UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUÍS FERNANDO ALEXANDRE DOS SANTOS

FERRAMENTA COMPUTACIONAL PARA ANÁLISE E CARACTERIZAÇÃO DE SISTEMAS DE CONTROLE UTILIZANDO PYTHON

PAU DOS FERROS - RN
2025

LUÍS FERNANDO ALEXANDRE DOS SANTOS

FERRAMENTA COMPUTACIONAL PARA ANÁLISE E CARACTERIZAÇÃO DE SISTEMAS DE CONTROLE UTILIZANDO PYTHON

Monografia apresentada à Universidade Federal Rural do Semi-Árido como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Cecílio Martins de Souza Neto

PAU DOS FERROS - RN
2025

S237f Santos, Luís Fernando Alexandre dos.
Ferramenta computacional para análise e caracterização de sistemas de controle utilizando python / Luís Fernando Alexandre dos Santos. - 2025.
60 f. : il.

Orientador: Cecílio Martins de Souza Neto.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Engenharia de Computação, 2025.

1. Sistemas de controle. 2. Python. 3. Ferramenta didática. 4. Análise de estabilidade. 5. Controladores PID. I. Souza Neto, Cecílio Martins de, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

LUÍS FERNANDO ALEXANDRE DOS SANTOS

FERRAMENTA COMPUTACIONAL PARA ANÁLISE E CARACTERIZAÇÃO DE SISTEMAS DE CONTROLE UTILIZANDO PYTHON

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Engenharia de Computação.

Defendida em: 12/12/2025.

BANCA EXAMINADORA

Prof. Dr. Cecílio Martins de Sousa Neto
Universidade Federal Rural do Semi-Árido (UFERSA)
(Presidente)

Prof. Dr. Pedro Thiago Valério de Souza
Universidade Federal Rural do Semi-Árido (UFERSA)
(Membro Examinador)

Prof. Dr. Ádller de Oliveira Guimarães
Universidade Federal Rural do Semi-Árido (UFERSA)
(Membro Examinador)

DEDICATÓRIA

Dedico este trabalho à memória de minha querida avó, Jerônima Vieira (in memoriam), carinhosamente conhecida como Jeroma, que me criou com amor, firmeza e uma dedicação que marcou profundamente minha vida. Durante os meus primeiros dezessete anos, esteve ao meu lado oferecendo cuidado, orientação e valores que moldaram o homem que sou hoje. Sua força, simplicidade e exemplo permanecem vivos em cada conquista que alcanço. Este trabalho é, também, fruto de tudo o que aprendi com ela.

Dedico igualmente este trabalho aos meus pais e aos meus avós, que sempre acreditaram em mim, me apoiaram nos momentos mais difíceis e celebraram cada pequeno avanço. Sem o amor, o incentivo e os valores que recebi, este sonho não seria hoje uma realidade.

AGRADECIMENTOS

A Deus, princípio de todas as coisas, agradeço pela dádiva da vida, pela sabedoria nos momentos de incerteza e por permitir que eu transformasse este sonho em realidade.

Aos meus pais, Adriana e Arinaldo, expresso minha eterna gratidão. Vocês são meus alicerces; obrigado pelo amor incondicional, pelo incentivo diário e por cada sacrifício feito para que eu pudesse chegar até aqui.

À memória de minha avó Jerônima, carinhosamente chamada de Jeroma. Minha maior referência de simplicidade e determinação, foi ela quem, com mãos dedicadas, guiou meus primeiros passos e moldou meu caráter. Sua presença se faz sentir em cada linha desta conquista.

À minha esposa, Lídia Andrade, agradeço com todo o meu coração. Obrigado por ser meu porto seguro, pela compreensão durante as longas horas de estudo e por sempre renovar minhas forças com palavras de ânimo. Esta vitória também é sua.

Ao meu orientador e amigo, Professor Dr. Cecílio, pela orientação de excelência. Agradeço pela paciência, pela confiança depositada em meu trabalho e pelos ensinamentos que transcendem a sala de aula e levarei para a vida profissional.

Aos meus amigos e colegas de jornada, obrigado pela parceria, pelos gestos de solidariedade e pelo apoio mútuo nos momentos mais desafiadores da graduação. Vocês tornaram a caminhada mais leve.

A todos que, direta ou indiretamente, contribuíram para a concretização deste trabalho, deixo aqui o meu sincero muito obrigado.

Não espere o futuro mudar tua vida, porque o futuro é a consequência do presente.

— Racionais MC's (Música: A Vida É Desafio)

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta computacional didática, de código aberto e multiplataforma, voltada à análise e caracterização de sistemas lineares de controle utilizando Python. A aplicação, implementada em Python com interface gráfica construída por meio da biblioteca *CustomTkinter*, integra quatro módulos fundamentais: (1) análise de estabilidade pelo Critério de Routh–Hurwitz; (2) caracterização automática de sistemas de segunda ordem, com extração de parâmetros como frequência natural (ω_n), coeficiente de amortecimento (ζ) e ganho estático (K), além do cálculo de métricas de desempenho transitório; (3) geração do Lugar Geométrico das Raízes (LGR), com caracterização de polos e zeros, segmentos, assíntotas, pontas de saída e entrada sobre o eixo real, pontos de cruzamento com o eixo imaginário e/ou ângulos de partida de um polo complexo e de chegada em um zero complexo; e (4) simulação interativa de controladores clássicos Proporcional e Derivativo (PD), Proporcional e Integral (PI), e Proporcional, Integral e Derivativo (PID), permitindo ajuste em tempo real dos ganhos, comparação de respostas ao degrau, análise de estabilidade e verificação do atendimento a especificações de projeto baseadas em polos desejados. A ferramenta foi validada por meio de exemplos clássicos da literatura de sistemas de controle, demonstrando a interface interativa e acessível ao usuário. Ao oferecer uma solução gratuita, intuitiva e pedagogicamente orientada, o projeto contribui para a democratização do ensino de engenharia de controle, especialmente em ambientes acadêmicos com recursos computacionais limitados.

Palavras-chave: Sistemas de controle; Python; Ferramenta didática; Análise de estabilidade; Controladores PID.

ABSTRACT

This work presents the development of a didactic, open-source, and cross-platform computational tool aimed at the analysis and characterization of linear control systems using Python. The application, implemented in Python with a graphical interface built using the *CustomTkinter* library, integrates four fundamental modules: (1) stability analysis using the Routh–Hurwitz Criterion; (2) automatic characterization of second-order systems, extracting parameters such as natural frequency (ω_n), damping ratio (ζ), and static gain (K), along with the calculation of transient performance metrics; (3) generation of the Root Locus (RL), with characterization of poles and zeros, segments, asymptotes, breakaway and break-in points on the real axis, imaginary axis crossing points, and/or angles of departure from a complex pole and arrival at a complex zero; and (4) interactive simulation of classic Proportional-Derivative (PD), Proportional-Integral (PI), and Proportional-Integral-Derivative (PID) controllers, allowing real-time gain adjustment, step response comparison, stability analysis, and verification of compliance with design specifications based on desired poles. The tool was validated using classic examples from control systems literature, demonstrating an interactive and user-friendly interface. By offering a free, intuitive, and pedagogically oriented solution, the project contributes to the democratization of control engineering education, especially in academic environments with limited computational resources.

Keywords: Control systems; Python; Educational tool; Stability analysis; PID controllers.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GUI	Interface Gráfica do Usuário (*Graphical User Interface*)
LGR	Lugar Geométrico das Raízes
SLIT	Sistema Linear Invariante no Tempo
P	Controlador Proporcional
PI	Controlador Proporcional–Integral
PD	Controlador Proporcional–Derivativo
PID	Controlador Proporcional–Integral–Derivativo
Tr	Tempo de Subida (*Rise Time*)
Tp	Tempo de Pico (*Peak Time*)
Ts	Tempo de Acomodação (*Settling Time*)
Mp	Máximo Sobressinal (*Maximum Overshoot*)
e_{ss}	Erro em Regime Permanente
UFERSA	Universidade Federal Rural do Semi-Árido
RN	Rio Grande do Norte
CSV	*Comma-Separated Values* (Valores Separados por Vírgula)
PDF	*Portable Document Format*
PNG	*Portable Network Graphics*

LISTA DE SÍMBOLOS

$G(s)$	Função de transferência da planta (processo).
$C(s)$, $G_c(s)$	Função de transferência do controlador.
$H(s)$	Função de transferência do elemento de realimentação (sensor).
$G(s)H(s)$	Função de transferência em malha aberta.
$R(s)$	Sinal de entrada (referência) no domínio de Laplace.
$Y(s)$	Sinal de saída no domínio de Laplace.
$e(t)$	Sinal de erro no domínio do tempo.
$u(t)$	Ação de controle (saída do controlador).
s	Variável complexa de Laplace ($s = \sigma + j\omega$).
p_i	Polos do sistema.
z_j	Zeros do sistema.
$\Delta(s)$	Equação característica do sistema (denominador em malha fechada).
ω_n	Frequência natural não amortecida.
ζ	Fator de amortecimento.
ω_d	Frequência natural amortecida ($\omega_n\sqrt{1-\zeta^2}$).
K	Ganho estático do sistema.
t_r	Tempo de subida (Rise Time).
t_p	Tempo de pico (Peak Time).
t_s	Tempo de acomodação (Settling Time).
$M_p(\%)$	Máximo sobressinal (percentual de overshoot).
e_{ss}	Erro em regime permanente.
$c(t)$	Resposta temporal do sistema.
$c(\infty)$	Valor final da resposta (regime permanente).
K_p	Ganho proporcional.
K_i	Ganho integral.
K_d	Ganho derivativo.
T_i	Constante de tempo integral.
T_d	Constante de tempo derivativa.
σ_a	Centroide das assíntotas.

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de controle são uma parte integrante da sociedade moderna. Controlar uma grandeza ou variável física significa alterar o seu valor de acordo com uma intenção. Nem sempre isso é realizável perfeitamente, seja porque não se dispõe de energia suficiente ou porque há perturbações muito importantes ou rápidas. Por exemplo, para controlar variáveis climáticas não há energia suficiente; já para controlar variáveis de processos industriais e de transporte, dependendo dos objetivos, excelentes resultados são possíveis (CASTRUCCI; BITTAR; SALLES, 2011).

Inúmeras aplicações estão à nossa volta. Os foguetes são acionados e o ônibus espacial decola para orbitar a Terra; envolta em jatos de água de resfriamento, uma peça metálica é usinada automaticamente; um veículo autônomo, distribuindo materiais para estações de trabalho em uma oficina de montagem aeroespacial, desliza ao longo do piso buscando seu destino (NISE, 2013, p. 28).

O estudo de sistemas de controle é, portanto, uma área essencial na engenharia. Disciplinas de análise, caracterização e projeto apresentam conceitos fundamentais, como a análise da resposta transitória, a estabilidade de sistemas dinâmicos e o projeto de controladores clássicos. No entanto, o domínio dessas técnicas exige compreensão sólida de conceitos teóricos frequentemente abstratos, como estabilidade e resposta transitória. A natureza fortemente matemática desses conteúdos representa um obstáculo frequente para estudantes, especialmente quando há pouca oportunidade de experimentação prática ou visualização intuitiva dos fenômenos envolvidos.

Além da complexidade teórica, o ensino desses conteúdos enfrenta desafios de ordem econômica e pedagógica. As ferramentas computacionais mais utilizadas em disciplinas de engenharia, como MATLAB e Simulink, são soluções proprietárias de alto custo, cujo acesso é frequentemente inviável em instituições públicas e para estudantes com recursos financeiros limitados. Mesmo quando disponíveis, essas plataformas muitas vezes não foram projetadas especificamente para fins educacionais, exigindo do docente um esforço adicional para adaptar funcionalidades genéricas a objetivos didáticos claros.

Diante desse desafio, este trabalho propõe o desenvolvimento de uma ferramenta computacional didática, implementada em *Python*, voltada ao ensino e à análise interativa de sistemas de controle lineares. A aplicação foi projetada com foco na acessibilidade, na usabilidade e na fidelidade teórica, utilizando a biblioteca *CustomTkinter* para construir uma interface gráfica moderna, responsiva e compatível com os principais sistemas operacionais, incluindo Windows e Linux. A escolha do *Python*, aliada a bibliotecas científicas consolidadas como python-control, NumPy, SciPy e Matplotlib, permitiu criar um ambiente integrado, de código aberto e livre de custos, capaz de substituir parcialmente soluções proprietárias em contextos educacionais.

A ferramenta é organizada em quatro módulos principais, cada um dedicado a uma

etapa fundamental do projeto de sistemas de controle. O primeiro módulo implementa o Critério de Routh-Hurwitz, oferecendo uma análise de estabilidade sem a necessidade de resolver explicitamente o polinômio característico. O segundo módulo realiza a caracterização completa de sistemas de segunda ordem, extraindo automaticamente parâmetros canônicos como a frequência natural não amortecida e o coeficiente de amortecimento, além de calcular métricas de desempenho transitório. O terceiro módulo é focado no Lugar Geométrico das Raízes (LGR), permitindo a construção gráfica da trajetória dos polos e o detalhamento analítico de propriedades como assíntotas e pontos de ruptura. Por fim, o quarto módulo possibilita a simulação interativa de controladores clássicos do tipo PI, PD e PID, com ajuste em tempo real dos ganhos e visualização simultânea da resposta temporal e do diagrama de polos e zeros.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver uma ferramenta computacional interativa, multiplataforma e de código aberto em *Python*, com interface gráfica construída por meio da biblioteca *CustomTkinter*, destinada à análise e caracterização de sistemas de controle lineares. A aplicação tem como foco o apoio didático e a democratização do ensino de engenharia de controle, integrando quatro módulos centrais: análise de estabilidade pelo Critério de Routh-Hurwitz, caracterização de sistemas de segunda ordem com cálculo de métricas de desempenho transitório, construção e análise do Lugar Geométrico das Raízes (LGR) e simulação interativa de controladores clássicos do tipo PI, PD e PID. A ferramenta oferece uma interface intuitiva, responsiva e visualmente moderna, que permite ao usuário explorar, visualizar e compreender de forma prática os conceitos fundamentais da teoria de controle.

1.1.2 Objetivos específicos

- Implementar um módulo para análise de estabilidade de sistemas lineares contínuos, incluindo o critério de Routh-Hurwitz e a identificação de polos e zeros;
- Desenvolver um módulo para análise de sistemas de segunda ordem, permitindo o cálculo de parâmetros característicos (ω_n, ζ, K) e a visualização das respostas ao degrau e à rampa unitária;
- Implementar um módulo dedicado ao estudo do Lugar Geométrico das Raízes, incluindo o traçado do LGR, a determinação de assíntotas, ângulos, pontos de ruptura e a representação gráfica de polos e zeros;
- Criar um módulo para projeto e simulação de controladores clássicos (PI, PD e PID), permitindo a análise do desempenho temporal e a comparação direta entre sistemas controlados e não controlados;

- Desenvolver uma interface gráfica do usuário moderna, responsiva e acessível, com suporte a temas claro, escuro e alto contraste, ajuste de fonte e atalhos de teclado;
- Garantir compatibilidade multiplataforma (Windows e Linux), com adaptação automática de escala de interface e ícones;
- Incluir um sistema de ajuda interativa e documentação integrada que oriente o usuário sobre o uso de cada módulo;

1.2 Organização e estrutura

Este trabalho organiza-se em cinco capítulos: o Capítulo **1** contextualiza o tema e define os objetivos; o Capítulo **2** detalha a metodologia, os fundamentos teóricos de controle e os recursos tecnológicos utilizados; o Capítulo **3** descreve a implementação, abordando a arquitetura, bibliotecas e interface da ferramenta; o Capítulo **4** apresenta os resultados e a validação frente aos objetivos propostos; e, por fim, o Capítulo **5** expõe as conclusões, contribuições e sugestões para trabalhos futuros.

2 METODOLOGIA

Este capítulo apresenta a fundamentação teórica e o processo de criação da ferramenta computacional desenvolvida em Python. A abordagem metodológica adotada seguiu um modelo iterativo e incremental, inspirado nas práticas de Desenvolvimento Ágil, com ênfase em prototipagem rápida, experiência do usuário e validação pedagógica contínua.

2.1 Sistemas de controle de segunda ordem

Os sistemas de segunda ordem desempenham um papel central na teoria de controle, pois permitem representar de forma precisa uma grande diversidade de fenômenos físicos, como sistemas massa-mola-amortecedor e circuitos RLC. Assim como as constantes de tempo caracterizam a dinâmica de sistemas de primeira ordem, a resposta transitória de sistemas de segunda ordem é descrita por meio de duas grandezas fundamentais: a frequência natural não amortecida (ω_n) e o coeficiente de amortecimento (ζ) (NISE, 2013, p. 252).

A função de transferência de um sistema de segunda ordem em malha fechada é dada pela Equação (2.1):

$$G(s) = \frac{Y(s)}{R(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.1}$$

Em que:

- ω_n é a frequência natural Não amortecida (velocidade da resposta).
- ζ é o coeficiente de amortecimento (forma da resposta).
- K é o ganho estático (valor final em regime permanente para entrada degrau unitário, se estável).

A classificação das respostas ao degrau conforme o coeficiente de amortecimento (ζ) é descrita a seguir:

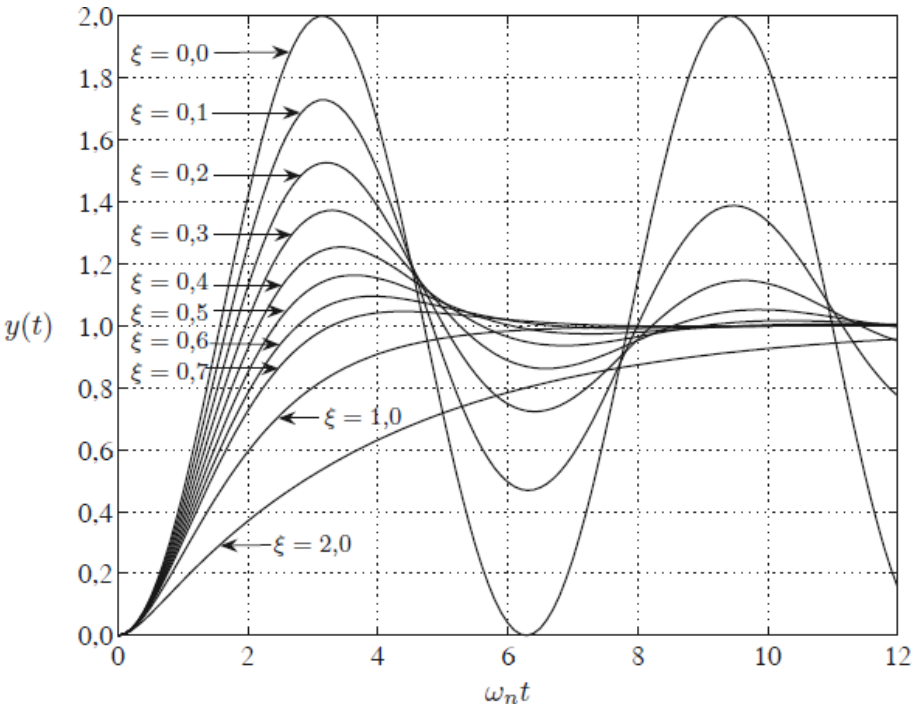
- Superamortecido ($\zeta > 1$): apresenta resposta lenta e sem oscilação. Os polos de malha fechada são reais e distintos.
- Criticamente amortecido ($\zeta = 1$): apresenta a resposta mais rápida possível sem oscilação. Os polos são reais e iguais.
- Subamortecido ($0 < \zeta < 1$): apresenta resposta rápida, porém com oscilação. Os polos são complexos conjugados e situam-se no semiplano esquerdo.

- Não amortecido ($\zeta = 0$): apresenta oscilação contínua e não decrescente. Os polos são imaginários puros.
- Instável ($\zeta < 0$): apresenta oscilação crescente ou divergente. Os polos localizam-se no semiplano direito.

As respostas de sistemas de controle são frequentemente avaliadas com entradas de teste padrão. A entrada em degrau unitário ($r(t) = 1$, para $t \geq 0$) é a mais comum, pois revela claramente o comportamento transitório e o regime permanente. Já a entrada em rampa unitária ($r(t) = t$) é utilizada para analisar a capacidade do sistema em acompanhar referências crescentes.

A Figura 1 apresenta as respostas ao degrau para os quatro regimes de amortecimento. O caso criticamente amortecido ($\zeta = 1$) é o mais rápido sem sobressinal, servindo como limite entre os sistemas superamortecidos ($\zeta > 1$, resposta lenta e não oscilatória) e subamortecidos ($0 < \zeta < 1$, resposta oscilatória mais rápida). O sistema não amortecido ($\zeta = 0$) oscila indefinidamente.

Figura 1 – Respostas ao degrau para os casos de amortecimento de sistemas de segunda ordem



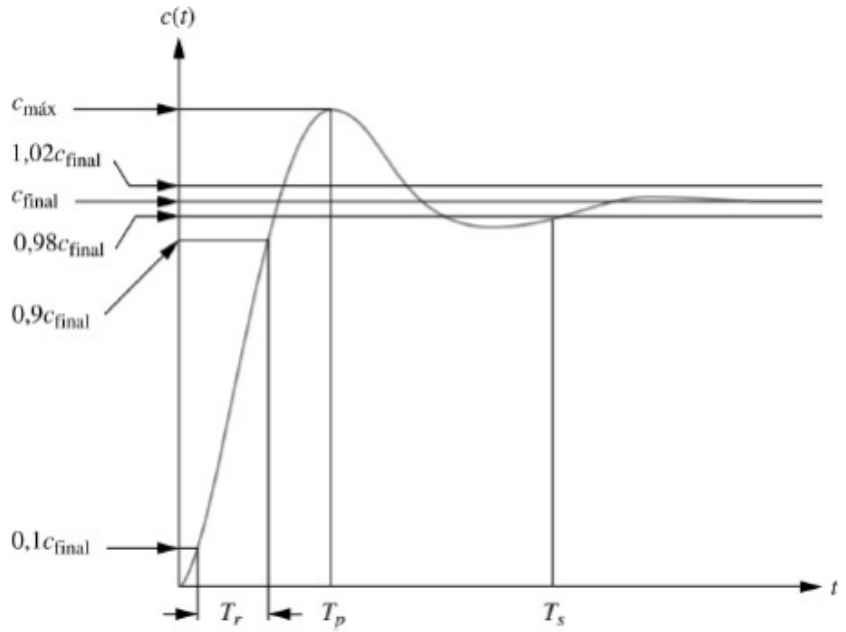
Fonte: (NISE, 2013).

2.2 Caracterização de respostas de sistemas de segunda ordem

Segundo Castrucci, em muitos casos, as características de desempenho desejadas em sistemas, especialmente em sistemas de controle com realimentação, podem ser sintetizadas por meio de um conjunto reduzido de indicadores extraídos diretamente da resposta temporal. Esses

indicadores são amplamente utilizados como especificações de projeto e, por definição, apenas possuem significado para sistemas estáveis. Na Figura 2 são ilustrados os principais parâmetros para caracterização de respostas de sistema de controle (CASTRUCCI; BITTAR; SALLES, 2011, p. 29).

Figura 2 – Especificações da resposta de sistemas de segunda ordem



Fonte: (NISE, 2013).

- t_d – Tempo de atraso (*delay time*): é o intervalo de tempo necessário para que a resposta alcance aproximadamente 50% do valor final em regime permanente. Pode ser estimado por:

$$t_d \approx \frac{0.7 * \zeta}{\omega_n} \tag{2.2}$$

- t_r – Tempo de subida (*rise time*): é o intervalo de tempo entre os instantes em que a resposta cresce de 10% a 90% do valor final (ou de 0% a 100%, conforme a convenção adotada). Esse parâmetro indica a rapidez inicial da resposta. Embora não exista uma relação analítica exata entre t_r e o fator de amortecimento ζ, uma aproximação prática para sistemas de segunda ordem é:

$$t_r \approx \frac{1.8}{\omega_n}, \quad \text{para } 0.4 < \zeta < 0.8 \tag{2.3}$$

- t_p – Tempo de pico (*peak time*): é o tempo necessário para que a resposta atinja o valor máximo de sobre-elevação. Pode ser calculado como:

$$t_p = \frac{\pi}{\omega_d} = \frac{\pi}{\omega_n \sqrt{1 - \zeta^2}} \tag{2.4}$$

- t_s – Tempo de acomodação (*settling time*): é o tempo necessário para que a resposta entre e permaneça dentro de uma faixa de tolerância em torno do valor final. Usualmente, considera-se 2% ou 5% como critério de tolerância:

$$t_{s(2\%)} \approx \frac{4}{\zeta\omega_n} \tag{2.5}$$

$$t_{s(5\%)} \approx \frac{3}{\zeta\omega_n} \tag{2.6}$$

- M_p – Máximo sobressinal (*maximum overshoot*): é a diferença percentual entre o valor máximo da resposta e o valor final em regime permanente, sendo calculado por:

$$M_p = \frac{c(t_p) - c(\infty)}{c(\infty)} \times 100\% \tag{2.7}$$

ou, de forma equivalente:

$$M_p = 100e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{2.8}$$

2.3 Análise de estabilidade pelo critério de Routh-Hurwitz

De acordo com (OGATA, 2010, p. 192), o critério de estabilidade de Routh-Hurwitz permite determinar a existência de raízes instáveis em uma equação polinomial sem a necessidade de resolvê-la explicitamente. Esse critério aplica-se a sistemas cuja equação característica pode ser expressa na forma de um polinômio de ordem finita n, conforme a Equação (2.9):

$$D(s) = a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0 = 0 \tag{2.9}$$

Quando utilizado na análise de sistemas de controle, o método fornece informações sobre a estabilidade absoluta diretamente a partir dos coeficientes a_n. A Tabela 1 apresenta a estrutura geral da Tabela de Routh-Hurwitz para um polinômio de quarta ordem, onde os coeficientes das linhas subsequentes são obtidos por meio de determinantes negativos das duas linhas anteriores.

Tabela 1 – Tabela de Routh-Hurwitz completa.

Potência de s	**Coluna 1**	**Coluna 2**	**Coluna 3**
s^4	a_4	a_2	a_0
s^3	a_3	a_1	0
s^2	$b_1 = \dfrac{-\begin{vmatrix} a_4 & a_2 \\ a_3 & a_1 \end{vmatrix}}{a_3}$	$b_2 = \dfrac{-\begin{vmatrix} a_4 & a_0 \\ a_3 & 0 \end{vmatrix}}{a_3}$	0
s^1	$c_1 = \dfrac{-\begin{vmatrix} a_3 & a_1 \\ b_1 & b_2 \end{vmatrix}}{b_1}$	0	0
s^0	$d_1 = \dfrac{-\begin{vmatrix} b_1 & b_2 \\ c_1 & 0 \end{vmatrix}}{c_1} = b_2$	0	0

Fonte: autoria própria (com base em (NISE, 2013, p.455)).

Segundo o critério, a condição necessária e suficiente para a estabilidade é que todos os coeficientes da primeira coluna da tabela tenham o mesmo sinal. O número de trocas de sinal corresponde diretamente ao número de raízes com parte real positiva (instáveis). Uma aplicação crítica deste método no projeto de controladores reside na determinação da faixa de um parâmetro ajustável, tipicamente o ganho K, necessária para assegurar a estabilidade. Ao manter K como uma variável na equação característica ($1 + KG(s)H(s) = 0$), os termos da primeira coluna tornam-se funções $f(K)$. A estabilidade é garantida resolvendo-se o sistema de desigualdades gerado por:

$$\text{Coluna } 1 > 0 \quad \Rightarrow \quad \{a_n > 0, \ldots, c_1(K) > 0, d_1(K) > 0\} \tag{2.10}$$

Adicionalmente, é possível determinar a condição de estabilidade marginal, que ocorre em um valor crítico (K_{crit}) onde uma linha inteira da tabela se anula. Neste cenário, define-se a *equação auxiliar* $A(s)$ utilizando os coeficientes da linha imediatamente superior à linha de zeros. Para uma linha de ordem s^2, a equação auxiliar assume a forma da Equação (2.11):

$$A(s) = \alpha s^2 + \beta = 0 \tag{2.11}$$

onde α e β são os elementos da linha s^2. A frequência de oscilação (ω_{osc}) no limiar da instabilidade é obtida substituindo $s = j\omega$ em (2.11), resultando em:

$$-\alpha\omega^2 + \beta = 0 \quad \Rightarrow \quad \omega_{osc} = \sqrt{\frac{\beta}{\alpha}} \quad \text{[rad/s]} \tag{2.12}$$

Dessa forma, o critério de Routh-Hurwitz transcende a verificação binária de estabilidade, fornecendo parâmetros quantitativos essenciais para a sintonia de controladores (NISE, 2013; DORF; BISHOP, 2010).

2.4 O método do lugar geométrico das raízes (LGR)

O método do lugar geométrico das raízes foi criado por W. R. Evans em 1948 e consiste em um gráfico, construído a partir do conhecimento dos polos e zeros do sistema em malha aberta, que permite visualizar de que forma os polos do sistema em malha fechada variam quando se altera o valor do ganho do sistema em malha aberta (CASTRUCCI; BITTAR; SALLES, 2011).

O LGR permite analisar como as raízes da equação característica de um sistema variam à medida que um parâmetro, tipicamente o ganho k, é alterado. Sua construção utiliza exclusivamente as informações do sistema em malha aberta, possibilitando prever o comportamento dinâmico em malha fechada a partir das posições relativas dos polos e zeros. O ponto de partida do LGR é a equação característica:

$$1 + kG(s) = 0, \tag{2.13}$$

em que $G(s) = \frac{N(s)}{D(s)}$. As raízes dessa equação determinam os polos do sistema realimentado, e o LGR representa graficamente todas as possíveis localizações desses polos no plano complexo à medida que k varia de 0 a ∞.

A construção do LGR baseia-se em propriedades geométricas e algébricas do sistema em malha aberta. Entre os principais elementos considerados destacam-se:

1. Número de ramos

O número de ramos do LGR é igual ao número total de polos de malha aberta. Os polos são marcados com “X” e os zeros com “O”.

2. Pontos de início e término

Cada ramo se inicia em um polo de malha aberta e termina em um zero. Caso haja mais polos do que zeros, os ramos restantes tendem ao infinito seguindo assíntotas definidas analiticamente.

3. Assíntotas

Os ramos associados aos polos em excesso seguem assíntotas que possuem centroide dado por:

$$\sigma = \frac{\sum \text{polos} - \sum \text{zeros}}{n - m}, \tag{2.14}$$

e ângulos de abertura determinados por:

$$\alpha_i = \frac{(2i+1)}{|n-m|}180^\circ, \qquad i = 0, 1, \ldots, |n-m|-1. \tag{2.15}$$

4. Pontos de saída e entrada

Os pontos em que raízes reais passam a formar pares complexos conjugados (saída) ou o processo inverso ocorre (entrada) são determinados pela condição:

$$\frac{dk}{ds} = 0, \tag{2.16}$$

sendo

$$k = -\frac{D(s)}{N(s)}. \tag{2.17}$$

5. Cruzamento com o eixo imaginário

Os valores de k para os quais o LGR cruza o eixo imaginário são obtidos resolvendo:

$$D(j\omega) + kN(j\omega) = 0, \tag{2.18}$$

ou alternativamente pelo critério de Routh-Hurwitz.

6. Ângulos de partida e chegada

Para polos e zeros complexos, os ângulos de partida e chegada dos ramos são determinados por:

$$\phi_{p_j} = \sum \phi_{z_i} - \sum_{i \neq j} \phi_{p_i} \pm 180^\circ, \tag{2.19}$$

$$\phi_{z_j} = \sum \phi_{p_i} - \sum_{i \neq j} \phi_{z_i} \pm 180^\circ. \tag{2.20}$$

Além das propriedades matemáticas, o LGR apresenta grande aplicabilidade prática no projeto de controladores clássicos, tais como PI, PD, avanço, atraso e PID, permitindo ajustar o comportamento dinâmico do sistema ao posicionar polos de malha fechada em regiões desejáveis do plano-s.

No âmbito deste trabalho, o LGR é utilizado como base para a ferramenta computacional desenvolvida, a qual realiza automaticamente a identificação de polos e zeros da planta, determina segmentos pertencentes ao LGR, calcula assíntotas, pontos de entrada e saída, ângulos de partida e chegada e, quando desejado, produz o gráfico completo do lugar das raízes. Toda a implementação foi realizada em Python.

2.5 Controladores clássicos (PID)

O controle de processos é fundamental para garantir a segurança e a produtividade de plantas industriais, sendo o controlador Proporcional-Integral-Derivativo (PID) a estratégia mais difundida devido à sua eficácia e simplicidade. Esse controlador atua corrigindo o erro entre a variável de processo e o valor de referência (setpoint) por meio de três ações distintas, Proporcional (P), Integral (I) e Derivativa (D), cujos parâmetros podem ser ajustados para modificar o comportamento do sistema (WOOLF, 2025). Graças à sua versatilidade e capacidade de operar satisfatoriamente mesmo sem um modelo matemático exato da planta, estima-se que mais de 90% dos laços de controle na indústria de processos utilizem algoritmos do tipo PID (ASTRöM; HäGGLUND, 1995; BEQUETTE, 2003, cap. 10).

2.5.1 Controlador Proporcional (P)

O controlador proporcional ajusta a saída do sistema em função direta do erro instantâneo. Sua função de transferência é dada por:

$$G_c(s) = K_p \tag{2.21}$$

e sua ação no domínio do tempo é expressa por:

$$u(t) = K_p\, e(t) \tag{2.22}$$

O termo proporcional reduz o tempo de subida e o erro em regime permanente, porém não o elimina completamente. Valores elevados de K_p aumentam o sobressinal e podem comprometer a estabilidade do sistema.

Aplicação típica: processos em que um pequeno erro em regime é aceitável e uma resposta rápida é desejável.

2.5.2 Controlador Proporcional–Integral (PI)

O controlador PI adiciona uma ação integral à proporcional, eliminando o erro em regime permanente (DORF; BISHOP, 2010). Sua função de transferência é:

$$G_c(s) = K_p + \frac{K_i}{s} = K_p\left(1 + \frac{1}{T_i s}\right), \quad T_i = \frac{K_p}{K_i} \tag{2.23}$$

A ação no domínio do tempo é dada por:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau \tag{2.24}$$

O termo integral acumula o erro ao longo do tempo, forçando o sistema a anular o erro de regime. Entretanto, a presença de um polo na origem pode reduzir a estabilidade e aumentar o

tempo de acomodação e o sobressinal.

Aplicação típica: controle de nível, vazão e temperatura em processos industriais.

2.5.3 Controlador Proporcional–Derivativo (PD)

O controlador proporcional–derivativo (PD) introduz uma ação derivativa que antecipa a tendência do erro, melhorando o amortecimento e a estabilidade do sistema (OGATA, 2010; NISE, 2013). Essa característica faz com que o controlador atue de forma preditiva, reduzindo o sobressinal e o tempo de pico, o que resulta em uma resposta mais suave e rápida.

Sua função de transferência é dada por:

$$G_c(s) = K_p + K_d s = K_p(1 + T_d s), \quad T_d = \frac{K_d}{K_p} \tag{2.25}$$

A ação no domínio do tempo pode ser expressa como:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} \tag{2.26}$$

O termo derivativo atua proporcionalmente à taxa de variação do erro, antecipando possíveis mudanças na variável controlada. Essa antecipação contribui para aumentar o amortecimento do sistema, reduzindo oscilações e melhorando a estabilidade transitória (BEQUETTE, 2003).

Aplicação típica: controle de movimento em robótica, sistemas mecatrônicos e suspensão ativa.

2.5.4 Controlador Proporcional–Integral–Derivativo (PID)

O controlador proporcional–integral–derivativo (PID) representa a forma mais completa e amplamente utilizada entre os controladores clássicos. Segundo Peter Woolf(WOOLF, 2025) e Nise (NISE, 2013), esse tipo de controle combina as vantagens das ações proporcional (P), integral (I) e derivativa (D), proporcionando ao sistema um equilíbrio entre rapidez, precisão e estabilidade.

O termo proporcional (P) fornece uma resposta imediata ao erro atual, reduzindo o tempo de subida; o termo integral (I) acumula o erro ao longo do tempo, eliminando o erro em regime permanente (ou offset); e o termo derivativo (D) antecipa o comportamento futuro do erro, reduzindo o sobressinal e melhorando a estabilidade do sistema.

A relação entre a saída do controlador e o erro é expressa matematicamente por 2.27:

$$c(t) = K_c \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{de(t)}{dt} \right] + C \tag{2.27}$$

onde:

- $c(t)$: saída do controlador;
- K_c: ganho proporcional do controlador;
- $e(t)$: erro entre o valor de referência e a variável de processo;
- T_i: tempo integral (constante de integração);
- T_d: constante de tempo derivativa;
- C: valor inicial do controlador.

No domínio de Laplace, a função de transferência do controlador PID é expressa por:

$$G_c(s) = K_c \left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.28}$$

Efeitos sobre o sistema:

- Termo proporcional (K_p): reduz o tempo de subida.
- Termo integral (K_i): elimina o erro de regime permanente.
- Termo derivativo (K_d): aumenta o amortecimento e reduz oscilações.

Aplicação típica: automação industrial, sistemas químicos, energia e manufatura. Na prática, o termo derivativo é implementado com filtros para reduzir a sensibilidade ao ruído.

2.6 Desenvolvimento e implementação da ferramenta

A ferramenta computacional foi desenvolvida em Python 3.8+, adotando uma arquitetura modular e orientada a objetos, que prioriza a manutenibilidade, a reutilização de código e a experiência do usuário. Seu principal propósito é oferecer um ambiente didático, interativo e de código aberto voltado ao estudo de sistemas lineares invariantes no tempo (SLIT), com ênfase na análise de estabilidade pelo Critério de Routh–Hurwitz, caracterização de sistemas de segunda ordem, lugar geométrico das raízes, análise e desempenho de controladores clássicos (PI, PD e PID).

O desenvolvimento da aplicação integrou um conjunto de bibliotecas de código aberto cuidadosamente selecionadas, levando em consideração critérios de estabilidade, qualidade da documentação e compatibilidade multiplataforma. As principais tecnologias empregadas são apresentadas a seguir:

- CustomTkinter: Responsável pela implementação da interface gráfica da aplicação.Permitindo à alternância entre temas claros e escuros, utiliza elementos estilizados como botões, sliders e caixas de entrada, e possui compatibilidade com telas de alta densidade (HiDPI).

Esses recursos tornam a interação mais intuitiva e visualmente agradável, contribuindo para o caráter didático da ferramenta (SCHIMANSKY, 2025; PYTHON SOFTWARE FOUNDATION, 2025).

- python-control: biblioteca especializada em teoria de controle, fornecendo funcionalidades essenciais para análise e projeto de sistemas de controle, incluindo manipulação de funções de transferência, simulação de respostas ao degrau e à rampa, cálculo de estabilidade, margens de ganho e fase, e geração automática do Lugar Geométrico das Raízes (LGR). Sua interface, similar à do MATLAB, facilita a migração de estudantes e profissionais para o ambiente Python sem perda de familiaridade (FULLER et al., 2021).

- NumPy e SciPy: bibliotecas amplamente utilizadas na computação numérica e álgebra linear. No contexto deste projeto, o módulo numpy.linalg foi empregado para o cálculo de raízes e autovalores, enquanto o scipy.integrate auxiliou na resolução de equações diferenciais e na geração de respostas temporais (HARRIS et al., 2020; VIRTANEN et al., 2020).

- Matplotlib: biblioteca completa para criar visualizações estáticas, animadas e interativas em Python. No projeto, essa biblioteca foi responsável pela geração das respostas temporais, diagramas de polos e zeros e pelo Lugar Geométrico das Raízes (LGR) em tempo real (HUNTER, 2007).

- ReportLab: empregada na geração automática de relatórios técnicos em formato PDF, reunindo informações sobre a planta analisada, parâmetros do controlador e gráficos de desempenho de forma consolidada e profissional (REPORTLAB Inc., 2025).

O código-fonte foi organizado em módulos independentes, permitindo a separação entre a lógica matemática, a interface gráfica e a camada de controle. Essa abordagem favorece a reutilização e facilita a manutenção do sistema. Os principais módulos e suas responsabilidades são descritos a seguir:

- tela.py: responsável pela gestão da interface gráfica principal da aplicação e pela navegação entre os três módulos funcionais, troca entre telas, alternância de temas e ajustes de fontes, renderização dinâmica de gráficos e tratamento centralizado de exceções.

- criterios_estabilidade.py: implementa o Critério de Routh–Hurwitz, recebendo os coeficientes do polinômio característico e construindo automaticamente a tabela completa de Routh, com validação numérica e tratamento de casos especiais, como linhas nulas ou presença de zeros na primeira coluna, permitindo determinar a estabilidade de um sistema linear (NISE, 2013).

- analise_segunda_ordem.py: responsável pela caracterização completa de sistemas lineares de segunda ordem, em malha aberta ou fechada, além de trabalhar com resposta do

tipo degrau unitário e rampa unitária. O módulo realiza a extração automática de parâmetros fundamentais, como a frequência natural (ω_n), o coeficiente de amortecimento (ζ) e o ganho estático (K), a partir dos coeficientes da função de transferência. Com base nesses parâmetros, o sistema calcula automaticamente as principais métricas de desempenho temporal, tempo de subida (T_r), tempo de pico (T_p), máximo sobressinal (M_p) e tempo de acomodação (T_s) e classifica o regime de amortecimento (subamortecido, criticamente amortecido ou superamortecido), conforme os critérios clássicos da teoria de controle (OGATA, 2010; DORF; BISHOP, 2010; NISE, 2013; CASTRUCCI; BITTAR; SALLES, 2011).

- lugar_geometrico_raizes.py: implementa as propriedades para construção do lugar geométrico das raízes com base na função de transferência em malha aberta $G(s)H(s)$. Calcula automaticamente as trajetórias dos polos para diferentes valores do ganho K, identificando pontos de ruptura, assíntotas e regiões de estabilidade, utilizando a biblioteca python-control (FULLER et al., 2021).

- controladores.py: implementa a simulação interativa de controladores clássicos do tipo PI, PD e PID, permitindo o ajuste dinâmico dos ganhos (K_p, K_i, K_d) em tempo real. O módulo gera respostas ao degrau comparativas entre os sistemas em malha aberta e malha fechada, além de exibir automaticamente o diagrama de polos e zeros e o Lugar das Raízes (LGR). Também realiza o cálculo das principais métricas de desempenho, oferecendo uma interface intuitiva e responsiva.

3 IMPLEMENTAÇÃO DA APLICAÇÃO

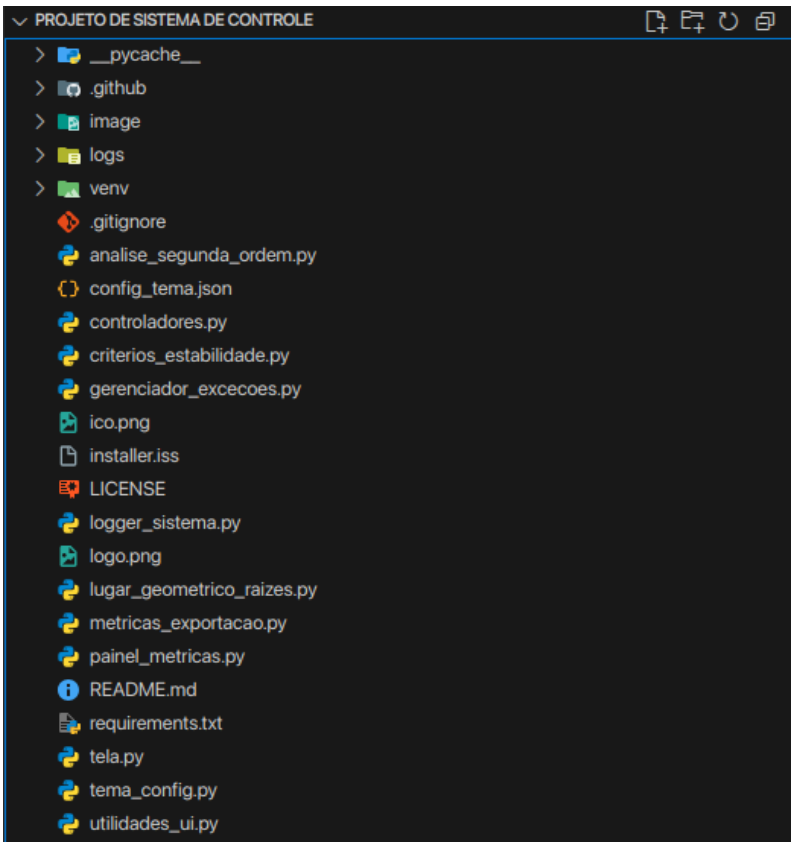
Este capítulo descreve a implementação da aplicação computacional desenvolvida neste trabalho, detalhando a arquitetura do *software*, as tecnologias empregadas, a estrutura modular do código e as funcionalidades implementadas. Todo o código-fonte desenvolvido, juntamente com a lista de requisitos e documentação técnica da ferramenta, encontra-se disponível publicamente em um repositório de versionamento. As informações completas para acesso ao código e instruções de instalação estão detalhadas no Apêndice A.

3.1 Arquitetura do software

A aplicação foi desenvolvida em Python 3.8+, empregando uma arquitetura modular e orientada a objetos inspirada no padrão *Model-View-Controller (MVC)* (REENSKAUG, 1979). Essa abordagem garante a separação de responsabilidades entre as camadas de interface, lógica matemática e controle de execução, facilitando manutenção e futuras expansões. A Figura 3 ilustra a organização geral do sistema, composta por três camadas principais:

- Modelo (Model): Contém os algoritmos matemáticos e métodos de análise, implementados em módulos como criterios_estabilidade.py, analise_segunda_ordem.py, lugar_geometrico_raizes.py e controladores.py.
- Visão (View): Corresponde à interface gráfica desenvolvida em CustomTkinter, responsável pela interação com o usuário e exibição dos resultados numéricos e gráficos.
- Controlador (Controller): Faz a comunicação entre interface e modelo, processando eventos de entrada, coordenando as simulações e atualizando a interface em tempo real.

Figura 3 – Arquitetura geral da aplicação desenvolvida



Fonte: Autoria própria (2025).

3.2 Arquitetura modular e lógica de cálculo

A aplicação foi estruturada seguindo um fluxo lógico que separa a camada de interface do usuário das rotinas de cálculo numérico. A Figura 4 apresenta o fluxograma de funcionamento do sistema, detalhando o caminho da informação desde a entrada dos parâmetros pelo usuário até a visualização dos resultados. O fluxo inicia-se no módulo principal (tela.py), que orquestra a navegação. Dependendo do módulo selecionado, os dados são validados e enviados para as classes de processamento específicas (CriteriosEstabilidade, AnalisadorSegundaOrdem, AnalisadorLGR ou JanelaControladores). Estas classes utilizam as bibliotecas *Control* e *NumPy* para executar os algoritmos de engenharia de controle, retornando os dados processados para renderização gráfica via *Matplotlib*.

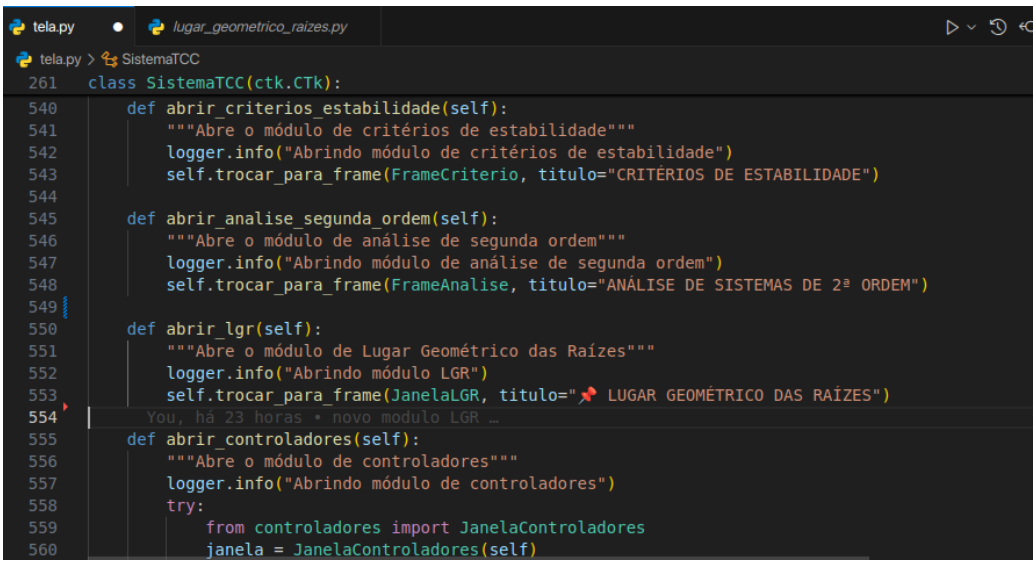
Figura 4 – Fluxograma funcional do sistema, destacando os quatro módulos de análise e o fluxo de dados

Fonte: Autoria própria (2025).

Abaixo, detalhamos as responsabilidades específicas de cada componente ilustrado no fluxo acima:

- Tela principal (tela.py): Atua como a camada de *View* principal, gerenciando a interface gráfica e a navegação. É responsável por instanciar os demais módulos conforme a seleção do usuário. A Figura 5 apresenta a estrutura central da interface.

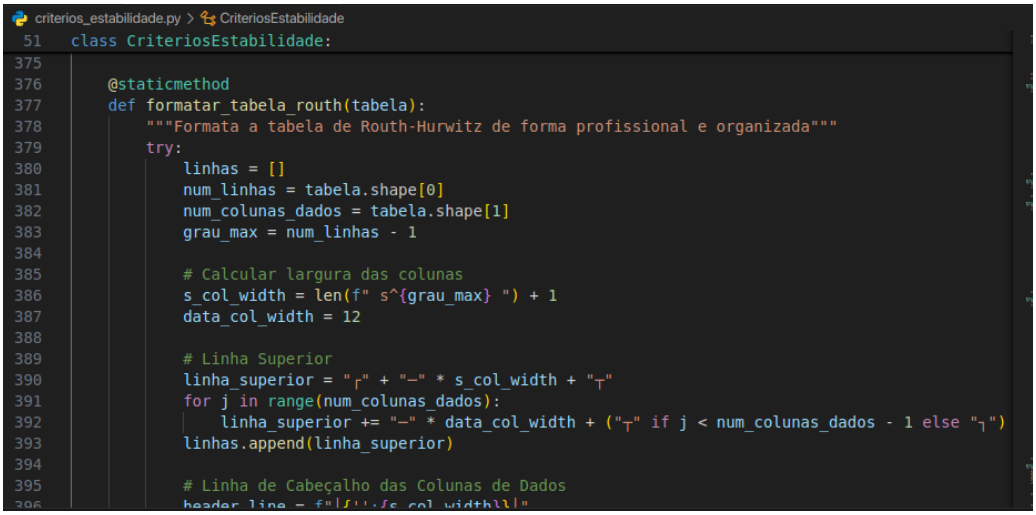
Figura 5 – Estrutura principal da interface gráfica desenvolvida em CustomTkinter



Fonte: Autoria própria (2025).

- Análise de estabilidade (criterios_estabilidade.py): Este módulo encapsula a lógica do Critério de Routh–Hurwitz. Ele recebe os coeficientes validados, constrói a tabela de Routh e contabiliza as trocas de sinal na primeira coluna para diagnosticar a estabilidade absoluta. A Figura 6 ilustra a interface dedicada a esta análise.

Figura 6 – Interface do módulo de análise de estabilidade utilizando o Critério de Routh–Hurwitz



Fonte: Autoria própria (2025).

- Sistema de segunda ordem (analise_segunda_ordem.py): Responsável pela matemática de sistemas de 2ª ordem. O módulo calcula ω_n, ζ e métricas temporais (T_r, T_p, M_p, T_s) baseando-se na função de transferência fornecida, classificando o sistema (subamortecido, crítico, etc.) automaticamente. A Figura 7 exibe os resultados gráficos e numéricos gerados.

Figura 7 – Interface do módulo de análise de sistemas de segunda ordem, com extração automática de parâmetros e métricas de desempenho

```python
class AnalisadorSegundaOrdem:
    def calcular_caracteristicas_temporais(self):
            resultado.append("   ⚠️ Sistema instável - características temporais não aplicáveis")
            return resultado

        if self.zeta == 0:
            resultado.append("   ⚠️ Sistema não amortecido - oscilação contínua")
            resultado.append(f"   Período de Oscilação (T): {2*math.pi/self.wn:.4f} s")
            resultado.append(f"   Frequência de Oscilação (f): {self.wn/(2*math.pi):.4f} Hz")
            return resultado

        if 0 < self.zeta < 1:
            wd = self.wn * math.sqrt(1 - self.zeta**2)
            beta = math.atan(math.sqrt(1 - self.zeta**2) / self.zeta)
            tr = (math.pi - beta) / wd
            resultado.append(f"   Tempo de Subida (Tr): {tr:.4f} s")
            resultado.append(f"      └─ Tempo para ir de 10% a 90% do valor final")
        elif self.zeta == 1:
            tr = 2.2 / self.wn
            resultado.append(f"   Tempo de Subida (Tr): {tr:.4f} s")
        else:
            tr = 2.2 / (self.zeta * self.wn)
            resultado.append(f"   Tempo de Subida (Tr): {tr:.4f} s (aproximado)")

        if self.zeta >= 1:
            tau = 1 / (self.zeta * self.wn)
            resultado.append(f"   Constante de Tempo (τ): {tau:.4f} s")

        return resultado
```

Fonte: Autoria própria (2025).

- Lugar geométrico das raízes (lugar_geometrico_raizes.py): Implementa o cálculo das trajetórias dos polos em malha fechada. Utiliza métodos simbólicos e numéricos para determinar assíntotas, pontos de ruptura e cruzamentos com o eixo imaginário, plotando o LGR interativo visto na Figura 8.

Figura 8 – Análise do lugar geométrico das raízes (LGR)

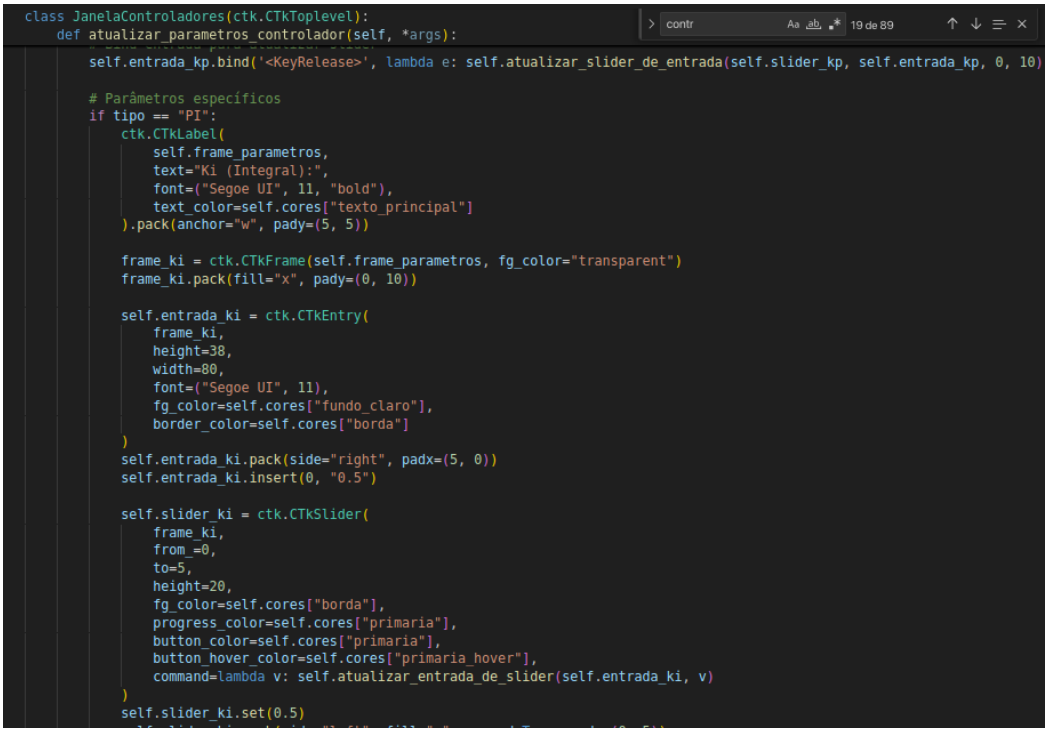
```python
class JanelaLGR(FrameBase):
    def plotar_lgr(self):

            # Configurações finais do gráfico
            ax.axhline(y=0, color='k', linewidth=0.8, alpha=0.5)
            ax.axvline(x=0, color='k', linewidth=0.8, alpha=0.5)
            ax.grid(True, alpha=0.3, linestyle=':', linewidth=0.5)
            ax.set_xlabel('Eixo Real', fontsize=12, fontweight='bold', color='black')
            ax.set_ylabel('Eixo Imaginário', fontsize=12, fontweight='bold', color='black')
            ax.set_title('Lugar Geométrico das Raízes (Root Locus)', fontsize=14, fontweight='bold', color='black', pad=

            # Configurar cores dos eixos e texto
            ax.tick_params(axis='both', colors='black', labelsize=10)
            for spine in ax.spines.values():
                spine.set_color('black')
                spine.set_linewidth(1.2)

            # Legenda com fundo branco
            legend = ax.legend(loc='best', fontsize=9, framealpha=0.9, facecolor='white', edgecolor='black')
            for text in legend.get_texts():
                text.set_color('black')

            # Ajustar layout
            plt.tight_layout()

            # Incorporar na interface
            self.canvas_grafico = FigureCanvasTkAgg(fig, master=self.grafico_container)
            self.canvas_grafico.draw()

            canvas_widget = self.canvas_grafico.get_tk_widget()
```

Fonte: Autoria própria (2025).

- Simulação de controladores (controladores.py): Módulo integrador que fecha a malha de controle. Ele combina a planta do usuário com ganhos PID ajustáveis, simulando a

resposta temporal e comparando o desempenho "sem controlador"versus "com controlador", conforme demonstrado na Figura 9.

Figura 9 – Simulação de controladores clássicos



```
class JanelaControladores(ctk.CTkToplevel):
    def atualizar_parametros_controlador(self, *args):
        self.entrada_kp.bind('<KeyRelease>', lambda e: self.atualizar_slider_de_entrada(self.slider_kp, self.entrada_kp, 0, 10)

        # Parâmetros específicos
        if tipo == "PI":
            ctk.CTkLabel(
                self.frame_parametros,
                text="Ki (Integral):",
                font=("Segoe UI", 11, "bold"),
                text_color=self.cores["texto_principal"]
            ).pack(anchor="w", pady=(5, 5))

            frame_ki = ctk.CTkFrame(self.frame_parametros, fg_color="transparent")
            frame_ki.pack(fill="x", pady=(0, 10))

            self.entrada_ki = ctk.CTkEntry(
                frame_ki,
                height=38,
                width=80,
                font=("Segoe UI", 11),
                fg_color=self.cores["fundo_claro"],
                border_color=self.cores["borda"]
            )
            self.entrada_ki.pack(side="right", padx=(5, 0))
            self.entrada_ki.insert(0, "0.5")

            self.slider_ki = ctk.CTkSlider(
                frame_ki,
                from_=0,
                to=5,
                height=20,
                fg_color=self.cores["borda"],
                progress_color=self.cores["primaria"],
                button_color=self.cores["primaria"],
                button_hover_color=self.cores["primaria_hover"],
                command=lambda v: self.atualizar_entrada_de_slider(self.entrada_ki, v)
            )
            self.slider_ki.set(0.5)
```

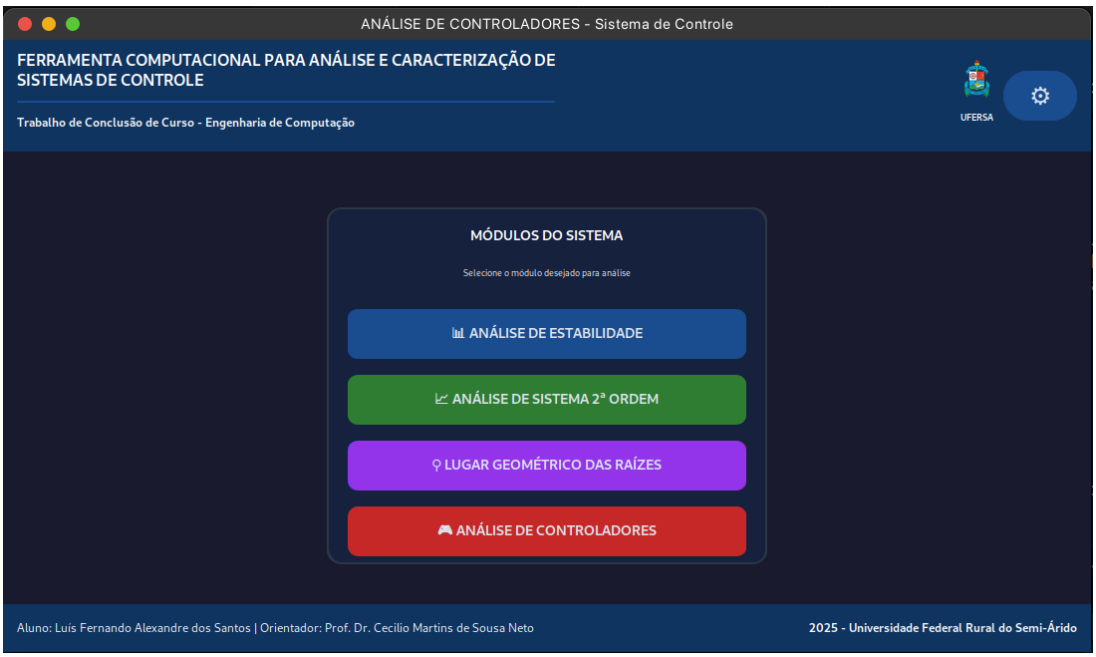
Fonte: Autoria própria (2025).

3.3 Interface gráfica e experiência do usuário

A interface foi projetada com foco em interatividade pedagógica, permitindo ao usuário visualizar instantaneamente os efeitos das alterações de parâmetros.

- Tela inicial: A Figura 10 apresenta a tela principal da aplicação, que funciona como ponto de acesso aos três módulos funcionais e ao botão de acessibilidade. Essa interface inicial organiza a navegação do usuário de forma simples e intuitiva.

Figura 10 – Tela principal da aplicação com acesso aos módulos



Fonte: Autoria própria (2025).

- Análise de estabilidade: Implementa o critério de Routh-Hurwitz para determinar a estabilidade absoluta do sistema. Por meio de um menu lateral para a inserção dos coeficientes do numerador e denominador, a ferramenta gera automaticamente a Tabela de Routh e contabiliza os polos no semiplano direito. A interface, ilustrada na Figura 11, dispõe de controles para executar a análise completa, verificar faixas de ganho K e limpar os dados.

Figura 11 – Módulo de análise de estabilidade com tabela de Routh-Hurwitz e diagnóstico automático

Fonte: Autoria própria (2025).

- Análise sistema de segunda ordem: Este módulo realiza automaticamente a caracterização de sistemas lineares de segunda ordem, extraindo a frequência natural não amortecida (ω_n), o coeficiente de amortecimento (ζ) e o ganho estático (K). Com base nesses valores, são calculadas as principais métricas transitórias: tempo de subida (T_r), tempo de pico (T_p), sobressinal máximo (M_p) e tempo de acomodação (T_s). A interface foi estruturada para fornecer uma experiência didática e interativa. Na Figura 12, à direita, observa-se o gráfico da resposta ao degrau ou à rampa unitária; já na Figura 13, à esquerda, encontra-se o painel de métricas calculadas e, ao lado, os campos para inserção dos coeficientes e execução das simulações.

Figura 12 – Módulo de análise de sistemas de segunda ordem: visualização da resposta ao degrau e métricas calculadas automaticamente

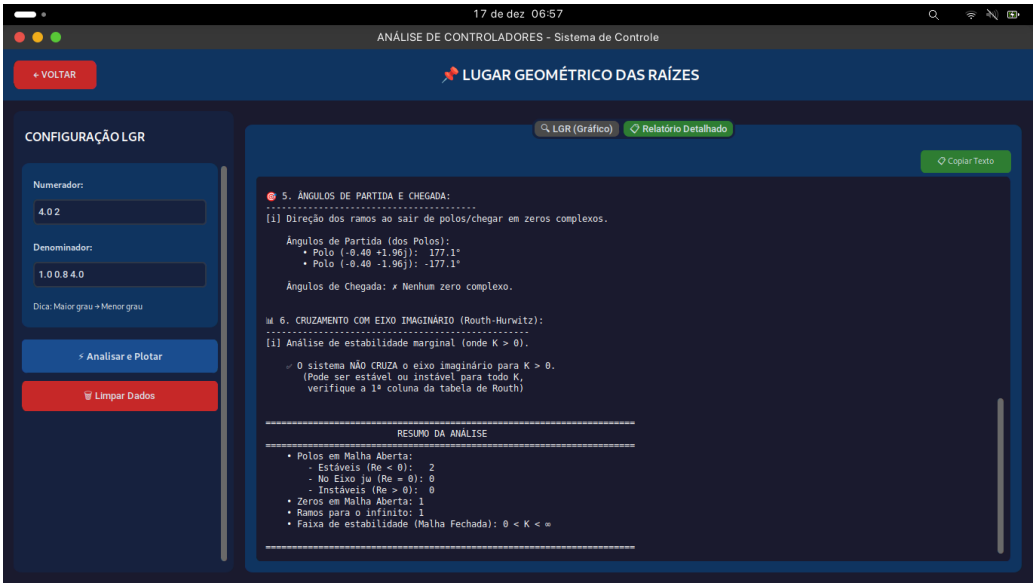
Fonte: Autoria própria (2025).

Figura 13 – Painel de relatório de análise para caracterização do sistema de segunda ordem

Fonte: Autoria própria (2025).

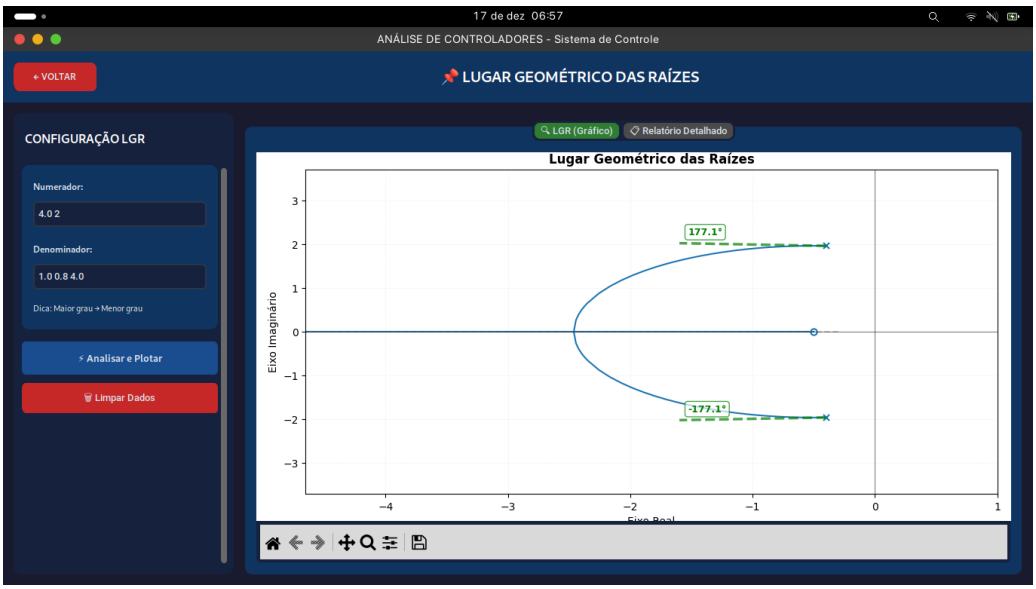
- Lugar geométrico das raízes: Permite analisar graficamente a movimentação dos polos do sistema em função do ganho K, recurso fundamental para o estudo da estabilidade, projeto de controladores e realimentação. Na Figura 14, à esquerda, encontram-se os campos para entrada da função de transferência em malha aberta $G(s)H(s)$, e à direita é exibido o relatório analítico com os dados calculados (polos, zeros, ângulos e pontos de ruptura). Já a Figura 15 apresenta o diagrama interativo gerado na aba "LGR (Gráfico)", destacando visualmente as trajetórias dos polos, as assíntotas e os limites de estabilidade no plano complexo.

Figura 14 – Interface do módulo LGR exibindo o relatório detalhado com os parâmetros analíticos do sistema



Fonte: Autoria própria (2025).

Figura 15 – Visualização gráfica do Lugar Geométrico das Raízes gerada pela ferramenta, evidenciando a trajetória dos polos

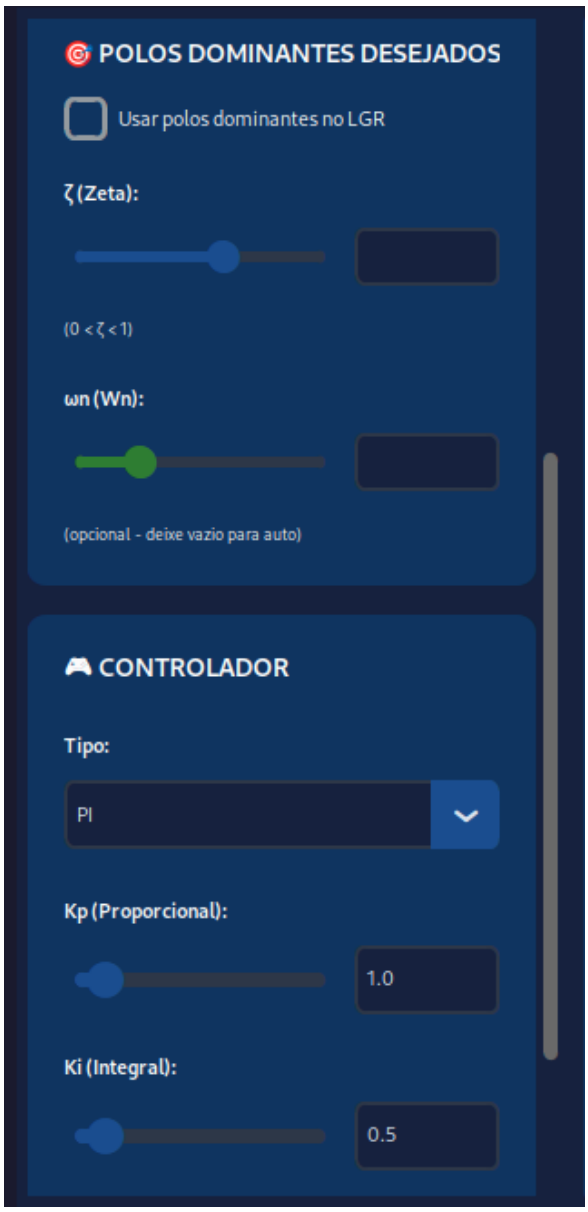


Fonte: Autoria própria (2025).

- Análise de controladores (PI, PD e PID): Implementa a simulação interativa dos controladores clássicos, permitindo ajustar dinamicamente os ganhos K_p, K_i e K_d e definir polos desejados para o projeto. A interface apresenta, lado a lado, a resposta temporal da planta original e do sistema controlado, facilitando a comparação de desempenho. O módulo também exibe o diagrama de polos e zeros, o Lugar Geométrico das Raízes e um painel completo de métricas. As Figuras 16a e 16b apresentam o painel principal de ajustes; a Figura 17 mostra a resposta temporal; e as Figuras 18, 19 e 20 exibem polos e zeros, LGR e métricas de desempenho, respectivamente.

Figura 16 – Painel de controle do módulo de controladores

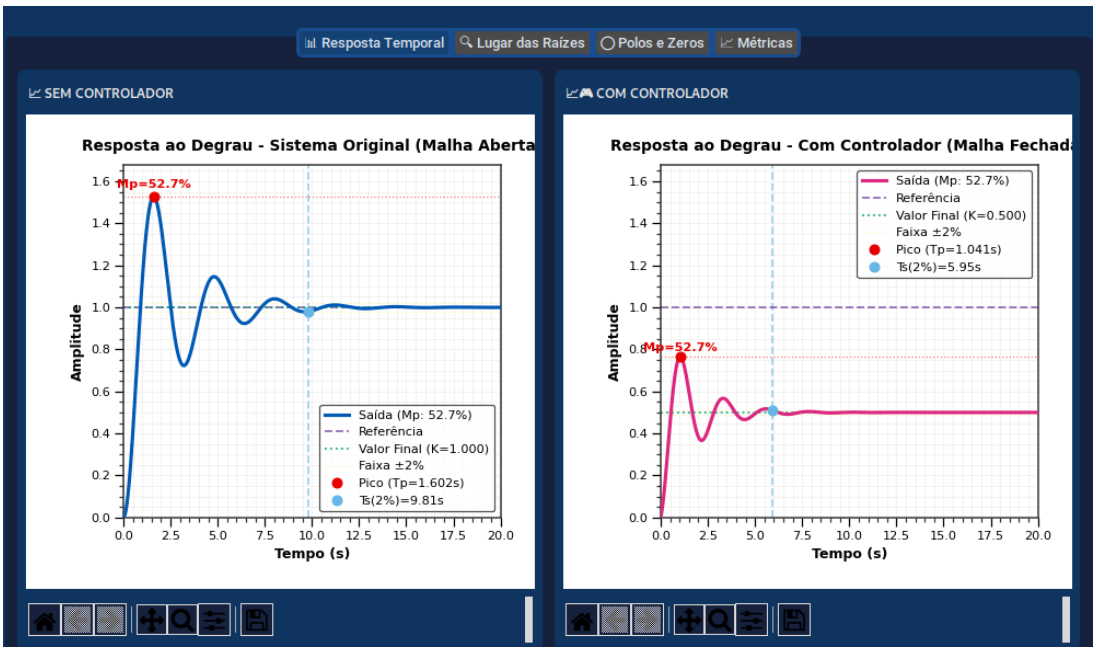
(a) Numerador, denominador e tipo de entrada



(b) Polos desejados e parte de controladores

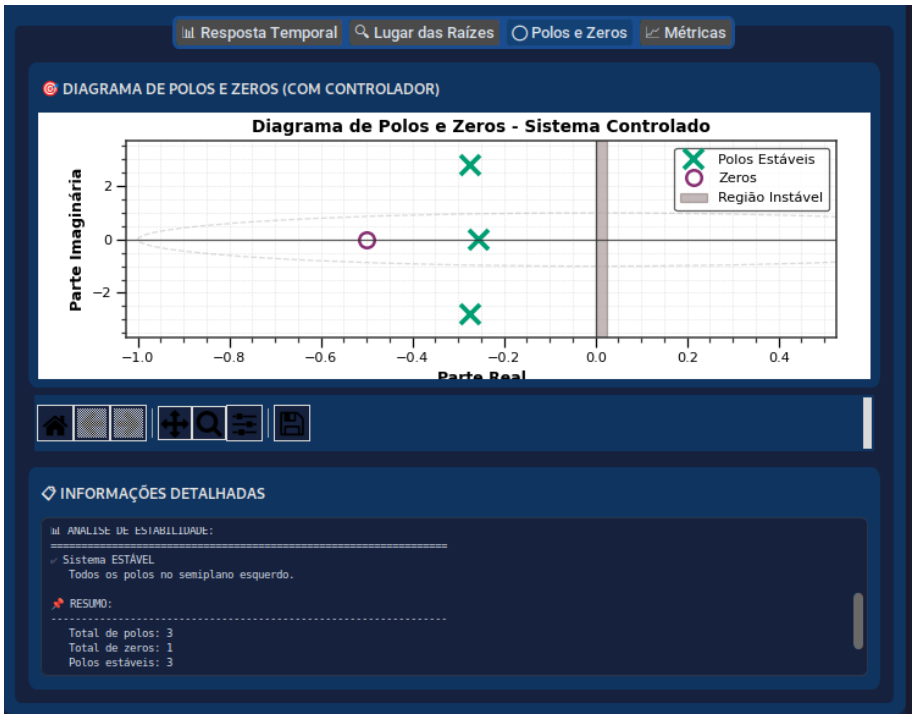
Fonte: Autoria própria (2025).

Figura 17 – Comparação das respostas temporais entre o sistema original e o sistema controlado



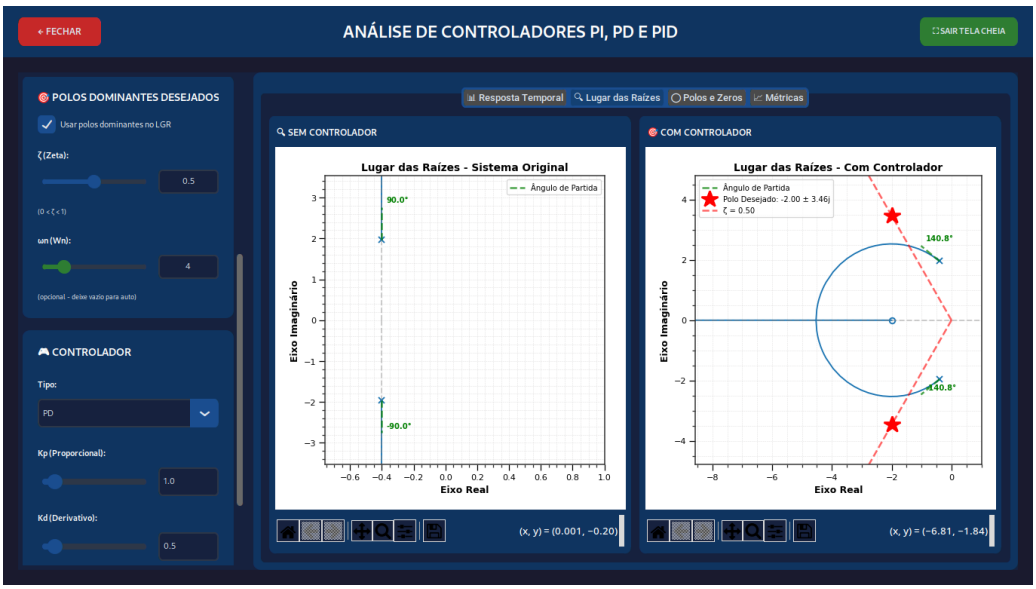
Fonte: Autoria própria (2025).

Figura 18 – Diagrama de polos e zeros com o novo posicionamento dos polos de malha fechada



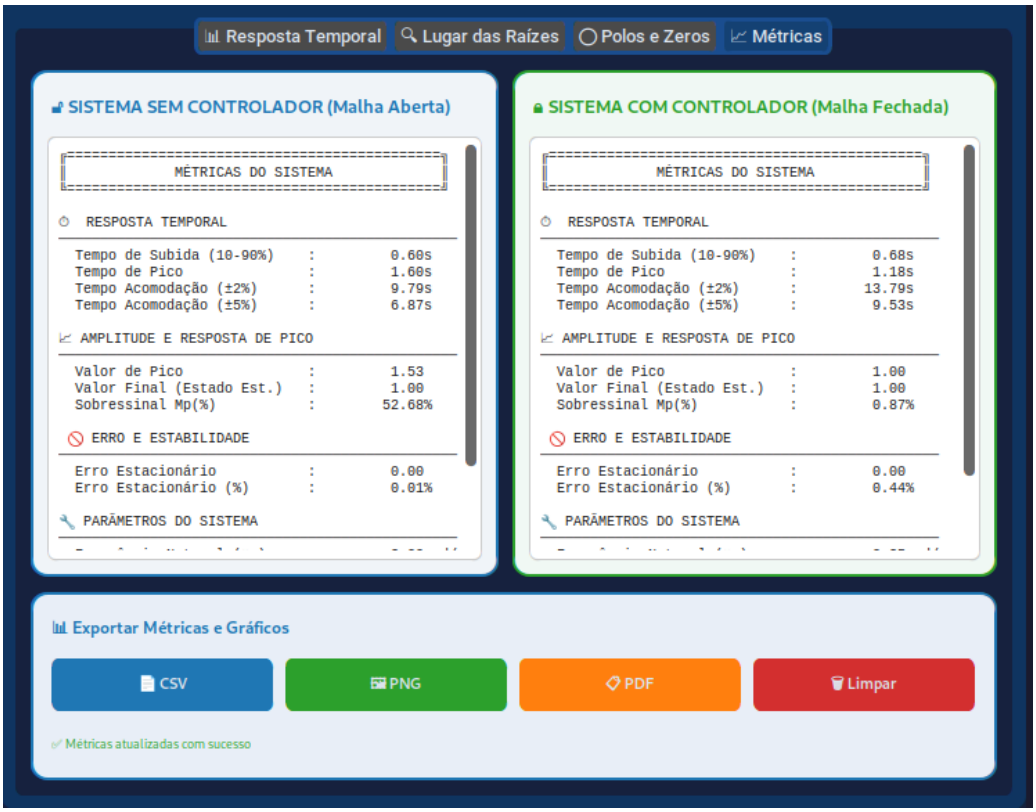
Fonte: Autoria própria (2025).

Figura 19 – LGR comparando o sistema original, o sistema controlado e os polos desejados



Fonte: Autoria própria (2025).

Figura 20 – Painel de métricas de desempenho: T_r, T_s, T_p, M_p e erro em regime permanente



Fonte: Autoria própria (2025).

4 RESULTADOS

Este capítulo apresenta a validação dos quatro módulos principais desenvolvidos na aplicação proposta. Cada validação foi elaborada para verificar de forma direta os objetivos específicos, que incluem a análise de estabilidade, a caracterização de sistemas de segunda ordem, o traçado do Lugar Geométrico das Raízes (LGR) e a simulação de controladores PID.

4.1 Módulo de análise de estabilidade (critério de Routh–Hurwitz)

O módulo de estabilidade tem como objetivo verificar se um sistema linear contínuo apresenta polos no semiplano direito, conforme o Critério de Routh-Hurwitz estudado no Capítulo **2**. Esse critério é amplamente utilizado por dispensar o cálculo explícito das raízes, permitindo determinar a estabilidade do sistema observando exclusivamente os sinais da primeira coluna da tabela de Routh. Para validar a implementação deste módulo, foram selecionados dois exemplos clássicos da literatura: um caso estável extraído de (CASTRUCCI; BITTAR; SALLES, 2011) e um caso instável baseado em (NISE, 2013).

- Caso 1: Sistema estável

  O primeiro caso de validação utiliza o polinômio do Exemplo 3.13 (item c) de (CASTRUCCI; BITTAR; SALLES, 2011, p. 76), que representa um sistema onde todos os polos encontram-se no semiplano esquerdo. A equação característica é dada por:

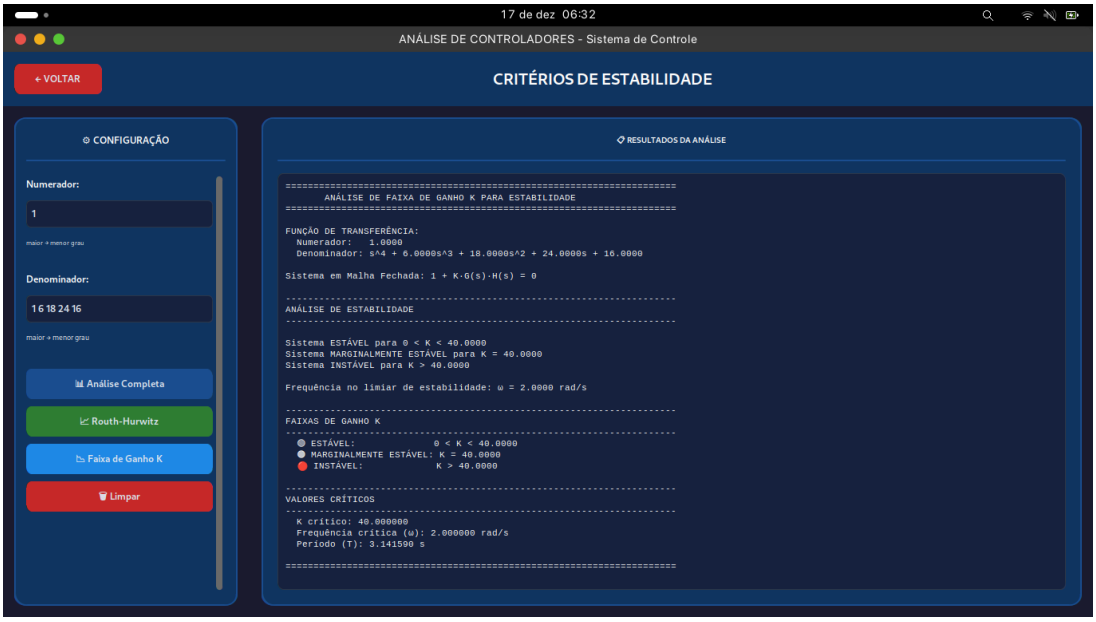
$$D(s) = s^4 + 6s^3 + 18s^2 + 24s + 16 \tag{4.1}$$

  A partir destas entradas de dados, a análise é processada pela atividade de Routh-Hurwitz", ativada mediante o acionamento da ferramenta. A Figura 21 expõe a tabela gerada a partir deste processo, em que a montagem se comporta de acordo com a teoria sobre o ensaio e, quando a primeira coluna é avaliada pelo arranjo dos valores 1, 6, 14, 24 e 16, não há mudanças de sinal, resultando em uma declaração correta sobre o sistema, que é, de fato, estável. A análise é estendida pela funcionalidade Faixa de Ganho K", conforme indicado na Figura 22, onde, de acordo com o cálculo simbólico, o software determina que as condições sobre a estabilidade seguem sendo cumpridas para o intervalo $0 < K < 40$, encontrando o ponto crítico de instabilidade no ganho crítico $K_{crit} = 40$, com uma frequência de oscilação associada de $\omega = 2.0$ rad/s.

Figura 21 – Análise do polinômio estável $D(s) = s^4 + 6s^3 + 18s^2 + 24s + 16$ (Exemplo 3.13-c de Castrucci). Saída obtida pela ferramenta

Fonte: Autoria própria (2025).

Figura 22 – Análise da faixa de ganho K e determinação da frequência crítica para o mesmo sistema



Fonte: Autoria própria (2025).

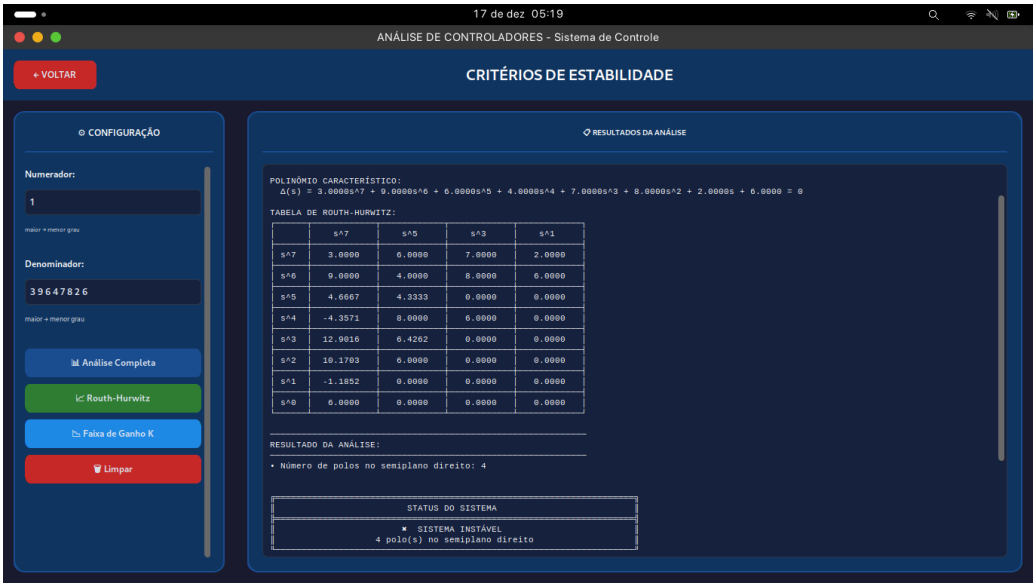
- Caso 2: Sistema instável

  O segundo caso avalia um sistema de ordem elevada instável, baseado no Exercício 6.1 de (NISE, 2013, p. 457). A equação característica analisada é um polinômio de grau 7:

$$P(s) = 3s^7 + 9s^6 + 6s^5 + 4s^4 + 7s^3 + 8s^2 + 2s + 6 \tag{4.2}$$

De acordo com a Figura 23, apresenta a tabela gerada, cujas mudanças de sinal na primeira coluna evidenciam a presença de quatro raízes no semiplano direito, caracterizando o sistema como instável.

Figura 23 – Análise do polinômio de sétima ordem do Exercício 6.1 de Nise. Resultados obtidos pela ferramenta



Fonte: Autoria própria (2025).

4.2 Módulo de análise de sistemas de segunda ordem

O módulo de análise de sistemas de segunda ordem tem como finalidade identificar automaticamente os parâmetros canônicos do sistema e calcular as principais características de desempenho transitório. Para validar sua implementação, utilizou-se o Exemplo 4.5 de (NISE, 2013, p. 267), que propõe a determinação analítica das especificações de desempenho a partir da função de transferência. Função de transferência e parâmetros canônicos, o sistema utilizado para validação é descrito pela equação:

$$G(s) = \frac{100}{s^2 + 15s + 100} \tag{4.3}$$

A comparação direta com o modelo padrão de segunda ordem,

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{4.4}$$

permite a identificação imediata dos parâmetros. Igualando os termos $\omega_n^2 = 100$ e $2\zeta\omega_n = 15$, obtêm-se:

$$\omega_n = 10\,\text{rad/s}, \qquad \zeta = 0,75.$$

Métricas teóricas e validação, aplicando as expressões analíticas para sistemas suba-

mortecidos, obtêm-se os seguintes valores exatos para as métricas de desempenho, conforme apresentado por Nise:

$$T_p = 0,475\,\text{s}, \qquad \%UP = 2,838\%, \qquad T_s = 0,533\,\text{s}.$$

Adicionalmente, o tempo de subida é calculado como $T_r = 0,23\,\text{s}$. Para consolidar a análise, a Tabela 2 organiza as grandezas teóricas utilizadas neste processo de validação.

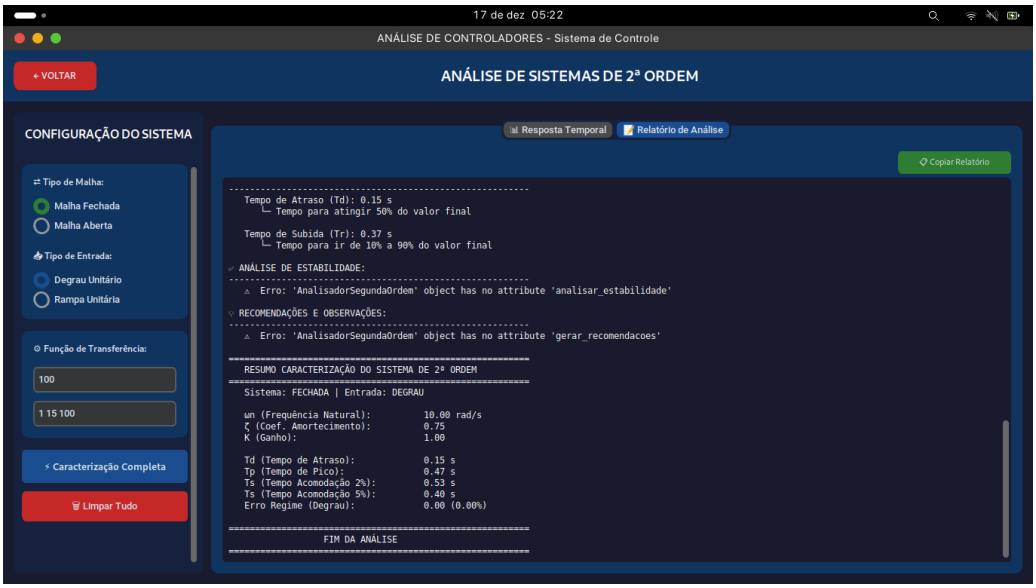
Tabela 2 – Parâmetros e métricas de referência extraídos do Exemplo 4.5 de Nise (2013).

Grandeza	**Símbolo**	**Valor Teórico (Nise)**
Frequência natural	ω_n	$10\,\text{rad/s}$
Coeficiente de amortecimento	ζ	$0,75$
Tempo de pico	T_p	$0,475\,\text{s}$
Sobressinal máximo	$\%UP$	$2,838\%$
Tempo de acomodação	T_s	$0,533\,\text{s}$
Tempo de subida	T_r	$0,23\,\text{s}$

Fonte: Adaptado de (NISE, 2013).

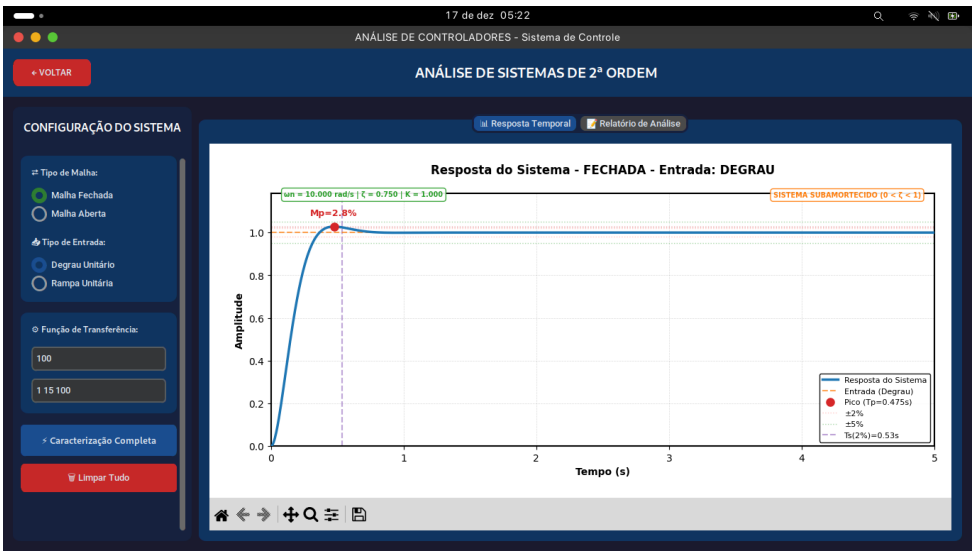
Na Figura 24 apresenta o painel de análise ilustrada pela ferramenta ao processar a função de transferência do exemplo. Observa-se a correta identificação dos parâmetros $\omega_n = 10,0000\,\text{rad/s}$ e $\zeta = 0,7500$. As características temporais foram calculadas com alta precisão decimal, indicando um sobressinal de $2,84\%$ e tempo de acomodação de $0,5333\,\text{s}$ (critério de 2%), valores extremamente próximos à referência teórica. Já a Figura 25 ilustra a resposta temporal ao degrau, confirmando visualmente o comportamento subamortecido e validando a fidelidade da simulação gráfica e numérica do módulo.

Figura 24 – Relatório de análise gerado pela ferramenta para o sistema do Exemplo 4.5



Fonte: Autoria própria (2025).

Figura 25 – Gráfico da resposta ao degrau gerado pela ferramenta, evidenciando as características temporais calculadas



Fonte: Autoria própria (2025).

4.3 Módulo do lugar geométrico das raízes (LGR)

O módulo de Lugar Geométrico das Raízes foi validado considerando dois casos representativos da literatura: o primeiro aborda um sistema com polos reais distintos, baseado no Exemplo 4.2 de (CASTRUCCI; BITTAR; SALLES, 2011, p. 15), e o segundo analisa um sistema com polos complexos, utilizando o Exemplo 8.6 de (NISE, 2013, p. 607). Além de realizar a construção gráfica do LGR, a ferramenta desenvolvida apresenta um relatório contendo os cálculos analíticos das propriedades fundamentais para o traçado.

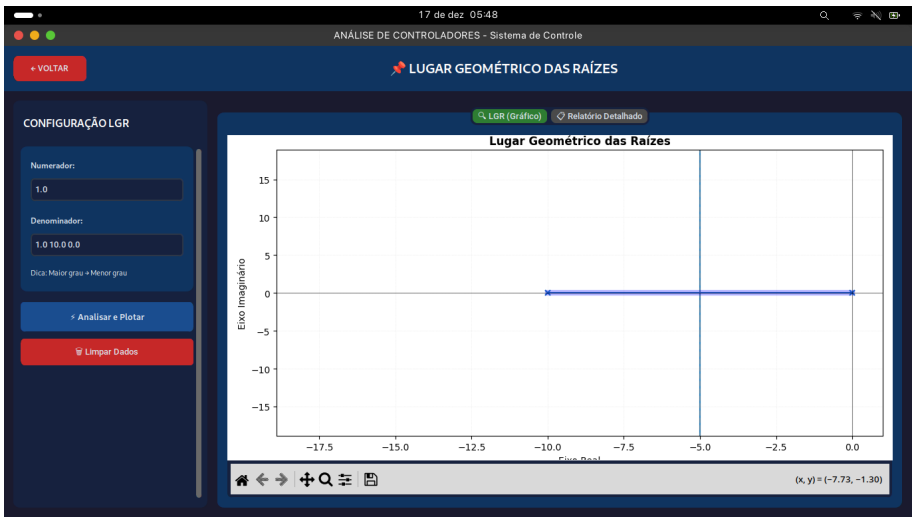
- Caso 1: Sistema com polos reais

  Para a validação de sistemas com polos reais, utilizou-se o Exemplo 4.2 de (CASTRUCCI; BITTAR; SALLES, 2011). A função de transferência de malha aberta considerada é:

$$G(s)H(s) = \frac{k}{s(s+10)}. \tag{4.5}$$

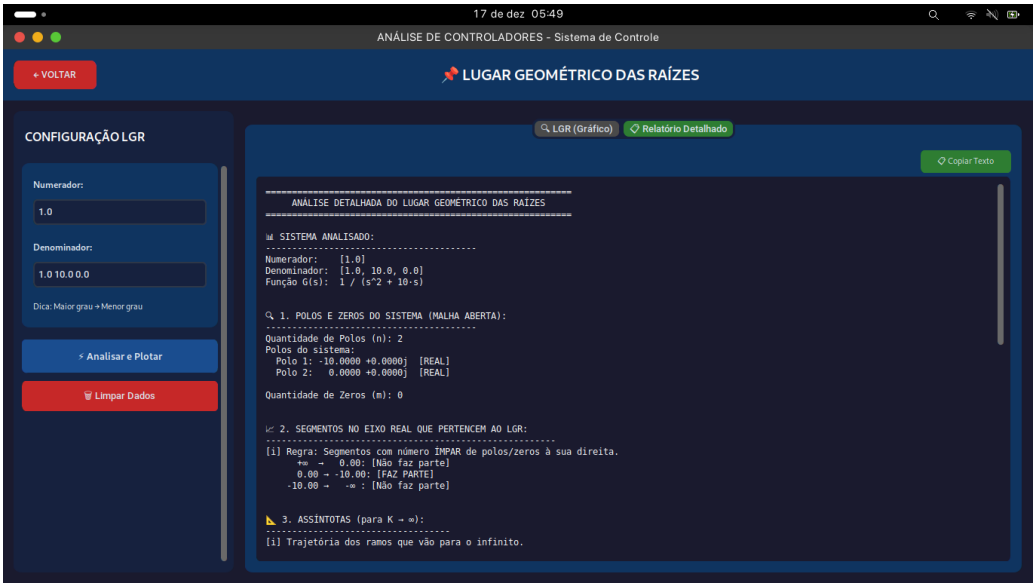
  De acordo com o exemplo, o sistema possui dois polos reais em malha aberta localizados na origem ($s_1 = 0$) e em $s_2 = -10$. O Lugar das Raízes deve apresentar um segmento sobre o eixo real entre os dois polos e duas assíntotas verticais partindo de um ponto de ruptura em $\sigma_a = -5$, com ângulos de 90° e 270°. Na Figura 26, apresenta-se a interface do módulo de LGR com a inserção $Num = [1]$ e $Den = [1, 10, 0]$. Ao selecionar a opção de plotagem, a ferramenta gerou o gráfico que reproduz fielmente o comportamento esperado: os ramos partem dos polos reais, encontram-se em $s = -5$ e divergem paralelamente ao eixo imaginário, confirmando a precisão do algoritmo de traçado. Nas Figuras 27 e 28 são apresentados os passos para esboçar o LGR.

Figura 26 – Interface do módulo de LGR analisando a planta $G(s) = 1/[s(s+10)]$. O gráfico exibe os polos em 0 e -10 e o ponto de ruptura das ramificações em -5, conforme Exemplo 4.2 de Castrucci



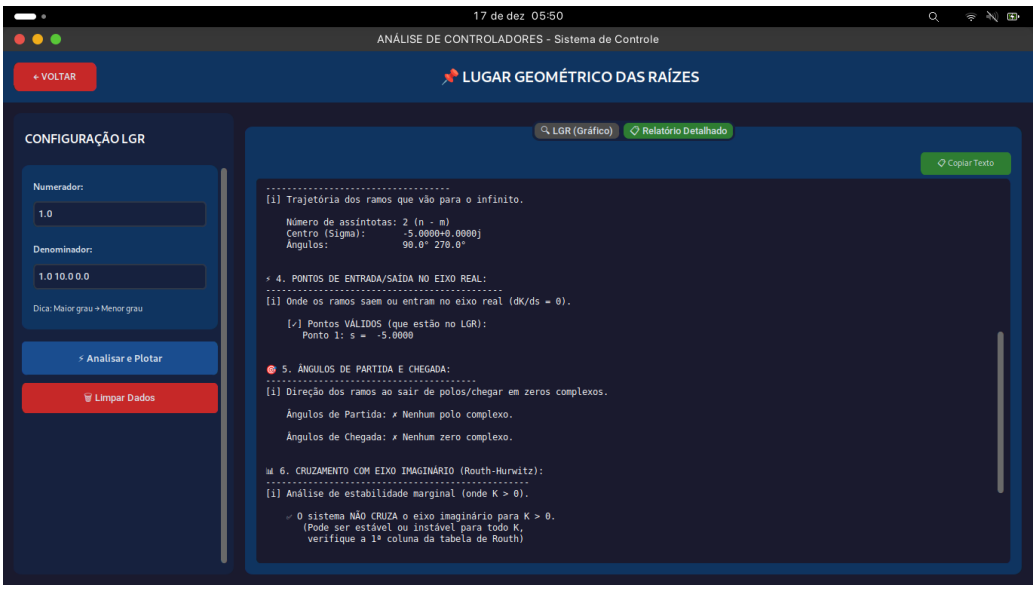
Fonte: Autoria própria (2025).

Figura 27 – Painel de resultados da análise para $G(s) = 1/[s(s+10)]$, detalhando polos e propriedades das assíntotas



Fonte: Autoria própria (2025).

Figura 28 – Continuação do relatório analítico, evidenciando o ponto de ruptura em -5 e a estabilidade do sistema para $k > 0$



Fonte: Autoria própria (2025).

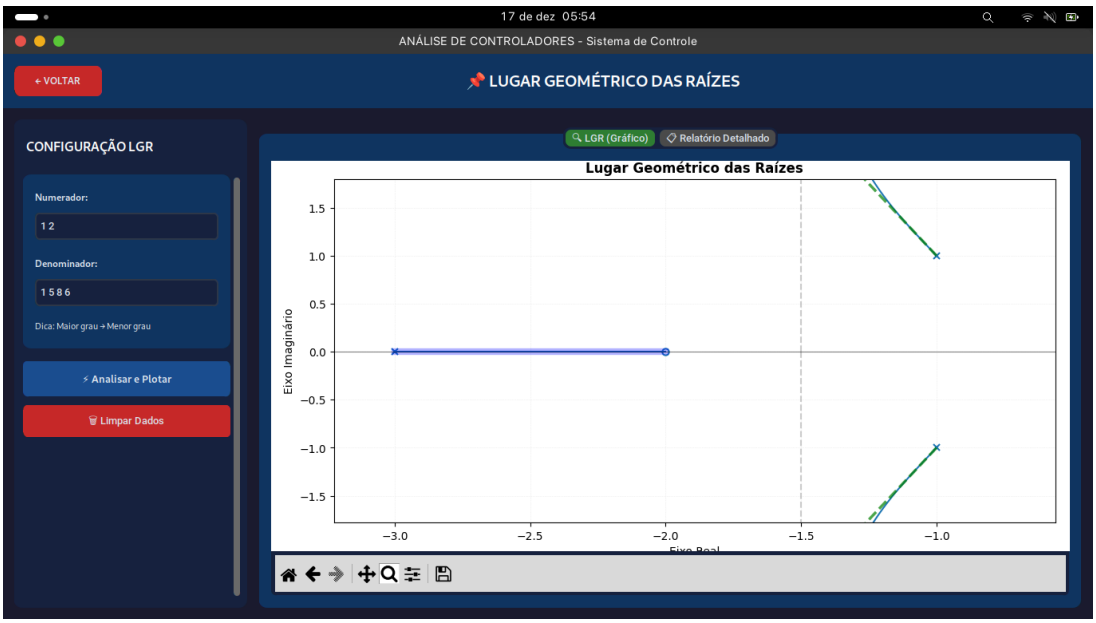
- Caso 2: Sistema com polos complexos

A segunda validação avaliou a ferramenta utilizando uma planta com polos complexos conjugados, baseada no Exemplo 8.6 de (NISE, 2013):

$$G(s) = \frac{K(s+2)}{(s+3)(s^2+2s+2)}. \tag{4.6}$$

O sistema possui um zero em $s = -2$, um polo real em $s = -3$ e um par de polos complexos em $s = -1 \pm j1$. A Figura 29 apresenta a interface da aplicação durante essa análise, para a qual foram fornecidos os coeficientes $[1, 2]$ no numerador e $[1, 5, 8, 6]$ no denominador. O gráfico gerado exibe corretamente a topologia do LGR, destacando o ângulo de partida dos polos complexos de aproximadamente $108,4°$ e a chegada dos ramos ao zero finito e aos zeros no infinito, validando a precisão angular do algoritmo.

Figura 29 – Análise do LGR para o sistema do Exemplo 8.6 de Nise. O gráfico evidencia o ângulo de partida dos polos complexos ($\theta \approx 108,4°$) e a influência do zero em -2



Fonte: Autoria própria (2025).

4.4 Módulo análise de controladores

O módulo de simulação de controladores foi validado utilizando três casos práticos extraídos de livros-texto de controle automático, permitindo comparar os resultados gerados pela ferramenta com soluções teóricas consolidadas. Os testes em PI, PD e PID, foram realizados sobre plantas de segunda ordem, e os parâmetros dos controladores foram ajustados conforme as especificações de desempenho apresentada (CASTRUCCI; BITTAR; SALLES, 2011).

- Caso 1: Controlador Proporcional-Integral (PI)
  Para validar o comportamento do controlador PI, foi utilizada a planta

$$G(s) = \frac{0.1}{(s+0.2)(s+0.644)}. \tag{4.7}$$

  O objetivo é projetar um compensador integral capaz de eliminar esse erro mantendo a resposta transitória próxima ao comportamento determinado pelos polos dominantes originais. Neste caso, os polos desejados foram determinados com um coeficiente de amortecimento $\zeta = 0.456$ e frequência natural $\omega_n \approx 0.706$ rad/s, resultando em

$$s_{1,2} \approx -0,322 \pm j\,0,3.$$

  A função de transferência do controlador PI é dada por:

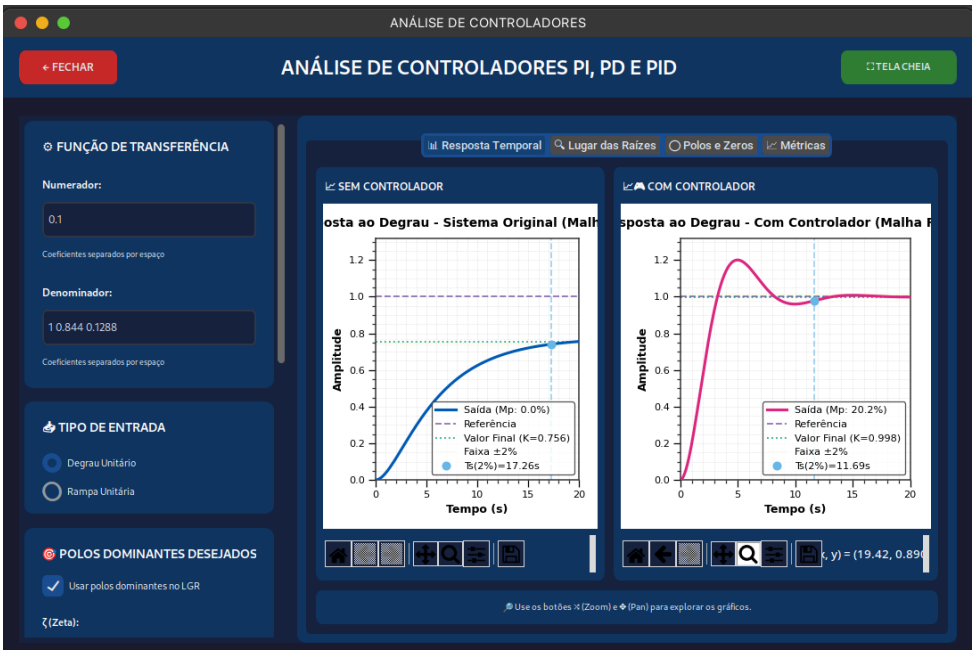
$$C(s) = K_p + \frac{K_i}{s} = \frac{K_p s + K_i}{s} \tag{4.8}$$

Neste caso, os ganhos do controlador PI obtidos (CASTRUCCI; BITTAR; SALLES, 2011, p. 44)

$$K_p = 5, \qquad K_i = 1.$$

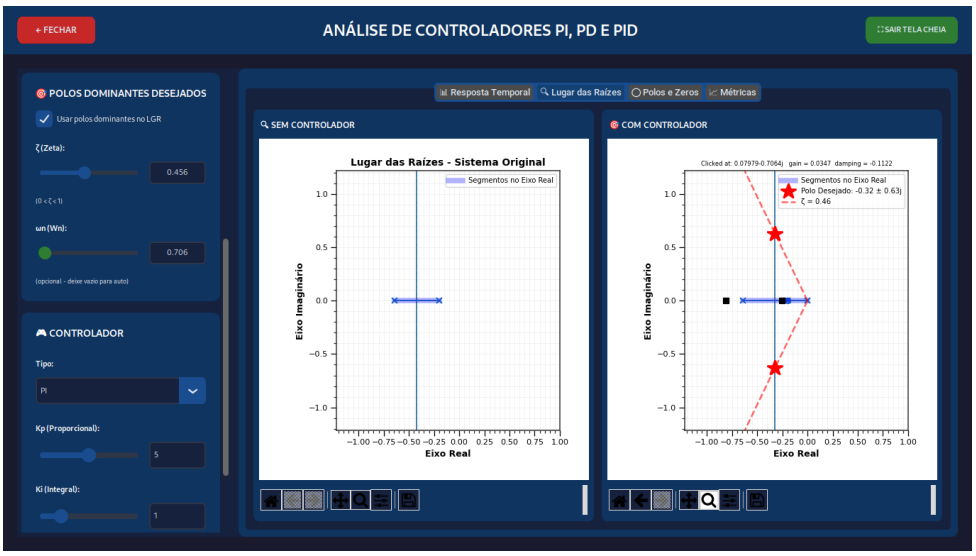
Na Figura 33 são apresentadas as respostas do sistema sem controlador e com controlador para uma entrada do tipo degrau unitário. Baseado na Figura 33, observa-se que, sem controlador, a resposta do sistema exibe a dinâmica da planta pura ($K = 1$), apresentando um comportamento lento. Com a inserção do controlador PD, a resposta do sistema atinge os objetivos de projeto, resultando em um tempo de acomodação de $1,16$ segundos e sobressinal de $13,10\%$, conforme ilustrado na Figura 34. A aplicação do controlador PD reduziu o tempo de acomodação.

Figura 30 – Resposta ao degrau do sistema original e com controlador PI. A figura destaca o erro permanente na malha aberta e sua eliminação após a aplicação do controlador



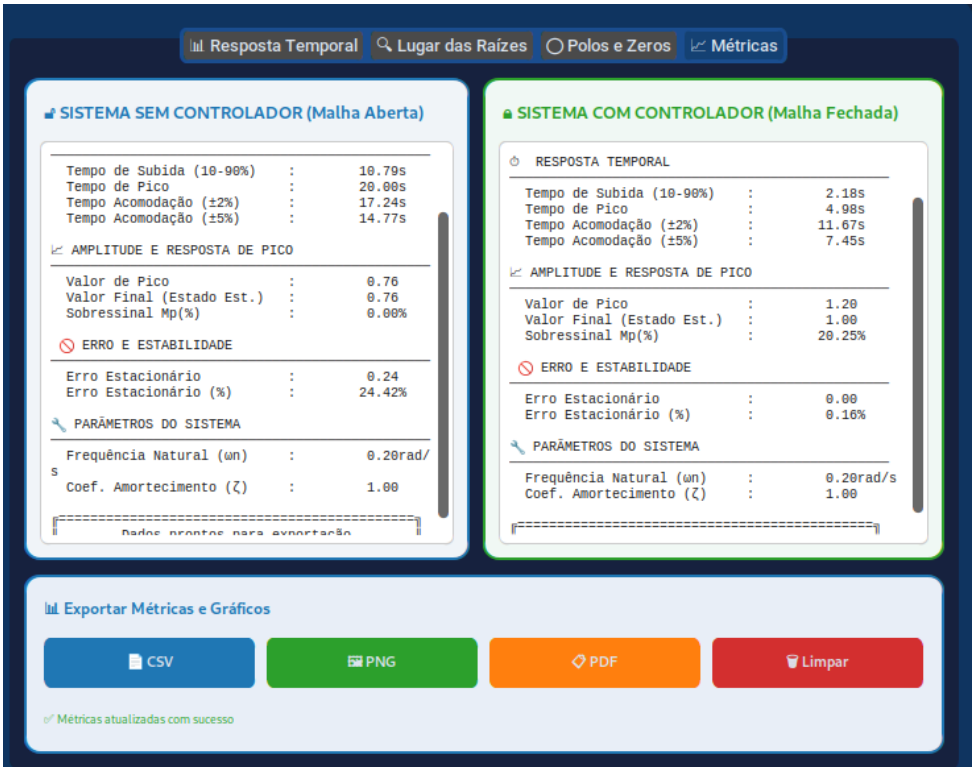
Fonte: Autoria própria (2025).

Figura 31 – Lugar Geométrico das Raízes do sistema em malha aberta, polos desejados e deslocamento dos polos após aplicação do controlador PI



Fonte: Autoria própria (2025).

Figura 32 – Comparação das métricas obtidas antes e depois da aplicação do controlador PI



Fonte: Autoria própria (2025).

- Caso 2: Controlador Proporcional-Derivativo (PD) — Otimização da Resposta Transitória Para validar a capacidade da ferramenta em projetar compensadores para velocidade, utilizou-se o Exemplo 9.3 de (NISE, 2013, p. 699). A planta do sistema é um modelo de terceira ordem dado por:

$$G(s) = \frac{1}{s(s+4)(s+6)}. \tag{4.9}$$

O problema proposto na literatura parte de um sistema que já opera com um ganho de ajuste $K = 43,35$ (resultando em $T_s = 3,32\,\text{s}$ e 16% de sobressinal) e deseja-se reduzir este tempo de acomodação em três vezes. A solução analítica para o controlador PD, projetada para atender a esses requisitos de desempenho, insere um zero em $-3,006$ e ajusta o ganho de malha, resultando na função:

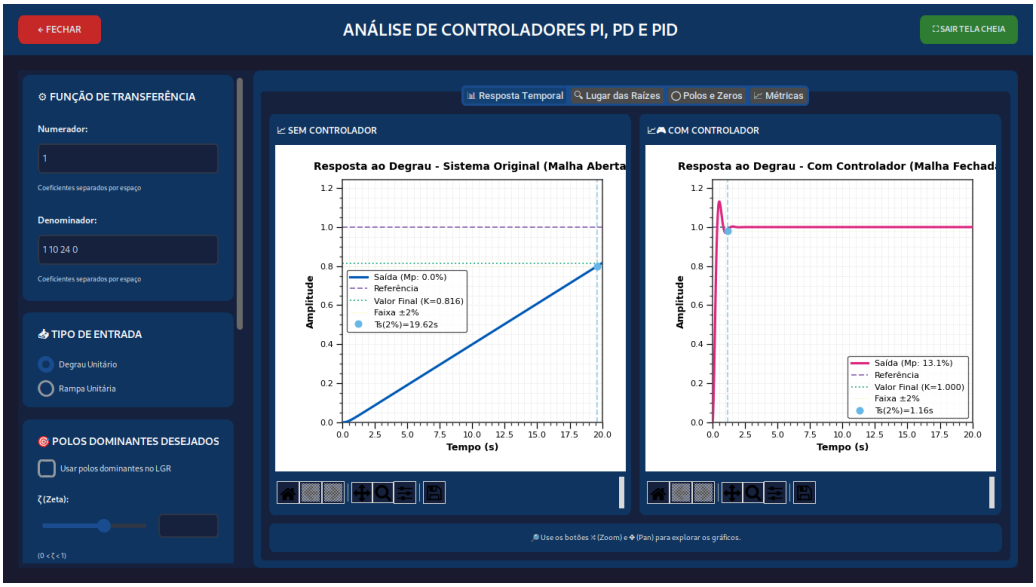
$$G_c(s) = 47,45(s + 3,006). \tag{4.10}$$

Para a validação na ferramenta desenvolvida, essa equação foi convertida para os ganhos da estrutura paralela:

$$K_p = 142,63, \qquad K_d = 47,45.$$

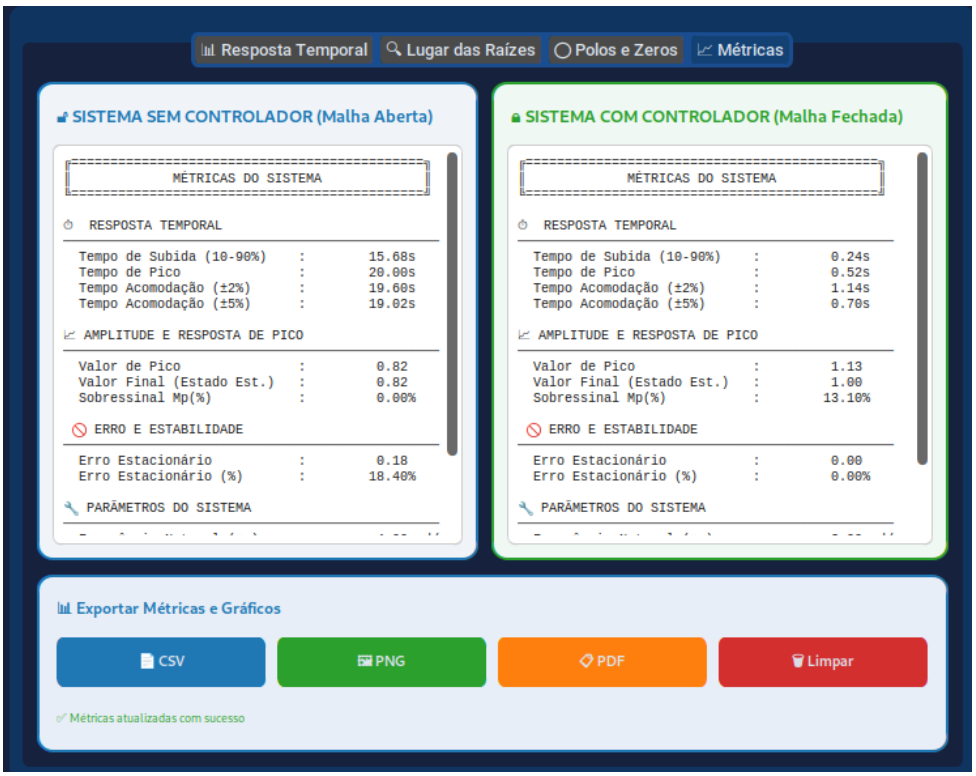
Na Figura 33 são apresentadas as respostas do sistema sem controlador e com controlador para uma entrada do tipo degrau unitário. Baseado na Figura 33, observa-se que, sem controlador, a resposta do sistema exibe a dinâmica da planta pura ($K = 1$), apresentando um comportamento lento. Com a inserção do controlador PD, a resposta do sistema atinge os objetivos de projeto, resultando em um tempo de acomodação de $1,16\,\text{s}$ e sobressinal de $13,10\%$, conforme ilustrado na Figura 34. A aplicação do controlador PD reduziu o tempo de acomodação de forma significativa, validando a eficácia do zero na otimização da resposta transitória.

Figura 33 – Resposta ao degrau comparativa. Nota: O gráfico à esquerda (’Sem Controlador’) exibe a dinâmica natural da planta com ganho unitário ($K = 1$), resultando em uma resposta lenta. O gráfico à direita (’Com Controlador’) demonstra a eficácia do PD aplicado, acelerando a resposta



Fonte: Autoria própria (2025).

Figura 34 – Quadro de métricas detalhado gerado pela ferramenta, confirmando a redução do tempo de acomodação para a faixa de 1,1s



Fonte: Autoria própria (2025).

- Caso 3: Controlador Proporcional–Integral–Derivativo (PID) — Levitador eletromagnético

Para a validação do controlador PID em um sistema instável, utilizou-se o problema do levitador eletromagnético apresentado em (CASTRUCCI; BITTAR; SALLES, 2011, p. 48). O objetivo do projeto é estabilizar a planta e garantir que os polos dominantes de malha fechada apresentem coeficiente de amortecimento $\zeta = 0{,}5$ e frequência natural $\omega_n = 2\,\text{rad/s}$.

A função de transferência da planta é dada por:

$$G(s) = \frac{1}{s^2 - 1} = \frac{1}{(s-1)(s+1)}. \tag{4.11}$$

Os polos de malha fechada desejados, calculados a partir das especificações de desempenho, são:

$$s_{1,2} = -\zeta\omega_n \pm j\,\omega_n\sqrt{1-\zeta^2} = -1 \pm j\sqrt{3}.$$

A estratégia de projeto adotada na literatura consiste no cancelamento do polo estável da planta ($s = -1$) por um dos zeros do controlador PID. Assume-se a estrutura do controlador na forma fatorada:

$$G_c(s) = \frac{K(s+a)(s+b)}{s}. \tag{4.12}$$

Fixando-se um zero em $a = 1$ para efetuar o cancelamento, o problema reduz-se a encontrar a posição do segundo zero (b) e o ganho (K). Utilizando a condição de fase do Lugar Geométrico das Raízes, determina-se que o segundo zero deve estar em $b \approx 1,33$. Pela condição de módulo, obtém-se o ganho $K = 3$.

A função de transferência resultante do controlador é:

$$G_c(s) = \frac{3(s+1)(s+1,33)}{s}. \tag{4.13}$$

Para a simulação na ferramenta desenvolvida, a equação (4.13) foi expandida para a forma paralela padrão (K_p, K_i, K_d):

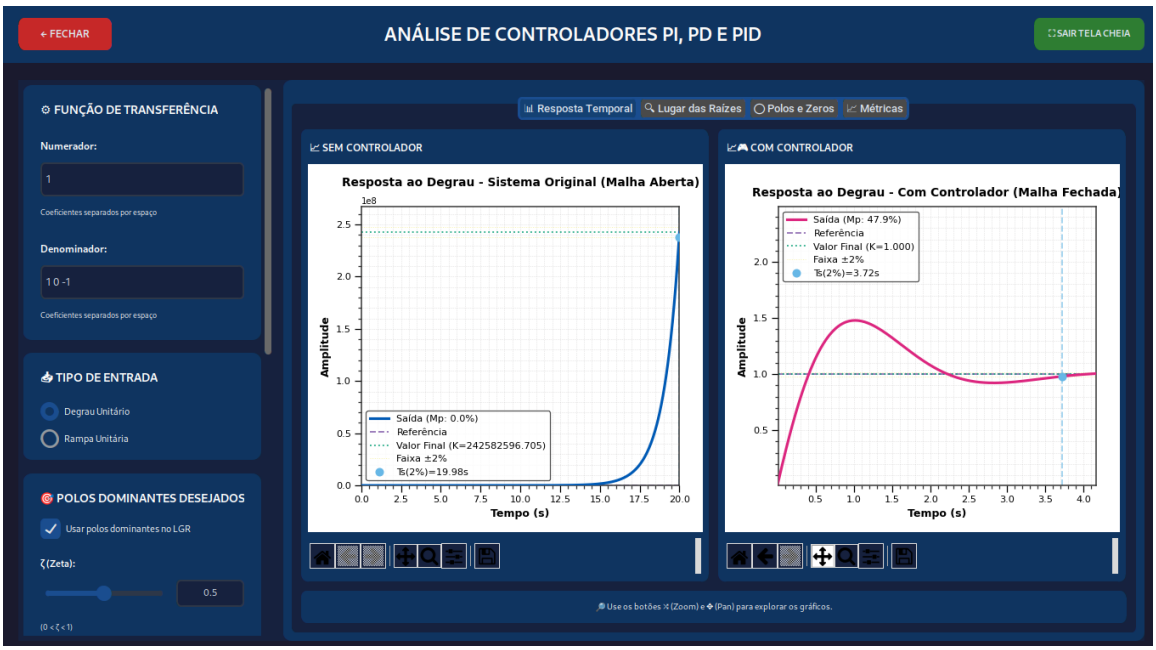
$$G_c(s) = \frac{3(s^2 + 2,33s + 1,33)}{s} = 3s + 7 + \frac{4}{s}.$$

Dessa forma, os ganhos configurados no módulo de análise foram:

$$K_p = 7, \qquad K_i = 4, \qquad K_d = 3.$$

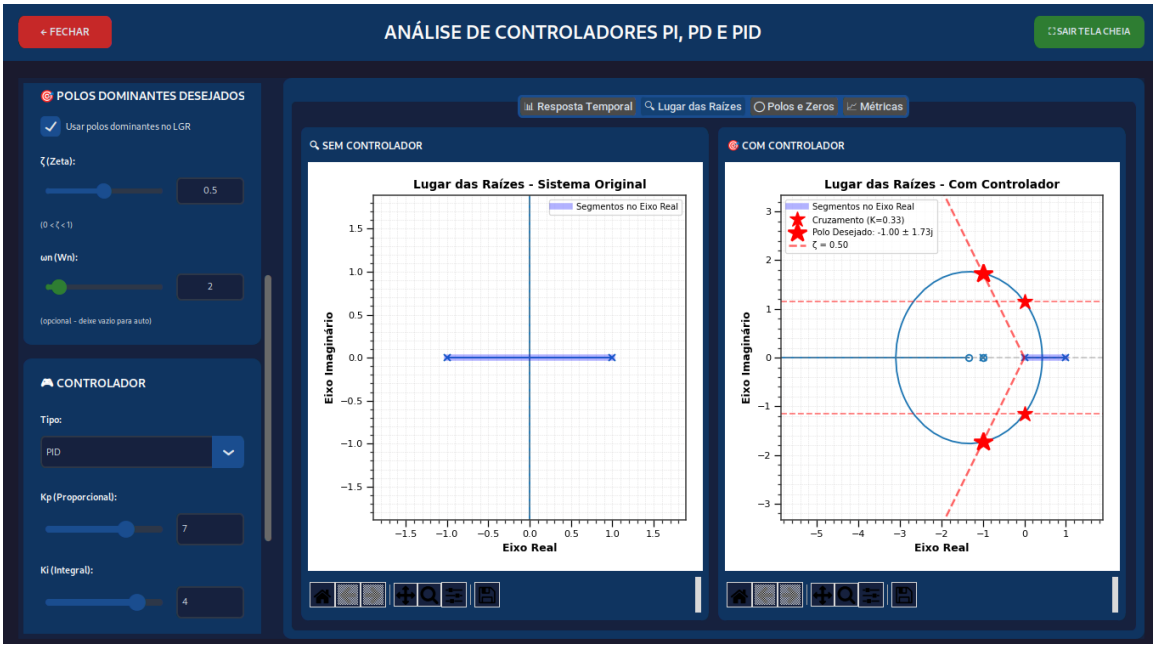
Na Figura 35, são apresentadas as respostas do sistema sem controlador e com controlador para uma entrada do tipo degrau unitário. De acordo com a Figura 35, o sistema sem controlador apresenta um erro em regime permanente de 3700%. Com a inserção do controlador PID, a resposta do sistema apresenta erro nulo em regime permanente, tempo de acomodação de $3,72$ s e sobressinal de aproximadamente $47,9\%$, conforme ilustrado na Figura 37. Na Figura 36, é apresentado o LGR do sistema sem controlador e com controlador. Com a inserção do PID, o LGR contempla os polos desejados, o que corrobora a estratégia de cancelamento polo-zero, alocando os polos de malha fechada exatamente na posição projetada.

Figura 35 – Resposta ao degrau do sistema levitador com PID. O sistema, originalmente instável, foi estabilizado e apresenta erro nulo



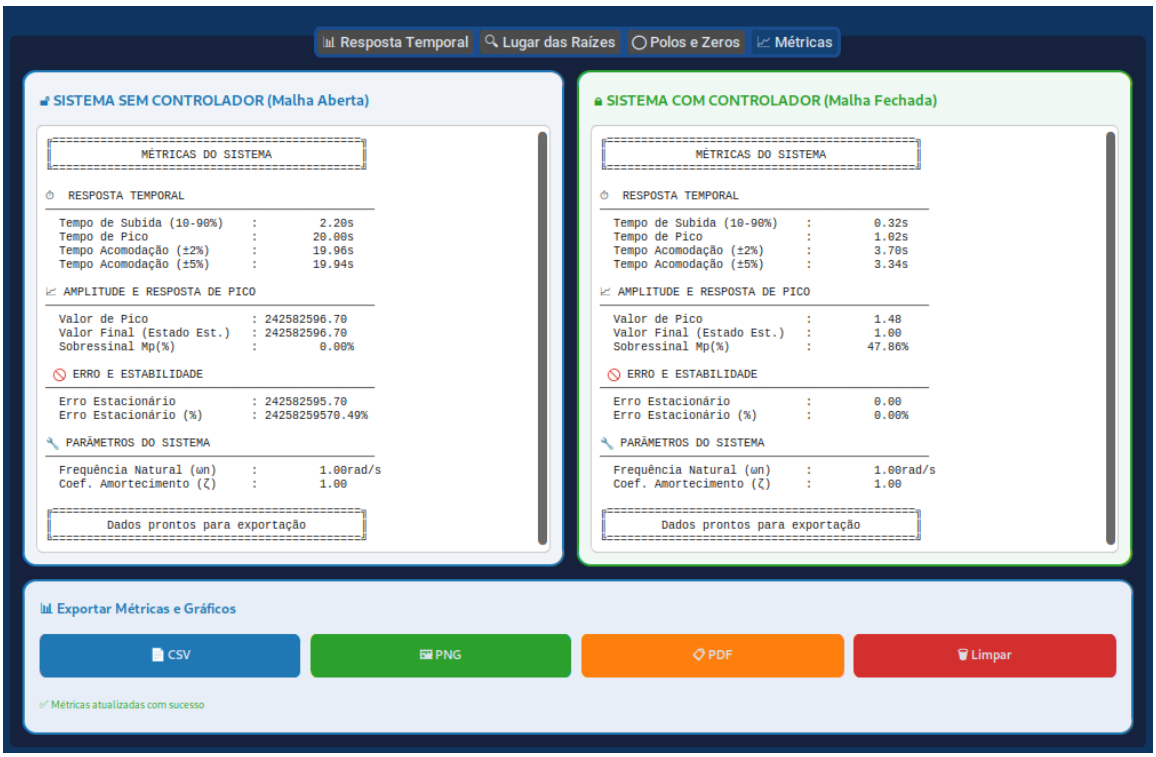
Fonte: Autoria própria (2025).

Figura 36 – Lugar Geométrico das Raízes gerado pela ferramenta. As marcas vermelhas confirmam a alocação exata dos polos em $-1 \pm j1,73$, validando o algoritmo de cálculo



Fonte: Autoria própria (2025).

Figura 37 – Métricas de desempenho calculadas pela ferramenta para o sistema compensado



Fonte: Autoria própria (2025).

5 CONCLUSÕES

O presente trabalho alcançou com sucesso seus objetivos ao desenvolver uma ferramenta computacional robusta, de código aberto e com interface amigável, capaz de apoiar o ensino e a prática de conceitos fundamentais da teoria de controle. A aplicação, implementada em Python com uma arquitetura modular, demonstrou eficácia técnica na análise de estabilidade via Routh-Hurwitz, na caracterização detalhada de sistemas de segunda ordem, na construção precisa do Lugar Geométrico das Raízes (LGR) e na simulação interativa de controladores PID.

Os resultados obtidos confirmam a aderência da ferramenta à teoria clássica de controle, com precisão numérica comprovada por meio da comparação com soluções analíticas e exemplos da literatura consagrada. Além disso, a capacidade de visualização simultânea de múltiplas representações (resposta temporal, plano-s e métricas de desempenho) e a possibilidade de exportação de relatórios técnicos reforçam o potencial da ferramenta como recurso de apoio pedagógico.

Diferentemente de ferramentas proprietárias de alto custo, a aplicação proposta é gratuita, multiplataforma e de fácil distribuição via executável, tornando-se especialmente relevante para instituições públicas e estudantes com recursos limitados. Dessa forma, o trabalho não apenas cumpre seu propósito acadêmico de engenharia de software aplicada ao controle, mas também promove a inclusão digital e a democratização do acesso a tecnologias educacionais.

5.1 Trabalhos futuros

Com base no desenvolvimento realizado e nos resultados obtidos, identificam-se oportunidades para a expansão e aprimoramento da ferramenta, bem como para a validação de sua eficácia no processo de ensino-aprendizagem. Sugere-se:

- Validação didática em sala de aula: Realizar a aplicação da ferramenta em turmas de graduação das disciplinas de Sistemas de Controle, seguida da aplicação de um formulário de percepção. O objetivo é quantificar a aceitação dos alunos e avaliar estatisticamente se o uso do *software* contribui efetivamente para a compreensão dos conceitos teóricos abordados.

- Módulo de modelagem de sistemas: Desenvolver um novo módulo dedicado à modelagem matemática, permitindo que o usuário insira as equações diferenciais do sistema físico (mecânico, elétrico, etc.) e a ferramenta realize a conversão automática para a Função de Transferência.

- Simplificação de diagramas de blocos: Implementar uma funcionalidade para a análise de diagramas de blocos e fluxo de sinal, permitindo a redução automática de sistemas

complexos interconectados para uma única função de transferência de malha fechada equivalente.

REFERÊNCIAS

ASTRöM, K. J.; HäGGLUND, T. **PID Controllers: Theory, Design, and Tuning**. 2. ed. Research Triangle Park, NC: Instrument Society of America (ISA), 1995.

BEQUETTE, B. W. **Process Control: Modeling, Design and Simulation**. Upper Saddle River, NJ: Prentice Hall, 2003.

CASTRUCCI, P. d. L.; BITTAR, A.; SALLES, R. M. **Controle Automático**. 2. ed. Rio de Janeiro: LTC, 2011.

DORF, R. C.; BISHOP, R. H. **Modern Control Systems**. Upper Saddle River, NJ: Prentice Hall, 2010. 723 p.

FULLER, S.; GREINER, B.; MOORE, J.; MURRAY, R.; PAASSEN, R. van; YORKE, R. The python control systems library (python-control). In: IEEE. **60th IEEE Conference on Decision and Control (CDC)**. 2021. p. 4875–4881. Disponível em: <https://github.com/python-control/python-control>. Acesso em: 13 nov. 2025.

HARRIS, C. R.; MILLMAN, K. J.; WALT, S. J. van der et al. Array programming with NumPy. **Nature**, v. 585, p. 357–362, 2020. Disponível em: <https://numpy.org/>. Acesso em: 13 nov. 2025.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE Computer Society, v. 9, n. 3, p. 90–95, 2007. Disponível em: <https://matplotlib.org/>. Acesso em: 13 nov. 2025.

NISE, N. S. **Engenharia de Sistemas de Controle**. 6. ed. Rio de Janeiro: LTC, 2013. 1285 p.

OGATA, K. **Engenharia de Controle Moderno**. 5. ed. São Paulo: Pearson Prentice Hall, 2010. Tradução de Heloísa Coimbra de Souza. Revisão técnica de Eduardo Aoun Tannuri.

PYTHON SOFTWARE FOUNDATION. **Python Programming Language – Official Website**. 2025. Disponível em: <https://www.python.org/>. Acesso em: 13 nov. 2025.

REENSKAUG, T. **Models–Views–Controllers**. Palo Alto, CA, 1979. Technical Note. Disponível em: <https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html>. Acesso em: 13 nov. 2025.

REPORTLAB Inc. **ReportLab Open Source Library**. [S.l.], 2025. Biblioteca de código aberto para geração de documentos PDF em Python. Disponível em: <https://www.reportlab.com/opensource/>. Acesso em: 13 nov. 2025.

SCHIMANSKY, T. **CustomTkinter: Modern GUI for Python**. 2025. Disponível em: <https://github.com/TomSchimansky/CustomTkinter>. Acesso em: 10 nov. 2025.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E. et al. SciPy 1.0: Fundamental algorithms for scientific computing in python. **Nature Methods**, v. 17, p. 261–272, 2020. Disponível em: <https://scipy.org/>. Acesso em: 13 nov. 2025.

WOOLF, P. **Chemical Process Dynamics and Controls**. Ann Arbor, MI: LibreTexts, 2025. Disponível em: <https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Book%3A_Chemical_Process_Dynamics_and_Controls_(Woolf)>. Acesso em: 12 nov. 2025.

APÊNDICE A– RECURSOS DIGITAIS DO PROJETO

Para garantir a transparência, a reprodutibilidade e a continuidade, este trabalho disponibilizou todos os artefatos de *software* produzidos em um repositório público de versionamento. O repositório do trabalho contém a implementação integral da ferramenta em Python, todos os módulos de cálculo numérico, a interface gráfica de usuário implementada em *CustomTkinter* e os *scripts* de automação:

Acesso ao Repositório

O código-fonte e a documentação técnica podem ser acessados diretamente através do endereço eletrônico abaixo:

URL DO PROJETO NO GITHUB:

<https://github.com/luisFernandoJv/sistema-de-controle---an-lise-de-sistema-de-segunda-ordem-.git>

Acesso Rápido (QR Code)

Para facilitar a consulta via dispositivos móveis, disponibiliza-se o código de resposta rápida a seguir:

Figura A.1 – Escaneie para acessar o código-fonte e o instalador.