INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Victor Jucá Martins

UTILIZAÇÃO DO MÉTODO DOS ELEMENTOS FINITOS EM APLICATIVO DE RESOLUÇÃO DO BALANCEAMENTO DE REDES HIDRÁULICAS MALHADAS MULTI-INPUT

Trabalho de Graduação
2017

Curso de Engenharia Civil-Aeronáutica

CDU: 628.10

Victor Jucá Martins

Orientador

Ten Cel Eng Márcio Antônio da Silva Pimentel – PhD (ITA)

ENGENHARIA CIVIL-AERONÁUTICA

SÃO JOSÉ DOS CAMPOS
INSTITUTO TECNOLÓGICO DE AERONÁUTICA

2017

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Martins, Victor
Utilização do método dos elementos finitos em aplicativo de resolução do balanceamento de redes hidráulicas malhadas multi-input / Victor Jucá.
São José dos Campos, 2017
Número de folhas no formato 59f.

Trabalho de Graduação – Engenharia Civil-Aeronáutica – Instituto Tecnológico de Aeronáutica, 2017. Orientador: Ten Cel Eng Márcio Antônio da Silva Pimentel – PhD.

1 Dimensionamento. 2. Redes de Abastecimento. 3. Cálculo automático. I. II. Instituto Tecnológico de Aeronáutica. III. Utilização do método dos elementos finitos em aplicativo de resolução do balanceamento de redes hidráulicas malhadas *multi-input*

REFERÊNCIA BIBLIOGRÁFICA

MARTINS, Victor. Utilização do Método dos Elementos Finitos em Aplicativo de Resolução do Balanceamento de Redes Hidráulicas Malhadas *Multi-input*. 2017. 59f. Trabalho de Conclusão de Curso. (Graduação em Engenharia Civil-Aeronáutica) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Victor Jucá Martins
TÍTULO DO TRABALHO: Utilização do Método dos Elementos Finitos em Aplicativo de Resolução do Balanceamento de Redes Hidráulicas Malhadas *Multi-input*.
TIPO DO TRABALHO/ANO: Graduação / 2017

Victor Jucá Martins
Rua Leopoldo Couto de Magalhães
Júnior,1442, apto 71- Itaim Bibi.
CEP: 04542-001, São Paulo – SP – Brasil.

UTILIZAÇÃO DO MÉTODO DOS ELEMENTOS FINITOS EM APLICATIVO DE RESOLUÇÃO DO BALANCEAMENTO DE REDES HIDRÁULICAS MALHADAS MULTI-INPUT

Essa publicação foi aceita como Relatório Final de Trabalho de Graduação

Victor Jucá Martins
Autor

Ten Cel Eng Márcio Antônio da Silva Pimentel – PhD (ITA)
Orientador

Prof. Dr. Eliseu Lucena Neto (ITA)
Coordenador do Curso de Engenharia Civil-Aeronáutica

São José dos Campos, 22 de novembro de 2017

Dedico este trabalho a Deus.

Agradecimentos

Agradeço a todos que de modo especial contribuirão para a realização deste trabalho.

Aos meus pais por todo o amor e cuidado que tiveram por toda a minha vida.

Aos meus orientadores, Cel Márcio e Prof. Paulo Ivo que tiveram a paciência para guiar os meus passos.

Á Aline por ter compreendido a dedicação que esta atividade exigiu e pelo apoio nas horas difíceis.

Aos meus amigos que contribuíram para tornar a jornada durante a graduação menos exaustiva.

Resumo

O principal componente de um sistema de abastecimento de água potável é a sua rede de distribuição. É conveniente que esta seja planejada visando a redundância de fluxo por meio da utilização de redes malhadas. O dimensionamento destas, porém, depende da resolução de um sistema de equações não-lineares. O método de resolução padrão presente na literatura, Hardy-Cross, apresenta ineficiências em termos de execução e convergência. O *software* desenvolvido se propõe a resolver esses problemas a partir da aplicação de metodologia específica, capaz de processar redes genéricas e com mais de um reservatório, sem a necessidade de fornecimento de solução inicial nem de identificação manual da localização de ciclos na rede. A aplicação utiliza o motor de cálculo da linguagem de programação Python, a partir da interface popular do Microsoft Excel. Desta maneira, a procura por soluções econômicas para implantação e reforma dos sistemas de abastecimento se tornam rápidas e fáceis.

Abstract

The main component of a potable water supply system is its distribution network. It is desirable that this be planned for redundancy of flow using looped networks. Their sizing, however, rely on a resolution of a system of nonlinear equations. The standard resolution method shown in the literature, Hardy-Cross, has inefficiencies in terms of execution and convergence. The developed software proposes to solve these problems by applying a specific methodology, capable of processing generic networks and with more than one reservoir, without the need of providing an initial solution or manually identifying the location of loops in the network. The application uses the calculation engine of the Python programming language and the popular Microsoft Excel's interface. In this way, the search for economic solutions for the implementation and reform of the supply systems becomes fast and easy.

Sumário

1 – Objetivo

O advento das modernas soluções tecnológicas de automação de processos vem modificando sobremaneira a relação do homem com o trabalho. De maneira geral, pode-se enxergar que as atividades mais repetitivas estão sendo substituída por máquinas. Estas ainda não conseguem atingir um nível de inteligência capaz de substituir um trabalho de alta demanda criativa como o de um Engenheiro, entretanto estão a ajudá-los nas atividades mais exaustivas que compõem o seu trabalho.

Com vista para esse cenário, o objetivo deste trabalho é desenvolver um software que auxilie o dimensionamento de redes de abastecimento de água no cálculo dos parâmetros de controle de redes malhadas em regime. O cálculo destes parâmetros, pressão e vazão, é dado a partir de equações não-lineares cujos métodos de resolução padrões são bastante demandantes.

O cálculo do dimensionamento das redes é constituído de um método iterativo de resoluções do problema proposto de pressões e vazões em regime. Dadas as limitações que compõe o escopo deste trabalho, não será fornecida uma solução integral para o dimensionamento totalmente automatizado deste trabalho. Entende-se que as escolhas dos diâmetros dos dutos que compõe uma rede é um processo altamente complexo que passa por crivos de natureza prática e econômica, entretanto, ao final do trabalho, será proposto métodos para resolução deste problema.

2 – Introdução

Rede de distribuição é a parte do sistema de abastecimento de água potável de uma cidade formada de tubulações e órgãos acessórios, destinados a por água potável à disposição dos consumidores, de forma contínua, em quantidade, qualidade, e pressão adequadas.

É a parte mais onerosa da distribuição, representando cerca de 50 a 75% total dos custos de implementação da distribuição.

Os diâmetros dos dutos mais à montante do fluxo de abastecimento são consideravelmente maiores do que os dos dutos mais à jusante, bem como a vazão que passa por eles. Assim, convencionou-se classificar a parte da rede mais próxima ao fornecimento de água para o sistema de rede principal e a parte da rede mais próxima ao consumidor de rede secundária.

A tubulação principal, que também pode ser denominada conduto tronco ou ainda canalização mestra, são tubulações de maior diâmetro que têm por função abastecer as canalizações secundárias. A rede secundária ou canalização secundária são tubulações de menor diâmetro e têm a função de abastecer diretamente os pontos de consumo do sistema de abastecimento de água.

Enquanto a rede secundária possui grande importância no projeto de instalações hidráulicas residenciais, a rede principal ganha ênfase no sistema de saneamento ambiental.

A rede principal pode ser dividida em dois tipos:

a) Ramificada.
   Possui configuração do tipo árvore, em que não existe pontos de junção de fluxo.
b) Malhada.
   Possui pontos de junção de fluxo que caracterizam circuitos fechados no traçado da rede.

O Dimensionamento dos dois tipos de redes é realizado de maneira diferente. Entretanto, pode-se aplicar a mesma metodologia empregada em redes malhadas em redes ramificadas.

Para efetuar este dimensionamento é preciso atender ao que é imposto.

A pressão estática máxima nas tubulações distribuidoras deve ser de 500 kPa, e a pressão dinâmica mínima, de 100 kPa, item 5.4.1 da NBR 12218 (ABNT, 2017).

A velocidade mínima nas tubulações deve ser de 0,6 m/s, e a máxima, de 3,5 m/s; estes limites referem-se às demandas máximas diárias no início e no final da etapa de execução da rede, item 5.7.1 da NBR 12218 (ABNT, 2017).

O dimensionamento dos circuitos fechados, formados de condutos principais, e a análise do funcionamento global da rede devem ser realizados por métodos de cálculo iterativos, que garantam resíduos máximos de vazão e de carga piezométrica de 0,1 l/s e 0,5 kPa, respectivamente, item 5.7.4 da NBR 12218 (ABNT, 2017).

3 – Revisão Bibliográfica

No meio profissional, existem vários programas disponíveis para o projeto de redes de abastecimento que, entre outras funcionalidades, fornecem o dimensionamento de redes malhadas a partir de inputs específicos.

Dentre eles destaca-se o EPANET 2.0, *software* de código aberto distribuído gratuitamente pela agência de proteção ambiental dos Estados Unidos, *United States Evironmental Protection Agency*– EPA.

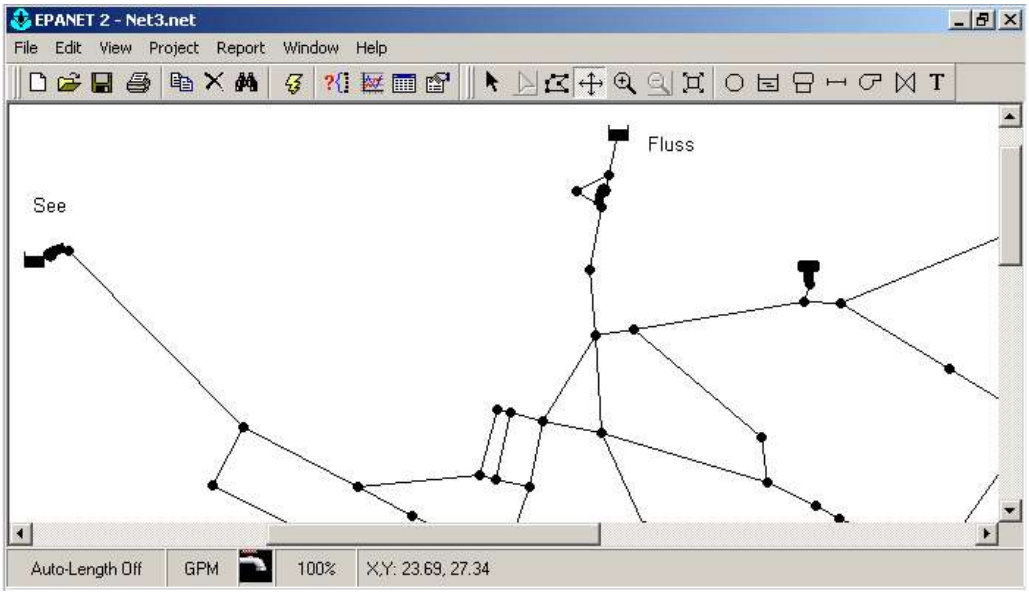


Figura 1- Tela do programa EPANET 2.0

Existem também outros programas que se dispõem a oferecer melhores implementações e/ou melhores suportes, mas que não possuem licença livre.

Como exemplo, podemos citar o software AFT FAThom, que trabalha com uma portfolio bem diversificado de fluidos, incluindo gases à baixa velocidade.

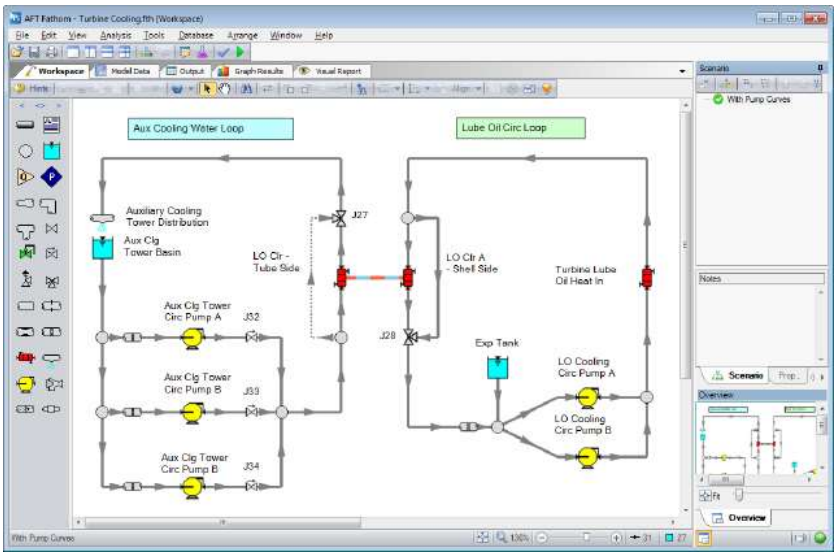


Figura 2 - Tela do programa AFT FAThom

Existe também empresas brasileiras capazes de oferecer esse tipo de produto. Dentre as quais, destaca-se a Multiplus que possui uma solução integrada com softwares CAD além de oferecer treinamento para o uso de seus programas.

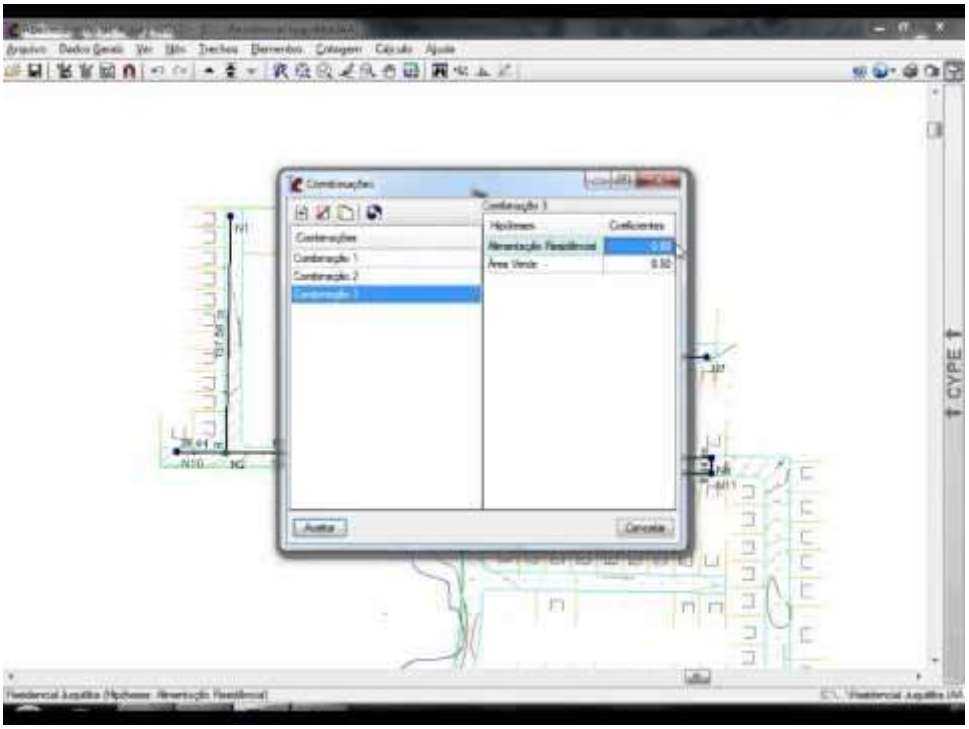


Figura 3 - Tela do software de abastecimento da Multiplus

4 – Embasamento teórico

Uma rede malhada pode ser enxergada como um grafo. Os dutos são representados pelas arestas e os pontos de união de dois ou mais dutos, bem como os pontos ligados aos reservatórios e os pontos de saída para a rede secundária são mostrados como nós.

Dessa maneira, pode-se extrair as informações necessárias para a modelagem do programa de balanceamento da rede, que se resumirá em um sistema de equações não-lineares.

Para uma caracterização desse grafo útil à modelagem do problema, deve-se adicionar atributos que representam as características de cada grafo em especial. São elas a pressão manométrica e a vazão em cada um dos nós e trechos, que representam as variáveis do sistema de equações não lineares.

As equações do sistema são extraídas a partir das condições de contorno física que a rede deve satisfazer.

A primeira destas restrições diz respeito à perda de carga devido ao fluxo no interior dos dutos.

$$P_i - P_j = H_{ij}$$

Em que,

H_{ij} é a perda de carga entre os nós i e j

P_i é a carga piezométrica no nó i.

Esta perda de carga também se relaciona com a vazão, de modo que se pode encontrar uma relação entre a carga piezométrica dos nós e a vazão do trecho que os conectam.

A formula que relaciona perda de carga e vazão assume vários formatos na literatra,

onde a mais usada para os diâmetros superiores à 50 mm é a formula de Hazen-Willians (TSUTIYA, 2006):

$$H = \frac{Q^{1,85}.L}{(0{,}2785.C)^{1,85}.D^{4,87}}$$

Em que,

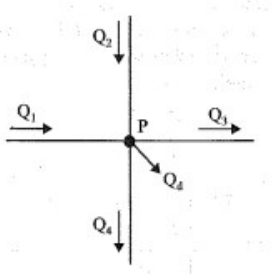
Q é a vazão em m^3/s.

L é comprimento do trecho em metros.

C é um coeficiente específico para cada material, chamado de coeficiente de Hazen-Willians.

D é o diâmetro interno, em metros, do duto que compõe o trecho, suposto constante.

A segunda diz respeito à condutividade ou conservação da massa. Para cada um dos nós:



Fonte: TSUTIYA

Figura 4- exemplo de fluxo de vazões

O somatório da vazão que chega em determinado nó por meio de outro nó ou reservatório deve ser igual à vasão que sai desse nó para outros nós ou para o abastecimento da população.

$$\sum Q_{in} = 0$$

Em que,

Q_{in}é a vazão que entra, positiva, ou que sai, negativa, em cada um dos nós.

Assim, percebe-se a não linearidade do sistema uma vez que a vazão na restrição anterior apresenta um expoente diferente de 1 e nesta apresenta o expoente 1.

A terceira diz respeito à pressão nos nós que estão conectados diretamente aos reservatórios.

$$p_i = p_{Ri}$$

Em que,

p_i, é a carga piezométrica no nó i.

P_{ri}é a pressão piezométrica do reservatório ligado ao no i que é igual à cota do nível da água do reservatório.

Com isso, é possível aplicar vários métodos para a resolução desta não linearidade de onde se escolhe o método proposto por P. I. B. Q. (2005).

Nesse método iterativo, aplica-se a seguinte linearização das variáveis do sistema.

$$Q_{ij} = \left(P_i - P_j\right).\left|P_i' - P_j'\right|^{\frac{1}{1,85}-1}.\left(\frac{D^{4,87}}{0{,}0021.L}\right)^{\frac{1}{1,85}}$$

Em que,

P_i' e P_j' são as pressões calculadas em uma iteração anterior.

Como a solução, quando existente, em sistemas de equações lineares é única, é possível reduzir a dimensão do problema ao usar apenas as pressões piezométricas como variáveis primitivas e embutir a condição de contorno 1 no cálculo das condições de contorno 2.

Entretanto, ao utilizar a condição de contorno 2 aos nós ligados à reservatórios, encontra-se uma variável não especificada até então: A vazão que escapa do reservatório

para o nó. Desta forma, é conveniente a utilização da condição de contorno 3 justamente nos nós que estão ligados à reservatórios.

Assim, se chega a um sistema de equações lineares com um número igual de variáveis e equações que pode ser resolvido facilmente por vários métodos matemáticos como a regra de Cramer, por exemplo.

Em resumo, aplicou-se uma linearização para que pudesse eliminar um número igual de variáveis e equações iguais ao número de trechos do sistema, entretanto, teve-se de se recorrer a um método iterativo que não possui convergência garantida.

É importante observar que poderia ser utilizada as vazões como variáveis primárias e não as pressões como faz TSUTIYA (2006). Entretanto, este método só funciona para o caso de um reservatório, já que neste caso é possível arbitrar o valor da vazão que entra no sistema pelo nó ligado ao reservatório, igual ao valor que sai do sistema para abastecimento, o que não pode ser feito para mais de um reservatório.

5 – Ferramentas

5.1 – Python

Python é uma linguagem de programação que está sendo cada vez mais usada. Particularmente, por causa principalmente dos módulos NumPy e SciPy esta linguagem vem ganhado destaque no meio acadêmica para pesquisas sobre programas já consagrados como MATLAB por exemplo.

5.2 – Numpy

Numpy é a principal ferramenta para computação científica para se usar em Python. Esta biblioteca possui uma série de funções implementadas para manipulação de vetores multidimensionais. Neste trabalho usaremos o Numpy para fornecer um método otimizado para solução de sistemas lineares, fornecendo assim, bastante confiabilidade à esta etapa dos cálculos.

5.3 – Microsoft Excel

Tendo em vista a difusão e a facilidade de aprendizado de uma nova ferramenta, é interessante se pensar em integralidade. Isto pode reduzir até mesmo uma possível barreira cultural de uso de um novo sistema.

Assim, tendo em vista a grande penetração que essa ferramenta tem tanto nos meios estudantis quanto no meio coorporativo, torna-se interessante a utilização desse ecossistema. Dessa forma, ganha-se não somente uma interface gráfica popular para o uso de um programa, mas acesso facilitado à todas as ferramentas que integram o pacote Office, como o Word por exemplo.

5.4 – xlwings

O módulo escrito para Python “xlwings” é a solução mais integrada com o Microsoft Excel disponível no momento (YEGULALP, 2014). Ele é usado em conjunto com a linguagem VBA para que se crie um serviço em python executado em paralelo às planilhas capaz de receber comandos e enviar respostas.

Neste trabalho, usaremos a solução apenas para implementar UDF’s, *Users Defineble functions*, disponíveis diretamente na pasta de trabalho.

6 – Desenvolvimento

A implementação dos métodos de resolução numéricos depende fortemente da plataforma utilizada. Como o objetivo deste trabalho é a construção de uma ferramenta operacional, capaz inclusive de ganhar melhorias e soluções numéricas diferentes, teve-se de se recorrer a soluções computacionais que permitissem esse tipo de construção. Por isso, foi definido com paradigma a Programação Orientada à Objeto – POO.

Assim, encontramos no código classes de objetos que representam todos os elementos relativos a modelagem e a solução do problema proposto. Cada classe possui atributos que correspondem à parâmetros da realidade física do sistema de abastecimento ou variáveis de controle da resolução. Cada classe possui também métodos que são responsáveis pela comunicação entre os diferentes objetos e a pela própria dinâmica de solução.

Além disso, optou-se por dividir a implementação do sistema em três módulos para melhor compreensão do código e possíveis intervenções. Além disso, tendo em vista uma possível comercialização e consequente manutenção deste software, é possível que duas ou até três pessoas trabalhem na alteração do código ao mesmo tempo.

6.1 – Estruturas

O módulo Estruturas abriga as classes responsáveis pelos objetos físicos considerados no modelo bem como suas propriedades e conexões.

6.1.1 – Classe “noh”

Esta classe representa os pontos de interconexão presente entre os diversos dutos presentes na rede de abastecimento. O nome dado é decorrente da versão 2.7 do Python não aceitar acentos.

Entre os atributos dessa classe estão:

a) Atributo “nome” - Uma simples etiqueta para diferenciar os diversos nós que compõe a rede. É prudente utilizar um desenho base com essa nomenclatura para melhor identificação.

b) Atributo “ordem” - Em virtude da melhor execução do programa, assim que um nó é inicializado ele recebe um atributo ordem, equivalente à posição em um fila de elementos similares para fins de busca. Diferente do Nome não é um parâmetro visível ao usuário. Ele é utilizado apenas em processos internos.

c) Atributo “q” - Vazão que sai da rede através daquele nó em L/s. Este é um parâmetro que deve ser previamente estimado pelo usuário. Este procedimento é bem descrito por TSUTIYA (2006) e pode corresponder à vazão que escapa para uma rede secundária, por exemplo. Este é um dos desafios do trabalho de um Engenheiro no Dimensionamento de Redes de Abastecimento que deve estar munido com boas ferramentas de previsão e assunções acuradas.

d) Atributo “p” - Altura em metros da linha piezométrica na posição do nó correspondente com relação ao referencial da cota adota pelos pontos. É composta pela parcela de pressão ao longo do fluido e a parcela gravitacional da equação de Bernoulli. É a principal variável de controle a ser aplicada no equilíbrio do sistema, por isto é inicializada, por padrão, com o valor da cota do ponto. O que permite que se trabalhe com uma solução inicial diferente de 0, aumentando a velocidade de convergência dos pontos, ou pelo menos, evitando problemas com a divisão de números nulos.

e) Atributo “h” - Altura manométrica. É a variável que representa a pressão no duto. É utilizada como output para a verificação de conformidade do dimensionamento conforme preconiza a norma.

6.1.2 – Classe “trecho”

Esta classe representa a tubulação da rede de abastecimento. É uma classe usada apenas para guardar as informações relativas a estes componentes, não possuindo nenhum método aplicado.

Os diferentes atributos dessa classe são:

a) Atributo “nohInicial” - Nome do nó a que está ligado. Deve ser fornecido imputado conforme a classe anterior que podem ser letras ou números.

b) Atributo “nohFinal” – Assim como o atributo anterior, deve ser fornecido o nome do nó da segunda extremidade. As diferentes possibilidades de alocação dos nós nessas posições determinará a orientação do trecho, que influenciará no sinal da vazão e da velocidade.

c) Atributo “material” - material componente do duto que forma o trecho que também deverá ser cadastrado conforme a classe “material”.

d) Atributo “l” – comprimento total do trecho em metros.

e) Atributo “d” – diâmetro do duto que compõe o trecho.

f) Atributo “q” – vazão do trecho L/s. Esse atributo é calculado após o balanceamento da rede.

g) Atributo “velocidade” – velocidade do fluido que passa pelo duto em m/s. Esse atributo é calculado após o balanceamento da rede.

6.1.3 – Classe “material”

É uma classe utilizada apenas para cadastrar os diversos tipos de materiais que podem ser utilizadas na rede.

Seus atributos são:

a) Atributo “nome” – nome do material. Deve ser cadastrado conforme o material utilizado na classe trecho.

b) Atributo “c” – coeficiente de Hazen-Willians. A determinação do valor desse coeficiente para cada usuário fica a critério do usuário.

6.1.3 – Classe “reservatorio”

Nesta metodologia, um reservatório não considerado um dos nós, mas apenas um componente ligado à um deles. Dessa maneira, se faz necessário a criação de uma classe específica para se trabalhar com esse componente.

Os diferentes atributos dessa classe são:

a) Atributo “nohConectado” – nome nó da rede ao qual está conectado o reservatório. É importante notar que o reservatório não é um nó da rede, mas apenas se liga a ela.

b) Atributo “cota” – cota do nível da água do reservatório em metros. Supõe-se que o nível do reservatório não se altera durante o regime.

c) Atributo “vazao” – vazão com que o reservatório alimenta o sistema em L/s. É possível que o sistema esteja alimentando o reservatório. Neste caso a vazão receberá um valor negativo. Esse parâmetro é calculado após o balanceamento da rede.

6.1.4 – Classe “rede”

É a principal classe desse módulo. Não tem apenas a atribuição de agregar todas as informações capturadas pelas outras classes, mas integra através de métodos estes componentes.

Os elementos dessa classe são:

a) Atributo “nome” - nome da rede. Usado apenas para diferenciar esta rede de demais redes utilizadas concomitantemente pelo programa.

b) Atributo “reservatórios” – lista com os reservatórios pertencentes à rede. O primeiro elemento é um objeto vazio da classe reservatório para identificação do tipo de elemento da lista.

c) Atributo “nohs” – lista com os nós pertencentes à rede. O primeiro elemento é um objeto vazio da classe “noh” para identificação do tipo de elemento da lista.

d) Atributo “trechos” – lista com os nós pertencentes à rede. O primeiro elemento é um objeto vazio da classe “noh” para identificação do tipo de elemento da lista.

e) Atributo “materiais” – lista com os materiais pertencentes à rede. O primeiro elemento é um objeto vazio da classe “material” para identificação do tipo de elemento da lista.

f) Método “convertNameOrdem” – transforma o nome do nó no número da sua ordem na lista dos nós. Assim é possível fazer manipulações algébricas com a posição do nó.

g) Método “materialNomeToMaterial” – recebe o nome de um material e retorna o objeto da classe material correspondente.

h) Método “calcAlturaManometrica” – recebe a pressão piezométrica de cada nó após o balanceamento da rede e fornece a altura manométrica ao subtrair a parcela da cota.

i) Método “calcVazoes” – Calcula as vazões de cada trecho a partir das pressões piezométricas fornecidas, por meio da formula de Hazen-Willians.

j) Método “calcVelocidades” – Calcula as velocidades de cada trecho a partir da vazão.

k) Método “calcPressaoDinamicaMinima” – Para cada trecho calcula o valor da pressão dinâmica mínima ao longo do seu perfil, ao somar a altura manométrica à parcela da velocidade da equação de Bernoulli.

6.2 – Calculadoras

É a principal classe do programa, onde são efetuados todos os cálculos numéricos e onde se geram os resultados. A relação dos diferentes elementos dessa classe coloca em prática a metodologia explicada no Tópico 5.

6.2.1 – Classe “hazenBase”

Esta classe representa cada iteração do método proposto.

Os principais elementos dessa classe são:

a) Atributo “redeInicial” – rede a ser balanceada.

b) Atributo “redeFinal” – rede balanceada.

c) Método “criaMatriz” – cria a matriz de linearização com a blibioteca numpy.

d) Método “preencheMatriz” – preenche a matriz com os coeficientes que correspondem ao cálculo das vazões.

e) Método “criaVetor” – cria o vetor de condições de contorno com a blibioteca numpy.

f) Método “preencheVetor” – preenche os elementos do vetor criado ao adicionar as vazões de saída e a pressão piezométrica dos reservatórios.

g) Método “executar” – executa na ordem certa todos os passos para o balanceamento da rede, incluindo a resolução do sistema linear.

6.2.2 – Classe “hazenFinal”

Esta classe efetua várias iterações da classe “hazenBase” para obter uma convergência completa das variáveis de decisão

Os elementos dessa classe são

a) Atributo “redeInicial” – Similar à classe “HazenBase” este é a rede de partida para o balanceamento

b) Atributo “redeFinal” - Rede que contém todos os parâmetros já calculados.

c) Atributo “limiteMetrica” – Métrica definida para o controle do método iterativo.

d) Atributo “limiteIteracoes” – Limite de iterações de cada chamada do método “HazenBase”. Este limite tem o intuito de evitar loopings infinitos.

e) Método “metrica”- Método que informa o nível de convergência entre duas iterações consecutivas do método.

f) Método “executar” – Método que irá efeturar o balanceamento da rede efetuando cálculos iterativos.

6.3 – Interface

Nomeada de Fluo, que também dá nome ao *addin* no formato xlam onde estarão disponíveis todas as funções para o usuário final é o módulo necessário para o uso do xlwings. A partir de elementos especiais presentes no código, habilita que a ferramenta possa utilizar o VBA – linguagem de programação da Microsoft – para a integração do código.

As funções presentes nesse módulo são:

a) Função “inserirMateriais” – cria uma lista de elementos da classe material e salva na memória com um nome especificado.

b) Função “inserirReservatorios” – cria uma lista de elementos da classe material e salva na memória com um nome específico.

c) Função “inserirNos” – cria uma lista de elementos da classe “noh” e salva na memória com um nome específico.

d) Função “inserirTrechos” – cria uma lista de elementos da classe “trecho” e salva na memória com um nome específico.

e) Função “criarProjeto” – a partir dos conjuntos formados pelas funções anteriores cria um objeto da classe “rede” e salva na memória com um nome específico.

f) Função “calcSolucao” – essa função, a partir das classes do módulo calculadoras, efetua o balanceamento da rede.

g) Função “listeAlturaManometrica” – após efetuado o balanceamento da rede, fornece a altura manométrica de cada nó em forma de lista descendente a partir da célula da qual a função é chamada.

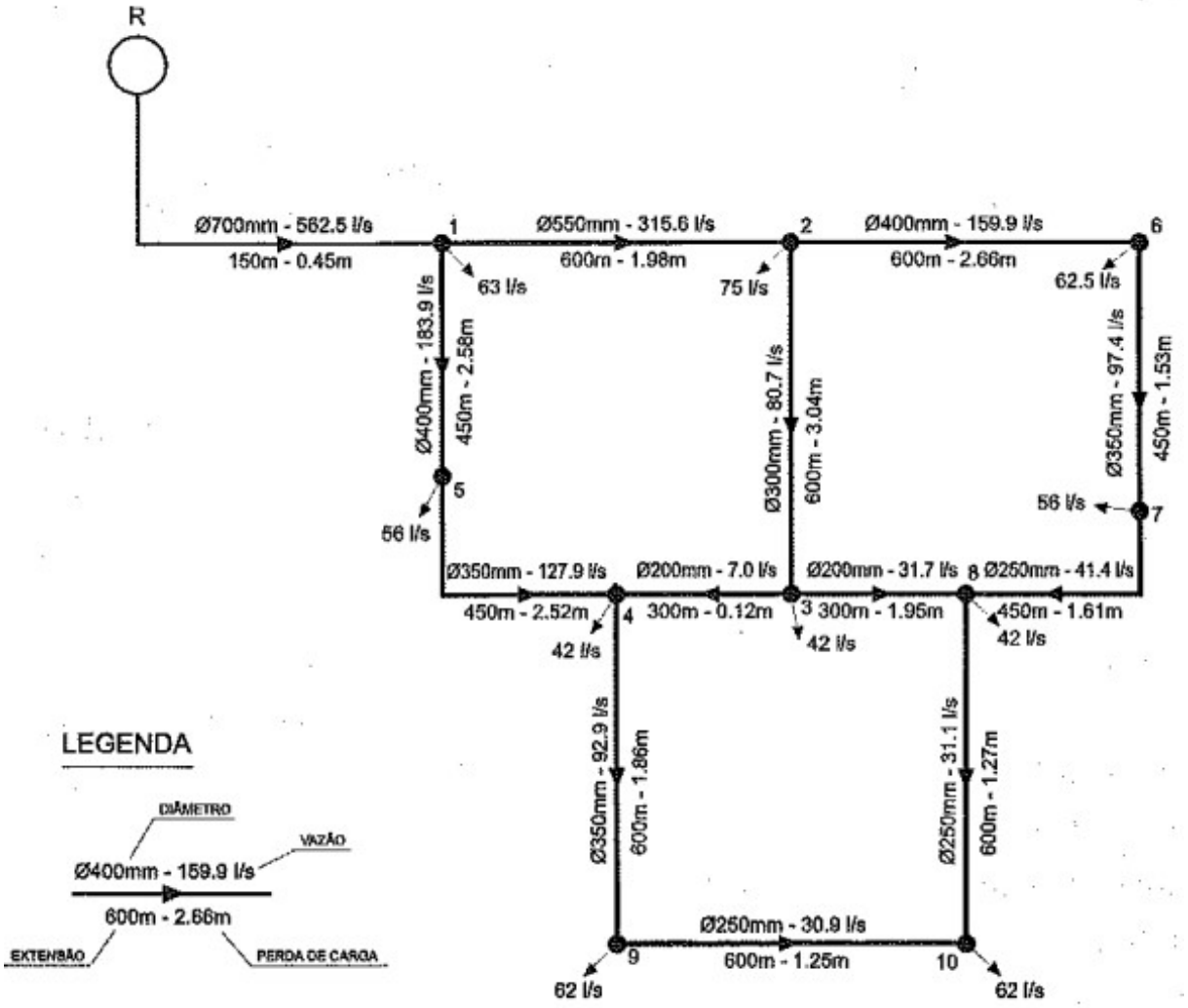
h) Função “listeVazaoReservatorio” – após efetuado o balanceamento da rede, fornece a vazão de cada reservatório em forma de lista descendente a partir da qual a função é chamada.

i) Função “listeVazaoTrechos” – após efetuado o balanceamento da rede, fornece a vazão de cada trecho em forma de lista descendente a partir da célula da qual a função é chamada.

j) Função “listePressaoDinamicaMinima” – após efetuado o balanceamento da rede, fornece a pressão dinâmica mínima de cada trecho em forma de lista descendente a partir da célula da qual a função é chamada.

7 – Resultados

A Rede da figura 5 foi tirado do livro “Abastecimento de Água” (TSUTIYA, 2006) e serve como uma referência confiável para se testar a implementação da metodologia implementada em um exemplo notório de solução.



Fonte: TSUTIYA

Figura 5 - Representação do problema proposto

As tabelas 1 e 2 mostram de forma compilada os dados imputados na ferramenta implementada. Por motivos de adequação com as imputações exigidas pelo programa, arbitrou-se o valor de 600m para a cota do nível de água no reservatório. Para efeitos de resultados, essa consideração não irá alterar os números.

Tabela 1 - Dados sobre os nós

Nó	Cota (m)	Vazão de Saída (L/s)
O	600,00	0,00
1	581,00	63,00
2	572,00	75,00
3	560,00	42,00
4	562,00	42,00
5	570,00	56,00
6	567,00	62,50
7	558,00	56,00
8	557,00	42,00
9	547,00	62,00
10	541,00	62,00

Tabela 2 - Dados dos trechos

Nó Inicial	Nó Final	Material	Comprimento (m)	Diâmetro Inicial (mm)
O	1	MAT1	150	700
1	2	MAT1	600	550
2	6	MAT1	600	400
1	5	MAT1	450	400
2	3	MAT1	600	300
6	7	MAT1	450	350
5	4	MAT1	450	350
3	4	MAT1	300	200
7	8	MAT1	450	250
3	8	MAT1	300	200
4	9	MAT1	600	350
9	10	MAT1	600	250
8	10	MAT1	600	250

Com os dados imputados corretamente conforme exemplifica o Anexo A. O programa convergiu para o resultado mostrado na tabela 3, que pode ser comparado com os dados fornecidos pelo TSUTIYA (2006).

Tabela 3 - Comparação de Vazões

Trecho	Vazão Fluo (L/s)	Vazão de referência (L/s)	Diferença absoluta (L/s)	Diferença Relativa
O-1	562,50	562,50	0,00	0,0%
1-2	313,78	315,60	1,82	0,6%
2-6	156,46	159,90	3,44	2,2%
1-5	185,72	183,90	1,82	1,0%
2-3	82,32	80,70	1,62	2,0%
6-7	93,96	97,40	3,44	3,5%
5-4	129,72	127,90	1,82	1,4%
3-4	6,48	7,00	0,52	7,5%
7-8	37,96	41,40	3,44	8,3%
3-8	33,84	31,70	2,14	6,8%
4-9	94,20	92,90	1,30	1,4%
9-10	32,20	30,90	1,30	4,2%
8-10	29,80	31,10	1,30	4,2%

É possível perceber que as vazões no geral são bastante aderentes. Entretanto, para confirmar a convergência do método é possível recalcular a condição de contorno 2 conforma mostrada no tópico 5. Os resultados são mostrados na tabela 4.

Tabela 4 - Verificação de Vazões

Nó Final	Nó Inicial	Vazão	Nó Inicial	Vazão	Nó Inicial	Vazão	Vazão de saida	Soma
O	1	-562,50	Reservatorio	562,50			0,00	0,00
1	O	562,50	2	-313,78	5	-185,72	-63,00	0,00
2	1	313,78	6	-156,46	3	-82,32	-75,00	0,00
3	2	82,32	4	-6,48	8	-33,84	-42,00	0,00
4	5	129,72	3	6,48	9	-94,20	-42,00	0,00
5	4	-129,72	1	185,72			-56,00	0,00
6	2	156,46	7	-93,96			-62,50	0,00
7	6	93,96	8	-37,96			-56,00	0,00
8	10	-29,80	3	33,84	7	37,96	-42,00	0,00
9	10	-32,20	4	94,20			-62,00	0,00
10	8	29,80	9	32,20			-62,00	0,00

É possível também realizar uma comparação envolvendo os valores das pressões manométricas obtidas. Esta comparação é mostrada na tabela 5.

Tabela 5 - Comparação de alturas manoméricas

Nó	Altura Manométrica (mca)	Altura Manométrica de referência(mca)	Diferença absoluta (mca)	Diferença Relativa
1	18,55	18,55	0,00	0,0%
2	25,60	25,57	0,03	0,1%
3	34,44	34,53	0,09	0,3%
4	32,34	32,45	0,11	0,3%
5	26,93	26,97	0,04	0,2%
6	28,05	27,91	0,14	0,5%
7	35,62	35,38	0,24	0,7%
8	35,25	34,77	0,48	1,4%
9	45,43	45,59	0,16	0,4%
10	50,08	49,50	0,58	1,2%

8 – Conclusão

A ferramenta conseguiu alcançar o objetivo ao fornecer uma alternativa facilitada de dimensionamento de redes de abastecimento hidráulico. Habilitando assim o seu uso em disciplinas de Engenharia Hidráulica.

Para o uso profissional, entretanto, seria interessante, o acréscimo de novas utilizações, de modo a ser comparável com os principais softwares disponíveis no mercado.

Seria interessante buscar a complementação dessa ferramenta com implementações que viessem a agregar a sua complexidade. Por exemplo, a introdução da fórmula universal no cálculo das vazões.

9 – Referências

QUIGLEY, Antony. *What is Python, and how is it used by today's coders?* 02 de novembro de 2017. Disponível em <https://www.codeinstitute.net/blog/python-used-todays-coders/> Acesso em 02/11/2017.

ROSEBROCK, Adrian. *7 Reasons Why You Should Learn Python Right Now. 18 de Janeiro de 2017*. Disponível em <https://simpleprogrammer.com/2017/01/18/7-reasons-why-you-should-learn-python/> Acesso em 02/11/2017.

YEGULALP, Serdar. *Excel gets Python programming power, thanks to Xlwings library*. 26/09/2014. Disponível em <https://www.infoworld.com/article/2687825/application-development/excel-gets-python-programming-power-thanks-to-xlwings-library.html> Acesso em 02/11/2017.

TSUTIYA, Milton. *Abastecimento de água*. 3º edição. Departamento de Engenharia Hidráulica e Sanitária da Escola Politécnica da Universidade de São Paulo, 2006. ISBN 85-900823-6-9

P. I. B. Q. *Algumas Aplicações do Método dos Elementos Finitos a Redes de Fluxo.* Notas de aula. Divisão de Engenharia Civil-Aeronáutica do Instituto Tecnológico de Aeronáutica. São José dos campos, 2005.

ABNT - Associação Brasileira de Normas Técnicas. NBR 12218: 2017 Projeto de Rede de Distribuição de Água para Abastecimento Público. Rio de Janeiro, 2017.

Apêndice A: Manual de Uso

Deve-se primeiramente efetuar a instalação do Python 2.7 bem como dos módulos numpy e xlwings.

Em seguida deve-se inicializar o Excel juntamente com o arquivo, Fluo.xlam. Assim, todas as funções necessárias já estarão disponíveis para uso em qualquer planilha desta instância.

É importante observar que é permitida a utilização de letras com acentos nem do cedilha em nenhuma das células, por limitações da versão do Python 2.7.

Para a exemplificação do uso das funções será usada a seguinte rede:

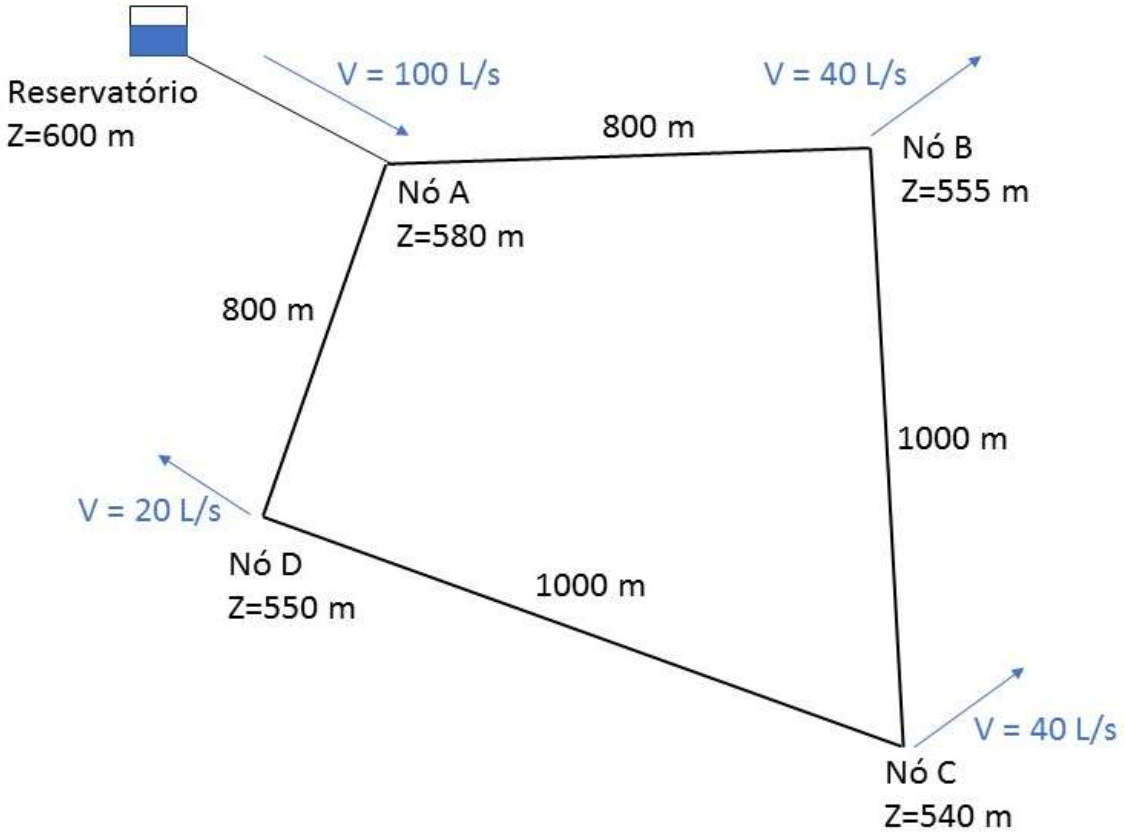


Figura 6 - Representação da rede utilizada no exemplo

Os demais parâmetros estaão explicitados nas figuras 7 à 10.

Deve-se utilizar as seguintes funções na ordem que se segue. É possível usar o botão “fx” do Excel para obter ajuda para o uso das funções.

A.1 – Função “inserirMateriais”

Figura 7 - Função inserirMateriais

Uma execução bem-sucedida da função retornará o horário atualizado (deve-se formatar a célula para ver em formato compreensível)

A.2 – Função “inserirNos”

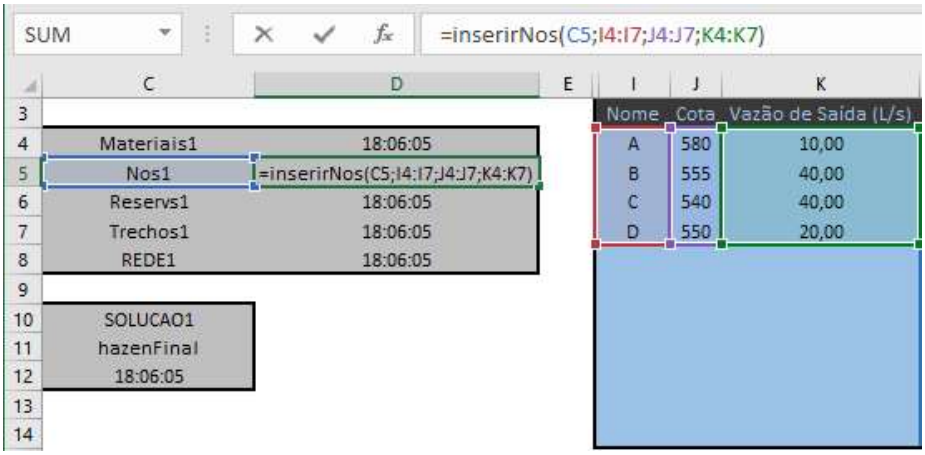


Figura 8 - função inserirNos

Uma execução bem-sucedida da função retornará o horário atualizado (deve-se formatar a célula para ver em formato compreensível)

A.3 – Função “inserirReservatorios”

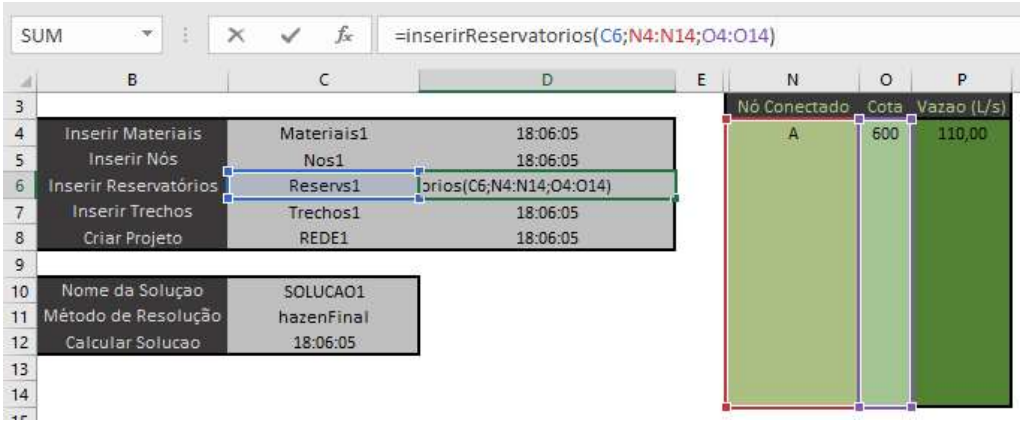


Figura 9 - Função inserirReservatorios

Uma execução bem-sucedida da função retornará o horário atualizado (deve-se formatar a célula para ver em formato compreensível)

A.4 – Função “InserirTrechos”

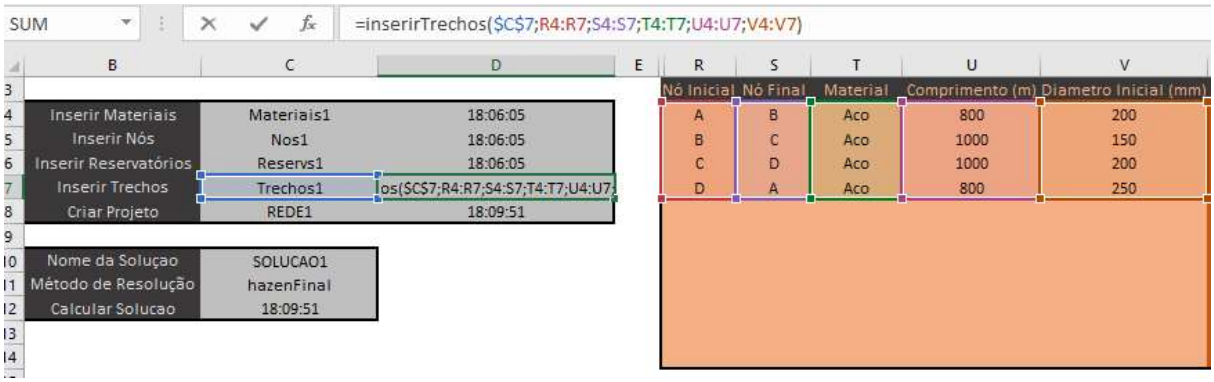


Figura 10 - Função inserirTrechos

Uma execução bem-sucedida da função retornará o horário atualizado (deve-se formatar a célula para ver em formato compreensível)

A.5 – Função “criarProjeto”

Esta função receberá a referência das entradas acima para criar uma instância da classe rede e adicioná-la à memória através de uma nova indexação.

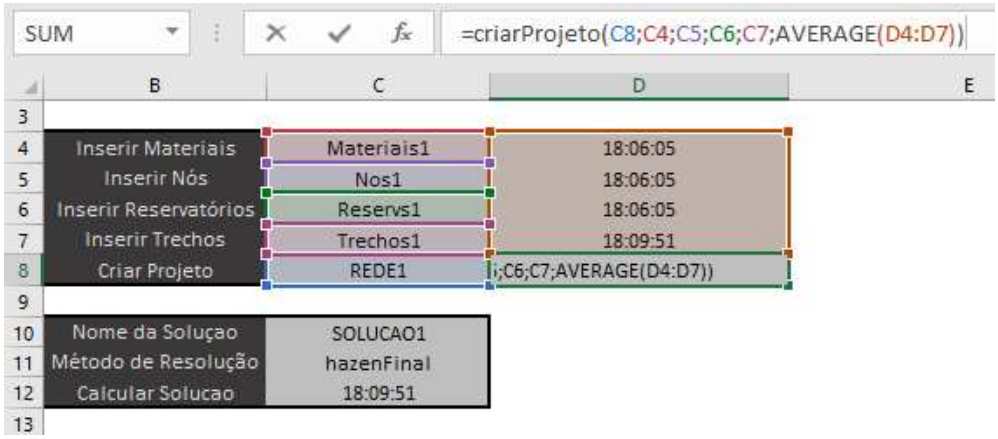


Figura 11- Função criarProjeto

A utilização dos resultados das funções anteriores tem por objetivo garantir a ordem certa de execução.

A.6 – Função “calcSolucao”

A partir da rede criada anteriormente e recebendo como parâmetro o método de resolução que representa uma classe do módulo Calculadoras, esta função irá balancear a rede e adicionar a solução a memória a partir de uma indexação.

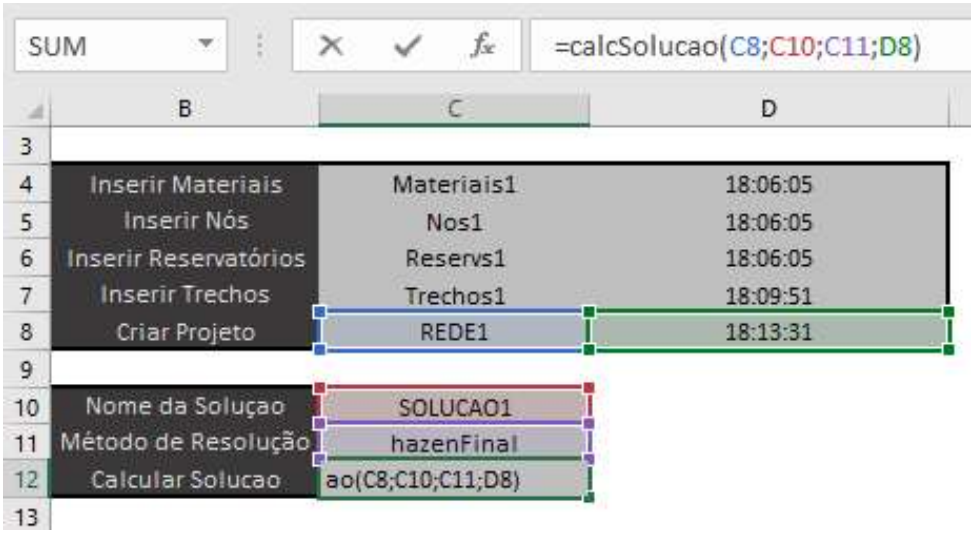


Figura 12 - Função calcSolucao

A utilização do resultado da função criarProjeto tem por objetivo garantir a ordem certa de execução.

Esta função não mostrará nenhum resultado. Para isso, deverá ser utilizado as funções a seguir.

A.7 – Função “listeAlturaManometrica”

Modifica os valores das células imediatamente abaixo para mostrar o valor da altura manométrica dos nós da rede balanceada por ordem de entrada.

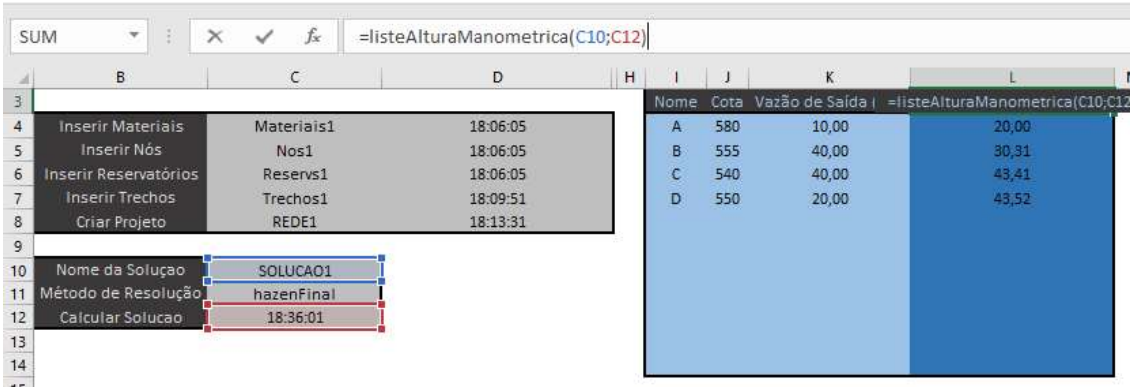


Figura 13 - Função listeAlturaManometrica

A utilização do resultado da função calcSolucao tem por objetivo garantir ordem certa de execução.

A.8 – Função “listeVazaoReservatorio”

Modifica os valores das células imediatamente abaixo para mostrar o valor das vazões de saída do reservatório da rede balanceada por ordem de entrada.

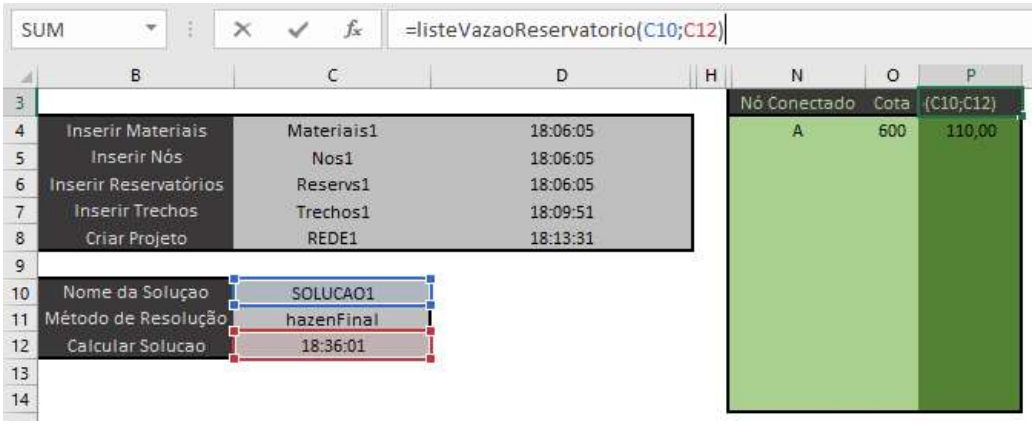


Figura 14 - Função listeVazaoReservatorio

A utilização do resultado da função calcSolucao tem por objetivo garantir ordem certa de execução.

A.9 – Função “listeVazaoTrechos”

Modifica os valores das células imediatamente abaixo para mostrar o valor das vazões de cada trecho da rede balanceada por ordem de entrada.

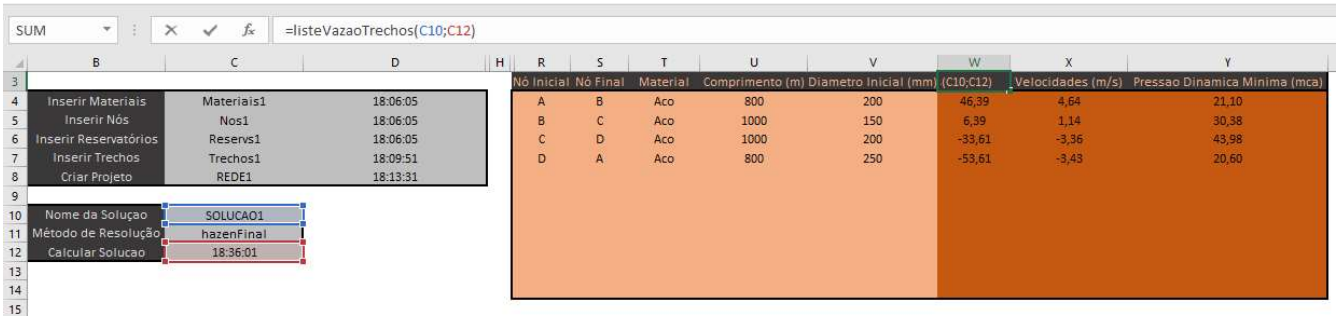


Figura 15 - Função listeVazaoTrechos

A utilização do resultado da função calcSolucao tem por objetivo garantir ordem certa de execução.

A.10 – Função “listeVelocidades”

Modifica os valores das células imediatamente abaixo para mostrar o valor das velocidades de cada trecho da rede balanceada por ordem de entrada.

Figura 16 - listeVelocidades

A utilização do resultado da função calcSolucao tem por objetivo garantir ordem certa de execução.

A.11 – Função “listePressaoDinamicaMinima”

Modifica os valores das células imediatamente abaixo para mostrar o valor das pressões dinâmicas mínimas de cada trecho da rede balanceada por ordem de entrada.

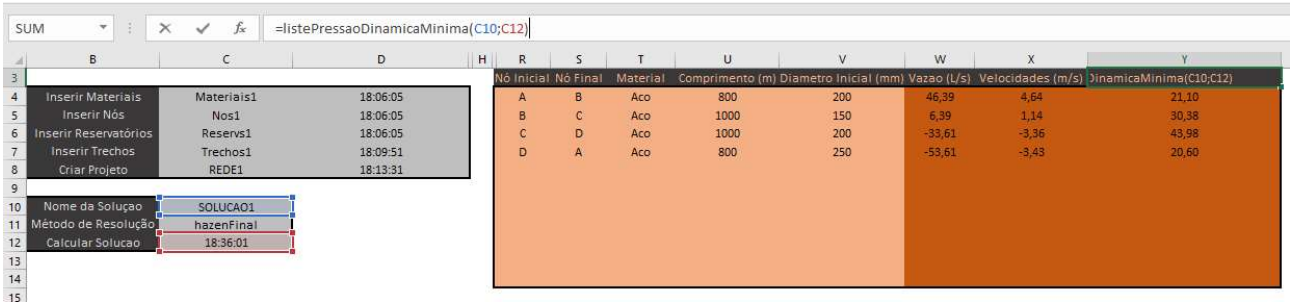


Figura 17 - Função listePressaoDinamicaMinima

A utilização do resultado da função calcSolucao tem por objetivo garantir ordem certa de execução.

Apêndice B – Código Fonte

B. 1 – Módulo Calculadoras

```
import Estruturas

import numpy as np

import copy

Solucoes = dict()



def deriva(rede = Estruturas.rede(),tipo = "hazenBase"):

    if tipo == "hazenBase": return hazenBase(rede)

    if tipo == "hazenFinal": return hazenFinal(rede)

    return 0.0

class solver():

    def __init__(self, rede = Estruturas.rede()):

        self.redeInicial = rede

        self.redeFinal = rede

        self.tipo = tipo

class hazenFinal(solver):

    def __init__(self,rede=Estruturas.rede()):

        self.redeInicial = rede

        self.redeFinal = rede

        self.limiteMetrica = 0.000000000000000001 #0.00005102040816326531
#conforme NBR12218
```

```
    self.limiteIteracoes = 100000
    self.executar()
def metrica(self,rede1=Estruturas.rede(),rede2=Estruturas.rede()):
    result = 0.0
    n = len(rede1.nohs)-1
    for i in range(1,n+1):
        p1 = rede1.nohs[i].p
        p2 = rede2.nohs[i].p
        result = max(result,(p1-p2)**2.0)
    return result
def executar(self):
    l = [self.redeInicial]
    dup = copy.deepcopy(l)
    redeAuxZero = dup[0]
    solucaoAux = hazenBase(self.redeInicial)
    redeAux = solucaoAux.redeFinal
    i = 0
    metrica = self.metrica(redeAuxZero,redeAux)
    while metrica > self.limiteMetrica and i  < self.limiteIteracoes:
        l = [redeAux]
        dup = copy.deepcopy(l)
        redeAuxZero = dup[0]
        solucaoAux = hazenBase(redeAux)
        redeAux = solucaoAux.redeFinal
```

```
            metrica = self.metrica(redeAuxZero,redeAux)
            i = i + 1
        self.redeFinal = redeAux
class hazenBase(solver):
    def __init__(self,rede=Estruturas.rede()):
        self.redeInicial = rede
        self.redeFinal = rede
        self.executar()
    def criaMatriz(self):
        n = len(self.redeInicial.nohs)-1
        matriz = np.ndarray(shape=(n,n), dtype=float)
        for i in range(n):
            for j in range(n):
                matriz[i][j] = 0.0
        return matriz
    def preencheMatriz(self):
        matrizAux = self.criaMatriz()
        matriz = self.criaMatriz()
        for k in range(1,len(self.redeInicial.trechos)):
            trecho = self.redeInicial.trechos[k]
            d = trecho.d*10.0**(-3.0)
            l = trecho.l
            nA = trecho.nohInicial
            nB = trecho.nohFinal
```

```
                n1 = self.redeInicial.convertNameOrdem(nA)

                n2 = self.redeInicial.convertNameOrdem(nB)

                p1 = self.redeFinal.nohs[n1].p

                p2 = self.redeFinal.nohs[n2].p

                mat = self.redeInicial.materialNomeToMaterial(trecho.material)

                cHazen = mat.c


                coef = abs(p1-p2)**(1/1.85-
1)*(1/l/10.6432283717327*(d)**4.87)**(1/1.85)*cHazen

                #coef = abs(p1-p2)**(1/1.852-
1)*(1/l/10.67*(d)**4.8704)**(1/1.852)*cHazen

                matriz[n1-1][n1-1] = matriz[n1-1][n1-1] - coef

                matriz[n1-1][n2-1] = matriz[n1-1][n2-1] + coef

                matriz[n2-1][n1-1] = matriz[n2-1][n1-1] + coef

                matriz[n2-1][n2-1] = matriz[n2-1][n2-1] - coef

            n = len(self.redeInicial.nohs)-1

            k = len(self.redeInicial.reservatorios)-1

            for i in range(1,k+1):

                reservatorio = self.redeInicial.reservatorios[i]

                nohName = reservatorio.nohConectado

                nohOrdem = self.redeInicial.convertNameOrdem(nohName)

                for j in range(1,n+1): matriz[nohOrdem-1][j-1] = 0.0

                matriz[nohOrdem-1][nohOrdem-1] = 1.0

            return matriz

        def criaVetor(self):
```

```
    n = len(self.redeInicial.nohs)-1
    vetor = np.ndarray(shape=(n), dtype=float)
    return vetor
def preencheVetor(self):
    vetor = self.criaVetor()
    n = len(self.redeInicial.nohs)-1
    for i in range(1,n+1):
        noh = self.redeInicial.nohs[i]
        vetor[i-1] = noh.q*10.0**(-3.0)
    k = len(self.redeInicial.reservatorios)-1
    for i in range(1,k+1):
        reservatorio = self.redeInicial.reservatorios[i]
        nohName = reservatorio.nohConectado
        nohOrdem = self.redeInicial.convertNameOrdem(nohName)
        vetor[nohOrdem-1] = reservatorio.cota
    return vetor
def executar(self):
    redeFinal = self.redeInicial
    matriz = self.preencheMatriz()
    vetor = self.preencheVetor()
    Result = np.linalg.solve(matriz,vetor)
    n = len(redeFinal.nohs)-1
    for i in range(1,n+1):
        noh = redeFinal.nohs[i]
```

```
                noh.p = Result[i-1]

            self.redeFinal = redeFinal
```

B.2 – Módulo Estruturas

```
Projetos = dict()

Nohs = dict()

Materiais = dict()

Trechos = dict()

Reservatorios = dict()


class material():

    def __init__(self,nome= "",c =0.0):

        self.nome = nome

        self.c = c

class noh():

    def __init__(self,nome="",cota=0.0,q=0.0,redeAux = None, pressaoEstaticaMaxima
= 50.0):

        if not redeAux:

            self.ordem = 0

        else:

            auxList = redeAux.nohs

            n = len(auxList)-1

            self.ordem = auxList[n].ordem+1
```

```
        self.nome = nome
        self.cota = cota
        self.q = q
        self.p = cota
        self.h = 0.0
class trecho():
    def
__init__(self,nohInicial="",nohFinal="",material="",comprimento=0.0,diametro=0.0):
        self.nohInicial = nohInicial
        self.nohFinal = nohFinal
        self.material = material
        self.l = comprimento
        self.d = diametro
        self.q = 0.0
        self.velocidade = 0.0
        self.pressaoDinamicaMinima = 0.0
class reservatorio():
    def __init__(self,nohConectado = 0.0, cota = 0.0):
        self.nohConectado = nohConectado
        self.cota = cota
        self.vazao = 0.0
class rede():
    def __init__(self,nome = None):
        self.g = 9.8
        self.nome = nome
```

```
    self.reservatorios = []
    self.reservatorios.append(reservatorio())
    self.nohs = []
    self.nohs.append(noh())
    self.trechos = []
    self.trechos.append(trecho())
    self.materiais = []
    self.materiais.append(material())
def convertNameOrdem(self,nome = None):
    for noh in self.nohs:
        if noh.nome == nome : return noh.ordem
def materialNomeToMaterial(self,nome = None):
    for material in self.materiais:
        if material.nome == nome : return material
    return None
def calcAlturaManometrica(self):
    nNohs = len(self.nohs)-1
    for i in range(1,nNohs+1):
        noh = self.nohs[i]
        noh.h = noh.p - noh.cota
def calcVazoes(self,metodo ="Hazen"):
    nTrechos = len(self.trechos)-1
    for i in range(1,nTrechos+1):
        trecho = self.trechos[i]
```

```
            d = trecho.d*10.0**(-3.0)
            l = trecho.l
            nA = trecho.nohInicial
            nB = trecho.nohFinal
            n1 = self.convertNameOrdem(nA)
            n2 = self.convertNameOrdem(nB)
            p1 = self.nohs[n1].p
            p2 = self.nohs[n2].p
            if metodo == "Hazen":
                mat = self.materialNomeToMaterial(trecho.material)
                cHazen = mat.c
                trecho.q = (p1-p2)*abs(p1-p2)**(1/1.85-
1)*(1/l/10.6432283717327*(d)**4.87)**(1/1.85)*cHazen*1000.0
                #trecho.q = (p1-p2)*abs(p1-p2)**(1/1.852-
1)*(1/l/10.67*(d)**4.8704)**(1/1.852)*cHazen*1000.0
        nReservatorios = len(self.reservatorios)-1
        for i in range(1,nReservatorios+1):
            reservatorio = self.reservatorios[i]
            nohNome = reservatorio.nohConectado
            q = 0.0
            for j in range(1,nTrechos+1):
                trecho = self.trechos[j]
                if trecho.nohInicial == nohNome: q = q + trecho.q
                if trecho.nohFinal == nohNome: q = q - trecho.q
            nohOrdem = self.convertNameOrdem(nohNome)
```

```
            noh = self.nohs[i]
            q = q + noh.q
            reservatorio.vazao = q
        return None
    def calcVelocidades(self):
        nTrechos = len(self.trechos)-1
        for i in range(1,nTrechos+1):
            trecho = self.trechos[i]
            q = trecho.q
            d = trecho.d
            v = q*10.0**(-3.0)/((d*10.0**(-3.0))**2.0/4.0)
            trecho.velocidade = v
        return
    def calcPressaoDinamicaMinima(self):
        nTrechos = len(self.trechos)-1
        for i in range(1,nTrechos+1):
            trecho = self.trechos[i]
            nA = trecho.nohInicial
            nB = trecho.nohFinal
            n1 = self.convertNameOrdem(nA)
            n2 = self.convertNameOrdem(nB)
            h1 = self.nohs[n1].h
            h2 = self.nohs[n2].h
            pDinamicaMinima = min(h1,h2)
```

```
            v = trecho.velocidade
            pDinamicaMinima = pDinamicaMinima + v**2.0/self.g/2.0
            trecho.pressaoDinamicaMinima = pDinamicaMinima
        return
```

B.3 – Módulo Utilidades

```
import time
import datetime
def excelTime(time = None):
    if not time :
        time = datetime.datetime.now()
    return time.hour / 24.0 + time.minute / (24.0*60.0) + time.second / (24.0*60.0*60.0)
+ time.microsecond / (24.0*60.0*60.0*1000000.0)
```

B.4 – Módulo Fluo

```
import numpy as np
import pandas as pd
import xlwings as xw
from TG import Estruturas
from TG import Utilidades
from TG import Calculadoras
import itertools
```

```
@xw.func

def criarProjeto(Projeto ="", ConjuntoMateriais = None , ConjuntoNoh = None ,
ConjuntoReservatorio = None , ConjuntoTrecho = None , gatilho =None):

    if not isinstance(gatilho,float): return "erro"

    if None in (ConjuntoMateriais,ConjuntoNoh,ConjuntoReservatorio,ConjuntoTrecho):
return "erro"

    mLista = Estruturas.Materiais[ConjuntoMateriais]

    nLista = Estruturas.Nohs[ConjuntoNoh]

    rLista = Estruturas.Reservatorios[ConjuntoReservatorio]

    tLista = Estruturas.Trechos[ConjuntoTrecho]

    instanciaCriada = Estruturas.rede(Projeto)

    instanciaCriada.materiais = mLista

    instanciaCriada.nohs = nLista

    instanciaCriada.reservatorios = rLista

    instanciaCriada.trechos = tLista

    Estruturas.Projetos[Projeto] = instanciaCriada

    return Utilidades.excelTime()

@xw.func

def inserirMateriais(Conjunto="",nome="",c=0.0):

    if isinstance(nome,(str,unicode)): nome = [str(nome)]

    if isinstance(c,(float,unicode)): c = [float(c)]

    eLista = [Estruturas.material()]

    nome = filter(None,nome)

    c = filter(None,c)
```

```
    for _c , _nome in zip(c,nome): eLista.append(Estruturas.material(_nome,_c))
    Estruturas.Materiais[Conjunto] = eLista
    return Utilidades.excelTime()
@xw.func
def inserirReservatorios( Conjunto = "",nohConectado = "", cota = 0.0):
    if isinstance(nohConectado,(str,unicode)): nohConectado = [str(nohConectado)]
    if isinstance(nohConectado,float): nohConectado = [float(nohConectado)]
    if isinstance(cota,float): cota = [float(cota)]
    eLista = [Estruturas.reservatorio()]
    nohConectado = filter(None,nohConectado)
    cota = filter(None,cota)
    for nohConectado, cota in zip(nohConectado,cota):
eLista.append(Estruturas.reservatorio(nohConectado,cota))
    Estruturas.Reservatorios[Conjunto] = eLista
    return Utilidades.excelTime()
@xw.func
def inserirNos(Conjunto = "",nome="",cota=0.0,q=0.0):
    if isinstance(nome,(str,unicode)): nome = [str(nome)]
    if isinstance(cota,(float,unicode)): cota = [float(c)]
    if isinstance(q,(float,unicode)): q = [float(c)]
    eLista = [Estruturas.noh()]
    nome = filter(None,nome)
    cota = filter(None,cota)
    q = filter(None,q)
    rede = Estruturas.rede()
```

```
    for _cota , _nome ,_q in zip(cota,nome,q):
        rede.nohs = eLista
        eLista.append(Estruturas.noh(_nome,_cota,_q,rede))
    Estruturas.Nohs[Conjunto] = eLista
    return Utilidades.excelTime()
@xw.func
def
inserirTrechos(Conjunto="",nohInicial="",nohFinal="",material="",comprimento=0.0,di
ametro=0.0):
    if isinstance(nohInicial,(str,unicode)): nohInicial = [str(nohInicial)]
    if isinstance(nohInicial,float): nohInicial = [float(nohInicial)]
    if isinstance(nohFinal,(str,unicode)): nohFinal = [str(nohFinal)]
    if isinstance(nohFinal,float): nohFinal = [float(nohFinal)]
    if isinstance(material,(str,unicode)): material = [str(material)]
    if isinstance(material,float): material = [float(material)]
    if isinstance(material,float): comprimento = [float(comprimento)]
    if isinstance(material,float): diametro = [float(diametro)]
    eLista = [Estruturas.trecho()]
    nohInicial = filter(None,nohInicial)
    nohFinal = filter(None,nohFinal)
    material = filter(None,material)
    comprimento = filter(None,comprimento)
    diametro = filter(None,diametro)
    for _nohInicial , _nohFinal , _material , _comprimento , _diametro in
zip(nohInicial,nohFinal, material, comprimento,diametro):
```

```
eLista.append(Estruturas.trecho(_nohInicial,_nohFinal,_material,_comprimento,_diame
tro))
    Estruturas.Trechos[Conjunto] = eLista
    return Utilidades.excelTime()
@xw.func
@xw.ret(expand="vertical",transpose=True)
def listeAlturaManometrica(Solucao = "",gatilho=0.0):
    #try:
    if True:
        if gatilho <= 0.0 or 0.0 < gatilho:
            rede = Calculadoras.Solucoes[Solucao].redeFinal
            result = []
            result.append("Altura Manometrica (mca)")
            for noh in rede.nohs:
                if noh.ordem <>0:
                    result.append(noh.h)
            return result
    else:
    #except:
        return "Erro em referencia de celulas"
@xw.func
@xw.ret(expand="vertical",transpose=True)
def listeVazaoReservatorio(Solucao = "",gatilho=0.0):
    #try:
```

```
    if True:
        if gatilho <= 0.0 or 0.0 < gatilho:
            rede = Calculadoras.Solucoes[Solucao].redeFinal
            result = []
            result.append("Vazao (L/s)")
            n = len(rede.reservatorios) -1
            for i in range(1,n+1):
                reservatorio = rede.reservatorios[i]
                result.append(reservatorio.vazao)
            return result
    else:
    #except:
        return "Erro em referencia de celulas"
@xw.func
@xw.ret(expand="vertical",transpose=True)
def listeVazaoTrechos(Solucao="",gatilho=0.0):
    rede = Calculadoras.Solucoes[Solucao].redeFinal
    result = []
    result.append("Vazao (L/s)")
    n =len(rede.trechos)-1
    for i in range(1,n+1):
        trecho = rede.trechos[i]
        result.append(trecho.q)
    return result
```

```
@xw.func
@xw.ret(expand="vertical",transpose=True)
def listePressaoDinamicaMinima(Solucao="",gatilho=0.0):
    rede = Calculadoras.Solucoes[Solucao].redeFinal
    result = []
    result.append("Pressao Dinamica Minima (mca)")
    n =len(rede.trechos)-1
    for i in range(1,n+1):
        trecho = rede.trechos[i]
        result.append(trecho.pressaoDinamicaMinima)
    return result
@xw.func
@xw.ret(expand="vertical",transpose=True)
def listeVelocidades(Solucao="",gatilho=0.0):
    rede = Calculadoras.Solucoes[Solucao].redeFinal
    result = []
    result.append("Velocidades (m/s)")
    n =len(rede.trechos)-1
    for i in range(1,n+1):
        trecho = rede.trechos[i]
        result.append(trecho.velocidade)
    return result

@xw.func
```

```
def calcSolucao(Projeto="",Solucao="",metodo="",gatilho=0.0):
    #try:
    if True:
        if gatilho <= 0.0 or 0.0 < gatilho:
            rede = Estruturas.Projetos[Projeto]
            #solucaoInst = Calculadoras.deriva(rede)
            solucaoInst = Calculadoras.deriva(rede,metodo)
            solucaoInst.redeFinal.calcAlturaManometrica()
            solucaoInst.redeFinal.calcVazoes()
            solucaoInst.redeFinal.calcVelocidades()
            solucaoInst.redeFinal.calcPressaoDinamicaMinima()
            Calculadoras.Solucoes[Solucao] = solucaoInst
            return Utilidades.excelTime()
    else:
    #except:
        return "Erro em referencia de celulas"
if __name__ == '__main__':
    # To run this with the debug server, set UDF_DEBUG_SERVER = True in the
xlwings VBA module
```

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. REGISTRO N°	4. N° DE PÁGINAS
TC	21 de novembro de 2017	DCTA/ITA/TC-092/2017	57

5. TÍTULO E SUBTÍTULO:

Cálculo automático do dimensionamento de redes genéricas de distribuição de água potável com um número variado de reservatórios.

6. AUTOR(ES):

Victor Jucá Martins

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

1. Dimensionamento. 2. Redes de abastecimento. 3. Calculo automático. 4. *Multi-input*.5. Python. 6. Excel. 7. xlWings.

9.PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Distribuição de água; Abastecimento de água; Cálculo; Equações não-lineares; Linguagem de programação; Engenharia sanitária.

10. APRESENTAÇÃO: **(X) Nacional ( ) Internacional**

ITA, São José dos Campos. Curso de Graduação em Engenharia Civil-Aeronáutica. Orientador: Major Márcio Antônio da Silva Pimentel; coorientador: Prof. Dr. Paulo Ivo Braga de Queiroz. Publicado em 2017.

11. RESUMO:

O principal componente de um sistema de abastecimento de água potável é a sua rede de distribuição. É conveniente que esta seja planejada visando a redundância de fluxo por meio da utilização de redes malhadas. O dimensionamento destas, porém, depende da resolução de um sistema de equações não-lineares. O método de resolução padrão presente na literatura, Hardy-Cross, apresenta ineficiências em termos de execução e convergência. O *software* desenvolvido se propõe a resolver esses problemas a partir da aplicação de metodologia específica, capaz de processar redes genéricas e com mais de um reservatório, sem a necessidade de fornecimento de solução inicial nem de identificação manual da localização de ciclos na rede. A aplicação utiliza o motor de cálculo da linguagem de programação Python, a partir da interface popular do Microsoft Excel. Desta maneira, a procura por soluções econômicas para implantação e reforma dos sistemas de abastecimento se tornam rápidas e fáceis.

12. GRAU DE SIGILO:

(X) OSTENSIVO ( ) RESERVADO ( ) SECRETO