CONTROLADOR DIRETO RST ADAPTATIVO COM IDENTIFICAÇÃO POR MÍNIMOS QUADRADOS RECURSIVO

Robert Kawe Linhares Nobre

Universidade Federal do Ceará
Centro de Tecnologia
Departamento de Engenharia Elétrica

Resumo: O projeto de um controlador RST combinado com identificação de sistemas por Mínimos Quadrados (MMQ) de forma direta oferece flexibilidade no ajuste de desempenho em sistemas dinâmicos, enquanto o método MMQ direto permite estimar os parâmetros do modelo de forma eficiente, sem necessidade de etapas intermediárias. A abordagem proposta visa melhorar o desempenho em tempo real de sistemas de controle, garantindo robustez e adaptabilidade.

Introdução

O controle de sistemas dinâmicos com incertezas paramétricas e perturbações externas representa um desafio significativo na área de controle automático [2]. Como ilustrado na Figura 1, o sistema em estudo considera tanto perturbações de entrada quanto de saída, exigindo uma estratégia de controle robusta. O controlador RST se apresenta como uma solução adequada para este cenário, pois sua estrutura polinomial permite o ajuste independente do desempenho transitório e em regime permanente [4]. Entretanto, a eficácia deste controlador depende fundamentalmente da qualidade da identificação do processo, o que motiva o uso de técnicas recursivas de identificação por Mínimos Quadrados (MMQ).

A abordagem de identificação recursiva por MMQ tem sido amplamente adotada em sistemas de controle adaptativo devido à sua eficiência computacional e capacidade de atualização em tempo real [5]. Como mostra a Figura 1, o estimador opera em paralelo com o controlador RST, fornecendo parâmetros atualizados do processo (a, b) que são utilizados para o cálculo da lei de controle (u). A implementação direta do MMQ recursivo, sem etapas intermediárias de transformação, reduz erros de modelagem e simplifica a implementação prática [6]. Esta característica é particularmente vantajosa em sistemas onde a dinâmica pode variar com o tempo ou apresentar não linearidades.

Este trabalho propõe uma metodologia integrada que combina o controlador RST com identificação recursiva por MMQ direto, conforme representado no diagrama de blocos da Figura 1. A abordagem visa melhorar o desempenho do sistema em malha fechada, mesmo na presença de perturbações e incertezas paramétricas. O artigo está organizado da seguinte forma: a seção Métodos Empregados detalha o projeto do controlador RST e o algoritmo de identificação MMQ recursivo; a seção Resultados apresenta análises comparativas por meio de simulações numéricas; e a seção Conclusão discute as contribuições do trabalho e perspectivas futuras.

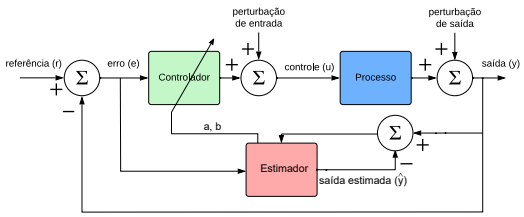

Figura 1: Diagrama de blocos do sistema de controle com identificação MMQ recursivo.

Métodos Empregados

Modelagem Matemática e Projeto do Controlador RST

A modelagem matemática do sistema é realizada por meio de identificação utilizando o método dos Mínimos Quadrados Recursivo (MMQ). Sabendo que a planta apresenta uma não linearidade quadrática, então, aplica-se o método de identificação baseado em dados experimentais, resultando em um modelo discreto de primeira

ordem na forma de uma equação com parâmetros a serem estimados. A estrutura do modelo adotado é do tipo $y(k) = -a_1 y(k-1) + b_0 u(k-1)$, que permite um projeto do controlador sem erro em regime permanente, mesmo diante da presença de não-linearidades moderadas.

O projeto do controlador RST é conduzido pelo método direto, utilizando o modelo identificado para calcular os polinômios $R(z)$, $S(z)$ e $T(z)$. O polinômio $S(z^{-1})$ é responsável pela realimentação, $R(z^{-1})$ ajusta a resposta dinâmica do sistema, e $T(z^{-1})$ garante o seguimento adequado da referência, como pode ser observado no diagrama de blocos da figura 2 Com essa abordagem, busca-se garantir estabilidade, rejeição de perturbações e desempenho dinâmico satisfatório, mesmo com as limitações impostas pela não linearidade da planta.

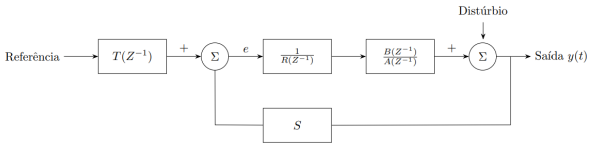

Figura 2: Diagrama de blocos do sistema de controle RST.

Para garantir erro nulo em regime permanente e ausência de sobressinal, adotou-se uma estratégia de controle adaptativo direto RST, inspirada no controle PI 2DOF (Two Degrees of Freedom). Como destacado por [1], a estrutura RST oferece maior flexibilidade que controladores convencionais, permitindo o ajuste independente dos polinômios R, S, e T para rastreamento de referência e rejeição de perturbações. Inicialmente, fixamos os valores de R e definimos o T como um ganho para facilitar a modelagem, enquanto os valores de S serão determinados através do MMQ.

Adota-se como planta o modelo discreto de primeira ordem com não linearidade quadrática descrito pela equação $y(k) = 0.08y(k-1)^2 + 0.56y(k-1) + 0.1u(k-1)$. O sistema de controle em malha fechada será projetado para que não apresente sobressinal a partir do segundo degrau e para que siga, com erro nulo em regime permanente, a sequência de referências:

$$\begin{gathered} 1.00, \text{ para } 0 < t < 1250\ s; \\ 0.50, \text{ para } 1250 < t < 2500\ s; \\ 0.75, \text{ para } 2500 < t < 3750\ s \text{ e} \\ 1.15, \text{ para } 3750 < t < 5000\ s \end{gathered}$$

A implementação do controlador será realizada utilizando um código em *Python*, o qual resolverá as equações necessárias para o cálculo dos polinômios $R(z)$, $S(z)$ e $T(z)$ e aplicará a lógica de controle sobre o modelo identificado.

Posteriormente, será considerada a introdução de um ganho $K_c \in \mathbb{R}_{>0}$, inserido nos polinômios $R(z)$, $S(z)$ e $T(z)$, que poderá ser livremente ajustado pelo projetista visando otimizar o desempenho do sistema. Além disso, será introduzida uma distúrbio no sistema em $t = 4000\ s$, para avaliar o desempenho do controlador obtido nos itens (a) — comportamento frente à referência e (c) — impacto da introdução do ganho K_c.

Identificação Recursiva

A atualização dos parâmetros θ via Mínimos Quadrados Recursivo (MMQ) implementa o algoritmo clássico de identificação adaptativa, conforme descrito em [5]. Este método possibilita a estimação online dos parâmetros do modelo, ajustando-os progressivamente a cada nova amostra, o que é essencial para sistemas dinâmicos sujeitos a variações.

O algoritmo segue os seguintes passos:

1. **Cálculo do ganho adaptativo**:

$$k_{\text{gain}} = \frac{P\varphi}{\lambda + \varphi^T P \varphi} \tag{1}$$

onde:

- P é a matriz de covariância associada à incerteza nos parâmetros;
- φ é o vetor de regressores na amostra atual;
- $\lambda \in (0, 1]$ é o fator de esquecimento.

Esse ganho determina a intensidade da correção aplicada ao vetor de parâmetros, ponderando a confiança nas informações disponíveis.

2. **Atualização do vetor de parâmetros**:

$$\theta = \theta + k_{\text{gain}} \cdot \text{erro}[t] \tag{2}$$

onde $\text{erro}[t] = y(t) - \hat{y}(t)$ representa o erro de predição entre a saída real e a saída prevista do modelo.

3. **Atualização da matriz de covariância**:

$$P = \frac{P - k_{\text{gain}} \varphi^T P}{\lambda} \tag{3}$$

Esta atualização ajusta a matriz de covariância P para refletir a nova incerteza após a correção dos parâmetros, enquanto o fator de esquecimento λ reduz gradualmente a influência de dados antigos.

Definição da Matriz de Regressores e Parâmetros

Para a identificação do modelo e o projeto do controlador RST, foi realizada uma simplificação estrutural que facilitou a estimação dos parâmetros e a síntese do controlador. Primeiramente, fixou-se o polinômio $R(z)$ como:

$$R(z) = 1 - z^{-1} \tag{4}$$

Além disso, o polinômio $T(z)$ foi tratado como um ganho escalar, o que reduz o grau de $T(z)$, mas mantém a capacidade de ajustar o seguimento da referência.

Dessa forma, a identificação do modelo foi orientada para a estimação dos parâmetros do polinômio $S(z)$, que atua na relação entre a entrada e a saída do sistema.

Considerando que a planta siga o modelo de primeira ordem $G(z^{-1})$:

$$G(z^{-1}) = \frac{bz^{-1}}{1 - az^{-1}} \tag{5}$$

A relação entre a entrada de controle $r(t)$ e a saída do sistema $y(t)$ considerando a função de transferência de malha aberta:

$$\frac{y(k)}{r(k)} = \frac{bTz^{-1}}{(1 - z^{-1})(1 - az^{-1})} \tag{6}$$

Escrevendo isso na forma de equações de diferenças e passando para a forma matricial, temos:

$$y(k) = \begin{bmatrix} r(k-1) \\ r(k-2) \\ -y(k-1) \\ y(k-2) \end{bmatrix} \begin{bmatrix} bTs_0 & bTs_1 & (1+a) & a \end{bmatrix} \tag{7}$$

- T é o ganho associado ao polinômio $T(z)$;
- b é o parâmetro de ganho da planta;
- s_0 e s_1 são os coeficientes do polinômio $S(z)$;
- a é o coeficiente associado à dinâmica do modelo.

Além disso, para a função de transferência de malha fechada do sistema obtemos:

$$\frac{y(k)}{r(k)} = \frac{BT}{AR + BS} \tag{8}$$

- B polinômio numerador da planta;
- A polinômio denominador da planta;

Rescrevendo isso, temos:

$$\frac{y(k)}{r(k)} = \frac{bTz^{-1}}{1 + (bs_0 - (1+a)z^{-1} + az^{-2}} \tag{9}$$

consideramos,

- $\theta 1 = s_o b$
- $\theta 2 = s_1 b$
- $\theta 3 = (1 + a)$
- $\theta 4 = a$

Sabendo que $\frac{y(k)}{r(k)}$ deve convergir para 1, podemos escrever tal expressão:

$$bT = 1 + (\theta 1 + \theta 2) - (\theta 3 + \theta 4) \tag{10}$$

Agora, para a determinação de T como um ganho temos que ele deve ser igual ao valor de $\frac{P(z^{-1})}{B(z^{-1})}$ aplicado ao teorema do valor final. Assim temos que T irá convergir para:

$$T = \frac{A(1)R(1) + B(1)S(1)}{B(1)} = s_0 + s_1 \tag{11}$$

Então, a partir das equações 10 e 11, podemos determinar os valores s_0 e s_1 que estão nos conjuntos de parâmetros na equação 7.

$$s_0 = \theta 1/Tb \tag{12}$$

$$s_1 = \theta 2/Tb \tag{13}$$

$$T = s_0 + s_1 \tag{14}$$

Dessa forma, podemos fazer identificação dos parâmetros θ por meio do algoritmo de Mínimos Quadrados Recursivo (MMQ) permite inferir tanto os ganhos do polinômio $S(z)$ e T quanto os coeficientes associados à dinâmica da planta. Esta parametrização também evidencia a relação direta entre os parâmetros estimados e as características estruturais do controlador RST.

Lei de controle

A lei de controle RST segue a formulação de [3]:

$$u(k)R(z^{-1}) = r(k)T(z^{-1}) - y(k)S(z^{-1}) \tag{15}$$

Resultados

Item 01: Controle MRAC RST direto

Ao implementar a teoria em um código na linguagem *python*, inicialmente consideramos o fatores de esquecimento ($\lambda 1$ e $\lambda 2$) iguais a um, porém tivemos um erro de resposta para valores de $r(k) > 1$, como é mostrado na

figura 3. Então, tivemos que ajustar o fator de esquecimento para obter a resposta desejeda de controle, obtido na figura 4 e os parâmetros determinados para o RST na figura 5 através dos valores $\lambda 1 = 1.1$ e $\lambda 2 = 0.75$:

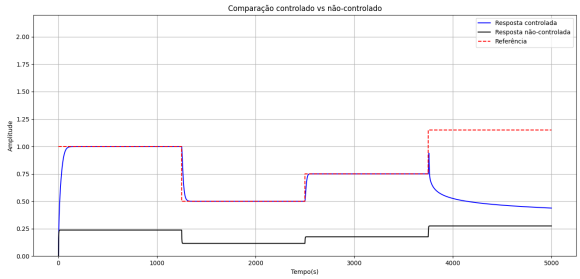

Figura 3: Resposta do controle sem ajuste de λ

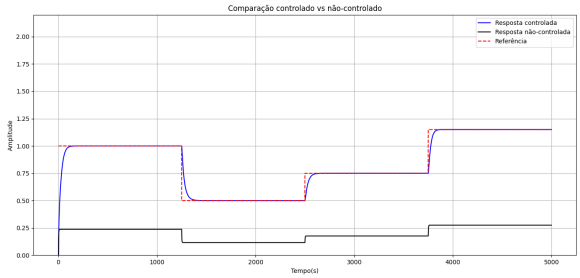

Figura 4: Resposta do controle com ajuste de λ

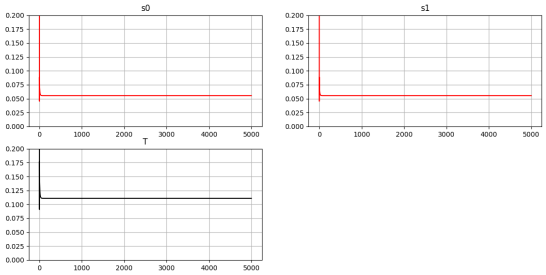

Figura 5: Parâmetros do controle RST

Item 02: Inclusão de ganho ajustável nos parâmetros de R, S e T

Ao colocar um ganho Kc em R, S e T percebemos que a resposta do sistema variou em amplitude e um pouco em tempo de assentamento. Nas figuras 6 e 7 temos, respectivamente, a respota do sistema para Kc=0.9 e Kc=1.2.

Porém, foi possível perceber que para apenas um ganho ajustável em T é possível obter o mesmo resultado de

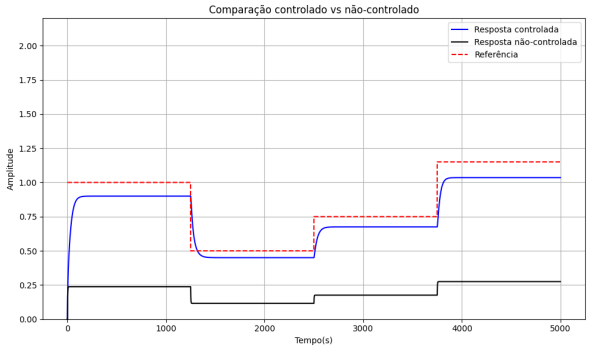

Figura 6: Resposta para Kc=0.9 em R, S e T

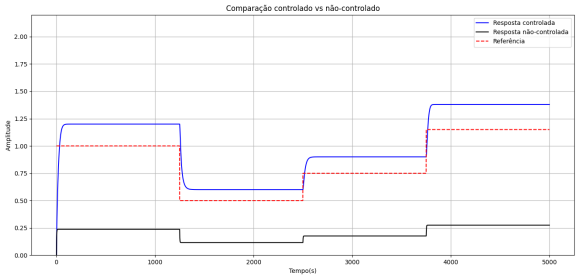

Figura 7: Resposta para Kc=1.2 em R, S e T

ajuste. E quando, variado apenas o R e S nada acontece, pois o T está em função de S, como está na equação 14, logo em toma esse ganho para sí também.

Item 03: Inclusão de pertubação na resposta do sistema

Adicionando no sinal de saída uma pertubação em degrau em t=4000s, obtemos a seguinte resposta na figura 8. Sendo possível notar que o sistema consegue identificar a pertubação no sinal de saída como uma pertubação no sinal de referência.

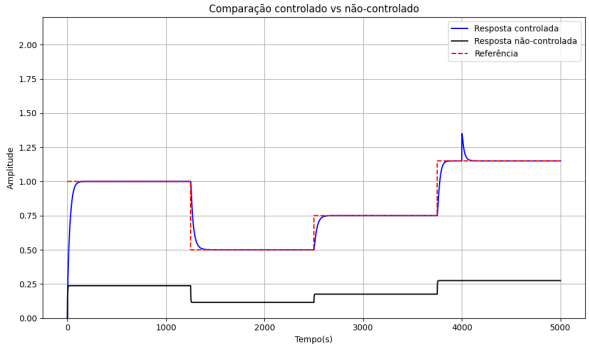

Figura 8: Resposta com pertubação no sinal de saída

Conclusão

Observou-se que o controle RST com identificação direta via Mínimos Quadrados Recursivo (MQR) é capaz de estabilizar sistemas não lineares utilizando unicamente as informações provenientes dos sinais de entrada e saída. Essa abordagem dispensa o conhecimento explícito do modelo interno do sistema, demonstrando-se eficiente mesmo frente à presença de não linearidades.

Além disso, verificou-se que o uso de fator de esquecimento foi fundamental para ajustar o desempenho do controle, especialmente em situações em que o sinal de referência possui valores superiores a 1. Esse fator permitiu que o algoritmo priorizasse informações recentes, favorecendo a adaptação do controle a diferentes regimes operacionais.

Outro aspecto relevante é que atribuir um ganho exclusivamente ao polinômio $T(z)$ se mostrou equivalente a aplicar ganhos simultâneos aos polinômios $R(z)$, $S(z)$ e $T(z)$. Essa escolha simplificou significativamente o processo de ajuste, uma vez que a modificação em um único polinômio foi suficiente para adequar a resposta do sistema.

Por fim, constatou-se que o controle implementado é capaz de interpretar perturbações no sinal de saída como se fossem alterações no sinal de referência. Essa característica evidencia a robustez e a capacidade de adaptação do controlador, uma vez que ele ajusta sua ação de forma a manter o sistema estabilizado mesmo frente a perturbações externas.

Referências

[1] K. J. Åström e T. Hägglund. *Advanced PID Control*. Research Triangle Park: ISA, 2006.

[2] K. J. Åström e B. Wittenmark. *Adaptive Control*. 2nd. New York: Dover Publications, 2013.

[3] G. C. Goodwin, S. F. Graebe e M. E. Salgado. *Control System Design*. Upper Saddle River: Prentice Hall, 2001.

[4] I. D. Landau et al. *Adaptive Control: Algorithms, Analysis and Applications*. London: Springer, 2011.

[5] L. Ljung. *System Identification: Theory for the User*. 2nd. Upper Saddle River: Prentice Hall, 1999.

[6] T. Söderström e P. Stoica. *System Identification*. London: Prentice Hall, 1989.

Anexos

Código

```
import numpy as np
import matplotlib.pyplot as plt


def sistema(u, y1):
    """Função que define o sistema dinâmico"""
    return 0.08 * y1**2 + 0.56 * y1 + 0.1 * u

# Configurações iniciais
np.random.seed(0)
K = 1
nit = 5000

# Gera sinal aleatório
eta = np.zeros(nit)
for i in range(nit):
    eta[i] = 1 if np.random.rand() > 0.5 else -1

# Parâmetros iniciais
theta = np.zeros(4)
ruido = eta * 0.00
P = 1000 * np.eye(4)

# Fatores de esquecimento
lambds0 = 1.1
lambds1 = .75
lambda_ = lambds0 / lambds1

# Sinal de referência
uc = np.concatenate([
    1.00 * np.ones(nit//4),
    0.50 * np.ones(nit//4),
    0.75 * np.ones(nit//4),
    1.15 * np.ones(nit//4)
])

# Perturbação
dist =  0.2 * np.concatenate([np.zeros(4000), np.ones(5000)])

# Inicialização de arrays
y = np.zeros(nit)
erro = np.zeros(nit)
bt, t0, s0, s1 = np.ones(nit), np.ones(nit),np.ones(nit),np.ones(nit)
a1, a2, a3, a4 = np.ones(nit),np.ones(nit),np.ones(nit),np.ones(nit)
yr, yd, ysc, ypi = np.zeros(nit), np.zeros(nit), np.zeros(nit), np.zeros(nit)
e, epi = np.zeros(nit), np.zeros(nit)
u = np.ones(nit)
Kc=1

# Condições iniciais
```

```
for t in range(2):
    y[t] = 0
    erro[t] = 0
    a1[t], a2[t], a3[t], a4[t] = theta

# Laço principal
for t in range(2, nit):
    yr[t] = sistema(u[t-1], yr[t-1])
    ysc[t] = sistema(uc[t-1], ysc[t-1])                    # sinal de saída sem controle
    y[t] = yr[t] #+ dist[t]                                 # sinal de saída com controle
    e[t] = uc[t] - y[t]

    fi = np.array([uc[t-1], uc[t-2], y[t-1], -y[t-2]])     # Matriz de regresores
    erro[t] = y[t] - fi.T @ theta                          # resíduo

    k_gain = P @ fi / (lambda_ + fi.T @ P @ fi)            # Ganho MMQ
    theta = theta + k_gain * erro[t]                       # Parâmetros MMq
    P = (P - np.outer(k_gain, fi.T) @ P) / lambds0         # Matriz de covariância MMQ

    a1[t], a2[t], a3[t], a4[t] = theta                     # Atualiza valores de parâmetros

    bt[t] = 1+(a1[t] +a2[t])-(a3[t]+a4[t])                 # Determinar T*b


    s0[t] = (a1[t]/bt[t])                                  # Determinar s_0
    s1[t] = (a2[t]/bt[t])                                  # Determinar s_1
    t0[t] = (s0[t] + s1[t])                                # Determinar T

    # Lei de controle
    u[t] = np.array([t0[t],s0[t], s1[t],1]) @ np.array([uc[t], -y[t], -y[t-1], u[t-1]])

# Plotagem
t = np.arange(nit)

plt.figure(figsize=(12, 12))
plt.subplot(2, 2, 1)
plt.plot(t, s0, 'r', linewidth=1.5)
plt.title('s0')
plt.ylim(0, .2)
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(t, s1, 'r', linewidth=1.5)
plt.title('s1')
plt.ylim(0, .2)
plt.grid(True)

plt.subplot(2, 2, 3)
plt.plot(t, t0, 'k', linewidth=1.5)
plt.title('T')
plt.ylim(0, .2)
plt.grid(True)
```

```
plt.figure(figsize=(10, 6))
plt.plot(t, y, 'b', label='Resposta controlada', linewidth=1.5)
plt.plot(t, ysc, 'k', label='Resposta não-controlada', linewidth=1.5)
plt.plot(t, uc, '--r', label='Referência', linewidth=1.5)
plt.ylim(0, 2.2)
plt.grid(True)
plt.title('Comparação controlado vs não-controlado')
plt.xlabel('Tempo(s)')
plt.ylabel('Amplitude')
plt.legend(loc='upper right')
plt.tight_layout()

plt.show()
```