

PYTHON

aplicado a

Sinais e Sistemas



ORGANIZAÇÃO



DIREITOS AUTORAIS

Esta apostila tem como objetivo familiarizar a comunidade interna e externa da UFG com a aplicação da linguagem de programação Python na análise de sinais e sistemas. O conteúdo desta apostila é uma combinação de material autoral do PET Engenharias - UFG e de materiais de terceiros, devidamente citados e referenciados. Portanto, este material está sujeito a direitos autorais e para reutilizar este material é necessário dar créditos ao grupo PET Engenharias - UFG e não utilizá-lo para fins comerciais.

Agradecimentos

O PET-Engenharias expressa seus sinceros agradecimentos a todos que contribuíram para a realização desta apostila.

À todos os membros do PET-EMC, cujas contribuições individuais foram essenciais para o desenvolvimento, conclusão e divulgação deste material.

Ao tutor atual do PET-Engenharias, Marlipe Garcia Fagundes Neto, por idealizar a proposta inicial do curso e fornecer consultoria fundamental na elaboração da apostila.

Aos professores Marco Antonio Assfalk de Oliveira, Sandrerley Ramos Pires, Sérgio Pires Pimentel e Alisson Assis Cardoso, que colaboraram ativamente na revisão do conteúdo e na divulgação do curso em suas respectivas disciplinas.

Ao Ministério da Educação (MEC) e ao Fundo Nacional de Desenvolvimento da Educação (FNDE) pelo financiamento fornecido ao Programa de Educação Tutorial.

Prefácio

Ministrado pelos membros do grupo PET-Engenharias (Programa de Educação Tutorial) da Universidade Federal de Goiás, este curso adota uma abordagem prática para o estudo de sinais e sistemas, abrangendo conteúdos das disciplinas de Análise de Sistemas Lineares e Sistemas de Controle, utilizando a linguagem de programação *Python* como ferramenta de apoio à análise computacional.

Cada conteúdo é apresentado com uma revisão teórica dos pontos mais relevantes, seguida de aplicações práticas que exploram a aplicação do Python na modelagem e análise de sistemas lineares. Todo o curso foi planejado e executado pelos próprios petianos, com o objetivo de oferecer um aprendizado claro e coeso, além de fortalecer as habilidades de programação em uma das linguagens mais versáteis da atualidade, amplamente utilizada no meio acadêmico para diversas aplicações.

Sumário

1	Introdução aos Sinais e Sistemas	8
1.1	Definição e classificação de Sinais	8
1.1.1	Sinais Contínuos e Discretos no Tempo	8
1.1.2	Sinais Analógicos e Digitais	8
1.1.3	Sinais Periódicos e Não-Periódicos	9
1.2	Propriedades dos Sinais	9
1.2.1	Tamanho do Sinal	9
1.2.2	Energia do Sinal	9
1.2.3	Potência do Sinal	10
1.3	Operações com Sinais	10
1.3.1	Deslocamento Temporal	10
1.3.2	Escalamento Temporal	11
1.3.3	Reversão Temporal	12
1.3.4	Operações Combinadas	13
1.4	Modelos Úteis de Sinais	13
1.4.1	Função Degrau Unitário	13
1.4.2	Função Impulso Unitário	14
2	Conceitos Fundamentais de Sistemas	16
2.1	Linearidade	16
2.2	Invariância no Tempo	17
2.3	Causalidade	18
2.4	Inversibilidade	19

2.5	Descrição Interna e Externa de um Sistema	20
2.5.1	Descrição Externa	20
2.5.2	Descrição Interna	20
2.5.3	Comparação e Escolha da Abordagem	21
3	Análise no Domínio do Tempo	22
3.1	Resposta a Entrada Zero	22
3.2	Resposta ao Impulso Unitário	26
3.3	Resposta ao Estado Nulo	28
3.4	Variáveis de Estado	30
4	Análise no Domínio da Frequência	33
4.1	Transformada de Laplace	33
4.2	Transformada Inversa de Laplace	37
4.3	Funções de Transferência	39
4.3.1	Resolução de EDO utilizando Transformada de Laplace	40
4.3.2	Polos e Zeros da Função de Transferência	43
4.4	Resposta em frequência	47
4.4.1	Diagrama de Bode	48
4.4.2	Módulo e Fase	50
5	Estabilidade	51
5.1	Estabilidade BIBO	51
5.2	Critério de Estabilidade Routh-Hurwitz	53
6	Projeto de Sistemas	61
6.1	Diagrama de Blocos	61
6.1.1	Introdução ao Diagrama de Blocos	61
6.1.2	Elementos de um Diagrama de Blocos	61
6.1.3	Exemplo de Diagrama de Blocos	62
6.2	Redução de Diagrama de Blocos	62

6.2.1	Introdução à Redução de Diagramas de Blocos . .	62
6.2.2	Regras de Redução de Diagrama de Blocos	63
6.2.3	Regras de Desentrelaçamento	64
6.2.4	Conclusão	65
6.2.5	66
6.2.6	Regra de Mason	67
7	Modelagem de Sistemas	74
7.1	Suspensão automobilística	74
7.2	Circuito integrador	78
7.2.1	Circuito integrador - Componentes passivos . . .	78
7.2.2	Circuito integrador - Componentes ativos	80

Capítulo 1

Introdução aos Sinais e Sistemas

O conteúdo deste capítulo é tratado de forma mais abrangente na obra de (LATHI, 2007). Para um estudo mais aprofundado é recomendado, tanto a consulta a esse material, como aos demais livros indicados.

1.1 Definição e classificação de Sinais

Um sinal é uma função que carrega informação, tipicamente variando com o tempo ou espaço. Matematicamente, costuma-se representar um sinal como $x(t)$, onde t é o tempo. Sinais podem ser unidimensionais (como sinais de áudio) ou multi-dimensionais (como imagens).

1.1.1 Sinais Contínuos e Discretos no Tempo

- Sinais Contínuos: definidos para todos os valores de sua variável independente. Exemplo: $x(t) = \sin(t)$.
- Sinais Discretos: definidos apenas para valores específicos da variável independente. Exemplo: $x[n] = \alpha^n$ para um inteiro n .

1.1.2 Sinais Analógicos e Digitais

- Sinais Analógicos: assumem valores contínuos no tempo. Exemplo: a tensão provida pela bateria em um determinado circuito elétrico.

- Sinais Digitais: assumem apenas valores discretos no tempo. Exemplo: dados binários em um computador (0 ou 1).

1.1.3 Sinais Periódicos e Não-Periódicos

- Sinais Periódicos: repetem seus valores em intervalos regulares. Um sinal $x(t)$ é periódico com período T se $x(t) = x(t + T)$ para todo t . Exemplo: $x(t) = \cos(2\pi t)$ tem um período de uma unidade de tempo.
- Sinais Não-periódicos: não exibem comportamento periódico. Exemplo: $x(t) = e^{-t}$.

1.2 Propriedades dos Sinais

1.2.1 Tamanho do Sinal

O tamanho ou amplitude de um sinal refere-se ao seu valor num determinado ponto. Por exemplo, em um sinal de áudio, a amplitude está relacionada ao volume do som.

- Amplitude de Pico: o valor máximo absoluto da amplitude de um sinal.
- Amplitude Pico-a-Pico: diferença entre as amplitudes máxima e mínima do sinal.

1.2.2 Energia do Sinal

A energia de um sinal é a medida total de sua força por toda a sua duração. Para um sinal contínuo no tempo $x(t)$, a energia E é definida como:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt$$

Para um sinal discreto no tempo $x[n]$, sua energia é:

$$E = \sum_{n=-\infty}^{\infty} |x[n]|^2 \Delta t \quad (1.1)$$

Sinais de energia têm energia total finita. Exemplos incluem um pulso unitário ou uma senóide amortecida.

1.2.3 Potência do Sinal

A potência média de um sinal é a energia média pelo tempo. Para um sinal contínuo no tempo $x(t)$, a potência média P é definida como:

$$P = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T |x(t)|^2 dt$$

Para um sinal discreto no tempo, a potência média é:

$$P = \lim_{N \rightarrow \infty} \frac{1}{2N} \sum_{n=-N}^N |x[n]|^2 \quad (1.2)$$

Sinais de potência têm potência média finita, porém energia infinita. Por exemplo, sinais senoidais e ondas quadradas.

1.3 Operações com Sinais

1.3.1 Deslocamento Temporal

Deslocamento temporal é avançar ou retroceder um sinal no tempo. Para um sinal $x(t)$, um deslocamento temporal por t_0 é denotado por $x(t - t_0)$.

- Se $t_0 > 0$, o sinal é avançado (movido para a esquerda).
- Se $t_0 < 0$, o sinal é atrasado (movido para a direita).

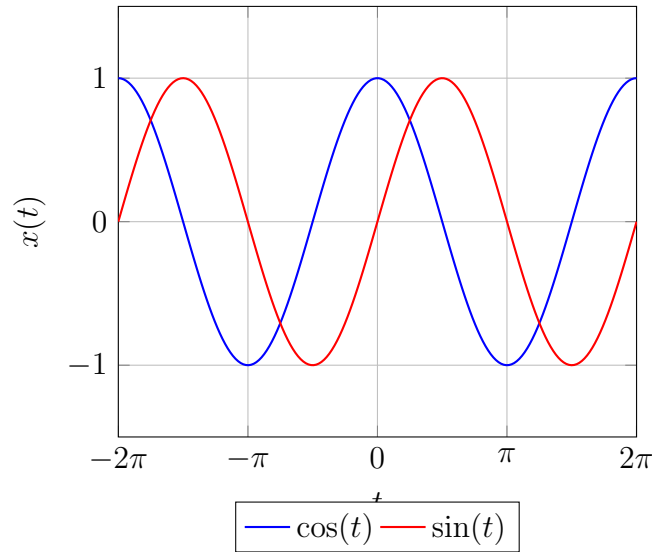


Figura 1.1: Gráficos das funções $\cos(t)$ e $\sin(t)$ no intervalo $[-2\pi, 2\pi]$.

Exemplo: Para $x(t) = \cos(t)$, $x(t - \pi/2) = \cos(t - \pi/2) = \sin(t)$ (Figura 1).

O gráfico mostra a função cosseno $\cos(t)$ em azul e sua versão deslocada $\sin(t)$ em vermelho, demonstrando como um deslocamento de $\pi/2$ transforma um sinal de cosseno em um sinal de seno.

1.3.2 Escalamento Temporal

O escalamento temporal expande ou comprime um sinal. Para um sinal $x(t)$, escalar por um fator a é denotado por $x(at)$.

- Se $|a| > 1$, o sinal é comprimido no tempo.
- Se $0 < |a| < 1$, o sinal é esticado no tempo.
- Se $a < 0$, o sinal é revertido no tempo.

Exemplo: Para $x(t) = \cos(t)$, $x(2t) = \cos(2t)$ é um sinal de frequência maior (Figura 2).

O gráfico mostra a função $\cos(t)$ em azul e a sua versão escalada no tempo em vermelho. Note que o sinal escalado tem o dobro da frequência

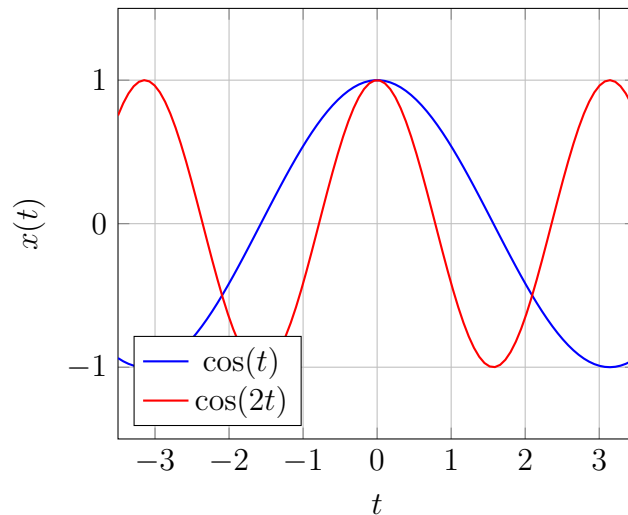


Figura 1.2: $x(t) = \cos(t)$ e $x(2t) = \cos(2t)$

do sinal original, ilustrando uma compressão no tempo por um fator $a = 2$.

1.3.3 Reversão Temporal

A reversão temporal reflete um sinal em relação ao eixo vertical. Para um sinal $x(t)$, a reversão temporal é denotada por $x(-t)$.

Exemplo: Para $x(t) = e^{-t}u(t)$, $x(-t) = e^t u(-t)$ é um sinal refletido e invertido (Figura 3).

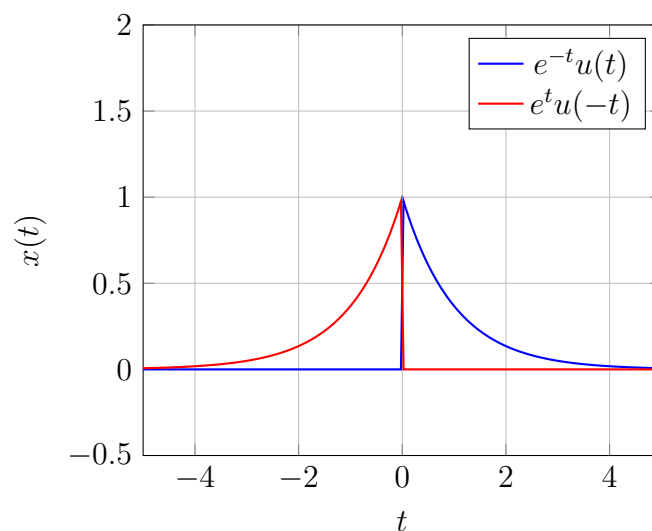


Figura 1.3: $x(t) = e^{-t}u(t)$ e $x(-t) = e^t u(-t)$

O gráfico mostra a função $x(t) = e^{-t}u(t)$ em azul e sua versão refletida e invertida $x(-t) = e^t u(-t)$ em vermelho. Note que o sinal azul decai exponencialmente para $t \geq 0$ e é zero para $t < 0$, enquanto o sinal vermelho cresce exponencialmente para $t \leq 0$ e é zero para $t > 0$. Ambas as funções se encontram no ponto $(0,1)$, ilustrando o efeito da reflexão temporal e inversão da função exponencial.

1.3.4 Operações Combinadas

Múltiplas operações podem ser aplicadas a um sinal em sequência. A ordem dessas operações importa.

Exemplo: Considere $y(t) = x(-2t + 3)$. Primeiro, multiplique t por -2 (escaleamento temporal e reversão) e, então, adicione 3 (deslocamento).

1.4 Modelos Úteis de Sinais

No estudo de sinais e sistemas, dois modelos fundamentais se destacam por sua importância e aplicabilidade: a função degrau unitário e a função impulso unitário. Estes modelos são essenciais para compreender o comportamento de sistemas lineares e são frequentemente utilizados como sinais de teste e análise.

1.4.1 Função Degrau Unitário

A função degrau unitário, denotada por $u(t)$, é definida matematicamente como:

$$u(t) = \begin{cases} 0, & \text{para } t < 0 \\ 1, & \text{para } t \geq 0 \end{cases}$$

Esta função representa uma mudança abrupta de 0 para 1 no instante $t = 0$. Sua importância reside na capacidade de modelar sistemas que

ligam ou desligam instantaneamente, como a ativação repentina de um interruptor elétrico.

Aplicações da função degrau unitário incluem:

- Análise de circuitos: Modelagem da ligação repentina de uma fonte de tensão ou corrente.
- Sistemas de controle: Representação de mudanças abruptas no setpoint de um sistema.
- Processamento de sinais: Entrada de teste para analisar a resposta de sistemas.

1.4.2 Função Impulso Unitário

A função impulso unitário, também conhecida como função delta de Dirac $\delta(t)$, é um conceito matemático ideal que representa um pulso infinitamente curto e infinitamente alto. Embora não possa ser realizada fisicamente, é extremamente útil na análise teórica de sistemas.

Propriedades fundamentais da função impulso:

1. $\delta(t) = 0$ para todo $t \neq 0$
2. $\int_{-\infty}^{\infty} \delta(t) dt = 1$ (a área sob a curva é 1)
3. Para qualquer função contínua $f(t)$: $\int_{-\infty}^{\infty} f(t) \delta(t - a) dt = f(a)$ (propriedade de amostragem)

Aplicações e interpretações da função impulso unitário:

- Determinação da resposta ao impulso de sistemas lineares.
- Análise espectral: A transformada de Fourier do impulso unitário é uma constante em todas as frequências.
- Modelagem de fenômenos físicos de curta duração, como colisões.

Uma relação importante entre a função degrau e a função impulso é:

$$\delta(t) = \frac{d}{dt}u(t)$$

Esta relação é particularmente útil na análise de sistemas, pois permite relacionar a resposta ao degrau com a resposta ao impulso de um sistema.

Capítulo 2

Conceitos Fundamentais de Sistemas

Este capítulo baseia-se principalmente na obra de (LATHI, 2007). Para um aprofundamento no tema, recomenda-se a consulta a esse material, assim como às demais referências bibliográficas apresentadas ao final desta apostila.

Ao estudar sistemas, várias propriedades fundamentais são consideradas para classificá-los e analisá-los. Entre estas, destacam-se a linearidade, a invariância no tempo, a causalidade e a inversibilidade.

2.1 Linearidade

A linearidade é uma propriedade crucial que divide os sistemas em duas categorias principais: lineares e não-lineares.

Sistemas Lineares:

Um sistema é considerado linear se obedece aos princípios de superposição e homogeneidade. Matematicamente, para um sistema linear T :

$$T[ax_1 + bx_2] = aT[x_1] + bT[x_2]$$

Características dos sistemas lineares:

- Podem ser analisados usando técnicas como transformadas de Laplace e Fourier.

- A resposta a entradas complexas pode ser decomposta em respostas a entradas mais simples.
- Exemplos incluem amplificadores ideais e filtros lineares.

Sistemas Não-Lineares:

Sistemas não-lineares são aqueles que não obedecem ao princípio da superposição.

Características dos sistemas não-lineares:

- Podem exibir comportamentos complexos como caos e bifurcações.
- A resposta pode depender da amplitude do sinal de entrada.
- Exemplos incluem sistemas com saturação e sistemas com histerese.

2.2 Invariância no Tempo

A invariância no tempo é outra propriedade fundamental que afeta significativamente a análise e o comportamento dos sistemas.

Sistemas Invariantes no Tempo (SIT):

Um sistema é invariante no tempo se sua resposta a uma determinada entrada não depende do instante específico em que essa entrada é aplicada. Matematicamente, para um SIT T :

Se $y(t) = T[x(t)]$, então $y(t - \tau) = T[x(t - \tau)]$ para qualquer τ

Características dos SIT:

- A resposta depende apenas da forma do sinal de entrada, não do momento em que é aplicado.
- São mais fáceis de analisar do que sistemas variantes no tempo.
- Exemplos incluem circuitos RC, RL e RLC com componentes fixos.

Sistemas Variantes no Tempo (SVT):

Um sistema é variante no tempo se sua resposta a uma determinada entrada depende do instante específico em que essa entrada é aplicada.

Características dos SVT:

- A resposta do sistema pode mudar ao longo do tempo, mesmo para a mesma entrada.
- Requerem técnicas de análise mais avançadas.
- Exemplos incluem sistemas de controle adaptativo e canais de comunicação com desvanecimento.

2.3 Causalidade

A causalidade é uma propriedade relacionada à dependência temporal entre entrada e saída.

Sistemas Causais:

Um sistema é causal se sua saída em qualquer instante depende apenas de entradas presentes e passadas, nunca de entradas futuras.

Características dos sistemas causais:

- São realizáveis fisicamente em tempo real.
- A resposta ao impulso $h(t)$ é zero para $t < 0$.
- Exemplos incluem a maioria dos sistemas físicos reais.

Sistemas Não-Causais:

Um sistema é não-causal se sua saída em um determinado instante pode depender de entradas futuras.

Características dos sistemas não-causais:

- Não são realizáveis fisicamente em tempo real.
- Úteis em processamento off-line de sinais.
- Exemplos incluem alguns filtros de fase linear.

2.4 Inversibilidade

A inversibilidade está relacionada à possibilidade de recuperar o sinal de entrada a partir do sinal de saída.

Sistemas Inversíveis:

Um sistema é inversível se existe um sistema inverso que, quando aplicado à saída, recupera exatamente a entrada original. Matematicamente, para um sistema inversível T , existe T^{-1} tal que:

$$T^{-1}[T[x(t)]] = x(t)$$

Características dos sistemas inversíveis:

- Existe uma correspondência um-para-um entre entrada e saída.
- Úteis em aplicações onde a reconstrução do sinal original é necessária.
- Exemplos incluem amplificadores lineares sem saturação.

Sistemas Não-Inversíveis:

Um sistema é não-inversível se não é possível recuperar unicamente a entrada a partir da saída.

Características dos sistemas não-inversíveis:

- Pode haver perda de informação no processamento do sinal.
- Múltiplas entradas podem produzir a mesma saída.
- Exemplos incluem sistemas com quantização.

Compreender estas propriedades fundamentais dos sistemas é crucial para sua análise e projeto eficazes. Cada propriedade influencia como o sistema processa sinais e como pode-se analisá-lo matematicamente. Na prática, muitos sistemas exibem uma combinação dessas propriedades, e a análise frequentemente envolve considerar trade-offs entre elas.

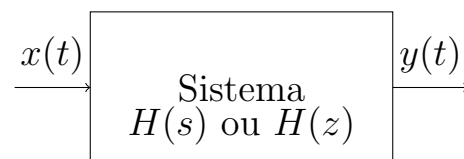
2.5 Descrição Interna e Externa de um Sistema

2.5.1 Descrição Externa

A descrição externa, também conhecida como abordagem entrada-saída, concentra-se na relação entre os sinais de entrada e saída do sistema, sem se preocupar com os detalhes internos de seu funcionamento. Esta abordagem trata o sistema como uma “caixa preta”, onde apenas as entradas e saídas são consideradas relevantes para a análise.

Um conceito fundamental na descrição externa é a função de transferência, que relaciona matematicamente a saída do sistema à sua entrada no domínio da frequência. Esta abordagem é particularmente útil para análise de resposta em frequência e é amplamente utilizada em engenharia de controle e processamento de sinais.

Exemplo gráfico de um sistema visto pela descrição externa:



Onde $x(t)$ é o sinal de entrada, $y(t)$ é o sinal de saída, e $H(s)$ ou $H(z)$ representa a função de transferência do sistema no domínio contínuo ou discreto, respectivamente.

2.5.2 Descrição Interna

Por outro lado, a descrição interna, também chamada de representação de estado, fornece uma visão detalhada da estrutura e dinâmica interna do sistema. Esta abordagem utiliza variáveis de estado para descrever o estado interno do sistema em qualquer instante de tempo.

Na representação de estado, o sistema é descrito por um conjunto de equações diferenciais de primeira ordem (para sistemas contínuos) ou equações de diferença (para sistemas discretos). Essas equações relacionam as variáveis de estado, as entradas e as saídas do sistema.

A descrição interna é particularmente vantajosa para sistemas complexos, especialmente aqueles com múltiplas entradas e saídas (MIMO - Multiple Input Multiple Output). Ela também facilita a análise de estabilidade e o projeto de controladores baseados em realimentação de estados.

2.5.3 Comparação e Escolha da Abordagem

A escolha entre a descrição externa e interna depende de vários fatores, incluindo a complexidade do sistema, o objetivo da análise e as informações disponíveis. A descrição externa é geralmente mais simples e direta, sendo ideal para análise de resposta em frequência e sistemas lineares invariantes no tempo (LTI). Já a descrição interna oferece uma compreensão mais profunda da dinâmica do sistema, sendo preferível para análise de sistemas não-lineares e projeto de controladores avançados.

Em muitos casos, uma abordagem complementar, utilizando ambas as descrições, pode fornecer uma compreensão mais completa e robusta do sistema em estudo. Engenheiros e pesquisadores frequentemente alternam entre essas duas perspectivas, aproveitando as vantagens de cada uma conforme necessário para resolver problemas específicos em sinais e sistemas.

Capítulo 3

Análise no Domínio do Tempo

Nesta seção, utilizando a linguagem de programação Python, será feita a análise no domínio do tempo de sinais, com a resolução de problemas de variáveis de estado e a busca computacional por respostas de um LCIT (Sistema Linear Contínuo e Invariante no Tempo), incluindo: resposta à entrada zero, resposta ao impulso e resposta ao estado nulo.

Os conceitos apresentados nesta seção são descritos de maneira aprofundada na obra de (LATHI, 2007) que é referenciada ao final deste trabalho.

3.1 Resposta a Entrada Zero

Sejam os sistemas lineares invariantes e contínuos no tempo (LCIT) descritos a partir de $Q_N(D)y(t) = P_M(D)x(t)$, onde D é o operador para representar $\frac{d}{dt}$ e $Q_N(D)$ e $P_M(D)$ são polinômios da seguinte forma.

$$\begin{aligned} Q(D) &= D^N + a_1 D^{N-1} + \dots + a_{N-1} D + a_N \\ P(D) &= b_{N-M} D^M + b_{N-M+1} D^{M-1} + \dots + b_{N-1} D + b_N \end{aligned}$$

Figura 3.1: Forma dos polinômios P e Q .

A resposta à entrada zero é produzida pelo sistema quando $x(t) = 0$, com uma resposta dependente das condições internas do sistemas. Assim, a solução gerada é representada por:

$$y_0(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} + \dots + c_N e^{\lambda_N t}$$

Figura 3.2: Solução genérica do problema.

▷ Implementando a solução

Importa-se o módulo *SymPy*, que utiliza matemática simbólica,¹ sob a palavra reduzida ***sp*** para simplificar as linhas de código no momento de escrever o programa.

```
import sympy as sp

def resolver_resposta_entrada_zero(QN_coeffs, cond_iniciais):
    t = sp.symbols('t')
    y = sp.Function('y')(t)

    # Definindo a equação diferencial QN(D)y(t) = 0
    QN = sum(coeff * sp.Derivative(y, t, n) for n, coeff in enumerate(QN_coeffs))
    equacao_homogenea = sp.Eq(QN, 0)
```

Figura 3.3: Definição da equação $Q_n(D)$.

1

¹A matemática simbólica trabalha com símbolos matemáticos, permitindo uma manipulação mais precisa de equações e fórmulas, oferecendo soluções exatas e uma melhor compreensão dos problemas.

A Figura anterior define os símbolos que serão utilizados na Resposta à Entrada Zero, além de definir a equação diferencial $Q_N(D)y(t) = 0$.

A seguir é feita a definição da equação característica $Q(\lambda) = 0$, substituindo D por λ .

$$\lambda^N + a_1\lambda^{N-1} + \dots + a_{N-1}\lambda + a_N = 0$$

Figura 3.4: Equação de λ geral.

então será calculada as suas raízes que serão utilizadas na formação da solução.

```
# Resolvendo a equação característica
lambda_ = sp.symbols('lambda')
equacao_caracteristica = sum(coeff * lambda_**n for n, coeff in enumerate(QN_coeffs))
raizes = sp.solve(equacao_caracteristica, lambda_)
```

Figura 3.5: Cálculo da raízes da equação característica.

Serão tratados essencialmente dos casos teóricos onde a resolução da equação característica $Q_N(\lambda)$ pode levar a dois casos: 1) Raízes distintas, com a solução genérica do problema (vista anteriormente); ou 2) Raízes iguais, onde a solução para $Q_N(\lambda)$ de grau 2 será dada por:

$$y_0(t) = (c_1 + c_2t)e^{\lambda t}$$

Figura 3.6: Solução genérica de grau 2.

As linhas de código a seguir tratam a multiplicidade das raízes iguais, adicionando fatores de t^n aos modos característicos da solução de $Q(\lambda)$.

Após isso, serão definidas as linhas de código que irão armazenar as condições iniciais para em seguida resolver o sistema de equações e determinar as constantes c_1 e c_2 .

Por fim, a equação da solução final é gerada, finalizando a função que calcula a resposta a entrada zero. Para ser usada basta o usuário digitar os coeficientes do polinômio $Q_N(D)$ e as condições iniciais, como mostrado na seguinte imagem:


```
# Montando a solução geral considerando multiplicidade de raízes
solucao_geral = 0
constantes = []
c_counter = 1

raizes_mult = sp.roots(equacao_caracteristica, lambda_)
for raiz, multiplicidade in raizes_mult.items():
    for m in range(multiplicidade):
        constante = sp.symbols(f'c{c_counter}')
        constantes.append(constante)
        solucao_geral += constante * t**m * sp.exp(raiz * t)
        c_counter += 1
```

Figura 3.7: Tratamento da multiplicidade de raízes iguais.

```
# Criando as condições iniciais
condicoes = []
for ordem, valor in enumerate(cond_iniciais):
    condicoes.append(sp.Eq(solucao_geral.diff(t, ordem).subs(t, 0), valor))

# Resolvendo o sistema de equações para encontrar os valores de c1, c2, ...
sistema_equacoes = []
for cond in condicoes:
    sistema_equacoes.append(cond.lhs - cond.rhs)

solucao_sistema = sp.solve(sistema_equacoes, constantes)
```

Figura 3.8: Cálculo das constantes C1 e C2.

```
# Substituindo as constantes na solução geral
solucao_final = solucao_geral.subs(solucao_sistema)

return solucao_final

# Exemplo de uso
QN_coeffs = [9, 6, 1] # Representa D^2 + 6D + 9
cond_iniciais = [5, -2] # y(0) = 5, y'(0) = -2

resposta_entrada_zero = resolver_resposta_entrada_zero(QN_coeffs, cond_iniciais)
print(resposta_entrada_zero)
```

Figura 3.9: Exemplo: caso de uso.

Dessa maneira, a solução do exemplo utilizado é representada na imagem a seguir.

$$13t \cdot \exp(-3t) + 5 \cdot \exp(-3t)$$

Figura 3.10: Solução da resposta à entrada zero.

3.2 Resposta ao Impulso Unitário

A resposta $h(t)$ ao impulso é a resposta do sistema a uma entrada impulsiva $\delta(t)$ aplicada em $t = 0$, com todas as condições iniciais nulas para $t = 0^-$. Dessa forma, calcula-se a resposta $h(t)$ dos sistemas lineares com coeficientes invariantes no tempo (LCIT) descritos pela equação $Q_N(D)y(t) = P_M(D)x(t)$, mas com $x(t)$ diferente de zero. A solução da resposta ao impulso é dada por $h(t) = h_0(t)$ para sistemas não instantâneos, e $h(t) = \frac{b_0}{a_0}\delta(t) + h_0(t)$ para sistemas instantâneos, em que a_0 e b_0 são os coeficientes que acompanham o termo de maior grau nos polinômios $Q_N(D)$ e $P_M(D)$, respectivamente.

►Implementando a solução

Como de costume, serão definidos os símbolos que serão utilizados, além da função de transferência $H(s)$ no domínio da frequência. Em seguida, é aplicada a transformada inversa de Laplace para obter a resposta $h_0(t)$ no domínio do tempo, finalizando a função que calcula esse resultado.

```
import sympy as sp

def resposta_ao_impulso(P_s, Q_s):
    # Define a variável simbólica para o tempo e a variável de Laplace
    t = sp.symbols('t', real=True, positive=True)
    s = sp.symbols('s')

    # Calcula a função de transferência H_s
    H_s = P_s / Q_s

    # Transformada inversa de Laplace para encontrar h_t
    h_t = sp.inverse_laplace_transform(H_s, s, t)

    return h_t
```

Figura 3.11: Função da resposta ao impulso.

Agora basta o usuário entrar com as equações $Q_N(D)$ e $P_M(D)$ e chamar a função criada no início para calcular $h_0(t)$ como mostrado na Figura a seguir:

Aqui, estas linhas de código verificam os graus dos polinômios $Q_N(D)$ e $P_M(D)$ para testar se o sistema é instantâneo ou não.

```
# Exemplo de uso:
# Definindo os polinômios diferenciais P(s) e Q(s)
# Por exemplo, Q(s) = s^2 + 7*s + 6 e P(s) = 2*s^2 + 1
s = sp.symbols('s')
Q_s = s**2 + 7*s + 6
P_s = 2*s**2 + 1

# Calculando a resposta ao impulso h_t
h_t = resposta_ao_impulso(P_s, Q_s)
```

Figura 3.12: Exemplo de $Q_N(D)$ e $P_M(D)$.

```
# Calcula os graus dos polinômios P(s) e Q(s)
grau_P = sp.degree(P_s, s)
grau_Q = sp.degree(Q_s, s)
```

Figura 3.13: Cálculo do grau de Q_N e $P_M(D)$.

Caso o sistema seja instantâneo, a resposta será exibida como mostrado na imagem abaixo.

```
# Verifica se o sistema é instantâneo (grau de P(s) == grau de Q(s)) e
# Adiciona o termo (b_0 / a_0) * DiracDelta(t) à resposta
if grau_P == grau_Q:

    # Obtém os coeficientes líderes dos polinômios P(s) e Q(s)
    b_0 = sp.LC(P_s, s) # Coeficiente líder de P(s)
    a_0 = sp.LC(Q_s, s) # Coeficiente líder de Q(s)

    # Exibindo a resposta ao impulso
    print(f"h(t) = {(b_0 / a_0)}*δ(t) + {h_t}")
```

Figura 3.14: Solução para um sistema instantâneo.

Já se o sistema não for instantâneo a resposta é do tipo $h(t) = h_0(t)$.

```
else:
    # Exibindo a resposta ao impulso
    print(f"h(t) = {h_t}")
```

Figura 3.15: Solução para um sistema não instantâneo.

Solução gerada para um sistema descrito por $(D^2 + 7D + 6)y(t) = (2D^2 + 1)x(t)$:

```
h(t) = 2*δ(t) + 3*exp(-t)/5 - 73*exp(-6*t)/5
```

Figura 3.16: Resposta para o exemplo.

3.3 Resposta ao Estado Nulo

A resposta de estado nulo de um sistema LCIT é a resposta $y(t)$ do sistema a uma entrada $x(t)$ quando o sistema está no estado nulo, ou seja, quando todas as condições são zero. Esta solução é dada pela integral de convolução:

$$\begin{aligned} y(t) &= \lim_{\Delta\tau \rightarrow 0} \sum_{\tau} x(n\Delta\tau)h(t - n\Delta\tau)\Delta\tau \\ &= \int_{-\infty}^{\infty} x(\tau)h(t - \tau) d\tau \end{aligned}$$

Figura 3.17: Definição de convolução no tempo.

Dessa forma, $y(t)$ é a resposta do sistema a uma entrada arbitrária $x(t)$ em termos da resposta $h(t)$ do impulso unitário.

▷ Implementando a solução

Primeiramente, serão definidos os símbolos e as funções $h(t)$ e $x(t)$ que serão utilizados na Resposta ao Estado Nulo.

```
import sympy as sp

# Definindo t como símbolo
t, tau = sp.symbols('t tau')

# Definindo as funções h(tau) e x(tau)
h_tau = 3 * sp.exp(-6 * tau) + sp.exp(-tau)
x_tau = sp.exp(tau)
```

Figura 3.18: Definições de cabeçalho.

Como mostrado na Figura a seguir, basta ser feita a operação de convolução por meio da função interna *integrate()* e exibir a solução $y(t)$.

```
# Definindo a convolução y(t)
convolution = sp.integrate(x_tau * h_tau.subs(tau, t - tau), (tau, 0, t))

# Exibindo a equação y(t)
print("y(t) = ", convolution.simplify())
```

Figura 3.19: Aplicação da operação de convolução sobre funções.

Com as equações $x(t)$ e $h(t)$ utilizadas de exemplo, a resposta ao Estado Nulo será dada por:

$$y(t) = (13 \cdot \exp(7 \cdot t) - 7 \cdot \exp(5 \cdot t) - 6) \cdot \exp(-6 \cdot t) / 14$$

Figura 3.20: Exemplo: Resposta ao estado nulo.

Além disso, é possível avaliar essa solução em um tempo t específico se necessário. Assim, tudo que precisa ser feito é definir um valor para t e aplicá-lo na integral de convolução.

```
# Avaliando y(t) para t = 1
t_value = 1
y_value = convolution.subs(t, t_value)
print(f"y({t_value}) =", y_value.evalf())
```

Figura 3.21: Convolução para um dado tempo t

A solução gerada para a Resposta ao Estado Nulo aplicada a um tempo $t = 1$:

$$y(1) = 2.33911679776482$$

Figura 3.22: Resposta à convolução para um tempo t .

3.4 Variáveis de Estado

As variáveis de estado são usadas para descrever o comportamento dinâmico de sistemas físicos em termos de equações diferenciais de primeira ordem. No contexto de sistemas de controle, o modelo no espaço de estados representa a dinâmica de um sistema em uma forma matricial compacta. O código a seguir faz uma conversão entre duas representações diferentes de sistemas dinâmicos: a função de transferência e o espaço de estados.

▷ Implementando a solução

Aqui, um sistema de função de transferência é definido.

```
import scipy.signal

G = scipy.signal.lti(1, [1, 1])
G

TransferFunctionContinuous(
array([1.]),
array([1., 1.]),
dt: None
)
```

Figura 3.23: Função de transferência: definição.

O primeiro argumento é o numerador 1 e o segundo argumento é o denominador $s + 1$ que pode-se acessar da seguinte forma:

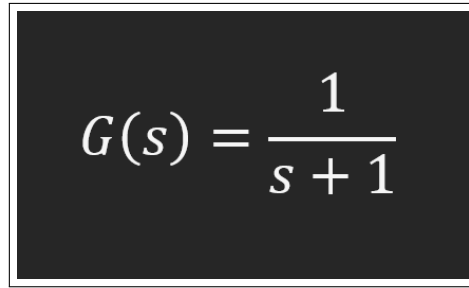
```
#Este objeto nos permite acessar o numerador e o denominador
G.num, G.den

(array([1.]), array([1., 1.]))
```

Figura 3.24: Numerador e denominador da função de transferência.

A Função de transferência $G(s)$ utilizada como exemplo foi:

O método `.to_ss()` converte a função de transferência para a representação no espaço de estados. O sistema no espaço de estados é descrito



$$G(s) = \frac{1}{s + 1}$$

Figura 3.25: Exemplo: função de transferência.

pelas matrizes A, B, C e D. Onde A é a matriz de estados, B a de entrada, C a de saída e D a matriz de alimentação.

```
#Para converter para espaço de estados, podemos usar o método .to_ss()
Gss = G.to_ss()
Gss.A, Gss.B, Gss.C, Gss.D

(array([[[-1.]]], array([[1.]]], array([[1.]]], array([[0.]]]))
```

Figura 3.26: Conversão entre função de transferência e espaço de Estados.

Esta linha recria o sistema dinâmico usando diretamente as matrizes de espaço de estados. O objeto *G2ss* agora representa o mesmo sistema, mas construído a partir de suas matrizes A, B, C e D.

```
#Podemos construir outro objeto usando as matrizes de espaço de estados em vez da forma de Laplace
G2ss = scipy.signal.lti(Gss.A, Gss.B, Gss.C, Gss.D)
G2ss

StateSpaceContinuous(
array([[[-1.]]],
array([[1.]]],
array([[1.]]],
array([[0.]]],
dt: None
)
```

Figura 3.27: Sistema representado por matrizes.

Aqui, o sistema no espaço de estados é convertido de volta para a forma de função de transferência. O método *.to_tf()* realiza essa conversão, retornando o sistema no formato de função de transferência.

```
#Podemos converter para a forma de função de transferência usando .to_tf()
G2 = G2ss.to_tf()

#Agora podemos acessar novamente o numerador e o denominador:
G2.num, G2.den
```

Figura 3.28: Conversão de espaço de Estados para função de transferência.

Finalmente, o numerador e o denominador da função de transferência $G_2(s)$ são acessados, confirmando que o sistema convertido é equivalente

ao original.



```
(array([1.]), array([1., 1.]))
```

Figura 3.29: Numerador e denominador da função de transferência equivalente.

Capítulo 4

Análise no Domínio da Frequência

Em engenharia, a análise e compreensão do comportamento de diversos tipos de sistemas são essenciais para a uma correta modelagem e representação que busque fornecer *insights* valiosos no desenvolvimento de projetos completos.

Na seção 3 foi abordado a resposta para sistemas com sinais no Domínio do Tempo, neste capítulo irá-se ir além na busca por soluções matemáticas, através da exploração da resposta dos sistemas ao observarmos os sinais no Domínio da Frequência.

Os conteúdos trabalhados nesta seção estão fundamentados e apresentados de forma mais detalhada na obras de (NISE, 2023) e de (SANDROCK, 2018), com referências presentes ao final deste trabalho.

4.1 Transformada de Laplace

A Transformada de Laplace é essencial para descrever sistemas físicos observando o sinal de entrada, a saída e o sistema (em sua totalidade) como entidades separadas. A operação realizada por essa transformada leva as funções no domínio do tempo $f(t)$ em funções do domínio da frequência $F(s)$:

▷ Implementando a solução

Utilizando o módulo *SymPy*, que utiliza matemática simbólica (rever

$$\mathcal{L}[f(t)] = F(s) = \int_{0^-}^{\infty} f(t)e^{-st} dt$$

Figura 4.1: Definição da Transformada de Laplace.

capítulo 2), é feita inicialmente a definição dos símbolos utilizados na Transformada de Laplace.

```
# Importando o módulo SymPy
import sympy as sp

# Definindo os símbolos usados na Transformada de Laplace
t, s = sp.symbols('t, s')
a = sp.symbols('a', real=True, positive=True)
omega = sp.Symbol('omega', real=True)
```

Figura 4.2: Escopo inicial do programa.

O *SymPy* dispõe de várias funções internas essenciais para o cálculo da Transformada de Laplace de forma integrada com o resto do programa. Essas funções podem ser a função seno, cosseno, exponencial, tangente, entre outras. São declaradas as funções abaixo.

```
exp = sp.exp # exponencial na base e (Euler)
sin = sp.sin # função seno
cos = sp.cos # função cosseno
```

Figura 4.3: Declaração de funções importantes para o uso no programa.

```
# Valor do coeficiente 'a'
a =

# Especificar a função 'f(t)' a ser calculada
f = t*exp(-a*t)

# Também pode ser definido um vetor de funções f(t), f_2(t), ..., f_n(t)
funcoes = [exp(-2*t), sin(3*t)]
```

Figura 4.4: Declarações das funções a serem calculadas.

Como visto no código presente na imagem anterior, se o usuário deseja especificar apenas uma função (a que será operada pela Transformada) ele pode simplesmente utilizar uma atribuição simples de variável. No caso acima foi solicitado ao programa que calculasse a Transformada de Laplace da função $f(t) = te^{at}$, onde deve ser especificado o valor da variável 'a'. Ainda, se preferir e for necessário, o usuário pode especificar um vetor de n funções na forma $[f_1(t), f_2(t), f_3(t), \dots, f_n(t)]$, foram especificadas as funções $f_1(t) = e^{-2t}$ e $f_2(t) = \text{sen}(3t)$.

A sintaxe de declaração do vetor de funções segue o mesmo padrão de declaração da linguagem Python, porém utilizando os símbolos (a, t e s) e as funções (sen, cos, exponencial, cosh, sinh, etc) que foram declaradas no escopo inicial do programa.

A função, do módulo *SymPy*, responsável por calcular a desejada Transformada de Laplace da função (ou das funções) especificadas nas linhas de código anteriores deve ser definida seguinte maneira:

```
# Definindo a função do SymPy que executa a Transformada de Laplace
def Laplace(f):
    return sp.laplace_transform(f, t, s, noconds=True)
```

Figura 4.5: Definição da função responsável por calcular a Transformada de Laplace.

A função *Laplace()* recebe como argumento a função ou o vetor de funções $f(t)$ fornecidos anteriormente. Ela retornará o cálculo da Transformada de Laplace, dessa vez como sendo uma função do tipo $F(s)$, já no domínio s de Laplace, como desejado. Para o bom funcionamento do programa o usuário deve estar atento a fazer uso de somente uma entre as duas declarações.

```
# Cálculo da Transformada de Laplace da função 'f' especificada anteriormente
Fs = Laplace(f)

# Caso tenhamos um vetor de funções, deve-se usar a sintaxe a seguir:
Fs = [Laplace(f) for f in funcoes]
```

Figura 4.6: Chamada da função que a Transformada de Laplace.

Por fim, através da chamada da função $Laplace(f)$, o resultado da operação é transferido para a variável **Fs**, e então é exibida a resposta ao usuário.

```
# Retorna o resultado da Transformada de Laplace
Fs
```

Figura 4.7: Resultado do cálculo da Transformada de Laplace.

As respostas da Transformada de Laplace para as funções $f(t) = te^{at}$, com $a = 7$, $f_1(t) = e^{-2t}$ e $f_2(t) = \text{sen}(3t)$, seguindo o processo mostrado até aqui, estão reunidas na imagem do terminal a seguir. Caso queira confirmar, o usuário é encorajado a obter essas respostas manualmente.

A respostas esperadas no terminal para a função $f(t) = te^{7t}$:

$$\frac{1}{(s+7)^2}$$

Figura 4.8: Transformada de Laplace $F(s)$ para a função $f(t) = te^{7t}$.

Para o vetor com as funções $f_1(t) = e^{-2t}$ e $f_2(t) = \text{sen}(3t)$, $f_3(t) = e^t \cos(t)$ e $f_4(t) = \cosh(7t)$. ✓LEMBRE-SE: para utilizar a função $f_4(t) = \cosh$ ou outras, você deve declarar no escopo inicial do programa, como $\text{cosh} = \text{sp.cosh}$.

```
funcoes = [exp(-2*t), sin(3*t), exp(1*t)*cos(1*t), coship(7*t)]
```

Figura 4.9: Exemplo: vetor de funções.

$$\left[\frac{1}{s+2}, \frac{3}{s^2+9}, \frac{s-1}{(s-1)^2+1}, \frac{s}{s^2-49} \right]$$

Figura 4.10: Exemplo: respostas para o vetor de funções.

4.2 Transformada Inversa de Laplace

Se na Transformada de Laplace saía-se de uma função $f(t)$ no domínio do tempo para uma função $F(s)$ no domínio s de Laplace, com a Transformada Inversa de Laplace (TIL) há o interesse no caminho inverso dessa análise em frequência. Utilizando a Transformada Inversa de Laplace é possível operar uma função $F(s)$ que esteja no domínio s e obter a sua resposta, $f(t)$, no domínio do tempo. Essa operação é formalizada na Figura a seguir.

$$\mathcal{L}^{-1}[F(s)] = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st} ds = f(t)u(t)$$

Figura 4.11: Definição da Transformada Inversa de Laplace.

A operação realizada aqui é particularmente útil, uma vez que a avaliação de uma resposta no domínio do tempo torna-se mais palpável na busca pelo entendimento da dinâmica do sistema estudado.

▷ Implementando a solução

Para começar, será importado o módulo SymPy de forma semelhante à seção anterior. Também é especificado o símbolo utilizado nas funções no domínio s .

```
# Importando o modulo SymPy que faz trata de Cálculo Avançado
import sympy

# Definir o símbolo das funções no domínio 's'
s = sympy.symbols('s')
t = sympy.symbols('t')
```

Figura 4.12: Importe do SymPy e declaração da simbologia.

O próximo passo é definir a função interna do *SymPy* que, ao ser chamada, faz o cálculo da Transformada Inversa de Lapalce. Em *noconds = True* é afirmado que no momento da declaração irá-se especificar apenas a função no domínio *s*.

```
# Definir a função interna SymPy que calcula da Transformada Inversa de Laplace
def Inversa(F):
    return sympy.inverse_laplace_transform(F, s, t, noconds = True)
```

Figura 4.13: Definição da função interna de cálculo da TIL.

Agora basta fornecer a função de transferência $F(s)$, no exemplo trabalhado na Figura a seguir foi definido $F(s) = \frac{7s-6}{s^2-s-6}$.

```
# Especificando da função no domínio 's' a ser calculada
F = (7*s-6)/(s**2-1*s-6)
```

Figura 4.14: Declaração da função $F(s)$ a ser calculada.

▷ Decomposição em frações parciais

Aqui trabalha-se com o quociente de funções polinomiais, e pode ser necessário e interessante fazer uso da decomposição em frações parciais, pois esta abordagem facilita os cálculos e diminui o tempo computacional necessário para se obter a resposta desejada.

Neste programa, será feito uso de um método interno para manipulação de polinômios *apart()* do *SymPy* que irá decompor a(s) função(ões) racional(is) especificada(s).

```
# Decompõe a função F, especificada anteriormente, em fracoes parciais
fracoes_parciais = F.apart(s)
```

Figura 4.15: Decomposição em frações parciais.

Por fim, ao ter a função $F(s)$ decomposta, utiliza-se a função interna definida *Inversa()* no início do programa passando $F(s)$ como argumento.

```
# Cálculo da TIL da função decomposta em frações parciais
Inversa(fracoes_parciais)
```

Figura 4.16: Cálculo da TIL de $F(s)$.

A resposta para a função $F(s) = \frac{7s-6}{s^2-s-6}$ pode ser vista a seguir.

✓LEMBRE-SE: a função $\theta(t)$ representa a função degrau de Heaviside, que essencialmente vale 0 para um valor de $t < 0$ e vale 1 para $t > a$, onde a assume valores positivos de tempo.

$$3e^{3t}\theta(t) + 4e^{-2t}\theta(t)$$

Figura 4.17: Resposta $f(t)$ para a $F(s)$.

4.3 Funções de Transferência

A Funções de Transferência (FT) podem ser vista como a caracterização, em frequência, de um dado sistema, através da relação entre a entrada $r(t)$ e a saída $c(t)$. Com uma Equação Diferencial de ordem n , linear e invariante no tempo (LCIT) da forma geral:

$$a_n \frac{d^n c(t)}{dt^n} + a_{n-1} \frac{d^{n-1} c(t)}{dt^{n-1}} + \dots + a_0 c(t) = b_m \frac{d^m r(t)}{dt^m} + b_{m-1} \frac{d^{m-1} r(t)}{dt^{m-1}} + \dots + b_0 r(t)$$

Figura 4.18: Forma geral de uma EDO no domínio do tempo.

Toma-se a Transformada de Laplace em ambos os lados da igualdade anterior e aplica-se condições iniciais nulas, isto é, $y(t) = 0$ e $\frac{dy(t)}{dt} = 0$, para uma função $y(t)$ genérica, chega-se em algo na forma:

$$(a_n s^n + a_{n-1} s^{n-1} + \dots + a_0)C(s) = (b_m s^m + b_{m-1} s^{m-1} + \dots + b_0)R(s)$$

Figura 4.19: EDO operada pela Transformada de Laplace.

Isolando os polinômios $C(s)$ (saída) e $R(s)$ (entrada), chega-se na forma $G(s) = \frac{C(s)}{R(s)}$, onde $G(s)$ representa a Função de Transferência com as características de entrada e saída que modelam o sistema estudado e devem ter condições iniciais nulas.

$$G(s) = \frac{C(s)}{R(s)} = \frac{(b_m s^m + b_{m-1} s^{m-1} + \dots + b_0)}{(a_n s^n + a_{n-1} s^{n-1} + \dots + a_0)}$$

Figura 4.20: Forma geral da Função de Transferência.

4.3.1 Resolução de EDO utilizando Transformada de Laplace

O próximo passo será abordar a resolução de Equações Diferenciais Ordinárias (EDO) utilizando a Transformada de Laplace de equações que estejam no domínio do tempo, aplicando-as condições iniciais nulas e ao final utilizando a Transformada Inversa de Laplace para transitar a resposta obtida no domínio s para o domínio do tempo, t .

► Implementando a solução

Fará-se uso do mesmo módulo *SymPy* utilizado em todas as soluções da seção 4.1, vista anteriormente. Será feito o *import* das funções a serem utilizadas e, posteriormente, dos símbolos a serem utilizados neste escopo e outras miscelâneas.

```
# Exibir tudo no mesmo documento
%matplotlib inline

import sympy as sympy
from sympy.integrals.transforms import inverse_laplace_transform
from sympy import *
c, C = symbols('c C', cls = Function)

# Print mais bonito no SymPy
init_printing(use_unicode=True)
```

Figura 4.21: Entrada e saída em uma FT.

Em seguida, serão definidas as condições iniciais nulas que serão aplicadas às equações diferenciais que estão apresentadas. Elas serão $y(t) = 0$ e $\frac{dy(t)}{dt} = 0$, pois serão restringidas às equações de ordem $n = 2$, veja na Figura a seguir:

```
# Condições Iniciais
y0 = 0
dy_0 = 0
```

Figura 4.22: Definição de condições iniciais nulas.

Agora é especialmente útil o conhecimento sobre a transformada de Laplace (L) em Equações Diferenciais do tipo da Figura 4.22. Para relembrar a você, segue uma forma geral da transformada de Laplace para funções de grau n :

$$L[f^{(n)}] = s^n F(s) - s^{n-1}f(0) - s^{n-2}f'(0) - s^{n-3}f''(0) - \dots - f^{(n-1)}(0)$$

Figura 4.23: Forma geral da Transformada de Laplace sobre equações diferenciais.

Nesse contexto, basta especificarmos a equações, do lado esquerdo direito e esquerdo da igualdade na Figura 4.23, no domínio s , para obtermos a função de transferência correspondente. Para o exemplo, tem-se:

$$(s^2C(s) - sy(0) - y'(0)) + 6(sC(s) - y(0)) + 2C(s) = 2(sR(s) - y(0)) + R(s)$$

```
# Defina aqui a equação diferencial no domínio 's'
# s^2C(s)-sy(0)-y'(0) + 6(sC(s)-y(0)) + 2C(s) = 2(sR(s)-y(0)) + R(s)
equacao_em_s = Eq(((s**2)*C(s)-s*y0-dy_0)+6*(s*C(s)-y0)+2*C(s), 2*(s-y0) + 1)
```

Figura 4.24: Declaração equação diferencial no domínio s .

Agora pode-se resolver a equação para a forma da função de transferência dada na Figura 4.24, $G(s) = \frac{C(s)}{R(s)}$.

```
# Resolve a equação para G(s) = C(s) / R(s)
G_s = solve(equacao_em_s, C(s))
```

Figura 4.25: Resolver para $G(s)$ s .

Com a resposta em frequência sendo a função de transferência $G(s) = \frac{2s+1}{s^2+6s+2}$ obtida, pode-se obter a sua resposta no domínio do tempo através da Transformada Inversa de Laplace, similar aquela vista na seção 4.1.1.

```
# Faz a Transformada Inversa de Laplace da função de transferência G(s)
solucao = inverse_laplace_transform(G_s[0], s, t)
```

Figura 4.26: Transformada Inversa de Laplace da função de transferência $G(s)$.

Afim de obtermos uma resposta que seja de fácil avaliação, utiliza-se o método *expand()* que fornece uma saída melhor formatada.

```
# Resposta formatada de maneira expandida e melhorada
solucao.expand()
```

Figura 4.27: Expansão da Transformada Inversa de Laplace da função de transferência $G(s)$.

Por fim, é obtida a resposta da função de transferência $G(s)$ no domínio do tempo.

$$-\frac{5\sqrt{7}e^{-3t} \sinh(\sqrt{7}t)\theta(t)}{7} + 2e^{-3t} \cosh(\sqrt{7}t)\theta(t)$$

Figura 4.28: Resposta no domínio do tempo para $G(s)$.

4.3.2 Polos e Zeros da Função de Transferência

A partir da função de transferência $G(s)$ da forma geral, já mostrada na seção anterior, mas resgatada aqui por conveniência, tem-se:

$$G(s) = \frac{C(s)}{R(s)} = \frac{(b_m s^m + b_{m-1} s^{m-1} + \dots + b_0)}{(a_n s^n + a_{n-1} s^{n-1} + \dots + a_0)}$$

Figura 4.29: Forma geral da Função de Transferência.

Logo, pode-se obter os chamados Polos e Zeros da função de transferência (FT) $G(s)$.

Os Zeros da função de transferência serão os valores de s no polinômio do numerador tais que zerem $G(s)$.

$$\begin{aligned} C_m(s) &= (b_m s^m + b_{m-1} s^{m-1} + \dots + b_0) = 0 \\ G(s) &= \frac{C(s)}{R(s)} = \frac{0}{(a_n s^n + a_{n-1} s^{n-1} + \dots + a_0)} = 0 \end{aligned}$$

Figura 4.30: Zeros da função de transferência.

Por outro lado, os Polos da FT serão os valores de s do denominador, que o zeram e consequentemente levam $G(s)$ à $+\infty$.

$$\begin{aligned} R_n(s) &= (a_n s^n + a_{n-1} s^{n-1} + \dots + a_0) = 0 \\ G(s) &= \frac{C(s)}{R(s)} = \frac{(b_m s^m + b_{m-1} s^{m-1} + \dots + b_0)}{0} = +\infty \end{aligned}$$

Figura 4.31: Polos da função de transferência.

▷ Implementando a solução

Para obtenção dos Zeros e Polos da FT irá-se utilizar o programa desenvolvido na seção anterior, com algumas pequenas mudanças. Dessa vez o cabeçalho deve incluir os métodos da biblioteca de Controle do *SymPy*, *TransferFunction()* e *pole_zero_plot*, esta última plota os Zeros e Polos de forma visual, utilizando a extensão *Matplotlib*.

```
# Exibir tudo no mesmo documento
%matplotlib inline

import sympy as sympy
from sympy.integrals.transforms import inverse_laplace_transform
from sympy import *
from sympy.physics.control.lti import TransferFunction
from sympy.physics.control.control_plots import pole_zero_plot
from sympy.abc import s
c, C = symbols('c C', cls = Function)

# Print mais bonito no SymPy
init_printing(use_unicode=True)
```

Figura 4.32: Cabeçalho para zeros e polos da FT.

Segue-se da mesma forma implementada anteriormente.

```
# Condições Iniciais
y0 = 0
dy_0 = 0
```

Figura 4.33: Definição de condições iniciais nulas.

```
# Defina aqui a equação diferencial no domínio 's'
# s^2C(s)-sy(0)-y'(0) + 6(sC(s)-y(0)) + 2C(s) = 2(sR(s)-y(0)) + R(s)
equacao_em_s = Eq(((s**2)*C(s)-s*y0-dy_0)+6*(s*C(s)-y0)+2*C(s), 2*(s-y0) + 1)
```

Figura 4.34: Declaração equação diferencial no domínio s .

```
# Resolve a equação para  $G(s) = C(s) / R(s)$ 
G_s = solve(equacao_em_s, C(s))
```

Figura 4.35: Resolver para $G(s)$.

Neste momento já tem-se a função de transferência na forma $G(s) = \frac{C(s)}{R(s)}$. Basta então obtermos separadamente o numerador $C_m(s)$ e o denominador $R_n(s)$.

```
num, den = fraction(G_s[0])
```

Figura 4.36: Obtenção do numerador e denominador da FT.

Por fim, utilizando o gerador de objetos do método *TransferFunction(numerador,denominador, variavelDeControle)*. Próximo, basta plotar os Zeros e Polos utilizando o método *pole_zero_plot(funcao_de_transferencia)*.

```
ft1 = TransferFunction(num, den, s)
pole_zero_plot(ft1)
```

Figura 4.37: Plotar os Zeros e Polos da FT.

Na Figura a seguir está ilustrado os Zeros e Polos da FT $G(s) = \frac{2s+1}{s^2+6s+2}$ que foram utilizados no exemplo anterior. Você pode verificar que para $C_m(s) = 2s + 1 = 0 \Rightarrow s = -\frac{1}{2}$ e $R_n(s) = s^2 + 6s + 2 = 0 \Rightarrow s_1 \simeq -0,35$ e $s_2 \simeq -5,6$.

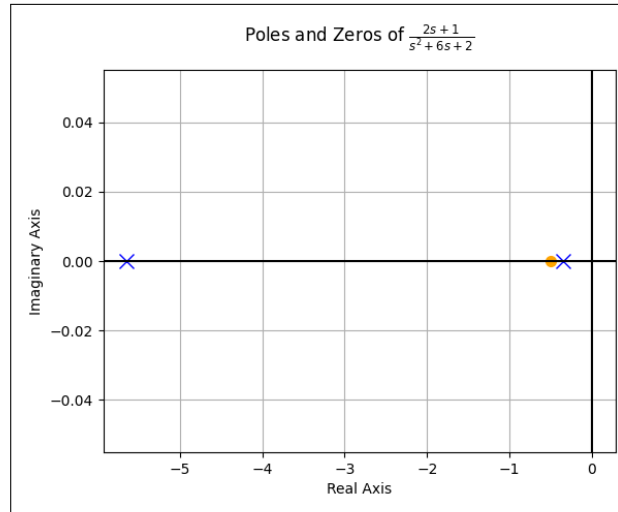


Figura 4.38: Gráfico de Zeros e Polos de $G(s)$.

4.4 Resposta em frequência

Um dos ápices da análise no Domínio da Frequência está na obtenção da resposta de um sistema às funções senoidais em estado estacionário de diferentes frequências. Essa resposta é conhecida como **Resposta em frequência**, e ela é composta de uma Magnitude $M(\omega)$ e de uma Fase $\phi(\omega)$, que por sua vez são funções de uma dada frequência ω ,

essa é resposta é particularmente útil para estudos que lidem com funções de transferência de sistema físicos, como o que será visto no Projeto Final deste curso.

$$M(\omega) = \frac{M_{\text{saída}}(\omega)}{M_{\text{entrada}}(\omega)}$$

$$\angle \varphi(\omega) = \varphi_{\text{saída}}(\omega) - \varphi_{\text{entrada}}(\omega)$$

$$M(\omega) \angle \varphi(\omega)$$

Figura 4.39: Definição Magnitude e Fase.

4.4.1 Diagrama de Bode

A análise da Resposta em Frequência é melhor fundamentada no **Diagrama de Bode**, que é a representação das curvas Magnitude-logarítmica (dada em decibéis, dB) e de Fase (dada em radianos), ambas funções da frequência ω visto anteriormente.

▷ Implementando a solução

Para dar início à implementação da Resposta em Frequência através do diagrama de Bode, serão retomados: o programa desenvolvido na seção 4.2.2 (Polos e Zeros); e os conceitos vistos até aqui, mas com algumas adaptações de cabeçalho.

```
%matplotlib inline
import sympy
from sympy import *
from sympy.abc import s
from sympy.physics.control.lti import TransferFunction
from sympy.physics.control.control_plots import bode_plot
c, C = symbols('c C', cls = Function)

# Print mais bonito no SymPy
init_printing(use_unicode=True)
```

Figura 4.40: Cabeçalho resposta em frequência.

O programa segue com a definição das condições iniciais, especificação da função de transferência $G(s)$ e obtenção de seu numerador e denominador.


```
# Condições Iniciais
y0 = 0
dy_0 = 0
```

Figura 4.41: Definição de condições iniciais nulas.

```
# Defina aqui a equação diferencial no domínio 's'
# s^2C(s)-sy(0)-y'(0) + 6(sC(s)-y(0)) + 2C(s) = 2(sR(s)-y(0)) + R(s)
equacao_em_s = Eq(((s**2)*C(s)-s*y0-dy_0)+6*(s*C(s)-y0)+2*C(s), 2*(s-y0) + 1)
```

Figura 4.42: Declaração equação diferencial no domínio s .

```
# Resolve a equação para G(s) = C(s) / R(s)
G_s = solve(equacao_em_s, C(s))
```

Figura 4.43: Resolver para $G(s)$.

```
num, den = fraction(G_s[0])
```

Figura 4.44: Obtenção do numerador e denominador da FT.

Agora, utilizando o método interno

`bode_plot(funcaTransf, ω InicialBase10, ω FinalBase10)`

Pode-se esboçar o diagrama de bode para a função $G(s) = \frac{2s+1}{s^2+6s+2}$.

O método anterior recebe como argumento a função de transferência que é objeto gerado a partir do método `TransferFunction()`. A faixa de valores da frequência (ω) utilizada neste exemplo é $10^{-1} \leq \omega \leq 10^2$.

```
ft1 = TransferFunction(num, den, s)
bode_plot(ft1, initial_exp=-1, final_exp=2)
```

Figura 4.45: Plotar o gráfico de Bode.

4.4.2 Módulo e Fase

Finalmente, é possível observar o diagrama de Bode, que essencialmente é a **Resposta em Frequência** do sistema representado pela função de transferência $G(s)$. Esse diagrama contém, em seus eixos verticais, os valores do Módulo ($M(\omega)$) e da Fase ($\phi(\omega)$) para valores iguais de frequência ω .

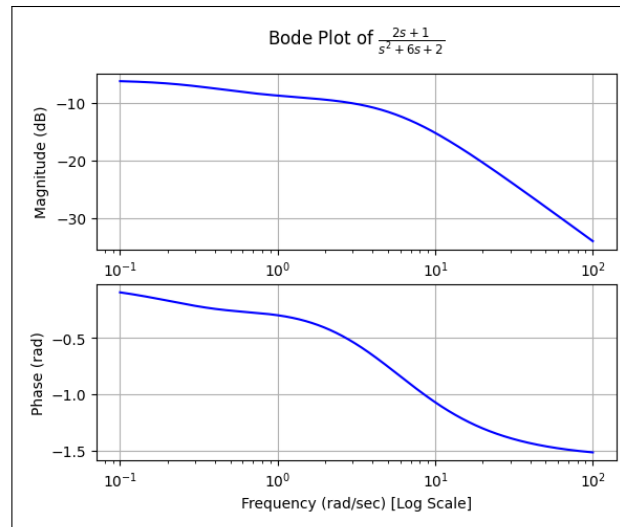


Figura 4.46: Diagrama de Bode para a função de transferência.

Capítulo 5

Estabilidade

5.1 Estabilidade BIBO

Conforme abordado na obra de (LATHI, 2007), para entender de forma intuitiva a estabilidade BIBO (bounded-input/bounded-output), é possível utilizar o exemplo de um cone. Ele pode permanecer indefinidamente sobre sua base circular, deitado sobre sua lateral ou de cabeça para baixo. Esses estados podem ser considerados estados de equilíbrio.

Entretanto, cada estado é muito diferente do outro em relação ao seu nível de estabilidade. Se o cone estiver sobre a sua base circular, e for perturbado, logo ele volta para a sua posição de equilíbrio original. Dessa forma, é dito que o cone está em equilíbrio estável. Quando o cone está deitado sobre sua lateral, com a menor perturbação ele não irá nem voltar nem se afastar do seu estado original de equilíbrio. Nesse caso, o cone está em equilíbrio neutro. E, quando o cone está de cabeça para baixo, a menor perturbação vai fazer com que ele se afaste cada vez mais do seu estado de equilíbrio original. Nesse caso, o cone está em equilíbrio instável.

Quando um sistema está no equilíbrio estável, uma pequena entrada resultará em uma pequena saída, sendo assim um sistema BIBO estável. Enquanto que, quando o sistema está em equilíbrio instável, uma pequena entrada resultará em uma saída ilimitada, ou seja, um sistema BIBO

instável.

Para um sistema LCIT:

$$y(t) = x(t) * h(t) \quad (5.1)$$

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t - \tau) d\tau \quad (5.2)$$

Pela propriedade associativa:

$$y(t) = \int_{-\infty}^{+\infty} h(\tau)x(t - \tau) d\tau \quad (5.3)$$

$$|y(t)| \leq \int_{-\infty}^{+\infty} |h(\tau)||x(t - \tau)| d\tau \quad (5.4)$$

Se $|x(\tau)|$ for limitado, logo:

$$\int_{-\infty}^{+\infty} |x(\tau)| d\tau = l \quad (5.5)$$

Logo, para $|x(t - \tau)|$, tem-se:

$$\int_{-\infty}^{+\infty} |x(t - \tau)| d\tau = k \quad (5.6)$$

Substituindo o k na equação 5.4 e extraíndo o mesmo da integral, tem-se:

$$|y(t)| \leq k \int_{-\infty}^{+\infty} |h(\tau)| d\tau \quad (5.7)$$

Para um sistema LCIT, se sua resposta $h(t)$ ao impulso for absolutamente integrável, o sistema é BIBO estável; caso contrário, é BIBO instável.

Para que a saída $y(t)$ seja limitada, toda a parte direita da desigualdade também precisa ser limitada. Como já é considerado k limitado, então $\int_{-\infty}^{+\infty} |h(\tau)| d\tau$ também precisa ser limitado, ou seja, para que um sistema

seja BIBO estável:

$$\int_{-\infty}^{+\infty} |h(\tau)| d\tau < \infty \quad (5.8)$$

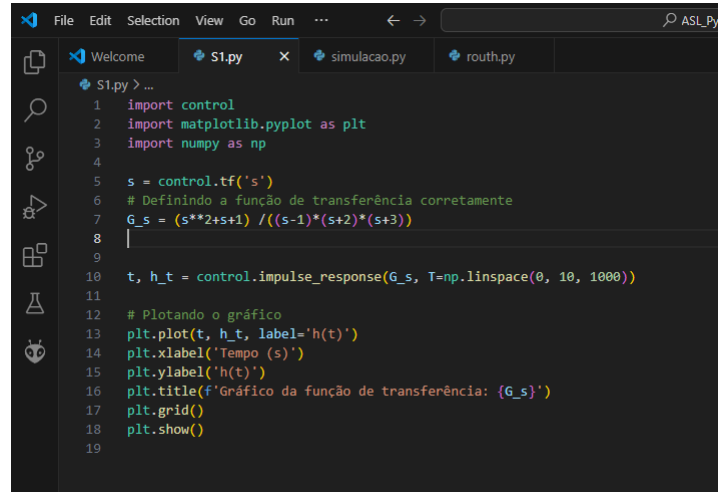


Figura 5.1: Código Para Gráfico de $h(t)$

A Figura acima é um exemplo de implementação em python de uma função de transferência qualquer, para determinar a estabilidade do sistema por meio do gráfico de $h(t)$.

5.2 Critério de Estabilidade Routh-Hurwitz

”O critério de Routh-Hurwitz é um critério necessário e suficiente para a estabilidade de sistemas lineares” (DORF; BISHOP, 2018). Ele permite avaliar a estabilidade do sistema a partir de sua equação característica. Para que o sistema seja estável, todos os coeficientes do polinômio devem ter o mesmo sinal e, para que isso ocorra, todas as raízes devem estar no semiplano s da esquerda do plano complexo.

Para determinar a estabilidade do sistema, basta calcular as raízes da equação característica. Porém, quando a equação característica possui muitas raízes diferentes, torna-se muito difícil determinar todas as raízes

sem a ajuda de um computador. Desta forma, se utiliza o Critério de Estabilidade Routh-Hurwitz.

$$\begin{array}{c|cccc}
 s^n & a_0 & a_2 & a_4 & \cdots \\
 s^{n-1} & a_1 & a_3 & a_5 & \cdots \\
 s^{n-2} & b_1 & b_2 & b_3 & \cdots \\
 s^{n-3} & c_1 & c_2 & c_3 & \cdots \\
 s^{n-4} & d_1 & d_2 & d_3 & \cdots \\
 \vdots & \vdots & \vdots & \vdots & \\
 s^2 & e_1 & e_2 & & \\
 s^1 & f_1 & & & \\
 s^0 & g_1 & & &
 \end{array}
 \quad
 \begin{array}{l}
 b_1 = \frac{a_1 a_2 - a_0 a_3}{a_1} = -\frac{1}{a_1} \begin{vmatrix} a_0 & a_2 \\ a_1 & a_3 \end{vmatrix} \\
 b_2 = \frac{a_1 a_4 - a_0 a_5}{a_1} = -\frac{1}{a_1} \begin{vmatrix} a_0 & a_4 \\ a_1 & a_5 \end{vmatrix} \\
 b_3 = \frac{a_1 a_6 - a_0 a_7}{a_1} = -\frac{1}{a_1} \begin{vmatrix} a_0 & a_6 \\ a_1 & a_7 \end{vmatrix} \\
 c_1 = -\frac{1}{b_1} \begin{vmatrix} a_1 & a_3 \\ b_1 & b_2 \end{vmatrix}
 \end{array}$$

Exemplo de tabela de Routh-Hurwitz.

Como mostrado na imagem acima, primeiro é feita uma coluna de s_n até s_0 . Nas linhas s_n e s_{n-1} , são adicionados os coeficientes de s da equação característica da função de transferência, e os valores abaixo da linha de s_{n-1} são os valores que serão determinados. Para determinar o b_1 , é calculado o determinante da matriz formada pelos coeficientes a_0 , a_1 , a_2 e a_3 , e este valor é multiplicado por $-1/(\text{valor inferior esquerdo da matriz})$.

$$b_1 = \frac{a_1 a_2 - a_0 a_3}{a_1} = -\frac{1}{a_1} \begin{vmatrix} a_0 & a_2 \\ a_1 & a_3 \end{vmatrix}$$

O cálculo do b_2 é feito de forma semelhante, a diferença é que a segunda coluna da matriz não será formada pela coluna dos coeficientes a_2 , a_3 , mas sim pela próxima coluna, que será formada por a_4 , a_5 .

$$b_2 = \frac{a_1 a_4 - a_0 a_5}{a_1} = -\frac{1}{a_1} \begin{vmatrix} a_0 & a_4 \\ a_1 & a_5 \end{vmatrix}$$

P Para o cálculo do c_1 , o procedimento é semelhante ao do b_1 , com a diferença que agora serão usados utilizados os coeficientes da próxima linha, ou seja, a matriz será formada pela coluna a_1 , b_1 e pela coluna a_3 , b_2 . E a matriz será multiplicada por $-\frac{1}{b_1}$.

$$c_1 = -\frac{1}{b_1} \begin{vmatrix} a_1 & a_3 \\ b_1 & b_2 \end{vmatrix}$$

Para encontrar os outros coeficientes o procedimento é o mesmo.

Exemplo 1

Será aplicado o critério de Routh-Hurwitz para verificação da estabilidade de um sistema com a equação característica a seguir:

$$s^4 + 2s^3 + 3s^2 + 4s + 5 = 0$$

Primeiro, é colocado os dois s de maior ordem na primeira coluna. Em seguida, é adicionado o coeficiente de s^4 (1) na primeira linha da primeira coluna arranjo, e o coeficiente de s^3 (2) na segunda linha da segunda coluna da tabela. Por fim, então são colocados os coeficientes restantes nas linhas da tabela de forma alternada.

$$\begin{array}{c|ccc} s^4 & 1 & 3 & 5 \\ s^3 & 2 & 4 & 0 \end{array}$$

Tabela 5.1: Arranjo de Routh

Após este passo, é preciso adicionar na tabela as linhas de s^2 até s^0 .

$$\begin{array}{c|ccc} s^4 & 1 & 3 & 5 \\ s^3 & 2 & 4 & 0 \\ s^2 & b_1 & b_2 & b_3 \\ s^1 & c_1 & c_2 & c_3 \\ s^0 & d_1 & d_2 & d_3 \end{array}$$

Tabela 5.2: Arranjo de Routh

Para encontrar o b_1 , calcule o produto do determinante da matriz formada por 1, 2, 3, 4 e multiplique por $-\frac{1}{2}$.

$$b_1 = -\frac{1}{2} \begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix} \Rightarrow b_1 = 1$$

Para encontrar b_2 , o cálculo é semelhante, porém a segunda coluna dessa matriz será a próxima coluna do arranjo de Routh, ou seja, a coluna será formada por 5 e 0;

$$b_2 = -\frac{1}{2} \begin{vmatrix} 1 & 5 \\ 2 & 0 \end{vmatrix} \Rightarrow b_2 = 5$$

Para o cálculo do b_3 , seria necessário pular mais uma coluna, porém como não há mais colunas no arranjo de Routh, tem-se $b_3 = 0$.

O cálculo do c_1 é semelhante ao cálculo de b_1 , mas é feito, seguindo-se à próxima linha do arranjo de Routh.

$$c_1 = -\frac{1}{b_1} \begin{vmatrix} 2 & 4 \\ b_1 & b_2 \end{vmatrix}$$

$$c_1 = -\frac{1}{1} \begin{vmatrix} 2 & 4 \\ 1 & 5 \end{vmatrix} \Rightarrow c_1 = -6$$

Para o cálculo do c_2 , seria necessário pular uma coluna do arranjo de Routh, mas como a segunda coluna da matriz seria formada por 0 e 0, o determinante é zero, logo $c_2 = 0$.

Para o cálculo do d_1 , assim como o c_1 , é necessário pular uma linha do arranjo de Routh.

$$d_1 = -\frac{1}{c_1} \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix}$$

$$d_1 = \frac{1}{6} \left| \begin{array}{cc} 1 & 5 \\ -6 & 0 \end{array} \right| \Rightarrow d_1 = 5$$

Agora, substitui-se os valores no arranjo original:

$$\begin{array}{c|ccc} s^4 & 1 & 3 & 5 \\ s^3 & 2 & 4 & 0 \\ s^2 & 1 & 5 & 0 \\ s^1 & -6 & 0 & 0 \\ s^0 & 5 & 0 & 0 \end{array}$$

Tabela 5.3: Arranjo de Routh com valores

Há duas mudanças de sinal no arranjo, ou seja, há duas raízes reais com partes reais positivas. Logo, o sistema é instável.

Exemplo 2

Considere o seguinte polinômio:

$$s^5 + 2s^4 + 2s^3 + 4s^2 + 11s + 10 = 0$$

Montando o arranjo de Routh:

$$\begin{array}{c|ccc} s^5 & 1 & 2 & 11 \\ s^4 & 2 & 4 & 10 \end{array}$$

Tabela 5.4: Arranjo de Routh do Exemplo 2

$$\begin{array}{c|ccc} s^5 & 1 & 2 & 11 \\ s^4 & 2 & 4 & 10 \\ s^3 & b_1 & b_2 & b_3 \\ s^2 & c_1 & c_2 & c_3 \\ s^1 & d_1 & d_2 & d_3 \\ s^0 & e_1 & e_2 & e_3 \end{array}$$

Tabela 5.5: Arranjo de Routh do Exemplo 2

Para encontrar o b_1

$$b_1 = -\frac{1}{2} \begin{vmatrix} 1 & 2 \\ 2 & 4 \end{vmatrix} \Rightarrow$$

$$b_1 = 0, \quad b_1 = \epsilon \text{ (logo este } \epsilon \text{ será explicado)}$$

Para encontrar o b_2 , pula-se uma coluna, resolve-se a matriz e multiplica-se o determinante por $-\frac{1}{2}$:

$$b_2 = \frac{1}{6} \begin{vmatrix} 1 & 11 \\ 2 & 10 \end{vmatrix} \Rightarrow b_2 = 6$$

Para encontrar o c_1 , pular uma linha e montar a matriz.

$$c_1 = -\frac{1}{b_1} \begin{vmatrix} 2 & 4 \\ b_1 & b_2 \end{vmatrix}$$

Como o valor de b_1 é 0, ele não pode ser colocado como denominador em uma fração, logo o valor de b_1 será substituído por ϵ , em que $\epsilon = \lim_{x \rightarrow 0^+} x$.

$$c_1 = -\frac{1}{\epsilon} \begin{bmatrix} 2 & 4 \\ \epsilon & 6 \end{bmatrix} \Rightarrow c_1 = \frac{4\epsilon - 12}{\epsilon} \approx -\frac{12}{\epsilon}$$

Agora o mesmo procedimento para encontrar o c_2 .

$$c_2 = -\frac{1}{\epsilon} \begin{bmatrix} 2 & 10 \\ \epsilon & 0 \end{bmatrix} \Rightarrow c_2 = 10$$

O cálculo para d_1 e e_1 será feito o mesmo método.

$$d_1 = -\frac{1}{c_1} \begin{bmatrix} b_1 & b_2 \\ c_1 & c_2 \end{bmatrix}$$

$$d_1 = \frac{1}{\frac{12}{\epsilon}} \begin{bmatrix} \epsilon & 6 \\ -\frac{12}{\epsilon} & 10 \end{bmatrix} \Rightarrow d_1 \approx 6$$

Cálculo do e_1 :

$$e_1 = -\frac{1}{d_1} \begin{bmatrix} c_1 & c_2 \\ d_1 & 0 \end{bmatrix}$$

$$e_1 = -\frac{1}{6} \begin{bmatrix} -\frac{12}{\epsilon} & 10 \\ 6 & 0 \end{bmatrix} \Rightarrow e_1 = 10$$

Agora, os valores calculados são substituídos e é verificada a estabilidade do sistema.

$$\begin{array}{c|ccc} s^5 & 1 & 2 & 11 \\ s^4 & 2 & 4 & 10 \\ s^3 & \epsilon & 6 & \\ s^2 & c_1 & 10 & \\ s^1 & d_1 & & \\ s^0 & 10 & & \end{array}$$

Há duas mudanças de sinal no arranjo, ou seja, há duas raízes reais com partes reais positivas, logo o sistema é instável. Para ser estável, todos os membros da coluna 2 deveriam ser positivos.

Exemplo de implementação do Critério de Routh-Hurwitz:

```

1
2
3 from tbcontrol.symbolic import routh
4 import sympy
5 import matplotlib.pyplot as plt
6
7 s= sympy.Symbol('s')
8
9 #FT De Malha Aberta
10 V_s=1;
11 P_s=1000/((s+2)*(s+3)*(s+5));
12 G_s=sympy.expand(V_s*P_s); #Calcula os termos do polinomio P(s)
13 print("Malha aberta G(s)={}".format(G_s))
14
15
16 #FT De Malha Fechada
17 H_s=1;
18 G1_s=sympy.cancel((G_s)/(1+G_s*H_s))
19 print("Malha fechada G1(s)={}".format(G1_s))
20 num_G1_s,den_G1_s= sympy.fraction(G1_s)
21 print("\n Tabela de Routh:")
22
23 den_G1_s = sympy.poly(den_G1_s)
24
25 print("\nTabela de Routh")
26 tabela=routh(den_G1_s)
27 print(tabela)
28
29
30 poles = sympy.solve(den_G1_s)
31 print("Os polos do sistema são{}".format(poles))

```

Figura 5.2: Código Parte 1

```

27 print(tabela)
28
29
30 poles = sympy.solve(den_G1_s)
31 print("Os polos do sistema são{}".format(poles))
32
33
34 plt.figure()
35 for pole in poles:
36     plt.plot(sympy.re(pole), sympy.im(pole),'rx')
37
38     plt.axhline(0, color='black', lw=0.5)
39     plt.axvline(0, color='black', lw=0.5)
40
41
42     plt.grid(True, which='both', linestyle='--', lw=0.5)
43
44
45
46 plt.xlabel('Parte Real')
47 plt.ylabel('Parte Imaginária')
48 plt.title('Polos no Plano Complexo')
49
50 plt.show()
51

```

Figura 5.3: Código Parte 2

Capítulo 6

Projeto de Sistemas

6.1 Diagrama de Blocos

6.1.1 Introdução ao Diagrama de Blocos

O diagrama de blocos é uma ferramenta gráfica amplamente utilizada na análise de sistemas de controle e automação. Ele representa graficamente as relações matemáticas entre os componentes de um sistema. Cada bloco no diagrama simboliza uma função ou operação matemática, como uma transformação ou uma função de transferência, que atua sobre o sinal de entrada para gerar um sinal de saída.

Os diagramas de blocos são usados para representar sistemas lineares, contínuos e de tempo invariantes. Através deles, é possível visualizar o fluxo de sinais e entender a interação entre diferentes subsistemas.

6.1.2 Elementos de um Diagrama de Blocos

Um diagrama de blocos é composto por elementos fundamentais:

- **Blocos:** Cada bloco representa uma operação matemática, geralmente uma função de transferência, que transforma um sinal de entrada em um sinal de saída.
- **Setas (Linhas de Conexão):** Representam o fluxo de sinais entre os blocos.

- **Somadores:** Pontos onde dois ou mais sinais são adicionados ou subtraídos.
- **Ramos (ou divisores de sinal):** Locais onde um sinal é dividido e enviado para dois ou mais destinos.

6.1.3 Exemplo de Diagrama de Blocos

Considere um sistema com duas funções de transferência $G_1(s) = \frac{1}{s+1}$ e $G_2(s) = \frac{2}{s+3}$. O sinal de entrada $R(s)$ passa pelo bloco $G_1(s)$, cujo resultado é somado a um sinal de retroalimentação, que passa por outro bloco $G_2(s)$. O diagrama de blocos correspondente seria o seguinte:

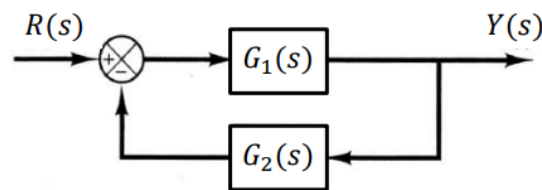


Figura 6.1: Exemplo diagrama de blocos

6.2 Redução de Diagrama de Blocos

6.2.1 Introdução à Redução de Diagramas de Blocos

A redução de um diagrama de blocos é o processo de simplificação de um diagrama complexo em um sistema equivalente mais simples, mantendo as mesmas relações de entrada e saída. Isso é essencial na análise de sistemas para entender o comportamento global sem a complexidade de lidar com todos os componentes individualmente.

A redução de diagramas de blocos é baseada em algumas regras fundamentais, como a combinação de blocos em série, blocos em paralelo, e o deslocamento de pontos de somadores e ramos de sinal.

6.2.2 Regras de Redução de Diagrama de Blocos

- **Blocos em Série:** Se dois blocos $G_1(s)$ e $G_2(s)$ estão conectados em série (um após o outro), eles podem ser combinados em um único bloco cuja função de transferência é o produto das funções de transferência individuais:

$$G(s) = G_1(s) \times G_2(s)$$

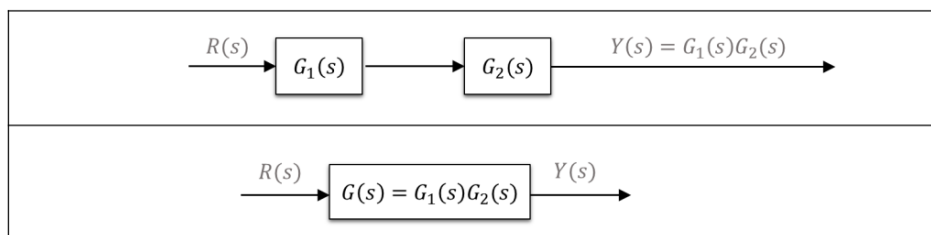


Figura 6.2: Diagrama em série

- **Blocos em Paralelo:** Se dois blocos $G_1(s)$, $G_2(s)$ e $G_3(s)$ estão em paralelo e suas saídas são somadas, eles podem ser combinados em um único bloco cuja função de transferência é a soma das funções de transferência individuais:

$$G(s) = G_1(s) + G_2(s) + G_3(s)$$

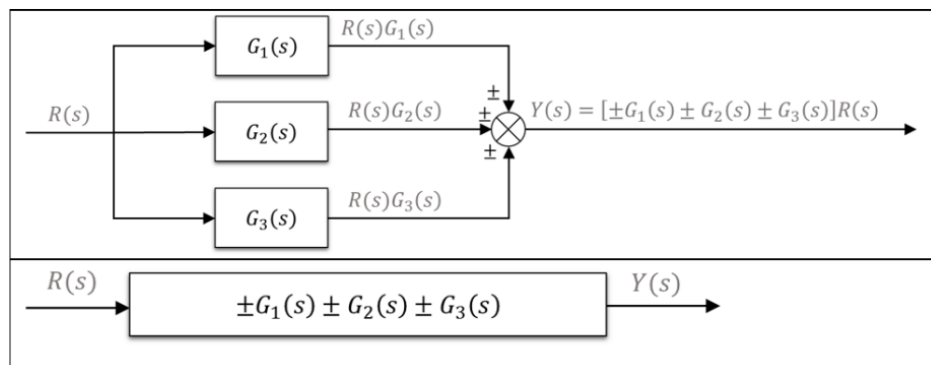


Figura 6.3: Diagrama em paralelo

- **Eliminação de Loops de Realimentação:** Para um loop de

realimentação com um bloco $G(s)$ e um ganho de realimentação $H(s)$, o diagrama de blocos pode ser reduzido para um único bloco com a função de transferência:

$$G_{fe}(s) = \frac{G(s)}{1 + G(s)H(s)}$$

para realimentação negativa.

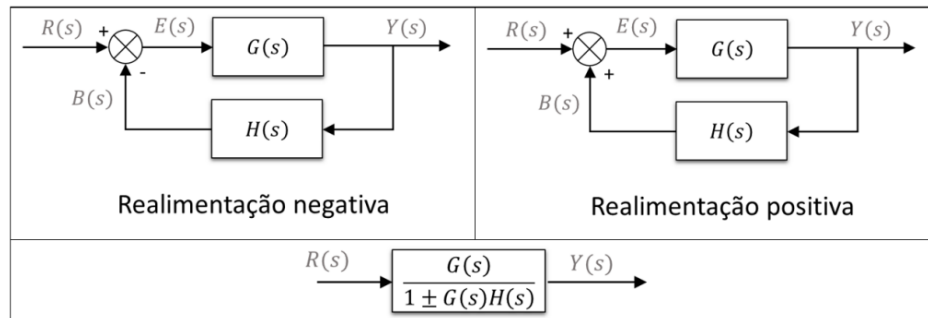


Figura 6.4: Sistema realimentado

6.2.3 Regras de Desentrelaçamento

Além das regras de combinação de blocos em série, paralelo e eliminação de loops, o desentrelaçamento é um processo utilizado para reorganizar diagramas de blocos, movendo somadores, ramos de sinal e blocos através de junções, para facilitar a redução. As principais regras de desentrelaçamento são:

- **Mover um Somador para Frente:** Se um somador precede um bloco de transferência $G(s)$, o somador pode ser movido para depois do bloco, mas cada termo que entra no somador deve passar pelo mesmo bloco. Ou seja:
- **Mover um Somador para Trás:** Um somador que segue um bloco pode ser movido para antes do bloco, desde que os sinais somados estejam divididos pelo bloco de função de transferência:

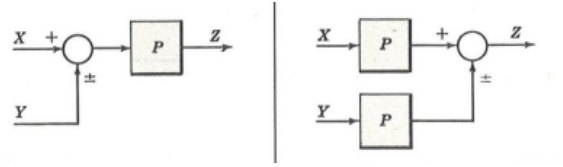


Figura 6.5: Somador para frente

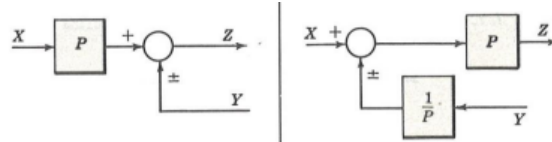


Figura 6.6: Somador para trás

- **Mover um Ramo para Frente:** Um ponto de ramificação pode ser movido para uma posição mais à frente no diagrama, desde que o ramo afetado preserve o mesmo sinal, e todos os blocos intermediários entre o ponto original e o novo ponto de ramificação sejam aplicados tanto ao sinal principal quanto aos ramos resultantes.

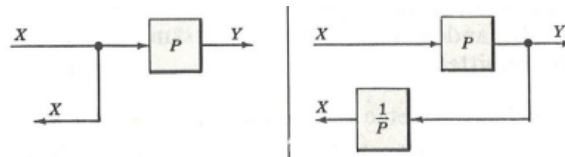


Figura 6.7: Ramificação para Frente

- **Mover um Ramo Antes de um Bloco:** Um ponto de ramificação pode ser movido para antes de um bloco de função de transferência, desde que cada ramo passe pelo bloco após a divisão do sinal:

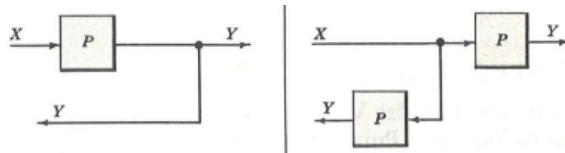


Figura 6.8: Ramificação para trás

6.2.4 Conclusão

A redução de diagramas de blocos constitui uma ferramenta essencial na análise de sistemas de controle, proporcionando uma abordagem

sistemática para a simplificação e compreensão de sistemas complexos. Essa técnica permite que engenheiros otimizem o projeto e a análise de sistemas, facilitando a modelagem e o estudo de seu comportamento dinâmico.

As propriedades e metodologias relacionadas à diagramas de blocos e redução de diagramas de blocos são apresentadas de forma detalhada nas obras de (OGATA, 2010) e (DORF; BISHOP, 2018), que serviram como referência para esta e a seção anterior. Nessas publicações, encontram-se ainda exemplos práticos e exercícios que contribuem para uma melhor compreensão do tema.

6.2.5

Os diagramas de blocos são amplamente utilizados para a representação gráfica da dinâmica dos sistemas de controle, sendo úteis para mostrar as interações entre variáveis de controle e de entrada. No entanto, à medida que os sistemas se tornam mais complexos, simplificar esses diagramas pode ser trabalhoso e difícil. Uma alternativa proposta por Mason é o método conhecido como *diagrama de fluxo de sinais*, que utiliza segmentos de arcos e apresenta a vantagem de permitir o cálculo do ganho de um sistema por meio de uma fórmula específica, sem a necessidade de manipular ou reduzir o diagrama.

O gráfico de fluxo de sinais representa um conjunto de equações algébricas lineares simultâneas, onde os nós são conectados por ramos orientados. Cada nó representa uma variável do sistema, e cada ramo indica a relação entre uma variável de entrada e outra de saída, de forma semelhante ao que ocorre em diagramas de blocos. O fluxo de sinais ocorre em uma única direção, sendo indicado por setas colocadas nos ramos, e o fator de multiplicação é indicado ao longo desses ramos.

Nos diagramas de fluxo de sinais, a relação entre as variáveis é indicada ao longo dos arcos, que transmitem o sinal de um nó de entrada para outro

de saída de forma unidirecional. A soma dos sinais que entram em um nó deve ser igual à variável representada por esse nó. Um *percurso* é definido como uma sequência de ramos conectados de um nó a outro. Um *laço* é um percurso fechado que retorna ao nó de origem sem passar por nenhum nó mais de uma vez. Quando dois laços não compartilham nenhum nó em comum, eles são chamados de *disjuntos*. Laços que compartilham nós são considerados *não disjuntos*.

Embora o diagrama de fluxo de sinais e o diagrama de blocos contenham essencialmente as mesmas informações, o primeiro oferece a vantagem de utilizar uma fórmula de ganho, conhecida como a *Regra de Mason*, para calcular a função de transferência sem a necessidade de redução gráfica.

A obtenção de funções de transferência a partir de diagramas de fluxo de sinal é abordada de forma detalhada na obra de (OGATA, 2010) e (DORF; BISHOP, 2018), a qual serviu de referência para a elaboração dessa seção. É recomendada a consulta a esse material para um aprofundamento no tema, incluindo exemplos práticos e exercícios resolvidos.

6.2.6 Regra de Mason

A Regra de Mason é utilizada para determinar a função de transferência de um sistema, que descreve a relação matemática entre a entrada e a saída. A aplicação da regra envolve identificar os *caminhos diretos* entre a entrada e a saída, bem como os *laços* no sistema. Os principais componentes da Regra de Mason incluem:

- **Caminhos diretos:** São percursos que conectam diretamente a variável de entrada à variável de saída, sem passar por laços que retornam ao ponto de origem. Esses caminhos representam as rotas pelas quais o sinal viaja de um ponto de entrada para um ponto de saída sem voltar sobre si mesmo.

- **Laços (loops):** São ciclos fechados dentro do sistema, em que o sinal retorna ao ponto de partida após passar por várias variáveis intermediárias. Os laços podem influenciar o comportamento do sistema, desviando parte do sinal e reinserindo-o em diferentes pontos do processo.
 - **Laços disjuntos:** Não compartilham variáveis com outros laços.
 - **Laços não disjuntos:** Compartilham pelo menos uma variável com outros laços.
- **Delta (Δ):** Representa o determinante do sistema, usado para calcular a função de transferência por meio da Regra de Mason.

O determinante de um sistema pode ser descrito por:

$$\begin{aligned} \Delta = & 1 - (\text{soma de todos os laços individuais}) \\ & + (\text{soma dos produtos de todos os pares de laços disjuntos}) \\ & - (\text{soma dos produtos de todos os triplos de laços disjuntos}) + \dots \end{aligned} \quad (6.1)$$

Esse determinante mede a interação entre os laços presentes no sistema.

Assim, o diagrama de fluxo de sinais, quando utilizado com a Regra de Mason, torna-se uma ferramenta poderosa para a análise de sistemas de controle, permitindo calcular a função de transferência sem a necessidade de simplificação direta do diagrama.

$$T = \frac{\sum_k P_k \Delta_k}{\Delta} \quad (6.2)$$

- P_k : ganho do caminho ou transmitância do caminho direto de ordem k ;
- Δ : determinante do diagrama;
- Δ_k : cofator do percurso P_k .

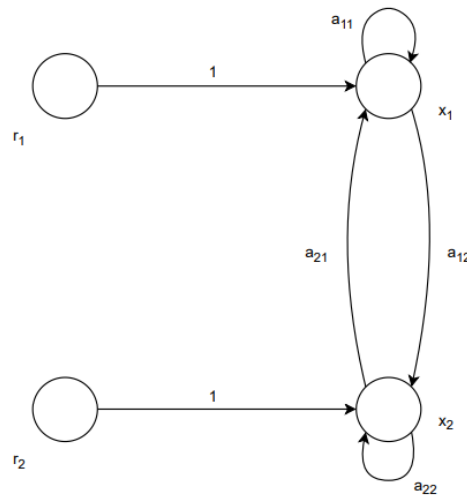


Figura 6.9: Diagrama de fluxo de sinal de duas equações algébricas

Para resolver o diagrama da Figura 6.9, é necessário encontrar os laços. Para isso, será enumerado a partir do zero para que o algoritmo desenvolvido em Python nos informe os laços.

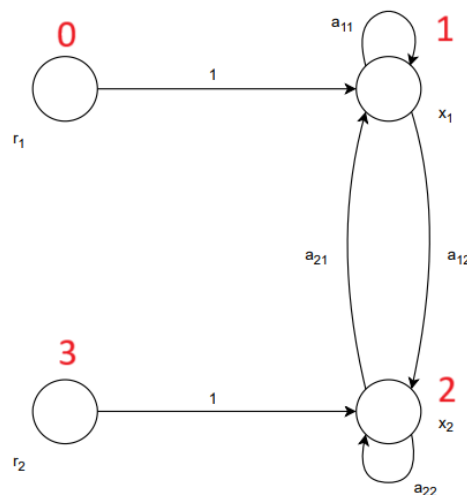


Figura 6.10: Diagrama de fluxo de sinal

Feito isso, basta colocar os nós e em quais outros nós eles estão conectados. Também é preciso informar onde há a entrada no diagrama, isso será feito entre a linha 40 e 51 da Figura 6.16.

```

41     # Definindo a lista de adjacências (exemplo dado)
42     adjlist[0] = [1]
43     adjlist[1] = [1, 2]
44     adjlist[2] = [1, 2]
45     adjlist[3] = [2]
46
47     # Chama a função para o nó inicial
48     f(0, 0)
49     f(3, 0)

```

Figura 6.11: Modelagem diagrama de fluxo

Após informar todos os nós, entradas e conexões, pode-se executar o algoritmo e ele nos informará os seguintes laços: 1, 1 2 e 2, que são os laços desse exemplo explicitados pela Figura 6.12, sendo a cor azul para os loops e a cor vermelha a aresta.

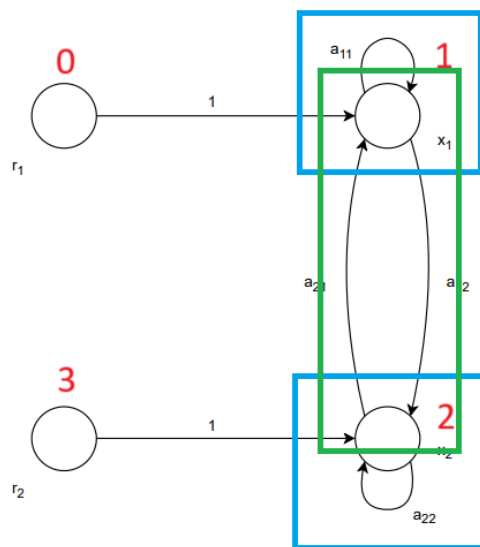


Figura 6.12: Laços explicitados diagrama de fluxo de sinais

Na Figura 6.13, tem-se outro exemplo de diagrama de fluxo de sinais com mais nós.

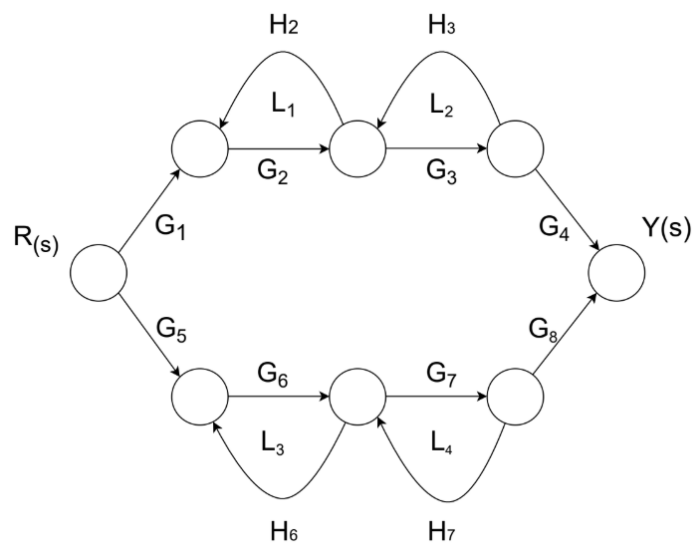


Figura 6.13: Diagrama com dois percursos

Observando este diagrama e seguindo a mesma lógica de enumeração a partir do zero, é tomado o primeiro nó da esquerda para a direita como o 0 e a ordem seguindo o fluxo superior até completar a volta.

```

41     # Definindo a lista de adjacências (exemplo dado)
42     adjlist[0] = [1, 7]
43     adjlist[1] = [2]
44     adjlist[2] = [1, 3]
45     adjlist[3] = [2, 4]
46     adjlist[4] = []
47     adjlist[5] = [4, 6]
48     adjlist[6] = [5, 7]
49     adjlist[7] = [6]
50     # Chama a função para o nó inicial
51     f(0, 0)
52     #f(3, 0)

```

Figura 6.14: Modelagem Figura 6.13

Feita esta modelagem no código e deixando a entrada apenas como o nó inicial, ele nos dará as arestas: 1 2, 2 3, 6 5 e 7 6, que são todos os laços do diagrama de fluxo de sinais explicitados na Figura 6.15.

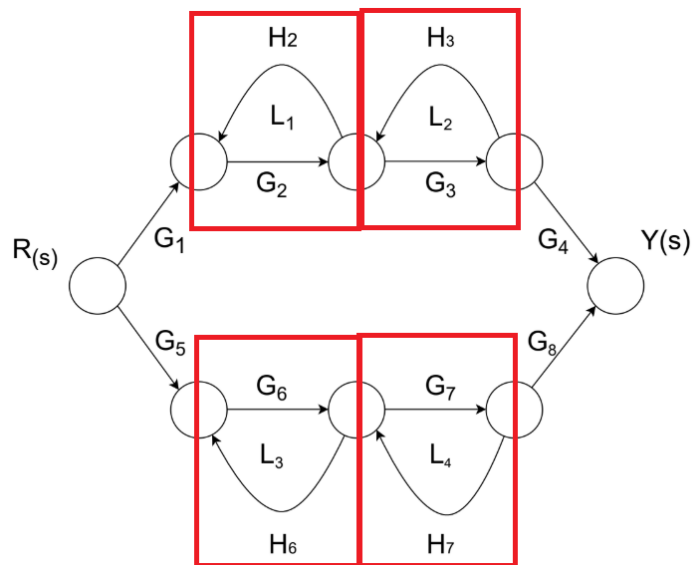


Figura 6.15: Arestas explicitadas

A seguir, tem-se o código completo em Python que também estará disponível no GitHub.

[https://github.com/ismascarenhas/
Diagrama-de-Fluxo-de-Sinais.git](https://github.com/ismascarenhas/Diagrama-de-Fluxo-de-Sinais.git)


```
1  from collections import defaultdict
2
3  # Conjuntos para armazenar ciclos encontrados
4  resp = set()
5  respCerto = []
6
7  # Lista de adjacências para o grafo
8  adjlist = defaultdict(list)
9
10 # Variáveis auxiliares
11 pos = [-1] * 100
12 vis = [-1] * 100
13
14 # Lista temporária para armazenar o caminho atual
15 v = []
16
17 # Função recursiva para percorrer o grafo e encontrar ciclos
18 def f(at, cnt):
19     v.append(at)
20     pos[at] = cnt
21     vis[at] = 1
22
23     for to in adjlist[at]:
24         if vis[to] != -1: # Se o nó já foi visitado
25             ciclo = []
26             for i in range(pos[to], len(v)):
27                 ciclo.append(v[i])
28             cicloOrdenado = sorted(ciclo)
29
30             if tuple(cicloOrdenado) not in resp:
31                 resp.add(tuple(cicloOrdenado))
32                 respCerto.append(ciclo)
33         else:
34             f(to, cnt + 1)
35
36     v.pop() # Remove o último elemento após a recursão
37     vis[at] = -1 # Marca o nó como não visitado
38
39 # Função principal
40 if __name__ == "__main__":
41     # Definindo a lista de adjacências (exemplo dado)
42     adjlist[0] = [1]
43     adjlist[1] = [1, 2]
44     adjlist[2] = [1, 2]
45     adjlist[3] = [2]
46
47     # Chama a função para o nó inicial
48     f(0, 0)
49     f(3, 0)
50     # Imprime todos os ciclos encontrados
51     for ciclo in respCerto:
52         print(" ".join(map(str, ciclo)))
53
```

Figura 6.16: Algoritmo completo em Python

Capítulo 7

Modelagem de Sistemas

Nesta seção serão tratados exemplos práticos e simples de modelagem de sistemas mecânicos e eletroeletrônicos. Sendo os escolhidos para tal, uma suspensão de automóvel básica e dois exemplos de circuitos integradores.

Todos os circuitos e dispositivos modelados nesta seção são adequadamente tratados e abordados nas obras de (GARCIA, 2022; OGATA, 2010). Outros exemplos de sistemas mecânicos e eletroeletrônicos são também tratados nas demais obras da seção "Referências" deste material.

7.1 Suspensão automobilística

Tem-se que uma suspensão de automóvel simplificada normalmente é disposta na seguinte configuração: uma mola, de coeficiente elástico K ; um amortecedor, de coeficiente de amortecimento B . Como ilustrado na Figura 7.1.

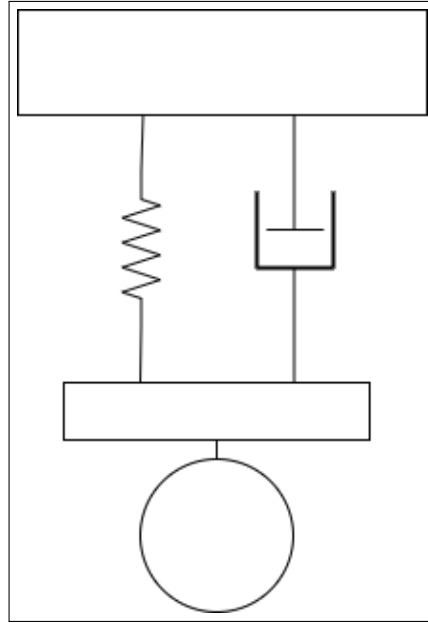


Figura 7.1: Diagrama da suspensão.

Sucessivamente, é elaborado o diagrama de forças:

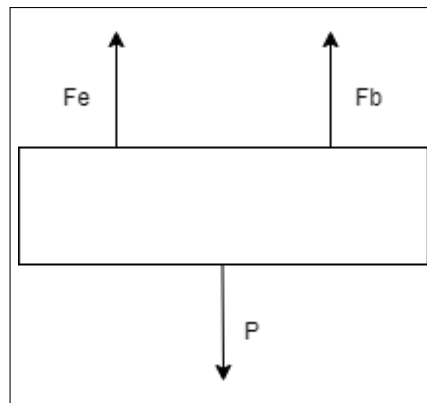


Figura 7.2: Diagrama de forças sobre o chassi do automóvel.

Tem-se que as forças dispostas pelo diagrama, sendo: F_r a Força Resultante, P a força peso, F_b a força do amortecedor e F_e sendo a força elástica da mola; se relacionam da seguinte forma pela segunda Lei de Newton:

$$Fr = \sum_{i=1}^n Fi = Fb + Fe - P \quad (7.1)$$

$$Fr + P = Fb + Fe \quad (7.2)$$

$$my''(t) + mg = B(u'(t) - y'(t)) + K(u(t) - y(t) + \delta) \quad (7.3)$$

Neste exercício de modelagem, será assumido que a força peso é compensada pela compressão estática da mola. Essa força de compressão estática, denotada por F_k , é definida como:

$$F_k = K\delta \quad (7.4)$$

Onde δ representa a deformação estática da mola causada pelo peso do sistema em equilíbrio, e é dado por:

$$\delta = \frac{mg}{K} \quad (7.5)$$

Dessa forma, o sistema já parte de um estado onde a contração da mola, causada pela força peso, foi absorvida, e a análise dinâmica considera apenas as variações em torno dessa nova posição de equilíbrio. Portanto, pode-se dar sequência à modelagem do nosso problema:

$$\begin{aligned} my''(t) &= B(u'(t) - y'(t)) + K(u(t) - y(t)) \\ my''(t) &= Bu'(t) - By'(t) + Ku(t) - Ky(t) \end{aligned}$$

$$my''(t) + By'(t) + Ky(t) = Bu'(t) + Ku(t) \quad (7.6)$$

Utilizando o operador "D", tal que: $D = \frac{d}{dt}$ e $D^{-1} = \int dt$.

$$y(t)(mD^2 + BD + K) = u(t)(BD + K)$$

$$\frac{y(t)}{u(t)} = \frac{BD + K}{mD^2 + BD + K} \quad (7.7)$$

E por fim, aplicando a transformada de Laplace em ambos os lados

da equação, obtém-se a função de transferência do sistema:

$$\frac{Y(s)}{U(s)} = H(s) = \frac{Bs + K}{ms^2 + Bs + K} \quad (7.8)$$

Com a função de transferência, é possível a obtenção da resposta do sistema para diferentes entradas. Eis um código que permite a obtenção da resposta ao degrau do amortecedor em questão, e sua respectiva saída, nas figuras 7.3 e 7.4.

```
1 import numpy as np
2 import control as ctrl
3 import matplotlib.pyplot as plt
4
5 m = 1
6 k = 1
7 b = 1
8
9 num = [b, k]
10 den = [m, b, k]
11
12 sys = ctrl.TransferFunction(num, den)
13
14 t, y = ctrl.step_response(sys)
15
16 plt.plot(t, y)
17 plt.xlabel('Tempo (s)')
18 plt.ylabel('Resposta')
19 plt.title('Amortecedor de automóvel - Resp ao degrau')
20 plt.grid()
21 plt.show()
22
```

Figura 7.3: Código da resposta ao degrau - Amortecedor

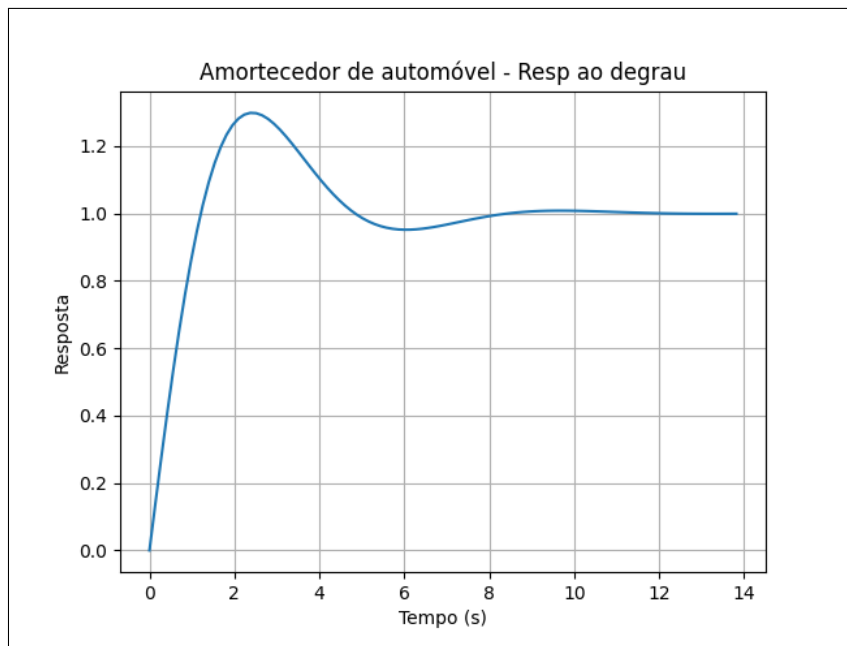


Figura 7.4: Resposta ao degrau - Suspensão

Ressaltando, o gráfico de resposta desconsidera a diferença de altura do chassi do automóvel em relação a roda. Dessa forma, a resposta real do sistema seria a altura do chassi com relação ao nível inicial do solo, mais a resposta apontada no gráfico.

7.2 Circuito integrador

Esta subseção tratará da modelagem de sistemas eletroeletrônicos simples. Aqui serão modelados circuitos integradores compostos por componentes passivos e ativos.

7.2.1 Circuito integrador - Componentes passivos

Um circuito integrador simples, fazendo uso de apenas capacitores e resistores, pode ser construído na seguinte configuração abaixo:

Começa-se modelando o circuito a partir de seu ganho, tal que:

$$G = \frac{V_{\text{out}}}{V_{\text{in}}} \quad (7.9)$$

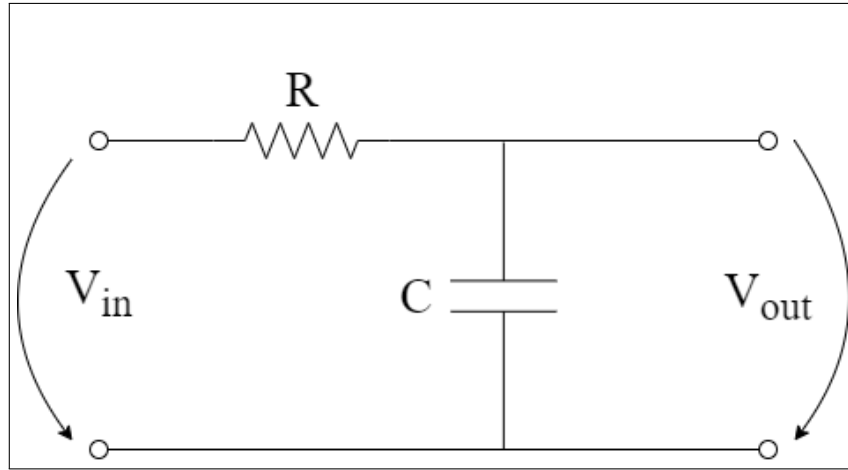


Figura 7.5: Circuito Integrador RC

Pela Lei de Kirchoff das Tensões, a tensão na malha de saída é igual a tensão do capacitor, e que a tensão na malha de entrada é igual a tensão do resistor somada a do capacitor, portanto:

$$G = \frac{V_C}{V_R + V_C} \quad (7.10)$$

A tensão de um resistor é calculada pela Lei de Ohm, já a tensão no capacitor é correspondente ao inverso da capacitância multiplicada pela integral da corrente em um dado intervalo de tempo, de forma que:

$$G = \frac{\frac{1}{C} \int_0^t i(\tau) d\tau}{Ri(t) + \frac{1}{C} \int_0^t i(\tau) d\tau} \quad (7.11)$$

Aplicando a transformada de Laplace, é obtida uma equação mais simples:

$$G(s) = \frac{\frac{1}{Cs} I(s)}{(R + \frac{1}{Cs}) I(s)} = \frac{\frac{1}{Cs}}{R + \frac{1}{Cs}} = \frac{1}{RCs + 1} \quad (7.12)$$

Sabe-se que $s = j\omega$; portanto, para valores de ω muito elevados, pode-se aproximar a equação de maneira que:

$$G(s) = \frac{1}{RCs} \quad (7.13)$$

Equacionando a função de transferência com a equação original do ganho, tem-se que:

$$G(s) = \frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = \frac{1}{RCs} \rightarrow V_{\text{out}}(s) = \frac{1}{RCs} V_{\text{in}}(s) \quad (7.14)$$

Por fim, aplicando a transformada inversa de Laplace na equação obtida, tem-se que a saída é correspondente ao inverso do produto da resistência R com a capacitância C , multiplicado pela integral da tensão de entrada num dado intervalo t de tempo.

$$v_{\text{out}}(t) = \frac{1}{RC} \int_0^t v_{\text{in}}(\tau) d\tau \quad (7.15)$$

7.2.2 Circuito integrador - Componentes ativos

Antes da elaboração do circuito integrador, primeiramente, será calculado o ganho da configuração inversora do Amplificador Operacional, configuração essa diagramada abaixo:

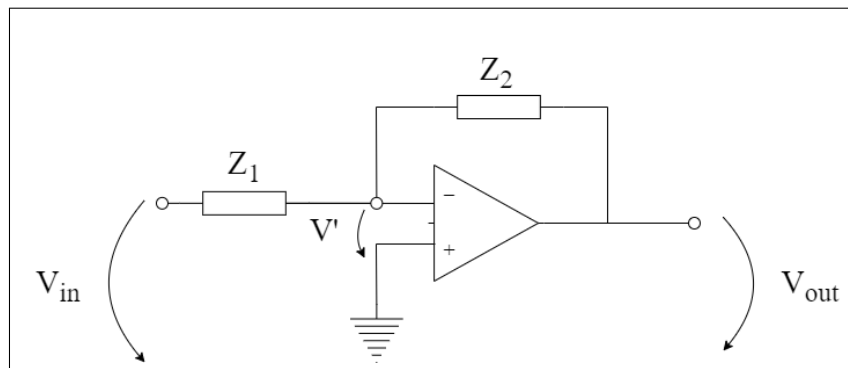


Figura 7.6: Configuração Inversora Amp-Op

Para modelar um circuito com Amplificador Operacional, é necessário ter algumas coisas em mente antes de prosseguir. Primeiramente, irá-se supor um "Amp-Op" ideal, de forma que sua impedância de entrada tenderá ao infinito. Segundamente, é importante conhecermos a equação de ganho do "Amp-Op", que é:

$$v_{\text{out}}(t) = K(v_2 - v_1) = -K(v_1 - v_2) \quad (7.16)$$

Tal equação representa a tensão de saída, aterrada, em função das tensões de entrada dos terminais do "Amp-Op", sendo v_1 a tensão do terminal inversor e v_2 a tensão do terminal não-inversor.

Com isso, pode-se começar a modelar o circuito da figura 103, começa-se definindo as correntes que passam pelas impedâncias Z_1 e Z_2 :

$$i_1 = \frac{v_{\text{in}} - v'}{Z_1}; i_2 = \frac{v' - v_{\text{out}}}{Z_2} \quad (7.17)$$

Será feita a escolha de um "Amp-Op" ideal, portanto a corrente que passa pelo "Amp-Op" beira a zero. Dessa forma, a corrente que passa pela impedância Z_1 é igual a que atravessa a impedância Z_2 , pode-se assim equacioná-las:

$$\frac{v_{\text{in}} - v'}{Z_1} = \frac{v' - v_{\text{out}}}{Z_2} \quad (7.18)$$

Agora, indo para a equação de ganho do Amplificador, sabe-se que a tensão de saída e a do terminal não-inversor estão ambas aterradas de forma que:

$$v_{\text{out}} = -K(v_1 - v_2) = -K(v_1 - 0) \rightarrow v_{\text{out}} = -Kv' \quad (7.19)$$

Sendo K um valor bastante alto, geralmente cerca de 10^5 até 10^6 , é possível aproximar a tensão v' para um valor próximo de zero, assim:

$$\begin{aligned} \frac{v_{\text{in}} - v'}{Z_1} &= \frac{v' - v_{\text{out}}}{Z_2} \\ \frac{v_{\text{in}}}{Z_1} &= \frac{-v_{\text{out}}}{Z_2} \end{aligned}$$

Por fim, será obtido o ganho da configuração inversora do "Amp-Op" para duas impedâncias Z_1 e Z_2 quaisquer, tal que:

$$\frac{v_{\text{out}}}{v_{\text{in}}} = G = -\frac{Z_2}{Z_1} \quad (7.20)$$

Com isso, basta substituir as impedâncias Z_1 e Z_2 da configuração inversora por um resistor de resistência R e um capacitor de capacitância C , conforme a figura abaixo, e calcular a função de transferência.

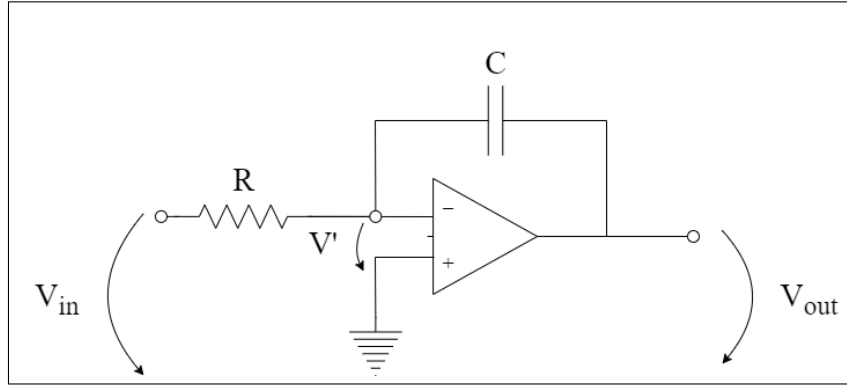


Figura 7.7: Circuito Integrador - Amp-Op

A impedância equivalente a Z_1 seria R , e a impedância equivalente a Z_2 seria $\frac{1}{Cs}$, dessa forma, pode-se obter a função de transferência:

$$G = -\frac{Z_2}{Z_1} = -\frac{\frac{1}{Cs}}{R} = -\frac{1}{RCs} \quad (7.21)$$

Agora basta terminar de manipular a equação e aplicar a transformada inversa de Laplace, assim obtendo uma nova equação bastante similar a obtida na análise do circuito RC.

$$G = \frac{v_{\text{out}}}{v_{\text{in}}} = -\frac{1}{RCs}$$

$$v_{\text{out}} = -\frac{1}{RCs} v_{\text{in}}$$

$$v_{\text{out}}(t) = -\frac{1}{RC} \int_0^t v_{\text{in}}(\tau) d\tau \quad (7.22)$$

Referências

DORF, Richard C.; BISHOP, Robert H. **Sistemas de Controle Modernos**. 13^a edição. [S.l.]: LTC, 2018. ISBN 978-85-2163-512-3.

GARCIA, Claudio. **Modelagem e Simulação de Processos Industriais e Sistemas Eletromecânicos**. 3^a edição. [S.l.]: EdUSP, 2022. ISBN 978-65-5785-067-1.

LATHI, B.P. **Sinais e Sistemas Lineares**. 2^a edição. [S.l.]: Bookman, 2007. ISBN 978-85-7780-391-0.

MEZA, Magno E.M. **Sistemas de Controle com Python: Resposta em Frequência**. 1^a edição. [S.l.]: Blucher, 2024. ISBN 978-85-2122-112-8.

NISE, Norman S. **Engenharia de Sistemas de Controle**. 8^a edição. [S.l.]: LTC, 2023. ISBN 978-85-2163-827-8.

OGATA, Katsuhiko. **Engenharia de Controle Moderno**. 5^a edição. [S.l.]: Pearson Universidades, 2010. ISBN 978-85-7605-810-6.

SANDROCK, Carl. **Dynamics and Control with Jupyter Notebooks**. [S.l.: s.n.], 2018.

Índice

Diagrama de Blocos	Propriedades de Sistemas, 16, 17,
Diagramação, 61	19, 20
Redução esquemática, 62, 64	Transformada de Laplace, 33, 37
Diagrama de Fluxo de Sinal	
Diagramação, 66	
Regra de Mason, 67	
Estabilidade, 51	
Critério BIBO, 51	
Critério Routh-Hurwitz, 53	
Funções de Transferência, 39, 62	
Zeros e Polos, 43	
Modelagem	
Sistemas Eletrônicos, 78	
Sistemas Mecânicos, 74	
Variáveis de Estado, 30	
Resposta de Sistemas	
Resposta em Frequência, 47, 50	
Resposta no Domínio do Tempo,	
22, 26, 28	
Sinais, 8	
Propriedades de Sinais, 9, 10	
Sistemas, 16	

ACOMPANHE NOSSAS REDES

INSTAGRAM



@petemcufg

SITE



pet.emc.ufg.br

CANAL DE AVISOS



WhatsApp