Biosystems Engineering (2003) **84** (1), 13–30
doi:10.1016/S1537-5110(02)00232-5
AE—Automation and Emerging Technologies

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Fault Detection and Diagnosis in Deep-trough Hydroponics using Intelligent Computational Tools

K.P. Ferentinos; L.D. Albright

Department of Biological and Environmental Engineering, Cornell University, Ithaca, NY 14853, USA; e-mail of corresponding author: kpf3@cornell.edu

(Received 17 December 2001; accepted in revised form 25 September 2002)

The intelligent computational tools of feedforward neural networks and genetic algorithms are used to develop a real-time detection and diagnosis system of specific mechanical, sensor and plant (biological) failures in a deep-trough hydroponic system. The capabilities of the system are explored and validated. In the process of designing the fault detection neural network model, a new technique for neural network designing and training parameterisation is developed, based on the heuristic optimisation method of genetic algorithms. Sensor and actuator faults are detected and diagnosed in sufficient time that the fault detection model can be applied on-line as a reliable supervisor of the operation of an unattended deep-trough hydroponic system. Biological faults were not detected in general. It seems that the interaction between plants and their root-zone microenvironment is not equally balanced, as the condition of the plants is highly influenced by the conditions in their root zone microenvironment, while these microenvironment conditions (as they are represented by the measurable variables) are not influenced in the same degree by the conditions of the plants. Finally, the genetic algorithm system developed here can be successfully applied to a combinatorial problem such as deciding the best neural network architecture, activation functions and training algorithm for a specific model.

© 2003 Silsoe Research Institute. All rights reserved
Published by Elsevier Science Ltd

1. Introduction

Greenhouse engineering and hydroponics are two very rapidly developing sectors of agriculture and are strongly linked with each other. Computational intelligence, mainly in the form of automatic monitoring and control, is a major tool of this development. Highly developed instrumentation and 'intelligent' control in hydroponics provides an opportunity for maximising both quality and quantity of production through the advanced management of all involved processes. The production systems are continuously monitored and precisely controlled. An important issue in these highly computerised and automated systems is the quality of information provided by the sensors, as well as the quality of decisions passed to the actuators. The quality of information received from or passed to the system is not checked in the vast majority of automated greenhouse or hydroponic facilities.

Artificial intelligence (AI) has developed to such a degree that it can lead to an intelligent control system capable of self-examination. The combined information from different sensors, through AI methodologies such as neural networks (NNs), can lead to quality classification of isolated information derived from specific sensors or actuators. In this way, a fault detection and diagnosis system could eventually be developed, capable of detecting and identifying specific failures in parts (sensors or actuators) of the hydroponic system, by simply reading the collected measurements of the system sensors and actuators.

As an extension of the 'speaking plant' approach (Hashimoto, 1989), the hydroponic system, which consists of mechanical and electronic equipment and also of cultivated plants, could be considered as one hybrid (bio-mechatronic) system. In this extent, biological failures, in the sense of stressed situations of the plants, could possibly be detected by a fault detection (FD) system, simply using the readings of the above-mentioned measurements.

In this work, an NN-based fault detection and diagnosis system is developed (Ferentinos, 2002). The main area of concentration is deep-trough hydroponic systems. The motivation was the fact that detection of

 © 2003 Silsoe Research Institute. All rights reserved
Published by Elsevier Science Ltd

stressed plants is very important for the final production. This detection can be made via measurements on the environment of the plants. This becomes easier in hydroponics, for several environmental variables of the plants, and particularly root zone dynamics, can readily be monitored. Another direction of this work was towards the development of a system for design and training parameterisation of NNs that would be more sophisticated than the trial-and-error methodology usually used for these tasks. The system was based on the evolutionary optimisation characteristics of genetic algorithms (GAs), a heuristic optimisation method inspired by Darwinian evolution. This automated process can overcome difficulties embedded in the human intervention of the trial-and-error approach, such as trapping in local minima or poor exploration of the search space.

NNs have been proved capable of identifying faults in several complex biological processes in which neither analytical models nor intermediate residual calculations were used. Watanabe *et al.* (1989) used a two-stage multi-layer NN to detect five different types of faults in a chemical reactor. The results showed that use of a two-stage NN could diagnose incipient faults from only three noisy process measurements, which is very encouraging considering the complexity of the examined process; thus, it was concluded that NNs have a great potential for FD in chemical plants.

Venkatasubramanian and Chan (1989) examined the detection of faults in a catalytic cracking unit using NNs. In particular, their methodology performed multiple fault diagnosis while being trained on knowledge of single faults. It was concluded that NNs, with their ability to learn from example and extract salient features from data and tolerate noisy and random data, were ideal for fault detection and diagnosis in processes in chemical engineering. Sorsa *et al.* (1991) reported on three different types of NNs used to detect faults on a simulated process consisting of a heat exchanger and a tank reactor. The conclusion was that the multi-layer perceptron network was best, and use of the hyperbolic tangent as the non-linear element improved the training rate of the whole network. Of course, it must be understood that the whole work was based on a simulation model and the network was trained with data coming from a mathematical model. Experiments with real processes have to be made in those networks.

Several other works such as, for example, Parlos *et al.* (1994), Hoskins *et al.* (1991), Chow and Yee (1991) and Xiaoming *et al.* (1997), showed similar results in applying NNs for fault detection and diagnosis in chemical plants. However, in all these works, simulation data were used for the training of the networks. This is dangerous and may lead to incorrect diagnoses, so further stabilisation of these systems should be done by training with real data.

2. Materials and methods

2.1. *Experimental arrangement*

The experiments were conducted in a section of the Kenneth Post Laboratory (KPL) Greenhouses at Cornell University, in Ithaca, NY, USA. The greenhouse section had a floor area of about $85\,m^2$ and it was fully equipped with a staged ventilation system, an evaporative cooling system, a lighting system and a movable shading system. The controlled environmental variables were the daily integral of light [photosynthetically active radiation (PAR)] and the air temperature. Light intensity, air temperature, relative humidity and CO_2 concentration were continuously monitored. During the experiments, temperature setpoints were 24°C during the day and 19°C during the night and were achieved within $\pm 0{\cdot}5$°C. The daily PAR integral setpoint was $17\,mol\,m^{-2}$ and was achieved by using supplemental lighting from 21 high-pressure sodium, 400 W, lamps during winter and the shading screen during summer. The relative humidity was maintained between 30 and 70%.

The cultivation system was a deep trough hydroponic system that consisted of three small growing ponds (tanks). The ponds of this system were filled with nutrient solution (Sonneveld & Straver, 1994) and the plants were placed in specially placed holes (*Fig. 1*) on Styrofoam panels that covered the ponds and floated on the surface. Each stainless steel pond had an area of $0{\cdot}75\,m^2$ (1·25 m by 0·60 m) and root zone control was completely independent of the others. In that way, the systems could be monitored and controlled in parallel and, thus, more data sets could be constructed. Lettuce (*Lactuca sativa*, var. Vivaldi) was chosen as the cultivated plant in these experiments. Seedlings were transplanted from a growth chamber into the ponds 12 days after sowing. Foam spacers of 2 cm width were used to provide sufficient room for the plants as they grew larger and were added incrementally, as shown in *Fig. 1*. As can be seen in the same figure, each pond had a range of plant ages ranging from 12 days old, which was the transplanting age of the plants from the germination chambers to the ponds, to 27 days old. Every 2 days, the largest plants of the hydroponic system (of 28 days of age) were harvested, the rest of the plants were moved upwards (see layout of *Fig. 1*), 12-day old plants (of dry weight around $0{\cdot}04\,g\,plant^{-1}$) were transplanted from the growth chamber into the ponds and new seeds were sown in the growth chamber.

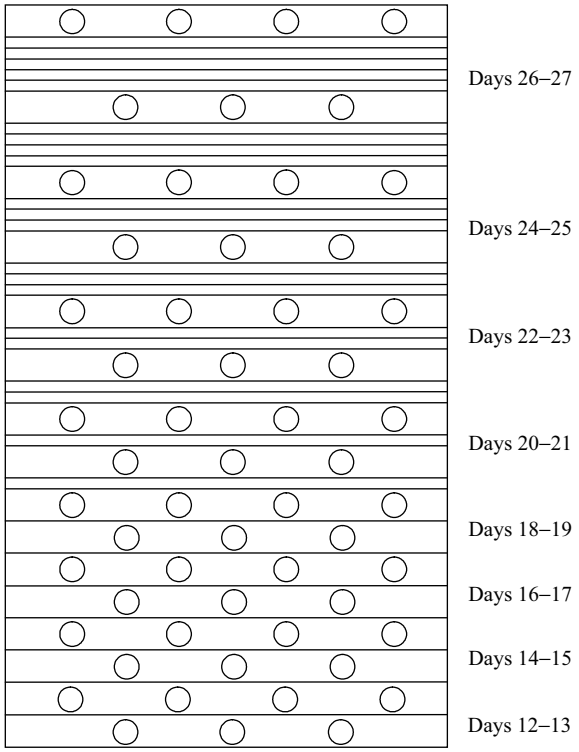

Fig. 1. Plant spacing and age distribution on each pond of the hydroponic system: horizontal lines represent 2 cm wide spacers

In this way, a 'continuous production' type of system was developed, which resembles real-life hydroponic production systems more closely than other techniques that have been used in NN modelling applications (*e.g.* Ferentinos, 1999) and, in addition, it produces nearly constant conditions.

The monitoring and control system consisted of a personal computer (PC) running LabVIEW software (available from National Instruments), a data acquisition board connected to the computer and several meters, sensors and actuators connected to the board. The monitored parameters were the pH, the electrical conductivity (EC), the temperature, the dissolved oxygen (DO), the weights of two of the three ponds (for measuring evapotranspiration) and the control signals of pH and DO. For each pond, there was a metering pump used to control the pH of the nutrient solution by adding acid or base, and solenoid valves that controlled the DO by adding oxygen. The program monitored and controlled the system every 10 s and logged its data every 5 min. The control methodology was based on the pseudo-derivative feedback (PDF) control algorithm (Phelan, 1977).

The experiments for collecting training and testing data sets were performed between November 2000 and May 2001. In addition to the monitored variables mentioned before, environmental variables such as air temperature, relative humidity and light (PAR) intensity at the level of the plants were also logged. The pH setpoint was 5·8 and the DO was maintained between 6·5 and 7 mg l^{-1}. The EC was not controlled automatically but its values were kept between the recommended setpoints of 115–125 mS m^{-1} by adjusting manually every 2 days by adding reverse-osmosis water to replace evapotranspiration and solution stocks to maintain the EC.

2.2. *Fault types and neural network model*

The NN approach to this FD application was chosen mainly because of the specific nature of hydroponics, but also for its simplicity. Hydroponic systems are highly non-linear because of the non-linearity of the biological processes involved. Thus, estimation methods for FD would not be suitable, mainly because of their high computational demands in the cases of non-linear systems. In addition, precise analytical models of the hydroponic system do not exist; thus, estimation methods become unacceptable. From the pattern recognition methods for FD, NNs were considered to be the most appropriate because it was decided that the autonomy that AI gives to this approach is preferable to some arbitrary selection of complicated mathematical tools that other pattern recognition techniques would require.

The procedure of training the NN model requires, first, an accurate definition of 'normal operation', defined in our case as unstressed plants in a system which is in control. There is no need to express this normal situation by means of specific values of the environmental variables because, as mentioned before, NNs do not need such a representation in order to learn the pattern. It is necessary to know only when the system and, in extension, the plants are in conditions considered to be normal by the producers, and also to know which training data sets correspond to those normal conditions. It is also required to define the 'faulty operation' and categorise this kind of operation into different types of faulty operations, one for each different kind of fault. In order to obtain data sets for each kind of fault, one has to impose those faults and take the corresponding measurements of the micro environment variables.

When the environmental and nutrient solution variables are within their limits and the plants appear healthy, the growth rate is optimised and operation of the system is considered 'normal'. The 'faulty operation' consisted of three different kinds of faults.

(1) *Actuator/mechanical faults*. These are failures in some actuator or some mechanical part of the hydro-

ponic system. The actuator fault considered was the malfunction of the pH control pump and the mechanical fault was the malfunction of the nutrient solution circulation pump.

(2) *Sensor faults*. These are failures in the sensors of the system. The ones considered were (a) pH sensor failure and (b) EC sensor failure.

(3) *Plant faults*. These are problems in the cultivated plants themselves and are divided into (a) root area faults and (b) shoot area faults.

For the first two categories of faulty operations, real data existed because sensor and actuator failures were encountered during daily operation of the system. In addition, several faults were especially imposed in order to train the NN model and investigate its inherent FD capabilities. For the third category however, faults were imposed directly on the plants.

The first experiments dealing with *plant faults* (or *biological faults*) consisted of imposing several different faults on the plants in order to examine their effect on the monitored variables of the nutrient solution of the system. Four different series of experiments were performed. In the first, most of the largest plants were removed from the pond for 5 min. In this way, possible problems in the root zone of the plants were imitated. In the second series, several leaves of the largest plants were removed. This action imposed permanent damage on the plants and imitated the effects of major problems in the shoot zone of the plants. A similar but less influential series of experiments was the one in which leaves of the plants were disturbed for intervals of 5 min and slightly damaged. These experiments imitated shoot problems less important than the ones imitated in the previous series of experiments. Finally, in the fourth series, the largest plants were covered with transparent plastic bags. This imposed a temporary fault that imitated major problems in the shoot zone, as transpiration was drastically reduced.

Effects of plant faults, unfortunately, in most cases, were not significant enough to be used in an FD scheme. The pH and the electrical conductivity appeared not to be affected by the faults. The transpiration, a variable known to be drastically affected by the condition of the plants, was so highly affected by the environmental conditions of the greenhouse (temperature, light intensity and relative humidity) that effects of plant damage, even when seemingly severe, were not noticeable. In the experiments in which some leaves of the plants were cut, where one would expect major impacts on the transpiration rate, the effects were 'hidden' by the high correlation of the transpiration with the environmental parameters, especially temperature and light intensity, probably because transpiration was occurring through the cut surfaces. The only fault that showed some correlation with transpiration, but not with any of the other variables, was the one where the largest plants were covered with plastic transparent bags. These results led to the decision to exclude the plant faults from the main fault detection NN (FDNN) model. A separate model was created for these kinds of faults and it is presented in Ferentinos *et al.* (2002), together with some results on the influences of biological faults in the measured variables of the hydroponic system.

The feedforward methodology of NNs (Fine, 1999) was used for the development of the detection and diagnosis system. The inputs of the NN were aerial environment parameters (air temperature T_{air}; relative humidity H_R; and light intensity L), the measurable variables of the root zone of the plant (pH, EC, DO and nutrient solution temperature T_s) and the control signals of the pH (U_{pH}) and the DO (U_{DO}) control (amounts of acid and oxygen added, respectively). Each output of the NN corresponded to a specific fault and one output corresponded to normal operation.

Thus, the NN model was formed to have five outputs: one for normal operation, two for actuator/mechanical faults and two for sensor faults (*Fig. 2*). In addition to the network inputs listed above, one- and two-step histories of the solution acidity pH, electrical conductivity σ and dissolved oxygen O_d variables were included. That is, for each of these variables, three inputs existed: one for time t (current time), one for time $t-1$ (previous time step) and one for time $t-2$ (two time steps before). Therefore, the network had 15 inputs (*Fig. 2*). The time step was 10 min. Fault types are listed in Table 1.

The values of the outputs used for training the NN were binary. An output of unity for a specific output denoted the existence of the corresponding fault type (or of normal operation when referring to the first output). The values of the other outputs were zero. The outputs during testing of the model were, of course, continuous, with values from 0 to 1. In this way, a decision process was formed to determine above which value an output should be meaningful in terms of the existence of a fault.

The basic training methodology used here was the Backpropagation Training Algorithm (Rumelhart *et al.*, 1986). This algorithm has several modifications according to the multi-dimensional minimisation algorithm that it uses to minimise the error estimator. Four different types of minimisation algorithms were considered: steepest descent, quasi-Newton, conjugate gradient and Levenberg–Marquardt algorithm. Their main difference is the way of approximating the inverse of the Hessian matrix. The steepest descent and the conjugate gradient algorithms replace the inverse of the Hessian with the identity matrix, while the other two algorithms try to approximate it with different methods

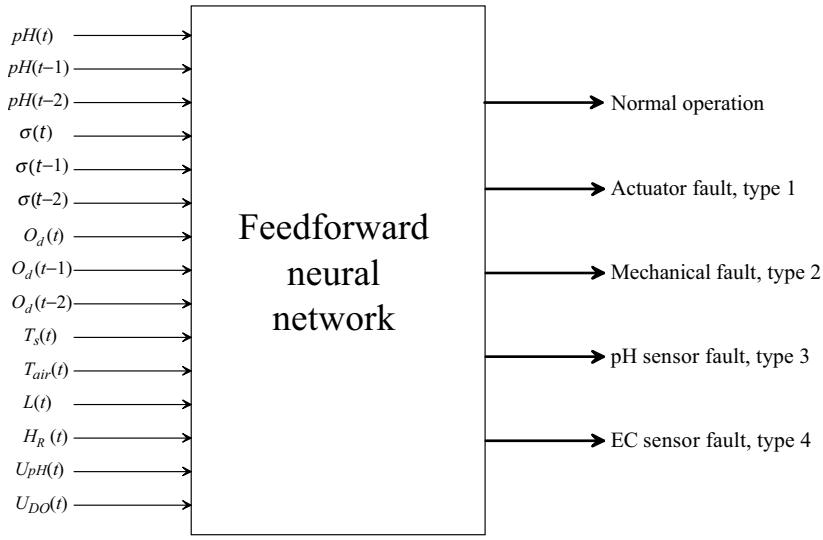

Fig. 2. Inputs and outputs of the fault detection neural network model: pH, σ, and O_d, variables for solution acidity, electrical conductivity and dissolved oxygen at time t and for the previous two time steps (t−1) and (t−2); $T_s(t)$, temperature of nutrient solution at time t; $T_{air}(t)$, air temperature at time t; L, light intensity at time t; $H_R(t)$, relative humidity at time t; $U_{pH}(t)$, control signal for the pH at time t; $U_{DO}(t)$, control signal for the dissolved oxygen (DO) at time t

Table 1
Definitions of faults; EC, electrical conductivity

Fault type	*Fault description*
1	pH control pump out of order
2	Circulation pump out of order
3	Failure in pH sensor
4	Failure in EC sensor

(Fine, 1999). A crucial point is the choice of the *learning rate* α, which must be such to minimise the error of the next iteration (epoch). In the large majority of works in the literature, a fixed-value learning rate is used during the training of the NN. Here, an on-line adjustable learning rate was used, as it seemed to perform better and, in addition, it has been proven successful in prior similar applications in hydroponic systems (Ferentinos, 1999). In the steepest descent algorithm, the Hessian was used to solve for the 'best' learning rate at each iteration. This is considered to be computationally prohibitive, but this algorithm turned out to be much faster than the other three, even when the learning rate was calculated by this way. For the other algorithms, the 'best' learning rate was calculated with an approximate line search using a cubic interpolation.

Both inputs and outputs (target values) were standardised to have mean zero and standard deviation one. The initial conditions, that is, the initial weights and biases of the NN, were uniformly generated (randomly) in the range $(-1/\sqrt{d}, 1/\sqrt{d})$ (Duda *et al.*, 2001), where d is the number of network inputs (in this case 15), and several experiments with different initial weights and biases were performed in order to take the best possible results. The two most common techniques of controlling the complexity of large NNs, validation and regularisation, were used.

2.3. *Genetic algorithm system*

The application of NNs in modelling non-linear processes has a central drawback: the lack of a precise method to choose the most appropriate network topology, type of activation functions and parameters of the training algorithm. These tasks are usually based on a 'trial-and-error' procedure performed by the developer of the model. In that way, optimality or even near-optimality is not guaranteed, as the explored space is just a small portion of the whole search space and the type of search is random. To overcome the problems associated with human network design and training parameterisation, an automated method, based on the evolutionary properties of the GAs, was developed. GAs evolve several network designs with different activation functions and several minimisation algorithms so that the best possible combination is finally chosen.

In the vast majority of relevant works in the literature, the minimisation algorithm of the backpropagation learning process is not considered in the encoding of the GA system (Harp *et al.*, 1989; Miller *et al.*, 1989; Kitano, 1990; Filho & de Carvalho, 1997; Iyoda & von

Zuben, 1999). With the exception of the work of Iyoda and von Zuben (1999), the types of activation functions of the NN are not considered either. In addition, some possible *a priori* knowledge of the system characteristics and possible general intuitions about the expected topology of the NN were not taken into account. Such *a priori* knowledge can drastically limit the huge search space of the problem of NN design, and more dimensions of the problem, like the minimisation algorithm or the types of activation functions, can also be encoded into the GA without making the encoding extremely complex and difficult to be optimised.

The main aspect in an evolutionary system such as the one developed here is the way of encoding the several possible phenotypes of the NN into specific genotypes. A phenotype, in our case, consists of the NN topology, its activation functions in the hidden and the output nodes and, in addition, the minimisation algorithm that is used by the backpropagation algorithm during training. A genotype is a sequence of bits (0 or 1) with a specific constant length. Each genotype corresponds to a unique phenotype.

The representations that encode network phenotypes into string genotypes are generally divided into two categories: the direct or strong specification or low-level representations (Miller *et al.*, 1989) and the indirect or weak specification or high-level representations (Harp *et al.*, 1989). The representations of the former category encode explicitly every network connection from node to node in a way that someone, by just looking at a bit string, can encode the corresponding network topology. A bit of 1 corresponds to a connection while a bit of 0 corresponds to lack of connection. This type of representation requires very large binary strings and can be problematic with all the restrictions that feedforward NNs comprise (*i.e.* no cycles, no connections between nodes of non-successive layers or between nodes of the same layer, *etc.*). The representation of the weak specification category uses specific correspondences of specific binary strings to specific network architectures that are pre-defined by the user. This is the point where the *a priori* knowledge of the specific application problem to be considered is taken into account so that the infinite search space is reduced drastically.

2.3.1. *Bit-string representation*

The weak specification representation was used here. The developed scheme incorporates three tasks of the NN design and training parameterisation:

(i) the selection of the minimisation algorithm used by the backpropagation training algorithm;
(ii) the architecture of the NN;
(iii) the types of the activation functions of the hidden nodes and of the output nodes.

Representation of the minimisation algorithm used by the backpropagation algorithm. There are four different multi-dimensional minimization algorithms considered by the GA: steepest descent, quasi-Newton, Levenberg-Marquardt and conjugate gradient algorithms. These four algorithms can be represented with two binary entries, as follows: 00, steepest descent; 01, quasi-Newton; 10, Levenberg–Marquardt; 11, conjugate gradient; and they form the first two bits of the binary string representation (*Fig. 3*).

Representation of the network architecture. The next six binary entries of the string (*Fig. 3*) represent 64 possible network architectures of one-hidden layer (1-HL) networks with two to 30 nodes and two-hidden-layer (2-HL) networks of several combinations of nodes in each layer.

Representation of the activation functions. Two activation functions were considered for the hidden-nodes of the NN: the logistic function and the hyperbolic tangent function. In addition, the output nodes were allowed to have either the same activation function as the hidden nodes, or a linear summation function. Thus, four different combinations were included in the genetic

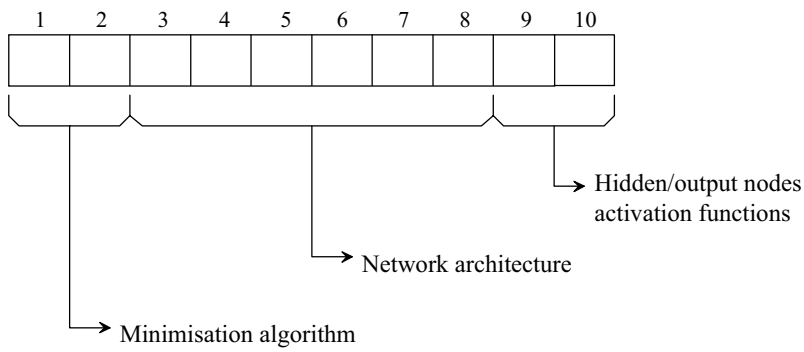

Fig. 3. Binary representation of neural network topology and training parameterisation

Table 2
Binary encoding of activation function combinations for hidden and output nodes of the neural network

Bit sequence	*Hidden nodes activation function*	*Output nodes activation function*
00	Logistic	Logistic
01	Logistic	Linear summation
10	tanh	tanh
11	tanh	Linear summation

representation and were encoded with two final binary entries in the bit string, as shown in Table 2.

Thus, each binary string (genotype) of an individual of the GA that represents a specific phenotype of NN architecture and training parameterisation, had a total of 10 bits, as shown in *Fig. 3*. A fitness function was applied to the GA system, in order to measure the performance of each individual. The fitness of each string was simply a large number minus the mean squared error (MSE) after the training of the specific network architecture with the corresponding activation functions, by the corresponding training algorithm. The large number was added to the function to assure that the final fitness was always a positive number. Each string was assigned a probability of reproduction (Fogel, 1995) and was selected according to that probability, which was proportional to the fitness of each string. All strings of the population were then subject to the evolutionary operations of crossover and mutation (Goldberg, 1989), after which the final population was formed. This process was repeated until the maximum number of generations was reached.

3. Results

The training data set consisted of a matrix with columns that represented the average measurements of 10 min periods and were of the form:

$$[pH(t)pH(t-1)pH(t-2)\sigma(t)\sigma(t-1)\sigma(t-2)\ldots$$
$$O_d(t)O_d(t-1)O_d(t-2)T_sT_{air}LH_RU_{pH}\mathrm{U}_{DO}]^{\mathrm{T}}$$

where: t is a specific time, $t-1$ is the previous time step (i.e. 10 min before), $t-2$ is two steps before, pH is the solution acidity, σ is the electrical conductivity, O_d is the dissolved oxygen, T_s is the temperature of the nutrient solution, T_{air} is the temperature of the air inside the greenhouse, H_R is the relative humidity inside the greenhouse, U_{pH} is the control signal for the pH and U_{DO} is the control signal for the DO, and T is the transpose operator. This training set was fed into the GA system for NN synthesis, that is, optimal design and training parameterisation for the specific process that had to be learned. Use of GAs is a probabilistic optimisation technique. Therefore, the entire optimisation process must be repeated several times, starting from different random initial populations each time. The basic parameters of GAs that must be explored are the population size M, the probability of crossover P_c, the probability of mutation P_m and the number of generations G. A number of experiments had to be carried out to determine the best possible parameters. The type of crossover used was the most common one, the one-point crossover (Goldberg, 1989). The first experimentation dealt with the determination of the best values of P_c and P_m. These runs of the system were made with a population size of 20 strings and for 30 generations. The best performance was achieved with a value for P_c of 0·9 and for P_m of 0·05. After the best solution was found by the optimal (concerning its parameters) GA system, that solution, i.e. best network architecture, training algorithm and activation functions combination, was further trained. Finally, the value of the FDNN model was evaluated on the performance achieved in new, testing data sets, with samples that contained specific faults or normal data.

Thus, the final GA system had a population size of 20 individuals and evolved for 30 generations with a probability of crossover between individuals equal to 0·9 and probability of mutation in each bit of the individuals equal to 0·05. *Figure 4* shows the best MSEs found after each generation during the best run of the GA system. In this run, the best solution was found after 27 generations. In *Fig. 5*, the corresponding average MSEs of the entire population after each generation are shown. One can see that the average performance of the population generally improves, because of the evolutionary selection process of the algorithm.

The best solution found was the string [0 0 0 1 1 0 1 0 1 1], which is interpreted as a 1-HL NN with 28 nodes in the hidden layer, hyperbolic tangent activation functions in hidden nodes and linear summation functions in

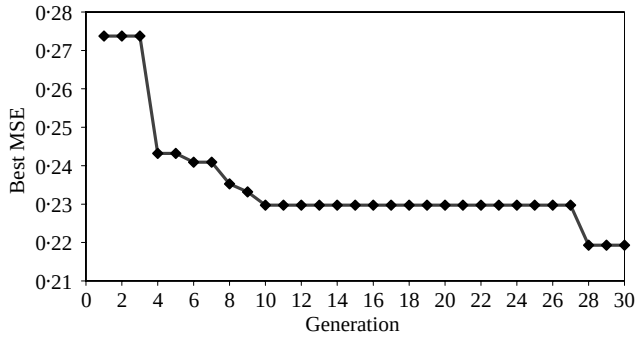

Fig. 4. Best mean squared error (MSE) found during the best genetic algorithm run

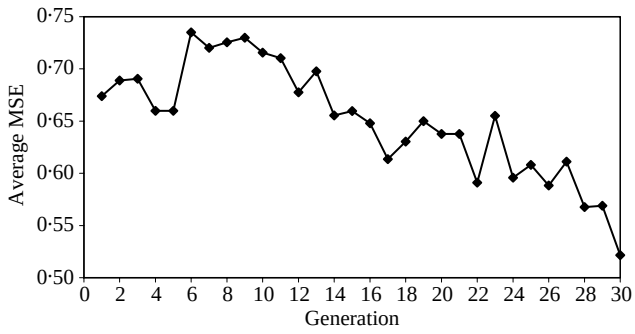

Fig. 5. Average mean squared errors MSEs of the entire population during the best genetic algorithm run

output nodes, trained with the steepest descent back-propagation algorithm. This solution gave a value for the MSE of 0·2193. It should be mentioned here that algorithms were trained for up to 50 iterations during the optimisation process of the GA system. The second and third best solutions, with slightly worse performances, were 1-HL networks with 22 and 18 nodes, respectively. They were each trained with the steepest descent algorithm and each had hyperbolic tangent activation functions in hidden nodes and linear summation functions in output nodes. The values for their MSEs were 0·2220 and 0·2297, respectively. From these results, it seems that generally the steepest descent algorithm was the most appropriate algorithm for the specific application and, in addition, hyperbolic tangent activation functions in hidden nodes and linear summation in output nodes had better performance than the other combinations of activation functions. Also, 1-HL architectures seemed to outperform in general the 2-HL ones.

Thus, the final NN consisted of 15 inputs, one hidden layer with 28 nodes and five outputs. It contained hyperbolic tangent activation functions in hidden nodes and linear summations in output nodes and was trained with the steepest descent algorithm. Further training was performed in this NN, with several random initial values of weights and thresholds and a variety of algorithm parameters. The MSE of the final NN was 0·1148 after 1000 iterations.

3.1. *Fault detection and diagnosis testing results*

The testing process on the FDNN consisted of presenting new data sets to the network, namely the testing sets, each of which contained some specific fault imposed at the moment that the set started, and exploring its performance. In addition, testing sets that contained only normal data were included in the testing process, to investigate the ability of the network to avoid false alarms. The testing tests were distributed throughout the entire period of data collection. Twelve data sets were constructed. The first three contained fault type 1 (pH control pump failure), the next six contained fault types 2 (circulation pump failure), 3 (pH sensor failure) and 4 (EC sensor failure), respectively, in pairs, and the last three data sets represented normal operation and differed from each other by periods of 1 month.

The decision support approach that was used to decide whether a fault has been indicated or not, in order to initially evaluate the performance of the NN model, was the following: output values above 0·6 indicate a fault, values below 0·4 indicate normal operation and values in the interval [0·4, 0·6] indicate the previously known condition of the output.

The most important issue in a fault detection and diagnosis application is how fast the faults are detected and diagnosed. After the first detection of some fault and after that detection has been supported by a series of 'faulty' indications of the model for some period of time, if the NN outputs return to 'normal operation', it is not crucial, as we know that faults are not reversible without human intervention. Thus, the most important factor in this FD system is its *on-line* performance and more specifically, the rapidness of detection of faults.

3.1.1. *Classification of results*

Table 3 shows the percentages of the testing sets of each fault type that were classified correctly (that is, the

Table 3
Percentages of testing sets of each fault that were correctly classified after specific time periods from the initiations of the faults; EC, electrical conductivity

Fault type	*Correct classification of fault, %*								
	Test period, h								
	0·17	*0·33*	*0·5*	*1*	*1·5*	*3·5*	*6*	*8*	*13*
Actuator fault, type 1	0	0	0	0	0	0	33	66	100
Mechanical fault, type 2	50	100							
pH sensor fault, type 3	0	0	0	0	50	100			
EC sensor fault, type 4	0	50	100						

corresponding fault was detected) according to the time of detection after each fault was imposed. For example, within 1·5 h, 100% of the testing sets containing faults 2 and 4 were correctly classified (*i.e.* their faults were detected), 50% of the testing sets containing fault 3 were correctly classified and none of the testing sets containing fault 1 was correctly classified. Fault types 2 and 4 were the most rapidly detected ones. Fault type 2 (circulation pump failure) was detected in average after 10–20 min and fault type 4 (EC sensor failure) was detected after 20–30 min. The detection of fault type 3 (pH sensor failure) took much longer; this fault was detected in 1·5–3·5 h. Finally, fault type 1 (pH control pump failure) was the slowest detected fault, as it was detected on average more than 9 h after the fault occurred. However, it should be noted here that this specific fault is situation-dependent because the pH control pump does not operate continuously. Thus, if in specific situations the pump does not operate, the fault cannot be detected. When the pump should have started to operate, the fault was usually detected within 30 min.

Another important issue in a fault detection and diagnosis system is the probability of false alarms. This leads to the evaluation of the *off-line* performance of the system. First, the following terms should be defined:

False alarm probability P_{FA} is the probability of detecting some fault while the system is in normal operation.

Misclassification probability P_{MC} is the probability of diagnosing a specific fault while the actual fault is a different one or diagnosing normal operation when a fault exists.

Correct classification probability P_{CC} is the probability of diagnosing a specific fault while this fault indeed exists or detect normal operation when indeed no fault exists in the system.

It is obvious that, if the probabilities are expressed as percentages, in situations of normal operation,

$$P_{CC} = 100 - P_{FA} \tag{1}$$

while, in situations of 'faulty' operation,

$$P_{CC} = 100 - P_{MC} \tag{2}$$

In the evaluation of the on-line performance, percentages of correctly classified testing sets were used because the classification of each sample of data was not the important factor. However, in off-line performance (i.e. the probabilities of false alarms and misclassifications), the percentages of correctly or incorrectly classified samples of data will need to be used. A data sample, or simply a sample, is defined as a set with the 15 inputs to the NN values, measured at a specific time interval. In the case of P_{FA}, the percentages concern the testing sets of normal operation. In the case of P_{MC}, they concern the testing sets that contain some fault. The important characteristic of this specific FD system is the false alarm probabilities.

Table 4
Classification percentages of data samples of normal operation; EC, electrical conductivity

Output class	*Actual data: 'normal', %*
Normal	99·2
Actuator fault, type 1	0·2
Mechanical fault, type 2	0·4
pH sensor fault, type 3	0·2
EC sensor fault, type 4	0

Table 4 shows the percentages of classification of the samples of the three 'normal' testing sets. The sum of the percentages that indicate the existence of some fault, represents the actual P_{FA} of the system. Thus, a value for P_{FA} of 0·8% is a very low probability. However, the situation where the FD returned neither normal nor faulty operation, was not included in the estimation of P_{FA}, because in that specific situation the FD system prompts for a possibility of 'unknown fault' (see Section 4); thus, according to the formal definition of the term 'false alarm', this situation cannot be included. The overall probability of 'unknown fault' indication during testing in 'normal' data sets was 14·3%. This means that the P_{CC} of normal operation was 99·2 – 14·2 = 84·9%. However, as it is not known exactly what had happened during the collection of the third of the three 'normal' testing sets, where all the 'unknown fault' indications occurred, it should be concluded that the actual value for P_{CC} of normal operation lies between 85 and 99%.

Table 5 shows the classification probabilities of all faulty data samples into the four fault types or the 'unknown fault' case. The grey boxes represent P_{CC} values. The percentages in the last column represent the probabilities of 'unknown fault' classification. All other entries represent the P_{MC} values. Some rare cases when more than one fault were indicated by the system, were included in the 'unknown fault' category.

With the exception of fault 1, all P_{CC} values are very low. In addition, as explained earlier, misclassification probabilities are not important in most cases because of the irreversible nature of the faults. Only the misclassifications that occurred before the correct classification of the actual fault are important and should be considered as true P_{MC}. Therefore, the actual probabilities of most misclassifications are smaller than those presented in Table 5, which makes the actual P_{CC} values larger.

Table 5
Classification percentages of data samples of faulty operations

Tested data set of type	*Classification, %*					
	Normal	*Fault 1*	*Fault 2*	*Fault 3*	*Fault 4*	*Unknown fault*
Fault 1	25·5	70·1	0·2	0	4·2	0
Fault 2	1·9	0	92·4	0	3·8	1·9
Fault 3	0	0	1·5	92·1	3·9	2·5
Fault 4	1·8	0	1·7	2·4	92·9	1·2

Fault 1, actuator; fault 2, mechanical; fault 3, pH sensor; fault 4, electrical conductivity.

3.1.2. *Fault diagnosis testing results*

Figures 6–10 show some examples of the NN model outputs during five testing data sets, one for each type of fault and one set that contains only normal data. The time step is 10 min. It should be noted here that each fault in these testing sets starts at the beginning of the tests. This means that the first interval shown in each plot represents the outputs of the network for measurements taken 10 min after the initiation of the fault. In *Fig. 6* the outputs of the network during the existence of fault type 1 (pH control pump failure) are shown. In this specific data set, the fault was detected in about 8 h after

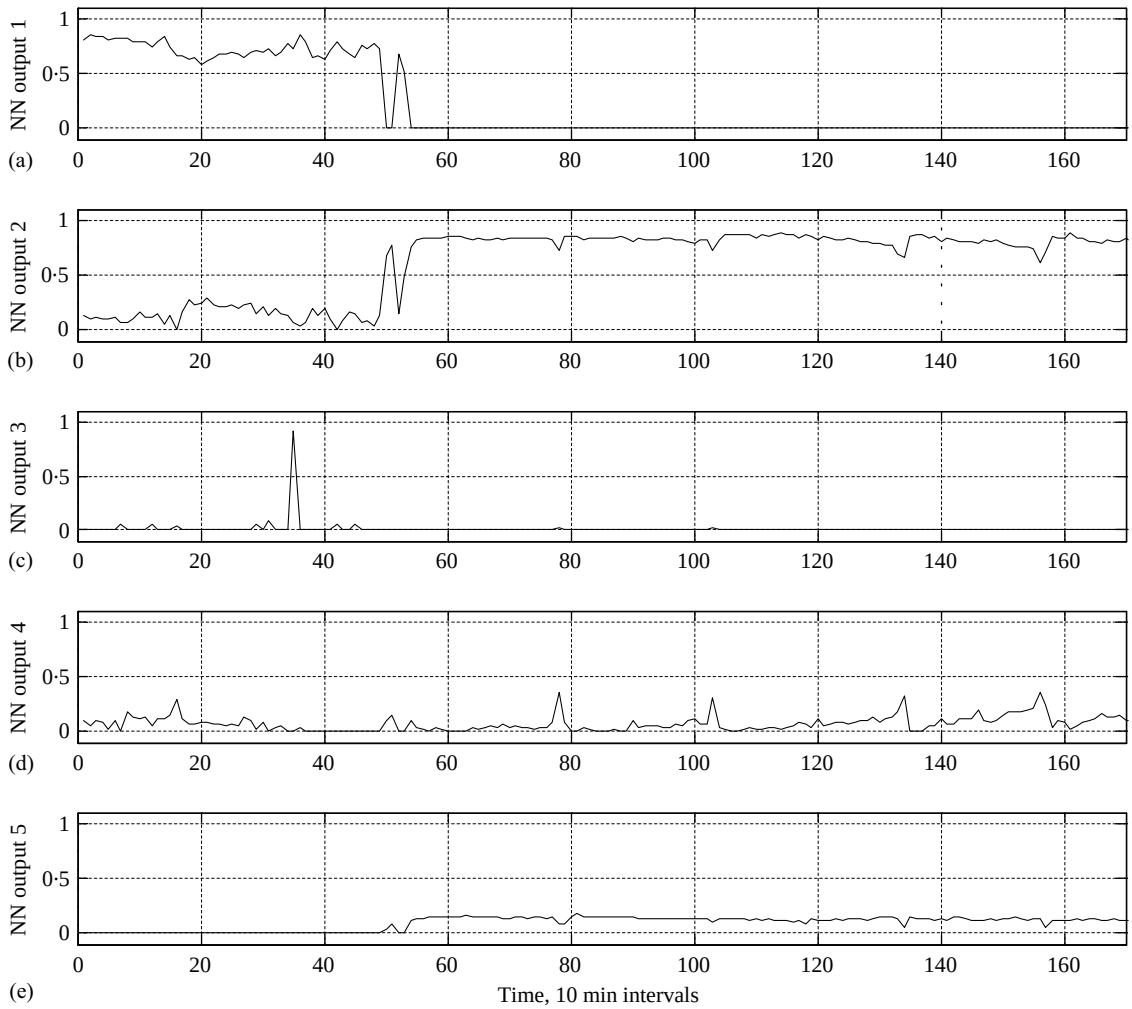

Fig. 6. Neural network (NN) outputs during a 'fault type 1' testing data set: (a) normal output; (b) actuator fault, type 1 output; (c) mechanical fault, type 2 output; (d) pH sensor fault, type 3 output; (e) electrical conductivity (EC) sensor fault, type 4 output

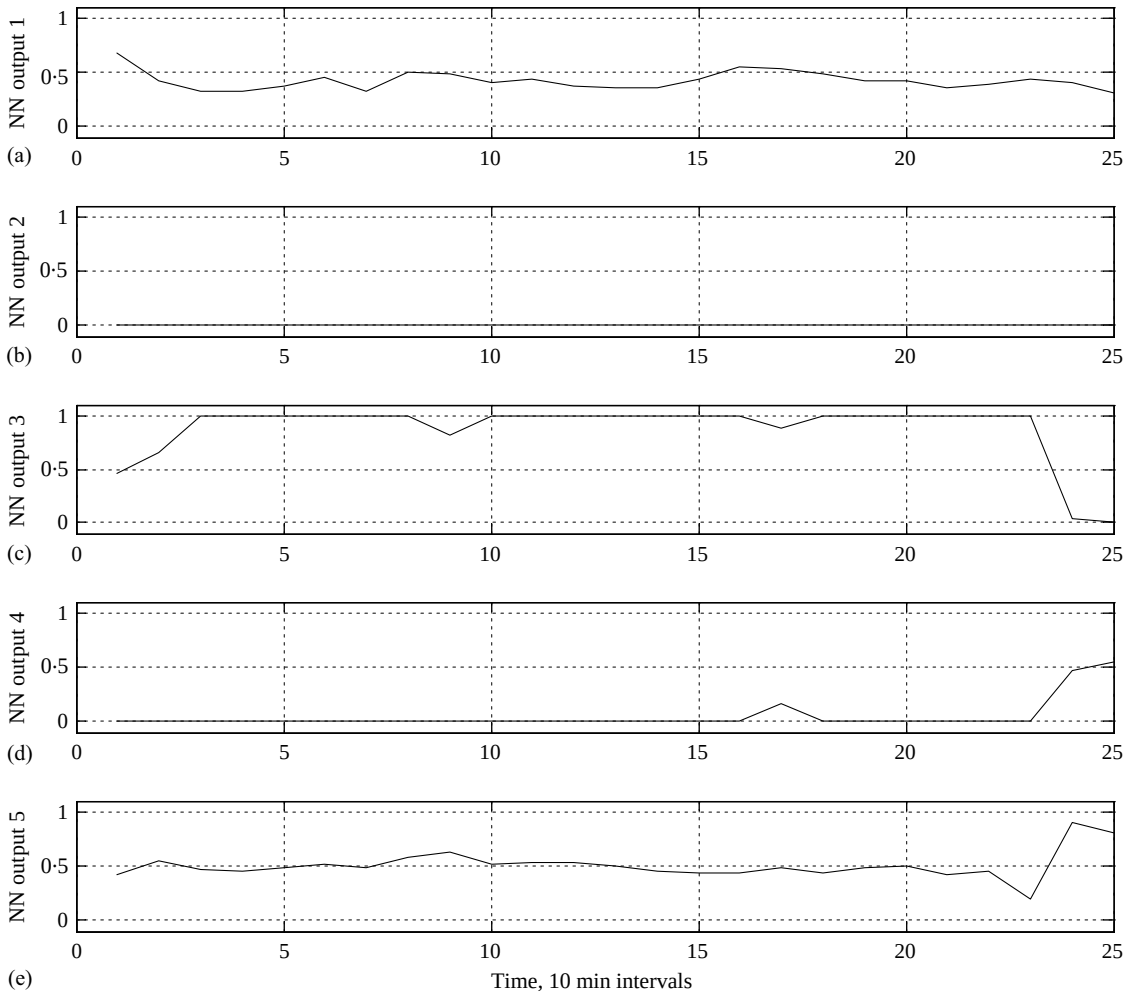

Fig. 7. Neural network (NN) outputs during a 'fault type 2' data set: (a) normal output; (b) actuator fault, type 1 output; (c) mechanical fault, type 2 output; (d) pH sensor fault, type 3 output; (e) electrical conductivity (EC) sensor fault, type 4 output

its occurrence, when the values of the first output of the NN (normal operation) fall to near zero and the values of the second output (fault type 1) increase drastically. The faulty indication of 'fault type 2', failure in the circulation pump, shown as a peak in the third NN output around interval 35, cannot be interpreted as an actual false alarm because at the following time step, that output value has returned to normal values. *Figure 7* shows the NN outputs during the exploration of a testing set that contains data during the existence of 'fault type 2', which is the circulation pump failure. The fault was detected 20 min after its occurrence (NN output 3). The values of 'normal' and 'fault type 4' outputs were quite high during the entire data set, but with the exception of 'fault type 4' output towards the end of the set, they were kept below the upper threshold of 0·6. In *Fig. 8*, the NN outputs during a testing set that contains data during the existence of 'fault type 3' (pH sensor failure) are shown. The fault was detected after approximately 1·5 h. However, from the moment that the fault was imposed, the output that corresponds to the fault (output No. 4), was continuously giving values around 0·5, while the output of 'normal' operation was almost zero. This indicated that something wrong was happening from the beginning. As before (*Fig. 6*), some isolated instances of some outputs cannot be interpreted as indications of the corresponding faults. However, there seems to be a periodic reduction (but not below 0·5) of the values of 'fault type 3' output. This can be explained by the nature of the imposed pH sensor fault, which was a periodic sine-wave noise added to the sensor readings. In that way, at periods when the noise was close to zero, the faulty indication returned to normal values. However, normal operation was not indicated for those periods, thus the system 'knew' that there was something wrong, even though the pH values

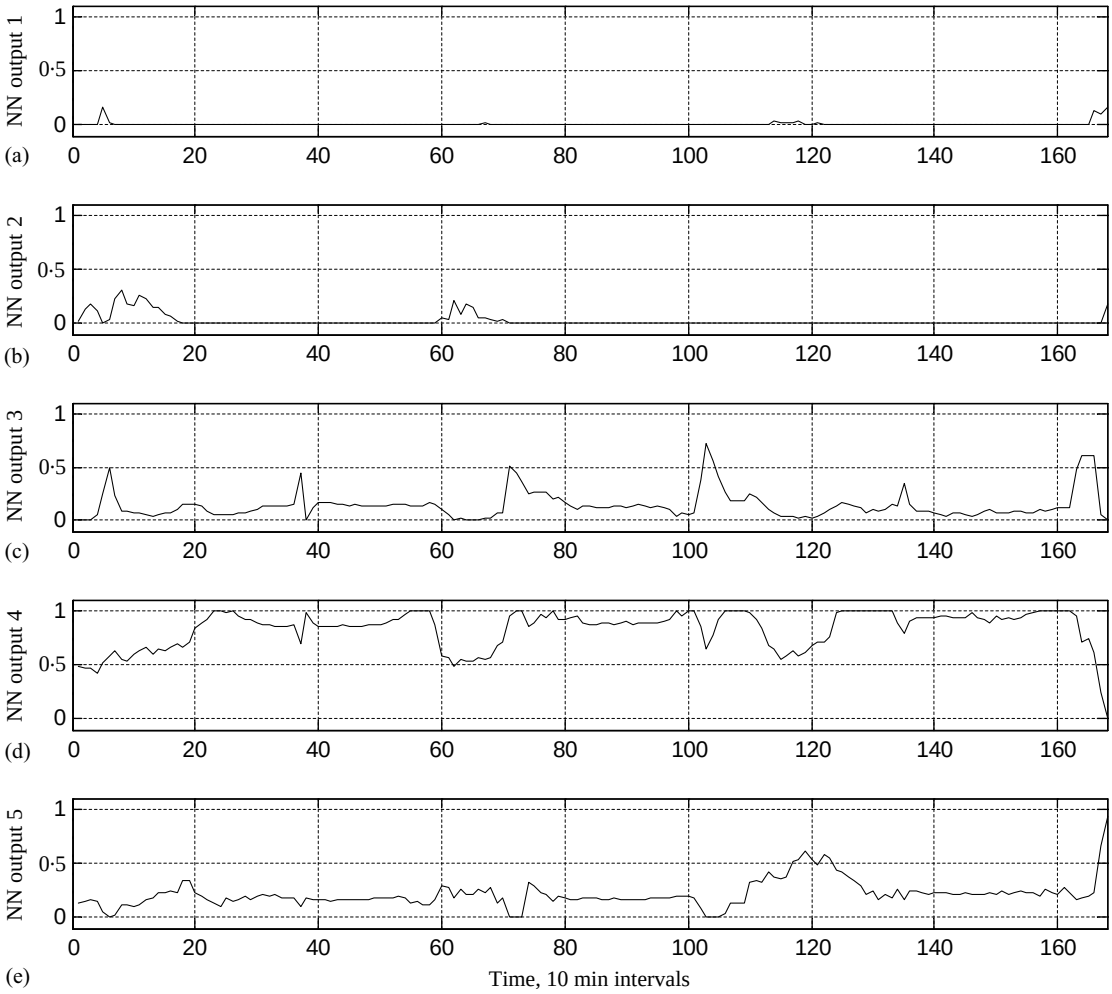

Fig. 8. Neural network (NN) outputs during a 'fault type 3' data set: (a) normal output; (b) actuator fault, type 1 output; (c) mechanical fault, type 2 output; (d) pH sensor fault, type 3 output; (e) electrical conductivity (EC) sensor fault, type 4 output

were normal. In addition, this effect is not crucial, as the fault had already been detected and diagnosed. *Figure 9* shows the NN outputs during a testing set that contains data during the existence of 'fault type 4', which is the EC sensor failure. The fault was detected after 20 min. Before that, an instantaneous indication of 'fault type 3' was given, but the value of that output returned to normal in the next time step. The periodic fluctuations of the 'fault type 4' output and, at the same time, the indication of normal operation (output No. 1) happened, like in the case of the pH sensor fault, because of the periodic nature of the imposed fault, which consisted of adding a sine-wave noise to the sensor readings. At periods when the noise was close to zero, the NN outputs indicated normal operation. This effect was stronger in this type of fault than it was in the pH sensor fault because EC was not controlled automatically. Thus, the FD system did not have additional information concerning EC, as it had for pH, from the values of the pH control actuator (pH control pump). That additional information gave to the NN the capability of not indicating normal operation during those periods in the case of the pH sensor ('normal' output always close to zero); something that did not happen in the case of the EC sensor fault. Finally, *Fig. 10* shows the NN outputs during a testing set that contains data of normal operation. The normal operation was indicated throughout the entire set, without any false alarms.

3.2. *Validation of the genetic algorithm system*

From the results presented in the previous section, it is evident that the GA system was capable of finding a good solution that gave satisfactory accuracy in training and testing of the NN. This system was developed in

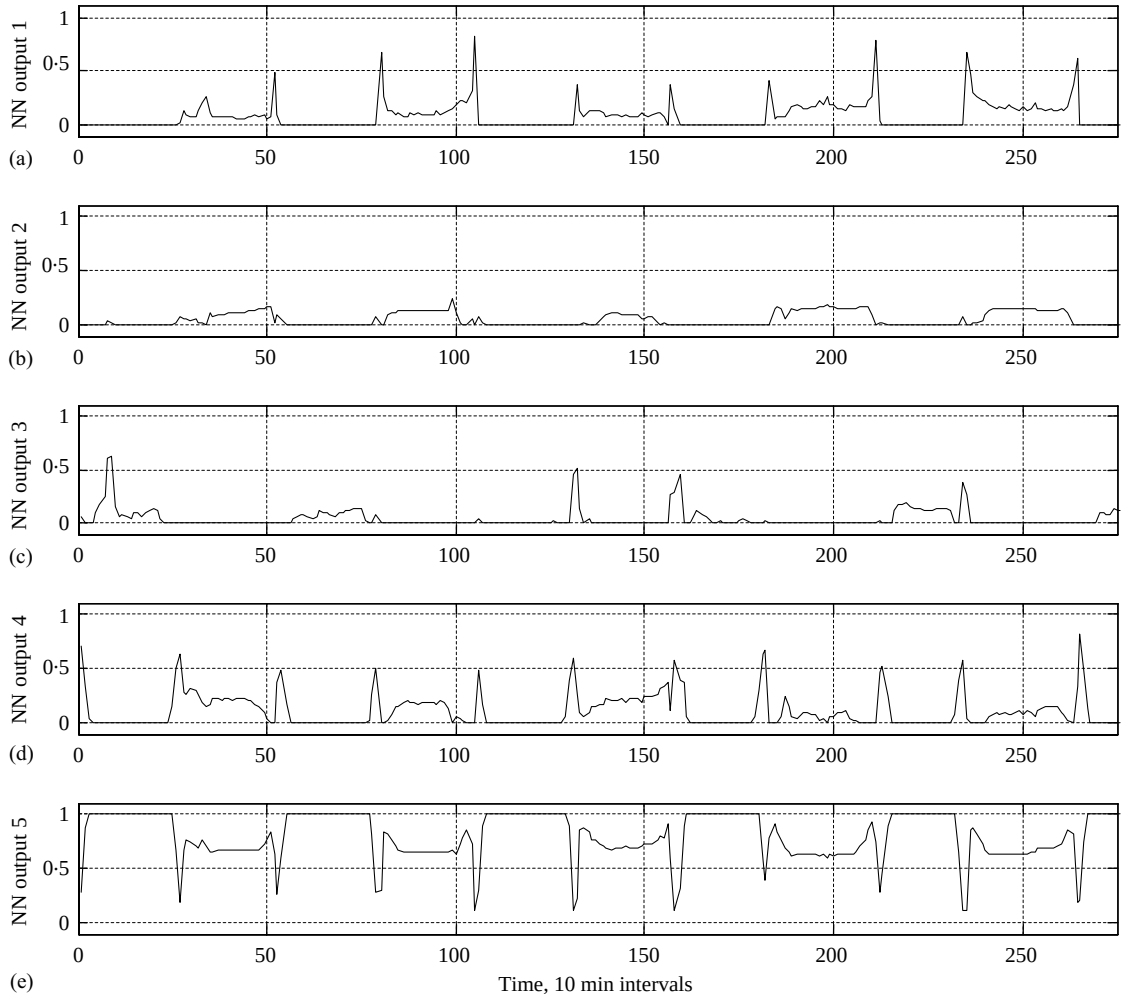

Fig. 9. Neural network (NN) outputs during a 'fault type 4' data set: (a) normal output; (b) actuator fault, type 1 output; (c) mechanical fault, type 2 output; (d) pH sensor fault, type 3 output; (e) electrical conductivity (EC) sensor fault, type 4 output

order to optimise and automate the procedure of finding the best combination of network architectures, training algorithms and activation functions for a specific NN model. The commonly used technique for this procedure is trial-and-error. The search space of the general problem under investigation is infinite. After some restrictions on the available training algorithms and types of activation functions, and mainly after some specific limitations on the candidate network architectures, the search space can be drastically diminished. However, even that limited search space is usually too large to be searched exhaustively. Trial-and-error techniques in this kind of search spaces cover limited portions of them. For some cases, the trial-and-error approach may be 'lucky', for some others it may be unfortunate. But when the search space is vast, the probabilities of being lucky decrease dramatically. However, the chances of trial-and-error approach to be successful increase with the use of experience and possible insight by the person who performs the trials.

In the specific problem under consideration, the search space consisted of 2^{10} (i.e. 1024) possible combinations. If one also thinks that each combination has to be trained several times with different initial conditions, then it is obvious that exhaustive search becomes practically infeasible and a trial-and-error approach becomes quite problematic. A sophisticated heuristic optimisation algorithm, like GA, is able to explore vast search spaces in an intelligent way so that computational power is drastically limited compared to an exhaustive search and the result is more likely to be a good solution than the result of a trial-and-error approach.

From these, it can be concluded that a direct comparison of the solutions achieved by the GA system and by a trial-and-error approach cannot lead to some

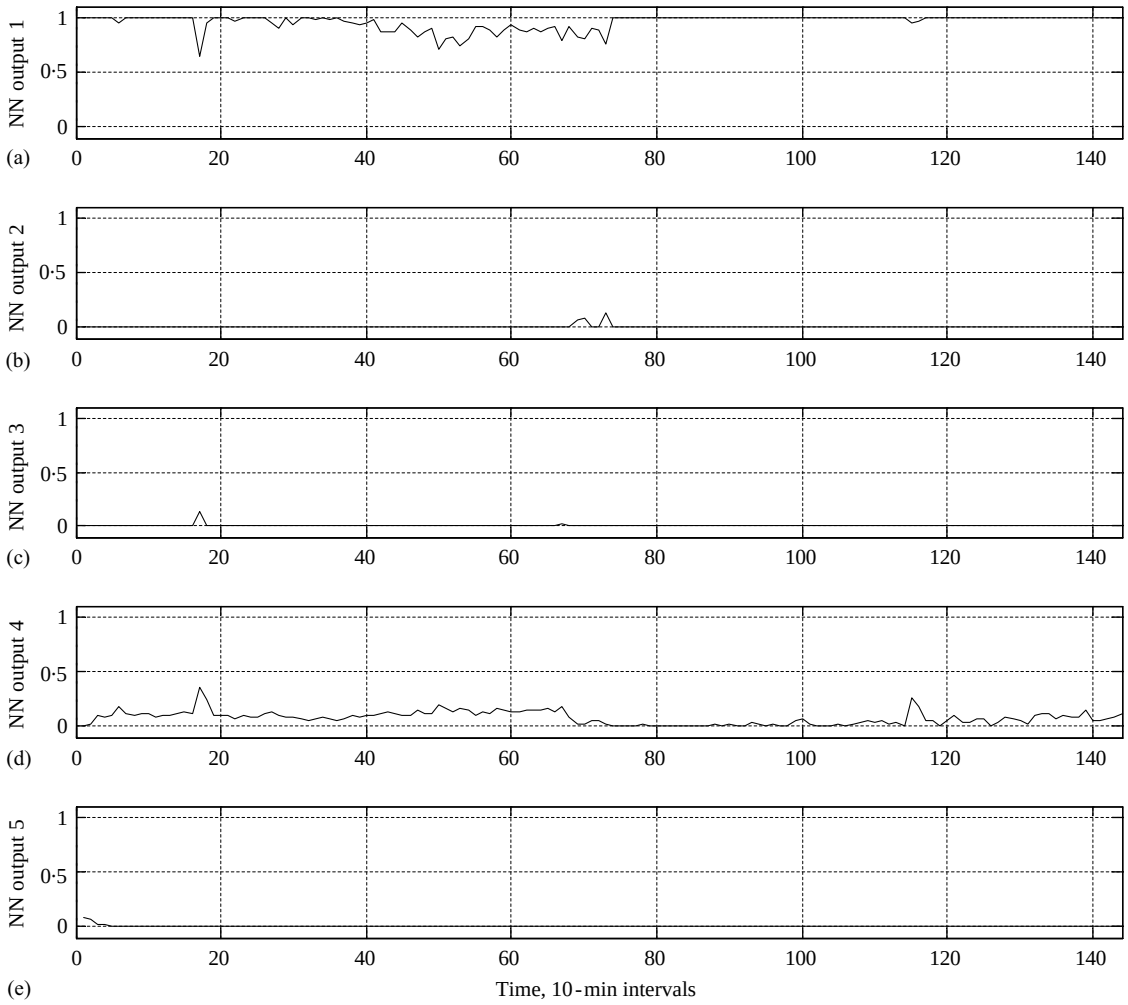

Fig. 10. Neural network (NN) outputs during a data set of normal operation: (a) normal output; (b) actuator fault, type 1 output; (c) mechanical fault, type 2 output; (d) pH sensor fault, type 3 output; (e) electrical conductivity (EC) sensor fault, type 4 output

absolute decision on which is the best method. However, it can give an insight into the degree of success of the GA system, especially if one thinks that human as well as computational efforts are reduced by such an automated approach. Thus, a preliminary training was performed, using the traditional trial-and-error experimentations, in order to find the best network architecture and training algorithm combination. Any information extracted from the results of the GA system that had already been run was assumed to be unknown. However, because it was impossible to explore these combinations with all four possible combinations of activation functions for the hidden and output nodes ('logistic/logistic', 'logistic/linear summation', 'tanh/tanh' and 'tanh/linear summation'), only the combinations containing linear summation functions were explored. This could be considered as 'cheating' because no previous information or experience existed or could exist for this decision, which was completely based on the results of the GA system and which gave clear indication that the 'logistic/logistic' and 'tanh/tanh' combinations seemed to perform poorly compared to the ones that had linear summation functions in the output nodes.

The preliminary explorations mainly focused on 1-HL architectures because most of the 2-HL ones that were investigated at the beginning of the experimentations performed poorly compared to the results of the 1-HL networks. The best solution according to the trial-and-error approach was the 1-HL NN with 30 hidden nodes, hyperbolic tangent activation functions in hidden nodes and linear summation functions in output nodes, trained with the steepest descent algorithm. This solution gave a value for the MSE of 0·2081, which is slightly better than the best solution of the GA system, and is the same as the fourth best solution of the GA system. Also, the

second best solution of the trial-and-error method was the one that was actually found by the GA system and was used as the final network (1-HL with 28 hidden nodes).

The high similarity of the results given by the two approaches is an indication that the performance of the GA system was indeed successful. Only the fourth solution of the trial-and-error approach proposes a different training algorithm (conjugate gradient algorithm), but its performance is much worse than the performances of the GA system solutions. In addition, if hyperbolic tangent activation functions had not been used in the trial-and-error approach, then the best solution of the trial-and-error method would have been much worse than the solution of the GA system.

The advantages and disadvantages of the GA system could be summarised as follows.

Advantages of the GA system

(1) It is based on some optimisation heuristic for combinatorial problems.
(2) It is automated, thus it requires much less human effort than trial-and-error.
(3) It incorporates some user experience about the modelling problem (mainly in the choice of the range of network architecture search space).
(4) It extends the user experience by using the evolutionary properties of the GA optimisation.
(5) It is very robust in applications of feedforward NNs.
(6) It can exploit the developing technology of parallel processing.

Disadvantages of the GA system

(1) It is not fully automated, as the parameters of the GA algorithm have to be adjusted.
(2) It can be trapped in local minima if the initial population is not good enough.
(3) It requires some general *a priori* knowledge or intuition about the modelled system.

4. Discussion

The NN model developed in this work, together with an 'alarm decision' tool, form the complete fault detection and diagnosis scheme, as shown in *Fig. 11*. The greenhouse that contains a hydroponic system with cultivated plants is subjected to the outside weather conditions and the inside climate control scheme. All these have an effect on the hydroponic system which, in addition, is subject to the hydroponic control scheme. The connection between this control scheme and the hydroponic system is where the actuator fault is introduced. The mechanical fault is within the hardware of the hydroponic system itself, while the biological fault concerns the plants of the system. Outputs from both the greenhouse and the hydroponic system are fed into the FDNN model. The last kind of fault considered, the sensor fault, is applied to the output of the hydroponic system. Some of the hydroponic system and greenhouse outputs, together with the transpiration data of the plants, are also fed into the 'biological fault detection (BFD) neural network' model (Ferentinos *et al.*, 2002). The outputs of the network go into the 'alarm decision' scheme, which is also provided with some user-defined output thresholds. These thresholds determine the triggering or not of an alarm.

The final performance of the developed FD model can be considered satisfactory, based on the fact that faults of all four types considered were detected in reasonable time, while the probability of false alarms was very limited. Some of these faults (*e.g.* the circulation pump failure) could have easily been detected by simpler methodologies than the one used here, *i.e.* feedforward NNs. However, the majority of faults detected by the developed system deal with complicated processes like actuators that do not operate continuously, and sensor failures. In addition, the effects of these faults are interrelated in complicated ways such that the detection and, more specifically, the isolation of the specific existing fault (diagnosis) requires advanced knowledge extraction and pattern recognition capabilities that simpler detection methods do not include.

The existence of the additional output for normal operation gives the model a high degree of adaptability in new, unknown failures of the system. With this output, the model can detect some 'unknown fault' by simply indicating neither 'normal operation' (zero in the 'normal operation' output) nor fault indication (zero in all four 'fault' outputs). This means that there is neither a known fault in the system, nor normal operation, which leads to the conclusion that some unknown failure exists. If the additional output of normal operation was not included in the network, then, in the case of an 'unknown' fault, normal operation would be — mistakenly — concluded. This capability was not tested in the current work because of the lack of appropriate data, but its conceptual validity is indisputable and can be validated with real data in the future.

As far as the biological faults are concerned, all observations showed that biological faults of the type imposed in this study could not be detected in this kind of cultivation system using the measurable variables that were used in this research. The interaction between plants and their root zone microenvironment is not equally balanced, as the condition of the plants is highly influenced by the conditions in their root zone

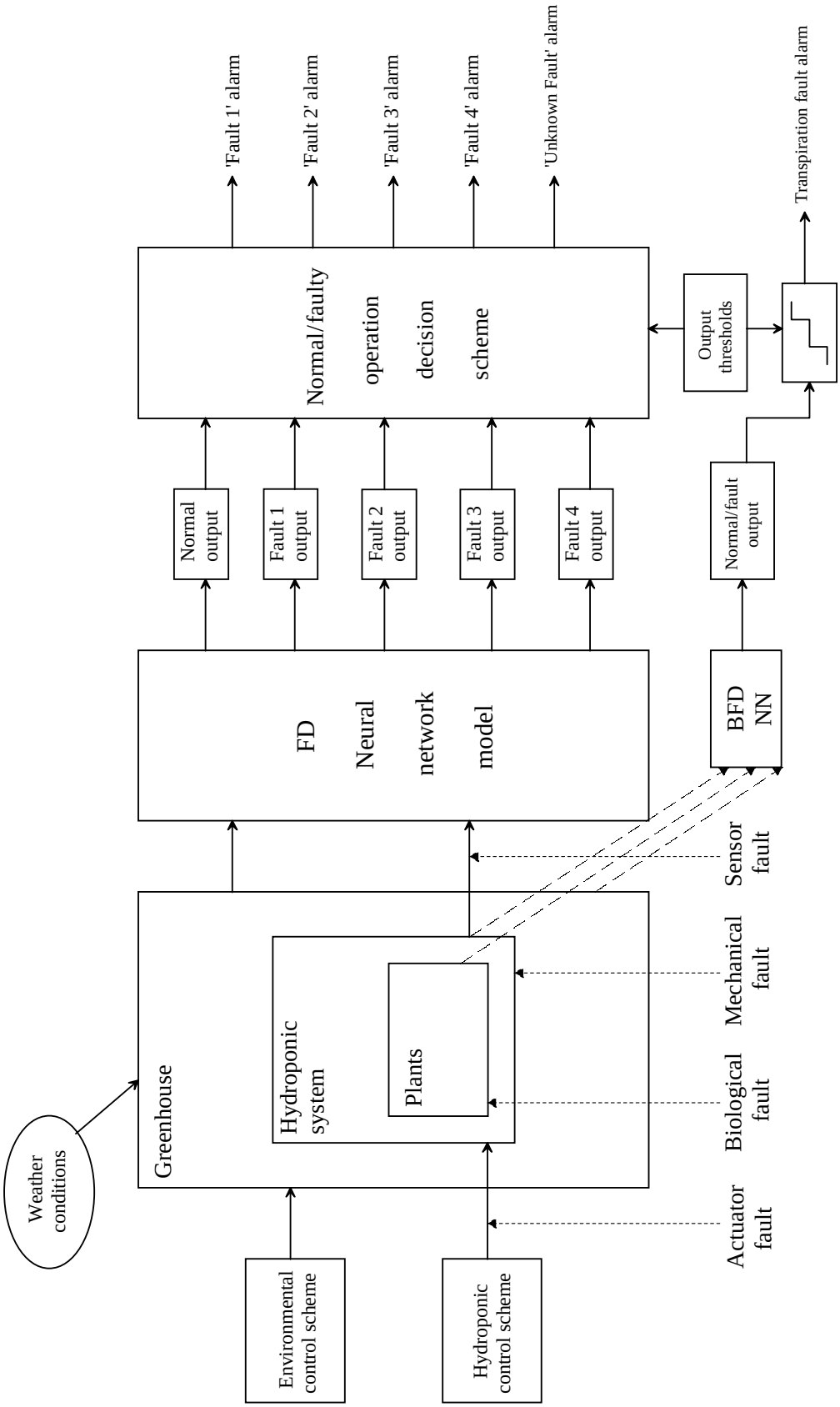

Fig. 11. Fault detection and diagnosis scheme: FD, fault detection; BFD NN, biological fault detection neural network

microenvironment, while these microenvironment conditions are not influenced in the same degree by the conditions of the plants. The most probable explanation of this property is the inertia that the overwhelming mass of the nutrient solution (compared to the mass of the plants) produces, in a way that it becomes a limitation factor to the influence of plants condition to their root zone microenvironment. However, the successful detection of some specific biological fault that has to do with the transpiration rate, even though it is understandable that that fault severely affected the plants and represents an extreme, rather unrealistic situation, gives motivation for further research in the area of biological fault detection and diagnosis. Several approaches could be followed. One possibility could be the use of additional measured variables of the plant microenvironmental conditions like, for example, changes of the nitrate concentration of the nutrient solution. Another approach could be the application of this kind of detection system to different types of hydroponics, such as nutrient film technique (NFT) systems. In NFT, the roots of the plants contact the nutrient solution only in thin film of solution, in contrast to the case of deep-trough systems where the roots are in contact with the entire mass of the solution of each pond. Finally, an approach that could lead to the desired detection of biological faults could be the combination of the proposed methods with the use of more 'plant-oriented' measurements that could more precisely represent the actual conditions of the plants than the measured variables of this work.

5. Conclusions

The work presented here investigated the capabilities of detecting and diagnosing failures in a deep trough hydroponic system at an early stage, that is, before they become visible or obvious and their effects become irreversible. The feedforward neural network methodology was used as the main tool for the development of the fault detection model. The major advantage was that no mathematical model of the process was needed; thus the fault detection model was entirely based on real, measured data. In the process of designing the fault detection neural network model, a new technique for neural network synthesis was developed, based on the heuristic optimisation method of genetic algorithms.

As far as the fault detection and diagnosis model of mechanical and sensor faults is concerned, the following conclusions can be derived.

(1) All mechanical and sensor faults considered (pH control pump failure, circulation pump failure, pH sensor failure and electrical conductivity sensor failure) were detected and diagnosed in real time at an early stage, so that the fault detection model can be applied on-line as a reliable supervisor of the operation of the human-unattended hydroponic system.
(2) Feedforward neural networks were capable of learning the physical and chemical interactions between the plants and the measured variables of the root zone and shoot zone microenvironments as well as the effects of specific failures of the system to these variables.
(3) The developed system showed great generalisation capabilities in the task of detecting and diagnosing faults in new data sets.
(4) The one-hidden-layer architecture of neural networks proved to be more successful than the two-hidden-layer one. The steepest-descent backpropagation training algorithm with on-line adjustable learning rate by solving the Hessian matrix, gave the best results. The most appropriate activation functions of the network were the hyperbolic tangent function for the hidden nodes and the linear summation function for the output nodes. The final neural network consisted of 28 hidden nodes.

As far as the application of the novel methodology for neural network synthesis based on genetic algorithms is concerned, the following conclusions can be derived.

(1) Genetic algorithm encoding and optimisation can be successfully applied to a combinatorial problem such as the decision of what is the best neural network architecture, activation functions and training algorithm for a specific model.
(2) Compared to the common for these tasks trial-and-error procedure, the genetic algorithm gave similar results.
(3) In most cases, the genetic algorithm model should be preferable, mostly because it is an automated methodology, compared to the trial-and-error one and it is based on optimisation properties.
(4) For some cases, the output of the genetic algorithm methodology could give useful information and insights for further, more successful application of the trial-and-error approach.

Finally, as far as the biological faults are concerned, it was discovered that biological faults of the type imposed in this study cannot be detected in this kind of cultivation system using the measurable variables that were used in this research. The inertia introduced

by the overwhelming mass of the nutrient solution compared to the mass of the plants, can be considered as the most important limiting factor that led to the lack of success in detecting, in general, biological faults.

References

Chow M; Yee S O (1991). Methodology for on-line incipient fault detection in single-phase squirrel-cage induction motors using artificial neural networks. IEEE Transactions on Energy Conversion, **6**(3), 536–545

Duda R O; Hart P E; Stork D G (2001). Pattern Classification. John Wiley & Sons Inc., New York, NY

Ferentinos K P (1999). Artificial neural network modeling of pH and electrical conductivity of hydroponic systems. MS Thesis, Cornell University Libraries, Ithaca, NY

Ferentinos K P (2002). Neural network fault detection and diagnosis in deep-trough hydroponic systems. PhD Dissertation, Cornell University Libraries, Ithaca, NY

Ferentinos K P; Albright L D; Selman B (2002). Neural network based detection of mechanical, sensor and biological faults in deep-trough hydroponics. Computers and Electronics in Agriculture, Special Issue on Artificial Intelligence in Agriculture

Filho E; de Carvalho A (1997). Evolutionary design of MLP neural network architectures. Proceedings of the Fourth Brazilian Symposium on Neural Networks, pp 58–65, Goiania, GO, Brazil, December 3–5

Fine T L (1999). Feedforward Neural Network Methodology. Springer-Verlag, New York, NY

Fogel D B (1995). Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, New York, NY

Goldberg D E (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA

Harp S A; Samad T; Guha A (1989). Towards the genetic synthesis of neural networks. Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, CA, pp 360–369

Hashimoto Y (1989). Recent strategies of optimal growth regulation by the speaking plant concept. Acta Horticulturae, **260,** 115–121

Hoskins J C; Kaliyur K M; Himmelblau D M (1991). Fault diagnosis in complex chemical plants using artificial neural networks. AIChE Journal, **37**(1), 137–141

Iyoda E M; von Zuben F J (1999). Evolutionary hybrid composition of activation functions in feedforward neural networks. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'99), Washington, DC, Vol. **6,** pp 4048–4053

Kitano H (1990). Designing neural networks using genetic algorithms with graph generation system. Complex Systems, **4,** 461–476

Miller G F; Todd P M; Hegde S U (1989). Designing neural networks using genetic algorithms. Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, CA, pp 379–384

Parlos A G; Muthusami J; Atiya A F (1994). Incipient fault detection and identification in process systems using accelerated neural network learning. Nuclear Technology, **105,** 145–161

Phelan R M (1977). Automatic Control Systems. Cornell University Press, Ithaca, NY

Rumelhart D E; Hinton G E; Williams R J (1986). Learning representations by back-propagating errors. Nature, **323,** 533–536

Sonneveld C; Straver N (1994). Nutrient solutions for vegetables and flowers grown in water or substrates. Proefstation voor de bloemisterij en glasgroente, Vol. **8**. Naaldwijk, The Netherlands

Sorsa T; Koivo H N; Koivisto H (1991). Neural networks in process fault diagnosis. IEEE Transactions on Systems, Man, and Cybernetics, **21**(4), 815-825

Venkatasubramanian V; Chan K (1989). A neural network methodology for process fault diagnosis. AIChE Journal, **35,** 1993–2002

Watanabe K; Matsuura I; Abe M; Kubota M; Himmelblau D M (1989). Incipient fault diagnosis of chemical processes via artificial neural networks. AIChE Journal, **35,** 1803–1812

Xiaoming L; Visier J-C; Vaezi-Nejad H (1997). A neural network prototype for fault detection and diagnosis of heating systems. ASHRAE Transactions: Symposia, **103**(1), 634–644