



UNIVERSITY
OF
JOHANNESBURG

COPYRIGHT AND CITATION CONSIDERATIONS FOR THIS THESIS/ DISSERTATION



- Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- NonCommercial — You may not use the material for commercial purposes.
- ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

How to cite this thesis

Surname, Initial(s). (2012) Title of the thesis or dissertation. PhD. (Chemistry)/ M.Sc. (Physics)/ M.A. (Philosophy)/M.Com. (Finance) etc. [Unpublished]: [University of Johannesburg](https://ujdigispace.uj.ac.za). Retrieved from: <https://ujdigispace.uj.ac.za> (Accessed: Date).

THE NEURAL MODELLING OF A DIRECT REDUCTION PROCESS

by

HENDRIK MARTHINUS VISSER

A Thesis Submitted as Partial Fulfilment of the Requirements for the Degree

MAGISTER INGENERIAE

in

MECHANICAL ENGINEERING

in the

FACULTY OF ENGINEERING

at the

RAND AFRIKAANS UNIVERSITY

Supervisor: Dr AL Nel

June 2000

Abstract

The goal of this study was to determine whether a SL/RN direct reduction process could be modelled with a neural network. The full name of the SL/RN process is the *Stelco, Lurgi, Republic Steel, and National Lead process*. A parallel goal was to identify, and test an alternative method to reduce the dimensionality of a model. A neural network software package named *Process Insights* was used to model the process. Two independent data reduction methods were used along with various *Process Insights* functions, to build, train, and test models. The best model produced by each of the two data reduction methods was used to report on.

The results showed that a SL/RN direct reduction process could be modelled successfully with a neural network. The large number of variables normally identified with such a process can be reduced without significant loss in model performance. The results also showed that the removal of the most significant variable does not affect the model accuracy significantly, which bodes well for the fault tolerance of the model in terms of individual sensor failures. The *Process Insights* functions important to the modelling process were highlighted.

Abstrak

Die doel van hierdie studie was om te bepaal of 'n SL/RN direkte reduksie proses met 'n neurale netwerk gemodelleer kan word. Die volle naam van die SL/RN proses is die *Stelco, Lurgi, Republic Steel, en National Lead proses*. 'n Parallele doelwit wat hieruit spruit was om 'n alternatiewe datareduksiemetode, wat die kompleksiteit van 'n model vereenvoudig, te identifiseer en te toets. 'n Neurale netwerkpakket genaamd *Process Insights* is gebruik vir die modelleringsproses. Twee onafhanklike datareduksiemetodes tesame met verskeie *Process Insights* funksies is gebruik om modelle te bou, leer, en toets. Die beste model wat deur elke datareduksiemetode gelewer sou word, is in die resultate gebruik.

Die resultate het getoon dat 'n SL/RN proses inderdaad suksesvol deur 'n neurale netwerk gemodelleer kan word. Dit toon dat 'n groot aantal veranderlikes wat normaalweg met so 'n proses gepaard gaan, verminder kan word sonder dat daar enige noemenswaardige verliese in die model ter sprake is. Die resultate het selfs getoon dat die akkuraatheid van die model nie veel beïnvloed word indien die belangrikste veranderlike uit die model verwyder word nie. Dit dui daarop dat die model in staat is om foute in terme van individuele sensorfalings te kan aanvaar. Die resultate het ook die *Process Insights* funksies wat belangrik vir die modelleringsproses was, uitgelig.

Acknowledgements

I would like to make use of the opportunity to extend my thanks to the following people whose help aided me directly or indirectly to complete this work. My supervisor, André Nel for his patience, guidance, and knowledge, without which this manuscript would not have been possible. The management of Iscor Ltd. for their kind permission in allowing me to use this project for study purposes. My colleague, Peter Lupton for his patient proofreading and editing of this manuscript. To my wife, Benette, for her love, patience, and assistance during all my years of study. Finally, and especially, to my son, Kyle, for the love and joy you have brought into our lives.



Table of contents

i.	Abstract	i
ii	Abstrak	ii
iii	Acknowledgements	iii
iii	List of figures	ix
iv	List of tables	xi
Chapter 1	Introduction	1
1.1	Research hypotheses	1
1.2	Overview of dissertation	2
Chapter 2	The Direct Reduction process	5
2.1	History	5
2.2	Classification	5
2.3	SL/RN Process	6
2.4	Raw materials	7
2.5	DRI product	8
2.6	Vanderbijlpark Direct Reduction plant	9
2.7	Operating procedures	10
2.8	Plant control	12
2.9	References	13
Chapter 3	Optimisation	15
3.1	Introduction	15
3.2	Non-neural methods	16
3.2.1	Mathematical modelling	16
3.2.1.1	Calculus techniques (Minima and Maxima)	17
3.2.1.2	Linear programming	18
3.2.1.3	Dynamic programming	19
3.2.1.4	Recursion techniques	20

3.2.1.5	Markov techniques	21
3.2.2	Statistical techniques	22
3.3	Neural methods	24
3.4	Neural and non-neural methods in the direct reduction process	26
3.4.1	The Vanderbijlpark plant	26
3.5	References	27
Chapter 4	Neural Networks	30
4.1	Introduction	30
4.1.1	Types of networks	30
4.1.2	Classification	30
4.1.3	Learning neural networks	31
4.1.4	Applications	31
4.1.5	Benefits of neural networks	32
4.2	The human brain	32
4.3	Artificial neuron	33
4.4	Artificial neural model	34
4.5	Multi-layer perceptron	35
4.6	<i>Process Insights</i>	36
4.6.1	Edit and Format	37
4.6.2	Pre-process	37
4.6.3	Model	37
4.6.4	Analyse	38
4.6.5	Setpoints and What Ifs	38
4.7	<i>Process Insights</i> implementation	39
4.7.1	Data selection	39
4.7.2	Variable selection	40
4.7.3	<i>Process Insights</i> pre-processing	41
4.7.4	Time merging data	43
4.7.4.1	<i>Boxcar</i> method	44
4.7.4.2	<i>Linear extend</i> method	44
4.7.5	Modelling	45
4.7.5.1	Model type selection	45

4.7.5.2	Input/Output selection	45
4.7.5.3	Specifying time delays	46
4.7.5.4	Setting model patterns	47
4.7.6	Training the model	48
4.7.6.1	Measures of training performance	48
4.7.6.2	Training types	50
4.7.6.3	Training parameters	50
4.7.7	Model analysis	51
4.7.8	Modelling methodology	52
4.8	References	53
Chapter 5	Data processing	55
5.1	Introduction	55
5.2	<i>Process Insights</i> method	56
5.3	Statistical method	58
5.3.1	Principal component analysis	58
5.3.2	Criteria for the number of components to extract	60
5.3.2.1	Kaiser criterion	61
5.3.2.2	Scree test criterion	61
5.3.3	Interpreting a factor matrix	62
5.3.3.1	Rotation of factors	63
5.4	Interpreting methodology used in this study	64
5.5	References	65
Chapter 6	Results	67
6.1	Introduction	67
6.2	Model Performance	67
6.2.1	Model: K4Mod37 (<i>Process Insights</i>)	68
6.2.2	Model: Stats2 (PCA)	70
6.3	Predicted vs. Actual performance	72
6.3.1	Model: K4Mod37 (<i>Process Insights</i>)	72
6.3.2	Model: Stats2 (PCA)	73
6.4	Most significant variable analysis	74

6.4.1	Model: K4Mod37 (<i>Process Insights</i>)	75
6.4.2	Model: Stats2 (PCA)	76
6.5	Model sensitivity analysis	78
6.6	Model comparison	80
6.6.1	Statistically reduced models vs. <i>Process Insights</i> reduced models	80
6.6.2	Sparse data algorithm	82
6.6.3	Time delay specification	84
6.6.4	Time merge method	86
Chapter 7	Discussion of results	89
7.1	Introduction	89
7.2	Model performance	89
7.3	Predicted vs. Actual	90
7.4	Most significant variable analysis	91
7.5	Model sensitivity analysis	91
7.6	Model comparison	92
7.6.1	Statistically reduced model vs. <i>Process Insights</i> reduced models	92
7.6.2	Sparse data algorithm	93
7.6.3	Time delay specification	93
7.6.4	Time merge method	93
Chapter 8	Conclusion	95
Addendum		97
Appendix A		98
A.	Model inputs and outputs	98
Appendix B		100
B.	Summary of model results	100
B1.	K4Mod35 (<i>Process Insights</i>)	100

B2. K4Mod36 (<i>Process Insights</i>)	101
B3. K4Mod38 (<i>Process Insights</i>)	102
B4. K4Mod39 (<i>Process Insights</i>)	103
B5. Stats1 (PCA)	104
B6. Stats3 (PCA)	105
B7. Stats4 (PCA)	106
Appendix C	107
C. Model training graphs	107
C1. Model K4Mod37 (First training run)	107
C2. Model K4Mod37 (Last training run)	107
C3. Model Stats2 (First training run)	108
C4. Model Stats2 (Last training run)	108



UNIVERSITY
OF
JOHANNESBURG

List of figures

Figure 2.1:	The SL/RN coal based direct reduction process.	7
Figure 2.2:	The SL/RN direct reduction plant, Iscor, Vanderbijlpark.	9
Figure 3.1:	Nodes representing a process.	20
Figure 4.1:	A biological neuron.	32
Figure 4.2:	Artificial neuron.	33
Figure 4.3:	A multi-layer perceptron with one hidden layer.	35
Figure 4.4:	<i>Process Insights</i> main processing function window.	36
Figure 4.5:	Pre-processor spreadsheet window.	37
Figure 4.6:	Pre-processor plot window.	37
Figure 4.7:	<i>Process Insights</i> analyse window.	38
Figure 4.8:	<i>Process Insights</i> Setpoint & "What If" window.	39
Figure 4.9:	Variable statistics in <i>Process Insights</i> .	41
Figure 4.10:	Date/time statistics for variables in <i>Process Insights</i> .	42
Figure 4.11:	Graphical representation of data in <i>Process Insights</i> .	43
Figure 4.12:	The <i>Boxcar</i> time merge method.	44
Figure 4.13:	The <i>Linear extend</i> time merge method.	44
Figure 4.14:	Time delay specification.	46
Figure 4.15:	Variable statistics for <i>Process Insights</i> models.	48
Figure 4.16:	<i>Process Insights</i> train window display.	49
Figure 5.1:	Graph showing sensitivities of input variables on output variables.	57
Figure 5.2:	Eigenvalue plot for Scree test criterion.	62
Figure 6.1:	Relative error plot for pattern sets of model K4Mod37.	69
Figure 6.2:	R ² plot for pattern sets of model K4Mod37.	69
Figure 6.3:	Relative error plot for pattern sets of model Stats2.	71
Figure 6.4:	R ² plot for pattern sets of model Stats2.	71

Figure 6.5:	Predicted vs. Actual plot for Metallisation output variable. (K4Mod37)	72
Figure 6.6:	Predicted vs. Actual plot for Sulphur output variable. (K4Mod37)	72
Figure 6.7:	Predicted vs. Actual plot for Metallisation output variable. (Stats2)	73
Figure 6.8:	Predicted vs. Actual plot for Sulphur output variable. (Stats2)	73
Figure 6.9:	K4Mod37 first model MSV analysis performance measurements.	75
Figure 6.10:	K4Mod37 last model MSV analysis performance measurements.	76
Figure 6.11:	Stats2 first model MSV analysis performance measurements.	77
Figure 6.12:	Stats2 last model MSV analysis performance measurements.	77
Figure 6.13:	Relative error plot for model K4Mod37 sensitivity analysis.	79
Figure 6.14:	R ² plot for model K4Mod37 sensitivity analysis.	80
Figure 6.15:	<i>Process Insights</i> vs. Statistical comparison (average performance measurements).	81
Figure 6.16:	<i>Process Insights</i> vs. Statistical comparison (standard deviations).	82
Figure 6.17:	Sparse data algorithm comparison (average performance measurements).	83
Figure 6.18:	Sparse data algorithm comparison (standard deviations).	84
Figure 6.19:	Time delay comparison (average performance measurements).	85
Figure 6.20:	Time delay comparison (standard deviations).	86
Figure 6.21:	K4Mod37 time merge method comparison.	87
Figure 6.22:	K4Mod37 (-15dB) time merge method comparison.	88
Figure 7.1:	Difference between <i>Boxcar</i> and <i>Linear extend</i> time merge methods.	94
Figure C1:	Relative error training graph for first training run of model K4Mod37.	107
Figure C2:	Relative error training graph for last training run of model K4Mod37.	107
Figure C3:	Relative error training graph for first training run of model Stats2.	108
Figure C4:	Relative error training graph for last training run of model Stats2.	108

List of tables

Table 2.1:	Typical chemical composition of iron ore.	8
Table 2.2:	Typical composition of sub-bituminous coal.	8
Table 2.3:	Typical chemical composition of dolomite.	8
Table 2.4:	Direct reduced iron chemical analysis.	8
Table 4.1:	Some common activation functions.	34
Table 4.2:	Summary of models.	52
Table 5.1:	Sensitivity values calculated by <i>Process Insights</i> .	57
Table 5.2:	Eigenvalues for principal component analysis.	61
Table 6.1:	Performance measurements of model K4Mod37.	68
Table 6.2:	Performance measurements of model Stats2.	70
Table 6.3:	Predicted vs. Actual performance measurements of model K4Mod37.	73
Table 6.4:	Predicted vs. Actual performance measurements of model Stats2.	74
Table 6.5:	Performance measurements of K4Mod37 models with MSV removed.	75
Table 6.6:	Performance measurements of Stats2 models with MSV removed.	76
Table 6.7:	Model sensitivity analysis performance measurements.	79
Table 6.8:	Model K4Mod37 average performance measurements for model pattern sets.	81
Table 6.9:	Model Stats2 average performance measurements for model pattern sets.	81
Table 6.10:	Sparse data algorithm comparison for model K4Mod37.	82
Table 6.11:	Sparse data algorithm comparison for model Stats2.	83
Table 6.12:	Average performance measurements of statistical models with researcher specified time delays.	84
Table 6.13:	Standard deviations of statistical models with automatic specified time delays.	85
Table 6.14:	K4Mod37 time merge method comparison.	86

Table 6.15:	K4Mod37 (-15dB) time merge method comparison.	87
Table A1:	Model input and output variables.	99
Table B1:	Performance measurements of model K4Mod35.	100
Table B2:	Performance measurements of model K4Mod36.	101
Table B3:	Performance measurements of model K4Mod38.	102
Table B4:	Performance measurements of model K4Mod39.	103
Table B5:	Performance measurements of model Stats1.	104
Table B6:	Performance measurements of model Stats3.	105
Table B7:	Performance measurements of model Stats4.	106



Chapter 1

Introduction

The goal of this study is to determine whether a direct reduction process, or more specifically, a SL/RN direct reduction process, can be modelled by a neural network. SL/RN is the direct reduction process developed from the *Stelco, Lurgi, Republic Steel, and National Lead process*. The direct reduction plant management at Iscor, Vanderbijlpark, identified a need to optimise a relatively well-behaved process, and a number of alternative software tools were identified and researched. The management then decided upon a neural network software package, named *Process Insights*. This software package had already been used in various other non-metallurgical processes for process control purposes with success.

The aim is to use *Process Insights* to build and train models of the direct reduction process. The models are then studied to determine how well they perform in terms of predicting the process. A parallel goal is to identify an alternative means of reducing the dimensionality of the models. This research is useful because it is, as far as could be determined, the first attempt at determining whether a SL/RN direct reduction process can be modelled by a neural network and whether the neural models can be used for control purposes. The direct reduction plant at Iscor, Vanderbijlpark is mainly controlled by various distributed control systems (DCSs) in conjunction with programmable logic controllers (PLCs). Process observation is executed by operators stationed in a control room. Each of these operators has a workstation that provides a window on every piece of equipment applicable to the operator's area of responsibility. Most of the equipment that is situated directly on the kiln cannot be controlled automatically and has to be changed manually by an operator on the plant.

1.1 Research hypotheses

The goal of this research can be summarised by the following three research hypotheses:

Research hypothesis 1:

A SL/RN direct reduction process can be modelled accurately by a neural network, where the number of parameters of the model have been minimised, without incurring significant penalties in terms of model performance.

Research hypothesis 2:

By trimming a neural network's input parameters using a statistically motivated process, a significant reduction in the number of variables required to model a non-linear process can be achieved, without incurring a penalty in terms of fault tolerance.

Research hypothesis 3:

The termination of a single input parameter of a control model will not result in significant deterioration of the model's performance, indicating that the model can tolerate a single sensor failure and still be able to perform its control activities.

UNIVERSITY
OF
JOHANNESBURG

1.2 Overview of dissertation

Chapter 2 gives an overview of the direct reduction process. In this chapter, the SL/RN process of the direct reduction plant at Iscor, Vanderbijlpark is discussed. In Chapter 3, some optimisation techniques are discussed. The discussion includes both traditional techniques, such as mathematical modelling and statistical process control, and newer techniques such as neural networks in optimisation. The role of both techniques in a direct reduction process is explained. An introduction to neural networks is given in Chapter 4. The chapter also discusses the *Process Insights* software, as well as the implementation of the software. In Chapter 5, both the *Process Insights* data reduction method and the principal component analysis statistical data reduction method are discussed. The results of the study are given in Chapter 6 and a discussion of the results follows in Chapter 7. The conclusions that are made as well as pointers for future research stemming from this study are discussed in Chapter 8.

The direct reduction plant being modelled in this study is a SL/RN process. The SL/RN process is a coal-based process in which raw materials in the form of iron ore, coal, and dolomite are fed into a rotary kiln. Inside the kiln, the iron ore is reduced from iron oxide to iron with a metallisation degree of 90 - 96%. The reduced iron is then used in the electric arc furnaces at the Vanderbijlpark plant as an iron source in the production of steel.

As far as could be determined, no similar study, in terms of the neural modelling of a direct reduction plant, has been reported worldwide. A literature study showed that work had been done on other optimisation techniques for direct reduction processes. The most popular of these techniques is mathematical modelling. Two references, Brimacombe and Graue ([5], [10], Chapter 3) were found where a SL/RN process was mathematically modelled, one of which was specifically developed for the direct reduction process of the Vanderbijlpark works. However, none of these models were fully implemented on an industrial scale. The main reason why the mathematical model developed for the Vanderbijlpark plant was never implemented is because of the complexity of the model.

A neural network was selected for the modelling because of the advantages it offers over the more traditional techniques. Some of the advantages of neural networks are their ability to handle non-linearity, their ability to handle noise in the data, adaptability, fault tolerance, and robustness. *Process Insights* was selected because of its user friendliness (well designed interface), and the fact that no in-depth knowledge of neural networks is necessary to be able to use the software and produce accurate results. A disadvantage is that rule extraction and determination of causal relationships become problematic.

Because *Process Insights* has various functions that can be applied to models, a number of different models was built to test each of these functions. The functions included the "time merge" method, a "sparse data algorithm", and an automatic "find time delay" function. Models were categorised according to the data reduction method, and the functions, or combination of functions, that were used in the models. Each model went through a number of iterations where insensitive or insignificant variables were removed, the model was rebuilt and trained, and then tested to determine if further reduction was possible. Five *Process Insights* reduced models, and four statistically reduced models were produced.

Among the results expected from this study are the following:

- 1) to determine whether a direct reduction process can be modelled successfully with a neural network,
- 2) to determine whether a principal component analysis can be used successfully to reduce the dimensionality of a SL/RN process model,
- 3) to determine what effect the reduction of the number of variables in the model has on the model performance,
- 4) to determine the importance of the most sensitive variable in the model,
- 5) and to determine which of the *Process Insights* functions are best suited to the modelling process of this study.



Chapter 2

The Direct Reduction process

2.1 History

The conventional integrated route for steelmaking, from iron ore by way of coke ovens, sinter plant, blast furnace and oxygen converter is still predominant, accounting for the majority of world steel production [5]. However, the direct reduction process together with the electric arc furnace has increasingly gained importance as an alternative steelmaking route since the mid-eighties [7]. The incentives for the development of coal-based direct ironmaking are varied and can be summarised as follows:

- Investment costs are significantly lower than traditional coke battery/blast furnace routes.
- Non-coking coal is used and thus coke ovens are not required.
- Fine iron ore can be directly used without agglomeration.
- The iron product is a superior scrap substitute.
- The environmental impact of such processes is well known, and is less than that of the integrated route.
- Direct reduction processes can adapt more easily to local requirements with regard to raw materials and energy sources.

2.2 Classification

Direct reduction processes are mainly classified by the reductant and energy source used. The two main reductants used are low-grade non-coking coal and natural gas. Of the various direct reduction processes, three processes, two gas-based and one coal based, have gained a leading position and are preferentially applied in the respective DR plants. The Midrex and SL/RN processes are the leading gas- and coal-based processes respectively [5]. For gas based reduction as used in the

Midrex process a shaft furnace is used as the reactor. The rotary kiln reactor is used for solid-fuel reduction as in the SL/RN process.

2.3 SL/RN Process

The SL/RN process was developed as a combination of the RN process, which originated between 1920 and 1930 and the SL process, which was conceived in 1960 [6]. The two outstanding characteristics of the SL/RN process is that melting does not occur, and that a solid flux is used to combine with the sulphur of the fuel so that less would combine with the iron. The SL/RN process accounts for between 3 and 4% of world direct reduced iron (DRI) production with the Midrex process being the dominant process accounting for more than 60% of world DRI production [4].

Figure 2.1 illustrates the main features of the SL/RN process. The process uses a rotary kiln of special design. The raw materials, namely iron ore, coal and dolomite, are fed into the kiln in predetermined ratios where they are heated and the ore is reduced to metallic iron, or direct reduced iron (DRI). The DRI is cooled in an indirect cooler. Upon exiting the cooler the DRI is screened and magnetically separated from the non-magnetic dolochar, which is the product formed from dolomite, sulphur, carbon, and coal ash. The DRI is used in an electric arc furnace for steel production [3].

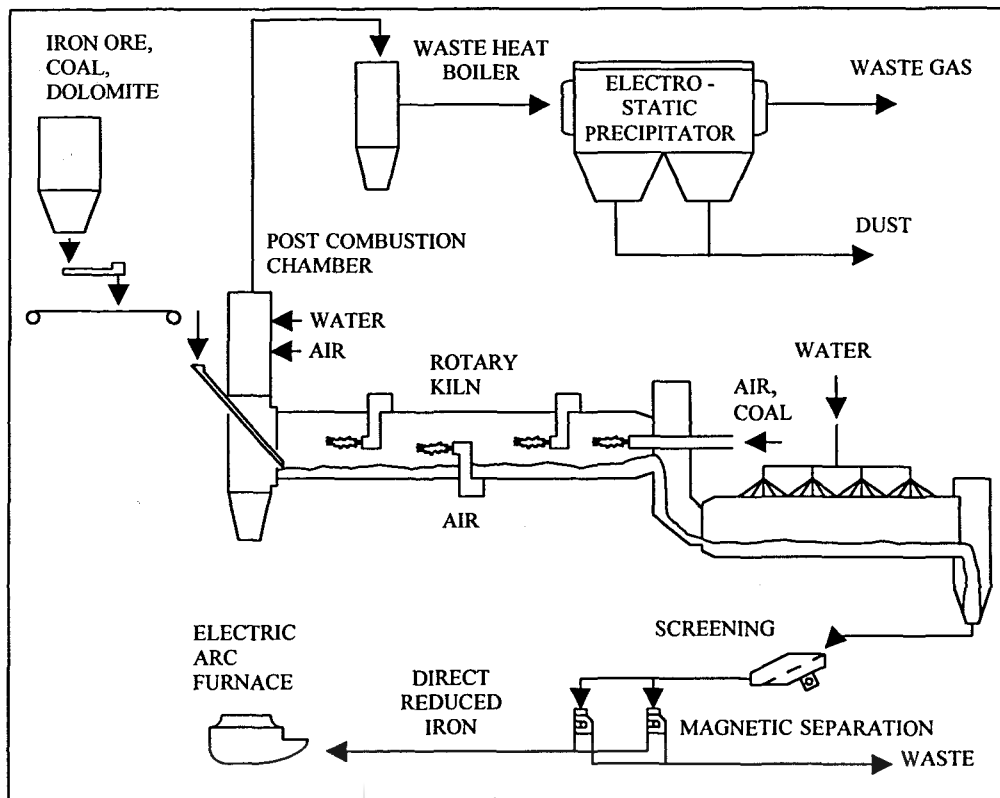


Figure 2.1: The SL/RN coal based direct reduction process.

The kiln waste gas leaves the kiln at the material feed end where it passes through a dust-settling chamber. After passing through the dust-settling chamber the gas enters a post-combustion chamber where gaseous and solid combustibles are burnt out. In-line waste heat boilers are used for waste heat recovery. When the gas is cooled down it passes through an electrostatic precipitator from where it is released into the atmosphere.

2.4 Raw materials

Tables 2.1, 2.2 and 2.3 show typical chemical analyses of feed materials used in operating SL/RN processes. An essential criterion for the suitability of lump ores is a high tumbling index. Tumbling index is a measurement of the degree of decrepitation. Very large quantities of fine material in the kiln are not acceptable. Coal used in the direct reduction process should be non-coking, have a low swelling index, and have an ash fusion temperature substantially above the maximum temperature in the process [1]. Another important parameter is the sulphur content of the coal. In order to keep

the sulphur in the DRI at an acceptably low level the sulphur content of the coal must be controlled. Other factors that are taken into account are moisture, ash and volatile content.

Type	Fe _{Tot}	SiO ₂	Al ₂ O ₃	K ₂ O	P	S	Moist
Lump	66.4	2.74	0.95	0.12	0.037	.007	0.45

Table 2.1: Typical chemical composition of iron ore.

Type	C _{fix}	Vol	Ash	S
Sub-bituminous	58	29.5	12	0.5

Table 2.2: Typical composition of sub-bituminous coal.

Type	SiO ₂	CaO	MgO	Al ₂ O ₃	FeO	MnO	K ₂ O	LOI
Dolo	2.5	29.8	19.6	0.2	0.9	0.7	0.1	46.2

Table 2.3: Typical chemical composition of dolomite.

2.5 DRI product

Table 2.4 shows a typical DRI product content analysis. The two most important measurements are the degree of metallisation and sulphur content as both have a significant influence in the arc furnace. A high sulphur content in the DRI leads to an increase in desulphurisation costs at the electric arc furnace. The lower the degree of metallisation of the DRI, the higher the energy costs of the arc furnace, due to greater amounts of FeO having to be reduced in the arc furnace.

Fe _{Tot}	Fe _{Met}	C	S	Metallisation
90-92	85-88	0.08	0.015	90.0- 96.0

Table 2.4: Direct reduced iron chemical analysis.

2.6 Vanderbijlpark Direct Reduction plant

The Iscor SL/RN direct reduction plant at Vanderbijlpark is one of the world's largest coal based direct reduction plants [1], [2] and [8]. Figure 2.2 shows a picture of the direct reduction plant of Iscor in Vanderbijlpark. The plant was completed in 1984 by Lurgi GmbH, and was commissioned in the same year. The plant has a design capacity of 720,000 t DRI per annum. Today it is still one of the world's leading producers of coal-based DRI in terms of availability, throughput and quality. The plant produces high quality DRI for the arc furnaces at the Vanderbijlpark works.



Figure 2.2: The SL/RN direct reduction plant, Iscor, Vanderbijlpark.

The direct reduction plant is divided into raw-materials handling-, product separation- and kiln sections. The raw materials handling section has storage facilities in the shape of a 60,000-t kidney shaped live ore storage pile, a 20,000-t kidney-shaped coal storage pile and a 2,000-t dolomite bunker. In the raw materials handling section the ore and coal are screened and crushed to the required particle distribution.

The plant has four eighty-metre long rotary kilns, each having an inside diameter of 4.8 meters. Each kiln has an inclination of 2.5% from material feed end to product discharge end and each kiln rests on three support stations. The kiln is lined with refractories consisting of both bricks and castable material.

Raw materials in the form of lump ore, coal and dolomite are fed into the kiln through a feed pipe (refer to figure 2.1). Coal is also injected pneumatically into the kiln from the discharge end of the kiln. Air is blown into the kiln through the coal injection pipe, the central burner and through eight air injection tubes positioned along the length of the kiln. The gasflow is countercurrent to the material flow inside the kiln. Each kiln has quick-response thermocouples with which temperatures inside the kiln are monitored. The plant has a centrally situated control room from where the plant is controlled.

Upon exiting the kiln the waste gas passes through a dust settling chamber and then a post-combustion chamber where solid and gaseous combustibles are burned. The waste gas then passes through a steam boiler where low-pressure steam is generated, and finally through an electrostatic precipitator where dust particles are removed before the gas is released into the atmosphere.

When the product exits the kiln it enters an inline rotary cooler. The cooler makes use of indirect cooling to cool the DRI to below 100°C. Each cooler is 50 meters long and has an inside diameter of 3.6 meter. The coolers are also inclined at 2.5%.

After being cooled down the DRI enters the product separation plant where the DRI is screened and magnetically separated. The magnetic part goes to a 20,000 t capacity store from where it is dispatched to the various plants that use DRI.

2.7 Operating procedures

At the start-up of the SL/RN operation it is necessary to fire the kiln with the central burner using coke-oven gas to bring the kiln and initial charging materials up to reaction temperature. After this is accomplished, a steady state is attained in which the heat produced by the combustion of a portion of the fuel charged with the burden is sufficient to raise the temperature of the incoming material to the correct temperature. There are two major steps, which correspond to the two zones in the kiln [6]. The first is the soaking zone where the material is preheated to reduction temperature and the other is the reaction zone where the ore is converted to metallic iron.

In the soaking zone moisture is driven off first, and then hydrocarbons and hydrogen are formed by the thermal decomposition of the coal. As the combustible gases from the coal rise from the bed of solid material, portions of these gases are burned in the freeboard above the bed by the controlled quantities of air that are introduced through the air injection pipes. The combustion of these gases in the freeboard radiates heat to the surface of the bed and also to the exposed area of the kiln lining. As the kiln rotates the lining carries this heat down into the bed and transfers it to the solid materials.

In the preheat zone the reduction of iron oxide proceeds only to the ferrous oxide level (FeO) according to the following equations:



The final reduction to metallic iron takes place in the reaction zone according to the following equation:



Most of the carbon dioxide from this reaction is converted back to carbon monoxide by reacting with the excess carbon in the burden according to the Boudouard reaction.



Because equation 4 is endothermic it restricts the temperature in the bed in the reducing zone so the bed can absorb heat rapidly without reaching such a high temperature that melting would become a problem.

2.8 Plant control

The direct reduction plant is divided into raw-materials handling, product separation and kiln sections, as mentioned in section 2.5. The control strategy used at the direct reduction plant is based upon these subdivisions of the plant. There are six dual-redundant distributed control systems (DCSs) on the plant. Each of the four kilns in the kiln section, the product separation plant, and the materials handling plant has its own distributed control system (DCS). The DCS of each kiln is responsible for the control of both the kiln and the low-pressure boiler. There are three distinct programmable logic controller (PLC) areas. The first covers kilns one and two, the second kilns three and four, and the third the raw materials handling and the product separation plants.

The DCSs are responsible for the control of analog data, such as temperatures, pressures, and flows in their assigned areas. The PLCs are responsible for discrete logic control such as start/stop, on/off, open/close, and forward/reverse. Each of the six sections controlled by the DCSs has their own specialised mass measurement equipment. The mass measurement equipment is connected to the DCS by means of a rack interface computer (RIC). All the DCSs and PLCs are connected to each other by means of a network, which allows communication between the equipment.

In the control room there are seven operator workstations, one for each of the four kilns, one for the four boilers, one for the raw materials handling section, and one for the product handling section. The workstations give the operators a window on every piece of equipment applicable to their areas of responsibility. Each workstation consists of a computer with a dual display. A supervisory control and data acquisition (SCADA) system supported by a real time applications platform (RTAP) database is located on each workstation. The SCADA is responsible for maintaining the graphical user interface (GUI) as well as the following functions: logging, archiving, trending, sending setpoints, processing, and retrieving data from different sources. The plant's control algorithms can be run from these systems but they are not. Instead they are run from the DCSs. The workstations are connected to a hub, which allows each individual workstation to communicate with any one of the other workstations. The hub is connected to a router, thence to the management information system (MIS) where all plant data are stored for a period of time.

A PI Historian mass storage system is connected to both the DCS network via a rack network interface (RNI), and to the workstation hub. Data generated on the plant are collected in a real time

SYBASE database that is also the MIS's database. At the same time data is written to the PI historian for high resolution archiving. Each workstation has the ability to retrieve data from either the DCS, the MIS or the PI historian.

The operators control the plant from their individual workstations. The operator can control a variable by selecting the variable on his display screen and then entering a new setpoint for that variable on his keypad. If the control of the variable entails that a motor has to be started for instance, the DCS will communicate with the PLC, which in turn will check the startup conditions. If all conditions are met the PLC will start the motor and the controller loops inside the DCS will control the variable at the selected setpoint. An operator can also change a variable by increasing or decreasing the variable's setpoint.

Most of the equipment that is situated directly on the kiln cannot be controlled automatically and has to be changed manually by an operator on the plant. This equipment includes shell air fans, central burner gas, central burner air, and plant air. There are readings available to the operator in the control room, and based on these readings the equipment can be changed manually.



2.9 References

- [1] J.P. Ackerman and C.J. Bornman, "World's largest coal-based direct reduction plant - nine years later.", *Steel Times International*, 18 - 22, March 1993.
- [2] G. Elsenheimer, L. Formanek, F. Serbent and H. Walden, "Latest SL/RN performance and combined steelmaking.", *Ironmaking Proceedings*, Vol 45, 63 - 70, 1986.
- [3] F. Grobler and R.C.A. Minnit, "The increasing role of Direct Reduced iron in Electric steelmaking.", *The Journal of the South-African Institute of Mining and Metallurgy*, 111 - 116, March/April 1999.
- [4] Midrex direct reduction corporation, "World DRI production increases 2.5% to 37.1 million tons in 1998.", *Skilling's Mining Review*, 4 - 5, March 20 1999.
- [5] F. Rose and H. Walden, "Midrex and SL/RN choices for direct reduction.", *Steel Technology International*, 91 - 96, 1989.
- [6] W.J. Rossiter, "Direct Reduction Process Theory.", *Iscor Ltd.*, 7 - 10, October 1995.

- [7] B. Sarma and R.J. Fruehan, "A review of coal-based direct ironmaking processes.", ICSTI/Ironmaking conference proceedings, 1537 - 1548, 1998.
- [8] R. Steffen, "DRI production in South-Africa using the SL/RN process.", Metallurgical Plant and Technology International, 34 - 38, March 1991.



Chapter 3

Optimisation

3.1 Introduction

The Collins English dictionary defines the word *optimum* as "a condition, degree, amount or compromise that produces the best possible result" and the word *optimise* as "to plan or carry out (an economic activity) with maximum efficiency". From this definition one can draw the conclusion that optimisation is a process where the goal is to reach a condition that produces the best possible result and thus leaving that which is to be optimised at a level of maximum possible efficiency. This principle is valid for many different systems and it can be applied to financial, industrial, medical, and business systems.

For the purpose of this study we look at optimisation from the viewpoint of the industrial process environment. Thus the optimisation of an industrial or chemical process can be defined as the process to search for the condition where the ratio of process benefits to process costs is maximised, in terms of the interaction between production rate, product quality and production costs.

Optimisation principles are of the utmost importance in modern design and system optimisation. An optimal solution is found by finding all possible solutions to a problem that also satisfy all the constraints of the problem. From the generally infinite number of solutions, the one or more, which satisfy some criteria of goodness, should be extracted by using optimisation principles [20]. Whenever we use best or optimum to describe a system, the immediate question to be asked is, "best with respect to what criteria and subject to what limitations?"

Over the years various techniques have been developed to aid the optimisation of processes. These techniques range from the more classical methods such as mathematical modelling and statistical process control to the much newer techniques developed over the last decade or two such as fuzzy logic and neural networks. To optimise a process does not necessarily require a causal model of the process. Optimisation can also be accomplished by changes in the way a process is operated. A

change of standard operating procedures can also have a positive effect on the operation of some part of a process. For the purpose of this study, the optimisation techniques will be divided into two categories, the first being non-neural methods and the second, neural networks.

3.2 Non-neural methods

The ability to optimise a system depends on the availability of alternate means of meeting system requirements and objectives. These alternatives include approaches and techniques that can be employed to meet objectives within the constraints of the resources [21]. The process of selecting an alternative that is optimal involves applying mathematical tools to determine how well the process objectives are met by the alternatives.

The study of a practical optimisation problem with most non-neural methods requires a realistic representation of the physical system by means of a suitable mathematical model and the explicit or implicit formulation of an appropriate performance criterion [11]. The mathematical model must describe correctly at least the qualitative features of the practical system in the complete range of the probable operating conditions, and the optimality criterion must be a valid representation of the practical meaning of optimality.

It can be very difficult to fit a model to the behaviour of a plant or process especially when considering the noisiness, variability, and non-linear behaviour of these plants and processes. However, there are some tools that were developed to help with the modelling of plants and processes, and with the selection of alternate solutions that satisfy the constraints of the problem. Some of the better known selection techniques are discussed in the following sections.

3.2.1 Mathematical modelling

System specifications, which, in themselves, comprise the basic guidelines for system design, are derived from a specified set of objectives and requirements. In order to apply a systematic approach to system design and development, the engineer must gain a working knowledge of the pertinent characteristics of all those mechanisms, techniques, interrelationships, and of the behaviour of the

system. This approach consists of quantitative analysis of all aspects of system operation, design, and development to optimise the system design and development so as to meet its operational requirements. This analysis is usually approached using the applied mathematical concepts and techniques common to operations research.

Operations research in the most common sense can be characterised as the application of scientific methods, techniques, and tools to problems involving the operations of systems so as to provide those in control of the operations with optimum solutions to the problems. The objective of operations research is usually to optimise. A mathematical model has as inputs certain system parameters and variables, and as outputs other system variables or values of previously selected system evaluation measures. The structure itself may consist of a family of equations, inequalities, or combinations of both, representing rules derived either from first principles or from empirical analysis. These relationships need not be linear and, in general, will be highly non-linear.

The development and utilisation of mathematical models is critical in many facets of the systems engineering process. These models can be of several types depending upon the behaviour of the input variables and the relationships between them. A few of the more common techniques are discussed in the following section. For a detailed discussion on how each technique works and the mathematics underlying each technique, the following references can be consulted: [11], [18], [20], and [21].

3.2.1.1 Calculus Techniques (Minima and Maxima)

Typical optimisation problems which arise in systems engineering involve the maximisation or minimisation of a function of n variables, say $f(x_1, \dots, x_n)$. The term *extremum* or *extreme value* is frequently used to refer to the value of $f(x_1, \dots, x_n)$ which is either a maximum or a minimum. The word optimum is used for the particular type of extremum desired in the problem at hand. The function $f(x_1, \dots, x_n)$ to be optimised is called the objective function. The point (x_1^0, \dots, x_n^0) in the domain of definition of the function at which $f(x_1, \dots, x_n)$ attains its optimal value is commonly referred to as an *extremal point* or *extreme value point*. Such points are not always unique and so it sometimes happens that more than one point exists at which an objective function assumes the same optimal value.

3.2.1.2 Linear Programming

The "linear" in linear-programming indicates that only linear equations are involved. The "programming" indicates that various variables are to be programmed - programmed in the sense of being scheduled or selected - to optimise a linear performance measure. In the usual linear programming problem, the performance measure is a specified linear algebraic equation referred to as the objective function, and other linear algebraic equations act as constraints on the optimisation process.

Many problems arise in practice, which deal with the determination of the optimal utilisation or allocation of a limited number of resources to meet given objectives. These resources may consist of manpower, materials, facilities, time, and funds. Typical objectives would be to determine which parts to make and which to buy in order to obtain the maximum profit margin, which items to stock in the inventory to minimise the quantity left over at the end of some specified time period, the best schedule of orders to machine centres to honour delivery commitments at least cost, the establishment of the best location of warehouses to minimise transportation costs, and the optimal blend of refinery resources to obtain maximum profit on commercial fuels.

In general there are some restrictions on all or some of the resources such as on the total quantity of each resource available, on the quantity and quality of the output product to be obtained, and on the total funds to be expended. Out of all permissible allocations of resources it is desired to find the one or ones which optimise some numerical quantity such as profit, cost, reliability, or systems effectiveness. If the quantity to be optimised is a linear function of the constraints, and if all relations between resource allocations are linear, then we have a linear optimisation problem.

If the type of optimisation is to maximise some quantity, then this type of problem usually has the form

$$\text{maximise } \sum_{j=1}^n c_j x_j \quad (3.1)$$

subject to constraints

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \text{ for } i = 1, \dots, m \quad (3.2)$$

$$x_j \geq 0 \text{ for } j = 1, \dots, n \quad (3.3)$$

and the minimisation problem usually has the form

$$\text{minimise } \sum_{j=1}^n c_j x_j \quad (3.4)$$

subject to constraints

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \text{ for } i = 1, \dots, m \quad (3.5)$$

$$x_j \geq 0 \text{ for } j = 1, \dots, n \quad (3.6)$$

In either case the quantity $\sum_{j=1}^n c_j x_j$ is called the objective function and the c_j 's are called the relative cost factors. Any vector (x_1, \dots, x_n) which satisfies the set of constraints is called a feasible solution. In the maximisation (minimisation) problem any feasible solution which maximises (minimises) the value of the objective function is called an optimal solution.

3.2.1.3 Dynamic programming

If we were to order the many optimisation theories on the basis of overall fertility, dynamic programming would, without a doubt, be at the top of the list. The modifier "dynamic" of dynamic programming is particularly appropriate. It is suggestive of the fact that a given problem to be solved is placed in a dynamic framework; the problem is embedded in a class of similar problems, the solutions of which are logically related and scaled in degree of difficulty. The phrase *multistage decision process* can be associated with all problems, which are solvable, by using dynamic programming.

The variables associated with dynamic programming problems can be grouped into two categories: state variables and decision variables. State variables are difficult to describe in terms broad enough to cover the spectrum of problems to which dynamic programming can be applied, but, basically, they are a measure of the conditions that exist at the beginning of any stage of solution. Decision variables are directly specified to obtain an optimal solution at each stage of a dynamic programming solution.

Unlike linear programming which is focused on the optimisation of a linear algebraic performance measure subject to satisfaction of linear algebraic constraint equations, dynamic programming can

be applied to a wide variety of different problems, both linear and non-linear, but ones that can generally be characterised in terms of a Markovian process. The approach to the solution is associated with a principle of optimality, i.e., an optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. The original aspect which appears in dynamic programming is the combined utilisation of optimisation theory and embedding procedures to obtain recurrence relations which relate one optimal solution to the next one down the line.

3.2.1.4 Recursion techniques

The solution of many types of system problems can be obtained by recursive means, that is, by expressing the system variables used in the problem formulation in terms of each other in such a way that by following an iterative procedure, the variables can be eliminated one at a time. This procedure requires that the number of variables be at most countable and usually requires an appeal to the techniques of difference equations to obtain a solution. Problems, which can be formulated using a recursive approach, arise in production processes, inventory and commodity flows, the derivation of the steady-state solution in queuing systems, and in network analysis.

A natural way to formulate a system problem to be solved by recursive means is to represent the system structure or its functional operation by means of a network. For example, figure 3.1 represents a process with each of the nodes representing a sub-process.

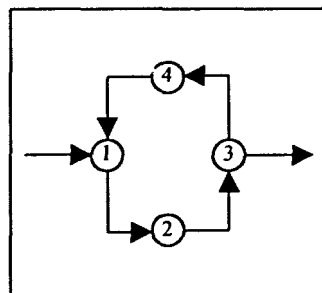


Figure 3.1: Nodes representing a process.

Referring to figure 3.1, let $f_i(X)$ denote the output of node i if X is the input to the node. Thus if X is the input to node 1 then $f_1(X)$ is both the output of node 1 and the input of node 2; hence

$f_2 (f_1 (X))$ is the output of node 2 and the input of node 3. Thus what we have done is to define the input of any node as a function of the output of the node which precedes and flows into it. In other words, given the function f_i for $1 \leq i \leq 4$ we first compute $f_1 (X)$, and then, given $f_1 (X)$, we can compute $f_2 (f_1 (X))$, the output of node 2. Now, knowing the output of node 2, we can then compute the output of node three, etc.

3.2.1.5 Markov techniques

Frequently when the behaviour of a system is described by saying it is in a certain state at a specified time, the probability law of its future state of existence depends only upon the state it is in and not on how the system arrived in that state. When this situation occurs, the system state behaviour can be described by a process called the Markov process. The number of possible states can be finite or countably infinite. In the case of a $(1, n)$ system, if we denote the system state by the number of failed components, then when j components have failed we would expect that the resulting probability law of time to failure depends only upon the fact that $n - j$ components are operating; hence, we have a Markov process. Actually, the occurrence of a particular state can be regarded as the occurrence of an event. If the times of interest are discrete and thus referred to as trials, suppose the possible outcomes of a trial are the events E_1, E_2, \dots, E_n . Define

$$p_{j,k} = \Pr \{E_k \text{ occurs at the next trial given that } E_j \text{ occurred on the preceding trial} \}$$

and

$$a_k = \Pr \{E_k \text{ is the outcome at the initial trial} \}$$

for $j, k = 1, 2, \dots, N$. Then suppose that we can write

$$\Pr \{E_{j_1}, \dots, E_{j_n}\} = a_{j_1} p_{j_1, j_2} \dots p_{j_{n-1}, j_n} \quad (3.7)$$

For all $n \geq 0$ and n -tuples (j_1, \dots, j_n) . Equation 3.7 gives the probability that event E_{j_1} occurs on the initial trial, E_{j_2} occurs on the next trial, \dots , and E_{j_n} occurs on the n^{th} trial. A sequence of trials with possible outcomes E_1, E_2, \dots, E_n such that the probability of a sample sequence of n events is defined by equation 3.7 is generally referred to as a Markov chain with finite state space of size N .

3.2.2 Statistical techniques

Statistical process control (SPC) is another technique that can be used to optimise a plant or process. Statistical process control is the use of statistically based methods to evaluate a process or its outputs to achieve or maintain a state of control [9]. By maintaining this state of control at one of the process's optimum levels one can optimise the process. A more comprehensive definition of SPC is that it is a technique, which is used to monitor, control, evaluate, and analyse a process.

The way to improve productivity is to improve quality by using statistical process control techniques. SPC involves using statistical analyses to monitor process performance with the goal of preventing quality problems, rather than detecting them after the fact. SPC comprises a set of systematic procedures for evaluating and improving quality, and hence, in the long run, productivity [16].

There are several statistically based tools used in SPC of which the following are the most commonly used: Pareto analysis, Cause and effect (Fishbone) diagram, Scatter diagram, Checklist, Control chart, Data gathering, and Histograms. To achieve SPC goals all seven of these statistically based tools need to be integrated in both the implementation phase and during operation. Thus in short, in order to reap a good portion of SPC potentiality, SPC implementation and operation must involve the use of all seven quality tools.

The goal of SPC is to continuously improve quality, reliability and service by reducing process variability. Some of the tangible benefits of SPC are: improved production efficiency, pinpointing problem occurrences, providing a usable measure of performance, product consistency, less downtime, and shorter deviation recovery periods. Also, many of the SPC benefits are intangible and their complete effect is often attainable only after the system matures. A prerequisite to the effective implementation of SPC is the existence and adequate application of the basic elements of an overall quality system.

The most fundamental principle of SPC is that all processes produce variation in the output. Variation, which occurs in both tangible products and services, has two sources: normal and abnormal. Normal variation reflects numerous natural, extraneous, and unsystematic factors that are inherent in the system. Such factors can be regarded as change or noise in the system. Normal

variation can be reduced but it can never be eliminated. Some factors that can cause normal variation in the output include: poorly trained groups of employees, inaccuracies inherent in instruments, vibration, humidity, poor product design and slight inconsistencies in the raw materials

Abnormal variations reflect the existence of special or unusual, non-random factors affecting the system's performance. Abnormal variations generally arise on an irregular basis and influence only some of the products. A broken tool or inappropriate adjustments are examples of factors that produce abnormal variation. Often, although not always, abnormal variation is caused by factors which can be corrected at the machine by the operator.

A primary objective of SPC is to determine whether the variation in a product or service is due to normal or abnormal causes. If the variation is caused by normal causes, then the system is in statistical control. When the system is in control the variation in its output will assume a normal distribution, and it will be stable over time. Statistical control is not a natural condition for a process. It must be achieved by eliminating the special causes in the process. When all the special causes have been eliminated, only the random, but limited variation remains.

When a process is in statistical control, it does not imply that the output is acceptable, only that the output is consistent and predictable. The variability may still be too large, and have to be reduced before acceptable products can be consistently produced. Failure to distinguish between normal and abnormal variation has two unfortunate consequences. First when normal variation is mistaken for abnormal variation, attempts by the operator to improve quality by greater effort or by tampering with the current work procedures are doomed to failure. Second, when normal variation is attributed to the worker rather than to the factors inherent in the system, the worker can be incorrectly rewarded or punished for results that are beyond his or her control.

The primary tool for determining whether product variability is normal or abnormal is the control chart. There are two types of control charts: control charts for attributes (used where the dimension either conforms or does not conform to standards), and control charts for variables (used where the dimension is measured as a continuous variable such as length or weight). Both types of charts serve as signalling devices that tell the worker when to intervene and remove abnormal causes, and just as important, when to leave the process alone. Without objective data and this charting method, it is easy to confuse normal and abnormal variation.

The control chart simply tracks the process, and identifies what is happening with the process mean and range. It identifies when an abnormal cause has entered the system, but does not necessarily identify what the abnormal cause is. Once a control chart has confirmed that a problem exists, a list of possible causes is created. These causes are located on a cause-and-effect diagram. Once the cause-and-effect diagram is completed, one or more of the causes are selected and evaluated using traditional experimental design methods. For a detailed description on how SPC works the following references can be consulted: [9], [16], [15], and [24].

3.3 Neural methods

The use of neural networks for advanced control and optimisation has increased over the past decade. When referring to neural networks in optimisation one assumes that a model is built to represent the process as closely as possible, and then this model is used to predict process dynamics. A neural network used to control a process is another form of optimisation. In this case the neural network controls the process or some part of a process at one of its optimum operation points, thus minimising disturbance, recovery time from delays, and offgrade product.

Neural networks used for optimisation have found a wide range of applications. Applications where neural networks were found to be used to optimise processes or plants are: metallurgical applications [2], [3], [4], [6], [12], [19], [22], [23], [25], and [26]; chemical applications [8], [14], and [17]; and general control applications [7], and [13]. The one application where a neural network was specifically used for optimisation was where it was used to optimise a gasloop process [14]. In this application one of the goals was to use the neural network to advise on the setpoint trajectory of certain process variables in order to optimise the gasloop process. The paper reported that the network's recommendations resulted in an increase in production. Further it reports that the network predicts the production of total oil and condensate within 1% of the actual production. In conclusion it shows that neural networks can be used successfully where traditional approaches are inferior to use for optimisation.

For the purpose of this study we are more interested in the applications where the neural networks were used in metallurgical optimisation. A comprehensive literature study showed that there are a vast number of cases reported in articles, and papers presented at conferences, where neural

networks were used in metallurgical applications. Some of the metallurgical applications are: predicting and controlling of pyro-metallurgical processes [19], prediction of nozzle clogging during casting [23], modelling of metal-slag equilibrium processes [22], monitoring and control of hydrometallurgical processes [3], analysis of steel plate processing [26], cost savings in an electric arc furnace controller [4], blast furnace neural system [2], prediction of roll force in hot rolling [12], prediction of silicon content in blast furnace hot metal [25], and the modelling of gas metal arc weld geometry [6].

What is evident from all these articles and papers is that the approaches used by the authors are very similar. Process data were collected and then used to train the neural networks. The authors used different networks for the different applications. In some of the applications different network types were used on the same problem to examine which network predicted the process more accurately [25]. In other applications more than one network of the same type was used to model different aspects of a process [12]. Some of the networks that were used in these studies were: backpropagation networks, dynamic learning rate networks, functional link networks, fuzzy neural networks, and self-organising maps. The trained networks were subjected to extensive training, and the results were reported in the articles and papers.

In all of the cases the authors reported positively on the use of neural networks in their applications. When saying that the results were positive it does not mean that neural networks are the solution to all problems and that they will replace the more traditional methods. In some of the cases the authors did report that the results obtained were better than that obtained by the more traditional methods. However, there were also instances where the authors reported that neural networks gave positive results and that they should only be used as an aid to more traditional methods. What should not be ignored is the fact that one can get good results with both methods and that both methods should be applied accordingly.

Neural networks are still fairly new in the fields of optimisation and control. One should not see neural networks as the mystical magical wonder tools everybody has been waiting for. There is still a lot of development needed on the use of neural networks in certain applications. However, results have shown that in some applications they produce results that are more accurate and credible than those produced by the more traditional methods.

3.4 Neural and non-neural methods in the direct reduction process

A literature study showed that relatively little work has been done in the statistical and neural fields with regard to the optimisation and control of a direct reduction process, as no information on this subject could be found. What the study did show is that a small number of mathematical models were developed specifically for direct reduction processes [1], [5], and [10].

The fact that no information could be found on statistical and neural methods being used to model, optimise or control a direct reduction process does not mean that no research has ever been done in this field. It could be that reported cases are few and far between and not easily attainable.

Three papers were found, that reported on the mathematical modelling of direct reduction processes. Two of them reported on the modelling of the SL/RN process [5], and [10], and the third on another process where pelleted iron ore and coal are fed through a rotary kiln [1]. In all three cases the authors reported that the models were fairly successful at predicting the attributes they modelled. In one of the cases the model was based on a pilot plant [5]. The model was scaled up to compare with a full scale operating plant. In the other two cases the models were based on actual operating plants [1], and [10].

The first model was used to predict materials transport, thermal transfer, and reaction kinetics [10]. The second was used to predict operating behaviour such as reduction rates, Boudouard reaction, coal devolatilisation, combustion in the freeboard gas, mass flows, and heat flows [5]. The model was used to investigate the influence of different process variables on the kiln performance. The third model was used to predict reaction time according to the temperature profile inside the kiln [1]. This model was used to investigate the influence of certain variables on the predicted variable.

3.4.1 The Vanderbijlpark plant

As mentioned earlier, no information on statistical or neural optimisation techniques could be found. A study of the archives on the direct reduction plant of the Vanderbijlpark works confirmed this. However a doctoral thesis report by a German student was found [10]. The report was on the mathematical modelling of the Vanderbijlpark direct reduction process.

No report on the actual implementation of the model on the plant could be found. The process engineers on the plant were consulted. They confirmed that the model was never implemented due to various reasons. The main reason was that the model was too complex. Some engineers have studied the model and found that the notation was difficult to follow and the meaning of symbols was not always clear. One reason for this could be the fact that the report was translated from German into English. The result was that the model was never implemented or tested and used to optimise the process.

One of the plant engineers revealed that an engineer took only a part of the model and analysed it. He studied the part on materials transport, and used it to predict the residence time of material inside the kiln. Part of the model was programmed and tested with known values for the process variables in the model. The results produced compared fairly well with process knowledge but results could never be evaluated, the main reason being that there was no viable means to determine the actual residence time of the material inside the kiln as the material is not visible at any point. The result was that research was terminated and the model was not implemented.

From this, it is clear that very little has been done in terms of using optimisation tools available to optimise the direct reduction process of the Vanderbijlpark works. It would also be fair to say that this study is the first attempt to implement a neural method for process optimisation of the direct reduction process in Vanderbijlpark. It could also be the first attempt in the world on a SL/RN process.

3.5 References

- [1] B.B. Agrawal, K.K. Prasad, S.B. Sarkar and H.S. Ray, "Prediction of Degree of Reduction of Iron Ore-Coal Composite Pellets in a Rotary Kiln Sponge Iron Plant.", *Steel India*, Vol. 19, No. 2, 73 - 76, October 1996.
- [2] M. Alaraasakka and O. Ritamäki, "The Rautaruukki blast furnace neural system.", *Steel World*, Vol. 3, No. 2., 12 - 14.
- [3] C. Aldrich, D.W. Moolman and J.S.J. van Deventer, "Monitoring and Control of Hydro-Metallurgical Processes with Self-Organising and Adaptive Neural Net Systems.", *European Symposium on Computer Aided Process Engineering - 5*, S803 - S808, 1995.

- [4] S. Algadrie and H. Palyama, "Cost saving in EAF Operation Neural Network Controller.", *SEAIISI Quarterly*, 55 - 60, July 1998.
- [5] J.K. Brimacombe and V. Venkateswaran, "Mathematical model of the SL/RN Direct Reduction Process.", *Metallurgical Transactions B*, Vol. 8B, 387 - 398, September 1977.
- [6] B. Chan, J. Pacey and M. Bibbys, "Modelling Gas Metal Arc Weld Geometry Using Artificial Neural Network Technology.", *Canadian Metallurgical Quarterly*, Vol. 38, No. 1, 43 - 51, 1999.
- [7] R. Damle, R. Lsahlee, V. Rao and F. Kern, *Identification and robust control of smart structures using artificial neural networks.*, 35 - 46, IOP Publishing Ltd. (1994).
- [8] A Fortpied and L. Heymans, "Neural Networks to Control an Alpha Olefin Plant.", *Honeywell Users Conference*, September 1995.
- [9] L.K. Gaafar and J.B. Keats, "Statistical Process Control: A Guide For Implementation", *International Journal of Quality & Reliability Management*, Vol. 9, No. 4, 9 - 20, 1992.
- [10] R.H. Graue, "Mathematical Model for the Direct Reduction of Iron Ores by a Coal rotary Kiln.", *Doctoral Thesis submitted at the Technische Universität Berlin*, 1983.
- [11] I. Gumowski and C. Mira, *Optimisation in Control Theory and Practice*, Cambridge University Press, London (1968).
- [12] Y. J. Hwu and J.G. Lenard, "Application of Neural Networks in the Prediction of Roll Force in Hot Rolling.", *37th MWSP Conference Proceedings ISS*, Vol. XXXIII, 549 - 554, 1996.
- [13] M. Jang, S. Cho, S. Lee and Y. Kwon, "Belt speed control in a sintering plant using neural networks.", *Steel research* 69, No. 10 + 11, 398 - 405, 1998.
- [14] H.W. Joubert, P.L. Theron and T. Lange, "The Use Of Neural Networks For Gasloop Process Optimization.", *ELEKTRON Journal of the South African Institute of Electrical Engineers*, August 1996.
- [15] T.R. Linde, "Implementation and Automation of Analytical Statistical Process Control", *Computerization and Data Management in the Metals Analysis Laboratory*, ASTM STP 973, M.A. Worthington and N.L. Botone, Eds, American Society for Testing and Materials, Philadelphia, 75 - 91, 1988.
- [16] L.E. Mainstone and A.S. Levi, *Organizational Management Behaviour and Statistical Process Control*, The Haworth Press (1987).
- [17] T.J. McAvoy and H.S. Su, "Long-term Predictions of Chemical Processes Using Recurrent Neural Networks: A Parallel Training Approach.", *Ind. Eng. Chem. Res.*, 1338 - 1352, 1992.

- [18] P. Nash, *Systems Modelling and Optimization*, The Institution of Electrical Engineers, London (1981).
- [19] X.W. Pan and B. Livneh, "Neural Networks to predict and Control Pyro-Metallurgical Processes.", KBE Users Group Conference, 1998.
- [20] D.A. Pierre, *Optimization Theory with Applications*, John Wiley & Sons Inc., New York (1969).
- [21] J.G. Rau, *Optimisation and Probability in Systems Engineering*, Van Nostrand Reinhold Company, New York (1970).
- [22] M.A. Reuter, T.J. van der Walt and J.S.J. van Deventer, "Modelling of Metal-Slag Equilibrium Processes Using Neural Nets.", *Metallurgical Transactions B*, Volume 23B, 643 - 650, October 1992.
- [23] E.J. Saarelainen, P.J. Saarinen, M. Veistaro and A. Visa, "Prediction of Nozzle Clogging - A Neural Computing Approach.", 1999 *Steelmaking Conference Proceedings*, 89 - 95.
- [24] D.E. Seborg, *Advances in Control: Highlights of ECC '99*, Springer-Verlag, London, 1999.
- [25] H. Singh, N.V. Sridhar and B. Deo, "Artificial neural nets for prediction of silicon content of blast furnace hot metal.", *Steel Research* 67, No. 12, 521 - 526, 1996.
- [26] S.B. Singh, H.K. Bhadeshia, D.J.C MacKay, H. Carey and I. Martin, "Neural network analysis of steel plant processing.", *Ironmaking and Steelmaking*, Vol. 25, No. 5, 355 - 365, 1998.

Chapter 4

Neural Networks

4.1 Introduction

Neural networks are one of the newer artificial intelligence methodologies available today. They were first developed by researchers working in the biological, physiological, and psychological areas as a means of modelling biological neural systems [1]. Neural networks are philosophically based on the network of biological neurons in the human brain and as such simulate the highly interconnected, parallel computational structure with many relatively simple, individual processing elements of the brain. The brain is a complex non-linear processing unit that can perform many tasks that a computer will probably never be able to. A comprehensive explanation on the history and development of neural networks can be found in references [1] and [3].



4.1.1 Types of networks

Today there are various types of neural network available classified by their particular topology. The most popular networks remain the Hopfield networks, Kohonen self organising maps, recurrent networks, multi-layer perceptrons, and backpropagation networks. This last type of network is the best known and most widely used neural network. For the purposes of this study we will concentrate on the backpropagation network, and its application in controller situations.

4.1.2 Classification

Neural networks can be classified as either feedforward or feedback networks [1] and [3]. A feedforward network is a network that feeds information in only one direction, from input to output without any feedback pathways in the network. Examples of feedforward networks are: multi-layer perceptrons and backpropagation networks. A feedback network is a network with feedback paths,

where a feedback path is defined as any path through the network that would allow the same processing element (PE) to be visited two or more times. Recurrent networks and Hopfield networks are examples of feedback networks.

4.1.3 Learning neural networks

The most appealing quality of neural networks for application specialists is their ability to learn. Neural networks learn by adjusting the weights between neurons. Weights are adjusted to minimise the error between the predicted value and the actual value. Thus, by exposing a neural network to a series of inputs and the related outputs, the network learns in a manner similar to biological networks, by adjusting the strength of connections between neurons. Learning can either be supervised or unsupervised [1]. Supervised learning is a process that utilises an external teacher and/or global information. Unsupervised learning, also referred to as self-organisation, incorporates no external teacher or supervisor and relies only upon local information during the entire learning process.



4.1.4 Applications

There are five application areas for which neural networks are generally considered to be best suited [1] and [3]. The first area is classification. An example is a decision whether or not a given segment of EEG data represents an epileptiform spike waveform or not. The second area is often referred to as constant addressable memory, or associative memory. A typical example is obtaining the complete version of a pattern as the output of the network by providing a partial version as the input. The third area is referred to as either clustering or compression. An example is the significant reduction of the dimensionality of an input, as in the case of speech recognition. The fourth area is somewhat different from the first three in that no classification is involved. It involves the generation of structured sequences or patterns from a network trained to examples. An example is a network trained to simulate or model a process. The fifth area is the use of neural networks in control systems, presently one of the fastest growing areas for various reasons.

4.1.5 Benefits of neural networks

A neural network derives its computing power through, firstly, its massively parallel distributed structure and, secondly, its ability to learn and therefore generalise. Therefore, it is clear that neural networks must offer some advantages over more traditional techniques [3]. Some of the advantages are: non-linearity, input-output mapping, adaptivity, evidential response, contextual information, fault tolerance (robustness), uniform analysis and design, and neurobiological analogy.

4.2 The human brain

A conceptual diagram of a biological neuron can be viewed in figure 4.1. A neuron consist of a cell body named the soma, dendrites through which signals access the cell body and axons through which signals exit the cell. Note that the signal flow goes from left to right, from the dendrites through the cell body, and out through the axon. The axon of one cell is connected to the dendrites of other cells. This connection is called a synapse.

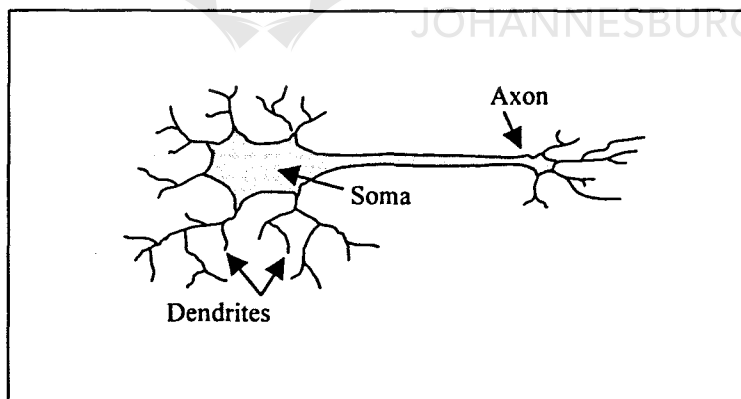


Figure 4.1: A biological neuron.

When the human brain thinks, an electrochemical signal activates the cell body. According to the strength of the signal a connection is made between activated neurons. The strength of the connection can vary between different cells. It is this basic concept of the human brain that neural networks simulate.

4.3 Artificial neuron

The artificial neuron has inputs representing the dendrites of a biological neuron, a cell body where the inputs are processed and an output representing the axon of a biological neuron. The most commonly used neuron model is depicted in figure 4.2, and is based on the model proposed by McCulloch and Pitts in 1943 [6]. Each neuron input x_1, x_2, \dots, x_n is weighted by the weights w_1, w_2, \dots, w_n . A bias or offset in the node is characterised by an additional constant input of 1 weighed by the value w_0 . The output y is obtained by summing the weighted inputs to the neuron and passing the result through a non-linear activation function, $f()$, where $f()$ typically is a sigmoid function.

$$y = f\left(\sum_{i=1}^N w_i x_i + w_0\right) \quad (4.1)$$

$$f(z) = \frac{1}{1 + \exp^{-az}} \quad (4.2)$$

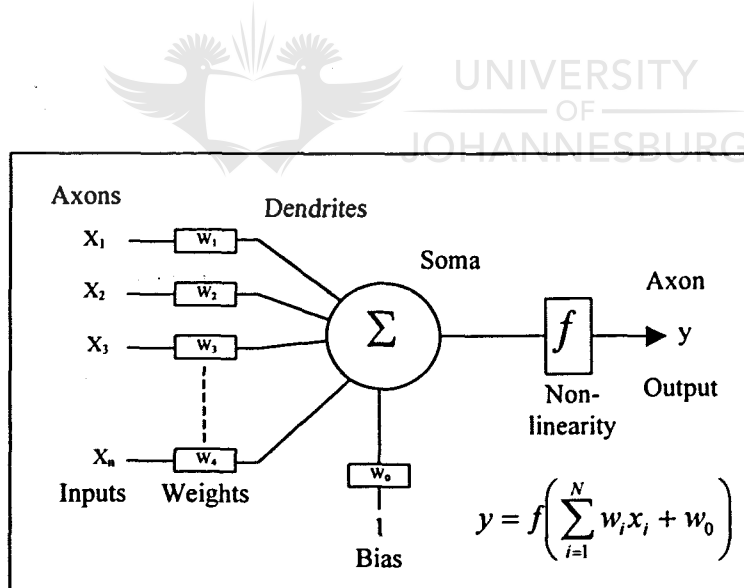


Figure 4.2: Artificial neuron.

An activation function maps a processing element's infinite domain to a prespecified range. Although the possible number of activation functions is infinite, there are four regularly employed by a majority of neural networks: the linear function, the step function, the ramp function and the sigmoid function [1], [2], [3], [5] and [8]. With the exception of the linear function all of these functions introduce a non-linearity into the network dynamics by bounding the output values within a fixed range. Each of the four activation functions is briefly discussed below in table 4.1.

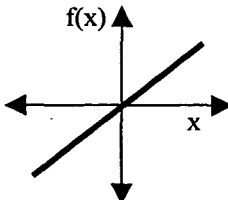
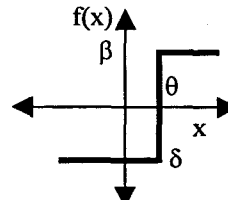
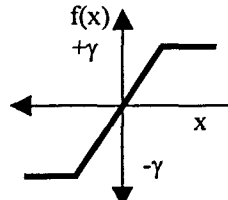
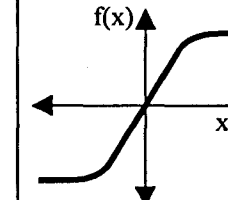
Linear Function	Step Function	Ramp Function	Sigmoid Function
			
$f(x) = \alpha x$	$f(x) = \begin{cases} \beta & \text{if } x \geq \theta \\ \delta & \text{if } x \leq \theta \end{cases}$	$f(x) = \begin{cases} \gamma & \text{if } x \geq \gamma \\ x & \text{if } x < \lambda \\ -\gamma & \text{if } x \leq -\gamma \end{cases}$	$f(x) = \frac{1}{1 + e^{-\alpha x}}$

Table 4.1: Some common activation functions.

4.4 Artificial neural model

The biological neural network consists of nerve cells (neurons) as in figure 4.1, which are interconnected. Neural activity passes from one neuron to the other in terms of electrical pulses which travel from one cell to the other down the neuron's axon, by means of an electrochemical process of vaulted gated ion exchange along the axon and of diffusion of neurotransmitter molecules through the membrane across the synaptic gap.

Artificial neural networks are, as their name indicates, computational networks, which attempt to simulate the network of biological neurons in the human brain. As with biological networks, the

neurons of an artificial network are interconnected. A very simple base algorithmic structure lies behind the neural network, but it is one, which is highly adaptable to a broad range of problems.

4.5 Multi-layer perceptron

The most popular neural network architecture is the multi-layer perceptron [1], [2], [3], and [8]. The network consist of an input layer, a number of hidden layers (typically only one or two are used) and an output layer as shown in figure 4.3. The output and hidden layer are made up of a number of nodes as described in section 4.4 above. Sigmoidal activation functions for the nodes in the hidden layer and output layer are the most common choice, although variants are possible. The outputs of each node in a layer are connected to the inputs of all nodes in the subsequent layer. Data flows through the network in one direction only, from input to output, hence this type of network is called a feedforward network.

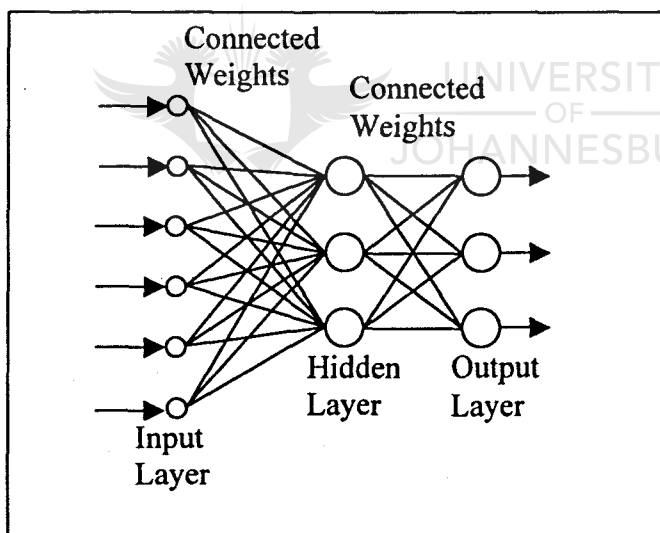


Figure 4.3: A multi-layer perceptron with one hidden layer.

The network is trained in a supervised fashion. This means that in training both the network inputs and required or target outputs are used. A number of algorithms for training the multi-layer perceptron have been proposed and the most popular remains the backpropagation algorithm. The backpropagation algorithm was proposed in 1986 by Rumelhart, Hinton and Williams [10], for setting weights and hence for the training of multi-layer perceptrons. Briefly, with this algorithm, a set of input and corresponding output data is collected that the network is trying to learn. The output

is compared to the target output and an error is produced. The error is then propagated back through the network, from output to input, and the network weights are adjusted in such a way to minimise a cost function, typically the sum of the errors squared. This procedure is repeated through all the data in the training set before the cost function is reduced to a sufficient value. A more in-depth discussion of the mathematics underlying the backpropagation algorithm can be found in [1], [2], [3], [4] and [7].

4.6 Process Insights

Process Insights is a software package developed by the company Pavilion Technologies, which is based in Austin, Texas. *Process Insights* is a very powerful optimisation and modelling software tool that is used in applications worldwide. The software makes use of a combination of neural networks, fuzzy logic, chaos theory and statistical methods to build accurate and non-linear process models based on enormous historical data sets [9].

Models are developed by following a series of processing functions: *Format*, *Pre-process*, *Model*, *Analyse* and *Setpoints & What Ifs*. All of these functions correspond to the buttons on the main window of *Process Insights* as shown in figure 4.4.

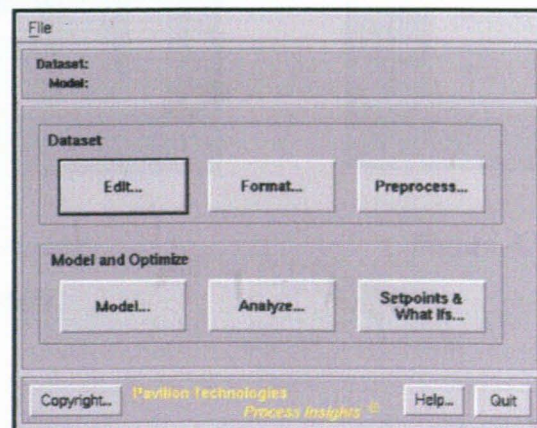


Figure 4.4: *Process Insights* main processing function window.

4.6.1 Edit and Format

The formatter is used to describe the format of the data files. The format function is extremely versatile, but occasionally a data file contains something the formatter cannot handle. In these rare cases the Edit function can be used to modify the raw data file.

4.6.2 Pre-process

The pre-processor is used to read the data files and examine and transform data as necessary. The data is displayed in a spreadsheet or as a variety of plots as can be viewed in figure 4.5 and figure 4.6. In the pre-processor data can be clipped to specific values, cut from the dataset or be changed to any specified value. New data can be derived from a variety of mathematical transforms operating on existing data values.

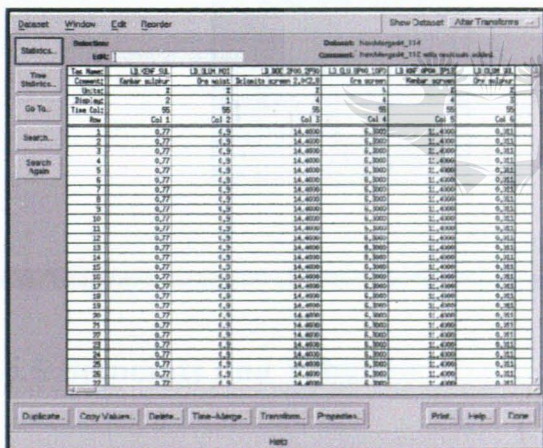


Figure 4.5: Pre-processor spreadsheet window.

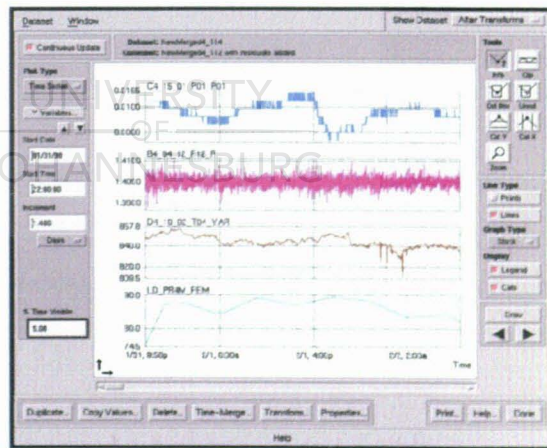


Figure 4.6: Pre-processor plot window.

4.6.3 Model

Modelling a process is done in two steps: building a model, and training it. A model is built by specifying which variables are inputs and which are outputs of the process, and identifying any time delay relationships that may exist among the variables. When the model is trained, *Process Insights* uses the historical data to enable the model to simulate the process.

4.6.4 Analyse

The analysis functions serve two purposes: to check the fidelity of the models after they have been trained, and to perform process analyses. The *Analyse* facility provides a set of *Predicted vs. Actual* functions, which allows evaluation of how well the model predicts the process behaviours.

For process analysis, the facility calculates the sensitivity of each output variable to each input variable, and generates response curves for each input. These facilities allow evaluation as to how the process will react to changes in setpoint values and disturbances acting on the process.

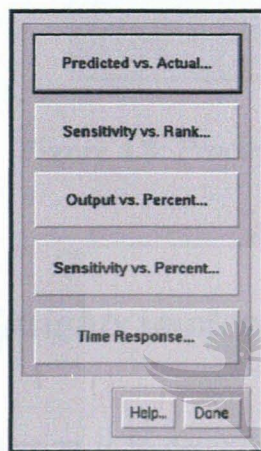


Figure 4.7: *Process Insights* analyse window.

4.6.5 Setpoints and What Ifs

The *Setpoints* function allows the determination of optimal control setpoints while the *"What Ifs"* allow a wide variety of *"What If"* experiments to be performed. Capabilities of the optimisation function include economic optimisation, desired values, hard or fuzzy constraints, and combinational constraints.

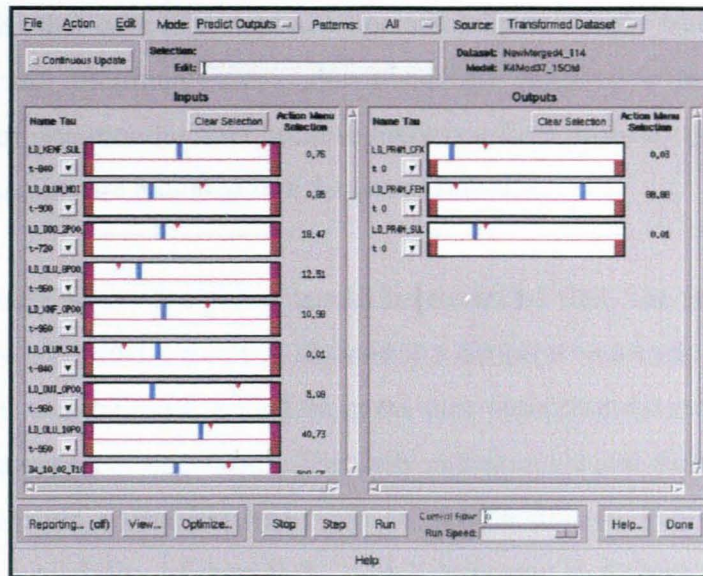


Figure 4.8: *Process Insights* Setpoint & "What If" window.

4.7 *Process Insights* implementation

The aim of this section is to describe the implementation of *Process Insights*. It gives a detailed description of the method followed to build, train, and evaluate models. This section describes the implementation process from the data and variable selection stages through the pre-processing and modelling stages, and finally the testing and evaluation stages.

4.7.1 Data selection

Before modelling could begin, data had to be selected. All the data of the direct reduction plant are stored in a SYBASE real-time database. The database has the capacity to store six months' data for each tag measured on the plant. Data older than six months are stored on a mass storage system called an historian. Thus, enough plant data are available for modelling purposes.

The data used in the modelling process had to satisfy certain selection criteria. First, and most important, all the process dynamics must be contained in the dataset. It would be a waste of resources to build a model on data that does not contain all the different process dynamics, since

such a model would not be able to predict process outputs accurately. The second requirement was that the data should be that of a kiln that was operationally relatively stable over a certain time period to ensure that the minimum process downtimes are included in the data. Further, it was decided to make use of one month's data because there is a high probability that all or most of the important process dynamics are included in a dataset of this size.

In order to decide which of the four kilns would be modelled first, the production manager and process engineers were consulted. From this discussion a decision was made to model kiln 4. Kiln 4 was selected because it was the most stable kiln in the time preceding the modelling. The next issue was to select data for a period of one month. The daily management and shift reports for kiln 4 were studied. From these reports a decision was made to use the data of February 1998. During this month kiln 4 had an availability of over 99%, which indicated that there were very few process disturbances during this production period.

The data of all the tags for kiln 4 for the month of February 1998 was downloaded from the SYBASE database onto the engineering workstation on which the modelling was to be done. The data were imported into *Process Insights* where they were formatted and saved as a dataset.

4.7.2 Variable selection

The next step in the modelling process was to select all the variables thought to have an influence on the process, and especially on the product qualities. The data downloaded from the database contained more than 250 variables for kiln 4. A list containing all the variables thought to be important or that could possibly have an influence on the process was made. The list was distributed to all technicians, engineers, and managers, who went through the list and added any other variables they thought should be included. After this process a final list was drawn up containing one hundred and nine variables.

With the list completed the next step was to select which variables would be inputs to the model, and which variables outputs. It was decided that the most logical outputs for the model would be the product quality measures viz.; degree of metallisation, sulphur content, and carbon content. All the remaining variables were classified as inputs to the model. The inputs consisted of raw material

feed rates, air and gas flow rates, chemical analyses of the raw materials, screen analyses of the raw materials, kiln and boiler pressures, kiln and boiler temperatures, sprinkler water flow rates, steam variables, kiln and cooler speeds, kiln and cooler main drive power, and product temperatures. A table with all the input and output variables can be viewed in Appendix A.

4.7.3 Process Insights pre-processing

Before any of the models could be built and trained, the data had to be pre-processed. The pre-processing stage is one of the most important stages in the modelling process. It is important because the simple rule of "garbage in, garbage out" is applicable. If a model is trained using data containing garbage, the end result would be a model that produces garbage as results. Thus, it is very important to remove all invalid data that represents downtime, abnormal operating conditions, or data reported incorrectly.

The following iterative procedure was followed to check the validity of the data for each variable in the dataset. The first step was to check the statistics of the data for each variable. The statistics showed the total number of patterns, number of valid patterns, minimum value, maximum value, mean value, and standard deviation for each variable in the dataset, as can be seen in figure 4.9.

Col	Variable	Total	Valid	Min	Max	Mean	Std Dev
1	merged_time	40321	40321	01/31/98 22:00:00	02/29/98 22:00:00		
2	CI_PRT10	40321	39524	0.449	0.514	0.47718	0.01295
3	LD_BUPT_M01	40321	40321	3.6	9.0	5.253	0.549
4	LD_BUPT_RSH	40321	40321	16.0	17.0	15.753	0.424
5	LD_BUPT_VOL	40321	40321	26.7	29.4	27.579	0.441
6	LD_BUPT_CFX	40321	40321	54.9	56.4	55.643	0.422
7	LD_BUPT_SUL	40321	40321	0.80	1.06	0.9244	0.0567
8	LD_KENF_M01	40321	40321	4.4	11.3	9.074	1.341
9	LD_KENF_RSH	40321	40321	12.8	15.1	14.540	0.667
10	LD_KENF_VOL	40321	40321	22.9	25.2	24.067	0.537
11	LD_KENF_CFX	40321	40321	60.2	62.8	61.359	0.710
12	LD_KENF_SUL	40321	40321	0.54	0.89	0.7724	0.0572
13	LD_OLUH_M01	40321	40321	0.5	1.6	0.976	0.266
14	LD_OLUH_PETot	40321	40321	66.6	66.9	66.318	0.329
15	LD_OLUH_SUL	40321	40321	0.006	0.023	0.01282	0.00406

Figure 4.9: Variable statistics in *Process Insights*.

From the statistics one can determine if there are any outliers. It is easy to detect an outlier if for example the maximum value is well outside the normal operating range for the specific variable, or if the standard deviation is very large. The amount of missing data or data containing errors can also be detected by inspecting the statistics on total and number of valid patterns.

The second step was to inspect the date/time statistics. The date/time statistics showed figures like the earliest date and time, latest date and time, time intervals, number of intervals, and number of invalid intervals. Figure 4.10 shows a window containing date/time statistics in *Process Insights*. From the date/time statistics one could determine if there were any significantly large time gaps in the dataset.

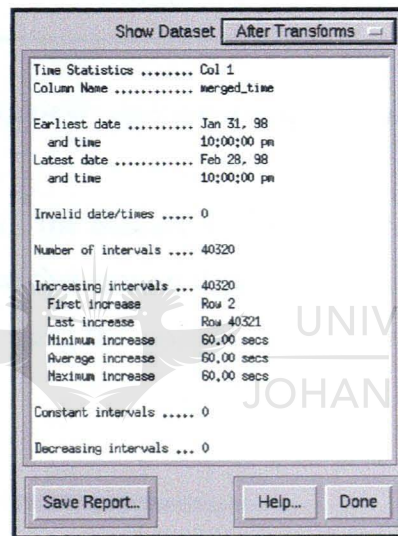


Figure 4.10: Date/time statistics for variables in *Process Insights*.

The third step was to look at a graphical representation of the data in the plot window. In this view outliers can be identified and investigated as can be seen in figure 4.11.

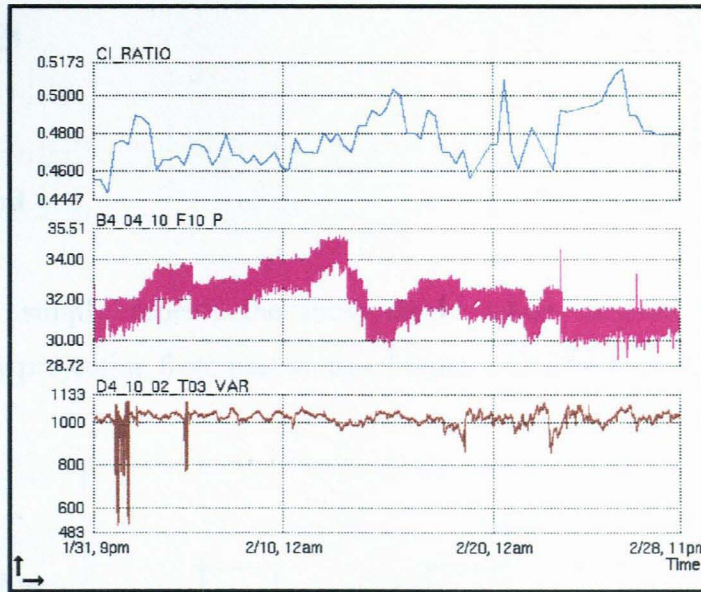


Figure 4.11: Graphical representation of data in *Process Insights*.

The final step was to examine all the outliers identified in the three preceding steps. If a valid reason could be found to confirm that certain data patterns were outliers, then, that specific data could be deleted from the dataset using the tools provided by *Process Insights*.

4.7.4 Time merging data

Before a dataset can be used in a model, it must be made row synchronous (that is, all data in a single row of the file must be associated with the same sampling time). For example, kiln temperatures are measured at one minute intervals and product qualities are measured at four hour intervals. To prepare a dataset for modelling purposes the data of these two variables has to be merged on the same time interval. Time merging is a process that converts a dataset to this required format, by expanding and/or compressing the data as necessary.

When doing a time merge, one has to specify the merge interval (that is, the sampling interval), the maximum time gap allowed in the dataset (that is the maximum time gap that would be filled by the time merge function), and the method used to expand and/or compress the data. *Process Insights* has five different methods to time merge data with. The two methods used most often are the *Boxcar* method and the *Linear extend* method. The datasets used in the modelling process were

merged on a 1 minute interval with a maximum time gap of 12.5 hours, using both the *Boxcar* and *Linear extend* methods.

4.7.4.1 Boxcar method

The *Boxcar* method simply repeats the most recent value. *Boxcar* extrapolation estimates intermediate values by projecting from past values. Figure 4.12 illustrates the *Boxcar* method.

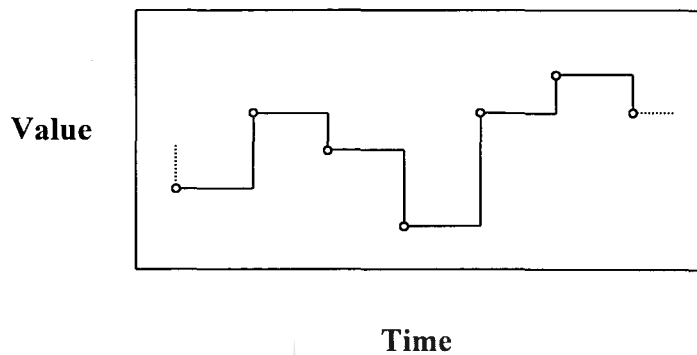


Figure 4.12: The *Boxcar* time merge method.

4.7.4.2 Linear Extend method

The *Linear extend* method interpolates between two points. *Linear extend* interpolation estimates intermediate values using future as well as past values. Figure 4.13 illustrates the *Linear extend* method.

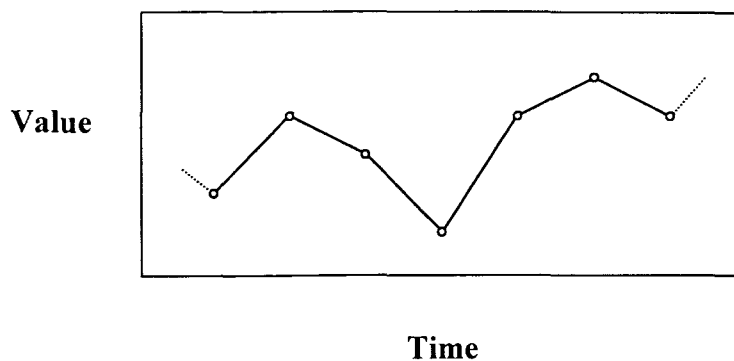


Figure 4.13: The *Linear extend* time merge method.

4.7.5 Modelling

This section explains how to build and train models using the data that has been pre-processed. Modelling a system is done in two steps: creating a model, and training it. To create (build) a model from a dataset one specifies which variables are inputs and which are outputs, identifies any time delay relationships that may exist among the variables, and select portions of the dataset for training, testing, and validation sets. When training is initiated, *Process Insights* uses the historical data to learn to simulate a process.

4.7.5.1 Model type selection

Models can be used to predict process behaviour, or to create setpoints required to produce desired outputs. The type of model that one should create depends both on how one desires to use the model, and on the types of variables that exist in the process. The four model types that are available are *Prediction*, *Control*, *Sensor Validation*, and *Custom*. The two most commonly used types are *Prediction*, and *Control*. The names are suggestive of the type of operations for which they are normally used; but a *Prediction* model can be used for control if all of the inputs to the model are independent, user-manipulated variables. A *Prediction* type was selected for the modelling in this study because it is recommended that this type of model is built first, and if it shows that a process can be simulated, other models are built from this model.

4.7.5.2 Input/Output selection

The input and output variables were specified as explained in section 4.7.2. Thus, one hundred and six variables were specified as inputs and three variables were specified as outputs.

A table with all the input and output variables can be viewed in Appendix A.

4.7.5.3 Specifying time delays

To identify the temporal relationships that existed among variables, the researcher consulted the plant's process engineers, plant operators, and the instrumentation manager. Time delays for all variables were specified in hourly intervals. Because time delays between variables are not fixed, a series of delays were specified for each variable, as illustrated in figure 4.14. In *Process Insights*, if a variable has four time delay values specified, then each of those time delays are interpreted as an independent input to the model. Thus, one variable with four time delays specified has four inputs to the model. After a model is trained, a *Sensitivity vs. Rank* analysis can be performed on the model. From this analysis, the time delay most sensitive to the process can be identified. Then all the remaining time delays for a specific variable can be deleted, keeping only the most sensitive delay.

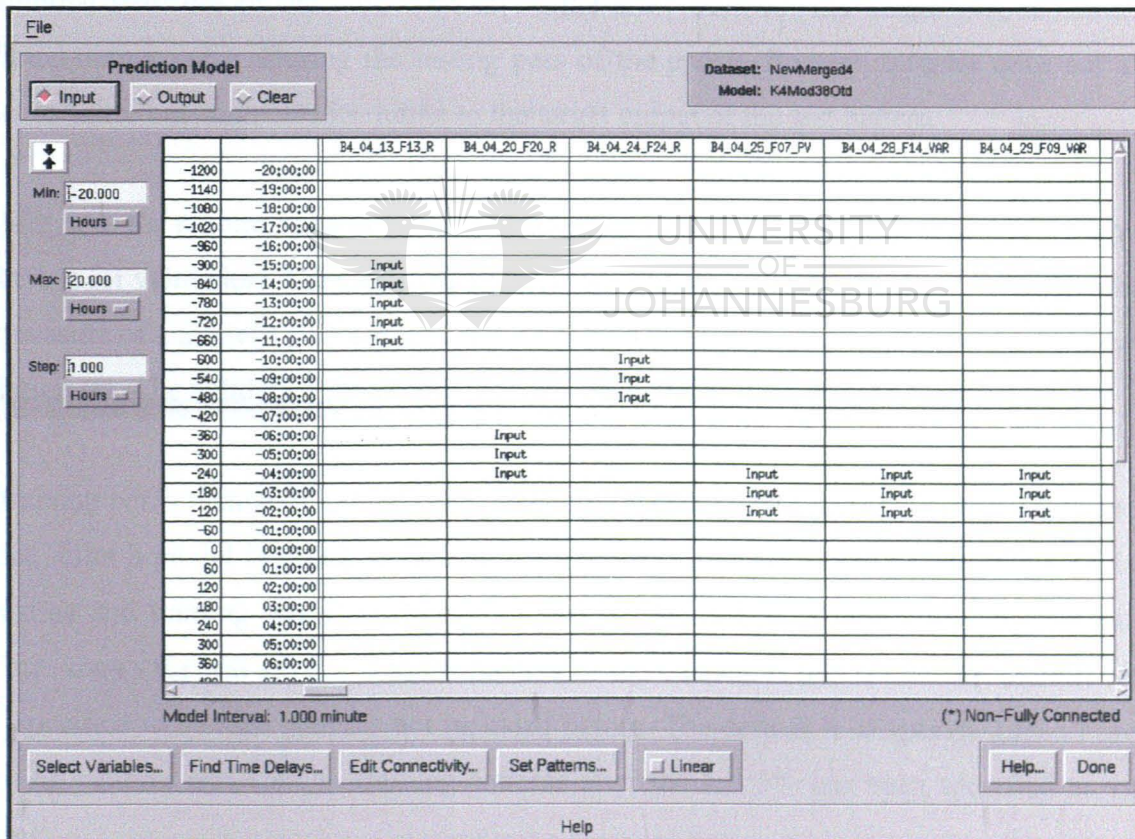


Figure 4.14: Time delay specification.

Process Insights has a function called *Find Time Delays*, which uses a non-linear correlation algorithm to automatically calculate the time delays between input and output variables. This function was used to build some of the models.

4.7.5.4 Setting model patterns

When a dataset is used in a model, each row of the dataset must be a single sample of all the input and output variables for the process at a given point in time. A data pattern is a complete set of inputs and outputs as presented to the model. If there are time delays among the variables, a data pattern will contain values gathered from different rows in the dataset.

In order to train a model, the dataset must be apportioned into training patterns and testing patterns. When training is initiated, *Process Insights* begins to tune the model. It does this by automatically alternating between a training pass and a testing pass. In the training pass, the training patterns are run through the model, and then, during the testing pass, the test patterns are run through the model. Each training/testing cycle is called an epoch. During the training pass, *Process Insights* modifies the internal structure of the model based on the error between the actual output value and the predicted output value. During the testing pass of the epoch, *Process Insights* does not allow the model to learn, but compares its output to the target output of the test pattern.

In addition to the required sets of training and testing patterns, it is advisable to reserve a portion of the dataset for validation. The model never sees the validation patterns while it is being trained. The first measure of a model's performance should be how closely it calculates the output values for the validation set. The validation set should normally be a block of data at the end of a dataset.

The training performance of a model can depend on which portions of the dataset are chosen as the test set. After a model is saved, model statistics are shown. The means and standard deviations of the testing and training sets should be examined. If they differ by more than approximately 5%, then one knows the test set is not representative of the dataset. If this is the case the test set should be re-specified to include patterns not included before. The default is to specify 15% of the dataset as testing patterns and 85% as training patterns after the last 5% has been specified as validation patterns.

Figure 4.15 shows the model statistics for a typical model. It shows the total number of patterns, the average of the sum of all variable values, as well as a standard deviation. From these statistics one can determine if the same process dynamics are represented in the training and testing sets.

	Total	Train	Test	Validation
Patterns:	15874	12788	2291	795
Mean:	28.2503	28.1882	28.1321	29.5898
Std Dev:	3.3158	3.3374	3.4581	0.4894

Figure 4.15: Variables statistics for *Process Insights* models.

4.7.6 Training the model

When training a model, both numerical information and two types of plots are presented on the training performance. Training continues until it is terminated, or until the training parameters that are set cause it to terminate automatically.

4.7.6.1 Measures of training performance

When training a model, *Process Insights* computes a *Relative Error* that indicates the discrepancy between the actual output values in the dataset and the predicted output values generated by the model. A *Relative Error*, if computable, is a real number greater than 0. A *Relative Error* of 0 would indicate that a model can perfectly predict outputs from inputs. The *Relative Error* of an entire dataset is defined as:

$$\text{rel-err}_{\text{tot}} = \sqrt{\frac{\text{sq-err}_{\text{tot}}}{N_{\text{pats}} \times N_{\text{outs}} \times \sigma^2_{\text{all-outs}}}} \quad (4.1)$$

where N_{pats} is the total number of patterns, N_{outs} the total number of outputs, and $\sigma^2_{\text{all-outs}}$ is the standard deviation of all the output values. Further the squared error for the entire dataset is defined as:

$$\text{sq-err}_{\text{tot}} = \sum_{\text{pat}=1}^{N_{\text{pats}}} \text{sq-err}_{\text{pat}} \quad (4.2)$$

where the square error for each pattern is defined as:

$$\text{sq-err}_{\text{pat}} = \sum_{\text{out}=1}^{N_{\text{outs}}} \text{sq-err}_{\text{out}} \quad (4.3)$$

and the squared error for an individual output is defined as:

$$\text{sq-err}_{\text{out}} = (u_{\text{out}} - \hat{u}_{\text{out}})^2 \quad (4.4)$$

where u_{out} is the actual value and \hat{u}_{out} is the predicted value.

The *Relative Error* is not the same as the commonly-used statistical measure, R^2 (coefficient of determination). The two measures are related as follows:

$$R^2 = 1 - (\text{rel-err})^2 \quad (4.5)$$

When training begins the discrepancy between the model's predicted values and the dataset's actual values is relatively high. At this point, the model cannot predict outputs very well from the data. As the training progresses, the model continues to modify its internal structure to better represent the relationships between input variables and output variables. As the model becomes better attuned to these relationships, both testing and training *Relative error* decrease, indicating that the model can more accurately predict output values. The *Relative error* training graphs for the two models used for the results in Chapter 6 can be viewed in Appendix C.

While a model is being trained, the epoch number, *Relative Error*, and R^2 for the current and the best epoch are displayed in the train window as illustrated in figure 4.16. If the model has more than one output variable, the error measures that are displayed are a composite over all output variables.

	Current: 250		Best: 250	
	Train	Test	Train	Test
Rel Error:	0.141	0.128	0.141	0.128
R-squared:	0.980	0.984	0.980	0.984

Figure 4.16: *Process Insights* train window display.

4.7.6.2 Training types

There are three types of training: *Regular*, *Stiff*, and *Ridge*. *Regular* is the general purpose neural net trainer, using gradient descent (backpropagation). It can be used with any model. *Stiff* training is an alternative for training neural networks, using the Stiff Differential Equation algorithm developed at Du Pont. The *Stiff* trainer is too computer-intensive to be useful for problems with more than 30 total input and output variables. The *Stiff* trainer can be considerably slower than other methods and also requires more memory. The *Ridge* trainer is available only for linear models. If the model being trained is actually linear, then the *Ridge* trainer will train the model better and faster than the other two training types. For the purpose of this study all models were trained with the *Regular* trainer.

4.7.6.3 Training parameters

Before training can commence certain training parameters have to be set. These parameters include stopping criteria, auto learning rate adjustment, and sparse data algorithm. *Process Insights* has an autostop algorithm, which recognises that training has stopped improving. When the training relative error begins and continues to increase, or if it remains essentially unchanged for an extended period of time the autostop algorithm will terminate training. However, training can be terminated at any time during the training period by clicking the *Stop Training* function.

The auto learning rate adjustment parameter applies only to *Regular* training, and is ignored by the other training types. The *Regular* training algorithm essentially consists of gradient descent with added noise. Gradient descent requires a choice of step size. The auto learning rate option adjusts the step size dynamically according to how learning is progressing. When this option is turned off a fixed step size is used. On average the auto learning rate option will outperform the fixed step size.

Sparse data occurs when the model input variables are sampled frequently, but the outputs are sampled relatively infrequently; for example, one may have one minute samples of the process variables, but the output comes from a lab analysis performed only every 4 hours. The time merge function interpolates or extrapolates the output values to the time interval specified. The certainty of a value records whether it already existed before the time merge or whether it was generated, and, if generated, how far away it was from known data. If the sparse data algorithm is turned on, the

training will be weighted by certainties, where actual data has certainties of 1 and generated data far from actual data has certainties of 0, paying more attention to the actual values provided in the original data than to generated values.

For all the models trained in this study the autostop function was turned off, which allowed training until the researcher terminated the training. The auto learning rate adjustment algorithm was left on for all models trained. The sparse data algorithm was on for some models and off for others. This was done to evaluate the algorithm's effect on certain models. Table 4.3 in the next section summarises all models built and indicates which models were built with the sparse data algorithm on.

4.7.7 Model analysis

The *Analyse* functions in *Process Insights* serve two functions: to check the fidelity of models after they have been trained, and to analyse the process. The *Analyse* function provides a set of *Predicted vs. Actual* functions which allows the evaluation of how well the model predicts process behaviour over the range of data used to train the model. For analysing the process, the facility calculates the sensitivity of each output variable to each input variable, and generates response curves for each input.

The two functions mainly used in this study are the *Predicted vs. Actual* function and the *Sensitivity vs. Rank* function. A *Predicted vs. Actual* analysis is used to run a dataset through a trained model and compare the model's predicted output values with the actual values recorded in the dataset. The principal reasons for doing this are: to validate a model by running it on data that it never saw during training, to identify points that a trained model did not learn well, to view training and testing *Relative errors* and R^2 values for individual variables in a model with multiple output variables, and to append predicted values to a dataset for residuals analysis.

A *Sensitivity vs. Rank* analysis calculates the sensitivity of output variables to input variables over the patterns in the dataset, and, for each output, ranks the inputs in order of sensitivity. The *Sensitivity vs. Rank* analysis function was used as a data reduction tool, and it is explained in more detail in the next chapter.

4.7.8 Modelling methodology

The first step was to prepare two different datasets for modelling purposes. The first dataset was time merged using the *Boxcar* method while the second dataset was time merged using the *Linear extend* method. Both datasets have exactly the same pre-process transforms. Further, two data reduction methods were chosen. The first was to make use of the *Process Insights* method of doing a *Sensitivity vs. Rank* analysis, identify insensitive variables, and remove them from the dataset. The second was to make use of a statistical method called principal component analysis. This method was used to identify variables that did not account for much of the variability in the dataset. After investigation, variables found to be insignificant were deleted from the dataset. Both these data reduction methods are discussed in the next chapter.

Using the first data reduction method, it was decided to build and train five different models using the researcher specified time delays. Table 4.2 gives a summary of the five models. The first two models were built using the first dataset merged by the *Boxcar* method. The first model was trained with the sparse data algorithm off and the second with the algorithm on. The third and fourth models were build using the second dataset merged with the *Linear extend* method. Again the model were trained with the sparse data algorithm on and off for the two models. The fifth model was trained using the *Linear extend* dataset but with no time delays between variables.

MODEL NAME	DATASET	TIME MERGE METHOD	SPARSE DATA ALGORITHM	DATA REDUCTION METHOD	TIME DELAYS
K4Mod35	NewMerged3	Boxcar	Off	Process Insights	Specified
K4Mod36	NewMerged3	Boxcar	On	Process Insights	Specified
K4Mod37	NewMerged4	Linear Extend	Off	Process Insights	Specified
K4Mod38	NewMerged4	Linear Extend	On	Process Insights	Specified
K4Mod39	NewMerged4	Linear Extend	On	Process Insights	No delays
StatsMod1	NewMerged3	Boxcar	On	PCA	Specified
StatsMod2	NewMerged4	Linear Extend	On	PCA	Specified
StatsMod3	NewMerged3	Boxcar	On	PCA	Automatic
StatsMod4	NewMerged4	Linear Extend	On	PCA	Automatic

Table 4.2: Summary of models.

With the second data reduction method four models were trained. All four models were trained with the sparse data algorithm on. The first two models were trained on the first dataset (*Boxcar merged*), and the last two models were trained using the second dataset (*Linear extend merged*). The first two models were trained with researcher specified time delays, and the last two models were trained using the automatically calculated time delays.

Initially the models had quite a number of variables because of different time delays, thus training was very slow. During the first couple of rounds the models were trained for a low number of epochs. As the reduction of variables increased so the training time decreased, which allowed the models to be trained for more epochs. The final models were trained for 250 epochs. One of the models was trained for more than 2500 epochs to see if there was not a second inflection point. However, none was found and the other models were not trained further.

After each round of training a *Predicted vs. Actual* analysis was performed to evaluate model performance. Then one of the two data reduction methods was used to remove certain variables and the models were retrained. This process was repeated until no more variables could be removed from the dataset.



4.8 References

- [1] R. Eberhart, P. Simpson and R. Dobbins, *Computational Intelligence PC Tools: An Indispensable Resource for the latest in: Fuzzy Logic, Neural Networks, Evolutionary Computation*, AP Professional, Boston (1996).
- [2] D. Graupe, *Principles of Artificial Neural Networks*, World Scientific Publishing, Singapore (1997).
- [3] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Edition, Prentice Hall, New Jersey (1999).
- [4] G.E. Hinton, "How Neural Networks Learn from Experience.", *Scientific American*, 105 - 109, September 1992.
- [5] R.P. Lippmann, "An Introduction to computing with Neural Nets.", *IEEE ASP Magazine*, 4 - 22, April 1987.

-
- [6] W.S. McCulloch and W.H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity.", *Bulletin of Mathematical Biophysics*, 5, 115 - 133.
- [7] A.A. Minai and R.D. Williams, "Back-propagation Heuristics: A Study of the Extended Delta-Bar-Delta Algorithm.", Center for Semicustom Integrated Systems, Department of Electrical Engineering, University of Virginia, Charlottesville, 1595 - 1600, 1993.
- [8] G.F. Page, J.B. Gomm and D. Williams, *Applications of Neural Networks to Modelling and Control*, Chapman & Hall, London (1993).
- [9] Pavilion Technologies Inc., *Process Insights* User's Guide, Version 4.1, 1997.
- [10] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning representations of back-propagation errors.", *Nature*, London, Vol. 323, 533 - 536, 1986.



Chapter 5

Data processing

5.1 Introduction

One of the goals of this study was to investigate two different data reduction methods. When referring to data reduction, one refers to the method used to reduce the number of variables used to build and train models. The aim of this reduction process is to remove one or more variables at a time and then study the effect on model performance. The reason why one would like to reduce the number of variables to the least number possible without losing too much in performance is evident when looking at processing time, system maintenance, fault detection, data transfer, and (most important) interpretation.

When running a model, these factors can play a role in system performance and costs, especially at the implementation stage. If one has two models, one having one hundred variables and the other only fifteen, then in terms of the factors mentioned above, it is obvious that the smaller of the two models requires less processing time to perform the same activities as the other model. The smaller model also puts less strain on what already could be an overloaded computer network, by requesting less data to be transferred at frequent intervals. It is also easier to detect and correct faults on a smaller model, and therefore system maintainability is better.

The first data reduction method used was the method provided with the *Process Insights* software. This method involves using the *Sensitivity vs. Rank* analysis function provided by the software. With this method input variables are ranked according to sensitivity to the output variables. The second method used was a statistical analysis method known as principal component analysis (PCA). A principal component analysis extracts variability from the data and based on this, certain variables accounting for little or no variability are identified as possible candidates for removal.

5.2 Process Insights method

Process Insights has a series of processing functions used to develop models. The *Analyse* function is one of these functions used to check the fidelity of the models, and to perform process analysis. *Sensitivity vs. Rank* is one of the *Analyse* processing functions used to perform these analysis tasks.

A *Sensitivity vs. Rank* analysis calculates the sensitivities of output variables to input variables over the patterns in the dataset, and, for each output, ranks the inputs in order of sensitivity [8]. In the iterative modelling process, one can build a model using every input that could possibly affect the outputs, run a *Sensitivity vs. Rank* analysis and remove from the model those inputs that have negligible sensitivity, then train this reduced model. This process can be repeated until no variables with negligible sensitivities are left in the model.

There are three types of sensitivity measures viz.: *Average Absolute*, *Average* and *Peak*. *Average Absolute* sensitivity is the distribution-averaged sum of the absolute values of the partial derivatives of the input-output pairs,



$$\text{Average Absolute} = \frac{\sum_{k=1}^N \left| \frac{\partial o_{k,i}}{\partial x_{k,j}} \right|}{N} \quad (5.1)$$

Where N is the number of patterns in the dataset over which the distribution is calculated, $x_{k,j}$ is the j^{th} input for the k^{th} pattern, and $o_{k,i}$ is the i^{th} output for the k^{th} pattern.

Average sensitivity is the average of the partial derivatives,

$$\text{Average} = \frac{\sum_{k=1}^N \frac{\partial o_{k,i}}{\partial x_{k,j}}}{N} \quad (5.2)$$

Peak sensitivity is the maximum of the partials over all patterns,

$$\text{Peak} = \max \left(\left| \frac{\partial o_{k,i}}{\partial x_{k,j}} \right|, k \in 1, 2, \dots, N \right) \quad (5.3)$$

All of these measures are calculated for each pair of input and output variables. Figure 5.1 shows a graphical representation of the ranking of variables according to sensitivity for each output and table 5.1 shows an extract of a typical report generated by *Process Insights*.

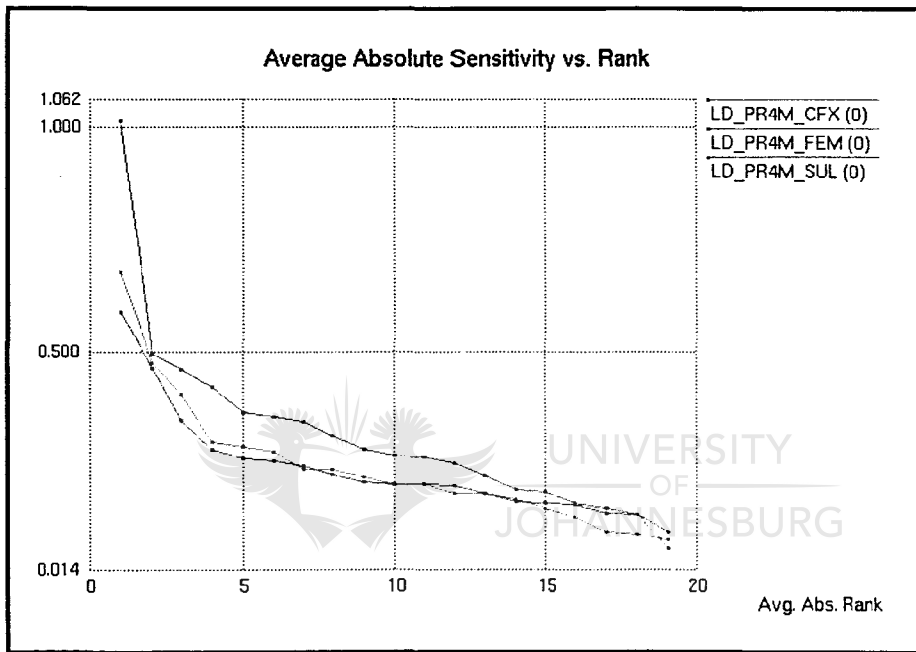


Figure 5.1: Graph showing sensitivities of input variables on output variables.

OUTPUT_NAME : LD_PR4M_CFX TAU : 0					
Rank#	input_name	tau#	Avg. Abs.	Avg.	Peak
1	CI_RATIO	-720	0.678	0.278	5.373
2	D4_12_01_T23_VAR	-120	0.474	-0.457	5.157
3	D4_10_02_T10_VAR	-240	0.403	-0.220	1.839
4	B4_04_12_F12_R	-780	0.297	0.292	1.368
5	C4_15_07_T06_PV	-840	0.286	-0.271	1.486

Table 5.1: Sensitivity values calculated by *Process Insights*.

5.3 Statistical method

A principal component analysis was the method selected to reduce the dimensionality of the dataset. Further, it was decided to use Statistica, which is a statistical software package developed by StatSoft, to do the principal component analysis with. A principal component analysis was chosen because this method was recommended in various literature sources as a data reduction method, which optimally determined the impact of variables on the determinant function.

5.3.1 Principal component analysis

Principal component analysis is a multivariate statistical method whose primary purpose is to define the underlying structure in a data matrix of metric variables. Broadly speaking, it addresses the problem of analysing the structure of the interrelationships among a large number of variables by defining a set of common underlying dimensions, known as factors. Once the dimensions and the explanation of each variable are determined, the two primary uses for principal component analysis, namely, summarisation and data reduction, can be achieved [4].

Principal component analysis is used to reduce a set of observed variables into a relatively small number of components that account for most of the observed variance. [7]. This is accomplished by mathematical linear transformation of the observed variables under two conditions. The first condition is that the first component accounts for the maximum amount of variance possible, the second the next, and so on and so forth. The second, and very important result, is that all the derived components are uncorrelated with each other.

Principal components are mathematical functions of observed variables. The PCA functions do not attempt to explain the intercorrelations among the variables, but to account for the variability observed in the data [7]. Consider an example where eight variables of interest are observed from a sample (i.e. x_1, x_2, \dots, x_8), and found to have a correlation matrix R . Using these eight observed variables, a PCA will attempt to construct linear combinations as follows:

$$Y_1 = \Gamma_1'X = \gamma_{11}x_1 + \gamma_{21}x_2 + \dots + \gamma_{81}x_8 \quad (5.4)$$

$$Y_2 = \Gamma_2' X = \gamma_{12} X_1 + \gamma_{22} X_2 + \dots + \gamma_{82} X_8 \quad (5.5)$$

$$Y_3 = \Gamma_3' X = \gamma_{13} X_1 + \gamma_{23} X_2 + \dots + \gamma_{83} X_8 \quad (5.6)$$

·
·
·

$$Y_8 = \Gamma_8' X = \gamma_{18} X_1 + \gamma_{28} X_2 + \dots + \gamma_{88} X_8 \quad (5.7)$$

or in matrix notation

$$Y = \Gamma' X \quad (5.8)$$

The Y s in these equations are the computed principal components, and the γ s are the coefficients of the observed variables. Thus, the principal components are the uncorrelated linear combinations of Y s whose variances are as large as possible. The first principal component is the linear combination with maximum variance. This basically is the component that maximises the variance Y_1 , written as $\text{VAR}(Y_1)$. However, because it is clear that $\text{VAR}(Y_1)$ can be increased by choosing any Γ_1 , a restriction that the product be equal to unity is also imposed. Thus, the first principal component is that selection of γ 's that maximises $\text{VAR}(Y_1)$ subject to $\Gamma_1' \Gamma_1 = 1$. The second principal component is the linear combination of $\Gamma_2' X$ that maximises $\text{VAR}(Y_2)$ subject to $\Gamma_2' \Gamma_2 = 1$, and ensures that the two combinations are uncorrelated. The analysis continues until all principal components are generated.

The actual coefficients of the observed variables (i.e. the γ 's), commonly referred to as principal component loadings, are determined by multiplying the eigenvectors of the observed correlation matrix by the square roots of the respective eigenvalues. The results for the component loadings are easily determined by expanding on the following relationship between any correlation matrix and its associated eigenvalues and eigenvectors:

$$R = V L V' \quad (5.9)$$

Where R is the correlation matrix, V the matrix of eigenvectors, and L a diagonal matrix with eigenvalues.

The correlation matrix can also be represented as:

$$R = V(\sqrt{L})(\sqrt{L})V' \quad (5.10)$$

Where \sqrt{L} represents the square root of the matrix with eigenvalues. This matrix can then be written as the product of two matrices, each a combination of eigenvectors and square roots of eigenvalues. Thus,

$$R = (V\sqrt{L})(\sqrt{L}V') = PP' \quad (5.11)$$

Where $P = V\sqrt{L}$ is the product of eigenvectors and square roots of the eigenvalues. The mathematical technique for finding the matrices V and L is the eigenvalue equation. For a more extensive explanation of the mathematics underlying a principal component analysis, the following references can be consulted [1], [2], [3], [5], [6], and [9].

5.3.2 Criteria for the number of components to extract

When a large set of variables is analysed, the method first extracts the combinations of variables explaining the greatest amount of variance and then proceeds to combinations that account for smaller and smaller amounts of variance. In deciding when to stop factoring (that is, how many components to extract), the researcher generally begins with some predetermined criteria, such as Kaiser criteria or percentage of variance criteria, to arrive at a specific number of factors to extract. By analogy, choosing the number of factors to be extracted is like focussing a microscope. Too high or too low an adjustment will obscure a structure that is obvious when the adjustment is just right. Therefore, by examining a number of different factor structures derived from several trial solutions, the researcher can compare and contrast to arrive at the best representation of the data. For the purposes of this study the two selection criteria used was the Kaiser criterion, and the Scree test criterion.

5.3.2.1 Kaiser criterion

The most commonly used technique is the Kaiser criterion. This technique is simple to apply to principal component analysis. The rationale for the Kaiser criterion is that any individual factor should account for the variance of at least a single variable if it is to be retained for interpretation. Each variable contributes a value of 1 to the total eigenvalue. Thus, only the factors having eigenvalues greater than 1 are considered significant, all factors with eigenvalues less than 1 are considered insignificant and are disregarded. Table 5.2 shows the eigenvalues for a typical analysis. From this example one can see that a Kaiser criterion would retain five factors, which accounts for 78% of the total variance.

EIGENVALUES				
Extraction: Principal components				
	Eigenval	% Total Variance	Cumul. Eigenval	Cumul. %
1	8.892157	37.05066	8.89216	37.05066
2	4.167708	17.36545	13.05987	54.41611
3	2.652485	11.05202	15.71235	65.46812
4	2.067032	8.61264	17.77938	74.08076
5	1.117678	4.65699	18.89706	78.73775
6	0.958804	3.99502	19.85586	82.73277

Table 5.2: Eigenvalues for principal component analysis.

5.3.2.2 Scree test criterion

The Scree test is used to identify the optimum number of factors that can be extracted before the amount of unique variance begins to dominate the common variance structure. The Scree test is derived by plotting the eigenvalues against the number of factors in their order of extraction, and the shape of the resulting curve is used to evaluate the cut-off point. Figure 5.2 plots the first 20 factors in a typical component analysis. Starting with the first factor, the plot slopes steeply downward initially and then slowly becomes an approximately horizontal line. The point at which

the curve begins to straighten out is considered to indicate the maximum number of factors to extract. In the example, the first 5 factors would qualify

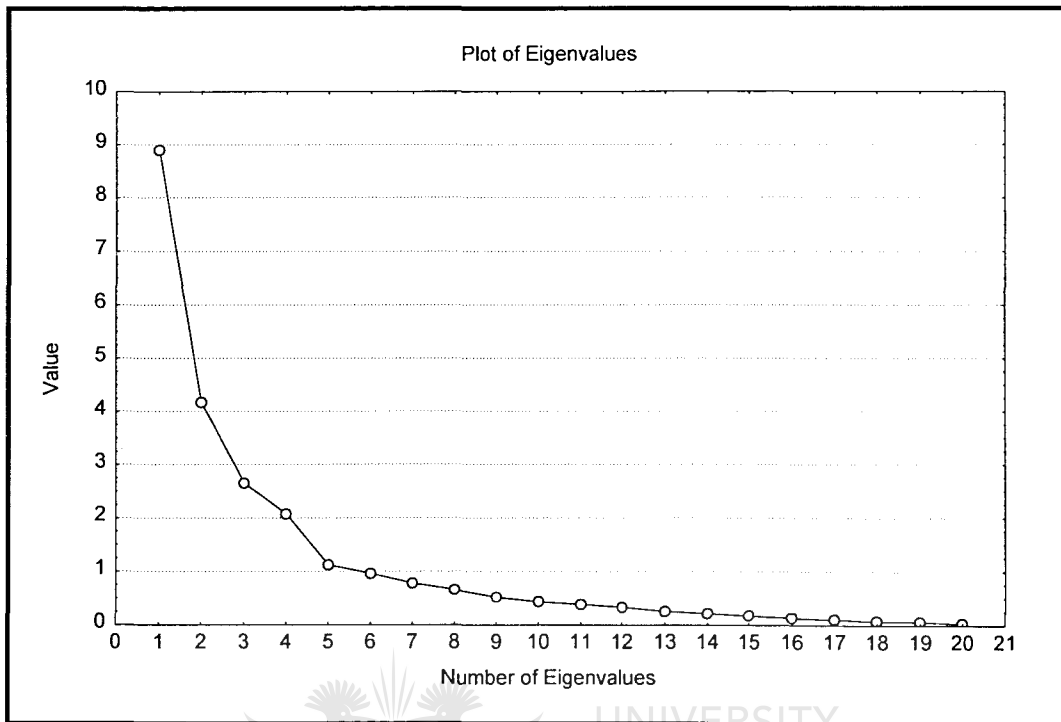


Figure 5.2: Eigenvalue plot for Scree test criterion.

5.3.3 Interpreting a factor matrix

Interpreting the complex interrelationships represented in a factor matrix is no simple matter. By following a simple procedure, one can considerably simplify the factor interpretation procedure. The first step is to examine the factor matrix of loadings. Each column of numbers in a factor matrix represents a separate factor. The columns of numbers are the factor loadings for each variable on that factor. Factors are numbered 1, 2, 3, 4 and so forth from left to right at the top of each column, and variables are numbered from top to bottom at the left side of each row.

The second step is to identify the highest loading for each variable. The interpretation should start with the first variable on the first factor and move horizontally from left to right, looking for the highest loading for that variable on any factor. When the highest loading is identified it should be

underlined if significant. This procedure should be continued for each variable until all variables have been underlined once for their highest loading on a factor.

When each variable has only one loading on one factor that is considered significant, the interpretation of the meaning of each factor is simplified considerably. In practice, however, many variables may have several moderate size loadings, all of which are significant, and the job of interpreting the factors is much more difficult. Thus, the researcher will, after underlining the highest loading for a variable, continue to evaluate the factor matrix by underlining all significant loadings for a variable on all factors. Ultimately the objective is to minimise the number of significant loadings on each row of the factor matrix. A variable with several high loadings is a candidate for deletion.

Once all the variables have been underlined on their respective factors, the researcher should examine the factor matrix to identify variables that have not been underlined, and therefore do not load on any factor. The researcher should view each variable's communality to assess whether it meets acceptable levels of explanation. For example, a researcher may specify that at least one-half of the variance of each variable must be taken into account. Using this guideline, the researcher would identify all variables with communalities less than .50 as not having sufficient explanation. If there are variables that do not load on any factor or whose communalities are deemed too low, then these variables can be evaluated for possible deletion.

5.3.3.1 Rotation of factors

An important tool in interpreting factors is factor rotation. The term "rotation" means exactly what it implies. Specifically, the reference axes of the factors are turned about the origin until some other position has been reached. Unrotated factor solutions extract factors in order of their importance. The first factor tends to be a general factor with almost every variable loading significantly, and it accounts for the largest amount of variance. The second and subsequent factors count for successively smaller portions of variance. The ultimate effect of rotating the factor matrix is to redistribute the variance from earlier factors to later ones to achieve a simpler, theoretically more meaningful factor pattern.

The VARIMAX rotation criterion centres on simplifying the columns of the factor matrix. With the VARIMAX rotational approach, the maximum possible simplification is reached if there are only 1's and 0's in a column. That is, the VARIMAX method maximises the sum of variance of required loadings of the factor matrix. With the VARIMAX rotation approach, there tends to be some high loadings and some low loadings in each column of the matrix. The logic is that interpretation is easiest when the variable-factor correlations are (1) close to either +1 or -1, thus indicating a clear positive or negative association between the variable and the factor; or (2) close to 0, indicating a clear lack of association.

5.4 Interpretation methodology used in this study

A principal component analysis was chosen to identify variables for possible deletion. Variables identified for possible deletion were evaluated according to a certain criterion. Once the criterion confirmed that certain variables could be deleted from the dataset, those variables were deleted and a new dataset was created without the deleted variables. Thus, a new dataset with reduced dimensionality was created. The process that was followed in determining which variables to delete from the dataset is described in the following paragraphs.

The first step was to extract a factor loading matrix for the dataset containing all the original variables. The next step was to examine the factors according to the Kaiser and Scree test criterion. From this examination the number of significant factors to be used was determined. The researcher used the number of factors indicated by one of the two criterion methods to be the highest. Thus if the Scree test criterion indicated more significant factors than the Kaiser criterion, then the number of factors indicated by the Scree test was used for analysis purposes and vice versa.

Once the number of significant factors was identified, the factor loadings were examined according to the method explained in section 5.3.3. After all the highest and other significant factor loadings were underlined, all the variables with no significant loadings were identified. The factor loadings were then rotated according to the VARIMAX rotation criterion and again the variables with no significant loadings on important factors were identified. The researcher studied the communalities of those variables and confirmed if variables did not have sufficient explanation because of communalities of less than .50. These variables were deleted from the dataset and a PCA was

performed on the new dataset. Thus, only variables performing poorly in both the unrotated and rotated factor loadings were identified for deletion. This process was repeated until no more variables with insignificant loadings were left in the dataset.

The next step was to examine all the variables with more than one significant factor loading on factors. For the first five rounds a factor loading of 0.40 was taken to be significant, the next four rounds a value of 0.45, the next six rounds a value of 0.50, and the final rounds a value of 0.55. Thus, as the number of variables was reduced, so the loading value considered to be significant increased. Again the factor loadings were examined using both the unrotated and rotated criterion. During this phase variables with more than one significant loading on the factors were removed systematically. Initially only variables with four or more loadings were examined. Once these variables had been removed, then variables with three or more loadings were examined and so forth until only variables with one significant loading on one of the extracted factors remained.

Thus, from the description it is clear that variables were first identified, then they were examined to confirm if they could be removed, and once they were confirmed to be deleted, they were removed from the dataset. This was an iterative procedure where only a few variables were removed after each analysis.

For the *Process Insights* method an absolute average sensitivity of 0.1 was seen as insignificant. Thus, using this criterion, variables with absolute average sensitivities of less than 0.1 were deleted from the dataset. New models were built and trained with the new datasets. After each model was trained, a *Sensitivity vs. Rank* analysis was performed and insignificant variables were deleted from the dataset. This process was repeated until no insensitive variables remained.

5.5 References

- [1] C. Chatfield and A.J. Collins, *Introduction to Multivariate Analysis*, Chapman and Hall, London (1980).
- [2] W.W. Cooley and P.R. Lohnes, *Multivariate Data Analysis*, John Wiley & Sons, New York (1971).

- [3] R. Gnanadesikan, *Methods for Statistical Data Analysis of Multivariate Observations*, John Wiley & Sons, New York (1977).
- [4] J.F. Hair, R.E. Anderson, R.L. Tatham and W.C. Black, *Multivariate Data Analysis*, fifth edition, Prentice Hall, New Jersey (1998).
- [5] R.A. Johnson and D.W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall, New Jersey (1992).
- [6] D.N. Lawley and A.E. Maxwell, *Factor Analysis as a Statistical Method*, The Butterworth Group, London (1971).
- [7] G.A. Marcoulides and S.L. Hershberger, *Multivariate Statistical Methods - A first course*, Lawrence Erlbaum Associates, New Jersey (1997).
- [8] Pavilion Technologies Inc., *Process Insights User's Guide*, Version 4.1, (1997).
- [9] M.S. Srivastava and E.M. Carter, *An Introduction to Applied Multivariate Statistics*, Elsevier Science Publishing Co., New York (1983).



Chapter 6

Results

6.1 Introduction

The results obtained by investigating the various models developed in the study are summarised in this chapter. The models that were built and trained can be divided into two main categories according to the variable reduction method used by these models. The first category contains those models built making use of the *Process Insights'* sensitivity analysis technique, and the second category contains those models built using a principal component analysis to reduce the number of variables.

Further, in each category different models were built and trained based on the time-merge method used in the datasets (*Boxcar* or *Linear extend*, Chapter 4, paragraph 4.7.4), with the sparse data algorithm on or off, and using different time delay selections.

Nine final models were generated based on the criteria explained in the preceding paragraph. The results of the best model generated in each of the two categories are given in this chapter. **K4Mod37** (*Process Insights* method) is the best model produced in the first category and **Stats2** (PCA method) is the best model produced in the second category.

A summary of the results of the other models is given in Appendix B.

6.2 Model performance

The overall model performance measurements for the two selected models are summarised in this section. The two performance measurements explained in Chapter 4 (paragraph 4.7.6.1) viz.; *Relative error* and R^2 are given in a table for each of the training, testing and validation pattern sets. The results are plotted on two graphs, the first for the *Relative errors* of each set, and the second for

the R^2 value of each set. On the graphs, the relevant performance measurement of each pattern set is plotted against the number of variables in the model being trained.

6.2.1 Model: K4Mod37 (*Process Insights*)

K4Mod37 was the best model produced by the *Process Insights* variable reduction method. This model was built with a dataset that was time merged with the *Linear extend* time merge method (Chapter 4, paragraph 4.7.4.2). The model was built using researcher specified time delays and trained with the sparse data algorithm off. The final model has 38 input variables, reduced from the initial 425 input variables.

K4Mod37						
Train		Test		Validation		Number of variables
Rel err	R sq	Rel err	R sq	Rel err	R sq	
0.167	0.972	0.162	0.974	0.929	0.137	425
0.161	0.974	0.161	0.974	1.500	0.000	346
0.107	0.988	0.104	0.989	0.328	0.892	254
0.103	0.989	0.094	0.991	0.796	0.367	201
0.097	0.991	0.098	0.991	0.853	0.273	172
0.096	0.991	0.104	0.990	0.847	0.283	117
0.110	0.988	0.111	0.989	0.914	0.165	97
0.116	0.987	0.126	0.988	0.582	0.662	75
0.113	0.987	0.122	0.984	0.743	0.448	54
0.118	0.986	0.123	0.985	0.301	0.909	49
0.108	0.988	0.109	0.985	0.110	0.988	38
0.106	0.989	0.107	0.988	0.099	0.990	38
0.105	0.989	0.109	0.988	0.183	0.966	38
0.107	0.989	0.110	0.988	0.126	0.984	38
0.107	0.989	0.107	0.989	0.260	0.932	38
0.114	0.987	0.114	0.987	0.150	0.978	38

Table 6.1: Performance measurements of model K4Mod37.

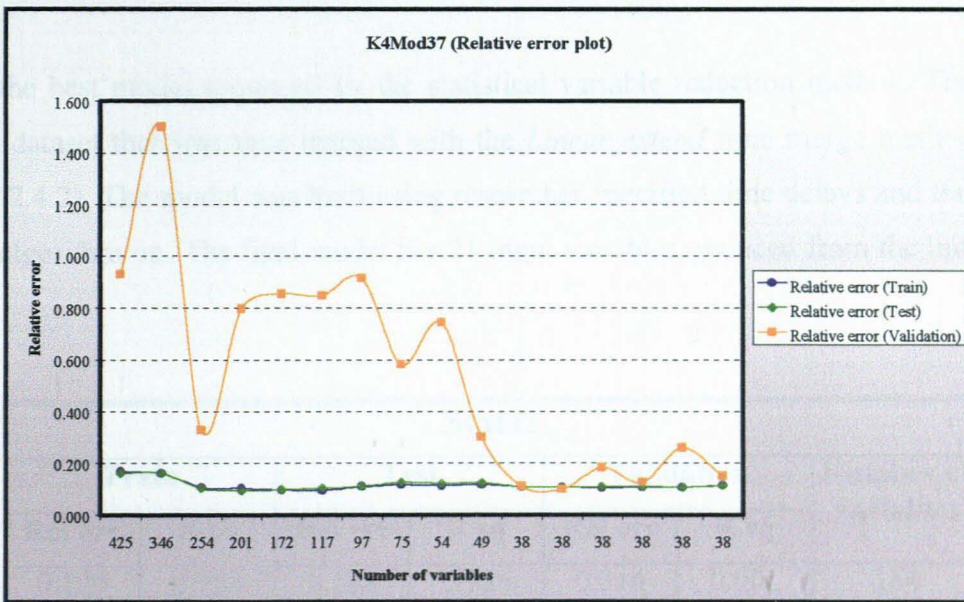


Figure 6.1: Relative error plot for pattern sets of model K4Mod37.

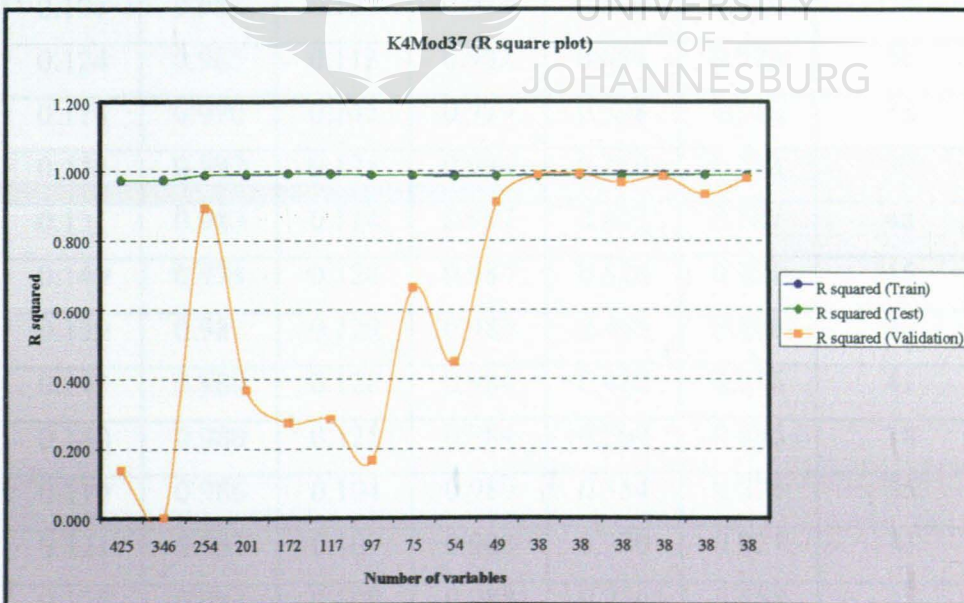


Figure 6.2: R² plot for pattern sets of model K4Mod37.

6.2.2 Model: Stats2 (PCA)

Stats2 was the best model produced by the statistical variable reduction method. This model was built with a dataset that was time merged with the *Linear extend* time merge method (Chapter 4, paragraph 4.7.4.2). The model was built using researcher specified time delays and trained with the sparse data algorithm on. The final model has 21 input variables, reduced from the initial 184 input variables.

Stats2						
Train		Test		Validation		Number of variables
Rel err	R sq	Rel err	R sq	Rel err	R sq	
0.120	0.986	0.118	0.986	0.314	0.901	184
0.108	0.988	0.103	0.989	0.659	0.566	162
0.112	0.988	0.098	0.990	0.633	0.599	148
0.134	0.982	0.112	0.987	0.414	0.829	138
0.142	0.980	0.125	0.984	0.482	0.768	130
0.137	0.981	0.127	0.984	0.596	0.645	106
0.124	0.985	0.113	0.987	0.654	0.573	51
0.174	0.970	0.145	0.979	0.798	0.512	53
0.134	0.982	0.125	0.984	0.719	0.483	50
0.131	0.983	0.114	0.987	0.462	0.787	48
0.149	0.978	0.128	0.984	0.528	0.721	45
0.139	0.981	0.129	0.983	0.447	0.800	42
0.142	0.980	0.128	0.984	0.430	0.815	41
0.140	0.980	0.125	0.984	0.560	0.686	38
0.119	0.986	0.104	0.989	0.354	0.875	35
0.121	0.985	0.109	0.988	0.270	0.927	31
0.125	0.984	0.109	0.988	0.736	0.458	25
0.126	0.984	0.106	0.989	0.721	0.480	24
0.134	0.982	0.125	0.984	0.957	0.085	23
0.141	0.980	0.128	0.984	0.850	0.277	21

Table 6.1: Performance measurements of model Stats2.

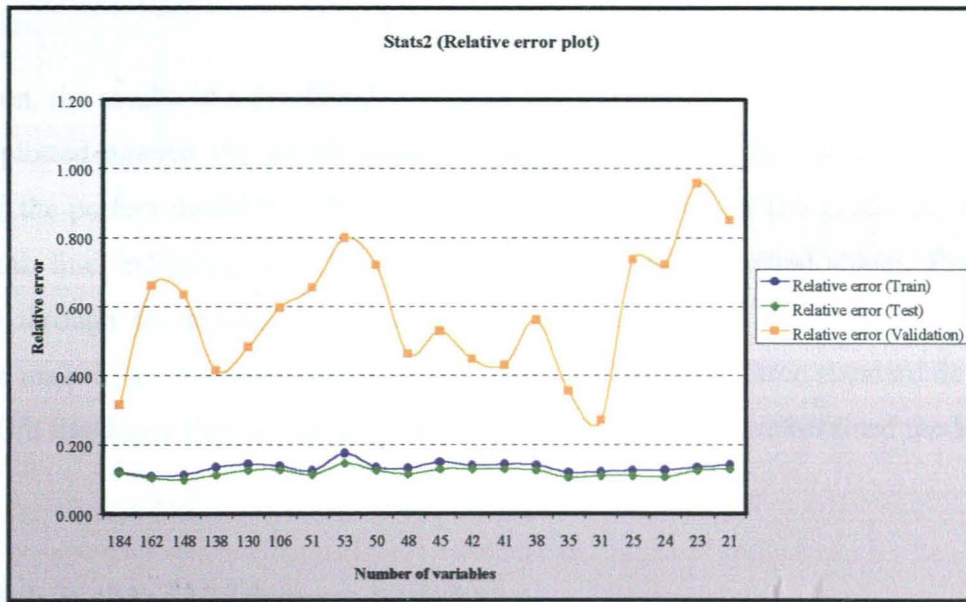


Figure 6.3: Relative error plot for pattern sets of model Stats2.

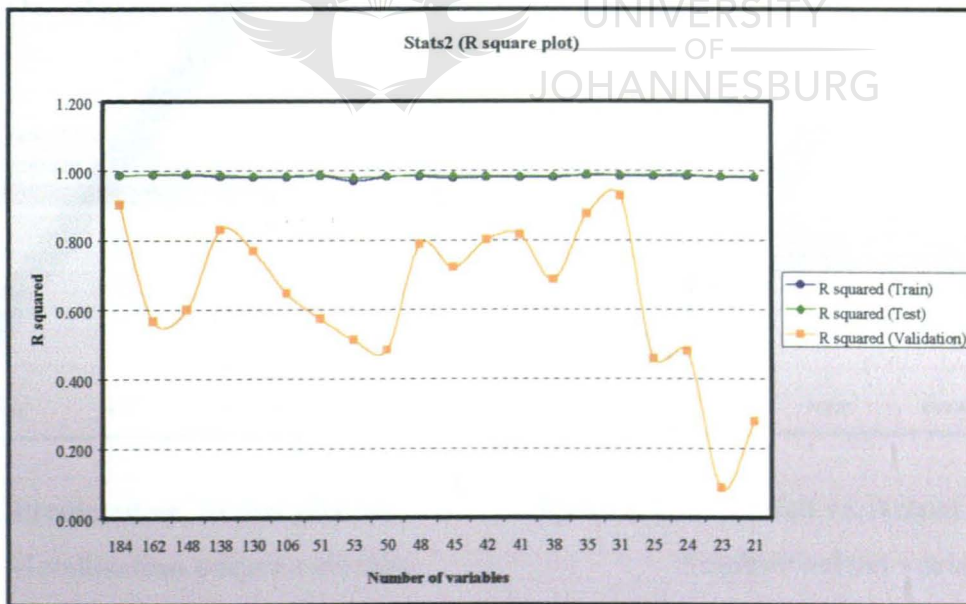


Figure 6.4: R^2 plot for pattern sets of model Stats2.

6.3 Predicted vs. Actual performance

In this section, the results of a *Predicted vs. Actual* analysis are given. The values predicted by the model are plotted against the actual values in the dataset. The plot has a line with slope 1, representing the perfect model line. If a model is 100% accurate, all the predicted points will lie exactly on this line, indicating that all predicted values are equal to actual values. Further, it has a best-fit line through the predicted points, as well as first, second, and third standard deviation bands. If the majority of the predicted vs. actual points lie within the three standard deviation bands and the best-fit line has a slope close to 1, then it is an indication of a well-trained model.

6.3.1 Model: K4Mod37 (*Process Insights*)

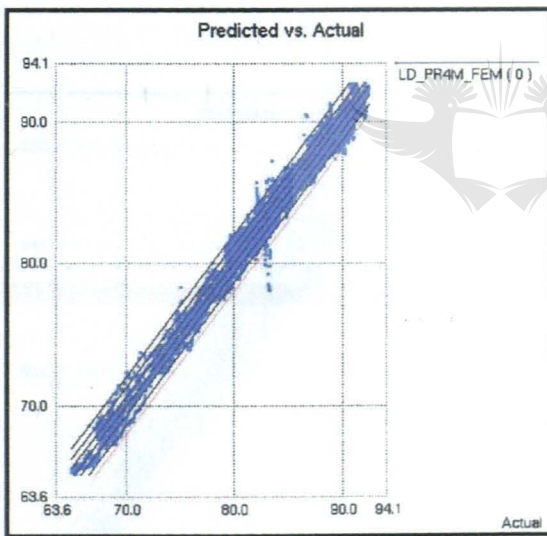


Figure 6.5: Predicted vs. Actual plot for Metallisation output variable. (K4Mod37)

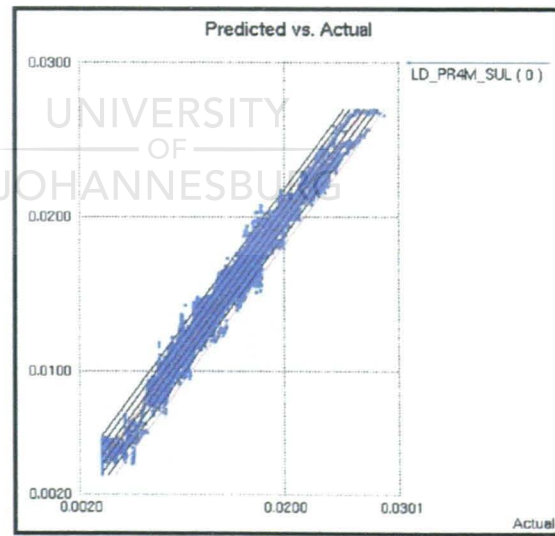


Figure 6.6: Predicted vs. Actual plot for Sulphur output variable. (K4Mod37)

Predicted vs. Actual Analysis: K4Mod37		
Variable	Metallisation	Sulphur
R^2	0.988	0.980
Rel err	0.109	0.142
Slope	0.999	0.984
Std. Dev.	0.672	0.001

Table 6.3: Predicted vs. Actual performance measurements of model K4Mod37.

6.3.2 Model: Stats2 (PCA)

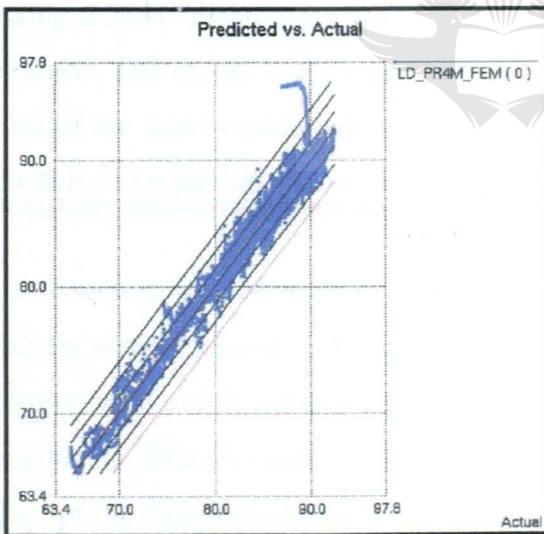


Figure 6.7: Predicted vs. Actual plot for Metallisation output variable. (Stats2)

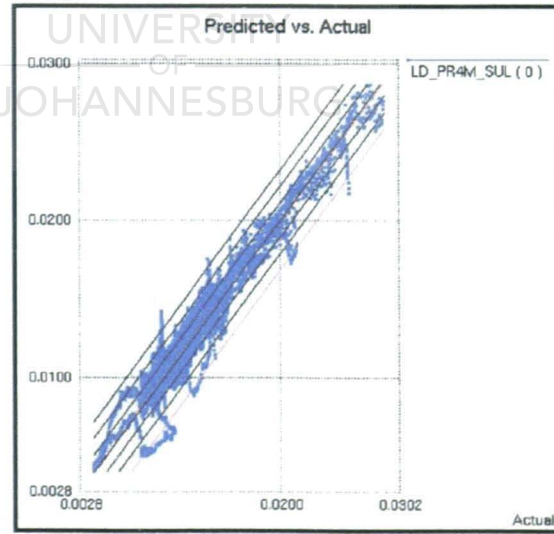


Figure 6.8: Predicted vs. Actual plot for Sulphur output variable. (Stats2)

Predicted vs. Actual Analysis: Stats2		
Variable	Metallisation	Sulphur
R²	0.942	0.911
Rel err	0.241	0.298
Slope	1.007	0.956
Std. Dev.	1.383	0.008

Table 6.4: Predicted vs. Actual performance measurements of model Stats2.

6.4 Most significant variable analysis

In this section the results for the most significant variable analysis are given. A *Sensitivity vs. Rank* analysis was performed on a model to identify the most significant variable (MSV) in the model. During a MSV analysis the most significant variable is removed from the model and the model is retrained. The model is then analysed and the model performance measurements are compared to those of the same model with the most significant variable included. The aim of this analysis is to determine to what extent a model is fault tolerant when one of the inputs to the model is lost.

In this section a MSV analysis was performed on model **K4Mod37** and model **Stats2**. The MSV analysis was performed on both the first and the last trained models of **K4Mod37** and **Stats2**. The *Relative error* and R^2 performance measurements for each of the training, testing and validation sets are given in tables 6.5 and 6.6. The results are plotted on four graphs, one for the first model and the other for the second model for both **K4Mod37** and **Stats2**.

6.4.1 Model: K4Mod37 (*Process Insights*)

Most Significant Variable Analysis: K4Mod37						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
First model	0.167	0.972	0.162	0.974	0.929	0.137
First model with MSV removed	0.165	0.973	0.148	0.978	1.079	0.000
Last model	0.114	0.987	0.114	0.987	0.150	0.978
Last model with MSV removed	0.111	0.988	0.114	0.987	0.283	0.920

Table 6.5: Performance measurements of K4Mod37 models with MSV removed.

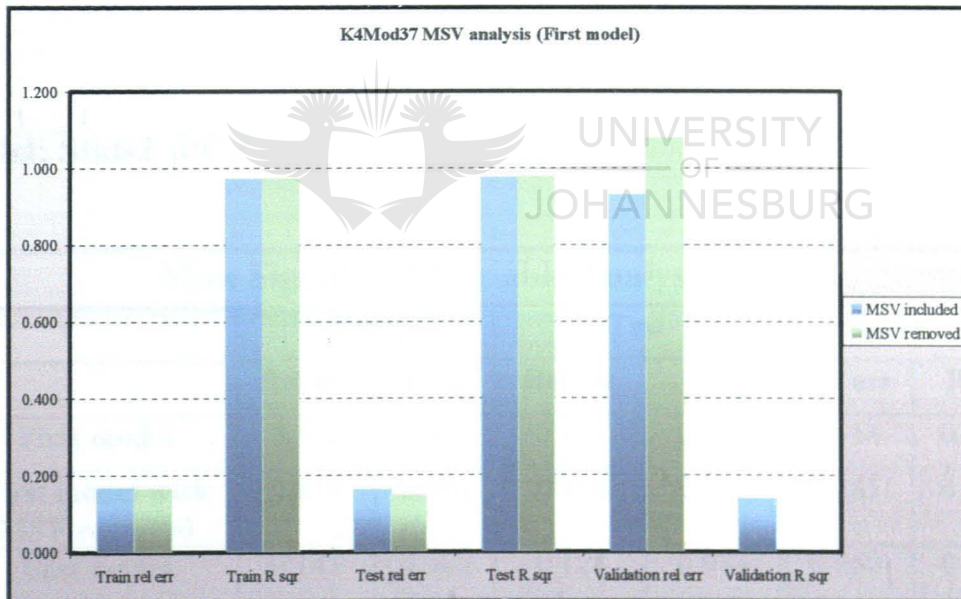


Figure 6.9: K4Mod37 first model MSV analysis performance measurements.

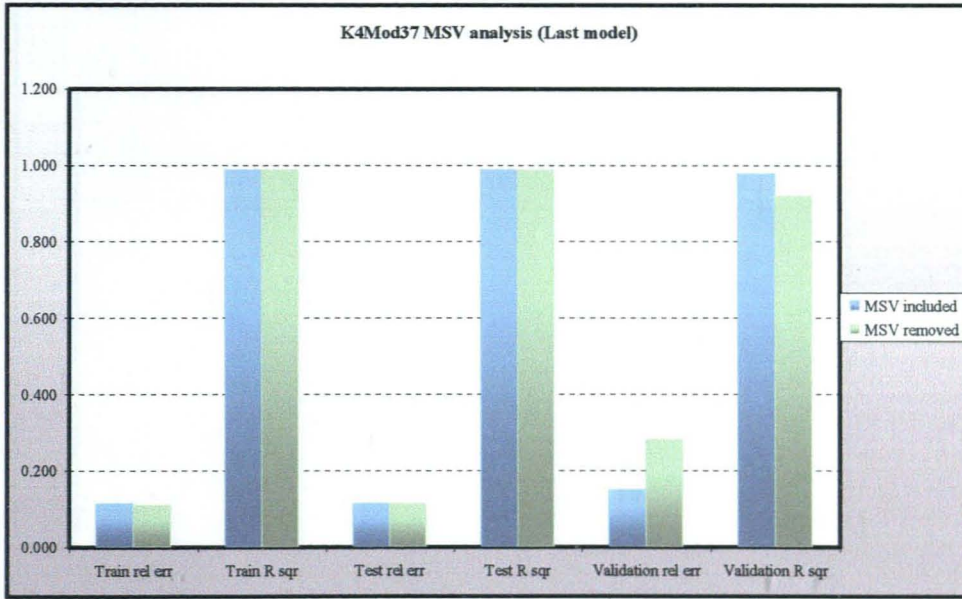


Figure 6.10: K4Mod37 last model MSV analysis performance measurements.

6.4.2 Model: Stats2 (PCA)



UNIVERSITY OF JOHANNESBURG

Most Significant Variable Analysis: Stats2						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
First model	0.120	0.986	0.118	0.986	0.314	0.901
First model with MSV removed	0.116	0.986	0.114	0.987	0.287	0.917
Last model	0.141	0.980	0.128	0.984	0.850	0.277
Last model with MSV removed	0.202	0.959	0.193	0.963	0.443	0.804

Table 6.6: Performance measurements of Stats2 models with MSV removed.

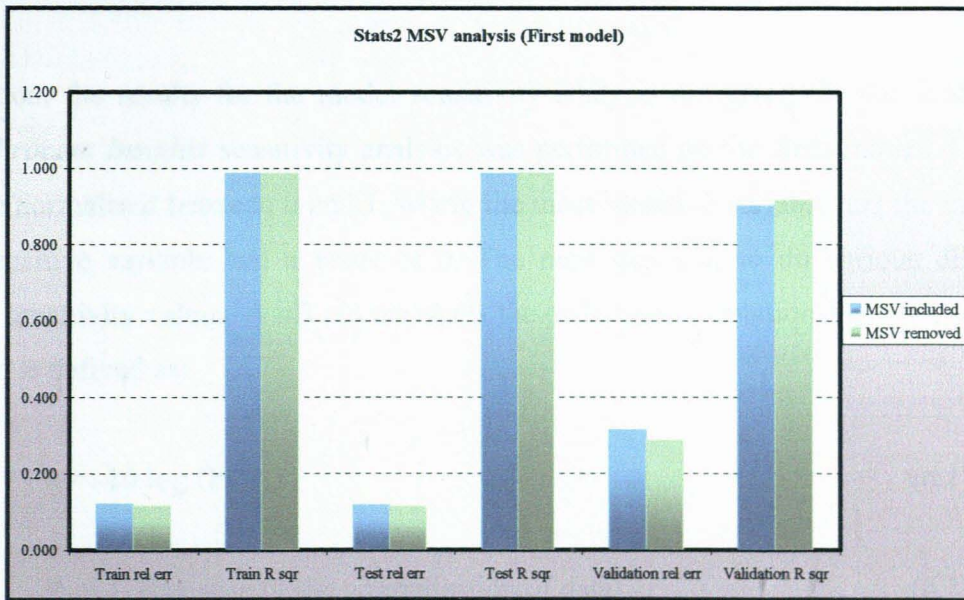


Figure 6.11: Stats2 first model MSV analysis performance measurements.

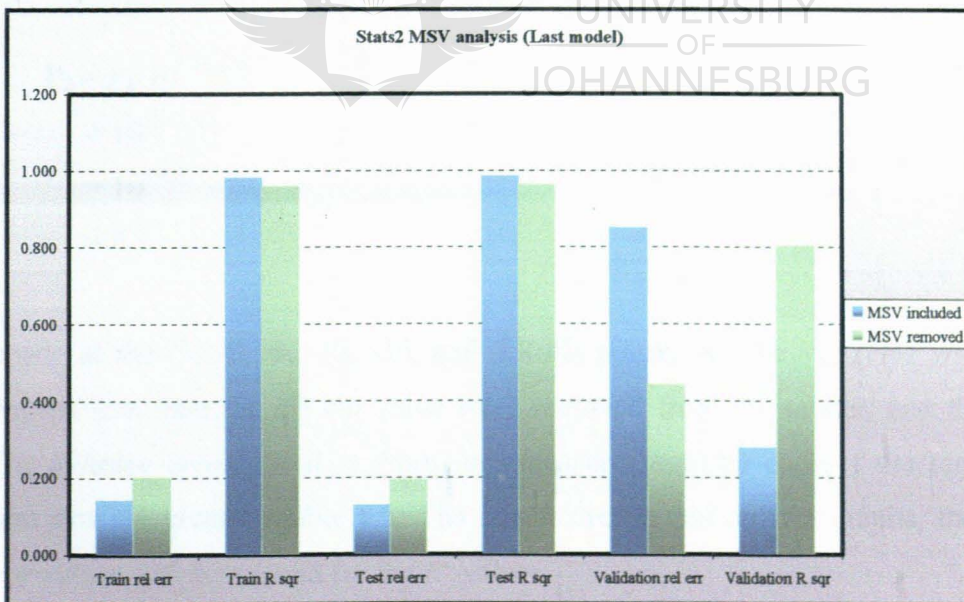


Figure 6.12: Stats2 last model MSV analysis performance measurements.

6.5 Model sensitivity analysis

In this section, the results for the model sensitivity analysis are given. In the model sensitivity analysis a *Process Insights* sensitivity analysis was performed on the final model. The sensitivity values were normalised between 0 and 1, where the most sensitive variable had the value of 1, and the least sensitive variable had a value of 0. The next step was to do various dB cuts on the normalised sensitivity values. A dB cut measures the difference in relationship from one number to another, and is defined as:

$$x = -10 \log (P_1/P_2) \quad (6.1)$$

where

$$P_1 = 1 \text{ (because of the normalisation of data)} \quad (6.2)$$

and

$$P_2 \leq P_1 \quad (6.3)$$

thus if

$$x = -3 \text{ dB} \quad (6.4)$$

then

$$\begin{aligned} P_2 &= P_1/10^{(-x/10)} \\ &= 10^{(x/10)} \\ &= 10^{-0.3} \\ &= 0.5 \end{aligned} \quad (6.5)$$

Cuts were made at the -3, -6, -9, -12, -15, and -18 dB points. All the variables with normalised sensitivity values less than the dB cut value were removed from the model, and the model was retrained. The *Relative error* and R^2 performance measurements for each of the training, testing, and validation sets are given in table 6.3. The results are plotted on two graphs, the first for the *Relative error* values and the second for the R^2 values.

Model Sensitivity Analysis: K4Mod37							
Train		Test		Validation			
Rel err	R sq	Rel err	R sq	Rel err	R sq	dB Number	Number of variables retained
0.105	0.989	0.111	0.988	0.226	0.949	-18	1
0.105	0.989	0.094	0.991	0.29	0.916	-15	2
0.113	0.987	0.108	0.988	0.358	0.872	-12	4
0.106	0.989	0.096	0.991	0.944	0.109	-9	8
0.146	0.979	0.135	0.982	0.586	0.656	-6	17
0.298	0.911	0.29	0.916	0.441	0.806	-3	28

Table 6.7: Model sensitivity analysis performance measurements.

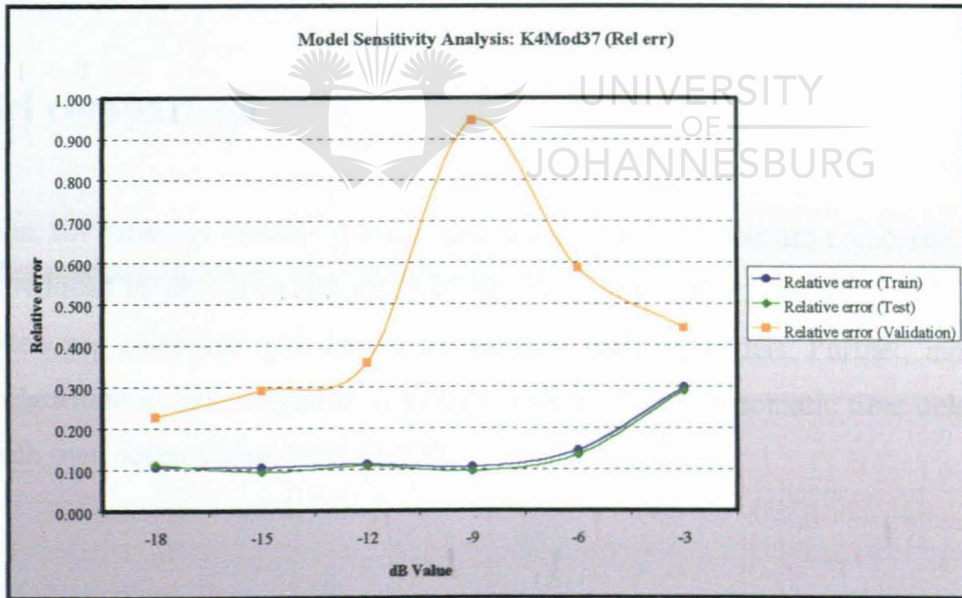


Figure 6.13: Relative error plot for model K4Mod37 sensitivity analysis.

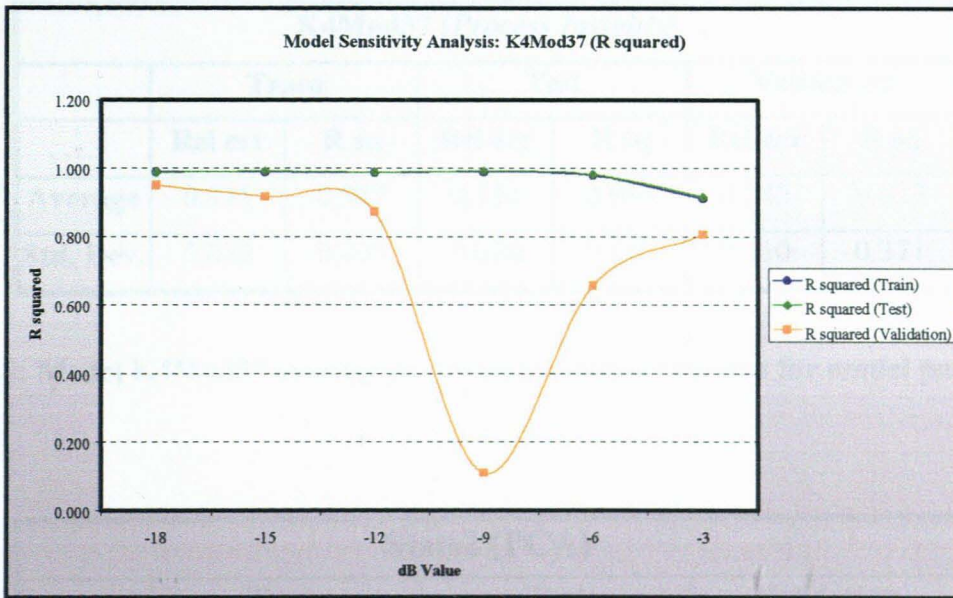


Figure 6.14: R^2 plot for model K4Mod37 sensitivity analysis.

6.6 Model comparison



UNIVERSITY
OF
JOHANNESBURG

In this section, the different modelling techniques used to train models are compared to each other. Statistically reduced models are compared with the *Process Insights* reduced models, *Boxcar* time-merged models are compared with *Linear extend* time-merged models. Further, models with the sparse data algorithm on are compared to models with it off, and automatic time delay models are compared with own defined time delay models.

6.6.1 Statistically reduced models vs. *Process Insights* reduced models

In this section the average R^2 and *Relative error* values for the best statistically reduced model are compared to the average R^2 and *Relative errors* for the best *Process Insights* reduced model. The two models being compared are **K4Mod37** (*Process Insights*) and **Stats2** (Statistical). The results are plotted on two graphs.

K4Mod37 (Process Insights)						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Average	0.115	0.987	0.116	0.986	0.545	0.623
Std. Dev.	0.020	0.005	0.020	0.005	0.410	0.371

Table 6.8: Model K4Mod37 average performance measurements for model pattern sets.

Stats2 (PCA)						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Average	0.133	0.982	0.119	0.986	0.579	0.639
Std. Dev.	0.015	0.004	0.012	0.003	0.185	0.217

Table 6.9: Model Stats2 average performance measurements for model pattern sets.

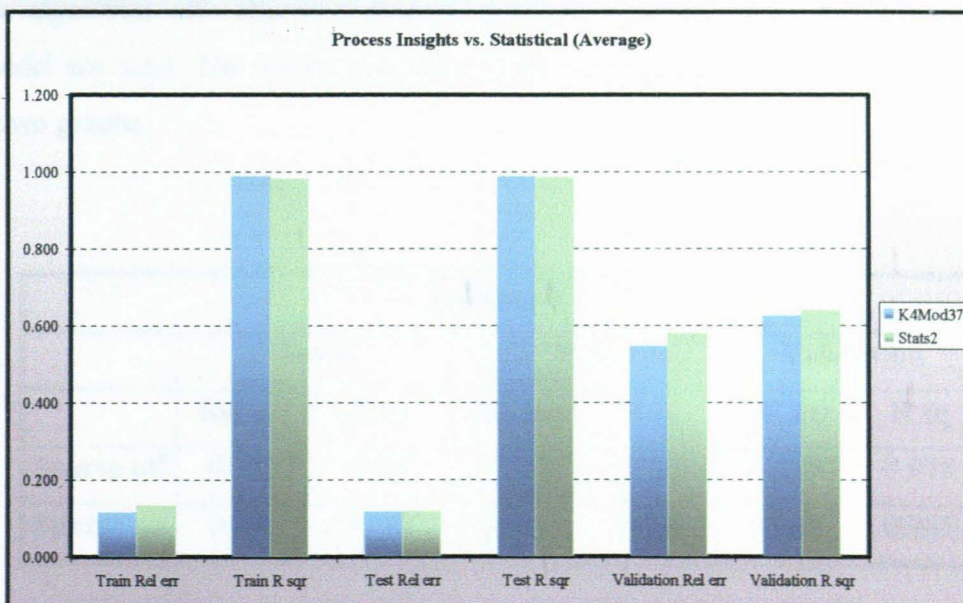


Figure 6.15: Process Insights vs. Statistical comparison (average performance measurements).

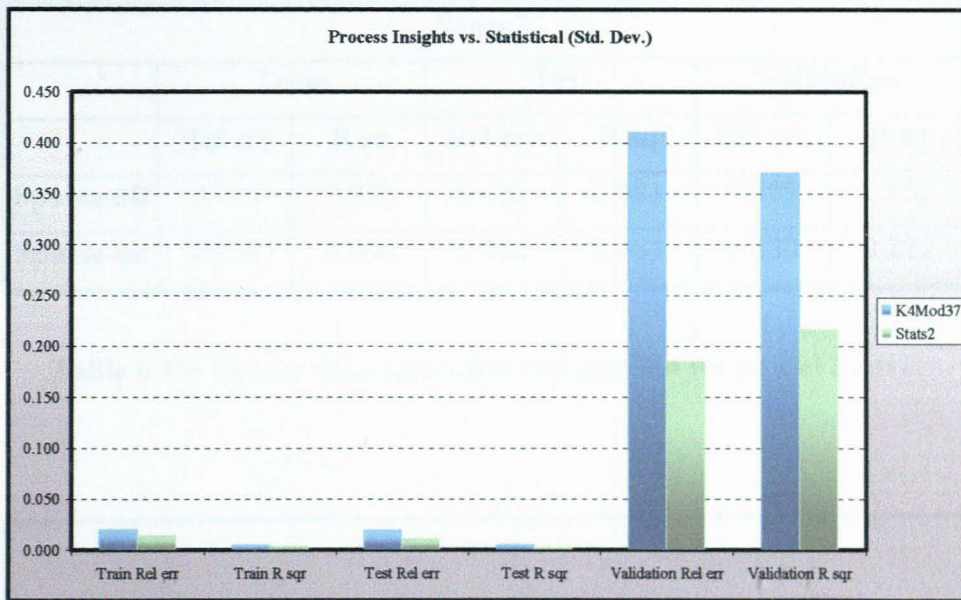


Figure 6.16: *Process Insights* vs. Statistical comparison (standard deviations).

6.6.2 Sparse data algorithm



UNIVERSITY
OF
JOHANNESBURG

Here, models trained with the sparse data algorithm on are compared to models trained with the sparse data algorithm off. The best *Process Insights* reduced model and the best statistically reduced model are used. The results are given in two tables, one for each model. Results are also plotted on two graphs.

K4Mod37						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Sparse off	0.114	0.987	0.114	0.987	0.150	0.978
Sparse on	0.096	0.991	0.095	0.991	0.188	0.965

Table 6.10: Sparse data algorithm comparison for model K4Mod37.

Stats2						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Sparse off	0.149	0.978	0.137	0.981	0.669	0.552
Sparse on	0.141	0.980	0.128	0.984	0.850	0.277

Table 6.11: Sparse data algorithm comparison for model Stats2.

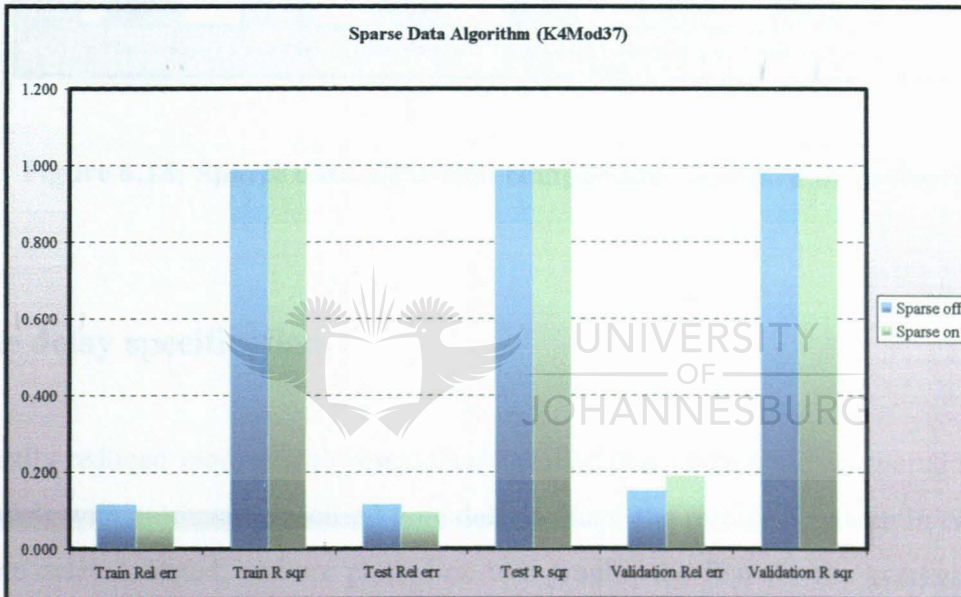


Figure 6.17: Sparse data algorithm comparison (average performance measurements).

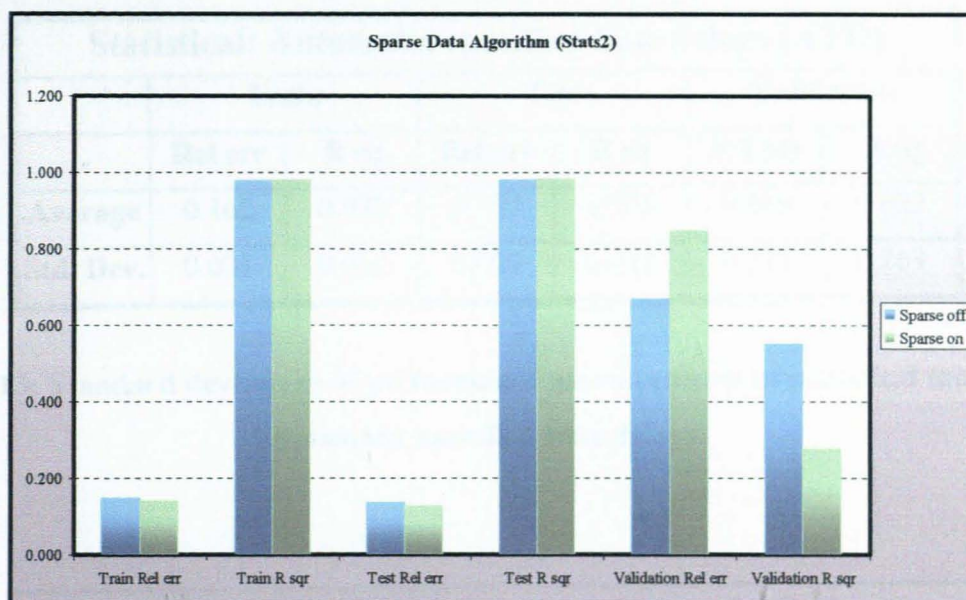


Figure 6.18: Sparse data algorithm comparison (standard deviations).

6.6.3 Time delay specification



UNIVERSITY
OF
JOHANNESBURG

The statistically reduced models with researcher specified time delays are compared to statistically reduced models with automatic specified time delays. Here, the results are given in two tables, one for each time delay selected, and are plotted on two graphs; the first for the average performance measurements and the second for the standard deviations.

Statistical: Researcher specified time delays (RTD)						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Average	0.191	0.960	0.173	0.967	0.568	0.656
Std. Dev.	0.019	0.008	0.011	0.004	0.189	0.227

Table 6.12: Average performance measurements of statistical models with researcher specified time delays.

Statistical: Automatic specified time delays (ATD)						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Average	0.162	0.972	0.157	0.973	0.586	0.622
Std. Dev.	0.031	0.011	0.029	0.011	0.233	0.263

Table 6.13: Standard deviations of performance measurements of statistical models with automatic specified time delays.

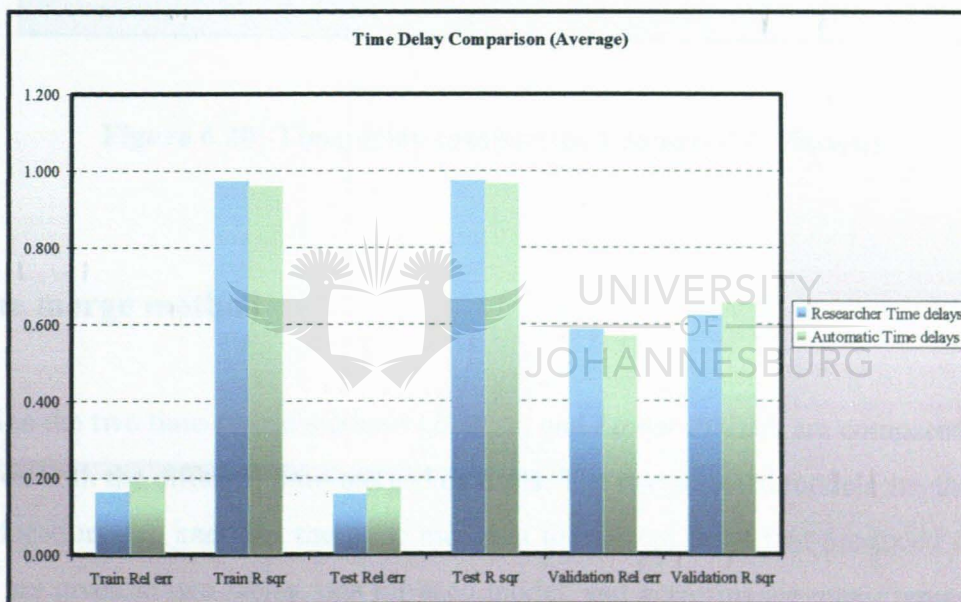


Figure 6.19: Time delay comparison (average performance measurements).

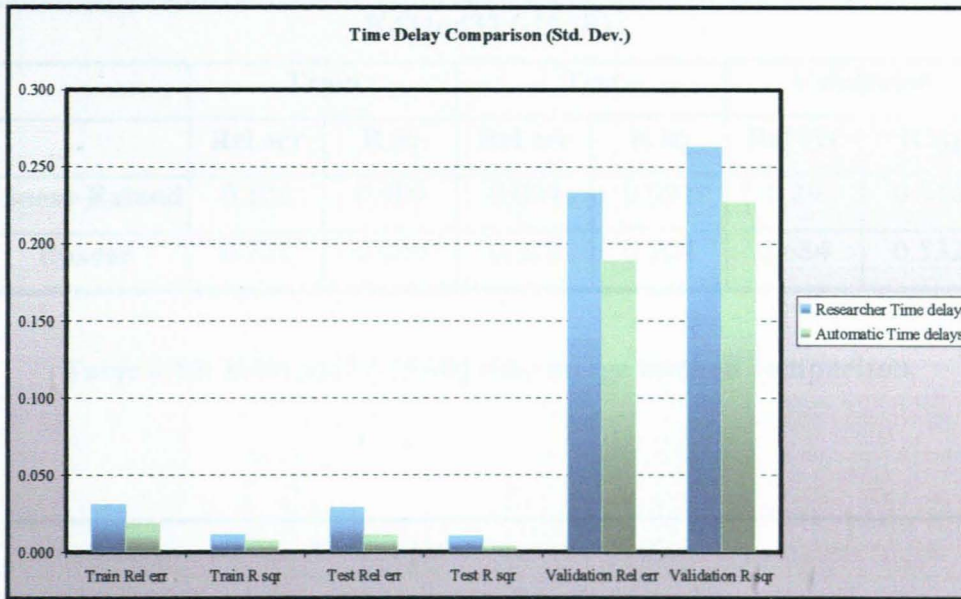


Figure 6.20: Time delay comparison (standard deviations).

6.6.4 Time merge method



UNIVERSITY
OF
JOHANNESBURG

In this section the two time-merge methods (*Boxcar*, and *Linear extend*) are compared. Two models are trained on both the different time-merged datasets. The two selected models are the best *Process Insights* reduced model, and then the same model at the dB cut point that produced the best result. The results are given in two tables, one for each model, and performance measurements are plotted on two graphs.

K4Mod37						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Linear Extend	0.114	0.987	0.114	0.987	0.150	0.978
Boxcar	0.286	0.918	0.313	0.902	0.149	0.978

Table 6.14: K4Mod37 time merge method comparison.

K4Mod37 (-15dB)						
	Train		Test		Validation	
	Rel err	R sq	Rel err	R sq	Rel err	R sq
Linear Extend	0.105	0.989	0.094	0.991	0.29	0.916
Boxcar	0.301	0.909	0.309	0.904	0.684	0.532

Table 6.15: K4Mod37 (-15dB) time merge method comparison.

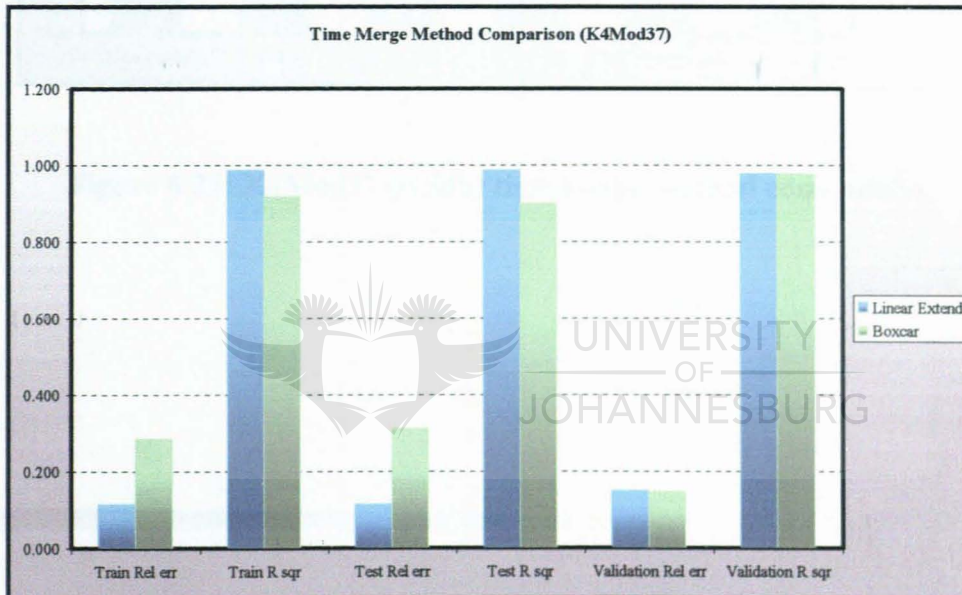


Figure 6.21: K4Mod37 time merge method comparison.

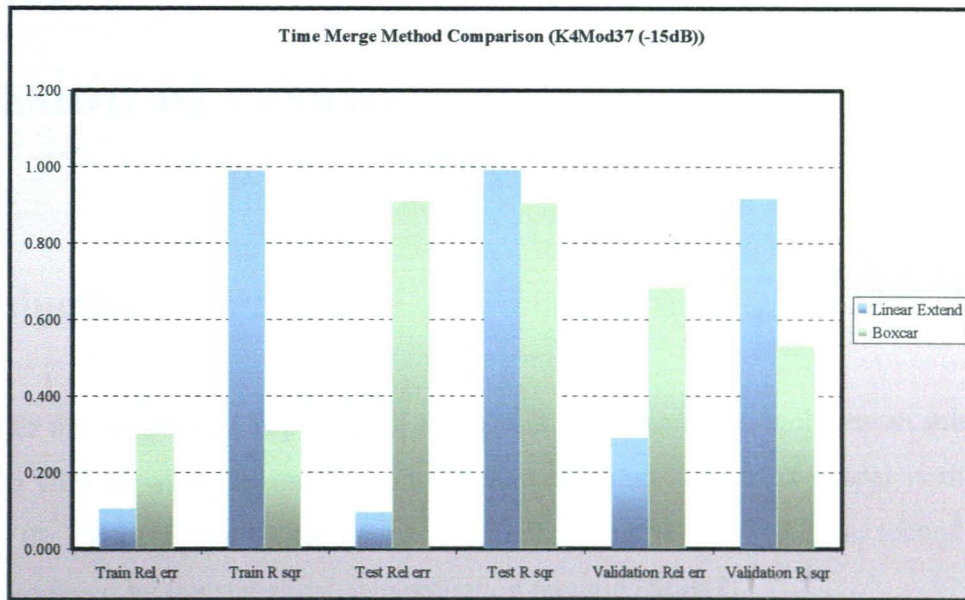


Figure 6.22: K4Mod37 (-15db) time merge method comparison.



Chapter 7

Discussion of results

7.1 Introduction

In this chapter the results of Chapter 6 are discussed. The discussion includes an interpretation of the results as well as what the implications of the results are. The first model is the best model produced by the *Process Insights* data reduction method (**K4Mod37**), and the second model is the best model produced by the statistical data reduction method (**Stats2**).

7.2 Model performance

The first observation that can be made is that both the selected models produced good results in terms of the performance measurements explained in Chapter 4 (paragraph 4.7.6.1). Both models trained to low *Relative errors*, and high R^2 values. The R^2 values produced by both models are close to a value of 1.0, which means that both models are statistically validated. This indicates that both the data reduction methods tested can be used to produce accurate models of the process.

The input variables of the statistically reduced model were reduced to a smaller number than the input variables of the *Process Insights* reduced model. This is due to the fact that with the *Process Insights* data reduction method it is recommended that variables with a sensitivity (*Sensitivity vs. Rank* analysis) of less than 0.1 must be removed from the model. Thus, all the variables with a sensitivity higher than 0.1 are included in the model. With the statistical data reduction method all insignificant variables were removed regardless of their sensitivities. This led to more variables being removed by the statistical method than by the *Process Insights* method. The fact that the statistical method produced an adequate model with fewer input variables appears to indicate that a sensitivity cut-off point higher than 0.1 (in *Process Insights*) can be used to reduce the model to a smaller number of input variables without degrading the model performance.

The performance measurements of the validation sets varied significantly over the training iterations. For model **K4Mod37** this could be because the validation set was moved around in the dataset to see the effect thereof. The results show that the best performance was obtained when the last 5% of the dataset were used as the validation set. The performance of the **Stats2** model varied even though the validation set was kept as the last 5% of the dataset. This could be due to the fact that variables were removed that helped the model to learn certain process dynamics. Another reason could be that some of the validation sets contain certain process dynamics that are not contained in the rest of the data and are thus not well learned by the model.

From the results one can see that the reduction of variables does not have a significant effect on the performance of the models. The performance measurements varied slightly after every reduction iteration. Some models performed worse while others performed better. On the whole, one can say that the reduction process has little effect on model performance. Thus, models can be built where the number of input variables has been minimised dramatically without significantly degrading the model's performance as a controller element.

7.3 Predicted vs. Actual



The *Predicted vs. Actual* results for the two selected models show that the two models perform well in terms of predicting the process. The performance measurements for the two most important output variables viz.; degree of metallisation and sulphur content are good for both models. The slopes of the best-fit lines through the predicted points are close to 1 in all the cases. The standard deviation bands of the two models are relatively small but do spread in the case of the statistically reduced models. Model **K4Mod37** has smaller standard deviation bands for the output variables than model **Stats2**. Based on this, one can say that model **K4Mod37** models the process better than model **Stats2**.

Another characteristic of the *Predicted vs. Actual* analysis is that very few of the predicted points lie outside the three standard deviation belts. This is an indication that the majority of points are predicted with good accuracy. The points lying outside the three standard deviation belts are patterns not well learned by the model. These inaccuracies are probably as a result of one, or a combination, of the following possibilities. The first possibility is that the models could not learn

important process dynamics because of poor representation of those dynamics in the data. The second is that the data leading to the inaccuracies comes from measurement errors or other process effects. In the first case, the remedy is to expand the learning data, this should ensure that the model can learn to predict the effects of the process dynamics. In the second case the data can be removed from the dataset to ensure that the model would learn only important process dynamics.

The *Predicted vs. Actual* graphs for model **Stats2** show interesting patterns of points falling outside the three standard deviation belts. For the metallisation output variable they lie at the top of the graph and for the sulphur output variable they lie at the bottom of the graph. These points are typically the patterns in the validation set that were not well learned and could be related to measurement problems.

7.4 Most significant variable analysis

The results of the MSV analysis show that the models are not adversely affected when the most significant variable is removed from the model and the model is retrained. The results show that the performance measurements of the models worsened slightly in some cases and stayed the same in others.

This indicates that the models do not depend on the most significant variable to remain adequate. Thus, if something happens to the MSV the model retains its integrity in terms of predicting the process. This shows that models will still be able to produce acceptable results when the MSV is lost. This bodes well for the fault tolerance of the model in terms of individual sensor failures. Thus, if the MSV is not overwhelmingly important for model performance then model control should not be affected by single sensor failures.

7.5 Model sensitivity analysis

The results of section 6.5 show that the -18, -15, -12, and -9 dB cuts had almost no effect on the performance of the model. The model performance started getting noticeably worse at the -6 and -3 dB cuts. Although the performance worsened at the -6 and -3 dB cuts, the models still performed

well enough for them to be regarded as adequate for process predictions. Such dramatic reductions in variables could form the basis of a real-time trainable model, which forms the model component of a model based non-linear classical controller design.

The fact that the performance does not seriously deteriorate is due to the fact that very few input variables lie below the -18, -15, -12, and -9 dB cut lines. Although there are more input variables below the -6 and -3 dB cut lines, there are still enough to ensure adequate model performance. Thus, after each dB cut, a good proportion of the input variables remain in the model, which ensures good model performance.

The fact that very few of the already trimmed variables lie below the dB cut lines, especially at -18, -15, -12, and -9 dB, show that a large number of the input variables have high sensitivities close to each other as a result of the *Process Insights* data reduction. Again, this indicates that a model with fewer input variables can be built without losing much in model performance. However, there is an optimal point for the number of input variables before the model performance starts getting significantly worse. In this case it is close to the -3 dB cut-off point where only ten input variables remained.



7.6 Model comparison

7.6.1 Statistically reduced models vs. *Process Insights* reduced models

The results show that the average performance measurements of the two models produced by the different reduction methods are very similar. From this, one can draw the conclusion that there is no significant difference between the performance of the models produced by either of the two reduction methods.

The only notable difference is on the standard deviation bands of the validation sets of the two models where the *Process Insights* model had a larger standard deviation band than the statistical model. This shows that the PCA reduction method can be used successfully in reducing the number of variables without losing much, if anything, in terms of model performance.

7.6.2 Sparse data algorithm

The results show that models trained with the sparse data algorithm on, produced slightly better results than those trained with the algorithm off. However, the difference in performance is very small.

The researcher would recommend that the sparse data algorithm be used when large time gaps are present between lab analyses (thus a large number of data points are interpolated to fill the four-hour gaps). The algorithm assigns certainties to actual points, and points close to actual points. When models are trained more attention is paid to points with high certainties, which makes the models better.

The implication of this is that models trained with the sparse data algorithm can tolerate poorly interpolated data because of the low, or zero certainties associated with these points. This ensures that the produced model performs well in bridging large gaps in training data.

7.6.3 Time delay specification



The results show that models with automatic calculated time delays produced slightly better results than models with researcher specified time delays. However, the performances produced by both methods are adequate and acceptable.

This shows that the *Process Insights* automatic time delay finder can be used with confidence without the fear of bad model performance because of incorrectly calculated time delays between input and output variables.

7.6.4 Time merge method

The results show that the *Linear extend* time merge method outperformed the *Boxcar* time merge method. The models produced by the *Linear extend* method are better than the models produced by the *Boxcar* method because of the method used to fill in the missing data. The *Linear extend*

method makes use of interpolation whereas the *Boxcar* method repeats the most recent actual value encountered.

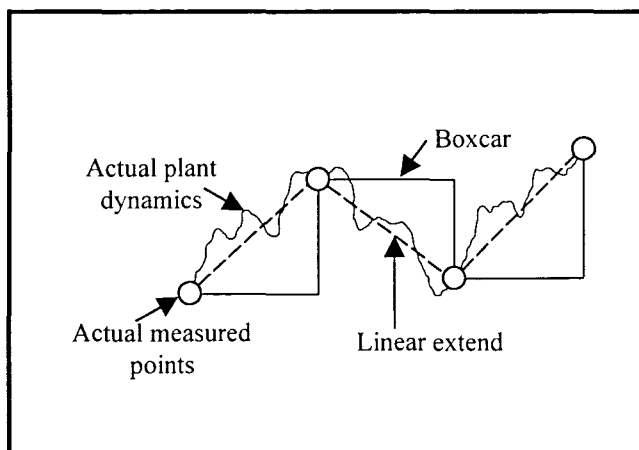


Figure 7.1: Difference between *Boxcar* and *Linear extend* time merge methods.

The difference in performance between the *Boxcar* and the *Linear extend* time merge methods can be explained in terms of the transfer phenomenon. With the *Boxcar* method the most recent actual measured value is repeated until a new actual measured value is encountered as can be seen in figure 7.1. Thus, the transfer phenomenon is a dramatic jump from one value to another. In this case the difference between the *Boxcar* calculated points and the actual plant dynamics increases, which leads to an increase in the error between predicted and actual points.

The *Linear extend* method interpolates between two actual measured points. Thus, the transfer phenomenon between two points is much smoother than the dramatic jump produced by the *Boxcar* method. The difference between the interpolated points and the actual plant dynamics are smaller than that of the *Boxcar*. This leads to a smaller error between the predicted and actual points, which accounts for the *Linear extend* method performing better than the *Boxcar* method, at the expense of slightly greater complexity.

Chapter 8

Conclusion

Once all the aspects of this study are taken into consideration, two questions must be answered: what was learned from this study and what possible future work stems from the work done in this study?

The answer to the first question can be summarised as follows:

- The first conclusion that can be made is that a SL/RN direct reduction process can successfully be modelled with neural networks. This is evident from the results, which showed that adequate models were built and that the performance of these models was good. This supports **Research hypothesis 1** for the case where the number of input variables is optimised.
- The number of variables in a model can be reduced significantly while maintaining good model performance. However, there is an optimal point for the number of variables before model performance starts to deteriorate. This was evident from the results of the model sensitivity analysis. This supports **Research hypothesis 1**.
- The most significant variable has little effect on model performance, which indicates that model control would not be significantly affected by single sensor failure. The model sensitivity analysis showed that a finite, but not very small, number of variables are significant to the model, which is why the removal of the most significant variable can be tolerated by a model. This supports **Research hypothesis 3**, because if the failure of the most significant variable can be tolerated the failure of any single input variable can be tolerated.



- A principal component analysis can be used successfully to reduce the dimensionality of a model. The study has shown that a PCA reduced method can produce adequate models that can compete with *Process Insights* models. This supports **Research hypothesis 2**.

The possible work that can stem from this research can be summarised as follow:

- The models created in this study can be implemented on the direct reduction plant. The models can be connected to the current control structure discussed in Chapter 2 (paragraph 2.7) to deliver parallel output of the predicted variables. The predicted values can then be compared to actual values to evaluate model performance. Models can be refined and results can be compared to the results of this study.
- Another useful study would be to test *Process Insights* on a different metallurgical process. Thus the same methodology can be followed but on a different process such as the blast furnace process, the electric arc furnace process or the basic oxygen furnace process. Because these processes can be even more complex it may only be viable to model some aspect of the process such as slag formation, temperature profile or the consumption rate of some expensive raw material.
- The current prediction models can be converted into control models. These control models can be connected to the control structure of the plant discussed in Chapter 2 (paragraph 2.7). The models can then be converted into real-time advanced process controllers with little or no human intervention.

Addendum



Appendix A

Model inputs and outputs

INPUTS:			
Carbon to iron ratio	(CI_RATIO)	ABC temp	(C4_15_07_T10_PV)
Ore feed rate	(B4_04_10_F10_P)	ABC temp	(C4_15_07_T11_PV)
Coal feed rate	(B4_04_10_F11_R)	ABC temp	(C4_15_07_T12_PV)
Dolomite feed rate	(B4_04_10_F12_R)	Boiler steam	(C4_16_01_F07_PV)
Material mix	(B4_04_10_F13_R)	Boiler pressure	(C4_16_01_P33_VAR)
Coarse inj feed	(B4_04_10_F20_R)	Boiler pressure	(C4_16_01_P34_VAR)
Fine inj feed	(B4_04_10_F24_R)	Boiler inlet temp	(C4_16_01_T13_PV)
Coal inj air	(B4_04_25_F07_PV)	Kiln main drive	(D4_10_05_J01_VAR)
Combustion air CB	(B4_04_28_F14_VAR)	Kiln Speed	(D4_10_05_S01_PV)
Coke gas to CB	(B4_04_29_F09_VAR)	Cooler speed	(D4_12_04_S02_PV)
Plant air	(B4_04_25_F06_PV)	Dolomite screen (6)	(LD_DOO_*)
ABC inj water	(C4_15_07_F01_VAR)	Duiker screen (7)	(LD_DUI_*)
ABC comb air	(C4_15_24_F02_PV)	Kenbar screen (7)	(LD_KNF_*)
ABC comb air	(C4_15_24_F03_PV)	Ore screen (7)	(LD_OLU_*)
Boiler feed water	(C4_16_01_F05_PV)	Duiker chemical (5)	(LD_DUPI_*)
Shell fan 1	(D4_10_03_F01_VAR)	Kenbar chemical (5)	(LD_KENF_*)
Shell fan 2	(D4_10_03_F02_VAR)	Ore chemical (3)	(LD_OLUM_*)
Shell fan 3	(D4_10_03_F03_VAR)	Ore chemical (3)	(LD_OLUM_*)
Shell fan 4	(D4_10_03_F04_VAR)	Kiln temp	(D4_10_03_T01_VAR)
Shell fan 5	(D4_10_03_F05_VAR)	Kiln temp	(D4_10_03_T02_VAR)
Shell fan 6	(D4_10_03_F06_VAR)	Kiln temp	(D4_10_03_T03_VAR)
Shell fan 7	(D4_10_03_F07_VAR)	Kiln temp	(D4_10_03_T04_VAR)
Shell fan 8	(D4_10_03_F08_VAR)	Kiln temp	(D4_10_03_T05_VAR)
Kiln inlet pressure	(C4_15_01_P01_P01)	Kiln temp	(D4_10_03_T06_VAR)
Kiln outlet pressure	(C4_15_01_P01_P02)	Kiln temp	(D4_10_03_T07_VAR)

ABC inlet temp	(C4_15_01_T02_PV)	Kiln temp	(D4_10_03_T08_VAR)
ABC chute temp	(C4_15_01_T03_VAR)	Kiln temp	(D4_10_03_T09_VAR)
DSC temp1	(C4_15_01_T04_VAR)	Kiln temp	(D4_10_03_T10_VAR)
DSC temp2	(C4_15_01_T05_VAR)	Transfer chute temp	(D4_12_01_T23_VAR)
ABC outlet temp	(C4_15_07_T06_PV)	Kiln Product	(E4_20_03_F03_P)
ABC temp	(C4_15_07_T07_PV)	Cooler prod temp	(D4_12_02_ET01_PV)
ABC temp	(C4_15_07_T08_PV)	Product screen (7)	(LD_PM4_*)
ABC temp	(C4_15_07_T09_PV)		
OUTPUTS:			
Product chemical (3)	(LD_PR4M_*)		

Table A1: Model input and output variables.



UNIVERSITY
OF
JOHANNESBURG

Appendix B

Summary of model results

B1: K4Mod35 (*Process Insights*)

Model K4Mod35 was built with a dataset that was time merged with the *Boxcar* time merge method. The model was built using researcher specified time delays and trained with the sparse data algorithm off. The final model has 38 input variables, reduced from the initial 425 input variables.

K4Mod35						
Train		Test		Validation		Number of variables
Rel err	Rsqr	Rel err	Rsqr	Rel err	Rsqr	
0.230	0.947	0.252	0.937	0.380	0.856	425
0.250	0.937	0.272	0.920	0.571	0.571	360
0.183	0.967	0.237	0.944	0.304	0.304	320
0.167	0.972	0.156	0.976	0.273	0.926	266
0.166	0.972	0.152	0.977	0.273	0.925	216
0.150	0.978	0.199	0.960	0.819	0.329	169
0.179	0.968	0.204	0.958	1.500	0.000	125
0.207	0.957	0.229	0.948	0.581	0.663	98
0.189	0.964	0.207	0.957	1.192	0.000	71
0.214	0.954	0.222	0.951	0.157	0.975	66
0.182	0.967	0.189	0.964	0.315	0.901	50
0.184	0.966	0.172	0.970	0.194	0.963	46
0.164	0.973	0.153	0.977	0.210	0.956	38
0.164	0.973	0.155	0.976	0.240	0.943	38
0.171	0.971	0.155	0.976	0.082	0.993	38

Table B1: Performance measurements of model K4Mod35.

B2: K4Mod36 (Process Insights)

Model K4Mod36 was built with a dataset that was time merged with the *Boxcar* time merge method. The model was built using researcher specified time delays and trained with the sparse data algorithm on. The final model has 35 input variables, reduced from the initial 425 input variables.

K4Mod36						
Train		Test		Validation		Number of variables
Rel err	Rsqr	Rel err	Rsqr	Rel err	Rsqr	
0.192	0.963	0.208	0.957	0.504	0.746	425
0.195	0.962	0.210	0.956	0.844	0.287	364
0.169	0.971	0.196	0.961	0.210	0.956	326
0.150	0.978	0.147	0.978	0.142	0.980	271
0.150	0.977	0.199	0.961	0.463	0.786	210
0.130	0.983	0.178	0.968	0.648	0.580	172
0.155	0.976	0.202	0.959	1.500	0.000	127
0.176	0.969	0.216	0.953	0.519	0.731	96
0.173	0.970	0.218	0.953	0.705	0.503	71
0.203	0.959	0.227	0.948	0.506	0.744	54
0.217	0.953	0.214	0.954	0.390	0.848	43
0.207	0.957	0.205	0.958	0.476	0.774	40
0.200	0.960	0.192	0.963	0.703	0.506	35
0.190	0.964	0.189	0.964	0.540	0.708	35

Table B2: Performance measurements of model K4Mod36.

B3: K4Mod38 (Process Insights)

Model K4Mod38 was built with a dataset that was time merged with the *Linear extend* time merge method. The model was built using researcher specified time delays and trained with the sparse data algorithm on. The final model has 33 input variables, reduced from the initial 425 input variables.

K4Mod38						
Train		Test		Validation		Number of variables
Rel err	Rsqr	Rel err	Rsqr	Rel err	Rsqr	
0.132	0.983	0.134	0.982	1.000	0.000	425
0.119	0.986	0.124	0.985	1.500	0.000	341
0.091	0.992	0.091	0.992	0.554	0.693	303
0.081	0.993	0.069	0.995	0.216	0.954	256
0.082	0.992	0.073	0.995	0.745	0.445	206
0.081	0.993	0.074	0.995	0.945	0.108	166
0.097	0.991	0.087	0.993	1.164	0.000	115
0.104	0.989	0.094	0.991	0.688	0.527	97
0.104	0.989	0.101	0.990	0.727	0.472	66
0.135	0.982	0.121	0.985	0.164	0.973	45
0.141	0.980	0.132	0.983	0.308	0.905	42
0.105	0.989	0.101	0.990	0.271	0.927	33
0.109	0.988	0.103	0.989	0.379	0.856	33
0.106	0.989	0.099	0.990	0.200	0.960	33
0.100	0.990	0.094	0.991	0.243	0.941	33
0.103	0.989	0.095	0.991	1.007	0.000	33

Table B3: Performance measurements of model K4Mod38.

B4: K4Mod39 (Process Insights)

Model K4Mod39 was built with a dataset that was time merged with the *Linear extend* time merge method. The model was built using no time delays and trained with the sparse data algorithm on. The final model has 39 input variables, reduced from the initial 96 input variables.

K4Mod39						
Train		Test		Validation		Number of variables
Rel err	Rsqr	Rel err	Rsqr	Rel err	Rsqr	
0.338	0.850	0.358	0.872	0.957	0.084	96
0.403	0.838	0.770	0.858	0.358	0.872	81
0.352	0.876	0.323	0.896	0.257	0.934	70
0.336	0.887	0.339	0.885	0.334	0.888	59

Table B4: Performance measurements of model K4Mod39.



B5: Stats1 (PCA)

Model Stats1 was built with a dataset that was time merged with the *Boxcar* time merge method. The model was built using researcher specified time delays and trained with the sparse data algorithm off. The final model has 19 input variables, reduced from the initial 184 input variables.

Stats1						
Train		Test		Validation		Number of variables
Rel err	Rsq	Rel err	Rsq	Rel err	Rsq	
0.209	0.956	0.224	0.950	0.836	0.302	184
0.247	0.939	0.213	0.954	0.294	0.914	162
0.234	0.945	0.218	0.952	0.342	0.883	144
0.270	0.927	0.229	0.947	0.486	0.764	134
0.267	0.929	0.231	0.947	0.387	0.850	128
0.266	0.929	0.230	0.947	0.500	0.750	116
0.299	0.910	0.247	0.939	0.418	0.825	56
0.247	0.939	0.221	0.951	0.398	0.842	54
0.228	0.948	0.223	0.950	0.432	0.813	51
0.238	0.943	0.215	0.954	0.825	0.320	49
0.227	0.949	0.233	0.946	0.407	0.834	47
0.236	0.944	0.224	0.950	0.624	0.610	44
0.236	0.944	0.230	0.947	0.974	0.052	38
0.237	0.944	0.218	0.952	0.510	0.740	33
0.231	0.946	0.223	0.950	0.498	0.752	28
0.265	0.930	0.234	0.945	0.744	0.775	23
0.281	0.921	0.251	0.937	0.702	0.508	20
0.265	0.930	0.245	0.940	0.641	0.590	19

Table B5: Performance measurements of model Stats1.

B6: Stats3 (PCA)

Model Stats3 was built with a dataset that was time merged with the *Boxcar* time merge method. The model was built using automatically calculated time delays and trained with the sparse data algorithm off. The final model has 19 input variables, reduced from the initial 254 input variables.

Stats3						
Train		Test		Validation		Number of variables
Rel err	Rsqr	Rel err	Rsqr	Rel err	Rsqr	
0.108	0.988	0.177	0.969	0.281	0.921	254
0.134	0.982	0.182	0.967	0.297	0.912	216
0.151	0.977	0.160	0.974	0.523	0.726	192
0.184	0.966	0.197	0.961	0.805	0.351	177
0.176	0.969	0.188	0.965	0.724	0.477	169
0.183	0.966	0.183	0.967	0.648	0.580	154
0.188	0.965	0.182	0.967	0.602	0.637	57
0.178	0.968	0.179	0.968	0.548	0.700	54
0.176	0.969	0.132	0.982	0.609	0.630	51
0.190	0.964	0.156	0.976	0.640	0.591	49
0.188	0.965	0.160	0.975	0.816	0.334	47
0.194	0.962	0.173	0.970	0.679	0.538	44
0.184	0.966	0.140	0.980	0.813	0.338	38
0.205	0.958	0.191	0.964	0.537	0.712	33
0.233	0.946	0.209	0.956	0.526	0.723	28
0.247	0.939	0.229	0.947	0.852	0.275	23
0.272	0.926	0.263	0.931	1.252	0.000	20
0.280	0.922	0.287	0.918	0.584	0.659	19

Table B6: Performance measurements of model Stats3.

B7: Stats4 (PCA)

Model Stats4 was built with a dataset that was time merged with the *Linear extend* time merge method. The model was built using automatically calculated time delays and trained with the sparse data algorithm on. The final model has 21 input variables, reduced from the initial 263 input variables.

Stats4						
Train		Test		Validation		Number of variables
Rel err	Rsqr	Rel err	Rsqr	Rel err	Rsqr	
0.100	0.990	0.091	0.992	0.590	0.652	263
0.098	0.990	0.090	0.992	0.521	0.728	222
0.110	0.988	0.098	0.990	0.653	0.573	201
0.112	0.987	0.108	0.988	0.514	0.736	188
0.127	0.984	0.116	0.987	0.509	0.741	176
0.136	0.982	0.130	0.983	0.387	0.851	149
0.122	0.985	0.119	0.986	0.414	0.829	53
0.149	0.978	0.141	0.980	0.283	0.920	51
0.119	0.986	0.116	0.986	0.461	0.787	50
0.132	0.983	0.128	0.984	0.580	0.664	48
0.137	0.981	0.130	0.983	0.366	0.866	45
0.120	0.986	0.121	0.985	0.334	0.889	42
0.143	0.980	0.134	0.982	0.674	0.546	41
0.135	0.982	0.129	0.983	0.982	0.035	38
0.120	0.986	0.129	0.983	0.243	0.941	35
0.150	0.977	0.140	0.980	0.337	0.886	31
0.146	0.979	0.139	0.981	0.262	0.931	25
0.141	0.980	0.139	0.981	1.111	0.000	24
0.15	0.977	0.15	0.977	0.921	0.152	23
0.173	0.97	0.163	0.973	0.262	0.932	21

Table B7: Performance measurements of model Stats4.

Appendix C

Model training graphs

C1: Model K4Mod37 (First training run)

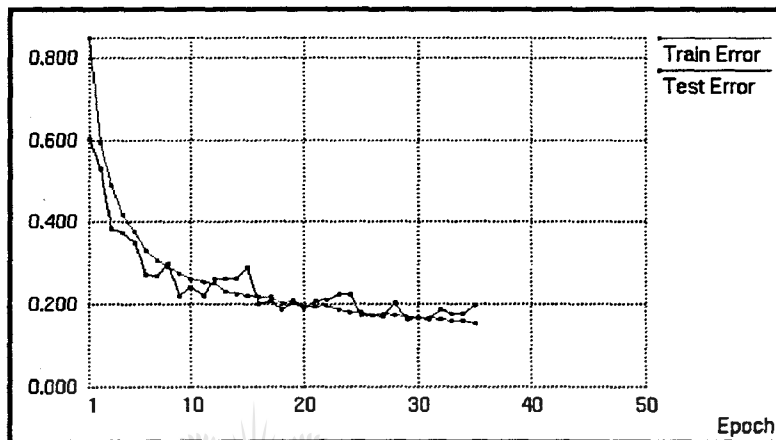


Figure C1: Relative error training graph for first training run of model K4Mod37.

C2: Model K4Mod37 (Last training run)

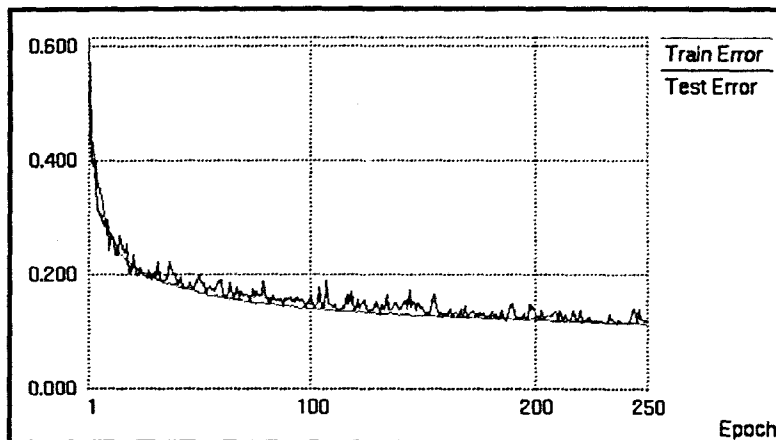


Figure C2: Relative error training graph for last training run of model K4Mod37.

C3: Model Stats2 (First training run)

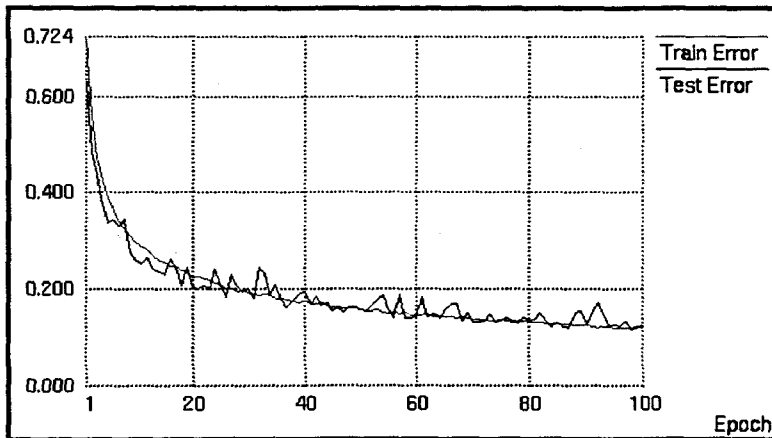


Figure C3: Relative error training graph for first training run of model Stats2.

C4: Model Stats2 (Last training run)

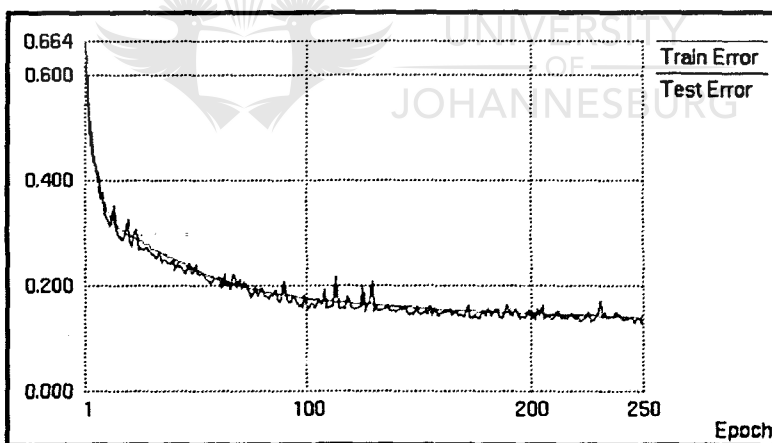


Figure C4: Relative error training graph for last training run of model Stats2.