

## Part four

# DESIGN OF THE INTELLIGENT CONTROL STRUCTURE

*In this chapter, the control structure component blocks will be designed from both algorithmic and soft development points of view. Mainly, three bioprocess types will be analyzed: enzymatic hydrolysis of wheat straw, alcoholoxydase obtaining with the methylotrophic yeast **Hansenula polymorpha** and yeast growth on simulated liquor. Hence, it will be interpreted the configuration aspects and the soft implementation to the block level structure (i.e. filter, estimation, pattern recognition, fuzzy control and expert system). Moreover, a specific database will be configured on AS400 system, with specific structure for each analyzed case.*

## 4.1. THE EXPERT SYSTEM

Following the demonstration presented in #3.3, the expert system must select, based on *a priori* information outfitted by the human operator, the bioprocess class corresponding to the considered fermentation process. After the class selection, and based on pattern recognition (through a neural network), the corresponding bioprocess model evolution is chosen. Once the model selection, the optimal values can be (analytically) calculated and transferred as set point values to the fuzzy control block. Hence, the expert system performs as *supervisor* within the framework of the intelligent control structure.

Structurally, the expert system is based on human expert knowledge, formalized as logical rules. The term “rules” has a large meaning. It includes different elements and knowledge corresponding to the bioprocess, i.e. the main process parameters, the general evolution curves, etc. The knowledge is configured as syntactical forms, **if...then**. Hence, after the rule selection by the expert system, the corresponding bioprocess control strategy (based on human knowledge) is applied. Moreover, the expert system must supervise the whole control system, in order to change the global control strategy (if there is non-optimal) and even stop the bioprocess if the metabolic evolutions are troubled.

### 4.1.1. The *a priori* information

Corresponding to the bioprocess start time ( $t_0=0$ ), *a priori* information is needed (cf. #3.3). These information concern to process type selection. Three bioprocess types will be analyzed in this work:

- Enzymatic hydrolysis of a lignocellulosic substrate (wheat straw);
- Alcoholoxydase obtaining with the methylotrophic yeast *Hansenula polymorpha*;

- Yeast cultivation on simulated liquor.

Hence, the human operator must select one of these three bioprocess types. The expert system checks the corresponding routines (see Table 1) enclosed in each block. Obviously, the modeling/control structure is good defined from the commencement (evolution patterns, necessary estimated parameters, and fuzzy strategy); only the evolution type (e.g. inhibitory/non-inhibitory process, etc.) and the optimum parameter values must be determined.

These steps can be performed (from procedural point of view) through two files: SELECT.H and SELECT.C. Note that the function SELECT.C is based on a *case* structure: each bioprocess type calls the specific routines (see Table 1).

### 4.1.2. The knowledge database

The necessary knowledge for bioprocess control can be defined at this level because the expert system must coordinate the control structure. The human expert defines this knowledge for each instance.

*The knowledge database for the enzymatic hydrolysis of wheat straw*

In this case the knowledge database is composed of two working hypotheses:

- Non-inhibitory process; in this case the Michaelis & Menten equation is available;
- Inhibitory process; the evolution corresponds to 3.1.4.1 equation.

Table 1. The expert system operation selections vs. *a priori* information

Bioprocess type	Selection			
	Set Point Estimation	Pattern Recognition (NN)	Parameter Estimation (NN)	Fuzzy Control
Enzymatic hydrolysis	<ul style="list-style-type: none"> <li><math>v_{\max}</math></li> <li><math>v_{\max}, S_{\text{opt}}</math></li> </ul>	<ul style="list-style-type: none"> <li>non-inhibitory process;</li> <li>inhibitory process</li> </ul>	$(v, S)$ $K_1, K_2, K_3$	$v=f(S)$
Alcoholoxydase production	$X_{\max}$	<ul style="list-style-type: none"> <li>Gaden eq;</li> <li>Extended eq.</li> </ul>	$(dP/dt, dX/dt, S)$ $X_P, K_1, K_2$	$dP/dt=f(S)$
Yeast cultivation	$\mu_{\max}$	<ul style="list-style-type: none"> <li>non- inhibitory process;</li> <li>inhibitory process</li> </ul>	$(\mu, S, I)$ $K_1, K_2, K_3$	$\mu=f(S)$

The neural net for pattern recognition detects the most probable model evolution. Hence, the expert system selects (in the Set Point Estimation Block) the corresponding

procedure for the hydrolysis rate estimation: the LS (Less Squares) method (in the first case) and the conjugate gradient method (for the second one).

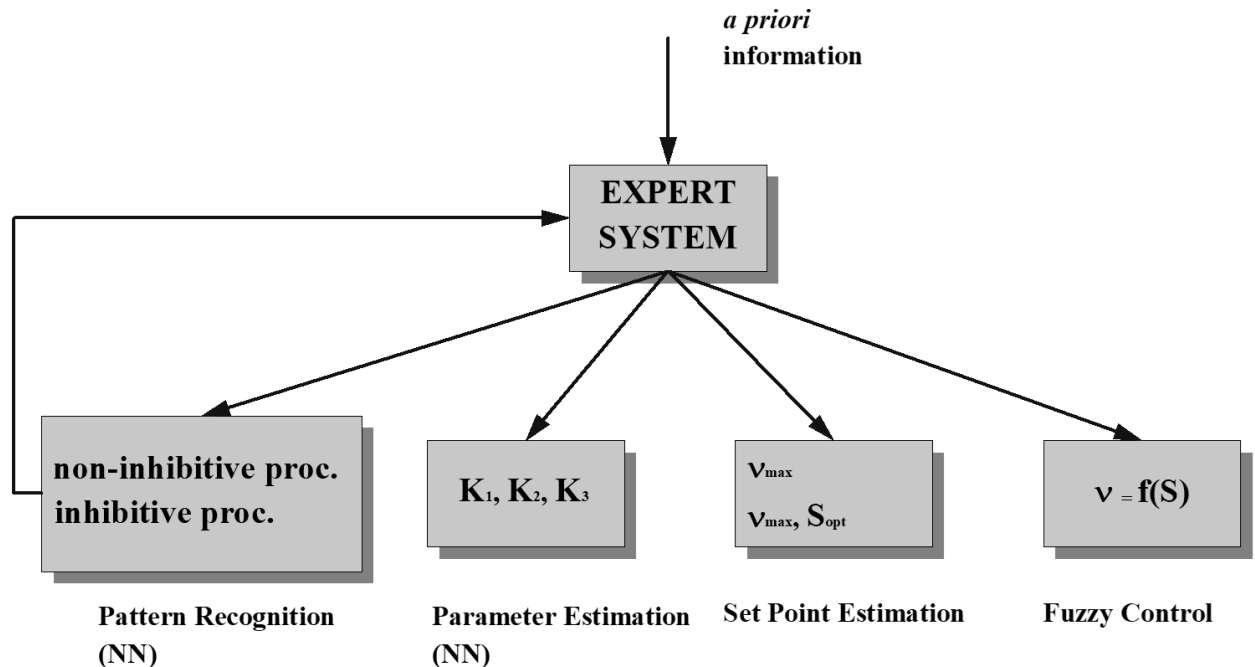


Fig. 4.1.1 Configuration structure for the control of enzymatic hydrolysis bioprocess

The Neural Net for Parameter Estimation is active only if equation 3.1.4.1 is checked (an

inhibitory bioprocess). Moreover, the expert system must stop the control structure

functionality if a component block of the intelligent control structure reports an irremediable error.

The soft implementation can be realized through two working files: KBEH.H and KBEH.C. The procedure KBEH.C is founded on a *case* statement with two possibilities: non- inhibitory and inhibitory bioprocess.

*The knowledge database for alcoholoxydase obtaining with the methylotrophic yeast Hansenula polymorpha*

In this case (cf. #3.1.3) the knowledge database consists of the following working hypothesis:

- The product formation respects Gaden equation (i.e. the cell concentration,  $X$ , is equal to the threshold value,  $X_p$ ;

- The product formation refers to equation 3.1.3.1.

Due to the fact that NNPR selects the bioprocess evolution curve, the expert system (through SPEB) picks out the corresponding procedure for  $X_p$  determination. If  $X=X_p$ , the bioprocess evolution agrees with Gaden model (cf. #3.1.3). The Neural Net for Parameter Estimation is active only if equation 3.1.3.1 is checked (extended Gaden model). Furthermore, the expert system must stop the control structure running if a component block of the intelligent control structure reports an irremediable error.

The soft configuration of knowledge database can be achieved through two files KBAP.H and KBAP.C; the procedure KBAP.C can be built on a *case* statement, too: Gaden model and extended Gaden equation.

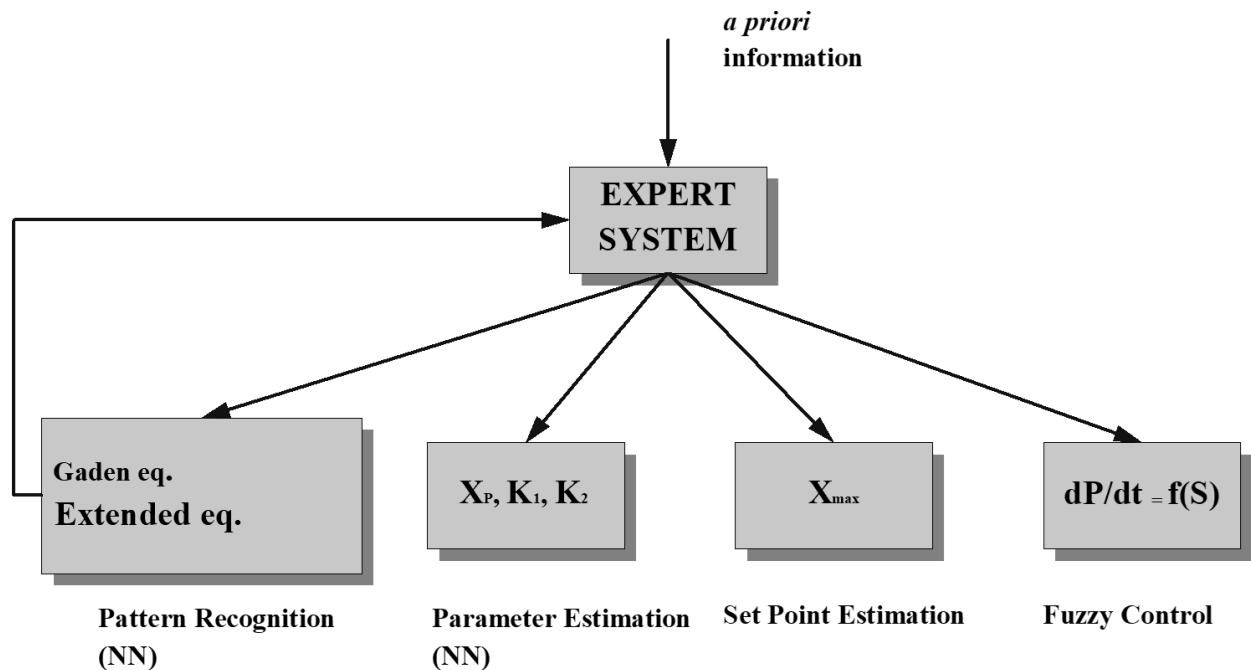


Fig. 4.1.2 Configuration structure for the control of alcoholoxydase production bioprocess

*The knowledge database for yeast cultivation on simulated liquor*

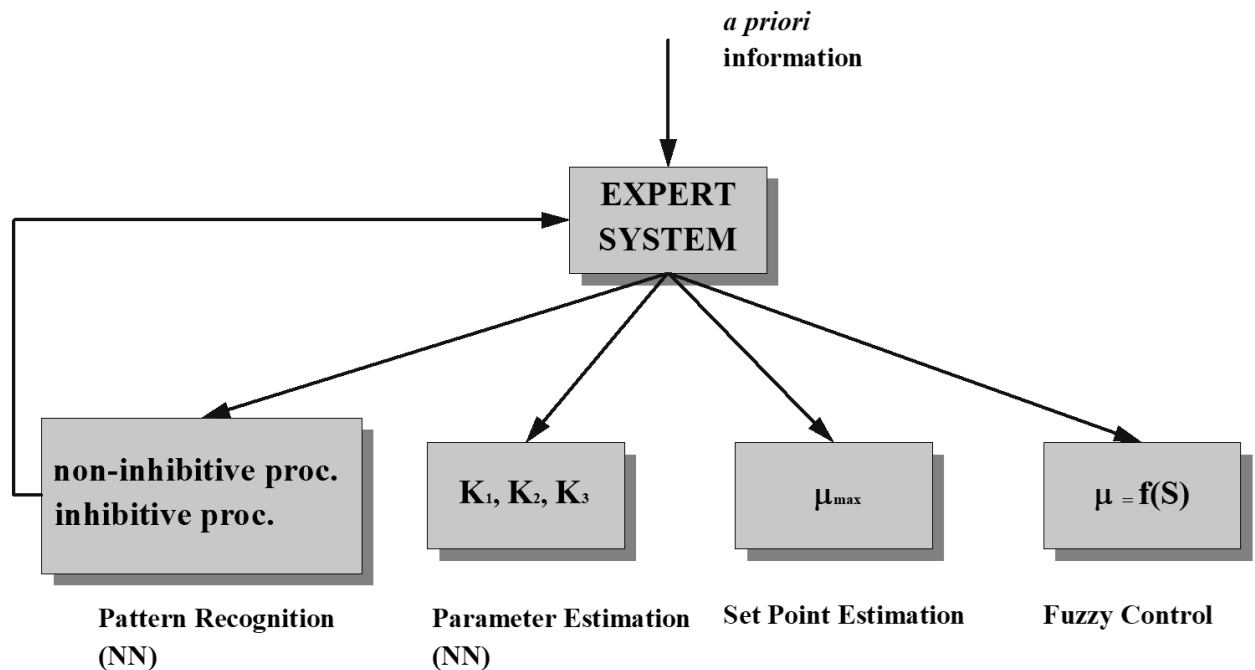
Conforming to #3.1.6, there are two bioprocess evolution possibilities, in relation with the presence/absence of inhibitors (i.e.

acetic acid). Hence, the process behavior can be conforming to the proposed equation (3.1.6.2) or with regard to Monod model. From this cause, there are two applicable hypotheses:

- Non-inhibitory process (i.e. Monod model);
  - Inhibitory process (i.e. equation 3.1.6.2).
- Once the pattern evolution curve is fixed, the expert system runs (through SPEB) the LS

procedure either for the specific growth rate ( $\mu_{\max}$ ) estimation (non- inhibitory bioprocess) or the conjugate gradients procedure (inhibitory bioprocess).

Like the precedent cases, the expert system ends the control process if the other component blocks signal an error.



**Fig. 4.1.3 Configuration structure for the control of yeast cultivation bioprocess**

The working hypotheses were realized through a *case* statement with two variants: non-inhibitory / inhibitory bioprocess. The corresponding files were KBYC.H and KBYC.C.

## 4.2. NEURAL NETWORKS

The most widely used and easily recognized neural network is the backpropagation network (Karim, Rivera, 1992). By some estimates (Welstead, 1994) it accounts for 90 percent of all neural network applications. The terminology *backpropagation* as a name for the network is misnomer, as this term actually refers to a training algorithm rather than network architecture. The network can be more properly called a *feedforward* neural network.

The widespread applicability of feedforward networks comes from the fact that *three-layer feedforward networks are universal approximators* (Hornik *et al.*, 1989; Kuschewski *et al.*, 1993). In fact, the Stone-Weierstrass theorem is used to prove this property. Note that this is an *existence* result, as opposed to a *constructive* result.

Following this fact, all neural networks used in this work will be designed with only *one* hidden layer.

### 4.2.1. Building a feedforward network structure

One of the details that must be addressed when building complete neural net structure is the matter of getting data to the network (Rohwer, Renals, 1991; Kazakos, Kazakos, 1989). In this section the necessary procedures to read, store and process training data will be described.

Hence, the header file TRN.H (for the training file procedure) defines a *training\_rec* structure, which holds all of the arrays and variables needed to store and manipulate the training data. Note the use of the variable *allocated*, which acts as boolean flag to indicate whether or not the arrays in the structure have been allocated in memory. This flag can be checked prior to freeing the allocated array pointers, rather than checking

each pointer separately (to avoid the potential problem of freeing an unallocated pointer – a sure way to crash the system).

The training data can be handled through an editor, in a text file. The procedure TRNPRCS.C is designed to manipulate the training data. Note that the training file does not directly provide the number of training items. This information is determined from the file size, i.e. the utility *open\_input\_text\_file* – in the UT.LIB library, header file UT.H (see Annex) – returns the file size. It is important that all the training data items occupy the same length in the file – obviously the procedure will not work.

The header file FF.H defines the types, variables, and functions needed for the feedforward network. A structure type called *network\_rec* defines the network itself. The hidden layer nodes as well as the weight arrays are of type double. The structure type *weights\_rec* contains both the input and output layer weight arrays.

The structure *ff\_setup\_rec* is the setup structure for the feedforward neural network. The explanation of each field is given below:

- *error\_threshold* – the error tolerance to be satisfied during training;
- *start\_iter* – normally set to 0; this field can have other values when training is interrupted and the user wishes to keep track of the total number of training iterations;
- *no\_of\_inputs* – number of inputs;
- *no\_of\_hd\_nodes* – number of hidden nodes;
- *no\_of\_outputs* – number of outputs;
- *in\_weight\_from\_file* – boolean flag: when set to *true* the order of training items is randomly shuffled using the function *shuffl* (UTSHUFFL.C in UT.LIB) at each iteration of backpropagation;

- *files* – substructure containing the file names defined in *files\_name\_rec* (UT.H);
- *sigmoid\_out* – boolean flag; indicates when a sigmoidal function is used in the output layer (*true*) or a linear function is used (*false*);
- *learning\_rate* – the learning rate;
- *weight\_rang* – the weights are initialized to random values between  $\pm$  *weight\_range*

Moreover, it is necessary to build an application library utilizable by each structure module (FF.LIB). Hence, the library consists of the following procedures:

- FFTRAIN: the feedforward net training procedure;
- FFBACKPR: the backpropagation training algorithm for feedforward network;

- FFDISPL: the procedure to display neural net training progress on screen;
- FFWTS: allocate, read in, write out feedforward network weights;
- FFEEDFW: procedure(s) for feedforward network.

The file FFEEDFW.C implements the definition of the feedforward network. The *sigmoid* function checks its arguments for size before applying the exponential function (in order to save potential floating-point overflow problems). The function *feedforward* receives its networks and weights arrays via parameters from an application-calling program.

The backpropagation algorithm is implemented in file FFBACKPR.C. The momentum weights values are initialized to 0.0 using *inititalize\_weights* (in FFWTS.C).

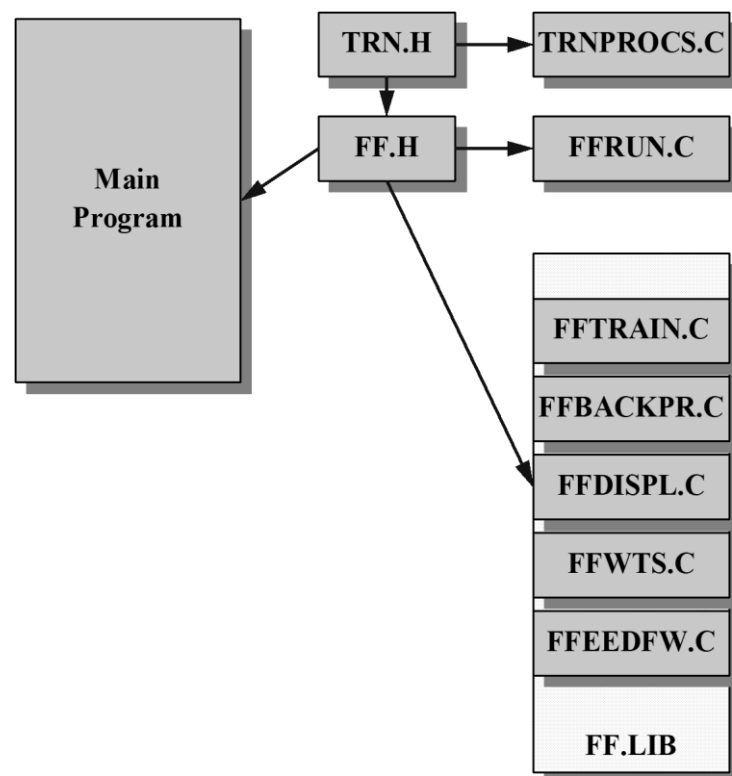


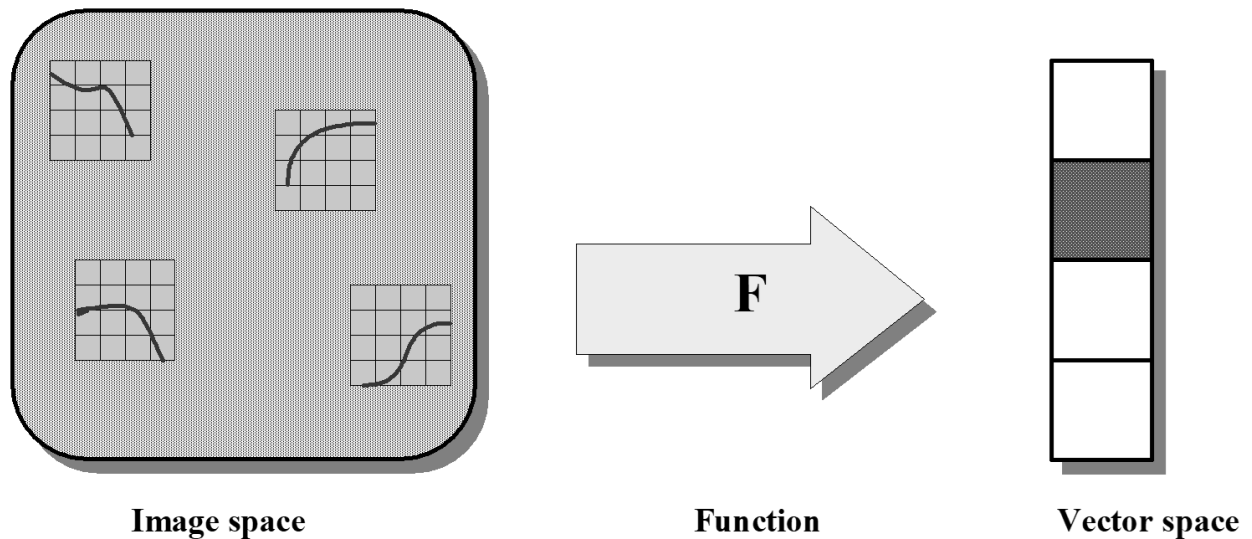
Fig. 4.2.1 Organization of files in the feedforward network system

The procedure *train\_network\_from\_data* in file FFTRAIN.C is the general-purpose training procedure; its logic flow (operational description) is shown below:

- Get the training data;
- Get the weights (either from a file or randomly initialized);
- Run the training algorithm;

- Free up the training data and weights.

Fig. 4.2.1 shows the file organization for this neural net implementation, indicating the relationships among the main program, procedures and header files to one another.



**Fig. 4.2.2** Image classification represented as a function from a space of images to a finite-dimensional vector space

Hence, the function assigns a specific vector to each image (Welstead, 1994). The neural networks can be considered as universal estimators (Wills *et al.*, 1991); they can approximate any well-defined function to any desired degree of accuracy. Thus, there is a feedforward network that can approximate the pattern classification function shown in fig. 4.2.2. The number of inputs to this network is the length of the image vector and the number of outputs is the length of the classification vector. As usual, it is not known the number of hidden-layer nodes. Conforming previous studies (Press, 1992; Chidambaram, 1992), a number corresponding to the number of output nodes seems to be sufficient.

In this work, we develop a system to classify simple 5x5 images (to adequately describe a formal image). The entry data are pattern

#### 4.2.2. Neural nets for pattern recognition

The pattern recognition procedure (Megan, Cooper, 1995; Megan, Cooper, 1994; Megan, Cooper, 1992; Hinde, Cooper, 1993) can be represented as a function from a space of images to a finite dimensional vector space (fig. 4.2.2):

evolution curves digitized as square images (pair of coordinates).

Hence, the header file PATTERNS.H defines the constants, types and variables needed for the pattern editing. The text characters BLOCK\_1 and BLOCK\_3 defined in the utility header UTYPES.H (see Annex) are the display characters for representing the two binary states for the image pixels. The binary states are represented internally by the value +1 (NODE\_HIGH) and -1 (NODE\_LOW). Furthermore, the structure *pattern\_display\_data* holds relevant information describing the type and number of patterns; this information comes from the user interface.

The file PATTERN.C contains the routines for reading and writing patterns to a file. The procedure *add\_noise\_to\_pattern* generates



*noisy* patterns by randomly changing some percentage of the pixels to the opposite binary values. This function is useful for performing experiments with the finished system, i.e. patterns corrupted with noise can be used as inputs to test the robustness of the trained network.

The pattern recognition network can be build based on four specific tasks; a program module handles each task:

1. *The module for new pattern data file generation* (necessary to symbolic bioprocess evolution representation): FFPDATA.C; the procedure *create\_new\_data\_file* and *new\_data\_file\_name* are utilized by the user interface.
2. *The module for pattern data file edition:* FFPEDIT.C, the pattern file to be edited is specified by *data\_file\_name*, and the edited file is *new\_data\_file\_name*. The two file names can be the same if it is not

necessary to save the old version of the data.

3. *The module for pattern data transformation into training data* (that is recognizable by the network): FFPCRT.C; the procedure *create\_training\_file* reads in a pattern data file and produces a training file in the appropriate format. This procedure assumes that the number of patterns is the same as the number of network outputs, i.e. each input pattern is the one and only representative of its class. The network output vector consists of all zeros except for a one appearing in the component number corresponding to the pattern number.
4. *The module for pattern recognition network running.*

The relationship of files specific to the pattern classification system is shown in Fig. 4.2.3.

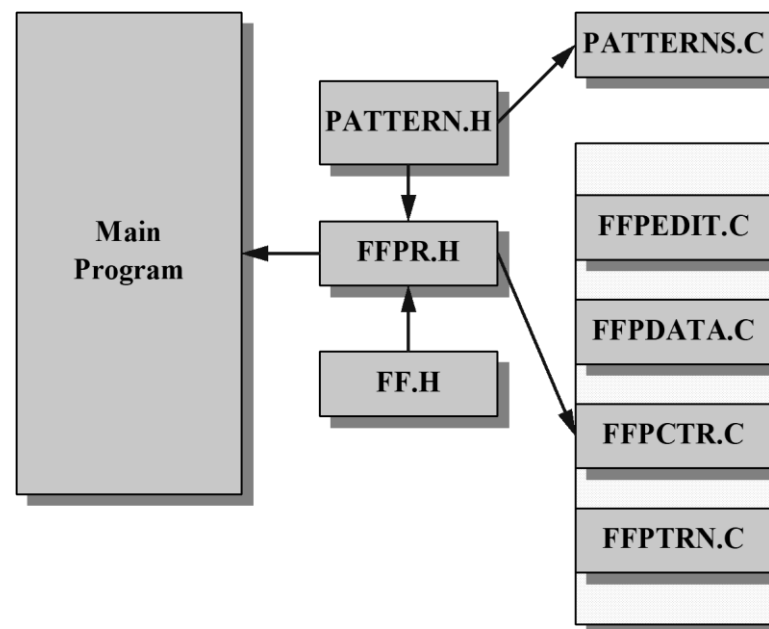


Fig. 4.2.3 Organization of files in the neural network for pattern recognition

#### 4.2.3. Neural net for parameter estimation

Taking into account the bioprocess specificity, the parametric implementation of neural nets presents distinct features, which will be

presented below. Hence, following the precedent demonstration, the neural networks can make good parameter estimation, if the training data is valuable. In the mean time, the bioprocess reproducibility is a relative

concept: there are only *qualitative* identically evolution curves (Garrido-Sanchez *et al.*, 1993; Muralikrishnan, Chidambaram, 1995); the *quantitative* identity is another desideratum (Fonteix *et al.*, 1995).

In the subsequent paragraphs it will be analyzed the neural nets for the specific bioprocesses, with focus on the specific particularities. From the structural point of view, the neural networks are identically designed:

- Each neural net is composed of three layers (one hidden layer);
- The neuron number from the hidden layer is just the same as the neuron number from the output layer.

#### 4.2.3.1. Enzymatic hydrolysis of wheat straw

Following the ideas presented in #3.1.4, the enzymatic hydrolysis rate is characterized by an equation  $v = f(S)$  which is strongly non-linear. Due to the laboratory measurement specificity, the substrate concentration and the enzymatic hydrolysis rate were determined off-line. For this reason, it is necessary to esteem these parameter values at each sample time. An adequate method is through an interpolating polynomial. In this case, the temperature and pH values are also determined.

Hence, the neural network structure for parameter estimation is shown in fig. 4.2.4:

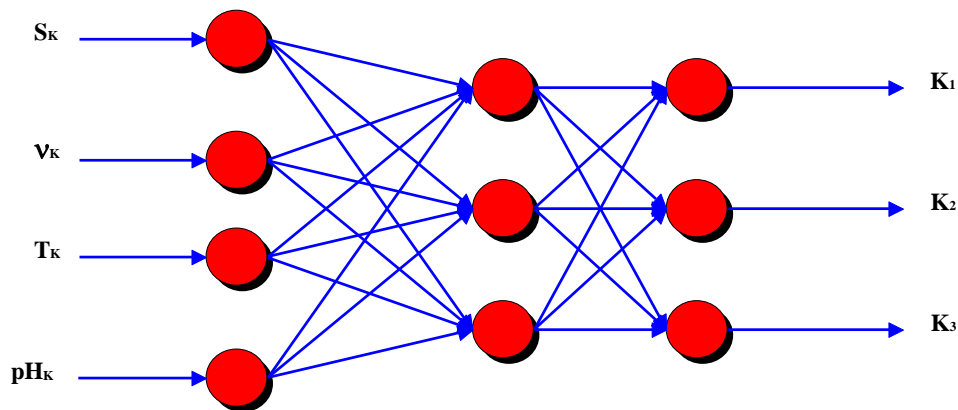


Fig. 4.2.4 Neural network for parameter estimation (enzymatic hydrolysis of wheat straw bioprocess)

The network is completely interconnected on account of theoretical impossibility to discern about the input influence on output variables.

Moreover, the weights are randomly initialized due to the scarcity of studies concerning the optimal values for weight initialization.

#### 4.2.3.2. Alcoholoxydase obtaining with the methylotrophic yeast *Hansenula polymorpha*

In this case (see #3.1.3) the on line measured parameter are temperature, pH, and dissolved oxygen. Substrate concentration, specific growth rate (i.e. dry weight) and enzymatic activity (product obtaining rate) are determined off line. Hence, it is necessary to esteem the threshold value of cell concentration ( $X_P$ ) and the model parameters (eq. 3.1.3.1). The network configuration is shown below:

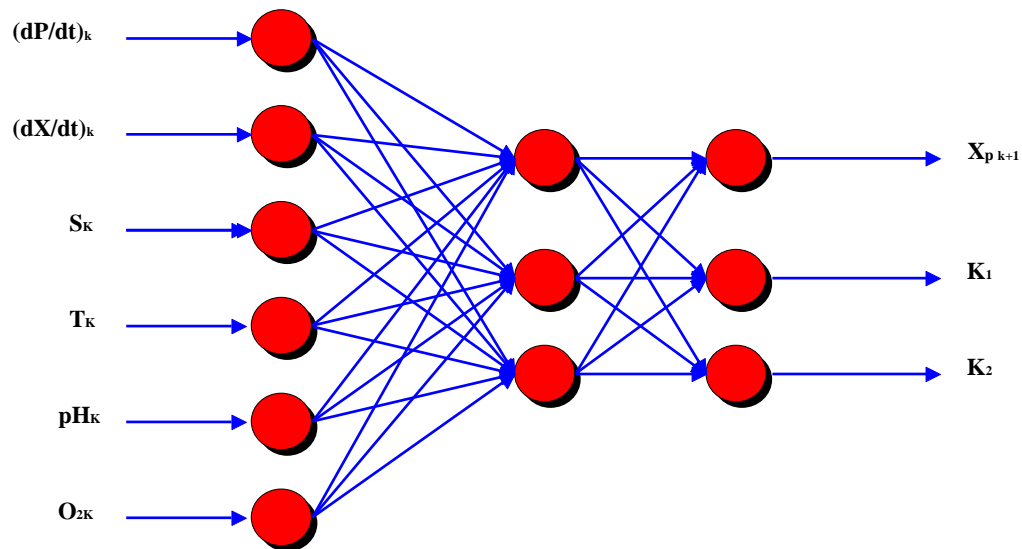


Fig. 4.2.5 Neural network for parameter estimation (alcoholoxydase obtaining)

The network is completely interconnected, too, for the same reasons as the previous case. Also, the weights are randomly initialized due to the scarcity of studies concerning the optimal values for weight initialization.

#### 4.2.3.3. Yeast cultivation on simulated liquor

For this type of bioprocess (cf. #3.1.6) the inhibitor concentration time variance and the specific growth rate connection with inhibitor

concentration (eq. 3.1.6.2) are to be taken into consideration. In this case the network inputs are cell concentration (measured off line), substrate concentration (measured off line), inhibitor concentration (measured off line), temperature and stirrer speed. The off line variables are esteemed at each sample time through an interpolating polynomial.

The network output variables are  $K_1$ ,  $K_2$ ,  $K_3$  (see eq. 3.1.6.2). The network configuration is presented in fig. 4.2.6:

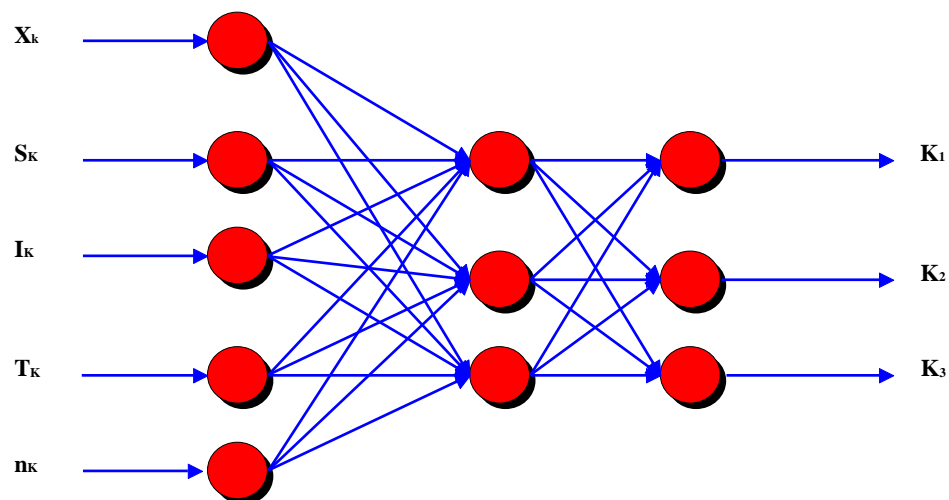


Fig. 4.2.6 Neural network for parameter estimation (yeast cultivation on simulated liquor)

Similar to precedent cases and for the same considerations the network is completely interconnected and the weights are randomly initialized.

#### 4.2.4. Neural network for pattern recognition design

Transformation of characteristic curves in (low resolution) standard pictures remove details but provides noise reduction and faster calculation (Garrido-Sanchez, *et al.*, 1993). Moreover, a local evolution variation has a punctual influence on the general bioprocess evolution and sampling time variation has only few influences. Indeed, each pixel usually corresponds to several experimental data, and thus, this method retains *only the general shape* of the curve. Picture size is chosen as small as possible to ensure correct determination of kinetic facts.

The neural network utilization is focused in these conditions on curve classification without parameter identification, in view of bioprocess qualitative evolution type.

From a structural point of view, the neural network contains 25 inputs (according to  $y=f(x)$  graphic formalization, i.e.  $(x_1, y_1); (x_2, y_1); \dots (x_1, y_2); \dots (x_5, y_5)$ ) and a number of outputs corresponding to the number of classification curves (e.g. bioprocess with/without inhibition, etc.). The neuron number on the hidden layer is identical to the neuron number on the output layer (cf. #4.2.3).

Concerning the former, human expert selects theoretical curves (*a priori* information) that determine the exemplar vectors. The picture resolution is 5x5 and a pixel is highlighted (i.e. set to high value +1) if at least one point is present in this area. When the pattern recognition (classification) occurs, only a single output is selected, according to human expert classification.

##### 4.2.4.1. Neural net for pattern recognition applied in enzymatic hydrolysis of wheat straw

The extension of Michaelis & Menten model to enzymatic hydrolysis bioprocess (cf. #3.1.4) makes evident two process evolution possibilities, i.e. the classical natural process (the substrate concentration in medium is lesser than optimum value) and the inhibition process (eq. 3.1.4.1).

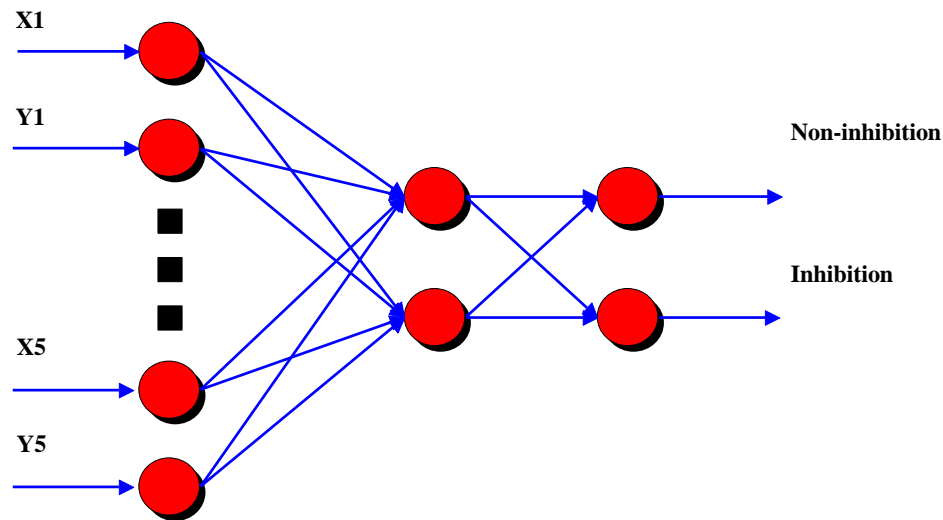


Fig. 4.2.7 Neural net for pattern recognition (enzymatic hydrolysis of wheat straw)

Hence, the neural network will have two outputs (and implicitly, two neurons on the hidden layer). The outputs accord with two linguistic variables: “non-inhibiting process”, i.e. equation Michaelis & Menten is applicable and “inhibition process”, i.e. the enzymatic hydrolysis process agrees with eq. 3.1.4.1.

The training values correspond to the square shown in fig. 4.2.8. Hence, the input values are:  $x_1=1$ ;  $y_1=1$ ;  $x_1=1$ ;  $y_2=0$ ;  $x_1=1$ ;  $y_3=1$ ...,  $x_4=0$ ;  $y_1=0$ ;... $x_4=1$ ;  $y_4=1$ ;... $x_5=0$ ;  $y_5=0$  (non-inhibition process), and  $x_1=1$ ;  $y_1=1$ ;  $x_1=1$ ;  $y_2=1$ ;  $x_1=1$ ;  $y_3=1$ ...,  $x_4=1$ ;  $y_1=1$ ;... $x_4=1$ ;  $y_4=1$ ;... $x_5=0$ ;  $y_5=0$  (inhibition process).

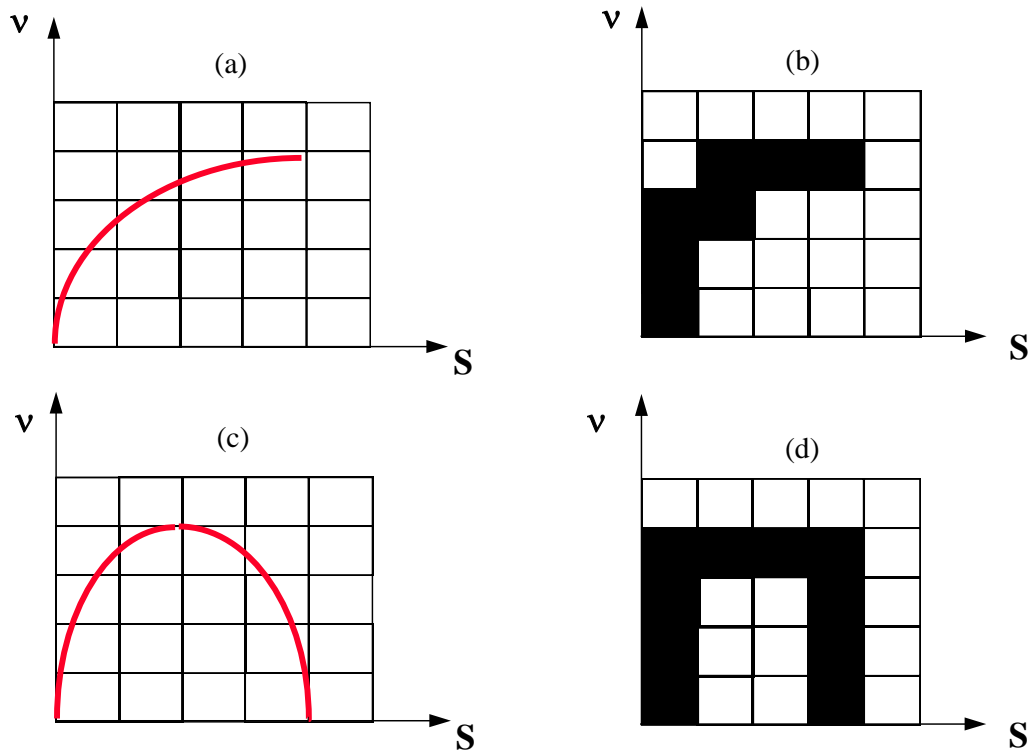
If the process evolution agrees with curve (a), the output “non-inhibition process” will be set; if the process evolution corresponds to curve

(b), the output “inhibition process” will be active.

The network is fully connected and the initial weight values are randomly initialized.

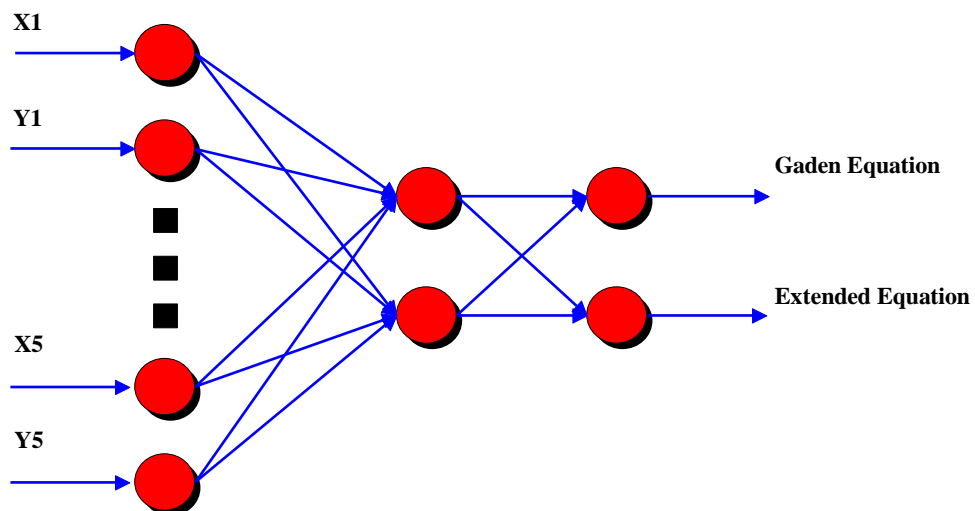
#### 4.2.4.2. Neural net for pattern recognition applied in alcoholoxydase obtaining with the methylotrophic yeast *Hansenula polymorpha*

There are three evolution classes for this bioprocess type, due to enzyme accumulation in the first region of the exponential phase (eq. 3.1.3.1). Taking into consideration the threshold value ( $X_P$ ), the three categories can be reduced to two: standard Gaden evolution (if  $X = X_P$ ) and conforming to eq. 3.1.3.1 otherwise.



**Fig. 4.2.8** The digitized evolution curves for the enzymatic hydrolysis of wheat straw; a), b) non-inhibition bioprocess; c) d) inhibition bioprocess

Hence, the neural net has 25 inputs, two outputs, and two neurons on the hidden layer, like the precedent case (see fig. 4.2.9).



**Fig. 4.2.9** Neural net for pattern recognition (alcoholoxydase production)

The two outputs are assigned to the following linguistic labels: “Gaden Equation” if the threshold value is equal to biomass

concentration and “Extended Equation” if the values are different.

An example of the digitized qualitative evolution curve for this bioprocess type is shown in fig. 4.2.10.

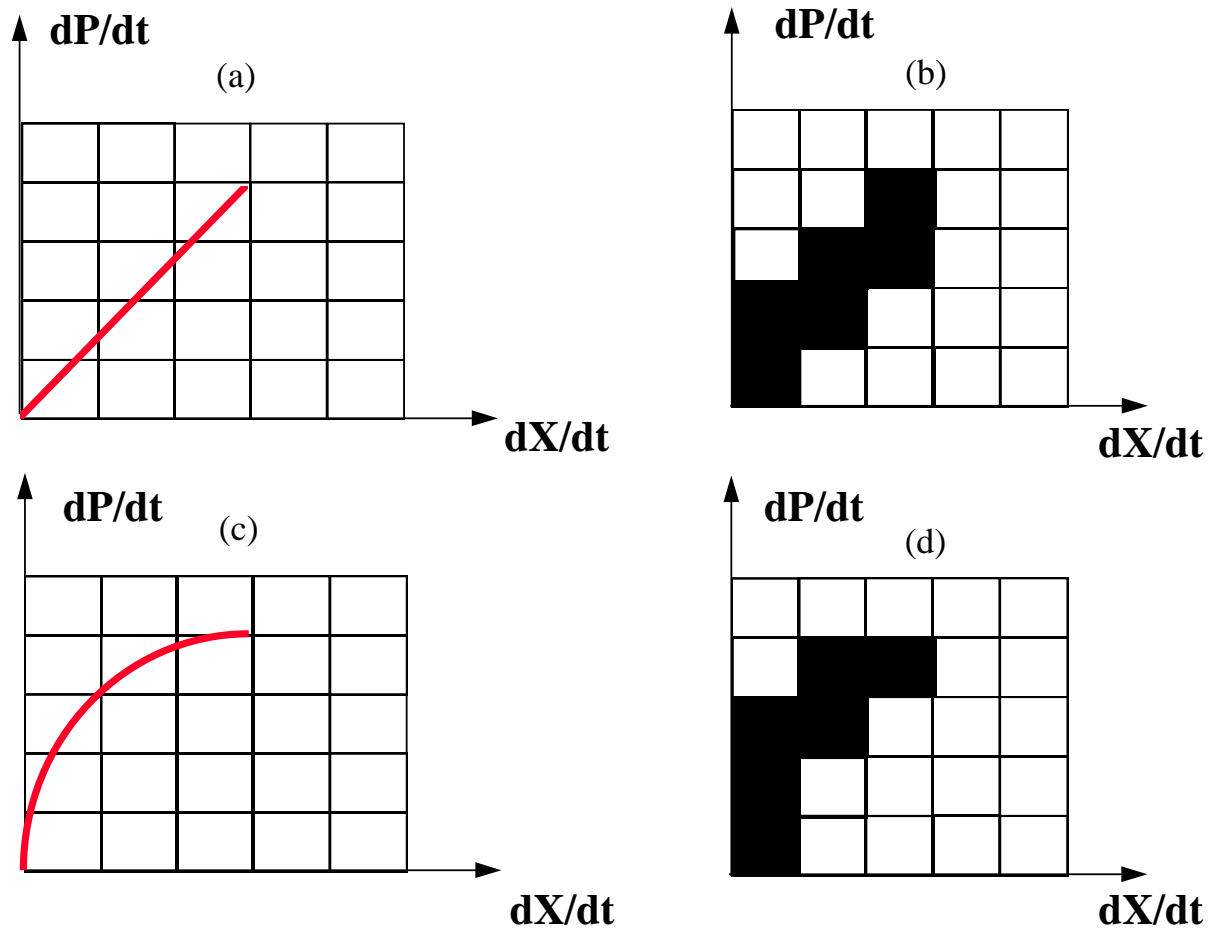


Fig. 4.2.10 The digitized evolution curves for an alcoholoxydase production bioprocess; a), b) classical evolution (conforming to Gaden eq.); c) d) non-standard bioprocess evolution (extended Gaden eq.)

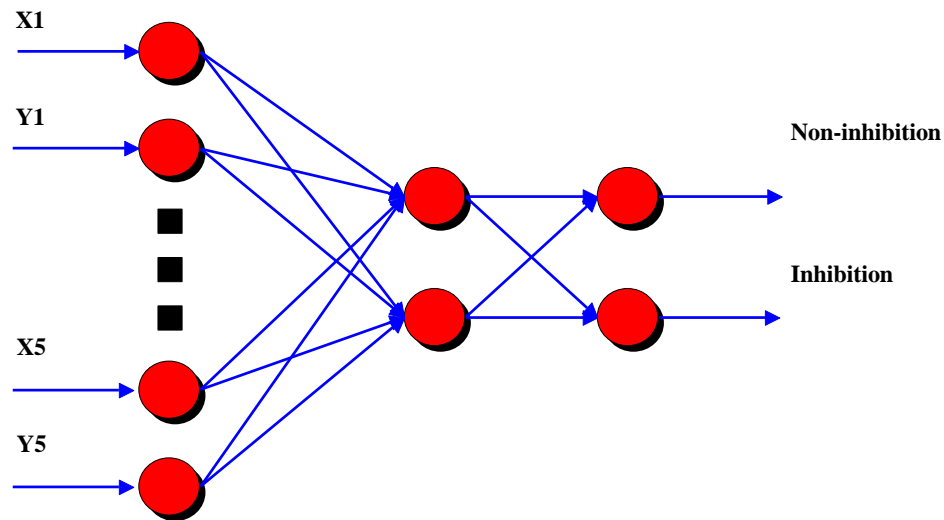
The training values agree with the pixel structure. The net is fully interconnected and the weight values are randomly initialized, too.

#### 4.2.4.3. Neural net for pattern recognition applied in yeast cultivation on simulated liquor

The yeast cultivation on simulated liquor focuses on inhibitory effect of the acetic acid delivered in medium (cf. #3.1.6). Hence, eq. 3.1.6.2 harmonizes with Monod equation taking into consideration the inhibitory effect. If the influence is low (inhibitor concentration

is small,  $I \cong 0$ ), the classical Monod relation is advantageous; in proportion as the acetic acid concentration increases (but not bigger than  $I_{lim}$ ) the population dynamic decreases and eq. 3.1.6.2 is to be applicable.

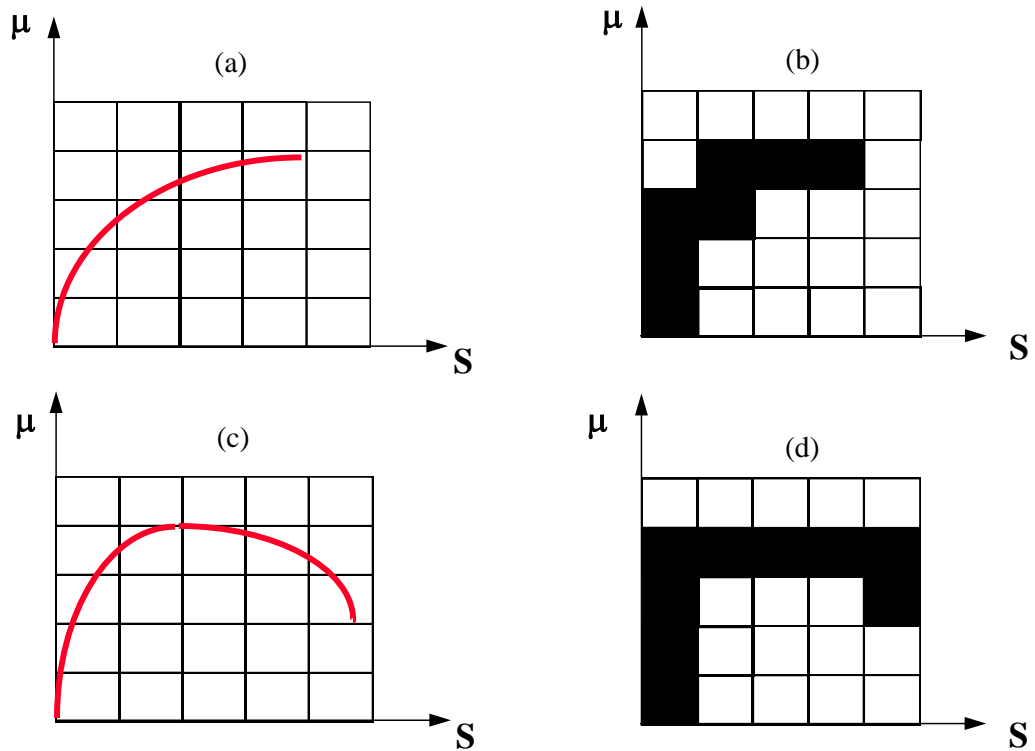
The neural net contains two outputs in agreement with the existence/absence of inhibitor.



**Fig. 4.2.11 Neural net for pattern recognition (yeast cultivation on simulated liquor)**

In the first case eq. 3.1.6.2 is to be used; in the second one, Monod relation can be applied.

The hidden neuron numbers is two, conforming to arguments presented above.



**Fig. 4.2.12 The digitized evolution curves for yeast cultivation bioprocess; a), b) non-inhibitory evolution; (c) d) inhibitory evolution**

The training values agree with the pixel structure. The net is fully interconnected and the weight values are randomly initialized, too.



### 4.3. FUZZY CONTROL BLOCK

#### 4.3.1. Overview

The design of a fuzzy control system arises from organization necessity of the human expert knowledge. The decisional quintessence of the control system is determined by the transition from the information *objective* level to the *subjective* one (i.e. the information version level) (Srinivas, Chidambaram, 1995). Thus, the interest is focussed on human expert experience (outlined through fuzzy rules) rather than information algorithmic process (Jecu, Caramihai, 1996).

Hence, in view of bioprocess specificity, the fuzzy control algorithm will be characteristic to a specific step evolution of the process (Besli *et al.*, 1995).

Generally, fuzzy membership functions are the mechanism used by the fuzzy system to interface with the outside world (Ramkumar, Chidambaram, 1995; Hu, Yuan, 1995). The domain of membership function is the set of possible values for a given variable and the possible output values are the real interval [0, 1].

In this work, a typical choice for a fuzzy membership function was preferred, i.e. the linear trapezoidal function. Other function shapes, such as gaussian function could also be used, but these would increase the computational price of the algorithm and provide no noticeable performance improvement (such functions typically appear in systems used for theoretical analysis, where their analytical properties, such as differentiability, are needed) (Kosko, Isaka, 1993).

Fuzzy rules combine two or more input fuzzy sets and associate an output to them. One method of storing and representing fuzzy rules is the use of a Fuzzy Associative Memory

(FAM) matrix. FAM matrices can have dimensions higher than two (i.e. the inputs number to the fuzzy rules determines the dimension). Hence, higher numbers of inputs produce “hypercube” FAM matrices. It is preferably to use more small FAM matrices (two- or three-dimensional) rather than a single higher dimensional matrix (Welstead, 1994), i.e. the system becomes more flexible and the computational power necessity is lower.

#### Defuzzification

The defuzzification process is based on SLIDE algorithm (Yager, Filev, 1993). Assume it is given a fuzzy set  $F$  with membership function  $w_i$  defined on the discrete universe of discourse  $X=\{x_1, \dots, x_n\}$ . It shall assume that  $M = \text{Max}_i[w_i]$ . The first step in the SLIDE method of defuzzification is to transform  $F$  into  $E$ , another fuzzy set defined on the same universe of discourse,  $X$ . It shall denote the membership grades in  $E$  as  $v_i$ . The  $v_i$ 's are derived from the  $w_i$ 's using the following transformation:

$$T_{\alpha, \beta}: F \longrightarrow E \quad 4.3.1$$

where:

$$v_i = \begin{cases} w_i, & \text{if } w_i \geq \alpha \\ (1 - \beta)w_i, & \text{if } w_i < \alpha \end{cases} \quad 4.3.2$$

Here  $\alpha$  and  $\beta$  are parameters of transformation such that  $\alpha \in [0, M]$ ,  $\beta \in [0, 1]$ .

The second step in the SLIDE defuzzification method is to normalize the sequence of  $v_i$ 's:

$$p_i = \frac{v_i}{\sum_{j=1}^n v_j}, \quad i = 1 \dots n \quad 4.3.3$$

It is easily seen that  $p_i$ 's have the characteristics of a probability distribution over the support set  $x_1 \dots x_n$ :

$$1) \sum_{i=1}^n p_i = 1 \quad 4.3.4$$

$$2) p_i \geq 0, \quad i = 1 \dots n \quad 4.3.5$$

The final step in the SLIDE defuzzification procedure is the determination of the defuzzified value:

$$d = \sum_i p_i x_i \quad 4.3.6$$

It is obvious that  $\alpha$  and  $\beta$  parameterize the family of all defuzzified values defined by the SLIDE method. The parameter  $\alpha$  can be regarded as a level of confidence. Its maximal value is associated with the maximal value of membership function. Parameter  $\beta$  characterizes the rejection of the membership values under the level of confidence,  $\alpha$ . For  $\beta = 1$ , the membership values under  $\alpha$  are rejected. For  $\beta = 0$ , the membership values under level of confidence are totally included. It also should be noted that if  $\alpha$  will be fixed  $\alpha = \text{Max}_I w_I$  then as  $\beta$  goes from 0 to 1 the SLIDE defuzzified value goes from the COA (Center of Area) to the MOM (Mean of Maxima) defuzzified values (Lee, 1990).

#### 4.3.2. Formalization

Fuzzy sets are determined by their membership functions; in this case trapezoidal functions are used.

There are three distinct types of trapezoid functions (fig. 4.3.1): those that open to left, those that open to right and the "regular" type

- closed at both ends. Note that the regular type does not have to be symmetric. Since the trapezoids are the graphs of membership functions, the maximum height is always 1. A regular trapezoid has three linear segments defined by the four points, a, b, c, and d. If  $b = c$ , the trapezoid becomes a triangle, which is also a valid membership function.

The header file FLOGIC.H specifies a structure type *trapezoid*, which contains all of the information needed to define the shape of a trapezoid. The structure includes an enumerative type specifying the trapezoid type (left, right, regular) and *float* values for the points a, b, c, and d. The structure also includes *float* fields for the slopes of the left- and right-slanting line segments in the trapezoid. While this slope information is redundant since it can be determined from the point values, it is included in the structure to save computational time since these values are needed every time the trapezoid function is evaluated.

Two procedures in the file FLPRCS.C facilitate the use of the *trapezoid* structure type. Hence, the procedure *init\_trapz* returns a fully initialized *trapezoid* structure. If *trapezoid* were a C++ class then *init\_trapz* would be its constructor.

The C function *trapz* is the actual trapezoid function. It takes a *float* value and a *trapezoid* structure as its arguments and returns the *float* value that is the output value of the mathematical trapezoid function.

The concept of FAM matrix suggests that it should logically store fuzzy rules in a multidimensional array architecture. The number of rows and columns can be adjusted in order to accommodate different numbers of fuzzy sets per variable. Changing the dimensions of a storage array is a major code change (particularly in C), not only affecting the array architecture itself, but also likely effecting changes in the number of "for" loops that maneuver through it.

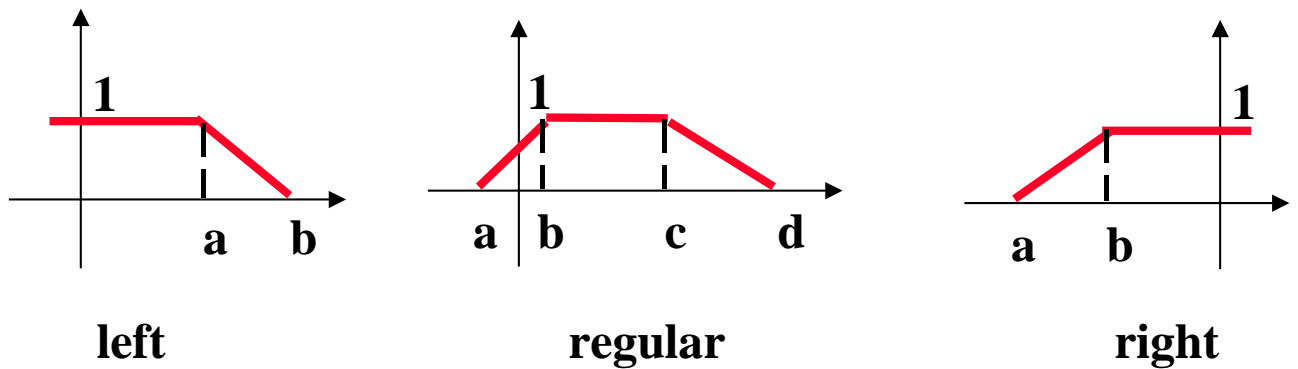


Fig. 4.3.1 Different types of trapezoids

An alternative to storing rules in a multidimensional array of fixed dimensions is to have each rule carry with it index information that specifies its location in a virtual multidimensional array. With this approach, the full array is not allocated; only those array entries that are actually used will be stored.

The structure type *rule* in FLOGIC.H defines the rules in this way. Hence, the one-dimensional array *inp\_index* holds the index values for the input variables. The one-dimensional array *inp\_fuzzy\_set* holds the index values for the corresponding fuzzy sets. These are index values that specify the location in the virtual FAM matrix. Finally,

the field *out\_fuzzy\_set* specifies the index value for the output.

The main advantages of this implementation type are:

- The ability to handle multiple lower-dimensional FAM matrices simultaneously;
- It is not necessary to specify all of the FAM matrix entries;
- Very compact implementation of the defuzzification procedure.

The structure type *fuzzy\_system\_rec* in the header file FLOGIC.H contains all of the information needed to completely define a fuzzy system, i.e. membership functions, rules and output values.

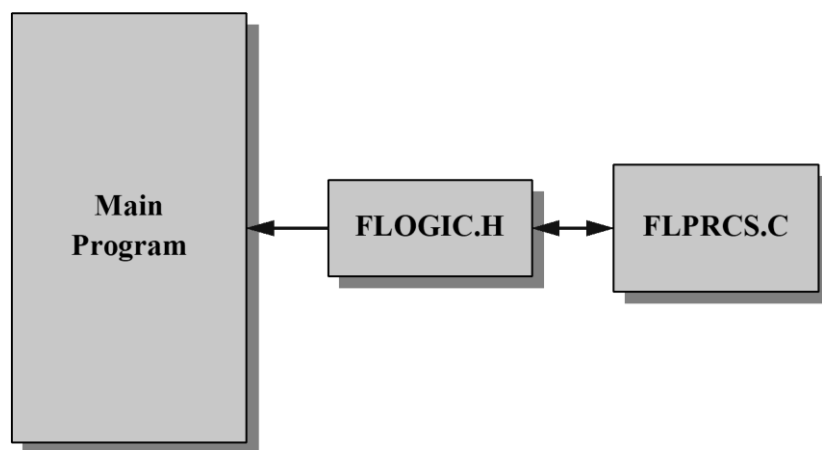


Fig. 4.3.2 Organization of files in the fuzzy control system

The procedure *fuzzy\_system* in FLPRCS.C takes the fuzzy system structure and an array of input values and returns the defuzzified

output value according to the scheme presented above (SLIDE algorithm).

#### 4.3.3. Implementation of fuzzy rules

1. Following the demonstration presented in #3.1.4, the enzymatic hydrolysis rate is nonlinearly dependent of substrate concentration (eq. 3.1.4.1). Hence, the fuzzy rules take into account the substrate

consumption rate and the enzymatic hydrolysis rate. Considering a fed-batch process, the substrate feed will be done in correlation with hydrolysis rate deviation from the optimum value, which is analytically determined. The conceptual control structure is shown in fig. 4.3.3.

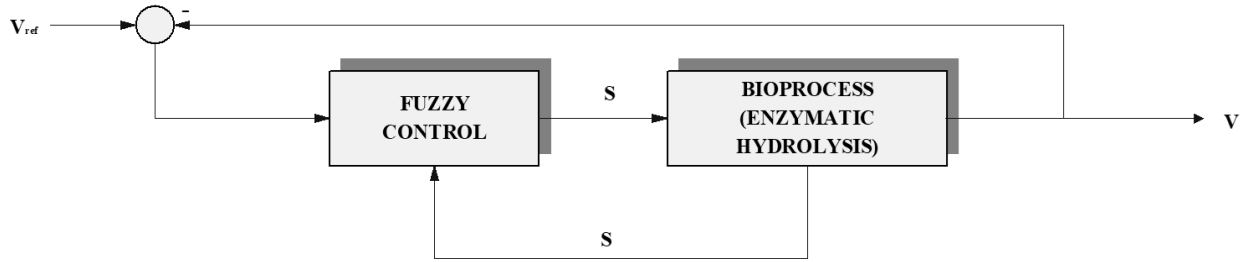


Fig. 4.3.3 Control structure of the enzymatic hydrolysis bioprocess

The fuzzy control input variables are substrate concentration and enzymatic hydrolysis rate error. The corresponding FAM matrix is presented below. The notation significance of fuzzy labels is Z= Zero, PS= Positive Small, PM= Positive Medium, PB= Positive Big

	Z	PS	PM	PB	Substrate Concentration
Z	X	PS	PS	Z	
PS	X	X	PS	Z	
PM	X	PS	PS	Z	
PB	PB	PM	X	Z	
Hydrolysis Rate Error					

Fig. 4.3.4 Fuzzy Associated Matrix for an enzymatic hydrolysis bioprocess

It can be seen that negative errors don't exist (i.e. the real value can not exceed the set-point

value) and there are only positive substrate concentrations ( $S \geq 0$ ). Moreover, some elements of FAM matrix cannot be defined due to the uncertainty regarding these cases. In all these circumstances, the process advances until a well-defined state.

2. As shown in #3.1.3, alcoholoxydase obtaining with the methylotrophic yeast *Hansenula polymorpha* is based on 3.1.3.1 equation. In view of the fact that product formation rate is greater than Gaden's case if cell concentration is smaller than the threshold value, product concentration (P) increasing can be performed through a set-point value  $X_{SP} < X_p$  and substrate concentration control. Moreover, the process will be stopped if the cell concentration exceeds the threshold value, ( $X > X_p$ ). The control structure is presented in fig. 4.3.5:

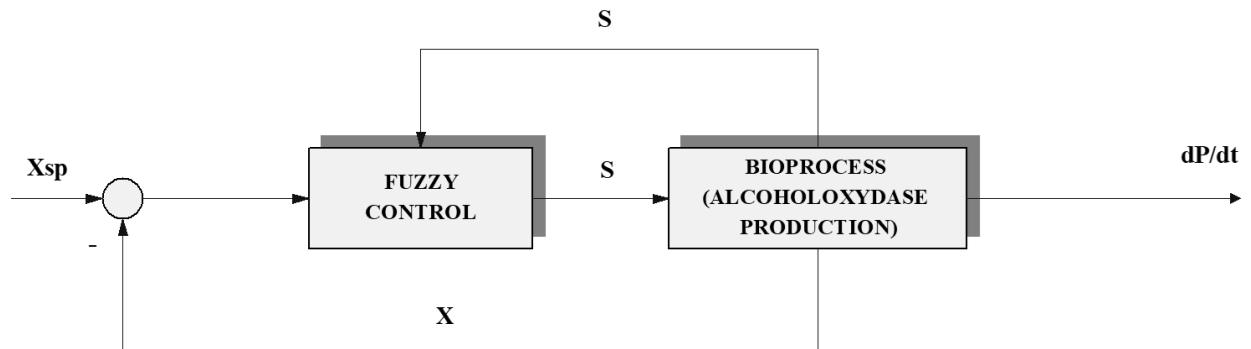


Fig. 4.3.5 Fuzzy control structure for alcoholoxydase production

The bioprocess output is dependent on substrate and biomass concentration, i.e.  $\frac{dP}{dt} = f(X, S)$ , but only by the substrate addition the product rate is controlled.

The FAM matrix (fig. 4.3.6) can be developed in two steps: firstly, global process evolution was investigated; secondly, performance improvement was considered. The notations are the same; in addition we have: NB= Negative Big, NM= Negative Medium, NS= Negative Small, N = Negative; P = Positive.

	Z	PS	PM	PB	Substrate Concentration
N	PB	PS	PS	Z	
Z	PB	PM	PS	Z	
P	PB	PM	PS	PS	
Biomass Concentration Error					

a)

	Z	PS	PM	PB	Substrate Concentration
NS	PB	PS	PS	Z	
NM	PB	PM	PS	Z	
NB	PB	PM	PS	PS	
Z	PB	PM	PS	Z	
PS	PB	PB	PB	PM	
PM	PB	PM	PS	PS	
PB	PB	PM	PS	Z	
Biomass Concentration Error					

b)

Fig. 4.3.6 Fuzzy Associated Matrix for an alcoholoxydase production bioprocess: a) first case; b) second case

3. Fuzzy control of yeast cultivation on simulated liquor (cf. #3.1.6) is based on substrate addition control, considering the specific growth rate dependence vs. substrate and inhibitor concentrations,  $\mu = f(S, I)$ , conforming to eq. 3.1.6.2. Hence,  $\mu$  deviation from the optimum value (analytically

determined) is due, on one hand, to inhibitor accumulation and on the other hand, to substrate concentration changes. Hence, a control structure is proposed (fig. 4.3.7), which can optimize cell growth and minimize inhibitor influences through substrate concentration.

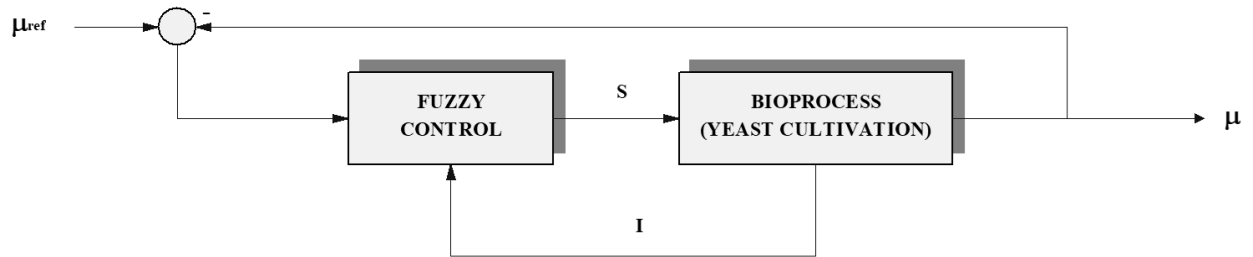


Fig. 4.3.7 Fuzzy control structure for yeast cultivation on simulated liquor

The inputs of the fuzzy controller are specific growth rate error and inhibitor concentration and the output corresponds to the substrate concentration. From a technological point of view, the specific growth rate (implicitly) offers information on substrate concentration. Hence, the following FAM matrix can be proposed (fig. 4.3.8).

As can be seen, specific cases (e.g. big inhibitor concentration with zero error) cannot be defined, conforming to real process behavior. Moreover, “zero” substrate addition is a “waiting” command, in order to check further process evolutions.

	Z	PS	PM	PB	Inhibitor Concentration
Z	Z	PM	<b>X</b>	<b>X</b>	
PS	PB	PB	PB	Z	
PM	PM	PM	PB	Z	
PB	PM	PB	PB	Z	
Specific Growth Rate Error					

Fig. 4.3.8 Fuzzy Associated Matrix for yeast cultivation on simulated liquor

#### 4.4. FILTER BLOCK

The control bioprocess procedures are generally built based on quantitative approach (i.e. correlation set-up) and starts with the measured/calculated process variables.

Hence, for an optimal control, it is necessary to filter the measured data, in order to eliminate the (inherent) mistaken information. The most possible causes that induce the data modification are:

- Signal lost on transmission lines;
- ADC (Analog/Digital Converter) defections;
- Measurement equipment errors.

These wrong data can generate undesirable phenomena in the information base post-processing:

- Increasing of the general level of noise;
- Induction of frequency false components via spectral density.

Consequently, it is necessary to detect and eliminate the wrong results from the bioprocess data sets.

Hence, it can be considered that correct data have a smooth evolution and the wrong values are deviations from this hypothesis. So, the following sequences can be defined:

- $[\bar{X}(i)]^2$  - The sequence  $X(t)$  is firstly smoothed and secondly squared;
- $\bar{\bar{X}}^2(i)$  - The sequence  $X(t)$  is firstly squared and secondly smoothed.

The first step is to generate the dispersion-actualized values:

$$s^2(i) = \bar{\bar{X}}^2(i) - [\bar{X}(i)]^2 \quad 4.5.1$$

and to calculate the standard deviation.

The second step is to test the next value of data sequence,  $X(i+1)$ ; this one is admitted as real (exact) value if:

$$\bar{X}(i) - K < X(i+1) < \bar{X}(i) + K, \quad K \in (3, 9) \quad 4.5.2$$

If the above condition isn't performed, the value  $X(i+1)$  will be replaced with the following expression (equivalent with a linear extrapolation):

$$\hat{X}(i+1) = 2X(i) - X(i-1) \quad 4.5.3$$

The procedure must be supplemented with an additional condition that restricts the successive number of extrapolations (i.e. a permanent extrapolation must be avoided). On that account, the extrapolated values can take additional direction to real trajectory, after the detection of wrong value series (and thus, the correct data will be rejected).

Another filter procedure is based on Tukey's theory. This one considers that is more comfortable to primarily generate an estimation of the smooth data evolution and then to reject the wrong values. The procedure utilizes the theorem that considers that median value of a data sequence represents a robust estimation. The algorithm is given below:

1. The sequence  $X'(i)$  is built firstly, based on median value of the sequence  $X(1)....X(5)$ . This will be  $X'(3)$ .  $X(1)$  is removed and  $X(6)$  added to data sequence. The new median value will be  $X'(4)$ . The procedure goes ahead till all data will be covered. Note that the sequence  $X'(i)$  will have four elements less than initial sequence;
2. The sequence  $X''(i)$  will be built based on  $X'(i)$  succession, like the precedent step; in this case the element number of the data

set which determines the mean value is three;

3. Finally, the sequence  $X'''(i)$  will be calculated in agreement with the following relation:

$$X'''(i) = \frac{1}{4} X''(i-1) + \frac{1}{2} X''(i) + \frac{1}{4} X''(i+1)$$

#### 4.5.4

Note that this operation is comparable to a Hamming smooth;

4. The difference  $X(i) - X'''(i)$  is analyzed to decide if  $|X(i) - X'''(i)| > k$ . Note that the human expert fixes  $k$  value. If the above relation is true,  $X(i)$  is replaced with an interpolated value.

#### Implementation

The two-algorithm implementations are realized through two C-functions, NETED.C and TUKEY.C, which agree with the mathematical algorithms.

The algorithm selection is performed by the human operator through an user interface. The filtered data are stored in a working file.



## 4.5. SET POINT ESTIMATION BLOCK

The conceptual structure of an intelligent control system based on hybrid techniques implies the integration of contributions that the bioprocess control using *a priori* model can offer, i.e. analytical determination of maximum/minimum values according to corresponding model (Chouakri, *et al.*, 1994; Chouakri, *et al.*, 1995).

After the values were determined (Karim, Rivera, 1992), they are transferred to the control block as set-point values.

### 4.5.1. Enzymatic hydrolysis bioprocess of wheat straw

In agreement with #4.2.4.1, this kind of bioprocess is able to advance either on classical equation (Michaelis & Menten) or on the law defined by eq. 3.1.4.1. The neural network for pattern recognition performs the model selection. Hence, the set-point estimation block must

- Establish the value of specific reaction rate ( $v_{\max}$ ), if the bioprocess is non-inhibitory;
- Determine the values of specific reaction rate and (corresponding) substrate concentration, if the bioprocess is inhibitory.

In the first case, the determination of  $v_{\max}$  value can be performed through LS (Least Square) method, because Michaelis & Menten model agrees with parameter linear dependence. The algorithm can be implemented via a C-function, CMMP.C. The input parameters are substrate concentration and specific reaction rate and output parameters are considered  $v_{\max}$  and  $K_m$ . This function calls the UT library (see Annex) for matrices operations.

In the second case, parameter estimation is achieved through the Gradient Conjugates method.

The algorithm can be implemented via two files, CG.H (*header*) and CGPROCS.C. The function CGPROCS.C must find the minimum/maximum point of a function whose definition is passed via the parameter *obj\_fn*. The final vector solution is returned through the parameter weights and the value of the objective function is returned through the parameter *obj\_value*. The procedure stops either the imposed error tolerance (*error\_tol*) is exceeded or the solution is successfully found.

### 4.5.2. Alcoholoxydase obtaining with the methylotrophic yeast *Hansenula polymorpha*

Conforming to #3.1.3, the alcoholoxydase production (enzyme accumulation) is characteristic to the early exponential growth phase. Hence, the corresponding model is the Gaden equation (if the threshold value,  $X_p$  is equal to the cell concentration,  $X$ ), or 3.1.3.1 equation in all other cases.

Therefore, the set-point estimation block must find the cell concentration,  $X$ . For this purpose, the cell concentration is passed to the control block, incremented with 1%, till the threshold value is reached.

The determination of the cell concentration via equation 3.1.3.1 is performed through Runge-Kuta method, at each sampling period. The integration procedure is defined in distinct file, RK.C

### 4.5.3. Yeast growth on simulated liquor.

Following the ideas presented above (cf. #3.1.6), the yeast growth on simulated liquor occurs in accordance with specific conditions (i.e. acetic acid accumulation). In agreement with eq. 3.1.6.2, biomass growth can cumulate via the classical Monod model (in the inhibitor absence) or through a  $\mu=f(S,I)$  equation (in the

other case). The neural network for pattern recognition performs the model selection. The set-point estimation block must

- Establish the value of specific growth rate ( $\mu_{\max}$ ), if the bioprocess is non-inhibitory ( $I \cong 0$ );
- Determine the values of specific growth rate and (corresponding) substrate concentration, if the bioprocess is inhibitory (cf. eq. 3.1.6.2).

In the first case, the determination of  $\mu_{\max}$  value can be performed through LS (Least Square) method, because Monod model agrees with parameter linear dependence. The algorithm can be implemented via the C-function, CMMP.C. The input parameters are substrate concentration and specific growth rate and output parameters are  $\mu_{\max}$  and  $K_s$ . This function calls the UT library (see Annex) for matrices operations.

In the second case, parameter estimation is achieved through the Gradient Conjugates method, presented above.

## 4.6. AS400 DATABASE

### 4.6.1. Overview

AS400 systems are well known for its *ease for use*, *open standards* (whereby the system architecture and applications allow for expansion and scalability) and *security* (which helps for the acceptance of information technology) (Abergel, 1993, Rambaud, 1989). The bioprocess databases designed on this computer type offer the advantages of a large-scale information storing, which is directly or through a client/server structure available. The security level is based on access control (specific information should only be accessible by authorized personnel; controlled access allows persons with the required authorization to read and alter protected data files) and data integrity (it can be ensured that data are not changed by unauthorized persons after their transmission) (Caramihai *et al.*, 1996).

The main advantages of this kind of database are:

- Direct data access;
- Maximum operation flexibility;
- Easy to upgrade the system.

The AS400 database is configured as *physical* and *logical* files. The logical files are based on the information stored by physical files, which are structured conforming different criteria (IBM, 1996). When a physical file is created, the key fields, the record name and the field description must be defined. Furthermore, the data type (packed, alphanumeric, etc.) must be determined in order to fix the database consistency.

If the physical files hold data, the logical files arrange the information by taking into consideration different criteria (*keys*). Hence, the main logical file characteristics are (Pichat, Bodin, 1990):

- The data can be treated conforming to operator option;

- The information selection (in the physical file) is directly performed;
- A logical file can joint different physical files for better data use.

The physical/logical files can be accessed through display files and reported through printer files (IBM, 1996).

A *printer file* consists of more *formats*. The ordinary format is a printing line. Each format consists of zone description (i.e. length, attribute, key). The entire data ensemble is stored in a PRTF source file.

Like the printer files, the display files consist of more formats. In this case, the ordinary format is the screen line. Each format consists of zone description (i.e. length, attribute, key). The entire data ensemble is stored in a DSPF source file.

### 4.6.2. Hard & soft configuration

The following hardware configuration is necessary for the AS400 database design in a client/server structure:

- AS400 computer with OS400/305 operating system;
- Novell network server;
- PAC network server;
- Eight workstations.

This configuration assumes a soft development on the workstations, program generation on PAC server and program compiling on AS400.

The programs were developed in PACBASE (version 8.0.2, 1995; copyright GNA/France). Hence, the procedures were designed in PAC; the source were generated in COBOL, transferred on AS400 and compiled.

The access to the system resources can be made under OS400 control or under

client/server configuration (i.e. through RUMBA, under Windows95 operating system).

The PACBASE developing medium is a great resources consumer on account of special predefined functions.

The PAC programs run only on AS400 systems. There are three program types:

- Display programs (TP): the operator can directly bring up to date the database;
- Batch programs (B): the operator can indirectly bring up to date the database (periodically or by request);
- Report programs

These programs can be launched via operator request or through CL routines (periodically).

The AS400 system communicates with the intelligent control system through a specific ODBC.

#### 4.6.3. Design of database. A case study.

##### 4.6.3.1. Database for yeast and fungi cultivation

Following the ideas presented above (cf. #3.1), the main parameters for a yeast or fungi cultivation bioprocess are pH, dissolved oxygen, stirred speed and pressure (for mechanical reactors), substrate concentration, dry weight (as measurement of cell concentration), specific growth rate and product concentration (product formation rate). The corresponding database will include the specific physical files corresponding to these parameters, logical files (for direct access) and special joint files. The files standardization is shown below:

**XX81**- physical file;

**XX01**- main logical file;

**XX02, XX03...**- secondary logical files;

**JJ...**- joint files.

Hence, the following physical files were created:

**TE81** – temperature;

**PH81** - pH;

**OX81** - oxygen;

**DE81** – air flow;

**TV81** – motor speed;

**SP81** - pressure;

**SB81** – substrate concentration;

**SU81** – cell concentration (dry weight);

**MU81** – specific growth rate,  $\mu$ ;

**PR81** – product concentration.

Each record has the following structure (fig. 4.6.1):

- *Experiment type*: specifies the experiment type: biomass growth, product formation, etc. ; format 10X;
- *Operation mode*: defines the operation mode, i.e. batch, fed-batch or continuous; format 5X;
- *Reactor type*: specifies the type of reactor: stirred speed reactor or air-lift reactor; format 5X;
- *Microorganism type*: names the manipulated microorganism (*Hansenula polymorpha*, *Candida*, etc.); format 10X;
- *Environment type*: specifies the culture medium type; format 10X;
- *Inoculum type*: specifies the manipulated inoculum quantity; format S999;
- *Experiment number*: format 9(4);
- *Date*: defines the date of experiment; format YYYYMMDD;
- *Time*: specifies the measurement instant; format HHMMSS (9(6));
- *Value*: holds the measured parameter value; format S9(4)V9(4).

Experiment	Operation Mode	Reactor Type	Microorg.	Medium	Inoculum	Experiment No.	Date	Time	Value
1	2	3	4	5	6	7	8	9	10

Fig. 4.6.1 Data record for yeast and fungus strain

Each physical file has various associated logical files:

**XX01** – main logical file (key structure: 1/2/3/4/5/6/7/8/9/);

**XX02** – logical file, key: 3/1/2/4/5/6/7/8/9/;

**XX04** - logical file, key: 5/1/2/3/4/6/7/8/9/

**XX05** - logical file, key: 4/2/3/5/6/7/8/9/;

**XX06** - logical file, key: 7/;

Moreover, the listed joint file were created for better information organization:

**JO01** – identical key with XX01;

**JO02** - identical key with XX02;

**JO03** - identical key with XX03;

**JO04** - identical key with XX04;

**JO05** - identical key with XX05;

**JO06** - identical key with XX06.

These joint files were constructed through the attachment of the following logical files: pH, oxygen, biomass, substrate and product concentration.

#### 4.6.3.2. Database for enzymatic hydrolysis bioprocess

In this case the following parameters are monitored: temperature, pH, stirred speed, specific reaction rate ( $v$ ), substrate and product concentrations. The corresponding physical and logical files are shown below:

**XX71** – physical file;

**XX11** – main logical file;

**XX12, XX13,...**, - secondary logical files;

**JO..**, - joint files.

Therefore, the following physical files were created:

**TE71** - temperature;

**PH71** - pH;

**TV71** – stirred speed;

**NU81** – specific reaction rate ( $v$ );

**SB71** – substrate concentration;

**PR71** – product concentration.

Each record has the following structure (fig. 4.6.2):

- *Operation mode*: defines the operation mode, i.e. static or stirred; format 8X;
- *Strain*: specifies the strain type (e.g. (*Aspergillus*, *Trichoderma*, etc.)); format 12X;
- *Substrate type*: consist of the type of substrate; format 14X;
- *Experiment number*: format 9(4);
- *Date*: defines the date of experiment; format YYYYMMDD;
- *Time*: specifies the measurement instant; format HHMMSS (9(6));
- *Value*: holds the measured parameter value; format S9(4)V9(4).

Operation Mode	Strain	Substrate	Experiment No.	Date	Time	Value
1	2	3	4	5	6	7

Fig. 4.6.2 Data record for enzymatic hydrolysis bioprocess

Each physical file has various associated logical files:

**XX11** – main logical file (key: 1/2/3/4/5/6/);

**XX12** – logical file (key: 2/1/3/4/5/6/);

**XX14** – logical file (key: 4/).

In this case the following joint files were created by attachment of substrate, product and specific reaction rate logical files:

**JO11** - identical key with XX11;

**JO12** - identical key with XX12;

**JO14** - identical key with XX14,

#### 4.6.3.3. Database for yeast growth on simulated liquor

Following the ideas presented in #3.1.6, the main parameters for yeast cultivation on simulated liquor bioprocess are temperature, pH, air flow, dissolved oxygen, inhibitor and substrate concentration, specific growth rate, cell concentration (dry weight). The associated physical and logical files are:

**XX61** – physical file;

**XX21** – main logical file;

**XX22, XX23...** – secondary logical files.

The following physical files were created:

**TE61** - temperature;

**PH61** - pH;

**DE61** – air flow;

**OX61** – dissolved oxygen;

**IN61** – inhibitor concentration;

**SB61** – substrate concentration;

**SU61** – cell concentration (dry weight);

**MU61** – specific growth rate.

Each record has the following structure (fig. 4.6.3):

- *Environment type*: specifies the culture medium type; format 10X;
- *Inoculum type*: specifies the manipulated inoculum quantity; format S999;
- *Experiment number*: format 9(4);
- *Date*: defines the date of experiment; format YYYYMMDD;
- *Time*: specifies the measurement instant; format HHMMSS (9(6));
- *Value*: holds the measured parameter value; format S9(4)V9(4).

Medium	Inoculum	Experiment No.	Date	Time	Value
1	2	3	4	5	6

Fig. 4.6.3 Data record for yeast growth on simulated liquor

Each physical file has associated the following logical files:

**XX21**- main logical file (key: 1/2/3/4/5/);

**XX22** – secondary logical file (key: 3/1/2/4/5/).

In this case no joint file is created.

This software, elaborated under PACBASE must supervise and edit (on operator request) the database elements. Hence, the operator can read (through AS400 system) the data files, in agreement with custom configurations or write different file information to printer. The direct data access (creation, modification, and

delete) is prohibit at this level. Only the system administrator can operate on the data structure.

## CONCLUSIONS

In this chapter, the component blocks of an intelligent control structure were designed. Three bioprocess types were taken into consideration: enzymatic hydrolysis of a lignocellulosic substrate (wheat straw); alcoholoxydase obtaining with the methylotrophic yeast *Hansenula polymorpha*; yeast cultivation on simulated liquor. Process class heterogeneity directs characteristic implementations; each case was treated separately, for all blocks.

Hence, the Filter block, identical for all cases, must “clean” input data, according to an imposed criterion.

The Neural Net for Parameter Estimation esteems the parameter values of the controlled bioprocess in agreement with *a priori* information transferred to expert system and the process evolution curve.

The Neural Net for Pattern Recognition must identify bioprocess type based on digitized standard curves, as part of a general process class, which was configured through *a priori* information.

The Fuzzy Control Structure controls bioprocess behavior, in agreement with specific rules established by the human expert.

The Expert System manages the general functionality of the control system. At this level, a specific information database was created compatible with process particularities.

Moreover, for each block of the intelligent control structure, the soft implementations and algorithm structures were specified

Finally, an AS400 database was built, in agreement with each bioprocess type.