Aplicação do Método do Relé para identificação e controle PID com escalonamento de ganhos de um sistema não linear

Otávio A. B. Maia * **Raul F. A. D. Bueno** *
Tiago G. de Oliveira * **Luiz F. Pugliese** * **Fadul F. Rodor** *

* *Instituto de Ciências Tecnológicas - ICT*
Universidade Federal de Itajubá – Campus Itabira
(e-mail: otavioabmaia@gmail.com, raulfadb@gmail.com,
fadulrodor@unifei.edu.br, tgaiba@unifei.edu.br, pugliese@unifei.edu.br,
fadulrodor@unifei.edu.br).

Abstract: In this paper, the Relay Method will be used to identify different regions of a nonlinear system, approximate them to a first order plus dead time (FOPDT) linear system and autotune PI controllers for each of those regions. A Gain Scheduled contoller will be applied, and it's time response will be compared to a PI controller with static gains tuned in the central region of the system. It's performance will also be evaluated as to disturbance rejection. The system is part of a learning-purpose level control plant, composed by two tanks, one pump and one proportional pneumatic valve. The results show that the fixed gain PI controller provides a lower settling time in higher regions of the tank, while the controller with variable gains performs faster in lower regions. Considering the industrial environment, the Relay Method showed to be a good autotuning technique, because it is little invasive and allows to tune controllers in a simple way, meeting the specifications even in nonlinear systems.

Resumo: Neste artigo, o Método do Relé será utilizado para identificar regiões diferentes de um sistema não linear, aproximá-las às de um sistema linear de primeira ordem com atraso de transporte e sintonizar controladores do tipo PI para estas regiões. Será aplicado o controle utilizando escalonamento de ganhos, e sua resposta temporal será comparada à de um controlador PI de ganhos fixos sintonizado na região central do sistema. Seu desempenho também será avaliado quanto a rejeição a distúrbios. O sistema utilizado é uma planta didática de controle de nível, composta por dois tanques, uma bomba e uma válvula proporcional pneumática. Os resultados indicam que o PI de ganhos fixos apresenta menor tempo de acomodação nas regiões mais altas do tanque, enquanto o de ganhos variáveis se mostrou mais rápido em regiões mais baixas. Considerando o ambiente industrial, o Método do Relé mostrou ser uma boa técnica de *autotuning*, pois é pouco invasivo e permite sintonizar controladores de maneira simples, atendendo às especificações mesmo em sistemas não lineares.

Keywords: Relay Method; Gain Scheduling; autotune.
Palavras-chaves: Método do Relé; escalonamento de ganho; *autotune*.

1. INTRODUÇÃO

O controlador PID (Proporcional, Integral e Derivativo) é o mais utilizado nas indústrias, que podem ter uma grande quantidade de malhas de controle, responsáveis por manter variáveis como nível, vazão e temperatura em um determinado valor. O desempenho dessas malhas é diretamente afetado pela qualidade da sintonia do controlador, e por isso, gerenciá-las pode não ser uma tarefa fácil, pois o ajuste manual pode demandar muito tempo e alguns processos são complexos e de difícil modelagem/identificação. Nesse contexto, a utilização de um método automático de sintonia de controladores que forneça parâmetros adequados ao processo de maneira rápida e confiável pode trazer grandes benefícios. Um método comumente aplicado para realizar esse procedimento é o Método do Relé.

Proposto por Åström and Hägglund (1984), o procedimento consiste em fechar a malha com um relé gerando o sinal de controle, desativando temporariamente o PID. Dessa forma, quando a variável de processo atinge valores definidos como limites, o relé comuta o sinal de controle, gerando oscilações no processo. Quando forem simétricas, obtém-se o período e a amplitude dessas oscilações, e, com esses parâmetros, é possível sintonizar um controlador PID. Essa sintonia é feita a partir de outros métodos que se baseiam nesses mesmos parâmetros para a obtenção dos ganhos de controladores, como o segundo método de Ziegler-Nichols (Ziegler and Nichols, 1942) e os métodos

[1] Reconhecimento do suporte financeiro deve vir nesta nota de rodapé.

de Tyreus-Luyben (Tyreus and Luyben, 1992) e Luyben (Luyben, 1996).

De acordo com Berner and Åström (2018), a aplicação deste método nas indústrias não mudou tanto até os dias atuais, embora o conhecimento sobre teoria de controle e o poder computacional dos dispositivos tenham crescido bastante. Apesar disso, vários estudos propuseram avanços no método original: (Friman and Waller, 1997), (Li and Luyben, 1991), (Wang and Zou, 1997) e (Luyben, 1987) modificaram-o para a aplicação em sistemas de primeira ou segunda ordem com atraso de transporte, enquanto (Kaya and Atherton, 2001) e (Shen and Yu, 1996) utilizaram oscilações assimétricas de relés no processo de excitação do sistema.

Segundo Wang and Zou (1997), o modelo dinâmico mais utilizado em processos industriais é o de primeira ordem com atraso de transporte. Ele propõe a utilização do Método do Relé não só para a sintonia de controladores, mas também para a identificação de sistemas. Em Liu and Huang (2013), é apresentada uma revisão do atual patamar da modelagem de processos a partir desta técnica.

Neste trabalho, o Método do Relé será aplicado em um Sistema de Controle Distribuído (*Distributed Control System* - DCS) da empresa ABB para identificar diferentes regiões de um sistema não linear, aproximando-as por funções de transferência de primeira ordem com atraso de transporte. O método também será utilizado na sintonia de controladores PI para até quatro regiões diferentes, e o controle será feito através do escalonamento dos ganhos obtidos com o *autotuning*. Para representar este tipo de sistema, será utilizada uma planta de controle de nível composta por dois tanques, uma bomba e uma válvula proporcional de acionamento pneumático. Considerando que o processo é não linear, serão comparados os desempenhos de controladores de ganhos fixos e ganhos variáveis, realizando, neste último caso, a aplicação de distúrbios.

O artigo está dividido da seguinte maneira: as Seções 2 e 3 detalham o processo de nível e o método utilizado, respectivamente, enquanto a Seção 4 descreve sua aplicação. A Seção 5 apresenta os resultados obtidos nas etapas de identificação e sintonia, bem como as análises referentes à comparação dos desempenhos apresentados pelos PI's e à rejeição de distúrbios.

2. SISTEMA DE CONTROLE E PLANTA DE NÍVEL

O processo a ser controlado é parte de uma planta didática do Laboratório de Automação (Figura 1). Esta planta representa, em escala reduzida, um processo industrial com várias malhas, possibilitando a implementação e testes de estratégias de controle através da manipulação de variáveis de nível, vazão e temperatura. Possibilita também o estudo de redes industriais através da configuração da comunicação entre os instrumentos.

A planta é controlada por um DCS AC700F, modelo da linha ABB Freelance. Ele é composto por uma CPU PM783F, um cartão de I/O digital DC732F, um cartão de I/O analógico AX722F e um mestre de rede Profibus DP CM772F. A comunicação com os dispositivos da rede Profibus PA é feita através da *gateway* DP/PA Pepperl-Fuchs HD2-GTB-2PA, enquanto os dispositivos Hart se

Figura 1. Planta didática do Laboratório de Automação.

comunicam com a CPU via cartão de I/O analógico. O *software* de programação do DCS utilizado no trabalho é o Control Builder F.

O diagrama do processo utilizado neste estudo é apresentado na Figura 2. Ele é composto por dois tanques acoplados através de uma tubulação plástica, sendo que o Tanque 1 funciona como um reservatório de água que abastece o Tanque 2 por meio de uma bomba centrífuga, cuja saída está conectada a uma válvula de abertura proporcional e acionamento pneumático. O objetivo deste trabalho é controlar o nível do Tanque 2 através da velocidade da bomba.

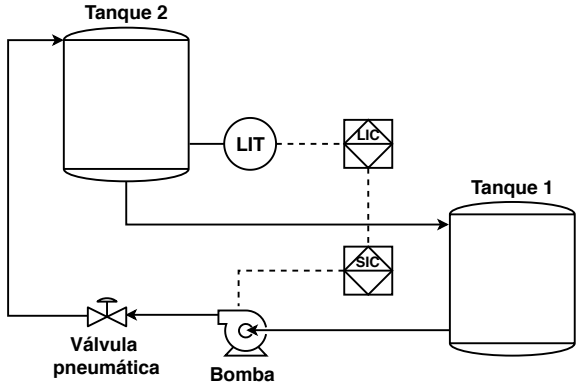

Figura 2. Esquema representativo do processo de controle de nível.

3. PARAMETRIZAÇÃO DE CONTROLADORES PID

O trabalho desenvolvido por Ziegler and Nichols (1942) apresentou dois dos métodos mais comumente utilizados no processo de sintonia de controladores. O primeiro método de Ziegler-Nichols, também conhecido como método da curva de reação, consiste em obter um par de parâmetros a partir da resposta ao degrau do processo em malha aberta. O segundo método de Ziegler-Nichols ou método do ganho crítico, consiste em operar o sistema em malha fechada com um controlador do tipo proporcional, aumentando-se gradativamente o ganho do controlador até que se observe oscilações sustentadas na saída do sistema. O ganho responsável por gerar estas oscilações é chamado de Ganho Crítico (K_u), e o período delas é conhecido como Período Crítico (P_u). Com esses dois parâmetros, é possível sintonizar controladores dos tipos P, PI ou PID a partir da tabela proposta por Ziegler e Nichols, que especifica um *overshoot* máximo de 25% na saída. A Tabela 1 (Yu, 2006) mostra como encontrar os termos K_P, T_I e T_D a partir de K_u e P_u.

Tabela 1. A tabela de Ziegler-Nichols para sintonia de controladores P, PI e PID.

Controlador	K_P	T_I	T_D
P	$K_u/2$	-	-
PI	$K_u/2.2$	$P_u/1.2$	-
PID	$K_u/1.7$	$P_u/2$	$P_u/8$

Embora seja prático, este método possui vários pontos negativos, pois leva o sistema a seus limites de operação, e dependendo da frequência das oscilações, estas podem causar desgastes nos atuadores ou fazer com que a sintonia demore um tempo impraticável (Åström and Hägglund, 1984). Por ser extremamente invasivo, o método se torna inviável para o ambiente industrial, visto que pode exigir uma parada do processo produtivo enquanto a rotina de sintonia é executada.

Sendo assim, o Método do Relé, proposto por Åström and Hägglund (1984), surge como uma alternativa. Este método propõe melhorias ao segundo método de Ziegler-Nichols, permitindo que o ganho crítico e período crítico sejam determinados para diferentes regiões do sistema de maneira automática, não sendo necessário realizar ajustes manuais no ganho. Isto torna o Método do Relé muito mais rápido e menos invasivo que o método original, e por isso sua utilização é mais viável em ambientes industriais. A possibilidade de identificar parâmetros da planta e sintonizar controladores em regiões específicas sem exigir que o atuador ou a saída oscile entre seus limites de operação o tornam uma excelente opção de *autotuner* para PID's (Wang and Zou, 1997).

3.1 Sintonia do controlador

Segundo Åström and Hägglund (1984), a sintonia do controlador pode ser automatizada através da introdução de um relé operando em malha fechada, conforme mostrado na Figura 3. Neste método, o operador do processo deve suspender temporariamente o sinal de controle u_c que é gerado pelo controlador PID, e submeter o processo ao sinal de controle u_r, que é gerado pelo relé.

Na Figura 3, o sinal SP é a referência do sistema (*set-point*), MV é o sinal de controle aplicado no atuador (variável manipulada) e PV é a saída do sistema (variável de processo).

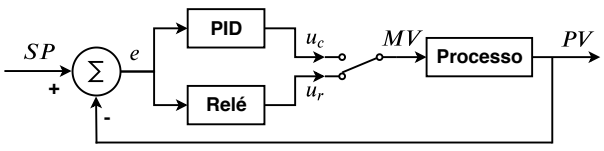

Figura 3. Malha fechada com o relé.

De maneira a ampliar o modo com que o relé era utilizado no método original, foram desenvolvidas algumas configurações alternativas (Yu, 2006). Uma delas acrescenta uma histerese ao mesmo com a finalidade de mitigar o efeito de possíveis ruídos no sinal da variável do processo. A Figura 4 representa o funcionamento do relé com a introdução da histerese. Ele deverá variar o sinal de controle u_r entre as amplitudes $\mu_0 - \mu$ e $\mu_0 + \mu$ de forma que a saída do sistema (PV) apresente oscilações sustentadas, ou seja, até que o tempo que ela demora para partir do limite inferior ($SP - \epsilon$) e atingir o limite superior ($SP + \epsilon$) seja igual ao tempo que demora para partir do limite superior e retornar ao limite inferior. Estes limites são calculados a partir de um valor Δ desejado, que representa o percentual de variação da PV ao redor do *setpoint*. Neves (2009) sugere valores típicos para Δ entre 2% e 5%, e neste trabalho o valor adotado foi 5%. Logo, calcula-se ϵ conforme a Equação (1).

$$\epsilon = SP \cdot \Delta \tag{1}$$

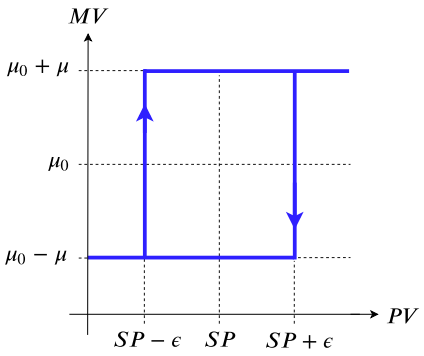

Figura 4. Relé com histerese.

A Figura 5 mostra o comportamento da PV quando o processo é submetido ao sinal de controle do relé da Figura 4. Percebe-se que o período crítico P_u pode ser determinado somando-se os tempos em que o sinal de controle permanece em seus níveis baixo (t_d) e alto (t_u), conforme a Equação (2). O termo L é o tempo morto (atraso de transporte), que é obtido medindo-se o tempo que a PV demora para partir do valor $SP + \epsilon$ e atingir sua amplitude de pico A_u. O valor de a é a diferença entre as amplitudes máxima e mínima da PV, respectivamente.

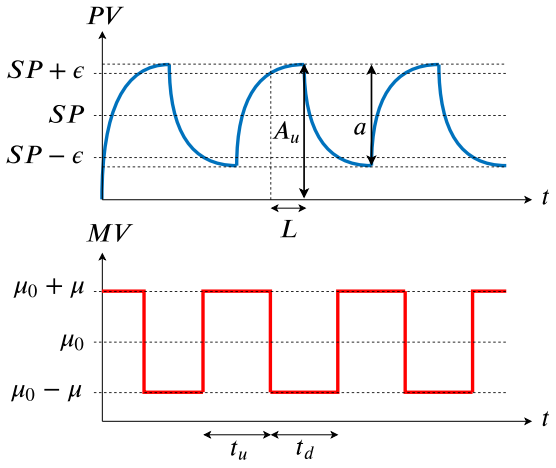

Figura 5. O sistema operado pelo relé.

$$P_u = t_d + t_u \tag{2}$$

Como o Método do Relé convencional exige que a saída oscile com $t_d = t_u$, faz-se necessário buscar uma maneira de determinar as amplitudes $\mu_0 - \mu$ e $\mu_0 + \mu$ do sinal de controle de forma a atender este requisito. Segundo (Neves, 2009), o sinal de controle pode ser obtido de forma iterativa a partir da Equação (3), que calcula um novo valor para u_r a partir da diferença entre t_d e t_u. A constante K_0 é um fator que pondera a intensidade da correção, e neste trabalho foi utilizada com o valor unitário.

$$u_r[k] = u_r[k-1] + K_0 \cdot u_r[k-1] \cdot \frac{t_u - t_d}{2P_u} \tag{3}$$

O sinal de controle deve ser recalculado até que a oscilação seja considerada simétrica dentro de uma faixa de tolerância. Neste trabalho, adotou-se o mesmo critério do autor, fazendo com que o procedimento se repita até que o valor absoluto da diferença entre t_d e t_u em relação a P_u seja menor que 10%, conforme a Equação (4).

$$\left| \frac{t_u - t_d}{P_u} \right| < 0,1 \tag{4}$$

Quando a Equação (4) for satisfeita, o ganho crítico pode ser calculado a partir da Equação (5). De posse dos valores de K_u e P_u, é possível utilizar a Tabela 1 para calcular os ganhos do controlador desejado.

$$K_u = \frac{4\mu}{\pi\sqrt{a^2 - \epsilon^2}} \tag{5}$$

3.2 Identificação

O avanço dos estudos sobre a utilização do Método do Relé permitiu a elaboração de novos métodos que estendem a proposta da aplicação em *autotuners*, permitindo sua utilização na identificação de sistemas. Este trabalho utilizará o procedimento proposto por Wang and Zou (1997) para estimar os parâmetros do sistema de primeira ordem com atraso de transporte, mostrado na Equação (6).

$$G(s) = \frac{Ke^{-Ls}}{\tau s + 1} \tag{6}$$

Quando o processo controlado por um relé com histerese, como o da Figura 4, atinge as oscilações simétricas, o ganho K da planta pode ser determinado dividindo-se a integral da saída $PV(t)$ pela integral da entrada $u_r(t)$, conforme apresentado na Equação (7).

$$K = G(0) = \frac{\int_0^{P_u} PV(t)dt}{\int_0^{P_u} u_r(t)dt} \tag{7}$$

A estimação do valor de τ exige o cálculo do parâmetro θ, que segundo o autor, é a constante de tempo normalizada do sistema. Seu valor é obtido através da Equação (8), na qual A_u é a amplitude máxima que a saída atinge durante o período crítico, como pode ser observado na Figura 5. Em seguida, τ é calculado utilizando-se a Equação (9).

$$\theta = ln\frac{(\mu_0 + \mu)K - \epsilon}{(\mu_0 + \mu)K - A_u} \tag{8}$$

$$\tau = t_u \left(ln\frac{2\mu K e^{\theta} + \mu_0 K - \mu K + \epsilon}{\mu K + \mu_0 K - \epsilon} \right)^{-1} \tag{9}$$

Com os métodos apresentados na presente seção e na seção anterior, é possível identificar os parâmetros e sintonizar o controlador de qualquer região desejada da planta, executando a rotina uma única vez.

4. MALHA DE CONTROLE APLICADA

A estratégia utilizada para manipular o nível do Tanque 2 (Figura 2) consiste na implementação de um controlador PID com *anti-windup* e ponderação do *setpoint* na realimentação do erro das parcelas P e D, com ganhos escalonados. A Figura 6 mostra a malha de controle utilizada neste trabalho. As subseções seguintes apresentarão com mais detalhes cada uma das implementações realizadas.

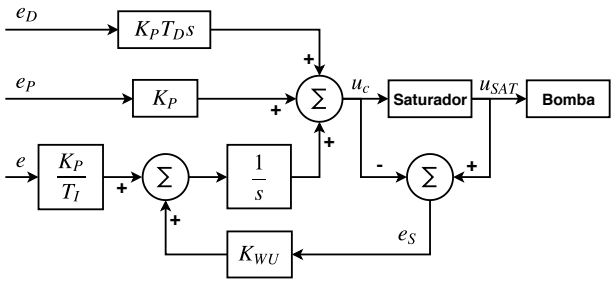

Figura 6. Malha de controle PI/PID com *anti-windup* e ponderação do *setpoint*.

4.1 Anti-windup

Para a implementação do *anti-windup*, calcula-se o erro do saturador (Equação (10)), que é multiplicado por um ganho K_{WU} e somado ao erro da saída do sistema a ser acumulado no integrador (Åström and Hägglund, 2006), como mostra a Figura (6).

$$e_S(t) = u_{SAT} - u_c \tag{10}$$

Quando o atuador não está saturado, $e_S = 0$ e, portanto, a ação *anti-windup* não interfere no integrador. Porém, quando a saturação ocorre, a ação integral é decrementada e isto impede seu acúmulo excessivo. Neste trabalho, adotou-se o valor unitário para o ganho K_{WU}.

4.2 Controlador PID

A lei de controle u_c do PID é apresentada na Equação (11), em que P é a ação proporcional, I é a ação integral e D a ação derivativa.

$$u_c(t) = P(t) + I(t) + D(t) \tag{11}$$

As equações das ações de controle $P(t)$, $I(t)$ e $D(t)$ são apresentadas em (12), (13) e (14), respectivamente. Os

termos K_P, T_I e T_D são resultados da interpolação linear realizada na etapa de *Gain Scheduling* e, portanto, são parâmetros que variam com o tempo. Os termos e_P e e_D são resultados da ponderação do *setpoint* e serão esclarecidos na seção seguinte.

$$P(t) = K_P \cdot e_P \tag{12}$$

$$I(t) = \int_0^t \left[\frac{K_P}{T_I} \cdot e(\tau) + K_{WU} \cdot e_S(\tau) \right] d\tau \tag{13}$$

Devido a alta sensibilidade a ruídos, Åström propõe que a ação derivativa tenha seu ganho limitado através da implementação de um filtro de primeira ordem, conforme a Equação (14). O autor sugere que valores típicos para N estão entre 8 e 20. Neste trabalho, adotou-se $N = 20$.

$$D(t) = K_P T_D \frac{de_D}{dt} - \frac{T_D}{N} \frac{dD(t)}{dt} \tag{14}$$

4.3 Ponderação do setpoint

Åström propõe um controlador mais flexível ao introduzir os coeficientes b e c, responsáveis por ponderar o *setpoint* no cálculo do erro das ações proporcional e derivativa, como mostrado nas Equações (15) e (16), respectivamente. Segundo Åström, a diminuição do coeficiente b pode resultar na diminuição do *overshoot* na saída do sistema.

O coeficiente c é escolhido com o intuito de reduzir o impacto de mudanças bruscas de *setpoint* no sinal de controle. No instante em que elas ocorrem, se c for maior que zero, deriva-se um degrau em e_D, fazendo com que um grande valor seja somado à parcela derivativa.

As respostas do controlador a distúrbios de carga e ruídos de medição independem de b e c. Seguindo as recomendações do autor, neste trabalho os valores de b e c serão 1 e 0, respectivamente.

$$e_P(t) = b \cdot SP - PV \tag{15}$$

$$e_D(t) = c \cdot SP - PV \tag{16}$$

4.4 Gain Scheduling

O controle utilizando *Gain Scheduling* será feito através da interpolação linear de até quatro regiões da PV. Optou-se por saturar a interpolação em seus valores mínimo e máximo quando a PV extrapolar os limites inferior e superior, respectivamente. A Equação (17) mostra como é feita interpolação de quatro valores do ganho K_P nas regiões R_1, R_2, R_3 e R_4. Este conceito estende-se no escalonamento dos tempos T_I e T_D.

$$K_P(PV) = \begin{cases} K_{P1}, & PV < R_1 \\ \dfrac{K_{P1}(R_2 - PV) + K_{P2}(PV - R_1)}{R_2 - R_1}, & R_1 \leq PV < R_2 \\ \dfrac{K_{P2}(R_3 - PV) + K_{P3}(PV - R_2)}{R_3 - R_2}, & R_2 \leq PV < R_3 \\ \dfrac{K_{P3}(R_4 - PV) + K_{P4}(PV - R_3)}{R_4 - R_3}, & R_3 \leq PV < R_4 \\ K_{P4}, & PV \geq R_4 \end{cases} \tag{17}$$

4.5 Discretização do controlador

As Equações contínuas (11), (12), (13) e (14) necessitam ser discretizadas a fim de serem implementadas no DCS. As derivadas serão aproximadas por *Backward differences* (Åström and Hägglund, 2006). As Equações (18), (19), (20) e (21) mostram, respectivamente, as discretizações do sinal de controle, ação proporcional, ação integral e ação derivativa. O termo T_a é o tempo de amostragem.

$$u_c[k] = P[k] + I[k] + D[k] \tag{18}$$

$$P[k] = K_P \cdot e_P[k] \tag{19}$$

$$I[k] = I[k-1] + \left(\frac{K_P}{T_I} \cdot e[k] + K_{WU} \cdot e_S[k] \right) T_a \tag{20}$$

$$D[k] = \frac{T_D}{T_D + NT_a} D[k-1] + \frac{K_P T_D N}{T_D + NT_a} (e_D[k] - e_D[k-1]) \tag{21}$$

5. RESULTADOS

Foram desenvolvidos quatro blocos de função no *software* Control Builder F: IDENT_SINT, N_IDENT_SINT, GAIN_SCHED e PID_WINDUP. Eles foram programados utilizando a linguagem Texto Estruturado (*Structured Text* - ST), que é especificada na norma IEC 61131-3. O sistema supervisório da planta de nível foi implementado no *software* DigiVis, também da linha ABB Freelance.

5.1 Blocos de função

O bloco IDENT_SINT executa o Método do Relé uma única vez para encontrar o ganho K_u e o período P_u de uma região desejada do sistema. O controlador é calculado com base na Tabela 1.

A sintonia dos controladores em mais de uma região é feita através do bloco N_IDENT_SINT, que executa a rotina IDENT_SINT até quatro vezes, de acordo com o desejo do usuário. Este bloco escolhe automaticamente as regiões que serão identificadas, calculando-as de forma a estarem equidistantes em relação aos limites de operação do sistema.

O bloco GAIN_SCHED realiza o escalonamento de ganhos conforme a Equação (17). Ele utiliza os ganhos obtidos com o N_IDENT_SINT para escalonar K_P, T_I e T_D em função da PV. O bloco PID_WINDUP implementa o controlador PID apresentado anteriormente.

Os blocos desenvolvidos são descritos com mais detalhes no Apêndice A.

5.2 Identificação, sintonia e controle

Utilizou-se o bloco IDENT_SINT para sintonizar um controlador PI no nível 50% do Tanque 2. A Figura 7 mostra as oscilações do nível e a variação do sinal de controle durante a execução do Método do Relé. A Tabela 2 apresenta os parâmetros da planta e do controlador obtidos com a rotina.

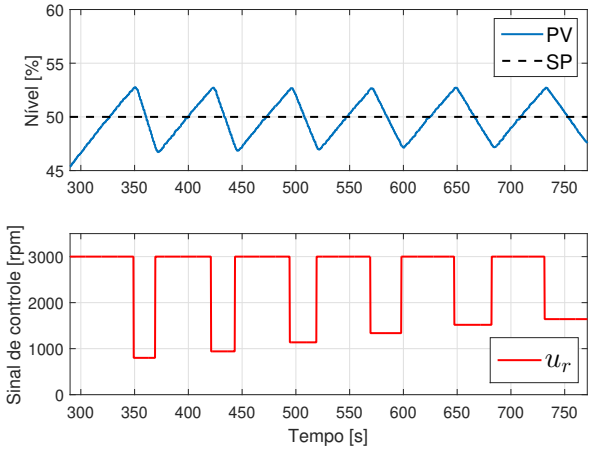

Figura 7. Rotina de identificação e sintonia do PI para a região 50% com o bloco IDENT_SINT.

Tabela 2. Parâmetros da planta e do controlador PI obtidos com o bloco IDENT_SINT para a região de nível 50%.

Parâmetros da planta		**Parâmetros do controlador**	
K	$0.0211\left[\frac{\%}{rpm}\right]$	K_P	$81.3473\left[\frac{rpm}{\%}\right]$
τ	$38.46[s]$	T_I	$71.25[s]$
L	$2.1[s]$	T_D	$0.0[s]$

O bloco N_IDENT_SINT foi utilizado na sintonia de controladores PI em 4 regiões do Tanque 2. As Tabelas 3 e 4 mostram, respectivamente, os parâmetros da planta e dos controladores obtidos para cada região.

Tabela 3. Parâmetros das regiões da planta obtidos com o bloco N_IDENT_SINT.

Região [%]	$\boldsymbol{K}\left[\frac{\%}{rpm}\right]$	$\boldsymbol{\tau}[s]$	$\boldsymbol{L}[s]$
5.5	0.0033	8.22	2.7
30.69	0.0163	18.05	1.8
55.83	0.0227	47.19	1.8
81.0	0.0297	102.99	2.4

Tabela 4. Parâmetros dos controladores PI obtidos com o bloco N_IDENT_SINT.

Região [%]	$\boldsymbol{K_P}\left[\frac{rpm}{\%}\right]$	$\boldsymbol{T_I}[s]$
5.5	353.8738	15.25
30.69	159.6915	27.5
55.83	63.9729	92.0
81.0	21.8503	272.5

A Figuras 8 e 9 mostram as respostas do sistema em malha fechada com os controladores de ganhos fixos e de ganhos escalonados, respectivamente, em 4 regiões diferentes: 8%, 18%, 68% e 82% de nível.

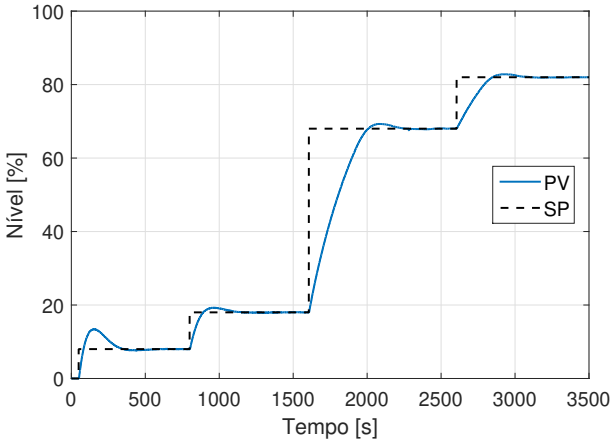

Figura 8. Respostas do sistema em malha fechada para o controlador PI de ganhos fixos sintonizado em 50%.

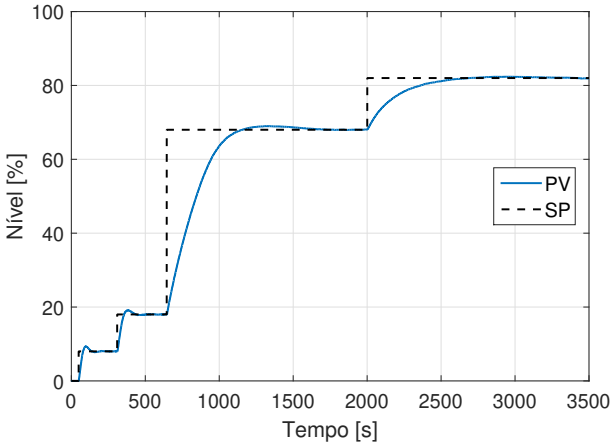

Figura 9. Respostas do sistema em malha fechada para o controlador PI com ganhos escalonados.

As Tabelas 5 e 6 sintetizam as medições da resposta temporal de cada região para os controladores de ganhos fixos e escalonados, respectivamente.

Tabela 5. Resposta do sistema à aplicação de 4 degraus com controlador PI de ganhos fixos.

Região [%]	**Instante de aplicação do degrau** [s]	**Tempo de acomodação** [s]	**Overshoot** [%]
8.0	50.7	289.2	67.97
18.0	801.6	284.4	7.15
68.0	1606.5	372.3	1.90
82.0	2606.7	193.2	0.97

Tabela 6. Resposta do sistema à aplicação de 4 degraus com o controlador PI de ganhos escalonados.

Região [%]	**Instante de aplicação do degrau** [s]	**Tempo de acomodação** [s]	**Overshoot** [%]
8.0	50.7	46.5	17.08
18.0	311.7	107.1	6.54
68.0	646.5	435.3	1.45
82.0	2003.7	387.3	0.43

Ao utilizar o controlador sintonizado na região de 50%, esperava-se que o desempenho fosse pior em regiões mais afastadas, visto que o processo de nível é não-linear. De fato, nota-se que em 8% o *overshoot* é 67.97%, superando o máximo de 25% previsto pela tabela de Ziegler-Nichols. Embora em todos os casos o sistema tenha rastreado a referência, os comportamentos do *overshoot* e do tempo de acomodação evidenciam as não linearidades da planta e a lentidão do PI de ganhos fixos para controlar o nível em regiões muito distantes para a qual ele foi sintonizado.

Ao comparar as respostas do controlador de ganhos fixos com o de ganhos escalonados, percebe-se que no segundo controlador a especificação do *overshoot* máximo de 25% foi atendida em todas as regiões, e a resposta do sistema ficou muito mais rápida nos níveis 8% e 18%. Em contrapartida, as regiões de 68% e 82% apresentaram um tempo de acomodação maior.

Este aumento é explicado pelo comportamento não linear do sistema de nível, evidenciado pelos parâmetros da Tabela 3. Quanto maior a quantidade de água no tanque, mais rápido ele tende a esvaziar, fazendo com que a diferença entre as velocidades necessárias para que a bomba faça o sistema oscilar com tempos iguais seja cada vez menor. Segundo a Equação (5), a diminuição de μ resultaria em um K_u menor, e, consequentemente, o ganho proporcional K_P também diminuiria. Isto pode ser confirmado ao observar a Tabela 4.

O PI de ganhos escalonados teve sua rejeição a distúrbios avaliada pela aplicação de um degrau em cada região do sistema, realizada após sua estabilização em regime permanente ao fechar-se a válvula proporcional de 90% para 65%. A Tabela 7 apresenta as medições dos impactos quanto ao tempo de recuperação e variação do nível em relação ao *setpoint*.

Tabela 7. Resposta do sistema à aplicação de distúrbios nas 4 regiões.

Região [%]	**Instante de aplicação do distúrbio** [s]	**Tempo de recuperação** [s]	**Variação da PV** [%]
8.0	229.2	27.9	3.46
18.0	282.3	33.0	2.39
68.0	1277.1	349.2	3.85
82.0	2162.1	729.0	5.58

Observando a Tabela 7, percebe-se que em todos os casos as variações na PV foram superiores à 2% do *setpoint*, ou seja, os distúrbios foram suficientes para tirar o sistema do regime permanente. Embora nas regiões mais altas a variação percentual da PV não tenha sido expressivamente maior em relação às regiões mais baixas, ao analisar o tempo de recuperação percebe-se que os impactos dos distúrbios foram bem mais elevados.

6. CONCLUSÃO

O Método do Relé mostrou ser uma excelente ferramenta em termos de praticidade. Seu algoritmo é simples de ser implementado em controladores que suportam programação via Texto Estruturado, como é o caso do DCS. Com ele foi possível sintonizar controladores que atendem a uma especificação e identificar parâmetros que aproximam determinadas regiões do sistema não linear às de um sistema linear de primeira ordem.

Os resultados mostram que os controladores obtidos com o Método do Relé para o sistema de nível só atendem as especificações da tabela de Ziegler-Nichols se a saída estiver próxima à região de sintonia. Porém, ao sintonizar mais de um controlador em regiões equidistantes, o controle com *Gain Scheduling* foi capaz de atender os critérios de desempenho em todas as regiões, embora em alguns casos a resposta temporal tenha sido mais lenta em relação ao controlador de ganhos fixos.

A rejeição de distúrbios ocorreu sem oscilações em todas as regiões, embora tenha sido mais demorada nos níveis mais altos, o que se atribui às características das ações de controle conferidas ao PI na etapa de sintonia. Nestes casos, para obter-se uma rejeição mais rápida, seriam indicados um maior ganho proporcional e um menor tempo integral.

Como ampliação deste trabalho, sugere-se a comparação dos resultados deste artigo aplicando-se novamente o Método do Relé, porém utilizando a válvula proporcional como atuador. Sugere-se também a aplicação do Método do Relé com oscilações assimétricas na sintonia dos controladores e o teste dos mesmos em outras malhas do sistema, como vazão e pressão.

REFERÊNCIAS

Berner, J., S.K.H.T. and Åström, K.J. (2018). An experimental comparison of PID autotuners. *Control Engineering Practice*, 73, 124 – 133.

Friman, M. and Waller, K.V. (1997). A two-channel relay for autotuning. *Industrial & Engineering Chemistry Research*, 36(7), 2662–2671.

Kaya, I. and Atherton, D. (2001). Parameter estimation from relay autotuning with asymmetric limit cycle data. *Journal of Process Control*, 11(4), 429 – 439.

Li, W., E.E. and Luyben, W.L. (1991). An improved autotune identification method. *Industrial & Engineering Chemistry Research*, 30(7), 1530–1541.

Liu, T., W.Q. and Huang, H. (2013). A tutorial review on process identification from step or relay feedback test. *Journal of Process Control*, 23(10), 1597 – 1623.

Luyben, W.L. (1987). Derivation of transfer functions for highly nonlinear distillation columns. *Industrial & Engineering Chemistry Research*, 26(12), 2490–2495.

Luyben, W.L. (1996). Tuning proportional-integral-derivative controllers for integrator/deadtime processes. *Industrial & Engineering Chemistry Research*, 35(10), 3480–3483.

Neves, M.G.d.S. (2009). *Auto-tuning de Controladores PID pelo método Relay*. Master's thesis, Universidade Técnica de Lisboa, Portugal.

Shen, S., W.J. and Yu, C. (1996). Use of biased-relay feedback for system identification. *AIChE Journal*, 42(4), 1174 – 1180.

Tyreus, B.D. and Luyben, W.L. (1992). Tuning PI controllers for integrator/dead time processes. *Industrial & Engineering Chemistry Research*, 31(11), 2625–2628.

Wang, Q., H.C. and Zou, B. (1997). Low-order modeling from relay feedback. *Ind. Eng. Chem. Res.*, 36, 375 – 381.

Yu, C. (2006). *Autotuning of PID Controllers - A Relay Feedback Approach*. Springer, 2 edition.

Ziegler, J.G. and Nichols, N.B. (1942). Optimum settings for automatic controllers. *trans. ASME*, 64(11).

Åström, K.J. and Hägglund, T. (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20(5), 645 – 651.

Åström, K.J. and Hägglund, T. (2006). *Advanced PID Control*. ISA - Instrumentation, Systems, and Automation Society.

A Figura A.1 mostra o bloco IDENT_SINT, desenvolvido para executar o Método do Relé em uma determinada região. Ao final da rotina, ele identifica os parâmetros de um sistema linear de primeira ordem com atraso de transporte e sintoniza um controlador do tipo PI ou PID baseando-se na Tabela 1. A Tabela A.1 descreve os pinos de entrada e saída do bloco e o Programa 1 apresenta o seu código fonte.

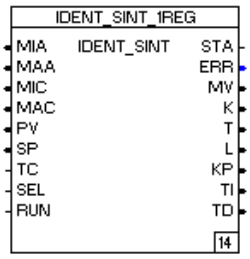

Figura A.1. Bloco IDENT_SINT.

Tabela A.1. Pinos de entrada e saída do bloco IDENT_SINT.

Entrada	**Descrição**
MIA	Valor mínimo de MV
MAA	Valor máximo de MV
MIC	Valor mínimo de PV
MAC	Valor máximo de PV
PV	Variável de processo
SP	Setpoint
TC	Tipo de controlador (0=PI/1=PID)
SEL	Origem dos comandos (0=Bloco/1=Faceplate)
RUN	Executar rotina
Saída	**Descrição**
STA	Status da rotina (0=Parado/1=Em execução)
ERR	Erro durante a execução do programa
MV	Sinal de controle
K	Ganho da planta
T	Constante de tempo da planta
L	Atraso de transporte da planta
KP	Ganho proporcional do controlador
TI	Tempo integral do controlador
TD	Tempo derivativo do controlador

```
1  (*agora: variável TIME que contém o valor atual do timer*)
2  (*t_inicio_subida: variável TIME que indica o instante que o nível começou a subir*)
3  (*t_inicio_descida: variável TIME que indica o instante que o nível começou a descer*)
4  (*direcao: variável BOOL que indica a direção do nível (TRUE=subindo, FALSE=descendo*)
5  (*primeira_exec: variável BOOL que indica se o loop do relé está sendo executado pela primeira vez*)
6  (*pico_pv_min: variável REAL que contém o menor valor de PV dentro do período Pu*)
7  (*pico_pv_max: variável REAL que contém o maior valor de PV dentro do período Pu*)
8  (*vec_pv: vetor REAL que contém o valor de PV. A posição [1] contém o valor atual, a posição [2] cont
       ém o valor atrasado em uma amostra, e assim por diante*)
9  int_pv := int_pv + ta*vec_pv[1];     (*Integral de PV*)
10 int_ur := int_ur + ta*mv;            (*Integral do sinal de controle*)
11
12 IF direcao THEN (*Se o nível está subindo*)
13
14     (*Buscando o valor de pico mínimo de PV*)
15     IF vec_pv[1] <= sp*(1.0-delta) THEN
16         pico_pv_min := vec_pv[1];
17         FOR i:=2 TO tam_vec DO
18             IF vec_pv[i] < pico_pv_min THEN
19                 pico_pv_min := vec_pv[i];
20             END_IF;
21         END_FOR;
22     END_IF;
```

```
23
24      (*vec_s_at_sp_min: vetor BOOL com o resultado da expressão "se o nível está subindo e atinge o
        limite inferior (SP-epsilon)". A posição [1] contém o valor atual, e a posição [2] contém o valor
         anterior. *)
25      vec_s_at_sp_min[2] := vec_s_at_sp_min[1];
26      vec_s_at_sp_min[1] := vec_pv[1] >= sp*(1.0-delta);
27
28      (*Detectando o instante em que o nível atinge SP-epsilon enquanto está subindo*)
29      IF vec_s_at_sp_min[1] AND NOT vec_s_at_sp_min[2] THEN
30          IF primeira_exec THEN
31              int_pv := 0.0;
32              int_ur := 0.0;
33              t_inicio_subida := agora;
34          END_IF;
35      END_IF;
36
37      (*vec_s_at_sp_max: vetor BOOL com o resultado da expressão "se o nível está subindo e atinge o
        limite superior (SP+epsilon)". A posição [1] contém o valor atual, e a posição [2] contém o valor
         anterior. *)
38      vec_s_at_sp_max[2] := vec_s_at_sp_max[1];
39      vec_s_at_sp_max[1] := vec_pv[1] >= sp*(1.0+delta);
40
41      (*Detectando o instante em que o nível atinge SP+epsilon enquanto está subindo*)
42      IF vec_s_at_sp_max[1] AND NOT vec_s_at_sp_max[2] THEN
43          direcao := FALSE;
44          t_inicio_descida := agora;
45          t_inicio_tm := agora;
46
47          (*Tempo total da subida *)
48          tu := TO_RE(TO_DI(agora - t_inicio_subida))/1000.0;
49      END_IF;
50
51      (*Enquanto o nível estiver subindo*)
52      IF NOT (vec_pv[1] >= sp*(1.0+delta)) THEN
53          IF vec_pv[1] > vec_pv[3] AND vec_pv[1] < sp*(1.0-delta) THEN
54              (*Lógica para que o tempo de subida despreze o tempo gasto para atingir SP-delta*)
55              tm_timer := agora;
56          END_IF;
57      END_IF;
58
59  ELSE (*Se o nível está descendo*)
60
61      (*Buscando o valor de pico máximo de PV*)
62      IF vec_pv[1] >= sp*(1.0+delta) THEN
63          pico_pv_max := vec_pv[1];
64          t_morto := TO_RE(TO_DI(agora - t_inicio_tm))/1000.0;
65          FOR i:=2 TO tam_vec DO
66              IF vec_pv[i] > pico_pv_max THEN
67                  pico_pv_max := vec_pv[i];
68                  t_morto := TO_RE(TO_DI(vec_t_pv[i] - t_inicio_tm))/1000.0;
69              END_IF;
70          END_FOR;
71      END_IF;
72
73      (*vec_d_at_sp_min: vetor BOOL com o resultado da expressão "se o nível está descendo e atinge o
        limite inferior (SP-epsilon)". A posição [1] contém o valor atual, e a posição [2] contém o valor
         anterior. *)
74      vec_d_at_sp_min[2] := vec_d_at_sp_min[1];
75      vec_d_at_sp_min[1] := vec_pv[1] <= sp*(1.0-delta);
76
77      (*Detectando o instante em que o nível atinge SP-epsilon enquanto está descendo*)
78      IF vec_d_at_sp_min[1] AND NOT vec_d_at_sp_min[2] THEN
79          direcao := TRUE;
80          t_inicio_subida := agora;
81
82          (*Tempo total da descida *)
83          td := TO_RE(TO_DI(agora - t_inicio_descida))/1000.0;
84
85          K_planta := int_pv/int_ur;
86          int_pv := 0.0;
```

```
87          int_ur := 0.0;
88
89          IF primeira_exec THEN
90              IF tu > td THEN
91                  corrige_subida := FALSE;
92              ELSE
93                  corrige_subida := TRUE;
94              END_IF;
95          END_IF;
96
97          primeira_exec := FALSE;
98
99          (*Se a diferença entre os tempos for maior que 10%, corrige-se o sinal de controle*)
100         IF abs(tu - td)/(tu + td) >= 0.10 THEN
101             IF NOT corrige_subida THEN
102                 (*Se o tempo de subida for o maior, corrige-se o sinal de controle da descida*)
103                 ur_descida := ur_descida + k0*ur_descida*(tu - td)/(2.0*(tu + td));
104             ELSE
105                 (*Se o tempo de descida for o maior, corrige-se o sinal de controle da subida*)
106                 ur_subida := ur_subida + k0*ur_subida*(tu - td)/(2.0*(tu + td));
107             END_IF;
108         END_IF;
109     END_IF;
110
111     IF vec_pv[1] <= sp*(1.0-delta) THEN (*Quando o nível acabou de descer e é hora de subir*)
112         IF NOT fim THEN
113
114             (*Se a diferença entre os tempos for menor que 10%*)
115             IF abs(tu - td)/(tu + td) < 0.10 THEN
116                 a := pico_pv_max - pico_pv_min;
117                 Au := pico_pv_max;
118                 epsilon := sp*delta;
119
120                 (* Identificação da planta *)
121                 mi_zero := (ur_subida + ur_descida)/2.0;
122                 mi := (ur_subida - ur_descida)/2.0;
123                 theta := ln(((mi+mi_zero)*K_planta-epsilon)/((mi+mi_zero)*K_planta-Au));
124                 tau := (tu)/LN((2.0*mi*K_planta*exp(theta) + mi_zero*K_planta - mi*K_planta + epsilon
    )/(mi*K_planta + mi_zero*K_planta - epsilon));
125
126                 (*Ganho crítico*)
127                 Kc := 4.0*mi/(3.1416*sqrt(a*a - epsilon*epsilon));
128
129                 (*Período crítico*)
130                 Tc := tu + td;
131
132                 (*PI Ziegler-Nichols*)
133                 kp_pi := 0.45*Kc;
134                 ti_pi := Tc/1.2;
135
136                 (*PID Ziegler-Nichols*)
137                 kp_pid := 0.6*Kc;
138                 ti_pid := Tc/2.0;
139                 td_pid := Tc/8.0;
140
141                 (*Rotina concluída*)
142                 fim := TRUE;
143             END_IF;
144         END_IF;
145     END_IF;
146 END_IF;
```

Programa 1. Código fonte do bloco IDENT_SINT.

O pino ERR é uma variável do tipo *WORD*. Quando ocorre algum erro durante a rotina, um determinado *bit* desta variável é alterado para o valor 1. Quando não há ocorrência de erros, o valor de ERR é zero. A Tabela A.2 detalha o significado de cada bit de erro.

Tabela A.2. Erros apresentados pelo pino ERR.

Posição do bit	Descrição do erro
1	Setpoint menor que o mínimo
2	Setpoint maior que o máximo
3	Sinal de controle menor que o mínimo
4	Sinal de controle maior que o máximo
5	Ganho da planta menor que zero
6	Constante de tempo da planta menor que zero
7	Atraso de transporte da planta menor que zero
8	Ganho proporcional menor que zero
9	Tempo integral menor que zero
10	Tempo derivativo menor que zero
11	Execução interrompida através da Faceplate
12	Execução interrompida através do pino RUN
13	Número de regiões inválido

A Figura A.2 apresenta o bloco N_IDENT_SINT, que executa a rotina IDENT_SINT até quatro vezes, permitindo a identificação de mais de uma região de operação do sistema e a sintonia de um controlador PI ou PID para cada uma delas.

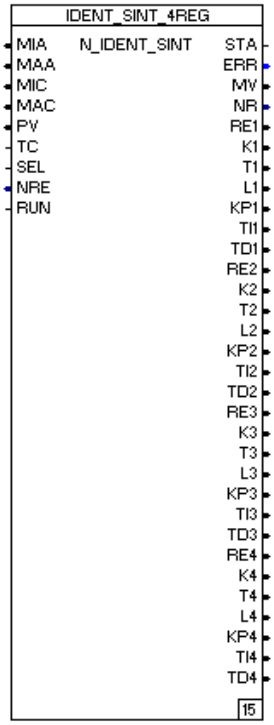

Figura A.2. Bloco N_IDENT_SINT.

Os pinos de entrada são praticamente os mesmos do bloco IDENT_SINT, com exceção do NRE, que define o número de regiões. Os pinos da saída incluem os parâmetros de identificação e sintonia de cada região, sendo que o valor correspondente a elas é apresentado nos pinos RE. O cálculo desses valores ocorre de forma automática, baseando-se no número de regiões escolhido pelo usuário através do pino NRE. O Programa 2 apresenta seu código fonte.

```
1  (*nre: variável UINT com o número de regiões escolhido pelo usuário*)
2  IF (exec_atual <= nre) AND executando THEN
3      (*Calculando as regiões*)
4      CASE nre OF
5          2:  R1 := MIC*1.1;
6              R2 := MAC*0.9;
7              CASE exec_atual OF
8                  1:  setpoint := MIC*1.1;
9                  2:  setpoint := MAC*0.9;
10             END_CASE;
11         3:  R1 := MIC*1.1;
```

```
12              R2 := (MIC*1.1 + MAC*0.9)/2.0;
13              R3 := MAC*0.9;
14              CASE exec_atual OF
15                  1:  setpoint := MIC*1.1;
16                  2:  setpoint := (MIC*1.1 + MAC*0.9)/2.0;
17                  3:  setpoint := MAC*0.9;
18              END_CASE;
19          4: R1 := MIC*1.1;
20              R2 := (MAC*0.9 - MIC*1.1)/3.0 + MIC*1.1;
21              R3 := 2.0*(MAC*0.9 - MIC*1.1)/3.0 + MIC*1.1;
22              R4 := MAC*0.9;
23              CASE exec_atual OF
24                  1:  setpoint := MIC*1.1;
25                  2:  setpoint := (MAC*0.9 - MIC*1.1)/3.0 + MIC*1.1;
26                  3:  setpoint := 2.0*(MAC*0.9 - MIC*1.1)/3.0 + MIC*1.1;
27                  4:  setpoint := MAC*0.9;
28              END_CASE;
29      END_CASE;
30
31      (*Se a execução da rotina no bloco IDENT_SINT estiver parada *)
32      IF (NOT ident_sint.RUN) AND executando THEN
33          IF exec_atual > 1 THEN
34              kp_pid[exec_atual-1] := ident_sint.KP;
35              ti_pid[exec_atual-1] := ident_sint.TI;
36              td_pid[exec_atual-1] := ident_sint.TD;
37          END_IF;
38
39          ident_sint.SP := setpoint;
40          ident_sint.RUN := TRUE;
41      ELSE
42          mv := ident_sint.MV;
43      END_IF;
44
45      (*Detecção de borda de descida no pino STATUS do bloco IDENT_SINT = sintonia finalizada*)
46      IF NOT vec_sta_ident[1] AND vec_sta_ident[2] THEN
47          IF err_ident = 0 THEN (*Se o bloco IDENT_SINT não retornou erros*)
48              k[exec_atual] := ident_sint.K;
49              t[exec_atual] := ident_sint.T;
50              l[exec_atual] := ident_sint.L;
51              kp_pi[exec_atual] := ident_sint.KP;
52              ti_pi[exec_atual] := ident_sint.TI;
53              ident_sint.RUN := FALSE;
54
55              exec_atual := exec_atual+1;
56          ELSE
57              err := err OR err_ident; (*O erro do bloco atual recebe os próprios erros mais os erros
    do bloco IDENT_SINT*)
58          END_IF;
59      END_IF;
60  END_IF;
61
62  (*Atribuição dos parâmetros da planta aos pinos correspondentes*)
63  K1 := k[1];
64  T1 := t[1];
65  L1 := l[1];
66
67  K2 := k[2];
68  T2 := t[2];
69  L2 := l[2];
70
71  K3 := k[3];
72  T3 := t[3];
73  L3 := l[3];
74
75  K4 := k[4];
76  T4 := t[4];
77  L4 := l[4];
78
79  (*Atribuição dos parâmetros do controlador aos pinos correspondentes*)
80  (*tc representa o tipo de controlador. Se tc = 0, é um PI. Se tc = 1, é um PID.*)
```

```
IF NOT tc THEN
    KP1 := kp_pi[1];
    TI1 := ti_pi[1];
    TD1 := 0.0;

    KP2 := kp_pi[2];
    TI2 := ti_pi[2];
    TD2 := 0.0;

    KP3 := kp_pi[3];
    TI3 := ti_pi[3];
    TD3 := 0.0;

    KP4 := kp_pi[4];
    TI4 := ti_pi[4];
    TD4 := 0.0;
ELSE
    KP1 := kp_pid[1];
    TI1 := ti_pid[1];
    TD1 := td_pid[1];

    KP2 := kp_pid[2];
    TI2 := ti_pid[2];
    TD2 := td_pid[2];

    KP3 := kp_pid[3];
    TI3 := ti_pid[3];
    TD3 := td_pid[3];

    KP4 := kp_pid[4];
    TI4 := ti_pid[4];
    TD4 := td_pid[4];
END_IF;
```

Programa 2. Código fonte do bloco N_IDENT_SINT.

O bloco GAIN_SCHED é apresentado na Figura A.3. Ele é responsável por fazer o escalonamento de ganhos através da Equação (17), em que os parâmetros obtidos na etapa de sintonia do controlador são linearizados de forma a permitir sua variação em função da PV. O Programa 3 apresenta o código fonte do bloco.

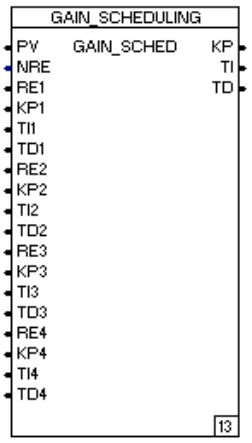

Figura A.3. Bloco GAIN_SCHED.

```
(*NRE é o número de regiões informado pelo usuário*)
IF NRE <= 1 THEN
    KP := KP1;
    TI := TI1;
    TD := TD1;

ELSIF NRE = 2 THEN
    IF PV < RE1 THEN
        KP := KP1;
```

```
10          TI := TI1;
11          TD := TD1;
12      ELSIF PV >= RE1 AND PV < RE2 THEN
13          KP := (KP1*(RE2 - PV) + KP2*(PV - RE1))/(RE2 - RE1);
14          TI := (TI1*(RE2 - PV) + TI2*(PV - RE1))/(RE2 - RE1);
15          TD := (TD1*(RE2 - PV) + TD2*(PV - RE1))/(RE2 - RE1);
16      ELSIF PV >= RE2 THEN
17          KP := KP2;
18          TI := TI2;
19          TD := TD2;
20      END_IF;
21
22  ELSIF NRE = 3 THEN
23      IF PV < RE1 THEN
24          KP := KP1;
25          TI := TI1;
26          TD := TD1;
27      ELSIF PV >= RE1 AND PV < RE2 THEN
28          KP := (KP1*(RE2 - PV) + KP2*(PV - RE1))/(RE2 - RE1);
29          TI := (TI1*(RE2 - PV) + TI2*(PV - RE1))/(RE2 - RE1);
30          TD := (TD1*(RE2 - PV) + TD2*(PV - RE1))/(RE2 - RE1);
31      ELSIF PV >= RE2 AND PV < RE3 THEN
32          KP := (KP2*(RE3 - PV) + KP3*(PV - RE2))/(RE3 - RE2);
33          TI := (TI2*(RE3 - PV) + TI3*(PV - RE2))/(RE3 - RE2);
34          TD := (TD2*(RE3 - PV) + TD3*(PV - RE2))/(RE3 - RE2);
35      ELSIF PV >= RE3 THEN
36          KP := KP3;
37          TI := TI3;
38          TD := TD3;
39      END_IF;
40
41  ELSIF NRE >= 4 THEN
42      IF PV < RE1 THEN
43          KP := KP1;
44          TI := TI1;
45          TD := TD1;
46      ELSIF PV >= RE1 AND PV < RE2 THEN
47          KP := (KP1*(RE2 - PV) + KP2*(PV - RE1))/(RE2 - RE1);
48          TI := (TI1*(RE2 - PV) + TI2*(PV - RE1))/(RE2 - RE1);
49          TD := (TD1*(RE2 - PV) + TD2*(PV - RE1))/(RE2 - RE1);
50      ELSIF PV >= RE2 AND PV < RE3 THEN
51          KP := (KP2*(RE3 - PV) + KP3*(PV - RE2))/(RE3 - RE2);
52          TI := (TI2*(RE3 - PV) + TI3*(PV - RE2))/(RE3 - RE2);
53          TD := (TD2*(RE3 - PV) + TD3*(PV - RE2))/(RE3 - RE2);
54      ELSIF PV >= RE3 AND PV < RE4 THEN
55          KP := (KP3*(RE4 - PV) + KP4*(PV - RE3))/(RE4 - RE3);
56          TI := (TI3*(RE4 - PV) + TI4*(PV - RE3))/(RE4 - RE3);
57          TD := (TD3*(RE4 - PV) + TD4*(PV - RE3))/(RE4 - RE3);
58      ELSIF PV >= RE4 THEN
59          KP := KP4;
60          TI := TI4;
61          TD := TD4;
62      END_IF;
63  END_IF;
```

Programa 3. Código fonte do bloco GAIN_SCHED.

A Figura A.4 apresenta o bloco PID_WINDUP, que foi desenvolvido para implementar o controlador com *anti-windup* e ponderação do *setpoint* nas parcelas Proporcional e Derivativa.

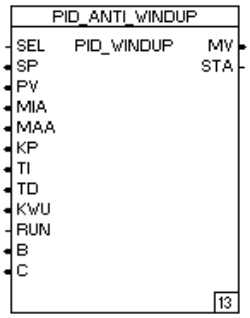

Figura A.4. Bloco PID_WINDUP.

Neste bloco são calculados os erros do saturador e das ações proporcional, integral e derivativa. A partir disso, calcula-se o sinal de controle, aplicando-o na MV. O Programa 4 mostra o seu código fonte.

```
erro_p := B*sp - pv;          (*Erro da ação proporcional*)

IF primeira_exec THEN
    erro := 0.0;
    erro_s := 0.0;
ELSE
    erro := sp - pv;          (*Erro da ação integral*)
    erro_s := uc_sat - uc;    (*Erro do saturador do sinal de controle*)
END_IF;

(*vec_erro_d: vetor REAL com o erro da ação derivativa. A posição [1] contém o valor atual, e a posição [2] contém o valor anterior.*)
vec_erro_d[2] := vec_erro_d[1];
vec_erro_d[1] := C*sp - pv;

(*Cálculo da ação proporcional*)
p := kp*erro_p;

(*Cálculo da ação integral (backward)*)
i := i + ta*((kp/ti)*erro + kwu*erro_s);

(*Evitando que o integrador fique negativo*)
IF i < 0.0 THEN
    i := 0.0;
END_IF;

(*Cálculo da ação derivativa (backward)*)
d := (td/(td + N*ta))*d + (kp*td*N/(td + N*ta))*(vec_erro_d[1] - vec_erro_d[2]);

(*Cálculo do sinal de controle*)
uc := p + i + d;

(*Saturando o sinal de controle*)
IF uc >= MIA AND uc <= MAA THEN
        uc_sat := uc;
    ELSIF uc < MIA THEN
        uc_sat := MIA;
    ELSE
        uc_sat := MAA;
END_IF;

(*Aplicação do sinal de controle*)
MV := uc_sat;
```

Programa 4. Código fonte do bloco PID_WINDUP.