[Presented during the CGOM11-ACTS 2014 Joint Meeting]

Particle Size Distribution reconstruction using a finite number of its moments though Artificial Neural Networks: A practical application.

G. Cogoni*, P. J. Frawley

Solid State Pharmaceutical Cluster (SSPC), Materials & Surface Science Institute (MSSI), Lonsdale Building, Department of Mechanical, Aeronautical and Biomedical Engineering, University of Limerick, Castletroy, Co. Limerick, Ireland.

***Corresponding author:** *Address:* L1-025, Lonsdale Building, Department of Mechanical, Aeronautical and Biomedical Engineering, University of Limerick, Castletroy, Co. Limerick, Ireland. *Tel:* +353 (0)61 213134. *Fax:* +353 (0)61 202944. *E-mail:* giuseppe.cogoni@ul.ie (G. Cogoni). *URL:* http://www.ul.ie/sspc (G. Cogoni).

ABSTRACT:

An Artificial Neural Network (ANN) approach to reconstruct the Particle Size Distribution (PSD) is proposed in this paper. This novel technique has been applied for acetaminophen crystallization in ethanol. Several experimental PSDs taken in different operating conditions, such as temperature and agitation degree, at different stages of the process have been considered, in order to ensure a wide range of different distributions for the system taken into account. The first stage of the ANN modeling is represented by the structure definition and the network training through a backpropagation algorithm in which the experimental PSDs and their associated vector of moments have been used. The second stage is represented by the feedforward application and validations of the proposed model estimating the PSDs using

a finite set of experimental moments compared with their associated PSDs and then, using a set of time dependent moments obtain the transitory PSD. The proposed approach represents a more suitable way to reconstruct the PSD in full, for the first time without assuming any reference distribution or knowing in advance the shape of the experimental PSD, leading to a generalized characterization of the PSDs with possible implementations in other multiphase unit operations.

Keywords: *Artificial neural networks, PSD reconstruction, Method of moments, Population balance modeling, Crystallization processes.*

1. INTRODUCTION

Particulate systems are usually modeled through Population Balance Equations (PBE), this approach can take into account different physical and thermodynamic aspects such as growth rate, nucleation, breakage, agglomeration, etc, leading to obtain the time evolution of a number density distribution and therefore a Particle Size Distribution (PSD). Describing the time evolution of the PSD from Population Balance Equations (PBE) is often computationally expensive. In order to minimize the computational requirements of these process models, a number of computations or outputs, and/or reduce the amount of memory required to perform these simulations. These techniques include the Method of Moments (MOM), or Standard Method of Moments (SMOM)[1,2], first proposed by Randolph & Larson[3], the Quadrature Method of Moments (QMOM)[1,4,5] and the Direct Quadrature Method of Moments (DQMOM)[6] which can significantly reduce the computational costs of numerical simulations of multiphase process. These techniques involve the transformation of the PBE from an integro-partial differential equation into a system or ODEs obtaining as a result the time evolution of the moments of the distribution in place of the actual PSD. This simplification eliminates the need to track each particle class which is very computationally expensive and where high accuracy would be required, as high accuracy translates to a high number of particle size classes. However, the main downfall of these methods is that the PSD remains unknown during the course of the simulation. The PSD generally constitutes a key process result which may be used to gauge the performance of the process. The size and size distribution of particles in particulate process, such as milling or crystallization are particularly important for several reasons. These properties heavily influence the efficiency of downstream processes, such as filtration and centrifugation, where a small product size can significantly increase processing time.

The PSD needs therefore to be reconstructed from its moments obtained. From a mathematical point of view, a necessary but not sufficient condition is that all the moments up to infinity are required in order to obtain a proper reconstruction. The reconstruction of the PSD from its corresponding moments has been studied since the first decade of the past century[7], known as the "finite moment problem". Since then, several reconstruction techniques have been developed. Nowadays two main reconstruction approaches are adopted and they can be grouped in two categories: methods that require prior knowledge of the PSD's shape and methods that use *a priori* distribution model in order to reconstruct the PSD.

Considering the first category spline-based reconstruction techniques have been proposed by some authors[8,9]. This technique implies to use spline polynomials to evaluate the shape of the PSD in discrete intervals of the distribution. This technique showed that it is highly sensitive to the order of the spline polynomials and the number of moments considered. However, some improvements were proposed, considering an adaptive grid for the nodes (or bins) of the distribution, showing a better accuracy, particularly for non-smooth distributions[9].

The second category requires assuming *a priori* distribution to reconstruct the PSD. The most recent method developed is called Moment Surface Method, proposed by Hutton et al[10]. This technique considers two reference distributions, respectively a Weibull and a log-normal distribution, where both parameters are estimated by intersecting the iso-i-th-moment lines obtained from the population balance simulation into a 3D plot which reports the two parameters of the pre-assumed functional form and its i-th moment. This method can be also extended for reference PSDs with more than two parameters, resulting into a hypersurface for each i-th moment where the corresponding parameters can be evaluated again by intersection with the iso-i-th-moment obtained from simulated data. The counterpart of this technique is

that it is not applicable to describe multimodal distributions, reducing the modeling accuracy as secondary nucleation is taken into account[11]. An alternative technique can be proposed by applying Artificial Neural Networks (ANN). In engineering fields, one of the most important applications of artificial neural networks is modeling a non-linear system with an unknown input-output relation. ANNs implement algorithms that attempt to achieve a neurological related performance, such as learning from experience, making generalizations from similar situations and judging states where poor results were achieved in the past. This powerful tool would represent the ideal technique to reconstruct a PSD from a finite number of its moments, without needing to know in advance the PSD shape or to assume a priori Probabilistic Density Function (PDF). This technique will be outlined in this paper using experimental PSDs obtained from the acetaminophen crystallization in different operating conditions in order to train the ANN. After the ANN has been trained, the results obtained have been first compared with experimental PSDs, obtained considering the same system, used during the network training and then the same ANN has been used to reconstruct the time evolution of a PSD obtained from the analytical solution of a simple PBE considering a simple growth process, with a normal distribution as initial condition.

2. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANN) have seen an explosion of interest over the last decade and they have been successfully applied across an extraordinary range of problems in areas as diverse as finance, medicine, engineering, geology, physics and biology. ANNs are non-linear data driven self adaptive approaches as opposed to the traditional model based methods. They are powerful tools for modeling, especially when the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data

sets and corresponding target values. After training, ANNs can be used to predict the outcome of new independent input data. ANNs imitate the learning process of the human brain and can process problems involving nonlinear and complex data even if the data are imprecise and noisy.

A neural network consists of a set of connected cells called neurons or perceptrons. The neurons receive impulses from either input cells or other neurons and perform a transformation of the input and transmit the outcome to other neurons or to output cells. The neural networks are built from layers of neurons connected so that one layer receives input from the preceding layer of neurons and passes the output on to the subsequent layer.

A neuron is a real function of the input vector $\boldsymbol{x} = [x_1,\ldots,x_n]$. The j-th output is obtained as:

$$y_j = \varphi_j\left[\sum_{i=1}^{n}\left(\omega_{ij}x_i - \theta_j\right)\right] \tag{1}$$

Where φ_j is a function, typically the sigmoid (logistic or hyperbolic tangent) function, ω_{ij} is the weight referred to the i-th input and θ_j is the bias. A graphical representation of neuron is given in Figure 1.

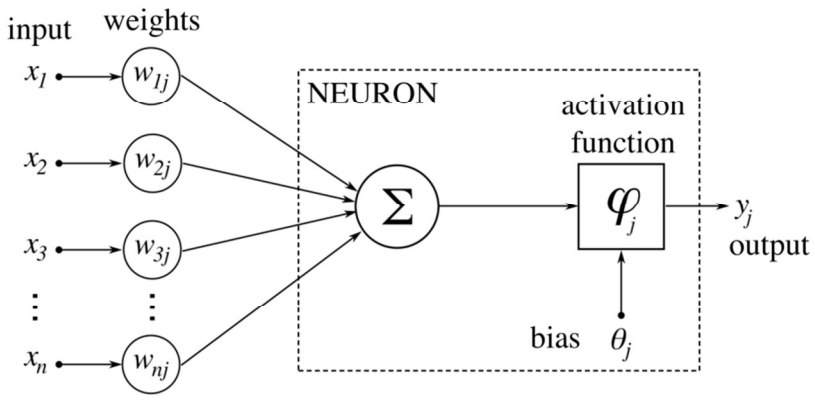

Figure 1: Schematic representation of a mathematical neuron, consisting in multiple inputs and one output.

The composition of the different layers consisting in the ANN, such as input layer, hidden layers and output layer is called Multi-Layer Perceptron network (MLP). MLP is a function consisting of compositions of weighted sums of the functions corresponding to the neurons (perceptrons).

In this work a feedforward architecture will be applied. In a feedforward network, information flow in one direction along connecting pathways, from the input layer via the hidden layers to the output layer.

Learning methods in neural networks can be broadly classified into three basic types: supervised, unsupervised and reinforced. Because the PSD reconstruction in this work will be based on its defined moments, this ANN learning method is classified as supervised.

In this, every input pattern that is used to train the network is associated with an output pattern, which is the target or the desired pattern. The error is used to change network parameters (weights), which result in an improvement in performances.

Given enough data, enough hidden units, and enough training time, an MLP with just one hidden layer can learn to approximate virtually any function to a high degree of accuracy (A statistical analogy is approximating a function with n-th order polynomials). For this reason MLPs are known as universal approximators and can be used when there is little prior knowledge of the relationship between inputs and targets.

3. PSD RECONSTRUCTION

3.1 ARTIFICIAL NEURAL NETWORK IMPLEMENTATION

The first stage of an Artificial Neural Network implementation is defining a proper topology/architecture. The problem considered in this work involves a transformation of a set of finite number of moments (inputs) into a vector that represents the ordinate axis of a Particle Size Distribution (PSD). Therefore, the output vector will have the same size of the number of bins of experimental PSDs considered during the ANN training process. The structure of the ANN applied for this work consist in a input layer with a number of nodes equal to the number of moments considered (first six moments of the PSD, starting from the first), a hidden layer consisting in a number of perceptrons equal to the number of inputs and, finally, an output layer of one hundred perceptrons in order to recognize each bin of the PSD. The activation functions used on the hidden layer neurons (perceptrons) consist on log-sigmoid functions, allowing a certain nonlinearity of the input data to be treated and constraining the processed hidden layer output within the range $[0,+\infty)$. The output layer instead consist in one hundred perceptrons with pure linear activation function each in order to process the signal coming from the output of the hidden layer into the final value of each bin of a PSD.

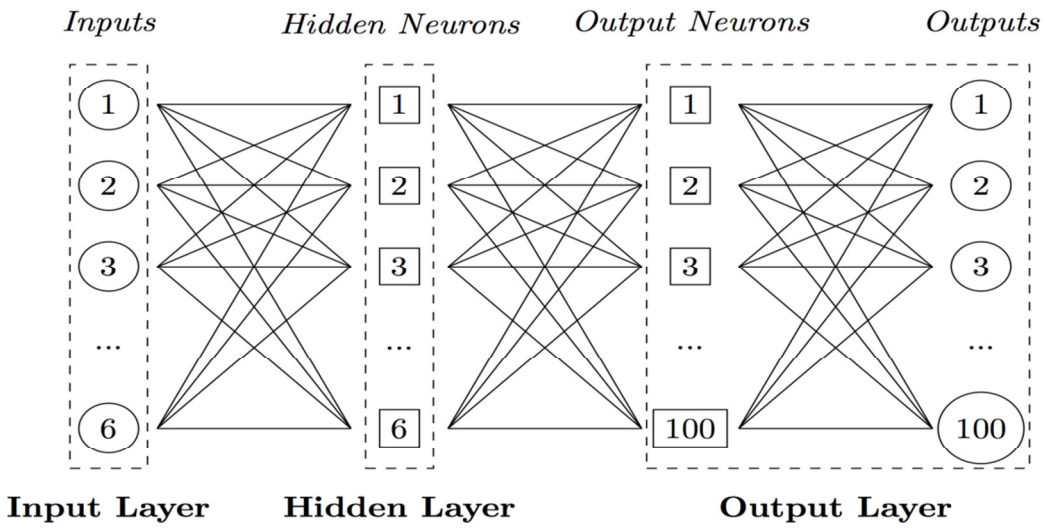

Figure 2: ANN structure showing the interconnections between layers and nodes (neurons).

After the ANN has been defined on its topology, the network needs to be trained. The selection of training data and number of the training examples play an important role in the performance of the supervised ANN. If the number of training examples is not sufficient, then the network cannot correctly learn the actual input-output relation of the system. For this work 50 PSDs with 100 channels/bins each taken from a Malvern Mastersizer 2000 for the crystallizing system Acetaminophen-Ethanol has been considered. All the PSDs have been randomly chosen in order to minimize any possible correlation between each other and therefore maximizing the learning performance of the ANN.

All the experimental PSDs (volume based) have been normalized before being used for the training of the ANN and the first six moments of each have been evaluated applying the equations (2a-2b).

$$\Psi_i(L) = \frac{\Psi_i^*(L)}{\int_0^{+\infty} \Psi_i^*(L) dL} \tag{2a}$$

$$m_{k,i} = \int_0^{+\infty} \Psi_i(L) L^k dL \quad with \quad k = 1,2,...,6; \quad i = 1,2,...,50 \tag{2b}$$

The zero-th moment has been not considered since it would not contribute any information for the training process, being equal to one for every experimental PSD. The training data set consist in a set of 50 experimental PSDs with its associated first six moments.

The training process has been conducted using the Neural Network Toolbox on MATLAB. The training data set has been then split in the following order, 80% of the data have been used for pure training of the ANN, 10% for the validation and the remaining 10% for testing the training performance of the ANN. The total number of iterations, without improvements, for the training algorithm, properly called epochs, has been fixed to 1500, as well as the maximum and the minimum value for the mean square error (mse).

Without *a priori* information about the final weights of the ANN, it is common practice to initialize all weights randomly with a small absolute value.

The weights then have been estimated applying a Levenberg-Marquardt algorithm[12-14] that will be discussed in detail on the next paragraph.

3.2 BACKPROPAGATION LEARNING ALGORITHM BASED ON LEVENBERG-MARQUARDT ALGORITHM AND TRAINING PERFORMANCE

Levenberg-Marquardt (LM) algorithm is the most widely used optimization algorithm. LM algorithm is an iterative technique that locates a local minimum of a multivariate function

that is expressed as the sum of squares of several nonlinear real-valued functions (SSE). It has become a standard technique for nonlinear least-squares problems, widely adopted in various disciplines for dealing data-fitting applications. Levenberg-Marquardt curve-fitting method is actually a combination of two minimization methods, the gradient descend method or steepest descend algorithm and the Gaussian-Newton method[15].

The steepest descent algorithm is a first-order algorithm. It uses the first-order derivative of total error function to find the minimum/a in error space. Normally, gradient **g** is defined as the first-order derivative of total function error E(**x**, $\boldsymbol{\omega}$):

$$E(x,\omega)=\frac{1}{2}\sum_{p=1}^{NP}\sum_{m=1}^{NM}e_{p,m}^{2} \tag{3}$$

Where x is the input vector, $\boldsymbol{\omega}$ is the weight vector and $e_{p,m}$ is the training error at output m when applying the training pattern p and it is defined as $e_{p,m} = d_{p,m} - o_{p,m}$, with d defined as the desired output vector and o is the actual output vector.

The gradient vector is thus defined as:

$$g=\frac{\partial E(x,\omega)}{\partial \omega}=\left[\frac{\partial E}{\partial \omega_1},\frac{\partial E}{\partial \omega_2},\ldots,\frac{\partial E}{\partial \omega_N}\right]^T \tag{4}$$

During the training process, the update rule for the steepest descent algorithm could be written as:

$$\omega_{k+1}=\omega_k-\alpha g_k \tag{5}$$

Where the index k represents the iteration index and α is the learning constant or step size.

This training process leads to a result that is asymptotically convergent, nevertheless, it is well known to be an inefficient algorithm because its slow convergence. The two main reasons for its slow convergence can be summarized first, its step size should be related to the gradients, therefore small step sizes should be taken where the gradient is steep, but once the gradient becomes gentle, the training process would be very slow. The second reason is that the curvature of the error surface may not be the same in all directions and may result in slow convergence in those surface regions where the gradient is very small.

The slow convergence of the steepest descent method can be greatly improved by the Gauss-Newton algorithm[15]. Using second-order derivatives of error function to evaluate the curvature of the error surface, it is possible to find proper step sizes for each direction of the error surface, resulting in a faster convergence.

The Gauss-Newton algorithm approximates the Hessian matrix as the product between the transpose Jacobian and the Jacobian matrix. This simplifies the computation of the weight update, since the second-order derivatives of total error function have to be calculated and could be very complicated.

$$\omega_{k+1} = \omega_k - H_k^{-1} g_k \tag{6}$$

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial \omega_1^2} & \frac{\partial^2 E}{\partial \omega_1 \partial \omega_2} & \cdots & \frac{\partial^2 E}{\partial \omega_1 \partial \omega_N} \\ \frac{\partial^2 E}{\partial \omega_2 \partial \omega_1} & \frac{\partial^2 E}{\partial \omega_2^2} & \cdots & \frac{\partial^2 E}{\partial \omega_2 \partial \omega_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial \omega_N \partial \omega_2} & \frac{\partial^2 E}{\partial \omega_N \partial \omega_2} & \cdots & \frac{\partial^2 E}{\partial \omega_N^2} \end{bmatrix} \tag{7}$$

$$H \approx J^T J \tag{8}$$

$$J^T = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial \omega_1} & \frac{\partial e_{1,2}}{\partial \omega_1} & \cdots & \frac{\partial e_{1,NM}}{\partial \omega_1} & \cdots & \frac{\partial e_{NP,1}}{\partial \omega_1} & \frac{\partial e_{NP,2}}{\partial \omega_1} & \cdots & \frac{\partial e_{NP,NM}}{\partial \omega_1} \\ \frac{\partial e_{1,1}}{\partial \omega_2} & \frac{\partial e_{1,2}}{\partial \omega_2} & \cdots & \frac{\partial e_{1,NM}}{\partial \omega_2} & \cdots & \frac{\partial e_{NP,1}}{\partial \omega_2} & \frac{\partial e_{NP,2}}{\partial \omega_2} & \cdots & \frac{\partial e_{NP,NM}}{\partial \omega_2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,1}}{\partial \omega_N} & \frac{\partial e_{1,2}}{\partial \omega_N} & \cdots & \frac{\partial e_{1,NM}}{\partial \omega_N} & \cdots & \frac{\partial e_{NP,1}}{\partial \omega_N} & \frac{\partial e_{NP,2}}{\partial \omega_N} & \cdots & \frac{\partial e_{NP,NM}}{\partial \omega_N} \end{bmatrix} \tag{9}$$

$$\omega_{k+1} = \omega_k - \left(J_k^T J_k\right)^{-1} J_k e_k \tag{10}$$

In order to make sure that the approximated Hessian matrix J^TJ is invertible, Levenberg-Marquardt algorithm introduces another approximation to Hessian matrix:

$$H \approx J^T J + \mu I \tag{11}$$

Where μ is the combination coefficient, which is always positive and I is the identity matrix.

The combination coefficient along with the identity matrix allows the diagonal of the approximated Hessian matrix to be always larger than zero and therefore allowing the Hessian matrix to be always invertible.

Combining the Equations (9) and (10), the updating rule for the Levenberg-Marquardt algorithm can be written as:

$$\omega_{k+1} = \omega_k - \left(J^T J + \mu I\right)^{-1} J_k e_k \tag{12}$$

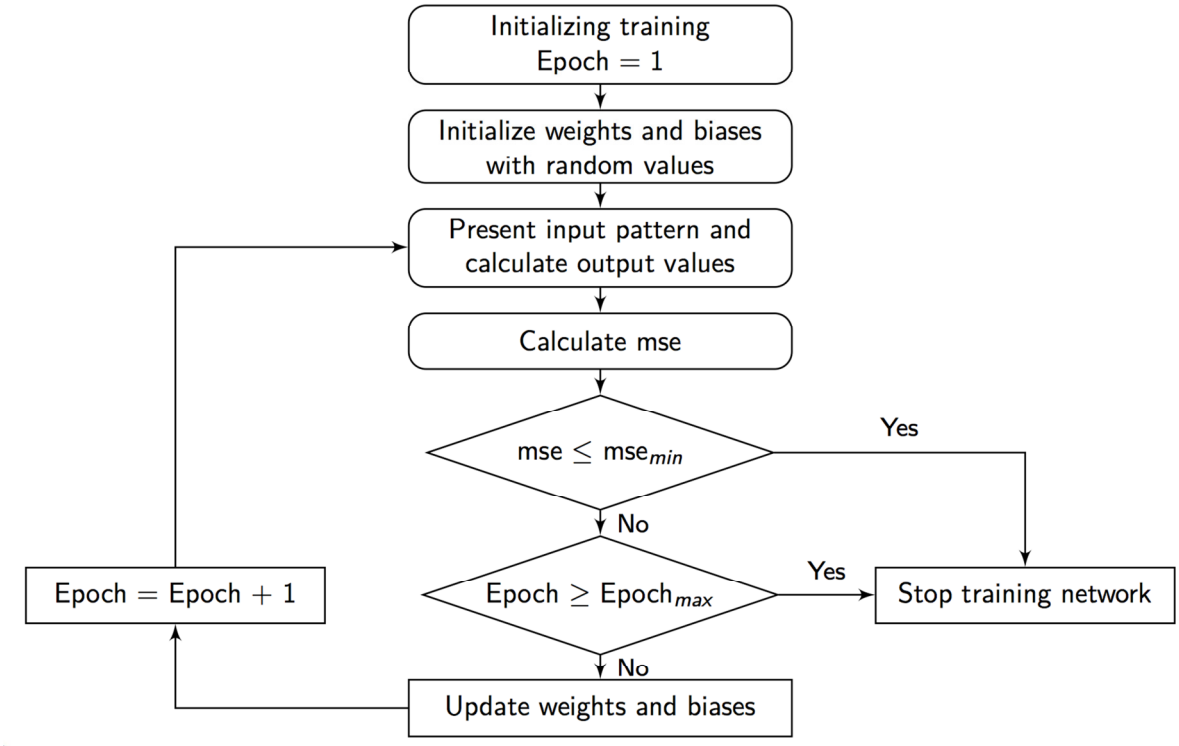

Figure 3: Backpropagation algorithm flow chart.

The training criteria includes fixing the maximum and minimum mean square error (mse) and the number of training iterations (epochs) without improvements. Once these given training

criteria are met, the training algorithm is stopped. If the training criteria are not met, one can either restart the training process, in order to randomly reinitialize the weights and obtaining another local minimum value of the mse, or change the activation functions or change the overall architecture of the ANN.

The training performances of the ANN obtained from the MATLAB Neural Network Toolbox have been summarized on Figure 4, showing a total number of epochs of 52 in order to achieve the minimum mse for the set of data provided.

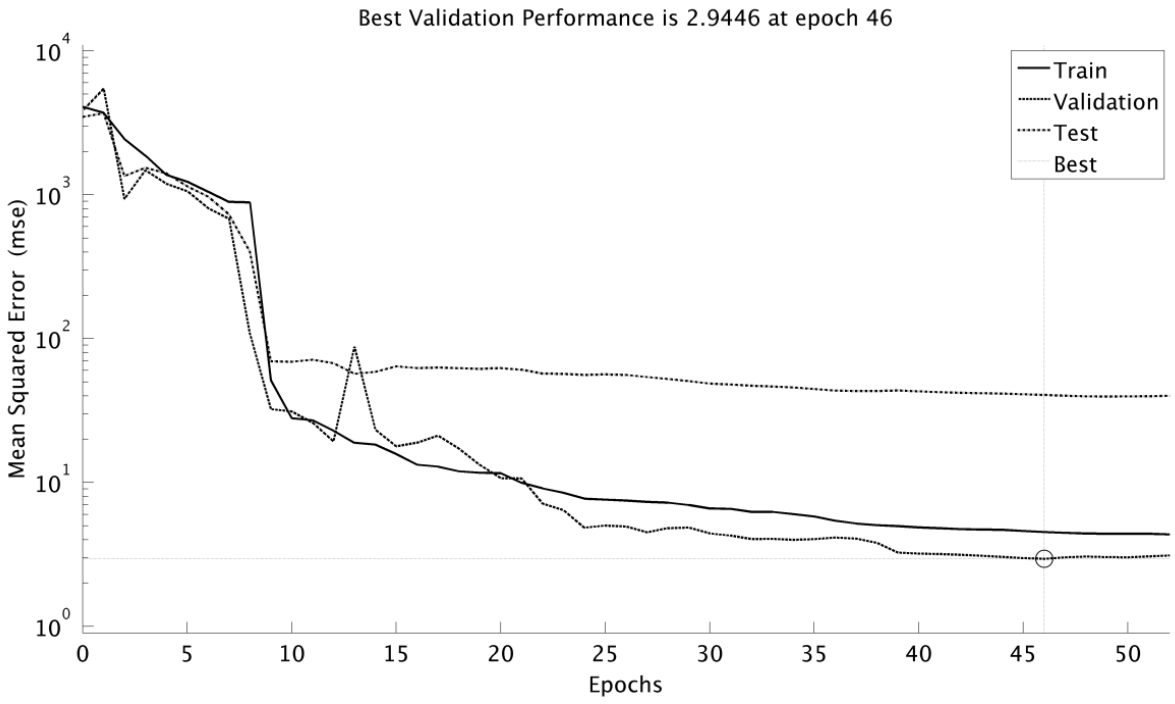

Figure 4: Performance plot of backpropagation algorithm representing the mse vs epochs. The solid black line represents the ANN training (80% of the total data), the black fine dotted line represents the ANN validation (10% of the total data) and the black dotted line represents the ANN test (10% of the total data).

From Figure 4 it is possible to state that all the curves converge almost monotonically to a finite small mean square error, suggesting that the configuration used is correct. The overall training process in an Intel Core 2 Duo processor of 2.4 GHz, with 4 GB of memory, running Linux took about 69 seconds. This implies that the training algorithm can be quickly implemented and applied even in a relatively old machine, allowing its application even for adaptive ANN[16], where the weight of the network are updated as it is applied online to estimate the PSDs.

4. RESULTS AND DISCUSSION

Once an Artificial Neural Network has been structured and trained, it is ready to be applied in feedforward conditions. Its feedforward application works as a normal mathematical function with the exception that in this case the function structure is not defined and cannot be either derived or integrated.

$$\Psi(L,t) = net[m_1(t), m_2(t), ..., m_6(t)] \tag{13}$$

All the points generated by the ANN must be plotted on the same abscissa axis from the data used during the training process. For this specific application the same bins generated by the particulate size analyzer Mastersizer 2000.

4.1 FEEDFORWARD PREDICTION OF THE PSD

In order to apply and verify the goodness of prediction of the trained ANN in feedforward conditions, four random experimental PSDs taken in similar operating conditions of the data used for the ANN training have been considered. The first six moments, excluding the zero-

th have been calculated and fed to the ANN. The reconstructed and the experimental PSDs have been then compared.

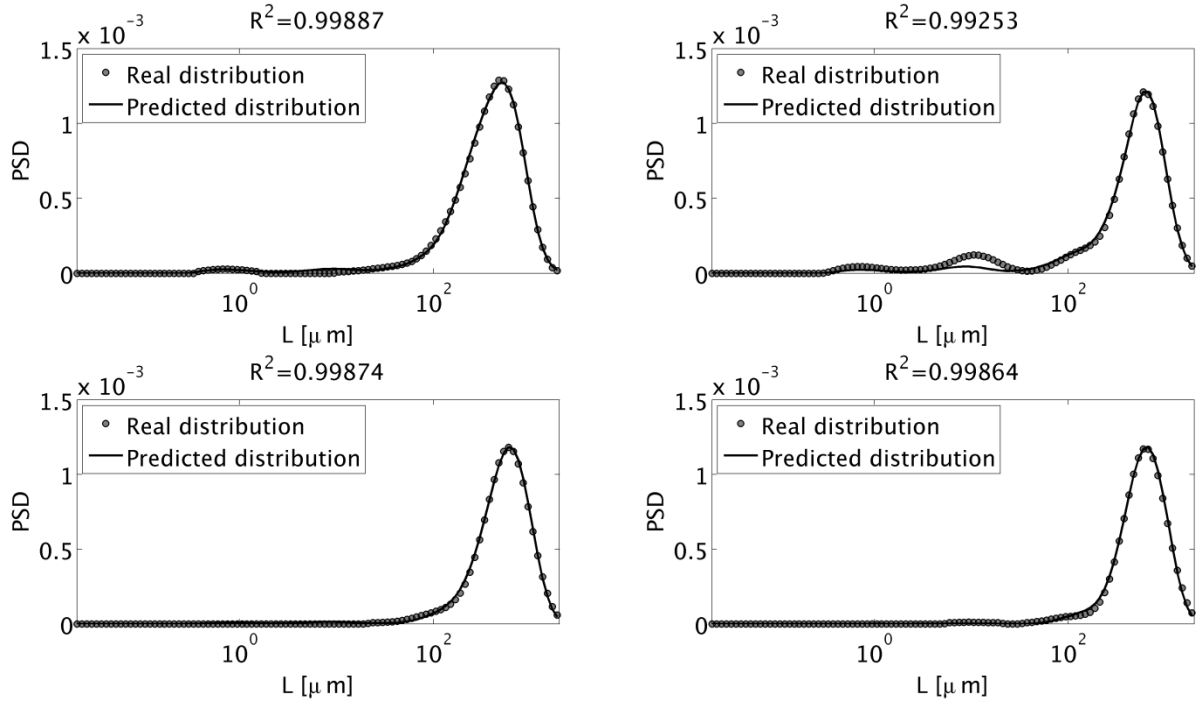

Figure 5: Experimental validation of the ANN reconstruction method. All four PSDs are taken in different operating conditions and in different stages of the process condition and plotted on a logarithmic abscissa. The solid black line represents the reconstructed PSDs using the ANN, while the gray dots represent the experimental PSDs.

From Figure 5 it is possible to observe the all four reconstructed PSDs present a good fit with the experimental data, with a high coefficient of determination R^2. The reconstructed PSDs showed also a good flexibility on the modality characteristic of the particle size distribution, allowing the reconstructed PSD to assume any possible, from an almost mono-modal distribution [bottom right] moving to almost bimodal distribution but with a minor peak in

within the range of nanoparticles to a range of fine particles [top left and bottom right], being able to describe even multimodal distributions [top right]. All these results come from the same ANN trained using experimental PSDs and applied in feedforward using experimental PSD's moments from the same system.

4.2 TRANSITORY APPLICATION USING A BENCHMARK PBE

In order to test the ANN in transitory conditions, a benchmark population balance amenable to analytical solution has been considered. The model (14) has been defined assuming an only growth process; where the growth rate does not depend either on the particle size *L* or the solution concentration[17]. The initial condition has been assumed as a Gaussian or normal distribution with a given initial mean size of crystals and variance. The boundary conditions are used in order to constrain the first derivative of the dependant variable n equal to zero for particles of null or infinite size.

$$\begin{cases} \dfrac{\partial n(L,t)}{\partial t} + G\dfrac{\partial n(L,t)}{\partial L} = 0 & L \in \Re_0^+ \qquad t > 0 \\ n(L,0) = \dfrac{1}{\sigma_{L,0}\sqrt{2\pi}} \exp\left[-\dfrac{(L-\mu_{L,0})^2}{2\sigma_{L,0}^2} \right] & L \in \Re_0^+ \qquad t = 0 \\ \dfrac{\partial n(0,t)}{\partial L} = \dfrac{\partial n(+\infty,t)}{\partial L} = 0 & \forall t \end{cases} \tag{14}$$

The analytical solution of the Equation (14) is trivial; any initial condition is shifted to the right. Therefore, the solution can be seen as the initial normal distribution where the initial mean size is shifted over time with velocity *G*. The analytical solution is represented as follows:

$$n(L,t) = \Psi(L,t) = \frac{1}{\sigma_{L,0}\sqrt{2\pi}} \exp\left\{ - \frac{\left[\left(L - \mu_{L,0} \right) - Gt \right]^2}{2\sigma_{L,0}^2} \right\} \qquad L \in \Re_0^+ \qquad t > 0 \qquad (15)$$

The non-central moments of the solution of the Population Balance Equation (PBE) (14) can be calculated by applying the (2b) to (15), obtaining the following set of time dependent first six moments:

$$\begin{cases} m_1(t) = \mu_{L,0} + Gt \\ m_2(t) = m_1^2(t) + \sigma_{L,0}^2 \\ m_3(t) = m_1^3(t) + 3m_1(t)\sigma_{L,0}^2 \\ m_4(t) = m_1^4(t) + 6m_1^2(t)\sigma_{L,0}^2 + 3\sigma_{L,0}^4 \\ m_5(t) = m_1^5(t) + 10m_1^3(t)\sigma_{L,0}^2 + 15m_1(t)\sigma_{L,0}^4 \\ m_6(t) = m_1^6(t) + 15m_1^4(t)\sigma_{L,0}^2 + 45m_1^2(t)\sigma_{L,0}^4 + 15\sigma_{L,0}^6 \end{cases} \qquad (16)$$

Considering the relations between a central and non-central moments and the definition of the zero-th moment, which is constant over time for this particular model: $m_0(0) = m_0(t) = 1$; $m_1(0) = \mu_{L,0}$ and $m_2(0) = \mu^2_{L,0} + \sigma^2_{L,0}$, the system (15) can be generalized with the following formula:

$$m_k(t) = m_{k,0} + \sum_{j=0}^{k} \binom{k}{j} m_{k-j,0} (Gt)^j \qquad with \quad k = 1,2,...,n \qquad (17)$$

Considering the initial mean $\mu_{L,0}$ and standard deviation $\sigma_{L,0}$ of the initial PSD equal respectively to 350 µm and 120 µm and, a constant, arbitrary chosen, growth rate (G) equal

to $1x10^{-2}$ μm/s, the set of time dependent moments (16) can be fed to the already trained ANN in order to feedforward predict the time evolution of the PSD.

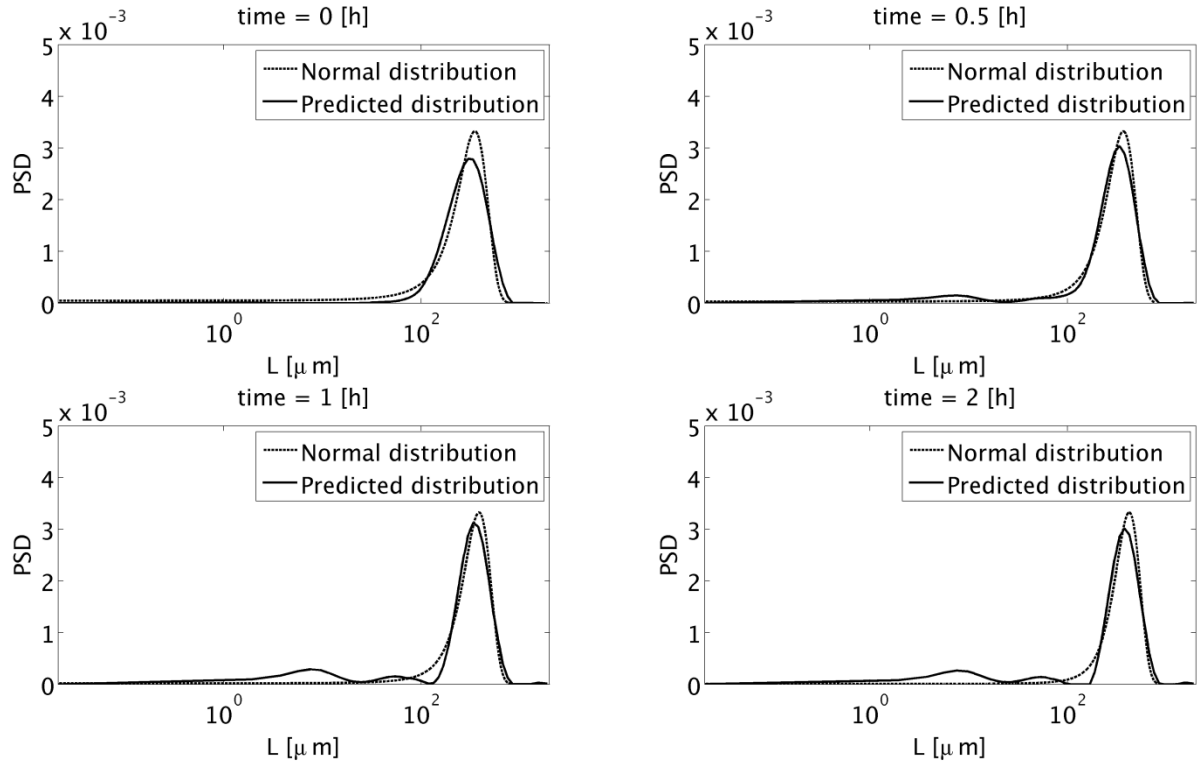

Figure 6: Time evolution of the PSD evaluated at different integration times, respectively [0 hr, 1 hr, 1.5 hrs and 2 hrs]. The solid and the dashed black line represent the reconstructed PSD using the ANN and the analytical solution of the benchmark Population Balance Equation (PBE).

The results showed on Figure 6, even though the analytical solution of the benchmark population balance (14) and the reconstructed PSD using the ANN present discrepancies, the main peak of the distribution can be still captured. The reason why the ANN reconstruction is not able to follow accurately the time evolution of (15), is because the artificial neural

network has been trained using experimental data from an acetaminophen crystallization process, therefore the first six moments of the transitory Gaussian distribution from equation (15) can give a wrong information to the ANN, being interpreted as they were from a real crystallization process. This effect could happen even if the same ANN is used to reconstruct a PSD using moments obtained from a different crystallizing system than acetaminophen, or from the same crystallizing system but with completely different operating conditions from those used on the experimental data during the training process.

It is a good practice, in order to minimize these discrepancies, to either use another ANN trained with experimental data from a different crystallizing system or update the ANN with new experimental data, therefore redo the training process, or modify the ANN as an adaptive artificial neural network, where the weight will be estimated on the go, as the operating conditions of the system considered change or directly when the crystallizing solute is different (even with the same solute but during a polymorphic transition).

5. CONCLUSIONS

In this work, an application of the Artificial Neural Network to reconstruct a full PSD from a finite number of its moments has been outlined. The ANN has been trained using experimental PSDs from acetaminophen crystallization and evaluating their first six moments, excluding the zero-th in order to allow the ANN to find a pattern between PSD and the PSD moments. The ANN has been tested in feedforward condition, in order to estimate the PSD from a given set of moments and then compared with experimental PSD, obtained in the same operating conditions of the ones used for the training process, showing high reconstruction accuracy even for multimodal distributions. In order to test the transitory performance of the trained ANN, it has been applied in feedforward using a dynamic set of

moments obtained from a benchmark population balance amenable to analytical solution. The results have shown good fit with the analytical PSD although the reconstructed PSDs where showing multiple peaks, not present in the analytical PSD. This effect has been justified by the lack of information for the ANN about the possible patterns between the first six moments and Gaussian PSDs, suggesting several methods to overcome to this problem in the practical application. The PSD reconstruction from a finite number of moments, using artificial neural networks has been demonstrated to be a more accurate and robust alternative to the already existing mathematical approaches. Because the flexibility of this approach, it is possible now to reconstruct multimodal PSDs or PSDs changing their modality characteristic. This approach can be applied to quantify and control the amount of fine particles during a crystallization process or in any particulate system.

Acknowledgements

This research has been conducted as part of the Solid State Pharmaceutical Cluster (SSPC) and founded by Science Foundation Ireland (SFI).

NOMENCLATURE

Acronyms

ANN	Artificial Neural Network;
DQMOM	Direct Quadrature Method Of Moments;
LM	Levenberg-Marquardt;
mse	mean square errors;
MLP	Multi-Layer Perceptron;
MOM	Method Of Moments;
ODE	Ordinary Differential Equation;
PSD	Particle Size Distribution;
PBE	Population Balance Equation;
QMOM	Quadrature Method Of Moments;
SSE	Sum of Square Errors;

Variables with units and symbols

$d_{p,m}$	Desired m-th output for the p-th pattern;
$e_{p,m}$	Training error at m-th output for the p-th pattern;
$E(x,\omega)$	Total function error;
g	Gradient vector;
G	Growth rate [];
H	Hessian matrix;
I	Identity matrix;
J	Jacobian matrix;
L	Characteristic size of crystals [m];
$m_{k,i}$	K-th moment evaluated for the i-th PSD;

$n(L,t)$ Number density distribution, dependent variable of the PBE [];

N Total number of weight defining the ANN;

NM Total number of output variables;

NP Total number of pattern inputs;

$o_{p,m}$ Actual m-th output vector for the p-th pattern;

$\Re_0^+$ Set of positive real numbers including zero;

R^2 Coefficient of determination;

t Time variable [s];

x_i I-th input variable;

y_j Output variable of the artificial neural network from the j-th perceptron;

Greek letters

α Learning constant or step size;

θ_j Bias for the j-th perceptron;

μ Combination coefficient for the Levenberg-Marquardt algorithm;

$\mu_{L,0}$ Initial mean size of crystals of the Gaussian PSD [];

$\sigma_{L,0}^2$ Initial variance of the Gaussian PSD [];

φ_j Activation function for the j-th perceptron;

$\Psi_i(L)$ Normalized i-th PSD used for the ANN training process;

$\Psi_i^*(L)$ I-th PSD used for the ANN training process, before normalization;

ω_{ij} Weight for the i-th input variable on the j-th perceptron;

REFERENCES

(1) Nagy, Z. K.; Aamir, E.; Rielly, C. D. Internal fines removal using population balance model based control of crystal size distribution under dissolution, growth and nucleation mechanism. *Crystal Growth & Design*, **2011**, 11, 2205-2219.

(2) Ó' Ciardhá, C. T.; Hutton, K. W.; Mitchell, N. A.; Frawley, P. J. Simultaneous parameter estimation and optimization of a seeded antisolvent crystallization. *Crystal Growth & Design*, **2012**, 12, 5247-5261.

(3) Randolph, A. D.; Larson, M. A. *Theory of particulate processes*; Academic Press: San Diego (CA), **1988**.

(4) Marchisio, D. L.; Vigil, R. D.; Fox, R. O. Quadrature method of moments for aggregation-breakage process. *J. Colloid Interface Sci.*, **2003**, 258, 322-334.

(5) McGraw, R.; Nemesure, S.; Schwartz, S. E. Properties and evolution of aerosols with size distributions having identical moments. *J. Aerosol Sci.*, **1988**, 29 (7), 761-772.

(6) Marchisio, D. L.; Fox, R. O. Solution of population balance equations using the direct quadrature of moments. *J. Aerosol Sci.*, **2005**, 36, 43-73.

(7) Von Smoluschowski, M. Versuch einer mathematischen theorie der koagulationskinetik kolloider losungen. *Zeitschrift fur Physikalische Chemie*, **1917**, 92, 129-168.

(8) John, V.; Angelov, I.; Öncül, A. A.; Thévenin, D. Techniques for the reconstruction of a distribution from a finite number of its moments. *Chem. Eng. Sci.*, **2007**, 62, 2890-2904.

(9) De Souza, L. G. M.; Janiga, G.; John, V.; Thévenin, D. Reconstruction of a distribution from a finite number of moments with an adaptive spline-based algorithm. *Chem. Eng. Sci.*, **2010**, 65, 2741-2750.

(10) Hutton, K.; Mitchell, N.; Frawley, P. J. Particle size distribution reconstruction: The moment surface method. *Powder Technology*, **2012**, 222, 8-14.

(11) Frawley, P. J; Mitchell, N. A.; Ó'Ciardhá, C. T.; Hutton, K. W. The effects of supersaturation, temperature, agitation and seed surface area on the secondary nucleation of paracetamol in ethanol solutions. *Chem. Eng. Sci.*, **2012**, 75, 183-197.

(12) Levenberg, K. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, **1944**, 5, 164-168.

(13) Marquardt, D. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. on Appl. Math.*, **1944**, 11 (2), 431-441.

(14) Hagan, M. T.; Menhaj, M. B. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. On Neural Networks*, **1994**, 5 (6), 989-993.

(15) Osborne, M. R. Fisher's method of scoring. *Int. Stat. Review*, **1992**, 86, 271-286.

(16) Acharya, R.; Roy, B.; Sivaraman, M. R.; Dasgupta, A. Prediction of ionospheric total electron content using adaptive neural network with in-situ learning algorithm. *Advances in Space Research*, **2011**, 47 (1), 115-123.

(17) Qamar, S.; Warnecke, G. Numerical solution of population balance equations for nucleation, growth and aggregation processes. *Comp. and Chem. Eng.*, **2007**, 31, 1576-1589.