Revista de metalurgia

Review of algorithms for modeling metal distribution equilibria in liquid-liquid extraction processes(•)

L.J. Lozano*, F.J. Alguacil**, M. Alonso** y C. Godínez*

Abstract

This work focuses on general guidelines to be considered for application of least-squares routines and artificial neural networks (ANN) in the estimation of metal distribution equilibria in liquid-liquid extraction process. The goal of the procedure in the statistical method is to find the values of the equilibrium constants (K_i) for the reactions involved in the metal extraction which minimizes the differences between experimental distribution coefficient (D_{exp}) and theoretical distribution coefficients according to the mechanism proposed (D_{theor}). In the first part of the article, results obtained with the most frequently routine reported in the bibliography are compared with those obtained using the algorithms previously discussed. In the second part, the main features of a single back-propagation neural network for the same purpose are discussed, and the results obtained are compared with those obtained with the classical methods.

Keywords Liquid-liquid extraction. Statistical methods. Artificial neural nets.

Revisión de algoritmos para modelización de equilibrios de distribución de metales en procesos de extracción líquido-líquido

Resumen

El trabajo presenta las líneas generales a considerar para la estimación del equilibrio de distribución de metales en procesos de extracción líquido-líquido, según dos métodos: algoritmo clásico de mínimos cuadrados y redes neuronales artificiales. El objetivo del procedimiento, en el caso del método estadístico, es encontrar los valores de las constantes de equilibrio (K_i) para las reacciones involucradas en la extracción del metal, que minimizan las diferencias entre el coeficiente de distribución experimental y el coeficiente de distribución teórico, de acuerdo al mecanismo propuesto. En la primera parte del artículo se comparan los resultados obtenidos a partir de los algoritmos usados más habitualmente en la bibliografía, con los datos obtenidos mediante el algoritmo previamente descrito. En la segunda parte, se presentan las características fundamentales para aplicar una red neuronal sencilla con algoritmo *back-propagation*, y los resultados obtenidos se comparan con los de los métodos clásicos.

Palabras clave Extracción líquido-líquido. Métodos estadísticos. Redes neuronales artificiales.

1. INTRODUCTION

The use of minimization routines to obtain the best value of equilibrium distribution constants for liquid-liquid systems that fits experimental data, is achieved by least-squares optimization using several computer tools. The program LETAGROP[1] is the most frequently used, but more accurate results can be obtained using the last algorithms developed for that purpose.

The main disadvantage of this "classical" method to fit experimental values to a liquid-liquid extraction model is that we need an "a priori" knowledge of the reactions involved in the process. Since mass transfer phenomena in hydrometallurgical systems may be very complex

(•) Trabajo recibido el día 10 de noviembre de 2004 y aceptado en su forma final el día 11 de julio de 2005.

(*) Department of Chemical & Environmental Engineering. Polytechnic University of Cartagena. Campus Muralla del Mar. C/ Doctor Fleming S/N, 30202 – Cartagena. SPAIN

(**) National Centre for Metallurgical Research (CENIM-CSIC). Avda. Gregorio del Amo, 8, 28040 – Madrid. SPAIN

(dimerization, aggregation of organic species, hydration processes, uncertainty in activity coefficients..) the mechanism proposed is always a "simplified" vision of the real problem.

Artificial neural networks (ANN) are being applied in many fields of chemical engineering due to their ability to approximate virtually any function in a stable and efficient way[2-8]. They are trainable dynamical systems and, unlike statistical estimators, they approximate a function without a mathematical model of how outputs depend on inputs[9]. These "model-free" estimators are widely recognized as an adequate technique for modeling complex or under-defined systems that are difficult to model otherwise.

ANN consists of numerous, simple processing units or "neurons" that we can globally program for computation. Despite the wide range of applications, ANN are still designed through a trial and error approach[10]. This design involves the following steps:

a) Structural design of the network, i.e. determination of number of layers and number of neurons in each layer and neuron transfer function.
b) Selection of the training methods and corresponding parameters (i.e. synaptic weights and neuron's biases).

2. METHODS FOR LEAST-SQUARES OPTIMIZATION

The basis of the method lies on the minimization of the function $U = \frac{1}{2}\sum\left(D_{theor} - D_{exp}\right)^2$

where U is the error square sum, D_{theor} is the theoretical distribution ratio predicted with the model and D_{exp} is the experimental distribution ratio.

The program "lsqcurvefit.m" included in the MATLAB Optimization Toolbox solve nonlinear curve-fitting (data-fitting) problems in the least-squares sense[11]. That is, given input data *xdata*, and the observed output *Dexp*, find coefficients K that "best-fit" the equation:

$$\min_K \sum\left(F(K, xdata) - D\exp\right)^2 \qquad (1)$$

where, *xdata* and *Dexp* are vectors of length equal to the number of experimental data points and F(K, *xdata*) is a vector valued function that computes *Dtheor* according to the extraction mechanism proposed.

The algorithm developed to solve the problem is based on a trust-region method for non linear minimization. Consider the unconstrained minimization problem, *min* $f(x)$, where the function takes vector arguments and returns scalars. Suppose you are at a point x in n-space and you want to improve, i.e., move to a point with a lower function value. The basic idea is to approximate f with a simpler function q which reasonably reflects the behavior of function f in a neighborhood around the point x. This neighborhood is the trust region. A trial step s is computed by minimizing (or approximately minimizing) over N. This is the trust-region subproblem,

$$\min_s \left\{q(s) \;\; s \in N\right\} \qquad (2)$$

The current point is updated to be $x+s$ if $f(x+s)>f(x)$; otherwise, the current point remains unchanged and N, the region of trust, is shrunk and the trial step computation is repeated.

The key questions in defining a specific trust-region approach to minimizing f(x) are:

a) How to choose and compute the approximation (defined at the current point x)
b) How to choose and modify the trust region N
c) How accurately to solve the trust-region subproblem.

In the standard trust-region method[12], the quadratic approximation q is defined by the first two terms of the Taylor approximation to f at x; the neighborhood N is usually spherical or ellipsoidal in shape. Mathematically the trust-region subproblem is typically stated:

$$min\left\{\frac{1}{2}s^T Hs + s^T g \quad such\ that \left\|Ds\right\| \le \Delta\right\} \qquad (3)$$

where, g is the gradient of f at the current point x, H is the Hessian matrix (the symmetric matrix of second derivatives), D is a diagonal scaling matrix, D is a positive scalar, and $||\ .\ ||$ is the 2-norm. Good algorithms exist for solving eq. 3 [12]; such algorithms typically involve the computation of a full eigensystem and a Newton process applied to the secular equation:

$$\frac{1}{\Delta} - \frac{1}{\left\|s\right\|} = 0 \qquad (4)$$

Such algorithms provide an accurate solution to equation 3. However, they require time proportional to several factorizations of *H*. Therefore, for large-scale problems a different approach is needed. Several approximation and heuristic strategies, based on eq. 3, have been proposed in the literature[13 y 14]. The approximation approach followed in the Matlab Optimization Toolbox is to restrict the trust-region subproblem to a two-dimensional subspace *S*[13, 15 y 16]. Once the subspace *S* has been computed, the work to solve eq. 3 is trivial even if full eigenvalue/eigenvector information is needed (since in the subspace, the problem is only two-dimensional). The dominant work has now shifted to the determination of the subspace.

The two-dimensional subspace *S* is determined with the aid of a preconditioned conjugate gradient process[11]. The toolbox assigns $S=<s_1,s_2>$, where s_1 is in the direction of the gradient *g*, and s_2 is either an approximate Newton direction, i.e., a solution to:

$$H \cdot s_2 = -g \tag{5}$$

or a direction of negative curvature,

$$s_2^T \cdot H \cdot s_2 < 0 \tag{6}$$

The philosophy behind this choice of *S* is to force global convergence (via the steepest descent direction or negative curvature direction) and achieve fast local convergence (via the Newton step, when it exists).

The framework for the Matlab Optimization Toolbox approach to unconstrained minimization using trust-region ideas is:

1) Formulate the two-dimensional trust-region subproblem.
2) Solve equation. 3 to determine the trial step *s*.
3) If $f(x+s) < f(x)$ then $x=x+s$.
4) Adjust Δ.

These four steps are repeated until convergence. The trust-region dimension D is adjusted according to standard rules. In particular, it is decreased if the trial step is not accepted, i.e., $f(x+s) \geq f(x)$ [17 y 18].

3. EQUILIBRIUM MODELS FOR LIQUID-LIQUID SYSTEMS

3.1. Cationic (acidic) extraction reagents

The general equation proposed is the next:

$$M^{n+} + a\,H_mX \Leftrightarrow \left[MX_a\right]^{n-a\cdot m} + a\cdot m\,H^+ \tag{7}$$

Where:

M = Metallic specie (eg. Fe^{3+}, VO_2^+,...)
n = Charge of metallic specie (positive)
m = Valence of the acid (positive)
X = Anion related to acid extraction reagent
a = Stoichometric coefficient for the extraction reagent

Considering that activity coefficients for all the species are approximately equal to 1 (ionic strength=0), the equilibrium constant can be written using concentrations according to:

$$K = \frac{\left[MX_a\right]\cdot\left[H^+\right]^{m\cdot a}}{\left[H_mX\right]^a\cdot\left[M^{+n}\right]} \tag{8}$$

The distribution coefficient, D, follows the next expression:

$$D = \frac{\left[M\right]_{org}}{\left[M\right]_{aq}} = \frac{\left[MX_a\right]}{\left[M^{n+}\right]} \tag{9}$$

where $[M]_{org}$ and $[M]_{aq}$ are the total concentration of metal in the aqueous and organic phases in equilibrium respectively. Combining both expressions:

$$D = \frac{K\left[H_mX\right]^a}{\left[H^+\right]^{m\cdot a}} \tag{10}$$

Equation 10 lets calculate the theoretical distribution coefficient (D_{theor}) if the next variables are known:

a) Equilibrium constant for the system (*K*)
b) Concentration of acid extraction reagent in equilibrium (approximately equals to the initial concentration).
c) Equilibrium pH.

If several simultaneous equilibria are proposed, the theoretical values of the distribution coefficient are calculated using:

$$D_{theor} = \sum_i D_i = \sum_i \frac{K_i\left[H_mX\right]^{a_i}}{\left[H^+\right]^{m\cdot a_i}} \tag{11}$$

where i denotes the number of equilibria proposed.

Experimental distribution coefficient (D_{exp}) can be calculated following two ways:

1) Analytical determination of the metal concentration in both phases:

$$D_{exp} = \frac{[M]_{org}}{[M]_{aq}}$$

2) Analytical determination of the metal concentration in the aqueous phase and calculation of the metal concentration in the organic phase in equilibrium using mass balance. The expression is:

$$[M]_{org} = \frac{V_{aq}\left([M]_0 - [M]_{aq}\right)}{V_{org}} = R\left([M]_0 - [M]_{aq}\right) \quad (12)$$

where, R denotes the ratio of aqueous to organic phase and $[M]_0$ the initial concentration of the metal. Finally:

$$D_{exp} = R\left(\frac{[M]_0}{[M]_{aq}} - 1\right) \quad (13)$$

The code for the program *caleq1.m* and the file *difcuad1.m* for the function that defines D_{theor} are showed in the appendix, and the data are stored in columns in an external Excel file (*nnur001.xls*).

3.2. Anionic extraction reagents

For the general case of a tertiary amine in the form of amine salt, the next expression is proposed:

$$\left(M_aX_c\right)^{na+cm} + \frac{-(na+cm)}{b}\left[\left(R_3NH^+\right)_b Y^{b-}\right] \Leftrightarrow \quad (14)$$
$$\Leftrightarrow \left[\left(R_3NH^+\right)_{-(na+cm)}\left(M_aX_c\right)^{na+cm}\right] + \frac{-(na+cm)}{b} Y^{b-}$$

Amine salt is formed according to the reaction:

$$b\,R_3N + H_bY \Rightarrow \left(R_3NH^+\right)_b Y^{b-} \quad (15)$$

where:

- M = Metal
- n = Oxidation state of the metal (positive)
- a = Number of atoms of the metal in the anionic complex
- X = Anionic ligand
- c = Number of atoms of the anion in the complex (positive)
- m = Charge of the anion in the complex (negative)
- R_3N = Anionic extraction reagent (tertiary amine)
- Y = Anion in the amine salt
- b = Valence of the acid (positive)

In a similar form that described in the previous section, we can write for this kind of systems:

$$K = \frac{\left[\left(R_3NH^+\right)_{-(na+cm)}\left(M_aX_c\right)^{na+cm}\right]\cdot\left[Y^{b-}\right]^{\frac{-(na+cm)}{b}}}{\left[\left(R_3NH^+\right)_b Y^{b-}\right]^{\frac{-(na+cm)}{b}}\cdot\left[\left(M_aX_c\right)^{na+cm}\right]} \quad (16)$$

We can use equation 16 and the distribution coefficient to obtain:

$$D = \frac{[M]_{org}}{[M]_{aq}} = \frac{\left[\left(R_3NH^+\right)_{-(na+cm)}\left(M_aX_c\right)^{na+cm}\right]/a}{\left[\left(M_aX_c\right)^{na+cm}\right]/a} \quad (17)$$

Combining both expressions we get:

$$D = K\left(\frac{[Y^{b-}]}{\left[\left(R_3NH^+\right)_b Y^{b-}\right]}\right)^{\frac{na+cm}{b}} = K\left(\frac{[Y^{b-}]}{[R_3N]/b}\right)^{\frac{na+cm}{b}} \quad (18)$$

Equation 18 allows to determine the theoretical distribution coefficient (D_{theor}) if the next values are known:

a) Equilibrium constant for the system (K)
b) Concentration of anionic extraction reagent in equilibrium (approximately equals to the initial concentration).
c) Concentration of the anion that forms amine salt, Y (it can be related to the pH of aqueous phase).

For multiple equilibria, a similar expression of equation 12 can be used. Experimental distribution coefficient is calculated from analytical determinations. The code for the program that resolves the minimization problem is similar to that presented for acid reagents (program *caleq2.m*) and is showed in the appendix. The function that define D_{theor} is included in the file *difcuad2.m*

3.3. Solvation (neutral) extraction reagents

The next expression is proposed:

$$\frac{-(na+cm)}{b} Y^{b+} + (M_a X_c)^{na+cm} + d\,L \Leftrightarrow \Leftrightarrow \left[Y_{\frac{-(na+cm)}{b}} \left(M_a X_c\right) \cdot d\,L \right] \quad (19)$$

Where:

- M = Metal
- n = Oxidation state of the metal (positive)
- a = Number of atoms of the metal in the complex
- X = Anionic ligand
- c = Number of atoms of the anion in the complex (positive)
- m = Charge of the anion in the complex (negative)
- L = Neutral ligand
- d = Stoichometric coefficient for the neutral ligand
- Y = Accompanying cation
- b = Charge of the cation (positive)

For this system we can write an analogous expression to that defined for previous system, thus:

$$K = \frac{\left[Y_{\frac{-(na+cm)}{b}} (M_a X_c) \cdot d\,L \right]}{\left[Y^{b+}\right]^{\frac{-(na+cm)}{b}} \cdot \left[(M_a X_c)^{na+cm} \right] \cdot \left[L\right]^d} \quad (20)$$

Distribution coefficient is defined as:

$$D = \frac{[M]_{org}}{[M]_{aq}} = \frac{\left[Y_{\frac{-(na+cm)}{b}} \left(M_a X_c\right) \cdot d\,L \right] / a}{\left[(M_a X_c)^{na+cm} \right] / a} \quad (21)$$

Combining both expressions we get:

$$D = K \left[Y^{b+}\right]^{\frac{-(na+cm)}{b}} \left[L^d\right] \quad (22)$$

Equation 22 lets determine theoretical distribution coefficient (D_{theor}) if the next values are known:

a) Equilibrium constant for the system
b) Concentration of neutral ligand (it can be approximated to initial concentration).
c) Concentration of the cation that forms the neutral complex, Y.

Experimental distribution coefficient can be calculated following the same scheme that defined for previous systems. The code for the program *caleq3.m* is presented in the appendix with the file that defines D_{theor} (*difcuad3.m*). The data are also stored in columns in an external Excel file (*nnur003.xls*).

For the case of the synergistic extraction of a metal using a mixture of an anionic reagent (e.g. amines) and a neutral ligand (e.g. phosphine oxides), similar equilibrium equations can be used according to the mechanism proposed.

4. RESULTS FOR LEAST-SQUARES OPTIMIZATION

Figure 1 shows the results of the program for the data obtained for the system Cu^{2+} / NO_3^- / ACORGA M5640 / IBERFLUID, considering that the extraction mechanism corresponds to:

$$Cu^{2+} + 2\,HR \leftrightarrow CuR_2 + 2\,H^+$$
$$\log K = 2.11 \pm 0.09 \text{ (LETAGROP)}$$

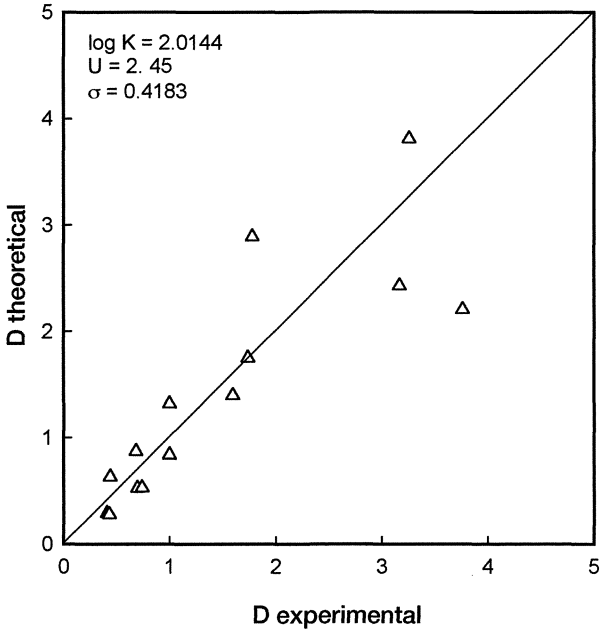

Figure 1. Results obtained for least-squares minimization using caleq1.m routine for modelling the system Cu^{2+} / NO_3^- / ACORGA M5640 / IBERFLUID.

Figura 1. Resultados obtenidos mediante la minimización de mínimos-cuadrados empleando la rutina caleq1.m para modelizar el sistema Cu^{2+} / NO_3^- / ACORGA M5640 / IBERFLUID.

Figure 2 shows the results of the program for the data obtained for the system Fe^{3+} / DEHPA / IBERFLUID, considering that the extraction mechanism corresponds to:

$$Fe^{3+} + HR \leftrightarrow FeR^{2+} + H^+$$
$$\log K_1 = [-1.6985\ -1.3642]\ \text{(LETAGROP)}$$
$$Fe^{3+} + 3\ HR \leftrightarrow FeR_3 + 3\ H^+$$
$$\log K_2 = -1.9943 \pm 0.0674\ \text{(LETAGROP)}$$

5. ARTIFICIAL NEURAL NET DESIGN

Although many different architectures have been reported, the use of back propagation (BP) neural networks are particularly widespread in chemical engineering research, owing to their simplicity, compact design and flexibility[7]. As mentioned in the introduction, this "model-free" estimator does not need to know an "hypothetic" mechanism for the liquid-liquid extraction process. We need only to know the variables that influence in the value of the distribution coefficient obtained for each experiment, because this system adaptively estimates continuous functions from data without specifying mathematically how outputs depend on inputs. In BP neural nets, units (neurons) are divided into distinct layers, as shown in figure 3. The processing

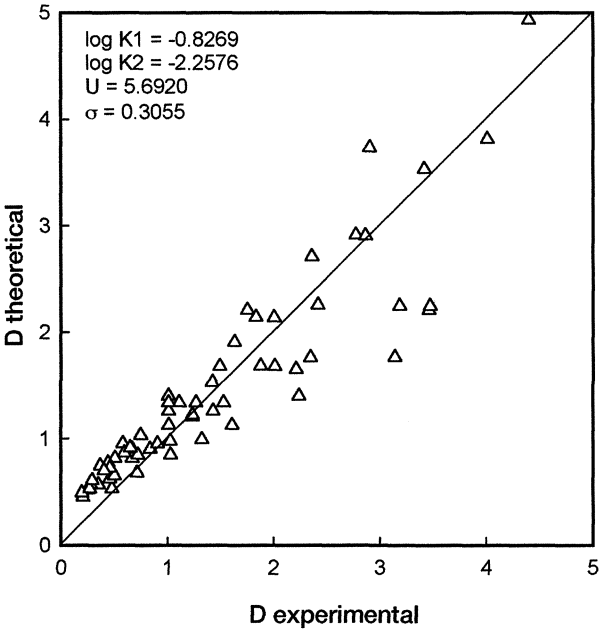

Figure 2. Results obtained for least-squares minimization using caleq1.m routine for modelling the system Fe^{3+} / DEHPA / IBERFLUID.

Figura 2. Resultados obtenidos mediante la minimización de mínimos-cuadrados empleando la rutina caleq1.m para modelizar el sistema Fe^{3+} / DEHPA / IBERFLUID

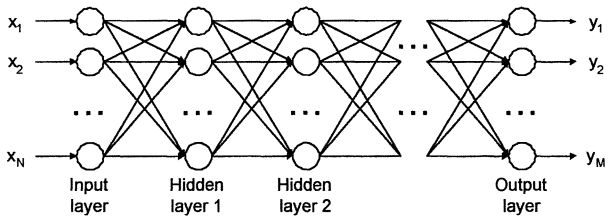

Figure 3. Structure of typical backpropagation neural net (biases have been omitted for simplicity of the scheme).

Figura 3. Estructura de una red neuronal con algoritmo back-propagation *(los bias se han omitido para simplicidad del esquema).*

units from each layer are linked to the processing units in sucessive layers by weighted connections. Each neuron processes its weighted inputs using an activation function, which results in an output for each neuron that passes to the next layer.

The methodology used to obtain a neural model is the next:

a) Depending on the activation function used in the neurons, input and output data vector must be scaled in an adequate range: [0 1] for sigmoidal function, [-1 +1] for hyperbolic tangent function, etc...
b) The first layer of neurons simply distributes the input data to the next layer, so the number of neurons of this layer is the same that the number of different input variables of our problem.
c) A bias or reference is added to each layer except at the input layer.
d) Neurons in the hidden and output layers receive weighted inputs from the previous layers, which are summed. The resulting weighted summation plus bias term are passed through the activation function choosen, so the output of each neuron is always in the range previously selected.
e) The learning process consists of identifying the weights that produce the best fit of the produced outputs over the entire training data set. The initial values for the weights are generally set to random values. During the training the weights are adjusted continuously based on the error generated by the discrepancy between the output of the network and the output of the training examples.

The initial structure chosen for modeling the liquid-liquid equilibrium data is shown in figure 4.

The number of input variables for the liquid-liquid extraction mechanisms discussed in previous section is four. Following the same notation, for a cationic extraction mechanism, we can assign:

1) $x_1 \rightarrow$ pH
2) $x_2 \rightarrow$ Meq (Metal concentration in aqueous phase in equilibrium)
3) $x_3 \rightarrow$ HmX (Concentration of acidic (cationic) extraction reagent)
4) $x_4 \rightarrow$ M0 (Initial concentration of metal in aqueous phase)
5) $y_1 \rightarrow$ Distribution coefficient

That means D= *f* (pH, Meq, HmX, M0). The common sigmoidal function will be used for the hidden layer neurons as first option. Table 1 shows the two data sets used for the design of the neural net.

Tabla I. Data sets for design of artificial neural net with back-propagation algorithm

Table I. Conjunto de datos para el diseño de la red neuronal artificial con el algoritmo back-propagation

Data set	pH	Meq ($mol \cdot l^{-1}$)	HmX ($mol \cdot l^{-1}$)	M0 ($mol \cdot l^{-1}$)	D_{exp}
	0.77	0.001117	0.009	0.001574	0.4085
	0.90	0.000929	0.009	0.001574	0.6949
	1.00	0.000787	0.009	0.001574	1.0000
	1.16	0.000574	0.009	0.001574	1.7397
	1.21	0.000331	0.009	0.001574	3.7619
	0.46	0.001102	0.018	0.001574	0.4286
1	0.60	0.000905	0.018	0.001574	0.7391
(training data)	0.70	0.000787	0.018	0.001574	1.0000
	0.81	0.000606	0.018	0.001574	1.5974
	0.93	0.000378	0.018	0.001574	3.1667
	0.24	0.001094	0.045	0.001574	0.4388
	0.31	0.000936	0.045	0.001574	0.6807
	0.40	0.000787	0.045	0.001574	1.0000
	0.57	0.000567	0.045	0.001574	1.7778
	0.63	0.000370	0.045	0.001574	3.2553
	0.84	0.001023	0.009	0.001574	0.5385
	0.95	0.000858	0.009	0.001574	0.8349
	1.08	0.000681	0.009	0.001574	1.3121
	1.19	0.000452	0.009	0.001574	2.4783
	0.53	0.001003	0.018	0.001574	0.5686
2	0.65	0.000846	0.018	0.001574	0.8605
(test data)	0.76	0.000696	0.018	0.001574	1.2599
	0.87	0.000492	0.018	0.001574	2.2000
	0.28	0.001015	0.045	0.001574	0.5504
	0.36	0.000862	0.045	0.001574	0.8265
	0.49	0.000677	0.045	0.001574	1.3256
	0.60	0.000468	0.045	0.001574	2.3613

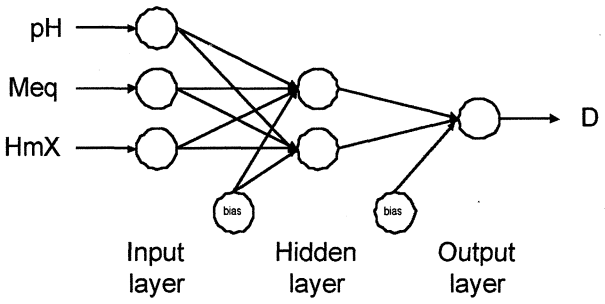

Figure 4. Structure of the BP neural net chosen for modeling liquid-liquid equilibrium data.

Figura 4. Estructura de la red neuronal BP escogida para modelizar los datos de equilibrio líquido-líquido.

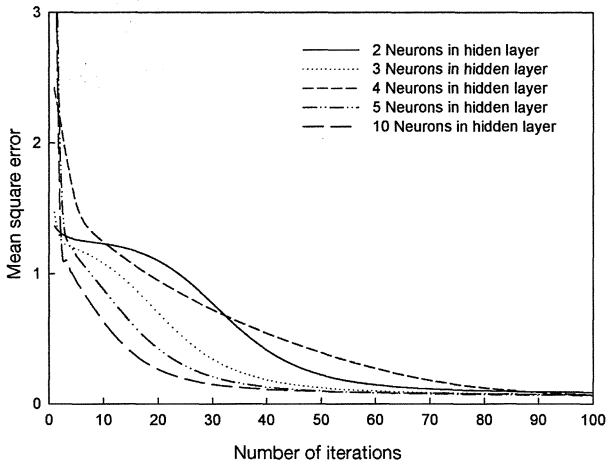

Figure 5. Evolution of the error in the training of the neural net.

Figura 5. Evolución del error en la fase de aprendizaje de la red neuronal.

Since M0 (Initial concentration of metal) is constant for all the data available, its influence will be included in the bias term, so the initial structure of the neural net is showed in figure 4.

6. RESULTS FOR THE NEURAL NET MODEL

The training algorithm developed in MATLAB reaches a minimum value after 100 iterations and is not dependant on the initial random values of the weights and biases, as can be seen in figure 5.

The final error of the neural net is below 0.07. Although small errors can be achieved using the backpropagation algorithm to adjust weights and biases in the neural net, it is necessary to check if the net emits appropriate response when faced with new data. The output of the net for the data set 2 showed in table I are summarized in table II and compared with results obtained with the K-model developed in previous section.

As can be seen, data predicted with this simple neural net fit better the experimental distributions coefficients.

Tabla II. Comparison of K-model and neural net model for predicting equilibrium data for a cationic system (D values are normalized in the range [0 +1])

Table II. Comparación de los modelos basados en la constante de equilibrio y la red neuronal en la predicción de los datos de equilibrio para un sistema catiónico (Los valores de D se han normalizado en el rango [0, 1])

D (K-model)	D (neural net model)	D_{exp}	$(D-D_{exp})^2$ (K-model)	$(D-D_{exp})^2$ (neural net model)
0.1041	0.1599	0.1431	1.52E-03	2.81E-04
0.1768	0.2089	0.2219	2.04E-03	1.70E-04
0.3218	0.3568	0.3488	7.29E-04	6.41E-05
0.5218	0.7621	0.6588	1.88E-02	1.07E-02
0.1022	0.1552	0.1512	2.40E-03	1.64E-05
0.1777	0.1958	0.2287	2.60E-03	1.08E-03
0.2881	0.2916	0.3349	2.19E-03	1.88E-03
0.4893	0.6139	0.5848	9.12E-03	8.46E-04
0.1975	0.1563	0.1463	2.62E-03	9.99E-05
0.2854	0.194	0.2197	4.32E-03	6.60E-04
0.5194	0.3258	0.3524	2.79E-02	7.06E-04
0.882	0.6896	0.6277	6.47E-02	3.83E-03
		$\Sigma =$	0.1389	0.0203

7. CONCLUSIONS

Statistical methods for fitting experimental liquid-liquid equilibrium data are quite developed but, in this particular case, requires for an "a priori" knowledge of the extraction mechanism for each system studied. In most cases, these mechanisms are an approximation of the real behaviour of the species implied due to the non-ideality of the system. Artificial neural nets can be easily used for fitting these experimental data without knowing "what happens" with the species that reacts in a liquid-liquid system. The design of the net must be adjusted for each case and standard backpropagation algorithm with sigmoidal activation function can offer adequate results.

APPENDIX

Program caleq1.m

```
clear all
global a j m n Dtheor Dexp;
data=xlsread('nnur001.xls');
pH=data(:,1);
Meq=data(:,2);
HmX=data(:,3);
M0=data(:,4);
xdata=[HmX pH];
n=input('Charge of metallic specie (positive): ');
m=input('Number of equivalents (protons) of the acid (positive): ');
j=input('Number of equilibria: ');
a=ones(1,j);
a=input('Stoichometric coefficient/s for the extraction reagent: ');
R=input('Phase ratio A/O: ');
K0=zeros(1,j);
K0=input('Initial values for equilibrium constants: ');
%
Dexp=R*(M0./Meq-1);
%
[K U]=lsqcurvefit('difcuad1',K0,xdata,Dexp);

% ---------- Results ----------
-
K
logK=log10(K)
U=U/2
sigma=sqrt(U/(length(Dexp)-j))
plot(Dexp,Dtheor,'b+',[0 max(Dexp)],[0 max(Dexp)],'r-')
axis([0 max(Dexp) 0 max(Dexp)])
xlabel('D experimental')
ylabel('D theoretical')
```

File *difcuad1.m*:

```
function [Dtheor]=difcuad1(K,xdata)
global a j m n Dtheor Dexp;
Dtheor=zeros(length(Dexp),1);
for i=1:j;
   D=K(i).*(xdata(:,1).^a(i))./(10.^(-xdata(:,2))).^(m*a(i));
   Dtheor=Dtheor+D;
end
```

Program caleq2.m

```
clear all
global a b c j m n Dtheor Dexp;
data=xlsread('nnur002.xls');
Y=data(:,1);
Meq=data(:,2);
R3N=data(:,3);
M0=data(:,4);
xdata=[R3N Y];
n=input('Oxidation state of the metal: ');
b=input('Number of equivalents (protons) of the acid (positive): ');
j=input('Number of equilibria: ');
a=ones(1,j);,c=ones(1,j);,m=ones(1,j);
a=input('Number of atoms of the metal in the complex: ');
c=input('Number of atoms of the anion in the complex: ');
m=input('Charge of the anion in the complex (negative): ');
R=input('Relacion de fases A/O: ');
K0=zeros(1,j);
K0=input('Initial values for equilibrium constants: ');
%
Dexp=R*(M0./Meq-1);
%
[K U]=lsqcurvefit('difcuad2',K0,xdata,Dexp);
% ---------- Results ----------
-
K
logK=log10(K)
U=U/2
sigma=sqrt(U/(length(Dexp)-j))
plot(Dexp,Dtheor,'b+',[0 max(Dexp)],[0 max(Dexp)],'r-')
axis([0 max(Dexp) 0 max(Dexp)]);
xlabel('D experimental');
ylabel('D theoretical');
```

File *difcuad2.m*

```
function [Dtheor]=difcuad2(K,xdata)
global a b c j m n Dtheor Dexp;
Dtheor=zeros(length(Dexp),1);
for i=1:j;
   D=K(i).*(xdata(:,2)./(xdata(:,1)/b)).^((n*a(i)+c(i)*m(i))/b);
   Dtheor=Dtheor+D
end
```

Program caleq3.m

```
clear all
global a b c d j m n Dtheor Dexp;
data=xlsread('nnur003.xls');
Y=data(:,1);
Meq=data(:,2);
L=data(:,3);
M0=data(:,4);
xdata=[L Y];
n=input('Oxistion state of the metal: ');
b=input('Charge of the cation: ');
j=input('Number of equilibria: ');
a=ones(1,j);,c=ones(1,j);,d=ones(1,j);,m=ones(1,j);
a=input('Number of atoms of the metal in the complex: ');
c=input('Number of atoms of the anion in the complex: ');
m=input('Charge of the anion in the complex (negative): ');
d=input('Stoichometric coefficient for the neutral ligand: ');
R=input('Phase ratio A/O: ');
K0=zeros(1,j);
K0=input('Initial values for equilibrium constants: ');
%
Dexp=R*(M0./Meq-1);
%
[K U]=lsqcurvefit('difcuad3',K0,xdata,Dexp);
%
% ———————— Results ————————————————————
————————-
K
logK=log10(K)
U=U/2
sigma=sqrt(U/(length(Dexp)-j))
plot(Dexp,Dtheor,'b+',[0 max(Dexp)],[0 max(Dexp)],'r-')
axis([0 max(Dexp) 0 max(Dexp)]);
xlabel('D experimental');
ylabel('D theoretical');
```

File *difcuad3.m*

```
function [Dtheor]=difcuad3(K,xdata)
global a b c d j m n Dtheor Dexp;
Dtheor=zeros(length(Dexp),1);
for i=1:j;

D=K(i).*(xdata(:,1).^d(i).*(xdata(:,2)/b)).^((n*a(i)+c(i)*m(i))/b);
  Dtheor=Dtheor+D
End
```

REFERENCES

[1] D.H. LIEM, *High-Speed Comput. Supplement to Graphical Methods* 12 (1971) 1521-1534.

[2] A.E. GILES, C. ALDRICH AND J.S.J. VAN DEVENTER, *Hydrometallurgy* 43 (1996) 241-255.

[3] E.J. MOLGA, *Chem. Eng. Process.* 42 (2003) 675-695.

[4] C. CHEN AND D. CHEN, *Comput. Chem.* 25 (2001) 541-550.

[5] A.P. ALEXANDRIDIS, C.I. SIETTOS, H.K. SARIMVEIS, A.G. BOUDOUVIS AND G.V. BAFAS, *Comput. Chem. Eng.* 26 (2002) 479-486.

[6] K. PIOTROWSKY, J. PIOTROWSKY AND J. SCHLESINGER, *Chem. Eng. Process.* 42 (2003) 285-289.

[7] A. CHOUAI, M. CABASSUD, M.V. LE LANN, C. GOURDON AND G. CASAMATTA, *Chem. Eng. Process.* 39 (2000) 171-180.

[8] L. GAO AND N.W. LONEY, *Comput. Chem. Eng.* 25 (2001) 1403-1410.

[9] B. KOSKO, *Neural Networks and Fuzzy Systems*. Prentice-Hall, New York, 1992, pp. 45-63.

[10] R.B. BOOZARJOMEHRY AND W.Y. SVRCEK, *Comput. Chem. Eng.* 25 (2001) 1075-1088.

[11] THE MATHWORKS, MATLAB *User Manual*, 2002.

[12] J.J. MORÉ AND D.C. SORENSEN, SIAM *J. Sci. Statistical Computing* 3 (1983) 553-572.

[13] R.H. BYRD, R.B. SCHNABEL AND G.A. SHULTZ, *Mathematical Programming* 40 (1988) 247-263.

[14] T. STEIHAUG, SIAM *J. Numerical Analysis* 20 (1983) 626-637.

[15] M.A. BRANCH, T.F. COLEMAN AND Y. LI, SIAM *J. Sci. Computing* 21 (1999) 1-23.

[16] T.F. COLEMAN Y Y. LI, SIAM, *J. Optimization* 6 (1996) 418-445.

[17] D.C. SORENSEN, *Technical Report TR94-27*, Department of Computational and Applied Mathematics, Rice University, 1994.

[18] T.F. COLEMAN AND A. VERMA, *Computational Optimization and Applications* 20 (2001) 61-72.