Sociedade Brasileira de Educação Matemática

RIPEM

Entre Facilidades e Aprendizagens: Conectando Habilidades Matemáticas Através da Programação em Python

Jailson França dos Santos
Universidade Federal do Oeste da Bahia
Barreiras, BA — Brasil
✉ jailson.santos@ufob.edu.br
0000-0002-9847-4081

Ilton Ferreira de Menezes
Universidade Federal do Oeste da Bahia
Barreiras, BA — Brasil
✉ Ilton.menezes@ufob.edu.br
0000-0002-9590-6731

2238-0345 ISSN

10.37001/ripem.v15i3.4528 doi

Recebido • 24/03/2025
Aprovado • 25/06/2025
Publicado • 01/09/2025

Editor • Gilberto Januario

Resumo: O uso de tecnologias digitais tem transformado o ensino de Matemática, modernizando práticas pedagógicas e alinhando-se às metodologias ativas. No Brasil, o Ministério da Educação tem incentivado a inserção dessas inovações, incluindo como componente curricular o ensino de linguagem de Programação. Entre as linguagens disponíveis, destaca-se o *Python*, que oferece diversas bibliotecas voltadas à resolução de problemas matemáticos. Este artigo, de caráter teórico, realiza uma análise crítica sobre o uso dessas bibliotecas, discutindo os benefícios e desafios pedagógicos envolvidos. A metodologia baseia-se em uma análise teórica com apoio de exemplos ilustrativos, que evidenciam como a praticidade do *Python* pode tanto favorecer quanto limitar o desenvolvimento do pensamento matemático e computacional. Espera-se que a pesquisa contribua para a discussão sobre a inserção da Programação no ensino de Matemática, promovendo um ambiente mais interativo e alinhado às metodologias ativas.

Palavras-chave: Tecnologias Digitais. Linguagem de Programação. Pensamento Computacional e Matemático. Metodologias Ativas.

Between Facilities and Learning: Connecting Mathematical Skills Through Python Programming

Abstract: The use of digital technologies has transformed the teaching of mathematics, modernizing pedagogical practices and aligning it with active methodologies. In Brazil, the Ministry of Education has encouraged the inclusion of these innovations, including the teaching of programming languages as a curricular component. Among the available languages, *Python* stands out, offering several libraries aimed at solving mathematical problems. This theoretical article conducts a critical analysis of the use of these libraries, discussing the benefits and pedagogical challenges involved. The methodology is based on a theoretical analysis supported by illustrative examples, which show how the practicality of *Python* can both favor and limit the development of mathematical and computational thinking. It is expected that the research will contribute to the discussion on the inclusion of programming in the teaching of mathematics, promoting a more interactive environment aligned with active methodologies.

Keywords: Digital Technologies. Programming Language. Computational and Mathematical Thinking. Active Methodologies.

Entre las instalaciones y el aprendizaje: conectando habilidades matemáticas mediante la programación en Python

Resumen: El uso de tecnologías digitales ha transformado la enseñanza de las matemáticas, modernizando las prácticas pedagógicas y alineándolas con metodologías activas. En Brasil, el Ministerio de Educación ha incentivado la inclusión de estas innovaciones, incluyendo la enseñanza de lenguajes de programación como un componente curricular. Entre los lenguajes disponibles, destaca *Python*, que ofrece varias bibliotecas orientadas a la resolución de problemas matemáticos. Este artículo teórico realiza un análisis crítico del uso de estas bibliotecas, discutiendo los beneficios y los desafíos pedagógicos involucrados. La metodología se basa en un análisis teórico respaldado por ejemplos ilustrativos, que muestran cómo la practicidad de *Python* puede tanto favorecer como limitar el desarrollo del pensamiento matemático y computacional. Se espera que la investigación contribuya a la discusión sobre la inclusión de la programación en la enseñanza de las matemáticas, promoviendo un entorno más interactivo alineado con las metodologías activas.

Palabras clave: Tecnologías Digitales. Lenguaje de Programación. Pensamiento Computacional y Matemático. Metodologías Activas.

1 Introdução

O avanço das Tecnologias da Informação e Comunicação (TIC) tem impulsionado o desenvolvimento de ferramentas digitais que, integradas às Tecnologias Digitais da Informação e Comunicação (TDIC), têm transformado de maneira positiva o ambiente educacional, principalmente quando associadas a metodologias ativas (Morán, 2015). No ensino de Matemática, historicamente visto como uma disciplina desafiadora e muitas vezes temida, essas tecnologias têm desempenhado um papel crucial na modernização das práticas pedagógicas.

O uso adequado das TDIC tem viabilizado a adaptação das práticas pedagógicas às demandas educacionais, alinhando-se ao documento normativo do Ministério da Educação do Brasil, a Base Nacional Comum Curricular [BNCC] (Brasil, 2018). Essa integração possibilita um ensino mais dinâmico, interativo e conectado às realidades tecnológicas dos estudantes, promovendo aprendizagens mais significativas e engajadoras, segundo Papert (1980) e Geldreich e Hubwieser (2020). Além disso, o uso estratégico das TDIC incentiva a aplicação de conceitos matemáticos em contextos reais, fortalecendo o desenvolvimento de habilidades como o pensamento crítico, a resolução de problemas e a criatividade (Siqueira, 2022; Burton *et al.*, 1991). Assim, as tecnologias digitais não apenas ampliam os horizontes educacionais, mas também favorecem uma educação mais contextualizada e inclusiva.

Diante das contribuições que as TDIC têm proporcionado no ensino e na aprendizagem (Bacich & Moran, 2018; Selwyn, 2021), inseri-las de forma consciente em sala de aula pode ser um passo natural, especialmente no que se refere ao conteúdo de Programação de computadores para o auxílio no ensino de Matemática. Papert (1980), idealizador do Construcionismo, uma teoria desenvolvida a partir dos conceitos do Construtivismo de Jean Piaget, propôs o uso do computador como ferramenta de aprendizagem, permitindo que o estudante construísse seu próprio conhecimento. Essa abordagem incentiva o estudante a assumir um papel ativo no processo de aprendizagem, destacando a importância de experiências criativas e significativas que promovam um aprendizado mais qualificado.

Algumas linguagens de Programação, como *Python*, são amplamente reconhecidas por sua simplicidade, flexibilidade, praticidade e eficiência (Lutz, 2013). Além disso, possuem um rico arcabouço de bibliotecas com funções pré-definidas que facilitam a implementação de

soluções computacionais. Essas características e o suporte oferecido pelas bibliotecas prontas tornam o *Python* uma excelente escolha do ponto de vista computacional. No entanto, na perspectiva pedagógica, especialmente para estudantes que estão iniciando o aprendizado de linguagens de Programação, essa abordagem pode não ser uma metodologia eficaz. Isso porque o uso de funções prontas pode limitar a exploração dos processos implícitos, utilizado no desenvolvimento de um algoritmo, dificultando o desenvolvimento do pensamento computacional e do pensamento matemático, que estão intimamente interligados (Paper, 1980).

Portanto, o *Python* revela um grande potencial como ferramenta de apoio ao ensino de Matemática, especialmente por meio de bibliotecas como *NumPy*, *SymPy* e *SciPy*, que simplificam a resolução de problemas matemáticos mais complexos. Partindo desse pressuposto, surge uma reflexão importante, particularmente no contexto da inserção do ensino de linguagens de Programação a estudantes da educação básica. A questão que emerge é: *"Embora as bibliotecas matemáticas do Python ofereçam ferramentas poderosas para facilitar cálculos e análises, até que ponto a dependência excessiva dessas bibliotecas prontas pode comprometer o desenvolvimento de habilidades fundamentais, como o pensamento lógico, matemático e computacional? Ademais, será que essa dependência pode transmitir uma falsa sensação de domínio de competências matemáticas?"*

Diante dessa pergunta-problema, torna-se essencial essa discussão na comunidade científica. Por meio dela, é possível gerar hipóteses do tipo: i) o uso excessivo de funções prontas do *Python* pode limitar a compreensão dos conceitos fundamentais de Programação e conceitos matemáticos, prejudicando o desenvolvimento de habilidades essenciais; ii) depender apenas de funções prontas acarreta em dificuldades para evoluir em conceitos na área, além de criar uma falsa sensação de competência, dificultando a aquisição das habilidades necessárias no que tange ao pensamento matemático computacional.

Para sugerir um caminho para a questão-problema e corroborar com as hipóteses mencionadas, este artigo defende uma metodologia de ensino que se concentra em desenvolver algoritmos computacionais de forma detalhada, seguindo passo a passo, estabelecendo conexões entre Programação e Matemática. Isso estimula o aprimoramento de habilidades essenciais, como lógica matemática computacional, raciocínio algorítmico e a capacidade de resolver problemas complexos, conforme discutido por Stephens (2018). Essa metodologia não apenas reforça o pensamento lógico e analítico, mas também facilita a transferência de conhecimento e o aprendizado em novos contextos.

Assim, este artigo contribui para a Educação Matemática ao discutir criticamente duas abordagens no uso da linguagem *Python*: de um lado, o uso excessivo de bibliotecas prontas que promovem facilidades computacionais, e, de outro, a Programação passo a passo como estratégia para o desenvolvimento de competências computacionais e matemáticas. Ao analisar como cada abordagem impacta o raciocínio lógico-computacional e a aprendizagem de conceitos matemáticos, o estudo oferece subsídios teóricos e práticos para práticas pedagógicas mais conscientes. Para ilustrar os efeitos dessas metodologias, são apresentados dois exemplos matemáticos, com comentários que evidenciam os impactos positivos e negativos de ambas as estratégias. Assim, o artigo fortalece o debate sobre práticas pedagógicas que desenvolvam, de fato, o pensamento computacional e matemático de forma integrada e crítica, oferecendo subsídios teóricos e práticos para professores e formadores que atuam na Educação Básica e na formação docente.

Espera-se que o presente artigo forneça contribuições significativas à área e colabore para a comunidade acadêmica e científica, no sentido de possibilitar lacunas para discussões cada vez mais amplas sobre a necessidade de inserção da linguagem de Programação como

ferramenta de apoio no ensino de Matemática, criando um ambiente mais engajador e motivador, que fomente características como a participação e o auxílio dos estudantes, alinhando-se às metodologias ativas de ensino.

2 Referencial Teórico

A inserção da linguagem de Programação no ensino básico ainda é, em muitos países, uma realidade voltada ao contexto extracurricular (Geldreich & Hubwieser, 2020). No entanto, essa situação, ao longo dos anos, tem mudado de forma gradativa, uma vez que as discussões na comunidade acadêmica têm revelado o potencial da computação como ferramenta auxiliar para o desenvolvimento do pensamento analítico, crítico e criativo (Papert, 1980). Além de fomentar habilidades de resolução de problemas e colaboração, a computação também promove a construção do mundo digital, pensamento computacional e a cultura digital (Siqueira, 2022).

Dada a contribuição na construção de habilidades e competências para área, países como o Reino Unido (Brown *et al.* 2014), Austrália (Falkner *et al.* 2014), Finlândia (Kwon & Schroderus, 2017), Japão (Lv, Yang & Zhang, 2022) têm dado passos significativos nessa direção, incluindo conteúdos de ciências da computação em seus currículos escolares. No Brasil, esse movimento tem sido acompanhado por iniciativas do Ministério da Educação (MEC), que busca incorporar essas inovações ao currículo escolar desde a base educacional. O parecer CNE/CEB n° 2/2022, recentemente homologado, representa um marco nesse processo, propondo uma resolução para o ensino de computação na educação básica, com a promulgação da Lei n° 14.533/2023, que atualiza a Lei de Diretrizes e Bases da Educação Nacional (LDB) ao incluir a obrigatoriedade da educação digital no currículo básico.

Essa inserção ainda caminha de forma inicial, e, portanto, não está muito claro até que ponto e de que forma o ensino de Computação e Programação podem e devem ser introduzidos no ensino básico, no currículo das escolas brasileiras. Mesmo neste patamar ainda incerto, amplas discussões vêm sendo defendidas para inserção da Programação, como Lovatti *et al*. (2017), Wing (2006) e Geldreich e Hubwieser (2020). Ao que se indica, até o momento, uma proposta curricular de referência em tecnologia e computação em complemento à BNCC para educação infantil e ensino fundamental foi apresentada por Raabe, Brackmann e Campos (2018). Assim como, para o ensino médio, por meio do Centro de Inovação para a Educação Brasileira (Campos & Dias, 2020). Enquanto, em 2022, foi apresentada uma proposta de normas sobre computação na educação básica como complemento à BNCC (Siqueira, 2022).

Essas mudanças possibilitam que os estudantes deixem de ser meros usuários passivos de tecnologia para se tornarem criadores no mundo digital (Garneli, Giannakos & Chorianopoulos, 2015). Ao assumir um papel ativo, os estudantes podem desenvolver, inovar e contribuir de forma significativa para a cultura digital (Resnick *et al*., 2009), tornando-se protagonistas no processo de construção do conhecimento. Esse enfoque no protagonismo discente reflete os princípios das metodologias ativas, que deslocam o foco do docente como principal transmissor do saber para a função de facilitador (Weisz, 2004) e (Bacich & Moran, 2018). Nesse contexto, o estudante é incentivado a participar ativamente do processo de ensino e aprendizagem, desenvolvendo autonomia, pensamento crítico e capacidade de raciocínio para resolver problemas propostos (Perim *et al.,* 2023) e (Pinto *et al.,* 2020).

Com base na experiência dos autores, acredita-se que as metodologias ativas oferecem suporte significativo ao desenvolvimento do pensamento computacional, alinhando-se às competências promovidas pelo pensamento matemático. Habilidades como resolução de problemas, raciocínio algébrico e pensamento algorítmico são comuns às abordagens apresentadas, reforçando a interseção entre Programação e Matemática. Dessa forma, a

obrigatoriedade da educação digital no currículo básico brasileiro fortalece essa conexão, ao estabelecer competências que interligam essas áreas de conhecimento. O pensamento computacional, contribui para a construção de habilidades analíticas essenciais à Matemática, como apontado por Wing (2006). Essa relação promove a ideia de que aprender a programar pode potencializar o aprendizado de conceitos matemáticos, conforme discutido por Morais, Basso e Fagundes (2017). Além disso, conforme Papert (1980), a utilização de computadores e a prática da Programação permitem que os alunos compreendam conceitos matemáticos de forma mais intuitiva.

A relação entre Programação e Matemática tem cada vez mais aumentado as discussões na comunidade acadêmica por meio de propostas de trabalhos em pós-graduações como Junior (2021), Navarro (2021) e Azevedo (2022). Diversos trabalhos têm explorado metodologias que integram o ensino matemático com o desenvolvimento de competências por intermédio da Programação. Em Giraldo, Mattos e Caetano (2012), os autores propõem o *design* de situações didáticas como uma abordagem para incorporar a Programação em sala de aula, facilitando a aprendizagem de conceitos matemáticos. De forma semelhante, Souza (2016) apresenta sequências didáticas voltadas ao ensino de tópicos específicos de Programação, pontuando habilidades matemáticas, enquanto Morais, Basso e Fagundes (2017) discutem a aplicação da Programação para ensinar operações matemáticas básicas, ampliando o engajamento e a compreensão dos alunos.

Alguns dos artigos citados utilizaram a linguagem de Programação *Python*. Para o ensino de Matemática, algumas bibliotecas do *Python* destacam-se por seu foco em cálculos e operações matemáticas, como *NumPy*, *SymPy* e *SciPy*. Alguns trabalhos, como Harris *et al*. (2020), exploram o potencial do *NumPy* para facilitar a compreensão de conceitos matemáticos, evidenciando sua eficiência principalmente na realização de operações matriciais. De forma semelhante, Meurer *et al.* (2017) destacaram a importância do uso do *SymPy* na álgebra simbólica, resolvendo problemas envolvendo derivadas, integrais e equações diferenciais. Já Virtanen *et al.* (2020) discutiram aplicações do *SciPy*, demonstrando como suas ferramentas avançadas podem ser empregadas no ensino de Matemática aplicada, incluindo resolução de problemas de otimização e análise de dados.

Do ponto de vista computacional e até mesmo matemático, essas bibliotecas são altamente eficientes, oferecendo comandos simples e diretos para cálculos complexos e manipulação de dados. No entanto, sob uma perspectiva pedagógica, especialmente para iniciantes na aprendizagem de Programação, o uso excessivo de funções predefinidas pode limitar a compreensão dos fundamentos tanto da Programação quanto da Matemática. Esse excesso pode restringir o desenvolvimento de habilidades essenciais, como o raciocínio lógico e a capacidade de construir algoritmos do zero.

Depender apenas de funções prontas pode acarretar em dificuldades para evoluir em conceitos na área, além de poder dar uma falsa sensação de domínio, uma vez que o estudante não precisa compreender os fundamentos por trás das operações realizadas. Isso se torna evidente em situações práticas, como no uso da biblioteca *NumPy* para resolver problemas matemáticos. Por exemplo, ao implementar um algoritmo para multiplicar duas matrizes (A) e (B), o estudante pode simplesmente utilizar o comando *numpy.dot(A, B)* para obter o resultado imediato, sem necessariamente entender a lógica da multiplicação matricial. Outro exemplo é o cálculo das raízes de polinômios. No caso de um polinômio de segundo grau, basta usar *numpy.roots([a, b, c]),* fornecendo uma lista com os coeficientes a, b e c. Isso demonstra que o estudante não precisaria, necessariamente, compreender como calcula o valor de delta nem os três casos que determinam a existência e a natureza das raízes segundo a fórmula de *Bhaskara*.

Uma analogia pertinente, embora não usual na literatura, defendida aqui, pode ser traçada entre o uso excessivo de funções computacionais prontas por estudantes iniciantes em linguagens de Programação e a resolução de questões de múltipla escolha em Matemática. Em ambas as situações, os estudantes têm duas possíveis abordagens: a abordagem superficial, em que, diante da falta de compreensão ou domínio do conteúdo, podem recorrer a tentativas aleatórias "chutes", que, embora eventualmente levem à resposta correta, não promovem o entendimento real dos conceitos envolvidos. Essa problemática é amplamente discutida na literatura, incluindo estudos como os de Burton *et al.* (1991) e outros que abordam as limitações das questões de múltipla escolha e as práticas pedagógicas associadas, como Nicol (2007).

Na Programação, o uso frequente de funções prontas pode dificultar a compreensão dos processos internos e da lógica necessária para a criação de algoritmos (Linge & Langtangen, 2020). A segunda possibilidade, de forma semelhante, se for feita uma analogia com a Matemática, corresponderia a responder questões de múltipla escolha sem uma análise exploratória das habilidades envolvidas, confiando apenas na sorte ou na identificação de padrões, podendo assim limitar o desenvolvimento do raciocínio lógico e crítico que essas atividades buscam estimular (Nicol, 2007). Ambas as práticas podem induzir uma falsa percepção de competência, comprometendo o desenvolvimento das habilidades fundamentais exigidas para a resolução de problemas de forma autônoma e criativa.

Por outro lado, há ainda a abordagem exploratória, que exige que os estudantes percorreram cada etapa do processo lógico até alcançar o resultado de um algoritmo computacional. Essa metodologia, promove um aprendizado mais qualificado e significativo, ao conectar Programação e Matemática, desenvolvendo habilidades cruciais como lógica matemática computacional, raciocínio algorítmico e resolução de problemas (Stephens, 2018). Além disso, ao construir algoritmos passo a passo, os estudantes fortalecem o pensamento lógico e analítico, que facilita a transferência de conhecimento para novos contextos e desafios. Essa prática não apenas fomenta a autonomia no aprendizado, mas também estimula uma compreensão mais abrangente dos conceitos abordados.

3 Aplicação interdisciplinar entre Matemática e Programação

Nesta seção, é explorada a dualidade das abordagens apresentadas no referencial teórico (superficial e exploratória) aplicadas à resolução de problemas matemáticos utilizando a linguagem de Programação *Python*. O objetivo é demonstrar diferentes estratégias para construir algoritmos computacionais com base em dois problemas matemáticos relevantes à educação básica. É importante salientar que os métodos apresentados não esgotam a totalidade das possibilidades de resolução. As etapas podem ser ajustadas, simplificadas ou reformuladas de diferentes maneiras, dependendo do nível de conhecimento em Matemática e Programação que cada pessoa possua.

3.1 Abordagens com Equação do Segundo Grau

Problemas polinomiais de segundo grau integram um componente obrigatório no currículo de Matemática no ensino básico brasileiro. São abordados com o objetivo de desenvolver habilidades analíticas, lógicas e aumentar a capacidade para resolver problemas de forma estruturada. Conforme a BNCC, essa temática está diretamente associada a várias habilidades, como a EF08MA09, que propõe para os estudantes a capacidade de "Resolver e elaborar, com e sem uso de tecnologias, problemas que possam ser representados por Equações Polinomiais de 2º grau" ou também a EM13MAT302, que incentiva os estudantes a "Construírem modelos empregando as Funções Polinomiais de 1º ou 2º graus, para resolver

problemas em contextos diversos". Além de outras habilidades como EF09MA09, EM13MAT402 e EM13MAT502.

Contextualização do Problema Proposto: Considere que uma empresa de tecnologia especializada na produção de fones de ouvido compra cada unidade básica por R$ 1,00 e, além disso, incorre em um custo adicional de R$ 6,00 por unidade para acabamento e distribuição, totalizando um custo de R$ 7,00 por unidade. Inicialmente, a empresa revende cada fone por R$ 15,00. No entanto, para incentivar grandes vendas, a empresa oferece um desconto progressivo: para cada unidade vendida, além das 100 iniciais, o preço de venda por unidade diminui R$ 0,01. A empresa também tem um custo fixo mensal de R$ 200,00, que cobre despesas como aluguel e energia.

Diante desses custos e da estratégia de descontos progressivos, o empresário percebeu a necessidade de otimizar suas vendas para evitar prejuízos e garantir o maior retorno possível. Para isso, contratou um matemático que, por meio de uma análise detalhada, poderia determinar a quantidade ideal de fones de ouvido a serem vendidos para maximizar o lucro da empresa.

Formalização Matemática: O problema descrito pode ser modelado matematicamente por meio das seguintes variáveis e funções: seja x a quantidade de unidades de fones de ouvido vendidas pela empresa. O preço de venda por unidade, p(x), é dado por:

$$p(x) = \begin{cases} 15, & \text{se } x \leq 100 \\ 15 - 0{,}01\,(x - 100), & \text{se } x > 100 \end{cases}$$

A receita total R(x) é o produto do preço por unidade p(x) e o número de unidades vendidas x, isto é R(x) = p(x)x. Portanto, para $x \leq 100$, a receita total será R(x) = 15x. Para $x > 100$, a receita total pode ser expressa como:

$$R(x) = (16 - 0{,}01x)x = 16x - 0{,}01\,x^2.$$

O custo total C(x) é composto por um custo fixo mensal de R$ 200,00, que cobre despesas gerais da empresa, e um custo variável de R$ 7,00 por unidade produzida, que inclui o custo de compra (R$ 1,00) e o custo de produção (R$ 6,00). Assim, a função custo total é dada por C(x) = 200 + 7x. Finalmente, o lucro L(x) é dado pela diferença entre a receita total R(x) e o custo total C(x), L(x) = R(x) – C(x). Para $x \leq 100$, o lucro é:

$$L(x)=15x - (200 + 7x) = 8x - 200.$$

Para $x > 100$, o lucro é:

$$L(x) = (16x - 0{,}01x^2) - (200 + 7x) = -0{,}01x^2 + 9x - 200.$$

Para maximizar o lucro, basta observar que a função de lucro é crescente para $x \leq 100$, e o maior lucro é obtido quando a empresa vende 100 unidades, resultando em um lucro de R$ 600. Agora, vamos analisar como o lucro comporta-se, quando o número de unidades vendidas ultrapassa 100, com a função de lucro dada por $L(x) = -0{,}01x^2 + 9x - 200$. Essa é uma função quadrática (parabólica), em que o coeficiente de x^2 é negativo (−0,01), o que indica que a parábola abre para baixo e, portanto, o vértice representará o ponto de máximo lucro. O vértice de uma parábola $ax^2 + bx + c$ ocorre na coordenada x_v, dada por $x_v = -b / 2a$.

Neste caso, temos $x_v = 450$. Portanto, o ponto de maximização do lucro ocorre quando a empresa vende 450 unidades. Esse é o número de unidades que maximiza o lucro. Agora, para verificar o valor máximo do lucro, basta substituir x = 450 na função de lucro L(x), ou seja, L(450) = 1825,00. Logo, o lucro máximo é R$ 1825,00.

Algoritmo Computacional (Forma Direta): Ao utilizar a biblioteca *NumPy*, a resolução de uma equação do segundo grau torna-se um processo simples e direto. Com o comando roots, é possível calcular as raízes da equação de maneira rápida e eficiente, como demonstrado no código do Quadro 1.

Quadro 1: Código para solução de um polinômio do segundo grau usando a função `roots`.

```
In [1]:  from numpy import roots
In [2]:  coeficinetes = [-0,01, 9, -200]
In [3]:  raiz = roots(coeficinetes)
In [4]:  print("As raízes são", raiz)
```

Fonte: Elaborado pelos autores.

Esse método otimiza o processo computacional ao acelerar o cálculo e apresentar os resultados quase instantaneamente. Na primeira linha do código, a função roots é importada diretamente da biblioteca *Numpy*, permitindo apresentar as raízes do Polinômio do segundo grau. Na segunda linha, os coeficientes da equação a = -0,01, b = 9 e c = -200 são fornecidos como entrada. Em seguida, na terceira linha, a função roots realiza o cálculo e aloca o(s) valor(es) de "x" em uma variável chamada raiz, que é, por fim, impresso na linha 4. No entanto, note que, ao utilizar a função pronta roots, o estudante é privado da oportunidade de compreender e aplicar o raciocínio lógico por trás de cada etapa do processo de resolução, o que limita o seu entendimento do problema e dos passos matemáticos envolvidos.

Embora essa abordagem seja eficiente para obter a solução final do problema, ela apresenta desvantagens significativas no que diz respeito à avaliação do raciocínio lógico e à compreensão dos conceitos matemáticos envolvidos. Estudos, como Sartori e Duarte (2021), destacam que a ausência de uma análise do processo de resolução pode levar os estudantes a focarem mais na memorização do que no entendimento do problema. Para este caso em específico, é possível considerar que os estudantes podem apenas memorizar que qualquer tipo de Polinômio pode ser resolvido com a simples chamada da função roots, ao passar os coeficientes do Polinômio que deseja resolver.

Na Matemática, esses tipos de problemas que podem permitir a memorização e chutes de respostas, podem estar diretamente ligados a atividades que compõem questões fechadas de múltipla escolha. Conforme apontado por Sartori e Duarte (2021), problemas matemáticos estruturados em formato fechado, como questões de múltipla escolha, frequentemente não refletem adequadamente o raciocínio empregado na solução. Isso pode levar os alunos a recorrerem ao acaso na escolha de respostas e a priorizarem o resultado, desconsiderando a importância do processo reflexivo.

Assim, no contexto interdisciplinar da Matemática e da Programação, utilizar apenas algoritmos prontos para resolver problemas pode comprometer o desenvolvimento dos estudantes na compreensão dos fundamentos essenciais. Esse mecanismo tende a gerar uma falsa percepção de aprendizado, tanto em relação aos conceitos matemáticos quanto à lógica e à escrita de códigos. Sem a exploração ativa dos passos de cada etapa da construção do algoritmo, os estudantes correm o risco de se tornarem meros executores de soluções automatizadas (Greefrath, Hertleif & Siller, 2018), perdendo a oportunidade de construir um raciocínio crítico e criativo (Perim *et al.,* 2023; Costa & Gontijo, 2024).

Algoritmo Computacional (Passo a Passo): Por outro lado, quando construído esse algoritmo computacional, seguindo uma sequência lógica e estruturada, o discente contempla a possibilidade de desenvolver competências e habilidades importantes, como as recomendadas pela BNCC. Por exemplo, a habilidade EF06MA23 incentiva a construção de algoritmos para resolver situações passo a passo. Além disso, a habilidade EF07MA05 propõe resolver um

mesmo problema utilizando diferentes algoritmos, promovendo a flexibilidade no pensamento e a capacidade de abordar uma questão por múltiplas perspectivas. Essas estão diretamente conectadas ao pensamento matemático computacional, uma vez que se estimula a decompor um problema e a criar uma solução que executa um conjunto de operações de forma ordenada.

A solução para esse problema é apresentada, primeiramente, por meio de um pseudocódigo que descreve a lógica do processo de forma abstrata, e, em seguida, com o código implementado em *Python* utilizando a biblioteca *Math* (apenas para calcular a raiz quadrada). No Quadro 02, são exibidas ambas as soluções, destacando como a estrutura lógica se traduz em instruções de Programação. O pseudocódigo serve como uma representação simplificada, ideal para entender a maneira conceitual, enquanto o algoritmo em *Python* mostra a aplicação prática desse raciocínio em um ambiente computacional.

Quadro 2: Código para solução de um polinômio do segundo grau passo a passo.

Pseudocódigo	Algoritmo Computacional
1.Biblioteca math	In [1]: import math
2.Início	In [2]: a = -0,01
3.Coeficientes a, b, c	In [3]: b = 9
4.Verifique:	In [4]: c = -200
5.Se a = 0:	
6. Não é função 2° grau	In [5]: if a == 0:
7.Fim do Se	In [6]: print("não é de segundo grau")
8.Senão:	In [7]: else:
9. Calcule delta	In [8]: delta = b**2 - 4 * a * c
10.Verifique:	In [9]: print("Delta é: ", delta)
11.Se delta < 0:	In [10]: if delta < 0:
12. Raiz complexa	In [11]: print("Raiz complexa")
13.Fim do Se	In [12]: elif delta == 0:
14.Senão delta = 0:	In [13]: x1= -b / 2*a
15. Calcule a raiz	In [14]: print("Raiz: ", x1)
16.Fim do Senão então	In [15]: else:
17.Senão:	In [16]: x1=(-b + math.sqrt(delta))/2*a
18. Calcule duas raízes	In [17]: x2=(-b - math.sqrt(delta))/2*a
19.Fim do Senão	In [18]: print("Raízes: ", x1, " ", x2)

Fonte: Elaborado pelos autores.

No pseudocódigo, o estudante é primeiramente incentivado a pensar de maneira ampla sobre como construir o algoritmo. O uso do pseudocódigo tem como objetivo ajudar o aluno a visualizar a organização computacional do programa e a estabelecer as conexões com a lógica matemática envolvida. Dessa forma, o estudante não apenas compreende a funcionalidade do programa, mas também percebe a importância da lógica estrutural, necessária para que o algoritmo execute de forma correta.

Agora, observe que, no algoritmo computacional, na linha 5, é inserido um comando condicional `if`, que conduz o discente a compreender que uma Função Polinomial do segundo grau só existe se, e somente se, o valor do coeficiente angular *a* for diferente de zero. Caso contrário, a equação resultará, no máximo, em uma Função Polinomial Linear. Assim, ao fornecer um valor para *a* diferente de zero, o algoritmo continua sua execução no bloco correspondente ao comando condicional `else`, localizado entre as linhas 7 e 18.

Assim, ao lembrar conceitos matemáticos sobre o tema, sabe-se que para calcular as raízes de uma função do segundo grau, o primeiro passo é determinar o valor de delta (linha 8).

Após esta etapa, o algoritmo pode seguir por três 3 caminhos diferentes, definidos por uma estrutura de condicionais aninhadas (linhas 10, se delta for menor que zero, linha 12 se delta for igual a zero e linha 15 caso contrário, ou seja, se delta for maior que zero). Isso imposto, é possível que os estudantes possam, de forma clara e objetiva, entender a razão por que, computacionalmente, é preciso utilizar a implementação destas três condicionais, uma vez que são as necessidades impostas pelo critério matemático que determinam a possibilidade de alcançar de forma correta as raízes de um polinômio específico.

Assim, nesse caso, a exploração dos passos sequenciais possibilita a construção de conhecimentos de maneira robusta e precisa, refletindo abordagens amplamente discutidas em teorias educacionais e científicas, levando ao conceito de aprendizado, ou seja, quando os indivíduos exploram perspectivas diversas e constroem ativamente seu próprio entendimento, rompendo com modelos passivos de ensino. Sobre isso, Papert (1980) enfatiza que a exploração ativa e a criação de múltiplos caminhos são fundamentais para um aprendizado significativo, especialmente no campo da computação educacional.

É possível, por meio da sequência de estrutura condicional aninhada, reforçar como os conceitos matemáticos podem ser integrados ao pensamento computacional, relacionando a diversos problemas práticos. A contextualização é um fator importante no ensino aprendizado, uma vez que possibilita o estudante enxergar a Matemática em diferentes contextos do cotidiano, como mostrado por Araújo *et al*. (2024) e Matos *et al*. (2024). Neste caso, a estrutura condicional aninhada assemelha-se à ideia de funções definidas por partes, utilizada na Matemática para modelar situações em que uma função assume diferentes comportamentos dependendo de um intervalo ou condições específicas. A seguir, é ilustrado um exemplo.

Exemplo: Em um sistema de tarifação de energia elétrica, o custo mensal de uma residência é calculado com base no consumo (em kWh) dividido em faixas de consumo. As regras são as seguintes:

- *Até 100 kWh: Tarifa mínima de R$ 80.*
- *De 101 kWh a 200 kWh: R$ 0,60 por kWh adicional.*
- *Acima de 200 kWh: R$ 1,50 por kWh adicional.*

Ao determinar uma função que represente essa situação, é possível escrever:

$$f(x) = \begin{cases} 80, & \text{se } 0 \leq x \leq 100 \\ 80 + 0{,}60(x - 100), & \text{se } 101 \leq x \leq 200 \\ 80 + 0{,}60 \cdot 100 + 1{,}50(x - 200), & \text{se } x > 200 \end{cases}$$

sendo *x* o valor de consumo em *(em kWh)*.

Neste contexto, é apresentada uma função definida por partes, em que cada intervalo ou condição pode ser descrito por uma estrutura condicional em linguagem de Programação. Assim, as três condições matemáticas são traduzidas para comandos condicionais de forma lógica: `if x <= 100:` para a primeira faixa, `elif 100 < x and x <= 200:` para a segunda, e, por fim, um bloco `else:` que cobre os casos em que `x > 200`. Essa transcrição evidencia como conceitos matemáticos podem ser diretamente mapeados para estruturas computacionais, facilitando tanto a compreensão quanto a implementação prática.

Portanto, é possível afirmar que a Programação e Matemática estão totalmente conectadas, especialmente no contexto dos problemas apresentados. Os algoritmos computacionais são capazes de gerar resultados corretos e eficientes justamente porque se

basearam em princípios matemáticos, como operações lógicas e o uso estruturado de controles condicionais. Por isso, na prática da Programação, a ausência de precisão matemática pode ocasionar grandes consequências, como resultados incorretos. Portanto, a precisão matemática desempenha um papel fundamental, garantindo não apenas a robustez dos cálculos, mas também a confiabilidade dos algoritmos desenvolvidos.

3.2 Abordagens com Matrizes

Um outro problema que pode ser mostrado para relacionar a conexão entre Matemática e Programação está no estudo de Matrizes e das Operações básicas entre si. Embora a BNCC não mencione explicitamente o ensino de Vetores e Matrizes como conteúdos obrigatórios, é possível conectar esses conceitos a algumas habilidades descritas no documento. Na habilidade EM13MAT203, destaca-se a aplicação de conceitos matemáticos na construção e análise de planilhas voltadas para controle de orçamento familiar, simuladores e outras finalidades práticas. Além disso, habilidades semelhantes são propostas no ensino fundamental I e II, em que o uso de planilhas eletrônicas ou alternativas são sugeridas. Assim, relaciona-se um problema contextualizado do tipo:

Contextualização do Problema Proposto: Considere um estudante que cursou três disciplinas: História, Matemática e Geografia, em três unidades letivas. Suas notas finais para cada unidade letiva ficaram na forma como apresentado no Quadro 3.

Quadro 3: Notas obtidas por um estudante em três disciplinas.

Unidade / Disciplina	I	II	III
História	7,50	6,90	7,10
Matemática	6,00	6,40	8,50
Geografia	8,30	7,50	9,20

Fonte: Elaborado pelos autores.

A professora de Matemática deste estudante, ao analisar as informações do Quadro 3, identificou uma oportunidade para apresentar os conceitos de matrizes. Ela o incentivou ao destacar que esses conceitos surgem naturalmente na resolução de diversos problemas e são essenciais, principalmente por organizarem e simplificarem os cálculos. Além disso, essa abordagem permite explorar, de forma prática, como as matrizes podem representar situações reais e como operações matemáticas podem ser aplicadas na análise de dados, facilitando a conexão entre o pensamento computacional e o ensino de Matemática. Essa abordagem permite explorar, de maneira prática, como as matrizes podem representar situações reais e como operações matemáticas podem ser aplicadas para análise de dados, facilitando a conexão entre o pensamento computacional e o ensino de Matemática.

Formalização Matemática: Com base no Quadro 3, a tabela de elementos organizados em linhas e colunas pode ser representada por uma matriz, denominada aqui de matriz M. Assim, seja M uma matriz 3×3, em que cada elemento m_{ij} corresponde à nota da disciplina i na unidade j, considerando as três disciplinas (História, Matemática e Geografia) e as três unidades letivas (I, II, III). Assim

$$M = \begin{pmatrix} 7{,}5 & 6{,}9 & 7{,}1 \\ 6{,}0 & 6{,}4 & 8{,}2 \\ 8{,}3 & 7{,}7 & 9{,}2 \end{pmatrix}$$

A partir da matriz M, é possível aplicar diversas operações matemáticas, conforme apresentado a seguir.

A soma de todos os elementos da matriz M é dada por $\sum_{i,j=1}^{3} m_{ij} = 67{,}3$. A soma das notas de História, que correspondem à primeira linha da matriz é $\sum_{j=1}^{3} m_{1j} = 21{,}5$. A soma das notas da unidade I, representadas pela primeira coluna da matriz, é $\sum_{i=1}^{3} m_{i1} = 21{,}8$. Por fim, a soma das notas da diagonal principal é $\sum_{i=1}^{3} m_{ii} = 23{,}1$. Essa formalização matemática ilustra como operações simples podem ser aplicadas a matrizes, permitindo ao aluno compreender os conceitos envolvidos.

Algoritmo Computacional (Forma Direta): Ao abordar um conteúdo desse tipo, é possível explorar diversas análises matemáticas por meio da Programação. No caso de matrizes, a utilização da biblioteca *NumPy* permite a manipulação eficiente de *arrays* (vetores e matrizes). Essa ferramenta disponibiliza diversos comandos prontos, que facilitam operações matemáticas. Contudo, assim como no exemplo anterior, o uso direto desses recursos pode limitar o desenvolvimento das habilidades matemáticas dos estudantes, caso não seja acompanhado de uma abordagem contextualizada e reflexiva. No Quadro 4, são mostradas algumas atividades que podem ser relacionadas neste contexto.

Quadro 4: Propostas de atividades com matriz do Quadro 3 e implementação computacional.

a) Apresente a soma de todos as notas da matriz	`In [1]: import numpy as np`
	`In [2]: M = np.array([[7.5, 6.9, 7.1],`
	`[6.0, 6.4, 8.2],`
	`[8.3, 7.7, 9.2]])`
	`In [3]: print("Soma todas notas:", np.sum(M))`
	`Out[3]: 67.3`
	`In [4]: print("História:", np.sum(M[0]))`
	`Out[4]: 21.5`
	`In [5]: print("Unidade I:", np.sum(M[:,0]))`
	`Out[5]: 21.8`
	`In [6]: print("D.Prin.:", np.sum(np.diag(M)))`
	`Out[6]: 23.1`

b) Apresente a soma das notas de história das três unidades	
c) Apresente a soma de todas as notas da unidade I	
d) Apresente a soma das notas da diagonal principal	

Fonte: Elaborado pelos autores.

No Quadro 4, nas linhas 2 a 4, é construída a matriz `M`, por meio do comando *array*, que utiliza a biblioteca *NumPy* importada na linha 1. No lado esquerdo do quatro, atividades são propostas de a) a d). Observa-se que cada questão é rapidamente resolvida com chamadas relativamente simples utilizando o comando *sum*, que realiza operações de soma. Para a soma de todas as notas da matriz, basta utilizar `np.sum(M)`. Além disso, para somar os elementos de uma linha específica, como a primeira linha (notas de História), usa-*se* `np.sum(M[0])`. Para somar outras linhas, substitui-se `[0]` por `[1]` (Matemática) ou `[2]` (Geografia). Para somar os elementos de uma coluna específica, altera-se a forma de indexação da matriz. Por exemplo, `np.sum(M[:,0])` soma os elementos da primeira coluna (unidade I), enquanto `[:,1]` corresponde à unidade II e `[:,2]` à unidade III. E, por fim, para a soma de todas as notas da diagonal principal, basta escrever `np.sum(np.diag(M))`. Essa abordagem

demonstra a praticidade da biblioteca *NumPy* para realizar cálculos matemáticos de forma eficiente e rápida.

O objetivo aqui não é ensinar linguagem de Programação com essa breve seção, mas sim, discutir que, do ponto de vista computacional, existe uma eficiência das rotinas prontas disponíveis para resolver este tipo de problema. No entanto, sob a ótica Matemática, essas abordagens podem dificultar para o estudante a compreensão das estruturas lógicas, como o processo utilizado para somar os valores em linhas, colunas e diagonais.

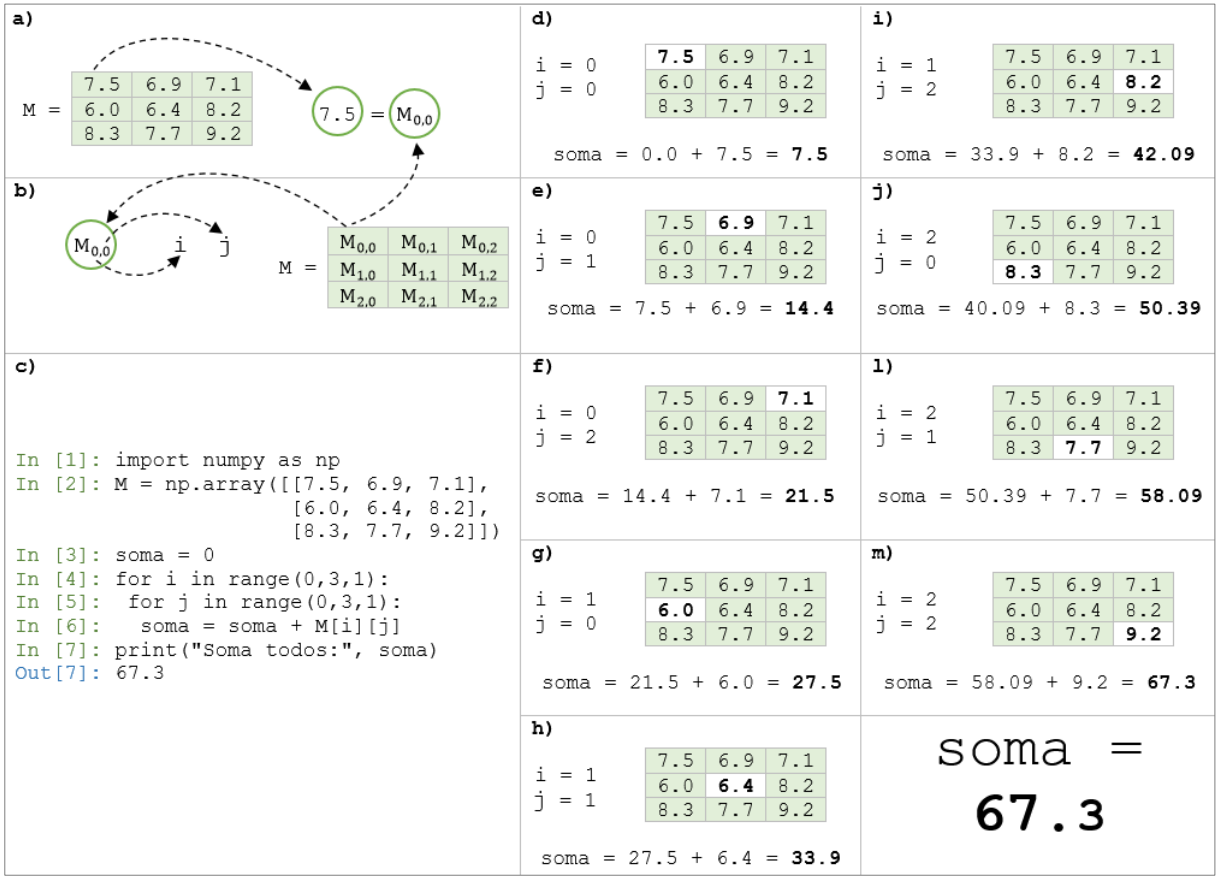
Algoritmo Computacional (Passo a Passo): Enfrentar este problema seguindo uma abordagem que construa, passo a passo, cada etapa da sequência lógica estrutural, proporciona ao estudante uma oportunidade de compreender os processos internos envolvidos na criação de um algoritmo computacional. Essa metodologia vai além da simples aplicação de comandos prontos, incentivando uma reflexão a mais sobre como as estruturas analíticas matemáticas estão diretamente conectadas à construção de algoritmos.

Para tornar clara a lógica por trás dessa rotina computacional, a Figura 1 detalha, passo a passo, o que ocorre durante a execução do código para somar todas as notas da matriz. Esse detalhamento ajuda a visualizar como os índices `i` e `j` evoluem ao longo das iterações. Essa breve apresentação ajuda a desenvolver habilidades fundamentais necessárias para que os estudantes compreendam e desenvolvam o algoritmo relacionado a este problema.

O primeiro passo nessa sequência é garantir que os estudantes compreendam como uma matriz matemática convencional, Figura 1 a), é relacionada com a matriz matemática com índices, Figura 1 b). Essa compreensão inclui a noção de que cada elemento da matriz está associado a um par de índices que identifica sua posição dentro da estrutura. Por exemplo, nessa matriz 3×3, os índices (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1) e (2,2) correspondem às posições específicas de seus elementos. Cada elemento dessa matriz está associado a um par de índices que variam de 0 a 2, permitindo sua representação na estrutura `M[i][j]`, em que `i`, `j` assumem os valores 0, 1, 2. Essa notação torna clara a organização dos elementos da matriz em linhas e colunas, que será utilizada para realizar operações matemáticas.

Partindo do pressuposto de que o estudante já possui conhecimento prévio sobre a estrutura de repetição `for`, é possível estabelecer a relação entre os índices `i` e `j` e seu comportamento dentro do laço de repetição. Essa conexão permite que os estudantes compreendam como os índices percorrem cada elemento da matriz, linha por linha e coluna por coluna, facilitando a execução solicitada. Na Figura 1 c), apresenta-se uma opção de implementação para somar todos os elementos da matriz utilizando a estrutura de repetição `for`. Esse exemplo ilustra como os índices `i` e `j` percorrem a matriz, acessando cada elemento individualmente, enquanto acumulam os valores em uma variável de soma. Essa abordagem permite explorar o conceito de indexação, do ponto de vista da linguagem computacional, além de permitir que o estudante entenda a lógica que permite somar todas as notas.

Figura 1: Etapas para realização da soma dos elementos de uma matriz 3 x 3.



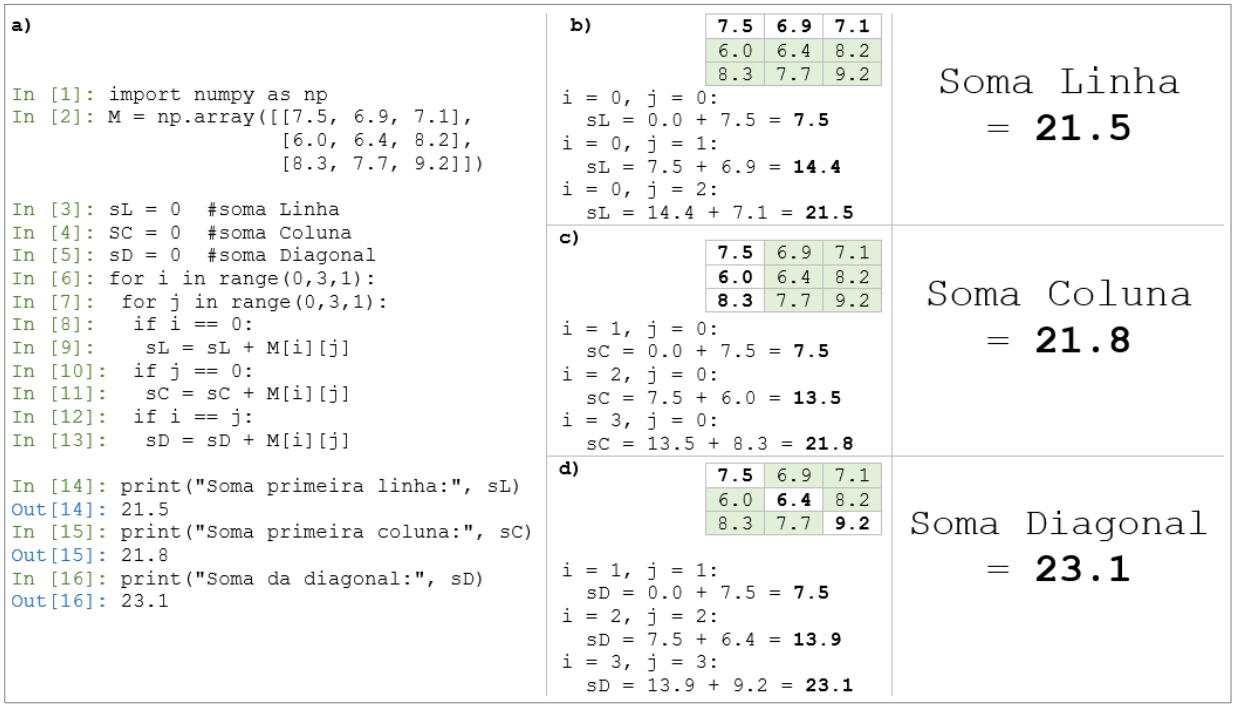
Fonte: Elaborado pelos autores.

Observe, na Figura 1 c), que a variável `soma` (linha 3) é inicializada fora do *loop*, pois seu valor precisa ser definido como zero apenas uma vez antes do início das iterações. Em seguida, os *loops* aninhados são iniciados: o índice `i` representa as linhas da matriz (linha 4), enquanto o índice `j` representa as colunas correspondentes (linha 5). Dentro desses *loops*, o processo de soma é realizado iterativamente, elemento por elemento, até que todos os valores da matriz tenham sido analisados. Esse mecanismo garante que cada elemento seja acessado e incluído na soma total. O processo somente é concluído após todas as combinações de `i` e `j` serem exploradas. A Figura 1 d) a m) mostra, passo a passo, o que ocorre internamente durante a execução do código para somar todas as notas da matriz. Esse detalhamento ajuda a visualizar como os índices `i` e `j` evoluem ao longo das iterações, evidenciando o comportamento do algoritmo e como a matriz é processada.

Para resolver as questões b), c) e d) propostas no Quadro 4, é fundamental identificar conexões e padrões existentes na matriz de índices como mostrado na Figura 1 b). Por exemplo, ao abordar o problema de somar todos os valores da linha 1, correspondente às notas de história, o estudante pode ser instigado com a seguinte pergunta: *Com base na matriz apresentada, é possível detectar algum padrão que permita acessar apenas os valores da linha 1 ao percorrer os loops*? De maneira análoga, o mesmo raciocínio pode ser aplicado à coluna 1 e à diagonal principal. Essas reflexões, que envolvem compreender como ajustar as condições do *loop* para acessar apenas os elementos desejados, incentivam o estudante a exercitar o raciocínio lógico e matemático. Esse processo, por sua vez, contribui diretamente para o desenvolvimento de habilidades fundamentais para o pensamento matemático e computacional.

A Figura 2 apresenta uma proposta de algoritmo computacional para resolução das atividades b), c) e d) propostas no Quadro 4.

Figura 2: Operações em uma matriz 3x3. Em a) soma dos elementos da primeira linha, em b) soma dos elementos da primeira coluna e em c) soma dos elementos da diagonal principal.



Fonte: Elaborado pelos autores.

Para somar todos os valores da linha 1 da matriz `M`, é necessário identificar que, nessa linha, todos os índices `i` são iguais a zero. Portanto, é possível incluir no algoritmo uma condição do tipo `if i == 0` (linha 8) ao algoritmo, garantindo que apenas os elementos da linha 1 sejam considerados na soma. De forma semelhante, para somar os valores da primeira coluna, o estudante deve observar que, para essa coluna, todos os índices `j` são iguais a zero. Consequentemente, basta adicionar uma condição `if j == 0` (linha 10). Por fim, para somar os elementos da diagonal principal, é necessário perceber que, nessa região da matriz, os índices `i` e `j`são sempre iguais. Assim, a inclusão de um condicional `if i == j` (linha 12) no algoritmo resolve o problema.

3.3 Análises Pedagógicas e Aplicações na Educação Matemática

A proposta de utilizar algoritmos em *Python* para trabalhar com equações quadráticas e operações com matrizes traz implicações significativas para o ensino e a aprendizagem da Matemática. Quando essa abordagem vai além da simples execução de códigos prontos e envolve os estudantes na construção manual dos algoritmos, o processo se torna pedagogicamente mais eficaz. Ao programar passo a passo a resolução de uma equação do segundo grau ou a multiplicação de matrizes, por exemplo, os alunos são levados a revisitar e consolidar conceitos fundamentais, como a identificação de raízes, a estrutura da função quadrática e propriedades algébricas como a distributividade, associatividade, elemento neutro, dentre outros. Essa prática favorece o desenvolvimento da argumentação Matemática e aproxima os estudantes de uma melhor compreensão dos conteúdos. Assim, essa abordagem corrobora com os princípios da Educação Matemática, pois estimula a construção ativa do conhecimento e amplia a autonomia dos alunos na resolução de problemas.

Portanto, o uso da Programação possibilita ao aluno explorar como e por que as operações funcionam. Do ponto de vista didático, essa estratégia contribui para a superação de dificuldades conceituais recorrentes no ensino da Matemática, ao oferecer uma abordagem interativa e experimental. Como apontam Ponte, Brocardo e Oliveira (2009), a resolução de problemas, com o suporte de tecnologias, permite maior envolvimento cognitivo dos estudantes e a construção de significados mais profundos sobre os objetos matemáticos. Assim, integrar Programação no ensino de Matemática favorece a interdisciplinaridade. Tal perspectiva amplia o letramento matemático (D'Ambrósio, 1996) e fortalece o protagonismo estudantil, visto que transforma a sala de aula em um espaço de investigação, criação e reconstrução de saberes matemáticos mediados por tecnologia.

Considerações Finais

Ao introduzir o ensino inicial de linguagens de Programação para estudantes do ensino básico, é essencial avaliar as metodologias adotadas para garantir que todos sejam estimulados a explorar o raciocínio lógico e matemático. Esse processo deve ser direcionado à construção de competências e habilidades tanto matemáticas quanto computacionais. Considerando que muitas linguagens de Programação oferecem um amplo conjunto de bibliotecas prontas, que facilitam e aceleram a criação de algoritmos, torna-se crucial discutir até que ponto o uso dessas funções prontas contribui para a construção de uma metodologia ativa e efetiva no processo de ensino e aprendizagem.

Essa discussão é ampla e de grande interesse para a comunidade acadêmica e científica, dado que muitos países têm reconhecido o potencial acelerado das tecnologias digitais e, consequentemente, inserido conteúdos de ciências da computação em seus currículos de educação básica. Nesse contexto, este artigo promoveu uma análise sobre os riscos metodológicos relacionados ao "facilitar o aprendizado computacional e matemático", por meio do uso excessivo de bibliotecas prontas do *Python*, em contraste com a abordagem de "desenvolver competências computacionais e matemáticas", que propõe o ensino de linguagens de Programação de forma detalhada e progressiva, passo a passo.

Para a demonstração dessas duas abordagens, foram selecionados de forma criteriosa dois problemas matemáticos, escolhidos por apresentarem uma construção algorítmica acessível e sem grandes complexidades, considerando o público-alvo composto por estudantes do ensino básico. A ideia central, ao propor esses problemas, foi evidenciar que, sem o uso da lógica Matemática e da precisão analítica, não é possível identificar corretamente onde e como os comandos computacionais devem ser aplicados. Isso ressalta a conexão entre Programação e Matemática, demonstrando como o pensamento computacional está intrinsecamente ligado ao pensamento matemático.

Assim, este texto deixa como reflexão sobre a importância de equilibrar, no ensino de linguagens de Programação, o uso de bibliotecas prontas com atividades que estimulem a construção e a compreensão dos conceitos fundamentais. Esse equilíbrio é essencial tanto para o desenvolvimento de competências computacionais quanto para o aprofundamento de conceitos matemáticos, promovendo um aprendizado coerente e eficaz.

Por fim, por se tratar de um estudo teórico, este artigo não apresenta dados empíricos que confirmem, em situações concretas de sala de aula, os efeitos das abordagens discutidas. Essa limitação abre espaço para investigações futuras que possam aplicar e avaliar, de forma sistemática, os impactos do ensino detalhado de Programação no desenvolvimento de competências matemáticas em estudantes da educação básica. Sugere-se, portanto, a realização de pesquisas empíricas com metodologias qualitativas e quantitativas, como estudos de caso,

que possibilitem testar as hipóteses levantadas neste trabalho, contribuindo com evidências práticas para o aprimoramento das metodologias de ensino de Programação no contexto da Educação Matemática.

Referências

Araújo, F. C., Paiva, C. D. C., do Nascimento, R. A., D'arc de Oliveira Passos, J., Martins, G. F. O., Sousa, M. A. D. M. A. & Alves, C. S. (2024). Uso das tecnologias digitais no ensino de matemática: um relato de experiência com aporte do GeoGebra. *Caderno Pedagógico*, *21*(8), e6715-e6715.

Azevedo, G. T. D. (2022). *Processo formativo em matemática: invenções robóticas para o Parkinson*. 2022. 213f. Tese (Doutorado em Educação Matemática). Universidade Estadual Paulista. Rio Claro, SP.

Bacich, L. & Moran, J. (2018). *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática*. Porto Alegre, RS: Penso Editora.

Brasil. (2018). Ministério da Educação. *Base Nacional Comum Curricular: Educação é a Base* (BNCC). Brasília, DF.

Brown, N. C., Sentance, S., Crick, T. & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *Association for Computing Machinery Transactions on Computing Education*, *14*(2), 1-22.

Burton, S. J., Sudweeks, R. R., Merrill, P. F. & Wood, B. (1991). *How to prepare better multiple-choice test items: Guidelines for university faculty*. Brigham Young University. Department of Instructional Science.

Campos, F. R. & Dias, R. A. (2020). Currículo de referência–Itinerário Formativo em Tecnologia e Computação. *Centro de Inovação para a Educação Brasileira CIEB*. São Paulo, SP.

Costa, I. L. & Gontijo, C. H. (2024). Avaliação Formativa e o Pensamento Crítico e Criativo em Matemática: Mapeamento de Pesquisas e Aplicações. *Paradigma*, e2024004-e2024004.

D'Ambrosio, U. (1996). *Educação Matemática: da teoria à prática*. Papirus Editora.

Falkner, K., Vivian, R. & Falkner, N. (2014). The Australian digital technologies curriculum: challenge and opportunity. In *Proceedings of the Sixteenth Australasian Computing Education Conference.* (148), 3-12.

Garneli, V., Giannakos, M. N. & Chorianopoulos, K. (2015). Computing education in K-12 schools: A review of the literature. *Institute of Electrical and Electronics Engineers Global Engineering Education Conference*. 543-551.

Geldreich, K. & Hubwieser, P. (2020). Programming in primary schools: Teaching on the edge of formal and non-formal learning. *Non-Formal and Informal Science Learning in the Information and Communication Technology Era*, 99-116.

Giraldo, V., Caetano, P. & Mattos, F. (2012). *Recursos computacionais no ensino de Matemática*. Rio de Janeiro. RJ: Sociedade Brasileira de Matemática.

Greefrath, G., Hertleif, C. & Siller, H. S. (2018). Mathematical modelling with digital tools-a quantitative study on mathematising with dynamic geometry software. *Zentralblatt für Didaktik der Mathematik*, (50), 233-244.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D. & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362.

Junior, H. N. P. (2021). *Matemática e Programação: Uma nova abordagem de ensino.* 2021. 40f. Dissertação (Mestrado em Matemática). Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, RJ.

Kwon, S. & Schroderus, K. (2017). *Coding in Schools: Comparing Integration of Programminginto Basic Education Curricula of Finland and South Korea*. Finnish Society on Media Education.

Linge, S. & Langtangen, H. P. (2020). *Programming for computations-Python: A gentle introduction to numerical simulations with Python 3.6.* p. 332. Springer Nature.

Lovatti, B. G., Vieira, L. S., Marques, K. & Scolforo, M. A. (2017). A programação no ensino básico: formando alunos para sociedade tecnológica. *Revista Ambiente Acadêmico*, *3*(1), 113-132.

Lutz, M. (2013). *Learning python: Powerful object-oriented programming*. O'Reilly Media, Inc.

Lv, W., Yang, C. & Zhang, W. (2022). Programming Education in Japanese Elementary Schools Integrated with Multiple Subjects: Origins, Practical Paths and Implications. *4th International Conference on Computer Science and Technologies in Education*. 11-21.

Matos, J. D. S. G., Paiva, C. D. C., Lima, F. F. R. R., do Nascimento, R. A., dos Santos, L. M. P., Pinheiro, A. A. & de Brito, I. E. G. (2024). A relação entre pensamento computacional e ensino de matemática no contexto da educação básica: oportunidades, desafios e perspectivas. *Caderno Pedagógico*, *21*(8), e6408-e6408.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M. & Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, *3*, e103.

Morais, A. D. D., Basso, M. V. D. A. & Fagundes, L. D. C. (2017). Educação Matemática & Ciência da Computação na escola: aprender a programar fomenta a aprendizagem de matemática? *Ciência & Educação (Bauru)*, *23*(2), 455-473.

Morán, J. (2015). Mudando a educação com metodologias ativas. *Coleção mídias contemporâneas. Convergências midiáticas, educação e cidadania: aproximações jovens*, *2*(1), 15-33.

Navarro, E. R. (2021). *O desenvolvimento do conceito de pensamento computacional na educação matemática segundo contribuições da teoria histórico-cultural*. 2021. 177f. Tese (Doutorado em Educação). Universidade Federal de São Carlos, São Carlos. SP.

Nicol, D. (2007). E-assessment by design: using multiple-choice tests to good effect. *Journal of Further and higher Education*, *31*(1), 53-64.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic books. Eugene, Oregon, United States of America: Harvester.

Perim, F. de C. R., Araújo, F. P. S., Trindade, M. C., Monteiro, M. R. A., Fragoso, N. da S. C., Batista, R. G. M., Benevides, S. R., Carvalho, V. M. & de Oliveira, F. G. (2023). Pensamento crítico na sala de aula: Capacitando alunos para o futuro em uma jornada educacional. *Revista Foco*, *16*(11), e3673.

Pinto, S. F., Ferreira, R. S., Costa, M. M. & Silva, A. M. (2020). A proposal for an active methodology for physics labs. *arXiv preprint arXiv:2005.06532*.

Ponte, J. P.; Brocardo, J.; Oliveira, H. (2009) *Investigação Matemática na Sala de Aula*. 2ª. Ed. Belo Horizonte: Autêntica.

Raabe, A. L., Brackmann, C. P. & Campos, F. R. (2018). *Currículo de referência em tecnologia e computação: da educação infantil ao ensino fundamental*. Centro de Inovação para a Educação Básica. São Paulo, SP.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the Association for Computing Machinery*, *52*(11), 60-67.

Sartori, A. S. T. & Duarte, C. G. (2021). Repetir, memorizar, recitar: mecanismos para a fabricação de corpos dóceis pela Educação Matemática. *Jornal Internacional de Estudos em Educação Matemática*, *14*(1), 84-91.

Selwyn, N. (2021). *Education and technology: Key issues and debates*. 3° Ed. London, UK: Bloomsbury Publishing.

Siqueira, I. C. P. (2022). *Normas sobre computação na educação básica–complemento à base nacional comum curricular (BNCC)*. Technical report, Conselho Nacional de Educação-Câmara de Educação Básica.

Souza, E. C. D. (2016). *Programação no ensino de matemática utilizando Processing 2: Um estudo das relações formalizadas por alunos do ensino fundamental com baixo rendimento em matemática*. 2016. 189f. Dissertação (Mestrado em Educação para a Ciência). Universidade Estadual Júlio de Mesquita Filho, Bauru, São Paulo. SP.

Stephens, M. (2018). Embedding algorithmic thinking more clearly in the mathematics curriculum. *Proceedings of the International Commission on Mathematical Instruction Study*, (*24)*, 483-490.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D. & van Mulbregt, P. (2020). Fundamental algorithms for scientific computing in python and SciPy 1.0 contributors. SciPy 1.0. *Nature Methods*, (*17)*, 261-272.

Weisz, T. (2004). *O diálogo entre o ensino e a aprendizagem*. 2° Ed. São Paulo, SP: Ática.

Wing, J. M. (2006). Computational thinking. *Communications of the Association for Computing Machinery*, *49*(3), 33-35.