UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEP. DE ENGENHARIA ELETRÔNICA – ESCOLA DE ENGENHARIA

TRABALHO PRÁTICO DE ATR

Hiury Goulart Sant Ana

2016096750

21 de Setembro de 2020

Trabalho Prático Final de ATR (PARTE 1)

Processos concorrentes sendo controlados.

Neste trabalho será desenvolvido uma simulação para controle de processos concorrentes.

21 de Setembro de 2020

1 Introdução

O trabalho prático desenvolvido tem o objetivo de trazer conhecimento sobre a simulação de um processo industrial que visa controle simultâneo de processos paralelos. Esses processos serão controlados por meio de um controle PID.

2 Desenvolvimento e Metodologia

Para realizar esse trabalho, inicialmente, tem-se em mente que a concorrência é uma estratégia de desacoplamento que permite desacoplar o que é executado de quando é executado. Nesse sentido, é como se houvesse muitos minicomputadores trabalhando juntos do que um único e grande main().

Para efetuar as atividades concorrentes por meio de threads, houve declaração de variáveis controladas e de referências de forma global. Frequências menores foram executadas primeiro. Foram criados arquivos diferentes para escopo da main, variáveis e funções. Para sincronizar o acesso a quaisquer recursos usados paralelamente de forma eficiente, foi feito uma condição a tarefas. Isso garante que a thread espere até que determinada condição seja atendida e notifique outras threads sobre a condição sendo cumprida para que possam se desbloquear. No código implementa-se os dois casos, pois isso é condição para quando as threads dos tanques e controlador tenham acesso exclusivo aos recursos compartilhados.

Para que o trabalho cumpra com as necessidades demandadas, deve-se usar condiçoes, as quais prevêem threads de alterar um estado interno ao mesmo tempo. Para isso, o método de aquisição envia um sinalizador de espera opcional, que pode ser usado para evitar o bloqueio se ele for mantido por outra operação.

Sabe-se que uma condiçao esta sempre associada com um lock. Por isso, foi usado os metodos acquire() e release() que sao correspondentes associados ao lock. Foi usado try e finally para garantir que isso fosse feito pelo código. Outro metodo associado que foi chamado no codigo é o wait(), que solta o lock e bloqueia ate outra thread acordá-lo com outra chamada. Entretanto, ele esta relacionado a eventos do período de funcionamento das threads e por isso seu prefixo é diferente.

Além disso, inicialmente, fala-se sobre implementação do método Runge–Kutta, o qual resolve Equações Diferenciais Elementares de primeira ordem sem o cálculo de qualquer derivada, mas depende de outra função que é definida avaliando a função em diferentes pontos. Nele, por default, é preciso declarar 4 valores iniciais e, por isso, foi escolhido 4 valores arbitrários de vazão, altura, raio inferior e raio superior. Isso define a dinâmica não linear dos diferentes tanques do problema.

Para testagem de controle e verificaçao de qual tolerancia seria melhor, foi escolhido 0.00001 da tolerância com base em experimentações gráficas e a que gerou melhores valores senoidais foi escolhido. Essa tolerancia garante que haja uma exatidao relativa aos valores reais e ao valor que esta sendo controlado pelo atuador. Foi feita uma funçao para testagem que e' referenciada pelas threads.

Em segundo lugar, outra dica de implementação sugerida foi o uso de controladores PID, para um sistema com comportamento linearizado em torno de um ponto de operação preespecificado. Somado a isso, sabe-se que o controle PID é muito útil em aplicações que há pequenas variações na variável controlada. Esse tipo de controlador aceita parâmetros que afetam sua capacidade de resposta e, naturalmente, o quanto ultrapassa o ponto de ajuste. Para o caso de uso do Trabalho Prático, o controlador recebe como referências de nível h1 e h2 a ser seguida pela variável controlada, as quais são funções senoidais ao longo do tempo.

Nesse sentido, a resposta do loop determina o que precisa ser melhorado. Ao utilizar uma biblioteca python para simulação do controle PID, foi feito um diagrama (Figura 1) para melhor compreensão dos valores de referência do loop. Os parâmetros enviados ao PID são:

1. Coeficiente proporcional (KP=1) ao valor atual do erro. Se o erro for grande e positivo, a saída será grande e positiva, levando em consideração o fator de ganho K. É necessário um erro para gerar uma resposta proporcional.

2. Coeficiente integrativo (KI=0.1) considera os valores passados do erro e os integra ao longo do tempo para produzir o termo I. Esse parâmetro busca eliminar o erro de "steady-state". Usou-se o tempo do período dos tanques.

3. Coeficiente derivativo (KD=0.1) representa a melhor estimativa para a tendência futura, com base em sua taxa de mudança atual. Quanto mais rápida a mudança, maior será o efeito de controle ou amortecimento. Reduz "overshoot".

Os valores acima escolhidos foram 1/4 da frequência de funcionamento dos tanques e controlador.

4. Set point: valor constante a ser alcançado pela variável medida, a altura desejada para os tanques (default=0).

5. OutValue: o último valor de saída retornado do método de saída.

Para fazer a análise disso, sabe-se que o comportamento da função controlada é visto em gráfico. Por isso, usou-se plotagens para melhores que serao mostrados a seguir.

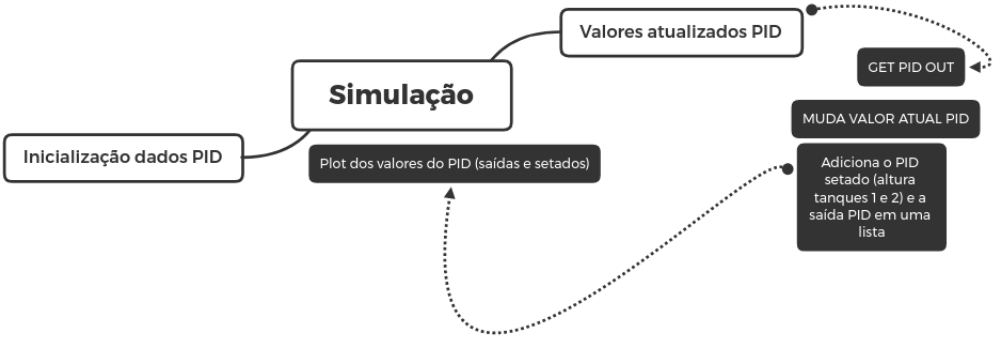

Figura 1: Diagrama para controlador PID em Python.

3 Código

Os programas foram testados em um computador Ubuntu 20.04, Acer intel core i5. Há um arquivo Makefile e readme.txt para compilação do programa e com instruções de compilação e operação do sistema.

Abaixo segue imagens relativas a plotagem do grafico que foi executada. Na figura 2, foi quando os valores iniciais estavam com condições mínimas para encher, menores valores em módulo no quesito de tolerância, e os valores das alturas a serem setadas estavam menores também ao comparar com os valores na Figura 3. Nesse sentido, o tempo de execução foi de 35.49 segundos.

Em relação à Figura 3, nota-se que o gráfico gerado possui mais controle. O programa foi deixado com os valores desse gráfico, pois a acurácia foi maior. A tolerância também foi menor. Entretanto, abdicou-se do tempo de execução, o qual demorou 498.71 segundos.

Em ambas as figuras observamos uma inclinação muito atenuada nos valores no início e logo depois, tem-se uma flutuação desempenhando-se controle baseada nos valores setados previamente.

4 Conclusão

Por meio do TP, foi possível adquirir muito conhecimento acerca de threads e controle entre dois sistemas diferentes, bem como o uso das condiçoes necessárias

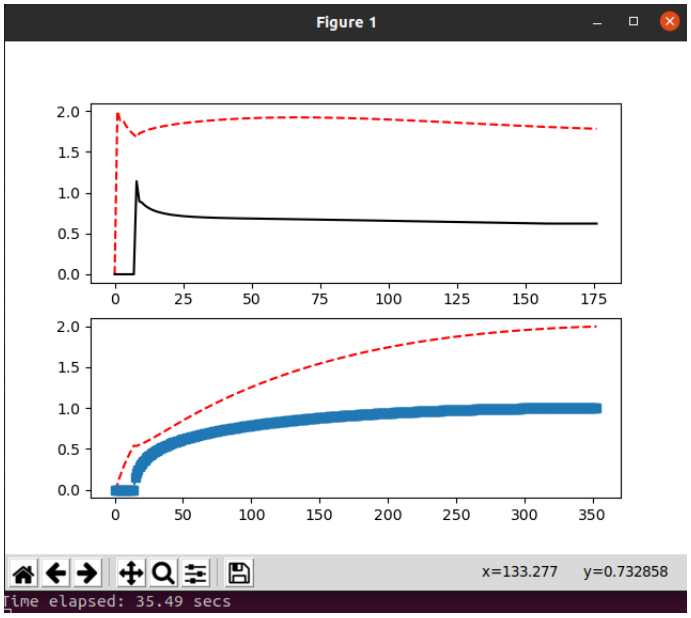

Figura 2: Print gerado 1.

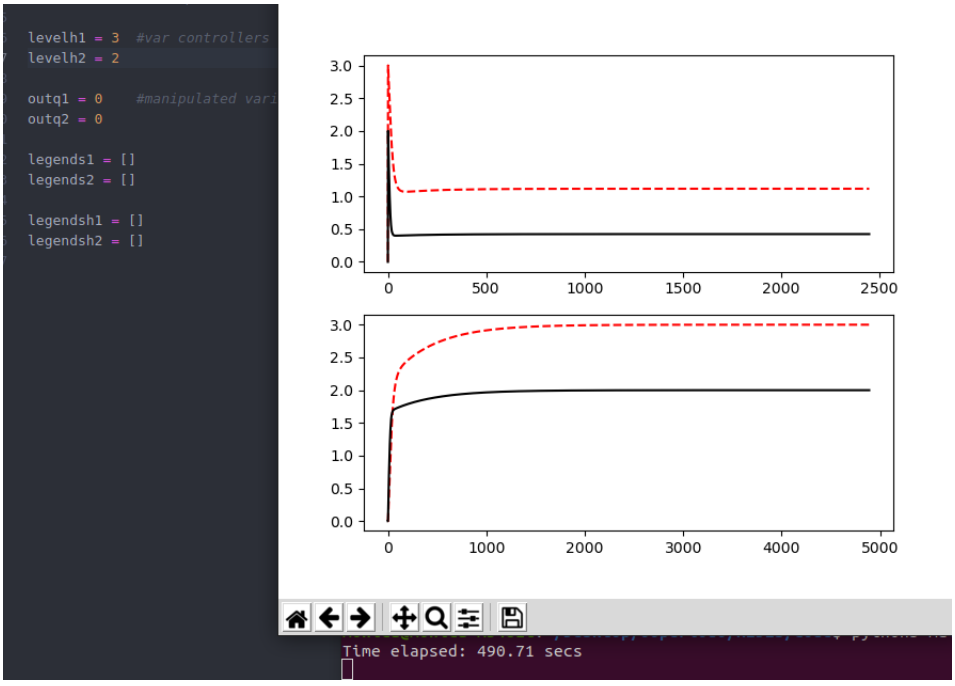

Figura 3: Print gerado 2. Como código final foi deixado.

para haver consumo de recursos eficientes. Além disso, a execução do trabalho permitiu visualizar que o python nao seria a melhor linguagem para aplicaçoes que exigem alto grau de confiabilidade, pois houve distorçoes de valores em algumas testagens. Isso tambem foi o motivo da separaçao entre numerador e denominador da dinamica dos processos dos tanques.

Por fim, abranger os graficos para visualizacao dos processos foi positivo. Alem de garantir que o trabalho estava desempenhando da forma correta.

Referências

http://en.wikipedia.org/wiki/PID$_{c}ontroller$
$https\ :\ //simple-pid.readthedocs.io/en/latest/simple_{p}id.htmlmodule-simple_{p}id.PID$