

# **Universidade da Beira Interior**

## **Departamento de Informática**



**Departamento de  
Informática**

### **Nº 1 - 2025: *Vetores e companhia***

Elaborado por:

**Gabriel Salazar N: 55652**  
**Jorge Altamirano N: 55456**  
**Daniel Colavito N: 55428**  
**Leandro Cabrera N: 55459**

Orientador:

**Professor Tiago Filipe Dias dos Santos Roxo**

13 de dezembro de 2025



# ***Agradecimentos***

A conclusão deste trabalho, bem como da grande maior parte da minha vida académica, não seria possível sem a ajuda de todos aqueles que, de forma direta ou indireta, contribuíram para o meu percurso.

Em primeiro lugar, gostaria de expressar a minha sincera gratidão ao professor responsável pela unidade curricular de Laboratório de Informática, pelo acompanhamento constante, pelas orientações precisas e pela disponibilidade em esclarecer dúvidas ao longo de todo o semestre.

Agradeço também aos meus colegas de grupo, pelo espírito de colaboração, pela partilha de conhecimentos e pelo esforço conjunto que tornaram possível a realização deste trabalho.

Não poderia deixar de mencionar a Universidade da Beira Interior, pela oportunidade de crescimento académico e pessoal que me proporcionou, bem como aos funcionários e técnicos dos laboratórios, que contribuíram para criar um ambiente favorável ao desenvolvimento deste projeto.

Por fim, um agradecimento especial à minha família e amigos, pelo apoio, incentivo e compreensão em todos os momentos deste percurso académico.

A todos, o meu sincero obrigado.



# Conteúdo

<b>Conteúdo</b>	<b>iii</b>
<b>Lista de Figuras</b>	<b>v</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Enquadramento . . . . .	1
1.2 Motivação <b>UBI!</b> . . . . .	1
1.3 Objetivos . . . . .	1
1.4 Organização do Documento . . . . .	1
<b>2 Estado da Arte</b>	<b>3</b>
2.1 Introdução . . . . .	3
2.2 Computação Científica em Python . . . . .	3
2.3 Conclusões . . . . .	3
<b>3 Tecnologias e Ferramentas Utilizadas</b>	<b>5</b>
3.1 Introdução . . . . .	5
3.2 Ferramentas e Recursos Tecnológicos . . . . .	5
3.2.1 Python . . . . .	5
3.2.2 Construção del proyecto . . . . .	6
3.2.3 Visual Studio Code (VS Code) . . . . .	6
3.2.4 Large Language Models (LLMS) . . . . .	6
3.3 Conclusões . . . . .	7
<b>4 Implementação e Testes</b>	<b>9</b>
4.1 Introdução . . . . .	9
4.2 Gestão da Interface Híbrida . . . . .	9
4.3 Algoritmos de Cálculo . . . . .	10
4.3.1 List Comprehensions . . . . .	10
4.4 Operações matriciais avançadas . . . . .	10
4.4.1 Outer Output . . . . .	10
4.5 Conclusões . . . . .	12
<b>5 Conclusões e Trabalho Futuro</b>	<b>13</b>
5.1 Conclusões Principais . . . . .	13
5.2 Trabalho Futuro . . . . .	13
<b>A Código-Fonte e Ilustrações</b>	<b>15</b>
A.1 Excerto de Código . . . . .	15
A.2 Imagem do trabalho no Google Colab . . . . .	16
Bibliografia . . . . .	17



# ***Lista de Figuras***

4.1	Comportamento da função <code>matriz_2x16</code> onde $N = 16$ . . . . .	11
4.2	Fluxo da Hierarquia de Dados no Produto Externo. . . . .	12
A.1	Extrato de trabalho no Google Colab . . . . .	16





# Glossário

**LaTeX** A ferramenta padrão na comunidade científica e académica para a escrita de teses e artigos. É inigualável na gestão de referências bibliográficas e na representação de fórmulas matemáticas complexas com precisão tipográfica. 5

**Don't repeat yourself (DRY)** A aplicação deste princípio passa por abstrair a lógica repetida em funções, classes ou módulos reutilizáveis. Isto melhora a manutenibilidade do software, pois uma alteração ou correção de bug precisa ser feita apenas num único sítio, propagando-se automaticamente a todo o sistema (Summerfield, 2010). 10

**LLMs** O motor tecnológico por trás da IA generativa moderna. Para programadores, estes modelos funcionam como assistentes super-avanzados capazes de escrever código, explicar erros (bugs), criar testes unitários e converter código de uma linguagem para outra em segundos (Patro, Namboodiri, & Agneeswaran, 2025). iii, 6, 7

**VS Code** Um editor de código-fonte gratuito e open-source desenvolvido pela Microsoft. Redefiniu o desenvolvimento moderno ao combinar a simplicidade de um editor de texto leve com funcionalidades poderosas típicas de um IDE, graças ao seu gigantesco ecossistema de extensões.. iii, 6, 7

**argparse** Uma ferramenta de parsing que define que argumentos o programa espera receber. Converte automaticamente as strings introduzidas na linha de comando em objetos Python nativos (como inteiros ou floats) e gera automaticamente mensagens de ajuda e erro caso o utilizador introduza dados inválidos (Summerfield, 2010). 6, 9

**Data Augmentation** Um conjunto de técnicas cruciais em Machine Learning (especialmente Deep Learning) para aumentar artificialmente o tamanho e a diversidade dos dados de treino. O objetivo principal é prevenir o overfitting (quando o modelo "decora" os dados) e garantir que o modelo consegue generalizar para dados do mundo real. 10

**Doxygen** Um poderoso sistema que não só documenta funções, mas também ajuda a visualizar a arquitetura do software. É capaz de gerar diagramas de dependência, gráficos de chamadas e hierarquias de classes automaticamente, facilitando a compreensão de projetos complexos (Machemedze, Gondo, & Ndlovu, 2025). 5–7

**GitHub** Frequentemente chamado de "rede social para programadores". É o espaço onde equipas de todo o mundo colaboram em projetos (Open Source ou privados), reveem o código uns dos outros e acompanham o progresso do desenvolvimento. 5–7

- Google Colab** Uma plataforma gratuita da Google que permite escrever e executar código Python diretamente no navegador. Funciona como um Jupyter Notebook na nuvem, sem necessidade de qualquer instalação ou configuração no computador do utilizador (Naik, 2023). 5–7
- HTML** A linguagem de marcação padrão utilizada para criar e estruturar páginas na Internet. Não é uma linguagem de programação no sentido estrito (não tem lógica nem variáveis), mas sim o esqueleto que diz ao navegador onde colocar os cabeçalhos, parágrafos, imagens e botões (Beningo, 2017). 6
- input** A função padrão do Python utilizada para "falar" com o utilizador. Quando o programa encontra esta função, pausa a execução imediatamente e espera que a pessoa escreva algo no teclado e carregue em "Enter". É a forma mais simples de recolher dados externos (Summerfield, 2010). 9
- math** Biblioteca desenhada para operações eficientes com números de ponto flutuante (floats). É ideal para cálculos escalares (número a número), mas para operações complexas com vetores ou matrizes, geralmente é substituída pela biblioteca Numpy (Summerfield, 2010). 5, 7
- numpy** A biblioteca base para toda a computação científica em Python. Introduce objetos chamados arrays multidimensionais (matrizes), que são infinitamente mais eficientes e rápidos do que as listas tradicionais do Python, permitindo realizar cálculos matemáticos complexos em grandes volumes de dados (Gupta & Bagchi, 2024). viii, 6, 7
- open-source** Um modelo de desenvolvimento de software onde o código-fonte é disponibilizado publicamente para todos. Ao contrário do software proprietário, permite que qualquer pessoa estude, modifique, melhore e distribua o software, promovendo a inovação colaborativa e a transparência.. vii
- opção do menuinterativo 1** Retorno de um elemento aleatório desse vetor. 5, 6
- opção do menuinterativo 10** Leitura de um novo vetor 1x16, cálculo e devolução da matriz 16x16 resultante do produto do vetor inicial com o novo vetor gerado;. 10
- opção do menuinterativo 11** Cálculo do determinante da matriz gerada anteriormente;. 6, 10
- opção do menuinterativo 2** Cálculo do seno (sin). 5
- opção do menuinterativo 3** Construção de uma matriz 2 por 16. 5, 6, 10, 11
- opção do menuinterativo 7** Ajuda. 6, 7
- Python** A linguagem líder indiscutível em Ciência de Dados e Machine Learning. Destaca-se pelo seu imenso ecossistema de bibliotecas (como Pandas ou TensorFlow) que permitem processar grandes volumes de informação e automatizar tarefas com poucas linhas de código (Summerfield, 2010). 5–7
- random** Um módulo da biblioteca padrão do Python usado para gerar números pseudo-aleatórios. É a ferramenta ideal quando precisas de simular o lançamento de dados, baralhar uma lista de músicas ou escolher um item à sorte numa lista (Summerfield, 2010). 5–7, 11

**sys** Um módulo fundamental que serve de ponte entre o teu código e o próprio interpretador Python (Summerfield, 2010). 6



# ***Acrónimos***

**CLI** Interface de Linha de Comando. 6, 9

**DRY** Don't repeat yourself. vii, 10, 12

**LLMS** Large Language Models. iii, vii, 6, 7

**VS Code** Visual Studio Code. iii, vii, 6, 7



# **Introdução**

## **1.1 Enquadramento**

Este relatório foi feito no contexto da unidade curricular de Laboratórios de Informática da **UBI!** (**UBI!**). Foi na **UBI!** que desenvolvi todo o trabalho. O projeto consiste na criação de uma aplicação de consola para análise estatística e matemática de um vetor de dados numéricos.

## **1.2 Motivação UBI!**

A motivação técnica para este projeto é a aplicação prática de conceitos fundamentais de programação (validação de dados, estruturas de controlo) e a integração de bibliotecas de computação científica (numpy) para resolver problemas matemáticos complexos, como operações com matrizes.

## **1.3 Objetivos**

O objetivo principal é criar uma aplicação em Python que:

- Solicite ao utilizador 16 números inteiros;
- Valide que os números pertencem ao intervalo  $[-10, 27]$ ;
- Apresente um menu com 12 operações, incluindo cálculos de seno, ordenação, geração de matrizes e cálculo de determinantes.

Objetivos secundários incluem a utilização de `git` para controlo de versões e a familiarização com o ambiente Linux.

## **1.4 Organização do Documento**

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

- O Capítulo 2, Estado da Arte, descreve as bibliotecas Python usadas;
- O Capítulo 3, Tecnologias e Ferramentas Utilizadas, detalha o *software* usado no desenvolvimento, conforme explorado nos guias laboratoriais;

- O Capítulo 4, Implementação e Testes, apresenta a lógica do código Python e os resultados da validação;
- O Capítulo 5, Conclusões e Trabalho Futuro, resume o trabalho e sugere melhorias.



## ***Estado da Arte***

### **2.1 Introdução**

Este capítulo descreve as principais tecnologias de *software* que servem de base ao projeto, com foco no ecossistema de computação científica da linguagem Python.

### **2.2 Computação Científica em Python**

A linguagem Python (Guia 6) é uma linguagem interpretada de alto nível, conhecida pela sua sintaxe clara e vasta biblioteca de módulos. O módulo `math` fornece acesso a funções matemáticas básicas (como `math.sin`), enquanto o módulo `random` permite a seleção de elementos aleatórios (?, ?).

Para operações vetoriais e matriciais complexas, a biblioteca `numpy` (Numeric Python) é o padrão da indústria. Ela fornece objetos `array` de alta performance e rotinas para manipulação destes, incluindo álgebra linear, como o cálculo de determinantes (`numpy.linalg.det`) (?, ?).

### **2.3 Conclusões**

A escolha de Python e `numpy` permitiu uma implementação rápida e robusta, focando na lógica do problema em vez de na gestão de memória de baixo nível, ao mesmo tempo que oferece um desempenho elevado para operações matemáticas.



## Tecnologias e Ferramentas Utilizadas

### 3.1 Introdução

Ao longo deste projeto foram usadas várias ferramentas; a mais importante foi, sem dúvida, a linguagem de programação, já que esta é o coração do projeto. A nossa escolha para a linguagem de programação era clara: Python. A sua sintaxe simples e o seu amplo catálogo de Bibliotecas e módulos, além de ser uma linguagem com a qual todo o grupo se encontrava bastante familiarizado, fizeram de Python a escolha ideal. A primeira plataforma que se usou para a criação do código foi Google Colab e posteriormente foi finalizado em . Embora fosse o mais importante, não era a única ferramenta que usaríamos; para apresentar o projeto tínhamos escolhas, por isso escolhemos três:  $\text{\LaTeX}$ , Github e Doxygen. Cada uma ajuda a apresentar o projeto de forma distinta e amplia a nossa capacidade de comunicar o que faz o nosso Script. Neste capítulo será explicado como foram utilizadas para alcançar o produto final.

### 3.2 Ferramentas e Recursos Tecnológicos

#### 3.2.1 Python

Python foi a nossa ferramenta principal ao longo deste projeto. O código inteiro está escrito nele, o que nos facilitou o desenvolvimento de muitas das funções que tínhamos que implementar. O seu catálogo de Bibliotecas e Módulos foi essencial para nós, pois muitas implementações necessitaram destes para tornar o código mais eficiente e económico. A seguir, listaremos todas as bibliotecas e módulos utilizado.

- **math**

glsmat Section Na opção do menu interativo 2 do nosso menu interativo "*Cálculo do seno ( $\sin$ )*" indica o cálculo do seno do vetor, para isto usamos a biblioteca math que vem integrada em Python. Com o seu módulo

*math.sin()* entrega-nos um método direto e validado para o cálculo da função trigonométrica sobre todos os elementos do Vetor, além de nos assegurarmos que o resultado não tenha erros.

- **random** A biblioteca /random encarrega-se da geração de dados aleatórios, chaves para a resolução de algumas operações. Especificamente a opção do menu interativo 1 e opção do menu interativo 3, requeriam os seguintes módulos:

1. *random.choice()* Na opção do menu interativo 1 o módulo *random* encarrega-se de selecionar um elemento aleatório do vetor gerado pelo utilizador.
2. *random.randint()* Na opção do menu interativo 3 o módulo *random* é essencial, já que gera números inteiros dentro do intervalo (-10,27) para introduzir na segunda linha da matriz 2\*16

- **argparse**

Na opção do menu interativo 7 o módulo integrado *argparse* tem como função mostrar uma Interface de Linha de Comando (CLI) ao utilizador.

- **sys**

A biblioteca *sys* controla o fluxo de finalização do programa para comunicar o seu estado ao sistema operativo: utiliza *sys.exit(1)* para parar a execução de imediato reportando um erro se os dados do vetor inicial forem inválidos, e *sys.exit(0)* para terminar o programa de forma limpa com sucesso após executar uma opção automática, evitando assim que o código continue e ative o menu interativo desnecessariamente.

- **numpy**

A opção do menu interativo 11 foi a mais complicada para nós, esta pede para calcular o determinante de uma matriz 16\*16, para esta tarefa tentámos um ciclo */for*, o que resultou no código referenciado A.1

O código funcionava perfeitamente, entregava o determinante da matriz, mas realizando uma pequena investigação e com a ajuda do modelo **•LLMS Google Gemini**, encontramos esta biblioteca de Python, que com a sua sub-biblioteca *llinag* e o seu módulo *llinag.det*, resolvia o problema com uma redução significativa do código. Simplificando o problema e aumentando a eficiência do nosso programa.

### 3.2.2 Construção del proyecto

No nosso projeto, a colaboração em grupo foi vital. Cada integrante devia estar atento e ajudar com o código, assim procurámos uma plataforma na qual pudéssemos trabalhar sem a necessidade de estar presentes um com o outro. Escolhemos Google Colab para esta tarefa. Permitiu-nos avançar no código até o termos completo e funcional. A.2

### 3.2.3 VS Code

Decidimos mudar para o VS Code por duas razões: a principal foi que mudámos o código inteiro, passámos de uma estrutura repleta de ciclos *for* e buscas infinitas para uma mais limpa e eficiente. Pronto para ser usado, a plataforma perfeita para a tarefa foi o VS Code. Ao ter integrações com o Github, pudemos continuar a contribuir para o projeto de onde quer que nos encontrássemos. O VS Code foi vital para a introdução do Doxygen no projeto, para o podermos ter também numa página em HTML. A seguir deixarei o *link* para o nosso *repo* no Github onde se evidencia a combinação de todas estas tecnologias: [https://github.com/Gaboespada123/Proyecto\\_Informatica](https://github.com/Gaboespada123/Proyecto_Informatica)

### 3.2.4 LLMS

A implementação de LLMS neste projeto teve dois objetivos claros: Não fazer o trabalho por nós e Agilizar o nosso trabalho. Como se viu previamente, a primeira

intervenção de um LLMS foi na opção do menu interativo 7s, onde este mesmo ajudou a encontrar uma solução mais econômica para uma ordem que considerávamos longa e que frequentemente se quebrava ou tinha *bugs*. Apenas se usaram dois modelos LLMS neste projeto: Google Gemini e Github Copilot. Os quais ajudaram o grupo a: Traduzir, Buscar fontes fiáveis e citáveis, reduzir e reestruturar código.

### 3.3 Conclusões

A escolha das tecnologias foi muito importante para este trabalho; graças a elas, pudemos organizar, criar e apresentar o projeto de acordo com a nossa visão. Python abriu-nos as portas para muitas soluções para algumas operações que à primeira vista parecem complicadas e confusas, complementado pelas suas bibliotecas *math*, *random* e *numpy*, demonstrou ser uma escolha incrível para organizar e entender o problema através do código. A experiência de passar pelo Google Colab e mudar para o VS Code para a integração de Github e Doxygen deixa-nos com muita aprendizagem e um projeto pronto para apresentar sob qualquer circunstância.



## Implementação e Testes

### 4.1 Introdução

O objetivo principal deste *Script* era criar uma unidade de processamento matemático, cumprindo os requisitos previamente estabelecidos; além disso, devia ser fácil de entender e de manipular. A solução mais técnica para o fazer foi, eventualmente, separar em dois ficheiros o *Script*: `main.py` e `calculos.py`, um encarregado de manter os cálculos e estruturas mais complexas e outro agrupando tudo num espaço mais simples, sem nunca perder de vista o utilizador, mostrando as funcionalidades de forma progressiva, desde os cálculos mais simples até às operações matriciais avançadas. Além disso, a implementação do CLI resultou num *Script* que pode ser usado e corrigido facilmente, se necessário.

### 4.2 Gestão da Interface Híbrida

A estrutura *Script*, ao dividir-se em módulos independentes, oferece-nos a comodidade de evitar criar um monólito e opta por uma estrutura modular onde tudo tem uma razão única para mudar. Poderia ser conceptualizado como um modelo Camada de Apresentação *versus* Camada Lógica. A seguir, expandiremos o funcionamento do *Script* usando como base o estabelecido:

- `main.py`  
Funciona como um gestor do ambiente. Encarrega-se de receber o input do utilizador e, verificando o mesmo, abstrai a complexidade e comunica-a com o núcleo do sistema. A sua função principal é orquestrar a interface, comunicando ao utilizador o que acontece dentro do código, interpretando os argumentos da linha de comandos `argparse 3.2.1`, além de determinar o ciclo de vida do *Script*.
- `calculos.py`  
Este tem como função ser o núcleo de processamento do sistema; as suas funções são agnósticas, são independentes da origem do dado com que deve trabalhar, e só recebe dados e os transforma. No seu desenho também existem variáveis globais, como pode ser a variável `matriz_resultante_final`, a qual se utiliza como persistência temporária que guarda os resultados de algumas funções para serem utilizadas por outras. Estas funções também podem ser encontradas em `main.py`, por exemplo, o vetor `Vector`, que mantém a harmonia entre ambos os módulos.

- `procesar_escolha`

Componente central de orquestração da estrutura, criado especificamente para cumprir o princípio DRY da programação. A sua função não é calcular, mas sim delegar a função concreta de acordo com a escolha do utilizador. Sem este componente, teríamos de reescrever o menu e todo o `main.py` sempre que uma função fosse terminada. É também o encarregado de terminar o ciclo de vida do sistema: enquanto `procesar_escolha` se mantiver em `False`, o sistema repetir-se-á infinitamente, mas se `procesar_escolha` comunicar a `main.py` que o utilizador mudou o seu booleano para `True`, o programa finaliza..

## 4.3 Algoritmos de Cálculo

Nesta Secção, descreve-se o uso das ferramentas utilizadas nas operações aritméticas e lógicas trabalhadas sobre o vetor `Vector` trabalhado em `calculos.py` como um `array`, explicadas em detalhe em 3.2.1. Expor-se-á o uso do *Python way* ao longo do código e o apoio em bibliotecas para assegurar a precisão no resultado.

### 4.3.1 List Comprehensions

Esta secção encarrega-se de descrever a implementação de técnicas nativas de Python, conhecidas como *List Comprehensions*, para evitar a redundância de ciclos. Utilizaram-se em grande medida conceitos de Python, como as listas vazias trabalhadas sob o conceito *Mapping&Filtering*. Por exemplo, nas primeiras linhas de `main.py`, para definir `Vector` e `matriz_resultante_final`, ambas explicadas em 4.2.

Tabela 4.1: Para todo  $f(x)$

Função	Tipo de Operação	Sintaxis Clave
<code>calcular_seno</code>	Mapping	<code>[calcular_seno for elemento in vector]</code>
<code>divisivel_3</code>	Filtering	<code>[expresion for elemento in vector if condición]</code>

Outro exemplo da utilização de *List Comprehensions* é a opção do menu interativo 3, onde o sistema demonstra a sua capacidade de *Data Augmentation*. Criar uma matriz de ordem  $2 \times N$  a partir de `vector` (dado carregado pelo utilizador). A função encarregada de processar esta tarefa é `matriz_2x16`, explicada em 4.3.1

## 4.4 Operações matriciais avançadas

Esta secção dedica-se a explicar a opção do menu interativo 10, encarregada de transformar um sistema unidimensional num bidimensional, focando na implementação de um produto externo de vetores (opção do menu interativo 10) e a sua análise (opção do menu interativo 11) (visto em 3.2.1).

### 4.4.1 Outer Output

Ao usar a opção do menu interativo 10, o sistema entende que deve converter dois vetores numa matriz bidimensional de ordem  $N \times N$ . Não é uma multiplicação por um produto escalar, é uma operação que aumenta as dimensões de um vetor. Esta



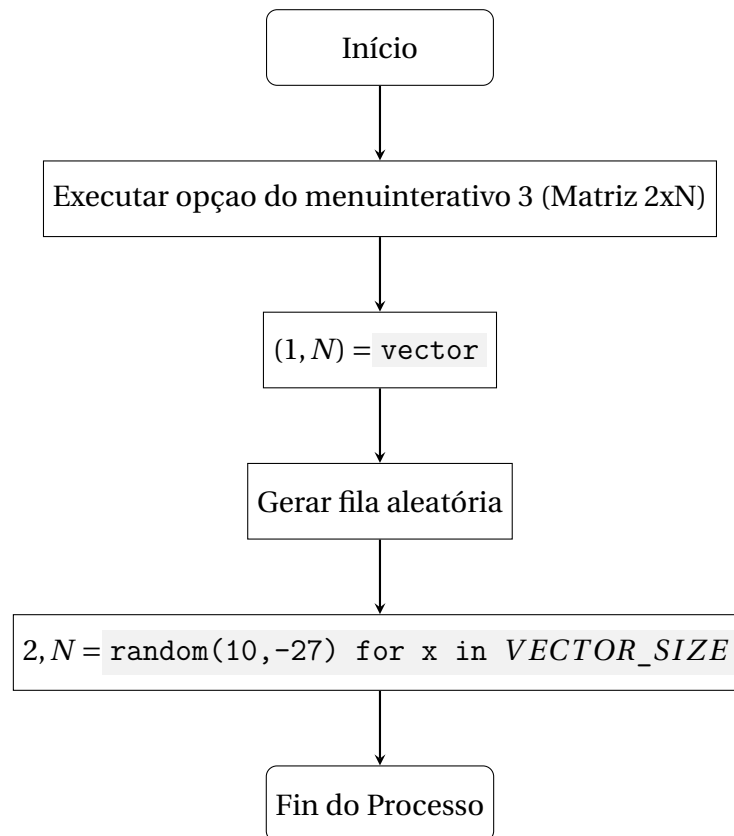


Figura 4.1: Comportamento da função `matriz_2x16` onde  $N = 16$ .

operação pode ser definida tal como apresentado em 4.4.1

$$M_{ij} = V_i \times U_j$$

Esta operação descreve o que ocorre no nosso código: temos um vetor  $V$  e introduzimos um vetor  $U$ , o algoritmo converte-a na estrutura bidimensional que queremos. As variáveis do código encarregadas da operação são referenciadas em 4.4.1.

Notação Matemática	Variável no Código
$V$	<code>vector_inicial</code>
$U$	<code>vector_novo</code>
$M_{ij}$	<code>row.append</code>
$ij$	<code>VECTOR_SIZE</code>

Tabela 4.2: Explicado no código.

A ordem em que o nosso código realiza a operação é muito importante: ao estar num ambiente controlado por `List Comprehensions`, é importante conhecer a ordem em que o sistema realiza esta operação complexa.

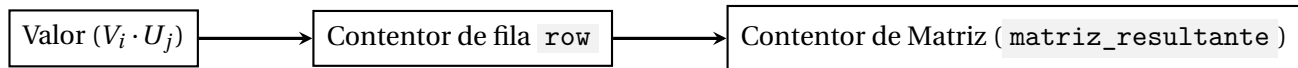


Figura 4.2: Fluxo da Hierarquia de Dados no Produto Externo.

## 4.5 Conclusões

Em resumo, o objetivo principal foi cumprido, fazendo uso da separação de responsabilidades nos módulos `main.py` e `calculos.py`, o que nos permite manutenibilidade e a centralização dos cálculos sob o princípio DRY. A nível algorítmico, alcançou-se a simplicidade e otimização, fazendo uso de *List Comprehensions* (para eficiência em tarefas sequenciais) e otimizações com o uso de *Numpy* (para precisão em Álgebra Linear). Finalmente, o sistema demonstra domínio no manuseamento de estruturas complexas e delegação das mesmas para manter a simplicidade sem sacrificar o processamento matemático.

## ***Conclusões e Trabalho Futuro***

### **5.1 Conclusões Principais**

No final deste projeto, podemos ver que o objetivo "Apresente um menu com 12 operações, incluindo cálculos de seno, ordenação, geração de matrizes e cálculo de determinantes" foi cumprido com êxito. Nos capítulos anteriores evidenciaram-se as soluções utilizadas para cumprir todos os objetivos propostos.

O trabalho em grupo foi essencial para o conseguir, como se evidenciou ao longo deste relatório. Permitiu desenvolver os módulos e encontrar soluções para os problemas de maneira rápida e otimizada, além de que os integrantes deste grupo encontraram crescimento ao desenvolver o sistema.

### **5.2 Trabalho Futuro**

Esta secção aborda as perspetivas de continuidade do projeto, identificando melhorias e novas funcionalidades que, embora pertinentes, não foram incluídas no âmbito inicial devido a restrições de tempo ou recursos. O trabalho futuro sugere caminhos para elevar a maturidade da solução desenvolvida. Durante os testes, identificou-se que seria interessante refinar a interface de utilizador e remover limitações no número de elementos do vetor. Embora a solução atual cumpra os requisitos, o sistema poderia ser significativamente otimizado para melhorar a experiência de utilização e o desempenho da aplicação em cenários de maior carga.



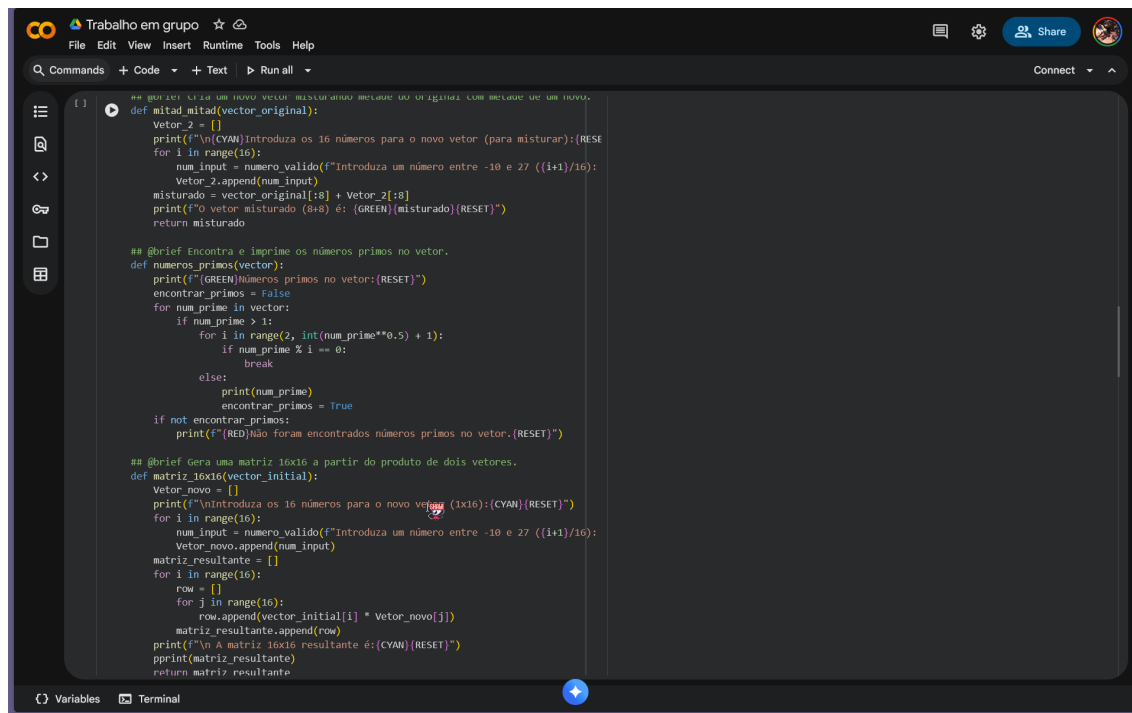
# A

## Código-Fonte e Ilustrações

### A.1 Excerto de Código

```
1 elif escolha == "10":
2     print("\nA calcular determinante (isto pode demorar)...")
3
4     # reconstrói a matriz (produto exterior)
5     matriz = []
6     for a in Vetor:
7         linha = []
8         for b in Vetor:
9             linha.append(a * b)
10            matriz.append(linha)
11
12    # cópia para evitar alterar
13    A = [linha[:] for linha in matriz]
14
15    det = 1
16    n = 16
17
18    for i in range(n):
19        if A[i][i] == 0:
20            trocada = False
21            for k in range(i+1, n):
22                if A[k][i] != 0:
23                    A[i], A[k] = A[k], A[i]
24                    det *= -1
25                    trocada = True
26                    break
27            if not trocada:
28                det = 0
29                break
30            pivot = A[i][i]
31            det *= pivot
32            for j in range(i+1, n):
33                A[i][j] = A[i][j] / pivot
34            for k in range(i+1, n):
35                fator = A[k][i]
36                for j in range(i+1, n):
37                    A[k][j] -= fator * A[i][j]
38
39    print(f"Determinante = {det}")
```

## A.2 Imagem do trabalho no Google Colab



```
## @brief Cria um novo vetor misturando metade do original com metade de um novo.
def mitad_mitad(vector_original):
    vector_2 = []
    print(f"\n(CYAN)Introduza os 16 números para o novo vetor (para misturar):(RESET)"
    for i in range(16):
        num_input = numero_valido(f"Introduza um número entre -10 e 27 {(i+1)}/16):
        vector_2.append(num_input)
    misturado = vector_original[:8] + vector_2[:8]
    print(f"O vetor misturado (8+8) é: (GREEN){misturado}(RESET)")
    return misturado

## @brief Encontra e imprime os números primos no vetor.
def numeros_primos(vector):
    print(f"(GREEN)Números primos no vetor:(RESET)")
    encontrar_primos = False
    for num_prime in vector:
        if num_prime > 1:
            for i in range(2, int(num_prime**0.5) + 1):
                if num_prime % i == 0:
                    break
            else:
                print(num_prime)
                encontrar_primos = True
    if not encontrar_primos:
        print(f"(RED)Nao foram encontrados números primos no vetor.(RESET)")

## @brief Gera uma matriz 16x16 a partir do produto de dois vetores.
def matriz_16x16(vector_initial):
    vector_novo = []
    print(f"\nIntroduza os 16 números para o novo vetor (1x16):(CYAN){RESET}")
    for i in range(16):
        num_input = numero_valido(f"Introduza um número entre -10 e 27 {(i+1)}/16):
        vector_novo.append(num_input)
    matriz_resultante = []
    for i in range(16):
        row = []
        for j in range(16):
            row.append(vector_initial[i] * vector_novo[j])
        matriz_resultante.append(row)
    print(f"\nA matriz 16x16 resultante é:(CYAN){RESET}")
    pprint(matriz_resultante)
    return matriz_resultante
```

Figura A.1: Extrato de trabalho no Google Colab

## Bibliografia

- Beningo, J. (2017). Documenting firmware with doxygen. In *Reusable firmware development: A practical approach to apis, hals and drivers* (pp. 121–148). Springer.
- Gupta, P., & Bagchi, A. (2024). Introduction to numpy. In *Essentials of python for artificial intelligence and machine learning* (pp. 127–159). Springer.
- Machemedze, T. A., Gondo, M., & Ndlovu, A. (2025). Development of a secure privacy preserving bert-based extractive api documentation generator.
- Naik, P. G. (2023). *Conceptualizing python in google colab: Hands-on practical sessions*. Shashwat Publication.
- Patro, B. N., Namboodiri, V. P., & Agneeswaran, V. S. (2025). Spectformer: Frequency and attention is what you need in a vision transformer. In *2025 ieee/cvf winter conference on applications of computer vision (wacv)* (pp. 9543–9554).
- Summerfield, M. (2010). *Programming in python 3: a complete introduction to the python language*. Addison-Wesley Professional.

