

Title of dissertation **Data Reconciliation for Refinery Hydrogen Network**

I SITI RAFIDAH BINTI AB. RASHID hereby allow my dissertation to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The dissertation becomes the property of UTP
2. The IRC of UTP may make copies of the dissertation for academic purposes only.
3. This dissertation is classified as

Confidential


Non-confidential

If this dissertation is confidential, please state the reason:

The contents of the dissertation will remain confidential for _____ years.

Remarks on disclosure:

Endorsed by



Signature of Author

Siti Rafidah Ab. Rashid
Bandar Sri Iskandar, Perak

Signature of Supervisor

Dr. Shuhaimi Mahadzir
Bandar Sri Iskandar, Perak

Date: 6/8/07

Date: _____

UNIVERSITI TEKNOLOGI PETRONAS

Approval by Supervisor (s)

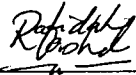
The undersigned certify that they have read, and recommend to The Postgraduate Studies Programme for acceptance, a dissertation entitled

“Data Reconciliation for Refinery Hydrogen Network”

submitted by

Siti Rafidah Binti Ab. Rashid

for the fulfilment of the requirements for the degree of Master of Science in Process
Integration



Date: 6/8/07

Signature : _____

Main Supervisor : *Dr. Shuhaimi Mahadzir*

Date : _____

Co-Supervisor 1 : _____

UNIVERSITI TEKNOLOGI PETRONAS

Data Reconciliation for Refinery Hydrogen Network

By

Siti Rafidah Binti Ab. Rashid

A DISSERTATION

SUBMITTED TO THE POSTGRADUATE STUDIES PROGRAMME

AS A REQUIREMENT FOR THE

DEGREE OF MASTER OF SCIENCE IN PROCESS INTEGRATION

Chemical Engineering


BANDAR SERI ISKANDAR,

PERAK

June, 2007

DECLARATION

I hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Signature : 

Name : *Siti Rafidah binti Ab. Rashid*

Date : 6/8/07

ACKNOWLEDGEMENT

It is a pleasure to thank the many people who made this thesis possible.

It is difficult to overstate my gratitude to my supervisors, Dr. Nan Zhang and Dr Shuhaimi Mahadzir. With their enthusiasms, their inspirations, and their great efforts to explain things clearly, they helped to make mathematics and programming fun for me. Throughout my thesis-writing period, they provided encouragements, sound advices, good teachings and lots of good ideas.

Special thanks to the management of Universiti Teknologi Petronas (especially Assoc Prof Dr Hilmi Mukhtar, Assoc Prof Dr Kamarul Ariffin and Assoc Prof Dr. Mohd Ibrahim) for making my attachment at The University of Manchester possible.

I would like to thank many people who have taught me: my high school teachers, my undergraduate lecturers at Skudai, Johor and my graduate teachers (Assoc. Prof Dr Kamarul Ariffin, Assoc. Prof Dr. Azmi Mohd Shariff, Assoc. Prof Dr. Suzana Yusup, Assoc. Prof. Dr Mohamed Ibrahim Abdul Mutalib, Dr Mohd Azmuddin Bin Abdullah and Dr Marappagounder Ramasamy).

I am indebted to my colleagues for providing a stimulating and fun environment. I am very grateful to Kalai, Anis, Imm, Shireen, Shal, Imran, Mike and others in Manchester. Not to forget my dear friends, Zana, Halim, Azli, Amlaku, Sadiq, Hammami, Kak Linda, Tazli and Nabilah here in Malaysia.

I wish to thank my entire family for providing a loving environment for me. My sisters, Aida and Aishah and brother, Shah were particularly supportive. Without their supports and loves, I doubt I can complete this thesis successfully.

My deepest gratitude and thanks are for my parents, Ab Rashid Mohd Noor and Hadibah Abu. They bore, raised, taught, supported and loved me. To them I dedicate this thesis.

ABSTRACT

Data reconciliation can be used for many applications namely as instrumentation maintenance, plant optimisation, advanced process control and many more. Data reconciliation is relatively new in Chemical Engineering field. Its applications especially in oil refineries are limited. Using process models (material balance) as constraints, data reconciliation enables us to obtain estimations of process variables by adjusting the measurements. This technique enhances the accuracy of the process variables as opposed to the measurements themselves.

This project focuses on refinery hydrogen networks. As the new environmental regulations for low sulfur fuel and heavier crude oil supplies, the refiners have no option but to really manage their hydrogen networks better. The hydrogen network management is based on plant data. Unfortunately the inaccurate plant data and unmeasured process streams cause difficulty in establishing of material balance of overall systems.

The purpose of this project is to reconcile refinery hydrogen network data specifically flowrates. A program was developed for linear steady state systems that comprise of reconciliation of measured flowrates, estimation of unmeasured flowrates and systematic error detection. The developed procedure is easily executable in Microsoft Excel which was programmed with Visual Basic.

The output of the program is a reconciled flowrates of a hydrogen network combined with estimation of unmeasured flowrates. It also gives an indication of the presence of systematic error in the measurement via Global Testing systematic error detection technique.

ABSTRAK

Penyelarasan data digunakan untuk pelbagai seperti penyelenggaraan instrumentasi, mengoptimumkan loji, kawalan proses termaju dan pelbagai lagi. Secara relatifnya, penyelarasan data adalah baru dalam bidang Kejuruteraan Kimia. Aplikasinya di loji penapisan minyak juga terhad. Dengan menggunakan model proses (keseimbangan jisim) sebagai kekangan, penyelarasan data membolehkan pembolehubah proses dianggarkan dengan mengubah bacaan ukuran. Teknik ini menambah ketepatan pembolehubah proses berbanding bacaan asal.

Projek ini difokuskan kepada rangkaian hidrogen di loji penapisan minyak. Memandangkan terdapat peraturan baru yang lebih ketat terhadap kandungan sulfur di dalam bahan api dan bekalan minyak mentah yang lebih 'berat', para penapis tidak mempunyai banyak pilihan selain menguruskan rangkaian hidrogen mereka dengan lebih baik. Pengurusan rangkaian hidrogen adalah bermula dengan data dari loji. Malangnya, data yang tidak tepat dan tidak diukur menimbulkan masalah dalam memberikan keseimbangan jisim untuk keseluruhan sistem.

Tujuan projek ini adalah untuk menyelaraskan data untuk rangkaian hidrogen di loji penapisan terutamanya kadar aliran. Satu aturcara telah didirikan untuk sistem yang linear dan pada keadaan mantap. Aturcara ini terdiri dari penyelarasan kadar aliran, penganggaran kadar aliran yang tidak diukur dan pengesanan ralat sistematik. Program tersebut telah didirikan di dalam *Microsoft Excel* menggunakan *Visual Basic Programming*.

Keluaran dari program ini adalah kadar aliran yang telah diselaraskan dan juga anggaran kepada kadar alir yang tidak diukur. Ia juga memberikan gambaran tentang kewujudan ralat sistematik di dalam ukuran melalui teknik pengesanan ralat sistematik iaitu '*Global Testing*'.

CONTENTS

STATUS OF THESIS	i
CERTIFICATION OF APPROVAL	ii
TITLE PAGE	iii
DECLARATION	iv
ACKNOWLEDGEMENT	v
ABSTRACT	vi
CONTENTS	viii
LIST OF FIGURES	xi
LIST OF TABLES	xii
NOMENCLATURE	xiii
CHAPTER ONE: INTRODUCTION	1
1.1 Measurement Error	3
1.1.1 Systematic Error	3
1.1.2 Random Error	4
1.2 Data Rectification	5
1.3 Data Reconciliation	5
1.4 Problem Classifications	7
1.5 Refinery Hydrogen Networks	8
1.6 Problem Statement	10
1.7 Objectives of the Study	10
1.8 Scope of the Study	10
1.9 Structure of Dissertation	11
CHAPTER TWO: LITERATURE REVIEW	13
2.1 Least Squares	13
2.2 Data Reconciliation	14
2.3 Systematic Error Detection	16
2.4 Estimation of Variances	18
2.5 Dynamic Data Reconciliation	20
2.6 Data Reconciliation and Refinery Hydrogen Network	20
2.7 Summary	21

CHAPTER THREE: METHODOLOGY	22
3.1 General Data Reconciliation Methodologies	22
3.2 Linear Data Reconciliation	25
3.2.1 Linear System with Measured and Unmeasured Variables	25
3.2.2 QR Decomposition	27
3.2.2.1 Householder Transformation	29
3.2.2.1.1 Example of Calculations Using Householder Transformation	30
3.2.3 Construction of Projection Matrix	33
3.2.4 Reconciliation of Measured Variables	34
3.2.5 Determination of Unmeasured Variables	35
3.3 Systematic Error Detection	35
3.3.1 Global Test for Systematic Error Detection	36
3.4 Algorithms and Flowcharts of Linear Data Reconciliation Program	38
3.5 Summary of Methodology	40
CHAPTER FOUR: CASE STUDY	41
4.1 Description of the Refinery Network	41
4.2 Data Gathering	45
4.2.1 Material Balance of Measured A_x & Unmeasured Variables A_u	46
4.3 Linear Data Reconciliation for Measured Variables	50
4.4 Determination of Unmeasured Variables	52
4.5 Systematic Error Detection	53
4.6 Summary	55
CHAPTER FIVE: CONCLUSIONS AND FUTURE WORK	56
5.1 Conclusions	56
5.2 Future Work	56

REFERENCES	58
APPENDICES	62
Appendix A1: Visual Basic Code for QR Decomposition Solver	62
Appendix A2: Visual Basic Code for Linear Data Reconciliation Solver	67
Appendix A3: Visual Basic Code for Linear Data Reconciliation Solver Functions	77

LIST OF FIGURES

CHAPTER 1

Figure 1.1: Type of Systematic Errors	4
Figure 1.2: Online Data Collection and Conditioning System	7
Figure 1.3: P&ID of a Single Producer – Single Consumer Oil Refinery Hydrogen Network	9

CHAPTER 3

Figure 3.1: General Data Reconciliation	23
Figure 3.2: Classes of Data Reconciliation	24
Figure 3.3: Variable Classifications	25
Figure 3.4: Subsets of Q and R, Q_1 , Q_2 , R_1 and R_2	34
Figure 3.5: Systematic Error Detection Technique Using Global Test	37
Figure 3.6: Flowchart of “QR Solver”	38
Figure 3.7: Flowchart of “Linear Data Reconciliation” Program	39

CHAPTER 4

Figure 4.1: Simplified Hydrogen Network for Analysis	43
Figure 4.2: Simplified PFD for Data Reconciliation	44
Figure 4.3: Elements of Measured Variable, Matrix Ax	48
Figure 4.5: Elements of Unmeasured Variable, Matrix Au	49
Figure 4.6: QR Decomposition Solver Interface	50
Figure 4.7: Linear Data Reconciliation Solver Interface	51
Figure 4.8: Chi-Square Calculator	54
Figure 4.9: Comparisons of Linear Data Reconciliation between VBA Excel and Matlab ®	55

LIST OF TABLES

CHAPTER 4

Table 4.1: Unit Description of Refinery Hydrogen Network	42
Table 4.2: Lists of Measured Streams	45
Table 4.3: Lists of Unmeasured Streams	46
Table 4.4: Linearly Reconciled Flowrates	51
Table 4.5: Calculated Unmeasured Flowrates	52

NOMENCLATURE

a is measurement adjustment

A_x corresponds to the measured variables with $m \times n$ dimensions

A_u corresponds to the unmeasured variables with $m \times p$ dimensions

A' is transpose of matrix A

D_i is diagonal matrix

H is Householder transformation

H_0 , *null hypothesis*, is that no systematic error is present, and

H_1 , *alternative hypothesis*, is that one or more systematic errors are present in the system.

m is number of unit operations involved

n is number of measured variables

p is number of unmeasured variables

Q is an orthogonal matrix,

Q_1 is $(m \times r)$ matrix subset of matrix Q

Q_2 is $[m \times (m-r)]$ matrix subset of Q

r is the vector of balance residuals

R is an upper triangular matrix

R_{11} is $(r \times r)$ matrix subset of matrix R

R_2 is $(r \times (n - r))$ matrix subset of matrix R

u is a nonzero vector of unmeasured variables

\hat{u} is estimates of variable u

V is variance-covariance matrix when random error is normally distributed with variance-covariance matrix Σ .

\bar{x} is a true value of the measured variable

\hat{x} refers to reconciled values

y is measured process variables

α is level of significance

Σ is variance-covariance matrix

ε is measurement error

Π is permutation matrix (adjusted identity matrix)

χ^2 is chi-squared distribution. It is used to prove the null hypothesis is true or not.

$\chi^2_{\alpha\nu}$ is the critical value of chi-square distribution at the chosen α level of significance and ν degrees of freedom
 γ refers to Global Testing function

CHAPTER 1

INTRODUCTION

For a chemical plant, especially an oil refinery, it is essential to monitor and optimize its performance as it will affect its profitability and flexibility. This very much relies on the data obtained from the process system. Reliable process data are required for process control, online optimisation and plant performance monitoring.

In an oil refinery, hydrogen is considered as an essential utility. Hydrogen is mainly used in hydrotreaters to desulfurise fractions and hydrocrackers to upgrade heavier fractions into lighter and valuable fractions. As the new environmental regulations for low sulfur fuel and heavier crude oil supplies, the refiners have no option but to manage their hydrogen networks better. The failure of managing the hydrogen network may create bottleneck the refinery productions as a result of hydrogen shortfall. It also involves capital investments if hydrogen production capacity needs to be scaled up. On top of that, if more hydrogen needs to be purchased or produced, a significant increase in operating cost will be introduced.

The above reasoning shows that hydrogen network management is an essential 'task' in oil refineries.

Hydrogen Network Management is based on plant data. Unfortunately the inaccurate plant data and unmeasured process streams cause difficulty in establishing of material balance of overall systems.

It is known that process measurements are inherently contaminated by errors during the measurement, processing and transmission of the measured signals. It should be emphasised that errors in measured data often lead to considerable deterioration in monitoring and measuring plant performance. Moreover, it is a normal practice that not all streams are measured for either flow rates or compositions. This is due to the high capital cost of installing instrumentation systems for relatively non-important streams. However, this makes the balance validation process more difficult and inaccurate.

The plant data are normally adjusted so that the known errors and measurement noise can be minimised. Measurement error is the amount by which an observation differs from its expected value. Errors can be classified as random errors and systematic or gross errors. Random errors are errors that affect the precision of a set of measurements. Random error scatters measurements above and below the mean, with small random errors being more likely than large ones. On the other hand, systematic errors or gross errors are the undetected mistakes that cause a measurement to be very much farther from the mean measurement than other measurements. It is also a bias in measurement which leads to measured values being systematically too high or too low. All measurements are prone to systematic error. In short, a systematic error is any biasing effect caused by either methods of observation or instruments used.

In the last 40 years, techniques which will reduce, if not eliminate, the random and systematic errors have been developed. Data reconciliation is a technique that has been developed to improve the accuracy of measurement. The main difference between data reconciliation and other filtering techniques is that data reconciliation uses process model constraints and obtains estimates of process variables by adjusting the process measurements to satisfy the constraints. Typical constraints in chemical plant include mass and energy conservations.

It is logical that the reconciled data are more accurate than the measurement values. The reconciled data are also more consistent as they satisfy the process constraints. However, in order for data reconciliation to be effective, there should be no systematic error presence in the measurement. Therefore, gross error detection is used concurrently with data reconciliation. It is developed to identify and eliminate systematic errors and thus improve the accuracy of the measured variables. The combination of data reconciliation and systematic error removal is referred as data rectification.

In this project, both techniques will be applied in a case study of a hydrogen network in an oil refinery.

1.1 Measurement Error

Measurement Error is the amount by which an observation differs from its expected value. In many processes, errors, in process data can cause deterioration in plant efficiency. Errors, ε , can be defined as follows:

$$\varepsilon = y - \bar{x} \quad (1.1)$$

where y is measured process variables and \bar{x} is a true value of the measured variable.

Therefore, a true value of measured variable, \bar{x} , can be expressed as

$$\bar{x} = y + a \quad (1.2)$$

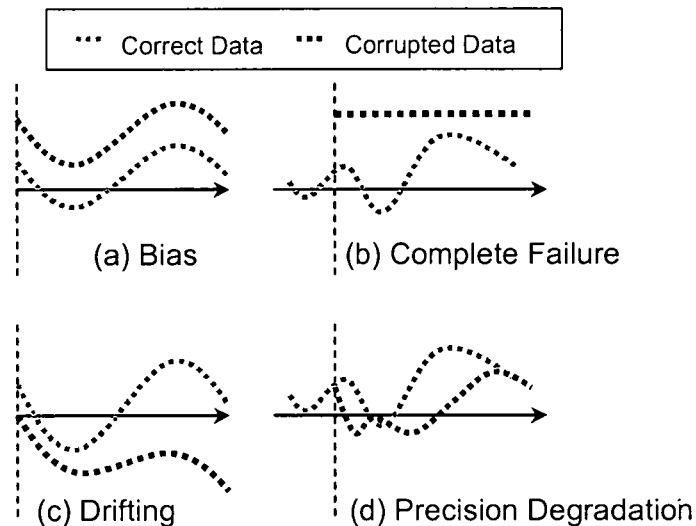
where a is measurement adjustment. Measurement adjustments are the differences between the measured and estimated variables. Measurement adjustments can be determined by satisfying the process model constraints.

Data can lead to significant deterioration of plant performance. Small errors can cause a significant decline in the control system performance, while the large errors can nullify the achievable gains via optimisation. The biggest concern is the plant operator might operate the plant in the unsafe operating regime as a result of measurement error.

1.1.1 Systematic Error

By definition, systematic errors are the undetected mistakes that cause a measurement to be very much farther from the mean measurement than other measurements. It is also a bias in measurement which lead to measured values being systematically too high or too low. Among the sources of systematic errors are fouling of the sensors, wear and tear, solid deposition on the probe, corrosion on the sensors, miscalibration and instrument malfunction. Systematic error occurs less frequent as opposed to random error. However the magnitude is much larger.

Figure 1.1 below shows the type of systematic errors. Figure 1.1 (a) shows the profile of bias. Bias is consistent or repeatable offset between the correct data and corrupted data. The profile of complete failure of the instrumentation is shown in Figure 1.1 (b). Figure 1.1 (c) shows the profile of drifting. Drifting normally occurs when instrument or test-system performance changing after a calibration has been done. Figure 1.1 (d) shows a profile of precision degradation. It is mostly caused by uncertainty in the determination of mechanical parameters



(Source: Narasimhan and Jordache, 2000)

Figure 1.1: Type of Systematic Errors

1.1.2 Random Error

Random errors are errors that affect the precision of a set of measurements. Random error scatters measurements above and below the mean. Small random errors are being more likely than the larger ones. Random errors normally occur during networks transmission, power supply fluctuation, signal conversion noise, filtering and changes in the ambient condition. It is difficult for us to predict the sign and the magnitude of the error. Therefore, we have to use the probability distribution to characterise random errors.

1.2 Data Rectification

Data rectification is the task of removing errors from measured data. It is an important task since most process operation tasks rely on the rectified data. It involves estimation of the underlying noise free variables from noisy measurements. Unmeasured variables and model parameters may also need to be estimated along with rectification. It may be posed as the following optimisation problem.

Filters, both analog and digital are used to attenuate the high frequency noise. Large systematic errors are detected using data validation check. The check is done based on the measured data and rate of changing as per predefined limits.

A smart sensor is used to perform diagnostic check. The main functions of the smart sensor are to check the hardware problem and to check whether the data are acceptable or not. Normally the outliers' detection is performed by using the SQC (Statistical Quality Control) test.

All these activities are part of data rectification process. In general data rectification is applied to each measured variable separately in contrast with data reconciliation which is done with respect to inter-relationship with other process variables. Therefore the consistency of the data is only assured via data reconciliation. However, data rectification must be used as the first step to reduce random errors.

1.3 Data Reconciliation

In short, data reconciliation can be defined as an adjusting process of data to satisfy process model constraints such as material and energy balances. The element of constraints is what differentiates data reconciliation from data rectification. It enables the estimation of process variables by adjusting the measurements. This technique improves the accuracy of the process variables as opposed to the measurements themselves. Good data reconciliation is when there is no systematic error in the measurements and uses the process models as the constraints. Therefore, there is always a need to have a systematic detection procedure in data reconciliation.

In an industrial process, data reconciliation is performed just before the application of parameter estimation, simulation, optimisation, advanced process control, accounting and instrumentation maintenance. Figure 1.2 shows various operations and the position occupied by data reconciliation in data conditioning for online industrial applications.

In order to reduce the gross and random errors, several methods have been developed. These include the analog and digital filters which can be used to ease the affects of high frequency noise. On top of that, with the availability of smart sensors, it is now possible to determine whether the measured data is acceptable and whether there is any hardware problem. A more recent technique such as Statistical Quality Control (SQC) test is able to detect significant errors. SQC is used to detect significant errors or outliers in process data (Narasimhan and Jordache, 2000). These three methods are usually applied to each variable individually. Hence, they do not ensure consistency of the data with respect to the interrelationships between different process variables.

Process data in industrial process can be used in several purposes especially for continuous process improvement. Therefore, data acquisition and historian software is required to gather and archive the plant data. Next in data collection and system is data retrieval process. It is a technique and process of searching, recovering, and interpreting information from large amounts of stored data. Data validation and reconstruction is a process of checking the data for correctness, or the determination of compliance with applicable standards, rules, and conventions. It involves data restoration by analyzing the original data. Next in online data collection and conditioning system is data filtering. It selects data that are matched to the pre-set conditions. Others are either deleted or neglected from the users. Prior to the applications of data reconciliation is data reconciliation. A great detail discussion of data reconciliation will be given in later sections.

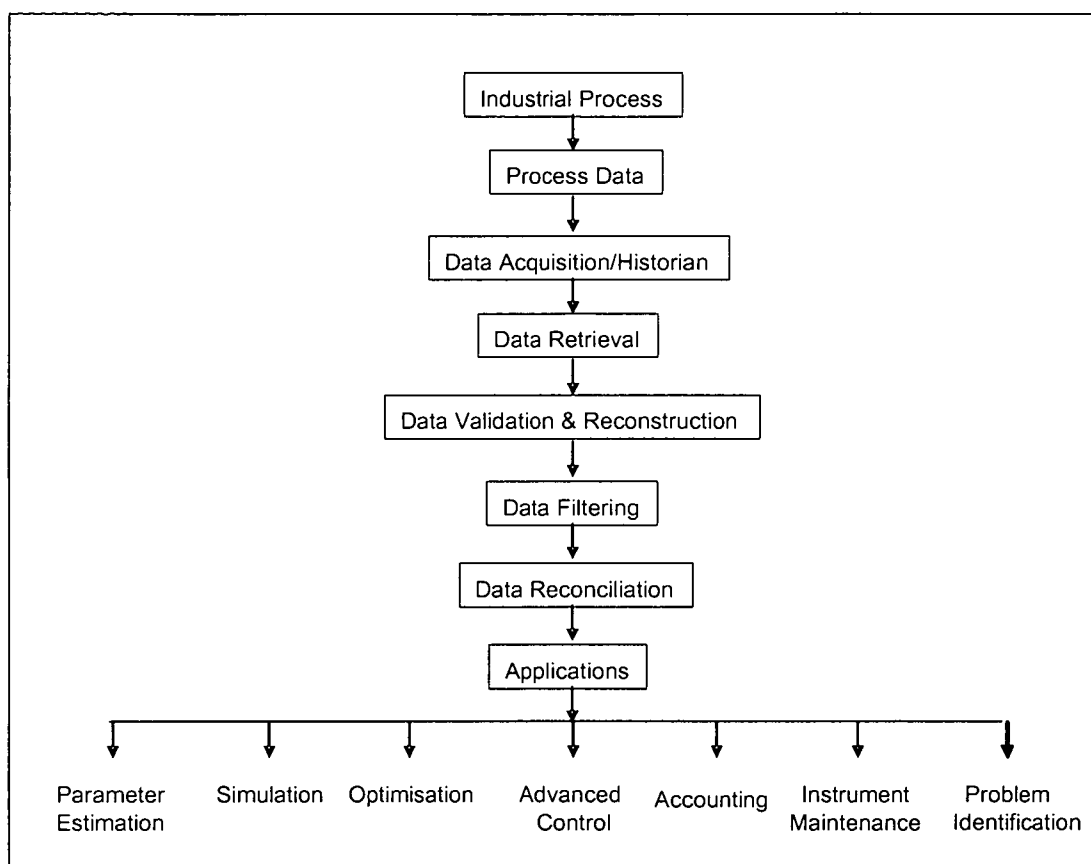


Figure 1.2: Online Data Collection and Conditioning System
 (Reference: <http://.che.iitm.ac.in/~naras/ch544/introDRGED.pdf>
 (Date last visited: 16th February 2007))

1.4 Problem Classifications

If a single variable is considered, the problem is a linear problem. For instance, we only consider flowrate as the only variable for the system.

On the other hand, function of two variables is bilinear if it is linear with respect to each of its variables. For example, flowrates and compositions or enthalpies are considered as variables, the problem now becomes a bilinear problem. In practice, data reconciliation considers component balance and energy balance other than mass balance alone. These additional constraints transform a linear problem to bilinear if

two variables are involved. It can also transform a linear problem to a non-linear problem if more than two variables are involved.

On top of that, if more than two variables are taken into account, then the problem is termed as a non-linear problem. For example, if we consider equilibrium relationship, physical properties and other correlations, the problem now has more than just two variables. Therefore, it is classified as a non-linear problem.

1.5 Refinery Hydrogen Networks

Refinery Hydrogen Network is the distribution path of the supply and demand of hydrogen in a refinery. The major hydrogen supply in a refinery is typically from a steam reforming process and/or a catalytic reforming process. These processes produce hydrogen as by-product. The hydrogen demands come from various processes in the refinery such as hydrocracking and hydrotreating. However, external supply of hydrogen could be considered to refineries if the supply from the steam reformer and the catalytic reformer is not enough to meet the demand. This supply-demand relationship of hydrogen in refineries creates the refinery hydrogen networks.

Figure 1.3 shows an oil refinery hydrogen network that consists of a single producer and a single consumer. The system shown is the hydrogen line at both producer and consumer. The producer in Figure 1.3 is denoted as unit 400, namely as naphtha catalytic reformer. The hydrogen produced in unit 400 is sent to the hydrogen header before distributed to its consumers. The only consumer shown in the figure is diesel desulphurisation unit and it is denoted as unit 300.

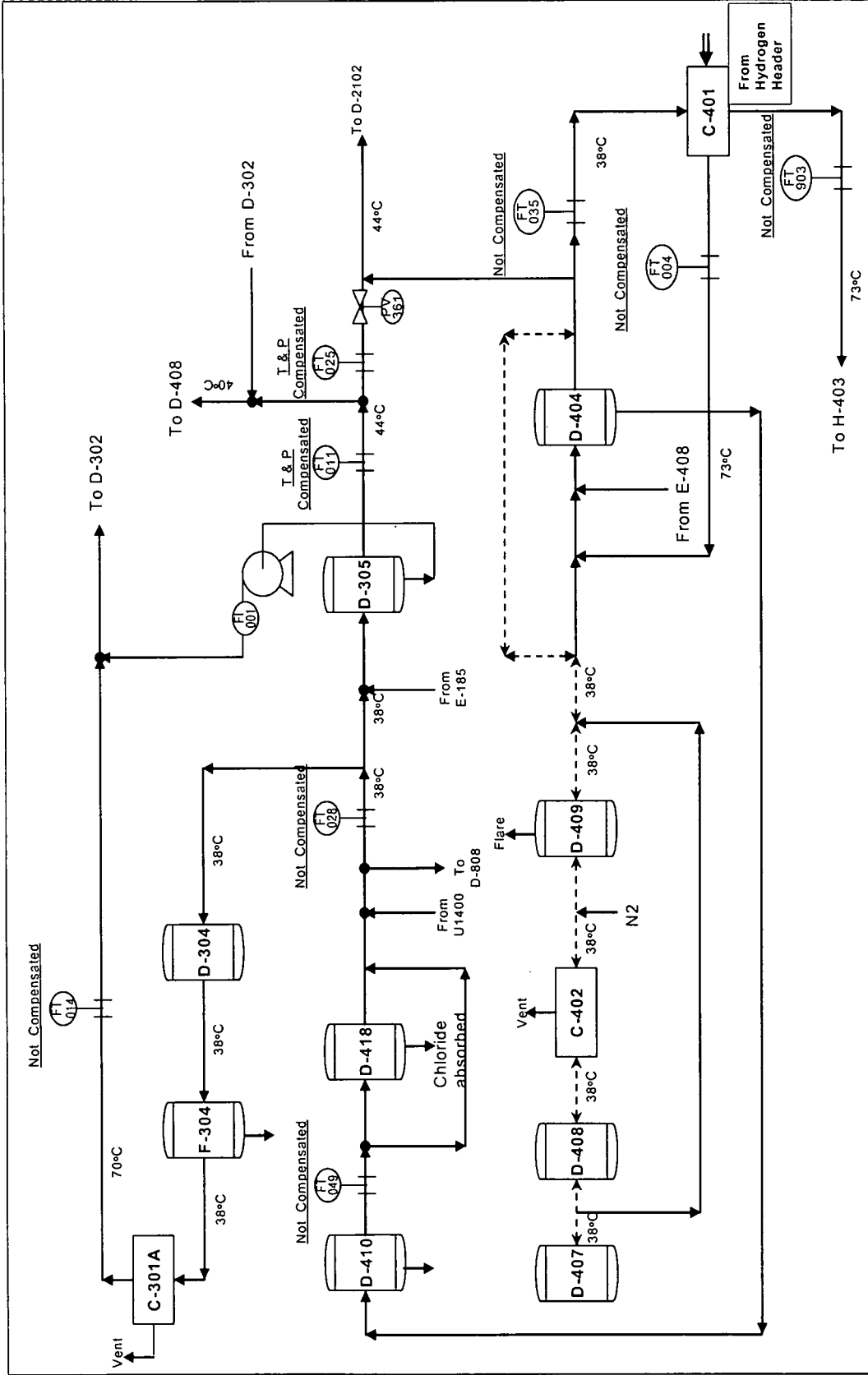


Figure 1.3: P&ID of a Single Producer - Single Consumer Oil Refinery Hydrogen Network

1.6 Problem Statement

Due to strict environmental regulations and heavier crude oil supply, hydrogen becomes a crucial utility in refineries. Therefore, hydrogen demand has increased significantly and need to be managed wisely. The introduction of Hydrogen Network Management helps the refiners to identify the best route to an optimised hydrogen network. However, the constraints are not all process data are reliable. On top of that, some streams are not even measured. This caused difficulties to the refiners to implement Hydrogen Network Management. In order to overcome this issue, data reconciliation is suggested. It helps the refiners to execute hydrogen network effectively.

In this work, a linear data reconciliation program has been developed. An existing oil refinery hydrogen network is selected as a case study. Plant data collected are used as a base case. The program will give the refiners the linearly reconciled flowrates of refinery hydrogen network. Therefore, the true state of the network can be pictured using the program.

1.7 Objectives of the Study

The main objective of this project is to study data reconciliation and systematic error techniques for linear systems. A systematic approach of data reconciliation is then developed. The developed technique is later applied to a case study of a refinery hydrogen networks.

1.8 Scope of the Study

This project is bounded to the following boundaries:

- This work only considers reconciliation of data for steady state process.
- Only one variable is taken into account ie. flowrate and this resulting in a linear system.
- Only the material balance is considered as the main constraint.

1.9 Structure of Dissertation

This dissertation consists of five chapters. A brief overview of each chapter is as follows.

Chapter 1

An overview of data analysis technique and definitions is presented. A brief discussion of the topics and definitions are given in this chapter. In order to give better and comprehensive understanding, terminologies and jargons that will be used in the thesis will be provided. As the case study of this project is a hydrogen network of a refinery, a brief overview of the system is given in this chapter along with the objectives and scope of this research.

Chapter 2

Current and future trends in the data reconciliation directions are discussed in Chapter 2. In the early part of this chapter, the focus is on the least squares and linear data reconciliation concepts and the histories behind them. This is to acknowledge the contributions made by some prominent researchers as their findings and works are the basis of this study. In the later section, the systematic error detection and elimination methods are reviewed. This is part of data rectification but apart from the data reconciliation. In this chapter, the attempt of applying data reconciliation in the refinery hydrogen networks will also be presented.

Chapter 3

In Chapter 3, the methodology, procedure, underlying assumptions and application of the linear data reconciliation technique is discussed. In this chapter, specific focus is given on the steady state data reconciliation for linear systems. On top of that, the procedure of detecting systematic error will be discussed briefly.

Chapter 4

A case study of a refinery hydrogen networks is used for the method developed. The method was applied to a real refinery hydrogen networks. At first, the linear data reconciliation is considered. Then the systematic error detection technique is applied.

Chapter 5

Chapter 5 presents the conclusions from the work done in this dissertation. Some suggestions are also made for the future work.

CHAPTER 2

LITERATURE REVIEW

In order to achieve an excellent process control, online optimization and plant performance monitoring, reliable information and data supplied by line instruments and sensors must be ensured. As availability of reliable and low cost sensors is very low, the combined sensors with digital data acquisition are more preferable nowadays. Therefore, the method of data monitoring and reconciliation has gained importance in ensuring reliable data.

Data reconciliation is a procedure of adjusting measured data so that they obey the constraint of the conservation laws ie. mass and energy balances (Ragot et al., 1990). The general assumption of data reconciliation is measurement errors. These errors must be eliminated before being fed to controllers and optimizers.

Inconsistent and inaccurate data cause a misleading and confusing picture of the actual situation of the plant condition. On top of the difficulty to eliminate the errors, some of the variables are not measured due to feasibility issues and cost considerations (Kim et al., 1996).

2.1 Least Squares

The method of least squares has the rule that the sum of the squares of the errors should be made a minimum to obtain the adjusted values of observed quantities. Andrien-Marie Legendre published this method in 1805 in Paris titled “New Methods for the Determination of Comet Orbits” (Merriman, 1877).

In 1809, Carl Friedrich Gauss, however, claimed his method of least squares was accomplished earlier in 1795 in estimating the Ceres asteroid trajectory (Merriman, 1877).

2.2 Data Reconciliation

Data reconciliation has received a great attention in the chemical engineering literature. It started by Kuehn and Davidson in 1961 where the use of Lagrange multipliers for the case where all component flow rates were measured. These authors were working in IBM Corporation. They formulated the problem of adjusting flow and temperature measurements on a crude oil distillation tower to satisfy steady-state material and energy balances. Their main idea of the publication was to correct the measurements in order to make them consistent with a good model, commonly the conservation law (Tariq, 2006).

The statistical tests have been constructed for detection of systematic errors. The global and collective tests by Reilly and Carpani in 1963 and later by Ripps in 1965 are based on the fact that the objective function of the data reconciliation problem, at the minimum is distributed as a variable if the measurements are normally distributed about their true values. The magnitude of the data reconciliation objective function is then compared to the tabulated value for a chosen confidence level and for degrees of freedom equal to the number of remaining balances. Reilly and Carpani 1963 formulated the collective test of all the data and the univariate test for constraints, based on normal distribution. They also showed that the individual imbalance errors could be tested against the univariate normal distribution. Ripps then defined test statistics which would directly reveal systematic errors in both imbalances and in adjustments and would avoid trial and error deletion in turn of suspect measurements (Crowe, 1996).

Mah et al. (1976) mentioned that systematic errors in raw process data that errors have already been eliminated from the raw process data by a prior treatment and that the pretreated data are now subject to a reconciliation procedure to yield consistent estimates.

Kretsovalis and Mah (1987) focused on the accuracy of the estimates obtained by data reconciliation. They also used a combinatorial search based on the effect of the variance of measurements on the precision of key variables.

Romagnoli and Stephanopoulos (1980) proposed a systematic strategy for recursively performing data reconciliation by sequentially adding measurements into the calculation. The proposed strategy reduces the size of the data reconciliation problem significantly, even for large-scale chemical processes, is computationally simple and it conforms to the general process of variable monitoring in a chemical plant.

In the presence of unmeasured variables several methods exist to obtain a matrix of only redundant measurement. Crowe et al. (1983) presented a method of directly eliminating the unmeasured quantities in linear constraints, called 'matrix projection', prior to reconciling the measurements. The method was later extended to a bilinear case where some concentrations are measured in streams where the total flow is unmeasured (Crowe, 1986). No nonnegative constraints were imposed on the flow rates and concentrations, nor were upper bounds imposed. When a negative value arose, this was taken as non-statistical evidence that the data contained systematic errors.

Pai and Fisher (1988) proposed a replacement of old derivatives by Crowe so that the rate of convergence can be improved without repeatedly evaluating the derivatives. This method introduced Broyden-type update.

Kalitventzeff and Joris (1987) proposed a procedure for classifying variables and measurements. This is done by permuting rows and columns of the projected matrix corresponding to the Jacobian matrix of the model equations. The mathematical drawback of this procedure is that the Jacobian matrix of a sparse matrix tends to be singular and thus fails to determine the indeterminable variables.

Swartz (1989) used the QR decomposition in the matrix projection. He used the orthogonal factorization in context with successive linearization techniques to eliminate the unmeasured variables.

Madron (1992) proposed classifying measured and unmeasured variables of linear systems according to pre-established criteria of “required” and “non-required.” Unmeasured variables were later ordered from “hardly measured” to “easily measured”. Unoptimal structures are found by means of matrix decomposition and an elaborate column permutation procedure.

Sanchez and Romagnoli (1996) utilised the QR factorization to decompose and solve linear and bilinear data reconciliation problems. The decomposition provides additional insight in identifying structural singularities in the system topology, allowing the problem to decompose into lower dimension sub problems.

Kelly (1998) used different matrix projection techniques and highlighted two relatively simple approaches to determine the matrix projection first introduced by Crowe et al. (1983) to solve data reconciliation problems when unmeasured variables exist. He compared the method of RMIP (Recursive Matrix Inversion by Partition) and MCF (Modified Cholesky Factorization). He then concluded that QR decomposition method was the most efficient of all approaches even if the matrices were sparse.

Romagnoli and Sanchez (2000) introduce the first unified approach of data reconciliation. They bridge the theory and practical aspect through numerous case studies. An introduction to the modern data reconciliation was presented together with in-depth coverage of the relevant theory of data reconciliation.

2.3 Systematic Error Detection

There is always a possibility for measurements to contain systematic error due to miscalibration, instrument malfunction and sensor corrosion. Therefore, it is a normal practice in industry to perform systematic error detection prior to data reconciliation process.

The Global detection test was first proposed by Reilly and Carpani in 1963. They formulated the collective test of all the data and the univariate test for constraints,

based on normal distribution. These data are compared to the optimal value of the objective function in the mathematical model of data reconciliation to an appropriate tabulated chi-square value. They also proposed the univariate constraint test, which examines each residual of the process constraints (Narasimhan and Jordache, 2000).

Ripps in 1965 proposed a method which eliminates the measurement that renders the largest reduction in a test statistics until no test fails. This situation is for the case of multiple systematic errors with serial elimination. In this method, the test function and variance of the resulting system have to be recomputed after deleting a new measurement (Crowe, 1996).

Romagnoli and Stephanopolous (1981) developed a systematic strategy to locate the source of systematic error. The method also rectifies the systematic and biased measurement errors in a chemical process. The proposed strategy reduces the size of the data reconciliation problem significantly, computationally simple and conforms to general process of variable monitoring in a chemical plant.

Global test and measurement test were presented by Almassy and Sztano (1975). They proposed a measurement test that possesses maximum power test when there is only one systematic error in the measurements, and is called the MPT. The MP constraint tests were also proposed by Crowe (1989, 1992).

The Constraint and Nodal Test was presented by Mah et al. (1976). This method requires linear constraints and measured variables. The unmeasured variables must be removed from the constraints. The test is based on the constraint residual values divided by standard deviation of the residuals.

The univariate measurement test, which examines each measurement adjustment, was proposed by Mah and Tamhane (1982). The statistical test based on the adjustment distribution by which first process data are reconciled. The reconciled data are used to examine if a measurement contains systematic error. They named the statistical test as Measurement Test (MT). The MT looks at the adjustment of each measurement to identify and rank the measurement that may be faulty. This method is based on the measurement adjustment divided by its standard deviation.

Narasimhan and Mah (1987) proposed a test for Systematic Error Detection named Generalised Likelihood Ratios. GLR is equivalent to Measurement Test (MT). This is because, GLR is the square of MT. It has the capability to identify the location of the error and differentiate the types of the error such as instrument related error or process model related error.

Rollins and Davis (1992) introduced the Unbiased Estimation Technique. This technique is limited to normally distributed errors, steady state and linear constraints. As a start, global test is conducted and only then the Unbiased Estimation Technique is performed. It is used to detect the number and locations of errors. The approach has also been applied to bilinear system.

Principal Component Analysis (PCA) was introduced by Tong and Crowe (1996). In this technique, a set of correlated variables is transformed into a new set of uncorrelated variables. PCA is a very effective method for multivariate data analysis.

In summary, the methods mentioned earlier are applied to linear system. In order to solve the non-linear problem, a linearization technique is required. However, this will introduce new errors to the system.

Mei et al. (2005) developed an NT-MT combined method based on nodal test (NT) and measurement test (MT) for gross error detection and data reconciliation. The NT-MT combined method makes use of both NT and MT tests and this combination helps to overcome the defects in the respective methods. It also avoids any artificial manipulation and eliminates the huge combinatorial problem that is created in the combined method based on the nodal test in the case of more than one gross error for a large process system.

2.4 Estimation of Variances

Most techniques for process data reconciliation and rectification start with the assumptions that the measurement errors are random variables obeying a known

statistical distribution. Direct method of estimation of variance covariance matrix is applied for a truly steady state problem. For a non-steady state process and for any processes that cannot be assumed as steady state, an indirect method of estimating variance covariance matrix is performed.

Almasy and Mah (1984) derived a method of estimating variance which makes use of the constraint residuals computed from available process data. They introduced a method that is suitable for non-linear problem. They overcome this issue by incorporating additional information of the process namely as linear balance equation. A computation procedure and necessary conditions for the existence of a solution are given. The procedure has been implemented on a computer and several simulation experiments were reported in their paper. However, for application of this method is sufficient spatial redundancy in the measurements.

Five years later, Darouach et al. (1988), proposed a method for estimating a diagonal covariance matrix based on the maximum likelihood estimator, material balance constraints and the statistical properties of their residuals. Keller et al. (1992) applied Darouach et al.'s method to non-diagonal covariance matrix. They presented an analytical algorithm based on the deduced from statistical properties of material balance constraints in order to estimate the covariance matrix of the measurement errors.

Some methods were known to be very sensitive to the outliers in the measurements resulting in error during matrix covariance estimations. Chen et al. (1997) came out with a robust method of estimating covariance matrices. His method was based on M-Estimator method introduced by Huber in 1964. In this method, each vector of the measured variables is given a weight from zero to one which reflects its distance to the median. As a result of that, the effect of the outliers is minimized.

Morad and Svrcek (1999) proposed a new method to measure errors in covariance calculation. The RDCE, Robust Direct Method for Covariance Estimation, which is an extension to Chen et al. (1997) uses M-Estimator to reject the outliers and tune the measured values for deviations from steady state.

Darouach et al (2002) addressed the problem of minimum variance estimation for discrete-time time-varying stochastic systems with unknown inputs. The objective is to construct an optimal filter in the general case where the unknown inputs affect both the stochastic model and the outputs.

2.5 Dynamic Data Reconciliation

Although reconciliation of steady-state process data is routinely applied in industrial practice, the theoretical understanding of the problem and its adequate formulation in a dynamic setting is still not mature.

Data reconciliation for linear dynamic systems was treated by Gertler and Almsay (1973). They showed that the dynamic material balance model can be represented by continuous-state space equations or after discretisation by a sampled input-output representation.

Karjala and Himmelblau (1992, 1994, and 1996) proposed a procedure to overcome auto correlated measurement error and bias in process measurement problem in data rectification. They proposed the use of RNN (Recurrent Neural Network) and EKF (Extended Kalman Filter) in 1992. The recurrent neural-network approaches to nonlinear, dynamic data rectification presented in this and previous work provide an attractive alternative to more traditional approaches.

2.6 Data Reconciliation and Refinery Hydrogen Network

Data reconciliation has been applied to many areas such as petroleum refining, gas processing, pulp and paper industries and pyrolysis reactors in the manufacturing ethylene and propylene (Weiss, 1995).

For refinery hydrogen networks, Bussani et al. (1995) developed Online Reconciliation and Optimisation (ORO) package. The package managed to recover the main errors in plant measurements and obtain true picture of hydrogen plant

performance. The approach used in this technique is Sequential Modular (to solve simulation problem) and Black Box method (to obtain the optimality of the condition).

Suarez-Chavez (2005) had done steady state data reconciliation of a refinery. He did linear data reconciliation of the refinery hydrogen network.

Tariq (2006) developed techniques for linear and bilinear data reconciliation. The method developed uses QR Decomposition for linear problem and unconstrained optimization method for bilinear problem. He used an existing refinery hydrogen networks that had inconsistent data.

2.7 Summary

Data reconciliation has been applied in the industries as a mean of obtaining accurate and consistent data. In the last four decades, it has been developed progressively in chemical engineering field. DATACON (SimSci-Esscor, 2007), SIGMAFINE (OSIsoft Inc, 2007) and VALI (Belsim S.A, 2007) are the examples of data reconciliation commercial software available as a result of this development.

Latest technique of data reconciliation is “bilinear data reconciliation using unconstrained optimisation technique”. Q-R decomposition is applied as the mode of calculations. Among the researchers who first applied this method are Swartz (1989), Sanchez and Romagnoli (1996) and Kelly (1998). Later Suarez-Chavez (2005) and Tariq (2006) applied the same method for different applications.

CHAPTER 3

METHODOLOGY

Data reconciliation is a technique that has been developed to improve the accuracy of measurements by reducing the effect of random errors in the data. Data reconciliation explicitly makes use of process model constraints and obtains estimates of process variables by adjusting process measurements so that the estimates satisfy the constraints.

3.1 General Data Reconciliation Methodologies

The reconciled estimates are expected to be more accurate than the measurements. They are also consistent with the known relationships between process variables as defined by the constraints. Data reconciliation will be effective if there are no systematic errors either in the measurements or in the process model constraints.

Systematic error detection is incorporated with data reconciliation that has been developed to identify and eliminate systematic errors. Thus data reconciliation and systematic error detection are applied together to improve accuracy of measured data.

Generally, in data reconciliation, the first activity involved is defining the scope or the boundary of the system of interest. The input-output of the system, including the streams and unit operations are identified. Then, the variables of the system such as flowrates and composition are classified into two categories, namely measured and unmeasured variables.

The next step is systematic error detection of the measurements, reconciliation of free measured variables producing bias-free variables and elimination of unmeasured variables are carried out. This exercise requires a process model, for example material balance, to be fed.

Figure 3.1 below shows the general data reconciliation activities. Identification and definition of the problem are the first activity in the process. Next, available plant data are gathered and analysed. This is followed by classification of the variables. The main activity of is to perform linear data reconciliation. In order to do that, QR decomposition need to be done. The next activity is, estimation of unmeasured variables or in this case flowrates. Process models (material balance) are used as commonly used as constraints. The presence of systematic error can be detected at the last stage of the process.

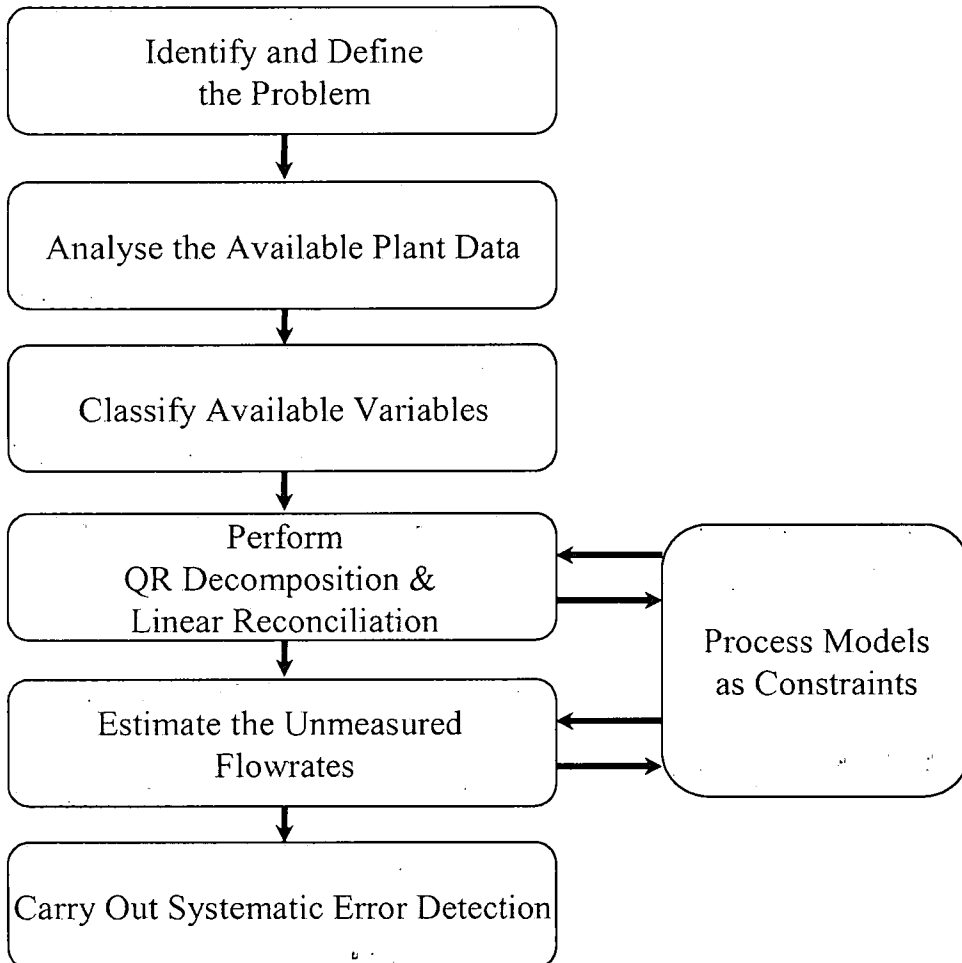


Figure 3.1: General Data Reconciliation

Data reconciliation can be categorised as either linear or non linear data reconciliation as well as steady state or dynamic data reconciliation. For this project, steady state linear data reconciliation is developed. Figure 3.2 shows graphically the categories of data reconciliation.

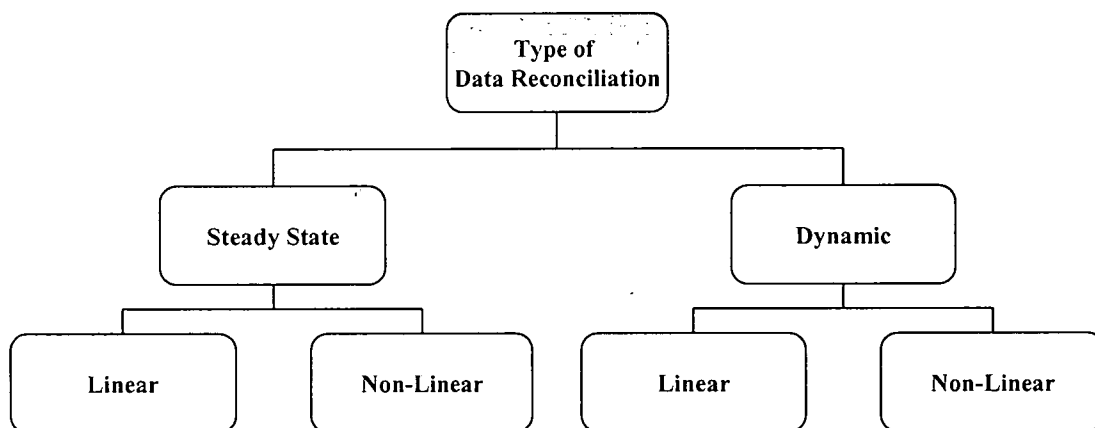


Figure 3.2: Classes of Data Reconciliation

Variables are further classified as redundant or non redundant measured variables and either observable or unobservable unmeasured variables. Figure 3.3 shows the classes of variables. Redundant variable is a measured process variable that is over-determined if it can also be computed from the balance equations and the rest of the measured variables. On the other hand, non-redundant variable is a measured variable that cannot be computed from the balance equations and the rest of the measured variables.

On top of that, the unmeasured variables can be grouped as observable or non-observable variables. Observable variable is unmeasured variable that is determinable if it can be evaluated from the available measurements using the balance equations. This is in contrast with unobservable variable that is indeterminable if it cannot be evaluated from the available measurements using the balance equations (Romagnoli and Sanchez, 2000).

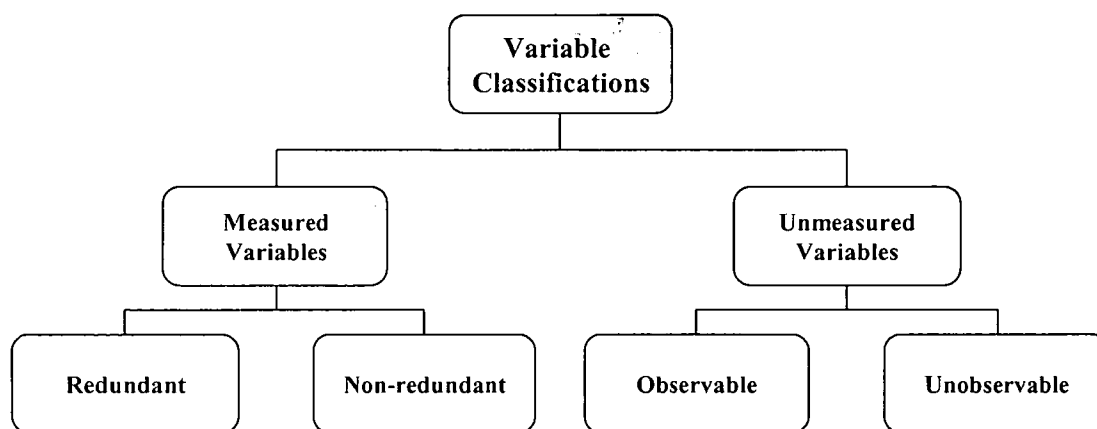


Figure 3.3: Variable Classifications

Unmeasured variables can be eliminated from the reconciliation model by matrix projection. A reduced model is obtained which can be used to reconcile the measured variables.

3.2 Linear Data Reconciliation

Linear data reconciliation is the simplest data reconciliation which involves a linear model. Vector notation is normally used because it provides a compact representation and allows powerful concepts from linear algebra and matrix theory to be exploited.

3.2.1 Linear System with Measured and Unmeasured Variables

A system, which consists of measured and unmeasured variables are usually solved in two sub-problems. The first will be the reconciliation of the measured variables and the second is the determination of the unmeasured variables.

The variables are classified into two sets. They are the vector x of measured variables and the vector u of unmeasured variables. The measurement model is still given by

$$y = x + \varepsilon \quad (3.1)$$

where y is a vector of n measurements, x is the corresponding vector of true values of the measured variables and ε is the vector of unknown random errors. While the objective function is given by

$$\text{Min } x (y-x)^T \Sigma^I (y-x) \quad (3.2)$$

where Σ is variance-covariance matrix. It contains information about the accuracy of the measurements and the correlations between them. Equation 3.2 is equivalent to normal least square problem used in regression.

$$\text{Min } \Sigma (y-x)^2 \quad (3.3)$$

In reality, some measurements are more accurate than the others. In order to consider the accuracies, weighted least square objective as a more general criterion given by

$$\text{Min } \Sigma W_i (y-x)^2 \quad (3.4)$$

Equation 3.3 can also be written in the form of

$$\text{Min } x (y-x)^T W (y-x) \quad (3.5)$$

By replacing the W (weightage) term with Σ^I in equation 3.5, the objective function of linear system with measured and unmeasured variables can be formed. In a nutshell, the objective function given in equation 3.2 has a similar meaning with normal least square problem but it is written in different form to accommodate the variance-covariance matrix, Σ (information about the accuracy of the measurements).

The estimates are also required to satisfy the constraints (3.1). However, the constraints have to be recast in terms of both the measured (x) and unmeasured (u) variables. The constraints can be represented in general by

$$A_x x + A_u u = 0 \quad (3.6)$$

where A_x corresponds to the measured variables with $m \times n$ dimensions, A_u corresponds to the unmeasured variables with $m \times p$ dimensions, m is number of unit operations involved, n is number of measured variables and p is number of unmeasured variables.

Each row corresponds to a constraint. It can be easily verified that for a flow reconciliation problem, the elements of each row of matrix A are either +1, -1 or 0, depending on whether the corresponding stream flow is respectively an input, output or not associated with the process unit for which the flow balance is written.

The unmeasured variables, u , has to be eliminated from equation (3.6) using suitable linear combinations of the constraints. This is equivalent to pre-multiplying the constraints by a matrix P , also known as a projection matrix. The matrix P should satisfy the property

$$PA_u=0 \quad (3.7)$$

Pre-multiplying equation (3.7) by matrix P , we get the reduced set of constraints involving only measured variables as

$$PA_x x=0 \quad (3.8)$$

The number of columns of P should clearly be equal to the number of constraints, m . As many independent rows as possible are constructed for P which satisfy the property shown in equation (3.8).

3.2.2 QR Decomposition

In linear algebra, the QR decomposition is a decomposition of a matrix into an orthogonal and a triangular matrix. The QR decomposition is often used to solve the linear least squares problem. Least Square method is a mathematical optimisation technique that when given a series of measured data, attempts to find a function which closely approximates the data. This process is also known as best fitting. It attempts to

minimize the sum of the squares of the ordinate differences or residuals between points generated by the function and corresponding points in the data.

The QR decomposition of a matrix is a decomposition of general rectangular matrix A , defined as:

$$AP = QR \quad (3.9)$$

where Q is an orthogonal matrix, R is an upper-triangular, and P is a permutation matrix. Permutation matrix is a matrix that has exactly one entry 1 in each row and each column and 0's elsewhere.

A QR decomposition object is constructed from a general rectangular matrix. Once an object exists, there are a variety of member functions that can be used to extract its components and to use the object for various operations.

The QR decomposition of a matrix is a decomposition of the matrix into an orthogonal and a triangular matrix. The QR decomposition is often used to solve the linear least squares problem. The QR decomposition is also the basis for a particular eigenvalue algorithm, the QR algorithm.

There are three methods for QR decomposition namely as Givens transformations, which is based on plane rotations, Gram-Schmidt orthogonalisation and Householder transformations (Olver, 2006).

The following are the brief discussions on Givens transformation and Gram-Schmidt orthogonalisation. As this project uses Householder transformations, a detail discussion on this method is given in the next section.

Givens transformations technique computes QR decompositions using a series of Givens rotations. Each rotation zeros an element in the subdiagonal of the matrix, forming the R matrix. The concatenation of all the Givens rotations forms the orthogonal Q matrix. In practice, Givens rotations are not actually performed by building a whole matrix and doing a matrix multiplication. A Givens rotation

procedure is used instead which does the equivalent of the sparse Givens matrix multiplication, without the extra work of handling the sparse elements. The Givens rotation procedure is useful in situations where only a relatively few off diagonal elements need to be zeroed, and is more easily parallelized than Householder transformations.

On the other hand, Gram-Schmidt orthogonalisation method uses the concept of Gram-Schmidt process. The Gram-Schmidt process is one of the premier algorithms of applied and computational linear algebra. Gram-Schmidt process is a method for orthogonalising a set of vectors in an inner product space, most commonly the Euclidean space.

The Householder transformation has been selected for the Linear Data Reconciliation Program developed in this work. It is chosen because it is stable and easy to code in Visual Basic Programming (Gunter and Van De Geijn, 2001).

3.2.2.1 Householder Transformation

Alston Scott Householder introduced the Householder transformation in 1958. It can be used to solve QR decomposition as described in the QR algorithm of a matrix, bringing the matrix A to upper Heisenberg matrix form with a finite sequence of orthogonal similarity transforms. Over general inner product spaces, this is known as the Householder operator.

In mathematics, a Householder transformation in 3-dimensional space is the reflection of a vector in a plane. In general Euclidean space it is a linear transformation that describes a reflection in a hyper-plane (containing the origin).

By Householder transform we mean a transformation matrix that has the form of

$$H = I - 2 \frac{uu^T}{u^T u} \quad (3.10)$$

where u is a nonzero vector. Transformation changes an arbitrary vector v into a multiple of a unit vector.

$$Hv = \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \alpha e_i \quad (3.11)$$

We get the transformation vector u from the formula

$$u = v - \alpha e_i, \quad \alpha = \pm \|v\| \quad (3.12)$$

There is no need to save the whole matrix H – vector u and constant α define the transformation. The Householder is used in QR Decomposition by transforming repetitively each column of A to get upper triangular matrix

$$R = H_n H_{n-1} \dots H_1 A \quad (3.13)$$

Matrix H_1 is the transformation matrix of the first column of A , matrix H_2 the transformation matrix for the last $n-1$ elements of the second column and so on. The product of $H_1, H_2 \dots H_n$ forms the orthogonal matrix Q .

$$Q = H_1 H_2 \dots H_n \quad (3.14)$$

3.2.2.1.1 Example of Calculations Using Householder Transformation

The following is the solved example of a matrix, A , using Householder transformation of QR decomposition.

Assuming matrix A is given as the following.

$$A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$$

First, a reflection that transforms the first column of matrix A need to be found.

$$a_1 = (12, 6, -1)^T \text{ to } \|a_1\|e_1 = (14, 0, 0)^T$$

Now, $u = x - \alpha e_1$,

$$\text{And } v = \frac{u}{\|u\|},$$

Here, $\alpha = 14$ and $x = a_1 = (12, 6, -1)^T$

Therefore,

$$u = (-2, 6, -4)^T \text{ and } v = \frac{1}{\sqrt{14}}(-1, 3, 2)^T,$$

and then

$$\begin{aligned} H_1 &= I - \frac{2}{\sqrt{14}\sqrt{14}} \begin{pmatrix} -1 \\ 3 \\ -2 \end{pmatrix} \begin{pmatrix} -1 & 3 & -2 \end{pmatrix} \\ &= I - \frac{1}{7} \begin{pmatrix} 1 & -3 & 2 \\ -3 & 9 & -6 \\ 2 & -6 & 4 \end{pmatrix} \\ &= \begin{pmatrix} 6/7 & 3/7 & -2/7 \\ 3/7 & -2/7 & 6/7 \\ -2/7 & 6/7 & 3/7 \end{pmatrix} \end{aligned}$$

Now:

$$H_1 A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & -49 & -14 \\ 0 & 168 & -77 \end{pmatrix}$$

At this point, a triangular matrix is almost found. Only (3, 2) entry need to be zeroed.

Take the (1, 1) minor, and then apply the process again to

$$A' = M_{11} = \begin{pmatrix} -49 & -14 \\ 168 & -77 \end{pmatrix}$$

By the same method as above, the matrix of the Householder transformation is obtained after performing a direct sum with 1 to make sure the next step in the process works properly.

$$H_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -7/25 & 24/25 \\ 0 & 24/25 & 7/25 \end{pmatrix}$$

Now,

$$Q = H_1 H_2 = \begin{pmatrix} 6/7 & -69/175 & 58/175 \\ 3/7 & 158/175 & -6/175 \\ -2/7 & 6/35 & 33/35 \end{pmatrix}$$

$$R = H_2 H_1 A = Q^T A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & -35 \end{pmatrix}$$

The matrix Q is orthogonal and R is upper triangular, so $A = QR$ is the required QR-decomposition.

3.2.3 Construction of Projection Matrix

There are several different matrix methods for the construction of the projection matrix. However, probably the most efficient method is to use the QR factorisation of the matrix $\mathbf{A}\mathbf{u}$. Such a method was first applied to data reconciliation solving by Stoks et al. (1994) and recently utilised by Sanchez and Romagnoli (1996) to decompose and solve linear and bilinear data reconciliation problems.

A QR decomposition of a real square matrix \mathbf{A} is a decomposition of \mathbf{A} as

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \quad (3.15)$$

where \mathbf{Q} is an orthogonal matrix. An orthogonal matrix \mathbf{Q} is such that it obeys

$$\mathbf{Q}^T\mathbf{Q}=\mathbf{D}_i \quad (3.16)$$

where \mathbf{D}_i is diagonal matrix and \mathbf{R} is an upper triangular matrix

More generally, we can factor a complex $m \times n$ matrix (with $m \geq n$) as the product of an $m \times n$ unitary matrix and an $n \times n$ upper triangular matrix, where m is the number of row of matrix \mathbf{A} and n is the number of column of matrix \mathbf{A} .

If rank of matrix $\mathbf{A}\mathbf{1}$, r is lesser than the number of column, n , ($r < n$), the QR decomposition approaching a failure. This shows that at least one entry in \mathbf{R} is zero and QR decomposition does not produce orthonormal basis for \mathbf{R} . An orthonormal set must be linearly independent, and so it is a vector space basis for the space it spans. Such a basis is called an orthonormal basis (Rowland, 2000).

In the case when number of independent column $\mathbf{A}\mathbf{u}$ is less than the number of columns, QR decomposition must be modified by multiplying matrix $\mathbf{A}\mathbf{1}$ with permutation matrix, \mathbf{P} which is basically an identity matrix (contains only 0 and 1 elements only) that permutes the row in order to avoid zeros in the diagonal.

Therefore,

$$A_1 P = [Q_1 \ Q_2] \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \quad (3.17)$$

Where Q_1 is $(m \times r)$ matrix subset of matrix Q , Q_2 is $[m \times (m-r)]$ matrix subset of Q , R_{11} is $(r \times r)$ matrix subset of matrix R , R_{12} is $(r \times (n-r))$ matrix subset of matrix R and P is permutation matrix (adjusted identity matrix).

The graphical explanation of the above statements is given in Figure 3.4 below.

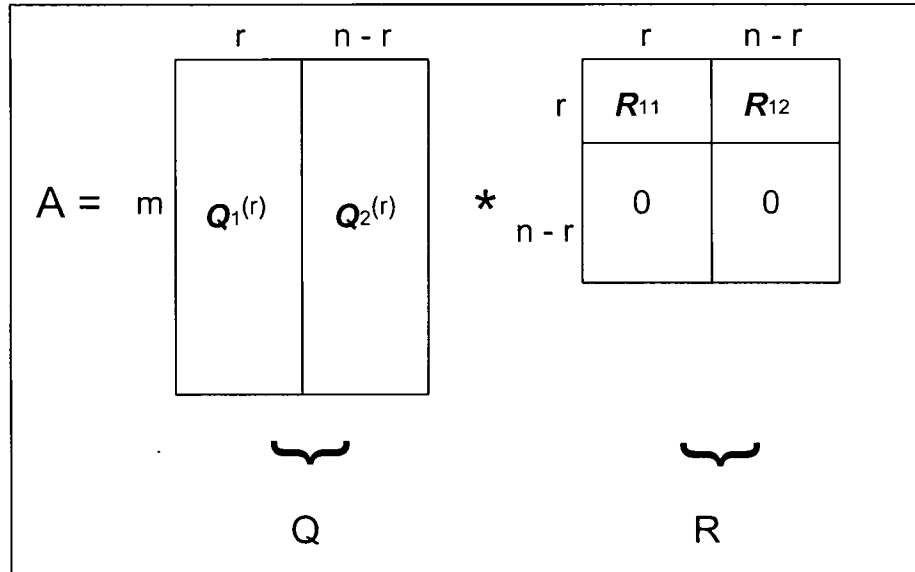


Figure 3.4: Subsets of Q and R , Q_1 , Q_2 , R_1 and R_2

3.2.4 Reconciliation of Measured Variables

The reduced data reconciliation problem is to minimise (3.2) and subject to constraint (3.17). The reconciled measured variables are calculated by

$$\hat{x} = y - \Sigma(PA_x)^T [(PA_x)^T \Sigma(PA_x)^T]^{-1} (PA_x) y \quad (3.18)$$

Using equation (3.8), we can now substitute for x in equation (3.18) and obtain the estimation for the variable, \hat{x} (Narasimhan and Jordache, 2000).

3.2.5 Determination of Unmeasured Variables

In order to estimate the unmeasured variables, the values of x in equation (3.8) are to be substituted to equation (3.19) below.

$$\mathbf{u} = -\mathbf{A}_u^T \mathbf{A}_u (\mathbf{A}_x \mathbf{x}) \quad (3.19)$$

However, to solve the unmeasured variables for non-linearly independent \mathbf{A}_u , the equation (3.19) is modified to equation (3.20) below.

$$\mathbf{u} = -\mathbf{R}_1^{-1} (\mathbf{Q}_1^T \mathbf{A}_x \mathbf{x} + \mathbf{R}_2 \mathbf{unr}) \quad (3.20)$$

where \mathbf{R}_1 , \mathbf{R}_2 and \mathbf{Q}_1 as per defined in Figure 3.4, and \mathbf{unr} is an arbitrary set of assigned values (Narasimhan and Jordache, 2000).

3.3 Systematic Error Detection

As mentioned in Chapter 1, there are two main categories of errors in measurements. The first is random error and the other is systematic error. In the previous section, the random errors are removed from the measurement readings. In this section, we will discuss the methodology of identifying and removing the systematic errors.

There are two major types of systematic errors. One is related to the instrument performance and includes measurement bias, drifting, miscalibration and total instrument failure. Any comprehensive systematic error detection strategy should possess the following capabilities (Narasimhan and Jordache, 2000).

The first criterion is the ability to detect the presence of one or more systematic errors in the data (the detection problem). Next is the ability to identify the type and location of the systematic error (the identification problem). On top of that, good systematic error detection must have the capability to locate and identify multiple systematic errors which may be present simultaneously in the data (the multiple systematic error

identification problems). Lastly, it must be capable to estimate the magnitude of the systematic errors (the estimation problem).

However, the program developed in this project is capable of fulfilling the first requirement only that is detecting the presence of systematic errors only. This component of a systematic error detection strategy simply attempts to answer the question of whether systematic errors are present in the data or not. It does not provide any information on the number of systematic errors, their types or locations. This is a purely deterministic method.

3.3.1 Global Test for Systematic Error Detection

The basic principle in systematic error detection is derived from the detection of outliers in statistical applications. The random error inherently present in any measurement is assumed to follow a normal distribution with zero mean and known variance. The normalized error, which is the difference between the measured value and the expected mean value divided by its standard deviation, follows a standard normal distribution. Most normalized errors fall inside a $(1-\alpha)$ confidence interval at a chosen level of significance α . Any value of normalized error which falls outside that confidence region is declared an outlier or a systematic error.

In this project, the statistical technique is used for detecting systematic errors. It is based on hypothesis testing. In systematic error detection case, H_0 , null hypothesis, is that no systematic error is present, and H_1 , alternative hypothesis, is that one or more systematic errors are present in the system.

Global testing uses the test statistic proposed by Ripps in 1965 (Narasimhan and Jordache, 2000). It is given by

$$\gamma = \mathbf{r}^T \mathbf{V}^{-1} \mathbf{r} \quad (3.21)$$

where \mathbf{r} is the vector of balance residuals, which is given by

$$\mathbf{r} = \mathbf{A}\mathbf{y} - \mathbf{c} \quad (3.22)$$

Furthermore, in equation (3.17), \mathbf{A} is the linear constraint matrix, in this case, Matrix $\mathbf{A1}$ and \mathbf{c} contains known coefficients and for linear flow processes, $\mathbf{c} = 0$. \mathbf{V} is variance-covariance matrix when random error is normally distributed with variance-covariance matrix Σ . \mathbf{V} is given as

$$\mathbf{V} = \text{cov}(\mathbf{r}) = \mathbf{A}\Sigma\mathbf{A}^T \quad (3.23)$$

The value of γ can be verified to be equal to the sum square of the differences between the reconciled and measured values. (Narasimhan and Jordache, 2000). Therefore, in this dissertation, the value of γ is calculated by the sum square of the differences between reconciled and measured values.

Under H_0 , the above statistic follows a χ^2 distribution with ν degrees of freedom, where ν is the rank of matrix \mathbf{A} . If the test criterion chosen is $\chi^2_{\alpha\nu}$ (where it is the critical value of chi-square distribution at the chosen α level of significance) then H_0 is rejected and a systematic error is detected if $\gamma \geq \chi^2_{\alpha\nu}$. $\chi^2_{\alpha\nu}$ also be written as critical value of global testing, γ_c .

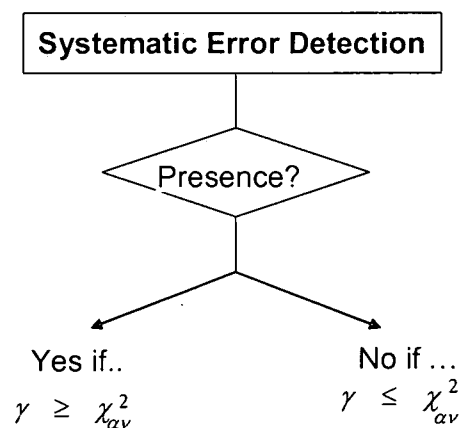


Figure 3.5: Systematic Error Detection Technique Using Global Test

3.4 Algorithms and Flowcharts of Linear Data Reconciliation Program

A sequence of instructions is called an algorithm. Algorithms are a fundamental part of computing. In this project, two programs have been written. The first program is “QR Solver”, which is used to solve QR Decomposition of matrices. The other program is “Linear Data Reconciliation” Program, which is the main objective of this project. Both programs are developed in Visual Basic environment in Microsoft Excel.

Figure 3.6 below is the flowchart of “QR Solver”. It is a simplified flowchart as the process or the mathematical part of the program is similar to the methodologies described in section 3.2.2.

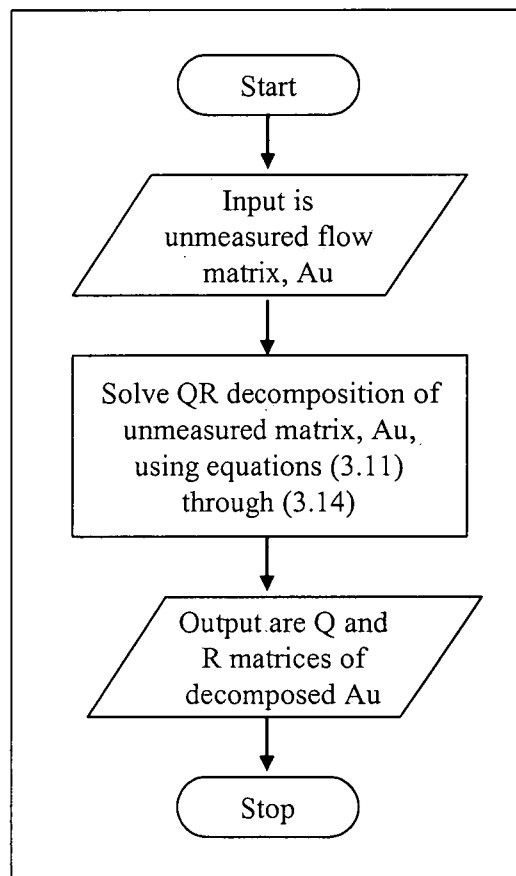


Figure 3.6: Flowchart of “QR Solver”

Figure 3.7 below shows the flowchart of “Linear Data Reconciliation” program. The process part of the flowchart is similar to the methodologies discussed in previous

sections. However, the flowchart below enables the user to understand the algorithm easier.

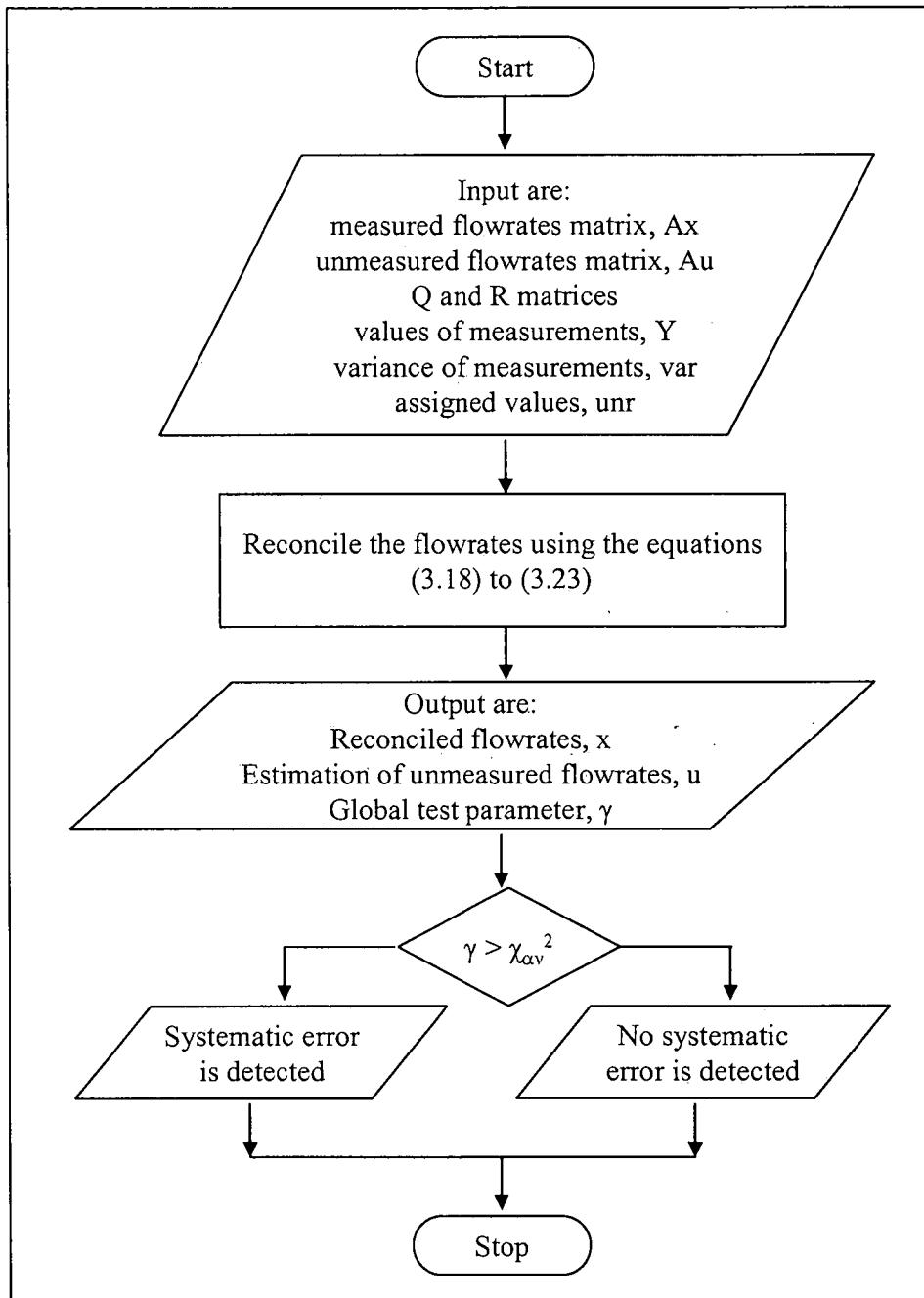


Figure 3.7: Flowchart of "Linear Data Reconciliation" Program

3.5 Summary of Methodology

In the previous sections, a systematic methodology is developed for linear data reconciliation and systematic error detection. The method is programmed in a Visual Basic Programming in Microsoft Excel application.

The program uses all the mentioned methodologies and can be applied to any linear reconciliation problem. It started with reconciliation of measurements and then completed with systematic error detection. The results obtained from the program are the reconciled values of measured variables and the estimation of unmeasured variables.

All the mathematical equations are incorporated in the program and verified or cross-checked with MATLAB® (MathWorks, 2007).

CHAPTER 4

CASE STUDY

Data reconciliation and systematic error detection program developed earlier was applied to a refinery hydrogen network. Nevertheless, the program is capable for any linear system. The only requirement is the material balance of the system for each unit of interest. The following sections describe the case study and the discussion on the results obtained.

4.1 Description of the Refinery Hydrogen Network

This existing refinery hydrogen network encountered a problem of imbalance or inconsistent material balance. The study was initiated in 2004 and data reconciliation was proposed (Tariq, 2006). This problem caused difficulties for future projects such as future plant debottlenecking, hydrogen recovery and optimisation in its consumption and production. The error-free flows are desired to give true picture of the network.

This particular hydrogen network is just like any other modern refineries' hydrogen networks. It has two hydrogen production units and seven units that consume hydrogen. The two production units are an independent hydrogen producer and a naphtha catalytic reformer unit. Catalytic reformer produces high purity hydrogen as by-product along with light gases and liquefied petroleum gas (LPG).

For a better study of the whole system, the network is divided into small sections. Therefore, in this study, the data reconciliation is applied on only one producer and one consumer. The producer chosen for this exercise is naphtha catalytic reformer denoted as Unit 400. The hydrogen produced from this unit has the purity of 85 to 88 mole%. This gas goes to the hydrogen header and then distributed to its consumers. Also entering the header is the hydrogen from the second catalytic reformer of the system, Unit 1400 with the hydrogen purity of 85 to 88 mole% too.

In this case study, the consumer selected is a diesel desulphurisation unit, which is labeled as Unit 300. This diesel desulphurisation unit consumes hydrogen to remove the sulphur compound impurities. Its hydrogen input is from the hydrogen header mentioned earlier.

The exercise involves line tracing of the PFD to collect relevant streams of the network, generation of mass balance equations around the units and application of linear data reconciliation method.

Figure 4.1 shows the P&ID Hydrogen network used in this case study. The simplified PFD for the analysis and simplified PFD for data reconciliation are further shown in Figure 4.2 and Figure 4.3 respectively. Table 4.1 below tabulates the details of the equipment involved in the case study:

Table 4.1: Unit Description of Refinery Hydrogen Network

Unit	Description
C-301	Hydrogen booster compressor
C-401	Compressor
C-402	Hydrogen storage Compressor
D-302	Reactor product separator
D-304	Booster compressor knock-out drum
D-305	Preflash drum
D-404	Reactor product separator
D-407	Hydrogen storage
D-408	After cooler knock-out drum
D-409	Storage compressor knock-out drum
D-410	MP flash drum
D-418	Chloride absorber
F-304	Vane filter
M	Mixers
S	Splitters
P	Pumps/Compressors

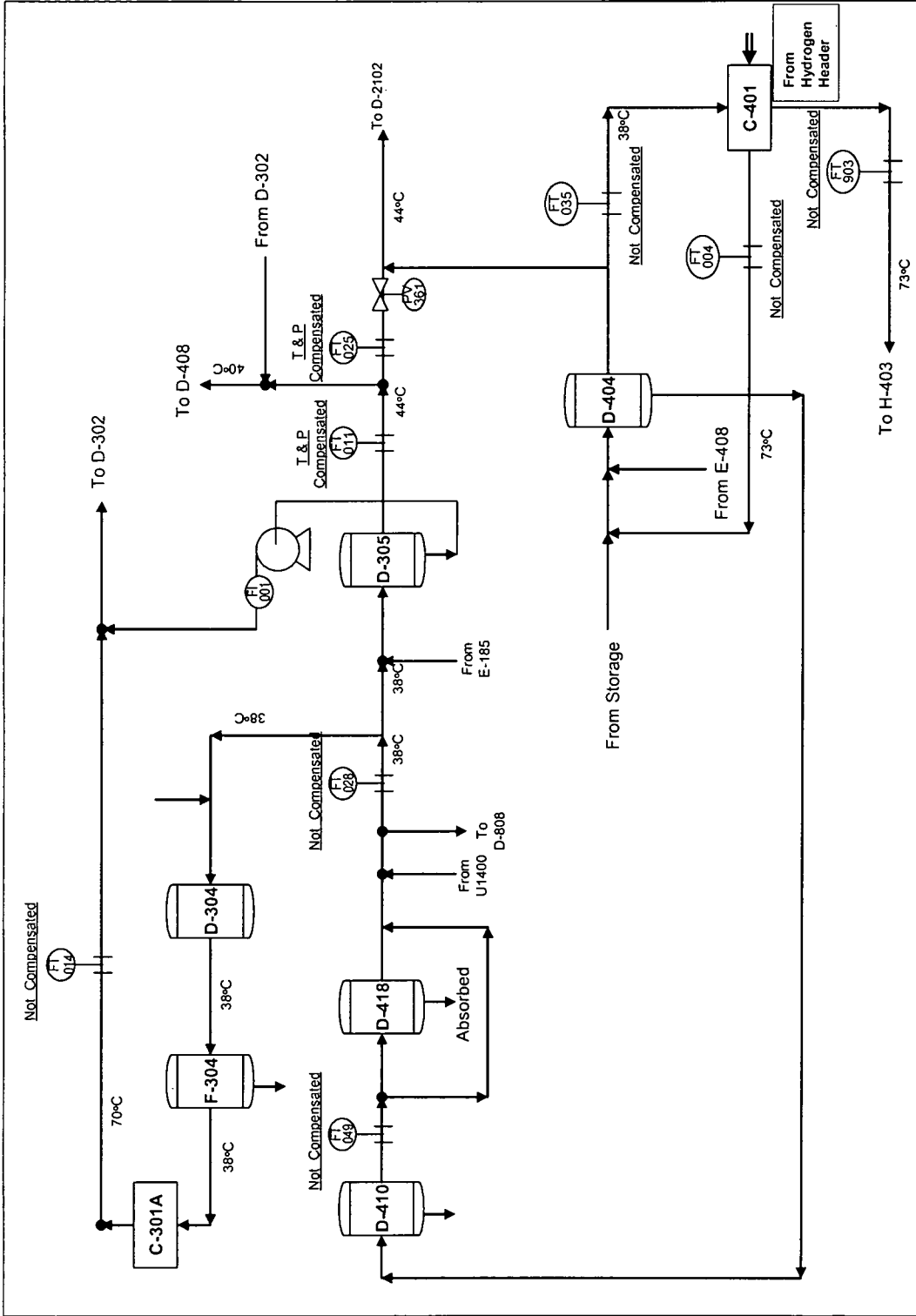


Figure 4.1: Simplified Hydrogen Network for Analysis

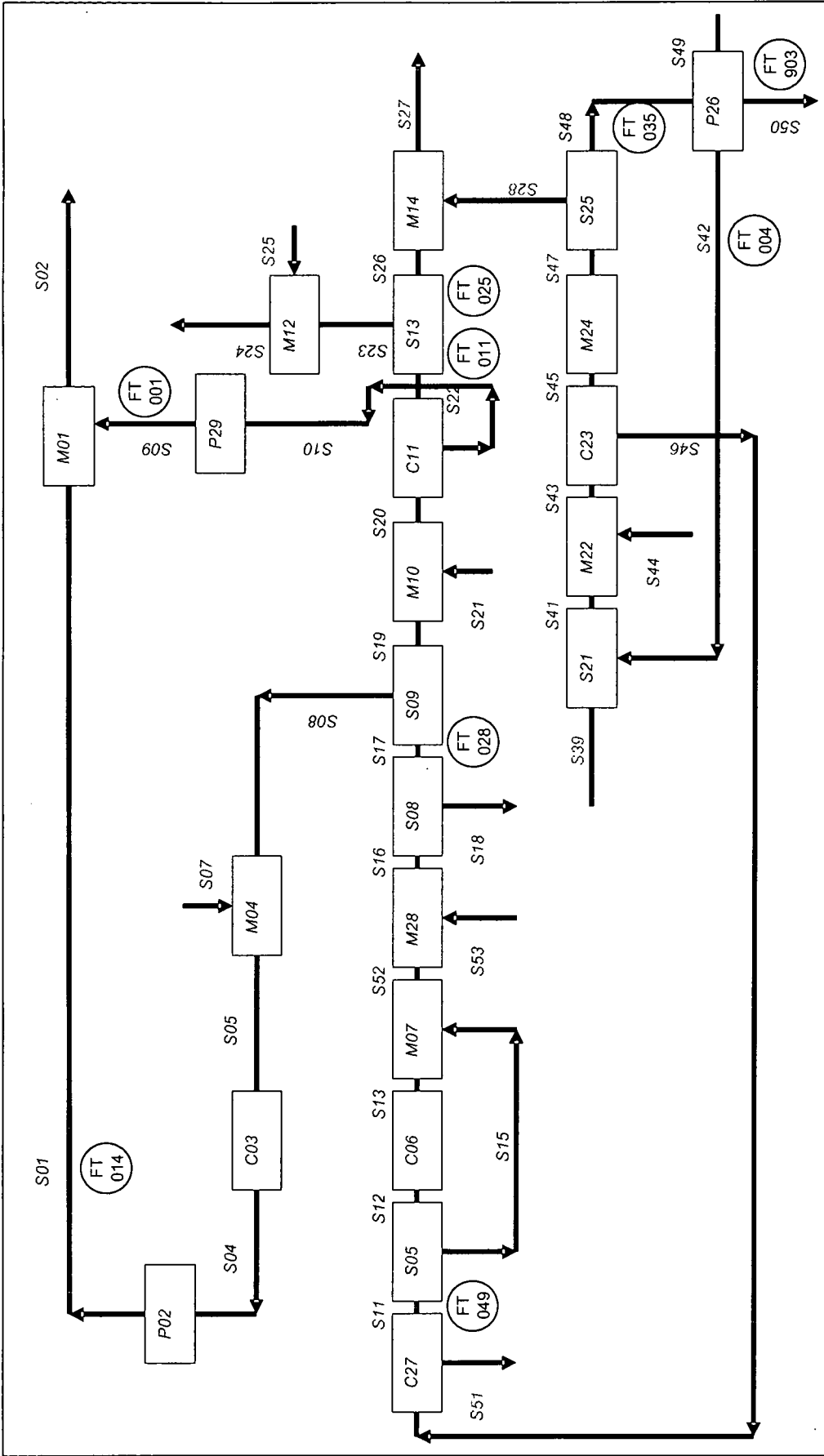


Figure 4.2: Simplified PFD for Data Reconciliation

4.2 Data Gathering

Flow rates data was obtained in February 2004 (Suarez-Chavez, 2005). Initially, the measurement data available was only from nine streams. The measured streams are indicated with 'FTs' or flow transmitters in Figure 4.1 through Figure 4.3. In order to obtain additional information and make the system determinable, process simulator HYSYS® v 3.2 (Suarez-Chavez, 2005) was used. The simulation results in the following data which produces 27 measured variables instead of nine plant data. The total number of streams in this work is 39. There are 27 measured streams and 12 unmeasured streams. Four streams are specified as zero flow rates, which includes the vent of compressor C-301, liquid stream from filter F-304, chloride absorbed from D-418 and recycle stream after D-404.

Table 4.2 and Table 4.3 show the lists of stream for both measured and unmeasured flowrates respectively of the oil refinery hydrogen network system.

Table 4.2: Lists of Measured Streams

No.	Measured Streams
1	S01
2	S02
3	S04
4	S05
5	S07
6	S08
7	S09
8	S10
9	S11
10	S12
11	S13
12	S15
13	S16
14	S17
15	S18
16	S19
17	S20
18	S21
19	S22

20	S23
21	S26
22	S42
23	S48
24	S49
25	S50
26	S52
27	S53

Table 4.3: Lists of Unmeasured Streams

No.	Unmeasured Streams
1	S24
2	S25
3	S27
4	S28
5	S39
6	S41
7	S43
8	S44
9	S45
10	S46
11	S37
12	S51

4.2.1 Material Balance of Measured Variables Ax and Unmeasured Variables Au

Ax is a matrix of dimension $m \times n$ and each row of Ax corresponds to a constraint. It can be easily verified that for a flow reconciliation problem, the elements of each row of matrix A are either +1, -1 or 0, depending on whether the corresponding stream flow is input, output or, respectively, not associated with the process unit for which the flow balance is written. Each row represents each unit of the system. In this case study the Ax has the dimension of 23×27 (23 units and 27 measured streams).

For example, for the first unit C301A, Hydrogen booster compressor, the element of matrix Ax can be written in the first row as the following:

$$Ax \text{ (Matrix } 1 \times n) = [-1 \ 0 \ 1 \ 0]$$

It shows that stream S04 is the input flow to Hydrogen booster compressor and stream S01 is the output flow of it. The other 25 streams are not associated to P02.

The same procedure will be done to the unmeasured variables. The only difference is the number of streams is 12 for the unmeasured streams. Au has the dimension of 23×12 comprising 23 units and 12 unmeasured streams.

The Ax and Au matrix for this case study are shown in Figure 4.4 and 4.5 respectively.

$Ax =$	S01	S02	S04	S05	S07	S08	S09	S10	S11	S12	S13	S15	S16	S17	S18	S19	S20	S21	S22	S23	S26	S42	S48	S49	S50	S52	S53	
M01	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P02	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C03	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M04	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S05	0	0	0	0	0	0	0	0	1	-1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C06	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M07	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0
S08	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
S09	0	0	0	0	0	-1	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
M10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0
C11	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	0	0	0	0
M12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
C13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0
M14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
S21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
M22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	1	-1	0	0	
C27	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
M28	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
P29	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4.3: Elements of Measured Variable Matrix Ax

$Au =$	S24	S25	S27	S28	S39	S41	S43	S44	S45	S46	S37	S51
M01	0	0	0	0	0	0	0	0	0	0	0	0
P02	0	0	0	0	0	0	0	0	0	0	0	0
C03	0	0	0	0	0	0	0	0	0	0	0	0
M04	0	0	0	0	0	0	0	0	0	0	0	0
S05	0	0	0	0	0	0	0	0	0	0	0	0
C06	0	0	0	0	0	0	0	0	0	0	0	0
M07	0	0	0	0	0	0	0	0	0	0	0	0
S08	0	0	0	0	0	0	0	0	0	0	0	0
S09	0	0	0	0	0	0	0	0	0	0	0	0
M10	0	0	0	0	0	0	0	0	0	0	0	0
C11	0	0	0	0	0	0	0	0	0	0	0	0
M12	-1	1	0	0	0	0	0	0	0	0	0	0
C13	0	0	0	0	0	0	0	0	0	0	0	0
M14	0	0	-1	1	0	0	0	0	0	0	0	0
S21	0	0	0	0	1	-1	0	0	0	0	0	0
M22	0	0	0	0	0	1	-1	1	0	0	0	0
C23	0	0	0	0	0	0	0	0	-1	-1	0	0
M24	0	0	0	0	0	0	0	0	1	0	-1	0
S25	0	0	0	-1	0	0	0	0	0	0	1	0
P26	0	0	0	0	0	0	0	0	0	0	0	0
C27	0	0	0	0	0	0	0	0	0	0	1	-1
M28	0	0	0	0	0	0	0	0	0	0	0	0
P29	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.4: Elements of Unmeasured Variable Matrix, Au

4.3 Linear Data Reconciliation for Measured Variables

The simulation data was fed to the linear data reconciliation program developed using VBA Excel. The program produces the following linearly reconciled flow rates. Figure 4.6 shows the graphical user interface (GUI) developed for Linear Data Reconciliation Solver. The end users are required to fill some data before the program solve linear data reconciliation. Prior to execution of this program, all data must be made available in Microsoft Excel Spreadsheet.

The first step of using this program is; the users must give input cell of the unmeasured variables matrix, Au . The solver will give the Q and R matrices of the unmeasured matrix in the Microsoft Excel® spreadsheet at the user-specified cell at the output box.

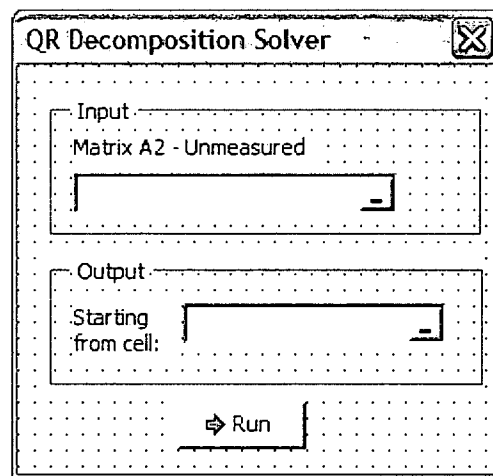


Figure 4.5: QR Decomposition Solver Interface

Figure 4.7 shows the interface of the Linear Data Reconciliation Solver. Similar to the QR Decomposition Solver form, the end users are required to fill in the boxes of required the required data. There are seven input boxes to be filled by end users in this form. By clicking the RUN button, the linearly reconciled data can be obtained. Table 4.4 below shows the comparisons from this Linear Data Reconciliation Solver and the simulation results.

Linear Data Reconciliation Solver

Input Matrix A1	Input Permuted A2 Matrix
Input Matrix Q	Input Matrix R
Input Y = measurement values	Input var - Variance of Measurements
Input unr - Assigned Values	Output Starting from cell:

Run

Figure 4.7: Linear Data Reconciliation Solver Interface

Table 4.4: Linearly Reconciled Flowrates

Stream No.	Stream Name	Measured Flow Rates (ton/hr)	Reconciled Flow Rates (ton/hr)	Variance (ton/hr)
1	S01	1.2190	1.2190	0.0000
2	S02	27.6300	27.6300	0.0000
3	S04	1.2190	1.2206	-0.0016
4	S05	1.2190	1.2021	0.0169
5	S07	0.0100	0.03324	-0.0232
6	S08	1.2109	1.5314	-0.3205
7	S09	26.4120	26.4120	0.0000
8	S10	26.4120	26.4120	0.0000
9	S11	2.4760	2.7048	-0.2288
10	S12	2.2280	2.4570	-0.2290
11	S13	2.2280	1.9992	0.2288
12	S15	0.2476	0.2478	-0.0002
13	S16	3.9760	3.0867	0.8893
14	S17	3.9330	4.4408	-0.5078
15	S18	0.0437	0.0945	-0.0508
16	S19	2.7240	2.7240	0.0000
17	S20	28.5500	26.9830	1.5670

18	S21	25.4160	26.9830	-1.5670
19	S22	1.7280	1.7280	0.0000
20	S23	0.0700	0.0700	0.0000
21	S26	1.6580	1.6580	0.0000
22	S42	15.3320	15.3320	0.0000
23	S48	0.3170	0.3170	0.0000
24	S49	16.9100	2.1060	14.804
25	S50	1.8950	1.8950	0.0000
26	S52	2.4760	2.2470	0.2290
27	S53	1.4570	1.4570	0.0000

As shown in Table 4.5, stream S49 or stream number 24 has a big variance. There is possibility that it is due to the nature of the stream which is a venting line. The purging activity is only done intermittently. The assigned values given by the plant personnel are valid when the purging is done. Due to the scope of this work, the program is only capable of detecting the presence of systematic error of the system but unable to identify the location of the stream.

4.4 Determination of Unmeasured Variables

From the results obtained, the 12 unmeasured flowrates can be obtained by assigning values to four key streams, $S39 = S44 = S51 = 0$, and $S24 = 0.0701$. Table 4.5 below shows the calculated values of the unmeasured flowrates. The calculations are done by the developed program with the input of the four key streams.

Table 4.5: Calculated Unmeasured Flowrates

Stream	Determined Unmeasured Flowrates (ton/hr)
S25	0.0000
S27	15.3320
S28	0.0000
S41	2.7048
S43	0.0000
S45	0.0000
S46	15.3320
S47	0.0000

4.5 Systematic Error Detection

After all the streams are determined, the next activity in the exercise is systematic error detection. In detecting the presence of systematic error in the measurements, the global test is applied.

The global test uses the method of chi-square test. It is a special case of the gamma distribution. The parameters of this statistical tool are degrees of freedom and probability level. This is done in order to check if the null hypothesis is valid or not, by looking at the critical chi-square value from the table that corresponds to the calculated ν . If the calculated Chi-square is greater than the value in the table, then the null hypothesis is rejected and it is concluded that the predictions made were incorrect

In this case, the chi-square test with 23 degrees of freedom and probability level of 10% is done. The result of the critical value of γ_c is 32.01. Using equation 3-11, the global test statistic is computed to be $\gamma = 225.54$. Since $\gamma > \gamma_c$, thus the global test rejects the null hypothesis and a systematic error is detected.

Figure 4.8 is the graphical chi-square distribution for this case. This calculator was developed by West (2006) from the Department of Statistics, University of South Carolina, USA in 2006.

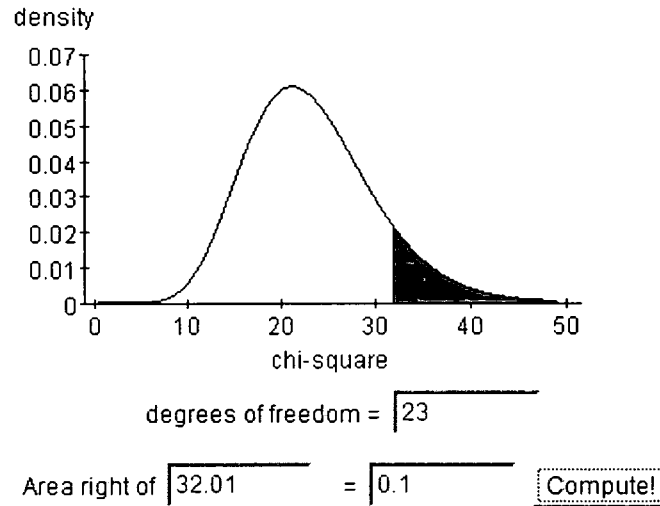


Figure 4.8: Chi-Square Calculator

Figure 4.9 shows the differences between the VBA Excel program developed in this work and the previous work by Tariq (2006). The average differences is 0.9%. This shows the VBA Excel program developed is acceptable and resulting very small differences only. The reason of the difference is that Tariq (2006) used Full QR Decomposition and the program developed here uses Economy-Size QR Decomposition. Full QR Decomposition produces an upper triangular matrix \mathbf{R} of the same dimension as \mathbf{A} , and a unitary matrix \mathbf{Q} so that $\mathbf{A} = \mathbf{Q} * \mathbf{R}$. On the other hand, "Economy-Size" QR Decomposition only computes the first n columns of \mathbf{Q} and \mathbf{R} is n -by- n for $m \times n$ matrix \mathbf{A} .

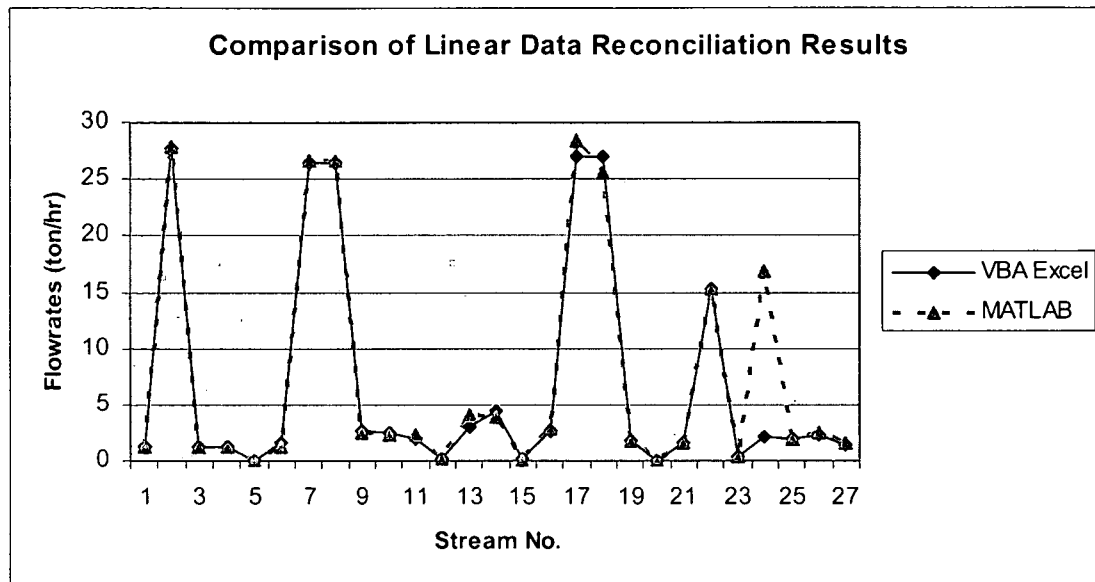


Figure 4.9: Comparisons of Linear Data Reconciliation between VB Programming and MATLAB®

4.6 Summary

In Chapter 3, the development of the program was shown. This program was used in the application of a refinery hydrogen network which has a problem with inconsistencies in material balance.

Linear data reconciliation using QR Decomposition was successfully conducted. HYSYS® process simulator was used to generate sufficient data. The system was then determinable and enabled unmeasured streams to be determined.

The program detected the presence of systematic error, which requires other techniques, beyond the scope of this project, to improve the data.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

As per objective of this study, a systematic approach of data reconciliation is developed. The developed program is for linear data reconciliation. The program is applied to a case study of a refinery hydrogen network.

Linear data reconciliation was done in the VBA Excel program using QR decomposition method. QR decomposition is used to solve data reconciliation problem when unmeasured variables exist. In solving matrix projection technique, QR decomposition is among the options. QR decomposition was noted as the most computationally efficient (Kelly, 1999). The program is able to reconcile measured variables, estimate unmeasured variables and detect the presence of gross error.

The results of the program were compared with MATLAB®. This is essential to ensure the mathematics is correct through out the work.

5.2 Future Work

There is a lot of missing elements that can be further added to the program in the future. This program has its limitations. It is limited to linear, steady state system and it reconciles flowrates only. On top of that, it simply uses material balance as constraints and no energy balance is considered. The other drawback of this program is, it only detects the presence of gross error without identifying the location of the error.

Therefore, it is suggested in the future, this work to be extended by developing data reconciliation for bilinear and nonlinear systems. Nonlinear system involves more than one variable, for instance, combinations of two or more variables ie flowrate,

composition and temperature. Energy and component balance can be considered for non-linear system as constraints as addition to material balance.

On top of that, the establishment of a program to identify the location of gross error is also essential. The information can be used by engineers for scheduling instrumentation calibration and/or maintenance.

Current trend of data reconciliation also focuses on dynamic reconciliation. This can be considered for future work.

REFERENCES

1. Almsy G.A. and Mah R.S.H., "*Estimation of Measurement Error Variances from Process Data*", Ind. Eng. Chem. Process Des. Dev., Volume 23 (1984), Pages 779 – 704.
2. Belsim S.A. (2007), "*VALI*", Houston, United States of America.
<http://www.belsim.com/VALI.aspx>
3. Bussani G., Chiari M., Grottoli M.G., "*Application of Data Reconciliation and Optimisation Procedure to Hydrogen Plant*" Computers & Chemical Engineering, Volume 19 (1995), Pages 299-304.
4. Chen J. and Romagnoli J.A., "*A Strategy for Simultaneous Dynamic Data Reconciliation and Outlier Detection*" Computers Chemical Engineering Volume 22 (1997), Pages 559-562.
5. Crowe, C.M., Campos Y.A.G. and Hrymak, A. "*Reconciliation of Process Flow Rates by Matrix Projection. I: Linear Case*", AIChE Journal., Vol.29 (1983), Pages 881-888.
6. Crowe, C.M. "*Reconciliation of Process Flow Rates by Matrix Projection, II. The Nonlinear Case*", AIChE J., Vol. 32 (1986), Pages 616-623.
7. Crowe, C.M., "*Data Reconciliation Progress and Challenges*", Journal of Process Control, Volume 6 (1996), Pages 89-98.
8. Darouach, M., Ragot J., Fayolle J., and Maquin D., "*Data Validation in Large-Scale Steady-State Linear Systems*," World Cong. on Scientific Computation, IMACS-IFAC, Paris (1988).
9. Darouach M., Zasadzinski M., Boutaveb M., "*Extension of Minimum Variance Estimation for Systems with Unknown Input*", Université Louis Pasteur-Strasbourg, (2002), Illkirch, France.
10. Gunter B. and van de Geijn R., "*Parallel Out-of-Core Computation and Updating of the QR Factorization*" (2003),
http://www.cs.utexas.edu/users/plapack/papers/pooclapack_qr.ps.gz. (date last visited: 13th April 2007)
11. Kalitventzeff, B. and P. Joris., "*Process Measurements Analysis and Validation*", (1987), The Use of Computers in Chemical Engineering, Taormina, Italy.

12. Karjala T. W., and Himmelblau D. M., "*Dynamic Data Rectification using the Extended Kalman Filter and Recurrent Neural Networks*", (1994), University of Texas, Austin, USA.
13. Karjala T. W., and Himmelblau D. M., "*Dynamic Rectification of Data via Recurrent Neural Nets and the Extended Kalman Filter*", (1996), University of Texas, Austin, USA.
14. Kelly J.D., "*A Regularization Approach to the Reconciliation of Constrained Data Sets*", *Computers Chemical Engineering* Volume 22 (1998), Pages 1771-1788.
15. Kelly J.D., "*On Finding the Matrix Projection in the Data Reconciliation Solution*" *Computers & Chemical Engineering*, Volume 22 (1999), Pages 1553-1557.
16. Keller J. Y., Zasadzinski M. and Darouach M., "*Analytical Estimator of Measurement Error Variances in Data Reconciliation*", *Computers Chemical Engineering*, Volume 16 (1992), Pages 185-188.
17. Kim I. et al., "*Robust Data Reconciliation and Gross Error Detection: The Modified MIMT Using NLP*", (1996), Yongin, Korea.
18. Kretsovalis, A. and Mah, R.S.H. "*Observability and Redundancy Classification in Generalized Process Networks. II: Algorithms*", *Computers Chem. Eng.*, Vol. 12 (1987), Pages 689-703.
19. Madron, F. "*Process Plant Performance: Measurement and Data Processing for Optimization and Retrofits*", Ellis Horwood Limited Co., (1992), Chichester, West Sussex, England
20. Mah R. S. H. and Tamhane A.C., "*R&D Notes: Detection of Gross Errors in Process Data*", *AIChE Journal* Volume 28 (1982), Pages 828-830.
21. Mah, R.S.H., Stanley, G.M. and Downing, D.W. "*Reconciliation and Rectification of Process Flow and Inventory Data*", *Ind. & EC Proc. Des. Dev.*, Vol.15 (1976), Pages 175-183.
22. MathWorks, "*MATLAB – The Language of Technical Computing*", (2007), <http://www.mathworks.com/products/matlab> (date last visited: 12th February 2007)
23. Mei C., Su H. and Chu J., "*An NT-MT Combined Method for Gross Error Detection and Data Reconciliation*", Institute of Advanced Process Control, Zhejiang University, (2005), Hangzhou, China.

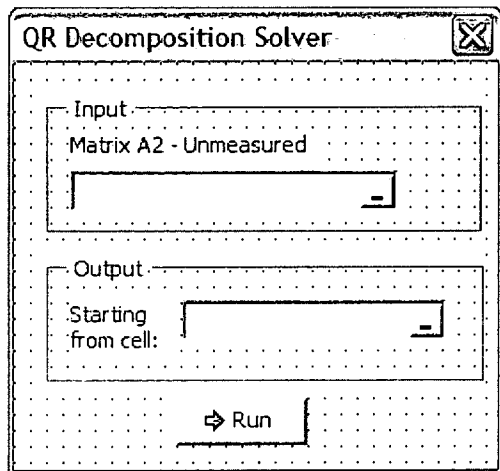
24. Merriman M., "*On the History of the Method of Least Squares*", Journal of The Analyst Vol. 4, No. 2 (1877), Pages 36.
25. Morad K., Svrcek W. Y., McKay I., "*A Robust Direct Approach for Calculating Measurement Error Covariance Matrix*", Computers and Chemical Engineering Volume 23 (1999), Pages 889–897.
26. Narasimhan S. and Jordache S., "*Data Reconciliation and Gross Error Detection: An Intelligent Use of Process Data*", Gulf Professional Publishing (2000), Houston Texas.
27. Narasimhan, S. and Mah, R.S.H. "*Generalized Likelihood Ratio Method for Gross Error Identification*", AIChE J., Vol. 33 (1987), Pages 1514-1521.
28. Narasimhan, S. (2007), "*Introduction to Data Reconciliation and Gross Error Detection*", <http://www.che.iitm.ac.in/~naras/ch544/introDRGED.pdf> (date last visited: 16th February 2007)
29. Olver P.J., "*Orthogonal Bases and the QR Algorithm*", (2006), Minnesota, USA. http://www.math.umn.edu/~olver/aims_/qr.pdf (date last visited: 16th February 2007)
30. OSIsoft Inc., "*Sigmafine*", (2007), California, United States of America. <http://www.osisoft.com/Products/Sigmafine.htm> (date last visited: 17th July 2007)
31. Pai, C.C.D. and Fisher, G.D. "*Application of Broyden's Method to Reconciliation of Nonlinearly Constrained Data*", AIChE J., Vol. 34 (1988), Pages 873-876.
32. Romagnoli J.A. and Stephanopoulos G., "*Rectification of Process Measurement Data in the Presence of Gross Errors*", Chemical Engineering Science, Volume 36 (1980), Pages 1849-1863.
33. Romagnoli J.A. and Sanchez M.C., "*Data Processing and Reconciliation for Chemical Process operations*", Academic Press, (2000), Florida, USA.
34. Rowland T., "*Orthonormal Basis*" From *MathWorld*--A Wolfram Web Resource, created by Eric W. Weisstein. <http://mathworld.wolfram.com/OrthonormalBasis.html> (date last visited: 28th June 2007).
35. Sanchez, M. and Romagnoli, J. "*Use of Orthogonal Transformations in Data Classification-Reconciliation.*" Computers Chem. Engng. Vol 20 (1996), Pages 483-493.

36. SimSci-Esscor (2007), "*DATACON*", London, United Kingdom.
www.simsi-esscor.com/us/eng/products/productlist/datacon/DATACON.htm
(date last visited: 17th Julu 2007)
37. Suarez-Chavez A., "*Data Monitoring and Reconciliation for Refinery Hydrogen Network*", (MSc Thesis), University of Manchester, Manchester (2005)
38. Stoks V.G.J, Klomp R.A.M, Terheggen C.P.F., and de Swart J.J.,
"*Construction of High Quality NN Potential Models*", Phys. Rev. C49, 2950–2962 (1994).
39. Swartz, C.L.E, "*Data Reconciliation for Generalized Flowsheet Applications*", American Chemical Society National Meeting, (1989), Dallas, Texas.
40. Tong, H. and Crowe C. M. "*Detection of Gross Errors in Data Reconciliation by Principal Component Analysis*", AIChE Journal, Vol. 41 (1996), Pages 1712-1722.
41. Tariq M.W., "*Data Monitoring and Reconciliation for Refinery Hydrogen Network*", (MSc Thesis), University of Manchester, Manchester (2006)
42. Weiss G. H., Romagnouz J. A. and Islam K. A., "*Data Reconciliation – An Industrial Case Study*", Computers Chemical Engineering Volume 20 (1995) Pages 1441-1449
43. West R.W, "*Chi-Square Demonstration*", University of South Carolina, USA. (2006), <http://www.stat.sc.edu/~west/applets/chisqdemo.html>, (date last visited: 23rd May 2007)

APPENDICES

Appendix A1

Visual Basic Code for QR Decomposition Solver



```
Private Sub CommandButton_exit_Click()
    Unload Me
End Sub
```

```
Private Sub CommandButton_OK_Click()
```

```
    If Me.RefEdit4 = "" Then
        MsgBox "Error: missing matrix A1", vbExclamation, "Matrix Operations"
        Me.RefEdit4.SetFocus
        Exit Sub
    End If
```

```
    If Me.RefEdit3 = "" Then
        MsgBox "Error: missing output cell", vbExclamation, "Matrix Operations"
        Me.RefEdit3.SetFocus
        Exit Sub
    End If
```

```
    Call ElaborationStarter
```

```
    'stop the time counter
    Time_Stop = Timer
    Time_Elaps = Time_Stop - Time_Start
```

```
    'reset the status message
    Application.StatusBar = False
```

```
'check error
If ErrMsg <> "" Then
    'show a warning or an error message

    If Left(ErrMsg, 3) = "Err" Then MsgBox ErrMsg, vbCritical

End If
Unload UserForm2
UserForm1.Show
End Sub
```

```
-----
Private Sub CommandButton1_Click()
    Dim Ref As String, ref_new As String
    If Me.RefEdit4 <> "" Then
        Ref = Me.RefEdit4
        With Range(Ref)
            If .Cells.Count = 1 Then
                ref_new = .CurrentRegion.Address
                RefEdit4.Value = ref_new
            End If
        End With
        n = Range(Me.RefEdit4).Rows.Count
        m = Range(Me.RefEdit4).Columns.Count
        Me.Label_dim1 = DimArrayFormat(n, m)
    End If
    RefEdit4.SetFocus
End Sub
```

```
-----
Private Sub CommandButton2_Click()
    Dim Ref As String, ref_new As String
    If Me.RefEdit2 <> "" Then
        Ref = Me.RefEdit2
        With Range(Ref)
            If .Cells.Count = 1 Then
                ref_new = .CurrentRegion.Address
                RefEdit2.Value = ref_new
            End If
        End With
        n = Range(Me.RefEdit2).Rows.Count
        m = Range(Me.RefEdit2).Columns.Count
        Me.Label_dim2 = DimArrayFormat(n, m)
    End If
    RefEdit2.SetFocus
End Sub
```

```

-----
Private Sub UserForm_Activate()

    If Me.RefEdit4 <> "" Then
        n = Range(Me.RefEdit4).Rows.Count
        m = Range(Me.RefEdit4).Columns.Count

        'set the default output cell
        getDimRange Me.RefEdit4, n, m
        r0 = Range(Me.RefEdit4).row
        c0 = Range(Me.RefEdit4).Column
        If IsEmpty(Cells(r0 + n + 1, c0)) Then
            Me.RefEdit3 = Cells(r0 + n + 1, c0).Address
        ElseIf IsEmpty(Cells(r0, c0 + m + 1)) And c0 + m + 1 Then
            Me.RefEdit3 = Cells(r0, c0 + m + 1).Address
        End If
    End If
    Me.RefEdit4.SetFocus
End Sub

```

```

-----
Private Sub UserForm_Initialize()
    FirstArrayInit Ref
    If Ref <> "" Then
        Me.RefEdit4 = Ref
    End If
End Sub

```

```

-----
Sub getDimRange(Ref, n, m)
    n = Range(Ref).Rows.Count
    m = Range(Ref).Columns.Count
End Sub

```

```

-----
Sub FirstArrayInit(Ref)
    Ref = ""
    If Selection.Cells.Count > 1 Then
        Ref = Selection.Address
    Else
        If Not IsEmpty(ActiveCell) Then
            Ref = ActiveCell.CurrentRegion.Address
        End If
    End If
End Sub

```

```
-----
Private Function DimArrayFormat(n, m)
    DimArrayFormat = "(" + CStr(n) + " x " + CStr(m) + ")"
End Function
-----
```

```
Private Function IsRef(Ref) As Boolean
    On Error Resume Next
    x = Range(Ref).Value
    IsRef = (Err.Number = 0)
End Function
-----
```

```
Private Sub ElaborationStarter()
    'QR decomposition

    Mat_QR_Decomp Me.RefEdit4, Me.RefEdit3
End Sub
-----
```

```
Private Sub Elaboration_Initialize()
'performs all actions for true starting elaboration
    ErrMsg = "" 'reset error message area
    Time_Start = Timer 'reset time counter
    DoEvents
End Sub
-----
```

```
Private Sub OutputMatrix(Mat, Ref, Optional sel As Boolean)
If IsMissing(sel) Then sel = False
n = UBound(Mat, 1)
If getArrayDim(Mat) = 2 Then m = UBound(Mat, 2) Else m = 1

r2 = Range(Ref).row
c2 = Range(Ref).Column
With Range(Ref).Worksheet. Activate
    .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)) = Mat
    If sel Then
        .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)).Select
    End If
End With
End Sub
-----
```

```
Private Sub InputMatrix1(Mat, Ref, Optional n, Optional m)
    r1 = Range(Ref).row
```

```

c1 = Range(Ref).Column
n = Range(Ref).Rows.Count
m = Range(Ref).Columns.Count
With Range(Ref).Worksheet
    Mat = .Range(.Cells(r1, c1), .Cells(r1 + n - 1, c1 + m - 1))
End With
End Sub

```

'////////// matrix qr decomposition operation routine //////////////////////////////////////'

```
Private Sub Mat_QR-Decomp(ref_in1, ref_out)
```

```

Dim A, b, c(), n&, m&, Fmp
InputMatrix1 A, ref_in1, n, m
'dimension check
If n < m Then
    ErrMsg = "Error: rows < columns": Exit Sub
End If
Elaboration_Initialize

```

```
'QR Decomposition...
```

```

b = Mat_QR(A)
r1 = Range(ref_out).row
c1 = Range(ref_out).Column
'write matrices Q R
Cells(r1, c1) = "matrix Q"
r = r1 + 1
ReDim c(1 To n, 1 To m)
For i = 1 To n
    For j = 1 To m
        c(i, j) = -b(i, j)          'matrix Q
    Next j, i
    Range(Cells(r, c1), Cells(r + n - 1, c1 + m - 1)) = c

```

```

-----
r = r + n + 2
Cells(r - 1, c1) = "matrix R"
ReDim c(1 To m, 1 To m)
For i = 1 To m
    For j = 1 To m
        c(i, j) = -b(i, m + j)      'matrix R
    Next j, i
    Range(Cells(r, c1), Cells(r + m - 1, c1 + m - 1)) = c

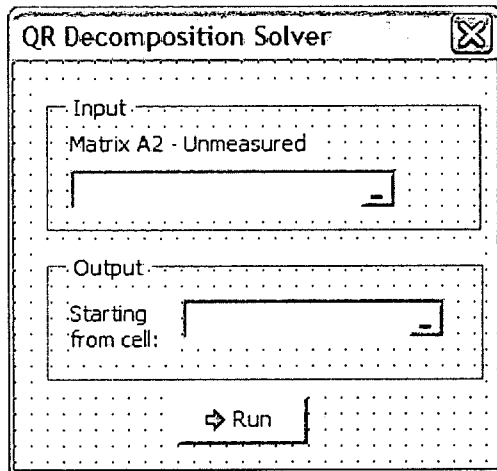
```

```
End Sub
```


APPENDICES

Appendix A1

Visual Basic Code for QR Decomposition Solver



```
Private Sub CommandButton_exit_Click()
    Unload Me
End Sub
```

```
Private Sub CommandButton_OK_Click()
```

```
    If Me.RefEdit4 = "" Then
        MsgBox "Error: missing matrix A1", vbExclamation, "Matrix Operations"
        Me.RefEdit4.SetFocus
        Exit Sub
    End If
```

```
    If Me.RefEdit3 = "" Then
        MsgBox "Error: missing output cell", vbExclamation, "Matrix Operations"
        Me.RefEdit3.SetFocus
        Exit Sub
    End If
```

```
    Call ElaborationStarter
```

```
    'stop the time counter
    Time_Stop = Timer
    Time_Elaps = Time_Stop - Time_Start
```

```
    'reset the status message
    Application.StatusBar = False
```

```
'check error
If ErrMsg <> "" Then
    'show a warning or an error message

    If Left(ErrMsg, 3) = "Err" Then MsgBox ErrMsg, vbCritical

End If
Unload UserForm2
UserForm1.Show
End Sub
```

```
Private Sub CommandButton1_Click()
    Dim Ref As String, ref_new As String
    If Me.RefEdit4 <> "" Then
        Ref = Me.RefEdit4
        With Range(Ref)
            If .Cells.Count = 1 Then
                ref_new = .CurrentRegion.Address
                RefEdit4.Value = ref_new
            End If
        End With
        n = Range(Me.RefEdit4).Rows.Count
        m = Range(Me.RefEdit4).Columns.Count
        Me.Label_dim1 = DimArrayFormat(n, m)
    End If
    RefEdit4.SetFocus
End Sub
```

```
Private Sub CommandButton2_Click()
    Dim Ref As String, ref_new As String
    If Me.RefEdit2 <> "" Then
        Ref = Me.RefEdit2
        With Range(Ref)
            If .Cells.Count = 1 Then
                ref_new = .CurrentRegion.Address
                RefEdit2.Value = ref_new
            End If
        End With
        n = Range(Me.RefEdit2).Rows.Count
        m = Range(Me.RefEdit2).Columns.Count
        Me.Label_dim2 = DimArrayFormat(n, m)
    End If
    RefEdit2.SetFocus
End Sub
```

```
-----
Private Sub UserForm_Activate()
```

```
    If Me.RefEdit4 <> "" Then
        n = Range(Me.RefEdit4).Rows.Count
        m = Range(Me.RefEdit4).Columns.Count

        'set the default output cell
        getDimRange Me.RefEdit4, n, m
        r0 = Range(Me.RefEdit4).row
        c0 = Range(Me.RefEdit4).Column
        If IsEmpty(Cells(r0 + n + 1, c0)) Then
            Me.RefEdit3 = Cells(r0 + n + 1, c0).Address
        ElseIf IsEmpty(Cells(r0, c0 + m + 1)) And c0 + m + 1 Then
            Me.RefEdit3 = Cells(r0, c0 + m + 1).Address
        End If
    End If
    Me.RefEdit4.SetFocus
End Sub
-----
```

```
Private Sub UserForm_Initialize()
```

```
    FirstArrayInit Ref
    If Ref <> "" Then
        Me.RefEdit4 = Ref
    End If
End Sub
-----
```

```
Sub getDimRange(Ref, n, m)
    n = Range(Ref).Rows.Count
    m = Range(Ref).Columns.Count
End Sub
-----
```

```
Sub FirstArrayInit(Ref)
    Ref = ""
    If Selection.Cells.Count > 1 Then
        Ref = Selection.Address
    Else
        If Not IsEmpty(ActiveCell) Then
            Ref = ActiveCell.CurrentRegion.Address
        End If
    End If
End Sub
```

```
-----
Private Function DimArrayFormat(n, m)
    DimArrayFormat = "(" + CStr(n) + " x " + CStr(m) + ")"
End Function
-----
```

```
Private Function IsRef(Ref) As Boolean
    On Error Resume Next
    x = Range(Ref).Value
    IsRef = (Err.Number = 0)
End Function
-----
```

```
Private Sub ElaborationStarter()
    'QR decomposition

    Mat_QR_Decomp Me.RefEdit4, Me.RefEdit3
End Sub
-----
```

```
Private Sub Elaboration_Initialize()
'performs all actions for true starting elaboration
    ErrMsg = "" 'reset error message area
    Time_Start = Timer 'reset time counter
    DoEvents
End Sub
-----
```

```
Private Sub OutputMatrix(Mat, Ref, Optional sel As Boolean)
If IsMissing(sel) Then sel = False
n = UBound(Mat, 1)
If getArrayDim(Mat) = 2 Then m = UBound(Mat, 2) Else m = 1

r2 = Range(Ref).row
c2 = Range(Ref).Column
With Range(Ref).Worksheet. Activate
    .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)) = Mat
    If sel Then
        .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)).Select
    End If
End With
End Sub
-----
```

```
Private Sub InputMatrix1(Mat, Ref, Optional n, Optional m)
    r1 = Range(Ref).row
```

```

c1 = Range(Ref).Column
n = Range(Ref).Rows.Count
m = Range(Ref).Columns.Count
With Range(Ref).Worksheet
    Mat = .Range(.Cells(r1, c1), .Cells(r1 + n - 1, c1 + m - 1))
End With
End Sub

```

```

'////////// matrix qr decomposition operation routine //////////

```

```

Private Sub Mat_QR-Decomp(ref_inp1, ref_out)

```

```

    Dim A, b, c(), n&, m&, Fmp
    InputMatrix1 A, ref_inp1, n, m
    'dimension check
    If n < m Then
        ErrMsg = "Error: rows < columns": Exit Sub
    End If
    Elaboration_Initialize

```

```

'QR Decomposition...

```

```

    b = Mat_QR(A)
    r1 = Range(ref_out).row
    c1 = Range(ref_out).Column
    'write matrices Q R
    Cells(r1, c1) = "matrix Q"
    r = r1 + 1
    ReDim c(1 To n, 1 To m)
    For i = 1 To n
        For j = 1 To m
            c(i, j) = -b(i, j)          'matrix Q
        Next j, i
    Range(Cells(r, c1), Cells(r + n - 1, c1 + m - 1)) = c

```

```

    r = r + n + 2
    Cells(r - 1, c1) = "matrix R"
    ReDim c(1 To m, 1 To m)
    For i = 1 To m
        For j = 1 To m
            c(i, j) = -b(i, m + j)    'matrix R
        Next j, i
    Range(Cells(r, c1), Cells(r + m - 1, c1 + m - 1)) = c

```

```

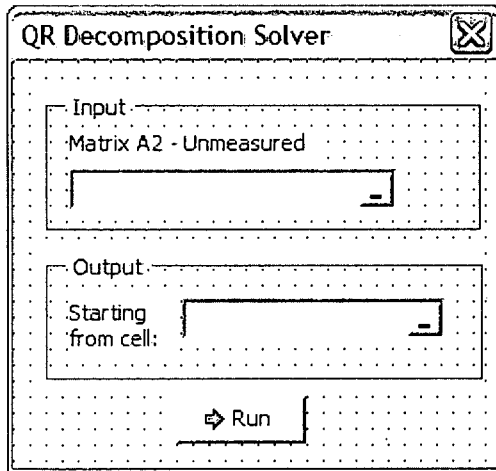
End Sub

```

APPENDICES

Appendix A1

Visual Basic Code for QR Decomposition Solver



```
Private Sub CommandButton_exit_Click()
    Unload Me
End Sub
```

```
Private Sub CommandButton_OK_Click()
```

```
    If Me.RefEdit4 = "" Then
        MsgBox "Error: missing matrix A1", vbExclamation, "Matrix Operations"
        Me.RefEdit4.SetFocus
        Exit Sub
    End If
```

```
    If Me.RefEdit3 = "" Then
        MsgBox "Error: missing output cell", vbExclamation, "Matrix Operations"
        Me.RefEdit3.SetFocus
        Exit Sub
    End If
```

```
    Call ElaborationStarter
```

```
    'stop the time counter
    Time_Stop = Timer
    Time_Elaps = Time_Stop - Time_Start
```

```
    'reset the status message
    Application.StatusBar = False
```

```
'check error
If ErrMsg <> "" Then
    'show a warning or an error message

    If Left(ErrMsg, 3) = "Err" Then MsgBox ErrMsg, vbCritical

End If
Unload UserForm2
UserForm1.Show
End Sub
```

```
Private Sub CommandButton1_Click()
    Dim Ref As String, ref_new As String
    If Me.RefEdit4 <> "" Then
        Ref = Me.RefEdit4
        With Range(Ref)
            If .Cells.Count = 1 Then
                ref_new = .CurrentRegion.Address
                RefEdit4.Value = ref_new
            End If
        End With
        n = Range(Me.RefEdit4).Rows.Count
        m = Range(Me.RefEdit4).Columns.Count
        Me.Label_dim1 = DimArrayFormat(n, m)
    End If
    RefEdit4.SetFocus
End Sub
```

```
Private Sub CommandButton2_Click()
    Dim Ref As String, ref_new As String
    If Me.RefEdit2 <> "" Then
        Ref = Me.RefEdit2
        With Range(Ref)
            If .Cells.Count = 1 Then
                ref_new = .CurrentRegion.Address
                RefEdit2.Value = ref_new
            End If
        End With
        n = Range(Me.RefEdit2).Rows.Count
        m = Range(Me.RefEdit2).Columns.Count
        Me.Label_dim2 = DimArrayFormat(n, m)
    End If
    RefEdit2.SetFocus
End Sub
```

```
-----
Private Sub UserForm_Activate()
```

```
    If Me.RefEdit4 <> "" Then
        n = Range(Me.RefEdit4).Rows.Count
        m = Range(Me.RefEdit4).Columns.Count

        'set the default output cell
        getDimRange Me.RefEdit4, n, m
        r0 = Range(Me.RefEdit4).row
        c0 = Range(Me.RefEdit4).Column
        If IsEmpty(Cells(r0 + n + 1, c0)) Then
            Me.RefEdit3 = Cells(r0 + n + 1, c0).Address
        ElseIf IsEmpty(Cells(r0, c0 + m + 1)) And c0 + m + 1 Then
            Me.RefEdit3 = Cells(r0, c0 + m + 1).Address
        End If
    End If
    Me.RefEdit4.SetFocus
End Sub
```

```
-----
Private Sub UserForm_Initialize()
```

```
    FirstArrayInit Ref
    If Ref <> "" Then
        Me.RefEdit4 = Ref
    End If
End Sub
```

```
-----
Sub getDimRange(Ref, n, m)
    n = Range(Ref).Rows.Count
    m = Range(Ref).Columns.Count
End Sub
```

```
-----
Sub FirstArrayInit(Ref)
    Ref = ""
    If Selection.Cells.Count > 1 Then
        Ref = Selection.Address
    Else
        If Not IsEmpty(ActiveCell) Then
            Ref = ActiveCell.CurrentRegion.Address
        End If
    End If
End Sub
```



```
-----
Private Function DimArrayFormat(n, m)
    DimArrayFormat = "(" + CStr(n) + " x " + CStr(m) + ")"
End Function
-----
```

```
Private Function IsRef(Ref) As Boolean
    On Error Resume Next
    x = Range(Ref).Value
    IsRef = (Err.Number = 0)
End Function
-----
```

```
Private Sub ElaborationStarter()
    'QR decomposition

    Mat_QR_Decomposition Me.RefEdit4, Me.RefEdit3
End Sub
-----
```

```
Private Sub Elaboration_Initialize()
'performs all actions for true starting elaboration
    ErrMsg = "" 'reset error message area
    Time_Start = Timer 'reset time counter
    DoEvents
End Sub
-----
```

```
Private Sub OutputMatrix(Mat, Ref, Optional sel As Boolean)
If IsMissing(sel) Then sel = False
n = UBound(Mat, 1)
If getArrayDim(Mat) = 2 Then m = UBound(Mat, 2) Else m = 1

r2 = Range(Ref).row
c2 = Range(Ref).Column
With Range(Ref).Worksheet. Activate
    .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)) = Mat
    If sel Then
        .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)).Select
    End If
End With
End Sub
-----
```

```
Private Sub InputMatrix1(Mat, Ref, Optional n, Optional m)
    r1 = Range(Ref).row
```

```

c1 = Range(Ref).Column
n = Range(Ref).Rows.Count
m = Range(Ref).Columns.Count
With Range(Ref).Worksheet
    Mat = .Range(.Cells(r1, c1), .Cells(r1 + n - 1, c1 + m - 1))
End With
End Sub

```

'////////// matrix qr decomposition operation routine //////////////////////////////////////'

```
Private Sub Mat_QR-Decomp(ref_inp1, ref_out)
```

```

Dim A, b, c(), n&, m&, Fmp
InputMatrix1 A, ref_inp1, n, m
'dimension check
If n < m Then
    ErrMsg = "Error: rows < columns": Exit Sub
End If
Elaboration_Initialize

```

```
'QR Decomposition...
```

```

b = Mat_QR(A)
r1 = Range(ref_out).row
c1 = Range(ref_out).Column
'write matrices Q R
Cells(r1, c1) = "matrix Q"
r = r1 + 1
ReDim c(1 To n, 1 To m)
For i = 1 To n
    For j = 1 To m
        c(i, j) = -b(i, j)          'matrix Q
    Next j, i
    Range(Cells(r, c1), Cells(r + n - 1, c1 + m - 1)) = c

```

```

-----
r = r + n + 2
Cells(r - 1, c1) = "matrix R"
ReDim c(1 To m, 1 To m)
For i = 1 To m
    For j = 1 To m
        c(i, j) = -b(i, m + j)    'matrix R
    Next j, i
    Range(Cells(r, c1), Cells(r + m - 1, c1 + m - 1)) = c

```

```
End Sub
```

Appendix A2

Visual Basic Code for Linear Data Reconciliation Solver

```
Private Sub CommandButton_exit_Click()
    Unload Me
End Sub
```

```
Private Sub CommandButton_OK_Click()
```

```
    If Me.RefEdit1 = "" Then
        MsgBox "Error: missing matrix Q", vbExclamation, "Matrix Operations"
        Me.RefEdit1.SetFocus
        Exit Sub
    End If
```

```
    If Me.RefEdit4 = "" Then
        MsgBox "Error: missing matrix Permutation", vbExclamation, "Matrix Operations"
        Me.RefEdit4.SetFocus
        Exit Sub
    End If
```

```
    If Me.RefEdit6 = "" Then
```

```

    MsgBox "Error: missing matrix Y: Values of Measurements", vbExclamation,
"Matrix Operations"
    Me.RefEdit6.SetFocus
    Exit Sub
End If

If Me.RefEdit7 = "" Then
    MsgBox "Error: missing matrix var: Variance", vbExclamation, "Matrix Operations"
    Me.RefEdit7.SetFocus
    Exit Sub
End If

If Me.RefEdit8 = "" Then
    MsgBox "Error: missing matrix unr: Assigned Values", vbExclamation, "Matrix
Operations"
    Me.RefEdit8.SetFocus
    Exit Sub
End If

If Me.RefEdit9 = "" Then
    MsgBox "Error: missing Matrix Y", vbExclamation, "Matrix Operations"
    Me.RefEdit1.SetFocus
    Exit Sub
End If

If Me.RefEdit11 = "" Then
    MsgBox "Error: missing Matrix A1", vbExclamation, "Matrix Operations"
    Me.RefEdit11.SetFocus
    Exit Sub
End If

If Me.RefEdit3 = "" Then
    MsgBox "Error: missing output cell", vbExclamation, "Matrix Operations"
    Me.RefEdit3.SetFocus
    Exit Sub
End If

Call ElaborationStarter

'stop the time counter
Time_Stop = Timer
Time_Elaps = Time_Stop - Time_Start

'reset the status message
Application.StatusBar = False
'check error

```

```

If ErrMsg <> "" Then
    'show a warning or an error message

If Left(ErrMsg, 3) = "Err" Then MsgBox ErrMsg, vbCritical

End If
Unload UserForm1
End Sub

```

```

Private Sub UserForm_Activate()

If Me.RefEdit1 <> "" Then
    n = Range(Me.RefEdit1).Rows.Count
    m = Range(Me.RefEdit1).Columns.Count

    'set the default output cell
    getDimRange Me.RefEdit1, n, m
    r0 = Range(Me.RefEdit1).row
    c0 = Range(Me.RefEdit1).Column
    If IsEmpty(Cells(r0 + n + 1, c0)) Then
        Me.RefEdit3 = Cells(r0 + n + 1, c0).Address
    ElseIf IsEmpty(Cells(r0, c0 + m + 1)) And c0 + m + 1 Then
        Me.RefEdit3 = Cells(r0, c0 + m + 1).Address
    End If
End If
Me.RefEdit1.1.SetFocus
End Sub

```

```

Private Sub UserForm_Initialize()
    FirstArrayInit Ref
    If Ref <> "" Then
        Me.RefEdit1 = Ref
    End If
End Sub

```

```

Sub getDimRange(Ref, n, m)
    n = Range(Ref).Rows.Count
    m = Range(Ref).Columns.Count
End Sub

```

```

Sub FirstArrayInit(Ref)
    Ref = ""

```

```

If Selection.Cells.Count > 1 Then
    Ref = Selection.Address
Else
    If Not IsEmpty(ActiveCell) Then
        Ref = ActiveCell.CurrentRegion.Address
    End If
End If
End Sub

```

```

Private Function DimArrayFormat(n, m)
    DimArrayFormat = "(" + CStr(n) + " x " + CStr(m) + ")"
End Function

```

```

Private Function IsRef(Ref) As Boolean
    On Error Resume Next
    x = Range(Ref).Value
    IsRef = (Err.Number = 0)
End Function

```

```

Private Sub Sleep(sec)
    t0 = Timer
    While Timer - t0 < sec
        DoEvents
    Wend
End Sub

```

```

Private Sub ElaborationStarter()
    'Start of the operation!

```

```

    Linear_Data_Recon Me.RefEdit1, Me.RefEdit4, Me.RefEdit6, Me.RefEdit7,
    Me.RefEdit9, Me.RefEdit11, Me.RefEdit8, Me.RefEdit3
End Sub

```

```

Private Sub Elaboration_Initialize()
'performs all actions for true starting elaboration
    ErrMsg = "" 'reset error message area
    Time_Start = Timer 'reset time counter
    DoEvents
End Sub

```

```

Private Sub OutputMatrix(Mat, Ref, Optional sel As Boolean)
If IsMissing(sel) Then sel = False
n = UBound(Mat, 1)

If getArrayDim(Mat) = 2 Then m = UBound(Mat, 2) Else m = 1

r2 = Range(Ref).row
c2 = Range(Ref).Column
With Range(Ref).Worksheet
    .Activate
    .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)) = Mat
    If sel Then
        .Range(.Cells(r2, c2), .Cells(r2 + n - 1, c2 + m - 1)).Select
    End If
End With
End Sub

```

```

'Matrix Q
Private Sub InputMatrix1(Mat, Ref, Optional n, Optional m)
    r1 = Range(RefEdit1).row
    c1 = Range(RefEdit1).Column
    n = Range(RefEdit1).Rows.Count
    m = Range(RefEdit1).Columns.Count
    With Range(RefEdit1).Worksheet
        Mat = .Range(.Cells(r1, c1), .Cells(r1 + n - 1, c1 + m - 1))
    End With
End Sub

```

```

'Matrix Permutation
Private Sub InputMatrix2(Mat2, Ref, Optional n3, Optional m3)
    r3 = Range(RefEdit4).row
    c3 = Range(RefEdit4).Column
    n3 = Range(RefEdit4).Rows.Count
    m3 = Range(RefEdit4).Columns.Count
    With Range(RefEdit4).Worksheet
        Mat2 = .Range(.Cells(r3, c3), .Cells(r3 + n3 - 1, c3 + m3 - 1))
    End With
End Sub

```

```

'Matrix R
Private Sub InputMatrix3(Mat3, Ref, Optional n4, Optional m4)
    r4 = Range(RefEdit6).row
    c4 = Range(RefEdit6).Column

```

```

n4 = Range(RefEdit6).Rows.Count
m4 = Range(RefEdit6).Columns.Count
With Range(RefEdit6).Worksheet
    Mat3 = .Range(.Cells(r4, c4), .Cells(r4 + n4 - 1, c4 + m4 - 1))
End With
End Sub

```

'Variance

```

Private Sub InputMatrix4(Mat4, Ref, Optional n5, Optional m5)
    r5 = Range(RefEdit7).row
    c5 = Range(RefEdit7).Column
    n5 = Range(RefEdit7).Rows.Count
    m5 = Range(RefEdit7).Columns.Count
    With Range(RefEdit7).Worksheet
        Mat4 = .Range(.Cells(r5, c5), .Cells(r5 + n5 - 1, c5 + m5 - 1))
    End With
End Sub

```

'Measurement Values

```

Private Sub InputMatrix5(Mat5, Ref, Optional n6, Optional m6)
    r6 = Range(RefEdit9).row
    c6 = Range(RefEdit9).Column
    n6 = Range(RefEdit9).Rows.Count
    m6 = Range(RefEdit9).Columns.Count
    With Range(RefEdit9).Worksheet
        Mat5 = .Range(.Cells(r6, c6), .Cells(r6 + n6 - 1, c6 + m6 - 1))
    End With
End Sub

```

'Matrix A1

```

Private Sub InputMatrix6(Mat6, Ref, Optional n7, Optional m7)
    r7 = Range(RefEdit11).row
    c7 = Range(RefEdit11).Column
    n7 = Range(RefEdit11).Rows.Count
    m7 = Range(RefEdit11).Columns.Count
    With Range(RefEdit11).Worksheet
        Mat6 = .Range(.Cells(r7, c7), .Cells(r7 + n7 - 1, c7 + m7 - 1))
    End With
End Sub

```

'Matrix unr

```

Private Sub InputMatrix7(Mat7, Ref, Optional n8, Optional m8)
    r8 = Range(RefEdit8).row

```



```

c8 = Range(RefEdit8).Column
n8 = Range(RefEdit8).Rows.Count
m8 = Range(RefEdit8).Columns.Count
With Range(RefEdit8).Worksheet
    Mat7 = .Range(.Cells(r8, c8), .Cells(r8 + n8 - 1, c8 + m8 - 1))
End With
End Sub

```

//////////////////////////////////// matrix operation routine //////////////////////////////////////

```

Private Sub Linear_Data_Recon(ref_inp1, ref_inp2, ref_inp3, ref_inp4, ref_inp5,
ref_inp6, ref_inp7, ref_out)

```

```

    Dim Mat_Q, Mat_A1, Mat_A2, Mat_R, Mat_Var, b, Mat_Y, Mat_Unr, c(), n&, m&,
Fmp

```

```

InputMatrix1 Mat_Q, ref_inp1, n, m      'matrix Q
InputMatrix2 Mat_A2, ref_inp2, n1, m1  'unmeasured MB
InputMatrix3 Mat_R, ref_inp3, n2, m2   'matrix R
InputMatrix4 Mat_Var, ref_inp4, n3, m3  'variance
InputMatrix5 Mat_Y, ref_inp5, n4, m4   'measured values Y
InputMatrix6 Mat_A1, ref_inp6, n5, m5   'measured MB
InputMatrix7 Mat_Unr, ref_inp7, n6, m6  'matrix unr

```

```

'dimension check

```

```

If n < m Then

```

```

    ErrMsg = "Error: rows < columns": Exit Sub

```

```

End If

```

```

Elaboration_Initialize

```

```

r1 = Range(ref_out).row

```

```

c1 = Range(ref_out).Column

```

```

Elaboration_Initialize

```

```

'bi = Rank Calculations

```

```

    bi = M_RANK(Mat_A2)

```

```

'b2 = Q1

```

```

    b2 = MatExtract(Mat_Q, Mat_A2, n, bi)

```

```

    r = r1

```

```

    Cells(r, c1) = "Submatrix Q1: row * rank"

```

```

    Range(Cells(r + 1, c1), Cells(r + n, c1 + bi - 1)) = b2

```

```

'b4 = Q2

```

```

b4 = MatExtract2(Mat_Q, Mat_A2, m, bi)
r = r + 2 + n
Cells(r, c1) = "Submatrix Q2: row * (col - rank)"
Range(Cells(r + 1, c1), Cells(r + n, c1 + (m - bi) - 1)) = b4

```

```

'b5 = R1
b5 = MatExtract3(Mat_R, Mat_A2, n, bi)
r = r + 2 + n
Cells(r, c1) = "Submatrix R1: rank * rank"
Range(Cells(r + 1, c1), Cells(r + bi, c1 + bi - 1)) = b5

```

```
'b6 = R2
```

```

b6 = MatExtract4(Mat_R, Mat_A2, m, bi)
r = r + bi + 2
Cells(r, c1) = "Submatrix R2: rank * (row-rank)"
Range(Cells(r + 1, c1), Cells(r + bi, c1 + (m - bi) - 1)) = b6

```

```
'b3 = Diagonal variance Matrix
```

```

b3 = M_DIAG(Mat_Var)
r = r + bi + 2
Cells(r, c1) = "diagonal"
Range(Cells(r + 1, c1), Cells(r + n3, c1 + n3 - 1)) = b3

```

```
'b7 = Transpose Q2, Q2'
```

```

b7 = M_T(b4)
r = r + n3 + 2
Cells(r, c1) = "Transpose Q2"
Range(Cells(r + 1, c1), Cells(r + (m - bi), c1 + n - 1)) = b7

```

```
'G=Q2*A1
```

```

G = WorksheetFunction.MMult(b7, Mat_A1)
r = r + (m - bi) + 2
Cells(r, c1) = "Matrix G"
Range(Cells(r + 1, c1), Cells(r + (m - bi), c1 + m5 - 1)) = G

```

```
'b8 = Transpose G, G'
```

```
b8 = M_T(G)
```

```
'fi=G*V*G'
```

```
'b9 = fi try = G*V
```

```
b9 = WorksheetFunction.MMult(G, b3)
```

```

fi = WorksheetFunction.MMult(b9, b8)
r = r + (m - bi) + 2
Cells(r, c1) = "Matrix fi"
Range(Cells(r + 1, c1), Cells(r + (m - bi), c1 + (m - bi) - 1)) = fi

```

```

'b10 = inverse fi
b10 = M_INV(fi)

```

```

'b11 = V*G'
b11 = WorksheetFunction.MMult(b3, b8)

```

```

'b12 = G * Y
b12 = WorksheetFunction.MMult(G, Mat_Y)

```

```

'b13= V*G'*inv(fi)
b13 = WorksheetFunction.MMult(b11, b10)

```

```

'T=V*G'*inv(fi)*G*Y
T = WorksheetFunction.MMult(b13, b12)
r = r + (m - bi) + 2
Cells(r, c1) = "T"
Range(Cells(r + 1, c1), Cells(r + (m5), c1 + 1 - 1)) = T

```

'Reconciled Values

```

'X = Y - T
x = M_SUB(Mat_Y, T)
r = r + m5 + 2
Cells(r, c1) = "X"
Range(Cells(r + 1, c1), Cells(r + (m5), c1 + 1 - 1)) = x

```

'Determination of unmeasured variables

```

't1=-inv(R1)*Q1'*A1*X
temp = M_INV(b5)
b14 = M_PRODS(temp)
b15 = M_T(b2)
b16 = WorksheetFunction.MMult(Mat_A1, x)
b17 = WorksheetFunction.MMult(b15, b16)
t1 = WorksheetFunction.MMult(b14, b17)
r = r + m5 + 2
Cells(r, c1) = "t1"
Range(Cells(r + 1, c1), Cells(r + bi, c1 + 1 - 1)) = t1

```

```

't2=-inv(R1)*R2*Mat_Unr
b18 = WorksheetFunction.MMult(b6, Mat_Unr)
t2 = WorksheetFunction.MMult(b14, b18)
r = r + bi + 2

```

```
Cells(r, c1) = "t2"
Range(Cells(r + 1, c1), Cells(r + bi, c1 + 1 - 1)) = t2
```

```
'Unmeasured values
```

```
'U=t1+t2
```

```
unmeas = M_ADD(t1, t2)
```

```
r = r + bi + 2
```

```
Cells(r, c1) = "Unmeasured"
```

```
Range(Cells(r + 1, c1), Cells(r + bi, c1 + 1 - 1)) = unmeas
```

```
'Gross Error Detection
```

```
'r=G*T
```

```
'tau=r'*inv(fi)*r
```

```
btau = M_SUB(Mat_Y, x)
```

```
r = r + bi + 2
```

```
Cells(r, c1) = "Step 1"
```

```
Range(Cells(r + 1, c1), Cells(r + 27, c1 + 1 - 1)) = btau
```

```
r = r + 29
```

```
Cells(r, c1) = "Step 2"
```

```
Cells(r + 1, c1).Select
```

```
ActiveCell.FormulaR1C1 = "=R[-29]C^2"
```

```
Cells(r + 1, c1).Select
```

```
Selection.AutoFill Destination:=Range(Cells(r + 1, c1), Cells(r + 26, c1)),
```

```
Type:=xlFillDefault
```

```
Cells(r + 28, c1 - 1).Select
```

```
ActiveCell.FormulaR1C1 = "Global Test"
```

```
Cells(r + 28, c1).Select
```

```
ActiveCell.FormulaR1C1 = "=SUM(R[-28]C:R[-2]C)"
```

```
Range(Cells(r + 28, c1), Cells(r + 28, c1)).Select
```

```
If Cells(r + 28, c1) > 32.01 Then
```

```
MsgBox "Gross Error Detected at 10% probability level with 23 DOF"
```

```
Cells(r + 28, c1).Select
```

```
End If
```

```
End Sub
```

Appendix A3

Visual Basic Code for Linear Data Reconciliation Solver Functions

```

Private Function Matrix_Mult(A1, a2)
Dim a3()
MMultiply A1, a2, a3() 'fast multiplication routine
Matrix_Mult = a3
End Function

-----
Sub MMultiply(A, b, c, Optional h)
Dim i&, j&, k&, n&, m&, p&
Dim ii&, jj&, kk&, n1&, m1&, p1&, i1&, j1&, nb&, mb&, pb&
Dim imax&, jmax&, kmax&, imin&, jmin&, kmin&
Dim A1(), b1(), c1
If IsMissing(h) Then h = 70
n = UBound(A, 1) 'rows of A
p = UBound(A, 2) 'columns of A = rows of B
m = UBound(b, 2) 'columns of B
If n <= h And m <= h Then
'fast multiplication
c = WorksheetFunction.MMult(A, b)
Exit Sub
End If
nb = Int(n / h) 'row-blocks of A
pb = Int(p / h) 'column-blocks of A = row-blocks of B
mb = Int(m / h) 'column-blocks of B
If nb * h < n Then nb = nb + 1
If pb * h < p Then pb = pb + 1
If mb * h < m Then mb = mb + 1
ReDim c(1 To n, 1 To m)
For ii = 1 To nb
For jj = 1 To mb
For kk = 1 To pb
imin = h * (ii - 1) + 1
imax = h * ii
If imax > n Then imax = n
kmin = h * (kk - 1) + 1
kmax = h * kk
If kmax > p Then kmax = p
n1 = imax - imin + 1
p1 = kmax - kmin + 1
ReDim A1(1 To n1, 1 To p1)
For i = 1 To UBound(A1, 1)
For k = 1 To UBound(A1, 2)
A1(i, k) = A(i + imin - 1, k + kmin - 1)

```

```

Next k, i
kmin = h * (kk - 1) + 1
kmax = h * kk
If kmax > p Then kmax = p
jmin = h * (jj - 1) + 1
jmax = h * jj
If jmax > m Then jmax = m
p1 = kmax - kmin + 1
m1 = jmax - jmin + 1
ReDim b1(1 To p1, 1 To m1)
For k = 1 To UBound(b1, 1)
For j = 1 To UBound(b1, 2)
    b1(k, j) = b(k + kmin - 1, j + jmin - 1)
Next j, k
c1 = WorksheetFunction.MMult(A1, b1)
imin = h * (ii - 1) + 1
jmin = h * (jj - 1) + 1
For i = 1 To UBound(c1, 1)
For j = 1 To UBound(c1, 2)
    i1 = i + imin - 1
    j1 = j + jmin - 1
    c(i1, j1) = c(i1, j1) + c1(i, j)
Next j, i
Next kk
Next jj
Next ii
End Sub

```

```

-----
Sub Mat_Transpose(A, b)
Dim i&, j&
On Error GoTo Transpose_Vector
    ReDim b(LBound(A, 2) To UBound(A, 2), LBound(A, 1) To UBound(A, 1))
    For i = LBound(A, 1) To UBound(A, 1)
    For j = LBound(A, 2) To UBound(A, 2)
        b(j, i) = A(i, j)
    Next j
    Next i
Exit Sub
Transpose_Vector:
On Error GoTo 0
    ReDim b(LBound(A, 1) To UBound(A, 1), LBound(A, 1) To LBound(A, 1))
    For i = LBound(A, 1) To UBound(A, 1)
        b(i, LBound(A, 1)) = A(i)
    Next i
End Sub
-----

```

```

Function M_T(Mat)
Dim A, b
A = Mat
Mat_Transpose A, b
M_T = b
End Function

```

```

-----
Sub MatHouseholder(u, v)
Dim i&, j&, tiny#
tiny = 10 ^ -100
n = UBound(v)
v_modulus = VectNorm(v)
For i = 1 To n
    v(i) = v(i) / v_modulus
Next
ReDim u(1 To n, 1 To n)
For i = 1 To n
For j = 1 To n
    u(i, j) = -2 * v(i) * v(j)
    If Abs(u(i, j)) < tiny Then u(i, j) = 0 'mop-up mod. 4-6-2005
    If i = j Then u(i, j) = 1 + u(i, j)
Next j
Next i
End Sub

```

```

-----
Function Mat_QR(Mat)
Dim r, q
Dim v() As Double, u() As Double, p As Double
r = Mat
m = UBound(r, 1)
n = UBound(r, 2)
ReDim v(1 To m)
If n = m Then p1 = n - 1 Else p1 = n
For k = 1 To p1
    'compute the modulus of vector k
    s = 0
    For i = 1 To m
        s = s + r(i, k) ^ 2
    Next
    s = Sqr(s)
    For i = 1 To m
        v(i) = r(i, k) / s
    Next
    D = 0
    For i = k To m
        D = D + v(i) ^ 2
    Next

```

```

Next
D = Sqr(D)
If v(k) > 0 Then D = -D
'compute V
For i = 1 To m
  If i < k Then
    v(i) = 0
  ElseIf i = k Then
    v(k) = Sqr((1 - v(k) / D) / 2)
    p = -D * v(k)
  Else
    v(i) = v(i) / p / 2
  End If
Next
Call MatHouseholder(u, v)
r = Matrix_Mult(u, r)
If k = 1 Then
  q = u
Else
  q = Matrix_Mult(u, q)
End If
Next
q = M_T(q)
For i = 1 To n
  If r(i, i) < 0 Then
    Change_Sign_Row r, i
    Change_Sign_Column q, i
  End If
Next
ReDim u(1 To m, 1 To 2 * n)
'load matrix out QR
For i = 1 To m
  For j = 1 To n
    u(i, j) = q(i, j) '-q(i, j)
    If i <= n Then
      u(i, j + n) = r(i, j) '-R(i, j)
    End If
  Next
Next
Next
Mat_QR = u
End Function
-----
Private Sub Change_Sign_Column(A, j)
For i = LBound(A, 1) To UBound(A, 1)
  A(i, j) = -A(i, j)
Next i

```



```
End Sub
```

```
-----
Private Sub Change_Sign_Row(A, i)
For j = LBound(A, 2) To UBound(A, 2)
  A(i, j) = -A(i, j)
Next j
End Sub
```

```
-----
Private Function VectNorm(v)
s = 0
For i = LBound(v) To UBound(v)
  s = s + v(i) ^ 2
Next
VectNorm = Sqr(s)
End Function
```

```
-----
Function SysLinSing(Mat, Optional v, Optional MaxErr)
Dim A1, A() As Double, m, n, Det, tol#
Dim b, elem_max, count1%, count2%
A1 = Mat
na = UBound(A1, 1)
ma = UBound(A1, 2)
If IsMissing(v) Then
  nb = na: mb = 1
Else
  b = v: nb = UBound(b, 1): mb = UBound(b, 2)
End If
If na <> nb Or mb <> 1 Then
  SysLinSing = "?": Exit Function
End If
n = Max(na, ma): m = 1
elem_max = 0
ReDim A(1 To n, 1 To ma + m)
For i = 1 To na
  For j = 1 To ma
    A(i, j) = A1(i, j)
    If Abs(A(i, j)) > elem_max Then elem_max = Abs(A(i, j))
  Next j
  For j = 1 To m
    A(i, j + ma) = 0
    If Not IsMissing(v) Then A(i, j + ma) = b(i, j)
  Next j
Next i
If IsMissing(MaxErr) Then tol = 10 ^ -13 Else tol = MaxErr
If elem_max > 1 Then MaxErrRel = tol * elem_max Else MaxErrRel = tol
If MaxErrRel > 10 ^ -6 Then MaxErrRel = 10 ^ -6
```

```

Call GaussJordan(A, n, n + m, Det, "D", MaxErrRel)
m = n + m
For i = 1 To n
For j = 1 To m
  If Abs(A(i, j)) < MaxErrRel Then A(i, j) = 0
Next j
Next i
For i = 1 To n
  Count = 0
  i1 = 0
  For j = 1 To n
    If A(i, j) <> 0 Then
      Count = Count + 1
      i1 = j
    End If
  Next j
  If Count = 1 And i1 <> i Then
    SwapRow A, i, i1
  End If
  If Count = 0 Then
    For j = n + 1 To m
      If A(i, j) <> 0 Then GoTo Error_Handler
    Next j
  End If
Next i

For k = 1 To n
  If A(k, k) <> 0 Then
    For i = k - 1 To 1 Step -1
      If A(i, k) <> 0 And i <> k Then
        pi = -A(i, k)
        pk = A(k, k)
        For j = 1 To m
          A(i, j) = pk * A(i, j) + pi * A(k, j)
          If Abs(a(i, j)) < tol Then a(i, j) = 0
        Next j
      End If
    Next i
  End If
Next k
Next k
For i = 1 To n
  If A(i, i) <> 0 And A(i, i) <> 1 Then
    pi = A(i, i)
    For j = 1 To m
      A(i, j) = A(i, j) / pi
    Next j
  End If
Next i

```

```

    End If
Next i
For i = 1 To n
    count1 = 0: count2 = 0
    For j = 1 To m
        If Abs(A(i, j)) > tol Then
            If j <= n Then count1 = count1 + 1 Else count2 = count2 + 1
        End If
    Next j
    If count1 = 0 And count2 > 0 Then GoTo Error_Handler
Next i
For j = 1 To n
    If A(j, j) = 0 Then
        For i = 1 To n
            A(i, j) = -A(i, j)
        Next i
        A(j, j) = 1
    Else
        A(j, j) = 0
    End If
Next j
tmp = MatMopUp(A, tol)
SysLinSing = tmp
Exit Function
Error_Handler:
SysLinSing = "?"
End Function
-----
Private Function Max(A, b)
If A > b Then Max = A Else Max = b
End Function
-----
Sub GaussJordan(A, n, m, Det, F, Optional dTiny)
Dim i As Integer, j As Integer, k As Integer
If IsMissing(dTiny) Then dTiny = 10 ^ -100
Det = 1
For k = 1 To n
    If F = "T" Then
        w = k + 1 ' Triangolarizza
    ElseIf F = "D" Then
        w = 1 ' Diagonalizza
    Else
        Exit Sub
    End If
    ipivot = k
    PivotMax = Abs(A(k, k))

```

```

For i = k + 1 To n
  If Abs(A(i, k)) > PivotMax Then
    ipivot = i: PivotMax = Abs(A(i, k))
  End If
Next i
If ipivot > k Then
  SwapRow A, k, ipivot
  Det = -Det
End If
If Abs(A(k, k)) <= dTiny Then
  A(k, k) = 0
  Det = 0
  Exit Sub
End If
pk = A(k, k)
Det = Det * pk
For j = 1 To m
  A(k, j) = A(k, j) / pk
Next j
For i = w To n
  If i <> k And A(i, k) <> 0 Then
    pk = A(i, k)
    For j = 1 To m
      A(i, j) = A(i, j) - pk * A(k, j)
    Next j
  End If
Next i
Next k
End Sub

```

```

Sub SwapRow(A, k, i)
  'Swaps rows k and i
  Dim j&, temp
  For j = LBound(A, 2) To UBound(A, 2)
    temp = A(i, j)
    A(i, j) = A(k, j)
    A(k, j) = temp
  Next
End Sub

```

```

Sub SwapCol(A, k, j)
  Dim i&, temp, n&
  n = UBound(A, 1)
  For i = 1 To n
    temp = A(i, j)
    A(i, j) = A(i, k)

```

```

    A(i, k) = temp
Next i
End Sub

```

```

-----
Function MatMopUp(Mat, Optional ErrMin)
Dim A
If IsMissing(ErrMin) Then ErrMin = 10 ^ -14
A = Mat
For i = 1 To UBound(A, 1)
For j = 1 To UBound(A, 2)
    If IsNumeric(A(i, j)) Then
        If Abs(A(i, j)) < ErrMin Then A(i, j) = 0
    End If
Next j
Next i
MatMopUp = A
End Function

```

```

-----
Function M_RANK(Mat)
Dim A, At, b, u
Const tiny = 10 ^ -18
A = Mat
n = UBound(A, 1): m = UBound(A, 2) 'get A dimension
If n <> m Then
    At = M_T(Mat)
    If n < m Then
        b = Application.WorksheetFunction.MMult(A, At)
        nb = n
    Else
        b = Application.WorksheetFunction.MMult(At, A)
        nb = m
    End If
Else
    b = A 'nothing to do
    nb = n
End If
u = SysLinSing(b, , tiny)
Rank = nb
For j = 1 To nb
    s = 0: For i = 1 To nb: s = s + Abs(u(i, j)): Next i
    If s > tiny Then Rank = Rank - 1
Next j
M_RANK = Rank - 1
End Function

```

```

-----
Function MatExtract(Mat, Mat2, newrow, newcol)

```

```

Dim A, D, c()
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
A = Mat
D = Mat2
n = UBound(A, 1) 'extract matrix has same no of rows
m = M_RANK(Mat2) 'extract matrix has rank no. of col
ReDim c(1 To n, 1 To m)
For i = 1 To n
For j = 1 To m
    i1 = i: j1 = j
    If i >= newrow Then i1 = i
    If j >= newcol Then j1 = j
    c(i, j) = A(i1, j1)
Next j
Next i
MatExtract = c
End Function

```

```

-----
Function MatExtract2(Mat, Mat2, newrow, newcol)
Dim A, D, c()
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
A = Mat
D = Mat2
k = M_RANK(Mat2)
n = UBound(A, 1) 'extract matrix has same no of rows
m = UBound(A, 2) - k 'extract matrix has rank no. of col
ReDim c(1 To n, 1 To m)
For i = 1 To n
For j = 1 To m
    j1 = j + k
    If i >= 0 Then i1 = i
    If j >= 0 Then j1 = j + k
    c(i, j) = A(i1, j1)
Next j
Next i
MatExtract2 = c
End Function

```

```

-----
Function MatExtract3(Mat3, Mat2, newrow, newcol)
Dim E, D, c()
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
E = Mat3
D = Mat2

```

```
n = M_RANK(Mat2) 'extract matrix has same no of rows
m = M_RANK(Mat2) 'extract matrix has rank no. of col
```

```
ReDim c(1 To n, 1 To m)
For i = 1 To n
For j = 1 To m
  i1 = i: j1 = j
  If i >= newrow Then i1 = i
  If j >= newcol Then j1 = j
  c(i, j) = E(i1, j1)
Next j
Next i
MatExtract3 = c
End Function
```

```
-----
Function MatExtract4(Mat3, Mat2, newrow, newcol)
Dim E, D, c()
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
E = Mat3
D = Mat2
n = M_RANK(Mat2) 'extract matrix has same no of rows
m = UBound(D, 2) - M_RANK(Mat2) 'extract matrix has rank no. of col
```

```
ReDim c(1 To n, 1 To m)
For i = 1 To n
For j = 1 To m
  j1 = j + n
  If i >= 0 Then i1 = i
  If j >= 0 Then j1 = j + n
  c(i, j) = E(i1, j1)
Next j
Next i
MatExtract4 = c
End Function
```

```
-----
Function M_DIAG(Diag)
Dim w(), D()
LoadVector w, Diag, n
ReDim D(1 To n, 1 To n)
For i = 1 To n
For j = 1 To n
  If i = j Then D(i, i) = w(i) Else D(i, j) = 0
Next j
Next i
M_DIAG = D
```

End Function

```

Sub LoadVector(vector, w, n)
If IsObject(w) Then
  Dim area As Range
  Set area = w
  If area.Columns.Count = 1 Then
    rows_max = ActiveSheet.Rows.Count
    n = area.Cells.Count
    If n = rows_max Then
      r1 = area.End(xlDown).row
      If area.Cells(2) = "" And r1 = rows_max Then
        n = 1
      Else
        n = r1
      End If
    End If
  End If
Else
  col_max = ActiveSheet.Columns.Count
  n = area.Cells.Count
  If n = col_max Then
    'full row selected. Example: (2:2)
    c1 = area.End(xlToRight).Column
    If area.Cells(2) = "" And c1 = col_max Then
      n = 1
    Else
      n = c1
    End If
  End If
End If
k = 0
If Not IsNumeric(area.Cells(1)) Then
  n = n - 1
  k = 1
End If
ReDim vector(1 To n)
For i = 1 To n
  vector(i) = area.Cells(i + k)
Next i
ElseIf IsMatrix(w) Then
  If UBound(w, 1) > UBound(w, 2) Then
    'vector column
    n = UBound(w, 1)
    ReDim vector(1 To n)
    For i = 1 To n: vector(i) = w(i, 1): Next
  Else

```



```

    n = UBound(w, 2)
    ReDim vector(1 To n)
    For i = 1 To n: vector(i) = w(1, i): Next
End If
ElseIf IsVector(w) Then
    n = UBound(w)
    ReDim vector(1 To n)
    For i = 1 To n: vector(i) = w(i): Next
Else
    n = 0 'something error
End If
End Sub

```

```

-----
Private Function IsMatrix(A) As Boolean
On Error GoTo Error_Handler
IsMatrix = False
n = UBound(A, 1)
n = UBound(A, 2)
IsMatrix = True
Error_Handler:
End Function

```

```

-----
Private Function IsVector(A) As Boolean
On Error GoTo Error_Handler
IsVector = False
n = UBound(A, 1)
IsVector = True
n = UBound(A, 2)
IsVector = False
Error_Handler:
End Function

```

```

-----
Function Mat_Block(Mat)
Dim A, n&, i&, j&, k&, Iter&, Iter_max&, s&, s0&
Dim Perm() As Integer
A = Mat
n = UBound(A, 1)
If n <> UBound(A, 2) Then GoTo Error_Handler "Matrix not square"
Iter_max = 2 * n
Block_Matrix_Reduction A, Perm, Iter
If Iter >= Iter_max Then GoTo Error_Handler "Iteration overflow"
Mat_Block = A
Exit Function
Error_Handler:
Mat_Block = "?"
End Function

```

```

-----
Function Mat_BlockPerm(Mat)
Dim A, n&, i&, j&, k&, Iter&, Iter_max&, s&, s0&
Dim Block(), Nblock, DimMax, Perm()
A = Mat
n = UBound(A, 1)
If n <> UBound(A, 2) Then GoTo Error_Handler  "Matrix not square"
Iter_max = 2 * n
Block_Matrix_Reduction A, Perm, Iter
If Iter >= Iter_max Then GoTo Error_Handler  "Iteration overflow"
Mat_Block_Extract A, Block, Nblock, DimMax
If Nblock = 1 Then GoTo Error_Handler  'irriducibile matrix
Mat_BlockPerm = PasteVector(Perm)
Exit Function
Error_Handler:
  Mat_BlockPerm = "?"
End Function
-----

```

```

Sub Block_Matrix_Reduction(A, Perm, Optional Iter)
Dim n&, i&, j&, k&, Iter_max&, s&, s0&
n = UBound(A, 1)
Iter_max = 2 * n
ReDim Perm(1 To 1, 1 To n)
For j = 1 To n: Perm(1, j) = j: Next j
s0 = 0
Iter = 0
Do
  s0 = s
  For i = 1 To n
    For j = n To i + 1 Step -1
      If A(i, j) <> 0 Then
        'trova uno zero sulla stella riga
        For k = 1 To j - 1
          If A(i, k) = 0 Then
            T = Score1(A, k, j)
            If T > 0 Then
              swap_rows A, k, j
              swap_columns A, k, j
              swap_columns Perm, k, j
              s = s + T
            Exit For
          End If
        End If
      End If
    Next k
  End If
Next j

```

```

Next i

For i = 1 To n - 1
  If A(i, i + 1) <> 0 And A(i + 1, i) = 0 Then
    T = Score1(A, i, i + 1)
    If T > 0 Then
      swap_rows A, i, i + 1
      swap_columns A, i, i + 1
      swap_columns Perm, i, i + 1
      s = s + T
    End If
  End If
Next
Iter = Iter + 1
Loop While s > s0 And Iter < Iter_max

End Sub
-----
Private Function PasteVector(v)
On Error GoTo Error_Handler
If Application.Caller.Rows.Count > 1 Then
  PasteVector = Application.WorksheetFunction.Transpose(v)
Else
  PasteVector = v
End If
Exit Function
Error_Handler:
  PasteVector = v
End Function
-----
Private Function Score1(Mat, p, q)
Dim A, i&, j&, k&, m&, n&
A = Mat
n = UBound(A)
s0 = 0
For j = p + 1 To n
  If A(p, j) = 0 Then s0 = s0 + Weig(p, j, n)
Next j
For j = q + 1 To n
  If A(q, j) = 0 Then s0 = s0 + Weig(q, j, n)
Next j
For i = 1 To p - 1
  If A(i, p) = 0 Then s0 = s0 + Weig(i, p, n)
Next i
For i = 1 To q - 1
  If A(i, q) = 0 Then s0 = s0 + Weig(i, q, n)

```

```

Next i
If A(p, q) = 0 Then s0 = s0 - Weig(p, q, n)

s1 = 0
For j = p + 1 To n
  If A(q, j) = 0 Then s1 = s1 + Weig(p, j, n)
Next j
For j = q + 1 To n
  If A(p, j) = 0 Then s1 = s1 + Weig(q, j, n)
Next j
For i = 1 To p - 1
  If A(i, q) = 0 Then s1 = s1 + Weig(i, p, n)
Next i
For i = 1 To q - 1
  If A(i, p) = 0 Then s1 = s1 + Weig(i, q, n)
ext i
If A(q, p) = 0 Then s1 = s1 + Weig(p, q, n)
Score1 = s1 - s0
End Function

```

```

-----
Sub swap_rows(A, r1, r2)
Dim j&, tmp
For j = 1 To UBound(A, 2)
  tmp = A(r1, j)
  A(r1, j) = A(r2, j)
  A(r2, j) = tmp
Next j
End Sub

```

```

-----
Sub swap_columns(A, c1, c2)
Dim i&, tmp
For i = 1 To UBound(A, 1)
  tmp = A(i, c1)
  A(i, c1) = A(i, c2)
  A(i, c2) = tmp
Next i
End Sub

```

```

-----
Sub Mat_Block_Extract(A, Block, Nblock, DimMax)
Dim n&, lb&, i&, j&, Edge&(), edge_max&, Block_dim&
n = UBound(A)
ReDim Block(1 To n), Edge(1 To n)
For i = 1 To n
  For j = n To 1 Step -1
    If A(i, j) <> 0 Then Exit For
  Next j

```

```

    Edge(i) = j
Next i
DimMax = 0
Nblock = 0
For i = 1 To n
    Block_dim = Block_dim + 1
    If Edge(i) > edge_max Then edge_max = Edge(i)
    If i >= edge_max Then 'one block found
        Ib = Ib + 1
        Block(Ib) = Block_dim
        If Block_dim > DimMax Then DimMax = Block_dim
        Block_dim = 0
    End If
Next i
Nblock = Ib
End Sub

```

```

-----
Function MatPerm(Permutations)
Dim v, A() As Integer
LoadVector v, Permutations, n
ReDim A(1 To n, 1 To n)

```

```

For i = 1 To n
    If 1 <= v(i) And v(i) <= n Then
        A(v(i), i) = 1
    End If
Next i
MatPerm = A
End Function

```

```

-----
Function M_INV(Mat, Optional IMode, Optional tiny)
Dim A, m, n, RetErr
A = Mat
If IsMissing(IMode) Then IMode = False
If IsMissing(tiny) Then tiny = 10 ^ -100
If IsArray(A) Then
    If UBound(A, 1) <> UBound(A, 2) Then
        RetErr = "?"
        GoTo ErrorHandler
    End If
    If Not IMode Then
        Call GJ(A, , , tiny, RetErr) 'Gauss-Jordan subroutine
    Else
        Call GJI(A, , , tiny, RetErr) "Gauss-Jordan integer subroutine
    End If
    If RetErr <> "" Then GoTo ErrorHandler

```

```

M_INV = A
Else
  If Mat = 0 Then
    RetErr = "?"
    GoTo HerrorHandler
  End If
  M_INV = 1 / Mat
End If
Exit Function
HerrorHandler:
M_INV = RetErr
End Function

```

```

-----
Sub GJ(A, Optional b, Optional Det, Optional dTiny, Optional RetErr)
Dim i%, j%, irow%, icol%, ID(), sw%, m, CalcDet As Boolean
If IsMissing(dTiny) Then dTiny = 10 ^ -100 'change 10.12.05
If Not IsMissing(Det) Then CalcDet = True
If IsMissing(b) Then m = 0 Else m = UBound(b, 2)
n = UBound(A, 1)
ReDim ID(1 To 2 * n, 1 To 3) 'trace of swaps
sw = 0 'swap counter
Det = 1
RetErr = ""
On Error GoTo Error_Handler
For k = 1 To n
  irow = k: icol = k
  PivotMax = 0
  For i = k To n
    For j = k To n
      If Abs(A(i, j)) > PivotMax Then
        irow = i: icol = j: PivotMax = Abs(A(i, j))
      End If
    Next j
  Next i

  If irow = icol And Abs(A(k, k)) <> 0 Then
    irow = k
    icol = k
  End If

  If irow > k Then
    SwapRow A, k, irow
    If m > 0 Then SwapRow b, k, irow
    If CalcDet Then Det = -Det
    sw = sw + 1
    ID(sw, 1) = k
  End If

```

```

    ID(sw, 2) = irow
    ID(sw, 3) = 1
End If
If icol > k Then
    SwapCol A, k, icol
    If CalcDet Then Det = -Det
    sw = sw + 1
    ID(sw, 1) = k
    ID(sw, 2) = icol
    ID(sw, 3) = 2
End If
If Abs(A(k, k)) <= dTiny Then
    A(k, k) = 0: Det = 0
    RetErr = "singular"
    Exit Sub
End If
pk = A(k, k)
If CalcDet Then Det = Det * pk
A(k, k) = 1
For j = 1 To n
    A(k, j) = A(k, j) / pk
Next j
For j = 1 To m
    b(k, j) = b(k, j) / pk
Next j
For i = 1 To n
    If i <> k And A(i, k) <> 0 Then
        pk = A(i, k)
        A(i, k) = 0
        For j = 1 To n
            A(i, j) = A(i, j) - pk * A(k, j)
        Next j
        For j = 1 To m
            b(i, j) = b(i, j) - pk * b(k, j)
        Next j
    End If
Next i
Next k
'scramble rows
For i = sw To 1 Step -1
    If ID(i, 3) = 1 Then
        SwapCol A, ID(i, 1), ID(i, 2)
    Else
        SwapRow A, ID(i, 1), ID(i, 2)
        If m > 0 Then SwapRow b, ID(i, 1), ID(i, 2)
    End If

```

```

Next
Exit Sub
Error_Handler:
RetErr = "overflow"
End Sub

-----
Sub GJI(A, Optional b, Optional Det, Optional dTiny, Optional RetErr)
Dim i%, j%, irow%, pk#, pi#, Ai#(), det_d#(), CalcDet As Boolean
If IsMissing(dTiny) Then dTiny = 10 ^ -100
If Not IsMissing(Det) Then CalcDet = True
If IsMissing(b) Then m = 0 Else m = UBound(b, 2)
n = UBound(A, 1)
ReDim Ai(1 To n, 1 To n), det_d(1 To n)
RetErr = ""
On Error GoTo Error_Handler
'initialization
For i = 1 To n
    Ai(i, i) = 1
    det_d(i) = 1
Next
For k = 1 To n
    'search max pivot
    irow = k
    PivotMax = 0
    For i = k To n
        If Abs(A(i, k)) > PivotMax Then
            irow = i: PivotMax = Abs(A(i, k))
        End If
    Next i
    'swap rows
    If irow > k Then
        SwapRow A, k, irow
        SwapRow Ai, k, irow
        If m > 0 Then SwapRow b, k, irow
        If CalcDet Then det_d(k) = -det_d(k)
    End If
    'check pivot 0
    If Abs(A(k, k)) <= dTiny Then
        A(k, k) = 0: Det = 0
        RetErr = "singular"
        Exit Sub
    End If
    'integer linear reduction
    For i = 1 To n
        If Abs(A(i, k)) <= tiny Then A(i, k) = 0 'mop-up Aik
        If i <> k And A(i, k) <> 0 Then

```



```

MCM_ = MCM_2(Abs(A(k, k)), Abs(A(i, k)))
pk = MCM_ / A(k, k)
pi = -MCM_ / A(i, k)
If CalcDet Then det_d(k) = det_d(k) * pi
For j = 1 To n
    A(i, j) = pi * A(i, j) + pk * A(k, j)
    Ai(i, j) = pi * Ai(i, j) + pk * Ai(k, j)
Next j
For j = 1 To m
    b(i, j) = pi * b(i, j) + pk * b(k, j)
Next j
End If
Next i
Next k
'determinant computing
If CalcDet Then
    Det = 1
    For i = 1 To n
        Det = Det * (A(i, i) / det_d(i))
    Next
End If
'normalization
For i = 1 To n
    For j = 1 To n
        Ai(i, j) = Ai(i, j) / A(i, i)
    Next j
    For j = 1 To m
        b(i, j) = b(i, j) / A(i, i)
    Next j
Next i
A = Ai 'substitute the given matrix with its inverse
Exit Sub
Error_Handler:
RetErr = "overflow" 'overflow
End Sub
-----
Function M_PROD(ParamArray Mat())
Dim b, Mi
b = Mat(0)
For i = 1 To UBound(Mat, 1)
    b = Application.WorksheetFunction.MMult(b, Mat(i))
Next
M_PROD = b
End Function
-----
Sub M_Multiply(A1, a2, a3())

```

```

Dim i&, j&, k&
Dim n1&, m1&, n2&, m2&
n1 = UBound(A1, 1)
m1 = UBound(A1, 2)
n2 = UBound(a2, 1)
m2 = UBound(a2, 2)
ReDim a3(1 To n1, 1 To m2)

```

```

For i = 1 To n1
For j = 1 To m2
For k = 1 To m1
    a3(i, j) = a3(i, j) + A1(i, k) * a2(k, j)
Next k, j, i

```

```
End Sub
```

```

-----
Private Sub Mat_Multiplication(ref_inp1, ref_inp2, ref_out)
    Dim A1, a2, b(), n1&, m1&, n2&, m2&, Fmp, u()
    InputMatrix A1, ref_inp1, n1, m1
    InputMatrix a2, ref_inp2, n2, m2
    If m1 <> n2 Then ErrMsg = "Error: wrong dimension": Exit Sub
    Elaboration_Initialize
    MMultiply A1, a2, b
    OutputMatrix b, ref_out, True
End Sub

```

```

-----
Function M_SUB(Mat1, Mat2)
'matrix subtraction
Dim A, b, c()
Dim na As Integer, ma As Integer, nb As Integer, mb As Integer
Dim i As Integer, j As Integer
A = Mat1: b = Mat2
na = UBound(A, 1): ma = UBound(A, 2)
ReDim c(1 To na, 1 To ma)
For i = 1 To na
For j = 1 To ma
    c(i, j) = A(i, j) - b(i, j)
Next j
Next i
M_SUB = c
End Function

```

```

-----
Function M_ADD(Mat1, Mat2)
Dim A, b, c()
Dim na As Integer, ma As Integer, nb As Integer, mb As Integer
Dim i As Integer, j As Integer
A = Mat1: b = Mat2

```

```

na = UBound(A, 1): ma = UBound(A, 2)
ReDim c(1 To na, 1 To ma)
For i = 1 To na
For j = 1 To ma
    c(i, j) = A(i, j) + b(i, j)
Next j
Next i
M_ADD = c
End Function

```

```

Function M_PRODS(Mat)
Dim b
k = -1
b = Mat
For i = 1 To UBound(b, 1)
For j = 1 To UBound(b, 2)
    b(i, j) = k * b(i, j)
Next j
Next i
M_PRODS = b
End Function

```

```

Function mident(n) 'nxn identity matrix
Dim row, col
ReDim matrix(1 To n, 1 To n)
For col = 1 To n
    For row = 1 To n
        If row = col Then
            matrix(row, col) = 1
        Else
            matrix(row, col) = 0
        End If
    Next row
Next col
mident = matrix
End Function

```

```

Function mzeros(n) 'nxn identity matrix
Dim row, col
ReDim matrix(1 To n, 1 To n)
For col = 1 To n
    For row = 1 To n
        If row = col Then
            matrix(row, col) = 0
        Else
            matrix(row, col) = 0
        End If
    Next row
Next col
mzeros = matrix
End Function

```

```
    End If
  Next row
Next col
mident = matrix
End Function
```

```
-----
Function Mat_Extract(Mat, newrow, newcol)
Dim D, c()
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
n = newrow 'extract matrix has same no of rows
m = newcol 'extract matrix has rank no. of col
ReDim c(1 To n, m)
If j = newcol Then
  For i = 1 To n
    i1 = i: j1 = j
    If i >= newrow Then i1 = i
    If j >= newcol Then j1 = j
    c(i, j) = Mat(i1, j1)
  Next i
End If
Mat_Extract = c
End Function
-----
```