Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

Ítalo Araújo Ferreira de Lucena

Trabalho de Conclusão de Curso

Modelagem e Simulação de uma Coluna de Destilação

Campina Grande
Setembro 2013

Ítalo Araújo Ferreira de Lucena

Modelagem e Simulação de uma Coluna de Destilação

Trabalho de Conclusão de Curso submetido à Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador:
Péricles Rezende Barros

Campina Grande
Setembro 2013

Ítalo Araújo Ferreira de Lucena

Modelagem e Simulação de uma Coluna de Destilação

Trabalho de Conclusão de Curso submetido à Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Professor Péricles Rezende Barros
Universidade Federal de Campina Grande
Professor Orientador

Professor Convidado
Universidade Federal de Campina Grande

Campina Grande
Setembro 2013

Agradecimentos

Agradeço muito a minha família, em especial aos meus pais (Flauberto e Mônica) que sempre me apoiaram em todos os momentos e nunca me deixaram desistir. Ao meu irmão Guilherme que me aturou nesses anos.

Um agradecimento em especial para a minha noiva Kamylla, que não me deixou desistir do curso quando achei que não permaneceria nele.

Agradecer também aos meus professores que passaram tanto conhecimento nesses anos. Em especial ao professor Péricles Rezende, que me aceitou como bolsista desde o segundo período e me permitiu trabalhar no Laboratório de Instrumentação Eletrônica e Controle, e assim adquirir uma experiência inestimável.

"*Bom mesmo é ir à luta com determinação, abraçar a vida com paixão, perder com classe e vencer com ousadia...pois o triunfo pertence a quem se atreve.*"

Charles Chaplin

Resumo

Este trabalho de conclusão de curso tem por objetivo a completa descrição, modelagem e simulação de uma coluna de destilação binária ideal. Será estudado o modelo desse sistema e assim analisar as equações diferenciais que regem o comportamento desse processo. Após o estudo será criado um simulador no ambiente Visual Studio 2012 e realizar ensaios na simulação e compará-la com os resultados mostrados na bibliografia.

Palavras-chave: modelagem, simulação, coluna de destilação, OPC.

Abstract

This project's objective is to do a complete description, modeling and simulation of a binary distillation collum. It will be studied the the system model so it is possible to analyze the differential equations that controls the process behavior. The next step will be to create a simulator using Visual Studio 2012 and do some experiments to compare the results to the ones shown in the research material.

Key-words: modeling, simulation, distillation collum, OPC.

Sumário

1 Introdução **5**

2 Modelo do Sistema **6**

2.1 Destilação 6

2.2 Descrição Concitual da Destilação 7

2.3 Balanço Dinâmico de Matéria 9

2.3.1 Todos os Níveis, exceto Alimentação, Condensador e Refervedor 9

2.3.2 Nível de Alimentação 9

2.3.3 Condensador 10

2.3.4 Refervedor 10

2.3.5 Resumo das Equações 10

3 Integração Numérica **12**

3.1 Integração de Euler 12

3.1.1 Método Explícito 12

3.1.2 Método Implícito 13

3.2 Integração de Runge-Kutta 13

3.2.1 Runge-Kutta de Segunda Ordem 13

3.2.2 Runge-Kutta de Quarta Ordem 14

3.3 Método Escolhido 15

4 Tecnologias Utilizadas **16**

4.1 OLE for Process Control (OPC) 16

4.1.1 Definição 16

4.1.2 OPC-DA Toolkit Advosol 17

4.2 Microsoft Visual Studio 18

5 Simulador da Coluna de Destilação 20

5.1 Software de Simulação 20

5.1.1 Configuração OPC 20

5.1.2 Configuração da Simulação 21

5.1.3 Aba de Monitoramento 23

5.2 Servidor OPC 25

5.3 Cliente OPC 25

6 Experimentos Realizados 26

6.1 Cálculo do Ponto de Operação 26

6.2 Mudança no Refluxo 27

6.3 +1% do Ponto de Operação 27

6.4 -1% do Ponto de Operação 27

6.5 Dinâmica Usando o MatLab 28

6.6 Comparação 28

7 Considerações Finais 29

Appendices 31

A Códigos Fonte 32

A.1 Servidor OPC 32

A.2 Simulador 37

A.3 Classe Euler 53

Lista de Figuras

2.1 Sistema fechado com líquido e vapor em equilíbrio 6

2.2 Diagrama de equilíbrio 7

2.3 Diagrama esquemático para uma coluna de destilação 8

2.4 Balança de matéria para um estágio típico e para o estágio de alimentação 8

4.1 Configuração do Servidor OPC 17

4.2 Cliente de Teste Criado pelo OPCDA-Toolkit 18

5.1 Aba de configuração OPC 21

5.2 Aba de simulação 23

5.3 Aba de monitoramento 24

5.4 Cliente OPC 25

6.1 Cliente OPC 27

6.2 Cliente OPC 27

6.3 Dinâmcas dos níveis superior e inferior 28

Lista de Tabelas

5.1 Valores Iniciais 23

6.1 Comparações das frações molares iniciais 26

1 Introdução

A modelagem é o processo de formulação dos efeitos dinâmicos do sistema que serão considerados através de equações matemáticas. Essa é uma maneira de se adquirir um maior conhecimento a respeito de um processo real, e poder analisar conhecer seus estados e respostas a entradas e a pertubações sem que se necessite do processo real.

A simulação é uma alternativa muito satisfatória de se pensar como uma solução de problemas que, de antemão, é muito cara ou mesmo impossível através de experimentos ou mesmo para problemas que são muito complexos para tratamento analítico. Como benefício se tem uma forma de testar idéias e condições de funcionamento do sistema que poderiam levar a planta real a uma condição de risco.

O objetivo desse trabalho é modelar e simular uma coluna de destilação binária ideal. Tal coluna possuirá 41 bandejas e terá saídas nos níveis superior e no inferior. Além disso, será feito com que as variáveis de entrada e de saída do sistema estejam ligadas a um servidor OPC para que assim se possa conectar equipamentos de controle, como por exemplo um CLP e aplicar estratégias de controle como se fosse um processo real.

A simulação será feita no ambiente de programação Visual Studio 2012 com a linguagem C#. O motivo por trás desssa escolha foi a quantidade de ferramentas encontradas que permitem uma simulação flexível além de uma interface amigável para o usuário.

2 | Modelo do Sistema

2.1 Destilação

Destilação é uma técnica de separação de líquidos que contém dois ou mais componentes e é uma das operações mais importantes em processos químicos relacionados a manufatura. O design e controle da destilação é importante para produzir produtos que contenham um determinado grau de pureza, seja para venda ou para uso em outros processos químicos.

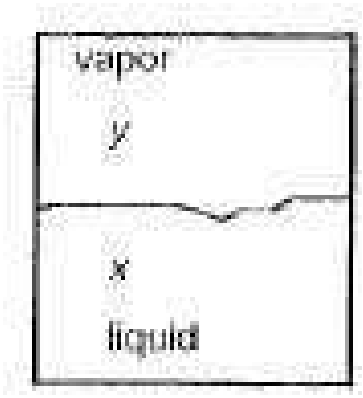


Figura 2.1: Sistema fechado com líquido e vapor em equilíbrio

Destilação é baseada na separação de componentes de uma mistura líquida devido a diferença no ponto de ebulição dos mesmos. Nesse trabalho será discutida a separação de líquidos contendo apenas dois componentes (mistura binária). O componente que ferve a uma temperatura mais baixa será referido como o componente leve e o que ferve a uma temperatura mais alta como sendo o componente pesado.

Uma mistura saturada que possui dois ou mais componentes a uma dada concentração em equilíbrio com a fase que é vapor que possui uma concentração mais alta do componente leve do que da fase líquida. Sendo x a fração molar do componentena fase líquida e y a fração molar do componente leve em vapor.

A figura 2 é um exemplo de um diagrama de equilíbrio que representa a relação entre as frações

molares das fases líquidas e gasosa.

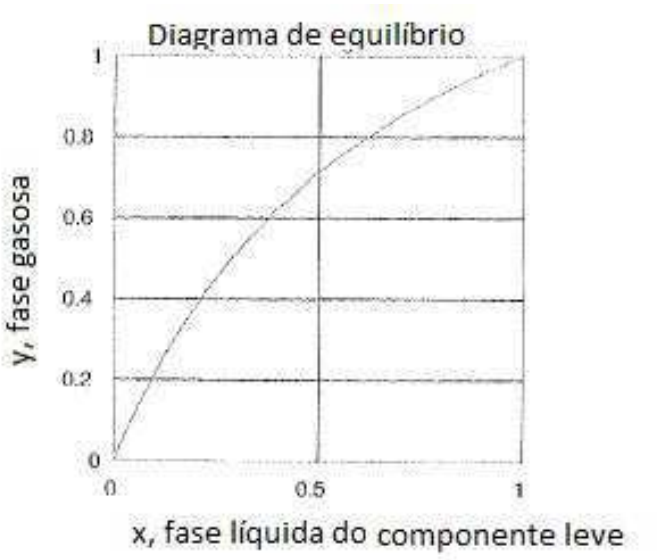


Figura 2.2: Diagrama de equilíbrio

Para misturas ideais, o modelo comum para essa relação é dada por:

$$y = \frac{\alpha x}{1 + (\alpha - 1)x} \tag{2.1}$$

Onde α é conhecido como volatilidade relativa.

2.2 Descrição Concitual da Destilação

Para o processo binário a alimentação normalmente está localizada no meio da coluna. Gás flui do nível mais baixo para o mais alto enquanto que o líquido flui dos níveis mais altos para os mais baixos. A vapor da bandeja mais alta é condensado e uma porção do líquido retorna como refluxo. O resto é retirado como o produto da parte superior, esse fluxo contém uma alta concentração do componente leve. Uma porção do líquido no nível inferior da coluna sai como o produto inferior (contém uma alta concentração do componente pesado), enquanto que o resto é vaporizado no refervedor e volta para a coluna.

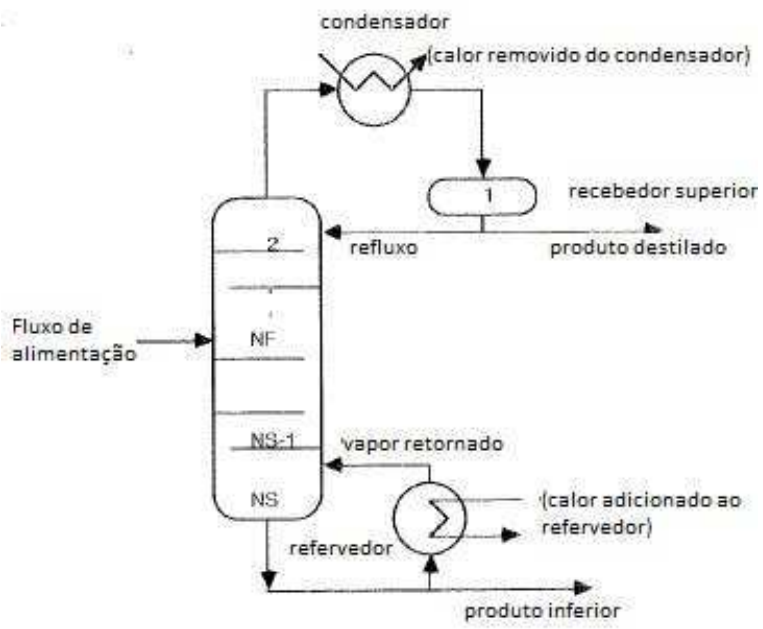


Figura 2.3: Diagrama esquemático para uma coluna de destilação

O líquido de uma bandeja flui para a próxima bandeja que se situa abaixo. Quando o líquido se move pela bandeja, ele entra em contato com o vapor da bandeja de baixo.

Normalmente, quando o vapor entra em contato com o líquido uma mistura turbulenta ocorre. Assumindo que a mistura é perfeita, pode-se modelar a bandeja como um conjunto aglomerado. Deve-se notar que o vapor do nível i é modelado como uma única entrada com fluxo V_i e fração molar y_i. O líquido deixando o nível i é modelado como um único fluxo com vazão L_i e fração molar líquida x_i.

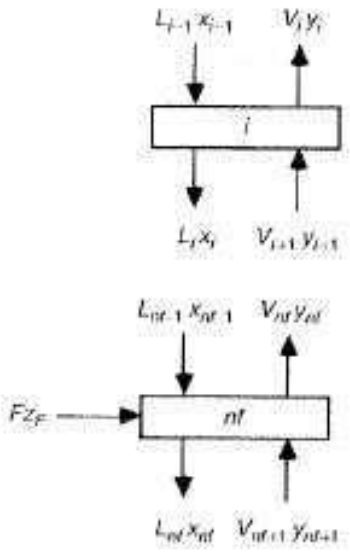


Figura 2.4: Balança de matéria para um estágio típico e para o estágio de alimentação

2.3 Balanço Dinâmico de Matéria

2.3.1 Todos os Níveis, exceto Alimentação, Condensador e Refervedor

O balanço de componentes da fase líquida de um nível comum é:

$$\frac{dM_i x_i}{dt} = L_{i-1}x_{i-1} + V_{i+1}y_{i+1} - L_i x_i - V_i y_i \tag{2.2}$$

Onde M_i é a massa molar líquida no nível i.

Para esse estudo será assumido o fluxo equimolar. Para o fluxo de vapor, o fluxo de um nível é igual ao de nível inferior.

$$V_i = V_{i+1} \tag{2.3}$$

E para o líquido que deixa um nível é igual ao que chega de um nível superior.

$$L_i = L_{i-1} \tag{2.4}$$

2.3.2 Nível de Alimentação

Sendo q_f a qualidade do fluxo de alimentação. Se este for igual a 1 é um líquido saturado, enquanto que se for igual a 0 é um vapor saturado. O vapor que sai do nível de alimentação é (sendo Nf p nível de alimentação):

$$V_{NF} = V_{NF+1} + F(1 - q_f) \tag{2.5}$$

De maneira similar, o fluxo do líquido deixando esse nível é:

$$L_{NF} = L_{NF-1} + Fq_f \tag{2.6}$$

Onde F é o fluxo de alimentação.

2.3.3 Condensador

Uma condensação total remove a energia do vapor e provê um líquido saturado. Assumindo uma massa molar constante, o fluxo de líquido que vem do destilado mais o refluxo é igual ao fluxo de vapor da bandeja de cima.

$$L_D + D = V_2 \tag{2.7}$$

Onde L_D e D representam o refluxo e fluxo molar do destilado, respectivamente.

2.3.4 Refervedor

O balanço total de matéria no refervedor é dado por:

$$B = L_{NS-1} - V_{reboiler} \tag{2.8}$$

Onde NS é um número de níveis presente na coluna e $V_{reboiler}$ é o fluxo do refervedor e B é o fluxo do produto inferior.

2.3.5 Resumo das Equações

Na parte acima do nível de alimentação o fluxo de líquido é dado por:

$$L_R = L_D \tag{2.9}$$

E o de vapor

$$V_R = V_S + F(1 - q_f) \tag{2.10}$$

Na região abaixo do nível de alimentação:

$$L_S = L_R + Fq_f \tag{2.11}$$

E de vapor

$$V_S = V_{reboiler} \tag{2.12}$$

Será assumido uma massa molar constante, ou seja,

$$\frac{dM}{dt} = 0$$

A componente do nível mais alto é dada por:

$$\frac{dx_1}{dt} = \frac{1}{M_D}[V_R(y_2 - x_1)] \tag{2.13}$$

Para os níveis abaixo da alimentação:

$$\frac{dx_i}{dt} = \frac{1}{M_T}[L_R x_{i-1} + V_R y_{i+1} - L_R x_i - V_R y_i] \tag{2.14}$$

O balanço para o nível de alimentação:

$$\frac{dx_{NF}}{dt} = \frac{1}{M_T}[L_R x_{NF-1} + V_S y_{NF+1} - L_S x_N F - V_S y_N F + F z_F] \tag{2.15}$$

A componente para os níveis superiores é dada por:

$$\frac{dx_i}{dt} = \frac{1}{M_T}[L_S x_{i-1} + V_S y_{i+1} - L_S x_i - V_S y_i] \tag{2.16}$$

E para o refervedor:

$$\frac{dx_{NS}}{dt} = \frac{1}{M_B}[L_S x_{NS-1} - B x_N S - V_S y_N S] \tag{2.17}$$

3 Integração Numérica

O objetivo deste capítulo é de mostrar as técnicas que de integração numérica que foram estudadas antes de se escolher qual seria utilizada.

3.1 Integração de Euler

3.1.1 Método Explícito

Usando-se uma aproximação para e derivada em relação ao tempo tem-se que:

$$\frac{dx}{dt} = \frac{x(k+1) - x(k)}{t(k+1) - t(k)} \tag{3.1}$$

Onde k representa o k-ésimo passo discreto de tempo da integração. Agora, assumindo que uma função f(x) seja avaliada em x(k), tem-se que:

$$\frac{x(k+1) - x(k)}{t(k+1) - t(k)} = f(x(k)) \tag{3.2}$$

Normalmente seria utilizado um incremento constante de tempo, ou seja,

$$t(k+1) - t(k) = \Delta t$$

Onde Δt é o passo de integração. Assim pode-se reescrever a equação como:

$$\frac{x(k+1) - x(k)}{\Delta t} = f(x(k)) \tag{3.3}$$

Resolvendo para x(k+1)

$$x(k+1) = x(k) + \Delta t f(x(k)) \tag{3.4}$$

Pose-de analisar a equação anterior como sendo uma predição de x no tempo k+1 baseado no valor de x e de k e a inclinação da função no tempo k.

A equação mostrada é a expressão para o método Explícito de Euler para uma única variável.

3.1.2 Método Implícito

Esse método usa uma aproximação diferente para uma derivada. A função é avaliada no tempo k+1 ao invés do tempo k.

$$\frac{x(k+1) - x(k)}{\Delta t} = f(x(k+1)) \tag{3.5}$$

O qual pode ser reescrito como

$$x(k+1) = x(k) + \Delta t f(x(k+1)) \tag{3.6}$$

A equação anterior é conhecida como método Implícito de Euler. Ela é chamada assim pois é necessário que se conheça x(k+1) para que se resolva a equação. Geralmente é necessário outro método de resolução de equações não-lineares, como o método de Newton.

3.2 Integração de Runge-Kutta

Essa técnica é uma extensão do método de Euler. Nesse método, a derivada no tempo k era usada para encontrar o valor da função no tempo k+1. Os métodos de Runge-Kutta usa o método de Euler para determinar a função em intervalos intermediários, então utiliza diferentes inclinações para encontrar o resultado baseado no tempo inicial.

3.2.1 Runge-Kutta de Segunda Ordem

O primeiro método usa Euler para encontrar um primeiro resultado no tempo $\Delta/2$. A derivada é encontrada nesse ponto médio, em seguida o método é usado novamente para encontrar o valor de x no tempo desejado.

Sendo m_1 a inclinação no ponto inicial e m_2 a inclinação no ponto médio:

$$m_1 = f(x()) \tag{3.7}$$

$$m_2 = f(x(k) + \frac{\Delta t}{2} m_1) \tag{3.8}$$

$$m_2 = f(x(k) + \frac{\Delta t}{2} m_1) \tag{3.9}$$

O que nos leva a

$$x(k+1) = x(k) + \Delta t f(f(x(k) + \frac{\Delta t}{2} f(x(k))) \tag{3.10}$$

Deve-se notar que esse método é preciso até a ordem de Δt^2 enquanto que o método explícito de Euler é até a ordem de Δt

3.2.2 Runge-Kutta de Quarta Ordem

A idéia por trás desse método é usar a inclinação inicial m_1 para gerar os primeiros "chutes"do valor 'de x no ponto médio. Essa primeira tentativa é usada para encontrar a inclinação no ponto médio, que é corrigida por um m_3 usando m_2. Um última inclinação é então encontrada para essa integração. O algoritmo é o que se segue:

$$m_1 = f(x()) \tag{3.11}$$

$$m_2 = f(x(k) + \frac{1}{2} m_1 \Delta t) \tag{3.12}$$

$$m_3 = f(x(k) + \frac{1}{2} m_2 \Delta t) \tag{3.13}$$

$$m_4 = f(x(k) + m_3 \Delta t) \tag{3.14}$$

$$x(k+1) = x(k) + f(f(x(k) + [\frac{m_1}{6} + \frac{m_2}{3} + \frac{m_3}{3} + \frac{m_4}{6}]\Delta t \tag{3.15}$$

3.3 Método Escolhido

Após a análise de todos os métodos e dos seus benefícios, o escolhido inicialmente para o projeto do simulador foi o método **Explícito de Euler**, uma vez que os intervalos de tempo que serão analisados são bem pequenos o que resultará numa precisão muito próxima dos métodos mais sofisticados.

4 Tecnologias Utilizadas

4.1 OLE for Process Control (OPC)

4.1.1 Definição

OPC são especificações baseadas nas tecnologias OLE, COM e DCOM, que foram desenvolvidas pela Microsoft para sistemas que operam com arquitetura baseada em Microsoft Windows. Essa especificação define um padrão de objetos, interfaces e métodos para o uso em controle de processos e em aplicações de automação em geral para facilitar a interoperabilidade.

OPC foi desenvolvido para conectar softwares baseados em Windows e hardwares de controle. O padrão define métodos consistentes que acessam o dado de dispositivos não importanto o seu tipo, fabricante ou versão. Assim permitindo que o desenvolvimento de um software não precise se preocupar com o hardware a que ele vai se conectar. Assim o programa pode ser escrito apenas uma vez e depois ser reutilizado em diversas aplicações.

Uma vez que um servidor OPC é escrito para um dispositivo em particular, ele pode ser conectado por quaquer aplicação que seja apaz de se conectar com este dispositivo como um cliente OPC. Estes servidores usam a tecnologia OLE da Microsoft que é baseada COM (tecnologia que permite a comunicação entre softwares variados) para se conectar com os clientes. A tecnologia COM premite um padrão de troca de informação em tempo real entre as aplicações em software e o hardware que controla o processo a ser monitorado.

Nesse projeto foi utilizado o padrão OPC DA, ou Data-Access. Ele é usado para mover dados de PLC‘s, DCS‘s e outros dispositivos de controle para clientes conectados ao servidor em tempo real, sem se preocupar com a armazenação deste dado para uso posterior. Tal funcionalidade deve ser realizada pelo usuário que queira que a mesma seja implementada.

4.1.2 OPC-DA Toolkit Advosol

C# e Visual Basic .Net são as ferramentas preferidas de muitos desenvolvedores de software. Como COM e DCOM são ferramentas muito complexas, o toolkit da Advosol permite que sejam criados servidores OPC como se os mesmos fossem aplicações .Net quaisquer, desde que a mesma seja baseada nas bibliotecas já criadas pelo próprio kit de desenvolvimento. O servidor genérico lida com a interface DCOM do cliente e com a conversão para .Net. Usando essa ferramenta é possível fazer com que os servidores OPC possam interagir com dispositivos externos usando comunicação TCP/IP, serial ou até mesmo usando um banco de dados.

Uma das desvantagens dessa ferramenta é a necessidade de aumentar o acesso remoto a estes servidores, uma vez que o mesmo só permite acessos praticamente locais, de ambientes que estejam muito próximos do servidor.

A Advosol também promoveu um kit para desenvolvimento de clientes OPC, também baseados em C# e Visual Basic. Ele dispõe de classes, controles e ferramentas para o desenvolvimento de clientes OPC DA.

O programa de configuração do servidor OPC e o programa de um cliente OPC estão representados abaixo .

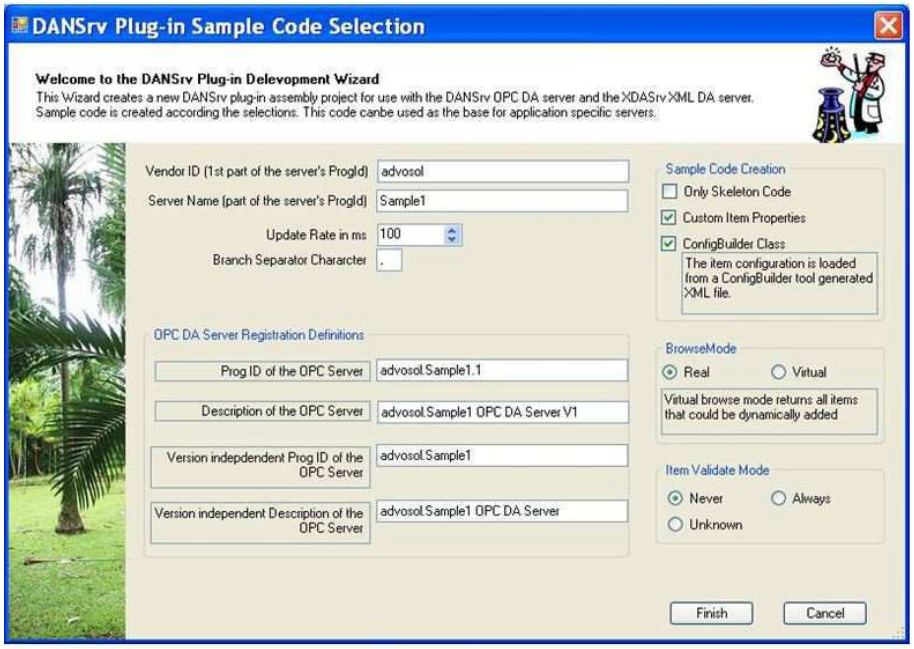


Figura 4.1: Configuração do Servidor OPC

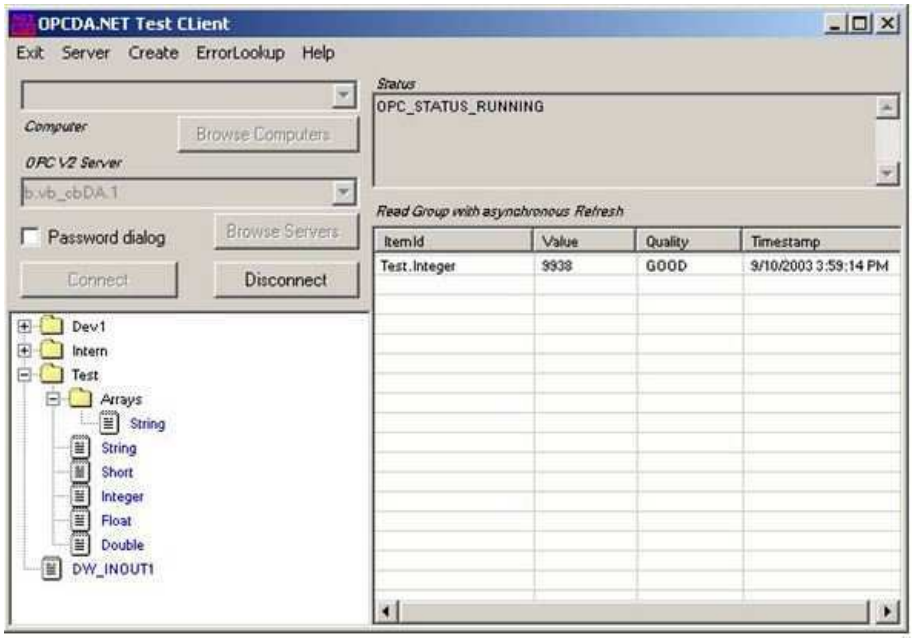


Figura 4.2: Cliente de Teste Criado pelo OPCDA-Toolkit

4.2 Microsoft Visual Studio

Microsoft Visual Studio é um ambiente de desenvolvimento integrado (IDE) da Microsoft. Ele é usado para desenvolver console applications com interface gráfica para o usuário, juntamente com o Windows Forms ou aplicações WPF, web sites, aplicações web e serviços web em código nativo juntamente com o código gerenciado para todas as plataformas suportadas pelo Microsoft Windows, Windows Mobile, Windows CE, . NET Framework,. NET Compact Framework e Microsoft Silverlight.

Visual Studio inclui um editor de código de suporte IntelliSense, bem como refatoração de código. O depurador integrado funciona tanto como um depurador de nível de fonte e um depurador de nível de máquina. Outros ferramentas incluem um designer de formulários para a construção de aplicações GUI, web designer, designer de classe, e designer de esquema de banco de dados. Ele aceita plug-ins que melhoram a funcionalidade em quase todos os níveis, incluindo a adição de suporte para sistemas de controle de origem (como Subversion e Visual SourceSafe) e adicionando novos conjuntos de ferramentas, como editores e designers visuais de idiomas ou de conjuntos de ferramentas específicas de domínio para outros aspectos da ciclo de vida de desenvolvimento de software (como o cliente Team Foundation Server: Team Explorer). Visual Studio oferece suporte a diferentes linguagens de programação, por meio de serviços linguísticos, que permitem que o editor de código e depurador para apoiar (em diferentes graus) quase qualquer linguagem de programação, desde um serviço específico do idioma existe. Entre as

linguagens incluem-se C / C + +, VB.NET (via Visual Basic. NET), C # (via Visual C #) e F #.

5 | Simulador da Coluna de Destilação

Para analisar a dinâmica do sistema em tempo real foi necessária a criação de um simulador. O seu objetivo é mostrar o desenvolvimento do sistema utilizando as equações não lineares que foram vistas no capítulo 2.

Utilizando a linguagem de programação C# e o ambiente de programação Visual Studio 2012, foi desenvolvido um software que a partir de diversas integrações numéricas fosse capaz de obter a solução das equações estudadas no modelo, e fornecer o valor das variáveis durante o regime transitório para estudo do comportamento do sistema bem como o monitoramento das variáveis de interesse (frações molares dos estados líquidos e gasosos em cada bandeja).

Deve-se lembrar que o sistema usa um servidor OPC como fonte de suas entradas, e que todos os dados de estados também são escritos nesse mesmo servidor. Mais adiante será mostrado como o servidor foi configurado.

5.1 Software de Simulação

5.1.1 Configuração OPC

O programa possui três abas de visualização, sendo a primeira a de configuração de itens OPC. Ela permite que o usuário relacione as variáveis de entrada e saída do sistema com itens OPC de um servidor que seja escolhido por ele em qualquer máquina que se encontre na rede local onde o simulador esteja sendo executado. Essa aba é mostrada na figura 5.1.

Os componentes presentes nessa aba são:

- Botão Browse Computadores: localiza o nome de todos os computadores que estão na mesma rede que a máquina onde o simulador funciona.
- Botão Browse OPC Servers: encontra os servidores OPC presentes no computador selecionado

previamente.

- Botão Connect: se conecta com o servidor escolhido e realiza automaticamente o browse dos itens OPC do servidor.
- Botão Disconnect: se desconecta do servidor OPC atualmente conectado.
- ComboBox Variável do sitema: mostra todas as variáveis da coluna de destilação que podem ser associadas a itens OPC.
- Botão Associar: associa o item OPC selecionado com a variável do sistema escolhida.

Como não existe nenhuma restrição quanto aos itens serem do mesmo servidor ou mais de uma variável estar associada ao mesmo item OPC, o usuário deve ser cuidadoso em relação a isso para que não ocorram problemas de execução do programa.

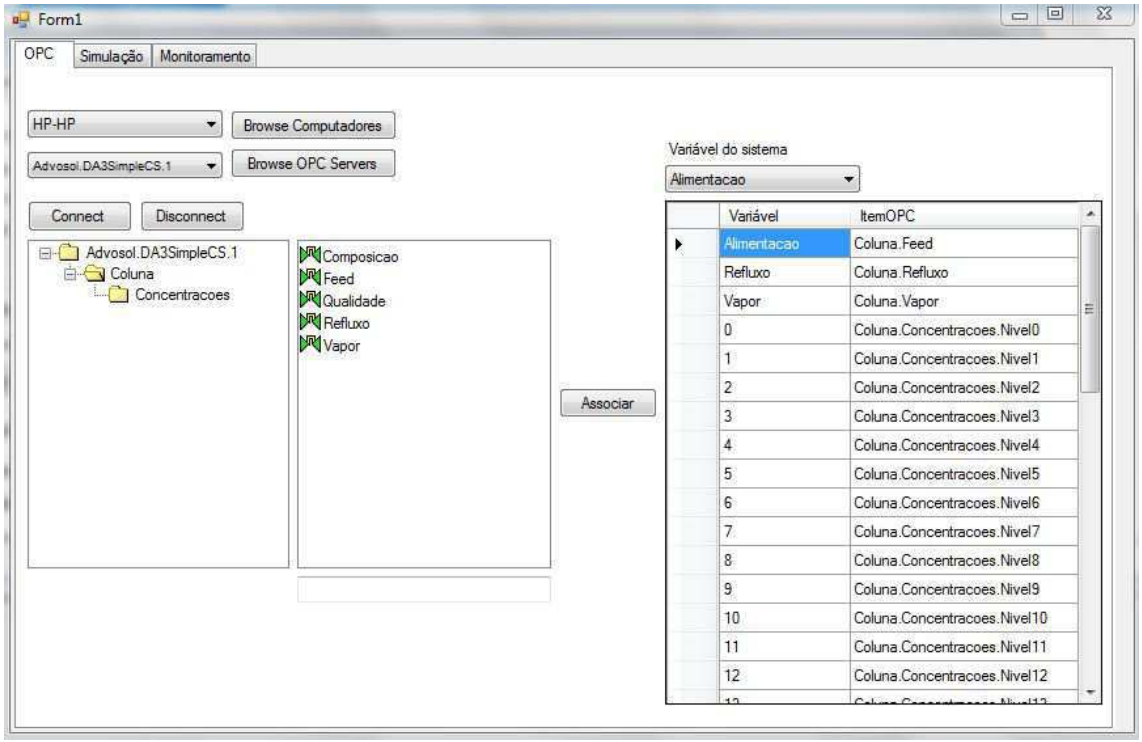


Figura 5.1: Aba de configuração OPC

5.1.2 Configuração da Simulação

Na aba simulação o usuário pode escolher os parâmetros do sistema, com exceção das entradas (fluxo de alimentação, refluxo do destilado e fluxo do refervedor) uma vez que elas são informadas através do servidor OPC que foi selecionado anteriormente.

Ao usuário é permitido controlar os parâmetros que geralmente não podem ser mensurados tão facilmente. Essas variáveis são:

- qualidade da alimentação;
- volatilidade relativa;
- composição da alimentação;
- massa molar do nível superior;
- massa molar dos níveis intermediários;
- massa molar do nível inferior.

Os componentes presentes nessa aba são:

- Botão Confirmar parâmetros: usa os parâmetros selecionados pelo usuário para a simulação.
- Botão Calcular ponto de operação: como a constante de tempo do sistema é de cerca de 3 horas, o ponto de operação é calculado de maneira recursiva para que não seja necessária a espera pela constante de tempo.
- Botão Iniciar simulação: após calcular o ponto de operação, inicia a resolução das equações diferencias que mostram o comportamento do sistema de maneira que os dados são mostrados no gráfico (apenas frações molares dos níveis de alimentação e dos níveis superior e inferior).
- Botão Mudar fator de aceleração: como foi mencionado anteriormente a constante de tempo é muito lenta e qualquer alteração leva muito tempo para alcançar o regime permanente, pensando nisso essa opção foi adicionada para que seja possível aumentar a velocidade de simulação pelo fator que o usuário desejar.

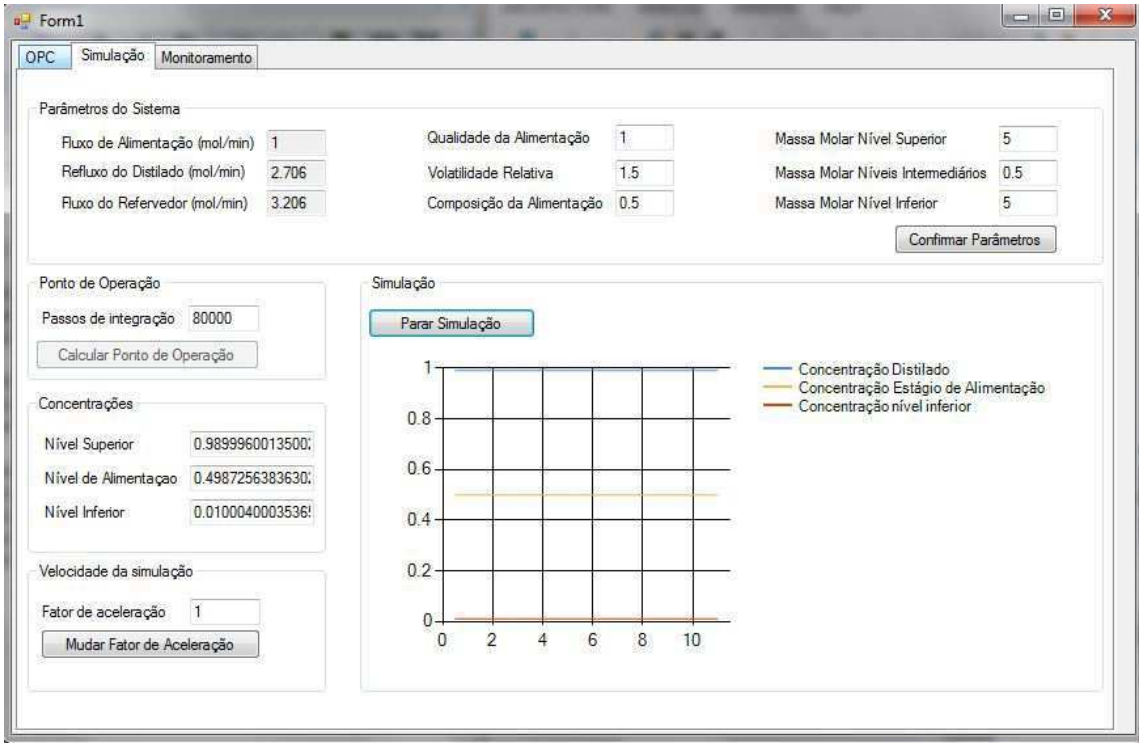


Figura 5.2: Aba de simulação

Usando os parâmetros inicias indicados por [1].

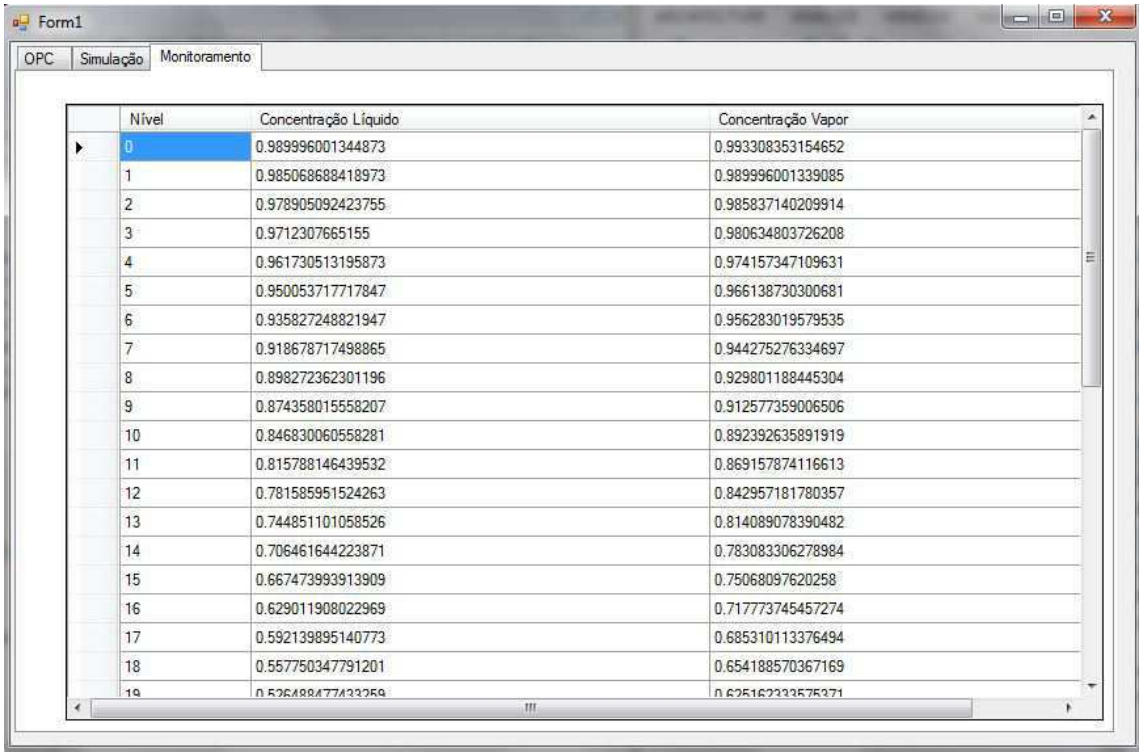
Tabela 5.1: Valores Iniciais

Fluxo de Alimentação	1 mol/min
Refluxo do Destilado	2,706 mol/min
Fluxo do Refervedor	3.206 mol/min
Qualidade da alimentaçãao	1
Volatilidade relativa	1.5
Composição da alimentação	0.5
Massa molar do nível superior	5
Massa molar dos níveis intermediários	0.5
Massa molar do nível inferior	5

5.1.3 Aba de Monitoramento

A função dessa aba é adquirir em tempo real as frações molares tanto do fase líquida quanto da fase gasosa de todas as bandejas e mostrá-las para o usuário, assim é possível realizar um acompanhamento

mais detalhado do que se ocorre no processo, algo que o gráfico nem sempre pode fazer uma vez que colocar 41 curvas tornaria a visualização pouco eficiente.



Nível	Concentração Líquido	Concentração Vapor
0	0.989996001344873	0.993308353154652
1	0.985068688418973	0.989996001339085
2	0.978905092423755	0.985837140209914
3	0.9712307665155	0.980634803726208
4	0.961730513195873	0.974157347109631
5	0.950053717717847	0.966138730300681
6	0.935827248821947	0.956283019579535
7	0.918678717498865	0.944275276334697
8	0.898272362301196	0.929801188445304
9	0.874358015558207	0.912577359006506
10	0.846830060558281	0.892392635891919
11	0.815788146439532	0.869157874116613
12	0.781585951524263	0.842957181780357
13	0.744851101058526	0.814089078390482
14	0.706461644223871	0.783083306278984
15	0.667473993913909	0.75068097620258
16	0.629011908022969	0.717773745457274
17	0.592139895140773	0.685310113376494
18	0.557750347791201	0.654188570367169
19	0.526488477433259	0.625162333575371

Figura 5.3: Aba de monitoramento

5.2 Servidor OPC

Utilizando o toolkit da Advosol foi desenvolvido um servidor OPC que poderia atender as necessidades do simulador. Neste caso ele possui um item para cada entrada do sistema e a fração molar da fase líquida de cada uma das bandejas. A fase gasosa não foi adicionada pois é possível calculá-las utilizando a fase líquida.

5.3 Cliente OPC

Como um programa de teste da Advosol é fornecido juntamente com com o toolkit ele foi escolhido como maneira de modificar as entradas do simulador.

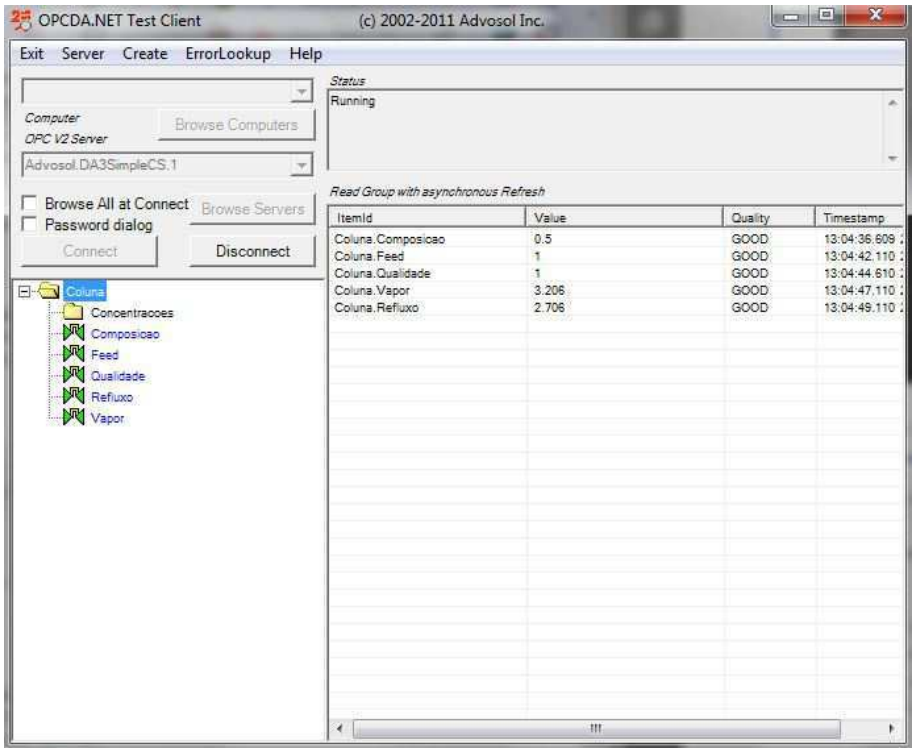


Figura 5.4: Cliente OPC

6 Experimentos Realizados

Como não foi possível comparar os resultados obtidos dos experimentos realizados com um processo real, visto a dificuldade de se encontrar algo assim em tempo limitado, as comparações realizadas serão feitas entre o que foi obtido no simulador e os resultados obtidos na referência [1]. Isso servirá de guia para saber se a modelagem foi realizada de maneira correta e ver as diferenças nos resultados entre a maneira de simular do software e a do MatLab (usada no referência).

6.1 Cálculo do Ponto de Operação

A maneira como o MatLab resolve essa equação é resolvendo as equações diferenciais de maneira que elas sejam iguais a zero, ou seja, o sistema alcançou o regime permanente. Como isso envolve a resolução de equações não-lineares e isso é muito difícil de ser feito em C# a menos que se tenha uma biblioteca que possa resolver esse tipo de sistema. A proposta do simulador foi de, dados as condições iniciais, usar a função de integração inúmeras vezes de maneira que a variação final seja muito pequena.

Como a quantidade de níveis é muito grande, serão comparados os resultados dos níveis de alimentação, superior e inferior.

Tabela 6.1: Comparações das frações molares iniciais

Nível	MatLab	Simulador	Diferença
Superior	0,99	0,989	0,1%
Alimentação	0,499	0,498	0,2%
Inferior	0,01	0,01	0%

Como os níveis são dependentes entre si, pode-se afirmar que as demais bandejas estão sendo representadas com um nível de precisão semelhante aos resultados mostrados anteriormente.

6.2 Mudança no Refluxo

6.3 +1% do Ponto de Operação

Realizando uma mudança do ponto de operação do refluxo obteve-se os seguintes resultados nas frações molares.

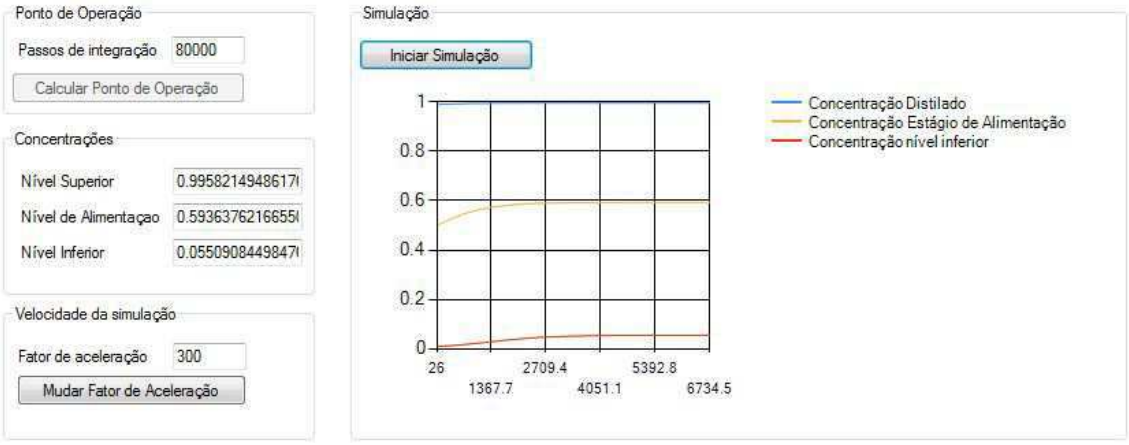


Figura 6.1: Cliente OPC

Foi utilizado um fator de aceleração de 300 para obter o resultado mais rapidamente.

6.4 -1% do Ponto de Operação

Realizando uma mudança do ponde de operação do refluxo obteve-se os seguintes resultados nas concentrações.

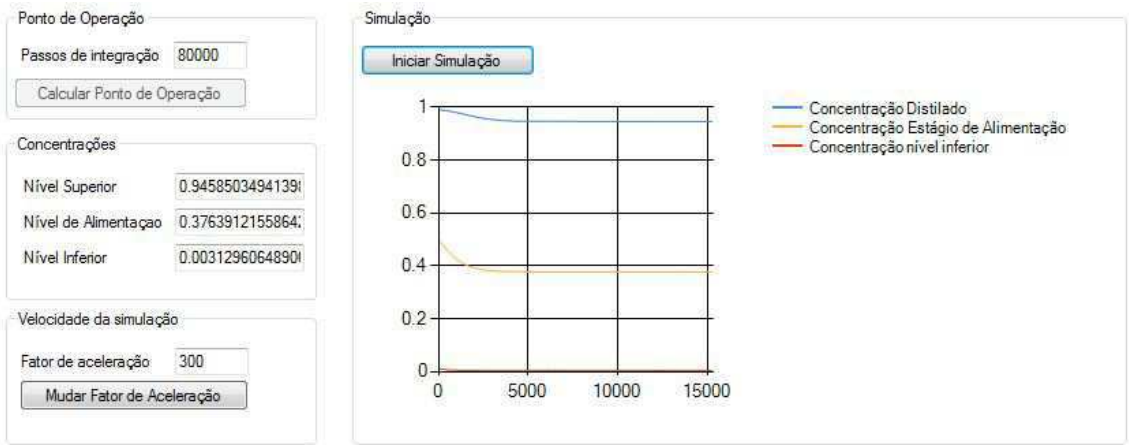


Figura 6.2: Cliente OPC

O mesmo fator de aceleração foi utilizado.

6.5 Dinâmica Usando o MatLab

O resultado mostrado pela fonte [1] é a seguinte:

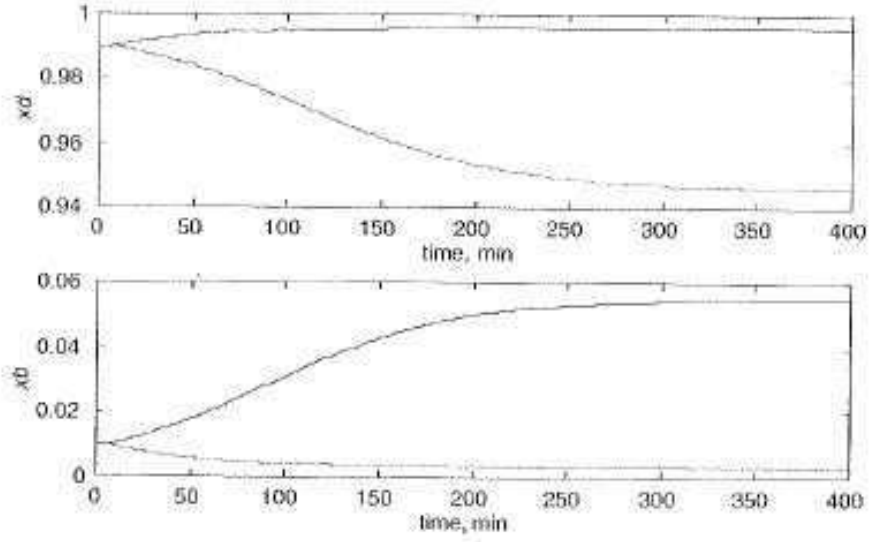


Figura 6.3: Dinâmcas dos níveis superior e inferior

Onde as curvas crescentes mostram o alteração para +1%, enquanto que as decrescentes mostram a de -1%.

6.6 Comparação

Analisando os valores finais, pode-se dizer que o simulador se aproximou bastante da dinâmica do MatLab. Não foi mostrado toda a dinâmica no simulador uma vez que a quantidade de pontos que seria necessária não mostraria toda a dinâmica, uma vez que o programa remove todos os pontos muito antigos para não sobrecarregar o gráfico e fazer com que a execução do programa se torne muito lenta.

Porém analisando os gráficos da fonte [1] nota-se que as variações mais drásticas ocorrem até os 250 minutos após esse intervalo as mudanças são mais desprezíveis.

7 | Considerações Finais

A modelagem matemática de sistemas e a simulação utilizando as mesmas são de extrema importância no momento de estudar um processo, uma vez que é possível aprender sobre sua dinâmica, resposta a pertubações e comportamento em regime permanente. Além disso, a possibilidade de realizar experimentos de maneira simples e no conforto de uma sala favorece muito o uso dessa ferramenta, uma vez que alguns ensaios podem levar a planta real a condições críticas.

Poder conectar um simulador a uma rede usando OPC faz com que possa-se sintonizar o controlador para o processo usando um CLP e depois usá-lo no processo real.

Outra vantagem muito grande é o fato de que sistemas complexos e caros que muitas vezes não são possíveis de serem adquiridos, podem ficar ao alcance de várias pessoas através de um clique.

Analisando o sistema desenvolvido, nota-se que por ele apresentar resultados semelhantes aqueles da referência [1] o simulador apresenta resultados satisfatórios, abrindo portas para futuros estudos de identificação e controle da planta virtual.

Referências Bibliográficas

[1] BEQUETTE, Wayne. *Process Dynamis Modeling, Analysis and Simulation.* New Jersey 1998 Prentice Hall.

[2] OPC DA .NET Server Toolkit. Disponível em: <*https://www.advosol.us/pc-5-4-dansrv-net-server-toolkit.aspx*>. Acesso em: 12 set. 2013, 13:42.

[3] About OPC - What is OPC?.
Disponível em: <*http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC*>.
Acesso em: 12 set. 2013, 14:00.

[4] Fractional distillation.
Disponível em: <*http://en.wikipedia.org/wiki/Fractional_distillation*>.
Acesso em: 14 set. 2013, 16:42.

Appendices

A | Códigos Fonte

A.1 Servidor OPC

```
using System; using System.Threading; using
System.Runtime.InteropServices;

namespace NSPlugin {

   //============================================================================
   // OPC Server Configuration and IO Handling
   //============================================================================
   public class AppPlugin : GenericServer
   {

   ew public int CreateServerItems( string cmdParams )
      {
         // create all items from the definition table
         for( int i=0 ; i<Items.Length ; ++ i )
         {
            AddItem( i, Items[i].ID, (int)OPCAccess.READWRITEABLE, Items[i].Value,
               (short)OPCQuality.GOOD, DateTime.Now, 500 );
         }

         // create a thread for simulating signal changes
         // in real application this thread reads from the device
```

```
      myThread = new Thread( new ThreadStart( RefreshThread ) ) ;
      myThread.Name = "Item Simulation" ;
      myThread.Priority = ThreadPriority.AboveNormal;
      myThread.Start();

      return HRESULTS.S_OK;
   }

   new public void ShutdownSignal()
   {
      ////////////////// TO-DO /////////////////
      // close the device communication

      // terminate the simulation thread
      StopThread = new ManualResetEvent( false );
      StopThread.WaitOne( 5000, true );
      StopThread.Close();
      StopThread = null;
   }

   new public int WriteItems(int instanceHandle, DeviceItemValue[] values, out int[] errors)
   {
      errors = new int[ values.Length ];       // result array
      for( int i=0 ; i<values.Length ; ++i )   // init to S_OK
         errors[i] = HRESULTS.S_OK;

      // TO-DO: write the new values to the device
      for( int i=0 ; i<values.Length ; ++i )     // handle all items
      {
         int ItemHandle = values[i].Handle ;
         Items[ItemHandle].Value = values[i].Value; // write value into buffer
```

```
        }

        return HRESULTS.S_OK;
    }

    new public int WriteItems(DeviceItemValue[] values, out int[] errors)
    {
        return WriteItems(0, values, out errors);
    }

    static ItemDef[]   Items = { new ItemDef("Coluna.Feed", (double)1 ),
                                 new ItemDef("Coluna.Composicao", (double)0.5 ),
                                 new ItemDef("Coluna.Qualidade", (double)1 ),
                                 new ItemDef("Coluna.Refluxo", (double)2.706 ),
                                 new ItemDef("Coluna.Vapor", (double)3.206 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel0", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel1", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel2", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel3", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel4", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel5", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel6", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel7", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel8", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel9", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel10", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel11", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel12", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel13", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel14", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel15", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel16", (double)0.5 ),
                                 new ItemDef("Coluna.Concentracoes.Nivel17", (double)0.5 ),
```

```
                                new ItemDef("Coluna.Concentracoes.Nivel18", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel19", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel20", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel21", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel22", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel23", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel24", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel25", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel26", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel27", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel28", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel29", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel30", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel31", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel32", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel33", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel34", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel35", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel36", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel37", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel38", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel39", (double)0.5 ),
                                new ItemDef("Coluna.Concentracoes.Nivel40", (double)0.5 ),

                            };
    static Thread      myThread ;
    static ManualResetEvent StopThread = null ;



    //============================================================================
    // This method simulates item value changes and writes the changed values to the generic
        server cache. xxxxxx
```

```
    void RefreshThread()
    {
       for(;;)  // forever thread loop
       {
          // increment readable items of type Int, Short, Float or Double
          for( int i=0 ; i<Items.Length ; ++ i )
          {

             SetItemValue( i, Items[i].Value, (short)OPCQuality.GOOD, DateTime.UtcNow );
          }

          Thread.Sleep( 500 ) ;  // ms

          if( StopThread != null )
          {
             StopThread.Set();
             return;             // terminate the thread
          }
       }
    }

}

//-----------------------------------------------------------------
// Item definition and value buffer
public class ItemDef
{
   public string ID ;
   public object Value;

   public ItemDef( string id, object val )
   {
      ID = id ;
```

```
        Value = val ;
      }
   }


}
```

A.2 Simulador

```
using System; using System.Collections.Generic; using
System.ComponentModel; using System.Data; using System.Drawing;
using System.Linq; using System.Text; using System.Threading.Tasks;
using System.Windows.Forms; using OPCDA.NET; using OPCDA; using OPC;

namespace ColunaDistilacao {
    public partial class Form1 : Form
    {
        ShowBrowseTreeList ItemShowTreeList;

        OpcServer OpcSrv = null;

        OpcServer servidor;

        Euler integrador = new Euler();

        List<Concentracoes> ListaConcentracoes = new List<Concentracoes>();

        string computador = "";

        double Xvelho = 0;
        double Xnovo = 0;
```

```
double alpha = 1.5;  //volatilidade relativa
int ns = 40;         //número de níveis (41)
int nf = 20;         //estágio de alimentação (21)
double feed = 1;     //fluxo de alimentação (mol/min) (1)
double zfeed = 0.5;  //composição inicial da alimentação
double qf = 1;       //qualidade da alimentação
double reflux = 2.706; //refluxo
double vapor = 3.206; //fluxo de vapor do refervedor
double md = 5;       //distillate molar hold-up (5)
double mb = 5;       //bottoms molar hold-up (5)
double mt = 0.5;     //stage molar hold-up (0.5)
double lr = 0;       //fluxo de líquido acima do estágio de alimentação
double ls = 0;       //fluxo de líquido abaixo do estágio de alimentação
double vs = 0;       //fluxo de vapor acima do estágio de alimentação
double vr = 0;       //fluxo de vapor abaixo do estágio de alimentação
double dist = 0;     //fluxo do destilado
double lbot = 0;     //fluxo do produto de baixo
int fatorAceleracao = 1;

double[] x = new double[41];    //concentrações de líquido
double[] funcao = new double[41]; //variável para Euler
double[] y = new double[41];    //concentrações de vapor

Dictionary<int, string> dicionario = new Dictionary<int, string>();
Dictionary<string, string> parametros = new Dictionary<string, string>();

List<Concentracoes> lc = new List<Concentracoes>();

public Form1()
{
    InitializeComponent();
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    int j = 0;
    for (j = 0; j <= ns; j++)
    {
        x[j] = 0.5;
        y[j] = 0;
    }

    parametros.Add("Alimentacao", "Coluna.Feed");
    parametros.Add("Refluxo", "Coluna.Refluxo");
    parametros.Add("Vapor", "Coluna.Vapor");

    dicionario.Add(0, "Coluna.Concentracoes.Nivel0");
    dicionario.Add(1, "Coluna.Concentracoes.Nivel1");
    dicionario.Add(2, "Coluna.Concentracoes.Nivel2");
    dicionario.Add(3, "Coluna.Concentracoes.Nivel3");
    dicionario.Add(4, "Coluna.Concentracoes.Nivel4");
    dicionario.Add(5, "Coluna.Concentracoes.Nivel5");
    dicionario.Add(6, "Coluna.Concentracoes.Nivel6");
    dicionario.Add(7, "Coluna.Concentracoes.Nivel7");
    dicionario.Add(8, "Coluna.Concentracoes.Nivel8");
    dicionario.Add(9, "Coluna.Concentracoes.Nivel9");
    dicionario.Add(10, "Coluna.Concentracoes.Nivel10");
    dicionario.Add(11, "Coluna.Concentracoes.Nivel11");
    dicionario.Add(12, "Coluna.Concentracoes.Nivel12");
    dicionario.Add(13, "Coluna.Concentracoes.Nivel13");
    dicionario.Add(14, "Coluna.Concentracoes.Nivel14");
    dicionario.Add(15, "Coluna.Concentracoes.Nivel15");
    dicionario.Add(16, "Coluna.Concentracoes.Nivel16");
```

```
dicionario.Add(17, "Coluna.Concentracoes.Nivel17");
dicionario.Add(18, "Coluna.Concentracoes.Nivel18");
dicionario.Add(19, "Coluna.Concentracoes.Nivel19");
dicionario.Add(20, "Coluna.Concentracoes.Nivel20");
dicionario.Add(21, "Coluna.Concentracoes.Nivel21");
dicionario.Add(22, "Coluna.Concentracoes.Nivel22");
dicionario.Add(23, "Coluna.Concentracoes.Nivel23");
dicionario.Add(24, "Coluna.Concentracoes.Nivel24");
dicionario.Add(25, "Coluna.Concentracoes.Nivel25");
dicionario.Add(26, "Coluna.Concentracoes.Nivel26");
dicionario.Add(27, "Coluna.Concentracoes.Nivel27");
dicionario.Add(28, "Coluna.Concentracoes.Nivel28");
dicionario.Add(29, "Coluna.Concentracoes.Nivel29");
dicionario.Add(30, "Coluna.Concentracoes.Nivel30");
dicionario.Add(31, "Coluna.Concentracoes.Nivel31");
dicionario.Add(32, "Coluna.Concentracoes.Nivel32");
dicionario.Add(33, "Coluna.Concentracoes.Nivel33");
dicionario.Add(34, "Coluna.Concentracoes.Nivel34");
dicionario.Add(35, "Coluna.Concentracoes.Nivel35");
dicionario.Add(36, "Coluna.Concentracoes.Nivel36");
dicionario.Add(37, "Coluna.Concentracoes.Nivel37");
dicionario.Add(38, "Coluna.Concentracoes.Nivel38");
dicionario.Add(39, "Coluna.Concentracoes.Nivel39");
dicionario.Add(40, "Coluna.Concentracoes.Nivel40");

for (int i = 0; i < 41; i++)
{
    Concentracoes c = new Concentracoes();

    c.concentracaoLiquido = x[i];
    c.concentracaoVapor = y[i];
    c.index = i;
```

```
            ListaConcentracoes.Add(c);
        }




        for (int i = 0; i < 44; i++)
        {
            Concentracoes c = new Concentracoes();


            lc.Add(c);
        }


        dataGridView2.DataSource = lc;
        dataGridView2.Refresh();


        dataGridView2.Rows[0].Cells[0].Value = "Alimentacao";
        dataGridView2.Rows[0].Cells[1].Value = parametros["Alimentacao"];


        dataGridView2.Rows[1].Cells[0].Value = "Refluxo";
        dataGridView2.Rows[1].Cells[1].Value = parametros["Refluxo"];


        dataGridView2.Rows[2].Cells[0].Value = "Vapor";
        dataGridView2.Rows[2].Cells[1].Value = parametros["Vapor"];


        for (int i = 3; i < dataGridView2.Rows.Count; i++)
        {
            dataGridView2.Rows[i].Cells[0].Value = i-3;
            dataGridView2.Rows[i].Cells[1].Value = dicionario[i-3];
        }

    }
```

```
public void dinamica()
{
    lr = reflux;
    ls = reflux + qf * feed;

    vs = vapor;
    vr = vs + feed * (1 - qf);

    dist = vr - reflux;
    lbot = ls - vs;

    for (int i = 0; i <= ns; i++)
    {
        y[i] = (alpha * x[i]) / (1 + (alpha - 1) * x[i]);
    }

    funcao[0] = (1 / md) * (vr * y[1] - (dist + reflux) * x[0]);

    for (int i = 1; i <= nf - 1; i++)
    {
        funcao[i] = (1 / mt) * (lr * x[i - 1] + vr * y[i + 1] - lr * x[i] - vr * y[i]);
    }

    funcao[nf] = (1 / mt) * (lr * x[nf - 1] + vs * y[nf + 1] - ls * x[nf] - vr * y[nf] +
         feed * zfeed);

    for (int i = nf + 1; i <= ns - 1; i++)
    {
        funcao[i] = (1 / mt) * (ls * x[i - 1] + vs * y[i + 1] - ls * x[i] - vs * y[i]);
    }

    funcao[ns] = (1 / mb) * (ls * x[ns - 1] - lbot * x[ns] - vs * y[ns]);
```

```
        x = integrador.integrate(x, funcao);
    }

    public void pontoOperacao(int passos)
    {
        for (int q = 0; q < passos; q++)
        {
            dinamica();
        }
    }


    private void timerSimulacao_Tick(object sender, EventArgs e)
    {
        SyncIOGroup SyncRWGroup = new SyncIOGroup(servidor);

        object value = "";

        OPCItemState Rslt;

        int rtc = SyncRWGroup.Read(OPCDATASOURCE.OPC_DS_CACHE, parametros["Alimentacao"],
            out Rslt);
        feed = (double)Rslt.DataValue;

        int rtc2 = SyncRWGroup.Read(OPCDATASOURCE.OPC_DS_CACHE, parametros["Refluxo"], out
            Rslt);
        reflux = (double)Rslt.DataValue;

        int rtc3 = SyncRWGroup.Read(OPCDATASOURCE.OPC_DS_CACHE, parametros["Vapor"], out
            Rslt);
        vapor = (double)Rslt.DataValue;

        textBoxFluxoAlimentação.Text = feed.ToString();
```

```
textBoxRefluxo.Text = reflux.ToString();
textBoxRefervedor.Text = vapor.ToString();




for (int i = 0; i < fatorAceleracao; i++)
{
    dinamica();
}




atualizarGrafico();




textBox1.Text = x[0].ToString();
textBox2.Text = x[nf].ToString();
textBox3.Text = x[ns].ToString();

Xvelho = Xnovo;

for (int i = 0; i < 41; i++)
{
    //SyncIOGroup SyncRWGroup = new SyncIOGroup(servidor);
    int rtc4 = SyncRWGroup.Write(dicionario[i], x[i]);

    ListaConcentracoes[i].concentracaoLiquido = x[i];
    ListaConcentracoes[i].concentracaoVapor = y[i];
```

```
        }

        dataGridView1.DataSource = ListaConcentracoes;
        dataGridView1.Refresh();

        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            dataGridView1.Rows[i].Cells[0].Value = ListaConcentracoes[i].index;
            dataGridView1.Rows[i].Cells[1].Value = ListaConcentracoes[i].concentracaoLiquido;
            dataGridView1.Rows[i].Cells[2].Value = ListaConcentracoes[i].concentracaoVapor;
        }


    }

    int a = 0;


    public class Concentracoes
    {
        public int index;
        public double concentracaoLiquido;
        public double concentracaoVapor;
    }

    private void buttonConfirmar_Click_1(object sender, EventArgs e)
    {

        servidor = new OpcServer();
        servidor.Connect(computador, "Advosol.DA3SimpleCS.1");

        try
        {
```

```
            alpha = Convert.ToDouble(textBoxVolatilidade.Text);
            feed = Convert.ToDouble(textBoxFluxoAlimentação.Text);
            zfeed = Convert.ToDouble(textBoxComposicao.Text);
            qf = Convert.ToDouble(textBoxQualidade.Text);
            reflux = Convert.ToDouble(textBoxRefluxo.Text);
            vapor = Convert.ToDouble(textBoxRefervedor.Text);
            md = Convert.ToDouble(textBoxMd.Text);
            mb = Convert.ToDouble(textBoxMb.Text);
            mt = Convert.ToDouble(textBoxMt.Text);

            if (a == 0)
                buttonPassos.Enabled = true;
        }

        catch
        {
            MessageBox.Show("Um ou mais parâmetros não são números", "Erro");
        }
    }

    private void groupBox5_Enter(object sender, EventArgs e)
    {

    }

    private void buttonPassos_Click(object sender, EventArgs e)
    {
        try
        {
            int passos = Convert.ToInt32(textBoxPassos.Text);
            pontoOperacao(passos);
            buttonIniciar.Enabled = true;
```

```
            buttonPassos.Enabled = false;
        }



        catch
        {
            MessageBox.Show("Número de passos deve ser inteiro.", "Erro");
        }


        textBox1.Text = x[0].ToString();
        textBox2.Text = x[nf].ToString();
        textBox3.Text = x[ns].ToString();
    }


    private void button1_Click_2(object sender, EventArgs e)
    {
        fatorAceleracao = Convert.ToInt32(textBoxFatorAceleracao.Text);
    }


    private void buttonIniciar_Click(object sender, EventArgs e)
    {
        if (buttonIniciar.Text == "Iniciar Simulação")
        {
            timerSimulacao.Enabled = true;
            buttonIniciar.Text = "Parar Simulação";
            buttonPassos.Enabled = false;
            a = 1;
        }


        else if (buttonIniciar.Text == "Parar Simulação")
        {
            timerSimulacao.Enabled = false;
            buttonIniciar.Text = "Iniciar Simulação";
```

```
    }
}

public void atualizarGrafico()
{

    int numberOfPointsInChart = 200;
    int numberOfPointsAfterRemoval = 150;



    Xnovo = (Xvelho + 0.5 * fatorAceleracao);

    chartConcentracoes.Series[0].Points.AddXY(Xnovo, x[0]);
    chartConcentracoes.Series[1].Points.AddXY(Xnovo, x[nf]);
    chartConcentracoes.Series[2].Points.AddXY(Xnovo, x[ns]);

    chartConcentracoes.ResetAutoValues();

    while (chartConcentracoes.Series[0].Points.Count > numberOfPointsInChart)
    {
        // Remove data points on the left side
        while (chartConcentracoes.Series[0].Points.Count > numberOfPointsAfterRemoval)
        {
            chartConcentracoes.Series[0].Points.RemoveAt(0);
            chartConcentracoes.Series[1].Points.RemoveAt(0);
            chartConcentracoes.Series[2].Points.RemoveAt(0);
        }

        int numpontos = chartConcentracoes.Series[0].Points.Count();

        // Adjust X axis scale
        chartConcentracoes.ChartAreas[0].AxisX.Minimum =
             chartConcentracoes.Series[0].Points[0].XValue;
```

```
            chartConcentracoes.ChartAreas[0].AxisX.Maximum =
                chartConcentracoes.Series[0].Points[numpontos - 1].XValue;
        }

        // Invalidate chart
        chartConcentracoes.Invalidate();

    }

    private void button2_Click(object sender, EventArgs e)
    {
        cbMachine.Items.Clear();



        try
        {
            string[] comps = OPC.Common.ComApi.EnumComputers();
            this.cbMachine.Items.AddRange(comps);
            cbMachine.Text = cbMachine.Items[0].ToString();

            if (cbMachine.Items.Count == 0)
            {
                cbMachine.Text = Environment.MachineName;
            }

        }

        catch
        {
            cbMachine.Text = Environment.MachineName;
        }
    }
```

```
private void btnBrowseOPCServers_Click(object sender, EventArgs e)
{
    try
    {
        cbOPCServers.Items.Clear();

        this.Update();
        OpcServerBrowser SrvList = new OpcServerBrowser(cbMachine.Text);
        string[] Servers = null;
        // Browse OPC DA V2 and V3 servers
        SrvList.GetServerList(true, true, out Servers);

        if (Servers != null)
        {
            cbOPCServers.Items.AddRange(Servers);
            cbOPCServers.SelectedIndex = 0;
        }
    }

    catch
    {
        MessageBox.Show("Não foi possível realizar o browse", "ERRO",
             MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void btnConnect_Click(object sender, EventArgs e)
{
    if (OpcSrv != null)
    {

        return;
```

```
        }

        tvServerItems.Nodes.Clear();
        this.Update();
        try
        {
            OpcSrv = new OpcServer();
            OpcSrv.Connect(cbMachine.Text, cbOPCServers.Text);

            try
            {
                ItemShowTreeList = OpcSrv.ShowBrowseTreeList(tvServerItems, lvBranchItems);
                ItemShowTreeList.BrowseModeOneLevel = true;
                ItemShowTreeList.Show(OpcSrv.ServerName);
            }
            catch (Exception ex)
            {

            }

        }
        catch (Exception ex)
        {
            OpcSrv = null;
        }
    }

    private void btnDisconnect_Click(object sender, EventArgs e)
    {
        tvServerItems.Nodes.Clear();
        lvBranchItems.Items.Clear();

        if (OpcSrv != null)
```

```
        {
            OpcSrv.Disconnect();
            OpcSrv = null;
        }
    }



    private void buttonAssociar_Click(object sender, EventArgs e)
    {
        try
        {

            computador = cbMachine.SelectedItem.ToString();

            ListViewItem lvItem = lvBranchItems.SelectedItems[0];
            string variavel = comboBoxVariavel.SelectedItem.ToString();

            if (variavel == "Alimentacao" || variavel == "Refluxo" || variavel == "Vapor")
            {
                parametros[variavel] = lvItem.Tag.ToString();
            }

            else
            {
                dicionario[Convert.ToInt32(variavel)] = lvItem.Tag.ToString();
            }



            for (int i = 0; i < dataGridView2.Rows.Count; i++)
            {
                if (dataGridView2.Rows[i].Cells[0].Value.ToString() ==
                    comboBoxVariavel.SelectedItem.ToString())
```

```
                {
                    dataGridView2.Rows[i].Cells[1].Value = lvItem.Tag.ToString();
                    break;
                }

            }



        }

        catch
        {

        }
    }


}
```

A.3 Classe Euler

```
using System.Linq; using System.Text; using System.Threading.Tasks;

namespace ColunaDistilacao {
    public class Euler
    {
        public Euler()
        {
            intType = "Euler";
            step = 0.033;
```

```
}


private string intType;

public string IntType
{
    get
    {
        return intType;
    }
    set
    {
        intType = value;
    }
}

private double step;

public double Step
{
    get
    {
        return step;
    }
    set
    {
        step = value;
    }
}
```

```
        public double[] integrate(double[] x1, double[] f)
        {
            double[] x2 = new double[x1.Length];

            for (int i = 0; i < x2.Length; ++i)
                x2[i] = x1[i];


            for (int i = 0; i < x2.Length; ++i)
                x2[i] = x2[i] + Step * f[i];

            return x2;

        }

    }
}
```