4= objAtomos.getY(aux4u5,torS[r])-objAtomos.
    getY(aux3u5,torS[r]);
z4= objAtomos.getZ(aux4u5,torS[r])-objAtomos.
    getZ(aux3u5,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu5 = (prod)/(pmod);
angdu5 = Math.acos(auxdu5);
```

```
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu5= -angdu5;
}
pangdu5= ((getVnU(molS[i][j],molS[i+1][j+1],molS
    [i+2][j+1],molS[i+3][j+2],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j+1],molS[i+2][j
    +1],molS[i+3][j+2],sem,num,indice))*(1+(Math.
    cos((getN(molS[i][j],molS[i+1][j+1],molS[i
    +2][j+1],molS[i+3][j+2],sem,indice)*angdu5)-
    getPeriU(molS[i][j],molS[i+1][j+1],molS[i+2][
    j+1],molS[i+3][j+2],sem,indice)))));
//arredondamento TINKER
pangdu5= (Math.round((pangdu5 * 10000.0))) /
    10000.0;
pangdu5= pangdu5+auxu5;
auxu5= pangdu5;
indiceD= indiceD + 1;
ind++;
sem++;
while (num==0){
  //transforma pra graus o cos
  pangdu5= ((getVnU(molS[i][j],molS[i+1][j+1],
      molS[i+2][j+1],molS[i+3][j+2],sem,num,
      indice)/getPath(molS[i][j],molS[i+1][j+1],
      molS[i+2][j+1],molS[i+3][j+2],sem,num,
      indice))*(1+(Math.cos((getN(molS[i][j],molS
      [i+1][j+1],molS[i+2][j+1],molS[i+3][j+2],
      sem,indice)*angdu5)-getPeriU(molS[i][j],
      molS[i+1][j+1],molS[i+2][j+1],molS[i+3][j
      +2],sem,indice)))));
  //arredondamento TINKER
  pangdu5= (Math.round((pangdu5 * 10000.0))) /
      10000.0;
  pangdu5= pangdu5+auxu5;
  auxu5= pangdu5;
```

```
            indiceD= indiceD + 1;
            ind++;
            sem++;
          }
        }
      }//fim if
      if(((i<qntdlinha-2)&&(j<qntdcol-1)&&(molS[i][j]!="")
          && (molS[i+1][j]!="") && (molS[i+2][j+1]!="") &&
          (molS[i+1][j+1]!="") )){
        if ((((molS[i][j].substring(9, 10).equals("1")))&&(
            molS[i+1][j].substring(9, 10).equals("1"))))&&(
            molS[i+1][j].substring(10, 11).equals("1"))&&(
            molS[i+2][j+1].substring(11, 12).equals("1"))
            &&(molS[i+2][j+1].substring(9, 10).equals("1"))
            &(molS[i+1][j+1].substring(9, 10).equals("1")))
            {
          indice=0;
          //isolar a b e c dos planos
          aux1u7= objAtomos.setAtomoU(molS[i][j],torS[r]);
          aux2u7= objAtomos.setAtomoU(molS[i+1][j],torS[r
              ]);
          aux3u7= objAtomos.setAtomoU(molS[i+2][j+1],torS[
              r]);
          aux4u7= objAtomos.setAtomoU(molS[i+3][j+1],torS[
              r]);
          v1= objAtomos.getX(aux1u7,torS[r])-objAtomos.
              getX(aux2u7,torS[r]);
          x1= objAtomos.getY(aux1u7,torS[r])-objAtomos.
              getY(aux2u7,torS[r]);
          z1= objAtomos.getZ(aux1u7,torS[r])-objAtomos.
              getZ(aux2u7,torS[r]);
          v2= objAtomos.getX(aux3u7,torS[r])-objAtomos.
              getX(aux2u7,torS[r]);
          x2= objAtomos.getY(aux3u7,torS[r])-objAtomos.
              getY(aux2u7,torS[r]);
```

```
z2= objAtomos.getZ(aux3u7,torS[r])-objAtomos.
    getZ(aux2u7,torS[r]);
v3= objAtomos.getX(aux2u7,torS[r])-objAtomos.
    getX(aux3u7,torS[r]);
x3= objAtomos.getY(aux2u7,torS[r])-objAtomos.
    getY(aux3u7,torS[r]);
z3= objAtomos.getZ(aux2u7,torS[r])-objAtomos.
    getZ(aux3u7,torS[r]);
v4= objAtomos.getX(aux4u7,torS[r])-objAtomos.
    getX(aux3u7,torS[r]);
x4= objAtomos.getY(aux4u7,torS[r])-objAtomos.
    getY(aux3u7,torS[r]);
z4= objAtomos.getZ(aux4u7,torS[r])-objAtomos.
    getZ(aux3u7,torS[r]);
a1= ((x1*z2)-(x2*z1));
b1= ((z1*v2)-(v1*z2));
c1= ((v1*x2)-(x1*v2));
a2= ((x3*z4)-(x4*z3));
b2= ((z3*v4)-(v3*z4));
c2= ((v3*x4)-(x3*v4));
prod =(a1*a2)+(b1*b2)+(c1*c2);
mod= Math.sqrt(Math.pow(a1,2)+Math.pow(b1,2)+
    Math.pow(c1,2));
mod2= Math.sqrt(Math.pow(a2,2)+Math.pow(b2,2)+
    Math.pow(c2,2));
pmod= mod*mod2;
auxdu7= (prod)/(pmod);
angdu7= Math.acos(auxdu7);
double sinal= (v1*a2)+(x1*b2)+(z1*c2);
if (sinal<=0.0){
  angdu7= -angdu7;
}
pangdu7= (getVnU(molS[i][j],molS[i+1][j],molS[i
    +2][j+1],molS[i+1][j+1],sem,num,indice)/
    getPath(molS[i][j],molS[i+1][j],molS[i+2][j
    +1],molS[i+1][j+1],sem,num,indice)*(1+(Math.
```

```
                cos((getN(molS[i][j],molS[i+1][j],molS[i+2][j
                +1],molS[i+1][j+1],sem,indice)*angdu7)-
                getPeriU(molS[i][j],molS[i+1][j],molS[i+2][j
                +1],molS[i+1][j+1],sem,indice)))));
            //arredondamento TINKER
            pangdu7= (Math.round((pangdu7 * 10000.0))) /
                10000.0;
            pangdu7= pangdu7+auxu7;
            auxu7= pangdu7;
            indiceD= indiceD + 1;
            ind++;
            sem++;
            while (num==0){
              //transforma pra graus o cos
              pangdu7= (getVnU(molS[i][j],molS[i+1][j],molS[
                  i+2][j+1],molS[i+1][j+1],sem,num,indice)/
                  getPath(molS[i][j],molS[i+1][j],molS[i+2][j
                  +1],molS[i+1][j+1],sem,num,indice)*(1+(Math
                  .cos((getN(molS[i][j],molS[i+1][j],molS[i
                  +2][j+1],molS[i+1][j+1],sem,indice)*angdu7)
                  -getPeriU(molS[i][j],molS[i+1][j],molS[i
                  +2][j+1],molS[i+1][j+1],sem,indice)))));
              //arredondamento TINKER
              pangdu7= (Math.round((pangdu7 * 10000.0))) /
                  10000.0;
              pangdu7= pangdu7+auxu7;
              auxu7= pangdu7;
              indiceD= indiceD + 1;
              ind++;
              sem++;
            }
          }//fim if
        }//fim if
      }//fim else
    }//fim for
  }//fim for
```

```
    pangd[r]= pangdz+pangdz1+pangdz2+pangdu+pangdu2+pangdu3+
        pangdu4+pangdu5+pangdu6+pangdu7+pangdu8+pangdz3+pangdz4
        +pangdz5+
    pangdz6+pangdz7+pangdz8+pangdb4+pangdbb4+pangdb5+pangdbb5+
        pangdb6;
    }//fim for r
    return pangd;
  }
  //faz busca dos parametros
  public double calculaFit(int semente) throws JXLException,
     IOException{
    //posso unir o lh, lelet e lj para otimizar, pois os lacos
        sao iguais
    double[] diedro, diedro3, diedroAnel, diedro2= new double[
        torS.length];
    double erro2= 0.0;
    double aux= 0.0;
    diedro= new double[torS.length];
    diedroAnel= new double[torS.length];
    diedro3= new double[torS.length];
    String log="";

    diedro= angdU(semente);
    diedro2= angd2U(semente);
    diedro3= angd3U(semente);
    diedroAnel= angdAnelU(semente);
    for (int i=1; i<torS.length; i++){
    //Adenina
      if (i==1){
        aux1[i]= (-0.02304317-((diedro[i]+diedro2[i]+diedro3[i
            ]+diedroAnel[i
            ]+4.2237+4.8935+0.0021+6.8913+(-72.0658))-
              (diedro[0]+diedro2[0]+diedro3[0]+diedroAnel
                  [0]+4.1201+4.4607+0.0280+6.6095+(-71.1718))));
        aux1[i]= Math.pow(aux1[i], 2);
      }else if (i==2){
```

```
    aux1[i]= (0.00597640-((diedro[i]+diedro2[i]+diedro3[i
        ]+diedroAnel[i
        ]+4.1665+4.7558+0.0007+6.7875+(-71.4734))-
          (diedro[0]+diedro2[0]+diedro3[0]+diedroAnel
              [0]+4.1201+4.4607+0.0280+6.6095+(-71.1718))));
    aux1[i]= Math.pow(aux1[i], 2);
  }else if (i==3){
    aux1[i]= (0.01719609-((diedro[i]+diedro2[i]+diedro3[i
        ]+diedroAnel[i
        ]+4.1871+4.7706+0.0560+6.6880+(-71.3213))-
          (diedro[0]+diedro2[0]+diedro3[0]+diedroAnel
              [0]+4.1201+4.4607+0.0280+6.6095+(-71.1718))));
    aux1[i]= Math.pow(aux1[i], 2);
  }else if (i==4){
    aux1[i]= (5.12209257-((diedro[i]+diedro2[i]+diedro3[i
        ]+diedroAnel[i
        ]+3.3633+4.6939+0.2138+4.7869+(-62.0200))-
          (diedro[0]+diedro2[0]+diedro3[0]+diedroAnel
              [0]+4.1201+4.4607+0.0280+6.6095+(-71.1718))));
    aux1[i]= Math.pow(aux1[i], 2);
  }else if (i==5){
    aux1[i]= (5.23069997-((diedro[i]+diedro2[i]+diedro3[i
        ]+diedroAnel[i
        ]+3.3331+4.4795+0.1951+4.9644+(-62.3117))-
          (diedro[0]+diedro2[0]+diedro3[0]+diedroAnel
              [0]+4.1201+4.4607+0.0280+6.6095+(-71.1718))));
    aux1[i]= Math.pow(aux1[i], 2);
  }else if (i==6){
    aux1[i]= (0.03939564-((diedro[i]+diedro2[i]+diedro3[i
        ]+diedroAnel[i
        ]+4.0917+4.3828+0.0338+6.4224+(-70.8748))-
          (diedro[0]+diedro2[0]+diedro3[0]+diedroAnel
              [0]+4.1201+4.4607+0.0280+6.6095+(-71.1718))));
    aux1[i]= Math.pow(aux1[i], 2);
  }else if (i==7){
```

```
        aux1[i]= (-0.01860182-((diedro[i]+diedro2[i]+diedro3[i
            ]+diedroAnel[i
            ]+4.2144+4.6511+0.0068+6.8488+(-71.9076))-
              (diedro[0]+diedro2[0]+diedro3[0]+diedroAnel
                  [0]+4.1201+4.4607+0.0280+6.6095+(-71.1718))));
        aux1[i]= Math.pow(aux1[i], 2);
      }
      aux= aux1[i]+aux;
    }
    fit= Math.sqrt(aux/(aux1.length-1));
    return fit;

  }//fim calculaFitness
  public double getFitD(){
    String fitness = "" + fit;
    return fit;

  }//fim getFit
  public double getFit(){
    String fitness = "" + fit;
    return fit;

  }//fim getFit

  public double getVnU(String str, String str2, String str3,
     String str4, int semente, int n, int ind){
    double vn= 0.0;
    String auxS,auxS2,auxS3,auxS4 = "";

    for (int i= ind; i< vnS.length; i++){
      if(vnS[i]!=""){
      auxS= vnS[i].substring(0,2);
      auxS2= vnS[i].substring(3,5);
      auxS3= vnS[i].substring(6,8);
      auxS4= vnS[i].substring(9,11);
```

```
        if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
            str2.substring(0,2)))&&(auxS3.equals(str3.substring
            (0,2)))&&(auxS4.equals(str4.substring(0,2))))||((auxS
            .equals(str.substring(0,2)))&&(auxS2.equals(str3.
            substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))
            &&(auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
   substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))){
          if((str.substring(0, 2).equals("OS"))&&(str2.substring
              (0, 2).equals("CT"))&&(str3.substring(0, 2).equals
              ("N9"))&&(str4.substring(0, 2).equals("C1"))){
            indMutacao2= i;
            if (num==1){
              vn= vnDInd[0];
            }else if(rep==1){
              vn= vnDInd[1];
            }else if(rep==2){
              vn= vnDInd[2];
            }else if(rep==3){
              vn= vnDInd[3];
            }
            cruzaVn[1]=i;
          }else if((str.substring(0, 2).equals("OS"))&&(str2.
              substring(0, 2).equals("CT"))&&(str3.substring(0,
              2).equals("N9"))&&(str4.substring(0, 2).equals("C2
              ")))){
            indMutacao3= i;
            if (num==1){
              vn= vnDInd[0];
            }else if(rep==1){
              vn= vnDInd[1];
            }else if(rep==2){
```

```
            vn= vnDInd[2];
          }else if(rep==3){
            vn= vnDInd[3];
          }
          cruzaVn[2]=i;
        }else if((str.substring(0, 2).equals("OS"))&&(str2.
            substring(0, 2).equals("PP"))&&(str3.substring(0,
            2).equals("OS"))&&(str4.substring(0, 2).equals("CT
            "))){
          indMutacao4= i;
          vn= vnD[i];
          cruzaVn[3]=i;
        }else{
          vn= vnD[i];
        }
        //checar pra ver se nao tem outra forma igual. COmo no
            meu caso soh tem 4 casos no maxio, foram feitos 4
            ifs
        if ((i+1)<vnS.length-1){
          auxS= vnS[i+1].substring(0,2);
          auxS2= vnS[i+1].substring(3,5);
          auxS3= vnS[i+1].substring(6,8);
          auxS4= vnS[i+1].substring(9,11);

if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str2.
    substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
    auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
    substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
    auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
    substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
    auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
    substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
    auxS4.equals(str.substring(0,2))))){
```

```
        num = 0;
        indice= i+1;
        // o AG soh vai trabalhar nesse angulo diedro
        if((auxS.substring(0, 2).equals("OS"))&&(auxS2.
            substring(0, 2).equals("CT"))&&(auxS3.substring
            (0, 2).equals("N9"))&&(auxS4.substring(0, 2).
            equals("C2"))){
          if (rep==0){
            indMutacao3Rep[0]= i;
            cruzaVnRep[0]= i;
            rep=1;
            break;
          }else if (rep==1){
            indMutacao3Rep[1]= i;
            cruzaVnRep[1]= i;
            rep=2;
            break;
          }else if (rep==2){
            indMutacao3Rep[2]= i;
            cruzaVnRep[2]= i;
            rep=3;
            break;
          }else if (rep==3){
          }
        }else{
          rep=1;
          break;
        }
      }else{
        if (num==0){
          indice = ind;
          indMutacao3Rep[3]= i;
          cruzaVnRep[3]= i;
          rep=4;
        }else{
          indice = 0;
```

```
            }
            num = 1;
            break;
          }
        }else{
          break;
        }
    }//fim if vazio
    semente++;
  }//fim fori
  return vn;

  }//fim getVn

  public double getVn(String str, String str2, String str3,
     String str4, int semente, int n, int ind){
    double vn= 0.0;
    String auxS,auxS2,auxS3,auxS4 = "";

    for (int i= ind; i< vnS.length; i++){
      if(vnS[i]!=""){
      auxS= vnS[i].substring(0,2);
      auxS2= vnS[i].substring(3,5);
      auxS3= vnS[i].substring(6,8);
      auxS4= vnS[i].substring(9,11);
      if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
          str2.substring(0,2)))&&(auxS3.equals(str3.substring
          (0,2)))&&(auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
   substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))||((auxS.equals(str4.
   substring(0,2)))&&(auxS2.equals(str3.substring(0,2)))&&(
   auxS3.equals(str2.substring(0,2)))&&(auxS4.equals(str.
```

```
   substring(0,2))))){

         if (vnD[i]== 1000){
           gerarIndVnD(i,semente);
           vn= vnD[i];
         }else{
           vn= vnD[i];
           //checar pra ver se nao tem outra forma igual
           if ((i+1)<vnS.length-1){
             auxS= vnS[i+1].substring(0,2);
             auxS2= vnS[i+1].substring(3,5);
             auxS3= vnS[i+1].substring(6,8);
             auxS4= vnS[i+1].substring(9,11);
             if(((auxS.equals(str.substring(0,2)))&&(auxS2.
                 equals(str2.substring(0,2)))&&(auxS3.equals(
                 str3.substring(0,2)))&&(auxS4.equals(str4.
                 substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
   substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))){
               num = 0;
               indice= i+1;
               break;
             }else{
               if (num==0){
                 indice = ind;
               }else{
                 indice = 0;
               }
               num = 1;
```

```
            break;
          }
        }else{
          break;
        }
      }
    }//fim if vazio
    semente++;
  }//fim fori
  return vn;

}//fim getVn
public double getPath(String str, String str2, String str3,
   String str4, int semente, int n, int ind){
  double path= 0.0;
  String auxS,auxS2,auxS3,auxS4 = "";

  for (int i= ind; i< vnS.length; i++){
    if(vnS[i]!=""){
    auxS= vnS[i].substring(0,2);
    auxS2= vnS[i].substring(3,5);
    auxS3= vnS[i].substring(6,8);
    auxS4= vnS[i].substring(9,11);
    if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
       str2.substring(0,2)))&&(auxS3.equals(str3.substring
       (0,2)))&&(auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
  substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
  auxS4.equals(str4.substring(0,2))))||((auxS.equals(str4.
  substring(0,2)))&&(auxS2.equals(str2.substring(0,2)))&&(
  auxS3.equals(str3.substring(0,2)))&&(auxS4.equals(str.
  substring(0,2))))||((auxS.equals(str4.substring(0,2)))&&(
  auxS2.equals(str3.substring(0,2)))&&(auxS3.equals(str2.
  substring(0,2)))&&(auxS4.equals(str.substring(0,2))))){
      if (pathD[i]== 1000){
        //gerarIndPathD(i,semente);
```

```
          path= pathD[i];
        }else{
          path= pathD[i];
          break;
        }
      }
      }//fim if vazio
      semente++;
    }//fim fori
    if (path==0){
      path=1;
    }
    return path;

  }//fim getPath
  public double getN(String str, String str2, String str3,
     String str4, int semente, int ind){
    double n1= 0.0;
    String auxS,auxS2,auxS3,auxS4 = "";
    int j=0;
    int w=0;

    if (num==0){
    for (int i= ind-1; i< vnS.length; i++){
      if(vnS[i]!=""){
      auxS= vnS[i].substring(0,2);
      auxS2= vnS[i].substring(3,5);
      auxS3= vnS[i].substring(6,8);
      auxS4= vnS[i].substring(9,11);
      if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
          str2.substring(0,2)))&&(auxS3.equals(str3.substring
          (0,2)))&&(auxS4.equals(str4.substring(0,2))))||((auxS
          .equals(str.substring(0,2)))&&(auxS2.equals(str3.
          substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))
          &&(auxS4.equals(str4.substring(0,2))))||((auxS.equals
          (str4.substring(0,2)))&&(auxS2.equals(str2.substring
```

```
          (0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(auxS4.
          equals(str.substring(0,2))))||((auxS.equals(str4.
          substring(0,2)))&&(auxS2.equals(str3.substring(0,2)))
          &&(auxS3.equals(str2.substring(0,2)))&&(auxS4.equals(
          str.substring(0,2))))){
          if (n[i]== 1000){
            n1= n[i];
          }else{
            n1= n[i];
            break;
          }
        }

        }//fim if vazio
        semente++;
      }//fim fori
      }else{
        for (int i= ind; i< vnS.length; i++){
          if(vnS[i]!=""){
          auxS= vnS[i].substring(0,2);
          auxS2= vnS[i].substring(3,5);
          auxS3= vnS[i].substring(6,8);
          auxS4= vnS[i].substring(9,11);
          if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
              str2.substring(0,2)))&&(auxS3.equals(str3.substring
              (0,2)))&&(auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
    substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
    auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
    substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
    auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
    substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
    auxS4.equals(str.substring(0,2))))){
            if (n[i]== 1000){
```

```
            n1= n[i];
          }else{
            n1= n[i];
            break;
          }
        }

        }//fim if vazio
        semente++;
      }//fim fori
    }
    return n1;

  }//fim getn
  public double getPeriU(String str, String str2, String str3,
      String str4, int semente, int ind){
    double peri= 0.0;
    String auxS,auxS2,auxS3,auxS4 = "";
    int j=0;
    int w=0;

    repPeri=rep-1;
    if(num==0){
    for (int i= ind-1; i<vnS.length; i++){
      if(vnS[i]!=""){
      auxS= vnS[i].substring(0,2);
      auxS2= vnS[i].substring(3,5);
      auxS3= vnS[i].substring(6,8);
      auxS4= vnS[i].substring(9,11);
      if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
          str2.substring(0,2)))&&(auxS3.equals(str3.substring
          (0,2)))&&(auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str4.substring(0,2))))||
```

```
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
    substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
    auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
    substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
    auxS4.equals(str.substring(0,2))))){
        if((str.substring(0, 2).equals("OS"))&&(str2.substring
            (0, 2).equals("CT"))&&(str3.substring(0, 2).equals
            ("N9"))&&(str4.substring(0, 2).equals("C1"))){
          cruzaPeri[1]=i;
          if (repPeri==0){
            peri= periDInd[0]*(Math.PI/180);
            cruzaPeriRep[0]=i;
            break;
          }else if (repPeri==1){
            peri= periDInd[1]*(Math.PI/180);
            cruzaPeriRep[1]=i;
            break;
          }else if (repPeri==2){
            peri= periDInd[2]*(Math.PI/180);
            cruzaPeriRep[2]=i;
            break;
          }else if (repPeri==3){
            cruzaPeriRep[3]=i;
            rep=0;
            break;
          }
        }else if((str.substring(0, 2).equals("OS"))&&(str2.
            substring(0, 2).equals("CT"))&&(str3.substring(0,
            2).equals("N9"))&&(str4.substring(0, 2).equals("C2
            "))){
          cruzaPeri[2]=i;
          if (repPeri==0){
            peri= periDInd[0]*(Math.PI/180);
            cruzaPeriRep[0]=i;
            break;
```

```
        }else if (repPeri==1){
          peri= periDInd[1]*(Math.PI/180);
          cruzaPeriRep[1]=i;
          break;
        }else if (repPeri==2){
          peri= periDInd[2]*(Math.PI/180);
          cruzaPeriRep[2]=i;
          break;
        }else if (repPeri==3){
          cruzaPeriRep[3]=i;
          rep=0;
          break;
        }
      }else if((str.substring(0, 2).equals("OS"))&&(str2.
          substring(0, 2).equals("PP"))&&(str3.substring(0,
          2).equals("OS"))&&(str4.substring(0, 2).equals("CT
          "))){
        gerarIndPeriD(i,semente);
        peri= periD[i]*(Math.PI/180);
        cruzaPeri[3]=i;
        break;
      }else{
        peri= periD[i]*(Math.PI/180);
        break;

    }
    }//fim if vazio
    semente++;
  }//fim fori
  }else{
    for (int i= ind; i<vnS.length; i++){
      if(vnS[i]!=""){
      auxS= vnS[i].substring(0,2);
      auxS2= vnS[i].substring(3,5);
      auxS3= vnS[i].substring(6,8);
      auxS4= vnS[i].substring(9,11);
```

```
        if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
            str2.substring(0,2)))&&(auxS3.equals(str3.substring
            (0,2)))&&(auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
   substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))){
          if((str.substring(0, 2).equals("OS"))&&(str2.
              substring(0, 2).equals("CT"))&&(str3.substring(0,
               2).equals("N9"))&&(str4.substring(0, 2).equals("
              C1"))){
            if (repPeri==3){
              peri= periDInd[3]*(Math.PI/180);
              cruzaPeriRep[3]=i;
              rep=0;
              repPeri= 0;
              break;
            }else{
              peri= periD[i]*(Math.PI/180);
              cruzaPeri[1]=i;
              break;
            }
          }else if((str.substring(0, 2).equals("OS"))&&(str2.
              substring(0, 2).equals("CT"))&&(str3.substring(0,
               2).equals("N9"))&&(str4.substring(0, 2).equals("
              C2"))){
            if (repPeri==3){
              peri= periDInd[3]*(Math.PI/180);
              cruzaPeriRep[3]=i;
              rep=0;
              repPeri= 0;
```

```
            break;
          }else{
            peri= periD[i]*(Math.PI/180);
            cruzaPeri[2]=i;
            break;
          }
        }else if((str.substring(0, 2).equals("OS"))&&(str2.
            substring(0, 2).equals("PP"))&&(str3.substring(0,
            2).equals("OS"))&&(str4.substring(0, 2).equals("
            CT"))){
          gerarIndPeriD(i,semente);
          peri= periD[i]*(Math.PI/180);
          cruzaPeri[3]=i;
          break;
        }else{
          peri= periD[i]*(Math.PI/180);
          repPeri=0;
          rep=0;
          break;
        }
      }
      }//fim if vazio
      semente++;
    }//fim fori
  }
  return peri;

}//fim getPeri
public double getPeri(String str, String str2, String str3,
    String str4, int semente, int ind){
  double peri= 0.0;
  String auxS,auxS2,auxS3,auxS4 = "";
  int j=0;
  int w=0;

  if(num==0){
```

```
    for (int i= ind-1; i<vnS.length; i++){
      if(vnS[i]!=""){
      auxS= vnS[i].substring(0,2);
      auxS2= vnS[i].substring(3,5);
      auxS3= vnS[i].substring(6,8);
      auxS4= vnS[i].substring(9,11);
if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str2.
   substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
   auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
   substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))){

        if (periD[i]== 1000){
          peri= ((periD[i]*Math.PI)/180);
        }else{
          peri= (periD[i]*(Math.PI/180));
          break;
        }
      }
      }//fim if vazio
      semente++;
    }//fim fori
    }else{
      for (int i= ind; i<vnS.length; i++){
        if(vnS[i]!=""){
        auxS= vnS[i].substring(0,2);
        auxS2= vnS[i].substring(3,5);
        auxS3= vnS[i].substring(6,8);
        auxS4= vnS[i].substring(9,11);
```

```
        if(((auxS.equals(str.substring(0,2)))&&(auxS2.equals(
            str2.substring(0,2)))&&(auxS3.equals(str3.substring
            (0,2)))&&(auxS4.equals(str4.substring(0,2))))||((
            auxS.equals(str.substring(0,2)))&&(auxS2.equals(
            str3.substring(0,2)))&&(auxS3.equals(str2.substring
            (0,2)))&&(auxS4.equals(str4.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str2.
   substring(0,2)))&&(auxS3.equals(str3.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))||
((auxS.equals(str4.substring(0,2)))&&(auxS2.equals(str3.
   substring(0,2)))&&(auxS3.equals(str2.substring(0,2)))&&(
   auxS4.equals(str.substring(0,2))))){

          if (periD[i]== 1000){
            peri= ((periD[i]*Math.PI)/180);
          }else{
            peri= ((periD[i]*Math.PI)/180);
            break;
          }
          //falta verificar se tem outra estrutura como nos
             metodos Vn e Path
        }
        }//fim if vazio
        semente++;
      }//fim fori
    }
    return peri;

  }//fim getPeri
  public void gerarIndAngd (int semente){
    int i=0;
    //soh as estruturas (4)
    for (int j= 0; j< 4; j++){
      vnDInd[j]= gerarIndVnD(j, semente);
      periDInd[j]= gerarIndPeriD(j,semente);
      semente= semente+100;
```

```
    }
  }
  public void gerarIndAngdCp (int semente){
    int i=0;
    //soh as estruturas (4)
    for (int j= 28; j< 32; j++){
      vnDInd[i]= vnD[j];
      periDInd[i]= periD[j];
      i= i+1;
    }
  }
  public double gerarIndVnD (int ind, int semente){
    //valor maximo que pode assumir (intervalo)
    double cte = 2.0;
    String cteS= "";

    if (semente==0){
      semente=semente+1;
    }
    getSemente(semente);
    vnDInd[ind]= random()*cte;
    //arredondamento
    vnDInd[ind] = (Math.round((vnDInd[ind] * 10000.0))) /
       10000.0;
    while (vnDInd[ind]==0.000){
      vnDInd[ind]= random()*cte;
      vnDInd[ind] = (Math.round((vnDInd[ind] * 10000.0))) /
         10000.0;
    }
    if (vnDInd[ind]<0){
      vnDInd[ind]=vnDInd[ind]*(-1);
    }
    return vnDInd[ind];

  }
  public double gerarIndPeriD (int ind, int semente){
```

```
    //valor maximo que pode assumir (intervalo)
    double cte = 190;

    if (semente==0){
      semente=semente+1;
    }
    getSemente(semente);
    //180 eh pq o valor do artigo 2011 mostra esse valor
       (179.35)
    periDInd[ind]= random()* cte;
    //arredondamento
    periDInd[ind] = (Math.round((periDInd[ind] * 100.0))) /
       100.0;
    if (periDInd[ind]<0.0){
      periDInd[ind]=periDInd[ind]*(-1);
    }
    return periDInd[ind];

}
public void mutacao (double tm, int pos, int gera, int
    semente){
    double pm= 0;
    double sinal =0.0;
    double pmRep= 0;

        objAtomos.getSemente(semente);
      //linhas
        //sao soh 4 pq soh tenho 4 diedros que posso mutar
        for (int i= 0; i< 4; i++){
          pm= (objAtomos.random());
          sinal= (objAtomos.random());
          if (pm<= tm){//0.5) {
            if (sinal<=0.5){
              vnDInd[i]= vnDInd[i]-(objAtomos.random()*0.02);
              vnDInd[i] = (Math.round((vnDInd[i] * 10000.0)))
                  / 10000.0;
```

```
            periDInd[i]= periDInd[i]-(objAtomos.random()
                *0.02);
            periDInd[i] = (Math.round((periDInd[i] *
                10000.0))) / 10000.0;
          }else{
            vnDInd[i]= vnDInd[i]+(objAtomos.random()*0.02);
            vnDInd[i] = (Math.round((vnDInd[i] * 10000.0)))
                / 10000.0;
            periDInd[i]= periDInd[i]+(objAtomos.random()
                *0.02);
            periDInd[i] = (Math.round((periDInd[i] *
                10000.0))) / 10000.0;
          }//fim else
        }//fim if
          if (vnDInd[i]<0){
            vnDInd[i]= vnDInd[i]*(-1);
          }
          if (vnDInd[i]==0.0){
            vnDInd[i]= vnDInd[i]+(0.01);
          }
          if (periDInd[i]<0){
            periDInd[i]= periDInd[i]*(-1);
          }
    }//fim for
  }//fim mutacao
  public void setFit (double f){
    fit = f;
  }//fim setFit
  public String getIndM() throws JXLException, IOException {
    String aux= "";
    double[] atm = new double[3];

    for (int i= 0; i< vnDInd.length; i++){
      aux= aux+"\n"+" VnD:\t"+String.valueOf(vnDInd[i])+"\t"+
          String.valueOf(periDInd[i]);
    }
```

```
    vetorMol = "" + aux;
    calculaFit(10);
    return vetorMol;

  }//fim getIndM

}//fim class Individuo
```

Anexo A

Currículo do autor

Doutorando em bioinformática pela Universidade Federal de Minas Gerais (2014) nas temáticas algoritmos genéticos, cálculos quânticos e campo de força. Graduado em Bachareledo em Ciência da Computação pelo Centro Universitário do Pará (2002). Mestrado em Engenharia Elétrica aplicada a Computação com ênfase em Inteligência Artificial pela Universidade Federal do Pará (2008). Experiência de coodenador em dois projetos de pesquisas (2004-2006). Experiência no serviço público trabalhando como analista de sistema e de suporte (2008-2010).

A.1 Formação acadêmica

1999-2002 Graduação em Bachareledo em Ciência da Computação pelo Centro Universitário do Pará (CESUPA).

2005-2008 Mestrado em Engenharia Elétrica aplicada a Computação com ênfase em Inteligência Artificial pela Universidade Federal do Pará (UFPa).

2010- Doutorando em Bioinformática pela Universidade Federal de Minas Gerais (UFMG).

2013 KATO, Rodrigo Bentes; SILVA, F.; PAPPA, G. L.; BELCHIOR, J. C. Refinement parameters of he AMBER nucleic acids force field using genetic algorithms and ab initio calculations.

2012 KATO, R. B.; BAPTISTA, R. P.; COUTO, B. R. G. M.; SANTOS, M. A.; PAPPA, G. L.; BELCHIOR, J. C.; MACEDO, A. M. Detection of target sites in

genomic sequences for phylogenetic studies using logistic regression: application on T.cruzi amastin gene.

2008 KATO, Rodrigo Bentes; OLIVEIRA, Roberto Célio Limão de. Classificação de Dados Utilizando Algoritmo Genético e Logica Fuzzy.

2005 GONÇALVES, Aruanda Simões; FREITAS, Alex Alves; KATO, Rodrigo Bentes; OLIVEIRA, Roberto Célio Limão de. Using Genetic Algorithms to Mine Interesting Dependence Modeling Rules.

A.2 Contato

Email rbkato@gmail.com