Uma metodologia para desenvolvimento de uma ferramenta para solução de um sistema real em linguagem python.

GUSTAVO BRAZ KURZ;
RÉGIS QUADROS; DANIELA BUSKE.
Universidade Federal de Pelotas – gustavobrk@gmail.com
Universidade Federal de Pelotas – quadros99@gmail.com
Universidade Federal de Pelotas – danielabuske@gmail.com

1. INTRODUÇÃO

A solução e interpretação de sistemas reais são alvos de estudos de diversas áreas. Para conseguir ter algum resultado e ser produtivo é necessário fazer o uso de ferramentas computacionais. Este trabalho tem como princípio iniciar um estudo sobre um sistema real e resolvê-lo computacionalmente utilizando a distribuição Anaconda Python que é voltada para computação científica e expor como foi feito passo a passo de todo o projeto, tendo em vista, que é um estudo da ferramenta, do sistema real e de como resolver esse sistema real.

Este trabalho tem como interesse demonstrar passo a passo a criação de uma ferramenta computacional para a resolução de um sistema real simplificado. Com isso queremos demonstrar nossas facilidades na escolha de uma distribuição da linguagem Python, que é usada para computação científica e também nossas dificuldades para a obtenção dos resultados. Partimos de um problema simples que é a mistura dentro de um tanque, de dois fluxos de água com diferentes temperaturas e fluxos. Queremos verificar qual a temperatura da água no tanque depois de um determinado tempo, possibilitando o usuário ver uma saída gráfica.

2. METODOLOGIA

2.1. Descrição do experimento do sistema real:

O sistema real que vamos simular em Python neste trabalho é esquematizado como na figura 1.

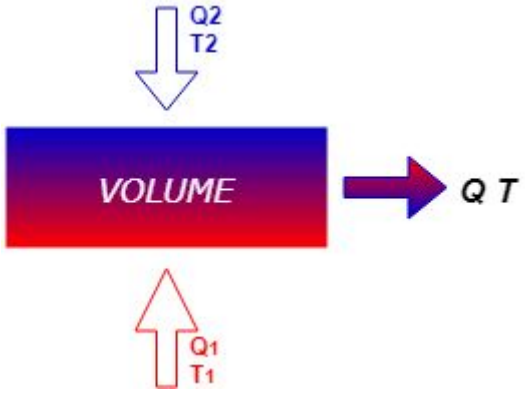

Figura 1.

A simulação consiste em um tanque de volume V, dois fluxos de água Q1 e Q2, com diferentes temperaturas T1 e T2. A água de descarga é dada pelo fluxo Q e temperatura T. Nesse sistema a mistura é perfeita de acordo com S. Hubalovsky (2012).

2.2. Desenvolvimento do modelo estático:

As propriedades estáticas do nosso sistema podem ser descritas pelas seguintes equações:

$$Q = Q_1 + Q_2 \quad [1]$$

$$QT = T_1 Q_1 + T_2 Q_2 \quad [2]$$

Sendo, de [1] e [2], temos Q_1, Q_2, T_1 e T_2 como entradas do sistema e Q, QT são sáidas. Para melhor exemplificar a equação a seguir nomearemos *SS* como "*sinal de saída*",*SE "sinal de entrada" e GA "como ganho de ação"* .

$$GA = \Delta SS / \Delta SE \quad [3]$$

De [3] se soubermos o valor de *GA* para determinado sistema regulado e requerido valor do sinal de saída, o valor de *SE*, que deve ser aplicado à entrada do sistema, pode ser calculado para atingir os valores de saída necessários.

2.3 Busca de um ambiente de desenvolvimento Python:

Para o início do desenvolvimento da ferramenta foi requisitado três características principais para o ambiente de programação: fácil visualização do código, integração com outras linguagens (Html, Julia, R entre outras) e ambiente de desenvolvimento com uma vertente OP (*Open Source*) que é um termo em inglês que significa código aberto. Isso também diz respeito ao código-fonte de um software, que pode ser adaptado para diferentes fins.
Como ambiente de desenvolvimento utilizaremos o Jupyter Notebook que é um aplicativo da web de código aberto que permite criar e compartilhar documentos que contêm código ativo, equações, visualizações e texto narrativo. Para melhor aproveitamento da linguagem python também utilizamos a distribuição Anaconda que nos possibilitou facilidade para incluir bibliotecas gráficas e de simulações numéricas, tais como, numpy, pyplot.

3. RESULTADOS E DISCUSSÃO

Para a demonstração dos resultados optamos por uma saída de dados (gráfica). Os sinais de entrada são imutáveis, tendo em vista que esse trabalho é para comparação com os resultados da literatura. É necessário ressaltar que trabalhamos com alocação dinâmica de memória.

O gráfico 1 abaixo mostra relação das temperaturas T, T_1, T_2:

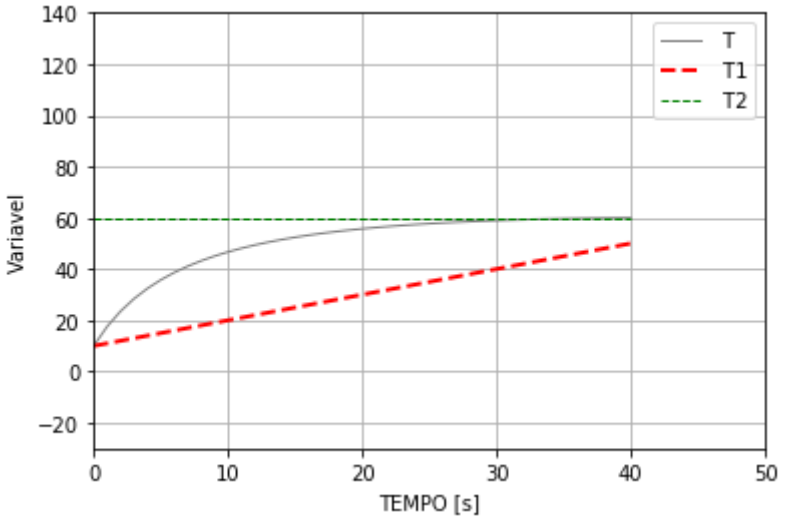

Gráfico 1

```
import matplotlib.pyplot as plt
plt.axis( [ 0 , 50 , -30 , 140 ] )
plt.plot( T,'gray',w=1 )
plt.plot( vectorTemperature1, 'r--', lw=2)
plt.plot( [0, 40], [60, 60],  'g--', lw=1)
plt.grid(True)
plt.xlabel( “TEMPO [min]" )
plt.ylabel( "Variavel" )
plt.show()
```

O gŕafico 2 demonstra a dependência que o fluxo tem em relação ao tempo:

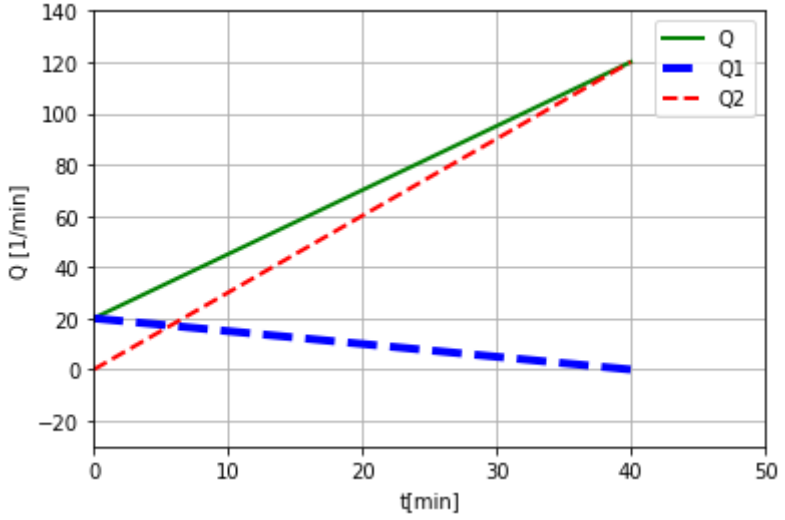

Gráfico 2

```
import matplotlib.pyplot as plt
plt.plot( Q,'green', lw=1, label =”Q”)
plt.plot( vectorFlowQ1, ’br--', lw= 4,label =”Q1”,)
plt.plot( vectorFlowQ2,  'r--', lw= 2 ,label =”Q2”)
plt.grid(True)
plt.xlabel( “Q [1/min]" )
plt.ylabel( "t [min]" )
plt.show()
```

4. CONCLUSÕES

Conseguimos obter os resultados esperados e assim dar continuidade no nosso trabalho. Através de um estudo do problema real e das ferramentas matemáticas necessárias para o resolver o sistema,obtivemos muita facilidade ao programar as equações em python devido a uso da biblioteca numpy. Para expor nossos resultado usamos a biblioteca pyplot que também é de fácil interpretação e nos possibilita utilizar alocação dinâmica de memória para produzir nossos gráficos variando conforme o tempo.

Além dessas vantagens conseguimos fazer uma interação das equações com as saídas/resultados em um mesmo programa, ou seja, não precisamos buscar um software A e depois outr B para assim conseguir resolver nosso problema, ou seja, a ideia principal da ferramenta , aumentar o conhecimento em python, reunir diversos mecanismos matemáticos e os resolver em um mesmo programa,foi atingida com sucesso.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Hubalovsky, S. Mixing of two different temperature water flows as feedback controlled system mathematically modeled and simulated in MS Excel spreadsheet. **INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTERS IN SIMULATION**, v 6, n.1, p. 46 - 57, 2012.

Aroca, R.V. **Análise de Sistemas Operacionais de Tempo Real para Aplicações de Robótica e Automação**. 2008. Dissertação- Departamento de Engenharia Mecânica da Escola de Engenharia de São Carlos da Universidade de São Paulo.