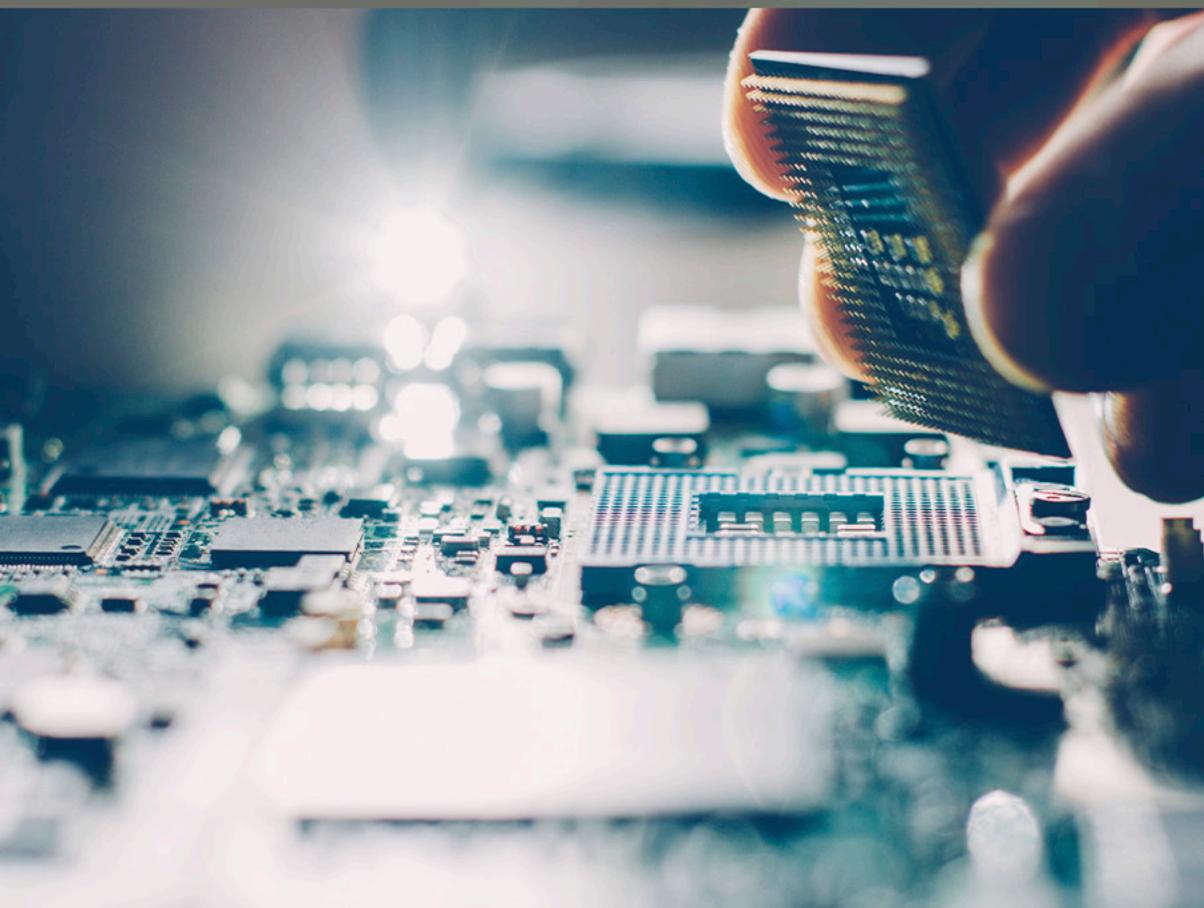


COLEÇÃO

# DESAFIOS DAS ENGENHARIAS:

ENGENHARIA DE COMPUTAÇÃO 3

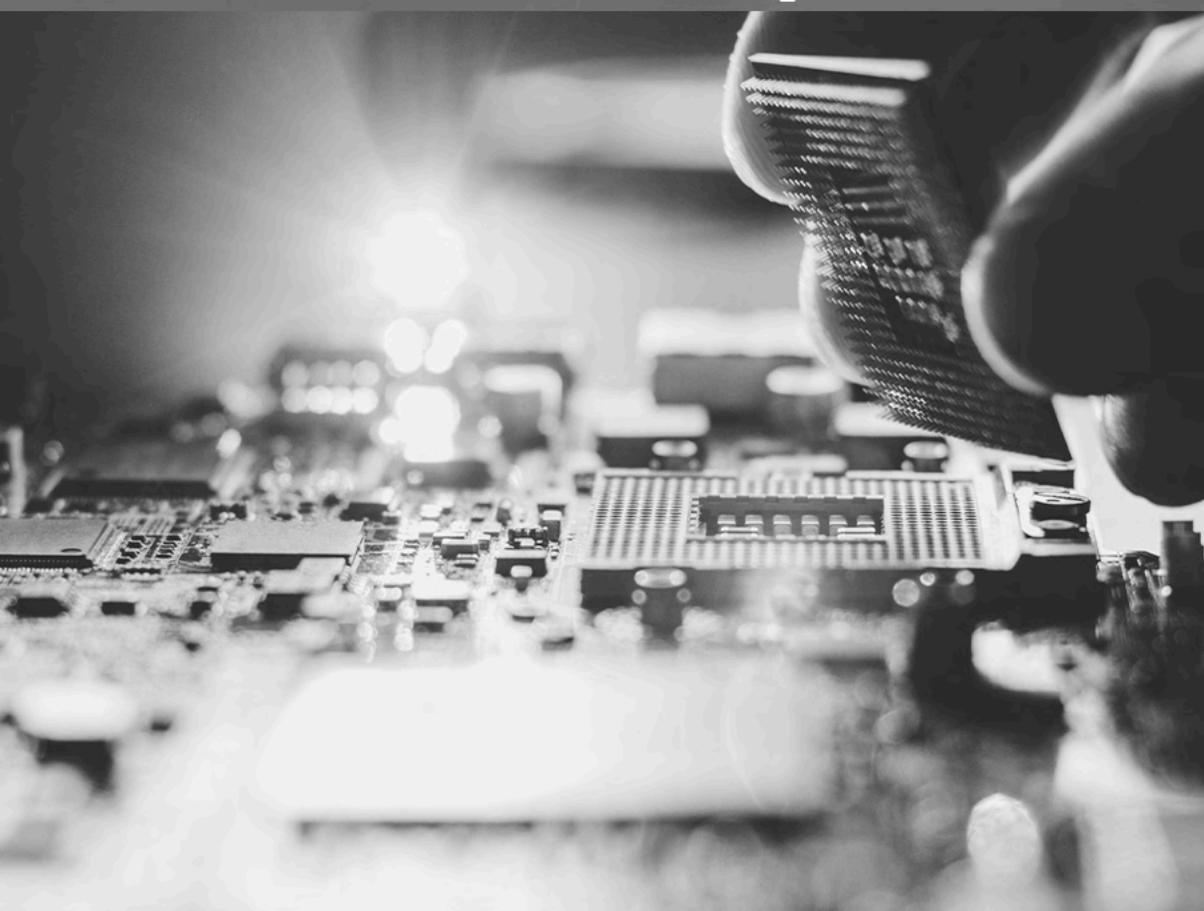


LILIAN COELHO DE FREITAS  
(ORGANIZADORA)

Atena  
Editora  
Ano 2021

COLEÇÃO  
**DESAFIOS**  
DAS  
**ENGENHARIAS:**

**ENGENHARIA DE COMPUTAÇÃO 3**



LILIAN COELHO DE FREITAS  
(ORGANIZADORA)

**Atena**  
Editora  
Ano 2021

**Editora chefe**

Profª Drª Antonella Carvalho de Oliveira

**Editora executiva**

Natalia Oliveira

**Assistente editorial**

Flávia Roberta Barão

**Bibliotecária**

Janaina Ramos

**Projeto gráfico**

Camila Alves de Cremo

Daphynny Pamplona

Gabriel Motomu Teshima

Luiza Alves Batista

Natália Sandrini de Azevedo

**Imagens da capa**

iStock

**Edição de arte**

Luiza Alves Batista

2021 by Atena Editora

Copyright © Atena Editora

Copyright do texto © 2021 Os autores

Copyright da edição © 2021 Atena Editora

Direitos para esta edição cedidos à Atena Editora pelos autores.

Open access publication by Atena Editora



Todo o conteúdo deste livro está licenciado sob uma Licença de Atribuição *Creative Commons*. Atribuição-Não-Comercial-NãoDerivativos 4.0 Internacional (CC BY-NC-ND 4.0).

O conteúdo dos artigos e seus dados em sua forma, correção e confiabilidade são de responsabilidade exclusiva dos autores, inclusive não representam necessariamente a posição oficial da Atena Editora. Permitido o *download* da obra e o compartilhamento desde que sejam atribuídos créditos aos autores, mas sem a possibilidade de alterá-la de nenhuma forma ou utilizá-la para fins comerciais.

Todos os manuscritos foram previamente submetidos à avaliação cega pelos pares, membros do Conselho Editorial desta Editora, tendo sido aprovados para a publicação com base em critérios de neutralidade e imparcialidade acadêmica.

A Atena Editora é comprometida em garantir a integridade editorial em todas as etapas do processo de publicação, evitando plágio, dados ou resultados fraudulentos e impedindo que interesses financeiros comprometam os padrões éticos da publicação. Situações suspeitas de má conduta científica serão investigadas sob o mais alto padrão de rigor acadêmico e ético.

**Conselho Editorial**

**Ciências Exatas e da Terra e Engenharias**

Prof. Dr. Adélio Alcino Sampaio Castro Machado – Universidade do Porto

Profª Drª Ana Grasielle Dionísio Corrêa – Universidade Presbiteriana Mackenzie

Prof. Dr. Carlos Eduardo Sanches de Andrade – Universidade Federal de Goiás

Profª Drª Carmen Lúcia Voigt – Universidade Norte do Paraná

Prof. Dr. Cleiseano Emanuel da Silva Paniagua – Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Prof. Dr. Douglas Gonçalves da Silva – Universidade Estadual do Sudoeste da Bahia  
Prof. Dr. Eloi Rufato Junior – Universidade Tecnológica Federal do Paraná  
Profª Drª Érica de Melo Azevedo – Instituto Federal do Rio de Janeiro  
Prof. Dr. Fabrício Menezes Ramos – Instituto Federal do Pará  
Profª Dra. Jéssica Verger Nardeli – Universidade Estadual Paulista Júlio de Mesquita Filho  
Prof. Dr. Juliano Carlo Rufino de Freitas – Universidade Federal de Campina Grande  
Profª Drª Luciana do Nascimento Mendes – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte  
Prof. Dr. Marcelo Marques – Universidade Estadual de Maringá  
Prof. Dr. Marco Aurélio Kistemann Junior – Universidade Federal de Juiz de Fora  
Profª Drª Neiva Maria de Almeida – Universidade Federal da Paraíba  
Profª Drª Natiéli Piovesan – Instituto Federal do Rio Grande do Norte  
Profª Drª Priscila Tessmer Scaglioni – Universidade Federal de Pelotas  
Prof. Dr. Sidney Gonçalo de Lima – Universidade Federal do Piauí  
Prof. Dr. Takeshy Tachizawa – Faculdade de Campo Limpo Paulista

**Diagramação:** Daphynny Pamplona  
**Correção:** Gabriel Motomu Teshima  
**Indexação:** Amanda Kelly da Costa Veiga  
**Revisão:** Os autores  
**Organizadora:** Lilian Coelho de Freitas

**Dados Internacionais de Catalogação na Publicação (CIP)**

C691 Coleção desafios das engenharias: engenharia de computação 3 / Organizadora Lilian Coelho de Freitas. – Ponta Grossa - PR: Atena, 2021.

Formato: PDF

Requisitos de sistema: Adobe Acrobat Reader

Modo de acesso: World Wide Web

Inclui bibliografia

ISBN 978-65-5983-619-2

DOI: <https://doi.org/10.22533/at.ed.192212911>

1. Engenharia de computação. I. Freitas, Lilian Coelho de (Organizadora). II. Título.

CDD 621.39

**Elaborado por Bibliotecária Janaina Ramos – CRB-8/9166**

**Atena Editora**

Ponta Grossa – Paraná – Brasil

Telefone: +55 (42) 3323-5493

[www.atenaeditora.com.br](http://www.atenaeditora.com.br)

[contato@atenaeditora.com.br](mailto:contato@atenaeditora.com.br)

## DECLARAÇÃO DOS AUTORES

Os autores desta obra: 1. Atestam não possuir qualquer interesse comercial que constitua um conflito de interesses em relação ao artigo científico publicado; 2. Declaram que participaram ativamente da construção dos respectivos manuscritos, preferencialmente na: a) Concepção do estudo, e/ou aquisição de dados, e/ou análise e interpretação de dados; b) Elaboração do artigo ou revisão com vistas a tornar o material intelectualmente relevante; c) Aprovação final do manuscrito para submissão.; 3. Certificam que os artigos científicos publicados estão completamente isentos de dados e/ou resultados fraudulentos; 4. Confirmam a citação e a referência correta de todos os dados e de interpretações de dados de outras pesquisas; 5. Reconhecem terem informado todas as fontes de financiamento recebidas para a consecução da pesquisa; 6. Autorizam a edição da obra, que incluem os registros de ficha catalográfica, ISBN, DOI e demais indexadores, projeto visual e criação de capa, diagramação de miolo, assim como lançamento e divulgação da mesma conforme critérios da Atena Editora.

## DECLARAÇÃO DA EDITORA

A Atena Editora declara, para os devidos fins de direito, que: 1. A presente publicação constitui apenas transferência temporária dos direitos autorais, direito sobre a publicação, inclusive não constitui responsabilidade solidária na criação dos manuscritos publicados, nos termos previstos na Lei sobre direitos autorais (Lei 9610/98), no art. 184 do Código Penal e no art. 927 do Código Civil; 2. Autoriza e incentiva os autores a assinarem contratos com repositórios institucionais, com fins exclusivos de divulgação da obra, desde que com o devido reconhecimento de autoria e edição e sem qualquer finalidade comercial; 3. Todos os e-book são *open access*, desta forma não os comercializa em seu site, sites parceiros, plataformas de *e-commerce*, ou qualquer outro meio virtual ou físico, portanto, está isenta de repasses de direitos autorais aos autores; 4. Todos os membros do conselho editorial são doutores e vinculados a instituições de ensino superior públicas, conforme recomendação da CAPES para obtenção do Qualis livro; 5. Não cede, comercializa ou autoriza a utilização dos nomes e e-mails dos autores, bem como nenhum outro dado dos mesmos, para qualquer finalidade que não o escopo da divulgação desta obra.

## APRESENTAÇÃO

A Atena Editora tem a honra de presentear o público em geral com a série de *e-books* intitulada “*Coleção desafios das engenharias: Engenharia de computação*”. Em seu terceiro volume, esta obra tem o objetivo de divulgar aplicações tecnológicas da Engenharia de Computação na resolução de problemas atuais, com o intuito de facilitar a difusão do conhecimento científico produzido em várias instituições de ensino e pesquisa do país.

Organizado em 20 capítulos, este volume apresenta temas como utilização de aprendizagem de máquina na avaliação de riscos de infecção por COVID-19; dispositivos automatizados para administração de remédios; comunicação científica apoiada por realidade aumentada; métodos de elementos finitos aplicados na análise de materiais para indústria aeronáutica; aplicações de processamento digital de imagens e de algoritmos genéticos; entre diversas outras aplicações da automação e do desenvolvimento de *software*, combinados para melhorar as atividades do nosso dia-a-dia.

Dessa forma, esta obra contribuirá para aprimoramento do conhecimento de seus leitores e servirá de base referencial para futuras investigações.

Os organizadores da Atena Editora, agradecem especialmente os autores dos diversos capítulos apresentados, parabenizam a dedicação e esforço de cada um, os quais viabilizaram a construção deste trabalho.

Boa leitura.

Lilian Coelho de Freitas

## SUMÁRIO

### **CAPÍTULO 1..... 1**

#### **EVALUATING THE RISK OF COVID-19 INFECTION BASED ON MACHINE LEARNING OF SYMPTOMS AND CONDITIONS VERSUS LABORATORY METHODS**

Daniel Mário de Lima  
João Henrique Gonçalves de Sá  
Ramon Alfredo Moreno  
Marina de Fátima de Sá Rebelo  
José Eduardo Krieger  
Marco Antonio Gutierrez

 <https://doi.org/10.22533/at.ed.1922129111>

### **CAPÍTULO 2..... 16**

#### **DISPOSITIVO AUTOMATIZADO PARA ADMINISTRAÇÃO DE REMÉDIOS**

João Roberto Silva Teixeira  
Alessandro Mainardi de Oliveira  
Ricardo Neves de Carvalho

 <https://doi.org/10.22533/at.ed.1922129112>

### **CAPÍTULO 3..... 22**

#### **INTEGRAÇÃO ENTRE DADOS TEXTUAIS DE PRONTUÁRIOS ELETRÔNICOS DO PACIENTE (PEPS) E TERMINOLOGIAS CLÍNICAS**

Amanda Damasceno de Souza  
Eduardo Ribeiro Felipe  
Fernanda Farinelli  
Jeanne Louize Emygdio  
Lívia Marangon Duffles Teixeira  
Maurício Barcellos Almeida

 <https://doi.org/10.22533/at.ed.1922129113>

### **CAPÍTULO 4..... 35**

#### **COMPARATIVE ANALYSIS OF THE PERFORMANCE OF A ENRICHED MIXED FINITE ELEMENT METHOD WITH STATIC CONDENSATION FOR POISSON PROBLEMS**

Ricardo Javier Hanco Ancori  
Jose Diego Ayñayanque Pastor  
Rómulo Walter Condori Bustincio  
Eliseo Daniel Velasquez Condori  
Roger Edwar Mestas Chávez  
Fermín Flavio Mamani Condori  
Jorge Lizardo Díaz Calle

 <https://doi.org/10.22533/at.ed.1922129114>

### **CAPÍTULO 5..... 45**

#### **COMPORTAMENTO DE PAREDE DE ALVENARIA ESTRUTURAL EM SITUAÇÃO DE INCÊNDIO: ANÁLISE NUMÉRICA**

Jean Marie Désir

Luana Zanin

 <https://doi.org/10.22533/at.ed.1922129115>

**CAPÍTULO 6..... 58**

**COMUNICAÇÃO CIENTÍFICA APOIADA POR REALIDADE AUMENTADA: O CASO DO APLICATIVO AUMENTANDO KIRIMURÊ**

Vinícius Pires de Oliveira  
Fernanda Vitória Nascimento Lisboa  
Jéssica Duarte Souza  
Brisa Santana Brasileiro  
Hilma Maria Passos de Oliveira  
Ingrid Winkler  
Andrea de Matos Machado  
Karla Schuch Brunet

 <https://doi.org/10.22533/at.ed.1922129116>

**CAPÍTULO 7..... 64**

**CONTEXTUALIZAÇÃO DO CPS DE UMA CÉLULA ROBÓTICA, ATRAVÉS DO GÊMEO DIGITAL UTILIZANDO PROTOCOLO DE COMUNICAÇÃO OPC UA**

Rogério Adas Pereira Vitalli

 <https://doi.org/10.22533/at.ed.1922129117>

**CAPÍTULO 8..... 75**

**DESENVOLVIMENTO DE UMA ARQUITETURA DE SOFTWARE BASEADA EM CENÁRIOS ARQUITETURAIS, MEMORANDOS TÉCNICOS E VISÕES DO MODELO 4+1**

Everson Willian Pereira Bacelli  
Bruno Ferreira Cardoso  
Wilson Vendramel

 <https://doi.org/10.22533/at.ed.1922129118>

**CAPÍTULO 9..... 90**

**DEVELOPMENT OF AN AIDING TOOL FOR THE OPTIMAL DETAIL OF ACTIVE REINFORCEMENT USING GENETIC ALGORITHM**

Victória Carino Neves  
Guilherme Coelho Gomes Barros

 <https://doi.org/10.22533/at.ed.1922129119>

**CAPÍTULO 10..... 106**

**ANÁLISE DOS EFEITOS DA MÉTRICA DE DISTÂNCIA NA EXTRAÇÃO DE CONJUNTOS DE SIMILARIDADE**

André Eduardo Alessi  
Bruno Duarte  
Ives Renê Venturini Pola  
Dalcimar Casanova  
Marco Antonio de Castro Barbosa

 <https://doi.org/10.22533/at.ed.19221291110>

<b>CAPÍTULO 11</b> .....	<b>119</b>
ESTUDO SOBRE AUTOMATIZAÇÃO DE EQUIVALÊNCIA DE FUNÇÕES	
Lucas Fernando Frighetto	
Fábio Hernandez	
 <a href="https://doi.org/10.22533/at.ed.19221291111">https://doi.org/10.22533/at.ed.19221291111</a>	
<b>CAPÍTULO 12</b> .....	<b>142</b>
ESTUDO SOBRE O CONTROLE REMOTO DE DISPOSITIVOS MICROCONTROLADOS UTILIZANDO DISPOSITIVOS MÓVEIS	
João Vítor Fernandes Dias	
Fermín Alfredo Tang Montané	
 <a href="https://doi.org/10.22533/at.ed.19221291112">https://doi.org/10.22533/at.ed.19221291112</a>	
<b>CAPÍTULO 13</b> .....	<b>163</b>
HERRAMIENTAS TECNOLÓGICAS APLICADAS EN EL DIBUJO ASISTIDO POR COMPUTADORA EN LA MODALIDAD A DISTANCIA	
Liliana Eneida Sánchez Platas	
Celia Bertha Reyes Espinoza	
Olivia Allende Hernández	
 <a href="https://doi.org/10.22533/at.ed.19221291113">https://doi.org/10.22533/at.ed.19221291113</a>	
<b>CAPÍTULO 14</b> .....	<b>174</b>
HISTÓRICO DAS MULHERES NA TECNOLOGIA DA INFORMAÇÃO E ANÁLISE DA PARTICIPAÇÃO FEMININA NOS CURSOS SUPERIORES DO BRASIL	
Vívian Ludimila Aguiar Santos	
Thales Francisco Mota Carvalho	
Maria do Socorro Vieira Barreto	
 <a href="https://doi.org/10.22533/at.ed.19221291114">https://doi.org/10.22533/at.ed.19221291114</a>	
<b>CAPÍTULO 15</b> .....	<b>186</b>
IDENTIFICAÇÃO DO MODELO DINÂMICO DE UMA TURBINA EÓLICA: ESTUDO DE CASO DA NORDTANK NTK 330F	
Gustavo Almeida Silveira de Souza	
Edgar Campus Furtado	
Leandro José Evilásio Campos	
Cristiane Medina Finzi Quintão	
 <a href="https://doi.org/10.22533/at.ed.19221291115">https://doi.org/10.22533/at.ed.19221291115</a>	
<b>CAPÍTULO 16</b> .....	<b>199</b>
COMFORT IN VIBRATIONS FOR THE STEEL-CONCRETE COMPOSITE FLOORS: AN APPRAISAL FOR REVIEW OF ABNT NBR 8800:2008	
João Vitor V. Freire	
André V. Soares Gomes	
Adenícia Fernanda G. Calenzani	
Johann A. Ferrareto	
 <a href="https://doi.org/10.22533/at.ed.19221291116">https://doi.org/10.22533/at.ed.19221291116</a>	

<b>CAPÍTULO 17</b> .....	<b>224</b>
FINITE ELEMENT METHOD APPLIED TO MECHANICAL ANALYSIS OF AERONAUTICAL RIBS IN CARBON FIBER AND 7075 ALUMINUM ALLOY	
Alex Fernandes de Souza	
 <a href="https://doi.org/10.22533/at.ed.19221291117">https://doi.org/10.22533/at.ed.19221291117</a>	
<b>CAPÍTULO 18</b> .....	<b>236</b>
MÉTODO PARA CALCULAR A ÁREA DE SUPERFICIAL DE RAÍZES POR PROCESSAMENTO DIGITAL DE IMAGENS	
Marcio Hosoya Name	
 <a href="https://doi.org/10.22533/at.ed.19221291118">https://doi.org/10.22533/at.ed.19221291118</a>	
<b>CAPÍTULO 19</b> .....	<b>244</b>
LOCAL MESHFREE METHOD OPTIMIZATION WITH GENETICALGORITHMS	
Wilber Vélez	
Flávio Mendonça	
Artur Portela	
 <a href="https://doi.org/10.22533/at.ed.19221291119">https://doi.org/10.22533/at.ed.19221291119</a>	
<b>CAPÍTULO 20</b> .....	<b>258</b>
NAVEGACIÓN VIRTUAL 2D Y 3D EN UN ENTORNO WEB	
Víctor Tomás Tomás Mariano	
Felipe de Jesús Núñez Cárdenas	
Jorge Hernández Camacho	
Isaura Argüelles Azuara	
Guillermo Canales Bautista	
 <a href="https://doi.org/10.22533/at.ed.19221291120">https://doi.org/10.22533/at.ed.19221291120</a>	
<b>SOBRE A ORGANIZADORA</b> .....	<b>268</b>
<b>ÍNDICE REMISSIVO</b> .....	<b>269</b>

## ESTUDO SOBRE AUTOMATIZAÇÃO DE EQUIVALÊNCIA DE FUNÇÕES

Data de aceite: 01/11/2021

### Lucas Fernando Frighetto

Departamento de Ciência da Computação –  
Universidade Estadual do Centro-Oeste  
Guarapuava– PR– Brasil  
<http://lattes.cnpq.br/1095181674675390>

### Fábio Hernandes

Departamento de Ciência da Computação –  
Universidade Estadual do Centro-Oeste  
Guarapuava– PR– Brasil  
<http://lattes.cnpq.br/2110808766116337>

**RESUMO:** Este projeto tem por objetivo propor um método exato que, prova uma equivalência entre funções. As transformações em estudo são comprovadas por meio da indução matemática. Foi proposto um algoritmo que comprova a equivalência entre um conjunto de funções utilizando manipulação de strings. Para encontrar a igualdade entre funções são realizadas operações matemáticas sobre o números naturais, tais como: distributiva, associativa, comutativa e mínimo múltiplo comum. Como resultado, foram comprovadas as equivalências entre um conjunto de funções constante do banco de dados.

**PALAVRAS-CHAVE:** Indução matemática, funções, equivalência

### A STUDY ABOUT EQUIVALENCE AUTOMATIZATION OF FUNCTIONS

**ABSTRACT:** This project aims to propose an

exact method that, referring to a database, returns an equivalent function. The study transformations are provided through mathematical induction. It was proposed an algorithm that using strings manipulation, proves an equivalence between a set of functions. To find the equality between functions are used mathematical operations on the natural numbers, such as: distributive, associative, commutative and minimum common multiple. As a result, they were proved the equivalences between a set of functions in the database.

**KEYWORDS:** Mathematical induction, functions, equivalence.

## 1 | INTRODUÇÃO

Um dos principais problemas da Ciência da Computação é a otimização automática de códigos, os compiladores atuais possuem diversas técnicas para otimizá-los, porém, são limitadas [Aho et al. 2008]. A principal dificuldade nesta otimização está em descobrir como transformá-los preservando seu significado e reduzindo o tempo computacional. Algumas transformações são triviais, como código morto e recálculo de funções, porém outras são mais difíceis, como, por exemplo, transformar um *bubblesort* em *quicksort*. Com isso, foi realizado neste trabalho uma pesquisa com a meta de obter conclusões sobre automatização de otimização de funções.

Existem várias formas pelas quais um

compilador pode efetuar melhorias num programa preservando a sua função, os exemplos mais comuns de melhorias que preservam o significado do código podem ser encontradas em [Aho et al. 2008].

Entre outros trabalhos encontrados na literatura relacionados à otimização de código, destacam-se a otimização de laços [Allen 1969], que foi um dos estudos pioneiros sobre otimização de código, este é sobre otimização em estruturas de repetição. Outros trabalhos citados aqui são a otimização de código em *Power Architectures* [IBM 2016] e a aproximação estática para o problema de compactação de código horizontal para a classe paralela, sincronizada em arquiteturas não homogêneas [De Gloria 1991]. Além destes, ressalta-se também o trabalho sobre ferramentas para otimização de código e sistema de avaliação de processamento de imagem para o sistema *PAPRICA-3* [Bertozzi 1999].

Seguindo a temática de otimização de códigos, este projeto tem por objetivo o desenvolvimento de uma técnica capaz de comprovar a equivalência entre funções envolvendo números naturais, técnica esta que poderá ser aplicada à otimização de código, pois se provada a equivalência entre duas funções, o compilador poderá trocar uma função recorrente de alto custo por outra função fechada de custo computacional menor. O desenvolvimento dessa técnica consiste em implementar operações matemáticas (distributiva, associativa, comutativa, mínimo múltiplo comum), por meio de manipulação de *strings*, a fim de provar a equivalência entre funções por meio da indução matemática.

Este trabalho está organizado da seguinte forma: na Seção 2 estão os materiais e métodos, a Seção 3 aborda o algoritmo proposto, na Seção 4 estão os resultados computacionais, enquanto que na Seção 5 são destacadas as conclusões e trabalhos futuros.

## 2 | MATERIAIS E MÉTODOS

Nesta seção são abordados alguns conceitos e definições utilizados no trabalho. Inicialmente é introduzido o problema da preservação do significado do código e as técnicas de transformação, em seguida é abordado o método da indução matemática, sendo esta a base para confirmação das equivalências.

### 2.1 Indução matemática

Dentre todos os números já considerados, os naturais ( $N$ ) foram os pioneiros, inicialmente com o intuito de contar. Apesar desses números serem os mais simples, não quer dizer que sejam totalmente entendidos, havendo ainda muitos mistérios que os cercam [Iezzi e Murakami 2004, Hazzan 2004, Menezes 2005].

Provar operações matemáticas entre números naturais não é tão simples, sendo o princípio de indução matemática o mais utilizado [Iezzi 2005]. Porém, além da prova entre operações matemáticas, a indução matemática [Antonio 2017] é uma importante ferramenta utilizada para verificar conjecturas sobre padrões de termos em sequências.

Neste estudo, a indução matemática é utilizada para verificar a equivalência entre funções a fim de compará-las em seus respectivos custos computacionais e aplicar estes resultados à otimização de código.

A fim de visualizar a ideia da indução matemática, imagine uma coleção de dominós colocados numa sequência (formação) de tal forma que a queda do primeiro dominó força a queda do segundo, que força a queda do terceiro, e assim sucessivamente, até todos os dominós caírem. Em outras palavras, se a equivalência entre funções for encontrada para um número qualquer e para o próximo número, então consequentemente será válida para todos os números subsequentes.

A igualdade entre as funções pode ser encontrada aplicando transformações de propriedades algébricas. O problema é descobrir a sequência de transformações que levam as funções à um resultado em comum.

### 2.1.1 Gauss e a Indução Matemática

Conta-se a seguinte história [Hefez 2009] sobre o matemático alemão Carl Friedrich Gauss (1777-1855), quando ainda garoto. Na escola, o professor, para aquietar a turma de Gauss, mandou os alunos calcularem a soma de todos os números naturais de 1 a 100. Qual não foi a surpresa quando, pouco tempo depois, o menino deu a resposta: 5050. Indagado como tinha descoberto tão rapidamente o resultado, Gauss, então com nove anos de idade, descreveu o método a seguir.

$S_n = 1 + 2 + \dots + n$ , o objetivo é encontrar uma fórmula fechada para  $S_n$ . Somando a igualdade de  $S_n$ , membro a membro, com ela mesma, porém com as parcelas do segundo membro em ordem invertida, temos que:

$$\begin{aligned} S_n &= 1 + 2 + \dots + n \\ S_n &= n + (n-1) + \dots + 1 \\ \hline 2S_n &= (n+1) + (n+1) + \dots + (n+1) \\ 2S_n &= n(n+1) \text{ e, portanto, } S_n = \frac{n(n+1)}{2}. \end{aligned}$$

### 2.1.2 Princípio de Indução Matemática

Dado um subconjunto  $S$  do conjunto dos números naturais  $\mathbb{N}$ , tal que e sempre que um número  $n \in S$ , o número  $n+1$  também pertence a  $S$ , tem-se que  $S = \mathbb{N}$ .

Esta simples propriedade fornece uma poderosa técnica de demonstração matemática, a *demonstração por indução*, que consiste em [Antonio 2017]:

*“Suponha que seja dada uma sentença matemática  $P(n)$  que dependa de uma variável natural  $n$ , a qual se torna verdadeira ou falsa quando substituimos  $n$  por um número natural dado qualquer. Tais sentenças são ditas sentenças abertas definidas sobre o conjunto dos naturais”.*

Logo, a demonstração por indução matemática segue o teorema a seguir:

**Teorema.** Seja  $P(n)$  uma sentença aberta sobre  $\mathbb{N}$ . Suponha que:

(i)  $P(1)$  é verdadeira; e

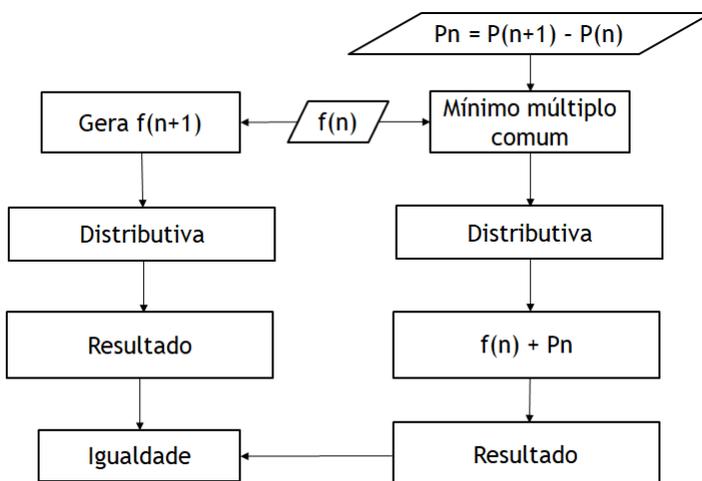
(ii) qualquer que seja  $n \in \mathbb{S}$ , sempre que  $P(n)$  é verdadeira, segue que  $P(n + 1)$  é verdadeira.

Então,  $P(n)$  é verdadeira para todo  $n \in \mathbb{N}$ .

### 3 | ALGORITMO PROPOSTO

O algoritmo proposto procura a igualdade entre uma  $P(n)$ , neste trabalho associado à funções recorrentes e uma função  $f(n)$ , neste trabalho associado à funções fechadas. O método tem como entrada uma função  $f(n)$  e o termo necessário da  $P(n)$  que é  $P(n+1) - P(n)$ . Existe um número muito grande de possibilidades de transformações e portanto não há um método exato e ótimo para encontrar a igualdade entre funções. Para resolver este problema, foram desenvolvidos dois caminhos (possibilidades de encontrar a equivalência), ambos compostos de transformações válidas, porém com resultados finais diferentes, ou seja, se entre as funções da base de dados não for encontrada a equivalência utilizando o caminho 1, esta será encontrada por meio do caminho 2 (2ª possibilidade).

Inicialmente é gerada uma função  $f(n+1)$ , a partir da função  $f(n)$ , ao substituir variável  $n$  por  $n+1$ , então, aplica-se a operação distributiva para a função e, após, é realizada a soma da função  $f(n)$  inicial com o termo da  $P(n)$ . Para isso, primeiramente é executada uma operação de mínimo múltiplo comum, então aplica-se a distributiva e, para finalizar, somam-se os termos, se o resultado for igual ao obtido por meio da substituição de  $n$  por  $n+1$  seguido da distributiva, então conclui-se que a equivalência é verdadeira. Esta parte da implementação está ilustrada no fluxograma a seguir.



Algoritmo 1. Passos de parte da implementação.

O que difere o Caminho 2 (segunda possibilidade) do primeiro na implementação é que para algumas funções é necessário utilizar uma sequência de transformações não triviais como, por exemplo, quando se busca verificar se o divisor de uma função está contido no divisor da outra, o que minimiza o cálculo do mínimo múltiplo comum, porém utilizam-se os mesmos métodos do Caminho 1. Outra diferença é a necessidade de multiplicar a  $f(n+1)$  por  $a/a$  tal que  $f(n+1) = a/b$ .

**Exemplo.** Caminho 2.

Dada a seguinte soma:

$$\frac{n}{(n+1)} + \frac{(n+1)}{(n+1)(n+2)}.$$

Utilizando o Caminho 1 (Figura 2) tem-se o seguinte resultado:

$$\frac{n}{(n+1)} + \frac{(n+1)}{(n+1)(n+2)} = \frac{n(n+1)(n+2) + (n+1)(n+1)}{(n+1)(n+1)(n+2)}.$$

Já considerando o Caminho 2, o resultado será:

$$\frac{n}{(n+1)} + \frac{(n+1)}{(n+1)(n+2)} = \frac{n(n+2) + (n+1)}{(n+1)(n+2)},$$

isso ocorre porque o denominador da primeira fração está contido no denominador da segunda.

Os métodos implementados podem ser encontrados no Apêndice.

## 4 | RESULTADOS COMPUTACIONAIS

A linguagem de programação utilizada foi JAVA, pois possui uma biblioteca de *strings* que oferece muitos recursos para facilitar a implementação de operações de propriedades algébricas.

As principais operações matemáticas implementadas foram: distributiva, associativa, comutativa, mínimo múltiplo comum, soma e reconhecedores de expressões.

Para a validação do resultados, foi proposta uma base de dados constituída de cinco funções, que estão descritas e provadas matematicamente por meio da indução matemática. Estas funções foram escolhidas pois já são utilizadas para exemplos em outros trabalhos da literatura sobre indução matemática [Antonio 2017], [Hefez 2009].

### 4.2 Base de dados e resultados obtidos

1. Soma dos  $n$  ímpares -  $P(n) 1 + 3 + \dots + (2n - 1) = n^2$

#### Prova por indução matemática

- I. Observe que  $P(1)$  é verdadeira, já que a fórmula é trivialmente válida para  $n = 1$ .
- II. Suponha que, para algum  $n$  natural,  $P(n)$  seja verdadeira; ou seja, que:

$$1 + 3 + \dots + (2n - 1) = n^2$$

Provando que  $P(n+1)$  é verdadeira. Somando  $2n+1$ , que é o próximo número ímpar, após  $2n-1$ , a ambos os lados da igualdade de  $P(n)$ , obtêm-se que a igualdade também verdadeira, pois:

$$1 + 3 + \dots + (2n - 1) + (2n + 1) = n^2 + (2n + 1) = (n + 1)^2 = n^2 + 2n + 1.$$

O Quadro 1 contém a saída da implementação para a função anterior.

```
f(n) = ((1n)(1n))/(1)
f(n+1) = (((1)(1n+1))((1)(1n+1)))/(1)
pn = (((2)(1n+1)-1)/(1)
Após o mínimo múltiplo comum:
pn = (((1)(2)(1n+1)-1)/(1)
f(n) = ((1)(1n)(1n))/(1)
f(n) + p(n+1) = f(n+1):
((1)(1n)(1n))/(1)+(((1)(2)(1n+1)-1)/(1) = (((1)(1n+1))((1)(1n+1)))/(1)
distributiva: (((1)(1n+1))((1)(1n+1)))/(1)
((1n+1)(1n+1))/(1)
(1nn+2n+1)/(1)
f(n+1) = (1nn+2n+1)/(1)

distributiva: ((1)(1n)(1n))/(1)
((1n)(1n))/(1)
(1nn)/(1)
distributiva: (((1)(2)(1n+1)-1)/(1)
(((2)(1n+1)-1)/(1)
((2n+2-1)/(1)
f(n) = (1nn)/(1)
pn = ((2n+2-1)/(1)
f(n) + pn = (1nn+2n+1)/(1)
```

Quadro 1. Resultados da soma dos n ímpares  $P(n)$

2. Soma dos n números naturais -  $P(n): 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$

## Prova por indução matemática

I. Observe que  $P(1)$  é verdadeira, pois:

$$P(1): 1 = \frac{1(1+1)}{2}.$$

II. Suponha que, para algum n natural,  $P(n)$  seja verdadeira, ou seja, que:

$$P(n): 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

Provando que  $P(n+1)$  é verdadeira. Somando  $n+1$  a ambos os lados da igualdade de  $P(n)$ , obtêm-se:

$$P(n+1): 1 + 2 + 3 + \dots + n + (n+1) = \frac{n(n+1)}{2} + (n+1)$$

desenvolvendo a segunda parte da igualdade tem-se:

$$\frac{n(n+1)}{2} + (n+1) = \frac{n(n+1) + 2(n+1)}{2} = \frac{(n+1)(n+2)}{2},$$

que mostra que  $P(n+1)$  também verdadeira.

Vale destacar que:

$$\frac{(n+1)(n+2)}{2} = \frac{n^2 + 3n + 2}{2}$$

No Quadro 2 está contida a saída da implementação para a função anterior.

```
f(n) = ((1n)(1n+1))/(2)
f(n+1) = (((1)(1n+1))((1)(1n+1)+1))/(2)
pn = p(n+1) - p(n) = (1n+1)/(1)
Após o mínimo múltiplo comum:
pn = (2)(1n+1)/(2)
f(n) = ((1)(1n)(1n+1))/(2)
f(n) + p(n+1) = f(n+1):
((1)(1n)(1n+1))/(2) + ((1)(1n+1))/2 = (((1)(1n+1))((1)(1n+1)+1))/(2)
distributiva: (((1)(1n+1))((1)(1n+1)+1))/(2)
((1n+1)(1n+1+1))/(2)
(1nn+3n+2)/(2)
f(n+1) = (1nn+3n+2)/(2)

distributiva: ((1)(1n)(1n+1))/(2)
((1n)(1n+1))/(2)
(1nn+1n)/(2)
distributiva: (2)(1n+1)/(2)
(2n+2)/(2)
f(n) = (1nn+1n)/(2)
pn = (2n+2)/(2)
f(n) + pn = (1nn+3n+2)/(2)
```

Quadro 2. Resultados da soma dos n números - P (n): 1 + 2 + 3 + ... + n

3. Soma dos quadrados -  $P(n): 1^2 + 2^2 + 3^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6}$

### Prova por indução matemática:

I. Observe que  $P(1)$  é verdadeira, pois:

$$P(1): 1 = \frac{1(1+1)(2(1)+1)}{6}$$

II. Suponha que, para algum  $n$  natural,  $P(n)$  seja verdadeira, ou seja, que:

$$P(n): 1^2 + 2^2 + 3^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Prova de que  $P(n+1)$  é verdadeira. Somando  $n+1$  a ambos os lados da igualdade de  $P(n)$ , obtêm-se:

$$P(n+1): 1^2 + 2^2 + 3^2 + \dots + n^2 + (n+1)^2 = \frac{n(n+1)(2n+1)}{6} + (n+1)^2,$$

desenvolvendo a segunda parte da igualdade tem-se:

$$\frac{n(n+1)(n+2)}{6} + (n+1) = \frac{n(n+1)(n+2) + 6(n+1)}{6} = \frac{(n+1)[n(n+2) + 6]}{6}$$

$$= \frac{(n+1)[(n+1)+1][2(n+1)+1]}{6} = \frac{(n+1)(n+2)(2n+3)}{6}$$

que mostra que  $P(n+1)$  também verdadeira.

Vale destacar que:

$$\frac{(n+1)(n+2)(2n+3)}{6} = \frac{2n^3 + 9n^2 + 13n + 6}{6}$$

O Quadro 3 contém a saída da implementação para a função anterior.

```
f(n) = ((1n)(1n+1)(2n+1))/(6)
f(n+1) = (((1)(1n+1))((1)(1n+1)+1)((2)(1n+1)+1))/(6)
pn = ((1n+1)(1n+1))/(1)
Após o mínimo múltiplo comum:
pn = ((6)(1n+1)(1n+1))/(6)
f(n) = ((1)(1n)(1n+1)(2n+1))/(6)
f(n) + p(n+1) = f(n+1):
((1)(1n)(1n+1)(2n+1))/(6) + ((6)(1n+1)(1n+1))/(6) = (((1)(1n+1))((1)(1n+1)+1)((2)(1n+1)+1))/(6)
distributiva: (((1)(1n+1))((1)(1n+1)+1)((2)(1n+1)+1))/(6)
((1n+1)(1n+1+1)((2)(1n+1)+1))/(6)
((1nn+3n+2)((2)(1n+1)+1))/(6)
((1nn+3n+2)(2n+2+1))/(6)
(2nnn+13n+9nn+6)/(6)
f(n+1) = (2nnn+13n+9nn+6)/(6)

distributiva: ((1)(1n)(1n+1)(2n+1))/(6)
((1n)(1n+1)(2n+1))/(6)
((1nn+1n)(2n+1))/(6)
(2nnn+1n+3nn)/(6)
distributiva: ((6)(1n+1)(1n+1))/(6)
((6n+6)(1n+1))/(6)
(6nn+12n+6)/(6)
f(n) = (2nnn+1n+3nn)/(6)
pn = (6nn+12n+6)/(6)
f(n) + pn = (2nnn+13n+9nn+6)/(6)
```

Quadro 3. Resultados da soma dos quadrados —  $P(n): 1^2 + 2^2 + 3^2 + \dots + n^2$

4. Soma das  $n$  frações  $P(n): \frac{1}{1(2)} + \frac{1}{2(3)} + \frac{1}{3(4)} + \dots + \frac{1}{n(n+1)} = \frac{n}{(n+1)}$

### Prova por indução matemática

I. Observe que  $P(1)$  é verdadeira, pois:

$$P(1): \frac{1}{1(2)} = \frac{1}{(1+1)}$$

II. Suponha que, para algum  $n$  natural,  $P(n)$  seja verdadeira, ou seja, que:

$$P(n): \frac{1}{1(2)} + \frac{1}{2(3)} + \frac{1}{3(4)} + \dots + \frac{1}{n(n+1)} = \frac{n}{(n+1)}$$

Provando que  $P(n+1)$  é verdadeira. Somando a ambos os lados da igualdade de  $P(n)$ , obtêm-se:

$$P(n+1): \frac{1}{1(2)} + \frac{1}{2(3)} + \frac{1}{3(4)} + \dots + \frac{1}{n(n+1)} + \frac{1}{(n+1)(n+2)}$$

$$= \frac{n}{(n+1)} + \frac{1}{(n+1)(n+2)}$$

desenvolvendo a segunda parte da igualdade tem-se:

$$\frac{n}{(n+1)} + \frac{1}{(n+1)(n+2)} = \frac{n(n+2)}{(n+1)(n+2)} + \frac{1}{(n+1)(n+2)} = \frac{(n+1)^2}{(n+1)(n+2)} =$$

$$= \frac{n+1}{n+2}$$

que mostra que  $P(n+1)$  também verdadeira.

O Quadro 4 contém a saída da implementação para a função anterior.

```
f(n) = (1n)/(1n+1)
f(n+1) = ((1)(1n+1))/((1)(1n+1)+1)
pn = (1)/((1n+1)(1n+2))
distributiva: ((1)(1n+1))
(1n+1)
distributiva: ((1)(1n+1)+1)
(1n+1+1)
f(n+1) = f(n+1) * (1n+1)(1n+1)
distributiva: (1n+1)(1n+1)
(1nn+2n+1)
distributiva: (1n+1+1)(1n+1)
(1nn+3n+2)
f(n+1) = (1nn+2n+1)/(1nn+3n+2)

f(n) = (1n)/(1n+1)
pn = (1)/((1n+1)(1n+2))
distributiva: ((1n)(1n+2))
(1nn+2n)
distributiva: ((1n+1)(1n+2))
(1nn+3n+2)
distributiva: (1)
distributiva: ((1n+1)(1n+2))
(1nn+3n+2)
f(n) = (1nn+2n)/(1nn+3n+2)
pn = (1)/(1nn+3n+2)
f(n) + pn = (1nn+2n+1)/(1nn+3n+2)
```

Quadro 4. Resultados da soma das  $n$  frações  $P(n): \frac{1}{1(2)} + \frac{1}{2(3)} + \frac{1}{3(4)} + \dots + \frac{1}{n(n+1)}$

Analisando o resultado obtido (Figura 7) com o resultado da prova por indução tem-se que:

$$\frac{n+1}{n+2} = \frac{(n+1)^2}{(n+1)(n+2)} = \frac{n^2+2n+1}{n^2+3n+2},$$

mostrando que o resultado da Figura 7 é equivalente.

5. Soma dos ímpares quadrados  $P(n): 1^2 + 3^2 + 5^2 + \dots + (2n-1)^2$

$$= \frac{1}{3}n(2n-1)(2n+1)$$

### Prova por indução matemática:

- a. Observe que  $P(1)$  é verdadeira, pois

$$P(1): 1^2 = \frac{1}{3} 1(2(1) - 1)(2(1) + 1).$$

- b. Suponha que, para algum  $n$  natural,  $P(n)$  seja verdadeira, ou seja, que:

$$P(n): 1^2 + 3^2 + 5^2 + \dots + (2n - 1)^2 = \frac{1}{3} n(2n - 1)(2n + 1),$$

Provando que  $P(n+1)$  é verdadeira. Somando  $(2n + 1)^2$  a ambos os lados da igualdade de  $P(n)$ , obtêm-se:

$$\begin{aligned} P(n+1): 1^2 + 3^2 + 5^2 + \dots + (2n - 1)^2 + (2n + 1)^2 &= \\ &= \frac{1}{3} n(2n - 1)(2n + 1) + (2n + 1)^2 \end{aligned}$$

desenvolvendo a segunda parte da igualdade tem-se:

$$\begin{aligned} \frac{1}{3} n(2n - 1)(2n + 1) + (2n + 1)^2 &= \frac{1}{3} (2n + 1)[n(2n - 1) + 3(2n + 1)] = \\ &= \frac{1}{3} (2n + 1)[n(2n - 1) + 6n + 3] = \frac{1}{3} (2n + 1)[n(2n - 1) + 4n + 2n + 3] = \\ &= \frac{1}{3} (2n + 1)[2n^2 - n + 4n + 2n + 3] = \frac{1}{3} (2n + 1)[2n^2 + 3n + 2n + 3] = \\ &= \frac{1}{3} (2n + 1)[n(2n + 3) + (2n + 3)] = \frac{1}{3} (2n + 1)[(2n + 3)(n + 1)] = \\ &= \frac{1}{3} (n + 1)(2n + 1)(2n + 3) \end{aligned}$$

que mostra que  $P(n+1)$  também verdadeira.

Vale destacar que:

$$\frac{1}{3} (n + 1)(2n + 1)(2n + 3) = \frac{4n^3 + 12n^2 + 11n + 3}{3}.$$

O Quadro 5 contém a saída da implementação para a função anterior.

$$f(n) = ((1n)(2n-1)(2n+1))/(3)$$

$$f(n+1) = (((1)(1n+1))((2)(1n+1)-1)((2)(1n+1)+1))/(3)$$

$$pn = (2n+1)(2n+1)/(1)$$

Após o mínimo múltiplo comum:

$$pn = (3)(2n+1)(2n+1)/(3)$$

$$f(n) = ((1)(1n)(2n-1)(2n+1))/(3)$$

$$f(n) + p(n+1) = f(n+1):$$

$$(((1)(1n)(2n-1)(2n+1))/(3)+(3)(2n+1)(2n+1)/(3) = (((1)(1n+1))((2)(1n+1)-1)((2)(1n+1)+1))/(3)$$

distributiva: (((1)(1n+1))((2)(1n+1)-1)((2)(1n+1)+1))/(3)

$$(((1n+1)((2)(1n+1)-1)((2)(1n+1)+1))/(3)$$

$$(((1n+1)(2n+2-1)(2n+2+1))/(3)$$

$$(((2nn+3n+1)(2n+2+1))/(3)$$

$$(4nnn+11n+12nn+3)/(3)$$

$$\mathbf{f(n+1) = (4nnn+11n+12nn+3)/(3)}$$
  

distributiva: ((1)(1n)(2n-1)(2n+1))/(3)

$$((1n)(2n-1)(2n+1))/(3)$$

$$((2nn-1n)(2n+1))/(3)$$

$$(4nnn-1n+0nn)/(3)$$

distributiva: (3)(2n+1)(2n+1)/(3)

$$(6n+3)(2n+1)/(3)$$

$$(12nn+12n+3)/(3)$$

$$f(n) = (4nnn-1n+0nn)/(3)$$

$$pn = (12nn+12n+3)/(3)$$

$$\mathbf{f(n) + pn = (4nnn+11n+12nn+3)/(3)}$$

Quadro 5. Resultados da soma dos ímpares quadrados  $- P(n): 1^2 + 3^2 + \dots + (2n - 1)^2$

### 4.3 Comentários gerais

Como verificou-se, por meio dos resultados obtidos, o algoritmo proposto conseguiu obter a equivalência entre algumas funções que trabalham com números naturais, cumprindo assim o objetivo deste projeto. Porém, ressalta-se que este esteve limitado a funções de baixa complexidade, ou seja, funções com sequências simples.

Com relação aos resultados vale ressaltar que apenas os obtidos na soma das  $n$  frações foram destoantes dos demais, porém matematicamente a equivalência foi comprovada.

## 5 | CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi estudado como automatizar a equivalência entre funções com números naturais, a fim de explicar processos de preservação do significado do código na etapa de otimização. Como contribuição, foi proposto um algoritmo com a possibilidade otimizar funções recorrentes, fazendo uso das operações básicas da matemática elementar (associativa, distributiva, comutativa, soma, dentre outras), já as equivalências entre funções foram provada por meio da indução matemática.

Considerando que para algumas funções não é trivial encontrar a sequência de operações matemáticas, que resultem numa igualdade com outra função, os testes executados ficaram restritos às funções contidas neste artigo.

A grande dificuldade deste estudo foi a análise e manipulação de expoentes e somatórios complexos, isto devido à complexidade da análise léxica e semântica, por isso foram implementados métodos limitados às funções relativamente simples.

Para trabalhos futuros pode ser estudado como implementar a técnica de Gauss na fase de otimização de código, no processo de compilação, a fim de transformar funções recorrentes em funções fechadas. Funções recorrentes possuem complexidade em função de  $n$ , enquanto que funções fechadas possuem complexidade constante, que considera-se um ganho em desempenho computacional significativo.

## REFERÊNCIAS

Aho, A. V. et al. Compiladores – Princípios, Técnicas e Ferramentas. Massachusetts: Pearson Addison – Wesley, 2ª edição, 2008.

Allen, F. O tratado fundamental sobre técnicas para otimização de loops – 1969. Disponível em: <<http://www.francesallen.com/>>. Acesso: junho/2017>. Data de acesso: 07/07/2017.

Antonio, A.F. - Sequências Indução Matemática – UFMG. Disponível em: <[http://homepages.dcc.ufmg.br/~loureiro/md/md\\_4SequenciaEInducaoMatematica.pdf](http://homepages.dcc.ufmg.br/~loureiro/md/md_4SequenciaEInducaoMatematica.pdf)>. Data de acesso: 07/07/2017.

Bertozzi, M. Broggi, A. Elsevier - Tools for code optimization and system evaluation of the image processing system PAPRICA-3 - UNIPR. 1999. Disponível em: <<http://www.ce.unipr.it/people/broggi/publications/jsa.pdf>>. Data de acesso: 07/07/2017.

De Gloria, A. Faraboschi, P. Elsevier: A boltzmann machine approach to code optimization 9ª edição, 1991. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167819105800420?via%3Dihub>>. Data de acesso: 07/07/2017.

Hazzan, S. Fundamentos de matemática elementar: combinatória e probabilidade. São Paulo: Atual, 7ª edição, 2004.

Hefez, A. UFF. OBMEP: Indução Matemática. 2009. Disponível em: <<http://www.obmep.org.br/docs/apostila4.pdf>>. Data de acesso: 07/07/2017.

IBM. Code optimization with the IBM XL compilers on Power architectures. 2016. Disponível em: <<http://www-01.ibm.com/support/docview.wss?uid=swg27005174&aid=1>>. Data de acesso: 07/07/2017.

Iezzi, G. Fundamentos de matemática elementar: complexos, polinômios e equações. São Paulo: Atual, 7ª edição, 2005.

Iezzi, G. e Murakami, C. Fundamentos de matemática elementar: conjuntos e funções. São Paulo: Atual, 8ª edição, 2004.

Menezes, P. B. Matemática discreta: para computação e informática. Porto Alegre: Sagra Luzzatto, 2ª edição, 2005.

## APÊNDICE

```
private static String addTerm(String f, int n, String nstr) {
    if (nstr.length() == 0) {
        if (!Character.isDigit(f.charAt(f.length() - 2))) {
            if (n >= 0) {
                return f.substring(0, f.length() - 1) + "+" + n + " ";
            }
            return f.substring(0, f.length() - 1) + n + " ";
        }
    }
    int i;
    for (i = f.length() - 2; Character.isDigit(f.charAt(i)); i--) {}
    if (f.charAt(i) == '-') {
        n = n + Integer.parseInt(f.substring(i, f.length()));
        return f.substring(0, i) + n + " ";
    } else {
        n = n + Integer.parseInt(f.substring(i + 1, f.length() - 1));
        return f.substring(0, i) + "+" + n + " ";
    }
}

int instr = -1;
for (int i = 0; i < f.length(); i++) {
    if (f.charAt(i) == 'n') {
        int x = nstr.length();
        instr = i;
        for (i = i; f.charAt(i) == 'n'; i++) {
            x--;
        }
        if (x == 0) {
            break;
        } else {
            instr = -1;
        }
    }
}

if (instr == -1) {
```

```

int cont;
for (int i = 0; i < f.length(); i++) {
    if (f.charAt(i) == 'n') {
        for (cont = 0; f.charAt(i) == 'n'; i++) {
            cont++;
        }
        if (cont < nstr.length()) {
            for (i = i; Character.isDigit(f.charAt(i)) || f.charAt(i) == 'n'; i--) {}
        }
        if (n < 0) {
            return f.substring(0, i) + n + nstr + f.substring(i, f.length());
        } else {
            return f.substring(0, i) + "+" + n + nstr + f.substring(i, f.length());
        }
    }
}
return "(" + n + nstr + f.substring(1);
} else {
    int i;
    for (i = instr - 1; i >= 0 && Character.isDigit(f.charAt(i)); i--) {}
    if (f.charAt(i) == '-') {
        n = n + Integer.parseInt(f.substring(i, instr));
        if (n >= 0) {
            f = f.substring(0, i) + "+" + n + f.substring(instr, f.length());
        } else {
            f = f.substring(0, i) + n + f.substring(instr, f.length());
        }
    } else {
        n = n + Integer.parseInt(f.substring(i + 1, instr));
        f = f.substring(0, i) + "+" + n + f.substring(instr, f.length());
    }
}
return f;
}
}

```

```
private static String distr(String f) {
```

```

f = f.replace("(", " "), (")");
String parts[] = f.split(",");
f = "()";
for (int i = 0; i < parts[0].length(); i++) {
    for (i = i; i < parts[0].length() && !Character.isDigit(parts[0].charAt(i));
i++) {}
    int j;
    for (j = i; j < parts[0].length() && Character.isDigit(parts[0].charAt(j)); j++)
    {}
    if (j > i) {
        int n = Integer.parseInt(parts[0].substring(i, j));
        String nstr = "";
        for (int temp = j; parts[0].charAt(temp) == 'n'; temp++) {
            nstr = nstr + "n";
        }
        for (int k = 0; k < parts[1].length(); k++) {
            for (k = k; k < parts[1].length() &&
!Character.isDigit(parts[1].charAt(k)); k++) {}
            int l;
            for (l = k; l < parts[1].length() &&
Character.isDigit(parts[1].charAt(l)); l++) {}
            String nstr1 = "";
            for (int temp = l; temp < parts[1].length() && parts[1].charAt(temp) ==
'n'; temp++) { nstr1 = nstr1 + "n"; }
            if (l > k) {
                int n1 = Integer.parseInt(parts[1].substring(k, l)) * n;
                if ((parts[0].charAt(i - 1) == '+' && (parts[1].charAt(k - 1) == '+' ||
parts[1].charAt(k - 1) == '(')) || ((parts[0].charAt(i - 1) == '+' || parts[0].charAt(i - 1)
== '(') && parts[1].charAt(k - 1) == '-')) { n1 *= -1; }
                nstr1 = nstr + nstr1; f = addTerm(f, n1, nstr1);
            }
            k = l;
        }
        i = j;
    }
}
if (f.charAt(1) == '+') {

```

```

    f = "(" + f.substring(2);
}
return f;
}

```

```

private static int middleIndex(String f) {
    String aux = f;
    int middle = f.indexOf("(");
    while (middle != -1) {
        if (f.charAt(middle - 1) == ')' || f.charAt(middle + 2) == '(') {
            aux = aux.substring(0, middle) + "$" + aux.substring(middle + 1,
aux.length());
            middle = aux.indexOf("(");
        } else {
            break;
        }
    }
    return middle;
}

```

```

public static String distributive(String f) {
    System.out.println("distributiva: "+f);
    int middle = middleIndex(f);
    int start = 0, end = 0;
    f = checkparents(f);
    while (middle != -1) {
        for (int i = middle; i >= 0; i--) {
            if (f.charAt(i) == '(') {
                start = i;
                break;
            }
        }
        for (int i = middle + 1; i < f.length(); i++) {
            if (f.charAt(i) == ')') { end = i; break; }
        }
        String aux = f.substring(start, middle + 1) + f.substring(middle + 1, end +
1);
    }
}

```

```

    f = "(" + f.substring(2);
}
return f;
}

```

```

private static int middleIndex(String f) {
    String aux = f;
    int middle = f.indexOf("(");
    while (middle != -1) {
        if (f.charAt(middle - 1) == ')' || f.charAt(middle + 2) == '(') {
            aux = aux.substring(0, middle) + "$" + aux.substring(middle + 1,
aux.length());
            middle = aux.indexOf("(");
        } else {
            break;
        }
    }
    return middle;
}

```

```

public static String distributive(String f) {
    System.out.println("distributiva: "+f);
    int middle = middleIndex(f);
    int start = 0, end = 0;
    f = checkparents(f);
    while (middle != -1) {
        for (int i = middle; i >= 0; i--) {
            if (f.charAt(i) == '(') {
                start = i;
                break;
            }
        }
        for (int i = middle + 1; i < f.length(); i++) {
            if (f.charAt(i) == ')') { end = i; break; }
        }
        String aux = f.substring(start, middle + 1) + f.substring(middle + 1, end +
1);
    }
}

```

```

        f = f.replace(aux, distr(aux));
        f = checkparents(f);
        System.out.println(f);
        middle = middleIndex(f);
    }
    System.out.println("");
    return f;
}

private static String checkparents(String f) {
    for (int i = 1; i < f.length(); i++) {
        if (f.charAt(i) == '(' && f.charAt(i - 1) == '(') {
            for (int j = i + 1; j < f.length() - 1; j++) {
                if (f.charAt(j) == ')' && f.charAt(j + 1) == ')') {
                    f = f.substring(0, i) + f.substring(i + 1, j) + f.substring(j + 1,
f.length());
                    i = j;
                    break;
                } else if (f.charAt(j) == ')' && f.charAt(j + 1) != ')') {
                    break;
                }
            }
        }
    }
}

for (int i = 1; i < f.length(); i++) {
    if (f.charAt(i) == '(') {
        for (int j = i + 1; j < f.length() - 1; j++) {
            if (f.charAt(j) == ')') {
                if (f.charAt(i - 1) == '(' && (f.charAt(j + 1) == '+' || f.charAt(j + 1)
== '-')) {
                    f = f.substring(0, i) + f.substring(i + 1, j) + f.substring(j + 1,
f.length());
                } else {
                    break;
                }
            }
        }
    }
}
}

```

```

    int j;
    for (j = i - 1; Character.isDigit(j); j--) {
    }
    int k;
    for (k = i + 1; Character.isDigit(k); k++) {
    }
    int newn = Integer.parseInt(f.substring(j, i))
Integer.parseInt(f.substring(i + 1, k + 1));

    f = f.substring(0, j) + newn + f.substring(k + 1, f.length());
    }
}

return f;
}

```

```

public static String mmc(String f, String fl) {
    int a0 = -1;
    int a1 = -1;
    int b0 = -1;
    int b1 = -1;
    for (int i = 0; i < f.length() - 1; i++) {
        if (f.charAt(i) == '/') {
            for (i = i; i < f.length(); i++) {
                if (f.charAt(i) == '(') { a0 = i; }
                if (f.charAt(i) == ')') {
                    a1 = i + 1;
                    break;
                }
            }
            break;
        }
    }

    for (int i = 0; i < fl.length() - 1; i++) {
        if (fl.charAt(i) == '/') {

```

```

    for (i = i; i < f1.length(); i++) {
        if (f1.charAt(i) == '(') {
            b0 = i;
        }
        if (f1.charAt(i) == ')') {
            b1 = i + 1;
            break;
        }
    }
    break;
}
}
String a = f.substring(a0, a1);
String b = f1.substring(b0, b1);
String result = distr(a + b);
f = f.substring(0, a0) + result + f.substring(a1, f.length());
int i;
for (i = 0; f.charAt(i) == '('; i++) { }
f = f.substring(0, i - 1) + b + f.substring(i - 1, f.length());
return f;
}
}

```

```

private static String add2(String f, String f1) {
    String temp = f.substring(0, f.indexOf("/"));
    for (int i = 1; i < f1.indexOf("/"); i++) {
        int j;
        for (j = i; Character.isDigit(f1.charAt(j)); j++) { }
        int n;
        if (j > i) {
            if (f1.charAt(i - 1) == '-') {
                n = Integer.parseInt(f1.substring(i - 1, j));
            } else {
                n = Integer.parseInt(f1.substring(i, j));
            }
        }
        String nstr = "";
        int l;
    }
}

```

```

        for (l = j; f.charAt(l) == 'n'; l++) {
            nstr += "n";
        }
        temp = addTerm(temp, n, nstr);
    }
    i = j;
}
return temp + f.substring(f.indexOf("/"));
}

```

```

public static String npn1(String f) {
    for (int i = 0; i < f.length(); i++) {
        if (f.charAt(i) == 'n') {
            int aux = 0;
            for (int j = i - 1; j >= 0; j--) {
                if (!Character.isDigit(j)) {
                    f = f.substring(aux, j) + "(" + f.substring(j, i) + ")(1n+1)" +
                        f.substring(i + 1, f.length());
                    aux = i + 1;
                    i += 6;
                    break;
                }
            }
        }
    }
    return f;
}
}

```

```

public static void inducao2(String f, String pn) {
    String passoIndutivo = npn1(f);
    System.out.println("f(n+1) = "+passoIndutivo);
    System.out.println("f(n) = "+f);
    System.out.println("p(n+1) = " + pn);
    String divIndutive[] = passoIndutivo.split("/");
    divIndutive[0] = distributive(divIndutive[0]);
    divIndutive[1] = distributive(divIndutive[1]);
    String aux = divIndutive[0];
    divIndutive[0] = distributive(divIndutive[0] + aux);
    divIndutive[1] = distributive(divIndutive[1] + aux);
    System.out.println("f(n+1) = " + divIndutive[0] + "/" + divIndutive[1]);
    System.out.println("");
    String divf[] = f.split("/");
    String divfn[] = pn.split("/");
    System.out.println("f(n) = " + divf[0] + "/" + divf[1]);
    System.out.println("p(n+1) = " + divfn[0] + "/" + divfn[1]);
    System.out.println("");
    if (!divf[1].equals(divfn[1])) {
        String temp = checkparents(divfn[1].replace(divf[1], ""));
        if (temp.length() != divfn.length) {
            divf[1] = "(" + divf[1] + temp + ")";
            divfn[0] = "(" + divfn[0] + temp + ")";
        }
    }
    divf[0] = distributive(divf[0]);
    divf[1] = distributive(divf[1]);
    divfn[0] = distributive(divfn[0]);
    divfn[1] = distributive(divfn[1]);
    System.out.println("f(n) = " + divf[0] + "/" + divf[1]);
    System.out.println("p(n+1) = " + divfn[0] + "/" + divfn[1]);
    System.out.print("f(n) + p(n+1) = ");

    System.out.println(add2(divf[0] + "/" + divf[1], divfn[0] + "/" + divfn[1]));
}

```

```
System.out.println("0");
FunctionOperations.inducao("((1n)(1n))/(1)", "(((2)(1n+1)-1)/(1)");
System.out.println("1");
FunctionOperations.inducao("((1n)(1n+1)(2n+1))/(6)", "((1n+1)(1n+1))/(1)");
System.out.println("2");
FunctionOperations.inducao2("(1n)/(1n+1)", "(1)/((1n+1)(1n+2))");
System.out.println("4");
FunctionOperations.inducao("((1n)(2n-1)(2n+1))/(3)", "(2n+1)(2n+1)/(1)");
System.out.println("6");
FunctionOperations.inducao("((1n)(1n+1))/(2)", "(1n+1)/(1)");
```

## ÍNDICE REMISSIVO

### A

Acoplamento termomecânico 44, 48, 52

Algoritmo genético (AG) 244

Alvenaria estrutural 4, 44, 48

Análise de imagem 235, 240, 241

Aprendizado de máquina 2

Arduino 17, 18, 19, 20, 141, 142, 144, 145, 146, 147, 148, 152, 154, 157, 158, 159, 160, 161

Arquitetura de software 5, 74, 75, 76

### B

Balanced spaces 34

Biblioteconomia clínica 21

Bluetooth 141, 142, 143, 144, 146, 147, 148, 151, 152, 154, 155, 156, 157, 158, 159, 160, 177

### C

Cenários arquiteturais 5, 74, 87

Ciclo de vida arquitetural 74, 76, 77, 85, 87

Comunicação científica 3, 5, 57, 58

Conjuntos de similaridade 5, 105, 107, 108, 116

Correlação 235, 236, 240

### D

Dados complexos 105, 106, 107, 108

Design science research 57, 58, 59, 62

Desigualdade de gênero na TI 173, 174

Dibujo asistido por computadora 6, 162, 163, 164, 171

### E

Educación a distancia 162, 164, 165, 168, 170, 171

Elementos finitos 3, 48, 52, 53, 223

Energia renovável 185

Equivalência de funções 6, 118

## F

Fibra de carbono 223

## G

Gêmeo digital 5, 63, 64, 68, 71

Grafos 105, 112, 259, 261

## H

Herramientas tecnológicas 6, 162, 163, 164, 170

Histórico feminino na TI 173, 174

Human comfort 198

## I

Identificação de sistemas 185, 188, 189

Idosos 16, 17, 20

Indústria 4.0 63, 65, 66, 67

Infecções por Coronavirus 2

Interoperabilidade 21, 23, 24, 25, 26, 30, 32, 63, 64, 66, 67

## J

JavaCV 235, 236, 237, 240, 241

JavaScript 141, 142, 153, 263

## L

Ligas de alumínio 223

## M

Memorandos técnicos 5, 74, 76, 78, 80, 81, 86, 87

Método sem malha local 243, 244

Método sem malha local com integração reduzida (ILMF) 244

Métrica de distância 5, 105, 113, 116

Microcontrolador 17, 141, 152

Mixed finite elements 34

Mulheres na TI 173, 174, 182, 183

Mulheres nos cursos superiores de TI 173, 174

## O

Ontologias 21, 22, 23, 24, 25, 29, 30, 31, 32

opencv 241

OpenCV 235, 236, 237, 240, 241

Optimal detailing 89

## **P**

Poisson's equation 34, 36

Prestressed concrete 89, 90, 91, 92, 96, 103

## **R**

Rami 4.0 65

RAMI 4.0 63, 64, 65, 66, 67, 68, 69, 71

Realidade aumentada 3, 5, 57, 58, 60, 62

Remédios 3, 4, 16, 17, 20

Resistência ao fogo 44, 45, 49, 50, 56

Resistência mecânica 50, 55, 223

Robotista 63

## **S**

Sistemas ciberfísicos (CPS) 63, 64, 71

Static condensation 4, 34, 35, 36

Steel-concrete 6, 198, 199, 200, 202, 204, 205, 206, 216, 218, 221

## **T**

Terminologias clínicas 4, 21, 23, 24, 25, 30

Teste de hipótese 105

## **U**

Usinas eólicas 185

## **V**

Vibrations 6, 198, 199, 212, 219, 220, 222

Visões do modelo 4+1 5, 74, 87

Visualização de dados 57

## **W**

Wi-Fi 141, 142, 147, 148, 152, 153, 157, 158

COLEÇÃO

# DESAFIOS DAS ENGENHARIAS:

## ENGENHARIA DE COMPUTAÇÃO 3

-  [www.atenaeditora.com.br](http://www.atenaeditora.com.br)
-  [contato@atenaeditora.com.br](mailto:contato@atenaeditora.com.br)
-  [@atenaeditora](https://www.instagram.com/atenaeditora)
-  [www.facebook.com/atenaeditora.com.br](https://www.facebook.com/atenaeditora.com.br)

COLEÇÃO

# DESAFIOS DAS ENGENHARIAS:

ENGENHARIA DE COMPUTAÇÃO 3

- 🌐 [www.atenaeditora.com.br](http://www.atenaeditora.com.br)
- ✉ [contato@atenaeditora.com.br](mailto:contato@atenaeditora.com.br)
- 📷 @atenaeditora
- 📘 [www.facebook.com/atenaeditora.com.br](https://www.facebook.com/atenaeditora.com.br)