Automação em Tempo Real: Trabalho Final

Universidade Federal de Minas Gerais

Júlia de Backer Pacífico — 2019021476

Junho de 2022

I. Introdução

O presente trabalho propõe uma situação de controle clássica, ilustrada pela Figura 1, em que um tanque, cuja área de secção transversal varia com a altura, é preenchido por um líquido.

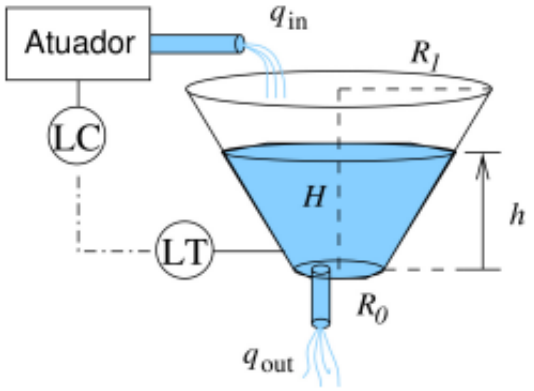

Figura 1: *Processo industrial simulado*

Nesse processo, o nível do tanque *h(t)* é fornecido por um transmissor de nível **LT**. Por outro lado, a entrada de líquido no tanque $q_{in}(t)$ é governada por meio de um atuador, que por sua vez é regulado por um controlador de nível **LC**. Em contrapartida, a saída do tanque é expressa por

$$q_{out}(t) = C_v\sqrt{h}, \tag{1}$$

em que C_v coeficiente de descarga da saída do tanque.

Portanto, a dinâmica não linear do tanque segue a equação (2) mostrada abaixo:

$$h(t) = \frac{-C_v\sqrt{h(t)}}{\pi[R_0 + \alpha h(t)]^2} + \frac{u(t)}{\pi[R_0 + \alpha h(t)]^2}, \tag{2}$$

onde $u(t) = q_{in}(t)$ e $\alpha = \frac{R_1 - R_0}{H}$.

Dessa forma, foi projetado um sistema de tempo real que simula a situação exposta acima em um programa em Python. Para isso, foi estabelecido uma comunicação por protocolo tcp/ip entre dois programas, seguindo a arquitetura cliente-servidor: um deles simula a planta de controle (o cliente) e o outro representa um painel sinóptico (o servidor).

O programa cliente ("plant.py") possui três threads: uma para simulação da dinâmica do tanque, outra para a simulação de um controlador PID, e ainda uma última para a coordenação do período de simulação de cada uma das demais threads (50ms para a thread da dinâmica tanque e 25ms para a thread de controle). Já o programa servidor ("synoptic.py"), ao se conectar com o cliente, requisita um valor de entrada como referência à altura que se deseja manter no tanque cônico pelo método de controle.

Aferido um valor desejado para a altura h, o sistema de controle é iniciado no programa cliente. Durante a simulação de tempo limitado (neste trabalho, imposta para 15 segundos de duração), o programa cliente envia os valores das vazões de entrada e saída para o processo sinóptico, junto da altura h atual

II. Simulação da planta de controle

II.1. Modelagem do tanque cônico

Com base na representação de diagrama de blocos, pode-se representar a dinâmica do tanque por um subsistema em malha aberta cuja entrada e saída sejam dadas, respectivamente, pelas vazões q_{in} e q_{out}. Dessa forma, a função de transferência para o bloco da Figura 2 se torna a própria função $h(t)$, exposta na Equação 2.

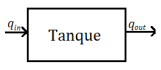

Figura 2: *Dinâmica do tanque*

Assim, o processo em malha aberta pode ser representado pelo diagrama de blocos da Figura 3.

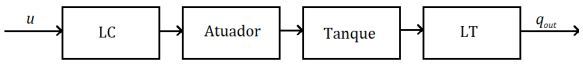

Figura 3: *Processo sem o sistema de controle*

Dessa forma, nota-se que o processo aborda um sistema dinâmico não-linear de 4ª ordem. Por essa razão, para simular a dinâmica do tanque neste trabalho, utilizou-se do método Runge-Kutta de 4ª ordem para solução da equação diferencial da dinâmica do tanque (Equação 2).

Os parâmetros do tanque modelado se encontram na tabela abaixo:

Parâmetro	Valor
Cv	0,75
H	10 m
R0	1 m
R1	2 m

Assim, a *process_thread* criada no programa cliente é encarregada por simular a dinâmica do tanque conforme as equações descritas. Dessa forma, ao acessar sua seção crítca, ela retorna novos valores para a altura relativa h do tanque e para a vazão de saída q_{out}, ambos em função da vazão de entrada q_{in} manipulada pela $softPLC\_thread$.

II.2. Sistema de controle e comunicação via socket

Para controle da planta representada pelo tanque cônico, o controlador projetato consiste em um simples PID. Sua implementação se deu pela biblioteca "simple-PID"e seus parâmetros foram definidos por aproximação segundo o método de Tustin.

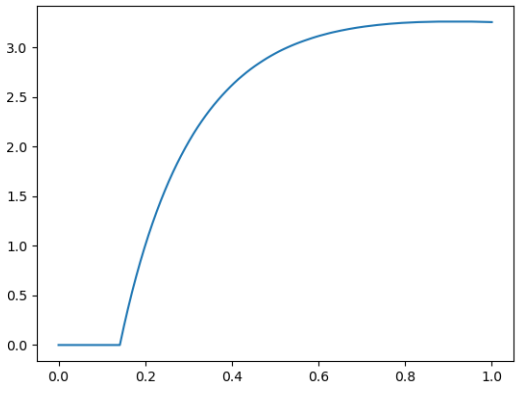

Figura 4: *Simulação do controlador com referência $h = 3m$*

Portanto, ao entrar em sua seção crítica,a $softPLC\_thread$ retorna um novo valor para a vazão de entrada q_{in}, a partir da altura relativa h do tanque, regida pela *process_thread*.

Além disso, logo que iniciada a execução do programa cliente, a $softPLC\_thread$ também é responsável por solicitar comunicação via socket com o programa servidor. Estabelecida a comunicação entre processos, ela então recebe um valor de referência desejado para a altura h e o processo de controle é iniciado.

II.3. Implementação da planta de controle em tempo real

No programa cliente, que representa a planta de controle do processo em questão, apenas uma thread por vez pode alterar as variáveis globais h, q_{in} e q_{out}. Por essa razão, o acesso simultâneo a qualquer uma delas é protegido por um mutex. Dessa forma, ilustrando um problema de característica "consumidor x produtor", a $softPLC\_thread$ só executa sua seção crítica quando a *process_thread* não tiver a posse do mutex, e vice-versa.

Além disso, para definir o tempo de execução de ambas as threads, foi implementado um sistema de eventos no programa cliente. Dessa forma, um evento foi configurado para ser habilitado conforme o período de simulação de cada thread — portanto, o evento responsável por habilitar a $softPLC\_thread$ é ativado a cada 25ms; já o evento que habilita a *process_thread*, a cada 50ms.

A alternância entre os eventos é executada por uma terceira thread: a *timers_thread*, cujo efeito prático é apenas garantir o período de execução das demais.

III. Simulação do sistema supervisório

O programa servidor deve ser o primeiro iniciado na simulação proposta neste trabalho. Logo que em execução, ele abre um socket para um servidor de endereço e porta fixos, conhecidos pelo cliente instanciado na planta de controle, e se comunica por protocolo IPv4.

Reconhecida a conexão do programa cliente, o *synoptic_process* — também responsável por instanciar o servidor — requisita do operador do sinóptico um valor h_{ref} desejado para a altura de controle. Após verificar que o valor de referência não ultrapassa a altura máxima $H = 10m$ do tanque, o sinóptico envia h_{ref} à planta de controle pelo socket e aguarda o recebimento das informações esperadas da planta de controle.

Enquanto a simulação deste sistema em tempo real estiver em curso, o socket é mantido aberto e o processo sinóp-

tico imprime os valores recebidos. Paralelamente, ele também verifica a existência de um arquivo "historiador.txt"no diretório de execução do programa. Caso não exista, o processo cria um novo arquivo de mesmo nome; do contrário, ele apenas abre o arquivo preexistente e concatena as informações advindas da planta de controle.

O processo mantém sua execução até que a comunicação seja encerrada pelo programa cliente, o que acontece decorridos os 15 segundos predefinidos como tempo total desta simulação. O valor para este tempo pode ser alterado em código e em nada influência quaisquer dos processos executados.

IV. Resultados e Conclusões

Os resultados obtidos em testes para diversos valores de entrada foram satisfatórios. Conforme ilustrado na Figura 4, o controlador implementado segue corretamente a referência imposta e, portanto, o valor final da altura relativa condiz com o desejado.

Adiante, a Figura 5 mostra a interface dos programas cliente e servidor, que simulam respectivamente a planta de controle e o sistema supervisório.

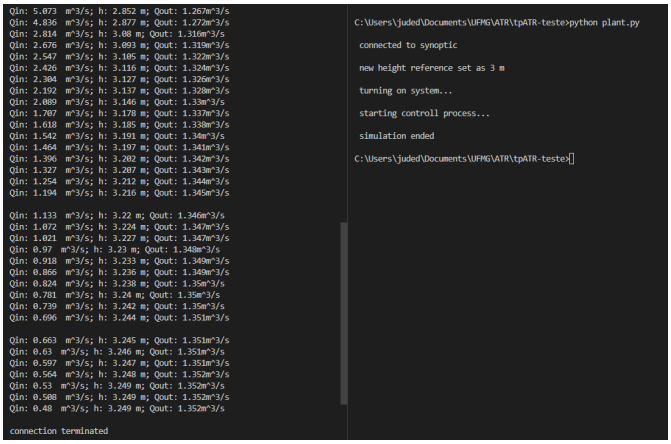

Figura 5: *Saída no prompt de comando*

Conclui-se, portanto, que a linguagem Python é eficaz na simulação de sistemas em tempo real, uma vez que dispõe de diversas ferramentas que viabilizam o uso de diretivas do sistema operacional no controle de variáveis compartilhadas. O valor final para a altura relativa, ainda que não seja exatamente igual ao fornecido como referência, também é considerado satisfatório, uma vez que o erro de aproximação se dá na ordem de casas decimais.

V. Fontes

V.1. "Runge-Kutta methods for ODE integration in Python"— github.com/Naereen/notebooks

V.2. "Implementing PID Controllers in Python"— pub.dev/packages/simple$_pid$