UNIVERSIDADE FEDERAL DE ALAGOAS

Mestrado Profissional em Matemática em Rede Nacional

PROFMAT

DISSERTAÇÃO DE MESTRADO

O SageMath como ferramenta didática no ensino de funções reais para o ensino médio: uma abordagem prática

Jefferson David Moreira dos Santos

Instituto de Matemática

Maceió, Maio de 2024

PROFMAT

Universidade Federal de Alagoas - UFAL

Mestrado Profissional em Matemática em Rede Nacional - PROFMAT

Dissertação de Mestrado

O SageMath como ferramenta didática no ensino de funções reais para o ensino médio: uma abordagem prática

Jefferson David Moreira dos Santos

Maceió - Alagoas

Março de 2024

O SageMath como ferramenta didática no ensino de funções reais para o ensino médio: uma abordagem prática

Jefferson David Moreira dos Santos

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Mestrado Profissional em Matemática em Rede Nacional do Instituto de Matemática da Universidade Federal de Alagoas.

Orientador: Prof. Dr. Gregório Manoel da Silva Neto.

Maceió - Alagoas

Março de 2024

.

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 – 1767

S237s Santos, Jefferson David Moreira dos.
O SageMath como ferramenta didática no ensino de funções reais para o ensino médio : uma abordagem prática / Jefferson David Moreira dos Santos. - 2024.
98 f. : il.

Orientador: Gregório Manoel da Silva Neto.
Dissertação (Mestrado Profissional em Matemática) – Universidade Federal de Alagoas. Instituto de Matemática. Mestrado Profissional em Matemática em Rede Nacional. Maceió, 2024.

Bibliografia: f. 80-81.
Apêndices: f. 82-98.

1. SageMath (Software). 2. Sequência didática. 3. Funções (Matemática). 4. Python (Linguagem de programa de computador). 5. Ensino médio. I. Título.

CDU: 517.52

Folha de Aprovação

JEFFERSON DAVID MOREIRA DOS SANTOS

O SAGEMATH COMO FERRAMENTA DIDÁTICA NO ENSINO DE FUNÇÕES REAIS PARA O ENSINO MÉDIO: UMA ABORDAGEM PRÁTICA

Dissertação submetida ao corpo docente do Programa de Mestrado Profissional em Matemática em Rede Nacional (Profmat) do Instituto de Matemática da Universidade Federal de Alagoas, como requisito parcial para obtenção do grau de mestre e aprovada em 02 de Maio de 2024.

Banca Examinadora:

gov.br Documento assinado digitalmente
GREGORIO MANOEL DA SILVA NETO
Data: 12/07/2024 15:22:01-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Gregório Manoel da Silva Neto (UFAL – Presidente)

gov.br Documento assinado digitalmente
ANDRE LUIZ FLORES
Data: 12/07/2024 19:28:01-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. André Luiz Flores (UFAL – Examinador Interno)

gov.br Documento assinado digitalmente
ALVARO KRUGER RAMOS
Data: 12/07/2024 15:25:55-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Álvaro Krüger Ramos (UFRGS – Examinador Externo)

"É um mundo grande e lindo. A maioria de nós vive e morre no mesmo lugar onde nascemos e nunca chega a ver nada disso. Não quero ser a maioria de nós"

— George R. R. Martin, A Guerra dos Tronos

Agradecimentos

Ao Senhor Jesus, que me concedeu saúde e perseverança para seguir neste caminho.

Aos meus pais, Lúcia e Givanildo que com muito trabalho e esforço, me incentivaram e propiciaram educação de qualidade.

À minha esposa Izabelle Fiamma, e aos meus filhos, Ana Luiza e João Callebe, meus motivos e bens mais preciosos!

Ao Prof. Dr. Gregório Manoel da Silva Neto, que me ensina, orienta e aconselha com muita calma e paciência.

Aos *Opallas* (Horlando e José Cícero), amigos e companheiros do PROFMAT.

Aos professores do programa e da graduação, que foram essenciais para o meu desenvolvimento acadêmico e profissional.

Por fim, aos meus eternos professores Benedito e Ricardo, que me apresentaram as primeiras demostrações matemáticas.

Resumo

Este trabalho propõe mostrar as vantagens em utilizar o *SageMath* como ferramenta didática por professores e alunos da Educação Básica, dando suporte às aulas sobre funções polinomiais para o 1º ano do Ensino Médio. Especificamente, na manipulação de expressões algébricas e interação com os parâmetros das funções no *software*. Ao mesmo tempo, introduzir conceitos iniciais da linguagem de programação *Python* para construir algoritmos, de modo a modelar e resolver problemas matemáticos, e assim, visualizar os respectivos gráficos e representações.

Palavras-Chave: SageMath; sequência didática, funções, *Python*, Ensino Médio.

Abstract

This work proposes to show the advantages of using *SageMath* as a digital tool practice by basic education teachers and students, providing support in classes on functions polynomials for the 1st year of high school. Specifically, in the manipulation of algebraic expressions and interaction with function parameters in the software. At the same time, to use introductory concepts from the *Python* programming language to build algorithms, in order to model and solve mathematical problems, and thus, visualize the respective graphics and representations.

Keywords: SageMath; didactic sequence; Functions; Python; High School.

Lista de Figuras

Lista de Tabelas

Sumário

INTRODUÇÃO

Durante os anos finais do Ensino Fundamental, a escola propicia o ensino de álgebra em seu currículo, abordando conceitos básicos e desenvolvendo a compreensão em notação, operações e estruturas algébricas para os alunos. Isso possibilita a transição entre a matemática básica (*operações aritméticas*) e a abstrata (*operações algébricas*).

A BNCC orienta que os estudantes desenvolvam habilidades como: identificar padrões e sequências numéricas, expressar relações entre grandezas com linguagem simbólica e utilizar estruturas algébricas para representar situações diversas. Dessa forma, é possível modelar e resolver problemas do cotidiano matematicamente.

Nessa perspectiva, os livros didáticos de matemática seguem as indicações do documento normativo mencionado no parágrafo anterior, no sentido de incorporar e incentivar o uso de ferramentas tecnológicas nas aulas de matemática. Assim, recursos como o *Scratch*, as planilhas eletrônicas e o GEOGEBRA tornam a aula mais dinâmica e interessante para os alunos, que ficam cada vez mais inseridos e familiarizados com a tecnologia.

> [...] no Ensino Médio o foco é a construção de uma visão integrada da Matemática, aplicada à realidade, em diferentes contextos. Consequentemente, quando a realidade é a referência, é preciso levar em conta as vivências cotidianas dos estudantes do Ensino Médio – impactados de diferentes maneiras pelos avanços tecnológicos, pelas exigências do mercado de trabalho, pelos projetos de bem viver dos seus povos, pela potencialidade das mídias sociais, entre outros. Nesse contexto, destaca-se ainda a importância do recurso a tecnologias digitais e aplicativos tanto para a investigação matemática como para dar continuidade ao desenvolvimento do pensamento computacional, iniciado na etapa anterior. (BRASIL, 2018, p. 528).

Portanto, neste trabalho, iremos abordar a utilização do *software SageMath* como ferramenta didática nas aulas de matemática, de modo a promover a continuidade e integração do ensino de matemática com recursos tecnológicos educacionais, em especial, na introdução de conceitos básicos e iniciais da linguagem de programação *Python* na construção, visualização e análise dos gráficos e comportamento das funções estudadas no 1º ano do ensino médio.

Inicialmente, no Capítulo 1, traremos o refteórico que embasa o desenvolvimento desta dissertação. Nele, abordaremos características de *softwares* matemáticos conhecidos e similares ao *SageMath* no contexto educacional e do *Python* como linguagem de programação para iniciantes.

No Capítulo 2, apresentamos as formas de uso do *SageMath*, a saber: o modo *online*, com o *SageMathCell* para o uso rápido e prático de uma única célula de comando; e o *CoCalc*, para projetos maiores com a possibilidade de integração com textos e outras células de comando. E também, o modo *offline* com a instalação completa e gratuita do *software*.

No decorrer do Capítulo 3, veremos a interface gráfica do programa, como executar códigos e realizar operações aritméticas nas células de comando. Ainda, como utilizar o *SageMath* algebricamente, definido e declarando variáveis para trabalhar com fórmulas e funções matemáticas.

No Capítulo 4, descrevemos a metodologia seguida neste trabalho, a natureza e perspectiva da pesquisa, as motivações, justificativas e objetivos a serem alcançados e investigados em sua aplicação no ensino público.

Para averiguação dos fatos citados no parágrafo anterior, o capítulo 5 é composto por uma sequência de aulas e resoluções de exercícios a serem desenvolvidos no *SageMath*. Seguido pelo Capítulo 6, com os resultados apresentados após aplicação com alunos do 1º ano do ensino médio.

1 *Softwares* Matemáticos na Educação

1.1 Documentos Norteadores e Inclusão Digital

A utilização de recursos tecnológicos nas aulas de matemática é indicação dos Parâmetros Curriculares Nacionais (PCN). O documento recomenda a incorporação de computadores e softwares no ensino e o redirecionamento curricular em favor do desenvolvimento de novas habilidades e procedimentos. Na área da álgebra, também destaca a importância da compreensão e análise do comportamento dos gráficos de funções; bem como das alterações ocasionadas pela mudança de parâmetros em suas respectivas funções. (BRASIL, 1998, p. 41-45).

Conforme as Orientações Curriculares para o Ensino Médio, é fundamental que a escola organize os conteúdos disciplinares como ferramentas para compreensão da sociedade, promova a reciprocidade entre matemática e a tecnologia e capacite os indivíduos para um mundo tecnológico e informatizado. (BRASIL, 2006, p. 87-89). Consequentemente, conectar saberes, inovações e suas aplicações no cotidiano.

Em harmonia com os parâmetros e orientações acima, o Referencial Curricular de Alagoas propõe, em suas competências, que os estudantes elaborarem, modelem, resolvam e apresentem soluções de problemas do cotidiano, utilizando *softwares* de geometria dinâmicos. (BRASIL, 2021, p. 222). Contudo, é fato que o ensino de matemática nas escolas públicas estaduais ainda não condiz com uma realidade tecnológica que o favoreça.

Para CARMARGO e DAROS (2018, p. 46), a inovação tecnológica na educação não se realiza apenas com a modernização dos equipamentos e ferramentas, mas em completude da associação de metodologias ativas centradas na autonomia do estudante,

no desenvolvimento da aprendizagem colaborativa e na interdisciplinaridade. Assim, o professor deve assumir o papel de mediador entre os conteúdos curriculares e a utilização dos recursos digitais na sala de aula, distanciando-se dos métodos expositivos tradicionais e proporcionando a inclusão digital com funcionalidade pedagógica.

A escola é um local de promoção da igualdade no acesso e no desenvolvimento de habilidades tecnológicas, inserindo os alunos na era digital e criando um ambiente inclusivo. Segundo SOUZA (2011, p.78-79), a acessibilidade e a inclusão digital permitem não somente a pessoa com deficiência potencializar seu aprendizado, mas também ampliar suas oportunidades culturais e profissionais, melhorando sua condição de vida.

1.2 Contextualização

A matemática é percebida como uma área complexa, caracterizada por conceitos abstratos que podem apresentar dificuldades para os alunos em sua visualização e compreensão. Assim, a falta de embasamento teórico sólido não permite sua conexão, em sala de aula, com a aplicação prática no cotidiano dos alunos, e pode contribuir com a falsa percepção de que matemática é irrelevante em suas vidas. O desinteresse e a falta de engajamento foram ampliados pelas consequências da pandemia de COVID-19, principalmente para aqueles que não possuíam acesso adequado a dispositivos eletrônicos e à internet, resultando em um déficit significativo no aprendizado da matemática, que requer atenção e intervenção.

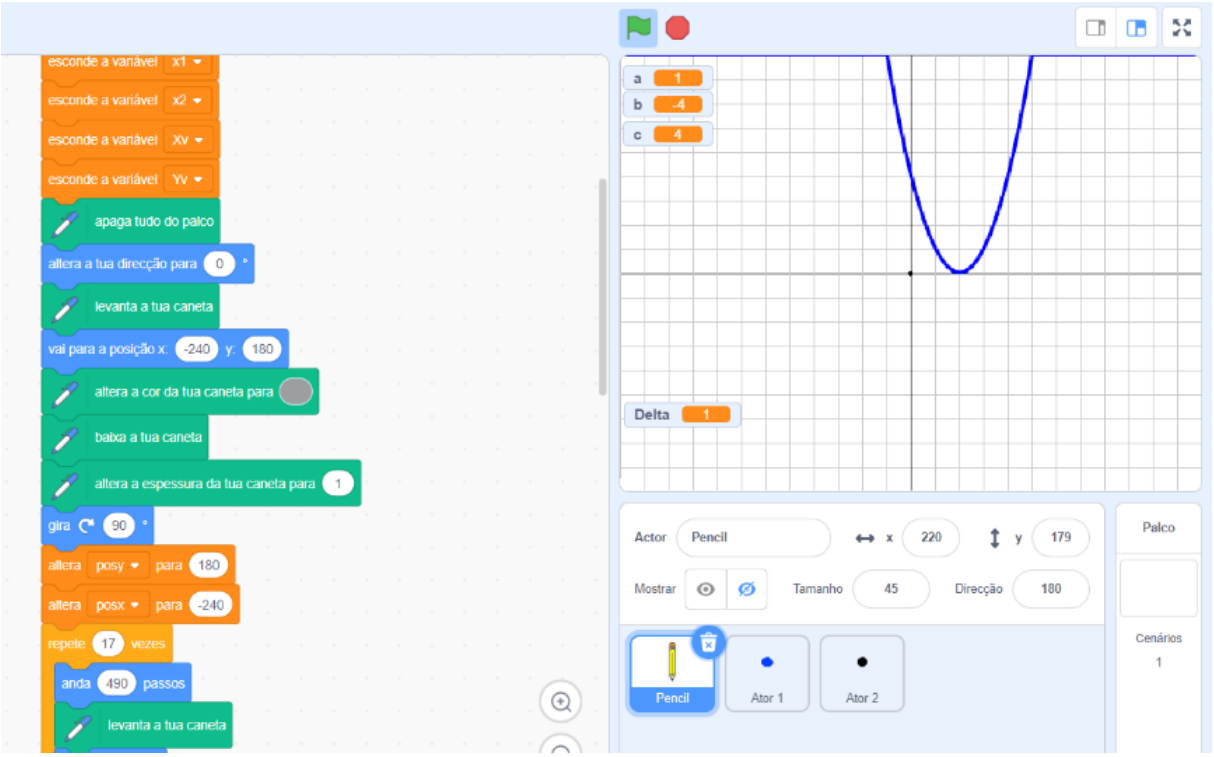
Simultaneamente, o período de quarentena impulsionou a transformação digital nas escolas, compelindo as instituições a adotarem novas ferramentas e metodologias, e se inserirem enquanto ambiente tecnológico educacional. Uma dessas inovações é a ampliação no uso de aplicativos matemáticos durante as aulas, que oferecem uma experiência de aprendizagem imersiva e alinhada com a realidade "conectada" dos alunos. Portanto, este trabalho visa propagar a utilização de *softwares* matemáticos como recursos didáticos, na educação básica.

1.3 *Softwares* Matemáticos

Existem diversos *softwares* matemáticos empregados no contexto educacional, os quais fornecem recursos e ferramentas específicas que facilitam o ensino e a aprendizagem tecnológica da disciplina através da programação. Esses programas possuem uma variedade de funcionalidades e aplicações que podem auxilixar os alunos na compreesão dos conceitos matemáticos de forma mais visual e interativa, apresentando interfaces intuitivamente amigáveis e com linguagens próprias para o desenvolvimento de algoritmos e projetos matemáticos. Nesta perspectiva, se enquadram o *Scratch*, o *Geogebra*, o *MATLAB*, o *Maple* e o *SageMath*.

O ***Scratch*** é uma linguagem de programação em blocos extremamente popular, indicada para crianças aprenderem os conceitos iniciais de programação, como ilustra a Figura 1.1. Amplamente utilizado na educação, o *software* promove o pensamento computacional na resolução de problemas na criação de programas, animações e jogos.

Figura 1.1: *Scratch*



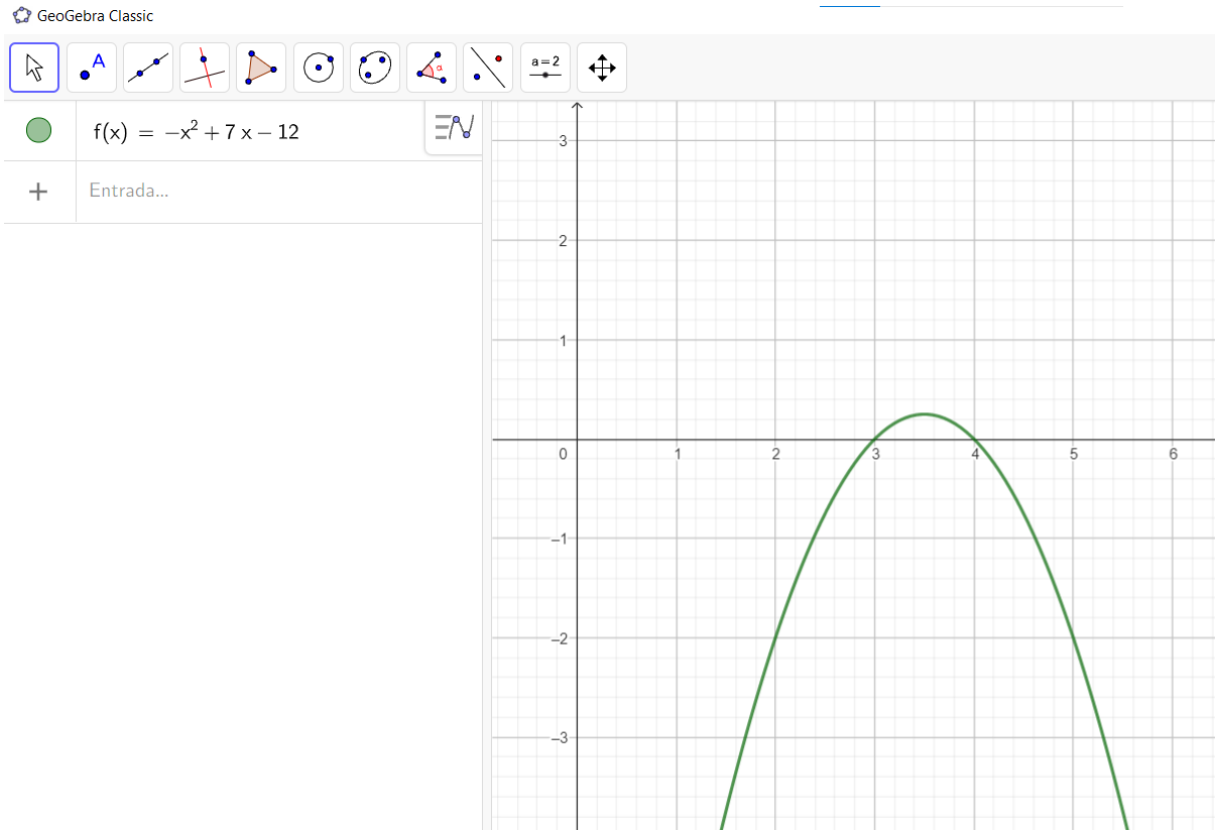
Fonte: O autor.

Apesar de ser um programa de fácil aprendizado devido a sua sintaxe baseada em

blocos de código, com a evolução dos estudos e aplicações, o *Scratch* vai se tornando limitado em termos de recursos e lento em relação a projetos maiores e mais complexos.

O ***GeoGebra*** é um *software* educacional de matemática dinâmica gratuito, num ambiente que permite a plotagem de funções e construções geométricas de maneira dinâmica, como ilustrado na Figura 1.2. A compreensão e o aprendizado são maximizados de forma visual e interativa, estimulando o aluno a explorar os conteúdos abordados.

Figura 1.2: *Geogebra*



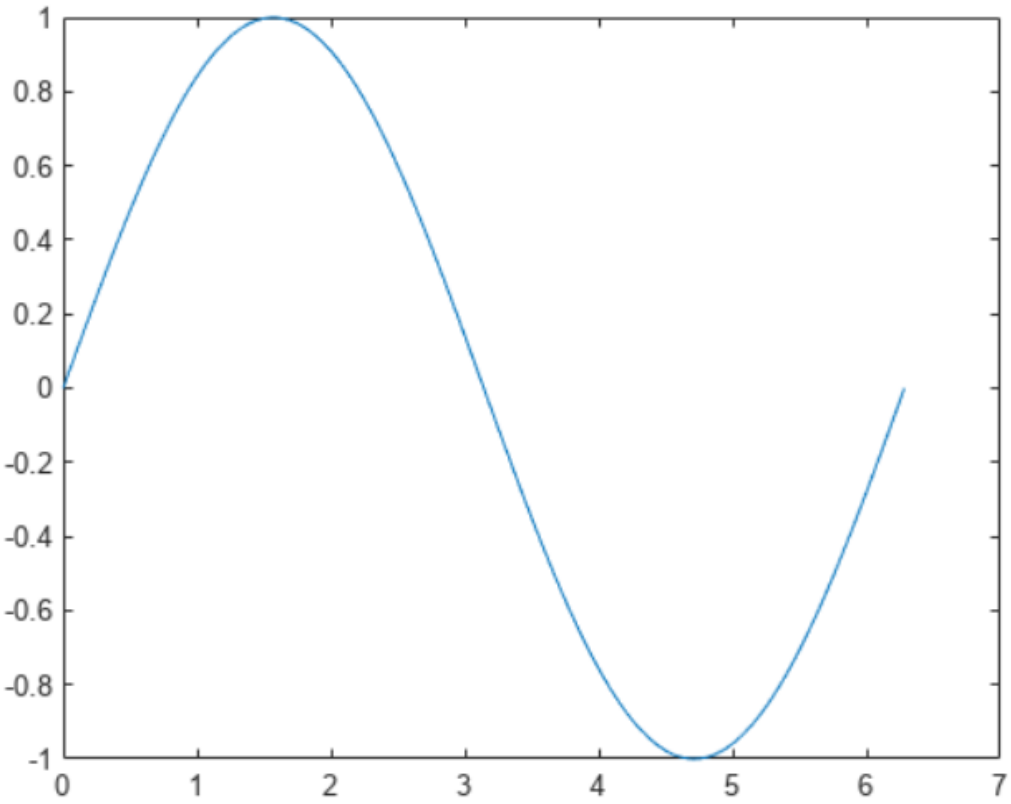
Fonte: O autor.

No programa é possível elaborar construções matemáticas interativas através da implementação de *scripts*. Embora tenha funcionalidades algébricas, não é tão robusto quanto um *software* de cálculo simbólico.

O ***MATLAB*** é um *software* matemático muito utilizado em ambientes acadêmicos, ciências aplicadas e engenharia, que fornece uma plataforma que favorece o aprendizado em pesquisas matemáticas, também sendo utilizado no âmbito educacional. Veja a sintaxe e a representação gráfica da função *seno* exibidas na Figura 1.3:

Figura 1.3: *MATLAB*

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

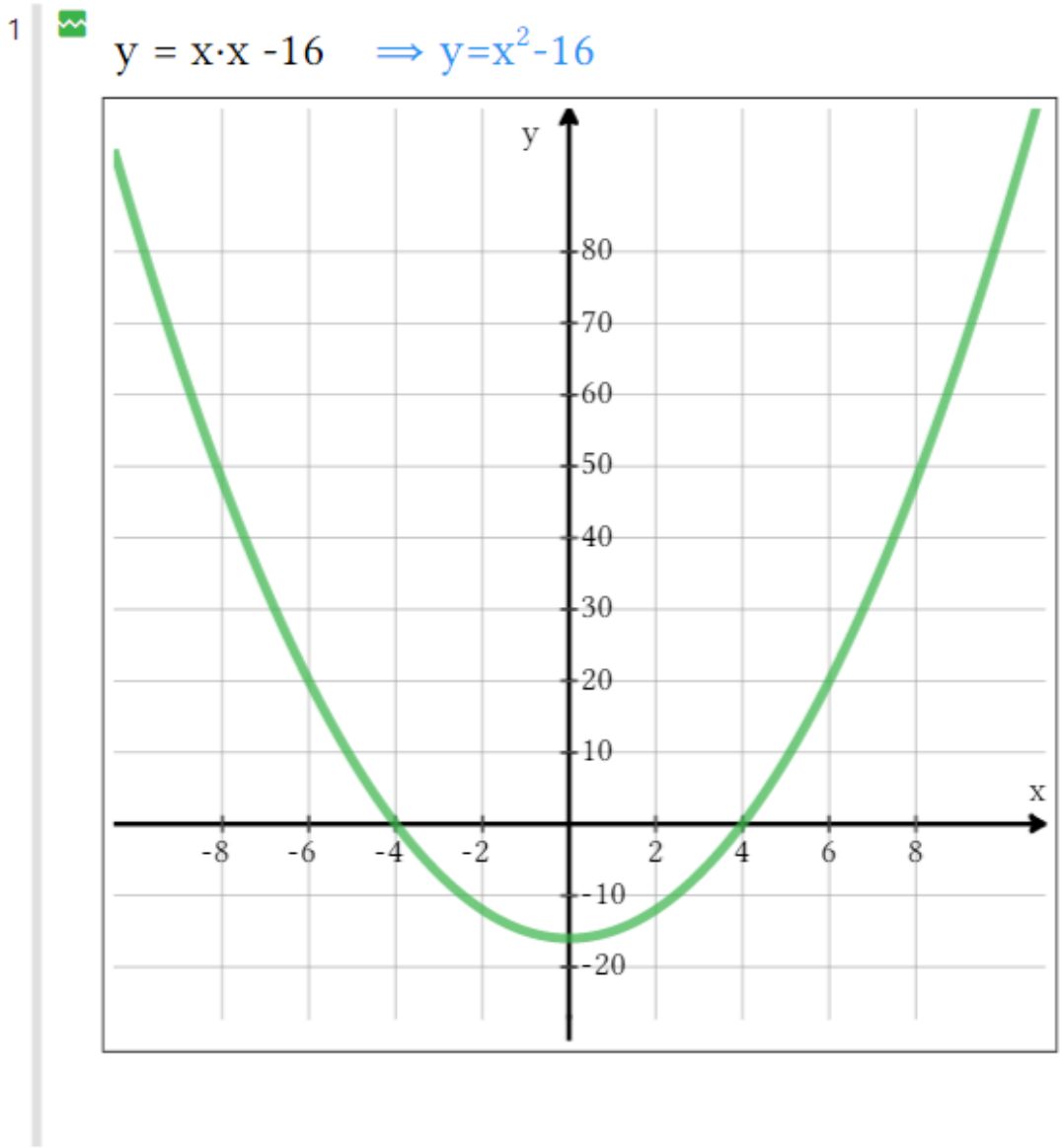


Fonte: O autor.

O programa oferece recursos avançados de plotagem de gráficos, o que é valioso para visualização de dados e representações de funções matemáticas. Contudo, é uma ferramenta comercial e sua licença pode ser custosa para indivíduos e instituições de ensino com recursos financeiros limitados.

O ***Maple*** possui uma interface intuitiva que torna muito fácil analisar, visualizar e explorar problemas matemáticos. O *software*, conhecido por suas capacidades de cálculo simbólico avançado, possibilita manipulações algébricas complexas e resolução simbólica de equações; também permite visualizações detalhadas e personalizadas de dados e funções matemáticas. Mas, seu uso exige uma licença comercial paga. Observe a sintaxe do comando e a plotagem do gráfico da função $y = x^2 - 16$ na Figura 1.4:

Figura 1.4: *Maple*



Fonte: O autor.

Apesar de todas as qualidades do *Maple*, a aquisição de licenças e a curva de aprendizado associada às particularidades da linguagem de programação, podem dificultar o acesso e sua utilização eficaz nas instituições educacionais públicas. Além disso, existem alternativas de código aberto e gratuitas, como o *software SageMath* e a linguagem de programação *Python*, com bibliotecas como *SymPy*, *SciPy* e *Matplotlib*, que oferecem recursos similares.

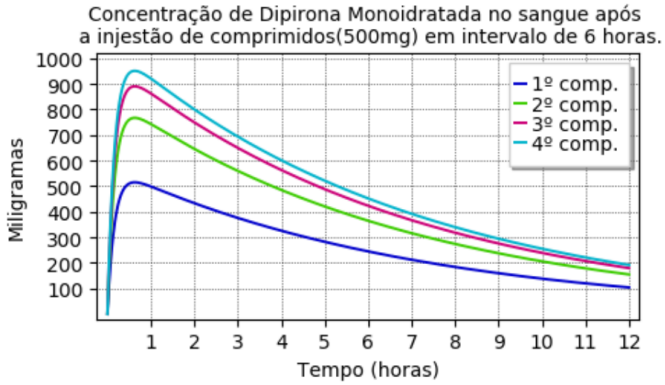
O ***SageMath*** é um *software* matemático com linguagem de programação baseada

em *Python*; veja a sintaxe da linguagem na Figura 1.5. Antes conhecido como *Sage*, abreviação de (Sistema lgébrico e geométrico de experimentações), foi criado como alternativa aos principais *softwares* matemáticos como *Magma*, *Maple*, *Mathematica* e *Matlab*. (Silva, 2019, p. 1-3).

Figura 1.5: *SageMath*

```
In [19]: var('k1 k t m0'); t = 6; k1 = 6; m0 = 500; k =1/7
         f(x) = m0*(1-e^(-k1*x))*e^(-(1/7)*(x-1)); g(x) = (f(t) + m0)*(1-e^(-k1*x))*e^(-(1/7)*(x-1))
         h(x) = (g(t) + m0)*(1-e^(-k1*x))*e^(-(1/7)*(x-1)); p(x) = (h(t) + m0)*(1-e^(-k1*x))*e^(-(1/7)*(x-1))
         plot([f,g,h, p],0,12, ymax=1000, ticks=[[1..12],[100,200..1400]], axes_labels=['Tempo (horas)', 'Miligr
             axes=False, frame=True, gridlines='normal', legend_label=['1º comp.', '2º comp.', '3º comp.', '4º c
             title='Concentração de Dipirona Monoidratada no sangue após \n a injestão de comprimidos(500mg) em
             axes_labels_size=1, figsize=[5,3])
```

Out[19]:



Fonte: O autor.

Os programas mencionados anteriormente podem ser usados no contexto educacional, cada um com funcionalidades e características próprias. No entanto, o *SageMath* é um *software* de matemática gratuito e de código aberto, baseado na linguagem de programação *Python* que possibilita a implementação de algoritmos complexos, análises de dados e a resolução de equações eficientemente. Com base nesses parâmetros, nos parece a melhor opção, dentre as opções acima citadas, como ferramenta didática na escola pública.

Embora o público alvo do *Sage* seja composto por estudantes universitários, professores acadêmicos e pesquisadores em matemática, sua interface é convidativa ao aprendizado, mostrando-se um recurso valioso para o usuário que queira trabalhar com matemática computacional. Logo, alunos e professores podem utilizá-lo na compreensão de conceitos matemáticos e resolução de problemas durante todo o ensino médio, especialmente em álgebra, geometria, matrizes, sistemas lineares e análise combinatória.

1.4 *Python* e *SageMath* na Educação Básica

A Base Nacional Comum Curricular promove, em suas habilidades e competências, a construção do conhecimento com técnicas e tecnologias digitais. Por exemplo, na habilidade **EM13MAT405**, o documento orienta utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática. (BRASIL, 2018, p. 539). Também, de acordo com Bellapu (2022), a linguagem mais utilizada em 2022 foi o *Python*, considerada uma das mais acessíveis, fáceis de aprender e recomendada para iniciantes. Portanto, a combinação dos recursos matemáticos do *SageMath* com a linguagem de programação, é uma escolha adequada para o desenvolvimento de habilidades em programação, na resolução de problemas e na identificação de padrões e algoritmos, práticas que capacitam os alunos para os desafios encontrados no mundo do trabalho cada vez mais digital e tecnológico.

Como aponta Matthes (2026, p. 30), Python é uma linguagem extremamente eficiente, que necessita de menos linhas de código para seus programas em comparação a outras linguagens de programação. No mesmo sentido, a comunidade do *software* indica-o como a primeira a ser aprendida por iniciantes, pois mesmo com uma sintaxe mais simples, é bastante utilizada em pesquisas acadêmicas, no desenvolvimento de aplicativos, na criação de jogos, automação de tarefas e aplicações na internet.

Durante o primeiro ano do ensino médio, os estudantes aprofundam-se nos estudos das funções reais, em particular, das funções lineares, quadráticas, exponenciais e logarítmicas. Ao mesmo tempo, a abordagem interativa e tecnológica deste conteúdo, proporciona um aprendizado significativo na análise de suas representações gráficas no plano cartesiano, assim como das características e propriedades das funções matemáticas referidas. Oportunamente, o comando `plot( )` do *SageMath* possibilita a representação de funções através da plotagem de gráficos. De modo técnico, este comando renderiza o gráfico utilizando a biblioteca *MatPlotLib* do *Python*, conjunto de códigos prontos que fornece funcionalidades adicionais da linguagem de programação. Assim, é possível construir, visualizar e interagir com os gráficos em seus parâmetros, obtendo uma melhor compreensão das funções reais.

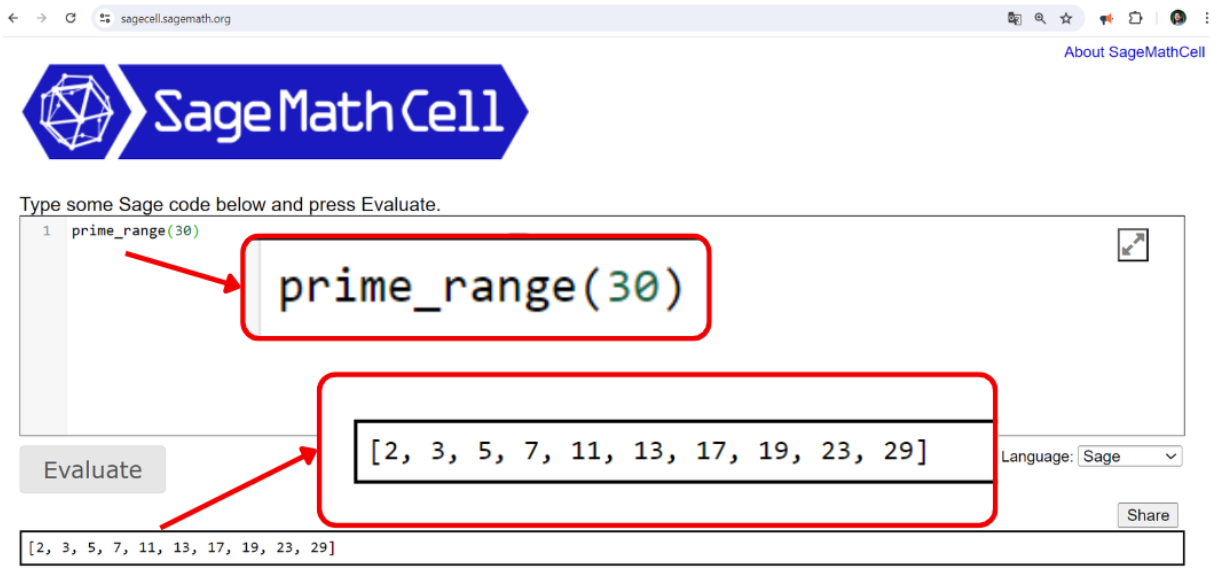
2 *Sagemath*: Uso e Instalação

Originalmente, as informações completas sobre a documentação, modos de uso e instalação do *software* nos sistemas *Windows*, *Linux* e *Mac OSX* podem ser encontradas no site oficial: *https://www.sagemath.org/*. Entretanto, neste capítulo traremos um resumo prático sobre o modo de uso *on-line*: no *SageMathCell* e no *CoCalc*; e sobre o modo de uso *off-line*: *software* instalado computador.

2.1 Modos de uso *on-line*

O *SageMathCell* é uma interface web que permite executar comandos do *Sage* diretamente na página *https://sagecell.sagemath.org/*; com acesso rápido, prático e sem efetuar qualquer cadastro ou login pelo usuário. Contudo, possui uma única célula de comando e o código executado é deletado após a atualização da página para a inserção uma nova entrada de código. Vejamos um exemplo do comando `prime_range(30)`, que apresenta os números primos no intevalo $[0, 30]$, na Figura 2.1:

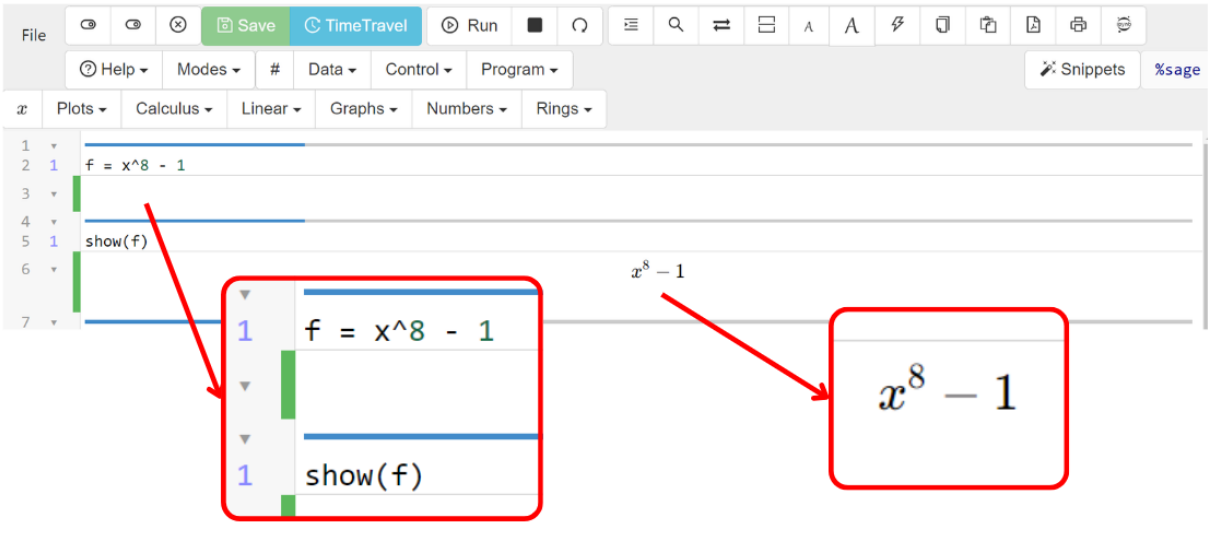
Figura 2.1: *SageMathCell*



Fonte: O autor.

O *CoCalc (Collaborative Calculation in the Cloud)* é o ambiente *on-line* para criar, executar e salvar os comandos e textos no *Sage*, sem a necessidade de fazer instalações. Para tanto, basta efetuar o cadastro e o login em *https://cocalc.com/*. A vantagem desse modo, em relação ao *SageMathCell*, é a possibilidade da criação de projetos: *worksheets* do *SageMath* (arquivo que combina código e células de texto formatado), *notebooks* (arquivo do *Jupyter Notebook*, com suporte ao *Sage, Python, Octave, R e Sage)*; e sua integração com *LaTeX*. Ver figura 2.2.

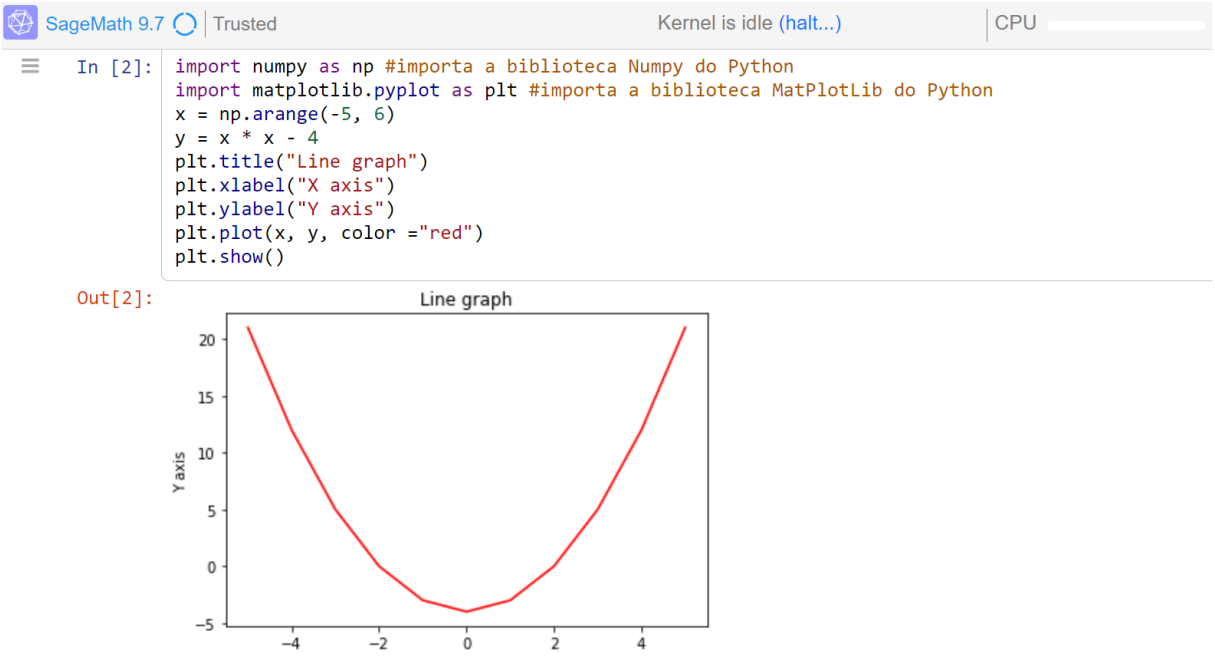
Figura 2.2: *CoCalc*



Fonte: O autor.

O *Jupyter Notebook* é um ambiente de desenvolvimento interativo baseado na *web* para *notebooks*, código e dados. No *browser* (navegador), ele é a plataforma para uso do *SageMath* no modo *off-line*. Em acréscimo, ambos os ambientes permitem importar bibliotecas do *Python* para acrescentar mais funcionalidade à linguagem. Na Figura 2.3, temos um exemplo *on-line* do *SageMath* no *CoCalc*, onde executamos comandos das bibliotecas: *NumPy* (operações com arrays) e *MatPlotLib* (criação de gráficos 2D e 3D).

Figura 2.3: Bibliotecas do *Python* no *CoCalc*



Fonte: O autor.

Logo, a utilização do *CoCalc* oferece todos os recursos do *software* e a integração com outras ferramentas, tornando-se um ambiente computacional versátil.

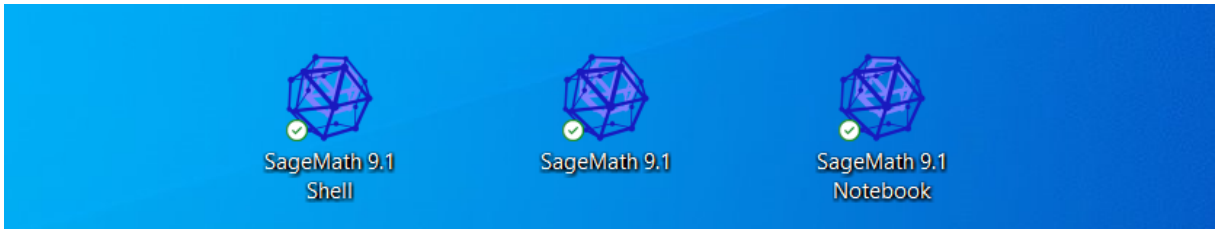
2.2 Modo de uso *off-line*

O *software* é livre e multiplataforma, e sua instalação para o modo *off-line* é compatível com os sistemas *Windows*, *Linux* e *Mac OSX*. O download está disponível oficialmente em *https://doc.sagemath.org/html/en/installation/index.html* ou, preferivelmente, no site *https://sagectu.com.br/instalacao.html* do livro SILVA, Leon; MACHADO, Ricardo; SANTOS, Marcelo. **Elementos de Computação Matemática com SageMath**. 1ª Edição. Rio de Janeiro: SBM, 2019. Neste último, com guias

de instalação, links e informações em português para cada um dos sistemas operacionais citados.

Após o download da versão para o sistema desejado, um arquivo executável será baixado no computador do usuário. Agora, é só executar este arquivo, escolher a pasta de destino, marcar a opção para criar ícones de atalho e aguardar o processo de instalação. Em seguida, serão criados ícones no *Desktop* (Área de trabalho) para acesso pelo *Shell* (interpretador de linhas de comando), pelo terminal de comandos do sistema; ainda, para o acesso ao *notebook* pela interface do *Jupyter Notebook*. Ver Figura 2.4:

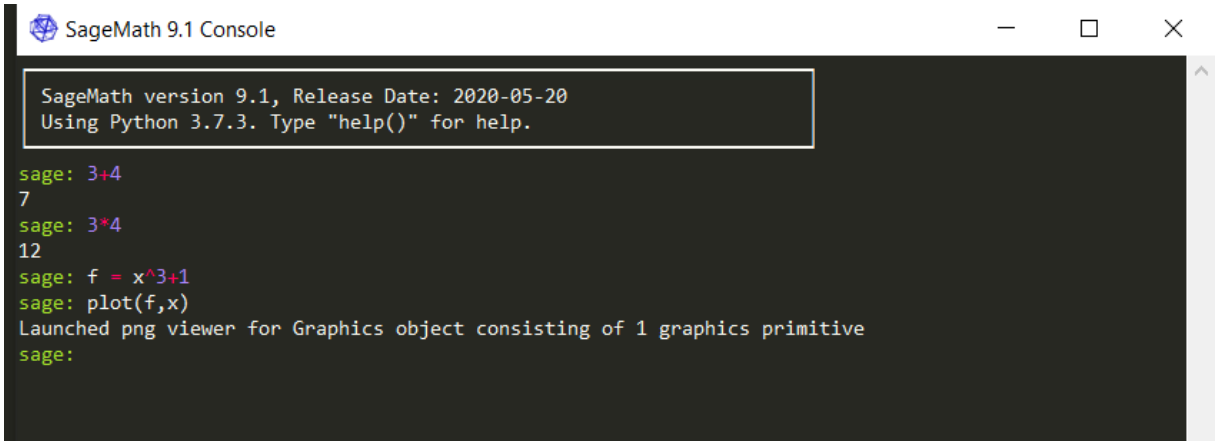
Figura 2.4: Ícones de acesso no sistema operacional



Fonte: O autor.

Shell é um programa utilizado para processar comandos, basicamente é onde há a interação com o *Kernel* (núcleo) do sistema operacional. O terminal, onde roda um *shell*, é um programa onde gerenciamos os recursos mais complexos do sistema; geralmente é uma tela preta, sem botões ou elementos gráficos, onde executamos as linhas de comando. Na Figura 2.5, temos um exemplo do uso da versão 9.1 do *Sagemath* iniciada diretamente no terminal do *Windows*.

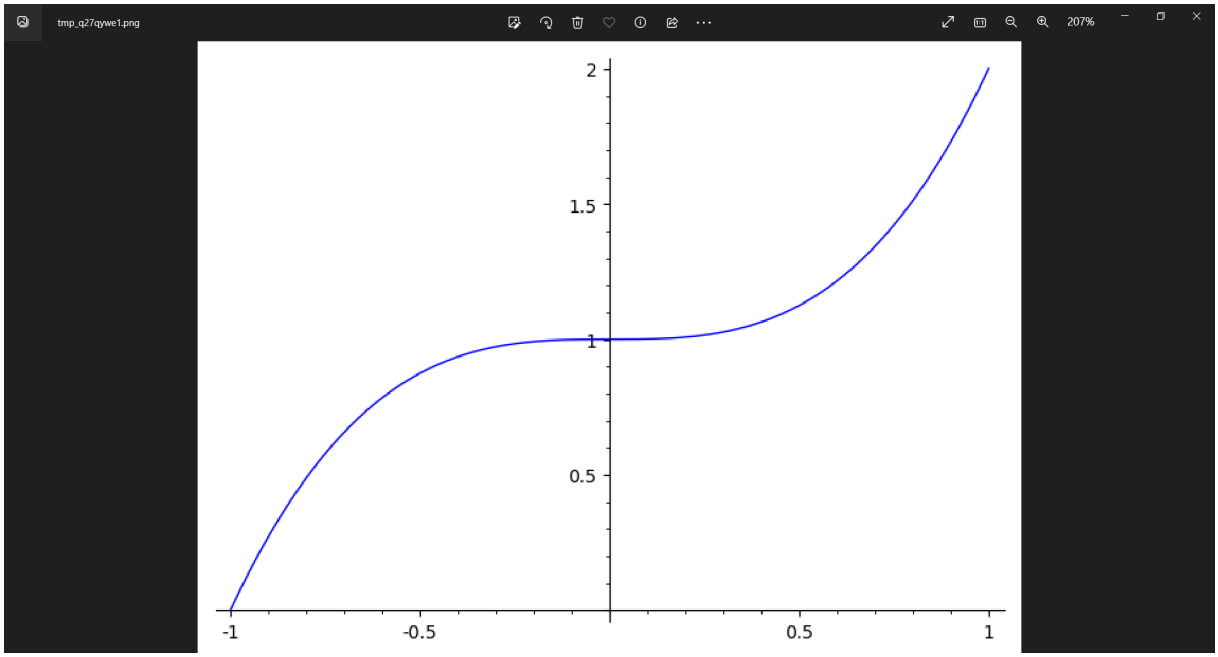
Figura 2.5: *Sagemath* no terminal do *Windows*



Fonte: O autor.

Como podemos verificar na imagem anterior, foram digitados quatro comandos nos *(inputs)* (entradas de código) e, após as execuções, o *sage* emitiu os resultados apenas em dois *(outputs)* (saídas das informações processadas pelo programa). Entretanto, perceba que os dois primeiros *inputs* referentes aos operadores aritméticos de adição (+) e multiplicação (∗), geraram *outputs* numéricos no terminal. O terceiro *input*, armazenou a expressão $x^3 + 1$ na letra f (definindo a função $f = x^3 + 1$), através do operador igual (=) sem gerar *output*. Agora, no quarto *input*, temos o comando `plot(f, x)` para plotagem de gráficos em duas dimensões, que gerou um arquivo na pasta temporária do *Windows* em formato PNG, apresentando o gráfico de f no plano cartesiano. Ver figura 2.6:

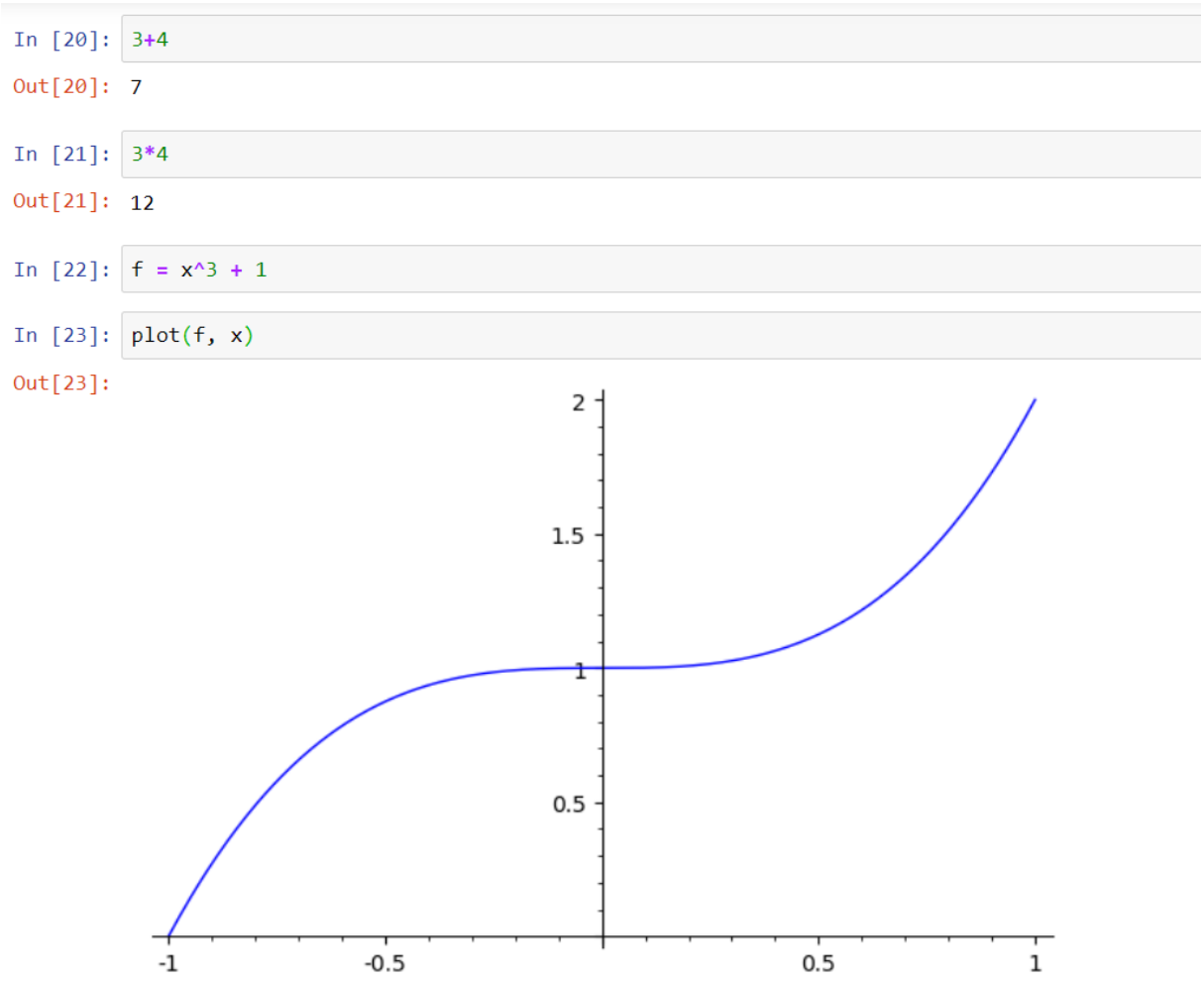
Figura 2.6: Gráfico de f na pasta temporária do *Windows*



Fonte: O autor.

A experiência de uso no *Sagemath* executado diretamente no terminal, se mostra funcional, mas não tão agradável e intuitiva quanto na interface do *Jupyter Notebook* ou do *CoCalc*. Entretanto, ao clicar no ícone *Notebook* da Figura 2.4, o *software* será inicializado *off-line* numa aba do navegador de internet. Observe a interface do *Jupyter Notebook* na Figura 2.7, com os mesmos comandos executados na Figura 2.5.

Figura 2.7: *Sagemath* no *Jupyter Notebook*



Fonte: O autor.

Por mostrar praticidade e alta funcionalidade na criação, apresentação e organização dos projetos (*Worksheets*), o *Jupyter Notebook* será a interface gráfica, junto com a versão 9.1 do *SageMath* instalada, que utilizaremos durante o desenvolvimento e aplicação deste trabalho.

3 Primeiros Passos no *SageMath*

Neste capítulo, veremos conceitos introdutórios de linguagem de programação para dar os primeiros passos no *SageMath*, percorrendo um caminho fundamentalmente prático na compreensão e execução dos comandos no *software*.

3.1 Interface Gráfica

Para acessar um arquivo na interface gráfica do *SageMath* instalado, basta clicar sobre o ícone correspondente (*SageMath 9.1 Notebook*) no local de instalação, Figura 2.4, o que abrirá o *Jupyter Notebook* diretamente em uma aba do navegador de internet do usuário. Em seguida, clicar em *"New"* (novo) e selecionar a versão *"SageMath 9.1"* para criar um novo arquivo e iniciar a utilização deste ambiente, como ilustra a Figura 3.1:

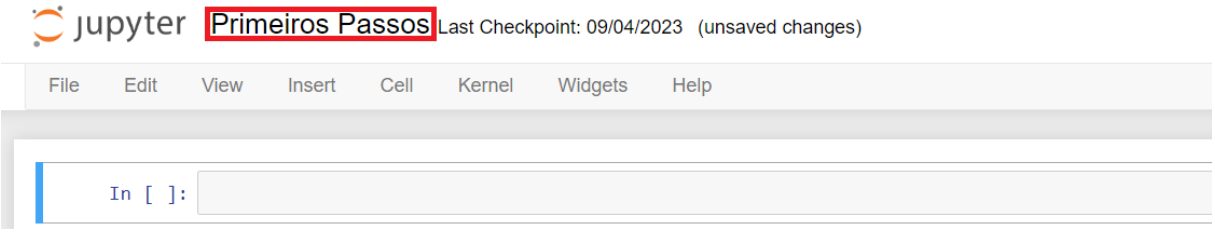
Figura 3.1: Criar e abrir um novo arquivo no *Jupyter Notebook*

Fonte: O autor.

Após os procedimentos do parágrafo anterior, um arquivo *"Untitled"* (sem título)

será aberto em uma nova aba. Mas, ao clicar em *"Untitled"*, é possível renomeá-lo. Na Figura 3.2, o arquivo foi renomeado para "Primeiros Passos".

Figura 3.2: Renomeando arquivo no *Jupyter Notebook*



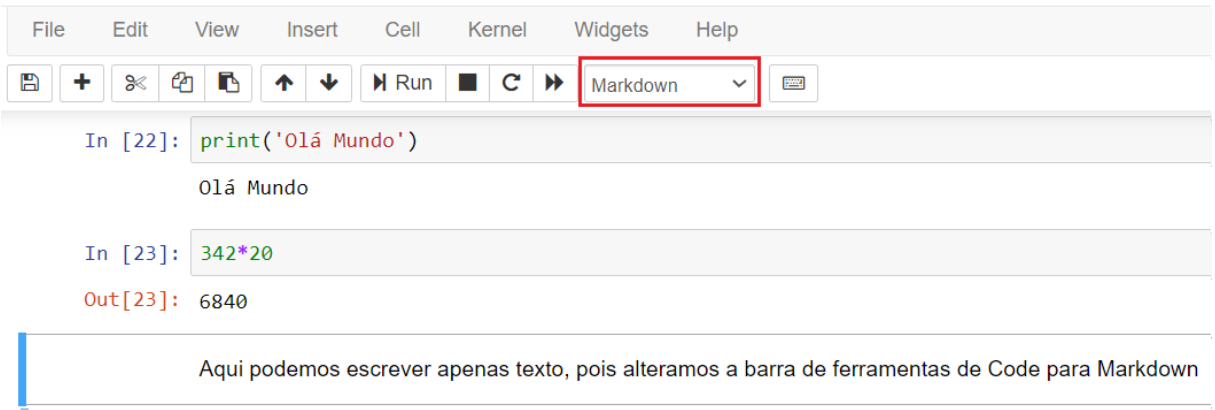
Fonte: O autor.

Observe que o *layout* do *Jupyter Notebook* é similar aos *softwares* mais conhecidos do *Microsoft Office* (*Word*, *Excel* e *Power Point*). Assim, se torna intuitivo para o usuário acessar as demais funcionalidades da barra de ferramentas do programa.

3.2 Células de comando

O programa fornece células de entrada "In" (abreviação de *input*), onde se insere o código; e para executá-lo, clica-se em *Run* (correr em inglês) na barra ferramentas ou pressiona-se os atalhos "ctrl + enter" e "shift + enter", que a resposta será emitida em "Out" (abreviação de *output*). Após executar o código, o programa seguirá para a próxima célula de comando.

Figura 3.3: Primeiros comandos



Fonte: O autor.

Na Figura 3.3, executamos o comando `print('Olá Mundo')` na entrada 1, imprimindo a frase: "Olá Mundo" como saída. Em seguida, na entrada 2, o comando `342*20` mostrou o resultado do produto entre 342 por 20. Por último, na linha 3 foi digitado texto livre ao alterar a opção *Code* (código) para *Markdown* na barra de ferramentas.

Code permite escrever e executar comandos *SageMath*, como códigos, cálculos matemáticos, definições de variáveis e visualização de gráficos; e *Markdown*, permite formatar texto livremente, incluindo títulos, parágrafos, listas, imagens e links.

3.3 Operadores aritméticos e Comentários no código

O *SageMath* consegue efetuar operações matemáticas da mesma forma que uma calculadora científica, seguindo a ordem de precedência e dos operadores aritméticos. Veja na Tabela 3.1, a sintaxe para a realização de algumas dessas operações:

Tabela 3.1: Operadores aritméticos

Operações fundamentais	`a+b, a-b, a*b, e a/b`
Potenciação	`a^b`
Raiz quadrada	`sqrt(a)`
Raiz n-ésima	`a^(1/n)`

Fonte: O autor.

Textos, números e símbolos precedidos pelo caractere `#` (cerquilha ou *hashtag*), serão lidos como comentários e não como um comando. Este recurso permite inserir explicações, anotações e observações que não são interpretadas como código executável.

Na Figura 3.4, temos a aplicação das 4 operações básicas e do `#` para escrever comentários em cada célula de comando, destacando o modo de obter números inteiros (*int*) ou reais (números de ponto flutuante - *float*). Note que na divisão 7/8, o *SageMath* compreende a operação como uma divisão entre inteiros, mostrando o resultado em forma de fração. Entretanto, para obter o resultado real (*float*) é preciso escrever pelo menos um dos números como *float* (com um ponto entre casas decimais). Neste caso, $7.0/8 = 0,875$ ou $7/8.0 = 0,875$.

Figura 3.4: Operadores aritméticos e comentários

Fonte: O autor.

Para a operação de potenciação temos os símbolos ‘‘∧’’ (circunflexo) ou ‘‘**’’ (duplo asterisco); e para raiz quadrada, temos a função `sqrt( )`, abreviação do inglês *square root* (raiz quadrada). Mas, também pode-se utilizar a radiciação com outros índices, ao inserir o expoente fracionário na potenciação. Por exemplo, $1024 \wedge (1/10)$ será interpretado como $\sqrt[10]{1024}$. Veja a Figura 3.5:

Figura 3.5: Operadores de potenciação e radiciação.

```
In [6]: 2^10 # Potenciação da forma a^b
Out[6]: 1024

In [7]: 2**10 # Potenciação da forma a**b
Out[7]: 1024

In [8]: sqrt(625) # Radiciação da forma sqrt(a)
Out[8]: 25

In [9]: 1024^(1/10) # Raiz décima da forma a^(1/n)
Out[9]: 2
```

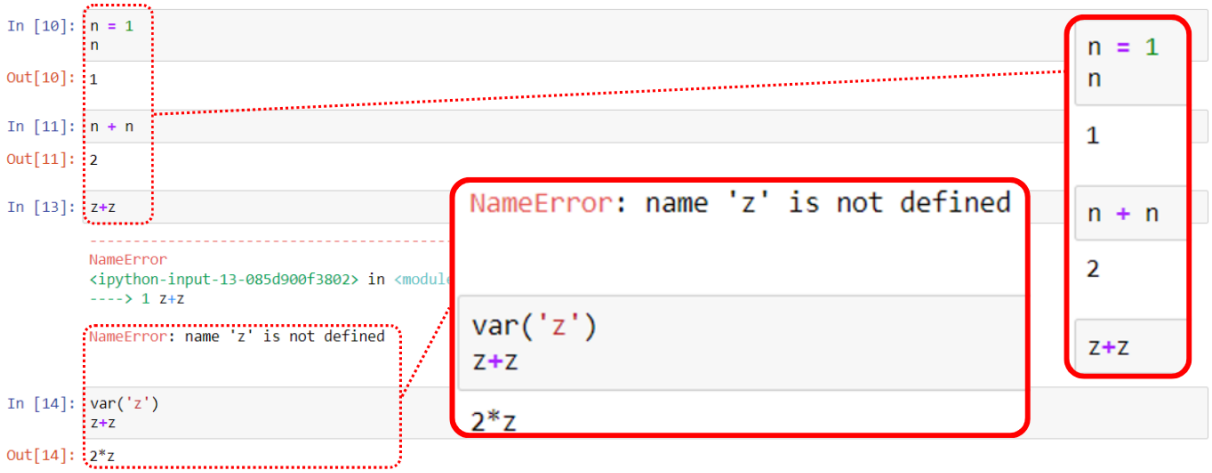
Fonte: O autor.

3.4 Variáveis no *SageMath*

Em linguagem de programação, a variável comum permite armazenar valores ou texto na memória do computador, enquanto as variáveis simbólicas são utilizadas em expressões algébricas para simplificar ou solucionar equações matemáticas. Por exemplo, se o usuário digitar `n = 1`, em seguida executar o comando `n + n`, aparecerá o valor 2 na saída, pois `n` (variável comum) assumiu o valor numérico 1.

Na manipulação de expressões algébricas, as variáveis assumem valores simbólicos ou indeterminados. No entanto, se forem utilizadas no código de qualquer maneira, acarretará em erro de sintaxe, informando que a variável ainda não está definida, consequentemente, impossibilitando a realização de operações algébricas. Ver Figura 3.6:

Figura 3.6: Variáveis comuns e simbólicas



Fonte: O autor.

Para o *SageMath* simplicar ou solucionar equações, as variáveis simbólicas devem ser definidas anteriormente com o comando `var(' ')`. Observe, ainda na Figura 3.6, que a variável `z` foi definida com o comando `var('z')` antes da operação algébrica `z+z`, produzindo como resultado `2z` na saída do comando. Além disso, o software também reconhece expressões escritas em função de variáveis já declaradas anteriormente no mesmo arquivo.

3.4.1 Expressões Algébricas no *SageMath*

A expressão algébrica da função horária do espaço - Movimento Uniformemente Variado (MUV), é uma expressão matemática que descreve a posição de um objeto em função do tempo.

$$s(t) = s_0 + v_0 t + \frac{at^2}{2}$$

$s(t)$ - posição no tempo t;

s_0 - posição inicial;

v_0 - velocidade inicial;

t - tempo;

a - aceleração.

Observe que $s(t)$ é dependente dos parâmetros s_0, v_0 e a e da variável t. Do ponto de vista computacional, para escrever a função horária do espaço no *SageMath*, não haverá distinção entre parâmetros e variáveis, de modo que será necessário definir as variáveis independentes antes da expressão algébrica no código.

Inclusive, pode-se declarar as variáveis num único comando `var`(‘s_0, v_0, a, t’). Ou seja, ao escrevê-las entre aspas (‘ ’) e separadas por vírgula (,), o usuário estará definindo todas as variáveis listadas como simbólicas. Além disso, $s(t)$ não precisa ser definida, pois depende apenas de variáveis já declaradas. Na Figura 3.7, está inserida a função horária do espaço no (MUV) no *SageMath*, e foi utilizado o comando `show`( ), que mostra a expressão textual da sentença matemática.

Figura 3.7: Declarando expressões algébricas

```
In [16]: var('s_0, v_0, a, t')
         s(t) = s_0 + v_0*t + (a*t^2)/2
         s(t)
Out[16]: 1/2*a*t^2 + t*v_0 + s_0

In [17]: show(s(t))
```

$$\frac{1}{2}at^2 + tv_0 + s_0$$

Fonte: O autor.

3.4.2 Notação de função no *SageMath*

O *Sage* reconhece automaticamente, em sua sintaxe, a incógnita x e seus termos dependentes como variáveis simbólicas, dispensando a necessidade de serem declaradas previamente. Na Figura 3.8, temos um exemplo com a variável y não declarada, em função da variável x.

Figura 3.8: Notação para funções no *SageMath*

```
In [21]: y
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/opt/sagemath-9.1/local/lib/python3.7/site-packages/sage/all_cmdline.py in
----> 1 y

NameError: name 'y' is not defined

In [22]: y = x^3 - 2*x

In [23]: y
Out[23]: x^3 - 2*x
```

Fonte: O autor.

A notação $f(x) =$ '*expressão algébrica na variável* x', não exige o uso do comando `var( )` para definir $f(x)$, como visto anteriormente. Em acréscimo, é possível substituir f e x por outras letras (variáveis), pois continuarão reconhecidas como variáveis simbólicas nesta notação. Ainda, o valor numérico da expressão, será acessado apenas substituindo o valor da variável independente por um valor real. Por exemplo, na Figura 3.9, para a função $g(a) = a^3$, $g(2)$ fornecerá 8 como resultado.

Figura 3.9: Valor numérico de uma função

```
In [24]: g(a) = a^3
         g
Out[24]: a |--> a^3

In [25]: g(2)
Out[25]: 8
```

Fonte: O autor.

As aplicações do *SageMath* apresentadas neste capítulo, trazem praticidade e interatividade no estudo das funções reais. Para a Educação Básica, sua implementação como ferramenta didática possibilita a manipulação dos parâmetros das funções e a visualização das alterações em tempo real nas respectivas representações gráficas, promovendo uma melhor compreensão do comportamento de cada função. A adoção de recursos tecnológicos na educação matemática, irá colaborar com a inclusão digital e com a aprendizagem ativa em preparação para o mundo do trabalho.

4 Metodologia e Objetivos

Esta pesquisa foi planejada sob a forma de projeto piloto para um estudo de caso e executada numa perspectiva qualitativa, justificada pela necessidade de observar e compreender as experiências de um grupo de alunos da rede pública, durante o aprendizado de funções reais e suas representações gráficas com a utilização do *SageMath*, aonde os conceitos introdutórios de uma linguagem de programação foram inseridos no ambiente natural da sala de aula. Esta abordagem em menor escala, permitiu a averiguar o processo didático da adapatação dos conteúdos na assimilação ativa desses conhecimentos em cada momento da pesquisa.

> [...] Os conteúdos escolares não são informações, fatos, conceitos, ideias etc.que sempre existiram na sua forma atual, registrada nos livros didáticos, nem são estáticos e definitivos. Os conteúdos vão sendo elaborados e reelaborados conforme as necessidades práticas de cada época histórica e os interesses sociais vigentes em cada organização social. O sentido histórico dos conteúdos se manifesta no trabalho docente quando se busca explicitar como a prática social de gerações passadas e das gerações presentes interveio e intervém na determinação dos atuais conteúdos, bem como o seu papel na produção de novos conhecimentos para o avanço da ciência e para o progresso social da humanidade. (LIBÂNEO, 1990, p. 137).

Embora a utilização de *softwares* matemáticos seja amplamente recomendada pelos documentos norteadores da educação básica, a maior parte das escolas públicas de Alagoas não possui laboratórios adequados ou computadores suficientes para a realização de aulas práticas, o que não favorece a inserção dos alunos no aprendizado digital nem a atualização dos professores no ensino com o uso das tecnologias educacionais. Por isso, a pesquisa apresenta natureza exploratória das funcionalidades educacionais do *SageMath*, sobretudo

associadas aos conceitos introdutórios de uma linguagem de programação.

> [...] Pode-se dizer que o computador está sendo usado para "*programar*" a criança. Na minha perspectiva, *é a criança que deve programar o computador* e, ao fazê-lo, ela adquire um sentimento de domínio sobre um dos mais modernos e poderosos equipamentos tecnológicos e estabelece um contato íntimo com algumas das ideias mais profundas da ciência, da matemática e da arte de construir modelos intelectuais. (PAPERT, 1985, p. 17-18).

Através de práticas pedagógicas inovadoras e da reelaboração de conteúdos, o Capítulo 5 apresenta uma sequência didática composta de instruções, códigos, exemplos e exercícios práticos, de modo a promover um melhor entendimento sobre as características do *software* na práxis educacional. Ao mesmo tempo, introduz a sintaxe básica da linguagem *Python*, na manipulação de algoritmos e parâmetros das funções, a fim de analisar e compreender o comportamento das funções matemáticas estudadas durante o ensino médio.

Assim, o objetivo principal da aplicação deste trabalho é promover a adoção do *SageMath* como ferramenta didática nas aulas de matemática do 1º ano do Ensino Médio, destacando seus benefícios no ensino de funções reais. Isso inclui investigar como o *software* impacta no desempenho dos alunos e na inclusão digital, bem como a introdução à linguagem *Python* contribui para o ensino. Também, avaliar a percepção dos alunos sobre o *SageMath*, sua facilidade de uso e motivação gerada, averiguar os desafios enfrentados por estudantes e professores nas questões técnicas e na adaptação dos conteúdos, analisar sua viabilidade na escola pública, considerando a disponibilidade de recursos e formação docente, e identificar benefícios pedagógicos do programa, como visualização gráfica e diferentes abordagens na resolução de problemas.

As etapas utilizadas na execução deste trabalho estão descritas no Capítulo 6 e foram divididas em três partes: a aplicação da sequência didáticas no decorrer de cinco aulas sobre Conjuntos, Equações e Sistema de Equações, Plotagem dos Gráficos de Funções, Função Exponencial e Função Logarítmica; a coleta de dados, na observação e acompanhamento do grupo na realização dos exercícios propostos e; por fim, a análise e discussão dos resultados obtidos.

5 Sequência Didática

5.1 Aula 1: Conjuntos no *SageMath*

5.1.1 Objetivos da aula

- Aprender a criar, manipular e visualizar conjuntos no *SageMath*;

- Revisar as propriedades dos conjuntos;

- Desenvolver o pensamento lógico e computacional na resolução de problemas.

5.1.2 Habilidades BNCC

(**EM13MAT405**) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

5.1.3 Motivação

O uso do *SageMath* nas aulas sobre conjuntos, não apenas simplifica a representação e a manipulação de elementos únicos ou agrupados, mas também permite a aplicação das operações entre conjuntos de maneira intuitiva e eficiente. Além disso, seja na resolução de problemas, na análise de dados com elementos distintos ou na implementação de algoritmos, a capacidade de explorar conjuntos por meio do *SageMath* pode acelerar a compreensão da teoria durante sua aplicação nas aulas práticas. Veja os

códigos utilizados nesta aula na Tabela 5.1:

Tabela 5.1: Lista de códigos utilizados na aula 1

Códigos e métodos	**Descrição**
`set`([ ])	Cria um novo conjunto, passando uma lista [ ] de elementos como argumento. Se a lista estiver vazia, o conjunto criado também estará vazio.
`.union`( )	Fornece a união entre dois conjuntos. O resultado é um novo conjunto que contém todos os elementos dos dois conjuntos originais, sem duplicações.
`.intersection`( )	Fornece a interseção de dois conjuntos. O resultado é um novo conjunto que contém apenas os elementos que estão presentes em ambos os conjuntos originais.
- ou `.difference`( )	Fornece a diferença entre dois conjuntos. O resultado é um novo conjunto que contém apenas os elementos que estão presentes no primeiro conjunto, mas não no segundo.
`.issubbset`( )	Verifica se um conjunto é subconjunto de outro. O resultado será um valor *booleano*, *True* se o conjunto for um subconjunto e *False* se não for.
`factor`( )	Fornece a fatoração de um número inteiro como produto de números primos; ou uma expressão algébrica como produto de expressões de menor grau.
`prime_divisors`( )	Fornece os divisores primos de um número inteiro ou de um elemento do anel de inteiros

Fonte: O autor.

5.1.4 Atividades Propostas

Inicialmente, traremos a sintaxe e exemplos com alguns comandos utilizados no *SageMath* para interagir com conjuntos. Em seguida, aplicaremos estes códigos na resolução de problemas no ambiente do *software*.

Para representar um conjunto, usamos chaves $\{elementos\ do\ conjunto\}$ e, entre as chaves, escrevemos seus elementos separados por vírgulas. Por exemplo, na notação matemática corrente, o conjunto A das vogais é representado por $A = \{a, e, i, o, u\}$. Entretanto, no *SageMath* iremos utilizar colchetes [ ] (operador de lista em *Python*) para escrever estes elementos e o comando `set( )` para atribuir estes elementos à variável A. Então, por que não criar uma lista $A =$ [ ] ao invés de $A =$ `set([ ])`? Porque, diferentemente de uma lista em *Python*, o comando `set( )` elimina os elementos repetidos, uma vez que estes elementos são considerados como um só, independentemente da ordem que foram inseridos. Além disso, também podemos verificar a relação de pertinência, se um elemento está contido ou não no conjunto, através do operador `in`. Por exemplo, ao realizar o comando: "`elemento`" `in` "`conjunto`", verifica-se que quando o elemento estiver contido a saída no *Sage* será `True` (verdade em inglês), caso contrário, `False`. Observe a Figura 5.1:

Figura 5.1: Função `set( )` e o operador *in*.

```
In [2]: A = set([0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]) # Definimos o conjunto A;

In [3]: A # Ao chamarmos o conjunto A o elemento repetido "1" será considerado como único;
Out[3]: {0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144}

In [4]: 1 in A # Verificando a pertinência do elemento "1";
Out[4]: True

In [5]: 4 in A # Verificando a pertinência do elemento "4";
Out[5]: False
```

Fonte: O autor.

Armazenar códigos em variáveis, "`variável`" = `set([elementos do conjunto])`, traz praticidade e economia de tempo ao reutilizar ou consultar algo relacionado a estes dados. Vejamos como as operações de `união`, `intersecção` e `diferença` entre conjuntos

podem ser executadas apenas utilizando as variáveis onde os conjuntos foram definidos e armazenados no *SageMath*.

Dados os conjuntos $B = \{1, 3, 5\}$ e $C = \{2, 3, 4\}$, declarados nas variáveis B e C, representaremos a relação de *união* $(B \cup C)$ com o código `B.union(C)`. Ver Figura **??**.

Figura 5.2: Função `set( )` e o código `.union`.

```
In [7]: B = set([1,3,5]) # Armazena o conjunto B = {1,3,5} na variável B, definido com o comando set([])
        B # Mostra os elementos do conjunto B.
Out[7]: {1, 3, 5}

In [8]: C = set([2,3,4]) # Armazena o conjunto C = {2,3,4} na variável C, definido com o comando set([])
        C # Mostra os elementos do conjunto C.
Out[8]: {2, 3, 4}

In [9]: B.union(C) # Mostra os elementos da relação B U C
Out[9]: {1, 2, 3, 4, 5}
```

Fonte: O autor.

Dessa forma, após armazenados em B e C, os conjuntos definidos podem ser acessados citando as variáveis em todo o arquivo. Simultaneamente, as relações de *interseção* $(B \cap C)$, *diferença* $(A \backslash B$ ou $A - B)$ e de *inclusão* $(B \subset C)$, são verificadas com os códigos `B.intersection`(C), `B.difference`(C) ou `B - C`; e `B.issubset`(C), respectivamente. Ver Figura **??**.

Figura 5.3: Códigos `.intersection( )`, `.difference( )` e `.issubset( )`.

```
In [10]: B.intersection(C) # Mostra os elementos da relação B n C;
Out[10]: {3}

In [11]: B.difference(C) # Mostra os elementos da relação B - C;
Out[11]: {1, 5}

In [12]: B - C # Mostra os elementos da relação B - C de modo mais simplificado;
Out[12]: {1, 5}

In [13]: B.issubset(C) # Verifica se B é um subconjunto de C.
Out[13]: False
```

Fonte: O autor.

Uma das vantagens da linguagem de programação *Python*, é a possibilidade de automatizar ações repetitivas com um *loop* (laço de repetição). O *loop* é uma estrutura

de controle de fluxo que permite a execução de um bloco de código várias vezes.

O comando [“`expressão`” `for` “`variável da expressão`” `in` [“`início`”..“`fim`”]], é um *loop* que vai mostrar uma lista com os valores da “`expressão`”, quando a “`variável`” assumir cada valor inteiro no intervalo [“`início`”, “`fim`”]. Exemplo:

```
[3*x for x in [1..5]] = [3, 6, 9, 12, 15]
```

O código acima criou uma lista com os 5 primeiros múltiplos de 3. Também podemos alterar a expressão e o intervalo desejado para gerar listas personalizadas, como mostrado na Figura 5.4:

Figura 5.4: Laço `for` - Estrutura de repetição em *Python*

```
In [1]: [3*x for x in [1..5]]
Out[1]: [3, 6, 9, 12, 15]

In [2]: [x^2 for x in [1..10]]
Out[2]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

In [3]: [(x,x^2) for x in [-3..3]]
Out[3]: [(-3, 9), (-2, 4), (-1, 1), (0, 0), (1, 1), (2, 4), (3, 9)]
```

Fonte: O autor.

5.1.5 Exercícios da aula 1

1. Utilize o comando `set( )` para definir o conjunto $A = \{7x \ / \ x \in \mathbb{N} \text{ e } 1 \leq x \leq 10\}$.

2. Sejam B e C, respectivamente, os conjuntos dos divisores primos de 30 e de 42. Verifique:

   (a) $B \cap C$;

   (b) $B \cup C$;

   (c) $B \setminus C$.

3. Verifique se para os conjuntos $A = \{1, 2, 3\}$ e $B = \{2, 4, 6\}$ as afirmações abaixo são verdadeiras:

   (a) $(A \cup B) = (B \cup A)$;

   (b) $(A \cap B) = (B \cap A)$;

   (c) $(A \setminus B) = (B \setminus A)$.

4. Utilize o comando `set( )` para determinar os conjuntos A, B e C de modo que: $A = \{x \in \mathbb{N} \ / \ 1 \leq x \leq 4\}$; $B = \{y \in \mathbb{N} \ / \ -3 \leq x \leq 6\}$ e $C = \{z \in \mathbb{N} \ / \ 3 \leq x < 8\}$;

5. O comando `A = set([x^2 for x in [1,11]])`, constrói o conjunto $A = \{x^2 \ / \ x \in \mathbb{Z} \text{ e } 1 \leq x < 11\}$, ou seja, $A = \{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$.

   (a) Analisando o comando que construiu o conjunto A com os 10 primeiros números quadrados perfeitos, construa o conjunto B com os números da forma 2^{2x+1} para $x \in \mathbb{N}$ no intervalo $[1, 5]$.

   (b) Coloque um número no comando "`.... in B`" de modo que o resultado na saída seja `True`.

5.2 Aula 2: Equação, Sistema de Equações e Expressões Algébricas no *SageMath*

5.2.1 Objetivos da aula

- Aprender a usar, solucionar e manipular equações no *SageMath*;

- Revisar propriedades e conceitos básicos sobre funções polinomiais;

- Desenvolver o pensamento lógico e computacional na resolução de problemas.

5.2.2 Habilidades BNCC

(**EM13MAT302**) Construir modelos empregando as funções polinomiais de 1º ou 2º graus, para resolver problemas em contextos diversos, com ou sem apoio de tecnologias digitais.
(**EM13MAT405**) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

5.2.3 Motivação

O estudo das equações é fundamental para uma compreensão mais ampla da utilização dos conceitos algébricos, tanto na resolução de problemas quanto na modelagem matemática de situações do cotidiano.

No ambiente do *SageMath*, o usuário pode visualizar graficamente soluções, manipular variáveis, aplicar operações algébricas e realizar análises paramétricas de maneira eficiente. Neste processo, o desenvolvimento de habilidades computacionais assumirá grande relevância, se estendendo às aplicações práticas e tecnológicas do mundo real. Veja os códigos utilizados nesta aula na Tabela 5.2.

Tabela 5.2: Lista de códigos utilizados na aula 2

Códigos	Descrição
`var( )`	Declara variáveis simbólicas.
`solve( )`	Fornece as soluções de equações e sistemas de equações.
`show( )`	Exibe a equação recebida como argumento em formato de expressão algébrica no LaTeX.
`.factor( )`	fatora um número inteiro como produto de números primos ou uma expressão algébrica como produto de expressões de menor grau.
`.expand( )`	Fornece a expansão de expressões em variáveis simbólicas, como produtos, somas, potências e etc.
`.lhs( )`	Fornece o lado esquerdo da igualdade entre duas expressões algébricas.
`.rhs( )`	Fornece o lado direito da igualdade entre duas expressões algébricas.
`point( )`	Mostra a representação gráfica de um ponto no plano ou no espaço. A representação pode ser especificada como uma lista de pontos.

Fonte: O autor.

5.2.4 Atividades Propostas

De modo análogo ao apresentado na aula 1, traremos a sintaxe e exemplos de alguns comandos utilizados no *Sage* para interagir com as equações. Em seguida, aplicaremos estes códigos para a resolução de problemas no ambiente do *software*.

Assim como vimos no Capítulo 3, a variável x não precisa ser declarada para que o programa interprete-a como uma variável simbólica. Mas, em outros casos, as variáveis devem ser definidas anteriormente com o comando `var( )`. Outro fato importante para a interação com funções no *SageMath*, é que os sinais `"="` e `"=="` possuem significados diferentes. O sinal único `"="` é usado para atribuir números, *strings*, expressões e equações a uma variável, enquanto o duplo `"=="`, para a igualdade entre duas sentenças matemáticas. Assim, permitindo verificar se duas expressões são matematicamente iguais e realizar operações como simplificação, derivação, integração e resolução simbólica de equações. Observe a Figura 5.5:

Figura 5.5: Exemplos com os operadores "=" e "==".

```
In [6]: var('y')
        x=5; y==5
Out[6]: y == 5

In [8]: type(x) # O valor 5 foi atribuído à x, então x é visto como um número inteiro no Sage.
Out[8]: <class 'sage.rings.integer.Integer'>

In [9]: type(y) # Mesmo que y seja igual a 5, ainda será vista como uma variável simbólica no software.
Out[9]: <class 'sage.symbolic.expression.Expression'>
```

Fonte: O autor.

Como visto no exemplo acima, temos apresentação dos operadores de igualdade na variável comum `x = 5`, e na variável simbólica e `y == 5`. Note que em `x = 5`, a variável comum recebe o número 5, tornando-se um número inteiro. Mas, em `y == 5`, y foi declarada como variável simbólica e comparada ao valor 5, ou seja, possui o valor 5 mas continua sendo uma expressão simbólica. Assim, o sinal duplo de igualdade será necessário para a manipulação e resolução de equações no *SageMath*.

No *Sage*, é possível resolver equações de modo rápido e prático. Para que o software nos forneça as soluções de uma ou mais equações, basta utilizar o comando `solve([“equação ou sistema de equações”], “variáveis”)`. Dessa forma, as soluções serão armazenadas e mostradas em uma lista[ ], como ilustrado na Figura 5.6.

Figura 5.6: Função `solve( )` na resolução de equações e sistemas.

```
In [72]: solve(x-3 == 5, x) # Mostra uma lista com a solução da equação x - 3 = 5 para a variável x;
Out[72]: [x == 8]

In [73]: solve(x^2+3 == 28, x) # Mostra uma lista com as soluções da equação x^2 + 3 = 28 para a variável x;
Out[73]: [x == -5, x == 5]

In [74]: var('x y') # Declara as variáveis x e y;
         solve([x + y == 5, x - y == 1], x, y) # Mostra uma lista com a solução do sistema;
Out[74]: [[x == 3, y == 2]]
```

Fonte: O autor.

Note que independentemente de existirem uma ou mais soluções para a equação, como dito no parágrafo anterior, elas serão apresentadas em forma de lista. Contudo, pode-se utilizar colchetes como operador `[‘índice’]` para acessar um único elemento da

lista, ou seja, uma solução específica. Por exemplo, na lista $A = [1, 2, ..., i, ..., n-1, n]$, a posição do elemento i está definida pelo índice $i - 1$, o operador `[i-1]` acessa o i-ésimo elemento da lista. Logo, é possível acessar o primeiro elemento com `A[0]`, o segundo elemento com `A[1]`, e assim por diante até o último elemento `A[n - 1]`.

Para $x^4 - 16 = 0$, o comando `solve(x**4 - 16 == 0, x)` mostra as quatro soluções da equação na lista `[x = = 2i, x = = -2, x = = -2i, x = = 2]`. Ao acessá-las pelo índice, teremos no lado esquerdo a variável e no lado direito o valor numérico. Veja a Tabela 5.3.

Tabela 5.3: Relação entre os operadores com índice e as soluções de $x^4 - 16 = 0$.

Entrada: operador com índice	**Saída**	**Lado esquerdo**	**Lado direito**
`A[0]`	`x == 2i`	`x`	`2i`
`A[1]`	`x == -2`	`x`	`-2`
`A[2]`	`x == -2i`	`x`	`-2i`
`A[3]`	`x == 2`	`x`	`2`

Fonte: O autor.

Também, podemos acessar um dos lados das soluções. O método `.lhs( )`, para o lado esquerdo da solução; e `.rhs( )`, para o lado direito. Por exemplo, na tabela 5.3 a saída de `A[3]` é `x == 2`, a saída de `A[3].lhs( )` é `x`; e a saída de `A[3].rhs( )` é `2`. Veja a Figura 5.7.

Figura 5.7: Função `solve( )`, operador de índice [ ], `.lhs( )` e `.rhs( )`.

```
In [4]: A = solve(x^4-16 == 0, x) # Armazena a lista com a solução do sistema;
        A
Out[4]: [x == (2*I), x == -2, x == (-2*I), x == 2]

In [5]: A[3] # Acessa o quarto elemento da solução na lista.
Out[5]: x == 2

In [6]: A[3].lhs() # Acessa apenas o lado esquerda da solução x = 2;
Out[6]: x

In [7]: A[3].rhs() # Acessa apenas o lado direito da solução x = 2;
Out[7]: 2
```

Fonte: O autor.

No *Python*, (linguagem do *SageMath*), é possível programar com conceitos de Programação Orientada a Objetos (POO). Isso significa que os objetos criados estão

bem definidos e isolados. Portanto, o código permite reutilização e extensibilidade, economizando tempo e tornando o código mais flexível e adaptável. É o caso da utilização de métodos como `.lhs( )` e `.rhs( )` na Figura 5.7, que acessam os atributos de `A`, lista de soluções criadas com o comando `solve( )`.

5.2.5 Exercícios da aula 2

1. Após declarar as variáveis a, b e c com `var('a b c')`, com o comando `solve( )`, verifique que para a, b e c reais:

   (a) A equação da forma $ax + b = 0$ possui solução: $x = \dfrac{-b}{a}$;

   (b) A equação da forma $ax^2 + bx + c = 0$ possui solução: $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

2. Dada a equação $x^3 + x - 2 = 0$.

   (a) Armazene as soluções da equação na variável `S`;

   (b) Execute o comando `S`, "variável que armazenou a função `solve( )`". Em seguida, o comando `show(S)`;

   (c) Mostre apenas a raiz real através de seu índice no operador `[ ]`.

3. A fatoração e expansão de expressões algébricas são técnicas essenciais para a observação do comportamento das funções e resolução de problemas. Então:

   (a) Associe a expressão algébrica $x^3 - 2x^2 - 5x + 6$ a variável comum `eq1`. Execute o método `.factor( )` em `eq1` para mostrar a forma fatorada da expressão.

   (b) Associe a expressão algébrica $(x + 2)(x - 1)(x - 3)$ a variável comum `eq2`. Execute o método `.expand( )` em `eq2` para mostrar a forma expandida da expressão.

   (c) Utilize o método `.rhs( )` em `eq1` para mostrar que $x_1 + x_2 + x_3 = 2$.

4. Dado o sistema de equações: $\begin{cases} x + y &= 6 \\ x - y &= 1 \end{cases}$

   (a) Resolva o sistema com o comando `solve( )`;

(b) Associe a variável `P` à solução do sistema.

(c) Execute o comando: `point(P, xmin = -5, xmax = 5, ymin =- 5, ymax = 5)`.

Observação: O item (c) traz o primeiro contato com o plano cartesiano através da função `point( )`, que fornece a representação gráfica de pontos no plano cartesiano. Além disso, os parâmetros: `xmin`, `xmax`, `ymin` e `ymax` estão inseridos na função, personalizando o gráfico para atender às necessidades de visualização do usuário. Tais parâmetros, serão abordados com maiores detalhes no decorrer da próxima aula.

5.3 Aula 3: Introdução à plotagem de gráficos no *SageMath*

5.3.1 Objetivos da aula

- Aprender a usar a função `plot( )` e seus parâmetros no *SageMath*;

- Analisar as propriedades das funções e equações, como: sinal, raízes, pontos de máximo, mínimo e interceptações;

- Desenvolver a capacidade de criar gráficos claros e informativos para analisar e solucionar problemas.

5.3.2 Habilidades BNCC

(**EM13MAT302**) Construir modelos empregando as funções polinomiais de 1º ou 2º graus, para resolver problemas em contextos diversos, com ou sem apoio de tecnologias digitais.
(**EM13MAT405**) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

5.3.3 Motivação

Estudar a representação gráfica das funções polinomiais no *SageMath*, é uma maneira prática e eficiente de se aprofundar na compreensão dos conceitos e da teoria, construindo habilidades de modelagem e visualização do comportamento das funções, além de estabelecer uma base sólida para tópicos mais avançados em matemática e na ciência da computação. Veja os códigos utilizados nesta aula na Tabela 5.4.

Tabela 5.4: Lista de códigos utilizados na aula 3

Códigos	Descrição
`plot( )`	Cria gráficos de funções, conjuntos de pontos dentre outras representações gráficas no plano cartesiano.
`-x` e `x` ou `xmin` e `xmax`	Estes parâmetros definem o intervalo de plotagem do gráfico no eixo das abcissas.
`ymin` e `ymax`	Estes parâmetros definem o intervalo de plotagem do gráfico no eixo das ordenadas.
`title`	Este parâmetro define o título do gráfico.
`legend_label`	Este parâmetro define as etiquetas de cada curva, ou seja, as legendas dos gráficos.
`axes_labels`	Este parâmetro define os rótulos dos eixos no plano cartesiano.
`point( )`	Fornece a representação gráfica de um ponto no plano ou no espaço. A representação pode ser especificada como uma lista de pontos.
`text( )`	Esta função é usada para adicionar texto a um gráfico. Ela recebe como entrada um texto e uma posição (x, y) no plano cartesiano. Ainda, é possível especificar a fonte, o tamanho e a cor do texto.
`size`	Este parâmetro especifica o tamanho do ponto no gráfico.
`color`	Este parâmetro define a cor da representação gráfica.

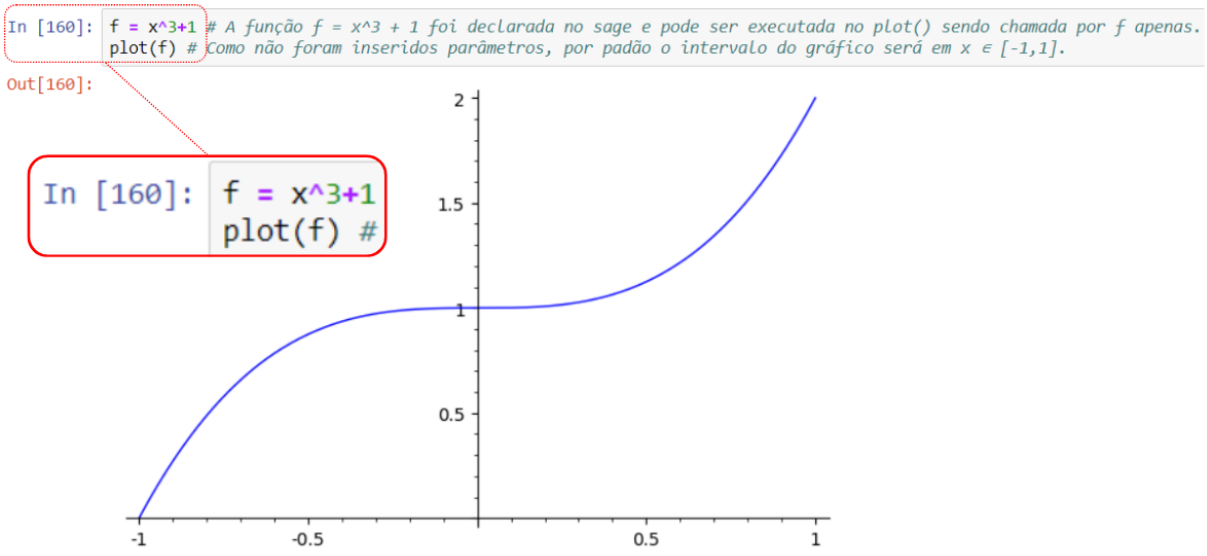
Fonte: O autor.

5.3.4 Atividades Propostas

Na aula anterior, através da função `point( )`, tivemos a visualização do plano cartesiano no *software*, pensonalizando o gráfico com a utilização dos parâmetros: `xmin`, `xmax`, `ymin` e `ymax`, no corpo do comando. Semelhantemente ao comando `point( )`, a função `plot( )` retorna um objeto gráfico e aceita uma variedade de argumentos e parâmetros, tais como: os intervalos de plotagem no eixo das abcissas x (domínio) e no eixo das coordenadas y (imagem), rótulos, legendas e outros efeitos visuais, incluindo a concatenação de outras funções no mesmo plano cartesiano.

Dada uma função $f : \mathbb{R} \to \mathbb{R}$, o código `plot(f)`, por padrão, plotará o gráfico de f no intervalo $x \in [-1, 1]$. Veja na Figura 5.8, o exemplo abaixo para $f(x) = x^3 + 1$.

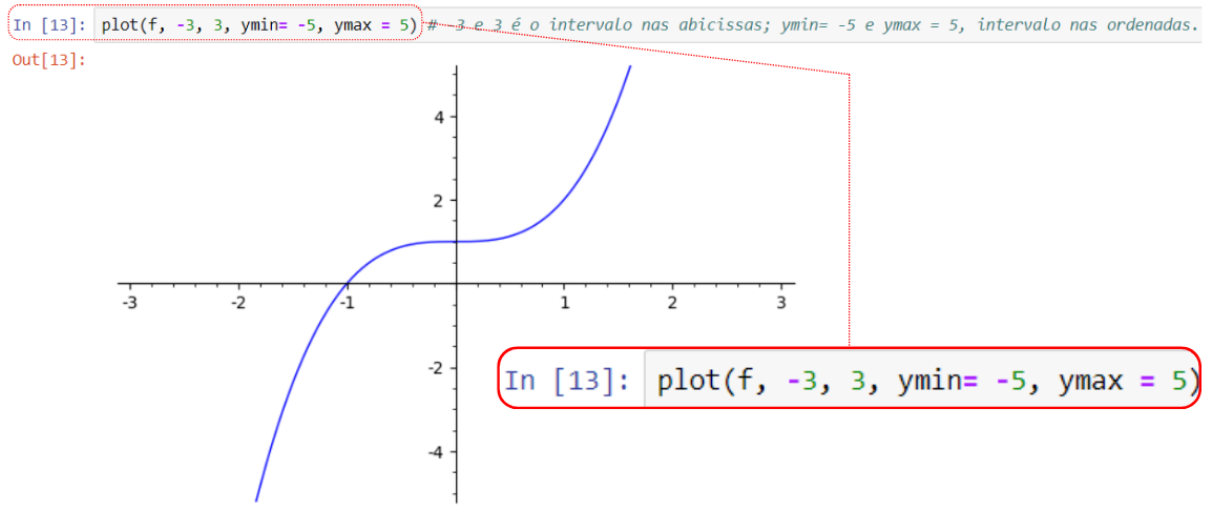
Figura 5.8: Função `plot(f)` para $f(x) = x^3 + 1$.



Fonte: O autor.

Com os parâmetros `xmin` e `xmax` para o domínio; `ymin` e `ymax` para a imagem da função, podemos alterar o intervalo de apresentação do gráfico no plano cartesiano. Assim, separados por vírgula e com seus respectivos valores, estes parâmetros personalizam a vizualização e análise do gráfico da função. Veja a Figura 5.9.

Figura 5.9: Parâmetros `-x`, `x`, `ymin` e `ymax` na função `plot`(f).

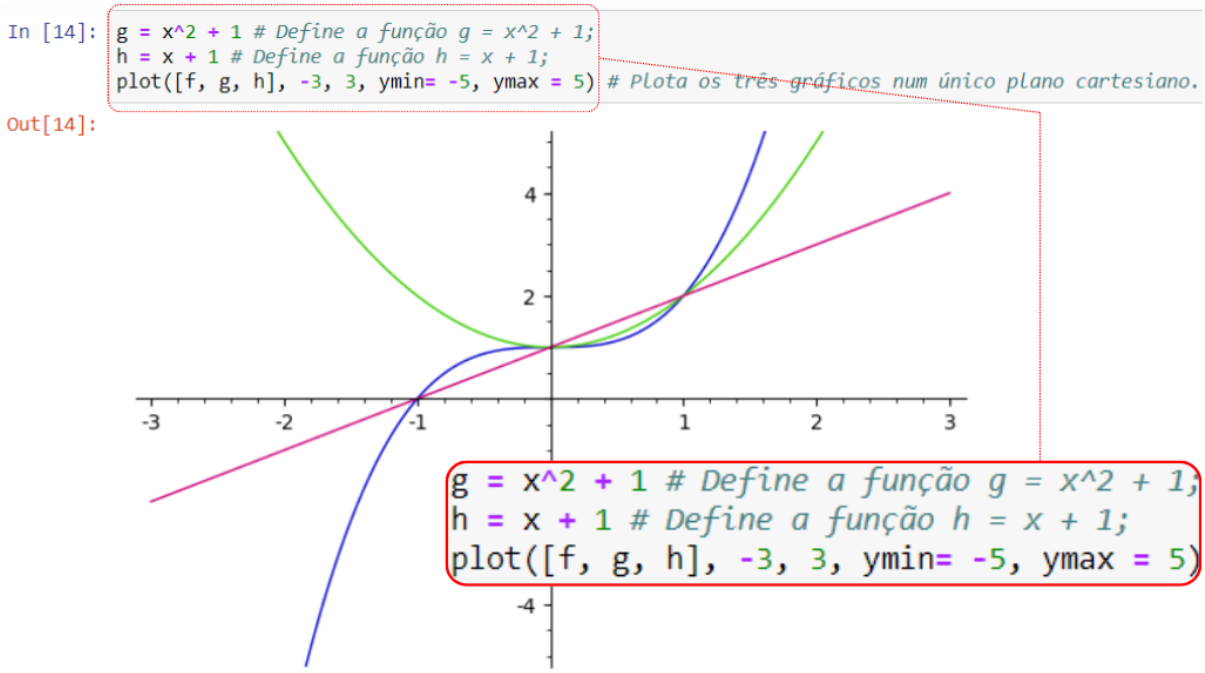


Fonte: O autor.

Note que os parâmetros `xmin` e `xmax`, podem ser substituídos diretamente por valores numéricos no corpo da função `plot( )`.

Para efeito de comparação entre diferentes gráficos, é possível inserir uma lista de funções no comando `plot( )`. Assim, podemos visualizar as representações gráficas simultaneamente. Por exemplo, sejam $f(x)$ da Figura 5.8, $g(x) = x^2 + 1$ e $h(x) = x + 1$; o comando `plot([f, g, h], -3, 3, ymin = -5, ymax = 5)` plota os gráficos das funções f, g e h, no mesmo plano cartesiano. Veja a Figura 5.10.

Figura 5.10: Função `plot([f, g, h])`.



Fonte: O autor.

O parâmetro `title` é usado para criar um título para o gráfico, enquanto `legend_label` permite adicionar legendas às funções representadas; e `axes_labels` insere rótulos aos eixos x e y. É importante observar que o texto inserido no código deve ser colocado entre aspas simples (’), mesmo ao optar pela notação latex. Por exemplo:

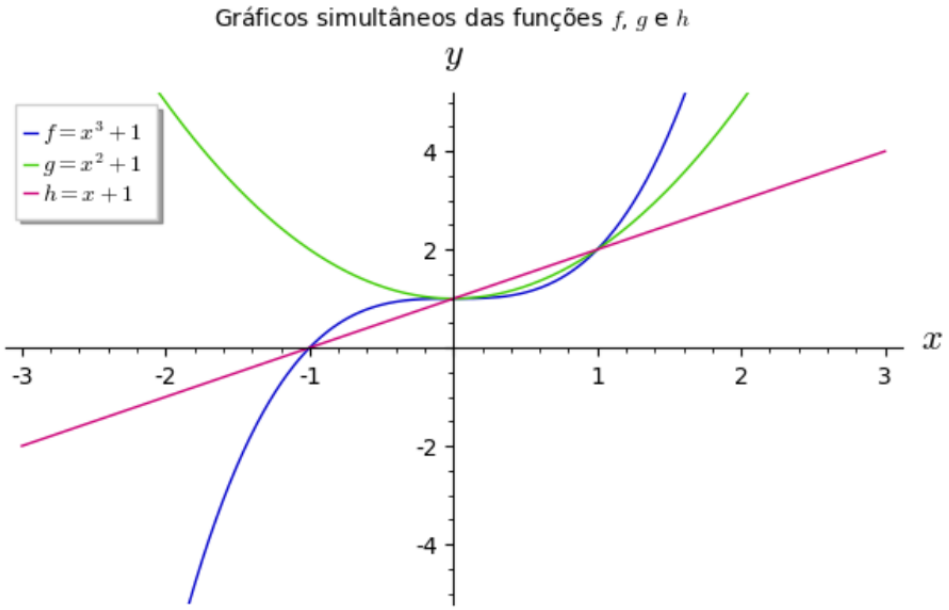
```
title = ‘Gráficos simultâneos das funções $f$, $g$ e $h$’;

legend_label = [‘$f = x^3 + 1$’, ‘$g = x^2 + 1$’, ‘$h = x + 1$’];

axes_labels=[‘$x$’, ‘$y$’]).
```

Desse modo, os parâmetros `title`, `legend_label` e `axes_labels` inserem elementos textuais ao gráfico, com o caracter $ (cifrão) diferenciando o texto matemático do normal. Ver Figura 5.11.

Figura 5.11: Parâmetros `title`, `legend_label` e `axes_labels` em `plot`([f, g, h])

```
In [9]: plot([f, g, h], -3, 3, ymin= -5, ymax = 5,
        title = 'Gráficos simultâneos das funções $f$, $g$ e $h$ \n',
        legend_label=['$f=x^3 + 1$', '$g=x^2 + 1$', '$h=x + 1$'], axes_labels=['$x$', '$y$'])
```

Out[9]:



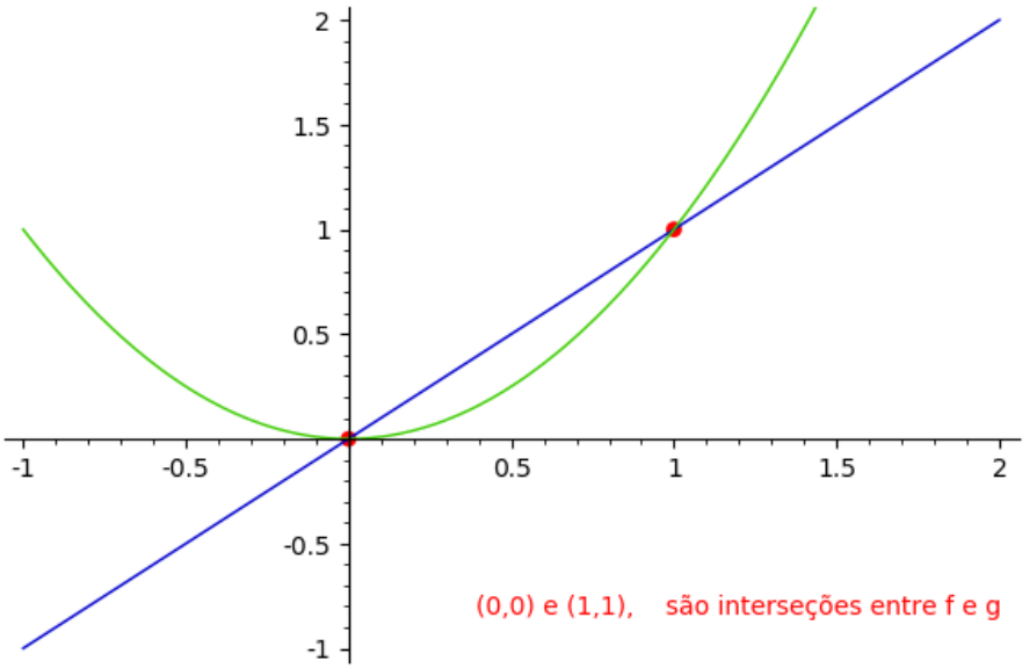
Fonte: O autor.

Oportunamente, o software também dispõe da possibilidade de concatenar comandos gráficos no mesmo plano cartesiano. Com o operador + antes de comandos como `point( )` ou `text( )`, podemos acrescentar novos elementos ao plano cartesiano já existente. Veja a Figura 5.12.

Figura 5.12: Concatenação entre os comandos `point( )`, `text( )` e `plot( )`.

```
In [2]: f = x; g = x^2 # Armazena x na variável f e x^2 na variável g;
        A = plot([f,g], -1, 2, ymax = 2) # Armazena os gráficos de f e g em A;
        A + point([(0,0),(1,1)], color='red', size = 40) + text('(0,0) e (1,1),\
            são interseções entre f e g', (1.2,-0.8), color='red')
        # funções plot(), point() e text(); parâmetros color e size.
```

Out[2]:



Fonte: O autor.

A construção do aprendizado e a familiarização com os comandos de plotagem no *SageMath*, acontecerá de modo natural e gradativo, de acordo com as necessidades de personalização e criatividade de cada usuário. Consequentemente, será na interatividade com os parâmetros dos comandos do software que acontecerá a análise dos gráficos e a compreensão do comportamento de cada função.

5.3.5 Exercícios da aula 3

1. Com o comando `plot( )`, construa:

   (a) O gráfico de f, uma função do 1^{o} grau para x no intervalo $[-3, 3]$, que intersecte os eixos do plano cartesiano em $(-1, 0)$ e $(0, 1)$.

   (b) O gráfico de g, uma função do 2^{o} grau para x no intervalo $[-1, 5]$ que intersecte os eixos do plano cartesiano em $(0, 3)$, $(1, 0)$ e $(3, 0)$.

Observação:

Após finalizar a resposta do item a, acrescente o comando:

```
+ point([(-1,0), (0,-1)], size=30, color='red');
```

Após finalizar a resposta do item b, acrescente o comando:

```
+ point([(0,6),(1,0), (3,0)], size=30, color='red').
```

2. Dado o sistema de equações:

$$\begin{cases} x - y & = 2 \\ 2x + 3y & = 14 \end{cases}$$

   (a) Mostre a representação gráfica das duas retas no mesmo plano cartesiano;

   (b) Indique a solução do sistema com comandos `point( )` e `text( )`;

   (c) Acrescente os parâmetros `title`, `legend_label` e `axes_labels` ao gráfico.

3. Seja a função $f(x) = -x^2 + 3x + 1$.

   (a) Use o comando `solve( )` para encontrar as raízes de f;

   (b) Escreva um código que armazene e mostre as coordenadas do vértice de f na variável V;

   (c) Plote o gráfico de f e concatene com o comando `point( )`, de modo que evidencie as raízes e o vértice da função. "utilize os parâmetros `size = 30` e `color ='red'` em `point( )`".

4. Considere uma função $f : \mathbb{R} \to \mathbb{R}$ dada pela expressão $f(x) = -x^2 + bx + c$, com b e c reais, cujo gráfico possui eixo de simetria na reta $x = 1$ e a diferença entre as raízes seja igual a -4. Construa o gráfico que representa f.

5.4 Aula 4: Função Exponencial no *SageMath*

5.4.1 Objetivos da aula

- Praticar a análise e representação gráfica das funções exponenciais no *SageMath*;

- Revisar conceitos básicos das funções exponenciais e suas propriedades;

- Desenvolver o pensamento computacional na resolução de problemas.

5.4.2 Habilidades BNCC

(**EM13MAT403**) Analisar e estabelecer relações, com ou sem apoio de tecnologias digitais, entre as representações de funções exponencial e logarítmica expressas em tabelas e em plano cartesiano, para identificar as características fundamentais (domínio, imagem, crescimento) de cada função.
(**EM13MAT405**) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

5.4.3 Motivação

A abordagem interativa do *SageMath*, permite investigar propriedades, traçar gráficos e realizar cálculos mais complexos relacionados à função exponencial, propiciando aos alunos uma compreensão teórica mais profunda, associada à aplicação prática dos conceitos estudados.

Também, promove o desenvolvimento de habilidades analíticas na observação do comportamento das funções e na construção de suas representações gráficas, utilizando as funcionalidades do *software* em conjunto com conceitos introdutórios da linguagem de programação *Python*. Os códigos utilizados nesta aula, encontram-se na Tabela 5.5.

Tabela 5.5: Lista de códigos utilizados na aula 4

Códigos	Descrição
`factor( )`	Esta função é usada para fatorar um número inteiro em um produto de números primos ou uma expressão algébrica em um produto de expressões mais simples.
`ticks`	Este parâmetro controla a distância entre os `ticks`. O valor padrão é 1, o que significa que haverá um `tick` para cada número inteiro nos eixos cartesianos.
`tick_formatter`	Este parâmetro permite a formatação de valores numéricos nos `ticks` em forma de texto. Inclusive, em notação latex.
`gridlines`	Este parâmetro insere uma malha quadriculada no plano cartesiano. As entradas `minor`, `normal` e `major`, inserem malhas com espaçamentos diferentes.

Fonte: O autor.

5.4.4 Atividades Propostas

A decomposição dos naturais em fatores primos, em conjunto com a utilização das propriedades da potenciação, são ferramentas essenciais para a resolução de equações exponenciais. No *SageMath*, o comando `factor( )`, anteriormente utilizado como `.factor( )`, porém a mesma funcionalidade, recebe um número ou expressão como entrada e retorna o produto de seus fatores primos. Na Figura 5.13, veja exemplos da fatoração numérica e algébrica no *software*.

Figura 5.13: Fatoração em primos com a função `factor( )`.

```
In [8]: factor(1350)
Out[8]: 2 * 3^3 * 5^2

In [9]: factor(2*x^3 + 3*x^2 - 2*x - 3) # 2x^3 + 3x^2 - 2x - 3
Out[9]: (2*x + 3)*(x + 1)*(x - 1)
```

Fonte: O autor.

Na prática da sala de aula, os alunos fazem a decomposição em ambos os lados da equação exponencial. Assim, para auxiliar o processo de aprendizado, podemos realizar este método no *SageMath*. Por exemplo, simularemos o modo de resolução para a equação $8^x = 64$ no software, utilizando o comando `factor( )` e inserindo os devidos comentários

em cada linha código, como ilustra a Figura 5.14. Veja também, a resolução da equação com o comando `solve( )`.

Figura 5.14: Resolução de equações exponenciais com a função `factor( )`.

```
In [12]: 8^x == 64; factor(8)
Out[12]: 2^3

In [13]: factor(64)
Out[13]: 2^6

In [14]: # 3x = 6, x = 2. Agora, veremos com o comando solve().
         solve(8^x == 64, x)
Out[14]: [x == 2]
```

Fonte: O autor.

Objetivamente nesta aula, o uso do comando `factor( )` apenas auxilia na compreensão da solução da equação exponencial, uma vez que `solve( )` nos fornece a resposta de modo mais prático e direto. Portanto, a escolha entre os comandos citados depende do contexto educacional de sua aplicação.

Agora, acrescentando funcionalidade ao uso do comando `factor( )`, apresentaremos parâmetros que tornarão os gráficos mais informativos. Assim, os parâmetros `ticks`, `tick_formatter` e `gridlines` permitem o ajuste dos valores exibidos nos eixos cartesianos e a formatação desses valores de acordo com as necessidades específicas da análise, tornando a representação gráfica mais acessível e interpretável.

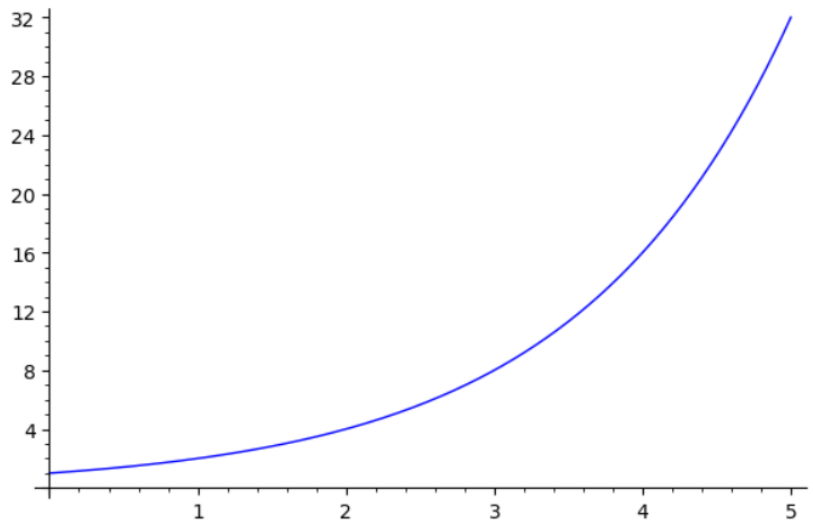
O parâmetro `ticks` é responsável por determinar os marcadores numerados em cada eixo do plano. Por exemplo, `ticks = [x, y]`, onde o valor numérico de `x` determina os valores múltiplos que serão mostrados no eixo das abcissas; e `y`, de modo análogo, no eixo das ordenadas.

As funções exponenciais crescem ou decrescem rapidamente em um dos eixos no plano cartesiano. Assim, os *ticks* numerados em todos os marcadores podem causar um excesso de informação ao gráfico. Então, no exemplo da Figura 5.15, alteramos a apresentação o eixo y com o parâmetro `ticks = [1, 4]`, variando os valores mostrados do eixo y aos múltiplo de 4.

Figura 5.15: Parâmetro `ticks` em esacalas na função `plot( )`.

```
In [8]: plot(2^x, 0, 5, ymin=0, ticks=[1, 4])
        # A escala em x é mostrada de 1 em 1;
        # A escala em y é mostrada de 4 em 4.
```

Out[8]:



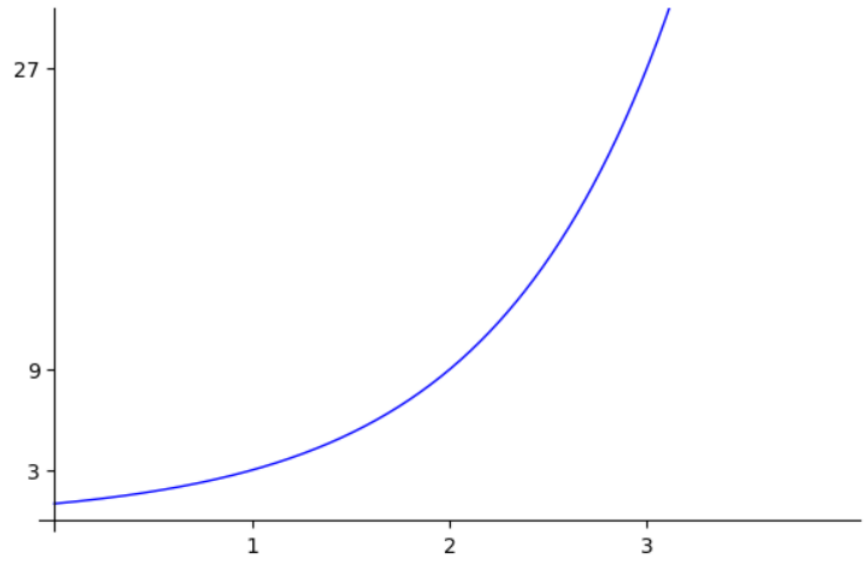
Fonte: O autor.

Pode-se escolher quais `ticks` serão mostrados especificamente, criando uma lista para o eixo x, e outra para o eixo y. Por exemplo, `ticks = [[1..3],[3, 9 ,27]]`, mostrará 1, 2 e 3 no eixo x; e os valores 3, 9 e 27 no eixo y, como na Figura 5.16.

Figura 5.16: Parâmetro `ticks` pensonalizado na função `plot( )`.

```
In [7]: plot(3^x,0,4,ymin=0,ymax=30,ticks=[[1..3],[3,9,27]])
        # A primeira lista mostra a numeração 1, 2 e 3 no eixo x;
        # A segunda lista mostra a numeração 3, 9 e 27.
```

Out[7]:



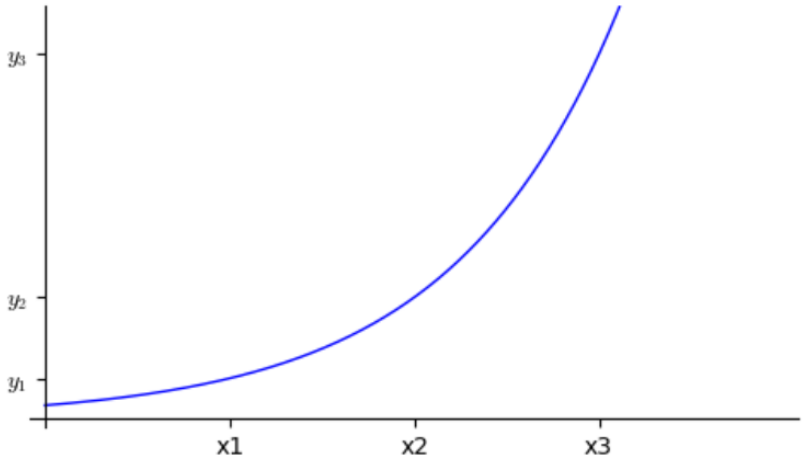
Fonte: O autor.

O `tick_formatter` permite a formatação de marcadores dos eixos cartesianos. Essa formatação é responsável por determinar como estes valores serão exibidos (em forma de texto) no gráfico. Por exemplo, na Figura 5.16 temos: `ticks = [[1..3],[3, 9, 27]]`, então `tick_formatter = [[‘x1’, ‘x2’, ‘x’], [‘y1’, ‘y2’, ‘y3’]]`, substitui cada número do `tick` pelo seu correspondente de mesmo índice, inclusive em notação latex. Veja a Figura 5.17.

Figura 5.17: Parâmetro `tick_formatter` na função `plot( )`.

```
In [27]: plot(3^x,0,4,ymin=0,ymax=30,ticks=[[1..3],[3,9,27]], figsize=(5,3),
              tick_formatter=[['x1','x2','x3',],['$y_1$','$y_2$','$y_3$']])
         # figsize=(5,3) altera a proporção do plano xy para 5/3
         # '$y_1$', '$y_2$' e '$y_3$'; (texto matemático em latex).
```

Out[27]:



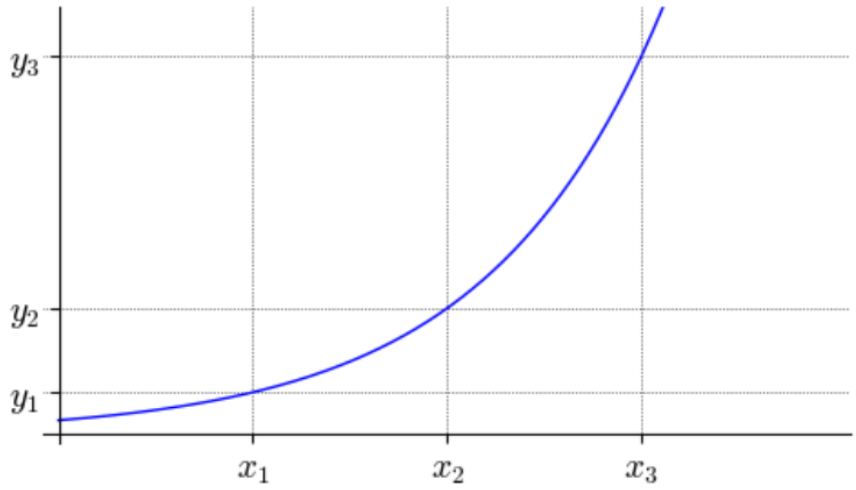
Fonte: O autor.

O parâmetro `gridlines` insere linhas de grade no plano cartesiano, criando uma referência visual que facilita a localização de pontos do gráfico. Esta grade auxilia na compreensão da relação entre as variáveis independentes e dependentes em uma função. O parâmetro referido recebe como entrada as opções: `‘minor’`, `‘normal’` e `‘major’`. Ver Figura 5.18.

Figura 5.18: Parâmetro `gridlines` na função `plot( )`.

```
In [31]: plot(3^x,0,4,ymin=0,ymax=30,ticks=[[1..3],[3,9,27]], figsize=(5,3),
             tick_formatter=[['$x_1$','$x_2$','$x_3$',],['$y_1$','$y_2$','$y_3$']],
            fontsize = 14, gridlines='normal')
         # fontsize=14 altera o tamanho da fonte nos ticks.
```

Out[31]:



Fonte: O autor.

Em resumo, os parâmetros `ticks`, `ticks_formatter` e `gridlines` melhoram significativamente a qualidade da representação gráfica, aprimorando-a como uma ferramenta analítica de suas características. Portanto, a adaptabilidade do *SageMath* é especialmente vantajosa para uma observação mais precisa do comportamento das funções exponenciais.

5.4.5 Exercícios da aula 4

1. Os modelos de crescimento populacional de Malthus, são os mais simples para descrever o crescimento de uma população ao longo do tempo. A função $f(t) = t_0.e^{kt}$ representa o modelo contínuo com taxa de crescimento instantânea (exponencial - maior precisão) e $g(t) = t_0.(1 + k)^t$, o modelo discreto com taxa de crescimento anual (linear por etapas - mais simples), onde:
   $f(t)$ e $g(t)$ representam a população total no instante t;
   k, a taxa de crescimento populacional;
   t_0, a população no instante inicial.

   Para responder os items abaixo, considere o estudo de uma cultura de bactérias em um ambiente ideal para o crescimento populacional, inicialmente com 100 bactérias a taxa de crescimento de 40% a cada 30 minutos.

   (a) Qual a população de bactérias em cada um dos modelos após uma, duas e três horas?

   (b) Crie uma representação gráfica com o comando `plot( )`, que apresente a comparação entre os modelos discreto e o contínuo de Malthus, representando o crescimento populacional dessa cultura durante um intervalo de 5 horas.

   (c) Acrescente os parâmetros: `axes_labels`, `legend_label`, `title`, `ticks` e `tick_formatter`, inserindo as informações necessárias ao gráfico.

2. Dados 100 gramas de uma amostra de urânio-238, o intervalo de tempo em que uma amostra deste elemento se reduz à metade (meia-vida) é de $4,5$ bilhões de anos.

   (a) Crie uma representação gráfica com os parâmetros: `axes_labels`, `legend_label`, `ticks` e `gridlines`, mostrando o decaimento radioativo da quantidade de urânio-238 restante ao longo de 45 bilhões de anos.
   Utilize a função $f(t) = t_0e^{-kt}$.

   (b) A quantidade do elemento será reduzida a 5 gramas após quanto tempo? Mostre geometricamente a resposta com um ponto vermelho no gráfico.

5.5 Aula 5: Função logarítmica no *SageMath*

5.5.1 Objetivos da aula

- Aprender a utilizar logaritmos e analisar a representação gráfica da função logarítmica no *SageMath*;

- Revisar conceitos básicos sobre função logarítmica e sua relação com a função exponencial;

- Compreender a aplicação prática para os logaritmos em situações do mundo real, analisando e resolvendo problemas no *SageMath*.

5.5.2 Habilidades BNCC

(**EM13MAT305**) Resolver e elaborar problemas com funções logarítmicas nos quais seja necessário compreender e interpretar a variação das grandezas envolvidas, em contextos como os de abalos sísmicos, pH, radioatividade, matemática financeira, entre outros.
(**EM13MAT403**) Analisar e estabelecer relações, com ou sem apoio de tecnologias digitais, entre as representações de funções exponencial e logarítmica expressas em tabelas e em plano cartesiano, para identificar as características fundamentais (domínio, imagem, crescimento) de cada função.
(**EM13MAT405**) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

5.5.3 Motivação

Uma das vantagens em estudar funções logarítmicas no *SageMath*, é a possibilidade de alteração da escala de apresentação do gráfico para para revelar mais detalhes sobre o comportamento dessas funções. A variação da escala logarítmica transforma o gráfico de da função exponencial em um da função linear, o que facilita a compreensão de suas características. Veja os códigos que serão utilizados nesta aula, na Tabela 5.6.

Tabela 5.6: Lista de códigos utilizados na aula 5

Códigos	Descrição
`log( )`	Fornece o logaritmo natural.
`.n( )`	Fornece uma aproximação numérica.
`digits`	Define a quantidade de casas decimais da aproximação fornecida pelo método *.n( )*.
`gridlines`	Insere uma malha quadriculada no gráfico. As entradas `'minor'`, `'normal'` e `'major'` mostram malhas com espaçamentos diferentes.
`scale`	Especifica a escala dos eixos do gráfico. O valor padrão do parâmetro `scale` é $(1, 1)$, o que significa que os eixos são igualmente espaçados.

Fonte: O autor.

5.5.4 Atividades Propostas

No *SageMath*, o comando `log( )` é utilizado para calcular o logaritmo em diferentes bases, e a sintaxe `log('logaritmando', 'base')`, calcula o logaritmo na base fornecida. Assim, `log(64,2)` calcula $\log_2 64$. Se a base não for especificada, por padrão, o logaritmo natural é calculado. Em acréscimo, caso seja necessário mostrar o valor decimal, o método 'expressão'`.n( )` mostra o valor aproximado dessa expressão, podendo limitar o número de algarismos da aproximação com o parâmetro `digits`. Por exemplo: `log(5).n(digits = 2)` será a aproximação de $\log_e 5$ com duas casas decimais, como mostra a Figura 5.19.

Figura 5.19: Função `log( )` e os parâmetros `.n( )` e `digits`.

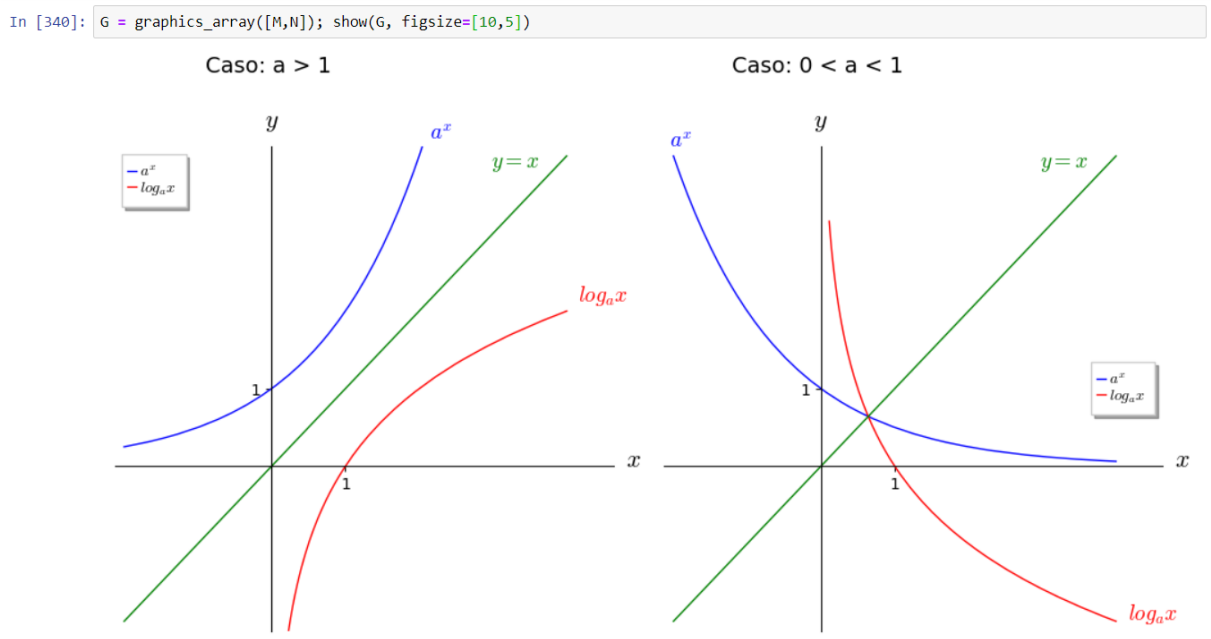
```
In [54]: log(64,2)
Out[54]: 6

In [56]: log(e)
Out[56]: 1

In [58]: log(5).n()
Out[58]: 1.60943791243410

In [59]: log(5).n(digits=2)
Out[59]: 1.6
```

Fonte: O autor.

A função logarítmica $y = log_a\ x$ é crescente quando $a > 1$ e decrescente quando $0 < a < 1$. Outra caraterística, é sua relação com a função exponencial. Por exemplo, se o ponto (m, n) a pertence $y = \log_a x$, então $n = \log_a m$; por definição, $m = a^n$ e o ponto (n, m) pertence a $y = a^x$. Como os pontos (m, n) e (n, m) são simétricos em relação a reta $y = x$, consequentemente, a simetria se estende para $y = \log_a x$ e $y = a^x$.

Sob a premissa de manter o foco na observação do comportamento da função, os comandos utilizados na construção dos gráficos que ilustram o parágrafo anterior, com funcionalidades mais complexas do *SageMath*, serão omitidos na Figura 5.20.

Figura 5.20: Sinal da função logaritmica e sua relação com a função exponencial.



Fonte: O autor.

Agora, aproximando os conceitos revisados com situações do mundo real, veremos uma das aplicações práticas em que são usados os logaritmos, a escala de pH (potencial hidrogeniônico). Dada por $pH = -\log_{10}\ H^+$, onde pH indica a acidez de uma solução de acordo com a concentração de íons de hidrogênio H^+. A escala varia entre 0 e 14, quanto menor é o valor de pH, mais ácida é a solução, sendo 7 o valor neutro. Portanto, as substâncias com pH inferior a 7 são consideradas ácidas, enquanto as substâncias com pH superior a 7 são consideradas básicas. Veja o exemplo da Figura 5.21.

Figura 5.21: Exemplo do pH de uma solução com concentração de ions H^+ de 10^{-6} mol/L

Vamos avaliar o pH de uma solução com concentração de íons H+ de 10^-6 mol/L:

In [7]:
```
var('pH H') # Declarando as variáveis pH e H;
H = 10^-6 # A concentração de íons H+ da solução é de 10^-6 mol/L;
pH = log(1/H,10) # função pH;
pH.n(digits=3) # Aproximação com 3 casas decimais de 6*log(10)
```

Out[7]: 6.00

Como o pH = 6, a solução é considerada levemente ácida.

Fonte: O autor.

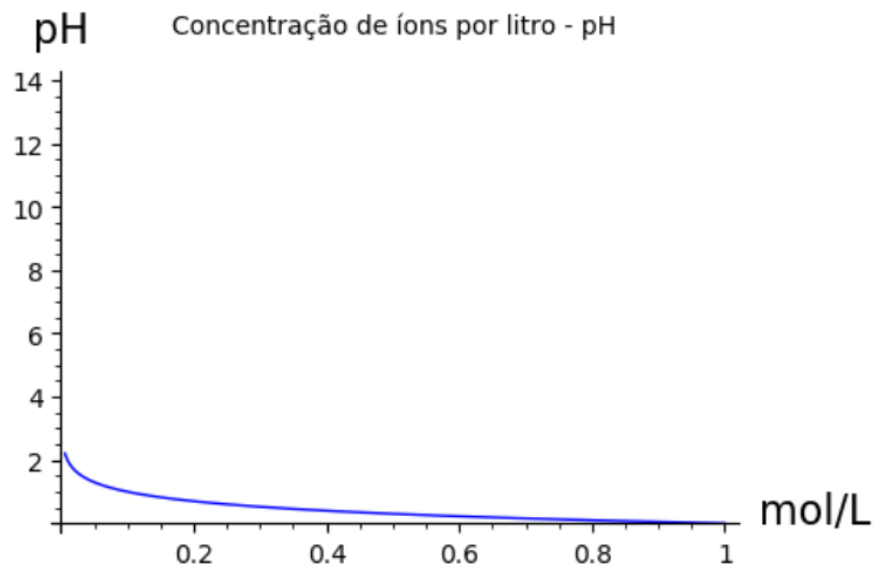
Na Figura 5.21, verificamos que o pH de uma solução com concentração de ions H^+ de 10^{-6} mol/L é 6. Portanto, uma solução levemente ácida.

Note que quanto mais alcalina (básica) é a solução, menor é a concentração de íons de hidrogênio por litro. Entretanto, como a escala de pH é logarítmica, cada unidade de alteração no pH representa uma alteração de 10 vezes na concentração de íons hidrogênio. Observe o gráfico da função $P(h) = -\log_{10} h$ na Figura 5.22.

Figura 5.22: Gráfico da função $P(h) = -\log_{10} h$

In [19]:
```
P(h) = -log(h,10)
plot(P, 0, 1, ymax = 14, axes_labels = ['mol/L', 'pH'],
    title='Concentração de íons por litro - pH', figsize=5)
```

Out[19]:



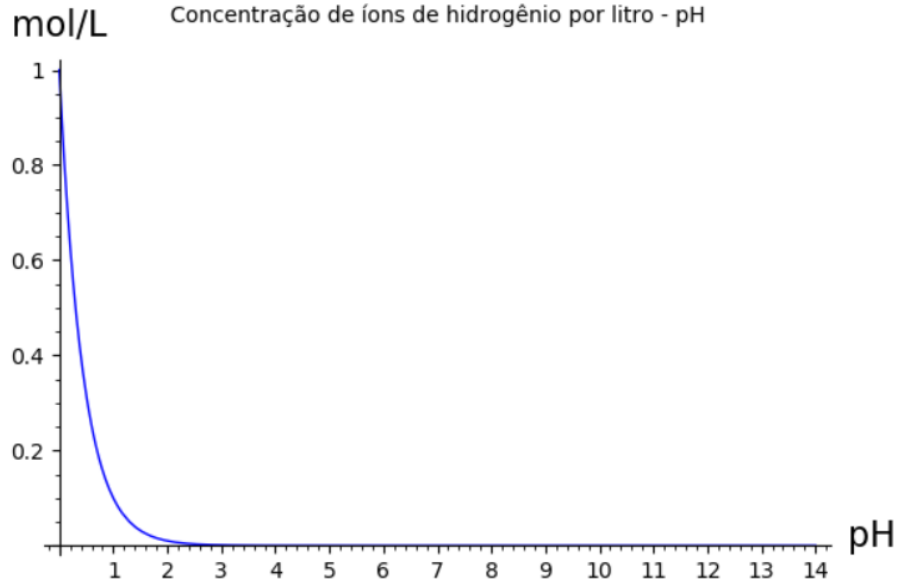
Fonte: O autor.

Devido à proximidade do gráfico ao eixo y se tornar exponencialmente microscópica, é visualmente incompreensível perceber a variação na concentração de íons de hidrogênio por litro na escala atual do plano cartesiano. De mesmo modo, vamos observar o gráfico da função $P'(h) = 10^{-h}$, inversa de $P(h)$, também invertemos os rótulos dos eixos cartesianos. Observe a Figura 5.23.

Figura 5.23: Gráfico da função $P'(h) = 10^{-h}$

```
In [106]: plot(10^-h, 0, 14,ticks=[1,None],axes_labels=['pH', 'mol/L'],
              title='Concentração de íons de hidrogênio por litro - pH')
```

Out[106]:



Fonte: O autor.

Contudo, é possível alterar a escala dos eixos cartesianos para escala logarítimica na função `plot( )`. Basta utilizar o parâmetro `scale = ('entrada', base)`, com base 10 caso não seja fornecida. Os tipos de entrada são:

`'semilogx'` – apresenta o eixo x em escala logarítmica;

`'semilogy'` - apresenta o eixo y em escala logarítmica;

`'loglog'` - apresenta ambos os eixos em escala logarítmica.

Veja na Figura 5.24, um exemplo do parâmetro `scale='semilogy'` na representação gráfica da função $P'(h) = 10^{-h}$, alteração que favorece a análise da função na escala logaritmica.

Figura 5.24: Gráfico da função $P'(h) = 10^{-h}$ com parâmetro `scale`.

```
In [105]: plot(10^-h, 0, 14, scale='semilogy',ticks=[1,None],axes_labels=['pH', 'mol/L'],
              title='Concentração de íons de hidrogênio por litro - pH')
```

Out[105]:

Fonte: O autor.

Nos próximos exercícios, vamos aplicar as funcionalidades do *SageMath* para interagir com as representações gráficas das funções logarítmicas, interpretando algumas situações matematicamente, modelando estes fenômenos.

5.5.5 Exercícios da aula 5

1. Apresente a solução da equação $\log_2 3x = 7$, nos seguintes passos:

   (a) Utilize o comando `solve( )` e armazene a solução na variável s;

   (b) Apresente o resultado utilizando o código `[índice].rhs( )` para acessar o lado direito da equação.

   (c) Mostre a aproximação do resultado com quatro casas decimais. Use o código `.n( )` e o parâmetro `digits`.

2. A velocidade máxima, em *bits* por segundo, com a qual os sinais podem passar por canais de comunicação, é obtida por meio da fórmula: $v_{max} = 3400 \log_2(x+1)$, onde 3400 *hertz* é a frequência limite da voz humana e x é a potência do sinal.

   Calcule o valor de x correspondente a uma velocidade máxima de 27200 bits por segundo.

3. O ICMBio – Instituto Chico Mendes de Conservação da Biodiversidade, ligado ao Ministério do Meio Ambiente, em parceria com unidades de conservação federais, fomenta recursos para a proteção e mapeia a evolução populacional das espécies F e G ameaçadas de extinção. O repovoamento desses animais na fauna brasileira é descrito, aos milhares, pelas funções $f(t) = \log_3(t+3)$ para a espécie F e $g(t) = \log_3(0.5t + 12)$ para G, onde t representa o tempo em anos.

   (a) Construa os gráficos de f e g no mesmo plano cartesiano com o domínio no intervalo $[0, 60]$ e acrescente o parâmetro `gridlines = 'normal'`;

   (b) Insira rótulos nos eixos do gráfico (`axes_labels`) e legenda para as espécies F e G (`legend_label`);

   (c) Qual é a previsão da quantidade desses animais no Brasil, após 30 anos do início da ação que protege a reprodução dessas espécies?

6 Aplicação da Sequência Didática

6.1 Descrição do Ambiente e Aplicação

6.1.1 Ambiente

A sequência didática foi aplicada com alunos do 1º ano do Ensino Médio na Escola Estadual Professora Benedita de Castro Lima, situada à rua Santa Rita, S/N - Clima Bom, Maceió - AL, pertencente à 13ª Gerência Regional de Educação do Estado. Confira algumas imagens da escola nas Figuras 6.1 e 6.2.

Figura 6.1: Escola Estadual Prof.ª Benedita de Castro Lima (*área externa*).

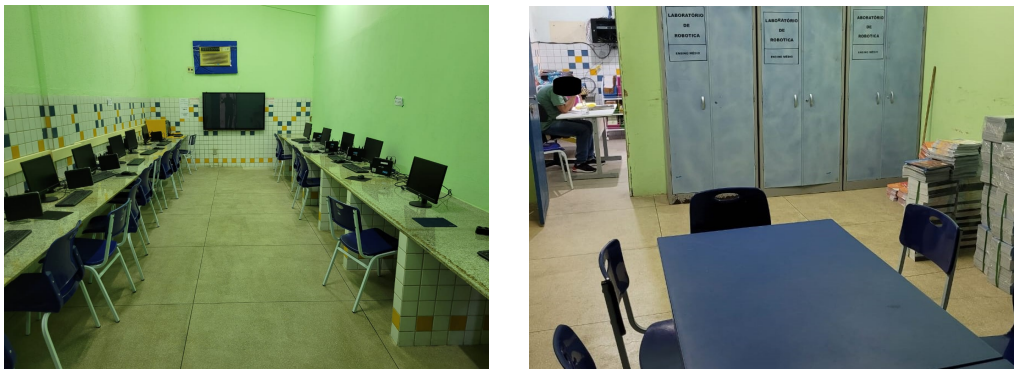
Fonte: O autor.

Figura 6.2: Escola Estadual Prof.ª Benedita de Castro Lima (*área interna*).

Fonte: O autor.

A instituição é atendida pelo Programa Alagoano de Ensino Integral (pALei), promovido pelo Governo do Estado, por meio da Secretaria de Estado da Educação (Seduc). Em sua estrutura, dispõe de biblioteca, ginásio de esportes, auditório e dezesseis salas de aula refrigeradas em funcionamento. Além disso, possui laboratórios de Informática, Robótica e Ciências, ilustrados nas Figuras 6.3 e 6.4.

Figura 6.3: Laboratórios: de Informática (*à esquerda*) e de Robótica (*à direita*).



Fonte: O autor.

Figura 6.4: Laboratório de Ciências.

Fonte: O autor.

O laboratório de Informática é composto por uma sala refrigerada, *smart* TV e dez computadores. Porém, até o período de desenvolvimento deste trabalho, alguns dispositivos não funcionavam; e outros, apresentaram lentidão na execução de tarefas, inviabilizando aulas com uma maior quantidade de participantes.

O laboratório de Robótica possui uma sala para armazenamento de componentes (periféricos e eletrônicos), e uma sala menor com dois *notebooks* para pesquisa e programação dos robôs durante as aulas. Convenientemente, estes *notebooks*, adicionados ao do próprio autor, foram utilizados na aplicação da sequência didática nos momentos

em que aconteceram os problemas citados no parágrafo anterior.

Recentemente ampliado e reformado, o laboratório de Ciências é o que apresenta melhores condições para a realização de atividades práticas extra classe. Equipado com ferramentas didáticas e experimentais para aulas em disciplinas das ciências humanas, quadro branco, *Smart* TV e área de jogos matemáticos. Oportunamente, um dos locais onde ocorreu o desenvolvimento dos exercícios propostos, em alternativa ao laboratório de Informática.

6.1.2 Aplicação

Como visto no capítulo 5, a sequência didática contempla um total de cinco aulas, cada uma planejada e executada em 120 minutos, divididos em dois momentos. O primeiro, na revisão de conceitos básicos dos conteúdos e apresentação dos códigos necessários, com exemplos na lousa e no *software*. O segundo momento, dedicado à resolução dos respectivos exercícios no *SageMath*, sob supervisão e orientação do autor.

A utilização do *SageMath* no estudo de funções é uma proposta inovadora e disruptiva com as metodologias tradicionais do ensino público. Portanto, o modelo de um estudo piloto foi escolhido de modo que possibilite a análise detalhada da aplicação, o acompanhamento individualizado e o aprimoramento ou adaptação da metodologia, como também da sequência didática, posteriormente para um grupo maior de participantes.

Observando a infraestrutura, equipamentos em funcionamento e ambientes disponíveis pela escola, em conjunto com as particularidades do modelo de estudo referido no parágrafo anterior, fizemos uma seleção durante uma das aulas de matemática, de voluntários para a aplicação do projeto. Assim, após a explanação sobre o *SageMath*, linguagem de programação *Python* e apresentação prática do *software*, tivemos quatro alunos que se propuseram a participar da aplicação de toda a sequência didática.

A aplicação da sequência didática ocorreu no horário das aulas semanais de Estudos Orientados, disciplina inserida através do pALei no currículo da rede pública do estado. Nestas aulas, outro professor é designado para acompanhar a turma no desenvolvimento de técnicas e hábitos de estudo, construindo a autonomia do estudante na busca pelo

conhecimento. Portanto, a aplicação da pesquisa não alterou o planejamento anual da turma, pois apenas os voluntários se dirigiam aos laboratórios com o pesquisador, enquanto os demais permaneciam em sala com o professor orientador. Logo, não interferiu nos horários de aula dos componentes curriculares obrigatórios assegurados pela BNCC.

Os dados investigados e resultados colhidos referem-se ao público participante em todas as aulas da sequência didática, identificados como alunos A, B, C e D. Contudo, na aula 2, tivemos a visita de alguns discentes do 3º ano e o apoio dos "*pibidianos - Ufal*" na aula. Com a permissão do professor orientador da turma, houve a participação como espectadores dos alunos do 1º ano "D" nas aulas 4 e 5, também no horário da disciplina de Estudos Orientados.

É importante ressaltar que as duas últimas aulas da sequência coincidiram com o período de revisão bimestral na sala regular, sobre funções exponenciais e logarítmicas para o simulado de matemática no 3º bimestre.

6.2 Coleta de dados

O período em que ocorreu a coleta de dados se deu entre os meses de maio e setembro de 2023, compreendendo o segundo e terceiro bimestres letivos da unidade de ensino.

6.2.1 Aula 1

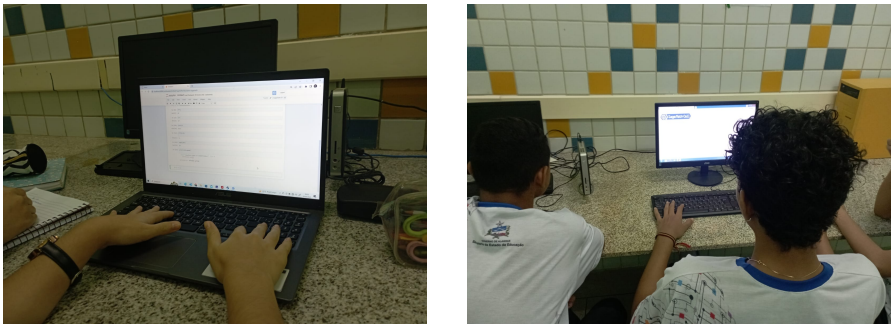
Os estudantes visitaram o laboratório de Robótica, vide Figura 6.5, onde tiveram o primeiro contato com o *SageMath* em sua versão *on-line* (*SageMathCell*), nos computadores do laboratório e na versão instalada no *notebook* do autor. Inicialmente, foram apresentados os operadores aritméticos da Tabela 3.1, compreendendo o funcionamento do *software*, em particular, da relação entre "entrada" (*in*) e "saída" (*out*) dos códigos. Em seguida, praticaram e analisaram os comandos da Tabela 5.1 em exemplos sobre as propriedades e operações relacionadas aos conjuntos numéricos.

Os alunos B e D não possuiam familiaridade com o teclado, de modo que necessitaram de ajuda para utilizar caracteres específicos necessários na sintaxe dos códigos. Além disso, questionaram sobre o uso do inglês nos comandos apresentados na Tabela 5.1. Contudo, realizaram todos os exemplos trabalhados anteriormente.

Como mostrado na Figura 6.6, os estudantes realizaram a atividade durante o segundo momento no *SageMathCell* em uma única célula de comando. Nos exercícios de 1 à 4, aplicaram os comandos na resolução de problemas sobre intervalos numéricos e operações entre conjuntos, armazenando listas de elementos em variáveis com a função `set( )`. Agora, com o objetivo de praticar e desenvolver o pensamento computacional no exercício 5, utilizaram a estrutura de repetição *loop* (laço) do comando `for`, com o auxílio dos conceitos introdutórios da linguagem *Python*: operadores aritméticos, variáveis e listas como estruturas de armazenamento, na resolução de problemas.

Os participantes tentaram resolver os problemas individualmente, apenas consultando o material em formato PDF da aula 1. O percentual de acertos na execução correta dos códigos foi de: 75% no exercício 1, 68, 75% no exercício 2, 25% no exercício 3, 77, 77% no exercício 4 e de 25% no exercício 5. Posteriormente, compartilharam os códigos executados com os colegas e concluiram os exercícios com as orientações do pesquisador.

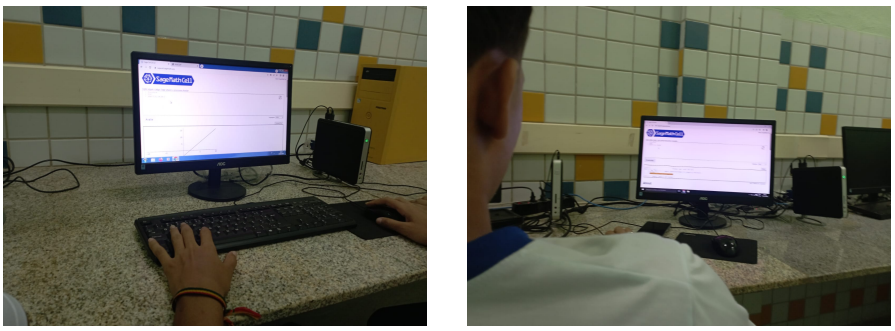
Figura 6.5: Aula 1 - Laboratório de Informática - Momento 1



Fonte: O autor.

Os erros de código que foram observados durante a resolução dos exercícios, se deram na utilização do operador = no lugar de == para comparar as expressões no exercício 3; na aplicação incorreta dos operadores `( )` e `[ ]` do comando `set([ ])` para executar o *loop*; e, na ordem de precedência ao escrever 2^{2x+1} na sintaxe do *SageMath* no exercício 5. Então, no final da aula houve a discussão destes detalhes da linguagem, em paralelo com uma revisão sobre as propriedades e operações entre conjuntos numéricos no *software*.

Figura 6.6: Aula 1 - Laboratório de Informática - Momento 2



Fonte: O autor.

6.2.2 Aula 2

A aula 2 se desenvolveu no laboratório de Ciências devido à falta de internet e a impossiblidade de instalação do *software*, no momento da pesquisa. Mas a alteração de local oportunou a paticipação de estudantes do 3º ano, que aguardavam o resultado de um experimento com o professor de Biologia no laboratório. Assim, por alguns minutos antes da aula, os visitantes foram instruídos pelos alunos A, B, C e D nos comandos da aula anterior. Ver Figuras 6.7 e 6.8.

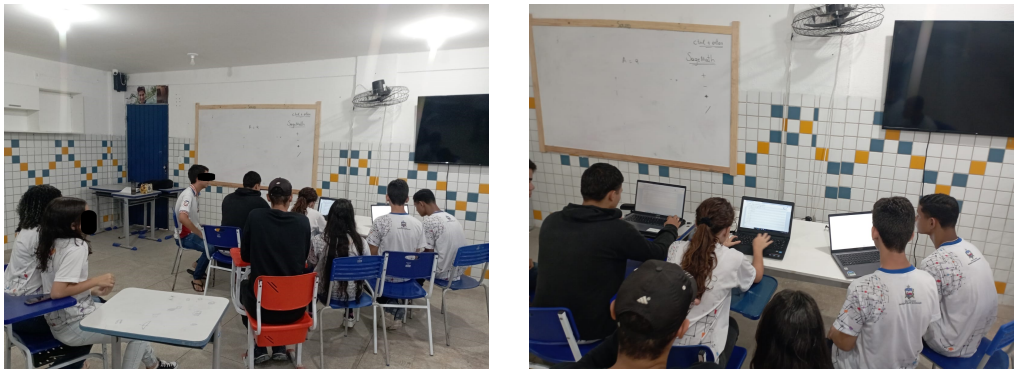
No primeiro momento, os alunos aprenderam como declarar variáveis com o comando `var( )`, procedimento necessário para que o *software* entenda e manipule estas variáveis como símbolos matemáticos. Também, como usar a função `solve( )` na resolução de equações e sistemas de equações.

Nesta etapa, os alunos sinalizaram dificuldade na compreensão da sintaxe dos comandos descritos na tabela 5.2, em como escrever sistema de equações nos parâmetros da função `solve( )` e nas instruções para acessar numericamente as soluções fornecidas. Portanto, foi necessário um tempo além do planejamento para a compreensão de alguns detalhes do cógido, da relação entre os índices de posicionamento `[ ]` e os elementos da lista `[ ]`; e da concatenação entre códigos e funções no *SageMath*.

Esta atividade foi baseada no desenvolvimento prático da manipulação de expressões matemáticas num ambiente de computação algébrica, na resolução de equações e na interação com os parâmetros das funções gráficas que estudaremos nas próximas aulas.

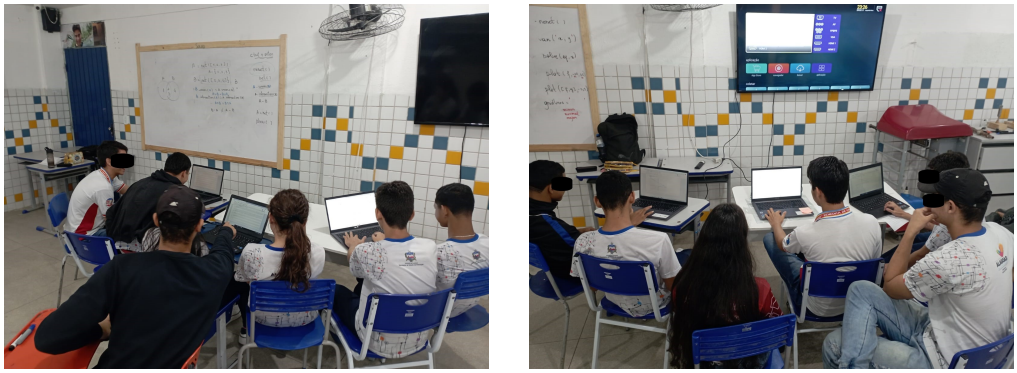
Os alunos A, B, C e D alcançaram 100% de acertos na resolução do exercício 1, mostrando algebricamente as soluções das equações do 1º e do 2º grau; 66, 6% em acertos nos exercícios 2 e 3, pois não conseguiram executar o item *c* das duas questões. Por fim, 0% de acerto no exercício 4, onde todos incorreram em erro de sintaxe na escrita do comando `solve([x + y == 6, x - y == 1], x, y)`, ainda no item *a*.

Figura 6.7: Aula 2 - Laboratório de Ciências - Momento 1



Fonte: O autor.

Figura 6.8: Aula 2 - Laboratório de Ciências - Momento 2



Fonte: O autor.

As dificuldades encontradas nos detalhes do comando `solve( )` e em acessar o valor numérico de uma solução com o método `.rhs( )` no exercício 4, despertou o senso de desafio nos alunos durante a execução da atividade. Embora persistissem nas tentativas e análise dos erros de sintaxe apontados pelo *SageMath*, os estudantes continuaram engajados no cumprimento do exercício. Finalmente, concluíram a atividade com a ajuda do pesquisador e solicitaram revisão sobre sistema de equações e mais exemplos com a aplicação da função `point( )` no plano cartesiano.

6.2.3 Aula 3

Na aula 3, tivemos a participação de graduandos do curso de licenciatura em matemática da UFAL, componentes do PIBID (Programa de Institucional de Bolsa de Iniciação à Docência). Os Pibidianos acompanharam e auxiliaram os participantes nos dois momentos da aplicação.

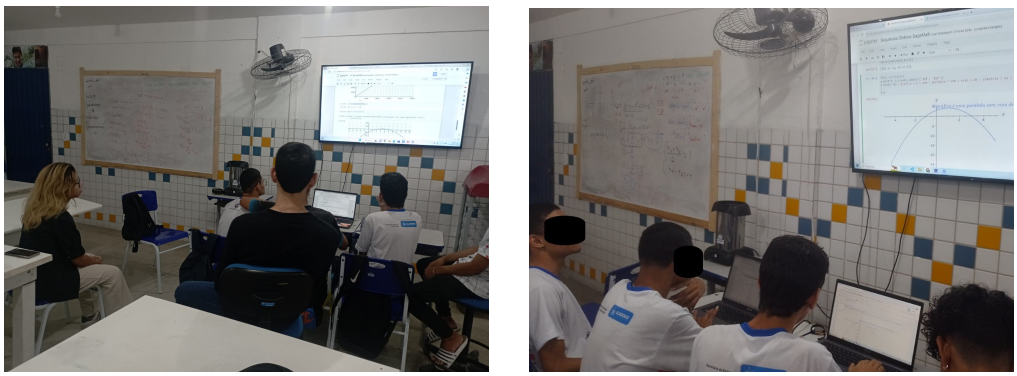
O primeiro momento aconteceu no Laboratório de Informática e foi destinado a contrução da representação das funções polinomais com o comando `plot( )`, na prática dos comandos dispostos na Tabela 5.4 e nos exemplos apresentados na aula. Ao mesmo tempo, também na observação e análise das mudanças relacionadas a cada alteração feita nos parâmetros das funções e dos comandos no *SageMath*.

A Tv e o ar condicionado do laboratório de Informática apresentaram problemas, então o segundo momento aconteceu no laboratório de Ciências. Os alunos realizaram a atividade ainda sob acompanhamento dos participantes do PIBID, que tiraram dúvidas e deram dicas no quadro branco sobre as características das funções do 1^{o} e 2^{o} graus. Assim, auxiliando os alunos na resolução de problemas através do *software*.

Figura 6.9: Aula 3 - Laboratório de Informática - Momento 1

Fonte: O autor.

Figura 6.10: Aula 3 - Laboratório de Ciências - Momento 2



Fonte: O autor.

Os alunos B e D tiveram dificuldades nas questões 1 e 2, mas após algumas intervenções dos pibidianos, compreenderam os conceitos algébricos abordados nos exercícios, interpretando graficamente os coeficientes e as raízes da função na questão 1; e na questão 4, também o vértice e o eixo de simetria do gráfico. Além disso, ainda sob supervisão dos graduandos, os quatro participantes permaneceram no laboratório durante o intervalo, praticando e realizando construções extras em questões abordadas anteriormente apenas na sala de aula regular.

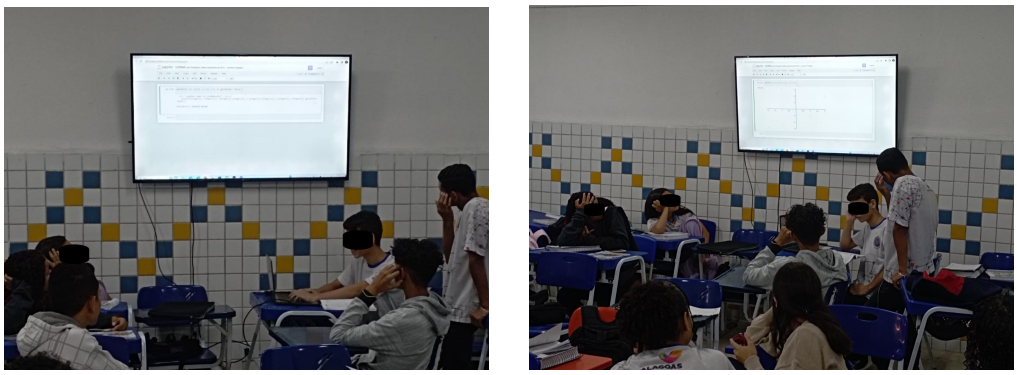
6.2.4 Aula 4

O desenvolvimento da quarta aula aconteceu na sala convencional. Inicialmente, com o restante da turma observando e tirando dúvidas sobre questões que estavam sendo solucionadas no quadro, enquanto os alunos participantes também praticavam os novos códigos e sintaxes no *SageMath*, junto com a resolução da atividade proposta.

Embora o planejamento do primeiro momento reservasse esse tempo para o acompanhamento dos exemplos nos códigos necessários, os alunos já sentiam confiança em lidar autodidaticamente com as instruções no *software*. Porém, a turma estava em semana de revisão para o simulado sobre funções exponenciais do 3º bimestre, mesmo tema da aula 4. Por isso, as construções e interações no *SageMath* com os gráficos da aula de revisão, auxiliaram a turma na comparação com as respostas e explicações fornecidas pelo professor.

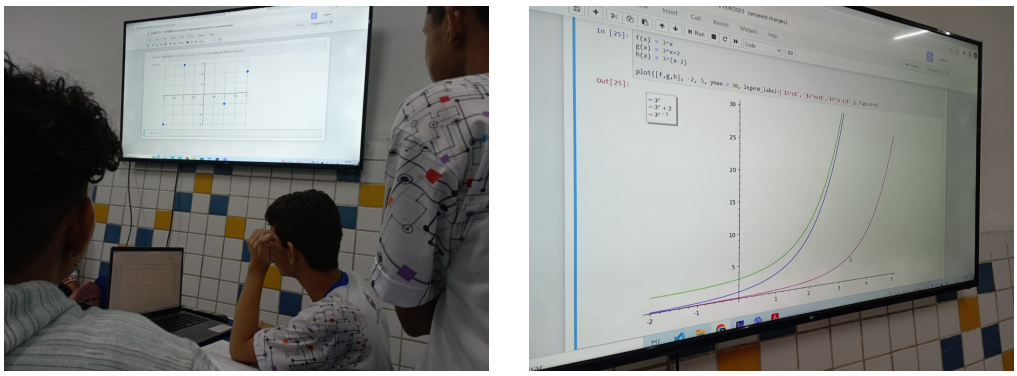
No segundo momento, os alunos participantes continuaram a interação com os colegas explicando comandos de plotagem dos gráficos, como `point( )` e `plot( )`, descreveram cada passo da resolução da atividade proposta, chamando atenção para interpretação do gráfico e relacionado-o ao contexto de cada questão.

Figura 6.11: Aula 4 - Sala de aula - Momento 1



Fonte: O autor.

Figura 6.12: Aula 4 - Sala de aula - Momento 2



Fonte: O autor.

A resolução dos exercícios da aula 4 foi uma das mais construtivas, juntamente com os exercícios da aula 5 que mencionaremos posteriormente. Durante toda a atividade houve engajamento e trabalho em equipe, na discussão e utilização dos conceitos relacionados a funções exponenciais. Os alunos demonstraram domínio na sintaxe dos comandos estudados até o momento no *SageMath*, na sua aplicabilidade; e, exceto pelo item b da questão 2 que necessitou de manipulação algébrica no quadro, na solução das situações propostas.

6.2.5 Aula 5

Na quinta e última aula, os dois momentos aconteceram em dias diferentes, pois durante o primeiro momento, apesar do número de funções e parâmetros da aula serem reduzidos, os estudantes sinalizaram dificuldades na compreensão e utilização dos logaritmos no *software*, visto que até o momento tinham estudado uma única vez o conteúdo na sala de aula. Portanto, antes do segundo momento tivemos um horário para rever a definição e alguns exemplos.

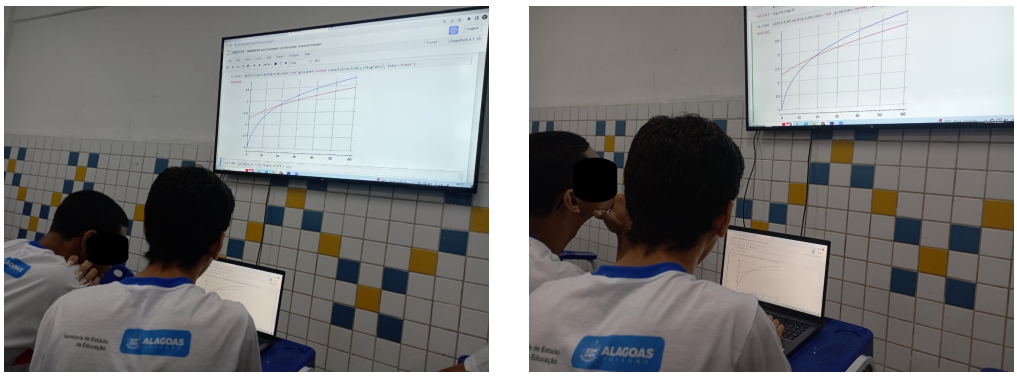
Assim, no primeiro momento houve uma discussão relacionando a função logarítmica e sua inversa, a função exponencial, sobre a simetria entre seus pontos no plano cartesiano em relação a reta $y = x$, espelhamento observado na construção dos gráficos de $y = 2^x$ e de $y = \log_2 x$ no *SageMath*. Também, uma aplicação dos logaritmos na verificação da acidez de uma solução. Por fim, como alterar a escala dos eixos do plano para uma escala com base logarítmica.

Durante o segundo momento, os alunos praticaram a utilização de comandos já vistos para os exercícios 1 e 2, na sintaxe de resolução de uma equação e como acessar a aproximação decimal desta solução. Ainda, no exercício 3 conjecturaram situações na interpretação dos gráfico das funções f e g com o desenvolvimento das espécies F e G mencionadas na questão.

Figura 6.13: Aula 5 - Sala de aula - Momento 1

Fonte: O autor.

Figura 6.14: Aula 5 - Sala de aula - Momento 2



Fonte: O autor.

No exercício final, os estudantes se interessaram bastante pela diferença do crescimento populacional de uma cultura com recursos ilimitados e a outra com recursos limitados, fazendo a conexão entre os comportamentos dos gráficos das funções exponencial e logarítmica. Após a aula, os alunos perguntaram sobre outras situações reais que pudessem ser descritas com a utilização de logaritmos, levantamos alguns questionamentos sobre o crescimento populacional de fungos, vírus e bactérias; e sobre a taxa de contágio e proliferação de doenças. Posteriormente, este tema foi levado ao professor de Biologia da turma, promovendo uma discussão e aprendizado interciscisplinar.

6.3 Discussão dos Resultados

Averiguando o impacto no aprendizado e na participação dos alunos durante e ao final da aplicação didática, observamos que a utilização do *SageMath* em ambientes adequados, são recursos importantes que possibilitaram o desenvolvimento prático dos conteúdos sugeridos, e que as atividades produziram resultados satisfatórios entre os participantes A, B, C e D.

Durante a realização das duas primeiras aulas, percebemos algumas dificuldades em relação à compreensão e utilização da sintaxe nos códigos; e em acessar alguns caracteres no teclado. Estes obstáculos apontam a necessidade de promover a inclusão digital dos alunos da rede pública, no uso das tecnologias em preparação para o mercado de trabalho. Entretanto, os estudantes superaram os desafios encontrados no decorrer da aplicação.

Embora a instituição possua melhores equipamentos e estrutura em comparação com outras escolas da comunidade, ficou claro a necessidade de investimentos em equipamentos e na adequação dos espaços, de modo a comtemplar uma maior quantidade de alunos no ensino prático e tecnológico. Além disso, o planejamento das aulas consumiu um tempo considerável no estudo do *software*, da linguagem, e em pesquisas de materiais teóricos, dispostos majoritariamente em inglês, atos que requerem a formação contínua e especializada por parte do docente.

Os conteúdos e exercícios trabalhados no decorrer deste trabalho, foram personlizados e adaptados com facilidade devido ‘ grande flexibilidade do *SageMath*, assim como a apresentação dos conceitos de modo objetivo e contextualizado nas construções das representações gráficas de cada função.

Ainda, a introdução à linguagem de programação *Python* na interação com exemplos e questões, possibilitou o desenvolvimento do pensamento lógico e computacional nos alunos, estimulando a percepção de padrões e a compreensão de conteúdos nos processo de construção e resolução dos exercícios.

Em pesquisa realizada com os participantes, foram coletadas algumas informações sobre a satisfação, desempenho e aplicabilidade dos conhecimentos desenvolvidos no *SageMath*.

Nas Figuras 6.15, 6.16, 6.17, e 6.18, temos os questionários respondidos pelos estudandes; e na Tabela 6.1 as respectivas transcrições:

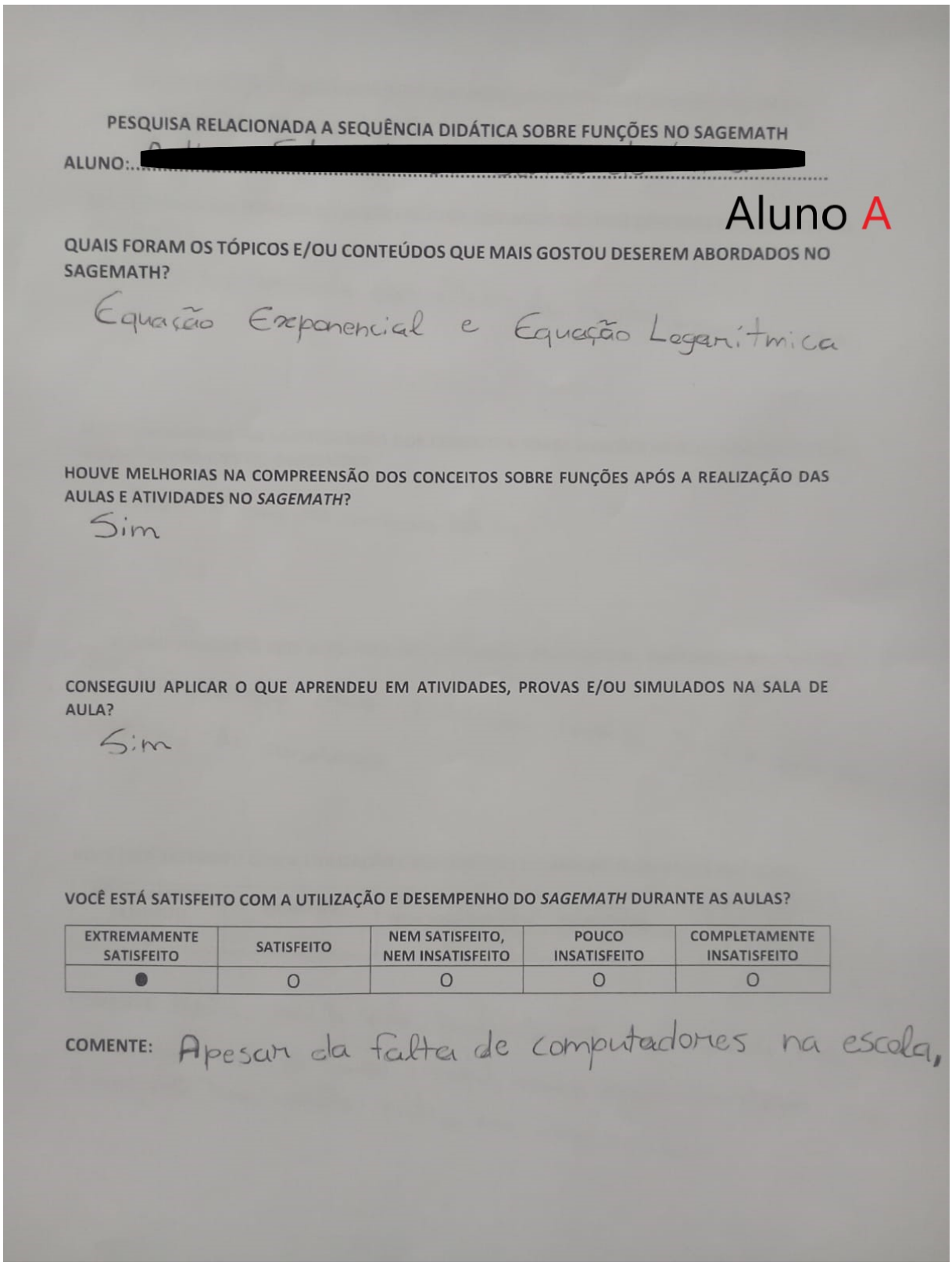
1. Quais foram os tópicos e/ou conteúdos que mais gostou de serem abordados no *SageMath*?

2. Houve melhora na compreensão dos conceitos sobre funções após a realização das aulas e atividades no *SageMath*?

3. Conseguiu aplicar o que aprendeu em atividades, provas e/ou simulados na sala de aula?

4. Você está satisfeito com a utilização e desempenho do *SageMath* durante as aulas?

Figura 6.15: Pesquisa de participação - Aluno A



PESQUISA RELACIONADA A SEQUÊNCIA DIDÁTICA SOBRE FUNÇÕES NO SAGEMATH

ALUNO: [redacted]

Aluno A

QUAIS FORAM OS TÓPICOS E/OU CONTEÚDOS QUE MAIS GOSTOU DESEREM ABORDADOS NO SAGEMATH?

Equação Exponencial e Equação Logaritmica

HOUVE MELHORIAS NA COMPREENSÃO DOS CONCEITOS SOBRE FUNÇÕES APÓS A REALIZAÇÃO DAS AULAS E ATIVIDADES NO *SAGEMATH*?

Sim

CONSEGUIU APLICAR O QUE APRENDEU EM ATIVIDADES, PROVAS E/OU SIMULADOS NA SALA DE AULA?

Sim

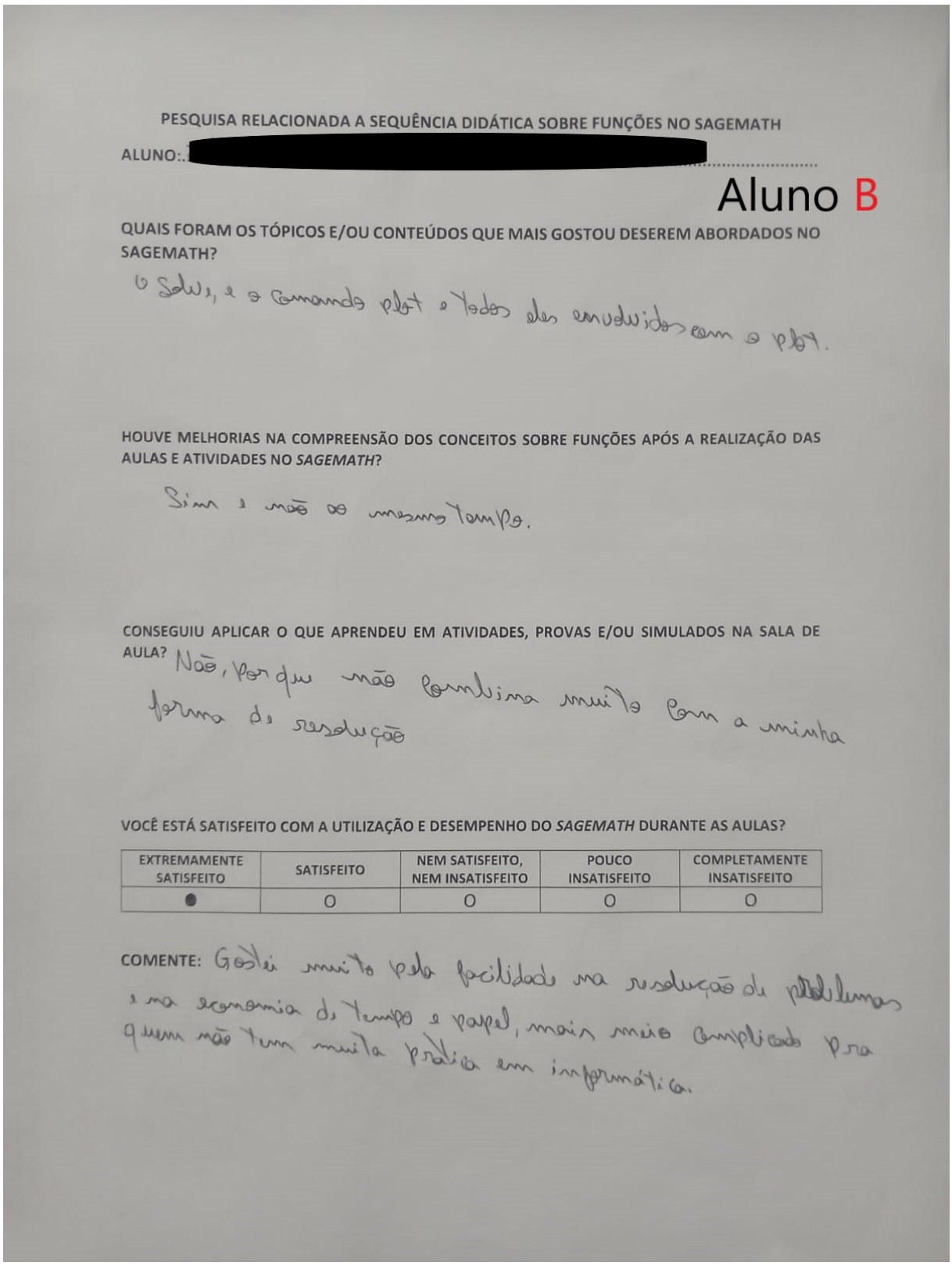
VOCÊ ESTÁ SATISFEITO COM A UTILIZAÇÃO E DESEMPENHO DO *SAGEMATH* DURANTE AS AULAS?

EXTREMAMENTE SATISFEITO	SATISFEITO	NEM SATISFEITO, NEM INSATISFEITO	POUCO INSATISFEITO	COMPLETAMENTE INSATISFEITO
●	O	O	O	O

COMENTE: Apesar da falta de computadores na escola,

Fonte: O autor.

Figura 6.16: Pesquisa de participação - Aluno B



PESQUISA RELACIONADA A SEQUÊNCIA DIDÁTICA SOBRE FUNÇÕES NO SAGEMATH

ALUNO:.[redacted]

Aluno B

QUAIS FORAM OS TÓPICOS E/OU CONTEÚDOS QUE MAIS GOSTOU DESEREM ABORDADOS NO SAGEMATH?

O Solve, e o comando plot e todos eles envolvidos com o plot.

HOUVE MELHORIAS NA COMPREENSÃO DOS CONCEITOS SOBRE FUNÇÕES APÓS A REALIZAÇÃO DAS AULAS E ATIVIDADES NO *SAGEMATH*?

Sim e não ao mesmo tempo.

CONSEGUIU APLICAR O QUE APRENDEU EM ATIVIDADES, PROVAS E/OU SIMULADOS NA SALA DE AULA? Não, porque não combina muito com a minha forma de resolução

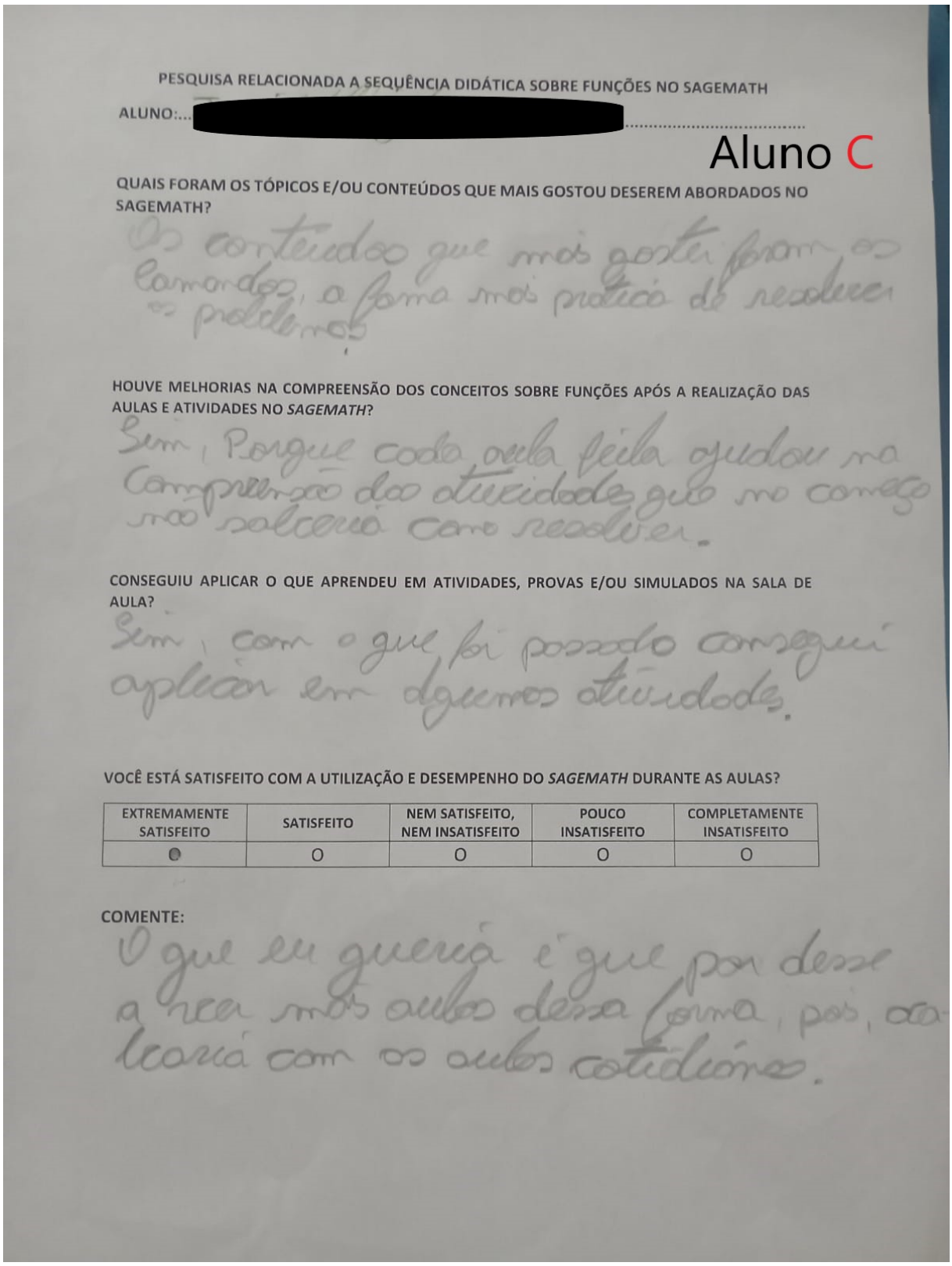
VOCÊ ESTÁ SATISFEITO COM A UTILIZAÇÃO E DESEMPENHO DO *SAGEMATH* DURANTE AS AULAS?

EXTREMAMENTE SATISFEITO	SATISFEITO	NEM SATISFEITO, NEM INSATISFEITO	POUCO INSATISFEITO	COMPLETAMENTE INSATISFEITO
●	O	O	O	O

COMENTE: Gostei muito pela facilidade na resolução de problemas e na economia de tempo e papel, mais mais complicado pra quem não tem muita prática em informática.

Fonte: O autor.

Figura 6.17: Pesquisa de participação - Aluno C



PESQUISA RELACIONADA A SEQUÊNCIA DIDÁTICA SOBRE FUNÇÕES NO SAGEMATH

ALUNO:... [redacted] .....

Aluno C

QUAIS FORAM OS TÓPICOS E/OU CONTEÚDOS QUE MAIS GOSTOU DESEREM ABORDADOS NO SAGEMATH?

Os conteúdos que mais gostei foram os comandos, a forma mais prática de resolver os problemas.

HOUVE MELHORIAS NA COMPREENSÃO DOS CONCEITOS SOBRE FUNÇÕES APÓS A REALIZAÇÃO DAS AULAS E ATIVIDADES NO *SAGEMATH*?

Sim, Porque cada aula feita ajudou na Compreenção das atividades que no começo não sabia como resolver.

CONSEGUIU APLICAR O QUE APRENDEU EM ATIVIDADES, PROVAS E/OU SIMULADOS NA SALA DE AULA?

Sim, com o que foi passado consegui aplicar em alguns atividades.

VOCÊ ESTÁ SATISFEITO COM A UTILIZAÇÃO E DESEMPENHO DO *SAGEMATH* DURANTE AS AULAS?

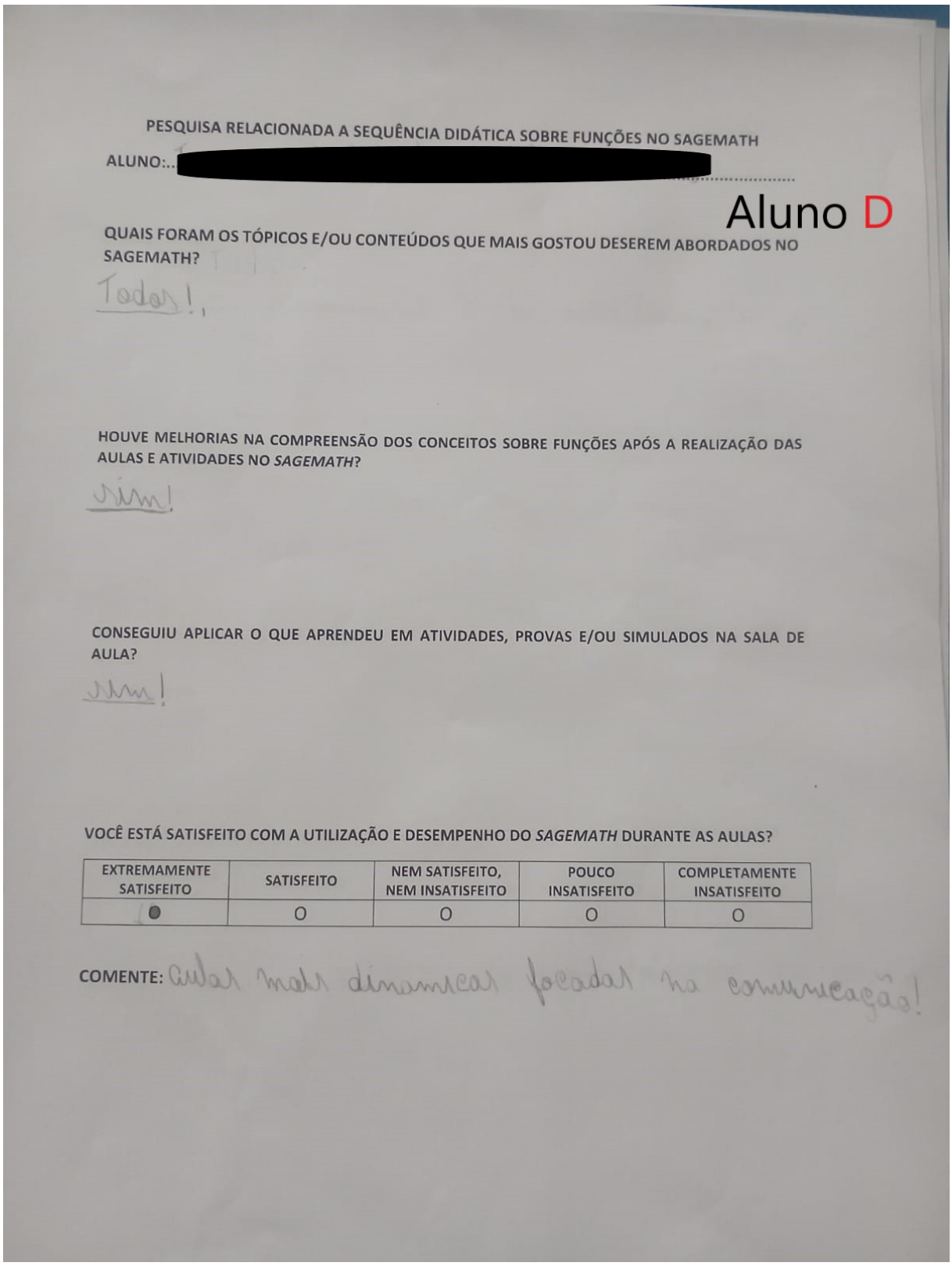
EXTREMAMENTE SATISFEITO	SATISFEITO	NEM SATISFEITO, NEM INSATISFEITO	POUCO INSATISFEITO	COMPLETAMENTE INSATISFEITO
●	O	O	O	O

COMENTE:

O que eu queria é que por desse a vez nos aulas dessa forma, pois, aprenderia com os aulas estudiosas.

Fonte: O autor.

Figura 6.18: Pesquisa de participação - Aluno D



PESQUISA RELACIONADA A SEQUÊNCIA DIDÁTICA SOBRE FUNÇÕES NO SAGEMATH

ALUNO:..[redacted]

Aluno D

QUAIS FORAM OS TÓPICOS E/OU CONTEÚDOS QUE MAIS GOSTOU DESEREM ABORDADOS NO SAGEMATH?

Todos!

HOUVE MELHORIAS NA COMPREENSÃO DOS CONCEITOS SOBRE FUNÇÕES APÓS A REALIZAÇÃO DAS AULAS E ATIVIDADES NO *SAGEMATH*?

Sim!

CONSEGUIU APLICAR O QUE APRENDEU EM ATIVIDADES, PROVAS E/OU SIMULADOS NA SALA DE AULA?

Sim!

VOCÊ ESTÁ SATISFEITO COM A UTILIZAÇÃO E DESEMPENHO DO *SAGEMATH* DURANTE AS AULAS?

EXTREMAMENTE SATISFEITO	SATISFEITO	NEM SATISFEITO, NEM INSATISFEITO	POUCO INSATISFEITO	COMPLETAMENTE INSATISFEITO
●	O	O	O	O

COMENTE: Gostei mais dinamicas focadas na comunicação!

Fonte: O autor.

Tabela 6.1: Transcrição das respostas apresentadas durante a pesquisa.

Alunos	**Pergunta 1**	**Pergunta 2**	**Pergunta 3**	**Pergunta 4**
Aluno A	Equação exponencial e equação logarítimica	Sim	Sim	Extremamente satisfeito
Aluno B	O solve, o comando plot e todos os envolvidos com o plot	Sim e não ao mesmo tempo	Não, porque não combina muito com minha forma de resolução	Extremamente satisfeito
Aluno C	Os conteúdos que mais gostei foram os comandos, a forma mais prática de resolver os problemas	Sim, porque cada aula feita ajudou na compreensão das atividades que no começo não saberia como resolver	Sim, com o que foi passado consegui aplicar em algumas atividades	Extremamente satisfeito
Aluno D	Todos	Sim	Sim	Extremamente satisfeito

Fonte: O autor.

As respostas mostram pontos de vista diferentes entre os participantes, embora culminem no mesmo resultado para a pergunta 4. Ou seja, apesar da sequência didática no *SageMath* ter sido baseada no mesmo itinerário para todos os participantes, a aplicação atingiu aspectos diferentes do aprendizado em cada um deles.

Segundo o planjemanto anual de matemática para a turma do 1º ano, estavam previstos os seguintes conteúdos:

- 1º Bimestre - Conjuntos e Função Afim;
- 2º Bimestre - Equação do 2º Grau e Função Quadrática;
- 3º Bimestre - Equação e Função Exponencial;
- 4º Bimestre - Equação e Função Logarítmica.

Para a sala de aula, o desenvolvimento dos alunos foi observado durante todo o processo de aplicação da sequência didática. Nesse sentido, apresentaram uma

melhora significativa da participação e realização das atividades. Consequentemente, esse progresso se refletiu nas notas no 3º e 4º bimestres em matemática. Ver Figura 6.19.

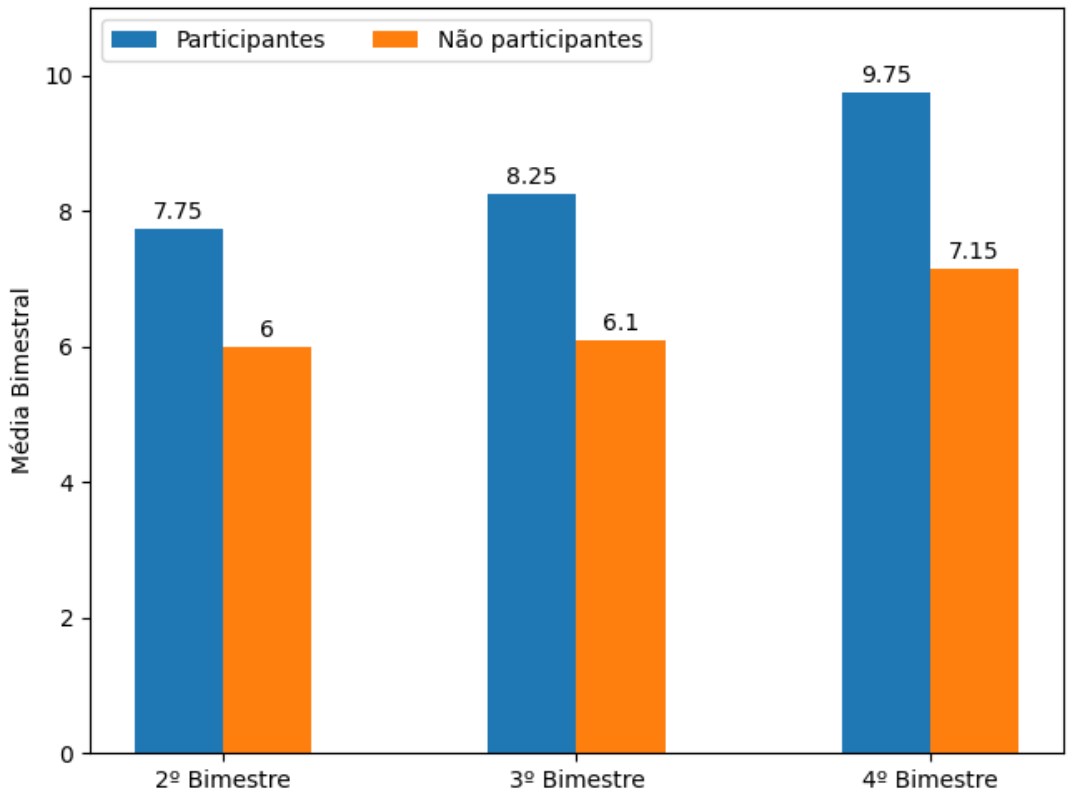
Figura 6.19: Boletins escolares contendo as notas e faltas em cada um dos quatro bimestres do ano de 2023 dos alunos A, B, C e D.

ALUNO A	MATEMATICA	10,0	6	-	-	7,0	1	-	-	9,5	8	-	-	10,0	0
ALUNO B	MATEMATICA	9,0	3	-	-	8,0	0	-	-	9,0	1	-	-	10,0	3
ALUNO C	MATEMATICA	7,5	0	-	-	8,0	0	-	-	8,5	0	-	-	10,0	0
ALUNO D	MATEMATICA	9,5	0	-	-	8,0	6	-	-	6,0	14	-	-	9,0	2

Fonte: O autor.

A turma dos alunos A, B, C e D é composta por 36 alunos, ou seja, 32 não participaram do estudo. Como a aplicação da sequência didática ocorreu entre os meses de maio e setembro, faremos uma comparação entre as médias bimestrais desses dois grupos no 2º, 3º e 4º bimestres. Ver Figura 6.20.

Figura 6.20: Gráfico com as médias bimestrais referentes ao 2º, 3º e 4º bimestres dos alunos participantes e não participantes da pesquisa.



Fonte: O autor.

Analisando as notas apresentadas pela turma dos alunos A, B, C e D entre o 2º e o 4º bimestre, o grupo participante da pesquisa apresentou um aumento na média bimestral de 25,8%, enquanto o grupo não participante obteve um aumento de 17,2%.

Mas, para além dos resultados numéricos do parágrafo anterior, os pontos principais a serem relatados ao final deste trabalho, são: a disposição dos alunos em estudar funções com a abordagem da linguagem de programação; a aplicabilidade do *SageMath* no ensino médio e a compreensão do comportamento dos gráficos plotados pelos estudantes.

Então, apesar das limitações estruturais ainda existentes na educação básica, os resultados produzidos pela utilização do *SageMath*, como ferramenta didática nas aulas de matemática, foram promissores, especialmente, como alternativa para a modernização do ensino da disciplina, inserção da linguagem de programação na grade curricular e na promoção da inclusão digital nas escolas públicas, ajudando na preparação dos alunos para o mundo moderno e tecnológico do trabalho.

Referências Bibliográficas

BELLAPU, A. **Analytcs Insight**. Disponível em: <https://www.analyticsinsight.net/top-10-most-used-programming-languages-among-software-developers/>. Acesso em 03 Abr 2023.

BONJORNO, J. R.; JÚNIOR, J. R. G.; SOUSA, P. R. C. **Coleção Prisma Matemática**: conjuntos e funções. – 1. ed. – São Paulo : Editora FTD, 2020.

BRASIL. Ministério da Educação. SEE AL. **Referencial Curricular de Alagoas – Ensino Médio**. Brasília, MEC, 2021.Disponível em: < https://www.gov.br/mec/pt-br/novo-ensino-medio/pdfs/copy_of_RCSEEAL.pdf >. Acesso em 15 Mar 2023.

BRASIL, Ministério da Educação. SEF. **Parâmetros Curriculares Nacionais para o Ensino Médio**. Brasília, MEC, 1998. Disponível em: <http://portal.mec.gov.br/seb/arquivos/pdf/ciencian.pdf>. Acesso em 10 Mar 2023.

BRASIL. Ministérios da Educação. Secretaria da Educação Básica. **Orientações Curriculares para o Ensino Médio. Volume 2. Ciências da Natureza, Matemática e suas Tecnologias**. Brasília: MEC, 2006. 135 p. Disponível em: <http://portal.mec.gov.br/seb/arquivos/pdf/book_volume_02_internet.pdf>. Acesso em 12 Mar 2023.

BRASIL. Ministério da Educação. **Base Nacional Comum Curricular**. Brasília, 2018.

CARMARGO, F.; DAROS, T. **A sala de aula inovadora**: estratégias pedagógicas para fomentar o aprendizado ativo. Porto Alegre: Editora Penso, 2018.

GEOGEBRA.ORG. **Geogebra**. Disponível em:<https://www.geogebra.org/>. Acesso em 23 Mai 2023.

IEZZI, G.; MURAKAMI, C. **Fundamentos de matemática elementar 1**: conjuntos, funções — 9. ed. — São Paulo : Atual, 2013.

LIBÂNEO, J. C. **Didática.** São Paulo: Editora CORTEZ, 1990.

MATTHES, E. **Curso Intensivo de Python**: Uma introdução prática e baseada em projetos à programação. 1ª ed. São Paulo: Editora Novatec, 2016.

MATHWORKS. **MATLAB Documentation**. Disponível em:<https://www.mathworks.com/help/matlab/>. Acesso em 26 Jul 2023.

MAPLESOFT. **Maple**. Disponível em:<https://www.maplesoft.com/>. Acesso em 24 Mai 2023.

MIT MEDIA LAB. **Scratch**. Disponível em:<https://scratch.mit.edu/>. Acesso em 23 Mai 2023.

PAPERT, S. **Logo:** computadores e educação. Tradução de José A. Valente. São Paulo: Editora Brasiliense, 1985.

PYTHON.ORG. **Python Documentation**. Disponível em:<https://www.python.org/doc/>. Acesso em 27 Mai 2023.

SAGEMATH.ORG. **SageMath Documentation**. Disponível em:<https://doc.sagemath.org/>. Acesso em 11 Fev 2023.

SAGEMATH.ORG. **SageMathCell**. Disponível em:<https://sagecell.sagemath.org/>. Acesso em 15 Fev 2023.

SAGEMATH.ORG. **CoCalc**. Disponível em:<https://cocalc.com/features/sage>. Acesso em 04 Mar 2023.

SILVA, Leon; MACHADO, Ricardo; SANTOS, Marcelo. **Elementos de Computação Matemática com SageMath**. 1ª Edição. Rio de Janeiro: SBM, 2019.

SOUSA, RP., MIOTA, FMCSC., and CARVALHO, ABG., orgs. **Tecnologias digitais na educação [online]**. Campina Grande: EDUEPB, 2011. ISBN 978-85-7879-124-7. Available from SciELO Books <http://books.scielo.org >.

A Gabarito - Sequência Didática

A.1 Respostas - Exercícios da aula 1

1. Utilize o comando `set( )` para definir o conjunto $A = \{7x \ / \ x \in \mathbb{N} \text{ e } 1 \leq x \leq 10\}$.

```
In [22]: A = set([7, 14, 21, 28, 35, 42, 49, 56, 63, 70]) # Múltiplos de 7 inseridos manualmente;
         A

Out[22]: {7, 14, 21, 28, 35, 42, 49, 56, 63, 70}

In [23]: A = set([7*x for x in [1..10]]) # (ALTERNATIVA) - Múltiplos de 7 inseridos com o laço "for";
         A

Out[23]: {7, 14, 21, 28, 35, 42, 49, 56, 63, 70}
```

2. Sejam B e C, respectivamente, os conjuntos dos divisores primos de 30 e de 42. Verifique:

```
In [49]: B = set([2,3,5]) # Divisores encontrados manualmente;
         B

Out[49]: {2, 3, 5}

In [50]: B = set(prime_divisors(30)) # Divisores encontrados com o código prime_divisors(30);
         B

Out[50]: {2, 3, 5}

In [51]: C = set([2,3,7]) # Divisores encontrados manualmente;
         C

Out[51]: {2, 3, 7}

In [52]: C = set(prime_divisors(42)) # Divisores encontrados com o código prime_divisors(42);
         C

Out[52]: {2, 3, 7}
```

   (a) $B \cap C$;

   (b) $B \cup C$;

   (c) $B \setminus C$.

```
In [57]: B.union(C) # Resposta do item a;
Out[57]: {2, 3, 5, 7}

In [58]: B.intersection(C) # Resposta do item b;
Out[58]: {2, 3}

In [59]: B.difference(C) # Resposta do item c;
Out[59]: {5}
```

3. Verifique se para os conjuntos $A = \{1, 2, 3\}$ e $B = \{2, 4, 6\}$ as afirmações abaixo são verdadeiras:

   (a) $(A \cup B) = (B \cup A)$;

   (b) $(A \cap B) = (B \cap A)$;

   (c) $(A \setminus B) = (B \setminus A)$.

```
In [60]: A = set([1,2,3])
         B = set([2,4,6])

In [61]: A.union(B) == B.union(A) # A relação (A ∪ B = B ∪ A) verificada como verdadeira;
Out[61]: True

In [62]: A.intersection(B) == B.intersection(A) # A relação (A ∩ B = B ∩ A) verificada como verdadeira;
Out[62]: True

In [63]: A.difference(B) == B.difference(A) # A relação (A - B = B - A) verificada como falsa.
Out[63]: False
```

4. Utilize o comando `set( )` para determinar os conjuntos A, B e C de modo que: $A = \{x \in \mathbb{N} \ / \ 1 \leq x \leq 4\}$; $B = \{y \in \mathbb{N} \ / \ -3 \leq x \leq 6\}$ e $C = \{z \in \mathbb{N} \ / \ 3 \leq x < 8\}$;

```
In [66]: A = set([1,2,3,4]) # x natural / x ∈ [1,4];
         B = set([0,1,2,3,4,5,6]) # x natural / x ∈ [0,6];
         C = set([3,4,5,6,7]) # x natural / x ∈ [3,8);
```

5. O comando `A = set([x**2 for x in [1,11]])`, constrói o conjunto $A = \{x^2 \ / \ x \in \mathbb{Z} \text{ e } 1 \leq x < 11\}$, ou seja, $A = \{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$.

   (a) Analisando o comando que construiu o conjunto A com os 10 primeiros números quadrados perfeitos, construa o conjunto B com os números da forma 2^{2x+1} para $x \in \mathbb{N}$ no intervalo $[1, 5]$.

   (b) Coloque um número no comando "…. `in B`" de modo que o resultado na saída seja `True`.

In [67]:
```
B = set([2^(2*x+1) for x in [1..5]]) # o laço for ao lado, cria uma lista com os valores de 2
B
```

Out[67]:
```
{8, 32, 128, 512, 2048}
```

In [68]:
```
128 in B # Basta escolher um dos elementos de B e verificar a pertinência com o método "in".
```

Out[68]:
```
True
```

A.2 Respostas - Exercícios da aula 2

1. Após declarar as variáveis a, b e c com `var('a b c')`, com o comando `solve( )`, verifique que para a, b e c reais:

   (a) A equação da forma $ax + b = 0$ possui solução: $x = \frac{-b}{a}$;

```
In [1]: var('a b c')
Out[1]: (a, b, c)

In [2]: solve(a*x + b == 0, x) # Mostra a solução de uma equação do tipo ax + b = 0;
Out[2]: [x == -b/a]

In [3]: show(solve(a*x + b == 0, x)) # Mostra a solução de uma equação do tipo ax + b = 0
```

$$\left[x = -\frac{b}{a}\right]$$

   (b) A equação da forma $ax^2 + bx + c = 0$ possui solução: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

```
In [4]: solve(a*x^2 + b*x + c == 0, x) # Mostra as soluções de uma equação do tipo ax^2 + bx + c = 0;
Out[4]: [x == -1/2*(b + sqrt(b^2 - 4*a*c))/a, x == -1/2*(b - sqrt(b^2 - 4*a*c))/a]

In [5]: show(solve(a*x^2 + b*x + c == 0, x)) # Mostra as soluções da eq com o comando show()
```

$$\left[x = -\frac{b + \sqrt{b^2 - 4\,ac}}{2\,a}, x = -\frac{b - \sqrt{b^2 - 4\,ac}}{2\,a}\right]$$

2. Dada a equação $x^3 + x - 2 = 0$.

   (a) Armazene as soluções da equação na variável `S`;

```
In [6]: A = solve(x^3 + x - 2 == 0, x) # Armazena as soluções da equação x^3 + x - 2 = 0;
```

   (b) Execute o comando `S`, "variável que armazenou a função `solve( )`". Em seguida, o comando `show(S)`;

```
In [8]: S = solve(x^3 + x - 2 == 0, x) # Armazena as soluções da equação x^3 + x - 2 = 0;

In [9]: S # Mostra as soluções da equação x^3 + x - 2 = 0;
Out[9]: [x == -1/2*I*sqrt(7) - 1/2, x == 1/2*I*sqrt(7) - 1/2, x == 1]

In [10]: show(S) # Mostra as soluções da equação x^3 + x - 2 = 0 com o comando show(S);
```

$$\left[x = -\frac{1}{2}i\sqrt{7} - \frac{1}{2}, x = \frac{1}{2}i\sqrt{7} - \frac{1}{2}, x = 1\right]$$

   (c) Mostre apenas a raiz real através de seu índice no operador `[ ]`.

```
In [11]: # Observe que a raiz real é o terceiro item da lista, para acessá-lo utilizamos o operador [3-1]
         S[2] # Mostra a raiz real da equação x^3 + x - 2 = 0 através do operador[índice].
Out[11]: x == 1
```

3. A fatoração e expansão de expressões algébricas são técnicas essenciais para a observação do comportamento das funções e resolução de problemas. Então:

   (a) Associe a expressão algébrica $x^3 - 2x^2 - 5x + 6$ a variável comum `eq1`. Execute o código `.factor( )` em `eq1` para mostrar a forma fatorada da expressão.

```
In [15]: eq1 = x^3 - 2*x^2 - 5*x + 6 == 0

In [16]: eq1.factor() # Fatora a equação x^3 - 2*x^2 - 5*x + 6 = 0;
Out[16]: (x + 2)*(x - 1)*(x - 3) == 0
```

   (b) Associe a expressão algébrica $(x + 2)(x - 1)(x - 3)$ a variável comum `eq2`. Execute o código `.expand( )` em `eq2` para mostrar a forma expandida da expressão.

```
In [17]: eq2 = (x + 2)*(x - 1)*(x - 3) == 0

In [18]: eq2.expand()
Out[18]: x^3 - 2*x^2 - 5*x + 6 == 0
```

   (c) Utilize o código `.rhs( )` em `eq1` para mostrar que $x_1 + x_2 + x_3 = 2$.

```
In [12]: A = solve(x^3 - 2*x^2 - 5*x + 6 == 0, x) #Armazena as soluções na variável A
         # A[0].rhs() Acessa o lado direito (valor numérico) da primeira solução;
         # A[1].rhs() Acessa o lado direito (valor numérico) da segunda solução;
         # A[2].rhs() Acessa o lado direito (valor numérico) da terceira solução;
         A[0].rhs() + A[1].rhs() + A[2].rhs() # Soma o lado direito (valor numérico)
Out[12]: 2
```

4. Dado o sistema de equações:

$$\begin{cases} x + y &= 6 \\ x - y &= 1 \end{cases}$$

   (a) Resolva o sistema com o comando `solve( )`;

```
In [14]: var('y')
         eq1 = x + y == 6 # Armazena x + y == 5 na variável eq1;
         eq2 = x - y == 1 # Armazena x - y == 1 na variável eq2;
         solve([eq1, eq2], x, y) # Observe as soluções são fornecidas em uma lista dentro de outra lista.
Out[14]: [[x == (7/2), y == (5/2)]]

In [15]: show(solve([eq1, eq2], x, y)) # opcional
```

$$\left[\left[x = \left(\frac{7}{2}\right), y = \left(\frac{5}{2}\right)\right]\right]$$

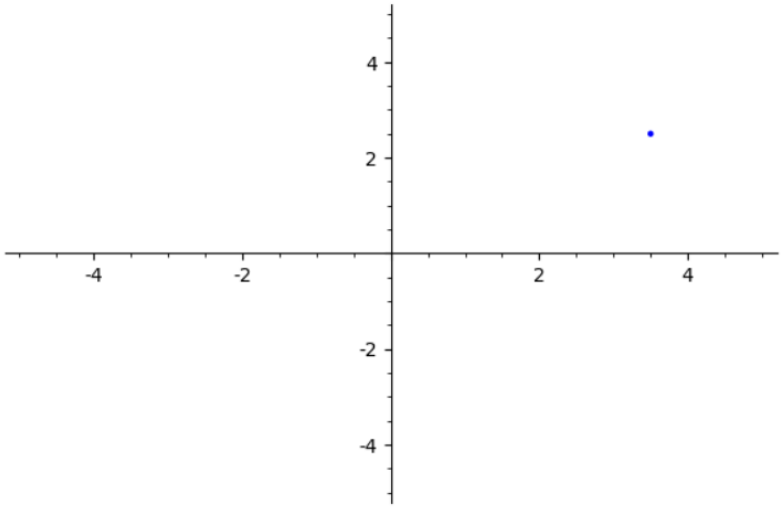
   (b) Associe a variável `P` à solução do sistema.

```
In [18]: A = solve([eq1, eq2], x, y)
         # A[0] acessa a lista_interna [x == (7/2), y == (5/2)] e A[0][0] acessa o 1º elemento: x == (7/2);
         # A[0][0].rhs() acessa o valor numérico de x == (7/2), analogamente para A[0][1].rhs();
         P = (A[0][0].rhs(), A[0][1].rhs()) # Armazena em P o par ordenado (7/2, 5/2).
         P
Out[18]: (7/2, 5/2)
```

(c) Execute o comando: `point(P, xmin = -5, xmax = 5, ymin =- 5, ymax = 5)`.

```
In [23]: point(P, xmin = -5, xmax = 5, ymin =- 5, ymax = 5)
```

Out[23]:



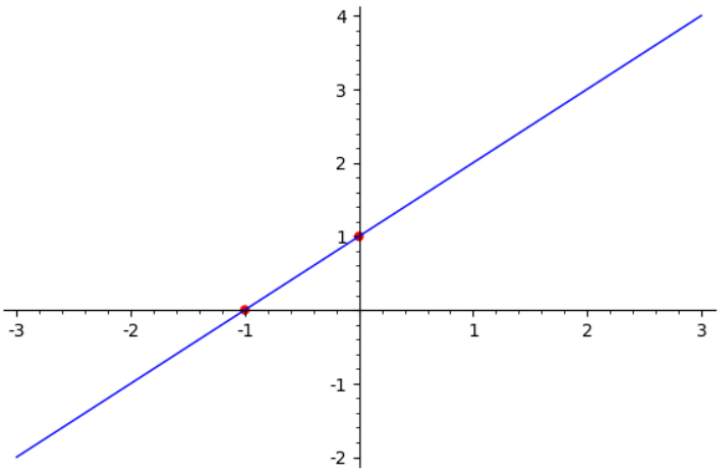
A.3 Respostas - Exercícios da aula 3

1. Com o comando `plot( )`, construa:

   (a) O gráfico de f, uma função do 1^{o} grau para x no intervalo $[-3, 3]$, que intersecte os eixos do plano cartesiano em $(-1, 0)$ e $(0, 1)$.

```
In [37]: # Para f = ax + b, temos: -a + b = 0 e b = 1, então a = 1. Logo, f = x + 1;
         f = x+1
         plot(x+1, -3, 3) + point([(-1,0), (0,1)], size=30, color='red')
```

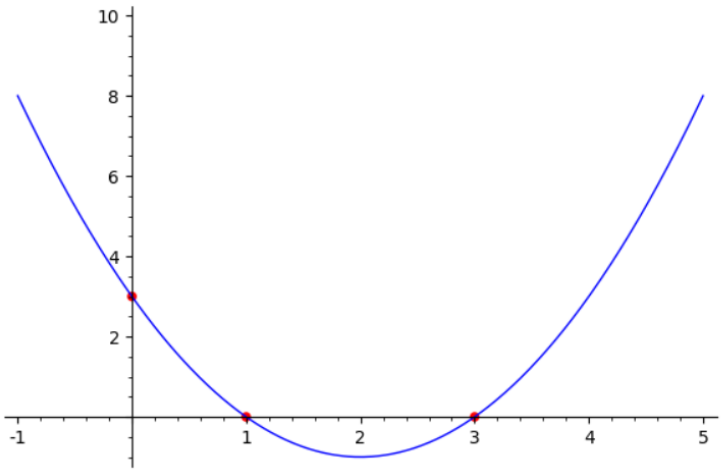
Out[37]:



   (b) O gráfico de g, uma função do 2^{o} grau para x no intervalo $[-1, 5]$ que intersecte os eixos do plano cartesiano em $(0, 3)$, $(1, 0)$ e $(3, 0)$.

```
In [38]: # Para g = ax^2 + bx + c, temos: c = 3, a + b + 3 = 0 e 9a + 3b + 3 = 0, então a = 1 e b = -4.
         g = x^2 - 4*x + 3
         plot(g, -1, 5, ymax = 10) + point([(0,3),(1,0), (3,0)], size=30, color='red')
```

Out[38]:

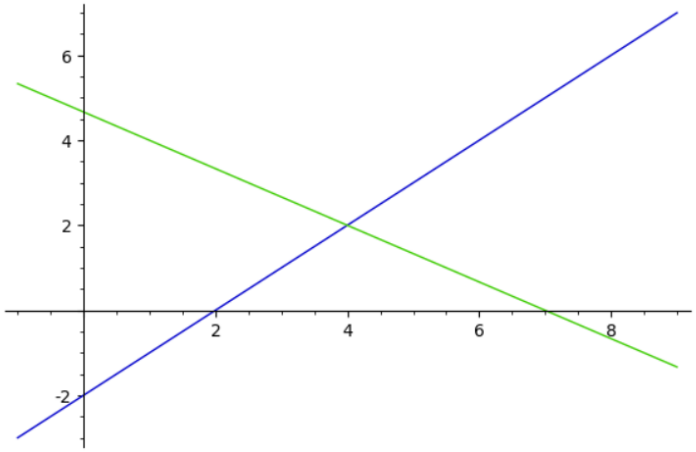


2. Dado o sistema de equações: $\begin{cases} x - y & = 2 \\ 2x + 3y & = 14 \end{cases}$

   (a) Mostre a representação gráfica das duas retas no mesmo plano cartesiano;

In [46]:
```
f = x - 2 # x - y = 2 ==> y = x - 2;
g = (14 - 2*x)/3 # 2x + 3y = 14 ==> 3y = 14 - 2x ==> y = (14 - 2x)/3;
plot([f,g], -1, 9)
```

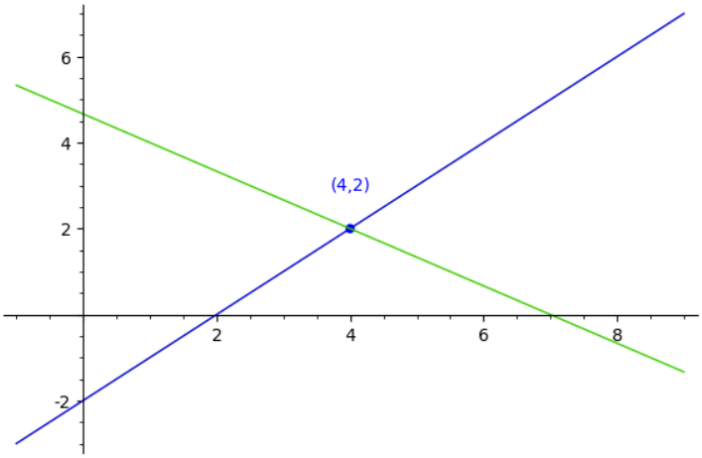
Out[46]:



(b) Indique a solução do sistema com comandos `point( )` e `text( )`;

In [53]:
```
plot([f,g], -1, 9) + point((4,2), size = 30) + text('(4,2)', (4,3))
```

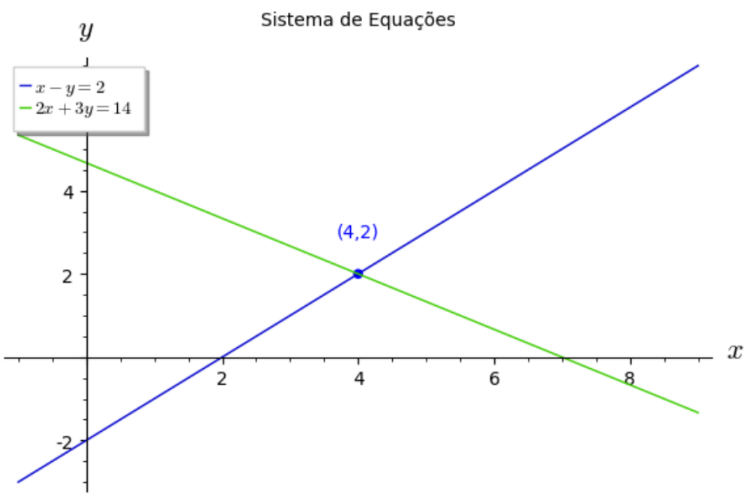
Out[53]:



(c) Acrescente os parâmetros `title`, `legend_label` e `axes_labels` ao gráfico.

In [62]:
```
A = plot([f,g], -1, 9, title='Sistema de Equações',
         legend_label=['$x - y = 2$', '$2x + 3y = 14$'], axes_labels =['$x$', '$y$'])
A + point((4,2), size = 30) + text('(4,2)', (4,3))
```

Out[62]:



3. Seja a função $f(x) = -x^2 + 3x + 1$.

   (a) Use o comando `solve( )` para encontrar as raízes de f;

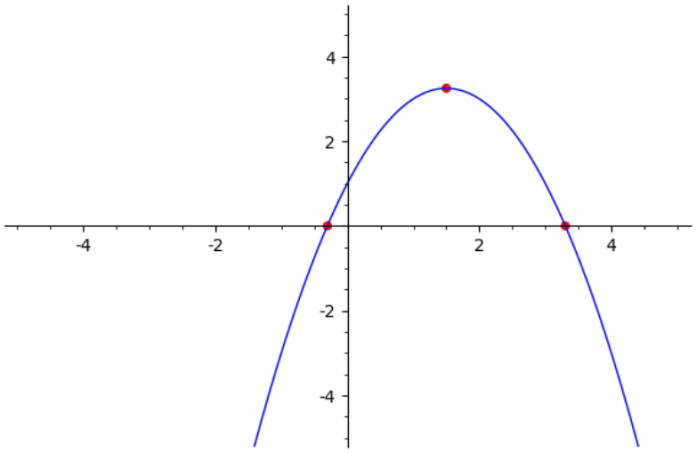
```
In [25]: f(x) = -x^2 + 3*x + 1 # Define f.

In [28]: A = solve(f,x) # Atribui as soluções de f à variável A através do comando solve(f,x);
         A # Executa o comando solve(f,x).

Out[28]: [x == -1/2*sqrt(13) + 3/2, x == 1/2*sqrt(13) + 3/2]
```

   (b) Escreva um código que armazene e mostre as coordenadas do vértice de f na variável V;

```
In [33]: var('a b') # Define as variáveis a e b;
         a = -1 # Atribui valor -1 para a variável a;
         b = 3 # Atribui valor 3 para a variável b;
         Vx = -b/(2*a) # Atribui valor -b/2a para a variável Vx;
         V = (Vx, f(Vx)) # Atribui as cordenadas do vértice para a coordenada V.
         V # Mostra as coordenadas do vértice de f.

Out[33]: (3/2, 13/4)
```

   (c) Plote o gráfico de f e concatene com o comando `point( )`, de modo que evidencie as raízes e o vértice da função. "utilize os parâmetros `size = 30` e `color ='red'` em `point( )`".

```
In [32]: x1 = A[0].rhs() # Atribui o valor numérico da primeira solução da lista à variável x1;
         x2 = A[1].rhs() # Atribui o valor numérico da segunda solução da lista à variável x2;
         plot(f, -5, 5, ymin = -5, ymax = 5) + point([(x1,f(x1)), (x2, f(x2)), V], size = 30, color = 'red')
         # Para melhor visualização, foi atribuído os parâmetros -5, 5, ymin = -5, ymax = 5 à plot() e
         # size = 30, color = 'red' à point()

Out[32]:
```



4. Considere uma função $f : \mathbb{R} \to \mathbb{R}$ dada pela expressão $f(x) = -x^2 + bx + c$, em que b e c são reais, cujo gráfico tem eixo de simetria na reta $x = 1$ e a diferença entre as raízes igual a -4. Construa o gráfico que representa f.

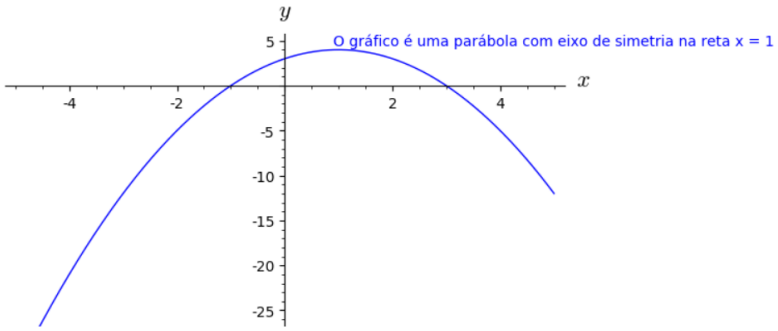
In [5]:
```
var('b c') # Declara as variáveis b e c;
f(x) = -x^2 + b*x + c # Declara f;
# Temos: -b/2a = 1 ==> -b/(2.(-1)) = 1 ==> -b/-2 = 1 ==> b = 2, então
f(x) = -x^2 + 2*x + c # Declara f após encontrar o valor de b;
```

In [6]:
```
var('x1 x2') # Declara as variáveis x1 e x2 (zeros de f);
eq1 = (x1 + x2)/2 == 1 # Equação 1 no sistema de equações;
eq2 = x1 - x2 == -4 # Equação 2 no sistema de equações;
solve([eq1, eq2], x1, x2) # Solução do sistema.
```

Out[6]:
```
[[x1 == -1, x2 == 3]]
```

In [8]:
```
# c/a = x1.x2 ==> c/-1 = (-1).3 ==> c = 3;
f(x) = -x^2 + 2*x + 3 # Declara f após encontrar o valor de c e finaliza com a plotagem na linha de código abaixo
plot(f, -5, 5, axes_labels=['$x$', '$y$']) + text('O gráfico é uma parábola com eixo de simetria na reta x = 1', (5,5))
```

Out[8]:



A.4 Respostas - Exercícios da aula 4

1. Os modelos de crescimento populacional de Malthus, são os mais simples para descrever o crescimento de uma população ao longo do tempo. A função $f(t) = t_0.e^{kt}$ representa o modelo contínuo com taxa de crescimento instantânea (exponencial - maior precisão) e $g(t) = t_0.(1 + k)^t$, o modelo discreto com taxa de crescimento anual (linear por etapas - mais simples), onde:
   $f(t)$ e $g(t)$ representam a população total no instante t;
   k, a taxa de crescimento populacional;
   t_0, a população no instante inicial.

   Para responder os items abaixo, considere o estudo de uma cultura de bactérias em um ambiente ideal para o crescimento populacional, inicialmente com 100 bactérias a taxa de crescimento de 40% a cada 30 minutos.

```
In [60]: var('t0, t, k') # Declara as variáveis utilizadas nas funções
         f(t)=t0*e^(k*t) # Define f(t), crescimento populacionalno modelo contínuo;
         f(t)=100*e^(0.4*t) # Define f(t) quando t0 = 100 e k = 0.4;
         g(t)=t0*(1+k)^t # Define g(t), crescimento populacionalno modelo discreto;
         g(t)=100*(1+0.4)^t # Define g(t) quando t0 = 100 e k = 0.4.
```

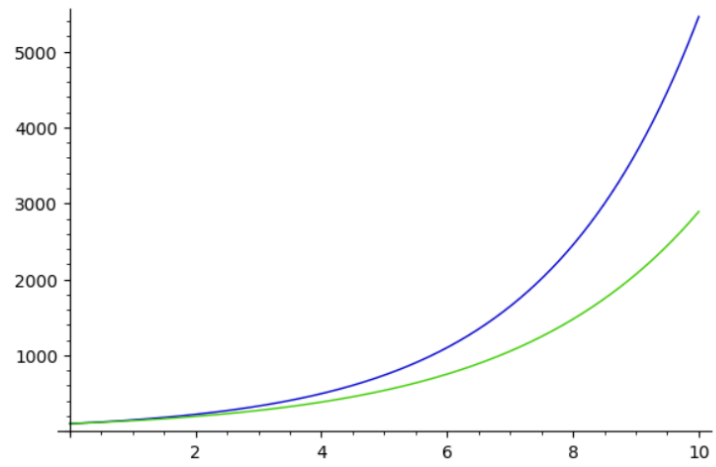
   (a) Qual a população de bactérias em cada um dos modelos após uma, duas e três horas?

```
In [61]: f(2) # Quantidade de bactérias após 1 hora (2 ciclos) no modelo contínuo;
Out[61]: 222.554092849247

In [62]: f(4) # Quantidade de bactérias após 2 horas (4 ciclos) no modelo contínuo;
Out[62]: 495.303242439512

In [63]: f(6) # Quantidade de bactérias após 3 horas (6 ciclos) no modelo contínuo;
Out[63]: 1102.31763806416

In [64]: g(2) # Quantidade de bactérias após 1 hora (2 ciclos) no modelo discreto;
Out[64]: 196.000000000000

In [65]: g(4) # Quantidade de bactérias após 2 horas (4 ciclos) no modelo discreto;
Out[65]: 384.160000000000

In [66]: g(6) # Quantidade de bactérias após 3 horas (6 ciclos) no modelo discreto.
Out[66]: 752.953600000000
```

(b) Crie uma representação gráfica com o comando `plot( )`, que apresente a comparação entre os modelos discreto e o contínuo de Malthus, representando o crescimento populacional dessa cultura durante um intervalo de 5 horas.

```
In [46]: plot([f,g],0,10)
```

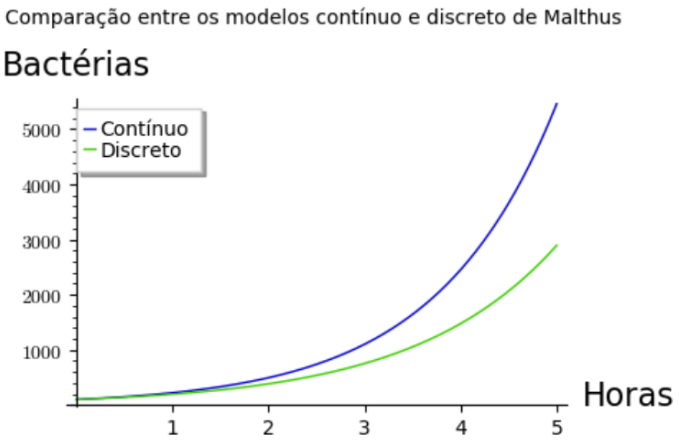
Out[46]:



(c) Acrescente os parâmetros: `axes_labels`, `legend_label`, `title`, `ticks` e `tick_formatter`, inserindo as informações necessárias ao gráfico.

```
In [12]: plot([f,g],0,10,ticks=[[2,4..10],1000],tick_formatter=[[1,2..5],1000], figsize=5,
             axes_labels=['Horas','Bactérias'], legend_label=['Contínuo', 'Discreto'],
             title='Comparação entre os modelos contínuo e discreto de Malthus \n \n')
```

Out[12]:

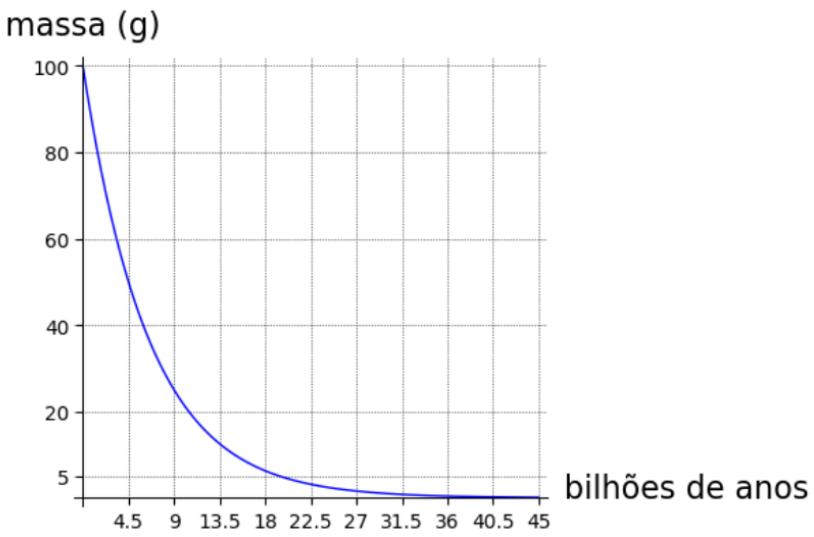


2. Dados 100 gramas de uma amostra de urânio-238, o intervalo de tempo em que uma amostra deste elemento se reduz à metade (meia-vida) é de $4,5$ bilhões de anos.

(a) Crie uma representação gráfica com os parâmetros: `axes_labels`, `legend_label`, `ticks` e `gridlines`, mostrando o decaimento radioativo da quantidade de urânio-238 restante ao longo de 45 bilhões de anos.
Utilize a função $f(t) = t_0 e^{-kt}$.

```
In [13]: var('t0, k')
         f(t) = t0*e^(-k*t)
         f(t) = 100*e^(-k*t)
         # Para f(4.5) = 50, temos que k = (ln 2)/4.5. Então:
         f(t)=100*e^((-log(2)/4.5)*t)

In [15]: plot(f,0,45, axes_labels=['bilhões de anos', 'massa (g)'],
              gridlines='normal', ticks=[[4.5,9..45],[5,20,40..100]], figsize=6)
```

Out[15]:



(b) Após quanto tempo a quantidade do elemento será reduzida a 5 gramas? Mostre geometricamente a resposta com um ponto vermelho no gráfico.

```
In [15]: sol = solve(f(t)==5, t) # Tempo em que a massa seja reduzida para 5 gramas;
         sol
```

Out[15]: `[t == 9/2*(log(5) + 2*log(2))/log(2)]`

```
In [16]: t1 = sol[0].rhs() # Acessando ao valor de t quando f(t) = 5;
         t1
```

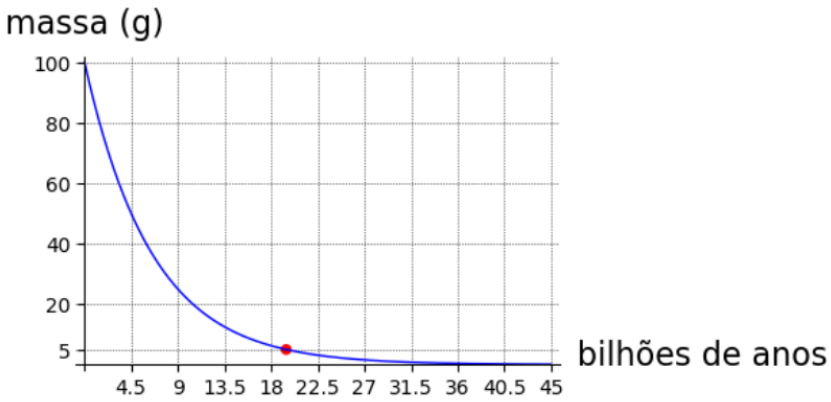
Out[16]: `9/2*(log(5) + 2*log(2))/log(2)`

```
In [17]: n(t1) #A função n(t1) traz o valor numérico da expressão anterior.
```

Out[17]: `19.4486764269931`

In [22]:
```
p = plot(f,0,45, axes_labels=['bilhões de anos', 'massa (g)'], gridlines='normal',
         ticks=[[4.5,9..45],[5,20,40..100]], figsize=(6,3))
p + point((t1,f(t1)), size=30, color='red')
```

Out[22]:



A.5 Respostas - Exercícios da aula 5

1. Apresente a solução da equação $\log_2 3x = 7$, nos seguintes passos:

   (a) Utilize o comando `solve( )` e armazene a solução na variável s;

   ```
   In [110]: log(3*x, 2) == 7
             s = solve(log(3*x, 2) == 7,x) # Armazena a solução na variável s.
             s
   Out[110]: [x == (128/3)]
   ```

   (b) Apresente o resultado utilizando o método `[índice].rhs( )` para acessar o lado direito da equação.

   ```
   In [111]: s[0].rhs()
   Out[111]: 128/3
   ```

   (c) Mostre a aproximação do resultado com quatro casas decimais. Use o método `.n( )` e o parâmetro `digits`.

   ```
   In [114]: s[0].rhs().n(digits=4)
   Out[114]: 42.67
   ```

2. A velocidade máxima, em *bits* por segundo, com a qual os sinais podem passar por canais de comunicação, é obtida por meio da fórmula: $v_{mx} = 3400 \log_2(x+1)$, onde 3400 *hertz* é a frequência limite da voz humana e x é a potência do sinal.

   Calcule o valor de x correspondente a uma velocidade máxima de 27 200 bits por segundo.

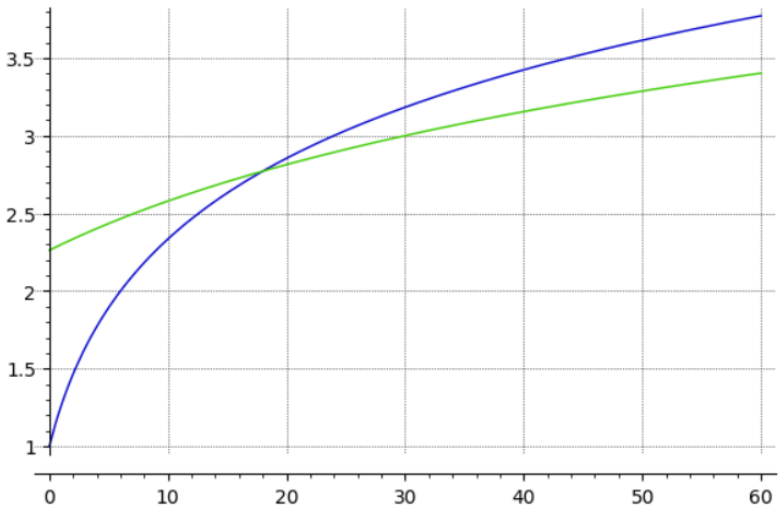
   ```
   In [115]: var('vmax')
             vmax == 3400*log(x+1,2)
             27200 == 3400*log(x+1,2)
             s = solve(27200 == 3400*log(x+1,2), x)
             s
   Out[115]: [x == 255]
   ```

3. O ICMBio – Instituto Chico Mendes de Conservação da Biodiversidade, ligado ao Ministério do Meio Ambiente, em parceria com unidades de conservação federais, fomenta recursos para a proteção e mapeia a evolução populacional das espécies F e G ameaçadas de extinção. O repovoamento desses animais na fauna brasileira é descrito, aos milhares, pelas funções $f(t) = \log_3(t+3)$ para a espécie F e $g(t) = \log_3(0.5t + 12)$ para G, onde t é o tempo em anos.

(a) Construa os gráficos de f e g no mesmo plano cartesiano com o domínio no intervalo $[0, 60]$ e acrescente o parâmetro `gridlines = 'normal'`;

```
In [6]: f(t) = log(t+3, 3)
        g(t) = log(0.5*t+12, 3)
        plot([f, g], 0, 60, gridlines='normal')
```

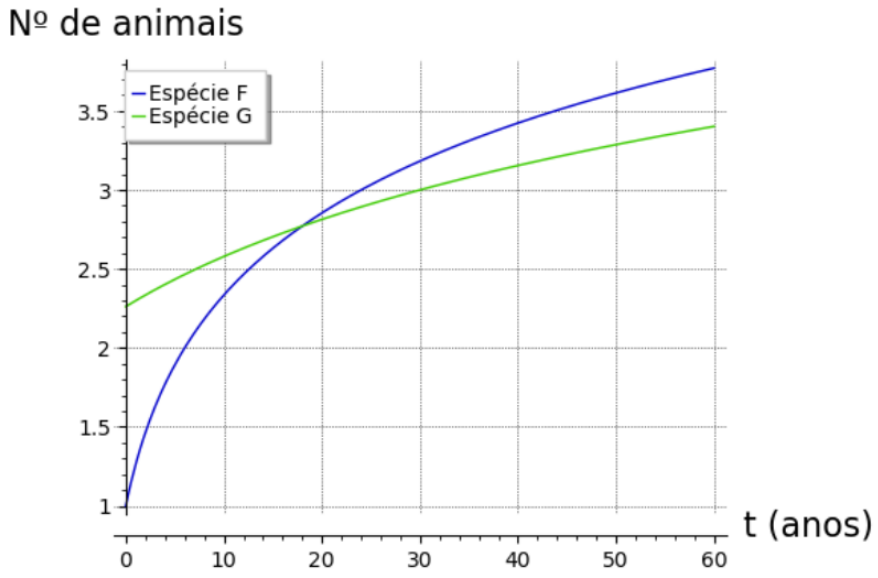
Out[6]:



(b) Insira rótulos nos eixos do gráfico (`axes_labels`) e legenda para as espécies F e G (`legend_label`);

```
In [23]: f(t) = log(t+3, 3)
         g(t) = log(0.5*t+12, 3)
         plot([f, g], 0, 60, gridlines='normal', axes_labels=['t (anos)', 'Nº de animais'],
              legend_label = ['Espécie F', 'Espécie G'])
```

Out[23]:



(c) Qual a previsão da quantidade desses animais no Brasil, após 30 anos do início da ação que protege a reprodução dessas espécies?

```
In [14]: f(30) # Quantidade de animais da espécie F após 30 anos;
Out[14]: log(33)/log(3)

In [15]: f(30).n(digits=4) # Aproximadamente 3183 animais da espécie F;
Out[15]: 3.183

In [16]: g(30) # Quantidade de animais da espécie G após 30 anos;
Out[16]: 3.29583686600433/log(3)

In [17]: g(30).n(digits=4) # Aproximadamente 3000 animais da espécie G
Out[17]: 3.000
```