UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA QUÍMICA
ÁREA DE CONCENTRAÇÃO: SISTEMAS DE PROCESSOS QUÍMICOS E INFORMÁTICA

RECONCILIAÇÃO DE DADOS DE PROCESSOS E DETECÇÃO DE ERROS GROSSEIROS EM SISTEMAS COM RESTRIÇÕES NÃO-LINEARES

por
ANTONIO CÉSAR TEIXEIRA
Engenheiro Químico

orientador
Dr. JOÃO ALEXANDRE FERREIRA DA ROCHA PEREIRA
Professor Titular

Dissertação submetida à Comissão de Pós-Graduação da Faculdade de Engenharia Química como parte dos requisitos para obtenção do título de Mestre em Engenharia Química.

Campinas(SP):Agosto/97.

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

T235r

Teixeira, Antonio César
Reconciliação de dados de processos e detecção de erros grosseiros em sistemas com restrições não-lineares.-- Campinas, SP: [s.n.], 1997.

Orientador: João Alexandre Ferreira da Rocha Pereira
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Química.

1. Localização de falhas (Engenharia). 2. Programação não-linear. 3. C++ (Linguagem de programação por computador). 4. Controle de processos químicos. 5. Lagrange, Funções de. 6. Matrizes (Matemática). I. Pereira, João Alexandre Ferreira da Rocha. II. Universidade Estadual de Campinas. Faculdade de Engenharia Química. III. Título.

Esta versão corresponde à redação final da Tese de Mestrado defendida por ANTONIO CÉSAR TEIXEIRA e aprovada pela Comissão Julgadora em 14/08/1997.

Orientador

Prof. Dr. João Alexandre Ferreira da Rocha Pereira

Dissertação defendida e aprovada, em 14 de agosto de 1997, pela banca examinadora constituída pelos professores doutores:

Prof. Dr. João Alexandre Ferreira da Rocha Pereira -Unicamp

Prof. Dr. Fernando Augusto Pinto Garcia-Universidade de Coimbra

Profª. Dra. Ana Maria Frattini Fileti-Unicamp

Campinas(SP), 14 de Agosto de 1997.

AGRADECIMENTOS

Aos professores da FEQ-UNICAMP, em geral, e aos do DESQ, em particular.

Ao professor Pereira.

Ao Professor Roger.

À Professora Ana Frattini.

À Andréia Pio.

À CAPES pelo bolsa-mestrado.

Aos colegas do Departamento, em geral e ao José Edson e ao Marcone, em especial.

Aos funcionários.

À Minha Família
À Cláudia
Ao Professor Pereira

LISTA DE SÍMBOLOS

Os símbolos apresentados aqui são genéricos, ou seja, forma usados em todos os capítulos. Onde houver especificidade de notação, o respectivo capítulo onde o símbolo é válido é indicado entre parenteses.

a	vetor dos ajustes das taxas de fluxos medidas
A	matriz incidência do processo
B	matriz balanço do processo
$\mathbf{B_0}$, $\mathbf{B_1}$, $\mathbf{B_3}$	colunas dos **B** correspondentes às taxas de fluxos fixas, medidas e não medidas.
$\mathbf{B_3}$	colunas de **B** correspondentes aos componentes com concentração medidas em correntes com taxas de fluxos totais não-medidas
c (3)	vetor das taxas de fluxos constantes
c	vetor concentração correspondente a $\mathbf{B_1}$
$\mathbf{c_i}$	número de espécies na corrente k
C	número de componentes
d	vetor das concentrações medidas
$\mathbf{d_k}$	elementos de **d** para a corrente k
D (5)	$\mathbf{Y^T C}$
D (4)	equação (4.12)
e	vetor desbalanço
F(a)	função objetivo
H	$\equiv \mathbf{Y^T B_1} \sum \mathbf{B_1^T Y}$
I	matriz identidade
j	índice do número da corrente
J	número de correntes
k	índice do número de nós
K	número de unidades de processso (nós)
m	número de equações de restrição
M	taxa de fluxo total das correntes com taxas de fluxos dos componentes medidas
$\mathbf{n_i}$ (4)	número de variáveis na categoria i = 0,1, 2 3
n	vetor das taxas de fluxos não medidas na categoria 2
N	matriz diagonal dos elementos de **n**
p	número de reações independentes
P (3,4)	$= \left[\mathbf{B_3} : \mathbf{S^T}\right]$
P (5)	matriz das retrições para as quantidades não-medidas na categoria 3
$\mathbf{r_P}$	posto da matriz P
$\mathbf{R_k}$	número de reações no nó k
$\mathbf{s_i}$	número de correntes na categoria i
$\mathbf{s_2}$	número de correntes na categoria 2
$\mathbf{S_k}$	matriz estequimétrica para a reação no nó k
S	matriz estequiométrica do processo
t	vetor das variações de v
T	$\mathbf{Z^T H Z}$

$\mathbf{u}_k$	vetor unitário, k-ésima coluna de I
$\mathbf{u}$	vetor das taxas de fluxo não-medidas
$\mathbf{v}$ (3,4)	$=\left[\mathbf{u}^T \xi^T\right]^T$
$\mathbf{v}$ (5)	vetor das quantidades não medidas na categoria 3
$\mathbf{w}$ (3)	vetor do teste estatístico normal
$\mathbf{W}$	matriz para o teste qui quadrado
$\mathbf{x}$	vetor das taxas de fluxo verdadeiras
$\tilde{x}$	vetor das taxas de fluxos medidos
$\mathbf{Y}$	definido por $\mathbf{Y}^T\mathbf{P} = \mathbf{0}$
$\mathbf{z}$	teste estatístico

Letras Gregas

δ_j	vetor unitário **j**
λ	vetor dos multiplicadores de Lagrange
ξ	vetor extensão das reações
σ_m^2	variância de **M**
Σ	variância de $\tilde{x}$
Σ_1	componente da taxa de fluxo x
Σ_c	concentrações **c**
Σ_d	concentrações **d**
Σ_2	matriz da equação (4.19).

ÍNDICE

PÁGINA

RESUMO

O tratamento de dados de processos industriais envolve uma série de medidas as quais visam a dar mais confiabilidade aos valores medidos diretamente e aos inferidos indiretamente, para sua utilização no controle dos mesmos. Estão entre estas medidas, a classificação, a reconciliação e a retificação de dados.

Este trabalho apresenta uma metodologia para reconciliação de dados de processos industriais onde não existam erros grosseiros entre os valores das variáveis medidas, sejam as restrições lineares ou não-lineares. A ferramenta utilizada é a projeção matricial a qual é utilizada para simplificar as equações de balanços (restrições) de massa e/ou energia de processos complexos. O objetivo é minimizar o erro ou a diferença entre os valores reconciliados e os valores reais.

A partir de cálculos intermediários do procedimento de reconciliação, foi desenvolvido um segundo procedimento para detecção de erros grosseiros entre os valores das variáveis medidas. A presença de erros grosseiros entre as medidas inutiliza os dados reconciliados, contudo fornece subsídios para, a partir deste segundo procedimento, determinar a presença do erro grosseiro.

Os três procedimentos, acima citados, para o tratamento de dados do processo, são descritos neste trabalho, com os elementos teóricos desenvolvidos de modo detalhado. Dois programas computacionais são escritos e aqui apresentados, sendo que o primeiro faz a reconciliação de dados e o segundo detecta a existência ou não de erros grosseiros entre os valores apresentados.

1 INTRODUÇÃO

Um processo químico real, onde se monitoram variáveis a fim de se controlar o perfeito funcionamento do mesmo, pode apresentar falhas tanto devido à saída do mesmo das condições normais de operação, a qual pode estar, ou não, em estado estacionário, bem como devido a falhas nos instrumentos de medida e controle.

Confiança nos dados medidos é de fundamental importância para diagnóstico, identificação e controle de um processo, por questões econômicas ou de segurança. Medidas com erros grandes, sejam randômicos, de bias(dos instrumentos) ou grosseiros, levam a resultados que conduzem a um falso controle do processo. Desta maneira, a reconciliação de dados é um procedimento que está situado entre a aquisição de dados e a decisão a ser tomada para controle, KELLER(1991).

Os equipamentos de medida comportam erros os quais lhes são inerentes e desde que estejam dentro de uma estreita faixa (definida para cada tipo de processo) pode ser aceitável para os propósitos da reconciliação. Por erros aceitáveis, entende-se erros pequenos que apresentem comportamento estatístico conhecido, sejam randômicos ou devido ao vício dos instrumentos(bias). Erros grosseiros entre as medidas inviabilizam a reconciliação de dados.

Medidas com erros aleatórios, somados a outros desvios, também de natureza conhecida, devidos às próprias características do processo, podem ser "refinadas" de tal modo que se tenha o processo funcionando o mais próximo possível das condições ideais, que é onde não se verificam desvios de qualquer natureza (situação apenas teórica). Aqui, condições ideais significam fechar balanços de massa e/ou energia de maneira exata.

Este refinamento pode ser conseguido aplicando-se procedimentos da reconciliação de dados, os quais são baseados na obtenção de estimativas otimizadas de parâmetros medidos, e em alguns casos não-medidos (coaptação), levando-se em conta o histórico do processo através da matriz variância-covariância das medidas. Desta forma é possível diminuir o efeito de "pequenos" desvios nos valores das variáveis através do uso de redundâncias (temporais, espaciais ou funcionais) das medidas, LIEBMAN(1992). Medidas são espacialmente redundantes se há mais dados que os suficientemente necessários para definir o modelo do processo em qualquer instante de tempo, isto é, são sistemas superdimensionados. São temporalmente redundantes se valores medidos, passados, estão disponíveis e podem ser usados para propósitos de estimação de valores. Apenas modelos dinâmicos comportam ambos os tipos de redundâncias, as quais são expressas em equações algébricas e diferenciais.

Apesar de, em alguns dados medidos, a correção ser mínima, pode num sistema complexo ser de grande valor para as necessidades de otimização e controle. Com a reconciliação deixa-se de corrigir erros que não ocorreram no processo e que por ventura foi detectado em alguma variável por uma falha de medida qualquer. Diminui-se portanto as flutuações do mesmo.

O grande desenvolvimento dos controladores digitais integrados por meio de computadores em detrimento dos controladores não integrados, possibilita tratar dados de um sistema complexo com rapidez suficiente para se corrigir possíveis desvios antes que eles cheguem aos controladores ou atuadores.

Processos e equipamentos que causem erros ou desvios que estejam fora da faixa aceitável ou que possibilitem o aparecimento de erros grosseiros, os quais não podem ser encarados do ponto de vista estatístico como "previsíveis", não servem ao procedimento

de reconciliação de dados. Tais erros aparecem devido a fatores como vazamentos, queda de tensão nas linhas de transmissão, intempéries e até mesmo por má manutenção de equipamentos de medida. Medidas corrompidas por erros desta natureza precisam ser identificadas e corrigidas ou simplesmente deletadas antes que possam ser usadas na reconciliação dos dados.

Um fluxograma simplificado do caminho a ser seguido a partir da coleta de dados de um processo pode ser visto na figura 1.1

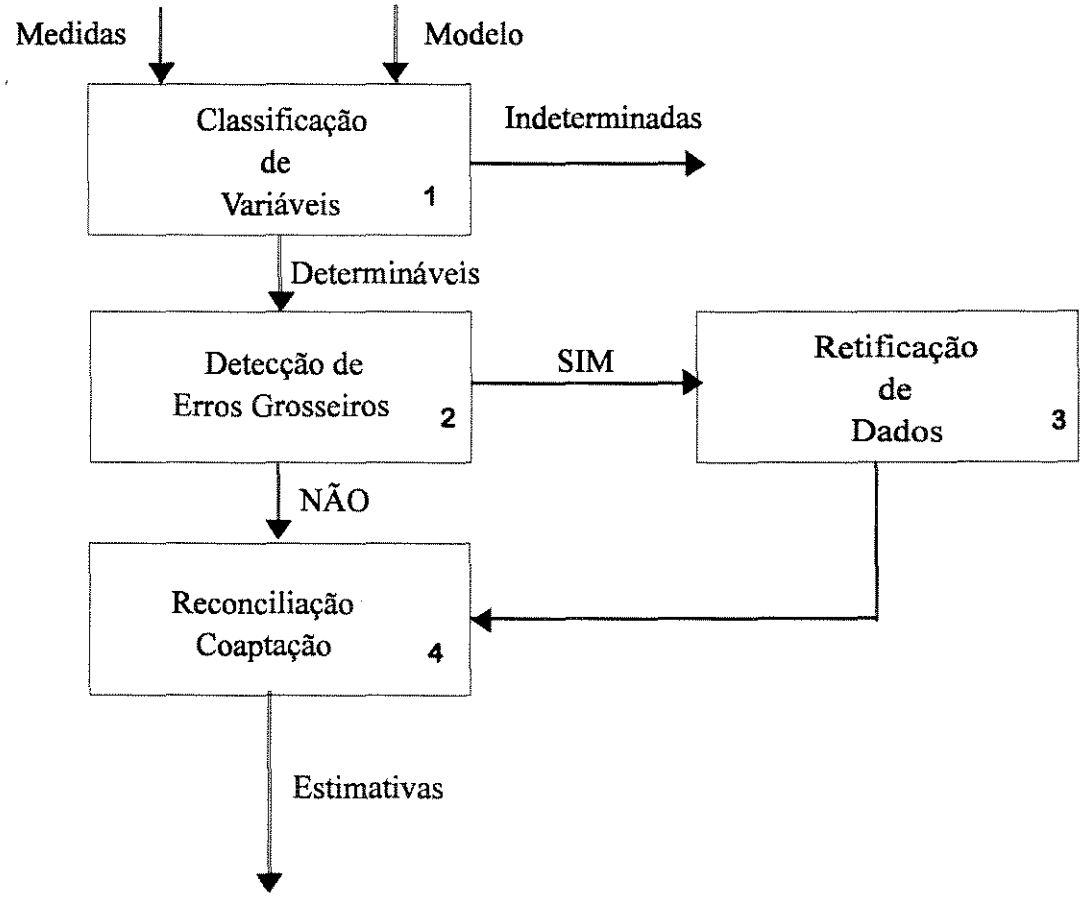


Figura1.1-Fluxograma simplificado de tratamento de dados

O primeiro passo se refere à determinação de quais variáveis são observáveis ou não e quais são redundantes ou não, (CROWE(1986), STANLEY E MAH(1981a)).

Na segunda etapa, todos os erros grosseiros são identificados e removidos. Estratégias de detecção de erros grosseiros são tipicamente baseadas em testes de hipóteses estatísticas, MAH(1987). As medidas com estes erros grosseiros são então retificadas na terceira etapa. A utilização de medidas com erros grosseiros inviabilizaria a etapa quatro.

A reconciliação de dados é feita a partir das medidas de fluxos e concentrações ao longo do processo e o controle dos dados de fluxos mássicos e/ou volumétricos

permite controlar o sistema em todas as suas extensões uma vez que as diversas variáveis estão "amarradas" entre si por leis físicas e químicas.

Partindo-se desta premissa, pode-se classificar um sistema como linear quando tem os valores de taxas de fluxos totalmente medidas, onde se conhece as concentrações nas correntes, e não-linear quando existirem correntes onde as taxas de fluxo totais são desconhecidas, onde se conhecem as concentrações dos componentes que atravessam a corrente.

Apesar de o objetivo do trabalho ser o de tratar dados em processos com restrições não-lineares, nota-se que os procedimentos de cálculo em ambos os casos são de mesma natureza, por isso, devido a algumas particularidades existentes entre ambos os casos, será aqui destinado um capítulo para descrever o caso linear, o capítulo 3.

No capítulo 4 serão descritos os procedimentos para o caso com restrições não-lineares, mostrando as particularidades do programa de reconciliação e a forma de utilização do mesmo.

O capítulo 5 descreve o procedimento de detecção de erros grosseiros para o qual também existe um programa computacional. A detecção de erros grosseiros na verdade é parte de um conjunto de técnicas mais amplo denominado de retificação de dados. Contudo, para isso seria necessário que se fizesse um estudo de observabilidade e redundância nas variáveis não-medidas e nas medidas, respectivamente. Não sendo esta uma proposta deste trabalho, não será aqui apresentado nenhum programa computacional para tais determinações mas isto não impede que se descreva as equações e os algoritmos empregados por CROWE(1989b), completando assim o circuito de tratamento de variáveis de processo utilizando a projeção matricial.

Finalmente o capítulo 6 apresenta o estudos de 3 casos distintos apresentados na literatura citada neste trabalho.

2 REVISÃO BIBLIOGRÁFICA

A reconciliação de dados requer o conhecimento completo da estrutura do processo. Informações sobre as correntes e sobre os componentes das correntes, possibilitam escrever o problema em forma de balanços levando ao equacionamento completo do sistema em termos de massa e/ou energia. A maneira de tratar tais conjuntos de equações varia de autor para autor assim como também os modelos matemáticos por eles utilizados. Teoria dos grafos, projeção matricial ou redes neurais são algumas destas ferramentas que possibilitam manipular as restrições do problema de reconciliação, sejam eles com restrições lineares ou não. O estado (estacionário/dinâmico) em que operam as unidades em estudo é também de grande determinância na modelagem do problema. Assim, sistemas operando em estado estacionário terão uma descrição muito menos complexa que aquela necessária para descrever sistemas em estado não-estacionário.

Nem todas as variáveis de uma planta são medidas e várias são as razões para que tal não ocorra. Custos, conveniências ou factibilidades técnicas estão entre algumas destas razões. Em alguns casos é possível, contudo, se estimar os valores de algumas variáveis (coaptação) não-medidas através de balanços, seja de massa, seja de energia, ou de ambos simultaneamente. A estrutura do processo e a disposição dos instrumentos de medida passa a ser de fundamental importância determinando, em última análise, a possibilidade ou não de se reconciliar os dados. Para uma dada rede de processos com uma dada disposição de instrumentos, se é possível se fazer uma mudança em uma variável sem que esta mudança seja observada pelos instrumentos, a variável é dita não observável. De acordo com esta definição de KRETSOVALIS e MAH(1987), uma variável medida é obviamente observável, mas uma variável não-medida pode ser ou não observável. O desejável seria ter todas as variáveis envolvidas no processo medidas ou observáveis.

Se uma medida associada a uma variável for deletada e ainda assim ela permanecer observável, diz-se que a mesma é redundante. Na análise de um processo são requeridas certas variáveis observáveis, outras medidas mas não redundantes e ainda outras medidas e redundantes. Esta classificação tem grande importância pois permite por exemplo a análise das conseqüências que a falha de um instrumento de medida ou de um equipamento qualquer do processo produziria no sistema como um todo. Além do mais, o conhecimento do "status" de cada variável possibilita uma melhor distribuição dos instrumentos de medida dentro da planta, diminuindo os custos de instalação e manutenção dos mesmos. Quanto mais complexa e integrada uma planta, mais se faz necessária a utilização de algoritmos de classificação de variáveis a fim de melhor instrumentá-la.

Proposto o problema da determinação da observabilidade e redundância das variáveis de um processo, há diferentes caminhos a se seguir a fim de conseguir chegar à melhor distribuição das medidas a serem tomadas.

Dentre os primeiros métodos conhecidos, estão os propostos por VACLAVEK et al (1976, 1976a, 1976b) e por ROMAGNOLI E STEPHANOPOULOS(1980) nos quais eles assumem que a composição de cada corrente deve ser completamente medida ou completamente não-medida. VACLAVEK(1976) e seus colaboradores propuseram o algoritmo baseado em grafos-orientados onde as regras para classificação de não-redundância são empregadas para "reduzir o esquema de balanço" e identificar as medidas redundantes. A aplicação deste algoritmo é limitada pelo fato de que estas

regras constituem condições apenas suficientes para classificação de não-redundância. Em outras palavras, medidas classificadas como não-redundantes, são realmente não-redundantes, mas algumas medidas não-redundantes podem também, erroneamente, ser classificadas como redundantes. O algoritmo empregado por ROMAGNOLLI e STEPHANOPOULOS(1980) é a equação-orientada. A possibilidade de solução das equações do balanço nodal é examinada e um conjunto de algoritmos é empregado para classificar simultaneamente medidas e variáveis, STADTHERR et al.(1973). A classificação da observabilidade de uma variável não-medida depende de se ela pode ou não ser designada à saída do balanço nodal da equação a qual contém somente variáveis observáveis. Entretanto, se somente os balanços nodais forem considerados, pode-se levar a uma classificação incorreta, uma vez que as variáveis não-medidas não constantes entre as variáveis de saída , do balanço nodal podem também não ser observável.

Serão apresentadas aqui as análises dos trabalhos de vários autores cobrindo os temas apresentados nesta dissertação. São trabalhos que tratam da reconciliação de dados, da observabilidade e da redundância de variáveis e da detecção de erros grosseiros seja em processos dinâmicos ou estacionários com restrições lineares e não-lineares. O critério aqui adotado será o do ano de publicação do trabalho o que se por um lado leva a perda de sequencionamento entre trabalhos sobre o mesmo tema, por outro mostra o entrelaçamento entre os mesmos evitando mostrá-los de forma estanque como se fossem assuntos separados sem que o desenvolvimento de um não tivesse importância no equacionamento do outro.

STADHERR et al. (1973) desenvolveram um algoritmo para a resolução de sistemas de equações não-lineares os quais apresentam mais equações que incógnitas, ditas por tal, esparsas. O procedimento consiste em "partir" um conjunto grande de equações separando-as em conjuntos menores de tal modo que possam ser resolvidos seqüencialmente em uma determinada ordem, isto quando possível, pois quando não, é necessário resolver o conjunto de equações de modo iterativo e segundo os autores, a melhor maneira de se conseguir isto, até então, é através do algoritmo de CHRISTENSEN(1970). O modelo proposto por STADHERR et al., contudo, apresentou resultados superiores àquele, e apesar de reconhecerem que o critério de exclusão das variáveis ser incompleto, asseguram que o método é suficiente para ordenar pequenos sistemas até mesmo sem ajuda de computador. O problema de encontrar soluções se torna sério quando o problema é muito esparso para permitir a exclusão de algumas seqüências, mas mesmo neste caso, o algoritmo pode ser usado para fazer uma busca parcial de sequências as quais possam ser utilizadas em outras estratégias.

VACLAVEK e LOUCKA(1976) propuseram dois algoritmos os quais classificam os parâmetros de uma planta e determinam as medidas necessárias para caracterizar um processo. Os resultados obtidos identificam parâmetros que podem ser agrupados em três conjuntos: 1-o dos parâmetros que devem obrigatoriamente ser medidos; 2-o dos parâmetros que não precisam ser medidos; 3-o dos parâmetros que deveriam ser medidos. Os algoritmos de classificação são baseados na idéia de redução das equações de balanços (RBS) e as correntes do processo são submetidas a uma das duas operações: 1-adição dos nós conectados com as correntes internas (entre nós), ou; 2-eliminação de um nó conectado com uma corrente externa de modo que todas as correntes internas, originalmente coincidentes com os nós eliminados, se tornam correntes externas. Quanto mais complicado o sistema, mais complicado o esquema de redução. Uma utilização sucessiva do algoritmo de classificação permite de maneira indireta determinar a

disposição dos valores a serem medidos. Contudo, o segundo algoritmo de "escolha" foi desenvolvido de modo a se evitar tal procedimento.

VACLAVEK et al.(1976a) propuseram um algoritmo de classificação para parâmetros de correntes de processo a fim de reduzir a quantidade de arcos e para a classificação das variáveis em medidas ou não medidas. O algoritmo é definido de tal modo que se possa obter resultados adicionais que tornarão possível a simplificação da solução das equações no cálculo dos balanços materiais. Foi aqui considerado que se um parâmetro não-medido é determinável a partir de um cálculo de balanço, então uma medida adicional deste parâmetro dará um valor que, em geral, não concordará com o valor calculado. Consequentemente, o mesmo é corrigível. De outro modo, se o parâmetro é indeterminável, é incorrigível.

HLAVACEK(1977) apresenta um dos primeiros trabalhos de análise para os sistemas operando nos dois modos de funcionamento (transiente e em estado estacionário). Comenta sobre a necessidade de ajustes dos dados os quais são obtidos sob condições não-ideais (flutuações das condições ambientais, imprecisões nas leituras medidas, etc.) a fim de satisfazer as relações entre os parâmetros das correntes da planta. HLAVACEK faz o balanço para o ajuste de sistemas multicomponentes mostrando o aumento da complicação do problema em comparação ao caso de sistemas com apenas um componente. No caso de sistemas multicomponentes, diz que geralmente se depara com restrições não-lineares mostrando os cinco caminhos possíveis de se seguir na busca da solução. No caso de sistemas mal-condicionados, o autor mostra que a utilização do critério dos mínimos quadrados pode levar a resultados incompatíveis. Apresenta com alternativa a tais situações, o ajuste pelo critério de minimização de Chebyshev. Uma análise da conseqüência da presença de erros grosseiros entre os valores das variáveis medidas também é apresentada, discutindo também a possibilidade ou não de se descartar pura e simplesmente uma medida com erro grosseiro.

ROMAGNOLI e STEPHANOPOULOS(1980) considerando que quando a composição de certas correntes são medidas elas o são para todos os componentes, estudaram o problema da retificação de dados numa planta química e desenvolveram algoritmos que permitem classificar as variáveis medidas e não-medidas de um processo bem como estimar os valores das não-medidas e retificar as medidas tanto de processos com restrições lineares quanto de processos com restrições não-lineares. A classificação das variáveis permite reduzir consideravelmente o tamanho do problema. Testes estatísticos são propostos para checar a consistência dos dados do processo permitindo detectar a presença de erros grosseiros nas medidas.

STANLEY e MAH(1981) baseados em conceitos desenvolvidos em trabalhos anteriores, onde tratavam do comportamento dinâmico de processos, desenvolveram, para o caso de sistemas em estado estacionário, os conceitos de observabilidade e redundância. Propõem-se a determinar se o conhecimento da observabilidade permite saber se um dado conjunto de medidas é suficiente para se determinar o comportamento de um sistema. Através de teoremas demonstram em que condições isto é possível. Mostram como a observabilidade local de uma variável está diretamente relacionada à unicidade da solução local para a equação, mostram ainda como a não-observabilidade local leva a uma falha no estimador seja no estado estacionário, seja no estado quase-estacionário. Concluem ainda, que para restrições lineares, as condições para a observabilidade local

são válidas para a observabilidade global e o sistema é decomposto em subsistemas os quais são redundantes.

STANLEY e MAH(1981b) a partir de resultados do trabalho anterior, desenvolveram algoritmos para a classificação quanto à observabilidade e a redundância para variáveis individuais. Mostram que o procedimento utilizado no trabalho anterior, (1981), por estar baseado em testes com o posto de matrizes, é computacionalmente inconveniente e, além do mais, não pode ser utilizado para classificar variáveis individualmente. Alguns conceitos do método gráfico-teórico são discutidos apresentando a terminologia necessária aos algoritmos de classificação. São oferecidas referências para um estudo mais completo da teoria dos grafos a qual segundo os autores é uma ferramenta muito útil de "representação topológica da estrutura dos processos", permitindo a utilização de certos resultados na reconciliação de dados. Uma vez representado o processo com arcos, tendo-se os valores das variáveis medidas e os valores das entalpias, o algoritmo classifica cada valor de temperatura, fluxo de massa e de energia em uma das seguintes categorias: globalmente observáveis, localmente observáveis, localmente não-observáveis, globalmente não-observáveis e finalmente em bloco não-observável. Sendo que um bloco não-observável contém no mínimo uma variável não-observável.

MAH e TAMHANE(1982) desenvolveram um procedimento para detectar a presença de erros grosseiros entre valores medidos das correntes de processo com restrições lineares baseado nos resíduos entre os valores medidos e os valores reconciliados. Os autores também apresentam um teste que mostra o quão "fácil" é a detecção de um erro grosseiro entre um conjunto de medidas dadas. Uma curva de distribuição normal para a potência do teste é apresentada de tal modo que se pode concluir que quanto maior a magnitude do erro grosseiro, mais provavelmente o teste acusará a presença do mesmo. O teste foi desenvolvido inicialmente para se detectar apenas a presença de um único erro grosseiro, contudo, os autores afirmam ser perfeitamente possível se estender o procedimento para a detecção de mais de um erro.

ROMAGNOLI e STEPHANOPOULOS(1983) desenvolveram um procedimento sistemático para manusear um conjunto de restrições (balanços) na resolução do problema de reconciliação de dados. O desenvolvimento é baseado no processamento seqüencial da covariância da estimativa do erro a qual surge a partir da solução do problema dos mínimos quadrados. O modelo é portanto composto de uma parte estatística e uma não-estatística. O trabalho trata da importância do conhecimento da presença de variáveis redundantes e relevância da estimabilidade de dados. As condições gerais para a completa determinabilidade do sistema são definidas utilizando as propriedades de resolução das equações de balanço.

SERTH e HEENAN(1986) utilizando vários métodos de detecção de erros grosseiros já existentes e outros modificados, apresentam um estudo comparativo para a utilização em reconciliação de dados para sistemas de medidas em linhas de vapor (um único componente) de um processo real, com restrições lineares. Os métodos utilizados são o Teste de Medida Iterativo Modificado (MINT) o qual se revelou ao lado do teste de Grade Combinatorial (SC) como os dois melhores para o tipo de processo estudado. O Teste de Medida simples (MT) , MAH e TAMHANE(1982) não apresentou bons resultados. O MINT, o MT e o IMT são três dos testes estatísticos utilizados pelos autores na detecção de erros grosseiros. Outros quatro algoritmos baseados em testes

estatísticos de desbalanços nodais também foram estudados, como exemplo, o já citado SC. Os demais são o Método dos Pseudo-nós (MP), o teste dos Pseudo-nós Modificado (MMP) e o método combinatorial, o qual juntamente com o MMP dá origem ao SC.

KRETSOVALIS e MAH(1987) expandindo o trabalho de STANLEY e MAH(1981a), e utilizando o mesmo procedimento com grafos, faz um desenvolvimento mais geral sobre a observabilidade e redundância utilizando agora sistemas com fluxos multicomponentes e três tipos de variáveis (fluxo de massa total, fluxo de massa dos componentes e frações de massa dos componentes). A mudança aqui fica por conta da forma de se identificar a observabilidade das equações de restrições, de tal modo que se para um subconjunto de equações de restrições existem soluções com respeito a variáveis medidas no subconjunto, diz-se que as variáveis são observáveis. De outro modo, uma variável não-medida é não-observável se todos os subconjuntos de equações nas quais ela ocorre não são solúveis. Os autores expõem a melhor maneira de se derivar estes subconjuntos pois nem todos são de interesse e a determinação combinatorial deles além de tediosa é inútil. Durante o procedimento, alguns dos arcos são descartados simplesmente, outros agrupados de acordo com o tipo de medidas disponíveis no processo. O algoritmo utilizado é dividido em duas fases sendo que a primeira fase é a de pré-tratamento dos dados, e a segunda a fase principal. Existem teoremas os quais servem para pré-classificar as variáveis em cada fase. Tais teoremas servem também para agregar ou deletar os arcos, conforme dito anteriormente. A redundância ou não de uma medida pode ser obtida a partir de sua classificação quanto à observabilidade. Retira-se a medida do conjunto tornando a variável correspondente não-medida e a seguir aplica-se o teste da observabilidade. Se a mesma continua observável, a medida é redundante. Este não é um procedimento muito eficiente, segundo os autores, devido ao fato de ter que se aplicá-lo a cada variável medida o que seria inviável em processos de grandes proporções. Desta forma, os autores apresentam teoremas os quais possibilitam classificar quanto a redundância das variáveis sem precisar classificar quanto a observabilidade. Um novo algoritmo é derivado destes teoremas. Os autores alertam para o fato de que a inclusão de novas variáveis como balanços de energia e/ou reações químicas pode tornar a solução mais completa ao passo que aumenta também o nível de complexidade do mesmo.

NARASIMHAN e MAH(1987) descrevem um procedimento para a identificação de erros grosseiros no qual é possível detectá-lo sabendo de que tipo se trata. Um erro por vazamento é diferenciado de um erro que ocorre porque o sistema sai do estado estacionário ou porque o sistema conta com medidas "viciadas". O método aqui desenvolvido é o de Máxima Verossimilhança Generalizada (GLR) e é aplicável a sistemas em estado estacionário. Como dado adicional, o método oferece a possibilidade de se estimar a magnitude do erro grosseiro, o que é útil para se determinar o impacto do erro grosseiro no processo. Considerando-se os métodos estatísticos apresentados por exemplo por ROMAGNOLI e STEPHANOPOULOS(1980), ROSENBERG et al.(1987), SERTH e HEENAN(1986), os quais eliminam variáveis suspeitas, através de uma estratégia em série, das medidas suspeitas de conterem erros grosseiros, percebe-se que estes métodos não podem ser aplicados, por exemplo a sistemas com vazamento pois o erro não está diretamente relacionado à medida propriamente mas ao processo em si (não tem como ser removido). Pela estratégia aqui proposta, uma compensação em série para os erros grosseiros é feita utilizando-se a técnica GLR. O erro é identificado "a tempo" e compensado utilizando estimativas da magnitude, antes de se identificar os

erros subsequentes. Para uma abordagem diferente do mesmo problema, pode ser consultado o trabalho de BRITT e LUCKE(1973) o qual trata da estimação de parâmetros não-lineares em modelos implícitos. Ao contrário das estratégias anteriores, a estratégia de compensação serial pode ser utilizada para identificar múltiplos erros grosseiros. O método é recomendável em relação aos anteriores pois mostra-se, sozinho, capaz de identificar e classificar os erros grosseiros ainda identificando uma possível saída do processo do estado estacionário. O mecanismo de compensação também foi comparado ao de eliminação serial de SERTH e HEENAN(1986) e os resultados são bastante semelhantes nas condições em que se processaram as simulações levando, contudo, o método de compensação enorme vantagem em termos de tempo computacional.

ROSENBERG et al. (1987) buscando resolver os problemas de conflitos entre os diversos testes estatísticos que atuam de forma separada na detecção de erros grosseiros, propõem dois novos testes compostos DMT e EMT os quais indicam os testes de medidas dinâmico e estendido, respectivamente. Estes dois testes fazem uso de mais de um teste estatístico por vez além de levar em consideração as violações nos limites das condições de contorno dos problemas (restrições). São ainda complementados com a utilização dos Teste Global e do Teste de Medidas. A meta principal é reduzir a freqüência de detecção de erros tipo I (com pelo menos um erro grosseiro). Os testes são originalmente desenvolvidos para sistemas com restrições lineares e se restrições bilineares ou não-lineares de qualquer ordem aparecem, é necessário linearizá-las antes de aplicar os esquemas de detecção. ROSENBERG et al descreve e mostra o impacto dos vários fatores que influenciam o desempenho dos esquemas de detecção dos erros grosseiros como por exemplo as magnitudes das medidas e seus limites físicos; a magnitude dos erros grosseiros (g) e a razão magnitude dos erros e a sua variância (v); restrições, tamanho ,da rede e configuração, e posição das correntes, as quais estão englobadas nas restrições através da matriz de incidência. Com base nestes fatores, os autores concluem que a performance de qualquer esquema de detecção de erros grosseiros depende da razão e da magnitude g/v, a configuração da rede e a posição dos erros entre as medidas. Os esquemas por eles considerados não são apropriados para sistemas com correntes paralelas. Finalmente, mostram que para o tipo de sistemas estudados, os testes propostos se mostram mais confiáveis que os anteriormente propostos.

CROWE(1988) tem como proposta neste trabalho, apresentar um método através do qual possa predizer o efeito de se remover um grupo dado de medidas nos testes estatísticos sem que seja preciso calcular totalmente a reconciliação destas medidas a cada grupo removido. O procedimento foi desenvolvido para sistemas com restrições lineares. O alvo é encontrar um ou mais grupos de medidas que removidas, satisfaçam os testes estatísticos. O Teste Global é um dos testes aqui utilizados, contudo, não garante, sozinho, mesmo que satisfeito, que não existam erros grosseiros entre as medidas testadas. Para fugir desta limitação, CROWE também utiliza os Testes dos Desbalanços e o Teste das Medidas (MAH(1982)), mostrando num algoritmo desenvolvido a ordem de utilização de cada um dos testes apresentados. A meta é encontrar um número mínimo de remoções o qual reduza o valor da função objetivo (utilizada na reconciliação) tal que ainda obedeçam os critérios de restrições empregados: como por exemplo o da não-negatividade dos fluxos e concentrações e também que a soma dos fluxos dos componentes seja igual ao fluxo total na corrente. O

algoritmo prevê que um valor é removido isoladamente e assim se processa o cálculo da função objetivo. Grupos de dois, três (e assim por diante, enquanto possível) são seqüencialmente testados pelo mesmo caminho até que se encontre a melhor seqüência, que removida, produza a reconciliação mais satisfatória entre as medidas remanescentes ou quando o número máximo de remoções for atingido. O autor diz não ter comparado este teste para medir suas performance frente a outros testes existentes.
PAI e FISHER(1988) aplicaram o método de BROYDEN(1965) para fazer a reconciliação de dados com restrição não-linear. Compararam os métodos utilizados por BRITT e LEUCKE(1973) e por STEPHENSON e SHEWCHUK(1986) com os aqui desenvolvidos apresentando como inconvenientes àqueles, o fato de que problema pode se tornar indevidamente grande devido à utilização dos multiplicadores de Lagrange e também o fato de as derivadas das restrições aparecerem nas equações normais, necessitando assim serem calculadas a cada iteração. O ponto de partida deste trabalho é o esquema de cálculo da reconciliação proposto por CROWE(1983), utilizando a projeção matricial. Aqui, os autores criticam os pontos fracos do método de CROWE apontando para o fato de que o método infelizmente ser muito específico para ser útil em casos mais gerais. Mostra que a metodologia proposta por KNEPPER e GORMAN(1980) diminui o esforço computacional mas é limitado para ser utilizado para ser utilizado em problemas superdeterminados. Deste modo, o método de BROYDEN(1965) é proposto para substituir as derivadas antigas tal que a taxa de convergência possa ser melhorada sem que seja necessário repetidamente avaliar as derivadas. PAI e FISHER concluem que pelo esquema iterativo, por eles proposto, em muitos casos uma avaliação completa da matriz das restrições é suficiente para o algoritmo alcançar boa solução, sem contar o fato de que a convergência é mais rápida quando uma direção de convergência constante é adotada.

BROYDEN(1965) desenvolveu um método de atualização da matriz Jacobiana de tal forma a transpor as desvantagens do método de Newton para a solução de sistemas de equações. A idéia atrás do método é a de usar o resíduo da função no novo ponto para a partir daí calcular uma nova aproximação do Jacobiano, tomando como base a matriz antiga, de tal modo que o cálculo repetido da matriz possa ser evitado.

MAH(1987) desenvolve o procedimento para a reconciliação de dados de processos através da teoria dos grafos. Utiliza os resultados apresentados em trabalho conjunto com STANLEY(1976) onde descrevem a teoria. Segundo os resultados básicos desta teoria, o problema de reconciliação de dados pode ser decomposto da seguinte maneira: 1-reconciliação com falta de medidas, pode se resolvida em subproblemas separados sendo o primeiro por um processo dígrafo modificado formado por criteriosa agregação de nós, unidos por arcos de fluxos não medidos, e depois pela estimação de fluxos não medidos nas árvores de arcos do processo dígrafo; 2-Medidas de fluxos não conhecidas, podem ser determinadas unicamente se e somente se os arcos não medidos, sem consideração de suas direções, formam um subgrafo acíclico. Estudando três casos representados por um dígrafo de processo gerado para uma secção atmosférica de uma unidade de destilação de óleo cru, contendo 61 correntes e 32 nós, os autores mostram que há uma significativa melhora global na precisão das medidas em cada caso, mas que a extensão da melhora é diferente para cada uma das situações estudadas. A redução no número de erros grandes é acompanhado por um acréscimo no número de pequenos erros, o que indica que os pequenos erros constituem a maior parcela do erro incidente após a reconciliação. Os autores remetem ao trabalho de KRETSOVALIS e MAH(1987)

para aplicarem estas conseqüências em termos de redundância das medidas. MAH trata da reconciliação de dados para processos com restrições lineares bem como para restrições não-lineares. Um procedimento para identificação de erros grosseiros é também apresentado para complementar o trabalho, afirmando que a introdução do teste de medidas(estatístico) deveria ser capaz de eliminar os problemas causados pela presença de erros grosseiros entre as medidas. São identificadas três situações onde o teste pode falhar: 1-erros grosseiros de grande magnitude podem causar diversos desvios nos resultados dos testes de medidas; 2-a interação de dois ou mais erros grosseiros pode obscurecer a verdadeira localização dos erros; 3-um erro grosseiro pode tornar um modelo melhor que os valores reais dificultando a sua localização.

KRETSOVALIS e MAH(1988a,b) apresentam nestes dois trabalhos, uma complementação do trabalho anterior, aqui discutindo a respeito da inclusão de reações químicas e de balanços de energia nos processos analisados. No primeiro, desenvolvem a base teórica apresentando os teoremas necessários à compreensão do problema e no segundo, elaboram algoritmos os quais permitem efetivamente classificar as variáveis de um sistema complexo. São apresentados os dois caminhos os quais poderiam se seguidos a fim de incluir os parâmetros de análise que estavam faltando. Um dos possíveis caminhos é aquele no qual poderia se tratar sistemas multicomponentes com balanços de energia baseando-se em algoritmos combinados para massa e energia conforme os apresentados por STANLEY e MAH(1981a); uma segunda possibilidade seria a extensão do algoritmo para sistemas multicomponentes incluindo balanços de energia tratando, agora, a temperatura como um componente adicional entre as variáveis medidas do processo. Em ambos os casos, existem complicadores, seja por causa da base teórica em que estão definidos os algoritmos ou devido à presença de fluxos de energia "pura" em unidades de troca térmica.Os dois algoritmos desenvolvidos são o GENOBS (GEneralized process Networks OBServability algoritm) e GENERED (GEneralized process NEtworks REDundancy algoritm) onde incluem a análise de sistemas nos quais ocorrem reações químicas e divisores de fluxo. Para efeito de classificação de observabilidade (e redundância), a observabilidade foi definida em termos de solvabilidade das equações de restrição. Tal procedimento permite a classificação de cada variável individualmente.

LAURENCE(1989) analisa o problema da inconsistência de dados, que precisam ser reconciliados, numa planta. A análise é feita sob uma ótica econômica destacando-se as vantagens, em termos de custo, em se ter dados mais confiáveis. Descreve a economia que é possível se conseguir com pequenas melhoras nos resultados medidos na planta quando se tem processos trabalhando com grandes escalas de produtos. O autor aponta ainda para a possibilidade de se detectar falhas em determinados equipamentos de medidas que passem a apresentar erros de modo sistemático. A abordagem aqui é um pouco diferente das dos demais autores no que diz respeito às considerações das restrições. Parte de considerações simples e faz a reconciliação esperando que os resultados apontem ou não onde as considerações precisam ser mais rigorosamente consideradas. Traça considerações a respeito da resolução de problemas de reconciliação utilizando técnicas de regressão linear versus técnicas de programação linear. Um projeto de implementação da reconciliação é descrito como apresentrado abaixo: 1-exame da instrumentação, para se saber se os instrumentos são suficientes para descrever os fluxos do processo; 2-extensão de medidas adicionais, a fim de se conseguir redundância ou no mínimo observabilidade em linhas de processo chave, (a análise anterior permite fazer

esta extensão de maneira criteriosa sem que seja preciso a instalação desnecessária de equipamentos de medidas que não acrescentam informações relevantes ao problema); 3-benefícios do exame, forçar a observância dos balanços, dar uma indicação da qualidade das medidas e mais importante de todos, justificar a presença de cada instrumento no contexto da resolução das equações; 4-histórico do balanço de massa, tomando-se dados médios observados nos instrumentos de medida. Alerta para o problema da qualidade diferente de medidas para diferentes equipamentos devido às características de construção e operação de cada um; 5-finalmente, reconciliação on-line, com o objetivo de reduzir drasticamente o trabalho de "alimentação" do sistema de controle de uma planta complexa com muitas saídas de dados.

CROWE(1989a) utilizando os resultados derivados por MAH e TAMHANE(1982) para o Teste de Máxima Potência (MP), o qual segundo o autor se mostra mais útil para se encontrar um erro particular que outros métodos baseados na combinação de outras medidas, estendeu o conceito de máxima potência para o teste das restrições. O teste prevê que se o resultado do teste estatístico de Máxima Potência (MP) é muito grande, é possível que o balanço possua um valor de fluxo não esperado (possível fonte de erro grosseiro) ou um acúmulo. A remoção daquele balanço é equivalente a se adicionar um fluxo não-medido ao conjunto de restrições, conforme descrição de CROWE(1988), onde foi mostrado que a reconciliação após a remoção de uma medida poderia ser conduzida a partir do caso base da reconciliação proposta. O teste é limitado a casos lineares e é assumido que as taxas de fluxos das espécies não-medidas já tenham sido removidas.

CROWE(1989b) ao contrário de autores anteriores, não emprega a teoria dos grafos para resolver o problema da observabilidade e redundância de variáveis e medidas. O desenvolvimento de seu trabalho se baseia na projeção matricial. Partindo das idéias desenvolvidas por VACLAVEK et al.(1987a,b), são propostos algoritmos os quais procuram evitar as principais limitações dos métodos iniciais como por exemplo o método de ROMAGNOLI e STEPHANOPOULOS(1980). Comparativamente ao método de KRETSOVALIS et al.(1988a,b), o de CROWE não requer a construção de diferentes grupos de subgrafos e também a aplicação repetitiva de teoremas para a observabilidade e redundância. O autor assim propõe, a partir da reconciliação de dados efetuada, CROWE(1983,1986) utilizar elementos já determinados para conseguir classificar fluxos medidos, concentrações ou entalpias quanto à redundância e quantidades não-medidas quanto à observabilidade. O algoritmo utiliza propriedades algébricas lineares das matrizes envolvidas. A inclusão de reações químicas, divisores de fluxos e fluxos "puros" de energia são permitidos. Não se faz nenhuma consideração a respeito da estrutura das restrições além do fato de que as quantidades não medidas sejam lineares.

NARASIMHAN e MAH(1989) segundo e estudo realizado em (1987), propõem-se , agora, a transformar modelos gerais em estados estacionários em modelos simplificados em estado estacionário. São considerados casos em que as variáveis de processo não são diretamente medidas e nas quais as variáveis não-medidas estão presentes nas restrições. O procedimento permite utilizar testes estatísticos previamente desenvolvidos sem a necessidade de novas adaptações para cada caso particular.

ALMASY(1990) desenvolveu os princípios da reconciliação de dados utilizando processos dinâmicos. Ao contrário de processos estáticos, onde apenas o conhecimento de variáveis como taxas de fluxo, reações químicas ou energia é necessário, aqui se leva em consideração a correlação de tempo dos dados. ALMASY mostra que esta é a diferença essencial entre processos estáticos e dinâmicos. Contudo, somente onde se conhece totalmente a dinâmica do processo é possível se aplicar as técnicas aqui citadas — simplificações não garantem perfeito funcionamento do método — e mesmo assim, o modelo só é suficientemente rápido em sistemas com restrições lineares. O autor contudo alerta para o fato de se negligenciar não linearidades o que pode comprometer enormemente as estimativas. A utilização de modelos muito simplificados pode levar a resultados reconciliados piores que os medidos. A solução foi formulada e resolvida com a consideração de um intervalo discreto de tempo.

DAROUACH e ZASADZINSKI(1991) apresentam um algoritmo para determinar sistemas com equações de balanço material dinâmicos onde a representação espacial do estado padrão e o filtro de Kalman não podem ser usados. O modelo considerado é linear e determinístico com todas as variáveis medidas (entradas, saídas e as que mudam). O modelo é descrito em equações de diferenças discretas onde existem mais variáveis que restrições. São, portanto, chamadas equações singulares. Equações diferenciais algébricas estão presentes nestes processos. A formulação usada para desenvolver o novo método para resolução dos problemas de reconciliação obtém estimativas recursivas que incluem a filtragem de dados e o "alisamento" das curvas, representando assim, uma aproximação sistemática do processo em tempo real. A convergência do método foi comprovada para sistemas singulares.

HODOUIN et al. (1991) descrevem que a utilização de programas computacionais comerciais, não específicos, no processamento de dados de sistemas onde se faz a aquisição de dados através de sensores diretos da planta (on-line) ou através de análises laboratoriais muitas vezes são dificultados porque os métodos empregados nestes programas levam em consideração que o problema tenha sempre solução factível, isto é, a estrutura do conjunto de dados é compatível com os requerimentos implícitos no método utilizado. Tais requerimentos engloba a necessidade de que as variáveis medidas no processo formem uma base suficientemente completa para que o cálculo de variáveis não medidas seja estável, ou seja, é necessário que haja redundância entre as variáveis. Os autores destacam, entretanto, que apenas o conceito de redundância é bastante vago e que não descreve o "status" de cada variável no problema. Diante disso se propõem a desenvolver um método que encontre a classificação das variáveis de qualquer processo complexo em quatro categorias por eles assim definidas: 1-variáveis não medidas que podem ser classificadas usando os modelos de conservação de massa e os valores das variáveis não medidas — calculáveis; 2-variáveis medidas que não podem ser calculadas a partir de dados disponíveis e das correspondentes equações de balanços de massa — incalculáveis; 3-variáveis medidas as quais possam ser otimamente modificadas de modo a forçá-las a seguir o modelo de conservação de massa — ajustável ou redundante; 4-variáveis medidas com valores que não podem ser melhoradas pelo programa — não-ajustável ou não-redundante. Destacam ainda que a classificação das variáveis do processo do modo acima pode trazer grandes vantagens, entre as quais citam que a análise de redundância permite selecionar exatamente as variáveis que precisarão ser medidas, obtendo-se assim as informações desejadas a um menor custo; além do mais, mostram que a classificação quanto à redundância é um pré-requisito para qualquer

cálculo de balanços de massa usando grandes conjuntos de dados, os quais não são obviamente redundantes para todos os aspectos de computação. Finalmente, mostram que após esta classificação, é possível reduzir o problema de balanço de massa à sua região factível. O método utilizado está baseado na linearização do conjunto de equações, utilizando decomposições hierarquizadas para reduzir o trabalho computacional.

KIM et al.(1991) estendeu o método de erros-em-variáveis (EVM) para processos dinâmicos não-lineares, em procedimentos on-off-line. Para este novo método, (NDEVM), os erros nas medidas em todas as variáveis são "tratados" e a reconciliação dos dados medidos e a estimativa dos não-medidos é feita. Duas maneiras para se resolver o problema geral do NDEVM são propostas. A primeira consiste em uma solução e otimização seqüencial e a segunda numa solução e otimização simultânea. O método simultâneo resulta em um problema de NLP (programação não-linear) relativamente grande e só é recomendável de ser usado, segundo o autor, quando realmente necessário. O método seqüencial é eficiente e mais direto para ser implementado além da vantagem de ter controles bem-desenvolvidos para erros (neste caso uma possível técnica a se utilizar é a colocação ortogonal). Três outros métodos foram aqui implementados com o intuito de se comparar os resultados aos do NDEVM. São dois métodos de mínimos quadrados modificados e um método utilizando equações discretizadas. Em situações em que somente erros randômicos estavam presentes, os métodos usados para comparação e o NDEVM apresentaram resultados semelhantes; quando nas medidas simuladas haviam erros sistemáticos (mesmo não grosseiros, como por exemplo erros de bias) o NDEVM apresentava estimativas muito mais eficientes. Com relação à metodologia de resolução do NDEVM por integração ou por colocação ortogonal, aquela apresenta-se muito mais rápida que este apesar de em termos numéricos, os resultados serem bastante parecidos.

KELLER et al.(1991) propõem um método indireto para estimar as variâncias das medidas dos erros as quais em muitas situações práticas não são conhecidas ou são apenas aproximadamente conhecidas. O método se baseia em relação deduzida a partir de propriedades estatísticas dos balanços de massa das restrições. É, segundo o autor, a maneira mais robusta com respeito às medidas correlacionadas e mais apropriado em situações práticas que os métodos desenvolvidos por alguns autores aqui anteriormente citados. Diversas simulações foram apresentadas comprovando a tese.

TJOA e BIEGLER(1991) empregam estratégias simultâneas para reconciliar dados de processo. Ao contrário de CROWE(1983), o qual utilizou a projeção matricial para resolver sistemas com restrições lineares estendendo depois para o caso de restrições bilineares, se dedicam a desenvolver um procedimento o qual permite resolver problemas com restrições não-lineares gerais. A alternativa aqui utilizada é um teste baseado na função distribuição bivariada apresentada em FARISS e LAW(1979) derivada a partir dos princípios de máxima verossimilhança. É apresentada uma descrição dos métodos baseados na técnica de Gauss-Newton para a reconciliação comparativamente ao método SQP desenvolvido pelo autor para reconciliação de dados cujas funções objetivos estão definidas em mínimos quadrados. O autor chama a atenção para o fato de que em problemas de reconciliação de dados com restrições não-lineares muitos testes de detecção de erros grosseiros são derivados baseando-se em modelos linearizados, o que é questionável, e ainda requer muitos estudos. Para evitar isto, o autor usa uma função

objetivo que é construída usando os princípios de máxima probabilidade em uma função distribuição combinada, ou seja, a função leva em conta as contribuições dos erros grosseiros e randômicos e por isso é também conhecida como função gaussiana contaminada, discutida por VERDINELLI e WASSERMAN(1990). As vantagens apontadas são que na minimização da função objetivo bivariadas são levadas em conta as contribuições causadas pela presença de erros grosseiros, o que introduz erros nas estimativas e ainda permite simultaneamente aplicar um teste de detecção de erros grosseiros baseado na distribuição das funções.

CROWE(1992) retoma o trabalho começado em 1989(a) onde estendeu o teste de máxima potência de MAH(1982) para o caso de sistemas com restrições lineares. Aqui, o teste de máxima potência é também aplicado ao caso com restrições bilineares. Algumas questões, antes que seja feita a remoção de uma restrição dentre as originais, surge aqui, como por exemplo, saber se o teste de Máxima Potência estatístico pode ser um critério válido em alto nível de confiança. O autor diz que felizmente a adaptação do caso estudado em CROWE(1988) é fácil. A redução na função objetivo a qual resulta da remoção de uma restrição original é precisamente igual ao quadrado da correspondente MP estatística. Segundo o autor, quando os testes estatísticos para medidas ou restrições são idênticos, seus efeitos nos dados são estatisticamente confundidos e os testes não oferecem meios para se decidir, entre os dados, quais os possíveis detentores de erros grosseiros. Os testes apresentados no trabalho, ilustram a utilidade dos testes de MP sobre as medidas e sobre as restrições originais. O teste provê as informações necessárias para se decidir entre a localização dos erros grosseiros, se entre as medidas ou se entre os balanços.

LIEBMAN et al. (1992) traçam uma escala de desenvolvimento dos métodos apresentados até então para se proceder à reconciliação de dados com restrições lineares e não-lineares em sistemas dinâmicos e estáticos. Mostram as desvantagens de cada método no que diz respeito às restrições possíveis de serem utilizadas e se propõem a desenvolver um método que se mostre eficiente tanto na computação de casos lineares quanto não-lineares utilizando sistemas em condições dinâmicas. A partir do trabalho de LIEBMAN e EDGAR(1988), o qual demonstra as vantagens de se utilizar a programação não-linear, PNL, para resolver os problemas de reconciliação de dados em estado estacionário, os autores, incluindo variáveis de contorno desenvolvem um eficiente algoritmo utilizando PNL para conseguir a reconciliação para processos dinâmicos. O problema geral de reconciliação de dados envolve a otimização de uma função objetivo restringida por igualdades algébricas, diferenciais e também inigualdades para a utilização em problemas nos quais o filtro de Kalman não é apropriado, tais como, processos operando em regiões fortemente não-lineares, as quais envolvem restrições de inigualdade e funções objetivo que não estão baseadas nos mínimos quadrados. As principais desvantagens do método proposto são que um modelo dinâmico deve ser usado e ainda há um enorme aumento no esforço computacional o que faz com que a utilização de matrizes esparsas seja extremamente importante para a eficiência da otimização.

MEYER et al. (1993) utilizando trabalhos anteriores ao deles, comparam as técnicas de resolução da reconciliação de dados como a projeção matricial e a teoria dos grafos as quais elegem como as mais rigorosas, com a estrutura de matrizes, apresentada por KALITVENTZEFF(1987) considerada das menos rigorosas. Chegam à conclusão que a

teoria dos grafos é mais eficiente quando se trata de grandes processos envolvendo restrições não-lineares. Citam para tanto o trabalho de HODOUIN(1991). A partir disto, desenvolvem dois algoritmos rápidos e seguros, baseados no conceito de teoria dos grafos os quais são conduzidos a partir da classificação quanto à observabilidade e a redundância em redes de processos multicomponentes com ou sem reações químicas. A principal característica do método apresentado é a classificação das variáveis, o que permite uma formulação apropriada do problema, possibilitando um decréscimo no número de variáveis a serem otimizadas.

RAMAMURTHI et al.(1993) mostram que as estratégias de controle de processos não-linearizados, desenvolvidas recentemente, requerem informações de estado do processo para computar as mudanças no controle, e tais informações devem obrigatoriamente ser precisas. O autor propõe um modelo para reconciliação dinâmica de dados utilizando o estimador SLHE, sigla em inglês para Estimador Horizontal Sucessivamente Linearizado, em sistemas de ciclo fechado, ilustrando a importância da reconciliação de dados para um perfeito funcionamento do controlador. É feita uma comparação entre o SLHE, o EKF (Filtro de Kalman Estendido) e as técnicas de programação não-linear (NPL). Tomando, como parâmetros, trabalhos de outros autores, anteriores a este, chegam a conclusão que o SLHE tem desempenho, em termos de precisão, comparável ao NPL, mas este perde em termos de tempo computacional. No entanto ambos, SLHE e NPL, são superiores ao EKF, o que já havia sido descrito por LIEBMAN(1992). A única vantagem comparativa deste último é o tempo negligenciável de computação. Como última parte da proposição do trabalho, os autores implementam os dados obtidos segundo o estimador (SLHE) em um circuito de malha fechada provido de um controlador preditivo não-linear (NLPC), comparam com os resultados obtidos quando se usa o estimador NPL e encontram resultados bastante próximos.

NARASIMHAN e HARIKUMAR(1993 a,b) analisam as desvantagens de métodos anteriores mostrando que a falta de restrições como por exemplo, quanto à não-negatividade dos fluxos, pode em alguns casos, levar os resultados da reconciliação de dados a apresentar valores sem significado físico. O fato de os métodos conhecidos utilizarem como regra, à retificação de dados, a retirada dos valores com probabilidade de apresentarem erros grosseiros, inviabiliza, em muitos casos, a estimativa dos valores medidos apenas com os valores remanescentes. Baseados nestes aspectos, os autores desenvolvem um procedimento para incorporar limites às variáveis do processo no problema, utilizando um algoritmo de programação quadrática baseado no conjunto de restrições ativas de Wolfe, desenvolvida em DANTZIG(1963). E mais importante, obtém as distribuições estatísticas dos resíduos das restrições e dos resíduos das medidas, propondo dois métodos, utilizando estas distribuições, para detecção de erros grosseiros. Na detecção dos erros grosseiros são utilizados duas metodologias as quais utilizam o método da razão de Máxima Probabilidade Generalizada (GLR), apresentado em NARASIMHAN e MAH(1987), onde primeiro a detecção de erros é feita (antes da reconciliação) e por isso as limitações não têm efeito. No segundo procedimento, a reconciliação é feita primeiro a fim de se incluir as limitações e em seguida é novamente utilizado o GLR. Para cada um dos procedimentos, o autor renomeia o método GLR, sendo no primeiro caso UGLR(sem limitações) e no segundo BGLR(com limitações). Segundo simulações de NARASIMHAN et al, ambos os procedimentos do método apresentaram resultados mais reais que os conseguidos com os métodos EMT e MINT, desenvolvidos por MAH e TAMHANE(1982) os quais eram até então considerados os

mais eficientes na detecção de erros grosseiros. Apesar de terem sido testados para caso lineares, podem ser estendidos para restrições não-lineares.

ALDRICH e DEVENTER (1994) apresentam um método utilizando redes neurais para classificar erros de medidas em processo sem fazer limitações quanto às restrições do processo. Os autores expõem o fato de os melhores testes estatísticos de determinação de erros grosseiros até aqui desenvolvidos serem incômodos para se utilizar e muitas vezes ineficientes, especialmente quando se analisa grandes e complexos sistemas. Citam por exemplo os trabalhos de CROWE(1983), ROMAGNOLI e STEPHANOPOULOS(1980) e o de SERTH e HEENAN(1986), e aponta como desvantagens dos mesmos, o fato de não distinguirem entre os tipos de erros grosseiros. Esta falha foi corrigida no trabalho de NARASIMHAN e MAH(1989) onde é proposto o teste de Razão de Máxima Verossimilhança (GLR). Para os casos com restrições não-lineares são apenas parcialmente úteis e, geralmente, não capturam as características reais dos processos. Aqui, ALDRICH et al., mostra que o procedimento com redes neurais tem grandes possibilidades de detectar erros grosseiros e uma capacidade sem paralelo entre os métodos passados para detectar erros em medidas cujos sistemas tenham restrições não-lineares, o que pode ser comprovado com a utilização de testes estatísticos tradicionais juntamente com os propostos utilizando-se redes neurais. Para a utilização desta última técnica, não é necessário o conhecimento explícito da distribuição randômica dos erros nos valores das medidas. A última vantagem do método é a fácil implementação, segundo o autor, em sistemas que já estejam operando.

KARJALA e HIMMELBLAU(1994) apresentam uma forma diferente das tradicionais para conseguirem a reconciliação de dados de um processo dinâmico. Na verdade, tratam o problema não apenas como reconciliação mas como retificação de dados pois esta inclui uma filtragem e/ou predição e/ou hipóteses de teste para as variáveis. O foco do problema aqui é centrado na remoção dos erros grosseiros e não no ajuste de dados a cada tempo. As técnicas utilizadas para tanto são: 1-utilização de redes neurais artificiais; 2-filtro de Kalman estendido; 3-programação não-linear restringida. Para descrever a utilização destes métodos, os autores mostram que a principal atração em se utilizar a rede neural está no potencial para se resolver rápida e facilmente problemas em sistemas lineares unicamente com base no histórico dos dados do processo e ao contrário dos métodos tradicionais, esta técnica não utiliza os valores medidos para estimar os valores retificados. O filtro de Kalman, requer o conhecimento anterior das matrizes covariância apesar de ser computacionalmente eficiente e poder ser implementado de maneira on-line no processo. No caso do método de retificação dinâmica pelo NPL, o tempo computacional pode ser muito significante e de modo geral é bem superior ao requerido pelo filtro de Kalman. Depois de comparar os três métodos, KARJALA et al mostra que o filtro de Kalman e o NPL são técnicas que indubitavelmente produzem estimativas melhores que os valores medidos na planta tendo como maiores inconvenientes a necessidade de conhecimento da matriz covariância. No caso da utilização de rede neural, esta exigência não precisa ser satisfeita, fato que em processos reais nem sempre é possível satisfazer. A facilidade matemática de implementação do método por redes neurais é também invocado em comparação a outros métodos.

BOSSEN et al. (1994) utilizam estratégias que permitem a solução de problemas relacionados à simulação, otimização e reconciliação de dados de processos industriais químicos utilizando o pacote computacional GHEMB desenvolvido por

CHRISTIANSEN(1992). O processo estudado é o da síntese de amônia. A reconciliação de dados é conseguida utilizando-se o algoritmo SQP baseado na descrição de HIMMELBLAU(1988), segundo os autores. Criaram uma interface entre o algoritmo SQP e o GHEMB tal que os problemas pudessem ser resolvidos. A principal meta da utilização da reconciliação de dados neste processo era a de se determinar a atividade de catalisadores no reator e consequentemente se avaliar quando seria necessária a sua troca. Os resultados mostraram a utilidade da estratégia com modelos de simulação rigorosa diretamente incorporado à otimização.

BUSSANI et al.(1995) apresentam o desenvolvimento de um pacote o qual aplicam à reconciliação de dados de uma planta de hidrogênio, onde se faz a otimização dos procedimentos no sistema. O pacote utilizado é o ORO(*On-line Reconciliation and Optimization*) o qual adota a técnica de seqüência modular desenvolvida por BIEGLER(1983) e ainda permite a descoberta dos erros principais na planta. O problema de otimização é resolvido tratando-se o sistema como se fosse uma "Black Box".

TONG e CROWE (1995) propõem a utilização de uma "nova" ferramenta na determinação de erros grosseiros entre variáveis de processo, prometendo cobrir alguns pontos falhos não possíveis de serem resolvidos com os métodos estatísticos, até aqui apresentados. A ferramenta é a Análise de Componente Principal (PCA), a qual transforma um grupo de variáveis correlacionadas em não-correlacionadas, conhecidas como Componentes Principais (PC). Os testes realizados foram comparados aos resultados obtidos com o teste de Máxima Potência (MP) e com os Testes Globais (GT) apresentando resultados não melhores que o MP apenas em condições de limites não-reais. A utilização de estatística de PC é útil no controle de variáveis com erros tipo I porque remove a correlação entre as variáveis. O estudo mostrou que o método é útil também para detectar erros grosseiros de pequena dimensão em faixas onde os demais testes estatísticos falham.

ZHANG et al. (1995) utilizaram o pacote comercial ASPEN PLUS para proceder à otimização e à estimação de parâmetros numa planta de ácido sulfúrico da MONSANTO utilizando interfaces em FORTRAN capazes de processar dados on-line. Para fazer a detecção de erros grosseiros e a reconciliação de dados utilizaram a linguagem de simulação-otimização GAMS/MINOS utilizando como método de modelagem as técnicas desenvolvidas por TJOA e BIEGLER(1991). Segundo os autores, o lucro teve um aumento de 9% após a adoção do procedimento.

HEYEN et al. (1996) descrevem a dificuldade de interpretação dos dados resultantes dos programas de reconciliação de dados, chamando a atenção para o fato de quando acreditar que uma estimativa é de fato melhor que a medida, ou seja, quanto uma dada medida vai influenciar nas demais variáveis. Segundo os autores, um procedimento que assegure a precisão dos resultados leva a um melhor conhecimento das variáveis chave do processo. Mostram que regras para solucionar pontos de medidas ótimas podem ser derivados a partir da matriz sensitividade das equações do modelo usadas como restrições no problema de reconciliação de dados. A análise de sensitividade foi usada na reconciliação dos dados de uma planta industrial para separar as medidas e classificá-las de acordo com o critério explícito de importância. Alguns valores são absolutamente necessários para identificar o estado do processo: eles não podem ser confirmados por

cálculos e portanto seus valores devem ser cuidadosamente checados. Sensores extra precisam ser adicionados às localizações de alta sensitividade. Outras medidas são reconciliadas e por fim outras determinadas pelas equações do método. A grande vantagem desta classificação é o fato de melhorar o conhecimento do sistema permitindo selecionar melhor a localização dos sensores permitindo uma maior confiança nas estimativas das variáveis e uma diminuição nos custos dos investimentos.

SCHRAA e CROWE(1996) propõem uma nova maneira de resolver os problemas de reconciliação de dados com restrição bilinear conforme estudado por CROWE(1986), onde empregou a projeção matricial conforme descrito nesta dissertação , no capítulo 4. A razão do trabalho de SCHRAA é encontrar um método de solução robusto para o problema bilinear, resolvendo assim os problemas os quais apresentam dificuldade através do método da projeção matricial empregando uma formulação não restrita da função objetivo original e suas condições de contorno. Os métodos de otimização utilizados para resolver a forma não restrita dos problemas bilineares são baseados no uso de derivadas analíticas, são iterativos e alternam-se na escolha da direção de busca e na determinação de quão distante da solução está a direção. São conduzidos de modo a se encontrar um mínimo local e são baseados na aproximação da função objetivo com um modelo quadrático. A direção de busca é escolhida de modo a determinar o mínimo local. Aqui são usados os métodos de Newton, o BFGS e o de Gauss-Newton. Cada um deles se diferencia na maneira como aproximam a matriz hessiana das derivadas segundas usadas no modelo quadrático. Newton utiliza a matriz hessiana cheia, enquanto os métodos BFGS e de Gauss-Newton aproximam a matriz hessiana utilizando informações da derivada primeira. A qualidade dos resultados obtidos por estes caminhos é comparada à obtida pela projeção matricial e também à obtida pelo MINOS/GAMS, o qual utiliza um algoritmo lagrangeano. A robustez dos métodos pôde ser testada para cada ponto de partida testado, em termos do número de iterações necessárias, do números total de pontos flutuantes e do tempo de computação requerido. De modo geral, o método de Gauss-Newton foi o mais eficiente pois para várias tentativas ele apresentou o mais rápido tempo de execução e o menor número de operações em ponto flutuante. Foi observado que em todos os casos em que houve remoção de medidas e onde os problemas permaneciam possíveis de serem resolvidos, o valor da função objetivo foi sempre próximo ao valor predito. Apesar da praticidade dos métodos, houve situações em que algumas medidas eram removidas e a solução não continuava possível de ser encontrada. A resolução deste tipo de problema, que acontecia quando se utilizava a projeção matricial, era uma das metas deste trabalho, mas contudo, não pôde ser solucionado da maneira como se previa inicialmente. Estes casos são ditos degenerados e não apresentam soluções bem definidas. A conclusão a que chegaram os autores é de que apesar de serem robustos, os métodos não resolvem os problemas que a projeção não resolve e ainda em alguns casos, o tempo computacional é menor com a utilização da projeção matricial. Os métodos matemáticos utilizados estão descritos em DENNIS e SCHNABEL(1983).

TONG e CROWE(1996) apresentam um modelo para utilização do teste seqüencial para o principal componente, para identificar e principalmente remover erros grosseiros persistentes em reconciliação de dados, combinando para tanto, a Análise do Componente Principal com a Análise Seqüencial. O método descreve a determinação do principal componente e mostra como encontrá-lo. A análise seqüencial é descrita como um método de inferência estatística onde o número de observações requeridas não é

conhecido de antemão mas depende das observações as quais são feitas. A vantagem é que o método depende de menos observações que outros baseados num número fixo de observações. O teste seqüencial aqui aplicado é conhecido como SPRT, ou seja, Teste Seqüencial de Razão de Probabilidade, proposto inicialmente por WALD(1947), o qual é aplicado aos componentes principais. Depois de detectado, o erro grosseiro pode ser identificado utilizando o método de Análise de Contribuição, TONG e CROWE(1995). O procedimento é derivado para reconciliação de dados em estado estacionário.

3 RECONCILIAÇÃO DE DADOS DE PROCESSO- CASO LINEAR

3.1 INTRODUÇÃO

Conforme pode ser comprovado no capítulo 2 existem várias formas e enfoques os quais podem ser dados ao tratamento de dados de processos. O procedimento que será adotado neste trabalho é o que utiliza a projeção matricial na resolução das equações do sistema de otimização e tem como ponto de partida o trabalho de CROWE(1983) conforme o descrito no capítulo 2. O desenvolvimento das equações e das ferramentas teóricas são apresentadas dentro do próprio texto e quando são de caráter mais genérico, ou seja, que serão utilizadas em outras partes deste trabalho, estarão descritas nos apêndices.

O valor das variáveis medidas numa planta contém sempre erros, por melhores que sejam os equipamentos de medidas e os processos de amostragens.

Seja $\tilde{x}_j$ o valor medido de uma variável x_j na corrente j. Define-se erro, como a diferença

$$\mathbf{e_j} = \tilde{x}_j - x_j$$

o valor de x_j é apenas teórico; é na prática o valor que seria medido se não houvesse erros.

Fazendo-se a reconciliação destes dados medidos, tem-se o ajuste dos dados, assim definido:

$$\mathbf{a_j} = \hat{x}_j - \tilde{x}_j$$

é portanto, a diferença entre o valor reconciliado e o valor medido.

3.2 DESCRIÇÃO DO PROBLEMA

3.2.1 A RECONCILIAÇÃO

Reconciliar é estimar valores de variáveis medidas (e algumas não medidas), em determinado sistema , tendo em vista diminuir o erro destas variáveis, ou seja, diminuir a diferença entre o valor real (teórico) e o valor medido. O valor reconciliado é, portanto, o valor a ser utilizado no controle do processo. Não é necessariamente o valor real da variável mas é aquele que diminui as flutuações no controle, mantendo o processo com um comportamento mais estável.

Para se conduzir a reconciliação de dados é preciso observar uma seqüência ordenada de procedimentos.

A- Estruturas do processo

A obtenção de informações a respeito das estruturas do processo e das medidas das variáveis envolvidas (fluxos e concentrações — outros dados como por exemplo temperatura e entalpia podem ser agrupados). O conhecimento da estrutura é

fundamental pois é esta estrutura que fornecerá informações a respeito das ligações entre as correntes e os equipamentos. Os equipamentos não precisam, aqui, ser descritos, portanto, funcionam como caixas-pretas não tendo relevância qual o tipo do mesmo nem as particularidades acerca do seu funcionamento. Só interessam as correntes que dele entram e saem. Do mesmo modo, nos instrumentos de medidas somente interessam suas faixas de precisão, pois do funcionamento de um instrumento de medida depende a possibilidade ou não de se reconciliar valores medidos.

Fundamental também é se conhecer os valores das variâncias das variáveis medidas. É este conjunto de medidas que possibilitará a construção do "histórico" do sistema, permitindo diminuir o erro das medidas quando se faz a estimativa dos mesmos. É preciso, assim, uma série de medidas de cada variável para se chegar a estes valores das variâncias. Quanto maior o conjunto, mais confiável será a base de cálculo. Neste trabalho será utilizado o método descrito em MADRON et al.(1977).

Dado um processo complexo qualquer, pode-se representá-lo através de um fluxograma simplificado contendo a totalidade de correntes e equipamentos ou apenas a parte estudada (quando o processo como um todo não estiver sendo analisado).

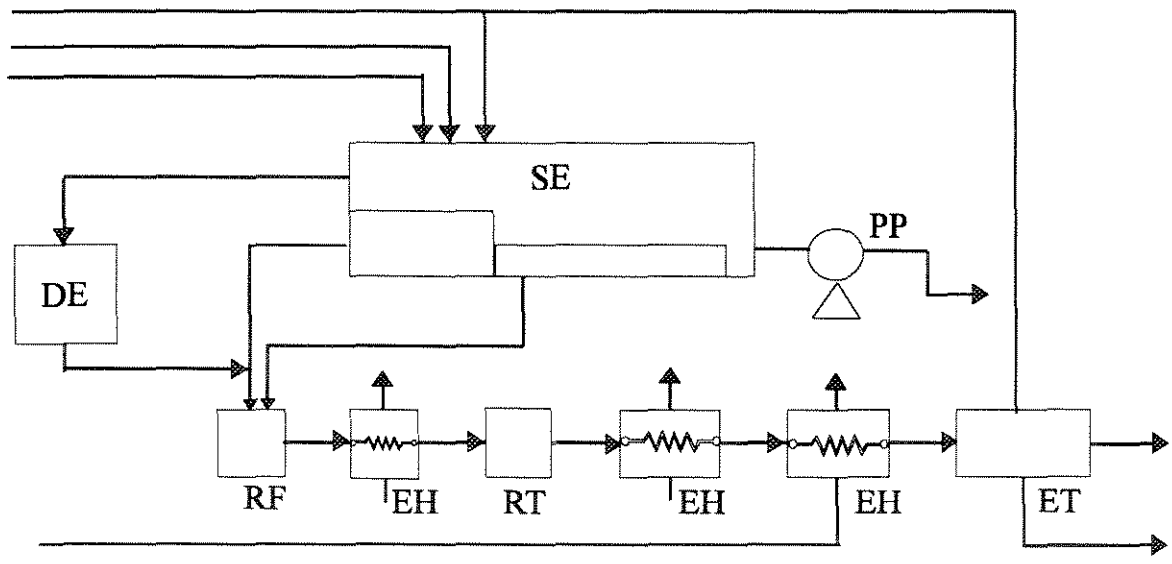


Figura 3.1: Fluxograma simplificado de uma planta de reforma gasosa com hidrogenação.

onde⇒DE: Dessulfurizador,
RF: Reformador,
EH: Trocador de Calor,
RT: Reator,
ET: Torre de Extração,
SE: Super Trocador de Calor,
PP: Bomba.

A partir do fluxograma, constrói-se o gráfico de processo o qual fornecerá as informações úteis para o equacionamento do problema.

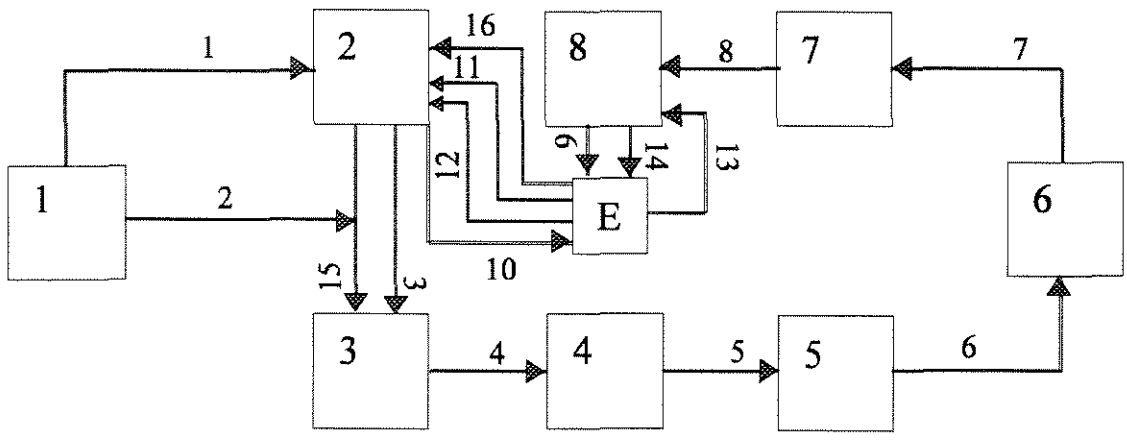


Figura 3.2: Gráfico de processo da planta de reforma gasosa.

O gráfico de processo permitirá estabelecer a interdependência entre as vazões e o dados do "histórico" do processo, expressa pelas equações de balanço de massa. O nó E representa o meio ambiente, onde entram e saem as correntes não interligadas a dois nós.

B- Simplificação do sistema de equações

No equacionamento do problema, faz-se uso de procedimento que relaciona as correntes aos nós(equipamentos) de maneira geral, ou seja, tudo se passa como se todas as correntes entrassem em todos os nós e ainda que todos os valores de fluxos e concentrações são medidos. Na prática, nem uma coisa nem outra se verificam. As correntes têm ligações específicas e nem todas as correntes são medidas. Problemas de ordem técnica ou econômica podem inviabilizar a obtenção de algumas medidas. Deste modo, é preciso retirar do conjunto de equações as variáveis não medidas, de modo a tornar o problema solúvel. Existem procedimentos específicos para se determinar quais as variáveis necessárias para se caracterizar um processo, mas isto está fora do âmbito deste trabalho. Aqui, se trabalha com o pressuposto que as variáveis medidas sejam no mínimo suficientes para se caracterizar o sistema.

C- Ajuste

Finalmente o ajuste de dados é conseguido por uma técnica apropriada. Neste caso conforme já citado, utiliza-se a projeção matricial para se "otimizar" o problema proposto.

3.2.2 EQUACIONAMENTO DO PROBLEMA

Todo problema para ser resolvido precisa estar identificado e devidamente equacionado. No caso da reconciliação de dados, o problema está identificado — corrigir os erros de flutuação aleatória nos valores das variáveis monitoradas. Como observação, vale incluir que os termos em negrito que aparecem nas equações deste

trabalho representam matrizes ou vetores, exceto o termo x e suas variantes que também é um vetor mas não aparece em negrito.

Para se equacionar o problema, busca-se uma relação matemática que possa relacionar os valores das variáveis medidas aos valores esperados. Aqui, será utilizado o método dos mínimos quadrados (MMQ) descrito no Apêndice B, definido para minimizar os erros de medida.

Dada a função objetivo, e tendo-se as restrições do processo, utiliza-se um método para se chegar à simplificação das variáveis. As restrições e o método de simplificação estão descritos na secção anterior.

Feitos o equacionamento e a simplificação, parte-se para a resolução do problema. A metodologia é baseada nos multiplicadores de Lagrange, Apêndice B.

O cálculo dos balanços de massa e/ou energia nas correntes de um processo químico é uma ferramenta básica para monitorar sua performance. Para se calcular estes balanços é preciso se determinar tantas medidas quanto possível de taxas de fluxo, de concentrações e de temperaturas nas correntes do processo. Estas medidas são sujeitas a erros randômicos e também a erros grosseiros, os quais podem ou não serem consistentes com as leis de conservação de massa e energia. Os dados devem, então, ser ajustados de tal modo que os valores obedeçam às leis de conservação e ainda os valores medidos devem ser de tal maneira ponderados de modo a se minimizar os ajustes necessários.

Conforme exposto anteriormente, os valores medidos não podem estar corrompidos por erros grosseiros. Instrumentos mal calibrados, mal ajustados, vazamentos, períodos fora do estado estacionário (partidas de unidades, por exemplo), tanques intermediários, devem ser eliminados do conjunto de valores medidos. A utilização destes valores corrompidos poderá distorcer seriamente os resultados trazendo riscos ao processo.

KUHEN e DAVIDSON(1961) foram os primeiros a considerarem o problema da reconciliação. Utilizaram os multiplicadores de Lagrange para conseguirem os ajustes ótimos, considerando o caso em que todas as correntes eram medidas. Subsequente a este trabalho, existem diversos outros, citados no capítulo 2.

Os problemas estudados até recentemente consideravam só casos restritos onde as concentrações medidas na corrente ou eram todas medidas ou nenhuma era medida. MAH e STANLEY(1981a,b) quebraram esta restrição. Reações químicas podem, usualmente, ser acrescentadas às restrições do problema, adicionando-se alimentações e produtos fictícios às correntes no nó onde a reação ocorre.

A proposta de CROWE(1983) é mais geral pois não depende de restrições consideradas por outros autores. Sua proposta está baseada na projeção matricial. A única restrição aqui é que o sistema seja linear (no capítulo 4 está a teoria sobre o caso não-linear). O caso linear é baseado na consideração de que para qualquer corrente na qual se conhece a concentração ou a temperatura, a taxa de fluxo total também deve ser medida. O número de concentrações dos componentes as quais são medidas em qualquer corrente é arbitrária. A taxa de fluxo da entalpia total ou também de um componente na corrente pode ser calculada, se os dados forem medidos, a partir do produto da taxa de fluxo total e da entalpia específica ou concentração usada como medida a ser ajustada. Pelo fato da taxa de fluxo total ser um fator comum a todas as taxas dos componentes ou da entalpia na corrente, haverá variância entre estas taxas. CROWE assume que as taxas de fluxo medidas são independentes, não impondo isto nenhuma restrição à generalidade do problema.

Considere uma planta química na qual existem K unidades de processo (ou nós), J correntes e C componentes em cada corrente. A estrutura da planta é expressa pela matriz incidência A com linhas correspondendo aos nós e as colunas correspondendo às correntes. Então

A_{kj}=1 se a corrente j entra no nó k
A_{kj}=-1 se a corrente j sai do nó k
A_{kj}=0 se a corrente não está em contato com aquele nó.

para k=1,2,3,....,K e j=1,2,3,....,J.

Do gráfico de processo, expresso na figura 3.2, tem-se a seguinte matriz incidência. (o nó meio ambiente não faz parte da matriz incidência).

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	-1	0	0	0	0	0	0	-1	1	1	1	0	-1	1
3	0	1	1	-1	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	-1	0	0	0	0	-1	0	0

Figura 3.3: Matriz incidência **A**, onde as linhas são correspondentes aos nós e as colunas às correntes.

Uma vez construída a matriz **A**, substitui-se cada elemento ± 1 (em **A**) pela matriz identidade (com os elementos unitários com o mesmo sinal dos elementos de **A**). Onde os elementos na matriz inicial forem nulos, substitui-se por matrizes nulas do mesmo tamanho das identidades. A matriz resultante é chamada de matriz dos balanços mássicos **B**, e é composta por blocos ou sub-matrizes de tamanho c x c, ou seja, c-linhas e c-colunas. Para todo nó k no qual haja reações químicas (R_k), constrói-se a matriz estequiometria $\mathbf{S_k}$.

Cada linha de $\mathbf{S_k}$ corresponde a uma reação química r(=1,2,....,R_k) e cada coluna corresponde a um componente c(=1,2,...,C).
O elemento $\mathbf{S_{k,rc}}$ é o coeficiente estequiométrico do componente c na reação r ocorrendo no nó k, seguindo-se a convenção na qual os coeficientes serão negativos, positivos ou nulos se se tratarem de reagentes, produtos ou inertes, respectivamente. Um vetor ε_k das extensões da reação é definido para cada nó reacional. Então, define-se uma matriz estequiométrica global com blocos $\mathbf{S_k}$ (R_k x C) diferentes de zero somente onde se tiver nó reacional. Cada uma das K bloco-colunas contêm C colunas e corresponde um nó.

Por exemplo se

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{s}_4 & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.1}$$

ocorrem reações químicas somente nos nós 1, 3 e 4. Nos nós 2, 5 e 6 os processos são apenas físicos.

Seja, agora, x_j o vetor dos componentes de taxas de fluxo na corrente j e $x^T=[(x_1)^T(x_2)^T\,(x_j)^T]$, então o balanço material para a planta é escrito como

$$\mathbf{B}\,x + \mathbf{S}^T\xi = \mathbf{0} \tag{3.2}$$

onde,

B representa a matriz dos componentes que entram e saem do processo,
$\mathbf{S}^T$ é a matriz das reações químicas,
ξ é o vetor das extensões das reações para todos os nós reacionais.

O vetor ξ não é medido ou conhecido a princípio.

Balanços de energia podem ser adicionados à definição do problema adicionando-se taxas de fluxos de entalpia para o (c+1)-ésimo componente, ou seja, adiciona-se a cada linha ao final dos c-elementos (correspondente a um bloco) o termo referente à entalpia. Então, **B** terá k blocos-linha cada um. A matriz S_k é modificada adicionando-se uma coluna contendo as mudanças de entalpia padrão para cada reação. O balanço de energia pode ter, além do mais, para cada nó, uma taxa de fluxo de valor $\mathbf{q_k}$ (normalmente não-medida).

Quando todos os componentes na corrente são balanceados deve-se garantir, se o fluxo total for fornecido, que o ajuste total das taxas de fluxo é igual à soma das taxas ajustadas para os componentes. Assim, se x_{tj} é a taxa de fluxo total na corrente j,

$$\mathbf{1}^T x_j = x_{tj} \tag{3.3}$$

onde $\mathbf{1}^T$ é um vetor linha de 1's. A matriz é, deste modo, aumentada adicionando uma linha e uma coluna para cada corrente, e o vetor x_t das taxas de fluxo totais é adicionado ao vetor x, tal que, a equação (3.2) se torna

$$\left[\begin{array}{cccc} \mathbf{B} & \cdots & \cdots & \mathbf{0} \\ \hline \mathbf{1}^T & \mathbf{0} & \cdots & \mathbf{-I} \\ \mathbf{0} & -\mathbf{1}^T & \mathbf{0} & \mathbf{-I} \\ \cdots & \cdots & \cdots & \cdots \end{array}\right] * \left[\begin{array}{c} x \\ \hline x_t \end{array}\right] + \left[\begin{array}{c} \mathbf{S}^T \\ \hline \mathbf{0} \end{array}\right] * \varepsilon = \mathbf{0} \tag{3.4}$$

Na prática, a equação (3.3) é uma restrição a mais do problema, a qual deve ser obedecida na resolução.

É importante não alterar os dados originalmente medidos de forma que a equação (3.3) seja forçada a ser válida para eles. A inclusão do vetor de vazões totais no conjunto

de variáveis medidas destruirá informações estatísticas, introduzindo dependência entre os dados, BOX et al.(1973).

Apesar de a equação (3.4) ser mais completa em termos de informações, aqui, será usada a equação (3.2) (mais simplificada) para se mostrar o desenvolvimento das equações básicas necessárias à reconciliação de dados.

É bom notar que uma restrição comum como a igualdade de concentrações e temperatura, entre as correntes que deixam um divisor de fluxo, é desconhecida e para satisfazê-la seria necessário igualar as razões entre vazão do componente e a total em duas correntes, resultando em uma restrição não-linear. Tais restrições deveriam ser reescritas para conterem frações de divisões desconhecidas as quais devem ser encontradas por iteração. ALMASY et al (1969), propôs um método para iterá-los de modo que todas as frações desconhecidas estão juntas. Abaixo é proposto um método que itera somente frações de divisão e outros parâmetros similares.

O vetor $\tilde{x}$ das taxas de fluxo dos componentes medidas é novamente escrito para todos os vetores $\tilde{x}_j$ de cada corrente mas contém tantos elementos quantos forem os valores medidos. A taxa de fluxo de qualquer valor não medido é incluído em **u**, o vetor das taxas de fluxo desconhecidas.

As colunas de **B** podem, então, ser particionadas de forma que

$$\mathbf{B} = [\, \mathbf{B}_0 \mid \mathbf{B}_1 \mid \mathbf{B}_3 \,]$$

Foi utilizado o $\mathbf{B}_3$ e não $\mathbf{B}_2$ por causa da parte não-linear, onde será empregado este termo.

$\mathbf{B}_0$, $\mathbf{B}_1$, e $\mathbf{B}_3$ contêm, respectivamente, as colunas das c taxas de fluxo exatamente conhecidas, as taxa de fluxo $\tilde{x}$ medidas e as taxa de fluxo **u** não medidas.

As quantidades $\tilde{x}$ são assim ajustadas por quantidades **a** tal que os balanços materiais como dados na equação (3.2) sejam válidos para as vazões ajustadas, desta maneira

$$\mathbf{B}_0\mathbf{c} + \mathbf{B}_1.(\tilde{x} + \mathbf{a}) + \mathbf{B}_3\mathbf{u} + \mathbf{S}^T \xi = \mathbf{0} \qquad (3.5)$$

conforme definições de $\tilde{x}$ e **a** feitas na secção 3.1.

Apesar de já definido anteriormente, não é demais lembrar que aqui estão sendo contempladas apenas variáveis corrompidas por erros randômicos.

Assume-se que os balanços são linearmente independentes.

As taxas de fluxos, $\hat{x}$ (reconciliadas) são dadas por

$$\hat{x} = \tilde{x} + \mathbf{a} \qquad (3.6)$$

A fim de se simplificar a notação matricial, define-se as matrizes **P** e o vetor **v** tais que:

$$\mathbf{P} = [\, \mathbf{B}_3 \vdots \mathbf{S}^T \,] \qquad (3.7)$$

e

$$\mathbf{v}^T = [\, \mathbf{u}^T \vdots \varepsilon^T \,] \qquad (3.8)$$

A notação apresentada indica que a matriz **P** é definida pela simples junção das colunas de $\mathbf{B}_3$ às de $\mathbf{S}^T$, da mesma forma ocorre com o vetor **v**, em relação a **u** e ε.

Com base nestas definições, pode-se reescrever a equação (3.5) de modo que

$$\mathbf{B}_0\mathbf{c} + \mathbf{B}_1.(\tilde{x} + \mathbf{a}) + \mathbf{P}\mathbf{v} = \mathbf{0} \qquad (3.9)$$

Tem-se agora uma equação com três termos, onde

$\mathbf{B}_0 \rightarrow$ indica os coeficientes das taxas de fluxo exatamente conhecidas

$\mathbf{B}_1 \rightarrow$ indica os coeficientes das taxas de fluxo reconciliadas

$\mathbf{P} \rightarrow$ indica os coeficientes das taxas de fluxo não-medidas (inclui os demais termos não-medidos como por exemplo as extensões das reações químicas).

3.3 DECOMPOSIÇÃO DO PROBLEMA

A equação (3.9) é, portanto, a equação (3.5) reescrita em notação mais útil para a resolução do problema. Tem-se desta equação todos os balanços materiais do sistema a ser estudado.

Para se determinar os valores reconciliados, representados pelo vetor multiplicado por $\mathbf{B}_1$, é preciso que se conheça os ajustes. Uma equação que contenha estes ajustes deve ser proposta.

Se a taxa de fluxo total em uma corrente é medida, quando qualquer concentração naquela corrente é medida, então, o problema de reconciliação de dados pode ser escrito como

PROBLEMA P.1

$$\mathbf{F(a)} \equiv \mathbf{a}^T \Sigma_1 \mathbf{a} \qquad (3.10)$$

O problema, então, é o de minimizar os ajustes, deste modo, tem-se

$$\underset{\mathbf{a},\mathbf{v}}{\mathbf{Min}}\ \mathbf{F(a)} \equiv \mathbf{a}^T \Sigma_1 \mathbf{a}$$

sujeita às restrições dadas pela equação (3.9).

A matriz Σ é, geralmente, uma matriz ponderante inversa, definida positiva. Contudo, aqui, é a matriz variância-covariância das medidas $\tilde{x}$. É, ususalmente, admitida ser uma matriz bloco-diagonal com cada bloco diagonal correspondendo a uma corrente, entretanto, esta não é uma consideração obrigatória. Quaisquer taxas de fluxo consideradas exatamente conhecidas deverão ser separadas em um vetor de valores constantes para evitar fazer Σ singular.

O problema **P.1** pode ser resolvido prontamente utilizando-se os multiplicadores de Lagrange, entretanto será definido um segundo problema cuja solução também é solução do primeiro. O que se quer neste ponto é desaparecer com variáveis não-medidas, o que diminui o esforço de cálculo. Aqui, entra a projeção matricial, Apêndice B, para solucionar o problema de simplificação de equações matriciais.

A projeção matricial é usada para obter um conjunto de equações de balanço a partir do balanço original dado pela equação (3.9). Esta proposta é equivalente ao algoritmo de VACLAVEK(1976) para o caso onde a corrente é completamente medida ou não medida.

Define-se uma matriz **Y** cujas colunas completam ("span") o espaço nulo de de $\mathbf{P}^T$. Assim,

$$\mathbf{P}^T\mathbf{Y} = \mathbf{0} \tag{3.11}$$

A princípio, qualquer matriz **Y**, que satisfaça a equação acima, poderia ser utilizada. Qual vetor **w** ($\neq\mathbf{0}$) tal que $\mathbf{P}^T\mathbf{w} = \mathbf{0}$, é uma combinação linear das colunas de **Y**. Contudo, a matriz precisa ser calculada de uma forma sistemática a fim de que se tenha uma forma única e garantida de se determiná-la.

Da álgebra linear tem-se que a matriz **Y** pode ser construída como se segue

* reduz-se as colunas de **P** transformando-a em $\mathbf{P}_r$, com as colunas linearmente independentes,

$$[\,\mathbf{P}_r \vdots \mathbf{0}\,] = \mathbf{PF} \tag{3.12}$$

onde **F** representa a matriz não singular a qual transforma as colunas **P** de maneira adequada.

* a partir de $\mathbf{P}_r$, separa-se a mesma de modo que,

$$\mathbf{P}_r = \left[\frac{\mathbf{P}_1}{\mathbf{P}_2}\right]$$

com $\mathbf{P}_1$ a matriz máxima quadrada não-singular. a partir daí define-se **Y**

$$\mathbf{Y}^T = [\,-\mathbf{P}_2\mathbf{P}_1^{-1} \vdots \mathbf{I}\,] \tag{3.14}$$

Para casos em que não hajam reações químicas, o uso de **Y** equivale ao esquema de balanço de VACLAVEK(1969b) aplicado a cada componente separadamente.

Retornando à equação (3.9), tem-se

$$\mathbf{Y}^T[\mathbf{B}_0\mathbf{c} + \mathbf{B}_1\mathbf{c}(\tilde{x} + \mathbf{a}) + \mathbf{P}\mathbf{v}] = \mathbf{0} \tag{3.15}$$

aplicando-se a propriedade distributiva válida para multiplicação de matrizes, tem-se

$$\mathbf{Y}^T [\mathbf{B}_0\mathbf{c} + \mathbf{B}_1(\tilde{x} + \mathbf{a})] + \mathbf{Y}^T\mathbf{P}\mathbf{v} = \mathbf{0} \tag{3.16}$$

mas

$$\mathbf{Y}^T\mathbf{P} = \mathbf{P}^T\mathbf{Y} = \mathbf{0} \tag{3.17}$$

assim

$$\mathbf{Y}^T[\mathbf{B}_0\mathbf{c} + \mathbf{B}_1(\tilde{x} + \mathbf{a})] = \mathbf{0} \tag{3.18}$$

Isto leva à definição do

PROBLEMA P2

(a) minimizar $\mathbf{F}(\mathbf{a})$ sujeito à equação (3.18)

(b) se $\mathbf{a}_*$ é a solução mínima de **P2**(a), pode-se resolver

$$\mathbf{P}\mathbf{v}_* = -\mathbf{B}_1(\tilde{x} + \mathbf{a}_*) - \mathbf{B}_0\mathbf{c} \tag{3.19}$$

para $\mathbf{v}_*$, encontrando-se assim os valores não-medidos das variáveis observáveis.

É claro que qualquer solução ($\mathbf{a}_*$, $\mathbf{v}_*$) que satisfaça à equação (3.9) também satisfará à equação (3.19). Então, o mínimo ajuste ($\mathbf{a}_*$) deve ser o mesmo para ambos os problemas. Note-se que o problema **P2** é uma extensão do esquema de balanço reduzido (RBS) de VACLAVEK(1969b) na qual as equações de balanço (3.18) têm somente taxas de fluxo medidas mas as combinações remanescentes dos nós são em geral diferentes para diferentes componentes. O RBS é também chamado esquema de reconciliação gráfica, MAH et al.(1976). Aqui, a matriz **Y** funciona como uma extensão do trabalho de MAH no particionamento da matriz incidência para o problema com taxa de fluxo para um componente único. A escolha da matriz **Y** não afeta os valores do ajuste $\mathbf{a}_*$ mas uma possível escolha de **Y** para o caso de não haver reações químicas é equivalente a deletar ou juntar nós conforme sugerido por VACLAVEK(1969a), para cada componente separadamente.

A solução mínima para **P2**(a) existe por causa matriz variância Σ ser definida positiva e consequentemente o valor o qual minimiza $\mathbf{F}(\mathbf{a})$ é único.

O vetor $\mathbf{v}_*$ o qual satisfaz a equação (3.19) existe e é único se e somente se as colunas da matriz **P** são linearmente independentes. Isto significa que

$$\mathbf{P}\mathbf{v} = \mathbf{0} \Rightarrow \mathbf{v} = \mathbf{0} \tag{3.20}$$

o que leva a um teste computacional direto da unicidade de $\mathbf{v}_*$.

A partir das definições de **P** e de **v** , a equação (3.20) torna-se

$$\mathbf{B}_2\mathbf{u} + \mathbf{S}^T\xi = \mathbf{0} \tag{3.21}$$

Assim, a não unicidade de $\mathbf{v}_*$ é equivalente a se poder satisfazer o balanço material envolvendo somente parâmetros não-medidos. Notar que os valores dos elementos de $\mathbf{u}$ e ξ devem ser de igual sinal, tal que os balanços possam ser feitos sem consideração das reais direções de fluxos ou das reações.

As variáveis não-medidas são classificadas, VACLAVEK(1969b), como determináveis ou indetermináveis (observáveis ou não-observáveis) e as variáveis medidas como superdeterminadas ou criticamente determinadas (redundantes ou não redundantes). Quando a matriz Σ é diagonal, uma medida não-redundante não é ajustada, mas no caso presente, a medida é em geral ajustada se tem covariância não-nula com a medida redundante.

De acordo com as classificações generalizadas de STANLEY e MAH(1981a), as aqui presentes, variáveis não-observáveis, correspondem ao grupo de colunas linearmente dependentes da matriz $\mathbf{P}$ e podem ser identificadas pela redução de $\mathbf{P}$. As variáveis restantes (não-medidas) são observáveis. Uma variável $\mathbf{v}_i$ é observável somente se há uma medida $\tilde{x}_j$, não-redundante, a qual se fosse removida tornaria $\mathbf{v}_i$ não-observável. Esta variável corresponderia à coluna de $\mathbf{P}$ que é linearmente independente de outras colunas da mesma matriz, mas que é dependente das colunas em $\mathbf{B}_1$ correspondentes a $\tilde{x}_j$.

As medidas não-redundantes correspondem às colunas zero de $(\mathbf{Y}^T\mathbf{B}_1)$ e não contribuem diretamente para os cálculos dos ajustes.

3.4 DETERMINAÇÃO DOS AJUSTES

Utilizando os multiplicadores de Lagrange, pode-se resolver o problema **P2** encontrando o ponto estacionário $(\mathbf{a}_*, \mathbf{v}_*)$ do Lagrangiano.
Assim

$$L(\mathbf{a},\lambda) \equiv 0.5\, \mathbf{F}(\mathbf{a}) - \lambda^T\mathbf{Y}^T[\mathbf{B}_0\mathbf{c} + \mathbf{B}_1(\tilde{x} + \mathbf{a})] \tag{3.22}$$

O desenvolvimento completo desta equação está detalhado no Apêndice B.

Resumidamente, como resultados, tem-se

$$\mathbf{a}_* = \Sigma\mathbf{B}_1\mathbf{Y}\lambda_* \tag{3.23}$$

e

$$\mathbf{H}\lambda_* = -\mathbf{Y}^T[\mathbf{B}_0\mathbf{c} + \mathbf{B}_1\tilde{x}] \tag{3.24}$$

onde

$$\mathbf{H} \equiv \mathbf{Y}^T\mathbf{B}_1\Sigma\mathbf{B}_1{}^T\mathbf{Y} \tag{3.25}$$

Desta maneira, a equação (3.24) pode ser resolvida para λ_* e então $\mathbf{a}_*$ pode ser calculado. Como $\mathbf{H}$ é uma matriz quadrada, pode ser facilmente invertida. Resolvida a equação (3.20), pode-se também determinar os valores de $\mathbf{v}_*$, quando isto for possível.

A determinação dos valores de $\mathbf{v}_*$ é chamada coaptação de dados, MAH et al.(1976).

Duas secções mais ainda dizem respeito a este capítulo, segundo o modelo utilizando restrições lineares. São a estrutura do programa e os procedimentos de utilização do mesmo.

O fato de serem os procedimentos praticamente idênticos aos dois tipos de restrições (lineares e não-lineares) mais o fato de o tema central deste trabalho ser a reconciliação com restrições não-lineares, torna-se pertinente que estas duas secções sejam descritas apenas no próximo capítulo que é justamente onde se trata do desenvolvimento de sistemas não-lineares. Particularidades, quando houver, estarão destacadas.

4 RECONCILIAÇÃO DE DADOS DE PROCESSO- CASO NÃO-LINEAR

4.1 INTRODUÇÃO

Assim como o apresentado no capítulo 3, a reconciliação de dados com restrições não-lineares seguirá o procedimento de utilização da projeção matricial, para fins de minimização do erro entre os valores medidos e os reconciliados. As mesmas convenções descritas para o capítulo anterior valerão também neste; pontos específicos, pertinentes apenas à reconciliação com restrições não-lineares, serão destacados dentro do texto para melhor entendimento do mesmo.

4.2 A RECONCILIAÇÃO

A pedra fundamental para o monitoramento da performance de uma planta é um conjunto de balanços em estado-estacionário para as taxas de fluxo totais e de componentes. Estas taxas de fluxos são normalmente obtidas a partir de medidas de taxas de fluxo totais e de concentrações, as quais estão sujeitas a erros randômicos (ou estatísticos) e algumas vezes a erros grosseiros, os quais de maneira geral violam as leis de conservação para o sistema. As medidas, assim obtidas, precisam ser reconciliadas, da melhor maneira possível, de modo a conciliar as restrições e as leis de conservação.

No caso linear, foi assumido que em qualquer corrente onde houvesse uma taxa de fluxo medida (ou a concentração), a taxa de fluxo total daquela corrente seria medida. Agora, esta consideração não se faz presente tal que as equações de balanço que contenham vazões desconhecidas faz o problema se tornar não-linear (ou bilinear no caso atual).

Conflitos com as leis de conservação podem surgir em secções do processo onde as medidas tenham sido feitas em todas as correntes nas equações de balanço. Trabalhos anteriores como os de VACLAVEK(1969), MAH et al.(1976) e os de ROMAGNOLI e STEPHANOPOULOS(1980) foram dirigidos de tal modo a encontrar de forma eficiente um conjunto mínimo de secções do processo com medidas redundantes, o chamado RBS ou balanço de esquema reduzido.

Infelizmente, esses procedimentos devem ser aplicados separadamente para cada componente ou para cada elemento em caso de reações, CROWE et al.(1983) propuseram um método para encontrar o RBS através da projeção matricial o qual elimina todos os fluxos não-medidos.

O modelo descrito para o caso linear é aqui estendido construindo-se duas matrizes projeções matriciais sucessivas. A primeira elimina todas as taxas de fluxo dos componentes desconhecidas e suas concentrações. A segunda elimina as taxas de fluxo totais não-medidas das equações de balanço.

O problema original é assim subdividido em três sub-problemas para serem seqüencialmente resolvidos. Aproximações iniciais das taxas de fluxo totais não-medidas são utilizadas para resolver os ajustes para as taxas de fluxo dos componentes. Estas aproximações são atualizadas iterativamente no segundo sub-problema até que a

convergência seja atingida. Quando estas taxas de fluxo dos componentes não-medidas são determinadas, é possível resolver o terceiro sub-problema.

Erros grosseiros nas medidas podem, aqui também, ocorrer devido ao mal funcionamento ou descalibração dos instrumentos, erros de amostragem, insuspeitos vazamentos ou afastamento do estado-estacionário. As medidas nas quais estão presentes erros grosseiros devem ser detectadas antes de se realizar o procedimento de reconciliação. Se as mesmas estiverem presentes, deverão ser corrigidas ou eliminadas, quando possível.

VACLAVEK et al.(1976a,b) desenvolveram a base teórica do problema bilinear e discutiram as condições sob as quais as equações de balanço original poderiam ser reduzidas em número. Eles assumiram que ou uma ou nenhuma das concentrações na corrente eram medidas e assim poderiam dividir as correntes em quatro categorias, a saber:

Categoria(tipo)	Taxa de Fluxo Total	Concentração
1	M	M
2	NM	M
3	M	NM
4	NM	NM

onde

M ∈ medida

NM ∈ não-medida

Entretanto, com uma distribuição arbitrária das medidas, a classificação deve se referir aos componentes nas correntes, onde as variáveis nas categorias 3 e 4 possam ser combinadas desde que as taxas de fluxos dos componentes são não-medidas em ambos os casos. Assim, haverá 3 categorias de variáveis (tipos):

1. Taxa de fluxo total e concentração medida e ajustável.
2.Concentração medida e ajustável.
3.Taxa de fluxo total não-medida ou desconhecida, taxa de fluxo dos componentes não-medida.

A classificação feita por VACLAVEK(1976a) de variáveis não conhecidas em determináveis e indetermináveis pode ser estendida aqui para as variáveis nas categorias 2 e 3 separadamente, de modo que também possam ser classificadas. As variáveis indeterminadas em cada caso corresponderão às colunas dependentes nas respectivas matrizes nas equações de balanço. VACLAVEK também dividiu as medidas de acordo com a possiblidade ou não de elas serem corrigidas. Esta divisão dependerá não somente dos fluxos mas também das variâncias-covariâncias (dispersão) das medidas.

A fim de reduzir o número de equações de balanço, sem sacrificar informações, VACLAVEK propôs a redução em dois passos. A formulação apresentada, aqui, levará a um diferente esquema de redução em dois passos, o qual primeiro elimina as variáveis da categoria ou tipo 3 e outras variáveis não-medidas, como por exemplo as extensões das reações, e então elimina as taxas de fluxo correspondentes às variáveis da categoria ou tipo 2.

4.3 EQUACIONAMENTO DO PROBLEMA

A base para o desenvolvimento das equações de balanço para o caso não-linear é o mesmo apresentado no capítulo 3 e o ponto de partida é o artigo de CROWE(1986). Alguns conceitos já foram apresentados no caso linear (capítulo 3) e não serão aqui, repetidos. A fim de facilitar a associação entre os dois procedimentos será usado o mesmo conjunto de símbolos. Aqueles que porventura forem diferentes poderão ser consultados na lista de símbolos.

Se o vetor x representa as n taxas de fluxos dos componentes e das taxas de fluxo totais em todas as correntes de um processo, os m balanços materiais e as demais restrições entre eles podem ser escritas como

$$\mathbf{B}\,\tilde{x} + \mathbf{S}^{T}\xi = \mathbf{0} \tag{4.1}$$

onde a matriz **S** representa a (p x m) matriz estequiométrica e ξ é o vetor composto de p extensões de reações para todas as reações no processo, conforme descrito no capítulo 3.

As colunas da (m x n) matriz **B** são então permutadas e particionadas de tal maneira que

$$\mathbf{B} \rightarrow [\ \mathbf{B}_0 \mid \mathbf{B}_1 \mid \mathbf{B}_2 \mid \mathbf{B}_3\ ]$$

onde as colunas de $\mathbf{B}_0$ correspondem às taxas de fluxo exatamente conhecidas e as de $\mathbf{B}_i$ (i=1,2,3) aos n_i componentes nas categorias 1, 2 e 3 respectivamente.

A taxa de fluxo do componente c na corrente j, na categoria 1, é definida por

$$x_{cj} = \mathbf{M}_j \mathbf{c}_{cj} \tag{4.2}$$

onde $\mathbf{M}_j$ é a taxa de fluxo total e $\mathbf{c}_{cj}$ é a concentração.

Os valores medidos serão representados por ~ acima do símbolo, como no capítulo 3, e os valores estimados por ^.

Assim,

$$\hat{x}_{cj} = \tilde{x}_{cj} + \mathbf{a}_{cj} \tag{4.3}$$

onde $\mathbf{a}_{cj}$ é o ajuste na corrente j para a taxa de fluxo medida do componente c.

Para uma corrente com taxa de fluxo total não-medida, usa-se **d** e δ para a sua concentração e ajuste respectivamente. Então,

$$\hat{\mathbf{d}}_{cl} = \tilde{\mathbf{d}}_{cl} + \delta_{cl} \tag{4.4}$$

para a concentração estimada do componente c na corrente l (ele).

A taxa de fluxo total desconhecida será N_l.

como no capítulo 3, define-se

$$\mathbf{P} = [\ \mathbf{B}_3 \mid \mathbf{S}^T\ ] \tag{4.5}$$

e representa todas as taxas de fluxos e extensões de reações pelo vetor **v**. Então, as restrições devem ser obedecidas pelos valores estimados, tal que

$$\mathbf{B}_0\mathbf{c} + \mathbf{B}_1(\tilde{x} + \mathbf{a}) + \mathbf{B}_2\mathbf{N}(\tilde{\mathbf{d}} + \delta) + \mathbf{P}\mathbf{v} = \mathbf{0} \tag{4.6}$$

onde **N** é a matriz diagonal das taxas de fluxo desconhecidas, com tantas entradas para cada corrente quantas forem a concentrações naquela corrente. O problema da reconciliação pode, a partir das relações básicas, ser definido.

4.4 DEFINIÇÃO DOS PROBLEMAS

4.4.1 PROBLEMA P1

Minimização dos ajustes

Seja minimizar a função $\mathbf{F}(\mathbf{a},\delta)$ sujeitas às equações (4.6)

$$\underset{\mathbf{a},\delta,\mathbf{v},\mathbf{n}}{\mathbf{Min}}\ \mathbf{F}(\mathbf{a},\delta) \equiv \mathbf{a}^T(\Sigma_1)^{-1}\mathbf{a} + \delta^{\ T}(\Sigma_d)^{-1}\delta \tag{4.7}$$

Σ_1 e Σ_d são estimadas a partir das matrizes variância-covariância.

A partir dessa função objetivo (4.7), e suas restrições (4.6), existem diversos passos que podem ser tomados para simplificar este problema. O primeiro passo é definir, como no capítulo 3, a matriz **Y** tal que

$$\mathbf{P}^T\mathbf{Y} = \mathbf{0} \tag{4.8}$$

onde as colunas de **Y** formam a base para o espaço nulo de $\mathbf{P}^T$.

Deste modo, tem-se que a equação (4.6), pode ser reduzida a

$$\mathbf{Y}^T\left[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1(\tilde{x} + \mathbf{a}) + \mathbf{B}_2\mathbf{N}(\tilde{\mathbf{d}} + \delta)\right] = \mathbf{0} \tag{4.9}$$

Este é o primeiro estágio na redução do número de equações eliminando todas as variáveis na categoria 3, ou do tipo 3 representadas pelo vetor **v**. A construção de **Y** pode ser feita como na parte linear, particionando **P**

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{P}_1 & \mathbf{P}_2 \\ \hline \mathbf{P}_3 & \mathbf{P}_4 \end{array}\right] \tag{4.10}$$

onde $\mathbf{P}_1$ representa a máxima sub-matriz quadrada não-singular contida em **P**. Pode-se desta maneira escolher **Y** de forma que

$$\mathbf{Y}^T = \left[-\mathbf{P}_2\mathbf{P}_1^{-1} \vdots \mathbf{I}\right] \tag{4.11}$$

onde I representa a matriz identidade que completa as linhas de $\mathbf{Y}^T$.

Por definição, as colunas do lado direito de **P** dependem das colunas do lado esquerdo (são linearmente dependentes) tal que a equação (4.8) possa ser satisfeita. As colunas dependentes de **P** correspondem às variáveis indeterminadas na categoria (ou do tipo) 3.

Deve-se notar que quando $\mathbf{P}_1$ não estiver no lado esquerdo superior, as colunas de cada partição de $\mathbf{Y}^T$ na equação (4.11) deverão ser permutadas para corresponder respectivamente às linhas de $\mathbf{P}_1$ e $\mathbf{P}_2$.

O segundo passo da simplificação envolve a definição da matriz **Z** tal que como no caso de **P**, seja a base do espaço nulo de D.

Assim, tem-se

$$\mathbf{D}^T \equiv \left[\mathbf{B}_{21}\tilde{\mathbf{d}}_1\mathbf{B}_{22}\tilde{\mathbf{d}}_2\cdots\mathbf{B}_{2l}\tilde{\mathbf{d}}_l\cdots\right]^T \mathbf{Y} \tag{4.12}$$

onde $\mathbf{B}_{2l}$ é o conjunto das colunas de $\mathbf{B}_2$ correspondentes à corrente l, e $\mathbf{d}_l$ é o vetor das concentrações medidas naquela corrente.

Então,

$$\mathbf{D}^T\mathbf{Z} \equiv \mathbf{0} \tag{4.13}$$

e

$$\mathbf{Dn} = \mathbf{Y}^T\mathbf{B}_2\mathbf{N}\tilde{\mathbf{d}} \tag{4.14}$$

onde **n** é o vetor das taxas de fluxo totais distintas e desconhecidas na categoria 2, tal que a equação (4.9) possa ser simplificada a

$$\mathbf{Z}^T\mathbf{Y}^T\left[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1(\tilde{x} + \mathbf{a}) + \mathbf{B}_2\mathbf{N}\delta\right] = \mathbf{0} \tag{4.15}$$

consequentemente, tem-se mais uma vez, uma simplificação do problema eliminando-se do mesmo as variáveis na categoria 2. Pode-se representar **Z**, analogamente a **Y**, por

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{D}_2 \\ \mathbf{D}_3 & \mathbf{D}_4 \end{bmatrix} \tag{4.16}$$

com $\mathbf{D}_1$ sendo a máxima sub-matriz quadrada não-singular em **D**.

$$\mathbf{Z}^T = \left[-\mathbf{D}_2\mathbf{D}_1^{-1} \vdots \mathbf{I}\right] \tag{4.17}$$

As taxas de fluxo total que correspondem às colunas dependentes de **D** são indeterminadas.

Há diversas vantagens em se definir **Y** e **Z** separadamente. Primeiro, as inversas de $\mathbf{P}_1$ e $\mathbf{D}_1$ são mais eficientes se computadas separadamente que se forem combinadas numa única e grande matriz. Em segundo lugar, condições separadas são obtidas para indeterminação das variáveis na categoria 3 e da taxa de fluxo na categoria 2. (A condição é um fator de extrema importância na determinação de variáveis das categorias 2 e 3. Sobre este problema, ver capítulo 6 exemplo 2 e também Apêndice B). Em terceiro lugar, **Y** é uma matriz constante ao passo que **Z** contém todas as variabilidades devido a $\tilde{\mathbf{d}}$, o que influi também na condição da matriz.

Retornando ao terceiro passo de simplificação, a equação (4.6), tem-se que rescrevendo o segundo termo na função objetivo, equação (4.7), vem

$$\delta^T(\Sigma_d)^{-1}\delta = (\mathbf{N}\delta)^T(\mathbf{N}\Sigma_d\mathbf{N})^{-1}(\mathbf{N}\delta) \tag{4.18}$$

Se forem feitas estimativas de **N**, chamadas $\mathbf{N}_0$, então define-se

$$\Sigma_2 \equiv \mathbf{N}_0\Sigma_d\mathbf{N}_0 \tag{4.19}$$

com base nestas suposições, pode-se reescrever o problema **P1** renominando-o como problema **P2**.

4.4.2 PROBLEMA P2

Equacionamento do problema simplificado

A partir das equações (4.7) e (4.9), pode-se redefinir uma função objetivo, qual seja

(a) Resolver

$$\mathbf{Min}\ G(\mathbf{a},\ \mathbf{N}\delta) \equiv \mathbf{a}^T\Sigma_1^{-1}\mathbf{a} + (\mathbf{N}\delta)^T\Sigma_2^{-1}(\mathbf{N}\delta) \tag{4.20}$$

sujeita agora à equação dos balanços simplificada,

(b) a partir da resolução das equações da parte (a), resolve-se a equação (4.9) a qual é reescrita com a ajuda da equação (4.14) de tal modo que

$$\mathbf{Dn} = -\mathbf{Y}^T\left[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1(\bar{x}+\mathbf{a}) + \mathbf{B}_2(\mathbf{N}\delta)\right] \tag{4.21}$$

para **n**.

(c) a partir da solução das partes (a) e (b), então pode-se resolver

$$\mathbf{Pv} = -\left[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1(\bar{x}+\mathbf{a}) + \mathbf{B}_2(\mathbf{N}\delta)\right] \tag{4.22}$$

para **v**.

A condição necessária para existir a solução para cada umas das equações (4.21) e (4.22) é satisfeita pelas definições de **Y** e **Z**. Se as variáveis indeterminadas são assumidas como zero, as demais variáveis desconhecidas podem se resolvidas por

$$\mathbf{n}_d = \mathbf{D}_1^{-1} [\, \mathbf{I} \mid \mathbf{0}] \, \mathbf{R}(21) \tag{4.23}$$

e

$$\mathbf{v}_d = \mathbf{P}_1^{-1} [\, \mathbf{I} \mid \mathbf{0}] \, \mathbf{R}(22) \tag{4.24}$$

onde, **R**(21) e **R**(22) são os lados direitos das equações (4.21) e (4.22) das quais retêm-se somente as linhas correspondentes a D_1 e P_1, respectivamente.

Uma vez que neste ponto do procedimento já existem as matrizes $\mathbf{D}_1$ e $\mathbf{P}_1$ invertidas, as equações são prontamente resolvidas. O espaço de todas as soluções para a equação (4.21) e (4.22) podem ser encontradas pelo fato de que

$$\mathbf{D}\left[\frac{-\mathbf{D}_1^{-1}\mathbf{D}_3}{\mathbf{I}}\right] = \mathbf{0} \tag{4.25}$$

assim,

$$\mathbf{n} = \left[\frac{\mathbf{n}_d}{\mathbf{0}}\right] + \left[\frac{-\mathbf{D}_1^{-1}\mathbf{D}_3}{\mathbf{I}}\right]\mathbf{z} \tag{4.26}$$

para um **z** arbitrário. Similarmente, uma solução do problema pode ser escrita para **v**. Não há, entretanto, a garantia de que as soluções para $\mathbf{n}_d$ e $\mathbf{v}_d$ sejam não-negativas tal que as soluções gerais possam ser usadas para encontrar os vetores **n** e **v**.

4.5 SOLUÇÕES ANALÍTICAS DOS PROBLEMAS

Resta, agora, resolver a equação (4.20) representada no ítem a da secção 4.4.2. Utilizando-se o Lagrangiano L_2, pode-se definir

$$\mathbf{L}_2 \equiv \mathbf{0{,}5}.\left[\mathbf{a}^T \Sigma_1^{-1} \mathbf{a} + (\mathbf{N}\delta)^T \Sigma_2^{-1} (\mathbf{N}\delta)\right] - \lambda^T \mathbf{Z}^T \mathbf{Y}^T \left[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1(\tilde{x} + \mathbf{a}) + \mathbf{B}_2(\mathbf{N}\delta)\right] \tag{4.27}$$

com λ, sendo o vetor multiplicador lagrangiano. O desenvolvimento desta equação está descrito no Apêndice B.

Tomando-se as derivadas com respeito a **a** e a ($\mathbf{N}\delta$) e igualando-se a zero, tem-se respetivamente,

$$\mathbf{a} = \Sigma_1 \mathbf{B}_1^T \mathbf{Y}\mathbf{Z}\lambda \tag{4.28}$$

e

$$(N\delta) = \Sigma_2 B_2^T Y Z \lambda \tag{4.29}$$

Se **D** tiver posto igual ao número de linhas, **Z** = **0**, **a** = **0** e **Nδ**= **0**.

A partir da equação (4.15),

$$\mathbf{Z}^T\mathbf{Y}^T[\mathbf{B}_1\mathbf{a} + \mathbf{B}_2\mathbf{N}\delta] = -\mathbf{Z}^T\mathbf{Y}^T[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1 x] \tag{4.30}$$

das equações 28 e 29, com

$$\mathbf{H} \equiv -\mathbf{Y}^T\left(\mathbf{B}_1 \Sigma_1 \mathbf{B}_1^T + \mathbf{B}_2 \Sigma_2 \mathbf{B}_2^T\right)\mathbf{Y} \tag{4.31}$$

obtém-se,

$$\lambda = -\left(\mathbf{Z}^T\mathbf{H}\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{Y}^T[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1\tilde{x}] \tag{4.32}$$

Isto permite a computação de λ, **a** e **Nδ** a partir das equações (4.32), (4.28) e (4.29). O problema **P2**(b) pode então ser resolvido para **n**. Agora, pelo fato de Σ_2 ter sido calculado utilizando-se uma aproximação inicial para **N** (ou **n**), precisa-se iterar atualizando-se os dados de Σ_2 e resolvendo **P2**(a) novamente. Neste ponto aparece a quarta vantagem de trabalhar as matrizes **Y** e **Z** separadamente. Tem-se que a iteração não envolve variáveis do tipo 3.

Se se deseja separar os ajustes **a** das taxas de fluxo dos componentes $\tilde{x}_j$ na corrente j entre as taxas de fluxo totais $\mathbf{M}_j$ (tipo 1) e as concentrações $\mathbf{c}_j$, será necessário que se faça

$$\hat{\mathbf{c}}_j = \hat{x}_j / \hat{\mathbf{M}}_j \tag{4.33}$$

Isto leva a somente mais uma condição suplementar a ser especificada. Tendo suprimido o subscrito j para simplicidade, pode-se listar as seguintes condições:

a) Uma concentração c_i é exatamente conhecida. Então

$$\mathbf{M} = x_i / c_i \tag{4.34}$$

e a equação (4.33) fornece as demais concentrações.

b) A taxa de fluxo total **M** é conhecida exatamente. Então todos os **c** são calculados a partir da equação (4.33).

c) As concentrações devem relacionar-se a uma quantidade ρ especificada, por exemplo, as frações de massa ou molares de todas as espécies na corrente. Assim

$$\mathbf{1}^T\hat{\mathbf{c}} = \rho \tag{4.35}$$

onde **1** é um vetor de números um. Então

$$\mathbf{M} = \left(\mathbf{1}^{T}\hat{x}\right)/\rho \tag{4.36}$$

e c pode ser encontrado pela equação (4.33).

d) Na ausência de qualquer das condições acima, pode-se minimizar o peso da soma dos quadrados dos ajustes como segue:

$$\underset{M}{\mathbf{Min}}\ \mathbf{J} = \left[\left(\hat{\mathbf{c}} - \tilde{\mathbf{c}}\right)^{T}\Sigma_{c}^{-1}\left(\hat{\mathbf{c}} - \tilde{\mathbf{c}}\right) + \left(\hat{\mathbf{M}} - \tilde{\mathbf{M}}\right)^{2}/\sigma_{m}^{2}\right] \tag{4.37}$$

J é função somente de **M** porque c pode ser eliminado pela equação (4.33).

Se se fizer d**J**/d**M** = **0**, obtém-se

$$\hat{\mathbf{M}}_{4} - \tilde{\mathbf{M}}\hat{\mathbf{M}}_{3} + \sigma_{m}^{2}\left(\hat{\mathbf{M}}\tilde{\mathbf{c}} - \tilde{x}\right)^{T}\Sigma_{c}^{1}\hat{x} = \mathbf{0} \tag{4.38}$$

o qual pode ser prontamente resolvido usando o método de Newton, com uma aproximação inicial de $\hat{\mathbf{M}} = \tilde{\mathbf{M}}$.

4.6 ESTRUTURA DO PROGRAMA

No caso de sistemas lineares, tem-se a definição de uma matriz projeção ao passo que em sistemas não-lineares, tem-se a definição de duas dessas matrizes projeções. A primeira destas duas matrizes é coincidente em ambos os tipos de sistemas.

A determinação dos ajustes de dados é conseguida a partir da definição de três matrizes "básicas" **B**, **P** e Σ. A partir destas três matrizes, todas as demais são derivadas. São utilizadas propriedades descritas pela álgebra linear as quais permitem o isolamento das variáveis desejadas. Deste modo, a caracterização destas três variáveis é suficiente para caracterizar a resolução do problema de reconciliação.

4.6.1 DETERMINAÇÃO DA MATRIZ DOS BALANÇOS (B)

Inicialmente, a partir da estrutura do processo, constrói-se a matriz incidência a qual é transformada na matriz dos balanços mássicos **B** conforme a secção (3.2.2). Após cada grupo de **c** elementos, considera-se sempre a vazão total de cada corrente.

Foi criado então um ponteiro que nada mais é que um vetor de valores inteiros que possibilita a separação das colunas de **B**. No caso linear, tem-se três possibilidades para os valores das variáveis. Exatamente conhecidas, medidas ou não-medidas. Deste modo faz-se corresponder a cada uma destas possibilidades os números 0, 1 ou 3. Composto assim o vetor viv, vetor de informações das variáveis, pode-se associá-lo às respectivas colunas de **B**, separando-as respectivamente em $\mathbf{B}_0$, $\mathbf{B}_1$ e $\mathbf{B}_3$.

Para o caso não-linear tem-se a presença de componentes cujos valores de suas correntes totais não são medidos, apenas os valores de suas concentrações. Deste modo,

a matriz **B** passa, agora, a ter componentes com índice 2, representados em viv pelo mesmo algarismo. A matriz **B** separada passa a ter as sub-matrizes $\mathbf{B_0}$, $\mathbf{B_1}$, $\mathbf{B_2}$ e $\mathbf{B_3}$.

Na sua forma mais completa, a matriz **B** é formada pela matriz incidência **A**, expandida para se incluir os termos referentes à vazão total (**AE**), pela matriz estequiometria **S**, pelas restrições nas correntes do processo, como as de um divisor de fluxo e finalmente pelos vetores definidos pelas equação (3.3) os quais garantem a redundância às equações. Esquematicamente, a matriz pode ser assim representada:

$$\mathbf{B} = \left[\begin{array}{c|c} \mathbf{AE} & \mathbf{S^T} \\ \hline \mathbf{REST} & \mathbf{0} \\ \hline \mathbf{1^T} & \mathbf{-I} \end{array}\right] \tag{4.39}$$

onde,

REST ▷ significa restrições,
AE ▷ significa matriz A expandida.

É à matriz em (4.39) que se "aplica" o vetor viv. As colunas correspondentes a viv=1, 2 e 3 pertencerão às matrizes $\mathbf{B_1}$, $\mathbf{B_2}$ e **P** respectivamente.

Um segundo vetor vie, vetor de informações dos equipamentos também é definido para se simplificar o conjunto de equações que compõem as linhas das três matrizes decompostas. Como descrito na secção (3.2.1), é preciso retirar as linhas das matrizes de balanços que não façam parte das restrições. Assim vie é composto de "zeros" e "uns" conforme a restrição não pertença ou pertença ao problema, respectivamente. No caso da sub-matriz com os elementos do tipo 2 ($\mathbf{B_2}$), foi criada a matriz auxiliar $\mathbf{B_2F}$ a qual contém colunas correspondentes a viv = 2 e viv =100, é na verdade apenas um recurso para marcar os elementos pertencentes às variáveis do tipo 2 não-medidos. É portanto a equivalente a viv = 0 para variáveis do tipo 2.

4.6.2 MATRIZ VARIÂNCIA-COVARIÂNCIA

De acordo com o apresentado na secção (3.3), a matriz Σ é composta pelas variâncias-covariâncias das medidas $\tilde{x}$.

A variância de uma medida é conseguida a partir de um conjunto de dados da mesma, aqui contudo, os valores das variâncias ou foram aleatórios para se estudar o comportamento do sistema frente às mesmas ou então foram utilizados valores apresentados pelos autores seguidos em cada um dos exemplos apresentados. Para se utilizar erros aleatórios segundo uma distribuição normal, pode-se consultar a tabela constante em FREUND(1988).

A forma de determinação da estrutura da matriz variância se encontra detalhada no Apêndice A, mas pode ser simplificadamente descrita pela equação dada por MADRON et al.(1997).

$$\Sigma = \mathbf{MFM}\,\Sigma^{+}\,\mathbf{MFM^T} \tag{4.40}$$

onde

MFM representa a matriz das frações molares das medidas e
Σ^{+} representa a matriz das variâncias das correntes totais.

4.6.3 MATRIZ DOS VALORES NÃO-MEDIDOS (P)

A matriz **P** aparece quando se aplica o referido vetor viv à equação (4.39), que é a equação geral dos balanços. Esta matriz contém os termos desconhecidos do problema, referentes às correntes não medidas e às reações químicas.

Uma vez tendo **P**, é preciso como em $\mathbf{B_0}$, $\mathbf{B_1}$ e $\mathbf{B_2}$ retirar-se as linhas correspondentes a restrições não-existentes. O vetor vie aqui também é utilizado.

Partindo-se da matriz **P** em sua forma original, utilizando-se uma estratégia de eliminação transforma-se uma cópia **PL** da matriz **P** em uma matriz triangular superior.

Na diagonal encontram-se os pivôs (portanto elementos diferentes de zero) e no canto superior esquerdo se encontra a maior matriz não-singular, os demais elementos acima da diagonal principal podem ou não ser nulos. As trocas de linhas e colunas precisam ser registradas, o que é feito com os ponteiros ip e ic, respectivamente.

Tendo-se obtido **PL** na forma desejada, permuta-se a linhas e colunas da matriz original de acordo com os referidos ponteiros. Separa-se a matriz original da seguinte forma

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{P_1} & \mathbf{P_2} \\ \hline \mathbf{P_3} & \mathbf{P_4} \end{array}\right]$$

De tal forma que $\mathbf{P_1}$ corresponda à maior matriz quadrada não-singular contida em **P**. Os tamanhos das demais sub-matrizes é conseqüência desta separação. Excluindo-se as colunas de $\mathbf{P_1}$, as demais colunas pertencem a $\mathbf{P_3}$, do mesmo modo excluindo-se as linhas de $\mathbf{P_1}$ as demais pertencem a $\mathbf{P_2}$. O número de colunas de $\mathbf{P_2}$ é igual ao de $\mathbf{P_1}$ e o de $\mathbf{P_3}$ é igual ao de $\mathbf{P_4}$. Entre as linhas, o das sub-matrizes 1 e 3 são iguais, bem como 2 e 4.

Da definição apresentada na equação (3.14), tem-se

$$\mathbf{Y}^{T} = \left[-\mathbf{P_2}\mathbf{P_1}^{-1} \vdots \mathbf{I}\right]$$

aqui, não se trata de mera justaposição de sub-matrizes mas de produto matricial entre ambas. Apenas a matriz identidade é justaposta para adequar o tamanho de **Y**. Multiplica-se, finalmente, a matriz $\mathbf{Y}^{T}$ pela matriz **H** que é a matriz identidade com as colunas arranjadas segundo o ponteiro ip.

Se no processo todas as quantidades forem medidas, $\mathbf{P} = \mathbf{0}$ e então $\mathbf{Y}^{T} = \mathbf{I}$.

4.6.4 MATRIZ D

A determinação da matriz **Z** que torna a equação (4.13) verdadeira segue o mesmo procedimento descrito na secção anterior para o cálculo de **P**.

De posse da matriz **D**, determina-se a matriz que fornece a base do seu espaço nulo. Deste modo, a diferença aqui fica para a determinação da matriz propriamente dita a qual não é derivada diretamente da matriz **B** conforme o caso anterior.

Reescrevendo-se a equação (4.12), tem-se que a definição de **D** é

$$\mathbf{D}^T \equiv \left[\mathbf{B}_{21}\tilde{\mathbf{d}}_1\mathbf{B}_{22}\tilde{\mathbf{d}}_2\cdots\mathbf{B}_{2l}\tilde{\mathbf{d}}_l\cdots\right]^T\mathbf{Y}$$

Da análise desta equação, vê-se que a matriz **D** é composta pela multiplicação das linhas da sub-matriz $\mathbf{B}_2$ pelos vetores das frações molares dos componentes correspondentes às correntes com variáveis do tipo 2 (secção 4.2). É por este motivo que foi necessária a definição de $\mathbf{B}_2\mathbf{F}$. O fato de algumas correntes não ser composta por todos os elementos obriga a construir esta matriz auxiliar. Deste modo, no lugar de $\mathbf{B}_{2l}$, utiliza-se efetivamente $\mathbf{B}_2\mathbf{F}_l$.

Uma vez concluído o produto das linhas de $\mathbf{B}_2\mathbf{F}$ pelos valores de **d** em cada corrente, inverte-se a mesma, multiplicando-a a seguir por **Y**, que foi detalhado na secção anterior.

Os demais passos de determinação de **Z** são descritos na determinação de **Y**, não precisam ser , portanto, repetidos.

4.7 PROCEDIMENTOS PARA UTILIZAÇÃO DO PROGRAMA RECON

A obtenção de valores coerentes com os esperados para a reconciliação de dados de um processo depende da correta construção do arquivo de dados, seja para leitura automática ou para leitura via teclado (o programa RECON oferece estas duas possibilidades).

Já foi dito que este programa serve para reconciliar dados medidos em sistemas com restrições lineares e não-lineares. O programa é o auto-explicativo e conta com uma opção (I) a qual fornece instruções básicas quando for utilizado pela primeira vez. Aqui será contudo, efetuado um maior detalhamento do modo de apresentação de cada variável. A construção de todas as variáveis serão aqui detalhadas.

Algumas limitações do programa

*número máximo de nós possíveis é igual a 10;

*número máximo de correntes totais é igual a 20;

*número máximo de espécies químicas é igual a 10;

*número máximo de reações químicas é igual a 5;

*número máximo de restrições adicionais é igual a 3.

Estas limitações são devidas ao grande número de matrizes que precisam ser manipuladas durante a execução do programa. Para contornar tal problema, o ideal seria a utilização de matrizes esparsas, as quais ocupam muito menos memória por não precisar alocar espaço para elementos nulos, uma outra possibilidade é a utilização de microcomputadores com capacidade de memória estendida (acima de 8Mbytes de RAM). Isto não foi feito neste trabalho resultando nas limitações impostas acima. Este programa é compatível com equipamentos iguais ou superiores aos PC486 DX /66MHz.

Quando utilizar a entrada de dados via teclado, acompanha-se a seqüência em que os dados são pedidos diretamente na tela. No caso de dados compostos (não explicitados) nas mensagens via teclado e também no caso de se querer construir um arquivo de dados, tem-se a seguinte seqüência :

A-número de correntes totais (ni);

Digite aqui todas as correntes que entram e saem do sistema, inclusive as que estão ligadas ao nó meio-ambiente.

B- números de nós do sistema(mi);

Todos os nós exceto o nó meio-ambiente.

C-número de espécies químicas (c);

Reagentes, produtos e inertes desde que façam parte de pelo menos uma corrente.

D-número de reações químicas (rq);

Digite zero se não ocorrem reações químicas ou o número das mesmas caso ocorram.

E-números de restrições adicionais (nr);

Esta variável se refere por exemplo a divisores de fluxo, ou outras restrições que possam amarrar os componentes.

F-número de vazões medidas (q); TIPO 1

Aqui, q se refere ao número de vazões dos componentes mais o número de vazões totais medidas, referentes a $\mathbf{B}_1$.

G-número de vazões medidas (vd); TIPO 2

Para sistemas lineares vd = 0, pois não existem variáveis referentes a $\mathbf{B}_2$.

H-número de zeros referentes a $\mathbf{B}_2$ (zero);

Para sistemas lineares, zero = 0, pelo mesmo motivo acima.

I-tipo de apresentação dos dados (porc);

Digite 1 se forem apresentadas as correntes em porcentagem e 0 se forem em valores absolutos.

J-matriz das informações das correntes (**M**);

Esta matriz deve conter **ni** linhas por duas colunas, uma linha, portanto para cada corrente. A numeração das correntes pode ser aleatória, mas a partir daqui deve permanecer constante. A numeração dos nós também.

Em cada linha, então, digite dois valores, o primeiro referente ao nó em que a corrente entra e o segundo referente ao nó que a corrente sai. A ordem das linhas é a mesma das correntes. Caso a corrente entre ou saia do meio ambiente digite o valor 100.

L- vetor variância das correntes totais(varct);

Numa mesma linha digite **ni** valores correspondentes às variância das correntes totais. Quando uma corrente total não tiver o seu valor medido, acrescente o valor zero.

M-matriz das frações parciais (MFM);

Esta matriz deve conter **ni** linhas por (**c+1**)colunas. Cada corrente tomada em ordem numérica escreve-se na linha correspondente os valores das frações parciais dos componentes de 0 a 1. Quando um determinado componente pertencer àquela corrente mas não tiver o seu valor medido, deve-se atribuir o valor 2. Quando o componente não fizer parte da corrente, atribui-se o valor 0. No último elemento de cada corrente, se a mesma contiver apenas um elemento digite 0, se contiver mais que um digite 1 ou 2 conforme seja medido ou não.

N-matriz estequiométrica (TE);

Esta matriz conterá tantas linhas quantas forem as reações químicas. As colunas serão em número de **c** componentes mais duas unidades.

No primeiro elemento de cada linha digite o número do nó em que ocorre a reação química. Os coeficientes a seguir se referem ao coeficiente estequimétrico de cada componente naquela reação, obedecendo à convenção citada. Zero para componente inerte ou não participante, positivo para produtos e negativo para reagentes. No último elemento digite 0 se a corrente for de um único elemento ou 1 se não.

O-matriz das restrições (TR);

Tem número de linhas igual a **nr** e o número de colunas igual ao número de correntes totais. Cada linha representa uma restrição e cada coluna uma corrente total. Se a corrente não fizer parte da restrição, adiciona-se algum valor diferente de zero. Se se tem um divisor de fluxo tal que entra em uma corrente T5 e saem em T6 e T7, num sistema com 7 correntes totais, onde a vazão total de T6 é 2% de T5, tem-se

$$0{,}02T5-T6 = 0$$

a linha de TR correspondente conteria os seguintes elementos

$$0 \quad 0 \quad 0 \quad 0 \quad 0{,}02 \quad -1 \quad 0$$

As matrizes TR e TE só precisam ser escritas se **rq** e **nr** forem diferentes de zero.

P-vetor das vazões medidas (x);

Este vetor terá o tamanho **q** e deverá ser escrito em uma única linha. As vazões dos componentes e das correntes devem obedecer à ordem numérica de ocorrência.

Q-vetor de informações das variáveis (viv);

Este vetor terá o tamanho da matriz definida na equação (3.26) o qual é ($\mathbf{ni.(c+1)+rq}$). Serão, portanto **mi** blocos de (**c+1**) elementos os quais correspondem às vazões dos componentes, seguida da vazão total daquela corrente. Os blocos podem estar ou não na mesma linha. Para elementos cujos componentes não pertencem àquela corrente, atribui-se o valor zero. Se o componente fizer parte da corrente e tiver o valor medido, atribua o valor 1. Se o elemento pertencer à corrente mas não tiver o valor medido, atribua o valor 3. No elemento de viv que, for referente à corrente total, atribua zero quando a corrente for monofásica. Quando a corrente contiver mais de um elemento e pelo menos um for medido, atribua 1, se não houver nenhum valor medido, atribua 3.

No bloco final de rq elementos de viv, atribua o valor 3 pois não são medidos e dizem respeito à conversão de cada reação química.
R-vetor das informações dos equipamentos (vie);

Este vetor simplificará as linhas da matriz representada pela equação (3.26), terá portanto o mesmo número de elementos tantas quantas forem a linhas daquela. Serão, portanto, $(\mathbf{mi}.(\mathbf{c}+1)+\mathbf{nr}.(\mathbf{c}+1)+\mathbf{ni})$ elementos. Cada grupo de (**c+1**) elementos diz respeito a um nó ou restrição adicional. Se a espécie química não fizer parte daquele nó ou daquela restrição, atribui-se o valor zero no elemento de vie, caso contrário, atribua o valor 1. Aos **ni** últimos elementos, os quais correspondem às equações dos balanços individuais, atribua zero ao elemento que representar uma corrente com apenas um componente e 1 se a corrente contiver mais que um componente. Os blocos podem ou não ser escritos na mesma linha.

O tipo de cada valor a ser fornecido ao programa está especificado no programa. Onde for necessário variáveis inteiras ou reais está especificamente destacado.

5 DETECÇÃO DE ERROS GROSSEIROS

5.1 INTRODUÇÃO

Toda medida feita em uma planta industrial está sujeita a erros. Tais erros podem ser de diversas naturezas e o conhecimento desta natureza é muito importante quando se faz o controle do processo. Os erros podem ser aleatórios, aqui normalmente considerados gaussianamente distribuídos, com média zero. Podem ser erros devido à descalibração de instrumentos, os quais podem ser sistemáticos mas não gaussianamente distribuídos. Finalmente, existem os erros grosseiros, devidos a vazamentos nas linhas de transmissão, quedas de tensão nos aparelhos de medida elétricos ou de pressão nos pneumáticos.

Quando se pensa na reconciliação de dados, conforme detalhado nos capítulos 3 e 4, apenas os aleatórios podem estar presentes. São erros com comportamento "previsível" do ponto de vista estatístico e, portanto, podem ser levados em conta para ajustar os valores medidos no sistema. A presença de erros não aleatórios no conjunto de medidas e consequentemente nas restrições do sistema, inviabiliza a reconciliação de dados. Os valores corrigidos poderão guardar pouca relação com os valores reais do processo. É necessário retirar as medidas corrompidas por erros grosseiros do conjunto de valores medidos, quando possível, (NARASIMHAN(1987), deletando-as simplesmente ou retificando-as. Os procedimentos requeridos para um procedimento ou outro serão tratados mais tarde , ainda neste capítulo.

A princípio, é necessário saber que não existem algoritmos, confiáveis, que sozinhos, corrijam mais que dois erros presentes num conjunto de medidas simultaneamente. Isto torna tedioso o trabalho de retificação por requerer recursivas tentativas até se encontrar a medida ou conjunto de medidas que apresentam erros grosseiros dentro do universo medido.

Neste trabalho foi desenvolvido um programa computacional que utilizando dados intermediários do programa RECON, "confere" os balanços de dados apresentados a fim de detectar a possível presença de erros grosseiros entre as variáveis medidas. Segundo alguns autores, CROWE(1989) por exemplo, é preciso sempre se fazer mais que um teste estatístico pois a utilização de um único teste pode levar a conclusões erradas a respeito da presença de possíveis erros grosseiros.

Assim como é preciso separar sistemas com restrições lineares dos com restrições não-lineares quando se deseja fazer a reconciliação, na detecção de erros grosseiros também é preciso se fazer esta separação.

5.2 A RETIFICAÇÃO DE DADOS E A OBSERVABILIDADE

5.2.1 RETIFICAÇÃO

Chama-se retificação de dados ao procedimento de correção de valores que apresentam erros grosseiros. A retificação não é um procedimento isolado, ou seja, não se pode retificar os dados sem, obviamente, antes identificá-los (detecção) e em antes classificá-los (classificação).

Neste capítulo serão descritas as bases da detecção dos erros grosseiros, contudo a classificação de variáveis é um procedimento que está fora do universo deste trabalho, não sendo desenvolvido nenhum procedimento computacional para tratá-lo. Isto não inviabiliza a utilidade do proposto aqui, uma vez que é possível ao menos saber com o apresentado se o conjunto de dados contém ou não erros grosseiros.

Recordando a figura (1.1), a classificação de variáveis é o pressuposto inicial da reconciliação de dados. Aqui, sempre se parte da consideração de que os dados já estão classificados e sempre que não medidos são no mínimo observáveis.

A título de informação suplementar, será apresentado o procedimento teórico de observabilidade e redundância de dados de processos para reconciliação de dados em estado estacionário, utilizando projeção matricial.

5.2.2 OBSERVABILIDADE E REDUNDÂNCIA

CROWE(1989) chama a atenção para a questão primeira que surge quando se quer avaliar um conjunto de medidas de taxas de fluxos e concentrações em processos em estado estacionário. É possível se determinar de maneira unívoca os dados não-medidos a partir dos disponíveis? Em caso afirmativo, se estará diante de variáveis observáveis, de outro modo, serão não-observáveis e os valores calculados a partir desse conjunto não terá confiabilidade. Em segundo, surge a questão sobre uma quantidade medida poder ser determinada caso não fosse medida. Se isto é possível, as medidas são chamadas de redundantes caso contrário serão essenciais.

A partir desta definições, pode-se concluir, da literatura, que não é possível se fazer a retificação dos dados sem que se tenha medidas redundantes. Baseado neste conceito, será apresentado um método direto de classificação de quantidades não-medidas em observáveis e não-observáveis e também de identificação de quantidades medidas em redundantes ou não. O método é baseado em matrizes que descrevem a estrutura da planta ou em qualquer conjunto de valores consistentes com as restrições.

Será apresentado o método em sua forma teórica apenas com o objetivo de orientar um possível detalhamento posterior do procedimento.

5.2.2.1 DESCRIÇÃO

Desde que medidas de taxas de fluxos e concentrações em processos operando em estado estacionário estão sujeitos a erros randômicos, os dados são reconciliados utilizando as leis da conservação e outras restrições sendo os dados ajustados da melhor maneira possível. Antes ou durante a reconciliação, é importante saber quais as variáveis não-medidas poderiam ser unicamente determinadas, a partir dos valores medidos, de modo a ainda satisfazerem as restrições impostas. É muito importante saber quais as variáveis não-medidas não podem ter seus valores unicamente determinados. O projetista de um processo deve colocar os instrumentos de medida tal que quantidades específicas sejam medidas ou determináveis.

Uma medida é dita ser redundante se ela pode ser calculada a partir dos demais dados medidos. A redundância leva em geral a conflitos entre dados e balanços.

Um balanço em volta de uma secção de um processo é dito ser redundante se todo o fluxo que entra ou que sai daquela secção é medido. Neste caso, cada uma das medidas é redundante. Se um balanço não é redundante, não há possibilidade de conflito com as leis de conservação desde que as medidas faltantes podem ser escolhidas para satisfazer o balanço. Contudo, isto pode ser fonte de erros muito grandes se o sistema contiver erros fora da tolerância para o sistema em estudo, um exemplo disto pode ser visto no capítulo 6, EXEMPLO 2. Redundância também, ocorre entre as medidas de uma corrente onde todas as taxas de fluxo dos componentes são medidas. Um ótimo exemplo de redundância surge quando se tem medidas as concentrações em cada corrente tantas quantas forem as correntes incidentes no processo e somente uma taxa de fluxo total medida. Neste caso, há mais equações de balanço que taxas de fluxo total desconhecidas.

VACLAVEK(1969) introduziu o conceito de redundância em que cada taxa de fluxo medida pode ser colocada em um de dois grupos, de acordo se ela pode ou não ser calculada a partir de outro valor medido. Também definiu taxas de fluxo não-medidas observáveis como aquelas que podem ser unicamente determinadas a partir das taxas de fluxo ajustadas. Ele mostrou também que uma condição necessária e suficiente para uma taxa de fluxo não-medida ser observável, para um componente por corrente, era ela estar ou em um caminho a partir de uma corrente externa para outra, ou em um ciclo, onde aquele fluxo não é medido em nenhuma corrente.

VACLAVEK et al.(1976a,b) estenderam estas idéias para sistemas multicomponentes. Apresentaram passos para eliminar as quantidades desconhecidas de valores de fluxos baseados nas propriedades da solução ótima para o problema da reconciliação. O método proposto é limitado em:

(1) ou todas as concentrações em uma corrente são medidas, ou nenhuma delas é,

(2) o algoritmo é baseado em restrições de igualdade contendo ou um simples ± 1 para a entrada e a saída ou, separadamente, +1, -1 para cada fluxo medido,

(3) quantidades medidas são definidas como não-redundantes se o ajuste é nulo,

(4) a restrição que as taxas de fluxos das espécies soma-se às taxas de fluxo total é levada em conta a fim de eliminar um balanço por nó, ao invés de adicionar uma restrição por corrente.

ROMAGNOLI e STEPHANOPOULOS(1980), desenvolveram um algoritmo para a identificação de quantidades observáveis e redundantes. Suas análises são similarmente limitadas porque devem assumir que a composição de uma corrente ou é completamente medida ou não é nada medida. Eles também assumiram que medidas não-redundantes deveriam ser corrigidas pela reconciliação, excluindo assim, explicitamente a covariância entre os dados redundantes e os não.

MAH e seus colaboradores (MAH et al.(1976); STANLEY e MAH(1981a,b); KRETSOVALIS e MAH(1987,1988a,b)) apresentaram uma teoria e algoritmos para a classificação das quantidades não-medidas como observável ou não-observável e das quantidades medidas como redundante ou não-redundante. KRETSOVALIS e MAH(1988a,b) incluíram medidas colocadas arbitrariamente, reações químicas divisores de fluxo e fluxos de energia puro, em seus trabalhos. Basearam-se na utilização dos grafos, envolvendo nós correspondentes para restrições conectadas por linhas correspondentes a taxas de fluxo. A aplicação de sua teoria requer a construção de vários diferentes grupos de subgrafos e o teste de um grande número de teoremas a respeito da obervabilidade e da redundância.

A proposta do trabalho de CROWE(1989)é apresentar um caminho alternativo a estas classificações usando a projeção matricial. O caminho a ser apresentado aqui evitará as limitações descritas.

RESTRIÇÕES

A forma geral da conservação e de outras restrições impostas em quantidades extensivas, usando as mesmas notações dos capítulos 3 e 4, é

$$\mathbf{B}_1 x + \mathbf{B}_2 \mathbf{N} \mathbf{d} + \mathbf{P} \mathbf{v} = \mathbf{0} \qquad (5.2.1)$$

Os três termos no lado esquerdo correspondem respectivamente ao fluxo das espécies medidas (tipo 1) às concentrações medidas com taxas de fluxos totais não-medidas (tipo 2) e taxas de fluxos não-medidas e extensões das reações (tipo 3). A matriz diagonal **N**, contém as taxas de fluxo não-medidas na categoria (ou tipo) 2, com tantas entradas, para cada corrente, quantas forem as concentrações $\underline{\mathbf{d}}$ medidas. A fim de mostrar o vetor $\underline{\mathbf{n}}$ dos fluxos não-medidos, define-se **C** por

$$\mathbf{C}(\mathbf{d})\mathbf{n} \equiv \mathbf{B}_2 \mathbf{N} \mathbf{d} \qquad (5.2.2)$$

Cada coluna da matriz **C** corresponde a uma corrente e contém a soma das colunas de $\mathbf{B}_2$ para aquela corrente, sendo que cada coluna é multiplicada pela correspondente concentração medida.

Além do mais, para as restrições acima, pode haver restrições adicionais impostas por variáveis intensivas. Exemplos deste tipo são as igualdades de concentrações das espécies à entrada e à saída de um divisor de fluxo e o equilíbrio de fases entre reações químicas. Essas serão referidas como restrições intensivas.

Definições e Lemas

Define-se observabilidade por

Definição 1: Uma quantidade, em um processo em estado estacionário, é observável se e somente se ela pode ser unicamente determinada a partir de grupos fixos de valores, correspondentes a taxas de fluxos e concentrações medidas, as quais são consistentes com todas as restrições dadas.
Qualquer quantidade não-medida a qual não é assim determinável, é não-observável.

Isto é compatível para o problema específico tratado aqui, com uma definição matematicamente mais precisa, para o caso geral a partir da definição local de STANLEY e MAH(1981a). Deve-se notar que o conjunto de valores correspondentes às medidas pode ser qualquer conjunto fixo o qual seja consistente com as restrições. Isto garante que no mínimo uma solução para as quantidades não-medidas pode ser encontrada. Para se ter um conjunto consistente de dados não é preciso se aplicar a reconciliação de dados.

Redundância é definida da seguinte maneira por MAH(1981a)

Definição 2: Uma quantidade medida é redundante se e somente se ela fosse observável se uma quantidade não fosse medida. De outra forma, a quantidade medida é não-redundante.

Os quatro lemas seguintes dão a base dos testes para classificação das quantidades não-medidas como observável ou não-observável. Os dois primeiros dão as condições suficientes para encontrar taxas de fluxos não-observáveis nas categorias 3 e 2 respectivamente. O lema 3 apresenta as condições suficientes e necessárias para taxas de fluxos não-observáveis. No lema 4, condições necessárias e suficientes para concentrações das espécies observáveis na categoria 3 são dadas. Para os fluxos não-medidos na categoria 3, tem-se

Lema 1: Se existe um vetor não nulo $\mathbf{t}$ tal que $\mathbf{Pt} = \mathbf{0}$, então, cada quantidade extensiva não-medida correspondente a um elemento não-zero de $\mathbf{t}$ é não-observável, a menos que ele seja unicamente determinado por uma restrição intensiva.

Prova: Suponha que haja uma solução $\mathbf{v}$ a qual satisfaz às restrições na equação (5.2.1). Então, se $\mathbf{t} \neq \mathbf{0}$ e $\mathbf{Pt} = \mathbf{0}$, o vetor $\mathbf{v} + \alpha\,\mathbf{t}$ também satisfaz àquelas restrições para qualquer escalar α. Assim, o vetor $\mathbf{v}$ não é único e cada elemento correspondente ao elemento não nulo de $\mathbf{t}$ é não-observável, a menos que ele seja unicamente determinado por uma restrição intensiva. Um elemento de $\mathbf{v}$ é determinado por uma restrição intensiva se sua concentração é assim determinada e o fluxo total naquela corrente é medido ou observável.

Lema 2: Se um vetor $\mathbf{w} \neq \mathbf{0}$ existe tal que $\mathbf{C}(\mathbf{d}^+)\mathbf{w} = \mathbf{0}$, onde $\mathbf{d}^+$ é consistente com as restrições, então cada elemento de $\mathbf{n}$ correspondente a um elemento não nulo de $\mathbf{w}$ é não-observável, a menos que seja observável através de uma restrição intensiva.

Prova: A prova é análoga ao Lema 1. O requerimento de que um conjunto consistente de valores de $\mathbf{d}$ seja usado é necessário para que haja no mínimo uma solução para as quantidades não-medidas. Um fluxo total, $\mathbf{n}_k$, seria observável como a razão de um fluxo de uma espécie e sua concentração na categoria 3 naquela corrente, se o último fosse observável através de uma restrição intensiva.

Uma condição necessária e suficiente para a não-observabilidade é dada pelo Lema seguinte

Lema 3: Se e somente se existe um vetor não-nulo

$$\begin{bmatrix} \mathbf{w} \\ \cdots \\ \mathbf{t} \end{bmatrix} \tag{5.2.3}$$

tal que

$$\mathbf{C}(\mathbf{d}^+)\mathbf{w} + \mathbf{Pt} = \mathbf{0} \tag{5.2.4}$$

onde $\mathbf{d}^+$ é um vetor fixo de concentrações consistentes, cada elemento de $\mathbf{n}$ e $\mathbf{v}$ correspondente a um elemento não-zero de $\mathbf{w}$ e $\mathbf{t}$, respectivamente, é não-observável, e é garantido também que é não-observável através de restrições intensivas.

Prova: A prova da não-observabilidade dada pelas equações (5.2.3) e (5.2.4)segue novamente o exposto no Lema 1.

Suponha que alguns elementos são não-observáveis tal que haja duas diferentes soluções $(\mathbf{n}_i, \mathbf{v}_i; \mathbf{i} = \mathbf{1,2})$ para as quantidades não-medidas. Desde que ambas satisfaçam

às equações de restrições (5.2.1), bem como às restrições intensivas, a subtração de cada restrição (5.2.1), usando os outros termos cancelados na categoria 1 é dado por

$$\mathbf{C}^{+}(\mathbf{n}_1 - \mathbf{n}_2) + \mathbf{P}(\mathbf{v}_1 - \mathbf{v}_2) = \mathbf{0} \qquad (5.2.5)$$

Para cada elemento não-observável existem duas diferentes soluções, assim o vetor (5.2.3) existe e satisfaz à equação (5.2.4).

Uma concentração não-medida (ou, mais geralmente, uma quantidade intensiva correspondente a uma propriedade de estado) é observável se, mas não somente se, o fluxo daquelas espécies é observável e o fluxo total naquela corrente é observado ou medido. Se ambos, as espécies e os fluxos totais, são não-observáveis, é possível à concentração ser observável. Uma definição precisa destas condições para a observabilidade das concentrações é

Lema 4: Uma concentração não-medida, $\mathbf{c}_{ik}$, da espécie i na corrente k é observável se e somente se uma das três seguintes condições se confirma:

(a) a taxa de fluxo da espécie i, $\mathbf{v}_{ik}$, é observável e a taxa de fluxo total, $\mathbf{v}_{0k}$, na corrente k é observável ou medida.
ou
(b) nem as taxas de fluxo da espécie i nem a taxa de fluxo total na corrente k é observável e nenhuma mudança factível naqueles fluxos (como mudanças que são consistentes com as restrições) são restringidas pela equação

$$\mathbf{v}_{jk} = \alpha\, \delta\, \mathbf{v}_{jk} \qquad (\mathbf{j = 0 \quad ou \quad 1}) \qquad (5.2.6)$$

para o escalar α . Aqui $\delta\, \mathbf{v}_{jk}$ é uma mudança factível em $\mathbf{v}_{jk}$ dos fluxos das espécies (j=1) ou o fluxo total (j=0), o qual mantém as restrições à força.
ou
(c) há uma restrição intensiva a partir da qual a concentração das espécies i na corrente k pode ser unicamente determinada como função de uma quantidade intensiva medida e observável.

Prova: (a) Se o fluxo das espécies é observável e o fluxo total é observável ou medido cada um é unicamente determinado e desta forma as suas razões. Deste modo, a concentração é observável.
(b) Suponha que o fluxo das espécies i e fluxo total na corrente k sejam ambos observáveis. Se a equação (5.2.6) se verifica, a razão entre o fluxo das espécies i e o fluxo total em uma factível $\mathbf{v}_k$ é invariante tal que a concentração das espécies i seja observável.
(c) Se esta condição se verifica, a concentração, $\mathbf{c}_{ik}$, é observável.

De outro modo, suponha que nenhuma das condições (a), (b) ou (c) se verificam. Se um dos fluxos é não-observável e o outro é ou medido ou observável, então o fluxo não-observável não é unicamente determinado enquanto o outro é único. Sua razão é então não-única e a concentração é não-observável. Se ambos fluxos são não-observáveis e mudanças factíveis em $\mathbf{v}_k$ não são limitadas pela equação (5.2.6), a razão

entre o fluxo das espécies i e o fluxo total pode ser mudado, tal que a concentração seja não-unicamente determinável e assim não-observável.

Observações: A não-observabilidade consistente com a condição (b) no Lema 4 não pode envolver extensões de reações não-observáveis desde que a reação mudará a taxa de fluxo das espécies mas deixará a taxa de fluxo de massa total constante, fazendo a equação (5.2.6) se tornar inválida.

Define-se a taxa de fluxo de uma espécie na categoria 2 ser observável se e somente se a taxa de fluxo total naquela corrente for observável.

REDUÇÃO DAS RESTRIÇÕES

A solução para o problema de reconciliação pode ser efetivamente simplificado eliminando-se as variáveis não-medidas a partir das equações de restrição (5.2.1). Isto foi feito usando matrizes ortogonais à matriz **P**, a chamada projeção matricial.

A matriz $\mathbf{Y}^T$ é construída para ser de máximo posto e ser ortogonal às colunas de **P** tal que,

$$\mathbf{Y}^T\mathbf{P} = \mathbf{0} \tag{5.2.7}$$

a matriz **P** é particionada em

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{P}_1 & \mathbf{P}_2 \\ \hline \mathbf{P}_3 & \mathbf{P}_4 \end{array}\right] \tag{5.2.8}$$

onde

$\mathbf{P}_1$ é a máxima matriz quadrada não-singular. Não há perda de generalidade em se assumir que ela está no canto superior esquerdo desde que as colunas da matriz podem ser rearranjadas e as permutações das linhas apenas corresponderiam às permutações das colunas de $\mathbf{Y}^T$. Pode-se escolher

$$\mathbf{Y}^T = \left[-\mathbf{P}_2\mathbf{P}_1^{-1} \middle| +\mathbf{I}\right] \tag{5.2.9}$$

Note que as colunas da parte direita de **P** são dependentes das do lado esquerdo, o que implica que a equação (5.2.7) é válida para $\mathbf{Y}^T$ da equação (5.2.9).

As restrições, equação (5.2.1) com Equação (5.2.2), são então multiplicadas por $\mathbf{Y}^T$ para dar, em geral

$$\mathbf{Y}^T\mathbf{B}_1 x + \mathbf{Dn} = \mathbf{0}\,, \tag{5.2.10}$$

$$\mathbf{D} \equiv \mathbf{Y}^T\mathbf{C} \tag{5.2.11}$$

A matriz **D** é, então, particionada em

$$\mathbf{D} = \left[\begin{array}{c|c} \mathbf{D}_1 & \mathbf{D}_2 \\ \hline \mathbf{D}_3 & \mathbf{D}_4 \end{array}\right] \tag{5.2.12}$$

com $\mathbf{D}_1$ sendo a maior matriz quadrada não-singular contida em **D**. A matriz **Z** é definida como

$$\mathbf{Z}^T\mathbf{D} = \mathbf{0} \tag{5.2.13}$$

com

$$\mathbf{Z}^T = \left[-\mathbf{D}_2\mathbf{D}_1^{-1} \vdots \mathbf{I}\right] \tag{5.2.14}$$

a fim de eliminar os termos do tipo 2 das restrições.

DETERMINAÇÃO DAS QUANTIDADES NÃO-OBSERVÁVEIS

Para determinar quais quantidades não-medidas seriam não-observáveis sem as restrições intensivas, busca-se uma solução geral para **w** e **t** na equação (5.2.4). A solução geral usando somente colunas de **P** é

$$\mathbf{w} = \mathbf{0};\ \ \mathbf{t} = \begin{bmatrix} -\mathbf{P}_1^{-1}\mathbf{P}_3 \\ \hline \mathbf{I} \end{bmatrix}\mathbf{u}_k \tag{5.2.15}$$

para qualquer vetor unitário conformável $\mathbf{u}_k$. Cada elemento de **v** correspondente ao elemento não-zero de **t** é não-observável pelo Lema 1, desde que **Pt** = **0** da equação (5.2.8). Claramente os elementos de **v** correspondentes às colunas de $\mathbf{P}_3$ são sempre não-observáveis. Um procedimento formalmente equivalente para encontrar **t** para o caso linear foi dado por DAROUACH e al.(1986).

A solução geral de (5.2.4) incluindo as colunas de **C** é

$$\mathbf{w} = \begin{bmatrix} -\left(\mathbf{D}_1^+\right)\mathbf{D}_3^+ \\ \hline \mathbf{I} \end{bmatrix} \tag{5.2.16}$$

para qualquer vetor unitário conformável $\mathbf{u}_j$ e

$$\mathbf{t} = \begin{bmatrix} -\mathbf{P}_1^{-1}\mathbf{C}_1^+\mathbf{w} \\ \hline \mathbf{0} \end{bmatrix} \tag{5.2.17}$$

onde $\mathbf{C}_1$ é a linha-partição de **C** correspondente às linhas de $\mathbf{P}_1$. Notar que **C** é avaliada com valores consistentes de **d** tal que a partição de **D** deve diferir daquela que usa valores medidos de **d**. Novamente, as linhas não-zero da matriz que multiplica $\mathbf{u}_j$ dá os elementos não-observáveis de **n**. Quaisquer elementos correspondentes, não-nulos, de **t** indicam juntamente elementos não-observáveis de **v**, pelo Lema 3. Todos os elementos de **n** e de **v** os quais não foram classificados como não-observáveis são, então, observáveis. A partir das soluções gerais (5.2.15 - 5.2.17), é claro que todas as quantidades não-medidas são observáveis se **P** e **D** têm posto máximo, e $\mathbf{P}_3$ e $\mathbf{D}_3$ têm colunas vazias. As restrições intensivas podem então ser "quebradas" para redesignar algumas quantidades não-observáveis como observáveis.

O caso especial para $\mathbf{D} = \mathbf{0}$ deve ser observado. Aqui, $\mathbf{D}_1$ é vazio de modo que sua inversa é também vazia e $\mathbf{D}_3$ tem uma linha vazia mas não tem as colunas vazias. Pode-se aplicar as regras de multiplicação de matrizes para aquelas com linhas e/ou colunas vazias. Então, $\mathbf{w}$ na equação (5.2.16) transforma-se em alguma coluna da matriz identidade, com ordem igual ao número de colunas de $\mathbf{C}$.

LEMAS PARA A REDUNDÂNCIA

Quatro casos de redundância podem ser distinguidos, a saber, uma taxa de fluxo de uma espécie, uma concentração ou uma taxa de fluxo total na categoria 1, e uma concentração na categoria 2. Os quatro Lemas seguintes proporcionam os testes necessários para se decidir quais medidas, dentro de cada um dos quatro casos, são não-redundantes.

Lema 5: Uma taxa de fluxo de uma espécie x na categoria 1 é não-redundante se e somente se

$$\left[\mathbf{Z}^{+}\right]^{\mathrm{T}} \mathbf{Y}^{\mathrm{T}}\mathbf{b} = \mathbf{0} \tag{5.2.18}$$

onde $\mathbf{b}$ representa as colunas de $\mathbf{B}_1$ correspondentes àquelas espécies que em $\mathbf{Z}^{+}$ são avaliadas a partir de $\mathbf{D}\left(\mathbf{d}^{+}\right)$ e sua concentração não poderia ser unicamente determinada a partir de restrições intensivas.

Prova: Se as taxas de fluxos das espécies i na corrente k, um elemento x_{ik} da categoria 1, é deletado, a coluna $\mathbf{b}$ correspondente de $\mathbf{B}_1$ é adicionada a $\mathbf{P}$. Por definição, o elemento é não-redundante se e somente se ele é não-observável quando deletado. Se a concentração $\mathbf{c}_{ik}$ puder ser unicamente determinada a partir das restrições intensivas, as taxas de fluxos das espécies serão observáveis, quando deletadas, como o produto da concentração e a taxa de fluxo total medida.

Assuma que a concentração $\mathbf{c}_{ik}$ não possa ser unicamente determinada através das restrições intensivas. Pelo Lema 3, o elemento é observável se e somente se

$$\mathbf{b} = \mathbf{C}^{+}\mathbf{w} + \mathbf{P}\mathbf{t} \tag{5.2.19}$$

para algum $\mathbf{w}$, $\mathbf{t}$ e $\mathbf{C}^{+}$ avaliado usando concentrações consistentes $\mathbf{d}^{+}$. Isto implica que

$$\left(\mathbf{Z}^{+}\right)\mathbf{Y}^{\mathrm{T}}\mathbf{b} = \left(\mathbf{Z}^{+}\right)^{\mathrm{T}} \mathbf{Y}^{\mathrm{T}}\mathbf{C}^{+}\mathbf{w} = \left(\mathbf{Z}^{+}\right)^{\mathrm{T}} \mathbf{D}^{+}\mathbf{w} = \mathbf{0} \tag{5.2.20}$$

De outro modo, assume-se que

$$\left(\mathbf{Z}^{+}\right)^{\mathrm{T}} \mathbf{Y}^{\mathrm{T}}\mathbf{b} = \mathbf{0} \tag{5.2.21}$$

A definição de $\mathbf{Z}^T$ na equação (5.2.13) implica que para algum **w**,

$$\mathbf{Y}^T\mathbf{b} = \mathbf{D}^+\mathbf{w} = \mathbf{Y}^T\mathbf{C}^+\mathbf{w} \tag{5.2.22}$$

e então,

$$\mathbf{Y}^T\left(\mathbf{b} - \mathbf{C}^+\mathbf{w}\right) = \mathbf{0} \tag{5.2.23}$$

Assim, a partir da definição de **Y** na equação (5.2.7)

$$\mathbf{b} = \mathbf{C}^+\mathbf{w} + \mathbf{P}\mathbf{t} \tag{5.2.24}$$

para algum **t**.

Consequentemente, os elementos não-redundantes de x podem ser encontrados a partir das colunas zero de $\mathbf{Z}^{+T}\mathbf{Y}^T\mathbf{B}_1$. Isto não quer dizer que os elementos não-redundantes recebem ajuste zero durante a reconciliação, desde que a covariância entre aqueles elementos e uma medida redundante pode levar ao ajuste de um elemento não-redundante. Somente se a matriz variância é diagonal, o conjunto de elementos com ajuste zero coincidirá com os elementos não-redundantes, conforme o caso tratado por VACLAVEK et al.(1967a,b).

Para concentrações em correntes na categoria 1, tem-se

Lema 6: A concentração das espécies i na categoria 1 correspondentes às suas taxas de fluxos na correntes k, a saber elemento x_{ik} de x, é redundante se e somente se ou

(a) as taxas de fluxos das espécies é redundante.
ou
(b) é unicamente determinada a partir de outras quantidades intensivas medidas e observáveis através de restrições intensivas.

Prova: Se a concentração correspondente à taxa de fluxo das espécies, x_{ik}, é deletada, a taxa de fluxo correspondente é transferida para a taxa de fluxo na categoria 3, $\mathbf{v}_{ik}$. Desde que o fluxo total para a corrente permanece medido, pelo Lema 4, a concentração correspondente a $\mathbf{v}_{ik}$ é observável se e somente se ou a taxa de fluxo $\mathbf{v}_{ik}$ é observável ou a concentração é determinada a partir de restrições intensivas.

O seguinte Lema trata a redundância de uma taxa de fluxo total de uma corrente:

Lema 7: A taxa de fluxo total medida de uma corrente k é não-redundante se e somente se ambos

$$\text{(a)} \left(\mathbf{Z}^{+}\right)^{\mathrm{T}} \mathbf{Y}^{\mathrm{T}} \mathbf{B}_{1k} x_{k}^{+} = \mathbf{0} \qquad (5.2.25)$$

onde $\mathbf{B}_{1k}$ é o conjunto de colunas de $\mathbf{B}_1$ para a corrente k e x_k^+ é um vetor de taxa de fluxos consistentes de espécies para aquela corrente
e
(b) não puder ser unicamente determinada a partir de outras quantidades medidas e observáveis através de restrições intensivas.

Prova: Se a taxa de fluxo total da corrente k é medida, então ela e qualquer espécie com concentração medida estão na categoria 1. Se aquela taxa de fluxo total for deletada, as colunas de $\mathbf{B}_1$ para aquela corrente serão transferidas para a categoria 2, com concentrações ajustadas $\mathbf{d}_k^+$ proporcionais a x_k^+ e as colunas de $\mathbf{B}_{1k}\mathbf{d}_k^+$ justapostas a $\mathbf{C}^+$. As taxas de fluxo totais k serão não não-redundantes se e somente se elas forem não-observáveis quando deletadas. O argumento segue o do Lema 5, substituindo **b** por $\mathbf{B}_{1k}\mathbf{d}_k^+$.

Para as concentrações nas correntes na categoria 2, tem-se

Lema 8: A concentração medida $\mathbf{d}_{ik}$ das espécies i da categoria 2 na corrente k é não-redundante se e somente se ambos

$$\text{(a)} \left(\mathbf{Z}^{+}\right)^{\mathrm{T}} \mathbf{Y}^{\mathrm{T}} \mathbf{B}_{2,ik} = \mathbf{0} \qquad (5.2.26)$$

onde $\mathbf{B}_{2,ik}$ é a coluna de $\mathbf{B}_2$ para as espécies i na corrente k, e

(b) não é unicamente determinada por restrições intensivas.

Prova: Uma concentração medida é redundante se ela pode ser unicamente determinada, quando deletada, a partir de outras quantidades através de restrições intensivas. Assim a condição (b) é necessária. Deve-se considerar as duas possibilidades de um fluxo total $\mathbf{n}_k$ em uma corrente k ser não-observável ou observável.

(1) Assuma que $\mathbf{n}_k$ é não-observável. Então há mudanças factíveis em **n** e **v** tais que

$$\mathbf{C}_k \delta\, \mathbf{n}_k'' + \mathbf{C}'' \delta\, \mathbf{n}'' + \mathbf{P} \delta\, \mathbf{v} = \mathbf{0} \qquad (5.2.27)$$

onde $\mathbf{C}_k$ é a coluna de **C** para a corrente k, $\mathbf{C}''$ contém as colunas remanescentes de **C** correspondentes ao fluxo total $\mathbf{n}''$, e $\delta\, \mathbf{n}_k$ é não-zero. Pode-se separar as contribuições das espécies i para $\mathbf{C}_k$ desde que

$$\mathbf{C}_k = \sum_{j=0}^{c_k} \mathbf{B}_{2,jk}\mathbf{d}_{jk} \tag{5.2.28}$$

com j = 0 correspondente ao fluxo total. Aqui, $\mathbf{c}_k$ é o número de espécies na categoria 2 na corrente k. Então, a equação (5.2.27) torna-se

$$\mathbf{B}_{2,ik}\mathbf{d}_{i,k}\delta\,\mathbf{n}_k + \mathbf{C}_k^{-}\delta\,\mathbf{n}_k + \mathbf{C}''\delta\,\mathbf{n}'' + \mathbf{P}\delta\,\mathbf{v} = \mathbf{0} \tag{5.2.29}$$

onde $\mathbf{C}_k^{-}$ é $\mathbf{C}_k$ sem a contribuição das espécies i.

Agora suponha que a concentração $\mathbf{d}_{ik}$ das espécies i na corrente k seja não-redundante. Então, por definição, a concentração $\mathbf{d}_{ik}$ seria não-observável se ela não fosse medida. Da equação (5.2.29), o fluxo das espécies i na corrente k é não-observável, ou sua concentração é medida ou não. Isto implica , do Lema 4, que a mudança factível em suas taxas de fluxo, $\mathbf{v}_{ik}$, obedece

$$\mathbf{B}_{2,ik}\mathbf{d}_{ik}\delta\,\mathbf{n}_k + \mathbf{C}_k^{-}\delta\,\mathbf{n}_k + \mathbf{C}''\delta\,\mathbf{n}' + \mathbf{P}\delta\,\mathbf{v}' = \mathbf{0} \tag{5.2.30}$$

onde os primeiros termos indicam algumas mudanças factíveis em outros **n** e **v** com

$$\delta\,\mathbf{v}_{ik} \neq \mathbf{d}_{ik}\delta\,\mathbf{n}_k. \tag{5.2.31}$$

Mas isto implica que subtraindo-se a equação (5.2.29) de (5.2.30)

$$\mathbf{B}_{2,ik} = \mathbf{C}''\mathbf{w} + \mathbf{P}\mathbf{t} \tag{5.2.32}$$

para algum **w** e **t**, e deste modo

$$\left(\mathbf{Z}^{+}\right)^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{B}_{2,ik} = \mathbf{0}. \tag{5.2.33}$$

De outra maneira, suponha que a equação (5.2.33) se verifique. Isto implica, como mostrado na prova do Lema 5, que há vetores $\mathbf{w}_2$ e **t** e escalares $\mathbf{w}_1$ tais que

$$\mathbf{B}_{2,ik} + \mathbf{C}_k\mathbf{w}_1 + \mathbf{C}''\mathbf{w}_2 + \mathbf{P}\mathbf{t} = \mathbf{0}. \tag{5.2.34}$$

Pode-se novamente se separar as contribuições das espécies i em $\mathbf{C}_k$ para obter

$$\mathbf{B}_{2,ik}\left(\mathbf{1} + \mathbf{w}_1\mathbf{d}_{ik}\right) + \mathbf{C}_k^{-}\mathbf{w}_1 + \mathbf{C}''\mathbf{w}_1 + \mathbf{P}\mathbf{t} = \mathbf{0}. \tag{5.2.35}$$

Faça $\mathbf{d}_{ik}$ representar um valor calculado, quando não for medido. O coeficiente de $\mathbf{B}_{2,ik}$ na equação (5.2.35) é uma mudança factível nos fluxos das espécies no fluxo total. A razão destes coeficientes pode nunca ser igual a $\mathbf{d}_{ik}$ para um finito $\mathbf{w}_1$ tal que nenhum valor de $\mathbf{d}_{ik}$ possa ser encontrado. Assim, a concentração $\mathbf{d}_{ik}$ seria não-observável se fosse não medido, pelo Lema 4. Deste modo, $\mathbf{d}_{ik}$ é não-redundante.

(2) Assuma que n_k é observável. A concentração $\mathbf{d}_{ik}$ é não-redundante se e somente se ela for não-observável se não for medida. Pelo Lema 4, este é o caso se e somente se as taxas de fluxo das espécies $\mathbf{v}_{ik}$ forem não-observáveis. Isto é verdade se e somente se

$$\left(\mathbf{Z}^{+}\right)^{T}\mathbf{Y}^{T}\mathbf{B}_{2,ik}=0 \qquad (5.2.36)$$

por argumentos similares aos expostos no Lema 5.

Os casos especiais onde as matrizes **P** e **D** têm posto igual ao número de linhas leva respectivamente a **Y** = **0** ou **Z** = **0**. Nestes casos, quaisquer valores medidos são não-redundantes desde que as condições dos Lemas 5-8 para não-redundância são trivialmente satisfeitas.

STANLEY e MAH(1981a) definiram uma variável não-medida como sendo apenas observável se no mínimo uma medida não-redundante for necessária para calcular o seu valor. A deleção daquela medida não-redundante leva-a a ser não-observável e assim dependente de outras quantidades não-medidas. As quantidades apenas observáveis podem ser identificadas examinando-se aquelas colunas de **C** e de **P** sob as quais estas medidas seriam dependentes, quando deletadas, para encontrar novamente variáveis não observáveis, que é uma das quais era previamente observável.

O conjunto consistente de valores correspondentes às medidas na categoria 1 é explicitamente usado somente no Lema 7. Se a reconciliação é feita, as computações necessárias para o teste de redundância já terão sido feitas.

CROWE(1989) apresenta um algoritmo para se conseguir fazer a classificação das variáveis de um processo.

O algoritmo pode ser utilizado como parte do programa de reconciliação de dados, o que além de classificar as variáveis a serem reconciliadas, pode aproveitar-se dos cálculos intermediários necessários àquele procedimento.

5.3 DETECÇÃO DE ERROS GROSSEIROS EM SISTEMAS COM RESTRIÇÕES LINEARES

Seguindo o mesmo método descrito nos capítulos 3 e 4, o da projeção matricial, serão apresentadas as equações e a base teórica para a detecção de erros grosseiros. Outros métodos empregando diferentes análises se encontram descritos na análise da literatura (capítulo 2).

Será considerado que cada concentração e cada taxa de fluxo tem medidas independentes e normalmente distribuídas com média desconhecida. Apesar de o produto de duas variáveis não ser normalmente distribuído, pode-se assim considerar de maneira aproximada desde que ambas as variáveis assumam somente valores positivos e seus desvios padrões relativos atinjam valores suficientemente pequenos, algo como 5%. Esta consideração não é na prática excessivamente restritiva uma vez que para um intervalo de 95% de confiança, uma variável poderá ser escrita com diferença de 10% em seu valor. Será assumido que o vetor de medidas $\tilde{x}$ é uma amostra de distribuição multivariada normal com média desconhecida x mas com matriz variância-covariância conhecida Σ, a qual é uma bloco-diagonal, MADRON et al.(1977).

Definindo assim, os resíduos das equações de balanços, o vetor desbalanço **e**, fica

$$\mathbf{e} \equiv \mathbf{B}_0\mathbf{c} + \mathbf{B}_1\tilde{x} + \mathbf{P}\mathbf{v}_* \tag{5.3.1}$$

onde $\mathbf{v}_*$ é o vetor dos valores estimados de $\mathbf{v}$, valores cooptados.

Das equações (3.11) e (3.24), tem-se

$$\mathbf{Y}^T\mathbf{e} = \mathbf{Y}^T\left[\mathbf{B}_0\mathbf{c} + \mathbf{B}_1\tilde{x}\right] = -\mathbf{H}\lambda_* \tag{5.3.2}$$

a fim de estabelecer distribuição estatística de $\mathbf{Y}^T\mathbf{e}$, nota-se que o valor real de $\mathbf{v}$ deve satisfazer à equação

$$\mathbf{B}_0\mathbf{c} + \mathbf{B}_1 x + \mathbf{P}\mathbf{v} = \mathbf{0} \tag{5.3.3}$$

então,

$$\mathbf{E}\left(\mathbf{Y}^T\mathbf{e}\right) = \mathbf{0} \tag{5.3.4}$$

a partir das equações (3.11) e (5.3.3).

A variância de $\mathbf{Y}^T\mathbf{e}$ é dada por

$$\mathbf{var}\left(\mathbf{Y}^T\mathbf{e}\right) = \mathbf{E}\left(\mathbf{Y}^T\mathbf{e}\mathbf{e}^T\mathbf{Y}\right) = \mathbf{Y}^T\mathbf{B}_1\,\Sigma\,\mathbf{B}_1^T\mathbf{Y} = \mathbf{H} \tag{5.3.5}$$

CROWE(1983), mostra que conforme definido por REILLY e CARPANI(1963), e mais tarde por ALMASY e SZTANO(1975), a quantidade

$$\chi^2_Y = \mathbf{e}^T\mathbf{Y}\mathbf{H}^{-1}\mathbf{Y}^T\mathbf{e} \tag{5.3.6}$$

tem uma distribuição qui-quadrada (Apêndice A) com o número de graus de liberdade igual ao posto da matriz $\mathbf{Y}$. Assim, os desbalanços, ou seja, a medida do erro das medidas podem ser simultaneamente testados comparando-os com os valores tabelados de χ^2.

Em conseqüência das equações (3.17) (3.19) e (5.3.2),

$$\chi^2_Y = \mathbf{a}_*^T\,\Sigma^{-1}\,\mathbf{a}_* = \mathbf{F}(\mathbf{a}_*) \tag{5.3.7}$$

Este fato fornece a base estatística para rejeitar a medida quando sua remoção implica significante redução na função objetivo.

Além do mais, qualquer conjunto de combinações lineares de $\mathbf{Y}^T\mathbf{e}$ pode ser similarmente testado. Por exemplo, todos os balanços restantes para um nó particular, poderiam ser selecionados multiplicando-se $\mathbf{Y}^T\mathbf{e}$ por uma matriz cujas linhas são vetores unitários. Em geral, se $\mathbf{W}$ é uma matriz apropriada, com posto igual ao número de colunas, então correspondendo a $\mathbf{W}^T\mathbf{Y}^T\mathbf{e}$, a quantidade

$$\chi^2_{YW} = \mathbf{e}^T\mathbf{Y}\mathbf{W}\left(\mathbf{W}^T\mathbf{H}\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{Y}^T\mathbf{e} \tag{5.3.8}$$

também tem uma distribuição qui-quadrado, mas com graus de liberdade igual ao posto de **W** (menor que o posto de **Y**). Notar que a transformação $\mathbf{Y} \rightarrow \mathbf{YQ}$, onde **Q** é uma matriz quadrada não-singular apropriada, não tem efeito no valor de χ^2 na equação (5.3.6), nem nos ajustes de $\mathbf{a}_*$, na equação (3.17).

Se em particular, ao invés da matriz **W**, usar-se o vetor **w**

$$\mathbf{Z} \equiv \frac{\mathbf{w}^T\mathbf{Y}^T\mathbf{e}}{\left(\mathbf{w}^T\mathbf{H}\mathbf{w}\right)^{0,5}} \sim \mathbf{N}(0,1) \tag{5.3.9}$$

tal que qualquer combinação linear dos elementos de $\mathbf{Y}^T\mathbf{e}$ possa ser testada com distribuição normal e variância unitária.

Há duas escolhas específicas de **w** as quais possibilitam testes diretos para qualquer desbalanço ou ajuste respectivamente.
Deste modo, se

$$\mathbf{w} = \delta_k \tag{5.3.10}$$

onde o vetor δ_k é unitário

$$\mathbf{Z}_k^e \equiv \frac{\left(\mathbf{Y}^T\mathbf{e}\right)_k}{\mathbf{H}_{kk}^{0,5}} \tag{5.3.11}$$

Segundo CROWE(1983), REILLY e CARPANI(1963) foram os primeiros a sugerirem este teste.

Assim, $\mathbf{Z}_k^e$ pode ser calculado para todo elemento de $\mathbf{Y}^T\mathbf{e}$ e testado comparando com um **Z** que siga a distribuição normal. Se a causa de um erro grosseiro for um vazamento ou acúmulo provocado pela saída do estado estacionário (os quais podem apresentar erros muito grosseiros) em uma unidade da planta, deve-se esperar agrupamento de medidas (envelopes) os quais contenham valores de $\mathbf{Z}_k^e$ muito maiores que o valor crítico. Agrupamentos sem tais distorções não devem apresentar valores acima do valor crítico.

A segunda escolha de **w** é

$$\mathbf{w}^T = -\delta_j^T \Sigma \mathbf{B}_1^T \mathbf{Y}\mathbf{H}^{-1} \tag{5.3.12}$$

então,

$$\mathbf{w}^T\mathbf{Y}^T\mathbf{e} = -\delta_j \Sigma \mathbf{B}_1^T \mathbf{Y}\lambda, \tag{5.3.13}$$

a partir das equações (3.17) e (5.3.2),

$$\mathbf{w}^T\mathbf{H}\mathbf{w} = \left(\Sigma \mathbf{B}_1^T \mathbf{Y}\mathbf{H}^{-1}\mathbf{Y}^T\mathbf{B}_1 \Sigma\right)_{jj} \equiv \mathbf{Q}_{jj} \tag{5.3.14}$$

e

$$\mathbf{Z}_j^a \equiv \left(\frac{\mathbf{a}_j}{\mathbf{Q}_{jj}^{0,5}} \right) \sim \mathbf{N}(0,1) \qquad (5.3.15)$$

o que permite um teste direto de cada ajuste, novamente com distribuição normal.

Segundo MAH et al.(1982), deve-se concluir pela existência do erro se

$$|\mathbf{Z}_k| > |\mathbf{Z}_{\alpha/2}| \qquad (5.3.16)$$

onde $\mathbf{Z}_{\alpha/2}$ é o ponto superior da quantidade $\alpha/2$ da distribuição normal padrão e α é o nível de significância que se quer dar à medida (aqui 5%).

Deve-se concluir que a k-ésima medida contém um erro grosseiro com erro tipo I (Apêndice A) com probabilidade α .

O teste acima pode ser modificado para se levar em conta o efeito de se conduzir múltiplos testes (i=1, 2.....n). A idéia é que para múltiplos testes, todos ao nível α , a probabilidade de rejeitar no mínimo um $\mathbf{H_{0k}}$ quando na verdade todos os $\mathbf{H_{0k}}$ são verdadeiros, pode ser muito maior que α . Principalmente quando n tende ao "infinito". Para levar em conta este efeito e garantir que a probabilidade de erro global do tipo I seja controlada porα , o teste da equação (5.3.16) pode ser modificado da seguinte maneira:
rejeite $\mathbf{H_{0k}}$ e conclua que há erro grosseiro presente na observação se

$$|\mathbf{Z}_k| > \mathbf{Z}_{\beta/2} \qquad (5.3.17)$$

onde

$$\beta = 1 - (1-\alpha)^{1/n} \qquad (5.3.18)$$

desde que $\beta < \alpha$ para n > 1, $\mathbf{Z}_{\beta/2} > \mathbf{Z}_{\alpha/2}$ fazendo ser mais difícil rejeitar $\mathbf{H_{0k}}$. Aqui, $\mathbf{Z}_{\beta/2}$ é chamado crítico.

De fato, numa situação onde um simples erro grosseiro está presente, mas não determinado, é apropriado utilizar como $\mathbf{C_{crítico}}$, o valor $\mathbf{Z}_{\beta/2}$ e não $\mathbf{Z}_{\alpha/2}$. Isto porque em tal situação, o teste deve ser baseado em $|\mathbf{Z}|_{max} = \max_{1 \le k \le n} |\mathbf{Z}_k|$ e o erro grosseiro em observação correspondente a $|\mathbf{Z}|_{max}$ é indicado, se este excede o valor crítico.

MAH et al.(1982) também discutiram a potência deste teste mostrando que quanto maior o número de erros grosseiros, mais facilmente e confiavelmente serão detectados.

5.4 DETECÇÃO DE ERROS GROSSEIROS EM SISTEMAS COM RESTRIÇÕES NÃO-LINEARES

De acordo com o desenvolvido para o caso com restrições lineares, tem-se que todas as análises são válidas ainda para este caso, mudando basicamente a forma de

cálculo dos parâmetros de comparação. Uma vez que a mesma análise de resultados pode ser empregada aqui neste segundo caso, serão apresentadas basicamente as equações de cálculo.

No caso não-linear, a análise estatística é complicada pela dependência de Z do vetor **d**. Por este motivo, um desenvolvimento em forma de série de Taylor é empregada para descrever a distribuição normal de

$$\mathbf{e} \equiv \mathbf{Z}^T\mathbf{Y}^T\mathbf{B}_1\tilde{x} \sim \mathbf{N}(0,T) \tag{53.19}$$

$$\mathbf{a} \sim \mathbf{N}(0,Q1) \tag{5.3.20}$$

e

$$\delta \sim \mathbf{N}(0,Q2) \tag{5.3.21}$$

com **Q1** e **Q2** usualmente singular.

Deseja-se deduzir a média e a variância do resíduo (com k = 0, para simplicidade). Considera-se que os ajustes $\hat{x}$ e $\hat{\mathbf{d}}$ são independentes e normalmente distribuídos.
Assim,

$$\delta_x \sim N(0,\Sigma_1) \tag{5.3.22}$$

$$\delta_d \sim N(0, \Sigma_d) \tag{5.3.23}$$

$$\text{com } \delta_x = x - \hat{x} \tag{5.3.24}$$

$$\delta_d = \mathbf{d} - \hat{\mathbf{d}} \tag{5.3.25}$$

Define-se

$$\mathbf{T} \equiv \mathbf{Z}^T\mathbf{H}\mathbf{Z} \tag{5.3.26}$$

Nota-se das definições das equações (4.12), para **D** e (4.13), para **Z**, que

$$\frac{\partial \mathbf{Z}^T}{\partial \mathbf{d}_i}.\mathbf{D} = -\mathbf{Z}^T\mathbf{Y}^T\left[\mathbf{0}\cdots\mathbf{0}\mathbf{B}_{2,i},\mathbf{0}\cdots\mathbf{0}\right] \tag{5.3.27}$$

onde $\mathbf{B}_{2,i}$ é a coluna correspondente a $\mathbf{d}_i$ e i é correspondente a qualquer concentração na categoria 2. Deste modo,

$$\hat{\mathbf{Z}}^T \equiv \mathbf{Z}\left(\hat{\mathbf{d}}\right) \tag{5.3.28}$$

e por definição

$$\hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{D}(\hat{\mathbf{d}}) = \mathbf{0} \tag{5.3.29}$$

Das equações (4.9) e (4.14)

$$\mathbf{Z}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{B}_1\tilde{x} = \mathbf{0} \tag{5.3.30}$$

Expandindo o resíduo em série de Taylor de primeira ordem, e levando em conta equação (5.3.29),

$$\mathbf{e} = \hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{B}_1\delta x + \sum_i \frac{\partial \mathbf{Z}^{\mathrm{T}}}{\partial \mathbf{d}_i}\mathbf{Y}^{\mathrm{T}}\mathbf{B}_1\tilde{x}\delta\,\mathbf{d}_i \tag{5.3.31}$$

ágora, com as equações (5.3.27) , (4.90) e (4.14)

$$\mathbf{e} = \hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{B}_1\delta x + \hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{B}_2\mathbf{N}\delta\,\mathbf{d} \tag{5.3.32}$$

assim

$$\mathbf{E}(\mathbf{e}) = \mathbf{0} \tag{5.3.33}$$

e

$$\mathbf{E}(\mathbf{e}\mathbf{e}^{\mathrm{T}}) = \mathbf{Z}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\left(\mathbf{B}_1\Sigma_1\mathbf{B}_1^{\mathrm{T}} + \mathbf{B}_2\mathbf{N}\Sigma_d\mathbf{N}\mathbf{B}_2^{\mathrm{T}}\right)\mathbf{Y}\hat{\mathbf{Z}} \tag{5.3.34}$$

No limite, $\Sigma_2 \rightarrow \mathbf{N}\Sigma_d\mathbf{N}$ se a solução é convergente, tal que da equação (5.3.26) e (4.31) obtém-se

$$\mathbf{E}(\mathbf{e}\mathbf{e}^{\mathrm{T}}) = \hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{H}\hat{\mathbf{Z}} = \hat{\mathbf{T}} \tag{5.3.35}$$

deste modo, o resíduo é aproximadamente distribuído

$$\mathbf{e} \sim (\mathbf{0}, \hat{\mathbf{T}}) \tag{5.3.36}$$

podendo assim as distribuições de $\mathbf{a}$ e δ também serem aproximadas das equações (4.28), (4.29), e (4.32) tal que

$$\mathbf{a} = -\Sigma_1\mathbf{B}_1^{\mathrm{T}}\mathbf{Y}\mathbf{Z}\mathbf{T}^{-1}\mathbf{e} \tag{5.3.37}$$

$$\delta = -\mathbf{N}^{-1}\Sigma_2\mathbf{B}_2^{\mathrm{T}}\mathbf{Y}\mathbf{Z}\mathbf{T}^{-1}\mathbf{e} \tag{5.3.38}$$

uma expansão em série de Taylor com $\hat{\mathbf{e}} = \mathbf{0}$,

$$\mathbf{a} = -\mathbf{N}^{-1}\Sigma_2\mathbf{B}_2^{\mathrm{T}}\mathbf{Y}\hat{\mathbf{Z}}\hat{\mathbf{T}}^{-1}\mathbf{e} \tag{5.3.39}$$

$$\delta = -\mathbf{N}^{1}\Sigma_2\mathbf{B}_2^{\mathrm{T}}\mathbf{Y}\hat{\mathbf{Z}}\hat{\mathbf{T}}^{-1}\mathbf{e} \tag{5.3.40}$$

Os dados podem ser testados, como conjunto global, da seguinte maneira

$$\chi_e^2 \equiv \mathbf{e}^T\mathbf{T}^{-1}\mathbf{e} \tag{5.3.41}$$

onde χ_e^2 é a estratégia de qui-quadrado com o número de graus de liberdade igual ao posto da matriz **Z**.

Como na parte linear, as combinações de **e** podem ser testadas novamente, pois o teste da equação (5.3.41) não é conclusivo. Com distribuição normal desde que se tem

$$\mathbf{Z} \equiv \frac{\mathbf{w}^T\mathbf{e}}{(\mathbf{w}^T\mathbf{T}\mathbf{w})^{0.5}} \sim \mathbf{N(0,1)} \tag{5.3.42}$$

Em particular, $\mathbf{w} = \mathbf{u}_k$, o k-ésimo vetor unitário é

$$\mathbf{Z}_k^e = \frac{\mathbf{e}_k}{\mathbf{T}_{kk}^{0.5}} \tag{5.3.43}$$

Do mesmo modo, podem ser testados os elementos individuais de **a** e δ, tal que

$$\mathbf{Z}_j^a = \frac{\mathbf{a}_j}{\left(\mathbf{Q}_{1,jj}\right)^{0.5}} \sim \mathbf{N(0,1)} \tag{5.3.44}$$

com

$$\mathbf{Q}_{1,jj} \equiv \mathbf{B}_1^T\mathbf{Y}\hat{\mathbf{Z}}\hat{\mathbf{T}}^{-1}\hat{\mathbf{Z}}\mathbf{Y}^T\mathbf{B}_1\,\Sigma_1 \tag{5.3.45}$$

e

$$\mathbf{Z}_1^\delta \equiv \frac{\delta_1}{\left(\mathbf{Q}_{2,ll}\right)^{0.5}} \sim \mathbf{N(0,1)} \tag{5.3.46}$$

com

$$\mathbf{Q}_{2,ll=}\,\Sigma_d\,\mathbf{N}\mathbf{B}_2^T\mathbf{Y}\hat{\mathbf{Z}}\hat{\mathbf{T}}^{-1}\hat{\mathbf{Z}}\mathbf{Y}^T\mathbf{B}_2\mathbf{N}\,\Sigma_d \tag{5.3.47}$$

Comparando-se os valores de **e**, **a** e δ cujos valores absolutos são muito grandes para serem consistentes com as equações (5.3.42), (5.3.44) e (5.3.46), tem-se informações com as quais é possível diagnosticar quais medidas são mais propensas a estarem com erros grosseiros.

Uma desvantagem de se usar $\mathbf{Z}_k^e$ como teste estatístico para os resíduos nas restrições é que $\mathbf{Z}^T$ não está simplesmente relacionado aos balanços nos nós para as espécies individualmente. Uma alternativa para se contornar tais problemas é definir (com k = 0)

$$\varepsilon \equiv \mathbf{Y}^{\mathrm{T}}\left[\mathbf{B}_1\tilde{x} + \mathbf{B}_2\mathbf{N}\tilde{\mathbf{d}}\right] \tag{5.3.48}$$

Usando as equações (4.9), (4.28) e (4.29), pode-se obter

$$\varepsilon = -\mathbf{HZ}\lambda \tag{5.3.49}$$

e com as equações (4.32) e (4.42)

$$\varepsilon \equiv \mathbf{HZT}^{-1}\mathbf{e} \tag{5.3.50}$$

assim, ε tem uma distribuição normal degenerada,

$$\varepsilon \sim \mathrm{N}(0,\ \mathbf{HZT}^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{H}) \tag{5.3.51}$$

cujos elementos podem ser testados por

$$\mathbf{Z}_k^e \equiv \frac{\varepsilon_k}{\left(\mathbf{HZT}^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{H}\right)_{kk}^{0.5}} \tag{5.3.52}$$

5.5 PROCEDIMENTO DE UTILIZAÇÃO DO PROGRAMA RETIF

O programa RETIF foi desenvolvido com as mesmas características do programa RECON, portanto, não é necessário se repetir todo o procedimento. O programa RECON tem dois arquivos de saída sendo que um dá acesso aos dados calculados para a reconciliação e o outro serve como arquivo de entrada para a execução do RETIF. Os dados intermediários, necessários aos cálculos para a detecção dos erros grosseiros são, deste modo, transferidos automaticamente sem que seja necessário se digitar nada mais que o nome do arquivo de entrada e o nome do arquivo de saída, quando se está executando o programa RETIF. Os valores padrões para a estatística de comparação já estão embutidos no programa.

6 EXEMPLOS NUMÉRICOS

Os testes para os programas desenvolvidos envolvem a utilização de três sistemas diferentes conforme ilustrados na literatura. O fato de se utilizar de um método específico (projeção matricial), impede que se utilize os mesmos exemplos desenvolvidos utilizando-se outros métodos para fins de comparação de resultados numéricos. Isto deve ser considerado, entretanto, apenas como característica da reconciliação de dados uma vez que em todos os trabalhos publicados , e citados nesta dissertação, assume-se que determinada metodologia é conveniente para determinados sistemas e não para outros. Não é apresentado, por nenhum dos autores pesquisados, um procedimento que desenvolva ótimos resultados em todos os sistemas. Particularidades precisam ser observadas e isto faz toda a diferença quando se pensa em tratar os dados de um processo qualquer.

Seguindo nesta linha de raciocínio, foi feito um estudo detalhado para várias situações diferentes a fim de se confirmar a aplicabilidade do método aqui apresentado e mostrar que os programas aqui desenvolvidos estão de acordo com o previsto pela teoria.

Para facilitar a compreensão dos casos estudados, serão apresentados em tópicos separados os resultados da reconciliação de dados, cujos resultados foram conseguidos através da execução do programa RECON. Os resultados referentes à detecção de erros grosseiros foram conseguidos através da execução do programa RETIF. A forma de execução dos dois está detalhada nos respectivos capítulos que tratam de cada tema.

6.1 RESULTADOS DA RECONCILIAÇÃO

O primeiro exemplo estudado é o mesmo detalhado em CROWE(1986) o qual utiliza dados apresentados por SMITH e ICHIYEN(1973). Este exemplo é razoavelmente simples, envolvendo matrizes de tamanho reduzido (menores que 30x30) o que possibilita resolvê-los sem os problemas que se pode encontrar em sistemas maiores devido ao elevado número de matrizes que é preciso manipular durante a execução dos dois programas desenvolvidos (cerca de 300 matrizes, quadradas e não quadradas).

Trata-se de um circuito de flotação, fig 6.1, o qual segundo CROWE é um importante exemplo para ilustrar a teoria. As concentrações de cobre e zinco foram medidas em todas as correntes exceto na corrente número 8, entretanto nenhuma taxa de fluxo total foi dada, de modo que somente fluxos relativos podem ser estimados. Foi portanto definida uma taxa de fluxo na alimentação, em base 1. Os fluxos relativos nas demais correntes são conseguidos através de balanços de massa. A corrente 8 poderá ser determinada através dos procedimentos do programa.

Na descrição do exemplo, o autor não apresenta o detalhamento requerido para os valores das variâncias das correntes totais, apenas apresenta o desvio padrão relativo médio. Por causa disto, o problema pôde ser estudado sob mais este aspecto, a saber, a influência da mudança da variância nos resultados de reconciliação.

É preciso lembrar da secção 5.3, que não existem algoritmos capazes de apontar especificamente a presença de mais que um erro grosseiro de cada vez num conjunto de medidas dadas. Isto não quer dizer no entanto que eles não possam ser detectados se estivem nesta condição, mas neste caso exige-se uma investigação dos dados em blocos, fazendo-se testes com a retirada de duas ou mais medidas simultaneamente e

investigando-se os efeitos daí decorrentes. Segundo MAH et al. (1987), a interação de dois ou mais erros grosseiros pode obscurecer a busca das verdadeiras medidas com erros grosseiros. Caso mais que um erro grosseiro esteja presente ao mesmo tempo em um mesmo conjunto de medidas dado, haverá reflexos nos demais valores reconciliados, inviabilizando o procedimento (capítulo 1).

Serão apresentadas tabelas que resumem os resultados para cada caso estudado. Após cada tabela será feito um breve comentário a respeito dos resultados apresentados. Alguns termos utilizados neste capítulo não estão descritos nos anteriores e devem, portanto, ser apresentados para que se possa acompanhar os resultados aqui mostrados.

* Desvio Relativo: (DR)= [1-(valor reconciliado/valor esperado)]*100 , em porcentagem.

*Variância Proporcional: segundo o valor apresentado por CROWE(1986), as variâncias têm desvio relativo médio igual a 6,56%. Deste modo, multiplicou-se este valor pelo da corrente total, determinando-se assim valores de variância proporcionais ao fluxo de cada corrente.

*Variâncias iguais: variâncias iguais em todas as correntes totais iguais a 0,0656 que é o valor médio apresentado pelo mesmo autor.

*Taxas de fluxo esperadas(TFE): são valores de taxas de fluxos esperados caso a correção fosse nula, ou seja, não houvesse desvio nas medidas. Podem aparecer como Fluxo pelo Balanço de Massa.

*Variáveis tipo 1: são aquelas variáveis que identificam correntes que têm as frações molares e as taxa de fluxo totais conhecidas.

*Variáveis tipo 2: são aquelas que identificam correntes que têm apenas as frações molares conhecidas. As taxas de fluxo totais precisam ser determinadas.

*Variáveis tipo 3: são aquelas que identificam correntes que não têm nem fluxos totais nem concentrações medidas.

*Legendas das tabelas:

CC-correntes-componentes,
VM-vazões medidas,
VC-vazões corrigidas,
aj-ajuste das vazões medidas,
Delta-ajuste das concentrações medidas,
Er(%)-desvio relativo para as vazões e concentrações,
Ei(%)-erro introduzido propositadamente na variável,
CME-concentrações medidas,
CCOR-concentrações corrigidas,

*Componentes: os componentes nos processos são identificados sempre por uma letra maiúscula e um número. A letra se refere a um elemento ou composto químico particular e o número se refere à corrente em questão. Quando se tratar de fluxos totais, a corrente será identificada por T mais um número, com o mesmo significado anterior.

No estudo de casos de dados de processo reconciliados pode-se fazer infinitas simulações para se estudar infinitas combinações de arranjos de dados. Neste trabalho será limitado a 20 (vinte) o número de combinações consideradas, para o caso do circuito de flotação.(Não existe um motivo especial para este número de simulações, significando apenas que no decorrer das simulações foram estudados diferentes casos e quando se concluiu que o número já era representativo existiam 20 casos estudados). Para os demais exemplos, serão apenas apresentados os valores da reconciliação de dados levando-se em conta que não contenham erros grosseiros. O circuito de flotação foi o escolhido para detalhamento por motivos anteriormente explicados e também por ser o exemplo utilizado por CROWE(1986) para a descrição da detecção de erros grosseiros. Os valores medidos apresentados nas tabelas 6.1a e 6.1b são os valores apresentados na literatura citada. Não existem referências que sejam valores reais, mas serão aqui tomados como parâmetros de comparação para os valores conseguidos com a simulação do processo com o programa desenvolvido. De aqui em diante, qualquer referência a valores reais deverão ser entendidas como referentes aos valores apresentados nessas duas tabelas.

As medidas portadoras de erros aleatórios estão marcadas com um asterisco na frente do valor , as medidas com erros grosseiros estão marcadas com dois asteriscos.

Os desvios aleatórios e grosseiros foram introduzidos nas mesmas variáveis com a intenção de se ter uma base constante para comparação. Segundo ROSENBERG(1987), a posição dos erros grosseiros dentre as variáveis medidas é muito importante quando se utiliza estratégias estatísticas na sua determinação (capítulo 2). Para efeitos de comparação da influência da reconciliação de dados nas variáveis do processo, serão apresentados valores dos balanços de massa, com e sem a utilização da reconciliação. O caso do balanço sem reconciliação não leva em consideração as variâncias, apenas balanços mássicos simples. É na verdade uma simplificação da equação geral. Neste exemplo, os valores de Ei são dados em relação aos valores das tabelas 6.1

EXEMPLO 1: CIRCUITO DE FLOTAÇÃO

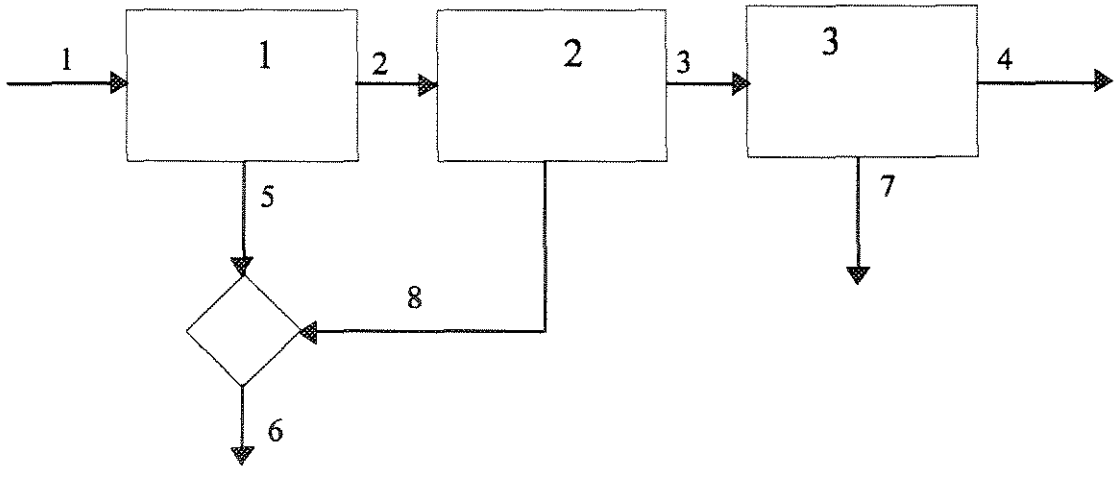


Figura 6.1: Fluxograma simplificado do circuito de flotação.

Componentes: Cobre(A) e Zinco(B)
Correntes com variáveis tipo 1: 1
Correntes com variáveis tipo 2: 2,3,4,5,6,7
Correntes com variáveis tipo 3: 8.

CASO 1: Variâncias iguais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019280	0,019280	0,000000	0,00	0,00
B-1	0,049500	0,049500	0,000000	0,00	0,00
T-1	1,000000	1,000000	0,000000	0,00	0,00

Tabela 6.1a:Resultados da reconciliação de dados do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004500	0,000000	0,00	0,00
A-3	0,001300	0,001280	0,000000	0,00	0,00
A-4	0,000900	0,000906	0,000006	0,00	0,00
A-5	0,198800	0,198800	0,000000	0,00	0,00
A-6	0,214300	0,214301	0,000001	0,00	0,00
A-7	0,005100	0,005144	0,000044	0,00	0,00
B-2	0,047800	0,047800	0,000000	0,00	0,00
B-3	0,049500	0,049500	0,000000	0,00	0,00
B-4	0,004100	0,004100	0,000000	0,00	21,95
B-5	0,070700	0,070700	0.000000	0.00	0,00
B-6	0,049200	0,049200	0,000000	0.00	0,00
B-7	0,518800	0,518801	0,000001	0,00	0,00
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	0,999999	0,000001	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	0,999994	0,000006	0,00	0,00
T-7	1,000000	0,999998	0,000002	0,00	0,00

Tabela 6.1b: Resultados da reconciliação de dados de variáveis tipo 1

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,008440	0,080800
Fluxo pelo Bal. Massa Desvio Real (%)	0,923996 (0,00)	0,915564 (0,06)	0,834804 (-0,02)	0,076003 (-0,13)	0,084438 (0,07)	0,080759 (-0,05)
Fluxo Determinado Desvio Ajustado(%)	0,923989 (-0,01)	0,915550 (0,06)	0,834803 (-0,16)	0,076003 (-0,13)	0,084439 (0,07)	0,080760 (-0,05)

Tabela 6.1c: Fluxos determinados das correntes com variáveis do tipo 2

CC	VM	VC
A-8	—	0,353800#
B-8	—	-0,154000#
T-8	—	0,00844

Tabela 6.1d: Fluxos determinados das correntes com variáveis do tipo 3 (cooptação). # valores requeridos para satisfazer os balanços mássicos.

Neste caso, em que se usou variâncias iguais em todas as correntes, não existiam desvios de espécie alguma, nem grosseiros e nem aleatórios. Esses dados reconciliados correspondem, como já descrito anteriormente, aos valores apresentados na literatura, as variações "infinitesimais" correspondem às sucessivas aproximações por que passam os cálculos até chegarem ao resultado final. Este exemplo serve apenas para ilustrar o fato de que em condições ideais de funcionamento, todo o procedimento de reconciliação não afeta os valores das variáveis de maneira significativa para o processo. Os resultados para a cooptação (variáveis tipo 3) apresentadas não têm significado físico pois existem entre os valores, quantidades negativas. No exemplo desenvolvido na literatura, o mesmo comportamento se verifica mostrando que o erro não se identifica entre os cálculos do procedimento aqui desenvolvido. Sobre o problema dos valores cooptados, ler a análise transcrita de CROWE no final deste EXEMPLO.

CASO 2: Variâncias proporcionais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019300	0,021256	0,001956	-10,14	0,00
B-1	0,038100**	0,046330	0,008230	-21,60	-23,03
T-1	1,000000	1,005402	0,005402	-0,54	0,00

Tabela 6.2a: Resultados da reconciliação dos dados das variáveis tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004502	0,000002	-0,04	0,00
A-3	0,001300	-0,000509	-0,001809	139,15	0,00
A-4	0,000900	0,038779	0,037879	-4208,79	0,00
A-5	0,198600*	0,198624	0,000024	-0,01	-0,10
A-6	0,214400	0,209699	-0,004701	2,19	0,00
A-7	0,005100	0,074410	0,069310	-1359,03	0,00
B-2	0,047200*	0,047200	0,000000	0,00	-1,25
B-3	0,053600**	0,053600	0,000000	0,00	8,28
B-4	0,004100	0,004168	0,000068	-1,67	0,00
B-5	0,070900*	0,070923	0,000023	-0,03	0,28
B-6	0,049500*	0,049476	-0,000024	0,05	0,61

B-7	0.521000*	0,521220	0,000220	-0,04	0,42
T-2	1,000000	0,999999	-0,000001	0,00	0,00
T-3	1,000000	0,999997	-0,000003	0,00	0,00
T-4	1,000000	1,000068	0,000068	-0,01	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	1,000031	0,000031	0,00	0,00
T-7	1,000000	1,000012	0,000012	0,00	0,00

Tabela 6.2b: Resultados da reconciliação de dados das variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa Desvio real (%)	0,926844 (0,30)	0,917686 (0,29)	0,829806 (0,62)	0,072622 (-4,50)	0,081547 (-3,3)	0,087881 (8,70)
Fluxo Determinado Desvio Ajustado(%)	0,884902 (-4,23)	0,852433 (-6,84)	0,838050 (0,36)	0,086962 (14,27)	0,093880 (11,23)	0,073407 (-9,15)

Tabela 6.2c: Fluxos determinados nas correntes com variáveis do tipo2.

A presença de dois erros grosseiros entre as medidas compromete o procedimento de reconciliação. Dentre as medidas do primeiro tipo existe um erro claramente grosseiro na segunda medida. No grupo de medidas do segundo tipo existe um erro grosseiro na oitava medida . Os demais erros são aleatórios.

Pode-se notar que o procedimento de reconciliação de dados reorganiza os balanços mássicos das correntes permitindo que se "perceba" a presença dos erros grosseiros. Isto pode ser comprovado comparando-se, por exemplo, os valores dos fluxos determinados com e sem o ajuste dos valores.

CASO 3: Variâncias proporcionais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019300	0,019303	0,000003	-0,02	0,00
B-1	0,049500	0,049515	0,000015	-0,03	0,00
T-1	1,000000	1,000009	0,000009	0,00	0,00

Tabela 6.3a: Resultados da reconciliação de dados de variáveis tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004500	0,000000	0,00	0,00
A-3	0,001300	0,001307	0,000007	0,00	0,00
A-4	0,000900	0,000750	-0,000150	0,00	0,00
A-5	0,198600*	0,198600	0,000000	0,00	-0,10

A-6	0,214400	0,214386	-0,000014	0,00	0,00
A-7	0,005100	0,005307	0,000207	-4,06	0,00
B-2	0,047800	0,047800	0,000000	0,00	0,00
B-3	0,053600**	0,053600	0,000000	0,00	8,28
B-4	0,004100	0,004100	0,000000	-0,01	0,00
B-5	0,070900*	0,070900	0,000000	0,00	0,28
B-6	0,049500*	0,049500	0,000000	0,00	0,61
B-7	0,521000*	0,520999	0,000001	0,00	0,42
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	1,000000	0,000000	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	1,000000	0,000000	0,00	0,00
T-7	1,000000	1,000000	0,000000	0,00	0,00

Tabela 6.3b: Resultados da reconciliação de dados das variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa	0,921221	0,917686	0,825302	0,078751	0,087072	0,087403
Desvio Real(%)	(-0,30)	(0,29)	(-1,16)	(3,48)	(3,17)	(8,17)
Fluxo Determinado	0,921381	0,912539	0,825367	0,078738	0,087061	0,087360
Desvio Ajustado(%)	(-0,28)	(-0,27)	(-1,15)	(3,47)	(3,15)	(8,12)

Tabela 6.3c: Fluxos determinados das correntes com variáveis do tipo 2.

Comparando-se os valores das tabelas 6.3 com os apresentados nas tabelas 6.2, nota-se que ao se retirar o erro grosseiro de uma variável medida do tipo 1, tem-se a aproximação dos resultados da reconciliação em direção aos valores reais. A presença de erro grosseiro (de pequena intensidade) numa variável do tipo 2 ainda compromete o resultado da reconciliação dos dados. Pode-se comprovar que neste último caso a reconciliação alterou menos os balanços que no caso apresentado nas tabelas 6.2, por causa da presença de menor quantidade de erros.

CASO 4: Variâncias proporcionais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019300*	0,019299	-0,000001	0,01	0,10
B-1	0,049500	0,049497	-0,000003	0,01	0,00
T-1	1,000000	0,999998	-0,000002	0,00	0,00

Tabela 6.4a: Resultados da reconciliação das variáveis tipo 1

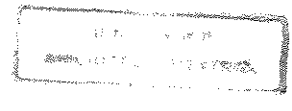
CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004500	0,000000	0,00	0,00
A-3	0,001300	0,001298	-0,000002	0,14	0,00
A-4	0,000900	0,000939	0,000039	4,31	0,00
A-5	0,198600*	0,198600	0,000000	0,00	-0,10
A-6	0,214400*	0,214405	0,000005	0,00	0,01
A-7	0,005100	0,005307	0,000207	-4,06	0,00
B-2	0,047800	0,047800	0,000000	0,00	0,00
B-3	0,049000*	0,049000	0.000000	0,00	1,01
B-4	0,004100	0,004100	0,000000	0,00	0,00
B-5	0,070900*	0,070900	0,000000	0,00	0,28
B-6	0,049500*	0,049500	0,000000	0,00	0,61
B-7	0,521000*	0,520999	-0,000001	0,00	0,42
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	1,000000	0,000000	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	1,000000	0,000000	0,00	0,00
T-7	1,000000	1,000000	0,000000	0,00	0,00

Tabela 6.4b: Resultados da reconciliação para as variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa Desvio Real(%)	0,924089 (0,01)	0,915860 (0,09)	0,836304 (0,14)	0,075909 (-0,25)	0,084163 (-0,28)	0,079556 (-1,55)
Fluxo Determinado Desvio Ajustado(%)	0,924050 (0,01)	0,915789 (0,09)	0,836298 (0,15)	0,075910 (-0,25)	0,084165 (-0,27)	0,079559 (-1,54)

Tabela 6.4c: Fluxos determinados das correntes com variáveis tipo 2

Este exemplo, juntamente com os três próximos, servirão para se estudar a influência da variância e, consequentemente do número de medidas das variáveis monitoradas na planta uma vez que a mesma depende também do número de amostras, em conjunto de valores que não apresentem erros grosseiros entre suas medidas.
Podem ser comparados com o exemplo apresentado nas tabelas 6.3. Aqui não foram mudadas as condições das variâncias em relação ao caso anterior. Os resultados mostram que quando não existem erros grosseiros entre as medidas, os valores reconciliados estão muito próximos dos valores esperados, e consequentemente os valores dos ajustes das variáveis tendem a zero tanto para as variáveis do tipo 1 quanto para as do tipo 2. Conforme os resultados apresentados na tabela 6.4c, os balanços dos ajustes são quase idênticos aos balanços reconciliados para o caso dos fluxos.



CASO 5: Variâncias proporcionais (valor base multiplicado por 10)

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019280	0,019280	0,000000	0,00	0,00
B-1	0,049500	0,049500	0,000000	0,00	0,00
T-1	1,000000	1,000000	0,000000	0,00	0,00

Tabela 6.5a: Resultados da reconciliação dos valores das variáveis tipo 1

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004500	0,000000	0,00	0,00
A-3	0,001300	0,001300	0,000000	0,00	0,00
A-5	0,000900	0,000906	0,000006	0,00	0,00
A-6	0,198800	0,198800	0,000000	0,00	0,00
A-7	0,214300	0,214300	0,000001	0,00	0,00
B-2	0,005100	0,005100	0,000014	0,00	0,00
B-3	0,047800	0,047800	0,000000	0,00	0,00
B-4	0,049500	0,049500	0,000000	0,00	0,00
B-5	0,004100	0,041000	0,000000	0,00	0,00
B-6	0,070700	0,070700	0,000000	0,00	0,00
B-7	0,049200	0,049200	0,000000	0,00	0,00
T-2	0,518800	0,518801	0,000001	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	0,999999	0,000001	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	0,999994	0,000006	0,00	0,00
T-7	1,000000	0,999998	0,000002	0,00	0,00

Tabela 6.5b: Resultados da reconciliação para as variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa	0,923996	0,915564	0,834804	0,075003	0,084438	0,080759
Desvio Real(%)	(0,00)	(0,06)	(-0,02)	(-1,44)	(0,05)	(-0,51)
Fluxo Determinado	0,923989	0,915550	0,834803	0,076003	0,084439	0,080760
Desvio Ajustado (%)	(-0,11)	(0,06)	(-0,16)	(-0,13)	(-0,07)	(-0,05)

Tabela 6.5c:Fluxos determinados das variáveis do tipo 2.

Este exemplo ilustra o caso onde a variância base, foi multiplicada por 10. Serve para a análise proposta no CASO 4.

CASO 6: Variâncias proporcionais (valor base multiplicado por 100)

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019280	0,019280	0,000000	0,00	0,00
B-1	0,049500	0,049500	0,000000	0,00	0,00
T-1	1,000000	1,000000	0,000000	0,00	0,00

Tabela 6.6a: Resultados da reconciliação das variáveis do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004500	0,000000	0,00	0,00
A-3	0,001300	0,001280	0,000020	0,00	0,00
A-4	0,000900	0,000906	0,000006	0,00	0,00
A-5	0,198800	0,198800	0,000000	0,00	0,00
A-6	0,214300	0,214301	0,000000	0,00	0,00
A-7	0,005100	0,005144	0,000014	0,28	0,00
B-2	0,047800	0,047800	0,000000	0,00	0,28
B-3	0,049500	0,049500	0,000000	0,00	0,00
B-4	0,004100	0,004100	0,000000	0,00	0,00
B-5	0,070700	0,070700	0,000000	0,00	0,00
B-6	0,049200	0,049200	0,000000	0,00	0,00
B-7	0,518800	0,518800	0,000000	0,00	0,00
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	0,999999	-0,000001	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	0,999994	-0,000006	0,00	0,00
T-7	1,000000	0,999998	-0,000002	0,00	0,00

Tabela 6.6b: Resultados da reconciliação para as variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa Desvio Real (%)	0,923996 (0,00)	0,915564 (0,06)	0,834803 (-0,02)	0,076003 (-0,13)	0,084438 (0,06)	0,080759 (-0,05)
Fluxo Determinado Desvio Ajustado (%)	0,923989 (0,00)	0,915550 (0,06)	0,834803 (-0,23)	0,076003 (0,-,13)	0,084439 (0,05)	0,080760 (-0,05)

Tabela 6.6c: Fluxos determinados das variáveis tipo 2.

Este caso foi estudado multiplicando-se a variância base por 100. Não havia erros grosseiros ou randômicos entre as variáveis. Os desvios são idênticos aos apresentados nas tabelas 6.5.

CASO 7: Variâncias proporcionais (valor base multiplicado por 0,1)

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019280	0,019280	0,000000	0,00	0,00
B-1	0,049500	0,049500	0,000000	0,00	0,00
T-1	1,000000	1,000000	0,000000	0,00	0,00

Tabela 6.7a: Resultados da reconciliação para as variáveis tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004500	0,000000	0,00	0,00
A-3	0,001300	0,001280	-0,000020	0,02	0,00
A-4	0,000900	0,000906	0,000006	0,00	0,00
A-5	0,198800	0,198800	0,000000	0,00	0,00
A-6	0,214300	0,214301	0,000001	0,00	0,00
A-7	0,005100	0,005144	0,000044	-0,27	0,00
B-2	0,047800	0,047800	0,000000	0,00	0,00
B-3	0,049500	0,049500	0,000000	0,00	0,00
B-4	0,004100	0,004100	0,000000	0,00	0,00
B-5	0,070700	0,070700	0,000000	0,00	0,00
B-6	0,049200	0,049200	0,000000	0,00	0,00
B-7	0,518800	0,518801	0,000001	0,00	0,00
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	0,999999	-0,000000	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	0,999994	-0,000006	0,00	0,00
T-7	1,000000	0,999998	-0,000002	0,00	0,00

Tabela 6.7b: Resultados da reconciliação para variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa	0,923996	0,915564	0,834804	0,076003	0,084438	0,080759
Desvio Real (%)	(0,00)	(0,06)	(-0,02)	(7,50)	(0,04)	(-0,51)
Fluxo Determinado	0,923989	0,915550	0,834803	0,076003	0,084439	0,080760
Desvio Ajustado (%)	(0,00)	(0,06)	(-0,02)	(7,50)	(0,05)	(-0,05)

Tabela 6.7c: Fluxos determinados das variáveis tipo 2.

Utilizando-se uma variância 10 vezes menor que o valor base, chegou-se a resultados muito parecidos que os conseguidos quando se multiplicou a variância por 10,

e por 100. Com base portanto, nos resultados apresentados nas tabelas 6.4, 6.5, 6.6 e nestas três últimas, pode-se concluir que na ausência de erros grosseiros, o sistema não fica mais nem menos sensível aos valores das variâncias das medidas, ou seja, a variância tem pouca influência nos resultados da reconciliação para sistemas operando dentro das condições de normalidade.

CASO 8: Variância proporcionais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019300	0,019600	0,000300	-1,36	0,00
B-1	0,049500	0,050700	0,001200	-2,36	0,00
T-1	1,000000	1,000700	0,000700	-0,07	0,00

Tabela 6.8a: Resultados da reconciliação das variáveis do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004500	0,000000	0,02	0,00
A-3	0,001300	0,002200	0,000900	-65,81	0,00
A-4	0,000900	-0,017000	-0,017900	1990,55	0,00
A-5	0,198600*	0,198600	0,000000	0,00	0,10
A-6	0,214400*	0,212800	-0,001600	0,73	0,05
A-7	0,005100	0,030700	0,025600	-501,66	0,00
B-2	0,047200*	0,047200	0,000000	0,00	-1,25
B-3	0,053600**	0,053600	0,000000	0,00	8,28
B-4	0,005000**	0,005000	0,000000	0,00	21,95
B-5	0,070900*	0,070900	0,000000	0,00	0,00
B-6	0,049500*	0,049500	0,000000	0,00	0,00
B-7	0,521000*	0,520900	-0,000100	0,00	0,42
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	1,000000	0,000000	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	1,000000	0,000000	0,00	0,00
T-7	1,000000	1,000000	0,000000	0,00	0,00

Tabela 6.8b: Resultados da reconciliação das variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa	0,920500	0,911900	0,826000	0,079500	0,087800	0,085900
Desvio Real (%)	(-0,37)	(-0,38)	(-1,07)	(4,46)	(4,03)	(6,31)
Fluxo Determinado	0,940424	0,892126	0,833844	0,077180	0,085792	0,081093
Desvio Ajustado (%)	(1,77)	(-2,50)	(-0,14)	(1,42)	(1,65)	(0,36)

Tabela 6.8c: Fluxos determinados das variáveis tipo 2

Este exemplo, juntamente com os próximos 3, servirá para investigar a influência da posição do erro grosseiro nos resultados da reconciliação. Lembrar que erros grosseiros em qualquer posição inviabiliza o conjunto de dados. O que se quer aqui é saber se existe diferença significativa entre o erro estar nas variáveis do tipo 1 ou 2. Neste caso, introduziu-se duas medidas do tipo 2 com erros grosseiros. Nenhuma medida do tipo 1 contém tal erro.

CASO 9: Variâncias proporcionais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019300	0,019613	0,000313	-1,62	0,00
B-1	0,049500	0,050959	0,001459	-2,94	0,00
T-1	1,000000	1,000931	0,000931	-0,09	0,00

Tabela 6.9a: Resultados da reconciliação de dados do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004499	-0,000001	0,02	0,00
A-3	0,001300	0,001930	0,000630	-48,49	0,00
A-4	0,000900	-0,012239	-0,013139	1459,94	0,00
A-5	0,198600*	0,198608	0,000008	0,00	-0,10
A-6	0,214400*	0,212992	-0,001478	0,69	0,05
A-7	0,005100	0,026198	0,021098	413,69	0,00
B-2	0,047200*	0,047200	0,000000	0,00	-1,25
B-3	0,053600**	0,053600	0,000000	0,00	8,28
B-4	0,004100	0,004122	0,000022	0,00	0,00
B-5	0,080000**	0,079988	-0,000012	0,01	13,15
B-6	0,049500*	0,049511	0,000011	0,00	0,61
B-7	0,521000*	0,520918	-0,000083	0,02	0,42
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	0,999999	-0,000001	0,00	0,00
T-4	1,000000	1,000023	0,000023	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	1,000010	0,000010	0,00	0,00
T-7	1,000000	1,000004	0,000004	0,00	0,00

Tabela 6.9b: Resultados da reconciliação das medidas do tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal.Massa	0,920647	0,912208	0,824852	0,079315	0,087570	0,087356
Desvio Real (%)	(-0,36)	(-0,31)	(-1,22)	(12,18)	(3,76)	(8,11)
Fluxo Determinado	0,935668	0,896015	0,831809	0,077564	0,086010	0,083092
Desvio Ajustado (%)	(1,26)	(-2,07)	(-0,38)	(0,19)	(1,90)	(3,84)

Tabela 6.9c: Fluxos determinados das variáveis do tipo 2.

Neste caso mudou-se a posição dos erros grosseiros no conjunto de variáveis do tipo 2. Mantendo-se o número de erros grosseiros presentes e a ordem de grandeza dos mesmos, tem-se que a mudança continua a influenciar pouco os resultados reconciliados das variáveis do tipo 1 e os fluxos das variáveis do tipo 2. No caso das concentrações (ou medidas do tipo 2) continua a haver maior influência apresentando graves desvios nos valores reconciliados.

CASO 10: Variâncias proporcionais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019300	0,021256	0,001956	-10,14	0,00
B-1	0,038100**	0,046330	0,008230	-21,60	-23,03
T-1	1,000000	1,005402	0,005402	-0,54	0,00

Tabela 6.10a: Resultados da reconciliação das medidas do tipo 1

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004502	0,000002	-0,05	0,00
A-3	0,001300	-0,000509	-0,001809	139,15	0,00
A-4	0,000900	0,038779	0,037879	-4208,79	0,00
A-5	0,198600*	0,198624	0,000024	-0,12	-0,10
A-6	0,214400*	0,209699	-0,004701	2,19	0,05
A-7	0,005100	0,074410	0,069310	-1359,03	0,00
B-2	0,047200*	0,047200	0,000000	0,00	-1,25
B-3	0,053600**	0,053600	0,000000	0,00	8,28
B-4	0,004100	0,004168	0,000068	0,00	-1,67
B-5	0,070900*	0,070923	0,000023	0,00	0,28
B-6	0,049500*	0,049476	0,000024	0,00	0,61
B-7	0,521000*	0,521220	0,000220	0,00	0,42
T-2	1,000000	0,999999	-0,000001	0,00	0,00
T-3	1,000000	0,999997	-0,000003	0,00	0,00
T-4	1,000000	1,000068	0,000068	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	1,000031	0,000031	0,00	0,00
T-7	1,000000	1,000012	0,000012	0,00	0,00

Tabela 6.10b: Resultados da reconciliação para as variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa	0,926844	0,917686	0,829806	0,072622	0,081547	0,087881
Desvio Real (%)	(0,31)	(0,29)	(-0,62)	(-4,57)	(-3,38)	(8,76)
Fluxo Determinado	0,884902	0,852433	0,838050	0,086962	0,093880	0,073407
Desvio Ajustado (%)	(-4,23)	(-7,33)	(0,36)	(14,27)	(11,10)	(-9,15)

Tabela 6.10c: Fluxos determinados para as variáveis tipo 2.

Mantendo-se um erro grosseiro (mais erros aleatórios) entre as medidas do tipo 2 e introduzindo-se um erro grosseiro entre as do tipo 1, verifica-se que todo o conjunto de medidas reconciliadas fica comprometido, inclusive os fluxos determinados. Isto sugere que erros nas medidas do tipo 1 influenciam mais o conjunto reconciliado que as do tipo 2.

CASO 11: Variâncias proporcionais

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019399	0,020996	0,001696	-8.78	0,00
B-1	0,038100**	0,045359	0,007259	-19,05	-23,03
T-1	1,000000	1,004765	0,004765	-0,48	0,00

Tabela 6.11a: Resultados reconciliados das variáveis tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004503	0,000003	-0,06	0,00
A-3	0,001300	-0,001372	-0,002672	205,52	0,00
A-4	0,000900	0,056174	0,055274	-6141,60	0,00
A-5	0,198800	0,198796	-0,000004	0,00	0,00
A-6	0,214300	0,211869	-0,002431	1,13	0,00
A-7	0,005100	0,057325	0,052225	-1024,02	0,00
B-2	0,047800	0,047800	0,000000	0,00	0,00
B-3	0,049500	0,049500	0,000000	0,00	0,00
B-4	0,004100	0,004063	-0,000037	0,90	0,00
B-5	0,070700	0,070758	0,000058	-0,81	0,00
B-6	0,049200	0,049138	-0,000062	0,13	0,00
B-7	0,518800	0,519388	0,000588	-0,11	0,00
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	0,999997	-0,000003	0,00	0,00
T-4	1,000000	1,000065	0,000065	0,01	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	0,999994	-0,000006	0,00	0,00
T-7	1,000000	0,9999997	-0,000003	0,00	0,00

Tabela 6.11b: Resultados da reconciliação para as variáveis tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal Massa Desvio Real (%)	0,930064 (0,65)	0,921137 (0,67)	0,839886 (0,59)	0,069403 (-8,80)	0,078326 (-7,20)	0,081251 (0,56)
Fluxo Determinado Desvio Ajustado (%)	0,870721 (-5,77)	0,866762 (-5,27)	0,840059 (0,61)	0,085865 (12,83)	0,092731 (9,74)	0,07200 (-10,89)

Tabela 6.11c: Fluxos determinados das variáveis do tipo 2.

Retirando-se agora quaisquer tipos de erros dentre as variáveis do tipo 2, pode-se ter maiores evidências de que o procedimento é mais sensível a erros grosseiros entre as medidas do tipo 1. Combinações dos últimos quatro casos parece dar razão a esta proposição uma vez que os desvios em todas as medidas reconciliadas e determinadas apresentaram os desvios nos mesmos locais em ordens de grandeza semelhantes. Erros grosseiros entre as variáveis do tipo 1, portanto, comprometem mais o procedimento de reconciliação de dados não-lineares que erros grosseiros presentes entre as medidas de concentração das correntes.

CASOS 12, 13, 14: Variâncias proporcionais (variância base multiplicada por 10, 100 e 0,1 respectivamente)

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019300	0,021256	0,001956	-10,13	0,00
B-1	0,038100**	0,046330	0,008230	-21,60	-23,03
T-1	1,000000	1,005402	0,005402	-0,54	0,00

Tabela 6.12, 6.13 e 6.14a: Resultados da reconciliação de dados do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004500	0,004502	0,000002	-0,05	0,00
A-3	0,001300	-0,000509	-0,001809	139,15	0,00
A-4	0,000900	0,038779	0,037879	-4208,79	0,00
A-5	0,198600*	0,198624	0,000024	-0,01	-0,10
A-6	0,214400*	0,209699	-0,004701	2,19	0,05
A-7	0,005100	0,074410	0,069310	-1359,03	0,00
B-2	0,047200	0,047200	0,000000	0,00	0,00
B-3	0,053600**	0,053600	0,000000	0,00	8,28
B-4	0,004100	0,004168	0,000068	-1,68	0,00
B-5	0,070900*	0,070923	0,000023	-0,03	0,28
B-6	0,049500*	0,049476	-0,000024	0,05	0,61
B-7	0,521000*	0,521220	0,000220	0,00	0,42
T-2	1,000000	0,999999	-0,000001	0,00	0,00
T-3	1,000000	0,999997	-0,000003	0,00	0,00
T-4	1,000000	1,000068	0,000068	0,01	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	1,000031	0,000031	0,00	0,00
T-7	1,000000	1,000012	0,000012	0,00	0,.00

Tabela 6.12, 6.13 e 6.14b: Resultados da reconciliação das variáveis do tipo 2

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa	0,926844	0,917686	0,829806	0,072622	0,081547	0,087881
Desvio Real (%)	(0,31)	(0,29)	(-0,62)	(-15,65)	(-3,38)	(8,76)
Fluxo Determinado	0,884902	0,852433	0,838050	0,086962	0,093880	0,073407
Desvio Ajustado (%)	(-4,23)	(-6,84)	(0,36)	(14,27)	(11,23)	(-16,47)

Tabela 6.12, 6.13, 6.14c: Fluxos determinados das variáveis tipo 2.

Complementando o estudo iniciado no CASO 4 e descrito no CASO 7 a respeito da influência da variância das medidas sob a presença ou não de erros grosseiros, tem que, aqui com a presença de erros muito grosseiros entre as variáveis, assim como nos exemplos anteriores quando não existiam erros de nenhuma natureza, as variâncias das medidas não são fatores de diferenciação nos resultados, ou seja, verificou-se que a influência dos erros grosseiros é maior que a das variâncias e são determinantes conforme prevê a teoria na possibilidade ou não de se proceder à reconciliação de dados de um processo. Multiplicando-se por 1000 uma variância dada, observou-se os mesmos valores reconciliados e determinados, os quais estão resumidos nas três últimas tabelas.

Uma vez concluído o estudo sobre as variâncias das medidas e sobre a influência da posição dos erros grosseiros nas variáveis medidas, pode-se estudar a influência da ordem de grandeza de um erro grosseiro em uma variável do tipo 1. É isto que será feito nos próximos cinco casos. Aqui adotou-se medidas que apresentavam erros de 5, 10, 15 20 e 30% em relação ao valor real da segunda medida das variáveis do tipo 1. Existe também um erro grosseiro entre as variáveis do tipo 2. Os resultados podem ser comparados abaixo.

CASOS 15 a 19: Variáveis proporcionais

CC	%	VALOR REAL	VC(-5%)	VC(-10%)	VC(-15%)	VC(-20%)	VC(-30%)
A-1		0,019280	0,019910	0,020280	0,020645	0,021025	0,021786
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(3,26)	(5,18)	(7,08)	(9,05)	(12,99)
B-1		0,049500	0,049982	0,049150	0,048217	0,047098	0,044269
**	DM		(-5,00)**	(-10,00)**	(-15,00)**	(-20,00)**	(30,00)**
	DRec		(0,97)	(-0,70)	(-2,59)	(-4,85)	(-10,57)
T-1		1,000000	1,001866	1,002949	1,003937	1,004875	1,006475
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,19)	(0,29)	(0,39)	(0,48)	(0,65)
%	Dmed1	-	(1,67)	(3,33)	(5,00)	(6,66)	(10,00)
	Dmed2	-	(1,47)	(2,06)	(3,35)	(4,80)	(8,07)

Tabela 6.15, 6.16, 6.17, 6.18, 6.19a: Resultados da reconciliação de dados das medidas do tipo 1.

onde,

DM= desvio do valor medido em relação ao valor real,
DRec= desvio do valor reconciliado em relação ao valor real,
Dmed1= desvio médio (em módulo) dos valores medidos em relação ao valor real,
Dmed2= desvio médio (em módulo) dos valores reconciliados em relação ao valor real.
Os demais seguem as convenções anteriores.

Pode-se notar que a reconciliação de dados utilizando medidas com erros grosseiros não "consegue" corrigir os valores medidos contaminados com erros grosseiros. Quanto mais errada a medida, menos os valores reconciliados se aproximam dos valores reais. Pode-se notar isto olhando-se a tabela acima (tabela 6.15 a 6.19a) a despeito da pequena melhora nos valores dos desvios médios, os quais mostram a tendência do método em aproximar os valores medidos aos valores reais.

CC	%	CME REAL	CCOR(5)	CCOR(10)	CCOR(15)	CCOR(20)	CCOR(30)
A-2		0,004500	0,004500	0,004500	0,004501	0,004501	0,004503
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,00)	(0,00)	(0,00)	(0,22)	(-0,07)
A-3		0,001280*	0,001742	0,001238	0,000633	0,000026	-0,001841
	DM		(1,56)	(1,56)	(1,56)	(1,56)	(1,56)
	DRec		(3,40)	(0,29)	(-94,87)	(98,00)	(241,59)
A-4		0,000900	0,008346	0,002209	0,141540	0,028663	0,066666
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(827,93)	(145,44)	(-1472,67)	(-3084,81)	(-7307,34)
A-5		0,198600*	0,198612	0,198614	0,198617	0,198621	0,198631
	DM		(0,05)	(0,05)	(0,05)	(0,05)	(0,05)
	DRec		(396,04)	(-0,09)	(-0,09)	(-0,07)	(-0,08)
A-6		0,214400*	0,212268	0,211692	0,211040	0,210249	0,208189
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(-0,95)	(-1,22)	(1,49)	(1,89)	(2,85)
A-7		0,005130	0,035974	0,044598	0,054352	0,066181	0,097003
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(605,37)	(774,47)	(-965,73)	(-1197,67)	(-1802,01)
B-2		0,004780*	0,047200	0,047200	0,047200	0,04720	0,047200
	DM		(-1,25)	(-1,25)	(-1,25)	(-1,25)	(-1,25)
	DRec		(-1,25)	(-1,26)	(-1,25)	(-1,25)	(-1,25)
B-3		0,049500**	0,053600	0,053600	0,053600	0,053600	0,053600
	DM		(8,28)	(8,28)	(8,28)	(8,28)	(8,28)
	DRec		(8,28)	(8,28)	(8,28)	(8,28)	(8,28)
B-4		0,004100	0,004132	0,004140	0,004149	0,004161	0,004189
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,78)	(0,98)	(-1,19)	(-1,48)	(2,17)

B-5		0,070700*	0,070894	0,070901	0,070908	0,070917	0,070937
	DM		(0,28)	(0,28)	(0,28)	(0,28)	(0,28)
	DRec		(0,27)	(0,28)	(-0,29)	(-0,31)	(0,34)
B-6		0,049200*	0,049506	0,049499	0,040401	0,049482	0,049461
	DM		(0,61)	(0,61)	(0,61)	(0,61)	(0,61)
	DRec		(0,62)	(0,61)	(0,59)	(0,57)	(0,53)
B-7		0,051880*	0,520940	0,521008	0,521081	0,521165	0,521361
	DM		(0,42)	(0,43)	(0,43)	(0,43)	(0,43)
	DRec		(0,41)	(0,42)	(0,44)	(-0,45)	(-0,99)
T-2		1,000000	0,999999	0,999999	0,999999	0,999999	0,999999
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
T-3		1,000000	0,999999	0,999999	0,999998	0,999997	0,999995
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
T-4		1,000000	1,000033	1,000041	1,000050	1,000060	1,000087
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
T-5		1,000000	1,000000	1,000000	1,000000	1,000000	1,000000
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
T-6		1,000000	1,000013	1,000017	1,000021	1,000027	1,000042
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	Drec		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
T-7		1,000000	1,000005	1,000007	1,000000	1,000010	1,000017
	DM		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
	DRec		(0,00)	(0,00)	(0,00)	(0,00)	(0,00)
%	Dme1	-	(0,61)	(0,61)	(0,61)	(0,61)	(0,61)
	Dme2	-	(102,52)	(52,08)	(141,49)	(244,17)	(520,42)

Tabela 6.15, 6.16, 6.17, 6.18, 6.19b: Resultados da reconciliação das variáveis do tipo 2.

Na reconciliação das medidas do tipo 2 pode-se notar que a mesma tendência anteriormente observada para o caso das variáveis do tipo 1 aqui se repete. Exceto para o caso de 10% de erro na segunda variável do tipo 1, em todos os demais casos, quanto maior a porcentagem de erro do valor medido em relação ao valor real, maior o desvio do valor reconciliado em relação ao valor real.

	FLUXO REAL	FLUXO (5%)	FLUXO (10%)	FLUXO (15%)	FLUXO (20%)	FLUXO (30%)
F-2	0,924000	0,931325	0,920487	0,908530	0,894445	0,859717
DRec		(0,79)	(-0,38)	(-1,67)	(-3,19)	(-6,96)
F-3	0,915000	0,887037	0,879025	0,870133	0,859599	0,833433
DRec		(3,15)	(-3,93)	(-4,90)	(-6,05)	(-8,91)
F-4	0,835000	0,833328	0,833316	0,835619	0,837057	0,840765
Drec		(-0,20)	(-0,70)	(0,70)	(0,25)	(0,69)

F-5 DRec	0,076100	0,079156 (4,01)	0,081257 (6,77)	0.083359 (9,54)	0,085583 (12,45)	0,090185 (18,51)
F-6 Drec	0,084400	0,087439 (3,47)	0,089200 (5,56)	0,090944 (7,62)	0,092765 (9,78)	0,096439 (14,13)
F-7 DRec	0,080800	0,081068 (0,33)	0,079295 (-1,86)	0,077328 (-4,30)	0,074995 (-7,18)	0,069186 (-14,34)
Dmed3	-	(1,99)	(1,43)	(1,95)	(2,59)	(5,06)

Tabela 6.15, 6.16, 6.17, 6.18, 6.19c: Fluxos determinados das variáveis tipo 2.

onde,

F=Fluxo total determinado,
Dmed3= Desvio médio entre os valores determinados e os valores reais.

Os valores dos fluxos determinados para os últimos cinco casos estudados estão de acordo com o esperado e conforme já observado para as demais variáveis referentes a estes casos. À medida que se afasta dos valores reais, tem-se piores estimativas para os valores determinados das variáveis não medidas e pode-se, nas tabelas acima, comprovar isto quando se observa os desvios entre os valores esperados e os valores reais que eram teoricamente conhecidos (através de balanço de massa).

O último caso a ser estudado dentro do exemplo apresentado (circuito de flotação) é quando se tem apenas flutuações randômicas nos valores das variáveis. Foram, a partir dos resultados apresentados na literatura, inseridos alguns erros "aleatórios" entre 2 e 3% na variáveis do tipo 1 e 2. Os resultados podem ser conferidos na tabela 6.20.

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,019286	0,019860	0,000002	-3,00	3,00
B-1	0,049500	0,049544	0,000010	-0,02	0,00
T-1	1,000000	1,000006	0,000006	0,00	0,00

Tabela 6.20a: Valores reconciliados das variáveis do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-2	0,004590*	0,004590	0,000000	0,00	2,00
A-3	0,001300	0,012800	0,000200	0,00	0,00
A-4	0,000900	0,000900	0,000000	-0,01	0,00
A-5	0,202776*	0,198797	0,003798	1,80	2,00
A-6	0,214300*	0,214300	0,000000	0,00	0,00
A-7	0,005500*	0,005102	0,000398	-7,20	7,28
B-2	0,047800*	0,047815	0,000015	-0,03	2,00
B-3	0,049500	0,049470	0,000030	0,00	0,00
B-4	0,004100*	0,040540	0,004600	0,00	0,00

B-5	0,070700	0,070700	0,000000	0,00	0,00
B-6	0,050184*	0,049200	0,000984	1,90	2,00
B-7	0,518800	0,518800	0,000000	0,00	0,00
T-2	1,000000	1,000000	0,000000	0,00	0,00
T-3	1,000000	1,000000	0,000000	0,00	0,00
T-4	1,000000	1,000000	0,000000	0,00	0,00
T-5	1,000000	1,000000	0,000000	0,00	0,00
T-6	1,000000	0,999998	0,000002	0,00	0,00
T-7	1,0000000	0,999999	0,000001	0,00	0,00

Tabela 6.20b: Resultados da reconciliação dos valores das variáveis do tipo 2.

Correntes (tipo 2)	2	3	4	5	6	7
Fluxo Real	0,924000	0,915000	0,835000	0,076100	0,084400	0,080800
Fluxo pelo Bal. Massa	0,923900	0,913689	0,833228	0,076056	0,086311	0,080461
Desvio Real (%)	(-0,10)	(-0,15)	(-0,21)	(-0,06)	(2,26)	(-0,42)
Fluxo Determinado	0,923889	0,913762	0,833211	0,076074	0,084327	0,080478
Desvio Ajustado (%)	(-0,01)	(-0,13)	(-0,21)	(-0,03)	(-0,09)	(-0,38)

Tabela 6.20c: Fluxos determinados das variáveis do tipo 2.

Este último caso, a exemplo dos apresentados nas tabelas de 6.4 a 6.7, apenas utiliza medidas com erros não-grosseiros. Aqui foram introduzidos erros sistemáticos e mais uma vez se comprovou que o procedimento diminui o desvio entre os valores medidos e os valores teóricos. Não se preocupou aqui em medir este desvio devido ao fato, já mencionado anteriormente, segundo o qual não são conhecidos os valores reais dos fluxos e das concentrações. A utilização de apenas dois algarismos significativos nos valores dos desvios não produziria um retrato fiel da situação, deixando os resultados quantitativos dos desvios sem significação real. Qualitativamente, contudo, os resultados apresentados nestas tabelas comprovam completamente os resultados previstos pela teoria.

A consideração dos valores descritos em CROWE(1986) como sendo os valores reais não poderia ter sido levada a termo visto que a introdução de erros entre 2 e 3% nos valores de algumas variáveis medidas não melhorou significativamente os valores reconciliados. Os desvios entre os valores supostamente reais e os reconciliados apresentaram desvios da mesma ordem de grandeza o que pode significar que o procedimento apenas "trouxe" os valores modificados para os mesmos níveis que os apresentados pelo citado autor. A comparação entre os valores medidos aqui reconciliados e os apresentados na literatura não apresenta desvios que possam ser abrangidos pela escala de significância dos valores, ou seja, todos os desvios entre os do modelo e os determinados ficaram abaixo de 0,01%. Sobre a confiabilidade dos valores descritos inicialmente em ICHIYEN(1973), pode-se ler em CROWE(1986) : "é claro que há alguma coisa errada com os dados e seria interessante que IYCHIEN e SMITH notassem que as análises em raio-X para o Zn estão erradas". Isto serve para explicar os valores da corrente 8 que inicialmente não havia sido medidas.

EXEMPLO 2: CIRCUITO DE MOAGEM

O segundo exemplo estudado se refere a um circuito de moagem apresentado em SERTH(1987). Trata-se de um sistema constituído por quatro nós, sendo que dois deles (1 e 3) são moinhos. O nó 2 é uma bomba e o último (4) é uma bateria de ciclones. O material sólido é classificado em três faixas granulométricas, cujas frações são tratadas de forma análoga às espécies químicas de um sistema multicomponente; os ciclones desempenham papel semelhante ao de colunas de destilação simples e os moinhos, papel semelhante ao de um reator químico ao efetuar uma mudança nas quantidades das diferentes faixas granulométricas. Entretanto, neste tipo de nó não é conhecida a "estequiometria" do processo, somente são consideradas a conservação dos sólidos totais e da água. Na bomba e nos ciclones, além disto, levam-se em conta as restrições relativas à conservação de massa das faixas granulométricas distintas dos sólidos.

O sistema é tratado como se tivesse 4 componentes, a água é representada por A e os demais sólidos por B, C e D de acordo com sua classificação granulométrica. A vazão total é representada por T, como no exemplo anterior. Todas as vazões totais são medidas. O exemplo se refere inicialmente a um caso linear em que serão consideradas 5 correntes como não tendo os valores medidos para as vazões dos componentes, tornando assim o exemplo em um caso não- linear. As vazões as quais serão consideradas não medidas são as de número 1, 5, 6, 7 e 8. As frações mássicas são conhecidas. A figura 6.2 mostra o fluxograma simplificado do circuito de moagem. Os dados e os resultados da reconciliação estão expressos em toneladas métricas por hora.

Este exemplo servirá para mostrar a importância em se classificar o sistema que se quer controlar. Será visto aqui porque é importante se conhecer a forma correta de se distribuir os instrumentos de medida em posições que possam representar o sistema. A distribuição dos instrumentos de medida é um fator determinante na caracterização do processo, pois se for feita sem critérios pode levar a um mal condicionamento das matrizes dos balanços, impossibilitando se atingir todos os resultados corretos. Quanto pior o condicionamento de uma matriz, pior será a sua "inversa". Foram aqui utilizados dois métodos de cálculo diferentes de cálculos de "inversas" de matrizes não quadradas em dois programas computacionais diferentes (método SVD e RQ) (Apêndice B), os resultados não apresentaram diferenças numéricas mostrando que o problema é mesmo de condicionamento das matrizes como consequência de uma má distribuição dos instrumentos de medida. Os cálculos subsequentes ficam inviabilizados se se prossegue com os resultados incorretos.

Este tipo de problema não ocorre em sistemas lineares pois nenhuma matriz mal condicionada precisa ser invertida. Para um sistema com estas características, é necessário que se use um método de CLASSIFICAÇÃO de variáveis a fim de se distribuir corretamente os instrumentos pela planta. Este procedimento não faz parte deste trabalho, contudo, fica aqui registrado como alerta a sua importância. STEWART(1977) mostra a importância das perturbações nos valores de uma matriz não quadrada.

CIRCUITO DE MOAGEM

Como seria de se esperar, o mal condicionamento da matriz D (equação 4.12) leva a resultados pouco satisfatórios em termos de balanços das equações, o que faz com que os valores dos fluxos determinados (não conhecidos previamente) apresentem

desvios bastante significativos em relação ao esperado. Os resulados ficam comprometidos por uma limitação matemática e não por causa do método de reconciliação. A parcela referente aos valores reconciliados das variáveios do tipo 1 e 2 não são afetadas por estes problemas aqui explicados. A tabela 6.21 sumariza os resultados da reconciliação de dados. Neste exemplo eram dados os valores reais e os valores de Ei são dados em relação a eles.

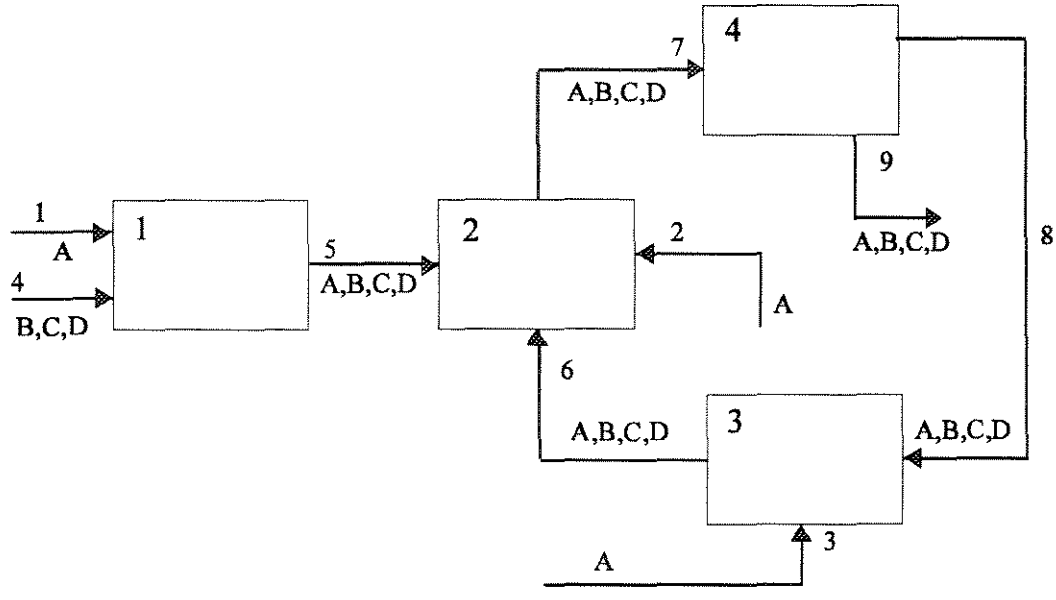


Figura 6.2: Fluxograma simplificado do circuito de moagem.

Componentes: A (água), B, C e D (elementos sólidos de diferentes granulometrias)
Correntes do tipo 1: 2, 3, 4 e 9.
Correntes do tipo 2: 1, 5, 6, 7 e 8.
Correntes do tipo 3: nenhuma.

CC	VM	VC	aj	Er(%)	Ei(%)
A-2	314,12*	314,12	0,000	0,00	1,91
A-3	24,81*	24,80	0,010	-0,04	-0,76
B-4	24,34*	24,38	0,036	0,12	0,45
C-4	142,16*	141,82	-0,340	-0,24	0,24
D-4	91,76*	91,90	0,140	0,15	0,01
T-4	258,45*	257,93	-0,520	-0,20	0,21
A-9	435,95*	438,81	2,860	0,65	-1,99
B-9	0,1386*	0,1386	0,000	0,00	-1,00
C-9	49,91*	49,80	-0,110	-0,22	-,148
D-9	205,62*	207,17	1,550	0,75	-0,71
T-9	691,23*	688,99	-2,240	-0,32	-1,62

Tabela 6.21a; Resultados da reconciliação dos dados do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-1	1,000000	1,000000	0,000000	0,00	0,00
A-5	0,264716*	0,264822	0,000106	-0,04	-5,01
B-5	0,068033*	0,068000	-0,000033	-0,01	0,64
C-5	0,405661*	0,396400	-0,009261	-2,28	2,28
D-5	0,261596*	0,261600	0,000323	-0,12	1,79
T-5	1,000000	1,000012	0,000012	0,00	0,00
A-6	0,247975*	0,243912	-0,004063	-0,16	-0,40
B-6	0,010234*	0,010300	0,000066	0,64	-0,64
C-6	0,460900*	0,461567	0,000692	-0,15	-0,28
D-6	0,280914*	0,281130	0,000215	-0,08	-0,59
T-6	1,000000	1,000012	0,000012	0,35	0,00
A-7	0,396204*	0,388800	-0,007404	-1,86	2,40
B-7	0,198333*	0,198440	0,000107	0,05	-0,83
C-7	0,360589*	0,365400	0,004811	1,33	-1,55
D-7	0,223374*	0,224084	0,000711	-0,32	-1,51
T-7	1,000000	1,000123	0,000123	-0,08	0,00
A-8	0,232892*	0,235632	0,006712	1,17	-3,48
B-8	0,031571*	0,031926	0,000354	-1,12	-3,75
C-8	0,571004*	0,561211	-0,009793	-1,71	2,58
D-8	0,182954*	0,183064	0,000118	-0,13	-3,35
T-8	1,000000	1,000760	0,000760	-0,08	0,00

Tabela 6.21b: Resultados da reconciliação de dados do tipo 2.

Correntes (tipo 2)	1	5	6	7	8
Fluxo Real	99,70	357,60	1109,40	1787,00	1084,30
Fluxo pelo Bal. Massa	96,41	364,19	883,01	1404,02	790,06
Desvio Real (%)	(-3,30)	(1,84)	(-20,41)	(-21,43)	(-27,14)
Fluxo Determinado	96,39	364,09	882,13	1402,23	789,86
Desvio Ajustado (%)	(-3,32)	(1,81)	(-20,46)	(-21,53)	(-27,15)

Tabela 6.21c: Fluxos determinados das variáveis do tipo 2.

Nas tabelas (6.21a,b), evitou-se colocar os valores reais, os quais foi dito serem conhecidos, porque estes valores não são, na prática, conhecidos, são apenas frutos de balanços teóricos os quais não se verificam com a precisão com que podem ser calculados. Desta forma, aqui foram utilizados apenas para mostrar que os valores reconciliados se aproximam dos valores teóricos, sendo expressos implicitamente através dos valores de Ei.

Este exemplo mostra, então, as consequências do mal-condicionamento de matrizes não quadradas quando é preciso invertê-las (tabela 6.21c). Nas tabelas 6.21a e b, onde os cálculos não dependiam da inversão de matrizes do tipo especificado, nota-se que os resultados apresentam os desvios previstos. Todos os valores reconciliados (não os determinados) estão em conformidade com os resultados apresentados na literatura

não apresentando desvio relativo que possa ser representado com dois algarismos decimais significativos, são portanto, suprimidos tais desvios.

EXEMPLO 3: SISTEMA DE SEPARAÇÃO

O terceiro e último exemplo estudado para reconciliação de dados de processo é baseado num exemplo sugerido em CROWE(1983). Trata-se de um circuito no qual é possível se verificar a presença de vazamento ou não em alguma corrente. Foi feito um balanço global de massa onde se conhecia a princípio os valores de todas as correntes totais e suas respectivas composições. De posse de tal conjunto de dados, pode-se arranjá-los de tal modo que as correntes medidas e as não-medidas pode ser escolhida de forma a se ter um conjunto de equações que representem uma matriz de balanços bem condicionada. Este exemplo representa um sistema de separação não particular, conforme descrito em CROWE. Não são portanto, associados elementos ou substâncias químicas particulares a cada corrente. As variâncias das correntes totais são unitárias, também segundo o mesmo exemplo. A figura 6.3 mostra um fluxograma representativo do processo estudado. O vazamento a ser determinado pode estar em qualquer uma das correntes do sistema. As tabelas 6.22 mostram o sistema sem vazamento e portanto sem erro grosseiro, onde todos os valores estão dentro das variações aceitáveis pelo método e portanto comprova a utilidade do mesmo quando não se tem erros grosseiros. Na tabela 6.23 é apresentado um caso em que existe um vazamento o que se configura como erro grosseiro. O vazamento comprometeu o equilíbrio de massa entre as correntes tornando a reconciliação inviável neste caso. Apenas como elemento de comparação ao EXEMPLO 2, foram escolhidas duas configurações de valores medidos que expõem novamente o problema de condicionabilidade das matrizes de balanço. As tabelas 6.24 e 6.25 mostram estes caso. A base de massa de entrada, aqui, é unitária portanto as correntes apresentam valores fracionários.

SISTEMA DE SEPARAÇÃO

Processo de Separação

Componentes: A, B e C (não particulares).
Correntes do tipo 1: 1, 4, 5, 6, 7, 9 e 12 (sistema bem condicionado).
Correntes do tipo 2: 2, 3, 8, 10 e 11.
Correntes do tipo 3: nenhuma.

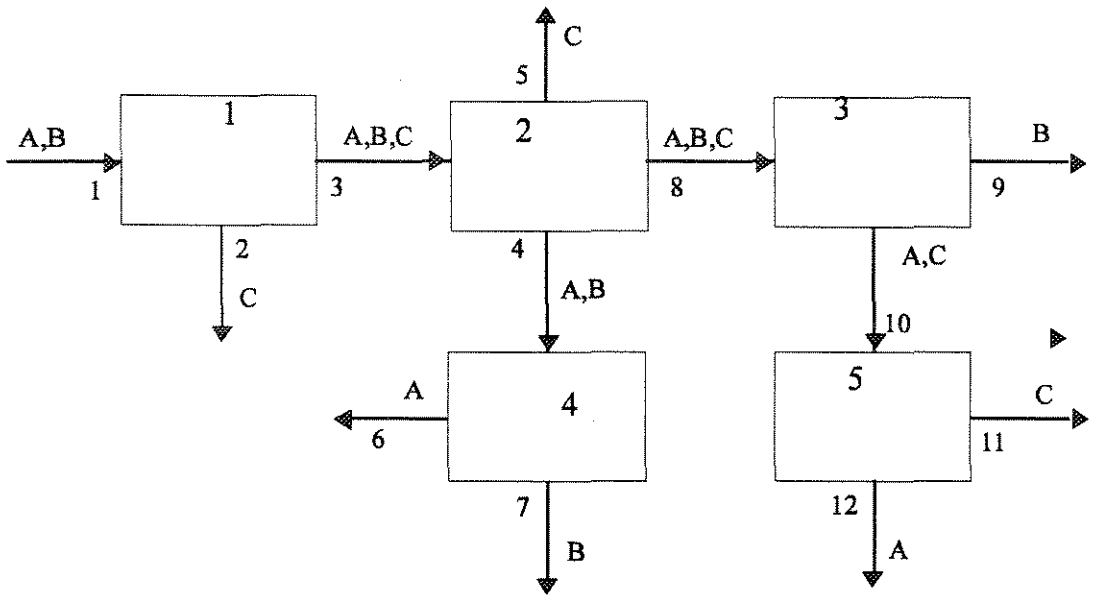


Figura 6.3: Fluxograma simplificado do processo de separação de três componentes A,B,C

CASO 1: Sistema sem erros grosseiros

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,717500*	0,717500	0,000000	0,00	2,50
B-1	0,295500*	0,295854	0,000354	-0,11	-1,50
T-1	1,008500*	1,002220	-0,006278	1,35	0,85
A-4	0,525000	0,523952	-0,001048	0,19	0,00
B-4	0,175000	0,175059	0,000059	-0,03	0,00
T-4	0,700000	0,700243	0,000243	-0,03	0,00
C-5	0,387600*	0,388003	0,000437	-0,11	3,10
T-5	0,400000	0,399512	-0,000488	0,12	0,00
A-6	0,532087*	0,530969	-0,001119	0,21	1,35
T-6	0,537600*	0,528001	-0,009599	-1,78	2,40
B-7	0,176365*	0,176410	0,000045	-0,02	0,78
T-7	0,175000	0,175000	0,000000	0,00	0,00
B-9	0,150000	0,150000	0,000000	0,00	0,00
T-9	0,147750*	0,148789	0,001039	0,70	1,50
A-12	0,297300*	0,297300	0,000000	0,00	-0,90
T-12	0,303900*	0,300002	-0,003898	-1,28	1,30

Tabela 6.22a: Resultados da reconciliação dos dados do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-3	0,351200*	0,350000	-0,001200	-0,34	0,34
A-8	0,333300	0,332967	-0,000333	0,09	0,00
A-10	0,400000	0,400108	0,000107	-0,03	0,00
B-3	0,150000	0,150000	0,000000	0,00	0,00
B-8	0,163500*	0,165689	0,002189	1,35	-0,52
C-2	1,000000	0,999894	0,000105	0,01	0,00
C-3	0,498600*	0,498510	-0,000090	0,02	-0,28
C-8	0,498600*	0,498665	0,000065	-0,01	-0,28
C-10	0,629000*	0,613672	-0,015328	-2,44	4,83
C-11	1,000000*	0,999983	-0,000017	0,00	-1,10
T-2	1,000000	1,000024	0,000024	0,00	0,00
T-3	1,001700*	1,001735	0,000035	0,00	0,17
T-8	1,000000	0,000884	-0,000116	0,01	0,00
T-10	1,004000	1,004052	0,000053	0,00	0,00
T-11	1,000000*	0,999947	-0,000053	0,00	1,30

Tabela 6.22b: Resultados da reconciliação dos valores do tipo 2

Correntes (tipo 2)	2	3	8	10	11
Fluxo Real	1,000000	2,000000	0,900000	0,750000	0,450000
Fluxo pelo Bal. Massa	1,000614	2,006084	0,913836	0,761261	0,459099
Desvio Real (%)	(0,06)	(0,30)	(1,53)	(1,50)	(2,00)
Fluxo Determinado	0,994610	1,995128	0,913780	0,761249	0,459043
Desvio Ajustado (%)	(-0,54)	(-0,24)	(1,53)	(1,49)	(2,01)

Tabela 6.22c: Fluxos determinados das variáveis do tipo 2.

As tabelas 6.22 mostram a reconciliação de dados de um sistema sem erros grosseiros. Os resultados estão de acordo com o apresentado na literatura e mostram aqui também que nas condições previstas pelo método, o procedimento realmente diminui os desvios das variáveis medidas em relação aos valores reais (teóricos).

CASO 2: Sistema com vazamento em uma das correntes.

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,700000	0,700000	0,000000	0,00	0,00
B-1	0,300000	0,320513	0,020513	-6,84	0,00
T-1	1,000000	1,144121	0,144121	-14,41	0,00
A-4	0,375000**	0,400000	0,025000	-6,67	-40,00
B-4	0,125000**	0,170513	0,045513	-36,41	-40,00
T-4	0,700000	0,714062	0,014062	-2,01	0,00
C-5	0,400000	0,444214	0,044214	-11,05	0,00
T-5	0,400000	0,344122	-0,055878	13,96	0,00
A-6	0,525000	0,400000	-0,125000	23,81	0,00

T-6	0,525000	0,539062	0,014062	-2,67	0,00
B-7	0,175000	0,170513	-0,004487	2,56	0,00
T-7	0,175000	0,175000	0,000000	0,00	0,00
B-9	0,150000	0,150000	0,000000	0,00	0,00
T-9	0,150000	0,150000	0,000000	0,00	0,00
A-12	0,300000	0,300000	0,000000	0,00	0,00
T-12	0,300000	0,300000	0,000000	0,00	0,00

Tabela 6.23a: Resultados da reconciliação de dados das variáveis do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-3	0,350000	0,369344	0,019344	-5,52	0,00
A-8	0,333333	0,313956	-0,019344	5,80	0,00
A-10	0,400000	0,409044	0,009044	-2,26	0,00
B-3	0,150000	0,176523	0,026523	-17,68	0,00
B-8	0,166667	0,069906	-0,096754	58,05	0,00
C-2	1,000000	0,990851	-0,009149	0,91	0,00
C-3	0,500000	0,501026	0,001026	-0,20	0,00
C-8	0,500000	0,500518	0,0000518	-0,10	0,00
C-10	0,600000	0,598080	-0,001920	0,32	0,00
C-11	1,000000	0,997654	-0,002346	0,23	0,00
T-2	1,000000	0,001504	0,001504	-0,15	0,00
T-3	1,000000	0,998772	-0,001228	0,12	0,00
T-8	1,000000	0,998287	-0,001713	0,17	0,00
T-10	1,000000	1,002164	0,002165	-0,21	0,00
T-11	1,000000	0,997835	-0,002165	0,21	0,00

Tabela 6.23b: Resultados da reconciliação das variáveis do tipo 2.

Corrente (tipo 2)	2	3	8	10	11
Fluxo Real	1,000000	2,000000	0,900000	0,750000	0,450000
Fluxo pelo Bal. Massa	0,980716	1,974288	0,932121	0,776152	0,470921
Desvio Real (%)	(-1,92)	(-1,29)	(3,57)	(3,48)	(4,65)
Fluxo Determinado	0,862423	1,976595	0,905785	0,756955	0,458284
Desvio Ajustado (%)	(-13,76)	(-1,17)	(0,64)	(0,92)	(1,84)

Tabela 6.23c: Fluxos determinados das variáveis do tipo 2.

A corrente 4, em condições normais de operação, recebe 35% do fluxo da corrente 3. Aqui, fez-se esta porcentagem passar a 25% daquele valor. Tem-se caracterizado, então, um erro grosseiro causado por um vazamento. Deste modo, os 10% correspondentes à corrente 3 significam 28,57% da corrente 4 (a que está com o vazamento). Os resultados da reconciliação como não poderia deixar de ser, não

guardam muita proximidade com os valores esperados, demonstrando mais uma vez que em sistemas com erros grosseiros não é possível reconciliar os dados ser incorrer em análises falsas. Este caso está previsto em NARASIMHAN(1987) onde a origem dos erros grosseiros deveria ser identificada antes de se fazer a reconciliação. No caso descrito neste exemplo não se poderia utilizar as técnicas comuns de detecção de erros grosseiros uma vez que as mesmas prevêem o descarte da medida com o erro grosseiro, o que seria impossível neste caso. O fato de o erro estar em uma variável do tipo 1 também pode ter colaborado para piorar os resultados da reconciliação, conforme descrito no EXEMPLO 1.

CASO 3: Sistema de separação com configuração de variáveis medidas diferente. (Teste para o condicionamento da matriz dos balanços).

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,700000	0,700571	0,000571	-0,08	0,00
B-1	0,300000	0,300290	0,000209	-0,07	0,00
T-1	1,000000	1,000794	0,000794	-0,08	0,00
A-6	0,525000	0,519440	-0,000560	0,11	0,00
T-6	0,525000	0,519580	-0,000419	0,08	0,00
A-8	0,299970	0,299901	-0,000069	0,03	0,00
B-8	0,149990	0,150017	0,000027	-0,02	0,00
C-8	0,450000	0,449920	-0,000080	0,02	0,00
T-8	0,900000	0,899852	-0,000148	0,02	0,00
A-12	0,300000	0,299931	-0,000069	0,02	0,00
T-12	0,300000	0,299944	-0,000056	0,02	0,00

Tabela 6.24a: Resultados da reconciliação dos valores do tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-3	0,350000	0,349993	-0,000007	0,002001	0,00
A-4	0,750000	0,750007	0,000007	-0,000930	0,00
A-10	0,400000	0,400059	0,000059	-0,014827	0,00
B-3	0,150000	0,149969	-0,000031	-0,020812	0,00
B-4	0,250000	0,249981	-0,000019	0,007719	0,00
B-7	1,000000	1,000077	0,000077	-0,007725	0,00
B-9	1,000000	0,999677	-0,000323	-0,032312	0,00
C-2	1,000000	1,000126	0,000127	-0,012648	0,00
C-3	0,500000	0,499925	-0,000075	0,015050	0,00
C-5	1,000000	1,000000	0,000000	0,000000	0,00
C-10	0,600000	0,600000	0,000000	0,000000	0,00
C-11	1,000000	1,000047	0,000047	-0,004697	0,00
T-2	1,000000	0,999953	-0,000047	0,004691	0,00
T-3	1,000000	0,999977	-0,000023	0,002283	0,00
T-4	1,000000	1,000023	0,000023	-0,002289	0,00
T-5	1,000000	1,000018	0,000019	-0,001848	0,00
T-7	1,000000	0,999951	-0,000049	0,004894	0,00

T-9	1,000000	1,000053	0,000035	-0,003529	0,00
T-10	1,000000	0,999985	-0,000015	0,001526	0,00
T-11	1,000000	1,000015	0,000015	-0,001526	0,00

Tabela 6.24b: Resultados da reconciliação das variáveis do tipo 2.

Correntes (tipo 2)	Fluxo Real	Fluxo pelo Bal. Massa (Desvio Real (%))	Fluxo Determinado (Desvio Ajustado (%))
2	1,000000	1,067041 (6,70)**	1,067135 (6,71)**
3	2,000000	2,089388 (4,46)	2,090019 (4,50)
4	0,700000	0,633239 (-9,54)**	0,633341 (-9,52)**
5	0,400000	0,575422 (43,85)**	0,575998 (43,99)**
7	0,175000	0,135774 (-22,41)**	0,136063 (-22,25)**
9	0,150000	0,149998 (0,00)	0,150002 (0,00)
10	0,750000	0,749994 (0,00)	0,749836 (0,02)
11	0,450000	0,449996 (0,00)	0,449899 (0,00)

Tabela 6.24c: Fluxos determinados para as variáveis do tipo 2.

No CASO 3, como os valores medidos são iguais aos valores reais, os desvios apresentados em Er e os desvio ajustados são numericamente iguais aos valores dos desvios nas medidas reconciliadas.

CASO 3a: Segunda modificação na configuração das variáveis medidas.

CC	VM	VC	aj	Er(%)	Ei(%)
A-1	0,700000	0,700410	0,000410	-0,06	0,00
B-1	0,300000	0,300063	0,000063	-0,02	0,00
T-1	1,000000	1,000260	0,000260	-0,03	0,00
C-5	0,400000	0,400991	0,000991	-0,25	0,00
T-5	0,400000	0,400632	0,000632	-0,16	0,00
A-6	0,525000	0,524292	-0,000708	0,13	0,00
T-6	0,525000	0,524324	-0,000676	0,13	0,00
B-7	0,175000	0,174764	-0,000236	0,13	0,00
T-7	0,175000	0,174732	-0,000268	0,15	0,00
A-12	0,300000	0,299868	-0,000132	0,04	0,00
T-12	0,300000	0,299941	-0,000059	0,02	0,00

Tabela 6.25a: Resultados da reconciliação para as variáveis tipo 1.

CC	CME	CCOR	Delta	Er(%)	Ei(%)
A-3	0,350000	0,350002	0,000003	0,00	0,00
A-4	0,750000	0,749997	-0,000003	0,00	0,00
A-8	0,333333	0,333736	0,000436	-0,13	0,00
A-10	0,400000	0,400010	0,000010	0,00	0,00
B-3	0,150000	0,149530	-0,000470	0,31	0,00
B-4	0,250000	0,249628	-0,000372	0,14	0,00
B-8	0,166667	0,166666	0,000000	0,00	0,00
B-9	1,000000	1,000000	0,000000	0,00	0,00
C-2	1,000000	1,000000	0,000000	0,00	0,00
C-3	0,500000	0,499867	-0,000133	0,03	0,00
C-8	0,500000	0,500001	0,000001	0,00	0,00
C-10	0,600000	0,600149	0,000149	-0,25	0,00
C-11	1,000000	1,000096	0,000096	-0,01	0,00
T-2	1,000000	0,999989	-0,000011	0,00	0,00
T-3	1,000000	1,000011	0,000011	0,00	0,00
T-4	1,000000	0,999906	-0,000094	0,01	0,00
T-8	1,000000	1,000104	0,000104	0,01	0,00
T-9	1,000000	1,000063	0,000063	0,01	0,00
T-10	1,000000	0,999988	-0,000012	0,00	0,00
T-11	1,000000	1,000012	0,000012	0,00	0,00

Tabela 6.25b: Resultados da reconciliação das variáveis do tipo 2.

Correntes (tipo 2)	Fluxo Real	Fluxo pelo Bal. Massa (Desvio Real (%))	Fluxo Determinado (Desvio Ajustado (%))
2	1,000000	0,984850 (-1,50)	0,984883 (-1,50)
3	2,000000	1,979800 (-1,01)	1,980377 (-0,98)
4	0,700000	0,641004 (-8,42)**	0,640650 (-8,48)**
8	0,900000	0,950711 (5,63)*	0,950488 (5,62)*
9	0,150000	0,162477 (8,32)**	0,162404 (8,26)**
10	0,750000	0,784103 (4,54)	0,784101 (4,54)
11	0,450000	0,477362 (6,08)*	0,477512 (6,11)*

Tabela 6.25c: Fluxos determinados das variáveis do tipo 2.

As tabelas 6.24 e 6.25 mostram o efeito do mal condicionamento das matrizes de balanço. No último CASO, também por se tratar de medidas sem erros de nenhuma natureza, os valores dos desvios representados por Er e pelos desvios ajustados podem

ser considerados os valores dos desvios absolutos em relação aos valores reais. Aqui, fica mais uma vez evidenciada a conseqüência de se tomar medidas em uma planta em lugares errados. Comparando-se o sistema representado pelas tabelas 6.24 e 6.25, pode-se notar através da quantidade e da qualidade dos desvios, que o sistema representado pelo CASO 3a está menos mal condicionado que o representado pelo CASO 3. Os desvios médios no último caso são menores.

Após a apresentação do estudo de três sistemas-exemplos diferentes, pode-se dizer que em todos eles, onde foram obedecidas as restrições do método de reconciliação de dados através da projeção matricial, os resultados apresentaram desvios médios dos valores reconciliados e determinados menores que os dos valores medidos na planta diretamente. Está demonstrado assim que o programa aqui desenvolvido para reconciliação de dados funciona em perfeita sintonia com o previsto na literatura citada, e que o objetivo presente, qual seja, o de diminuir o erro médio entre as variáveis medidas foi alcançado.

6.2 RESULTADOS DA DETECÇÃO DE ERROS GROSSEIROS

A detecção de erros grosseiros faz parte, conforme explicado anteriormente, de um procedimento mais amplo que é a retificação de dados de processo. Na verdade, o procedimento de detecção, conforme descrito no capítulo 1 (figura 1.1), deve ser feito antes da reconciliação, contudo isto implicaria a utilização de outros métodos de cálculo que não a projeção matricial. Não é este o objetivo deste trabalho de modo que o trabalho aqui será limitado à detecção dos erros grosseiros que possam estar presentes num dado conjunto de medidas previamente reconciliadas, conforme o procedimento desenvolvido nos capítulos 3 e 4, deste trabalho.

Não será aqui feita nenhuma comparação entre os diferentes modos de se detectar erros grosseiros pois isto seria objeto de estudo de uma outra dissertação em virtude da vastidão de maneiras existentes na literatura. Para uma melhor compreensão e obtenção de maiores detalhes, podem ser consultadas as referências citadas no capítulo 2 deste trabalho.

Conforme descrito na literatura, nem sempre é possível se ter certeza de onde está o erro pois a presença do mesmo pode comprometer, como visto no ítem 6.1, os valores das variáveis reconciliadas e isto se refletirá na estatística de medida dos resultados, podendo assim se ter detecções não verdadeiras de erros grosseiros. MENDES(1995), faz o estudo para três exemplos com restrições lineares comprovando o exposto acima. No exemplo descrito por CROWE(1986) existe caso de variáveis com menos de 2% de desvio de estatística de medida onde o procedimento detectou (erradamente) a presença de erro grosseiro.

O exemplo utilizado é, conforme dito anteriormente, o descrito pelo EXEMPLO 1 no ítem 6.1. Este foi escolhido por não ter apresentado problemas de condicionamento de matrizes e por estar detalhado na literatura utilizada.

Os resultados não são numericamente coincidentes com os apresentados pela literatura uma vez que devido às dimensões muito pequenas dos valores os quais precisavam ser invertidos tem-se grande variação de dimensões dos mesmos conforme a aproximação utilizada. Isto não compromete de modo algum o trabalho pois a detecção dos erros grosseiros, quando eles existiam, sempre foi conseguida, e isto é provado pelo fato de que se conheciam os erros e suas localizações previamente, estando em concordância com os apresentados nas descrições do capítulo 2. No trabalho de CROWE(1986), está detalhada esta complicação, a qual encontra-se citada no ítem 5.4 deste trabalho (a respeito do fator complicador que é a relação de Z com d). A rigor, seria necessário um estudo à parte dos erros introduzidos em cada variável conforme a aproximação utilizada. Não foi feito aqui tal estudo uma vez que não é o valor numérico que interessa neste ponto do trabalho. Busca-se mais uma análise qualitativa dos resultados que quantitativa e serão estes os resultados que deverão ser considerados.

Os testes estatísticos utilizados serão o teste global (TEG), representado pelo cálculo dos valores chi-quadrados e o teste estatístico representado por Z (TEZ) (um para as variáveis do tipo 1, um para as variáveis do tipo 2 e um para os desbalanços). O capítulo 5 traz detalhadas as equações que representam cada teste. Especificamente tem-se, a equação (5.41), para o TEG, e as equações (5.43), (5.44) e (5.46) para o TEZ. O valor apresentado entre parênteses diante do TEG é referente ao posto da matriz $\mathbf{Y}$ e é chamado de crítico. Os testes estatísticos Z são feitos para um intervalo de confiança de 95% e os valores teóricos que representam os limites deste intervalo são também chamados de críticos, o modo de determinação deste fator está detalhado no capítulo 5.

Valores dos testes estatísticos das medidas superiores aos valores críticos são evidência de erros grosseiros.

Os CASOS estudados são os do EXEMPLO 1 do ítem 6.1.

Símbolos utilizados

TEG: Teste Estatístico Global ,

CHI2: Chi-quadrado, representa o teste estatístico global,

Z-a: Z referente às variáveis do tipo 1,

Z-d: Z referente às variáveis do tipo 2,

Z-e: Z referente aos desbalanços ou erros dos balanços mássicos, não se referem às correntes em particular.

Zc: Z crítico (será utilizado o mesmo símbolo para cada um dos Z's).

Os valores que apresentarem evidência de erros grosseiros serão assinalados com um asterisco.

Os valores utilizados para os testes estatísticos aparecem com quatro casas decimais na literatura e serão aqui seguidas estas convenções.

Os mesmos títulos utilizados quando da análise do EXEMPLO 1 serão aqui empregadas para se facilitar a associação CASO a CASO.

CASO 1: Variâncias iguais (sem erros grosseiros)

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	0,0000
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	0,0000	<	:	0,0000
B-1	0,0000	<	:	0,0000
T-1	0,0000	<	:	0,0000
A-2	>	0,0000	:	0,0000
B-2	>	0,0000	:	0,0000
T-2	>	0,0000	:	0,0000
A-3	>	0,0000	:	0,0000
B-3	>	0,0000	:	0,0000
T-3	>	0,0000	:	0,0000
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.26: Resultados da detecção de erros grosseiros do CASO 1 do EXEMPLO 1.

Nenhum dos quatro testes estatísticos acusou a presença de erros grosseiros.

COMENTÁRIO: O teste está correto pois se trata de um CASO onde todas as variáveis apresentam valores corretos — CROWE(1986) — livres portanto de erros grosseiros.

CASO 2: Variâncias proporcionais

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	15,0499*
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	2055,9192*	<	:	0,0000
B-1	2844,5000*	<	:	0,0000
T-1	2365,8560*	<	:	0,0000
A-2	>	0,0000	:	0,0358
B-2	>	0,0000	:	0,0291
T-2	>	0,0000	:	0,0000
A-3	>	0,0000	:	0,0748
B-3	>	0,0000	:	0,0006
T-3	>	0,0000	:	0,0032
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.27: Resultados da detecção de erros grosseiros do CASO 2 do EXEMPLO 1.

O TEG e o Z-a indicaram a presença de erros grosseiros entre os elementos do tipo 1 mas não indicaram a presença de um erro "menos" grosseiro (em relação ao primeiro) nos elementos das concentrações medidas.

COMENTÁRIO: O teste funcionou pois realmente existe uma variável com erro grosseiro entre as do tipo 1. O maior valor de Z-a refere-se justamente a este valor. O fato de o modelo não ter detectado a presença do erro na variável de tipo 2 pode ser explicado pelo fato de os valores do erro do tipo 1 influenciar mais na estatística de erro que os do tipo 2, conforme já descrito nos comentários dos CASOS 10 e 11.

CASO 3: Variâncias proporcionais

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	0,7678
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	60,4861*	<	:	0,0000
B-1	1,4998	<	:	0,0509
T-1	9,9745*	<	:	0,0000
A-2	>	0,0000	:	0,0689
B-2	>	0,0000	:	0,0369
T-2	>	0,0022	:	0,4229
A-3	>	0,0000	:	0,0000
B-3	>	6,1527	:	0,0200
T-3	>	0,0000	:	0,0160
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.28: Resultados da detecção de erros grosseiros do CASO 3 do EXEMPLO 1.

O TEG não indicou a presença de erro grosseiro . Os TEZ's indicaram a presença de dois erros nas variáveis do tipo 1 e um nas do tipo 2.

COMENTÁRIO: O teste funcionou pois realmente existe um erro grosseiro nas variáveis do tipo 2. O fato de o TEG não ter indicado a presença de erro e o "baixo" valor de Z-a (proporcionalmente aos valores apresentados), indicam que o teste está correto

CASO 4: Variâncias proporcionais.

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	0,0183
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	0,0427	<	:	0,0000
B-1	0,0040	<	:	0,0023
T-1	0,0224	<	:	0,0000
A-2	>	0,0000	:	0,0015
B-2	>	0,0000	:	0,0035
T-2	>	0,0000	:	0,1098
A-3	>	0,0000	:	0,0002
B-3	>	0,0000	:	0,0026
T-3	>	0,0000	:	0,0013
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.29: Resultados da detecção de erros grosseiros do CASO 4 do EXEMPLO 1.

Os quatro testes estatísticos não detectaram a presença de nenhum erro grosseiro.

COMENTÁRIO: o teste está correto pois não existe realmente nenhum erro grosseiro entre as variáveis (dos dois tipos). Existem apenas erros randômicos que deixaram os valores de TEG e TEZ abaixo dos valores críticos.

CASO 5: Variâncias proporcionais (valor base multiplicado por 10).

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	0,0000
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	0,0000	<	:	0,0000
B-1	0,0000	<	:	0,0000
T-1	0,0000	<	:	0,0000
A-2	>	0,0000	:	0,0000
B-2	>	0,0000	:	0,0000
T-2	>	0,0000	:	0,0000
A-3	>	0,0000	:	0,0000
B-3	>	0,0000	:	0,0000
T-3	>	0,0000	:	0,0000
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.30: Resultados da detecção de erros grosseiros do CASO 5 do EXEMPLO 1.

Nenhum dos quatro testes detectou a presença de erros grosseiros entre as variáveis.

COMENTÁRIO: Este CASO é referente ao CASO 4 com os valores das variância multiplicado por 10. Realmente não existiam erros grosseiros e o teste funcionou corretamente. Notar, somente, que com o aumento das variância das correntes totais, os valores dos testes estatísticos diminuíram.

CASO 6: Variâncias proporcionais (valor base multiplicado por 100).

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	0,0000
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	0,0000	<	:	0,0000
B-1	0,0000	<	:	0,0000
T-1	0,0000	<	:	0,0000
A-2	>	0,0000	:	0,0000
B-2	>	0,0000	:	0,0000
T-2	>	0,0000	:	0,0000
A-3	>	0,0000	:	0,0000
B-3	>	0,0000	:	0,0000
T-3	>	0,0000	:	0,0000
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.31: Resultados da detecção de erros grosseiros do CASO 6 do EXEMPLO 1.

Não foi detectada a presença de nenhum erro grosseiro entre as medidas.

COMENTÁRIO: O procedimento está correto pois realmente não existem erros desta natureza entre as variáveis. O fato de com o aumento da variância diminuir os valores dos testes estatísticos também foi observado neste CASO, confirmando o CASO anterior.

CASO 7: Variâncias proporcionais (valor base multiplicado por 0,1).

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	0,0000
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	0,0000	<	:	0,0000
B-1	0,0000	<	:	0,0000
T-1	0,0000	<	:	0,0000
A-2	>	0,0000	:	0,0000
B-2	>	0,0000	:	0,0000
T-2	>	0,0000	:	0,0000
A-3	>	0,0000	:	0,0000
B-3	>	0,0000	:	0,0000
T-3	>	0,0000	:	0,0000
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.32: Resultados da detecção de erros grosseiros do CASO 7 do EXEMPLO 1.

Nenhum teste estatístico indicou a presença de erros grosseiros entre as variáveis.

COMENTÁRIO: Este é o último CASO referente ao estudo da influência da variância nos resultados da reconciliação. O teste funcionou pois também aqui, não existiam erros grosseiros para serem detectados. Notar que o acontecido quando se aumentou a variância não se verificou, aqui, em sentido inverso, quando se diminuiu a mesma.

CASO 8: Variâncias proporcionais (com erros grosseiros)

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	1,0372
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	22,6992*	<	:	0,0000
B-1	132,8653*	<	:	0,0833
T-1	109,0633*	<	:	0,0000
A-2	>	0,0000	:	0,0613
B-2	>	0,0000	:	0,0390
T-2	>	0,0047	:	0,3045
A-3	>	0,0000	:	0,0019
B-3	>	0,0000	:	0,0207
T-3	>	11,6398*	:	0,0166
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.33: Resultados da detecção de erros grosseiros do CASO 8 do EXEMPLO 1.

Dois entre os testes estatísticos detectaram a presença de erros grosseiros. O TEG não detectou a presença de nenhum erro.

COMENTÁRIO: Realmente existem duas medidas com erros grosseiros entre as variáveis do tipo 2 e os testes TZE's indicaram corretamente a existência dos mesmos. O fato de o erro não estar nas variáveis do tipo 1 pode ser a causa da falha no TEG.

CASO 9: Variáveis proporcionais

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	0,7101
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	62,4402*	<	:	0,0000
B-1	29,2690*	<	:	0,0535
T-1	36,7775*	<	:	0,0000
A-2	>	0,0000	:	0,0766
B-2	>	0,0000	:	0,0411
T-2	>	0,0015	:	0,4555
A-3	>	0,0000	:	0,0017
B-3	>	0,0000	:	0,0217
T-3	>	4,6292*	:	0,0196
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.34: Resultados da detecção de erros grosseiros do CASO 9 do EXEMPLO 1.

O TEG não acusou a presença de erros grosseiros, mas os TEZ's indicaram a presença de erros entre as variáveis do tipo 1 e do tipo 2.

COMENTÁRIO: A exemplo do ocorrido com os valores reconciliados, aqui, a retirada do erro mais grosseiro, diminuiu os valores das estatísticas das medidas, apesar de as mesmas ainda continuarem a indicar, corretamente, a presença de erros grosseiros entre as medidas do tipo 2.

CASO 10: Variâncias proporcionais

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	15,5902*
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	1319.2618*	<	:	0,0000
B-1	2023.1194*	<	:	0,5031
T-1	1586,0060*	<	:	0,0000
A-2	>	0,0000	:	0,0009
B-2	>	0,0000	:	0,0060
T-2	>	0,0000	:	0,1364
A-3	>	0,0000	:	0,0000
B-3	>	0,0000	:	0,0006
T-3	>	0,0000	:	0,0045
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.35: Resultados da detecção de erros grosseiros dos CASOS 10 e 11 do EXEMPLO 1.

O TEG e o TEZ detectou a presença de erros grosseiros entre as variáveis do tipo 1. Não foi identificada nenhuma variável do tipo 2 com erro grosseiro.

COMENTÁRIO: Existe realmente, um erro bastante grosseiro entre as variáveis do tipo 1, mas existe também um entre as do tipo 2, não identificado. A grande magnitude do erro nas variáveis do tipo 1 possivelmente mascarou a estatística de medida referente ao erro na variável do tipo 2.
A mesma análise feita para o CASO 10 serve também para o CASO 11 e não será repetida.

CASO 12: Variâncias proporcionais (com erros grosseiros)

TESTE ESTATÍSTICO GLOBAL	
CHI2 (calculado)	15,0499*
CHI2 (teórico)	7,8150

CC	Z-a Zc=2,3900	Z-d Zc=2,9700		Z-e Zc=2,7600
A-1	1805,9998*	<	:	0,0000
B-1	2328,4988*	<	:	0,4289
T-1	2021,8341*	<	:	0,0000
A-2	>	0,0000	:	0,1080
B-2	>	0,0000	:	0,0559
T-2	>	0,0083	:	0,6698
A-3	>	0,0000	:	0,0000
B-3	>	0,0000	:	0,0273
T-3	>	4,9550*	:	0,0206
A-4	>	0,0000	:	<>
B-4	>	0,0000	:	<>
T-4	>	0,0000	:	<>
A-5	>	0,0000	:	<>
B-5	>	0,0000	:	<>
T-5	>	0,0000	:	<>
A-6	>	0,0000	:	<>
B-6	>	0,0000	:	<>
T-6	>	0,0000	:	<>
A-7	>	0,0000	:	<>
B-7	>	0,0000	:	<>
T-7	>	0,0000	:	<>

Tabela 6.36: Resultados da detecção de erros grosseiros dos CASOS 12, 13 e 14 do EXEMPLO 1

Conforme análise feita sobre o CASO 10(na tabela 6.12), vale aqui, a mesma observação. A retirada dos erros das variáveis do tipo 2 variou pouco as respostas conseguidas no CASO 10. O TEG e o TEZ continua detectando 1 erro grosseiro nas variáveis do tipo 2.

COMENTÁRIO: A presença de erro claramente grosseiro (23,03%) nas variáveis do tipo 1 está causando influências em todos os testes estatísticos. O maior valor do teste estatístico corresponde à variável errada.

CASOS 15 a 19: Variâncias proporcionais.

TESTE ESTATÍSTICO GLOBAL					
CASO / Desvio	15/(5%)	16(10%)	17(15%)	18(20%)	19(30%)
CHI2 (calculado)	1,2989	2,7240	5,5950	10,6307*	30,7351*
CHI2 (crítico)	7,8150	7,8150	7,815	7,8150	7,8150

CC	Z-crítico	Z-a (5%)	Z-a (10%)	Z-a (15%)	Z-a (20%)	Z-a (30%)
A-1	2,3900	475,5552*	935,9598*	1370,1154*	1750,1407*	2855,6274*
B-1	2,3900	574.9536*	1231,2870*	1854,6055*	2402,4790*	4017,8269*
T-1	2,3900	493,2854*	1036,0470*	1548,8179*	2004,4351*	3318,6606*

Tabela 6.37: Resultados da detecção de erros grosseiros dos CASOS 15 a 19 do EXEMPLO 1.

Os CASOS de 15 a 19 são referentes àqueles em que se variou a intensidade dos erros grosseiros entre as variáveis do tipo 1 fazendo-os passar de erro zero até 30%, mantendo-se as variáveis do tipo 2 com apenas um pequeno erro grosseiro (8,28%) entre as suas medidas.

Após a execução dos cinco CASOS, onde se aumentava os erros nas variáveis do tipo 1, nota-se que em todos eles, o TEG e o TEZ detectaram a presença de três erros grosseiros. Os valores de TZE's para as varáveis do tipo 2 e para os desbalanços não detectaram a presença dos erros grosseiros.

COMENTÁRIO: Realmente, conforme mostrado, o conjunto de medidas continha erros cada vez mais grosseiros à medida que se avançava nas simulações. E este erro estava sempre localizado nas segundas variáveis do primeiro tipo de cada CASO, o que foi corretamente localizado.

O CASO 20 não será aqui mostrado uma vez que se trata de mera repetição de tantos outros aqui mostrados que não apresentavam erros grosseiros. O procedimento funcionou bem neste CASO não detectando erros não presentes entre as variáveis medidas.

Em todos os CASOS de detecção de erros grosseiros estudados, pode-se notar que exceto nos de números 15 e 16, todos os demais só tiveram os erros acusados pelo TEG quando as medidas corrompidas com erros eram as do tipo 1. Parece que neste procedimento existe uma tendência já antes verificada quando da reconciliação dos dados, em se ter uma maior influência dos valores medidos das variáveis do tipo 1. Esta maior influência possivelmente se deve ao fato de que tais variáveis estão "amarradas" por duas medidas conhecidas que são a concentração e os fluxos totais. No caso das variáveis do tipo 2, somente as concentrações são conhecidas. Deste modo, pode-se resumir as tendências aqui verificadas pela supremacia de influências das variáveis do tipo 1(um) em relação às do tipo 2, tanto quando se faz a reconciliação dos dados como quando se faz a detecção dos erros grosseiros. Comprova-se também o previsto na

literatura de que a utilização de mais que um teste estatístico se faz necessária para garantir a correta detecção de possíveis erros grosseiros.

7.CONCLUSÕES E SUGESTÕES

7.1 CONCLUSÕES

O estudo da influência da variação das medidas em variáveis de processo é um aspecto muito importante dentro do controle de sistemas reais uma vez que o perfeito funcionamento e controle de um processo depende de medidas feitas dentro de procedimentos confiáveis. A utilização da reconciliação/retificação de dados se mostra, assim, de grande valia uma vez que em muitas situações, pequenas distorções e até ruídos nos instrumentos pode causar correções indevidas nas variáveis controladas. Por outro lado, existem situações em que a margem de previsibilidade dos erros extrapola qualquer mecanismo estatístico que seja capaz de "prever" os valores verdadeiros ou próximos aos verdadeiros que a variável deveria apresentar, em condições normais. Está-se aqui falando das medidas que apresentem erros grosseiros, ou erros não randômicos. Existe uma grande variedade de erros que podem ser classificados nesta segunda categoria, como por exemplo o emperramento de válvulas devido a problemas mecânicos ou elétricos, o vazamento em linhas de transmissão de fluidos, o vazamento em vasos ou tanques de estocagem, falhas em atuadores, etc. Para os erros causados devido a esta segunda classe de falhas, não é possível , na maioria das vezes, se ter um mecanismo de compensação, estatístico ou não, que corrija os valores das variáveis de modo a trazer o processo às condições normais de funcionamento. Em nenhum dos exemplos, aqui estudados, pode-se conseguir uma boa estimativa de valores onde se tinha a presença de erros grosseiros, apesar de na literatura existirem indicações de que em alguns sistemas, com erros grosseiros, a estimativa ficar melhor que os valores medidos diretamente na planta. Muitas vezes, não é possível levar a termo o mecanismo de detecção de erros grosseiros uma vez que o aqui proposto pressupõe descartar medidas que porventura apresentem erros grosseiros, e isto nem sempre permite manter o sistema com um conjunto mínimo de medidas de tal forma que ainda continue determinável. Isto faz com que se entenda a importância de associação do mecanismo de reconciliação de dados aos demais mecanismos descritos neste trabalho, de tal modo que se possa garantir que não existam erros grosseiros e que as medidas estejam devidamente classificadas. A fim de sintetizar este problema, desenvolveu-se a retificação de dados que é na verdade um procedimento que, mais geral, engloba a reconciliação de dados em sua última fase. Existem mecanismos de detecção de erros grosseiros, como o utilizado neste trabalho que detectam a presença ou não de erros grosseiros após ter sido feita a reconciliação dos dados, mas também existem aqueles que detectam a presença do erro antes que seja feita a reconciliação.

À primeira vista, pode parecer que um tenha muita vantagem em relação ao outro, contudo, quando se pensa que o funcionamento de ambos deveria ser em tempo real, a saída do processo das condições normais de funcionamento, causa desvios de previsão em ambos os modelos. A detecção da presença de um erro grosseiro antes de se processar a reconciliação de dados não é garantia de que se irá conseguir amenizar o problema, pois os mecanismos de que se dispõe levam em conta a possibilidade de descarte da variável com problemas. Entretanto, conforme visto antes, esta variável pode ser essencial à reconciliação de dados e o fato de descartá-la torna o sistema de equações impossível de ser resolvido. Mais importante ainda, o fato de se descartar uma variável defeituosa não garante que cessem seus efeitos, veja por exemplo os casos citados no primeiro parágrafo.

Inerentemente ao problema de condicionabilidade da matriz das restrições (**B**), pode-se dizer sem sombra de dúvidas que a distribuição dos instrumentos dentro da planta é um fator de extrema importância, uma vez que é esta distribuição em última análise que vai permitir ou não a realização do procedimento de reconciliação de dados.

Finalmente, pode-se dizer que o mecanismo de reconciliação de dados é um instrumento útil de otimização de processos, e quando utilizado combinado aos demais procedimentos de controle de processos se mostra como um poderoso instrumento para um efetivo monitoramento de plantas a nível industrial. O tempo de execução dos programas é mínimo, questão de segundos, portanto o procedimento pode tranquilamente ser utilizado em tempo real dentro do processo.

7.2 SUGESTÕES

Baseado nas conclusões apresentadas no ítem anterior, onde a presença de erros grosseiros inviabiliza o procedimento de reconciliação de dados, sugerimos que seja tentado um mecanismo que consiga detectar falhas e diagnosticar sua causas, fazendo com que o sistema esteja somente sujeito a erros de flutuação randômica quando em condições normais de funcionamento. A reconciliação de dados pode assim fazer parte de um "pacote" computacional mais amplo que englobe a localização de falhas nos diversos componentes (instrumentos e equipamentos) do processo, quando estas se fizerem presentes e que apenas reconcilie quando os valores medidos estiverem com flutuações randômicas.

8.BIBILIOGRAFIA

ALDRICH, C.; VAN DEVENTER, J. S. J., (1994), **"Identification of Gross Errors in Material Balance Measurements by means of Neural Nets"**, Chemical Engineering Science, v49, n9, pp1357-1368.

ALMASY, G. A.; SZTANO, T.; CSILLAG, G.; VERESS, G.; HOLDERITH, J., (1969), **"Complex Material and Heat Balances as a Tool of Computer control in Continuous Process"**, 3rd CHISA Congress, Paper D3.10, Marianske Lazne, Czechoslovakia.

ALMASY, G.; SZTANO, P., (1975), **"Checking and Correction of Measurements on the Basis of Linear System Model"** , Problems of Control and Information Theory, v4, pp57-64.

ALMASY, G. A., (1990), **"Principles of Dynamic Balancing"**, AIChe. Journal, v36, pp1321-1330.

BIEGLER, L. T.; HUGHES, R. R., (1983), **"Process Optimization: A Comparative Case Study"**, Computers and Chemcal Engineering, v7, n5, pp645-661.

BOX, M. J., (1970), **"Improved Parameter Estimation", Technometrics"**, v12, pp219-240.

BOX, G. E. P.; HUNTER, W. G.; MacGREGOR, J. F.; ERJAVEC, (1973), "Some Problems with the Analysis of Multiresponse Data" ,Technometrics, 15, pp33-39.

BOSSEN, B. S.; CHRISTIANSEN, L. J.,; JARVAN, J. E., (1994), **"Simulation, Optmization and Data-Reconciliation of Industrial Chemical Process"**, Trans. IChemE, v72, Part A, pp376-381.

BRITT, H. I.; LUCKE, R. H., (1973), **"The Estimation of Parameters in Nonlinear Implicit Models"**, Technometrics, v15, pp233-247.

BROYDEN, C. G., (1965), **"A Class of Methods for Solving Nonlinear Simultaneous Equations"**, Math. Comp., v19, pp577-581.

BUSSANI, G.; CHIARI, M.; GROTTOLI, M. G.; PIERUCCI, S.; FAVARELLI, T.; RICCI, G.; GIOVENTÚ, G., (1995), **"Application of Data-Reconciliation and Optimization Procedure Hidrogen-Plant"**, Computers and Chemical Engineering, Suppl, v19, pp-S299-S304.

CHRISTIANSEN, L. J., (1992), **"A Computer Aided Engineering System for Development of Catalytic Process"**, Computers and Chemical Engineering, Suppl v16, ppS55-S68.

CROWE, C. M .; GARCIA CAMPOS, Y. A.; HRYMAK, A., (1983), **"Reconciliation of Process Flow Rates by Matrix Projection. Part I: Linear Case"** , AIChe. Journal, v29, n12, pp881-888.

CROWE, C. M. , (1986), **" Reconciliation of Processs Flow Rates by Matrix Projection. Part II : the Nonlinear Case."** , AIChe. Jorunal, v32, n4, pp616-623.

CROWE, C. M., (1988), **"Recursive Identification of Gross Errors in Linear Data Reconciliation"**, AIChe. Journal, v34, n4, pp541-549.

CROWE, C. M., (1989a), **"Test of Maximum Power for Detection of Gross Errors in Process Constraints"** , AIChe. Journal, v35, n5, pp869-872.

CROWE, C. M., (1989b), **"Observability an Redundancy of Process Data for Steady State Reconciliation"**, Chem. Eng. Sci., v44, n12, pp2909-2917.

CROWE, C. M., (1992), **"The Maximum-Power Test for Gross Errors in the Original Constraints in Data Reconciliation"**, The Canadian Journal of Chemical Engineer, v70, pp1030-1036.

DANTZIG, G., (1963), **"Linear Programming and Extensions"**, Princeton University Press, NJ.

DAROUACH, M.; RAGOT, J. FAYOLLE, J.; MAQUIN, D., (1986), **"Calcul Hierachise et Decomposition des Systems Lineires et Bilineaires aplique a la Violation des Donnees", RAIRO** Automat. Product. Informat. Indust., v20, pp405-432.

DAROUACH, M.; ZASADZINSKI, M., (1991), **"Data Reconciliation in Generalized Linear Dynamic Systems"**, AIChe Journal, v37, n2, pp193-201.

DENNIS, J. E., Jr., SCHANABEL, R. B., (1983), **"Numerical Methods for Unconstrained Optmization and Non Linear Equations"**, Prentice-Hall Englewoods Cliffs, NJ.

FARISS, R. H.; LAW, V. J., (1979), **"An Efficient Computacional Technique for Generalized Application of Maximum Likelihood to Improve Correlation of Experimental Data"**, Computers and Chemical Engineering, v3, pp95-115.

FLECHTER, R., (1987), **"Practical Methods of Optimization"**, John Wiley and Sons, 2nd edition, 436p.

FREUND, J. E., (1988), **"Modern Elementary Statistics"** , Prentice Hall, New Jersey, 590pp.

GILL, P. E.; MURRAY, W.; WRIGHT, M. H., (1981), **"Numerical Linear Algebra and Optmization"**, Addison-Wesley Publishing Company, 426p.

HEYEN, G.; MARÉCHAL, E.; KALITVENTZEFF, B., (1996), **"Sensitivity Calculations and Variance Analysis in Plant Measurement Reconciliation"**, Computers and Chemical Engineering, Suppl, v20, ppS539-S544.

HIMMELBLAU, D. M., (1978), **"Fault Detection and Diagnosis in Chemical and Petrochemical Processes"**. Amsterdam: Elsevier Scientific Publishing Company, 421p.

HIMMELBLAU, D.M.; EDGAR, T. F., (1988), **"Optimization of Chemical Process"**, McGraw Hill, USA.

HLAVACEK, V., (1977), **"Analysis of a Complex Plant-Steady State and Transient Behavior"**, Computers and Chemical Engineering, v1, pp75-100.

HODOUIN, D.; ALLIOT, N.; FLAMENT, F., (1991), **"Redundancy Analysis of Complex Sets of mineral Processing Data for Mass Balance Computation"**, Int. Journal of Mineral Processing, v32, pp213-231.

KALITVENTZEFF, B.; JORIS, B, P., (1987), **"Process Measurements Analysis and Validation"**, CEF '87, pp41-46.

KARJALA, T. W.; HIMMELBLAU, D. M., (1994), **"Dynamic Data Retification by Recurrent Neural networks vs Traditional Methods"**, AIChe. Journal, v40, n11, pp1865-1874.

KELLER, J. Y.; ZASADZINSKI, M.; DAROUACH, M., (1991), **"Analytical Estimator of Measurement Error Variances in Data Reconciliation"**, Computers and Chemical Engineering, v16, n2, pp185-188.

KIM, I. W.; LIEBMAN, M. J.; EDGAR, T. F., (1991), **"A Sequential Error-in Variables Method for Nonlinear Dynamic Systems"**, Computers and chemical Engineering, v15, n9, pp663-670.

KNEPPER, J. C.; GORMAN J. W., (1980), **"Statistical Analysis of Constraints Data Sets"**, AIChe. Journal, v26, pp260.

KRETSOVALIS, A.; MAH, R. S. H., (1987), "**Observation and Redundancy Classifiation in Multicomponents Process Networks**", AIChe J., v33, n1, pp 70-82.

KRETSOLVALIS, A.; MAH, R. S. H., (1988a), "**Observability and Redundancy Classification in Generalized Process Networks -I: Theorems**", Computers and Chemical Engineering, v12, n7, pp661-687.

KRETSOVALIS, A.; MAH, R. S. H., (1988b), "**Observability and Redundancy Classification in Generalized Process Networks -II: Algorithms**", Computers and Chemical Engineering, v12, n7, pp669-703.

KUEHN, D. R.; DAVIDSON, H., (1961), **"Computer Control -II. Mathematics of Control"** , Chem. Eng. Progress, v57, n6, pp44-47.

LAURENCE, P. J., (1989), **"Data Reconciliation : Getting Better Information"**, Hidrocarbon Processing.

LIEBMAN, M.J.; EDGAR, T. F., (1988), **"Data Reconciliation for Nonlinear Process"**, Paper Presented at the Annual Meeting, Washington, DC.

LIEBMAN, M. J.; EDGAR, T. F.; LASDON, L. S., (1992), **"Efficient Data Reconciliation and Estimation for Dynamic Process Using Nonlinear Programming Techniques"**, Computers and Chemical Engineering, v16, n10/11, pp963-986.

MADRON, F.; VEVERKA, V.; VANECEK, V., (1977), **"Statistical Analysis of Material Balance of a Chemical Reactor"**, AIChe Journal, v23, n4, pp482-486.

MAH, R. S.; STANLEY, G. M.; DOWNING, D. M., (1976), **"Reconciliation and Retification of Process Flow and Inventory Data"** , Ind. Eng. Chem. Proc. Des. Dev., v15, n1, pp175-183.

MAH, R. S.H., (1987), **"Chemical Process Structures and Information Flows"**, Butterworts Series in Chemical Engineering, 500pp.

MAH, R.S.; TAMHANE, A. C., (1982), **"Detection of Gross Errors in Process Data"**, AIChe. Journal, v28, n5, pp828-830.

MENDES, T. F., (1995), Tese de Doutorado, **"Reconciliação e Retificação de Dados e Classificação de Variáveis de Processo"** , FEQ-UNICAMP.

MEYER, M.; KOEHRET, B.; ENJALBERT, M., (1993), **"Data Reconciliation on Multicomponent Network Process"**, Computers and Chemical Engineering, v17, n8, pp807-817.

MILLER, J.; FREUND, J. E., JOHNSON, R. A., (1990), **"Probability and Statistics for Engineers"** , Englewood Cliffs: Prentice Hall, 624p.

NARASIMHAN, S.; HARIKUMAR, P., (1993a), **"A Method to Incorporate Bounds in Data Reconciliation and Gross Error Detection-I . The Bounded Data Reconcilition Problem"**, Computers and Chemical Engineering, v17, n11, pp-1115-1120.

NARASIMHAN, S.; HARIKUMAR, P., (1993b), **"A Method to Incorporate Bounds in Data Reconciliation and Gross Error Detection-II . Gross Error Detection Strategies"**, Computers and Chemical Engineering, v17, n11, pp-1121-1128.

NARASIMHAN, S.; MAH, R. S. H., (1987), **"Generalized Likelihood Ratio Method for Gross Error Identification"**, AIChe Journal, v33, n9, pp1514-1521.

NARASIMHAN, S.; MAH, R. S. H., (1989), **"Treatment of General Steady State Process Models in Gross Error Identification"**, Computers and Chemical Engineering, v13, n7, pp851-853.

PAPPAS, C. H.; MURRAY, W. H., (1991), "Turbo C ++ _ Completo e Total" , McGaw-Hill, Makron Books, São Paulo, 771pp.

PRESS, W. H.; TEUKOLKY, S. A.; VETTERING, W. T.; FLANNERY, B. P., (1992), **"Numerical Recipes in C"** , Cambridge University, 2nd edition..

OTHMER, D. F.; MCKETTA, J. J.,(1966), **"Encyclopedia of Chemical Technology"** , 2nd Edition, v10, John Willey and Sons.

PAI, C. C. D.; FISHER, G. R., (1988), **"Aplication of Broyden's Method to Reconciliaation of Nonlinearly Constrained Data"** , AIChe. Journal, v34, n5, pp873-876.

RAMAMURTHI, Y.; SISTU, P. B.; BEQUETTE, B. W., (1993), **"Control-Relevant Dynamic Data Reconciliation and Parameter Estimation",** Computers and Chemical Engineering, v17, n1, pp41-59.

REILLY, P. S.; CARPANI, R. E., (1963), **"Application of Statistical Theory of Adjustment to Material Balances"** , 13th Can. Chem. Eng. Conf., Montreal, Que.

ROMAGNOLI, J. A., (1983), **"On Data Reconciliation: Constraints Processing and Treatment of Bias",** Chemical Eng.Science, v38, pp1107-1117.

ROMAGNOLI, J. A.; STEPHANOPOULOS, G., (1980), **"On the Retification of Measurements Errors for Complex Chemical Plants" ,** Chemical Eng. Science, v35, pp1067-1081.

ROSENBERG, J., MAH, R. S. H., IORDACHE, C., (1987), **"Evaluation of Schems for Detecting and Identifying Gross Errors in Process Data",** Ind. Eng. Chem. Res., v26, pp555-564.

SCHARAA, O. J.; CROWE, C. M., (1996), **"The Numerical Solution of Bilinear Data Reconciliation Problems Using Unconstrained Optimization Methods",** Computers and Chemical Engineering, Suppl, v20, ppS727-S732.

SCHILDT, H., (1988), **"Turbo C : Guia do Usuário"** , McGraw-Hill, São Paulo, 427pp.

SCHILDT, H., (1990), **"Turbo C Avançado",** McGraw-Hill, São Paulo, Guia do Usuário, 568pp.

SERTH, R. W.; HEENAN, W. A., (1986), **"Gross Error Detection and Data Reconciliation in Steam-Metering Systems",** AIChe. Journal, v32, n5, pp733-742.

SERTH, R. W.; VALERO, C. M.; HEENAN, W. A., (1987), **"Detection of Gross Errors in Nonlinearly Constrained Data : a Case Study"** , Chem. Eng. Comm., v51, p89-104.

SHELDON, M. R., (1986), **"Introduction to Probability and Statistics for Engineers and Scientists"**, 2nd Edition, Wiley Series in Probability and Mathematic Statistics.

SMITH, H. W.; ICHIYEN, N., (1973), **"Computer Adjustment of Metalurgical Balances"**, C.I.M. Bull., pp97-100.

STADTHERR, M. A.; GIFFORD W.A., SCRIVEN, L. E., (1973), **"Efficient Solution of Sparse Sets of Design Equations"**, Chemical Engineering Science, v29, pp1025-1034.

STANLEY, G. M.; MAH, R. S. H., (1976), **"Reconciliation and Retification of Process Flow an Inventory Data"**, Ind. Chem. Proc. Des. Dev., v15, pp175-183.

STANLEY, G. M.; MAH, R. S. H., (1977), **"Estimation of Flows and Temperatures in Process Networks"**, AIChe. Journal, v23, pp642-650.

STANLEY, G. M.; MAH, R. S. H, (1981a), **"Observability and Redundancy in Process Data-Estimation"**, Chem. Eng. Sci., v36, pp259-272.

STANLEY, G. M.; MAH, R. S. H., (1981b), **"Observability and Redundancy Classification in Process Network- Theorems and Algorithms"**,Chem. Eng. Sci., v36, n12, pp1941-1945.

STEPHENSON, G. R.; SHEWCHUK, C. F., (1986), **"Reconciliation of Process Data with Process Simulation"**, v32, pp247-252.

STEWART, G. W., (1977), **"On the Perturbation of pseudo-Inverse Projections and Linear Least-Square Problems"**, SIAM REVIEW, v19, n4, pp634-661.

TAMHANE, A. C., (1982), **"A Note on the Use of Residuals for Detecting an Outlier in Linear Regression"** , Biometrika, v69, pp488-493.

TJOA, I. B.; BIEGLER, L. T. (1991), **"Simultaneous Strategies for Data Reconciliation and Gross Error Detection of Nonlinear Systems"**, Computers and Chemical Engineering, v15, n10, pp679-690.

TONG, H.; CROWE, C. M., (1995), **"Detection of Gross Errors in Data Reconciliation by Principal Componensts Analysis"**, AIChe. Journal, v7, pp1712-1722.

TONG, H.; CROWE, C. M., (1996), **"Detecting Persistent Gross Errors by Sequential Analysis of Principal Components"**, Computers and Chemical Engineering, Suppl.,v20, ppS733-S738.

VACLAVEK, V.; KUBICEK, M.; LOUCKA, M., (1976a), **"Calculations of Material Balances in Chemical Engineeering Systems with Allowance for Errors in Measurement Errors"**, Theor. Found. Chem. Engineering, v9, pp242-245.

VACLAVEK, V.; KUBICEK, M.; LOUCKA, M., (1976a), **"Calculations of Material Balances in Chemical Engineeering Systems with Allowance for Errors in Measurement Classification of Stream Parameters"**, Theor. Found. Chem. Engineering, v10, pp256-260.

VACLAVEK, V.; KUBICEK, M.; LOUCKA, M., (1976b), **"Calculation of Material Balances for Chemical Engineering Systems with Due Allowance for Errors in Measurement- Classification of Stream Parameters"**, Theor. Found. Chem. Engineering, v10, pp256-260.

VACLAVEK, V.; LOUCKA, M., (1976), **"Selection of Measurements Necessary to Achieve Multicomponent Mass Balances in Chemical Plant"**, Chemical Eng. Science, v31, pp1199-1205.

VERDINELLI, I.; WASSERMAN L., (1990), **"Bayesian Analysis of Outlier Problems Using the Gibbs Sampler"**, Thecnical Report 469, Dept of Statistics, Carnegie Mellon University, Pittsburg, PA.

YAMANE, T., (1973), **"Statistics an Introduction Analysis"**, 3rd Edition, Haper International Edition.

WALD, A., (1947), **"Sequential Analysis"**, John Willey and Sons, Inc, New York.

WILLIAM, H. B., (1987), **"Handbook of Tables for Probability and Statistics"**, CRC Press, 2nd edition,

ZHANG, Z.; PIKE, R. W.; HERTWIG. T. A., (1995), **"An Approach to on-line Optimization of Chemical Plants"**, Computers and Chemical Engineering, Suppl, v19, ppS305-S310.

APÊNDICES

A.CONCEITOS ESTATÍSTICOS

B.CONCEITOS MATEMÁTICOS

C.LISTAGEM DOS PROGRAMAS- RECON E RETIF

D.LISTA DE FIGURAS

A- CONCEITOS ESTATÍSTICOS

A reconciliação e retificação de dados é dependente de conjuntos de dados medidos durante a operação do sistema. Estes dados, em conjunto, apresentam características de comportamento que podem ser traduzidas por conceitos estatísticos. Visando a dar uma noção dos ítens abordados neste trabalho, será aqui apresentada uma pequena descrição dos aspectos relevantes em estudo.

A1. CONCEITOS BÁSICOS

A estatística está interessada nos métodos científicos para coleta, organização, resumo, apresentação e análise de dados bem como na obtenção de conclusões válidas e na tomada de decisões razoáveis baseadas em tais análises.

Devido a grandes dificuldades do ponto de vista matemático, visto que um número limite real pode verdadeiramente não existir, a teoria moderna de estatística aceita de forma axiomática o conceito de probabilidade, sem defini-la conceitualmente. Alguns outros conceitos, entretanto, podem assim ser descritos:

\ **Experimento**: é qualquer situação onde possam se anotadas observações diferentes.

\ **População ou universo**: é um conjunto de observações possíveis relativas a determinado experimento.

\ **Amostra**: é uma quantidade representativa de uma população sobre a qual se podem tirar importantes conclusões, via inferenciamento, para a população global.

\ **Evento**: é uma das possíveis respostas do estudo das observações da amostra.

\ **Variável**: é um símbolo, que pode assumir qualquer um de um conjunto de valores que lhe são atribuídos, conjunto este chamado de domínio da variável. Se apenas um valor pode ser atribuído à variável, esta é dita ser constante.

\ **Freqüência de distribuição**: é o número de ocorrências de eventos em determinada classe.

\ **Dispersão ou variação**: é o grau segundo o qual os dados numéricos tendem a dispersar-se em torno de um valor médio. Pode ser representada em termos da amplitude total do desvio médio e do desvio padrão, basicamente. Existem outras formas, que não necessitam ser aqui consideradas.

\ **Amplitude total**: é a diferença entre o valor mais baixo e o mais alto, dentro de um conjunto de valores.

\ **Desvio médio**: é definido, para um conjunto de números, como o módulo do somatório das diferenças de cada valor em relação à média, dividido pelo tamanho do espaço amostral.

\ **Desvio padrão**: dado um conjunto de números X_1, X_2, ..., X_N, é representado por σ e é definido por

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}(X_j - \overline{X})^2}{N}} = \sqrt{\frac{\sum (X - \overline{X})^2}{N}} = \sqrt{\frac{\sum x^2}{N}} = \overline{(X - \overline{X})^2} \qquad \text{(A-1)}$$

em que x representa o desvio de cada um dos números x_j em relação à média $\overline{X}$.
Então, σ é a raiz quadrática média dos desvios, em relação à média ou, como é muitas vezes denominada, o desvio da raiz quadrática média.
Se X_1, X_2, ... X_K ocorrem com as freqüências f_1, f_2,..., f_K, respectivamente, o desvio padrão pode ser definido como:

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N} f_j(X_j - \overline{X})^2}{N}} = \sqrt{\frac{\sum f(X - \overline{X})^2}{N}} = \sqrt{\frac{\sum fx^2}{N}} = \overline{(X - \overline{X})^2} \qquad \text{(A-2)}$$

onde $N = \sum_{j=1}^{K} = \sum f$

as equações (A-1) e (A-2) podem ser escritas de forma equivalentes, como

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N} X_j^2}{N} - \frac{\left(\sum_{j=1}^{N} X_j\right)^2}{N}} = \sqrt{\frac{\sum X^2}{N} - \left(\frac{\sum X}{N}\right)^2} \qquad \text{(A1.1)}$$

e

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N} f_j X_j^2}{N} - \frac{\left(\sum_{j=1}^{N} f_j X_j\right)^2}{N}} = \sqrt{\frac{\sum fX^2}{N} - \left(\frac{\sum fX}{N}\right)^2} \qquad \text{(A2.1)}$$

O desvio padrão pode ser também definido por

$$\sigma = \sqrt{\frac{\sum (X_j - a)^2}{N}},$$

em que a é uma média próxima da aritmética. De todos esses desvios padrões, o mínimo é aquele o qual $a = \overline{X}$.

\ **Esperança**: um dos mais importantes conceitos usados em teoria da probabilidade é o da variável esperada. Se X é uma variável randômica assumindo um dos possíveis

valores x_1, x_2,..., então, a esperança ou valor esperado de X é denotado por E[X] , e é definido por

$$E[X] = \sum_{i=0}^{N} x_i P\{X = x_i\}$$
$$= \sum x_i p(x_i)$$

Em palavras, o valor esperado de X é a média dos possíveis valores que X pode assumir, onde cada valor é ponderado pelo peso das ocorrências de X.

E[X] = primeiro momento de X
$E[X^n]$ = n-ésimo momento de X, tem-se

$$E[X^n] = \begin{cases} \sum x^n p(x) \\ \int_{-\infty}^{\infty} x^n f(x) dx \end{cases}$$

caso x seja discreto ou contínuo, respectivamente.

\ **Variância**: É um conceito que mede a dispersão de X. Se X é uma variável randômica com média μ então a variância de X, denotada por Var(X) e por σ^2, é definida por

$$Var(X) = E\left[(X - \mu)^2\right]$$

em termos da esperança

ou de modo equivalente

$$Var(X) = E\left[(X - \mu)^2\right]$$
$$Var(X) = E\left[X^2 - 2\mu X + \mu^2\right]$$
$$Var(X) = E\left[X^2\right] - E[2\mu X] + E\left[\mu^2\right]$$
$$Var(X) = E\left[X^2\right] - \mu^2$$

de onde vem.

$$Var(X) = E\left[X^2\right] - (E[X])^2$$

ou em palavras, a variância de X é igual ao valor esperado do quadrado de X menos o quadrado do valor esperado.

De outra forma, a variância é simplesmente o desvio quadrado médio, de n valores medidos de x, em relação à média da amostra X.
Assim,

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x - \overline{x})^2}{n}$$

ou

$$\sigma^2 = \frac{\sum x^2 - \left[\left(\sum x \right)^2 / n \right]}{n}$$

onde a média (x) é obtida por

$$\overline{x} = \frac{\sum x}{n}$$

O valor da variância se torna mais confiável à medida que se aumenta o número de elementos da amostra e a exatidão do procedimento de medição é dado pela variância.

\ **Covariância**: é o conjunto que relaciona variáveis aleatórias, X e Y, definida num mesmo espaço amostral, representada por Cov (X,Y). Covariância entre duas variáveis só existe quando elas são dependentes uma da outra.
É definida por

$$\mathrm{Cov}(X,Y) = E\left[(X - \mu_x)(Y - \mu_y) \right]$$

ou

$$\mathrm{Cov}(X,Y) = E[XY] - \mu_x \mu_y$$

onde μ_x e μ_y são as médias de X e Y respectivamente.

Uma expressão útil para a Cov(X,Y) pode ser obtida expandindo-se o lado direito da definição. Tem-se assim

$$\mathrm{Cov}(X,Y) = E\left[XY - \mu_x Y - \mu_y X + \mu_x \mu_y \right]$$
$$\mathrm{Cov}(X,Y) = E[XY] - \mu_x E[Y] - \mu_y E[X] + \mu_x \mu_y$$
$$\mathrm{Cov}(X,Y) = E[XY] - \mu_x \mu_y - \mu_y \mu_x + \mu_x \mu_y$$
$$\mathrm{Cov}(X,Y) = E[XY] - E[X]E[Y]$$

da definição, tem-se

$$\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$$
$$\mathrm{Cov}(X,X) = \mathrm{Var}(X)$$

$$Cov(aX,Y) = aCov(X,Y)$$

\ **Valor verdadeiro de uma variável**: é aquele que seria obtido na medição se não houvesse característica aleatória associada à medição.

\ **Erro padrão**: é a denominação dada ao desvio padrão da distribuição amostral de uma grandeza estatística.

\ **Erro aleatório**: é um erro que representa a diferença entre o valor medido da variável e seu valor verdadeiro.

\ **Erro sistemático**: é um erro introduzido continuamente, devido, por exemplo, a erro de calibração, desvios instrumentais, erro de técnica e, também devido à representação inadequada de um processo (como por exemplo a não-consideração da ocorrência de vazamentos, depósitos, etc.).

\ **Variáveis independentes aleatórias**: dado um número finito de variáveis aleatórias, X,Y,...,Z, num espaço amostral definido, são consideradas independentes se:

$$P(X=x_i, Y=y_j, ..., Z=z_k) = P(X=x_i), P(Y=y_j) ,... ,P(Z=z_k)$$

para quaisquer valores de x_i, y_j, z_k,. Têm as seguintes propriedades

(i) $E(XY) = E(X)E(Y)$
(ii) $Var(X+Y) = Var(X) + Var(Y)$
(iii) $Cov(X,Y)= 0$,

onde Cov indica a covariância.

\ **Valor esperado**: dada uma população, chama-se valor esperado à soma dos valores ponderados pelas suas respectivas probabilidades. É representado por

$$E(X) = x_1f(x_1) + x_2f(x_2) + ... + x_nf(x_n)$$

$$E(X) = \Sigma x_iP(X=x_i), \qquad \text{para } i=1, 2, ... , n.$$

\ **Matriz variância-covariância**: dada a variância de X_i, definida como

$$Var(X_i) = E\,[X_i - E(X_i)\,]^2$$

para o caso de duas variáveis, tem-se $X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$

e a matriz variância-covariância do vetor X é definida como segue.

$$Var(X) = E\left[X - E(X)\right]\left[X - E(X)\right]'$$

$$Var(X) = E\begin{bmatrix} X_1 & E(X_1) \\ X_2 & E(X_2) \end{bmatrix}\begin{bmatrix} X_1 - E(X_1) & X_2 - E(X_2) \end{bmatrix}$$

$$Var(X) = \begin{bmatrix} E[X_1 - E(X_1)]^2 & E[X_1 - E(X_1)][X_2 - E(X_2)] \\ E[X_2 - E(X_2)][X_1 - E(X_1)] & E[X_2 - E(X_2)]^2 \end{bmatrix}$$

$$Var(X) = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) \\ Cov(X_1, X_2) & Var(X_2) \end{bmatrix}$$

Esta é uma extensão da definição da variância simples para uma variável simples para o caso onde tem-se duas variáveis. A extensão para 3 ou mais variáveis segue o mesmo padrão.

MADRON et al(1977) descrevem um método, sugerido por BOX(1973), para o cálculo aproximado da matriz variância-covariância (Σ). Consideram as variáveis medidas indiretamente (n_i^+), escritas como função das variáveis primárias ,medidas (t_k^+). O método consiste na linearização das funções f_i em relação às variáveis primárias. Então, é possível aproximar o erro e_i pelas expressões

$$e_i = \sum_k \left(\frac{\partial f_i}{\partial t_k^+} \right) e_{tk} = \sum b_{ik} e_{tk}$$

onde

$$b_{ik} = \left(\partial f_i / \partial t_k^+ \right)$$

e e_{tk} são os erros das variáveis primárias. Então, a matriz covariância Σ é dada por

$$\Sigma = B \Sigma^+ B^T$$

onde Σ^+ é a matriz diagonal das variâncias das variáveis primárias.

No caso mais simples, em que todos os dados são independentes, a matriz variância-covariância é uma matriz diagonal.

\ A matriz Var-Cov para o termo de perturbação (ε_i)

$$\text{seja} \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{bmatrix}$$

desde que $$E(\varepsilon) = \begin{bmatrix} E(\varepsilon_1) \\ E(\varepsilon_2) \\ E(\varepsilon_3) \\ E(\varepsilon_4) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0$$

encontra-se que

$$Var(\varepsilon) = E(\varepsilon\varepsilon')$$

$$Var(\varepsilon) = E\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{bmatrix}\begin{bmatrix} \varepsilon_1\varepsilon_2\varepsilon_3\varepsilon_4 \end{bmatrix}$$

$$Var(\varepsilon) = \begin{bmatrix} Var(\varepsilon_1) & Cov(\varepsilon_1,\varepsilon_2) & Cov(\varepsilon_1,\varepsilon_3) & Cov(\varepsilon_1,\varepsilon_4) \\ Cov(\varepsilon_2,\varepsilon_1) & Var(\varepsilon_2) & Cov(\varepsilon_2,\varepsilon_3) & Cov(\varepsilon_2,\varepsilon_4) \\ Cov(\varepsilon_3,\varepsilon_1) & Cov(\varepsilon_3,\varepsilon_2) & Var(\varepsilon_3) & Cov(\varepsilon_3,\varepsilon_4) \\ Cov(\varepsilon_4,\varepsilon_1) & Cov(\varepsilon_4,\varepsilon_3) & Cov(\varepsilon_4,\varepsilon_3) & Var(\varepsilon_3) \end{bmatrix}$$

mas tem-se que

$Var(\varepsilon_1) = Var(\varepsilon_2) = Var(\varepsilon_3) = Var(\varepsilon_4) = 0$

Se assumirmos que

$Cov(\varepsilon_i, \varepsilon_j) = 0, \qquad i \neq j.$

é o mesmo que dizer que ε_i , ε_j são estatisticamente independentes, ou seja, um não afeta o outro. Então a matriz variância-covariância se torna

$$Var(\varepsilon) = E(\varepsilon\varepsilon')$$

$$Var(\varepsilon) = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & \sigma^2 \end{bmatrix}$$

$$Var(\varepsilon) = \sigma^2\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Var(\varepsilon) = \sigma^2 I$$

A2 DISTRIBUIÇÃO NORMAL

Uma variável randômica é dita ser normalmente distribuída com média μ e variância σ^2 e escreve-se $X \sim N(\mu,\sigma^2)$ se a sua função densidade é

$$f(x) = \frac{1}{2\sqrt{2\pi\sigma}} e^{-(x-\mu)^2/2\sigma^2} \qquad \text{para } -\infty < x < +\infty$$

A densidade normal f(x) é uma curva em forma de sino que é simétrica em relação a μ e que atinge a altura máxima em

$$\frac{1}{\sigma\sqrt{2\pi}} \approx \frac{0,399}{\sigma} \quad (\text{em } x=\mu)$$

Para as distribuições normais,

(a) 68,27% dos casos estão incluídos entre $\overline{X}-\sigma$ e $\overline{X}+\sigma$,
(b) 95,45% dos casos estão incluídos entre $\overline{X}-2\sigma$ e $\overline{X}+2\sigma$,
(c) 99,73% dos casos estão incluídos entre $\overline{X}-3\sigma$ e $\overline{X}+3\sigma$,

A distribuição normal é utilizada para aproximar probabilidades associadas a variáveis randômicas binomiais quando o número de parâmetros é muito grande. A base teórica de utilização da distribuição normal é dada pelo teorema do limite central.

Sem apresentar deduções, as quais podem ser conferidas em SHELDON(1986), tem-se

$$E[X] = \phi'(0) = \mu$$
$$E[X] = \phi''(0) = \sigma^2 + \mu^2$$

e assim,

$$Var(X) = E[X^2] - (E[X])^2 = \sigma^2$$

deste modo, μ e σ^2 representam respectivamente a média e a variância de distribuição normal.

Um fato importante sobre variáveis normalmente distribuídas se refere ao fato de que se X se encontra nestas condições, com parâmetros μ e σ^2, então $Y=\alpha X+\beta$ é normalmente distribuída com parâmetros $\alpha\mu + \beta$ e $\alpha^2\sigma^2$.

Segue que se $X \sim N(\mu,\sigma^2)$, então

$$Z = \frac{X - \mu}{\sigma} \rightarrow N\sim(0,1)$$

é a variável randômica normal com média zero e variância 1. Esta variável randômica Z é considerada como tendo uma distribuição normal ou unitária. Os valores desta função distribuição são tabelados e com eles se tem um meio de obter as probabilidades associadas a qualquer variável normal, conhecidos a média e o desvio padrão e a definição de variável aleatória padronizada (Z).

A3 DISTRIBUIÇÃO QUI-QUADRADA

Se $Z_1, Z_2,...,Z_n$, são variáveis randômicas independentes com $\sigma=1$, então, X definida por

$$X = Z_1^2 + Z_2^2 + ... + Z_n^2$$

é dita ser a distribuição qui-quadrada com os graus de liberdade em notação simplificada

$$X \sim \chi^2_n$$

computando-se o momento gerador da função qui-quadrada da variável randômica com n graus de liberdade, tem-se

p/ n=1

$$E\left[e^{tx}\right] = E\left[e^{tz^2}\right] \quad \text{onde } Z \sim N(0,1)$$

$$E\left[e^{tx}\right] = \int_{-\infty}^{\infty} e^{tx^2} f_z(x)dx$$

$$E\left[e^{tx}\right] = \frac{1}{\sqrt{2\pi}} \int e^{tx^2} e^{-x^2/2} dx$$

$$E\left[e^{tx}\right] = \frac{1}{\sqrt{2\pi}} \int e^{-x^2(1-2t)/2} dx$$

$$E\left[e^{tx}\right] = \frac{1}{\sqrt{2\pi}} \int e^{-x^2/2\bar{\sigma}^2} dx$$

$$E\left[e^{tx}\right] = (1-2t)^{-0,5} \frac{1}{\sqrt{2\pi\bar{\sigma}}} \int e^{-x^2/2\bar{\sigma}^2} dx$$

$$E\left[e^{tx}\right] = (1-2t)^{-0,5}$$

para o caso geral,

$$E\left[e^{tx}\right] = E\left[e^{t\sum_{i=1}^{n} Z_i^2}\right]$$

$$E\left[e^{tx}\right] = E\left[\prod_{i=1}^{n} e^{tz_i^2}\right]$$

$$E\left[e^{tx}\right] = \prod_{i=1}^{n} E\left[e^{tz_i^2}\right]$$

$$E\left[e^{tx}\right] = (1-2t)^{-n/2}$$

Da definição de função gama, tem-se que $[1/(1-2t)]^{n/2}$ é o seu momento gerador com parâmetros (n/2, 1/2). conseqüentemente, pela unicidade do momento gerador de função, segue que estas duas distribuições, qui-quadrado com n graus de liberdade e a função gama com parâmetros n/2 e 1/2, são idênticas, o que leva a concluir que a função densidade X é dada por

$$f(x) = \frac{0{,}5e^{-x/2}\left(\frac{x}{2}\right)^{(n/2)-1}}{\Gamma(n/2)} \qquad \text{para } x>0.$$

A distribuição qui-quadrada tem uma propriedade aditiva tal que X_1+X_2 são independentes, com n_1 e n_2 graus de liberdade, respectivamente, então, X_1+X_2 é qui-quadrada com $n_1 + n_2$ graus de liberdade.

Se X é uma variável randômica qui-quadrada, com n graus de liberdade então para qualquer $\alpha \in (0,1)$ a quantidade $\chi^2_{\alpha,n}$ é definida tal que

$$P\{ X >= \chi^2_{\alpha,n} \} = \alpha$$

A.4 TESTES DE HIPÓTESES

Uma hipótese estatística é, usualmente, uma declaração sobre um conjunto de parâmetros da distribuição de uma população. É chamada de hipótese porque não se sabe se é ou não verdadeira. Um problema inicial é desenvolver um procedimento para determinar se os valores de uma amostra randômica desta população está ou não de acordo com a hipótese. Por exemplo, considere uma população particular normalmente distribuída tendo um valor médio associado θ e uma variância 1. A declaração "θ é menor que 1" é a hipótese estatística que podemos testar observando uma amostra randômica desta população. Se a amostra randômica é julgada ser consistente com a hipótese em consideração, diz-se que a hipótese foi aceita, de outro modo, que a hipótese foi rejeitada. Deve-se notar, no entanto, que ao aceitar uma dada hipótese, o que se está fazendo, em verdade, é dizer que os resultados dos dados são consistentes mas não que são verdadeiros.

Níveis de significância

Considere a população com a distribuição F_θ onde o θ é desconhecido e suponha que desejemos testar uma hipótese sobre θ. Denotaremos esta hipótese por H_0 e a chamaremos de hipótese nula. Se F_θ é uma função distribuição normal com média θ e variância 1, então duas possíveis hipóteses nulas sobre θ são

(a) $H_0 : \theta = 1$
(b) $H_0 : \theta <= 1$

A primeira destas hipóteses declara que a população é normal com média 1 e variância 1, ao passo que a segunda declara que é normal de variância 1 e média igual ou menor que 1. Notar que se a declaração (a) for verdadeira, fica especificado completamente a distribuição da população, se ao contrário, o caso (b) for o verdadeiro, então não estará totalmente especificada. A hipótese (a) é chamada simples e a (b) de composta.

A fim de testar uma hipótese H_0 específica, tem-se que uma amostra de uma população de tamanho n — X_1, ..., X_n — é observada. Baseado nestes n valores, deve-se decidir se H_0 deve ser aceita ou não. Um teste para H_0 pode ser especificado definindo-se uma região C num espaço n-dimensional com o compromisso que a hipótese será rejeitada se a amostra estiver fora do espaço C e aceita se ocorrer de forma contrária. a região C é chamada de região crítica. Em outras palavras, o teste estatístico determinado para a região crítica C é tal que

aceita H_0 se $(X_1, X_2, ..., X_n) \notin C$
e
rejeita H_0 se $(X_1, X_2, ... , X_n) \in C$

De outro modo, pode-se dizer, se a hipótese de que $\theta=\theta_0$ é verdadeira, por quanto θ pode diferir de θ_0 antes que a hipótese seja rejeitada por parecer errada? Se a hipótese $\theta=\theta_0$ for verdadeira, $E(\theta)=\theta_0$ e a probabilidade de que o valor de $\hat{\theta}$ seja menor ou igual a $\theta_{\alpha/2}$ é

$$\mathbf{P}\left(\hat{\theta} \leq \theta_{\alpha/2}\right) = \frac{\alpha}{2}$$

e devido à simetria da curva de distribuição Normal

$$\mathbf{P}\left(\hat{\theta} > \theta_{1-\alpha/2}\right) = \frac{\alpha}{\mathbf{2}}$$

Para tomar uma decisão concernente à hipótese, seleciona-se um valor de α, que é chamado o nível de significância, para o teste, antes de coletar a amostra. usualmente, α é escolhido de forma arbitrária, para ser suficientemente pequeno, a fim de que à vista do analista, seja bastante improvável que $\hat{\theta}$ exceda o valor selecionado de $\theta_{1-\alpha/2}$ ou seja menor do que $\theta_{\alpha/2}$. Por exemplo, α pode ser 0,01 ou 0,05. Então, a amostra é coletada e $\hat{\theta}$ é calculado. Se $\hat{\theta}$ for maior do que $\theta_{1-\alpha/2}$ ou menor do que $\theta_{\alpha/2}$, a hipótese é rejeitada. Caso contrário, ela é aceita. HIMMELBLAU (1978) observa que a rejeição da hipótese não implica um resultado definitivo, mas indica que os dados e o procedimento experimental devem ser submetidos a um exame cauteloso para averiguar se ocorreu alguma coisa errada com a coleta de medidas ou com a instrumentação.

Podem-se distinguir dois tipos de erros ao se testar uma hipótese:

(i) erro do Primeiro Tipo (Erro Tipo I), que é o risco de declarar-se falsa uma hipótese verdadeira;

(ii) erro do SegundoTipo (Erro Tipo II), que é o risco de não rejeitar uma hipótese, quando ela é falsa.

Isto pode se resumido no esquema abaixo, onde a hipótese que está sendo testada é a hipótese H.

HIPÓTESE	ACEITAR H	REJEITAR H
H é verdadeira	Decisão Correta	Erro do Tipo I
H é falsa	Erro do Tipo II	Decisão correta

Certamente o erro do Tipo I, existe porque α é selecionado para ser um valor não-zero. Quando a hipótese é verdadeira, e $\alpha = 0{,}05$, por exemplo, em 5% dos testes a hipótese será rejeitada, o que é uma decisão errada.

A probabilidade β é a probabilidade de não rejeitar uma diferença quando ela existe. Existem curvas chamadas de operação características, para determinar a probabilidade β. A probabilidade $(1-\beta)$ é chamada de potência do teste e representa a probabilidade de tomar-se a decisão correta (rejeitar a hipótese), quando ela é realmente errada. Quando a diferença entre as médias (δ) aumenta, $(1-\beta)$ aumenta e β diminui. A seguir, apresenta-se de forma resumida , um exemplo:

SE	PROB. DE CONCLUIR QUE	PROB. DE CONCLUIR QUE
	$\mu=\mu$	$\mu\neq\mu$
$\mu=\mu_A$	$1-\alpha$	α
$\mu=\mu_A+\delta$	β	$1-\beta$

HIMMELBLAU(1978) relata que, pela descrição dos dois tipos de erros, pode-se observar que a tentativa de diminuir um tipo de erro resulta em um aumento no outro tipo de erro. O único modo de diminuir os dois tipos de erros , simultaneamente, é aumentar o tamanho da amostra, o que pode ser caro, na prática. Observa que talvez um tipo de erro tenha conseqüências menos sérias do que o outro, e neste caso, há alguma decisão adequada referente à seleção de valores de α e ao número de observações a ser feito. A experiência leva em conta os instrumentos, o projeto do processo e os custos, de modo a tomar-se uma decisão econômica para α e β. Em geral, os estudos descritos na literatura para a detecção e identificação de erros grosseiros, na reconciliação de dados de processo, consideram-se a probabilidade de ocorrência de erro Tipo I.

B. CONCEITOS MATEMÁTICOS

B.1 MATRIZES E VETORES

VETORES

Chama-se vetor a um conjunto ordenado de escalares.

Ex.: $\mathbf{p} = [22 \quad 6 \quad 180]$

para denotar um componente particular do vetor, usualmente se usa um subscrito junto ao nome do vetor. Ex.: $p_2=6$.

Dimensão do vetor é o seu número de componentes. Vetor com n-componentes é um n-vetor.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad x_i \text{ é um escalar.} \tag{B-0}$$

O grupo de todos os n-vetores da forma (B-0) é denotado por R^n.

Dois vetores são ditos serem iguais se todos os seus componentes forem iguais, isto implica que devem ter a mesma dimensão.

vetor-zero é aquele em que todos os componentes são nulos (a sua dimensão deve estar clara).
vetor-não-zero é aquele em que pelo menos um dos seus componentes não é zero.

Dados dois vetores, x e y, seus produtos internos (denotado por $\mathbf{x}^T\mathbf{y}$) é um escalar e é definido por

$$\mathbf{x}^T\mathbf{y} = x_1y_1 + x_2y_2 + \ldots + x_ny_n = \sum_{i=1}^{n} x_iy_i = \mathbf{y}^T\mathbf{x}$$

define-se comprimento Euclidiano de um vetor como sendo o produto $\sqrt{\mathbf{x}^T\mathbf{x}}$ e tem-se que o vetor cujo comprimento Euclidiano é unitário é chamado de normalizado.

Dado x, o vetor normalizado com a mesma direção é $\mathbf{u} = \left(\frac{1}{\sqrt{\mathbf{x}^T\mathbf{x}}}\right)\mathbf{x}$

*Para dois vetores não-zero cujo produto interno é nulo, diz-se serem ortogonais.
*Dados três vetores a, b e c não-zero de dimensões maiores que 1 tal que

$$\mathbf{a}^T\mathbf{b} = \mathbf{a}^T\mathbf{c},$$

não é necessário que **b** seja igul a **c**, entretanto, é verdade que

$$\mathbf{a}^T(\mathbf{b} - \mathbf{c}) = 0$$

e que consequentemente **a** é ortogonal a (**b-c**). Somente o vetor zero é ortogonal a todos os vetores.

MATRIZES

Uma matriz é um conjunto de escalares sujeitos a duas ordenanças

$$\mathbf{A} = \begin{bmatrix} \rightarrow \\ \downarrow \end{bmatrix} \qquad A = m \times n \begin{cases} \text{m linhas} \\ \text{n colunas} \end{cases}$$

Se m = n, então a matriz é quadrada. $\mathbf{A} = \mathbf{B}$ se e somente se $\mathbf{a_{ij}} = \mathbf{b_{ij}}$. Têm a mesma dimensão se tiverem a mesma quantidade de linhas e de colunas. Se a dimensão das linhas ou das colunas é zero, a matriz é nula.

Define-se matriz transposta de uma matriz dada **A** e representa-se por $\mathbf{A}^T$ à matriz dada por

$$\mathbf{A}^T_{ij} = \mathbf{A}_{ji} \quad \begin{cases} \mathbf{1 \le i \le n} \\ \mathbf{1 \le j \le m} \end{cases} \text{ e se } \mathbf{A} = m \times n,\ \mathbf{A}^T = n \times m.$$

se $\mathbf{A} = \mathbf{A}^T$, a matriz é dita ser simétrica.

Somente duas matrizes de mesma dimensão podem ser somadas ou subtraídas
Uma matriz pode se multiplicada por um escalar e neste caso, o escalar multiplica cada um dos termos da matriz.
Para existir o produto entre duas matrizes, é necessário que o número de colunas da primeira matriz seja igual ao número de linhas da segunda matriz. Assim, dadas **A** (m x n) e **B** (n x p), tem-se que **AB** (m x p)

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{n} \mathbf{a}_{ik}\mathbf{b}_{kj}\ , \text{ para i=1, 2, 3,..., m e j = 1, 2, 3, ... , p}$$

Quando as matrizes A, B e C são matrizes de dimensões coerentes, tem-se que

$$\begin{cases} \mathbf{AB(C) = (AB)C} \\ \mathbf{A(B + C) = (A + B) + C} \\ \mathbf{AB \neq BA,\ em\ geral} \end{cases}$$

$$\mathbf{(AB)^T = B^T A^T}$$

Algumas matrizes têm estrutura especial e isto pode ser útil pois elas representam características especiais.
*Pode-se escrever uma matriz cujos elementos são blocos e não elementos simplesmente.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2^T \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix}$$

*Um outro tipo de matriz especial é a matriz identidade de ordem n, represenada por I_n. É uma matriz onde os elementos da diagonal principal são 1's e os elementos fora da mesma, são nulos. É por estas características, o elemento neutro na multiplicação de matrizes.

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

*Matriz diagonal é aquela em que todos os elementos fora da diagonal principal são nulos. Quando o tamanho de uma matriz diagonal não é especificado, assume-se que seja quadrada.

*Uma matriz é dita ser triangular superior se

$\mathbf{u_{ij}} = \mathbf{0}$, para todo i > j.

*Uma matriz é dita ser triangular inferior se

$\mathbf{l_{ij}} = \mathbf{0}$, para todo i < j.

Se todos os elementos da diagonal da matriz diagonal são unitários, tem-se a matriz triangular unitária.

Um grupo de vetores não-nulos $\{ \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k \}$ é dito ser ortogonal se

$\mathbf{p}_i^T\mathbf{p}_j = \mathbf{0}$, para todo $i \neq j$. Se além disto, cada vetor em um grupo ortogonal tem também a propriedade de $\mathbf{p}_i^T\mathbf{p}_j = \mathbf{1}$, para todo $1 \leq \mathbf{i} \leq \mathbf{k}$, então o grupo de vetores é dito ser ortonormal. Tem-se que uma matriz **P** com colunas ortonormais, satisfaz

$$\mathbf{P}^T\mathbf{P} = \mathbf{I}_n$$

As linhas de uma matriz ortonormal não são necessariamente ortonormais, a menos que m = n. Usa-se a letra Q para denotar uma matriz ortogonal. Uma matriz quadrada com colunas ortonormais é ortonormal. Se Q é ortonormal, implica que

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{I} \qquad \text{(B-0a)}$$

Se **x** e **y** são vetores de qualquer dimensão, a matriz da forma $\mathbf{xy}^T$ é chamada o produto externo de **x** e **y** e é uma matriz de posto 1, e toda coluna da matriz produto é um múltiplo de **x** e toda linha é um múltiplo de $\mathbf{y}^T$. Existe maior interesse quando a matrix é quadrada.

Define-se matriz elementar à matriz da forma

$$\mathbf{E} = \mathbf{I} - \alpha \mathbf{u}\mathbf{v}^T$$

onde
α é um escalar
u, **v** são vetores de mesma dimensão.

Uma característica prática da matriz **E** é que seu produto por um vetor pode ser calculado sem precisar explicitar os elementos da matriz.

$$\mathbf{E}\mathbf{x} = \left(\mathbf{I} - \alpha \mathbf{u}\mathbf{v}^T\right)\mathbf{x} = \mathbf{x} - \alpha \mathbf{u}\left(\mathbf{v}^T\mathbf{x}\right) = \mathbf{x} - \xi \mathbf{u},$$

onde $\xi = \alpha \mathbf{v}^T\mathbf{x}$ é um escalar.

ESPAÇOS VETORIAIS

Assuma $\{ a_1, a_2, a_3, \ldots, a_n\}$ como sendo um grupo de vetores de uma mesma dimensão. Estes vetores são ditos serem linearmente dependentes se o vetor zero pode ser escrito como uma combinação linear não trivial destes vetores.

$\alpha_1 a_1 + \alpha_2 a_2 + \alpha_3 a_3 + \cdots + \alpha_n a_n$ com $\alpha_i \neq 0$ para no mínimo 1 i.

conseqüentemente, os vetores são ditos linearmente independentes se

$\alpha_1 a_1 + \alpha_2 a_2 + \alpha_3 a_3 + \cdots + \alpha_n a_n$ implicar $\alpha_1 = \alpha_2 = \alpha_3 = \cdots = \alpha_n = 0$ (trivial)
α é chamado de coeficiente da combinação linear e indica o peso de cada parcela.

O número máximo trivial de dependência linear entre n-vetores é n.

O produto matriz-vetor, vê-se que é uma combinação linear das colunas da matriz **A**. Consequentemente, é conveniente considerar um grupo de vetores $\{a_j\}$ j =1,2,...,n como as colunas de A. Usando tais definições, dependência linear de A é equivalente a

$\mathbf{A}\mathbf{z} = \mathbf{0}$, para algum vetor $\mathbf{z} \neq \mathbf{0}$.

independência linear das colunas de **A** é equivalente à condição
$\mathbf{A}\mathbf{z} = \mathbf{0}$, implica $\mathbf{z} = \mathbf{0}$.

Considere $\mathbf{A}\mathbf{x} = \mathbf{b}$, com respeito às relações entre as colunas de **A**. Primeiro assuma que $\mathbf{A}\mathbf{x} = \mathbf{b}$ para algum **x**, mas que as colunas de **A** sejam linearmente dependentes, tal que $\mathbf{A}\mathbf{z} = \mathbf{0}$ para algum vetor não-zero. Então é também possível de ser expresso na forma $\mathbf{A}(\mathbf{x} + \alpha\, \mathbf{z})$ para qualquer escalar α, desde que a linearidade da transformação **A**, implica que

$$\mathbf{A}\left(\mathbf{x} + \alpha \mathbf{z}\right) = \mathbf{A}\mathbf{x} + \alpha \mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x} = \mathbf{b}$$

por esta razão, o sistema $\mathbf{Ax} = \mathbf{b}$ tem infinitas soluções e a representação de **b** como uma combinação linear das colunas de **A** não é única.

Por outro lado, se as colunas de **A** são linearmene independentes, e $\mathbf{Ax} = \mathbf{b}$ para algum vetor **x**, então **x** é a única solução de $\mathbf{Ax} = \mathbf{b}$. A conclusão sobre a unicidade de **x** no caso de colunas LI, freqüentemente vem quando **A** é não quadrada.

Finalmente, é possível que $\mathbf{Ax} = \mathbf{b}$ não possa ser satifeito, freqüentemente quando as colunas de **A** são linearmente independentes (a menos que A seja quadrada).

FAIXAS E ESPAÇOS NULOS

Seja **S** contendo um conjunto de vetores de dimensão m. Diz-se que **S** é um subespaço de $\mathbf{R}^m$ se, para quaisquer escalares α e β

$$\mathbf{x}, \mathbf{y} \in \mathbf{S} \text{ implica } \alpha\mathbf{x} + \beta\mathbf{y} \in \mathbf{S} \qquad \text{(B-1)}$$

Esta propriedade imediatamente implica que todo subespaço contém o vetor nulo (fazendo $\alpha = \beta = 0$).

Dado um subespaço **S**, um grupo de vetores k $\{a_j\}$, j=1, 2, ... , K é dito cobrir **S** se o vetor **x** em **S** pode ser escrito como uma combinação linear do grupo dos vetores. Então se terá um grupo de vetores onde qualquer outro vetor pode ser escrito como uma combinação linear dos vetores do grupo. O subespaço que contém somente o vetor nulo é dito ser trivial e tem dimensão zero.

Para qualquer subespaço não-trivial, há um único inteiro positivo e pequeno chamando posto ou dimensão do subespaço tal que todo vetor no subespaço pode ser expresso como uma combinação linear de um conjunto fixo de **r** vetores.Qualquer grupo de **r** vetores é dito formar a base para o subespaço. Os vetores na base não são únicos, mas eles devem ser independentes. Para qualquer inteiro positivo k, um grupo de k vetores, linearmente independentes(LI) (senão ao menos um deles) poderá ser expresso como uma combinação linear dos outros e conseqüentemente **r** não seria mínimo. Para qualquer inteiro k, um grupo de k vetores LI é uma base para um subespaço k-dimensional.

O grupo de todos os m-vetores que são combinações lineares das colunas da matriz A, m x n é chamado de faixa de **A** e será denotado por R(**A**).

$$\mathbf{b} \in \mathbf{R(A)} \leftrightarrow \exists\, \mathbf{x} \in \mathbf{R}^m / \mathbf{b} = \mathbf{Ax} \qquad \text{(B-2)}$$

O sistema $\mathbf{Ax} = \mathbf{b}$ é então compatível se e somente se **b** está em alguma faixa de **A**. Uma observação crucial é que a faixa de **A** é um subespaço de $\mathbf{R}^m$. Considere quaisquer dois vetores b_1, $b_2 \in \mathbf{R(A)}$. Por definição de faixa (A), deve-se manter $\mathbf{b}_1 = \mathbf{Ax}_1$ e $\mathbf{b}_2 = \mathbf{Ax}_2$ para algum $\mathbf{x}_1$ e $\mathbf{x}_2$. Por causa de A ser uma transformação linear, para quaisquer escalares α_1 e α_2, tem-se

$$\alpha_1\mathbf{b}_1 + \alpha_2\mathbf{b}_2 = \alpha_1\mathbf{Ax}_1 + \alpha_2\mathbf{Ax}_2 = \mathbf{A}(\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2) = \mathbf{A}\,\mathbf{y} \in \mathbf{R(A)} \qquad \text{(B-3)}$$

o que significa que **R(A)** satisfaz (B-1).

Por exemplo:

dada a matriz $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}$, tem-se que a faixa (**A**)inclui todos dos vetores da forma

$$\mathbf{A}\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = \begin{bmatrix} \gamma_1 + \gamma_2 \\ \gamma_1 \\ \gamma_1 - \gamma_2 \end{bmatrix} \qquad \text{(B-4)}$$

para quaisquer escalares γ_1 e γ_2.

Similarmente, o grupo de todos os vetores que são combinações lineares das linhas (transposta) de **A** define um subespaço de vetores expressos como $\mathbf{A}^T\mathbf{v}$ para algum m-vetor **v**. Este subespaço é denotado por $\mathbf{R}(\mathbf{A}^T)$ ou a linha-espaço de **A**.

A faixa $\mathbf{A}^T$ é definida como segue

$$\mathbf{x} \in \mathbf{R}(\mathbf{A}) \leftrightarrow \exists\, \mathbf{v} \in \mathbf{R}^n \,/\, \mathbf{A}^T\mathbf{v} = \mathbf{x}.$$

O posto-coluna da matriz **A** é definido como o número máximo de colunas linearmente independentes de **A**. Similarmente, o posto-linha da matriz é o número máximo de linhas linearmente independentes.

Pode ser mostrado que o posto-linha e o posto-coluna devem ser iguais. Seu valor comum é chamado posto de **A**. Obviamente, o posto de **A** não pode exceder à menor dimensão de suas linhas e colunas. Somente a matriz nula (de dimensão nula) ou matriz zero (de qualquer dimensão), têm posto zero.

$$\begin{cases} \text{posto} \rightarrow \text{conjunto de vetores LI} \\ \text{faixa} \rightarrow \text{conjunto de vetores LD} \end{cases}$$

A matriz cujas linhas são linearmente independentes é dita ter posto-linha completo (com definição similar para posto-coluna). Se **A** tem posto-coluna completo, suas colunas formam a faixa **A**, e se **A** tem posto-linha completo, as colunas de $\mathbf{A}^T$ formam a base para a faixa de $\mathbf{A}^T$.

A matriz **A** é dita ter posto total ou completo se ela tiver posto-linha ou posto-coluna completos.

Se a matriz tem posto menor que o min (m x n), a matriz é chamada de posto-deficiente.

Qualquer matriz **A** define dois outros subespaços (separados), a partir da faixa **A** e faixa $\mathbf{A}^T$. Para uma matriz **A** (m x n), o conjunto de todos os m-vetores ortogonais na faixa **A** são chamados espaço nulo de $\mathbf{A}^T$ e é denotado por

$$\text{NULL}(\mathbf{A}^T) \text{ ou } \eta(\mathbf{A}^T)$$

Os elementos de $\eta(\mathbf{A}^T)$ são definidos como

$$\mathbf{Z} \in \eta\left(\mathbf{A}^{\mathrm{T}}\right) \leftrightarrow \mathbf{A}^{\mathrm{T}} = 0$$

Para verificar que $\eta(\mathbf{A}^{\mathrm{T}})$ é um subespaço, mostra-se que a equação (B-1) é satisfeita. Se $\mathbf{Z}_1$ e $\mathbf{Z}_2$ estão em $\eta(\mathbf{A}^{\mathrm{T}})$, então. $\mathbf{A}^{\mathrm{T}}\mathbf{Z}_1 = 0$ e $\mathbf{A}^{\mathrm{T}}\mathbf{Z}_2 = 0$. Para quaisquer escalares α_1 e α_2, a linearidade de $\mathbf{A}^{\mathrm{T}}$ implica que

$$\mathbf{A}^{\mathrm{T}}\left(\alpha_1\mathbf{Z}_1 + \alpha_2\mathbf{Z}_2\right) = \alpha_1\mathbf{A}^{\mathrm{T}}\mathbf{Z}_1 + \alpha_2\mathbf{A}^{\mathrm{T}}\mathbf{Z}_2 = 0 \qquad \text{(B-5)}$$

tal que $\alpha_1\mathbf{Z}_1 + \alpha_2\mathbf{Z}_2$ estão em $\eta(\mathbf{A}^{\mathrm{T}})$. O espaço nulo de **A**, denotado por $\eta(\mathbf{A})$, é o subespaço que consiste de todos os n-vetores **q** tais que $\mathbf{Aq} = \mathbf{0}$.

Qualquer m-vetor pode ser escrito como uma combinação linear dos vetores na faixa de **A** e $\eta(\mathbf{A}^{\mathrm{T}})$, e a intersecção da faixa de **A** e $\eta(\mathbf{A}^{\mathrm{T}})$ é vazia, exceto para o vetor zero. Conseqüentemente, pode-se dizer que soma das dimennsões da faixa (**A**) e $\eta(\mathbf{A}^{\mathrm{T}})$ é m. Em particular, se **A** tem colunas linearmente independentes (as quais implicam que $m \geq n$), as dimensões da faixa e do espaço nulo de $\mathbf{A}^{\mathrm{T}}$ são n e m-n, respectivamente. Propriedades análogas valem para a faixa de $\mathbf{A}^{\mathrm{T}}$ e $\eta(\mathbf{A})$. Qualquer n-vetor pode ser escrito como combinação linear dos vetores na faixa ($\mathbf{A}^{\mathrm{T}}$) e NULL (**A**), e a soma das dimensões da faixa e do espaço nulo é n.

Pelo fato de a faixa de (**A**) e o NULL (**A**) conterem somente o vetor zero em comum, a expressão de qualquer m-vetor não zero **b** na seguinte forma é única

$$\mathbf{b} = \mathbf{b}_R + \mathbf{b}_N \quad \text{com } \mathbf{b}_R \in \text{faixa } (\mathbf{A}) \text{ e } \mathbf{b}_N \in \text{NULL } (\mathbf{A}^{\mathrm{T}}) \qquad \text{(B-6)}$$

e os vetores b_R e b_N satisfazem

$\mathbf{b}_R^{\mathrm{T}}\mathbf{b}_N = 0$, o que implica serem ortogonais, e

$$\mathbf{b}^{\mathrm{T}}\mathbf{b} = \mathbf{b}_R^{\mathrm{T}}\mathbf{b}_R + \mathbf{b}_N^{\mathrm{T}}\mathbf{b}_N \qquad \text{(B-7)}$$

Quando a especificação de **A** está clara no contexto, os vetores $\mathbf{b}_R$ e $\mathbf{b}_N$ são chamados de componentes do espaço de faixa e do espaço nulo do vetor **b**.

Mudando o espaço da faixa (**A**), qualquer n-vetor não-zero **x** similarmente, tem uma única representação da forma

$$\mathbf{x} = \mathbf{x}_R + \mathbf{x}_N \qquad \text{(B-8)}$$

onde

$\mathbf{x}_R \in$ faixa (**A**) e $x_N \in$ NULL (**A**).

MATRIZES SINGULARES E NÃO-SINGULARES

Quando a matriz é quadrada, usa-se termos especiais para caracterizar a dependência linear de suas colunas. Uma matriz quadrada com colunas LI é dita ser não-singular ao passo que uma matriz quadrada com colunas LD é dita ser singular. Se **A** é uma matriz n x n e não-singular, as colunas de A abrangem tudo de $\Re^n$, tal que os

subespaços nulos de ($\mathbf{A}^T$) e de (**A**) contêm somente o vetor zero. Tem-se as seguintes propriedades acerca da independência linear:

* **A** é não-singular se $\mathbf{Ax} = \mathbf{0}$ somente quando $\mathbf{x} = \mathbf{0}$.
* Se **A** é não-singular. $\mathbf{Ax} = \mathbf{b}$ tem sempre solução única.
* **A** é singular se e somente se existe $\mathbf{x} \neq \mathbf{0}$ tal que $\mathbf{Ax} = \mathbf{0}$.
* Se **A** é singular e $\mathbf{Ax} = \mathbf{b}$ tem uma solução, ela tem um número infinito de soluções.

Para toda matriz não-singular **A**, existe uma matriz $\mathbf{A}^{-1}$, chamada inversa, tal que $\mathbf{A}^{-1}$ "desfaz" o efeito de **A**, que é $\mathbf{A}^{-1}(\mathbf{Ax}) = x$ para todo vetor **x**.

O inverso da matriz satisfaz

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{B-9}$$

Se $\mathbf{A}^{-1}$ existe, ela é única, não-singular e tem a propriedade $\mathbf{AA}^{-1} = \mathbf{I}$, se A é singular, sua inversa não existe.

Se **A** e **B** não são singulares, o produto **AB** é também não-singular, e

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \tag{B-10}$$

Se **A** é não-singular, $\mathbf{A}^T$ é também não-singular e

$$(\mathbf{A}^T)^{-1} = \left(\mathbf{A}^{-1}\right)^T$$

a qual indica que podemos utilizar a notação $\mathbf{A}^{-T}$ sem ambigüidade.

O inverso de certos tipos de matrizes tem uma forma especial. Se **D** é uma matriz não-singular e diagonal, $\mathbf{D} = \text{diag}\,(d_1, d_2, \ldots, d_n)$, então $\mathbf{D}^{-1}$ é também diagonal.

$$\mathbf{D}^{-1} = \text{diag}(\mathbf{d}_1^{-1}, \cdots, \mathbf{d}_n^{-1})$$

Uma matriz quadrada é singular se e somente se no mínimo um dos seus elementos da diagonal for zero. Quando Q é ortogonal, as relações (B-0a) e (B-9) combinadas com a unicidade da inversa, implica que

$$\mathbf{Q}^{-1} = \mathbf{Q}^T$$

o que significa que a inversa de uma matriz ortogonal é sua transposta.

O inverso de uma matriz elementar não-singular é também uma matriz elementar envolvendo os mesmos vetores. Em particular pode ser mostrado que, se $\alpha\mathbf{u}^T\mathbf{v} \neq \mathbf{1}$

$$\mathbf{E}^{-1} = \left(\mathbf{I} - \alpha\mathbf{uv}^T\right)^{-1} = \mathbf{I} - \beta\mathbf{uv}^T, \text{ onde} \qquad \beta = \frac{\alpha}{\alpha\mathbf{uv}^T - 1}$$

Cálculos explícitos de inversas é fortemente desencorajador, entretanto, a matriz inversa é freqüentemente um formalismo útil em análise teórica. As seguintes fórmulas simples para o inversa de uma inversa 2 x 2 é incluída como exemplo:

$$\begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix}^{-1} = \frac{1}{\alpha\delta - \beta\gamma}\begin{pmatrix} \delta & -\gamma \\ -\beta & \alpha \end{pmatrix} \qquad \text{(B-11)}$$

B.2 MÉTODO DOS MÍNIMOS QUADRADOS LINEAR (MMQL)

Quando **Ax** = **b** não é compatível, usa-se o MMQ para descobrir o melhor (embora imperfeito) ajuste entre **Ax** e **b**.

Para uma matriz geral **A** (m x n) e um vetor **b** arbitrário, pode não haver um n-vetor tal que **Ax** seja exatamente igual a **b**. Por exemplo se m=3 e n =1,

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix} \qquad \text{e} \quad \mathbf{b} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}$$

nenhum múltiplo da coluna simples de **A** pode produzir **b**.

Uma alternativa "natural" para satisfazer **Ax** = **b** é encontrar um **x** tal que **Ax** esteja tão próximo quanto possível de **b** o que em termos matemáticos corresponde a minimizar $\|\mathbf{b} - \mathbf{Ax}\|$ por alguma norma conveniente. A primeira questão a ser resolvida é a definição de proximidade, isto é, a norma que mede a diferença entre **b** e **Ax**. A solução do problema de escolha da norma de minimização varia na natureza e interpretação com a escolha da norma.

A norma mais amplamente utilizada em minização é a norma 2 ou o comprimento Euclidiano. Quando $\|\mathbf{b} - \mathbf{Ax}\|_2$ é tão pequeno quanto possível, diz-se que **x** é uma solução (ótima) do problema de mínimos quadrados:

$$\underset{x \in \Re^n}{\textbf{minimizar}} \quad \|\mathbf{b} - \mathbf{Ax}\|_2^2 \qquad \text{(B-12)}$$

onde a designação abaixo da palavra minimizar especifica a natureza da quantidade que é livre para variar. Desde que as normas são restritas a serem não-negativas, minimizar uma norma equivale a minimizar o seu quadrado e daqui por diante será usado indistintamente as normas quadradas e não-quadradas.

Problemas de MMQ linear têm uma longa história, particularmente em aplicações estatísticas. A formulação dos MMQ é excepcionalmente popular em grande parte porque resolver um problema de MMQ é substancialmente mais fácil que minimizar outras normas de (**b** - **Ax**).

FORMULAÇÃO E DEFINIÇÃO DO PROBLEMA DE MMQ

Geralmente falando, problemas de MMQ surgem quando uma variável dependente é escrita como uma combinação linear de funções envolvendo parâmetros específicos. Enfatiza-se que a designação linear significa que os parâmetros desconhecidos no problema são os coeficientes na combinção linear. As formas funcionais na combinação linear não precisam ser iguais ao número de parâmetros.

A solução do problema MMQ, dá o melhor grupo de coeficientes na combinação linear e possibilita a medida do erro total no melhor modelo. A solução de MMQ não ajuda a escolher o melhor grupo de parâmetros ou a melhor forma funcional, o qual fica a cargo da experiência do "construtor" do modelo. (Um grande erro no modelo pode , freqüentemente, indicar que o modelo é inadequado).

Considere que se tenha colecionado m observações. Para a i-ésima observação, i = 1, 2, ... , m, o modelo linear tem a forma

$$\mathbf{b}_i = \mathbf{a}_{i1}\mathbf{x}_1 + \mathbf{a}_{i2}\mathbf{x}_2 + \cdots + \mathbf{a}_{in}\mathbf{x}_n = \sum_{j=1}^{n} \mathbf{a}_{ij}\mathbf{x}_j, \qquad i = 1, 2, \ldots, m.$$

onde $\mathbf{b}_i$ (o valor da variável dependente) é aproximado pela soma ponderada por $\mathbf{a}_{i1}$, $\mathbf{a}_{i2}$, ..., $\mathbf{a}_{in}$ (o n indica funções avaliadas na observação i). Os valores de $\{\mathbf{b}_i\}$ e $\{\mathbf{a}_{ij}\}$, i = 1, 2, ... , m e j = 1, 2, ... , n, são assumidos serem conhecidos. Os pesos $\{\mathbf{x}_j\}$, j = 1, 2, ..., n são variáveis desconhecidos a serem determinados.

A solução do problema MMQ linear é um vetor que produz a menor soma dos quadrados no modelo acima do dado conjunto de observações, i.e., que resolve

$$\underset{x \in \Re^n}{\textbf{minimizar}} \quad \sum_{i=1}^{m} \left(\mathbf{b}_i - \sum_{i=1}^{n} \mathbf{a}_{ij}\mathbf{x}_j \right) \qquad j = \text{parâmetros e } i = \text{observações}$$

Escrito na forma de matriz, o problema de MMQ linear envolve encontrar um vetor x que

minimiza $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$,

onde

$$\mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \vdots \\ \mathbf{bm} \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} \mathbf{a}_{11} & \mathbf{a}_{12} & \cdots & \mathbf{a}_{1n} \\ \mathbf{a}_{21} & \cdots & \cdots & \mathbf{a}_{2n} \\ \vdots & \cdots & \cdots & \vdots \\ \mathbf{a}_{m1} & \cdots & \cdots & \mathbf{a}_{mn} \end{pmatrix} \quad \text{e} \quad \mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \vdots \\ \mathbf{x}_n \end{pmatrix}$$

A i-ésima linha de **A** e o i-ésimo componente de **b** correspondem à i-ésima observação e a j-ésima coluna de **A** e o j-ésimo componente de **x** correspondem à j-ésima variável desconhecida no modelo.

Se o problema de MMQL tem m > n, é chamado de sobredeterminado porque um modelo linear com n coeficientes livres não pode em geral igular-se a um número de observações maior que n.

Assumindo que **A** e **b** sejam dados, o resíduo do problema de MMQL correspondente ao vetor **x** é o o m-vetor $\rho = \mathbf{b} - \mathbf{A}\mathbf{x}$ cujo i-ésimo componente é $\rho_i = \mathbf{b}_i - \mathbf{a}_i'\mathbf{x}$, onde $\mathbf{a}_i'$ é o vetor linha de (i-ésima linha) de **A**. O i-ésimo componente do resíduo dá-nos o erro no modelo para a i-ésima observação com a corrente escolha de **x**, e o problema de MMQL é equivalente a minimizar a NORMA DOIS do resíduo para todos os possíveis valores de **x**.

PROPRIEDADES DA SOLUÇÃO DO MMQL

É útil pensar no MMQL envolvendo a matriz **A** em termos de subespaços definidos por **A**. O subespaço complementar (o NULL ($\mathbf{A}^T$)) contém todos os m-vetores **z** ortogonais às colunas de **A**, i. e., tais que $\mathbf{A}^T\mathbf{z} = \mathbf{0}$. Faça r denotar o posto de **A**, a menos que especificado de modo diferente, será sempre considerado que r > 0. A dimensão da faixa (**A**) é r e dimensão espaço nulo é m - r.

Dada uma matriz **A**, qualquer m-vetor não-zero **c** pode ser expresso como a soma de um vetor $\mathbf{c}_R$ na faixa (**A**) e um vetor $\mathbf{c}_N$ no espaço nulo ou NULL ($\mathbf{A}^T$). Tem-se assim

$$\mathbf{c} = \mathbf{c}_R + \mathbf{c}_N \tag{B-13}$$

onde $\mathbf{c}_R$ e $\mathbf{c}_N$ são únicos e satisfazem

$$\mathbf{c}_R = \mathbf{A}\,\mathbf{c}_A \text{ para algum } \mathbf{c}_A, \qquad \mathbf{A}^T\mathbf{c}_N = \mathbf{0} \qquad \text{e} \qquad \mathbf{c}_R^T\mathbf{c}_N = 0 \tag{B-14}$$

Em geral, os suspeitos "R"e "N" em qualquer m-vetor denotará os componente de sua faixa e de seu espaço nulo, onde a matriz **A** associada deve ser clara a partir do contexto. A notação $\mathbf{c}_A$ se refere a qualqquer n-vetor que satisfaça $\mathbf{c}_R = \mathbf{A}\mathbf{c}_A$. A unicidade de $\mathbf{c}_R$ e $\mathbf{c}_N$ implica que os componentes da faixa e do espaço nulo de vetores iguais devem ser também iguais. De fato,

$$\mathbf{c} = \mathbf{d}, \qquad \text{se e somente se } \mathbf{c}_R = \mathbf{d}_R \qquad \text{e} \qquad \mathbf{c}_N = \mathbf{d}_N \tag{B-15}$$

Apesar de **c** ser único, o vetor $\mathbf{c}_A$ é único somente se as colunas de **A** forem linearmente independentes.

Por causa da maneira que a norma Euclidiana é definida em termos de um produto interno, as relações (B-13) e (B-14) têm uma conseqüência importante,

$$\|\mathbf{c}\|_2^2 = \mathbf{c}^T\mathbf{c} = \|\mathbf{c}_R\|_2^2 + \|\mathbf{c}_N\|_2^2 \tag{B-16}$$

Tal que a representação (B-13) de um m-vetor em termos de seus componente de sua faixa e de seu espaço nulo separa sua norma Euclidiana em duas partes independentes. (Esta propriedade não vale para outras normas).

Estas observações são importantes para o problema de MMQL porque ela determina a menor norma Euclidiana possível do resíduo $\rho = \mathbf{b} - \mathbf{A}\mathbf{x}$ para todos os vetores **x**. Desde que o lado direito **b**(RHS) e o resíduo são m-vetores, ambos podem ser escritos na forma da equação (B-13)

$$\begin{aligned} &\mathbf{b} = \mathbf{b}_R + \mathbf{b}_N && \mathbf{b} = \mathbf{A}\mathbf{b}_A \text{ e} \\ &\rho = \rho_R + \rho_N && \rho_R = \mathbf{A}\rho_A \text{ os quais estão na faixa (A)} \end{aligned} \tag{B-17}$$

Combinando a definição de ρ e de $\mathbf{b} - \mathbf{A}\mathbf{x}$ com as relações acima, tem-se

$$\rho = \rho_R + \rho_N = \mathbf{b} - \mathbf{A}\mathbf{x} = \mathbf{b}_R + \mathbf{b}_N - \mathbf{A}\mathbf{x}$$

$$\rho = \rho_R + \rho_N = \mathbf{b}_R + \mathbf{b}_N - \mathbf{Ax}$$

Um ponto óbvio mas crucial nesta expressão é que o vetor **Ax** está inteiramente na faixa (**A**). Quando **Ax** é subtraído de **b** para criar o resíduo, segue de (B-15) que o componente na faixa de **b-Ax** deve ser igual a $\mathbf{b}_N$. Os componente da faixa e do espaço nulo satisfazem com isto a

$$\rho_R = \mathbf{b}_R - \mathbf{Ax} \qquad \text{e} \qquad \rho_N = \mathbf{b}_N \tag{B-18}$$

O MMQL envolve minimizar a norma dois do resíduo. Sabe-se de (B-16) e (B-18) que a norma Euclidiana quadrada do resíduo ρ para **x** satisfaz

$$\|\mathbf{b} - \mathbf{Ax}\|_2^2 = \|\rho\|_2^2 = \|\rho_R\|_2^2 + \|\rho_N\|_2^2$$

$$\|\mathbf{b} - \mathbf{Ax}\|_2^2 = \|\mathbf{b}_R - \mathbf{Ax}\|_2^2 + \|\mathbf{b}_N\|_2^2$$

$$\|\mathbf{b} - \mathbf{Ax}\|_2^2 \geq \|\mathbf{b}_N\|_2^2.$$

Desde que $\mathbf{b}_N$ é retido em sua totalidade no resíduo, $\|\mathbf{b} - \mathbf{Ax}\|_2$ será minimizado quando a norma dois de seu componente na faixa $\|\mathbf{b} - \mathbf{Ax}\|_2$ for o menor possível. Por causa de $\mathbf{b}_R$ estar, por definição, na faixa(**A**), um vetor **x** deve existir tal que $\mathbf{Ax} = \mathbf{b}_R$. Para este **x** especial, a faixa completa dos componentes de **b** é removido por subtração de **Ax**, o que significa que $\mathbf{b} - \mathbf{Ax} = \mathbf{b}_N$ e a norma Euclidiana do resíduo é igual à sua menor banda $\|\mathbf{b}_N\|_2$.

A escolha de **x** tal que $\mathbf{Ax} = \mathbf{b}_R$ não somente minimiza a norma dois do resíduo mas também força o resíduo a estar inteiramente no espaço nulo de $\mathbf{A}^T$. Duas características fundamentais são então derivadas, na solução do MMQL ótimo

$$\mathbf{x} \text{ minimiza } \|\mathbf{b} - \mathbf{Ax}\|_2^2 \text{ se o somente se } \mathbf{A}^T(\mathbf{b} - \mathbf{Ax}) = 0 \tag{B-19}$$

$$\mathbf{x} \text{ minimiza } \|\mathbf{b} - \mathbf{Ax}\|_2^2 \text{ se e somente se } \mathbf{Ax} = \mathbf{b}_R \text{ e } \mathbf{b}\text{-}\mathbf{Ax} = \mathbf{b}_N \tag{B-20}$$

Um problema de MMQL é dito ser compatível se seu ótimo residual é zero (onde $\mathbf{b}_N = \mathbf{0}$ e $\mathbf{b} = \mathbf{b}_R$) e ser incompatível em caso contrário.

A ortogonalidade do vetor resíduo às linhas de **A** corresponde ao princípio geométrico familiar que a mais curta distância de um ponto a um plano é o comprimento da ligação perpendicular a eles. Aqui, o "ponto" é o vetor **b**, e o "plano" é o que contém os vetores na faixa(**A**).

A unicidade de $\mathbf{b}_N$ implica que o vetor ótimo do resíduo é sempre único. O vetor $\mathbf{b}_R$ é também único, e o sistema $\mathbf{Ax} = \mathbf{b}_R$ é compatível, por definição. De qualquer modo, sabe-se que o vetor que satisfaz $\mathbf{Ax} = \mathbf{b}_R$ é único se e somente se as colunas de **A** são linearmente independentes.

Desde que qualquer vetor **x** que satisfaça $\mathbf{Ax} = \mathbf{b}_R$ é uma solução do MMQL, uma conclusão importante a que se chega é:

A solução de $\min_{x \in \Re^n} \|\mathbf{b} - \mathbf{Ax}\|_2^2$ é única se e somente se **A** tem posto-coluna completo.

MUDANÇAS NO LADO DIREITO DA EQUAÇÃO (**b**) -RHS

Uma característica crucial na relação (B-20) é que ela revela como uma mudança em **b** afeta a ambas à solução **x** e ao resíduo ρ do MMQL $\|\mathbf{b} - \mathbf{Ax}\|_2^2$. Este ponto precisa ser enfatizado porque o problema do MMQL é bastante diferente do sistema de equações não-singular e compatível. Para estes problemas, qualquer mudança no lado direito muda a solução de **x**. Esta propriedade não se aplica ao MMQL, cuja análise é conseqüentemente mais complicada.

Assuma para simplicidade que A tem posto-coluna completo, e que x é a única solução de MMQL de $\min \|\mathbf{b} - \mathbf{Ax}\|_2^2$. Faça ρ denotar o resíduo ótimo $\mathbf{b} - \mathbf{Ax}$. Suponha agora que b é perturbado por um vetor que esteja inteiramente na faixa (A), dito ser $\tilde{\mathbf{b}}_R = \mathbf{b} + \delta\mathbf{b}_R$ onde $\delta\mathbf{b}_R = \mathbf{A}\delta\mathbf{x}$ para um único n-vetor $\delta\mathbf{x}$:

$$\tilde{\mathbf{b}}_R = \mathbf{b}_R + \delta\mathbf{b}_R = \mathbf{b}_R \mathbf{A}\delta\mathbf{x} \qquad \text{e} \qquad \tilde{\mathbf{b}}_N = \mathbf{b}_N$$

Segue da equação (B-20) que a solução MMQL $\tilde{\mathbf{x}}$ e o resíduo $\tilde{\rho}$ correspondentes a $\tilde{\mathbf{b}}$ satisfaz

$$\tilde{\mathbf{x}} = \mathbf{x} + \delta\mathbf{x}, \qquad \mathbf{e} \qquad \tilde{\rho} = \rho$$

desse modo mostrando que mudanças em **b** na faixa (**A**) alteram a solução mas não o resíduo.

Ao contrário, suponha que **b** seja perturbado por um vetor que esteja inteiramente no espaço nulo de $\mathbf{A}^T$, tal que $\bar{\mathbf{b}} = \mathbf{b} + \mathbf{z}$, onde $\mathbf{A}^T z = 0$, de tal modo que $\bar{\mathbf{b}}_R = \mathbf{b}_R$ e $\bar{\mathbf{b}}_N = \mathbf{b}_N + \mathbf{z}$. Neste caso, (B-20) mostra que a solução do MMQL correspondente, $\bar{\mathbf{x}}$ e o resíduo são

$$\bar{\mathbf{x}} = \mathbf{x} \qquad \text{e} \qquad \bar{\rho} = \rho + \mathbf{z}$$

Quando **b** é mudado por uma perturbação **z** (de qualquer tamanho) no espaço nulo de $\mathbf{A}^T$, o resíduo é mudado por **z**, mas a solução permanece inalterada.

EQUAÇÕES NORMAIS

A mais significante propriedade do resíduo ótimo do MMQL é que ele está inteiramente no NULL ($\mathbf{A}^T$). Em termos algébricos, a solução **x** do MMQL satisfaz

$$\mathbf{A}^T(\mathbf{b} - \mathbf{Ax}) = \mathbf{0}$$

ver (B-19). O rearranjo trivial leva à famosa equação normal:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \qquad \text{(B-21)}$$

A matriz simétrica $\mathbf{A}^T\mathbf{A}$ (equação normal da matriz) é positiva e semidefinida para qualquer matriz **A**, e é definida positiva se e somente se as colunas de **A** são linearmente independentes. As equações normais são sempre compatíveis, i. e., a solução (B-20) existe sempre que $\mathbf{A}^T\mathbf{A}$ é singular. Qualquer solução do MMQL deve satisfazer às equações normais, e qualquer vetor **x** que satisfaça às equações normais deve ser solução do MMQL.

As equações normais são de grande importância prática porque elas oferecem um caminho direto para calcular a solução do MMQL. Quando $\mathbf{A}^T\mathbf{A}$ é definido positivo, as equações normais têm uma solução única. O problema de MMQL pode então ser resolvido com o seguinte algoritmo:

Algoritmo de resolução de um problema de MMQL com matriz de posto completo com equações normais.

1. Formar a matriz da equação normal $\mathbf{A}^T\mathbf{A}$ e o vetor $\mathbf{A}^T\mathbf{b}$.
2. Computar a fatorização Cholesky $\mathbf{A}^T\mathbf{A} = \mathbf{R}^T\mathbf{R}$, onde **R** é triagular superior.
3. Resolver os dois sistemas triangulares $\mathbf{R}^T\mathbf{y} = \mathbf{A}^T\mathbf{b}$ e $\mathbf{R}\mathbf{x} = \mathbf{y}$, o vetor **x** é a solução desejada.

Freqüentemente, se $\mathbf{A}^T\mathbf{A}$ é singular, uma solução as equações normais pode ser computada usando-se uma versão da fatorização Cholesky que inclui intermudanças simétricas.

Infelizmente a utilização de equações normais tem características indesejáveis com respeito à estabilidade numérica por causa do número da condição de $\mathbf{A}^T\mathbf{A}$ que é o quadrado da condição de **A**. Conseqüentemente, a equação normal é grandemente mal-condicionada quando A é somente moderadamente mal-condicionada.

O mal condicionamento na matriz da equação normal pode levar não somente à imprecisão na solução computada da equação normal mas também a perda de informação quando o posto de A é marginal.

TRANSFORMAÇÕES ORTOGONAIS

Por causa das possíveis dificuldades numéricas usando as equações normais, modernos métodos de resolução do MMQL foram desenvolvidos baseados em tranformações ortogonais, os quais preservam o comprimento Euclidiano e não piora a condição de **A**. Como no tratamento de sistemas não singulares e não compatíveis, a intenção é transformar o problema de MMQL tal que ele se torne fácil de ser resolvido – neste caso, pela redução de **A** à uma forma reveladora do posto, como por exemplo fatoração LU, SVD, QR.

Transformação ortogonal é uma escolha óbvia porque não altera a norma dois.

Faça **A** ser uma matriz m x n não zero com posto r. Suponha que uma matriz ortogonal Q possa ser encontrada tal que produza uma forma reveladora do posto

$$\mathbf{Q}^T\mathbf{A} = \begin{pmatrix} \mathrm{F} \\ \mathbf{0} \end{pmatrix} \qquad \text{(B-22)}$$

onde **F** é r x n e tem linhas linearmente independentes, o bloco de zeros, completando a matriz, estará ausente se r = m. Em situações de interesse, **F** é usualmente um triângulo superior com colunas permutadas. A forma (B-22) permite resolver o problema de MMQL usando as matrizes **Q** e **F**. Faça **d** ser o vetor transformado

$$\mathbf{d} = \mathbf{Q}^T\mathbf{b} = \begin{pmatrix} \mathbf{d}_r \\ \mathbf{d}_{m-r} \end{pmatrix} \begin{matrix} \}\mathbf{r} \\ \}\mathbf{m} \quad \mathbf{-r} \end{matrix}$$

Usando esta definição e a estrutura mostrada em (B-22), pode-se escrever os resíduos transformados como

$$\mathbf{Q}^T(\mathbf{b} - \mathbf{Ax}) = \mathbf{Q}^T\mathbf{b} - \mathbf{Q}^T\mathbf{Ax} = \begin{pmatrix} \mathbf{d}_r \\ \mathbf{d}_{m-r} \end{pmatrix} - \begin{pmatrix} \mathbf{Fx} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{d}_r - \mathbf{Fx} \\ \mathbf{d}_{m-r} \end{pmatrix}$$

Porque $\mathbf{Q}^T$ é ortogonal, sua aplicação ao resíduo não altera o comprimento Euclidiano, e

$$\|\mathbf{b} - \mathbf{Ax}\|_2^2 = \|\mathbf{Q}^T(\mathbf{b} - \mathbf{Ax})\|_2^2 \qquad \text{(B-23)}$$

Combinando-se esta relação e a forma especial do resíduo transformado, conclui-se que

$$\|\mathbf{b} - \mathbf{Ax}\|_2^2 = \|\mathbf{Q}^T(\mathbf{b} - \mathbf{Ax})\|_2^2$$
$$\|\mathbf{b} - \mathbf{Ax}\|_2^2 = \|\mathbf{d}_r - \mathbf{Fx}\|_2^2 + \|\mathbf{d}_{m-r}\|_2^2$$
$$\|\mathbf{b} - \mathbf{Ax}\|_2^2 \geq \|\mathbf{d}_{m-r}\|_2^2$$

o que significa que $\|\mathbf{b} - \mathbf{Ax}\|_2$ não pode ser menor que $\|\mathbf{d}_{m-r}\|_2$ para qualquer escolha de **x**.

O menor valor possível de $\|\mathbf{b} - \mathbf{Ax}\|_2^2$ é sua banda inferior. Igualdade na banda inferior ocorre se e somente se **x** satisfaz

$\mathbf{d}_r - \mathbf{Fx} = \mathbf{0}$, ou equivalentemente, se $\mathbf{Fx} = \mathbf{d}_r$ (B-24)

Desde que o posto de F é igual a r, o sistema $\mathbf{Fx} = \mathbf{d}_r$ deve ser compatível e p resíduo transformado satisfaz a

$$\mathbf{Q}^T(\mathbf{b} - \mathbf{Ax}) = \begin{pmatrix} \mathbf{0} \\ \mathbf{d}_{m-r} \end{pmatrix} \text{ e } \qquad \|\mathbf{b} - \mathbf{Ax}\|_2 = \|\mathbf{d}_{m-r}\|_2$$

Esta discussão mostra que o vetor **x** satisfazendo $\mathbf{Fx} = \mathbf{d}_r$ é uma solução do MMQL.

Sistemas construídos envolvendo **F** são fáceis de se resolver, segue que o problema MMQL pode ser resolvido encontrando uma matriz ortogonal $\mathbf{Q}^T$ que siga a forma (B-22).

A FATORIZAÇÃO QR

O processo da redução de Householder com mudanças entre as colunas, transforma uma matriz **A** de posto r e, uma matriz triangular superior mostrando sua forma reveladora do posto. A fatorização resultante pode ser escrita como

$$\mathbf{Q}^T\mathbf{A} = \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}\mathbf{P}^T = \begin{pmatrix} \mathbf{RP}^T \\ \mathbf{0} \end{pmatrix} \qquad \text{(B-25)}$$

onde $\mathbf{Q}^T$ representa r transformações linearmente independentes, tal que a matriz transformada $\mathbf{Q}^T\mathbf{A}$ tem a forma (B-25) desejada com $\mathbf{F} = \mathbf{RP}^T$. O MMQL pode então ser resolvido com o seguinte procedimento:

Algoritmo de resolução de problemas MMQL com o uso da fatorização QR.

1. Calcular a fatorização QR de A usando redução Householder com mudanças nas colunas. O procedimento Householder determina r (posto de A) e as matrizes P, Q e R.
2. Formar $\mathbf{d} = \mathbf{Q}^T\mathbf{b}$, e fazer $\mathbf{d}_r$ denotar os primeiros r componentes de **d**.
3. Calcular qualquer solução **y** do sistema $\mathbf{Ry} = \mathbf{d}_r$, onde $\mathbf{d}_r$ denota os r primeiros componente de **d**.
4. Formar $\mathbf{x} = \mathbf{Py}$.

Quando **A** tem colunas linearmente independentes, a matriz **R** é não singular, a solução **y** de $\mathbf{Ry} = \mathbf{d}_r$ é única e a solução do MMQL é única. Neste caso, resolvendo o problema de MMQL os QR fatores são duas vezes mais trabalhosos em termos de dificuldade que as equações normais.

Quando as colunas de **A** são linearmente dependentes, tal que $r < n$, o sistema $\mathbf{Ry} = \mathbf{d}_r$ (r x n) tem um número infinito de soluções. Recordar que **R** tem a forma

$$\mathbf{R} = (\mathbf{R}_{11}\ \mathbf{R}_{12})$$

onde $\mathbf{R}_{11}$ é uma matriz triangular superior r x r. A presença da matriz não singular $\mathbf{R}_{11}$ (r x r) no lado esquerdo de **R** sugere um vetor **y** satisfazendo $\mathbf{Ry} = \mathbf{d}_r$, a saber

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_B \\ \mathbf{0} \end{pmatrix}, \text{ com} \qquad \mathbf{R}_{11}\mathbf{y}_B = \mathbf{d}_r \qquad \text{(B-26)}$$

Quando **y** tem a forma da equação acima, (B-26), o vetor $\mathbf{x} = \mathbf{Py}$, associado é simplesmente um rearranjo de **y** e é chamado uma solução básica do problema de MMQL.

A SOLUÇÃO PELA FATORIZAÇÃO LU

A forma da matriz reveladora do posto para a fatorização LU é

$$\mathbf{A} = \mathbf{GH}\text{, com} \qquad \mathbf{G} = \tilde{\mathbf{P}}^T\mathbf{L}, \qquad \mathbf{H} = \mathbf{U}\mathbf{P}^T,$$

onde $\tilde{\mathbf{P}}$ e $\mathbf{P}$ são matrizes permutação, $\mathbf{L}$ é uma matriz triangular unitária inferior (m x r) e $\mathbf{U}$ é uma matriz triangular superior (r x n). Desde que $\tilde{\mathbf{P}}$ e $\mathbf{P}$ são ortogonais, as matrizes $\mathbf{G}^T\mathbf{G}$ e $\mathbf{HH}^T$ são dadas por $\mathbf{G}^T\mathbf{G} = \mathbf{L}^T\mathbf{L}$ e $\mathbf{HH}^T = \mathbf{UU}^T$.

Por substituição de matrizes particulares, tem-se

$$\mathbf{L}^T\mathbf{Ls} = \mathbf{L}^T\mathbf{Pb}\text{, ou } \mathbf{s} = \left(\mathbf{L}^T\mathbf{L}\right)^{-1}\mathbf{L}^T\tilde{\mathbf{P}}\mathbf{b} \tag{B-27}$$

e

$$\mathbf{b}_R = \tilde{\mathbf{P}}^T\mathbf{L}\left(\mathbf{L}^T\mathbf{L}\right)^{-1}\mathbf{L}^T\tilde{\mathbf{P}}\mathbf{b} \tag{B-28}$$

A SOLUÇÃO PELA DECOMPOSIÇÃO DO VALOR SINGULAR-(SVD)

Devido às particularidades das matrizes que precisaram ser invertidas, nem sempre foi possível utilizar métodos como o LU, portanto, o SVD foi utilizado como uma poderosa ferramenta para resolução de equações matriciais onde as matrizes a serem invertidas não eram quadradas.

O métdodo SVD é baseado no seguinte teorema da álgebra: "Qualquer matriz m x n
cujo número de linhas m é maior ou igual ao número de colunas n, pode ser escrito como o produto de uma matriz-coluna ortogonal **U**, e uma matriz **W** com elementos positivos ou zero (os valores singulares) e a transposta de uma n x n matriz ortogonal **V**.

$$\begin{pmatrix} \\ \mathbf{A} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \mathbf{U} \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \mathbf{w}_1 & & & \\ & \mathbf{w}_2 & & \\ & & \cdots & \\ & & \cdots & \\ & & & \mathbf{w}_n \end{pmatrix} \cdot \begin{pmatrix} \\ \mathbf{V}^T \\ \\ \end{pmatrix} \tag{B-29}$$

As matrizes **U** e **V** são ortigonais visto que suas colunas são ortonormais.

$$\sum_{i=1}^{m} \mathbf{U}_{ik}\mathbf{U}_{in} = \delta_{kn} \qquad \begin{cases} 1 \le k \le n \\ 1 \le n \le n \end{cases} \tag{B-30}$$

$$\sum_{j=1}^{n} \mathbf{V}_{jk}\,\mathbf{V}_{jn} = \delta_{kn}, \qquad \begin{cases} \mathbf{1 \le k \le n} \\ \mathbf{1 \le n \le n} \end{cases} \tag{B-31}$$

SVD PARA MAIS EQUAÇÕES QUE INCÓGNITAS

Dada uma conjunto de equções em número maior que o de incógnitas a serem descobertas, não existe uma solução única e isto poderá influenciar em demasia os resultados esperados. A boa condicionabilidade da matriz é de extrema importância para que se obtenha a resposta o mais próximo possível da realidade. Deste modo, dada uma matriz **A** nas condições acima descritas, como parte de um sistema linear da forma **Ax** = **b**, tem-se que a solução **x** do mesmo é dada por

$$\left(\mathbf{x}\right) = \left(\quad \mathbf{V} \quad\right)\cdot\left(\quad \mathbf{diag}(1/w_j) \quad\right)\cdot\left(\quad \mathbf{U}^T \quad\right)\cdot\left(\mathbf{b}\right)$$

B.3 RESOLUÇÃO DO PROBLEMA DE QP-MÉTODOS LAGRANGIANOS

Como problemas de programação linear, um outro problema de otimização linear o qual pode ser resolvido em um número finito de passos é o problema de programação quadrática (QP). Este é um problema no qual a função objetivo **f(x)** é quadrática e as funções restrições são funções lineares $\mathbf{c}_i(\mathbf{x})$. Assim, o problema é encontrar uma solução $\mathbf{x}^*$ para o problema geral

$$\underset{\mathbf{x}}{\textbf{minimizar}} \quad q(\mathbf{x}) \overset{\Delta}{=} \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{g}^T\mathbf{x} \tag{B-32}$$

$$\textbf{sujeito a} \quad \mathbf{a}_i^T = \mathbf{b}_i, \quad \mathbf{i} \in \mathbf{E}$$

$$\mathbf{a}_i^T\mathbf{x} \ge \mathbf{b}_i, \quad \mathbf{i} \in \mathbf{I}$$

onde é sempre possível arranjar a matriz **G** que seja simétrica.

Este ítem tem como meta descrever a maneira de se encontrar uma solução x^* para o problema com restrições de igualdade

$$\underset{\mathbf{x}}{\textbf{minimizar}} \quad q(\mathbf{x}) \overset{\Delta}{=} \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{g}^T\mathbf{x} \tag{B-33}$$

$$\textbf{sujeito a} \quad \mathbf{A}^T\mathbf{x} = \mathbf{b}$$

Assume-se que há m $\leq$ n restrições tal que $\mathbf{b} \in \Re^m$, e **A** é n x m. Assume-se também que **A** tem posto m. Estas considerações também asseguram que o multiplicador de Lagrange (ML) existe.

MÉTODOS DE LAGRANGE

O método de Lagrange é um caminho alternativo para se chegar à solução x^* das equações do problema de QP associando-as aos multiplicadores de Lagrange. Tem-se que da definição dos multiplicadores de Lagrange, seja dada uma função

$$F(x_1, ..., x_n) = f(x_1, ..., x_n) + \lambda_1\phi_1(x_1, ..., x_n) + \lambda_2\phi_2(x_1, ..., x_n) + ... + \lambda_n\phi_n(x_1, ..., x_n)$$

onde os ϕ's são as expressões que que definem as N restrições e os λ's são (Multiplicadores de Lagrange) os parâmetros desconhecidos. Então, as primeiras derivadas parciais de F são calculadas e o conjunto igualado a zero, dando

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}_1} = \mathbf{0}, \qquad \frac{\partial \mathbf{F}}{\partial \mathbf{x}_2} = \mathbf{0}, \qquad \frac{\partial \mathbf{F}}{\partial \mathbf{x}_3} = \mathbf{0}, \ ..., \ \frac{\partial \mathbf{F}}{\partial \mathbf{x}_n} = \mathbf{0}$$

Finalmente, resolvendo estas equações (n), juntas com as N equações de restrição

$$\phi_1 = \mathbf{0}, \qquad \phi_2 = \mathbf{0}, \qquad \phi_3 = \mathbf{0}, \qquad \phi_n = \mathbf{0}.$$

À primeira vista, a utilização dos ML pode parecer ineficiente, uma vez que isto requer a resolução de n + N equações simultâneas, mais que as n equações anteriores, portanto. Entretanto, é freqüentemente o processo mais conveniente, desde que não requer a decisão de quais variáveis são consideradas independentes e freqüentemente permite aproveitar a vantagem de simetria das variáveis.

Para o caso de duas restrições suponha $f(x_1, x_2, x_3)$ para se determinar o extremo e $\phi_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{0}$ e $\phi_2(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$

pelo método dos múltiplicadores de Lagrange, tem-se

$$\mathbf{F} = \mathbf{f} + \lambda_1\phi_1 + \lambda_2\phi_2$$

$$\mathbf{dF} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_1} + \lambda_1\frac{\partial \phi_1}{\partial \mathbf{x}_1} + \lambda_2\frac{\partial \phi_2}{\partial \mathbf{x}_1}\right)\mathbf{dx}_1 + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_2} + \lambda_1\frac{\partial \phi_1}{\partial \mathbf{x}_2} + \lambda_2\frac{\partial \phi_2}{\partial \mathbf{x}_2}\right)\mathbf{dx}_2 +$$
$$+ \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_3} + \lambda_1\frac{\partial \phi_1}{\partial \mathbf{x}_3} + \lambda_2\frac{\partial \phi_2}{\partial \mathbf{x}_3}\right)\mathbf{dx}_3$$

como condição necessária para se ter os extremos. A partir disto, desde que x_1, x_2, e x_3 são agora considerados variáveis independentes, infere-se que

Aplicando-se a definição à equação (B-32) ou (B-33), tem-se

$$\hbar(\mathbf{x},\mathbf{y}) = \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{g}^T\mathbf{x} - \lambda\left(\mathbf{A}^T\mathbf{x} - \mathbf{b}\right) \tag{B-34}$$

onde $\hbar(\mathbf{x},\mathbf{y})$ representa a função Lagrangiano.
desenvolvendo-se a equção ,tem-se que a derivada da equação (B-34) em relação a $\mathbf{x}$ e a λ dá

$$\nabla_x\hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_x\left(\mathbf{x}^T\mathbf{G}\mathbf{x}\right)\right) + \nabla_x\left(\mathbf{g}^T\mathbf{x}\right) - \nabla_x\left(\lambda^T\left(\mathbf{A}^T\mathbf{x} - \mathbf{b}\right)\right)$$

por FLETCHER(1987), tem-se

$$\nabla\left(\mathbf{U}^T\mathbf{V}\right) = \left(\nabla\mathbf{U}^T\right)\mathbf{V} + \left(\nabla\mathbf{V}^T\right)\mathbf{U}$$

assim,

$$\nabla_x\hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_x\left(\mathbf{x}^T\right)\mathbf{G}\mathbf{x} + \nabla_x\left(\mathbf{G}\mathbf{x}\right)^T\mathbf{x}\right) + \nabla_x\left(\mathbf{g}^T\right)\mathbf{x} + \nabla_x\left(\mathbf{x}^T\right)\mathbf{g} - \nabla_x\left(\lambda^T\right)\mathbf{A}^T\mathbf{x} - \nabla_x\left(\mathbf{A}^T\mathbf{x}\right)^T\lambda$$

$$\nabla_x\hbar = \mathbf{0} = \frac{1}{2}\left(1.\mathbf{G}\mathbf{x} + 1.\mathbf{G}\mathbf{x}\right) + 1.\mathbf{g} - \mathbf{A}\lambda$$
$$\nabla_x\hbar = \mathbf{0} = \mathbf{G}\mathbf{x} + \mathbf{g} - \mathbf{A}\lambda$$

De maneira análoga para λ , tem-se

$$\nabla_\lambda\hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_\lambda\left(\mathbf{x}^T\right)\mathbf{G}\mathbf{x} + \nabla_\lambda\left(\mathbf{G}\mathbf{x}\right)^T\mathbf{x}\right) + \nabla_\lambda\left(\mathbf{g}^T\right)\mathbf{x} + \nabla_\lambda\left(\mathbf{x}^T\right)\mathbf{g} - \nabla_\lambda\left(\lambda^T\right)\mathbf{A}^T\mathbf{x} - \nabla_\lambda\left(\mathbf{A}^T\mathbf{x}\right)^T\lambda$$

$$\nabla_\lambda\hbar = \mathbf{0} = -\nabla_\lambda\left(\lambda^T\right)\mathbf{A}^T\mathbf{x} - \nabla_\lambda\left(\mathbf{A}^T\mathbf{x}\right)^T\lambda + \nabla_\lambda\lambda^T\mathbf{b}$$
$$\nabla_\lambda\hbar = \mathbf{0} = -\mathbf{A}^T\mathbf{x} + \mathbf{b}$$
$$\mathbf{A}^T\mathbf{x} - \mathbf{b} = \mathbf{0}$$

e os pontos estacionários resultam em

$$\nabla_x\hbar = \mathbf{0}: \quad \mathbf{G}\mathbf{x} + \mathbf{g} - \mathbf{A}\lambda = \mathbf{0}$$
$$\nabla_\lambda\hbar = \mathbf{0}: \quad \mathbf{A}^T\mathbf{x} - \mathbf{b} = \mathbf{0}$$

os quais podem ser rearranjados na forma do sistema linear

$$\begin{bmatrix} \mathbf{G} & -\mathbf{A} \\ -\mathbf{A}^T & \mathbf{0} \end{bmatrix}\begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = -\begin{pmatrix} \mathbf{g} \\ \mathbf{b} \end{pmatrix} \tag{B-35}$$

A matriz coeficiente é referida como sendo a matriz Lagrangiana e é simétrica mas não é definida como positiva. Se a inversa existe e é expressa como

$$\begin{bmatrix} \mathbf{G} & -\mathbf{A} \\ -\mathbf{A}^T & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{H} & -\mathbf{T} \\ -\mathbf{T}^T & \mathbf{U} \end{bmatrix} \tag{B-36}$$

então a solução para a equção (B-35) pode ser escrita como

$$\mathbf{x}^* = -\mathbf{Hg} + \mathbf{Tb} \tag{B37a}$$
$$\lambda^* = \mathbf{T}^T\mathbf{g} - \mathbf{Ub} \tag{B37b}$$

B.4 DEDUÇÃO DOS AJUSTES DA RECONCILIAÇÃO DE DADOS

CASO LINEAR

Dado o problema de ajuste para o caso linear, tem-se que usando os multiplicadores de Lagrange é possível encontrar o ponto estacionário $(\mathbf{a}_*, \lambda_*)$ do Lagrangiano

$$\hbar(\mathbf{a}, \lambda) \equiv \frac{1}{2}\mathbf{F}(\mathbf{a}) - \lambda^T\mathbf{Y}^T\left[\mathbf{B}_0\mathbf{c} + \mathbf{B}_1(\tilde{\mathbf{x}} + \mathbf{a})\right] \tag{B-38}$$

tirando-se a dervida de (B-38) em relação a $\mathbf{a}$ e a λ, tem-se

$$\nabla_a\hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_a\left(\mathbf{a}^T\Sigma^{-1}\mathbf{a}\right)\right) - \nabla_a\left(\lambda^T\mathbf{Y}^T\left[\mathbf{B}_0\mathbf{b}_0 + \mathbf{B}_1(\tilde{\mathbf{x}} + \mathbf{a})\right]\right)$$
$$\nabla_a\hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_a\mathbf{a}^T\left(\Sigma^{-1}\mathbf{a}\right)\right) + \frac{1}{2}\left(\nabla_a\left(\Sigma^{-1}\mathbf{a}\right)^T\mathbf{a}\right) + \nabla_a\left(\lambda^T\mathbf{Y}^T\mathbf{B}_0\mathbf{b}_0\right) + \nabla_a\left(\lambda^T\mathbf{Y}^T\mathbf{B}_1(\tilde{\mathbf{x}} + \mathbf{a})\right)$$
$$\nabla_a\hbar = \mathbf{0} = \Sigma^{-1}\mathbf{a} + \nabla_a\left(\lambda^T\mathbf{Y}^T\mathbf{B}_1\tilde{\mathbf{x}}\right) + \nabla_a\left(\lambda^T\mathbf{Y}^T\mathbf{B}_1\mathbf{a}\right)$$
$$\nabla_a\hbar = \mathbf{0} = \Sigma^{-1}\mathbf{a} + \mathbf{0} + \nabla_a\left(\left(\lambda^T\mathbf{Y}^T\mathbf{B}_1\right)^T\right)^T\mathbf{a} + \nabla_a(\mathbf{a})^T\left(\left(\lambda^T\mathbf{Y}^T\mathbf{B}_1\right)^T\right)$$
$$\nabla_a\hbar = \mathbf{0} = \Sigma^{-1}\mathbf{a} + \mathbf{0} + \mathbf{0} + \mathbf{1}.\mathbf{B}_1^T\mathbf{Y}\lambda$$
$$\Sigma^{-1}\mathbf{a} = \mathbf{B}_1^T\mathbf{Y}\lambda$$

$\mathbf{a}_* = \Sigma\mathbf{B}_1^T\mathbf{Y}\lambda_*$, que é o primeiro ajuste desejado.

em relação a λ, vem

$$\nabla_\lambda\hbar = \mathbf{0} = \underbrace{\frac{1}{2}\left(\nabla_\lambda\left(\mathbf{a}^T\Sigma^{-1}\mathbf{a}\right)\right)}_{\text{I}} \quad \underbrace{-\nabla_\lambda\left(\lambda^T\mathbf{Y}^T\left[\mathbf{B}_0\mathbf{b}_0 + \mathbf{B}_1(\tilde{\mathbf{x}} + \mathbf{a})\right]\right)}_{\text{II}}$$

Para I,

$$\nabla_\lambda\hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_\lambda\left(\left(\Sigma\mathbf{B}_1^T\mathbf{Y}\lambda\right)^T\Sigma^{-1}\Sigma\mathbf{B}_1^T\mathbf{Y}\lambda\right)\right) + \ldots \text{II}$$

$$\nabla_{\lambda}\hbar = \mathbf{0} = \frac{\mathbf{1}}{\mathbf{2}}\left(\nabla_{\lambda}\lambda^{T}\left(\left(\sum \mathbf{B}_1^{T}\mathbf{Y}\right)^{T}\mathbf{B}_1^{T}\mathbf{Y}\lambda\right) + \nabla_{\lambda}\left(\left(\sum \mathbf{B}_1^{T}\mathbf{Y}\right)^{T}\mathbf{B}_1^{T}\mathbf{Y}\lambda\right)^{T}.\lambda \right) + \ldots \mathbf{II}$$

$$\nabla_{\lambda}\hbar = \mathbf{0} = \frac{\mathbf{1}}{\mathbf{2}}\left(\left(\left(\sum \mathbf{B}_1^{T}\mathbf{Y}\right)^{T}\mathbf{B}_1^{T}\mathbf{Y}\lambda\right) + \nabla_{\lambda}\left(\left(\sum \mathbf{B}_1^{T}\mathbf{Y}\right)^{T}\mathbf{B}_1^{T}\mathbf{Y}\lambda\right)^{T}.\lambda \right) + \ldots \mathbf{II}$$

$$\nabla_{\lambda}\hbar = \mathbf{0} = \frac{\mathbf{1}}{\mathbf{2}}\left(\mathbf{Y}^{T}\mathbf{B}_1 \sum^{T} \mathbf{B}_1^{T}\mathbf{Y}\lambda + \left(\left(\sum \mathbf{B}_1^{T}\mathbf{Y}\right)^{T}\mathbf{B}_1^{T}\mathbf{Y}\right)^{T}\lambda \right) + \ldots \mathbf{II}$$, aproveitando a simetria de $\sum$, vem

$$\nabla_{\lambda}\hbar = \mathbf{0} = \frac{\mathbf{1}}{\mathbf{2}}\left(2\mathbf{Y}^{T}\mathbf{B}_1^{T} \sum^{T} \mathbf{B}_1^{T}\mathbf{Y}\lambda\right) + \ldots \mathbf{II}$$

Para II,

$$\nabla_{\lambda}\hbar = \mathbf{0} = \mathbf{I} + \ldots \nabla_{\lambda}\left(\lambda^{T}\mathbf{Y}^{T}\left(\mathbf{B}_0\mathbf{b}_0 + \mathbf{B}_1(\tilde{\mathbf{x}} + \mathbf{a})\right)\right)$$

$$\nabla_{\lambda}\hbar = \mathbf{0} = \mathbf{I} + \ldots \frac{\mathbf{1}}{\mathbf{2}}\nabla_{\lambda}\left(\lambda^{T}\right)\left(\mathbf{Y}^{T}\left(\mathbf{B}_0\mathbf{b}_0 + \mathbf{B}_1\tilde{\mathbf{x}} + \mathbf{B}_1 \sum \mathbf{B}_1^{T}\mathbf{Y}\lambda\right)\right) + \frac{\mathbf{1}}{\mathbf{2}}\nabla_{\lambda}\left(\mathbf{Y}^{T}\left(\mathbf{B}_0\mathbf{b}_0 + \mathbf{B}_1\tilde{\mathbf{x}} + \mathbf{B}_1 \sum \mathbf{B}_1^{T}\mathbf{Y}\right)\right)$$

$$\nabla_{\lambda}\hbar = \mathbf{0} = \mathbf{I} + \ldots \frac{\mathbf{1}}{\mathbf{2}}\mathbf{Y}^{T}\left(\left(\mathbf{B}_0\mathbf{b}_0 + \mathbf{B}_1\tilde{\mathbf{x}} + \mathbf{B}_1 \sum \mathbf{B}_1^{T}\mathbf{Y}\lambda\right)\right) + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{Y}^{T}\left(\left(\mathbf{B}_0\mathbf{b}_0 + \mathbf{B}_1\tilde{\mathbf{x}} + \mathbf{B}_1 \sum^{T} \mathbf{B}_1^{T}\mathbf{Y}\lambda\right)\right)$$

fazendo $\mathbf{H} = \mathbf{Y}^{T}\mathbf{B}_1 \sum \mathbf{B}_1^{T}\mathbf{Y}\lambda$, e rearranjando os termos, tem-se

$$\mathbf{H}\lambda = -\mathbf{Y}^{T}\mathbf{B}_1 \sum \mathbf{B}_1^{T}\mathbf{Y}$$

que é o segundo termo a ser determinado.

CASO NÃO-LINEAR

Os parâmetros a serem determinados para o caso não-linear podem ser obtidos como no caso linear da seguinte maneira:

Dada a equação

$$\underset{\mathbf{a},(\mathbf{N}\delta)}{\mathbf{Min}} \quad \mathbf{G}(\mathbf{a},\mathbf{N}\delta) = \mathbf{a}^{T} \sum_1^{-1} a + (\mathbf{N}\delta)^{T} \sum_2^{-} (\mathbf{N}\delta)$$

Sujeita a

$$\mathbf{Z}^{T}\mathbf{Y}^{T}\left[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1(\tilde{\mathbf{x}} + \mathbf{a}) + \mathbf{B}_2\mathbf{N}\delta\right] = \mathbf{0}$$

Aplicando-se o operador Lagrangiano, tem-se

$$\hbar(\mathbf{a},\mathbf{N}\delta.\lambda) = \left[\mathbf{a}^{T} \sum_1^{-1} \mathbf{a} + \overset{\wedge}{(\mathbf{N}\delta)}^{T} \sum_2^{-1} (\mathbf{N}\delta)\right] - \lambda^{T}\mathbf{Z}^{T}\mathbf{Y}^{T}\left[\mathbf{B}_0\mathbf{k} + \mathbf{B}_1\left(\tilde{\mathbf{X}} + \mathbf{a}\right) + \mathbf{B}_2(\mathbf{N}\delta)\right]$$

Derivando-se em relação a cada um dos três parâmetros. vem

$$\nabla_a \hbar = \mathbf{0} = \underbrace{\frac{1}{2}\left(\nabla_a \mathbf{a}^T \Sigma_1^{-1} \mathbf{a} + \nabla_a (\mathbf{N}\delta)^T \Sigma_2^{-1}(\mathbf{N}\delta)\right)}_{\mathrm{I}} \underbrace{-\nabla_a \left(\lambda^T \mathbf{Z}^T \mathbf{Y}^T (\mathbf{B}_1 \mathbf{a})\right)}_{\mathrm{II}}$$

os demais termos para a derivada em relação a a forma suprimidos por serem nulos.

Para I,

$$\nabla_a \hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_a \mathbf{a}^T \left(\Sigma_1^{-1} \mathbf{a}\right) + \nabla_a \left(\Sigma_1^{-1} \mathbf{a}\right)^T \mathbf{a}\right) + \frac{1}{2}\left(\nabla_a (\mathbf{N}\delta)^T \left(\Sigma_2^{-1}(\mathbf{N}\delta)\right) + \nabla_a \left(\Sigma_2^{-1}(\mathbf{N}\delta)\right)^T \mathbf{N}\delta\right)$$

$$\nabla_a \hbar = \mathbf{0} = \frac{1}{2}\left(\mathbf{1}.\left(\Sigma_1^{-1} \mathbf{a}\right)\right) + \frac{1}{2}\nabla_a \left(\mathbf{a}^T \Sigma_1^{-T}\right).\mathbf{a} + \mathbf{0} + \mathbf{0} + \ldots \mathbf{II}$$

$\nabla_a \hbar = \mathbf{0} = \frac{1}{2}\Sigma_1^{-1} \mathbf{a} + \Sigma_1^{-T} \mathbf{a} + \ldots \mathbf{II}$, aproveitando a simetria de Σ , tem-se

$\nabla_a \hbar = \mathbf{0} = \Sigma_1^{-1} \mathbf{a} + \ldots \mathbf{II}$

Para II,

$$\nabla_a \hbar = \mathbf{0} = \mathbf{I} \ldots - \nabla_a \left(\lambda^T \mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_1 \mathbf{a}\right)$$

$$\nabla_a \hbar = \mathbf{0} = \mathbf{I} \ldots - \nabla_a \lambda^T \left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_1 \mathbf{a}\right) - \nabla_a \left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_1 \mathbf{a}\right)^T \lambda$$

$$\nabla_a \hbar = \mathbf{0} = \mathbf{I} \ldots - \mathbf{0} - \nabla_a \mathbf{a}^T \left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_1\right)^T \lambda$$

$$\nabla_a \hbar = \mathbf{0} = \mathbf{I} \ldots - \mathbf{B}_1^T \mathbf{Y} \mathbf{Z} \lambda$$

De I e II,

$$\mathbf{0} = \Sigma_1^{-1} \mathbf{a} - \mathbf{B}_1^T \mathbf{Y} \mathbf{Z} \lambda$$

$$\Sigma_1^{-1} \mathbf{a} = \mathbf{B}_1^T \mathbf{Y} \mathbf{Z} \lambda$$

$$\mathbf{a} = \Sigma_1 \mathbf{B}_1^T \mathbf{Y} \mathbf{Z} \lambda$$

o qual é o primeiro parâmetro a ser determinado na resolução do problema de reconciliação de dados nao-lineares.

$$\nabla_{\mathrm{N}\delta} \hbar = \mathbf{0} = \underbrace{\frac{1}{2}\left(\nabla_{\mathrm{N}\delta a} \mathbf{a}^T \Sigma_1^{-1} \mathbf{a} + \nabla_{\mathrm{N}\delta} (\mathbf{N}\delta)^T \Sigma_2^{-1}(\mathbf{N}\delta)\right)}_{\mathrm{I}} \underbrace{-\nabla_{\mathrm{N}\delta} \left(\lambda^T \mathbf{Z}^T \mathbf{Y}^T \left(\mathbf{B}_2(\mathbf{N}\delta)\right)\right)}_{\mathrm{II}},$$ os

demais termos suprimidos são nulos.

Para I,

$$\nabla_{\mathrm{N}\delta} \hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_{\mathrm{N}\delta} \mathbf{a}^T . \left(\Sigma_1^{-1} \mathbf{a}\right) + \nabla_{\mathrm{N}\beta\delta} \left(\Sigma_1^{-1} \mathbf{a}\right)^T . \mathbf{a}\right) + \frac{1}{2}\left(\nabla_{\mathrm{N}\delta} (\mathbf{N}\delta)^T \left(\Sigma_2^{-1}(\mathbf{N}\delta)\right) + \nabla_{\mathrm{N}\delta} \left(\Sigma_2^{-1}(\mathbf{N}\delta)\right)^T (\mathrm{N}\right.$$

$\nabla_{\mathrm{N}\delta} \hbar = \mathbf{0} = \frac{1}{2}(\mathbf{0} + \mathbf{0}) + \frac{1}{2}\left(\Sigma_2^{-1}(\mathbf{N}\delta) + \Sigma_2^{-T}(\mathbf{N}\delta)\right) + \ldots \mathbf{II}$, aproveitando a simetria de Σ

$$\nabla_{N\delta}\hbar = \mathbf{0} = \Sigma_2^{-1}(N\delta) + \ldots \mathbf{II}$$

Para II,

$$\nabla_{N\delta}\hbar = \mathbf{0} = \mathbf{I} - \ldots \nabla_{N\delta}\left(\lambda^T \mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_2 N\delta\right), \text{ os demais termos são nulos.}$$

$$\nabla_{N\delta}\hbar = \mathbf{0} = \mathbf{I} - \ldots \nabla_{N\delta}\left(\lambda^T\right)\left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_2 (N\delta)\right) + \nabla_{N\delta}\left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_2 (N\delta)\right)^T \lambda$$

$$\nabla_{N\delta}\hbar = \mathbf{0} = \mathbf{I} - \ldots \mathbf{0} + \nabla_{N\delta}(N\delta)^T\left(\left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_2\right)^T\right) + \nabla_{N\delta}\left(\left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_2\right)^T\right)^T (N\delta)$$

$$\nabla_{N\delta}\hbar = \mathbf{0} = \mathbf{I} - \ldots + \mathbf{0} + \left(\mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_2\right)^T + \mathbf{0}$$

juntando I e II, tem-se

$$\mathbf{0} = \Sigma_2^{-1}(N\delta) - \mathbf{B}_2^T \mathbf{Y}\mathbf{Z}\lambda$$

rearranjando,

$$N\delta = \Sigma_2 \mathbf{B}_2^T \mathbf{Y}\mathbf{Z}\lambda$$

o qual é o segundo parâmetro a ser determinado.

$$\nabla_\lambda \hbar = \mathbf{0} = \underbrace{\frac{1}{2}\nabla_\lambda\left[\mathbf{a}^T \Sigma_1^{-1} \mathbf{a} + (N\delta)^T \Sigma_2^{-1}(N\delta)\right]}_{\mathrm{I}} - \underbrace{\nabla_\lambda\left(\lambda^T \mathbf{Z}^T \mathbf{Y}^T\left[\mathbf{B}_0 \mathbf{k} + \mathbf{B}_1\left(\tilde{\mathbf{X}} + \mathbf{a}\right) + \mathbf{B}_2 (N\delta)\right]\right)}_{\mathrm{II}}$$

Para I,

$$\nabla_\lambda \hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_\lambda\left(\mathbf{a}^T \Sigma_1^{-1} \mathbf{a}\right) + \nabla_\lambda\left((N\delta)^T \Sigma_2^{-1}(N\delta)\right)\right), \text{ mas } \mathbf{a} \text{ e } N\delta \text{ são dependentes de } \lambda$$

$$\nabla_\lambda \hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_\lambda\left(\Sigma_1 \mathbf{B}_1^T \mathbf{Y}\mathbf{Z}\lambda\right)^T \Sigma_1^{-1} \Sigma_1 \mathbf{B}_1^T \mathbf{Y}\mathbf{Z}\lambda\right) + \frac{1}{2}\left(\nabla_\lambda\left(\Sigma_2 \mathbf{B}_2^T \mathbf{Y}\mathbf{Z}\lambda\right)^T . \Sigma_2^{-1}\left(\Sigma_2 \mathbf{B}_2^T \mathbf{Y}\mathbf{Z}\lambda\right)\right)$$

$$\nabla_\lambda \hbar = \mathbf{0} = \frac{1}{2}\left(\nabla_\lambda\left(\left(\Sigma_1 \mathbf{B}_1^T \mathbf{Y}\mathbf{Z}\lambda\right)^T \Sigma_1^{-1} \Sigma_1 \mathbf{B}_1^T \mathbf{Y}\mathbf{Z}\lambda\right)\right) + \frac{1}{2}\left(\nabla_\lambda\left(\left(\Sigma_2 \mathbf{B}_2^T \mathbf{Y}\mathbf{Z}\lambda\right)^T . \Sigma_2^{-1}\left(\Sigma_2 \mathbf{B}_2^T \mathbf{Y}\mathbf{Z}\lambda\right)\right)\right)$$

$$\nabla_\lambda \hbar = \mathbf{0} = \mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_1 \Sigma_1 \mathbf{B}_1^T \mathbf{Y}\mathbf{Z}\lambda + \mathbf{Z}^T \mathbf{Y}^T \mathbf{B}_2 \Sigma_2 \mathbf{B}_2^T \mathbf{Y}\mathbf{Z}\lambda$$

$$\mathbf{0} = \mathbf{Z}^T \mathbf{Y}^T\left(\mathbf{B}_1 \Sigma_1 \mathbf{B}_1^T + \mathbf{B}_2 \Sigma_2 \mathbf{B}_2^T\right)\mathbf{Y}\mathbf{Z}\lambda$$

fazendo $\mathbf{Y}^T\left(\mathbf{B}_1 \Sigma_1 \mathbf{B}_1^T + \mathbf{B}_2 \Sigma_2 \mathbf{B}_2^T\right)\mathbf{Y} = \mathbf{H}$

tem-se

$$\mathbf{Z}^T \mathbf{H}\mathbf{Z}\lambda = \mathbf{Z}^T$$

Para II,

$$\nabla_{\lambda}\hbar = \mathbf{0} = \mathbf{I}...+\nabla_{\lambda}\left[\lambda^{T}\mathbf{Z}^{T}\mathbf{Y}^{T}\left(\mathbf{B}_{0}\mathbf{k}+\mathbf{B}_{1}\left(\tilde{\mathbf{x}}+\mathbf{a}\right)+\mathbf{B}_{2}\mathbf{N}\delta\right)\right]$$

$$\nabla_{\lambda}\hbar = \mathbf{0} = \mathbf{I}...+\frac{\mathbf{1}}{\mathbf{2}}\nabla_{\lambda}\left(\lambda^{T}\mathbf{Z}^{T}\mathbf{Y}^{T}\mathbf{B}_{1}\left(\sum_{1}\mathbf{B}_{1}^{T}\mathbf{Y}\mathbf{Z}\lambda\right)\right)+\frac{\mathbf{1}}{\mathbf{2}}\nabla_{\lambda}\left(\lambda^{T}\mathbf{Z}^{T}\mathbf{Y}^{T}\mathbf{B}_{2}\sum_{2}\mathbf{B}_{2}^{T}\mathbf{Y}\mathbf{Z}\lambda\right)+$$

$$+\frac{\mathbf{1}}{\mathbf{2}}\left(\nabla_{\lambda}\left(\lambda^{T}\mathbf{Z}^{T}\mathbf{Y}^{T}\left(\mathbf{B}_{0}\mathbf{k}+\mathbf{B}_{1}\tilde{\mathbf{x}}\right)\right)\right)$$

desenvolvendo os três termos do lado direito da última equação, tem-se

$$-2\mathbf{Z}^{T}\mathbf{Y}^{T}\mathbf{B}_{1}\sum_{1}\mathbf{B}_{1}^{T}\mathbf{Y}\mathbf{Z}\lambda\,,$$

$$-2\mathbf{Z}^{T}\mathbf{Y}^{T}\mathbf{B}_{2}\sum_{2}\mathbf{B}_{2}^{T}\mathbf{Y}\mathbf{Z}\lambda$$

e

$$-\mathbf{Z}^{T}\mathbf{Y}^{T}\left(\mathbf{B}_{0}\mathbf{k}+\mathbf{B}_{1}\tilde{\mathbf{x}}\right),\ \text{respectivamente.}$$

rearrranjando os termos e somando as partes I e II, tem-se que

$$\lambda = \left(\mathbf{Z}^{T}\mathbf{H}\mathbf{Z}\right)^{-1}\mathbf{Z}^{T}\mathbf{Y}^{T}\left(\mathbf{B}_{0}\mathbf{k}+\mathbf{B}_{1}\tilde{\mathbf{x}}\right)$$

o qual é o valor a ser determinado.

B.5 PROJEÇÃO MATRICIAL

O conceito de projeção em um subespaço é crucial em otimização restringida, e está intimamente relacionada ao problema de MMQL.

Projeção em um subespaço

Faça S um subespaço de $\Re^{m}$. O complemento ortogonal de **S**, denotado por $\mathbf{S}^{\perp}$, é definido como um conjunto de todos os m-dimensionais vetores que são ortogonais aos vetores em **S**, isto é,

$$\mathbf{S}^{\perp} = \left\{\mathbf{t}\in\Re^{m}\middle|\mathbf{t}^{T}\mathbf{s} = \mathbf{0}\ \textbf{para}\ \mathbf{s}\in\mathbf{S}\right\}$$

É fácil mostrar que $\mathbf{S}^{\perp}$ é um subespaço de $\Re^{m}$. Juntos, **S** e $\mathbf{S}^{\perp}$ compreendem tudo de $\Re^{m}$, e **S** e $\mathbf{S}^{\perp}$ não contêm nenhum vetor em comum exceto o vetor zero:

$$\mathbf{S}\cup\mathbf{S}^{\perp} = \Re^{m}\ \mathbf{e}\ \mathbf{S}\cap\mathbf{S}^{T} = \{\mathbf{0}\}$$

Qualquer m-vetor não zero **b** pode ser unicamente representado como $\mathbf{b} = \mathbf{b}_{S} + \mathbf{b}_{S^{\perp}}$, onde $\mathbf{b}_{S}\in\mathbf{S}$.e

A matriz projeção denotada por $\mathbf{P}_S$ é a matriz, única, m x m com as seguintes três propriedades:

(i) todo vetor no subespaço **S** pode ser escrito como uma combinação linear das colunas de $\mathbf{P}_S$, i. é., o vetor $\mathbf{b}_S$ pertence a S se e somente se $\mathbf{b}_S = \mathbf{P}_S\mathbf{v}$ para algum m-vetor **v**;
(ii) $\mathbf{P}_S^T = \mathbf{P}_S$ ($\mathbf{P}_S$ é simétrica);
(iii) $\mathbf{P}_S^2 = \mathbf{P}_S$. (A matriz **P** tal que $\mathbf{P}^2 = \mathbf{P}$ é chamada idempotente)

O significado intuitivo do projetor é que é sua aplicação a qualquer m-vetor **b** produz $\mathbf{b}_S$, a porção de **b** que pertence a **S**, e consequentemente, aniquila, a parte de **b** que não pertence a **S**. Se **b** já está contido em **S**, $\mathbf{P}_S$ projeta **b** em si mesmo. Se **b** está inteiramente no complemento ortogonal $\mathbf{S}^\perp$, $\mathbf{P}_S$ transforma **b** no vetor zero. Apesar de $\mathbf{P}_S$ ser algumas vezes chamado de projetor ortogonal em **S**, é preciso enfatizar que $\mathbf{P}_S$ não é, em geral, uma matriz ortogonal. Desde que $\mathbf{P}_S\mathbf{t} = \mathbf{0}$ para qualquer vetor não nulo **t** em $\mathbf{S}^\perp$, $\mathbf{P}_S$ deve ser singular e conseqüentemente não pode ser ortogonal.

Uma relação importante existe entre o projetor no espaço **S** e no seu complemento ortogonal. Dado qualquer vetor $\mathbf{b} = \mathbf{b}_S + \mathbf{b}_{S^\perp}$, onde **b** **S** e $\mathbf{b}_{S^\perp} \in \mathbf{S}^\perp$, tem-se que

$\mathbf{P}_S\mathbf{b} = \mathbf{b}_S$, tal que $(\mathbf{I} - \mathbf{P}_S)\mathbf{b} = \mathbf{b} - \mathbf{b}_S = \mathbf{b}_{S^\perp}$

É possível mostrar de forma direta que $\mathbf{I} - \mathbf{P}_S$ satisfaz às propriedades (i)-(iii) do projetor sobre o complemento de **S**, tal que

$\mathbf{I} - \mathbf{P}_S$ é o projetor sobre $\mathbf{S}^\perp$.

C-LISTAGEM DOS PROGRAMAS-RECON E RETIF

```
##########################################################################################################
LISTAGEM DOS PROGRAMAS DE RECONCILIAÇÃO E DETECÇÃO DE ERROS GROSSEIROS EM PROCESSOS, UTILIZANDO LINGUAGEM C.
##########################################################################################################

##########################################################################################################
PROGRAMA 1- RECONCILIAÇÃO DE DADOS- RECON.C
##########################################################################################################

**********************************************************************************************************
ESTA LISTAGEM CORRESPONDE AO PROGRAMA DE RECONCILIACÃO DE DADOS DESENVOLVIDO EM LINGUAGEM C. ESTE PROGRAMA MOSTRA NA TELA AS RESPOSTAS DAS VARIÁVEIS RECONCILIADAS E CONSTRÓI UM ARQUIVO COM DADOS PARA A RETIFICAÇÃO DE DADOS. EXISTE TAMBÉM A POSSIBILIDADE DE SE RETIRAR OS COMENTÁRIOS FAZENDO O PROGRAMA SAIR COM UM ARQUIVO DE DADOS RECONCILIADOS. ISTO NAO FOI AQUI INCLUIDO POR MOTIVOS DE MEMÓRIA DISPONIVEL PARA O COMPILADOR. OPTOU-SE POR DEIXAR, COMO COMENTARIO, FUNÇÕES AUXILIARES UTILIZADAS PARA MONITORAR O PROGRAMA.
**********************************************************************************************************
**********************************************RECON.C*****************************************************

#define NRANSI
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<alloc.h>
#include<ctype.h>
#include<string.h>
#include<conio.h>
#include ""... "\prog\o\nr.h"
#include ""..."\prog\o\nrutil.h"
#include ""..."\prog\o\nrutil.c"
#include ""..."\prog\r\svdcmp.c"
#include ""..."\prog\r\pythag.c"
#include ""..."\gaussy.c"

#define MFM(i,j) (*(mfm+c1*(i)+j))
#define MB(i,j) (*(mb+nctm *(i)+j))
#define ST(i,j) (*(st+nctm *(i)+j))
#define MS(i,j) (*(ms+nctm *(i)+j))
#define M(i,k) (*(fd+2 *(i)+ k ))
#define A(i,j) (*(a+mi *(i)+ j ))
#define AC(i,j) (*(ac+nic1 *(i)+j))
#define TE(i,j) (*(te+c2 *(i)+j))
#define ME(i,j) (*(me+mic1 *(i)+j))
#define Bi(i,j) (*(bi+n1 *(i)+j))
#define Pi(i,j) (*(ppi+n3 *(i)+j))
#define ACC(i,j) (*(acc+(nic1+rq) *(i)+j))
#define AR(i,j) (*(ar+(nic1+rq) *(i)+j))
#define BCI(i,j) (*(bci+(nic1+rq) *(i)+j))
#define ACB(i,j) (*(acb+(nic1+rq) *(i)+j))
#define TR(i,j) (*(tr+ni *(i)+j))
#define MR(i,j) (*(mr+nic1 *(i)+j))
#define B1(i,j) (*(b1+n1 *(i)+j))
#define P(i,j) (*(pp+n3*(i)+j))
#define P1(i,j) (*(p1+n3*(i)+j))
#define P3(i,j) (*(p3+n3*(i)+j))
#define UP3(i,j) (*(up3+(n3-rp)*(i)+j))
#define VNO(i,j) (*(vno+(n3-rp)*(i)+j))
#define VI(i,j) (*(vi+(n3-rp)*(i)+j))
#define U(j,z) (*(u+n3*(j)+z))
#define YHT(i,z) (*(yh+m*(i)+z))
#define YTB1(i,j) (*(yb+n1 *(i)+j))
#define S(j,z) (*(pps+n1 *(j)+z))
#define Sf(j,z) (*(ppsf+n1*(j)+z))
#define SB1T(i,z) (*(sb+m*(i)+z))
#define SB1TY(i,j) (*(sby+m*(i)+j))
#define G(i,j) (*(g+m*(i)+j))
#define I(i,j) (*(ui+p*(i)+j))
#define X(i,j) (*(x+1*(i)+j))
#define YTB1X(i,j) (*(yx+1*(i)+j))
#define L(i,j) (*(l+1*(i)+j))
#define AJ(i,j) (*(aj+1*(i)+j))
#define B1m(z,j) (*(b1m+q *(z)+j))
#define BX(i,j) (*(bx+1*(i)+j))
```

```
#define VO(i,j) (*(vo+1*(i)+j))
#define ALC(i,j) (*(alc+3*(i)+j))
#define CM(i,j) (*(cm+3*(i)+j))
#define CNM(i,j) (*(cnm+3*(i)+j))
#define DD(i,j) (*(dd+1*(i)+j))
#define DL(i,j) (*(dl+blo*(i)+j))
#define DE(i,j) (*(de+c1*(i)+j))
#define D(i,j) (*(d+blo*(i)+j))
#define B2DE(i,j) (*(bde+blo*(i)+j))
#define DT(i,j) (*(dt+p*(i)+j))
#define D1(i,j) (*(d1+blo*(i)+j))
#define D2(i,j) (*(d2+rd*(i)+j))
#define D3(i,j) (*(d3+blo*(i)+j))
#define U2(j,z) (*(u2+blo*(j)+z))
#define HN(i,j) (*(hnd+p*(i)+j))
#define UD(i,j) (*(ud+pd*(i)+j))
#define ZT(i,j) (*(zt+p*(i)+j))
#define ZHT(i,j) (*(zh+p*(i)+j))
#define B1X(i,j) (*(b1x+1*(i)+j))
#define S1B1TY(i,j) (*(s1b+p*(i)+j))
#define Aa(i,j) (*(aa+pd*(i)+j))
#define Ndel(i,j) (*(nl+pd*(i)+j))
#define U2I(i,j) (*(u2i+p*(i)+j))
#define YHTB1X(i,j) (*(ybx+1*(i)+j))
#define Nd(i,j) (*(nd+1*(i)+j))
#define Ndi(i,j) (*(ndi+1*(i)+j))
#define D1D3(i,j) (*(ddd+td*(i)+j))
#define ZE(i,j) (*(ze+1*(i)+j))
#define HH(i,j) (*(hh+p*(i)+j))
#define YHTB1(i,j) (*(ytb+n1*(i)+j))
#define S1B1T(i,j) (*(sbt+m*(i)+j))
#define B2TY(i,j) (*(bty+p*(i)+j))
#define ZTYT(i,j) (*(zy+m*(i)+j))
#define Zb(i,j) (*(zb+1*(i)+j))
#define Za(i,j) (*(za+pd*(i)+j))
#define B2S2(i,j) (*(bs+n2*(i)+j))
#define BBTY(i,j) (*(bb+p*(i)+j))
#define ZTH(i,j) (*(zht+p*(i)+j))
#define LB(i,j) (*(lb+1*(i)+j))
#define AJ2(i,j) (*(aj2+1*(i)+j))
#define S2Ndel(i,j) (*(sdl+pd*(i)+j))
#define Ndelta(i,j) (*(ndt+1*(i)+j))
#define Noo(i,j) (*(noo+p*(i)+j))
#define No(i,j) (*(no+n2*(i)+j))
#define Nov(i,j) (*(nov+1*(i)+j))
#define S2(i,j) (*(ss+n2*(i)+j))
#define Sd(i,j) (*(ppd+vd*(i)+j))
#define NoSd(i,j) (*(ns+n2*(i)+j))
#define H1Y(i,j) (*(hy+p*(i)+j))
#define H2Y(i,j) (*(hyy+p*(i)+j))
#define Bii(i,j) (*(bii+n2*(i)+j))
#define DM(i,j) (*(dm+3*(i)+j))
#define Ni(i,j) (*(nii+1*(i)+j))
#define N(i,j) (*(n+1*(i)+j))
#define H1(i,j) (*(h1+m*(i)+j))
#define D1D3ZE(i,j) (*(dze+1*(i)+j))
#define D1D3ZEi(i,j) (*(dzei+blo*(i)+j))
#define B2(i,j) (*(b2+n2*(i)+j))
#define XC(i,j) (*(xc+1*(i)+j))
#define Nf(i,j) (*(nf+n2*(i)+j))
#define COL(i,j) (*(co+1*(i)+j))
#define YHTCOL(i,j) (*(yco+1*(i)+j))
#define B1AJ2(i,j) (*(ba+1*(i)+j))
#define B2Ndelta(i,j) (*(bdlt+1*(i)+j))
#define Np(i,j) (*(np+n2*(i)+j))
#define XD(i,j) (*(xd+1*(i)+j))
#define Ninv(i,j) (*(ninv+n2*(i)+j))
#define Delta(i,j) (*(dlt+1*(i)+j))
#define ADD(i,j) (*(add+po*(i)+j))
#define DF(i,j) (*(df+blo*(i)+j))
#define Zainv(i,j) (*(zai+pd*(i)+j))
#define BiiF(i,j) (*(bif+nulo*(i)+j))
#define B2F(i,j) (*(b2f+nulo*(i)+j))
#define Nol(i,j) (*(nol+1*(i)+j))
#define DAF(i,j) (*(daf+blo*(i)+j))
#define DBF(i,j) (*(dbf+p*(i)+j))
```

```
#define DDE(i,j) (*(dde+1*(i)+j))
#define COMP(i,j) (*(comp+blo*(i)+j))
#define D_(i,j) (*(d_+p*(i)+j))
#define Dd(i,j) (*(dc+1*(i)+j))
#define NDd(i,j) (*(ndd+1*(i)+j))
#define B2Nd(i,j) (*(bnd+1*(i)+j))
#define US(i,j) (*(us+blo*(i)+j))
#define WS(i,j) (*(ws+1*(i)+j))
#define WSM(i,j) (*(wsm+blo*(i)+j))
#define VTS(i,j) (*(vts+blo*(i)+j))
#define DAUX_(i,j) (*(daux+blo*(i)+j))
#define N_saj(i,j)  (*(nsaj+1*(i)+j))
#define DEF(i,j) (*(def+blo*(i)+j))

#define TM(size)  (float *)malloc(size*sizeof(float))
#define TMI(size) (int *)malloc(size*sizeof(int))
#define TMC(size) (char *)malloc(size*sizeof(char))

#define MNO  10
#define MCOR 20
#define MCOM 10
#define MREA  5
#define MRES  3
#define limite 0.01
#define NP 20
#define MP 30

 char tt;  int nm;  int mm;  int ni;  int nulo;
 int td;  int mi;  int c;  int rq;  int nr;  int q;  int porc;  int qq;  int ij;  int n1;  int n2;  int n3;
 int n100;  int vdt;  int vd;  int zero;  int po;  int blo;  int cont;  int iter;  int niter;  int tst;
 int M[MCOR][2];
 float snf;
 float varct[MCOR];
 float varct2[MCOR];
 float varcts[MCOR];
 float MFM[MCOR][MCOM+1];
 float X[MCOR*(MCOM+1)][1];
 float DD[MCOR*(MCOM+1)][1];
 float TE[MREA][MCOM+2];
 float TR[MRES][MCOR];
 int viv[MCOR*(MCOM+1)+MREA];
 int vivm[MCOR*(MCOM+1)+MREA];
 int vie[MNO*(MCOM+1)+MRES*(MCOM+1)+MCOR];

void entrar(void), mostrar(void), gravar(void),ler(void);
char menu(void);
void reconciliar(void), instrucao(void);
void backs(int rf,int rs,int *irf,float *p1f,float *uf);
void prod_m(int s,int rp,int w,float *y,float *h,float *yh);
int  lu(int rf, int *irf, float *p1);
void vnom(int c,int rq,int ni,int nic1,int *viv,char *cm,char *cnm,char *dm );
void variancia( int ni,int c,int n1,int tst,float *varct,float *mfm,float *pps);

main()
{
 char ch;
 for(;;){
              ch=menu();
              switch(ch) {
              case 'I' : instrucao();
                         break;
              case 'E' : entrar();
                         break;
              case 'M' : mostrar();
                         break;
              case 'G' : gravar();
                         break;
              case 'L' : ler();
                         break;
              case 'R' : reconciliar();
                         break;
              case 'S' : exit(0);

              }
              }
 }
```

```
void reconciliar(void)
{

 float *vnob,*daf,*dbf,*d_,*nsaj;
 int *fd;
 float  *a, *ac;
 int c1, c2;
 int  nic1,mic1,nrc1;
 int m,m1,n1,n2,n3,p11;
 int rp,*rpf,*rdf,rd, pd,*pdf,p,*pf,matI,rf,rs;
 int *ig;
 float *dc,*ndd,*bnd;
 float *te, *me, *acc, *mfm;
 float *bci,*acb;
 float *bi, *ppi; float *b1; float *b2; float *pp; float *bii;
 float *tr, *mr,*ar,*hnd;
 int  *ip, *ic, *irp, *ic2, *ip2, *ird,*ird1,*igd,*ind,*izd,*irf;
 float  *pl,  *p1,*p1_,*p1f, *p2,*p2f, *u,*uf, *pu,*puf, *up,*d2, *d3,*d1;
 float *zh, *dl, *de, *bde, *dt,*dlt;
 float *ud, *du,*u2;
 float *p3, *up3, *vno, *vi,*bif,*b2f,*zti,*dzei,*add,*df,*zai;
 float *h, *y, *yh, *zt, *n,*noo,*nov,*nol;
 float *b1x,*ybx,*nd,*ndi,*ddd,*ze,*dze,*no;
 float *hh,*ytb,*sbt,*h1,*hy,*bty;
 float *zht,*za,*hyy,*lb,*sdl;
 float *zy,*zb,*s1b,*aa,*nl;
 float *ss,*ns,*ndt,*bs,*bb,*ba,*co,*bdlt,*yco,*nf,*nii,*np;
 float   *yb, *pps,*ppd,*ppsf, *sb, *sby, *g, *ui,*u2i, *x, *dd, *dde, *d,*def;
 float *yx, *l, *aj,*aj2;
 float *b1m, *bx, *vo,*xd,*ninv,*comp;
 char *cm , *cnm, *dm, *comptroc;
 float a1, a2, aux, aux1,aux7,mat,ant;
 float *wm,**ai,**um,**v,*us,*ws,*wsm,*daux,*vts;

 double fabs();

 register int i,j,k,w,ki,kj,kz,kt,s,t,z;
 register int k2,t2;
            FILE *fp;
 char pa[20];

 {
 printf("\nNome do Arquivo dos Resultados:");
 gets(pa);
 if((fp=fopen(pa,"w"))==NULL){
 printf("Nao posso abrir arquivo\n");
 return;
 }
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Este simbolo identificará os dados que estão sendo mandados para o arquivo que sera lido no programa H_RETIFIC.C
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


 vnob=NULL; *vnob=0; daf=NULL; *daf=0; dbf=NULL; *dbf=0; d_=NULL;  *d_=0;
fd=NULL;  *fd=0; a=NULL;  *a=0; ac=NULL;  *ac=0; ig=NULL;  *ig=0; dc=NULL;  *dc=0; ndd=NULL; *ndd=0;
 bnd=NULL; *bnd=0; te=NULL;  *te=0; me=NULL;  *me=0; acc=NULL; *acc=0; mfm=NULL; *mfm=0;
 bci=NULL; *bci=0; acb=NULL; *acb=0; bi=NULL;  *bi=0; ppi=NULL; *ppi=0; b1=NULL;  *b1=0; b2=NULL;  *b2=0;
 pp=NULL;  *pp=0; bii=NULL; *bii=0; tr=NULL;  *tr=0; mr=NULL;  *mr=0; ar=NULL;  *ar=0; hnd=NULL; *hnd=0;
 ip=NULL;  *ip=0; ic=NULL;  *ic=0; irp=NULL; *irp=0; ic2=NULL; *ic2=0; ip2=NULL; *ip2=0; ird=NULL; *ird=0;
 ird1=NULL; *ird1=0; igd=NULL; *igd=0; ind=NULL; *ind=0; izd=NULL; *izd=0; pl=NULL;  *pl=0; p1=NULL;  *p1=0;
 p2=NULL;  *p2=0; u=NULL;   *u=0; pu=NULL;  *pu=0; up=NULL;  *up=0; d1=NULL;  *d1=0; d2=NULL;  *d2=0;
 d3=NULL;  *d3=0; zh=NULL;  *zh=0; dl=NULL;  *dl=0; de=NULL;  *de=0; bde=NULL; *bde=0; dt=NULL;  *dt=0;
dlt=NULL; *dlt=0; ud=NULL;  *ud=0; du=NULL;  *du=0; u2=NULL;  *u2=0; p3=NULL;  *p3=0; up3=NULL; *up3=0;
 vno=NULL; *vno=0; vi=NULL;  *vi=0; bif=NULL; *bif=0; b2f=NULL; *b2f=0; zti=NULL; *zti=0; dzei=NULL; *dzei=0;
 add=NULL; *add=0; df=NULL;  *df=0; zai=NULL; *zai=0; h=NULL;   *h=0; y=NULL;   *y=0; yh=NULL;  *yh=0;
 zt=NULL;  *zt=0; n=NULL;   *n=0; noo=NULL; *noo=0; nov=NULL; *nov=0; nol=NULL; *nol=0; b1x=NULL; *b1x=0;
 ybx=NULL; *ybx=0; nd=NULL;  *nd=0; ndi=NULL; *ndi=0; ddd=NULL; *ddd=0; ze=NULL;  *ze=0; dze=NULL; *dze=0;
 no=NULL;  *no=0; hh=NULL;  *hh=0; ytb=NULL; *ytb=0; sbt=NULL; *sbt=0; h1=NULL;  *h1=0; hy=NULL;  *hy=0;
 bty=NULL; *bty=0; zht=NULL; *zht=0; za=NULL;  *za=0; hyy=NULL; *hyy=0; lb=NULL;  *lb=0; sdl=NULL; *sdl=0;
zy=NULL;  *zy=0; zb=NULL;  *zb=0; s1b=NULL; *s1b=0; aa=NULL;  *aa=0;nl=NULL;  *nl=0; ss=NULL;  *ss=0;
 ns=NULL;  *ns=0; ndt=NULL; *ndt=0; bs=NULL;  *bs=0; bb=NULL;  *bb=0; ba=NULL;  *ba=0; co=NULL;  *co=0;
 bdlt=NULL; *bdlt=0; yco=NULL; *yco=0; nf=NULL;  *nf=0; n=NULL;   *n=0; nii=NULL; *nii=0; np=NULL;  *np=0;
 yb=NULL;  *yb=0; pps=NULL; *pps=0; ppd=NULL; *ppd=0; sb=NULL;  *sb=0; sby=NULL; *sby=0; g=NULL;   *g=0;
 ui=NULL;  *ui=0; u2i=NULL; *u2i=0; x=NULL;   *x=0; dd=NULL;  *dd=0; dde=NULL; *dde=0; d=NULL;   *d=0;
```

```
yx=NULL;  *yx=0; l=NULL;  *l=0; aj=NULL;  *aj=0; aj2=NULL; *aj2=0; b1m=NULL; *b1m=0; bx=NULL;  *bx=0;
vo=NULL;  *vo=0; xd=NULL;  *xd=0; ninv=NULL; *ninv=0; comp=NULL; *comp=0; cm=NULL; cnm=NULL; dm=NULL;
daux=NULL; *daux=0; vts=NULL; *vts=0; wsm=NULL; *wsm=0; ws=NULL;  *ws=0; us=NULL;  *us=0; comptroc=NULL;

c1=c+1;
c2=c+2;
mic1=mi*c1;
nic1=ni*c1;
nrc1=nr*c1;
clrscr();
tst=0;
comptroc=TMC(3);
fd=TMI(ni*2);
mfm=TM(ni*c1);
x=TM(n1*1); *x=0;
dd=TM((vd+zero)*1); *dd=0;

for(i=0;i<ni;i++)
        for(k=0;k<2;k++) M(i,k)=M[i][k];

for(i=0;i<ni;i++)
        for(j=0;j<c1;j++) MFM(i,j)=MFM[i][j];

for(j=0,i=0;i<q;i++) X(i,j)=X[i][j];

if (vd)
{
 for(j=0,i=0;i<vd+zero;i++)
    {
      DD(i,j)=DD[i][j];
      mat=DD(i,j);
     }
 }

a=TM(ni*mi);

for(i=0;i<ni;i++)
        for(j=0;j<mi;j++)
                {
                A(i,j)=(M(i,0)==j+1 ? 1 : 0);
                        if(A(i,j)==0)
                                A(i,j)=(M(i,1)==j+1 ? -1 : 0);
                }

free(fd);

/* OBTENÇÃO DA MATRIZ DOS COMPONENTES AC */

ac=TM(mic1*nic1);

for(i=0; i<mi; i++)
        for(ki=0;ki<c1;ki++)
                for(j=0;j<ni;j++)
                                for(kj=0;kj<c1;kj++)
        AC( (c1*i+ki),(c1*j+kj) ) = ( ki==kj ? A(j,i) : 0 );  /* AT */

/* OBTENÇÃO DA MATRIZ ESTEQUIMETRIA ME */

te=TM(rq*c2);
me=TM(rq*mic1);

for(i=0; i<rq; i++)
        for(j=0; j<c2; j++) TE(i,j)= TE[i][j];

for(i=0; i<rq; i++)
        for(j=0; j<mi; j++)
                for(kj=0; kj<c1; kj++)
ME(i,c1*j+kj)=( c1*j+kj==(TE(i,0)-1)* c1+kj ? TE(i,kj+1) : 0 );

free(te);
free(a);

/* COLOCAR MET AO LADO DE AC --> ACC */

acc=TM(mic1*(nic1+rq));
```

```
for(i=0; i<mic1; i++)
        for(j=0; j<nic1; j++)
                {
                        ACC(i,j)=AC(i,j);
                }

for(i=0;i<mic1;i++)
        for(j=nic1;j<nic1+rq;j++)
                ACC(i,j)=ME(j-nic1,i); //MET

free(ac);

//Expandir TR--> MR

tr=TM(nr*ni);
mr=TM(nrc1*nic1);

for(i=0; i<nr; i++)
        for(j=0; j<ni; j++)
                TR(i,j)=TR[i][j];

for(i=0;i<nr;i++)
        for(ki=0;ki<c1;ki++)
                for(j=0;j<ni;j++)
                                for(kj=0;kj<c1;kj++)
                                        MR((c1*i+ki),(c1*j+kj))=(ki==kj ? TR(i,j) : 0);

free(tr);
free(me);

/* OBTER AR A PARTIR DE ACC E MR E ZEROS */

ar=TM((mic1+nrc1)*(nic1+rq));

for(i=0;i<mic1;i++)
        for(j=0;j<nic1+rq;j++)
                AR(i,j)=ACC(i,j);

for(i=mic1;i<mic1+nrc1;i++)
        for(j=0;j<nic1;j++)
                AR(i,j)=MR(i-mic1,j);

for(i=mic1;i<mic1+nrc1;i++)
        for(j=nic1;j<nic1+rq;j++)
                AR(i,j)=0;

free(mr);
free(acc);

/* MATRIZ DOS BALANÇOS INDIVIDUAIS */

bci=TM(ni*(nic1+rq));

for(i=0;i<ni;i++)
        {
         for(j=0;j<nic1+rq;j++)
                        BCI(i,j)=(j>=(c1*i) && j<(c1*i)+c1 ? 1 : 0);
        }

for(i=0;i<ni;i++)
        {
         for(j=0;j<nic1+rq;j++)
                        if( BCI(i,j) && j==i*c1+(c1-1))
                                BCI(i,j)=-1;
        }

/* INCLUIR BCI wm R --> ACB */

acb=TM((mic1+nrc1+ni)*(nic1+rq));

for(i=0;i<mic1+nrc1;i++)
        for(j=0;j<nic1+rq;j++)
                ACB(i,j)=AR(i,j);

for(i=mic1+nrc1;i<mic1+nrc1+ni;i++)
        for(j=0;j<nic1+rq;j++)
```

```
                {
                        ACB(i,j)=BCI(i-(mic1+nrc1),j);
                        mat=ACB(i,j);
                }

free(ar);

/* SEPARAR AS COLUNAS DA ACB DE ACORDO COM O PONTEIRO */

for(n1=0,n2=0,n3=0,n100=0,j=0;j<(nic1+rq);j++)
        {
        if(viv[j]==1) n1++;
                else if(viv[j]==2) n2++;
                                        else if(viv[j]==3) n3++;
                                                                else if(viv[j]==100) n100++;
        }


nulo=(n2+n100);
bi=TM((mic1+nrc1+ni)* n1);
bii=TM((mic1+nrc1+ni)*n2);
bif=TM((mic1+nrc1+ni)*nulo);
ppi=TM((mic1+nrc1+ni)* n3);

for(kj=0,ki=0,kz=0,kt=0,j=0;j<nic1+rq;j++)
        {
        ant=viv[j];
        if(viv[j]==1)
                {
                for(i=0;i<mic1+nrc1+ni;i++)
                                {
                                ant=ACB(i,j);
                                Bi(i,kj)=ACB(i,j);
                                mat=Bi(i,kj);
                                }
                                kj++;
                        }
                else if(viv[j]==2)
                                        {
                                        for(i=0;i<mic1+nrc1+ni;i++)
                                                {
                                                        ant=ACB(i,j);
                                                        Bii(i,kz)=ACB(i,j);
                                                        mat=Bii(i,kz);
                                                        BiiF(i,kt)=ACB(i,j);
                                                        mat=BiiF(i,kt);
                                                        }
                                                        kz++;
                                                        kt++;
                                        }
                else if(viv[j]==100)
                                        {
                                        for(i=0;i<(mic1+nrc1+ni);i++)
                                                {
                                                        ant=ACB(i,j);
                                                        BiiF(i,kt)=ACB(i,j);
                                                        mat=BiiF(i,kt);
                                                        }
                                                        kt++;
                                        }
                else if(viv[j]==3)
                                        {
                                                for(i=0;i<nic1+nrc1+ni;i++)
                                                        {
                                                        ant=ACB(i,j);
                                                        Pi(i,ki)=ACB(i,j);
                                                        mat=Pi(i,ki);
                                                        }
                                                        ki++;
                                                }
        }

mat=BiiF(0,0);
```

```
/* RETIRA LINHAS ZERO DE Bi */

for(m=0,i=0;i<mic1+nrc1+ni;i++)
        if( vie[i]==1) m++;

b1=TM(m*n1);

for(ki=0,i=0;i<mic1+nrc1+ni;i++)
        if(vie[i]==1)
                {
                for(j=0;j<n1;j++)
                                        {
                                        B1(ki,j)=Bi(i,j);
                                        mat=B1(ki,j);
                                        }
                                        ki++;
                }

free(bi);
free(acb);

b2f=TM(m*nulo); *b2f=0;
b2=TM(m*n2);  *b2=0;

/* RETIRADA DAS LINHAS NULAS DE B2 E B2F */

for(ki=0,i=0;i<mic1+nrc1+ni;i++)
        if(vie[i]==1)
                {
                for(j=0;j<n2;j++)
                                        {
                                        B2(ki,j)=Bii(i,j);
                                        mat=B2(ki,j);
                                        }
                                        ki++;
                }

for(kt=0,i=0;i<mic1+nrc1+ni;i++)
        if(vie[i]==1)
                {
                for(j=0;j<nulo;j++)
                                        {
                                        ant=BiiF(0,0);
                                        B2F(kt,j)=BiiF(i,j);
                                        mat=B2F(kt,j);
                                        }
                                        kt++;
                }

free(bii);

/*
clrscr();
printf("\nVALORES DE B1\n");
for(i=0;i<m;i++)
{
for(j=0;j<n1;j++)
printf("%f",B1(i,j));
printf("\n");
}
tt=getche();

clrscr();
printf("\nVALORES DE B2F\n");
for(i=0;i<m;i++)
{
  for(j=0;j<nulo;j++)
printf("%2.0f",B2F(i,j));
printf("\n");
}
tt=getche();
*/

cm= TMC(n1*3);
dm=TMC(n2*3);
cnm=TMC(n3*3);
```

```
*****************************************************************************************************************************
vnom(c,rq,ni,c1,viv,cm,cnm,dm)
*****************************************************************************************************************************

*****************************************************************************************************************************
COMECO DO PROCEDIMENTO DE CALCULO DO YHT
*****************************************************************************************************************************

yh=TM(m*m); *yh=0;

if (n3)
{ //abre o if(n3)

/* RETIRA AS LINHAS ZERO DE Pi --> P */

pp=TM(m*n3); *pp=0;

for(ki=0,i=0; i<mic1+nrc1+ni ;i++)
            if(vie[i]==1)
              {
                        for(j=0;j<n3;j++)
                                    P(ki,j)=Pi(i,j);
                                    ki++;
                        }
/*
clrscr();
printf("VALORES DE B3\n");
for(i=0;i<m;i++)
 {
 for(j=0;j<n3;j++)
             printf("  %f",P(i,j));
             printf("\n");
             }
tt=getche();
*/

free(ppi);

p1=TM(n3*n3); //*p1=0;
p1_=TM(n3*n3);  *p1_=0; //não sabemos o tamanho exato ainda
p3=TM((n3)*(n3)); //*p3=0; //porque o tamanho para o D pode ser maior
u=TM(n3*n3);  *u=0;

for(i=0;i<n3;i++)
 for(j=0;j<n3;j++)
             {
                        U(i,j)=0;
             }

*****************************************************************************************************************************
gaussy(m,n3,&pf,&rpf,pp,p1_,p3,u,yh);
*****************************************************************************************************************************

rp=*rpf;
p=*pf;

clrscr();

printf("TESTE COM OS VALORES DA MATRIZ SEPARADA\n\n");

aux=0;
for(i=0;i<n3;i++)
  for(j=0;j<p;j++)
             for(z=0;z<m;z++)
                        {
                                    aux+=P(z,i)*YHT(j,z);
                        }

printf("PRODUTO MAT(t)*PROJ=%10.4f",aux);
if(fabs(aux)>0.001)
printf("\n\nA PROJECAO DA MATRIZ P ESTA' ERRADA\n");
else
printf("\n\nA PROJECAO DA MATRIZ P ESTA' CORRETA\n");
tt=getche();
```

```
pps=TM(n1*n1); //*pps=0;
tst=0;

*****************************************************************************************************************************************
variancia(n1,c,n1,tst,varct,mfm,pps);
*****************************************************************************************************************************************
free(h);
free(y);

}//fecha o if(n3)

*****************************************************************************************************************************************
FIM DO PROCEDIMENTO DE CÁLCULO DO YHT
*****************************************************************************************************************************************

if( !n3)
        {

p=m;
pps=TM(n1*n1); *pps=0;
tst=0;

for(i=0;i<m;i++)
        for(j=0;j<m;j++) YHT(i,j)=(i==j ? 1 : 0 );

*****************************************************************************************************************************************
variancia(ni,c,n1,tst,varct,mfm,pps);
*****************************************************************************************************************************************

//printf("\nVALORES DE S-para n3=0\n");
for(i=0;i<n1;i++)
        for(j=0;j<n1;j++)
                if(i==j)
                        {
// printf("%10.4f",S(i,j));
// printf("\n");
                        }
//tt=getche();

        }

if(!n2)
        {

yb=TM(m*n1); *yb=0;
sb=TM(n1*m); *sb=0;
sby=TM(n1*m); *sby=0;
g =TM(m*m);  *g=0;
yx=TM(m*1); *yx=0;
l=TM(m*1); *l=0;
aj=TM(n1*1); *aj=0;

/* OBTER A MATRIZ G : YTB1=YHT*B1 */

for(a1=0,i=0; i<p; i++)
        for(j=0;j<n1;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                a1+= YHT(i,z)*B1(z,j) ;
                                }
                                YTB1(i,j)=a1;
                                mat=YTB1(i,j);
                                a1=0;
                }

/*
clrscr();
printf("\nVALORES DE S\n");
 for(i=0;i<n1;i++)
        for(j=0;j<n1;j++)
        if(i==j)
         {
         printf("%10.4f",S(i,j));
         printf("\n");
         }
```

```
tt=getche();
*/

/* SBT = S * BT */

for(a1=0,i=0; i<n1; i++)
        for(z=0;z<m;z++)
                {
                        for(j=0;j<n1;j++)
                                {
                                a1+= S(i,j)*B1(z,j) ;     /* BT */
                                }
                                SB1T(i,z)=a1;
                                mat=SB1T(i,z);
                                a1=0;
                }

/* SBY = SBT * YTT */

for(a1=0,i=0;i<n1;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                a1+=SB1T(i,z)*YHT(j,z);    /* YTT= (YHT)T */
                                }
                                SB1TY(i,j)=a1;
                                mat=SB1TY(i,j);
                                a1=0;
                        }

/* G = YTB1 * SB1TY */

for(a1=0,i=0;i<p;i++)
        for(z=0;z<p;z++)
                {
                        for(j=0;j<n1;j++)
                                {
                                a1+=YTB1(i,j)*SB1TY(j,z);
                                }
                                G(i,z)=a1;
                                mat=G(i,z);
                                a1=0;
                }

/* INVERSA DE G */
ui=TM(p*p); *ui=0;

for(i=0;i<p;i++)
        for (j=0;j<p;j++) I(i,j)=(i==j ? 1 : 0 );

ig=TMI(p); *ig=0;

for(i=0;i<p;i++) ig[i]=i;

*************************************************************************************************************************
lu(p,ig,g);
backs(p,p,ig,g,ui);
*************************************************************************************************************************

for(i=0;i<p;i++)
        for(j=0;j<p;j++)

/* OBTENÇÃO DE L = -inv G(=I)*[YTB1 * X ] */
/* YTX = [ YTB1 * X ] */

for(a1=0,i=0;i<p;i++)
        {
         for(j=0,z=0; z<n1; z++)
                        {
                         a1+=YTB1(i,z)* X(z,j);
                         }
                         YTB1X(i,j)=a1;
                         mat=YTB1X(i,j);
                         a1=0;
        }
```

```
/* L = - I * YTB1X */

for(a1=0,i=0;i<p;i++)
        {
        for(j=0.z=0; z<p; z++)
                {
                a1+=I(i,z)*YTB1X(z,j);
                }
                L(i,j)=-a1;
                mat=L(i,j);
                a1=0;
        }

/*OBTENÇÃO DOS AJUSTES DO CASO LINEAR: AJ = SB1TY * L */

for(a1=0,i=0;i<n1;i++)
        {
        for(j=0,z=0;z<p;z++)
                {
                        a1+=SB1TY(i.z)*L(z,j);
                }
                        AJ(i,j)=a1;
                        mat=AJ(i,j);
                        a1=0;
        }

/* OBTENÇÃO DOS VALORES CORRIGIDOS */

for(j=0,i=0;i<n1;i++)
        X(i,j)+=AJ(i,j);
/*
printf("%d", fprintf(fp,"\nVazoes  Medidas:\n"));
for(i=0;i<n1;i++)
        {
        for(j=0;j<3;j++)
printf("%d", fprintf(fp,"%c",CM(i,j)));
printf("%d", fprintf(fp," "));
        }

printf("%d", fprintf(fp,"\n"));
printf("%d", fprintf(fp,"\nValores Corrigidos\n"));

for(j=0,i=0;i<n1 ;i++)
printf("%d", fprintf(fp,"%10.4f", X(i,j)));
printf("\n");
*/

free(yx);
free(l);
free(yb);
free(sb);
free(sby);
free(g);

if (n3)
{ //abre o if(n3)

vnob=TM(n3); *vnob=0;

for(i=0;i<n3;i++) vnob[i]=0;

/* MUDA vnm DE ACORDO COM ic[] */

for(j=0;j<n3-1;j++)
        if(j<ic[j])
                for(i=0;i<3;i++)
                        {
                                comptroc[i]=CNM(j,i);
                                CNM(j,i)=CNM(ic[j],i);
                                CNM(ic[j],i)=comptroc[i];
                        }

printf("%d", fprintf(fp,"\n\nVazoes nao-medidas apos troca\n"));
```

```
for(i=0;i<n3;i++)
            {
            for(j=0;j<3;j++)
printf("%d", fprintf(fp,"%c",CNM(i,j)));
printf("%d", fprintf(fp," "));
            }
printf("%d", fprintf(fp,"\n"));

if(rp==n3)

printf("%d",fprintf(fp,"\nTodos valores nao medidos sao observaveis\n"));
            else
                        {
                                    p3=TM(rp*(n3-rp));
                        }

for(i=0;i<rp;i++)
            for(j=rp;j<n3;j++)
                        P3(i,j-rp)=P(i,j);

/* OBTENÇÃO DE VNO=[P1-1*P3 :I] */

up3=TM(rp*(n3-rp));

for(a1=0,i=0;i<rp;i++)
            for(j=0;j<n3-rp;j++)
                        {
                                    for(z=0;z<rp;z++)
                                                {
                                                a1+=U(i,z)*P3(z,j);
                                                }
                                                UP3(i,j)=a1;
                                                a1=0;
                        }

/* VNO=[UP3|I] */
vi=TM((n3-rp)*(n3-rp));

for(i=0;i<n3-rp;i++)
            for(j=0;j<n3-rp;j++)
                        VI(i,j)=(i==j ? 1:0);

vno=TM(n3*(n3-rp));

for(i=0;i<rp;i++)
            for(j=0;j<n3-rp;j++)
                        VNO(i,j)=UP3(i,j);

for(i=rp;i<n3;i++)
            for(j=0;j<n3-rp;j++)
                        VNO(i,j)=VI(i-rp,j);

/* VERIFICA QUAIS LINHAS DE vno TEM ELEMENTO NAO ZERO */

for(a1=0,i=0;i<n3;i++)
            {
            for(j=0;j<n3-rp;j++)
                        {
                                    a1+=fabs(VNO(i,j));
                        }
                                    vnob[i]=a1;
                                    a1=0;
            }


printf("%d", fprintf(fp,"\nValores nao-observaveis: "));
for(i=0;i<n3;i++)
            if(vnob[i] !=0 )
                        {
                        for(j=0;j<3;j++)
printf("%d", fprintf(fp,"%c",CNM(i,j)));
printf("%d", fprintf(fp, " "));
                        }
printf("%d", fprintf(fp,"\n"));

} //fecha if(n3)
```

```
/* CALCULA OS VALORES OBSERVÁVEIS */
/* MUDA AS LINHAS DE B1 DE ACORDO COM ip PARA RETIRAR AS LINHAS LD */

m1=m-1;

for(ip[m1]=m1,i=0;i<m1;i++)
        for(j=0; j<n1; j++)
                {
                        a1=B1(i,j);
                        B1(i,j)=B1(ip[i],j);
                        B1(ip[i],j)=a1;
                }

/* REDUZ AS LINHAS DE B1 */

b1m=TM(rp*q); *b1m=0;

for(i=0;i<rp;i++)
        for(j=0;j<n1;j++)
                B1m(i,j)=-B1(i,j);

/* FAZ O PRODUTO DE B1m * X */

bx=TM(rp*1); *bx=0;

for(a1=0,i=0;i<rp;i++)
        {
         for(z=0,j=0;j<n1;j++)
                {
                        a1+=B1m(i,j)*X(j,z);
                }
                        BX(i,z)=a1;
                        a1=0;
         }

vo=TM(n3*1); *vo=0;

/* FAZ O PRODUTO DA INVERSA DE P1 * BX */

for(a1=0,i=0;i<rp;i++)
        if(vnob[i]==0)
                {
                for(z=0,j=0;j<rp;j++)
                                        {
                                        a1+=U(i,j)*BX(j,z);
                                        }
                                        VO(i,z)=a1;
                                        a1=0;
                }
        else
                VO(i,z)=0;

for(z=0,i=rp;i<n3;i++)
        VO(i,z)=0;

/*
printf("%d", fprintf(fp,"\nValores Observaveis\n"));
for(j=0,i=0;i<n3 ;i++)
        {
printf("%d", fprintf(fp,"%10.4f", VO(i,j)));
        }
else
printf("%d", fprintf( fp,"\nTodos valores sao medidos\n"));
*/

fclose(fp);

free(vnob);
free(b1m);
free(bx);

} //fecha o if(!n2)


blo=0;
cont=0;
```

```
for(k=0;k<ni;k++)
            {
            {
                        for(j=0;j<c1;j++)
                                    {
                                     if(viv[k*c1+j]==2)
                                                cont++;
                                    }
            }
            if(cont)
                        {
                                    blo+=1;
                                    cont=0;
                        }
                       }

if(n2)   //COMECO A PARTE NAO LINEAR
 {

//TODOS OS LOCAIS ONDE SE ENCONTRAR mat PODEM SER SUPRIMIDOS POIS
//SERVEM APENAS PARA FINS DE ACOMPANHAMENTO.

vdt=vd+n100;
d=TM(p*blo); *d=0;
de=TM(blo*c1); *de=0;
bde=TM(m*blo); *bde=0;
dt=TM(blo*p); *dt=0;

*****************************************************************************************************************************
COMEÇO DO PROCEDIMENTO PARA O CALCULO DE Z.
*****************************************************************************************************************************

//printf("VALORES DE d\n");
for(ij=0,i=0;i<blo;i++)
            {
             for(j=0;j<c1;j++)
                                    {
                                     ant=DD(ij,0);
                                     DE(i,j)=DD(ij,0);
//printf("  DE(%d,%d)=%8.4f",i,j,DE(i,j));
                                     mat=DE(i,j);
                                     ij++;
                                    }
// printf("\n");
                        }
//tt=getche();

for(ki=0,i=0;i<ni;i++)
            for(j=0;j<c1;j++)
                        {
                                    if((viv[i*c1+j+rq]==2) || (viv[i*c1+j+rq]==100))
                                                {
                                                vivm[ki]=viv[i*c1+j+rq];
                                                mat=vivm[i*c1+j+rq];ki++;
                        }
             }

for(i=0;i<m;i++)
            for(j=0;j<nulo;j++)
                        mat=B2F(i,j);

for(i=0;i<blo;i++)
            for(j=0;j<c1;j++)
                        ant=DE(i,j);

 aux1=0; kt=0;

for(t=0,ki=0,i=0;i<blo;i++)
            {
             for(k=0;k<m;k++)
                        {
                                    for(j=0;j<c1;j++)
                                                {
                                                 if((vivm[i*c1+j+rq]==2 )||(vivm[i*c1+j+rq])==100)
                                                                        {
                                                                                    ki=j;
```

```
                                                ant=B2F(k,i*c1+ki);
                                                mat=DE(i,kt);  //DE(j,i)
                                                aux1+=B2F(k,i*c1+ki)*DE(i,kt);
                                                mat=aux1;
                                                kt++;
                                        }
                                }
                                kt=0;
                                        {
                                        //if (fabs(aux1)<0.0001) // PARA GARANTIR QUE ZERA
                                        //B2DE(k,t)=0;
                                        //else if(fabs(aux1>0.0001))
                                        B2DE(k,t)=aux1;
                                        mat=B2DE(k,t);
                                        aux1=0;
                                        ki=0;
                                        }
                        } t++;
                }

/*
clrscr();
printf("VALORES DE B2DE\n");
for(i=0;i<m;i++)
                {
                 for(j=0;j<blo;j++)
printf("%8.4f",B2DE(i,j));
printf("\n");
                }
tt=getche();
*/

aux1=0;

for(i=0;i<blo;i++)
                for(j=0;j<p;j++)
                        {
                                for(z=0;z<m;z++)
                                        {
                                        ant=B2DE(z,i);  mat=YHT(j,z);
                                        aux1+=B2DE(z,i)*YHT(j,z);
                                        }

                                        {
                                        DT(i,j)=aux1;
                                        mat=DT(i,j);
                                        aux1=0;
                        }
                        }

/*
clrscr();
printf("\nVALORES DE DT\n");
for(i=0;i<blo;i++)
                {
                 for(j=0;j<p;j++)
printf("%10.4f",DT(i,j));
printf("\n");
                }
tt=getche();
*/

/* PARTIÇÃO DE D PARA DETERMINAR Z */

free(bde);
free(dd);

//clrscr();
//printf("VALORES DE D(i,j)\n");
for(i=0;i<p;i++)
                {
                 for(j=0;j<blo;j++)
                                {
                                 ant=DT(j,i);
                                 D(i,j)=DT(j,i);
//printf("%10.4f",D(i,j));
```

```
                    }
//printf("\n");
            }
//tt=getchar();

daf=TM(p*blo); *daf=0;

//clrscr();
//printf("VALORES DE d(em DAF)\n");
for(i=0;i<p;i++)
            {
            for(j=0;j<blo;j++)
                                    {
                                    ant=D(i,j);
                                    DAF(i,j)=DT(j,i);
//printf("%8.4f",DAF(i,j));
                                    }
//                      printf("\n");
            }
// tt=getche();

free(dt);


*****************************************************************************************************************************
FIM DO PROCEDIMENTO PARA O CALCULO DE Z.
*****************************************************************************************************************************

zh=TM(p*p);
d1=TM(blo*blo); *d1=0;//porque nao tenho o valor para o tamanho certo
d3=TM(blo*blo);  *d3=0;//porque nao temos o tamanho ainda
u2=TM(blo*blo);  *u2=0;

/*
for(i=0;i<blo;i++)
 for(j=0;j<blo;j++)
            {
            D1(i,j)=0;
            U2(i,j)=0;
            }
*/

*****************************************************************************************************************************
gaussy(p,blo,&pdf,&rdf,d,d1,d3,u2,zh);
*****************************************************************************************************************************
pd=*pdf;
rd=*rdf;
td=blo-rd;

free(u2);

*u2=0; u2=NULL;

u2=TM(rd*rd); *u2=0;

clrscr();
printf("TESTE COM OS VALORES DA MATRIZ SEPARADA\n\n");
//printf("MATRIZ D RECUPERADA\n");
/*
for(i=0;i<p;i++)
 {
          for(j=0;j<blo;j++);
           printf("%8.4f",D(i,j));
           printf("\n");
          }
tt=getche();
*/

aux=0;
aux1=0;
for(i=0;i<blo;i++)
            {
            for(j=0;j<pd;j++)
                                    {
                                    for(z=0;z<p;z++)
                                                {
```

```
                                        aux+=DAF(z,i)*ZHT(j,z);
                                        }
                                        aux1+=aux;
//printf("DZHT=%10.6f",aux);
                                        aux=0;
                                }
//printf("\n");
        }

printf("PRODUTO MAT(t)*PROJ=%10.4f",aux1);
if(fabs(aux1)>0.001)
printf("\n\nA PROJECAO DA MATRIZ D ESTA' ERRADA\n");
else
printf("\n\nA PROJECAO DA MATRIZ D ESTA' CORRETA\n");
tt=getche();

//printf("VALORES DE D1(f)\n");
for(i=0;i<rd;i++)
        {
         for(j=0;j<rd;j++)
                                {
                                 if(fabs(D1(i,j))<0.0001)
                                        D1(i,j)=0;
//printf("%8.4f",D1(i,j));
                                }
//printf("\n");
        }
//tt=getche();

mat=0;//so para segurar o cursor

if(rd==blo)
{
for(i=0;i<rd;i++)
        for(j=0;j<p-blo;j++)
                D3(i,j)=0;
}

for(i=0;i<rd;i++)
        for(j=0;j<rd;j++)
                {
                        U2(i,j)=(i==j ? 1 : 0);
                        mat=U2(i,j);
                }

ird=TMI(rd);
ird1=TMI(rd);
*ird1=0;
for(i=0;i<rd;i++)
        {
         ird1[i]=i;
         mat=ird1[i];
//printf("ird1[%d]=%d",i,ird1[i]);
        }
//tt=getche();

free(pp);
free(p3);
free(p1);
free(d);

clrscr();

*****************************************************************************************************************************
lu(rd,ird1,d1);
backs(rd,rd,ird1,d1,u2);
*****************************************************************************************************************************
/*
clrscr();
printf("VALORES DE D1_\n");
for(i=0;i<rd;i++)
        {
         for(j=0;j<rd;j++)
printf("%12.4f",U2(i,j));
printf("\n");
        }
```

```
tt=getche();
mat=0; //segurar o cursor
*/

/*
clrscr();
printf("VALORES DE ZHT\n");
 for(i=0;i<pd;i++)
            {
            for(j=0;j<p;j++)
                        printf("%8.4f",ZHT(i,j));
                        printf("\n");
            }
tt=getche();
*/

//@//

printf("%d",fprintf(fp,"%d"" ""%d"" ""%d"" ""%d"" ""%d",n1,n2,m,vd,pd));
printf("%d",fprintf(fp,"\n"));
printf("%d",fprintf(fp,"%d"" ""%d"" ""%d"" ""%d"" ""%d"" ""%d",p,rq,c1,blo,nulo,ni));
printf("%d",fprintf(fp,"\n"));

for(j=0,i=0;i<n1;i++)
printf("%d",fprintf(fp,"%10.4f",X(i,j)));
printf("%d",fprintf(fp,"\n"));

for(i=0;i<m;i++)
            {
             for(j=0;j<n1;j++)
                                    {
printf("%d",fprintf(fp,"%10.4f",B1(i,j)));
                                    }
printf("%d",fprintf(fp,"\n"));
            }

for(i=0;i<m;i++)
            {
             for(j=0;j<n2;j++)
                                    {
printf("%d",fprintf(fp,"%8.2f",B2(i,j)));
                                    }
printf("%d",fprintf(fp,"\n"));
             }

for(i=0;i<m;i++)
            {
             for(j=0;j<nulo;j++)
                                    {
printf("%d",fprintf(fp,"%8.2f",B2F(i,j)));
                                    }
printf("%d",fprintf(fp,"\n"));
             }

for(i=0;i<p;i++)
            {
             for(j=0;j<m;j++)
                                    {
printf("%d",fprintf(fp,"%2.0f",YHT(i,j)));
                                    }
printf("%d",fprintf(fp,"\n"));
             }

for(i=0;i<n1;i++)
            {
             for(j=0;j<n1;j++)
                                    {
printf("%d",fprintf(fp,"%12.8f",S(i,j)));
                                    }
printf("%d",fprintf(fp,"\n"));
             }
//@//

free(b2f);

ppd=TM(vd*vd); *ppd=0;
```

```
tst=1;

clrscr();

*****************************************************************************************************************************
variancia(ni,c,vd,tst,varct2,de,ppd);
*****************************************************************************************************************************
/*
printf("\nVALORES DE Sd\n");
for(i=0;i<vd;i++)
        for(j=0;j<vd;j++)
                if(i==j)
                        {
printf("%10.4f",Sd(i,j));
printf("\n");
                        }
tt=getche();
*/

//@//

for(i=0;i<pd;i++)
        {
         for(j=0;j<p;j++)
                        {
printf("%d",fprintf(fp,"%12.8f",ZHT(i,j)));
                        }
printf("%d",fprintf(fp,"\n"));
        }

for(i=0;i<vd;i++)
        {
         for(j=0;j<vd;j++)
                        {
printf("%d",fprintf(fp,"%12.8f",Sd(i,j)));
                        }
printf("%d",fprintf(fp,"\n"));
         }
//@//


/* DETERMINAÇÃO DOS PARÂMETROS QUE PARTICIPAM DO LOOPING */
/* DETERMINAÇÃO DE nd */

/* MULTIPLICAÇÃO DE B1*X */

//clrscr();
//printf("VALORES DE B1X\n");

b1x=TM(m*1); *b1x=0;
aux1=0;
for(i=0;i<m;i++)
        {
         for(j=0;j<1;j++)
                        {
                         for(z=0;z<n1;z++)
                                {
                                        ant=B1(i,z); mat=X(z,j);
                                        aux1+=B1(i,z)*X(z,j);
                                }
                                        B1X(i,j)=aux1;
 //printf("%8.6f",B1X(i,j));
                                        aux1=0;
                        }
 //printf("\n");
        }
 //tt=getche();

 /* COMPLETANDO AS LINHAS DE D1inv */

 u2i=TM(rd*p); *u2i=0;
 ybx=TM(p*1); *ybx=0;
 nd=TM(blo*1); *nd=0;
 ndi=TM(rd*1); *ndi=0;
 ddd=TM(rd*rd); *ddd=0;
```

```
for(i=0;i<rd;i++)
        for(j=0;j<p;j++)
                if(j<rd)
                        {
                                U2I(i,j)= U2(i,j);
                                mat=U2I(i,j);
                        }
                        else if(j>rd-1)

                                                {
                                                        U2I(i,j)=0;
                                                        mat=U2I(i,j);
                                                }

/* MULTIPICAÇÃO DE YHT*B1X */

aux1=0;

//clrscr();
//printf("VALORES DE YHTB1X\n");
for(i=0;i<p;i++)
        {
         for(j=0;j<1;j++)
                        {
                         for(z=0; z<m;z++)
                                {
                                        ant=YHT(i,z); mat=B1X(z,j);
                                        aux1+=YHT(i,z)*B1X(z,j);
                                }
                                        YHTB1X(i,j)=aux1;
//printf("%6.4f",YHTB1X(i,j));
                                        mat=YHTB1X(i,j);
                                        aux1=0;
                        }
//printf("\n");
  }
//tt=getche();

/* MULTIPLICAÇÃO DE U2I*YHTB1X */

//printf("\nVALORES DE Ndi\n");

aux1=0;
for(i=0;i<rd;i++)
        {
         for(j=0;j<1;j++)
                {
                        for(z=0;z<p;z++)
                                {
                                ant=U2I(i,z); mat=YHTB1X(z,j);
                                aux1+=U2I(i,z)*YHTB1X(z,j);
                                }
                                Ndi(i,j)=-aux1;
//printf("%8.4f",Ndi(i,j));
                                mat=Ndi(i,j);
                                aux1=0;
         }
//printf("\n");
  }
//tt=getche();

/* COMPLEMENTO DE Nd ATÉ CHEGAR A blo */

for(j=0,i=0;i<blo;i++)
        {
         Nd(i,j)=(i<rd ? Ndi(i,j) : 0);
         mat=Nd(i,j);
        }

free(ndi);
free(d1);

/* CÁLCULO DE D1inv *D3 */
```

```
for(aux1=0,i=0;i<rd;i++)
        for(j=0;j<td;j++)
                {
                        for(z=0;z<rd;z++)
                                {
                                ant=U2(i,z); mat=D3(z,j);
                                aux1+=U2(i,z)*D3(z,j);
                                }
                                D1D3(i,j)=-aux1;
                                mat=D1D3(i,j);
                                aux1=0;
                }

/* MULTIPLICAÇÃO DE D1D3*ZE */

nii=TM(p*1); *nii=0;
ze=TM(blo*1); *ze=0;
dze=TM(blo*1); *dze=0;
dzei=TM(blo*blo); *dzei=0;
n=TM(blo*1); *n=0;
comp=TM(blo*blo); *comp=0;

for(j=0,i=0;i<td;i++)                                   **********************************
        ZE(i,j)=0.5;                                      ******** CHUTE INICIAL*********
                                                          **********************************

for(i=0;i<rd;i++)
        for(j=0;j<td;j++)
                {
                        D1D3ZEi(i,j)=D1D3(i,j);
                        mat=D1D3ZEi(i,j);
                }

for(i=0;i<td;i++)
        for(j=0;j<td;j++)
                {
                        COMP(i,j)=(i==j ? 1 : 0);
                        mat=COMP(i,j);
                }

for(i=0;i<blo;i++)
        for(j=0;j<td;j++)
                {
                        if(i<rd)
                                {
                                D1D3ZEi(i,j)=D1D3ZEi(i,j);
                                mat=D1D3ZEi(i,j);
                                }
                        if(i>rd-1)
                                {
                                D1D3ZEi(i,j)=COMP(i-rd,j);
                                mat=D1D3ZEi(i,j);
                                }
                }

aux1=0;
for(i=0;i<blo;i++)
        for(j=0;j<1;j++)
                {
                        for(z=0;z<td;z++)
                                {
                                ant=D1D3ZEi(i,z); mat=ZE(z,j);
                                aux1+=D1D3ZEi(i,z)*ZE(z,j);
                                }
                                D1D3ZE(i,j)=aux1;
                                mat=D1D3ZE(i,j);
                                aux1=0;
                }

free(comp);
free(d3);

/* SOMA DE Nd E D1D3ZE */
```

```
for(j=0,i=0;i<blo;i++)
            {
             ant=Nd(i,j); mat=D1D3ZE(i,j);
             Ni(i,j)=Nd(i,j)+ D1D3ZE(i,j);
             mat=Ni(i,j);
             }

free(dzei);
free(ddd);
free(ze);
free(nd);

/* DETERMINAÇÃO DO No */

noo=TM(p*p); *noo=0;
no=TM(n2*n2); *no=0;
nol=TM(n2*1); *nol=0;


for(ki=0,i=0;i<ni;i++)
          for(j=0;j<c1;j++)
                    {
                              if((viv[i*c1+j+rq]==2) || (viv[i*c1+j+rq]==100))
                                        {
                                         vivm[ki]=viv[i*c1+j+rq];
                                         mat=vivm[i*c1+j+rq];ki++;
                                         }
                    }

for(kj=0,ki=0,i=0;i<blo;i++)
            {
             for(j=0;j<c1;j++)
                              {
                               if(vivm[i*c1+j]==2)
                                        {
                                         ant=Ni(ki,0);
                                         Nol(kj,0)=Ni(ki,0);
                                         mat=Nol(kj,0);kj++;
                                         }
                               } ki++;
            }

//clrscr();
//printf("VALORES DE No\n");
for(i=0;i<n2;i++)
            {
             for(j=0;j<n2;j++)
                              {
                               No(i,j)=(i==j ? (Nol(i,0)) : 0);
//printf("%8.4f",No(i,j));
                               mat=No(i,j);
                              }
//printf("\n");
//tt=getche();
             }
//tt=getche();

/* DETERMINAÇÃO DOS PARÂMETROS NÃO ITERATIVOS */
/* DETERMINAÇÃO DOS PARÂMENTROS DE HH */

ytb=TM(p*n1); *ytb=0;
sbt=TM(n1*m); *sbt=0;
h1=TM(p*m); *h1=0;
hy=TM(p*p); *hy=0;
bty=TM(n2*p); *bty=0;

/* MULTIPLICAÇÃO DE YHT*B1 */

aux1=0;
for(i=0;i<p;i++)
          for(j=0;j<n1;j++)
                    {
                              for(z=0;z<m;z++)
                                        {
                                         ant=YHT(i,z); mat=B1(z,j);
```

```
                                        aux1+=YHT(i,z)*B1(z,j);
                                        }
                                        YHTB1(i,j)=aux1;
                                        mat=YHTB1(i,j);
                                        aux1=0;
                        }

clrscr();

/* MULTIPLICAÇÃO DE S*B1T */

aux1=0;
for(i=0;i<n1;i++)
        for(j=0;j<nn;j++)
                {
                        for(z=0;z<n1;z++)
                                {
                                ant=S(i,z);  mat=B1(j,z);
                                aux1+=S(i,z)*B1(j,z);
                                }
                                S1B1T(i,j)=aux1;
                                mat=S1B1T(i,j);
                                aux1=0;
                        }

/* MULTIPLICAÇÃO DE YHTB1 *S1B1T */

aux1=0;
for(i=0;i<p;i++)
        for(j=0;j<m;j++)
                {
                        for(z=0;z<n1;z++)
                                {
                                ant=YHTB1(i,z); mat=S1B1T(z,j);
                                aux1+=YHTB1(i,z)*S1B1T(z,j);
                                }
                                H1(i,j)=aux1;
                                mat=H1(i,j);
                                aux1=0;
                        }

/* MULTIPLICAÇÃO DE H1*Y */

aux1=0;
for(i=0;i<p;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                ant=H1(i,z); mat=YHT(z,j);
                                aux1+=H1(i,z)*YHT(z,j);
                                }
                                H1Y(i,j)=aux1;
                                mat=H1Y(i,j);
                                aux1=0;
                        }

/* MULTIPLICAÇÃO DE B2T*Y */

for(i=0;i<m;i++)
        for(j=0;j<n2;j++)
                mat=B2(i,j);

aux1=0;
for(i=0;i<n2;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                ant=B2(z,i); mat=YHT(j,z);
                                aux1+=B2(z,i)*YHT(j,z);
                                }
                                B2TY(i,j)=aux1;
                                mat=B2TY(i,j);
                                aux1=0;
                        }
```

```
free(pps);
free(h1);
free(ytb);

for(i=0;i<n2;i++)
        for(j=0;j<p;j++)
                mat=B2TY(i,j);

/* DETERMINAÇÃO DE ALGUNS PARÂMETROS NÃO ITERATIVOS DE a E lb */

zy=TM(pd*m); *zy=0;
zb=TM(pd*1); *zb=0;
s1b=TM(n1*p); *s1b=0;
aa=TM(n1*pd); *aa=0;

aux1=0;
for(i=0;i<pd;i++)
        for(j=0;j<m;j++)
                {
                        for(z=0;z<p;z++)
                                {
                                ant=ZHT(i,z); mat=YHT(z,j);
                                aux1+=ZHT(i,z)*YHT(z,j);
                                }
                                ZTYT(i,j)=aux1;
                                mat=ZTYT(i,j);
                                aux1=0;
                }


/* MULTIPLICAÇÃO DE ZTYT*B1X */

aux1=0;
for(i=0;i<pd;i++)
        for(j=0;j<1;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                        ant=ZTYT(i,z); mat=B1X(z,j);
                                        aux1+=ZTYT(i,z)*B1X(z,j);
                                }
                                Zb(i,j)=aux1;
                                mat=Zb(i,j);
                                aux1=0;
                        }

/* DETERMINAÇÃO DE ALGUNS PARÂMETROS DE a */
/* MULTIPLICAÇÃO DE S1B1T*Y */

aux1=0;
for(i=0;i<n1;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                ant=S1B1T(i,z); mat=YHT(j,z);
                                aux1+=S1B1T(i,z)*YHT(j,z);
                                }
                                S1B1TY(i,j)=aux1;
                                mat=S1B1TY(i,j);
                                aux1=0;
                }

//@//
for(j=0,i=0;i<pd;i++)
        {
        if(fabs(Zb(i,j))<0.0001) //para erros maiores que 0.01%
                Zb(i,j)=0;
printf("%d",fprintf(fp,"%12.8f",Zb(i,j)));
printf("%d",fprintf(fp,"\n"));
        }

for(i=0;i<n1;i++)
        {
        for(j=0;j<m;j++)
                        {
```

```
printf("%d",fprintf(fp,"%12.8f",S1B1T(i,j)));
                                        }
printf("%d",fprintf(fp,"\n"));
                }

/*
for(i=0;i<n2;i++)
                {
                 for(j=0;j<n2;j++)
                                        {
printf("%d",fprintf(fp,"%12.8f",No(i,j)));
                                        }
printf("%d",fprintf(fp,"\n"));
                }
*/

//@//

aux1=0;
for(i=0;i<n1;i++)
                for(j=0;j<pd;j++)
                .                       {
                                                for(z=0;z<p;z++)
                                                        {
                                                        ant=S1B1TY(i,z); mat=ZHT(j,z);
                                                        aux1+=S1B1TY(i,z)*ZHT(j,z);
                                                        }
                                                        Aa(i,j)=aux1;
                                                        mat=Aa(i,j);
                                                        aux1=0;
                                }

/* DETERMINAÇÃO DE ALGUNS PARÂMETROS DE Ndelta */
/* DETERMINAÇÃO DE B2TY*Z */

nl=TM(n2*pd); *nl=0;

aux1=0;
for(i=0;i<n2;i++)
                for(j=0;j<pd;j++)
                                {
                                                for(z=0;z<p;z++)
                                                        {
                                                        aux1+=B2TY(i,z)*ZHT(j,z);
                                                        }
                                                        Ndel(i,j)=aux1;
                                                        aux1=0;
                                                }

free(sbt);
free(s1b);
free(zy);
d_=TM(p*p); *d_=0;

*****************************************************************************************************************************
 INSERÇÃO DA SUBROTINA DE CÁLCULO DE SISTEMAS LINEARES ATRAVÉS DO SVD
*****************************************************************************************************************************

us=TM(p*blo); *us=0; ws=TM(blo*1); *ws=0;
wsm=TM(blo*blo); *wsm=0; vts=TM(blo*blo); *vts=0;
daux=TM(blo*blo); *daux=0; wm=vector(1,NP);
ai=matrix(1,MP,1,NP); //recebe a matriz original
um=matrix(1,MP,1,NP);
v=matrix(1,NP,1,NP);
mm=p; nm=blo;

/* LEITURA DOS VALORES VINDOS DO PROGRAMA PRINCIPAL PARA A SUBROTINA SVD */

for(i=1;i<=mm;i++)
                for(j=1;j<=nm;j++)
                                {
                                                ant=DAF(i-1,j-1);
                                                ai[i][j]=DAF(i-1,j-1);
                                                um[i][j]=DAF(i-1,j-1);
                                                mat=um[i][j];
                                }
```

```
/* DETERMINAÇÃO DAS MATRIZES DECOMPOSTAS */

*****************************************************************************************************************************
svdcmp(um,mm,nm,wm,v);
*****************************************************************************************************************************

/* MATRIZ u */

for(i=1;i<=mm;i++)
        for(j=1;j<=nm;j++)
                US(i-1,j-1)=um[i][j];

//clrscr();
//printf("MATRIZ US\n");
for(i=0;i<mm;i++)
        {
         for(j=0;j<nm;j++)
                                {
//printf("%10.6f",US(i,j));
//printf("  ");
                                }
//printf("\n");
        }
//tt=getche();

//MATRIZ DIAGONAL w

for(i=1;i<=nm;i++)
        for(j=1;j<=1;j++)
                WS(i-1,j-1)=wm[i];

//clrscr();

//printf("MATRIZ TRANSPOSTA v\n");

/* MATRIZ TRANSPOSTA v-transposta */

for(i=1;i<=nm;i++)
        for(j=1;j<=nm;j++)
                VTS(i-1,j-1)=v[j][i];

for(i=0;i<nm;i++)
        {
         for(j=0;j<nm;j++)
                                {
//printf("%10.6f",VTS(i,j));
//printf(" ");
                                }
//printf("\n");
         }
//              tt=getche();

//clrscr();

for(i=0;i<nm;i++)
        for(j=0;j<nm;j++)
                {
                        if(WS(i,0)!=0)
                                WSM(i,j)=(i==j ? (1/WS(i,0)) : 0);
                        else
                                WSM(i,j)=WS(i,0);
                }

//printf("MATRIZ WSM\n");
for(i=0;i<nm;i++)
        {
         for(j=0;j<nm;j++)
                                {
//printf("%10.6f",WSM(i,j));
//printf(" ");
                                }
//printf("\n");
         }

//tt=getche();
```

```
for(i=0;i<nm;i++)
        for(j=0;j<nm;j++)
                {
                        for(z=0;z<nm;z++)
                                {
                                a1+=VTS(z,i)*WSM(z,j);
                                }
                                DAUX_(i,j)=a1;
                                a1=0;
                }
//tt=getche();

//clrscr();
/*
for(i=0;i<nm;i++)
        {
         for(j=0;j<nm;j++)
//printf("%12.6f",DAUX_(i,j));
//printf("\n");
        }
//tt=getche();
*/

for(i=0;i<nm;i++)
        for(j=0;j<mm;j++)
                {
                        for(z=0;z<nm;z++)
                                {
                                ant=DAUX_(i,z); mat=US(j,z);
                                a1+=DAUX_(i,z)*US(j,z);
                                mat=a1;
                                }
                                D_(i,j)=a1;
                                mat=D_(i,j);
                                a1=0;
                }

//clrscr();
//printf("TRANSPOSTA DA MATRIZ D_\n");
for(i=0;i<p;i++)
        {
         for(j=0;j<blo;j++)
                        {
//printf("%10.6f",D_(j,i));
//printf(" ");
                        }
//printf("\n");
         }
//tt=getche();
clrscr();

/* A PARTIR DAQUI, TEM-SE A MATRIZ D DECOMPOSTA PARA RESOLUÇÃO DO SISTEMA LINEAR  FINAL */

free_matrix(v,1,NP,1,NP);
free_matrix(um,1,MP,1,NP);
free_matrix(ai,1,MP,1,NP);

**********************************************FIM DO SVD**********************************************
clrscr();
printf("VALORES DE N ESPERADOS (sem ajuste)\n");
nsaj=TM(blo*1); *nsaj=0;

for(i=0;i<blo;i++)
        {
         for(j=0;j<1;j++)
                        {
                         for(z=0;z<p;z++)
                                {
                                        aux1+=D_(i,z)*YHTB1X(z,j);
                                }
                                        N_saj(i,j)=-aux1;
printf("  %8.4f",N_saj(i,j));
                                        aux1=0;
                        }
        }
printf("\n\n");
```

```
ndt=TM(n2*1); *ndt=0;

free(u2);
free(add);
free(df);
free(nsaj);
free(ybx);

*****************************************************************************************************************************
*****************************************************************************************************************************
*****************************************COMECO DO LOOPING*******************************************************************
*****************************************************************************************************************************
*****************************************************************************************************************************

iter=0;
niter=10;  // ÉMAIS QUE SUFICIENTE PARA A CONVERGÊNCIA (OCORRE EM TRÊS OU QUATRO PASSAGENS)

do
{ //ABRE O LOOPING

snf=0.0;
aux7=0.0;

ss=TM(n2*n2);
ns=TM(n2*n2);
bs=TM(m*n2);
bb=TM(m*p);

for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                ant=No(i,j);

for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                ant=Sd(i,j);

/* MULTIPLICAÇÃO DE No*Sd */

aux=6; mat=0; ant=0;
for(i=0;i<m;i++)
        for(j=0;j<n2;j++)
                {
                        NoSd(i,j)=0;
                        mat=NoSd(i,j);
                }
                        aux1=0;

for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                {
                        for(z=0;z<n2;z++)
                                {
                                ant=No(i,z);  mat=Sd(z,j);
                                aux1+=No(i,z)*Sd(z,j);
                                }
                                mat=aux1;
                                NoSd(i,j)=aux1;
                                mat=NoSd(i,j);
                                aux1=0;
                        }

/* MULTIPLICAÇÃO DE NoSd*No */

aux1=0; ant=0; mat=0;
for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                {
                        for(z=0;z<n2;z++)
                                {
                                ant=NoSd(i,z);  mat=No(z,j);
                                aux1+=NoSd(i,z)*No(z,j);
                                }
                                S2(i,j)=aux1;
                                mat=S2(i,j);
                                aux1=0;
                }
```

```
for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                mat=S2(i,j);

/*
printf("\nVALORES DE S2\n");
for(i=0;i<n2;i++)
        {
         for(j=0;j<n2;j++)
printf("%10.4f",S2(i,j));
printf("\n");
        }
tt=getche();
*/

free(ns);

/* DETERMINAÇÃO DE HH */
/* MULTIPLICAÇÃO DE B2*S2 */

mat=0;
ant=0;
for(i=0;i<m;i++)
        for(j=0;j<n2;j++)
                mat=B2(i,j);

aux1=0;
for(i=0;i<m;i++)
        for(j=0;j<n2;j++)
                {
                        for(z=0;z<n2;z++)
                                {
                                ant=B2(i,z);  mat=S2(z,j);
                                aux1+=B2(i,z)*S2(z,j);
                                }
                                B2S2(i,j)=aux1;
                                mat=B2S2(i,j);
                                aux1=0;
                }

/* MULTIPLICAÇÃO DE B2S2*B2TY */

aux1=0;
for(i=0;i<m;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<n2;z++)
                                {
                                ant=B2S2(i,z);  mat=B2TY(z,j);
                                aux1+=B2S2(i,z)*B2TY(z,j);
                                }
                                BBTY(i,j)=aux1;
                                mat=BBTY(i,j);
                                aux1=0;
                        }

free(bs);

zht=TM(pd*p); za=TM(pd*pd);
hyy=TM(p*p); hh=TM(p*p);

/* MULTIPLICAÇÃO DE YHT*BBTY */

aux1=0;
for(i=0;i<p;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                ant=YHT(i,z);  mat=BBTY(z,j);
                                aux1+=YHT(i,z)*BBTY(z,j);
                                }
                                H2Y(i,j)=aux1;
                                mat=H2Y(i,j);
                                aux1=0;
                }
```

```
/* DETERMINAÇÃO DE HH */

for(i=0;i<p;i++)
        for(j=0;j<p;j++)
                {
                        ant=H1Y(i,j);  mat=H2Y(i,j);
                        HH(i,j)=H1Y(i,j)+H2Y(i,j);
                        mat=HH(i,j);
                }

/* DETERMINÇÃO DE lb */
/* DETERMINAÇÃO DE (ZTHZ)inv */
/* MULTIPLICAÇÃO DE ZHT*HH */

free(hyy);
free(bb);

/*
//@//
if(iter==9)
        {
        for(i=0;i<p;i++)
                {
                        for(j=0;j<p;j++)
                                {
printf("%d",fprintf(fp,"%18.4f",HH(i,j)));
                                }
printf("%d",fprintf(fp,"\n"));
                }
        }
//@//
*/

aux1=0;
for(i=0;i<pd;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<p;z++)
                                {
                                ant=ZHT(i,z);  mat=HH(z,j);
                                aux1+=ZHT(i,z)*HH(z,j);
                                }
                                ZTH(i,j)=aux1;
                                mat=ZTH(i,j);
                                aux1=0;
                }

/* MULTIPLICAÇÃO DE ZTH*Z */

aux1=0;
for(i=0;i<pd;i++)
        {
        for(j=0;j<pd;j++)
                {
                for(z=0;z<p;z++)
                        {
                                ant=ZTH(i,z);  mat=ZHT(j,z);
                                aux1+=ZTH(i,z)*ZHT(j,z);
                        }
//if((fabs(aux1))<0.0000001)
//Za(i,j)=0;
//else
        Za(i,j)=aux1;
                if(iter==niter-1)
//printf("  Za=%12.8f",Za(i,j));
        aux1=0;
        }
//if(iter==niter-1)
//printf("\n");
 }
//tt=getche();
free(zht);

for(i=0;i<pd;i++)
        for(j=0;j<pd;j++)
                mat=Za(i,j);
```

```
zai=TM(pd*pd);
izd=TMI(p);

for(i=0;i<pd;i++) izd[i]=i;

for(i=0;i<pd;i++)
        for(j=0;j<pd;j++)
                {
                        Zainv(i,j)=(i==j ? 1 : 0);
                        mat=Zainv(i,j);
                }

for(i=0;i<pd;i++)
        for(j=0;j<pd;j++)
                mat=Za(i,j);

*****************************************************************************************************************************
lu(pd,izd,za);
backs(pd,pd,izd,za,zai);
*****************************************************************************************************************************

/*
clrscr();
//if(iter==niter-1)
{
printf("VALORES DE Zainv\n");
for(i=0;i<pd;i++)
        {
         for(j=0;j<pd;j++)
printf("%12.8f",Zainv(i,j));
        }
tt=getche();
}
*/

/* DETERMINAÇÃO DE lb */

free(za); free(izd);

lb=TM(pd*1); aj2=TM(n1*1);

for(aux1=0,i=0;i<pd;i++)
        for(j=0;j<1;j++)
                {
                        for(z=0;z<pd;z++)
                                {
                                ant=Zainv(i,z);  mat=Zb(z,j);
                                aux1+=Zainv(i,z)*Zb(z,j);
                                }
                                LB(i,j)=-aux1;
                                mat=LB(i,j);
                                aux1=0;
                }

/* DETERMINAÇÃO DE a */
/* MULTIPLICAÇÃO DE Aa*LB */

aux1=0;
//clrscr();
//printf("VALORES DE AJ2\n");
for(i=0;i<n1;i++)
        for(j=0;j<1;j++)
                {
                        for(z=0;z<pd;z++)
                                {
                                ant=Aa(i,z);  mat=LB(z,j);
                                aux1+=Aa(i,z)*LB(z,j);
                                }
                                AJ2(i,j)=(aux1);
                                        if(porc==1)
                                                        AJ2(i,j)=AJ2(i,j)*0.01; //para os casos dados
//printf("AJ(%d)=%12.8f",i,AJ2(i,j));      //em porcentagem
                                                        aux1=0;
//printf("\n");
                }
//printf("\n");
```

```
/*
//@//
if(iter==9){
for(i=0;i<pd;i++)
                {
                 for(j=0;j<pd;j++)
                                        {
printf("%d",fprintf(fp,"%10.4f",Zainv(i,j)));
                                        }
printf("%d",fprintf(fp,"\n"));

                 }

for(j=0,i=0;i<n1;i++)
                {
printf("%d",fprintf(fp,"%10.3f",AJ2(i,j)));
printf("%d",fprintf(fp,"\n"));
                }
}

//@//
*/

/* DETERMINAÇÃO DE Ndelta */
/* MULTIPLICAÇO DE S2*Ndel */

sdl=TM(n2*pd);

aux1=0;
for(i=0;i<n2;i++)
                for(j=0;j<pd;j++)
                                {
                                        for(z=0;z<n2;z++)
                                                {
                                                ant=S2(i,z);  mat=Ndel(z,j);
                                                aux1+=S2(i,z)*Ndel(z,j);
                                                }
                                                S2Ndel(i,j)=aux1;
                                                mat=S2Ndel(i,j);
                                                aux1=0;
                                        }

/* MULTIPLICAÇÃO DE S2Ndel*LB */

aux1=0;
//clrscr();
//printf("VALORES DE Ndelta\n");
for(i=0;i<n2;i++)
                for(j=0;j<1;j++)
                                {
                                        for(z=0;z<pd;z++)
                                                {
                                                ant=S2Ndel(i,z);  mat=LB(z,j);
                                                aux1+=S2Ndel(i,z)*LB(z,j);
                                                }
                                                Ndelta(i,j)=(aux1);
                                                if(porc==1)
                                                Ndelta(i,j)=Ndelta(i,j)*0.01; //dados em porcentagem
//printf("Ndelta(%d)=%12.8f",i,Ndelta(i,j));
                                                aux1=0;
printf("\n");
                                }
//tt=getche();

/* DETERMINAÇÃO DE B1AJ2 */

free(ss);

ba=TM(m*1);

aux1=0;

//clrscr();
//printf("VALORES DE B1AJ2\n");
```

```
for(i=0;i<m;i++)
        {
        for(j=0;j<1;j++)
                {
                for(z=0;z<n1;z++)
                        {
                                ant=B1(i,z);  mat=AJ2(z,j);
                                aux1+=B1(i,z)*AJ2(z,j);
                        }
                                B1AJ2(i,j)=aux1;
//printf("%8.6f",B1AJ2(i,j));
                                aux1=0;
                }
//printf("\n");
        }
//tt=getche();

/* DETERMINAÇÃO DE B2Ndelta */

bdlt=TM(m*1);

aux1=0;
//clrscr();
//printf("\nVALORES DE B2Ndelta\n");
for(i=0;i<m;i++)
        {
        for(j=0;j<1;j++)
                {
                for(z=0;z<n2;z++)
                        {
                                ant=B2(i,z);  mat=Ndelta(z,j);
                                aux1+=B2(i,z)*Ndelta(z,j);
                        }
                                B2Ndelta(i,j)=aux1;
//printf("B2Ndelta(i)=%8.6f",i,B2Ndelta(i,j));
                                mat=B2Ndelta(i,j);
                                aux1=0;
                }
//printf("\n");
        }
//tt=getche();

/* SOMA DOS TERMOS DA EQUAÇÃO 21 CROWE(1986) */

co=TM(m*1);

for(i=0;i<m;i++)
        for(j=0;j<1;j++)
                {
                        ant=B1X(i,j);
                        ant=B1AJ2(i,j); mat=B2Ndelta(i,j);
                        COL(i,j)=(B1X(i,j)+B1AJ2(i,j)+B2Ndelta(i,j));
                }

free(dc);
free(ndd);
free(bnd);
free(ba);

/* MULTIPLICAÇÃO DE YHT*COL */

yco=TM(p*1);

for(i=0;i<p;i++)
        for(j=0;j<1;j++)
                {
                        for(z=0;z<m;z++)
                                {
                                ant=YHT(i,z);  mat=COL(z,j);
                                aux1+=YHT(i,z)*COL(z,j);
                                }
                                YHTCOL(i,j)=aux1;
                                aux1=0;
                }
```

```
/* MULTIPLICAR Dinv POR YHTCOL */

aux1=0;
for(i=0;i<blo;i++)
        {
        for(j=0;j<1;j++)
                {
                for(z=0;z<p;z++)
                        {
                                ant=D_(i,z); mat=YHTCOL(z,j);
                                aux1+=D_(i,z)*YHTCOL(z,j);
                        }
                                N(i,j)=-aux1;
//printf("N(%d)=%8.6f",i,N(i,j));       //AQUI SAI O n DESEJADO
                                aux1=0;                 //E EQUIVALENTE AO Ni
                }
//printf("\n");
        }
//tt=getche();

if(iter<1)
printf("\nVALORES DE N CALCULADOS\n");

for(j=0,i=0;i<blo;i++)
        {
        mat=N(i,j);
printf("%10.4f" ,N(i,j));
        }
printf("  iter=%d",iter);
printf("\n");

 /* PRECISAMOS TIRAR SOMENTE A PARTE DE n CALCULADO QUE NOS INTERESSA*/

free(yco);
free(co);
free(bdlt);
free(lb);
free(sdl);

for(ki=0,i=0;i<ni;i++)
        for(j=0;j<c1;j++)
                {
                if((vivm[i*c1+j+rq]==2) || (vivm[i*c1+j+rq]==100))
                        {
                        vivm[ki]=viv[i*c1+j+rq];
                        mat=vivm[ki];ki++;
                        }
                }

for(kj=0,ki=0,i=0;i<blo;i++)
        {
        for(j=0;j<c1;j++)
                {
                if(vivm[i*c1+j]==2)
                        {
                        Nol(kj,0)=N(ki,0);
                        mat=Nol(kj,0); kj++;
                        }
                } ki++;
        }

for(j=0,i=0;i<n2;i++)
        {
        mat=Nol(i,j);
        }

for(ki=0,i=0;i<n2;i++)
        {
        for(j=0;j<n2;j++)
                {
                No(i,j)=(i==j ? Nol(ki,0) : 0);
                mat=No(i,j);
                } ki++;
        }

iter++;
```

```
} //FECHA O LOOPING

while(((fabs(aux7))>limite*snf) || (iter<niter));

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
##############################################FIM DO LOOPING#############################################
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

tt=getche(); //para se ver os valores de N

free(ppd);free(b1x);free(b1);free(bty);free(zh);free(hy);free(nl);free(aa);free(zb);

/*
clrscr();
if(iter==niter-1)
{
 printf("VALORES DE Zainv\n");
 for(i=0;i<pd;i++)
                {
                for(j=0;j<pd;j++)
                                printf("%12.8f",Zainv(i,j));
                }
tt=getche();
}
*/

xd=TM(n1*1);

clrscr();
printf("VALORES DE X RECONCILIADOS\n");
for(j=0,i=0;i<n1;i++)
                {
                ant=XD(i,j); mat=AJ2(i,j);
                XD(i,j)=X(i,j);
                XD(i,j)+=AJ2(i,j);
printf("X(%d)=%8.4f",i,XD(i,j));
printf("\n");
                }
tt=getche();

/*
printf("%d",fprintf(fp,"\nCORRENTE-COMPONENTE(CC)/VAZOES MEDIDAS(VM)/VAZOES CORRIGIDAS(VC)\n\n"));
printf("%d",fprintf(fp,"CC"));
printf("%d",fprintf(fp," ....."));
printf("%d",fprintf(fp,"VM"));
printf("%d",fprintf(fp,"....."));
printf("%d",fprintf(fp,"VC"));
printf("%d",fprintf(fp,"....."));
printf("%d",fprintf(fp,"a"));
printf("%d",fprintf(fp,"....."));
printf("%d",fprintf(fp,"Er(%)"));
printf("%d",fprintf(fp,"\n\n"));
for(i=0;i<n1;i++)
                {
                 {
                                for(j=0;j<3;j++)
 printf("%d",fprintf(fp,"%c",CM(i,j)));
 printf("%d",fprintf(fp," "));
 printf("%d",fprintf(fp,"%10.4f",X(i,0)));
 printf("%d",fprintf(fp,"%10.4f",XD(i,0)));
 printf("%d",fprintf(fp,"%10.4f",AJ2(i,0)));
 printf("%d",fprintf(fp,"%10.4f",(((X(i,0)-XD(i,0))*100)/X(i,0))));
                                }
 printf("%d",fprintf(fp,"\n"));
                }
 */

 /*
 printf("%d",fprintf(fp,"\n\n"));
 printf("%d",fprintf(fp,"\nVAZOES CORRIGIDAS\n\n"));
 for(j=0,i=0;i<q;i++)
 printf("%d",fprintf(fp,"%10.4f",XD(i,j)));
 */

 //@//
```

```
for(i=0;i<n2;i++)
        {
        for(j=0;j<n2;j++)
                {
printf("%d",fprintf(fp,"%12.8f",No(i,j)));
                }
printf("%d",fprintf(fp,"\n"));
        }

for(i=0;i<p;i++)
        {
        for(j=0;j<p;j++)
                {
printf("%d",fprintf(fp,"%10.6f",HH(i,j)));
                }
printf("%d",fprintf(fp,"\n"));
        }

for(i=0;i<pd;i++)
        {
        for(j=0;j<pd;j++)
                {
printf("%d",fprintf(fp,"%16.8f",Zainv(i,j)));
                }
printf("%d",fprintf(fp,"\n"));
        }

for(j=0,i=0;i<n1;i++)
        {
printf("%d",fprintf(fp,"%12.8f",AJ2(i,j)));
printf("%d",fprintf(fp,"\n"));
        }

//@//
free(hh);free(b2);free(zai);free(aj2);free(x);

np=TM(n2*n2);

for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                {
                        mat=No(i,j);
                }

for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                {
                        ant=No(i,j);
                        Np(i,j)=No(i,j);
                        mat=Np(i,j);
                }

free(nol);free(no);free(xd);

ninv=TM(n2*n2);

for(i=0;i<n2;i++)
        for(j=0;j<n2;j++)
                {
                        Ninv(i,j)=(i==j ? 1 : 0);
                        mat=Ninv(i,j);
                }

/* DETERMINAÇÃO DO AJUSTE DE DELTA */
dlt=TM(n2*1); *dlt=0;
dde=TM(n2*1); *dde=0;
dd=TM(n2*1); *dd=0;
ind=TMI(n2); *ind=0;

for(i=0;i<n2;i++) ind[i]=i;

*****************************************************************************************************************
lu(n2,ind,np);
backs(n2,n2,ind,np,ninv);
*****************************************************************************************************************
```

```
/* DETERMINAÇÃO DE DELTA */

free(np);
free(ind);

clrscr();
for(i=0;i<n2;i++)
        {
        for(j=0;j<1;j++)
                {
                for(z=0;z<n2;z++)
                        {
                                ant=Ninv(z,i); mat=Ndelta(z,j);
                                aux1+=Ninv(z,i)*Ndelta(z,j);
                        }
                                Delta(i,j)=aux1;
//printf("Delta(%d)=%12.8f",i,Delta(i,j));
                                aux1=0;
                }
//printf("\n");
        }
//tt=getche();

for(j=0,i=0;i<n2;i++)
        mat=Delta(i,j);

//@//
for(j=0,i=0;i<n2;i++)
        {
printf("%d",fprintf(fp,"%12.8f",Delta(i,j)));
printf("%d",fprintf(fp,"\n"));
        }
//@//

/*
for(i=0;i<n1;i++)
        {
        for(j=0;j<3;j++)
printf("%d", fprintf(fp," %c",CM(i,j)));
printf("%d", fprintf(fp," "));
        }

//@//
*/
//printf("TESTE\n");
ki=0;
for(i=0;i<c1;i++)
        for(j=0;j<blo;j++)
                {
                if(DE(j,i))
                        {
                        ant=DE(j,i);
                        DD(ki,0)=DE(j,i);
//printf("%8.4f",DD(i,j));
                        mat=DD(ki,0);
                        DDE(ki,0)=DE(j,i);
                        mat=DDE(ki,0);
                        ki++;
                        }
//printf("\n");
                }
//tt=getche();

/* DETERMINAÇÃO DE d */
//printf("VALORES MANDADOS PARA O AJUSTE DE d\n");
for(i=0;i<n2;i++)
        {
        for(j=0;j<1;j++)
                {
//printf("  d_s/ajuste(%d)=%8.4f",i,DD(i,j));
//printf("    Delta(%d)=%12.8f",i,Delta(i,j));
                DD(i,j)+=Delta(i,j);
//printf("  d_ajust(%d)=%8.4f",i,DD(i,j));
                }
//printf("\n");
        }
```

```
def=TM(c1*blo); *def=0;

//clrscr();
//printf("VALORES DE d RECONCILIADOS\n\n");
for(k=0,i=0;i<c1;i++)
        {
                for(j=0;j<blo;j++)
                        {
                        mat=DD(k,0);
                        DEF(i,j)=DD(k,0);
//printf("d(%d)=%8.4f",k,DEF(i,j));
                        k++;
//printf("\n");
                        }
                }

for(k=0,i=0;i<blo;i++)
        for(j=0;j<c1;j++)
                {
                        mat=DEF(j,i);
                        DD(k,0)=DEF(j,i);
//printf("\nDD(%d,%d)=%8.4f",j,i,DEF(j,i));
                        k++;
                }
//tt=getche();

//@//
for(i=0;i<n2;i++)
        {
        for(j=0;j<1;j++)
                {
printf("%d",fprintf(fp,"%8.4f",DD(i,0)));
                }
printf("%d",fprintf(fp,"\n"));
        }

free(def);

for(j=0;j<(nic1+rq);j++)
printf("%d",fprintf(fp," %d",viv[j]));
printf("%d",fprintf(fp,"\n"));

//@//

free(yh);
free(hnd);
free(ndt);

for(i=0;i<blo;i++)
        for(j=0;j<blo;j++)
                if(i==j)
                        {
                                Noo(i,0)=Noo(i,j);
                        }

for(j=0,i=0;i<blo;i++)
        mat=N(i,j);

/*
printf("%d",fprintf(fp,"\n\nCONCENTRACOES MEDIDAS(CME)/CONCENTRACOES CORRIGIDAS(CC0R)\n\n"));
printf("%d",fprintf(fp,"CC"));
printf("%d",fprintf(fp,"........"));
printf("%d",fprintf(fp,"CME"));
printf("%d",fprintf(fp,"........"));
printf("%d",fprintf(fp,"CC0R"));
printf("%d",fprintf(fp,"......"));
printf("%d",fprintf(fp,"Delta"));
printf("%d",fprintf(fp,"......"));
printf("%d",fprintf(fp,"Er(%)"));
printf("%d",fprintf(fp,"\n\n"));
for(ki=0,i=0;i<n2;i++)
        {
        for(j=0;j<3;j++)
                {
fprintf(fp,"%c",DM(i,j));
                }
```

```
printf("%d",fprintf(fp,"  "));

printf("%d",fprintf(fp,"%10.4f",DDE(i,ki)));

printf("%d",fprintf(fp,"  "));

printf("%d",fprintf(fp,"%10.4f",DD(i,ki)));

printf("%d",fprintf(fp,"%10.4f",Delta(i,ki)));

printf("%d",fprintf(fp,"%10.4f",(DDE(i,ki)-DD(i,ki))*100/(DDE(i,ki))));

printf("%d",fprintf(fp,"\n"));
}

printf("\%d",fprintf(fp,"\n\n"));
printf("%d",fprintf(fp,"\nVAZOES DETERMINADAS-(NAO MEDIDAS)\n\n"));
for(j=0,i=0;i<blo;i++)
        {
printf("%d",fprintf(fp,"%10.4f",N(i,j)));
printf("%d",fprintf(fp,"  "));
        }
printf("\n\n\n\n");
printf("%d",fprintf(fp,"\nTESTE COM OS VALORES DE B2\n"));
for(i=0;i<m;i++)
        {
        for(j=0;j<n2;j++)
                {
printf("%d",fprintf(fp,"%8.2f",B2(i,j)));
                }
printf("%d",fprintf(fp,"\n"));
        }
*/

free(n);
free(noo);
free(ninv);

/*
printf("%d",fprintf(fp,"\n\n"));
printf("%d",fprintf(fp,"\nVALORES CORRIGIDOS DAS CONCENTRACOES\n\n"));
for(j=0,i=0;i<n2;i++)
printf("%d",fprintf(fp,"%10.4f",DD(i,j)));
*/

printf("\n\n\n\n\n\n");
printf("TECLE ALGUMA LETRA PARA RETORNAR AO MENU PRINCIPAL");
tt=getchar();

free(dd);
free(de);
ant+=mat+aux;

}//FIM DA PARTE NAO LINEAR

//free_matrix(v,1,NP,1,NP);
//free_matrix(um,1,MP,1,NP);
//free_matrix(a,1,MP,1,NP);
//free_matrix(a,1,MP,1,NP);
//free_vector(wm,1,NP);

#undef NRANSI

} //FECHA O do

fclose(fp);

} //FECHA A 'FUNCTION' RECONCILIAR

*****************************************************************************************************************
COMECO DAS SUBROTINAS UTILIZADAS NO PROGRAMA
*****************************************************************************************************************

/* SUBROTINA DE TRIANGULARIZAÇÃO */

int lu(int rf,int *irf,float *p1f)
```

```
{ //INÍCIODA SUBROTINA

int r1 = rf-1; register  int i,j,k; float a1,a2; double fabs();

for(irf[r1]=r1,i=0;i<r1;i++)
        {
        for(j=(k=i)+1;j<rf;j++)
                if(fabs(P1f(j,i)) > fabs(P1f(k,i))) k=j;
                        if(k!=i)
                                for(j=i;j<rf;j++)
                                        {
                                        a1=P1f(i,j);
                                        P1f(i,j)=P1f(k,j);
                                        P1f(k,j)=a1;
                                        }
if(!(a2=P1f(i,i), a2))
return(-1);

for(irf[i]=k,j=i+1;j<rf;j++)
        {
        a1=(P1f(j,i)/=-a2);
                for(k=i+1;k<rf;k++)
                        P1f(j,k) += a1 * P1f(i,k) ;
        }
}

if(!P1f(r1,r1))
return (-1) ;
return(0);

} //FIM DA SUBROTINA

##########################################################################################################

/* SUBROTINA DE INVERSÃO DE MATRIZES QUADRADAS */

void backs(int rf,int rs,int *irf,float *p1f,float *uf)

{ //INÍCIO DA SUBROTINA

register int i,j,k;
int r1=rf-1;
float a1,mat,ant;

for(i=0;i<r1;i++)
        {
        if((j=irf[i]) != i)
                for(k=0;k<rs;k++)
                        {
                        a1=Uf(i,k);
                        Uf(i,k)=Uf(j,k);
                        Uf(j,k)=a1;
                        }
                        for(j=i+1;j<rf;j++)
                                for(k=0;k<rs;k++)
                                        Uf(j,k)+=P1f(j,i)*Uf(i,k);
        }
ant=P1f(r1,r1);

for(k=0;k<rs;k++)
        Uf(r1,k)/=P1f(r1,r1);

for(i=r1-1;i>=0;i--)
        {
        for(a1=P1f(i,i),j=i+1;j<rf;j++)
                for(k=0;k<rs;k++)
                        Uf(i,k)-=P1f(i,j)*Uf(j,k);
                                for(k=0;k<rs;k++)
                                        {
                                        Uf(i,k)/=a1;
                                        mat=Uf(i,k);
                                        }
        }
mat+=mat;ant+=ant;
} //FIM DA SUBROTINA
##########################################################################################
```

```
/* SUBROTINA DE PRODUTO DE MATRIZES */

void prod_m(int s,int rf,int w,float *p2f,float *uf,float *puf)

{ //INÍCIO DA SUBROTINA

register int i,j,z;
float a1,mat,ant;

for(a1=0,i=0;i<s;i++)
        for(z=0;z<w;z++)
                {
                        for(j=0;j<rf;j++)
                                {
                                ant=P2f(i,j); mat=Uf(j,z);
                                a1+=P2f(i,j)*Uf(j,z);
                                }
                                P2Uf(i,z)=a1;
                                mat=a1;
                                a1=0;
                        }
                                mat+=ant;

} //FIM DA SUBROTINA

##########################################################################################

/* SUBROTINA PARA A ESCOLHA DO USUÁRIO */

char menu(void)

{ //INÍCIO DA SUBROTINA

char ch;

do
  {
clrscr();
printf("ESCOLHA UMA DAS OPCOES SEGUINTES ABAIXO\n");
printf("PARA REALIZAR A OPERACAO DESEJADA\n");
printf("Tecle (I) se desejar algumas instrucoes iniciais\n");
printf("Tecle (E) se desejar entrar com os dados 'via' teclado\n");
printf("Tecle (M) se desejar ver os dados que voce digitou\n");
printf("Tecle (G) para gravar os dados via teclado\n");
printf("Tecle (L) para ler um arquivo\n");
printf("LEIA O ARQUIVO ANTES DE RECONCILIAR\n");
printf("Tecle (R) para reconciliar e crie um arquivo de saida\n");
printf("Tecle (S) quando quiser sair do programa\n");
printf("\nDigite a sua opcao:");
ch=toupper(getche());
}

while(ch!='I' && ch!='E' && ch!='M' && ch!='G' && ch!='S '&& ch!='L'
                        &&  ch!='R');
printf("\n");
return ch;

} //FIM DA SUBROTINA

/* SUBROTINA PARA ENTRADA DE DADOS VIA TECLADO */

void entrar()

{ //INICIO DA SUBROTINA

int i,j,nviv,nvie;

printf("Digite o numero(inteiro) de correntes globais, ni: ");
scanf("%d%*c",&ni);

printf("Digite o numero(inteiro) de nos do processo, mi: ");
scanf("%d%*c",&mi);

printf("Digite o numero(inteiro) de componentes do processo, c: ");
scanf("%d%*c",&c);
```

```
printf("Digite o numero(inteiro) de reacoes quimicas no processo, rq: ");
scanf("%d%*c",&rq);

printf(" Digite o numero(inteiro) de restricoes no processo, nr: ");
scanf("%d%*c",&nr);

printf("Digite o numero(inteiro) de vazoes medidas no processo, q: ");
scanf("%d%*c",&q);

printf("Digite o numero(inteiro) de vazoes desconhecidas no processo");
printf("em que se conhece APENAS as concentracoes,vd: ");
scanf("%d%c",&vd);

printf("Entre com o numero (inteiro) de zeros em B2.zero:");
scanf("%d%c", &zero);

nviv=ni*(c+1)+rq;
nvie=mi*(c+1)+nr*(c+1)+ni;

clrscr();

printf("nviv=%d\n",nviv);
printf("Entre com informacoes das correntes\n");
printf("As correntes devem estar em ordem numerica\n");
printf("Digite dois valores, separados por um espaco, para cada corrente\n");
printf("No primeiro valor, coloque o numero do no que a corrente entra\n");
printf("No Segundo valor, coloque o numero do no que a corrente sai\n");
printf("OS VALORES DEVEM SER INTEIROS\n");

for(i=0;i<ni;++i)
        {
printf("Corrente %d:",i+1);
scanf("%d%*c%d%*c",&M[i][0],&M[i][1]);
        }

clrscr();

printf("Entre com as variancias das correntes totais:\n");
printf("Digite 0(zero) caso a variancia da corrente nao tenha sido medida\n");
printf("Digite o valor 'real' se for medida\n");
printf("Um valor para cada uma das %d correntes\n",ni);

if(q)
{
for(i=0;i<ni;i++)
        {
printf("Entre com a variancia da corrente %d: ",i+1);
scanf("%f",&varct[i]);
        }
}

/*
if(tst)
{

if(vd)
{
printf("Ente com a variancia da corrente ref a B2 %d: ",i+j);
scanf("%f",&varct2[i]);
}

}
printf("\n");

clrscr();
*/

printf("Entre com as fracoes parciais de cada corrente\n");
printf("Separe cada valor por um espaco\n");
printf("Digite o valor 'real' quando a fracao for medida\n");
printf("Digite 0(zero) se o componente nao fizer parte da corrente\n");
printf("Digite 2(dois) se fizer parte mas nao for medida\n");
printf("O seu exemplo deve conter %d valores\n",c+1);
printf("No ultimo valor de cada corrente, digite\n");
printf("0 (zero) se a corrente contiver apenas um elemento\n");
printf("1 (um) se a corrente tiver mais que um elemento e for medida\n");
```

```
printf("2 (dois) se a corrente tiver mais que um elemento e nao for medida\n");
printf("CUIDADO, TODOS OS VALORES DEVEM SER 'REAIS'\n");

for(i=0;i<ni;i++)
        {
printf("Corrente %d:",i+1);
                for(j=0;j<c+1;j++)scanf("%f",&MFM[i][j]);
        }

clrscr();

printf("Entre com informacoes sobre reacoes\n");
printf("Este processo contem %d reacao(oes) quimicas\n",rq);
printf(" Portanto, seu exemplo  deve conter %d elementos\n", c+2);
printf("Digite no primeiro elemento de cada reacao o numero do no\n");
printf("em que ocorre a reacao\n");
printf("Digite os coeficientes estequiometricos dos componentes\n");
printf("Obedeca a seguinte convencao\n");
printf("'+' se se tratar de produtos\n");
printf("'-' se se tratar de reagentes\n");
printf("'0 (zero)' se for inerte o estiver fora daquela reacao\n");
printf("CUIDADO, TODOS OS ELEMENTOS DEVEM SER 'REAIS'\n");

for(i=0;i<rq;i++)
        {
printf("Reacao %d\n:",i+1);
                for(j=0;j<c+2;j++)
                                                {
printf("Elemento %d:",j+1);
scanf("%f",&TE[i][j]);
printf("\n");
                                                }
        }

clrscr();

printf("Entre com informacoes sobre restricoes nas correntes\n");
printf("Digite 0(zero) quando a corrente nao fizer parte da restricao\n");
printf("Digite um valor 'reais' correspondente para as demais correntes\n");

for(i=0;i<nr;i++)
        {
printf("Restricao %d: \n",i+1);
                for(j=0;j<ni;j++)scanf("%f",&TR[i][j]);
printf("\n");
        }

clrscr();

printf("Entre com as vazoes medidas:\n");
printf("Este processo tem %d vazoes medidas\n",q);
printf("Portanto, voce deve digitar %d valores\n",q);
printf("Digite os valores medidos em cada corrente\n");
printf("Siga em ordem numerica para cada corrente\n");
printf("Ao fim de cada corrente digite o valor total da corrente\n");

for(j=0,i=0;i<q;i++)
        {
printf("vazao %d= ",i+1);
scanf("%f",&X[i][j]);
printf("\n");
        }

if(vd)
{
clrscr();

printf("Entre com as concentracoes medidas (referentes a B2) ");
for(j=0,i=0;i<vd;i++)
        {
printf("Concentracao %d=",i+1);
scanf("%f",&DD[i][j]);
printf("\n");
        }

clrscr();
```

```
printf("Entre com as informacoes das variaveis\n");
printf("Voce deve digitar %d valores\n",nviv);
printf("Digite 0(zero) se o componente nao passar pela corrente\n");
printf("Digite 1(um) se o componente passar e tiver o fluxo total \n");
printf("da corrente medido (OU PARTICULAR) e as concentracoes medidas\n");
printf("Digite 2(dois) se APENAS as concentracoes forem medidas\n");
printf("Digite 3(tres) se a taxa de fluxo total for medida ou nao,\n");
printf("mas nao tiver NENHUMA concentracao medida\n");
printf("Digite nos elementos finais de cada corrente, ou seja, apos\n");
printf("cada grupo de %d elementos\n",c);
printf("Digite 0(zero) se a corrente contiver uma unica especie\n");
printf("Digite 1(um) se tiver pelo menos uma vazao e as concentracoes\n");
printf("Digite 2(dois) se SO conhecer as concentracoes\n");
printf("Digite 3(tres) se nada for medido\n");
printf("Digite 3(tres) nos ultimos %d elementos\n",rq);

for(i=0;i<nviv;i++)
        {
printf("viv[%d]=",i+1);
scanf("%d",&viv[i]);printf("\n");
        }
}

if(!vd)
{
clrscr();

printf("Entre com informacoes das variaveis:\n");
printf("Voce deve digitar aqui %d valores, segundo a convencao\n",nviv);
printf("Siga cada corrente em ordem numerica\n");
printf("Digite 0(zero) se o componente nao passar por esta corrente\n");
printf("Digite 1(um) se o componente passar e tiver a vazao medida\n");
printf("Digite 3(tres) se o componente passar pela corrente e nao\n");
printf("tiver a vazao medida\n");
printf("nviv=%d\n",nviv);
printf("TODOS OS VALORES AQUI DEVEM SER INTEIROS\n");

for(i=0;i<nviv;i++)
        {
printf("viv[%d]=",i+1);
scanf("%d",&viv[i]);
printf("\n");
        }
}

clrscr();

printf("Entre com informacoes dos equipamentos:\n");
printf("Voce deve digitar %d grupos de valores\n",(mi+nr)*(c+1)+ni);
printf("Siga a seguinte convencao\n");
printf("Os %d primeiros grupos de valores correspondem aos nos\n",mi);
printf("Cada grupo contem % elementos\n",c+1);
printf("O ultimo elemento de cada grupo se refere a corrente total\n");
printf("A partir do grupo numero %d estao as informacoes\n",ni+1);
printf("correspondentes as restricoes\n");
printf("Este processo contem %d restricoes de corrente\n",nr);
printf("Portanto voce deve digitar mais %d grupo(s) de valor(es)\n",nr);
printf("Para os grupos referentes aos nos e as restricoes, voce\n");
printf("deve seguir a seguinte convencao\n");
printf("Digite 0(zero) quando a especie quimica nao fizer contato\n");
printf("com aquele no ou nao participar da restricao\n");
printf("Digite 0(zero) ao ultimo elemento destes grupos\n");
printf("Digite 1(um) quando o componente entrar ou sair do no em questao\n");
printf("ou quando fizer parte da restricao\n");
printf("No ultimo grupo de %d elementos devem ser digitadas as\n",ni);
printf("informacoes dos balancos individuais\n");
printf("Digite 0(zero) se a corrente contiver apenas um elemento\n");
printf("Digite 1(um) se a corrente contiver mais que um elemento\n");
printf("nvie=%d\n",nvie);
printf("TODOS OS VALORES DEVEM SER INTEIROS\n");

for(i=0;i<nvie;i++)
        {
printf("vie[%d]=",i);
scanf("%d",&vie[i]);
printf("\n");
```

```
            }
} //FIM DA SUBROTINA

##############################################################################################

/* SUBROTINA PARA MOSTRAR NA TELA OS DADOS ALIMENTADOS */

void mostrar(void)

{ //INÍCIO DA SUBROTINA

int i,j;

printf("\nNumero de correntes globais:%d\n",ni);

printf("Numero de nos:%d\n",mi);

printf("Numero de componentes:%d\n",c);

printf("Numero de reacoes quimicas:%d\n",rq);

printf("Numero de restricoes:%d\n",nr);

printf("Numero de vazoes medidas:%d\n",q);

printf("Numero de concentracoes para B2:%d\n",vd);

printf("Numero de zeros em B2:%d\n",zero);

printf("Decisao sobre a porcentagem:%d\n",porc);

tt=getche();

printf("\nInformacoes das correntes:\n");

for(i=0;i<ni;++i)
printf("Corrente %d:%d,%d\n",i+1,M[i][0],M[i][1]);
printf("\n");

printf("\nVariancia das correntes globais:\n");

if(q)
{
for(i=0;i<ni;i++)
printf("%8.2f\t",varct[i]);
}

/*  if(tst)*/

printf("\nVariancia das concentracoes:\n");

if(vd)
  {
            for(i=0;i<ni;i++)
printf("%8.2f\t",varct2[i]);
  }

printf("\n");

printf("Fracoes parciais\n");
for(i=0;i<ni;i++)
            {
printf("Corrente %d:",i+1);
for(j=0;j<c+1;j++)printf("%8.7f",MFM[i][j]);
printf("\n");
tt=getche();
            }

printf("\nMatriz estequiometria:\n");
for(i=0;i<rq;i++)
            {
             for(j=0;j<c+2;j++)
printf("%6.2f",TE[i][j]);
printf("\n");
            }
```

```
printf("\nMatriz das restricoes:\n");
for(i=0;i<nr;i++)
                {
                 for(j=0;j<ni;j++)
printf("%6.2f",TR[i][j]);
printf("\n");
                }

printf("\nVazoes Medidas:\n");
for(j=0,i=0;i<q;i++) printf("%8.4f ",X[i][j]);
printf("\n");
tt=getche();

if(vd)
{
printf("\nConcentracoes medidas para B2:\n");
for(j=0,i=0;i<vd+zero;i++)
printf("%8.4f",DD[i][j]);
printf("\n");
}

printf("\nInformacoes das variaveis:\n");
for(i=0;i<(ni*(c+1))+rq;i++) printf("%d ",viv[i]);

printf("\nInformacoes dos equipamentos:\n");
for(i=0;i<mi*(c+1)+nr*(c+1)+ni;i++) printf("%d ",vie[i]);
tt=getche();

} //FIM DA SUBROTINA

##############################################################################

/* SUBROTINA PARA INSTRUÇÃO DE USO DO PROGRAMA */

void instrucao(void)

{ //INÍCIO DA SUBROTINA

clrscr();

printf("\n\n\n\n\n");
printf("PROGRAMA PARA RECONCILIZACAO E COAPTACAO DE DADOS DE PROCESSO\n");
printf("\nElaborado por:ANTONIO CESAR TEIXEIRA\n");
printf("\nOrientador:JOAO ALEXANDRE F. R. PEREIRA\n");
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\Tecle qualquer letra\n");
tt=getche();

clrscr();

printf("ESTE PROGRAMA FAZ A RECONCILIACAO DE DADOS DE UM PROCESSO\n");
printf("OFERECENDO TAMBEM ESTIMATIVAS PARA OS VALORES OBSERVAVEIS\n");
printf("NAO MEDIDOS - COAPTACAO\n");
printf("\n");
printf("Voce pode utilizar o programa atraves de dois procedimentos\n");
printf("distintos\n");
printf("1.Entrando com os dados via teclado\n");
printf("2.Entrando com os dados via arquivo de dados\n");
printf("\n\n");
printf("O procedimento indicado pelo numero 1, permite que se veja \n");
printf("passo a passo os dados que vao sendo fornecidos ao programa\n");
printf("Contudo, em caso de erro no valor fornecido ao programa\n");
printf("as mudancas se tornam muito mais cansativas uma\n");
printf("vez que e necessario comecar tudo novamente.\n");
printf("\n\n");
printf("O procedimento indicado pelo numero 2 por outro lado permite\n");
printf("que se facam tantas modificacoes quantas forem necessarias\n");
printf("no arquivo de dados previamente contruido, o que facilita em\n");
printf("caso de simulacao. Perde entretanto em visualizacao para o\n");
printf("procedimento 1 por nao permitir visualizar no arquivo o que\n");
printf("significa cada valor.\n");
printf("Digite qualquer letra\n");
tt=getche();

clrscr();

printf("Como sugestao para melhor utilizacao dos recursos do programa\n");
```

```
printf("e conveniente que numa primeira entrada de dados se utilize o\n");
printf("procedimento 1-entrada de dados via teclado(E)\n");
printf("\n\n");
printf("Uma vez que todos os dados tenham sido fornecidos ao programa\n");
printf("voce pode visualiza-los, se desejar, apertando a tecla(M).\n");
printf("Apos a entrada de dados voce deve grava-los em um arquivo\n");
printf("reservado para tal finalidade.\n");
printf("A partir deste arquivo, todas as modificacoes necessarias\n");
printf("podem ser efetuadas.\n");
printf("\n\n\n\n\n\n\n\n\n\n\n");
printf("Digite qualquer letra\n");
tt=getche();

clrscr();

printf("Todas as instrucoes para a utilizacao do procedimento 1\n");
printf("se encontram detalhadas dentro do proprio programa\n");
printf("ou seja, e auto-explicativo.\n");
printf("\n\n\n");
printf("Para utilizacao direta do procedimento 2, voce precisara\n");
printf("um modelo de arquivo anteriormente gravado, ou entao saber\n");
printf("exatamente onde cada valor deve estar detalhado.\n");
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
printf("Digite qualquer letra");
tt=getche();

clrscr();

printf("\n\n\n\n\n\n\n\n\n\n");
printf("OBEDECA A TODAS AS CONVENCOES PREEVISTAS NO PROGRAMA");
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
printf("Boa sorte!\n");
printf("Digite qualquer letra");
tt=getche();

} //FIM DA SUBROTINA

##########################################################################################

/* SUBROTINA PARA GRAVAR EM ARQUIVO OS DADOS VIA TECLADO */
void gravar(void)

{ //INÍCIO DA SUBROTINA
FILE *fp;
int i,j;
char s[20];

printf("Nome do arquivo:");
gets(s);

if((fp=fopen(s,"w"))==NULL)
        {
printf("Nao posso abrir arquivo\n");
return;

        }

printf("%d",fprintf(fp," %d %d %d %d %d %d %d\n",ni,mi,
c,rq,nr,q,vd,zero));

for(i=0;i<ni;i++)
printf("%d",fprintf(fp,"%d %d ",M[i][0],M[i][1]));
printf("%d",fprintf(fp,"\n"));

printf("%d",fprintf(fp,"\n"));

if(q)
{
for(i=0;i<ni;i++)
printf("%d",fprintf(fp,"%f\t",varet[i]));
}
if(vd)
{
for(i=0;i<ni;i++)
printf("%d",fprintf(fp,"%f\t",varet2[i]));
}
```

```
printf("%d",fprintf(fp,"\n"));

for(i=0;i<ni;i++)
                {
                 for(j=0;j<c+1;j++)
printf("%d",fprintf(fp,"%f\t",MFM[i][j]));
printf("%d",fprintf(fp,"\n"));
                }

for(i=0;i<rq;i++)
                {
                                for(j=0;j<c+2;j++)
printf("%d",fprintf(fp,"%6.2f",TE[i][j]));
printf("%d",fprintf(fp,"\n"));
                }

for(i=0;i<nr;i++)
                {
                 for(j=0;j<ni;j++)
printf("%d",fprintf(fp,"%6.2f",TR[i][j]));
printf("%d",fprintf(fp,"\n"));
                }

for(j=0,i=0;i<q;i++)
printf("%d",fprintf(fp,"%6.2f ",X[i][j]));
printf("%d",fprintf(fp,"\n"));

if(vd)
{
for(j=0,i=0;i<vd+zero;i++)
printf("%d",fprintf(fp,"%6.2f",DD[i][j]));
printf("%d",fprintf(fp,"\n"));
}

for(i=0;i<(ni*(c+1))+rq;i++)
printf("%d",fprintf(fp,"%d ",viv[i]));

printf("%d",fprintf(fp,"\n"));

for(i=0;i<(mi*(c+1))+(nr*(c+1))+ni;i++)
printf("%d",fprintf(fp,"%d ",vie[i]));

printf("%d",fprintf(fp,"\n"));

// fclose(fp);

} //FIM DA SUBROTINA

##########################################################################################

/* LER UM ARQUIVO DE DADOS */

void ler(void)

{ //INÍCIO DA SUBROTINA

FILE *fp;
int i,j; char s[20];

printf("\nNome do arquivo:" ):
gets(s);

if((fp=fopen(s,"r"))==NULL)
{
printf("Nao posso abrir arquivo\n");
exit(1);
}

fscanf(fp,"%d%*c%d%*c%d%*c%d%*c%d%*c%d%*c%d%*c%d%*c%d%*c%d%*c",&ni,&mi,&c,&rq,&nr,&q,&vd,&zero,&porc);

for(i=0;i<ni;i++)
fscanf(fp,"%d%*c%d%*c",&M[i][0],&M[i][1]);
if(q)
{
for(i=0;i<ni;i++)  fscanf(fp,"%f%*c",&varct[i]);
}
```

```
if(vd)
{
 for(i=0;i<ni;i++) fscanf(fp,"%f%*c",&varct2[i]);
}

for(i=0;i<ni;i++)
        for(j=0;j<c+1;j++)fscanf(fp,"%f%*c",&MFM[i][j]);

for(i=0;i<rq;i++)
        for(j=0;j<c+2;j++)fscanf(fp,"%f%*c",&TE[i][j]);

for(i=0;i<nr;i++)
        for(j=0;j<ni;j++)fscanf(fp,"%f%*c",&TR[i][j]);

for(j=0,i=0;i<q;i++)fscanf(fp,"%f%*c",&X[i][j]);

if(vd)
{
for(j=0,i=0;i<vd+zero;i++)fscanf(fp,"%f%*c",&DD[i][j]);
}

for(i=0;i<(ni*(c+1))+rq;i++)fscanf(fp,"%d%*c",&viv[i]);

for(i=0;i<(mi*(c+1))+(nr*(c+1))+ni;i++)
        fscanf(fp,"%d%*c",&vie[i]);

} // FIM DA SUBROTINA

###############################################################################################

/* SUBROTINA PARA NOMINAR AS VARIAVEIS MEDIDAS OU NAO */

void vnom(int c,int rq,int ni,int nic1,int *viv,char *cm,char *cnm,char *dm)

{ //INÍCIO DA SUBROTINA

char *compo;
char *numcor1;
char *numcor2;
char *alc;
int c1=c+1;
register int i,j,ki,kj,kz;

compo=TMC(MCOM+1);
numcor1=TMC(MCOR);
numcor2=TMC(MCOR);
alc=TMC(MCOR*(MCOM+1)*3);

for(i=0;i<c;i++) compo[i]='A'+i;

for(i=c;i<c1;i++) compo[i]='T';

for(i=0;i<9;i++)
        {
         numcor1[i]='-';
         numcor2[i]='1'+i;
        }

for(j=0,i=9;i<ni;i++,j++)
        {
         numcor1[i]='1';
         numcor2[i]='0'+j;
        }

if(ni==20)
        {
        numcor1[ni-1]= '2';
        numcor2[ni-1]='0';
        }

/* FORMA O VETOR DE TODAS AS CORRENTES */

for(i=0;i<ni;i++)
        for(j=0;j<c1;j++)
                {
                        ALC(j+i*c1,0)=compo[j];
```

```
                                ALC(j+i*c1,1)=numcor1[i];
                                ALC(j+i*c1,2)=numcor2[i];
                        }

for(i=nic1;i<nic1+rq;i++)
        {
        ALC(i,0)='R';
        ALC(i,1)='1'+(i-nic1);
        ALC(i,2)=' ';
        }

/* FORMA O VETOR DAS CORRENTES MEDIDAS E NÃO-MEDIDAS */

for(ki=0,kj=0,kz=0,i=0;i<nic1+rq;i++)
        {
        if(viv[i]==1)
                {
                        for(j=0;j<3;j++)
                                CM(ki,j)=ALC(i,j);
                                ki++;
                }
        else
                        if(viv[i]==2)
                                {
                                for(j=0;j<3;j++)
                                        DM(kz,j)=ALC(i,j);
                                        kz++;
                                }
        else
                        if(viv[i]==3)
                                {
                                for(j=0;j<3;j++)
                                        CNM(kj,j)=ALC(i,j);
                                        kj++;
                                }
        }

} //FIM DA SUBROTINA

##########################################################################################

/* SUBROTINA PARA O CALCULO DA VARIANCIA */

void variancia(int ni,int c,int n1,int tst,float *varct,float *mfm,float *pps)

{ //INÍCIO DA SUBROTINA

int nctm ,c1=c+1;
float *mb,*st,*ms;
float *varm;
float a1,mat,ant;
register int i,j,t,z,k;

/* CONTADOR DO NUMERO TOTAL DE CORRENTES MEDIDAS */

for(nctm=0,i=0;i<ni;i++)
        {
        ant=varct[i];
        if(varct[i] !=0)
                nctm++;
        }


/* OBTER A MATRIZ MB */

mb=TM(n1*nctm);

for(i=0;i<n1;i++)
        for(j=0;j<nctm;j++)
                MB(i,j)=0 ;

if(!tst)
{
for(t=0,k=0,j=0;j<ni;j++)
        if(varct[j] != 0)
                {
```

```
                    for(i=0;i<c1;i++)
                                if(MFM(j,i) != 0 && MFM(j,i) !=2)
                                        {
                                        mat=MFM(j,i);
                                        MB(k,t)=MFM(j,i);
                                        k++;
                                        }
                                        t++;
                    }
}

if(tst)
{
for(t=0,k=0,z=0,j=0;j<ni;j++)
        if(varct[j]!=0)
                {
                for(i=0;i<c1;i++)
                                if(MFM(z,i) !=0 && MFM(z,i) !=2)
                                        {
                                        ant=MFM(z,i);
                                        MB(k,t)=MFM(z,i);
                                        mat=MB(k,t);
                                        k++;
                                        }
                                        t++;
                                        z++;
                }
}

for(i=0;i<n1;i++)
        for(j=0;j<nctm;j++)
                ant=MB(i,j);

/* OBTER O VETOR VARIÂNCIA MODIFICADO */

varm=TM(nctm); *varm=0;

for(t=0,i=0;i<ni;i++)
        if(varct[i] != 0)
                {
                varm[t]=varct[i];
                t++;
                }

/* OBTER A MATRIZ VARIÂNCIA DAS CORRENTES TOTAIS */

st=TM(nctm*nctm);

for(i=0;i<nctm;i++)
        for(j=0;j<nctm;j++)
                ST(i,j)=(i==j ? varm[i] : 0);

for(i=0;i<nctm;i++)
        for(j=0;j<nctm;j++)
                ant=ST(i,j);

ms=TM(n1*nctm);

/* MS= MB * ST */

for(a1=0,i=0;i<n1;i++)
        for(j=0;j<nctm;j++)
                {
                        for(t=0;t<nctm;t++)
                                {
                                ant=MB(i,t); mat=ST(t,j);
                                a1 += MB(i,t)*ST(t,j);
                                }
                                MS(i,j)=a1;
                                mat=MS(i,j);
                                a1=0;
                }

free(st);
```

```
/* S = MS * MBT */

for(a1=0,i=0;i<n1;i++)
        for(t=0;t<n1;t++)
                {
                        for(j=0;j<nctm;j++)
                                {
                                ant=MS(i,j);  mat=MB(t,j);
                                a1+=MS(i,j)*MB(t,j);
                                }
                                S(i,t)=a1;
                                mat=S(i,t);
                                a1=0;
                }

for(i=0;i<n1;i++)
        for(j=0;j<n1;j++)
                mat=S(i,j);

free(mb);

for(i=0;i<n1;i++)
        for(j=0;j<n1;j++)
                {
                        if(i!=j)
                                S(i,j)= S(i,j)/2;
                                mat=S(i,j);
                }
mat+=ant;

} //FIM DA SUBROTINA

##########################################################################################################################
FIM DAS SUBROTINAS EXCLUSIVAS AO PROGRAMA 1.
##########################################################################################################################
```

```
##########################################################################################################
PROGRAMA 2- DETECÇÃO DE ERROS GROSSEIROS- RETIF.C
##########################################################################################################

**********************************************************************************************************
ESTE PROGRAMA É UM GERENCIADOR DE SUBROTINAS E ARQUIVOS PARA DETECÇÃO DE ERROS GROSSEIROS
EM VARIAVEIS DE PROCESSO.
**********************************************************************************************************
**************************************************RETIF.C*************************************************

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<alloc.h>
#include<ctype.h>
#include<string.h>
#include<conio.h>
#include ""..."sub\subs.h"


#define Zaj(i,j) (*(zaj+n1*(i)+j))
#define Zajr(i,j) (*(zajr+n1*(i)+j))
#define Zcr(i,j) (*(zcr+30*(i)+j))
#define Zcr_a(i,j) (*(zcr_a+1*(i)+j))
#define Zcr_d(i,j) (*(zcr_d+1*(i)+j))
#define Zcr_e(i,j) (*(zcr_e+1*(i)+j))
#define CHI2_tab(i,j) (*(chit+30*(i)+j))
#define Zdelta(i,j) (*(zdelta+n2*(i)+j))
#define Zdeltar(i,j) (*(zdeltar+n2*(i)+j))
#define HZTe(i,j) (*(hzte+p*(i)+j))
#define Zerr(i,j) (*(zerr+p*(i)+j))
#define X(i,j) (*(x+1*(i)+j))
#define DD(i,j) (*(dd+1*(i)+j))
#define DE(i,j) (*(de+c1*(i)+j))
#define B1(i,j) (*(b1+n1*(i)+j))
#define B2(i,j) (*(b2+n2*(i)+j))
#define B2F(i,j) (*(b2f+nulo*(i)+j))
#define B2DE(i,j) (*(bde+blo*(i)+j))
#define S1B1T(i,j) (*(sbt+m*(i)+j))
#define ZHT(i,j) (*(zh+p*(i)+j))
#define Zainv(i,j) (*(zai+pd*(i)+j))
#define YHT(i,z) (*(yh+m*(i)+z))
#define YT(i,j) (*(y+m*(i)+j))
#define No(i,j) (*(no+n2*(i)+j))
#define Delta(i,j) (*(dlt+1*(i)+j))
#define HH(i,j) (*(hh+p*(i)+j))
#define Zb(i,j) (*(zb+1*(i)+j))
#define S(i,j) (*(pps+n1*(i)+j))
#define Sd(i,j) (*(ppd+vd*(i)+j))
#define AJ2(i,j) (*(aj2+1*(i)+j))
#define CM(i,j) (*(cm+3*(i)+j))
#define P1f(i,j) (*(p1f+rf*(i)+j))
#define Uf(i,j) (*(uf+rf*(i)+j))
#define P2Uf(i,j) (*(puf+rf*(i)+j))
#define P2f(i,j) (*(p2f+rf*(i)+j))
#define DT(i,j) (*(dt+c1*(i)+j))
#define D(i,j) (*(d+blo*(i)+j))
#define DL(i,j) (*(dl+blo*(i)+j))
#define HN(i,j) (*(hnd+p*(i)+j))
#define DAF(i,j) (*(daf+blo*(i)+j))
#define D1(i,j) (*(d1+blo*(i)+j))
#define D2(i,j) (*(d2+rd*(i)+j))
#define D3(i,j) (*(d3+blo*(i)+j))
#define U2(i,j) (*(u2+blo*(i)+j))
#define D2U(i,j) (*(du+rd*(i)+j))
#define ZT(i,j)  (*(zt+rd*(i)+j))
#define ZTI(i,j) (*(zti+rd*(i)+j))
#define UD(i,j) (*(ud+pd*(i)+j))
#define ZHTg(i,j) (*(zhg+p*(i)+j))
#define CHI2(i,j) (*(chi2+1*(i)+j))
#define Q1jj(i,j) (*(q1+n1*(i)+j))
#define Err(i,j) (*(err+p*(i)+j))
#define Q2II(i,j) (*(q2+n2*(i)+j))
#define HZT(i,j) (*(hzt+pd*(i)+j))
#define YY(i,j) (*(yy+m*(i)+j))
#define ZHTIIH(i,j) (*(zhth+p*(i)+j))
```

```
#define T(i,j) (*(ttr+pd*(i)+j))
#define Tinv(i,j) (*(ti+pd*(i)+j))


#define TM(size)  (float *)malloc(size*sizeof(float))
#define TMI(size) (int *)malloc(size*sizeof(int))
#define TMC(size) (char *)malloc(size*sizeof(char))

#define MNO  10
#define MCOR 20
#define MCOM 10
#define MREA  5
#define MRES  3
#define MCOM 10

char tt;char aster;int acomp;int blo;int nulo;int td;int rd;nt s;int c1;int bool;
int n1;int n2;int ni;int c;int rq;int q;int m;int vd;int pd;int p;int p11;
int viv[223];
int vivd[223];
float XI[30][1];
float B1I[30][30];
float B2I[30][20];
float YHTI[30][30];
float SI[30][30];
float ZHTI[30][30];
float SdI[30][30];
float ZbI[30][1];
float S1B1TI[30][30];
float NoII[30][30];
float HHI[30][30];
float ZainvI[30][30];
float AJ2I[30][1];
float DeltaI[30][1];
float B2FI[30][30];
float DDI[30][1];

char menu(void);
int TESTE_G(int pd,float *chi2,float *chit);
void recon_retif(void);
void vbacks(int rf,int rs,int *irf,float *p1f,float *uf);
int lu(int rf,int *irf,float *p1f);
void ler(void);

main()

{

 char ch;

 for(;;){
              ch=menu();
               switch(ch)
                        {
              case 'L' : ler();
                          break;
              case 'R' : recon_retif();
                          break;
              case 'S' : exit(0);

                        }
               }
}

void recon_retif(void)

{
/* REFERENTE ÀS SUBROTINAS INCLUDE E RETIFIC */

int ntes,tg,t1,t2,t3;
float *zaj,*zajr,*zcr,*zcr_a,*zcr_e,*zcr_d,*chit,*zdelta,*zdeltar,*hzte,*zerr;
float *yy,*q1;
float *q2,*sdn,*sdyy,*sdnb,*sdyb,*sdn2;
float *hz,*hzt,*err,*hztz;
float *chi,*chi2;
int *mza,*mze,*mzd,*mvs,*mvs_a,*mvs_d,nms;
float aux1,aux2,aux3,aux4,aux5,aux,mat,ant;
```

```
float *bde,*dt,*de,*hnd,*daf,*d1,*d2,*d3,*d,*dl,*u2,*du,*zt,*zti,*ud;
int *ic2,*ip2,*ird;

/* REFERENTE A RECON */

float *x,*xd;float *b1;float *b2;float *b2f;float *zh;float *zhg;float *zai;float *yh;float *y;float *no;float *dlt;
float *hh;float *zb;float *ppd;float *pps;float *sbt;float *aj2;float *dd;float *zhth;float *ttr;float *ti;int *irt;char  *cm;double fabs();

float vZcr[30]={1.96,2.235,2.39,2.49,2.57,2.63,2.68,2.73,2.76,2.79,2.825,
                2.855,2.88,2.905,2.93,2.94,2.959,2.977,2.995,3.021,3.028,
                3.043,3.048,3.053,3.077,3.077,3.101,3.105,3.109,3.132};

float vCHI_tab[30]={3.81,5.991,7.815,9.488,11.070,12.592,14.067,15.507,16.919,
                18.317,19.675,21.026,22.362,23.685,24.996,26.296,27.587,
                28.869,30.144,31.410,32.672,33.924,35.172,36.515,37.652,
                38.885,40.113,41.337,42.557,43.773};

register int i,ij,t,z,k,ki,k2,j,w;

FILE *fp;
char pa[20];

printf("Nome do Arquivo dos Resultados:");
gets(pa);

if((fp=fopen(pa,"w"))==NULL)
{
printf("Nao posso abrir arquivo\n");
return;
}

printf("%d", fprintf(fp,"\nRESULTADOS DA RETIFICACAO DE DADOS\n\n"));

if(!vd)
{
printf("%d",fprintf(fp,"SISTEMA LINEAR\n"));
}

/*
if(vd)
{
printf("%d",fprintf(fp,"SISTEMA NAO-LINEAR\n"));
}
*/

zaj=NULL; *zaj=0;zajr=NULL; *zajr=0;zcr=NULL; *zcr=0;zcr_a=NULL; *zcr_a=0;
zcr_d=NULL; *zcr_d=0;zcr_e=NULL; *zcr_e=0;
chit=NULL; *chit=0;zdelta=NULL; *zdelta=0;zdeltar=NULL; *zdeltar=0;hzte=NULL; *hzte=0;zerr=NULL; *zerr=0;
x=NULL; *x=0;dd=NULL; *dd=0;de=NULL; *de=0;b1=NULL; *b1=0;b2=NULL; *b2=0;b2f=NULL; *b2f=0;bde=NULL; *bde=0;
sbt=NULL; *sbt=0;zh=NULL; *zh=0;zai=NULL; *zai=0;yh=NULL; *yh=0;y=NULL; *y=0;no=NULL; *no=0;
dlt=NULL; *dlt=0;hh=NULL; *hh=0;zb=NULL; *zb=0;pps=NULL; *pps=0;ppd=NULL; *ppd=0;
aj2=NULL; *aj2=0;dt=NULL; *dt=0;d=NULL; *d=0;dl=NULL; *dl=0;hnd=NULL; *hnd=0;daf=NULL;  *daf=0;d1=NULL; *d1=0;
d2=NULL; *d2=0;d3=NULL; *d3=0;u2=NULL; *u2=0;du=NULL; *du=0;zt=NULL; *zt=0;ud=NULL; *ud=0;zhg=NULL; *zhg=0;
zhth=NULL; *zhth=0;yy=NULL; *yy=0;q1=NULL; *q1=0;q2=NULL; *q2=0;sdn=NULL; *sdn=0;sdyy=NULL; *sdyy=0;
sdnb=NULL; *sdnb=0;sdyb=NULL; *sdyb=0;sdn2=NULL; *sdn2=0;hz=NULL; *hz=0;hzt=NULL; *hzt=0;err=NULL; *err=0;
hztz=NULL; *hztz=0;chi=NULL; *chi=0;chi2=NULL; *chi2=0;xd=NULL; *xd=0;ttr=NULL; *ttr=0;ti=NULL; *ti=0;

if(n2) //SÓ PARA NÃO-LINEARES

printf("\n\n\n\n\nDESEJA FAZER ACOMPANHAMENTO DAS VARIAVEIS DE ENTRADA\n");
printf("E ALGUMAS DE SAIDA(valores intermediarios)? Tecle (1)(SIM) ou (0)(NAO)\n");
printf("SUA OPCAO=");
scanf("%d",&acomp);

zaj=NULL;

aster='*';

{
chi=TM(pd*1);
chi2=TM(1*1);
zb=TM(pd*1);
zai=TM(pd*pd);

//printf("\nVALORES dE Zb\n");
```

```
for(j=0,i=0;i<pd;i++)
                {
                 Zb(i,j)=Zbl[i][j]; //E IGUAL AO ERRO E
//printf("%12.8f",Zb(i,j));
//printf("\n");
                }
//tt=getche();

for(i=0;i<pd;i++)
                for(j=0;j<pd;j++)
                                {
                                                ant=Zainvl[i][j];
                                                Zainv(i,j)=Zainvl[i][j]; //E IGAUL AO (T=ZHT*H*Z)(inverso)
                                }

/*
clrscr();
printf("\nVALORES DE Zainv\n");

for(i=0;i<pd;i++)
{
for(j=0;j<pd;j++)
{
printf("%12.8f",Zainv(i,j));
printf("  ");
}
printf("\n");
}
tt=getche();
*/

*****************************************************************************************************************************************
ESTA SUBROTINA CALCULA O VALOR CHI_QUADRADO PARA OS DADOS

TEG(p,pd,n1,zb,zai,chi2);
*****************************************************************************************************************************************

printf("%d",fprintf(fp,"\n\nTESTE ESTATISTICO GLOBAL\n\n"));
printf("%d",fprintf(fp,"CHI2(calculado)=%10.4f\n",CHI2(0,0)));

clrscr();

printf("\nTESTE ESTATISTICO GLOBAL\n");
printf("\nCHI2(0.95)=%10.4f" "....."  "indice=%d\n",CHI2(0,0),pd-1);
mat=CHI2(0,0);

/* TRANSFORMAÇÃO DO VETOR EM MATRIZ */

chit=TM(30*1);

for(j=0,i=0;i<30;i++)
                {
                 CHI2_tab(j,i)=vCHI_tab[i];
                 mat=CHI2_tab(j,i);
                }

mat=CHI2_tab(0,pd-1);

printf("%d",fprintf(fp,"CHI2_tab(s/remocao)=%10.4f\n",CHI2_tab(0,pd-1)));
printf("\nVALOR PARA COMPARACAO\n");
printf("\nCHI2_tab(0,%d)=%10.4f\n",(pd-1),(CHI2_tab(0,(pd-1))));
tt=getchar();

/* ESTA SUBROTINA COMPARA OS VALORES CHI TABELADOS/CALCULADOS */

tg=TESTE_G(pd,chi2,chit);

if(!tg)
{
printf("\nO TESTE GLOBAL NAO ESTA DETECTANDO ERRO GROSSEIRO\n");

for(aux=0,j=0,i=0;i<pd;i++)
                aux+=Zb(i,j);

if((fabs(aux)<0.0000001))
printf("\n\n\nO ERRO DAS MEDIDAS POSSIVELMENTE ESTA' ABAIXO DA TOLERANCIA\n");
```

```
}
else
{
printf("\nEXISTE SUSPEITA DE ERRO GROSSEIRO ENTRE AS MEDIDAS\n");
}
printf("\n\n\n\n\n\n\n\n\n\nTECLE ALGO PARA FAZER O SEGUNDO TESTE\n");

tt=getchar();

zaj=TM(n1*1);b1=TM(m*n1);yh=TM(p*m);sbt=TM(n1*m);pps=TM(n1*n1);zh=TM(pd*p);aj2=TM(n1*1);
q1=TM(n1*n1);yy=TM(m*m);

for(i=0;i<m;i++)
        for(j=0;j<n1;j++)
                                {
                                B1(i,j)=B1I[i][j];
                                mat=B1(i,j);
                                }

//clrscr();
//printf("VALORES DE YHT(em RETIF)\n");

for(i=0;i<p;i++)
        {
        for(j=0;j<m;j++)
                                {
                                YHT(i,j)=YHTI[i][j];
//printf("%2.0f",YHT(i,j));
                                mat=YHT(i,j);
                                }
//printf("\n");
        }
//tt=getche();

for(i=0;i<n1;i++)
        for(j=0;j<m;j++)
                S1B1T(i,j)=S1B1TI[i][j];

for(i=0;i<n1;i++)
        for(j=0;j<n1;j++)
                S(i,j)=SI[i][j];

for(i=0;i<pd;i++)
        for(j=0;j<p;j++)
                ZHT(i,j)=ZHTI[i][j];

for(j=0,i=0;i<n1;i++)
        AJ2(i,j)=AJ2I[i][j];

for(i=0;i<n1;i++)
        for(j=0;j<n1;j++)
                Q1jj(i,j)=1;

dd=TM(n2*1);
de=TM(blo*c1);
b2f=TM(m*nulo);

for(i=0;i<n2;i++)
        {
        for(j=0;j<1;j++)
                                {
                                DD(i,j)=DDI[i][j];
                                }
        }

for(i=0;i<m;i++)
        {
        for(j=0;j<nulo;j++)
                                {
                                B2F(i,j)=B2FI[i][j];
                                }
        }
zhg=0;
zhg=NULL;
zhg=TM(pd*p);
```

```
for(i=0;i<ni*c1+rq;i++)
vivd[i]=viv[i];

*****************************************************************************************************************************
ESTA SUBROTINA CALCULA OS VALORES DA MATRIZ Z APÓS A RECONCILIAÇÃO

ZHTf(ni,p,m,rq,blo.nulo.c1,viv.dd,yh,b2f.zhg);
*****************************************************************************************************************************

/* DETERMINAÇÃO DO NOVO VALOR DE Zainv */

hh=TM(p*p);

/*
clrscr();
printf("VALORES DE HH\n");
for(i=0;i<p;i++)
        {
         for(j=0;j<p;j++)
                                {
                                 HH(i,j)=HH[i][j];
printf("%10.6f",HH(i,j));

                                }
printf("\n");
        }
tt=getche();
*/

zhth=TM(pd*p);
ttr=TM(p*pd);
ti=TM(pd*pd);
irt=TMI(pd);

clrscr();
printf("VALORES DE ZHTg(RECALCULADO)\n");

for(i=0;i<pd;i++)
        {
         for(j=0;j<p;j++)
                                {
printf("%10.6f",ZHTg(i,j));
                                 ZHT(i,j)=ZHTg(i,j);
                                }
printf("\n");
        }
tt=getche();

aux=0;
for(i=0;i<pd;i++)
        for(j=0;j<p;j++)
                {
                        for(z=0;z<p;z++)
                                {
                                ant=ZHT(i,z); mat=HH(z,j);
                                aux+=ZHT(i,z)*HH(z,j);
                                }
                                ZHTHH(i,j)=aux;
                                aux=0;
                }

clrscr();

printf("VALORES DE ZHTHH\n");

for(i=0;i<pd;i++)
        {
         for(j=0;j<p;j++)
printf("%10.6f",ZHTHH(i,j));
printf("\n");
        }
tt=getche();

clrscr();

printf("VALORES DE T\n");
```

```
aux=0;
for(i=0;i<pd;i++)
        {
        for(j=0;j<pd;j++)
                {
                for(z=0;z<p;z++)
                        {
                                ant=ZHTHH(i,z); mat=ZHT(j,z);
                                aux+=ZHTHH(i,z)*ZHT(j,z);
                        }
                                T(i,j)=aux;
printf("%10.6f",T(i,j));
                                aux=0;
                }
printf("\n");
        }
tt=getche();

for(i=0;i<pd;i++) irt[i]=i;

for(i=0;i<pd;i++)
        for(j=0;j<pd;j++)
                Tinv(i,j)=(i==j ? 1 : 0);

*****************************************************************************************************************************
lu(pd,irt,ttr);
vbacks(pd,pd,irt,ttr,ti);
*****************************************************************************************************************************

if(acomp==1)
{
clrscr();
printf("VALORES DE T_^\n");
}

for(i=0;i<pd;i++)
        {
        for(j=0;j<pd;j++)
                {
                Zainv(i,j)=Tinv(i,j);
                 if(acomp==1)
                        {
printf("T^(%d,%d)=%12.8f",i,j,Zainv(i,j));
printf("\n");
                        }
                }
        }
if(acomp==1)

tt=getche();

*****************************************************************************************************************************
ESTA SUBROTINA DETERMINA OS PARÂMETROS PARA O CÁLCULO DOS Z's_ajuste

DEN2a(p,pd,n1,m,b1,sbt,pps,yh,zai,zh,yy,q1);
*****************************************************************************************************************************

for(k=0,j=0,i=0;i<n1;i++)
        {
        if(Q1jj(k,k))
                {
                        if(AJ2(i,j))
                                {
                                ant=AJ2(i,j);  mat=Q1jj(k,k);
                                Zaj(j,i)=(AJ2(i,j)/(sqrt(fabs(Q1jj(k,k)))));
                                mat=Zaj(j,i);
                                k++;
                                }
                if(!AJ2(i,j))
                                {
                                Zaj(j,i)=0;
                                k++;
                                }
        }
                if(!Q1jj(k,k))
                                {
```

```
                        if(AJ2(i,j))
                                {
                                        Zaj(j,i)=1000.0;
                                        k++;
                                }
                        if(!AJ2(i,j))
                                {
                                        Zaj(j,i)=0;
                                        k++;
                                }
                }
}

if(acomp==1)
{
clrscr();
printf("\nVALORES DE AJ2, Zaj e sqrt(Q1jj)\n\n");

for(j=0,i=0;i<n1;i++)
                {
printf("AJ2(%d,%d)=%10.4f""  ",j,i,AJ2(j,i));
printf("sqrt(Q1jj(%d,%d)=%10.4f""  ",i,i,sqrt(fabs(Q1jj(i,i))));
printf(" Zaj(%d,%d)=%10.4f\n",j,i,Zaj(j,i));
                }
tt=getche();

}


free(b1);free(b2f);free(sbt);free(yh);free(pps);free(aj2);free(zhg);

zdelta=TM(1*n2);
ppd=TM(vd*vd);
b2=TM(m*n2);
no=TM(n2*n2);
dlt=TM(n2*1);
q2=TM(n2*n2);

for(i=0;i<vd;i++)
  for(j=0;j<vd;j++) Sd(i,j)=Sdl[i][j];

for(i=0;i<m;i++)
  for(j=0;j<n2;j++) B2(i,j)=B2l[i][j];

for(i=0;i<n2;i++)
  for(j=0;j<n2;j++) No(i,j)=Nol[i][j];

if(acomp==1)
{
clrscr();
printf("VALORES DE Delta\n");
}
for(j=0,i=0;i<n2;i++)
                {
                Delta(i,j)=Deltal[i][j];
                if(acomp==1)
                                {
printf("Delta(%d)=%12.8f",i,Delta(i,j));
printf("\n");
                                }
                }

if(acomp==1)
tt=getche();

for(i=0;i<n2;i++)
for(j=0;j<n2;j++) Q2ll(i,j)=1;

*****************************************************************************************************************
ESTA SUBROTINA DETERMINA OS PARÂMETROS PARA O CÁLCULO DE Z's_delta

DEN2delta(n2,m,ppd,no,yy,b2,q2);
*****************************************************************************************************************
```

```
if(acomp==1)
{
clrscr();
printf("\nVALORES DE Q2ll\n");

for(i=0;i<n2;i++)
for(j=0;j<n2;j++)
{
if(i==j)
printf("\nQ2ll(%d,%d)=%12.6f",i,j,Q2ll(i,j));
}
tt=getche();
}

for(j=0,k=0,i=0;i<n2;i++)
        {
        if(Q2ll(k,k)!=0 && Q2ll(k,k)>0.00001)
                {
                Zdelta(j,i)=(Delta(i,j)/(sqrt(fabs(Q2ll(k,k)))));
                k++;
                }
                else
                        {
                        Zdelta(j,i)=0;
                        k++;
                        }
        }


for(j=0,i=0;i<n2;i++)
mat=Zdelta(i,j);

/*
if(acomp==1)
{
clrscr();
printf("\nVALORES DE Zdelta\n");
for(j=0,i=0;i<n2;i++)
{
printf("\nZdelta(%d,%d)=%12.6f",j,i,Zdelta(j,i));
}
tt=getche();
}
*/

free(b2);free(no);free(ppd);free(dlt);

hzt=TM(p*pd);err=TM(p*p);hzte=TM(1*p);zcr=TM(30*1);zerr=TM(1*p);

for(i=0;i<p;i++)
for(j=0;j<pd;j++) HZT(i,j)=0;

for(i=0;i<p;i++)
for(j=0;j<p;j++) Err(i,j)=0;

clrscr();

*****************************************************************************************************************************
ESTA SUBROTINA CALCULA OS PARÂMETROS PARA O CÁLCULO DE Z's_erro

DEN2err(p,pd,zai,zb,hh,zh,hzt,err);
*****************************************************************************************************************************

/* DETERMINAÇÃO DO ERRO-cálculo de HZTe. */

aux=0;
for(i=0;i<p;i++)
        for(j=0;j<1;j++)
                {
                        for(z=0;z<pd;z++)
                                {
                                aux+=HZT(i,z)*Zb(z,j);
                                }
                                HZTe(j,i)=aux;
                                aux=0;
                }
```

```
free(hzt);free(zh);free(zai);free(hh);free(zb);*zerr=0;

clrscr();
for(i=0;i<p;i++)
        for(j=0;j<p;j++)
                if(j==i)
                        {
printf("Err(%d,%d)=%12.6f",i,j,Err(i,j));
printf("\n");
                        }
tt=getche();

for(j=0,k=0,i=0;i<p;i++)
        {
        ant=Err(k,k); mat=HZTe(j,i);
        if(fabs(Err(k,k))>0.00000001)
                {
                Zerr(j,i)=HZTe(j,i)/(sqrt(fabs(Err(k,k))));
                k++;
                }
               else  if(fabs(Err(k,k)<0.00000001))
                        {
                        Zerr(j,i)=0.0;
                        k++;
                        }

        }

if(acomp==1)
{
clrscr();
printf("\nVALORES DE Zerr\n");

for(i=0;i<p;i++)
printf("\nZerr(0,%d)=%12.6f",i,Zerr(0,i));
tt=getche();
}

free(hzte);
free(err);

/* TRANSFORMAÇÃO DO VETOR EM MATRIZ */

for(j=0,i=0;i<30;i++)
        {
        Zcr(i,j)=vZcr[i];
        mat=Zcr(i,j);
        }

mza=TMI(n1);
mzd=TMI(n2);
mze=TMI(p);

printf("%d",fprintf(fp,"\nTESTES DOS DESBALANCOS\n"));
clrscr();
printf("NUMERO DE MEDIDAS SUSPEITAS DE TEREM ERROS(nms)\n");

**********************************************************************************************************************
ESTAS SUBROTINAS DETERMINAM O NÚMERO DE MEDIDAS COM SUSPEITA DE ERRO CASO HAJA ALGUMA
MEDIDA COM ERROS GROSSEIROS

zcr_a=TM(1*1);
zcr_d=TM(1*1);
zcr_e=TM(1*1);

bool=1;
t1=nmsus(n1,mza,zaj,zcr.zcr_a,bool,fp);
mat=Zcr_a(0,0);
bool=2;
t2=nmsus(n2,mzd,zdelta,zcr,zcr_d,bool,fp);
bool=3;
t3=nmsus(p,mze,zerr,zcr,zcr_e,bool,fp);
**********************************************************************************************************************
```

```
ntes=0;
if(tg) ntes++;
if(t1) ntes++;
if(t2) ntes++;
if(t3) ntes++;

mat=ntes;

printf("%d",fprintf(fp,"\n\nVALORES DE Z - AJUSTE(1), DELTA(2), DESBALANCO(3)\n\n\n"));

printf("%d",fprintf(fp,"AJUSTE\n"));

for(i=0,j=0;j<n1;j++)
                        {
printf("%d",fprintf(fp,"%10.4f",Zaj(i,j)));
                        if(Zaj(i,j)>Zcr_a(0,0))
printf("%d",fprintf(fp,"%c",aster));
printf("%d",fprintf(fp,"\n"));
                        }
if(t1)
printf("%d",fprintf(fp,"\nEXISTE UM PROVAVEL ERRO NAS MEDIDAS DE VAZAO AJUSTADAS(*)\n"));
else
{
printf("%d",fprintf(fp,"\nNAO ESTA SENDO DETECTADO ERRO NAS MEDIDAS DE VAZAO AJUSTADAS\n"));
printf("%d",fprintf(fp,"OS VALORES DE Z ESTAO ABAIXO DO VALOR CRITICO\n"));
printf("%d",fprintf(fp,"\n\n"));
 }
 printf("%d",fprintf(fp,"DELTA\n"));
for(i=0,j=0;j<n2;j++)
                        {
printf("%d",fprintf(fp,"%10.4f",Zdelta(i,j)));
                        if(Zdelta(i,j)>Zcr_d(0,0))
printf("%d",fprintf(fp,"%c",aster));
printf("%d",fprintf(fp,"\n"));
                        }

printf("%d",fprintf(fp,"\n\n"));

if(t2)
printf("%d",fprintf(fp,"EXISTE UM PROVAVEL ERRO NAS MEDIDAS DE CONCENTRACAO(*)\n\n"));

if(!t2)
{
printf("%d",fprintf(fp,"NAO ESTA DETECTANDO PRESENCA DE UM PROVAVEL ERRO NAS MEDIDAS DE
CONCENTRACAO\n"));
printf("%d",fprintf(fp,"OS VALORES DE Z ESTAO ABAIXO DO VALOR CRITICO\n\n"));
}
printf("%d",fprintf(fp,"DESBALANCO\n"));

for(i=0,j=0;j<p;j++)
                        {
printf("%d",fprintf(fp,"%10.4f",Zerr(i,j)));
printf("%d",fprintf(fp,"\n"));
                        }

if((t1!=0&&t3!=0)||(t2!=0&&t3!=0))
{
printf("%d",fprintf(fp,"\nO TESTE DO DESBALANCO PARA A FUNCAO OBJETIVO ESTA CONFIRMANDO\n"));
printf("%d",fprintf(fp,"A PRESENCA DE POSSIVEL ERRO NAS MEDIDAS\n\n"));
}

if(ntes==1 && tg!=0)
{
printf("\nAPENAS O TESTE GLOBAL ESTA APRESENTANDO EVIDENCIA DE ERRO GROSSEIRO\n");
printf("%d",fprintf(fp,"\n\nAPENAS O TESTE GLOBAL ESTA APRESENTANDO EVIDENCIA DE ERRO GROSSEIRO\n"));
}
if(ntes>0)
{
printf("\n\n%d, ENTRE OS 4 TESTES APRESENTOU(ARAM) EVIDENCIAS DE ERROS GROSSEIROS\n",ntes);
printf("%d",fprintf(fp,"\n%d, ENTRE OS 4 TESTES APRESENTARAM EVIDENCIAS DE ERROS GROSSEIROS\n",ntes));
}
if((ntes==1 && tg==1) || ntes==0)
{
printf("\nNAO EXISTE EVIDENCIA DE ERROS GROSSEIROS ENTRE AS MEDIDAS\n");
printf("%d",fprintf(fp,"\n\n\nNAO EXISTE EVIDENCIA DE ERROS GROSSEIROS NAS MEDIDAS\n"));
}
```

```
printf("\n\n\nTECLE ALGO");

if((ntes>0)&&(tg==0)||(ntes>1)&&(tg!=0))
{
x=TM(n1*1);

for(j=0,i=0;i<n1;i++) X(i,j)=Xl[i][j];

free(chit);free(zetr);free(dd);free(de);free(zcr_e);

clrscr();
q=n1;
mvs_a=TMI(n1);
mvs_d=TMI(n2);

*****************************************************************************************************************
ESTA SUBROTINA DETERMINA QUAIS AS MEDIDAS APRESENTADAS ESTÁ APRESENTANDO EVIDÊNCIA DE ERROS GROSSEIROS

DETECTAR(n1,ni,c1,rq,q,t1,t2,vivd,mvs_a,mvs_d,mza,mzd,zaj,zdelta,zcr_a,zcr_d,x,fp);
*****************************************************************************************************************

free(zaj);free(zdelta);free(zcr);free(zcr_a);free(zcr_d);free(x);
}
} //fechar if(vn2)

fclose(fp);
ant+=mat;
fclose(fp);

} //FECHA A FUNCAO RECON_RETIF

*****************************************************************************************************************
COMEÇO DAS SUBROTINAS INTERNAS AO GERENCIADOR
*****************************************************************************************************************

/* SUBROTINA PARA TESTAR QUAL DOS VALORES DE QUI-QUADRADO É MAIOR */

int TESTE_G(pd,chi2,chit)
int pd;
float *chi2,*chit;
{
if(CHI2(0,0)>CHI2_tab(0,pd-1)) return 1;
return 0;
}
#################################################################################################################

/* SUBROTINA PARA ESCOLHA DA OPÇÃO PELO USUÁRIO */

char menu(void)

{  //INÍCIO DA SUBROTINA

  char ch;
  do {  clrscr();
            printf("*********************************************************\n");
            printf(".....ESCOLHA A OPERACAO DESEJADA.....\n");
            printf("*********************************************************\n");
            printf("\n\n\n");
            printf("OPERACOES DISPONIVEIS: ");
            printf("(OBS: LEIA UM ARQUIVO ANTES DE RETIFICAR)\n");
            printf("\n\n\n\n");
            printf("#1.LER UM ARQUIVO DE DADOS.....Tecle (L)\n\n\n");
            printf("#2.RETIFICAR DADOS DE PROCESSO.....Tecle (R)\n\n\n");
            printf("#3.VOLTAR AO MENU PRINCIPAL.....Tecle (S)\n");
            printf("\n\n\n\n\n\n");
            printf("SUA OPCAO");
            ch=toupper(getche());
            }
            while( ch !='S'&& ch!='L'&&  ch !='R');
            printf("\n");
            return ch;
} //FIM DA SUBROTINA

#################################################################################
```

```
/* SUBROTINA PARA LEITURA DE VALORES */

void ler(void)

{ //INÍCIO DA SUBROTINA

FILE *fp;
int i,j;
char s[20];

 printf("\nNome do arquivo:" );
 gets(s);

 if((fp=fopen(s,"r"))==NULL)
            {
                        printf("Nao posso abrir arquivo\n");
                        exit(1);
            }

fscanf(fp,"%d%*c%d%*c%d%*c%d%*c%d%*c",&n1,&n2,&m,&vd,&pd);
//printf("%d"" ""%d"" ""%d"" ""%d"" ""%d",n1,n2,m,vd,pd);
//tt=getche();
fscanf(fp,"%d%*c%d%*c%d%*c%d%*c%d%*c%d%*c%d%*c",&p,&rq,&c1,&blo,&nulo,&ni);
//printf("  %d"" ""%d"" ""%d"" ""%d"" ""%d"" ""%d"" ""%d",p,rq,c1,blo,nulo,ni);
//tt=getche();

for(j=0,i=0;i<n1;i++)
            {
fscanf(fp,"%f%*c",&Xl[i][j]);
//printf("\nX[%d][%d]=%f",i,j,Xl[i][j]);
            }
//tt=getche();

//clrscr();

//printf("VALORES DE B1\n");
for(i=0;i<m;i++)
            {
             for(j=0;j<n1;j++)
                                    {
fscanf(fp,"%f%*c",&B1l[i][j]);
//printf("%10.4f",B1l[i][j]);
                                    }
//printf("\n");
             }
//tt=getche();

for(i=0;i<m;i++)
            for(j=0;j<n2;j++)
fscanf(fp,"%f%*c",&B2l[i][j]);

for(i=0;i<m;i++)
            for(j=0;j<nulo;j++)
fscanf(fp,"%f%*c",&B2Fl[i][j]);

//clrscr();

//printf("VALORES DE YHT\n");
for(i=0;i<p;i++)
            {
             for(j=0;j<m;j++)
                                    {
fscanf(fp,"%f%*c",&YHTl[i][j]);
//printf("%2.0f",YHTl[i][j]);
                                    }
//printf("\n");
             }
//tt=getche();

for(i=0;i<n1;i++)
            for(j=0;j<n1;j++)
fscanf(fp,"%f%*c",&Sl[i][j]);

for(i=0;i<pd;i++)
            for(j=0;j<p;j++)
fscanf(fp,"%f%*c",&ZHTl[i][j]);
```

```
//clrscr();
for(i=0;i<vd;i++)
                {
                 for(j=0;j<vd;j++)
                    {
fscanf(fp,"%f%*c",&Sdl[i][j]);
//printf("%8.4f",Sdl[i][j]);
                    }
//printf("\n");
                }
//tt=getche();

for(j=0,i=0;i<pd;i++)
                {
fscanf(fp,"%f%*c",&Zbl[i][j]);
//printf("\nZb[%d][%d]=%10.4f",i,j,Zbl[i][j]);
                }
//tt=getche();

for(i=0;i<n1;i++)
                for(j=0;j<nt;j++)
fscanf(fp,"%f%*c",&S1B1Tl[i][j]);

for(i=0;i<n2;i++)
                for(j=0;j<n2;j++)
                                {
fscanf(fp,"%f%*c",&Noll[i][j]);
//printf("%10.4f",Noll[i][j]);
                                }
//tt=getche();

//clrscr();

//printf("VALORES DE HH\n");

for(i=0;i<p;i++)
                {
                 for(j=0;j<p;j++)
                                                {
fscanf(fp,"%f%*c",&HHl[i][j]);
//printf("%18.4f",HHl[i][j]);
//tt=getche();
                                                }
//printf("\n");
                }
//tt=getche();

for(i=0;i<pd;i++)
                for(j=0;j<pd;j++)
                                {
fscanf(fp,"%f%*c",&Zainvl[i][j]);
//printf("\nZainv[%d][%d]=%10.4f",i,j,Zainvl[i][j]);
//tt=getche();
                                }
//tt=getche();

for(j=0,i=0;i<n1;i++)
                {
fscanf(fp,"%f%*c",&AJ2l[i][j]);
//printf("\nAJ2[ %d][ %d]=%12.8f",i,j,AJ2l[i][j]);
                }
//tt=getche();

//clrscr();

for(j=0,i=0;i<n2;i++)
                {
fscanf(fp,"%f%*c",&Deltal[i][j]);
//printf("\nDelta[ %d][ %d]=%12.8f",i,j,Deltal[i][j]);
                }
//tt=getche();

//clrscr();

//printf("VALORES DE DD\n");
```

```
for(j=0,i=0;i<n2;i++)
            {
fscanf(fp,"%f%*c",&DDI[i][j]);
//printf("\nDD[%d][%d]=%8.4f",i,j,DDI[i][j]);
            }
//tt=getche();

for(j=0;j<(ni*c1+rq);j++)
            {
fscanf(fp,"%d%*c",&viv[j]);
//printf("\nviv[%d]=%d\n",j,viv[j]);
//tt=getche();
            }
//tt=getche();

fclose(fp);

} //FECHA A SUBROTINA

#########################################################################################################################

/* SUBROTINA DE TRIANGULARIZACAO */

int lu(int rf,int *irf,float *p1f)

{ //INÍCIO DA SUBROTINA

int r1=rf-1;
register int i,j,k;
float a1,a2;
double fabs();

for(irf[r1]=r1,i=0;i<r1;i++)
            {
             for(j=(k=i)+1;j<rf;j++)
                                    if(fabs(P1f(j,i)) > fabs(P1f(k,i))) k=j;
                                    if(k!=i)
                                                for(j=i;j<rf;j++)
                                                            {
                                                             a1=P1f(i,j);
                                                             P1f(i,j)=P1f(k,j);
                                                             P1f(k,j)=a1;
                                                             }
                                    if(!(a2=P1f(i,i),a2))
                                    return (-1);

for(irf[i]=k,j=i+1;j<rf;j++)
            {
             a1=(P1f(j,i)/=-a2);
                        for(k=i+1;k<rf;k++)
                                     P1f(j,k)+=a1*P1f(i,k);
             }
 }

if(!P1f(r1,r1))
            return (-1) ;
            return(0);

} //FIM DA SUBROTINA

#########################################################################################################################

/* SUBROTINA DE INVERSAO DE MATRIZES QUADRADAS */

void vbacks (int rf ,int rs, int *irf, float *p1f, float *uf )

{ //INÍCIO DA SUBROTINA

register int i,j,k;
int r1=rf-1;
float a1,mat,ant;
for(i=0;i<r1;i++)
            {
             if((j=irf[i]) != i
                                    for(k=0;k<rs;k++)
                                                {
```

```
                                        a1=Uf(i,k);
                                        Uf(i,k)=Uf(j,k);
                                        Uf(j,k)=a1;
                                        }

for(j=i+1;j<rf;j++)
        for(k=0;k<rs;k++)
                Uf(j,k)+=P1f(j,i)*Uf(i,k);
        }
                ant=P1f(r1,r1);

for(k=0;k<rs;k++)
        Uf(r1,k)/=P1f(r1,r1);

for(i=r1-1;i>=0;i--)
        {
for(a1=P1f(i,i),j=i+1;j<rf;j++)
        for(k=0;k<rs;k++)
                Uf(i,k)-=P1f(i,j)*Uf(j,k);

for(k=0;k<rs;k++)
        {
         Uf(i,k) /= a1;
         mat=Uf(i,k);
        }
}
mat+=mat;ant+=ant;

}

#include ""..."\sub\DEN2a.c"
#include ""..."\sub\DEN2delta.c"
#include ""..."\sub\DEN2Err.c"
#include ""..."\sub\TEG.c"
#include ""..."\sub\nmsus.c"
#include ""..."\sub\DETECTAR.c"
#include ""..."\sub\ZHTf.c"

// FIM DA SUBROTINA

##########################################################################################################
FIM DO PROGRAMA GERENCIADOR RETIF.C
##########################################################################################################

##########################################################################################################
CONJUNTO DE SUBROTINAS, EXTERNAS AOS PROGRAMAS, NECESSÁRIAS AO FUNCIONAMENTO DOS
PROGRAMAS RECON E RETIF. DEVEM ESTAR DEFINIDAS EM OUTROS ARQUIVOS CONFORME ESPECIFICADO NOS
RESPECTIVOS GERENCIADORES.
##########################################################################################################

##########################################################################################################
SUBROTINA DEN2a.C
##########################################################################################################

SUBROTINA COM OS VALORES DO DENOMINADOR AO QUADRADO REFERENTE AO AJUSTE

#include<math.h>
#include<math.h>
#include<alloc.h>
#include<ctype.h>
#include<string.h>
#include<conio.h>

#define B1(i,j) (*(b1+n1*(i)+j))
#define S1B1T(i,j) (*(sbt+m*(i)+j))
#define S(i,j) (*(pps+n1*(i)+j))
#define YHT(i,j) (*(yh+m*(i)+j))
#define Zainv(i,j) (*(zai+pd*(i)+j))
#define ZHT(i,j) (*(zh+p*(i)+j))
#define YY(i,j) (*(yy+m*(i)+j))
#define Q1jj(i,j) (*(q1+n1*(i)+j))

#define YZT(i,j) (*(yzt+p*(i)+j))
#define YZTi(i,j) (*(yzti+pd*(i)+j))
#define YZTT(i,j) (*(yztt+p*(i)+j))
#define YYB1(i,j)  (*(yyb1+n1*(i)+j))
```

```
#define YY(i,j)  (*(yy+m*(i)+j))
#define YBS(i,j)  (*(ybs+n1*(i)+j))
#define Q1jj(i,j)  (*(q1+n1*(i)+j))

#define TM(size) (float *)malloc(size*sizeof(float))

void DEN2a(p,pd,n1,m,b1,sbt,pps,yh,zai,zh,yy,q1)

 int p,n1,m,pd;
 float *b1,*sbt,*pps,*yh,*zai,*zh;
 float *yy,*q1;
 {
 register int i,j,z;
 double aux;
 float *syt,*yzt,*yzti,*yztt,*yyb1,*ybs,ant,mat;
 char tt;
aux=0;

//DETERMINACAO DE Q1jj

yzt=TM(m*pd);
yzti=TM(m*pd);
yztt=TM(m*p);
yyb1=TM(m*n1);
//yy=TM(m*m);
ybs=TM(m*n1);

/*
clrscr();
printf("VALORES DE ZHT\n");
for(i=0;i<pd;i++)
            {
            for(j=0;j<p;j++)
                        printf("%8.4f",ZHT(i,j));
                        printf("\n");
             }
tt=getche();

clrscr();
printf("VALORES DE YHT\n");
for(i=0;i<p;i++)
  {
  for(j=0;j<m;j++)
            printf("%2.0f",YHT(i,j));
            printf("\n");
  }
tt=getche();
*/

 for(i=0;i<m;i++)
            for(j=0;j<pd;j++)
             {
                        for(z=0;z<p;z++)
             {
              ant=YHT(z,i); mat=ZHT(j,z);
              aux+=YHT(z,i)*ZHT(j,z);
             }
             YZT(i,j)=aux;
             mat=aux;
             aux=0;
                        }
/*
clrscr();
printf("VALORES DE Zainv\n");
for(i=0;i<pd;i++)
  {
  for(j=0;j<pd;j++)
             printf("%8.4f",Zainv(i,j));
             printf("\n");
            }
tt=getche();
*/


 for(i=0;i<m;i++)
            for(j=0;j<pd;j++)
```

```
                {
                        for(z=0;z<pd;z++)
                {
                ant=YZT(i,z);  mat=Zainv(z,j);
                aux+=YZT(i,z)*Zainv(z,j);
                }
                YZTi(i,j)=aux;
                aux=0;
                        }

free(yzt);

 for(i=0;i<m;i++)
                for(j=0;j<p;j++)
                        {
                for(z=0;z<pd;z++)
                 {
                        ant=YZTi(i,z);  mat=ZHT(z,j);
                        aux+=YZTi(i,z)*ZHT(z,j);
                 }
                 YZTT(i,j)=aux;
                 mat=YZTT(i,j);
                 aux=0;
                        }

 free(yzti);

 for(i=0;i<m;i++)
                for(j=0;j<m;j++)
                        {
                for(z=0;z<p;z++)
                 {
                        ant=YZTT(i,z);  mat=YHT(z,j);
                        aux+=YZTT(i,z)*YHT(z,j);
                 }
                 YY(i,j)=aux;
                 mat=YY(i,j);    //ESTE VALOR VAI SER USADO NA SEGUNDA PARTE
                 aux=0;          //POIS SAO IGUAIS.
                        }


 free(yzt);

 for(i=0;i<m;i++)
                for(j=0;j<n1;j++)
                        {
                for(z=0;z<m;z++)
                 {
                        ant=YY(i,z); mat=B1(z,j);
                        aux+=YY(i,z)*B1(z,j);
                 }
                 YYB1(i,j)=aux;
                 mat=YYB1(i,j);
                 aux=0;
                        }


  for(i=0;i<m;i++)
                 for(j=0;j<n1;j++)
                        {
                for(z=0;z<n1;z++)
                 {
                        ant=YYB1(i,z);  mat=S(z,j);
                        aux+=YYB1(i,z)*S(z,j);
                 }
                 YBS(i,j)=aux;
                 mat=YBS(i,j);
                 aux=0;
                }


  free(yyb1);


  for(i=0;i<n1;i++)
                 for(j=0;j<n1;j++)
```

```
                    {
                    for(z=0;z<m;z++)
            {
             ant=S1B1T(i,z); mat=YBS(z,j);
             aux+=S1B1T(i,z)*YBS(z,j);
            }
            Q1jj(i,j)=aux;
            mat=Q1jj(i,j);
            aux=0;
                    }
/*
printf("\nVALORES DE Q1jj\n");
for(i=0;i<n1;i++)
 {
 for(j=0;j<n1;j++)
 if(i==j)
            {
            {
             printf("\n%10.8f",Q1jj(i,j));
            }
            printf("\n");
            }
 }
             tt=getche();
*/
free(ybs);
} //FECHA A SUBROTINA


#############################################################################################################
SUBROTINA DEN2Delta.C
#############################################################################################################

/* SUBROTINA PARA O CÁLCULO DO DENOMINADOR AO QUADRADO REF AO DELTA */

#define B2(i,j) (*(b2+n2*(i)+j))
#define Sd(i,j) (*(ppd+vd*(i)+j))
#define No(i,j) (*(no+n2*(i)+j))
#define YY(i,j) (*(yy+m*(i)+j))
#define Q2ll(i,j) (*(q2+n2*(i)+j))

#define SdNo(i,j) (*(sdn+n2*(i)+j))
#define SdYY(i,j) (*(sdyy+n2*(i)+j))
#define SdNB2(i,j) (*(sdnb+m*(i)+j))
#define SdYB(i,j) (*(sdyb+n2*(i)+j))
#define SdN2(i,j) (*(sdn2+n2*(i)+j))
#define Q2ll(i,j) (*(q2+n2*(i)+j))

#define TM(size)  (float *)malloc(size*sizeof(float))

void DEN2delta(n2,m,ppd,no,yy,b2,q2)

int n2,m;
float *ppd,*no,*yy,*b2;
float *q2;

{
float *sdn,*sdyy,*sdnb,*sdyb,*sdn2,ant,mat;
register int i,j,z;
double aux;
char tt;
aux=0;

sdn=TM(n2*n2);
sdyy=TM(n2*m);
sdnb=TM(n2*m);
sdyb=TM(n2*n2);
sdn2=TM(n2*n2);
//q2=TM(n2*n2);

/*
printf("CONFERIR OS VALORES LIDOS EM DEN2delta\n");
printf("\nVALORES DE Sd e No\n");
for(i=0;i<n2;i++)
            for(j=0;j<n2;j++)
```

```
            if(i==j)
            {
            printf("%10.4f"".....""%10.4f",Sd(i,j),No(i,j));
            printf("\n");
            }
tt=getche();

printf("VALORES DE YY\n");
for(i=0;i<m;i++)
 {
 for(j=0;j<m;j++)
            printf("%8.4f",YY(i,j));
            printf("\n");
 }
tt=getche();

clrscr();
printf("VALORES DE B2\n");
for(i=0;i<m;i++)
            {
            for(j=0;j<n2;j++)
                        printf("%2.0f",B2(i,j));
                        printf("\n");
            }
printf("\nFIM DO DEN2delta\n");
tt=getchar();
*/

for(i=0;i<n2;i++)
 for(j=0;j<n2;j++)
            {
                        for(z=0;z<n2;z++)
            {
            ant=Sd(i,z); mat=No(z,j);
            aux+=Sd(i,z)*No(z,j);
            }
            SdNo(i,j)=aux;
            mat=SdNo(i,j);
            aux=0;
                        }

for(i=0;i<n2;i++)
 for(j=0;j<m;j++)
            {
                        for(z=0;z<n2;z++)
            {
                        ant=SdNo(i,z); mat=B2(j,z);
                        aux+=SdNo(i,z)*B2(j,z);
            }
            SdNB2(i,j)=aux;
            mat=SdNB2(i,j);
            aux=0;
                        }

free(sdn);

for(i=0;i<n2;i++)
 for(j=0;j<m;j++)
            {
            for(z=0;z<m;z++)
                        {
            ant=SdNB2(i,z); mat=YY(z,j);
            aux+=SdNB2(i,z)*YY(z,j);  //MESMO YY DA OUTRA SUBRTINA(DEN2a.C)
                        }
                        SdYY(i,j)=aux;
                        mat=aux;
                        aux=0;
            }


free(sdnb);


for(i=0;i<n2;i++)
 for(j=0;j<n2;j++)
```

```
                {
                        for(z=0;z<m;z++)
                {
                 ant=SdYY(i,z); mat=B2(z,j);
                 aux+=SdYY(i,z)*B2(z,j);
                }
                SdYB(i,j)=aux;
                mat=aux;
                aux=0;
                 }
free(sdyy);

for(i=0;i<n2;i++)
 for(j=0;j<n2;j++)
                {
                        for(z=0;z<n2;z++)
                {
                 ant=SdYB(i,z);  mat=No(z,j);
                 aux+=SdYB(i,z)*No(z,j);
                }
                SdN2(i,j)=aux;
                mat=SdN2(i,j);
                aux=0;
                 }
free(sdyb);

for(i=0;i<n2;i++)
 for(j=0;j<n2;j++)
                {
                        for(z=0;z<n2;z++)
                {
                        ant=SdN2(i,z); mat=Sd(z,j);
                        aux+=SdN2(i,z)*Sd(z,j);
                }
                Q2ll(i,j)=aux;
                mat=aux;
                aux=0;
                        }

/*
printf("\nVALORES DE Q2ll\n");
for(i=0;i<n2;i++)
for(j=0;j<n2;j++)
{
if(i==j)
printf("\nQ2ll(%d,%d)=%10.4f",i,j,Q2ll(i,j));
}
tt=getche();
*/

free(sdn2);
mat+=ant;

}//FECHA A SUBROTINA


##########################################################################################################
SUBROTINA DEN2ERR.C
##########################################################################################################

/*SUBROTINA PARA O CALCULO DO DENOMINADOR QUADRADO REF AO ERRO */

#define Zainv(i,j) (*(zai+pd*(i)+j))
#define Zb(i,j) (*(zb+1*(i)+j))
#define HH(i,j) (*(hh+p*(i)+j))
#define ZHT(i,j) (*(zh+p*(i)+j))
#define HZ(i,j) (*(hz+pd*(i)+j))
#define HZT(i,j) (*(hzt+pd*(i)+j))
#define HZTZ(i,j) (*(hztz+p*(i)+j))
#define Err(i,j)  (*(err+p*(i)+j))

#define TM(size) (float *)malloc(size*sizeof(float))

void DEN2err(p,pd,zai,zb,hh,zh,hzt,err)
int p,pd;
float *zai,*zb,*hh,*zh;
```

```
float *hzt,*err;
{
float *hz,*hztz,ant,mat;
register i,j,z;
double aux;

aux=0;
hz=TM(p*pd);
hztz=TM(p*p);
/*
clrscr();
printf("PARA CONFERIR OS VALORES EM DEN2err\n");
printf("VALORES DE Zainv\n");
for(i=0;i<pd;i++)
            {
            for(j=0;j<pd;j++)
                        printf("%12.8f",Zainv(i,j));
                        printf("\n");
             }
tt=getche();

clrscr();
printf("VALORES DE Zb\n");
for(j=0,i=0;i<pd;i++)
 {
            printf("%12.8f",Zb(i,j));
            printf("\n");
 }
tt=getche();

clrscr();
printf("VALORES DE HH\n");
for(i=0;i<p;i++)
 {
 for(j=0;j<p;j++)
             printf("%12.8f",HH(i,j));
             printf("\n");
             }
tt=getche();

clrscr();
printf("VALORES DE ZHT\n");
for(i=0;i<pd;i++)
            {
            for(j=0;j<p;j++)
                        printf("%8.4f",ZHT(i,j));
                        printf("\n");
             }
tt=getche();

clrscr();
printf("VALORES DE HZT\n");
for(i=0;i<p;i++)
 {
 for(j=0;j<pd;j++)
             printf("%8.4f",HZT(i,j));
             printf("\n");
             }
printf("FIM DOS VALORES PARA DEN2err\n");
tt=getche();
*/
for(i=0;i<p;i++)
 for(j=0;j<pd;j++)
             {
             for(z=0;z<p;z++)
                        {
            ant=HH(i,z);  mat=ZHT(j,z);
            aux+=HH(i,z)*ZHT(j,z);
                        }
            HZ(i,j)=aux;
            mat=HZ(i,j);
            aux=0;
                        }

for(i=0;i<p;i++)
 for(j=0;j<pd;j++)
```

```
                {
                 for(z=0;z<pd;z++)
                                {
                ant=HZ(i,z); mat=Zainv(z,j);
                aux+=HZ(i,z)*Zainv(z,j);
                                }
                                HZT(i,j)=aux;
                                mat=HZT(i,j);
                                aux=0;
                 }

free(hz);

 for(i=0;i<p;i++)
                for(j=0;j<p;j++)
                 {
                                 for(z=0;z<pd;z++)
                 {
                                ant=HZT(i,z); mat=ZHT(z,j);
                                aux+=HZT(i,z)*ZHT(z,j);
                 }
                 HZTZ(i,j)=aux;
                 mat=HZTZ(i,j);
                 aux=0;
                                 }
//clrscr();

 for(i=0;i<p;i++)
  {
  for(j=0;j<p;j++)
                 {
                  for(z=0;z<p;z++)
                                 {
                ant=HZTZ(i,z); mat=HH(z,j);
                aux+=HZTZ(i,z)*HH(z,j);
                                 }
                                 Err(i,j)=aux;
                                 mat=Err(i,j);
//                               printf("%12.8f",Err(i,j));
                                 aux=0;
                                 }
//                               printf("\n");
                 }
 //tt=getche();

 free(hztz);

 }//FECHA A SUBROTINA

 ##########################################################################################################################
 SUBROTINA VMSUS.C
 ##########################################################################################################################

 /* SUBROTINA DE CONSTRUÇÃO DO VETOR DE MEDIDAS SUSPEITAS */

 void vmsus(n1,mza,mvs,zaj,zcr)

 int n1;
 int *mza,*mvs;
 float *zaj,*zcr;

 {
 float ant,mat;
 int counter;
 register int i,j,k;

 ************************************************************************************************************************
 /*clrscr();
 printf("TESTE COM OS VALORES QUE ENTRAM EM vmsus\n");
 for(j=0,i=0;i<n1;i++)
                 {
                 printf("Z(a/d)(0,%d)=%10.6f",i,Zaj(j,i));
                 printf("\n");
                 }
 printf("\n\n\n\n\n\n\n");
 printf("Zcr(a/d)(*)=%8.4f",Zcr(0,0));
```

```
tt=getche();
*/
*****************************************************************************************************************************

for(j=0;j<n1;j++) mza[j]=j;

for(k=0;k<n1-1;k++)
 {
        for(i=0,j=k+1;j<n1;j++)
                {
                 ant=Zaj(i,k); mat=Zaj(i,j);
                 if((fabs(fabs(Zaj(i,k))-fabs(Zaj(i,j)))<=0.01)&&((Zaj(i,k)!=0)&&(Zaj(i,j)!=0)))
        if(mza[j]==j) mza[j]=k;
        mat=mza[j];
                }
        }

for(counter=0,j=0;j<n1;j++)
        if(mza[j]==j) counter++;

for(j=0;j<n1;j++)
        {
        mvs[j]=j;
        mat=mvs[j];
        }

 for(i=0,j=0;j<n1;j++)
        {
                ant=fabs(Zaj(i,j)); mat=Zcr(0,0);
                if(fabs(Zaj(i,j))>Zcr(0,0))
                mvs[j]=1000; //MARCADOR (NO VETOR) DE MEDIDAS SUSPEITAS
                mat=mvs[j];
        }

}//FECHA A SUBROTINA

##########################################################################################################################
TEG.C
##########################################################################################################################

/* SUBROTINA DE DETERMINAÇÃO DO TESTE ESTATÍSTICO GLOBAL */

#include<math.h>
#include<math.h>
#include<alloc.h>
#include<ctype.h>
#include<string.h>
#include<conio.h>

#define CHI(i,j) (*(chi+pd*(i)+j))
//#define CHI2(i,j) (*(chi2+1*(i)+j))
#define TM(size) (float *)malloc(size*sizeof(float))

void TEG(int p,int pd,int n1, float *zb,float *zai,float *chi2)
{
float *chi,ant,mat;
register int i,j,z;
double aux;

chi=TM(1*pd);

for(i=0;i<1;i++)
 for(j=0;j<pd;j++)
  {
   for(z=0;z<pd;z++)
    {
        ant=Zb(z,i); mat=Zainv(z,j);
        aux+=Zb(z,i)*Zainv(z,j);
    }
    CHI(i,j)=aux;
    mat=CHI(i,j);
    aux=0;
  }

for(i=0;i<1;i++)
 for(j=0;j<1;j++)
```

```
  {
   for(z=0;z<pd;z++)
            {
            ant=CHI(i,z);  mat=Zb(z,j);
            aux+=CHI(i,z)*Zb(z,j);
            }
            CHI2(i,j)=aux;
            mat=CHI2(i,j);
            aux=0;
   }

free(chi);

}//FECHA A SUBROTINA
```

```
#########################################################################################################
NMSUS.C
#########################################################################################################

/*SUBROTINA QUE CALCULA O NUMERO DE MEDIDAS SUSPEITAS DE TEREM ERROS */


/*
#include<math.h>
#include<stdio.h>
#include<conio.h>
*/
#define Zaj(i,j) (*(zaj+n1*(i)+j))
#define Zdelta(i,j) (*(zdelta+n2*(i)+j))
#define Zerr(i,j) (*(zerr+p*(i)+j))
#define Zcr(i,j) (*(zcr+30*(i)+j))
#define Zcr_a(i,j) (*(zcr_a+1*(i)+j))
#define Zcr_d(i,j) (*(zcr_d+1*(i)+j))
#define Zcr_e(i,j) (*(zcr_e+1*(i)+j))

int nmsus(n1,mza,zaj,zcr,zcr_a,bool,fp)

int n1,*mza;
float *zaj,*zcr,*zcr_a;
int bool;
FILE *fp;

{
float mat,ant;
register int i,j,k;
int counter=0, nms=0;
char tt;

for(j=0;j<n1;j++) mza[j]=j;

for(k=0;k<n1-1;k++)
            {
             for(i=0,j=k+1;j<n1;j++)
                        {
                        if((fabs(fabs(Zaj(i,k))-fabs(Zaj(i,j)))<=0.01)&&((Zaj(i,k)!=0)&&(Zaj(i,j)!=0)))
             if(mza[j]==j) mza[j]=k;
                        }
             }

for(counter=0,j=0;j<n1;j++)
  if(mza[j]==j) counter++;

if(!counter) counter++;

for(j=0,i=0;i<30;i++)
{
mat=Zcr(i,j);
}

/* NÚMERO DE MEDIDAS SUSPEITAS */

printf("%d",fprintf(fp,"\nNUMERO DE MEDIDAS SUSPEITAS(nms)\n"));
for(nms=0,i=0,j=0;j<n1;j++)
            {
```

```
                ant=Zaj(i,j); mat=Zcr(counter-1,0);
                if(fabs(Zaj(i,j))>Zcr(counter-1,0)) nms++;
                }
               Zcr_a(i,0)=Zcr(counter-1,0);
               mat=Zcr_a(i,0);
printf("%d",fprintf(fp,"\nZc(tabelado(%d))=%10.4f\n",bool,Zcr(counter-1,0)));
printf("\nVALOR DE Zcr(%d)=%10.4f"".....""nms=%d",counter-1,Zcr(counter-1,0),nms);
tt=getche();
ant+=mat;
bool+=mat;
return nms;

}

##########################################################################################################
DETECTAR.C
##########################################################################################################

/* SUBROTINA PARA DETECCAO DE ERROS GROSSEIROS*/

#include<math.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#include ""..."\sub\vmsus.c"

#define Zcr(i,j) (*(zcr+30*(i)+j))
#define Zcr_a(i,j) (*(zcr_a+1*(i)+j))
#define Zcr_d(i,j) (*(zcr_d+1*(i)+j))
#define Zaj(i,j) (*(zaj+n1*(i)+j))
#define Zdelta(i,j) (*(zdelta+n2*(i)+j))
#define Zdeltar(i,j) (*(zdeltar+n2*(i)+j))
#define Zajr(i,j) (*(zajr+n1*(i)+j))
#define X(i,j) (*(x+1*(i)+j))

#define TM(size) (float *)malloc(size*sizeof(float))

void DETECTAR(n1,ni,c1,rq,q,t1,t2,viv,mvs_a,mvs_d,mza,mzd,zaj,zdelta,zcr_a,zcr_d,x,fp)

int n1,ni,c1,rq,q,*viv,*mvs_a,*mvs_d,*mza,*mzd,t1,t2; //t1=nms
float *zaj,*zdelta,*zcr_a,*zcr_d,*x;
FILE *fp;

{
char tt;
char aster;
int del,dec,rec,mar,mar_d,val;
float *zajr,*zdeltar,ant,mat,aux;
register int i,j,k,kj;

**********************************************************************************************************************
/*
clrscr();
printf("TESTE COM OS VALORES DE ENTRADA EM DETECTAR\n");
for(j=0,i=0;i<n1;i++)
               {
               printf("Zaj(%d)=%8.4f",i,Zaj(j,i));
               printf("\n");
               }
tt=getche();

clrscr();
                printf("Zcr_a(*)=%8.4f",Zcr_a(0,0));
                printf("\n");
tt=getche();


clrscr();
for(j=0,i=0;i<n1;i++)
  {
               printf("X(%d)=%10.6f",i,X(i,j));
               printf("\n");
               }
tt=getche();
```

```
for(i=0;i<ni*(c1)+rq;i++)
            {
            printf("viv[%d]=%d",i,viv[i]);
            printf("\n");
            }
tt=getche();
*/
*****************************************************************************************************************************
clrscr();
printf("\nEXISTEM EVIDENCIAS DE ERRO(S) NAS MEDIDAS ORIGINAIS\n");
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\TECLE ALGO PARA CONTINUAR\n");
printf("%d",fprintf(fp,"\nNUMERO DE MEDIDAS SUSPEITAS= %d\n",t1+t2));
tt=getche();

/* CHAMADA DA FUNCAO QUE DESCREVE AS MEDIDAS SUSPEITAS */

vmsus(n1,mza,mvs_a,zaj,zcr_a);
vmsus(n2,mzd,mvs_d,zdelta,zcr_d);

zajr=NULL;
zajr=TM(1*n1);
aster='*';

*****************************************************************************************************************************
/* TESTE COM OS VALORES QUE SAEM DE vmsus */
/*
clrscr();
for(i=0;i<n1;i++)
{
printf("mvs_a[%d]=%d",i,mvs_a[i]);
printf("\n");
}
tt=getche();

clrscr();
for(i=0;i<n2;i++)
{
printf("mvs_d[%d]=%d",i,mvs_d[i]);
printf("\n");
}
tt=getche();
*/
*****************************************************************************************************************************
/*
for(k=0,kj=0,i=0;i<n1;i++)
// for(j=0;j<3;j++)
            {
            if(mvs_a[i]==1000)
                        {
            mat=Zaj(k,i);
            Zajr(k,kj)=Zaj(k,i);   //VETOR DOS Za REDUZIDOS
//          printf("%d",fprintf(fp,"%c",CM(i,j)));
//          printf("%c",CM(i,j));
            kj++;
                        }
             }
*/

/* CASO NAO EXISTA ERRO EM NENHUMA MEDIDA, ABORTAR A DETECÇÃO. */
/*
dec=0;
printf("\nDigite (1) para CONTINUAR ou (0) para ABORTAR\n");
tt=getchar();
scanf("%d%*c",&dec);
if(dec==0)
{
printf("%d",fprintf(fp,"\nA DETECCAO FOI ABORTADA\n"));
goto fim;
}

*/


for(i=0;i<n1;i++)
            Zajr(0,i)=0;
```

```
for(k=0,i=0;i<n1;i++)
 {
            if(mvs_a[i]==1000)
             {
             mat=Zaj(0,i);
             Zajr(0,k)=Zaj(0,i);
             k++;
             }
 }

zdeltar=TM(1*n2);
for(j=0,i=0;i<n2;i++)
          Zdeltar(j,i)=0;
for(k=0,i=0;i<n2;i++)
            {
             if(mvs_d[i]==1000)
                        {
                                    Zdeltar(0,k)=Zdelta(0,i);
                                    if(t2==1)
                                    mar_d=i;
                                    mat=Zdeltar(0,k);
                                    k++;
                         }
                        }


/* SELEÇÃO DA MEDIDA COM O MAIOR ERRO */
for(aux=0,i=0;i<n1;i++)
            aux+=mvs_a[i];

mar=0;
if(aux>999)
{
mar=0;
 for(i=0,k=0,j=k+1;j<t1;j++)
            if(fabs(Zajr(i,k)<Zajr(i,j)))
            {
             ant=Zajr(i,k);  mat=Zajr(i,j);
             Zajr(i,k)=Zajr(i,j);
             mar=j;
            }

printf("%d",fprintf(fp,"\nEXCLUSAO DA MEDIDA MAIS SUSPEITA\n"));
printf("MEDIDA COM MAIOR SUSPEITA DE ERRO GROSSEIRO NOS AJUSTES E'A NUM= %d",mar+1);
tt=getche();
if(!mar)
mar=1;
}
free(zajr);

for(aux=0,i=0;i<n2;i++)
            aux+=mvs_d[i];

if(!mar_d)
mar_d=0;
if(aux>999)
{
if(t2>1)
{
 for(kj=0,i=0,k=0,j=k+1;j<t2;j++)
            {
            if(fabs(Zdeltar(i,k)<Zdeltar(i,j)))
            {
             ant=Zdeltar(i,k);  mat=Zdeltar(i,j);
             if(kj<t2)
//           {
             Zdeltar(i,k)=Zdeltar(i,j);
             mar_d=j;
             kj++;
//           }
            }
            }
 }
printf("%d",fprintf(fp,"\nEXCLUSAO DA MEDIDA MAIS SUSPEITA\n"));
printf("MEDIDA COM MAIOR SUSPEITA DE ERRO GROSSEIRO NAS CONC E'A NUM= %d",n1+mar_d+1);
```

```
tt=getche();
}

free(zdeltar);
free(zajr);

/* MUDANCA NOS VALORES DE viv E X */

 if((mar!=0 && mar_d==0) || (mar==0 && mar_d!=0))

{
del=0;
if(mar!=0 && mar_d==0)
k=mar;
if(mar_d!=0 && mar==0)
k=mar_d;
{
for(j=0;j<ni;j++)
for(i=0;i<(c1);i++) //tirei o rq
{
if(del<1)
{
 if(mar)
 {
 if(viv[j*(c1)+i]==1 && (j*c1+i)==k)
              {
               if(k!=((c1-1)*(j+1)))
                           {
                            viv[k]=3;
                            val=k;
                            del++;
                            X(k,0)=0;
                           }
                           else
                           {
                           printf("%d",fprintf(fp,"TODOS AS MEDIDAS DESTA CORRENTE PRECISAM SER ABORTADAS\n"));
                           printf("%d",fprintf(fp,"POIS EXISTE EVIDENCIA DE MAIOR ERRO NA CORRENTE TOTAL\n"));
                           del++;
                           }
              }
             }
             else if(mar_d)
              {
               if(viv[j*c1+i]==2 && (j*c1+i)==k)
                           {
                            viv[n1+k]=3;
                            val=n1+k;
                            del++;
                           }
              }
}

else
mat+=mat;
}
}

if(mar)
{
printf("%d",fprintf(fp,"\n\nNOVOS VALORES DE X\n"));

printf("%d",fprintf(fp,"A MEDIDA RETIRADA APARECE COM VALOR NULO\n"));
for(i=0,j=0;j<q;j++)
{
printf("%d",fprintf(fp,"X(%d,%d)=%10.4f",j,i,X(i,j)));
printf("%d",fprintf(fp,"\n"));
}
}
if(!mar)
printf("%d",fprintf(fp,"OS VALORES DE X NAO SERAO MUDADOS\n\n"));

printf("%d",fprintf(fp,"NOVOS VALORES DE viv(%d valores)\n\n\n",(ni*c1+rq)));
for(i=0;i<ni*c1+rq;i++)
 {
 printf("%d",fprintf(fp,"%d",viv[i]));
 if(viv[i]==viv[val] && val!=0)
```

```
 {
 printf("%d",fprintf(fp,"(%c)",aster));
 val=0;
 }
 printf("%d",fprintf(fp,"  "));
 }
}

else
{
printf("%d",fprintf(fp,"EXISTEM MEDIDAS CUJOS VALORES ESTAO AFETANDO OS TESTES ESTATISTICOS\n"));
printf("%d",fprintf(fp,"EXISTEM EVIDENCIAS DE ERROS NAS VARIAVEIS DOS TIPOS 1 e 2\n"));
}

fim:;
}//FECHA A SUBROTINA DETECTAR

#########################################################################################################
 ZHTf.C
#########################################################################################################

/*  SUBROTINA PARA O CÁLCULO DE Z DEPOIS DE CALCULADO O DELTA. */

#include<math.h>
#include<math.h>
#include<alloc.h>
#include<ctype.h>
#include<string.h>
#include<conio.h>
#include ""*..."sub\subs1.h"
#include ""*..."gaussy.c"

#define DE(i,j) (*(de+c1*(i)+j))
#define DD(i,j) (*(dd+1*(i)+j))
#define B2DE(i,j) (*(bde+blo*(i)+j))
#define UD(i,j) (*(ud+pd*(i)+j))
#define DT(i,j) (*(dt+p*(i)+j))
#define D(i,j) (*(d+blo*(i)+j))
#define DL(i,j) (*(dl+blo*(i)+j))
#define HN(i,j) (*(hnd+p*(i)+j))
#define DAF(i,j) (*(daf+blo*(i)+j))
#define D1(i,j) (*(d1+blo*(i)+j))
#define D2(i,j) (*(d2+rd*(i)+j))
#define D3(i,j) (*(d3+blo*(i)+j))
#define U2(i,j) (*(u2+blo*(i)+j))
#define D2U(i,j) (*(du+rd*(i)+j))
#define ZT(i,j)  (*(zt+rd*(i)+j))
#define ZTI(i,j) (*(zti+rd*(i)+j))
#define UD(i,j) (*(ud+pd*(i)+j))
#define P1f(i,j) (*(p1f+rf*(i)+j))
#define Uf(i,j) (*(uf+rf*(i)+j))
#define P2Uf(i,j) (*(puf+rf*(i)+j))
#define P2f(i,j) (*(p2f+rf*(i)+j))
#define ZHT(i,j) (*(zh+p*(i)+j))
#define ZHTg(i,j) (*(zhg+p*(i)+j))
#define YT(i,j) (*(y+m*(i)+j))
#define YHT(i,j) (*(yh+m*(i)+j))

#define TM(size) (float *)malloc(size*sizeof(float))
#define TMI(size) (int *)malloc(size*sizeof(int))

void ZHTf(ni,p,m,rq,blo,nulo,c1,viv,dd,yh,b2f,zhg)

float *dd,*yh,*b2f,*zhg;
int p,m,blo,ni,rq,nulo,c1,*viv;

{
float *de,*hnd,*dt,*bde,*d,*dl,*daf,*u2,*d1,*d2,*d3,*ud,*zt,*zti,*d2u,*du;
float *zh;
int*pdf,*rdf;
int *ic2,*ip2,*ird,p11,pd,td,rd;
float aux1,aux2,aux3,aux4,ant,aux5,mat;
register i,j,ij,k,k2,ki,z,t,t2,s,kt;
char ttt;
de=TM((blo*c1));
```

```
/*
clrscr();
printf("VALORES DE DD(reconciliados)\n");
for(j=0,i=0;i<n2;i++)
{
printf("DD(%d,0)=%8.4f",i,DD(i,j));
printf("\n");
}
ttt=getche();
*/

//printf("VALORES DE DE\n");
for (ij=0, i=0;i<blo;i++)
  { for(j=0; j<c1; j++)
                {
                ant=DD(ij,0);
                DE(i,j)=DD(ij,0);
//              printf("  DE(%d,%d)=%8.4f",i,j,DE(i,j));
                ij++;
                }
//              printf("\n");
 }
//ttt=getche();

for(ki=0,i=0;i<ni;i++)
 for(j=0;j<c1;j++)
                {
                        if((viv[i*c1+j+rq]==2) || (viv[i*c1+j+rq]==100))
                {
                viv[ki]=viv[i*c1+j+rq];
//              printf("viv[%d]=%d",ki,viv[i*c1+j+rq]);
                ki++;
                }
//              printf("\n");
                }
//ttt=getche();

/*
clrscr();
printf("VALORES DE B2F\n");
 for(i=0;i<m;i++)
                {
                for(j=0;j<nulo;j++)
                        printf("%2.0f",B2F(i,j));
                        printf("\n");
                }
ttt=getche();
*/
//
bde=TM(m*blo);
dt=TM(blo*p);

                for(i=0;i<c1;i++)
                        for(j=0;j<blo;j++)
                                ant=DE(i,j);
*bde=0;
aux1=0; kt=0;
//clrscr();
//printf("VALORES DE B2DE\n");
 for(t=0,ki=0,i=0;i<blo;i++)
                {  //1
                        for(k=0;k<m;k++)
                                {  //2
                                 for(j=0;j<c1;j++)
                                        {  //3
                                        ant=viv[i*c1+j+rq];
                                        if((viv[i*c1+j+rq]==2) || (viv[i*c1+j+rq])==100)
                                                { //4
                                                        ki=j;
                                                        ant=B2F(k,i*c1+ki);  mat=DE(i,kt);
                                                        aux1+=B2F(k,i*c1+ki)*DE(i,kt);
                                                        kt++;
                                                        mat=aux1;
                                                } //\\4
                                        }    //\\3
                                        kt=0;
```

```
                                                    { //5
                                                    //if (fabs(aux1)<0.0001)
                                                    //B2DE(k,t)=0;
                                                    //if(fabs(aux1>0.0001))
                                                    B2DE(k,t)=aux1;
                                                    aux1=0;
                                                    ki=0;
                                                    } //\\5
                                        } t++; //\\2
                                      }   //\\1

/*
for(i=0;i<m;i++)
 {
 for(j=0;j<blo;j++)
            printf("%8.4f",B2DE(i,j));
            printf("\n");
 }
ttt=getche();
*/
/*
clrscr();
printf("VALORES DE YHT(em ZHTf)\n");
for(i=0;i<p;i++)
            {
            for(j=0;j<m;j++)
                        printf("%2.0f",YHT(i,j));
                        printf("\n");
            }
ttt=getche();
*/

//clrscr();
//printf("VALORES DE DT\n");
            for(i=0;i<blo;i++)
            {
             for(j=0;j<p;j++)
                        {
                         for(z=0;z<m;z++)

                                    {
                                                ant=B2DE(z,i);  mat=YHT(j,z);
                                                aux2+=B2DE(z,i)*YHT(j,z);
                                    }
                                    {
                                                DT(i,j)=aux2;
//                                              printf("%8.4f",DT(i,j));
                                                aux2=0;
                                    }
                        }
//                      printf("\n");
            }
//ttt=getche();

//particao de D para determinar Z

d=TM(p*blo);
dl=TM(p*blo);

//clrscr();
//printf("VALORES DE D\n");
 for(i=0;i<p;i++)
            {
            for(j=0;j<blo;j++)
                        {
                         ant=DT(j,i);           //apenas para testar o D
                         D(i,j)=DT(j,i);
                         mat=D(i,j);
                        }
             }

free(bde);
/*
clrscr();
printf("VALORES DE D(em ZHTf)\n");
for(i=0;i<p;i++)
```

```
                {
 for(j=0;j<blo;j++)
                printf("%8.4f",D(i,j));
                printf("\n");
                }
ttt=getche();
*/

/*
//clrscr();
//printf("\nVALORES DE DL\n");
 for(i=0;i<p;i++)
                        {
                for(j=0;j<blo;j++)
                {
                DL(i,j)=D(i,j);
//              printf("%10.4f",DL(i,j));
                }
//              printf("\n");
                        }
//              ttt=getche();
*/
free(dl);free(de);

zh=TM(p*p);
d1=TM(blo*blo);
d3=TM(blo*blo);
u2=TM(blo*blo);

*****************************************************************************************************************************
ESTA SUBROTINA SEPARA OS VALORES DE UMA MATRIZ EM QUATRO PARTES SENDO QUE A SUPERIOR
ESQUERDA É' A MAIOR MATRIZ QUADRADA POSSÍVEL.

gaussy(p,blo,&pdf,&rdf,d,d1,d3,u2.zh);
*****************************************************************************************************************************
pd=*pdf;

//clrscr();
//printf("VALORES DE ZHT EM ZHTf\n");
for(i=0;i<pd;i++)
                {
                for(j=0;j<p;j++)
                 {
                 ZHTg(i,j)=ZHT(i,j);
                 mat=ZHT(i,j);
//               printf("%10.6f",ZHTg(i,j));
                 }
//               printf("\n");
                 }
//ttt=getche();
mat=0;
}

#############################################################################################################################
GAUSSY.C
#############################################################################################################################

#include "\"...\"\subs1.h"

#define P(i,j) (*(pp+n3*(i)+j))
#define P1_(i,j) (*(p1_+n3*(i)+j))
#define YHT(i,j) (*(yh+m*(i)+j))
#define D(i,j) (*(d+blo*(i)+j))
#define D1(i,j) (*(d1+blo*(i)+j))
#define D3(i,j) (*(d3+blo*(i)+j))
#define U2(i,j) (*(u2+blo*(i)+j))
#define ZHT(i,j) (*(zh+p*(i)+j))

#define PL(i,j) (*(pl+n3*(i)+j))
#define P1(i,j) (*(p1+n3*(i)+j))
#define P1aux(i,j) (*(p1a+n3*(i)+j))
#define PUAUX(i,j) (*(puaux+rp*(i)+j))
#define P2(i,j) (*(p2+rp*(i)+j))
#define P3(i,j) (*(p3+n3*(i)+j))
#define U(i,j)  (*(u+n3*(i)+j))
#define UP(i,j) (*(up+m*(i)+j))
```

```
#define YT(i,j) (*(y+m*(i)+j))
#define AY(i,j) (*(ay+m*(i)+j))
#define P2U(i,j) (*(pu+rp*(i)+j))
#define H(i,j) (*(h+m*(i)+j))
#define P1f(i,j) (*(p1f+rf*(i)+j))
#define P2f(i,j) (*(p2f+rf*(i)+j))
#define Uf(i,j) (*(uf+rf*(i)+j))
#define P2Uf(i,j) (*(puf+rf*(i)+j))


#define TM(size)  (float *)malloc(size*sizeof(float))
#define TMI(size) (int *)malloc(size*sizeof(int))

void g_vbacks(int rf,int rs,int *irf,float *p1f,float *uf);
void g_prod_m(int s,int rf,int w,float *p2f,float *uf,float *puf);
int  g_lu(int rf, int *irf, float *p1f);

void gaussy(m,n3,pf,rpf,pp,p1,p3,u,yh)

float *pp,*yh,*p1,*p3,*u;
int **rpf,**pf,m,n3;

{
float *h,*p1a,*pl,*p2,*up,*y,*ay,*pu,*puaux;
float *p1f,*uf,*p2f,*puf;
float ant,mat,aux,a2,a1;
int *ic,*ip,*irp,*irf,rf,r,rs,s,w,m1,sp1,sp2,pt,rp;
double fabs();
char tt;
{

register i,j,k,sr,z,t;


rf=0;rp=0;rs=0;s=0;w=0;m1=0;//pt=0;*pf=NULL;*rpf=NULL;h=NULL; *h=0;pl=NULL; *pl=0;p2=NULL; *p2=0;
up=NULL; *up=0;y=NULL;  *y=0;pu=NULL;*pu=0;p1f=NULL; *p1f=0;p1a=NULL;*p1a=0;uf=NULL;  *uf=0;
p2f=NULL;*p2f=0;puf=NULL;*puf=0;ic=NULL; *ic=0;ip=NULL; *ip=0;irp=NULL; *irp=0;irf=NULL;*irf=0;

//clrscr();
//printf("\nVALORES DA MATRIZ A SER SEPARADA(P/D)\n");
pl=TM(m*n3);

 for(i=0;i<m;i++)
            {
            for(j=0;j<n3;j++)
                        {
                        ant=P(i,j);
                        PL(i,j)=P(i,j);
//                      printf("%8.4f",PL(i,j));
                        }
//                      printf("\n");
            }
//tt=getche();

 ic=TMI(n3);
 ip=TMI(m);
 for(i=0; i<m; i++)  ip[i]=i;
 for(i=0; i<n3; i++)  ic[i]=i;

 h=TM(m*m);

 for(i=0;i<m;i++)
            for(j=0;j<m;j++)
           H(i,j)=(i==j ? 1 : 0);

for(i=0;i<n3;i++)
            {                                   //abre if(i)
            for(j=(k=i)+1;j<m;j++)          //aqui j e' indice de linha
                        {
                        ant=PL(j,i); mat=PL(k,i);
                        ant-=mat;
                        if(fabs(PL(j,i)) > fabs(PL(k,i)))
                                    k=j;
                                    }
                         if(k!=i)
                                    for(j=i;j<n3;j++)
```

```
                                        {
                                        a1=PL(i,j); PL(i,j)=PL(k,j); PL(k,j)=a1;
                                        }

/* VERIFICA SE NAS COLUNAS SEGUINTES HÁ ELEMENTO NÃO-ZERO E TROCA AS LINHAS */

a2=PL(i,i);
t=0;

 if(a2==0 && t==0)
                for(j=i+1;j<n3;j++)
                                for(sr=i;sr<m;sr++)
                if(PL(sr,j)!=0 && t==0)
                                for(j=0;j<n3;j++)
                                 {
                                 a1=PL(i,j); PL(i,j)=PL(sr,j); PL(sr,j)=a1;
                                 t=1;
                                 k=sr;
                                 }

/* VERIFICA SE HÁ ELEMENTOS NÃO-ZERO E TROCA AS COLUNAS */

a2 = PL(i,i) ;
t=0;
 if(a2==0 && t==0)
                for(sr=i+1;sr<n3;sr++)
                                 if(PL(i,sr)!=0 && t==0)
                for(j=0;j<m;j++)
                                 {
                                  a1=PL(j,i); PL(j,i)=PL(j,sr); PL(j,sr)=a1;
                                  t=1;
                                  ic[i]=sr;
                                 }

 a2=PL(i,i);
 ip[i]=k;// ip[k]=i;
 for (j=i+1;(j<m && a2!=0);j++)
                {
                 mat=PL(j,i); ant=a2;
                 a1= PL(j,i)/=-a2 ; PL(j,i)=0;//  )/=-a2
                 for(k=i+1; k<n3 ;k++)
                                 {
                                 ant=PL(j,k); mat=a1;
                                 PL(j,k)+=a1*PL(i,k);
                                 mat=PL(j,k);
                                 }
                }
 } //fecha o if(i)

/* MUDAS AS LINHAS DE P */

 m1=m-1;
 for(ip[m1]=m1,i=0;i<m1;i++)
                for(j=0;j<n3;j++)
                                {
                                a1=P(i,j); P(i,j)= P(ip[i],j); P(ip[i],j)=a1;
                                }

/* MUDA AS COLUNAS DE P */

 for(j=0;j<n3-1;j++)
                if(j<ic[j])
                                 for(i=0;i<m;i++)
                {
                                 a1=P(i,j); P(i,j)=P(i,ic[j]); P(i,ic[j])=a1;
                }

/*
clrscr();
printf("VALORES DE P DEPOIS DE TROCADAS AS LINHAS E COLUNAS\n");
for(i=0;i<m;i++)
 {
                for(j=0;j<n3;j++)
                                printf("%8.4f",P(i,j));
                                printf("\n");
                }
```

```
tt=getche();
*/

/* PARTICIONA P */

for(a1=0,rp=0,i=0;(i<n3 && i<m);i++)
                if(fabs(PL(i,i)) > 0.0001) rp++;

 pt=(m-rp);
 *pf=&pt;
 *rpf=&rp;
 //p1=TM(rp*rp);
 p2=TM(pt*rp);

 irp=TMI(rp);
 //u=TM(rp*rp);

free(pl);
free(ic);

 for(i=0;i<rp;i++)
                {
                irp[i]=i;
                mat=irp[i];
                }

sp1=0;
sp2=0;

for(i=0;i<rp;i++)
 {
 for(j=0;j<rp;j++)
                {
                P1(i,j)=P(i,j);
                if(P1(i,j)!=0) sp1++;
                }
 }
 for(i=0;i<pt;i++)
                {
                for(j=0;j<rp;j++)
               {
                if(P(i+rp,j))
                {
                P2(i,j)=-P(i+rp,j);
                if(P2(i,j)!=0) sp2++;
                }
                else

                P2(i,j)=0;
                }
                }

 p1a=TM(rp*rp);

 //clrscr();

 //printf("VALORES DE P1/D1(em gaussy).....1\n");

 for(i=0;i<rp;i++)
                {
                for(j=0;j<rp;j++)
                 {
                 P1aux(i,j)=P1(i,j);
//              printf("%8.4f",P1(i,j));
                 }
//              printf("\n");
                }

 //tt=getche();
 /*
 clrscr();
 printf("VALORES DE P2\n");
 for(i=0;i<pt;i++)
                {
                for(j=0;j<rp;j++)
                                printf("%8.4f",P2(i,j));
```

```
                        printf("\n");
                }
tt=getche();
*/

//printf("VALORES DE P3/D3\n");
for(i=0;i<rp;i++)
 {
 for(j=rp;j<n3;j++)
                {
                P3(i,j-rp)=P(i,j);
//              printf("%8.4f",P3(i,j-rp));
                }
//              printf("\n");
 }
//tt=getche();

//free(pp);

 for(j=0;j<rp;j++)
                for(z=0;z<rp;z++) U(j,z)=(j==z ? 1 : 0);

if((sp1!=0) && (sp2!=0))
{//abre o if(&&)

pu=TM(pt*rp);

*****************************************************************************************************************************************
 g_lu(rp,irp,p1);
 g_vbacks(rp,rp,irp,p1,u);
 g_prod_m(pt,rp,rp,p2,u,pu);
*****************************************************************************************************************************************

 /* PROCEDIMENTO PARA EVITAR ENTRADA DE RESÍDUOS NAS MATRIZES */

 for(i=0;i<rp;i++)
                for(j=0;j<rp;j++)
                  if(fabs(U(i,j))<0.0001)
                                U(i,j)=0;
 /*
 clrscr();
 printf("VALORES DE P1/D1(em gaussy).....2\n");
 for(i=0;i<rp;i++)
                {
                for(j=0;j<rp;j++)
                  printf("%8.4f",P1aux(i,j));
                  printf("\n");
                }
 */
 /*
 clrscr();
 printf("VALORES DE P2P1_(atraves de PROD_M)\n");
 for(i=0;i<pt;i++)
                {
                for(j=0;j<rp;j++)
                                printf("%8.4f",P2U(i,j));
                                printf("\n");
                 }
 tt=getche();
 */
 puaux=TM(pt*rp);
 //clrscr();
 //printf("VALORES DE P2P1_(direto)\n");
 for(aux=0,i=0;i<pt;i++)
                {
                for(j=0;j<rp;j++)
                                {
                                for(z=0;z<rp;z++)
                                 {
                                 ant=P2(i,z);  mat=U(z,j);
                                 aux+=P2(i,z)*U(z,j);
                                 }
                                 PUAUX(i,j)=aux;
                                 mat=PUAUX(i,j);
 //                               printf("%8.4f",PUAUX(i,j));
                                 aux=0.0;
```

```
                        }
//                      printf("\n");
            }
//tt=getche();

free(puaux);

for(i=0;i<rp;i++)
            for(j=0;j<rp;j++)
                        {
                        P1(i,j)=P1aux(i,j);
                        }

/*
printf("\nVALORES DE P1/D1inversa(em gaussy)\n");
for(i=0;i<rp;i++)
            {
            for(j=0;j<rp;j++)
                        printf("%8.4f",U(i,j));
                        printf("\n");
            }
tt=getchar();
*/

/* COMPLETA AS COLUNAS */

up=TM(m*m);  // up É ptxpt MAS AQUI ESTÁ ESTENDIDO
y=TM(m*m);

 for( i=0; i<m ; i++)
            {
            for (j=0; j<m ; j++)
                        {
                        UP(i,j)=(i==j ? 1 : 0 );
//                      printf("%6.2f",UP(i,j));
                        }
//                      printf("\n");
             }
//tt=getche();

//clrscr();
//printf("VALORES DE YT com P2U\n");
 for (i=0; i<pt ;i++)
             {
            for(j=0; j<rp; j++)
                        {
                        YT(i,j)=P2U(i,j);
//                      printf("%8.4f",YT(i,j));
                        }
//                      printf("\n");
                        }
//tt=getche();

 for (i=0; i<pt; i++)
            for (j=rp; j<m; j++)
                        {
                        ant=UP(i,j-rp);
                        YT(i,j)=UP(i,j-rp);
                        mat=YT(i,j);
                        }
/*
clrscr();
printf("\YT\n");
for(i=0;i<pt;i++)
                        {
            for(j=0;j<m;j++)
                        printf("%8.4f",YT(i,j));
                        printf("\n");
                        tt=getche();
                        }
tt=getche();
*/

free(up);
free(pu);
```

```
// TROCA AS LINHAS DE UMA MATRIZ mxm DE ACORDO COM O PONTEIRO PARA AS LINHAS

for(ip[m1]=m1,i=0;i<m1;i++)
            for(j=0;j<m;j++)
                        {
                        a1=H(i,j); H(i,j)=H(ip[i],j); H(ip[i],j)=a1;
                        }
/*
clrscr();
printf("VALORES DE H/ip[]\n");
for(i=0;i<m;i++)
            {
            for(j=0;j<m;j++)
              printf("%2.0f",H(i,j));printf("  ");
              printf("ip[%d]=%d",i,ip[i]);
              printf("\n");
             }
tt=getche();

*/

*****************************************************************************************************************************
g_prod_m(pt,m,m,y,h,yh);
*****************************************************************************************************************************

free(h);free(y);free(p1a);free(p2);

}//fecha o if(&&)
/*
else
{
for(i=0;i<pt;i++)
  for(j=0;j<pt;j++)
            YHT(i,j)=(i==j ? 1 : 0);
}//fecha o else
*/
/*
clrscr();
printf("\nVALORES DE YHT/ZHT(gaussy)\n");
for(i=0;i<pt;i++)
  {
            for(j=0;j<m;j++)
              printf("%10.6f",YHT(i,j));
              printf("\n");
              tt=getche();
  }
tt=getche();
*/
/*
clrscr();
printf("TESTE COM OS VALORES DA MATRIZ SEPARADA\n");
aux=0;
for(i=0;i<n3;i++)
  for(j=0;j<pt;j++)
            for(z=0;z<m;z++)
                        {
                                    aux+=Pol(z,i)*YHT(j,z);
                        }
/*
printf("PRODUTO MAT(t)*PROJ=%10.4f",aux);
if(fabs(aux)>0.001)
printf("\n\nA PROJECAO ESTA ERRADA\n");
tt=getche();
free(pol);
*/
//w=s+rs;
//mat+=w*(w*ant)-rf;
}//fecha o main

} //FECHA A SUBROTINA



##############################COMEÇO DAS SUBROTINAS INTERNAS AO GAUSSY#############################
```

```
int g_lu(int rf,int *irf,float *p1f)
{
 int r1 = rf-1;
 register  int i,j,k;
 float a1,a2;
 double fabs();

 for (irf[r1]=r1,  i=0  ; i<r1 ; i++) {
            for ( j = ( k=i)  +1 ; j<rf ; j++)
                        if (fabs( P1f(j,i)) > fabs( P1f(k,i))) k=j;
           if ( k != i ) for ( j=i ; j<rf ; j++) {
                        a1= P1f(i,j) ; P1f(i,j) = P1f(k,j) ; P1f(k,j) = a1 ;
                        }
 if ( !(a2 = P1f(i,i), a2 )) return (-1):

 for( irf[i]=k, j=i+1; j<rf; j++)
             {
             a1=  ( P1f(j,i) /= -a2 );
             for ( k = i+1 ; k<rf ; k++) P1f(j,k) += a1 * P1f(i,k) ;
             }
 }
           if( !P1f(r1,r1) ) return (-1) ;
 return(0);
}

void g_prod_m(int s,int rf,int w,float *p2f,float *uf,float *puf)
{
 register int i,j,z;
 float a1,mat,ant;

 for(a1=0,i=0; i<s; i++)
            for(z=0;z<w;z++) {
                        for(j=0;j<rf;j++)

                         {
                                    ant=P2f(i,j); mat=Uf(j,z);
                                    a1 += P2f(i,j)*Uf(j,z) ;
                         }
            P2Uf(i,z)=a1;
            mat=a1;
            a1=0;
            }       mat+=ant;
}

 void g_vbacks (int rf ,int rs, int *irf, float *p1f, float *uf )
 {
  register int i, j, k;  int r1 = rf- 1;  float a1,mat,ant;
  for ( i = 0; i < r1; i++ ) {
                                    if ( ( j = irf[i] ) != i ) for ( k = 0; k < rs; k++ ){
                                                a1 = Uf(i,k); Uf(i,k) = Uf(j,k); Uf(j,k) = a1;
                                                }
                                    for ( j = i + 1; j < rf; j++ )
                                                for ( k = 0; k < rs; k++ ) Uf(j,k) += P1f(j,i) * Uf(i,k);
                         }
                                     ant=P1f(r1,r1);
                                    for ( k = 0; k < rs; k++ ) Uf(r1,k) /= P1f(r1,r1);
                                                for ( i = r1 - 1; i >= 0; i-- ) {
                                    for ( a1 = P1f(i,i), j = i + 1; j < rf; j++ )
                                                for ( k = 0; k < rs; k++ ) Uf(i,k) -= P1f(i,j) * Uf(j,k);
                                    for ( k = 0; k < rs; k++ )

                                                {

                                                            Uf(i,k) /= a1;
 mat=Uf(i,k);
 }
                         }mat+=mat;ant+=ant;
             }

 // FIM DAS SUBROTINAS INTERNAS AO GAUSSY

 //FIM DA SUBROTINA GAUSSY


 ###############################################################################
 NR.H - NUMERICAL(1986)
```

```
##############################################################################################

/*  SUBROTINA AUXILIAR AO SVDCMP */

#ifndef _NR_H_
#define _NR_H_

#ifndef _FCOMPLEX_DECLARE_T_
typedef struct FCOMPLEX {float r,i;} fcomplex;
#define _FCOMPLEX_DECLARE_T_
#endif /* _FCOMPLEX_DECLARE_T_ */

#ifndef _ARITHCODE_DECLARE_T_
typedef struct {
        unsigned long *ilob,*iupb,*ncumfq,jdif,nc,minint,nch,ncum,nrad;
} arithcode;
#define _ARITHCODE_DECLARE_T_
#endif /* _ARITHCODE_DECLARE_T_ */

#ifndef _HUFFCODE_DECLARE_T_
typedef struct {
        unsigned long *icod,*ncod,*left,*right,nch,nodemax;
} huffcode;
#define _HUFFCODE_DECLARE_T_
#endif /* _HUFFCODE_DECLARE_T_ */

#include <stdio.h>

#if defined(__STDC__) || defined(ANSI) || defined(NRANSI) /* ANSI */

void addint(double **uf, double **uc, double **res, int nf);
void airy(float x, float *ai, float *bi, float *aip, float *bip);
void amebsa(float **p, float y[], int ndim, float pb[],      float *yb,
        float ftol, float (*funk)(float []), int *iter, float temptr);
void amoeba(float **p, float y[], int ndim, float ftol,
        float (*funk)(float []), int *iter);
float amotry(float **p, float y[], float psum[], int ndim,
        float (*funk)(float []), int ihi, float fac);
float amotsa(float **p, float y[], float psum[], int ndim, float pb[],
        float *yb, float (*funk)(float []), int ihi, float *yhi, float fac);
void anneal(float x[], float y[], int iorder[], int ncity);
double anorm2(double **a, int n);
void arcmak(unsigned long nfreq[], unsigned long nchh, unsigned long nradd,
        arithcode *acode);
void arcode(unsigned long *ich, unsigned char **codep, unsigned long *lcode,
        unsigned long *lcd, int isign, arithcode *acode);
void arcsum(unsigned long iin[], unsigned long iout[], unsigned long ja,
        int nwk, unsigned long nrad, unsigned long nc);
void asolve(unsigned long n, double b[], double x[], int itrnsp);
void atimes(unsigned long n, double x[], double r[], int itrnsp);
void avevar(float data[], unsigned long n, float *ave, float *var);
void balanc(float **a, int n);
void banbks(float **a, unsigned long n, int m1, int m2, float **al,
        unsigned long indx[], float b[]);
void bandec(float **a, unsigned long n, int m1, int m2, float **al,
        unsigned long indx[], float *d);
void banmul(float **a, unsigned long n, int m1, int m2, float x[], float b[]);
void bcucof(float y[], float y1[], float y2[], float y12[], float d1,
        float d2, float **c);
void bcuint(float y[], float y1[], float y2[], float y12[],
        float x1l, float x1u, float x2l, float x2u, float x1,
        float x2, float *ansy, float *ansy1, float *ansy2);
void beschb(double x, double *gam1, double *gam2, double *gampl,
        double *gammi);
float bessi(int n, float x);
float bessi0(float x);
float bessi1(float x);
void bessik(float x, float xnu, float *ri, float *rk, float *rip,
        float *rkp);
float bessj(int n, float x);
float bessj0(float x);
float bessj1(float x);
void bessjy(float x, float xnu, float *rj, float *ry, float *rjp,
        float *ryp);
float bessk(int n, float x);
float bessk0(float x);
```

```
float bessk1(float x);
float bessy(int n, float x);
float bessy0(float x);
float bessy1(float x);
float beta(float z, float w);
float betacf(float a, float b, float x);
float betai(float a, float b, float x);
float bico(int n, int k);
void bksub(int ne, int nb, int jf, int k1, int k2, float ***c);
float bnldev(float pp, int n, long *idum);
float brent(float ax, float bx, float cx,
        float (*f)(float), float tol, float *xmin);
void broydn(float x[], int n, int *check,
        void (*vecfunc)(int, float [], float []));
void bsstep(float y[], float dydx[], int nv, float *xx, float htry,
        float eps, float yscal[], float *hdid, float *hnext,
        void (*derivs)(float, float [], float []));
void caldat(long julian, int *mm, int *id, int *iyyy);
void chder(float a, float b, float c[], float cder[], int n);
float chebev(float a, float b, float c[], int m, float x);
void chebft(float a, float b, float c[], int n, float (*func)(float));
void chebpc(float c[], float d[], int n);
void chint(float a, float b, float c[], float cint[], int n);
float chixy(float bang);
void choldc(float **a, int n, float p[]);
void cholsl(float **a, int n, float p[], float b[], float x[]);
void chsone(float bins[], float ebins[], int nbins, int knstrn,
        float *df, float *chsq, float *prob);
void chstwo(float bins1[], float bins2[], int nbins, int knstrn,
        float *df, float *chsq, float *prob);
void cisi(float x, float *ci, float *si);
void cntab1(int **nn, int ni, int nj, float *chisq,
        float *df, float *prob, float *cramrv, float *ccc);
void cntab2(int **nn, int ni, int nj, float *h, float *hx, float *hy,
        float *hygx, float *hxgy, float *uygx, float *uxgy, float *uxy);
void convlv(float data[], unsigned long n, float respns[], unsigned long m,
        int isign, float ans[]);
void copy(double **aout, double **ain, int n);
void correl(float data1[], float data2[], unsigned long n, float ans[]);
void cosft(float y[], int n, int isign);
void cosft1(float y[], int n);
void cosft2(float y[], int n, int isign);
void covsrt(float **covar, int ma, int ia[], int mfit);
void crank(unsigned long n, float w[], float *s);
void cyclic(float a[], float b[], float c[], float alpha, float beta,
        float r[], float x[], unsigned long n);
void daub4(float a[], unsigned long n, int isign);
float dawson(float x);
float dbrent(float ax, float bx, float cx,
        float (*f)(float), float (*df)(float), float tol, float *xmin);
void ddpoly(float c[], int nc, float x, float pd[], int nd);
int decchk(char string[], int n, char *ch);
void derivs(float x, float y[], float dydx[]);
float df1dim(float x);
void dfour1(double data[], unsigned long nn, int isign);
void dfpmin(float p[], int n, float gtol, int *iter, float *fret,
        float (*func)(float []), void (*dfunc)(float [], float []));
float dfridr(float (*func)(float), float x, float h, float *err);
void dftcor(float w, float delta, float a, float b, float endpts[],
        float *corre, float *corim, float *corfac);
void dftint(float (*func)(float), float a, float b, float w,
        float *cosint, float *sinint);
void difeq(int k, int k1, int k2, int jsf, int is1, int isf,
        int indexv[], int ne, float **s, float **y);
void dlinmin(float p[], float xi[], int n, float *fret,
        float (*func)(float []), void (*dfunc)(float [], float[]));
double dpythag(double a, double b);
void drealft(double data[], unsigned long n, int isign);
void dsprsax(double sa[], unsigned long ija[], double x[], double b[],
        unsigned long n);
void dsprstx(double sa[], unsigned long ija[], double x[], double b[],
        unsigned long n);
void dsvbksb(double **u, double w[], double **v, int m, int n, double b[],
        double x[]);
void dsvdcmp(double **a, int m, int n, double w[], double **v);
void eclass(int nf[], int n, int lista[], int listb[], int m);
```

```
void eclazz(int nf[], int n, int (*equiv)(int, int));
float ei(float x);
void eigsrt(float d[], float **v, int n);
float elle(float phi, float ak);
float ellf(float phi, float ak);
float ellpi(float phi, float en, float ak);
void elmhes(float **a, int n);
float erfcc(float x);
float erff(float x);
float erffc(float x);
void eulsum(float *sum, float term, int jterm, float wksp[]);
float evlmem(float fdt, float d[], int m, float xms);
float expdev(long *idum);
float expint(int n, float x);
float f1(float x);
float f1dim(float x);
float f2(float y);
float f3(float z);
float factln(int n);
float factrl(int n);
void fasper(float x[], float y[], unsigned long n, float ofac, float hifac,
        float wk1[], float wk2[], unsigned long nwk, unsigned long *nout,
        unsigned long *jmax, float *prob);
void fdjac(int n, float x[], float fvec[], float **df,
        void (*vecfunc)(int, float [], float []));
void fgauss(float x, float a[], float *y, float dyda[], int na);
void fill0(double **u, int n);
void fit(float x[], float y[], int ndata, float sig[], int mwt,
        float *a, float *b, float *siga, float *sigb, float *chi2, float *q);
void fitexy(float x[], float y[], int ndat, float sigx[], float sigy[],
        float *a, float *b, float *siga, float *sigb, float *chi2, float *q);
void fixrts(float d[], int m);
void fleg(float x, float pl[], int nl);
void flmoon(int n, int nph, long *jd, float *frac);
float fmin(float x[]);
void four1(float data[], unsigned long nn, int isign);
void fourew(FILE *file[5], int *na, int *nb, int *nc, int *nd);
void fourfs(FILE *file[5], unsigned long nn[], int ndim, int isign);
void fourn(float data[], unsigned long nn[], int ndim, int isign);
void fpoly(float x, float p[], int np);
void fred2(int n, float a, float b, float t[], float f[], float w[],
        float (*g)(float), float (*ak)(float, float));
float fredin(float x, int n, float a, float b, float t[], float f[], float w[],
        float (*g)(float), float (*ak)(float, float));
void frenel(float x, float *s, float *c);
void frprmn(float p[], int n, float ftol, int *iter, float *fret,
        float (*func)(float []), void (*dfunc)(float [], float []));
void ftest(float data1[], unsigned long n1, float data2[], unsigned long n2,
        float *f, float *prob);
float gamdev(int ia, long *idum);
float gammln(float xx);
float gammp(float a, float x);
float gammq(float a, float x);
float gasdev(long *idum);
void gaucof(int n, float a[], float b[], float amu0, float x[], float w[]);
void gauher(float x[], float w[], int n);
void gaujac(float x[], float w[], int n, float alf, float bet);
void gaulag(float x[], float w[], int n, float alf);
void gauleg(float x1, float x2, float x[], float w[], int n);
void gaussj(float **a, int n, float **b, int m);
void gcf(float *gammcf, float a, float x, float *gln);
float golden(float ax, float bx, float cx, float (*f)(float), float tol,
        float *xmin);
void gser(float *gamser, float a, float x, float *gln);
void hpsel(unsigned long m, unsigned long n, float arr[], float heap[]);
void hpsort(unsigned long n, float ra[]);
void hqr(float **a, int n, float wr[], float wi[]);
void hufapp(unsigned long index[], unsigned long nprob[], unsigned long n,
        unsigned long i);
void hufdec(unsigned long *ich, unsigned char *code, unsigned long lcode,
        unsigned long *nb, huffcode *hcode);
void hufenc(unsigned long ich, unsigned char **codep, unsigned long *lcode,
        unsigned long *nb, huffcode *hcode);
void hufmak(unsigned long nfreq[], unsigned long nchin, unsigned long *ilong,
        unsigned long *nlong, huffcode *hcode);
void hunt(float xx[], unsigned long n, float x, unsigned long *jlo);
```

```
void hypdrv(float s, float yy[], float dyyds[]);
fcomplex hypgeo(fcomplex a, fcomplex b, fcomplex c, fcomplex z);
void hypser(fcomplex a, fcomplex b, fcomplex c, fcomplex z,
        fcomplex *series, fcomplex *deriv);
unsigned short icrc(unsigned short crc, unsigned char *bufptr,
        unsigned long len, short jinit, int jrev);
unsigned short icrc1(unsigned short crc, unsigned char onech);
unsigned long igray(unsigned long n, int is);
void iindexx(unsigned long n, long arr[], unsigned long indx[]);
void indexx(unsigned long n, float arr[], unsigned long indx[]);
void interp(double **uf, double **uc, int nf);
int irbit1(unsigned long *iseed);
int irbit2(unsigned long *iseed);
void jacobi(float **a, int n, float d[], float **v, int *nrot);
void jacobn(float x, float y[], float dfdx[], float **dfdy, int n);
long julday(int mm, int id, int iyyy);
void kendl1(float data1[], float data2[], unsigned long n, float *tau, float *z,
        float *prob);
void kendl2(float **tab, int i, int j, float *tau, float *z, float *prob);
void kermom(double w[], double y, int m);
void ks2d1s(float x1[], float y1[], unsigned long n1,
        void (*quadvl)(float, float, float *, float *, float *, float *),
        float *d1, float *prob);
void ks2d2s(float x1[], float y1[], unsigned long n1, float x2[], float y2[],
        unsigned long n2, float *d, float *prob);
void ksone(float data[], unsigned long n, float (*func)(float), float *d,
        float *prob);
void kstwo(float data1[], unsigned long n1, float data2[], unsigned long n2,
        float *d, float *prob);
void laguer(fcomplex a[], int m, fcomplex *x, int *its);
void lfit(float x[], float y[], float sig[], int ndat, float a[], int ia[],
        int ma, float **covar, float *chisq, void (*funcs)(float, float [], int));
void linbcg(unsigned long n, double b[], double x[], int itol, double tol,
        int itmax, int *iter, double *err);
void linmin(float p[], float xi[], int n, float *fret,
        float (*func)(float []));
void lnsrch(int n, float xold[], float fold, float g[], float p[], float x[],
        float *f, float stpmax, int *check, float (*func)(float []));
void load(float x1, float v[], float y[]);
void load1(float x1, float v1[], float y[]);
void load2(float x2, float v2[], float y[]);
void locate(float xx[], unsigned long n, float x, unsigned long *j);
void lop(double **out, double **u, int n);
void lubksb(float **a, int n, int *indx, float b[]);
void ludcmp(float **a, int n, int *indx, float *d);
void machar(int *ibeta, int *it, int *irnd, int *ngrd,
        int *machep, int *negep, int *iexp, int *minexp, int *maxexp,
        float *eps, float *epsneg, float *xmin, float *xmax);
void matadd(double **a, double **b, double **c, int n);
void matsub(double **a, double **b, double **c, int n);
void medfit(float x[], float y[], int ndata, float *a, float *b, float *abdev);
void memcof(float data[], int n, int m, float *xms, float d[]);
int metrop(float de, float t);
void mgfas(double **u, int n, int maxcyc);
void mglin(double **u, int n, int ncycle);
float midexp(float (*funk)(float), float aa, float bb, int n);
float midinf(float (*funk)(float), float aa, float bb, int n);
float midpnt(float (*func)(float), float a, float b, int n);
float midsql(float (*funk)(float), float aa, float bb, int n);
float midsqu(float (*funk)(float), float aa, float bb, int n);
void miser(float (*func)(float []), float regn[], int ndim, unsigned long npts,
        float dith, float *ave, float *var);
void mmid(float y[], float dydx[], int nvar, float xs, float htot,
        int nstep, float yout[], void (*derivs)(float, float[], float[]));
void mnbrak(float *ax, float *bx, float *cx, float *fa, float *fb,
        float *fc, float (*func)(float));
void mnewt(int ntrial, float x[], int n, float tolx, float tolf);
void moment(float data[], int n, float *ave, float *adev, float *sdev,
        float *var, float *skew, float *curt);
void mp2dfr(unsigned char a[], unsigned char s[], int n, int *m);
void mpadd(unsigned char w[], unsigned char u[], unsigned char v[], int n);
void mpdiv(unsigned char q[], unsigned char r[], unsigned char u[],
        unsigned char v[], int n, int m);
void mpinv(unsigned char u[], unsigned char v[], int n, int m);
void mplsh(unsigned char u[], int n);
void mpmov(unsigned char u[], unsigned char v[], int n);
```

```
void mpmul(unsigned char w[], unsigned char u[], unsigned char v[], int n,
        int m);
void mpneg(unsigned char u[], int n);
void mppi(int n);
void mprove(float **a, float **alud, int n, int indx[], float b[],
        float x[]);
void mpsad(unsigned char w[], unsigned char u[], int n, int iv);
void mpsdv(unsigned char w[], unsigned char u[], int n, int iv, int *ir);
void mpsmu(unsigned char w[], unsigned char u[], int n, int iv);
void mpsqrt(unsigned char w[], unsigned char u[], unsigned char v[], int n,
        int m);
void mpsub(int *is, unsigned char w[], unsigned char u[], unsigned char v[],
        int n);
void mrqcof(float x[], float y[], float sig[], int ndata, float a[],
        int ia[], int ma, float **alpha, float beta[], float *chisq,
        void (*funcs)(float, float [], float *, float [], int));
void mrqmin(float x[], float y[], float sig[], int ndata, float a[],
        int ia[], int ma, float **covar, float **alpha, float *chisq,
        void (*funcs)(float, float [], float *, float [], int), float *alamda);
void newt(float x[], int n, int *check,
        void (*vecfunc)(int, float [], float []));
void odeint(float ystart[], int nvar, float x1, float x2,
        float eps, float h1, float hmin, int *nok, int *nbad,
        void (*derivs)(float, float [], float []),
        void (*rkqs)(float [], float [], int, float *, float, float,
        float [], float *, float *, void (*)(float, float [], float [])));
void orthog(int n, float anu[], float alpha[], float beta[], float a[],
        float b[]);
void pade(double cof[], int n, float *resid);
void pccheb(float d[], float c[], int n);
void pcshft(float a, float b, float d[], int n);
void pearsn(float x[], float y[], unsigned long n, float *r, float *prob,
        float *z);
void period(float x[], float y[], int n, float ofac, float hifac,
        float px[], float py[], int np, int *nout, int *jmax, float *prob);
void piksr2(int n, float arr[], float brr[]);
void piksrt(int n, float arr[]);
void pinvs(int ie1, int ie2, int je1, int jsf, int jc1, int k,
        float ***c, float **s);
float plgndr(int l, int m, float x);
float poidev(float xm, long *idum);
void polcoe(float x[], float y[], int n, float cof[]);
void polcof(float xa[], float ya[], int n, float cof[]);
void poldiv(float u[], int n, float v[], int nv, float q[], float r[]);
void polin2(float x1a[], float x2a[], float **ya, int m, int n,
        float x1, float x2, float *y, float *dy);
void polint(float xa[], float ya[], int n, float x, float *y, float *dy);
void powell(float p[], float **xi, int n, float ftol, int *iter, float *fret,
        float (*func)(float []));
void predic(float data[], int ndata, float d[], int m, float future[], int nfut);
float probks(float alam);
void psdes(unsigned long *lword, unsigned long *irword);
void pwt(float a[], unsigned long n, int isign);
void pwtset(int n);
float pythag(float a, float b);
void pzextr(int iest, float xest, float yest[], float yz[], float dy[],
        int nv);
float qgaus(float (*func)(float), float a, float b);
void qrdcmp(float **a, int n, float *c, float *d, int *sing);
float qromb(float (*func)(float), float a, float b);
float qromo(float (*func)(float), float a, float b,
        float (*choose)(float (*)(float), float, float, int));
void qroot(float p[], int n, float *b, float *c, float eps);
void qrsolv(float **a, int n, float c[], float d[], float b[]);
void qrupdt(float **r, float **qt, int n, float u[], float v[]);
float qsimp(float (*func)(float), float a, float b);
float qtrap(float (*func)(float), float a, float b);
float quad3d(float (*func)(float, float, float), float x1, float x2);
void quadct(float x, float y, float xx[], float yy[], unsigned long nn,
        float *fa, float *fb, float *fc, float *fd);
void quadmx(float **a, int n);
void quadvl(float x, float y, float *fa, float *fb, float *fc, float *fd);
float ran0(long *idum);
float ran1(long *idum);
float ran2(long *idum);
float ran3(long *idum);
```

```
float ran4(long *idum);
void rank(unsigned long n, unsigned long indx[], unsigned long irank[]);
void ranpt(float pt[], float regn[], int n);
void ratint(float xa[], float ya[], int n, float x, float *y, float *dy);
void ratlsq(double (*fn)(double), double a, double b, int mm, int kk,
        double cof[], double *dev);
double ratval(double x, double cof[], int mm, int kk);
float rc(float x, float y);
float rd(float x, float y, float z);
void realft(float data[], unsigned long n, int isign);
void rebin(float rc, int nd, float r[], float xin[], float xi[]);
void red(int iz1, int iz2, int jz1, int jz2, int jm1, int jm2, int jmf,
        int ic1, int jc1, int jcf, int kc, float ***c, float **s);
void relax(double **u, double **rhs, int n);
void relax2(double **u, double **rhs, int n);
void resid(double **res, double **u, double **rhs, int n);
float revcst(float x[], float y[], int iorder[], int ncity, int n[]);
void reverse(int iorder[], int ncity, int n[]);
float rf(float x, float y, float z);
float rj(float x, float y, float z, float p);
void rk4(float y[], float dydx[], int n, float x, float h, float yout[],
        void (*derivs)(float, float [], float []));
void rkck(float y[], float dydx[], int n, float x, float h,
        float yout[], float yerr[], void (*derivs)(float, float [], float []));
void rkdumb(float vstart[], int nvar, float x1, float x2, int nstep,
        void (*derivs)(float, float [], float []));
void rkqs(float y[], float dydx[], int n, float *x,
        float htry, float eps, float yscal[], float *hdid, float *hnext,
        void (*derivs)(float, float [], float []));
void rlft3(float ***data, float **speq, unsigned long nn1,
        unsigned long nn2, unsigned long nn3, int isign);
float rofunc(float b);
void rotate(float **r, float **qt, int n, int i, float a, float b);
void rsolv(float **a, int n, float d[], float b[]);
void rstrct(double **uc, double **uf, int nc);
float rtbis(float (*func)(float), float x1, float x2, float xacc);
float rtflsp(float (*func)(float), float x1, float x2, float xacc);
float rtnewt(void (*funcd)(float, float *, float *), float x1, float x2,
        float xacc);
float rtsafe(void (*funcd)(float, float *, float *), float x1, float x2,
        float xacc);
float rtsec(float (*func)(float), float x1, float x2, float xacc);
void rzextr(int iest, float xest, float yest[], float yz[], float dy[], int nv);
void savgol(float c[], int np, int nl, int nr, int ld, int m);
void score(float xf, float y[], float f[]);
void scrsho(float (*fx)(float));
float select(unsigned long k, unsigned long n, float arr[]);
float selip(unsigned long k, unsigned long n, float arr[]);
void shell(unsigned long n, float a[]);
void shoot(int n, float v[], float f[]);
void shootf(int n, float v[], float f[]);
void simp1(float **a, int mm, int ll[], int nll, int iabf, int *kp,
        float *bmax);
void simp2(float **a, int n, int l2[], int nl2, int *ip, int kp, float *q1);
void simp3(float **a, int i1, int k1, int ip, int kp);
void simplx(float **a, int m, int n, int m1, int m2, int m3, int *icase,
        int izrov[], int iposv[]);
void simpr(float y[], float dydx[], float dfdx[], float **dfdy,
        int n, float xs, float htot, int nstep, float yout[],
        void (*derivs)(float, float [], float []));
void sinft(float y[], int n);
void slvsm2(double **u, double **rhs);
void slvsml(double **u, double **rhs);
void sncndn(float uu, float emmc, float *sn, float *cn, float *dn);
double snrm(unsigned long n, double sx[], int itol);
void sobseq(int *n, float x[]);
void solvde(int itmax, float conv, float slowc, float scalv[],
        int indexv[], int ne, int nb, int m, float **y, float ***c, float **s);
void sor(double **a, double **b, double **c, double **d, double **e,
        double **f, double **u, int jmax, double rjac);
void sort(unsigned long n, float arr[]);
void sort2(unsigned long n, float arr[], float brr[]);
void sort3(unsigned long n, float ra[], float rb[], float rc[]);
void spctrm(FILE *fp, float p[], int m, int k, int ovrlap);
void spear(float data1[], float data2[], unsigned long n, float *d, float *zd,
        float *probd, float *rs, float *probrs);
```

```
void sphbes(int n, float x, float *sj, float *sy, float *sjp, float *syp);
void splie2(float x1a[], float x2a[], float **ya, int m, int n, float **y2a);
void splin2(float x1a[], float x2a[], float **ya, float **y2a, int m, int n,
        float x1, float x2, float *y);
void spline(float x[], float y[], int n, float yp1, float ypn, float y2[]);
void splint(float xa[], float ya[], float y2a[], int n, float x, float *y);
void spread(float y, float yy[], unsigned long n, float x, int m);
void sprsax(float sa[], unsigned long ija[], float x[], float b[],
        unsigned long n);
void sprsin(float **a, int n, float thresh, unsigned long nmax, float sa[],
        unsigned long ija[]);
void sprspm(float sa[], unsigned long ija[], float sb[], unsigned long ijb[],
        float sc[], unsigned long ijc[]);
void sprstm(float sa[], unsigned long ija[], float sb[], unsigned long ijb[],
        float thresh, unsigned long nmax, float sc[], unsigned long ijc[]);
void sprstp(float sa[], unsigned long ija[], float sb[], unsigned long ijb[]);
void sprstx(float sa[], unsigned long ija[], float x[], float b[],
        unsigned long n);
void stifbs(float y[], float dydx[], int nv, float *xx,
        float htry, float eps, float yscal[], float *hdid, float *hnext,
        void (*derivs)(float, float [], float []));
void stiff(float y[], float dydx[], int n, float *x,
        float htry, float eps, float yscal[], float *hdid, float *hnext,
        void (*derivs)(float, float [], float []));
void stoerm(float y[], float d2y[], int nv, float xs,
        float htot, int nstep, float yout[],
        void (*derivs)(float, float [], float []));
void svbksb(float **u, float w[], float **v, int m, int n, float b[],
        float x[]);
void svdcmp(float **a, int m, int n, float w[], float **v);
void svdfit(float x[], float y[], float sig[], int ndata, float a[],
        int ma, float **u, float **v, float w[], float *chisq,
        void (*funcs)(float, float [], int));
void svdvar(float **v, int ma, float w[], float **cvm);
void toeplz(float r[], float x[], float y[], int n);
void tptest(float data1[], float data2[], unsigned long n, float *t, float *prob);
void tqli(float d[], float e[], int n, float **z);
float trapzd(float (*func)(float), float a, float b, int n);
void tred2(float **a, int n, float d[], float e[]);
void tridag(float a[], float b[], float c[], float r[], float u[],
        unsigned long n);
float trncst(float x[], float y[], int iorder[], int ncity, int n[]);
void trnspt(int iorder[], int ncity, int n[]);
void ttest(float data1[], unsigned long n1, float data2[], unsigned long n2,
        float *t, float *prob);
void tutest(float data1[], unsigned long n1, float data2[], unsigned long n2,
        float *t, float *prob);
void twofft(float data1[], float data2[], float fft1[], float fft2[],
        unsigned long n);
void vander(double x[], double w[], double q[], int n);
void vegas(float regn[], int ndim, float (*fxn)(float [], float), int init,
        unsigned long ncall, int itmx, int nprn, float *tgral, float *sd,
        float *chi2a);
void voltra(int n, int m, float t0, float h, float *t, float **f,
        float (*g)(int, float), float (*ak)(int, int, float, float));
void wt1(float a[], unsigned long n, int isign,
        void (*wtstep)(float [], unsigned long, int));
void wtn(float a[], unsigned long nn[], int ndim, int isign,
        void (*wtstep)(float [], unsigned long, int));
void wwghts(float wghts[], int n, float h,
        void (*kermom)(double [], double ,int));
int zbrac(float (*func)(float), float *x1, float *x2);
void zbrak(float (*fx)(float), float x1, float x2, int n, float xb1[],
        float xb2[], int *nb);
float zbrent(float (*func)(float), float x1, float x2, float tol);
void zrhqr(float a[], int m, float rtr[], float rti[]);
float zriddr(float (*func)(float), float x1, float x2, float xacc);
void zroots(fcomplex a[], int m, fcomplex roots[], int polish);

#else /* ANSI */

/* traditional - K&R */


void addint();void airy();void amebsa();void amoeba()float amotry();float amotsa();
```

```
void anneal();double anorm2();void arcmak();void arcode();void arcsum();void asolve();void atimes();void avevar();
void balanc();void banbks();void bandec();void banmul();void bcucof();void bcuint();void beschb();float bessi();float bessi0();
float bessi1();void bessik();float bessj();float bessj0();float bessj1();void bessjy();float bessk();float bessk0();float bessk1();float bessy();
float bessy0();float bessy1();float beta();float betacf();float betai();float bico();void bksub();float bnldev();float brent();void broydn();
void bsstep();void caldat();void chder();float chebev();void chebft();void chebpc();void chint();float chixy();void choldc();void cholsl();
void chsone();void chstwo();void cisi();void cntab1();void cntab2();void convlv();void copy();void correl();void cosft();void cosft1();
void cosft2();void covsrt();void crank();void cyclic();void daub4();float dawson();float dbrent();void ddpoly();int decchk();void derivs();
float df1dim();void dfour1();void dfpmin();float dfridr();void dftcor();void dftint();void difeq();void dlinmin();double dpythag();
void drealft();void dsprsax();void dsprstx();void dsvbksb();void dsvdcmp();void eclass();void eclazz();float ei();void eigsrt();
float elle();float ellf();float ellpi();void elmhes();float erfcc();float erff();float erffc();void eulsum();float evlmem();float expdev();
float expint();float f1();float f1dim();float f2();float f3();float factln();float factrl();void fasper();void fdjac();void fgauss();void fill0();
void fit();void fitexy();void fixrts();void fleg();void flmoon();float fmin();void four1();void fourew();void fourfs();void fourn();
void fpoly();void fred2();float fredin();void frenel();void frprmn();void ftest();float gamdev();float gammln();float gammp();
float gammq();float gasdev();void gaucof();void gauher();void gaujac();void gaulag();void gauleg();void gaussj();void gcf();
float golden();void gser();void hpsel();void hpsort();void hqr();void hufapp();void hufdec();void hufenc();void hufmak();void hunt();
void hypdrv();fcomplex hypgeo();void hypser();unsigned short icrc();unsigned short icrc1();int igray();void iindexx();void indexx();
void interp();int irbit1();int irbit2();void jacobi();void jacobn();long julday();void kendl1();void kendl2();void kermom();void ks2d1s();
void ks2d2s();void ksone();void kstwo();void laguer();void lfit();void linbcg();void linmin();void lnsrch();void load();void load1();
void load2();void locate();void lop();void lubksb();void ludcmp();void machar();void matadd();void matsub();void medfit();
void memcof();int metrop();void mgfas();void mglin();float midexp();float midinf();float midpnt();float midsql();float midsqu();
void miser();void mmid();void mnbrak();void mnewt();void moment();void mp2dfr();void mpadd();void mpdiv();void mpinv();
void mplsh();void mpmov();void mpmul();void mpneg();void mppi();void mprove();void mpsad();void mpsdv();void mpsmu();
void mpsqrt();void mpsub();void mrqcof();void mrqmin();void newt();void odeint();void orthog();void pade();void pccheb();
void pcshft();void pearsn();void period();void piksr2();void piksrt();void pinvs();float plgndr();float poidev();void polcoe();void polcof();
void poldiv();void polin2();void polint();void powell();void predic();float probks();void psdes();void pwt();void pwtset();float pythag();
void pzextr();float qgaus();void qrdcmp();float qromb();float qromo();void qroot();void qrsolv();void qrupdt();float qsimp();float qtrap();
float quad3d();void quadct();void quadmx();void quadvl();float ran0();float ran1();float ran2();float ran3();float ran4();void rank();
void ranpt();void ratint();void ratlsq();double ratval();float rc();float rd();void realft();void rebin();void red();void relax();void relax2();
void resid();float revcst();void reverse();float rf();float rj();void rk4();void rkck();void rkdumb();void rkqs();void rlft3();float rofunc();
void rotate();void rsolv();void rstrct();float rtbis();float rtflsp();float rtnewt();float rtsafe();float rtsec();void rzextr();void savgol();
void score();void scrsho();float select();float selip();void shell();void shoot();void shootf();void simp1();void simp2();void simp3();
void simplx();void simpr();void sinft();void slvsm2();void slvsml();void sncndn();double snrm();
void sobseq();void solvde();void sor();void sort();void sort2();void sort3();void spctrm();void spear();void sphbes();void splie2();
void splin2();void spline();void splint();void spread();void sprsax();void sprsin();void sprspm();void sprstm();void sprstp();void
sprstx();voidstifbs();voidstiff();voidstoerm();void svbksb();void svdcmp();void svdfit();void svdvar();void toeplz();void tptest();void
tqli();float trapzd();void tred2();void tridag();float trncst();void trnspt();void ttest();void tutest();void twofft();void vander();void vegas();
void voltra();void wt1();void wtn();void wwghts();int zbrac();void zbrak();float zbrent();void zrhqr();float zriddr();void zroots();

#endif /* ANSI */

#endif /* _NR_H_ */

###################################################################################################################
NRUTIL.H-NUMERICAL(1986)
###################################################################################################################

/* SUBROTINA AUXILIAR AO SVDCMP */

#if defined(__STDC__) || defined(ANSI) || defined(NRANSI) /* ANSI */

#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#define NR_END 1
#define FREE_ARG char*

void nrerror(char error_text[])
/* Numerical Recipes standard error handler */
{
        fprintf(stderr,"Numerical Recipes run-time error...\n");
        fprintf(stderr,"%s\n",error_text);
        fprintf(stderr,"...now exiting to system...\n");
        exit(1);
}

float *vector(long nl, long nh)
/* allocate a float vector with subscript range v[nl..nh] */
{
        float *v;

        v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
        if (!v) nrerror("allocation failure in vector()");
        return v-nl+NR_END;
}

int *ivector(long nl, long nh)
```

```
/* allocate an int vector with subscript range v[nl..nh] */
{
        int *v;

        v=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
        if (!v) nrerror("allocation failure in ivector()");
        return v-nl+NR_END;
}

unsigned char *cvector(long nl, long nh)
/* allocate an unsigned char vector with subscript range v[nl..nh] */
{
        unsigned char *v;

        v=(unsigned char *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(unsigned char)));
        if (!v) nrerror("allocation failure in cvector()");
        return v-nl+NR_END;
}

unsigned long *lvector(long nl, long nh)
/* allocate an unsigned long vector with subscript range v[nl..nh] */
{
        unsigned long *v;

        v=(unsigned long *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(long)));
        if (!v) nrerror("allocation failure in lvector()");
        return v-nl+NR_END;
}

double *dvector(long nl, long nh)
/* allocate a double vector with subscript range v[nl..nh] */
{
        double *v;

        v=(double *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(double)));
        if (!v) nrerror("allocation failure in dvector()");
        return v-nl+NR_END;
}

float **matrix(long nrl, long nrh, long ncl, long nch)
/* allocate a float matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        float **m;

        /* allocate pointers to rows */
        m=(float **) malloc((size_t)((nrow+NR_END)*sizeof(float*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl]=(float *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(float)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

double **dmatrix(long nrl, long nrh, long ncl, long nch)
/* allocate a double matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        double **m;

        /* allocate pointers to rows */
        m=(double **) malloc((size_t)((nrow+NR_END)*sizeof(double*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;
        /* allocate rows and set pointers to them */
        m[nrl]=(double *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(double)));
```

```
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

int **imatrix(long nrl, long nrh, long ncl, long nch)
/* allocate a int matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        int **m;

        /* allocate pointers to rows */
        m=(int **) malloc((size_t)((nrow+NR_END)*sizeof(int*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl]=(int *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(int)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

float **submatrix(float **a, long oldrl, long oldrh, long oldcl, long oldch,
        long newrl, long newcl)
/* point a submatrix [newrl..][newcl..] to a[oldrl..oldrh][oldcl..oldch] */
{
        long i,j,nrow=oldrh-oldrl+1,ncol=oldcl-newcl;
        float **m;

        /* allocate array of pointers to rows */
        m=(float **) malloc((size_t) ((nrow+NR_END)*sizeof(float*)));
        if (!m) nrerror("allocation failure in submatrix()");
        m += NR_END;
        m -= newrl;

        /* set pointers to rows */
        for(i=oldrl,j=newrl;i<=oldrh;i++,j++) m[j]=a[i]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

float **convert_matrix(float *a, long nrl, long nrh, long ncl, long nch)
/* allocate a float matrix m[nrl..nrh][ncl..nch] that points to the matrix
declared in the standard C manner as a[nrow][ncol], where nrow=nrh-nrl+1
and ncol=nch-ncl+1. The routine should be called with the address
&a[0][0] as the first argument. */
{
        long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1;
        float **m;

        /* allocate pointers to rows */
        m=(float **) malloc((size_t) ((nrow+NR_END)*sizeof(float*)));
        if (!m) nrerror("allocation failure in convert_matrix()");
        m += NR_END;
        m -= nrl;

        /* set pointers to rows */
        m[nrl]=a-ncl;
        for(i=1,j=nrl+1;i<nrow;i++,j++) m[j]=m[j-1]+ncol;
        /* return pointer to array of pointers to rows */
        return m;
}
```

```
float ***f3tensor(long nrl, long nrh, long ncl, long nch, long ndl, long ndh)
/* allocate a float 3tensor with range t[nrl..nrh][ncl..nch][ndl..ndh] */
{
        long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1,ndep=ndh-ndl+1;
        float ***t;

        /* allocate pointers to pointers to rows */
        t=(float ***) malloc((size_t)((nrow+NR_END)*sizeof(float**)));
        if (!t) nrerror("allocation failure 1 in f3tensor()");
        t += NR_END;
        t -= nrl;

        /* allocate pointers to rows and set pointers to them */
        t[nrl]=(float **) malloc((size_t)((nrow*ncol+NR_END)*sizeof(float*)));
        if (!t[nrl]) nrerror("allocation failure 2 in f3tensor()");
        t[nrl] += NR_END;
        t[nrl] -= ncl;

        /* allocate rows and set pointers to them */
        t[nrl][ncl]=(float *) malloc((size_t)((nrow*ncol*ndep+NR_END)*sizeof(float)));
        if (!t[nrl][ncl]) nrerror("allocation failure 3 in f3tensor()");
        t[nrl][ncl] += NR_END;
        t[nrl][ncl] -= ndl;

        for(j=ncl+1;j<=nch;j++) t[nrl][j]=t[nrl][j-1]+ndep;
        for(i=nrl+1;i<=nrh;i++) {
                t[i]=t[i-1]+ncol;
                t[i][ncl]=t[i-1][ncl]+ncol*ndep;
                for(j=ncl+1;j<=nch;j++) t[i][j]=t[i][j-1]+ndep;
        }

        /* return pointer to array of pointers to rows */
        return t;
}

void free_vector(float *v, long nl, long nh)
/* free a float vector allocated with vector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_ivector(int *v, long nl, long nh)
/* free an int vector allocated with ivector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_cvector(unsigned char *v, long nl, long nh)
/* free an unsigned char vector allocated with cvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_lvector(unsigned long *v, long nl, long nh)
/* free an unsigned long vector allocated with lvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_dvector(double *v, long nl, long nh)
/* free a double vector allocated with dvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_matrix(float **m, long nrl, long nrh, long ncl, long nch)
/* free a float matrix allocated by matrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch)
/* free a double matrix allocated by dmatrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
```

```
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch)
/* free an int matrix allocated by imatrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_submatrix(float **b, long nrl, long nrh, long ncl, long nch)
/* free a submatrix allocated by submatrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_convert_matrix(float **b, long nrl, long nrh, long ncl, long nch)
/* free a matrix allocated by convert_matrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_f3tensor(float ***t, long nrl, long nrh, long ncl, long nch,
        long ndl, long ndh)
/* free a float f3tensor allocated by f3tensor() */
{
        free((FREE_ARG) (t[nrl][ncl]+ndl-NR_END));
        free((FREE_ARG) (t[nrl]+ncl-NR_END));
        free((FREE_ARG) (t+nrl-NR_END));
}

#else /* ANSI */
/* traditional - K&R */

#include <stdio.h>
#define NR_END 1
#define FREE_ARG char*

void nrerror(error_text)
char error_text[];
/* Numerical Recipes standard error handler */
{
        void exit();

        fprintf(stderr,"Numerical Recipes run-time error...\n");
        fprintf(stderr,"%s\n",error_text);
        fprintf(stderr,"...now exiting to system...\n");
        exit(1);
}

float *vector(nl,nh)
long nh,nl;
/* allocate a float vector with subscript range v[nl..nh] */
{
        float *v;

        v=(float *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(float)));
        if (!v) nrerror("allocation failure in vector()");
        return v-nl+NR_END;
}

int *ivector(nl,nh)
long nh,nl;
/* allocate an int vector with subscript range v[nl..nh] */
{
        int *v;

        v=(int *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(int)));
        if (!v) nrerror("allocation failure in ivector()");
        return v-nl+NR_END;
}

unsigned char *cvector(nl,nh)
long nh,nl;
/* allocate an unsigned char vector with subscript range v[nl..nh] */
{
```

```
        unsigned char *v;

        v=(unsigned char *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(unsigned char)));
        if (!v) nrerror("allocation failure in cvector()");
        return v-nl+NR_END;
}

unsigned long *lvector(nl,nh)
long nh,nl;
/* allocate an unsigned long vector with subscript range v[nl..nh] */
{
        unsigned long *v;

        v=(unsigned long *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(long)));
        if (!v) nrerror("allocation failure in lvector()");
        return v-nl+NR_END;
}

double *dvector(nl,nh)
long nh,nl;
/* allocate a double vector with subscript range v[nl..nh] */
{
        double *v;

        v=(double *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(double)));
        if (!v) nrerror("allocation failure in dvector()");
        return v-nl+NR_END;
}

float **matrix(nrl,nrh,ncl,nch)
long nch,ncl,nrh,nrl;
/* allocate a float matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        float **m;

        /* allocate pointers to rows */
        m=(float **) malloc((unsigned int)((nrow+NR_END)*sizeof(float*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl]=(float *) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(float)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

double **dmatrix(nrl,nrh,ncl,nch)
long nch,ncl,nrh,nrl;
/* allocate a double matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        double **m;

        /* allocate pointers to rows */
        m=(double **) malloc((unsigned int)((nrow+NR_END)*sizeof(double*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl]=(double *) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(double)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;
        /* return pointer to array of pointers to rows */
        return m;
```

```
}

int **imatrix(nrl,nrh,ncl,nch)
long nch,ncl,nrh,nrl;
/* allocate a int matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        int **m;

        /* allocate pointers to rows */
        m=(int **) malloc((unsigned int)((nrow+NR_END)*sizeof(int*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;


        /* allocate rows and set pointers to them */
        m[nrl]=(int *) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(int)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

float **submatrix(a,oldrl,oldrh,oldcl,oldch,newrl,newcl)
float **a;
long newcl,newrl,oldch,oldcl,oldrh,oldrl;
/* point a submatrix [newrl..][newcl..] to a[oldrl..oldrh][oldcl..oldch] */
{
        long i,j,nrow=oldrh-oldrl+1,ncol=oldcl-newcl;
        float **m;

        /* allocate array of pointers to rows */
        m=(float **) malloc((unsigned int) ((nrow+NR_END)*sizeof(float*)));
        if (!m) nrerror("allocation failure in submatrix()");
        m += NR_END;
        m -= newrl;

        /* set pointers to rows */
        for(i=oldrl,j=newrl;i<=oldrh;i++,j++) m[j]=a[i]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

float **convert_matrix(a,nrl,nrh,ncl,nch)
float *a;
long nch,ncl,nrh,nrl;
/* allocate a float matrix m[nrl..nrh][ncl..nch] that points to the matrix
declared in the standard C manner as a[nrow][ncol], where nrow=nrh-nrl+1
and ncol=nch-ncl+1. The routine should be called with the address
&a[0][0] as the first argument. */
{
        long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1;
        float **m;

        /* allocate pointers to rows */
        m=(float **) malloc((unsigned int) ((nrow+NR_END)*sizeof(float*)));
        if (!m)     nrerror("allocation failure in convert_matrix()");
        m += NR_END;
        m -= nrl;

        /* set pointers to rows */
        m[nrl]=a-ncl;
        for(i=1,j=nrl+1;i<nrow;i++,j++) m[j]=m[j-1]+ncol;
        /* return pointer to array of pointers to rows */
        return m;
}

float ***f3tensor(nrl,nrh,ncl,nch,ndl,ndh)
long nch,ncl,ndh,ndl,nrh,nrl;
/* allocate a float 3tensor with range t[nrl..nrh][ncl..nch][ndl..ndh] */
```

```
{
        long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1,ndep=ndh-ndl+1;
        float ***t;

        /* allocate pointers to pointers to rows */
        t=(float ***) malloc((unsigned int)((nrow+NR_END)*sizeof(float**)));
        if (!t) nrerror("allocation failure 1 in f3tensor()");
        t += NR_END;
        t -= nrl;

        /* allocate pointers to rows and set pointers to them */
        t[nrl]=(float **) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(float*)));
        if (!t[nrl]) nrerror("allocation failure 2 in f3tensor()");
        t[nrl] += NR_END;
        t[nrl] -= ncl;

        /* allocate rows and set pointers to them */
        t[nrl][ncl]=(float *) malloc((unsigned int)((nrow*ncol*ndep+NR_END)*sizeof(float)));
        if (!t[nrl][ncl]) nrerror("allocation failure 3 in f3tensor()");
        t[nrl][ncl] += NR_END;
        t[nrl][ncl] -= ndl;

        for(j=ncl+1;j<=nch;j++) t[nrl][j]=t[nrl][j-1]+ndep;
        for(i=nrl+1;i<=nrh;i++) {
                t[i]=t[i-1]+ncol;
                t[i][ncl]=t[i-1][ncl]+ncol*ndep;
                for(j=ncl+1;j<=nch;j++) t[i][j]=t[i][j-1]+ndep;
        }

        /* return pointer to array of pointers to rows */
        return t;
}

void free_vector(v,nl,nh)
float *v;
long nh,nl;
/* free a float vector allocated with vector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_ivector(v,nl,nh)
int *v;
long nh,nl;
/* free an int vector allocated with ivector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_cvector(v,nl,nh)
long nh,nl;
unsigned char *v;
/* free an unsigned char vector allocated with cvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_lvector(v,nl,nh)
long nh,nl;
unsigned long *v;
/* free an unsigned long vector allocated with lvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_dvector(v,nl,nh)
double *v;
long nh,nl;
/* free a double vector allocated with dvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_matrix(m,nrl,nrh,ncl,nch)
float **m;
long nch,ncl,nrh,nrl;
```

```
/* free a float matrix allocated by matrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_dmatrix(m,nrl,nrh,ncl,nch)
double **m;
long nch,ncl,nrh,nrl;
/* free a double matrix allocated by dmatrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_imatrix(m,nrl,nrh,ncl,nch)
int **m;
long nch,ncl,nrh,nrl;
/* free an int matrix allocated by imatrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_submatrix(b,nrl,nrh,ncl,nch)
float **b;
long nch,ncl,nrh,nrl;
/* free a submatrix allocated by submatrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_convert_matrix(b,nrl,nrh,ncl,nch)
float **b;
long nch,ncl,nrh,nrl;
/* free a matrix allocated by convert_matrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_f3tensor(t,nrl,nrh,ncl,nch,ndl,ndh)
float ***t;
long nch,ncl,ndh,ndl,nrh,nrl;
/* free a float f3tensor allocated by f3tensor() */
{
        free((FREE_ARG) (t[nrl][ncl]+ndl-NR_END));
        free((FREE_ARG) (t[nrl]+ncl-NR_END));
        free((FREE_ARG) (t+nrl-NR_END));
}

#endif /* ANSI */
```

```
##########################################################################################################
NRUTIL.C-NUMERICAL(1986)
##########################################################################################################

/* SUBROTINA AUXILIAR AO SVDCMP */

#ifndef _NR_UTILS_H_
#define _NR_UTILS_H_

static float sqrarg;
#define SQR(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)

static double dsqrarg;
#define DSQR(a) ((dsqrarg=(a)) == 0.0 ? 0.0 : dsqrarg*dsqrarg)

static double dmaxarg1,dmaxarg2;
#define DMAX(a,b) (dmaxarg1=(a),dmaxarg2=(b),(dmaxarg1) > (dmaxarg2) ?\
    (dmaxarg1) : (dmaxarg2))

static double dminarg1,dminarg2;
#define DMIN(a,b) (dminarg1=(a),dminarg2=(b),(dminarg1) < (dminarg2) ?\
```

```
        (dminarg1) : (dminarg2))

static float maxarg1,maxarg2;
#define FMAX(a,b) (maxarg1=(a),maxarg2=(b),(maxarg1) > (maxarg2) ?\
        (maxarg1) : (maxarg2))

static float minarg1,minarg2;
#define FMIN(a,b) (minarg1=(a),minarg2=(b),(minarg1) < (minarg2) ?\
        (minarg1) : (minarg2))

static long lmaxarg1,lmaxarg2;
#define LMAX(a,b) (lmaxarg1=(a),lmaxarg2=(b),(lmaxarg1) > (lmaxarg2) ?\
        (lmaxarg1) : (lmaxarg2))

static long lminarg1,lminarg2;
#define LMIN(a,b) (lminarg1=(a),lminarg2=(b),(lminarg1) < (lminarg2) ?\
        (lminarg1) : (lminarg2))

static int imaxarg1,imaxarg2;
#define IMAX(a,b) (imaxarg1=(a),imaxarg2=(b),(imaxarg1) > (imaxarg2) ?\
        (imaxarg1) : (imaxarg2))

static int iminarg1,iminarg2;
#define IMIN(a,b) (iminarg1=(a),iminarg2=(b),(iminarg1) < (iminarg2) ?\
        (iminarg1) : (iminarg2))

#define SIGN(a,b) ((b) >= 0.0 ? fabs(a) : -fabs(a))

#if defined(__STDC__) || defined(ANSI) || defined(NRANSI) /* ANSI */

void nrerror(char error_text[]);
float *vector(long nl, long nh);
int *ivector(long nl, long nh);
unsigned char *cvector(long nl, long nh);
unsigned long *lvector(long nl, long nh);
double *dvector(long nl, long nh);
float **matrix(long nrl, long nrh, long ncl, long nch);
double **dmatrix(long nrl, long nrh, long ncl, long nch);
int **imatrix(long nrl, long nrh, long ncl, long nch);
float **submatrix(float **a, long oldrl, long oldrh, long oldcl, long oldch,
        long newrl, long newcl);
float **convert_matrix(float *a, long nrl, long nrh, long ncl, long nch);
float ***f3tensor(long nrl, long nrh, long ncl, long nch, long ndl, long ndh);
void free_vector(float *v, long nl, long nh);
void free_ivector(int *v, long nl, long nh);
void free_cvector(unsigned char *v, long nl, long nh);
void free_lvector(unsigned long *v, long nl, long nh);
void free_dvector(double *v, long nl, long nh);
void free_matrix(float **m, long nrl, long nrh, long ncl, long nch);
void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch);
void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch);
void free_submatrix(float **b, long nrl, long nrh, long ncl, long nch);
void free_convert_matrix(float **b, long nrl, long nrh, long ncl, long nch);
void free_f3tensor(float ***t, long nrl, long nrh, long ncl, long nch,
        long ndl, long ndh);

#else /* ANSI */
/* traditional - K&R */

void nrerror();
float *vector();
float **matrix();
float **submatrix();
float **convert_matrix();
float ***f3tensor();
double *dvector();
double **dmatrix();
int *ivector();
int **imatrix();
unsigned char *cvector();
unsigned long *lvector();
void free_vector();
void free_dvector();
void free_ivector();
void free_cvector();
void free_lvector();
```

```
void free_matrix();
void free_submatrix();
void free_convert_matrix();
void free_dmatrix();
void free_imatrix();
void free_f3tensor();

#endif /* ANSI */

#endif /* _NR_UTILS_H_ */

##########################################################################################################
SVDCMP.C-NUMERICAL(1986)
##########################################################################################################

/* SUBROTINA DE RESOLUÇÃO DE SISTEMAS LINEARES NÃO-HOMOGENEOS */

#include <math.h>
#define NRANSI
#include ""..."prog\nrutil.h"

void svdcmp(float **a, int m, int n, float w[], float **v)
{
        float pythag(float a, float b);
        int flag,i,its,j,jj,k,l,nm;
        float anorm,c,f,g,h,s,scale,x,y,z,*rv1;

        rv1=vector(1,n);
        g=scale=anorm=0.0;
        for (i=1;i<=n;i++) {
                l=i+1;
                rv1[i]=scale*g;
                g=s=scale=0.0;
                if (i <= m) {
                        for (k=i;k<=m;k++) scale += fabs(a[k][i]);
                        if (scale) {
                                for (k=i;k<=m;k++) {
                                        a[k][i] /= scale;
                                        s += a[k][i]*a[k][i];
                                }
                                f=a[i][i];
                                g = -SIGN(sqrt(s),f);
                                h=f*g-s;
                                a[i][i]=f-g;
                                for (j=l;j<=n;j++) {
                                        for (s=0.0,k=i;k<=m;k++) s += a[k][i]*a[k][j];
                                        f=s/h;
                                        for (k=i;k<=m;k++) a[k][j] += f*a[k][i];
                                }
                                for (k=i;k<=m;k++) a[k][i] *= scale;
                        }
                }
                w[i]=scale *g;
                g=s=scale=0.0;
                if (i <= m && i != n) {
                        for (k=l;k<=n;k++) scale += fabs(a[i][k]);
                        if (scale) {
                                for (k=l;k<=n;k++) {
                                        a[i][k] /= scale;
                                        s += a[i][k]*a[i][k];
                                }
                                f=a[i][l];
                                g = -SIGN(sqrt(s),f);
                                h=f*g-s;
                                a[i][l]=f-g;
                                for (k=l;k<=n;k++) rv1[k]=a[i][k]/h;
                                for (j=l;j<=m;j++) {
                                        for (s=0.0,k=l;k<=n;k++) s += a[j][k]*a[i][k];
                                        for (k=l;k<=n;k++) a[j][k] += s*rv1[k];
                                }
                                for (k=l;k<=n;k++) a[i][k] *= scale;
                        }
                }
                anorm=FMAX(anorm,(fabs(w[i])+fabs(rv1[i])));
        }
        for (i=n;i>=1;i--) {
```

```
		if (i < n) {
			if (g) {
				for (j=l;j<=n;j++)
					v[j][i]=(a[i][j]/a[i][l])/g;
				for (j=l;j<=n;j++) {
					for (s=0.0,k=l;k<=n;k++) s += a[i][k]*v[k][j];
					for (k=l;k<=n;k++) v[k][j] += s*v[k][i];
				}
			}
			for (j=l;j<=n;j++) v[i][j]=v[j][i]=0.0;
		}
		v[i][i]=1.0;
		g=rv1[i];
		l=i;
	}
	for (i=IMIN(m,n);i>=1;i--) {
		l=i+1;
		g=w[i];
		for (j=l;j<=n;j++) a[i][j]=0.0;
		if (g) {
			g=1.0/g;
			for (j=l;j<=n;j++) {
				for (s=0.0,k=l;k<=m;k++) s += a[k][i]*a[k][j];
				f=(s/a[i][i])*g;
				for (k=i;k<=m;k++) a[k][j] += f*a[k][i];
			}
			for (j=i;j<=m;j++) a[j][i] *= g;
		} else for (j=i;j<=m;j++) a[j][i]=0.0;
		++a[i][i];
	}
	for (k=n;k>=1;k--) {
		for (its=1;its<=30;its++) {
			flag=1;
			for (l=k;l>=1;l--) {
				nm=l-1;
				if ((float)(fabs(rv1[l])+anorm) == anorm) {
					flag=0;
					break;
				}
				if ((float)(fabs(w[nm])+anorm) == anorm) break;
			}
			if (flag) {
				c=0.0;
				s=1.0;
				for (i=l;i<=k;i++) {
					f=s*rv1[i];
					rv1[i]=c*rv1[i];
					if ((float)(fabs(f)+anorm) == anorm) break;
					g=w[i];
					h=pythag(f,g);
					w[i]=h;
					h=1.0/h;
					c=g*h;
					s = -f*h;
					for (j=1;j<=m;j++) {
						y=a[j][nm];
						z=a[j][i];
						a[j][nm]=y*c+z*s;
						a[j][i]=z*c-y*s;
					}
				}
			}
			z=w[k];
			if (l == k) {
				if (z < 0.0) {
					w[k] = -z;
					for (j=1;j<=n;j++) v[j][k] = -v[j][k];
				}
				break;
			}
			if (its == 30) nrerror("no convergence in 30 svdcmp iterations");
			x=w[l];
			nm=k-1;
			y=w[nm];
			g=rv1[nm];
			h=rv1[k];
```

```
                        f=((y-z)*(y+z)+(g-h)*(g+h))/(2.0*h*y);
                        g=pythag(f,1.0);
                        f=((x-z)*(x+z)+h*((y/(f+SIGN(g,f)))-h))/x;
                        c=s=1.0;
                        for (j=l;j<=nm;j++) {
                                i=j+1;
                                g=rv1[i];
                                y=w[i];
                                h=s*g;
                                g=c*g;
                                z=pythag(f,h);
                                rv1[j]=z;
                                c=f/z;
                                s=h/z;
                                f=x*c+g*s;
                                g = g*c-x*s;
                                h=y*s;
                                y *= c;
                                for (jj=1;jj<=n;jj++) {
                                        x=v[jj][j];
                                        z=v[jj][i];
                                        v[jj][j]=x*c+z*s;
                                        v[jj][i]=z*c-x*s;
                                }
                                z=pythag(f,h);
                                w[j]=z;
                                if (z) {
                                        z=1.0/z;
                                        c=f*z;
                                        s=h*z;
                                }
                                f=c*g+s*y;
                                x=c*y-s*g;
                                for (jj=1;jj<=m;jj++) {
                                        y=a[jj][j];
                                        z=a[jj][i];
                                        a[jj][j]=y*c+z*s;
                                        a[jj][i]=z*c-y*s;
                                }
                        }
                        rv1[l]=0.0;
                        rv1[k]=f;
                        w[k]=x;
                }
        }
        free_vector(rv1,1,n);
}
#undef NRANSI
```

##

PYTHAG.C-NUMERICAL(1986)

##

```
/* SUBROTINA AUXILIAR AO SVDCMP */

#include <math.h>
#define NRANSI
#include "\"...\"\prog\o\nrutil.h"

float pythag(float a, float b)
{
        float absa,absb;
        absa=fabs(a);
        absb=fabs(b);
        if (absa > absb) return absa*sqrt(1.0+SQR(absb/absa));
        else return (absb == 0.0 ? 0.0 : absb*sqrt(1.0+SQR(absa/absb)));
}
#undef NRANSI
```

##

SUBS.H

```
##########################################################################################################

/* SUBROTINA INDICADORA DE BIBLIOTECA */

void DEN2a(int p,int pd,int n1,int m,float*b1,float*sbt,float*pps,
        float*yh,float*zai,float*zh,float*yy,float*q1);
void DEN2delta(int n2,int m,float*ppd,float*no,float*yy,float*b2,float*q2);
void DEN2err(int p,int pd,float*zai,float*zb,float*hh,float*zh,float*hzt,float*err);
void TEG(int p,int pd,int n1, float *zb,float *zai,float *chi2);
int nmsus(int n1,int*mza,float*zaj,float*zcr,int bool,FILE*fp);
void vmsus(int n1,int*mza,int*mvs,float*zaj,float*zcr);
void DETECTAR(int n1,int ni,int c,int rq,int q,int t1,int*viv,int*mvs,int*mza,float*zaj,float*zcr,float*x,FILE*fp);
int lu(int rf,int*irf,float*p1f);
void vbacks(int rf,int rs,int*irf,float*p1f,float*uf);
void prod_m(int s,int rf,int w,float*p2f,float*uf,float*puf);
void ZHTf(int ni,int p,int m,int rq,int blo,int nulo,int c1,int *viv,float *dd,
float *yh,float *b2f,float *zh);

##########################################################################################################
 SUBS1.H
##########################################################################################################

/* SUBROTINA INDICADORA DE BIBLIOTECA */

int lu(int rf,int *irf,float *p1f);
 void vbacks(int rf ,int rs, int *irf, float *p1f, float *uf);
 void prod_m(int s,int rf,int w,float *p2f,float *uf,float *puf);

##########################################################################################################
FIM DAS SUBROTINAS: FIM DOS PROGRAMAS.
##########################################################################################################
```

D. LISTA DE FIGURAS