A Neural Network Approach to the Modeling of Blast Furnace

by

Angela X. Ge

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 22 1999

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author_____

Department of Electrical Engineering and Computer Science
May 22, 1999

Certified by_____

[Amar Gupta]
Thesis Supervisor

Accepted by_____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses
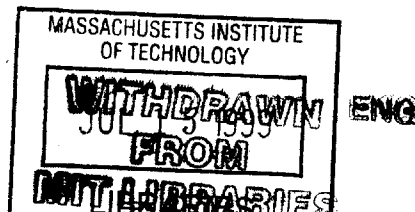
A Neural Network Approach to the Modeling of Blast Furnace
by
Angela X. Ge

Submitted to the
Department of Electrical Engineering and Computer Science

May 22, 1999

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The inside of a blast furnace process is very complicated and impossible to model mathematically. A new approach in this field relies on using artificial neural networks which have proven successful in dealing with complex systems that involve many variables. One of the most important parameters to model is the hot metal temperature - the temperature the output of the blast furnace, pig iron. This thesis presents a multi-layer perceptron neural network which predicts hot metal temperature based on 11 input variables and the actual output value from the previous time period. The best result achieved was a mean squared error of 0.0086 between the predicted hot metal temperature and the actual hot metal temperature.

Acknowledgment

I would like to thank the following people:

Dr. Amar Gupta for providing direction and guidance on this project.

Brad Banks and Ravi Sarma for collaborating with me on this work.

Ashish Agarwal, Sanjeev Vadhavkar for their help and input on blast furnace and neural networks.

Most importantly, I want to thank my parents for their love and support for the past twenty two years.

Table of Contents

1.0 Introduction

Iron working dates back to approximately 1700 B.C. in Europe. The preparation of iron remained the same from the time of the Hittites to the end of the Middle Ages: alternating layers of ore and wood (or charcoal) were heated until a mass of molten ore was obtained. The molten ore was then hammered while hot in order to remove the impurities - and thereby obtain the raw iron, ready to be forged. The forge was set up a few steps away from the hearth where the metal was prepared. Originally a simple conical hole in the ground, the hearth evolved into a furnace, and was gradually perfected. The quantity of the iron produced from the furnace was a few kilograms at first, and then increased to 50 to 60 kilograms by the Middle Ages [1]. From the beginning, small quantities of steel, i.e., iron enriched with carbon, were manufactured. Steel proved to be both harder and more resistant than iron.

Iron production involves a series of complex chemical and thermal reactions as well as intricate mechanical operations. Many variables are involved in the process. Because of the complexity, exact mathematical modeling of these processes has proven to be a difficult, in some instances, almost impossible task. As a result, iron manufacturers often deal with incomplete and uncertain data by relying on the experience of individual experts. Many iron-makers worldwide now use techniques such as expert systems, fuzzy logic, and neural nets to improve quality and efficiency of their iron production [2].

A crucial process in iron production is the use of a blast furnace, which evolved from the traditional furnace. A blast furnace melts down ore and coke. Then, through series of

chemical reductions, it outputs pig iron. Although a number of newer technologies have been introduced for this process, most iron ore is still converted to iron using a blast furnace. More than twenty stories high, the temperature inside a blast furnace can reach as high as two thousand degrees Celsius [3]. Due to the coexistence of several phases and the complex flow conditions with mass and heat transfer inside, a blast furnace is very difficult to model. For many years, blast furnace operators have been aware of the fact that there are no universally accepted methods for accurately controlling blast furnace operation and predicting the outcome.

The performance of the blast furnace is based on pig iron analysis, its temperature and the condition of the slag. The conditions of these output parameters can be influenced, in various degrees, by any of the many variables involved in the blast furnace operations [3].

In order to optimize the operations of the blast furnace, one needs a model that can automatically predict pig iron contents, its temperature, and the condition of the slag. There have been several neural network applications in this field [4] [5] [6]. Based on this information and the critical need for a more accurate model for one of its blast furnaces, Tata Iron and Steel Company (TISCO) in Jamshedpur, India, sponsored a research project at MIT. Our task is to develop a predictive model which exhibits close proximity to the operations of a particular blast furnace at TISCO. The model uses data mining techniques, particularly neural networks.

The neural network technology to date is still an art, rather than an exact science. There are few set rules to follow and it is very difficult to foresee what kind of model would work well for the given set of data. This thesis focuses on how our team at MIT managed to overcome these difficulties, through as much trial and error as deliberate planning and methodical execution.

2.0 Background

2.1 Data Mining

Data mining is the process of intelligently extracting hidden trends and information from corporate and industry databases. In the case of the TISCO data too, it becomes very difficult to find underlying trends using mathematical modeling and conventional OLAP. The use of new techniques can help TISCO to learn more about their blast furnace and help their operators automatically predict the output.

2.1.1 Tasks

Data mining can be used to tackle five types of tasks: classification, prediction, estimation, clustering, and association. Most business and industry problems can be phrased in terms of one or more of the five types of tasks. Each task is performed using one or more underlying techniques. In commercial products, whether or not a technique can perform all the tasks it is capable of depends on the actual implementation.

Classification is defined as examining the features of a newly presented object and assigning it to one of a predefined set of classes [7]. An example of classification is when banks classify each loan applicant as low, medium or high risk. Decision trees, neural nets, genetic algorithms and memory-based reasoning are techniques well suited for this task. Link analysis can also apply in certain cases. Data mining companies usually mean classification when they use terms such as customer profiling, targeted marketing, and churn analysis.

7

Prediction, sometimes called time-series forecasting, is similar to classification or estimation except that the records are classified according to some predicted future behavior or estimated future value [7]. The emphasis here is the dependence of these values on time. Market basket analysis, memory based reasoning, decision trees, and neural nets are all suitable for use in prediction-oriented applications.

Estimation deals with continuously valued outcomes whereas classification only deals with discrete outcomes. Given some inputs, estimation outputs values for some unknown continuous variables. For example, estimating the income of a particular family. Neural nets are well suited for estimation tasks.

Clustering is the task of segmenting heterogeneous population into a number of more homogeneous subgroups, or clusters [7]. Clustering does not depend on predefined classes, which differentiates it from classification and prediction. Clustering can be implemented using market basket analysis, memory-based reasoning, cluster detection, decision trees and neural algorithms.

Association, sometimes called affinity grouping, involves items that occur together in a given event. A rule such as: if item A is part of an event, then x percent of the time, item B is part of the event. Sequence association also falls into this category. (If surgical procedure X is performed, then x percent of the time infection Y will occur.) Market basket analysis, memory-based reasoning, and link analysis are often used to perform association tasks.

The techniques used in data mining are not new. They are borrowed and adapted from other disciplines such as AI (artificial intelligence), graph theory, probability and statistics, and even genetics. Every data mining product incorporates one or more techniques, depending on the level of sophistication of the product. Often, the more techniques a product incorporates, the greater its capability and flexibility, and the higher the price.

2.1.2 Neural Nets

Artificial neural network is one of the oldest and most frequently used techniques in data mining. The strengths of neural nets include their broad flexibility (they can handle a wide range of problems), and their ability to handle both categorical and continuous variables.

Even though a neural net arguably offers the most advanced data mining power, it has several weaknesses. The most serious weakness is that it lacks explicitness - the process is often not explained. It is almost as if the pattern discovery process is handled within a "black box" procedure. The interesting aspect is that a neural net's weakness is also its strongest asset. The difficulty that lies in explaining the process means that the results are often new and non-intuitive. They have never been thought of before, so the results could potentially lead to a radical change of view or the birth of a revolutionary new idea.

2.1.3 Decision Trees

Decision trees are borrowed from AI and statistics. The decision trees algorithms commonly used are CART[1], CHAID[2], and C4.5[3]. CART can only build binary trees and it grows the full tree before pruning it, causing over-fitting[4]. C4.5 is similar to CART except that it can produce varying numbers of branches per node. While CART and C4.5 can accept both categorical and continuous values, CHAID is restricted to categorical variables. Continuous variables will have to be broken down into categories when using CHAID. Unlike CART and C4.5, CHAID does pruning while it builds the tree so that over-fitting does not occur.

There are several major advantages as well as disadvantages in using decision trees. Decision trees generate understandable rules no matter how complicated the inputs are. It is generally easy to follow any one path through the tree, so explaining the decisions along the way is easy. Computation cost for each split is inexpensive. In practice, algorithms tend to produce decision trees with a low branching factor with simple tests at each node. The tree does not grow out of hand and these tests translate into simple Boolean and integer operations that are fast and inexpensive. Using decision trees, the field which is the best at splitting the training records can be singled out for analysis. This enables the user to figure out which variable influences his/her data the most.

---

[1] CART stands for classification and regression trees. The method was published in 1984
[2] CHAID stands for chi-squared automatic interaction detection. It was published in 1975
[3] C4.5 is successor to ID3.
[4] Over-fitting occurs when the information is too detailed. For example, some road maps are very detailed, including every street in a relatively small area. Other maps are more general, covering the major roads in a larger area. Which is the better map to use? If we need to travel a large distance, it may be difficult to figure out the best path from a patchwork of detailed maps. The detailed map overfits the information.

Even though decision trees are very good at classifying data, they are not appropriate for estimation. It is also hard to use decision trees for problems involving time series data unless a lot of effort is put into presenting the data in such a way that trends are made visible.

2.1.4 Memory-based reasoning

Memory-based reasoning (MBR) is also known as nearest neighbors. Like decision trees and neural nets, the idea came from AI. MBR looks for the nearest neighbors in the known instances and combines their values to assign classification or prediction values.

Because MBR generates a list of the nearest neighbors, the results produced can be readily understood. MBR is applicable to arbitrary data types, even non-relational data since the technique does not depend on the underlying representation of the data. The performance of MBR does not depend on the number of fields in the records, which makes it practical to use when other techniques such as neural nets do not perform well.

However, MBR requires costly computation to perform when doing classification and prediction since finding the nearest neighbors involves applying the distance function to all the fields in the record and all the records in the training set.

2.1.5 Genetic Algorithms

Genetic algorithms apply the mechanics of genetics and natural selection to perform a search used for finding the optimal sets of parameters that describe a predictive function. Genetic algorithms are frequently employed to train neural nets. Many neural nets package incorporate genetic algorithms as an option for the training phase.

Genetic algorithms produce results that are explainable. The results can be easily applied because they take the form of parameters in the fitness functions. In many cases, genetic algorithms are used for finding optimal values. They are not limited to the types of input data - as long as the data can be represented as a string of bits of a fixed length.

Even though genetic algorithms are suitable in optimization, they do not guarantee optimality. They may strike a local optima and never find the best solution (global optima). Genetic algorithms can be quite computationally intensive; therefore products incorporating them tend to be enterprise-level products that run on powerful servers.

2.1.6 Link Analysis

Link analysis is sometimes referred to as association analysis. The technique originated from the field of graph theory. Link analysis follows relationships between records to develop models based on patterns in the relationships. It is particularly useful for crime solving and the telephone industry because applications in these areas involve natural

links. Link analysis also offers powerful visualization and makes the process easily understandable.

However, link analysis is not appropriate for many types of problems. It does not apply to classification and prediction like neural nets that take data in and produce an answer. Link analysis finds specific patterns that can then be applied to data.

2.1.7 Market Basket Analysis

Market basket analysis applies probability and statistics measures to the database. The statistic methods used here are different from the standard statistic methods such as regression. Market basket analysis is a form of clustering used for finding groups of items that tend to occur together in a transaction.

The output using market basket analysis is in the forms of rules, which are clear and understandable. When approaching a large set of database without knowing where to begin (i.e., there is no one predefined notions or conjectures), market basket analysis is very useful because it looks for the patterns inherent in the data. It is also simple to use compared to more complex techniques such as genetic algorithms or neural nets.

However, the computational effort grows exponentially as the problem size grows. Even though methods are available to generalize the items analyzed, important rules might be eliminated in the process of generalizing. Market basket analysis works best when all

items have approximately the same frequency in the data. Items that rarely occur are in very few transactions and will be pruned.

2.1.8 Clustering Detection

In the clustering detection approach, there is no pre-classified data and no distinction between independent and dependent variables. Instead, the algorithms search for groups of records - the clusters that are similar to one another, with the assumption that similar records represent similar customers who will behave in similar ways. Cluster detection is rarely used in isolation because finding clusters is not an end in itself. Once clusters have been detected, other methods must be applied in order to figure out what the cluster means.

Cluster detection is undirected data mining - it can be applied even when there is no prior knowledge of the internal structure of a database. Hidden structures are uncovered to improve the performance of more directed techniques. Since there is no need to specify independent values, dependent values, input and outputs, cluster detection is very easy to apply.

The flip side of undirected knowledge discovery is that when one does not know what one is looking for, one may not recognize it even when one finds it. The clusters one discovers may not offer any practical value.

2.2 Neural Networks

In a number of disciplines such as artificial intelligence, physics, psychology, linguistics, biology and medicine, the current focus of research is connectionism. Connectionist systems consist of many primitive cells (units) working in parallel, which are connected via directed links (connections). The main processing principle of these cells is the distribution of activation patterns across the links similar to the basic mechanism of the human brain, where information processing is based on the transfer of activation from one group of neurons to others through synapses.

The human brain exhibits high performance when dealing with highly complex cognitive tasks like visual and auditory pattern recognition. This is a great motivation for modeling the human brain. Because of this motivation, connectionist models are also called neural nets. However, most current neural network architectures do not imitate the biological model in rigid terms

In the current neural network models, knowledge is usually distributed throughout the net and is stored in the structure of the topology and the weights of the links. The networks are organized by (automated) training methods, which greatly simplify the development of specific applications. There are two advantages of using connectionist models. In situations where no clear set of logical rules can be given, the first advantage allows the classical logic in ordinary AI systems to be replaced by vague conclusions and associative recalls (exact match versus best match). The inherent fault tolerance of connectionist models is another advantage. Furthermore, neural nets can be made tolerant against noise

in the input and generally exhibits graceful performance degradation: with increased noise, the quality of the output usually degrades at a slow pace [8].

Neural networks, as an associative memory or as computational algorithms, can perform massively parallel and distributed information processing. A neural network consists of a large number of neurons that exchange information through synaptic connections internally and with the real-world environment. A neuron receives as input a weighted sum of the outputs (or activation) of the other neurons and produces a nonlinear (often step-like or s-shaped) transformation of the input. Individual neurons can only perform a simple signal processing function. However, when many neurons are organized into a number of layers with a specific topology, the networks can provide interesting and sophisticated functions.

Many different network configurations are being proposed in the literature. An especially popular configuration is the multi-layer Feed-forward network. This network consists of an input layer, some hidden layers (which have no direct connections to the environment), and an output layer. The information flows from the environment to the input layer, passes through the hidden layers to the output layer, where the activation of the nodes represents the output of the neural network. A node in an upper (i.e., hidden or output) layer receives as input a weighted sum of the activation of all outputs in the layer below it.

The power of the neural networks originates in the non-linearity provided by the hidden layer. A neural network can outperform all statistical analysis methods such as linear regression, making it especially useful in dealing with a complex process such as the inside of a blast furnace.

3.0 Blast Furnace

Historically, most iron ore is converted to iron using a blast furnace. The production of iron requires three import raw materials: iron ore, "coke" converted from coal, and limestone. Iron is the fourth most abundant element in the Earth's crust. It is found in high concentration in ore deposits, where it resides as oxide. Intense heat is required as an input in the production of iron. Part of the heat is necessary to increase the temperature of the raw materials and part of it is required for endothermic chemical reactions. The coal is converted to a product called coke in coking ovens. Coke plays two important roles in the blast furnace - first as the reducing agent, and second to generate the required heat levels to smelt steel. The limestone acts as a flux to draw off the impurities from the ore.

The three raw materials are added to the top of the blast furnace. A blast of air containing oxygen is injected in from the bottom of the furnace. This causes the coke to burn in the lower part of the furnace, where preheated air and additives, such as oil or pulverized coal, humidity and oxygen, are injected through nozzles, tuyeres, with an intense heat of almost 2000 C. The coke moves further down the furnace to be oxidized to carbon monoxide by reacting with oxygen in the hot blast. The products from the reaction, mainly carbon monoxide and nitrogen, rise through the descending bed of burden materials, to heat, reduce, and melt the iron bearing materials. The partially oxidized gas finally leaves the process at the top of the blast furnace at a temperature of a few hundred degrees Celsius. The temperature in the furnace varies widely: from a few hundred degrees (upper shaft) to about 2000 C (lower zone where the coke is oxidized to carbon

monoxide). The pig iron temperature is about 1450-1500 C. During this process, the

molten iron, which is in contact with coke, absorbs carbon up to its thermodynamic limit

[3]. See Figure 1 for an illustration of a blast furnace and the chemical reactions that take

place inside it. Molten pig iron and slag, present in the lower part of the blast furnace

(bottom and hearth), trickles down from the upper part. They are tapped at irregular

intervals through the taphole.
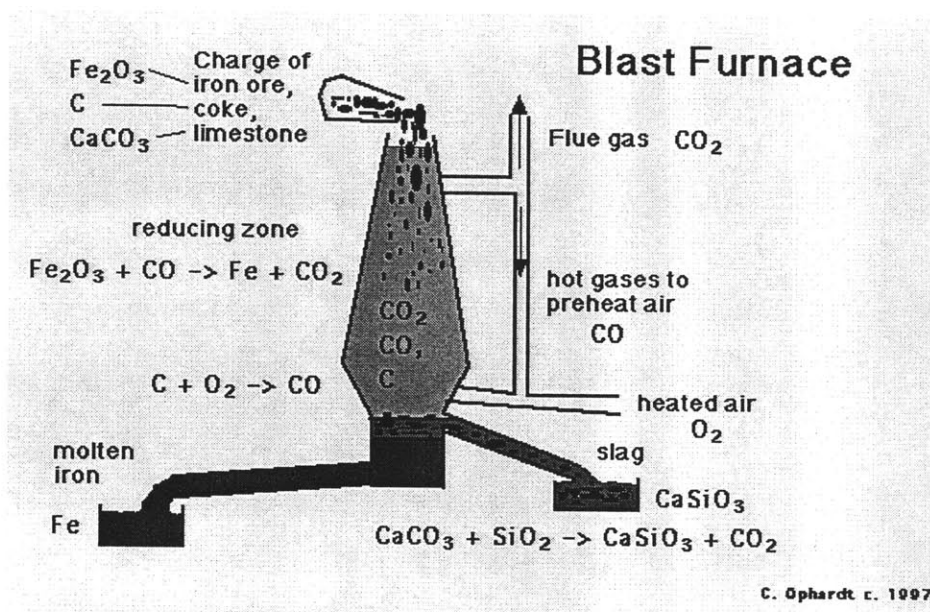
Figure 1: A Typical Blast Furnace [10]

Silicon enters the blast furnace through the ores as well as in coke ash in the form of

oxide and silicates. In the highly reducing atmosphere prevailing in the high-temperature

region of the blast furnace, a part of silicates gets converted via silicon oxide-gas to

silicon dissolved in molten iron. The steel produced from pig iron should not have more

than a certain amount of silicon in it, depending on its end use. An excess of silicon in

the pig iron is uneconomical in the sense that expensive metallurgical coke is wasted in

the blast furnace in the reduction reactions [4].

Impurities in the iron ore such as silicon dioxide react with the limestone to produce slag, $CaSiO_3$, and carbon dioxide. The slag floats on top of the molten iron because of its lower density and can be drawn off separately. The lower hearth is operated batch-wise because metal and slag are tapped intermittently at intervals of a few hours.

The pig iron is treated in a second step called the basic oxygen furnace. Pure oxygen is blown into the molten pig iron to oxidize the impurities of sulfur, phosphorus, and carbon to their respective oxides: SO_2, $P2O_5$, and CO_2. The result of this operation is the production of carbon steel.

The neural network modeling is done on the new 'G' blast furnace TISCO commissioned. It has a working volume of 1800 m^3, a hearth diameter of 9.2 m and a shaft height of 16.5 m [9].

4.0 Stuttgart Neural Network Simulator

The software package used to model the process of the 'G' blast furnace is the Stuttgart Neural Network Simulator (SNNS). It was chosen for its ease of use, portability and availability.

SNNS is a software simulator for neural networks on Unix workstations developed at the Institute for Parallel and Distributed High Performance Systems (IPVR) at the University of Stuttgart. The goal of the SNNS project is to create an efficient and flexible simulation environment for research on and application of neural nets [8].

The SNNS simulator consists of two main components:

1) Simulator kernel written in C

2) Graphical user interface under X11R4 or X11R5

The simulator kernel operates on the internal network data structures of the neural nets and performs all operations of learning and recall. It can also be used its user interface as a C program embedded in custom applications. It supports arbitrary network topologies and the concept of sites. SNNS can be extended by the user with user defined activation functions, output functions, site functions and learning procedures, which are written as simple C programs and linked to the simulator kernel.

The neural networks included in SNNS are the following:

- Feed-Forward Networks

- Time-Delay Networks

- ART Networks

- Self-Organizing Maps

- Auto-associative Memory Networks

- Partial Recurrent Networks such as Jordan Networks and Elman Networks

The graphical user interface XGUI (X Graphical User Interface), built on top of the

kernel, gives a 2D and a 3D graphical representation of the neural networks and controls

the kernel during the simulation run. In addition, the 2D-user interface has an integrated

network editor, which can be used to directly create, manipulate and visualize neural nets

in various ways.

4.1 Neural Network Terminology Used by SNNS

A network consists of units and directed weighted links (connections) between them.

Analogous to biological neurons, each unit receives a net input that is computed from the

weighted outputs of prior units with connections leading to this unit. Figure 2 shows a

small network.

The actual information processing within the units is modeled in the SNNS simulator

with the activation function and the output function. The activation function first

computes the net input of the unit from the weighted output values of prior units.

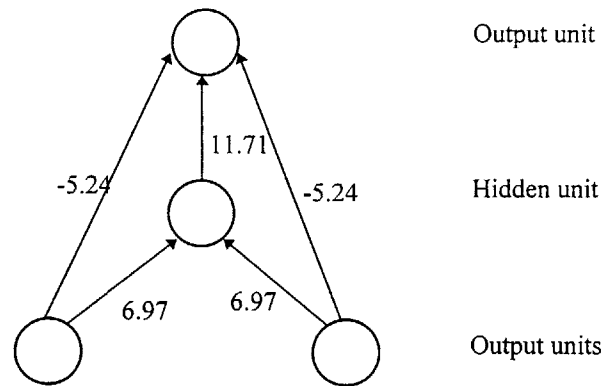It then computes the new activation from this net input (and possibly its previous

Figure 2: A small network with three layers of units

activation). The output function takes this result to generate the output of the unit. These

functions can be arbitrary C functions linked to the simulator kernel and may be different

for each unit. The graphical user interface XGUI (X Graphical User Interface), built on

top of the kernel, gives a 2D and a 3D graphical representation of the neural networks and

controls the kernel during the simulation run. In addition, the 2D-user interface has an

integrated network editor, which can be used to directly create, manipulate and visualize

neural nets in various ways.

The SNNS simulator offers the use of sites as additional network element. Sites constitute

a simple model of the dendrites of a neuron, facilitating the grouping of the expectation of

different treatment of the input signals of a cell. Each site can have a different site

function. This selective treatment of incoming information allows the creation of more

powerful connectionist models. Figure 3 shows one unit with sites and one without.

Depending on their function in the net, one can distinguish three types of units: the units whose activations are the problem input for the net are called input units; the units whose output represent the output of the net output units; and the remaining units which are called hidden units, because they are not visible from the outside.

In most neural network models, the type correlates with the topological position of the unit in the net. If a unit does not have input connections but only output connections, then it is an input unit. If it lacks output connections but has input units, it is an output unit. If it has both types of connections, it is a hidden unit.

The direction of a connection (link) shows the direction of the transfer of activation. The unit from which the connection starts is called the source unit, or source for short, while the other is called the target unit, or target. Connections where source and target are identical (recursive connections) are possible. Multiple connections between one unit and the same input port of another unit are redundant, and therefore prohibited. This is checked by SNNS.

Each connection has a weight (or strength) assigned to it. The effect of the output of one unit on the successor unit is defined by this value: if it is negative, then the connection is inhibitory, i.e. decreasing the activity of the target unit; if it is positive, it has an excitatory, i.e. activity enhancing, effect.

The most frequently used network architecture is built hierarchically bottom-up. The

input into a unit comes only from the units of preceding layers. Because of the

unit with sites   unit without sites

to other units   to other units

output function ——   output function ——

site value ——
site function ——
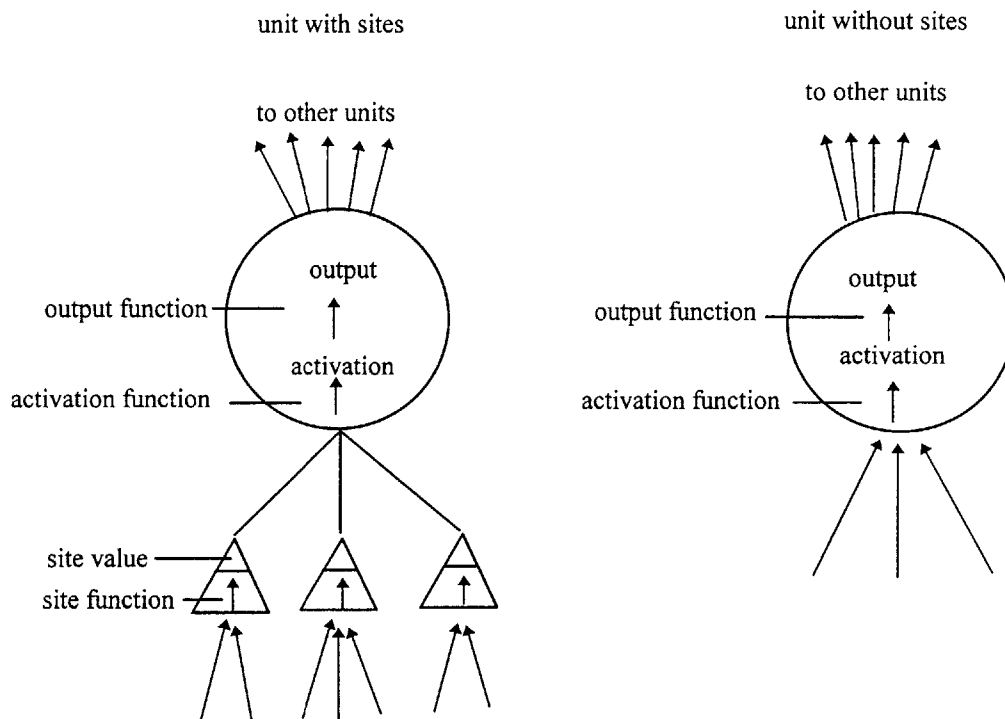
Figure 3: One unit with sites and one without

unidirectional flow of information within the net they are also called feed-forward nets

(as example see the neural net classifier introduced in chapter). In many models, a full

connectivity between all units of adjoining levels is assumed.

A unit with sites does not have a direct input any more. All incoming links lead to

different sites, where the arriving weighted output signals of preceding units are

processed with different user-definable site functions (see Figure 3). The result of the site

function is represented by the site value. The activation function then takes this value of

each site as the network input.

The SNNS simulator does not allow multiple connections from a unit to the same input port of a target unit. Connections to different sites of the same target units are allowed. Similarly, multiple connections from one unit to different input sites of itself are allowed as well.

4.2 Update Modes

To compute the new activation values of the units, the SNNS simulator running on a sequential workstation processor has to visit all of them in some sequential order. This order is defined by the Update Mode. Five update modes for general use are implemented in SNNS. The first is a synchronous mode, all other are asynchronous, i.e. in these modes units see the new outputs of their predecessors if these have fired before them.

1. Synchronous: The units change their activation all together after each step. To do this, the kernel first computes the new activations of all units from their activation functions in some arbitrary order. After all units have their new activation value assigned, the new output of the units is computed. The outside spectator gets the impression that all units have fired simultaneously (in sync).

2. Random permutation: The units compute their new activation and output function sequentially. The order is defined randomly, but each unit is selected exactly once in every step.

3. Random: The order is defined by a random number generator. Thus it is not guaranteed that all units are visited exactly once in one update step, i.e. some units may be updated several times, some not at all.

4. Serial: The order is defined by ascending internal unit number. If units are created with ascending unit numbers from input to output units, this is the fastest mode. Note that the use of serial mode is not advisable if the units of a network are not in ascending order.

5. Topological: The kernel sorts the units by their topology. This order corresponds to the natural propagation of activity from input to output. In pure feed-forward nets the input activation reaches the output especially fast with this mode, because many units already have their final output which doesn't change later.

4.3 Learning in Neural Nets

An important focus of neural network research is the question of how to adjust the weights of the links to get the desired system behavior. This modification is very often based on the Hebbian rule, which states that a link between two units is strengthened if both units are active at the same time.

Training a feed-forward neural network with supervised learning consists of the following procedure. An input pattern is presented to the network. The input is then propagated forward in the net until activation reaches the output layer. This constitutes the so-called forward propagation phase.

The output of the output layer is then compared with the teaching input. The error, i.e. the difference (delta) between the output and the teaching input of a target output unit j is

then used together with the output of the source unit i to compute the necessary changes

of the link. To compute the deltas of inner units for which no teaching input is available,

(units of hidden layers) the deltas of the following layer, which are already computed, are

used in a formula given below. In this way the errors (deltas) are propagated backward, so

this phase is called backward propagation.

In online learning, the weight changes are applied to the network after each training

pattern, i.e. after each forward and backward pass. In offline learning or batch learning

the weight changes are cumulated for all patterns in the training file and the sum of all

changes is applied after one full cycle (epoch) through the training pattern file.

The most famous learning algorithm, which works in the manner described, is currently

backpropagation. In the backpropagation learning algorithm online training is usually

significantly faster than batch training, especially in the case of large training sets with

many similar training examples [8].

4.4 Generalization of Neural Networks

One of the major advantages of neural nets is their ability to generalize. This means that a

trained net could classify data from the same class as the learning data that it has never

seen before. In real world applications developers normally have only a small part of all

possible patterns for the generation of a neural net. To reach the best generalization, the

dataset should be split into three parts:

- The training set is used to train a neural net. The error of this data set is minimized during training.

- The validation set is used to determine the performance of a neural network on patterns that are not trained during learning.

- A test set for finally checking the over all performance of a neural net.

Typically, a data set is divided into the training set and the testing set. In general, the split is based on an 80-20 rule: 80 percent of the data set is used for training and the rest 20 percent is used for testing. This is a good proportion for splitting the data set. Some use the 75-25 rule or the 70-30 rule [4]. Many vary the rule and adjust the training and testing sets to see how it affects their relates. Unless specifically mentioned, all of our modeling is done based on the 80-20 rule.

The learning should be stopped in the minimum of the validation set error. At this point the net generalizes best. When learning is not stopped, over-training occurs and the performance of the net on the whole data decreases, despite the fact that the error on the training data still gets smaller. After finishing the learning phase, the net should be finally checked with the third data set, the test set.

SNNS performs one validation cycle every n training cycles. Just like training, validation is controlled from the control panel.

It was decided that only neural network models that have one layer of hidden nodes will be explored for this modeling effort. This decision was based on past experience as well as some literature search [4] [5]. The power of neural nets comes from its non-linear nature and the hidden layer provides that capability. Using two hidden layers provides the same quality of non-linearity as using one hidden layer, so it is not necessarily better to increase the number of layers. It has been shown, empirically, using two or more hidden layers in their models hurt the performance [4] [5]. In this case, a single layer of hidden nodes is sufficient.

5.0 Methodology

In the context of this thesis, a neural network model is a set consisting of a specific neural network algorithm, a learning algorithm (including the learning rate) for the neural net, and a correct representation of data to be modeled. Using a set of inputs, the objective is to formulate a neural network model that can accurately predict the various output aspects of TISCO's 'G' blast furnace.

Before the selection of a neural network algorithm and a learning algorithm is made, an important factor is capturing the most appropriate *state* of the blast furnace. The *state* of a blast furnace is defined to be a description of its condition at a point in time. The description can encompass every measurable detail of the blast furnace, each represented as an input parameter. The challenge lies in defining the scope of the *state* - which parameters to include in the model. The raw data received from TISCO, based on its past operational records, consist of 35 available input parameters for the prediction of hot metal temperature, each taken at 5-minute intervals. Due to structural and chemical complexities, not all the parameters of the blast furnace are useful for predicting a certain aspect of the output. The selection of the appropriate *state* (i.e. proper set of parameters) for a particular aspect of the output has a great impact on the model. Incorporating parameters that have little relevance to the particular output parameter would cause excessive noise in the model. Using too few of the highly related parameters gives an inadequate representation of the *state* of the blast furnace, thus producing an unrealistic model. Furthermore, depending on which output parameter is being modeled, the selection of the appropriate state of the blast furnace could be totally different.

5.1 Output Parameters

A blast furnace is used to produce pig iron. The quality and the quantity of the different input materials as well and many environmental factors all influence the quality of the pig iron. The three most important parameters that define the quality of the pig iron are hot metal temperature, hot metal silicon content, and slag basicity. This thesis focuses mainly on the prediction of hot metal temperature, with brief descriptions of the other two output parameters. The prediction of hot metal temperature was the first to be examined in the modeling effort since it is used as an input variable in the modeling of hot metal silicon content and slag basicity. The work done in the case of the other two output variables followed similar paths as the modeling of hot metal temperature.

5.1.1 Hot Metal Temperature

Hot metal is also referred to as pig iron and its temperature is in the range of 1450 - 1500 degrees Celsius. Very crucial in determining the quality of the pig iron, hot metal temperature is an important parameter in both the control and the prediction of the blast furnace [3]. The temperature of hot metal is measured whenever pig iron is tapped. The tapping is done at irregular intervals.

5.1.2 Hot Metal Silicon Content

Hot metal silicon content is important both for quality and control purposes. Temperature of the hot metal, silica activity in the slag, oxygen activity in the hot metal and other

solutes that are present directly influence the silicon content. Not only is silicon content an important quality parameter, it also reflects the internal state of the high-temperature lower region of the blast furnace where many of the dynamic changes in pig iron silicon content correspond to similar changes in the pig iron temperature and/or shifts in the heat level, making it an important tool for blast furnace control. Often, an operator can judge the extent to which the other reactions have proceeded based on the extent of the silica reduction. Uniformity in silicon content and its accurate and advance prediction can greatly help to stabilize blast furnace operations. The silicon content of the hot metal is measured when pig iron is tapped. After the pig iron leaves the blast furnace, it goes through another process become steel. The silicon content of the pig iron determines the charge balance in oxygen steel-making converters - an important tool in making steel [4].

5.1.3 Slag Basicity

Slag, along with $CaSiO_3$ and carbon dioxide, is produced when impurities in the iron ore, such as silicon dioxide, react with limestone. Once produced, it floats on top of the molten iron and can be drawn off separately. In reduction processes in which liquid iron is produced, the slag plays an important role. It absorbs components contained in the raw material that are undesirable in iron, such as silicon dioxide. Slag can be easily separated from the metal because of its low viscosity, low density and the low vapor pressure of its components. Even though slag is essentially the waste product of the chemical reactions, it has properties suitable to make it salable. The quality of the hot metal is determined by the concentration of silicon, manganese, phosphorus and sulfur. These concentrations depend on the temperature, the slag basicity (in its simplest but not entirely accurate form

33

expressed by the lime to silica ratio, CaO:SiO₂) and the ratio of the slag and metal

quantities. The desulfurization of the hot metal increases with slag basicity [3].

5.2 Input Parameters for Prediction of Hot Metal Temperature

For the prediction of hot metal temperature, TISCO provided a data set consisting of 35

input variables. A list of the more important variables and a description of each is given

in Table 1. An in-depth knowledge of metallurgy and blast furnace is required to

understand these inputs. Since the focus of this thesis is the modeling aspect , the

understanding of each input variable is not necessary. Therefore, only a brief description

of each variable is given here.

5.3 Normalization of the Data Set

The process of treating the data set used by the neural networks is just as important as

deciding what type of neural networks to use. Not only is normalization a way to get the

data into an acceptable format to neural algorithms, it is also used to ensure that all

parameters can be compared and contrasted against each other easily, by scaling them

down to within the same range of values. For example, hot metal temperature has values

in the range of thousands of degrees while others have a range of between 0 and 1.

In order for the inputs to be acceptable to neural networks, all values of the parameters

were normalized to between 0 and 1. To maximize the capability of the neural networks,

values need to be spread as evenly between 0 and 1 as possible, generating sharper

contrast and more deviation among values. This conclusion was reached after some

Input Variable	Description
RAFT_TH	RAFT temperature
G1HF	Group 1 heat flux (lower part of blast furnace)
G2HF	Group 2 heat flux (upper part of blast furnace)
FIC46	Steam (tons/hr)
INJ_ACT	Actual coal injection
EH2	Eta hydrogen (from top gas analysis)
QZTOTAL	Total quartz
DP	Differential pressure
CWINDEX	Central working index
LSTOTAL	Total limestone
PI_59	Wind volume
CMOIST7	Coke moisture from bin #7
AL12	Top gas H2%
CMOIST9	Coke moisture from bin #9
AL13	Top gas CO%
PERM_K	Permeability index, K
ECO	Eta CO (from to gas analysis)
COKTOTAL	Total coke
O2ENR_PV	Oxygen enrichment, %
NCKTOTAL	Total nut coke
SINTOTAL	Total sinter
SINTPERC	% sinter in burden
ORETOTAL	Total ore charged
CHRGDTIM	Charge time for 10 semi charges
PI92	Hot blast pressure
OREBYCOK	ore/coke ratio
PIC33	Top pressure in kg/cm^3
AL13	Top gas CO%
DOLTOTAL	Total dolo
THC_R	Theoretical carbon rate

Table 1: List of Input Parameters

initial failure where the neural nets provided "flat" results, and all the data points

clustered around the middle. The problem disappeared once the values were spread out

using an adjusted normalization scheme. The presence of negative values for certain

parameters required an adjustment on the normalization scheme in order to accommodate

it. After examining the data set, the following normalization scheme was proposed:

$$(\ | \ Value \ | - | \ Min \ | \ ) \ / \ (\ | \ Max \ | \ - | \ Min \ | \ )$$

where | Value | is the absolute value of the variable at the specific times. | Min | is the minimum of the absolute values of that variable in the data set. |Max | is the maximum of the absolute values of that variable. Taking the absolute value of everything eliminates negative numbers. The normalized data set contains approximate 8000 data points at 5-minute intervals. Certain values were completely out of range compared to values in the same column - mistakes made when the data was recorded. They were manually removed since the number of irregular data points is less than 15. The integrity of the data set was not compromised because the number of irregulars is extremely small compared to the total number of data points in the data set.

5.4 Sensitivity Analysis

Before proceeding with neural net modeling, each input variable is examined in relation to the output variable, hot metal temperature. Naturally, some input parameters have a bigger impact on the output variable than others. Two methods were used to gauge how well an input variable relates to the output variable: plotting the input with the output and visually examining them; and calculating correlation coefficients. Based on operational experience, TISCO's blast furnace controllers recommended some variable that are known to have a high impact on hot metal temperature. The same methods mentioned above were used to verify whether or not these variables were indeed highly correlated to hot metal temperature, as well as to explore alternative high-correlation parameters.

Each input parameter was plotted with hot metal temperature on the same graph in order to detect patterns. There is one graph for each input variable, resulting in 35 graphs total. Comparing the input variable and the output variable graphically can show the high level relationship and the general trend. Any abnormality in the trend of each variable can be easily detected. Figure 4 shows the graph of one of the variables. The input variable is plotted in solid lines and hot metal temperature is plotted in dotted lines. The x-axis represents the time index (at 5-minute interval). The y-axis represents the normalized value of the input/output variables at the corresponding time index.

Most likely, the actual values of the input parameter and hot metal temperature have very different ranges, therefore, all the graphs shown in this thesis are plotted in terms of normalized values to allow more than one curve to be plotted on the same graph for comparison purposes. This method of plotting clearly indicates that all the normalized values were spread evenly between 0 and 1.

By examining the graphs visually, it was obvious that hot metal is a step-like function while the input parameters vary at each interval, indicating that either hot metal temperature naturally stays constant for a time period or that the input parameter
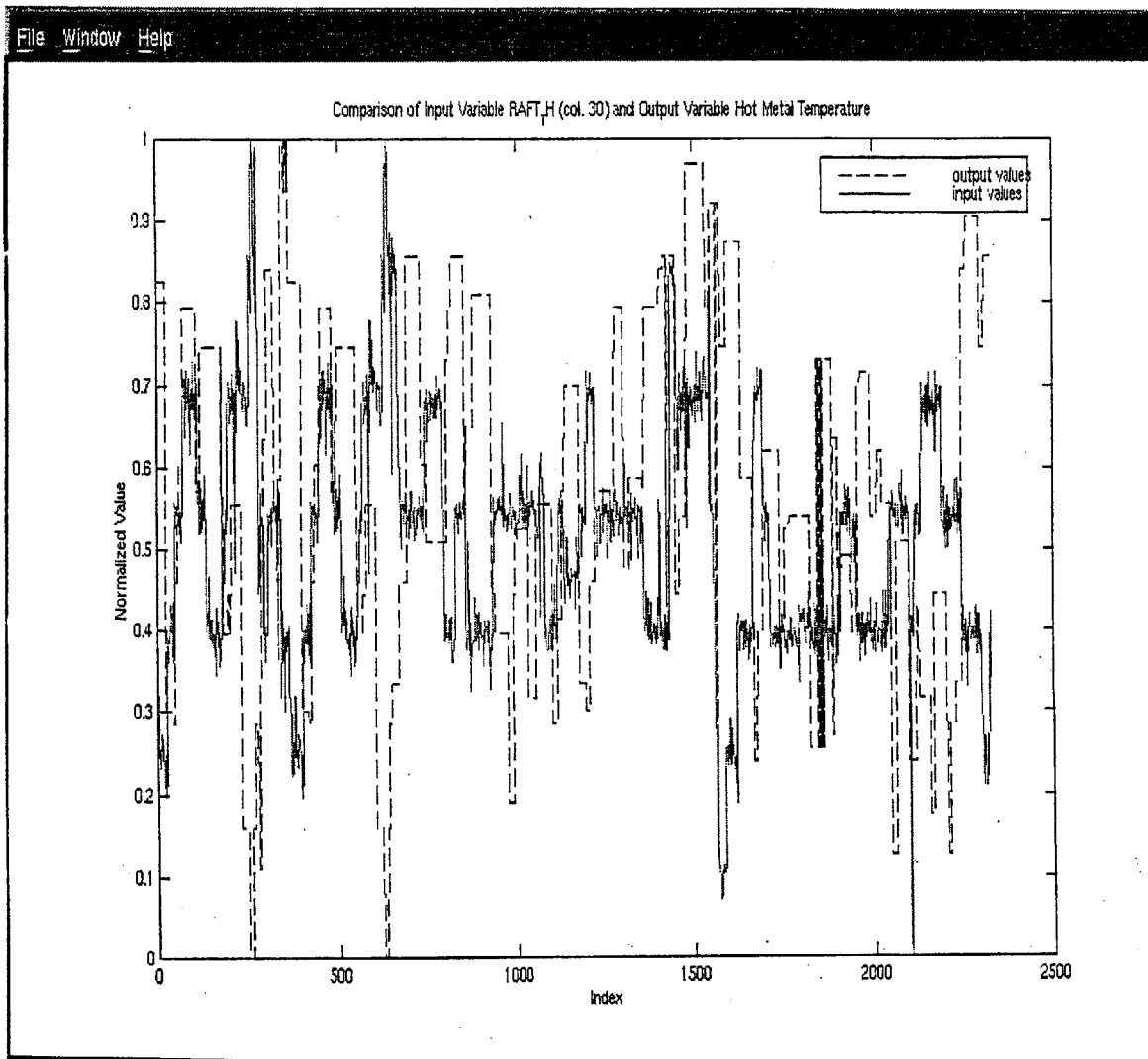
Figure 4: Input Parameter RAFT_TH with Hot Metal Temperature

is measured more frequently than the output parameter. It is unlikely that the hot metal temperature would stay constant for a long period of time given the nature and the complexity of the blast furnace. Therefore, it is more likely that the measuring frequencies of the input parameters and hot metal temperature are not the same. The suspicion was confirmed by TISCO - while the input variables are measured and recorded at 5-minute intervals, hot metal temperature is only measured when the hot metal is tapped, irregularly, at hourly intervals on average. The out-of-sync measuring of input and output variables would prove to be an obstacle later on.

The second step in the sensitivity analysis is to examine the data from an empirical angle. This involves calculating correlation coefficient of each input variable against the output variable. This method has been used in prior works in this field [4] [5]. Correlation coefficient is a commonly used statistic measurement of how closely one variable trails another variable. It is similar to another measure, covariance. Covariance also measures how much the two variables move in tandem. However, the correlation coefficient scales the covariance to a value between -1 (perfect negative correlation) and +1 (perfect positive correlation) by dividing their covariance by the product of the standard deviations of the two variables. Correlation coefficient is easier to interpret and compare because of the scaling. See Table 2 for a list of all the correlation coefficients of the input variables. The positive sign of the correlation coefficient indicates that the input parameter "moves" in the same direction as the output parameter. The negative sign indicates that the input parameter "moves" in the opposite direction as the output parameter. The greater the absolute value of the coefficient is, the higher the correlation.

The largest correlation coefficient is -0.3409 while most of the correlation coefficients are under plus or minus 0.10, indicating weak relationships between individual input parameters and hot metal temperature in general. If incorporated into a model, these variables would most likely introduce more noise than useful information. In the beginning, the *state* of the blast furnace constitutes all the 35 variables provided by TISCO. Later on, the scope of the *state* will be narrowed by eliminating low-correlation variables in order to reduce the noise and improve the model.

5.5 Linear Regression Model

A linear regression model is essentially a neural network with only one hidden node; therefore it does not have the nonlinear characteristic that makes a neural network so powerful. Linear regression models were built as benchmarks for the neural network models. Because a neural network is capable of learning linear as well as nonlinear relationships, they are expected to perform at least as well as linear regression models, if not better [4]. The benchmark linear regression models can give an estimate of the quality of the data set and detect any potential errors in using the neural network software - if the neural network results turn out to be significantly poorer than the results of the linear regression model, then either the software was not used properly or there is some anomaly in the data set. The result of a linear regression model using all 35 input variables is shown in Figure 5.

Input Variable	Correlation Coefficient
AL12	0.101
AL13	0.0070
AL14	-0.0629
BGVEL	-0.1868
BGVOL	0.1860
CHRGDTIM	0.0584
CMOIST7	-0.0788
CMOIST9	-0.1052
COKTOTAL	0.0310
CWINDEX	0.1466
DOLTOTAL	0.0898
DP	-0.1199
ECO	-0.0235
EH2	0.1860
FI59	-0.1092
FIC46	0.2491
G1HF	-0.3065
G2HF	-0.3134
INJ_ACT	-0.2117
LSTOTAL	-0.1132
NCKTOTAL	-0.0437
O2ENR_PV	-0.0393
OREBYCOK	-0.0625
ORETOTAL	-0.0538
PERM_K	0.0148
PI89	-0.0594
PI92	-0.0606
PIC33	0.0627
QZTOTAL	0.1240
RAFT_TH	-0.3409
SINTOTAL	-0.0529
SINTPERC	0.0529
THC_R	N/A
TIC91	-0.0759
TUYVEL	-0.0934
TY21_PV	Not Used

Table 2: Correlation Coefficients of Input Variables against Output Variable
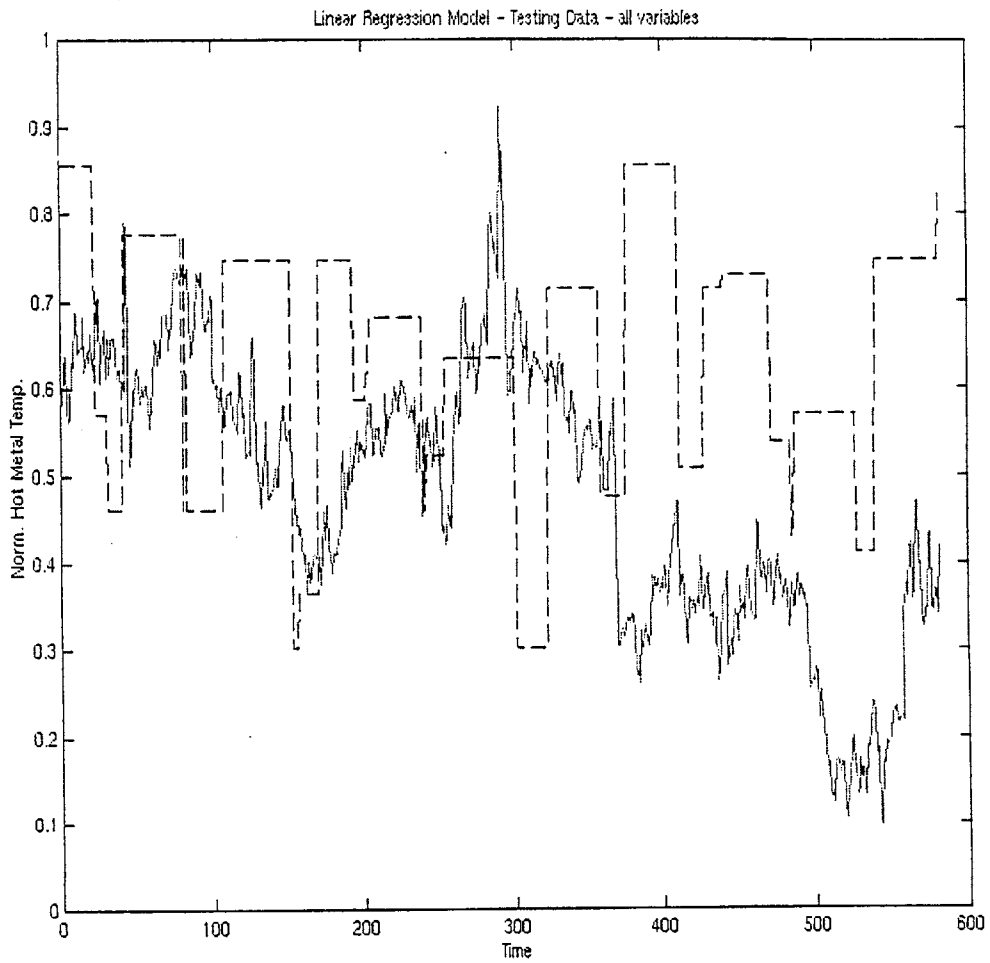
Figure 5: Results from Running the Linear Regression Model on 35 Input Variables

The linear regression model running on the 35 input variables showed poor results which indicates that the quality of the data set needs to be improved. The next step is to run some neural network models using the same set of data to see if they can outperform the linear regression model.

5.6 Results from Running the Neural Network Model on the Initial Data Set

Three different neural network algorithms, Feed-forward, Time-delay, and Jordan; with standard Back-propagation were run using SNNS. Feed-forward (also known as Multi-layer perceptron) networks are standard and have been shown to solve many different types of problems. Based on this information, Feed-forward networks were chosen to be run initially. In additionl, Time-delay and Jordan networks were selected because of time-dependencies in the data, that is, there is a time delay before a particular set of inputs at one time instant affects the hot metal temperature. Even though they are more complex, Time-delay and Jordan networks are known for their ability to distinguish time-dependencies in the data set [8]. The learning algorithm used is standard Back-propagation. Backpropagation is applicable to a wide variety of problems and has proven itself very successful in modeling nonlinear systems [5].

The results from running the Feed-forward neural network are shown in Figure 6. The results from the Time-delay and the Jordan models (not shown) are very similar to the Feed-forward model. The Feed-forward model has a mean squared error (MSE) of 0.036. MSE is a measure of how close predicted and actual output values are.
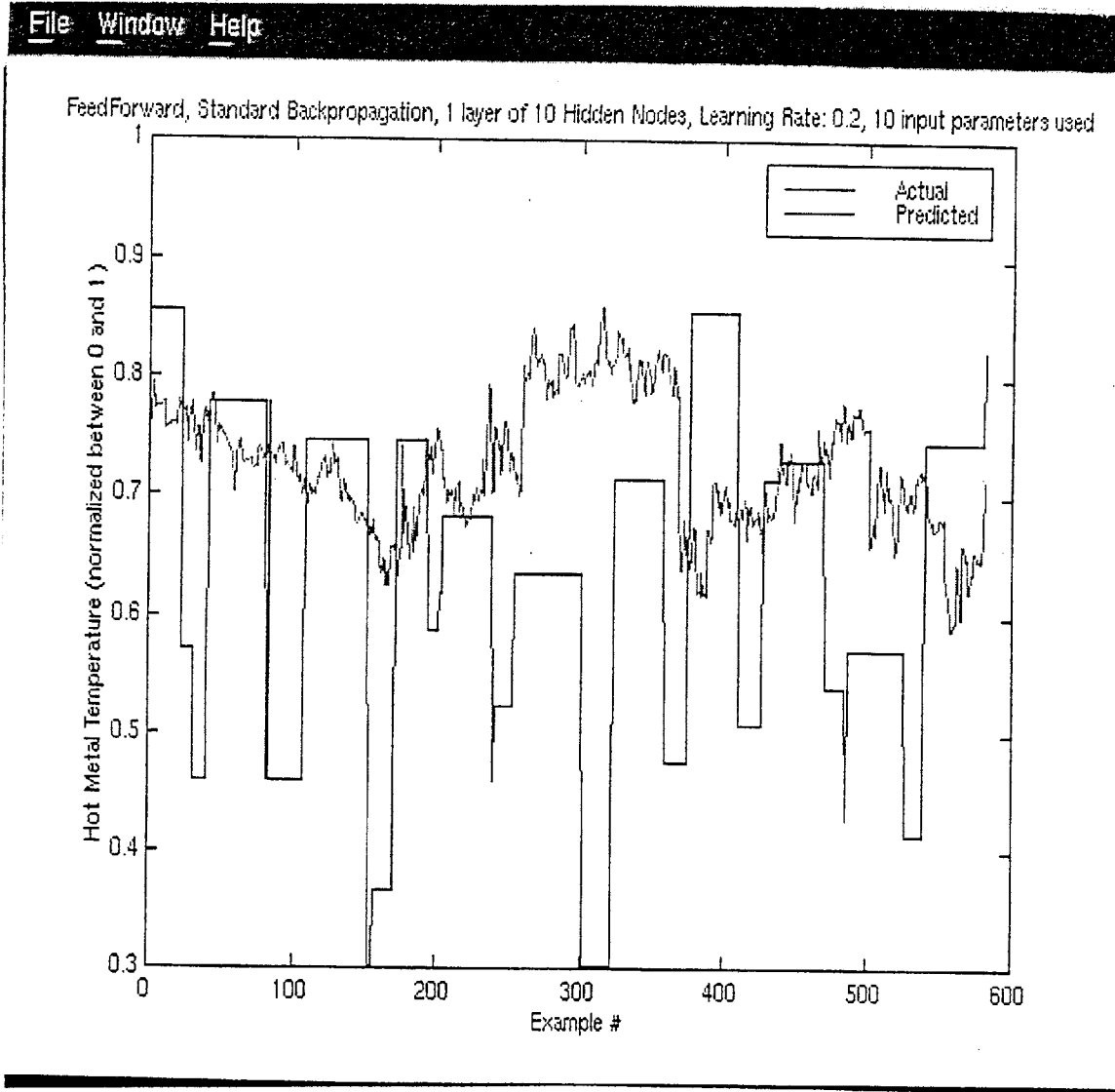
Figure 6: Results from the Feed Forward Neural Net Model

The results from running the neural networks mentioned above lack substantial predictive power. A visual examination of the two graphs indicates that the neural network model does not outperform the linear regression model by much. This is largely due to the infrequent measurements of hot metal temperature. The existing data set does not give a realistic representation of the changes in hot metal temperature.

5.7 Limitations of the Data Set

The biggest obstacle to building better models is the lack of frequency with hot metal temperature, is measured. Section 4.4 first hinted at this difficulty, and this section will elaborate on that. Under real operational settings, it is practically impossible to obtain more frequent measurements, since hot metal temperature is measured whenever the metal is tapped from the blast furnace and it is too expensive and impractical to tap the metal more frequently. The original format of the data displays a constant value for temperatures between hot metal temperature measurements. Thus, the output generally stays constant for 8-14 data points (40-70 minutes), and then jumps to the value of the next measurement, where it again stays constant until the next jump. Obviously this is an unrealistic model of the physical process of temperature change. Neural networks tend to perform poorly when modeling data that exhibit step-like patterns.

To overcome this difficulty, hot metal temperature is modeled at hourly intervals where an actual measured value is available. Temperature for the data points between actual measurements were linearly interpolated. Linear interpolation connects one measurement to the next with a line; the correct temperature between measurements is assumed to lie

on this line. This is clearly a more realistic representation and more accurately captures

the relationship between input variables and hot metal temperature.

Even though linear interpolation is the best way to treat the data set under the

circumstance, temperature between measurements do not strictly lie on a straight line.

Something is needed to "soften" this assumption, such as taking the hourly average data

to narrow the difference between the linearized data and the behavior of the original data

set. Therefore, after linearly interpolating the hot metal temperatures, the data set was

sliced into blocks of 12 data points (because each input parameter is measured

approximately every 5 minutes). For each block, the 12 values for each of the 35 inputs

were averaged, as were the 12 output values. Each 12-data point block was reduced to a

single, hourly-averaged data point. The resultant data set is reduced to one twelfth of the

original data set and now only contains 702 data points. The data set is referred to as the

hourly-averaged-data from now on.

5.8 Correlation and Time Lag

Because the production of hot metal is such a complex and lengthy process, it is

inevitable that there are time lags between inputs and output [4]. A change in the value of

input may not have an effect on the output until hours later since on average, coal and ore

take a few hours to travel from the top of the blast furnace to the bottom; during which

chemical reactions occur. Furthermore, this lag may vary across different inputs (i.e. each

input parameter has a different time lag with hot metal temperature). To explore the

effect of this time lag, correlation coefficients for each input variable for up to 14 hours

from the output, using the hourly-averaged-data were calculated, thus empirically finding

the best estimate of the lag effect for each input variable. Taking this into account yields

a more accurate model.

The correlation coefficient between the output and each input variable for a number of

lags between 0 and 14 hours were calculated. For each input variable, the optimal time

lag is determined by the time lag that has the largest correlation coefficient. The results

can be seen in Table 3. The positive sign of the correlation coefficient indicates that the

input parameter "moves" in the same direction as the output parameter. The negative

sign indicates that the input parameter "moves" in the opposite direction as the output

parameter. The first column is the name of the input variable, the second column

contains the highest correlation coefficients for each corresponding input variable, and

the third column contains the time lag with the highest correlation coefficient.

Initially, some of the correlation coefficients seemed wrong. Based on the experience of

blast furnace controllers, certain parameters have complete opposite signs than the ones

computed. For example, it is fairly well known that RAFT_TH has a positive correlation

with hot metal temperature. The strong negative correlation was surprising. The

correlation that model is capturing is the reaction of the operator. When the blast furnace

operator sees that the hot metal temperature is dropping, one of his first reactions is to

raise RAFT_TH.

Input Variable	Correlation Coefficient	Time Lag
AL12	0.2601	0
AL13	-0.2162	9
AL14	-0.0502	7
BGVEL	-0.2585	2
BGVOL	0.1470	10
CHRGDTIM	0.1566	0
CMOIST7	0.2289	14
CMOIST9	0.1312	13
COKTOTAL	-0.2758	9
CWINDEX	0.1403	6
DOLTOTAL	0.2376	0
DP	0.1888	11
ECO	0.2153	9
EH2	0.1372	0
FI59	0.2140	11
FIC46	0.3281	0
G1HF	0.1731	14
G2HF	-0.3279	0
INJ_ACT	-0.1153	4
LSTOTAL	0.1714	0
NCKTOTAL	-0.0582	1
O2ENR_PV	-0.1366	14
OREBYCOK	0.0902	0
ORETOTAL	0.1231	0
PERM_K	0.2623	3
PI89	0.1839	11
PI92	0.1785	11
PIC33	0.1360	10
QZTOTAL	0.1366	1
RAFT_TH	-0.4253	0
SINTOTAL	0.0233	2
SINTPERC	-0.0667	0
THC_R	N/A	0
TIC91	-0.1624	7
TUYVEL	-0.1710	2
TY21_PV	Not Used	Not Used

Table 3: Correlation coefficients of input parameters to the output parameter and related time lags.

Same is the case with COKTOTAL and INJ_ACT. With AL12 and FIC46, the correlation was expected to negative, but the correlation coefficients are positive. At high hot metal temperature, the operator raises steam to lower the temperature. Such reactions are always present in complex processes.

After calculating the optimal time lag for each of the input parameters (as described above), the data set was adjusted in order to implement these lags. Each column was shifted according to its respective time lag with hot metal temperature.

5.9 Difference Between Training and Testing Data

When the models were run on the adjusted hourly-averaged-data set, which accounts for time-dependencies, the results did not improve as much as expected. The problem was traced to an inconsistency between the training data set and the testing data set. The testing set has an unusual trend, which is inconsistent with the training set. This is most likely because the testing set was constructed by holding out a block of 100 hours worth of data. Perhaps there are novel operating conditions in that block of 100 hours that did not occur during the hours captured in the training set. If so, those conditions will cause the test set to yield poor results. Constructing the hourly-averaged-data set caused the number of data to shrink to one-twelfth of its original size, so using only the training data set and discarding the testing set is not viable.

A solution to this problem is making the testing and training sets more uniformly sampled. This way, instead of holding out one large block of data, sample hours from the

entire period covered by hourly-averaged-data were selected. 200 randomly selected data

points were chosen to be the testing data set. The remaining points make up the training

set. This ensures that the testing and training data sets are more uniform.

5.10 Results

Using the randomized training and testing data sets, a linear regression model was trained

to set the benchmark. Then several feed-forward neural network models and Jordan

recurrent neural network models were run. Every model was run with a different

applicable learning algorithm and a different learning rate.

These changes have given the best results so far. Using the hourly-averaged-data set,

applying lag information, and revamping the train and test sets, the MSE for all types of

models were lowered significantly. Table 4 illustrates this.

Model	Old MSE	New MSE
Linear Model	0.0735	0.0114
Feed Forward ANN (25 hidden)	0.0415	0.0098
Jordan Rec. ANN (12 hidden)	0.0415	0.0098

Table 4: Results

5.11 Revising the Data Set

Still, the model's predictive performance could be better. One of the most urgent

problems is the lack of data. The number of available data points is not adequate for the

number of input variables modeled. Obviously, the more data points used to train the

neural networks, the better the results. Realistically, however, most applications do not

have tons of data ready at hand. Generally, the minimum number of data points required

depends on the number of input variable used. The more input variables used in the model, the larger number of data points are needed.

There are two ways to improve the results: using more data or manipulating the existing data set to increase the number of data points. Ideally, the first method is preferable since manipulating the data set distorts some of its characteristics. However, since more data points was not easily obtainable, the only choice was to use the second method. A moving window of size 12 units was suggested. In the linearized (before averaging) data set (at 5-minute intervals), the first 12 data points were averaged to produce one data point in the new data set. For the second data point in the new set, another 12 data points (starting from data point 2 in the old set) were averaged. Similarly, for the third data point, the average was taken starting from data point 3 in the old set, so on. In effect, there is a moving window of size 12. A new data point is obtained by averaging all the data points inside windows. The window slides by 1 every time. For example, the window first averages data points 1 through 12 to produce the first data point, then data points 2 through 13 to produce the second data point, and then3 through 14 for the third data point. Using this scheme, there are 702*12 = 8424 data points instead of just 702. The downside of this scheme is that the new data set reflects less of the quality of the original data set, hence the real behavior of the system.

Since a new set of data is being used, correlation coefficients and lags must be re-computed for each input parameter, using the same methods. They are shown in Table 5.

Input Variable	Correlation Coefficient	Time Lag
AL12	0.2579	0
AL13	-0.1155	80
AL14	0.0510	39
BGVEL	-0.2256	29
BGVOL	-0.0727	27
CHRGDTIM	0.0721	9
CMOIST7	0.0837	0
CMOIST9	0.0754	37
COKTOTAL	-0.2046	84
CWINDEX	0.0886	15
DOLTOTAL	0.3081	0
DP	0.1042	84
ECO	0.1164	68
EH2	0.0830	0
FI59	0.0914	84
FIC46	0.2906	0
G1HF	-0.1109	61
G2HF	-0.1496	68
INJ_ACT	-0.1294	44
LSTOTAL	0.0886	84
NCKTOTAL	0.0645	84
O2ENR_PV	0.0508	84
OREBYCOK	-0.0796	82
ORETOTAL	-0.0681	84
PERM_K	0.2493	35
PI89	0.1102	84
PI92	0.1114	84
PIC33	0.0961	84
QZTOTAL	-0.0933	84
RAFT_TH	-0.3665	0
SINTOTAL	-0.0707	64
SINTPERC	0.0536	36
THC_R	N/A	0
TIC91	-0.1661	84
TUYVEL	-0.1719	32
TY21_PV	0.8339	0

Table 4: New Correlation and Lags

Because a moving window is used, each unit of lag represents a 5-minute interval.

Therefore, 84 corresponds to a delay of 7 hours.

5.12 Results Using 35 Variables (Without the Previous Hot Metal Temperature)

An extensive number of models were run using the new data set. Each neural net algorithm was paired with different learning algorithms, different learning rates, and different number of hidden nodes. While the different learning rates produced dramatically different results, the number of hidden nodes and the learning algorithms did not have much impact. The results of running Feed-forward on the new data set are shown in Figure 7. The results do not seem to have improved much. The lack of data points, which was a problem before the implementation of moving windows now does not seem to be the biggest obstacle to better models.

5.13 Results Using 11 Variables (Without the Previous Hot Metal Temperature)

Based on the experience of TISCO personnel regarding which input parameters has the greatest influence and the correlation coefficients obtained using empirically methods, 11 input variables were chosen among the original 35 variables, with the purpose of reducing noises created by the other 24 less-correlated variables. The new *state* used in the modeling is substantially smaller.

The 11 chosen variables are CHRGDTIM, COKTOTAL, ECO, EH2, STEAM, G1HF, G2HF, INJ_ACT, O2ENR_PV, OREBYCOK, and TIC91.
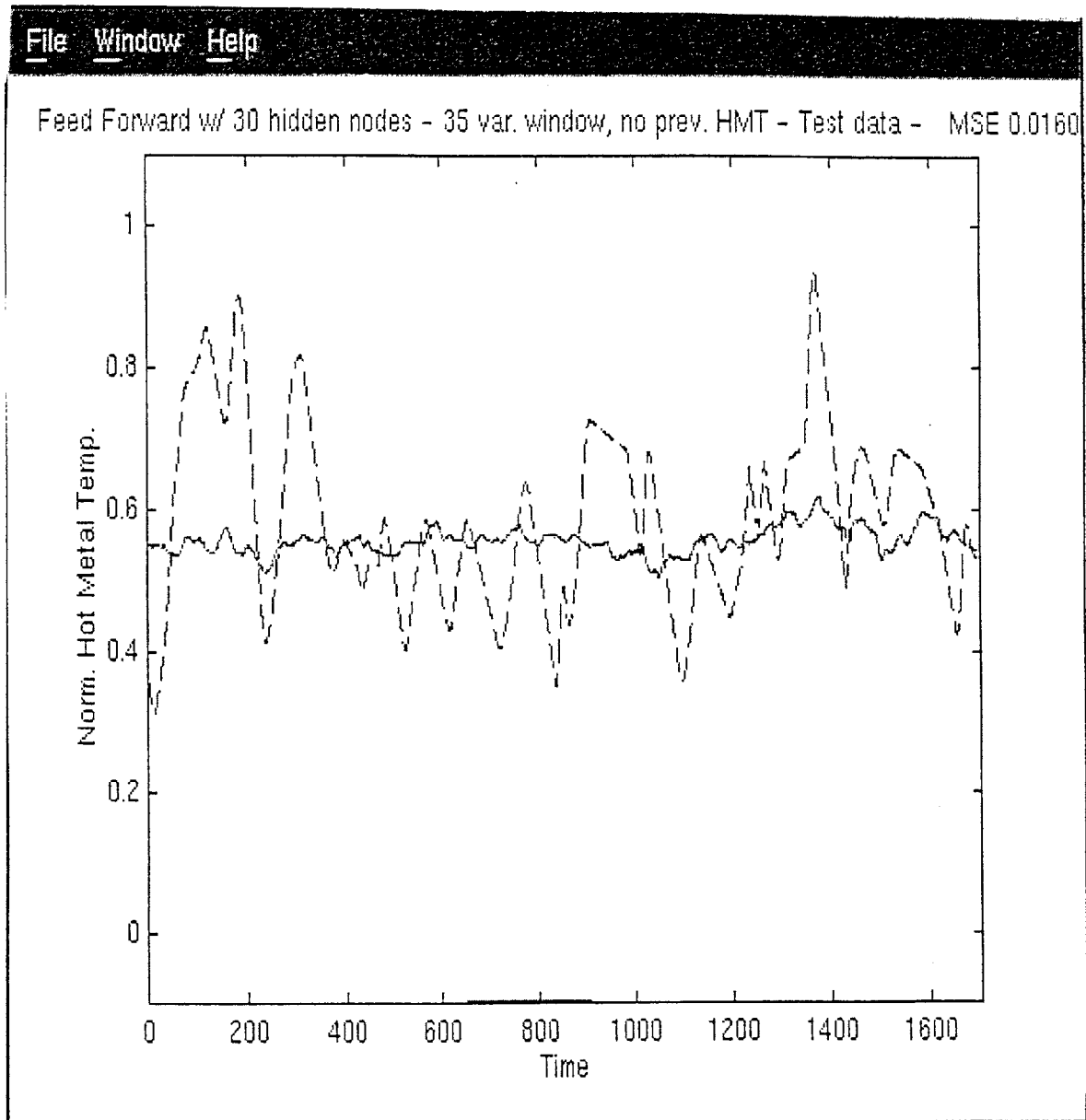
Figure 7: Results from Running Feed-Forward Back-propagation

As before, linear regression models were run in order to set a benchmark. Our neural network models are expected to be at least as good as or better than the linear regression models. The results of the linear regression are shown in Figure 8. The MSE for the testing data is 0.0292. Again, it is not a substantial improvement.

With the knowledge of the benchmark results, An extensive set of neural networks were run. The best results were Feed-forward neural network algorithm with standard Back-propagation, using 10 hidden units in one single hidden layer. The results are shown in figure 9. The MSE is 0.0164. The results from running the Feed-Forward model are better than our benchmark. It still needs improvement.

5.14 Using the Previous Hot Metal Temperature as an Input

In addition to the moving window, another input parameter, the actual hot metal temperature value of the previous time was added [4]. Because of its time-dependent nature, when predicting the output of a time-series data, data points cannot be looked at separately. Each output affects the successive outputs. The next output of the model is dependent on the actual output of the previous time index. Since the current output temperature trails the previous temperature well, the new models were expected to perform better.
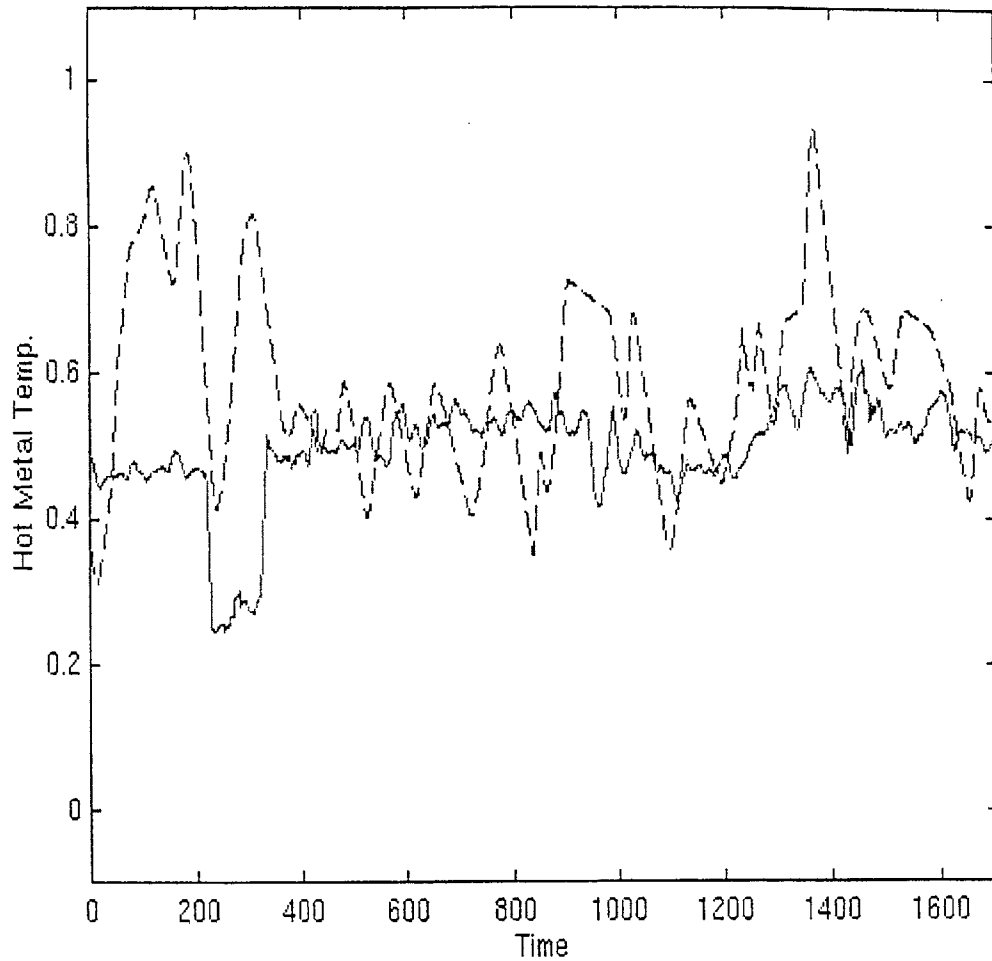
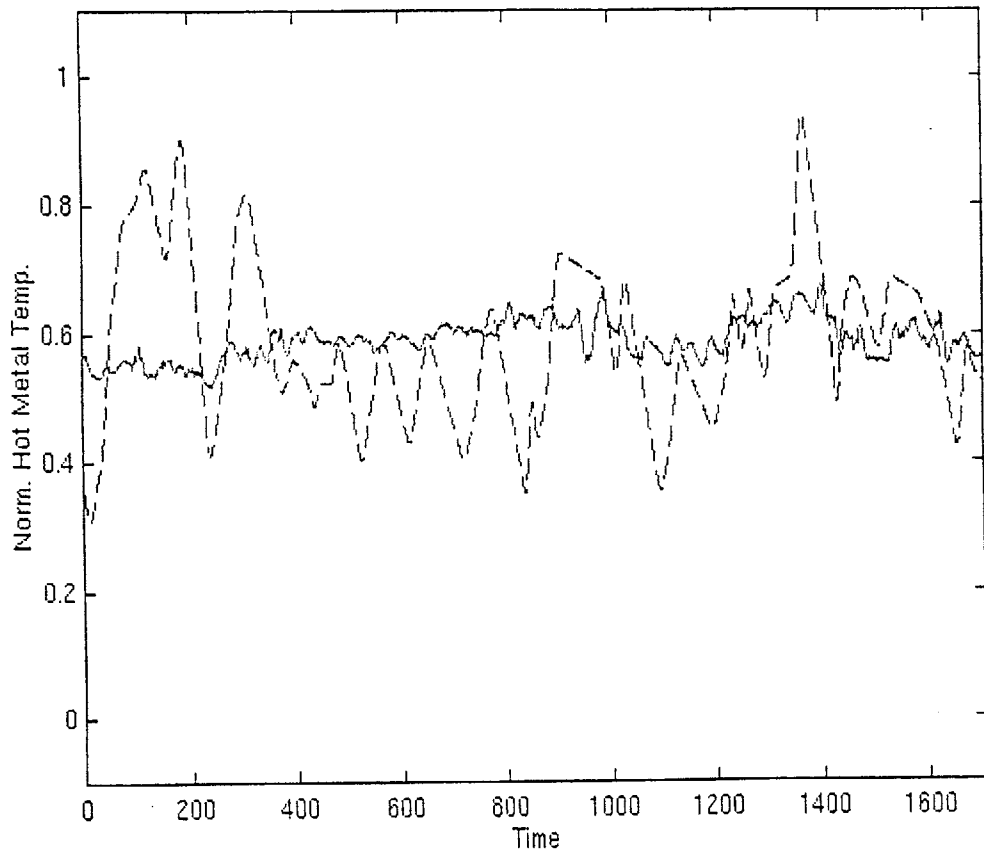*Figure 8: Linear Regression of 11 Variables without Previous Hot Metal temperature*

Figure 9: Results from Running Feed-Forward on 11 Variables Without Hot Metal Temperature

Just as expected, the results from using the previous hot metal temperature improved tremendously. Figure 10 shows the results. The model used was Feed-Forward, with Backpropagation. This has produced the best results so far. The MSE is 0.0083.

In the case where the previous hot metal temperature value was used, the results are much better. Here both the neural net and linear models (Figure 11) seem to be converging on the same solution. This is good news because it means that the models are capturing all of the useful information in the data, while avoiding over-fitting to the noise. Most of the increase seems to be directly attributable to using previous hot metal temperature, indicating that interactions between previous hot metal temperature and other variables may not be important.

5.15 Issues Related to Using Previous Hot Metal Temperature

Modeling the blast furnace using neural networks serves three purposes:

1) Prediction - During the production process, the model should give the most accurate prediction possible of the next hot metal temperature value. Using previous values of hot metal temperature is compatible with this goal, and in fact has proven to be a great benefit.

2) Simulation - In offline situations, the model might be used to experiment with different control settings. Thus, different combinations or procedures for controlling the process could be simulated on the model to see what kind of results might be expected if tried in real life. Unfortunately, this goal is incompatible with using previous hot metal temperature because the model would be unstable. As the model iterated through time, it
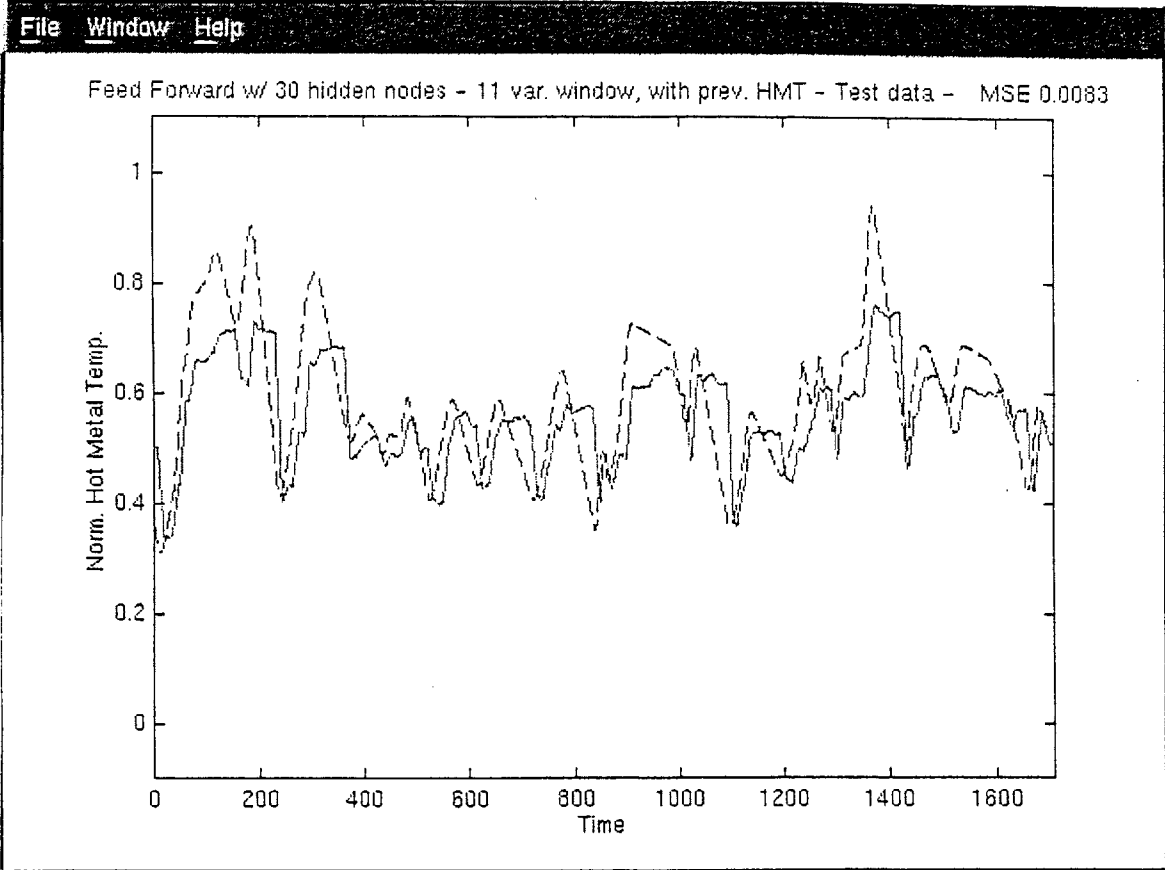
Figure 10: Results From Running Feed-Forward on 11 Variables with Previous Hot Metal Temperature
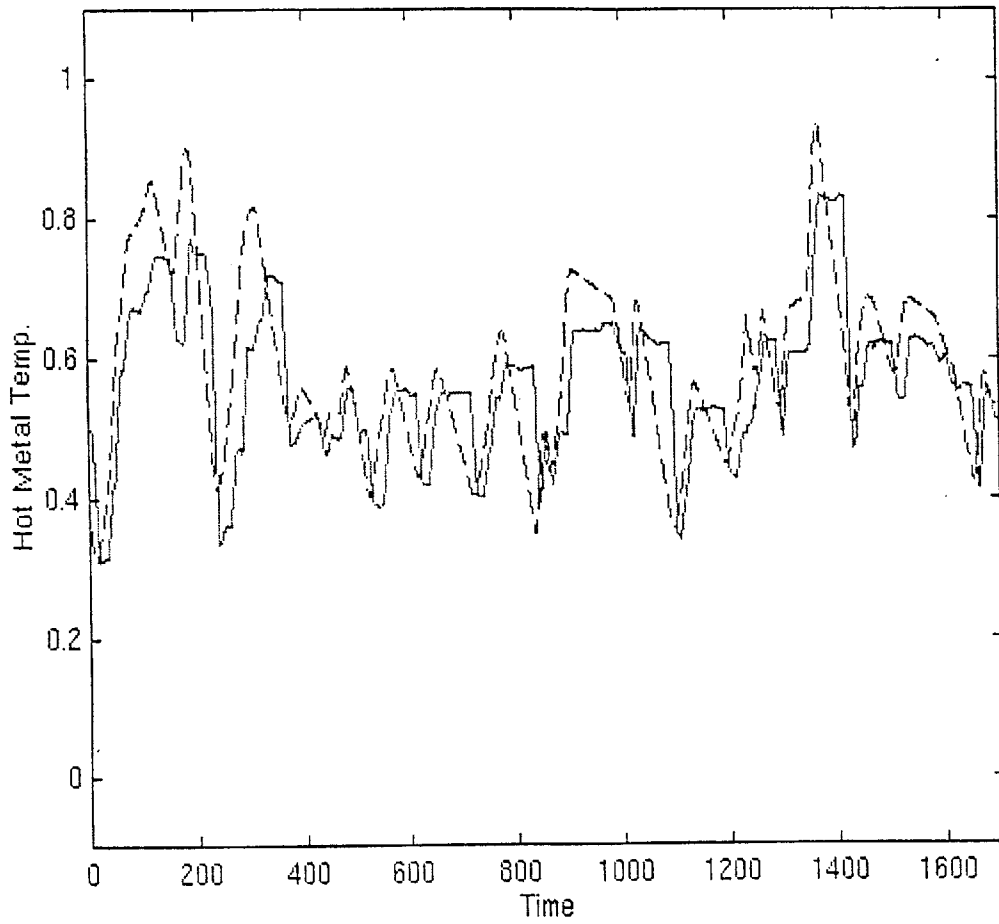
Figure 11: Results from Running Feed-Forward Model on 11 Variables with Previous Hot Metal

Temperature.

would have to use its own prediction for hot metal temperature as the input to the next step. Any error in that prediction would compound in the next iteration, and after any significant number of steps the model would be out of touch with reality, because it lacks the feedback that it would get in real life from the actual values of hot metal temperature it receives.

3) Control - The model could be used to help automate the control of pig-iron production. This also is unaffected by using previous values of hot metal temperature.

Prediction and simulation/control must be treated separately. Then benefit of hot metal temperature as an input variable can be enjoyed without corrupting the simulation and control efforts.

6.0 Conclusion

In general, the most difficult problems encountered in the neural network based modeling of the blast furnace were the lack of data and the correct *state* (i.e. set of input parameters) of the blast furnace to model. Three different sets of parameters were modeled separately: the set of all 35 input variables, the set of 11 of the highest-correlated variables, and the set of 11 of the highest-correlated variables plus the use of the previous hot metal temperature as an additional input variable. Significantly higher accuracy was achieved when one used the previous value of hot metal temperautre in the training phase; in such a scenario, the number of variables used (either 35 or 11 variables) had little impact. Overall, the latter scenario provided the best model of the blast furnace.

The problem of too few and relatively irregular timing of data points was addressed in multiple ways. Linearization was used to overcome the erratic frequency with which the hot metal temperature was measured; the concept of moving windows was used to boost the number of data points reduced by the use of linearization. Both methods are good at resolving the problems they are intended for, but inherently, they distort the way the parameters are represented. This is a side effect which one has to cope with, in situations involving missing and/or infrequent data.

In the process of modeling, different settings of neural algorithms were experimented with, such as using different number of nodes in the hidden layer, using different learning algorithms and using different learning rates. The variations on the number of nodes and the different algorithms did not produce very different results, indicating that these factors

are not important. Various learning rates, from 0.01 to 9, were tried. The lower learning rates worked better, since a neural network with a high learning rate tends to over-fit.

Even though in general, Time-delay and Jordan are more powerful networks because of their ability to capture time-dependencies in the data set, in this case, they did not outperform simple Feed-forward neural networks in this case. Time-dependencies of the data set has already been explicitly defined and the effect of that compensated. This helps to explain why Feed-forward nets performed just as well as Time-delay and Jordan in the modeling of the blast furnace.

The best neural network algorithm employs a simple Feed-Forward net with Back-propagation and a learning rate of 0.02. The MSE is 0.0083.

Future research on this project could focus on the following issues:

1) Prediction of other important output variables. The modeling of hot metal temperature can be extended to analyze variables like silicon content and slag basicity.

2) Separating the effects of operator's reaction from the blast furnace process. Empirically, some input variables have different behavior than expected. When an operator sees a input variable skewing in one way, he/she immediately reacts to compensate. Hence, the data set captures the blast furnace as well as the operator's reaction. The latter needs to be separated or handled in a different model.

3) Enhancing the quality of prediction further by extending the idea of using previous

    hot metal temperature. Instead of using the hot metal temperature from the previous

    time period, hot metal temperature from the previous two time periods could be used.

The blast furnace environment is a complex one, and multiple neural network models will

need to be created to handle different parameters and situations.

7.0 Reference and Bibliography

[1] Source: Federation Francaise de l'Acier, site: http://www.ffa.fr/wisteel.htm

[2] Mallick S.K. Basu, "Automation in Blast Furnace: Scope and Recent Trends". Tata Search 1998. Pp 89-96.

[3] Ullmann's Encyclopedia of Industrial Chemistry, 5th complete revised edition. VCH 1985. Vol. A 14. P 517-540

[4] Abhay B. Bulsari, Henrik Saxen, and Bjorn Saxen, "Time-Series Prediction of Silicon In Pig Iron Using Neural Networks". Pp 117-120

[5] Singh, H., Sridhar, N.V., and Deo, B. "Artificial neural nets for prediction of silicon content of blast furnace hot metal". Steel Research 67 (1996) No. 12. 117-120.

[6] Osamu Lida, et. al. Application of AI techniques to blast furnace operations, Iron and Steel Engineer, (October 1995) 24-28

[7] Michael J. A. Berry, Gordon Linoff. "Data Mining Techniques". Wiley Computer Publishing. 1997

[8] Zell A., et. al. Stuttgart Neural Netowrk Simulator User Manual, Version 4.1, report No. 6, 1995.

[9] Pandey B.D., et al, "G Blast Furnace at Tata Steel: A Success Story". Tata Search 1997. Pp 4-9.

[10] Source site: http://elmhcx9.elmhurst.edu/~chm/onlcourse/chm110/outlines/steel.htm

[11] V.V. Mahashabde, et. al, "Breakout Detection System at Tata Steel's Slab Caster using Neural Networks". Tata Search. 1999

[12] Hirata, T.; Yamamura, K.; Morimoto, s; Takada, H.; "Blast Furnace Operation System using Neural Networks and Knowledge-base", Proc. 6th Intern. Iron and Steel Congr., Nagoya, Japan, 1990, ISIJ. P.23-27.

[13] Lippmann, R.P. "An introduction to computing with neural nets". IEEE ASSP Magazine, (April 1987) 4-22.

[14] Widrow, B. and A. Lehr. "30 Years of Adaptive Neural Networks: Perceptrons. Madaline, and Backpropagation". IEEE Neural Networks I: Theory & Modeling, (1990) 1415-1442

[15] Jones, W.P. and J. Hoskins, "Back-Propagation: A generalized delta learning rule", Byte, (October 1987) 155-162

[16] Silva, F.M. and L.B. Almeida. "Acceleration Techniques for the Backpropagation Algorithm", in L.B. Almeida and C.J. Wellekens (Eds.) Neural Networks. Lecture Notes in Computer Science 412, 1990, pp. 110-119.

[17] Rumelhart, D.E., G.E. Hinton and R.J. Williams. "Learning representation by back-propagating error". Nature 323 (1986) 533-536

[18] Leonard, J. and M.A. Kramer, "Improvement of the backpropagation algorithm for training neural networks", Compu.chem Engng. 14 (1900) 337-341.

[19] Camargo, F.A., "Learning Algorithms in Neural Networks". Technical report CUCS-062-90. DCC Laboratory. Columbia University, New York. 1990.

[20] Marquardt, D.W., "An algorithm for least-squares estimation of nonlinear parameters". J. SIAM 11 (1963) 431-441.

[21] Levenberg. K., "A Method for Solution of Certain Non-linear Problems in Least-squares", Quart.Appl.Math. 2 (1944) 164-168.

[22] Bulsari, A. and H. Saxen. "Applicability of an artificial neural network as a simulator for a chemical process". Proceedings of the Fifth International Symposium on Computer and Information Sciences, Nevsehir, Turkey, October 1990. 143-151.

[23] Bulsari, A., H. Saxen, "Efficient acquisition and storage of heuristics in a neural network", paper to be presented at IASTED 9th Symp. Applied Informatics. Innsbruck, Austria, February 1991.

[24] Hoskins, J.C. and D.M. Himmelblau, "Artificial neural network models of knowledge representation in chemical engineering", Comput. Chem. Engng. 12 (1988) 881-890.

[25] Venkatasubramanian, V. and K. Chan, "A neural netowrk methodology for process fault diagnosis", AIChE Jornal 35 (1989) 1993-2002.

[26] Venkatasubramanian, V., R, Vaidyanathan and Y. Yamamoto, "Process fault detection and diagnosis using neural networks I: Steady state processes". Comput.chem.Engng 14 (1990) 699-712.

[27] Watanabe, K., I. Matsuura, M. Abe and M. Kubota, "Incipient fault diagnosis of chemical rocess control", Comput. Chem. Engng 14 (1990) 23-27.

[28] Bhat, N. V., P. A. Minderman, Jr., T. McAvoy and N.S. Wang, "Modeling chemical process systems via neural computation", IEEE Control systems magazine. (April 1990) 24-30.

[29] Bhat, N. and T. J. McAvoy, "Use of neural nets for dynamic modeling and control of chemical process systems", Comput. Chem. Engng 14 (1990) 573-582.

[30] Ydstie, B. E., "Forecasting and control using adaptive connectionist netowrks", Comput. Chem. Engng. 14 (1990) 583-599.

[31] Weigend, A.S., B. A. Huberman and D. E. Rumelhart, "Predicting the Future: A Connectionist Approach", IJNS 1 No 3 (1990) 193-209.

[32] Phadke, M.S. and S.M. Wu, "Identification of Multiinput-Multioutput Transfer Function and Noise Model of a Blast Furnace from Closed-Loop Data", IEEE Trans on Automatic Control AC-19 (1974) 944-951.

[33] Chang, T.N., M.W. Wu and J.C. Chiou, "Blast Furnace System Identification by Maximum Likelihood Method", IFAC Automation in Mining Mineral and Metal Processing, Tokyo, Japan, 1986, pp. 229-234.

[34] Nayeb Hashemi, A.A., J.A. Clum and S.M. Wu, "Blast furnace modeling and control by the DDS method", 36th Ironmaking Conf., AIME 1977, pp. 283-297.

[29] Chao, Y.C., C. H. Su and H. P. Huang, "The adaptive autoregressive models for the system dynamics and prediction of blast furnace", Chem. Eng. Commun. 44 (1986) 309-330.

[35] Yan-Jiong, Z., Y. Liangt-Tu and Z. Min, "Adopt three criterions to choose MISO prediction model of blast furnace", IFAC Automation in Mining. Mineral and Metal Processing. Tokyo, Japan, 1986, pp. 241-246.

[36] Unbehauen, H. and K. Diekmann, "Application of MIMO-Identification to a Blast Furnace", IFAC Identification and System Parameter Estimation, Washington, USA, 1982, pp. 235-240.

[37] Uusi-Honko, H., "Prediction of hot metal silicon in blast furnace by multivariate time series modeling", Technical report 88-90-A, Process Design Lab. Abo Akademi, Abo, Finland 1988.

[38] Box, G.E.P. and G.M. Jenkins, Time series analysis, forecasting and control. Holden-Day, Oakland, USA, 1976.

[39] Gustafsson, T., MSc Thesis (in preparation), Heat Eng Lab., Abo Akademi, Abo. Finland 1991.

[40] Chauvin, Y., "Generalization Performance of Overtrained Back-Propagation Networks", in L.B. Almeida and C.J. Wellekens(Eds.) "Neural Networks", Lecture Notes in Computer Science 412 (1990) 46-55. Springer-Verlag.

[41] Hanson, S.J. and L. Y. Pratt, "Comparing Biases for Minimal Network Construction with Back-Propagation", in D. Touretzky (Ed.) Neural Information Processing Systems 1 (1989) 177-185, Morgan Kaufmann Publishers, Palo Alto.

[42] Niwa Y., Sumigama T., Maki A., Yamaguchi A., Inoue H., Tamura T., Proc. 6th Intern. Iron & Steel Congr., Nagoya, Japan. 1990. Vol. 2, pp. 527-534.

[43] Singh H., Deo B., Iron & Steelmaker 22 (1995) No. 10, pp. 85-92.

[44] Batra N.K., "Silica Reduction Studies in a Blast Furnace", Mission Management Board, R & D Center for Iron and Steel, Steel Authority of India, Ranchi, India, Sept. 1992.

[45] Tsuchiya N., Tokuda M., Ohtani M., Met. Trans. 7B (1976), pp. 315-320.

[46] Venkatadri A.S., Bell H.B., J. Iron and Steel Inst. 207 (1969), pp. 1110-1113.

[47] Croft V., Iron mak. Steelmak. 7 (1980), pp. 116-122.

[48] Batra N.K., Bhaduri P.S., Ironmak Steelmak. 17 (1990) No 6, pp. 389-393.

[49] Biswas A.K., Principles of Blast Furnace Ironmaking, SBA Publications, Calcutta, 1984, pp. 309.