UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LEONARDO MALDANER

DESENVOLVIMENTO DE SOFTWARE ACADÊMICO PARA ENGENHARIA QUÍMICA UTILIZANDO LINGUAGEM PYTHON : SEPARADOR DE MISTURA BINÁRIA

FRANCISCO BELTRÃO
2019

LEONARDO MALDANER

DESENVOLVIMENTO DE SOFTWARE ACADÊMICO PARA ENGENHARIA QUÍMICA UTILIZANDO LINGUAGEM PYTHON: SEPARADOR DE MISTURA BINÁRIA

Monografia apresentada a disciplina de Trabalho de Conclusão de Curso 2, do curso de Engenharia Química da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de bacharel em Engenharia Química.
Orientador: Dr. Vilmar Steffen

FRANCISCO BELTRÃO
2019

FOLHA DE APROVAÇÃO

LEONARDO MALDANER

DESENVOLVIMENTO DE SOFTWARE ACADÊMICO PARA ENGENHARIA QUÍMICA UTILIZANDO LINGUAGEM PYTHON: SEPARADOR DE MISTURA BINÁRIA

Monografia apresentada a disciplina de Trabalho de Conclusão de Curso 2, do curso de Engenharia Química da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de bacharel em Engenharia Química.
Orientador: Dr. Vilmar Steffen

Data de aprovação: 26 de novembro de 2019

_____

Orientador: Prof. Dr. Vilmar Steffen

_____

Membro da Banca Prof. Dr. Jeconias Rocha Guimarães

UTFPR - FB

_____

Membro da Banca Prof. Me. Maiquiel Schmidt de Oliveira

UTFPR - FB

“A folha de aprovação assinada encontra-se na coordenação do curso.”

AGRADECIMENTOS

Inicialmente gostaria de agradecer aos meus pais, Geraldo Ignácio Maldaner e Lisete Terezinha Finger Maldaner, pelo apoio incondicional durante essa jornada, por proverem sempre mais do que eu precisei e por me incentivarem em momentos onde eu não acreditei em mim. Esse trabalho e todas as minhas conquistas sempre serão por vocês.

Gostaria de agradecer a todos os amigos da minha cidade natal, a minha irmã Patrícia e meu sobrinho Miguel que mesmo de longe sempre estiveram presentes e, sem dúvidas, tornaram tudo isso mais simples e suportável. Aos meus colegas de turma que estiveram comigo diariamente durante esses anos de graduação, meu muito obrigado por cada aprendizado e cada experiência, a engenharia teria sido ainda mais difícil sem vocês.

Por fim, mas não menos importante, agradeço ao meu orientador, professor Doutor Vilmar Steffen, pela disponibilidade, pela paciência e por todos os conhecimentos repassados. Foi um prazer ter tido a oportunidade de trabalhar com alguém tão competente.

LISTA DE ILUSTRAÇÕES

RESUMO

No presente trabalho buscou-se desenvolver um software acadêmico para resolução de balanços de massa em separador de mistura binária, visando a solução de todas as possíveis combinações de incógnitas com a mesma rotina de solução, em que o usuário escolhe as incógnitas e insere os valores das variáveis conhecidas, assim como é feito em simuladores modulares sequenciais. O tema foi escolhido devido a falta de softwares que realizam esse tipo de cálculo e que sejam de fácil utilização, sendo que a maioria deles exigem conhecimento de linguagem de programação. Para chegar no programa final, inicialmente fez-se o projeto de desenvolvimento da solução numérica, assim como o projeto da interface gráfica, posteriormente para a implementação computacional utilizou-se a linguagem de programação Python juntamente com as bibliotecas disponíveis; Kivy para a interface gráfica, o numpy para a parte numérica, entre outras. Para a implementação foram desenvolvidas rotinas separadamente para a parte numérica e para a parte gráfica, que então, foram integradas. Ao final da pesquisa foi possível obter um software intuitivo de fácil utilização, onde o usuário insere os valores das incógnitas e pressiona um botão que efetua o cálculo e mostra a situação do problema para os valores inseridos, o software desenvolvido é capaz de resolver o problema proposto para 24 possíveis combinações de incógnitas, utilizando a mesma rotina numérica. O trabalho desenvolvido pode servir de base para estudos futuros que busquem integrar os conhecimentos obtidos com a engenharia química com o desenvolvimento de softwares com interface gráfica.

Palavras-chave: Software acadêmico. Balanço de massa em separadores. Python.

ABSTRACT

In this paper we sought to develop an academic software for mass balance calculations in a binary mixture separator, aiming to solve all sets of variables with the same computational routine, in which the user chooses the variables and sets its values, just like it is done in sequential modular simulators. This subject was choose due the lack of software available for this kind of calculations that are easy to use, most of them require some knowledge of specific programming languages. To achieve the final program the first step was to draw a project of the numeric solution and the graphic interface, after that, for the computational implementation the programming language Python was used with its available libraries; such as Kivy for graphic development, Numpy for numeric purposes among others. In order to make the implementation process easier the numeric and graphic part of the code were developed separately and them they were integrated using object oriented programming concepts. In the end of the research an intuitive and easy to use software was developed, where the user had only to put the values that he had and press a button to calculate and show the problem status for the imputed data, the software is capable to calculate the problem for 24 different sets of variables using the same numerical routine. This paper can be used as a basis for futures works that seek to use the knowledge from chemical engineering with software and graphical interface development.

Key words: Academic Software. Mass balance in separators. Python.

SUMÁRIO

1 INTRODUÇÃO

A utilização de computadores e softwares é cada dia mais comum na rotina de engenheiros e cientistas. E os engenheiros químicos estão se mostrando capazes de aproveitar as novas oportunidades que surgem devido as rápidas mudanças trazidas pela tecnologia para o mercado de trabalho (ASSIS, A. J. DE; LOPES, 2005). Um dos exemplos mais discutidos atualmente é o da indústria 4.0, onde em um futuro próximo espera-se desenvolver produtos e processos inteligentes que são monitorados e controlados apenas por algoritmos integrados com a internet (PREUVENEERS; ILIE-ZUDOR, 2017). Sendo assim, é cada dia mais importante que o profissional de engenharia domine as linguagens de programação.

Essa capacidade de adaptação do profissional de engenharia química pode ser explicada pela experiência adquirida durante a graduação onde muitas das disciplinas necessitam de estudos avançados e detalhados que envolvem modelos matemáticos complexos que não possuem solução analítica, ou seja, precisam ser resolvidos numericamente por rotinas computacionais (GREPINO, 2015).

Atualmente existe uma variedade muito grande de softwares, que podem ser utilizados em diversas áreas da engenharia química. Por exemplo, programas como MATLAB são utilizados na graduação para resoluções de problemas numéricos. Enquanto que programas como o ASPEN HYSYS podem ser utilizados para projetar plantas industriais inteiras (WONG; BARFORD, 2010)

Cada software apresenta uma complexidade e um propósito diferente. Tendo isso em vista, o presente trabalho busca desenvolver um software acadêmico para ser utilizado na resolução de balanços de massa em um separador de uma mistura binária sem reação química, sendo que o modelo do processo é descrito por um sistema de equações não lineares.

Para o desenvolvimento desse programa, é necessária a utilização de conhecimentos de três diferentes áreas, sendo elas: modelagem matemática do processo, soluções numéricas de sistemas de equações não lineares e utilização de linguagem de programação (utilizou-se Python, por sua versatilidade e facilidade de aplicação). A aplicação da linguagem de programação pode ser dividida em duas partes; na primeira desenvolve-se a solução numérica do problema e, na segunda parte tem-se a interface gráfica do programa para que o software seja de fácil utilização por qualquer usuário.

No presente estudo tem-se como objetivo geral o desenvolvimento de um programa intuitivo, robusto e eficiente que seja capaz de resolver vários problemas de balanço de massa em um separador de mistura binária (uma corrente de entrada e duas de saída), ou seja, o usuário poderá escolher qual o conjunto de incógnitas (correntes e/ou frações de cada componente) segundo o grau de liberdade do modelo matemático que descreve o comportamento do sistema, de forma similar ao que é possível nos módulos de simuladores modulares sequenciais.

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo são apresentadas as principais teorias que servem como base para a execução do trabalho, trazendo conceitos já consolidados da matemática e da engenharia, fazendo a conexão com os problemas mais atuais.

2.1 OPERAÇÕES UNITÁRIAS E SEPARADORES

Em um processo químico ideal seria misturada a quantidade exata de cada reagente que seriam completamente consumidos para formar o produto. Entretanto, em um processo real isso é praticamente impossível de ocorrer. Muitos fatores impedem que a conversão seja completa, por exemplo, a reação forma produtos secundários, além das impurezas presentes nos reagentes, limitações de equilíbrio químico, etc. O resultado disso é que o material que deixa o reator é uma mistura com o material de interesse junto de outros indesejados (RINARD, 1999).

Por isso é fundamental separar a mistura nos diversos componentes. O produto precisa ser separado de todo o restante da mistura e trazido para um grau aceitável de pureza para que possa ser comercializado. Se houver algum catalisador ou componente que possa ser reutilizado também é preciso fazer a separação. Sendo assim, o processo de separação é utilizado em praticamente toda indústria química. Sendo que alguns são relativamente simples, enquanto que outros constituem a maior parte de um processo, e podem elevar consideravelmente o custo final do produto (RINARD, 1999).

2.2 BALANÇO DE MASSA

O balanço de massa é a ferramenta fundamental da engenharia química, é a base para a análise e para o projeto plantas de processos químicos. O objetivo principal dos processos envolvidos nestas plantas é transformar uma matéria prima de menor valor em um produto com maior valor agregado e, em muitos casos, são gerados subprodutos indesejados que devem ser separados. E o balanço de massa é a ferramenta que permite ao engenheiro acompanhar tudo o que está entrando e saindo do processo, além de saber o que acontece no interior (RINARD, 1999).

Os processos químicos podem ser classificados em processos contínuos, onde há fluxo constante de matéria durante toda a operação; processos por batelada, em que

não há fluxo de massa durante o processo, ou seja, sistema fechado; processos semi-batelada, um sistema aberto em que material é adicionado durante a operação, porém nenhum material deixa o sistema.

Além disso os processos podem ser separados em processos transientes, em que pelo menos uma das variáveis de processo tem variação do valor com o tempo, ou em estado estacionário, onde todas as variáveis são constantes ao longo do tempo. A maioria dos processos industriais são modelados como sendo processos em estado estacionário apesar de apresentarem pequenas variações nas variáveis de controle que são originadas nas perturbações sofridas pelo sistema (FELDER, R. M.; ROUSSEAU, 2005; HIMMELBLAU, D. M.; RIGGS, 2014).

Para a modelagem do sistema é preciso partir da equação geral do balanço, que pode ser escrito de acordo com o que é mostrado na Equação (1).

$$entrada + geração - saída - consumo = acúmulo \tag{1}$$

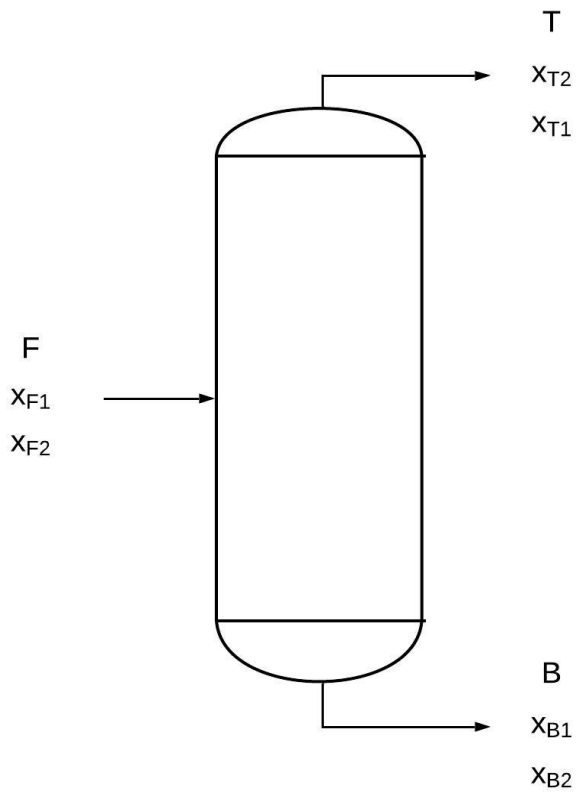
A Equação (1) pode ser aplicada para a conservação da massa total, para a massa e o número de mols de um componente e para a massa e o número de mols de uma espécie química (HIMMELBLAU, D. M.; RIGGS, 2014). Além disso, o balanço pode ser escrito na forma diferencial, que indica o que acontece no sistema em um instante determinado de tempo e na forma integral que descreve o que acontece no sistema entre dois instantes de tempo (FELDER, R. M.; ROUSSEAU, 2005).

Os termos de geração e consumo são incluídos nos balanços dos componentes químicos e no balanço de massa total, quando há reação química no sistema, caso contrário eles podem ser desprezados. Para processos em estado estacionário o termo de acúmulo também pode ser igualado a zero, porque nenhuma variável de processo sofre qualquer alteração com o tempo. Com essas simplificações restam apenas os termos de entrada e saída (FELDER, R. M.; ROUSSEAU, 2005).

A Figura 1 mostra um separador de mistura binária com uma corrente de entrada representada por F e duas correntes de saída, uma de topo T, e uma de fundo B, já as frações molares são representadas por x_{Fi} na corrente de entrada, x_{Ti} na corrente de topo e x_{Bi} na corrente de fundo, em que *i* representa o Índice do componente naquela corrente. Os índices 1, 2 representam os índices dos componentes n respectiva corrente.

Na modelagem do sistema em estudo é possível obter um balanço de massa global e um balanço de massa para cada componente presente no sistema, as Equações (2), (3) e (4) mostram o balanço de massa global e o balanço de massa para os componentes 1 e 2, respectivamente, já nas Equações (5), (6) e (7) estão apresentados as somas das frações dos componentes em cada uma das correntes, sendo que, a soma destas frações deve ser igual a unidade (1,0). Logo, o modelo matemático que representa um separador de uma mistura binária é representado pelas Equações (1)-(7)

Figura 1 – Representação do separador utilizado na modelagem.



Fonte: Autoria própria (2019).

Equação (2), balanço de massa global:

$$F - B - T = 0 \tag{2}$$

Equação (3), balanço de massa para o componente 1:

$$F \cdot x_{F1} - B \cdot x_{B1} - T \cdot x_{T1} = 0 \tag{3}$$

Equação (4), balanço de massa para o componente 2:

$$F \cdot x_{F2} - B \cdot x_{B2} - T \cdot x_{T2} = 0 \quad (4)$$

Equação (5), soma das frações molares na corrente de entrada:

$$x_{F1} + x_{F2} - 1 = 0 \quad (5)$$

Equação (6), soma das frações molares na corrente de topo:

$$x_{T1} + x_{T2} - 1 = 0 \quad (6)$$

Equação (7), soma das frações molares na corrente de fundo:

$$x_{B1} + x_{B2} - 1 = 0 \quad (7)$$

Para fazer a análise dos graus de liberdade do sistema é necessário saber o número de variáveis desconhecidas e em seguida subtrair do número de equações independentes (FELDER; ROUSSEAU, 2005). Para a solução do sistema de equações que é gerado com esse problema não é possível utilizar o balanço de massa global, pois o balanço de massa global é a soma dos balanços de massa por componente o que o torna linearmente dependente.

Portanto, tem-se um problema com cinco equações e nove variáveis, o que possibilita resolver o problema para cinco incógnitas, ou seja, é necessário ter o conhecimento do valor de quatro variáveis. Através da análise dos balanços de massa por componente, é possível perceber que esse sistema pode assumir formas não lineares, isso ocorre quando se tem como incógnita uma corrente e uma das frações daquela corrente. Sendo assim, um método numérico para sistemas não lineares se faz necessário.

2.3 MÉTODO DE NEWTON PARA A SOLUÇÃO DE SISTEMAS NÃO LINEARES

Sistemas de equações não lineares aparecem em muitos dos problemas reais. Assim, surge a necessidade de ter-se métodos eficientes para a resolução desse tipo sistemas.

O método de Newton é a ferramenta mais utilizada para a solução de sistemas não lineares, e pode ser utilizado de maneiras diferentes com base no objetivo do problema. Inicialmente o método foi desenvolvido para encontrar as raízes de uma equação não linear de uma variável. Para estender a solução para sistemas de equações não-lineares é preciso substituir o problema de uma variável por um vetor problema, que abrange todas as variáveis. Um sistema de equações não lineares, contendo n equações, pode ser representado pela Equação (8) (RUGGIERO; LOPES, 1997).

$$F(X)=\begin{bmatrix} f_1(X) \\ f_2(X) \\ \vdots \\ f_3(X) \end{bmatrix}=\begin{bmatrix} f_1(x_{1,}\, x_{2,\ldots}, x_n) \\ f_2(x_{1,}\, x_{2,\ldots}, x_n) \\ \vdots \\ f_n(x_{1,}\, x_{2,\ldots}, x_n) \end{bmatrix}=0 \qquad (8)$$

Em que X representa o vetor coluna das variáveis independentes, enquanto F(X) é o vetor do conjunto de equações não lineares. A resolução desse tipo de sistema é feita através de processos iterativos em que se parte de um ponto inicial, $X^{(0)}$, para gerar uma sequência de vetores, $X^{(k)}$. Para qualquer processo iterativo é necessário estabelecer um critério de parada onde $X^{(k)}$ é uma aproximação adequada para a solução exata X^{*} (RUGGIERO; LOPES, 1997; SPERANDIO; MENDES; SILVA, 2014).

Outra definição fundamental utilizada no método de Newton é a matriz Jacobiana. Essa matriz é formada pelas derivadas parciais das funções $f_i(X)$ em relação a cada uma das variáveis independentes, o formato da matriz jacobiana é demonstrada na Equação (9) (RUGGIERO; LOPES, 1997).

$$J(X)=\begin{vmatrix} \frac{(\partial f_1(X))}{(\partial x_1)} & \frac{(\partial f_1(X))}{(\partial x_2)} & \cdots & \frac{(\partial f_1(X))}{(\partial x_n)} \\ \frac{(\partial f_2(X))}{(\partial x_1)} & \frac{(\partial f_2(X))}{(\partial x_2)} & \cdots & \frac{(\partial f_2(X))}{(\partial x_n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{(\partial f_n(X))}{(\partial x_1)} & \frac{(\partial f_n(X))}{(\partial x_2)} & \cdots & \frac{(\partial f_n(X))}{(\partial x_n)} \end{vmatrix} \qquad (9)$$

Com a definição dos conceitos acima, tem-se as ferramentas para entender o método de Newton, que consiste em se tomar um modelo linear local da função em torno

de um ponto $X^{(k)}$, esse modelo é uma reta tangente a função F(X) no ponto $X^{(k)}$. E pode ser escrito conforme a Equação (10).

$$F(X) \approx F(X^k) + J(X^k)(X - X^k) \tag{10}$$

Para que se obtenha uma aproximação do próximo ponto X^{k+1} é necessário obter o zero de F(X), se a função for zerada na Equação (10) e os termos forem rearranjados obtém-se a equação de recorrência do método de Newton, que é apresentada na Equação (11) (RUGGIERO; LOPES, 1997).

$$-F(X^k) = J(X^k)(X - X^k) \tag{11}$$

O sistema linear definido na Equação (11) precisa ser resolvido para cada iteração do método de Newton fazendo com que o método seja computacionalmente custoso para um numero elevado de iterações, pois para que o sistema seja solucionado é necessário que a matriz jacobiana seja calculada para cada iteração. Na prática, isso é muito difícil de ser feito, para contornar esse problema foram criadas variações do método de Newton, um deles faz com que a matriz jacobiana seja mantida fixa por um número determinado de iterações e só depois seja recalculada, diminuindo muito o trabalho computacional necessário (SPERANDIO; MENDES; SILVA, 2014).

2.3.1 Relaxação no método de Newton e o método de Broyden

Uma desvantagem encontrada no método de Newton é que o método facilmente diverge se for utilizado na forma original. Para resolver esse problema é possível fazer a modificação mostrada na Equação (12), onde o escalar positivo *s,* que é chamado de fator de relaxação é utilizado para acelerar o processo de convergência, quando *s* assume valores maiores que um, e também pode ser usado para transformar um processo iterativo que está divergindo em um processo iterativo convergente. Os valores do fator de relaxação devem ser mantidos entre 0 e 2 (RUGGIERO; LOPES, 1997).

$$X^{(k+1)} = X^{(k)} + s \cdot \Delta X^{(k)} \tag{12}$$

Na aplicação do método de Broyden utiliza-se deste artefato apenas quando se faz necessário, ou seja, utiliza-se o fator de relaxação diferente da unidade apenas quado o processo iterativo está divergindo. O valor de *s* será alterado apenas se a desigualdade da Equação (13) não for verdadeira (BROYDEN, 1965).

$$\sqrt{\left(\sum_{j=1}^{n} f_j^2\left(X^{(k)}+s_1^{(k)}\Delta X^{(k)}\right)\right)}<\sqrt{\left(\sum_{j=1}^{n} f_j^2\left(X^{(k)}\right)\right)} \tag{13}$$

Para a correção do valor de *s* é utilizada a expressão mostrada na Equação (14), em que o η é a razão entre as normas euclidianas ao quadrado, calculado pela Equação (15) (BROYDEN, 1965).

$$s_2^{(k)}=\frac{\left(\sqrt{(1+6\,\eta)}-1\right)}{(3\,\eta)} \tag{14}$$

$$\eta=\frac{\left(\sum_{j=1}^{n} f_j^2\left(X^{(k)}+s_1^{(k)}\Delta X^{(k)}\right)\right)}{\left(\sum_{j=1}^{n} f_j^2\left(X^{(k)}\right)\right)} \tag{15}$$

2.4 LINGUAGEM DE PROGRAMAÇÃO

A linguagem de programação é a ferramenta que faz a comunicação entre o programador e o computador. As linguagens podem ser classificadas de acordo com a distância sintática da língua humana. Sendo que as linguagens entendidas pelas máquinas (linguagem de máquina) são chamadas de linguagens de baixo nível, enquanto que as linguagens desenvolvidas para serem entendidas pelo usuário são chamadas de linguagens de alto nível (PESSOA ; DIAS ; FORMIGA, 2014). A ferramenta responsável por traduzir a linguagem de alto nível para a de baixo nível é chamada de compilador (ZELLE, 2004).

Os programas eram escritos inicialmente em linguagens de baixo nível, fazendo com que o desenvolvimento de algoritmos se tornasse um processo lento e ineficiente,

diante disso surgiram as linguagens de alto nível, que elevaram a produtividade e democratizaram o desenvolvimento de programas (PESSOA ; DIAS ; FORMIGA, 2014).

2.4.1 Python e programação orientada a objetos

Existem várias linguagens de programação, dentre as quais tem-se Python, desenvolvido no começo dos anos 90 foi baseado na linguagem ABC, que era voltada para o ensino de programação (ZELLE, 2004). Ao longo do tempo, foi sofrendo modificações e ganhando novas versões, sua última edição é o Python 3 que foi lançado em 2008. Com o passar dos anos foram-se desenvolvendo ferramentas que tornaram o Python uma opção viável para projetos mais complexos. Atualmente existe uma comunidade cientifica bem estabelecida que conta com engenheiros, cientistas e pesquisadores que buscam sempre melhorar e promover o uso do Python na ciência (MILLMAN; AIVAZIS, 2011).

As ferramentas necessárias para programação científica não estão originalmente inclusas no Python. Sendo assim, foram criadas bibliotecas que contêm as rotinas computacionais que são utilizadas por várias áreas da ciência e das engenharias, sendo que, para a utilização dessas rotinas é necessário a instalação da biblioteca e a importação da rotina desejada. Os maiores exemplos de pacotes são o NumPy e o Scipy, que são pacotes muito sólidos e estão em constante atualização.

NumPy é a abreviação da sigla em inglês que traduzida significa Python Numérico, a principal função desse pacote é fornecer um novo tipo de dado chamado de arranjo (OLIPHANT, 2007). Os arranjos são uma espécie de lista multidimensional, feita originalmente para armazenar apenas um tipo de variável, porém através do NumPy os arranjos podem conter uma variável arbitrária em cada elemento, uma característica fundamental de um arranjo é que qualquer item contido nesse arranjo pode ser selecionado por indexação (FANGOHR, 2008).

A principal função desse tipo de organização de dados na programação científica é a utilização em problemas envolvendo matrizes (arranjos bidimensionais) e/ou vetores (arranjos unidimensionais). Além da introdução do arranjo o NumPy possui diversas outras ferramentas como a rotina para implementação da transformada de Fourier, operações de álgebra linear além de muitas outras funções úteis (FANGOHR, 2008)

A junção entre Python e NumPy já é bastante robusta e tem um leque de ferramentas considerável, entretanto em muitos casos somente essa combinação não é suficiente. Quando precisamos tratar de problemas mais avançados utilizando o Python é preciso que isso seja feito com o auxílio do SciPy (OLIPHANT, 2007)

O SciPy é extensão responsável por adicionar as ferramentas para otimização, funções especiais, processamento de imagens, integração numérica, resolução de equações diferenciais ordinárias, além de possibilitar a realização de operações com os arranjos introduzidos pelo NumPy. Por isso, a combinação do SciPy com NumPy possibilita com que o Python forneça as principais ferramentas que são utilizadas na engenharia matemática e física (FANGOHR, 2008; MILLMAN; AIVAZIS, 2011).

Outro conceito importante para o desenvolvimento do trabalho é o da programação orientada a objetos (POO). Esse tipo de programação foca na resolução de problemas na forma *bottom-up,* ou seja, primeiro se resolve os problemas menores para depois chegar em uma solução completa. Para o trabalho foram usados dois conceitos fundamentais da POO, o primeiro é o conceito de classes, que podem ser definidas como a descrição de um modelo que especifica as propriedades e o comportamento para um conjunto de objetos similares. O outro conceito utilizado foi o de objeto, que pode ser definido como instância que é gerada a partir de uma mesma classe (MACHADO, 2016).

2.4.2 Kivy

Kivy é uma biblioteca gratuita e de código aberto desenvolvida usando linguagem Python, essa biblioteca possui várias funcionalidades e permite o desenvolvimento rápido e eficiente de aplicações interativas compatíveis com várias plataformas. A velocidade de execução dos aplicativos desenvolvidos com o Kivy é semelhante a velocidade de execução dos aplicativos feitos usando as linguagens nativas dos sistemas, como por exemplo o Java para o Android e o Objective-C para iOS, porém o Kivy apresenta a vantagem de ser multiplataforma (ULLOA, 2015).

Existem muitas outras ferramentas que se assemelham ao Kivy, como o MT4J e o GestureWorks, baseados em Java e em ActionScript, respectivamente. Cada um apresenta uma tecnologia diferente e possui vantagens e desvantagens . O Kivy apresenta um bom desempenho porque o seu framework é escrito em linguagem C, por meio da biblioteca Cython, e em Python. Além disso a biblioteca Cython é usada para

acelerar todos os componentes relacionados com processamento gráfico, cálculo de matrizes, além de outros eventos relacionadas a programação orientada a objetos (VIRBEL; HANSEN; LOBUNETS, 2011). Isso faz com que o Kivy apresente um equilíbrio entre performance e portabilidade em diferentes plataformas (ULLOA, 2015).

Outra ferramenta importante do Kivy é a sua linguagem de programação própria, chamada de linguagem KV, que é baseada na sintaxe do Python o que torna a linguagem KV intuitiva e de fácil utilização. Apesar de não ser obrigatória no desenvolvimento de um aplicativo usando o Kivy, a linguagem KV pode ser muito útil, pois agiliza na escrita do código além de ser mais eficiente. A linguagem permite que os *widgets* sejam criados e vinculados às suas propriedades utilizando indentação, além de permitir a criação de rápidos protótipos e modificações instantâneas na interface do usuário (PHILLIPS, 2014).

3 METODOLOGIA

Neste capítulo são apresentados os métodos utilizados para o desenvolvimento da pesquisa, que é caracterizada como uma pesquisa de natureza aplicada, ou seja, envolve a aplicação prática dos conhecimentos científicos (PRODANOV; FREITAS, 2013).

O primeiro passo para alcançar o objetivo de desenvolver um software para resolução de balanços de massa em um separador de mistura binária com uma corrente de entrada e duas correntes de saída, foi a elaboração do projeto do software, nesse projeto inicial foram colocadas todas as principais ideias para que o programa final alcançasse os objetivos de ser robusto e fácil de usar.

A segunda parte do trabalho foi o desenvolvimento dos modelos matemáticos que descrevem o processo do separador de mistura binária e, posteriormente, as equações encontradas foram implementadas utilizando linguagem de programação. Nesta parte do trabalho também foram estudados os métodos numéricos para a resolução de sistemas de equações não lineares, dando ênfase ao método de Newton e as suas variações, para escolher qual é o modelo mais adequado para o sistema de equações do problema modelado.

Após a obtenção do modelo matemático do processo e do método de solução aplicou-se o método de pesquisa exploratória, não estruturada, para que se pudesse aprimorar os conhecimentos sobre a linguagem de programação e obter os conhecimentos necessários sobre a biblioteca Kivy para o desenvolvimento da interface gráfica.

Para o desenvolvimento do software o passo inicial foi a criação da interface gráfica utilizando a biblioteca Kivy, fez-se um esboço do que esperava-se obter com o programa para posteriormente iniciar o desenvolvimento no Python. A interface foi desenvolvida utilizando a linguagem kv que é própria do Kivy. Para tal cria-se dois arquivos, um da linguagem do Python no formato “.py” e outro no formato da linguagem kv. O arquivo do Python é o executado pelo interpretador, o arquivo na linguagem kv é apenas um documento de texto que serve como uma ferramenta facilitadora para deixar o programa mais organizado, para acessar o arquivo kv pelo documento do Python se utilizou uma ferramenta do Kivy chamada *Builder.*

Em paralelo ao desenvolvimento da interface gráfica foram elaboradas estratégias de tradução, para a linguagem de programação, do sistema de equações do problema,

além do método numérico escolhido. Nesta etapa foram realizados testes para implementar a rotina computacional que avalia o sistema de equações, posteriormente esse código desenvolvido foi complementado adicionando o método de solução numérica do sistema.

Para a parte da solução numérica, o método de Broyden não foi implementado, pois esse tipo de código é encontrado pronto e a implementação de rotinas numéricas não é o foco principal do trabalho, outro fator que influenciou na escolha de rotinas já implementadas é que essas rotinas são mais eficientes e robustas, pois já foram testadas e modificadas. A solução efetiva do problema ocorre no mesmo arquivo onde foram implementadas as equações em que foi necessária somente a importação das rotinas numéricas, devido a utilização do conceito de modularização do Python, em que os códigos são separados em arquivos, necessitando apenas a importação.

Após todos os testes e correções tem-se a parte gráfica e a parte das equações separadamente. Para a junção das duas rotinas foram utilizados princípios da programação orientada a objetos. Isso se faz necessário devido a arquitetura do Kivy, onde cada uma das telas do aplicativo é uma classe. Sendo assim, a execução da rotina de cálculos foi definida dentro de uma função que por sua vez é um objeto da classe, essa função é chamada ao pressionar o botão “calcular” na tela da interface gráfica do aplicativo.

Na etapa de junção dos códigos levou-se em conta os erros que ocorreram durante todos os testes, para contornar essas situações utilizou-se o tratamento de erros do Python que impede que o programa pare de funcionar na ocorrência de um erro. Outro aspecto importante nesta etapa do processo foi a construção do vetor das estimativas iniciais, foi definido por padrão do programa que se a corrente na entrada for uma variável a ser calculada a estimativa inicial é de 1000, para as correntes de topo e fundo foi definida uma estimativa inicial de 500 e para as frações se utilizou a estimativa inicial de 0.5 para todas.

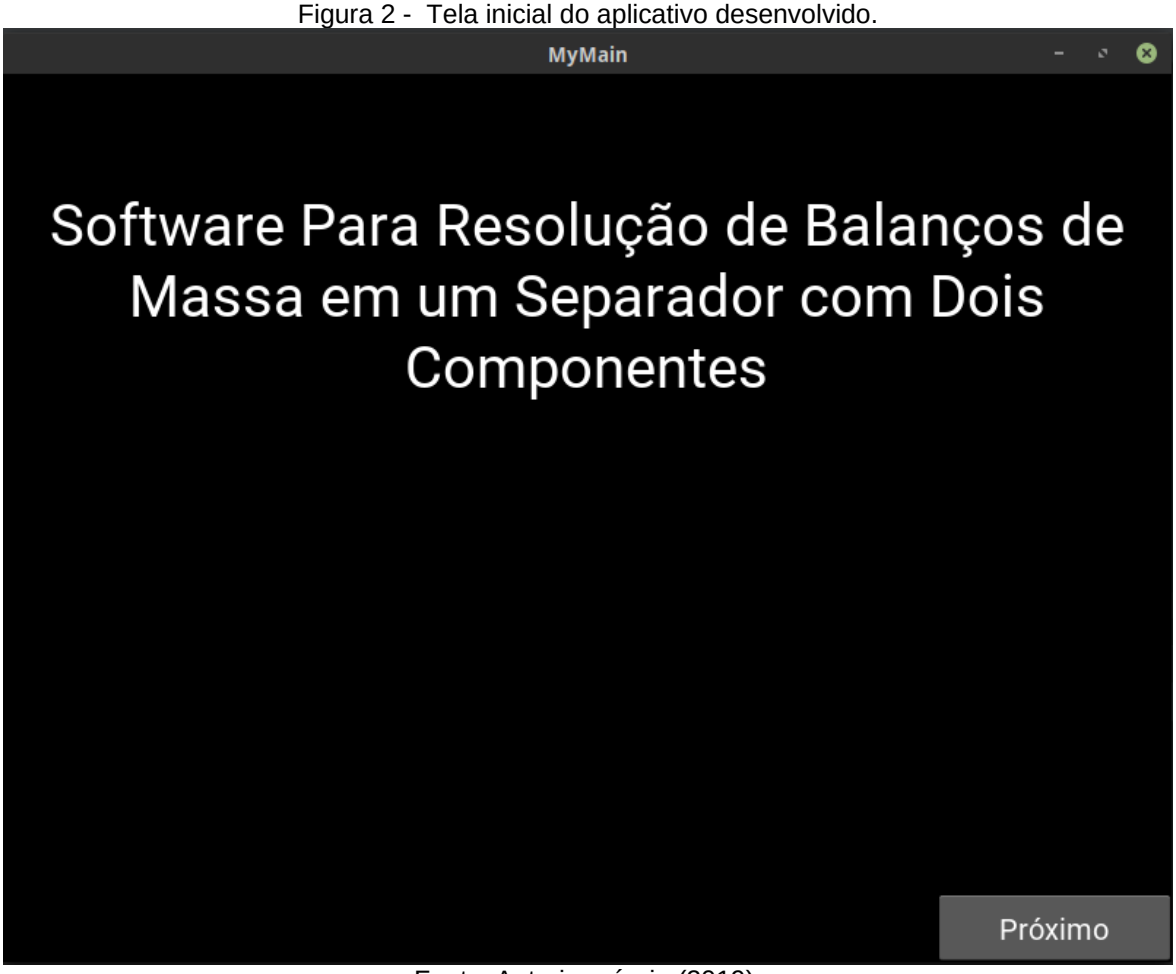
Independente do conjunto de variáveis definidas pelo usuário o software foi construído de modo que se use a mesma rotina para todos os casos, isso pode ocorrer porque o conjunto de equações será sempre o mesmo.

4 RESULTADOS E DISCUSSÕES

Neste capítulo serão apresentados todos os resultados obtidos com o estudo bem como as discussões referentes a relevância da pesquisa.

Através dos estudos realizados foi possível desenvolver um software acadêmico para a solução de balanços de massa em um separador de mistura binária (com uma corrente de entrada e duas correntes de saída). Para desenvolvimento do software foi necessária a utilização de métodos de solução de sistemas de equações não lineares, pois para algumas combinações de incógnitas o sistema de equações resultante possui esta característica. A execução do aplicativo desenvolvido é feita através do compilador da linguagem Python. Na Figura 2 é apresentada a página inicial do software composta pela apresentação do programa juntamente com o título e um botão para o usuário prosseguir para a página seguinte.

Figura 2 - Tela inicial do aplicativo desenvolvido.

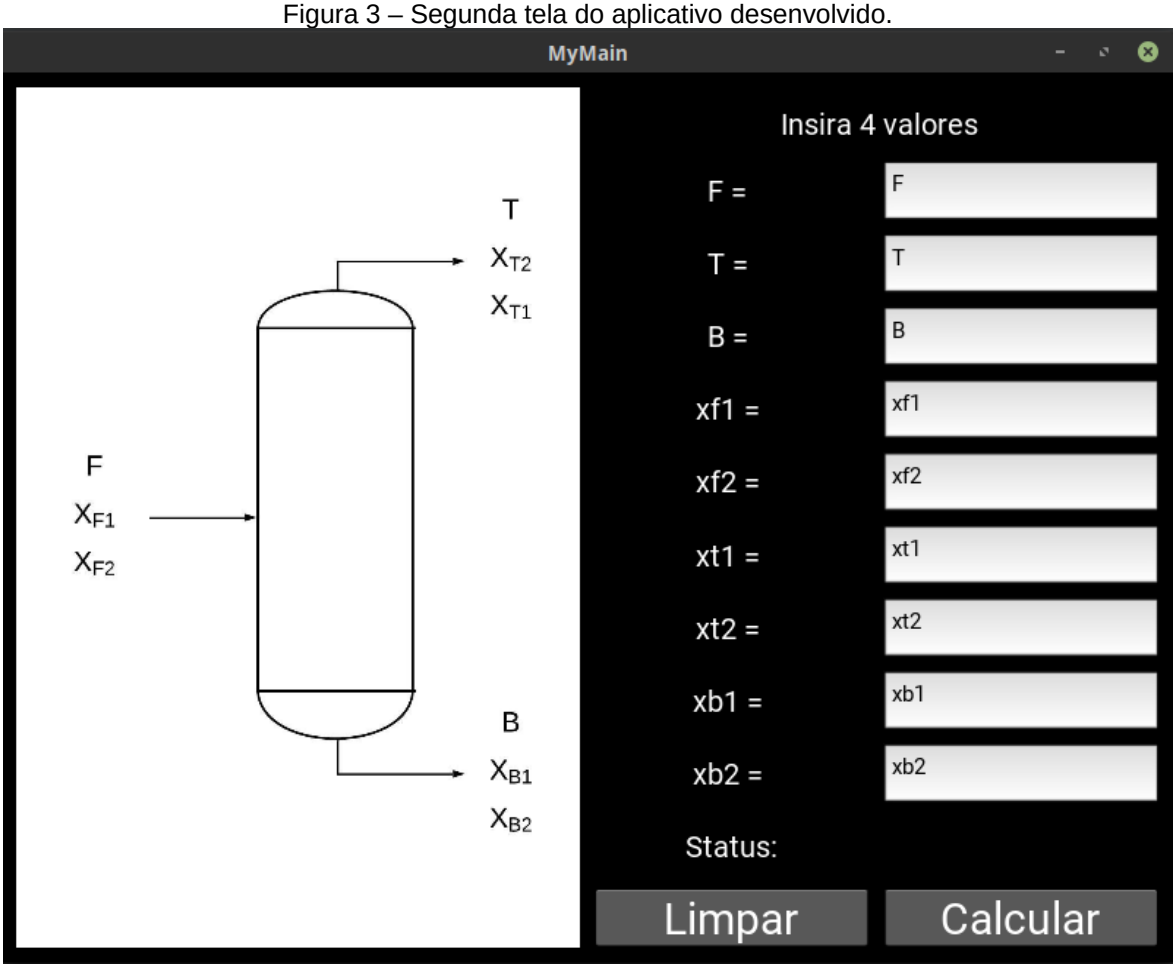


Fonte: Autoria própria (2019).

Após o clique no botão “próximo” na tela inicial a segunda tela será apresentada ao usuário, Figura 3. Nesta parte encontra-se uma figura representativa do separador com a notação utilizada para cada uma das variáveis, ao lado estão todas as variáveis com os seus respectivos campos para a entrada de texto, todos valores iniciais das variáveis foram fixados como sendo uma variável do tipo *string* com o valor do nome da variável, ou seja, o valor inicial de F é o próprio “F” e assim sucessivamente, logo abaixo das variáveis está o espaço para o status do problema. O campo status pode retornar dois valores, se o problema for resolvido a mensagem “Problema resolvido com sucesso” será apresentada, caso contrário será exibida a mensagem “Problema não resolvido”.

Ainda na Figura 3, acima dos campos das variáveis contém um texto que pede para o usuário explicitar quais são as quatro variáveis que ele possui, as cinco outras variáveis serão calculadas (se possível). Ainda nesta tela existem dois botões, o botão “calcular” que é responsável por ler os valores inseridos e executar a rotina numérica e o botão “limpar”, que define o valor das variáveis como sendo o valor inicial de cada uma.

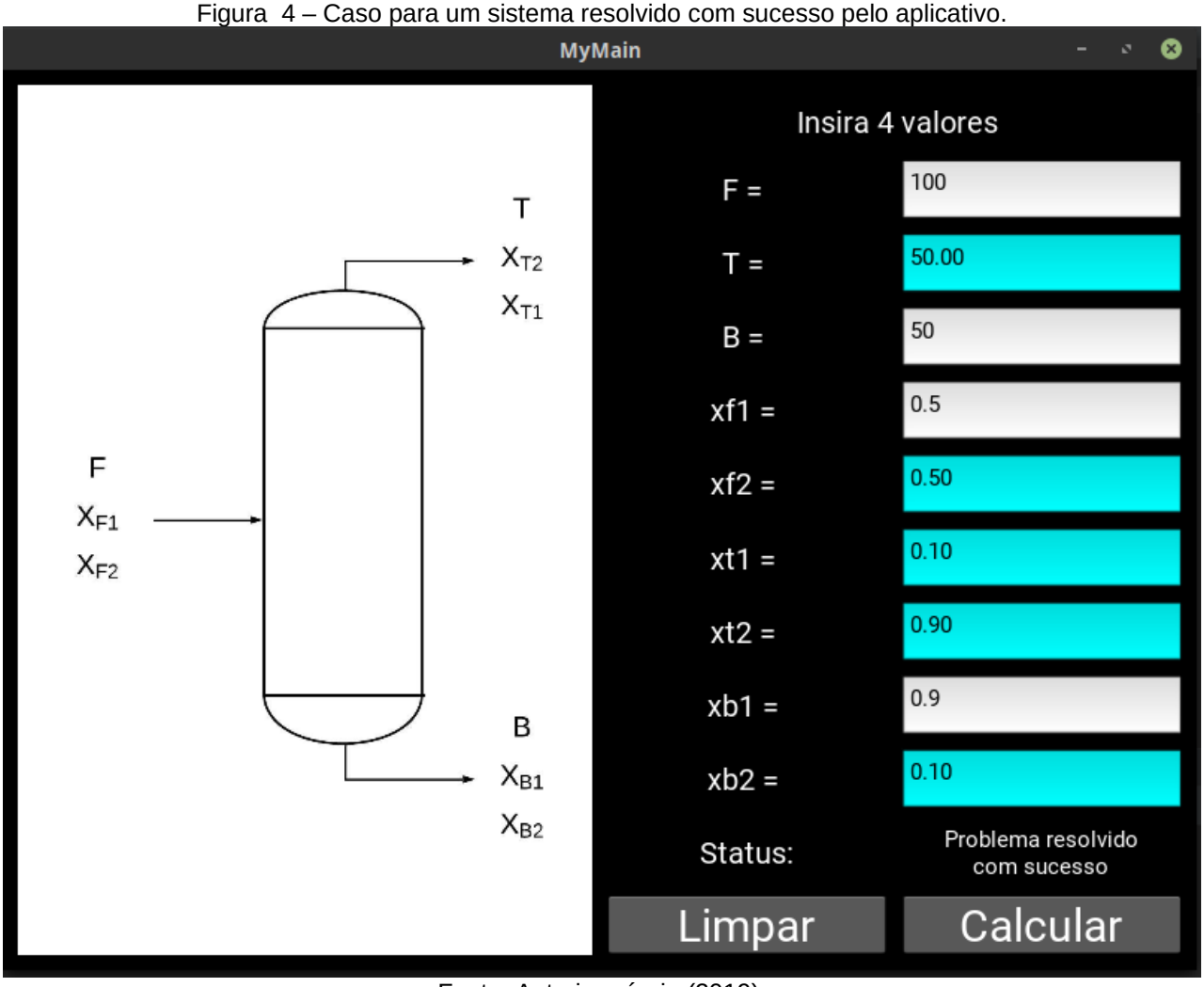
Figura 3 – Segunda tela do aplicativo desenvolvido.



Fonte: Autoria própria (2019).

Se o usuário inserir quatro valores válidos, e o problema tiver solução, o programa retorna os valores e a tela ficará da forma como é mostrado na Figura 4, que mostra o exemplo de solução do problema para os seguintes valores: F = 100, B = 30, XF1 = 0.5 e XB1 = 0.9. Como é possível observar o status para esse problema foi “Problema resolvido com sucesso”. Para facilitar a identificação do usuário a caixa de texto dos valores calculados pelo aplicativo mudará de cor quando o problema for resolvido com sucesso.

Figura 4 – Caso para um sistema resolvido com sucesso pelo aplicativo.



Fonte: Autoria própria (2019).

A situação com combinação das quatro variáveis definidas anteriormente é uma das vinte e quatro possibilidades de solução viável do problema. Todas as combinações possíveis das variáveis a serem calculadas são mostradas no Quadro 1. Essa quantidade limitada de soluções ocorre devido as restrições do modelo matemático, onde não se pode resolver o problema para os casos que são definidos, simultaneamente, valores

para as três correntes ou para as duas frações de cada corrente, isso ocorre devido aos graus de liberdade do sistema.

Se forem definidos os valores das três corrente ou das duas frações de uma mesma corrente, uma das equações do sistema se torna trivial, o que altera o grau de liberdade do sistema de equações, sendo assim não é possível resolver o sistema pela metodologia apresentada, pois ainda seria necessária a especificação do valor de mais uma variável. Através da análise do Quadro 1 é possível perceber que as soluções possíveis são compostas sempre por duas das correntes do separador e três composições.

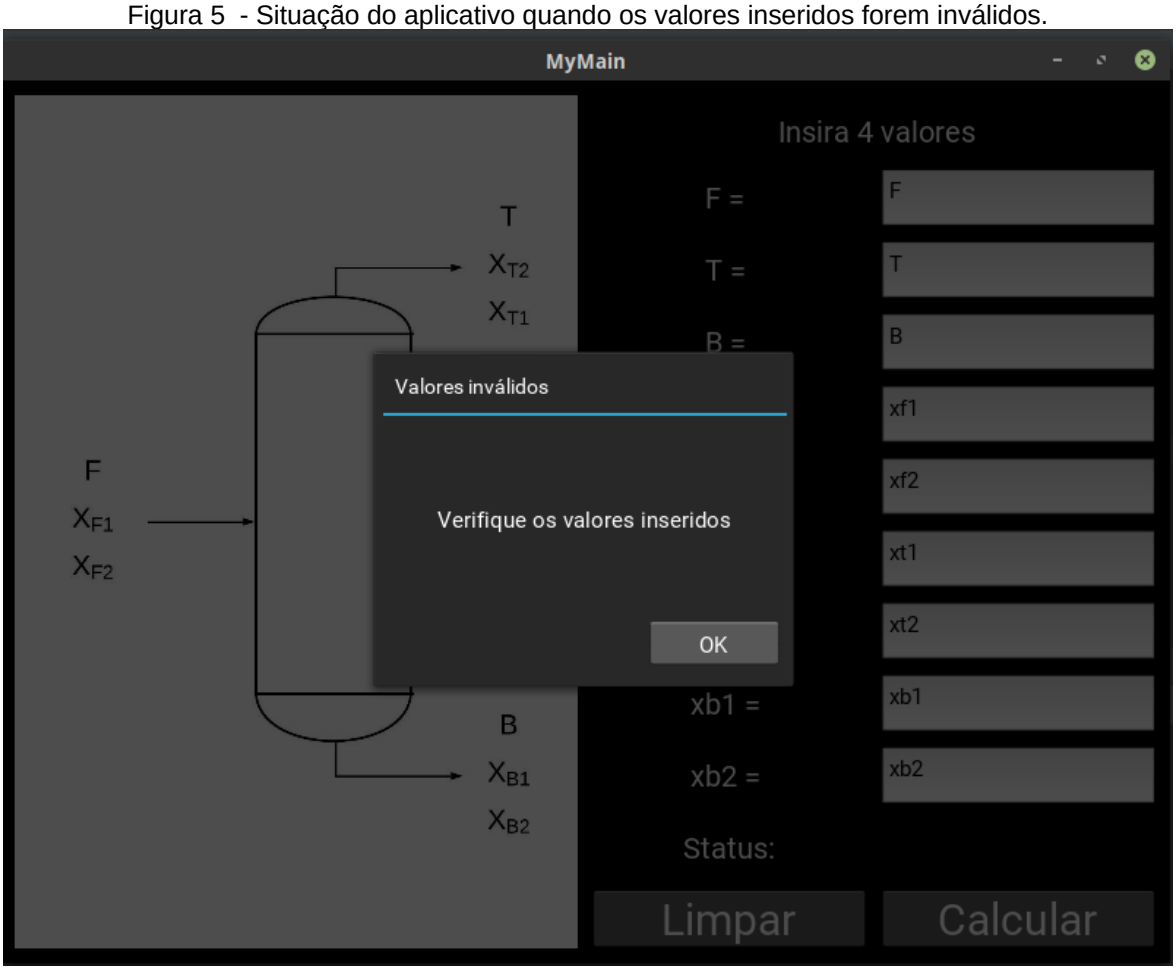
Quadro 1 – Possibilidades de combinações das variáveis a serem calculadas.

F, x_{F1}, B, x_{B1}, x_{T1}	F, x_{F2}, B, x_{B1}, x_{T1}	B, T, x_{F1}, x_{B1}, x_1
F, x_{F1}, B, x_{B1}, x_{T2}	F, x_{F2}, B, x_{B1}, x_{T2}	B, T, x_{F1}, x_{B1}, x_{T2}
F, x_{F1}, B, x_{B2}, x_{T1}	F, x_{F2}, B, x_{B2}, x_{T1}	B, T, x_{F1}, x_{B2}, x_{T1}
F, x_{F1}, B, x_{B2}, x_{T2}	F, x_{F2}, B, x_{B2}, x_{t2}	B, T, x_{F1}, x_{B2}, x_{T2}
F, x_{F1}, x_{B1}, T, x_{T1}	F, x_{F2}, x_{B1}, T, x_{T1}	B, T, x_{F2}, x_{B1}, x_{T1}
F, x_{F1}, x_{B1} T, x_{T2}	F, x_{F2}, x_{B1}, T, x_{T2}	B, T, x_{F2}, x_{B1}, x_{T2}
F, x_{F1}, x_{B2}, T, x_{T1}	F, x_{F2}, x_{B2}, T, x_{T1}	B, T, x_{F2}, x_{B2}, x_{T1}
F, x_{F1}, x_{B2}, T, x_{T2}	F, x_{F2}, x_{B2}, T, x_{T2}	B, T, x_{F2}, x_{B2}, x_{T2}

Fonte: Autoria Própria (2019).

Por fim, existem ainda duas outras situações possíveis para o aplicativo, a primeira, que é mostrada na Figura 5, que ocorre caso alguma inconsistência com os valores inseridos, ou seja, se o usuário definiu um número de variáveis diferentes de 4 ou se o usuário introduziu na entrada de texto alguma letra ou carácter inválido. Se isso ocorrer o programa irá disparar um *popup* informando ao usuário que os valores inseridos não são válidos e pedirá para que o usuário verifique os valores. O *popup* contém um botão “*ok*” após um clique no referido botão os valores são redefinidos aos valores iniciais automaticamente.

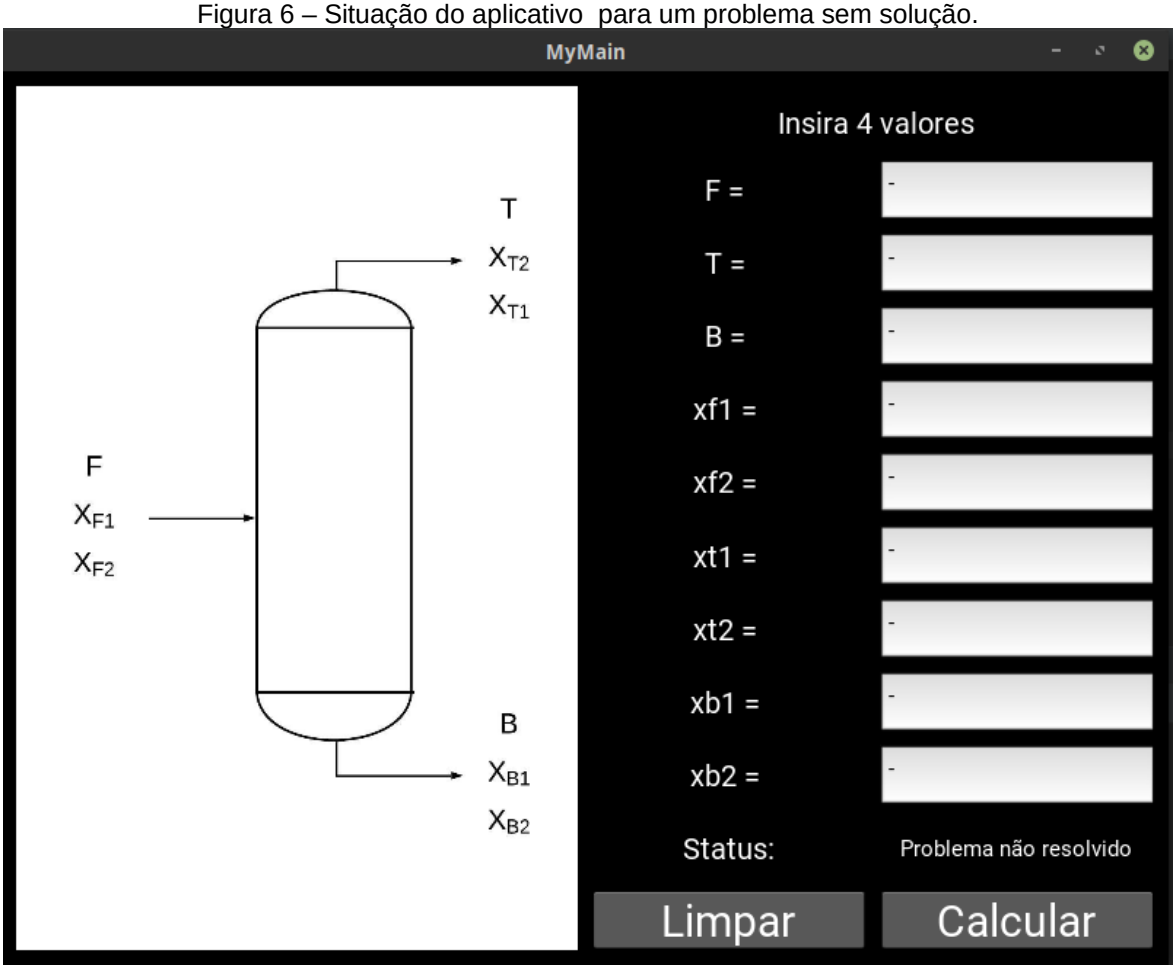
Figura 5 - Situação do aplicativo quando os valores inseridos forem inválidos.



Fonte: Autoria própria (2019).

O segundo caso ocorre quando o problema não tem solução, nesse caso o *status* irá retornar a mensagem “Problema não resolvido” e as variáveis terão seus valores alterados para “-”, Figura 6, e para que o usuário possa utilizar o aplicativo novamente o mesmo deverá utilizar o botão “Limpar”, que modificará os valores das variáveis para o seu valor inicial, e caso as caixas de texto tenham tido suas cores alteradas o botão “limpar” irá trocar a para coloração inicial.

Figura 6 – Situação do aplicativo para um problema sem solução.



Fonte: Autoria própria (2019).

Todo o código desenvolvido para o trabalho pode ser encontrados nos apêndices, no Apêndice A tem-se o algoritmo desenvolvido na linguagem Python. É nessa parte do código que os valores inseridos, *strings*, são lidos e convertidos para valores numéricos para que os cálculos sejam feitos e após a obtenção dos resultados os valores são novamente convertidos para *strings* para que possam ser impressos na interface gráfica.

A parte do algoritmo do programa que foi desenvolvida na linguagem kv é apresentada no Apêndice B. O que está contido nessa parte do código está relacionado com o desenvolvimento da interface gráfica e as suas modificações, onde tudo que for dinâmico na interface se comunica com o arquivo desenvolvido na linguagem Python. Para ambos os apêndices a indentação utilizada foi mantida, pois tanto na linguagem do Python quanto na linguagem kv o nível hierárquico dos blocos de código é determinado pela indentação, sendo assim a indentação é fundamental para o entendimento dos códigos desenvolvidos.

O programa desenvolvido se mostrou eficiente para o cálculo do problema proposto e não apresentou erros ou falhas durante os testes executados. Apesar de ser

limitado a um separador de dois componentes, o algoritmo é capaz de calcular com rapidez o problema para 24 cenários diferentes. Através da análise do Quadro 1 é perceptível que a grande vantagem do programa é que o usuário não necessita de conhecimento sobre linguagem de programação para sua utilização.

Em trabalhos futuros pode-se utilizar do algoritmo desenvolvido neste trabalho e acrescentar novos processos, como separadores com maior número de correntes e componentes, reatores, misturadores, entre outros. A modelagem de processos industriais e a implementação desses modelos em linguagem de programação são habilidades do engenheiro químico, e com o surgimento de ferramentas como o Kivy a criação de interfaces gráficas se torna mais eficiente, diante disso o trabalho desenvolvido pode servir como base para a criação de um simulador de processos químicos da UTFPR de Francisco Beltrão.

Outro aspecto a ser considerado é a análise dimensional dos problemas. Para o desenvolvimento do programa não foram levados em conta as possíveis inconsistências dimensionais dos problemas que serão resolvidos, sendo assim, esta análise deve ser feita pelo usuário do programa e/ou uma adequação feita para uma nova versão do software desenvolvido.

5 CONSIDERAÇÕES FINAIS

Neste trabalho projetou-se e desenvolveu-se um software capaz de resolver 24 possíveis combinações de incógnitas referêntes ao modelo de um separador de mistura binária (uma corrente de entrada e duas correntes de saída). Sendo que ao final obteve-se um software de fácil utilização, no qual o usuário pode escolher facilmente entre as possíveis combinações e informa os valores conhecidos, ou seja, tem sua utilização similar aos disponíveis em simuladores modulares sequenciais. Outro fator positivo do software foi o tratamento de erros sendo que para todos os testes executados o aplicativo não apresentou erros ou falhas.

O resultado desta pesquisa pode servir de base para futuros trabalhos na área, pois demostrou a possibilidade de se integrar os conhecimentos obtidos em diversas cadeiras do curso de engenharia química para desenvolver um software com interface gráfica. O código desenvolvido também pode ser utilizado para se criar um software mais completo, com a adição de outras rotinas e equipamentos.

REFERÊNCIAS

ASSIS, A. J. de; LOPES , L. C. O. **Free software Software for chemical engineer's educational needs** *In*: ENPROMER, 2005, Rio de Janeiro. 4th MERCOSUR Congress on Process Systems Engineering : 2nd MERCOSUR Congress on Chemical Engineering : Proceedings of ENPROMER 2005 [...].Rio de Janeiro: E-papers serviços editoriais, 2005. *E-book*.

BROYDEN, C. G. **A Class of Methods for Solving Nonlinear Simultaneous Equations.** Math. Comput., v. 19, n. 92, p. 577–593, 1965.

FANGOHR, H. Introduction to Python for Computational Science and Engineering. **Handbook of Computer Science and Engineering**, v. 2000, p. 86–110, 2008.

FELDER, R. M.; ROUSSEAU, R. W. **Princípios elementares dos processos químicos**. 3. ed. Rio de Janeiro, RJ: LTC, 2005. cap. 4, p. 81-109 ISBN 85-216-1429-2.

GREPINO, P. H. F.; RODRIGUES, F. de Á**. Utilização de softwares livres no ensino da engenharia química**. The Journal of Engineering and Exact Sciences, v. 1, n. 1, p. 16-29, jun. 2015. ISSN 2527-1075. Disponível em: <https://periodicos.ufv.br/ojs/jcec/article/view/2120>. Acesso em: 17 jun. 2019.

HIMMELBLAU, D. M.; RIGGS, J. L. **Engenharia química:** princípios e cálculos. 8. ed. Rio de Janeiro, RJ: LTC, 2014. cap. 3 e 4, p. 99-152.

MACHADO, R. P. **Desenvolvimento de software, v.3 : programação de sistemas web orientada a objetos em Java**, 2016. Disponível em: <http://search.ebscohost.com/login.aspx?direct=true&db=edsmib&AN=edsmib.000007550&lang=pt-br&site=eds-live&scope=site>. Acesso em: 11 nov. 2019.

MILLMAN, K. J.; AIVAZIS, M. Python for scientists and engineers. **Computing in Science and Engineering**, v. 13, n. 2, p. 9–12, 2011.

OLIPHANT, T. E. Python for Scientific Computing. **Computing in Science & Engineering**, v. 9, p. 10–20, 2007.

PESSOA, B. J. S.; DIAS JR., J.; FORMIGA, A. A.. **Introdução à Programação**. 1. ed. João Pessoa: Editora da UFPB, 2013. cap. 2, pag. 12-13.

PHILLIPS, D. **Creating Apps in Kivy**. 1. ed. Sebastopol: O'Reilly Media, 2014.

PREUVENEERS, D.; ILIE-ZUDOR, E. The intelligent industry of the future: A survey on emerging trends, research challenges and opportunities in Industry 4.0. **Journal of Ambient Intelligence and Smart Environments**, v. 9, n. 3, p. 287–298, 2017.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico**. 2. ed. Novo Hamburgo: Feevale, 2013.

Disponível em: <http://www.feevale.br/Comum/midias/8807f05a-14d0-4d5b-b1ad-1538f3aef538/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf>. Acesso em: 11 nov. 2019.

RINARD, I. **Material Balance Notes**, Revisão 3, Department of Chemical Engineering, City College of CUNY and Project ECSEL, 1999.

RUGGIERO, M. A. G.; LOPES, V. L. da R. **Cálculo numérico: aspectos teóricos e computacionais**. 2. ed. São Paulo, SP: Makron Books, 1997. cap. 4, 406 p. 192-200. ISBN 8534602042.

SPERANDIO, D.; MENDES, J. T.; SILVA, L. H. M. e. **Cálculo numérico.** 2.ed. São Paulo, SP: Pearson, 2014. cap 4, pag. 124-128 ISBN 9788543006536.

ULLOA, R. **Kivy – Interactive Applications and Games in Python**. 2 ed. Birmingham - UK: Packt Publishing Ltd, 2015. cap. 1, pag. 1. ISBN 978-1-78528-692-6.

VIRBEL, M.; HANSEN, T. E.; LOBUNETS, O. **Kivy-A Framework for Rapid Creation of Innovative User Interfaces.** Mensch & Computer Workshopband, p. 69–73, 2011.

WONG, K. W. W.; BARFORD, J. P. Teaching Excel VBA as a problem solving tool for chemical engineering core courses. **Education for Chemical Engineers**, v. 5, n. 4. e72--e77, 2010.

ZELLE, J. **Python Programming: an introduction to computer science**. 3 ed. Portland: Franklin, Beedle & Associates Inc., 2004. cap. 1, pag. 8.

APÊNDICE A – Parte do código do programa desenvolvida na linguagem Python

```
from kivy.app import App
from kivy.lang import Builder
from kivy.uix.screenmanager import Screen, ScreenManager
import numpy as np
from Broyden import Broyden_method
from kivy.uix.popup import Popup
class FirstWindow(Screen):
    pass
class SecondWindow(Screen):
    def get_num(self, a):
        f = self.f.text
        t = self.t.text
        b = self.b.text
        xf1 = self.xf1.text
        xf2 = self.xf2.text
        xt1 = self.xt1.text
        xt2 = self.xt2.text
        xb1 = self.xb1.text
        xb2 = self.xb2.text
        v_val = []
        v_var = []
        index_val = []
        index_var = []
        val_0 = []

        if f == 'f' or f =='F' or f == '':
            v_var.append('F')
            index_var.append(0)
            val_0.append(1000)
        else:
            try:
                v_val.append(float(f))
```

```
            index_val.append(0)
        except:
            po = PopUp()
            po.open()
            self.t.text = 'T'
            self.b.text = 'B'
            self.f.text = 'F'
            self.xt1.text = 'xt1'
            self.xt2.text = 'xt2'
            self.xb1.text = 'xb1'
            self.xb2.text = 'xb2'
            self.xf1.text = 'xf1'
            self.xf2.text = 'xf2'

    if b == 'b' or b == 'B' or b == '':
        v_var.append('B')
        index_var.append(3)
        val_0.append(500)
    else:
        try:
            v_val.append(float(b))
            index_val.append(3)
        except:
            po = PopUp()
            po.open()
            self.t.text = 'T'
            self.b.text = 'B'
            self.f.text = 'F'
            self.xt1.text = 'xt1'
            self.xt2.text = 'xt2'
            self.xb1.text = 'xb1'
            self.xb2.text = 'xb2'
            self.xf1.text = 'xf1'
            self.xf2.text = 'xf2'
```

```
if t == 't' or  t== 'T' or t == '':
    v_var.append('T')
    index_var.append(6)
    val_0.append(500)
else:
    try:
        v_val.append(float(t))
        index_val.append(6)
    except:
        po = PopUp()
        po.open()
        self.t.text = 'T'
        self.b.text = 'B'
        self.f.text = 'F'
        self.xt1.text = 'xt1'
        self.xt2.text = 'xt2'
        self.xb1.text = 'xb1'
        self.xb2.text = 'xb2'
        self.xf1.text = 'xf1'
        self.xf2.text = 'xf2'

if xf1 == 'xf1'or xf1 == 'XF1' or xf1 == '':
    v_var.append('xf1')
    index_var.append(1)
    val_0.append(0.5)
else:
    try:
        v_val.append(float(xf1))
        index_val.append(1)
    except:
        po = PopUp()
        po.open()
        self.t.text = 'T'
```

```
            self.b.text = 'B'
            self.f.text = 'F'
            self.xt1.text = 'xt1'
            self.xt2.text = 'xt2'
            self.xb1.text = 'xb1'
            self.xb2.text = 'xb2'
            self.xf1.text = 'xf1'
            self.xf2.text = 'xf2'

    if xf2 == 'xf2' or xf2 == 'XF2' or xf2 == '':
        v_var.append('xf2')
        index_var.append(2)
        val_0.append(0.5)
    else:
        try:
            v_val.append(float(xf2))
            index_val.append(2)
        except:
            po = PopUp()
            po.open()
            self.t.text = 'T'
            self.b.text = 'B'
            self.f.text = 'F'
            self.xt1.text = 'xt1'
            self.xt2.text = 'xt2'
            self.xb1.text = 'xb1'
            self.xb2.text = 'xb2'
            self.xf1.text = 'xf1'
            self.xf2.text = 'xf2'

    if xb1 == 'xb1' or xb1 == 'XB1' or xb1 == '':
        xb1 = 'xb1'
        v_var.append(xb1)
        index_var.append(4)
```

```
    val_0.append(0.5)
else:
    try:
        v_val.append(float(xb1))
        index_val.append(4)
    except:
        po = PopUp()
        po.open()
        self.t.text = 'T'
        self.b.text = 'B'
        self.f.text = 'F'
        self.xt1.text = 'xt1'
        self.xt2.text = 'xt2'
        self.xb1.text = 'xb1'
        self.xb2.text = 'xb2'
        self.xf1.text = 'xf1'
        self.xf2.text = 'xf2'

if xb2 == 'xb2' or xb2 == 'XB2' or xb2 == '':
    xb2 = 'xb2'
    v_var.append(xb2)
    index_var.append(5)
    val_0.append(0.5)
else:
    try:
        v_val.append(float(xb2))
        index_val.append(5)
    except:
        po = PopUp()
        po.open()
        self.t.text = 'T'
        self.b.text = 'B'
        self.f.text = 'F'
        self.xt1.text = 'xt1'
```

```
            self.xt2.text = 'xt2'
            self.xb1.text = 'xb1'
            self.xb2.text = 'xb2'
            self.xf1.text = 'xf1'
            self.xf2.text = 'xf2'

    if xt1 == 'xt1' or xt1 == 'XT1' or xt1 == '':
        xt1 = 'xt1'
        v_var.append(xt1)
        index_var.append(7)
        val_0.append(0.5)
    else:
        try:
            v_val.append(float(xt1))
            index_val.append(7)
        except:
            po = PopUp()
            po.open()
            self.t.text = 'T'
            self.b.text = 'B'
            self.f.text = 'F'
            self.xt1.text = 'xt1'
            self.xt2.text = 'xt2'
            self.xb1.text = 'xb1'
            self.xb2.text = 'xb2'
            self.xf1.text = 'xf1'
            self.xf2.text = 'xf2'

    if xt2 == 'xt2' or xt2 == 'XT2' or xt2 == '':
        xt2 = 'xt2'
        v_var.append(xt2)
        index_var.append(8)
        val_0.append(0.5)
    else:
```

```
    try:
        v_val.append(float(xt2))
        index_val.append(8)
    except:
        po = PopUp()
        po.open()
        self.t.text = 'T'
        self.b.text = 'B'
        self.f.text = 'F'
        self.xt1.text = 'xt1'
        self.xt2.text = 'xt2'
        self.xb1.text = 'xb1'
        self.xb2.text = 'xb2'
        self.xf1.text = 'xf1'
        self.xf2.text = 'xf2'
i_val = 0
i_var = 0
for i in v_var:
    i_var += 1
for i in v_val:
    i_val += 1

if i_val != 4 or i_var != 5:

    po = PopUp()
    po.open()
    self.t.text = 'T'
    self.b.text = 'B'
    self.f.text = 'F'
    self.xt1.text = 'xt1'
    self.xt2.text = 'xt2'
    self.xb1.text = 'xb1'
    self.xb2.text = 'xb2'
    self.xf1.text = 'xf1'
```

```
    self.xf2.text = 'xf2'

else:
    try:
        def FUNC_TCC(X, N):

            global N_fix, Inc_index, Fix_index, X_fix

            X_fix = np.array([v_val[0], v_val[1], v_val[2], v_val[3]])
            Inc_index = index_var
            Fix_index = index_val
            N_fix = 4

            F_val = np.zeros(N)

            Y = np.zeros(N + N_fix)

            for i in range(0, N):
                Y[Inc_index[i]] = X[i]

            for i in range(0, N_fix):
                Y[Fix_index[i]] = X_fix[i]

            F_val[0] = Y[0] * Y[1] - Y[3] * Y[4] - Y[6] * Y[7]
            F_val[1] = Y[0] * Y[2] - Y[3] * Y[5] - Y[6] * Y[8]
            F_val[2] = Y[1] + Y[2] - 1.0
            F_val[3] = Y[4] + Y[5] - 1.0
            F_val[4] = Y[7] + Y[8] - 1.0

            np.set_printoptions(precision=2)
            return F_val

        X0 = np.array([val_0[0], val_0[1], val_0[2], val_0[3], val_0[4]])
        N_inc = 5
```

```
        res = Broyden_method(FUNC_TCC, N_inc, X0, MaxIter=7000, h=1e-10, Tol=1e-
7)

        if index_var[0] == 0:
            self.f.text = str(res[0])
            self.f.background_color = 0, 1, 1, 1
        elif index_var[0] == 1:
            self.xf1.text = str(res[0])
            self.xf1.background_color = 0, 1, 1, 1
        elif index_var[0] == 2:
            self.xf2.text = str(res[0])
            self.xf2.background_color = 0, 1, 1, 1
        elif index_var[0] == 3:
            self.b.text = str(res[0])
            self.b.background_color = 0, 1, 1, 1
        elif index_var[0] == 4:
            self.xb1.text = str(res[0])
            self.xb1.background_color = 0, 1, 1, 1
        elif index_var[0] == 5:
            self.xb2.text = str(res[0])
            self.xb2.background_color = 0, 1, 1, 1
        elif index_var[0] == 6:
            self.t.text = str(res[0])
            self.t.background_color = 0, 1, 1, 1
        elif index_var[0] == 7:
            self.xt1.text = str(res[0])
            self.xt1.background_color = 0, 1, 1, 1
        elif index_var[0] == 8:
            self.xt2.text = str(res[0])
            self.xt2.background_color = 0, 1, 1, 1

        if index_var[1] == 0:
            self.f.text = str(res[1])
            self.f.background_color = 0, 1, 1, 1
```

```
elif index_var[1] == 1:
    self.xf1.text = str(res[1])
    self.xf1.background_color = 0, 1, 1, 1
elif index_var[1] == 2:
    self.xf2.text = str(res[1])
    self.xf2.background_color = 0, 1, 1, 1
elif index_var[1] == 3:
    self.b.text = str(res[1])
    self.b.background_color = 0, 1, 1, 1
elif index_var[1] == 4:
    self.xb1.text = str(res[1])
    self.xb1.background_color = 0, 1, 1, 1
elif index_var[1] == 5:
    self.xb2.text = str(res[1])
    self.xb2.background_color = 0, 1, 1, 1
elif index_var[1] == 6:
    self.t.text = str(res[1])
    self.t.background_color = 0, 1, 1, 1
elif index_var[1] == 7:
    self.xt1.text = str(res[1])
    self.xt1.background_color = 0, 1, 1, 1
elif index_var[1] == 8:
    self.xt2.text = str(res[1])
    self.xt2.background_color = 0, 1, 1, 1

if index_var[2] == 0:
    self.f.text = str(res[2])
    self.f.background_color = 0, 1, 1, 1
elif index_var[2] == 1:
    self.xf1.text = str(res[2])
    self.xf1.background_color = 0, 1, 1, 1
elif index_var[2] == 2:
    self.xf2.text = str(res[2])
    self.xf2.background_color = 0, 1, 1, 1
```

```
elif index_var[2] == 3:
    self.b.text = str(res[2])
    self.b.background_color = 0, 1, 1, 1
elif index_var[2] == 4:
    self.xb1.text = str(res[2])
    self.xb1.background_color = 0, 1, 1, 1
elif index_var[2] == 5:
    self.xb2.text = str(res[2])
    self.xb2.background_color = 0, 1, 1, 1
elif index_var[2] == 6:
    self.t.text = str(res[2])
    self.t.background_color = 0, 1, 1, 1
elif index_var[2] == 7:
    self.xt1.text = str(res[2])
    self.xt1.background_color = 0, 1, 1, 1
elif index_var[2] == 8:
    self.xt2.text = str(res[2])
    self.xt2.background_color = 0, 1, 1, 1

if index_var[3] == 0:
    self.f.text = str(res[3])
    self.f.background_color = 0, 1, 1, 1
elif index_var[3] == 1:
    self.xf1.text = str(res[3])
    self.xf1.background_color = 0, 1, 1, 1
elif index_var[3] == 2:
    self.xf2.text = str(res[3])
    self.xf2.background_color = 0, 1, 1, 1
elif index_var[3] == 3:
    self.b.text = str(res[3])
    self.b.background_color = 0, 1, 1, 1
elif index_var[3] == 4:
    self.xb1.text = str(res[3])
    self.xb1.background_color = 0, 1, 1, 1
```

```
elif index_var[3] == 5:
    self.xb2.text = str(res[3])
    self.xb2.background_color = 0, 1, 1, 1
elif index_var[3] == 6:
    self.t.text = str(res[3])
    self.t.background_color = 0, 1, 1, 1
elif index_var[3] == 7:
    self.xt1.text = str(res[3])
    self.xt1.background_color = 0, 1, 1, 1
elif index_var[3] == 8:
    self.xt2.text = str(res[3])
    self.xt2.background_color = 0, 1, 1, 1

if index_var[4] == 0:
    self.f.text = str(res[4])
    self.f.background_color = 0, 1, 1, 1
elif index_var[4] == 1:
    self.xf1.text = str(res[4])
    self.xf1.background_color = 0, 1, 1, 1
elif index_var[4] == 2:
    self.xf2.text = str(res[4])
    self.xf2.background_color = 0, 1, 1, 1
elif index_var[4] == 3:
    self.b.text = str(res[4])
    self.b.background_color = 0, 1, 1, 1
elif index_var[4] == 4:
    self.xb1.text = str(res[4])
    self.xb1.background_color = 0, 1, 1, 1
elif index_var[4] == 5:
    self.xb2.text = str(res[4])
    self.xb2.background_color = 0, 1, 1, 1
elif index_var[4] == 6:
    self.t.text = str(res[4])
    self.t.background_color = 0, 1, 1, 1
```

```
            elif index_var[4] == 7:
                self.xt1.text = str(res[4])
                self.xt1.background_color = 0, 1, 1, 1
            elif index_var[4] == 8:
                self.xt2.text = str(res[4])
                self.xt2.background_color = 0, 1, 1, 1

            if res[5] is True:
                self.status.text = 'Problema resolvido\ncom sucesso'
            else:
                self.status.text = 'Problema não resolvido'
                self.t.text = '-'
                self.b.text = '-'
                self.f.text = '-'
                self.xt1.text = '-'
                self.xt2.text = '-'
                self.xb1.text = '-'
                self.xb2.text = '-'
                self.xf1.text = '-'
                self.xf2.text = '-'

        except:
                self.status.text = 'Problema não resolvido'
                self.t.text = '-'
                self.b.text = '-'
                self.f.text = '-'
                self.xt1.text = '-'
                self.xt2.text = '-'
                self.xb1.text = '-'
                self.xb2.text = '-'
                self.xf1.text = '-'
                self.xf2.text = '-'

def limpar(self, d):
```

```
        self.t.text = 'T'
        self.t.background_color = 1, 1, 1, 1
        self.b.text = 'B'
        self.b.background_color = 1, 1, 1, 1
        self.f.text = 'F'
        self.f.background_color = 1, 1, 1, 1
        self.xt1.text = 'xt1'
        self.xt1.background_color = 1, 1, 1, 1
        self.xt2.text = 'xt2'
        self.xt2.background_color = 1, 1, 1, 1
        self.xb1.text = 'xb1'
        self.xb1.background_color = 1, 1, 1, 1
        self.xb2.text = 'xb2'
        self.xb2.background_color = 1, 1, 1, 1
        self.xf1.text = 'xf1'
        self.xf1.background_color = 1, 1, 1, 1
        self.xf2.text = 'xf2'
        self.xf2.background_color = 1, 1, 1, 1
        self.status.text = ''


class WindowManagement(ScreenManager):
    pass

class PopUp(Popup):
    pass

kv = Builder.load_file("projeto.kv")

class MyMainApp(App):
    def build(self):
        return kv
```

```
if __name__ == "__main__":
    MyMainApp().run()
```

APÊNDICE B - Parte do código do programa desenvolvida na linguagem kv

```
WindowManagement:
    FirstWindow:
    SecondWindow:
<FirstWindow>
    name:"First Window"
    GridLayout:
        cols:1
        Label:
            text:"Software Para Resolução de Balanços de\nMassa em um Separador com
Dois\nComponentes"
            halign: 'center'
            font_size: 40
        FloatLayout:
            Button:
                text: "Próximo"
                pos_hint: {"x":0.8, "y":0}
                size_hint: 0.2,0.15
                font_size:20
                on_release:
                    app.root.current = "Second Window"
                    root.manager.transition.direction = "left"
<SecondWindow>
    f: _f.__self__
    t: _t.__self__
    b: _b.__self__
    xf1:_xf1.__self__
    xf2:_xf2.__self__
    xt1:_xt1.__self__
    xt2:_xt2.__self__
    xb1:_xb1.__self__
    xb2:_xb2.__self__
    status: _status.__self__
```

```
name: "Second Window"
canvas.before:
    Color:
        rgba: 0, 0, 0, 1
    Rectangle:
        pos: self.pos
        size: self.size
GridLayout:
    cols:2
    padding:10
    spacing:10
    FloatLayout:
        canvas.before:
            Color:
                rgba: 1,1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        Image:
            source:'Separador.jpeg'
            pos_hint: {"x":0, "y":0}
            allow_strech:True
    GridLayout:
        rows:2
        Label:
            text: " Insira 4 valores"
            size_hint_y: None
            height:50
            font_size: 20
        GridLayout:
            cols: 2
            spacing:10
            Label:
                text:"F = "
```

```
    font_size:20
    color:1,1,1,1
TextInput:
    id: _f
    multiline: False
    text: 'F'
Label:
    text:"T = "
    font_size:20
    color:1,1,1,1
TextInput:
    id: _t
    text: 'T'
    multiline: False
Label:
    text:"B = "
    font_size:20
    color:1,1,1,1
TextInput:
    id: _b
    text: 'B'
    multiline: False
Label:
    text:"xf1 = "
    font_size:20
    color:1,1,1,1
TextInput:
    id: _xf1
    text: 'xf1'
    multiline: False
Label:
    text:"xf2 = "
    font_size:20
    color:1,1,1,1
```

```
TextInput:
    id: _xf2
    text: 'xf2'
    multiline: False
Label:
    text:"xt1 = "
    font_size:20
    color:1,1,1,1
TextInput:
    id: _xt1
    text: 'xt1'
    multiline: False
Label:
    text:"xt2 = "
    font_size:20
    color:1,1,1,1
TextInput:
    id: _xt2
    text: 'xt2'
    multiline: False
Label:
    text:"xb1 = "
    font_size:20
    color:1,1,1,1
TextInput:
    id: _xb1
    text:'xb1'
    multiline: False
Label:
    text:"xb2 = "
    font_size:20
    color:1,1,1,1
TextInput:
    id: _xb2
```

```
            text: 'xb2'
            multiline: False
        Label:
            text:"Status:"
            font_size:20
            color:1,1,1,1
        Label:
            id: _status
            text: ''
            halign: 'center'
            font_size:15
            color:1,1,1,1
        Button:
            font_size:30
            color: 1,1,1,1
            text:"Limpar"
            on_release:
                root.limpar(*args)
        Button:
            font_size:30
            color: 1,1,1,1
            text:"Calcular"
            on_press:
                root.get_num(*args)
<PopUp>
    name: 'pop_up'
    auto_dismiss: False
    size_hint: None, None
    size: 300, 240
    title: "Valores inválidos"
    BoxLayout:
        orientation: 'vertical'
        padding:10
        spacing:10
```

```
Label:
    text:'Verifique os valores inseridos'

Button:
    text: 'OK'
    size_hint_x: None
    size_hint_y: None
    size_hint:(0.35, 0.25)
    pos_hint:{'x': 0.67}
    on_press: root.dismiss()
```