UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE EDUCAÇÃO E SAÚDE
UNIDADE ACADÊMICA DE FÍSICA E MATEMÁTICA
CURSO DE LICENCIATURA EM MATEMÁTICA

JOÃO ELDER LAURENTINO DA SILVA

COMPARANDO AS LINGUAGENS DE PROGRAMAÇÃO FORTRAN E PYTHON ATRAVÉS DE PROBLEMAS MATEMÁTICOS

CUITÉ - PB
2019

JOÃO ELDER LAURENTINO DA SILVA

COMPARANDO AS LINGUAGENS DE PROGRAMAÇÃO FORTRAN E PYTHON ATRAVÉS DE PROBLEMAS MATEMÁTICOS

Monografia apresentada à Banca Examinadora, como exigência parcial à conclusão do Curso de Licenciatura em Matemática, da Universidade Federal de Campina Grande Campus Cuité.

Orientador: Prof. Dr. Aluizio Freire da Silva Junior

Coorientador: Prof. Dr. Vladimir Soares Catão

CUITÉ – PB

2019

FICHA CATALOGRÁFICA ELABORADA NA FONTE
Responsabilidade Rosana Amâncio Pereira – CRB 15 – 791

S586c Silva, João Elder Laurentino da.

Comparando as linguagens de programação Fortran e Python através de problemas matemáticos. / João Elder Laurentino da Silva – Cuité: CES, 2019.

66 fl.

Monografia (Curso de Licenciatura em Matemática) – Centro de Educação e Saúde / UFCG, 2019.

Orientação: Dr. Aluizio Freire da Silva Junior
Coorientação: Dr. Vladimir Soares Catão

1. Matemática. 2. Informática. 3. Fortran. 4. Python. 5. Problemas matemáticos. I. Título.

Biblioteca do CES – UFCG CDU 004.43:51

JOÃO ELDER LAURENTINO DA SILVA

COMPARANDO AS LINGUAGENS DE PROGRAMAÇÃO FORTRAN E PYTHON ATRAVÉS DE PROBLEMAS MATEMÁTICOS

Monografia apresentada à Banca Examinadora, como exigência parcial à conclusão do Curso de Licenciatura em Matemática, da Universidade Federal de Campina Grande Campus Cuité.

Aprovada em: 26/06/2019

BANCA EXAMINADORA

Prof. Dr. Aluizio Freire da Silva Junior
(Orientador)

Prof. Dr. Vladimir Soares Catão
(Coorientador)

Prof.ª Dr.ª Célia Maria Rufino Franco
(Examinadora interna)

CUITÉ - PB

2019

Este trabalho é dedicado primeiramente a Deus, por ser perfeito. As pessoas mais importantes da minha vida: meus pais, Maria José e José Luiz que são minha base. A minha namorada, Aline Santos que tanto me ajuda a superar todas as dificuldades. E também meus irmãos, Marcos Jair e Jardel Luiz pelo grande apoio.

AGRADECIMENTOS

Primeiramente a Deus, pelo dom da vida.

A meus pais, Maria José e José Luiz por serem minha base e meu exemplo para vida.

A minha namorada, Aline Santos, por todo carinho, incentivo e apoio.

A meus irmãos, Marcos Jair e Jardel Luiz pelo apoio de sempre.

Ao Prof. Dr. Aluisio Freire e o Prof. Vladimir Catão, pela orientação, confiança e incentivo depositados em mim, e extrema dedicação a este trabalho.

A meu amigo, Luciano Brito, pela colaboração no desenvolvimento deste trabalho.

A UFCG-CES, professores, direção e administração que me proporcionaram uma oportunidade de aprendizagem única.

A meus colegas universitários, pelo companheirismo durante todo o curso.

A todos que fizeram parte da minha formação, o meu muito obrigado.

A Matemática é o alfabeto com o qual Deus escreveu o Universo.

Galileu Galilei

RESUMO

Este trabalho apresenta a comparação entre duas linguagens de programação (Fortran e Python) através de problemas matemáticos. Esse estudo é importante devido ao grande uso de recursos computacionais nas diversas áreas da Matemática, tanto em setores acadêmicos como comercias. Por este motivo, o objetivo do trabalho é utilizar a Informática como ferramenta para fazer cálculos matemáticos, conhecer as linguagens Fortran e Python e comparar seus desempenhos em relação à velocidade de processamento, precisão nos cálculos e usabilidade da linguagem. Assim sendo, verificamos que dependendo da situação-problema, os resultados variam: em algumas situações, o Fortran mostrou-se melhor; em outras, o Python. Devido à gama de possibilidades, esse trabalho pode servir como base para mais estudos posteriores sobre o tema e, podendo ser complementado com os mais diversos problemas matemáticos. Neste contexto, observa-se que não existe necessariamente uma linguagem melhor que outra, deve-se buscar aquela mais adequada para o tipo de problema em questão.

Palavras-chave: Matemática. Informática. Fortran. Python. Problemas Matemáticos.

ABSTRACT

This work introduces a comparison between two programming languages (Fortran and Python) through mathematical problems. This study is important due to the great utilitisation of computational resources on the various areas of Mathematics, in academic as well commercial sectors. For this reason, the goal of this work is to use informatics as a tool to make mathematical calculations, to know the technologies Fortran and Python and to compare their performances regarding processing speed, accuracy in calculations and language usability. Therefore, we verified that depending on the problem situation, there's a variation in the results: in some situations, Fortran proved to be better; in others, Python. Due to the range of possibilities, this work can serve as base to more further studies about the theme and can be complemented with more diversified mathematical problems. In this context, it is observed that there is not necessarily a better language than another, one should seek the most appropriate for the type of problem in question.

Keywords: Mathematics. Informatics. Fortran. Python. Mathematical Problems.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

ANSI - American National Standards Institute.

CPU - Central Processing Unit.

HPF - High Performance Fortran.

IBM - International Business Machines.

IEC - International Electrotechnical Commission.

ISO - International Organization for Standardization.

JTC - Joint Technical Committee.

WG5 - Working Groups 5.

SUMÁRIO

1 INTRODUÇÃO

A tecnologia nunca esteve tão evoluída como atualmente; com a globalização, estamos cada vez mais tendendo a uma espécie de "era digital", onde, os avanços tecnológicos se fazem presentes no cotidiano da grande maioria das pessoas. Fernandes (2004, p. 43) complementa: "Hoje, com a velocidade de processamento e distribuição de informações via rede virtual, o computador tornou-se um instrumento indispensável para as realizações humanas." Nesse sentido, podemos observar quão rápido a tecnologia se difundiu e evoluiu, ao pegarmos a vida habitual das pessoas pouco mais de meio século atrás, quando não havia computadores, celulares e nem Internet.

Quanto às pesquisas e estudos, em quaisquer que sejam os níveis, eram feitas através de livros, enciclopédias, revistas, jornais e afins, podendo até conter informações desatualizadas e incorretas. Porém, atualmente, bastamos procurar o que queremos na Internet e automaticamente teremos um vasto acervo de informações à nossa disposição.

O mesmo ocorria para realização de estudos matemáticos, os acessos a informações mais confiáveis eram limitados exclusivamente a livros e, cálculos matemáticos eram restringidos a calculadoras pouco potentes e de baixa precisão. Todavia, com a evolução da tecnologia e, por sua vez, da Informática, cálculos extremamente complexos e, muitas vezes impossíveis de serem realizados à mão, tornaram-se atividades extremamente simples com a utilização do computador.

Portanto, como a Matemática é um elemento essencial em vários campos das atividades humanas e a Informática uma ferramenta poderosíssima nessa atividade, principalmente para cálculos matemáticos, iremos utilizar a Informática como ferramentas para estudar Matemática, através da comparação de duas linguagens de programação (Fortran e Python) em relação a seu desempenho, a fim de analisar qual das duas convém ser mais utilizada para cálculos matemáticos, levando em consideração os exemplos específicos desse trabalho.

Deste modo, através de exemplos matemáticos, considerou-se estudar o Fortran, que apesar de ser uma linguagem de programação bem mais antiga, ainda continua sendo muito utilizada na área da Matemática, e o Python por sua vez, uma linguagem bem mais

atual, mas um pouco menos utilizada nessa área, para uma análise sobre qual linguagem melhor se destaca nos estudos de cálculos matemáticos.

Nesse sentido, delineamos o seguinte objetivo geral da pesquisa: Investigar historicamente conceitos da Matemática e Informática e, então, de forma prática, comparar o desempenho das linguagens de programação Fortran e Python sob as mesmas condições inicias. E, para uma resposta mais consistente desse objetivo geral, traçamos os seguintes objetivos específicos: utilizar a Informática como ferramenta para fazer cálculos matemáticos, conhecer as linguagens Fortran e Python, além de comparar o desempenho entre ambas, no que se refere à velocidade de processamento, precisão dos cálculos e usabilidade da linguagem, isto é, qual das duas seria mais fácil de aprender.

Esse estudo torna-se de grande importância devido ao alto grau de usabilidade de recursos computacionais nas áreas de ensino da Matemática e em vários ramos das engenharias, por exemplo, que utilizam programas computacionais para modelarem situações reais, transformando em modelos computacionais que resolvem problemas diversos, contribuindo imensamente no âmbito comercial.

Essa pesquisa deu-se inicialmente de forma mais teórica, através de pesquisas bibliográficas para melhor entendimento do nosso objeto de estudo, em seguida utilizamos as linguagens Fortran e Python para testar modelos matemáticos e medir seus desempenhos, tal como, velocidade de processamento, precisão em cálculos e facilidade de construção de programas. Para comparar o desempenho dos dois programas foram estabelecidas as mesmas condições iniciais. O que são requisitos importantes quando se está utilizando um recurso computacional para resolver um problema matemático.

Desse modo, o presente trabalho foi estruturado num texto em sete capítulos, além das referências bibliográficas, apêndices e anexos. Nesta seção (Capítulo I), estão descritos os objetivos e uma visão geral do que está por vir no trabalho. Já no Capítulo II os temas tratados são a Matemática e a Informática, onde iremos falar do que é um programa de computador, lógica e linguagem de programação, além da relação da Matemática com a lógica de programação. E, então falaremos de como o computador é utilizado em áreas de aplicação da Matemática, ou seja, quais suas relações e a importância de estudar Matemática por meio de recursos computacionais.

No Capítulo III abordaremos um pouco do contexto histórico das linguagens utilizadas, apresentaremos informações gerais sobre cada linguagem trabalhada, nesse caso, o Fortran e o Python. Já no Capítulo IV é apresentado um breve histórico de cada problema estudado, isto é: Geração de dados em um arquivo através de funções, Elementos da esfera em uma matriz, Obtenção da raiz de funções através do Método da Bisseção e Aproximação do limite de uma série geométrica.

No Capítulo V, trataremos a metodologia com que foi desenvolvida este trabalho, isto é, como foi dada a comparação nos problemas matemáticos, comparação dos programas, informações dos métodos utilizados para realização do estudo e organização dos dados da pesquisa, tal como dados do computador utilizado e dos tradutores de linguagem.

O Capítulo VI, aborda os resultados e discursões da pesquisa, trazendo todos os resultados obtidos e comentários acerca dos temas. E, finalmente, no Capítulo VII teremos as considerações finais, com o objetivo de fazer uma reflexão sobre os elementos estudados no trabalho.

2 A MATEMÁTICA E A INFORMÁTICA

A Matemática e a Informática são duas áreas do conhecimento de extrema importância nos dias atuais. A primeira é uma área historicamente antiga e a segunda, uma área razoavelmente recente. Então, o objetivo deste capítulo é fazer uma exploração básica sobre essas áreas do conhecimento e apontar ligações entre ambas.

2.1 A MATEMÁTICA

Hoje em dia, onde quer que estejamos lá está ela, a Matemática, sejam telefones, senhas, nossa idade, preços de tudo que compramos ou vendemos, é representada por símbolos matemáticos.

A história da Matemática diz respeito à história da própria humanidade. Lopes (2016) afirma que a palavra Matemática tem sua "raiz grega *math*, ligada a noções como *aprender* e *conhecer*. Dessa raiz, muitas palavras são derivadas. [...]" O autor ainda diz que "[...] do adjetivo *mathematiké* derivamos o substantivo plural *mathematiká,* que se traduz como *as coisas cognoscíveis*.". Como vimos, o termo Matemática tem origem grega e pode ser entendida como as coisas que podem ser conhecidas, usando também o sentido de aprender e conhecer.

Todavia, a Matemática é algo mais complexo para denotarmos apenas por um simples termo como "conhecimento". A partir desse precedente, podemos deduzir que ela teve um papel central no modo de como o homem entende o mundo, o que levou os gregos assim a tratá-la como uma espécie de essência do conhecimento.

Feynman (1989) complementa que "[...] A Matemática não é apenas outra linguagem: é uma linguagem mais o raciocínio; é uma linguagem mais a lógica, é um instrumento para raciocinar". Entre várias e várias outras definições, podemos dizer que a Matemática é uma ciência que usa a lógica e outros artifícios para resolver problemas tal como desenvolver teses e hipóteses.

Nesse sentido, devido à grande importância da Matemática, foram criadas, ao longo dos tempos, diversas tecnologias que nos ajudam a entender, explicar, conhecer e a manipular mais facilmente diversas situações de cálculos, como calculadoras, computadores, celulares, etc. Mostrando quão a Matemática e a tecnologia estão associadas.

2.2 A INFORMÁTICA

Em relação à Informática, Norton (1977) conta que devido ao crescimento populacional, globalização e desenvolvimento do capitalismo no século XX, as pessoas passam a precisar cada vez mais de recursos que facilitem as suas vidas. A quantidade de dados e informações aumentando e o papel se tornando um meio inviável para um volume tão imenso de informações. Assim, a Informática surge, para de ajudar o ser humano a registrar e manipular grandes volumes de dados com precisão e velocidade.

Segundo Castells (2000, p. 37) "a inovação tecnológica e a transformação organizacional com enfoque na flexibilidade e na adaptabilidade foram absolutamente cruciais para garantir a velocidade e a eficiência da reestruturação", nesse sentido percebemos o quanto a tecnologia foi importante para evolução global.

Ainda falando sobre o contexto histórico, o termo Informática foi criado em 1957 pelo cientista da computação Karl Steinbruch e diz respeito ao processamento da informação, que segundo Luna:

> [...] o termo informática começou a ser amplamente difundido a partir de 1962, na França, e sua conotação mais conhecida é a divulgação da contração das palavras INFORmation e autoMATIQUE (INFORMAÇÃO AUTOMÁTICA). O objetivo inicial dessa ciência é auxiliar o ser humano nos trabalhos rotineiros e repetitivos, em geral cálculos e gerenciamento. Atualmente, o termo informática é comumente utilizado para se referir à manipulação e gênese da informação por meio de computadores, que são os responsáveis diretos pelo processamento dos dados (Luna (2009, p.18, apud ALCALDE, 1991)

Logo, a Informática é uma ciência responsável pelo processamento automático de informação. De maneira prática, a Informática é um campo do conhecimento que relaciona as necessidades humanas por meio da construção de interfaces em tecnologias voltadas para a solução ou para a automatização de problemas, causando assim, maior agilidade nos serviços e redução de ocorrências de falhas humanas.

A Informática é ainda um conjunto de ciências que utilizam meios digitais, estando neste grupo: a ciência da computação, os sistemas de informação, a teoria da informação, o processo de cálculo, a análise numérica e os métodos teóricos da representação dos conhecimentos e da modelagem dos problemas.

É importante abordarmos algumas áreas básicas sobre a Informática e, nada melhor, do que começar com a própria "Informática básica", que se trata do conjunto de conhecimentos e habilidades essenciais na área de Informática.

Um computador é composto basicamente por dois grupos, o *hardware* e o *software*. Segundo Keen (1996) o *hardware* é a parte física do computador, ou seja, o conjunto de peças e equipamentos que fazem o computador funcionar, além de aparatos eletrônicos acoplados em produtos computacionais.

E Maran (1999) diz que o *software* é a parte lógica do computador. É basicamente o cérebro do computador, ou seja, é o conjunto de instruções eletrônicas que dizem ao computador o que fazer.

Segundo Rezende e Abreu (2000), os softwares podem ser classificados em dois grupos *Softwares Básico/Sistemas* que dá permissão ao usuário se relacionar diretamente com o computador e suas partes, como por exemplo, os *firmwares*[1]e *drivers*[2], e, os *Softwares Aplicativos* que permite que o usuário faça uma tarefa específica como, por exemplo, editar texto, criar slides, planilhas eletrônicas, jogar etc.

Filho et al. (2008, p.132) informa que "Além desses elementos, o sistema de computador, como um todo, é composto por diversas partes eletrônicas e mecânicas, dentre as quais pode-se destacar: a Unidade Central de Processamento, conhecida como (CPU). "

Com tantas ferramentas já citadas anteriormente, os meios tecnológicos são métodos indispensáveis para todos os ramos, seja no trabalho, na escola e até mesmo no lazer. Então, vamos tomar como exemplo, a área da educação.

A Informática na educação refere-se ao uso de tecnologias da informação como: computadores, celulares e *tablets*, para objetivos pedagógicos. O propósito fundamental da Informática na educação é valer-se de recursos digitais, como ferramentas para melhorar o método de ensino-aprendizagem. Porém, a Informática educativa não se restringe apenas ao

---

[1]Também conhecidos pela nomenclatura "software embarcado", os Firmware são um conjunto de instruções operacionais que são programadas diretamente no hardware de equipamentos eletrônicos. Disponível

[2]Drivers são programas responsáveis pela comunicação entre o sistema operacional de computador e o hardware conectado a ele. Disponível em: https://www.techtudo.com.br/artigos/noticia/2013/06/entenda-que-sao-drivers-para-que-servem-e-como-instala-los.html.

processo de ensino-aprendizagem, numa relação mais ampla, ela também auxilia áreas administrativas por meio da informatização de dados como frequência de alunos, diários de notas, publicação de calendários escolares, comunicação com pais e alunos, etc.

Em suma, o mundo atualmente, gira em torno da Informática, e não é à toa que é uma das áreas que mais crescem economicamente em todo mundo.

2.3 PROGRAMA DE COMPUTADOR

Já falamos que para realizar nossas atividades no computador são necessários *softwares* que se comunicam com o computador. Outro ponto que é importante frisar, é que o termo *software* também pode referir-se a programas ou sistemas computacionais.

Um programa de computador é um grupo de instruções que apresentam um trabalho a ser realizado por um computador. Mas até surgir os programas como conhecemos hoje em dia, vários outros eventos ocorreram para o aprimoramento desse conteúdo.

A história conta que a primeira aplicação prática da programação deu-se com aparelhos baseados em cartões perfurados, que eram usados desde o século XVIII na indústria da confecção. Complementa Costa:

> Tomando como ponto de partida o invento do tecelão francês Joseph-Marie Jacquard, que no início do século XIX construiu um tear programável que usava cartões perfurados, este trabalho procura mostrar relações da programação mecânica do tear com a programação dos computadores atuais. (COSTA, 2008, p. 4)

Daí então, o estadunidense Hermann Hollerith utilizou de alguma forma a ideia no processamento dos dados do censo dos Estados Unidos no ano de 1890. Comenta Costa:

> [...] o censo de 1890 nos Estados Unidos da América foi realizado com a utilização da máquina leitora de cartões perfurados desenvolvida por Hollerith. Os cartões contendo os dados de cada cidadão codificados nos furos eram lidos, interpretados e sumarizados automaticamente. Estas máquinas logo foram também usadas no censo da Áustria em 1891, tendo sido elogiada na França. (COSTA, 2008, p. 71)

Chegando à era dos primeiros computadores eletrônicos, sabe-se que eles eram programados exclusivamente por meio de linguagens de baixo nível[3], que eram basicamente instruções baixadas diretamente nos circuitos eletrônicos. Já em torno da década de 1940, apareceram às linguagens de alto nível[4] que, fizeram da programação, uma tarefa mais fácil em relação a períodos anteriores.

Então, por volta da década de 1950, nasceu a primeira linguagem de programação de maior popularidade, o Fortran. Daí então, a programação ganhou escala e surgiram várias outras linguagens como Java, Java Script, C, C++, C#, Ruby, PHP, R, Objective-C e Python. Como já foi dito, iremos estudar posteriormente apenas duas dessas linguagens, no caso, Fortran e Python.

2.4 LÓGICA E LINGUAGEM DE PROGRAMAÇÃO

Quando queremos nos comunicar com as pessoas, precisamos entrar em contato com as mesmas e, falar um idioma que ambas entendam para que haja diálogo. Com um computador ocorre o mesmo, através dos programas, nos comunicamos com os componentes do computador a fim de efetuarmos a tarefa que desejamos.

Nesse sentido, entra a lógica de programação ou linguagem de programação, que trata do modo como se escreve um programa de computador, em outras palavras é um algoritmo. Filho afirma:

> Uma linguagem de programação torna-se assim, entre outras coisas, uma notação formal para a descrição de um algoritmo, entendendo-se por notação formal um simbolismo que não tenha as imprecisões nem a variabilidade de uma linguagem natural, que possibilite rigor nas definições e demonstrações sobre os procedimentos. (FILHO, 2017, p.110)

E, Cormenet al. (2002) descrevem algoritmo como:

---

[3]Linguagem de baixo nível trata-se de uma linguagem de programação que segue as características da arquitetura do computador. Assim, utiliza somente instruções que serão executadas pelo processador. Disponível em: https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o_de_baixo_n%C3%ADvel.

[4]Linguagem de alto nível é como se chama, na Ciência da Computação de linguagens de programação, uma linguagem com um nível de abstração relativamente elevado, longe do código de máquina e mais próximo à linguagem humana. Disponível em:https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o_de_alto_n%C3%ADvel

> Informalmente, um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saúda. Portanto, um algoritmo é uma sequência de passos computacionais que transformam a entrada na saída. (CORMEN, et al., 2002, p.3)

Logo, observamos que um algoritmo, nada mais é que uma sequência de passos para se realizar uma determinada função.

Uma forma bem popularmente conhecida de exemplificar um algoritmo, fora da Informática, seria uma receita de bolo. Para o bolo ficar pronto sem nenhum problema, é necessário seguir os passos corretamente. Na computação não é diferente, os programadores (pessoas que fazem ou editam os programas) escrevem as "receitas de bolo" (chamados de algoritmos na programação) de forma que o computador leia e entenda o que deve fazer.

Para isto, é indispensável uma linguagem de programação, que, nada mais é, que uma espécie de idioma em que escrevemos, para que o computador entenda o que queremos. Nesse caso, a linguagem de programação é como qualquer outro idioma, ou seja, um grupo de palavras com significados.

Também é importante ressaltar que a maioria das linguagens de programação é escrita em Inglês, devido essa ser a língua mais utilizada no mundo. Assim, podemos dizer que a lógica de programação é o que queremos fazer, organizados em passos e uma sequência lógica (algoritmo) e que a linguagem de programação é somente como se descreve formalmente esse algoritmo.

2.5 A MATEMÁTICA E A LÓGICA DE PROGRAMAÇÃO

Se pesquisarmos o que é um computador, encontraremos o seguinte significado de acordo com o Dicionário Aurélio: "Pessoa ou uma máquina que faz cálculos. Aparelho eletrônico usado para processar, guardar e tornar acessível informação de variados tipos." Veja que, os computadores têm essencialmente sua definição relacionada com a ação de efetuar cálculos. Então, não há como discordar de que a Informática está inteiramente ligada a Matemática.

Já que a Matemática está ligada a Informática, não é demais afirmar que a lógica Matemática está ligada a lógica de programação. Pois, a lógica Matemática está inteiramente

vinculada com a forma como são desenvolvidas as principais tarefas na computação, por meio dela podem ser criados *softwares* e circuitos que melhoram o desempenho das máquinas.

A Matemática está ligada desde a parte mais abstrata nos componentes de computadores, até a parte mais concreta dos componentes da máquina. Um exemplo bem simples é que ela é usada na resolução dos problemas e uso da lógica, até porque, não há como fazer operações Matemáticas, sem o uso da Matemática.

Todavia, vale salientar que não é apenas na parte do *software* que a Matemática se encontra, nas máquinas, tudo depende primordialmente dos conceitos matemáticos para elaboração da parte física do computador.

Nesse contexto Castells (1999, p. 51), fala que os "computadores, sistemas de comunicação, decodificação e programação genética são todos amplificadores e extensões da mente humana". Ou seja, podemos dizer que o computador é uma espécie de periférico da mente humana, que é utilizada para desempenhar funções que nosso corpo e mente é limitada e, assim, se tornando uma de nossas maiores conquistas.

2.6 O COMPUTADOR EM ÁREAS DE APLICAÇÃO MATEMÁTICA

Como já foi dito anteriormente, a Matemática teve um papel fundamental na evolução do homem e, essa foi aplicada constantemente em todos os períodos da história. No que se refere à área de aplicações da Matemática, há tempos os cientistas têm inventado modelos matemáticos para descrever fenômenos do mundo.

Sem dúvida, o surgimento do computador contribuiu amplamente para expandir a abrangência desse setor matemático. Segundo Ponte e Canavarro (1997), o computador tem tornado trabalhos longos e árduos, em processos muito mais simples de serem tratados devido à automatização das tomadas de decisões em relação aos dados, tornando seu uso mais dinâmico.

Os autores ainda dizem que na engenharia, por exemplo, tornou-se razoavelmente fácil resolver por métodos numéricos problemas de grande importância que até então eram impossíveis calcular à mão. Além de que, diversos processos físicos, químicos e biológicos podem ser modelados por meio de equações matemáticas. E, por meio do computador é possível testá-los sob diversas condições simulando a realidade.

Várias dessas aplicações são calculadas através de equações diferenciais ou sistemas de equações diferenciais. E, na maioria das vezes, só isso não é o suficiente para resolvê-las em termos exatos, tendo que apelar também para os métodos numéricos, aumentando assim a rapidez, eficácia e exatidão nos cálculos.

Na realidade, a capacidade de cálculos numéricos em relação aos modelos está intimamente limitada pelos recursos computacionais disponíveis, ou seja, os estudos só vão até onde as máquinas podem calcular. E para criar tais modelos, as principais ferramentas são os *softwares*, que podem ser criados desde o princípio do seu código ou simplesmente alterados, poupando assim, trabalhos análogos ou triviais.

Continuando o pensamento de Ponte e Canavarro (1997), vemos que a Matemática tem a cada dia se aliado mais à computação nas diversas necessidades humanas. Tal como na modalidade ambiental, no que se refere ao controle de populacional de animais, produção de madeira em relação ao plantio de árvores, qualidade da água que consumimos e até controle de poluentes no ar. No que diz respeito aos problemas globais, temos como exemplo o controle da produção a fim de evitar desperdício de recursos. Podemos citar ainda, a agricultura, na área de irrigação, domínio de pragas e na fertilização.

Outra área bastante relevante é produção de bens e serviços, tal como, quantidade de eletricidade, alimentos e produtos a serem distribuídos. Também nos transportes e sistemas de comunicações. Esses são apenas alguns exemplos em que a Matemática está interligada com a computação se fazendo presente a favor da modernização e globalização.

Nesse sentido, dizemos que o computador tem como função registrar os dados, manipulá-los e tomar decisões com base na lógica algorítmica em que foi programado, utilizando a Matemática como uma espécie de ferramenta para resolução de problemas. Desta forma, a Matemática se tornou um agente essencial nas Ciências da Computação, como afirma o relatório *Everybodycounts*:

> Tal como os computadores trazem novas oportunidades à Matemática, também é a Matemática que os torna incrivelmente eficazes.... As aplicações, o computador e a Matemática constituem um poderoso sistema fortemente unido produzindo resultados que anteriormente seriam impossíveis e originando ideias até aqui nunca imaginadas. (MSEB, 1989, p. 36)

Assim, vemos que a Matemática e a Computação, tem um papel fundamental na vida do ser humano e na evolução da Ciência.

3 CONTEXTO HISTÓRICO DAS LINGUAGENS UTILIZADAS

Foram escolhidas duas linguagens para serem comparadas nesse trabalho, o Fortran e o Python. Mas essa escolha não foi aleatória; a primeira diz respeito a uma das primeiras linguagens criadas e a outra, uma das mais atuais, e ambas, sendo linguagens bastante utilizadas no meio acadêmico e profissional. A seguir iremos trazer um contexto mais histórico do Fortran e Python.

3.1 O FORTRAN

Hoje em dia temos várias linguagens de programação, já citadas anteriormente e, devido à grande diversidade, existem vários guias, cursos e conteúdo de como aprendê-las, dando assim mais comodidade e facilidade a seu acesso.

Porém, nem sempre foi assim: algumas décadas atrás, programar era uma atividade extremamente difícil e cansativa. O programador deveria ter um conhecimento muito íntimo com a CPU do computador, já que era lá que o código era escrito. Basicamente, as instruções dos programas eram escritas em apenas dois símbolos numérico, 0 e 1: a essas instruções foi dado o nome de código binário.

Em seguida, surgiram algumas mnemônicas na programação, que são basicamente uma técnica de memorização, que usa sequências de letras ou palavras para organizar um aglomerado de dados, de uma forma mais fácil de lembrar. Um exemplo prático são as mnemônicas utilizadas para memorizar fórmulas físicas, veja a frase, "**V**ou **vo**ar mais **al**t**o**", usa-se para melhor memorização da fórmula V = V0 + at, que se trata da função horária do Movimento Retilíneo Uniformemente Variável. Assim, foi utilizando essa ideia de mnemônicas, que surgiu o designado *Assembly*[5].

Já na década de 50, Filho (2007) conta que "Em novembro de 1954 a equipe de Backus tinha criado o *IBM MathematicalFORmulaTRANslation System*, o FORTRAN" com a

[5]Assembly ou linguagem de montagem é uma notação legível por humanos para o código de máquina que uma arquitetura de computador específica usa, utilizada para programar códigos entendidos por dispositivos computacionais, como microprocessadores e microcontroladores. Disponível em: https://pt.wikipedia.org/wiki/Assembly.

finalidade de criar uma linguagem de programação de fácil compreensão e de eficiência análoga à linguagem Assembly. Assim, nasceu uma das primeiras linguagens de alto nível para o computador IBM 704, o FORTRAN (*FORmulaTRANslation*).

Com a criação do Fortran, os programadores passaram a se aplicar mais na resolução do problema. E, devido a sua facilidade de interpretação, em comparação com *Assembly*, o Fortran passou a ser uma linguagem bem mais aberta a quem quisesse conhecer o mínimo de programação, pois não necessitava ser um especialista para saber programar.

Filho (2007, p. 124, apud KNUTH, PARDO) afirma que "[...] os sistemas anteriores ofereciam duas escolhas: ou uma fácil codificação e uma execução lenta do programa ou uma laboriosa codificação com rápida execução, mas "o FORTRAN propiciava o melhor das duas opções"". Deste modo, o Fortran fornecia, naquele momento, a melhor escolha em programação já criada.

Henriques (1999) conta que a fama do Fortran foi tanta que, anos depois, empresas de Informática desenvolveram seus próprios compiladores de Fortran para seus computadores. Causando assim uma desordem, pois programas registrados em um computador, não serviriam para outro computador. Nesse contexto, surgiu a necessidade de padronizar a linguagem Fortran, ou seja, que os programas criados nessa linguagem pudessem ser usados em computadores distintos sem proceder com alterações em seu código.

Ele ainda afirma que, já em 1966, a *American National Standards Institute* (ANSI), no tempo, ainda chamada Associação Americana de Normalização, divulgou uma versão padronizada denotada por FORTRAN IV. Basicamente, era uma versão comum dos múltiplos dialetos do Fortran, porém padronizada.

Contudo, essa versão ainda não foi o suficiente para suprir a carência de regularidade nas diversas máquinas, muitas vezes devido ao trabalho para programar a versão padronizada na grande extensão de computadores, dentre outros motivos.

Então, Henriques (1999) declara que em 1978, lançaram uma nova versão denominada FORTRAN 77. Porém, a transição do FORTRAN IV para o FORTRAN 77 levou bastante tempo, somente nos meados da década de 80 o FORTRAN 77 passou a ter um uso majoritário.

O autor supracitado ainda diz que, três décadas após a criação do Fortran, tornaram as características do FORTRAN 77 ultrapassadas se comparadas com as novas linguagens que surgiram nesse período, como Pascal ou o C. Então, para o melhoramento da linguagem, a ANSI compôs um conselho técnico, o X3J3, denominado por ISO/IEC JTC1/SC22/WG5 (ou simplesmente WG5), com o objetivo de desenvolver e fazer a manutenção dos padrões de linguagem Fortran, criou primeiramente, o projeto Fortran 8x, que logo após foi chamado de Fortran 90.

Nesse contexto, o objetivo não era só padronizar as diferentes versões já existentes, como nos projetos anteriores, mas também atualizar a linguagem, devido à grande expansão de várias outras linguagens como Pascal, C, C++, APL, Algol, e Ada. Essa nova versão baseou-se em outras linguagens para incluir novas características ao Fortran, uma delas é de facilitar variáveis indexadas (*arrays*), com escrita mais clara e eficiente, além de versatilidade na manipulação de diversos tipos de dados definidos pelo usuário.

Uma inovação muito importante do FORTRAN 90 foi a simplificação para tratar problemas matemáticos, e quando aliada a computadores mais robustos, a linguagem se torna extremamente eficiente pela sua fácil adaptação ao *hardware*. Outra novidade foi a disponibilidade para os programadores escreverem em função do "tipo de dado" (ou tipo de variável), tornando assim, o programa perfeito para diversas atividades em áreas específicas de resolução de problemas. Deste modo, o Fortran voltou a ter destaque em várias áreas da comunidade científica e da engenharia.

Posteriormente à divulgação do Fortran 90, o WG5 começou a planejar atualizações para projetos futuros. A partir de revisões no Fortran 90, foi criado o Fortran 95 com apenas algumas correções e aprimoramentos, nada tão impactante.

Henriques (1999) diz que, mesmo assim, o Fortran 90 passou a ser sempre a base para outros desenvolvimentos, deu origem a uma extensão chamada *High Performance Fortran – HPF* (Fortran de Alta Performance) a qual é destinada para o uso de computadores com arquiteturas avançadas.

Logo após, tivemos o FORTRAN 2003, que implementou um conjunto de comandos, que dentre eles, o mais importante foi o domínio de exceções e programação orientada a objetos. Daí então, surgiram outras novas versões com alguns melhoramentos e a

que iremos utilizar nesse trabalho é a versão Fortran 2013. Assim, o Fortran foi e continua sendo um dos instrumentos mais importantes para aplicações científicas e numéricas.

3.2 O PYTHON

A Linguagem Python, é uma das mais conhecidas atualmente e foi projetada por Guido Van Rossum em 1989. Rossum, era um funcionário do *CentrumWiskunde & Informatica* (Centro de Matemática e Ciência da Computação - CWI) em Amsterdã, na Holanda, e segundo Rossum (1996), resolveu implementar o Python, no quadro de ampliação da Linguagem ABC.

Rossum (1996) conta, que o nome "Python" vem de um grupo humorístico do cinema britânico *Monty Python*, criador do programa *Monty Python Flying Circus*. E também faz referência a cobra "píton".

Segundo Leno (2014), já no início 1990 o Python já podia ser utilizado e, incrivelmente, ao final do mesmo ano, a linguagem já havia superado a ABC em termos de utilização no CWI. No ano seguinte, Guido foi remanejado da equipe Amoeba para a aquipe Multimídia. E em 20 de fevereiro de 1991, foi publicada a primeira versão do Python, a v0.9.0. (Também conhecida por Python 1.0), mas disponibilizada apenas em 1994.

Leno (2014) afirma que em 2000, publicou-se a Python 2.0 e, posteriormente, em 6 de março de 2001 foi criada a Python Software Foundation (PSF), uma organização sem fins lucrativos, que se dedica à manutenção e desenvolvimento da linguagem de programação Python, composta por membros do grupo de desenvolvimento daquele período e por Eric Raymond. Já em 2008, para modernizar a linguagem, já que nem tudo era compatível com a versão anterior, surge finalmente o Python 3.0 que é a versão mais atual do Python.

Atualmente, Leno (2014) ressalta que o Python está sob a licença *Python Software Foundation License*, que indica ser uma linguagem livre e de código aberto que permite modificações e estudos sobre o código. O Python também é multiplataforma, ou seja, funciona em qualquer sistema que possua o seu interpretador. Além de ser uma linguagem de alto nível, acessível e de desenvolvimento rápido. E, assim como o Fortran, também é uma linguagem com orientação a objetos.

O Python possibilita ao usuário criar programas, sejam jogos ou códigos científicos para cálculos avançados, isso para vários ambientes, desde desktop, web ou até mesmo móvel, além de que, sua sintaxe é uma das suas mais simples dentre todas as linguagens de programação. E tem uma ampla biblioteca de suporte para programar junto com uma comunidade enorme, que pode auxiliar ao usuário vários materiais em fóruns, redes sociais etc. Sem falar que atualmente, o Python possui grande mercado, contando com grandes empresas usuárias de seus serviços. Sem sombra de dúvida, é uma das principais linguagens de programação atualmente.

4 UM BREVE HISTÓRICO DE CADA PROBLEMA ESTUDADO

De modo mais específico, iremos comparar duas linguagens de programação, no caso, Fortran e Python, através dos problemas matemáticos:

- Geração de dados em um arquivo através de funções;
- Elementos da esfera em uma matriz;
- Obtenção da raiz de uma função através do Método da Bisseção;
- Aproximação do limite de uma série geométrica.

A seguir vamos falar um pouco mais sobre cada problema proposto.

4.1 PROBLEMA I - GERAÇÃO DE DADOS EM UM ARQUIVO ATRAVÉS DE FUNÇÕES

Um dos temas mais importantes e essenciais na Matemática é o assunto de funções. Dante (2013, p. 41) comenta que ao longo dos tempos, diversos matemáticos contribuíram para o conceito de função que temos atualmente, mas, foi o matemático alemão Leibniz que atribuiu a denominação função que utilizamos. Euler no século XVII, por sua vez, atribuiu a notação de função por $f(x)$, para denotar "f de x".

Posteriormente, o matemático alemão Dirichlet escreveu uma primeira definição de função análoga àquela que usamos hoje, Dante (2013, p. 41) cita: "Uma variável y se diz função de uma variável x se, para todo valor atribuído a x, corresponde, por alguma lei ou regra, um único valor de y. Nesse caso, x denomina-se variável independente e y, variável dependente.".

O autor supracitado, conta que já em torno do ano 2000 a.C., os babilônios, usavam a ideia de função fazendo, por exemplo, uma tabela de duas colunas, a primeira coluna com alguns números e a segunda com o produto desses respectivos números com um valor constante.

Por exemplo, dada uma função $y = x^2$. Escrevemos duas colunas denotadas por x e y, respectivamente. A coluna x se trata dos números quaisquer dados inicialmente, e a coluna y, trata do quadrado dos respectivos números. Observe o Quadro 1 a seguir:

Quadro 1: Valores de y em função de x

x	y
0	0
1	1
2	4
3	9
4	16
5	25
⋮	⋮
n	x^2

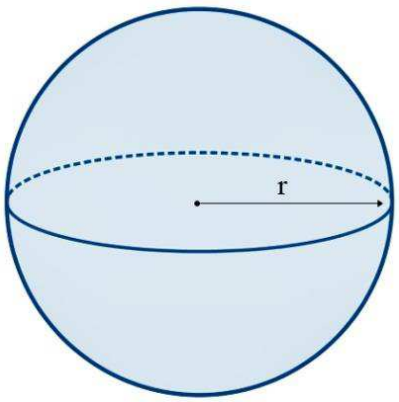
Fonte: Da pesquisa (2019)

No Quadro 1, calculamos o valor de y num intervalo de zero até n, assim, o Problema I consiste em elaborar um algoritmo que calcule o valor de y num certo intervalo, para valores de x inteiro e escrever os pontos x e y destes resultados num arquivo do formato .txt no computador.

4.2 PROBLEMA II - ELEMENTOS DA ESFERA EM UMA MATRIZ

De acordo com Dolce e Pompeo (2005, p. 250), considerando um ponto O e um segmento de reta r, denomina-se esfera de centro O e raio r, o conjunto de pontos P no espaço ($\mathbb{R}^3$), tais que a distância $\overline{OP}$ seja menor ou igual a r. Os autores ainda afirmam ainda que "A esfera é um sólido de revolução gerado pela rotação de um semicírculo em torno de um eixo que contém um diâmetro.". Observe a representação geométrica de uma esfera na Figura 1:

Figura 1: Representação geométrica de uma esfera



Fonte: Adaptado de Silva (2019)

Dolce e Pompeu (2005) ainda citam que, nesse caso, o diâmetro D é dado por:

$$D = 2r$$

A área A da superfície de uma esfera de raio r é igual a:

$$A = 4\pi r^2$$

E o volume V da esfera de raio r é dado por:

$$V = \frac{4}{3}\pi r^3$$

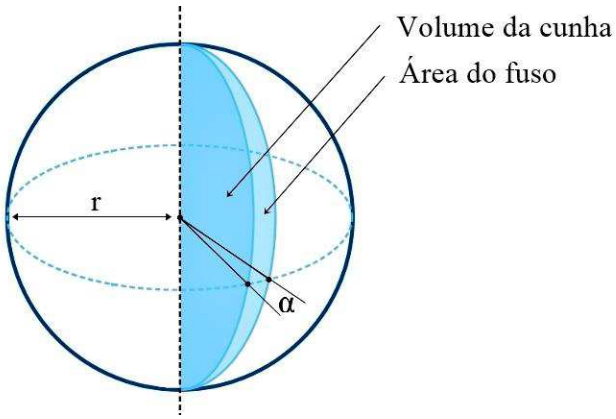
A área do fuso A_f, representado na Figura 2, sendo α dado em graus, dá-se por:

$$A_{fuso} = \frac{\alpha}{360} \cdot 4\pi r^2$$

E o volume da cunha V_c, também representado na Figura 2, é:

$$V_{cunha} = \frac{\alpha}{360} \cdot \frac{4}{3}\pi r^3$$

Figura 2: Representação geométrica da cunha e fuso esférico



Fonte: Adaptado de Silva (2019)

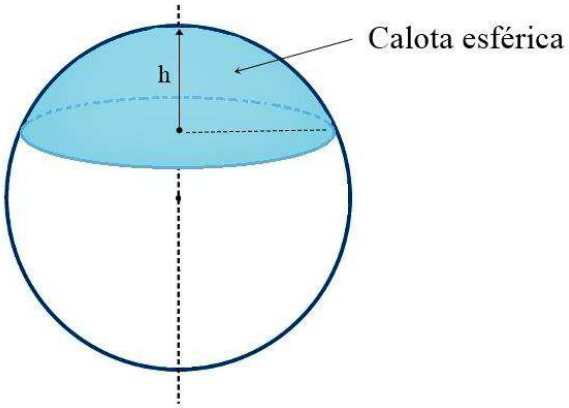
Outro elemento bastante importante da esfera é a calota, representado na Figura 3. Quanto à calota, podemos destacar que o volume da calota esférica é dada por:

$$V_{calota} = \frac{1}{3}\pi h^2(3r - h)$$

E, por fim, a área superficial da calota é:

$$A_{calota} = 2\pi r h$$

Figura 3: Representação geométrica da calota esférica



Fonte: Adaptado de Silva (2019)

Depois de calculados todos os Elementos da esfera supracitados, iremos colocá-los numa matriz, que, de acordo com Iezzi e Hazzan (1977, p.35) definem matriz como sendo: "Dados dois números m e n naturais e não nulos, chama-se matriz m por n $(m \times n)$, toda tabela M formada por números reais distribuídos em m linhas e n colunas.".

Os autores acima ainda dizem que em uma matriz qualquer M, cada elemento é indicado por a_{ij}. O índice i indica a linha e o j indica a coluna às quais o elemento pertence, onde as linhas são numeradas de cima para baixo (de 1 até m) e as colunas, da esquerda para direita (de 1 até n), uma matriz M porém é representada por:

$$M = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \text{ ou,}$$

$$M = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \text{ ou,}$$

$$M = \left\| \begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{matrix} \right\|$$

Além disso, uma matriz $(m \times n)$ também pode ser indicada por $M = (a_{ij})$, ou simplesmente, $M = (a_{ij})_{(m \times n)}$.

Sendo assim, o Problema II consiste em calcular o diâmetro, área, volume, área do fuso, volume da cunha, volume da calota esférica e área da calota da esfera, de raio r, h e α dados, e representar esses sete dados numa matriz de dimensão $(m \times 7)$:

$$M = \begin{bmatrix} D & A & V & A_f & V_c & V_{ca} & A_{ca} \\ D & A & V & A_f & V_c & V_{ca} & A_{ca} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & a_{m4} & a_{m5} & a_{m6} & a_{m7} \end{bmatrix}$$

com número de linhas suficientemente grandes de forma que consiga fazer a máquina trabalhar a fim de atingir um tempo considerável para melhor comparação dos resultados.

4.3 PROBLEMA III - OBTENÇÃO DA RAIZ DE UMA FUNÇÃO ATRAVÉS DO MÉTODO DA BISSEÇÃO

Para Justos e Sauter, et. al., (2018), o Método da Bisseção explora o fato de que se uma função contínua $f: [a, b] \to \mathbb{R}$ com $f(a) \cdot f(b) < 0$ tem uma raiz (ou zero) no intervalo (a, b), a ideia para se aproximar da raiz de tal função $f(x)$ é tomar como primeira aproximação, o ponto médio do intervalo $[a, b]$, ou seja:

$$x^{(0)} = \frac{(a+b)}{2}$$

Os autores ainda complementam que se acontecer de $f\left(x^{(0)}\right) = 0$, então a raiz de $f(x)$ é $x^* = x^{(0)}$. De outro modo, se $f(a) \cdot f\left(x^{(0)}\right) < 0$, então $x^* \in \left(a, x^{(0)}\right)$. Se isto ocorrer, tomamos como segunda aproximação da raiz de $f(x)$ o ponto médio do intervalo $[a, x^{(0)}]$, ou seja,

$$x^{(1)} = \frac{\left(a + x^{(0)}\right)}{2}$$

No outro caso, temos $f\left(x^{(0)}\right) \cdot f(b) < 0$, daí pegamos,

$$x^{(1)} = \frac{\left(x^{(0)} + b\right)}{2}$$

Deste modo repetimos este artifício até obtermos a aproximação que queremos.

Os autores supracitados ainda trazem a seguinte situação: suponha uma função contínua $f: [a, b] \to \mathbb{R}$ com $f(a) \cdot f(b) < 0$, tal que desejemos calcular uma aproximação com uma certa precisão (tolerância) para uma raiz x^*. Começamos, tomando $n = 0$ e: $a^{(n)} = a$, $b^{(n)} = b$ e $x^{(n)} = \frac{(a^{(n)} + b^{(n)})}{2}$.

Observamos o critério de parada, ou seja, se $f(x^{(n)}) = 0$ ou:

$$\frac{(a^{(n)} + b^{(n)})}{2} < \text{tolerância}$$

Se isso ocorrer, $x^{(n)}$ é a aproximação desejada. De forma contrária, seguimos para a próxima iteração $n + 1$:

- Se $f(a^{(n)}) \cdot f(x^{(n)}) < 0$, então definimos $a^{(n+1)} = a^{(n)}$ e $b^{(n+1)} = x^{(n)}$.
- Se $f(x^{(n)}) \cdot f(b^{(n)}) < 0$, então definimos $a^{(n+1)} = x^{(n)}$ e $b^{(n+1)} = b^{(n)}$.

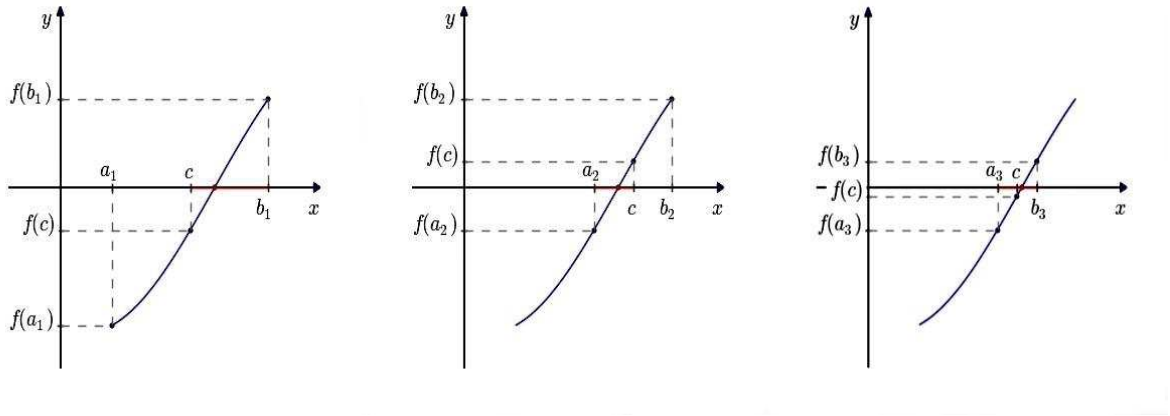
Substituindo n por $n + 1$, teremos a nova aproximação da raiz de $f(x)$:

$$x^{(n+1)} = \frac{(a^{(n+1)} + b^{(n+1)})}{2}.$$

Deste modo, verificamos o critério de parada acima e, caso não atendido, iteramos novamente até obter a aproximação desejada ou até atingir o número máximo de iterações.

De forma mais resumida, é um método que busca raízes aproximadas, através do corte bi seccional (Por este motivo, também chamado de Método da Bissecção) repetitivo de um intervalo, então seleciona um subintervalo contendo a raiz para processamento adicional, seguindo um critério de convergência para certa precisão (ou tolerância – quantidade de casas decimais exatas exigidas na raiz). Podemos observar esse comportamento, geometricamente, na Figura 4:

Figura 4: Interpretação geométrica do Método da Bisseção



Fonte: Adaptada de Wikipédia (2015)

Nesse sentido, iremos usar o algoritmo do Método da Bisseção para calcular a raiz de uma função, e repetir esse código várias vezes, fazendo a máquina trabalhar até obter um valor numérico de tempo significativo para comparações.

4.4 PROBLEMA IV - APROXIMAÇÃO DO LIMITE DE UMA SÉRIE GEOMÉTRICA

Em Matemática, Thomas (2012, p.13) diz que "Uma série infinita é a soma de uma sequência infinita de números.", do tipo:

$$a_1 + a_2 + a_3 + \cdots + a_n + \cdots$$

Se as somas parciais da série convergirem para um limite L, dizemos que a série converge e que sua soma resulta em L. Assim, escrevemos:

$$a_1 + a_2 + a_3 + \cdots + a_n + \cdots = \sum_{n=1}^{\infty} a_n = L$$

Além disso, existem as séries geométricas, que são expressões da forma:

$$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} + \cdots = \sum_{n=1}^{\infty} ar^{n-1}$$

A série também pode ser escrita como:

$$\sum_{n=0}^{\infty} ar^n,$$

onde a e r são números reais fixos, tal que $a \neq 0$ e r é a razão entre os termos.

Se $|\, r \,| < 1$, a série geométrica $a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} + \cdots$ converge para:

$$\sum_{n=0}^{\infty} ar^n = \frac{a}{1-r}$$

Caso contrário, se $|\, r \,| \geq 1$, a série diverge.

Em particular, tomando como exemplo uma série geométrica infinita com $a = r = \frac{1}{2}$, temos:

$$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} + \cdots =$$

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

Observe ainda que como $|\, r \,| < 1$, podemos fazer:

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots = \sum_{n=0}^{\infty} ar^n = \frac{a}{1-r} = \frac{\frac{1}{2}}{1-\frac{1}{2}} = 1$$

Portanto, nesse caso, $L = 1$.

Nesse sentido, vamos utilizar essa série geométrica para calcular a precisão que cada compilador tem em seus cálculos, de modo geral, quanto mais próximo de 1 for o resultado, mais precisão o compilador terá em seus cálculos.

5 METODOLOGIA

Este capítulo tem como função descrever os materiais e métodos utilizados nessa pesquisa, a fim de tornar mais claro como foi feito tal estudo.

5.1 COMPARAÇÃO DOS PROBLEMAS MATEMÁTICOS

Vamos comparar as linguagens Fortran e Python através de problemas matemáticos e, no caso, comparar suas velocidades de processamento na **Geração de dados em um arquivo através de funções**, onde iremos escrever os pontos x e y de uma determinada função destes resultados em um arquivo do formato .txt no computador.

Iremos também comparar suas velocidades de processamento em relação a alguns **Elementos de uma esfera**, tal como calcular o diâmetro, área, volume, área do fuso, volume da cunha, calota esférica e área da calota da esfera, de raio r, h e α dados (armazenando os dados em uma matriz).

Além disso, iremos comparar as linguagens pela velocidade em **Obtenção da raiz de uma função através do Método da Bisseção**, que nada mais é que calcular a raiz de uma função por esse método numérico.

Por fim, vamos comparar a **Aproximação do limite de uma série geométrica**, utilizando uma série geométrica dada, para calcular a precisão que cada compilador/interpretador da linguagem tem em seus cálculos, de modo geral, quanto mais próximo do limite da série geométrica for o resultado, mais precisão o compilador terá em seus cálculos.

5.2 COMPARAÇÃO DOS PROGRAMAS

Como o objetivo principal deste trabalho é comparar as linguagens Fortran e Python, para tal, vamos utilizar *scripts*, que de forma direta é a forma como o programa do problema é escrito no computador, ou seja, a sequência de passos escrita que a máquina vai interpretar para calcular os problemas propostos.

Nesse sentido, procuramos utilizar *scripts* com mesma sequência lógica para ambas linguagens, priorizando desta forma, a velocidade de execução de cada uma, para que não haja discrepância ou vantagem para qualquer que seja.

Desta forma, enfatizamos a necessidade de comparar tais linguagens, visto que, em determinadas demandas, o tempo de cálculo gasto é de extrema importância para obtenção de resultados melhores e mais rápidos.

5.3 ORGANIZAÇÃO DOS DADOS DA PESQUISA

Para melhor observação dos resultados, os dados da pesquisa foram organizados em tabelas. Nos três primeiros problemas, as diferenças percentuais de desempenho da velocidade foram calculadas da seguinte forma:

$$Dpd = \frac{Tm_{maior} \cdot 100}{Tm_{menor}} - 100$$

Onde:

- Dpd é a diferença percentual de desempenho da velocidade;
- Tm_{maior} é o maior tempo médio dentre o Fortran e Python;
- Tm_{menor} é o menor tempo médio dentre o Fortran e Python.

E o desempenho médio Dm será calculado da seguinte maneira:

$$Dm = \frac{\sum Dpd}{5}$$

Já no quarto problema, a diferença percentual de desempenho da precisão é dada:

$$Dpdp = \frac{nmax_{maior} \cdot 100}{nmax_{menor}} - 100$$

Onde:

- $Dpdp$ é a diferença percentual de desempenho da precisão;
- $nmax_{maior}$ é o maior termo máximo, antes de atingir o limite 1, dentre o Fortran e Python;
- $nmax_{menor}$ é o menor termo máximo antes de atingir o limite 1, dentre o Fortran e Python.

5.4 DADOS DO COMPUTADOR UTILIZADO

Os experimentos para este presente trabalho foram realizados num notebook da marca Dell, modelo Inspiron 15-3567, com sistema operacional Microsoft Windows 10 Home Single Language 64 bits. O processor utilizado foi o Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz com clock speed de 2.7Ghz e com base x64. O computador tem como configurações uma memória RAM de 4GB (utilizável: 3,87GB) e disco rígido de 1TB.

5.5 DADOS DO COMPILADOR E INTERPRETADOR UTILIZADOS

Os compiladores e interpretadores[6] são basicamente programas que converte as instruções de uma linguagem qualquer de programação (no caso, Fortran e Python) para linguagem de máquina, possível de ser entendida e processada pelo computador. Ou seja, é o programa utilizado para escrever as linguagens e criar outros programas. O compilador é um programa traduz o código da linguagem em questão, para linguagem de máquina e então gera um arquivo que possa ser executado pelo computador. Já o interpretador traduz o código do programa linha a linha, e como consequência, o programa vai sendo utilizado na medida em que vai sendo traduzido. A seguir veremos os compilador do Fortran e o Interpretador do Python utilizados.

5.5.1 Compilador Fortran

O compilador do Fortran utilizado foi o Intel® Visual Fortran 13.0, que é um componente do produto Intel® Visual Fortran Composer XE 2013 para Windows. Além disso, para melhor usabilidade foi instalado o "Microsoft Visual Studio 2012 (ou simplesmente VS2012)". O VS2012 é um ambiente de desenvolvimento integrado (IDE) da Microsoft para desenvolvimento de software voltado a diversas linguagens, inclusive ao Intel® Visual Fortran 13.0.

5.5.2 Interpretador Python

A versão utilizada foi a Python 3.7.3. com data de lançamento em 25 de março de 2019, trata-se da versão principal mais nova do Python atualmente. Além disso, para melhor

[6] Mais informações em:
https://www.oficinadanet.com.br/artigo/1527/diferencas_entre_compiladores_e_interpretadores

performance nas realizações dos testes em Python, foram necessárias instalações de pacotes extras em sua pasta, como Numpy, SciPy, Matplotlib, IPython, Jupyter, pandas, SymPy e Nose. Tudo isso possível apenas ao digitar no prompt de comando, na pasta onde está instalado o Python 37: “python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose”.

6 RESULTADOS E DISCUSSÕES

Nessa seção estão descritos os resultados da pesquisa, ou seja, o estudo de cada um dos problemas descritos no Capítulo IV, além de comentários acerca de cada caso e explanações gerais sobre os resultados.

6.1 ESTUDO DO PROBLEMA I - GERAÇÃO DE DADOS EM UM ARQUIVO ATRAVÉS DE FUNÇÕES

Esse estudo foi feito com o script dos "APÊNDICE A" e "APÊNDICE B". E todos os dados estão representados na Tabela 1, onde podemos conferir o desempenho dos compiladores Fortran e Python na parte de Geração de dados em um arquivo através de funções, nesse caso, a função $y = x^2$, a quantidade de cálculos dos valores de x_1 até x_2 e seu respectivo tempo médio de processamento, além do aumento de desempenho do processador mais rápido em relação ao mais lento.

Tabela 1: Tempo de processamento e diferença percentual de desempenho na Geração de dados em um arquivo através de funções em Fortran e Python

NÚMERO DE CÁLCULOS	TEMPO MÉDIO (S) - FORTRAN	TEMPO MÉDIO (S) - PYTHON	DESEMPENHO (%)
$\mathbf{2 \cdot 10^6}$	16,82412	5,11071	229,19
$\mathbf{4 \cdot 10^6}$	25,65166	10,10145	153,58
$\mathbf{6 \cdot 10^6}$	37,83062	15,11661	150,26
$\mathbf{8 \cdot 10^6}$	50,12654	19,71671	154,23
$\mathbf{10 \cdot 10^6}$	62,22131	25,42515	144,72
DESEMPENHO MÉDIO (%)	**166,39%**		

Fonte: Dados da pesquisa (2019).

Deste modo percebemos que o **Python é mais rápido em relação à soma do tempo de cálculo e escrita em arquivo** para a situação estudada, significando um desenvolvimento de **166,39%** superior em sua velocidade de processamento.

6.2 ESTUDO DO PROBLEMA II - ELEMENTOS DA ESFERA EM UMA MATRIZ

Esse estudo foi feito com os *scripts* dos "APÊNDICE C" e "APÊNDICE D". Para tal, na Tabela 2 podemos observar o desempenho do Fortran e Python na parte de Elementos da esfera em matriz, nesse caso, os dados em unidades são: $r = 1000$, $\alpha = 10$ e $h = 100$ e a matriz formada por m linhas e 7 colunas como informa a Tabela 2. Portanto, a Tabela 2 mostra a quantidade de linhas da matriz e seu respectivo tempo médio de processamento, tal como a diferença de desempenho entre as linguagens.

Tabela 2: Tempo de processamento e diferença percentual desempenho nos Elementos da esfera em matriz em Fortran e Python

NÚMERO DE CÁLCULOS	TEMPO MÉDIO (S) - FORTRAN	TEMPO MÉDIO (S) - PYTHON	DESEMPENHO (%)
$\mathbf{2 \cdot 10^6}$	0,03375	0,04710	39,55
$\mathbf{4 \cdot 10^6}$	0,09238	0,10182	10,22
$\mathbf{6 \cdot 10^6}$	0,12878	0,15562	20,84
$\mathbf{8 \cdot 10^6}$	0,15884	0,21287	34,01
$\mathbf{1 \cdot 10^7}$	0,21381	0,26856	25,61
DESEMPENHO MÉDIO (%)	**26,05%**		

Fonte: Dados da pesquisa (2019).

Portanto, percebemos que o **Fortran é mais rápido em relação à soma do tempo de cálculo e escrita em matriz** para a situação acima, significando um desenvolvimento cerca de **26,05%** superior em sua velocidade de processamento.

6.3 ESTUDO DO PROBLEMA III - OBTENÇÃO DA RAIZ DE UMA FUNÇÃO ATRAVÉS DO MÉTODO DA BISSEÇÃO

Esse estudo foi feito com o script dos "APÊNDICE E" e "APÊNDICE F" que são baseados nos *scripts* dos "ANEXOS A" de Santana (2012) e "ANEXOS B" de Justos (2018), respectivamente. Os dados estão representados na Tabela 3, onde podemos observar o desempenho do Fortran e Python na parte de Obtenção da raiz de uma função através do Método da Bisseção, nesse caso, a função $f(x) = x^4 - 9x^3 - 2x^2 + 120x - 130$, para

valores de intervalo $a = 0$ e $b = 2$, com número máximo de interações em 10^4 e tolerância de $1e^{-14}$. Portanto, a Tabela 3 mostra a quantidade de cálculos (vezes que encontrou uma raiz) e seu respectivo tempo médio de processamento.

Tabela 3: Tempo de processamento e diferença percentual desempenho na Obtenção da raiz de uma função através do Método da Bisseção em Fortran e Python

NÚMERO DE CÁLCULOS	TEMPO MÉDIO (S) - FORTRAN	TEMPO MÉDIO (S) - PYTHON	DESEMPENHO (%)
$\mathbf{10^6}$	0,26212	62,06671	23578,73875
$\mathbf{2 \cdot 10^6}$	0,52237	124,17883	23672,19787
$\mathbf{3 \cdot 10^6}$	0,79916	184,81715	23026,42650
$\mathbf{4 \cdot 10^6}$	1,03194	247,27645	23862,28947
$\mathbf{5 \cdot 10^6}$	1,30744	309,39339	23564,06030
DESEMPENHO MÉDIO (%)	**23540,74%**		

Fonte: Dados da pesquisa (2019).

Portanto, percebemos que o **Fortran é extremamente mais rápido em relação a Obtenção da raiz de uma função através do Método da Bisseção para a situação acima**, significando um desenvolvimento de **23540,74%** superior em sua velocidade de processamento.

6.4 ESTUDO DO PROBLEMA IV - APROXIMAÇÃO DO LIMITE DE UMA SÉRIE GEOMÉTRICA

Esse estudo foi feito com o script dos “APÊNDICE G” e “APÊNDICE H” onde na Tabela 4 podemos observar o desempenho de precisão dos cálculos em Fortran e Python na parte de Aproximação do limite de uma série geométrica, através dos valores dos somatórios para diversos números de termos (n).

Tabela 4: Diferença percentual de desempenho da precisão nos valores do somatório de uma série geométrica em relação ao número de termos (n) em Fortran e Python

N	SOMATÓRIO - FORTRAN	SOMATÓRIO – PYTHON
1	0,5	0,5
2	0,75	0,75
3	0,875	0,875
⋮	⋮	⋮
10	0,9990234375	0,9990234375
⋮	⋮	⋮
15	0,999969482421875	0,999969482421875
16	0,999984741210938	0,9999847412109375
⋮	⋮	⋮
20	0,999999046325684	0,99999904632568359375
⋮	⋮	⋮
24	0,999999940395355	0,999999940395355224609375
25	1	0,999999970197677612304687 5
⋮	⋮	⋮
53	1	0,99999999999999988897769753748434595763683319091796875
54	1	1
DIFERENÇA DE DESEMPENHO (%)		**120,83%**

Fonte: Dados da pesquisa (2019).

Na Tabela acima podemos observar que para os 15 primeiros termos, os valores de somatórios são iguais, só então a partir do 16º termo as casa decimais diferem, por motivo de arredondamento no caso do Fortran, o que irá ocasionar uma perda de precisão futuramente. É o que acontece do 24º termo, ele chega a seu limite máximo de casas decimais e no 25º ele já nos dá o valor 1 de seu somatório. Assim, o $nmax_{menor} = 24$.

No caso do Python, vemos que ele tem o dobro de casa decimais, o que tornam seus arredondamentos mais sutis, e seu limite se dá só então no 53º termo, e logo em seguida no 54º termo ele chega ao limite 1. Logo, $nmax_{maior} = 53$

Logo, nesse exemplo, o Python pode calcular 29 termos a mais que o Fortran, o que nos dá uma diferença de desempenho do Python de 120,83% a mais que o Fortran na precisão em relação à diferença de termos que cada um pode calcular. Portanto, o **Python tem uma precisão maior que o Fortran de 120,83% para essas condições**.

6.5 COMENTÁRIOS GERAIS SOBRE OS RESULTADOS

De modo geral, chegamos aos seguintes resultados:

- Problema I - O Python é 166,39% mais rápido em relação à soma do tempo de cálculo e escrita em arquivo;
- Problema II - O Fortran é 26,05% é mais rápido em relação à soma do tempo de cálculo dos elementos da esfera e escrita em matriz;
- Problema III - O Fortran é 23540,74% mais rápido em relação Obtenção da raiz de uma função através do Método da Bisseção;
- Problema IV - O Python tem uma precisão de 120,83% maior que o Fortran.

Podemos perceber que quando se trata de cálculos exaustivos no caso no Método da Bisseção, o Fortran é mais rápido, com isso, devido a seus cálculos mais velozes, também ganha muito tempo ao escrever dados numa matriz.

Já o Python tem mais facilidade em processos mais “artesanais”, um exemplo é a criação, escrita de dados em arquivo e sua precisão devido ao imenso número de casas decimais. Em relação à usabilidade das linguagens, todos os *scripts* em Python tiveram menos linhas de comando, isso indica sua facilidade na escrita e manutenção dos códigos em relação ao Fortran.

Com isso, percebemos que cada linguagem tem o seus pontos positivos e negativos, cabe ao utilizador escolher qual melhor para trabalhar de acordo com suas necessidades e qual problema deseja resolver.

7 CONSIDERAÇÕES FINAIS

O Fortran e o Python, atualmente, são linguagens parcialmente simples de aprender, entretanto, percebemos ao longo deste trabalho, que é bem mais fácil encontrar bons materiais de estudo em Python do que bons materiais em Fortran, isso, devido ao fato do Python ser mais utilizado popularmente e por ser uma linguagem de código aberta, a qual é lançada regularmente atualizações, sendo bastante aplicada em exemplos distintos na área acadêmica. Já o Fortran é atualmente, tido como uma linguagem utilizada mais particularmente para computação numérica.

No que se refere à instalação dos compiladores, instalar o Python em uma máquina Windows é muito mais simples e seus arquivos são muito mais leves do que o compilador Fortran. Em linhas gerais, como o Python é uma linguagem interpretada o que você sacrifica em desempenho, você ganha em produtividade, ou seja, pode resolver um problema simples em menos tempo, pois podemos escrever em menos linhas de código que o Fortran, já que o Fortran é uma linguagem compilada.

Foi observado também, que o Python é uma linguagem bastante aprimorada, com uma sintaxe intuitiva e cheia de recursos interessantes sustentando-se em uma manutenção bastante simples. Já o Fortran tem uma sintaxe mais complexas, e suas bibliotecas de uso acadêmico são bastante robustas, exigindo assim, ser uma linguagem compilada o que compensa em sua grande performance de processamento.

E, em termos de desempenho, notamos que, no geral, ambas têm resultados bastante semelhantes. Deste modo, convém usar o Fortran onde realmente exige-se bastante processamento e cálculos exaustivos, pois vimos que esse é o seu ponto positivo, já que onde o Python ganha em sintaxe, perde em performance. Todavia, em problemas mais simples, é viável utilizar o Python, uma vez que é muito mais conveniente e fácil escrever e modificar o Python do que o Fortran, pois, de fato, é uma linguagem de mais alto nível que o Fortran.

Neste sentido, observamos que o Python é indicado para tarefas mais gerais, onde se almeja alta produtividade, boa precisão e o desempenho é uma condição secundária e o Fortran para tarefas de computação numérica e científica que demandam mais desempenho. Portanto, concluímos que não é uma questão de qual a melhor linguagem, mas sim, qual o melhor instrumento para cada problema.

REFERÊNCIAS

ANDRADE, Júnior. **A matemática aplicada na Informática**. Disponível em: <https://prezi.com/drdjkxaljf3f/a-matematica-aplicada-na-informatica/>. Acesso em: 09 de jan. de 2019.

AURÉLIO. **Dicionário do Aurélio Online** 2018. Disponível em: <https://dicionariodoaurelio.com/computador>. Acesso em: 24 de mar 2019.

CASTELLS, Manuel. **A sociedade em rede**. São Paulo: Paz e Terra, 1999.

CLEMENTE, Beatriz. Slide Player. **Lógica de Programação – Aula 1**. Disponível em: <https://slideplayer.com.br/slide/1800795/?_gl=1*18qlwjy*_ga*TTMzZ1JNdWxBM0psbWhBbF9SM1NCdmlORUxBT2NISHp1M2hjekN1RUpXNG5qQklIYkhaNlVRSHNiRlpYa25Meg>. Acesso em: 07 de jan. de 2019.

CORMEN, T. H. et al. **Algoritmos**. Tradução da 2ª Edição Americana. Teoria e prática. Editora Campus. 2002.

COSTA, Eli Banks Libertado da. **O invento de Jacquard e os computadores: alguns aspectos das origens da programação no século XIX.** São Paulo. 2008. Disponível em: <https://tede2.pucsp.br/bitstream/handle/13377/1/Eli%20Banks%20Liberato%20da%20Costa.pdf>. Acesso em: 13 de abr. de 2019.

DANTAS, Tiago. **Hardware e Software**. Disponível em: <https://mundoeducacao.bol.uol.com.br/informatica/hardware-software.htm>. Acesso em: 24 de jan. de 2019.

DANTE, Luiz Roberto. **Contextos & Aplicações.** Volume 1. 2ª edição. Editora Ática. São Paulo-2013.

DICIO. Dicionário Online de Português. **Significado de Compilador**. Disponível em: <https://www.dicio.com.br/compilador/>. Acesso em: 14 de jun. de 2019.

DOLCE, Osvaldo; POMPEO, José Nicolau. **Fundamentos de Matemática Elementar 10**. Geometria Espacial Posição e Métrica. Atual Editora. 5ª edição. 6ª reimpressão. 2005.

FERNANDES, Natal L. R. **Professores e computadores: Navegar é preciso**. Porto Alegre: Mediação, 2004.

FEYNMAN, Richard P. **O que é uma lei física?.**col. Ciência Aberta, editora Gradiva, 1989.

FILHO, A. C. G. et al.**Importância do Hardware e Software em Organizações Ligadas ao Governo Eletrônico**. 2008.Disponível em:

<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwj_wpG3rb7hAhU_DrkGHWzODPIQFjAAegQIABAC&url=https%3A%2F%2Frevistas.unicentro.br%2Findex.php%2Fcapitalcientifico%2Farticle%2Fdownload%2F807%2F923&usg=AOvVaw0bUHRSYsdyXiE0O2zBE51d>. Acesso em 07 de abr. de 2019.

HELERBROCK, Rafael. **Informática**. Disponível em: <https://brasilescola.uol.com.br/informatica/>. Acesso em: 04 de jan. de 2019.

HENRIQUES, António Abel Ribeiro. **Programação e Computadores**. Licenciatura em Engenharia Civil. Faculdade de Engenharia Universidade do Porto – FEUP. 1999.

IEZZI, Gelson; HAZZAN, Samuel. **Fundamentos de Matemática Elementar.** 2ª edição. Atual Editora Ltda. São Paulo – 1977.

INTEL, Visual Fortran Compiler 13.0. **Fortran 13.0**. Disponível em: <https://www.copsmodels.com/gpifort13inst.htm>. Acesso em: 14 de jun. de 2019.

JULIANI, Kleber Sebastião. **Geometria espacial uma visão do espaço para a vida**. Londrina, 2008. Disponível em: < http://www.uel.br/projetos/matessencial/superior/pde/kleber-geometria-espacial.pdf>. Acesso em 13 de abr. de 2019.

JUSTO, Dagoberto A. R... [et al.]. **Cálculo Numérico**. Um Livro Colaborativo. Versão Python. 2018.

KEEN, P. G.W. **Guia gerencial para a tecnologia da informação: conceitos essenciais e terminologia para empresas e gerentes**. Rio de Janeiro: Campus, 1996.

KNUTH, D. E. & Pardo, L. T. **The early development of programming languages, in A history of computing in the twentieth century** (a collection of essays). London: Academic Press, 1980.

LENO, Magnun. **A História do Python.** 2014. Diponível em:<http://mindbending.org/pt/a-historia-do-python>. Acesso em 13 de abr. de 2019.

LOPES, Frederico. **A origem da palavra matemática**. 9 de dezembro de 2016. Disponível em:<https://fredlopes.noblogs.org/a-origem-da-palavra-matematica/>. Acesso em 07 de abr. de 2019.

LUNA, Alexandre José Henrique de Oliveira. **Um modelo para governança ágil em Tecnologia da Informação e Comunicação**. Recife, 2009. Disponível em :<https://repositorio.ufpe.br/bitstream/123456789/2272/1/arquivo2413_1.pdf>. Acesso em 07 de abr. de 2019.

MARAN, R. **Aprenda a usar o computador e a internet através de imagens**. Rio de Janeiro: Reader'sDigest Brasil, 1999.

MSEB (1989).**Everybodycounts: A reporttothenationonthe future ofmathematicseducation**. Washington, DC: NCR.

NORTON, P. **Introdução à Informática**. São Paulo: Makron Books, 1997.

OLIVEIRA, Naysa. Graduada em matemática. **O que é matemática?** Disponível em: <https://brasilescola.uol.com.br/o-que-e/matematica/>. Acesso em: 04 de jan de 2019.

PACIEVITCH, Yuri. Infoescola. **Lógica de Programação**. Disponível em: <https://www.infoescola.com/informatica/logica-de-programacao/>. Acesso em: 07 de jan de 2019.

PONTE, J. P.; CANAVARRO, P. (1997). **Matemática e novas tecnologias**. Lisboa: Universidade Aberta.

PYTHON, Software Foundation. **Python 3.7.3**. Disponível em: <https://www.python.org/downloads/release/python -373/>. Acesso em: 13 de jun. de 2019.

REZENDE, D. A.; ABREU, A. F. de. **Tecnologia da informação aplicada a sistemas de informação empresariais: o papel estratégico da informação e dos sistemas de informação nas empresas**. São Paulo: Atlas, 2000.

ROSSUM, Guido Van. **Foreword for "Programming Python" (1st ed.).** Reston, VA, May 1996.

SANTANA, Carlos. **Método da Bissecção**. 21 de abril de 2012. Disponível em: <https://dr-acss.blogspot.com/2012/04/metodo-da-bisseccao.html?m=> Acesso em: 13 de jun. de 2019.

SILVA, Luiz Paulo Moreira. **Elementos de uma esfera**. *Brasil Escola*. Disponível em: https://brasilescola.uol.com.br/matematica/elementos-uma-esfera.htm. Acesso em 13 de jun. de 2019.

TERRA. **Programação em python: 8 razões para você aprender a linguagem**. Disponível em: <http://idgnow.com.br/carreira/2018/09/14/programacao-em-python-8-razoes-para-voce-aprender-a-linguagem/>. Acesso em: 11 de jan. de 2019.

THOMAS, George B....[et al.]. **Cálculo**. Volume 2. 12. ed. – São Paulo: Pearson Education do Brasil, 2012.

VAZ, Welton. **Saiba mais como o python surgiu e qual o seu cenário atual**. Disponível em: <https://eusoudev.com.br/python-como-surgiu/>. Acesso em: 09 de jan de 2019.

WEISSTEIN, Eric W. **Esfera**. De *MathWorld - um* recurso da Web de Wolfram. Disponível em: <http://mathworld.wolfram.com/Sphere.html>. Acesso em: 28 de abr de 2019.

WIKIPÉDIA. **Informática**. Disponível em: <https://pt.wikipedia.org/wiki/inform%c3%a1tica>. Acesso em: 07 de jan. de 2019.

_____. **Programa de computador**. Disponível em: <https://pt.wikipedia.org/wiki/Programa_de_computador Acesso em: 08 de jan. de 2019.

_____. **Microsoft Visual Studio.** Disponível em:

<https://pt.m.wikipedia.org/wiki/Microsoft_Visual_Studio>. Acesso em: 14 de jun. 2019.

LISTA DE APÊNDICES

APÊNDICE A – Script da Geração de dados em um arquivo através de funções (Fortran)

```
Program Funcao_Gerar_Pontos
      !Gerar valores de y em função de x, para x inteiro.
      use portlib
      real(4) tempo
      doubleprecision x, y
      integer ponto, pontos
      print*,'Você deverá fornecer o intervalo x que deseja
      analisar'
      write(*,*) "Entre com x1: "
      read(*,*) x1
      write(*,*) "Entre com x2: "
      read(*,*) x2
      tempo = SECNDS(0.0)
      x=0.0
      open(7, file= "Dados.txt")
            do x = x1, x2
                  y = x**2
                  write(7,40) x, y
                  40 format( 2 ( 1pe10.2 ) )
            end do
      close(7)
      tempo = SECNDS(tempo)
      print*, 'tempo gasto: ', tempo, ' segundos'
end program
```

APÊNDICE B – Script da Geração de dados em um arquivo através de funções (python)

```
#Gerar valores de y em função de x, para x inteiro.
import time
print ('Você deverá fornecer o intervalo x que deseja
analizar')
x1 = int(input("Digite x1: "))
x2 = int(input("Digite x2: "))
inicio=time.time()
arq=open("Dados.txt", "w")
      x = 0
      for x in range(x1,x2+1):
      y = x**2
      arq.write(str(x) + " " + str(y) + "\n")
arq.close()
fim=time.time()
print ("Tempo gasto: ", fim-inicio, "segundos")
```

APÊNDICE C – Script dos Elementos da esfera em uma matriz (Fortran)

```
program Elementos_Esfera_Matriz
      ! Calculo do diâmetro, área, volume, área do fuso e
      volume da cunha, volume da calota e área da calota de uma
      esfera, dado o raio, altura da calota e graus do fuso e
      cunha; escrevendo em matriz.

      use portlib

      integer i,m

      doubleprecision r,pi,D,A,V,Af,Vc,alfa,h,Vca,Aca

      doubleprecision, allocatable :: matriz (:,:)

      real(4) tempo

      print*, 'Você deverá fornecer o valor do raio da esfera,
     o número de graus "alfa", a altura da calota e o número de
     repetições do cálculo.'

      print*,'Digite o raio: '

      read*,r

      print*,'Digite alfa (em graus): '

      read*,alfa

      print*,'Digite a altura da calota: '

      read*,h

      print*,'Digite o número de linhas: '

      read*,m

      tempo = SECNDS(0.0)

      allocate (matriz(m,7))
```

```
      matriz = 0
      pi = 3.141592653589793
      D = 2.*r
      A = 4.*pi*r**2.
      V = (4./3.)*pi*r**3.
      Af = (alfa/360.)*A
      Vc = (alfa/360.)*V
      Vca = (1./3.)*pi*h*h*(3.*r-h)
      Aca = 2.*pi*r*h
      do i=1,m
            matriz(i,1)=D
            matriz(i,2)=A
            matriz(i,3)=V
            matriz(i,4)=Af
            matriz(i,5)=Vc
            matriz(i,6)=Vca
            matriz(i,7)=Aca
      end do
      tempo = SECNDS(tempo)
      print*, 'Tempo gasto: ', tempo, ' segundos'
end program
```

APÊNDICE D – Script dos Elementos da esfera em uma matriz (Python)

```
#Calculo do diâmetro, área, volume, área do fuso e volume da cunha, volume da calota e área da calota de uma esfera, dado o raio, altura da calota e graus do fuso e cunha; escrevendo em matriz.

from math import pi

import numpy as np

import time

#print('Você deverá fornecer o valor do raio da esfera, o número de graus "alfa", a altura da calota e o número de repetições do cálculo.')

r = float(input('Digite o raio: '))

alfa = float(input('Digite alfa (em graus): '))

h = float(input('Digite a altura da calota: '))

n = int(input('Digite o número de repetições: '))

inicio = time.time()

D = 2*r

A = 4*pi*r**2

V = (4/3)*pi*r**3

Af = (alfa/360)*A

Vc = (alfa/360)*V

Vca = (1/3)*pi*h*h*(3*r-h)

Aca = 2*pi*r*h
```

```
matriz = np.zeros((m,7))

matriz[:] = [D, A, V, Af, Vc, Vca, Aca]

fim = time.time()

print ('Tempo gasto: ', fim-inicio, ' segundos')
```

APÊNDICE E – Script da Obtenção da raiz de uma função através do Método da Bisseção (Fortran)

```
program Medir_Bissecao
      !Método da Bisseção
      real(4) tempo
      doubleprecision a,b,p,tol
      integer i,j,k,nm
      print*, 'Você deverá fornecer os intervalos a e b,
      tolerancia, n° máximo de interações e n° de cálculos'
      print*,'Digite o intervalo a: '
      read*,a
      print*,'Digite o intervalo b:'
      read*,b
      print*,'Digite o número máximo de interações'
      read*,nm
      print*,'Digite a tolerância'
      read*,tol
      print*,'Digite o número de cálculos'
      read*,k
      tempo = SECNDS(0.0)
      do i=1,k
            a=0.0
            b=2.0
            call Calcular_bissecao(a,b,nm,tol)
      enddo
      tempo = SECNDS(tempo)
      print*, 'Tempo gasto: ', tempo, ' segundos'
      stop
end program
subroutine Calcular_bissecao (a,b,nm,tol)
      real(4) tempo
      doubleprecision a,b,p,tol
      integer i,j,nm
```

```
      i=1
      fa=f(a)
      do while(i.le.nm)
            p=a+(b-a)/2.
            fp=f(p)
             if((fp.eq.0.) .or.((b-a)/2..lt.tol) )then
                  goto 5
            endif
            i=i+1
            if(fa*fp.gt.0.)then
                  a=p
                  fa=fp
            else
                  b=p
            endif
      enddo
      print*, 'O método falhou depois de ',nm,' interações'
5     continue
      return
end subroutine
!Função
real function f(x)
doubleprecision x
      f = x**4.-9.*x**3.- 2.*x**2. +120.*x - 130.
      return
end
```

APÊNDICE F – Script da Obtenção da raiz de uma função através do Método da Bisseção (Python)

```
#Calcular raízes através do Método da Bisseção
from __future__ import division
import math
import time
def funcao(x):
      f= x**4.-9.*x**3.- 2.*x**2. +120.*x - 130.
      return f
def medir_bissecao():
      print('Você deverá fornecer os intervalos a e b,
tolerância, n° máximo de interações e n° de cálculos')
      a=float(input('Digite o intervalo a: '))
      b=float(input('Digite o intervalo b: '))
      nm=float(input('Digite o número máximo de interações: '))
      tol=float(input('Digite a tolerância: '))
      k=int(input('Digite o número de cálculos: '))
      inicio = time.time()
      for i in range(k):
            calcular_bissecao(funcao, a, b, nm, tol)
      fim = time.time()
      print ("Tempo gasto: ", fim - inicio)
def calcular_bissecao(funcao, a, b, nm, tol):
f=funcao
      i = 1
      fa = f(a)
      while (i <= nm):
           p = a + (b-a)/2
           fp = f(p)
            if ((fp == 0) or ((b-a)/2 < tol)):
                  break
            i = i+1
            if (fa * fp > 0):
```

```
            a = p
            fa = fp
      else:
            b = p
if i > nm:
      raise NameError('Num. max. de iter. excedido!');
```

APÊNDICE G – Script da Aproximação do limite de uma série geométrica (Fortran)

```
program Serie_Geometrica
      !Cálculo do somatório de uma série geométrica.
      integer n,i,limite
      doubleprecision sg
      print*, 'Você deverá fornecer o número de termos para
      obter o somatório da série.'
      Print*,'Entre com o número de termos: '
      read*,n
      limite = 1.0
      do i = 0,n
            sg = limite - 1./2.**n
      end do
      print *, 'O Somatório é = ', sg
end program
```

APÊNDICE H – Script da Aproximação do limite de uma série geométrica (Python)

```
#Cálculo do somatório de uma série geométrica.
from decimal import Decimal
print('Você deverá fornecer o número de termos para obter o
somatório da série.')
n = int(input('Digite o número de termos: '))
limite = 1.0
for i in range(0,n):
     sg = Decimal(limite-1/2**n)
print ("O somatório é ", sg)
```

ANEXOS

ANEXO A - Método da Bissecção (Fortran)

```
program bissecção
      real a,b,p,tol
      integer,n0
      Print*,'entre com a'
      read*,a
      Print*,'entre com b'
      read*,b
      Print*,'entre com N0'
      read*,n0
      Print*,'entre com a tolerancia'
      read*,tol
       i=1
      fa=f(a)
      do while(i.le.n0)
            p=a+(b-a)/2.
            fp=f(p)
            if((fp.eq.0.) .or.((b-a)/2..lt.tol) )then
                  print*,p
                  goto 5
            endif
      i=i+1
            if(fa*fp.gt.0.)then
                  a=p
                  fa=fp
            else
                  b=p
                  endif
      enddo
      print*,'o metodo falhou depois de ',n0,' interacoes'
5     stop
```

```
end
!!!!!!!!!!!!!!!!!funcao!!!!!!!!!!!
real function f(x)
real x
f=2.*x-1.
 return
end
```

ANEXO B - Método da Bissecção (Python)

```
from __future__ import division

def bissecao(f, a, b, TOL, N):

      i = 1

      fa = f(a)

      while (i <= N):

            #iteracao da bissecao

            p = a + (b-a)/2

            fp = f(p)

            #condicao de parada

            if ((fp == 0) or ((b-a)/2 < TOL)):

                  return p

            #bissecta o intervalo

             i = i+1

            if (fa * fp > 0):

                  a = p

                  fa = fp

            else:

                   b = p

raise NameError('Num. max. de iter. excedido!');
```