UNIVERSIDADE FEDERAL DE MINAS GERAIS – UFMG
DEPARTAMENTO DE ENGENHARIA QUÍMICA – DEQ
CURSO DE ENGENHARIA QUÍMICA

HISTORIADOR DE PROCESSOS INDUSTRIAIS: ANÁLISE E VISUALIZAÇÃO DE DADOS COM O SOFTWARE ELIPSE PLANT MANAGER (EPM)

BRUNO LACERDA CAMPOS, FILIPE PINHO DE MELLO, HARNON MARTINS RAMOS, MARIANA FARIA FRAGA, VINÍCIUS MACHADO CAMPOS E SOUZA

LABORATÓRIO DE OPERAÇÕES E PROCESSOS

BELO HORIZONTE
2016

HISTORIADOR DE PROCESSOS INDUSTRIAIS: ANÁLISE E VISUALIZAÇÃO DE DADOS COM O SOFTWARE ELIPSE PLANT MANAGER (EPM)

BRUNO LACERDA CAMPOS, FILIPE PINHO DE MELLO, HARNON MARTINS RAMOS, MARIANA FARIA FRAGA, VINÍCIUS MACHADO CAMPOS E SOUZA

Relatório apresentado à disciplina Laboratório de Operações e Processos, do Programa de Graduação em Engenharia Química da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do título de "Engenheiro Químico".

Orientado: Prof. Gustavo Matheus de Almeida

Co-orientador: Eng. Rodrigo Cesar de Miranda

BELO HORIZONTE

2016

Agradecimentos

Ao nosso orientador Prof. Gustavo Almeida e ao nosso co-orientador Rodrigo Miranda, pelo direcionamento, correções e suporte prestados no decorrer desse trabalho.

À Elipse Software, na figura de Humberto e Felipe Gabriel, pelo excelente suporte técnico prestado.

Aosalunos de Ciência da Computação, do Laboratório de Banco de Dados, pelo suporte na formatação do banco de dados utilizado.

Aos amigos Daniel Coelho e Márcio Neto pela ajuda com a escrita dos códigos em Python.

Ao Departamento de Engenharia Química da UFMG e a todos os colegas e professores que fizeram parte da construção do nosso aprendizado.

Resumo

Este trabalho apresenta a implementação de uma técnica de controle estatístico (Análise por Componentes Principais) na linguagem Python e sua utilização em um software historiador de processos (Elipse Plant Manager, da Elipse Software). Utiliza o *Benchmarking Tennessee* como referência para a validação do modelo estatístico e apresenta algumas formas de visualização de dados que facilitam a interpretação das informações contidas nas variáveis.

PALAVRAS-CHAVE: Análise por Componentes Principais (PCA). Elipse Plant Manager (EPM). Visualização de Dados.

Abstract

This work presents a statistic control technique (Principal Component Analysis), its implementation in python as well as its integrated functionality in an industrial data management system (Elipse Plant Manager, from Elipse Software). In this work the Tennessee Benchmarking is used as a reference for validation and it presents some data visualization techniques which enable a better understanding of the variables behavior.

KEYWORDS: Principal Component Analysis. Elipse Plant Manager (EPM).Data Visualization.

Sumário

1 INTRODUÇÃO

1.1 Volume de dados

Segundo a Harvard Business Review(MCAFEE; BRYNJOLFSSON, 2012), no ano de 2012, aproximadamente 2,5 exabytes de dados (equivalente a um bilhão de gigabytes) eram criados a cada dia, e esse valor estaria se duplicando a cada 40 meses. Extrapolando tal estatística, a quantidade de dados criada atualmente estaria próxima de 4,75 exabytes por dia. Traduzindo para termos práticos, um exabyte corresponde a aproximadamente 20 bilhões de armários de arquivos de texto. Portanto, atualmente estaríamos produzindo diariamente um volume de dados correspondente à aproximadamente 95 bilhões de armários de arquivos de texto a cada dia. Essa realidade de um enorme volume de dados por segundo oferece às empresas a possibilidade de extrair informações nunca antes vislumbradas.

1.2 Como transformar dados em informações relevantes?

Apesar de os dados estarem disponíveis e serem cada vez mais acessíveis para as empresas, é necessário que ocorra um tratamento e posterior interpretação dos dados para geração de valor(TAN et al., 2015). Dessa forma, uma ampla discussão sobre como transformar uma avalanche de dados em informações úteis e gerar impacto é realizada atualmente.

É fácil perceber que nem todos os dados gerados atualmente são tratados, ignorando-se potenciais informações relevantes para a tomada de decisões de uma empresa (VERHAPPEN, 2013). A transformação de dados em informação relevante é realizada através de plataformas computacionais, como os historiadores de processonas indústrias químicas.Algumas empresas atuam positivamente nesse cenário, criando plataformas próprias para o tratamento de dados e obtenção de informação, como a Chevron que desenvolveu o *i-field*, que monitora e trata dados sobre poços de petróleo. O *i-*

field chega a compartilhar 1,5 Terabytes de dados diariamente, atuando desde a geofísica dos poços até a produção(VERHAPPEN, 2013).

Os dados podem ser transformados em informações através de duas grandes metodologias(TAN et al., 2015): através de algoritmos matemáticos (que ainda são considerados limitados para muitas aplicações) ou pela interpretação visual dos dados, como gráficos e imagens. Assim, dependendo do método como o dado será tratado, diferentes resultados podem ser obtidos desse tratamento, devido às diferenças intrínsecas de cada método ou, se considerarmos um mesmo método, sobre quais variáveis o usuário focou para a obtenção de informação (SAEY, 2016). Além disso, mesmo com a evolução da área de análise de dados, é necessário que pessoas analisem criticamente os resultados obtidos, conferindo se a análise foi realizada de forma correta e transformando os dados tratados em informações para possíveis tomadas de decisão (KEIM et al., 2008).

1.3 Visualização e Análise de dados

Devido à imensa quantidade de dados gerada atualmente, é necessário que os métodos de análise se tornem cada vez mais eficientes e mais confiáveis para se tornarem relevantes. Caso contrário, os sistemas se tornam lentos ou não geram as respostas requisitadas (FAIRCLOTH, 2016). Por isso, muito se trabalha na inovação de métodos de análise de dados, alterando as equações matemáticas que regem os algoritmos já existentes ou através de propostas de novas equações que gerem novos algoritmos mais eficientes.

Outra forma de se solucionar os problemas é através da representação dos dados em gráficos, mudando a maneira de como a informação pode ser obtida. Através da visualização dos dados, o usuário pode julgar quais são as informações relevantes geradas e, a partir delas, tomar suas próprias decisões, transformando um problema automatizado, e ainda passível de erros computacionais, em um sistema semi automatizado, mas que pode gerar resultados melhores devido à capacidade humana de percepção visual. Além disso, o ser humano pode distinguir mais facilmente que a máquina algumas

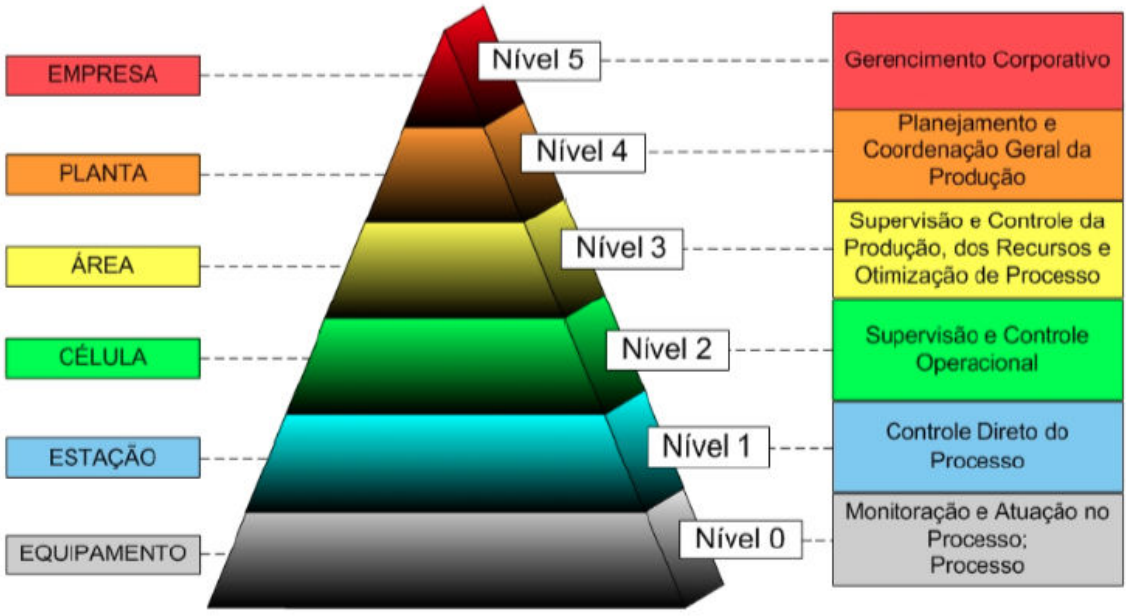
informações, como determinar informações ambíguas ou dados em conflito, reconhecer o esperado e o inesperado, e criar possíveis planos de ação para lidar com as informações geradas (KEIM et al., 2008).É importante ressaltar que a visualização de dados não é uma ciência exata e, por isso, é importante que haja um processo iterativo sobre os dados gerados. Dessa forma, é possível extrair informações mais valiosas uma vez que perspectivas diferentes podem resultar em aspectos diferentes e gerar *insights* mais relevantes.

Por outro lado, antes de se gerar dados de forma visual, é importante que ocorra um tratamento prévio desses dados, de modo geral. Além disso, nem todas as soluções são compatíveis com visualização de dados, sendo a análise tão importante e relevante nos problemas atuais. Um exemplo disso é a empresa ImageShack que usa a plataforma Hadoop – *software* em Java que é desenvolvido de forma comunitária e é voltado para *clusters* e processamento de *big data* – para analisar dados de seus usuários e gerar uma experiência mais agradável no domínio, aumentando a permanência dos usuários e lucrando mais através de anúncios. Ainda usando a metodologia Hadoop, a *start-up*Cloudera produz *softwares* exclusivos para empresas e indústrias tratarem seus dados e obterem informações da melhor maneira possível (FAIRCLOTH, 2016). Outro exemplo de metodologia de análise de dados é a Apache Spark, da empresa Intuit, que busca tratar dados industriais em tempo real e propiciar ao usuário uma tomada de decisão ainda durante o processo, não sendo necessário um atraso para o tratamento dos dados obtidos. Além disso, máquinas que podem “aprender” estão em desenvolvimento para abordarem problemas relacionados a*big data* e mostram grande evolução nos últimos anos, revelando grande potencial para a solução de problemas, especialmente do mundo financeiro (MITCHELL, 2014).

1.4 Processos químicos industriais

As indústrias de processos químicos correspondem a uma fatia da grande quantidade de dados gerada atualmente. Nesse caso, os dados percorrem um fluxo dentro do sistema de automação integrado da planta, cuja estrutura é ilustrada na Figura 1:

Figura 1: Estrutura hierárquica de um sistema de automação integrado



Fonte: YAMAGUCHI, 2006

A estrutura hierárquica, também conhecida como pirâmide da automação, é dividida em seis níveis, desde o chão de fábrica (nível 0) até o gerenciamento corporativo da empresa (nível 5). Essa estrutura permite uma melhor visualização, organização e entendimento dos sistemas de automação mais complexos (YAMAGUCHI, 2006).

Cada nível da pirâmide tem a sua função específica:

- Nível 5: Gerenciamento corporativo através de sistemas como o ERP (*Enterprise Resource Planning*) com a função de alcançar a missão da empresa e administrar a corporação;

- Nível 4: Planejamento da produção global da empresa através de sistemas corporativos de gerenciamento de produção com a função de planejar e programar a produção total;

- Nível 3: Supervisão e controle de produção, dos recursos e otimização do processo através de sistemas como o MES (*Manufacturing Execution System*), LIMS (*LaboratoryInformation Management System*), PIMS (*PlantInformation Management System*), AM (*Asset Management*) com a

função de coordenar a produção, dar suporte às atividades produtivas e cuidar da obtenção e recursos para as atividades produtivas;

● Nível 2: Coordenação de múltiplas máquinas e operações através de sistemas de supervisão e controle, supervisionando e controlando as atividades produtivas e serviços de suporte à produção no chão de fábrica;

● Nível 1: Comando de máquinas, sequências e equipamentos através de controladores numéricos e CLPs (Controladores Lógico Programáveis);

● Nível 0: Ativação de sequências e movimentos através de equipamentos, dispositivos, máquinas e atuadores diretos com a função de executar os comandos para os equipamentos de chão de fábrica (atuação direta no processo).

A troca de dados entre os equipamentos e sistemas em cada nível (comunicação horizontal) ou entre níveis adjacentes (comunicação vertical) é feita através de redes de comunicação.

A indústria de processos químicos também requer grande velocidade e capacidade de análise de dados. Por exemplo, uma planta típica de produção de olefinas possui mais de 5.000 variáveis que devem ser monitoradas. Plantas industriais de grande escala chegam a até 20.000 variáveis de processo. Além disso, como os dados são correlacionados, é necessário fazer avaliações simultâneas.

Os seres humanos têm dificuldade de analisar, de modo simultâneo,problemas que envolvam mais de três variáveis, e isso se torna ainda mais acentuado quando os dados estão corrompidos com ruídos e incertezas. A necessidade de utilizar assistência computacional no processamento de dados tem se tornado uma grande preocupação e é importante que sistemas de análise de dados sejam desenvolvidos e integrados à pirâmide de automação da empresa. Deve-se levar em conta que operadores, supervisores e gestores tem visões diferentes do processo (visão local *vs* visão global), de forma que o programa deve ter diferentes interfaces voltadas para cada usuário. Uma ferramenta importante utilizada para o

gerenciamento de dados, localizada no nível três da pirâmide integrada, é o historiador de processo.

1.5 Visão geral sobre historiadores de processo

Os historiadores de processo tornaram-se populares em plantas industriais na década de 1980. Atualmente, são usados em toda a indústria para monitorar as operações, identificar problemas e encontrar oportunidades de melhoria. Os primeiros precursores dos historiadores foram os registradores gráficos eletromecânicos, que produziam gráficos de uma ou mais variáveis medidas durante um período de tempo e atuavam como registro permanente de informações críticas.

Os historiadores coletam dados em tempo real para armazená-los de forma estruturada e manter uma cronologia de informações. Estas informações do processo industrial são, então, disponibilizadas para qualquer usuário para consulta e análise. A grande vantagem dos historiadores é a capacidade de pesquisar e correlacionar grandes volumes de dados, gerando formas de visualização mais acessíveis e claras para identificar tendências e relacionamentos, que após análise se tornam informações úteis para serem consideradas em uma tomada de decisão.

Dessa forma, entre os usos dos historiadores de processo, estão (AQUARIUS, 2016):

- Verificações legais e garantia da qualidade: as empresas devem manter um registro de genealogia de produção e de testes de qualidade, tanto para verificações legais quanto de conformidade. Os historiadores armazenam dados detalhados que podem ser usados na defesa contra processos judiciais e para determinar quais produtos apresentam uma não-conformidade e devem ser recolhidos para retrabalho ou descarte.
- Controlar e rastrear serialização: devido às iniciativas TnT (Trackand Trace) de serialização para manter o histórico dos produtos (especialmente em empresas alimentícias e farmacêuticas), há grande necessidade de capturar e reter os registros históricos de produção. Assim, é possível estocar esses dados de maneira organizada e permitir boa e rápida visualização.

- Análise de causa: quando ocorre algum problema/falha na produção, os dados históricos do processo são fundamentais na identificação de fontes do problema usando a análise da causa raiz, que, ao serem identificadas e corrigidas, impedirão que resultados indesejáveis ocorram novamente.
- Otimização: através dos dados históricos de um processo, pode-se conhecer melhor esse processo e identificar etapas operando fora do ponto ótimo. Através de um software de simulação, pode-se colocar essas informações como dados de entrada e simular métodos de otimização.
- Monitoramento de energia: usando um historiador, juntamente com a medição setorizada e o monitoramento de energia, pode-se alocar custos energéticos nas etapas de produção para alcançar melhor distribuição dos custos, encontrar problemas e propor melhorias.
- Análise preditiva: mediante análise de tendências e padrões dos dados de um historiador, pode-se usar técnicas avançadas para prever falhas e eventos.
- Justificativa de investimentos: ao utilizar dados históricos de plantas reais, têm-se propostas de investimento muito mais sólidas e bem justificadas que aquelas baseadas apenas em estimativas.

2 OBJETIVOS

O objetivo geral do presente trabalho é aplicar a análise e visualização de dados em historiadores de processo, e integrar um historiador de processo com ferramentas não diretamente disponíveis em sua plataforma.

Dessa forma, para atingir o objetivo geral, três objetivos específicos são propostos: implementação da técnica estatística Análise por Componentes Principais (PCA) através da linguagem Python para a análise de dados; geração de gráfico bolha via Python para a visualização e diagnóstico; e finalmente, a comunicação entre um historiador de processo e o PCA, integrando-o com uma ferramenta não nativa.

O foco do trabalho é a área de monitoramento de processos químicos. O estudo de caso utilizado é o Benchmark Tennessee(seção 4), e o historiador de processos utilizado é o Elipse Plant Manager (EPM), da empresa nacional Elipse Software (seção 3.1.5).

3 REVISÃO BIBLIOGRÁFICA

3.1 Historiadores de processos

Comparando os historiadores de processos disponíveis atualmente, percebe-se grande semelhança entre os recursos ofertados. A Corso Systems(2013)afirma que isso ocorre porque cada empresa absorve recursos de pacotes concorrentes que o mercado considera importantes, se diferenciando apenas em alguns pontos.

Normalmente, as diferenças entre os *softwares* são observadas na maneira de configuração do sistema ou armazenamento/recuperação dos dados. Dessa forma, de acordo com a Sylution(2013)**,** são listadas algumas características que devem ser observadas para identificar bons historiadores de processos dos demais:

1. Cumprimento das funcionalidades básicas;
2. Velocidade e exatidão dos cálculos;
3. Recursos extras;
4. Facilidade de inserir equações no sistema.

Assim, as características mais importantes para um historiador de processos são o armazenamento, compressão e recuperação de dados, atribuindo selos de tempo e junção dos dados. Então, o pacote deve fazer isso de forma rápida sem descartar a exatidão dos cálculos realizados, garantindo resultados confiáveis com o menor atraso possível. Os recursos extras também garantem melhores experiências aos usuários, como a redundância (modo alternativo de armazenamento de dados), função *storeandforward* ou a possibilidade de calcular funções estatísticas básicas, como máximos/mínimos e médias. A possibilidade de interação com as variáveis analisadas através de equações também é um fator que garante qualidade ao pacote, permitindo ao usuário o acompanhamento de parâmetros importantes para seu sistema diretamente a partir do historiador(SYLUTION,2013)**.**

Em um segundo momento, existem outros fatores importantes para se considerar na escolha do historiador mais adequado para a aplicação industrial (CORSO SYSTEMS, 2015):

1. Integração do historiador com demais *softwares* utilizados na empresa;
2. Custos de implantação/instalação e manutenção;
3. Sistema operacional empregado.

A conexão entre os programas computacionais utilizados pela empresa é valiosa, garantindo perfeita operação e compasso entre todos os sistemas, gerando os melhores resultados possíveis. Além disso, alguns historiadores apresentam conexão com pacotes básicos, como o *Microsoft Excel*, expandindo a possibilidade de utilização dos dados coletados. Por outro lado, o custo de implantação/instalação e manutenção também deve ser analisado, visto que alguns pacotes se tornam mais onerosos com o aumento das variáveis analisadas ("*tags*"). Finalmente, o sistema operacional vigente da empresa também deve ser compatível com o historiador, visto que a maioria deles é desenhada para *Microsoft Windows*, impossibilitando sua aplicação em sistemas como *Apple* ou *Linux*, por exemplo (CORSO SYSTEMS, 2015).

Em termos de acesso aos dados, existem algumas possibilidades atualmente. São ofertadas arquiteturas do tipo multi-camadas, *mult-site* e de alta disponibilidade/missão crítica. Assim, cada sistema possui uma maneira própria de interação entre usuários e máquinas (HMI - *Human-machine interface*), além do acesso aos dados fora do sistema, como aqueles que apresentam conectividade com o *Excel*.

A Tabela 1mostra os principais *players* do mercado no ano de 2015. Descreve-se, a seguir, alguns desses e outros historiadores de processos disponíveis no mercado.

Tabela 1 : Principais players do mercado de historiadores em 2015.

Empresa	Produto	Endereço
OSIsoft	PI System	www.osisoft.com
AspenTech	Infoplus 21	www.aspentech.com
General Electric	Proficy Historian	www.geautomation.com
Schneider Electric	Wonderware	www.wonderware.com

Fonte:Mina Andrawos (2015)

3.1.1 OSIsoft PI System

OSI Soft PI é um historiador que apresentou grande desenvolvimento nos últimos cinco anos; mas, quando comparado aos demais, é financeiramente mais caro. Seu núcleo historiador é incluído na plataforma *Asset Framework*, que também apresenta novas funcionalidades, como a combinação de séries de dados, gerando relatórios efetivos de desempenho. Além disso, também possui integração com a internet, ou API (*ApplicationPrograming Interface*). Ainda, apresenta um portal na internet para acesso de dados históricos e análise instantânea. Finalmente, ele também conta com recursos de alta disponibilidade e redundância, garantindo a segurança dos dados mesmo após eventuais incidentes, como a queda do servidor local (OSI SOFTWARES, 2016)**.**

3.1.2 AspenTechInfoplus 21

O Infoplus 21 (IP21) é um historiador que possui diversas extensões, se tornando popular ao longo do tempo. Cerca de 500 indústrias o utilizam atualmente. A AspenTech comprou a empresa que o desenvolveu inicialmente, alterando seu nome.

O IP21 possui adaptador chamado CIMIO, que envia dados ao historiador de forma mais fácil. Além disso, também suporta ODBC/JDBC e, com sua linguagem específica (SQL+), permite a programação de comportamentos complexos no pacote (ASPENTECH, 2016).

3.1.3 Schneider Electric Wonderware

A Wonderware possui sistemas de supervisão e aquisição de dados (SCADA) e lançou recentemente uma parceria com a Microsoft, elaborando um historiador em nuvem. Assim, é utilizado o ambiente *Microsoft Windows AzureCloud*. Em sua versão mais recente (2014), houve um aumento de 400% no número possível de *tags* disponíveis, chegando a 2 milhões (SCHNEIDER ELETRONIC SOFTWARE, 2016).

3.1.4 General Eletric Proficy Historian

A General Electric (GE) combinou seu historiador de processos com tecnologia de *big data*, como a Hadoop com MapReduce. A versão mais recente (ProficyHistorian 6.0) apresenta espelhamento de banco de dados e redundância, administração do historiador pela internet, além de suporte à linguagem Python para tratamento dos dados, e conexão com o Excel (GENERAL ELECTRIC, 2016).

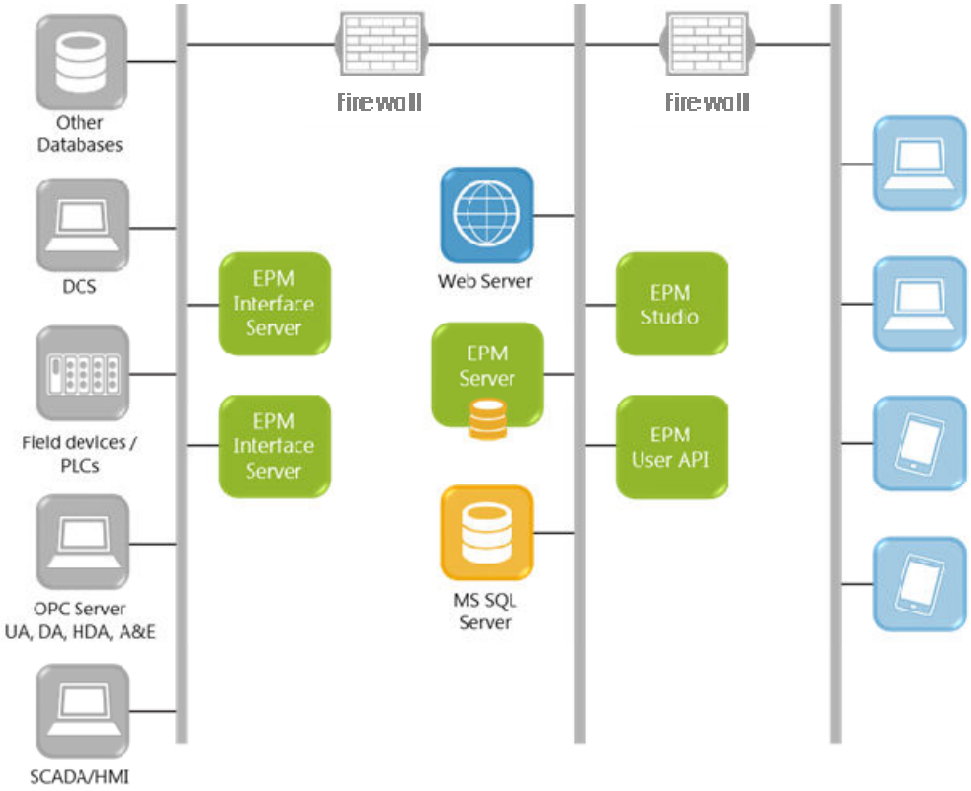
3.1.5 Elipse Plant Manager

O ElipsePlant Manager (EPM), utilizado nesse trabalho, é um *software* historiador de processos desenvolvido e lançado em 2008 pela Elipse Software. O EPM possibilita o armazenamento, gerenciamento e consolidação de dados provenientes de diversas fontes e promove a centralização desses dados em um só local, disponibilizando-os quando necessário, de maneira simples e rápida. O *software* auxilia na tomada de decisões em tempo real e na melhoria da cadeia produtiva, sendo, portanto, aplicável em todas as áreas onde seja necessária a consolidação de dados de processo.

O EPM adere ao padrão OPC UA (*OLE for ProcessControlUnifiedArchitecture*) e possui uma arquitetura orientada a serviços, com alta modularidade, escalabilidade e flexibilidade. O *software* oferece diversas ferramentas e possibilidades para o gerenciamento da informação e possui integração com as ferramentas da *Microsoft.*

O EPM é composto por diversos módulos para comunicação, processamento e armazenamento de dados, integrados a ferramentas de visualização e análise, ilustradas neste trabalho. Uma disposição típica desses módulos é mostrada no esquema da Figura 2. No entanto, essa disposição pode ser rearranjada de acordo com as necessidades a serem atendidas (ELIPSE SOFTWARE, 2016).

Figura 2: Diagrama de disposição do software EPM



Fonte: Elipse Software, 2016.

O sistema EPM é composto por alguns programas principais, descritos a seguir:

- EPM Server: servidor *OPC UA* que recebe e armazena os dados oriundos de diversas fontes, podendo disponibilizá-los para outros sistemas conectados a ele. Os dados podem chegar ao EPM Server por duas vias: uma para armazenamento e outra para disponibilização em tempo real. Da mesma forma, os sistemas conectados podem solicitar dados históricos ou de tempo

real. O EPM Server utiliza o *Microsoft SQL Server* para armazenamento dos dados de processo em disco. Os objetos do EPM Server com capacidade de armazenar dados históricos, ou seja, com valor, estampa de tempo e qualidade, são chamados de *Data Objects*. Os*DataObjects* são classificados em dois tipos, de acordo com a procedência dos dados relacionados a eles: *Basic Variables* (tipo mais elementar de variável, oriunda diretamente de interfaces de comunicação) e *Expression Variables* (variável proveniente de uma expressão utilizada, por exemplo, para o cálculo de indicadores em tempo real) (ELIPSE SOFTWARE, 2016);

-EPM Interface Server: aplicativo responsável pelo gerenciamento das interfaces de comunicação com as fontes de dados e pelo fluxo de dados com o EPM Server (ELIPSE SOFTWARE, 2016);

-Elipse OPC Proxy: módulo do sistema EPM que permite o acesso aos dados do EPM Server por sistemas de automação que sigam o padrão OPC Classic DA ou HDA. Possibilita a visualização de indicadores de desempenho calculados pelo EPM Server em telas, atuando como ferramenta de apoio à operação e auxiliando nas tomadas de decisão (ELIPSE SOFTWARE, 2016);

-EPM Studio: ferramenta composta por diversas funcionalidades de visualização de dados de processo, análise, gerenciamento e manutenção do sistema. Possui ferramentas intuitivas e ambiente integrado de análise com suporte à execução de códigos em linguagem Python (ELIPSE SOFTWARE, 2016);

-EPM Add-In for *Microsoft Excel*: permite a integração do EPM com o *Microsoft Excel*, facilitando as operações de consultas a dados do EPM a partir do *Excel* (ELIPSE SOFTWARE, 2016);

-EPM Webparts: Pacote de unidades modulares de informações que se comunicam diretamente com o EPM Server e são utilizadas para apresentação de informações relevantes em um *site*(gráficos de tendência, indicadores de desempenho e gráficos de pizza e de barras, por exemplo) (ELIPSE SOFTWARE, 2016).

3.2 Visualização de dados

A visualização de dados é a apresentação de quaisquer dados em forma gráfica ou visual, facilitando o entendimento de forma simples e intuitiva. Isso se deve ao fato de o cérebro humano interpretar mais facilmente as informações dispostas em formato visual quando comparado às informações dispostas em tabelas e textos.

A ideia de trabalhar com dados por meio de gráficos não é recente e as origens remotas da visualização estão nos diagramas geométricos, nas tabelas de posição das estrelas e nos mapas. No século XVI, com a expansão marítima da Europa, novas técnicas e instrumentos foram desenvolvidos, e com eles novas e mais precisasformas de apresentação visual do conhecimento, como ilustrado na Figura 3. (KANNO)

Figura 3:Representação antiga de uma cidade da Babilônia

Fonte: Infografe, 2016

Com o surgimento de novas necessidades, a infografia (ciência de representação) aperfeiçoou-se, buscando suprir as demandas existentes. A partir de 1975, com o desenvolvimento dos computadores com maior interação e fácil manipulação, grandes inovações nessa área foram desenvolvidas. Assim, novas técnicas gráficas bem como da visualização multidimensional surgiram.

Atualmente, com o desenvolvimento e a digitalização de ferramentas de trabalho, os sistemas empresariais e a visualização de dados travam outro embate: quantidades massivas de dados e a necessidade de extrair o máximo de informações que gerem vantagens para as empresas. Em processos

químicos industriais o cenário não é diferente. Para esse fim, vários tipos de gráfico são utilizados.

3.2.1 Tipos de gráficos

Inicialmente, algumas definições são importantes nesse contexto:

- Diagramas: gráficos geométricos dispostos em duas dimensões e que são mais usados na representação de séries estatísticas;

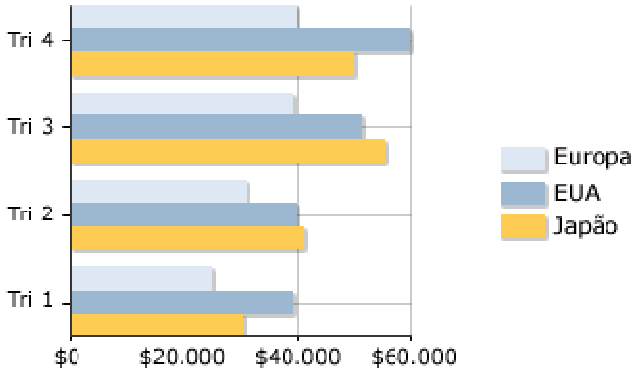
- Pictograma: os pictogramas representam a informação ilustrando-a com imagens relacionadas ao contexto do qual os dados fazem parte. Os símbolos mostrados devem ser auto-explicativos e de fácil entendimento, o que implica na desvantagem dos pictogramas de não mostrar detalhes minuciosos, limitando-se a uma visão geral do fenômeno;

-Estereogramas: são gráficos geométricos dispostos em três dimensões, usados nas representações gráficas das tabelas de dupla entrada;

Os tipos de gráficos mais comuns são sintetizados abaixo de acordo com suas principais funções. Muitos dos gráficos citados podem ser alocados em várias categorias, tamanha a versatilidade dos mesmos:

- Gráfico de Barras e de Colunas: o gráfico de barras (horizontais) ou de colunas é utilizado, em geral, para representar dados de uma tabela de frequências associadas a uma variável qualitativa ou a informações categóricas, permitindo a comparação entre diversas categorias, como ilustrado na Figura 4;

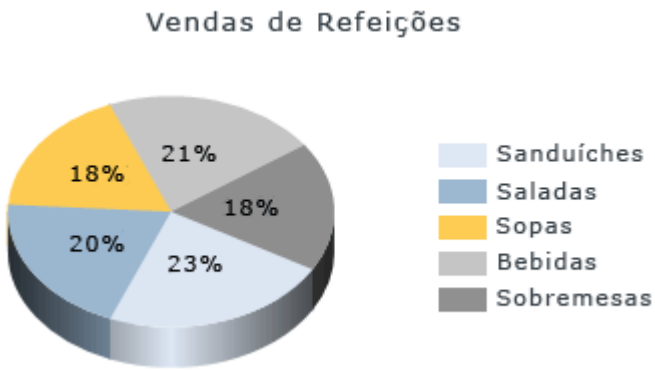
Figura 4: Gráfico de barras



Fonte: Microsoft, 2014

- Gráfico de Setores: o gráfico de setores, também conhecido como “gráfico de pizza”, é utilizado, em geral, para representar partes de um todo e fazer comparações entre grupos. Esse tipo de gráfico produz um impacto visual significativo, como mostrado na Figura 5.

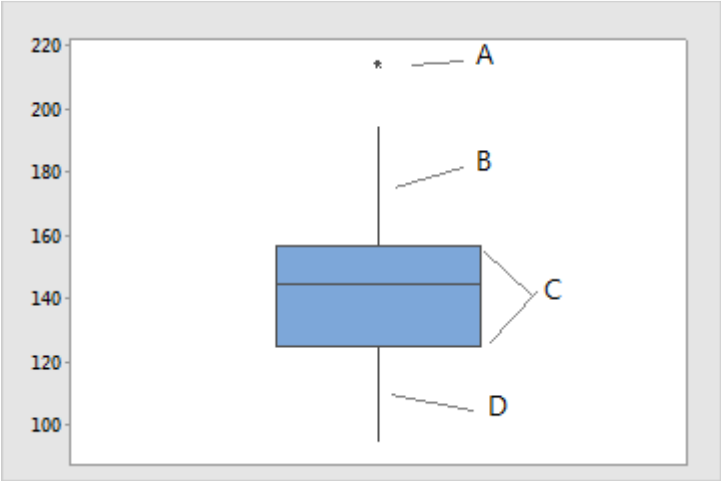
Figura 5: Gráfico de pizza



Fonte: Microsoft, 2014

- Boxplot: o boxplot, ou gráfico de caixa, é um sumário gráfico da distribuição de uma amostra que exibe sua forma, tendência central e variabilidade. Esse tipo de gráfico é apresentado na Figura 6.

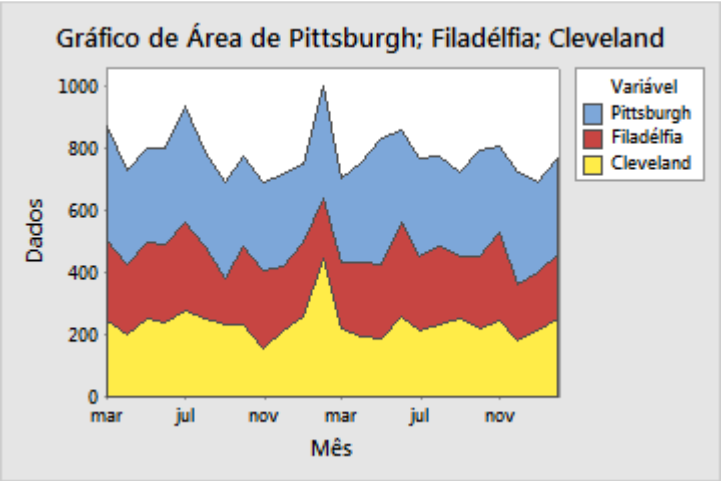
Figura 6:Boxplot



Fonte: “Suporte ao Minitab17”

- Gráfico de área: o gráfico de áreas exibe uma série como um conjunto de pontos conectados por uma linha, com toda a área preenchida abaixo da linha. Cada linha no gráfico é a soma acumulada e, portanto, pode avaliar contribuições de cada série a um total, ao longo do tempo. Enfatizam a magnitude da mudança no decorrer do tempo e podem ser usados para chamar a atenção para o valor total ao longo de uma tendência. Exibindo a soma dos valores, o gráfico de área mostra também a relação das partes com um todo (Figura 7).

Figura 7: Gráfico de área

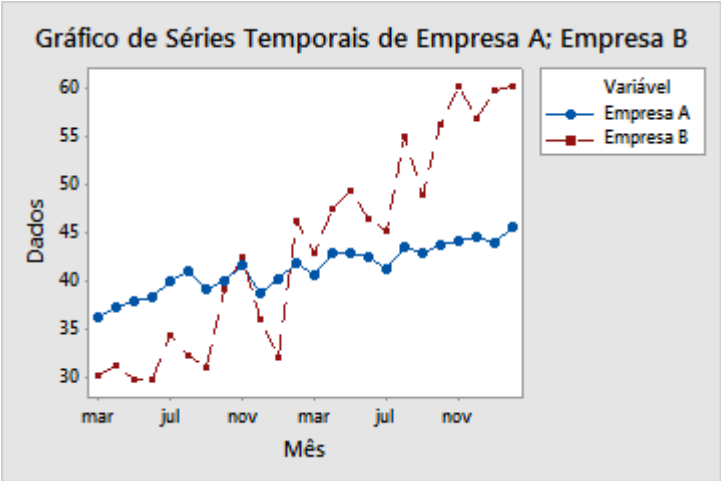


Fonte: “Suporte ao Minitab17”

- Gráficos de séries temporais: os gráficos de séries temporais são utilizados para avaliar padrões e comportamento dos dados ao longo do tempo, como por exemplo: examinar variações diárias, semanais, de sazonalidade ou

anuais, e efeitos antes e depois de uma mudança no processo, ilustrad na Figura 8.

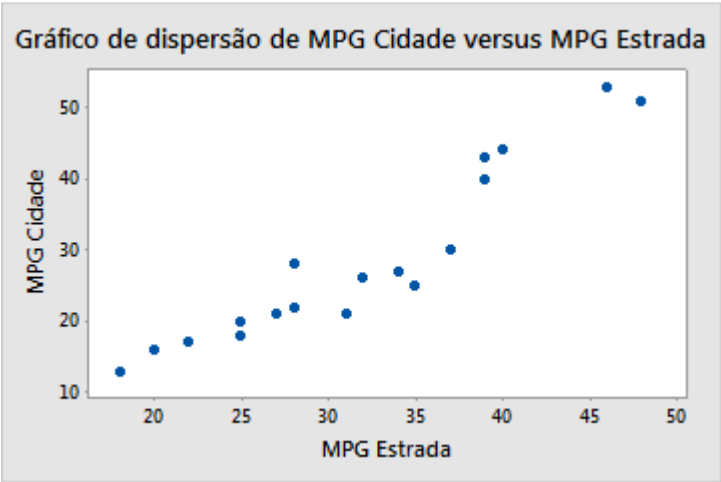
Figura 8:Gráfico de série temporal



Fonte: “Suporte ao Minitab17”

- Gráficos de dispersão (XY): O gráfico de dispersão é potencialmente interessante para explorar a relação entre um par de variáveis, sendo o mais utilizado em indústrias químicas para a análise de dados, mostrado na Figura 9.

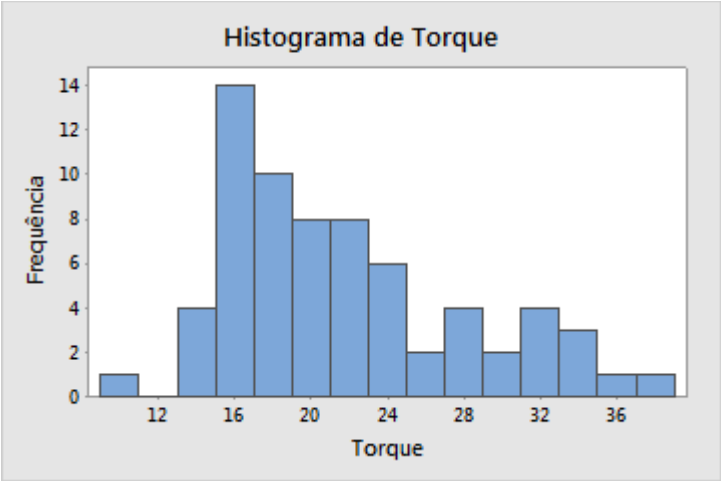
Figura 9: Gráfico de dispersão



Fonte: “Suporte ao Minitab17”

- Histograma: esse gráfico representa as frequências absolutas e relativas de dados agrupados em intervalos de classes para avaliar a forma e a distribuição de variáveis quantitativas contínuas, como visto na Figura 10.

Figura 10: Histograma

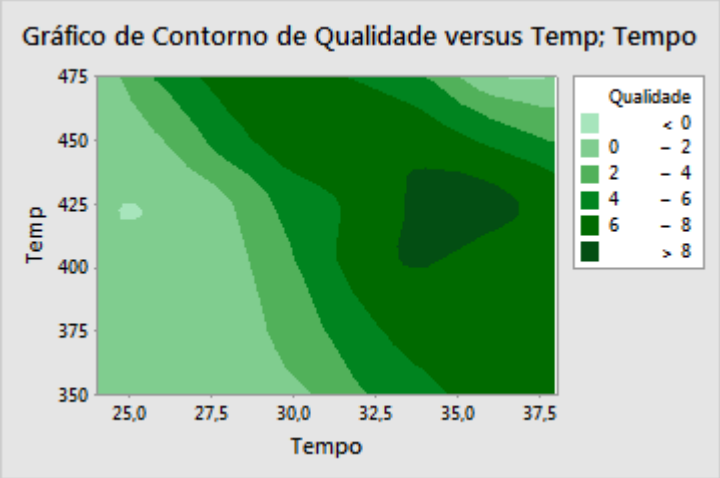


Fonte: “Suporte ao Minitab17”

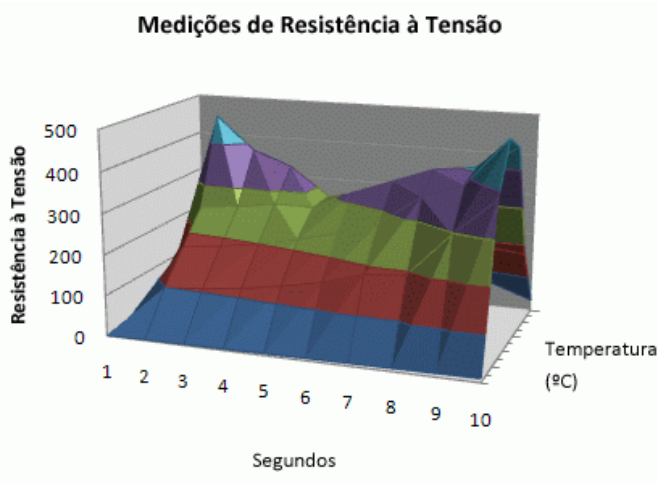
Além dos gráficos univariados, existem gráficos bivariados e trivariados, conforme aFigura 11:

Figura 11: Gráficos multivariados

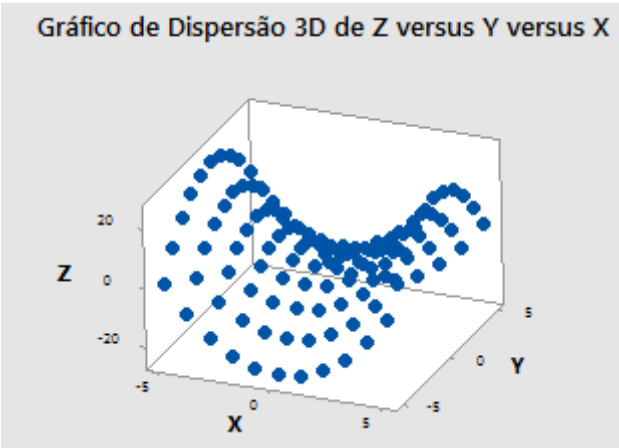
(a) Gráfico de Contornos



(b)Gráfico de Superfície 3D



(c) Gráfico de dispersão 3D

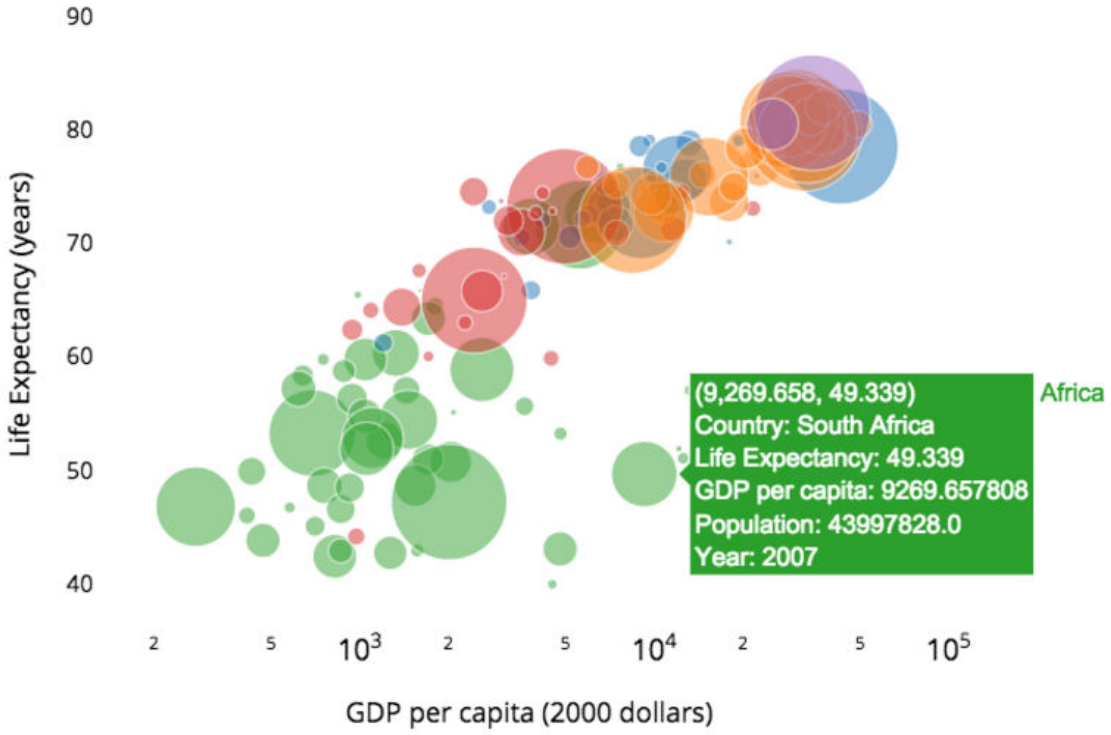


Fonte: “Suporte ao Minitab17”

Os diagramas anteriores representam bem a distribuição de valores de uma variável ou a relação entre duas ou três variáveis (geralmente uma ou duas variáveis de entrada e uma saída). Entretanto, os sistemas em processos químicos são multivariáveis e com correlação entre as variáveis. Sendo assim, é desejável visualizar o comportamento de muitas delas ao mesmo tempo, de forma a enxergar padrões de comportamento e facilitar a análise do processo. Apesar disso, o ramo industrial ainda utiliza pouco as representações multivariadas, atentando-se apenas a gráficos de dispersão, histogramas, *boxplot* e etc.. Gráficos mais elaborados, que existem para suprir essa necessidadee que são largamente utilizados em outras áreas (como a área financeira), ainda são pouco explorados em processos químicos industriais.

Um exemplo é o gráfico de bolhas (*BubblePlot*), aplicado neste trabalho. Seu uso é muito variado, pois nele pode-se visualizar, ao contrário do gráfico de dispersão convencional, até cinco dimensões de uma só vez, que são: posição no eixo horizontal, posição no eixo vertical, tamanho da bolha, cor da bolha e tempo. A Figura 12mostra um gráfico de bolhas contendo quatro dimensões. Elas são: expectativa de vida (posição no eixo vertical), PIB per capita (posição no eixo horizontal), população (tamanho da bolha) e continente (cor da bolha):

Figura 12: Gráfico de bolhas



Fonte: “The power of Bubble Charts”

Nesse diagrama, vários padrões de comportamento podem ser identificados: o crescimento razoavelmente linear da expectativa de vida em relação ao PIB per capita, a maioria de países africanos com baixo PIB per capita e baixa expectativa de vida, e a presença de países populosos em praticamente todas as regiões do gráfico (indicando que esse provavelmente não seja um fator determinante na expectativa de vida ou no PIB per capita). Esse gráfico ainda destaca pontos fora do padrão, como a pequena bolha

vermelha em meio a um emaranhado de bolhas verdes, algo extremamente útil em um monitoramento de processos ou em uma análise da raiz de uma falha.

Esses pontos podem ser dinâmicos (animados com o tempo, como um *GIF*) ou estáticos (bolhas fixas que não se movem). Ao adicionar o tempo como a quinta dimensão, tornando o gráficouma animação, pode-se, por exemplo, analisar a evolução dessas variáveis em cada país ao longo do século XX. Fatos históricos que interferem nessas variáveis, como guerras, pestes, acordos comerciais, avanços tecnológicos, etc., podem ser identificados e ajudar na análise das causas que levaram cada nação à situação em que se encontra hoje. Da mesma forma, em um processo químico, variáveis correlacionadas podem ser visualizadas em conjunto e padrões/falhas podem ser mais facilmente identificados e analisados.

3.3 Monitoramento de processo

O foco em análise de dados neste trabalho é a área de monitoramento de processos, onde se situa a ferramenta de qualidade denominada Controle Estatístico de Processos (CEP). Essa ferramenta, aplicada à produção, permite a redução sistemática da variabilidade em características chave da qualidade, contribuindo para a melhoria da qualidade, da produtividade, da confiabilidade e do custo de produção(RIBEIRO, CATEN, 2012)**.** Cada produto possui uma série de características que definem a sua qualidade, como densidade, comprimento, cor, gosto, voltagem ou durabilidade.Um produto ou serviço de qualidade é aquele que atende às especificações, atingindo o valor alvo com a menor variabilidade possível.O CEP é um sistema de inspeção por amostragem que identifica sua variabilidade e permite a verificação, análise e bloqueio de causas não naturais que afetem a qualidade do produto. É uma ferramenta importante para a melhoria contínua dos processos, pois possibilita que o próprio operador controle a qualidade em tempo real. São monitoradas as características de interesse, garantindo que elas se mantenham dentro dos limites estabelecidos e indicando rapidamente o momento de se tomar ações corretivas. Dessa forma, são reduzidos custos com refugo e retrabalho,

aumentando a capacidade do processo e qualidade dos produtos ou serviços. Nesse contexto, o CEP pode ser definido como(MONTGOMERY, 2009)**:**

> “Um método preventivo de se comparar continuamente os resultados de um processo com um padrão, identificando, a partir de dados estatísticos, as tendências para variações significativas, eliminando ou controlando estas variações com o objetivo de reduzi-las cada vez mais.”

A variabilidade é um fator inerente aos processos produtivos e sempre está presente, em maior ou menor intensidade. Diferentes fontes de variabilidade agem de forma distintas sobre o processo, podendo gerar desde alterações imperceptíveis até diferenças grandes no processo. Portanto, é importante identificar e diferenciar as fontes de variabilidade, que podem ser classificadas como causas comuns ou causas especiais.

As causas comuns são as fontes de variação que atuam de forma aleatória, gerando uma variabilidade inerente ao processo, sendo, portanto, um padrão natural e ocorrendo sob condições normais de operação. Um processo é dito estável ou sob controle estatístico quando apresenta apenas causas comuns, possuindo a mesma variabilidade ao longo do tempo, chamada de variabilidade natural ou “ruído de fundo”. Por outro lado, as causas especiais, ou causas atribuíveis, são causas que não seguem um padrão aleatório e fazem com que o processo saia de seu padrão natural de operação, sendo consideradas falhas de operação. Os seus resultados são claros e podem ser causados, por exemplo, por máquinas ajustadas de maneira inadequada, erros do operador ou matéria prima defeituosa. Essas causas afetam as características de qualidade e reduzem o desempenho do processo, por isso devem ser identificadas e neutralizadas, geralmente pelos próprios operadores. O objetivo do controle estatístico de processos é detectar rapidamente a presença de causas especiais, de modo que ações corretivas possam ser tomadas antes da produção de um grande volume de unidades não conformes MONTGOMERY(2009).

3.3.1 Carta de controleunivariada de Shewhart

Com o objetivo de distinguir entre causas comuns e causas especiais e auxiliar na tomada de decisões, surgiram as cartas de controle, desenvolvidas pelo Dr. Walter Shewhart no início do século XX nos Estados Unidos. O primeiro passo para a implementação de uma carta de controle é a coleta de dados da característica a ser analisada, realizada em uma certa frequência e com um determinado tamanho de amostra, compatíveis com a natureza da variável estudada e com as principais causas de variabilidade. Definem-se, então, a média, o desvio padrão e os limites de controle, associados às causas comuns de variabilidade. A partir dessas informações, constrói-se a carta de controle e, na sequência, os dados da característica analisada são coletados continuamente e plotados nessa carta. Em um processo estável, é esperado que os pontos plotados permaneçam dentro dos limites de controle. Em um processo instável, onde causas especiais estejam atuando, espera-se a ocorrência de pontos fora dos limites de controle ou a existência de padrões não aleatórios na distribuição desses pontos entre esses limites. Além do monitoramento do processo, as cartas de controle podem ser usadas para estimar os parâmetros de um processo de produção e para determinar a sua capacidade, com o foco na melhoria e redução da variabilidade no processoMONTGOMERY(2009).

A Figura 13mostra uma carta de controle típica, na qual é representada uma característica de qualidade medida ou calculada a partir de uma amostra. A linha central (LC) representa o valor médio da característica, correspondente ao estado sob controle. As linhas superior e inferior representam, respectivamente, o limite superior de controle (LSC) e o limite inferior de controle (LIC). Esses gráficos são, em geral, chamados de gráficos de controle de Shewhart, cujos limites são obtidos segundo as **Erro! Fonte de referência não encontrada.**,**Erro! Fonte de referência não encontrada.** e **Erro! Fonte de referência não encontrada.** (MONTGOMERY, 2009):

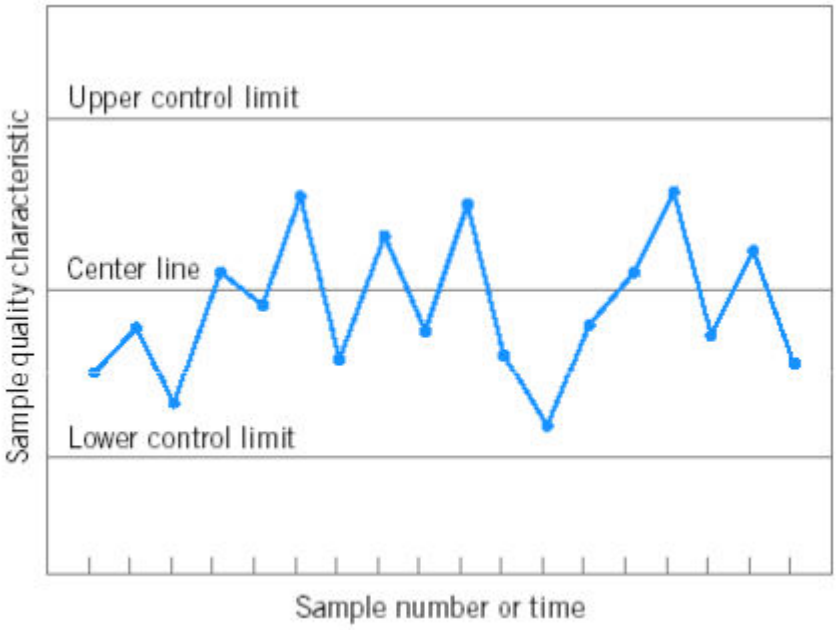
$$LSC = \mu + L\sigma \qquad \text{(EQ. 1)}$$

$$LC = \mu \qquad \text{(EQ. 2)}$$

$$LIC = \mu - L\sigma \qquad \text{(EQ. 3)}$$

onde μ representa a média da característica, σ, o seu desvio padrão, e L a “distância” dos limites de controle à linha central, expressa em unidades de desvio padrão (MONTGOMERY, 2009).

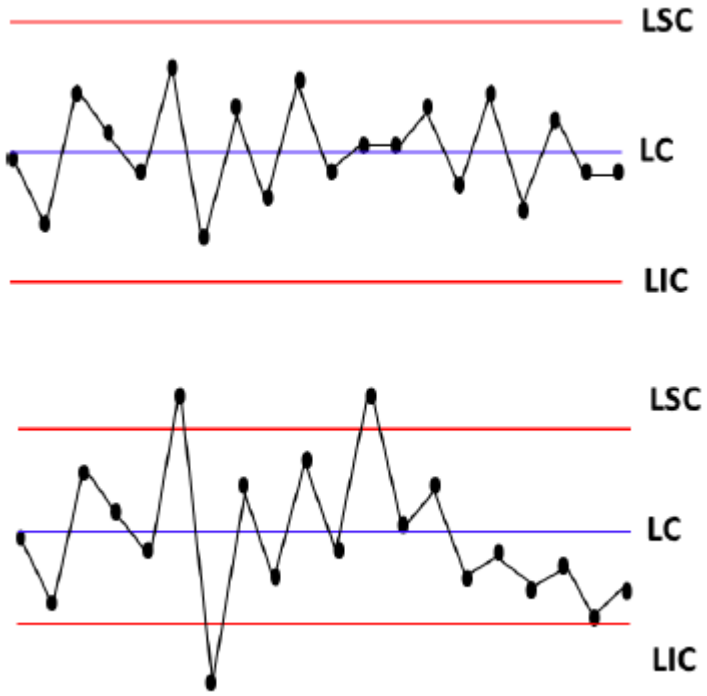
Figura 13 : Carta de controle típica



Fonte:MONTGOMERY, 2009.

Em um processo sob controle, quase todos os pontos amostrais devem estar entre esses dois limites e distribuídos de forma aleatória (Figura 14a). Quando pontos fora dos limites começam a aparecer ou quando a distribuição dos pontos apresentar um comportamento sistemático ou não aleatório, isso é uma indicação de que o processo está fora de controle estatístico e sob a influência de causas especiais e, portanto, ações corretivas devem ser tomadas (Figura 14b).

Figura 14. (a) Exemplo de processo sob controle estatístico e (b) Fora de Controle Estatístico



Fonte: RIBEIRO, CATEN, 2012

Gráficos de controle podem ser classificados em dois tipos: gráficos de controle para variáveis, que são a abordagem desse trabalho, e gráficos de controle para atributos. Os gráficos de controle para variáveis são usados quando a característica da qualidade pode ser expressa em uma escala contínua de medida e descrevem essa característica em termos de uma tendência central e de uma medida de variabilidade. Para controle da tendência central, o gráfico geralmente usado é o da média amostral $\bar{x}$ e, para o controle da variabilidade, são usados os gráficos baseados na amplitude amostral ou no desvio padrão amostral. Os gráficos de controle para atributos são usados quando as características não podem ser medidas em uma escala contínua ou nem mesmo em uma escala quantitativa e, nesses casos, cada unidade de produto é classificada como conforme ou não conforme, com base em certos parâmetros, e os números de não conformidades podem ser quantificados MONTGOMERY(2009).

Segundo Montgomery(2009), os gráficos de controle são populares e amplamente usados nas indústrias, pois:

1. São uma técnica comprovada para a melhoria da produtividade, pois reduzem a sucata e o retrabalho;
2. São eficazes na prevenção de defeitos e, portanto, mais eficientes e baratos do que a simples inspeção do produto final;
3. Evitam o ajuste desnecessário do processo, pois podem distinguir entre um ruído de fundo e uma variação anormal com eficiência satisfatória.
4. Fornecem informação de diagnóstico, permitindo a implantação de uma mudança no processo que melhore o seu desempenho;
5. Fornecem noções sobre a capacidade do processo, fornecendo assim direções importantes para os planejadores do produto e do processo.

3.3.2 Controle estatístico multivariado de processos

Em sistemas reais, e mais especificamente em processos químicos industriais, mais de uma característica de qualidade do produto ou variável de saída é analisada para garantir resultados de projetos. Dessa forma, embora aplicar gráficos de controle univariados para cada uma das variáveis seja uma abordagem possível, isso é ineficiente e pode levar a conclusões erradas, devido à correlação entre elas. Sendo assim, métodos multivariados são necessários para problemas dessa natureza (MONTGOMERY, 2009).

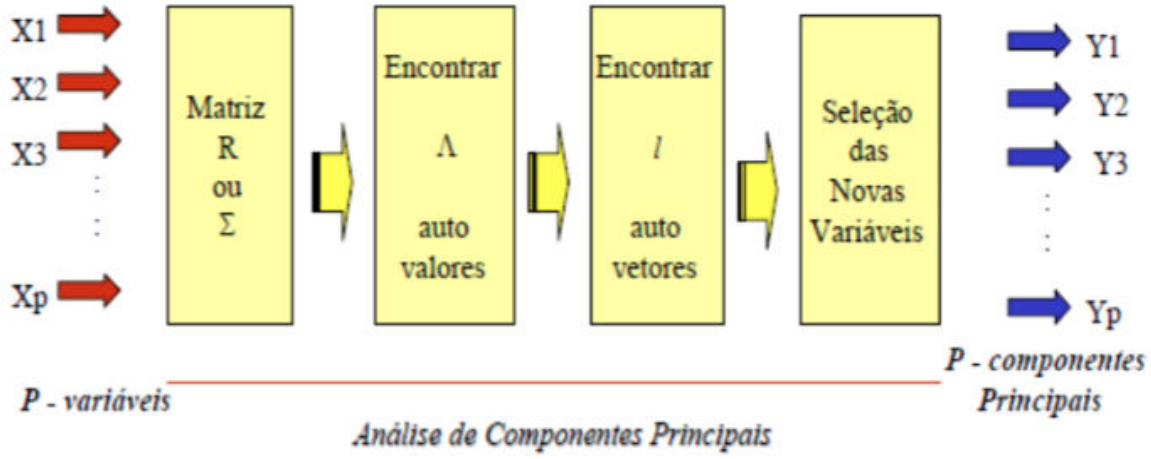
O gráfico T^2 de Hotelling e o gráfico de controle EWMA (*ExponentiallyWeightedMovingAverage*) são alternativas multivariáveis que funcionam bem quando o número de variáveis é menor ou igual a 10(MONTGOMERY, 2009). Para um número maior de variáveis, outros métodos devem ser empregados, como aquele denominado Análise dos Componentes Principais (PCA). Descreve-se a seguir a técnica PCA utilizada neste trabalho com o propósito de detecção de falhas.

3.3.3 Análise de componentes principais (PCA)

A Análise de Componentes Principais, ou *Principal ComponentAnalysis* (PCA), é uma técnica estatística multivariada que transforma um conjunto de variáveis correlacionadas em um outro conjunto (de mesmo tamanho) de variáveis independentes, denominadas variáveis latentes ou componentes principais, que são combinações lineares das variáveis originais.

Assim, com a utilização do PCA é possível resumir a informação proveniente de muitas variáveis correlacionadasem um grupo menor de variáveis independentes.É também conhecida como Transformada Discreta de Karhunen-Loève ou Transformada Hotteling.Para a determinação das componentes principais, é preciso calcularа matriz de variância-covariância (Σ) ou a matriz de correlação (R) a partir da matriz de dados **X** (onde **X** tem p variáveis originais), e então encontrar os autovalores (Λ) e os autovetores, denominados de componentes principais (Y ou CP, projeção da matriz **X** no sistema coordenado definido pelos autovetores), conformeo esquemada Figura 15:

Figura 15. Esquema de aplicação da Análise de Componentes Principais



Fonte: SOUZA, 2000.

Outras informações relativas à técnica PCA estão descritas na seção de metodologia.

4 ESTUDO DE CASO

O estudo de caso desse trabalho é o *Benchmark* Tennessee (DOWNS; VOGEL, 1993). O processo químico industrial real, que teve suas características originais alteradas pelos pesquisadores para preservar os direitos dos proprietários, é baseado no conjunto de reações químicas a seguir (equações 4 a 7). Tais reações são irreversíveis, exotérmicas e têm cinéticas descritas pela Lei de Arrhenius de primeira ordem de acordo com as concentrações dos reagentes.O fluxograma do processo pode ser observado naFigura 16.

$$A_{(g)} + C_{(g)} + D_{(g)} \rightarrow G_{(l)} - Produto\ 1 \qquad (4)$$

$$A_{(g)} + C_{(g)} + E_{(g)} \rightarrow H_{(l)} - Produto\ 2 \qquad (5)$$

$$A_{(g)} + E_{(g)} \rightarrow F_{(l)} - Subproduto \qquad (6)$$

$$3D_{(g)} \rightarrow 2F_{(l)} - Subproduto \qquad (7)$$

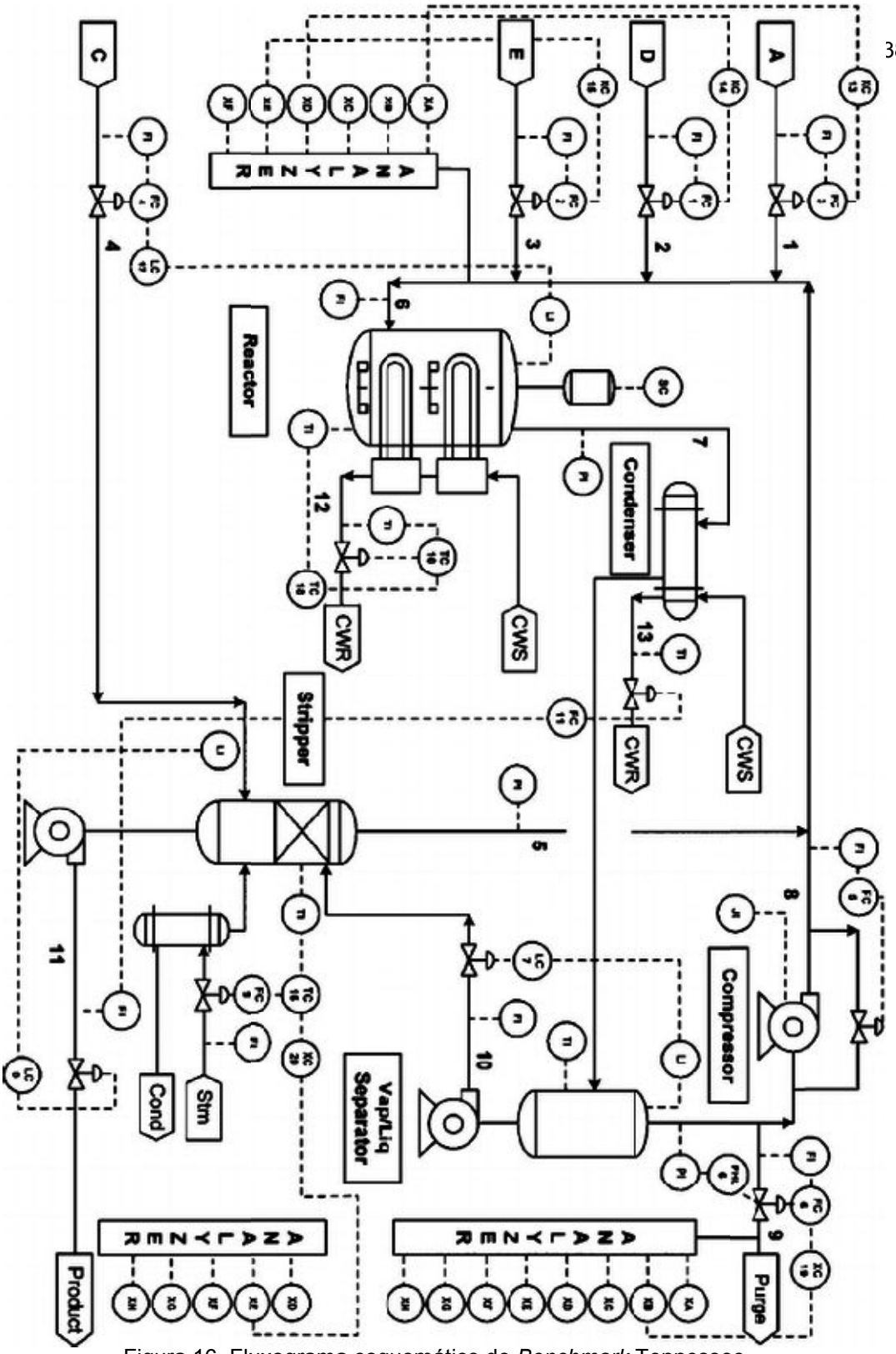


Figura 16. Fluxograma esquemático do *Benchmark* Tennessee

Fonte:DOWNS; VOGEL, 1993

Originalmente, o projeto possui cinco operações unitárias principais: o reator, o condensador, um separador vapor-líquido, um compressor de reciclo e um stripper. No processo, os reagentes gasosos são alimentados no reator,

formando produtos líquidos. As reações são catalisadas por um catalisador sólido dissolvido na fase líquida do reator e se mantém no reator durante todo o tempo. O reator é resfriado internamente por um sistema de resfriamento interno. Em seguida, os produtos são condensados e o separador vapor-líquido separa as fases, seguida de um stripper para retirar qualquer resquício de reagente nas fases líquidas. Os produtos H e G são separados e refinados em um processo não incluído no benchmark. Existem seis modos diferentes de produção, de acordo com as proporções de G e H obtidas no processo, conforme a Tabela 2.Assim, a produção é determinada a partir da demanda do mercado e cada modo é utilizado para atender cada demanda.

Tabela 2: Modos de operação do *Benchmark* Tennessee

MODO	RAZÃO MÁSSICA G/H	TAXA DE PRODUÇÃO (11)
1 (CASO BASE)	50/50	7038 KG/H G E 7038 KG/H H
2	10/90	1408 KG/H G E 12669 KG/H H
3	90/10	10000 KG/H G E 1111 KG/H H
4	50/50	PRODUÇÃO MÁXIMA
5	10/90	PRODUÇÃO MÁXIMA
6	90/10	PRODUÇÃO MÁXIMA

O estudo tem cinco objetivos principais:

1. Manter as variáveis desejadas dentro de intervalos pré-determinados.
2. Operar de acordo com as especificações dos equipamentos.
3. Minimizar a variabilidade dos produtos, garantindo sua qualidade.
4. Minimizar a movimentação das válvulas que afetam outros processos.
5. Recuperar facilmente de quaisquer falhas na produção.

As variáveis controladas e seus limites de processo foram estabelecidos para garantir a segurança de funcionamento da indústria, de acordo com as especificações de cada unidade de processo. A qualidade do processo é controlada pela variação na composição da linha 11 (Figura 16), sendo que

uma variação de 5% na composição dessa linha em um intervalo de 8 a 16 horas já representa um risco para o processo. Igualmente danoso, é uma variação de 5% molar de G num intervalo de 6 a 10 horas. Além disso, as composições das alimentações também são controladas, com o objetivo de minimizar possíveis variações no processo(ver Tabela 3).

Tabela 3. Variáveis e limites de operação

Variável	Nº variável	Valor (caso base)	Limite inferior	Limite superior	Unidades
Alimentação D (2)	XMV 1	3664	0	5811	kg/h
Alimentação E (3)	XMV 2	4509	0	8354	kg/h
Alimentação A (1)	XMV 3	0,273	0	1,107	kscm.h
Alimentação A e C (4)	XMV 4	9,35	0	15,25	kscm.h
Válvula de reciclo compressor	XMV 5	22	0	100	%
Válvula purga (9)	XMV 6	40	0	100	%
Vazão líquido separador (10)	XMV 7	25,04	0	65,71	m³/h
Produção líquidoStripper (11)	XMV8	22,85	0	49,10	m³/h
Válvula de vapor Stripper	XMV 9	47	0	100	%
Vazão água de resfriamento reator	XMV 10	93,4	0	227,1	m³/h
Vazão água de resfriamento condesador	XMV 11	49,4	0	272,6	m³/h
Velocidade do agitador	XMV 12	125	150	250	rpm

Todas as 21 falhas disponíveis no *Benchmark*(Tabela 4) foram utilizadas nesse trabalho.

Tabela 4: Falhas disponíveis no *Benchmark* Tennessee

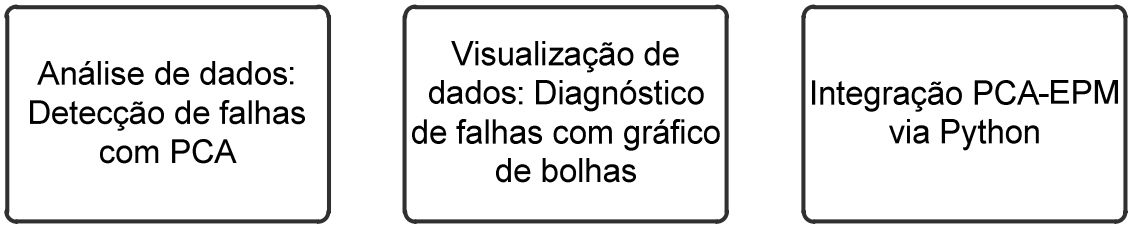
Variável	Descrição	Tipo
IDV (1)	Razão de alimentação A/C, B composição constante	Degrau
IDV (2)	Composição B, razão A/C constante	Degrau
IDV (3)	Temperatura de alimentação de D (linha 2)	Degrau
IDV (4)	Temperatura água de resfriamento entrada reator	Degrau
IDV (5)	Temperatura água entrada condensador	Degrau
IDV (6)	Perda de alimentação A (linha 1)	Degrau
IDV (7)	Queda de pressão de C – diminuição de disponibilidade	Degrau
IDV (8)	Composições alimentação A, B e C (linha 4)	Variação aleatória
IDV (9)	Temperatura alimentação D (linha 2)	Variação aleatória
IDV (10)	Temperatura alimentação C (linha 4)	Variação aleatória
IDV (11)	Temperatura água de resfriamento entrada reator	Variação aleatória
IDV (12)	Temperatura água entrada condensador	Variação aleatória
IDV (13)	Cinética do reator	Alteração lenta
IDV (14)	Válvula da água de resfriamento	Emperramento
IDV (15)	Válvula da água do condensador	Emperramento
IDV (16)	Desconhecido	
IDV (17)	Desconhecido	
IDV (18)	Desconhecido	
IDV (19)	Desconhecido	
IDV (20)	Desconhecido	
IDV (21)	Desconhecido	
IDV (22)	Válvula linha 4 fixa em estado estacionário	Posição fixa

Fonte: RUSSELL; CHIANG; BRAATZ(2000)

5 METODOLOGIA

A metodologia usada no presente trabalho pode ser esquematizada de acordo com as três etapas mostradas na Figura 17. Cada uma delas é descrita nos tópicos a seguir.

Figura 17: Metodologia usada



Foi usado neste trabalho um banco de dados em formato de arquivo de texto (.txt) contendo os dados do *Benchmark* Tennessee, referentes aos estados de operação normal e de falha.

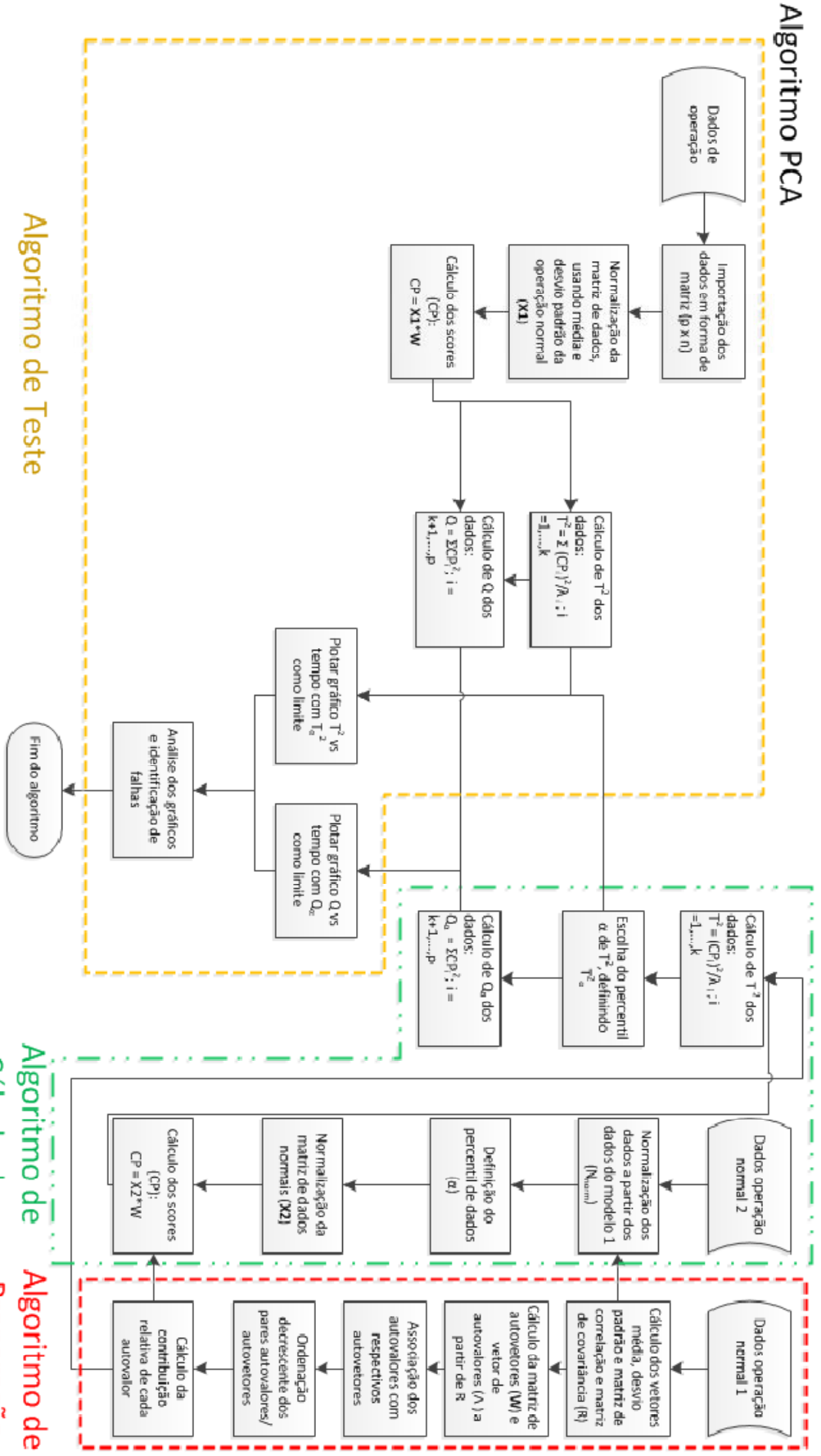
Primeiramente, foi escrito o código do PCA, com etapas envolvendo a construção do modelo e a definição dos limites de controle. A capacidade de detecção do modelo criado foi testada mediante a comparação com os resultados descritos no trabalho desenvolvido por Russel et al. (2000). A visualização de dados objetivando o diagnóstico das falhas foi exemplificada por meio de um gráfico de bolhas.

Um código em linguagem Python foi criado com a função de transformar os dados dos arquivos de texto em variáveis (*Data Objects*) do EPM.O código do modelo PCA foi então adaptado e inserido no Elipse Plant Manager,como um *plug-in,* com a criação de um botão na interface do programa. Dessa forma foi possível, através do EPM, gerar os gráficos de acompanhamento T^2 e Q do modelo PCA, de modo a analisar e identificar as falhas.

5.1 Análise de Dados: Detecção de Falhas com PCA

O desenvolvimento dessa etapa é mostrado naFigura 18**,** que compreende três partes: construção do modelo PCA, definição dos limites de controle, e detecção de falhas propriamente ditas.

Figura 18 : Algoritmo PCA



5.1.1 Construção do modelo PCA

Inicialmente, a partir dos dados de uma operação normal (ON1, matriz **X**), que contém p variáveis originais, são calculados as médias e os desvios padrão de todas as variáveis do sistema. Esses valores serão usados na normalização de uma segunda operação normal (ON2), na etapa de Definição dos Limites de Controle.Então, é calculada a matriz de covariância dos dados a partir de ON1, **R**:

$$\boldsymbol{R} = E[(\boldsymbol{X} - E[\boldsymbol{X}]) * (\boldsymbol{X} - E[\boldsymbol{X}])^T], \qquad (8)$$

sendo E[**X**] o vetor de valores esperados de **X**. Essa etapa garante que variáveis de ordens de grandeza diferentes possam ser normalizadas, garantindo que os valores absolutos das observações não sejam influentes.A seguir, são calculados a matriz de autovetores (**W**) e o vetor de autovalores (Λ) de**X**, a partir de **R**. Os autovalores são associados aos seus respectivos autovetores, criando um par autovalor/autovetor. Dessa forma, cada autovalor pode explicar uma porção da informação total de **X**, de acordo com sua contribuição individual, calculada conforme a **Erro! Fonte de referência não encontrada.**:

$$\frac{\lambda_i}{\lambda_1+\lambda_2+\cdots+\lambda_p} \quad (9)$$

Assim, é possível identificar quais variáveis latentes (componentes principais) são responsáveis por explicar a maior parte da variância total do sistema. Dessa forma, os pares autovalores/autovetores são ordenados em sequência decrescente de acordo com sua contribuição individual, ou seja, aqueles que possuem maior razão são os primeiros e os últimos são aqueles que possuem menor razão. O número de componentes principais (k) a reter no modelo PCA é, de modo geral, definido a partir desse cálculo de razões.

Então, um modelo PCA é definido pelo vetor de autovalores (Λ), pela matriz de autovetores (**W**) e pelo número de componentes principais a reter (k).

5.1.2 Definição dos Limites de Controle

A técnica PCA, nesse trabalho, é utilizada com o propósito de detecção de falhas. Nessa direção, é necessário definir os limites de controle para as métricas de monitoramento: T^2 e Q, dados porT_α^2 e Q_α, em que α é o nível de significância.

Desse modo, os parâmetros T_α^2 e Q_α atuarão como limites de controle superiores para T^2 e Q, permitindo a identificação de falhas nas operações, caso os valores calculados sejam maiores que tais limites.Para tal, são utilizados os dados de uma segunda operação normal (ON2). De fato, a definição dos limites de controle não utilizará α, sendo calculada diretamente a partir dos dados.

Nessa direção, os dados de ON2 são normalizados através das médias e dos desvios padrão armazenados de ON1, gerando a matriz **X2**. Em seguida, a matriz de*scores* dos dados normais, **CP**, é calculada de acordo com a**Erro! Fonte de referência não encontrada.**10, a partir de **W** obtida anteriormente:

$$\boldsymbol{CP}_{[n\,x\,p]} = \boldsymbol{X2}_{[n\,x\,p]} * \boldsymbol{W}_{[n\,x\,p]} \quad (10)$$

A matriz de componentes principaiscorresponde à matriz original **X2**; porém, no novo sistema de coordenadas, dado por **W**.Visto que a matriz **W** foi ordenada em ordem crescente anteriormente; desse modo, **CP** também estará ordenada, com os maiores valores (*scores*) nos primeiros elementos da matriz. Assim, os primeiros elementos representam valores que descrevem a maior parte da informação de **X2**. Então, calcula-se o valor de T^2, usando a matriz de autovaloresΛ, calculada na etapa anterior, a partir da **Erro! Fonte de referência não encontrada.**:

$$T^2 = \sum_{i=1}^{k} \frac{{CP_i}^2}{\lambda_i} \quad (11)$$

Na sequência, calcula-se o valor de Q, a partir da**Erro! Fonte de referência não encontrada.**.

$$Q \quad = \sum_{i=k+1}^{p} CP_i^2 \qquad (12)$$

Com os valores de T^2, característicos de operação normal (a partir de ON2), o limite de controle (equivalente a T^2_α) é dado pelo percentil 99, uma vez que a taxa de alarmes falsos foi pré fixada em 1%. De modo análogo, calcula-se o limite de controle para Q (equivalente a Q_α).

5.1.3 Detecção de Falhas

Após as etapas anteriores de construção do modelo PCA (seção 5.1.1) e da definição dos limites de controle para as métricas de monitoramento T^2 e Q (seção 5.1.2); passa-se à etapa de detecção de falhas.

Nessa direção, calculam-se, em separado, os valores de T^2 e Q para cada um dos conjuntos de dados de falha (Tabela 4). Na sequência, os valores de T^2 e Q são plotados nas respectivas cartas de controle contendo os limites superiores T^2_α e Q_α, respectivamente. Qualquer ponto acima dos limites, éum sinal de operação com falha; caso contrário, a operação é tida como normal.

5.2 Visualização de dados: Diagnóstico de falhas com o gráfico de bolhas

Constatada a falha na operação, deve-se realizar o seu diagnóstico, isto é, procurar as causas das alterações. Em seguida, elas devem ser sanadas o mais rápido possível e medidas devem ser tomadas para evitar que outra falha semelhante ocorra. Embora o foco deste trabalho seja a detecção de falhas e não o seu diagnóstico, a falha número 6 do *Benchmark Tennessee* foi diagnosticada por meio de um gráfico de bolhas, a título de exemplo.

Para visualizar o comportamento das variáveis no intervalo de tempo em torno da falha, foram gerados gráficos de dispersão contendo o valor absoluto de cada variável em função do tempo. Buscando reduzir o número de gráficos, algumas variáveis foram escolhidas e suas curvas de dispersão agrupadas em

um mesmo diagrama. Essas mesmas variáveis foram também representadas por um gráfico de bolhas.

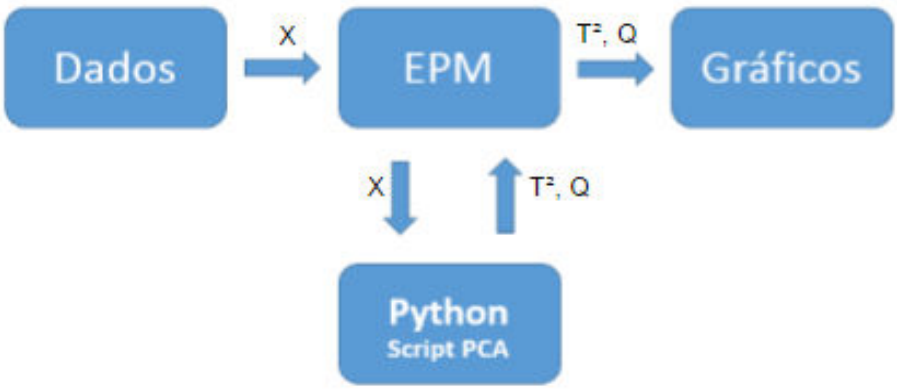
A partir dessas figuras, pôde-se determinar a causa da falha e verificou-se se ela estava de acordo com o motivo descrito pelo artigo deRusselet al (2000). A influência da falha em cada variável também foi analisada. Entretanto, para não tornar o trabalho muito extenso, apenas a análise de algumas variáveis arbitrariamente escolhidas está representada nesse estudo.

5.3 Integração PCA– EPM via Python

Aintegração PCA– EPM via Python foi feita com o objetivo de disponibilizar no EPM os recursos da ferramenta PCA. Para isso, utilizou-se a versão de demonstração do programa, disponível para *download* na página da Elipse Software. Essa versão possui as mesmas funcionalidades da versão completa, no entanto, permite apenas a utilização simultânea de um máximo devinte variáveis (*Data Objects*).

O processo realizado para a exibição dos gráficos das métricas de monitoramento T^2 e Q na interface do EPM seguiu o fluxograma mostrado na Figura 19:

Figura 19: Integração PCA– EPM via Python

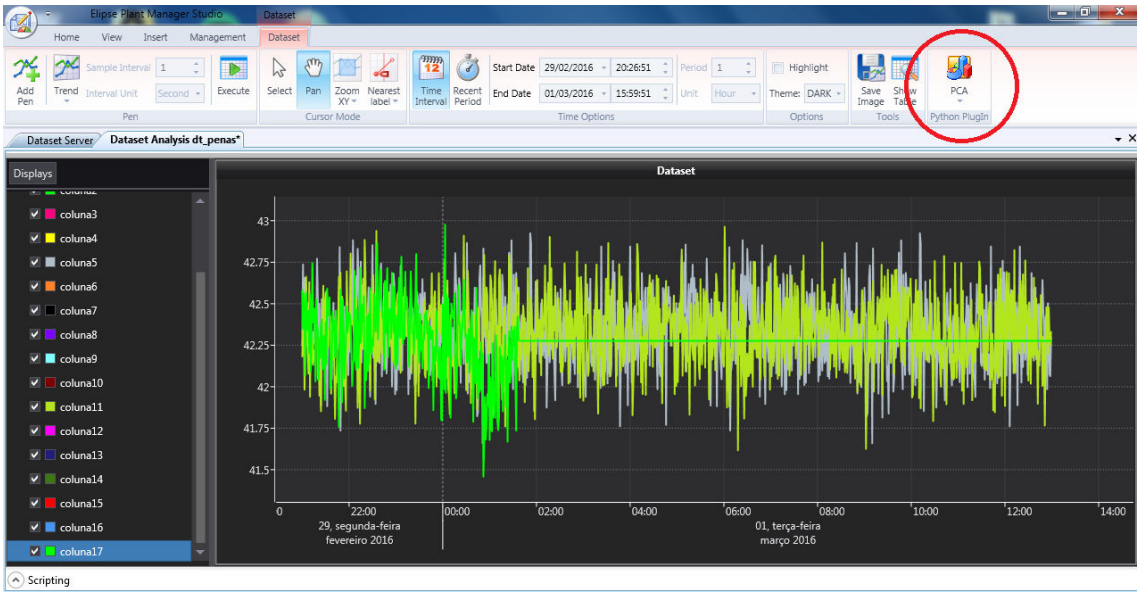


O primeiro passo correspondeu à importação dos dados em .txt, transformando-os em *Data Objects* do EPM. Isso foi realizado por meio de dois

códigos desenvolvidos em Python (códigos em Anexo). Vale ressaltar que o programa não possui uma maneira simples de se importar banco de dados em formato de texto, já que em um caso real isso não seria necessário, pois os dados de processo seriam obtidos diretamente por meio da interface com o sistema integrado de automação da planta.

Em seguida, o modelo PCA desenvolvido em Python foi adaptado de modo que pudesse interagir diretamente com o EPM (código em Anexo).Por meio da criação de um *plugin*, disponibilizou-se a ferramenta PCA, representada por um botão na interface do programa, como mostrado na Figura 20(círculo vermelho no canto superior direito da imagem).

Figura 20:Plugin PCA exibido no EPM



Dessa forma, a execução do PCA nos dados selecionados fornece como resultados gráficos as métricas de monitoramento T^2 e Q, com os respectivos limites de controle$T^{2\alpha}$e Q^{α}. Para exibir esses gráficos no EPM, os valores de T^2 e Q foram salvos em formato de texto e importados para o programa como *Data Objects*, utilizando-se os códigos para importação.

6 RESULTADOS E DISCUSSÃO

A seguir, são apresentados e discutidos os resultados obtidos para cada etapa da metodologia.

6.1 Análise de dados: Detecção de falhas com PCA

Para aplicar o PCA ao *Benchmark Tennessee*, são utilizados três conjuntos de dados. O primeiro deles é usado na construção do modelo PCA característico de operação normal. O segundo conjunto de dados é utilizado para obtenção dos limites de controle. Esses limites são calculados a partir um conjunto de dados de operação normal diferente do anterior. O terceiro diz respeito à detecção das vinte e uma falhas. Nessa etapa, outros momentos de operação contendo falhas são experimentados ao modelo PCA característico de operação normal.

6.1.1 Construção do modelo PCA

Mostra-se, a seguir, parte dos autovalores ordenados resultantes da etapa de construção do modelo PCA (Figura 21).

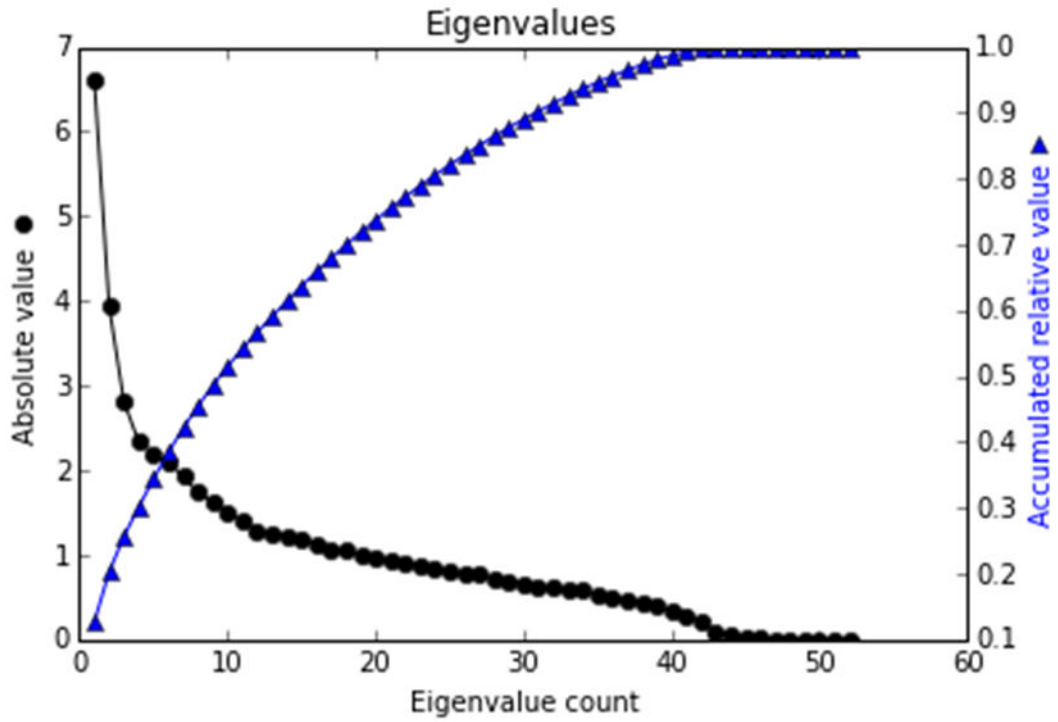
Figura 21: Vetor de autovalores em Python

```
    ...:
Autovalores em ordem decrescente
6.60744438054
3.93323628221
2.80935502895
2.33132860759
2.19472438943
2.08346458069
1.93404944532
1.73451927521
1.62614993673
1.50266333295
1.40347238958
1.28702997802
1.25481302828
1.22652047391
1.19460638633
1.12156102781
1.07244795913
1.05304279706
0.994682042602
0.961477794577
0.936017944402
0.89624362382
0.874467421617
0.829732990943
0.810779539929
0.771762701818
0.764501203609
0.720527305086
```

Ao todo são 52 autovalores, associados a 52 autovetores, um para cada variável original. Não existe uma regra geral para a determinação do número de componentes a serem mantidas e várias alternativas para auxiliar nessa decisão são encontradas.

Uma forma de tomar essa decisão é considerar a variância explicada por cada componente. E por meio da soma das variâncias explicadas das componentes principais, tem-se a variância explicada acumulada, como ilustrado na Figura 22.

Figura 22: Gráfico de autovalores e variância explicada acumulada



A Figura 22 mostra os 52 autovalores e a variância acumulada. Percebe-se que a cada componente o valor da variância explicada passa a contribuir menos para a variância acumulada, sendo, portanto, aquelas que serão descartadas durante o processo.

Desse modo, é possível programar uma rotina capaz de receber como entrada um percentual de variância explicada acumulada requerida. Em outras palavras, criando uma variável chamada critério, pode-se controlar o número k de componentes principais que farão parte do novo modelo até que essa combinação atinja um determinado valor de variância explicada acumulada. Dois modelos foram utilizados para rodar os dados dos experimentos com falhas: um utilizando os k = 27 primeiros autovalores, com 85% de variância total explicada, e o outro utilizando os k = 12 primeiros autovalores, com 55% de variância total explicada. Esse último foi escolhido para comparar com os valores do artigo utilizado com base (RUSSEL et al, 2000), em que o autor utilizou em média esse número de autovalores.

Depois de selecionado o número de componentes, a rotina utilizada nessa parte deve ficar desativada para que esse parâmetro (k) seja fixo ao analisar os dados de falhas.

6.1.2 Definição dos limites de controle

Para a Análise de Componentes Principais são utilizados duas métricas de monitoramento que possibilitam a identificação de comportamentos anormais de operação.

Utilizando k = 12 calcula-se T^2_{α} para T^2 e Q_{α} para Q com o segundo conjunto de dados do sistema sob operação normal. Ao invés de se utilizar α, fixou-se a taxa de alarmes falsos em 1%. Assim, a partir dos valores de T^2e Q, foram calculados os respectivos limites de controle, sendo mostrados na Figura 23 (linhas pontilhadas).

Figura 23. Determinação dos limites de controle com o segundo conjunto de dados de operação normal

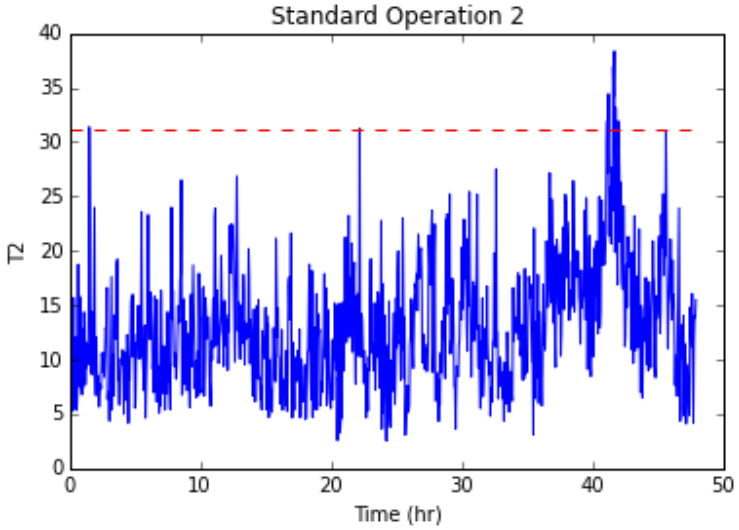
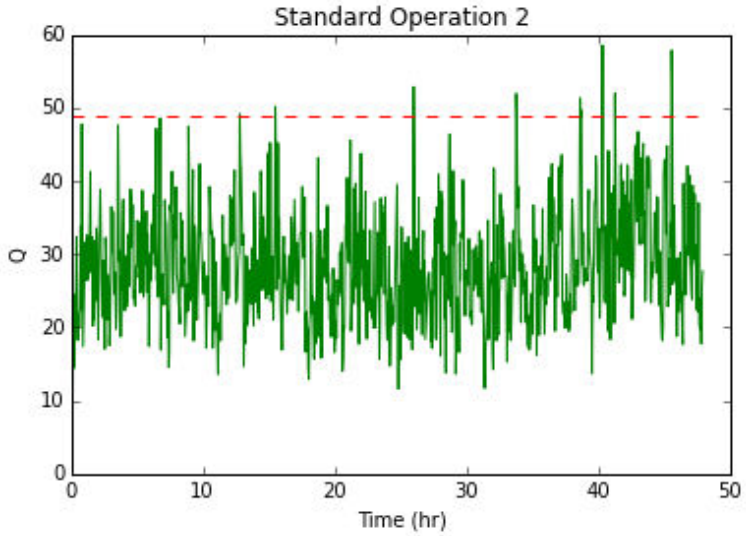


Figure 23 (continuação)



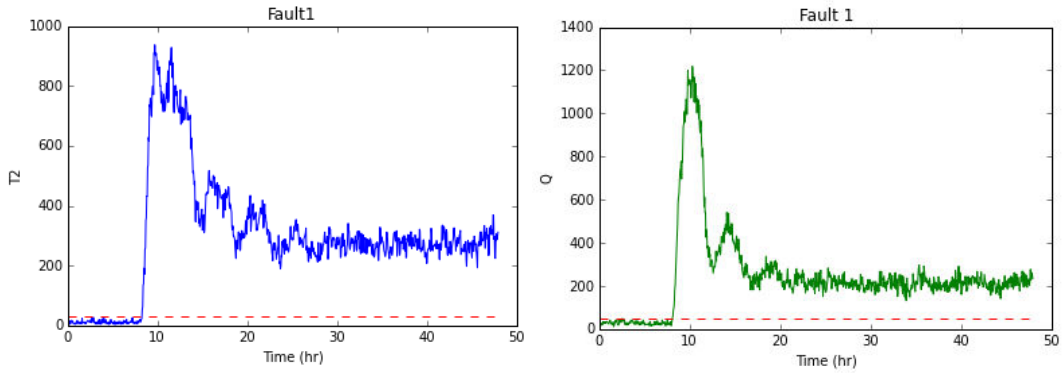
6.1.3 Detecção de falhas

Com o modelo PCA e os limites de controle para T^2 e Q, realizou-se a etapa de detecção de falhas.

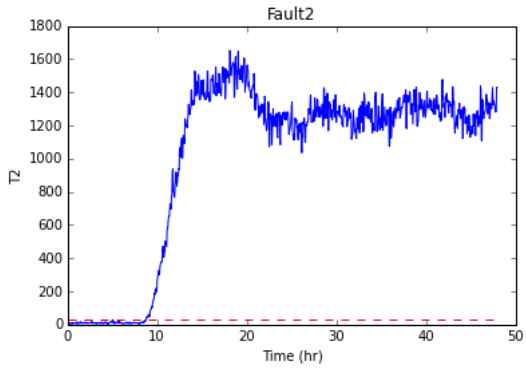
Para cada arquivo de falha, foram plotados os valores de T^2 e Q em função do tempo. A análise do comportamento temporal dessas métricas possibilita a detecção de anormalidades. Houve casos nos quais uma ou as duas métricas de acompanhamento sofreram alterações. Em outros, nenhuma delas foi capaz detectar a falha. A Figura 24mostra as cartas de controle T^2 e Q para as 21 falhas listadas na Tabela 4.

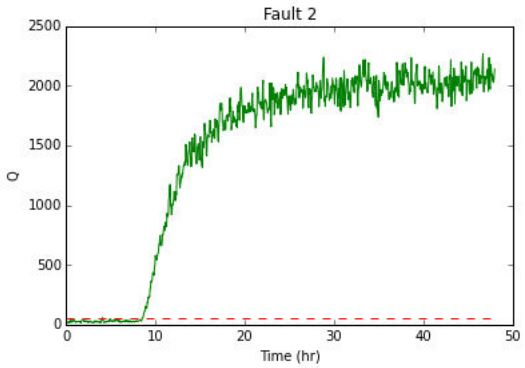
Figura 24. Cartas de controle para as 21 falhas

FALHA 1 (“d01_te.dat”)

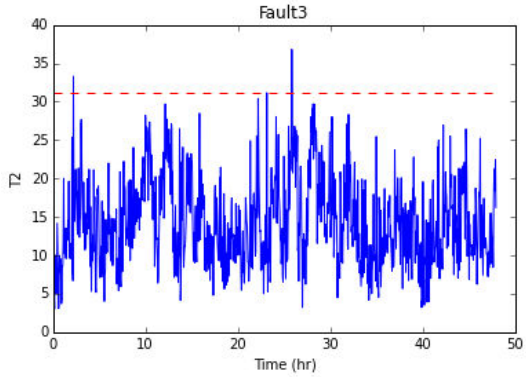


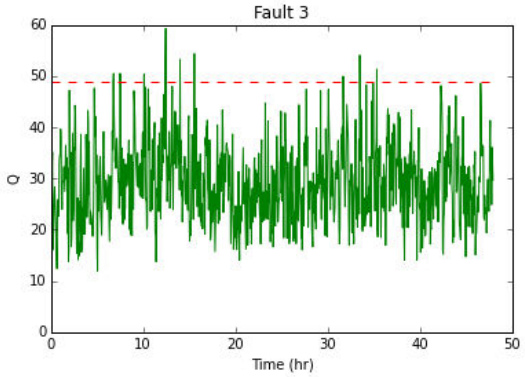
FALHA 2 (“d02_te.dat”)



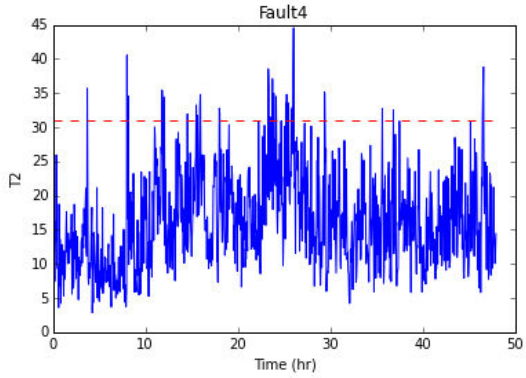


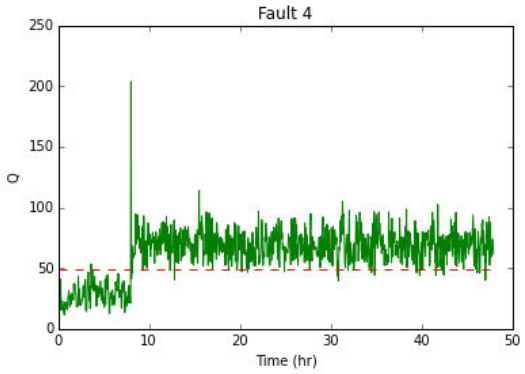
FALHA 3 (“d03_te.dat”)



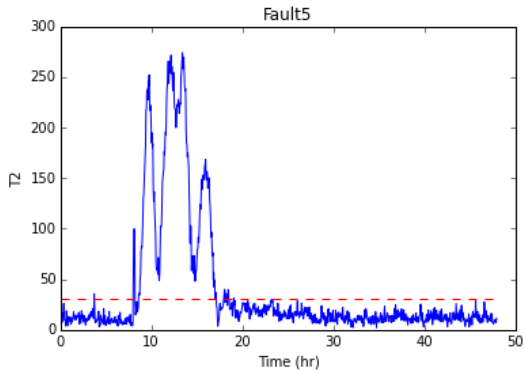


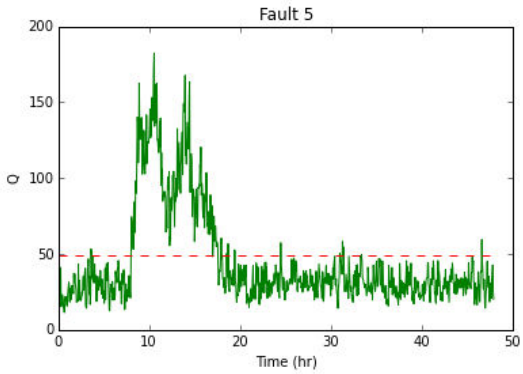
FALHA 4 (“d04_te.dat”)



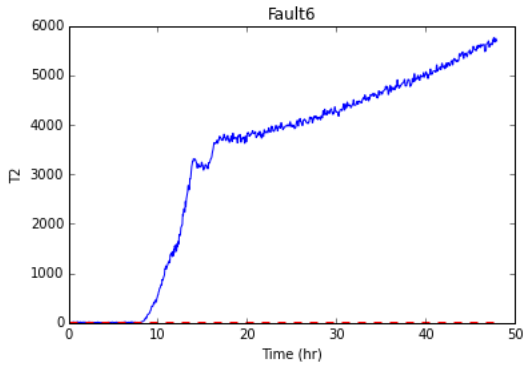


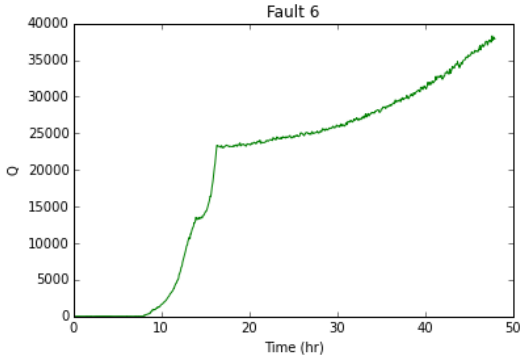
FALHA 5 (“d05_te.dat”)



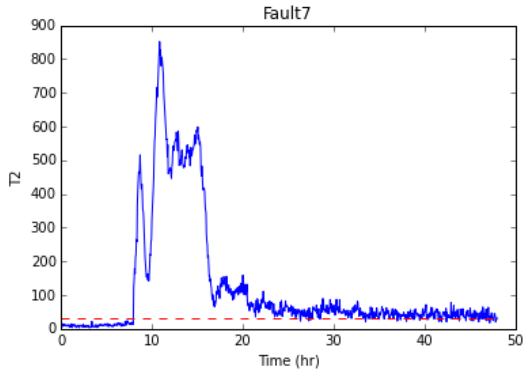


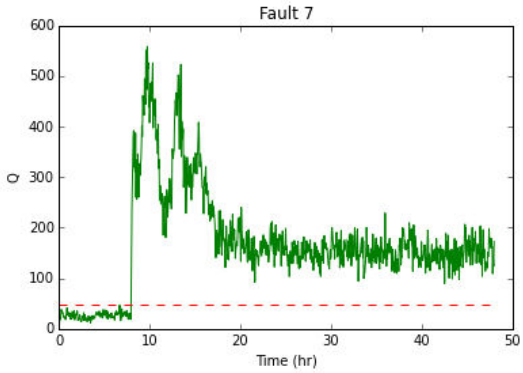
FALHA 6 (“d06_te.dat”)



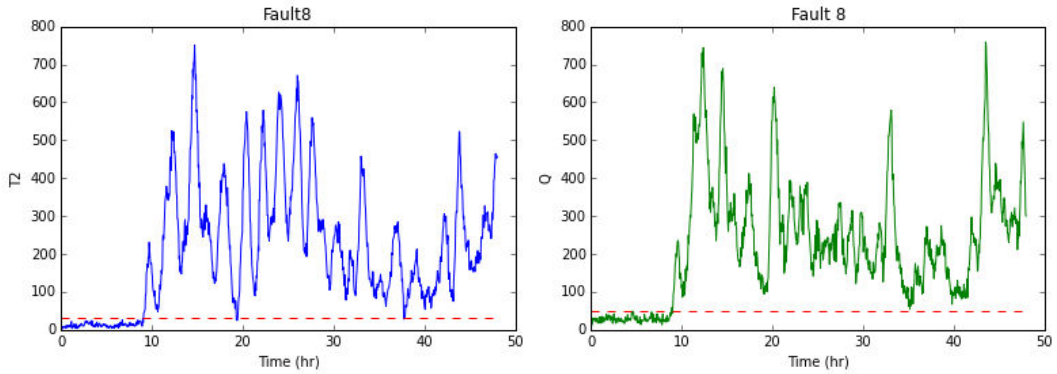


FALHA 7 (“d07_te.dat”)

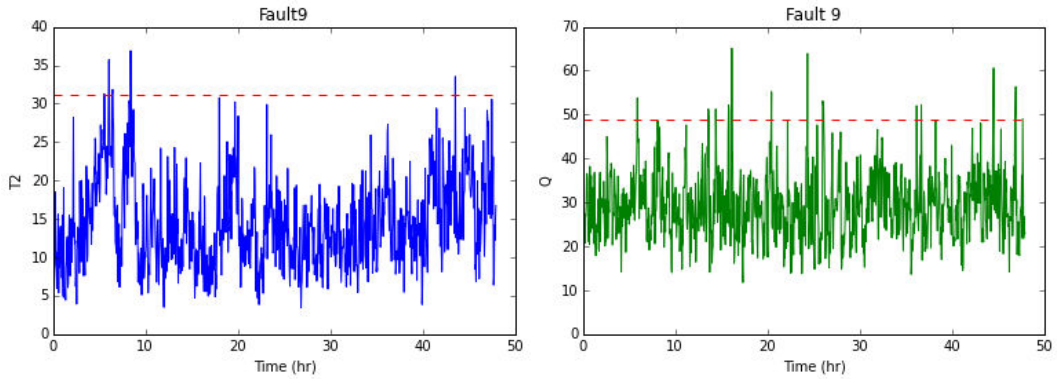




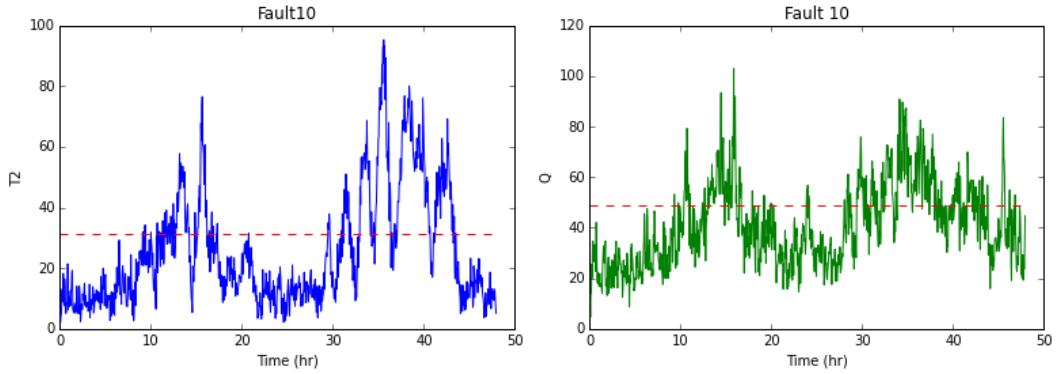
FALHA 8 (“d08_te.dat”)



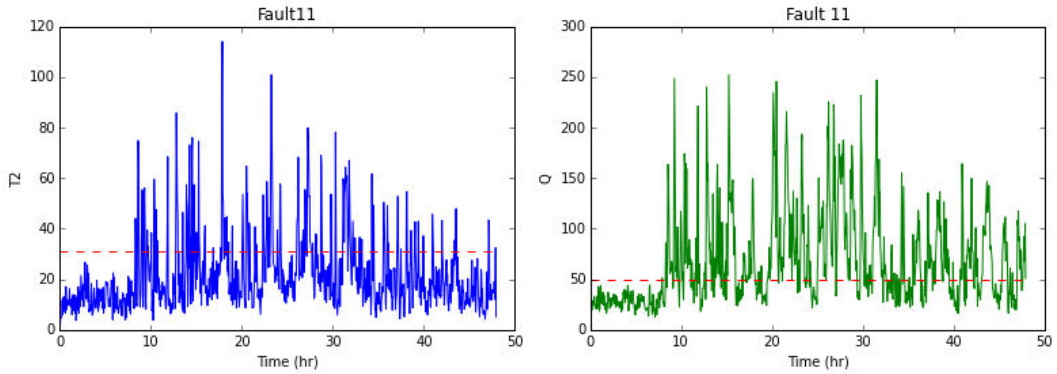
FALHA 9 (“d09_te.dat”)



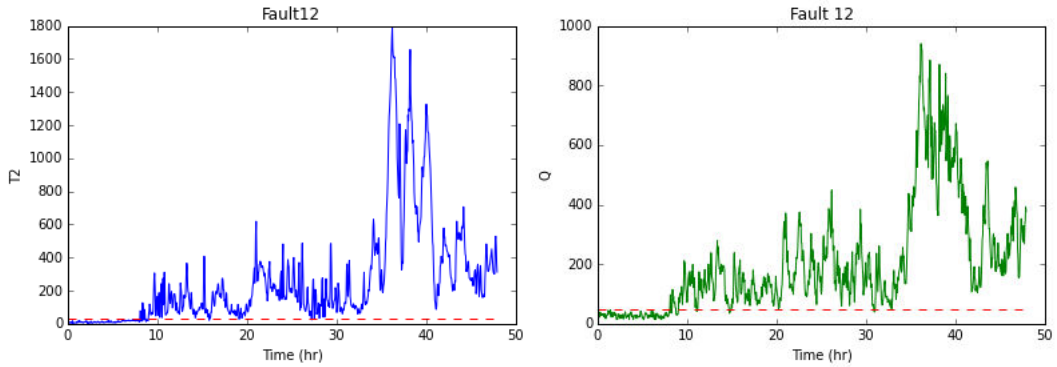
FALHA 10 (“d10_te.dat”)



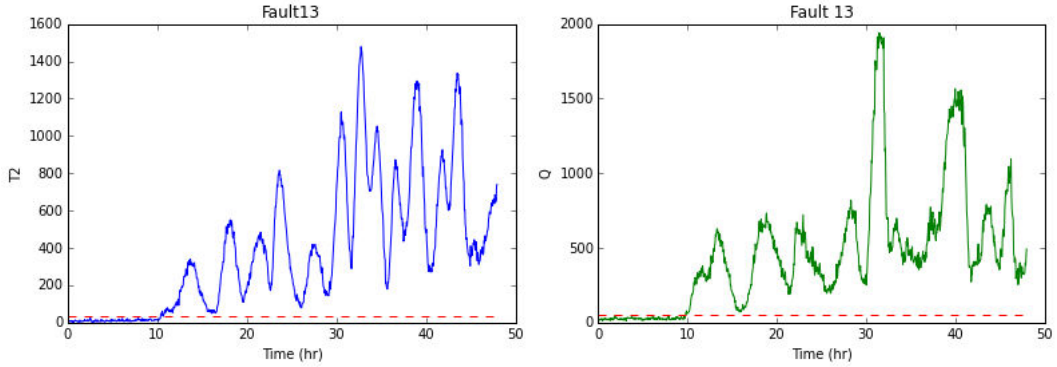
FALHA 11 (“d11_te.dat”)



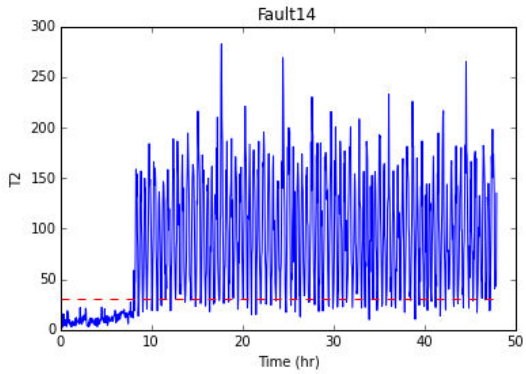
FALHA 12 (“d12_te.dat”)

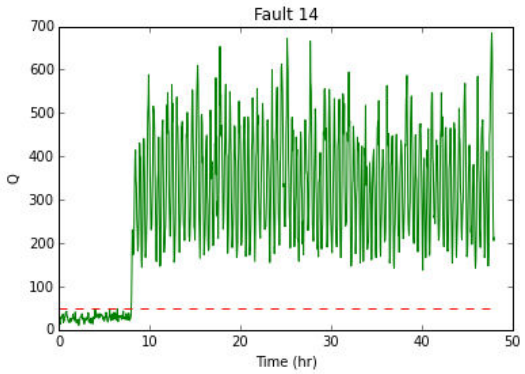


FALHA 13 (“d13_te.dat”)

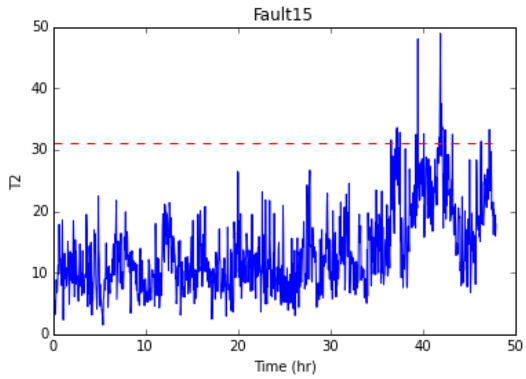


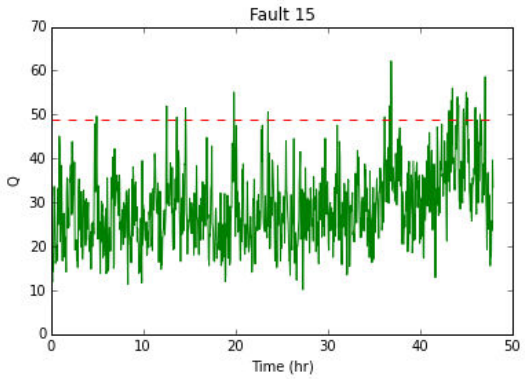
FALHA 14 (“d14_te.dat”)



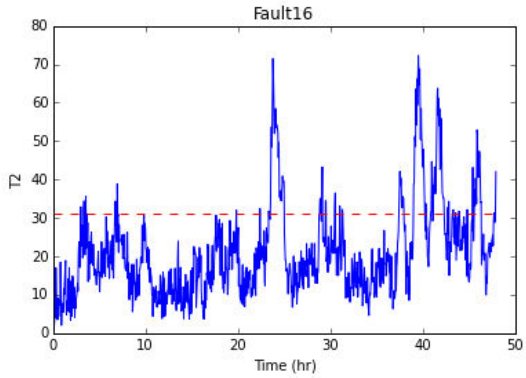


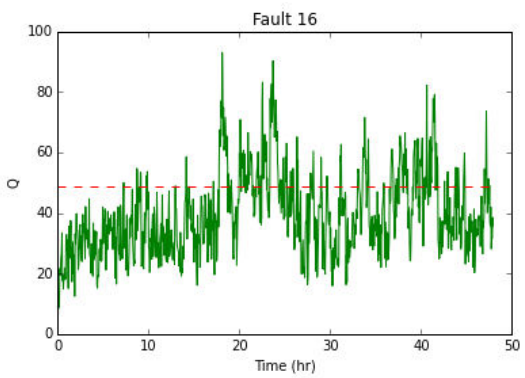
FALHA 15 (“d15_te.dat”)



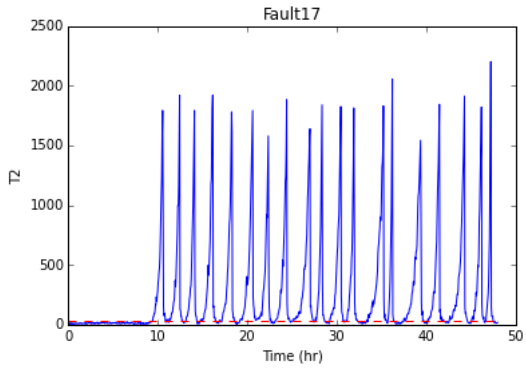


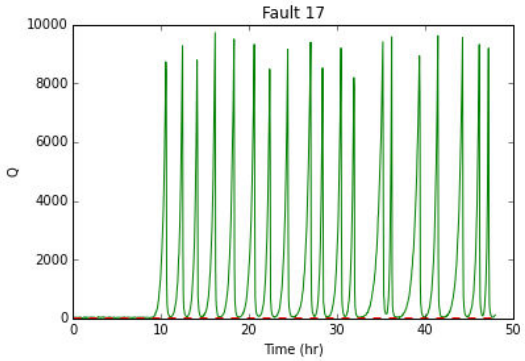
FALHA 16 (“d16_te.dat”)



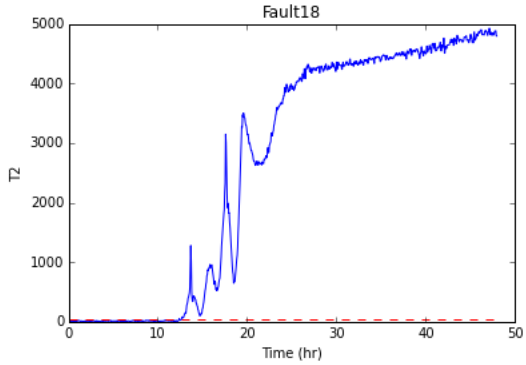


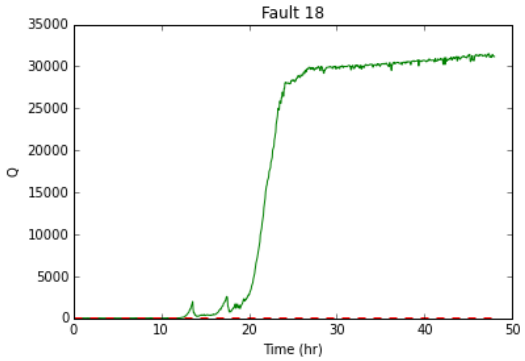
FALHA 17 (“d17_te.dat”)



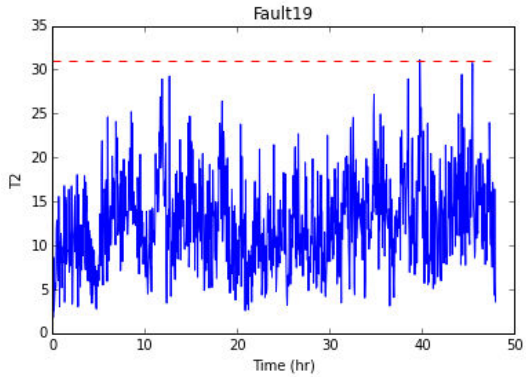


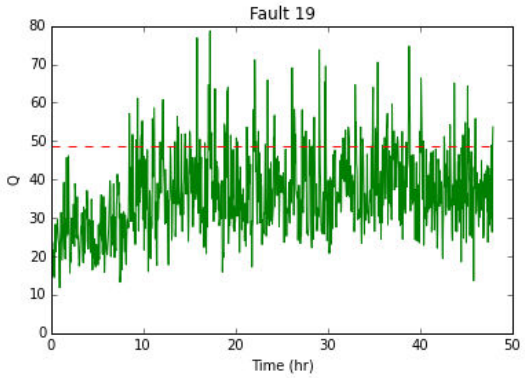
FALHA 18 (“d18_te.dat”)



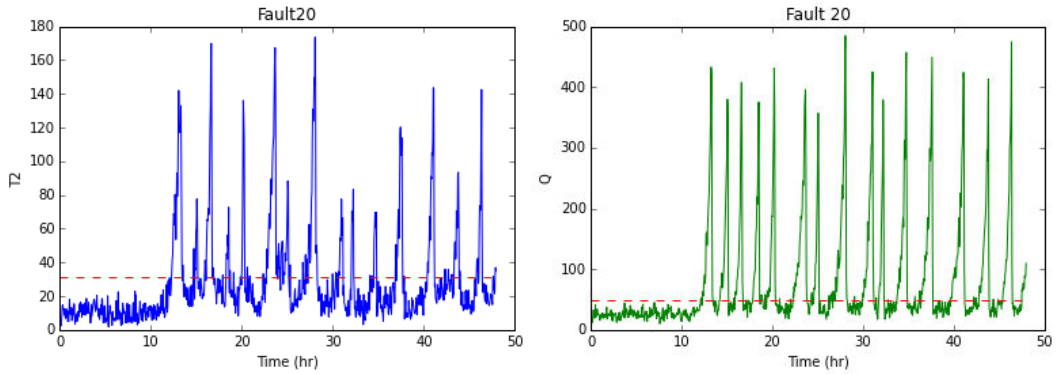


FALHA 19 (“d19_te.dat”)

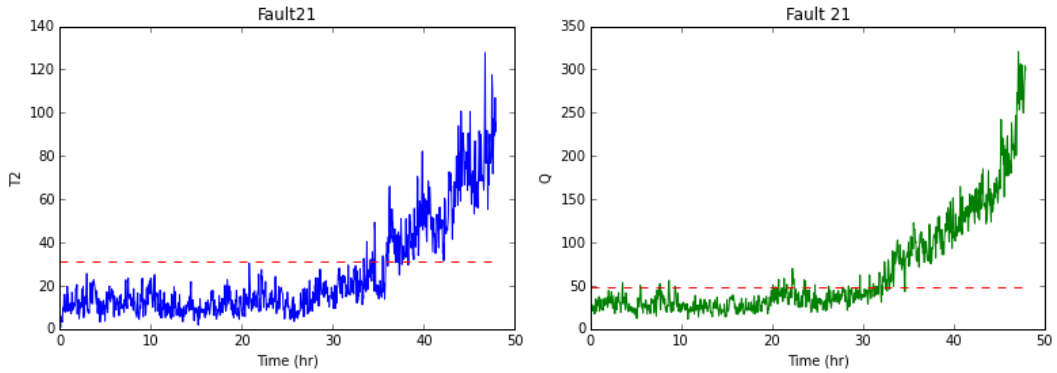




FALHA 20 (“d20_te.dat”)



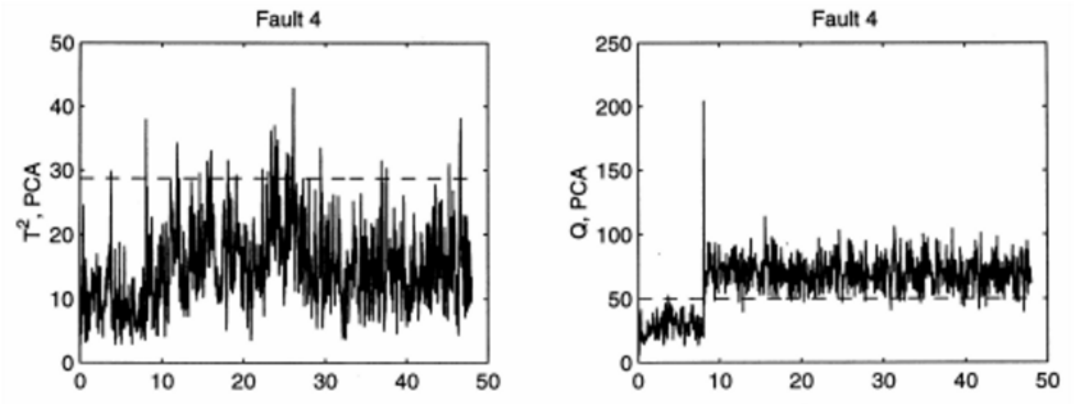
FALHA 21 (“d21_te.dat”)



De modo a validar qualitativamente o modelo PCA implementado na linguagem Python, comparou-se os resultados obtidos para a falha 4, com os gráficos para T^2 e Q disponíveis no artigo deRussel et al(2000), mostradosna Figura 25.

Pode-se perceber a similaridade entre os dois, comprovando o sucesso da programação em Python. Percebe-se que a partir de t=8horas, embora os valores de T^2 continuem normais, os valores de Q têm um salto para acima do limite Q alpha (linha vermelha) e permanecem nesse ponto, indicando a falha. A métrica de monitoramento Q está relacionada com os dados que ficam fora do modelo, isto é, a falha está nas componentes que foram descartadas. E por isso a necessidade de avaliá-las também, e não somente as consideradas pelo modelo.

Figura 25: Gráficos para T^2 e Q na falha 4



Fonte: RUSSELL; CHIANG; BRAATZ, 2000

Nos gráficos em que foi possível visualizar alterações no comportamento normal, essas alterações foram, em resumo:

- Picos acima do limite alpha – Falhas: 5,7,10,11,15,17 e 20
- Mudança na linha de base – Falhas: 7,8 e 19
- Picos desordenados e sem padrão – Falhas: 10,11,12,13 e 14
- Crescimento desordenado – Falhas: 6 e 8
- Degrau – Falhas: 1 e 7

Além da validação qualitativa do modelo, a partir da falha 4, realizou-se também a validação quantitativa, comparando-se os resultados obtidos com aqueles disponíveis emRussel et al(2000). Para tal, foi utilizada a métrica de performance denominada *MissedDetection Rate (MDR)* ou em tradução livre Taxa de Detecção Perdida. Essa métrica mede a quantidade de sinais que o sistema deixou de reportar ao usuário. Dessa forma, como os dados do *Tennessee* são elaborados para que nenhuma falha ocorra antes de oito horas de operação e que, após oito horas, todos os dados sejam de falha, o MDR foi calculado pela razão entre a quantidade de valores menores que os limites (T^2_α e Q_α) e os dados totais, no intervalo entre oito horas e o fim da operação. Dessa forma, obteve-se aTabela 5para cada uma das falhas estudadas. Pode-se observar a aderência entre os valores obtidos e os mostrados por RUSSELL et al (2000).

Tabela 5: Comparação entre MDR's obtidos

Número da Falha	PCA – LOP (k=12)		PCA – LOP (k=27)		PCA – Russel	
	T^2	Q	T^2	Q	T^2	Q
1	0,008	0,003	0,006	0,003	0,008	0,003
2	0,020	0,014	0,018	0,013	0,020	0,014
3	0,999	0,991	0,991	0,985	0,998	0,991
4	0,962	0,035	0,706	0,019	0,956	0,038
5	0,773	0,746	0,756	0,765	0,775	0,746
6	0,011	0	0,010	0	0,011	0
7	0,078	0	0,000	0	0,085	0
8	0,033	0,025	0,028	0,038	0,034	0,024
9	0,999	0,979	0,990	0,976	0,994	0,981
10	0,646	0,640	0,638	0,599	0,666	0,659
11	0,787	0,340	0,524	0,494	0,794	0,356
12	0,021	0,025	0,015	0,040	0,029	0,025
13	0,060	0,045	0,050	0,051	0,060	0,045
14	0,119	0	0,001	0,001	0,158	0
15	0,980	0,971	0,979	0,969	0,988	0,973
16	0,827	0,738	0,817	0,662	0,834	0,755
17	0,253	0,098	0,219	0,068	0,259	0,108
18	0,113	0,100	0,110	0,093	0,113	0,101
19	1,000	0,852	0,957	0,892	0,996	0,873
20	0,702	0,544	0,675	0,489	0,701	0,550
21	0,695	0,574	0,641	0,552	0,736	0,570

Outra métrica de performance é a Taxa de Alarmes Falsos ou *False AlarmDetection.* Essa métrica baseia-se na detecção de sinais quando esses não existem. Assim, como o *Benchmark Tennessee* foi desenvolvido para que os pontos sem falhas sejam encontrados somente entre o início da operação e as oito primeiras horas, foi calculado a quantidade de pontos acima dos limites de controle (T^2_α e Q_α) sobre a quantidade total de pontos nesse intervalo. Os resultados são mostrados na Tabela 6.

Tabela 6: Comparação dos valores de *False Alarme Rate* obtidos

Número da Falha	PCA – LOP (k=12)		PCA – LOP (k=27)	
	T^2	Q	T^2	Q
1	0	0,006	0	0
2	0	0,006	0,006	0,025
3	0,006	0,013	0	0,044
4	0,006	0,006	0,006	0,013
5	0,006	0,006	0,006	0,013
6	0	0	0	0,019
7	0	0	0	0,006
8	0	0,006	0	0
9	0,025	0,006	0,013	0,013
10	0	0	0	0
11	0	0	0	0,006
12	0	0	0	0,019
13	0	0	0	0,006
14	0	0	0,006	0,013
15	0	0,006	0	0006
16	0,050	0,006	0,019	0,013
17	0	0,013	0	0,006
18	0	0,006	0	0,025
19	0	0	0	0,006
20	0	0	0	0,019
21	0	0,005	0,006	0,025
Média	0,004	0,005	0,003	0,013

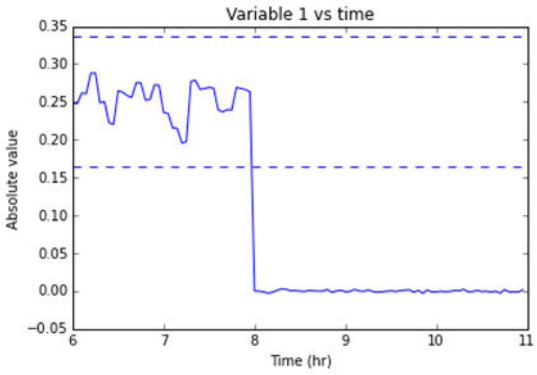
Da mesma forma, os menores valores foram destacados na tabela. Nesse caso, percebe-se que os dois modelos propostos apresentaram grande semelhança de resultados, com diferenças médias de 0,001 e 0,008 para T^2 e Q, respectivamente. Porém, como o modelo com k=27 apresentou um *False Alarm Rate* de cerca de 2,6 vezes maior que o apresentado no modelo k=12, esse apresentou melhor desempenho frente a esse fator.

6.2 Visualização de dados: Diagnóstico de falhas com gráficos de bolhas

Constatada a falha na operação, deve-se realizar seu diagnóstico, isto é, procurar a causa das alterações, saná-la o mais rápido possível e tomar medidas para evitar a repetição do ocorrido.

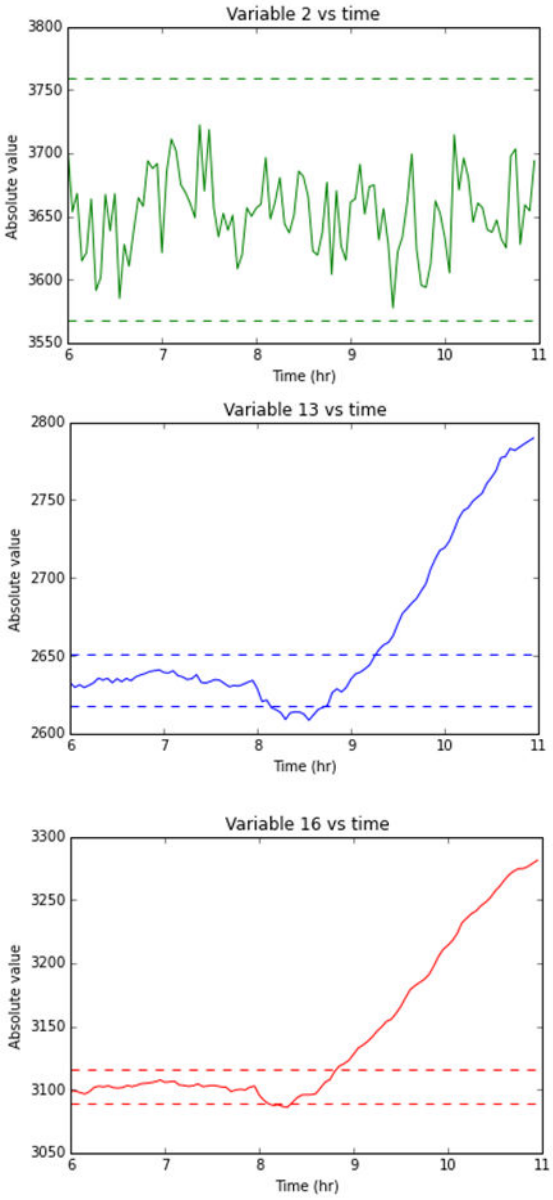
A Figura 26, mostra o gráfico de dispersão da variável 1 (Alimentação de A, Tabela 3. Variáveis e limites de operação página 40) em função do tempo para a falha 6. As linhas tracejadas foram calculadas a partir do modelo e representam a distância de três desvios padrão em relação ao valor médio, como em uma carta de controle univariada de Shewhart. Percebe-se que na primeira variável obteve-se uma alteração no momento esperado da falha (8hrs) em que o valor original caiu rapidamente para próximo de zero. Conclui-se que a variável 1 seja a provável causa da falha. Como a ela é a vazão de alimentação do componente A, provavelmente o reservatório ficou vazio e esse fluxo de entrada cessou. Essa explicação é confirmada pelo trabalho deRUSSELL et al (2000), que afirma que a causa da falha 6 é a parada da alimentação do componente A(Tabela 4).

Figura 26: Variável 1 (falha 6) *vs* tempo



Para visualizar quais variáveis são influenciadas pela queda na alimentação desse componente, foram gerados gráficos de dispersão de cada uma delas em relação ao tempo. Na Figura 27, o valor absoluto das variáveis 2, 13 e 16 foi representado em função do tempo.

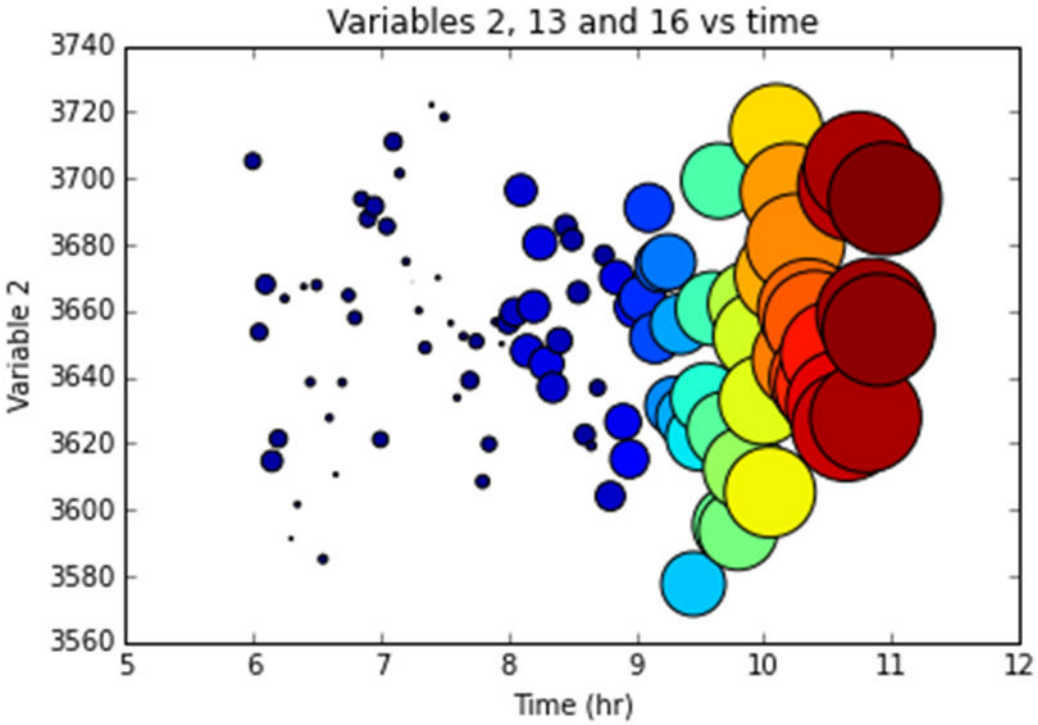
Figura 27: Gráficos separados das variáveis 2,13 e 16 em função do tempo



Pelos gráficos acima, percebe-se que a variável 2 não é afetada pela falha, enquanto as variáveis 13 e 16 são afetadas. Para analisar cada uma das variáveis e a influência sofrida, seriam necessários ainda muitos outros gráficos, o que tornaria dispendioso o trabalho e pouco eficiente. A opção com três ou mais curvas reduziria o número de gráficos. Entretanto, as variáveis analisadas precisam ter a mesma ordem de grandeza e o gráfico perde em clareza.

Buscando melhorar essa representatividade, gerou-se um gráfico de bolhas, mostrado na Figura 28. O tempo é representado no eixo x, a variável 2 no eixo y, a variável 13 como o tamanho da bolha e a variável 16 como cor da bolha.

Figura 28: Gráfico de bolhas para variáveis 2, 13 e 16 em função do tempo



Como a variabilidade das bolhas no eixo vertical permanece a mesma após o tempo de 8 horas, conclui-se que a variável 2 não é sensível à falha 6. Já o tamanho das bolhas é claramente crescente a partir de t = 8 horas, bem como a mudança de cor das bolhas, fatos que mostram que as variáveis 13 e 16 são influenciadas pela falha 6. O gráfico de bolhas ganha em clareza, mesmo representando muitas variáveis juntas.

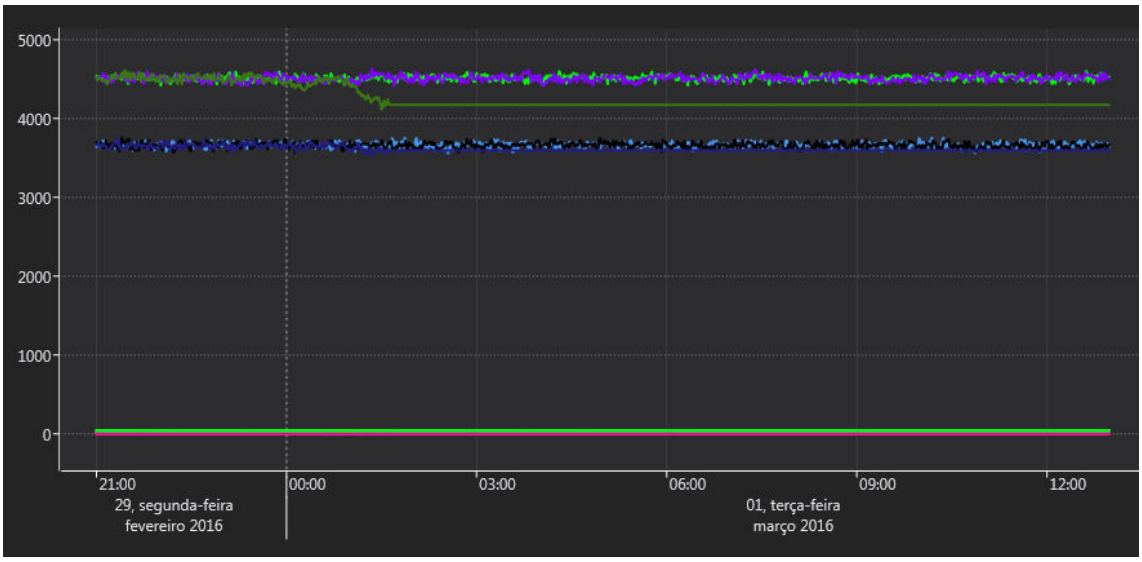
6.3 Integração com EPM: Exemplo de aplicação do PCA

A falha 6 do *Benchmark*Tennesseefoi analisada por meio do EPM. De modo a respeitar a restrição de vinte *Data Objects*, foram usadas apenas seis variáveis das cinqüenta e duas presentes no *Benchmark,*sendo assim*:*

- Carregaram-se doze*Data Objects* com dados das seis variáveis em estado de operação normal, para criação do modelo (cada variável em dois momentos diferentes, para garantia de independência);
- Carregaram-se seis *Data Objects* com dados das seis variáveis em estado de operação fora de controle;
- Os dois *Data Objects*restantes foram usados para armazenar os valores dos parâmetros T^2 e Q, gerados após a aplicação do PCA.

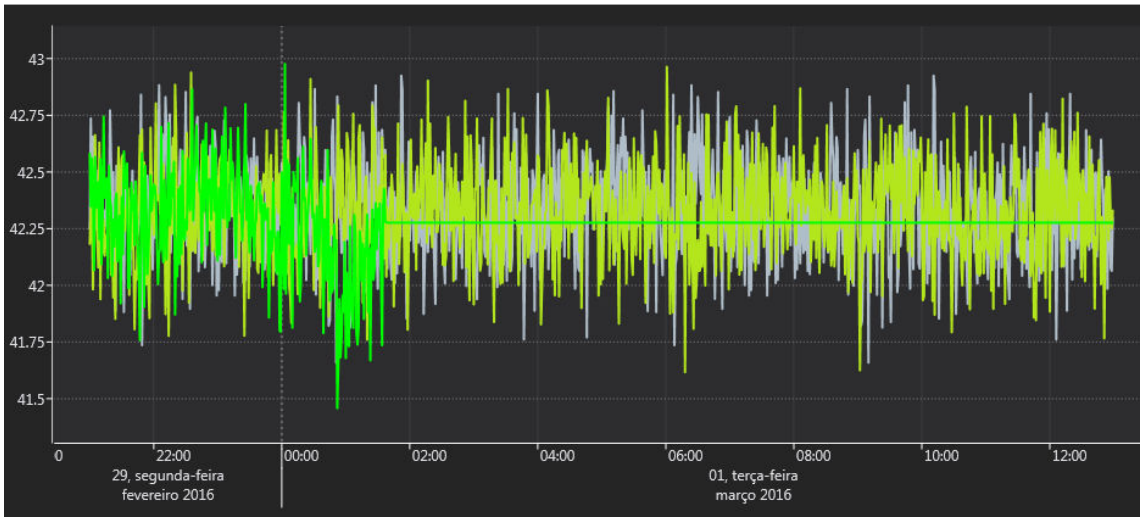
Uma vez importadas, as variáveis foram exibidas na interface do EPM como gráficos temporais, conforme mostrado naFigura 29:

Figura 29:Exibição das variáveis no EPM



A Figura 29 mostra a exibição das seis variáveis nas quais o modelo PCA foi utilizado para detecção de falhas, cada variável representada três vezes, duas no estado de operação normal para a calibração do modelo, e uma no estado de operação fora de controle. A Figura 30 mostra uma ampliação de uma das variáveis:

Figura 30: Ampliação em uma das variáveis utilizadas no modelo PCA



O passo seguinte à importação dos dados foi a aplicação do PCA, que gerou um arquivo .txt com os dados das métricas de monitoramento. Em seguida, o arquivo foi importado para o EPM da mesma forma como foram importadas as variáveis de processo originais. Assim, as métricas de monitoramento foram exibidas na interface gráfica do EPM na forma de gráficos de controle, com limites especificados pelo modelo PCA como mostrado nasFigura 31 eFigura 32.

Figura 31: Gráfico de monitoramento T^2

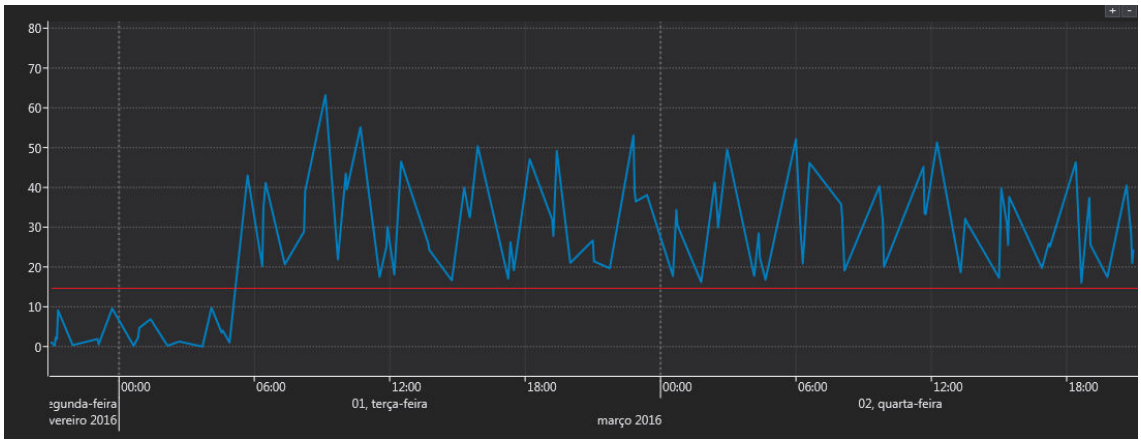
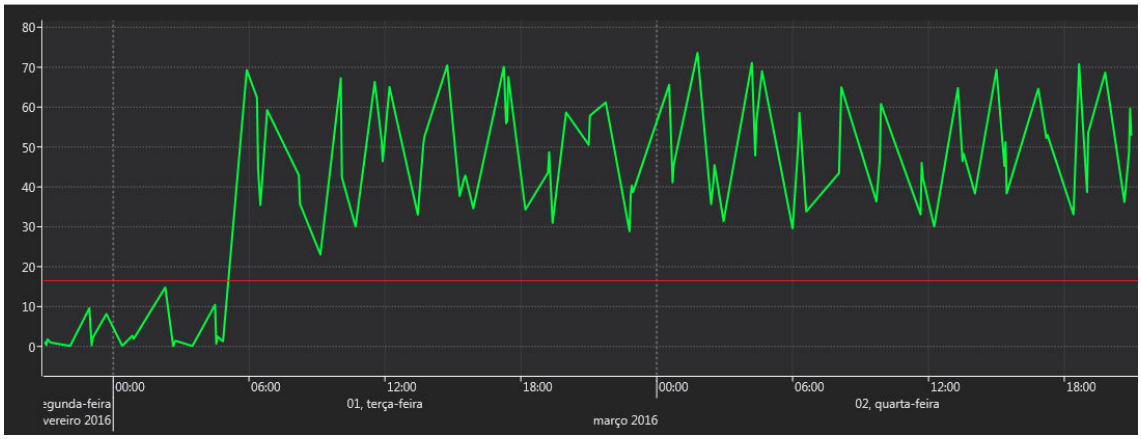


Figura 32: Gráfico de monitoramento Q



Por meio dos gráficos anteriores, e no contexto de seis variáveis originais, foi possível detectar o momento no qual o processo monitorado saiu de controle. Por volta das 6 horas, os valores de T^2 e Q extrapolaram os limites impostos pelas referências T^2_α e Q_α, indicando falha em alguma variável de processo.

Vale ressaltar ainda que o objetivo de mostrar a integração foi cumprido, no entanto, para a aplicação em uma indústria, essa integração deve ser feita de forma mais automatizada e robusta, com auxílio de profissionais da área de computação.

Os gráficos de controle exibidos no EPM ilustram a conveniência do monitoramento de processos com a técnica PCA, uma vez que diversas variáveis podem ser representadas em um único gráfico. Desse modo, agiliza-se a identificação de falhas e facilita-se a tomada de decisões assertivas, em um contexto atual de muitas informações de processo disponíveis, o que dificulta a extração de informações úteis por parte dos Engenheiros.

Mostrou-se, dessa forma, um exemplo de integração entre uma técnica estatística avançada de análise de dados de processo, o PCA, que geralmente não está presente no dia a dia das indústrias, e um software historiador de processos, que por sua vez é comumente usado no meio industrial.

7 CONSIDERAÇÕES FINAIS

A existência de um grande volume de dados de processo na indústria moderna oferece oportunidades e desafios no que se refere à geração de informações necessárias à tomada de decisões estratégicas. Nesse sentido, o presente trabalho abordou técnicas de análise e visualização de dados no contexto da indústria de processos químicos.

Nos esforços de análise de dados, a técnica estatística de qualidade PCA foi implementada em linguagem Python com o objetivo de detectar falhas em um processo químico estabelecido. A referência utilizada para a construção do código implementado foi o *Benchmark Tennessee*. A validação do modelo foi feita mediante comparação qualitativados resultados obtidos para a falha 4 com os gráficos para T^2 e Q disponíveis no trabalho deRUSSELL et al (2000), e comparação quantitativa, a partir da taxa de detecções perdidas.

Nos esforços de visualização de dados, uma sub-rotina do python foi empregada na elaboração de um *Bubble Plot* utilizado para a representação da falha 6 do *Benchmark Tennessee.*

Por fim, uma integração entre uma ferramenta estatística de qualidade e o historiador de processos EPM foi realizada com o intuito de disponibilizar no *software* um recurso não-nativo (PCA). Para tanto, um botão foi construído da interface do EPM na forma de um *plugin*, possibilitando ao usuário a confecção de gráficos em tempo real para monitoramento de processos.

Para trabalhos futuros, sugere-se a melhoria da integração entre a técnica PCA e o EPM de modo a torná-la mais automatizada. Ainda, sugerimos à Elipse Software a disponibilização de documentação das bibliotecas Python necessárias à criação de recursos adicionais.

Destaca-se, ainda, o caráter interdisciplinar do presente trabalho, envolvendo conhecimentos de processos químicos, análise estatística de dados, programação de computadores e noções de TI.

ANEXO

Programação de PCA em Python

A implementação da Análise de Componentes Principais na linguagem Python pode ser feita por várias rotas desde funções inseridas em bibliotecas (funcionando como uma caixa preta ao usuário) até a programação completa de todo algoritmo no cálculo de autovetores e autovalores. Nesse trabalho, desenvolveu-se um meio termo na busca pelo maior controle dos parâmetros da Análise, utilizando algumas funções matemáticas já implementadas no Python no escopo da técnica estatística.

O passo a passo descrito a seguir apresenta os códigos utilizados e alguma breve explicação sobre suas funções.

Importação de dados para o Python

Os bancos de dados utilizados são arquivos do tipo .txt e portanto devem ser importados como uma matriz de dados para o Python. Além disso, o tamanho da matriz é impresso na tela ao fim da importação através da função np.shape. Os dados são armazenados em linhas e colunas sem títulos na variável ‘X’ e podem ser acessados posteriormente. A matriz de dados do Benchmark Tennesse tem 52 linhas, uma linha por variável, e 500 colunas, ou seja, 500 observações por variável.

Figura 33: Importação dos dados na plataforma Python

```
In [15]: import numpy as np
    ...: X = np.genfromtxt("d00.dat") # Matriz de observações
    ...: sz_X = np.shape(X) # Verificando o tamanho da matriz de Dados
    ...: print (X)
    ...: print (sz_X)
    ...:
[[  2.49870000e-01   2.51180000e-01   2.51850000e-01 ...,   2.78800000e-01
    2.50250000e-01   2.49160000e-01]
 [  3.64260000e+03   3.69480000e+03   3.68350000e+03 ...,   3.64470000e+03
    3.72430000e+03   3.72060000e+03]
 [  4.53960000e+03   4.51380000e+03   4.50490000e+03 ...,   4.49690000e+03
    4.50590000e+03   4.48630000e+03]
 ...,
 [  4.74110000e+01   4.74420000e+01   4.75740000e+01 ...,   4.86640000e+01
    4.89660000e+01   4.89770000e+01]
 [  4.10930000e+01   4.13030000e+01   4.15200000e+01 ...,   4.11680000e+01
    4.20480000e+01   4.14520000e+01]
 [  1.83510000e+01   1.98310000e+01   2.04260000e+01 ...,   1.82990000e+01
    1.68250000e+01   1.99990000e+01]]
(52, 500)
```

Normalização dos dados importados

Para trabalhar com matrizes de correlação, os dados precisam estar normalizados. A biblioteca do Python dispõe de funções para normalização de dados e para isso deve-se importar a biblioteca Numpy. Os dados normalizados são armazenados na variável ‘data_std’.

Matriz de correlação e Matriz de covariância

A Análise de Componentes Principais pode ser feita de duas formas: via matriz de correlação e via matriz de covariância.

Matriz de correlação e auto-vetores

O Python conta com uma função para matrizes de correlação de dados sem que haja necessidade de normalização prévia dos dados. A matriz de correlação é armazenada na variável ‘cor_mat’. A função linalg.eig fornece os pares auto-vetores e auto-valores da matriz de dados.

Figura 34: Cálculo da matriz de correlação

```
In [20]: corr_X = np.corrcoef(X) # Função que calcula a matriz de correlações
    ...: egva_corr, egvec_corr = np.linalg.eig(corr_X) # Autovalores e autovetores de
corr_data
    ...:

In [21]: corr_X
Out[21]:
array([[ 1.        , -0.05314299,  0.08730773, ...,  0.0193955 ,
        -0.00822507,  0.03012154],
       [-0.05314299,  1.        , -0.08016995, ..., -0.0048019 ,
         0.26342159,  0.01517199],
       [ 0.08730773, -0.08016995,  1.        , ..., -0.10879532,
         0.00404709,  0.03123536],
       ...,
       [ 0.0193955 , -0.0048019 , -0.10879532, ...,  1.        ,
         0.02818679,  0.00230318],
       [-0.00822507,  0.26342159,  0.00404709, ...,  0.02818679,
         1.        ,  0.06714703],
       [ 0.03012154,  0.01517199,  0.03123536, ...,  0.00230318,
         0.06714703,  1.        ]])
```

Figura 35: Pares de autovetores e autovalores

```
In [22]: egva_corr
Out[22]:
array([  6.60744438e+00,   3.93323628e+00,   2.80935503e+00,
         2.33132861e+00,   2.19472439e+00,   2.08346458e+00,
         1.93404945e+00,   1.73451928e+00,   1.62614994e+00,
         1.50266333e+00,   1.40347239e+00,   1.28702998e+00,
         1.25481303e+00,   1.22652047e+00,   1.19460639e+00,
         1.12156103e+00,   1.07244796e+00,   1.05304280e+00,
         9.94682043e-01,   9.61477795e-01,   9.36017944e-01,
         8.96243624e-01,   8.74467422e-01,   2.13160863e-01,
         8.29732991e-01,   8.10779540e-01,   2.84436618e-01,
         3.36627879e-01,   7.64501204e-01,   7.71762702e-01,
         3.90905573e-01,   7.20527305e-01,   6.97765167e-01,
         4.43360347e-01,   4.51107062e-01,   4.91617086e-01,
         5.22560458e-01,   6.60936436e-01,   6.31249365e-01,
         5.81702307e-01,   5.79897774e-01,   6.10960969e-01,
         7.97975962e-02,   4.62961552e-02,   2.13918453e-02,
         1.09907140e-02,   7.97018198e-03,   3.71081523e-03,
         2.86249749e-03,   7.03350976e-05,   3.77068317e-08,
         4.75793479e-08])

In [23]: egvec_corr
Out[23]:
array([[  2.01469182e-02,  -1.30043381e-01,   3.22721696e-01, ...,
          7.43922099e-03,   3.62311837e-05,  -9.07921897e-08],
       [ -4.33036173e-02,   8.07259136e-02,   2.01432039e-01, ...,
          5.52292021e-04,  -9.71032508e-06,   1.06184984e-05],
       [ -7.85306092e-02,  -3.99376298e-02,  -3.53581219e-03, ...,
         -6.78473277e-05,  -7.80133520e-06,   1.24580024e-06],
       ...,
       [  2.10728355e-01,   3.55727565e-01,   6.51414896e-02, ...,
          2.65868970e-02,  -1.08076942e-04,   7.35250829e-06],
       [ -7.59863195e-02,   1.14740230e-01,   3.10401488e-01, ...,
         -4.93321785e-04,  -6.39916182e-06,   1.33031175e-05],
       [ -6.52529490e-04,   4.66005889e-03,   8.52874689e-02, ...,
         -7.06336799e-01,   5.89148034e-04,  -1.04942028e-03]])
```

Auto-valores ordenados

A seguir, os autovalores são ordenados em ordem decrescente para a escolha das variáveis que descreverão o sistema. Antes disso, deve-se

associar o autovalor ao seu respectivo autovetor, possibilitando a ordenação dos mesmos a seguir. Dessa forma, excluem-se os menores autovetores, visto que eles fornecem menos informações sobre o sistema. Uma possibilidade para ordenação é a listagem dos autovalores na ordem crescente e depois a inversão dessa lista para que o primeiro auto-valor da lista seja o maior. Ao ordenar os autovalores, o programa ordenará na verdade os pares autovalores/auto-vetores.

Figura 36: Ordenação dos autovalores em ordem decrescente

```
In [28]: eigpairs = [(np.abs(egva_corr[i]), egvec_corr[:,i]) for i in
range(len(egva_corr))] # Associando autovalores e autovetores
    ...: eigpairs.sort() # Ordenando autovalores por ordem crescente
    ...: eigpairs.reverse() # Invertendo a ordem (passando para decrescente)
    ...:

In [29]: print('Autovalores em ordem decrescente')
    ...: for i in eigpairs:
    ...:     print(i[0]) #Imprimindo os autovalores em Ordem Decrescente
    ...:
Autovalores em ordem decrescente
6.60744438054
3.93323628221
2.80935502895
2.33132860759
2.19472438943
2.08346458069
1.93404944532
1.73451927521
1.62614993673
1.50266333295
1.40347238958
1.28702997802
1.25481302828
1.22652047391
1.19460638633
1.12156102781
1.07244795913
1.05304279706
0.994682042602
0.961477794577
0.936017944402
0.89624362382
0.874467421617
0.829732990943
0.810779539929
0.771762701818
0.764501203609
0.720527305086
```

Ao todo são 52 autovalores, associados à 52 autovetores, um para cada variável.

Variância explicada

Nessa etapa, cara autovalor é dividido pela soma de todos os autovalores, resultando no percentual que o primeiro autovalor descreve todo o

sistema, chamado de variância explicada. Na Figura 40, a variância explicada é gravada na variável “egva_exp”.

Figura 37: Cálculo da variância explicada

```
In [32]: egva_tot = sum(egva_corr) # Somando os autovalores
    ...: # Contribuição relativa de cada autovalor
    ...: egva_exp = [(i/egva_tot)*100 for i in sorted(egva_corr, reverse=True)]
    ...: egva_sum = np.cumsum(egva_exp) # Soma acumulada das contribuições relativas
    ...:

In [33]: egva_exp
Out[33]:
[12.706623808723647,
 7.5639159273264385,
 5.4026058249094993,
 4.4833242453677702,
 4.2206238258193576,
 4.0066626551764708,
 3.7193258563930436,
 3.3356139907832447,
 3.1272114167829685,
 2.8897371787423691,
 2.6989853645844271,
 2.4750576500324297,
 2.4131019774581532,
 2.3586932190500089,
 2.2973199737027676,
 2.1568481303985925,
 2.0623999214075228,
 2.0250823020438031,
 1.9128500819276744,
 1.8489957588010772,
 1.8000345084657337,
 1.7235454304230604,
 1.6816681184950524,
 1.5956403671988171,
 1.5591914229404491,
 1.4841590419586095,
 1.4701946223246869,
 1.3856294328575218,
 1.3418560907473394,
 1.2710316084945006,
 1.21394108639052491
```

Com a soma dos autovalores, os primeiros elementos da matriz são aqueles que possuem maior informação sobre o sistema (em relação à explicação da variância total do sistema químico), explicando-o melhor. Dessa forma, com a soma de todos os elementos, o resultado é 100% do sistema explicado.

Figura 38: Soma dos autovalores ordenados

```
In [34]: egva_sum
Out[34]:
array([ 12.70662381,  20.27053974,  25.67314556,  30.15646981,
        34.37709363,  38.38375629,  42.10308214,  45.43869613,
        48.56590755,  51.45564473,  54.15463009,  56.62968774,
        59.04278972,  61.40148294,  63.69880291,  65.85565105,
        67.91805097,  69.94313327,  71.85598335,  73.70497911,
        75.50501362,  77.22855905,  78.91022717,  80.50586753,
        82.06505896,  83.549218  ,  85.01941262,  86.40504205,
        87.74689814,  89.01792975,  90.23187084,  91.40679578,
        92.52545406,  93.64064209,  94.64556605,  95.59098352,
        96.4584971 ,  97.31111316,  98.06285464,  98.71021595,
        99.25720944,  99.66713418,  99.8205911 ,  99.90962216,
        99.95076033,  99.97189632,  99.98722359,  99.99435977,
        99.99986458,  99.99999984,  99.99999993, 100.        ])
```

Assim, determina-se o percentual do sistema que deseja-se explicar (reduzindo as dimensões do sistema). No caso descrito, o critério de explicação escolhido foi de 85% do sistema, implicando em k = 27 das 52 variáveis totais.

Figura 39: Critério de explicação do sistema

```
In [40]: # Escolhendo o k (número de autovetores representativos do sistema)
    ...: k = 0
    ...: criterio = 0.85 # Porcentagem do sistema explicada pelas novas variáveis
    ...: while(egva_sum[k] < criterio):
    ...:     k = k + 1
    ...: k = k + 1
    ...:

In [41]: k
Out[41]: 27
```

Matriz de pesos completa e a Matriz de Componentes principais

Todos os autovetores ordenados compõem a matriz de pesos W. Cada coluna dessa matriz é um autovetor da matriz de correlação, isto é, associada a uma variável.

A multiplicação da matriz transposta de autovetores ordenados pela matriz de dados normalizada fornece a matriz de componentes principais Y ou matriz de scores. Cada componente principal é chamado de Cp e os autovalores respectivos são ƛ.

Métricas de Monitoramento

Para a Análise de Componentes Principais são utilizados dois parâmetros que possibilitam a identificação de comportamentos anormais de operação (falhas operacionais): T^2 e Q. Assim, são calculados as métricas de monitoramento para uma operação normal ($T^{2\alpha}$ e Q^{α}) e esses valores são comparados com os valores das métricas para os casos de operação em que deseja-se identificar possíveis falhas. Se os valores das operações forem superiores às métricas da operação normal por um intervalo de tempo, detecta-se uma falha.

Figura 40: Cálculo de T2 e Q em Python

```
# Calculando T2
T2 = np.zeros((sz_Xf[1]))
for i in range(sz_Xf[1]):
    for j in range (k):
        T2[i] = T2[i] + (Y[j,i]**2)/egva_corr[j]
# Calculando Q
Q = np.zeros((sz_Xf[1]))
xI = np.dot(Y[0:k,:].T,W_k.T)
xR = Xfn - xI.T
for i in range(sz_Xf[1]):
    for j in range (sz_Xf[0]):
        Q[i] = Q[i] + (xR[j,i]**2)
```

```
# Calculando T2_alpha
T2_0 = sorted(T2_0,reverse=True) # Ordenando T2 decrescentemente
N = alpha*sz_X0[1]/100.0 # Calculando o PERCENTIL (100-alpha) da matriz X0
if N - int(N) > 0.0001:
    N = N + 1
N = int(N)
T2_alpha = T2_0[N]
T2A = np.ones((sz_X0[1],1))*T2_alpha # Criando um vetor para T2_alpha
# Calculando Q_alpha
Q_0 = sorted(Q_0,reverse=True) # Ordenando T2 decrescentemente
Q_alpha = Q_0[N]
QA = np.ones((sz_X0[1],1))*Q_alpha # Criando um vetor para Q_alpha
```

T^2_{α}

O parâmetro $T^{2\alpha}$ é calculado de forma que acima dele fique apenas 1% dos valores de T^2 calculados para a operação normal.

T^2

O parâmetro T^2 é calculado com base nas k primeiras componentes principais da matriz de pesos, isto é, nas componentes que são retidas pelo modelo. Existem vários tipos de alterações no padrão gráfico de T^2, sendo cada uma delas um tipo de falha.

$$T^2 = \sum_{j=1}^{52} (\frac{Cp(j,i)^2}{\lambda j})$$

Q_α

Q_α é calculado de forma semelhante ao T^2 para os dados de Q.

Q

Ao contrário do parâmetro T^2, o parâmetro Q é calculado com base nas k+1 até p componentes, isto é, nas componentes descartadas pelo modelo e sua utilização é semelhante. É importante ressaltar que T^2 e Q são complementares e ambos devem ser analisados para cada operação uma vez que muitas falhas aparecem apenas em um dos parâmetros. O cálculo de Q é apresentado a seguir:

$$X_i = Cp\ (0\ até\ k)^T .\ W(0\ até\ k)^T$$

$$Xr = X\ norm - X$$

$$Q = \sum_{j=1}^{52} Xr(j,i)^2$$

Script do PCA completo

```
   defpca(X,X0,Xf,N_falha = 1,criterio_K = 85,alpha = 1):
       # Inserir criterio_K e alpha em porcentagem
   importnumpy as np

      # MODELO
   X_mean = np.mean(X,axis=1) # Calculando o vetor medio de
dados
   X_std = np.std(X,axis=1)  # Calculando o vetor desvio
padrao de dados
   corr_X = np.corrcoef(X) # Funcao que calcula a matriz de
correlacoes
   sz_X = np.shape(X) # Verificando o tamanho da matriz de
dados
   egva_corr, egvec_corr = np.linalg.eig(corr_X) #
Autovalores e autovetores de corr_data
       # Associando autovalores e autovetores
   eigpairs = [(np.abs(egva_corr[i]), egvec_corr[:,i]) for
i in range(len(egva_corr))]
   eigpairs.sort() # Ordenando autovalores por ordem
crescente
   eigpairs.reverse() # Invertendo a ordem (passando para
decrescente)
   egva_tot = sum(egva_corr) # Somando os autovalores
       # Contribuicao relativa de cada autovalor
   egva_exp = [(i/egva_tot)*100 for i in sorted(egva_corr,
reverse=True)]
   egva_sum = np.cumsum(egva_exp) # Soma acumulada das
contribuicoes relativas
       # Escolhendo o k (numero de autovetores
representativos do sistema)
   k = 0
   while(egva_sum[k] <criterio_K):
   k = k + 1
   k = k + 1
       # Matriz de autovetores completa
   W = np.hstack((eigpairs[i][1].reshape(sz_X[0],1)) for i
in range(sz_X[0]))
   egva_corr = sorted(egva_corr,reverse=True) # Reordenando
os autovalores para corrigir um erro
       # Matriz dos k primeiros autovetores ou loadings
   W_k = np.hstack((eigpairs[i][1].reshape(sz_X[0],1)) for
i in range(k))

   # OPERACAO SEM FALHAS
       X0 = X0.T # Transpondo a matriz Xf para a forma
usual de trabalho
   sz_X0 = np.shape(X0)
   # Normalizando a matriz de observacoes do experimento
com falhas
```

```
        X0n = np.zeros((sz_X0[0],sz_X0[1]))
    for i in range (sz_X0[1]):
    for j in range (sz_X0[0]):
    X0n[j,i] = (X0[j,i]-X_mean[j])/X_std[j]
        # Calculando a matriz de componentes principais ou
scores da matriz X0
    Y0 = np.dot(W.T,X0n)
    # Calculando T2_0
    T2_0 = np.zeros((sz_X0[1]))
    for i in range(sz_X0[1]):
    for j in range (k):
    T2_0[i] = T2_0[i] + (Y0[j,i]**2)/egva_corr[j]
    # Calculando Q
    Q_0 = np.zeros((sz_X0[1]))
    xI = np.dot(Y0[0:k,:].T,W_k.T)
    xR = X0n - xI.T
    for i in range(sz_X0[1]):
    for j in range (sz_X0[0]):
    Q_0[i] = Q_0[i] + (xR[j,i]**2)
        # Calculando T2_alpha
    T2_0 = sorted(T2_0,reverse=True) # Ordenando T2
decrescentemente
    N = alpha*sz_X0[1]/100.0 # Calculando o PERCENTIL (100-
alpha) da matriz X0
    if N - int(N) > 0.0001:
            N = N + 1
        N = int(N)
        T2_alpha = T2_0[N]
        T2A = np.ones((sz_X0[1],1))*T2_alpha # Criando um
vetor para T2_alpha
        # Calculando Q_alpha
        Q_0 = sorted(Q_0,reverse=True) # Ordenando T2
decrescentemente
    Q_alpha = Q_0[N]
        QA = np.ones((sz_X0[1],1))*Q_alpha # Criando um
vetor para Q_alpha

        # OPERACAO COM FALHAS
    Xf = Xf.T # Transpondo a matriz Xf para a forma usual de
trabalho
    sz_Xf = np.shape(Xf)
    # Normalizando a matriz de observacoes do experimento
com falhas
    Xfn = np.zeros((sz_Xf[0],sz_Xf[1]))
    for i in range (sz_Xf[1]):
    for j in range (sz_Xf[0]):
    Xfn[j,i] = (Xf[j,i]-X_mean[j])/X_std[j]
    # Calculando a matriz de componentes principais ou
scores da matriz Xf
    Y = np.dot(W.T,Xfn)
    # Calculando T2
```

```
   T2 = np.zeros((sz_Xf[1]))
   for i in range(sz_Xf[1]):
   for j in range (k):
   T2[i] = T2[i] + (Y[j,i]**2)/egva_corr[j]
       # Calculando Q
       Q = np.zeros((sz_Xf[1]))
   xI = np.dot(Y[0:k,:].T,W_k.T)
   xR = Xfn - xI.T
   for i in range(sz_Xf[1]):
   for j in range (sz_Xf[0]):
   Q[i] = Q[i] + (xR[j,i]**2)

       # Linhas de plot do sistema
   importmatplotlib.pyplot as plt
   passo = 0.05 # Tempo entre cada observacao, que
corresponde a 3min
       t = np.arange(0,48,passo)
   N_falha = str(N_falha)
   # Plotando T2
   plt.plot(t,T2,'b')
   plt.ylabel('T2')
   plt.xlabel('Time (hr)')
   plt.title('Fault ' + N_falha)
       #plt.title('Fault' + N_falha)
   plt.plot(t,T2A,'r--')
       # Plotando Q
   plt.figure(2)
   plt.plot(t,Q,'g')
   plt.ylabel('Q')
   plt.xlabel('Time (hr)')
   plt.title('Fault ' + N_falha)
       #plt.title('Fault ' + N_falha)
   plt.plot(t,QA,'r--')
   return

   # Rodar o conjunto de observações desejado
   import numpy as np
   import pca_function as pf
   X = np.genfromtxt("d00.dat") # Abrir o arquivo da
operação modelo
   X0 = np.genfromtxt("d00_te.dat") # Abrir o arquivo da
segunda operação modelo
   Xf = np.genfromtxt("d04_te.dat") # Abrir o arquivo da
falha desejada
   falha = 4 # Número da falha
   criterio_K = 85 # Critério de escolha de K (porcentagem
de variância explicada)
   alpha = 1 # Nível de significância dos limites de T2 e Q
   pf.pca(X,X0,Xf,falha,criterio_K,alpha)
```

Plugin PCA no EPM (Integração PCA-EPM via Python)

```
# -*- coding: utf-8 -*-
# Rodar o conjunto de observacoes desejado
import EpmDatasetPlugins as ds
import numpy as np
import matplotlib.pyplot as plt


def fazMatrizes(vetorVeriaveis,p,numObservacoes):
    n=p/3
    X = np.ones([n, numObservacoes])
    X0 = np.ones([n, numObservacoes])
    Xf = np.ones([n, numObservacoes])
for i in range(n):
X[i,:]=vetorVeriaveis[i]
        X0[i, :] = vetorVeriaveis[i+n]
        Xf[i, :] = vetorVeriaveis[i+2*n]
        #X0[i, :] = vetorVeriaveis[i]
        #Xf[i, :] = vetorVeriaveis[i]
    return X, X0, Xf

def funcaoLog(arquivoDeLog,stringParaSerSalva):
    arquivoDeLog.write(stringParaSerSalva + '\n')

def salvaGraficos(caminho,T2,Q):
    arquivo = open(caminho,'wb')
    n = len(T2)
    stringArquivo=''
    for i in range(n):
        stringArquivo += (str(T2[i]) + '  ')
    stringArquivo+='\n'
for i in range(n):
stringArquivo += (str(Q[i]) + '  ')
    arquivo.write(stringArquivo)
    arquivo.close()

def ReorganizaVetores(X,X0,Xf):
    # Colocando os vetores na ordem correta
    sz = np.shape(X)
    n = sz[0]
    numObservacoes = sz[1]
    x = np.ones([n, numObservacoes])
    x0 = np.ones([n, numObservacoes])
    xf = np.ones([n, numObservacoes])

    x[0, :] = Xf[3, :]
    x[1, :] = Xf[2, :]
    x[2, :] = Xf[1, :]
    x[3, :] = Xf[0, :]
    x[4, :] = X0[5, :]
```

```
    x[5, :] = X0[4, :]

    x0[0, :] = X0[3, :]
    x0[1, :] = X0[2, :]
    x0[2, :] = X0[1, :]
x0[3, :] = X0[0, :]
    x0[4, :] = X[3, :]
    x0[5, :] = X[2, :]

    xf[0, :] = X[1, :]
    xf[1, :] = X[0, :]
xf[2, :] = Xf[4, :]
if Xf[5,0] < 10:
        xf[3, :] = Xf[5, :]
        xf[4, :] = X[5, :]
    else:
        xf[3, :] = X[5, :]
xf[4, :] = Xf[5, :]
    xf[5, :] = X[4, :]

    X = x
    X0 = x0
Xf = xf
return X, X0, Xf

def PrintVetores(X,X0,Xf):
print("Matriz X, primeiros elementos: " + str(X[0, 0]) + "
" + str(X[1, 0]) + "   " + str(X[2, 0]))
print("   "+ str(X[3, 0]) + "   " + str(X[4, 0]) + "   " +
str(X[5, 0]) + "    ")
print(" Matriz X0, primeiros elementos: " + str(X0[0, 0]) +
"   " + str(X0[1, 0]) + "   " + str(X0[2, 0]) + "   ")
    print(str(X0[3, 0]) + "   " + str(X0[4, 0]) + "   " +
str(X0[5, 0]) + "    " + " Matriz Xf, primeiros elementos:
")
print(str(Xf[0, 0]) + "   " + str(Xf[1, 0]) + "   " +
str(Xf[2, 0]) + "   " + str(Xf[3, 0]) + "   " + str(Xf[4,
0]))
    print("   " + str(Xf[5, 0]))

@ds.epm_dataset_method_plugin('PCA',1)
def pca():
    caminhoDoArquivoLog =
'C:/Users/Daniel/Desktop/lop2016/codigos/arquivoLog.txt'
    arquivoLog = open(caminhoDoArquivoLog,'wb')

    p = len(ds.EpmDatasetAnalysisPens.SelectedPens)
vetoresVariaveis=[]

    for i in range(p):
```

```
vetoresVariaveis.append(ds.EpmDatasetAnalysisPens.SelectedP
ens[i].values['Value'])
    observacoes =
len(ds.EpmDatasetAnalysisPens.SelectedPens[0].values['Value
'])
X, X0, Xf = fazMatrizes(vetoresVariaveis,p,observacoes)
    #funcaoDebug(vetoresVariaveis)

    X,X0,Xf = ReorganizaVetores(X,X0,Xf)
    PrintVetores(X,X0,Xf)
    criterio_K = 55
    alpha = 1

    # MODELO
    X_mean = np.mean(X,axis=1) # Calculando o vetor medio
de dados
    X_std = np.std(X,axis=1)  # Calculando o vetor desvio
padrao de dados
    corr_X = np.corrcoef(X) # Funcao que calcula a matriz
de correlacoes
    sz_X = np.shape(X) # Verificando o tamanho da matriz de
dados
    egva_corr, egvec_corr = np.linalg.eig(corr_X) #
Autovalores e autovetores de corr_data

    funcaoLog(arquivoLog,str(egva_corr))
    funcaoLog(arquivoLog, str(egvec_corr))

    # Associando autovalores e autovetores
eigpairs = [(np.abs(egva_corr[i]), egvec_corr[:,i]) for i
in range(len(egva_corr))]
eigpairs.sort() # Ordenando autovalores por ordem crescente
    eigpairs.reverse() # Invertendo a ordem
    egva_tot = sum(egva_corr) # Somando os autovalores
    # Contribuicao relativa de cada autovalor
egva_exp = [(i*1.0/egva_tot)*100 for i in sorted(egva_corr,
reverse=True)]
egva_sum = np.cumsum(egva_exp) # Soma acumulada das
contribuicoes relativas
    # Escolhendo o k (numero de autovetores representativos
do sistema)
k = 0
    while(egva_sum[k] < criterio_K):
k = k + 1
k = k + 1
    # Matriz de autovetores completa
W = np.hstack((eigpairs[i][1].reshape(sz_X[0],1)) for i in
range(sz_X[0]))
egva_corr = sorted(egva_corr,reverse=True) # Reordenando os
autovalores para corrigir um erro
```

```
    # Matriz dos k primeiros autovetores ou loadings
W_k = np.hstack((eigpairs[i][1].reshape(sz_X[0],1)) for i
in range(k))
# OPERACAO SEM FALHAS
sz_X0 = np.shape(X0)
# Normalizando a matriz de observacoes do experimento com
falhas
    X0n = np.zeros((sz_X0[0],sz_X0[1]))
for i in range(sz_X0[1]):
        for j in range(sz_X0[0]):
X0n[j,i] = (X0[j,i]-X_mean[j])/X_std[j]
    # Calculando a matriz de componentes principais ou
scores da matriz X0
Y0 = np.dot(W.T,X0n)
# Calculando T2_0
    T2_0 = np.zeros((sz_X0[1]))
for i in range(sz_X0[1]):
        for j in range(k):
T2_0[i] = T2_0[i] + (Y0[j,i]**2)/egva_corr[j]
# Calculando Q
Q_0 = np.zeros((sz_X0[1]))
    xI = np.dot(Y0[0:k,:].T,W_k.T)
xR = X0n - xI.T
    for i in range(sz_X0[1]):
        for j in range(sz_X0[0]):
Q_0[i] = Q_0[i] + (xR[j,i]**2)
    # Calculando T2_alpha
T2_0 = sorted(T2_0,reverse=True) # Ordenando T2
decrescentemente
N = alpha*sz_X0[1]/100.0 # Calculando o PERCENTIL (100-
alpha) da matriz X0
    if (N - int(N)) != 0:
        N = int(N+1)
    N = int(N)
    # Calculando Q_alpha
    Q_0 = sorted(Q_0,reverse=True) # Ordenando T2
decrescentemente
    # OPERACAO COM FALHA
sz_Xf = np.shape(Xf)

# ========== LOG
    funcaoLog(arquivoLog,str(sz_Xf[0]))
    funcaoLog(arquivoLog, str(sz_Xf[1]))
# ========== LOG


    T2_alpha = T2_0[N]
T2A = np.ones((sz_Xf[1]))*T2_alpha # Criando um vetor para
T2_alpha
    Q_alpha = Q_0[N]
```

```
    QA = np.ones((sz_Xf[1]))*Q_alpha # Criando um vetor
para Q_alpha
    # Normalizando a matriz de observacoes do experimento
com falhas
    Xfn = np.zeros((sz_Xf[0],sz_Xf[1]))
for i in range(sz_Xf[1]):
        for j in range(sz_Xf[0]):
            Xfn[j,i] = (Xf[j,i]-X_mean[j])/X_std[j]
# Calculando a matriz de componentes principais ou scores
da matriz Xf
Y = np.dot(W.T,Xfn)
# Calculando T2
    T2 = np.zeros((sz_Xf[1]))
for i in range(sz_Xf[1]):
        for j in range(k):
T2[i] = T2[i] + (Y[j,i]**2)/egva_corr[j]
    # Calculando Q
    Q = np.zeros((sz_Xf[1]))
xI = np.dot(Y[0:k,:].T,W_k.T)
xR = Xfn - xI.T
    for i in range(sz_Xf[1]):
        for j in range(sz_Xf[0]):
Q[i] = Q[i] + (xR[j,i]**2)
    # Linhas de plot do sistema
    #passo = 0.05  # Tempo entre cada observacao, que
corresponde a 3min
t_final = sz_Xf[1]/20.0
    t = np.linspace(0,t_final,sz_Xf[1])    # Plotando T2

    funcaoLog(arquivoLog,str(t))



    plt.plot(t, T2, 'b')
plt.ylabel('T2')
    plt.xlabel('Time (hr)')
plt.title('Fault')

    plt.plot(t, T2A, 'r--')
    # Plotando Q
    plt.figure(2)
    plt.plot(t, Q, 'g')
plt.ylabel('Q')
plt.xlabel('Time (hr)')
    plt.title('Fault')
    plt.plot(t, QA, 'r--')
plt.show()

caminhoDoArquivo =
'C:/Users/Daniel/Desktop/lop2016/codigos/resultadosGrafico.
txt'
```

```
    salvaGraficos(caminhoDoArquivo,T2,Q)
    arquivoLog.close()
```

Importação de dados (arquivos .txt) para o EPM

```
# -*- coding: utf-8 -*-

import pandas as pd
import numpy as np
from datareader import getdata



##Módulos do EPM SDK para linguagem Python
import epmsdk
import epmsdk.communication as epmcomm
import epmsdk.dataaccess as epmda
import epmsdk.historicaldata as epmhda

from epmsdk.dataaccess.valuequality import ValueQuality

#variáveis

coluna = []

##Conexão ao EPM
SrvEntryName = 'localhost'
UserAuthName = 'sa'
UserAuthPass = 'lop2016'
ConnectionArgs = (None, None, SrvEntryName, UserAuthName,
UserAuthPass,)
try:
    epmserver = epmcomm.epmConnect(*ConnectionArgs)
except epmsdk.EpmException as ex:
```

```
    print 'Connection error:
{}'.format(epmsdk.EpmExceptionCode[ex.Code])
    print 'Details: {!r}'.format(ex)
raw_input('Program should stop now!')
    exit(1)
print 'Connection Succeeded!'

##Escolha das datas
##http://pandas.pydata.org/pandas-
docs/stable/timeseries.html
dayLength = 24*60  # em minutos
dateLength = 31*dayLength
dates = pd.date_range('20160301',periods=dateLength,
freq='3min')
print dates

##Leitura do DAT
desc =
np.dtype([('Value','>f8'),('Timestamp','object'),('Quality'
,'object')])
dataTmp = np.empty([2], dtype=desc) #quantidade de dados

caminhodoArquivo = 'resultadosgrafico.txt'
values = getdata(caminhodoArquivo) #busca valores -
datareader.py

#values = values[np.logical_not(np.isnan(values))] #remove
nans
j = 18
for j in range(2):
    column = values[j,:]

    i = 0
    for item in column:
```

```
        dataTmp['Value'] = item
        dataTmp['Timestamp'] = dates[i].to_datetime()
        dataTmp['Timestamp'] = dates[i].to_datetime()
        dataTmp['Quality'] = ValueQuality(192)

        print dataTmp
        print 'gravando...coluna'+str(j)+'...'+str(i)

        try:
            basic_variable                                    =
epmda.epmGetDataObject(epmserver,'coluna'+str(j))
except epmsdk.EpmException as ex:
            print'Erro  ao  buscar  a  Basic  Variable  no
EPM!{}'.format(ex)
exit(1)

        try:

epmhda.epmTagHistoryUpdate(basic_variable,dataTmp)
except epmsdk.EpmException as ex:
print'Erro ao gravar os dados no EPM!{}'.format(ex)
            exit(2)
        print 'Escrita OK!'

        i += 1
```

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRAWOS, M. **Thoughts on Process historians**. Disponível em: <https://www.linkedin.com/pulse/thoughts-process-historians-mina-andrawos-p-eng>. Acesso em: 25 jun. 2016.

ASPENTECH. **Manufacturing excellence depends on a good data foundation.**Disponível em: <http://www.aspentech.com/products/aspen-infoplus21/>.

CORSO SYSTEMS. **Choosing a Process Historian**. Disponível em: <http://corsosystems.com/white_papers/Top 5 Historian Criteria.pdf>. Acesso em: 25 jun. 2016.

DOWNS, J. J.; VOGEL, E. F. A plant-wide industrial process control problem. **Computers and Chemical Engineering**, v. 17, n. 3, p. 245–255, 1993.

FAIRCLOTH, K. Divide and conquer: analyzing big chunks of data Kelly Faircloth. v. 8, p. 4–5, 2016.

FILHO, C. S. **PIMS -Process Information Management System – Uma introdução**. Disponível em: <http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/Pims.PDF>.

GENERAL ELECTRIC. **Proficy Historian is now GE Historian**. Disponível em: <http://www.geautomation.com/products/proficy-historian>.

KANNO, M. Marcos na História da Visualização de dados. Disponível em: <http://www.ime.usp.br/~rvicente/historia_infografia.pdf>

KEIM, D. et al. Visual Analytics: Definition , Process , and Challenges. **Information Visualization - Human-Centered Issues and Perspectives**, p. 154–175, 2008.

MCAFEE, A.; BRYNJOLFSSON, E. **Big Data: The Management**

Revolution. Disponível em: <https://hbr.org/2012/10/big-data-the-management-revolution/ar>.

MITCHELL, R. L. **8 big trends in big data analytics**. Disponível em: <http://www.computerworld.com/article/2690856/big-data/8-big-trends-in-big-data-analytics.html>.

MONTGOMERY, D. **Introduction to statistical quality control**. [s.l: s.n.].

OSI SOFTWARES. **PI-System**. Disponível em: <http://www.osisoft.com/pi-system/>.

RIBEIRO, JOSÉ LUIS DUARTE, CATEN, C. S. TEN. Cartas de Controle para Variáveis, Cartas de Controle para Atributos, Função de Perda Quadrática, Análise de Sistemas de Medição. 2012.

RUSSELL, E.; CHIANG, L.; BRAATZ, R. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. **Chemometrics and Intelligent Laboratory Systems**, v. 51, p. 81–93, 2000.

SAEY, T. H. Big data, big challenges: as researchers begin analyzing massive datasets, opportunities for chaos and errors multiply Tina Hesman Saey. **Science News**, v. 187.3, p. 2, 2016.

SCHINIDER ELETRONIC SOFTWARE. Wonderware Historian Industrial Data Management. 2016.

SOUZA, A. M. Monitoração E Ajuste De Realimentação Em Processos Produtivos Multivariados. 2000.

SYLUTION. **The Process Historian – A Great Tool for Process Improvement**. Disponível em: <http://sylution.com/OPEX/?p=142>. Acesso em: 25 jun. 2016.

TAN, K. H. et al. Harvesting big data to enhance supply chain innovation capabilities: An analytic infrastructure based on deduction graph. **International Journal of Production Economics**, v. 165, p. 223–233, 2015.

VERHAPPEN, I. Turning big data into information benefts the bottom line Making better use of all available data and managing it so that it is transformed from data into knowledge not only improves the “ bottom line ” through better operations , but also increases oppor. 2013.

YAMAGUCHI, M. Y. Sincronização das bases de tempo de CLPs distribuídos numa rede de automação de processo industrial. 2006.