On line monitoring of crystallization process using FBRM and artificial neural network

Lukan Jiang

University of Claude Bernard Lyon 1
43 bd du 11 novembre
69622 Villeurbanne cedex, France

Academic year 2005-2006

Master of SIDS

On line monitoring of crystallization process using FBRM and artificial neural network

04/2006-07/2006

By:JIANG Lukan

Tutor: P. Dufour (LAGEP-UCBL1)
G. Fevotte (LAGEP-UCBL1)
S. Bhartiya (Dept. Chem. Eng., IIT Bombay, India)

TABLE OF CONTENTS

ACKNOLEDGEMENT

I express my deep sense of gratitude and appreciation to all my teachers for making learning and education a worthy goal.

I also thank all of my teachers who inspired me take a close look at the arguments presented in the classroom. Mr. Pascal DUFOUR trained me to think in terms of control issues relevant to process industry practice; he helped me to stay focused on the artificial neural network and encouraged me to work on many problems. The using of FBRM and principle of crystallization is due to Mr. Gills FEVOTTE. I thank Mr. PORTRAT Vincent for providing experiment data which makes possible the work. I thank Mr. Sharad BHARTIYA, he gave me abstract notion in artificial neural network and respond my queries although we have never seen each other.

I owe a special not of thanks to Mr. Hassam HAMMOURI for providing guidance and opportunities. I also thank the Department of GSA (Génie des systèmes automatisés) and LAGEP (Laboratoire d'Automatique et de Génie des Procédés).

CHAPTER 1 INTRODUCTION

Monitoring and control of crystallization processes is based on information of crystal size distribution (CSD). CSD monitoring, however, has been based on sampling and off-line analysis by different methods. However, the growing demand for improving the process controls.

Along the last 20 years the evolution in process instrumentation has resulted in the development of techniques for in-line measurement of variable such as particle size distribution (PSD), which is in most cases the most important information in the monitoring of industrial crystallization processes.

Laboratory experiments were carried out for measuring the chord length distribution (CLD) of different particle systems using a laser reflection sensor. Samples consisted of adipic acid particles of different size, ranging from 0 to 400 μ m. The experimental results, consisting of the particle number counting per chord length class, were used in fitting a neural network model for estimating the number-based size distribution. The results indicate the feasibility of using such a model as a software sensor in crystallization processes monitoring.

In a review on the methodology and problems related with the use of FBRM, the CLD data is the primary information provided by the equipment. In order to become useful as a process sensor, the CLD data must be transformed into a number or volume based PSD. In present work, the method is based on fitting a neural network (NN) software sensor, which is proposed for transforming an FBRM sensor into a number-based on line PSD estimated measure. (Figure 1-1)

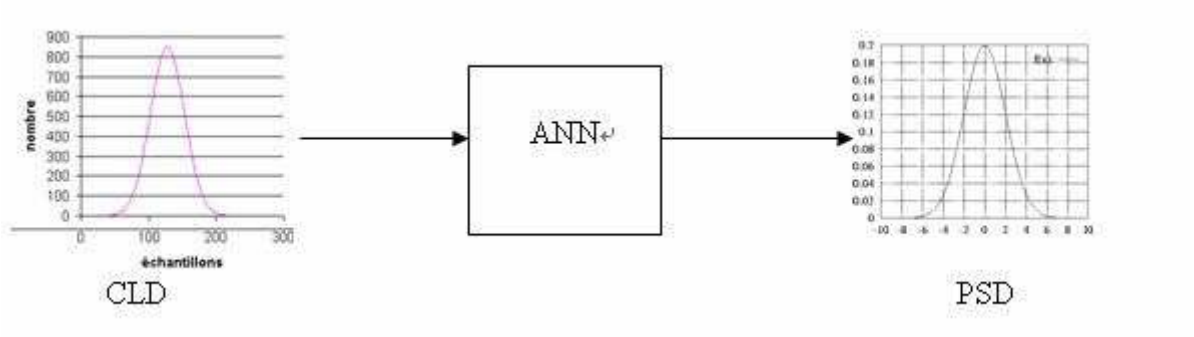


Figure 1-1 The principal idea

In the chapter 2, I will present some basic knowledge about measurement and crystallization process. The principle methods for focus on using improved NN models will be introduced in chapter 3.The laboratory experimental manipulation about crystallization forms the contents of chapter 4. All sampling data will play the important role in NN learning model, in chapter 5, I will say some idea about how to carry out my design work of NN model and the chapter 6 presents the result of NN simulation. At last, chapter 7, it's conclusion.

Chapter 2 The measurement of size of particle in crystallization:

Basic principles of crystallization:

Crystallization is the process of formation of solid crystals (Figure 2-1) from a homogeneous solution. The natural process is also known as crystal growth. Crystallization is also a chemical solid-liquid separation technique.

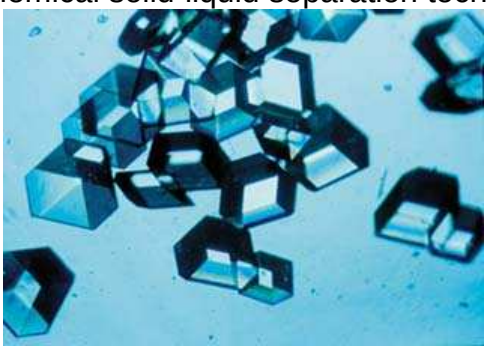


Figure 2-1: crystals

Crystallization may be used to:
•Prepare well formed crystals.
•Purify an impure sample of a solid.
•Separate the components of a mixture.

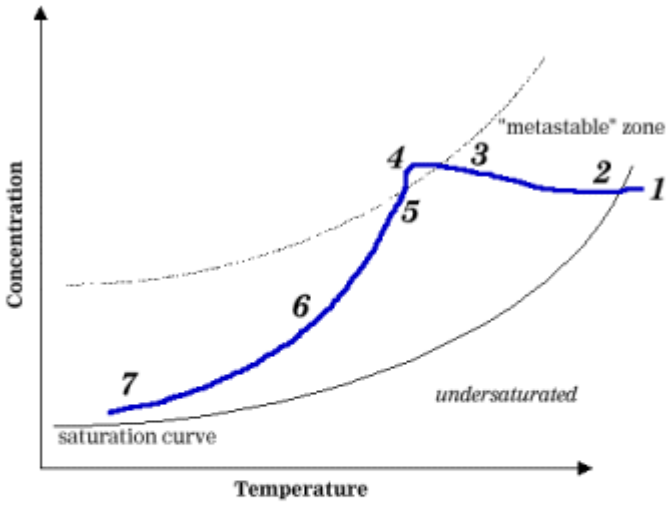
Notion:

•**Solubility**:
The maximum amount of the solute that can be dissolved in 100 grams of the solvent at a given temperature is termed its solubility at that temperature.
•**Saturated Solution**:
A solution in which more of the solute cannot be dissolved at a given temperature is called the saturated solution.

For crystallization to occur the solution most often is supersaturated. The solution should contain more solute molecules or ions than it would under ordinary conditions. This can be achieved by various methods: solvent evaporation, cooling, and chemical reaction, being the most common ones used in industrial practice.
Crystallization can be divided into two stages:

Primary nucleation is the nucleation of a new crystal on some seed or impurity or surface irregularity. On an industrial scale, a large supersaturation driving force is necessary to initiate primary nucleation. The initiation of primary nucleation via this

driving force is not fully understood which makes it difficult to model (experiments are the best guide). Usually, the instantaneous formation of many nuclei can be observed “crashing out” of the solution. You can think of the supersaturation driving force as being created by a combination of high solute concentration and rapid cooling. In the salt example, cooling will be gradual so we need to provide a “seed” for the crystals to grow on. In continuous crystallization, once primary nucleation has begun, the crystal size distribution begins to take shape. Think about our salty water, as you look at Figure 2-2 describing the progression of crystallization.



1 Feed location, undersaturated
2 Solution cools to saturation
3 Enter "metastable" zone, nucleation begins
4 Rapid nucleation
5 Concentration decreases with crystal growth
6 Crystal growth during main cooling cycle
7 Exit location, supersaturated

Figure 2-2: Progression of Crystallization

Secondary crystallization requires existing crystals to perpetuate crystal growth and is the main stage in crystallization resulting in the ‘mass production’ of crystals. In this phase of crystallization, crystal growth is initiated with contact. The contact can be between the solution and other crystals, a mixer blade, a pipe, a vessel wall, etc. This phase of crystallization occurs at lower supersaturation (than primary nucleation) where crystal growth is optimal. Again, no complete theory is available to model secondary nucleation and its behavior can only be anticipated by experimentation. Mathematic relationships do exist to correlate experimental data. However, correlating experimental data to model crystallization is time consuming and often considered extreme for batch operations, but can easily be justified for continuous processes where larger capital expenditures are necessary. For batch operations, only preliminary data measurements are truly necessary.

2.2 Monitoring of particle size distribution (PSD):

The particles in adipic acid are seldom all of the same size and they often have varying shapes. Describing the size and shape is therefore a significant problem.

The particle size may also vary over quite a wide range. It is not unusual for the particle of suspension produced in a grinding operation, for example, to vary by a factor of 100 from the smallest to the largest size. To describe such situations we normally break the range up into a number of classes and try to find out how many particles are in each size range.

This range is called particle size distribution (PSD), and it can be represented in the form of Table or a histogram (see Figure 2-3)

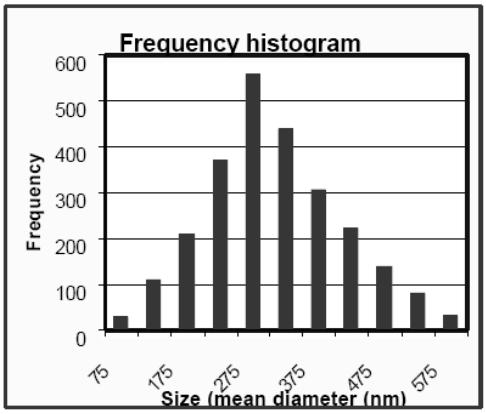


Figure 2-3 A typical PSD in the form of a histogram

PSD measurement, however, has been based on sampling and off-line analysis by different methods. However, the growing demand for improved product quality and process control has motivated the development of a number of sensors for monitoring process variables related to key process information. The figure 2-4 shows the Comparison of off-line optical microscopy image (left) of lactose crystals with in-situ particle vision measurement (right) during process operations

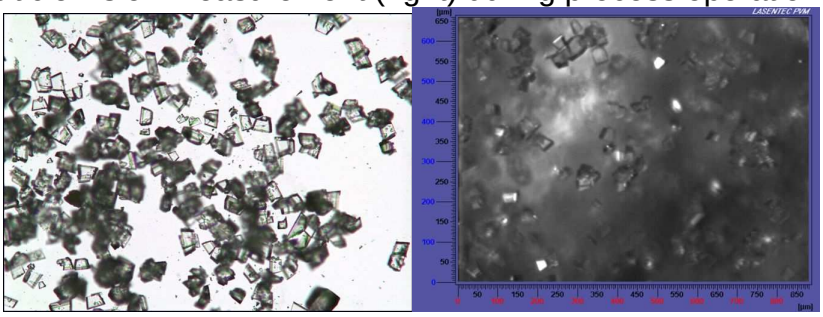


Figure 2-4 Comparison

In-line monitoring of PSD, which is closely related to product quality and process productivity, has therefore become an important aspect of crystallization research. A review of the literature in the field [1-4] shows that various in- line particle sizes are used. Laser light scattering methods are the favorite choice in many applications.

2.3 Lasentec® Focused Beam Reflectance Measurement

Recently, crystallization practitioners have become increasingly interested in the Focused Beam Reflectance Method FBRM○,R (Lasentec, 15524 NE 95th Srewwr, Redmond, WA 98052, USA), which allows on-line and in-situ measurements even in systems with high solid concentration.

The installed device of the FBRM is show in Figure 1-5. An FBRM probe is inserted into a flowing medium of any concentration or viscosity. A laser beam is projected through the sapphire window of the FBRM probe and highly focused just at the window surface. This focused beam is then moved so it follows a path around the circumference of the probe window. The focused beam is moving at a high rate of speed (2m/sec to 6m/sec, depending on the application) so that particle motion is insignificant to the measurement. (Figure 2-5)

As particles pass by the window surface, the focused beam will intersect the edge of a particle (figure 2-6, 2-7). The particle will then begin to backscatter laser light. The particle will continue to backscatter the light until the focused beam has reached the particle's opposite edge. The backscatter is collected by the FBRM optics and converted into an electronic signal.

Figure 2-5 Lasentec FBRM

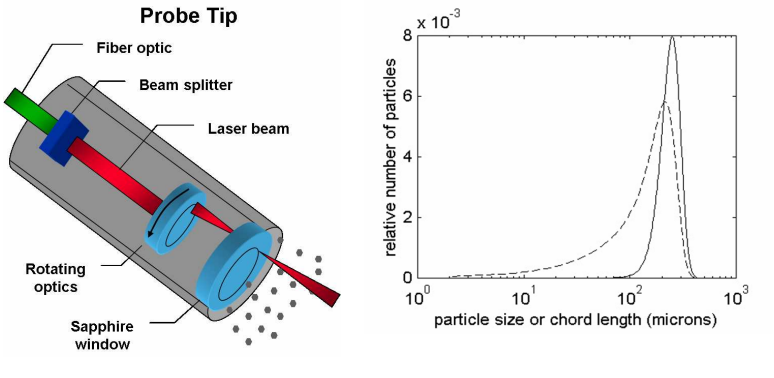
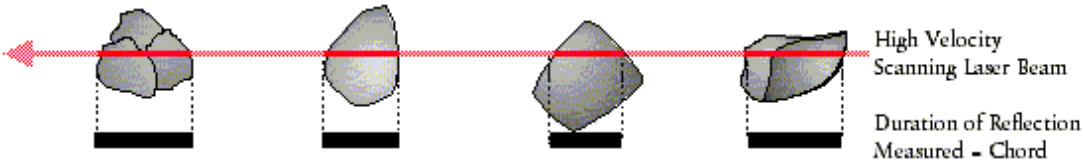
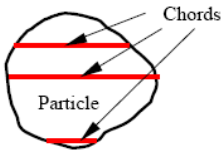


Figure 2-6 Schematic of the FBRM probe for measuring particle characteristics during process operations and Chord length distribution from FBRM (- - -) and particle size distribution (—) for polymer beads during a suspension polymerization reaction



(a)



(b)

Figure 2-7 (a) High velocity scanning laser beam and duration of reflection measured-chord, (b) possibility of cord length of particle. [Source: www.lasentec.com.]

The principle is recommended in other work such as [5, 6].

A disadvantage of the instrument at present is that the results are presented in the form of chord length distributions (CLDs) rather than particles-size distributions (PSDs). The CLDs are distributions of chord rates, reporting how frequently chords of a certain length are detected. The CLDs are reproducible characterizations of the particulate system, depending on the PSD and the shapes of the particles simultaneously. However, PSDs would be useful, for comparison with measurements from other instruments, and for a priori predictions of the systems properties.

2.4 Photomicroscopy, granulometry by Image Analysis

Photomicroscopy views crystals on-line and Image analysis requires quality images from photomicroscopes using the microscope, digital camera and image analyzing software. With simple algorithms and readily available hardware, image analysis is clearly the technique of choice for monitoring and controlling particle size and habit in crystallization processes. Today, acquisition is usually done using a CCD (charge-coupled device) camera mounted in the optical path of the microscope (figure 2-8). The camera may be full colour or monochrome.

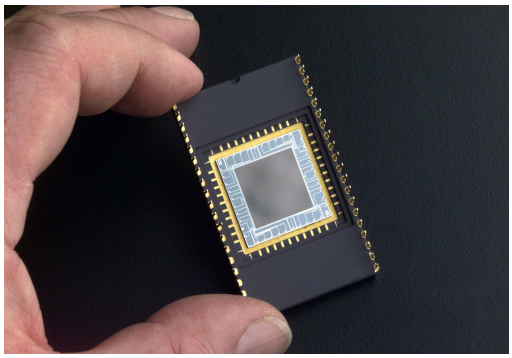


Figure 2-8 a specially developed CCD used for ultraviolet imaging in a wire bonded package. [Source of www. wikipedia.org]

Very often, very high resolution cameras are employed to gain as much direct information as possible. Often digital cameras used for this application provide pixel intensity data to a resolution of 12-16 bits, much higher than is used in consumer imaging products.

Now we have many ways to define the size of particle. For the cycle, the size is his diameter. But many of case, the particles are not cycles. So we often use equivalent spherical diameter (ESD), in the figure 2-9, the dFmax represent the largest distance of the two parallel lines on the opposite edges of the particle. And the dFmin represent the smallest distance of that. The dA is the diameter of the spherical who' surface is same as the particle.[

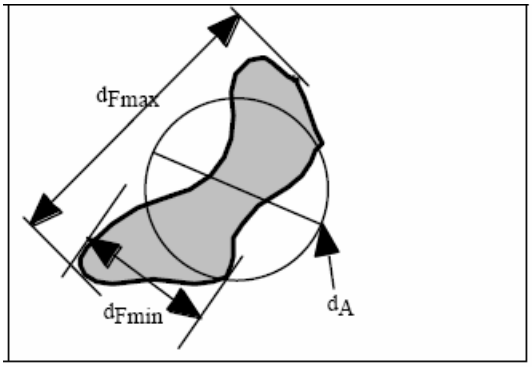


Figure 2-9 the different diameter of particle

The results of image analysis may be presented by different way, for example: the distribution in frequent and the distribution in cumuli (figure 2-10). Some software of measurement can give the distribution vs. the number or the mass

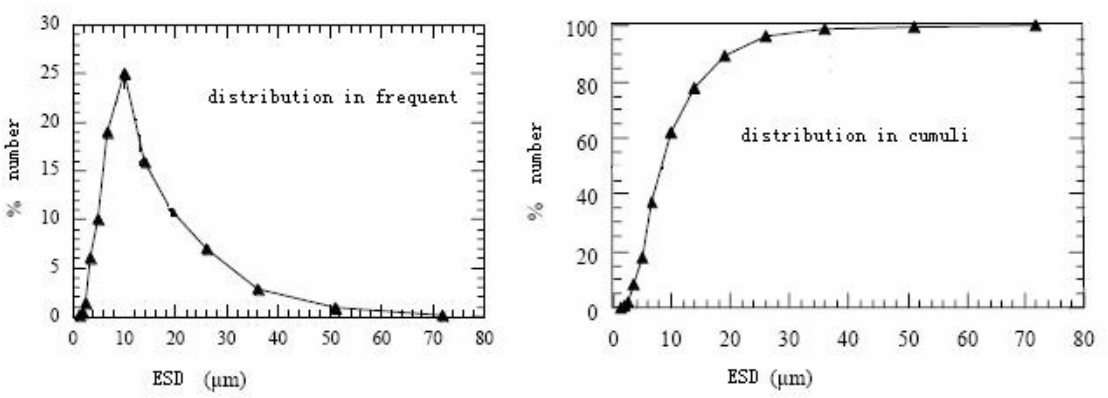


Figure 2-10 The 2 distribution du size

Conclusion: the compare between FBRM and image analysis (IA):

FBRM:

•Benefits:

- provides a precise and sensitive measurement that allows the user to quantify
- On-line and in-situ measurement, track particles as they exist in process.
- Utilize in-situ data to understand and optimize the dynamic process.

•Disadvantages:

- The results are presented in the form of chord length distributions (CLDs) rather than particles-size distributions (PSDs).
- The CLDs are reproducible characterizations of the particulate system, depending on the PSD and the shapes of the particles simultaneously.

IA:

•Benefits:

- Off-line measurements. The results are presented in the form of PSD.
- Gain as much direct information as possible.

•Disadvantages:

- Does not provide process dynamics (not effective for modeling and control).
- Sampling errors (unrepresentative measurements).
- Waste material.

•

CHAPTER 3 THE ARTIFICAL NEURAL NETWORK (ANN) AND THE APPLICAION OF NN TOOLBOX IN MATLAB

3.1 The background and history of ANN [8]

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.
Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

Many important advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. These pioneers were able to develop convincing technology which surpassed the limitations identified by Minsky and Papert. Minsky and Papert, published a book (in 1969) in which they summed up a general feeling of frustration (against neural networks) among researchers, and was thus accepted by most without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits. But the technology available at that time did not allow them to do too much.

3.2 Why use neural networks? [8]

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an “expert” in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer “what if” questions.

Other advantages include:
- *Adaptive learning*: An ability to learn how to do tasks based on the data given for training or initial experience.

- *Self-Organization*: An ANN can create its own organization or representation of the information it receives during learning time.

- *Real Time Operation*: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

- *Fault Tolerance via Redundant Information Coding*: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

3.3 Human and Artificial Neurons - investigating the similarities [8]

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons (figure 3-1). When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes (figure 3-2).

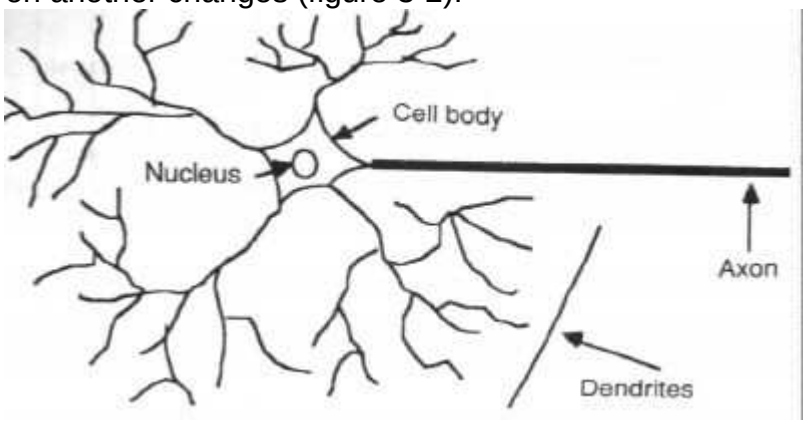


Figure 3-1 Components of a neuron [source of [www.doc.ic.ac.uk]

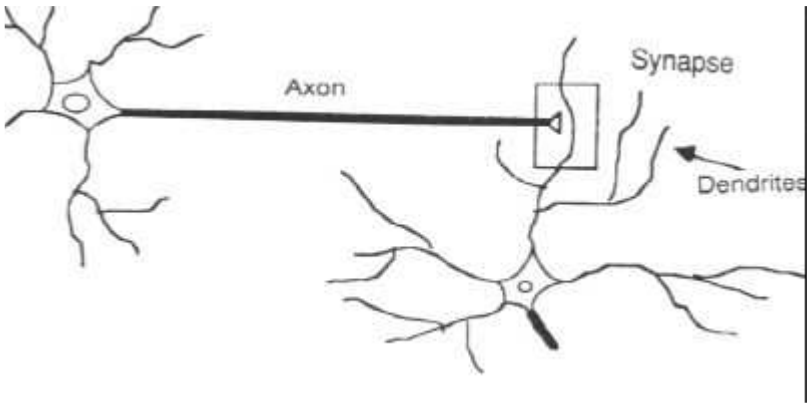


Figure 3-2 The synapse

We conduct these neural networks by first trying to deduce the essential features of neurons and their interconnections. We then typically program a computer to simulate these features. However because our knowledge of neurons is incomplete and our computing power is limited, our models are necessarily gross idealizations of real networks of neurons (figure 3-3).

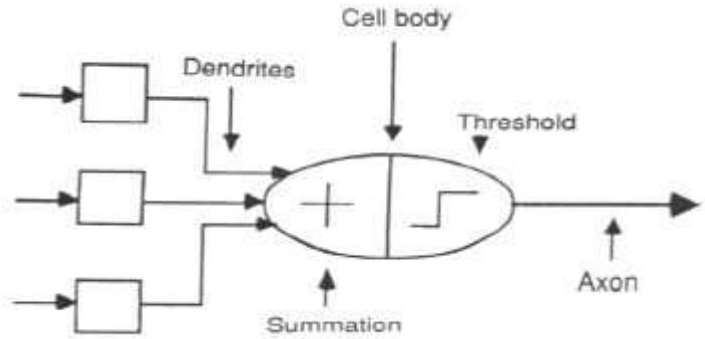


Figure 3-3 The neuron model

3.4 A simple neuron [8]

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not (figure 3-4).

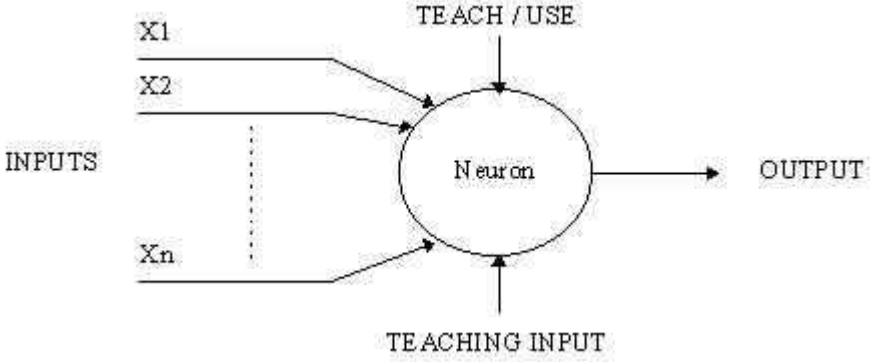
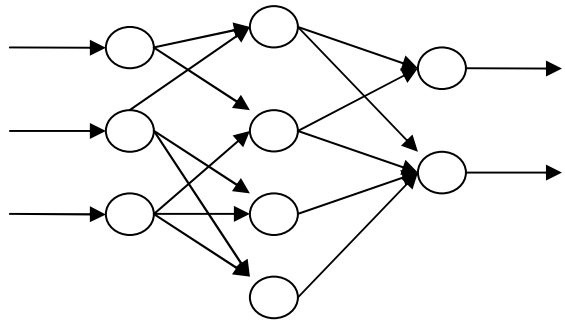


Figure 3-4 A simple neuron

3.5 Architecture of ANN

Feed-forward network

Feed-forward ANNs (figure 3-5) allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.



Input layer Hidden layer Output

Figure 3-5 The feed-forward network

Network layers:[8]
The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units. (See Figure3-5)

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- The behaviors of the output units depend on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

Perceptron:
The most influential work on neural nets in the 60's went under the heading of 'perceptrons' a term coined by Frank Rosenblatt. The figure 3-6 shows you the model of single layer perceptron. It is a kind of binary classifier that maps its input x (a real-valued vector in the simplest case) to an output value f(x) calculated as

$$f(x) = \langle w, x \rangle + b$$

Where w is a vector of weights and $\langle \cdot, \cdot \rangle$ denotes dot product. (We use the dot product as we are computing a weighted sum.) The sign of f(x) is used to classify x as either a positive or a negative instance. Since the inputs are fed directly to the output via the weights, the perceptron can be considered the simplest kind of feed-forward network. [From Wikipedia]

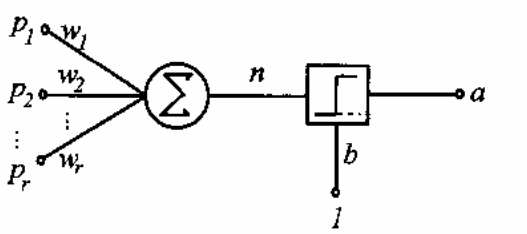


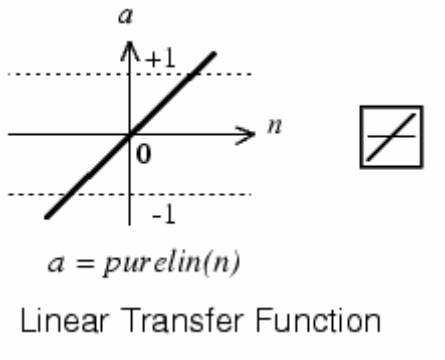
Figure 3-6 The model of single layer perceptron

In 1969 Minsky and Papert wrote a book in which they described the limitations of single layer Perceptrons. The impact that the book had was tremendous and caused a lot of neural network researchers to loose their interest. The book was very well written and showed mathematically that single layer perceptrons could not do some basic pattern recognition operations like determining the parity of a shape or determining whether a shape is connected or not. What they did not realize, until the 80's, is that given the appropriate training, multilevel perceptrons (MLP) can do these operations.

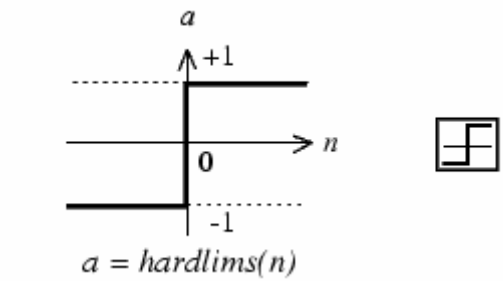
Transfer functions:

The behaviors of an ANN (Artificial Neural Network) depend on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories (figure 3-7):

Linear (or ramp)



Threshold



Sigmoid

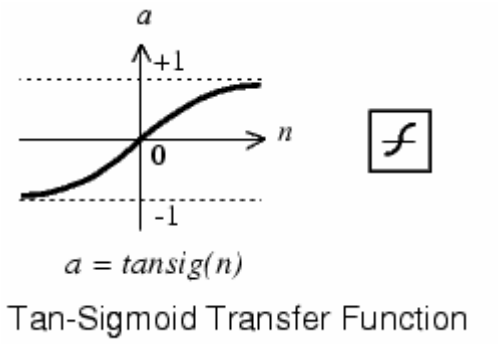


Figure 3-7 Three different transfer functions

- For linear units, the output activity is proportional to the total weighted output.
- For threshold units, the output are set at one of two levels, depending on whether the total input is greater than or less than some threshold value.
- For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another (see figure 2-6), and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

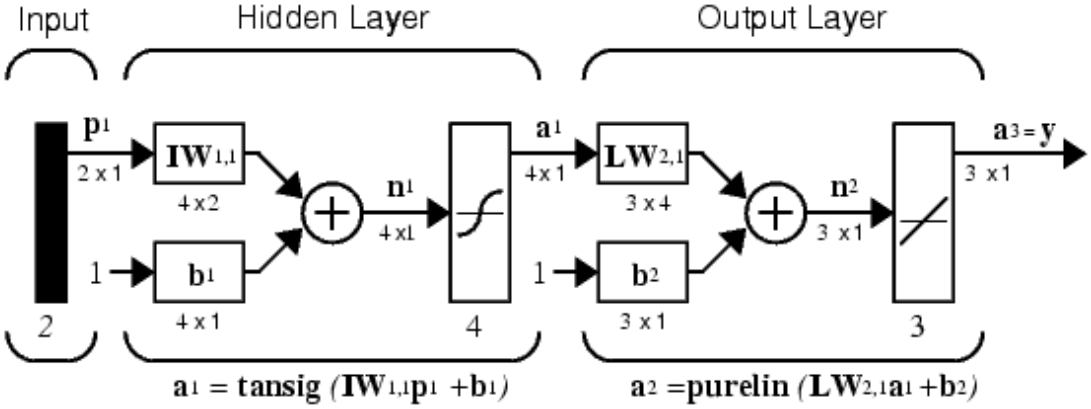


Figure 3-8 the three layer network

We can teach a three-layer network (figure 3-8) to perform a particular task by using the following procedure:

- We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
- We determine how closely the actual output of the network matches the desired output.
- We change the weight of each connection so that the network produces a better approximation of the desired output.

3.6 The Supervised learning algorithm: [9, 10]

All learning methods used for adaptive neural networks can be classified into two major categories: Supervised learning and unsupervised learning.

Supervised learning which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. Important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.

For unsupervised learning the networks are presented with only the input samples, and samples are grouped into classes which are self-similar. Network trained in this fashion are called self-organizing networks.

3.7 The neural network toolbox of Matlab

The Neural Network Toolbox extends the MATLAB® computing environment to provide tools for the design, implementation, visualization, and simulation of neural networks. Neural networks are uniquely powerful tools in applications where formal analysis would be difficult or impossible, such as pattern recognition and nonlinear system identification and control. The Neural Network Toolbox provides comprehensive support for many proven network paradigms, as well as a graphical user interface that allows you to design and manage your networks. The toolbox's modular, open, and extensible design simplifies the creation of customized functions and networks.

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. You can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are needed to train a network (figure 3-9).

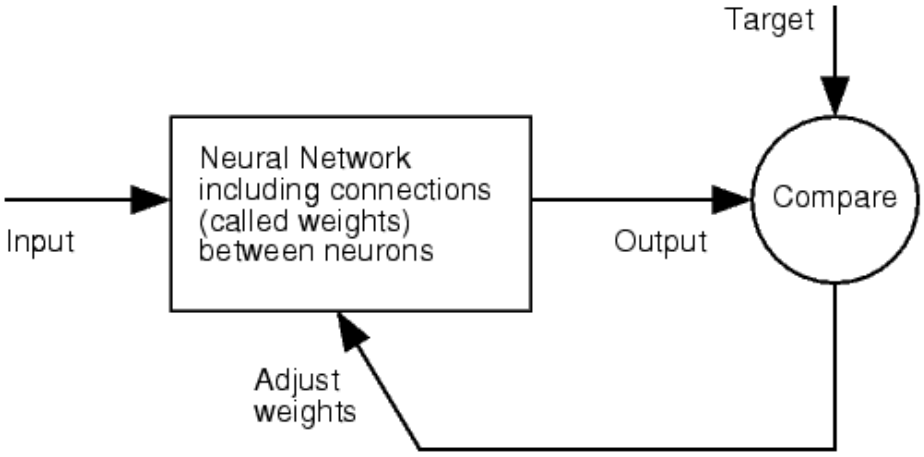


Figure 3-9 The typical structure of neural network

Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, and speech, vision, and control systems.

Today neural networks can be trained to solve problems that are difficult for conventional computers or human beings. Throughout the toolbox emphasis is placed on neural network paradigms that build up to or are themselves used in engineering, financial, and other practical applications.

The next schema (figure 3-10) recapitulates how we can carry out our program in Matlab and its main functions.

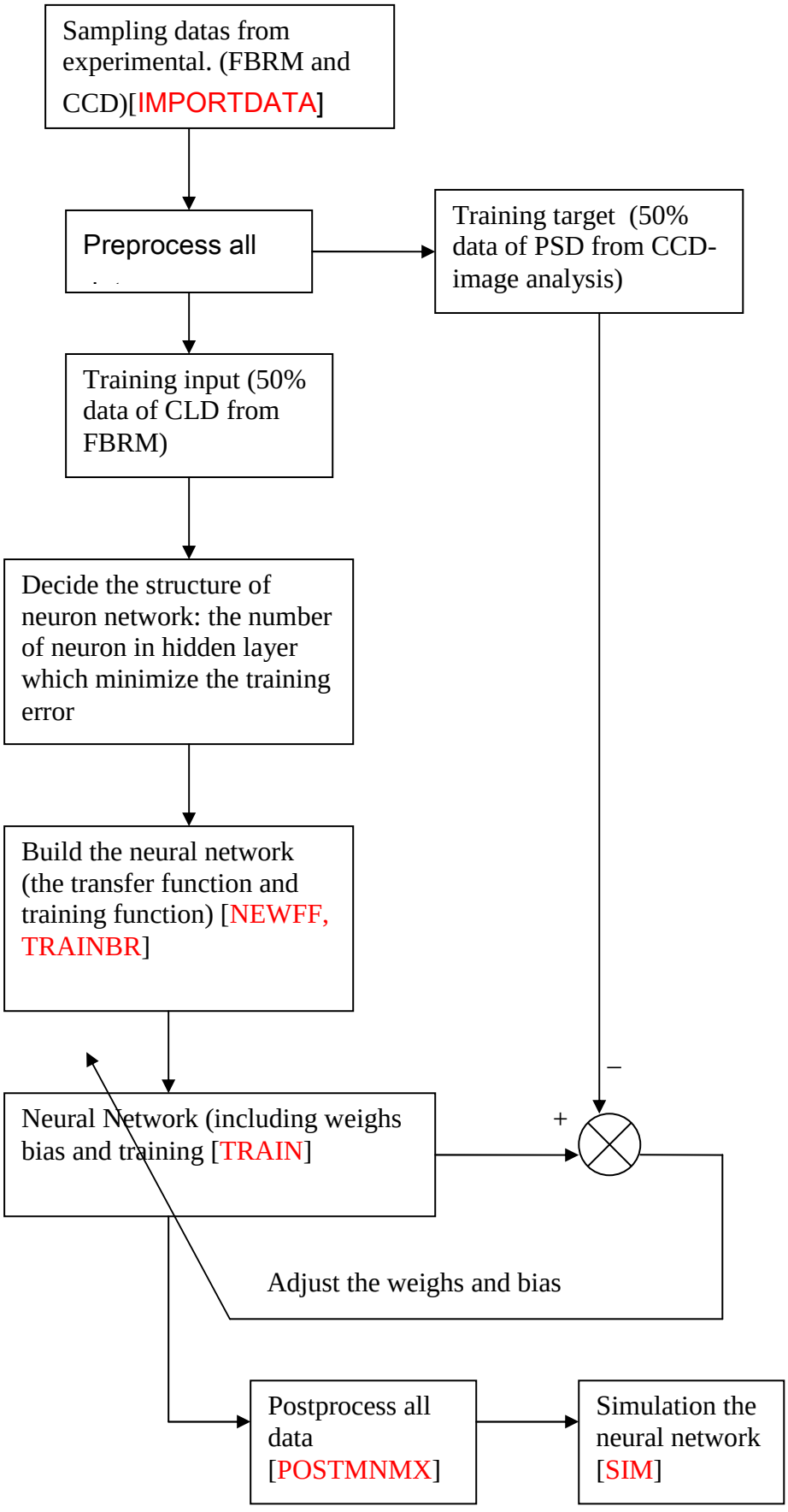


Figure 3-10 How the programmer is carried in Matlab

Table 1: the main function of Matlab NN toolbox:

Name of function in Matlab	Description
IMPORTDATA	Load data from disk file.
PREMNMX	Preprocess data so that minimum is -1 and maximum is 1
NEWFF	Create a feed-forward backpropagation network
TRAINBR	Bayesian regularization backpropagation
TRAIN	Train a neural network according the training function.
POSTMNMX	Postprocess data that has been preprocessed by premnmx
SIM	Simulate a neural network

In the appendix E There are some useful glossaries of the NN Toolbox.

CHAPTER 4 THE EXPERIMENTAL MANIPULATION OF CRYSTALLIZATION PROCESS

4.1 The principal equipment and materiel:

Adipic Acid:

Adipic acid is the common name of 1,6-hexanedioic acid (figure 4-1), a chemical compound of the class of carboxylic acids. It is a white crystalline powder appearing as an acid in aqueous circumstances, though it is not highly soluble.

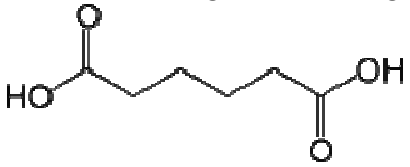


Figure 4-1 the chemical structure of Adipic acid

Table 2. The characteristics of samples used in the experiments

General		Properties	
Common name	adipic acid	Density and phase	1.36 g/cm3
Systematic name	hexanedioic acid	Solubility in water	slightly soluble
Other names	butane-1,4-dicarboxylic acid	Other solvents ethanol, acetone	soluble
Molecular formula	C6O4H10	Melting point	153 ℃ (426 K)
SMILES	OC(=O)CCCCC(=O)O	Boiling point	337 ℃ (610 K)
Molar mass	146.14 g/mol		
Appearance	White crystals		
CAS number	[124-04-9]		

Lasentec® FBRM Modèle D600L-C22-K

Software of acquisition: Lasentec FBRM Data Acquisition Control Interface 6.0
Software of exploitation: Lasentec FBRM Data Review

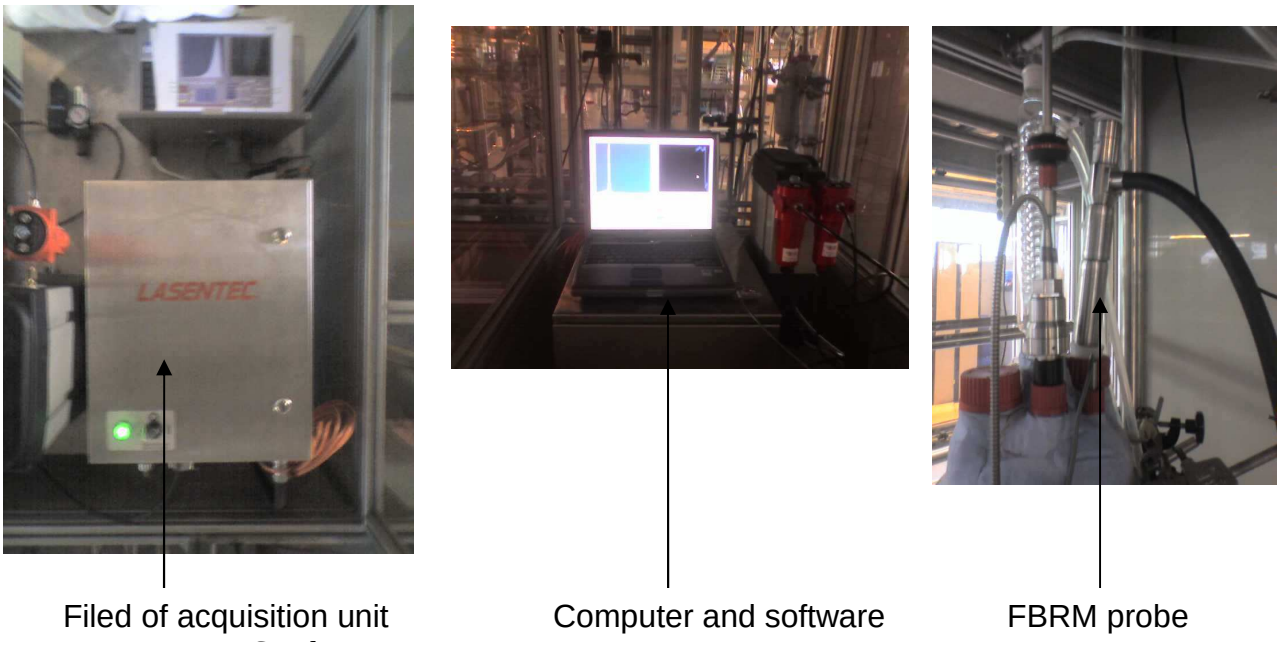


Figure 4-2 the equipment FBRM

Camera:
We used the optic camera. The lamp is with xenon guaranty that the observable zone should be clear. The real size of the image is 1mm.

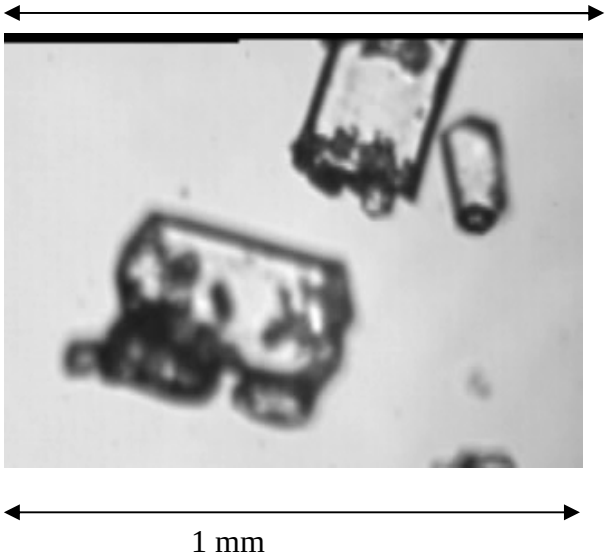


Figure 4-3 the image cached by camera

4.2 The Experimental manipulations

The laboratory experiments were carried out by the students of chemical department of IUT. All the manipulations utilized the adipc acid whose characteristics had been talked before. The next schema (figure 4-4) indicates all equipments in the experimental manipulations:

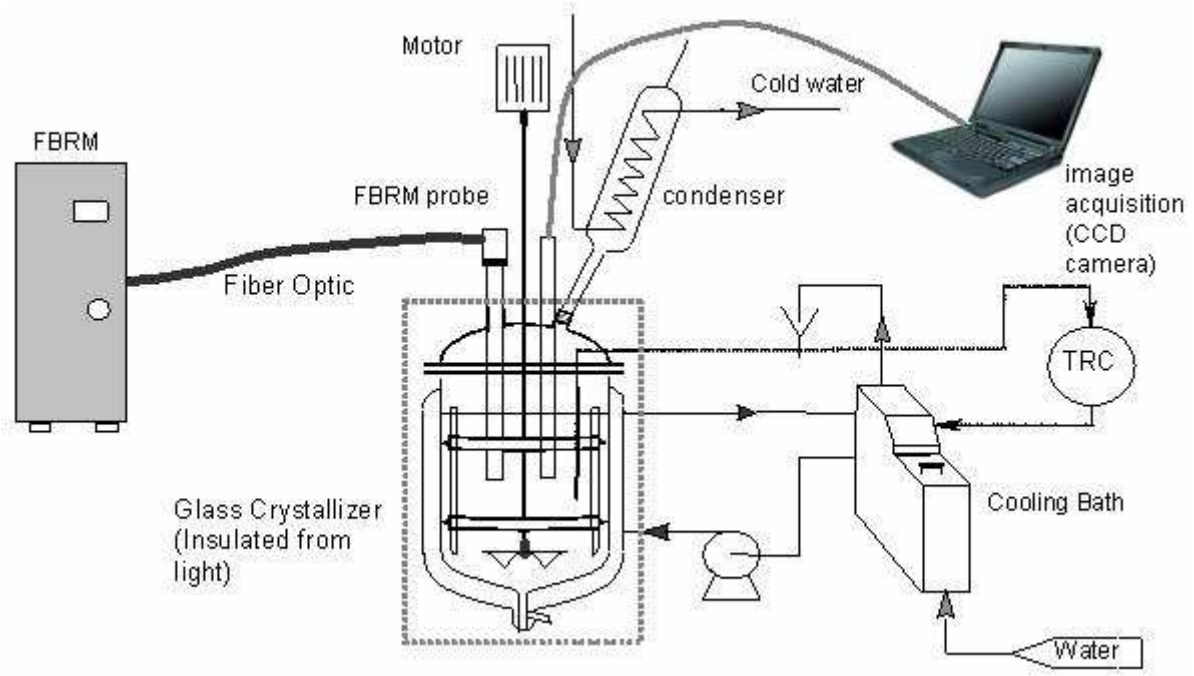


Figure 4-4 the schema of experimental equipments

The solubility of adipc acid is 7.03%, it means we can resolve 7.03g adipc acid in 100g water. Because our container is 2 liters, so m_{Ac} = 7,03 x 2000 /100

$$m_{Ac}=140,6g$$

The change of each experiment was the ramps of temperate. The manipulations were composed of 3 steps: first, cool down the solution with the ramp of temperature α1 (-7℃/h ~- 50℃/h), after the phenomenon of nucle ation, keep hold of this temperature during the time d (0~900s), finally, cooling down again with the ramp of temperatureα2. (-20℃/h ~- 50℃/h). The next figure 4-5 shows y ou the example of changing temperature and the number of nucleation in per second, also next table shows the ramps of each manipulation and the keeping time between 2 cool down. All manipulations begin with the temperature T(beginning)=63°C, and end of crystallization when the temperature becomes T(ending)=13°C.The values are fixed by our tutor.

Table 3: summary of changing temperature in each manipulation:

Manip n°	α 1	d	α 2
	(°C/h)	(s)	(°C/h)
1	?	0	?
2	-50	900	-50
3	-30	900	-30
4	-30	300	-30
5	-15	900	-20
6	-15	900	-20
7	?	300	-40
8	-30	0	?
9	-15	0	-20

10	-7,5	0	-20

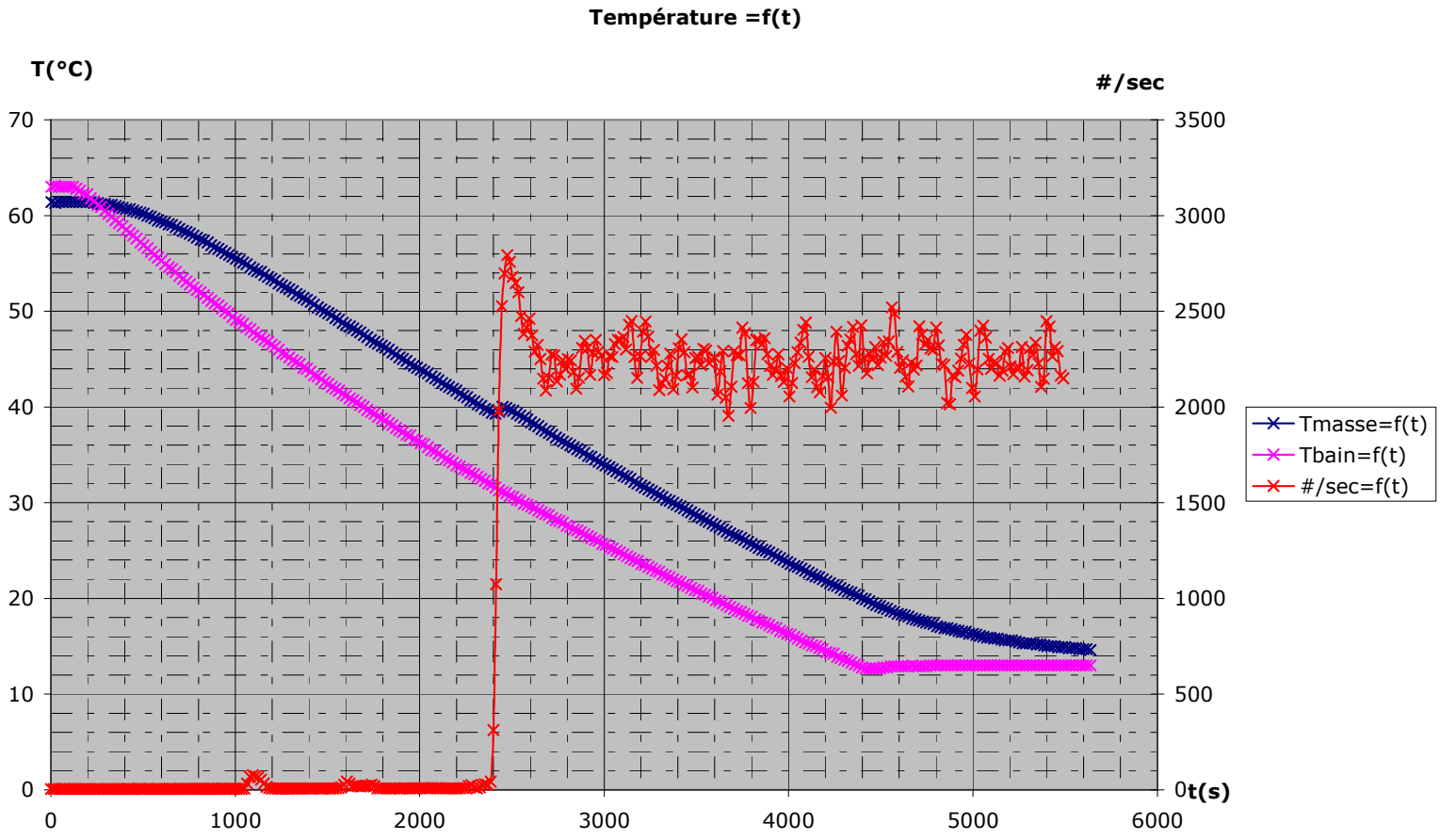
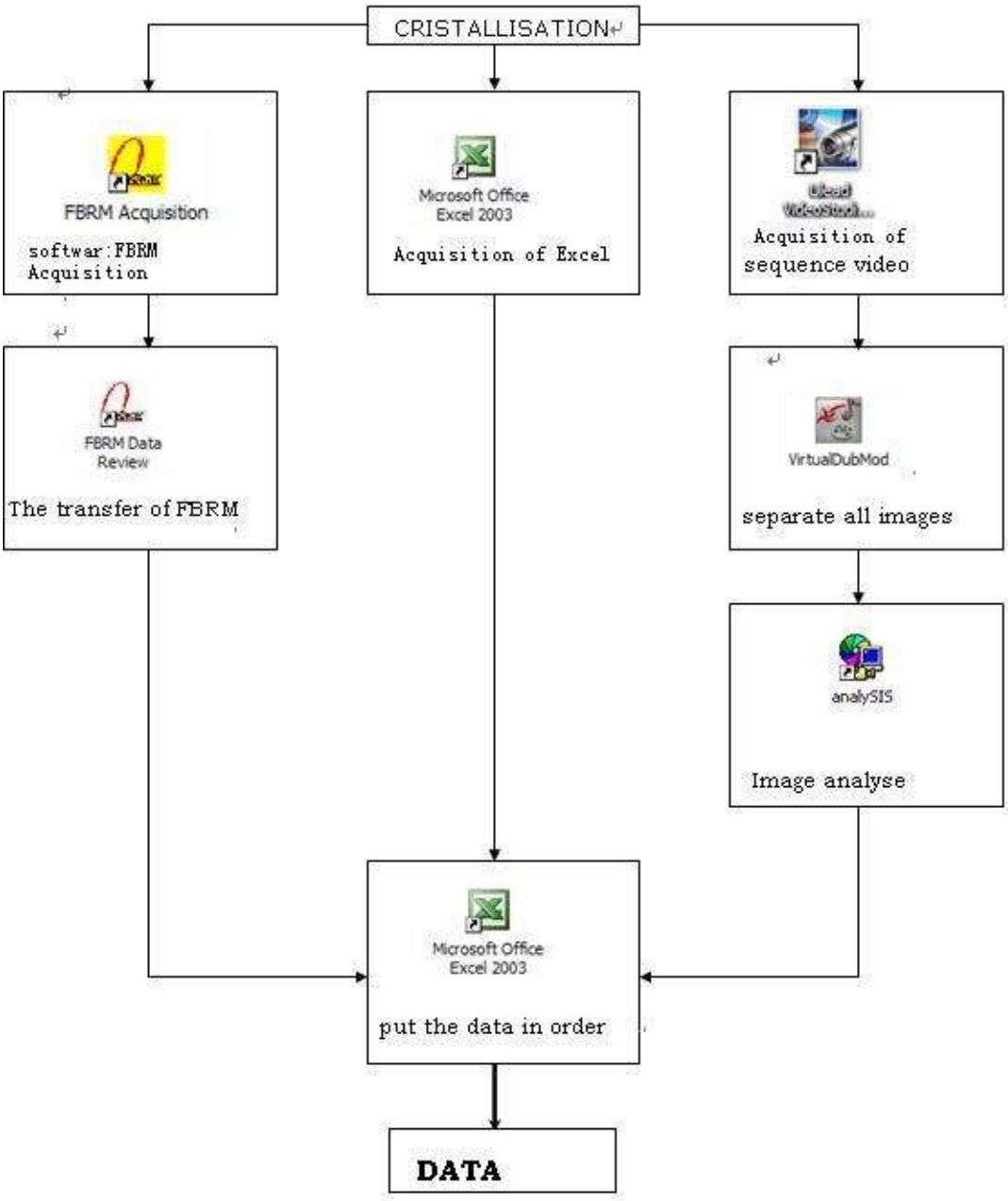


Figure 4-5 the change of temperature and the number of particle

After all of manipulations, we have the results that have same starting point of time for all of acquisition; we will take them to analyze the size distribution correspond different time of our crystallization.

The next schema recapitulate how can we botany all data after the manipulation:



We have all necessaries acquisition for ANN:
•CLD=f(t);
•PSD=f(t);
•Number of particle = f(t)
•The class of size: 40μm

In fact, the FBRM give us the class of size 20μm , in my work, as the request of our tutor Mr. Fevotte, we will consider it 40μm, the same for image analysis. I have to talk here the technique limitation of video, in order to have enough image, we have to take a video 35s duration, our video can take 2 image/s. And according to the limit number particle in one image, totally, we may have about 200~300 particle in one time.

Finally, we have 10 manipulations and 49 snapshots of CLD and PSD.

CHAPTER 5 THE DESIGN OF NEURAL NETWORK

5.1 The introduction du model:

In my work, the method which is based on fitting a neural network (NN) software sensor is proposed for transforming an FBRM sensor into a volume-based PSD monitoring sensor. The information is provided by the FBRM device and the experimental data for fitting and validating the model were obtained in chemical laboratory as I had mentioned before.

For each measurement, the FBRM sensor provided data on the total count of particles, as well as counts per class of chord length. These data were then used in fitting an NN model for estimating the PSD of particles in the suspension.

The adopted reference values for PSD the particle samples were measured by CCD.
Neural network models have been successfully applied by the authors in a number of complex systems in different fields of interest, such as the application to a continuous pulp digester [10], the classification of NIR spectral data [11], the model of the river flow rate [12] or in chemical process control [13]

5.2 The method: backpropagation algorithm:

The methodology for fitting the NN model was based on backpropagation (BP) algorithm. It is a supervised learning technique used for training artificial neural networks. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for “backwards propagation of errors”. Backpropagation requires that the transfer function used by the artificial neurons be differentiable.

The summary of the technique is as follows:

- Present a training sample to the neural network.
- Compare the network’s output to the desired output from that sample. Calculate the error in each output neuron.
- For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error. An internal gradient method is used.
- Adjust the weights of each neuron to lower the local error.
- Assign “blame” for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
- Repeat the steps above on the neurons at the previous level, using each one’s “blame” as its error.

As the algorithm’s name implies, the errors (and therefore the learning) propagate backwards from the output nodes to the inner nodes. So technically speaking, backpropagation is used to calculate the gradient of the error of the network with respect to the network’s modifiable weights. This gradient is almost always then used in a simple stochastic gradient descent algorithm to find weights that minimize the error. Often the term “backpropagation” is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in

stochastic gradient descent. Backpropagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

The backpropagation algorithm for calculating a gradient has been rediscovered a number of times, and is a special case of a more general technique called automatic differentiation in the reverse accumulation mode.
More detail in the mentioned papers [9].

5.3 The NN fitting:

In the present case a feedforward NN model was used, consisting of three layers: the input layer, the intermediate (or hidden) layer, and the output layer. The number of neurons in the input and output layers correspond to the number of input and output variable, respectively, about the question on how many neurons in the hidden layer: Classically, there are the two major strategies of dealing with this question:

Cross validation:
In this technique, we train networks with different neurons in hidden layer(for example we train 25 different networks ranging from say 6 neurons to 30 neurons in hidden layer). Now in each of the 25 networks, using cross validation (also known as early stopping), we obtain the best network. Now draw a graph of MSE (Mean square error) for TESTSET (y-axis) and number of neurons in hidden layer (x-axis). We may perceive a minimum in the graph.

Bayesian Regularization:
Here we use the “trainbr” algorithm as the training method. In this framework, the weights and biases of the network are assumed to be random variables with specified distributions. The regularization parameters are related to the unknown variances associated with these distributions. We can then estimate these parameters using statistical techniques.. Here the algorithm minimizes a combination of squared error and norm of the weights. When we use this the network prediction we will notice will be very smooth. Here by giving weightage to the norm of the weight/bias vector, we can afford to give an arbitrarily large number of neurons. For example, we give 20 neurons in hidden layer, the algorithm will tell you the effective number of parameters which will be different than 20.

The separation of data set:

The experimental data were first divided into two sets: the learning set, which is used in the fitting process and should contain the extreme values of all the input and output variables, and the test set, which is sued in validating the model. The error on the test set is monitored during the training process. The test error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the test set typically begins to rise. When the test error increases for a specified number of iterations (net.trainParam.max_fail), the training is stopped, and the weights and biases at the minimum of the test error are returned. There are two essential criteria to assuring that data are appropriate for training and validation, data quality and structural diversity. More specifically:

- The training data should be properly validated with respect to accuracy, mediating mechanisms and assay protocol, and
- The training data should provide a uniform representation over the structural diversity space to which the model will be applied

In order to make both set distribute homogeneously in sampling space, about one half of the data were allocated to the learning set and another half of data were used in validating the model. They could be selected in this way:
Training set = data(:,1:2:49)
Test set = data(:,2:2:49)

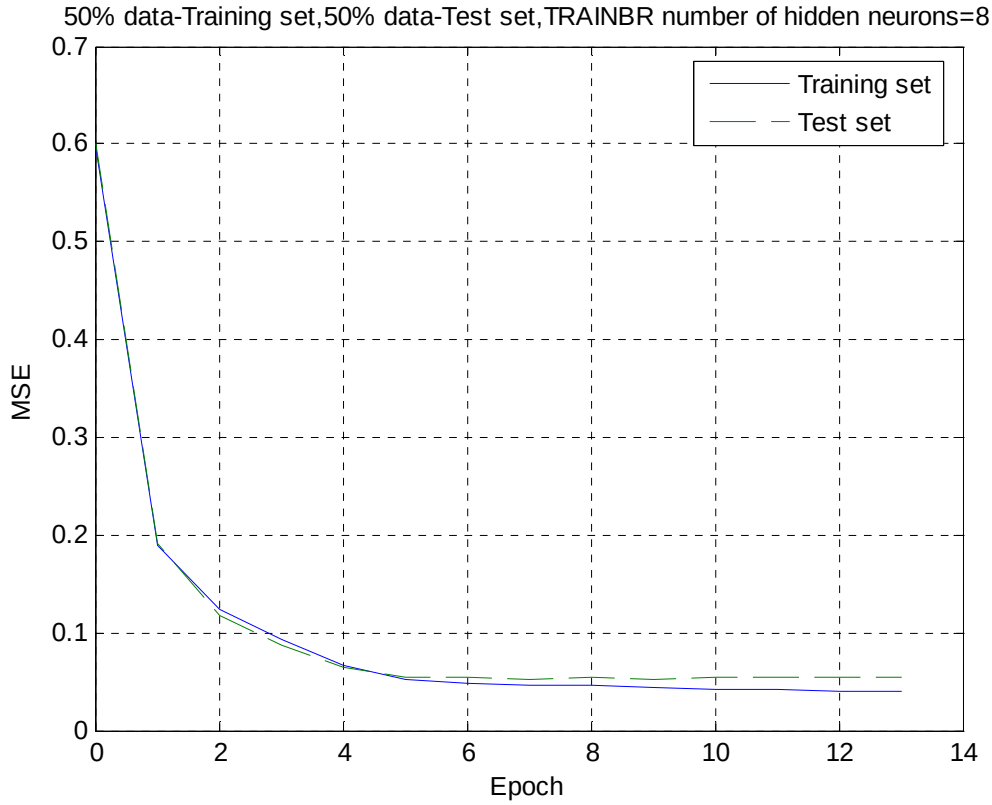


Figure 5-1 The design of data set (1)

In the figure 5-1, we can see the result of training by separating the data set to 1:1 as training set and test set. We can also compare it with the next case, figure 5-2, now the data set is separated to 2:1, it means 2/3 data as training set and 1/3 data as test set. It is worse than before.

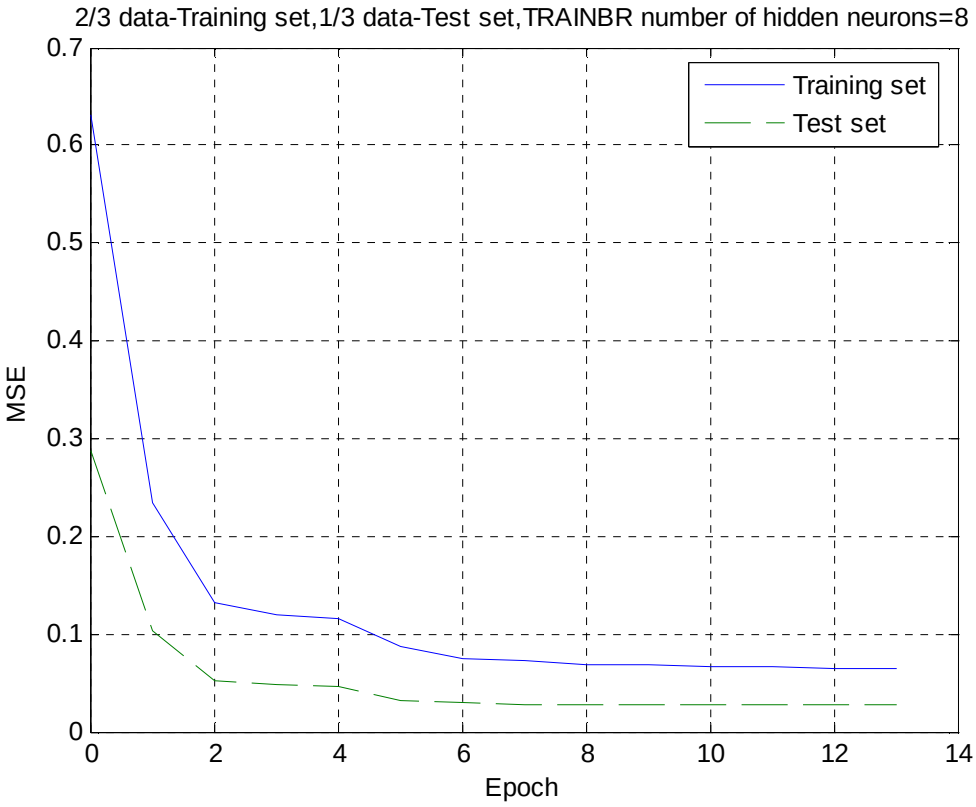


Figure 5-2 The design of data set (2)

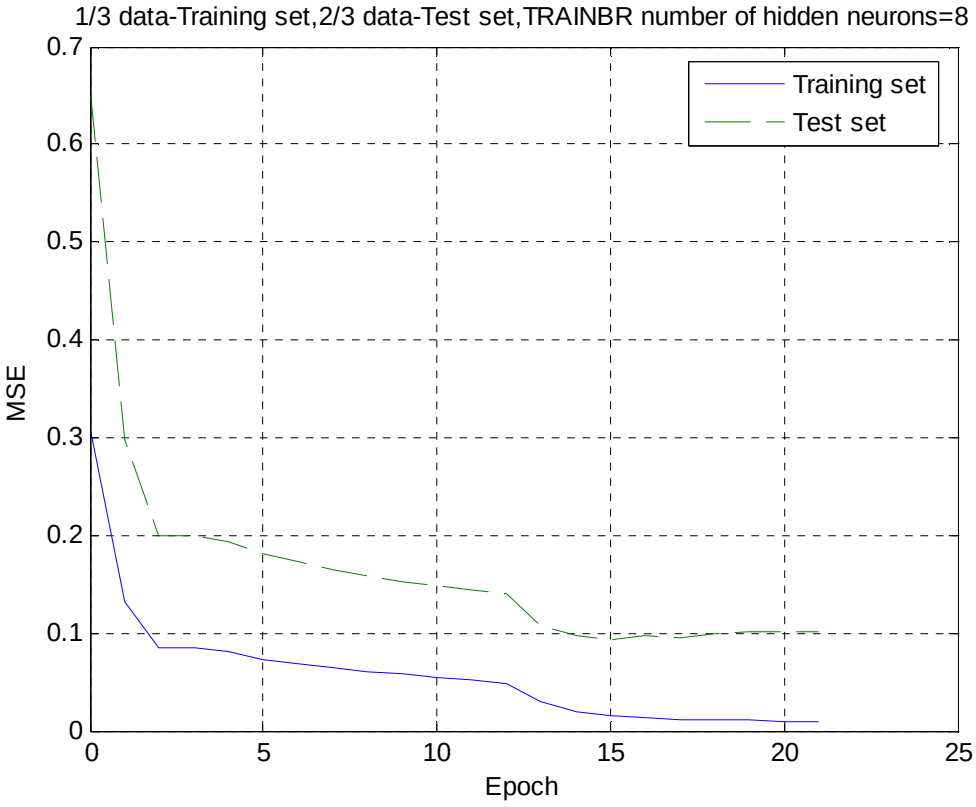


Figure 5-3 The design of data set (3)

In the next, we can see the first works well by separating in the way: 50% training set and 50% test set.

Others:
The fitting criterion was based on the minimum of the mean square error (MSE), which is defined as the mean of the sum of the squared difference between calculated and experimental values of each output variable.

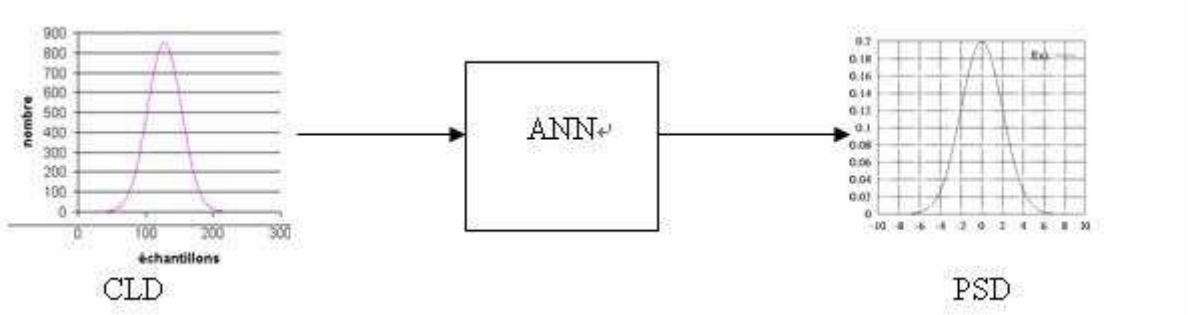
The prevention of over fitting problems was based on the monitoring of the amount of change in the value of the criterion as the number of neurons in hidden layer of the NN is changed. Based on this criterion, the simplest model (with the smallest number of neurons, which corresponds to the least number of parameters providing lower values of the error could be identified and selected.

5.4. Artificial neural network (ANN) design:

The NN fitting was based on the following input and output variables:

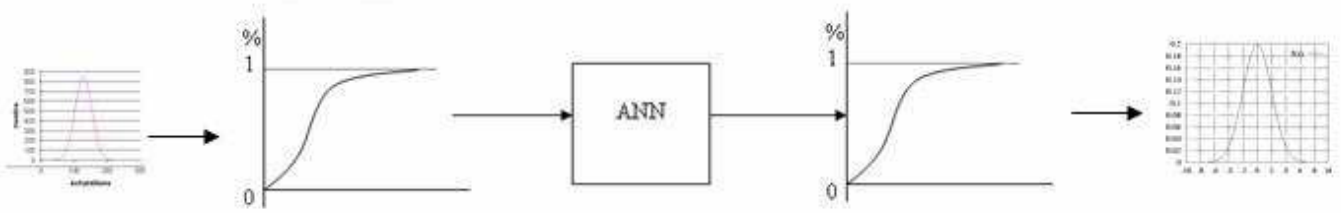
Model 1:

- Input variables: total counts/s, 11 classes of counts per second for chord lengths: CLD of each sample measured by FBRM
- Output variable: total counts/s, PSD (same 11 classes chosen as that of input), reference output values for the NN learning, each sample measured by image analysis.



Model 2:

- Input variables: percentage of cumulative distribution of chord lengths
- Output variable: percentage of cumulative difference of PSD



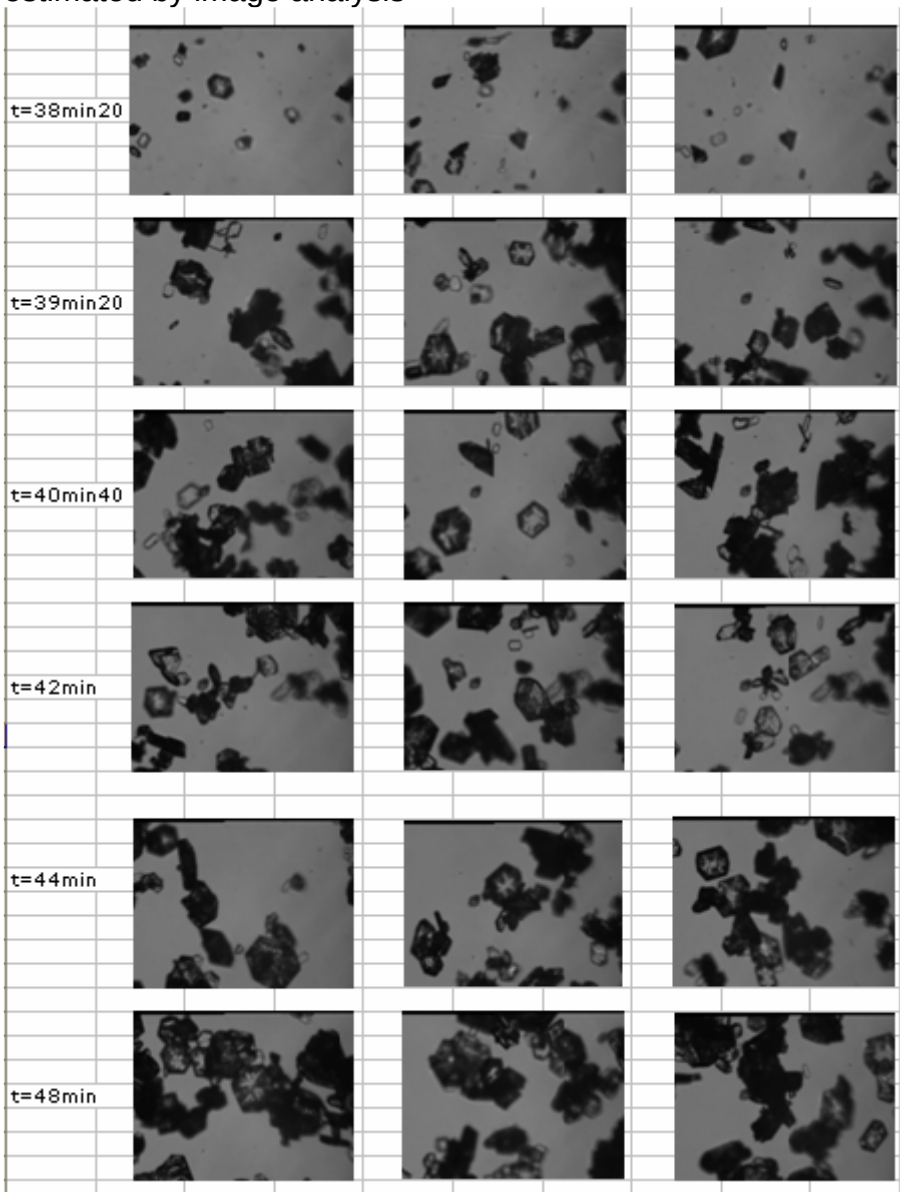
I must give you the notation of data, for the model 1; we have to normalize all of input variables and output variables. The goal of normalization is to create a set of relational tables that are free of redundant data and that can be consistently and correctly modified. Several functions in Matlab can be used to normalization. I will choose one so that all of data is actually done between -0.95 to 0.95 because of the transfer function is sensitive if the data is less that 0.95 and greater than -0.95.But it will be not effective for model 2 because of the data is between 0 to 1. (See the figure 3-7 sigmoid transfer function).

CHAPTER 6 RESULTS:

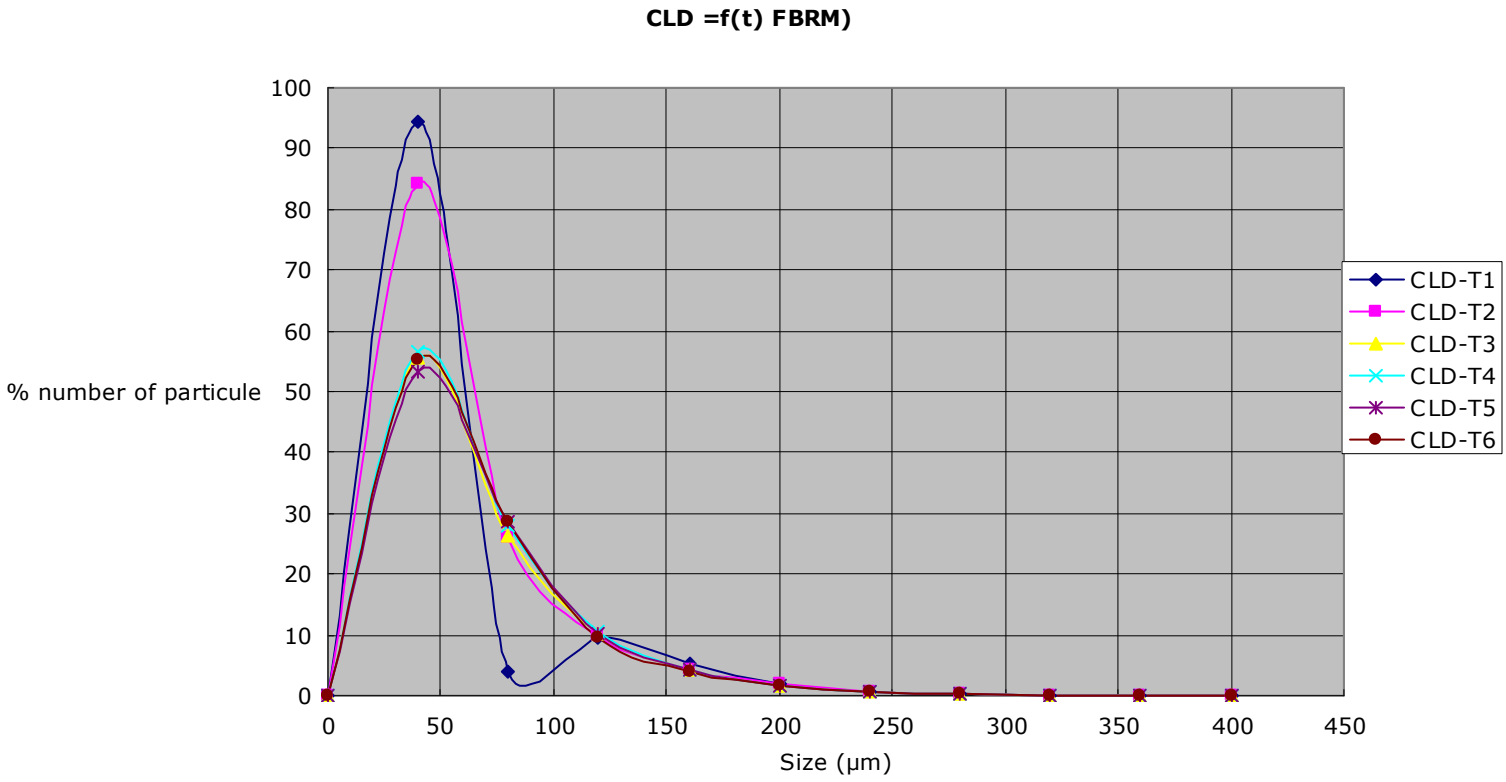
6.1 The behavior of the system:

Result from total 10 experiments gave the CLD provided by FBRM laser reflectance sensors. The PSD estimated by image analysis. I will show you the comportments of the first experiments; others will be showed in APPENDIX A.

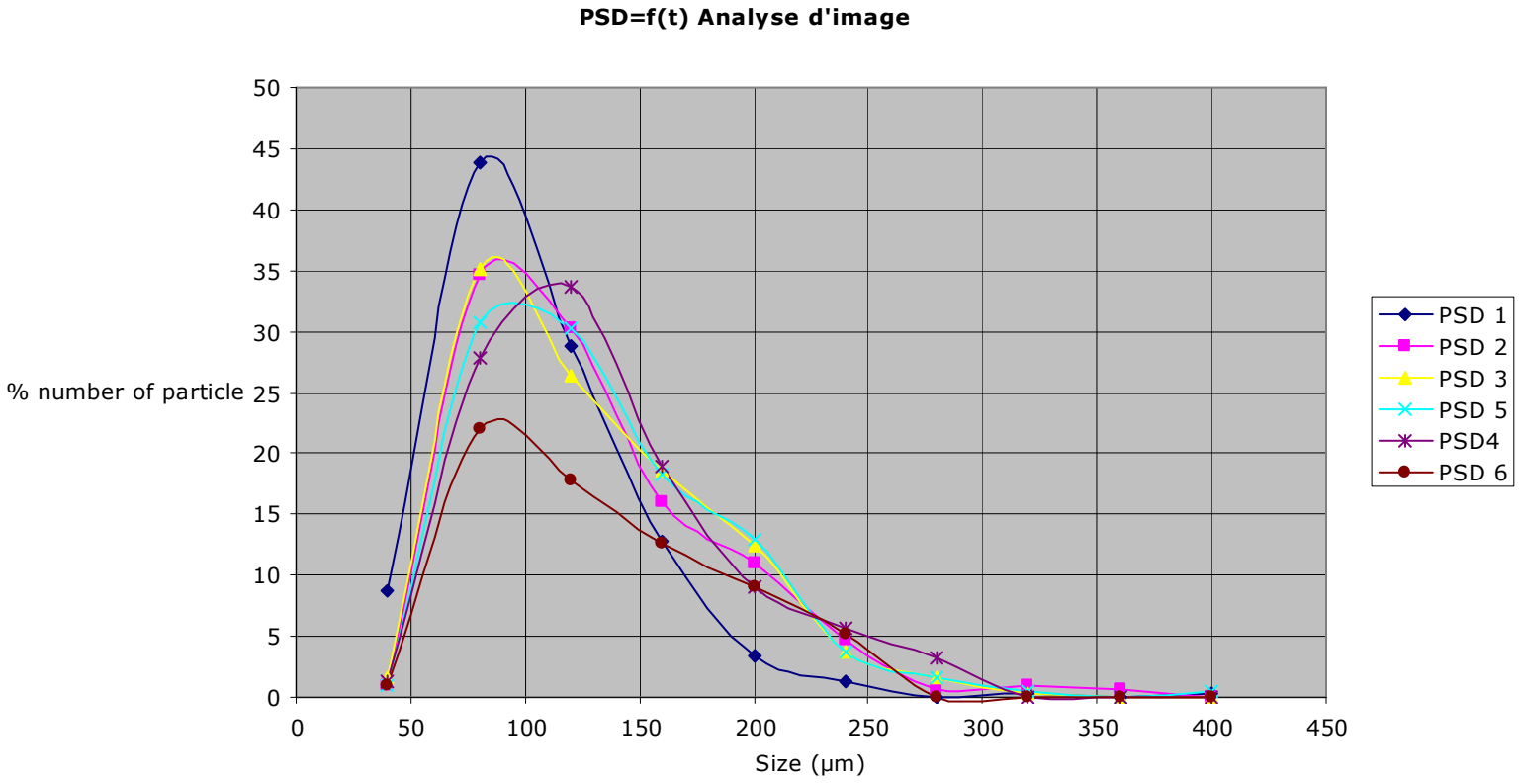
During the first experiment, we have 6 sampling time, the next figure 6-1 show you the images correspond each sampling time and the CLD measured from FBRM, PSD estimated by image analysis



(A)



(B)



(c)

Figure 6-1 (A) The photos from CCD of manipulation 1. (B)CLD. (C) PSD

6.2 The fitting NN structure:

The fitting process consisted of changing the NN structure: number of neurons in hidden layer, as well as parameters of the learning process (number of presentations, learning algorithm and others).

The criterion for model selection was the minimum value obtained for the mean square error E, for computed and experimental values of the output variable, defined as:

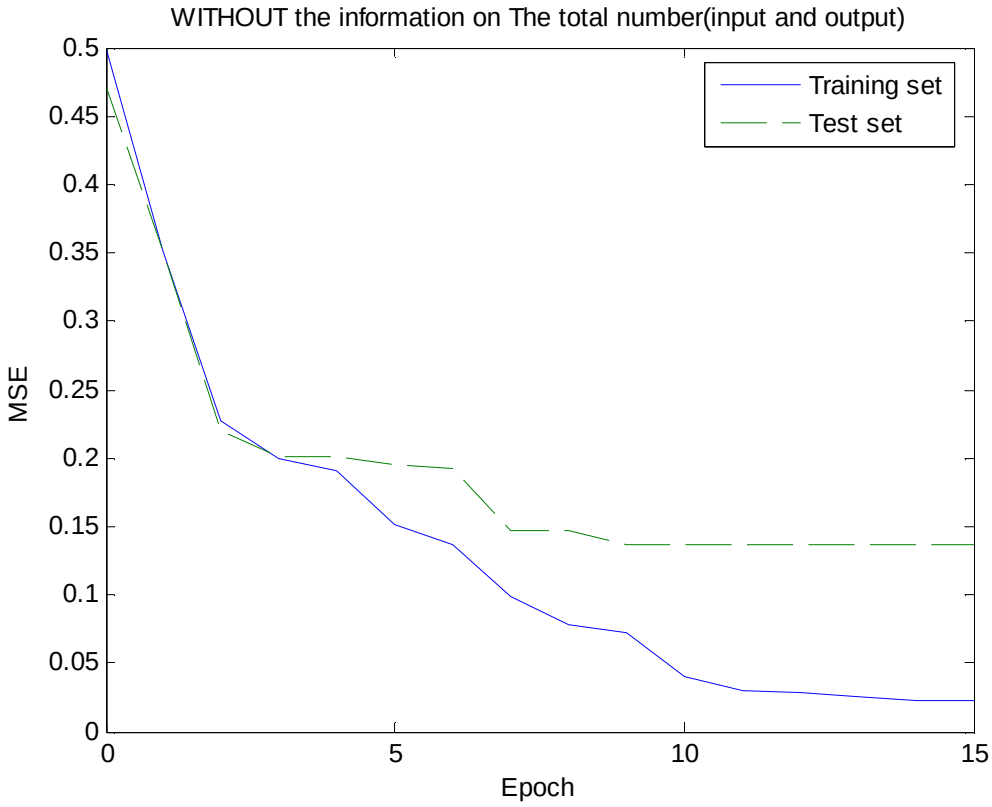
$$E = \frac{1}{g}\frac{1}{p}\sqrt{\sum_{m=1}^{g}\sum_{n=1}^{p}(y_{(m,n)} - o_{(m,n)})^2} \qquad (6\text{-}1)$$

Where y(m,n) represent the experimental value of the mth instant (total of g instant) and nth presentation (total of p presentation). O(m,n) also present the mth instant and nth presentation (of NN. The next table gives us the notion of the manipulations and the correspondent times of sampling:

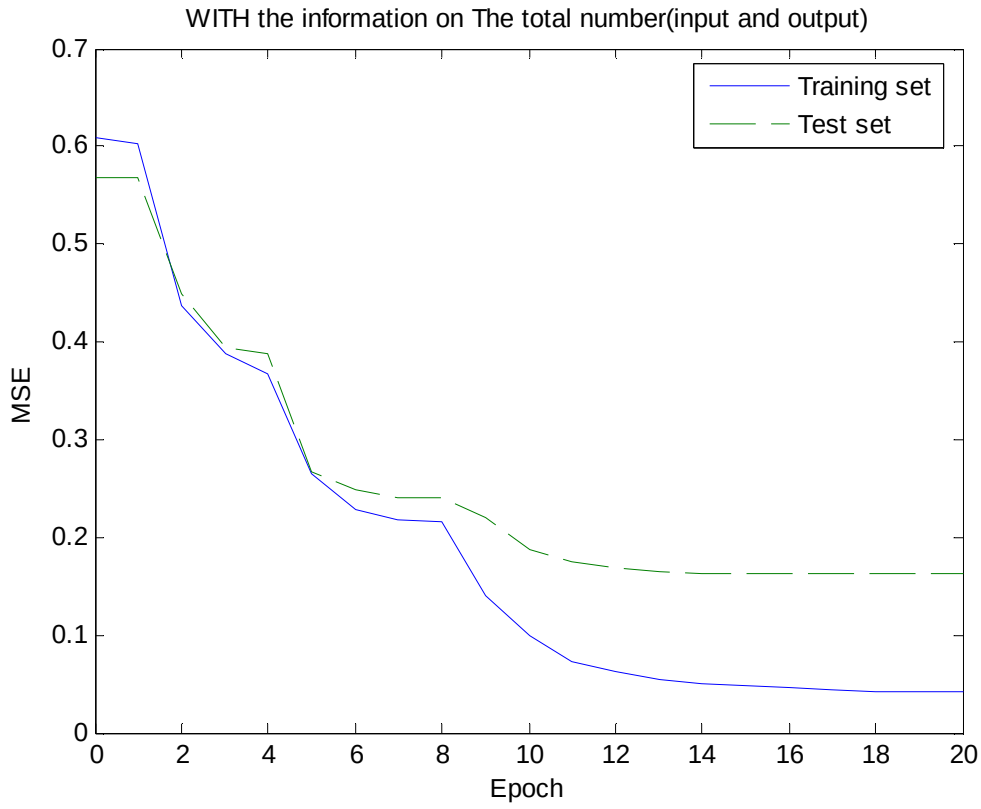
Table4: the different times of sampling in each manipulation

	Total times of sampling
Manipulation 1	6
Manipulation 2	5
Manipulation 3	4
Manipulation 4	8
Manipulation 5	2
Manipulation 6	2
Manipulation 7	5
Manipulation 8	6
Manipulation 9	4
Manipulation 10	7

In each experimental, we consider the particle whose size is between 0 to 400 μm, and the classification of the size always 40 μm. So we have 11 presentations in each instant. As an illustration, Fig 6-2 shows representative plots of the E as a function of epoch of presentations of the data set to the NN, for the training of Model 1 using Levenberg-Marquardt backpropagation algorithm.



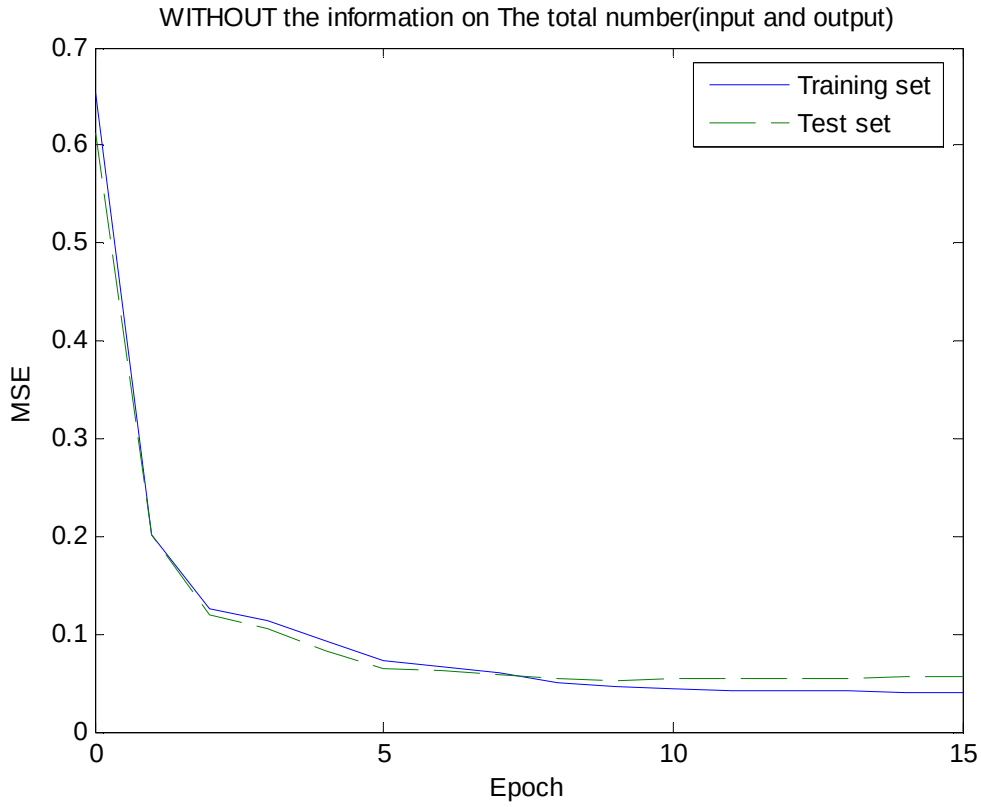
(A)



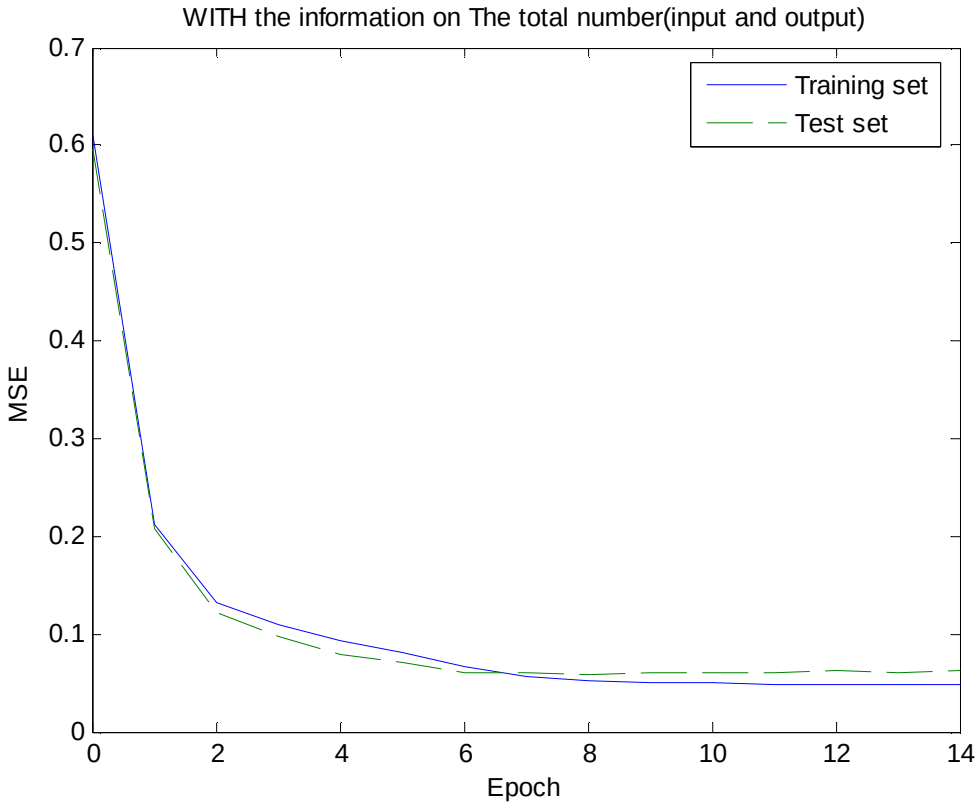
(B)

Figure 6-2 Training with TRAINLM, number of neurons in hidden layer is 11

The importance of the information provided as input to the NN can be observed in plots A and B in this figure. Figure A correspond an NN in which the total number of particles was used as an input variable. Plot B corresponds to an NN in which the aspect ratio was not included as an input variable. Also look at the next figures 6-3 that used the Bayesian regularization backpropagation algorithm:



(A)



(B)

Figure 6-3 training with TRAINBR, number of neurons in hidden layer is 11

We can also see the tendency to stable after certain point. But the air of error line between training set and test set is smaller than plot (A) and plot (B), indicating that the particle number is not a very important information but the we should choose the Bayesian regularization backpropagation as our algorithm. The same conclusion we can obtain is shown in next figures, when the neurons in hidden layers is 20.

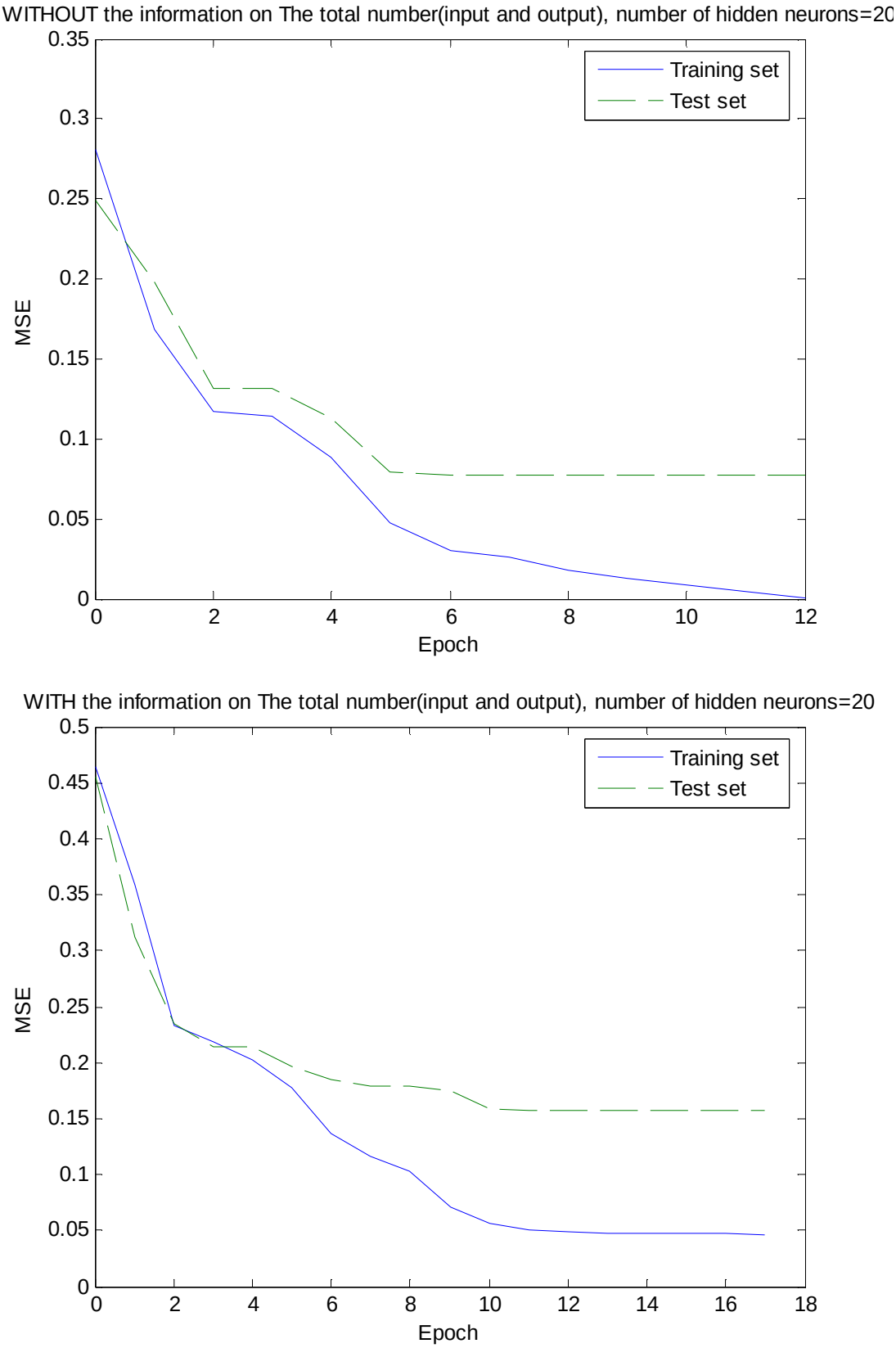
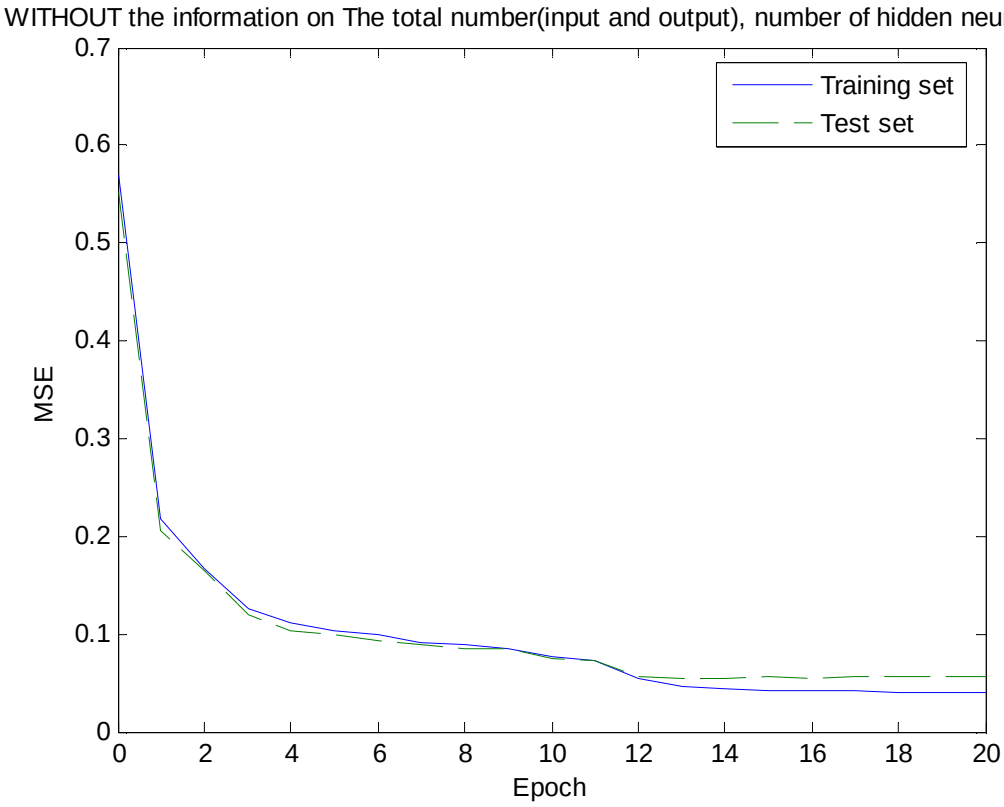
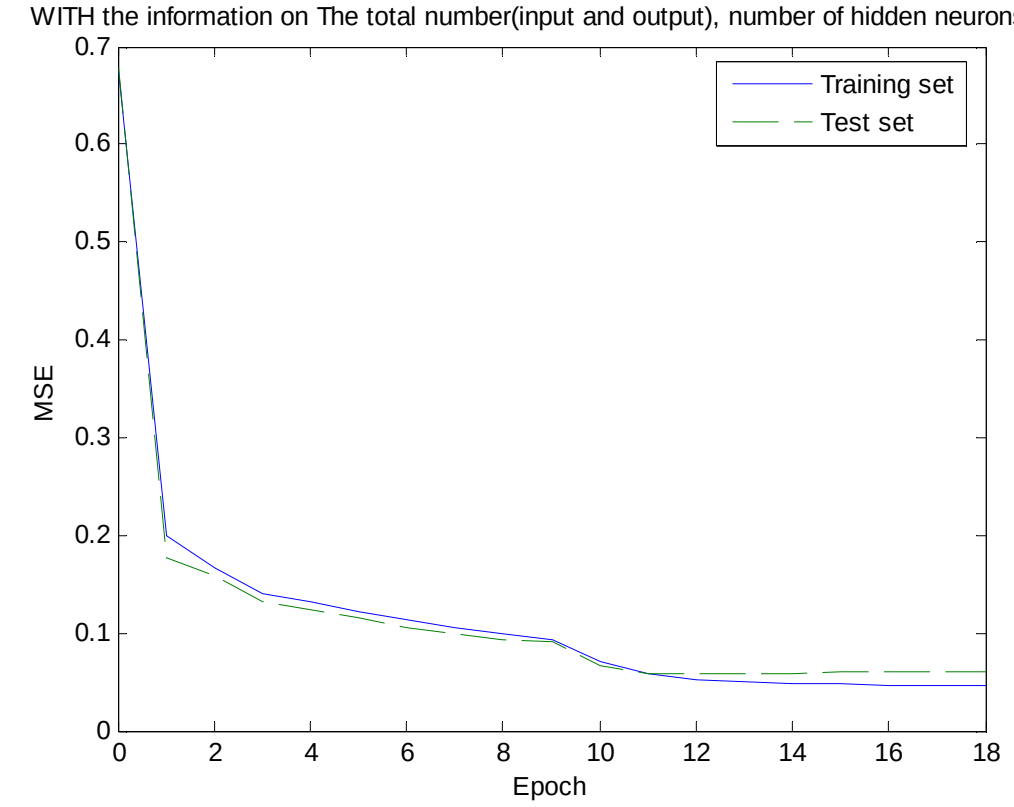


Figure 6-4 training with TRAINLM number of neurons in hidden layer is 20



(A)

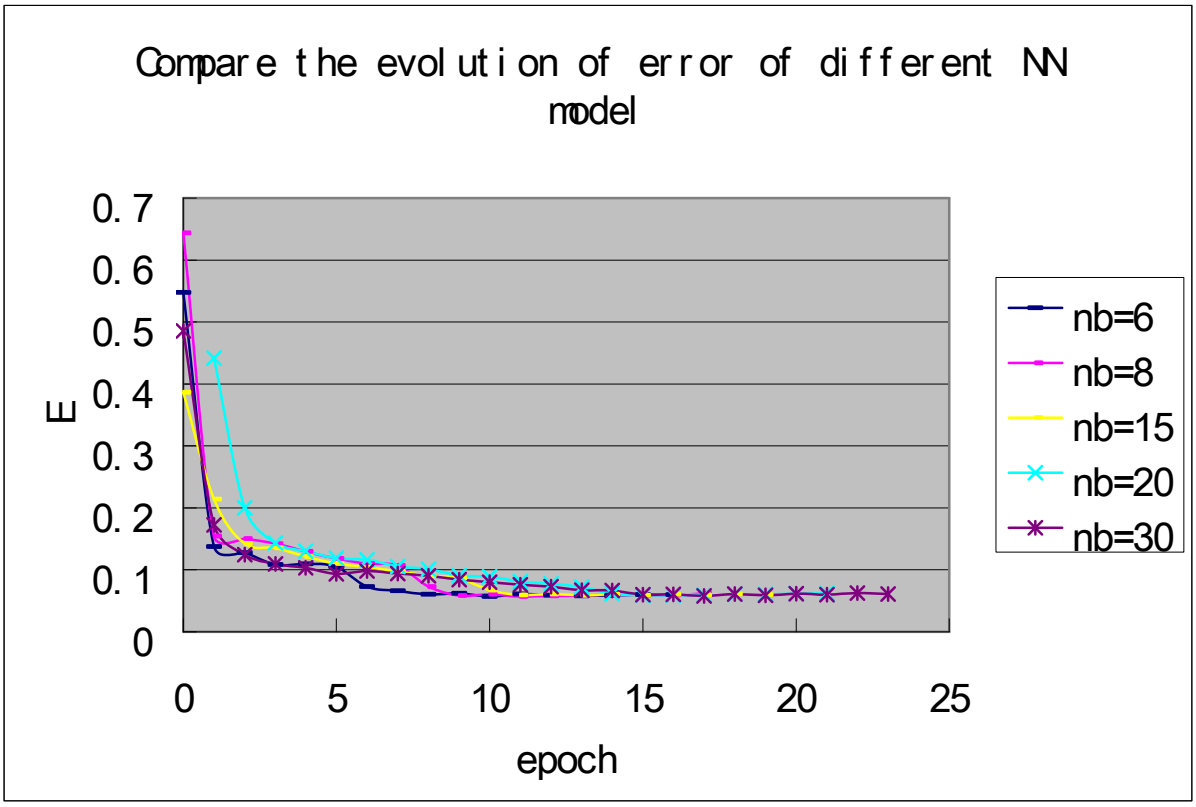


(B)

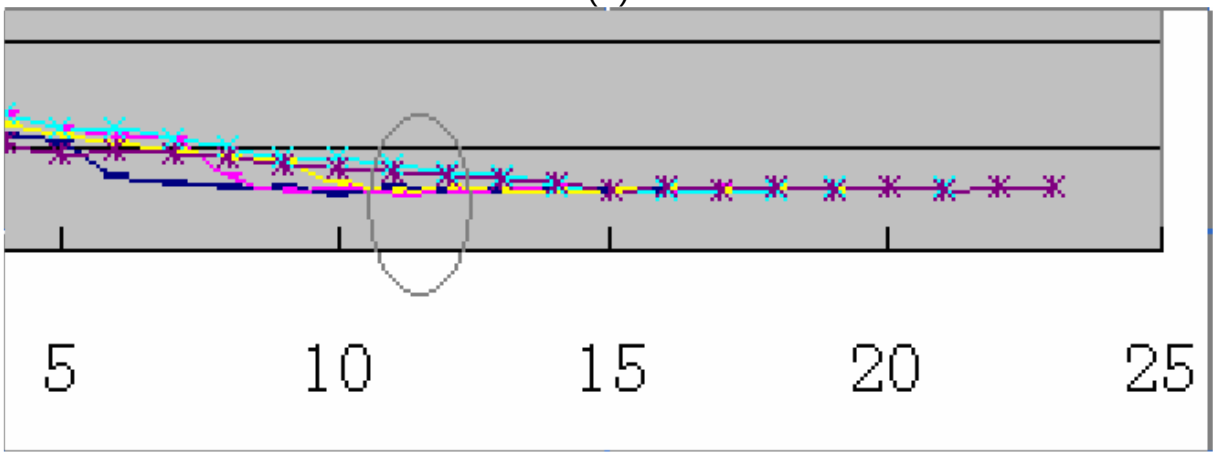
Figure 6-5 training with TRAINBRLM number of neurons in hidden layer is 20

Than we think about how many neurons in hidden layer cause the least E. As the before presented result, we use the cross validation to train the NN of model 1 with different number of neurons in hidden, and compare the result:
-Number of neurons from 6 to 30
-TRAINBR
-The criterion: E

Result is show in the figure 6-6. Because we have so many curbs, here I show you some of them.



(A)



(B)

Figure 6-6(A) The evolution of error with different number of neurons in hidden layer, TRAINBR, (B) zoom of figure A for 5<epoch<25.

Compare of this evolutions of error, we can see the best NN (who has the smallest error) has the 8 number of neurons in hidden layer, (epoch 11), E(11)=0.0567 (calculated by Matlab).

6.3 Discussion of two ANN models:

NN Model 2- the evolution of Model 1:

The evolution of CSD (Crystal size distribution): Introduction of cumulative distribution:

In the present part, I show you the input (CLD) and output (PSD) variable that is used in ANN Model 1. In the 2nd ANN model, I will train the NN using the cumulative distribution as input and output of model, I had mentioned before.

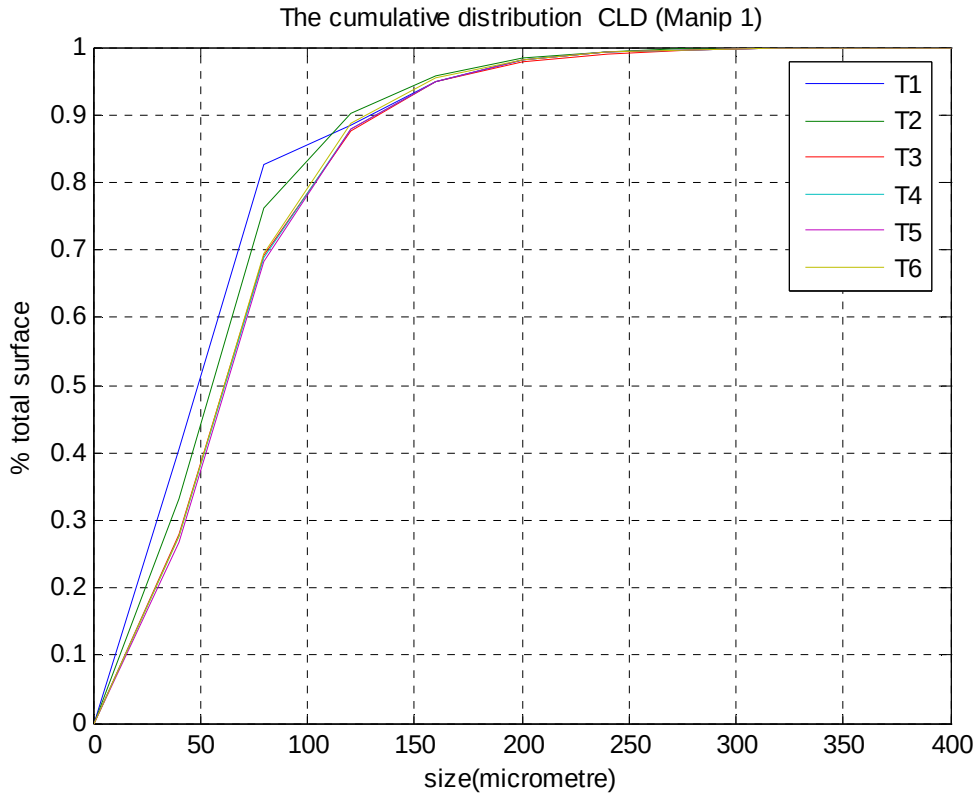
In probability theory, the cumulative distribution function (CDF) completely describes the probability distribution of a real-valued random variable, X. For every real number x, the CDF is given by

$$F(x) = \mathrm{P}(X \leq x), \tag{6-2}$$

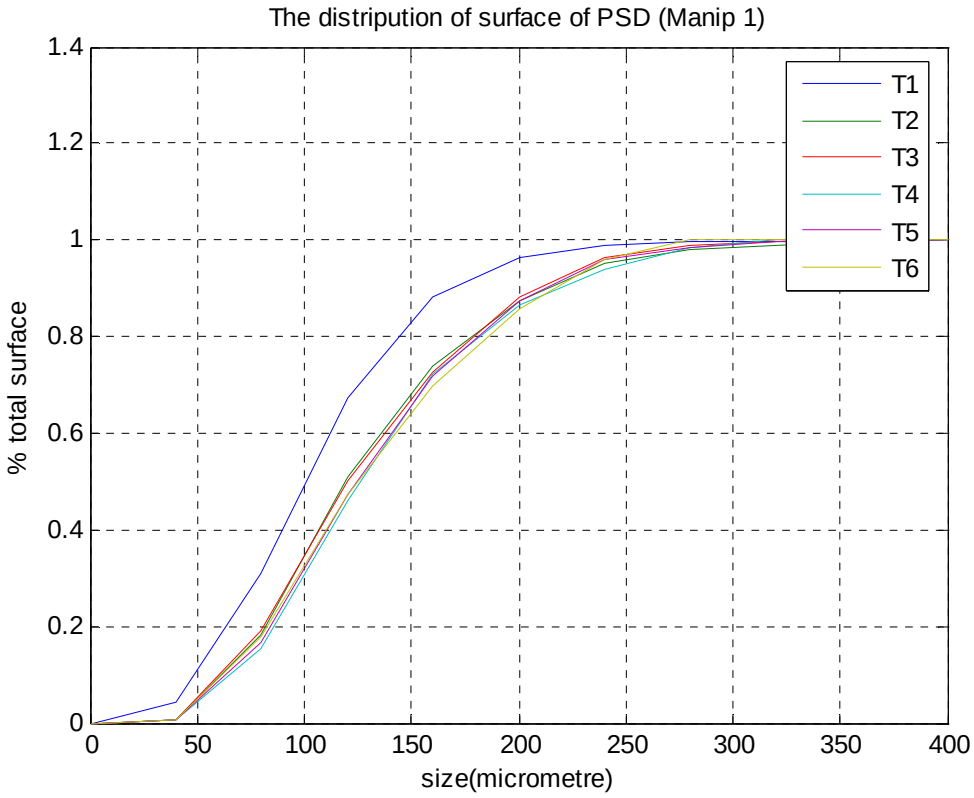
where the right-hand side represents the probability that the random variable X takes on a value less than or equal to x. Propriety:

$$0 \leq F(x) \leq 1, \qquad x_{\min} \leq x \leq x_{\max} \tag{6-3}$$

The time evolutions of the cumulative size distributions of CLD and PSD obtained by first experimental are shown in Figure 6-7. The curves of successive cumulative distributions, presented in this figure, clearly show that during the precipitation run the crystals become larger and larger. Others will be show in APPENDIX B



(A)



(B)

Figure 6-7 (A) the cumulative distribution of CLD (Manipulation 1). (B) the cumulative distribution of PSD (Manipulation 1).

The training error of two models:

After we used the different models to train, the evolution of error shown for both models correspond to fittings obtained after 14 epochs of the data set to the neural network. The best configuration obtained for model 1 is an NN consisting of 12 neurons in the intermediate layer. An illustrative result of tests with Model 1 is shown by blue curb, the green present that of Model 2.

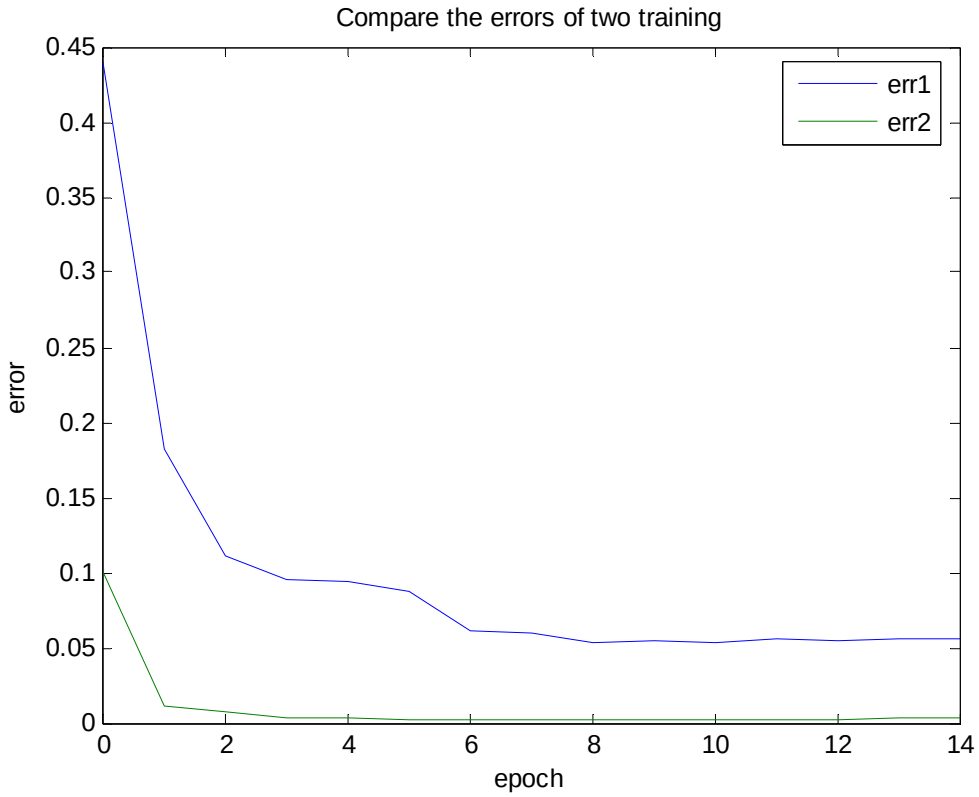
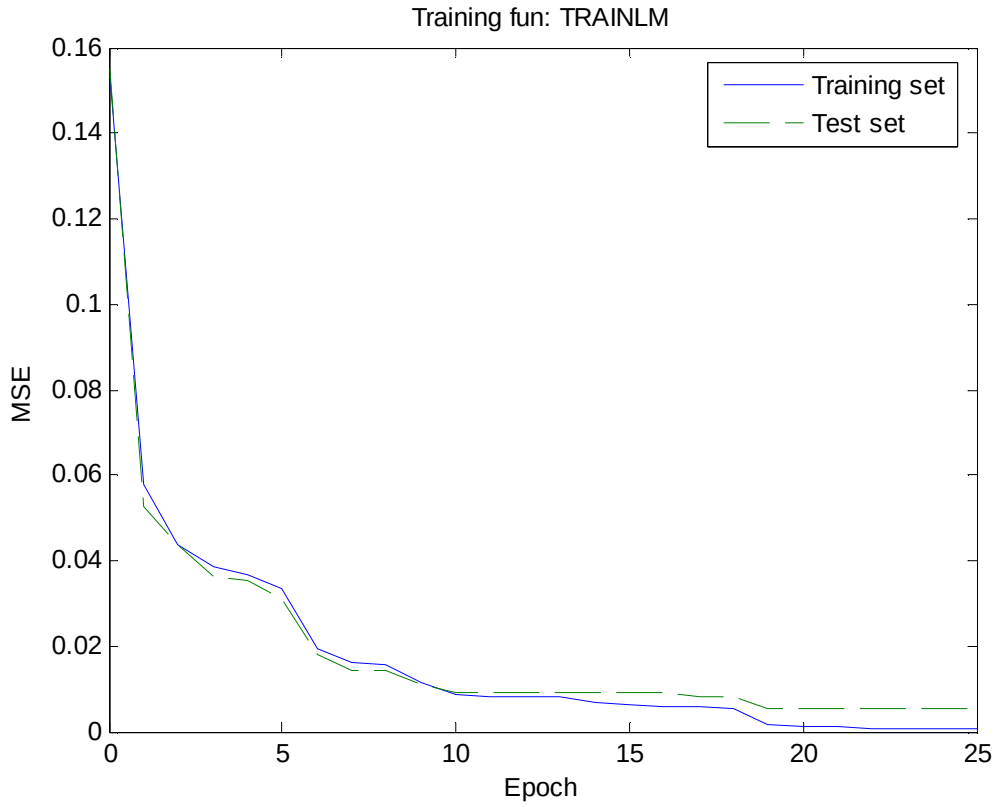


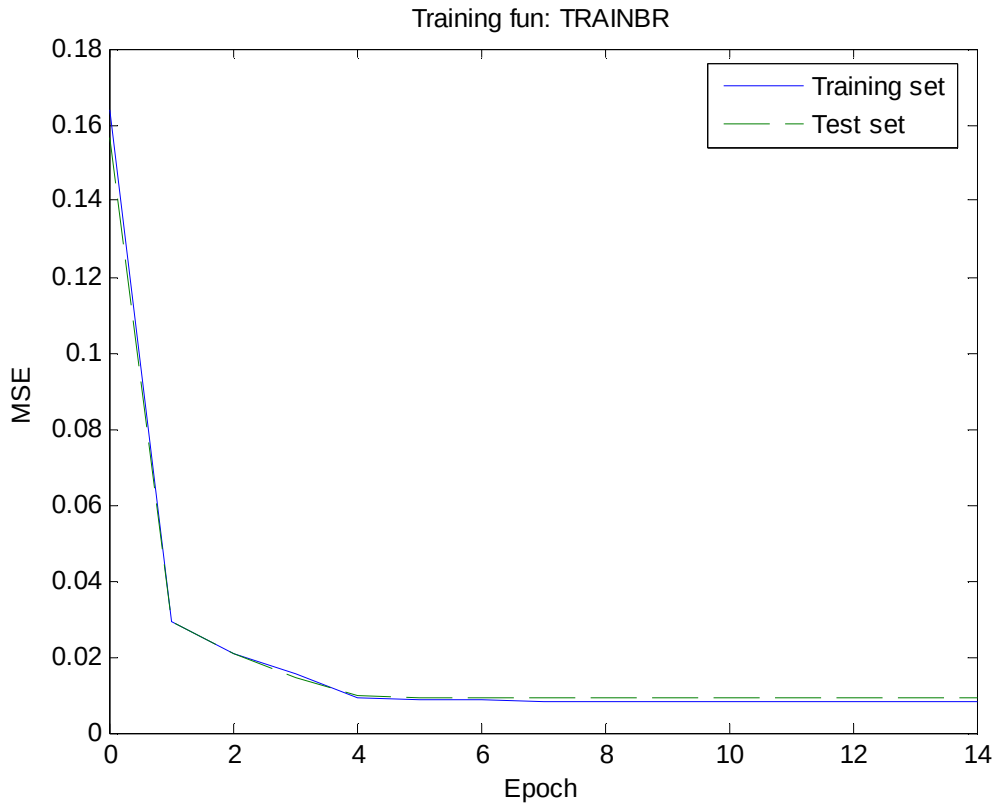
Figure 6-8 the evolution of training error of Model 1 and Model 2, TRAINBR

The result indicates that we have smaller error by train the 2nd model.
But we can't directly use the results of Model 2, because it gives us the cumulative distribution (CD) of PSD, so we have to off-cumulate this result that will have the final PSD.

We also have to discuss which training function should be suitable for Model 2 (as the way used in Model 1). The next figures 6-9 give the compared results of different training, TRAINLM or TRAINBR:



(A)



(B)

Figure 6-9 training of NN Model 2 using TRAINLM or TRAINBR

The result showed that the TRAINBR was the better training function because the air of error line between training set and test set in plot B is smaller than that in plot A.

The problem is that we may obtain the CD of PSD by NN who is not logical. In fact, all of curbs of CD should be increase. I couldn't tell this rule to NN, that's why I had some CD of PSD who might partly decrease, like this (figure 6-10):

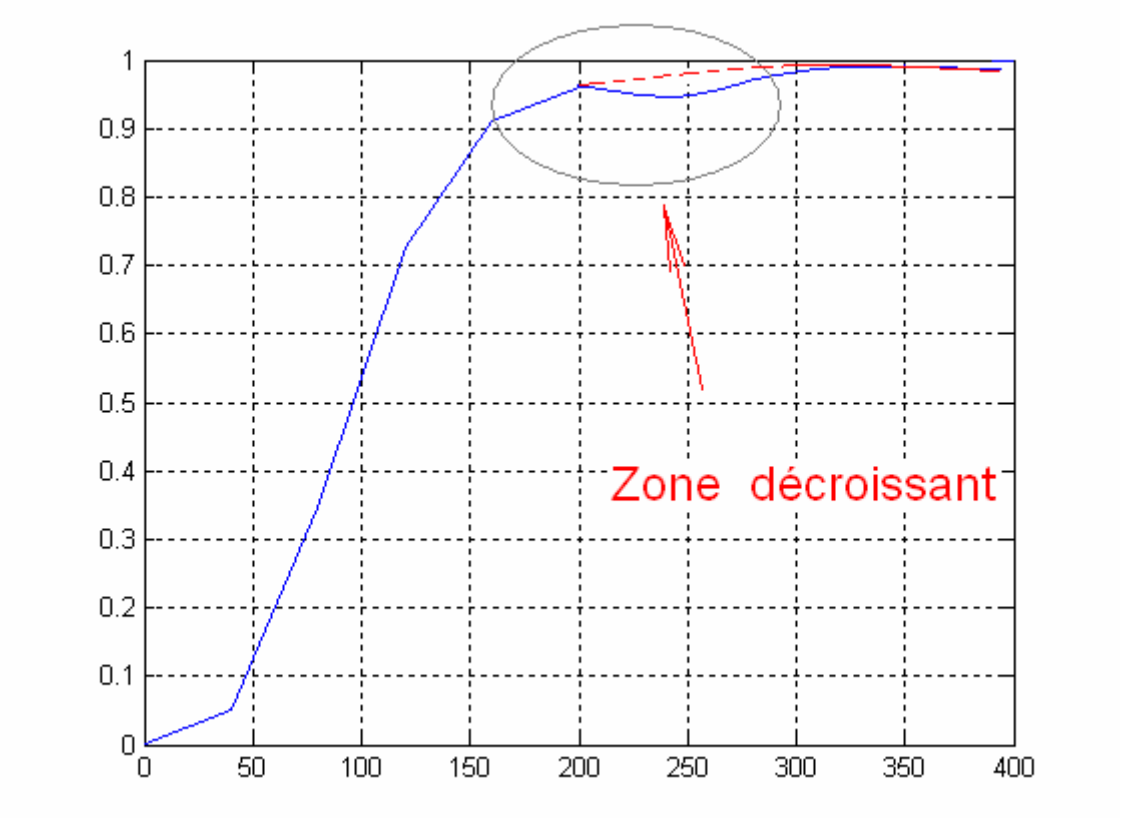


Figure 6-10 the demonstration of problem with Model 2

So I use the methods shown in the next:

```
N=10

While N>0

        Training the NN

        Simulation

        If all of curb is decrease
                Break
        End

N=N+1;

 end
```

Here, N is like the watch dog, it limit the times of circulation, if we can never find all the curbs are decrease, it will be stop after 10 times iteration. (If it is not enough, we

can augment it.). Then I correct the decrease curbs like the way show in figure 6-10, the red curb. By this method, I may have all curb increase, if not lucky, I may have some curb who has the partly decrease. This is encouraged for future work. Unfortunately it is not possible for the moment to introduce such constraints during the training.

Simulation of NN Model 1 and NN Model 2:

Finally, I simulation the NN Model 1(using all data here), the result s show in figure 6-11 (also the result of experimental 1, others will be shown in APPENDIX C).

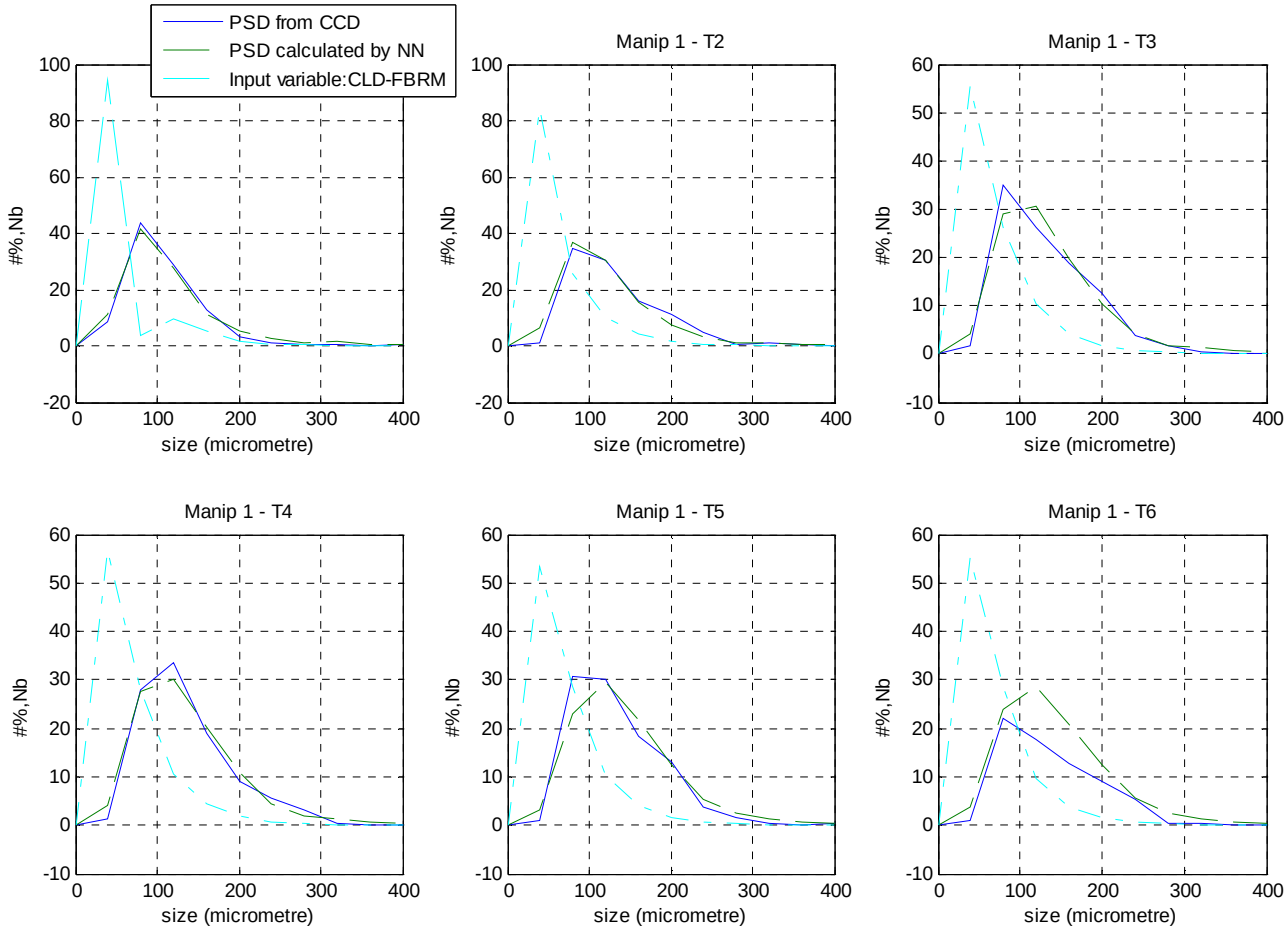


Figure 6-11 The result of NN model 1 correspond the 1 experimental

The best configuration obtained for Model 2 is an NN consisting of 23 neurons in the intermediate layer (I had used the same method named cross validation as Model 1). The results show in figure 6-12 (also the result of experimental 1, others will be shown in APPENDIX D).

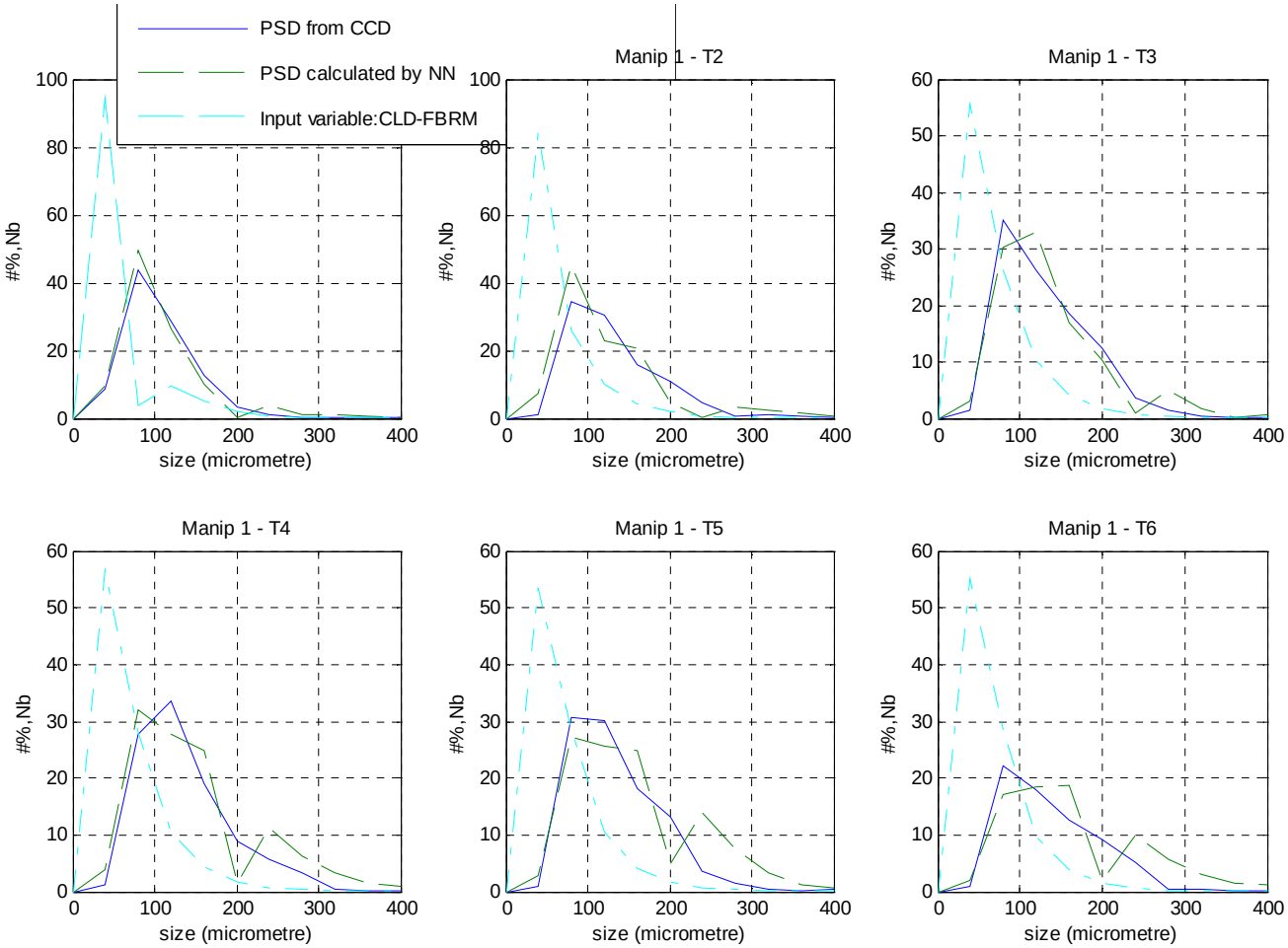


Figure 6-12 The result of NN model 2 correspond the 1 experimental

By the way, if we could have more PSD from image analysis, the model could be more exact. You can see it in the annex when you compare results of all manipulations.

If we calculate the E of simulation where E is the criterion as (6-1), the final errors of simulation were shown in the next table:

Table 5: (A) the final error of two NN model (data normalized), no unit

The error of training of Model 1	0.0135
The error of training of Model 2	0.0023

(B) the final error of two NN model (data without be normalized),unit: %

The error of simulation of Model 1	0.1926
The error of simulation of Model 2	0.2803

I have to talk about that the errors of simulation are not same as the error of training of NN, because we had used the normalization before training but we hadn't normalized all data for simulation. Although we had smaller error by training NN

Model 2, that I had give you before. As the reason of de-cumulate, I couldn't arrive the better PSD from NN Model 2.

The cause of error maybe:

- Limitation of CCD, which made the insufficient image analysis
- Be lack of sampling time in an experimental (by the results, at least of 5~6 times of measurement will be better for training a NN model)
- So many same comportments, advising to do PCA (principle comportment analysis).

CHAPTER 7 CONCLUSION

The results of fitting the two NN models indicate the feasibility of developing software sensors for in line and in situ monitoring of crystallization processes, either in laboratory of in industrial scale, based on the information on the CLD provided by FBRM laser reflectance sensors.

Based on the CSD, it is possible to implement different process control tools in industrial processes, to fit model parameters, and to investigate specific configuration problems in industrial equipments. Thus, the adequate calibration of NN models to the data constitutes a valuable tool in process design and operation.

In the future, we may improve our work from the next:

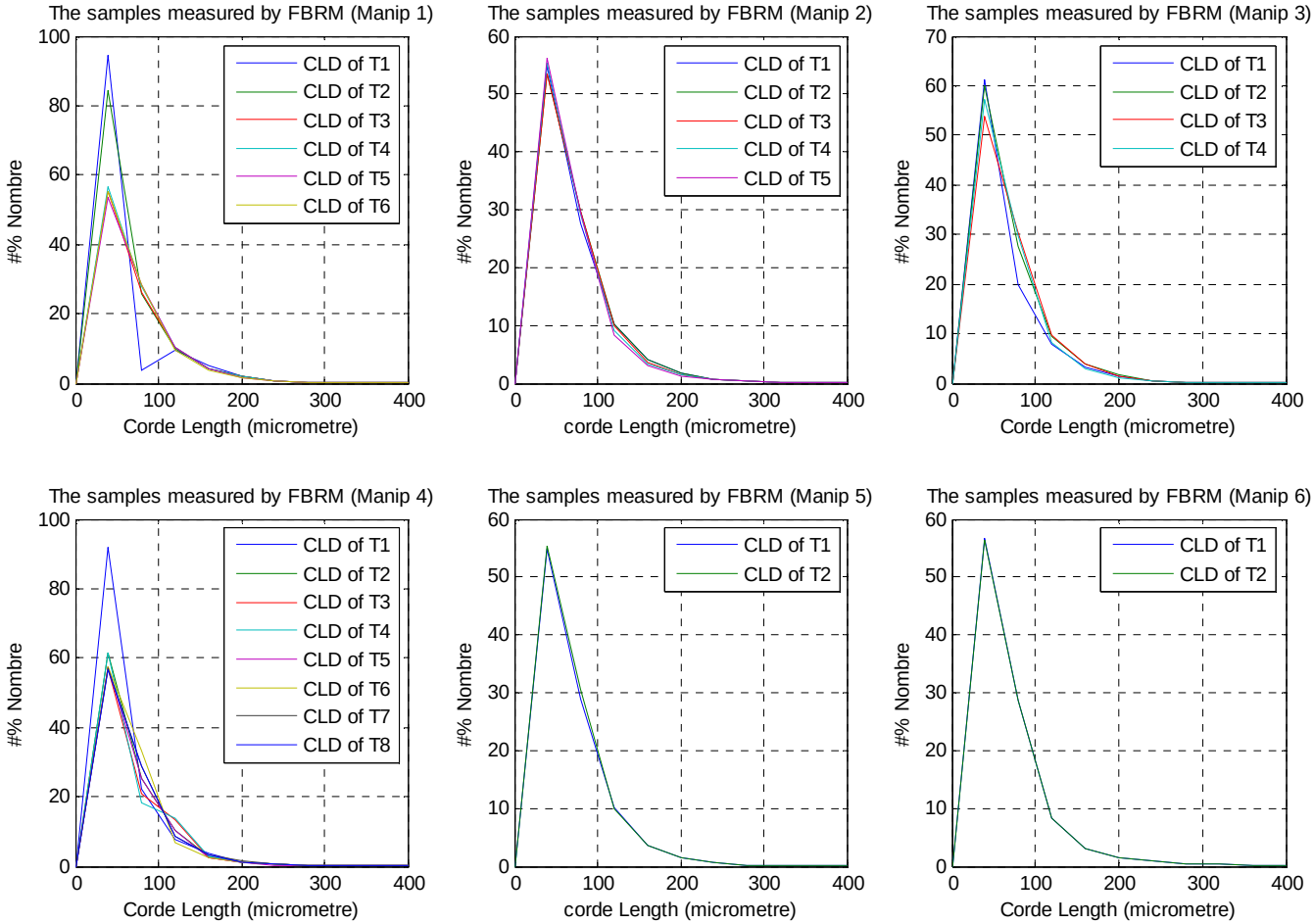
- Sample more data during an experimental crystallization, it will be better for training NN model.
- Improve the algorithm for solve the problem of constraint training as I had mentioned in chapter 5.
- Conjugation with PCA for the input data of NN model, it is a dimensionality reduction technique and correlated process input measurements into a smaller, informative latent variable space. In additionally, we need more sampling data for doing PCA, in this work, I had try to do it, but the result was not better than other because we have insufficient data.

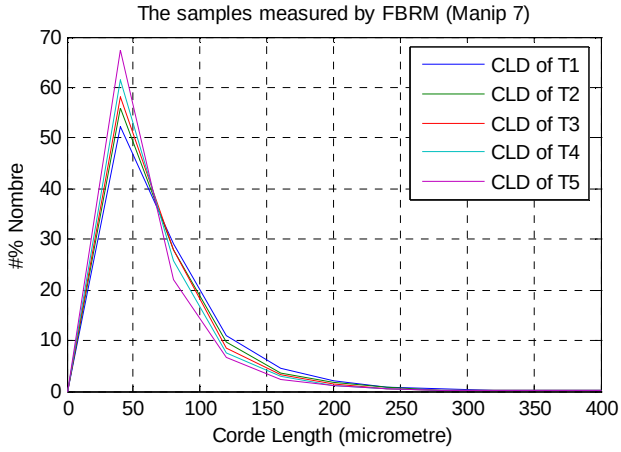
REFERENCES

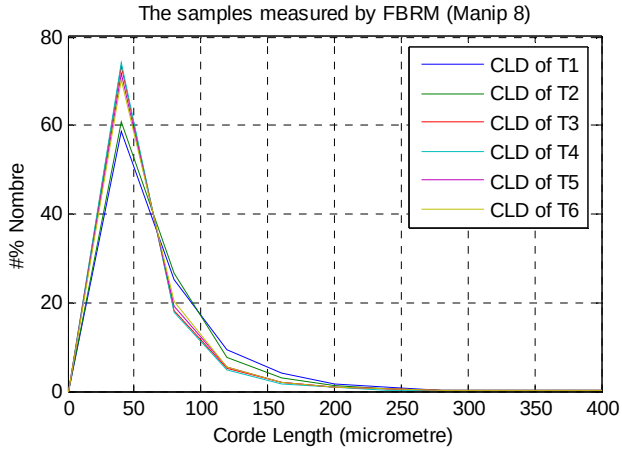
[1] J. B. Rawlings, S. M. Millerm, W. R. Witkowski: Model identification and control of solution crystallization processes-a review. Ind. Eng. Chem. Res. 32 (1993) 1275-1296.
[2] W. R. Witkowski, J. B. Rawlings, S. M. Millerm, Light-scattering measurements to estimate kinetic-parameters of crystallization. AIChE Symposium Series 438 (1990) 102-114.
[3] A. Boxman: Particle size measurement for the control of industrial crystallizers. PhD thesis, Tech. Univ. of Delft. The Netherlands, 1992.
[4] A. m. Neumann, H. J. M. Kramer, M. Zhemua, B. Scarlett: On-line measurement techniques for industrial crystallization processes. Proc. 14th International Symposium on Industrial Crystallization, Cambridge, UK, IChemE, Rugby, UK, Sep., 12-16 1999.
[5] E. J. W. Wynn: Relationship between particle-size and chord-length distributions in focused beam reflectance measurement: stability of direct inversion and weighting. Elsevier Powder Technology 133 (2003) 125-133.
[6] Alfred Ruf, Jörg Worlitschek, Marco Mazzotti : Modeling and experimental analysis of PSD measurements through FBRM. Part. Part. Syst. Charact. 17 (2000) 167-179.
[7] PORTRAT Vincent: Etude de la granulométrie de l'acide adipique lors de sa cristallisation Analyse d'images – Analyse par sonde FBRM, report of internship 2006 LAGEP and IUT Lyon 1
[8] Christos Stergiou, Dimitrios Siganos, NEURAL NETWORKS journal 2005 Imperial College London.
[9] Don.R.HUSH and Bill.G..HORNE Progress in supervised neural networks:what's new since Lippmann's, IEEE signal processing magazine January 1993.
[10] Pascal Dufour, Sharad Bhartiya, Prasad S.Dhurjati, Francis J. Doyle III, Neural network-based software sensor: training set design and application to a continuous pulp digester, Control Engineering Practice 23 February 2004 ELSEVIER
[11] W.Wu, B.Walczak, D.L.Massart, S.Heuerding, F.Erni, I.R.Last, K.A. Prebble, Articial neural network
[12] Robert Kocjancic, Jure Zupan, Modelling of the river flowrate: the influence of the training set selection. Chemometrice and Intelligent Laboratory systems 18 August 2000
[13] Mohamed Azlan Hussain, Review of application of neural networks in chemical process control-simulation a nd online implementation. 2 June 1998.

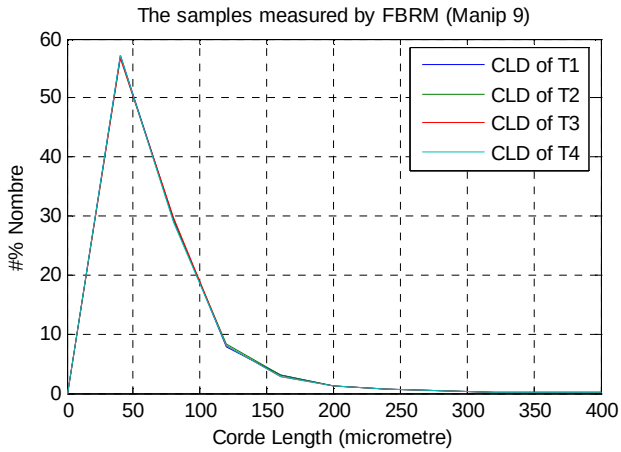
APPENDIX A The measurement of each experimental with CLD and PSD.(Manipulation1~Manipulation 10):

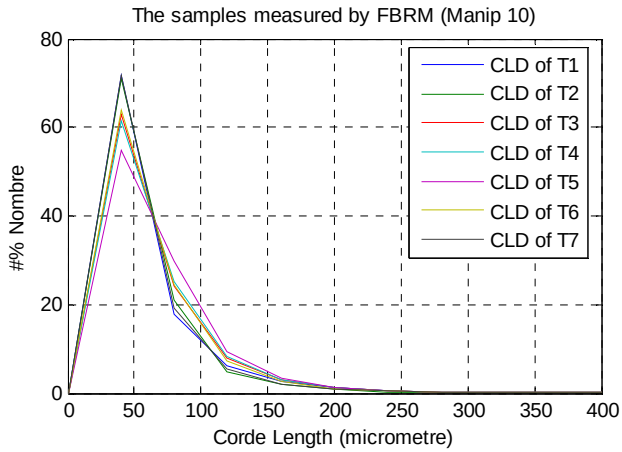
(1) CLD measured by FBRM of manipulation 1~10 simulated by Matlab:


The samples measured by FBRM (Manip 1)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4
CLD of T5
CLD of T6
The samples measured by FBRM (Manip 2)
#% Nombre
corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4
CLD of T5
The samples measured by FBRM (Manip 3)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4
The samples measured by FBRM (Manip 4)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4
CLD of T5
CLD of T6
CLD of T7
CLD of T8
The samples measured by FBRM (Manip 5)
#% Nombre
corde Length (micrometre)
CLD of T1
CLD of T2
The samples measured by FBRM (Manip 6)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2

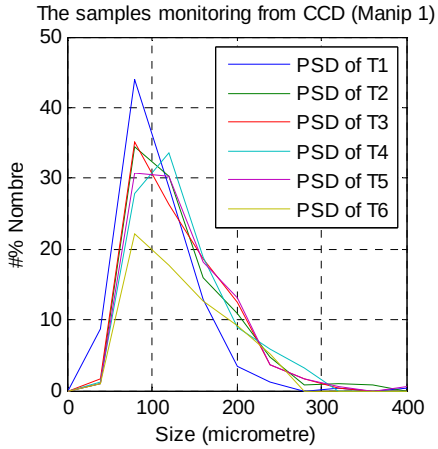

The samples measured by FBRM (Manip 7)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4
CLD of T5

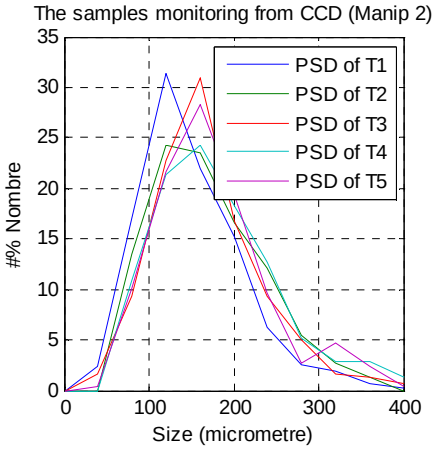

The samples measured by FBRM (Manip 8)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4
CLD of T5
CLD of T6

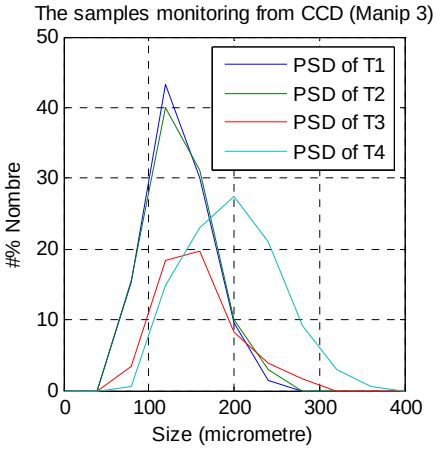

The samples measured by FBRM (Manip 9)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4

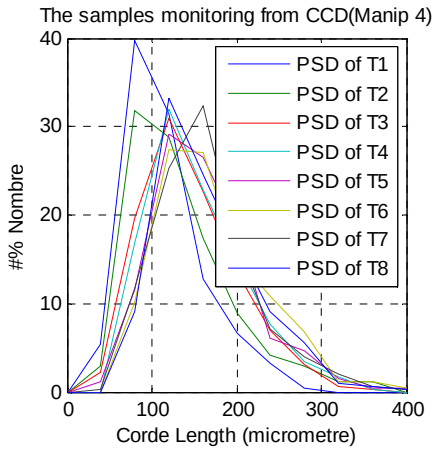

The samples measured by FBRM (Manip 10)
#% Nombre
Corde Length (micrometre)
CLD of T1
CLD of T2
CLD of T3
CLD of T4
CLD of T5
CLD of T6
CLD of T7

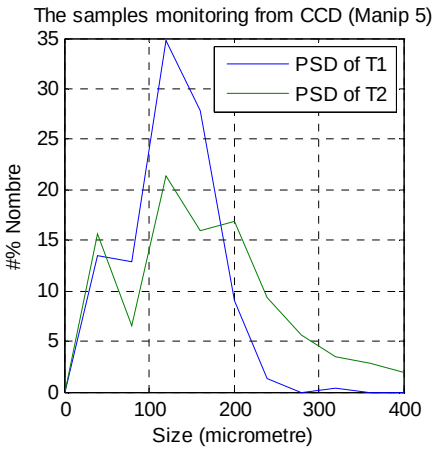
(2)PSD estimated by image analysis of manipulation 1~10

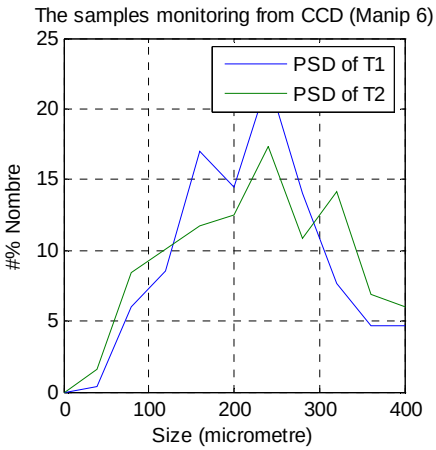

The samples monitoring from CCD (Manip 1)
PSD of T1
PSD of T2
PSD of T3
PSD of T4
PSD of T5
PSD of T6
#% Nombre
Size (micrometre)

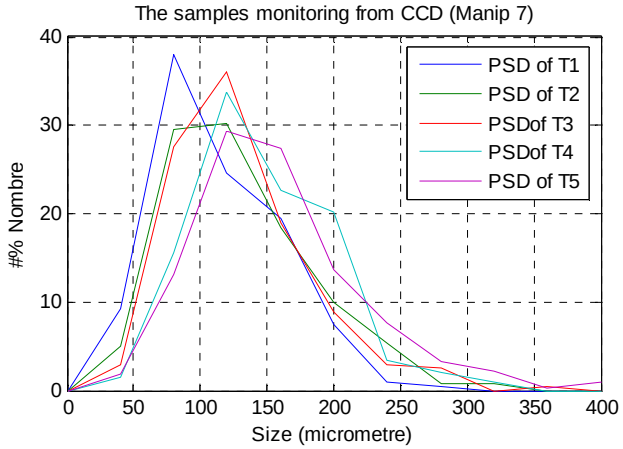

The samples monitoring from CCD (Manip 2)
PSD of T1
PSD of T2
PSD of T3
PSD of T4
PSD of T5
#% Nombre
Size (micrometre)

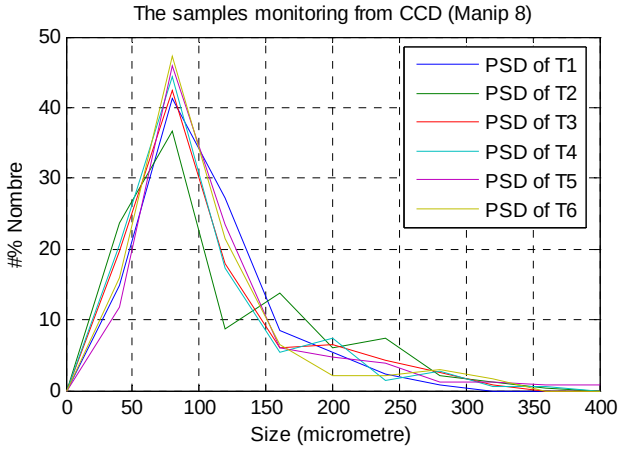

The samples monitoring from CCD (Manip 3)
PSD of T1
PSD of T2
PSD of T3
PSD of T4
#% Nombre
Size (micrometre)

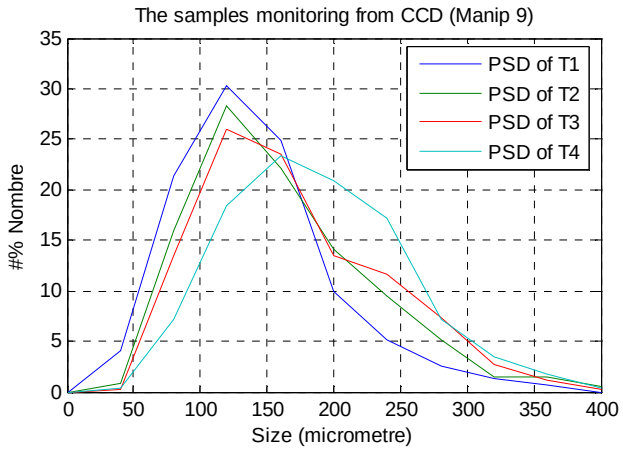

The samples monitoring from CCD(Manip 4)
PSD of T1
PSD of T2
PSD of T3
PSD of T4
PSD of T5
PSD of T6
PSD of T7
PSD of T8
#% Nombre
Corde Length (micrometre)

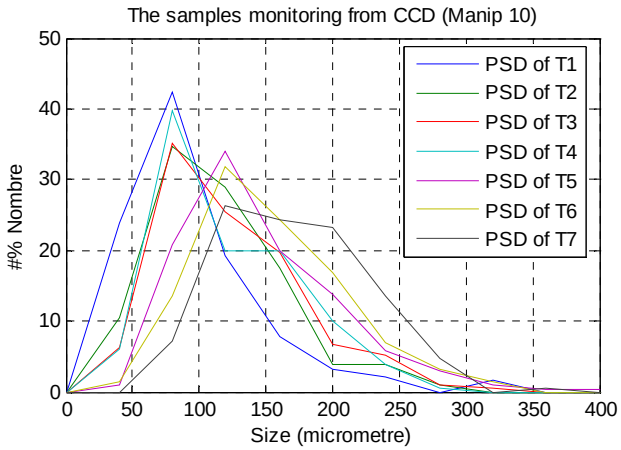

The samples monitoring from CCD (Manip 5)
PSD of T1
PSD of T2
#% Nombre
Size (micrometre)


The samples monitoring from CCD (Manip 6)
PSD of T1
PSD of T2
#% Nombre
Size (micrometre)



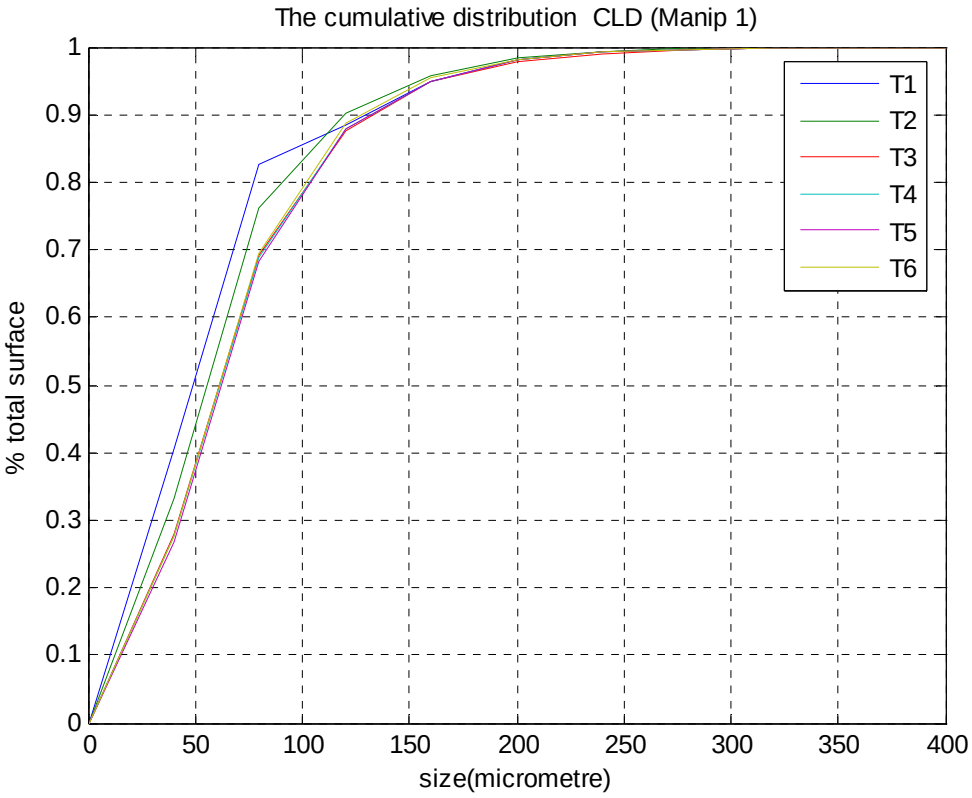


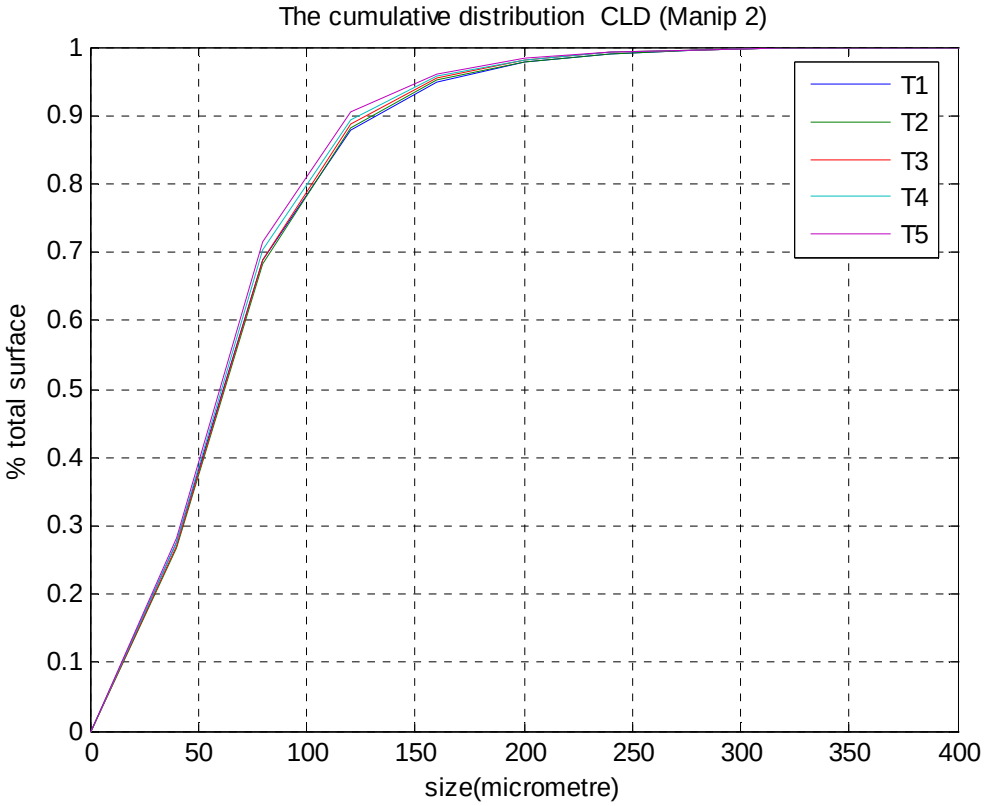


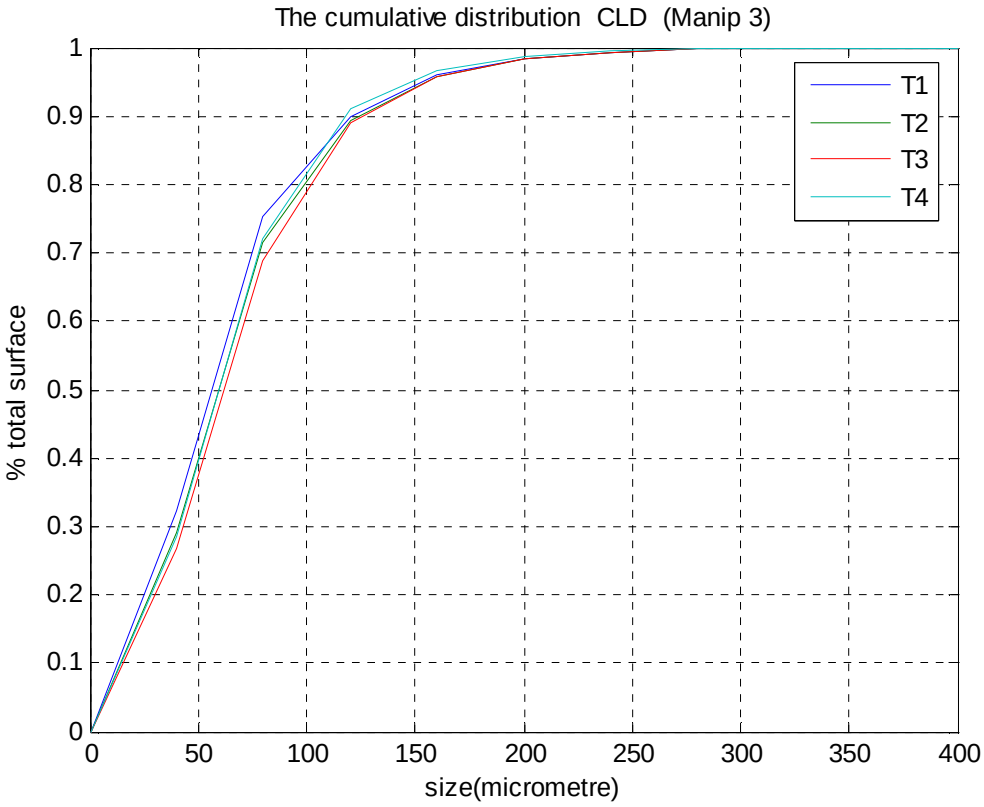


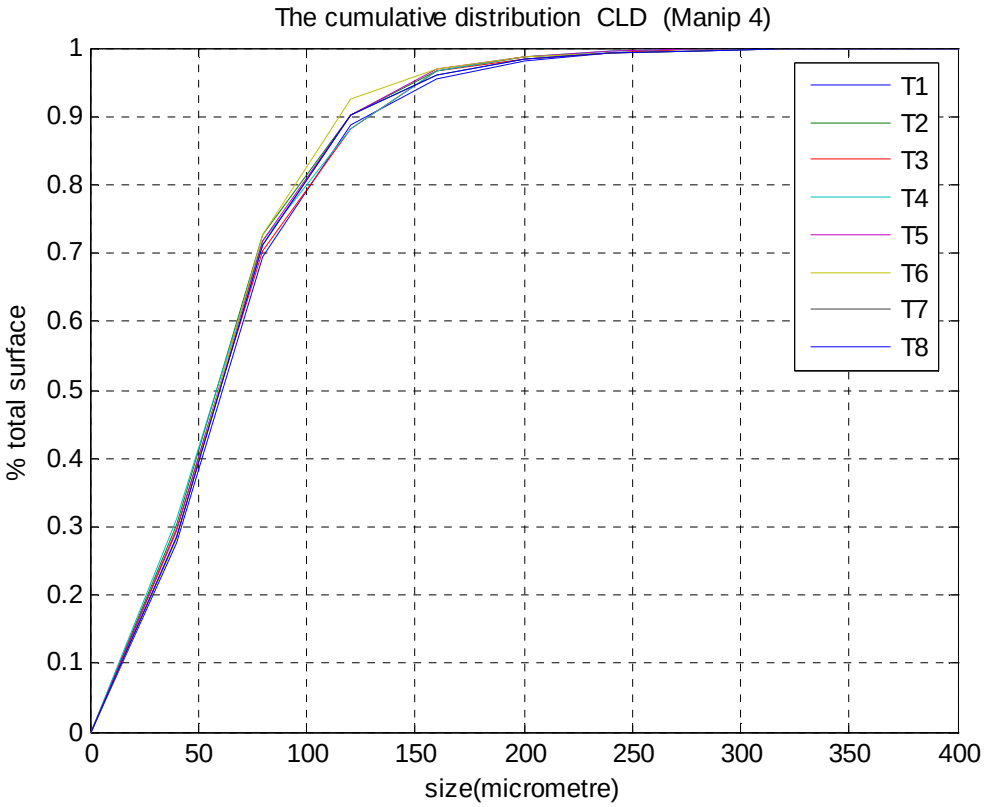
APPENDIX B The cumulative distribution for ANN model 2 of each manipulation.

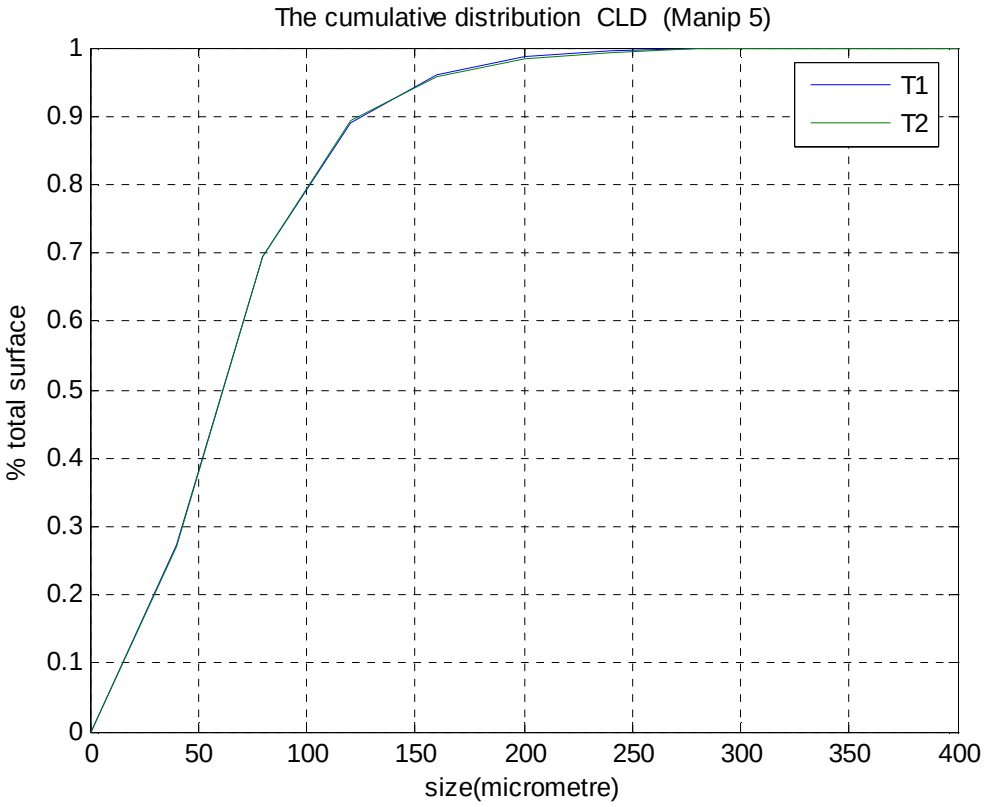
(1) The cumulative distribution of CLD

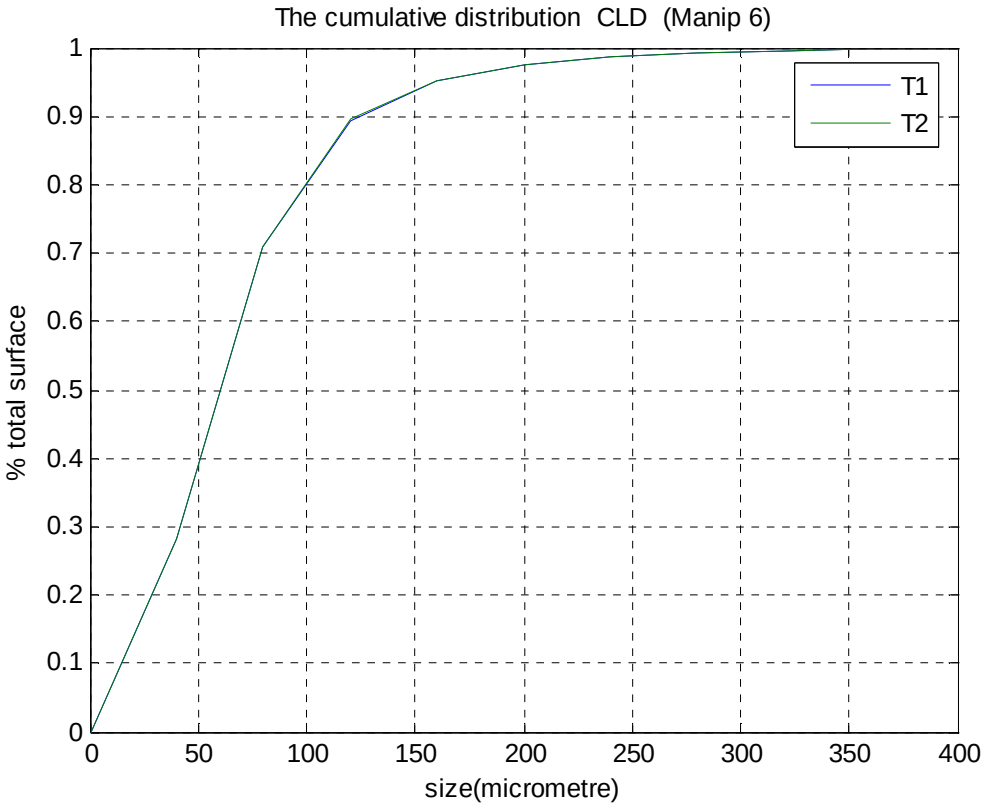

The cumulative distribution CLD (Manip 1)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
T6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

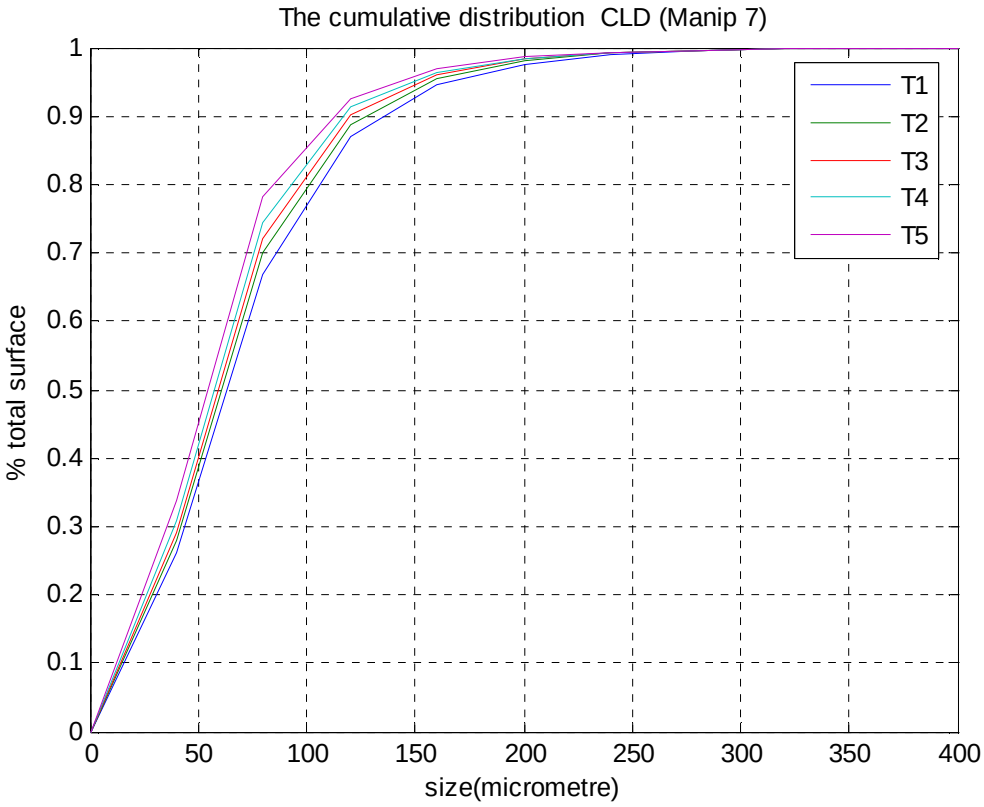

The cumulative distribution CLD (Manip 2)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

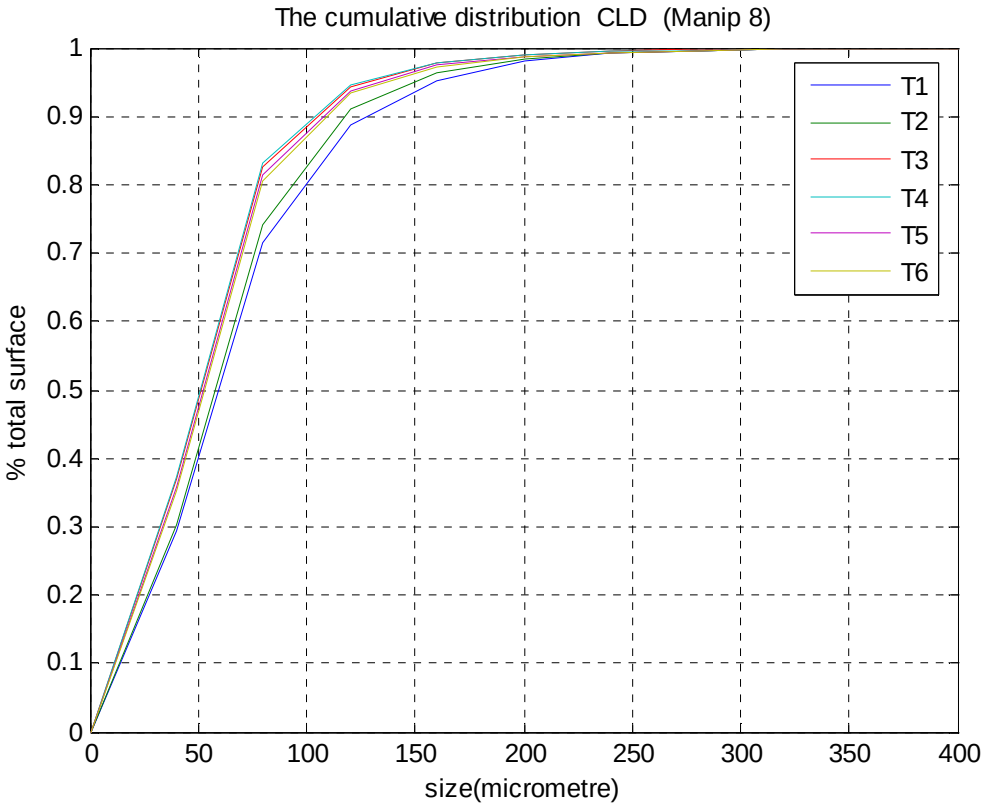

The cumulative distribution CLD (Manip 3)
% total surface
size(micrometre)
T1
T2
T3
T4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

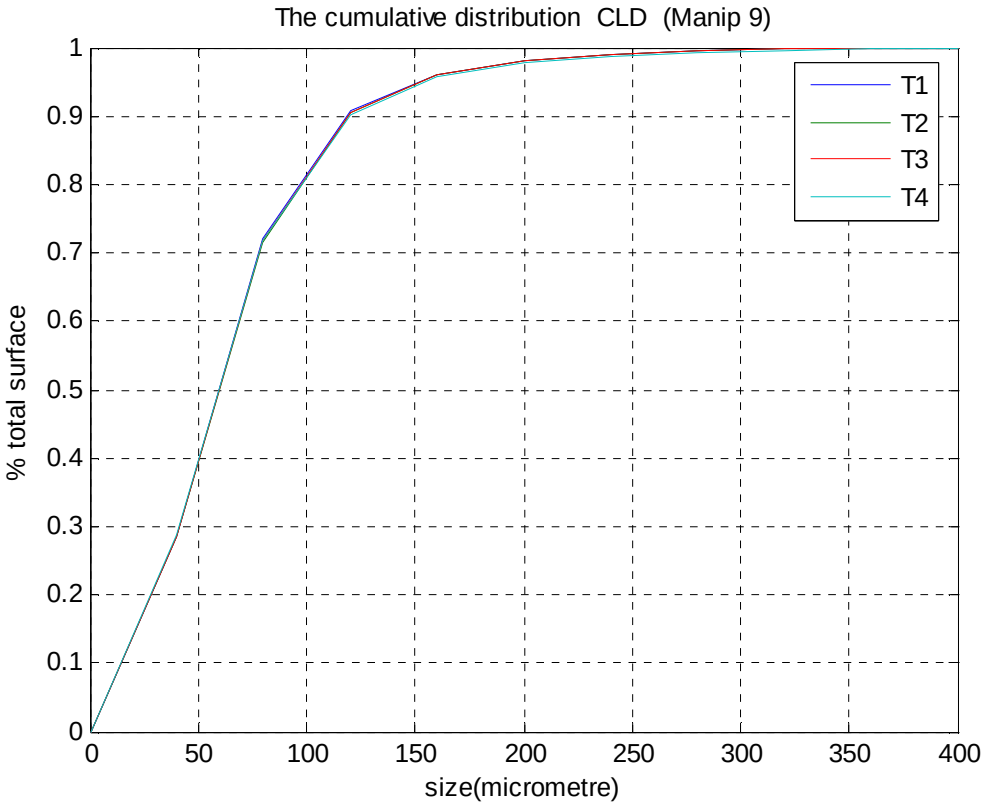

The cumulative distribution CLD (Manip 4)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
T6
T7
T8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

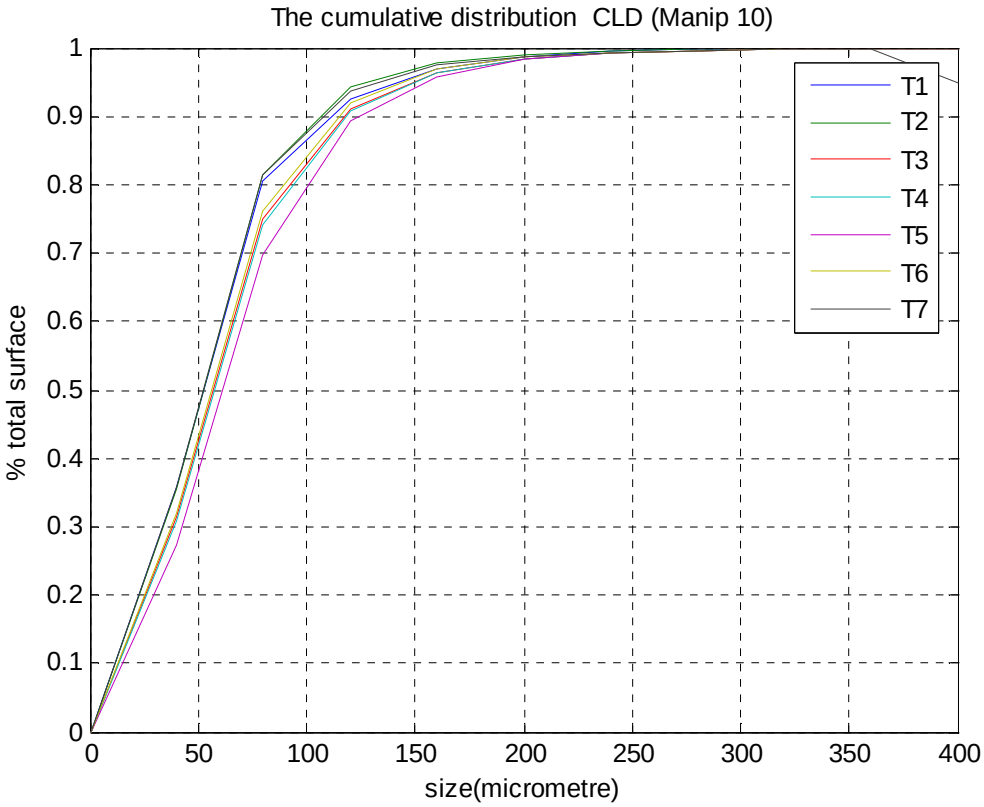

The cumulative distribution CLD (Manip 5)
T1
T2
% total surface
size(micrometre)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400


The cumulative distribution CLD (Manip 6)
T1
T2
% total surface
size(micrometre)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

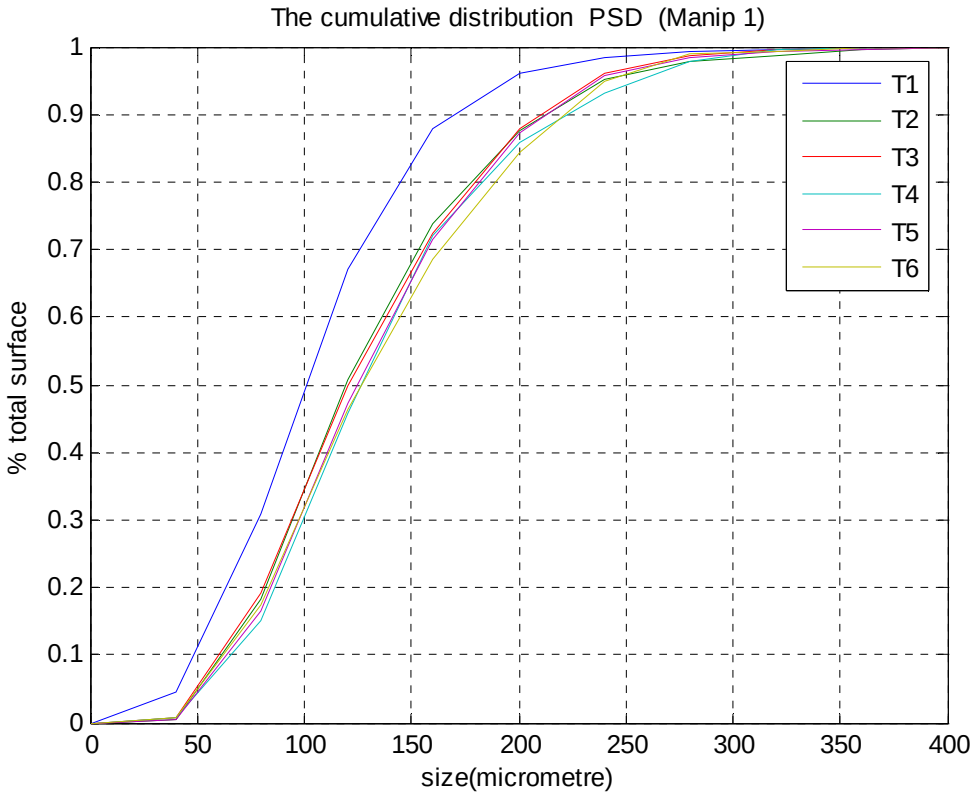

The cumulative distribution CLD (Manip 7)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

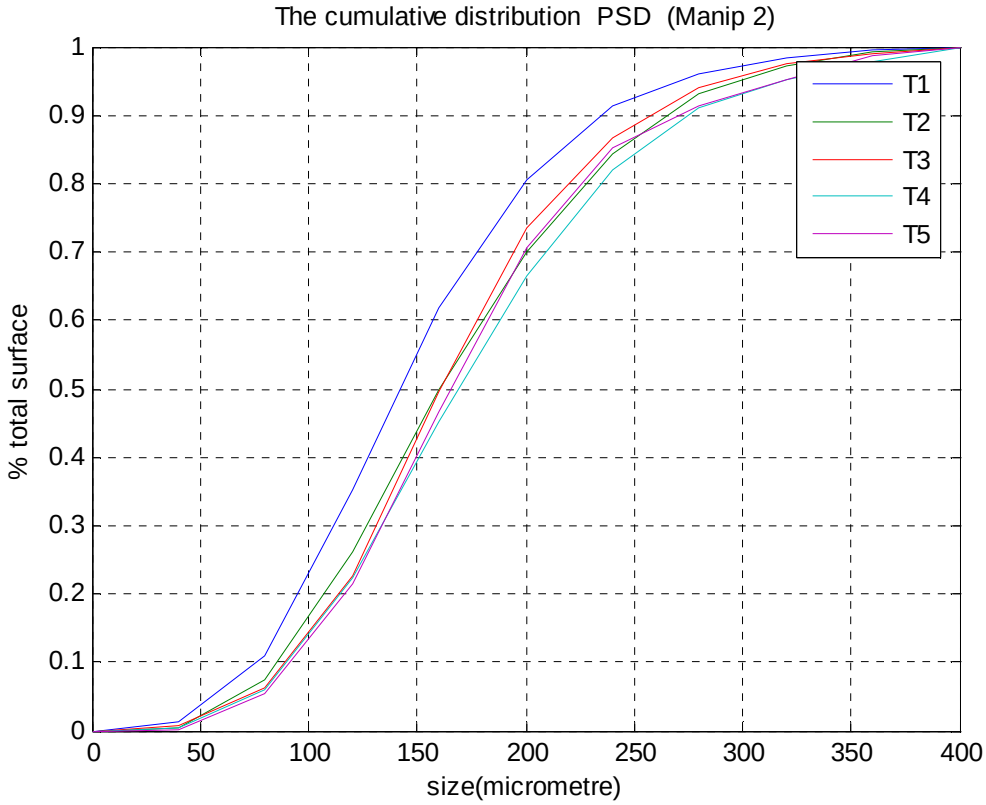

The cumulative distribution CLD (Manip 8)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
T6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

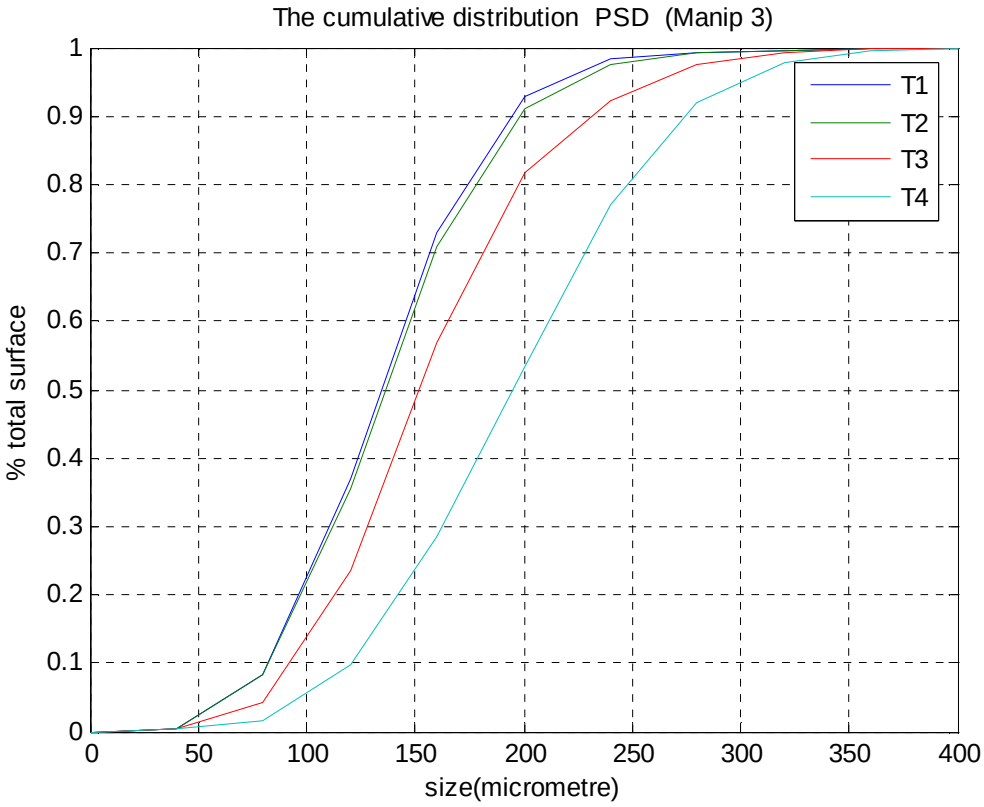

The cumulative distribution CLD (Manip 9)
% total surface
size(micrometre)
T1
T2
T3
T4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

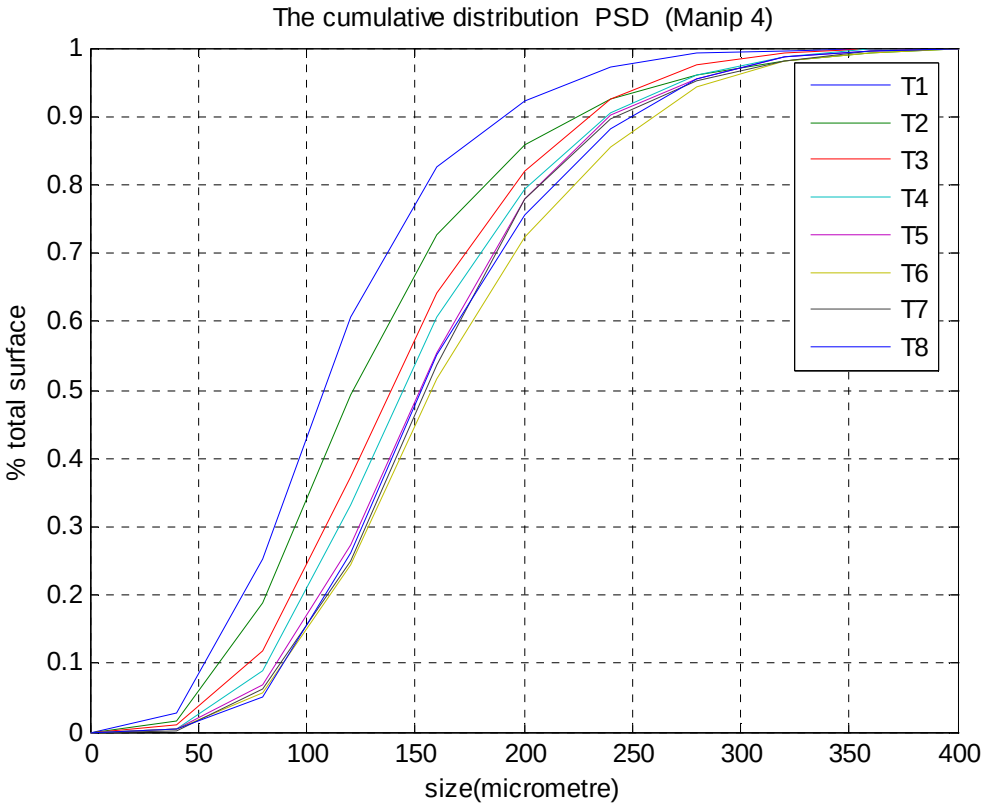

The cumulative distribution CLD (Manip 10)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
T6
T7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

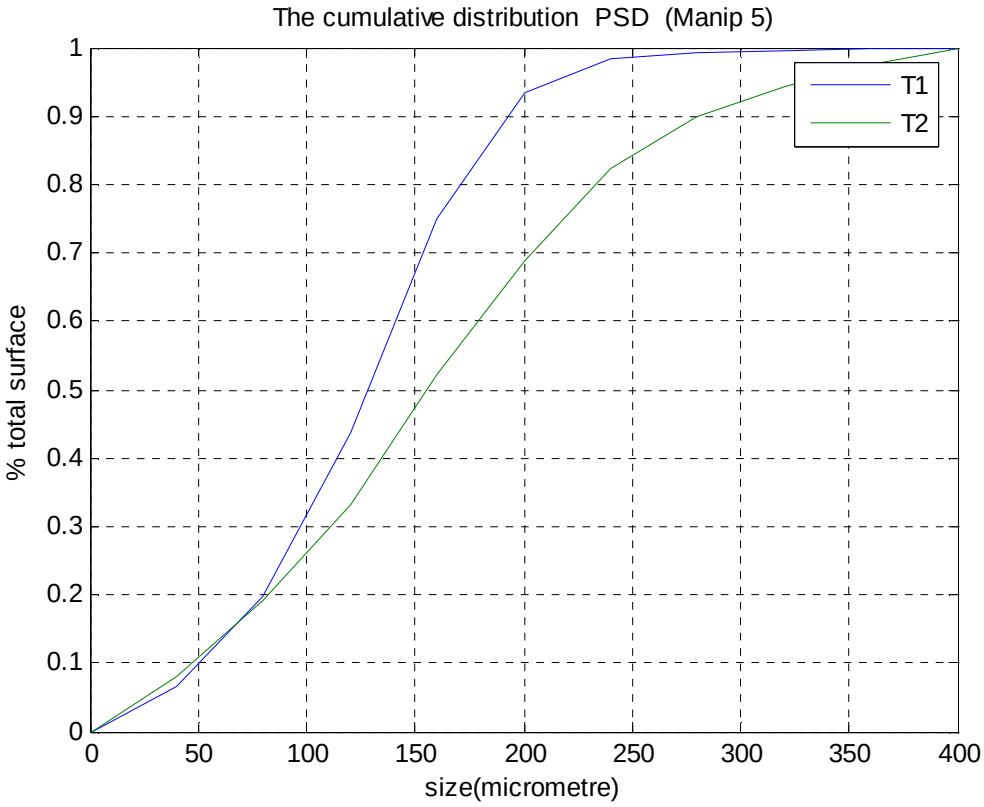
(2) The cumulative distribution of PSD estimated by image analysis

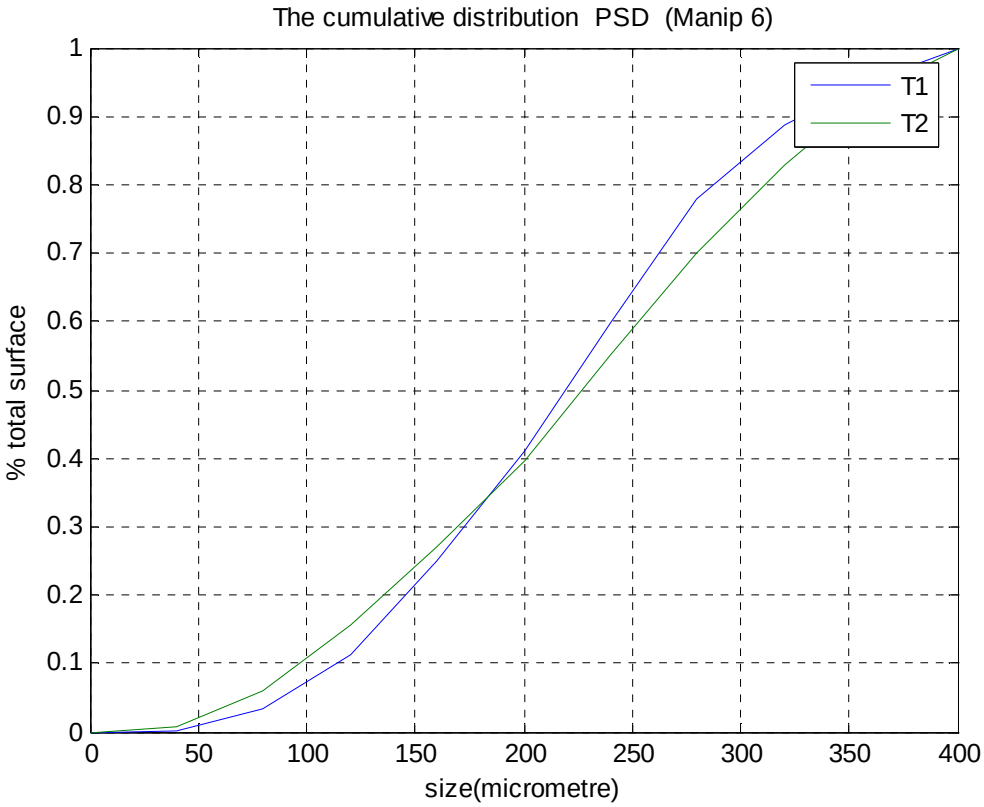


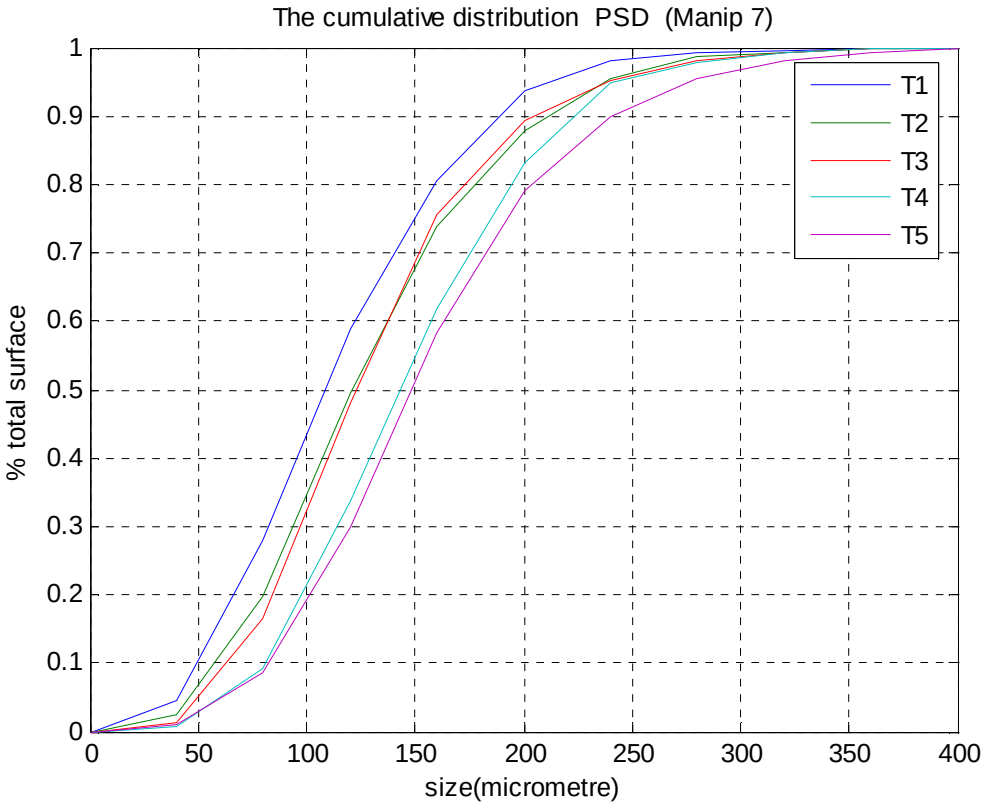


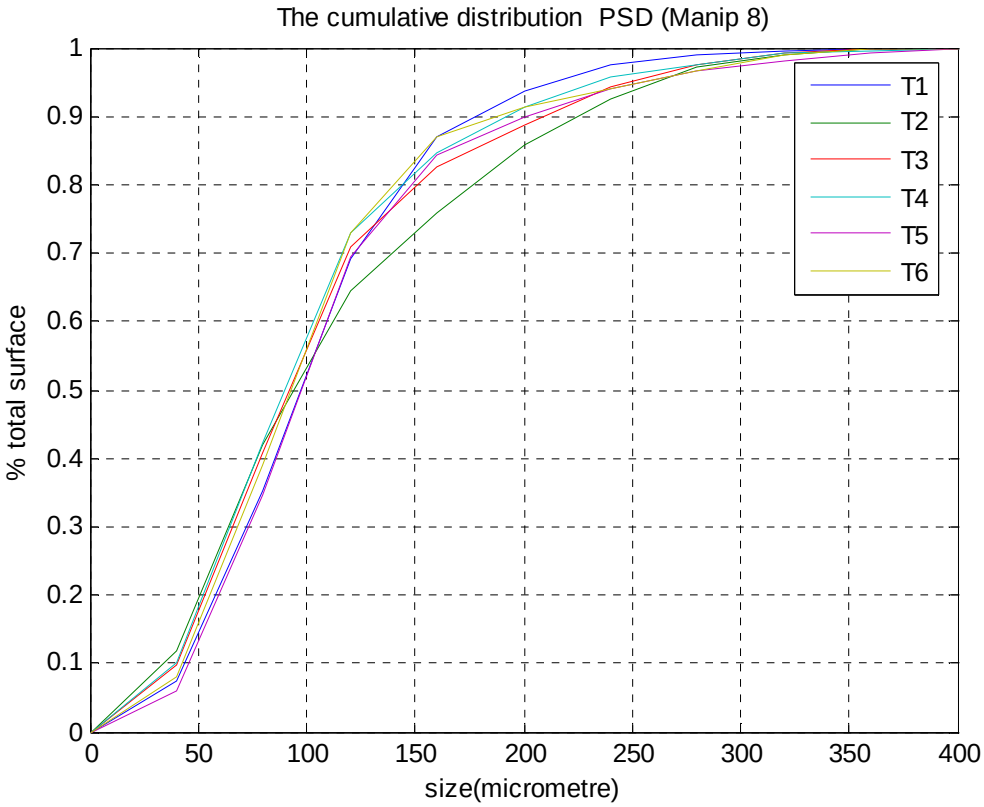

The cumulative distribution PSD (Manip 3)
% total surface
size(micrometre)
T1
T2
T3
T4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

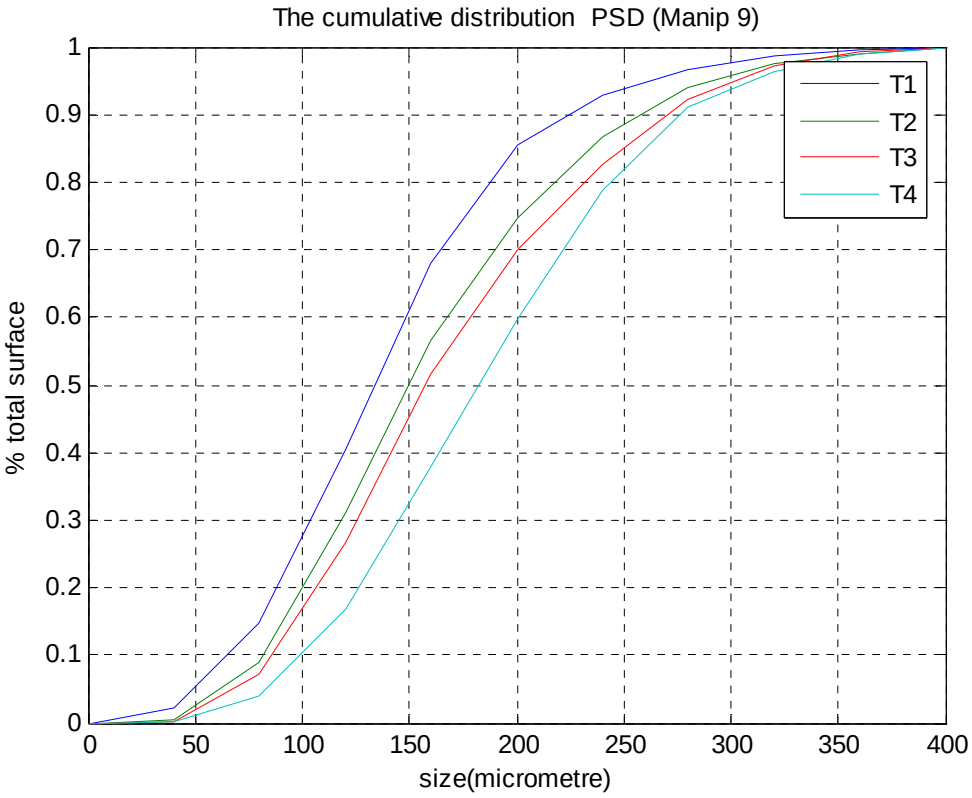

The cumulative distribution PSD (Manip 4)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
T6
T7
T8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

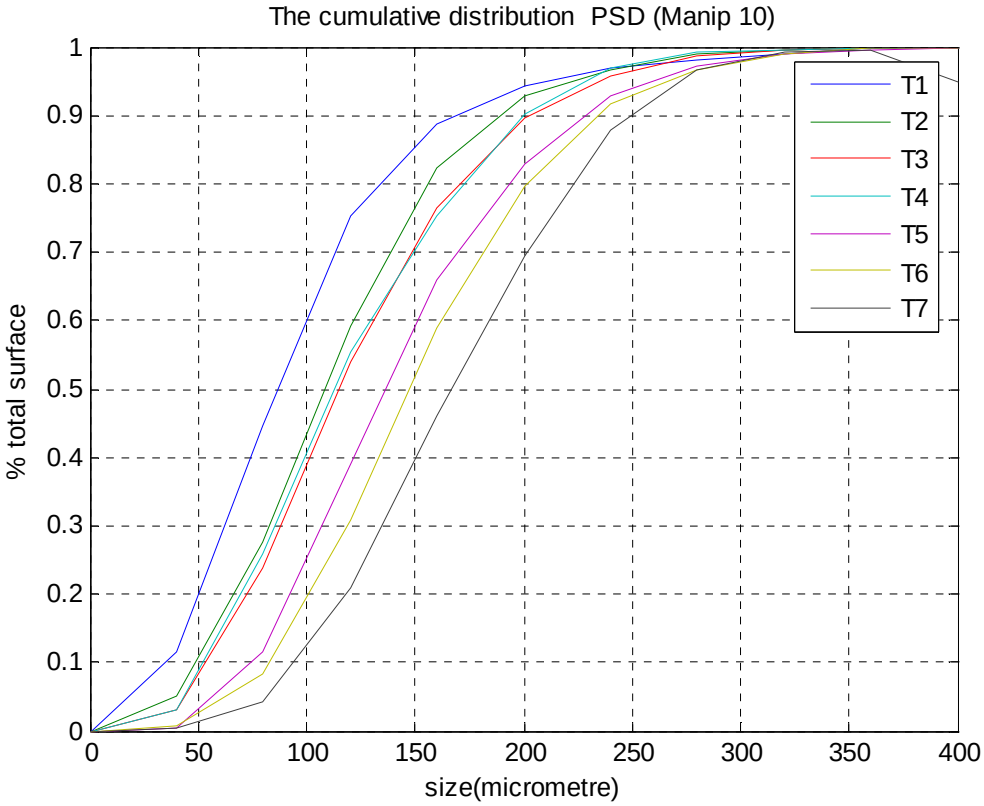

The cumulative distribution PSD (Manip 5)
T1
T2
% total surface
size(micrometre)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400


The cumulative distribution PSD (Manip 6)
T1
T2
% total surface
size(micrometre)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

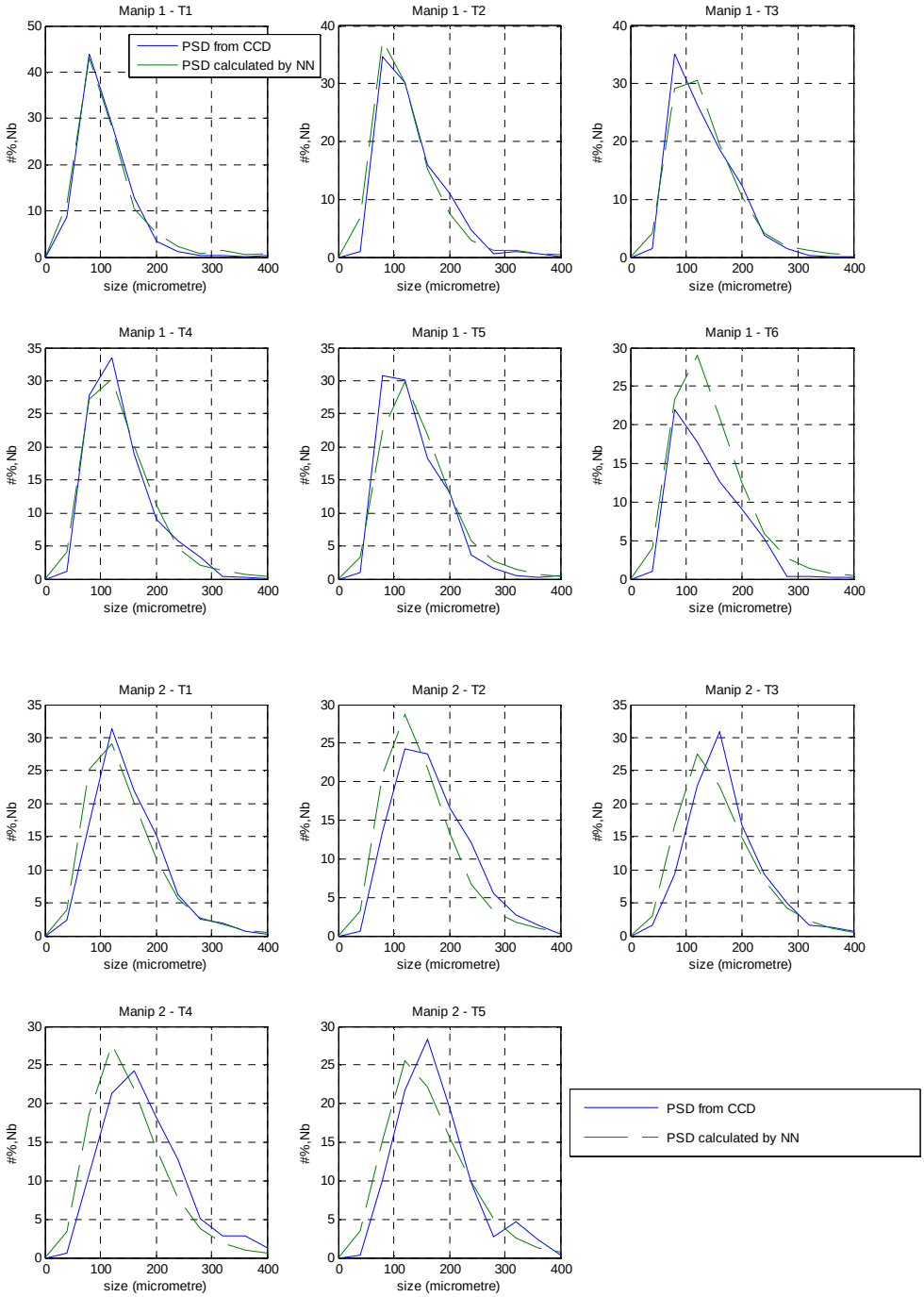

The cumulative distribution PSD (Manip 7)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

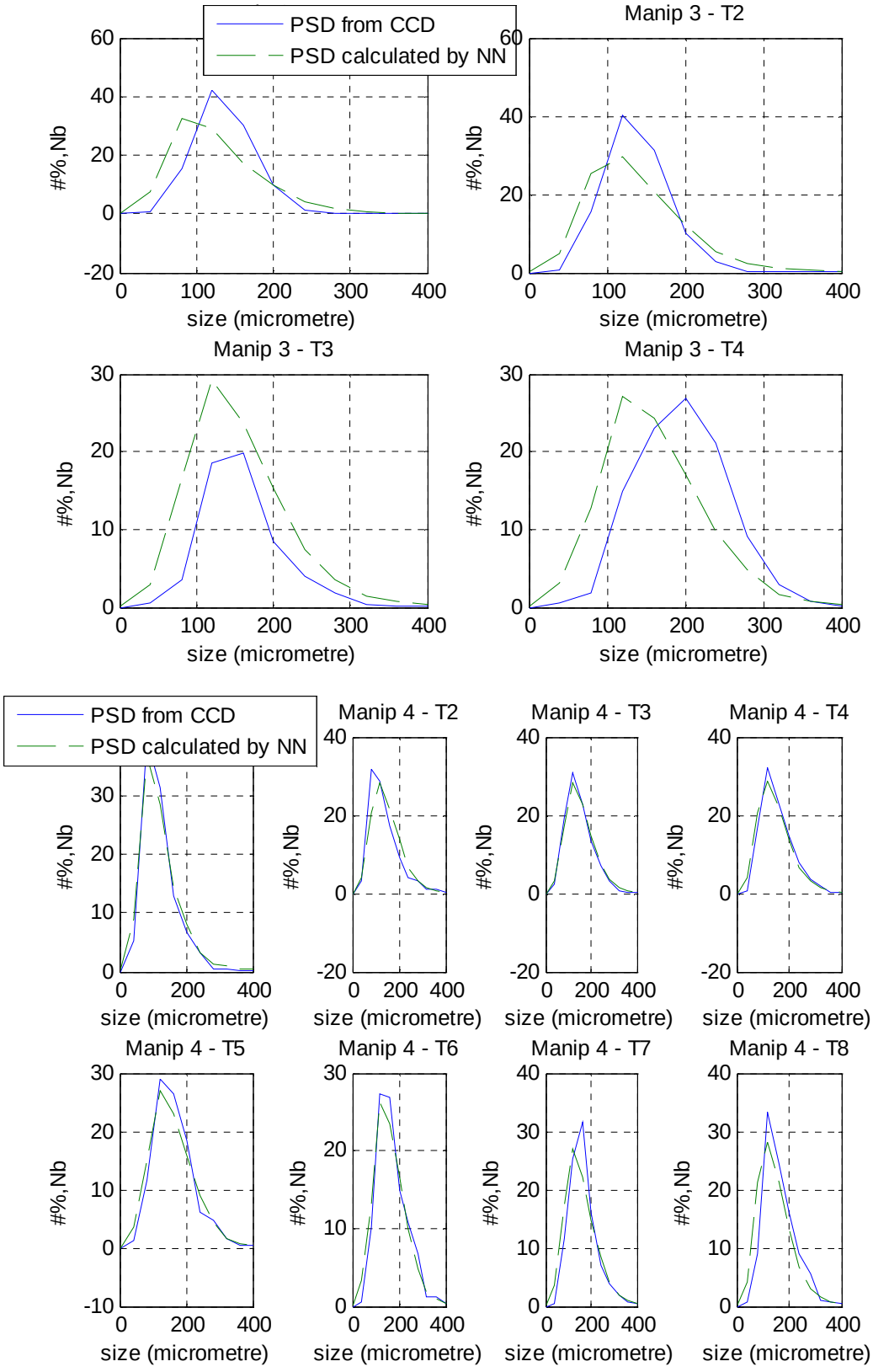

The cumulative distribution PSD (Manip 8)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
T6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

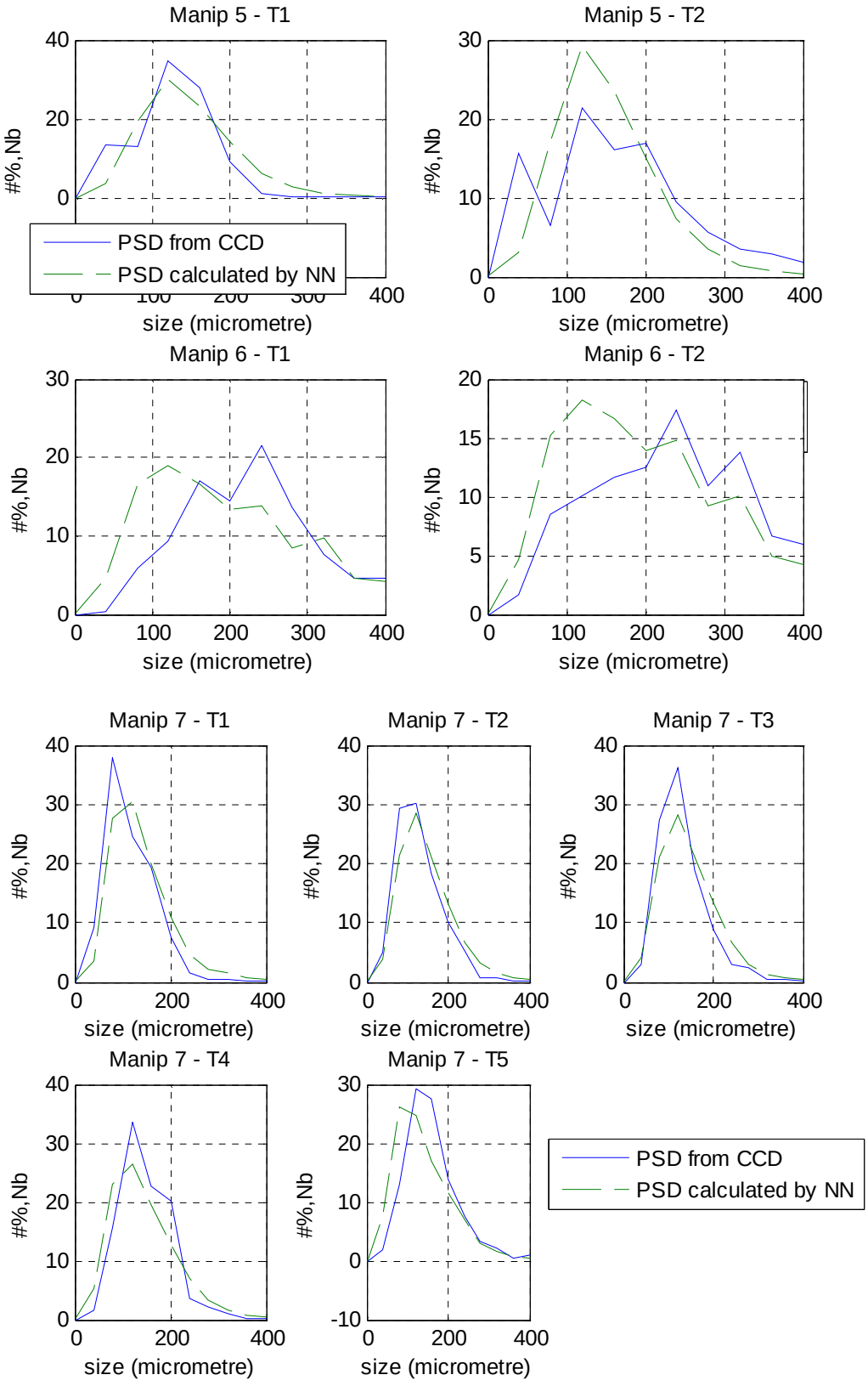

The cumulative distribution PSD (Manip 9)
% total surface
size(micrometre)
T1
T2
T3
T4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

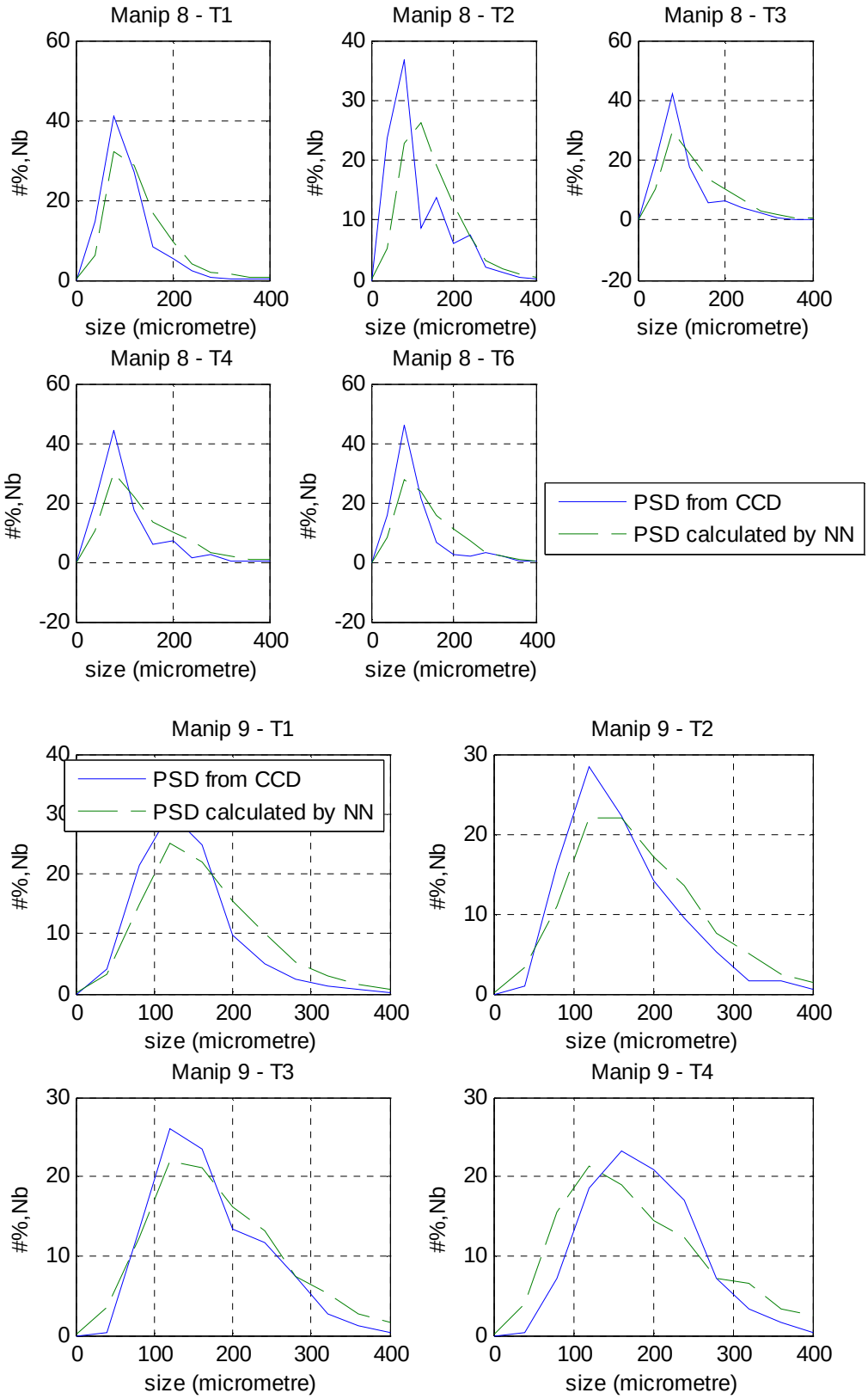

The cumulative distribution PSD (Manip 10)
% total surface
size(micrometre)
T1
T2
T3
T4
T5
T6
T7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400

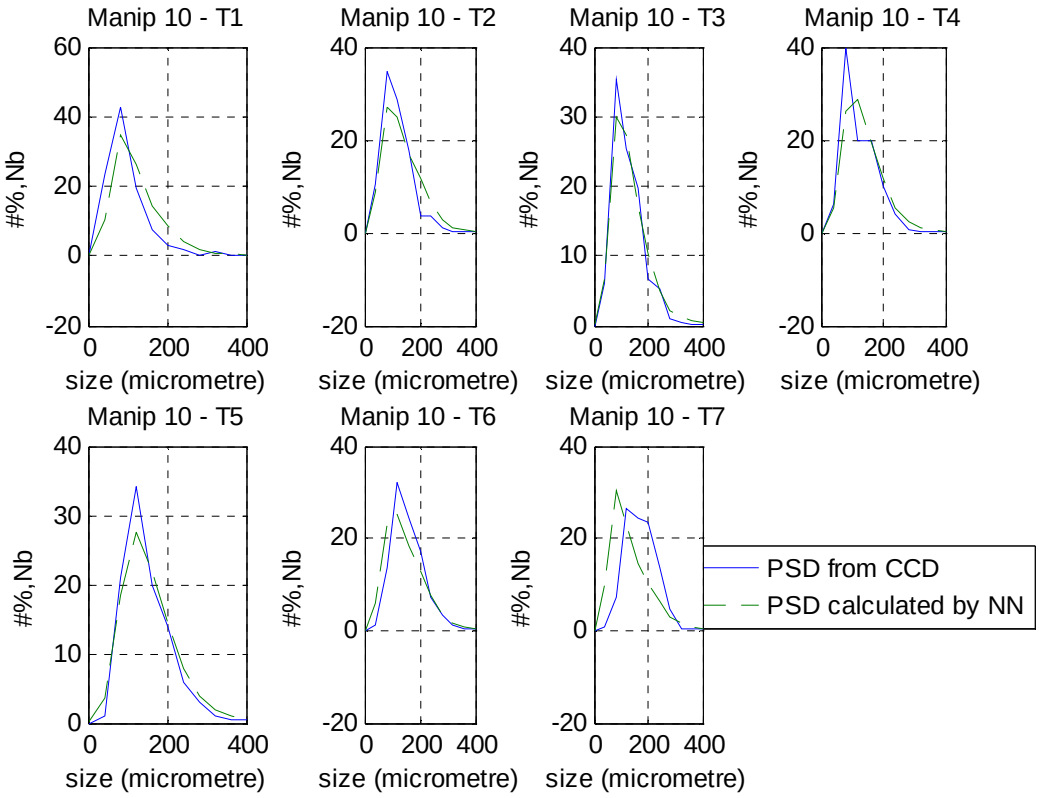
APPENDIX C The results of NN Model 1



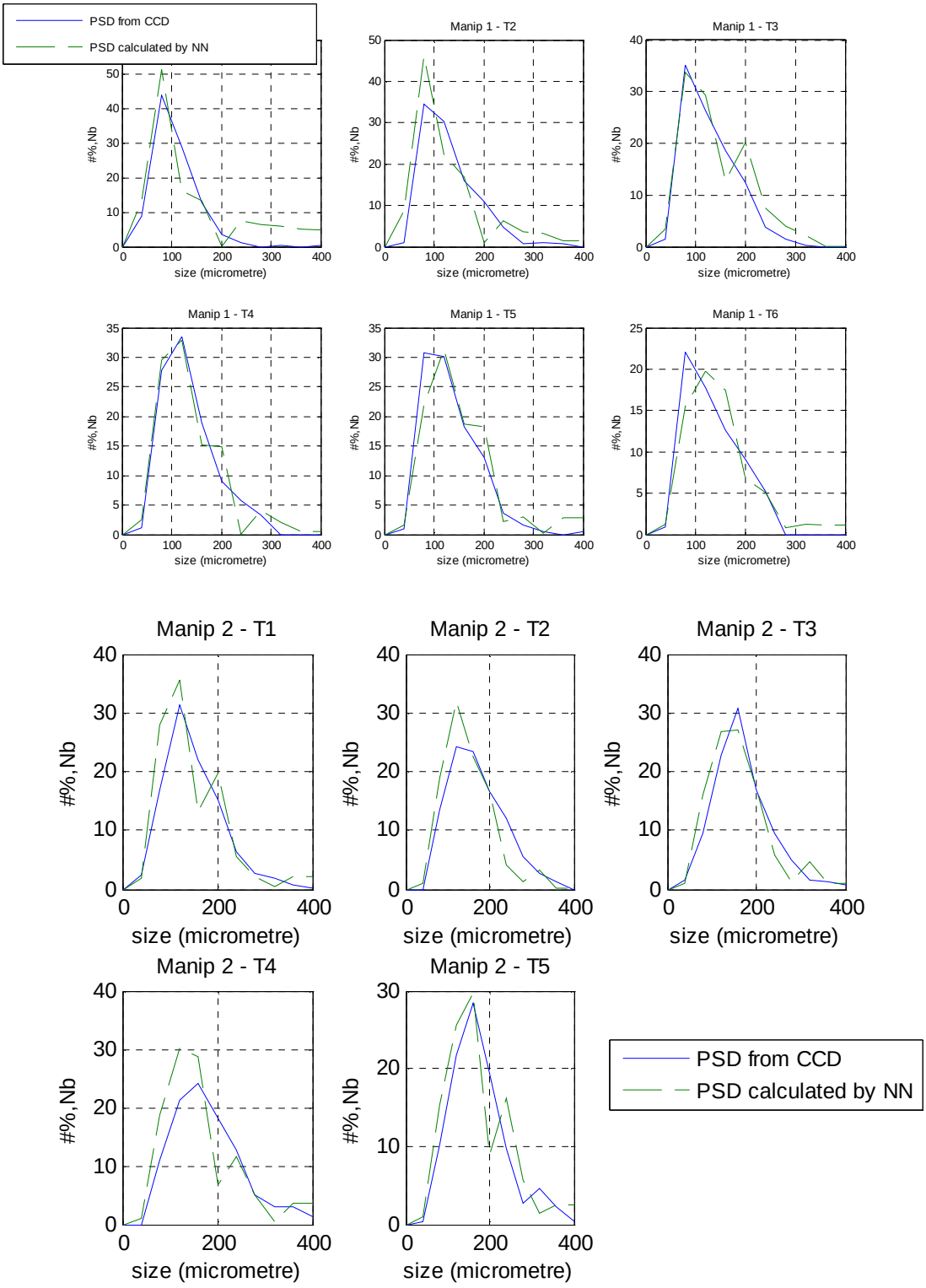

PSD from CCD
PSD calculated by NN
Manip 3 - T2
Manip 3 - T3
Manip 3 - T4
Manip 4 - T2
Manip 4 - T3
Manip 4 - T4
Manip 4 - T5
Manip 4 - T6
Manip 4 - T7
Manip 4 - T8
#%,Nb
size (micrometre)

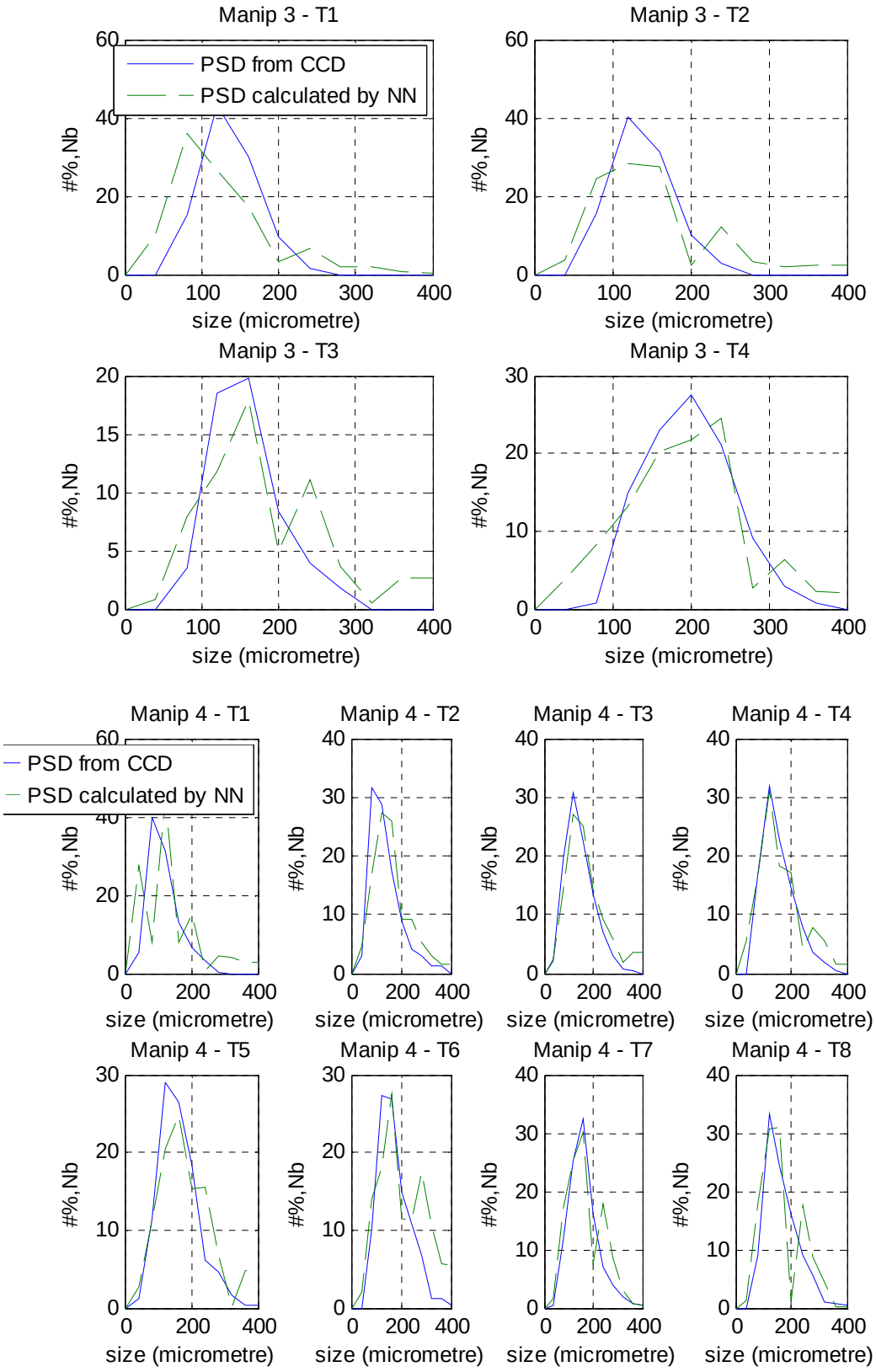

Manip 5 - T1
Manip 5 - T2
Manip 6 - T1
Manip 6 - T2
Manip 7 - T1
Manip 7 - T2
Manip 7 - T3
Manip 7 - T4
Manip 7 - T5
#%,Nb
size (micrometre)
PSD from CCD
PSD calculated by NN

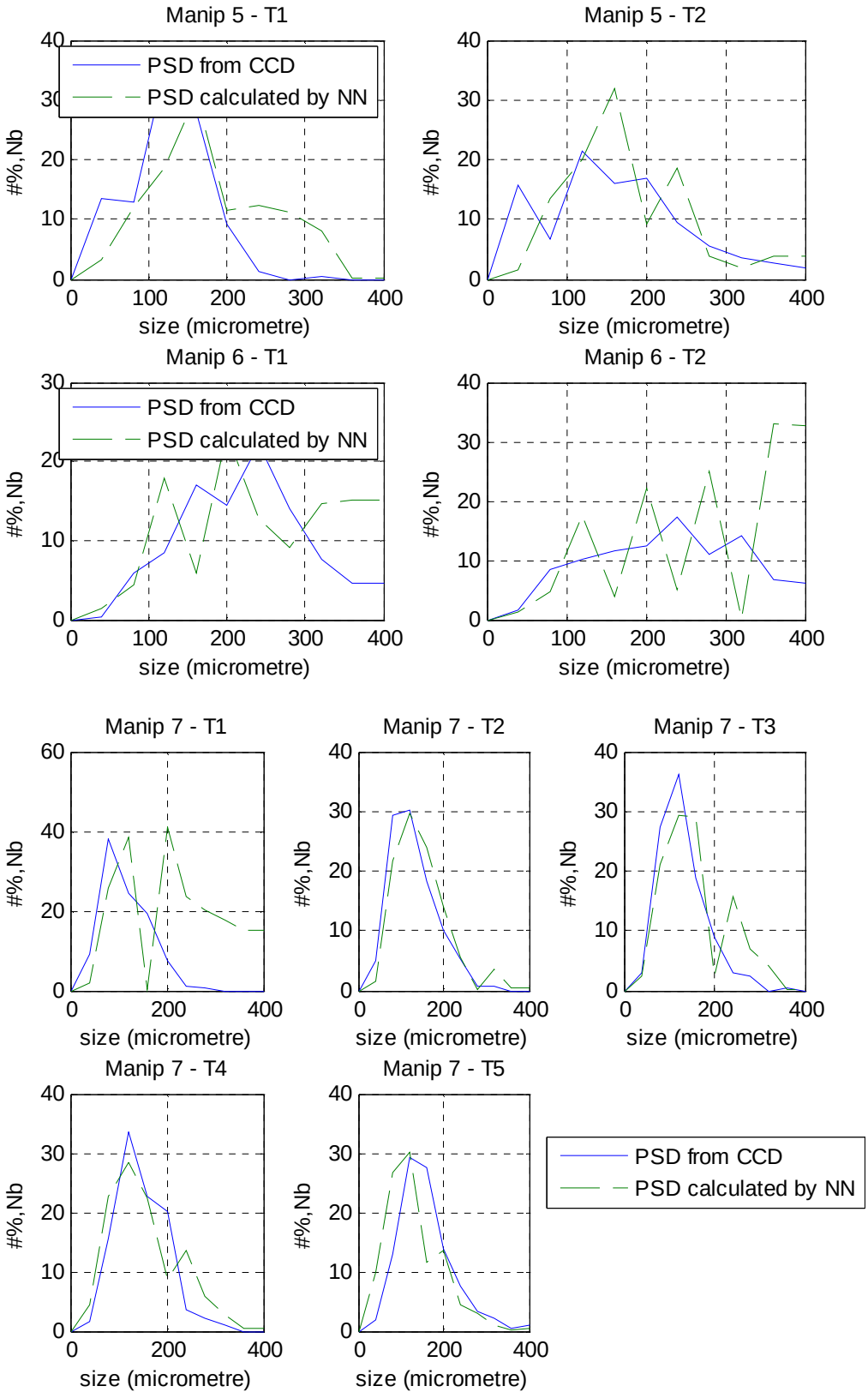

Manip 8 - T1
Manip 8 - T2
Manip 8 - T3
Manip 8 - T4
Manip 8 - T6
Manip 9 - T1
Manip 9 - T2
Manip 9 - T3
Manip 9 - T4
#%,Nb
size (micrometre)
PSD from CCD
PSD calculated by NN

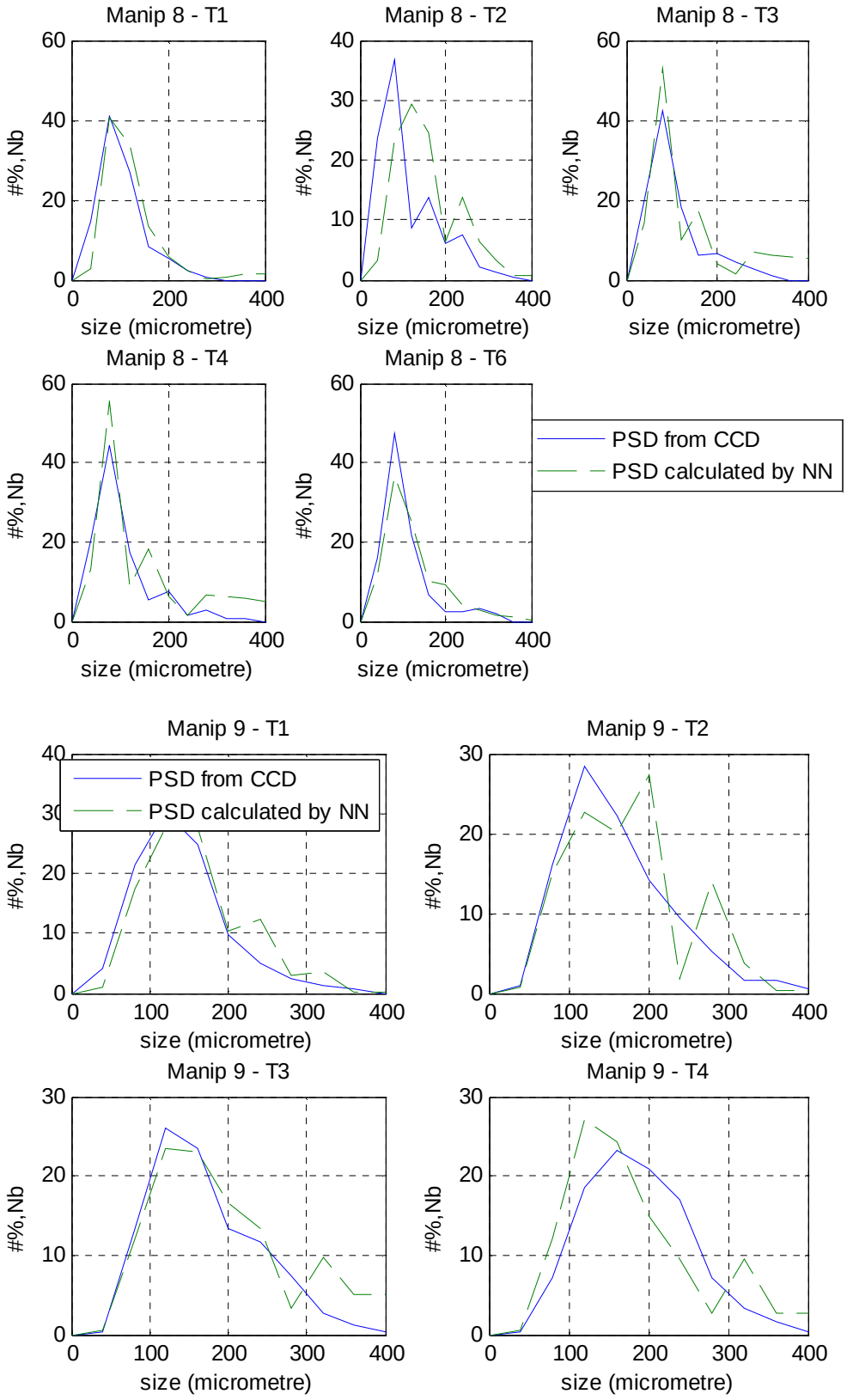

Manip 10 - T1
Manip 10 - T2
Manip 10 - T3
Manip 10 - T4
Manip 10 - T5
Manip 10 - T6
Manip 10 - T7
#%,Nb
size (micrometre)
PSD from CCD
PSD calculated by NN

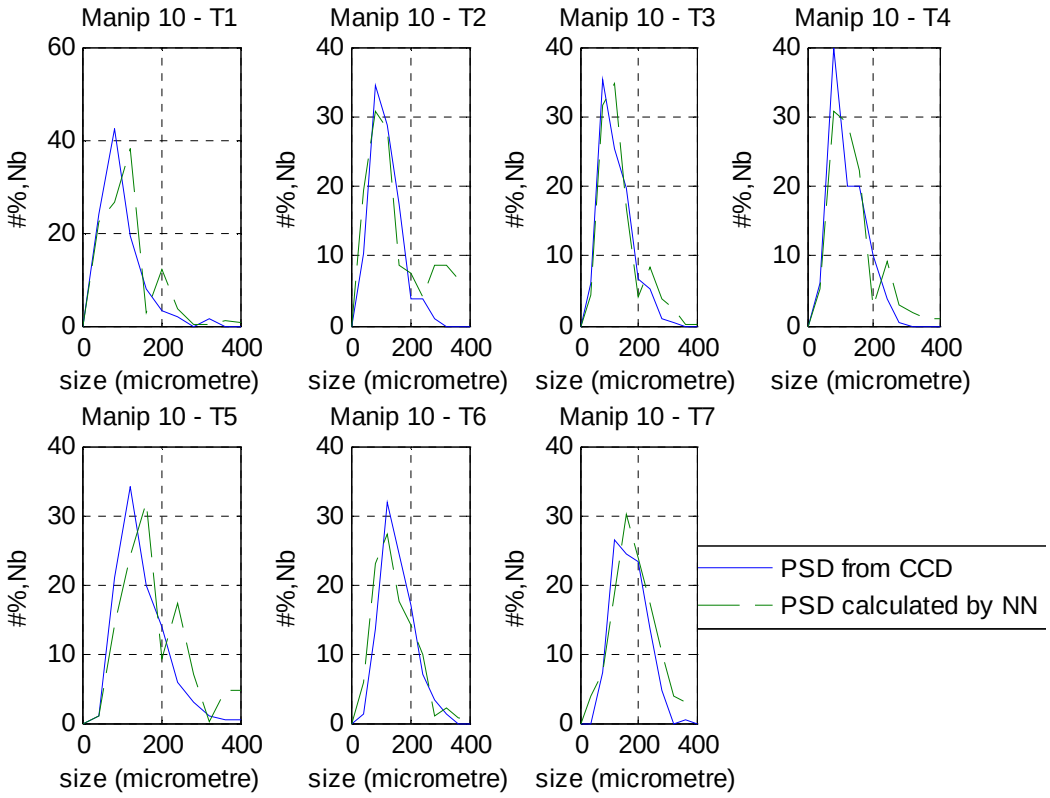
APPENDIX D The results of NN Model 2




Manip 3 - T1
PSD from CCD
PSD calculated by NN
#%,Nb
size (micrometre)
Manip 3 - T2
Manip 3 - T3
Manip 3 - T4
Manip 4 - T1
Manip 4 - T2
Manip 4 - T3
Manip 4 - T4
Manip 4 - T5
Manip 4 - T6
Manip 4 - T7
Manip 4 - T8


Manip 5 - T1
Manip 5 - T2
Manip 6 - T1
Manip 6 - T2
Manip 7 - T1
Manip 7 - T2
Manip 7 - T3
Manip 7 - T4
Manip 7 - T5
PSD from CCD
PSD calculated by NN
#%,Nb
size (micrometre)


Manip 8 - T1
Manip 8 - T2
Manip 8 - T3
Manip 8 - T4
Manip 8 - T6
#%,Nb
size (micrometre)
PSD from CCD
PSD calculated by NN
Manip 9 - T1
Manip 9 - T2
Manip 9 - T3
Manip 9 - T4


Manip 10 - T1
Manip 10 - T2
Manip 10 - T3
Manip 10 - T4
Manip 10 - T5
Manip 10 - T6
Manip 10 - T7
#%,Nb
size (micrometre)
PSD from CCD
PSD calculated by NN

APPENDIX E The glossary of the NN Toolbox.

adaption
Training method that proceeds through the specified sequence of inputs, calculating the output, error, and network adjustment for each input vector in the sequence as the inputs are presented.
architecture
Description of the number of the layers in a neural network, each layer's transfer function, the number of neurons per layer, and the connections between layers.
backpropagation learning rule
Learning rule in which weights and biases are adjusted by error-derivative (delta) vectors backpropagated through the network. Backpropagation is commonly applied to feedforward multilayer networks. Sometimes this rule is called the generalized delta rule.
batch
Matrix of input (or target) vectors applied to the network simultaneously. Changes to the network weights and biases are made just once for the entire set of vectors in the input matrix. (The term batch is being replaced by the more descriptive expression "concurrent vectors.")
batching
Process of presenting a set of input vectors for simultaneous calculation of a matrix of output vectors and/or new weights and biases.
Bayesian framework
Assumes that the weights and biases of the network are random variables with specified distributions.
bias
Neuron parameter that is summed with the neuron's weighted inputs and passed through the neuron's transfer function to generate the neuron's output.
cascade-forward network
Layered network in which each layer only receives inputs from previous layers.
classification
Association of an input vector with a particular target vector.
concurrent input vectors
Name given to a matrix of input vectors that are to be presented to a network simultaneously. All the vectors in the matrix are used in making just one set of changes in the weights and biases.
conjugate gradient algorithm
In the conjugate gradient algorithms, a search is performed along conjugate directions, which produces generally faster convergence than a search along the steepest descent directions.
connection
One-way link between neurons in a network.
connection strength
Strength of a link between two neurons in a network. The strength, often called weight, determines the effect that one neuron has on another.
cycle
Single presentation of an input vector, calculation of output, and new weights and biases.

distance
Distance between neurons, calculated from their positions with a distance function.
distance function
Particular way of calculating distance, such as the Euclidean distance between two vectors.
early stopping
Technique based on dividing the data into three subsets. The first subset is the training set, used for computing the gradient and updating the network weights and biases. The second subset is the validation set. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned. The third subset is the test set. It is used to verify the network design.
epoch
Presentation of the set of training (input and/or target) vectors to a network and the calculation of new weights and biases. Note that training vectors can be presented one at a time or all together in a batch.
error vector
Difference between a network's output vector in response to an input vector and an associated target output vector.
feedforward network
Layered network in which each layer only receives inputs from previous layers.
function approximation
Task performed by a network trained to respond to inputs with an approximation of a desired function.
generalization
Attribute of a network whose output for a new input vector tends to be close to outputs for similar input vectors in its training set.
generalized regression network
Approximates a continuous function to an arbitrary accuracy, given a sufficient number of hidden neurons.
global minimum
Lowest value of a function over the entire range of its input parameters. Gradient descent methods adjust weights and biases in order to find the global minimum of error for a network.
gradient descent
Process of making changes to weights and biases, where the changes are proportional to the derivatives of network error with respect to those weights and biases. This is done to minimize network error.
Hebb learning rule
Historically the first proposed learning rule for neurons. Weights are adjusted proportional to the product of the outputs of pre- and postweight neurons.
hidden layer
Layer of a network that is not connected to the network output (for instance, the first layer of a two-layer feedforward network).
initialization
Process of setting the network weights and biases to their original values.
input layer
Layer of neurons receiving inputs directly from outside the network.
input weights
Weights connecting network inputs to layers.

layer
Group of neurons having connections to the same inputs and sending outputs to the same destinations.
layer weights
Weights connecting layers to other layers. Such weights need to have nonzero delays if they form a recurrent connection (i.e., a loop).
learning
Process by which weights and biases are adjusted to achieve some desired network behavior.
learning rate
Training parameter that controls the size of weight and bias changes during learning.
learning rule
Method of deriving the next changes that might be made in a network or a procedure for modifying the weights and biases of a network.
Levenberg-Marquardt
Algorithm that trains a neural network 10 to 100 times faster than the usual gradient descent backpropagation method. It always computes the approximate Hessian matrix, which has dimensions n-by-n.
linear transfer function
Transfer function that produces its input as its output.
local minimum
Minimum of a function over a limited range of input values. A local minimum might not be the global minimum.
log-sigmoid transfer function
Squashing function of the form shown below that maps the input to the interval (0,1). (The toolbox function is logsig.)

$$f(n) = \frac{1}{1+e^{-n}}$$

mean square error function
Performance function that calculates the average squared error between the network outputs a and the target outputs t.
momentum
Technique often used to make it less likely for a backpropagation network to get caught in a shallow minimum.
neuron
Basic processing element of a neural network. Includes weights and bias, a summing junction, and an output transfer function. Artificial neurons, such as those simulated and trained with this toolbox, are abstractions of biological neurons.
output layer
Layer whose output is passed to the world outside the network.
overfitting
Case in which the error on the training set is driven to a very small value, but when new data is presented to the network, the error is large.
pass
Each traverse through all the training input and target vectors.
perceptron
Single-layer network with a hard-limit transfer function. This network is often trained with the perceptron learning rule.
perceptron learning rule

Learning rule for training single-layer hard-limit networks. It is guaranteed to result in a perfectly functioning network in finite time, given that the network is capable of doing so.

positive linear transfer function

Transfer function that produces an output of zero for negative inputs and an output equal to the input for positive inputs.

postprocessing

Converts normalized outputs back into the same units that were used for the original targets.

preprocessing

Transformation of the input or target data before it is presented to the neural network.

principal component analysis

Orthogonalize the components of network input vectors. This procedure can also reduce the dimension of the input vectors by eliminating redundant components.

quasi-Newton algorithm

Class of optimization algorithm based on Newton's method. An approximate Hessian matrix is computed at each iteration of the algorithm based on the gradients.

radial basis networks

Neural network that can be designed directly by fitting special response elements where they will do the most good.

radial basis transfer function

The transfer function for a radial basis neuron is

$$radbas(n) = e^{-n^2}$$

regularization

Modification of the performance function, which is normally chosen to be the sum of squares of the network errors on the training set, by adding some fraction of the squares of the network weights.

saturating linear transfer function

Function that is linear in the interval (-1,+1) and saturates outside this interval to -1 or +1. (The toolbox function is satlin.)

scaled conjugate gradient algorithm

Avoids the time-consuming line search of the standard conjugate gradient algorithm.

sigmoid

Monotonic S-shaped function that maps numbers in the interval (-∞,∞) to a finite interval such as (-1,+1) or (0,1).

simulation

Takes the network input p, and the network object net, and returns the network outputs a.

spread constant

Distance an input vector must be from a neuron's weight vector to produce an output of 0.5.

squashing function

Monotonically increasing function that takes input values between -∞ and +∞ and returns values in a finite interval.

sum-squared error

Sum of squared differences between the network targets and actual outputs for a given input vector or set of vectors.

supervised learning

Learning process in which changes in a network's weights and biases are due to the intervention of any external teacher. The teacher typically provides output targets.

symmetric hard-limit transfer function
Transfer that maps inputs greater than or equal to 0 to +1, and all other values to -1.
symmetric saturating linear transfer function
Produces the input as its output as long as the input is in the range -1 to 1. Outside that range the output is -1 and +1, respectively.
tan-sigmoid transfer function
Squashing function of the form shown below that maps the input to the interval (-1,1). (The toolbox function is tansig.)

$$f(n) = \frac{1}{1+e^{-n}}$$

target vector
Desired output vector for a given input vector.
test vectors
Set of input vectors (not used directly in training) that is used to test the trained network.
topology functions
Ways to arrange the neurons in a grid, box, hexagonal, or random topology.
training
Procedure whereby a network is adjusted to do a particular job. Commonly viewed as an offline job, as opposed to an adjustment made during each time interval, as is done in adaptive training.
training vector
Input and/or target vector used to train a network.
transfer function
Function that maps a neuron's (or layer's) net output n to its actual output.
unsupervised learning
Learning process in which changes in a network's weights and biases are not due to the intervention of any external teacher. Commonly changes are a function of the current network input vectors, output vectors, and previous weights and biases.
update
Make a change in weights and biases. The update can occur after presentation of a single input vector or after accumulating changes over several input vectors.
validation vectors
Set of input vectors (not used directly in training) that is used to monitor training progress so as to keep the network from overfitting.
weight function
Weight functions apply weights to an input to get weighted inputs, as specified by a particular function.
weight matrix
Matrix containing connection strengths from a layer's inputs to its neurons. The element wi,j of a weight matrix W refers to the connection strength from input j to neuron i.
weighted input vector
Result of applying a weight to a layer's input, whether it is a network input or the output of another layer.
Widrow-Hoff learning rule
Learning rule used to train single-layer linear networks. This rule is the predecessor of the backpropagation rule and is sometimes referred to as the delta rule.

APPENDIX F The programs in Matlab

Read me. txt
These programmers deal with the on-line monitor of PSD (Particle Size Distribution).

We have 10 manipulations; each manipulation has different times of monitor:

	nb of instant
manip 1	6(1-6)
manip 2	5(7-11)
manip 3	4(12-15)
manip 4	8(16-23)
manip 5	2(24-25)
manip 6	2(26-27)
manip 7	5(28-32)
manip 8	6(33-38)
manip 9	4(39-42)
manip 10	7(43-49)
Total	49

So we have 49 times of measurement (by FBRM and CCD)
The class of size is always 0:40:400 (11 classes) in each instant.

Software of acquisition : Lasentec FBRM Data Acquisition Control Interface 6.0
Software of exploitation the data : Lasentec FBRM Data Review

Data files : FBRM_10manip_nbtt.xls
AI_10manip_nbtt.xls

	Class of size	
T_0 T_{end}	The percentage #% corresponde each class of size	Total Nb in each time

T1-T6 --→Manip 1, T7-T11---→ Manip 2,... etc.. We can also get it from the table before (ReadMe_book1.xls).

*MAIN_10MANIP_V3.m: the main programmer. In this programmer we have some important segment:

1) segment_PLOT_10manip_EVAL.m: plot 49 curbs of CLD (Cord Length Distribution) and PSD.

2) segment_PREMNMX.m: preprocess all data
3) segment_NB_NEURONS.m: calculated the best number of neurons in hidden layer

4) segment_RESULT_10manip_EVAL.m: plot the result of NN and the real PSD from AI (Analysis of Image).
5) Sement_DIS_SURFACE_EVAL.m: plot the cumulative distribution of surface of CLD and PSD.

Result:
We can have the error analysis and all curbs of simulation of NN Model 1 and Model 2.

List of programme:
- MAIN_10MANIP_V3.m
- segment_PLOT_10manip_EVAL.m
- segment_PREMNMX.m
- segment_NB_NEURONS.m
- segment_RESULT_10manip_EVAL.m
- Sement_DIS_SURFACE_EVAL.m

```
%Main programme deal with the 10 manipulations
%   -we have 49 times of measurement of FBRM during this manip. and 49
%    times of monitoring by CCD.
%   -we are intrested in the CLD and PSD, who's size is between 0-400
%    micrometre.

%   -at first, i read all data from files excel and normalize them ( so
%    that minimum is -0.95 and maximum is 0.95)
%   -than, i will built a NN like a software sensor
%   -using Cross validation for picking up the best NN (deciding the nb of
%    neuron in hidden layers.
%   -finally, i do the simulation

%-------------------------------------------------------------------------%
%   TRAINBR
%   10 manipulations
%   The total nb can be the input and output variable of NN will be decided
%   49 times of measurement(by FBRM and CCD)
%   11 class of size(0:40:400)
%-------------------------------------------------------------------------%

clear all
%The used parameter: Beginning

L=40; %longue of sampling(micrometre)
TL=400;%total longue(micrometre)
r=0:L:TL; %the class of size
ff=40;%parameter for nb of figure

%The used parameter: Ending
```

```
choice=input('Warning : lunch this ''init'' part only the 1st time ! : lunch it (1) or not (2) ? : ...')
%keyboard

if (choice==1)
   % init : beginning
   clear all
   close all
   clc

   %Read datas of FBRM and AI, then store them in data1
   data.FBRM=importdata('D:\2004-2005\STAGE2\WORK\DATA\FBRM_10manip_nbtt.xls');
   data.AI=importdata('D:\2004-2005\STAGE2\WORK\DATA\AI_10manip_nbtt.xls');

   %Read the information about the manipulation
   INFO=importdata('D:\2004-
2005\STAGE2\WORK\PROGRAM_FILE_10MANIP\ReadMe_book1.xls');%INFO.data
   no_manip=INFO.data(:,1);    %Numero of manipulation
   nb_instant=INFO.data(:,2);  %number of instant of each manipulation
   % init : end
end

sizeofFBRM=size(data.FBRM)
sizeofAI=size(data.AI)
%
corde          = data.FBRM(1,1:sizeofFBRM(2)-1)';          %Matrix 11*1,micrometre
percent.FBRM    = data.FBRM(2:sizeofFBRM(1),1:sizeofFBRM(2)-1)';       %Matrix 11*49,percent of
nb(#%)
nbtt.FBRM       = data.FBRM(2:sizeofFBRM(1),sizeofFBRM(2))';         %Matrix 1*49, nb total of each
time from FBRM
%
classes        = data.AI(1,1:sizeofAI(2)-1)';            %Matrix 11*1,micrometre
percent.AI      = data.AI(2:sizeofAI(1),1:sizeofAI(2)-1)';         %Matrix 11*49,percent of nb(#%)
nbtt.AI        = data.AI(2:sizeofAI(1),sizeofAI(2))';           %Matrix 1*49, nb total of each time from AI


%-Fin of reading datas-

%choice=input('Warning : plot the  CLD-FBRM and PSD-AI ! : lunch it (1) or not (2) ? : ...')
%keyboard

%if (choice==1)
  %segment_PLOT_10manip_EVAL
%end

% %
% % %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%-%-%
% % %Beginning of NN  1                                             %
% % %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%-%-%
% %Preprocess all input data and output data, so that minimum is -0.95 and  %
% %maximum is 0.95
%
% segment_PREMNMX_10manip
%
% %Fin de PREMNMX, we have the *_n.FBRM and *_n.AI for the ANN1
%
% indata=percent_n.FBRM;
% target=percent_n.AI;
```

```
%
% % indata=[percent_n.FBRM
% %         nbtt_n.FBRM];
% % target=[percent_n.AI
% %         nbtt_n.AI];
%
% [Q,R]=size(indata);   %R must be 49 bcs we are 49 times of measurement,Q=11
% [S,R]=size(target);
%
%
% %segment_NB_NRURONS
%
% %Re-train the net using the best nb of neurons--------Program 1.-----------
%
% nb_neurons1=8;
%
%     net=newff(minmax(indata),[nb_neurons1 S],{'tansig','tansig'},'trainbr');
%
%     training_in     = indata(:,1:2:R);
%     testset.P       = indata(:,2:2:R);
%
%     training_target = target(:,1:2:R);
%     testset.T       = target(:,2:2:R);
%
%     %net.performFcn='sse';
%     net.trainParam.epochs=1000;
%     %net.trainParam.min_grad=1e-100;
%     %net.trainParam.mu=1e-50;
%
%    [net1,tr1]=train(net,training_in,training_target,[],[],testset);
%
%     figure
%     plot(tr1.epoch,tr1.perf/(Q*R),tr1.epoch,tr1.vperf/(Q*R),'--')
%
% %     title('1/3 data-Training set,2/3 data-Test set,TRAINBR number of hidden neurons=8')
% %     xlabel('Epoch')
% %     ylabel('MSE')
% %     legend('Training set','Test set')
%
%
% %-Fin of trainning-
%
%
% %Store the error of test set
% err1=tr1.vperf/(Q*R);
% %plot(tr1.epoch,err1)
%
%
% %Begin of simulation
% sim_input=percent_n.FBRM;
% sim_output=sim(net1,sim_input);
% sim_output=sim_output(1:Q,:);
% err_TRAIN_NN1=sse(sim_output-percent_n.AI)/(Q*R)
%
% %Postprocess the data so that we can have the finally result.
% output1=postmnmx(sim_output,minpercentAI,maxpercentAI);
% percent.AI=postmnmx(percent_n.AI,minpercentAI,maxpercentAI);
%
% fin_err1=mse(output1-percent.AI)
% %Plot the figures for compare the results of ANN and PSD of AI
```

```
% target=percent.AI;
% output=output1;
%
% % choice=input('Warning : plot the result of simulation, compare the PSD-NN1 and PSD-AI ! : lunch
it (1) or not (2) ? : ...')
% % keyboard
% %
% % if (choice==1)
%   segment_RESULT_10manip_EVAL
% %end
%

% %
% % %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%-%-%
% % %Beginning of NN  2                                    %
% % %-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%-%-%
%
%
% %Calculation of the distribution cumulative of CLD and PSD
%
% [Q,R]=size(percent.FBRM);   %R must be 49 bcs we are 49 times of measurement,Q=11
% [S,R]=size(percent.AI);
%
% %SURFACE_* :        matrix who store each instant of surface
% %percent_SURFACE_*: matrix who store the distribution cumulative in each
% %                  instant
%
% SURFACE_FBRM=zeros(Q,R);
% for n=1:R
%     for jj=2:Q
%         SURFACE_FBRM(jj,n)=(percent.FBRM(jj,n)+percent.FBRM(jj-1,n))*L/2;
%     end
% end
%
% percent_SURFACE_FBRM=zeros(Q,R);
% for n=1:R
%     SUM_row=sum(SURFACE_FBRM(:,n));
%     for jj=1:Q
%         percent_SURFACE_FBRM(jj,n)=sum(SURFACE_FBRM(1:jj,n))/SUM_row;
%     end
% end
%
%
% %-----------------------%
%
% SURFACE_AI=zeros(Q,R);
% for n=1:R
%     for jj=2:Q
%         SURFACE_AI(jj,n)=(percent.AI(jj,n)+percent.AI(jj-1,n))*L/2;
%     end
% end
%
% percent_SURFACE_AI=zeros(Q,R);
% for n=1:R
%     SUM_row=sum(SURFACE_AI(:,n));
%     for jj=1:Q
%         percent_SURFACE_AI(jj,n)=sum(SURFACE_AI(1:jj,n))/SUM_row;
%     end
```

```
% end
% %Fin of Calculation of the distribution cumulative of CLD and PSD
%
% %Re-train the net using the best nb of neurons--------Program 2.----------
% sizeofF=size(percent_SURFACE_FBRM);
% sizeofA=size(percent_SURFACE_AI);
%
% for ii=sizeofF(1)
%     for jj=sizeofF(2)
%         if percent_SURFACE_FBRM(ii,jj)==1
%             percent_SURFACE_FBRM(ii,jj)=0.95;
%         end
%         if percent_SURFACE_FBRM(ii,jj)==0
%             percent_SURFACE_FBRM(ii,jj)=0.5;
%         end
%     end
% end
%
% for ii=sizeofA(1)
%     for jj=sizeofA(2)
%         if percent_SURFACE_AI(ii,jj)==1
%             percent_SURFACE_AI(ii,jj)=0.95;
%         end
%         if percent_SURFACE_AI(ii,jj)==0
%             percent_SURFACE_AI(ii,jj)=0.5;
%         end
%     end
% end
%
% %choice=input('Warning : plot the cumulative distribution of CLD and PSD ! : lunch it (1) or not
(2) ? : ...')
% %keyboard
%
% %if (choice==1)
% %   segment_DIS_SURFACE_10manip
% %end
% %
%
% indata=percent_SURFACE_FBRM;
% target=percent_SURFACE_AI;
%
% [Q,R]=size(indata);   %R must be 49 bcs we are 49 times of measurement
% [S,R]=size(target);
%
% %segment_NB_NRURONS_NN2
%
% nb_neurons2=23;
%
%
% nn=0;
% while nn<1
%
%     net=newff(minmax(indata),[nb_neurons2 S],{'logsig','logsig'},'trainbr');
%
%     training_in     = indata(:,1:2:R);
%     testset.P       = indata(:,2:2:R);
%
%     training_target =target(:,1:2:R);
%     testset.T       =target(:,2:2:R);
%
```

```
%     %net.performFcn='sse';
%     net.trainParam.epochs=1000;
%     %net.trainParam.goal=1e-10;
%     %net.trainParam.min_grad=1e-100;
%     %net.trainParam.mu=1e-50;
%
%    [net2,tr2]=train(net,training_in,training_target,[],[],testset);
% %
% %      figure
% %      plot(tr2.epoch,tr2.perf/(Q*R),tr2.epoch,tr2.vperf/(Q*R),'--')
% %      title('Training fun: TRAINBR')
% %      xlabel('Epoch')
% %      ylabel('MSE')
% %      legend('Training set','Test set')
%
%    output_prog=sim(net2,indata);
%    output_prog=output_prog(1:11,:);
%
%    count=0;
%    for n=1:R
%        for jj=2:Q
%            if output_prog(jj,n)<output_prog(jj-1,n)
%                count=count+1;
%            end
%        end
%    end
%    if count<=0 break
%    end
%    nn=nn+1;
% end
%
%
%
% %-Fin of trainning-
%
% %Store the error of test set
% err2=tr2.vperf/(Q*R);
%
% % %Plot the errors of two NN
% % figure
% % plot(tr1.epoch,err1,tr2.epoch,err2)
% % title('Compare the errors of two training')
% % xlabel('epoch')
% % ylabel('error')
% % legend('err1','err2')
%
% sim_input=percent_SURFACE_FBRM;
% sim_output02=sim(net2,sim_input);
% sim_output02=sim_output02(1:11,:);
%
% sim_output2=sim_output02;
%
%
% err_TRAIN_NN2=sse(sim_output2-percent_SURFACE_AI)/(Q*R)
%
% %refind the PSD
% PSD_NN=zeros(Q,R);
% for n=1:R
%     surface_tn=sum(SURFACE_AI(:,n));
%     for jj=2:Q
```

```
%
%       if sim_output2(jj,n)<sim_output2(jj-1,n)
%          sim_output2(jj,n)=sim_output2(jj-1,n)+1e-5;
%       end
%
%       PSD_NN(jj,n)=abs(surface_tn*abs(sim_output2(jj,n)-sim_output2(jj-1,n))*2/L-PSD_NN(jj-1,n));
%    end
% end

% ff=50;

% target=percent.AI;
% output=PSD_NN;
% %choice=input('Warning : plot the result of simulation, compare the PSD-NN2 and PSD-AI ! : lunch
it (1) or not (2) ? : ...')
% %keyboard
%
% %if (choice==1)
% %   segment_RESULT_10manip_EVAL
% %end
%
% fin_err2=mse(PSD_NN-percent.AI)
%
%



%segment_PLOT_10manip_EVAL.m
%figure 1-10: 10 CLD-FBRM
%figure 11-20: 10 PSD-AI
%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%
%       Plot the figure of measurment of FBRM and AI                    %
%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%
% no_manip
% nb_instant

end_f=0;

for ii=1:size(no_manip)
   begin_f=1+end_f;
   end_f=begin_f-1+nb_instant(ii);
   n=1;
   for jj=begin_f:end_f
      figure(ii)
      subplot(2,4,n)
      plot(corde,percent.FBRM(:,jj))
      title(['CLD-FBRM   Manip ',num2str(ii),'Instant ',num2str(n)])
      xlabel('Corde Length (micrometre)')
      ylabel('#% Nombre')
      grid

      n=n+1;

   end

end
```

```
end_f=0;

for ii=1:size(no_manip)
   begin_f=1+end_f;
   end_f=begin_f-1+nb_instant(ii);
   n=1;
   for jj=begin_f:end_f
      figure(10+ii)
      subplot(2,4,n)
      plot(corde,percent.AI(:,jj))
      title(['PSD-AI   Manip ',num2str(ii),'Instant ',num2str(n)])
      xlabel('Corde Length (micrometre)')
      ylabel('#% Nombre')
      grid

      n=n+1;

   end

end


%----------------------Fin of segment_PLOT_10manip_EVAL.m-------------%


%segment_PREMNMX_10manip.m
%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%
%Preprocess all input data and output data, so that minimum is -0.95 and  %
%maximum is 0.95                                                  %
%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-%-
%
[percent_n.FBRM,minpercentF,maxpercentF]=premnmx(percent.FBRM);
[percent_n.AI,minpercentAI,maxpercentAI]=premnmx(percent.AI);

[nbtt_n.FBRM,minpercentNBF,maxpercentNBF]=premnmx(nbtt.FBRM);
[nbtt_n.AI,minpercentNBAI,maxpercentNBAI]=premnmx(nbtt.AI);

size_percentFBRM=size(percent_n.FBRM);
size_percentAI=size(percent_n.AI);
size_percentNBF=size(nbtt_n.FBRM);
size_percentNBAI=size(nbtt_n.AI);

for ii=1:size_percentFBRM(1)
   for jj=1:size_percentFBRM(2)
      if percent_n.FBRM(ii,jj)==1 percent_n.FBRM(ii,jj)=0.95;end
      if percent_n.FBRM(ii,jj)==-1 percent_n.FBRM(ii,jj)=-0.95;end
   end
end

for ii=1:size_percentAI(1)
   for jj=1:size_percentAI(2)
      if percent_n.AI(ii,jj)==1 percent_n.AI(ii,jj)=0.95;end
      if percent_n.AI(ii,jj)==-1 percent_n.AI(ii,jj)=-0.95;end
   end
end

for ii=1:size_percentNBF(1)
   for jj=1:size_percentNBF(2)
      if nbtt_n.FBRM(ii,jj)==1 nbtt_n.FBRM(ii,jj)=0.95;end
```

```
        if nbtt_n.FBRM(ii,jj)==-1 nbtt_n.FBRM(ii,jj)=-0.95;end
    end
end

for ii=1:size_percentNBAI(1)
    for jj=1:size_percentNBAI(2)
        if nbtt_n.AI(ii,jj)==1 nbtt_n.AI(ii,jj)=0.95;end
        if nbtt_n.AI(ii,jj)==-1 nbtt_n.AI(ii,jj)=-0.95;end
    end
end
%-------------------------Fin of Preprocess--------------------------%
%--------------------------------------------------------------------%

%segment_NB_NRURONS.m
%Calculat the best nb of neurons in hidden layers, return the No of training

ErrMat_test=[];
ErrMat_training=[];

epochs=[];
nb_neurons_serie=[];

%The iterations,train the simultaneously and then pick up a good network.

for nb_neurons=6:30
    net0=network;
    net0=newff(minmax(indata),[nb_neurons S],{'tansig','tansig'},'trainbr');%This net has 90 inputs
correspond the 11 classes of cord in FBRM and nb_neuron neurons in the intermediate layer, 11
outputs (correspond the chosen size classes in AI).
    net=net0;

    training_in     = indata(:,1:2:R);
    testset.P       = indata(:,2:2:R);

    training_target =target(:,1:2:R);
    testset.T       =target(:,2:2:R);

    %net.performFcn='sse';
    net.trainParam.epochs=1000;
    %net.trainParam.goal=1e-10;
    %net.trainParam.min_grad=1e-10;

    [net,tr]=train(net,training_in,training_target,[],[],testset);


    %Store the epochs and Error of training, Err of testing, the nb of
    %neurons in the hidden layer

    %err=tr.perf+tr.ssX;
    err=tr.perf;


    %err=tr.gamk;

    ErrMat_training=[ErrMat_training,err];
    epochs=[epochs,tr.epoch];
    nb_neurons_serie=[nb_neurons_serie,nb_neurons*ones(size(tr.epoch))];
    A=[nb_neurons_serie',epochs',ErrMat_training'];

end
```

```
%Chose the best NN
A(:,3)=A(:,3)/(Q*R);
[minErr,I]=min(A(:,3))
best_nb=A(I,1)
best_epoch=A(I,2)

%--------------Fin of segment_NB_NRURONS.m------------------------------%


%segment_RESULT_10manip_EVAL.m
%Plot all of result of simulation, compare the PSD-AI et PSD-NN


for ii=1:size(no_manip)
   begin_f=1+end_f;
   end_f=begin_f-1+nb_instant(ii);
   n=1;
   for jj=begin_f:end_f
      figure(ff+ii)
      subplot(2,4,n)
      plot(corde,target(:,jj),corde,output(:,jj),'--')
      title(['Reusults Manip ',num2str(ii),'Instant ',num2str(n)])
      xlabel('Corde Length (micrometre)')
      ylabel('#%,Nb')
      legend('PSD from CCD','PSD calculated by NN')
      grid

      n=n+1;

   end

end
%-----------------------Fin of segment_RESULT_10manip_EVAL.m-------------%


%segment_DIS_SURFACE_10manip_EVAL.m
%Plot the cumulative distribution of CLD-FBRM and PSD-AI

end_f=0;

for ii=1:size(no_manip)
   begin_f=1+end_f;
   end_f=begin_f-1+nb_instant(ii);
   n=1;
   for jj=begin_f:end_f
      figure(20+ii)
      subplot(2,4,n)
      plot(corde,percent_SURFACE_FBRM(:,jj))
      title(['Cumulative distribution of CLD  Manip ',num2str(ii),'Instant ',num2str(n)])
      xlabel('Corde Length (micrometre)')
      ylabel('%')
      grid

      n=n+1;

   end

end
```

```
for ii=1:size(no_manip)
   begin_f=1+end_f;
   end_f=begin_f-1+nb_instant(ii);
   n=1;
   for jj=begin_f:end_f
      figure(30+ii)
      subplot(2,4,n)
      plot(corde,percent_SURFACE_AI(:,jj))
      title(['Cumulative distribution of CLD  Manip ',num2str(ii),'Instant ',num2str(n)])
      xlabel('Corde Length (micrometre)')
      ylabel('%')
      grid

      n=n+1;

   end

end

%-----------------------Fin of segment_DIS_SURFACE_10manip_EVAL.m-------------%
```