Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Modelagem Matemática e Computacional

EXPLORANDO O ESPAÇO DE PARÂMETROS DO MÉTODO SEMIEMPÍRICO RM1 PELA UTILIZAÇÃO DE OTIMIZAÇÃO NÃO-LINEAR

Rafaela Souza Morais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional, UFPB, da Universidade Federal da Paraíba, como parte dos requisitos necessários à obtenção do título de Mestre em Modelagem Matemática e Computacional.

Orientador: Gerd Bruno da Rocha

João Pessoa
Fevereiro de 2022

EXPLORANDO O ESPAÇO DE PARÂMETROS DO MÉTODO SEMIEMPÍRICO RM1 PELA UTILIZAÇÃO DE OTIMIZAÇÃO NÃO-LINEAR

Rafaela Souza Morais

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL (PPGMMC) DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DA PARAÍBA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM MODELAGEM MATEMÁTICA E COMPUTACIONAL.

Examinada por:

Prof. Dr. Gerd Bruno da Rocha
(Presidente)

Dr. Gabriel Aires Urquiza de Carvalho
(Membro Externo Titular)

Prof. Dr. Hugo Leonardo Davi de Souza Cavalvante
(Membro Interno Titular)

JOÃO PESSOA, PB – BRASIL
FEVEREIRO DE 2022

M827e Morais, Rafaela Souza.
Explorando o espaço de parâmetros do método semiempírico RM1 pela utilização de otimização não-linear / Rafaela Souza Morais. – João Pessoa, 2022.
102 f. : il.

Orientador: Gerd Bruno da Rocha
Dissertação (Mestrado) – UFPB/CI.

1. Química Quântica. 2. RM1. 3. Métodos semiempíricos. 4. Otimização não-linear. 5. Algoritmos genéticos. I. Rocha, Gerd Bruno da. II. Título.

UFPB/BC CDU: 544.18(043)

A todos aqueles que acreditam na ciência.

Agradecimentos

Ao meu filho Ravi, por seu amor que me enche de felicidade e me deu forças para superar os momentos difíceis.

Aos meus pais, que me proporcionaram a base para mais esta conquista.

Aos meus irmãos Ansaldo, Renata e Rosa.

Ao meu esposo Rodrigo, por seu apoio, carinho e paciência.

Ao sempre atencioso professor Gerd, pela orientação e apoio, e por me guiar com maestria a cada etapa. O amor que tem à sua profissão serviu de grande incentivo para que eu chegasse até aqui e não deixou que eu desistisse.

Ao professor Jairo, por me receber em sua turma de graduação para o estágio de docência.

A Gabriel e ao professor Hugo, por aceitarem o convite para participação da banca e por suas contribuições para melhorias e conclusão deste trabalho.

Aos professores do PPGMMC: Boness, Claudio, Jan Sokolowski, José Miguel, Pedro, Tarciana, Thiagão e novamente a Gerd e Jairo, os quais me deram aulas e forneceram uma base sólida para meu crescimento acadêmico.

A todos os colegas do mestrado, em especial a Anderson e Josuel, por seu companheirismo e disposição em ajudar. E também pelos momentos de descontração.

A Gean, por ter sido sempre solícito quando precisei.

Às instituições de fomento CAPES e CNPq e à UFPB, pelo financiamento do mestrado e disponibilização do Laboratório de Química Quântica Computacional (LQQC).

Ao Laboratório Nacional de Computação Científica (LNCC/MCTI), por prover os recursos de computação de alto desempenho do supercomputador SDumont, que contribuíram com os resultados reportados neste trabalho. URL: `http://sdumont.lncc.br`.

E por fim, a todos aqueles que, direta ou indiretamente, contribuíram comigo para a realização deste trabalho.

Resumo da Dissertação apresentada ao PPGMMC/CI/UFPB como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

EXPLORANDO O ESPAÇO DE PARÂMETROS DO MÉTODO SEMIEMPÍRICO RM1 PELA UTILIZAÇÃO DE OTIMIZAÇÃO NÃO-LINEAR

Rafaela Souza Morais

Fevereiro/2022

Orientador: Gerd Bruno da Rocha

Programa: Modelagem Matemática e Computacional

A modelagem molecular permite calcular propriedades dos compostos moleculares, sendo utilizada principalmente na descoberta de novos fármacos ou na melhoria de protótipos existentes. Há várias abordagens para gerar esses modelos, sendo os métodos semiempíricos as alternativas mais eficientes computacionalmente, porém com uma acurácia que varia muito, dependendo da abordagem e dos parâmetros escolhidos ou ajustados. Um desses métodos é o RM1 (*Recife Model 1*), criado em 2006 como uma reparametrização do AM1 (*Austin Model 1*), um modelo muito bem-sucedido. O RM1 obteve bons resultados, mas é importante avaliar se a parametrização escolhida foi a melhor possível. Neste trabalho, o espaço de parâmetros para o método RM1 foi explorado, utilizando uma variação do algoritmo de otimização não-linear DFP a partir de diferentes pontos, avaliando se é possível oferecer uma melhoria substancial em sua exatidão unicamente com uma reparametrização. Usamos, como pontos de partida, parâmetros obtidos de um trabalho anterior do grupo, que utilizou algoritmos genéticos, que no momento de sua avaliação se mostraram ligeiramente melhores que o RM1. O procedimento de parametrização executado no presente trabalho conseguiu obter pontos que melhoraram apenas as distâncias de ligação, contudo, no geral, não conseguiu obter pontos melhores que os obtidos quando se empregaram algoritmos genéticos. Atribuímos a isso desequilíbrio na função custo usada, que provavelmente considerou os erros em distâncias de ligação como os preferenciais de serem reduzidos em detrimento das outras propriedades. Nossa conclusão é que, para melhorar os resultados, se faz necessário um estudo prévio mais amplo da função custo, para que se chegue em um conjunto de pesos adequados para se obter uma parametrização que se mostre mais eficiente.

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

EXPLORING THE PARAMETER SPACE FOR THE RM1 SEMIEMPIRICAL METHOD BY USING NONLINEAR OPTIMIZATION

Rafaela Souza Morais

February/2022

Advisor: Gerd Bruno da Rocha

Program: Computational Mathematical Modelling

Molecular modeling allows us to calculate properties of molecular compounds, being used mainly in the discovery of new drugs or in the improvement of existing prototypes. There are several ways to generate these models, with semiempirical methods being the most computationally efficient alternatives, but with an accuracy that varies a lot, depending on the approach and on the chosen or adjusted parameters. One such method is RM1 (Recife Model 1), created in 2006 as a reparameterization of AM1 (Austin Model 1), a very successful model. RM1 achieved good results, but it is important to assess whether the chosen parameterization was the best possible. In this work, the parameter space for the RM1 method was explored, using a variation of the nonlinear optimization algorithm DFP starting from different points, evaluating whether it is possible to offer a substantial improvement in its accuracy only with a reparameterization. We used, as starting points, parameters obtained from a previous work by the group, which used genetic algorithms, which at the time of their evaluation showed to be slightly better than RM1. The parameterization procedure performed in the present work was able to obtain points that only improved the bond distances, however, in general, it was not able to obtain better points than those obtained when genetic algorithms were used. We attribute this to an imbalance in the cost function used, which probably considered errors in bond distances to be preferred to be reduced to the detriment of other properties. Our conclusion is that, in order to improve the results, a broader prior study of the cost function is necessary, in order to reach a set of adequate weights to obtain a more efficient parameterization.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

E	Energia, p. 7
F_{resp}	Função resposta ou função custo, que representa o erro total calculado, p. 33
H_k	Matriz de métrica variável utilizada em métodos quase-Newton, durante o passo k, p. 22
R_{ij}	Distância entre os átomos i e j, p. 13
V	Energia potencial, p. 7
Z_i	Número de elétrons de valência do átomo i, p. 13
Ψ	Função de onda, p. 7
$\alpha_{A[B]}$	Termo de repulsão núcleo-núcleo entre átomos A [e B], p. 12
β_i	Termo da integral de ressonância de um elétron e dois centros, para o orbital i, p. 12
λ	Autovalor de uma matriz, p. 8
μ	Momento dipolar, p. 34
ω	Passo aplicado à direção de um método de otimização não-linear, p. 18
$\widehat{H}$	Operador Hamiltoniano, p. 7
ξ_i	Expoente do orbital i do tipo Slater, p. 12
ξ_{in}	Expoente interno do orbital i do tipo Slater, p. 12
d	Vetor de direção utilizado em um método de otimização não-linear, p. 18
h	Constante de Planck, p. 7

m	Massa da partícula, p. 7
q^{calc}	Valor calculado para uma propriedade molecular por um método semiempírico, p. 33
q^{exp}	Valor de referência para uma propriedade molecular, p. 33
G_{ij}	Parâmetro para integral de repulsão de um centro e dois elétrons, para orbitais $i - j$, p. 12
H_{sp}	Parâmetro para integral de troca de um centro e dois elétrons, para orbitais s-p, p. 12
U_{ii}	Parâmetro para integral de um elétron e um centro, para o orbital i, p. 12
a_{nA}	Multiplicador gaussiano para a enésima Gaussiana do átomo A, p. 12
b_{nA}	Multiplicador do expoente gaussiano para a Gaussiana do átomo A, p. 12
c_{nA}	Parâmetro para o raio do centro da enésima Gaussiana do átomo A, p. 12
w_j	Peso aplicado ao erro de uma propriedade calculada, para a molécula j, p. 33

Lista de Abreviaturas

AG	Algoritmo genético, p. 25
AM1	*Austin Model 1*, p. 2
BFGS	Broyden-Fletcher-Goldfarb-Shanno, p. 25
DFP	Davidon-Fletcher-Powell, p. 3
DFT	*Density Functional Theory*, p. 2
DQ	Departamento de Química, p. 42
HFR	Hartree-Fock-Roothaan, p. 9
LNCC	Laboratório Nacional de Computação Científica, p. 31
LQQC	Laboratório de Química Quântica Computacional, p. 42
MM	Modelagem Molecular, p. 1
MNDO	*Modified Neglect of Diatomic Overlap*, p. 2
MOPAC	*Molecular Orbital PACkage*, p. 11
OMx	*Orthogonalization Model x*, p. 16
PI	Potencial de Ionização, p. 34
PM3	*Parametric Method 3*, p. 11
PM5	*Parametric Method 5*, p. 14
PM6	*Parametric Method 6*, p. 11
PM7	*Parametric Method 7*, p. 16
RM1	*Recife Model 1*, p. 2
SDumont	Supercomputador Santos Dumont, p. 42

UFPB Universidade Federal da Paraíba, p. 42

VPN *Virtual Private Network*, p. 81

Capítulo 1

Introdução

O desenvolvimento de métodos teóricos de química quântica exige um esforço combinado de aplicação de técnicas numéricas de otimização com conhecimento químico. O desafio é conseguir o melhor custo-benefício entre o tempo de cálculo computacional e a exatidão dos resultados. O interesse em desenvolver tais métodos é que esses possam modelar sistemas moleculares e/ou materiais de interesse na sociedade.

Uma das principais utilizações da modelagem molecular (MM) é a descoberta de novos fármacos, bem como a melhoria de protótipos já existentes. Com o avanço dos recursos computacionais e da química computacional, a MM teve um grande desenvolvimento nos últimos anos [1].

Na modelagem de átomos e moléculas, o modelo matemático mais exato utiliza a equação de Schrödinger, que permite calcular as funções de onda e as energias permitidas aos elétrons [2]. Como essa é uma equação diferencial cuja solução analítica não é conhecida para sistemas multieletrônicos, sua utilização na descrição de sistemas mais realistas exige a aplicação de cálculos numéricos custosos e modelos físico matemáticos com aproximações. Por isso, quando há muitos átomos envolvidos, esse cálculo é extremamente lento ou as vezes impraticável. Portanto, foram buscados métodos que introduzissem parâmetros empíricos ou previamente calculados e também que introduzissem aproximações nos modelos mais exatos com o objetivo de diminuir esse tempo de cálculo, que são chamados métodos semiempíricos. Esses parâmetros podem ser ajustados para conseguir obter uma melhor exatidão nos resultados.

Métodos semiempíricos de química quântica são métodos aproximados de modelagem molecular que usam estratégias computacionais para diminuir a complexidade computacional dos cálculos de estrutura eletrônica com o mínimo prejuízo para suas exatidões [3]. Eles são uma alternativa importante para o cálculo de centenas de moléculas por vez, como no projeto de fármacos e no *design* de moléculas com propriedades específicas. Também são muito úteis no cálculo de funções de onda de

estruturas moleculares e supramoleculares com milhares de átomos, como proteínas [4].

Os métodos semiempíricos foram propostos como uma forma de obter resultados com boa exatidão e que não demandem muito tempo de processamento computacional. Antes deles, as alternativas numéricas para utilizar o formalismo de Schrödinger para prever propriedades atômico-moleculares foram os métodos *ab initio*, que fazem cálculos sem a necessidade de parâmetros empíricos e alcançam excelentes resultados, mas exigem um alto processamento, o que limita bastante o número de átomos calculados [5]. Uma alternativa mais eficiente foi a Teoria do Funcional da Densidade - DFT (*Density Functional Theory*), que possui um custo-benefício melhor [5]. Porém, os métodos semiempíricos possuem cálculos ainda mais rápidos, importantes no caso de moléculas maiores, e sua exatidão depende da abordagem e dos dados utilizados [5].

Há vários métodos semiempíricos, sendo o MNDO (*Modified Neglect of Diatomic Overlap*) [6] um precursor dos métodos semiempíricos modernos. Esse modelo foi melhorado com o método AM1 (*Austin Model 1*), criado em 1985 [7], que passou a usar funções gaussianas esféricas para calcular as integrais de repulsão núcleo-núcleo (o que permitiu descrever melhor alguns aspectos das ligações de hidrogênio) e usou mais dados experimentais na busca de melhores parâmetros [8]. A melhoria alcançada por essa abordagem tornou esse modelo muito bem-sucedido, com mais de 17 mil citações no Google Acadêmico [9].

Em 2006, o grupo de pesquisa de Rocha *et al* apresentou o modelo RM1 (*Recife Model 1*) [10], uma reparametrização do modelo AM1 [7]. Na época, a estratégia foi manter a mesma estrutura algébrica do modelo AM1, para que o novo método pudesse ser rapidamente incorporado aos softwares de química quântica então existentes. Tal foi feito e atualmente o RM1 encontra-se implementado em softwares tão variados quanto o MOPAC, Spartan, Hyperchem, AMPAC, Amber, PCGamess/Firefly, pDynamo, Phenix eLBOW, ConGENER, e muitos outros através de suas interfaces com o MOPAC2016 [11][12]. O artigo com o RM1 apresenta, até o momento, 810 citações no Google Acadêmico [13].

A proposta deste trabalho foi explorar o espaço de parâmetros do RM1, utilizando uma função resposta (ou função custo) que usa os valores dos parâmetros como variáveis de entrada e tem como saída um valor que representa o erro associado ao método. Foram utilizados como pontos de partida 10 conjuntos de parâmetros encontrados em um projeto anterior, que utilizou um algoritmo genético para buscar múltiplos mínimos para a função, obtendo parâmetros que conseguiram melhorar um pouco alguns resultados do RM1 [14].

Neste trabalho, foi feito um tratamento adicional aos dados encontrados pelo algoritmo genético, procurando novos mínimos locais nas proximidades dos melho-

res pontos fornecidos por ele. Para isso, foram executadas rotinas de otimização independentes para avaliar se é possível obter resultados ainda melhores. Esse procedimento foi realizado por meio do programa PARAM [15], que é um programa pronto, cedido pelo prof. James Stewart, capaz de minimizar a função resposta por meio do ajuste dos parâmetros do método semiempírico. O presente trabalho não desenvolveu nenhum código adicional ao programa, apenas o utilizou para realizar a busca dos mínimos locais e analisou os resultados.

Para decidir o próximo conjunto de parâmetros após cada ciclo, na busca por um mínimo local para a função resposta, o PARAM utiliza uma adaptação do algoritmo de otimização não-linear DFP (Davidon-Fletcher-Powell). As principais funcionalidades do PARAM são o cálculo da função resposta e a otimização de parâmetros, e esta utiliza a adaptação do DFP e atualiza a função resposta em cada ciclo.

O DFP é um método quase-Newton, tendo como base os métodos de Newton, do gradiente e da secante. Em geral, ele é mais rápido para encontrar o mínimo do que os métodos de Newton e do gradiente [16]. O uso de algoritmos eficientes é essencial neste projeto, pois o cálculo da função custo em cada ciclo tem um tempo de processamento muito elevado, mesmo em computadores de alto desempenho, por isso é importante escolher um algoritmo que se aproxime do mínimo local em poucos ciclos.

O cálculo da função resposta realizado pelo PARAM depende do conjunto de parâmetros utilizado (que pode ser representado pela estrutura de dados de um vetor de n posições, onde n é o número de parâmetros) e do banco de dados de moléculas, sobre as quais o método é aplicado. A Figura 1.1 (criada com o *site* Canva [17]) representa essas dependências em um esquema.

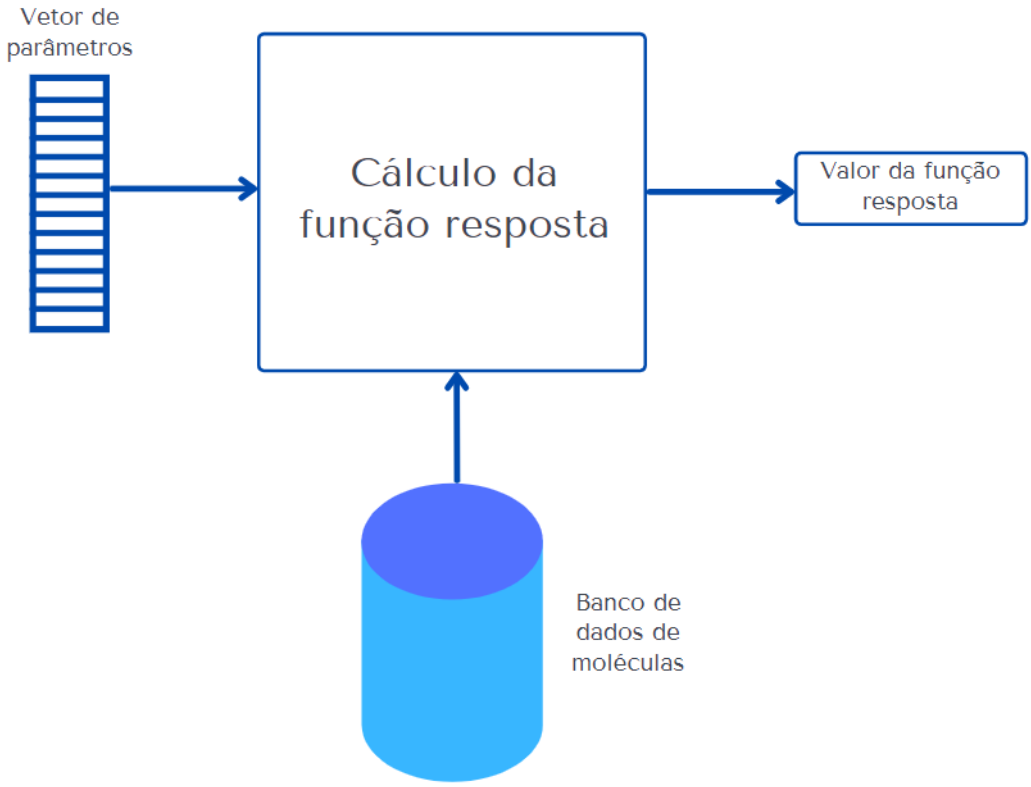


Figura 1.1: Dependências do cálculo da função resposta.

O banco de dados de moléculas contém as coordenadas que definem as moléculas e os valores de diversas propriedades para cada molécula, medidos experimentalmente ou calculados previmente por métodos com maior exatidão, podendo ser *ab initio* e/ou DFT. Alguns exemplos de propriedades incluídas são a entalpia de formação e o potencial de ionização. O que um método semiempírico faz é calcular essas propriedades tomando como base apenas as coordenadas da molécula. Então, as propriedades calculadas podem ser comparadas com as experimentais para verificar a exatidão do método. A Figura 1.2 (também criada no *site* Canva [17]) demonstra esse funcionamento.

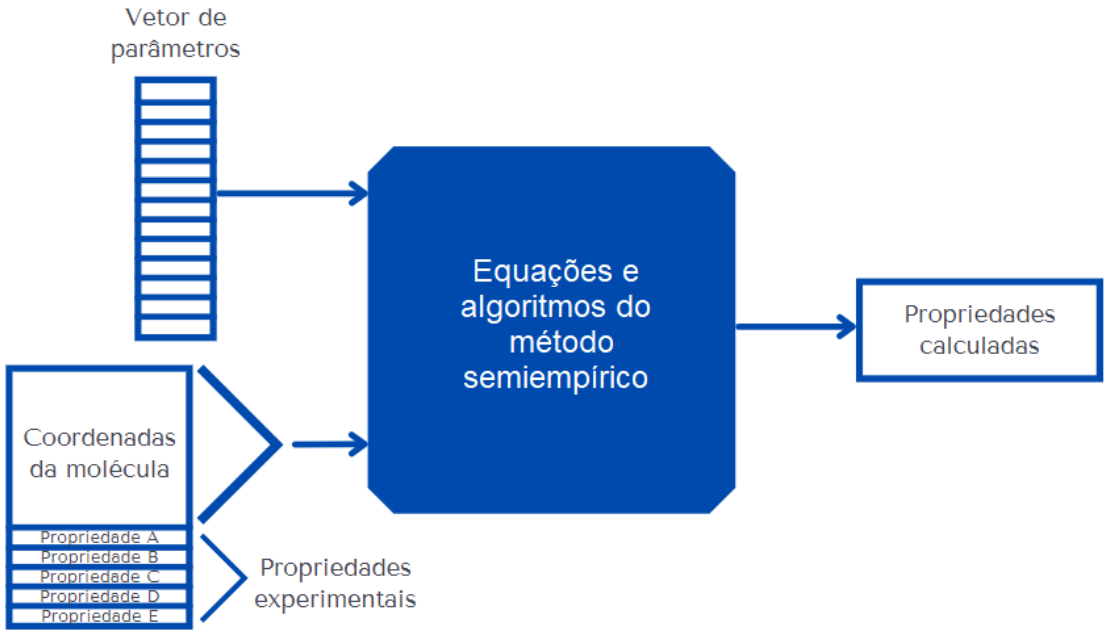


Figura 1.2: Funcionamento de um método semiempírico aplicado a uma mólécula de um banco de dados.

O formalismo matemático dos métodos semiempíricos e os algoritmos empregados para o cálculo das suas equações, junto com os parâmetros utilizados, determinam a exatidão do método. Uma aproximação para definir quão exato o método é pode ser obtida por uma soma de erros quadrados ponderados, considerando um elevado número de moléculas. Para cada molécula, a diferença entre o valor calculado e o experimental de cada propriedade de interesse é elevada ao quadrado e multiplicada por um peso ao quadrado que depende da propriedade. O cálculo da função resposta realiza esse procedimento em milhares de moléculas e fornece a soma total obtida.

Como, durante a execução da parametrização, o conjunto de moléculas considerado é fixo, ele pode ser considerado como parte do que define a função resposta. A estrutura do método também é mantida, então a única mudança que ocorre cada vez que a função é calculada é no vetor de n parâmetros do método. Portanto, a função resposta pode ser definida como uma função $f : \mathbb{R}^n \to \mathbb{R}$.

No decorrer deste trabalho, a adaptação do método DFP provida pelo PARAM foi suficiente para encontrar parâmetros que alcançaram uma considerável redução na função resposta, porém alguma característica não identificada da função se mostrou inadequada quando foi verificado o erro por propriedade, que aumentou consideravelmente em vez de diminuir, para quase todas as propriedades. Isso levou a uma sugestão de procedimento para um trabalho futuro.

1.1 Objetivos

Nesse trabalho buscamos avaliar o espaço de parâmetros do método semiempírico RM1, analisando a viabilidade de uma reparametrização, utilizando o programa PARAM para aplicar técnicas de otimização não-linear e minimizar o erro associado.

1.2 Organização do Trabalho

O trabalho está organizado da seguinte forma:

Capítulo 2: Discorre sobre a fundamentação teórica relacionada com os objetivos do trabalho.

Capítulo 3: Discorre sobre os procedimentos de duas abordagens para busca de novos parâmetros para o RM1: um trabalho anterior, que utilizou algoritmos genéticos, e a execução deste trabalho, que buscou mínimos locais nas proximidades de 10 pontos encontrados pelo trabalho anterior.

Capítulo 4: Discorre sobre os resultados obtidos na minimização da função e nos testes de validação dos parâmetros obtidos.

Capítulo 5: Apresenta as conclusões obtidas neste trabalho.

Capítulo 2

Fundamentação Teórica

2.1 Modelagem molecular

Métodos de modelagem molecular quânticos utilizam conceitos da química quântica para representar a estrutura molecular e a energia correspondente [18]. Isso permite a descoberta de novos fármacos e a melhoria de protótipos já existentes.

Existem várias abordagens para essas modelagens, mas todas usam como base a equação de Schrödinger.

2.1.1 Equação de Schrödinger

Essa equação, em sua forma independente do tempo, tem a seguinte fórmula simplificada:

$$\widehat{H}\Psi = E\Psi \tag{2.1}$$

Onde $\widehat{H}$ é o operador Hamiltoniano, Ψ é a função de onda (dependente das coordenadas espaciais e de spin) e E é a energia. No caso simples de uma partícula de massa m que tem movimento unidimensional e energia potencial $V(x)$, o Hamiltoniano é dado por:

$$\widehat{H} = \left[-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\right] \tag{2.2}$$

Onde:

$$\hbar = \frac{h}{2\pi} \tag{2.3}$$

E h é a constante de Planck, que corresponde a $6,626 \times 10^{-34}$ J·s [19].

Uma vez encontrada a função de onda do sistema (que é a solução da Equação de Schrödinger), pode-se obter qualquer observável física, desde que se escreva um operador para essa observável na linguagem da mecânica quântica. Por exemplo, se quisermos encontrar a energia total do sistema, basta fazermos atuar o operador Hamiltoniano na função de onda encontrada.

Podemos entender melhor essa equação comparando-a com os autovalores e autovetores de um tensor (transformação linear, representada por uma matriz). Autovetores são vetores $\mathbf{u}$ não nulos tais que, quando a transformação linear T é aplicada, o vetor resultante é múltiplo de $\mathbf{u}$, isto é:

$$T\mathbf{u} = \lambda\mathbf{u} \tag{2.4}$$

Onde $\mathbf{u}$ é um autovetor de T e o escalar λ é o seu autovalor correspondente.

Assim, podemos observar uma semelhança entre as equações 2.1 e 2.4. Da mesma forma como alguns vetores admitem autovalores para T, algumas funções de onda Ψ podem ser transformadas pelo operador $\widehat{H}$ de forma que o resultado seja o produto de uma energia E pela mesma função Ψ. Nesses casos, Ψ é chamada de autofunção de $\widehat{H}$ com autovalor E [20]. Apenas algumas energias específicas podem ser autovalores de $\widehat{H}$, o que é uma consequência de a energia ser quantizada.

Um exemplo simples de autofunção é a função e^{2x}, que é autofunção do operador d/dx com autovalor 2, dado que:

$$(d/dx)e^{2x} = 2e^{2x} \tag{2.5}$$

Como o operador $\widehat{H}$ utilizado na equação 2.1 aplica derivadas à função de onda, ela é uma equação diferencial, cuja solução (identificação das funções e energias correspondentes) utiliza integrais que têm alta complexidade no caso do movimento dos elétrons. Schrödinger resolveu essa equação de forma analítica para o caso do átomo de hidrogênio, o que permitiu calcular com exatidão os níveis de energia permitidos nesse caso e no de outros sistemas monoeletrônicos [19]. Mas quando há mais do que um elétron, a parcela de repulsão entre eles torna essa equação diferencial não-separável em qualquer sistema de coordenadas, o que impossibilita a sua solução analítica. Por isso é necessário utilizar métodos numéricos ou aproximações para resolver essa equação.

Assim como o operador Hamiltoniano permite encontrar as energias possíveis, há outros operadores aplicáveis à mesma função de onda para obter outras propriedades

observáveis, como a posição e o momento linear. Por isso o conhecimento da função de onda tem alta relevância na modelagem molecular.

2.1.2 Método de Hartree-Fock-Roothaan

O método de Hartree-Fock foi desenvolvido por Douglas Hartree e Vladimir Fock, nos anos seguintes ao da publicação da equação de Schrödinger [21]. É um método *ab initio* baseado na equação de Schrödinger, que a modifica para uma aproximação e utiliza o método variacional para se aproximar do valor exato procurado. Inicialmente, Hartree propôs, como aproximação inicial para o Hamiltoniano de muitos elétrons, o produto das funções de onda individuais de cada elétron. Anos depois, Vladimir Fock reformulou essa aproximação (com a colaboração de outros cientistas, como John Slater) para considerar o caráter antissimétrico dos elétrons, surgindo assim o método de Hartree-Fock [21].

O efeito dessa aproximação inicial para os orbitais é que, no lugar da repulsão elétron-elétron, considera-se a repulsão do elétron com uma densidade média dos demais elétrons. Efetuados os cálculos, a função é atualizada até convergir.

Graças a uma contribuição de Clemens Roothaan, surgiu uma atualização desse método que o tornou mais eficiente, passando a se chamar de método de Hartree-Fock-Roothaan (HFR). Ele passou a representar a integral de cada orbital como uma combinação linear de um conjunto de funções de base [22]. Dessa forma, o problema passou a ser resolvido com uma álgebra matricial.

O valor esperado para a energia de um sistema de elétrons é calculado pelo método de HFR com a seguinte fórmula:

$$E = \left\langle \Psi | \widehat{H} | \Psi \right\rangle = 2 \sum_{i}^{n} h_{ii} + \sum_{i}^{n} \sum_{j}^{n} (2J_{ij} - K_{ij}) \tag{2.6}$$

Onde h_{ii} é uma integral de um elétron, J_{ij} é uma integral de Coulomb, representando a repulsão do elétron i com o elétron j, e K_{ij} é uma integral de troca, que simula a impossibilidade de dois elétrons ocuparem o mesmo orbital com o mesmo spin (princípio de exclusão de Pauli). As fórmulas dessas integrais são as seguintes:

$$h_{ii} = \int \psi_i(1) \widehat{H}^{core}(1) \psi_i(1) d\tau_1 = \sum_{\mu} \sum_{\nu} c_{\mu}^{(i)} c_{\nu}^{(i)} h_{\mu\nu} \tag{2.7}$$

$$J_{ij} = \int\int \psi_i(1)\psi_j(2)\frac{1}{r_{12}}\psi_i(1)\psi_j(2)d\tau_1 d\tau_2 = \sum_{\mu}\sum_{\nu}\sum_{\lambda}\sum_{\sigma} c_{\mu}^{(i)}c_{\nu}^{(j)}c_{\lambda}^{(i)}c_{\sigma}^{(j)}(\mu\nu|\lambda\sigma) \tag{2.8}$$

$$K_{ij} = \int\int \psi_i(1)\psi_j(2)\frac{1}{r_{12}}\psi_j(1)\psi_i(2)d\tau_1 d\tau_2 = \sum_{\mu}\sum_{\nu}\sum_{\lambda}\sum_{\sigma} c_{\mu}^{(i)}c_{\nu}^{(j)}c_{\lambda}^{(j)}c_{\sigma}^{(i)}(\mu\sigma|\nu\lambda) \tag{2.9}$$

Onde a notação $(\mu\nu|\lambda\sigma)$ representa uma integral dupla dada por:

$$(\mu\nu|\lambda\sigma) = \int\int \phi_{\mu}(1)\phi_{\nu}(2)\frac{1}{r_{12}}\phi_{\lambda}(1)\phi_{\sigma}(2)d\tau_1 d\tau_2 \tag{2.10}$$

Definimos uma matriz densidade P com seus elementos dados por:

$$P_{\mu\nu} = 2\sum_{i}^{n} c_{\mu}^{(i)}c_{\nu}^{(i)} \tag{2.11}$$

Assim, a equação da energia (2.6) assume a seguinte forma:

$$E = \sum_{\mu\nu} P_{\mu\nu}h_{\mu\nu} + \frac{1}{2}\sum_{\mu\nu\lambda\sigma} P_{\mu\nu}P_{\lambda\sigma}\left((\mu\nu|\lambda\sigma) - \frac{1}{2}(\mu\sigma|\nu\lambda)\right) \tag{2.12}$$

2.2 Métodos semiempíricos

Métodos *ab initio* utilizam métodos numéricos para encontrar a solução da equação de Schrödinger, conseguindo, em geral, excelentes resultados, mas isso exige muito processamento computacional. Por isso, o uso prático dessas soluções atualmente é limitado à modelagem de, no máximo, algumas centenas de átomos [5].

Para modelagens contendo milhares ou milhões de átomos, a abordagem mais adequada são os métodos semiempíricos. Eles modificam o método de HFR, passando a usar um Hamiltoniano mais simples do que o de HFR exato e atribuindo valores empíricos para algumas das integrais, podendo ainda negligenciar algumas outras. Isso permite realizar os cálculos em sistemas moleculares de forma muito mais rápida do que os métodos *ab initio* [23, 24].

Com o surgimento do método semiempírico MNDO, foram introduzidos parâmetros adaptados para encontrar propriedades de interesse com exatidão química

[6]. Por exemplo, as integrais monoatômicas de repulsão eletrônica passaram a ser derivadas de dados experimentais, usando parâmetros que correspondem aos valores observados. Esse método serviu de base para o AM1 [7]. E o RM1, por sua vez, foi um método gerado pela reparametrização do AM1 [5].

A tabela 2.1 foi obtida a partir do manual do MOPAC (*Molecular Orbital PACkage*), que é um programa para cálculos de química quântica semiempírica [11]. A tabela apresenta uma visão geral dos parâmetros utilizados nos métodos semiempíricos MNDO, AM1, PM3 e PM6 [25]. O RM1 tem os mesmos parâmetros que o AM1, pois a estrutura algébrica das equações é a mesma, mudando apenas os seus valores.

Parâmetro	Descrição	Unidade	MNDO	AM1	PM3	PM6
U_{ss}, U_{pp}, U_{dd}	Integrais de um elétron e um centro, para os orbitais s, p, e d	eV	*	*	*	*
β_s, β_p, β_d	Termos da integral de ressonância de um elétron e dois centros, para os orbitais s, p, e d	eV	*	*	*	*
ξ_s, ξ_p, ξ_d	Expoente do orbital s, p, e d do tipo Slater	$bohr^{-1}$	*	*	*	*
ξ_{sn}, ξ_{pn}, ξ_{dn}	Expoente interno do orbital s, p, e d do tipo Slater	$bohr^{-1}$	*	*	*	*
α_A	Termo de repulsão núcleo-núcleo do átomo A	$Å^{-1}$	*	*	*	
α_{AB}	Termo de repulsão núcleo-núcleo dos átomos A e B	$Å^{-1}$				*
G_{ss}	Integral de repulsão de um centro e dois elétrons, para orbitais s-s	eV	+	+	*	*
G_{sp}	Integral de repulsão de um centro e dois elétrons, para orbitais s-p	eV	+	+	*	*
G_{pp}	Integral de repulsão de um centro e dois elétrons, para orbitais p-p	eV	+	+	*	*
G_{p2}	Integral de repulsão de um centro e dois elétrons, para orbitais p-p'	eV	+	+	*	*
H_{sp}	Integral de troca de um centro e dois elétrons, para orbitais s-p	eV	+	+	*	*
K_{nA} ou a_{nA}	Um multiplicador gaussiano para a enésima Gaussiana do átomo A	nenhuma		*	*	*
L_{nA} ou b_{nA}	Um multiplicador do expoente gaussiano para a Gaussiana do átomo A	$Å^{-2}$		*	*	*
M_{nA} ou c_{nA}	Um raio do centro da enésima Gaussiana do átomo A	Å		*	*	*

Tabela 2.1: Parâmetros usados em métodos semiempíricos. O símbolo * indica que o parâmetro foi otimizado para o método, e + significa que foi obtido a partir de experimentos (não otimizado). Fonte: Stewart, 2014b (adaptado).

Esses parâmetros substituem trechos das equações utilizadas no método de Hartree-Fock. Por exemplo, os parâmetros U_{ss}, U_{pp} e U_{dd} entram no lugar da equação 2.7.

2.2.1 Método AM1

O AM1 foi idealizado por Dewar e colaboradores na Universidade do Texas, em 1985, e trata apenas os elétrons de valência. Ele resolve equações que lembram as de Hartree-Fock para encontrar orbitais moleculares auto-consistentes, porém usa um Hamiltoniano aproximado e aproximações drásticas para muitas das integrais.[23]

O método recebeu o nome de *Austin Model 1* por ter sido desenvolvido na cidade de Austin, onde fica a Universidade do Texas. Ele é uma versão melhorada do MNDO, que era um dos métodos semiempíricos mais populares. MNDO é a sigla para *Modified Neglect of Diatomic Overlap* (Negligência Modificada de Sobreposição Diatômica). Nesse método, as integrais de elétrons de átomos diferentes foram substituídas por integrais aproximadas derivadas de interações multipolares, além de apresentar uma considerável melhoria no termo de repulsão núcleo-núcleo [5]. Porém, entre átomos separados por algo próximo de suas distâncias de van der Waals, as repulsões eram superestimadas.

Por isso, a principal diferença do AM1 para o MNDO é que as repulsões núcleo-núcleo foram modificadas, introduzindo funções gaussianas atrativas e repulsivas centradas em pontos internucleares [26]. No MNDO, a energia de repulsão núcleo-núcleo é calculada da seguinte forma:

$$E_{\text{núc}} = \sum_{i<j} E_N(i,j) \tag{2.13}$$

Onde:

$$E_N(i,j) = Z_A Z_B \,(AA|BB)\,(1 + e^{-\alpha_A R_{ij}} + e^{-\alpha_B R_{ij}}) \tag{2.14}$$

Em que o átomo i é do tipo A, o átomo j é do tipo B, $(AA|BB)$ é uma integral de dois centros do tipo $(ss|ss)$, Z_i é o número de elétrons de valência do átomo i, R_{ij} é a distância interatômica e α_A é um parâmetro otimizável. [27]. Essa fórmula é válida para interações diferentes de O-H ou N-H.

No AM1, a equação 2.14 foi modificada para:

$$\begin{aligned} E_N(i,j) = {} & Z_A Z_B \,(AA|BB)\,(1 + e^{-\alpha_A R_{ij}} + e^{-\alpha_B R_{ij}}) \\ & + \frac{Z_i Z_j}{R_{ij}} \left(\sum_k a_{kA}\, e^{-b_{kA}(R_{ij}-c_{kA})^2} + \sum_k a_{kB}\, e^{-b_{kB}(R_{ij}-c_{kB})^2} \right) \end{aligned} \tag{2.15}$$

Onde as quantidades a_{kA}, b_{kA} e c_{kA} são parâmetros otimizáveis.

Além dessa modificação, o método foi reparametrizado. Esses ajustes levaram o modelo a ser um dos métodos semi-empíricos de referência por muitos anos.

2.2.2 Método RM1

Analogamente ao AM1, o RM1 recebeu um nome que indica a cidade onde foi desenvolvido, por isso foi chamado de *Recife Model 1*. Publicado em 2006, esse método manteve a estrutura algébrica e o número de parâmetros do AM1, mas modificou os valores dos parâmetros para que reproduzissem melhores resultados.

O objetivo da manutenção do formato foi facilitar a implementação do método em *softwares* que já possuíssem o AM1 [8]. Esse método utiliza 191 parâmetros, contendo dados de átomos recorrentes em simulações de química orgânica (C, H, N e O), bioquímica (P e S) e farmácia (F, Cl, Br e I).

O cálculo dos parâmetros para o RM1 utilizou uma combinação das técnicas de Newton-Raphson e simplex para encontrar parâmetros otimizados. O resultado foi superior ao AM1 e a alguns outros métodos que surgiram posteriormente ao AM1, como o PM3 e o PM5, conforme pode ser observado nos gráficos abaixo, retirados do artigo de Rocha et al [10].

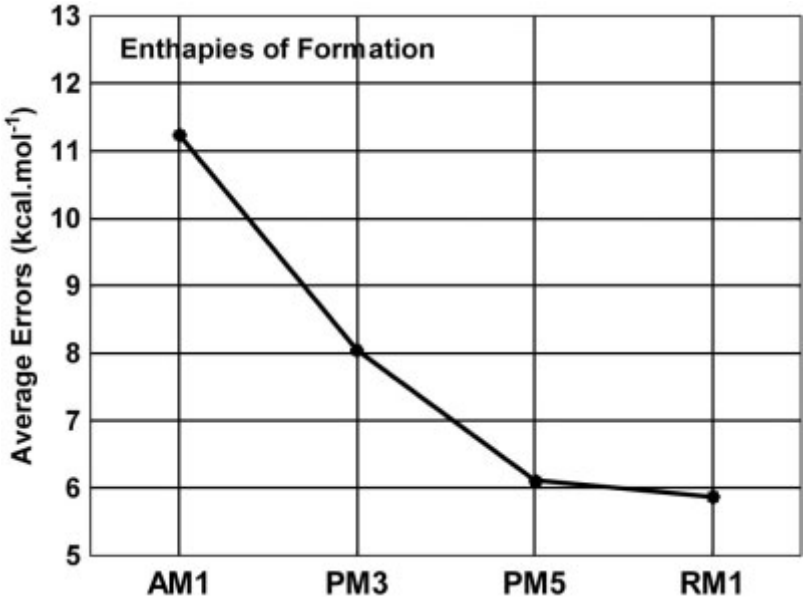


Figura 2.1: Comparativo dos erros médios das entalpias de formação calculadas para 1480 moléculas no AM1, PM3, PM5 e RM1. Fonte: Rocha et al, 2006, p. 1106.

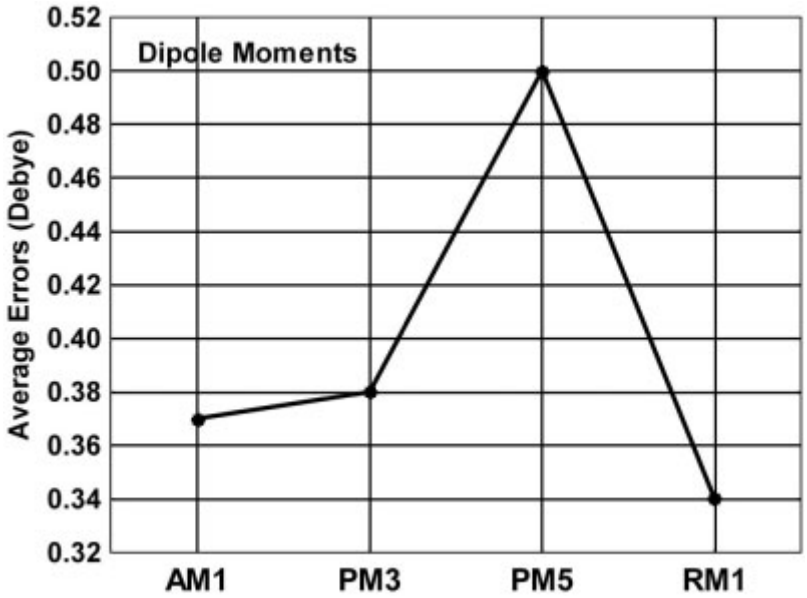


Figura 2.2: Comparativo dos erros médios dos momentos de dipolo calculados para 127 moléculas no AM1, PM3, PM5 e RM1. Fonte: Rocha et al, 2006, p. 1106.

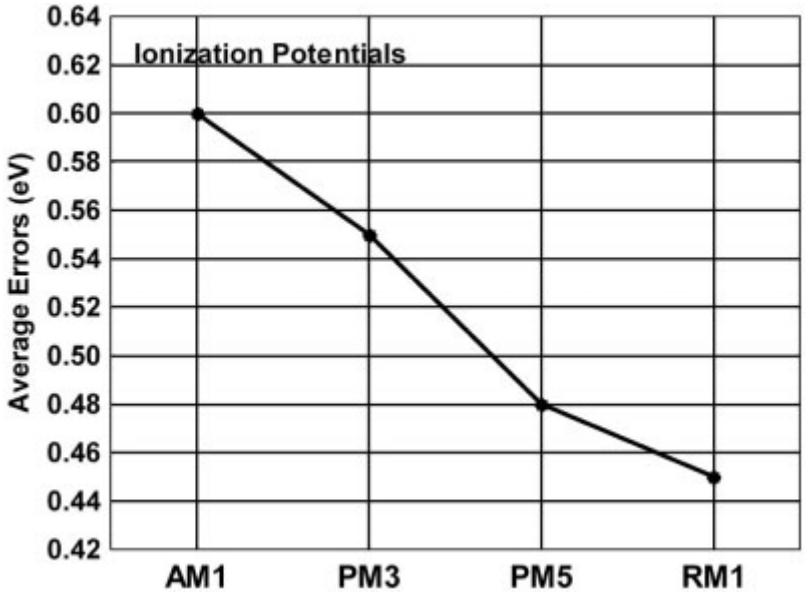


Figura 2.3: Comparativo dos erros médios dos potenciais de ionização calculados para 232 moléculas no AM1, PM3, PM5 e RM1. Fonte: Rocha et al, 2006, p. 1107.

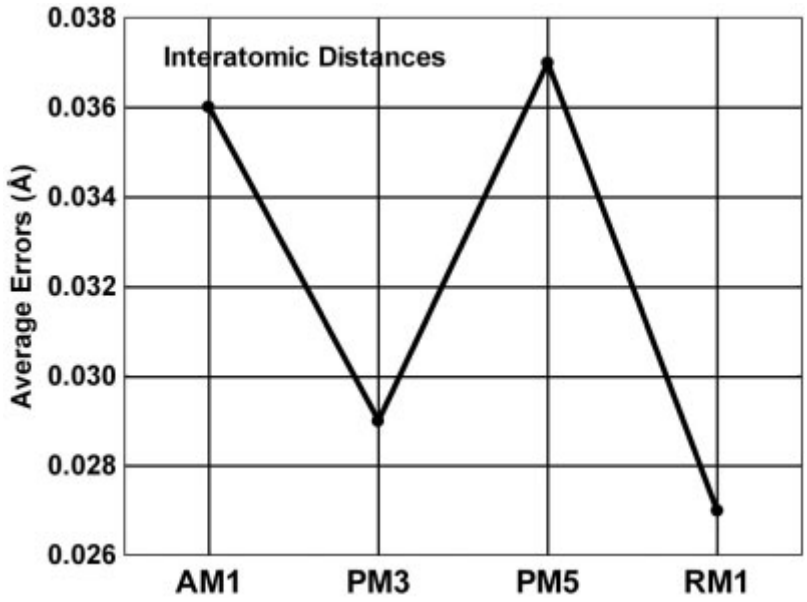


Figura 2.4: Comparativo dos erros médios de 904 distâncias interatômicas no AM1, PM3, PM5 e RM1. Fonte: Rocha et al, 2006, p. 1107.

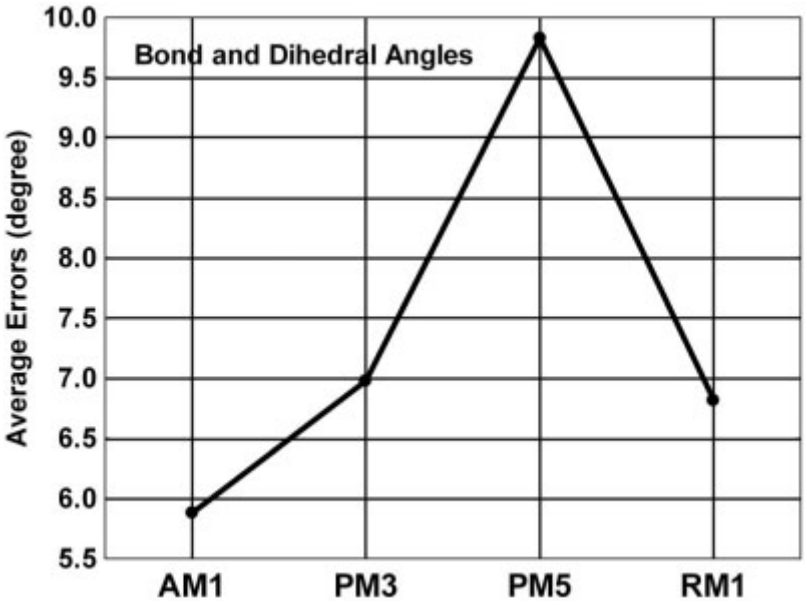


Figura 2.5: Comparativo dos erros médios de 910 ângulos de ligação e diédricos no AM1, PM3, PM5 e RM1. Fonte: Rocha et al, 2006, p. 1107.

2.2.3 Métodos semiempíricos mais atuais

Após o RM1, alguns novos métodos semiempíricos foram criados. Exemplos em destaque foram os métodos PM6 e PM7, evoluções do PM3, que, além refinarem suas modelagens, passaram a usar uma base de orbitais *spd*, o que permitiu cobrir quase toda a tabela periódica [28].

Indo além do modelo MNDO, surgiu a série OMx (OM1, OM2 e OM3), que também apresentou importantes melhorias nos dados calculados [28].

2.3 Otimização não-linear

Otimização é o processo de busca dos parâmetros de uma função $f(\mathbf{x})$ (em que $\mathbf{x} \in \mathbb{R}^n$) que produza um valor mínimo, local (em uma vizinhança de $\mathbb{R}^n$) ou global (em todo o $\mathbb{R}^n$). Opcionalmente, a busca pode ser pelo valor máximo, mas a estratégia é equivalente, pois o máximo de uma função $f(\mathbf{x})$ corresponde ao mínimo de $-f(\mathbf{x})$.

A otimização é dita linear quando a função tem a forma:

$$f(\mathbf{x}) = ax_1 + bx_2 + ... + kx_n \tag{2.16}$$

Como esse tipo de função corresponde a um hiperplano no espaço multidimensional, isso simplifica a busca pelo mínimo da função, havendo vários métodos para realizar essa busca.

Se $f(\mathbf{x})$ não tem a forma da equação 2.16, a função é não-linear. Para encontrar o mínimo de funções desse tipo, utilizam-se estratégias de otimização não-linear, que são muitas, a exemplo dos métodos do gradiente descendente e de Newton.

No caso deste trabalho, a otimização da função resposta foi realizada pelo PARAM, que é um programa criado para otimizar parâmetros para métodos semiempíricos, desenvolvido pelo Prof. James J. P. Stewart [8]. O método de otimização utilizado por esse programa é uma versão modificada do DFP [15] [29] [16]. Antes de explicar o seu funcionamento, é importante esclarecer os métodos do gradiente, de Newton e da secante, que servem de base para o DFP. Mas essas explicações só estão sendo expostas para que fiquem mais evidentes os conceitos relacionados ao DFP. Neste trabalho, apenas o DFP foi utilizado, por meio do PARAM.

2.3.1 Método do gradiente

O método do gradiente (ou de Cauchy) é um método iterativo que consiste em calcular, em determinado ponto pertencente à região do $\mathbb{R}^n$ de interesse, um vetor que aponta para a direção de máximo declive. O cálculo desse vetor utiliza as derivadas parciais em relação a cada uma das variáveis ou parâmetros, cujos valores compõem o vetor gradiente. Indo na direção negativa desse vetor, aproxima-se do mínimo local, o que é repetido até convergir para esse ponto.

A ideia principal do método é que, calculando a derivada parcial da função em relação a cada variável (ou uma aproximação), pode-se obter a direção em que essa variável muda mais rapidamente [30]. O vetor que obtém todas as derivadas parciais

é o gradiente e sua direção negativa é a direção d_k de descida utilizada no método:

$$d_k = -\nabla f(\mathbf{x}_k) \tag{2.17}$$

Onde $\mathbf{x} \in \mathbb{R}^n = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ e $\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix}$.

Portanto, em cada iteração, ocorre a seguinte atualização:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \omega_k \nabla f(\mathbf{x}_k) \tag{2.18}$$

Para a escolha do escalar $\omega_k > 0$, algumas das técnicas mais usadas são o método da seção áurea (que faz a busca exata do menor valor naquela direção, mas tem maior custo computacional) e a condição de Armijo (cuja busca é inexata, mas é menos custosa) [31].

Convergência do método

Seja $f(\mathbf{x})$ uma função diferenciável. Se $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, então $d_k = -\nabla f(\mathbf{x}_k)$ é uma direção de descida. Isso pode ser verificado observando que $\nabla f(\mathbf{x}_k)^T d_k < 0$, que é uma condição suficiente para essa direção ser de descida [31]. De fato:

$$\nabla f(\mathbf{x}_k)^T d_k = -\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0 \tag{2.19}$$

Como f é contínua e a sequência $(f(\mathbf{x}_k))_{k\in\mathbb{N}}$ é descrescente, caso ω_k seja escolhido pela busca exata, garante-se que $f(\mathbf{x}_k) \to f(\overline{\mathbf{x}})$, onde $\overline{\mathbf{x}}$ é um ponto de acumulação dessa sequência, correspondente a um mínimo local.

É possível provar que, calculando ω_k pela condição de Armijo, o método também coverge globalmente [31].

Quanto à velocidade de convergência, no caso de uma função quadrática, isto é:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \tag{2.20}$$

Em que $A \in \mathbb{R}^{n\times n}$ positiva definida, $\mathbf{b} \in \mathbb{R}^n$ e $c \in \mathbb{R}$. Essa função é convexa e

tem um único minimizador $\mathbf{x}^*$, onde:

$$A\mathbf{x}^* + \mathbf{b} = \nabla f(\mathbf{x}^*) = \mathbf{0} \tag{2.21}$$

Nesse caso, o algoritmo que usa busca exata converge linearmente para $\mathbf{x}^*$, com taxa de convergência $\sqrt{1 - \frac{\lambda_1}{\lambda_n}}$, onde λ_1 é o menor autovalor de A e λ_n é o maior [31].

2.3.2 Método de Newton

O método de Newton é um algoritmo iterativo que busca o mínimo de uma função a partir de um ponto inicial e do uso de uma função quadrática para aproximar o próximo ponto. Será apresentada inicialmente a explicação para o caso de funções de apenas uma variável, expandindo depois para o caso de múltiplas variáveis.

Considere uma função $f : \mathbb{R} \to \mathbb{R}$ em que se busca resolver o seguinte problema de otimização:

$$\min_{x \in \mathbb{R}} f(x) \tag{2.22}$$

Começando por um ponto inicial x_0, cada iteração irá calcular um novo x_k em que $f(x_k)$ deve se aproximar do mínimo da função. O método de Newton calcula o $x_{k+1} = x_k + t$ por meio da aproximação da função real por uma série de Taylor de segunda ordem, dada por:

$$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2 \tag{2.23}$$

Devemos procurar o valor de t que produza o mínimo dessa função, o que ocorre quando a derivada da função em relação a t é igual a zero (se a derivada segunda for positiva). Assim, temos:

$$\begin{aligned} &\frac{d}{dt}\left(f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2\right) = 0 \\ &f'(x_k) + f''(x_k)t = 0 \\ &t = -\frac{f'(x_k)}{f''(x_k)} \end{aligned} \tag{2.24}$$

Portanto, o termo seguinte do método de Newton é:

$$x_{k+1} = x_k + t = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{2.25}$$

Se a função for quadrática, a série de Taylor pode coincidir com a função e, consequentemente, o primeiro passo pode levar diretamente ao mínimo da função. No caso de funções de maior complexidade, mais passos são necessários, mas cada passo tende a se aproximar mais do mínimo.

Para funções multidimensionais ($\mathbf{x} \in \mathbb{R}^n$, $f(\mathbf{x}) \in \mathbb{R}$), a mesma ideia pode ser aplicada substituindo a derivada $f'(x_k)$ pelo gradiente $\nabla f(\mathbf{x}_k)$ e o inverso da segunda derivada pela inversa da matriz Hessiana $[\nabla^2 f(\mathbf{x}_k)]^{-1}$. Portanto, cada passo é calculado por:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k) \tag{2.26}$$

Onde o vetor $-[\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$ tem uma direção de descida se $\nabla^2 f(\mathbf{x}_k)$ for positiva definida [32].

Há 3 variantes para o algoritmo. Em uma delas, é utilizada exatamente a equação 2.26. Nas outras duas, o segundo termo é multiplicado por um passo ω_k que faz uso de busca (exata ou Armijo) [31].

No caso de uma função quadrática como a da equação 2.20, utilizar o passo padrão ($\omega_k = 1$) leva diretamente ao mínimo da função.

Convergência

O método de Newton não garante convergência, pois sua direção pode não ser de descida, caso a Hessiana não seja positiva definida no ponto observado. Porém, se for positiva definida, a direção de descida é garantida.

Se o palpite inicial está perto de um minimizador de f, onde a função é convexa, a convergência é muito rápida. No entanto, uma desvantagem é a necessidade de calcular a inversa da Hessiana em cada passo, o que tem alto custo computacional.

2.3.3 Método da secante

O método da secante é utilizado para encontrar a raiz de uma função de uma variável. Ele utiliza dois pontos de partida e é mais apropriado quando esses pontos se encontram próximos à raiz, assumindo que a função é aproximadamente linear na região de interesse [33].

O método funciona da seguinte forma: após definir os valores iniciais x_0 e x_1, é traçada uma reta secante à função, que passa por $(x_0, f(x_0))$ e $(x_1, f(x_1))$, encontrando o valor de x_2 que corresponde à raiz dessa reta. O procedimento é repetido considerando os dois últimos pontos, até encontrar um x_k tal que $f(x_k)$ esteja próximo de zero. Um exemplo pode ser visto na Figura 2.6 [33].

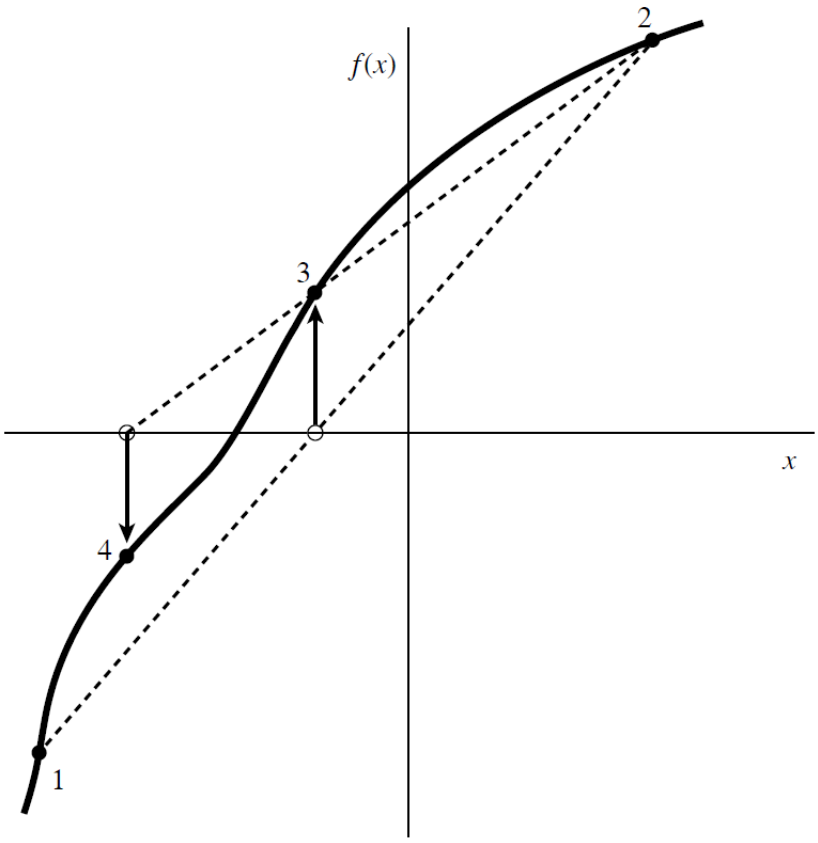


Figura 2.6: Exemplo de aplicação do método da secante. Fonte: Press et al, 1992, p. 348.

Para obter a fórmula utilizada nesse método, primeiro encontramos a reta que passa por $(x_{k-2}, f(x_{k-2}))$ e $(x_{k-1}, f(x_{k-1}))$. Na forma ponto-declividade, ela pode ser definida como:

$$\begin{aligned} \frac{y - f(x_{k-1})}{x - x_{k-1}} &= \frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}} \\ y &= \frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}}(x - x_{k-1}) + f(x_{k-1}) \end{aligned} \tag{2.27}$$

Em seguida, encontramos a raiz dessa reta estabelecendo que $y = 0$ e $x = x_k$:

$$\begin{aligned} \frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}}(x_k - x_{k-1}) + f(x_{k-1}) = 0 \\ x_k - x_{k-1} = -f(x_{k-1})\frac{x_{k-1} - x_{k-2}}{f(x_{k-1}) - f(x_{k-2})} \end{aligned} \tag{2.28}$$

Portanto, em cada iteração x_k é calculado como:

$$x_k = x_{k-1} - f(x_{k-1})\frac{x_{k-1} - x_{k-2}}{f(x_{k-1}) - f(x_{k-2})} \tag{2.29}$$

Convergência

Este método não garante convergência, pois as retas secantes podem escapar para posições cada vez mais distantes da raiz. Porém, caso x_0 e x_1 pertençam a um intervalo $[a, b]$ que atenda às condições abaixo, isso é suficiente para garantir convergência [34]:

- f seja uma função C^2 [a, b]
- $f(a)f(b) \leq 0$
- $f'(x) \neq 0 \; \forall x \in [a, b]$
- $f''(x) \geq 0$ ou $f''(x) \leq 0 \; \forall x \in [a, b]$
- Uma destas condições ser satisfeita:
  - $|f(a)/f'(a)| < |a - b|$ e $|f(b)/f'(b)| < |a - b|$ **ou**
  - $f(x_0)f''(x) \geq 0$ e $f(x_1)f''(x) \geq 0 \; \forall x \in [a, b]$

2.3.4 Método DFP

A utilização do método de Newton emprega muito esforço computacional, pois exige o cálculo da inversa da Hessiana $[\nabla^2 f(\mathbf{x}_k)]^{-1}$. Para diminuir esse tempo de processamento, foram desenvolvidos métodos quase-Newton (ou métodos de métrica variável) que passassem a calcular em cada iteração uma nova matriz H_k que se aproxima da inversa da Hessiana [33].

No método do gradiente, a direção de descida é dada por:

$$d_k = -\nabla f(\mathbf{x}_k) = -I \; \nabla f(\mathbf{x}_k) \tag{2.30}$$

Enquanto no método de Newton, temos:

$$d_k = -[\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k) \tag{2.31}$$

Por isso, métodos quase-Newton atualizam uma matriz H_k que tire proveito de ambos os métodos, definindo:

$$d_k = -H_k \nabla f(\mathbf{x}_k) \tag{2.32}$$

Onde H_k é uma matriz simétrica e positiva definida, o que faz d_k ser uma direção de descida [32]. Essa estratégia costuma ter um resultado melhor do que o método de Newton, que não garante uma direção de descida, especialmente quando se está distante do mínimo local, além de ter um custo computacional alto referente ao cálculo da inversa da Hessiana.

A atualização de H_k em cada iteração deve ocorrer de tal forma que:

$$\lim_{k \to \infty} H_k = [\nabla^2 f(\mathbf{x}_k)]^{-1} \tag{2.33}$$

Dessa forma, ocorre inicialmente uma descida (começando por $H_0 = I$ ou por alguma outra matriz positiva definida) e depois H_k se aproxima da inversa da Hessiana, o que agiliza a convergência para o mínimo, já que utiliza uma função de segunda ordem [33].

Um dos métodos quase-Newton mais conhecidos é o DFP. Nele, a atualização da matriz H_{k+1} é obtida por meio da condição quase-Newton (ou condição secante), baseada no método da secante [35], que determina a seguinte restrição:

$$\mathbf{x}_{k+1} - \mathbf{x}_k = H_{k+1}(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)) \tag{2.34}$$

Em funções de variável única, a solução para essa restrição é única em cada iteração (correspondendo ao método da secante). Para múltiplas variáveis, há várias soluções possíveis. A solução utilizada pelo DFP, que obedece a restrição e mantém

H_{k+1} positiva definida, é a seguinte:

$$H_{k+1} = H_k + \frac{(\mathbf{x}_{k+1} - \mathbf{x}_k) \otimes (\mathbf{x}_{k+1} - \mathbf{x}_k)}{(\mathbf{x}_{k+1} - \mathbf{x}_k) \cdot (\nabla f_{k+1} - \nabla f_k)} - \frac{[H_k(\nabla f_{k+1} - \nabla f_k)] \otimes [H_k(\nabla f_{k+1} - \nabla f_k)]}{(\nabla f_{k+1} - \nabla f_k) \cdot H_k(\nabla f_{k+1} - \nabla f_k)} \tag{2.35}$$

onde $\nabla f_j \equiv \nabla f(\mathbf{x}_j)$ e o operador $\otimes$ denota o produto tensorial entre dois vetores, que resulta em uma matriz tal que cada componente ij de $\mathbf{u} \otimes \mathbf{v}$ corresponde a $u_i v_j$ [33].

Definindo $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ e $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ e observando que $\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T$, essa fórmula também pode ser escrita da seguinte maneira [35]:

$$H_{k+1} = H_k + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k} - \frac{H_k \mathbf{y}_k \mathbf{y}_k^T H_k}{\mathbf{y}_k^T H_k \mathbf{y}_k} \tag{2.36}$$

Algumas propriedades importantes desse método são listadas a seguir [16]:
Para funções quadráticas (usando buscas exatas):

- Finalizam em no máximo n iterações, com $H_n = [\nabla^2 f(\mathbf{x_n})]^{-1}$;
- As condições quase-Newton anteriores são preservadas;
- Gera direções conjugadas e gradientes conjugados quando $H_0 = I$

Para funções genéricas:

- Preservam as matrizes H_k positivas definidas, mantendo a descida;
- Requer $3n^2 + O(n)$ multiplicações por iteração;
- Ordem de convergência superlinear;
- Convergência global para funções estritamente convexas (usando buscas exatas).

Adicionalmente, um grande benefício dos métodos quase-Newton é que precisam calcular somente a primeira derivada. Ou, indo além, podem ser aplicados a problemas sem derivada calculável, utilizando aproximações de diferença finita para estimar as primeiras derivadas usadas no vetor gradiente. Essa ideia foi tentada por Gilbert Stewart (1967), que sugeriu uma modificação do método DFP [36]. Essa foi a ideia implementada no código do PARAM.

Existem outros métodos que costumam ser um pouco mais eficientes do que o DFP, como por exemplo o BFGS [16], mas a adaptação do DFP utilizada no PARAM foi mantida porque já estava pronta para uso e fazia parte do procedimento que também aplica o método semiempírico e compara com os dados experimentais para obter a função objetivo a ser minimizada.

2.4 Algoritmos genéticos

Os conceitos de algoritmos genéticos apresentados nesta seção foram utilizados apenas no projeto de Maia et al (2019) [14], mas são importantes porque foi um projeto que serviu de base para o desenvolvimento do presente trabalho.

Um algoritmo genético (AG) é um método sem gradiente (que não utiliza o gradiente nem a Hessiana) [37] baseado na teoria da seleção natural de Charles Darwin.

A figura 2.7 é um fluxograma que representa o funcionamento de um AG padrão. Esse tipo de algoritmo funciona por um procedimento iterativo que evolui uma população de indivíduos, onde cada indivíduo representa uma possível solução para o problema em questão. Em cada iteração (chamada de geração), os indivíduos passam por uma função de aptidão (*fitness*) e são selecionados. Novos indivíduos são gerados com uma probabilidade de ocorrência de cruzamento ou mutação (chamados de operadores genéticos). O processo se repete até que uma condição de parada seja alcançada, que pode ser um número definido de gerações, uma detecção de convergência ou um tempo de execução atingido [38].

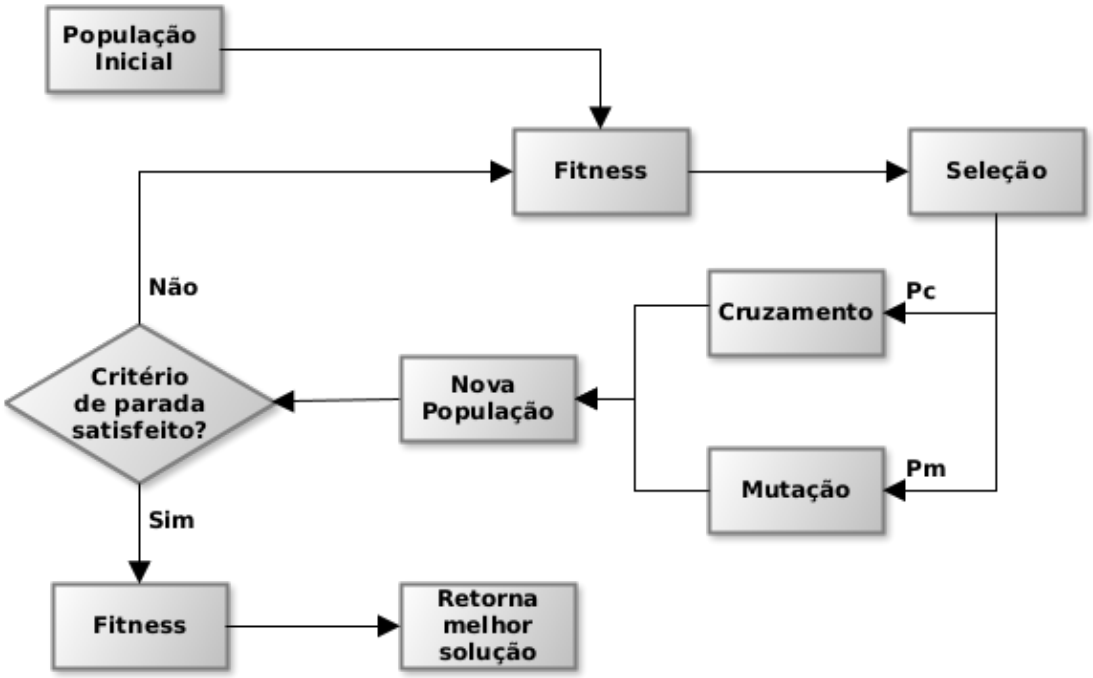


Figura 2.7: Fluxograma de um algoritmo genético padrão. Fonte: Kato et al, 2021.

Um indivíduo é uma simplificação de aspectos do mundo real. Apenas os dados importantes para a execução do algoritmo são considerados. No caso de uma reparametrização de métodos semiempíricos, um indivíduo pode ser definido simplesmente como um conjunto de parâmetros definidor do método.

A população é o conjunto de indivíduos utilizado no AG. Pode ser gerada aleatoriamente até que atinjam a quantidade definida, seguindo os limites estabelecidos, ou através de sementes (*seeds*), quando geralmente ocorre um pré-processamento de vários indivíduos aleatórios e os mais bem avaliados pela função *fitness* formam a população inicial [38].

A função *fitness* deve definir bem os requisitos que uma população deve alcançar para avançar para a geração seguinte, conforme as especificidades do problema. Dentre os que se enquadram nesses requisitos, é feita uma seleção que define quais deles gerarão descendentes.

2.4.1 Operadores genéticos

Após a seleção, são aplicados os operadores genéticos. Cada um deles tem uma probabilidade de ser utilizado. Os operadores mais conhecidos são os de cruzamento e mutação.

Operadores de cruzamento

Operadores de cruzamento ou recombinação (*crossover*) fazem a combinação entre dois indivíduos (pais) para gerar indivíduos descendentes (filhos) por meio da troca de partes aleatórias entre eles. Alguns dos principais tipos de cruzamento são o de ponto simples e o uniforme [38].

O cruzamento de ponto simples seleciona um ponto aleatório para corte, onde, a partir daquele ponto, os genes (informações que definem o indivíduo) serão trocados. Um exemplo pode ser visto na figura 2.8.

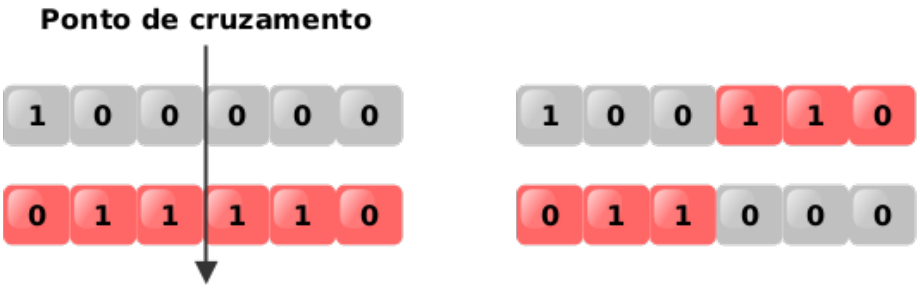


Figura 2.8: Exemplo de cruzamento de ponto simples. Fonte: Kato et al, 2021.

O cruzamento uniforme usa uma proporção fixa para determinar a contribuição de cada pai. Uma máscara aleatória de 0 e 1 é gerada respeitando essa proporção (por exemplo, se a taxa for 0,5, metade dos genes é herdada de cada um dos pais). Na figura 2.9, uma taxa de 0,5 foi utilizada, gerando uma máscara com metade de zeros (que selecionam os genes do pai 2) e metade de uns (para o pai 1) [38].

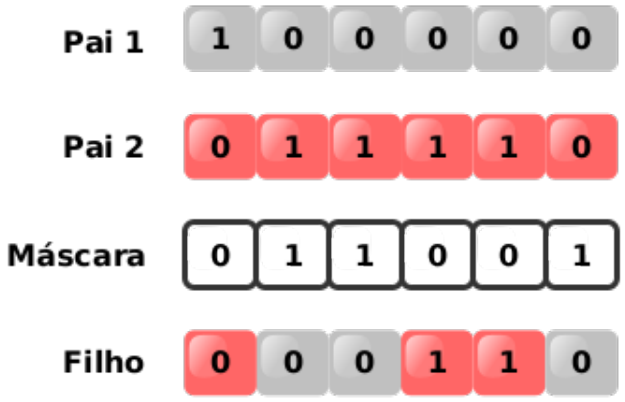


Figura 2.9: Exemplo de cruzamento uniforme. Fonte: Kato et al, 2021.

Operadores de mutação

Operadores de mutação alteram aleatoriamente algumas características genéticas dos indivíduos selecionados. A mutação utiliza apenas um pai para gerar o filho. Há vários tipos de mutação, como por exemplo a mutação uniforme [38].

Nesse tipo de mutação, ocorre a mudança de apenas um gene aleatório para um outro valor permitido para aquele mesmo gene. Na figura 2.10, a modificação foi de 0 para 1, mas essa mutação é mais utilizada quando há vários valores reais ou inteiros possíveis para o gene [38].

O novo valor é determinado por uma distribuição uniforme entre os limites mínimo e máximo permitidos. Também existe a mutação não-uniforme, em que o mesmo princípio é aplicado, mas o novo valor é extraído de uma distribuição não-uniforme [39].

Figura 2.10: Exemplo de mutação uniforme. Fonte: Kato et al, 2021.

Operadores de evolução diferencial

Outro tipo de operador é o de evolução diferencial. A principal diferença entre ele e os operadores descritos anteriormente é que ele utiliza informações de distância e direção para gerar um vetor unitário utilizado na reprodução [40].

Esse tipo de operador realiza um procedimento semelhante a uma mutação seguida de cruzamento, mas sem usar números aleatórios, e sim vetores. A geração do vetor unitário corresponde à mutação e envolve 3 indivíduos para determiná-lo. Esse vetor é cruzado com um quarto indivíduo, encerrando a operação com a seleção do mais apto entre o pai e o filho [40].

2.5 Estratégias de parametrização de métodos semiempíricos

Após desenvolver um novo método semiempírico, é necessário realizar a sua parametrização, isto é, definir o conjunto de parâmetros que ele irá utilizar.

O que se deseja com essa parametrização é que o método semiempírico seja capaz de predizer com boa exatidão as propriedades moleculares. Assim, a exatidão e o poder preditivo de métodos semiempíricos são determinados principalmente por três fatores: (1) os dados de propriedades que compõem o conjunto de referência ou treinamento, (2) a qualidade da sua parametrização e (3) o conjunto de aproximações que são usadas. Se os erros associados a estes três fatores são eliminados então teremos um método com alto poder preditivo. Dessa forma, estes três fatores constituem como etapas importantes na elaboração de um método semiempírico de alta qualidade.

Encontrar os parâmetros ótimos para um método semiempírico é, sem dúvida, a parte mais demorada do desenvolvimento desse tipo de método. Conforme visto na figura 1.2, esse procedimento exige que um vetor de parâmetros seja utilizado para calcular propriedades de uma grande quantidade de moléculas e compará-las aos dados experimentais, buscando minimizar o erro. Isso deve ser feito com cada vetor de parâmetros que se deseje verificar.

Só para dar uma ideia do desafio que está por trás da parametrização de um método semiempírico, vamos fazer um exercício mental. Tomando como exemplo o método RM1, ele possui 191 parâmetros para os átomos de H, C, N, O, F, P, S, Cl, Br e I. Por um momento, vamos imaginar que existem apenas dois valores para cada um desses parâmetros. Neste caso, haveriam 2^{191} ou da ordem de 3×10^{57} possibilidades de conjuntos de parâmetros, o que já é, em si, um número difícil de se visualizar de tão grande. Considerando o fato de que cada parâmetro é na realidade uma variável contínua, fica claro que é impossível varrer todo o espaço de parâmetros para se ter a certeza de que se obteve o melhor conjunto e, consequentemente, o melhor método possível.

Então, em geral se utiliza uma combinação de métodos de localização de pontos extremos de uma função resposta (também chamada de função custo). No trabalho do RM1 [10], foi utilizada uma combinação das técnicas Newton-Raphson e simplex, na tentativa de encontrar o melhor conjunto possível de parâmetros atômicos numa superfície de 191 dimensões que faça sentido químico. Para isto, tanto foi feita codificação, como também utilização de códigos computacionais que foram escritos para abordar problemas semelhantes em centenas de dimensões e com estabilidade numérica.

O RM1 foi publicado em 2006, e já alcançou um bom resultado com esse pro-

cedimento. Mas, conforme aumenta o poder de processamento dos computadores, torna-se possível realizar parametrizações ou reparametrizações mais robustas e massivas. Mais robustas, pois é possível utilizar a combinação de outras estratégias de otimização, refinamento e validação dos parâmetros que definem o método semiempírico. E massivos porque estão disponíveis computadores de alta performance, bem como códigos paralelizados que se aproveitam desses tipos de computadores.

Capítulo 3

Método Proposto

O método RM1 avançou muito, frente aos seus anteriores, e ainda se mostra competitivo em comparação aos mais modernos, pelo menos em se tratando de moléculas orgânicas. Mas será que, mantendo a mesma estrutura algébrica do RM1, podemos conseguir uma melhoria tal que defina um novo método semiempírico (chamado de RM2), apenas aplicando uma série de otimizações com diferentes abordagens? Melhor dizendo, será que os parâmetros atômicos que definem o método semiempírico RM1 são os melhores parâmetros, mantendo-se intacta a estrutura algébrica? Este trabalho veio para tentar responder isso.

A primeira abordagem para otimizar esses parâmetros foi um trabalho realizado nos anos de 2016 a 2019 por Maia et al (2019), em que foi realizada a primeira etapa do projeto "Buscando o mínimo global dos métodos quânticos semiempíricos NDDO (rm2)", em parceria com o Laboratório Nacional de Computação Científica (LNCC) [14]. Foi desenvolvido e aplicado um algoritmo genético que encontrou alguns pontos (conjuntos de parâmetros) com leve melhoria em relação ao RM1.

No presente trabalho, foi aplicada uma segunda abordagem, que serviu como continuação desse projeto. Foram utilizados 10 pontos obtidos por meio do AG como pontos de partida, para tentar encontrar pontos ainda melhores do que esses.

Como todo procedimento de parametrização, as etapas seguidas foram as seguintes:

1. Montagem do banco de dados de moléculas (conjunto de treinamento e de validação)

   (a) Felizmente, já possuímos esses bancos de dados.

   (b) Com o passar do tempo, o Prof. James Stewart, conseguiu compilar um banco de dados de mais de 10000 moléculas, onde várias propriedades experimentais e calculadas com métodos mais exatos estão disponíveis. Esse banco de dados foi altamente curado por vários anos.

(c) Neste projeto utilizamos esse banco de dados. Foi aplicado um particionamento e montado um conjunto de dados usado na fase de treinamento e o restante foi usado na fase de testes ou validação estatística.

2. Aplicação de técnicas de otimização não-linear

   (a) Da mesma forma que a parametrização do RM1, neste projeto, foram aplicadas técnicas de otimização não-linear de funções de muitos parâmetros.

   (b) Neste trabalho otimizamos os parâmetros atômicos dos átomos de C, H, O e N, que são os elementos de interesse na química orgânica. Caso o procedimento fosse bem sucedido, haveria uma segunda etapa em que iríamos fixar os parâmetros para esses átomos e parametrizar os átomos de P, S, F, Cl, Br e I. E, por fim, encontraríamos os parâmetros para os átomos de Se e Te.

   (c) Aqui cada ponto do $\mathbb{R}^n$ encontrado é o resultado do cálculo quântico semiempírico de milhares de moléculas. Ou seja, a obtenção de cada ponto tem um custo elevado.

   (d) Já possuímos um código de parametrização que foi cedido pelo prof. James Stewart (o PARAM). Ele utiliza uma modificação do método DFP para a otimização [15][27]. Por isso, não foi necessária a programação do algoritmo de otimização.

3. Validação estatística

   (a) Obtivemos vários conjuntos de parâmetros otimizados devido à presença de muitos mínimos locais na superfície multidimensional.

   (b) Os melhores valores encontrados foram avaliados por grupos de propriedades (relacionadas a calor de formação, momento dipolar, potencial de ionização e geometria), permitindo verificar qual a real contribuição de cada conjunto de parâmetros encontrado.

3.1 Definição do problema

O problema de otimização tratado tanto no trabalho de Maia et al (2019) quanto neste é uma reparametrização do RM1 que pode ser definida da seguinte forma:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{Sujeito a: limites mínimo} \\ &\text{e/ou máximo por parâmetro} \end{aligned} \qquad \text{onde } \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \tag{3.1}$$

Cada x_i é um parâmetro definido no RM1 para determinado elemento químico (veja a tabela 2.1). O RM1 tem 191 parâmetros definidos para 10 elementos químicos (H, C, N, O, F, P, S, Cl, Br e I) e buscou-se ainda acrescentar mais dois elementos (Te e Se), cada um destes com 18 parâmetros, totalizando 227 parâmetros. No trabalho de Maia et al (2019), todos esses parâmetros foram considerados. No trabalho presente, nos restringimos aos elementos H, C, N e O, com um total de 77 parâmetros, conforme explicado na seção 3.3.

Alguns desses parâmetros possuem restrições, como por exemplo: (i) os parâmetros U_{ss}, U_{pp} e U_{dd}, que são integrais de energia cinética, sempre assumem valores negativos; (ii) os parâmetros ξ_s, ξ_p, ξ_d, que são expoentes da função matemática que representa os orbitais atômicos, sempre serão quantidades negativas. Por isso, são determinados limites mínimo ou máximo para determinados parâmetros.

Os limites são tratados de forma diferente pelos dois métodos de otimização. Como não avalia a topologia da superfície da função custo através de derivadas, o algoritmo genético, simplesmente impõe valores elevados a função custo caso um dos limites máximo ou mínimo for extrapolado. Já o método DFP-modificado, implementado no programa PARAM, evita explorar regiões onde existam inconsistências físicas (como as relatadas no parágrafo anterior) impondo penalidades, caso o algoritmo sugira um passo além desses limites.

A função $f(\mathbf{x})$ é uma função resposta, F_{resp}, que corresponde a uma quantificação do erro de exatidão de um método semiempírico. Quanto maior o seu valor, mais o método erra na hora de prever os valores numéricos de certas propriedades moleculares consideradas na função.

A função F_{resp} é obtida a partir dos desvios das propriedades moleculares calculadas com determinados parâmetros, q^{calc}, comparadas com os dados de referência, q^{exp}, geralmente experimentais, de acordo com:

$$F_{resp} = \sum_j (q_j^{calc} - q_j^{exp})^2 \mathrm{w}_j^2 \tag{3.2}$$

Os dados de referência fazem parte de um banco de dados de propriedades moleculares e podem ter sido medidos experimentalmente ou calculados com métodos confiáveis, como alguns métodos *ab initio*.

Cada q_j representa o valor de uma propriedade diferente ou de uma molécula diferente. Isto é: para cada molécula, são obtidas as propriedades de interesse que estejam disponíveis para aquela molécula, e cada propriedade dessas é um novo q_j que entra no somatório, onde q_j^{exp} é obtido diretamente do banco de dados e q_j^{calc} é calculado pelo método semiempírico.

Apenas q_j^{calc} varia quando o vetor $\mathbf{x}$ é modificado. Cada q_j^{exp} se repete em cada ciclo da minimização, pois as moléculas obtidas do banco de dados são as mesmas em cada ciclo. Por isso, podemos dizer que F_{resp} é uma função de $\mathbf{x}$, e a equação 3.2 pode ser reescrita como:

$$f(\mathbf{x}) = \sum_j (q_j(\mathbf{x}) - q_j^{exp})^2 \mathrm{w}_j^2 \tag{3.3}$$

As propriedades moleculares que entram no somatório são: entalpia de formação (ΔH_f), que é o calor liberado ou absorvido na formação de 1 mol de uma substância a partir de substâncias simples, no estado padrão; potencial de ionização (PI), que é a energia aplicada para retirar um elétron do átomo (ou do íon) isolado no estado gasoso; momento dipolar (μ), que é o vetor que indica a polaridade da molécula, apontando para o sentido em que há maior eletronegatividade; distâncias e ângulos de ligação; e ângulos diedrais, que são os ângulos entre os planos determinados por dois conjuntos de 3 átomos, tendo 2 átomos em comum.

A utilização dos pesos, w_j, pode ser justificada pelo fato de estarmos somando diferenças de quantidades com unidades diferentes. A unidade de cada um desses pesos é o inverso da unidade da propriedade correspondente, de forma que o somatório tem uma grandeza adimensional. Pesos especiais podem ser atribuídos a moléculas especiais, como por exemplo: água, etanol, ácido acético, gliceraldeído, etc. Ou até mesmo para uma dada propriedade como, por exemplo, a distância de ligação entre os átomos de carbono do benzeno. O algoritmo genético utilizou pesos diferentes da otimização do presente trabalho, além de uma função resposta adaptada, conforme demonstrado nas seções 3.2 e 3.3.

O conjunto de parâmetros utilizados no RM1, os quais se tentou otimizar nesses dois trabalhos, são os seguintes para os elementos H, C, N e O [10]:

Parâmetro	H	C	N	O
U_{ss} (eV)	-11,96067697	-51,72556032	-70,85123715	-96,94948069
U_{pp} (eV)	—	-39,40728943	-57,97730920	-77,89092978
β_s (eV)	-5,76544469	-15,45932428	-20,87124548	-29,85101212
β_p (eV)	—	-8,23608638	-16,67171853	-29,15101314
α ($Å^{-1}$)	3,06835947	2,79282078	2,96422542	4,17196717
Gss(eV)	13,98321296	13,05312440	13,08736234	14,00242788
Gsp(eV)	—	11,33479389	13,21226834	14,95625043
Gpp(eV)	—	10,95113739	13,69924324	14,14515138
Gp2(eV)	—	9,72395099	11,94103953	12,70325497
Hsp(eV)	—	1,55215133	5,00000846	3,93217161
a_1	0,10288875	0,07462271	0,06073380	0,23093552
b_1	5,90172268	5,73921605	4,58892946	5,21828736
c_1	1,17501185	1,04396983	1,37873881	0,90363555
a_2	0,06457449	0,01177053	0,02438558	0,05859873
b_2	6,41785671	6,92401726	4,62730519	7,42932932
c_2	1,93844484	1,66159571	2,08370698	1,51754610
a_3	-0,03567387	0,03720662	-0,02283430	—
b_3	2,80473127	6,26158944	2,05274659	—
c_3	1,63655241	1,63158721	1,86763816	—
a_4	—	-0,00270657	—	—
b_4	—	9,00003735	—	—
c_4	—	2,79557901	—	—
ξ_s (au)	1,08267366	1,85018803	2,37447159	3,17936914
ξ_p (au)	—	1,76830093	1,97812569	2,55361907

Tabela 3.1: Parâmetros utilizados no RM1. Fonte: Rocha et al, 2006

3.2 Trabalho anterior - Aplicação de um algoritmo genético

A primeira abordagem de otimização, realizada por Maia et al (2019) anteriormente ao presente trabalho, utilizou um algoritmo genético de múltiplos mínimos, que é uma estratégia semelhante à utilizada por parte dos mesmos pesquisadores anteriormente [41]. Esse tipo de algoritmo genético pressupõe que a função tem muitos mínimos, buscando não se limitar a apenas uma região como ocorre em métodos de otimização não-linear tradicionais.

Inicialmente, foi escolhido o método de inicialização da população do algoritmo genético, de forma que os limites estabelecidos para os parâmetros fossem obedeci-

dos. Após alguns testes, foi definida uma inicialização em que cada indivíduo da população (descrito pelo conjunto de parâmetros correspondente) poderia ser gerado de duas formas (com 50% de chance para cada uma) [14]:

1. Seguindo uma distribuição de Cauchy no ponto central do intervalo entre os limites;

2. Seguindo uma distribuição de Cauchy no valor do RM1 para aquele parâmetro.

A figura 3.1 apresenta um exemplo de como esse tipo de inicialização da população funciona. Para uma população inicial de 20 indivíduos, são mostrados apenas 14 parâmetros (referentes ao elemento H). Cada símbolo "+" representa o valor daquele parâmetro para um indivíduo, que respeita os limites mínimo e máximo. O valor correspondente ao RM1 está representado por um "X" (apenas para informação, pois não faz parte da população) [14].

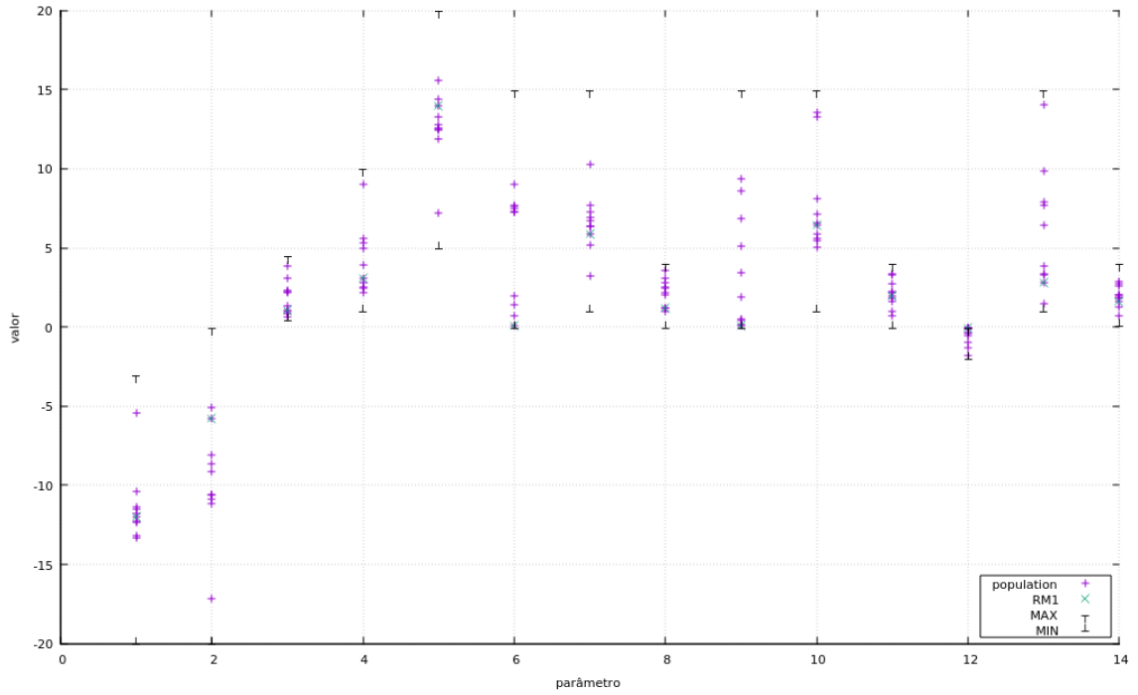


Figura 3.1: Panorama de uma população inicial de 20 indivíduos definidos por 14 parâmetros. Fonte: Maia et al, 2019

Foram definidos 50 indivíduos iniciais em duas ocasiões, com 227 parâmetros cada (191 já presentes no RM1 e 36 adicionais, referentes aos elementos Te e Se) e, para a exploração do espaço de soluções, foram empregados três operadores [14]:

1. Operador de Cruzamento

2. Operador de Mutação Não-uniforme

3. Operador de Evolução Diferencial

As probabilidades associadas a esses operadores foram 0,4, 0,4 e 0,2, respectivamente.

Dessa forma, cada nova combinação de parâmetros era avaliada, prevalecendo aquelas que alcançassem menor erro em relação aos dados experimentais, porém com indivíduos espalhados em diferentes regiões do $\mathbb{R}^n$.

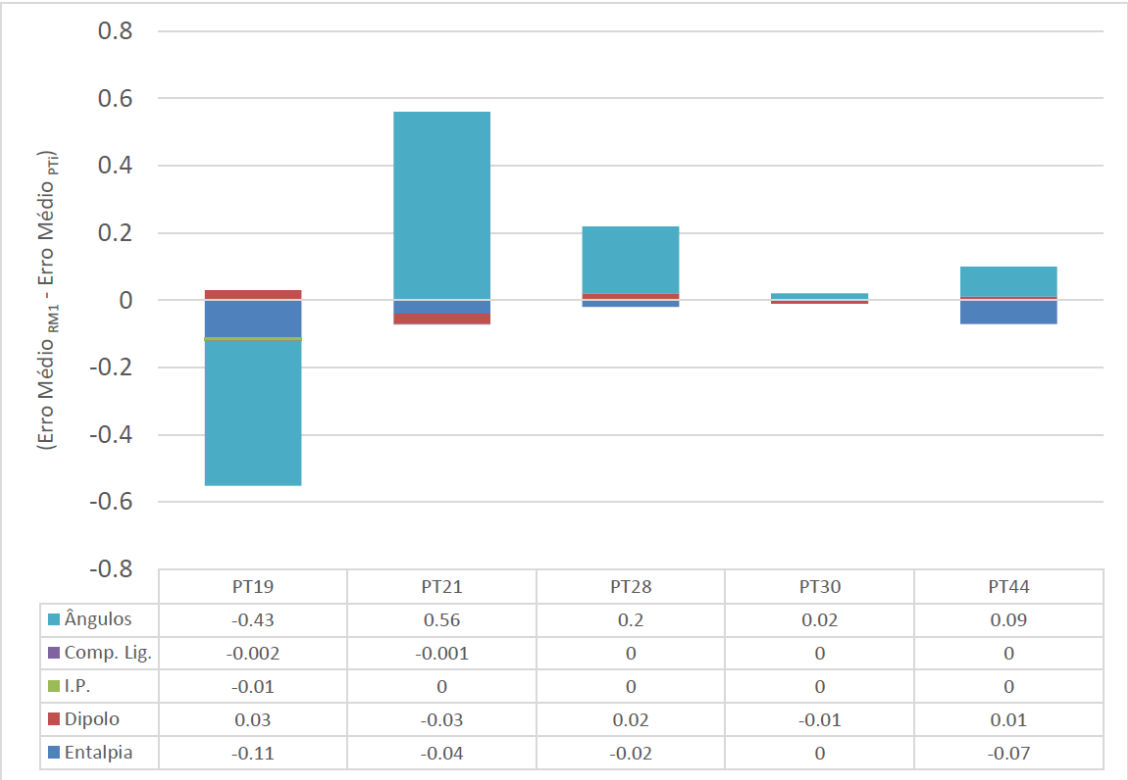
A abordagem utilizada para determinar a função custo foi diferente da do trabalho presente. Também foi utilizado o PARAM para verificar o erro, mas a função resposta calculada por ele foi ignorada. Em vez dela, foram utilizadas apenas as informações de erros médios por propriedade que o programa informa, e a função custo foi definida a partir deles.

Os pesos aplicados a cada um desses erros médios foi definido com base no valor dos erros médios das propriedades para o RM1. As propriedades consideradas, o erro médio associado a elas usando os parâmetros do RM1 (segundo o artigo que o apresentou [10]) e os pesos escolhidos podem ser vistos na tabela 3.2.

Propriedade	**Erro médio RM1**	**Peso AG**
Calor de formação	5,04 kcal $\cdot$ mol^{-1}	1,0 kcal^{-1} $\cdot$ mol
Potencial de ionização	0,42 eV	10,0 eV^{-1}
Momento de dipolo	0,23 D	10,0 D^{-1}
Ângulos de ligação	7,05 grau	1,0 grau^{-1}
Comprimentos de ligação	0,020 Å	100,0 Å^{-1}

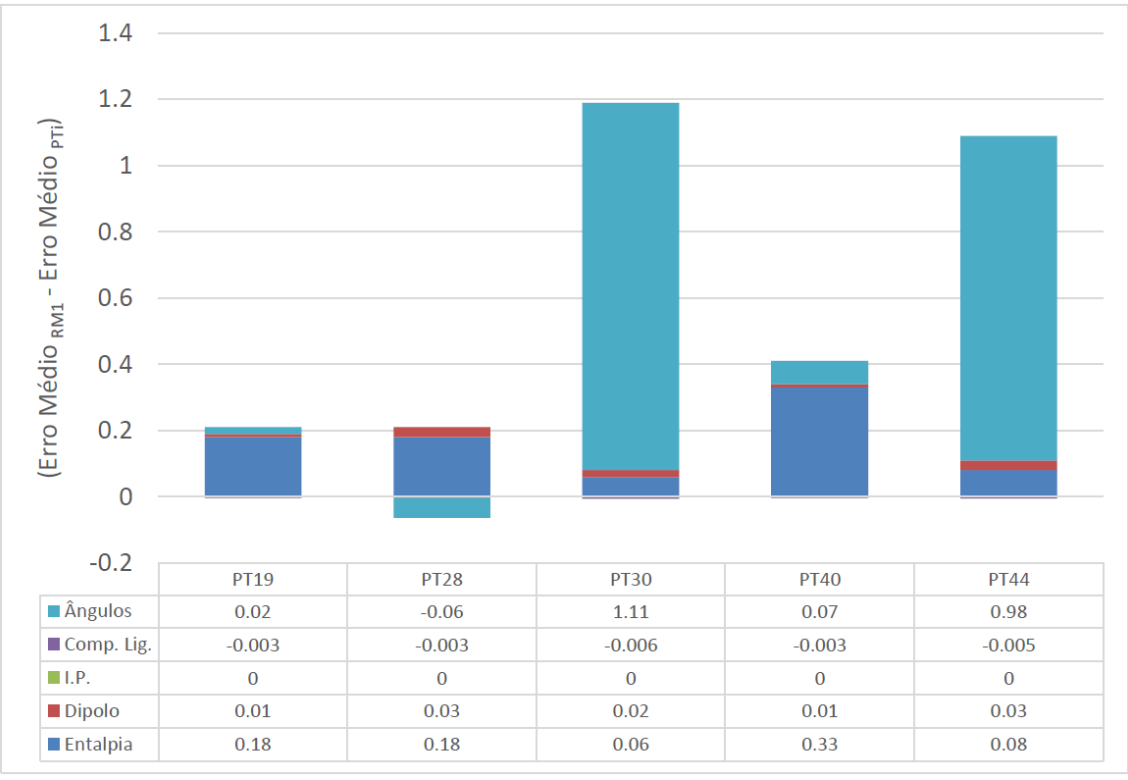
Tabela 3.2: Erros médios por propriedade no RM1 e pesos escolhidos para a função custo no AG. Fonte: Rocha et al, 2006; Maia et al, 2019

A execução desse algoritmo gerou 50 pontos em setembro e 50 em outubro de 2019, sendo selecionados os 10 melhores, que oferecem resultados semelhantes ou um pouco melhores que o RM1, conforme apresentado nos gráficos das figuras 3.2 e 3.3. Esses gráficos mostram a diferença entre o erro médio obtido no RM1 e com os pontos calculados pelo AG, agrupados pelo tipo de medida.



	PT19	PT21	PT28	PT30	PT44
Ângulos	-0.43	0.56	0.2	0.02	0.09
Comp. Lig.	-0.002	-0.001	0	0	0
I.P.	-0.01	0	0	0	0
Dipolo	0.03	-0.03	0.02	-0.01	0.01
Entalpia	-0.11	-0.04	-0.02	0	-0.07

Figura 3.2: Avaliação dos melhores pontos gerados pelo AG, usando 20000 avaliações da função objetivo (realizadas em setembro de 2019).



	PT19	PT28	PT30	PT40	PT44
Ângulos	0.02	-0.06	1.11	0.07	0.98
Comp. Lig.	-0.003	-0.003	-0.006	-0.003	-0.005
I.P.	0	0	0	0	0
Dipolo	0.01	0.03	0.02	0.01	0.03
Entalpia	0.18	0.18	0.06	0.33	0.08

Figura 3.3: Avaliação dos melhores pontos gerados pelo AG, usando 40000 avaliações da função objetivo (realizadas em outubro de 2019).

3.3 Estratégia de ação deste trabalho - Utilização do PARAM

Neste trabalho, os 10 pontos fornecidos pelo trabalho de Maia et al (2019) foram utilizados como pontos iniciais para serem otimizados com o programa PARAM.

O objetivo de utilizar um método quase-Newton (provido pelo PARAM) nos pontos encontrados pelo AG foi tentar aproveitar o melhor de cada técnica. Enquanto o AG de múltiplos mínimos teve êxito em explorar diferentes regiões do $\mathbb{R}^n$ e melhorar um pouco os resultados do RM1, os métodos quase-Newton são mais apropriados para descer mais rapidamente a um mínimo local. Assim, a expectativa era de encontrar 10 mínimos locais e escolher o melhor deles.

Foi utilizado o programa PARAM porque, além de estar disponível gratuitamente, ele tem o procedimento pronto para utilizar os parâmetros com determinados métodos semiempíricos e aplicá-los em um conjunto de modelos moleculares, obtendo o erros associados. Ele também realiza o procedimento completo de parametrização, calculando a função custo a partir desses erros e utilizando uma adaptação do método DFP para buscar o mínimo. Em cada ciclo, um novo conjunto de parâmetros é definido e a função custo é calculada novamente.

Os pesos utilizados para os erros por propriedade no cálculo da função custo, durante a otimização, foram os pesos padrão determinados pelo PARAM, que são os mesmos utilizados nas otimizações do MNDO e RM1 [10], com exceção do potencial de ionização e do momento de dipolo, cujos pesos foram reduzidos para a metade no código do programa [42]. A tabela 3.3 mostra os pesos.

Propriedade (q_i)	**Peso** (w_i)
Calor de formação	1 kcal^{-1} . mol
Potencial de ionização	5 eV^{-1}
Momento de dipolo	10 D^{-1}
Comprimento de ligação	100 Å^{-1}
Ângulo de ligação	2/3 grau^{-1}
Ângulo diédrico	1/3 grau^{-1}

Tabela 3.3: Pesos por propriedade utilizados na função custo calculada pelo PARAM.

3.3.1 Execução do PARAM

Para utilizar a funcionalidade de reparametrização provida pelo PARAM, foi preparado um arquivo de entrada para cada execução referente a um dos conjuntos de parâmetros iniciais, conforme o modelo abaixo:

```
am1 params=rm2_OUT19_params.19.txt ref=./inputs_rm2_param/new-chnops-sexte_2015-
    renamed deriv fine CYCLES=6000
 USS                            H
 BETAS                          H
 ZS                             H
 ALP                            H
 GSS                            H
 FN11                           H
 FN21                           H
 FN31                           H
 FN12                           H
 FN22                           H
 FN32                           H
 FN13                           H
 FN23                           H
 FN33                           H
 USS                            C
 UPP                            C
  .
  .
  .
 FN31                           O
 FN12                           O
 FN22                           O
 FN32                           O
end
Hydrogen,-cation.mop
Hydrogen,-atom.mop
h2r.mop
Hydrogen.mop
Carbon,-cation.mop
Carbon,-1D(g)-2s(2)2p(2).mop
  .
  .
  .
```

Arquivo: `rm2_2020_CHNO_running_OUT19_19.dat`

A primeira linha contém:

- Inicialmente, o método semiempírico a ser utilizado com os parâmetros dados. Foi escolhido o AM1, porque a estrutura dele é idêntica à do RM1, mudando apenas os parâmetros iniciais.
- Palavra-chave *params*: informa o nome do arquivo que contém os parâmetros iniciais, a serem otimizados. Nesse exemplo, o arquivo é o que contém os parâmetros do ponto 19 calculado pelo AG na execução de outubro de 2019.
- Palavra-chave *ref*: informa o caminho para a pasta onde estão os arquivos do banco de dados de referência.
- Palavra-chave *deriv*: solicita que o programa inclua no arquivo de saída informações sobre as derivadas calculadas.
- Palavra-chave *fine*: solicita que o programa calcule as derivadas por dois lados do ponto anterior, em vez de apenas um lado, como ocorre por padrão,

refinando o cálculo dessa derivada.

- Palavra-chave *CYCLES*: informa o número máximo de ciclos a ser executado. O padrão são 600 ciclos. Foi aumentado para 6000 para que a otimização durasse todo o tempo de execução disponível, aumentando as chances de encontrar um ponto com função custo menor.

A partir da segunda linha, são informados quais os parâmetros a ser otimizados, até que ocorra uma linha com a palavra "*end*".

E em seguida, até o final do arquivo, são informados os nomes dos arquivos do banco de dados de moléculas a serem utilizados (que ficam na pasta informada em *ref*). Esses arquivos possuem extensão `.mop`, pois foram gerados com o programa MOPAC [11].

Quanto aos arquivos com os parâmetros iniciais, um exemplo pode ser visto abaixo:

```
USS      H   −11.9606770
BETAS    H    −5.7654447
ZS       H     1.0826737
ALP      H     3.0683595
GSS      H    13.9832130
FN11     H     0.1028888
FN21     H     5.6060689
FN31     H     1.0673203
FN12     H     0.0645745
FN22     H     6.6097775
FN32     H     1.9384448
FN13     H    −0.0356739
FN23     H     2.8047313
FN33     H     1.5268554
USS      C   −51.7255603
UPP      C   −39.4072894
BETAS    C   −15.4593243
BETAP    C    −8.2360864
⋮
```

Arquivo: `rm2_OUT19_params.19.txt`

O arquivo possui apenas os valores de cada parâmetro, que são lidos pelo PARAM apenas no inicio da execução. Foram usados 10 arquivos como esse, um para cada conjunto de parâmetros calculado pelo AG e utilizado como ponto inicial neste trabalho.

Instruções mais detalhadas sobre como utilizar o PARAM podem ser encontradas no guia disponível no Apêndice A.

A figura 3.4 mostra as etapas seguidas em cada uma das 10 otimizações executadas no PARAM.

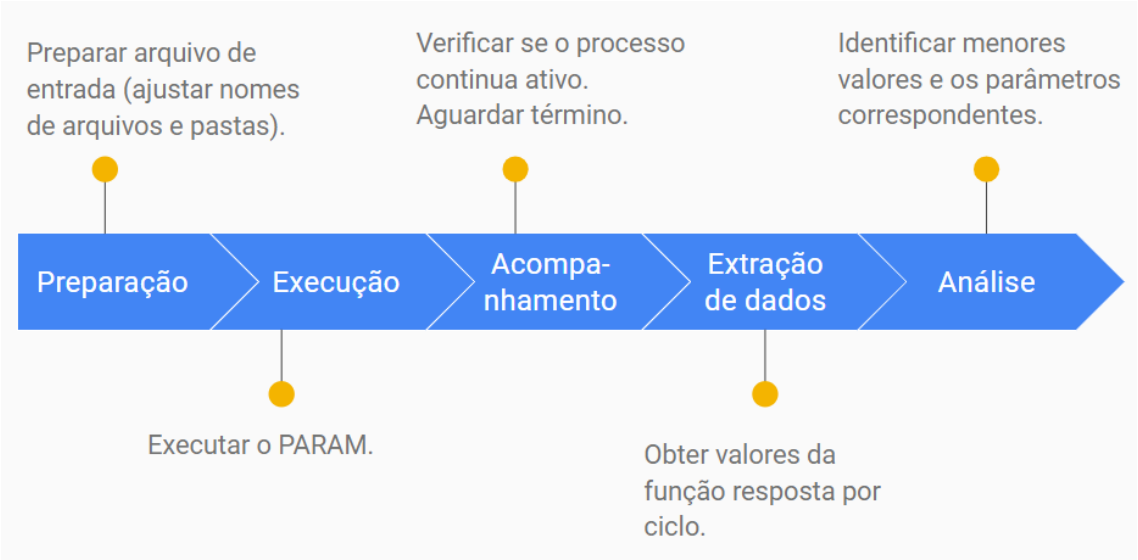


Figura 3.4: Atividades executadas em cada execução do PARAM.

O PARAM foi executado no supercomputador Santos Dumont (SDumont), localizado na sede do LNCC, em Petrópolis-RJ [43]. Um guia para uso dessa máquina está disponível no Apêndice B.

Foram submetidos *jobs* (rotinas) para serem executados durante o maior tempo possível permitido a um *job*: 31 dias (utilizando a fila chamada *cpu_long*). Foram solicitados 10 *jobs*, um para cada conjunto de dados obtidos previamente no trabalho mencionado na seção 3.2, que também foi realizado utilizando esse supercomputador.

Após o término da execução dos *jobs*, os resultados foram avaliados, inicialmente, por meio apenas do resultado da função custo. A partir do arquivo de saída gerado em cada *job*, foram extraídos os parâmetros correspondentes ao menor valor encontrado para essa função.

Em seguida, esses parâmetros foram submetidos a uma validação estatística, na qual também foram verificados os erros médios por propriedade. Essa etapa envolveu novas execuções no PARAM, mas essas foram mais rápidas por não envolverem otimização, por isso foram realizadas em uma máquina do Laboratório de Química Quântica Computacional (LQQC), que pertence ao Departamento de Química da Universidade Federal da Paraíba (DQ-UFPB) [44].

Capítulo 4

Resultados e Discussões

A execução no supercomputador SDumont foi feita utilizando apenas 1 nó e 1 tarefa por *job*, por isso o tempo de execução foi elevado. Também foram feitos testes utilizando mais nós ou tarefas, mas não se observou ganho de performance, o que indicou que a otimização do PARAM não está adaptada para utilizar o paralelismo ou que estava faltando alguma configuração para que a paralelização funcionasse de forma eficiente. No entanto, a análise dos resultados deixou claro que dificilmente seriam encontrados resultados melhores do que os obtidos ao longo dos 31 dias de processamento de cada *job*, mesmo se fosse encontrada uma forma de agilizar a execução do procedimento.

Devido à instabilidade da função custo avaliada na otimização, que converte 77 parâmetros em um somatório de erros quadrados ponderados, que por sua vez foram calculados numa comparação entre os dados de referência e os dos modelos obtidos usando esses parâmetros no método semiempírico, e também por ter derivadas calculadas por aproximação, nem sempre o valor calculado pela adaptação do DFP em determinado ciclo era menor do que o calculado no ciclo anterior.

O comportamento, utilizando cada um dos 10 dados iniciais, sempre foi o de diminuir até certo ponto e, aos poucos, escapar para um somatório de valor maior e ficar preso em valores maiores por muitos ciclos seguintes, incluindo alguns aumentos, de forma que o mínimo valor encontrado em cada cálculo não foi o último calculado. No entanto, houve uma diminuição considerável do valor da função. Comparados aos valores iniciais da função, esses aumentos não foram tão expressivos.

Após a finalização de cada *job*, foram obtidos os parâmetros correspondentes ao menor valor encontrado para a função resposta, por meio do procedimento descrito na seção A.5 do Apêndice A.

A tabela 4.1 mostra o total de ciclos (isto é, o número de avaliações da função custo) para cada ponto otimizado e qual a numeração do ciclo que encontrou a menor função resposta.

	Total de ciclos	Ciclo com menor função custo
OUT19_19:	1310	436
OUT19_28:	2635	1114
OUT19_30:	2507	1168
OUT19_40:	2715	449
OUT19_48:	2654	1887
SET19_19:	2576	192
SET19_21:	2646	288
SET19_28:	2191	214
SET19_30:	2608	2070
SET19_40:	2563	1848

Tabela 4.1: Total de ciclos por *job* e numeração do ciclo com menor função custo.

O tempo total de execução foram 31 dias para cada um dos pontos, exceto para o ponto OUT19_19, cuja execução durou 15 dias e meio. Como esse foi o primeiro *job* executado, somente depois de iniciado foi percebido que o tempo limite de execução estava automaticamente definido como metade do máximo permitido para a fila, que é o comportamento padrão quando o tempo de execução não é especificado, conforme explicado no manual de uso do supercomputador SDumont [45]. Após perceber isso, os demais *jobs* foram configurados para o tempo máximo permitido (31 dias).

Como a utilização dos recursos do supercomputador exige a espera do início da execução em uma fila, e a fila com tempo máximo (*cpu_long*) demorava dias para começar a execução de cada *job*, além do fato de que no máximo 2 *jobs* eram executados simultaneamente por um mesmo usuário, o tempo total para executar todos os *jobs* foi muito alto. Por isso não houve tempo hábil para executar o primeiro *job* também por 31 dias. No entanto, é possível que, assim como ocorreu com a maioria dos *jobs*, a função custo não fosse melhorada após 1200 ciclos.

A otimização alcançou uma boa redução da função custo. O valor inicial sempre estava na casa dos milhões, conforme gráfico da figura 4.1, e foi reduzido para abaixo de 300.000. Por causa disso, apesar da demora na execução, o quadro era animador. Esperava-se que ao final os parâmetros resultassem em uma redução expressiva do erro por propriedade.

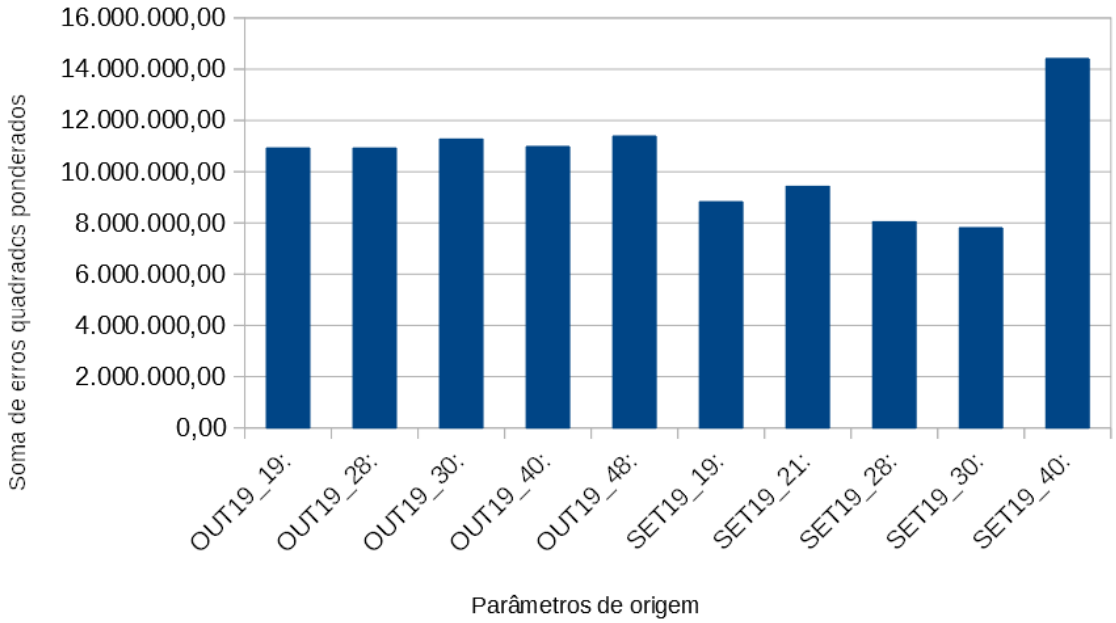


Figura 4.1: Valores iniciais da função custo durante a otimização.

O gráfico da figura 4.2 mostra qual o menor valor encontrado e qual o último, para cada *job* executado. Nota-se que o valor final nunca corresponde ao menor.

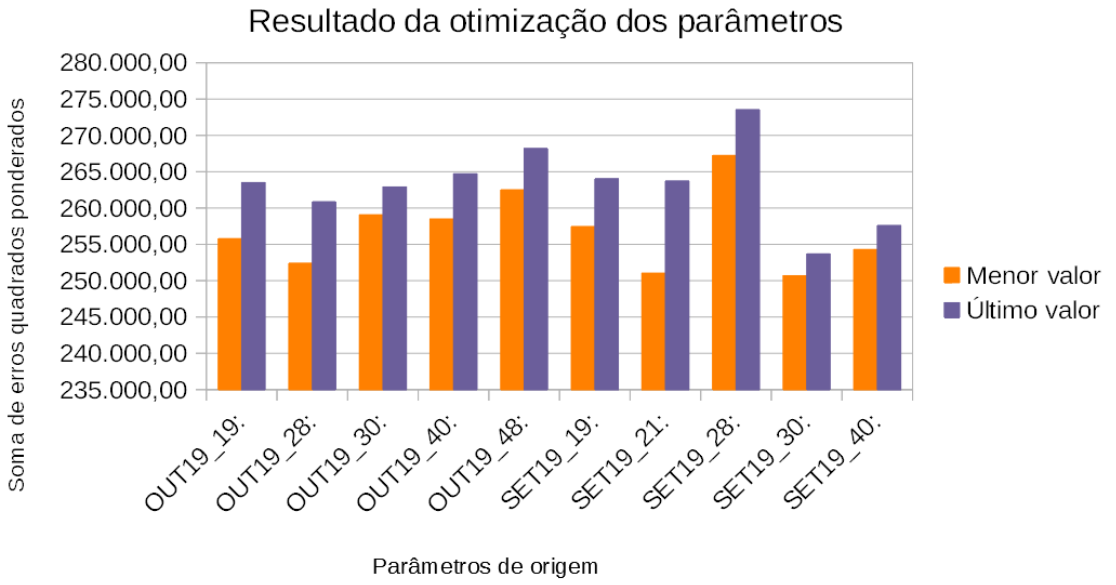


Figura 4.2: Menores e últimos valores obtidos durante a otimização.

O menor valor encontrado foi o que usou como dados iniciais o ponto PT30, de setembro de 2019. A soma de erros quadrados ponderados obtida foi 250.686,08. Aparentemente, era um bom valor, considerando que o valor inicial era de alguns milhões. Também de setembro, o PT21 teve uma otimização que alcançou o segundo menor valor: 251.043,54. E o terceiro menor veio do PT28 de outubro: 252.418,65.

Vistos isoladamente, esses valores podem parecer altos, já que estão longe do zero, que seria o erro ideal. Mas o motivo de tamanhos valores é simples: o erro médio associado ao RM1 está sempre na casa das unidades depois que cada propriedade é multiplicada pelo peso. Quando esse produto é elevado ao quadrado, passam à casa das dezenas. Somando os valores de todas as propriedades para uma molécula, chega às centenas, Considerando a soma para milhares de moléculas do banco de dados, facilmente chega às centenas de milhares. Portanto, o importante não é esse valor isolado, mas sim que essa soma seja minimizada.

Na figura 4.3, observa-se o comportamento nas proximidades do mínimo. Um súbito aumento de valor não permite que um valor menor seja alcançado posteriormente, ao longo de muitos ciclos. Este exemplo é do ponto OUT19_19, mas o comportamento é similar em todos os demais casos.

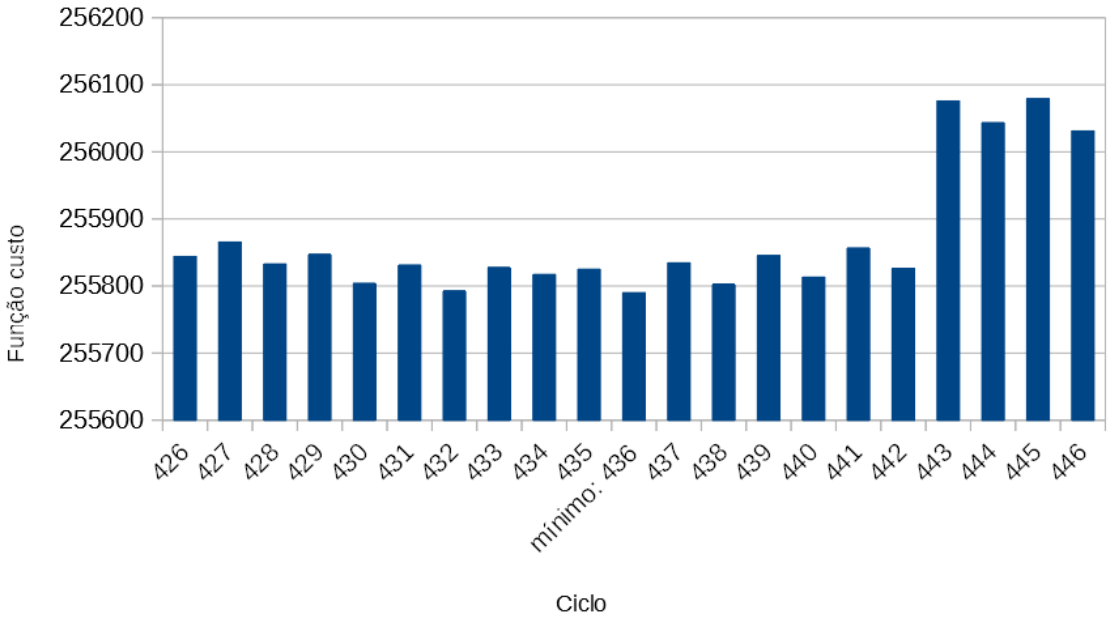


Figura 4.3: Comportamento da função custo nas proximidades do mínimo encontrado, na otimização do ponto OUT19_19.

Esses valores foram encontrados com um conjunto de dados de referência, utilizado para treinamento. Um conjunto de dados diferentes, mas igualmente válidos, foi utilizado para testes, para verificar a adequação do modelo a moléculas diferentes das de treinamento.

Assim, uma validação estatística foi realizada, utilizando o procedimento descrito na seção A.6 do Apêndice A. Inicialmente, foi verificado que a função custo continuou calculando um valor bem menor do que o calculado com os parâmetros do algoritmo genético, conforme pode ser observado na figura 4.4.

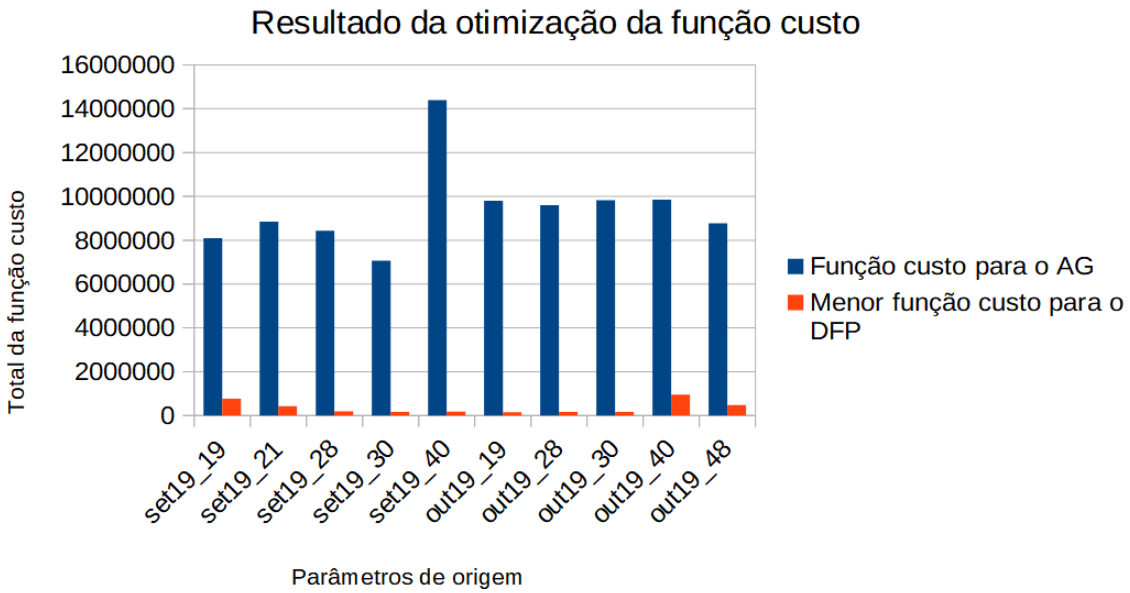


Figura 4.4: Valor inicial e otimizado da função custo, para cada ponto utilizado, considerando os dados de testes para validação estatística.

Comparando o menor custo inicial (calculado pelo AG) e o menor custo calculado pelo DFP com os custos calculados pelos métodos AM1, PM3 e RM1 utilizando as mesmas moléculas, observa-se a melhoria nessa função, conforme gráfico da figura 4.5.

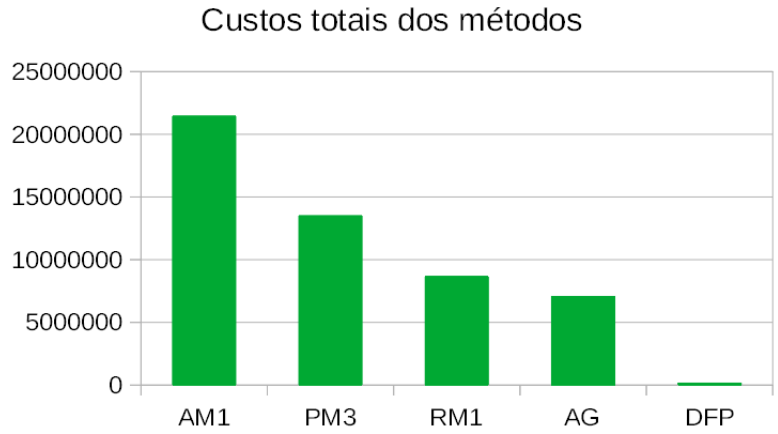


Figura 4.5: Custos totais calculados por método, na fase de testes.

Porém, ao avaliar o erro médio por propriedade (agrupando os valores analisados em 5 grupos: entalpia de formação, momento de dipolo, potencial de ionização (PI), comprimento de ligação e ângulos), foi observado um aumento do erro em quase todas as propriedades (com exceção do comprimento de ligação).

É possível que a função custo não estivesse adequada para refletir a qualidade das propriedades. De fato, é desafiador refletir, em uma única função, propriedades tão heterogêneas.

A comparação pode ser observada na tabela 4.2, que confronta as médias obtidas por propriedade, usando os métodos AM1, PM3 e RM1 e usando as parametrizações fornecidas pelos 10 pontos encontrados pelo AG e pela aplicação do DFP a cada um desses pontos.

Método	**Delta Hf**	**Dipolo**	**PI**	**Comp. Lig.**	**Ângulos**
AM1	10,64	0,51	0,48	0,032	7,28
PM3	6,79	0,68	0,55	0,023	7,44
RM1	5,78	0,52	0,42	0,024	7,48
AG_O19	5,61	0,51	0,42	0,027	7,46
AG_O28	5,6	0,49	0,42	0,028	7,48
AG_O30	5,72	0,5	0,42	0,03	6,54
AG_O40	5,46	0,51	0,42	0,027	7,41
AG_O48	6,71	0,58	0,45	0,033	8,03
AG_S19	5,89	0,49	0,43	0,026	7,92
AG_S21	5,82	0,53	0,42	0,025	6,96
AG_S28	5,8	0,5	0,42	0,024	7,31
AG_S30	5,79	0,52	0,42	0,024	7,5
AG_S40	14,01	0,51	0,42	0,024	6,5
DFP_O19	7,01	0,67	0,84	0,022	9,08
DFP_O28	7,17	0,7	1,09	0,022	7,93
DFP_O30	7,03	0,67	1,09	0,021	7,92
DFP_O40	6,77	0,65	0,84	0,021	8,59
DFP_O48	7,43	0,62	1,04	0,02	8,68
DFP_S19	6,87	0,63	0,67	0,022	8,69
DFP_S21	6,89	0,59	0,76	0,022	8,66
DFP_S28	6,93	0,58	0,71	0,022	8,66
DFP_S30	7,36	0,74	1,17	0,024	7,16
DFP_S40	7,53	0,68	1,17	0,02	7,33

Tabela 4.2: Erros das propriedades por método e por conjunto de parâmetros utilizados.

Os gráficos abaixo apresentam esses dados para cada propriedade individualmente:

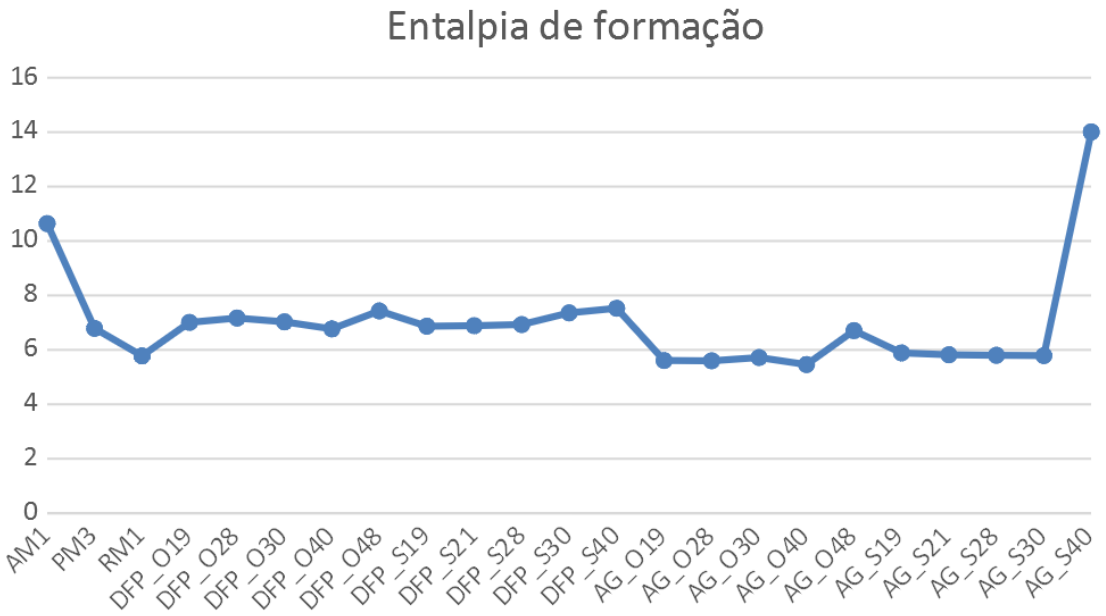


Figura 4.6: Entalpias de formação médias calculadas.

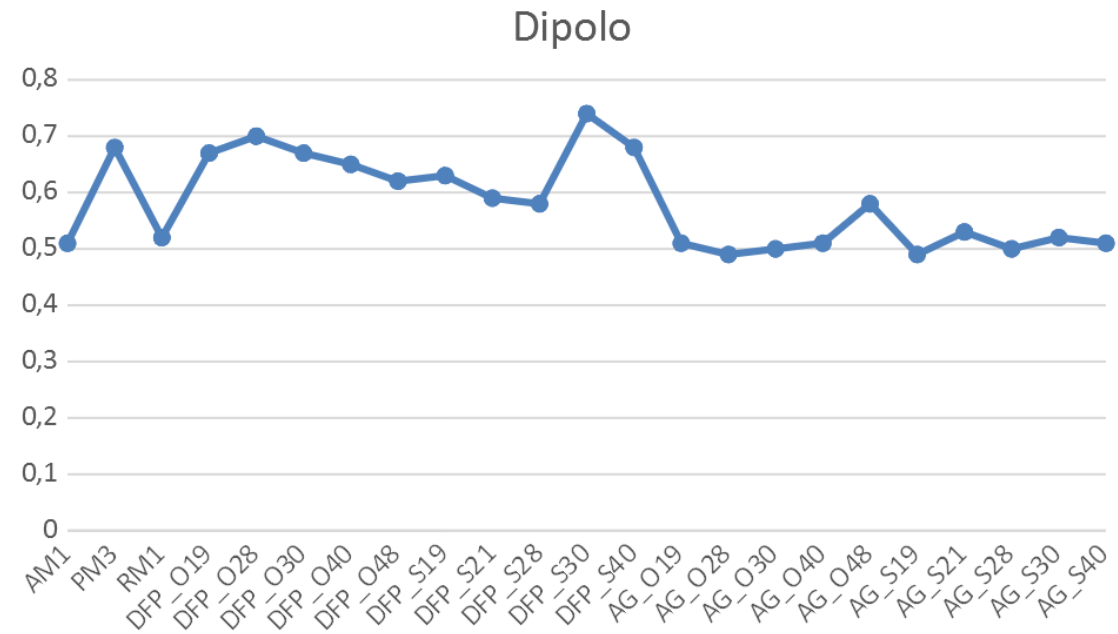


Figura 4.7: Momentos de dipolo médios calculados.

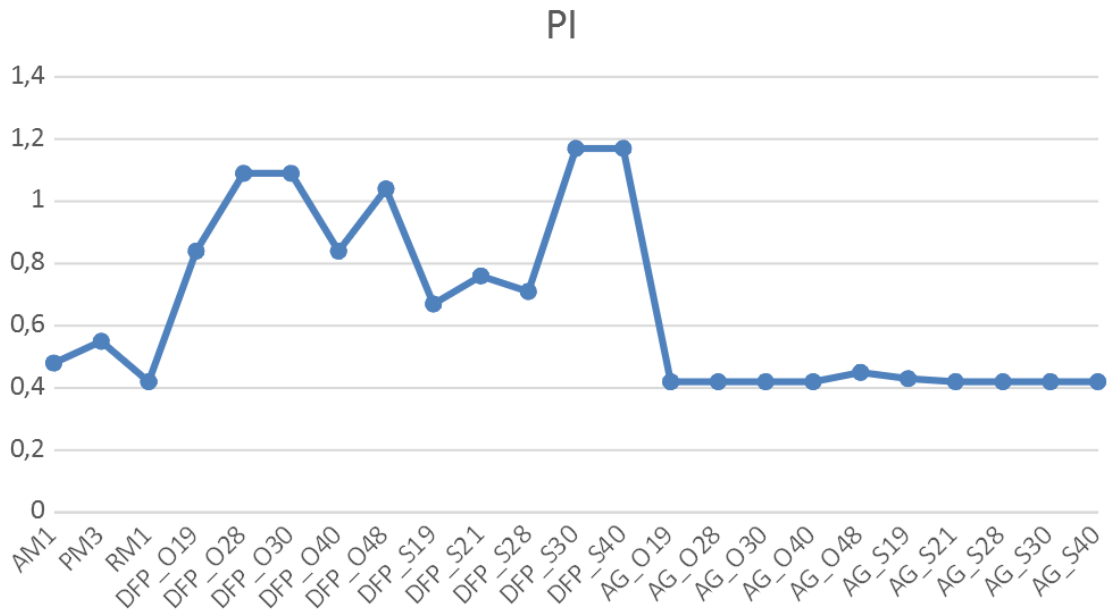


Figura 4.8: Potenciais de ionização médios calculados.

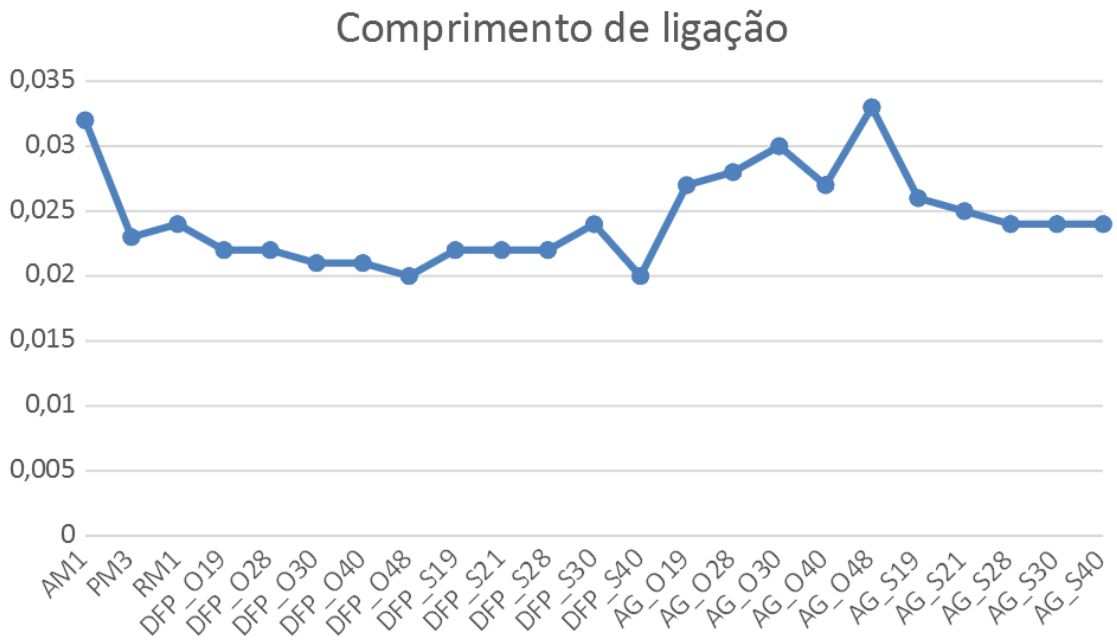


Figura 4.9: Comprimentos de ligação médios calculados.

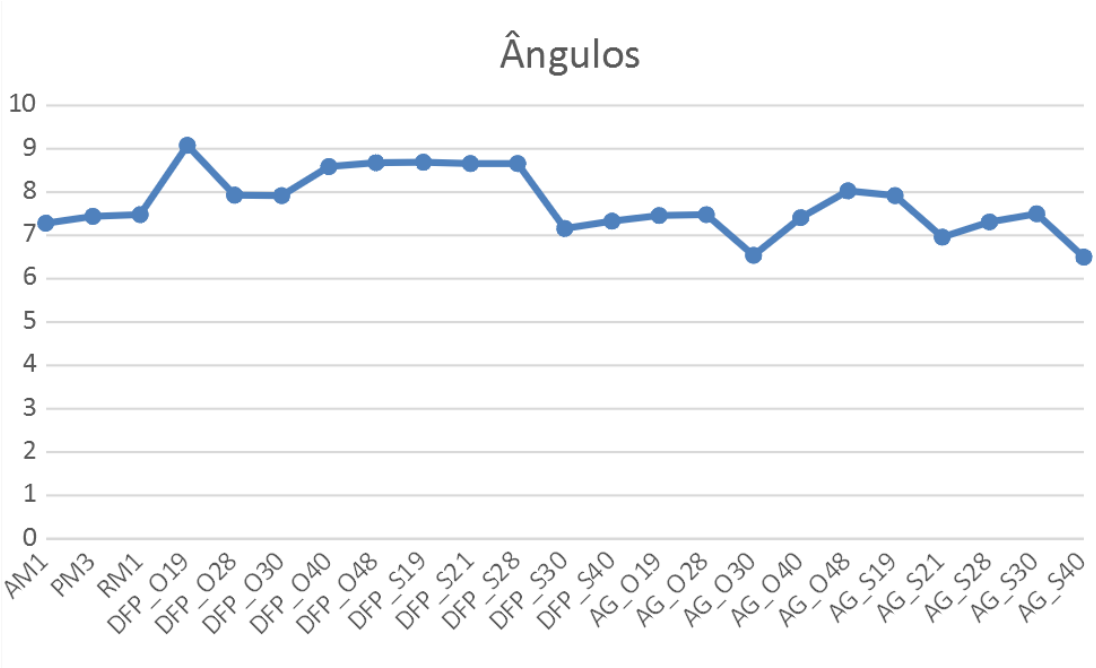


Figura 4.10: Ângulos médios calculados.

O algoritmo genético encontrou médias próximas às do RM1, chegando a melhorar algumas delas. Mas o DFP obteve erros médios piores do que o RM1 e o AG para a entalpia de formação, o momento de dipolo e o PI. Apenas o comprimento de ligação alcançou melhoria na otimização de todos os pontos. Os ângulos, em geral, também pioraram, com exceção do ponto DFP_S30 (cuja média foi melhor do que o RM1 e o AG_S30, porém não melhorou o comprimento de ligação) e do DFP_S40 (que obteve uma média melhor que a do RM1, mas pior do que a do ponto que o originou: AG_S40).

Podemos inferir que a função custo gerou resultados que supervalorizaram o comprimento de ligação ou os ângulos em detrimento das demais propriedades. Outra possibilidade é que a função era adequada, mas geraria melhores médias somente se fossem avaliados mais pontos e incluídas mais moléculas. Uma terceira hipótese é que algumas das moléculas do banco de dados que utilizaram pesos especiais estivessem com esse peso desajustado.

No caso do AG, resultados ligeiramente melhores foram encontrados, mas essas médias foram avaliadas em um programa próprio, que permitiu atribuir pesos diferentes dos utilizados no PARAM. A melhoria alcançada foi pequena, insuficiente para justificar a definição de um novo método semiempírico, mas houve uma melhoria, o que indica que a função custo funcionou melhor do que a do procedimento do PARAM. Por isso, é provável que, se a otimização utilizasse um procedimento semelhante na avaliação (usando os erros médios em vez da soma completa de erros e ignorando os pesos especiais), a otimização funcionaria a contento, encontrando erros médios próximos ou menores do que os do RM1, em vez de aumentá-los.

Considerando essa possibilidade, a principal contribuição deste trabalho é ter descoberto esse desajuste na função custo e deixar orientações e sugestões para um trabalho futuro que dê continuidade a esta busca:

- Assim como realizado pelo AG, utilizar somente a funcionalidade de avaliação de um conjunto de parâmetros, provida pelo PARAM, e utilizar os erros médios obtidos para calcular a função custo;
- Criar um programa próprio que implemente um método de otimização mais eficiente do que o DFP;
- Aproveitar a possibilidade de paralelismo do Santos Dumont, executando simultaneamente a minimização dos 10 pontos e, principalmente, paralelizando os cálculos da função custo, pois os arquivos do banco de dados podem ser divididos para calcular várias parcelas da função em nós diferentes, somando os valores calculados ao final. Isso diminuiria o tempo de cada ciclo.

É possível ainda que, mesmo com esse procedimento, nenhum ganho substancial seja alcançado com a otimização dos parâmetros. Considerando que o RM1 tem a mesma estrutura algébrica do AM1, mas já apresenta uma diminuição considerável no erro, e que o algoritmo genético não apresentou ganho considerável, é razoável a suposição de que os parâmetros do RM1 já estejam quase no limite mínimo de erro possível, e que não seja possível um grande avanço nos resultados enquanto essa estrutura for mantida. Porém, se a mudança de estrutura for difícil, ainda vale a pena verificar se é possível a diminuição mantendo a estrutura, pois a falta de correspondência entre a função custo e as médias dos erros por propriedade torna esse resultado inconclusivo.

Capítulo 5

Conclusões

Na busca por parâmetros otimizados para o RM1, o presente trabalho não encontrou resultados melhores que o trabalho de Maia et al (2019), que usou algoritmos genéticos no procedimento de parametrização. Por isso, nos parece que o ponto obtido para o método RM1 já é um excelente ponto.

A otimização utilizou uma função custo que caiu bastante, mas esse resultado não se refletiu em maior poder preditivo dos novos pontos otimizados, pois os erros médios das propriedades para o conjunto de testes foram todos mais elevados que o do próprio RM1. É possível que essa função custo não tenha sido eficiente e precisasse ser melhorada. Ou, talvez, melhores resultados fossem encontrados se fossem avaliados mais pontos.

Uma análise mais abrangente não foi possível devido ao prazo limitado para a defesa desta dissertação. Como a função custo tinha bons resultados quando comparados aos iniciais, a estratégia foi mantida, até que, ao final, a validação estatística incluiu dados extras que evidenciaram o problema, mas não havia mais tempo para mudar de estratégia. Porém, essa limitação foi aprendida e serve para orientar um trabalho futuro.

De qualquer forma, a dificuldade em encontrar parâmetros que ofereçam um ganho significativo levanta a hipótese de que, para baixar os erros médios das propriedades, seja necessário modificar a estrutura algébrica do método, provavelmente incluindo, por exemplo, correções de dispersão (que é um tratamento para as interações não covalentes entre moléculas). Muitas moléculas no banco de dados possuem interações não covalentes, que ocorrem quando duas ou mais moléculas estão próximas, mas essas interações não são tratadas corretamente pelo conjunto de equações que definem a estrutura algébrica do RM1. Incluir esse tratamento é um possibilidade viável na tentativa de melhorar o método semiempírico. Modificações como essa foram feitas em métodos semiempíricos mais atuais, como o PM7 e a série OMx.

Referências Bibliográficas

[1] ANDRADE, C. H., TROSSINI, G. H. G., FERREIRA, E. I. "MODELAGEM MOLECULAR NO ENSINO DE QUÍMICA FARMACÊUTICA", *Revista Eletrônica de Farmácia*, v. 7, n. 1, pp. 23, abr. 2010. doi: 10.5216/ref.v7i1.9603. Disponível em: <https://www.revistas.ufg.br/REF/article/view/9603>.

[2] SERWAY, R. A., JEWETT JR., J. W. *Física para cientistas e engenheiros - Luz, óptica e física moderna*, v. 4. São Paulo, Cengage Learning Brasil, 2018. ISBN: 9788522127139. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522127139/. Acesso em: 22 set. 2021.

[3] LASCHUK, E. F. *Novo Formalismo Semi-Empírico para Cálculos Químico-Quânticos*. Tese de Doutorado, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2005. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/7140/000495676.pdf>.

[4] ROCHA, G. B., SIMAS, A. M. "Métodos Semi-empíricos de Estrutura Eletrônica em Química Quântica". In: Morgon, N. H., Coutinho, K. (Eds.), *Métodos de Química Teórica e Modelagem Molecular*, Livraria da Física, p. 29–71, São Paulo, 2007.

[5] SILVA, D. A. *Implementação de uma correção de dispersão empírica aos métodos semiempíricos RM1 (Recife Model 1) e PM6 (Parametric method 6)*. Dissertação de Mestrado, Universidade Federal de Sergipe, São Cristóvão, 2017.

[6] DEWAR, M. J. S., THIEL, W. "Ground States of Molecules. 38. The MNDO Method. Approximations and Parameters", *Journal of the American Chemical Society*, v. 99(15), pp. 4899–4907, 1977.

[7] DEWAR, M. J. S., ZOEBISCH, E. G., HEALY, E. F., et al. "Development and use of quantum mechanical molecular models. 76. AM1: a new general

purpose quantum mechanical molecular model", *Am. Chem. Soc.*, v. 107, pp. 3902–3909, 1985.

[8] RAMOS, C. A. C. L. *O modelo RM1 para Boro e Selênio*. Dissertação de Mestrado, Universidade Federal de Pernambuco, Recife, 2016.

[9] "Google Acadêmico - Citações para o artigo "Development and use of quantum mechanical molecular models. 76. AM1: a new general purpose quantum mechanical molecular model", de Dewar et al". 2021. Disponível em: `https://scholar.google.com.br/scholar?cites=5325818828521355123`. Acesso em: 31 out. 2021.

[10] ROCHA, G. B., FREIRE, R. O., SIMAS, A. M., et al. "RM1: A reparameterization of AM1 for H, C, N, O, P, S, F, Cl, Br, and I", *Comput. Chem.*, v. 27, pp. 1101–1111, 2006.

[11] STEWART, J. J. P. "MOPAC web site". 2014. Disponível em: <`http://openmopac.net/`>. Acesso em 26 ago. 2021.

[12] MAIA, J. D. C., CARVALHO, G. A. U., MANGUEIRA JÚNIOR, C. P., et al. "GPU Linear Algebra Libraries and GPGPU Programming for Accelerating MOPAC Semiempirical Quantum Chemistry Calculations", *Chem. Theory Comput*, v. 8, pp. 3072–3081, 2012.

[13] "Google Acadêmico - Citações para o artigo "RM1: A reparameterization of AM1 for H, C, N, O, P, S, F, Cl, Br, and I", de Rocha et al". 2021. Disponível em: `https://scholar.google.com.br/scholar?cites=12276489539814680497`. Acesso em: 31 out. 2021.

[14] MAIA, J. D. C., DARDENE, L., CUSTÓDIO, F., et al. "Relatório apresentado ao SDumont, do projeto intitulado "Buscando o mínimo global dos métodos quânticos semiempíricos NDDO (rm2)"". 2019. Disponível em: `https://sdumont.lncc.br/projects_view.php?pg=projects&status=ongoing`. Acesso em: 30 out. 2021.

[15] "OPENMOPAC - Código do PARAM, subprojeto do MOPAC". 2021. Disponível em: <`https://github.com/openmopac/MOPAC/blob/master/src/param.F90`>.

[16] FLETCHER, R. *Practical Methods of Optimization*. 2 ed. Dundee, Wiley-Interscience, 1987. ISBN: 0 471 91547 5.

[17] "Canva". 2021. Disponível em: <`https://www.canva.com/`>.

[18] SANT'ANNA, C. M. R. "Uma Introdução à Modelagem Molecular Aplicada à Química Medicinal". In: *Química medicinal: as bases moleculares da ação dos fármacos*, Artmed, Porto Alegre, 2015. ISBN: 9788582711187. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582711187/. Acesso em: 22 set. 2021.

[19] ATKINS, P., JONES, L. *Princípios de química - questionando a vida moderna e o meio ambiente*. 5 ed. Porto Alegre, Bookman, 2012. ISBN: 9788540700543.

[20] MUNDIM, K. "Teoria dos Operadores - Aula de Química Quântica da Universidade de Brasília". 2016. Disponível em: http://ensinoadistancia.pro.br/EaD/QQ/aula-11/aula-11.htm. Acesso em: 20 out. 2021.

[21] BASTOS, C. M. O. "Método de Hartree-Fock". 2015. Disponível em: <https://www.ifsc.usp.br/~strontium/Teaching/Material2015-2%20SFI5814%20Atomicamolecular/Seminario%20-%20Carlos%20-%20Hartree-Fock.pdf>.

[22] SHERRILL, C. D. "An Introduction to Hartree-Fock Molecular Orbital Theory". 2000. Disponível em: <http://vergil.chemistry.gatech.edu/notes/hf-intro/hf-intro.pdf>.

[23] LEVINE, I. N. *Physical Chemistry*. 6 ed. New York, McGraw-Hill, 2009. ISBN: 9780072538625.

[24] LEVINE, I. N. *Quantum Chemistry*. New York, Pearson Education, 2013. ISBN: 978-0321803450.

[25] STEWART, J. J. P. "MOPAC Manual - Theory - Parameters used in Semiempirical Methods". 2014. Disponível em: http://openmopac.net/manual/parameters.html. Acesso em: 31 out. 2021.

[26] LEWARS, E. G. *Computational Chemistry: Introduction to the Theory and Applications of Molecular and Quantum Mechanics*. 2 ed. Londres, Springer, 2011. ISBN: 978-90-481-3860-9.

[27] STEWART, J. J. P. "Optimization of Parameters for Semiempirical Methods", *Comput. Chem.*, v. 10, n. 2, pp. 209–220, 1989.

[28] THIEL, W. "Semiempirical quantum-chemical methods", *WIREs Comput Mol Sci*, v. 4, pp. 145–157, 2014. doi: 10.1002/wcms.1161.

[29] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., et al. "Fortran Numerical Recipes". In: *Numerical Recipes in Fortran 90 : The Art of Parallel Scientific Computing*, v. 2, 2 ed., Press Syndicate of the University of Cambridge, Cambridge, 1996. ISBN: 0-521-57439-0. Disponível em: http://s3.amazonaws.com/nrbook.com/book_F210.html. Acesso em: 08 out. 2021.

[30] CONSTALES, D., YABLONSKY, G. S., D'HOOGE, D. R., et al. "Chapter 9 - Experimental Data Analysis: Data Processing and Regression". In: Constales, D., Yablonsky, G. S., D'hooge, D. R., et al. (Eds.), *Advanced Data Analysis & Modelling in Chemical Engineering*, Elsevier, pp. 285–306, Amsterdam, 2017. ISBN: 978-0-444-59485-3. doi: https://doi.org/10.1016/B978-0-444-59485-3.00009-6. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780444594853000096>.

[31] RIBEIRO, A. A., KARAS, E. W. *Otimização contínua: Aspectos teóricos e computacionais*, v. 4. São Paulo, Cengage Learning Brasil, 2014. ISBN: 9788522120024. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522120024/. Acesso em: 29 nov. 2021.

[32] FRIEDLANDER, A. *Elementos de Programação Não-Linear*. Campinas, Editora da Unicamp, 1994. Disponível em: https://www.ime.unicamp.br/~friedlan/livro.html. Acesso em: 25 set. 2021.

[33] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., et al. "Fortran Numerical Recipes". In: *Numerical Recipes in Fortran 77 : The Art of Scientific Computing*, v. 1, 2 ed., Press Syndicate of the University of Cambridge, Cambridge, 1992. ISBN: 0-521-43064-X. Disponível em: http://s3.amazonaws.com/nrbook.com/book_F210.html. Acesso em: 08 out. 2021.

[34] ALVES, C. J. S. "Material de aula do curso de Análise Numérica da Universidade de Lisboa - Método de Newton e da Secante". 1997. Disponível em: <https://www.math.tecnico.ulisboa.pt/~calves/courses/eqn/capii213.html>.

[35] DA SILVEIRA GOULART, E. *Matrizes Quase-Newton Esparsas para Problemas de Otimização Não Linear de Grande Porte*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005. Disponível em: <https://w1files.solucaoatrio.net.br/atrio/ufrj-pem_upl/THESIS/1118/pemufrj2005dscevandrodasilveiragoulart.pdf>.

[36] STEWART, G. W. "A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives", *ACM*, v. 14, pp. 72–83, 1967.

[37] WAINER, J. "Descida do gradiente com passos grandes, metodos quadráticos e sem gradiente - Aula de Álgebra linear e otimização para aprendizado de máquina, oferecida pela UNICAMP". 2020. Disponível em: <https://www.ic.unicamp.br/~wainer/cursos/1s2020/431/otimizacao4.html>.

[38] KATO, R., PAIVA, V., IZIDORO, S. "Algoritmos Genéticos", *BIOINFO - Revista Brasileira de Bioinformática*, v. 1, jul. 2021. doi: 10.51780/978-6-599-275326-13. Disponível em: <https://bioinfo.com.br/algoritmos-geneticos/>.

[39] CATARINA, A. S. "Algoritmos Genéticos - Aula apresentada no curso de Ciência da Computação da Unioeste - PR". 2018. Disponível em: <https://www.inf.unioeste.br/~adair/MOA/Notas%20de%20Aula/Slides%2002%20-%20Algoritmos%20Geneticos.pdf>.

[40] MORGAN, B. "Differential Evolution - Automated Machine Learning". 2021. Disponível em: <https://towardsdatascience.com/unit-7-differential-evolution-automated-machine-learning-eb22014e592e>.

[41] CUSTÓDIO, F. L., BARBOSA, H. J., DARDENNE, L. E. "A multiple minima genetic algorithm for protein structure prediction", *Applied Soft Computing*, v. 15, pp. 88–99, 2014. ISSN: 1568-4946. doi: https://doi.org/10.1016/j.asoc.2013.10.029. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494613003633>.

[42] "OPENMOPAC - Código do PARAM, subprojeto do MOPAC - definição dos pesos da otimização". 2021. Disponível em: <https://github.com/openmopac/MOPAC/blob/master/src/PARAM/datinp.F90>.

[43] "SDumont - Sistema Brasileiro de Computação Petaflópica". 2021. Disponível em: <https://sdumont.lncc.br/machine.php?pg=machine#>.

[44] "Departamento de Química da UFPB". 2021. Disponível em: <https://www.ufpb.br/dq>.

[45] "Supercomputador Santos Dumont (SDumont) - Manual". 2021. Disponível em: <https://sdumont.lncc.br/support_manual.php?pg=support#>.

[46] “Download PuTTY - a free SSH and telnet client for Windows”. 2021. Disponível em: <https://www.putty.org/>.

[47] “WinSCP :: Official Site :: Free SFTP and FTP client for Windows”. 2021. Disponível em: <https://winscp.net/>.

Apêndice A

Guia para uso do PARAM para reparametrização de um método semiempírico

O PARAM (programa que tem várias funções, sendo a principal delas a otimização de parâmetros de um método semiempírico) tem uma versão para Linux em que o arquivo executável se chama PARAM_Linux.exe. Este guia informa os passos necessários para executar essa versão do programa a partir do Windows, mas usando uma máquina Linux acessada remotamente.

O motivo de informar os passos para esta situação específica é que é muito frequente que os interessados em executar o programa sejam alunos de uma universidade que permite acesso remoto às suas máquinas Linux. Também é comum que esses alunos contenham uma máquina apenas com Windows, sendo conveniente utilizar um programa como o PuTTY para acesso remoto a partir desse sistema operacional. No caso de interessados que utilizem Linux, os passos referentes a ferramentas do Windows não são necessários (o terminal pode ser acessado diretamente por `ssh` e os arquivos copiados com o comando `scp`).

Os exemplos apresentados neste capítulo mostram o acesso ao supercomputador Santos Dumont, mas se aplicam a qualquer máquina Linux que possa ser acessada remotamente para executar o PARAM.

A.1 Instalação e uso do programa para acesso remoto

No Windows, uma das ferramentas mais intuitivas para acesso SSH a uma máquina Linux remotamente é o PuTTY [46]. Esta seção detalha como ela pode ser instalada e usada, mas qualquer outro programa que tenha essa funcionalidade pode

ser usado em seu lugar.

A.1.1 Instalação do PuTTY

Para realizar a instalação do PuTTY, acesse o *site* `https://www.putty.org/` e faça o *download* do instalador, conforme figuras abaixo:

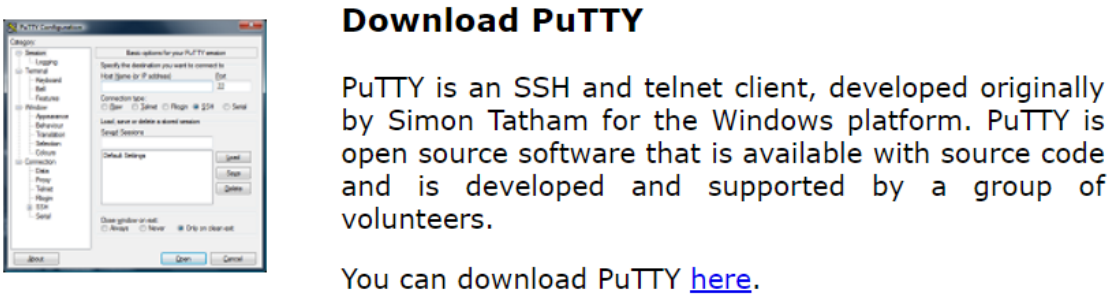


Figura A.1: Tela inicial para download do PuTTY.

Após clicar em *here*, faça o *download* do instalador adequado.

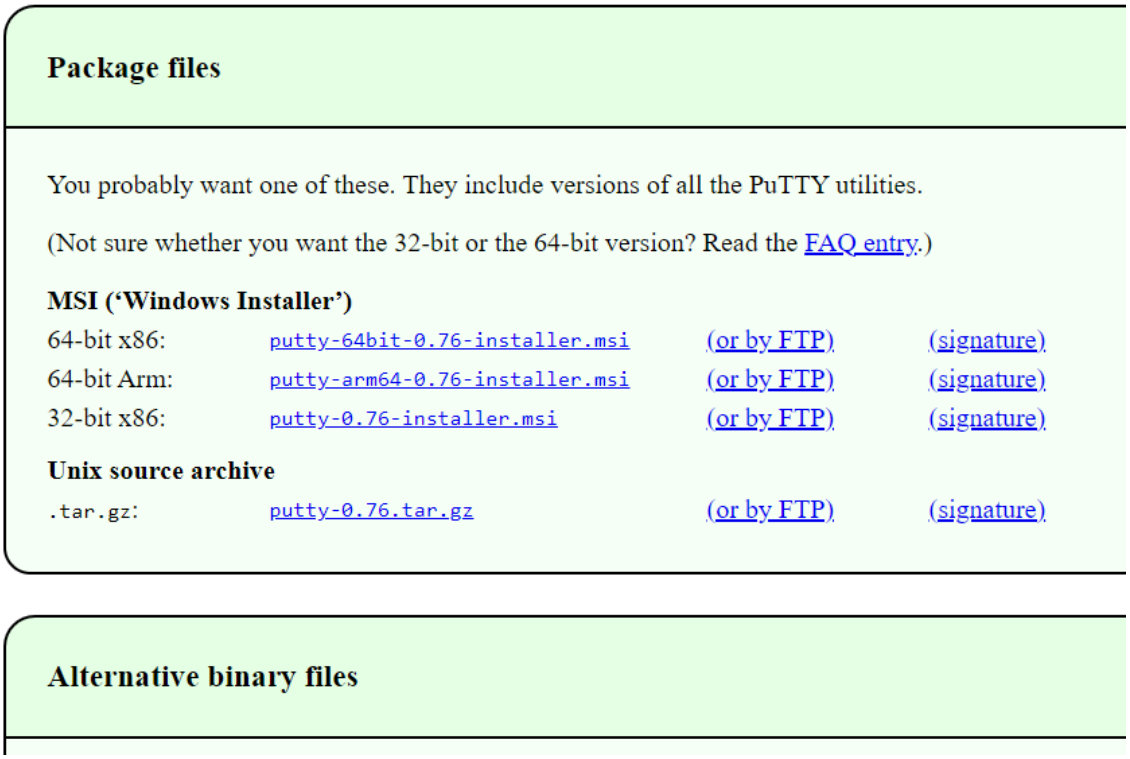


Figura A.2: Tela com opções de instaladores do PuTTY.

Execute o arquivo e faça a instalação padrão (*Next, Next, ..., Finish*).

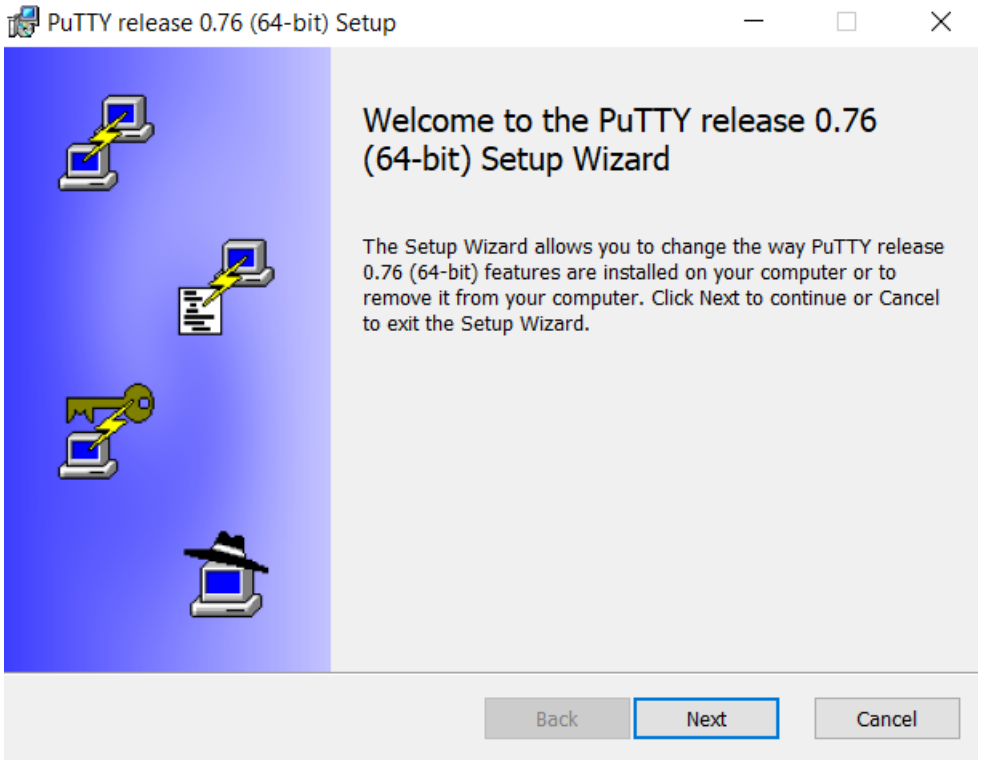


Figura A.3: Instalação do PuTTY.

A.1.2 Utilização do PuTTY

Após finalizar a instalação, execute o programa. Uma tela semelhante à abaixo será apresentada.

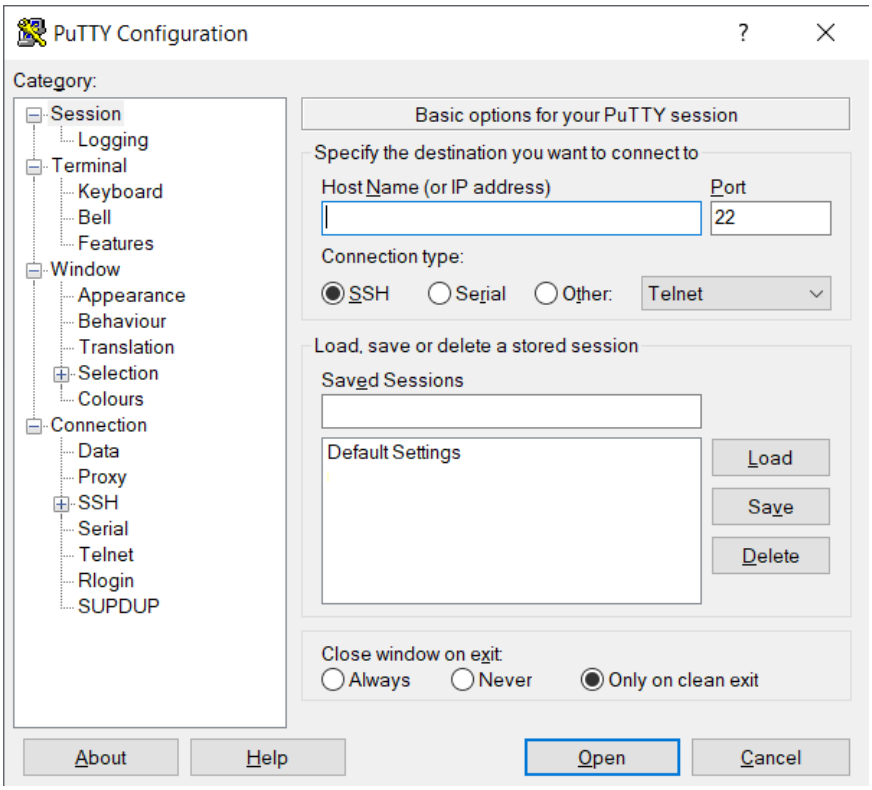


Figura A.4: Interface do PuTTY.

Preencha o campo *Host Name* com o nome ou o IP do servidor. Caso os responsáveis pelo servidor não tenham informado uma porta alternativa, mantenha a porta como 22, que é o padrão. Clique em *Open*. Se o endereço do servidor estiver correto e for permitido acessá-lo a partir da sua localização, uma tela de terminal Linux será aberta e solicitará *login* e senha.

Figura A.5: Solicitação de *login* no PuTTY.

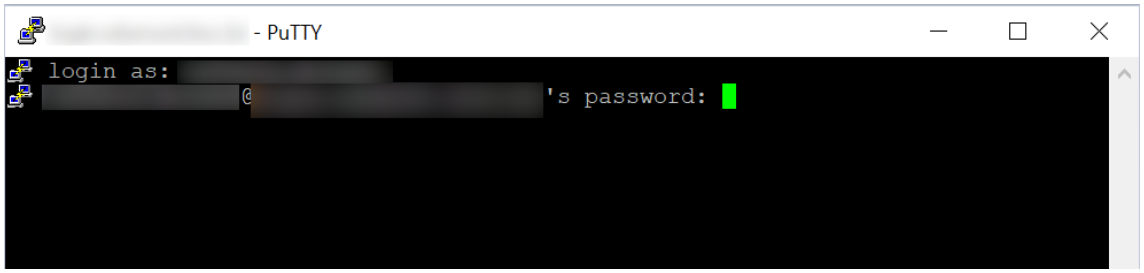


Figura A.6: Solicitação de senha após *login* no PuTTY.

Enquanto a senha é digitada, nada é apresentado, o que é comum no Linux. Digite-a e pressione *Enter*. Feito isso, mensagens de apresentação do servidor são exibidas e se torna possível digitar comandos que são executados na máquina, conforme figura abaixo.

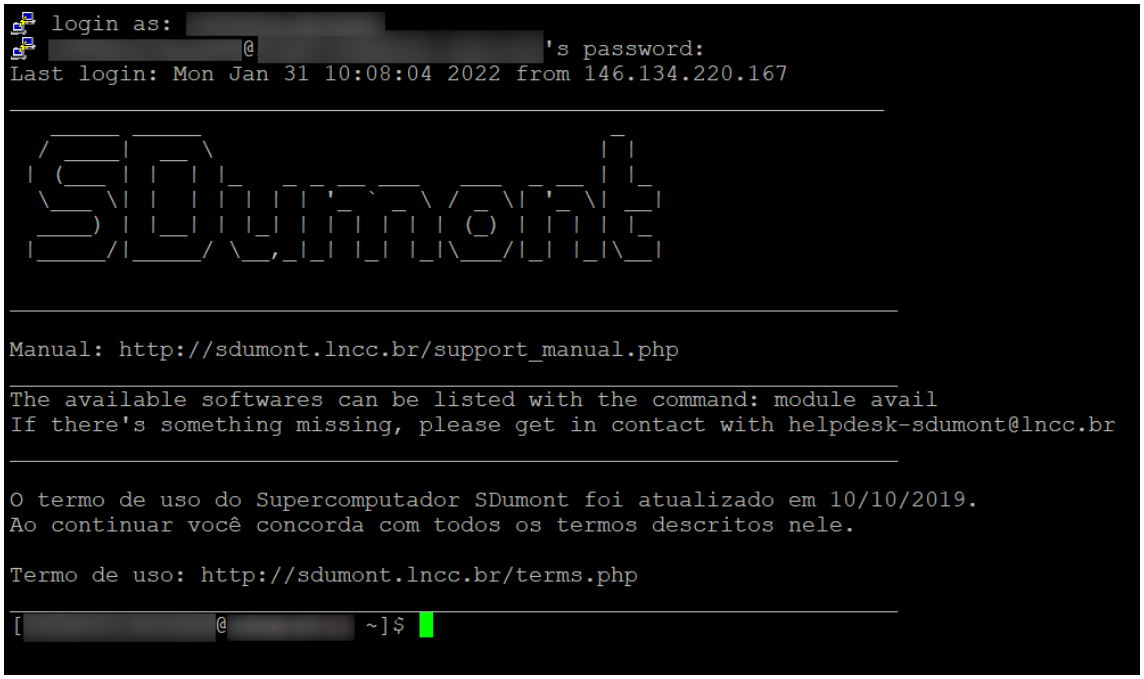


Figura A.7: Tela após *login* bem sucedido, no PuTTY.

Qualquer comando que for digitado após isso depende apenas de qual distribuição Linux está instalada no servidor e quais aplicações estão instaladas.

Algumas funcionalidades adicionais disponíveis no PuTTY se referem ao uso do *mouse*. Se algum conteúdo for selecionado (usando o botão esquerdo do *mouse*), ele é imediatamente copiado, sem necessidade de pressionar a combinação *Ctrl C* (a qual, no Linux, não funciona para copiar um conteúdo, e sim para cancelar um comando que está ativo). E se o botão direito do *mouse* for clicado, o conteúdo previamente copiado é colado (mesmo que seja um texto copiado a partir do Windows).

Para desconectar do servidor, pressione a combinação *Ctrl D* ou digite *exit* seguido de *Enter*. A janela do PuTTY será fechada.

Ao abrir o PuTTY novamente, uma forma de deixar o acesso mais rápido para os próximos acessos é salvar as informações do servidor e do usuário. Para isso, preencha o endereço do servidor (e a porta, se necessário), digite qualquer texto (que identifique o servidor) no campo *Saved Sessions* e salve.

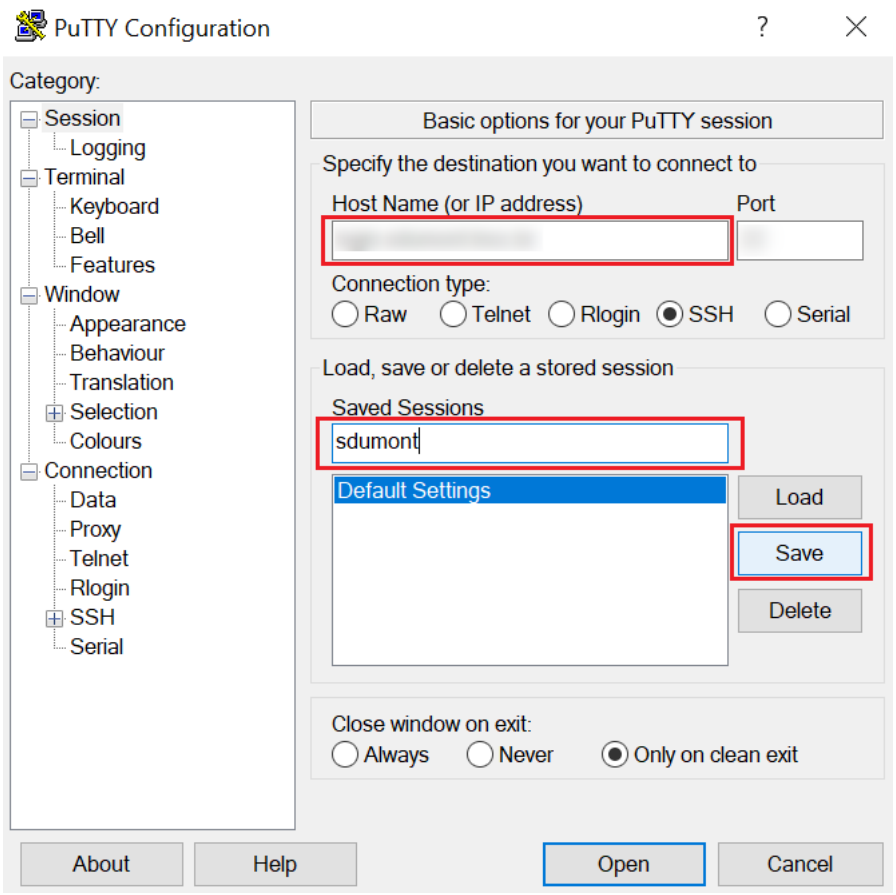


Figura A.8: Instruções para salvar uma sessão no PuTTY.

O texto identificador do servidor será listado entre as sessões salvas.

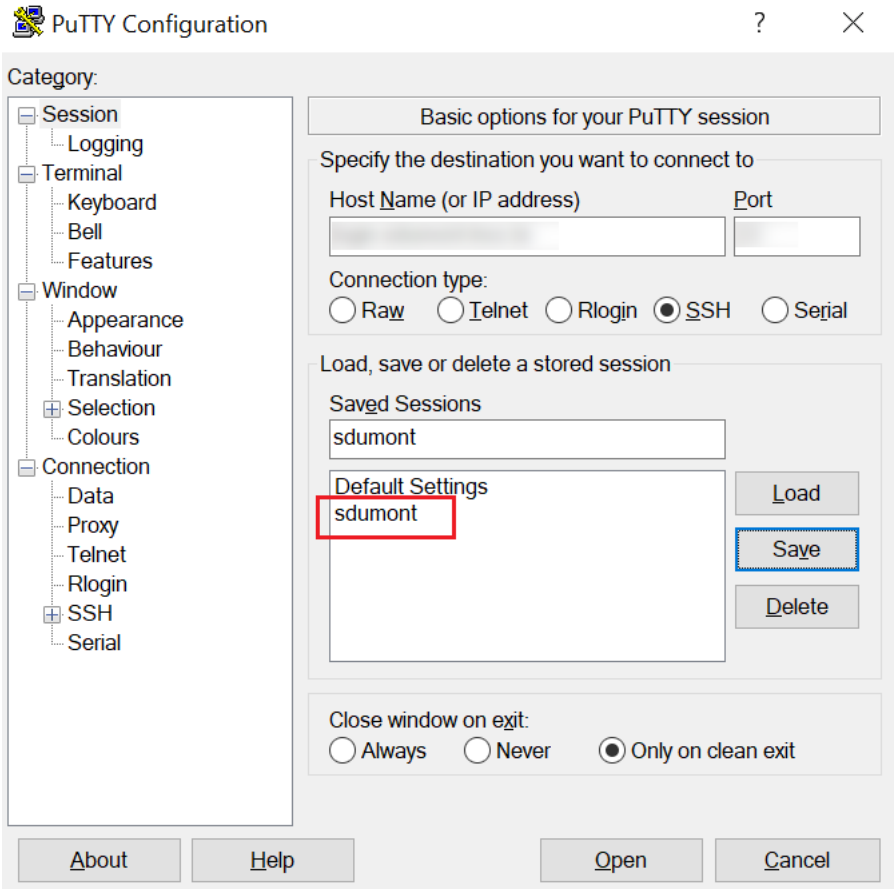


Figura A.9: Demonstração de sessão salva no PuTTY.

Selecione a categoria *Connection* - *Data* e informe o *login*. Assim, o *login* será preenchido automaticamente. O PuTTY não permite que também seja salva a senha.

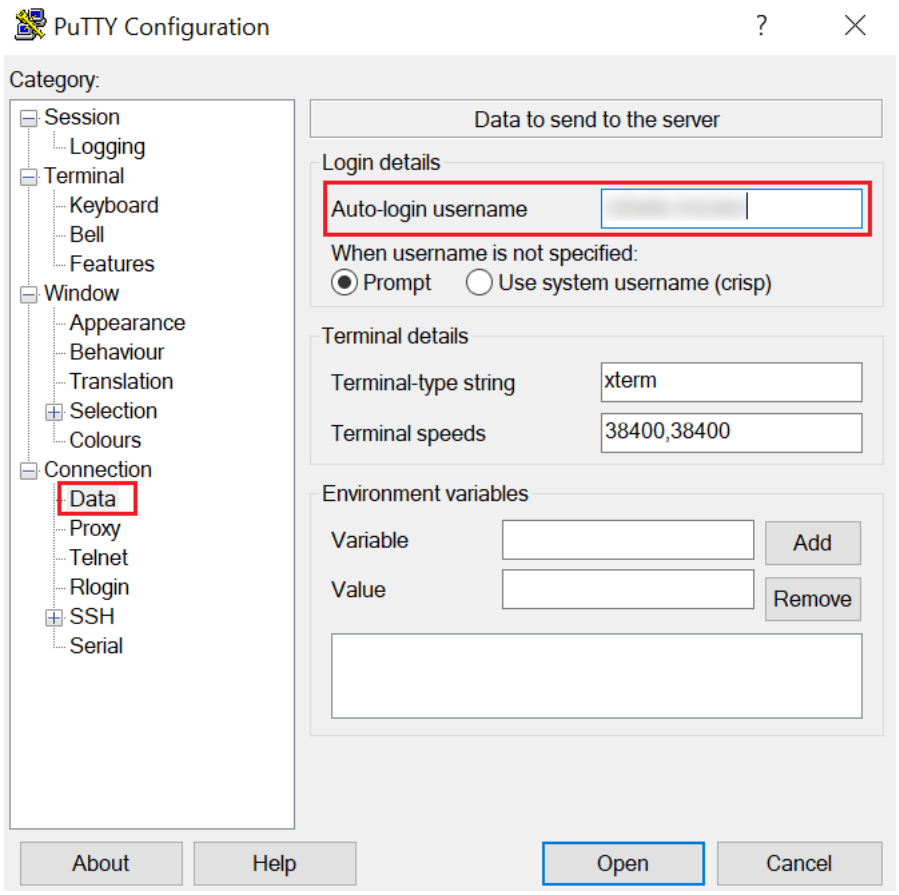


Figura A.10: Instruções para salvar o *login* junto à sessão do PuTTY.

Selecione a categoria *Session* e salve novamente.

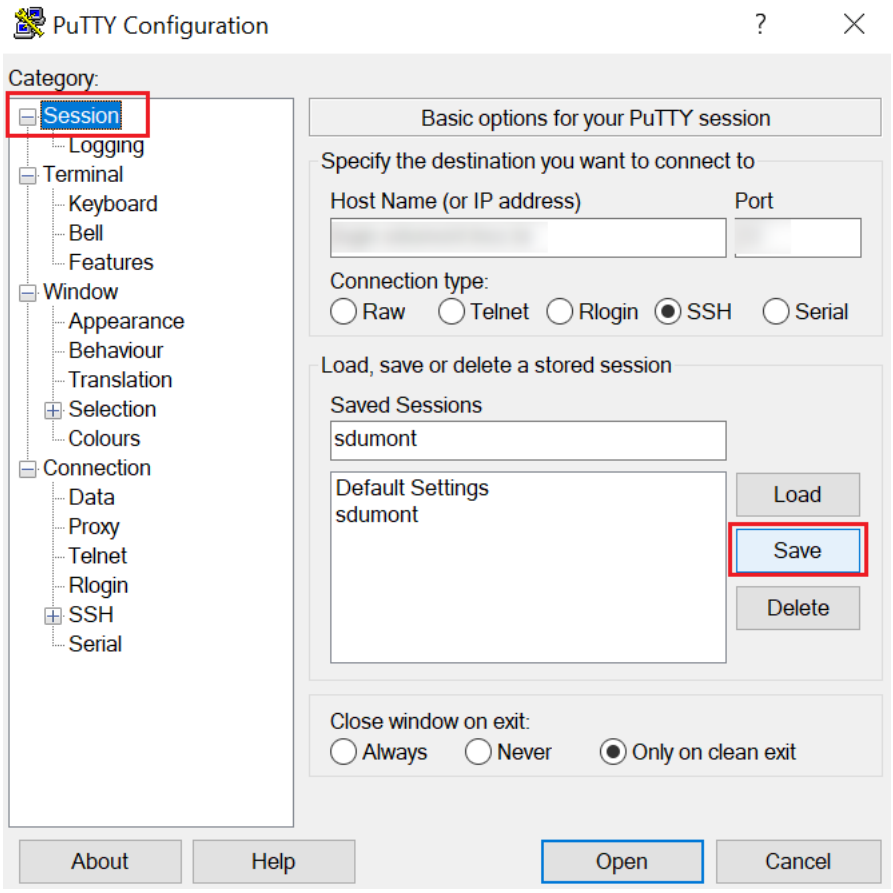


Figura A.11: Indicação do local onde deve ser clicado para voltar a salvar a sessão.

Após esse procedimento, para fazer o acesso remoto ao servidor, será suficiente abrir o PuTTY e dar um duplo clique sobre o nome da sessão salva correspondente. A tela abaixo será apresentada e será necessário apenas digitar a senha.

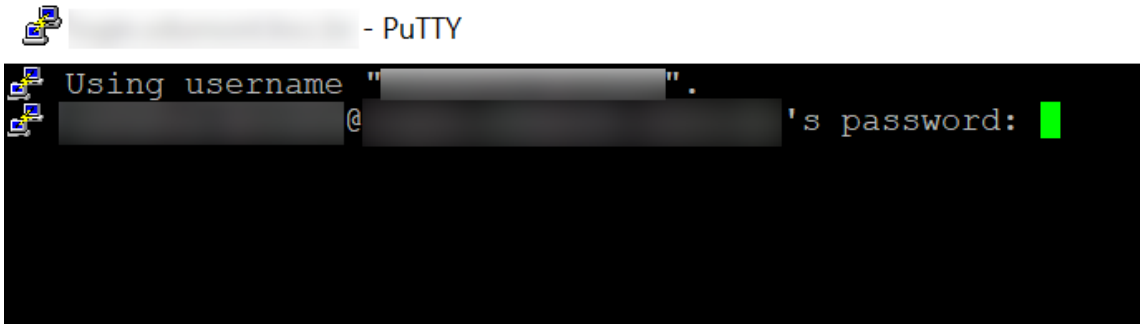


Figura A.12: Exemplo de *login* em que o nome de usuário já está salvo.

A.2 Cópia dos arquivos

Após conseguir acesso à máquina, é necessário copiar os arquivos necessários para a reparametrização, que são os seguintes:

- O executável do PARAM, nomeado como `PARAM_Linux.exe`;

- Ao menos um arquivo `.txt`, contendo o conjunto de parâmetros inicial (que será reparametrizado);
- Arquivos do banco de dados de moléculas utilizadas, com extensão `.mop`, preferencialmente organizados em uma subpasta;
- Arquivo `.dat` contendo informações utilizadas na execução do PARAM.

Como exemplo, a Figura A.13 mostra uma estrutura de pastas e arquivos prontos para execução no PARAM.

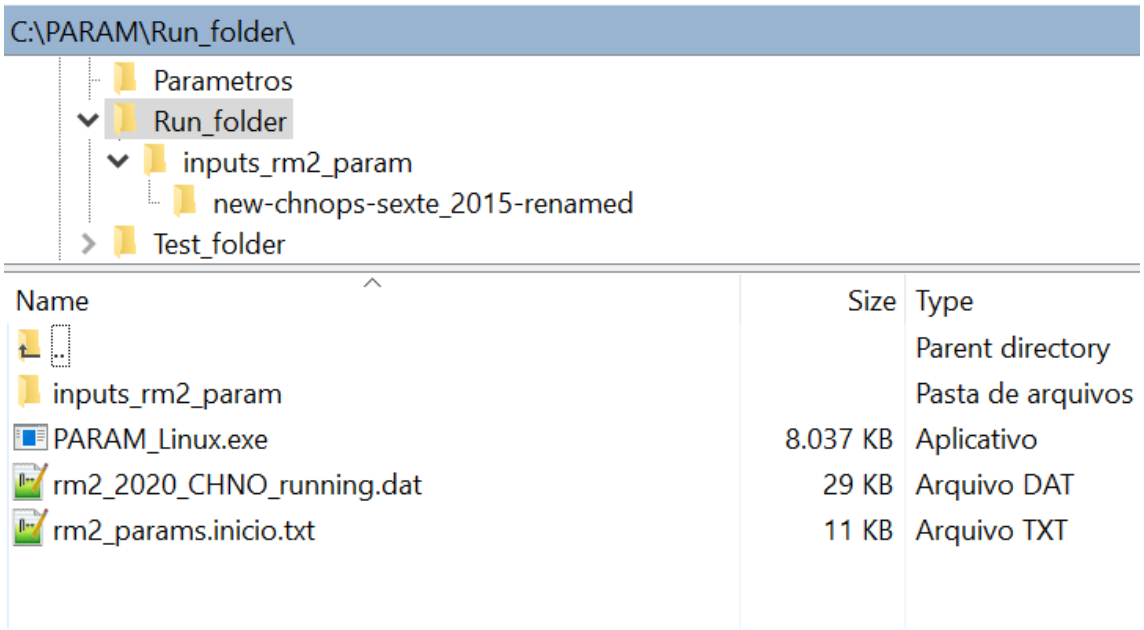


Figura A.13: Exemplo de estrutura de pastas e arquivos para execução no PARAM.

A pasta `inputs_rm2_param` contém uma subpasta `new-chnops-sexte_-2015-renamed`, que tem os arquivos de referência com extensão `.mop`, conforme Figura A.14.

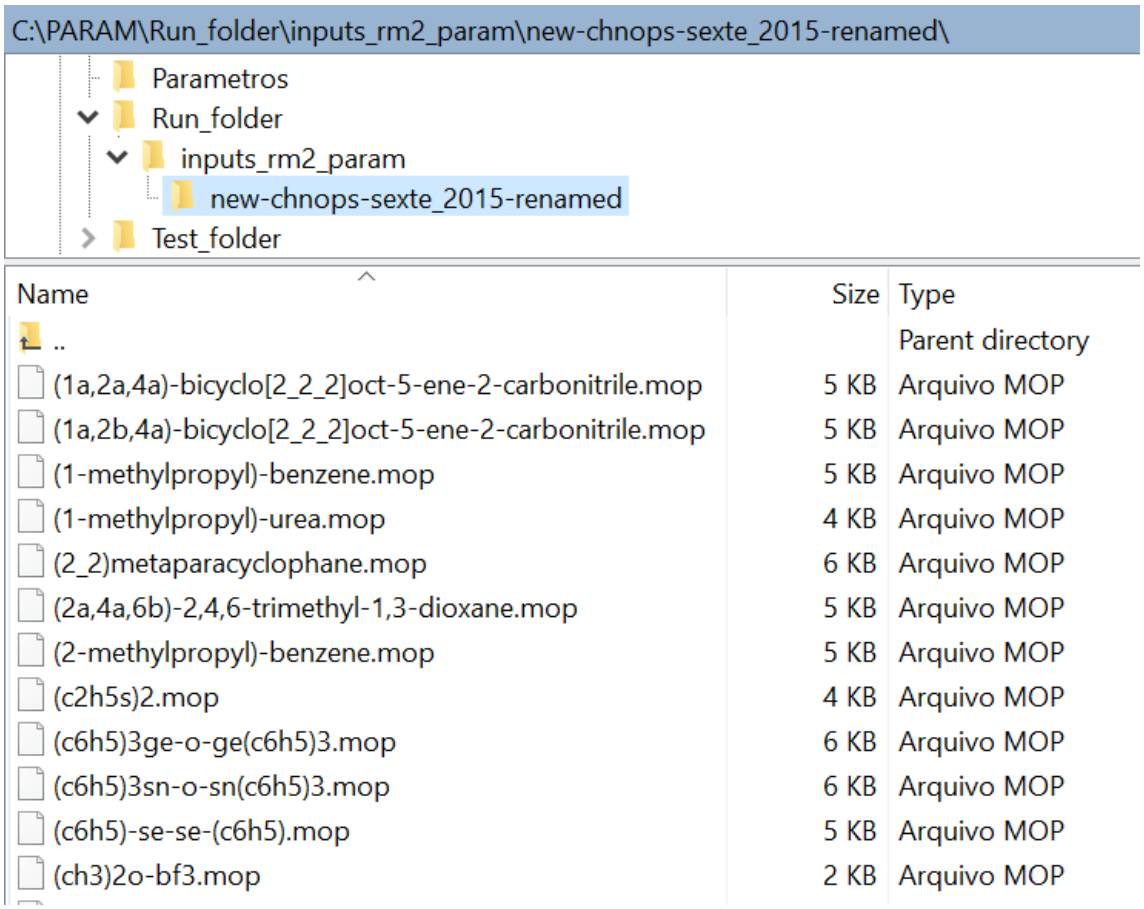


Figura A.14: Exemplo de pasta contendo arquivos de um banco de dados de moléculas de referência.

Com essa estrutura, a primeira linha do arquivo `rm2_2020_CHNO_running.dat` deve incluir estas informações:

- `params=rm2_params.inicio.txt`
- `ref=./inputs_rm2_param/new-chnops-sexte_2015-renamed`

Explicações sobre o conteúdo dos arquivos `.dat` podem ser encontradas no tópico 3.3.1.

Para copiar todos esses arquivos e pastas para o servidor Linux onde o PARAM será executado, existem várias possibilidades. Se o sistema operacional de origem também for Linux, pode ser utilizado o comando *scp*. Se for Windows, uma das ferramentas que simplifica essa tarefa é o WinSCP [47].

A.2.1 Instalação e uso do WinSCP

Para instalar o WinSCP, acesse o *site* `https://winscp.net/` e clique em "*Download Now*".

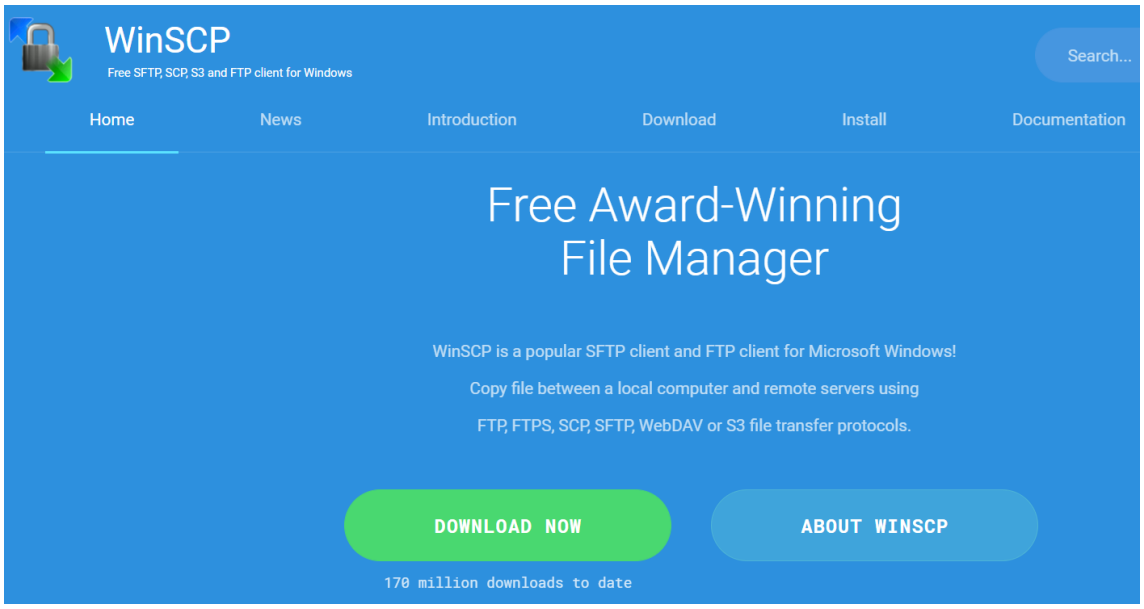


Figura A.15: Tela inicial do *site* do WinSCP.

Em seguida, clique no botão "*Download WinSCP*" referente à versão mais recente. Faça a instalação padrão. Será apresentada a opção de importar as sessões salvas no PuTTY, conforme figuras abaixo. Confirme.

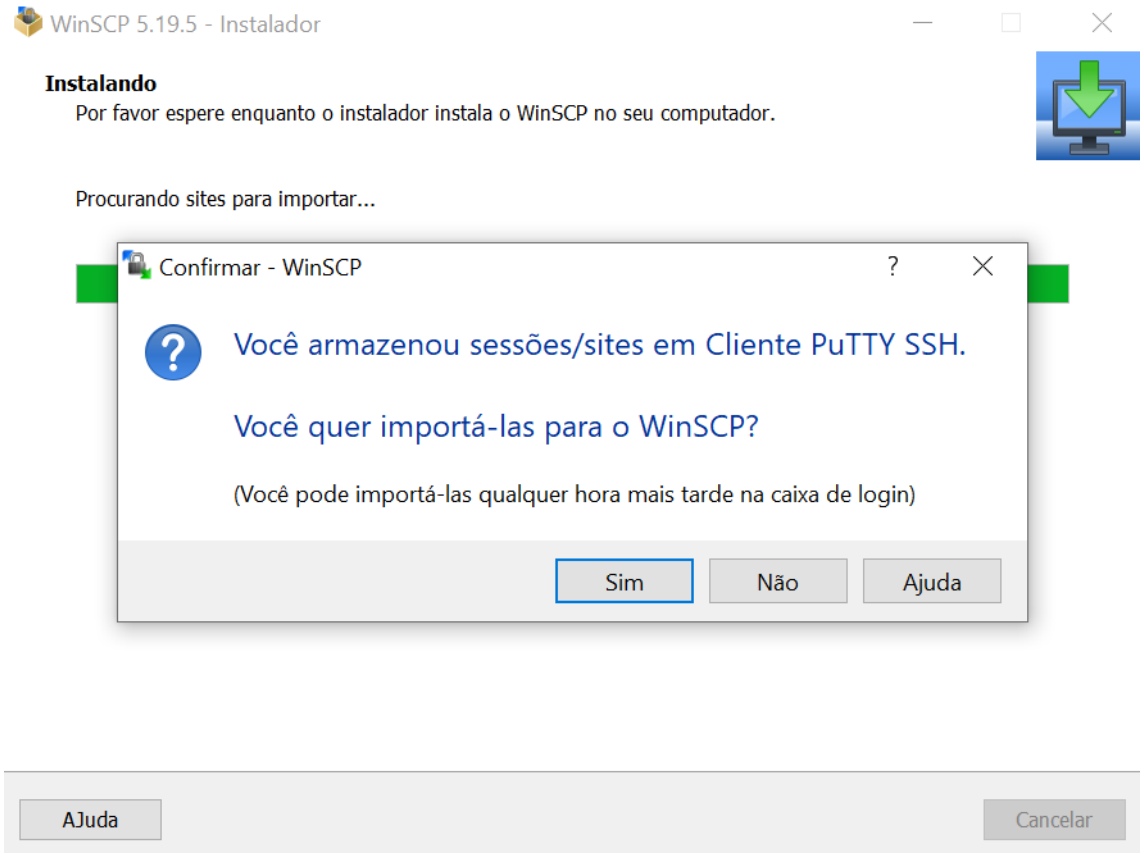


Figura A.16: Tela de instalação do WinSCP, oferecendo a opção de importar as sessões salvas no PuTTY.

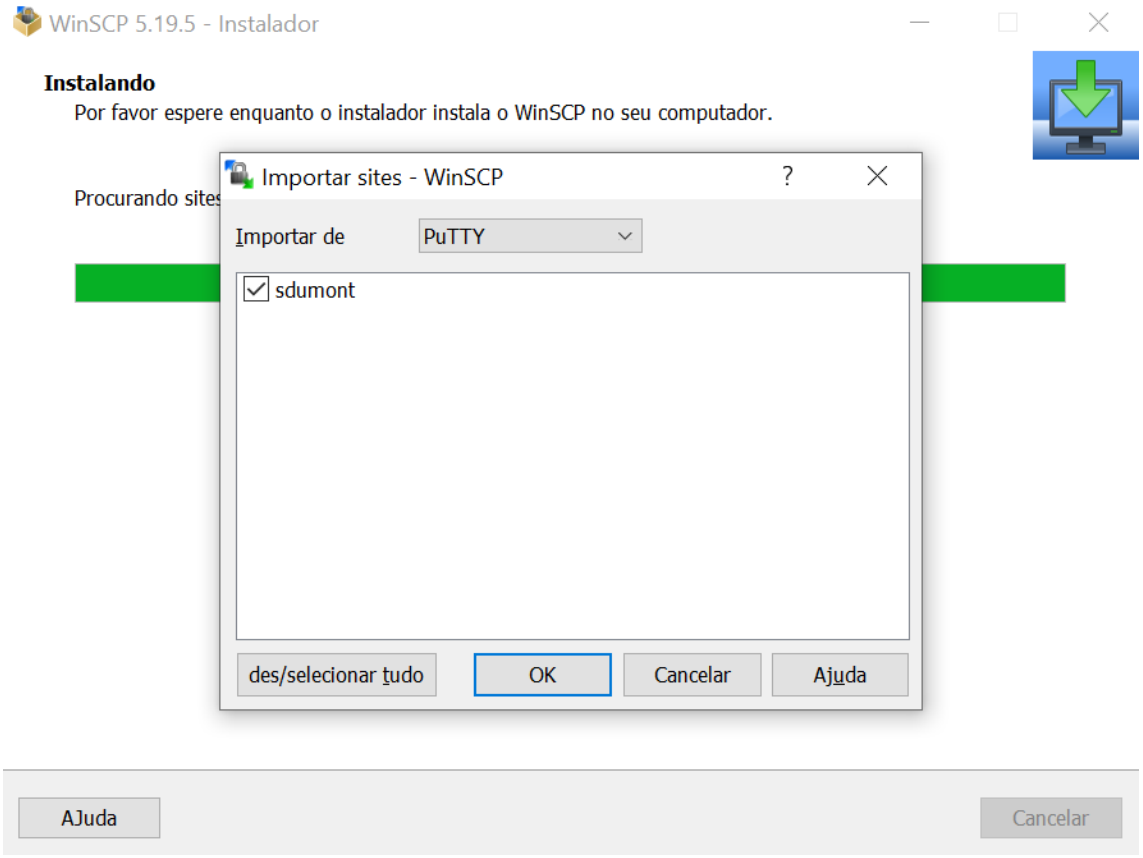


Figura A.17: Seleção das sessões a importar.

Abra o WinSCP e dê um duplo clique na identificação salva para o servidor.

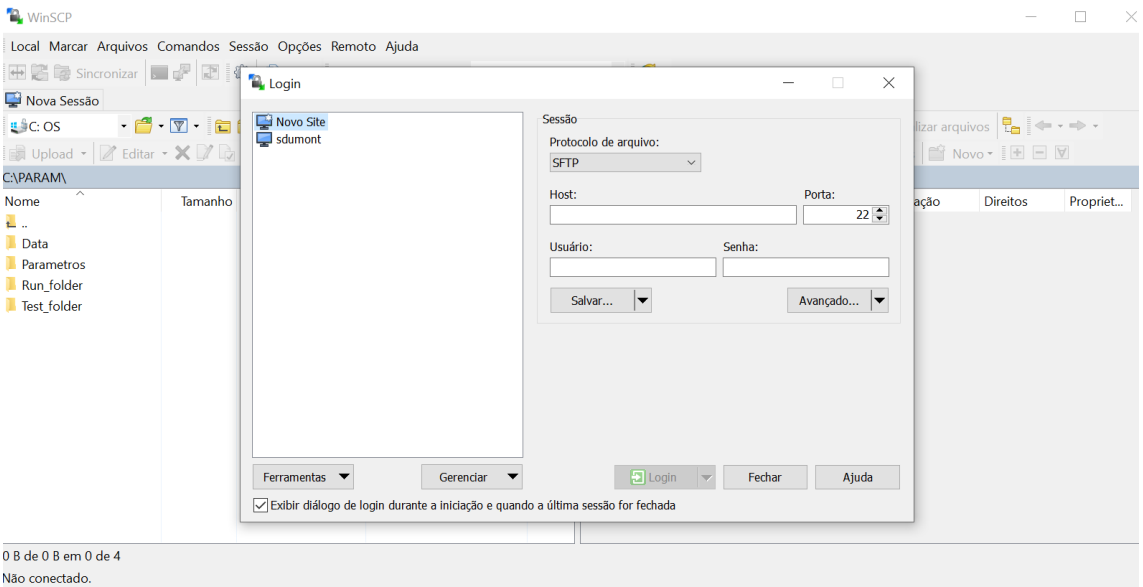


Figura A.18: Tela após abertura do WinSCP.

Será solicitada a senha. Informe-a e o *login* será efetuado no servidor, apresentando uma tela como a da Figura A.19.

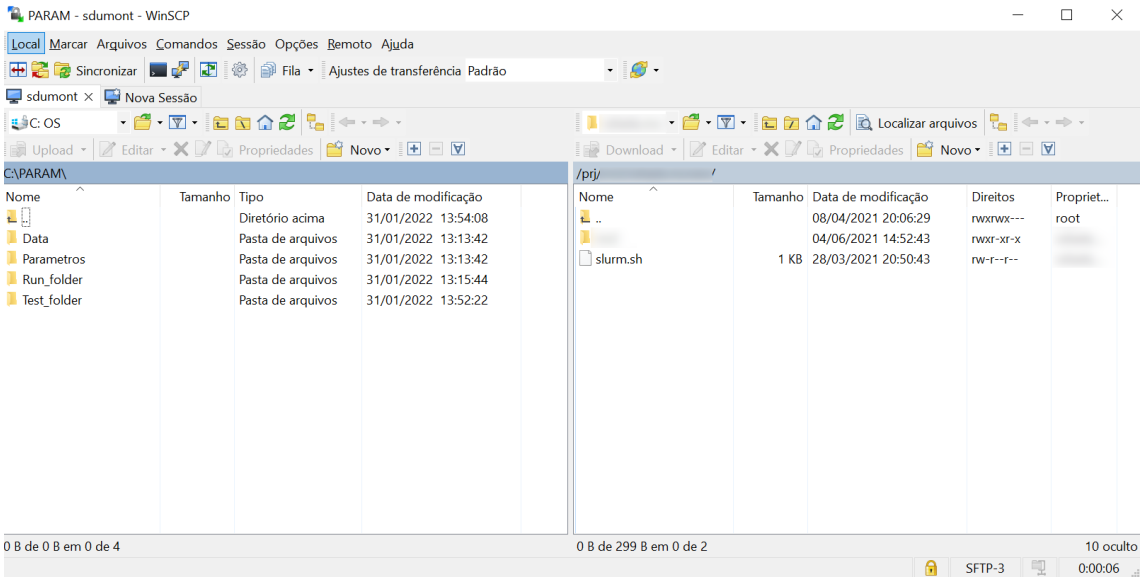


Figura A.19: Apresentação da interface do WinSCP.

No lado esquerdo, são apresentadas pastas do seu computador. No lado direito, as pastas do servidor acessado. Navegue até a pasta que contém o conteúdo que quer copiar. Para ir um nível acima na estrutura de pastas, entre na pasta ".." (por exemplo, caso esteja em uma pasta `C:\PARAM\`, o duplo clique sobre a pasta .. leva à pasta `C:\`).

Selecione as pastas e arquivos que deseja copiar e arraste da esquerda para a direita. Essa ação irá realizar o *upload* desse conteúdo para o servidor, na pasta selecionada (desde que seja uma pasta que o seu usuário tenha permissão de escrita). O conteúdo arrastado não é removido do local de origem, é apenas copiado.

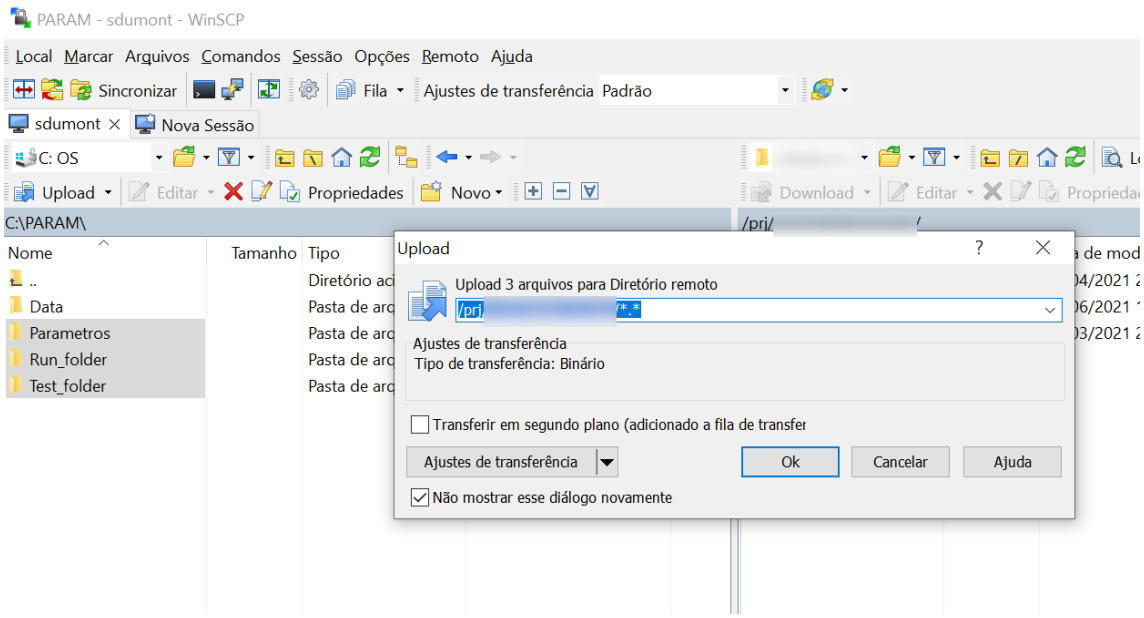


Figura A.20: Exemplo de *upload* com o WinSCP. Confirmação da ação.

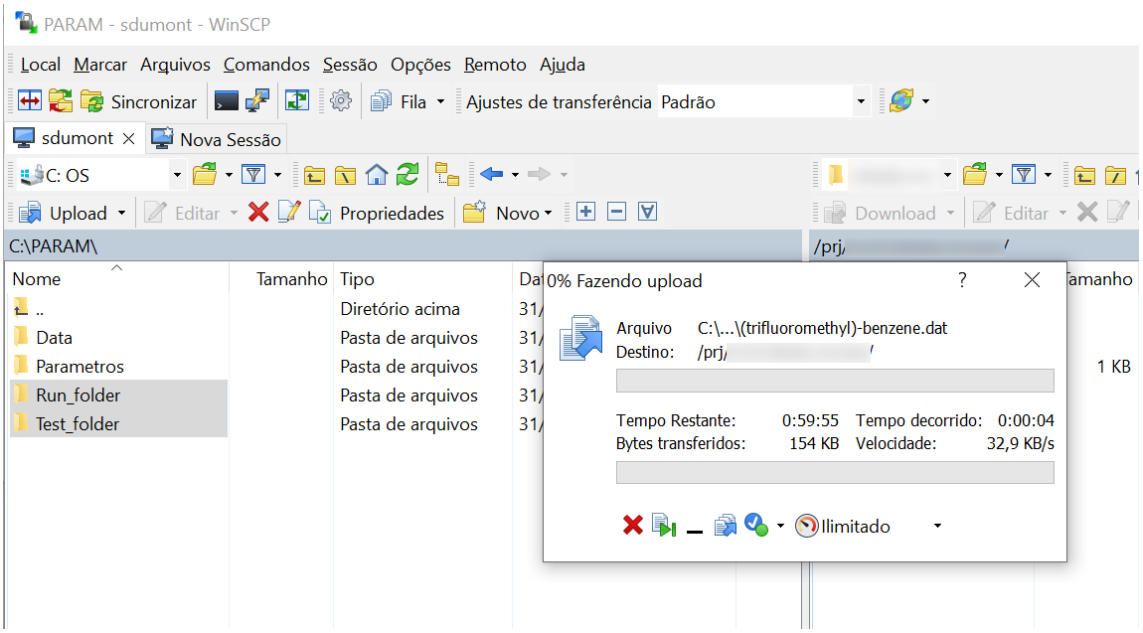


Figura A.21: Exemplo de *upload* em andamento no WinSCP.

O WinSCP também permite arrastar um conteúdo do servidor para o seu computador, realizando o *download* desse conteúdo, que pode ser feito a partir de qualquer pasta que tenha permissão de leitura.

Como a pasta de moléculas de referência envolve milhares de arquivos, apesar de serem pequenos, o *upload* leva muito mais tempo do que se fosse enviado um único arquivo `.zip`. Por isso, pode ser coveniente compactar a pasta como `.zip` antes de enviar, conforme exemplificado nas Figuras A.22 e A.23.

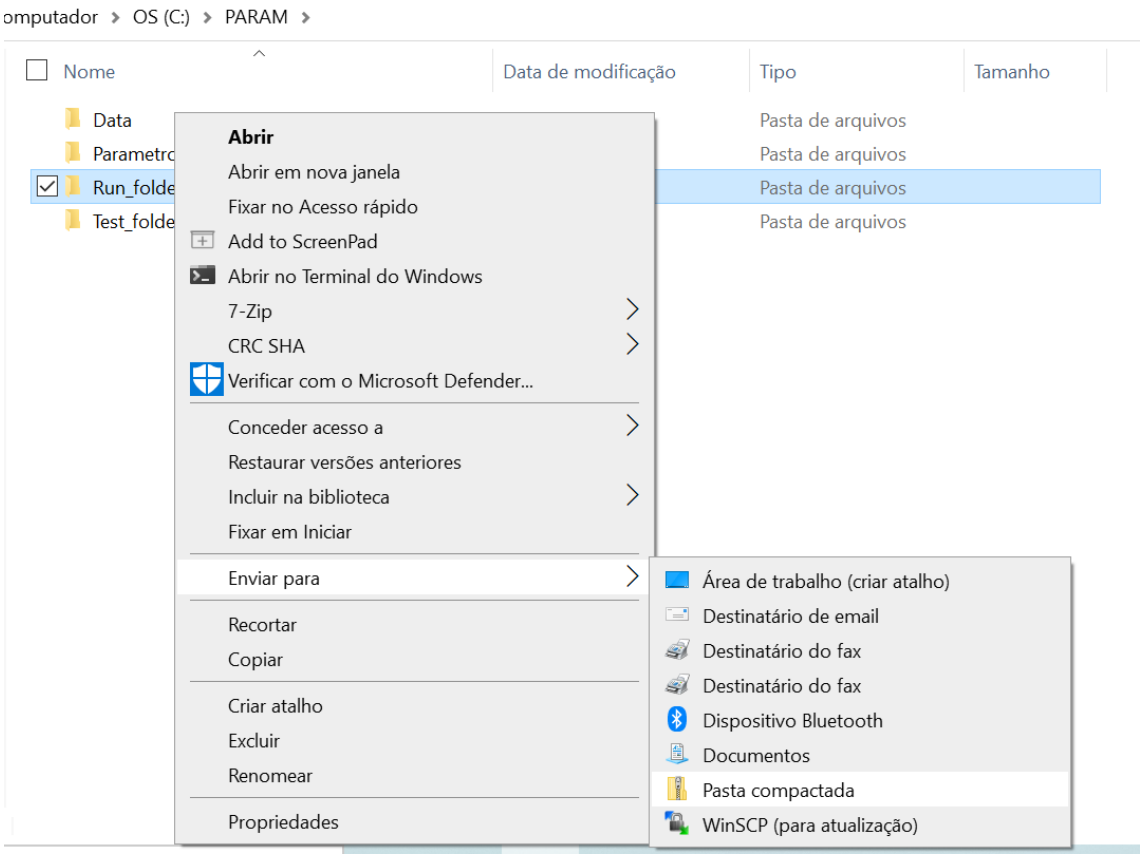


Figura A.22: Exemplo de compactação de pasta no Windows.

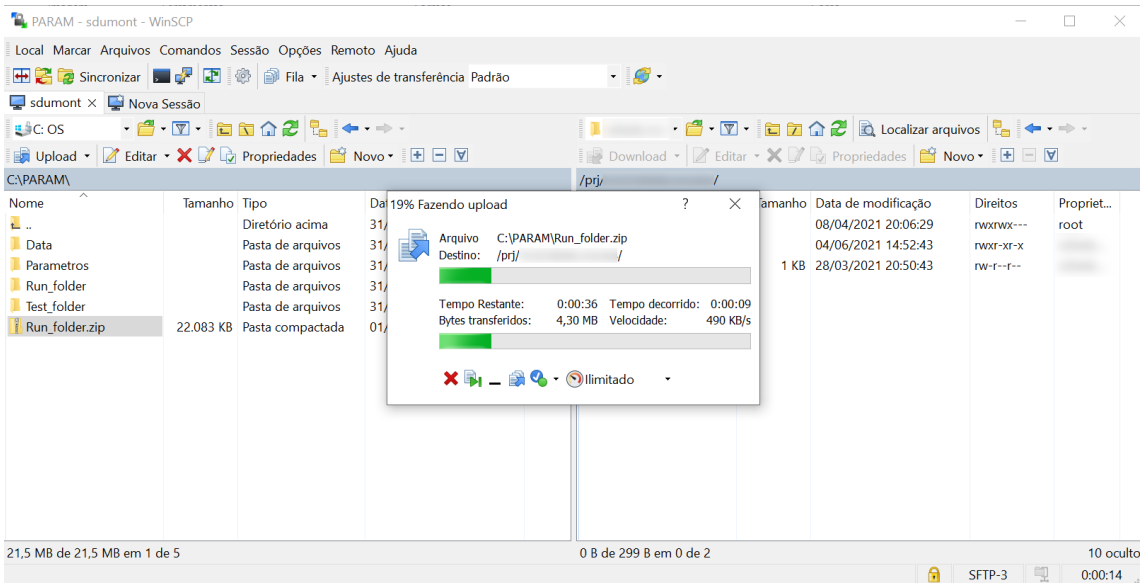


Figura A.23: Exemplo de *upload* de arquivo compactado.

O WinSCP é uma ferramenta muito prática para transmissão de arquivos, mas tem apenas essa utilidade. Não é possível executar programas a partir dele. Portanto, após o *upload* do arquivo compactado, é necessário entrar no terminal do servidor (por exemplo, com o PuTTY) para descompactar o arquivo e também para

executar o PARAM. O comando para descompactar um arquivo `.zip` é `unzip`. No caso do exemplo acima, depois de enviar o arquivo `Run_folder.zip`, deve-se entrar no terminal do servidor e executar: `unzip Run_folder.zip`

A.3 Preparação do ambiente

Após copiar os arquivos do PARAM para a máquina onde ele será executado, é necessário preparar o ambiente para que a execução do arquivo com extensão `.exe` seja possível. Um teste para verificar se o ambiente já está pronto é tentar fazer uma execução simples do programa, conforme exemplo abaixo.

```
$ cd Run_folder/
$ ./PARAM_Linux.exe
./PARAM_Linux.exe: error while loading shared libraries: libmkl_intel_lp64.so:
    cannot open shared object file: No such file or directory
```

Tentativa de execução do PARAM sem o módulo da Intel carregado.

Ao reproduzir os passos acima, o símbolo `$` não deve ser digitado, pois já aparece automaticamente antes dos comandos no terminal, sendo aqui apresentado somente para ficar visualmente mais próximo do que aparece no terminal. O comando `cd` é usado para mudar de diretório e `./` é o diretório atual. A terceira linha (que não se inicia com `$`) é a mensagem de resposta à execução do segundo comando.

Para acionar um executável, é preciso digitar o endereço (completo ou relativo) até ele, seguido ou não de parâmetros adicionais, e teclar *Enter*. Se a pasta em que ele está for a pasta atual, é necessário incluir o `./` antes. Neste exemplo, uma biblioteca necessária não foi encontrada. A solução para esse caso será apresentada a seguir. Se a mensagem fosse "`Permission denied`", seria necessário mudar o `PARAM_Linux.exe` para que ele tivesse permissão de execução. Nesse caso, ainda na pasta `Run_folder`, execute o comando `chmod 744 PARAM_Linux.exe` (que dá permissão de escrita e execução somente ao proprietário do arquivo) e tente novamente.

Para solucionar o problema da biblioteca não encontrada, no terminal do servidor, digite `module load intel` e pressione a tecla *Tab* duas vezes. Isso irá apresentar quais módulos da Intel estão disponíveis. No Linux, a tecla *Tab* autocompleta com as opções disponíveis. Se houver apenas uma opção que comece com o texto inicial já digitado, o restante já é preenchido logo na primeira vez em que a tecla é pressionada. Se houver mais opções, a primeira vez não apresenta nada, mas a segunda vez apresenta todas as opções disponíveis.

Por exemplo, no supercomputador Santos Dumont, uma execução desse procedimento teve o seguinte resultado:

```
$ module load intel
intel-opencl/2017     intel_psxe/2016     intel_psxe/2018     intel_psxe/DISABLED-2020
intel-opencl/2018     intel_psxe/2017     intel_psxe/2019
```

Listagem de módulos da Intel.

Por isso, uma boa escolha neste caso é usar este comando:

```
$ module load intel_psxe/2019
```

Carga do módulo da Intel.

Pois esse é o módulo ativo mais recente da Intel.

Após carregar esse módulo, o PARAM pode ser executado. Uma nova tentativa de execução deve exibir uma mensagem diferente:

```
$ ./PARAM_Linux.exe
Enter input file (without .dat or .mop)
```

Execução bem sucedida do PARAM, porém sem dados de entrada.

Isso significa que a execução funcionou, mas está faltando informar um arquivo com as informações que serão utilizadas. Logo, o ambiente está preparado para a execução.

A.4 Execução do PARAM

Antes da execução definitiva do PARAM, é preciso ajustar o arquivo `.dat`. O conteúdo desse arquivo é explicado no tópico 3.3.1. No caso dos arquivos e pastas apresentados na sessão A.2, o conteúdo do arquivo `rm2_2020_CHNO_running.dat` deve ser semelhante a este:

```
am1 params=rm2_params.inicio.txt ref=./inputs_rm2_param/new-chnops-sexte_2015-
    renamed deriv fine CYCLES=6000
 USS                            H
 BETAS                          H
 ZS                             H
 ALP                            H
 GSS                            H
 FN11                           H
 ⋮
 FN12                           O
 FN22                           O
 FN32                           O
end
(1a,2a,4a)-bicyclo[2_2_2]oct-5-ene-2-carbonitrile.mop
(1a,2b,4a)-bicyclo[2_2_2]oct-5-ene-2-carbonitrile.mop
(1-methylpropyl)-benzene.mop
 ⋮
```

Arquivo: `rm2_2020_CHNO_running.dat`

A lista de arquivos `.mop` pode seguir qualquer ordem, desde que todos estejam na pasta informada no campo `ref`.

O comando para execução do PARAM (após entrar no diretório `Run_Folder`) deve ser o seguinte:

```
$ nohup ./PARAM_Linux.exe rm2_2020_CHNO_running.dat &
```

Comando para execução do PARAM.

Após executar, pode-se fazer a combinação *Ctrl C* para cancelar o acompanhamento e executar outros comandos. O `&` ao final evita que essa ação também interrompa imediatamente o processo (isto é, a execução do programa). E o comando `nohup` do Linux significa *no hang up*, que impede a interrupção quando o usuário desloga do servidor.

Logo após a execução, outros arquivos são gerados na pasta, com o mesmo nome do arquivo `.dat`, mas com outras extensões. O arquivo mais importante para o acompanhamento é o de extensão `.out`, que acumula informações continuamente, tornando-se eventualmente muito grande.

Para confirmar que a execução está funcionando, utilize o comando `ll`, que é uma forma reduzida do comando `ls -l` (caso o primeiro não funcione, utilize o segundo). Os arquivos da pasta serão listados, assim como seus tamanhos em *bytes*. Se o processo estiver funcionando corretamente, duas execuções seguidas desse comando vão mostrar um aumento na quantidade de bytes do arquivo `.out`.

Caso ocorra algum problema, verifique o arquivo `nohup.out`, que recebe todas as mensagens que apareceriam diretamente no terminal se o comando `nohup` não tivesse sido utilizado.

A.5 Análise dos dados

Durante a execução, o comando abaixo permite obter apenas a parte relevante do arquivo `.out`, isto é, o valor da função resposta e o número do ciclo. Dessa forma, é possível identificar o número do ciclo com menor função resposta calculada.

```
$ grep UNSIGNED -B 3 rm2_2020_CHNO_running.out
```

Comando para extração dos dados relevantes de um arquivo `.out`.

Por exemplo, os dois primeiros ciclos de uma das execuções tiveram a seguinte saída como resposta a esse comando:

```
                    TOTALS    9502569.347  7363.122    607.63  1414778.294  10925318.392
 1    AVERAGE ERROR          980     0.69130  −0.26   102  0.18     1367
 1    ROOT MEAN SQUARE ERROR =       8.63       0.75        0.61       22.75
 1  UNSIGNED AVE. ERROR      =       5.209      0.506       0.416      12.007
—

                    TOTALS     188781.138  9450.393   1102.19   386395.949   585729.668
 2    AVERAGE ERROR          980    −1.87130   0.05   102  0.40     1367
 2    ROOT MEAN SQUARE ERROR =      11.34       0.85        0.79       21.15
 2  UNSIGNED AVE. ERROR      =       7.644      0.616       0.545      10.646
—
```

Dados extraídos dos primeiros dois ciclos de uma execução.

O número à esquerda que se repete por três vezes é o número do ciclo, e o último número da linha de totais é o valor da função resposta (que é a soma quadrada ponderada de todos os erros encontrados).

Caso haja muitos ciclos, de forma que esse resultado não fique completo no terminal, é possível resolver esse problema colocando toda a saída em um arquivo por meio de uma pequena adaptação no comando:

```
$ grep UNSIGNED -B 3 rm2_2020_CHNO_running.out > totais.txt
```

Extração dos dados para um arquivo `.txt`.

O funcionamento do comando segue a seguinte lógica: o comando `grep` obtém todas as linhas que têm a palavra *UNSIGNED*, e a instrução `-B 3` faz com que sejam incluídas também as 3 linhas anteriores (a letra B significa *before*), alcançando assim a linha de totais.

A numeração do ciclo e o valor da função resposta correspondente podem ser incluídos em uma planilha para facilitar a identificação do ciclo que obteve menor função resposta. Em seguida, os parâmetros correspondentes podem ser encontrados, pois eles também são listados no arquivo `.out`, pouco depois de uma linha que tem o número do ciclo e a expressão "*START SSQ:*". Como as linhas são posteriores à expressão procurada, utiliza-se `-A` (*after*). Para 77 parâmetros, é suficiente obter as 78 linhas seguintes. Então, por exemplo, se o número do ciclo é 49, é usado este comando:

```
$ grep " 49 START SSQ:" -A 78 rm2_2020_CHNO_running.out
```

Extração dos parâmetros usados em determinado ciclo.

A saída tem este formato:

```
 49 START SSQ:      314404.75      313836.17   GRAD:      6056612.1        47049.4
Parameter    Element      New value          Change              ...      Gradient
   USS          H       −12.22704761     −0.00759233                     106878.70
   BETAS        H        −5.54126324      0.00129800                    −177535.81
   ZS           H         1.03390680     −0.00021124                     100194.31
   ALP          H         3.32063642      0.00043995                     397931.10
   ⋮
```

Extração dos parâmetros usados em determinado ciclo.

A.6 Validação estatística

Após descobrir os parâmetros com menor função resposta, deve ser feita uma validação estatística, usando outro conjunto de moléculas, para que seja avaliado se o erro calculado continua pequeno. O procedimento para a execução é semelhante ao da reparametrização.

O arquivo `.dat` deve ter o início semelhante a este:

```
am1 large params=rm2_params.otimizado.txt ref=./test_folder survey
 USS                            H
 BETAS                          H
 ZS                             H
 ALP                            H
 GSS                            H
 FN11                           H
 ⋮
```

Arquivo `.dat` para validação estatística.

A opção *large* significa que mais informações vão ser retornadas. E *survey* indica que é uma validação estatística e, como tal, não faz uma reparametrização, apenas calcula a função resposta e os erros médios por propriedade, em resposta a um único conjunto de parâmetros.

Como são retornados muito menos dados do que durante a reparametrização, o procedimento de extração dos resultados também é mais simples, podendo ser realizado manualmente.

Apêndice B

Guia para uso do supercomputador Santos Dumont

Os exemplos de execução apresentados no Apêndice A, apesar de utilizarem o SDumont, não funcionam daquela forma nessa máquina. O SDumont tem regras e um sistema de filas. Os processos iniciados diretamente, fora desse padrão, são automaticamente encerrados cerca de 2 minutos após o início da execução.

É necessário copiar os arquivos utilizados na execução para uma partição específica, chamada *Scratch*, e preparar um *job* que irá para uma fila, tendo um tempo limitado de execução.

Além disso, o próprio acesso à máquina precisa da utilização de uma VPN (*Virtual Private Network*), por isso são necessários passos adicionais antes dos passos de acesso remoto. Esse acesso só é liberado após o envio de um formulário de cadastro de usuário, que pode levar semanas para aprovação.

B.1 Acesso à VPN

Após a aprovação do cadastro de usuário, a equipe do SDumont envia um e-mail com instruções de como acessar a VPN. São fornecidos 3 PDFs com instruções de instalação, para Linux, MAC e Windows. Nas instruções do Windows, é preciso ficar atento, pois há uma seção direcionada ao Windows 7 e outra ao Windows 8 ou 10. Siga atentamente os passos da seção correspondente ao seu sistema operacional. São necessárias algumas instalações e reinicializações do computador.

B.1.1 Instalação da VPN no Windows 10

No caso do Windows 10, versão 2004 ou posterior, a instalação exige um passo adicional, descrito ao final do documento. Por isso, nesse caso, é mais adequado começar por essas instruções, caso contrário será necessário desinstalar todos os

softwares instalados e instalar novamente incluindo o passo adicional. Como essa parte está um pouco mais confusa, ela será detalhada aqui.

Conforme indicado nas instruções, devem ser instaladas as ferramentas *Winfix* e *DNE*, com links providos no PDF. Em seguida, instalar o CiscoVPN Cliente. Para processadores de 64 bits, é fornecido o link para um arquivo que, ao ser executado, extrai o seu conteúdo para uma pasta determinada pelo usuário. Por padrão, esse campo aparece preenchido com um diretório temporário, conforme figura abaixo.

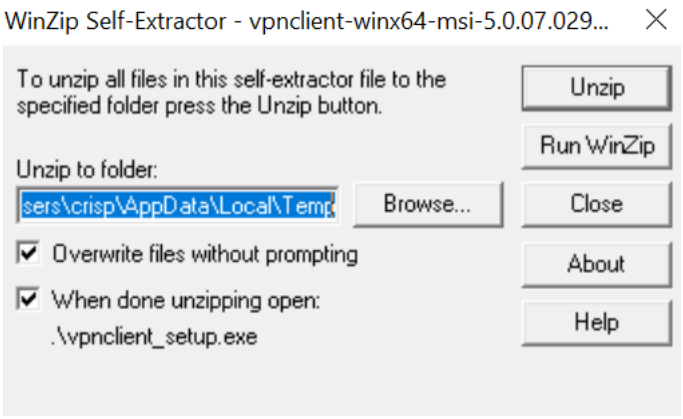


Figura B.1: Extração dos arquivos do cliente VPN sugerindo diretório temporário.

O campo deve ser modificado para `C:\VPN`.

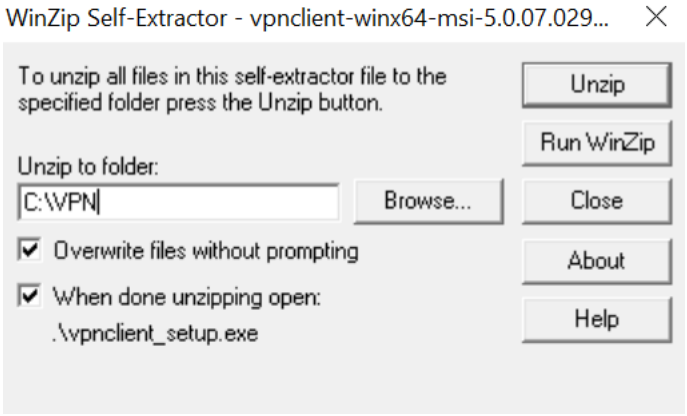


Figura B.2: Extração dos arquivos do cliente VPN, com a pasta corrigida.

Ao clicar em *Unzip*, os arquivos serão extraídos para essa pasta e essa janela de extração pode ser fechada. No caso de processadores de 32 bits, o link é de um arquivo `.zip`, o qual também deve ser extraído para a pasta `C:\VPN`.

Nesse momento, deve ser executado o passo adicional sugerido nas instruções. Deve ser instalado o *Cisco VPN Client Fix*. Somente depois disso, deve-se abrir a pasta `C:\VPN` e executar o arquivo `vpnclient_setup.msi`.

Após reiniciar o computador, é preciso fazer uma modificação no registro do Windows, conforme indicado na seção "Erros Conhecidos" das instruções. Clique na

barra de pesquisa, digite `regedit` e tecle *Enter*.

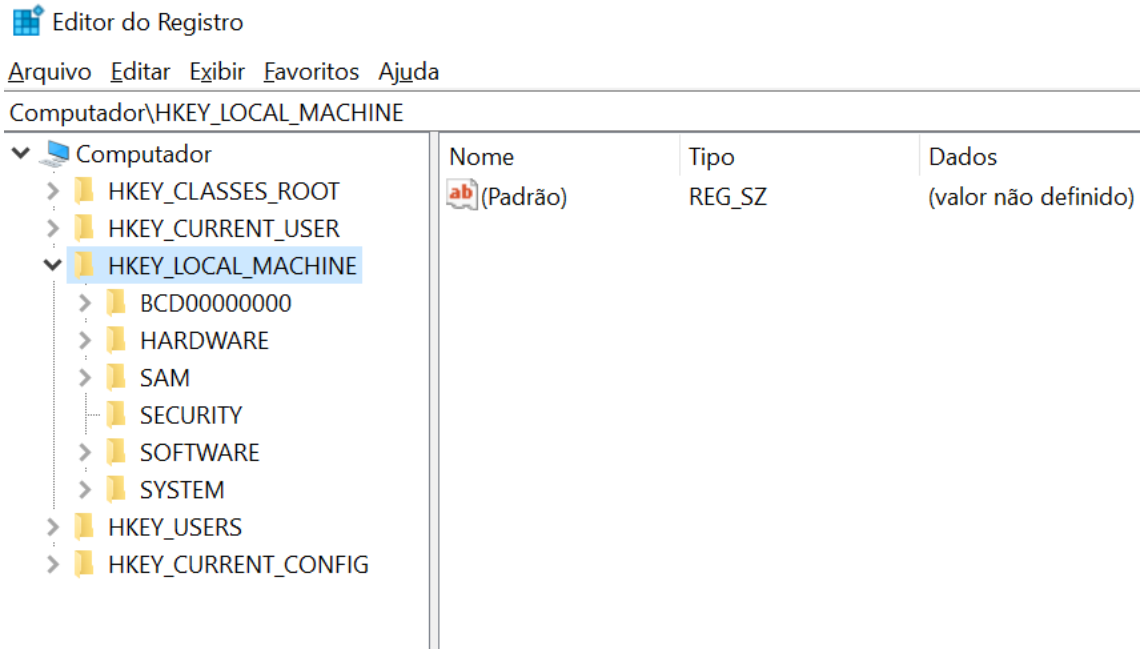


Figura B.3: Exibição do Editor do Registro.

Na árvore à esquerda, selecione a pasta `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CVirtA` e dê um duplo clique em `DisplayName`.

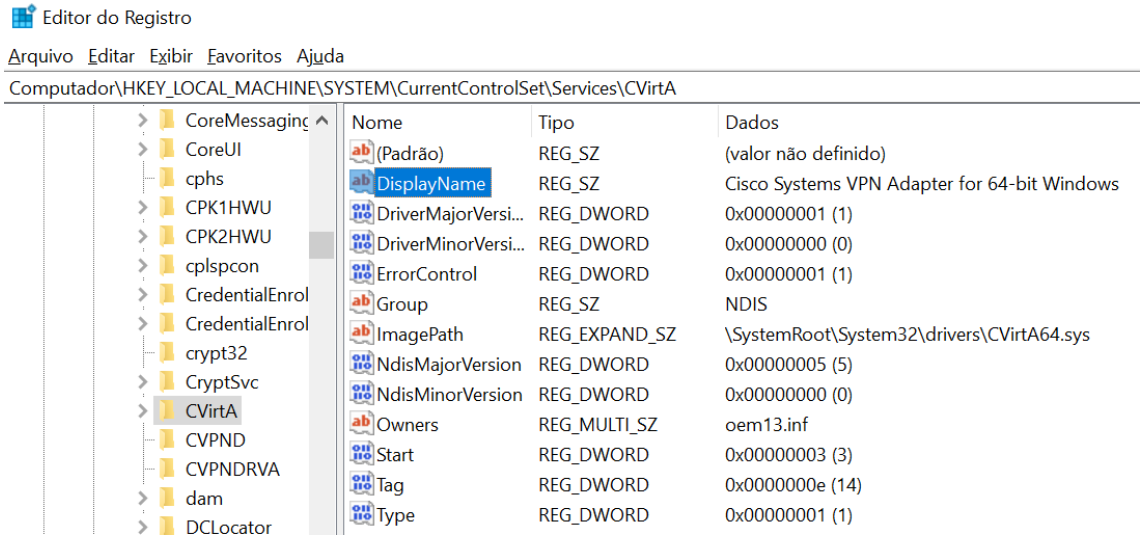


Figura B.4: Seleção da chave a ser modificada no Editor do Registro.

Caso o conteúdo do DisplayName tenha algum texto inicial como `@oem8.inf,%CVirtA_Desc%;` antes de `Cisco Systems`, remova-o, deixando apenas: `Cisco Systems VPN Adapter for 64-bit Windows` (para sistemas com 64 bits) ou `Cisco Systems VPN Adapter` (para 32 bits).

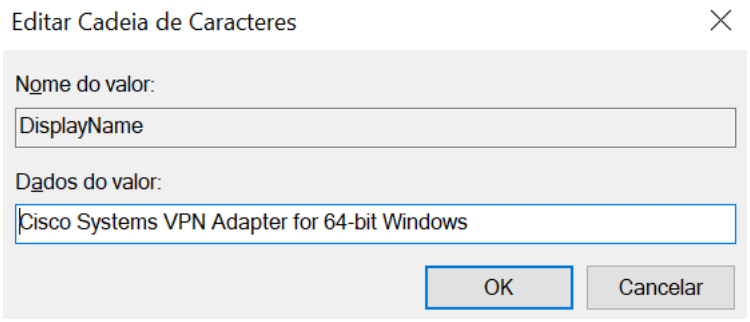


Figura B.5: DisplayName corrigido no registro.

Em seguida, clique em `OK` e feche o Editor do Registro. Isso conclui a instalação. Caso ainda ocorram problemas, a equipe do *Helpdesk SDumont* deve ser contatada.

B.1.2 Utilização da VPN

Após a instalação, a VPN é aberta ao pesquisar e selecionar o programa *VPN Client*. Siga atentamente as instruções enviadas, observando que, ao criar o perfil de conexão (o que só precisa ser feito uma vez), o nome e senha solicitados inicialmente são providos no PDF. Somente depois, ao clicar em *Connect*, deve ser informado o seu *login* e senha.

Uma vez conectado à VPN, torna-se possível o acesso remoto ao SDumont, que pode ser feito conforme apresentado no Apêndice A.

B.2 Execução de programas no SDumont

As regras para execução de programas estão descritas no manual do SDumont [45]. É uma máquina muito poderosa e permite a utilização de várias funcionalidades, mas aqui será descrito o mínimo necessário para uma execução simples, sem paralelismo, conforme foi executado neste trabalho.

Para executar um programa, é necessário submeter um *job*, que é uma rotina que fica agendada para execução, neste caso em uma fila.

Existem duas pastas que podem ser utilizadas: *HOME* (que fica na partição *prj*) e *SCRATCH* (partição *scratch*). O manual recomenda que fiquem na *HOME* os arquivos que precisam ser mantidos, e na *SCRATCH* os que forem necessários para a execução do *job*, inclusive os dados de entrada e saída, pois somente a partição *scratch* é acessível durante a execução do job. Após a execução, os dados resultantes que precisarem ser mantidos devem ser copiados para a *HOME*.

B.2.1 Cópia dos arquivos para a pasta *SCRATCH*

Ao entrar no terminal, a primeira pasta apresentada é a *HOME*, também representada por ~. Tanto a *HOME* quanto a *SCRATCH* estão armazenadas como variáveis de ambiente do usuário, sendo acessíveis acrescentando antes delas o símbolo $, conforme exemplo abaixo:

```
$ echo $SCRATCH
/scratch/nomeprojeto/nome.sobrenome
$ echo $HOME
/prj/nomeprojeto/nome.sobrenome
```

Exibição do conteúdo das variáveis de ambiente *HOME* e *SCRATCH*.

Portanto, para copiar a pasta `Run_Folder` de *HOME* para *SCRATCH* e depois entrar na *SCRATCH*, é possível utilizar os comandos abaixo, iniciando em *HOME*:

```
$ cp -rf Run_Folder/ $SCRATCH/
$ cd $SCRATCH/
```

Cópia de pasta de *HOME* para *SCRATCH*.

B.2.2 Preparação e inicialização do *job*

O gerenciador de filas do SDumont é o Slurm. Existem diversas filas disponíveis, listadas no manual [45]. Para submeter um *job* para uma dessas filas, é necessário criar um arquivo de texto com extensão `.srm`, como no exemplo abaixo, adaptado a partir de outros exemplos do manual.

```
#!/bin/bash
#SBATCH --nodes=1                          #Numero de Nos
#SBATCH --ntasks-per-node=1                #Tarefas por No
#SBATCH --ntasks=1                         #Total de tarefas
#SBATCH -p cpu_long                        #Fila a ser utilizada
#SBATCH -J rm2-rafaela-OUT19-48            #Nome job
#SBATCH --time=744:00:00                   #Tempo limite 31 dias

#Exibe os nos alocados para o Job
echo #SLURM_JOB_NODELIST
nodeset -e #SLURM_JOB_NODELIST

cd #SLURM_SUBMIT_DIR

#Configura os compiladores
module load intel_psxe/2019

#acessa o diretorio onde o executavel esta localizado
cd $SCRATCH/Run_folder/
```

```
#Configura o executavel
EXEC=$SCRATCH/Run_folder/PARAM_Linux.exe

#exibe informacoes sobre o executavel
/usr/bin/ldd $EXEC

srun -n $SLURM_NTASKS $EXEC rm2_2020_CHNO_running_OUT19_params.48.dat
```

Arquivo: `rodarOUT19_48.srm`.

No exemplo acima, foi escolhida a fila *cpu_long*, que é a que permite maior tempo de execução. O tempo total também foi ajustado para o máximo (31 dias), caso contrário o limite seria ajustado automaticamente para metade desse tempo. O número de nós (*nodes*) e de tarefas por nó (*ntasks-per-node*) foram definidos como 1, já que o PARAM não tem paralelismo, e o total de tarefas (*ntasks*) deve ser o produto dos dois anteriores, por isso também foi 1. O nome do *job* poderia ser qualquer um que facilitasse a identificação.

Após uma inicialização comum aos *jobs*, o módulo da Intel é carregado e é feita uma mudança para o diretório `Run_folder`. O endereço do executável é salvo em uma variável de ambiente *EXEC* e finalmente é gerada a solicitação de execução, por meio do comando `srun` seguido do número total de tarefas e em seguida o mesmo comando que seria executado normalmente: o executável seguido do arquivo `.dat`.

Para submeter o `job`, utiliza-se o comando `sbatch`.

```
$ sbatch rodarOUT19_48.srm
```

Submissão do *job*.

B.2.3 Acompanhamento do *job*

Depois que o *job* é submetido, a execução não é imediatamente iniciada. Ele fica em uma fila e pode demorar dias para ser atendido, dependendo da fila. Mais de um *job* pode ser submetido, mas, na experiência deste trabalho, no máximo dois rodavam simultaneamente.

Para acompanhar a situação dos *jobs* na fila, pode ser usado o comando `squeue`, conforme exemplo abaixo:

```
$ squeue -u nome.sobrenome -l
```

Verificação do status dos *jobs* de determinado usuário.

Após o `-u`, informa-se o usuário que submeteu os *jobs*, para que sejam listados todos os *jobs* do usuário. Duas das informações apresentadas são o tempo decorrido

desde o início da execução e o tempo limite. Para verificar o tempo previsto para um *job* agendado ser iniciado, pode ser acrescentado `--start` ao comando:

```
$ squeue -u nome.sobrenome -l --start
```

Verificação do horário previsto para o início da execução dos *jobs* de um usuário.

Esse comando também apresenta a razão pela qual o *job* ainda não foi iniciado, se for o caso.

Depois que a execução é iniciada, também é possível acompanhar seu andamento pelos arquivos de saída. No caso do PARAM, deve-se acessar a pasta `$SCRATCH/Run_folder/` e verificar o arquivo `.out` correspondente ao *job* em execução.