Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Diferenciação Automática

Autores: Nilvan Peres Costa e Peniel Etèmana Désirez-Jésus Zannoukou

Orientador: Prof. Dr. John Lenon Cardoso Gardenghi

Brasília, DF
2023

Nilvan Peres Costa e Peniel Etèmana Désirez-Jésus Zannoukou

Diferenciação Automática

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. John Lenon Cardoso Gardenghi

Brasília, DF

2023

Nilvan Peres Costa e Peniel Etèmana Désirez-Jésus Zannoukou
Diferenciação Automática/ Nilvan Peres Costa e Peniel Etèmana Désirez-Jésus Zannoukou. – Brasília, DF, 2023-
55 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. John Lenon Cardoso Gardenghi

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. Spectral Projected Gradient. 2. Diferenciação Automática. I. Prof. Dr. John Lenon Cardoso Gardenghi. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Diferenciação Automática

CDU 02:141:005.6

Nilvan Peres Costa e Peniel Etèmana Désirez-Jésus Zannoukou

Diferenciação Automática

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 11 de Dezembro de 2023:

Prof. Dr. John Lenon Cardoso Gardenghi
Orientador

Prof. Dr. Bruno César Ribas
Convidado 1

Prof. Dr. Edson Alves da Costa Júnior
Convidado 2

Brasília, DF
2023

Agradecimentos

Agradeço primeiramente a Deus todo poderoso por ter me dado uma saúde perfeita ao longo dessa grande jornada, a sabedoria nos momentos mais épicos (altos e baixos dessa vida). Ninguém disse que seria fácil, todavia, com a presença Dele eu pude chegar até onde estou chegando e acredito fortemente que é só o começo de grandes realização na minha vida.

Aos meus pais **ao Sr.° Telesphore Zannoukou e à Sr.ª Mireille Azavonon** que me apoiaram nessa tomada de decisão minha, ir continuar meus estudos fora e de tornar um dos meus sonhos uma realidade. Àos meus irmãos: **Louanges Fesch Soudéa Ermeline Zannoukou e Archange Marie Michel Midokpè Zannoukou** que sempre estiveram ao meu lado mesmo estando longe. Como um dos ditados bem popular do meu país: **Longe dos olhos, perto do coração «Loin des yeux, près du coeur»**

Como dizia o **Frei Jaime «Família não é questão de sangue, mas de quem quer segurar sua mão quando você mais precisa»**, sendo assim meu eterno agradecimento à família **Fernandes da Silva Filhos** e principalmente **ao Sr.° José Fernandes da Silva Filho e à Sr.ª Carmem de Oliveira Fernandes da Silva** que foram e são meus pais aqui no Brasil ao longo dessa jornada desde que Deus nos fez cruzar os caminhos. Infelizmente hoje o senhor não está mais entre nós fisicamente e não pude me ver colando grau mas de uma forma ou de outra eu bem sei que, do céu o verá, pois lá se tornou a nova morada do senhor. Eu falhei, sim, por não ter tido essa conquista nas mãos antes que o senhor partisse. Todavia fico feliz pelo senhor ter combatido o bom combate do início até o fim e guardou a fé..[...]. Meu eterno gratidão fica registrado aqui.

Gostaria de expressar minha sincera gratidão ao meu grande amigo **Nilvan Peres Costa** por ter aceitado este desafio de embarcarmos juntos no mundo da otimização. Sua colaboração e dedicação foram fundamentais ao longo dessa jornada, tornando-a ainda mais significativa e enriquecedora. A troca de conhecimentos e experiências compartilhadas foi de imensa importância para o desenvolvimento deste trabalho. Agradeço imensamente sua amizade e apoio ao longo dessa jornada acadêmica.

Agradeço ao meu orientador **Prof°. Dr. John L. Cardoso Gardenghi** por ter me ajudado no amadurecimento e na expressividade das minhas ideias, fazendo-me enxergar mais distante, e expandindo, mesmo antes desse trabalho final, a minha visão de Engenharia de Software. Agradeço aos meus verdadeiros amigos que deixei no meu país, os que eu pude fazer ao longo dessa jornada aqui também, para sempre estarão no meu coração.

Agradeço a Deus por ter me ajudado ao longo de todo o curso, por me dar força em todas as vezes que pensei em desistir. Minha gratidão aos meus pais **José Nilvan Costa** e **Clarinda Peres Guimarães**, porem terem me proporcionado uma educação de qualidade e me apoiarem ao longo da minha vida acadêmica. Aos meus irmãos **Marcos Peres Costa** e

Ana Claúdia Peres Costa, meus maiores exemplos de estudos e disciplina que tive na minha vida e que me ajudaram a ingressar a ingressar em uma Universidade Pública.

Quero agradecer a família **Alves**, por ter me acolhido em Brasília, e me proporcionar um segundo lar. Especialmente a minha namorada **Amanda Alves** pelo apoio incondicional, e por ser meu principal refúgio nos meus piores momentos ao longo da minha caminhada.

Por fim, queria expressar minha gratidão ao meu grande companheiro nessa jornada **Peniel Etèmana**. Por ter me dado a oportunidade de construir um TCC em conjunto. Por toda a troca de conhecimento e feedbacks, que certamente me ajudaram a evoluir, e por ter tornado esse trabalho acadêmico mais leve com seu entusiamo, bom humor e principalmente uma fé inabalável.

"E o Senhor respondeu-me assim: «Escreve esta visão, grava-a em tabuinhas, para que ela possa ser lida facilmente; porque há ainda uma visão para um termo fixado, ela se aproxima rapidamente de seu termo e não falhará. Mas, se tardar, espera-a, porque ela se realizará com toda a certeza e não falhará."

(Bíblia Sagrada, Habacuc 2, 2-3)

Resumo

A Diferenciação Automática (DA), é um conjunto de técnicas para avaliar derivadas, de funções matemáticas diferenciáveis a partir de um programa que calcule seu valor num determinado ponto. Este trabalho visa importar uma biblioteca de DA em um algoritmo clássico de otimização, o Gradiente Espectral Projetado (SPG). A intenção é automatizar a rotina responsável pelo cálculo das das derivadas de primeira ordem, eliminando a necessidade de configuração manual por parte do usuário. A pesquisa foi dividida em duas partes: a definição de uma biblioteca DA adequada, a partir da análise de algumas das bibliotecas de DA disponíveis na comunidade, e a integração dessa técnica ao algoritmo SPG. O resultado obtido é uma versão do SPG simplificada, que dispensa o cálculo das derivadas de primeira ordem, tornando seu uso mais simples do ponto de vista de implementação do problema de otimização a ser resolvido.

Palavras-chave: diferenciação automática. SPG. otimização.

Abstract

Automatic Differentiation (AD) is a set of techniques for evaluating derivatives of differentiable mathematical functions from a program that calculates their value at a certain point. This work aims to import an AD library into a classic optimization algorithm, the Projected Spectral Gradient (SPG). The intention is to automate the routine responsible for calculating first-order derivatives, eliminating the need for manual configuration by the user. The research was divided into two parts: the definition of a suitable AD library, based on the analysis of some of the AD libraries available in the community, and the integration of this technique into the SPG algorithm. The result obtained is a simplified version of the SPG, which dispenses with the calculation of first-order derivatives, making its use simpler from the point of view of implementing the optimization problem to be solved.

Key-words: automatic differentiation. SPG. optimization.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MGH	Moré-Garbow-Hillstrom
DA	Diferenciação Automática
AD	Automatic differentiation
UnB	Universidade de Brasília
SPG	Spectral Projected Gradient
ADOL-C	Automatic Differentiation by OverLoading in C++
TAPENADE	Tools for Algorithm PErformance ENhAncEmnt and DEbugging
MGP	Método do Gradiente Projetado

Lista de símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

Sumário

1 Introdução

A Diferenciação Automática (DA) consiste em um conjunto de técnicas que se baseiam na regra da cadeia e são utilizadas para calcular as derivadas de uma função de forma precisa e eficiente (BAYDIN et al., 2018). Existem diversas áreas de aplicação para a Diferenciação Automática, incluindo otimização de funções, controle de processos, modelagem matemática, aprendizado de máquina, entre outros. No entanto, é no ramo de otimização de funções não lineares que a DA tem maior destaque (GRIEWANK; WALTHER, 2003). Há duas técnicas de DA: a *diferenciação automática direta* e a *diferenciação automática reversa* (BARTHOLOMEW-BIGGS et al., 2000).

Na diferenciação automática direta, a função é representada como uma combinação de operações elementares, como adição, multiplicação, exponenciação, entre outras. Cada operação elementar é definida por uma regra matemática simples, que permite calcular a derivada da função em relação às suas variáveis independentes. Essas regras são combinadas em uma estrutura de dados chamada grafo de computação, que permite calcular a derivada da função em um ponto específico (BAYDIN et al., 2018).

Em contrapartida, na diferenciação automática reversa, a função é avaliada em uma direção reversa, partindo de sua saída em direção às variáveis de entrada. Essa abordagem permite calcular as derivadas parciais de uma função em relação a cada uma de suas variáveis independentes simultaneamente, com um número mínimo de avaliações da função. Essa técnica revela-se especialmente útil ao lidar com funções que possuem muitas variáveis independentes ou quando a função tem várias entradas e apenas uma saída (CORLISS et al., 2002).

Como mencionado anteriormente, a otimização desempenha um papel fundamental na computação, levando ao desenvolvimento de diversos algoritmos para abordar esse tipo de desafio. Um desses métodos notáveis é o SPG (*Spectral Projected Gradient* - SPG)[1], conhecido por minimizar funções suaves em um conjunto fechado e convexo. O algoritmo SPG realiza iterações projetando o gradiente de uma função em um subespaço que respeita as restrições do problema. Apesar de ser considerado um método lento, devido ao uso exclusivo de derivadas de primeira ordem, destaca-se por sua simplicidade de implementação e eficácia em problemas de larga escala (BIRGIN; MARTíNEZ; RAYDAN, 2000).

O SPG foi originalmente implementado em Fortran 77. De acordo com Birgin, Martínez e Raydan (2001), para utilizar o pacote, o usuário deve fornecer três rotinas: uma para avaliar a função objetivo, outra para avaliar seu gradiente e uma terceira para projetar um ponto arbitrário na região viável do problema. Por sua simplicidade e eficácia prática, esse software é amplamente utilizado em diversas áreas, incluindo finanças (JUNIOR; PAMPLONA; SALOMON, 2014), melhoria de previsões de trilha de ciclones tropicais (ZHOU; TOTH, 2020), reconheci-

[1] <https://www.ime.usp.br/~egbirgin/tango/codes.php#spg>

mento de expressões faciais (BEJAOUI; GHAZOUANI; BARHOUMI, 2019) e otimização na resolução de imagens (LIMA, 2012). Pela sua simplicidade e ampla aplicação, escolheu-se o SPG como método de otimização para importar uma biblioteca de diferenciação automática, com o objetivo de eliminar a necessidade do usuário fornecer uma rotina para calcular o gradiente da função.

Diante do exposto, o desafio é, objetivar as pesquisas acerca das bibliotecas que utilizam a Diferenciação Automática, selecionar a mais adequada para ser utilizada no SPG. Isso se deve ao fato de que a configuração manual das rotinas pelo usuário pode ser complexa e suscetível a erros. A biblioteca de DA assumirá a responsabilidade de calcular os gradientes, eliminando a necessidade do usuário de configurar essa rotina. Isso resultará na automação desse aspecto, facilitando o uso do solver.

1.1 Problema

A primeira etapa deste trabalho consistiu em compreendermos como poderíamos usar a diferenciação automática para auxiliar na automatização das rotinas de cálculo. Isso é particularmente relevante, considerando que o desenvolvimento manual de código para avaliar as derivadas parciais de uma função é um processo tedioso e propenso a erros (BARTHOLOMEW-BIGGS et al., 2000). Apesar de ser uma técnica amplamente utilizada, persistem dúvidas e desafios significativos em relação ao seu uso.

A segunda etapa deste trabalho envolve a seleção de uma biblioteca que contemple a implementação do algoritmo de Diferenciação Automática (DA), a ser posteriormente integrada ao software SPG. Dado que o SPG é um método de primeira ordem, ou seja, lida com derivadas de primeira ordem da função objetivo, a aplicação de técnicas de DA torna-se relevante para o projeto. Essa escolha visa proporcionar ao usuário a conveniência de não precisar calcular manualmente as derivadas parciais de uma função específica, que consequentemente resulta na redução do tempo gasto em configurações manuais, facilitando o uso do solver para o usuário.

Diante disso, a questão de pesquisa desse trabalho é:

Como integrar diferenciação automática num método de otimização como o SPG?

1.2 Objetivos

O objetivo deste trabalho é implementar uma estratégia de Diferenciação Automática (DA) no algoritmo SPG. Para alcançar esse objetivo, serão realizados os seguintes passos:

1. Fazer um levantamento das bibliotecas que implementam DA.
2. Avaliar a popularidade das bibliotecas de DA na literatura.
3. Integrar uma biblioteca de DA na implementação do SPG.

4. Validar a implementação do SPG com DA por meio de experimentação numérica.

1.2.1 Procedimentos Metodológicos

Com o propósito de alcançar o objetivo almejado, será executada uma revisão bibliográfica sobre a técnica de Diferenciação Automática (DA), suas aplicações e implicações para a otimização, incluindo uma análise das diferentes bibliotecas de DA existentes, avaliando suas características e adequação para o propósito deste trabalho.

1.3 Organização do Trabalho

Este trabalho de conclusão de curso está organizado em cinco capítulos, cada um abordando uma etapa do processo de pesquisa.

- **Capítulo 1 - Introdução:** neste capítulo são apresentados o contexto do trabalho, o problema que foi estudado e os objetivos deste trabalho.
- **Capítulo 2 - Referencial Teórico:** são apresentadas as principais técnicas de Diferenciação Automática, suas aplicações e implicações para a otimização, as principais bibliotecas de DA, e como implementar e integrar essa técnica em solvers. Também foi descrito sobre o software de Gradiente Espectral Projetado, além de explicar em síntese a projeção ortogonal.
- **Capítulo 3 - Implementação da Diferenciação Automática no SPG:** é descrito o processo de integração da técnica de Diferenciação Automática (DA) no algoritmo de Gradiente Espectral Projetado (SPG). Foram abordados os principais pontos relacionados à estrutura do SPG, detalhando como a DA foi incorporada ao algoritmo. Foram apresentados os objetivos e motivações para essa implementação, além de discutir as etapas e desafios envolvidos no processo. Ao final do capítulo, são destacadas as contribuições e os resultados obtidos com essa importação da biblioteca de DA junto ao solver SPG, reforçando que esse processo facilita na utilização do solver, pois não precisará configurar manualmente as derivadas de primeira ordem do SPG.
- **Capítulo 4 - Análise dos Resultados:** é apresentada a avaliação dos resultados obtidos a partir da implementação da Diferenciação Automática no SPG.
- **Capítulo 5 - Discussão e conclusão** é apresentada uma síntese dos principais resultados alcançados. Considerações finais sobre a eficácia das metodologias empregadas, a aplicabilidade dos resultados e as perspectivas para pesquisas futuras. Enfatizamos a importância deste trabalho na ampliação do entendimento sobre o tema estudado e como pode servir de base para futuros estudos e aplicações práticas.

2 Referencial Teórico

No processo de realização da revisão bibliográfica, foram adotadas estratégias para identificar fontes de informação relevantes para o tema em questão. Para isso, foram utilizadas palavras-chave específicas, como: DA, AutoDiff, AD, Automatic Differentiation, SPG, AD packages, optimization, entre outras. Após isso, delimitou-se um período de tempo específico, para ter fontes relativamente recentes sobre o tema. Além disso, foram consultadas as principais bases de dados acadêmicas, como Scopus, Siam, Periódico Capes, ACM Digital Library e Google Scholar visando obter uma ampla cobertura da literatura disponível. A revisão bibliográfica adotada neste trabalho seguirá uma estrutura de revisão narrativa, na qual serão apresentadas as diversas áreas de pesquisa abrangidas pela revisão. Isso inclui explorar o histórico do tema, analisar as aplicações práticas relacionadas e identificar as tendências recentes na área de estudo. Essa abordagem permite uma análise abrangente e contextualizada do conhecimento existente, proporcionando uma visão mais completa do estado atual da pesquisa.

Ao adotar essa metodologia, é esperado uma compreensão aprofundada sobre o tema, desafios e lacunas de conhecimento na área em questão. Isso possibilita embasar de forma sólida as discussões e argumentações apresentadas ao longo do trabalho, além de fornecer um contexto relevante para a formulação de hipóteses e a definição dos objetivos da pesquisa. Portanto, o referencial teórico desempenha um papel fundamental nesta etapa inicial do trabalho, provendo um panorama abrangente e atualizado das pesquisas existentes, auxiliando na identificação de hiatos a serem exploradas, contribuindo para o embasamento teórico e científico do estudo.

2.1 Gradiente Espectral Projetado (SPG)

O SPG (BIRGIN; MARTÍNEZ; RAYDAN, 2001) é uma extensão do método do gradiente projetado, que foi originalmente desenvolvido para otimização com restrições de igualdade e desigualdade. Essa técnica adiciona uma etapa de projeção do gradiente no conjunto viável do problema, em cada iteração, para garantir que o próximo ponto esteja dentro do conjunto viável. É um método eficaz para miniminizar funções suaves em conjuntos convexos, e foi introduzido em um estudo pioneiro de *Barzilai e Borwein* Birgin, Martínez e Raydan (2014).

Para entender melhor o SPG, é importante notar que a ideia central do método é a atualização da solução com base em um passo de tamanho variável ao longo da direção do gradiente, mas também incorporando uma projeção dentro das condições do problema. Como já foi destacado, o algoritmo foi projetado para lidar com problemas de otimização que incluem restrições, isso significa que há limitações em que as variáveis de decisão devem ser mantidas dentro de determinados intervalos. Essas restrições podem ser definidas pelas características do problema ou pelas regras impostas pelas leis físicas ou regulatórias. Vale a pena ressaltar que o método possui duas etapas principais, o cálculo do gradiente da função objetivo e projeção do

passo do gradiente dentro do conjunto definido pelas restrições do problema.

No cálculo do gradiente da função objetivo, a sua magnitude indica a taxa de mudança da função em relação às variáveis do problema. Em cada iteração do algoritmo, o gradiente é calculado e usado para definir a direção da busca pela solução ótima. O algoritmo então segue na direção oposta do gradiente, procurando minimizar a função objetivo. Por outro lado, a projeção do passo é necessária para garantir que a solução se mantenha dentro das restrições do problema (BIRGIN; MARTíNEZ; RAYDAN, 2000).

O algoritmo SPG possui várias vantagens, destacando-se nos seguintes aspectos:

- **Versatilidade:** Além de sua aplicação geral, o SPG demonstra versatilidade ao ser aplicável a uma ampla classe de problemas de otimização, incluindo, mas não se limitando a, problemas de programação quadrática (BIRGIN; MARTíNEZ; RAYDAN, 2000), problemas de programação semidefinida e problemas de programação não-linear.
- **Boas Propriedades de Convergência:** O SPG é conhecido por apresentar uma rápida taxa de convergência, aproximando-se da solução em um número limitado de iterações (BIRGIN; MARTíNEZ; RAYDAN, 2000). Sua capacidade de convergência global faz com que seja eficaz na busca de minimizadores locais aproximados da função objetivo.
- **Facilidade de Implementação:** A implementação do SPG é relativamente fácil e pode ser adaptada para lidar com diferentes tipos de restrições (LLAVE, 2012). Isso contribui para sua acessibilidade e uso em diferentes contextos de otimização.
- **Efetividade para Restrições de Caixa:** O SPG destaca-se por sua efetividade em problemas de otimização com restrições de caixa, comuns em muitas aplicações práticas. A projeção utilizada no SPG, explorando a propriedade da caixa, contribui para sua efetividade, procurando encontrar soluções que satisfaçam as restrições estabelecidas (BIRGIN; MARTÍNEZ; RAYDAN, 2001).

Além disso, o algoritmo SPG é amplamente utilizado em muitas áreas de pesquisa e aplicação, incluindo ciência da computação, engenharia, finanças, biologia, entre outras. Abaixo serão apresentados alguns exemplos de problemas onde o algoritmo SPG foi aplicado, que comprovam como o software está sendo utilizado em diferentes contextos:

- **Melhorias de previsões de trilha de ciclones tropicais**: O software foi aplicado com o objetivo de minimizar o erro de previsão da trajetória de ciclone, o estudou apresentou uma melhora significativa na otimização das previsões (ZHOU; TOTH, 2020).
- **Otimização de portfólio de investimentos:** O SPG foi utilizado para resolver problemas de otimização de portfólio de investimentos com restrições de caixa em que o objetivo era maximizar o retorno do investimento sujeito a restrições de risco e diversificações (JUNIOR; PAMPLONA; SALOMON, 2014).

- **Reconhecimento de expressões faciais**: Foi utilizado para otimizar a função de perda do modelo de aprendizado de máquina usado para classificar as expressões faciais (BEJAOUI; GHAZOUANI; BARHOUMI, 2019).
- **Otimização de redes neurais:** O SPG foi usado em problemas de otimização de redes neurais com restrições de caixa em que o objetivo era ajustar os pesos e viés da rede para minimizar a função de perda sujeita a restrições de capacidade de processamento e armazenamento (KEERTHI et al., 2001).
- **Ajuste de modelos de regressão linear com restrições lineares**: O projeto foi aplicado para auxiliar um modelo de regressão linear sujeito a um conjunto de restrições nas variáveis de entrada (POLYAK; JUDITSKY, 1992).
- **Otimização na resolução de imagem**: O software foi utilizado para solucionar um problema de *Compressive Sensing*, que consiste em recuperar o sinal original a partir das amostras (LLAVE, 2012).
- **Otimização associados a máquinas de suporte vetorial**: O software SPG foi utilizado para otimizar na tarefa de encontrar o hiperplano ótimo que separa as duas classes de dados com a maior margem possível (CORES et al., 2008).
- **Cálculo de perturbações condicionais não-lineares ótimas**: Foi aplicado para encontrar uma perturbação ótima em um sistema não-linear, que minimiza a norma de uma função objetivo sob uma restrição dada (LIMA, 2012).

Para utilizar o SPG, um usuário precisa fornecer três rotinas que definem o problema: uma que calcule a função objetivo num determinado ponto, outra que calcule sua derivada de primeira ordem e outra que calcule a projeção de um ponto no conjunto viável do problema.

2.1.1 O algoritmo

De acordo com Birgin, Martínez e Raydan (2001) o método do Gradiente Espectral Projetado é usado para resolver o seguinte problema:

$$\begin{aligned} \min \quad & f(x) \\ \text{suj. a} \quad & x \in A, \end{aligned} \tag{2.1}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é uma função que possui pelo menos derivadas de primeira ordem contínuas e A é o conjunto de pontos viáveis para o problema.

A ideia do algoritmo é a seguinte: na iteração k, calculamos uma direção de descida[1] d^k e calculamos um tamanho de passo $t_k \leq 1$ para que $x^k + t_k d^k$ satisfaça alguma condição de decréscimo suficiente da função objetivo. A direção d^k é calculada com base na projeção ortogonal do negativo do gradiente da função objetivo em x^k no conjunto A, de tal forma a garantir ainda que o novo ponto $x^k + d^k \in A$.

[1] Uma direção de descida é uma direção ao longo da qual garante-se decréscimo da função.

2.1.2 Rotinas do software SPG

O projeto SPG inicialmente foi desenvolvido em Fortran 77, e C[2]. Para encontrar uma solução aproximada para o problema (2.1), o usuário deve fornecer três subrotinas que definem o problema:

- `subroutine evalf(n,x,f,flag)`, que recebe um vetor `x` de tamanho `n` e calcula o valor da função objetivo `f` avaliada em `x`,
- `subroutine evalg(n,x,g,flag)`, que recebe um vetor `x` de tamanho `n` e calcula o gradiente da função objetivo `g` avaliada em `x`,
- `subroutine proj(n,x,flag)`, que recebe um vetor `x` de tamanho `n` e calcula a projeção ortogonal de `x` no conjunto viáve

2.2 Diferenciação Automática

A Diferenciação Automática de acordo com Griewank e Walther (2008), também chamada de diferenciação algorítmica ou diferenciação computacional, consiste em um conjunto de técnicas computacionais, construídas com base na teoria do cálculo diferencial, que avaliam numericamente a derivada de uma função especificada por um programa de computador.

Em muitos problemas de computação científica, é comum a utilização de funções matemáticas que são definidas através de programas que constroem as funções a partir de uma série de operações matemáticas elementares. É possível aplicar o método de Diferenciação Automática (DA) para explorar o fato de que esses programas executam uma sequência de operações aritméticas elementares, tais como adição, subtração, multiplicação, divisão, entre outras, e também funções elementares como exponenciação, logaritmo, seno, cosseno, e outras. Dessa perspectiva, um programa pode ser visto como uma composição de várias funções primitivas que possuem derivadas conhecidas. Assim, o cálculo da derivada de uma função composta torna-se um exercício de aplicação da *regra da cadeia* de forma criteriosa, e partir do uso dessa técnica é possível calcular derivadas de ordem arbitrária automaticamente e com alta precisão (GRIEWANK; WALTHER, 2008).

Além de ser uma ferramenta fundamental para o cálculo de derivadas de funções compostas, a regra da cadeia é muito útil para a resolução de problemas em várias áreas da matemática e das ciências. Essa regra é baseada na ideia de que, ao derivar uma função composta, é necessário levar em consideração as variações de ambas as funções que a compõem. É por isso que a derivada de uma função composta é o produto da derivada da função externa pela derivada da função interna, como mencionado anteriormente.

[2] <https://www.ime.usp.br/~egbirgin/tango/downloads.php>

Por exemplo, vamos considerar uma função $f(x) = (x^2 + 3x - 1)^3$. Para calcular sua derivada, é necessário aplicarmos a regra da cadeia, considerando que a função externa é a função cúbica e a função interna é a expressão $(x^2 + 3x - 1)$:

$$f'(x) = 3(x^2 + 3x - 1)^2 \times (2x + 3)$$

Nesse caso, a derivada da função externa (cúbica) é igual a $3(x^2+3x-1)^2$, e a derivada da função interna (expressão) é igual a $2x + 3$. Ao multiplicar essas duas derivadas, obtém-se a derivada da função composta $f(x)$.

Portanto, a regra da cadeia é uma ferramenta essencial para o cálculo de derivadas de funções compostas e pode ser aplicada em vários contextos.

Um outro exemplo, suponhamos que queiramos calcular a derivada da função $f(x) = sin(x^2)$ que é uma função trigonométrica composta com uma função quadrática no ponto $x = a$. O método de DA representaria essa função como uma sequência de operações:

$$u = x^2$$

$$v = sin(u)$$

$$f = v$$

Então, seria possível aplicar a regra da cadeia para calcular a derivada da função f em a, usando as derivadas das funções intermediárias u e v:

$$f'(a) = v'(u(a)) \times u'(a) = cos(u(a)) \times 2a = 2a \times cos(a^2)$$

Assim, o método de DA permite calcular a derivada de qualquer função em um ponto específico, com precisão arbitrariamente alta, apenas aplicando a regra da cadeia em uma sequência de operações elementares.

É importante ressaltar que o método de DA não é o mesmo que diferenciação numérica, que usa diferenças finitas para estimar a derivada de uma função. A diferenciação automática é mais precisa e eficiente, já que utiliza informações exatas sobre as derivadas das funções intermediárias.

Existem duas sequências de operações básicas para o cálculo da diferenciação automática: a sequência direta (ou direta para frente ou modo direto) e a sequência reversa (ou reversa para trás ou modo reverso).

2.2.1 Modo Direto

O modo direto, também conhecido como sequência direta, é um método de diferenciação automática que calcula as derivadas de uma função em relação a todas as suas variáveis

independentes simultaneamente. Utilizado em problemas de otimização e ajuste de curvas, esse método avalia as operações da função na ordem em que aparecem na expressão, calculando derivadas parciais de forma recursiva. Uma vantagem é a precisão e eficiência ao calcular gradientes de funções multivariáveis. No entanto, é mais adequado para funções com um número limitado de variáveis independentes, devido ao aumento do custo computacional com o número de variáveis Griewank e Walther (2008).

2.2.2 Modo Reverso

De acordo com (GRIEWANK; WALTHER, 2008), o modo reverso, ou sequência reversa, é um método de diferenciação automática que calcula a derivada de uma função em relação a uma variável específica, mantendo as outras constantes. Útil em problemas de otimização e aprendizado de máquina, esse método avalia as operações da função na ordem inversa à sua expressão, calculando derivadas parciais de forma recursiva. Sua vantagem está na eficiência ao calcular gradientes em relação a muitos parâmetros, sendo útil em aprendizado de máquina para atualização iterativa de parâmetros. Entretanto, sua implementação pode ser mais complexa, especialmente em funções com operações não lineares ou não diferenciáveis, e pode demandar mais memória, limitando seu uso em conjuntos de dados extensos.

Diversos trabalhos na literatura têm aplicado o método de diferenciação automática para resolver problemas em diferentes áreas do conhecimento. Alguns exemplos incluem:

1. O objetivo desse trabalho de Lobao (2015) foi de investigar o potencial de algoritmos computacionais evolutivos, construídos a partir das técnicas de programação genérica, combinados com diferenciação automática, na obtenção de soluções analíticas, exatas ou aproximadas, para problemas de equações diferenciais ordinárias (EDO), parciais (EDP) e estocásticas.

2. O trabalho de **"Automatic Differentiation in Machine Learning: a Survey"** de Baydin et al. (2017) teve por objetivo de pesquisa a intersecção de AD e aprendizado de máquina. Ademais, o artigo fornece uma visão geral sobre como essa técnica de diferenciação é utilizada em várias tarefas, incluindo otimização, inferência estatística e modelagem.

3. Bartholomew-Biggs et al. (2000) **"Automatic Differentiation of Algorithms"** apresenta uma abordagem geral básicas para a diferenciação automática, conhecida como diferenciação direta, que permite calcular gradientes e hessianos de forma precisa e eficiente. Além disso, este trabalho também fornece uma análise teórica da precisão e complexidade computacional da diferenciação automática.

4. Outro estudo interessante é o de Griewank e Walther (2008) que discute a utilização de DA em conjunto de algoritmos genéricos para resolver problemas de otimização, essa junção pode melhorar significativamente a assertividade para encontrar a solução ótima.

A importação de uma biblioteca de diferenciação automática em um solver de otimização tem como objetivo simplificar o processo para o usuário, eliminando a necessidade de programar as derivadas da função objetivo. A técnica de DA permite o cálculo automático de derivadas de ordem arbitrária com alta precisão. Isso se mostra útil em uma variedade de aplicações, incluindo otimização, modelagem e simulação, controle de processos, entre outras.

2.3 Bibliotecas que implementam DA

As bibliotecas de Diferenciação Automática (DA) são ferramentas de software que fornecem suporte para calcular automaticamente derivadas de funções matemáticas. Essas bibliotecas são frequentemente usadas em problemas de aprendizado de máquina, otimização e modelagem matemática. Existem várias bibliotecas de software que implementam a técnica de DA e estão disponíveis para diferentes linguagens de programação, acessando ao site autodiff.org[3] pode-se observar uma gama de bibliotecas.

1. **CasADi** (ANDERSSON; ÅKESSON; DIEHL, 2012)
2. **OpenAD/F** (UTKE et al., 2008)
3. **Tapenade** (HASCOET; PASCUAL, 2013)
4. **JuliaDiff**[4]
5. **ADOL-C** O (GRIEWANK; JUEDES; UTKE, 1996a)

Como foi descrito acima, foram levantadas algumas bibliotecas de DA, e a primeira a ser escrita é a CasADi de (ANDERSSON; ÅKESSON; DIEHL, 2012), que é uma ferramenta de código aberto para otimização não linear e diferenciação algorítmica. A base fundamental da CasADi consiste em um framework simbólico que incorpora os modos direto e reverso de Diferenciação Automática (DA) em grafos de expressões, esse mecanismo é utilizado para construir gradientes, Jacobianas extensas e esparsas, além das Hessianas[5]. Na figura 1 podemos analisar as métricas de citações na base de dados Scopus:

[3] <https://www.autodiff.org/?module=Tools>
[4] <https://arxiv.org/abs/1607.07892>
[5] <https://web.casadi.org/>

Figura 1 – Métricas CasADi. Fonte: https://shorturl.at/tSZ39

Outra biblioteca que está crescendo é a Open/ADF (UTKE et al., 2008), é uma ferramenta de Diferenciação Automática (DA) de código aberto que se concentra na diferenciação de código Fortran. A sigla "OpenAD/F" significa "Open Source Automatic Differentiation for Fortran". Essa biblioteca é projetada para permitir o cálculo eficiente de derivadas de funções implementadas em código Fortran, tornando a diferenciação automática acessível para programas escritos nessa linguagem[6]. Observa-se na Figura 2, o comportamento das métricas de citações da biblioteca:

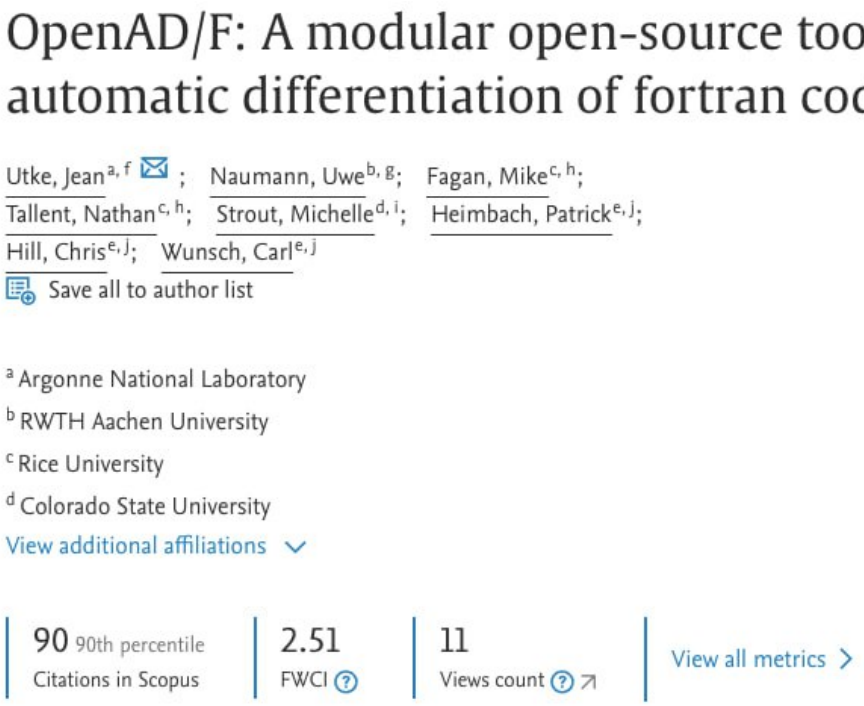

Figura 2 – Métricas OpenAD/F. Fonte: https://t.ly/yvCqT

[6] <https://www.anl.gov/mcs/openadf-automatic-differentiation-of-fortran-codes>

Agora uma biblioteca que está um pouco mais consolidada na comunidade é a Tapenade (HASCOET; PASCUAL, 2013). É mais uma biblioteca de Diferenciação Automática que permite calcular derivadas de forma acurada em programas escritos em Fortran, C e C++. A ferramenta oferece suporte a várias linguagens e é capaz de calcular derivadas de ordens superiores (HASCOET; PASCUAL, 2013). Observa-se na Figura 3, o comportamento das métricas da biblioteca:

Figura 3 – Métricas Tapenade. Fonte: https://t.ly/q75C1

Já a biblioteca JuliaDiff é escrita em Julia e suporta vários tipos de dados e estruturas de dados. O *ForwardDiff* é uma ferramenta que oferece métodos para calcular derivadas, gradientes, Jacobianas, Hessiana e derivadas de ordem superior de funções nativas do Julia (ou qualquer objeto chamável) usando a técnica de diferenciação automática de modo direto [7]. Observa-se na Figura 4, o comportamento das métricas de citações da biblioteca:

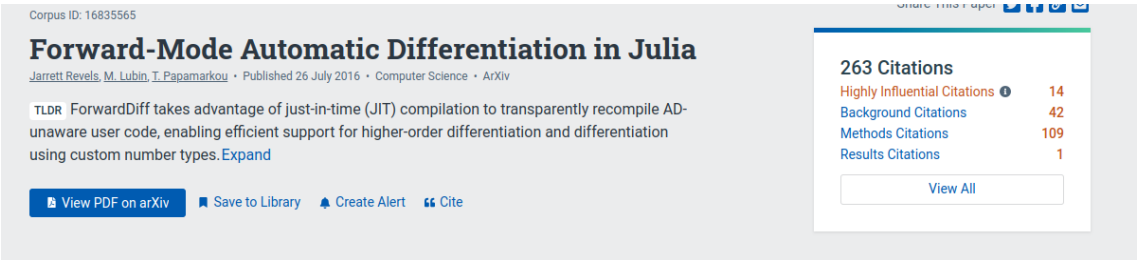

Figura 4 – Métricas JuliaDiff. Fonte: https://t.ly/46y1T

A biblioteca ADOL-C é uma ferramenta estabelecida na comunidade, tendo sido desenvolvida em C++ e C para facilitar a avaliação de derivadas de primeira e ordens superiores. As rotinas resultantes para avaliação de derivadas podem ser chamadas a partir de C, C++, Fortran ou qualquer outra linguagem que possa ser vinculada com C. Nesse site[8] de busca científica, podemos observar como a biblioteca é relevante conforme mostra a Figura 5:

[7] <https://github.com/JuliaDiff/ForwardDiff.jl>
[8] <https://dl.acm.org/doi/abs/10.1145/229473.229474>

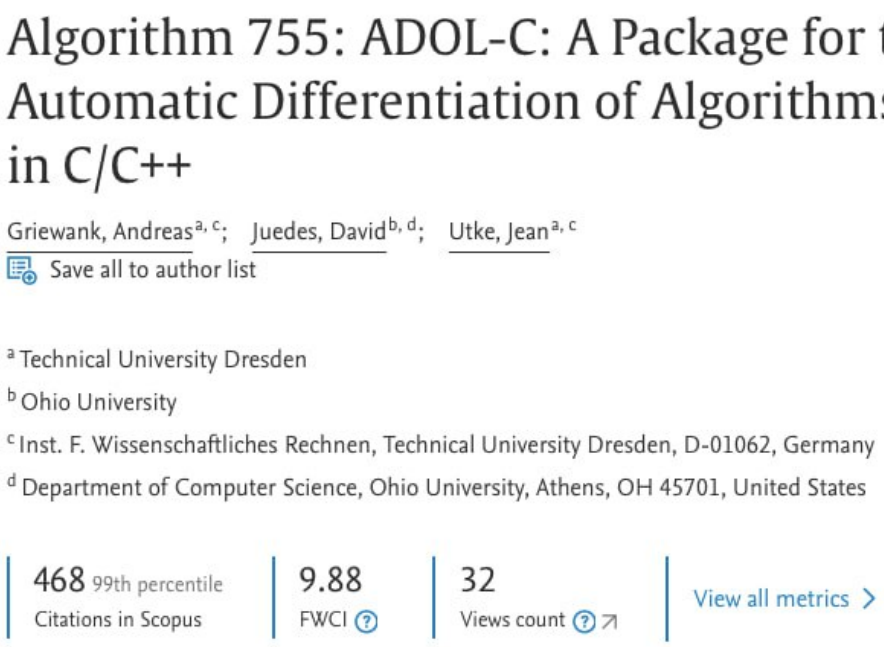

Figura 5 – Métricas ADOL-C. Fonte: https://t.ly/PKCds

Para determinar a biblioteca mais consolidada pela comunidade, realizaremos uma comparação das métricas apresentadas na Figura 6. A biblioteca mais citada será selecionada para a implementação em conjunto com o software SPG, pois indica que a mesma está há mais tempo consolidada na comunidade científica e, portanto, é mais provável que tenha uma ampla gama de materiais disponíveis e mais indivíduos ativos que possam prestar suporte.

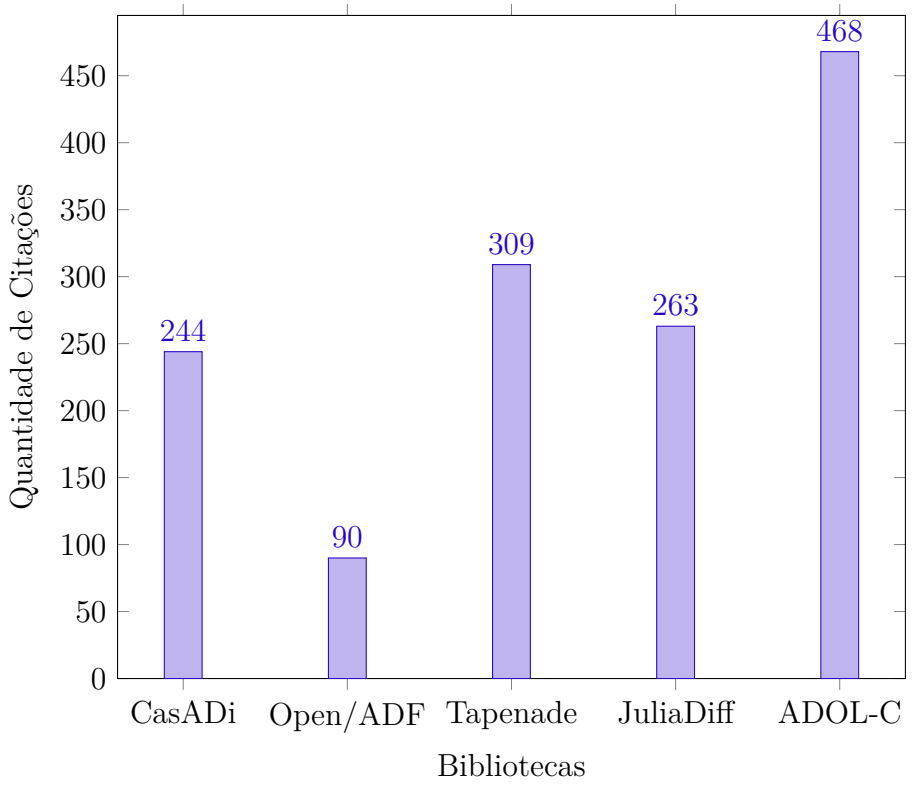

Figura 6 – Gráfico mostrando as bibliotecas e suas métricas em relação a quantidade de citações. Fonte: Autores

Essas bibliotecas oferecem uma variedade de funcionalidades para calcular derivadas automáticamente, e cada uma pode ser adequada para diferentes tipos de problemas e necessidades. É importante verificar a documentação e os exemplos de uso para determinar qual biblioteca é a melhor opção para o seu problema específico. Como a biblioteca ADOL-C foi a escolhida, a biblioteca será descrita com mais detalhes a seguir.

ADOL-C é um acrônimo para Automatic Differentiation by Overloading in C++, a essência da ferramenta é utilizar essa técnica de sobrecarga. Em contraste com abordagens de transformação de código-fonte, que geram novos trechos de códigos para calcular as derivadas, a sobrecarga de operadores (overloading) não gera um código-fonte intermediário (GRIEWANK; JUEDES; UTKE, 1996b). A biblioteca é open-source e gratuita para uso acadêmico e comercial, além disso a ferramenta possui algumas vantagens.

Uma dessas vantagens é que os valores numéricos dos vetores de derivadas são obtidos livres de erros de truncamento, e o programa de avaliação de função necessário requer apenas uma pequena quantidade de memória de acesso aleatório para avaliação da função objetivo. O cálculo das derivadas envolve uma quantidade considerável de dados, porém sempre previsível. A maior parte desses dados é acessada em ordem sequencial. Isso significa que, se necessário, podem ser automaticamente armazenados em arquivos externos para gerenciar o uso de memória de forma mais eficiente (GRIEWANK; JUEDES; UTKE, 1996b).

Como mencionado anteriormente, a técnica ADOL-C é baseada na sobrecarga de operadores. No entanto, é importante destacar que o conceito de variáveis ativas é fundamental para o funcionamento dessa técnica. Isto é, todas as variáveis que podem ser consideradas como quantidades diferenciáveis em algum momento durante a execução do programa devem ser do tipo ativo. Portanto, todas as variáveis que estão no caminho desde as variáveis de entrada, ou seja, as independentes, até as variáveis de saída, ou seja, as dependentes, precisam ser redeclaradas como do tipo ativo. Para esse propósito, o ADOL-C introduz o novo tipo de dado "adouble", cuja parte real é do tipo padrão "double", esses tipos de variáveis são essenciais na fase de gravação (WALTHER, 2009).

De acordo com Walther (2009) durante a fase de gravação (taping), o cálculo da derivada é realizado com base em uma representação interna da função. Essa fase inicia com a chamada da rotina "`trace_on`", fornecida pelo ADOL-C, e é finalizada com a chamada da rotina "`trace_off`" do ADOL-C. Todas as operações matemáticas que envolvem variáveis ativas e ocorrem entre as chamadas de função "`trace_on(tag,...)`" e "`trace_off(...)`" são registradas em um conjunto sequencial de dados chamado "tape".

Essa biblioteca suporta vários tipos de problemas de otimização, incluindo problemas de programação não-linear e problemas de equações diferenciais. A ferramenta também suporta diferenciação de ordem superior, o que significa que é possível calcular derivadas de ordem superior de uma função. Além das funcionalidades mencionadas anteriormente, a biblioteca ADOL-C também possui algumas características adicionais que a tornam atrativa para muitos usuários:

1. **Suporta diferenciação parcial:** Fornece suporte diferenciação parcial, o que significa que é possível calcular derivadas parciais de uma função. Isso pode ser útil em aplicações como análise de sensibilidade e projetos de experimentos.

2. **Suporta diferenciação vetor-por-escalar:** diferenciação vetor-por-escalar, o que significa que é possível calcular derivadas de funções que dependem de vetores e escalares. Isso pode ser útil em aplicações como otimização de sistemas dinâmicos e equações diferenciais parciais.

Outro beneficio da biblioteca que vale a pena mencionar é a capacidade de lidar com funções não-diferenciáveis, como mínimos locais, máximos locais, pontos de inflexão e singularidades. A biblioteca é intuitiva e de fácil uso, permitindo aos usuários realizar cálculos de derivadas com apenas algumas linhas de código.

Essa facilidade para utilizar a ADOL-C faz com que a mesma seja amplamente utilizada em diversas áreas, tais como otimização, controle, dinâmica de sistemas, inteligência artificial, entre outras. Vários trabalhos científicos recentes utilizam essa biblioteca para implementar algoritmos de Diferenciação Automática e aplicá-los em problemas reais. Dada a sua ampla utilização, é importante entender como utilizar a biblioteca ADOL-C para calcular derivadas.

De acordo com Griewank, Juedes e Utke (1996b) para calcular a derivada de uma função utilizando a biblioteca Adol-C, os seguintes passos são necessários. 1:

Inclusão de Cabeçalhos

- Certifique-se de incluir os cabeçalhos necessários para utilizar as funções da biblioteca Adol-C.
- Utilize a diretiva `#include <adolc.h>` para isso.

Definição da Região para DA

- Marque a seção ativa onde a DA será aplicada.
- Utilize os comandos `trace_on(tag)` para o início da seção ativa e `trace_off(file)` para o fim.

```
trace_on(1);
adouble *ax = new adouble[n];
for (int i = 0; i < n; i++)
    ax[i] <<= x[i];
```

Listing 1 – Cabeçalhos

```
af >>= *f;
delete[] ax
trace_off();
```

Listing 2 – Rodapé

Essas declarações definem a parte do programa para a qual uma representação interna será criada.

Seleção de variáveis ativas:

- Escolha o conjunto de variáveis que serão ativas na DA.
- Troque os tipos dessas variáveis de `double` ou `float` para `adouble`.

Inicialização de variáveis independentes:

- Selecione uma sequência de variáveis independentes e inicialize-as utilizando atribuições `«=` a partir de variáveis ou vetores passivos.

Seleção de variáveis dependentes:

- Selecione uma sequência de variáveis dependentes entre as variáveis ativas.
- Passe seus valores finais para variáveis ou vetores passivos através de atribuições `»=`.

A seguir no algoritmo 3, apresentamos um exemplo de como calcular a derivada de uma função simples usando a biblioteca Adol-C na linguagem de programação C++.

O objetivo desse código é demonstrar como usar a biblioteca ADOL-C para calcular a derivada de uma função. A biblioteca permite que se registre as variáveis independentes, calcule a função desejada e obtenha as derivadas parciais com facilidade. Neste exemplo, é calculada a derivada parcial da função $f(x) = (x^3 + 2x^2 + \sin(2x))$ em relação à variável x.

O resultado é impresso no console.A sáida do exemplo se apresenta da maneira a seguinte:

Derivada parcial em relação a x: 18.6927.

```cpp
#include <iostream>
#include <adolc/adolc.h>
#include <cmath>

int main()
{
  // Número de variáveis independentes:n  e dependentes:m
  const int n = 1;
  const int m = 1;
  double x = 2.0; // Valor da variável independente
  // Vetor de variáveis independentes
  double xad[n];
  xad[0] = x;
  // Vetor de variáveis dependentes
  double yad[m];
  trace_on(1);  // Iniciar gravação
  // Registrar variáveis independentes
  adouble ax;
  ax <<= xad[0];

  // Calcular função
  adouble af = pow(ax, 3) + 2.0 * pow(ax, 2) + sin(2.0 * ax);

  // Registrar variáveis dependentes
  af >>= yad[0];
  trace_off();     // Parar gravação
  double grad[n];  // Vetor de derivadas

  // Calcular derivadas
  gradient(1, n, xad, grad);
  std::cout << "Derivada parcial em relação a x:
  " << grad[0] << std::endl;

  delete[] xad;
  delete[] yad;
  return 0;
}
```

Listing 3 – Implementação em C++ para calcular a derivada de uma função com Adol-C

```
junior@junior-Q504UA:~/Documents/TCC/spg/github$ ./simple_example
Derivada parcial em relação a x: 18.6927
```

Figura 7 – Saida obtida no terminal. Fonte: Autores

Fazendo uso da mesma função num solver WolframAlpha[9] observamos na Figura 8 que os resultados são idênticos.

[9] <https://www.wolframalpha.com/>

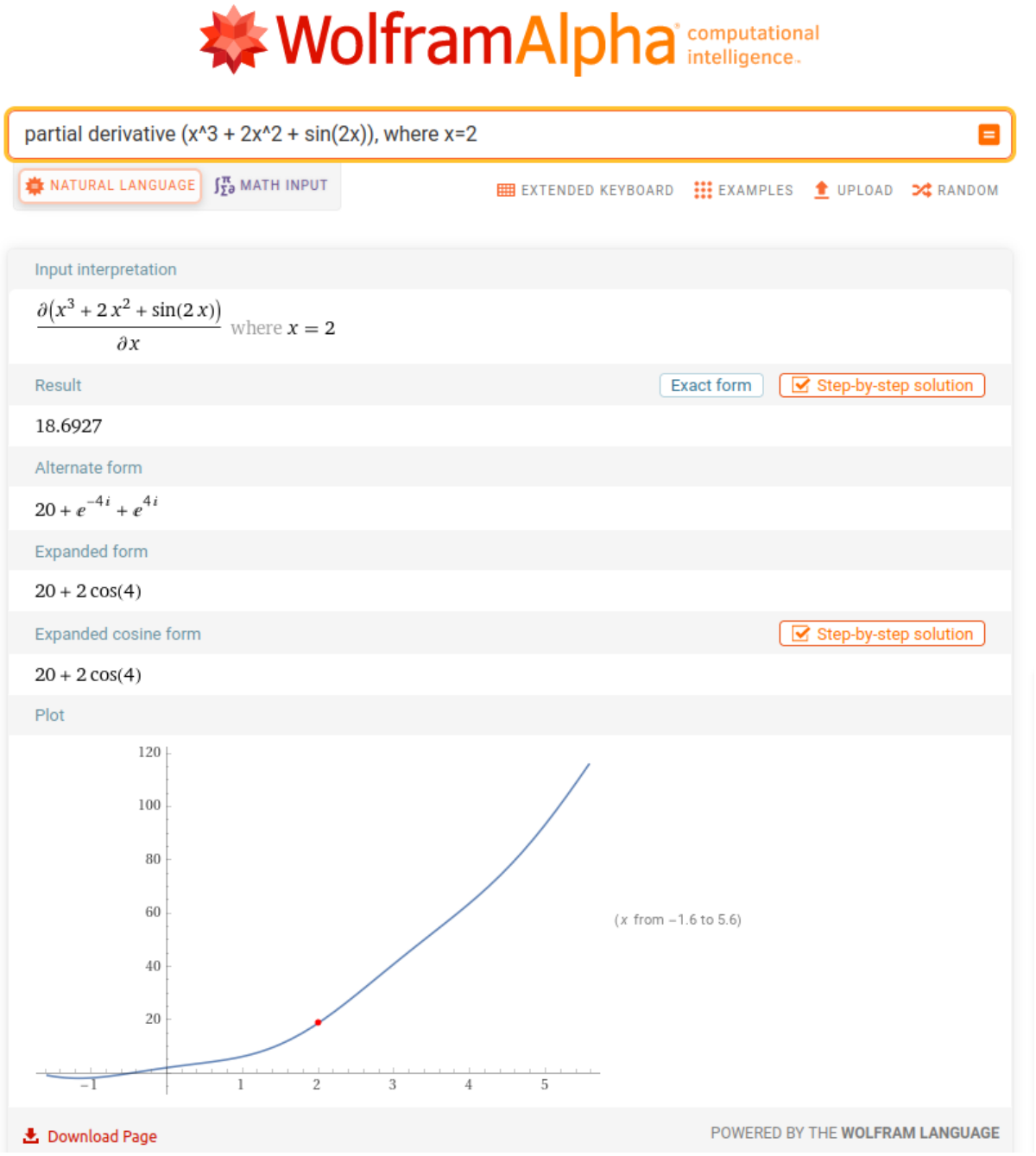

Figura 8 – Saida obtida num solver. Fonte: https://www.wolframalpha.com

Neste exemplo, a função a ser derivada é $f(x) = (x^3+2x^2+sin(2x))$, e queremos calcular sua derivada em relação a x no ponto $x = 2$. Primeiramente, definimos a função que retorna o valor da função para um dado valor de x. Em seguida, definimos as variáveis n e m para indicar o número de variáveis independentes e funções a serem derivadas, respectivamente. Definimos também o valor inicial de x e criamos as variáveis x e y como objetos adouble da biblioteca Adol-C.

Em seguida, utilizamos as funções $trace\_on$ e $trace\_off$ para definir a função a ser derivada e parar a gravação de operações, respectivamente. Por fim, utilizamos a função gradient para calcular a derivada da função em relação a x e imprimir o resultado na tela.

A biblioteca possui diversas outras funcionalidades, como suporte a funções. É possível encontrar mais informações e documentação sobre a biblioteca Adol-C no site oficial[10]. Para executar este código, é necessário instalar a biblioteca Adol-C e compilar o código usando um compilador C que suporta a Adol-C.

2.4 Considerações Finais

No capítulo atual, abordamos o Gradiente Espectral Projetado (SPG) e seu benefício em relação à Diferenciação Automática (DA) na simplificação do cálculo das derivadas parciais de problemas específicos. Exploramos diferentes bibliotecas que implementam a DA, com destaque para o ADOL-C, fornecendo uma explicação detalhada sobre seu funcionamento e como a biblioteca pode ser integrada ao SPG. Ao longo deste capítulo, discutimos os fundamentos teóricos do SPG e sua importância na otimização de problemas não lineares sujeitos a restrições convexas. Enfatizamos o papel da DA em fornecer as derivadas necessárias para otimizar as funções relevantes, eliminando a necessidade de realizar manualmente esses cálculos complexos.

Além disso, apresentamos uma análise das principais bibliotecas disponíveis para implementação da DA, destacando suas características, vantagens e desvantagens. Dentro desse contexto, o ADOL-C se destacou como uma opção robusta e amplamente utilizada, fornecendo uma interface amigável para incorporar a DA em algoritmos como o SPG. Ao combinar a DA com o SPG, esperamos simplificar o processo de otimização, permitindo que os usuários se concentrem na formulação do problema e na definição das restrições, enquanto a computação das derivadas é tratada de forma automática.

Este capítulo serve como uma introdução ao Gradiente Espectral Projetado (SPG) e à Diferenciação Automática (DA), estabelecendo um alicerce para o trabalho subsequente. Discutimos os principais conceitos e princípios subjacentes ao SPG e à DA, bem como suas aplicações, proporcionando ao leitor uma visão geral de suas funcionalidades. Com este conhecimento, servirá como base para explorar essas técnicas no decorrer do nosso trabalho.

[10] <https://github.com/coin-or/ADOL-C>

3 Implementação da Diferenciação Automática No SPG

Nesta seção, iniciamos a discussão sobre a implementação da Diferenciação Automática (DA) no SPG (Gradiente Espectral Projetado), abordando considerações essenciais. Exploraremos a estrutura atual do SPG e como a DA será incorporada a esse algoritmo de otimização não linear. Além disso, apresentaremos os objetivos e motivações que impulsionam a implementação da DA no SPG, fornecendo uma visão geral das etapas e desafios envolvidos nesse processo. Esta seção é fundamental para estabelecer o contexto e as bases necessárias para a compreensão da implementação da DA no SPG, que será discutida detalhadamente nos próximos tópicos.

3.1 Estrutura do SPG Atualmente

Para fazer o uso do software SPG os usuários devem fornecer algumas sub-rotinas para computar a função objetivo f(x), o gradiente de f(x) e as projeções de pontos arbitrários dentro de um conjunto(BIRGIN; MARTÍNEZ; RAYDAN, 2001). As funções responsáveis por isso são: `evalf, evalg e proj`. Essas funções definem o problema e são invocadas por SPG apropriadamente ao longo de sua execução.

O `evalf` recebe algumas informações essenciais, como o número de variáveis do problema e um conjunto de coordenadas que representam um ponto específico no espaço de soluções. Em seguida, a partir de iterações, a função evalf realiza cálculos e retorna o valor da função objetivo correspondente a esse ponto específico. A função evalf é responsável por analisar e quantificar a adequação de uma solução candidata em relação ao objetivo. Com base nos resultados fornecidos por essa função, o algoritmo SPG pode direcionar seus esforços para explorar regiões promissoras do espaço de soluções, buscando sempre melhorar o desempenho da função objetivo. Em suma, a função evalf desempenha um papel crítico ao fornecer informações essenciais para o algoritmo tomar decisões para otimizar o processo de busca pela solução ótima. A seguir, a assinatura dessa função:

```
void evalf (int n, double *x, double *f, int *flag);
```

É crucial ter conhecimento do gradiente da função objetivo. Nessa parte que entra a função evalg que nos fornece informações valiosas sobre a direção de maior crescimento da função objetivo em um determinado ponto do espaço. A função evalg recebe como entrada o número de variáveis do problema e as coordenadas de um ponto específico. Essa função evalg desempenha um papel muito importante ao fornecer informações sobre a direção que o algoritmo deve seguir para otimizar o desempenho da função objetivo. É graças a essa orientação fornecida pela função evalg que o algoritmo SPG consegue explorar de forma eficiente o espaço de soluções em busca da solução ótima. A seguir, a assinatura dessa função:

```
void evalg(int n, double *x, double *g, int *flag);
```

Por fim, mas não menos importante, temos a função proj. Essa função desempenha um papel crucial ao garantir que as soluções encontradas pelo algoritmo estejam dentro dos limites estabelecidos pelas restrições do problema em questão. A seguir, a assinatura dessa função:

```
void proj(int n, double *x, int *flag);
```

3.2 Agregação da Diferenciação Automática no SPG

A diferenciação automática foi adotada como uma abordagem para melhorar a automatização no processo de configurações de rotinas de otimizações. A principal motivação por trás dessa escolha foi a busca por um algoritmo que proporciona uma alta qualidade para lidar com o cálculo de derivadas parciais.

Ao aplicar a diferenciação automática, o software é capaz de lidar com o gradiente sem a rotina `evalg`. Isso significa que evitamos possíveis erros dos usuários a computar derivadas de equações complexas, além dos erros numéricos que podem surgir quando utilizamos métodos de diferenciação aproximada.

Dessa forma, ao empregar a diferenciação automática, podemos confiar em resultados de maior qualidade ao lidar com o gradiente. Essa maior qualidade no cálculo de derivadas parciais contribui para aprimorar a busca por soluções ótimas e otimizar o processo de otimização como um todo.

Em resumo, a adoção da diferenciação automática demonstrou que a qualidade da derivada parcial calculada por este método tem um impacto significativo no SPG. Isso não só elimina a necessidade do usuário configurar a rotina do gradiente, mas também evita possíveis erros humanos na programação dessa função, os quais poderiam atrapalhar ou até mesmo gerar resultados inconsistentes.

3.3 Experimento numérico

Após a integração da diferenciação automática no SPG, foram realizados testes e validações para verificar a corretude dessa implementação. Esses testes foram essenciais para garantir que a nova versão do algoritmo estivesse funcionando conforme o esperado. Foram comparados os resultados obtidos com a versão original do SPG e observou-se uma melhoria no desempenho e na capacidade do algoritmo em encontrar soluções ótimas em alguns cenários de problemas.

Os testes realizados envolveram diferentes aspectos. Primeiramente, foi analisada a convergência do algoritmo, ou seja, a capacidade de atingir uma solução ótima dentro de um número razoável de iterações. Também foram avaliadas a precisão das derivadas calculadas pela diferenciação automática e a comparação dos resultados com soluções conhecidas de problemas de otimização.

É importante ressaltar que, apesar dos benefícios trazidos pela diferenciação automática, existem algumas limitações e considerações adicionais a serem levadas em conta. Por exemplo, certas restrições podem estar presentes em relação às características das funções a serem otimizadas, como a continuidade ou a existência de derivadas de ordem superior. Além disso, a implementação da diferenciação automática pode exigir recursos computacionais adicionais, como memória e tempo de processamento. Em alguns problemas do conjunto de teste do MORÉ, Garbow e Hillstrom (1981) foram constatados um aumento na quantidade de iterações e avaliações funcionais necessárias para encontrar a solução ótima. Também vale ressaltar que esses testes foram programados tendo como base o código de John L. Cardoso Gardenghi que pode ser encontrado no github[1].

No entanto, mesmo com essas considerações, a agregação da diferenciação automática no SPG apresentou resultados interessantes em algumas funções, como a queda de iterações e redução do número de avaliações funcionais. Essa integração oferece uma abordagem mais automatizada, tendo em vista que é uma rotina a menos para o usuário configurar e eficaz para resolver problemas complexos de otimização, pois em todos os casos que convergiam sem a biblioteca, continuaram convergindo com o uso da mesma.

Os resultados obtidos são apresentados no gráfico de barras nas Figuras 9 a 11, onde comparamos o número de iterações que foram necessárias para cada problema do conjunto convergir a uma solução. As barras azuis representam os resultados obtidos sem o uso do ADOL-C, enquanto as barras verdes correspondem aos resultados obtidos com o uso do ADOL-C. É necessário destacar que, apenas os problemas que convergiram que farão parte dos gráficos. Analisando os resultados, é possível comparar as diferenças de desempenho entre os dois métodos para cada problema específico.

[1] <https://github.com/johngardenghi/mgh>

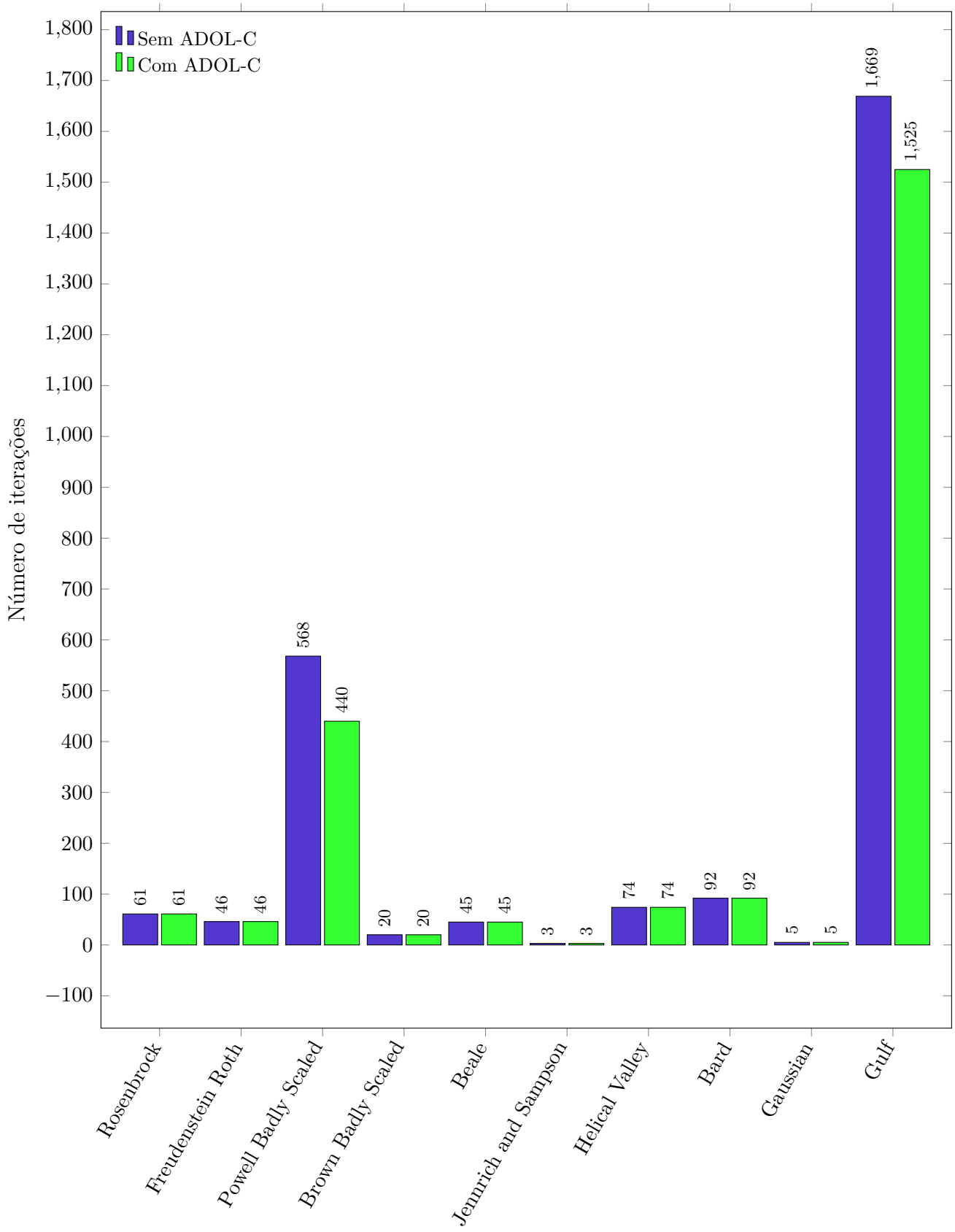

Figura 9 – Gráfico de iterações Parte I. Fonte: Autores

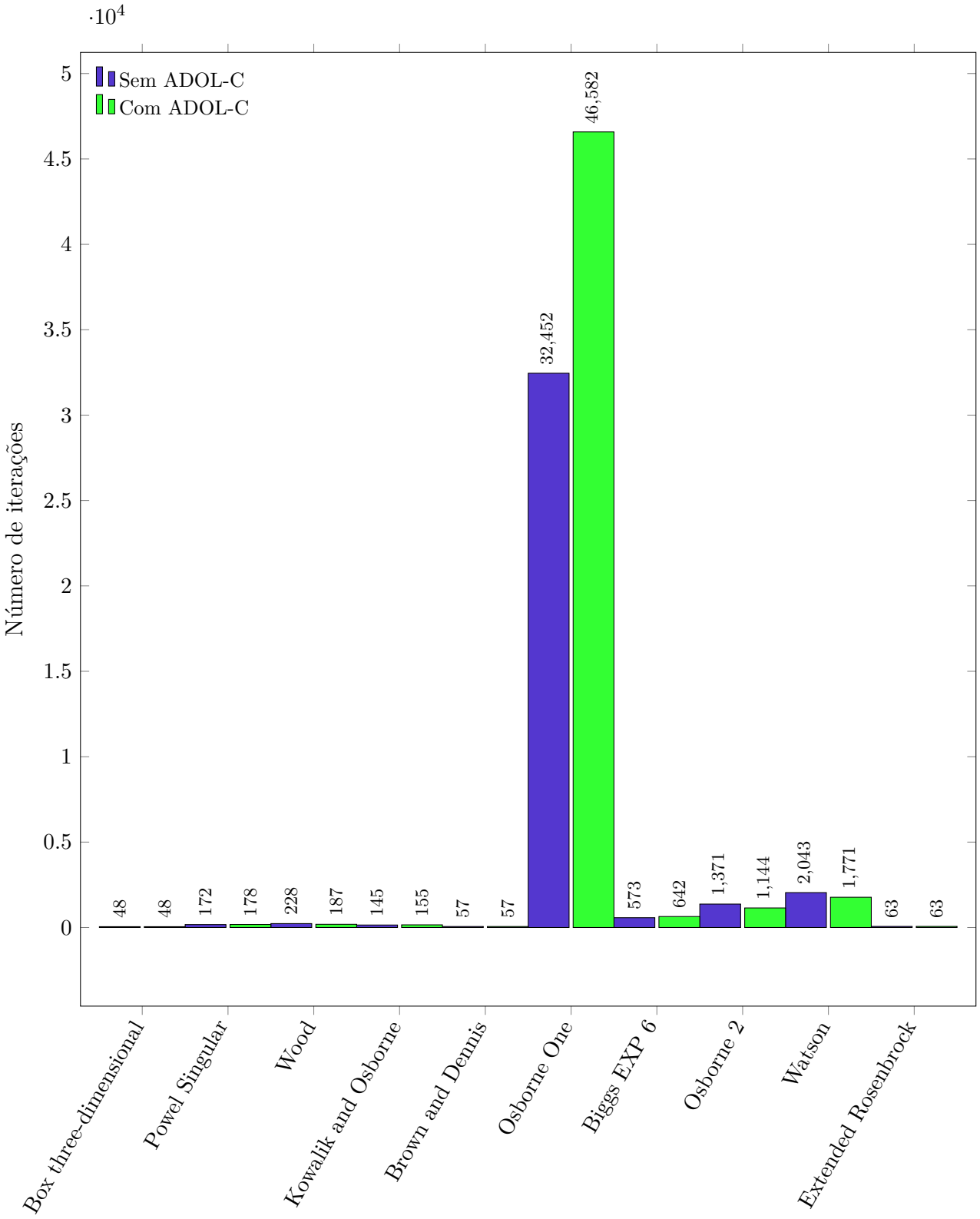

Figura 10 – Gráfico de iterações Parte II. Fonte: Autores

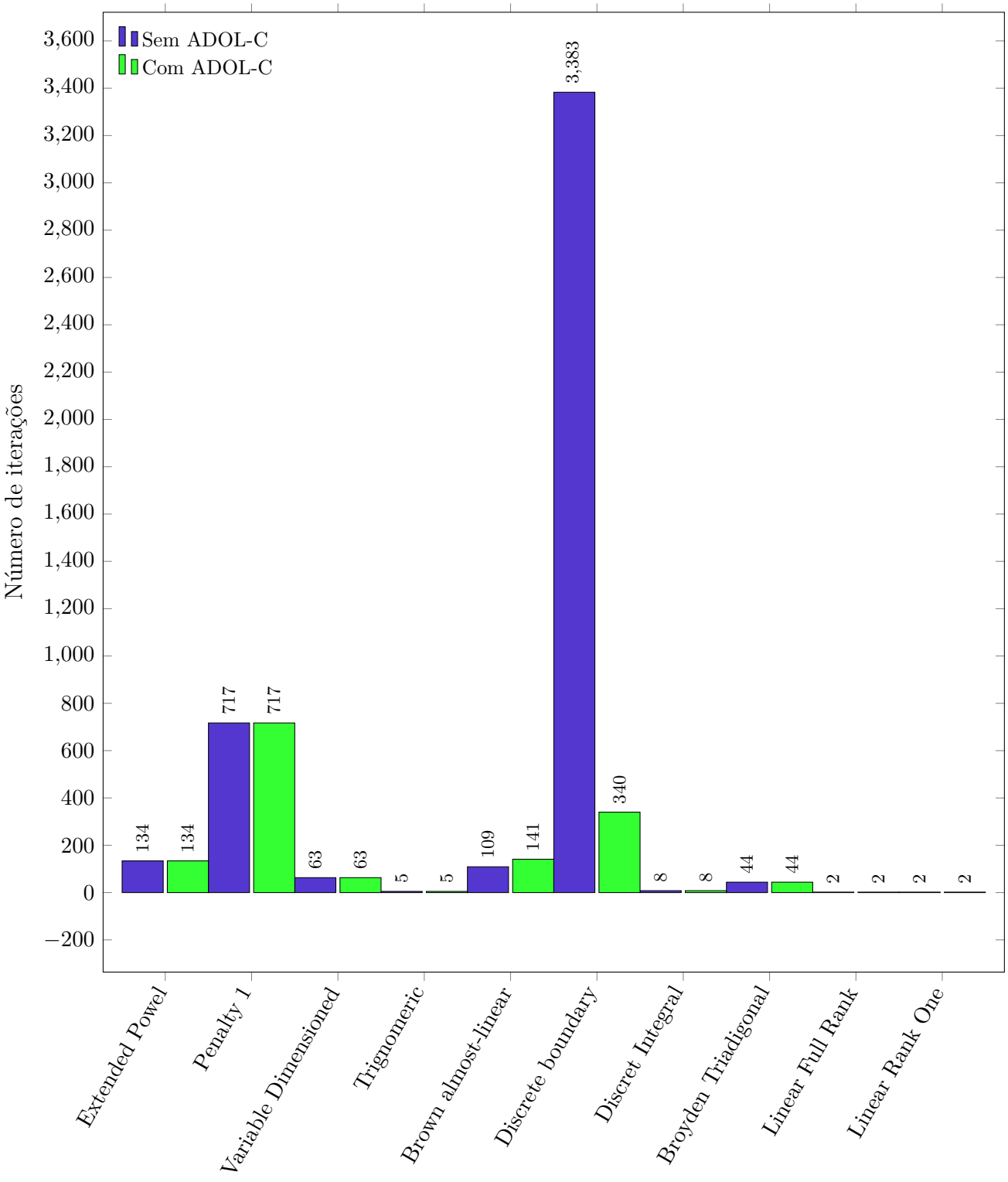

Figura 11 – Gráfico de iterações Parte III. Fonte: Autores

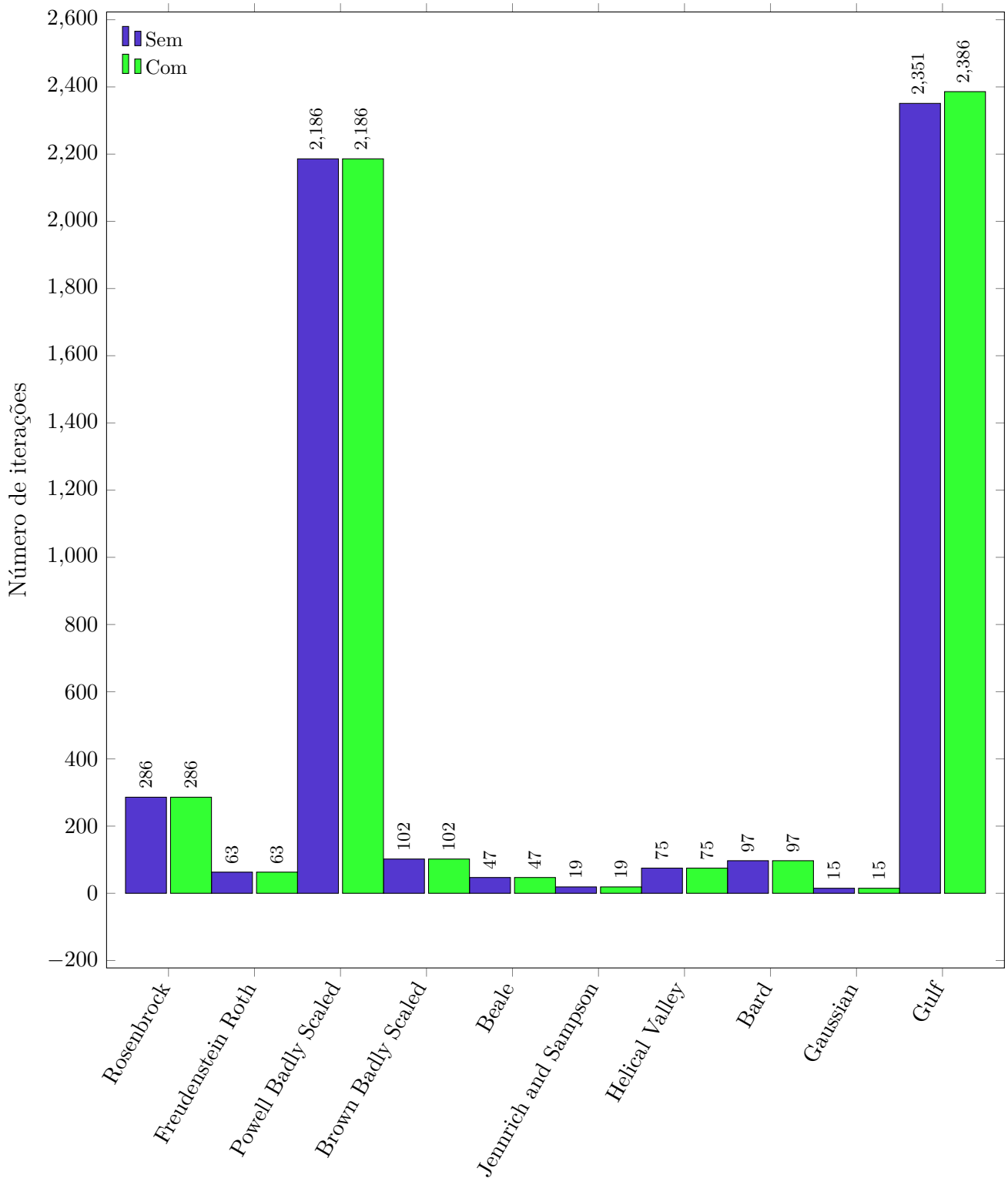

Figura 12 – Gráfico de avaliações funcionais Parte I. Fonte: Autores

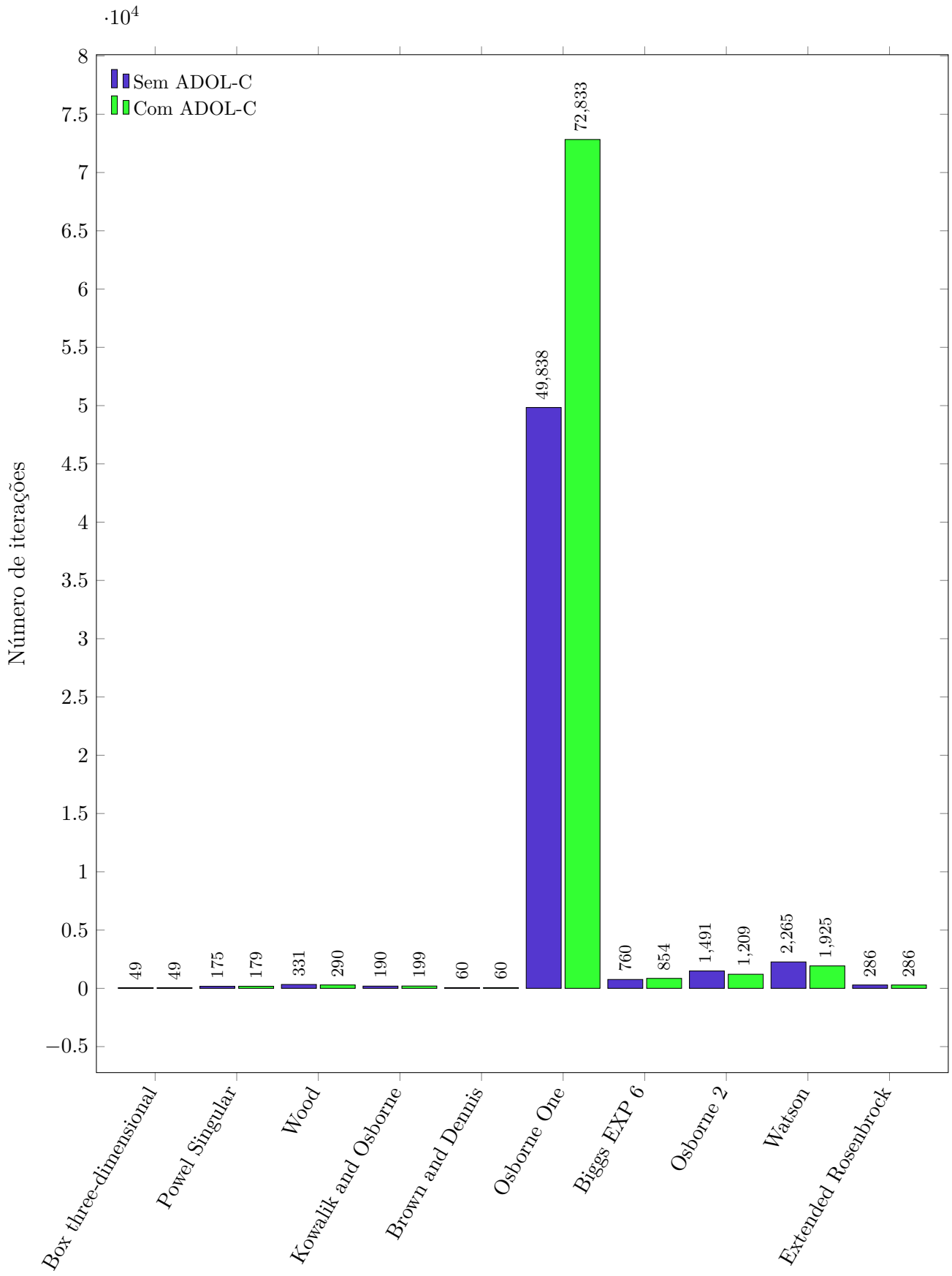

Figura 13 – Gráfico de avaliações funcionais Parte II. Fonte: Autores

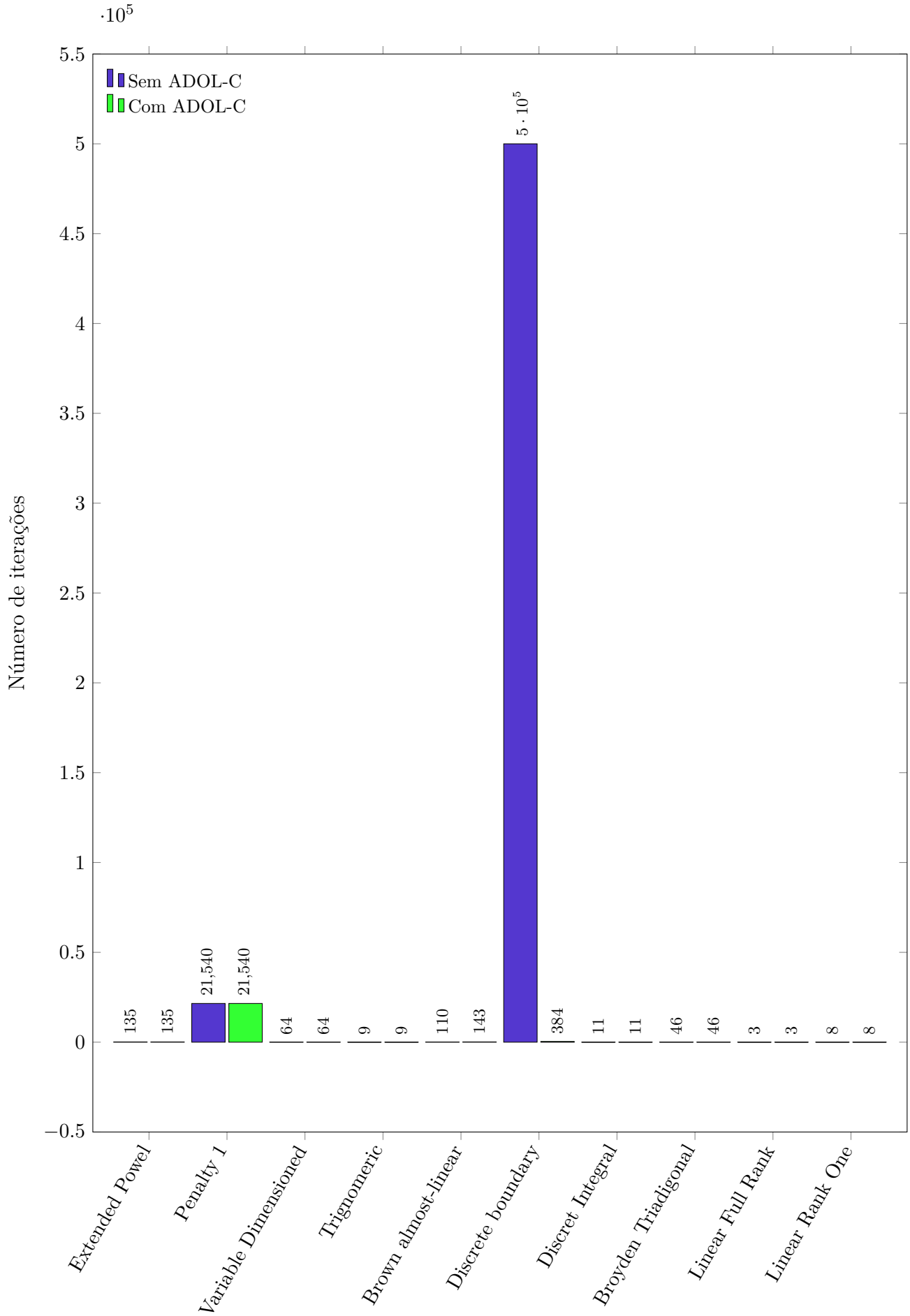

Figura 14 – Gráfico de avaliações funcionais Parte III. Fonte: Autores

Em resumo, a agregação da diferenciação automática no SPG foi validada por meio de testes que podem ser encontrados no github[2].

3.4 Considerações Finais

Destacamos os principais resultados e contribuições alcançados por meio dessa implementação, bem como discutimos suas limitações e possíveis aprimoramentos a serem realizados no futuro. Enfatizamos os benefícios obtidos com a utilização da DA no SPG, como uma possível melhoria dos números de iterações necessárias para convergir em alguns casos, e principalmente permite uma otimização de tempo para o usuário não ter que configurar a rotina `evalg` para utilizar o software. Por fim, oferecemos considerações finais sobre o trabalho realizado, ressaltando a importância da integração da DA no SPG como uma contribuição promissora para utilizar o solver.

[2] <https://github.com/NilvanPeres/AutoDiff>

4 ANÁLISE DOS RESULTADOS

Neste capítulo, realizaremos uma análise aprofundada dos resultados obtidos em nossas pesquisas, com o objetivo de avaliar o desempenho e a eficácia das abordagens adotadas. Os resultados aqui apresentados são fruto de estudo e esforço dedicado à resolução dos problemas propostos.

4.1 Procedimentos Metodológicos

Para atingir o objetivo proposto neste trabalho, foram realizados cuidadosamente dois passos fundamentais. O primeiro deles consistiu numa revisão bibliográfica, onde foram exploradas diversas fontes de referência relacionadas à técnica de Diferenciação Automática (DA), suas aplicações e como ela se relaciona com o campo da otimização. Nessa fase, buscou-se compreender os fundamentos teóricos da DA, suas vantagens e desafios, bem como seus campos de aplicação em diferentes áreas do conhecimento.

A revisão bibliográfica também permitiu identificar os avanços recentes na área de otimização com a utilização da DA, bem como as principais contribuições de pesquisadores e as tendências emergentes. Essa análise crítica do estado-da-arte da DA na otimização forneceu um embasamento para a seleção da melhor abordagem a ser adotada neste trabalho.

Em seguida, o segundo passo envolveu uma análise das diferentes bibliotecas de DA disponíveis atualmente. Foram avaliadas suas características, funcionalidades e desempenho, bem como sua compatibilidade e adequação para ser integrada ao propósito específico deste trabalho. Diversos critérios foram considerados, como a documentação disponível, a facilidade de uso, a comunidade de suporte e a escalabilidade da biblioteca para lidar com problemas de diferentes dimensões e complexidades.

Essa análise foi essencial para garantir a escolha da biblioteca mais adequada ao contexto do trabalho e para assegurar que a implementação da DA no algoritmo de otimização SPG fosse feita com qualidade e precisão. Com base nos resultados dessa avaliação, foi possível selecionar a biblioteca que apresentou as melhores características e que melhor se alinhou aos objetivos e requisitos deste trabalho.

Ambos os passos, a revisão bibliográfica e a análise das bibliotecas de DA, foram essenciais para o embasamento do desenvolvimento deste trabalho. Essa abordagem metodológica proporcionou uma compreensão embasada sobre a técnica de Diferenciação Automática e permitiu a escolha da biblioteca mais adequada para a implementação do método de otimização SPG, segundo nossos requisitos. Com essa base estabelecida, o trabalho prosseguiu para a fase de implementação, e avaliação da eficácia da DA no algoritmo proposto.

O cronograma de execução do trabalho foi dividido em etapas, cada uma delas teve

um objetivo específico e uma data de término cumprido. Algumas das atividades incluídas no cronograma foram listadas na tabela 1.

Tabela 1 – Atividades que foram desenvolvidas durante todo o trabalho

ID	Atividade
01	Revisão bibliográfica;
02	Estudo das bibliotecas que implementam DA;
03	Escolher uma das bibliotecas;
04	Implementar no SPG;
05	Elaboração do relato;
06	Apresentação;

Segue uma descrição de cada uma das etapas das atividades realizadas:

- **Atividade 01:** Nessa etapa, foram identificados os principais autores e trabalhos relacionados ao assunto, bem como as principais tendências e desenvolvimentos recentes na área. Além disso, foram identificadas lacunas de pesquisa e possíveis áreas de estudo futuro. A revisão bibliográfica foi realizada utilizando bases de dados como Socpus, Scielo, IEEE Xplore, ACM Digital Library, entre outras. O objetivo dessa atividade foi desbravar essa área de conhecimento, garantindo que a pesquisa fosse embasada na literatura existente e que os resultados obtidos fossem comparáveis com os trabalhos já publicados;

- **Atividade 02:** Essa atividade consistiu em buscar e estudar as principais referências sobre a técnica de Diferenciação Automática (*DA*) e sobre as bibliotecas disponíveis, selecionando uma delas para implementação desta técnica no trabalho;

- **Atividade 03:** A escolha de uma das bibliotecas foi uma atividade importante no processo de implementação da técnica de Diferenciação Automática (DA) utilizando o algoritmo SPG. Nessa etapa, revisamos diversas bibliotecas disponíveis que implementam a técnica de DA, com o objetivo de selecionar aquela que melhor se adequava às necessidades do nosso projeto. Essa decisão foi fundamental para garantir a qualidade e eficiência do trabalho, e foi realizada com cuidado e atenção aos detalhes;

- **Atividade 04:** A atividade de implementar o algoritmo SPG foi realizada utilizando uma das bibliotecas de Diferenciação Automática (DA) selecionada anteriormente. Nessa etapa, configuramos e personalizamos o código do solver SPG para atender às necessidades específicas do nosso trabalho. Além disso, validamos os resultados obtidos por meio de testes e comparações com trabalhos semelhantes já publicados na literatura. Essa etapa exigiu conhecimentos em programação, matemática e sobre a biblioteca escolhida. Dedicamos tempo e esforço para garantir uma implementação correta e eficiente;

- **Atividade 05:** A atividade de elaboração do relato foi concluída, documentando todas as etapas e resultados do trabalho de conclusão de curso. O relato incluiu uma descrição detalhada dos objetivos, resultados obtidos e conclusões alcançadas. Foram adicionadas

referências bibliográficas relevantes, gráficos e tabelas para facilitar a compreensão e interpretação dos resultados. O relato foi escrito de forma clara, objetiva e coerente, com o intuito de ser compreensível tanto para especialistas na área quanto para leitores não especialistas. Além disso, foram seguidas as normas e padrões adequados para garantir a qualidade e a consistência do relato final;

4.2 Resultados

Foram realizados testes em um conjunto de problemas do MGH (MORÉ; GARBOW; HILLSTROM, 1981) denominado: testando software de otimização irrestrita(Testing Unconstrained Optimization Software) com o objetivo verificar a viabilidade de uso da técnica DA e de um conjunto de bibliotecas que a implementam, em relação ao método SPG. Para cada problema, foram coletados os seguintes dados:

1. Número do Problema: Identificação numérica do problema do MGH.
2. Número de Iterações: Quantidade de iterações necessárias para atingir a convergência do algoritmo.
3. Número de Avaliações Funcionais: Contagem do número de vezes que a função objetivo foi avaliada durante o processo de otimização.
4. Valor da Função Objetivo: Valor mínimo da função objetivo obtido pelo algoritmo.
5. Sup-norma do Gradiente Projetado: Valor máximo do gradiente projetado em relação às restrições do problema.
6. Pontos Finais: Pares de coordenadas que representam os pontos finais encontrados pelo SPG.
7. Utilizando Adol-c: Indicação se a biblioteca ADOL-C foi utilizada no teste (Sim ou Não).

Foram conduzidos todos os trinta e cinco testes, dos quais dezoito são direcionados à minimização irrestrita, quatorze são de sistemas não lineares de equações e dezoito de sistemas mínimos quadráticos não lineares. Dentro desses subconjuntos, destacamos que, lamentavelmente, os testes: NumP 10 Meyern - NumP 24 - Penalty Function II, NumP 35 - Chebyquad function não alcançaram a convergência com ou sem o uso da biblioteca de diferenciação automática. Outrossim, os testes NumP 31 - Broyden Banded e Nump 34 - Linear Function Rank One With Zero Columns and Rows convergiram apenas utilizando a biblioteca ADOL-C. Essa informação é relevante para compreender a eficácia do algoritmo com e sem o uso da biblioteca em problemas complexos.

Vale a pena ressaltar que, além do sistema operacional Ubuntu 20.04, os testes foram conduzidos em uma máquina com as seguintes especificações:

- **Processador**: Intel Core i5 6200U CPU, com velocidade de 2.30GHz
- **Memória RAM**: 12 GB
- **Cores**: 2
- **Threads**: 4
- **Velocidade média do Core**: 2690.40MHz

A seguir, apresentam-se os resultados dos testes com as seguintes observações em relação aos nomes das colunas da tabela: NumP - Número do Problema, NI - Número de Iterações, VFO - Valor da Função Objetivo, SNGP - Sup-norma do Gradiente Projetado, Conv - Convergência, PF - Pontos Finais e Conv -Convergência ao resultado esperado.

As numerações dos problemas estão alinhadas com a classificação apresentada por MORÉ, Garbow e Hillstrom (1981). A seguir, são fornecidas as descrições associadas a cada NumP:

1. **NumP 1:** Refere-se ao teste de Rosenbrock.
2. **NumP 2:** Corresponde a Freudenstein and Roth.
3. **NumP 3:** Está associado ao Powell badly scaled.
4. **NumP 4:** Relaciona-se ao Brown badly scaled.
5. **NumP 5:** É associado à Beale function.
6. **NumP 6:** Representa a função de Jennrich and Sampson.
7. **NumP 7:** Caracteriza a Helical Valley function.
8. **NumP 8:** Caracteriza a Bard function.
9. **NumP 9:** Corresponde à Gaussian function.
10. **NumP 10:** Corresponde à Meyern function.
11. **NumP 11:** Refere-se à Gulf research and development function.
12. **NumP 12:** Está relacionado à Box three-dimensional function.
13. **NumP 13:** Relaciona-se à Powell Singular.
14. **NumP 14:** É atribuído ao teste de Wood.
15. **NumP 15:** É atribuído ao teste de Kowalik and Osborne.
16. **NumP 16:** Representa a Brown and Dennis function.

17. **NumP 17:** Representa a Osborne One function.
18. **NumP 18:** Caracteriza a Biggs EXP6 function.
19. **NumP 19:** Caracteriza a Osborne 2 Function.
20. **NumP 20:** Refere-se à Watson function.
21. **NumP 21:** Corresponde à Extended Rosenbrock function.
22. **NumP 22:** Está associado à Extended Powell singular function.
23. **NumP 23:** Refere-se à Penalty function I.
24. **NumP 24:** Relaciona-se à Penalty function II.
25. **NumP 25:** Corresponde à Variably dimensioned function.
26. **NumP 26:** Representa a Trigonometric function.
27. **NumP 27:** Está associado à Brown almost-linear function.
28. **NumP 28:** Representa a Discrete boundary value function.
29. **NumP 29:** Relaciona-se à Discret Integral Equation function.
30. **NumP 30:** Corresponde à Broyden Triadigonal function.
31. **NumP 31:** Caracteriza à Broyden Banded function.
32. **NumP 32:** Representa a Linear Function Full Rank.
33. **NumP 33:** Relaciona-se a Linear Function Rank One.
34. **NumP 34:** Representa a Linear Function Rank One With Zero Columns and Rows.
35. **NumP 35:** Caracteriza a Chebyquad function.

Tabela 2 – Tabela comparando resultados sem e com ADOL-C no conjunto de testes MGH (MORÉ; GARBOW; HILLSTROM, 1981)

NumP	**NI**	**NAF**	**VFO**	**SNGP**	**ADOL-C**	**Conv**
1	61	286	1.300765e-22	7.227963e-11	Não	Sim
1	61	286	1.300717e-22	7.232348e-11	Sim	Sim
2	46	63	4.898425e+01	8.803225e-07	Não	Sim
2	46	63	4.898425e+01	8.803226e-07	Sim	Sim
3	568	2775	2.372760e-07	5.833374e-07	Não	Sim
3	440	2186	3.398725e-07	8.101755e-07	Sim	Sim
4	20	102	8.673617e-19	1.862645e-09	Não	Sim
4	20	102	2.168404e-19	9.313226e-10	Sim	Sim
5	45	47	3.485428e-16	1.699635e-07	Não	Sim
5	45	47	3.481427e-16	1.697674e-07	Sim	Sim
6	3	19	1.243622e+02	1.764647e-08	Não	Sim
6	3	19	1.243622e+02	1.764642e-08	Sim	Sim
7	72	73	4.295152e-16	4.144949e-07	Não	Sim
7	74	75	2.540061e-20	5.043803e-09	Sim	Sim
9	5	15	1.127933e-08	3.607592e-08	Não	Sim
9	5	15	1.127933e-08	3.607592e-08	Sim	Sim
10	50000	218330	4.242790e+04	1.662505e+05	Não	Não
10	50000	211231	4.327260e+04	4.208585e+07	Sim	Não
11	1669	2351	5.193199e-09	3.759287e-07	Não	Sim
11	1525	2386	1.588291e-10	7.854052e-07	Sim	Sim
12	48	49	7.407840e-11	3.752579e-07	Não	Sim
12	48	49	7.408134e-11	3.755033e-07	Sim	Sim
13	172	175	1.176568e-09	1.424779e-09	Não	Sim
13	178	179	1.404320e-09	7.475338e-07	Sim	Sim
14	228	331	6.142911e-14	3.753757e-07	Não	Sim
14	187	290	2.529588e-17	3.860930e-08	Sim	Sim
15	145	190	3.075058e-04	7.635512e-07	Sim	Sim
15	155	199	3.075056e-04	2.652392e-07	Sim	Sim
16	57	60	8.582220e+04	3.513384e-07	Não	Sim
16	57	60	8.582220e+04	3.131764e-07	Sim	Sim

Tabela 3 – Tabela comparando resultados sem e com ADOL-C no conjunto de testes MGH (MORÉ; GARBOW; HILLSTROM, 1981)

NumP	NI	NAF	VFO	SNGP	ADOL-C	Conv
17	32452	49838	5.465218e-05	5.900058e-07	Sim	Sim
17	46582	72833	5.465320e-05	7.919591e-07	Sim	Sim
18	573	760	5.655650e-03	6.365884e-07	Não	Sim
18	642	854	5.655653e-03	9.594937e-07	Sim	Sim
19	1371	14910	4.013774e-02	9.857003e-07	Não	Sim
19	1144	1209	4.013774e-02	7.511502e-07	Não	Sim
20	2043	2265	2.287670e-03	4.923332e-07	Não	Sim
20	1771	1925	2.287670e-03	4.587229e-07	Sim	Sim
21	61	286	6.504088e-22	7.232437e-11	Não	Sim
21	61	286	6.503195e-22	7.227863e-11	Sim	Sim
22	134	135	3.108093e-09	7.944080e-07	Não	Sim
22	134	135	3.107123e-09	7.982340e-07	Sim	Sim
23	717	21540	7.087652e-05	2.631438e-07	Não	Sim
23	717	21540	7.087652e-05	2.631438e-07	Sim	Sim
24	293	706	1.218010e-02	3.764135e-07	Não	Não
24	469	1712	1.218010e-02	4.436872e-07	Sim	Não
25	63	64	2.131230e-23	4.924172e-12	Não	Sim
25	63	64	2.130557e-23	6.009859e-12	Sim	Sim
26	5	9	1.133278e-17	6.633368e-09	Não	Sim
26	5	9	1.133278e-17	6.633368e-09	Sim	Sim
27	109	110	5.976438e-12	1.163003e-07	Não	Sim
27	141	143	3.695838e-13	4.573764e-08	Sim	Sim
28	3383	500000	6.058900e-04	1.521699e+55	Não	Sim
28	340	384	2.277533e-11	5.728448e-07	Sim	Sim
29	8	11	5.639813e-15	7.962947e-08	Não	Não
29	8	11	4.258513e-15	7.030925e-08	Sim	Sim
30	44	46	7.530949e-15	6.623607e-07	Não	Sim
30	44	46	7.530949e-15	6.623607e-07	Sim	Sim
31	4	500000	9.263956e+00	3.285131e+01	Não	Não
31	51	54	2.148069e-15	3.777006e-07	Sim	Sim
32	2	3	0.000000e+00	0.000000e+00	Não	Sim
32	2	3	4.930381e-32	4.440892e-16	Sim	Sim
33	2	8	2.142857e+00	4.973799e-13	Não	Sim
33	2	8	2.142857e+00	5.062617e-13	Sim	Sim
34	2355	500000	2.618020e+05	2.602880e+36	Não	Não
34	2	3	3.647059e+00	1.705303e-12	Sim	Sim
35	496	648	1.421201e-01	7.470309e-07	Não	Não
35	782	1252	1.421201e-01	9.372234e-07	Sim	Não

A partir da análise da tabela, podemos extrair alguns gráficos que auxiliam a elucidar sobre a adição de uma biblioteca de DA junto ao SPG.

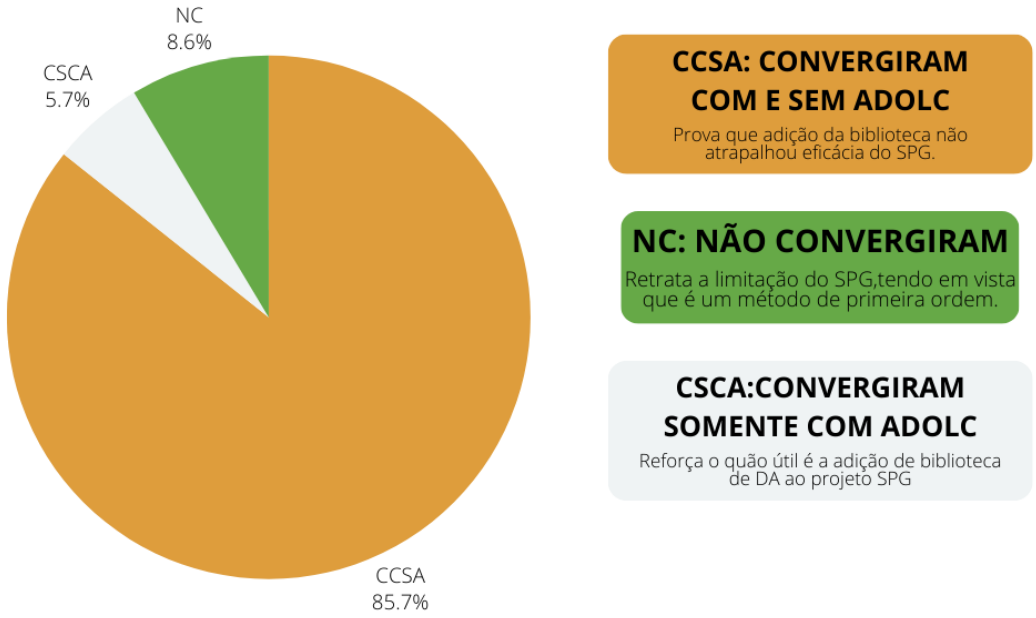

Figura 15 – Gráfico de Convergência. Fonte: Autores

Também, podemos destacar as seguintes observações a partir da análise dados da tabela:

- O SPG mostrou-se robusto em relação aos diferentes tipos de problemas do MGH, apresentando um desempenho consistente em todos os cenários testados tirando os problemas Nump 10 - Meyern, NumP 24 - Penalty Function II e NumP 35 - Chebyquad function que não alcançaram a convergência, isto é, com e sem a biblioteca ADOL-C.
- A utilização da biblioteca ADOL-C apresentou dois tipos de resultados diferentes: em alguns casos teve uma redução significativa do número de avaliações funcionais e, consequentemente, do número de iterações em alguns problemas. Entretanto, em outros casos teve um aumento no número de iterações para alcançar o resultado(a convergência).
- Em geral, a utilização do ADOL-C proporcionou resultados de qualidade em relação as derivadas parciais calculadas com DA, levando em consideração que em dois problemas (Nump 34 e Nump 31), o resultado só foi alcançando utilizando a biblioteca de DA.
- Os pontos finais obtidos pelo SPG foram próximos aos valores ótimos para ambos os cenários, com e sem o uso do ADOL-C, demonstrando a eficácia do algoritmo na busca por soluções próximas ao mínimo global. Também comprovando que é possível utilizar a biblioteca escolhida no SPG, sem perder a precisão para encontrar o resultado.

Os resultados evidenciam que a utilização da biblioteca ADOL-C mostrou-se benéfica, por facilitar a configuração das rotinas, trazendo uma certa automação para o usuário, além de proporcionar uma otimização mais rápida em alguns cenários. As soluções apresentadas neste relatório estabelecem uma base para futuros estudos e aplicações práticas na área de otimização. Além disso, seria proveitoso conduzir uma série de testes em problemas oriundos de diversos contextos, não se limitando apenas àqueles que não possuem restrições. O SPG com o suporte da biblioteca ADOL-C se provou promissor ao se destacar como uma ferramenta poderosa para lidar com as derivadas parciais e confiável pois não houve perda de precisão dos cálculos para resolver problemas complexos de otimização.

5 Conclusão

Durante o desenvolvimento deste trabalho, foram alcançados os objetivos propostos, sendo estudado o caso de aplicação da DA na implementação do algoritmo SPG. Ao investigar as bibliotecas disponíveis, selecionamos a opção mais amplamente utilizada pela comunidade.

A compreensão da técnica de DA revelou sua utilidade e benefícios no contexto da otimização. A DA calcula automaticamente e com precisão as derivadas das funções, evitando os erros numéricos associados a métodos de diferenciação aproximada.

Durante o processo, observamos que o uso de DA no SPG simplificou o processo de programação do problema de otimização, fazendo com que o usuário não precise desenvolver uma rotina para calcular a derivada de primeira ordem. Isso também implica que DA evita erros humanos na codificação.

Referências

ANDERSSON, J.; ÅKESSON, J.; DIEHL, M. CasADi: A symbolic package for automatic differentiation and optimal control. *Lecture Notes in Computational Science and Engineering*, v. 87 LNCSE, p. 297 – 307, 2012. ISSN 14397358. ISBN: 978-364230022-6 Type: Conference paper. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84865540406&doi=10.1007%2f978-3-642-30023-3_27&partnerID=40&md5=c9569a5fd5d06826bddd698caaa7def4>. Citado na página 23.

BARTHOLOMEW-BIGGS, M. et al. Automatic differentiation of algorithms. *Journal of Computational and Applied Mathematics*, v. 124, p. 171–190, dez. 2000. Citado 3 vezes nas páginas 14, 15 e 22.

BAYDIN, A. et al. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, v. 18, p. 1–43, abr. 2018. Citado na página 14.

BAYDIN, A. G. et al. Automatic Differentiation in Machine Learning: a Survey. 2017. Citado na página 22.

BEJAOUI, H.; GHAZOUANI, H.; BARHOUMI, W. Sparse coding-based representation of LBP difference for 3D/4D facial expression recognition. *Multimedia Tools and Applications*, v. 78, p. 22773–22796, ago. 2019. Citado 2 vezes nas páginas 15 e 19.

BIRGIN, E. G.; MARTÍNEZ, J. M.; RAYDAN, M. Algorithm 813: Spg—software for convex-constrained optimization. *ACM Trans. Math. Softw.*, Association for Computing Machinery, New York, NY, USA, v. 27, n. 3, p. 340–349, sep 2001. ISSN 0098-3500. Disponível em: <https://doi.org/10.1145/502800.502803>. Citado 5 vezes nas páginas 14, 17, 18, 19 e 33.

BIRGIN, E. G.; MARTíNEZ, J. M.; RAYDAN, M. Nonmonotone Spectral Projected Gradient Methods on Convex Sets. *SIAM Journal on Optimization*, v. 10, n. 4, p. 1196–1211, jan. 2000. ISSN 1052-6234, 1095-7189. Disponível em: <http://epubs.siam.org/doi/10.1137/S1052623497330963>. Citado 2 vezes nas páginas 14 e 18.

BIRGIN, E. G.; MARTíNEZ, J. M.; RAYDAN, M. Spectral Projected Gradient Methods: Review and Perspectives. *Journal of Statistical Software*, v. 60, n. 3, 2014. ISSN 1548-7660. Disponível em: <http://www.jstatsoft.org/v60/i03/>. Citado na página 17.

CORES, D. et al. On the use of the Spectral Projected Gradient method for Support Vector Machines. *Computational & Applied Mathematics*, v. 28, p. 327–364, dez. 2008. Citado na página 19.

CORLISS, G. F. et al. Automatic Differentiation of Algorithms: From Simulation to Optimization. 2002. ISBN: 9781461265436 9781461300755 Place: New York, NY Publisher: Springer New York. Disponível em: <http://link.springer.com/10.1007/978-1-4613-0075-5>. Citado na página 14.

GRIEWANK, A.; JUEDES, D.; UTKE, J. Algorithm 755: Adol-c: A package for the automatic differentiation of algorithms written in c/c++. *ACM Trans. Math. Softw.*, Association for Computing Machinery, New York, NY, USA, v. 22, n. 2, p. 131–167, jun 1996. ISSN 0098-3500. Disponível em: <https://doi.org/10.1145/229473.229474>. Citado na página 23.

GRIEWANK, A.; JUEDES, D.; UTKE, J. Algorithm 755: ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++. *ACM Trans. Math. Softw.*, v. 22, p. 131–167, jun. 1996. Citado 2 vezes nas páginas 27 e 28.

GRIEWANK, A.; WALTHER, A. Introduction to Automatic Differentiation. *PAMM*, v. 2, n. 1, p. 45–49, mar. 2003. ISSN 1617-7061, 1617-7061. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1002/pamm.200310012>. Citado na página 14.

GRIEWANK, A.; WALTHER, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. 2nd ed. ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2008. OCLC: ocn227574816. ISBN 978-0-89871-659-7. Citado 2 vezes nas páginas 20 e 22.

HASCOET, L.; PASCUAL, V. The Tapenade Automatic Differentiation tool: principles, model, and specification. *ACM Transactions on Mathematical Software*, v. 39, n. 3, 2013. Publisher: Association for Computing Machinery. Disponível em: <https://inria.hal.science/hal-00913983>. Citado 2 vezes nas páginas 23 e 25.

JUNIOR, P. R.; PAMPLONA, E. D. O.; SALOMON, F. L. R. OTIMIZAÇÃO DE PORTFÓLIOS: ANÁLISE DE EFICIÊNCIA. *Revista de Administração de Empresas*, v. 54, p. 405–413, ago. 2014. ISSN 0034-7590, 2178-938X. Publisher: Fundação Getulio Vargas, Escola de Administração de Empresas de S.Paulo. Disponível em: <http://www.scielo.br/j/rae/a/Rxp3hRsyfzwxL6dz8YfpRwf/?lang=pt>. Citado 2 vezes nas páginas 14 e 18.

KEERTHI, S. S. et al. Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, v. 13, n. 3, p. 637–649, mar. 2001. ISSN 0899-7667, 1530-888X. Disponível em: <https://direct.mit.edu/neco/article/13/3/637-649/6485>. Citado na página 19.

LIMA, J. A. G. d. *Otimização em Meteorologia: cálculo de perturbações condicionais não-lineares ótimas*. Tese (text) — Universidade de São Paulo, maio 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-25052012-170217/>. Citado 2 vezes nas páginas 15 e 19.

LLAVE, B. C. *Aplicação do método do Gradiente Espectral Projetado ao problema de Compressive Sensing*. Tese (Mestrado em Ciência da Computação) — Universidade de São Paulo, São Paulo, set. 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-02102012-162650/>. Citado 2 vezes nas páginas 18 e 19.

LOBAO, W. J. D. A. *SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS ORDINÁRIAS, PARCIAIS E ESTOCÁSTICAS POR PROGRAMAÇÃO GENÉTICA E DIFERENCIAÇÃO AUTOMÁTICA*. Tese (DOUTOR EM ENGENHARIA ELÉTRICA) — PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, Rio de Janeiro, Brazil, abr. 2015. Disponível em: <http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=29824@1>. Citado na página 22.

MORÉ, J. J.; GARBOW, B. S.; HILLSTROM, K. E. Testing Unconstrained Optimization Software. *ACM Transactions on Mathematical Software*, v. 7, n. 1, p. 17–41, mar. 1981. ISSN 0098-3500. Disponível em: <https://dl.acm.org/doi/10.1145/355934.355936>. Citado 6 vezes nas páginas 10, 35, 45, 46, 48 e 49.

POLYAK, B. T.; JUDITSKY, A. B. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, v. 30, n. 4, p. 838–855, jul. 1992. ISSN 0363-0129, 1095-7138. Disponível em: <http://epubs.siam.org/doi/10.1137/0330046>. Citado na página 19.

UTKE, J. et al. OpenAD/F: A modular open-source tool for automatic differentiation of fortran codes. *ACM Transactions on Mathematical Software*, v. 34, n. 4, 2008. Type: Article. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-42149157630&doi=10.1145%2f1377596.1377598&partnerID=40&md5=0747bf112b5e03b06a92da0cf8879ea3>. Citado 2 vezes nas páginas 23 e 24.

WALTHER, A. Getting Started with ADOL-C. In: NAUMANN, U. et al. (Ed.). *Combinatorial Scientific Computing*. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2009. (Dagstuhl Seminar Proceedings (DagSemProc), v. 9061), p. 1–10. ISSN: 1862-4405. Disponível em: <https://drops.dagstuhl.de/opus/volltexte/2009/2084>. Citado na página 27.

ZHOU, F.; TOTH, Z. On the Prospects for Improved Tropical Cyclone Track Forecasts. *Bulletin of the American Meteorological Society*, v. 101, p. 1–55, ago. 2020. Citado 2 vezes nas páginas 14 e 18.