PROGRAMMING AND SIMULATION OF DENSIFICATION OF ZTA NANO-COMPOSITE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Ceramic Engineering

By

K. ROHAN

Department of Ceramic Engineering

National Institute of Technology

Rourkela

2007

PROGRAMMING AND SIMULATION OF DENSIFICATION OF ZTA NANO-COMPOSITE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Ceramic Engineering

By

K. ROHAN

Under the Guidance of

Dr. D. Sarkar

Department of Ceramic Engineering

National Institute of Technology

Rourkela

2007

National Institute Of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "**Programming and Simulation of Densification of ZTA nano-composite**" submitted by **Sri K. Rohan** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in **Ceramic Engineering** at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by her under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Dr. Debasish Sarkar
Dept. of Ceramic Engineering
National Institute of Technology
Rourkela -769008

ACKNOWLEDGEMENT

I express my sincere gratitude to Dr. D. Sarkar, Lecturer (SS), Department of Ceramic Engineering, N. I. T. Rourkela who has been a constant source of inspiration, encouragement and motivation and also for the constant advice through discussion for the successful progress of this project work. I thank him for the help he extended in doing my project work.

K. ROHAN
10308015

CONTENTS

ABSTRACT

Alumina is an important structural ceramics having wide application potential. However, its use in real life systems is limited due to its low fracture toughness resulting in catastrophic failure. Among the several measures that are being taken to overcome the draw- back, the addition of zirconia (platelet, particle, fiber) in the alumina matrix remarkably improves the toughness on account of stress induced transformation toughening.

Al_2O_3–ZrO_2 composite powder containing 15 mol% ZrO_2 was prepared by sol-gel route using Aluminum nitrate (E-Merck, India) and zirconium oxychloride (E-Merck, India) as precursor. The sol–gel derived powder was properly characterized by X-ray Diffraction study (XRD). The analysis reveal the gel contained pseudoboehmite and amorphous $Zr(OH)_4$ was decomposed in three and two stages respectively. The phase transformation of alumina during calcination followed the sequence of pseudoboehmite → bayerite → boehmite → γ-Al_2O_3 → θ-Al_2O_3 → α-Al_2O_3, while that of ZrO_2 follows amorphous ZrO_2 → t-ZrO_2 → (t + m) ZrO_2. Densification behavior of the sol gel precursor was studied through compaction of the powder at uniaxial press and subsequently sintered at muffle furnace.

The relative density of the uniaxial pressed specimen depends on several factors; such as pressure, temperature, sintering profile and sample position during firing. An Artificial Neural Network (ANN) can predict the characteristics densification features of sintered specimens at any intermediate processing parameters. To evaluate this, simulation of relative density of ZTA Nano-composite has been carried out through MATLAB program. The simulation was focused on stepwise single layer to multilayered ANN theory. Herein, the project work is emphasized on the prediction of change in densification behavior with respect to change in processing parameters.

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1. Introduction

Alumina is one of the most widely used structural ceramics. Several properties of alumina are controlled by its microstructure and its matrix stability. Moreover, additive interactions can modify and achieve tailor made properties of alumina ceramics. Zirconia is one such additive which can increase the strength and toughness of alumina matrix either by stress-induced transformation toughening or by microcrack toughening. The extent of stress-induced transformation toughening depends on the dispersion of tetragonal zirconia (t-ZrO_2) in alumina matrix, its volume fraction and transformability. On the other hand, a uniform distribution of ZrO_2 in ceramic matrix is an important factor for optimization of microcrack nucleation-induced toughening. The uniform dispersion of zirconia particles in the alumina matrix can be controlled by homogeneous powder synthesis techniques. A series of powder processing techniques have been investigated to synthesize homogenous powder mixture and amongst them the precipitation and the sol–gel methods are the easy and commercialized chemical synthesis routes for producing zirconia doped nanoparticles.

Sol–gel processing technology has been developed for the fabrication of a high quality ceramic-based composite. For complex oxides, it achieves ultra-homogeneous mixing of the several components on a molecular scale, reduces sintering temperature and hence develops a fine-grained microstructure. Thus, the liquid precursor technology offers processing advantages and gives flexibility in tailoring the composite chemistry to obtain the desired properties. Moreover, the processing conditions, composition, retention of the *t*-phase of zirconia and the calcination temperature strongly influence the morphology of the powder and sintering behavior. The arguments and observations responsible for such modifications are changeable with the phase composition, crystallinity, crystallite size and pore morphology, specific surface area and subsequent shrinkage.

The synthesis of ZrO_2 dispersed Al_2O_3 precursor powders from multiphase hydrogel is a complex subject, which depends on the configuration of hydroxide of Al and Zr and its polymerization. The chemistry and polymerization of aluminum oxide and hydroxide depend on acid/base properties. The proton activity is significantly affected by the oxygen coordination surrounding Al_3^+. In the deprotonated form of the aluminum hydroxide in water $[Al(OH)_4]^-$, Al atom is tetrahedrally coordinated with oxygen atoms, whereas all the protonated cationic species in the series $Al(OH)_2^+(aq)$ to Al^{3+} (aq) are six-coordinated. This amphoteric nature of $Al(OH)_3$ in water is the cause for the stability of hydrogel.

The above discussion summarizes that the phase evolution sequence during crystallization of alumina–zirconia from its hydrated precursor depend among other factors on the nature and type of bonding present in the hydroxides of Al and Zr. Thus the decomposition behavior phase evaluation sequence and the nature of bond type on the phase evaluation of both Al and Zr hydroxide are studied by thermal analysis, X-ray Diffraction study (XRD). Densification of the zirconia particles on alumina is depending upon the processing parameters.

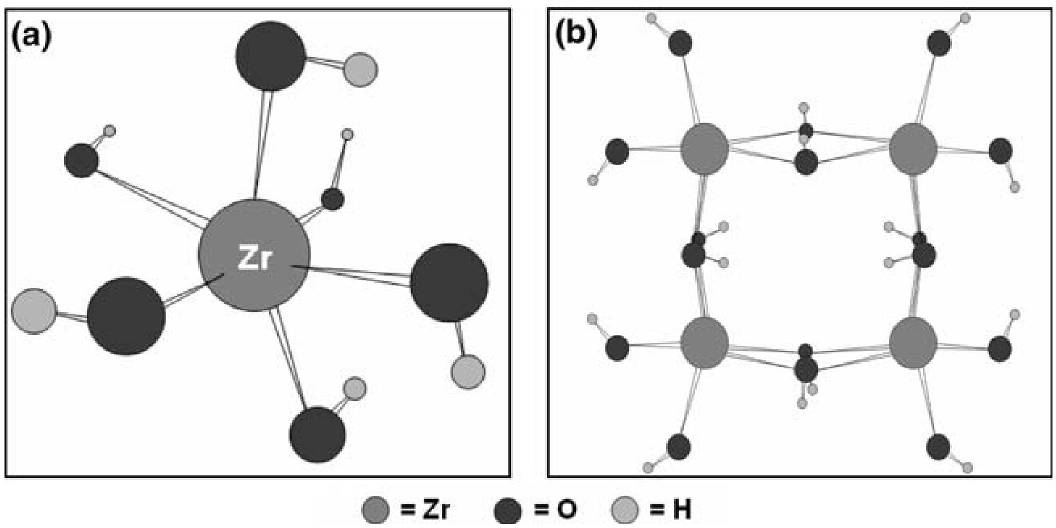


Fig 1.1. Cluster structure of $[Zr\ (OH)_6]^-_2$ (a) ordered structure and (b) polycrystalline precursors

1.1 ARTIFICIAL NEURAL NETWORKS

1.1.1 LET'S START WITH SOME BIOLOGY

Nerve cells in the brain are called neurons, there are an estimated 10^{10} to the power (10^{13}) neurons in the human brain, each neuron can make contact with up to several thousand other neurons. Neurons are the unit in the brain to process information.

1.1.2 NEURONS LOOK LIKE

A neuron consists of a cell body, with various extensions from it. Most of these are branches called dendrites. There is one much longer process (possibly also branching) called the axon the dashed line shows the axon hillock, where transmission of signals starts
The following diagram illustrates this.

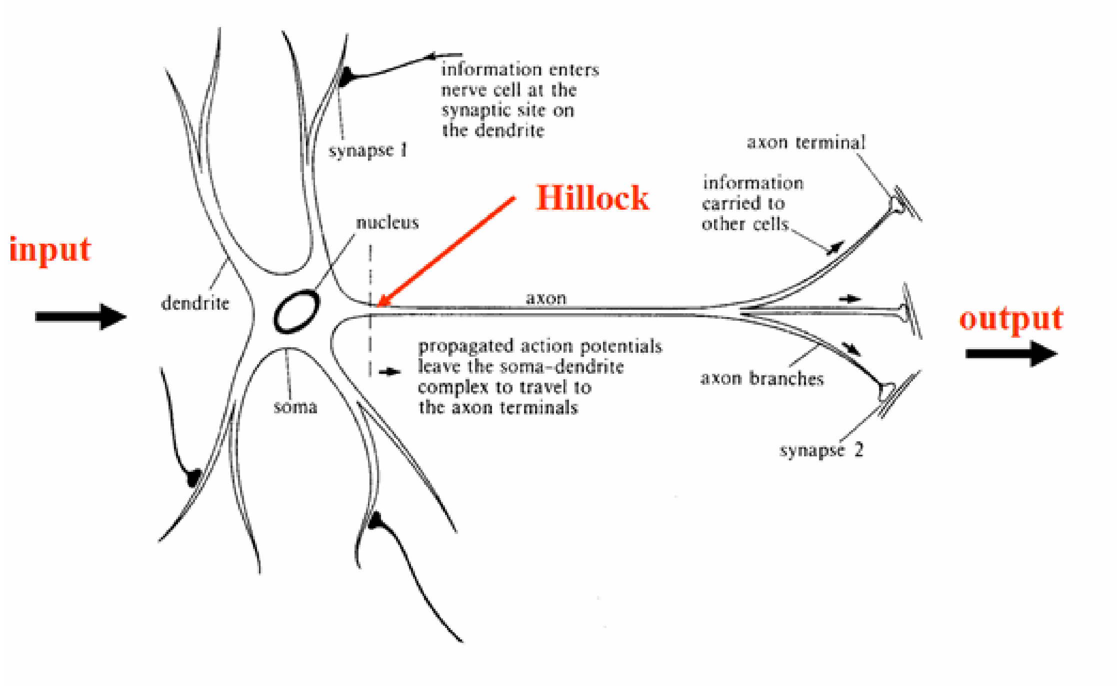


Fig 1. 2. The Neuron

The boundary of the neuron is known as the cell membrane. There will be a voltage difference (the membrane potential) between the inside and outside of the membrane.
If input is large enough, an action potential is then generated. The action potential (neuronal spike) then travels down the axon, away from the cell body.

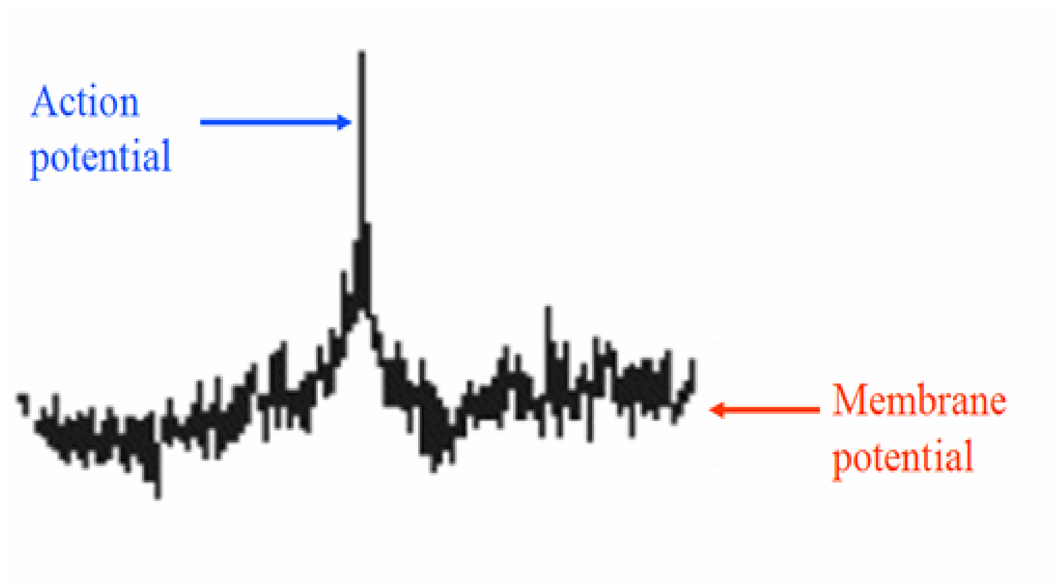


Fig1. 3. Neuron Spiking

1.1.3 SYNAPSES

The connections between one neuron and another are called synapses. Information always leaves a neuron via its axon (see Figure 2 above), and is then transmitted across a synapse to the receiving neuron.

1.1.4 NEURON FIRING

Neurons only fire when input is bigger than some threshold. It should however be noted that firing doesn't get bigger as the stimulus increases, it’s an all or nothing arrangement. Spikes (signals) are important, since other neurons receive them. Neurons communicate with spikes. The information sent is coded by spikes.

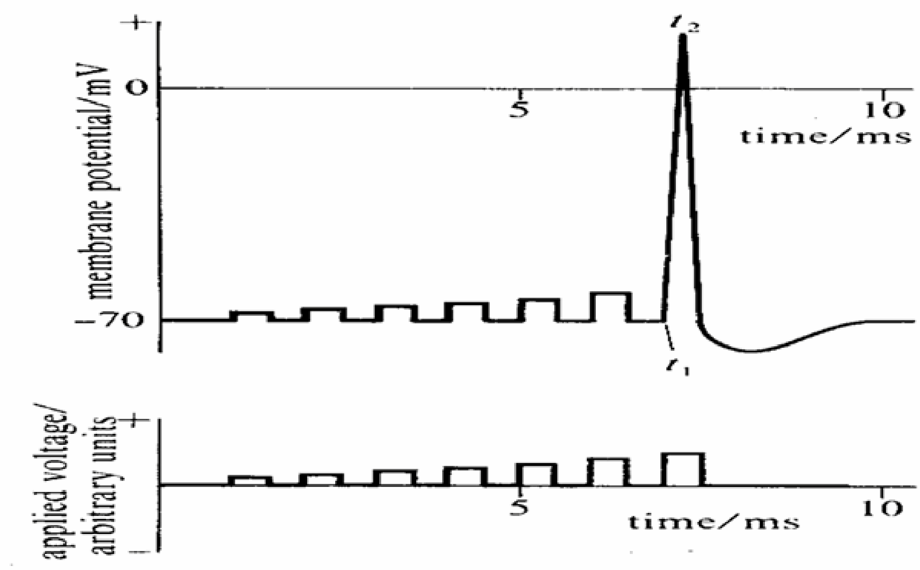


Fig 1.4. Neuron Firing

1.1.5 THE INPUT TO A NEURON

Synapses can be excitatory or inhibitory. Spikes (signals) arriving at an excitatory synapse tend to cause the receiving neuron to fire. Spikes (signals) arriving at an inhibitory synapse tend to inhibit the receiving neuron from firing. The cell body and synapses essentially compute (by a complicated chemical/electrical process) the difference between the incoming excitatory and inhibitory inputs (spatial and temporal summation).

When this difference is large enough (as compared to the neuron's threshold) then the neuron will fire. Roughly speaking, the faster excitatory spikes arrive at its synapses the faster it will fire, similarly for inhibitory spikes.

1.1.6 ARTIFICIAL NEURAL NETWORKS

Suppose that we have a firing rate at each neuron, and also suppose that a neuron connects with m other neurons and so receives m-many inputs x1, x2… … xm, we could imagine this configuration looking something like

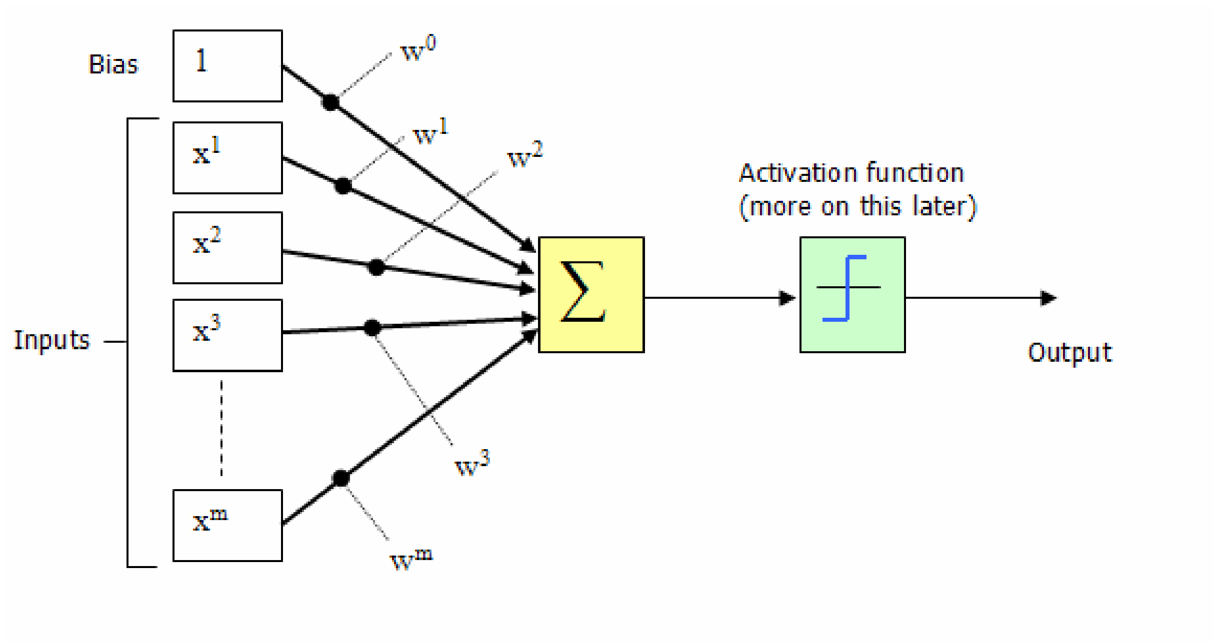


Fig 1.5 Artificial Neuron Configurations, with bias as additional input

This configuration is actually called a **Perceptron.** The perceptron (an invention of Rosenblatt [1962]), was one of the earliest neural network models. A perceptron models a neuron by taking a weighted sum of inputs and sending the output 1, if the sum is greater than some adjustable threshold value (otherwise it sends 0, this is the all or nothing spiking described in the biology, neuron firing section above) also called an activation function.

The inputs (x_1, x_2, x_3...x_m) and connection weights (w_1, w_2, w_3...w_m) in figure 5 are typically real values, both positive (+) and negative (-). If the feature of some x_i tends to cause the perceptron to fire, the weight w_i will be positive; if the feature x_i inhibits the perceptron, the weight w_i will be negative.

The perceptron itself consists of weights, the summation processor, and an activation function, and an adjustable threshold processor (called bias here after). For convenience the normal practice is to treat the bias, as just another input. The following diagram illustrates the revised configuration.

The bias can be thought of as the propensity (a tendency towards a particular way of behaving) of the perceptron to fire irrespective of its inputs. The perceptron configuration network shown in Figure 5 fires if the weighted sum > 0, or if you're into maths type explanations

$$\boldsymbol{\Sigma_{(i=1,m)}\ \text{bias}+ (w^i\, x^i)}$$

1.1.7 ACTIVATION FUNCTION

The activation usually uses one of the following functions.

1.1.7 (A) SIGMOID FUNCTION

The stronger the input the faster the neuron fires (the higher the firing rates). The sigmoid is also very useful in multi layer networks, as the sigmoid curve allows for differentiation (which is required in Back Propagation training of multi layer networks).

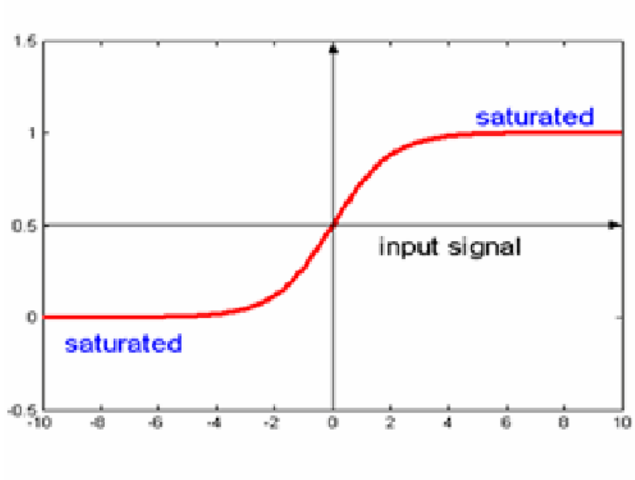


Fig 1.6 Sigmoid Function

Or if you're into maths type explanations

$$\mathbf{f(x) = 1 / (1+ e^{-\beta x})}$$

1.1.8 LEARNING

A FOREWORD ON LEARNING

Before we carry on to talk about perceptron learning lets consider a real world example:

How do you teach a child to recognize what a chair is? You show him examples telling him "This is a chair; that one is not a chair" until the child learns the concept of what a chair is. In this stage, the child can look at the examples we have shown him and answer correctly to the question "Is this object a chair?"

Furthermore, if we show to the child new objects, that he didn't see before, we could expect him to recognize correctly whether the new object is a chair or not, providing that we've given him enough positive and negative examples.

This is exactly the idea behind the perceptron.

1.1.8(A) LEARNING IN PERCEPTRONS

Learning is the process of modifying the weights and the bias. A perceptron computes a binary function of its input. Whatever a perceptron can compute it can learn to compute.

"The perceptron is a program that learns concepts, i.e. it can learn to respond with true (1) or false (0) for inputs we present to it, by repeatedly "studying" examples presented to it. The Perceptron is a single layer neural network whose weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector. The training technique used is called the perceptron learning rule. The perceptron generated great interest due to its ability to generalize from its training vectors and work with randomly distributed connections. Perceptrons are especially suited for simple problems in pattern classification. "

1.1.8(B)THE LEARNING RULE

The perceptron is trained to respond to each input vector with a corresponding target output of either 0 or 1. The learning rule has been proven to converge on a solution in finite time if a solution exists.

The learning rule can be summarized in the following two equations:

$$\mathbf{b = b + [T - A]} \qquad \mathbf{(1)}$$

For all inputs i:

$$\mathbf{W\ (i) = W\ (i) + [T - A] * P\ (i)} \qquad \mathbf{(2)}$$

Where W is the vector of weights, P is the input vector presented to the network, T is the correct result that the neuron should have shown, A is the actual output of the neuron, and b is the bias.

1.1.9 TRAINING

Vectors from a training set are presented to the network one after another. If the network's output is correct, no change is made. Otherwise, the weights and biases are updated using the perceptron learning rule (as shown above). When each epoch (An entire pass through all of the input training vectors is called an epoch) of the training set has occurred without error, training is complete.

At this time any input training vector may be presented to the network and it will respond with the correct output vector. If a vector P not in the training set is presented to the network, the network will tend to exhibit generalization by responding with an output similar to target vectors for input vectors close to the previously unseen input vector P.

So what can we use do with neural networks. Well if we are going to stick to using a single layer neural network, the tasks that can be achieved are different from those that can be achieved by multi layer neural networks. As this article is mainly geared towards dealing with single layer networks, let's discuss those further:

1.1.10 (A) SINGLE LAYER NEURAL NETWORKS

Single layer neural networks (perceptron networks) are networks in which the output unit is independent of the others, each weight effects only one output. Using perceptron networks it is possible to achieve linear separability functions like the diagrams shown below (assuming we have a network with 2 inputs and 1 output)

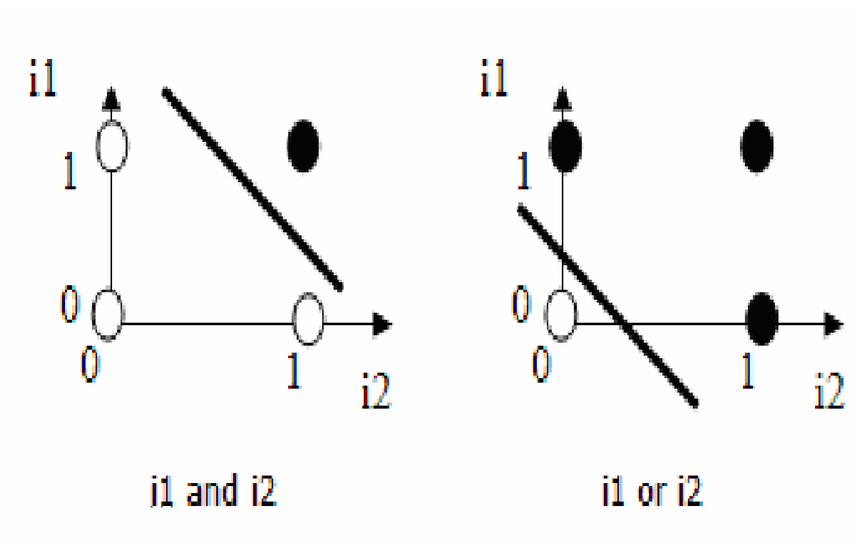


Fig 1.7 Input and Output

So that's a simple example of what we could do with one perceptron (single neuron essentially), but what if we were to chain several perceptrons together, we can imagine that we could build some quite complex functionality, basically we would be constructing the equivalent of an electronic circuit.

Perceptron networks do however, have limitations. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly. The most famous example of the perceptron's inability to solve problems with linearly nonseparable vectors is the Boolean XOR problem.

1.1.10(B) MULTI LAYER NEURAL NETWORKS

With multi layer neural networks we can solve non linear separable problems such as the XOR problem mentioned above, which is a not achievable using single layer (perceptron) network. The next part of this article series will show how to do these using multi layer neural networks, using the back propagation training method.

1.1.11 THE NEW STUFF (MORE LAYERS)

In the summary at the top, the problem we are trying to solve was how to use a multi-layer neural network to solve the XOR logic problem. So how is this done? Well it's really an incremental build on what the article1 already discussed. So let’s march on. Remember with a single layer (perceptron) we can’t actually achieve the XOR functionality, as its not linearly separable. But with a multi-layer network, this is achievable.

1.1.11(A) THE NEW NETWORK LOOKS LIKE

The new network that will solve the XOR problem will look similar to a single layer network. We are still dealing with inputs / weights / outputs. What is new is the addition of the hidden layer.

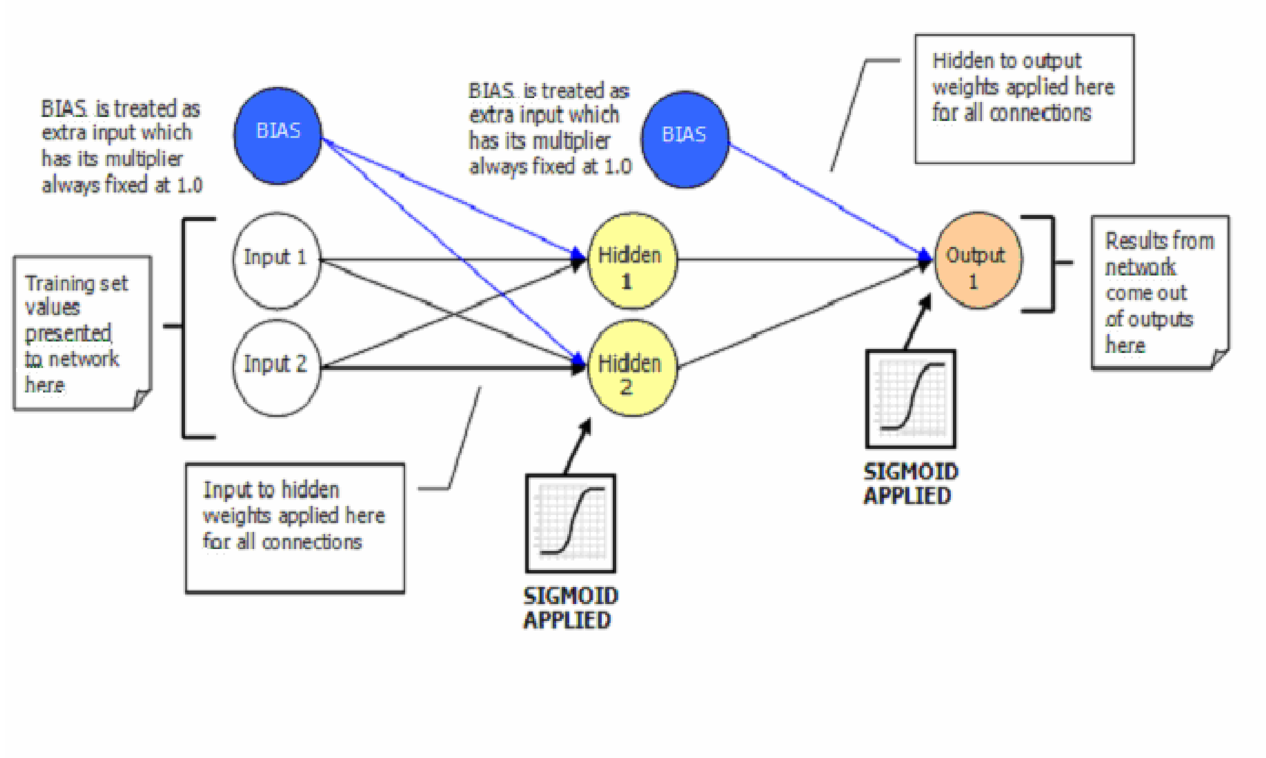


Fig 1.8 New Network looks like a single layer network

As already explained above there is one input layer, one hidden layer and one output layer.
It is by using the inputs and weights that we are able to work out the activation for a given node. This is easily achieved for the hidden layer as it has direct links to the actual input layer. The output layer however knows nothing about the input layer as it is not directly connected to it. So to work out the activation for an output node we need to make use of the output from the hidden layer nodes, which are used as inputs to the output layer nodes.
This entire process described above can be thought of as a pass forward from one layer to the next. This still works like it did with a single layer network; the activation for any given node is still worked out as follows:

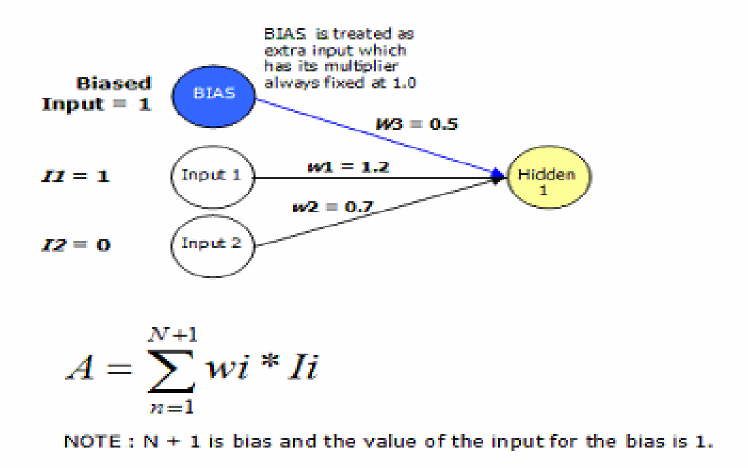


FIG 1.9 Single Layer Network

Where (w_i is the weight (i), and Ii is the input (i) value)

You see it the same old stuff, no demons, smoke or magic here. Its stuff we've already covered. So that's how the network looks/works.

1.1.12 TYPES OF LEARNING

There are essentially 2 types of learning that may be applied, to a Neural Network, which is "Reinforcement" and "Supervised"

1.1.12 (A) REINFORCEMENT

In Reinforcement learning, during training a set of inputs is presented to the Neural Network, the Output is 0.75, when the target was expecting 1.0. The error (1.0 - 0.75) is used for training (‘wrong by 0.25’). What if there are 2 outputs then the total error is summed to give a single number (typically sum of squared errors). e.g. “your total error on all outputs is 1.76.”

Note that this just tells you how wrong you were, not in which direction you were wrong.

Using this method we may never get a result, or could be hunt the needle.

NOTE: Part 3 of this series will be using a GA to train a Neural Network, which is Reinforcement learning. The GA simply does what a GA does, and all the normal GA phases to select weights for the Neural Network. There is no back propagation of values. The Neural Network is just good or just bad. As one can imagine this process takes a lot more steps to get to the same result.

1.1.12(b) Supervised

In Supervised Learning the Neural Network is given more information. Not just ‘how wrong’ it was, but ‘in what direction it was wrong' like ‘Hunt the needle’ but where you are told ‘North a bit’ ‘West a bit’.

So you get, and use, far more information in Supervised Learning, and this is the normal form of Neural Network learning algorithm. Back Propagation (what this article uses, s Supervised Learning)

1.1.13 LEARNING ALGORITHM

In brief to train a multi-layer Neural Network, the following steps are carried out:

- Start off with random weights (and biases) in the Neural Network
- Try one or more members of the training set, see how badly the output(s) are compared to what they should be (compared to the target output(s))
- Jiggle weights a bit, aimed at getting improvement on outputs
- Now try with a new lot of the training set, or repeat again, jiggling weights each time
- Keep repeating until you get quite accurate outputs

This is what this article submission uses to solve the XOR problem. This is also called "Back Propagation" (normally called BP or BackProp)

Backprop allows you to use this error at output, to adjust the weights arriving at the output layer, but then also allows you to calculate the effective error 1 layer back, and use this to adjust the weights arriving there, and so on, back-propagating errors through any number of layers.

The trick is the use of a sigmoid as the non-linear transfer function (which was covered in article1. The sigmoid is used as it offers the ability to apply differentiation techniques.

$$y = g(x) = \frac{1}{1+e^{-x}}$$

$$\frac{dg}{dx} = g'(x) = g(x)(1-g(x))$$

This in context of the article can be written as

delta_outputs[i] = outputs[i] * (1.0 - outputs[i]) * (targets[i] - outputs[i])

It is by using this calculation that the weight changes can be applied back through the network.

Chapter 2

EXPERIMENTAL PROCEDURE

2. EXPERIMENTAL PROCEDURE

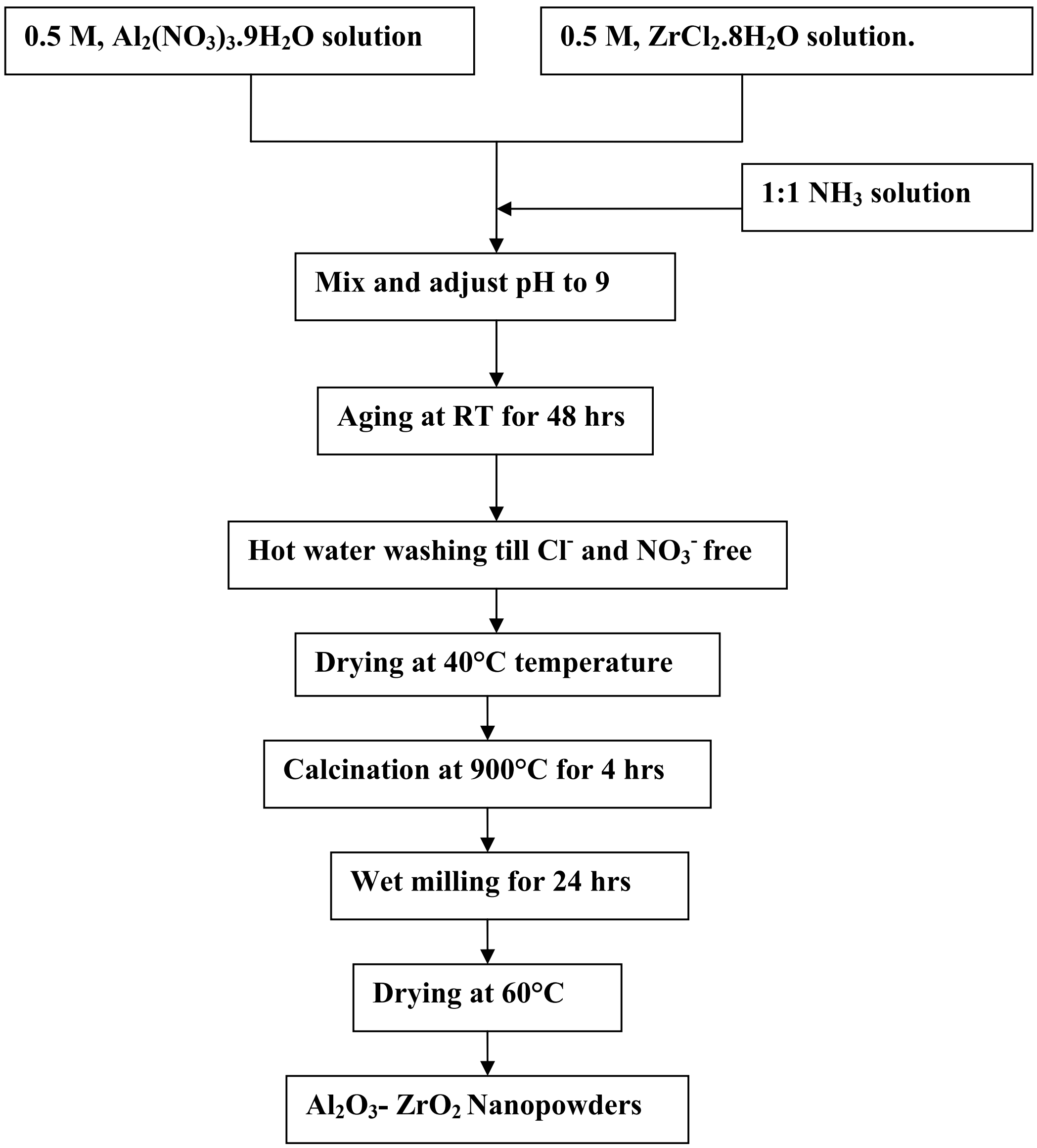
2.1 SYNTHESIS PROCESS

Aluminum nitrate (E-Merck, India) and zirconium oxychloride (E-Merck, India) were used as precursor for preparing high alumina–zirconia composite powder. Solutions of aluminum nitrate (0.5 M) and zirconium oxychloride (0.5 M) were mixed together in the required proportions to yield different Al_2O_3–$xZrO_2$ (where x = 15 mol %) batches. The mixed hydrogel was obtained by drop wise addition of 1:1NH3 hydroxide solution into the continuously stirred mixed aqueous solution of Al and Zr salt maintained at 25°C. The viscosity of the batch gradually increased and finally set to an enblock gel at pH ~ 9. The gels were then aged at room temperature for 48 h. Subsequently, the gel of each composition was washed repeatedly with boiling distilled water to remove chloride and nitrate ions and filtered. The filter cake was oven dried.

Table 2.1 Composition of the gel precursor

Identification	Al_2O_3 (mol %)	ZrO_2 (mol %)
A15Z	85	15

2.2 FLOW CHART OF THE SYNTHESIS OF Al_2O_3-ZrO_2



2.3 PRESSING

After the cake had fully dried, it was disintegrated into powder form. This powder was further pressed into pellets by a punch and dye arrangement. Pressing was done at 3 different loads (3, 5, 8 ton). The diameter of the die is 10mm. The arrangement is shown below.

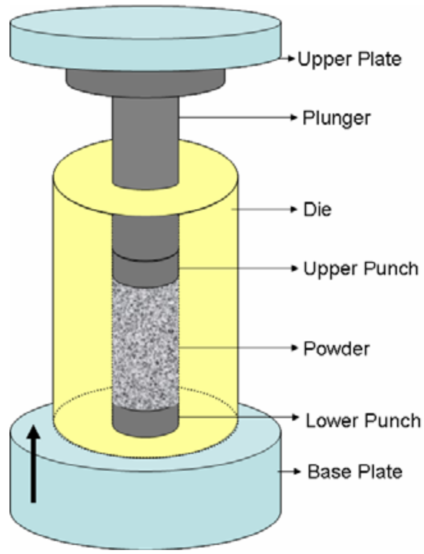
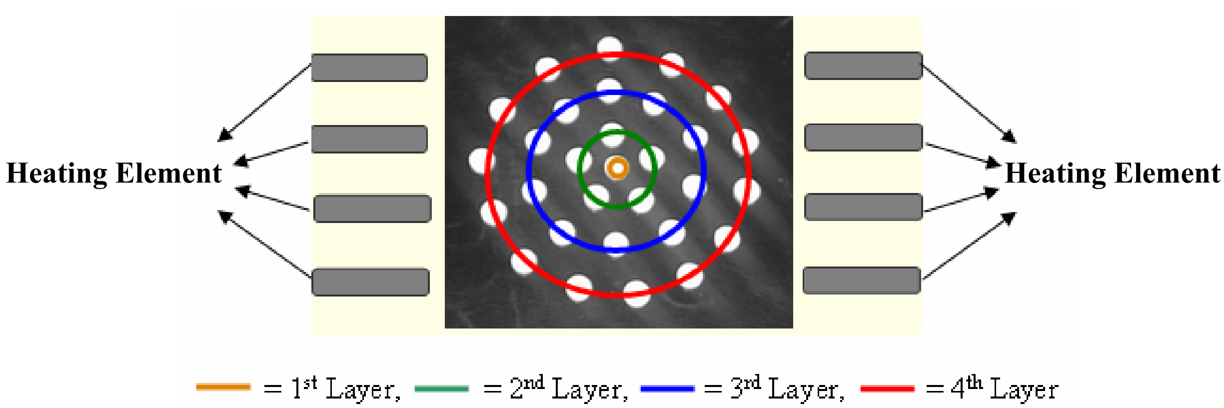


Fig 2.1: A schematic view of uniaxial pressing arrangement with Die-Plunger arrangement. The filled powder was compressed through upward direction. The diameter of the annular space of die was 10mm.

2.3 SINTERING

The pellets were calcined in air in a muffle furnace at 3 different temperatures 1500°C, 1550°C and 1600°C with a holding time of 2, 3 and 4 hrs at the corresponding peak temperatures. Before firing the pellets were kept in the furnace in a particular arrangement as shown below with the first layer receiving maximum heat from the coils and the inner layers lesser heat than each of its preceding layers. .



<u>**Fig 2.2**</u>: **Position of compact specimens during sintering within muffle furnace is represented by top view, eight super-kanthal heating elements are placed uniformly two-side of the cavity. The size of the cavity is 6" x 8". Different layer is indicating the pattern of the sample loading distribution**.

2.4 X-RAY DIFFRACTION

For phase analysis, XRD of the dried gel and calcined powders were carried out in a Philips X-ray diffractometer (PHILIPS PW1830), using Cu-Ka radiation. The voltage and current setting were 35 kV and 30 mA respectively. The samples were continuously scanned with a step size of 0.020 (2h) and a count time of 1 s per step. Silicon was used as an internal standard. The crystallite size of the synthesized powder was determined from X-ray line broadening using the Scherrer's equation as follows:

$$\mathbf{D = 0.9\,\lambda / (B \cos\theta)}$$

where D is the crystallite size (nm), k is the wavelength of 151 the X-ray radiation (1.54056 A˚), h is the Bragg's angle and B is the full width at half maximum(FWHM), where $B = (B_{meas}{}^{2} - B_{Equip}{}^{2})^{1/2}$. B_{meas} = Measured FWHM and B_{equip} = FWHM due to instrumental broadening.

2.5 NEURAL NETWORK PROGRAMMING

Simulation of relative density of ZTA Nano-composite has been carried out through MATLAB program. The simulation was focused on stepwise single layer to multilayered ANN theory. Following programs are written for the simulation:

1. single layer neural network
2. multilayer neural network
    - 2 layer ANN with 108 data point (4×1)
    - 2 layer ANN with 27 data point (5×1)
    - Three layer ANN with 108 data point (5×3×1)
    - Three layer Ann with 27 data points (5×3×1)

Chapter 3

RESULT AND DISCUSSION

3. RESULTS AND ANALYSIS

3.1 XRD ANALYSIS

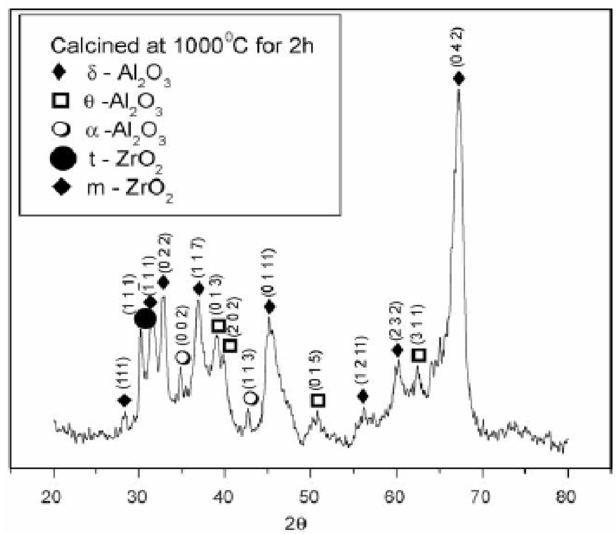


Fig 3.1 X-ray diffraction pattern

The sequence of phase evaluation in the calcined A15Z hydrogel was studied by XRD. The XRD pattern of the as dried gel (Fig. 3.1) shows broad peak of bayerite only. The broad peak of bayerite indicates the presence of fine crystallites (crystallite size 5–20 nm). XRD of hydrogel calcined at 200 °C have both bayerite (Al $(OH)_3$) and boehmite (Al(O)OH). Boehmite crystallizes from bayerite on heating according to the reaction:

$$\mathbf{Al\,(OH)_3 = Al(O)OH + H_2O}$$

3.2 RELATIVE DENSITIES

Load applied during pressing is 3, 5, 8 tons. The diameter of the specimen is 10mm. Particle size is around 20-190nm. Theoretical Density of Al_2O_3 is 3.98 gm/cc; t-ZrO_2 is 6.078 gm/cc; m-ZrO_2 is 5.815 gm/cc.

No. of sample/Batch = 30 (1^{st} Layer =1, 2^{nd} Layer =5, 3^{rd} Layer = 10, 4^{th} Layer =14)

Green Density (%) = 42.32(3 ton), 43.65(5ton), 43.82(8ton).

Table 3.1 Relative density of the compact specimens after sintering at muffle furnace.

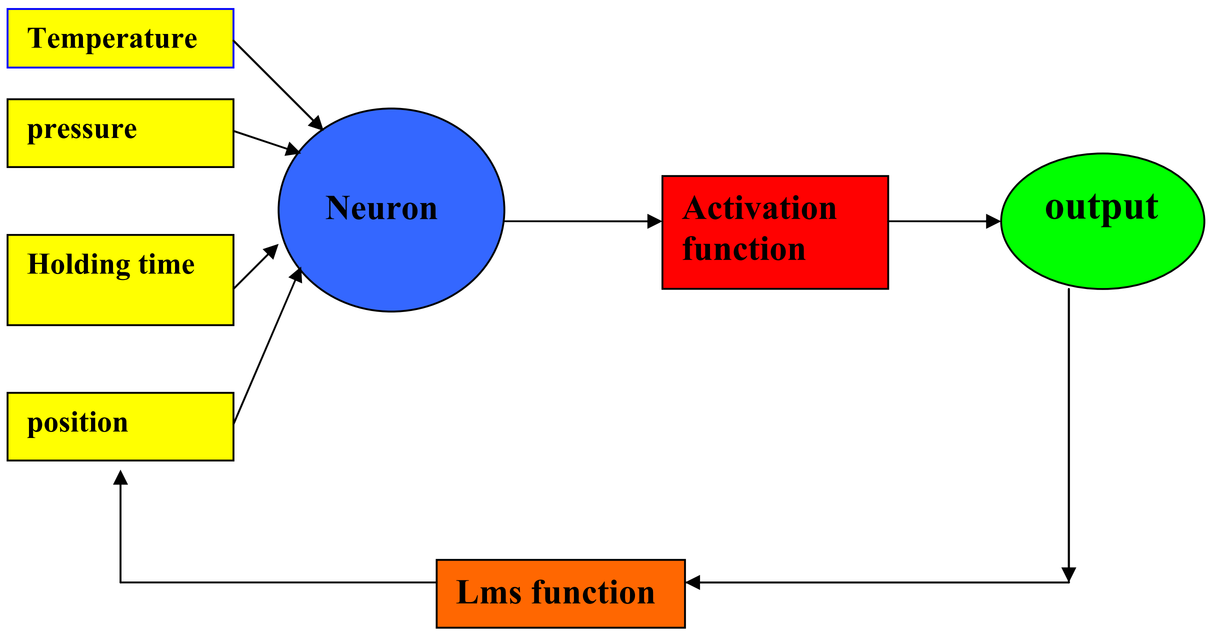
Batch no.	Temperature (° C)	Holding time (hr)	Pressure (Mpa)	Average Relative Density (%)			
				1st Layer	2nd Layer	3rd Layer	4th Layer
B1	1500	2	375	80.13	8.01	79.63	79.21
B2	1500	2	625	80.12	80.12	80.03	79.62
B3	1500	2	1000	80.93	80.67	79.95	80.06
B4	1500	3	375	80.61	80.93	80.23	80.45
B5	1500	3	625	81.72	81.23	80.12	80.32
B6	1500	3	1000	81.93	81.52	80.63	81.53
B7	1500	4	375	82.13	81.92	81.71	81.74
B8	1500	4	625	82.91	82.23	82.34	81.56
B9	1500	4	1000	82.98	82.65	82.47	82.49
B10	1550	2	375	85.63	85.32	85.41	85.02
B11	1550	2	625	85.98	85.43	86.22	87.47
B12	1550	2	1000	86.32	86.71	86.12	86.49
B13	1550	3	375	91.68	91.57	91.63	91.03
B14	1550	3	625	91.92	92.41	91.49	91.47
B15	1550	3	1000	92.68	92.31	92.62	92.81
B16	1550	4	375	9821	98.34	97.61	97.03
B17	1550	4	625	98.43	98.31	97.93	97.83
B18	1550	4	1000	98.61	98.52	97.68	97.33
B19	1600	2	375	98.33	97.92	97.62	97.22
B20	1600	2	625	98.01	97.83	97.07	97.45
B21	1600	2	1000	97.92	97.68	97.46	97.66
B22	1600	3	375	98.73	98.42	97.42	98.11
B23	1600	3	625	98.42	98.56	98.03	97.93
B24	1600	3	1000	98.36	98.17	98.37	98.01
B25	1600	4	375	98.86	98.35	97.48	97.11
B26	1600	4	625	98.92	98.62	98.02	97.95
B27	1600	4	1000	98.93	98.46	98.22	98.32

Chapter 4

PROGRAMMING

4. PROGRAMMING

4.1 SINGLE LAYER NEURAL NETWORK



clear all;

close all;

clc;

m=0.5;

wt=rand (1, 4);

t=[1500 1500 1500 1500 1500 1500 1500 1500 1500 1550 1550 1550 1550 1550 1550 1550 1550 1550 1600 1600 1600 1600 1600 1600 1600 1600 1600];

p=[375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000];

tt=[2 2 2 3 3 3 4 4 4 2 2 2 3 3 3 4 4 4 2 2 2 3 3 3 4 4 4];

result=[80.13 80.12 80.93 81.61 81.72 81.93 82.13 82.91 82.98 85.63 85.98 86.32 91.68 91.92 92.68 98.21 98.43 98.61 98.33 98.01 97.92 98.73 98.42 98.36 98.86 98.92 98.93];

p=(p)/max(p);

```
tt=tt/max(tt);
t=t/max(t);
result=result/100;
meane=zeros(1,1000);
For k=1:5000
    e=0;
For i=1:27
    slide=[t(i) p(i) tt(i) 1];
    af=wt*slide';
    a=logsig(af);
    e=result(i)-a;
    wt=wt+m*e*slide;
End

End


For j=1:27
    slide=[t(j) p(j) tt(j) 1];
    a=wt*slide';
    a1=logsig(a);
    (a1-result(j))*98.93
End
```

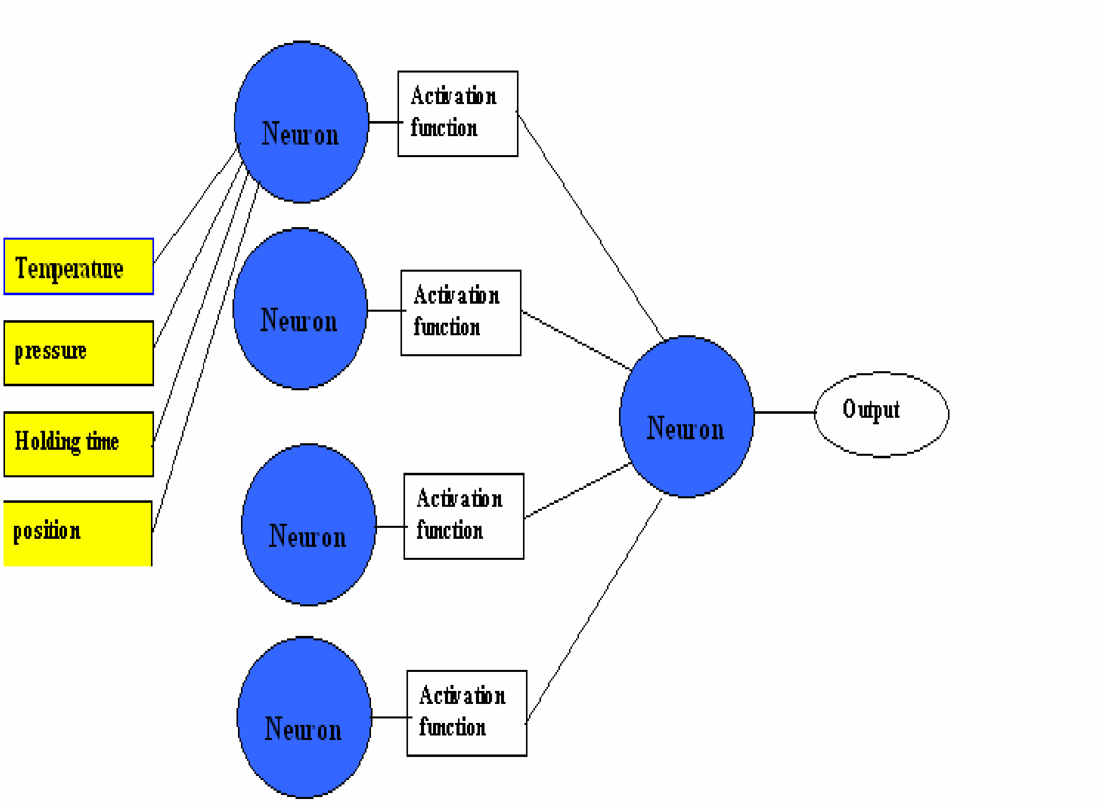
OUTPUT

Result	**NnResult**	**Error**
80.13	79.8651	-0.2248
80.12	80.3976	0.3124
80.93	81.1762	0.2821
81.61	85.1525	3.5354
81.72	85.5704	3.8403
81.93	86.1789	4.2348

82.13	89.2385	7.0569
82.91	89.5555	6.599
82.98	90.0155	6.985
85.63	90.9954	5.3086
85.98	91.2658	5.2304
86.32	91.6575	5.2824
91.68	93.5944	1.895
91.92	93.7921	1.8536
92.68	94.0778	1.385
98.21	95.4805	- 2.699
98.43	95.6227	- 2.7757
98.61	95.8279	- 2.7503
98.33	96.2611	- 2.0547
98.01	96.3796	- 1.6203
97.92	96.5507	- 1.3613
98.73	97.3840	- 1.3369
98.42	97.4678	-0.9469
98.36	97.5887	-0.7674
98.86	98.1760	-0.6802
98.92	98.2349	-0.681
98.93	98.3198	-0.6065

4.2 MULTILAYER NEURAL NETWORKS

4.2.1 TWO LAYER ANN WITH 108 DATA POINTS (4X1)



clear all

close all

t=[1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,
1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,150
0,1500,1500,1500,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1
550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550
,1550,1550,1550,1550,1550,1550,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,16
00,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,
1600,1600,1600,1600,1600,1600,1600,1600,1600];

ht=[2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,2,2,2,2,2,2,2,2,2,2,2,2
,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,4,4
,4,4,4,4,4,4,4,4,4,4];

```
pr=[375,375,375,375,625,625,625,625,1000,1000,1000,1000,375,375,375,375,625,625,625,6
25,1000,1000,1000,1000,375,375,375,375,625,625,625,625,1000,1000,1000,1000,375,375,3
75,375,625,625,625,625,1000,1000,1000,1000,375,375,375,375,625,625,625,625,1000,1000,
1000,1000,375,375,375,375,625,625,625,625,1000,1000,1000,1000,375,375,375,375,625,62
5,625,625,1000,1000,1000,1000,375,375,375,375,625,625,625,625,1000,1000,1000,1000,37
5,375,375,375,625,625,625,625,1000,1000,1000,1000];

p=
[1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2
,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4
,1,2,3,4,1,2,3,4];

rd=[80.13,80.01,79.63,79.21,80.12,80.12,80.03,79.62,80.93,80.67,79.95,80.06,81.61,80.93,8
0.23,80.45,81.72,81.23,80.12,80.32,81.93,81.52,80.63,81.53,82.13,81.92,81.71,81.74,82.91,8
2.23,82.34,81.56,82.98,82.65,82.47,82.49,85.63,85.32,85.41,85.02,85.98,85.43,86.22,87.47,8
6.32,86.71,86.12,86.49,91.68,91.57,91.63,91.03,91.92,92.41,91.49,91.47,92.68,92.31,92.62,9
2.81,98.21,98.34,97.61,97.03,98.43,98.31,97.93,97.83,98.61,98.52,97.68,97.33,98.33,97.92,9
7.62,97.22,98.01,97.83,97.07,97.45,97.92,97.68,97.46,97.66,98.73,98.42,97.42,98.11,98.42,9
8.56,98.03,97.93,98.36,98.17,98.37,98.01,98.86,98.35,97.48,97.11,98.92,98.62,98.02,97.95,9
8.93,98.46,98.22,98.32];

t=t/max(t);
ht=ht/max(ht);
p=p/max(p);
pr=pr/max(pr);
rd=rd/max(rd);

n1=4;
n2=4;
n3=1;

for i=1:n1
        w1(1:n2,i)=rand(n2,1);
end
b1(1:n2,1)=rand(n2,1);
```

```
w2(1:n2)=rand(1,n2);

b2=rand;
for k=1:108
  for z=1:1000
            a(1:n1)=[t(k) ht(k) pr(k) rd(k)];          (%INPUT TO THE NEURAL NETWORK)

                a1=logsig(w1*a' + b1);                      (%FIRST FUNCTION ...............)

        a2=(w2(1:n2)*a1 + b2);                      (%SECOND FUNCTION...............)

        e= rd(k)-a2;
                s2= -2*1*e;
               b=(1-a1).*a1;
        b= [b(1) 0 0 0; 0 b(2) 0 0; 0 0 b(3) 0;0 0 0 b(4)];
                s1= b*w2'*s2;
                w2=w2-(.05*s2*a1');
        b2=b2-.05*s2;
        for l=1:4
            w1(1:n2,l)=w1(1:n2,l)-(.05*s1*a(l));
        end
        b1=b1-.05*s1;
    end
end

for k=1:108
   a(1:n1)=[t(k) ht(k) pr(k) rd(k)];         %INPUT TO THE NEURAL NETWORK
   a1=logsig(w1*a' + b1);                    %FIRST FUNCTION ...............
   a2=(w2(1:n2)*a1 + b2);                    %SECOND FUNCTION...............
   err(k)= (rd(k)-a2)*98.93;
   b5(k)=a2*98.93;
end
```

OUTPUT

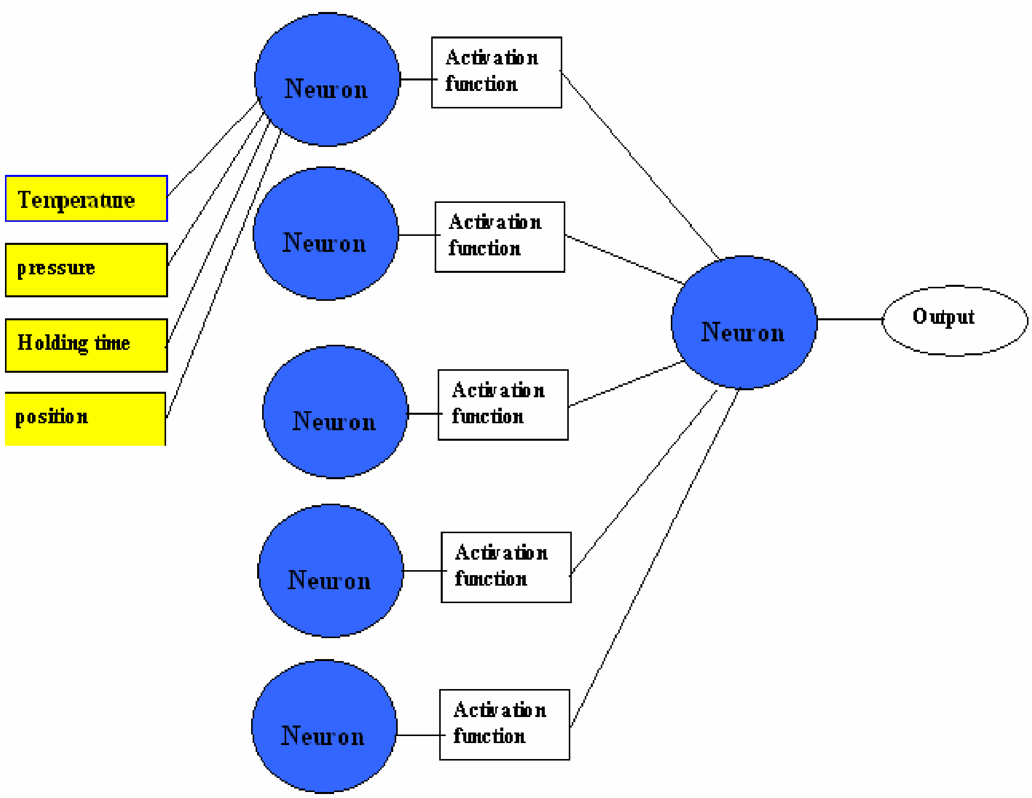
Result	NnResult	Error
80.13	94.123	-13.993
80.01	94.1189	-14.1089
79.63	94.1057	-14.4757
79.21	94.0912	-14.8812
80.12	94.729	-14.609
80.12	94.729	-14.609
80.03	94.7261	-14.6961
79.62	94.7128	-15.0928
80.93	95.5937	-14.6637
80.67	95.586	-14.916
79.95	95.5646	-15.6146
80.06	95.5679	-15.5079
81.61	95.4749	-13.8649
80.93	95.4531	-14.5231
80.23	95.4307	-15.2007
80.45	95.4378	-14.9878
81.72	96.0341	-14.3141
81.23	96.0193	-14.7893
80.12	95.9859	-15.8659
80.32	95.9919	-15.6719
81.93	96.8082	-14.8782
81.52	96.797	-15.277
80.63	96.7726	-16.1426
81.53	96.7972	-15.2672
82.13	96.72	-14.59
81.92	96.7138	-14.7938
81.71	96.7076	-14.9976
81.74	96.7085	-14.9685
82.91	97.2509	-14.3409
82.23	97.2321	-15.0021
82.34	97.2352	-14.8952

81.56	97.2135	-15.6535
82.98	97.9538	-14.9738
82.65	97.9455	-15.2955
82.47	97.941	-15.471
82.49	97.9415	-15.4515
85.63	94.3074	-8.6774
85.32	94.2969	-8.9769
85.41	94.2999	-8.8899
85.02	94.2867	-9.2667
85.98	94.9136	-8.9336
85.43	94.896	-9.466
86.22	94.9213	-8.7013
87.47	94.9611	-7.4911
86.32	95.7469	-9.4269
86.71	95.7583	-9.0483
86.12	95.7411	-9.6211
86.49	95.7519	-9.2619
91.68	95.7876	-4.1076
91.57	95.7842	-4.2142
91.63	95.786	-4.156
91.03	95.7674	-4.7374
91.92	96.331	-4.411
92.41	96.3453	-3.9353
91.49	96.3185	-4.8285
91.47	96.3179	-4.8479
92.68	97.0919	-4.4119
92.31	97.0821	-4.7721
92.62	97.0903	-4.4703
92.81	97.0953	-4.2853
98.21	97.1785	1.0315
98.34	97.1822	1.1578
97.61	97.1617	0.4483
97.03	97.1454	-0.1154
98.43	97.665	0.765

98.31	97.6619	0.6481
97.93	97.6519	0.2781
97.83	97.6492	0.1808
98.61	98.3319	0.2781
98.52	98.3298	0.1902
97.68	98.3096	-0.6296
97.33	98.3013	-0.9713
98.33	94.7269	3.6031
97.92	94.7135	3.2065
97.62	94.7036	2.9164
97.22	94.6905	2.5295
98.01	95.2868	2.7232
97.83	95.2812	2.5488
97.07	95.2578	1.8122
97.45	95.2695	2.1805
97.92	96.0737	1.8463
97.68	96.0669	1.6131
97.46	96.0608	1.3992
97.66	96.0664	1.5936
98.73	95.9994	2.7306
98.42	95.99	2.43
97.42	95.9597	1.4603
98.11	95.9806	2.1294
98.42	96.5137	1.9063
98.56	96.5177	2.0423
98.03	96.5026	1.5274
97.93	96.4997	1.4303
98.36	97.2355	1.1245
98.17	97.2305	0.9395
98.37	97.2357	1.1343
98.01	97.2264	0.7836
98.86	97.1928	1.6672
98.35	97.1785	1.1715
97.48	97.1541	0.3259

97.11	97.1437	-0.0337
98.92	97.6735	1.2465
98.62	97.6656	0.9544
98.02	97.6498	0.3702
97.95	97.648	0.302
98.93	98.3346	0.5954
98.46	98.3233	0.1367
98.22	98.3176	-0.0976
98.32	98.32	0

4.2.2 TWO LAYER ANN WITH 27 DATA POINTS (5X1)



close all;

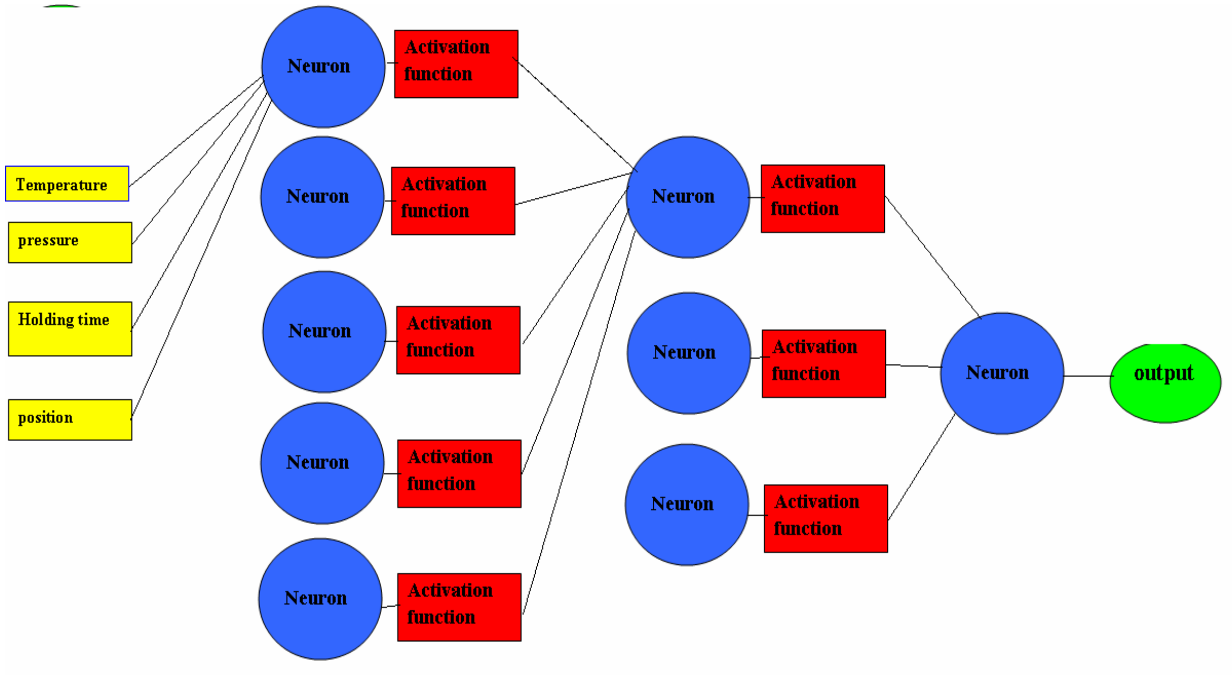
```
clear all;
clc;

t=[1500 1500 1500 1500 1500 1500 1500 1500 1500 1550 1550 1550 1550 1550 1550 1550
1550 1550 1600 1600 1600 1600 1600 1600 1600 1600 1600];

p=[375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000 375
625 1000 375 625 1000 375 625 1000];

ht=[2 2 2 3 3 3 4 4 4 2 2 2 3 3 3 4 4 4 2 2 2 3 3 3 4 4 4];

rd=[80.13 80.12 80.93 81.61 81.72 81.93 82.13 82.91 82.98 85.63 85.98 86.32 91.68 91.92
92.68 98.21 98.43 98.61 98.33 98.01 97.92 98.73 98.42 98.36 98.86 98.92 98.93];

t=t/max(t);
ht=ht/max(ht);
p=p/max(p);
rd=rd/max(rd);
b1=rand(5,1);
b2=rand(1,1);

% 1st layer 5 neurons and 2nd layer 1neuron

wt1=rand(5,4);
wt2=rand(1,5);
for j=1:1000
  avg_delta_wt1=zeros(5,4);
  avg_delta_wt2=zeros(5,1);
  avg_delta_b1=zeros(5,1);
  avg_delta_b2=zeros(1,1);
  for i=1:27
      %forward path
      slide=[t(i), ht(i),1,p(i)];
      a1=wt1*slide';
```

```
        a2=logsig(a1+b1);
        a3=wt2*a2;
        a4=logsig(a3+b2);
        res=rd(i);
        %backward path
        err=res-a4;
        s2=-2*(1-a4)*a4*err;
        s1=[(1-a2(1,1))*a2(1,1) 0 0 0 0;0 (1-a2(2,1))*a2(2,1) 0 0 0 ;0 0 (1-a2(3,1))*a2(3,1)
            0 0;0 0 0 (1-a2(4,1))*a2(4,1) 0;0 0 0 0 (1-a2(5,1))*a2(5,1)]*wt2'*s2];
        delta_wt2=-0.1*s2*a2;
        delta_wt1=-0.1*s1*slide;
        avg_delta_wt2=avg_delta_wt2+delta_wt2/27;
        avg_delta_wt1=avg_delta_wt1+delta_wt1/27;
        avg_delta_b1=avg_delta_b1-.1*s1/27;
        avg_delta_b2=avg_delta_b2-.1*s2/27;
    end
    wt1=wt1+avg_delta_wt1;
    wt2=wt2+avg_delta_wt2';
    b1=b1+avg_delta_b1;
    b2=b2+avg_delta_b2;
end
for k=1:27
    slide=[t(k), ht(k),1,p(k)];
    b1=wt1*slide';
    b2=logsig(b1);
    b3=wt2*b2;
    b4=logsig(b3);
    res=rd(k);
    err2(k)=(res-b4)*98.93;
    b5(k)=b4*98.93;
end
```

OUTPUT

Result	NnResult	Error
80.13	82.084	-1.954
80.12	82.6033	-2.4833
80.93	83.2429	-2.3129
81.61	82.4349	-0.8249
81.72	82.9019	-1.1819
81.93	83.4759	-1.5459
82.13	82.7465	-0.6165
82.91	83.1665	-0.2565
82.98	83.682	-0.702
85.63	82.1484	3.4816
85.98	82.6587	3.3213
86.32	83.287	3.033
91.68	82.4931	9.1869
91.92	82.9519	8.9681
92.68	83.5157	9.1643
98.21	82.7991	15.4109
98.43	83.2117	15.2183
98.61	83.7178	14.8922
98.33	82.2116	16.1184
98.01	82.7132	15.2968
97.92	83.3303	14.5897
98.73	82.5502	16.1798
98.42	83.001	15.419
98.36	83.5546	14.8054
98.86	82.8507	16.0093
98.92	83.256	15.664
98.93	83.753	15.177

4.2.3 THREE LAYER ANN WITH 108 DATA POINTS (5X3X1)




```
clear all;
close all;
clc;

t=[1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,
1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,150
0,1500,1500,1500,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1
550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550,1550
,1550,1550,1550,1550,1550,1550,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,16
00,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,1600,
1600,1600,1600,1600,1600,1600,1600,1600,1600];

ht=[2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,2,2,2,2,2,2,2,2,2,2,2,2
,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,4,4
,4,4,4,4,4,4,4,4,4,4];

pr=[375,375,375,375,625,625,625,625,1000,1000,1000,1000,375,375,375,375,625,625,625,6
25,1000,1000,1000,1000,375,375,375,375,625,625,625,625,1000,1000,1000,1000,375,375,3
75,375,625,625,625,625,1000,1000,1000,1000,375,375,375,375,625,625,625,625,1000,1000,
```

```
1000,1000,375,375,375,375,625,625,625,625,1000,1000,1000,1000,375,375,375,375,625,62
5,625,625,1000,1000,1000,1000,375,375,375,375,625,625,625,625,1000,1000,1000,1000,37
5,375,375,375,625,625,625,625,1000,1000,1000,1000];

p=[1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,
1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,
3,4,1,2,3,4,1,2,3,4];

rd=[80.13,80.01,79.63,79.21,80.12,80.12,80.03,79.62,80.93,80.67,79.95,80.06,81.61,80.93,8
0.23,80.45,81.72,81.23,80.12,80.32,81.93,81.52,80.63,81.53,82.13,81.92,81.71,81.74,82.91,8
2.23,82.34,81.56,82.98,82.65,82.47,82.49,85.63,85.32,85.41,85.02,85.98,85.43,86.22,87.47,8
6.32,86.71,86.12,86.49,91.68,91.57,91.63,91.03,91.92,92.41,91.49,91.47,92.68,92.31,92.62,9
2.81,98.21,98.34,97.61,97.03,98.43,98.31,97.93,97.83,98.61,98.52,97.68,97.33,98.33,97.92,9
7.62,97.22,98.01,97.83,97.07,97.45,97.92,97.68,97.46,97.66,98.73,98.42,97.42,98.11,98.42,9
8.56,98.03,97.93,98.36,98.17,98.37,98.01,98.86,98.35,97.48,97.11,98.92,98.62,98.02,97.95,9
8.93,98.46,98.22,98.32];

t=t/max(t);
ht=ht/max(ht);
p=p/max(p);
pr=pr/max(pr);
rd=rd/max(rd);

% 1st layer 5 neurons and 2nd layer 3 neuron and 3rd layer 1 neuron
wt1=rand(5,4);
wt2=rand(3,5);
wt3=rand(1,3);
b=rand(1,1);
for j=1:1000
   avg_delta_wt1=zeros(5,4);
   avg_delta_wt2=zeros(3,5);
   avg_delta_wt3=zeros(1,3);
   for i=1:108
      %forward path
      slide=[t(i), ht(i),pr(i),p(i)];
```

```
        a1=logsig(wt1*slide');
        a2=logsig(wt2*a1);
        a3=logsig(wt3*a2+b);
        res=rd(i);

        %backward path
        err=res-a3;
        s3=-2*(1-a3)*a3*err;
        s2=[(1-a2(1,1))*a2(1,1) 0 0;0 (1-a2(2,1))*a2(2,1) 0;0 0 (1-
           a2(3,1))*a2(3,1)]*wt3'*s3];
        s1=[1-a1(1,1))*a1(1,1) 0 0 0 0;0 (1-a1(2,1))*a1(2,1) 0 0 0 ;0 0
           (1-a1(3,1))*a1(3,1) 0 0;0 0 0 (1-a1(4,1))*a1(4,1) 0;0 0 0 0
           (1-a1(5,1))*a1(5,1)]*wt2'*s2];
        delta_wt3=-0.1*s3*a2;
        delta_wt2=-0.1*s2*a1';
        delta_wt1=-0.1*s1*slide;
        avg_delta_wt3=avg_delta_wt3+delta_wt3'/108;
        avg_delta_wt2=avg_delta_wt2+delta_wt2/108;
        avg_delta_wt1=avg_delta_wt1+delta_wt1/108;
      end
    wt1=wt1+avg_delta_wt1;
    wt2=wt2+avg_delta_wt2;
    wt3=wt3+avg_delta_wt3;
    b=b-0.1*s3;
end
for k=1:108
      slide=[t(k), ht(k),pr(k),p(k)];
      b1=logsig(wt1*slide');
      b2=logsig(wt2*b1);
      b3=wt3*b2;
      res=rd(k);
      err2(k)=(res-b3)*98.93;
end
```

OUTPUT

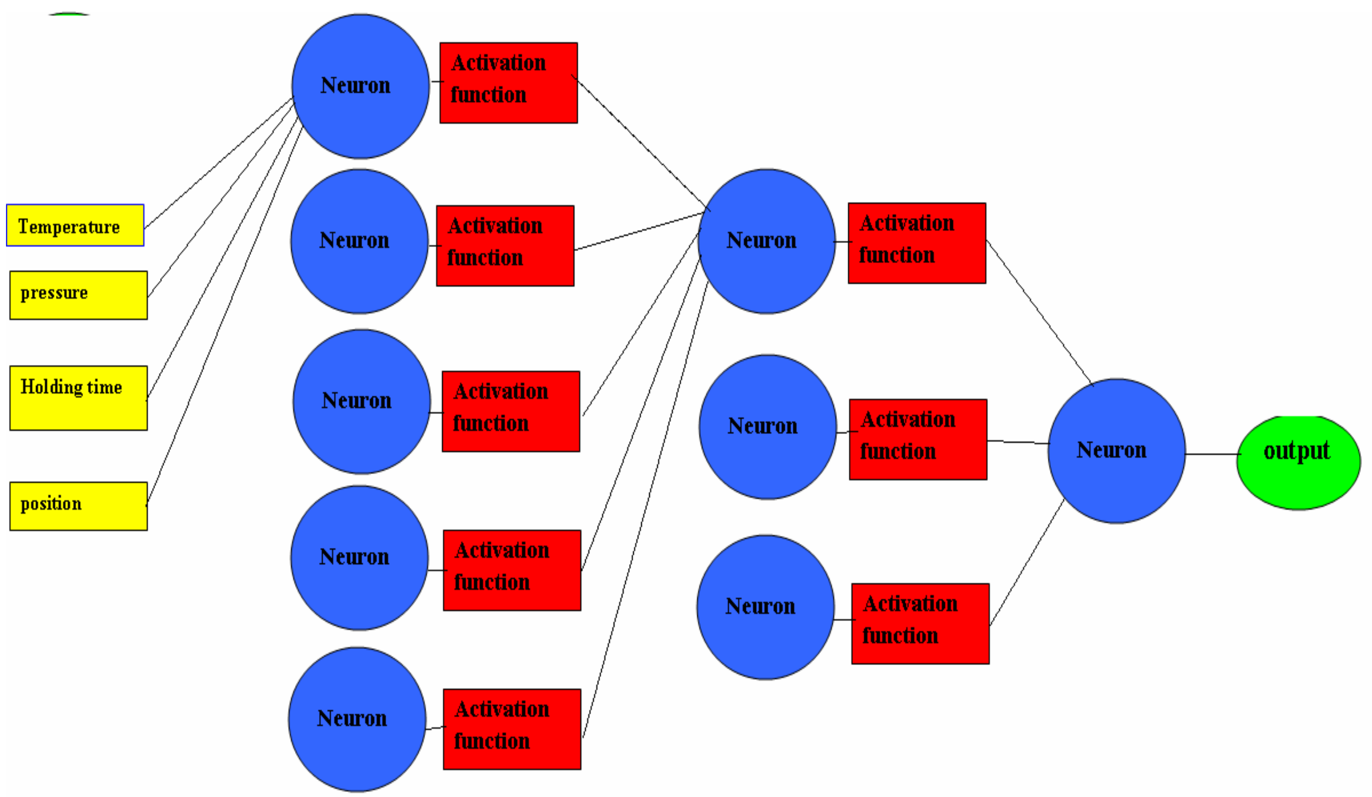
Result	NnResult	Error
80.13	89.8989	-9.7689
80.01	90.0481	-10.0381
79.63	90.1751	-10.5451
79.21	90.2831	-11.0731
80.12	90.0598	-9.9398
80.12	90.1866	-10.0666
80.03	90.2944	-10.2644
79.62	90.386	-10.766
80.93	90.2534	-9.3234
80.67	90.3528	-9.6828
79.95	90.4374	-10.4874
80.06	90.5095	-10.4495
81.61	89.9929	-8.3829
80.93	90.1297	-9.1997
80.23	90.2459	-10.0159
80.45	90.3447	-9.8947
81.72	90.1399	-8.4199
81.23	90.2560	-9.026
80.12	90.3547	-10.2347
80.32	90.4385	-10.1185
81.93	90.3164	-8.3864
81.52	90.4075	-8.8875
80.63	90.4850	-9.855
81.53	90.5511	-9.0211
82.13	90.0788	-7.9488
81.92	90.2040	-8.284
81.71	90.3104	-8.6004
81.74	90.4007	-8.6607
82.91	90.2129	-7.3029
82.23	90.3193	-8.0893

82.34	90.4096	-8.0696
81.56	90.4864	-8.9264
82.98	90.3737	-7.3937
82.65	90.4573	-7.8073
82.47	90.5284	-8.0584
82.49	90.5890	-8.099
85.63	89.9161	-4.2861
85.32	90.063	-4.743
85.41	90.1879	-4.7779
85.02	90.2942	-5.2742
85.98	90.0745	-4.0945
85.43	90.1992	-4.7692
86.22	90.3053	-4.0853
87.47	90.3955	-2.9255
86.32	90.265	-3.945
86.71	90.3628	-3.6528
86.12	90.4461	-4.3261
86.49	90.5171	-4.0271
91.68	90.0087	1.6713
91.57	90.1432	1.4268
91.63	90.2576	1.3724
91.03	90.3548	0.6752
91.92	90.1534	1.7666
92.41	90.2676	2.1424
91.49	90.3647	1.1253
91.47	90.4472	1.0228
92.68	90.327	2.353
92.31	90.4166	1.8934
92.62	90.493	2.127
92.81	90.558	2.252
98.21	90.0932	8.1168
98.34	90.2165	8.1235
97.61	90.3211	7.2889
97.03	90.4101	6.6199

98.43	90.2252	8.2048
98.31	90.3299	7.9801
97.93	90.4188	7.5112
97.83	90.4944	7.3356
98.61	90.3834	8.2266
98.52	90.4657	8.0543
97.68	90.5357	7.1443
97.33	90.5953	6.7347
98.33	89.933	8.397
97.92	90.0776	7.8424
97.62	90.2006	7.4194
97.22	90.3051	6.9149
98.01	90.0889	7.9211
97.83	90.2117	7.6183
97.07	90.3161	6.7539
97.45	90.4049	7.0451
97.92	90.2764	7.6436
97.68	90.3727	7.3073
97.46	90.4547	7.0053
97.66	90.5245	7.1355
98.73	90.0242	8.7058
98.42	90.1566	8.2634
97.42	90.2692	7.1508
98.11	90.3648	7.7452
98.42	90.1666	8.2534
98.56	90.279	8.281
98.03	90.3746	7.6554
97.93	90.4558	7.4742
98.36	90.3374	8.0226
98.17	90.4257	7.7443
98.37	90.5008	7.8692
98.01	90.5648	7.4452
98.86	90.1074	8.7526
98.35	90.2287	8.1213

97.48	90.3317	7.1483
97.11	90.4192	6.6908
98.92	90.2374	8.6826
98.62	90.3404	8.2796
98.02	90.4279	7.5921
97.95	90.5022	7.4478
98.93	90.393	8.537
98.46	90.474	7.986
98.22	90.5429	7.6771
98.32	90.6016	7.7184

4.4 THREE LAYER ANN WITH 27 DATA POINTS (5X3X1)



```
t=[1500 1500 1500 1500 1500 1500 1500 1500 1500 1550 1550 1550 1550 1550 1550 1550
1550 1550 1600 1600 1600 1600 1600 1600 1600 1600 1600];

pr=[375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000 375 625 1000
375 625 1000 375 625 1000 375 625 1000];

ht=[2 2 2 3 3 3 4 4 4 2 2 2 3 3 3 4 4 4 2 2 2 3 3 3 4 4 4];

rd=[80.13 80.12 80.93 81.61 81.72 81.93 82.13 82.91 82.98 85.63 85.98 86.32 91.68 91.92
92.68 98.21 98.43 98.61 98.33 98.01 97.92 98.73 98.42 98.36 98.86 98.92 98.93];

t=t/max(t);
ht=ht/max(ht);
%p=p/max(p);
pr=pr/max(pr);
rd=rd/max(rd);

% 1st layer 5 neurons and 2nd layer 3 neuron and 3rd layer 1 neuron

wt1=rand(5,4);
wt2=rand(3,5);
wt3=rand(1,3);
b=rand(1,1);
for j=1:1000
  avg_delta_wt1=zeros(5,4);
  avg_delta_wt2=zeros(3,5);
  avg_delta_wt3=zeros(1,3);
  for i=1:27
     %forward path
     slide=[t(i), ht(i),pr(i),1];
     a1=logsig(wt1*slide');
     a2=logsig(wt2*a1);
     a3=logsig(wt3*a2+b);
     res=rd(i);
```

```
        %backward path
        err=res-a3;
        s3=-2*(1-a3)*a3*err;
        s2=[(1-a2(1,1))*a2(1,1) 0 0;0 (1-a2(2,1))*a2(2,1) 0;0 0
              (1-a2(3,1))*a2(3,1)]*wt3'*s3];
        s1=[ (1-a1(1,1))*a1(1,1) 0 0 0 0;0 (1-a1(2,1))*a1(2,1) 0 0 0 ;0 0
           (1-a1(3,1))*a1(3,1) 0 0;0 0 0 (1-a1(4,1))*a1(4,1) 0;0 0 0 0
           (1-a1(5,1))*a1(5,1)]*wt2'*s2];

        delta_wt3=-0.1*s3*a2;
        delta_wt2=-0.1*s2*a1';
        delta_wt1=-0.1*s1*slide;
        avg_delta_wt3=avg_delta_wt3+delta_wt3'/27;
        avg_delta_wt2=avg_delta_wt2+delta_wt2/27;
        avg_delta_wt1=avg_delta_wt1+delta_wt1/27;
        avg_
    end

   wt1=wt1+avg_delta_wt1;
   wt2=wt2+avg_delta_wt2;
   wt3=wt3+avg_delta_wt3;
   b=b-0.1*s3;
end
for k=1:27
    slide=[t(k), ht(k),pr(k),1];
    b1=logsig(wt1*slide');
    b2=logsig(wt2*b1);
    b3=logsig(wt3*b2+b);
    res=rd(k);
    err2(k)=(res-b3)*98.93;
    b4(k)=b3*98.93;
end
```

OUTPUT

Result	NnResult	Error
80.13	92.6813	-12.5513
80.12	92.6868	-12.5668
80.93	92.693	-11.763
81.61	92.6863	-11.0763
81.72	92.6909	-10.9709
81.93	92.696	-10.766
82.13	92.6904	-10.5604
82.91	92.6942	-9.7842
82.98	92.6985	-9.7185
85.63	92.6819	-7.0519
85.98	92.6873	-6.7073
86.32	92.6933	-6.3733
91.68	92.6868	-1.0068
91.92	92.6912	-0.7712
92.68	92.6963	-0.0163
98.21	92.6908	5.5192
98.43	92.6945	5.7355
98.61	92.6988	5.9112
98.33	92.6824	5.6476
98.01	92.6877	5.3223
97.92	92.6937	5.2263
98.73	92.6872	6.0428
98.42	92.6916	5.7284
98.36	92.6966	5.6634
98.86	92.6912	6.1688
98.92	92.6948	6.2252
98.93	92.699	6.231

Chapter 5

CONCLUSION

5. CONCLUSION

Using ANN we synchronized the various parameters (**pressure, temperature, holding time and specimen position with respect to the heating element**); this synchronization led to the prediction of relative density of the pellets at any temperature, pressure, holding time and position with respect to the furnace. This simulation was done on different layer forms like single layer and multilayered. Finally the relative density of the pellets was taken, using this result a simulation was done on MATLAB programming using the concept of Artificial Neural Networks.

Since the data was very random and with several governing parameters a normal single layer network could not be used to provide the desired results. Multi layer network illustrates

(a) **(4×1 and 5×1)** not able to provide the desired results because of the large amount of randomness in the data provided and the small number of data points available.

(b) range of results of

- **Single layer**: -2.7757 to 7.0569
- **Multilayer**(4x1): -15.8659 to 3.6831
- **Multilayer**(5x1): -2.4833 to 16.498

(c) **(5×3×1)** ANN program more suitable for the above experiments, however, more number of data points could synchronize smoother curve and predict desirable densification data.

Hence we would suggest that with greater number of data points we can predict the relative density better and that would help in further research.

Chapter 6

REFERENCES

6. REFERENCES

- Sarkar, D, Mahapatra. D, Ray. S, Bhattacharyya. S, Adak. S, Mitra. N "Nanostructured Al_2O_3–ZrO_2 composite synthesized by sol–gel technique: powder processing and microstructure." Springer Science+ Buisness Media, Accepted-28 july 2006. (In Press).
- Sarkar, D, Mahapatra. D, Ray. S, Bhattacharyya. S, Adak. S, Mitra. N, "Synthesis and characterization of sol–gel derived ZrO_2 doped Al_2O_3 nanopowder" Ceramics International (Elsevier), Accepted-2 May 2006.
- Sarkar, D, Adak. S, Mitra. N," Preparation and characterization of an Al_2O_3-ZrO_2 nanocomposite,Part-1 powder synthesis and transformation behavior during fracture" Accepted 10 January 2006, available online 24 February 2006.
- Mucahit Sutcu , Sedat Akkurt, "ANN model for prediction of powder packing" Ecers(Elsevier), Available online 19 May 2006
- J.O. Olajide , J.C. Igbeka , T.J. Afolabi , O.A. Emiola, "Prediction of oil yield from groundnut kernels in an hydraulic press using artificial neural network (ANN)", Received 11 April 2005; accepted 13 June 2006,Available online 23 February 2007.
- C. Lombardi and A. Mazzola, "Prediction of two-phase mixture density using artificial neural networks" ***Ann.*** *Nucl. Energy,* Vol. 24, No. 17, pp. 1373-1387, 1997.