A NOVEL GA-BASED NEURAL MODELLING PLATFORM FOR NONLINEAR DYNAMIC SYSTEMS

Jian-Xun Peng and Kang Li

School of Electrical & Electronic Engineering
Queen's University Belfast, UK

Abstract: A novel genetic algorithm based neural modelling platform is developed for nonlinear dynamic systems. In this platform, the selection of neural inputs, optimisation of the neural network structure and identification of the optimal connection weights are formulated as an integrated optimisation problem. Both one-step-ahead and long-term prediction performances of the neural model are incorporated into the performance index. Genetic algorithms are then used for this mixed integer nonlinear optimisation problem. The platform provides user-friendly environment that gives users maximal flexibility in defining the modelling project and detailed visualisations of the optimisation process. This platform is then applied to the NOx emission modelling and prediction of a coal-fired power generation plant to confirm its effectiveness.

Keywords: Genetic algorithms, neural networks, optimisation, neural modelling, nonlinear system, software development, power generation

1. INTRODUCTION

Neural networks have been widely used in nonlinear dynamic system modelling due to their universal approximation ability and simple model structure. Several issues in neural modelling have been examined (Anders and Korn, 1999; Li, Thompson and Peng, 2004; Yu, Gomm, and Williams, 2000), but they have not been fully exploited as a whole.

The first issue is the dilemma between the short-term and long-term prediction performances of a neural model. Many training algorithms for neural modelling are based on a criterion which assesses the one-step-ahead prediction performance of the neural model. In practice, a one-step-ahead prediction model may carry little system information, since the sampling rate in a process may be high enough that the immediate past outputs give the most significant contributions to the cost function. Therefore the contributions of other useful variables become much less important and indistinctive. To avoid this situation, the long-term prediction performance is often preferred to be a part of the performance index to assess the neural model. However most batch algorithms cannot be directly used to effectively train external recurrent neural networks against a performance index taking into account the long-term prediction performance.

The second issue is how to control the complexity of a neural model (Anders and Korn, 1999). It is well understood that although a network having more hidden neurons are better in fitting the training data, a complex network is often poor in predicting unseen data. The popular methods are to prune redundant hidden nodes based on some information criteria or to regularise the network training by adding

additional term on the cost function to penalise large weights.

The third important issue is to select the neural inputs (Yu, Gomm, and Williams, 2000). To feed the neural model with all possible variables in dynamic forms often produce a complex neural network with poor generalisation performance. The existing methods are to select possible neural inputs from multiple local linear models.

Although the three issues have been addressed separately in the literature, they are rarely considered as a whole, partly due to the complexity of the problem and partly due to the lack of powerful computation tools. In this paper the three issues are dealt with in one integrated framework with the support of a powerful optimisation tool.

This paper is organised as follows. In section 2, an optimisation criterion combining both the one-step-ahead and the long-term prediction performance of the neural model is proposed. Neural input selection, network structure optimization, and optimal connection weight identification are formulated as one problem. Then a scheme is proposed to solve the integrated optimization problem using genetic algorithms (GAs). In section 3, an integrated software platform is described. This platform is applied to model NOx emissions from a coal-fired power generation plant in section 4 and concluding remarks are presented in section 5.

2. FORMULATION OF THE NEURAL NETWORK OPTIMISATION PROBLEM

To simplify the notation, MISO systems are considered in this paper. A discrete nonlinear dynamical system of *m* inputs and single output can then be represented as:

$$y(t) = f(y(t-1),\cdots,y(t-l_y),\boldsymbol{u}(t-1),\cdots,\boldsymbol{u}(t-l_u)) \quad (1)$$

where t is the time instant, $y(t) \in \boldsymbol{R}$ is the output, $\boldsymbol{u}(t) = [u_1(t),\cdots,u_m(t)]$ the input vector, $f(\bullet)$ is some nonlinear function, and l_u and l_y are the maximal orders for input and output variables.

Suppose a neural network model is used to approximate the nonlinear system (1) with one or more hidden layers using feed-forward structure. All system input variables in past time instant, $u_i(t-1),\cdots,u_i(t-l_u), i=1,\cdots,m$, and the system output in past time instant, $y(t-1),\cdots,y(t-l_y)$, are fed into the network to generate the network output $y(t)$ which is the system output in current time instant. Let the number of hidden layers as *h*, and then the neural network can be formularized as follows:

$$\left.\begin{aligned} y(t) &= w_0 + \sum_{i=1}^{n_h} w_i x_i^{(h)}(t) \\ x_i^{(k)}(t) &= \phi_i^{(k)}\left(w_{0\sim i}^{(k)} + \sum_{j=1}^{n_k} w_{j\sim i}^{(k)} x_j^{(k-1)}(t) \right) \\ & k = 1,\cdots,h \end{aligned}\right\} \quad (2)$$

where $[x_1^{(0)},\cdots,x_{n_0}^{(0)}] = [y(t-1),\cdots,y(t-l_y), u_1(t-1),\cdots,u_1(t-l_u),\cdots,u_m(t-1),\cdots,u_m(t-l_u)]$ denotes the input layer to the network; n_k, $k=1,\cdots,h$ denotes the number of neurons in the *k*'th hidden layer; $x_i^{(k)}$, $i=1,\cdots,n_k$ is the output of the *i*'th neuron in the *k*'th hidden layer; w_i, $i=1,\cdots,n_h$ is the output weight from the *i*'th neuron of the last (the *h*'th) hidden layer to the output and w_0 is the output bias; $w_{j\sim i}^{(k)}$, $i=1,\cdots,n_k$, $j=1,\cdots,n_{k-1}$, is the connection weight from the the *j*'th neuron of the (*k*-1)'th hidden layer to the *i*'th neuron of the *k*'th hidden layer and $w_{0\sim i}^{(k)}$ is the bias; $\phi_i^{(k)}$, $i=1,\cdots,n_k$, denotes the activation function of the *i*'th neuron of the *k*'th hidden layer; $x_1^{(0)},\cdots,x_{n_0}^{(0)}$ denote the n_0 input signals to the network and $n_0 = l_y + ml_u$.

In general, activation functions are selected 'a priori' for traditional neural networks, e.g. hyperbolic tangent or Gaussian, etc. Given the activation functions, the network can be formulated as a function of the connection weights and biases, which are to be optimised against a criterion. Now suppose a set of data of *N* samples, denoted as Ω, is obtained, and the neural network needs to be trained over the data set, i.e. to minimise a cost function, normally the least squares:

$$\boldsymbol{W}^* = arg \min_{\boldsymbol{W}\in\boldsymbol{R}^K} \left(\sum_{t=1}^{N} [\hat{y}^{(1)}(t) - y(t)]^2 \right) \quad (3)$$

where $\boldsymbol{W}$ denotes all the weights and biases, including $w_{j\sim i}^{(k)}$ and w_i, and *K* denotes the total number of the weights together with biases,

$$K = \sum_{k=0}^{h} (n_k + 1) n_{k+1} \quad (4)$$

with n_{h+1} denoting the number of network outputs, here $n_{h+1} = 1$; $\{ y(t), t = 1,\cdots,N \}$ is the output series and $\{ \hat{y}^{(1)}(t), t = 1,\cdots,N \}$ are network predictions, where the superscript "(1)" indicates that they are the one-step-ahead predictions of the system output.

To train the network using (3), the network structure, including the number of hidden layers and the number of neurons in each hidden layer, has to be predetermined. Furthermore if all possible neural inputs are used, the network will inevitably get more complex. In this paper, optimisation of the network structure and identification of the optimal connection weights are formulated as one optimisation problem - Suppose all the inputs $x_i^{(0)}$'s to the network and the outputs $x_i^{(k)}$'s from all the hidden neurons are all referred to as *signal channels*, then the total number of signal channels is

$$M = \sum_{k=0}^{h}(n_k + 1)n_{k+1} \tag{5}$$

where n_{h+1} is the number of network outputs, and in this paper, n_{h+1}=1 for single-output networks. Each signal channel in the network is tagged with a Boolean variable, say $s_i^{(k)}$ for $x_i^{(k)}$, which is called selection variable. Obviously we have *M* selection variables for the *M* signal channels. If $s_i^{(k)} = 1$ then the signal channel $x_i^{(k)}$ is enabled (or being selected) and the weights connecting to the signal channel are also enabled, otherwise the signal channel and its associated weights in the network are disabled (or unselected). Then network (2) can be reformulated as

$$\left.\begin{aligned} y(t) &= w_0 + \sum_{i=1}^{n_h} w_i s_i^{(h)} x_i^{(h)}(t) \\ x_i^{(k)}(t) &= \phi_i^{(k)}\left(w_{0\sim i}^{(k)} + \sum_{j=1}^{n_k} w_{j\sim i}^{(k)} s_j^{(k-1)} x_j^{(k-1)}(t) \right) \end{aligned}\right\} \quad k = 1,\cdots,h \tag{6}$$

All the *M* selection variables then can be re-written using a vector,

$$\boldsymbol{S} = [\, s_1^{(0)},\cdots,s_{n_0}^{(0)},\cdots,s_1^{(h)},\cdots,s_{n_h}^{(h)}\,] \in \{0,1\}^M \tag{7}$$

which is referred to as the *selection vector*. Obviously the selection vector defines the actual network structure.

To minimize the least squares index (3), the output weights $\boldsymbol{w} = [\, w_0, w_1,\cdots,w_{n_h}\,]^T$ can always be explicitly expressed as

$$\boldsymbol{w} = (\boldsymbol{X}_h^T \boldsymbol{X}_h)^{-1} \boldsymbol{X}_h^T \boldsymbol{y} \tag{8}$$

$\boldsymbol{y} = [\, y(1),\cdots,y(N)\,]^T$, $\boldsymbol{X}_h = [\, \boldsymbol{x}_0^{(h)},\cdots,\boldsymbol{x}_{n_h}^{(h)}\,]$,

$\boldsymbol{x}_i^{(h)} = [\, x_i^{(h)}(1),\cdots,x_i^{(h)}(N)\,]^T, i = 0,1,\cdots,n_h$,

and $x_0^{(h)}(t) = 1, t = 1,\cdots,N$.

Now that the output weights are given in (8), the task to optimise the network structure and the connection weights is then to identify the best selection vector and a parameter vector that are associated with the selected selection vector, excluding the output weights $\boldsymbol{w}$. This parameter vector excluding the output weights is defined as:

$$\begin{aligned} \tilde{\boldsymbol{W}} = [\, & w_{0\sim 1}^{(1)},\cdots,w_{n_0\sim 1}^{(1)}, w_{0\sim 2}^{(1)},\cdots,w_{n_0\sim 2}^{(1)}, \\ & \cdots, w_{0\sim n_h}^{(h)},\cdots,w_{n_{h-1}\sim n_h}^{(h)}\,] \end{aligned} \tag{9}$$

The number of elements in vector $\tilde{\boldsymbol{W}}$ is $\tilde{K} = \sum_{k=1}^{h}(n_{k-1} + 1)n_k$.

Given the above definitions, a one-to-one mapping can be established between the networks and the vector pair (**S**, $\boldsymbol{W}$), i.e. a vector pair (**S**, $\boldsymbol{W}$) uniquely specifies a neural network. The least squares criterion index in (3) can be reformulated as

$$\begin{aligned} J^{(1)}(\boldsymbol{S},\boldsymbol{W}) &= \sum_{t=1}^{N}[\,\hat{y}^{(1)}(t) - y(t)\,]^2 \\ &= \boldsymbol{y}^T\left[\boldsymbol{I} - \boldsymbol{X}_h(\boldsymbol{X}_h^T\boldsymbol{X}_h)^{-1}\boldsymbol{X}_h^T\right]\boldsymbol{y} \end{aligned} \tag{10}$$

where the criterion index $J^{(1)}$ is defined as the sum squares of one-step-ahead prediction errors. And the network input selection, structure optimization and identification of optimal connection weights are performed using

$$\begin{aligned} &(\boldsymbol{S}^*,\boldsymbol{W}^*) = arg\,min\{J^{(1)}(\boldsymbol{S},\boldsymbol{W})\} \\ &\quad \text{s.t. } \boldsymbol{S} \in \{0,1\}^K \text{ and } \boldsymbol{W} \in \boldsymbol{R}^M. \end{aligned} \tag{11}$$

To control the complexity of neural networks, an additional constraint is imposed on the network optimisation, i.e. the total number of neural inputs and hidden neurons is restricted, which is denoted as *C*, i.e. $C = sum(\boldsymbol{S})$. The constrained optimization problem is then reformulated as

$$\begin{aligned} &(\boldsymbol{S}^*,\boldsymbol{W}^*) = arg\,min\{J^{(1)}(\boldsymbol{S},\boldsymbol{W})\}, \\ &s.t. \quad \boldsymbol{S} \in \{0,1\}^K, sum(\boldsymbol{S}) = C, \boldsymbol{W} \in \boldsymbol{R}^M \end{aligned} \tag{12}$$

where *K* and *M* are dimensions of vector $\boldsymbol{S}$ and $\boldsymbol{M}$, which are defined in (4) and (5), respectively; *C* is a predetermined constant integer, which restricts the total number of neural inputs and hidden neurons. *C* is an index relating to the neural model complexity therefore its choice should reflect the complexity of the problem. Alternatively, *C* could be incorporated into the performance index.

In (12), $J^{(1)}(\boldsymbol{S},\boldsymbol{W})$ uses the one-step-ahead prediction errors. As pointed out in the introduction section that, in modelling nonlinear dynamic systems however the long-term prediction performance of a

model is often more important. To produce a neural model with good long-term prediction performance, the optimisation criterion defined in (12) has to be modified. Given a network defined by the vector pair ***(S, W)*** with its output weights determined in (8), its long-term predictions is defined as

$$\begin{cases} \hat{y}^{(l)}(t) = w_0 + \sum_{i=1}^{n_h} w_i s_i^{(h)} x_i^{(h)}(t) \\ x_i^{(k)} = \phi_i^{(k)}\left(w_{0\sim i}^{(k)} + \sum_{j=1}^{n_k} w_{j\sim i}^{(k)} s_j^{(k-1)} x_j^{(k-1)} \right), k = 1,\cdots,h \\ [x_1^{(0)}(t),\cdots,x_{n_0}^{(0)}(t)] = [\hat{y}^{(l)}(t-1),\cdots,\hat{y}^{(l)}(t-l_y), \\ u_1(t-1),\cdots,u_1(t-l_u),\cdots,u_m(t-1),\cdots,u_m(t-l_u)] \end{cases} \tag{13}$$

In this long-term prediction model, past values of the system output fed into the neural networks have been replaced with their predicted values by the neural network. The sum squares of the long-term prediction errors is defined as

$$J^{(L)}(\boldsymbol{S},\boldsymbol{W}) = \sum_{t=1}^{N} [\hat{y}^{(l)}(t) - y(t)]^2 \tag{14}$$

Now integrated optimisation criteria for the neural network can be defined as

$$J(\boldsymbol{S},\boldsymbol{W}) = [a_1 J^{(1)}(\boldsymbol{S},\boldsymbol{W}) + a_l J^{(L)}(\boldsymbol{S},\boldsymbol{W})]/(a_1 + a_l) \tag{15}$$

where $a_1, a_l \geq 0$ are predefined weights for the one-step-ahead and long-term prediction performance of the model. Finally the neural network optimisation problem can be formulated as

$$\begin{aligned} &(\boldsymbol{S}^*,\boldsymbol{W}^*) = \arg\min\{J(\boldsymbol{S},\boldsymbol{W})\}, \\ &s.t. \quad \boldsymbol{S} \in \{0,1\}^K, sum(\boldsymbol{S}) = C, \ \boldsymbol{W} \in \boldsymbol{R}^M \end{aligned} \tag{16}$$

Obviously (16) is a real-integer mixed hard problem. Conventional analytic methods often fail to search the optimal solution for such problems. As a stochastic optimisation tool, the genetic algorithm has ever been used to train neural networks (Blanco, Delgado and Pegalajar, 2001). In this paper, it will be used for the above mixed optimisation problem.

3. NEURAL NETWORK OPTIMIZATION PLATFORM USING GENETIC ALGORITHMS

To solve an optimization problem using genetic algorithms, a solution-encoding scheme is first required, through which each solution to the problem is represented as a chromosome. With regard to the optimization problem (16), a solution comprises two parts, i.e. a boolean selection vector ***S*** and a real-valued parameter vector ***W***. The scheme of encoding the solutions is illustrated in Fig. 1.

As shown in Fig. 1, each chromosome has C integer genes and M floating-point genes; C and M have been defined above. In a GA search, it is always ensured that $I_i \in [0, K-1], i = 1,..,C$ (K is the length of the vector **S**) and $I_i \neq I_j$ for $i \neq j, i, j = 1,\cdots,C$. Each chromosome is tagged with a positive number, called fitness. Fitness is a measure of the performance of a solution coded in chromosome. In this paper, $J(\boldsymbol{S},\boldsymbol{W})$ defined in (15) is used as the raw fitness of a chromosome. To adjust the competition scale between different chromosomes in a population, fitness ranking is often applied.

Based on the above analysis, GA-based C++ software has been developed as an integrated platform for neural network based nonlinear dynamic modelling. The main characteristics of the software are:

1) Maximal flexibility for project definition. The GA based neural modelling is initialised by a project definition file. Users can define GA parameters, the system inputs and outputs, training and validation data sets, the network structure, and the criterion to be minimised etc.

2) Both homogenous and heterogeneous networks are supported. An activation function pool is designed to enable users to uses any of the functions listed for the hidden and output nodes. The software also supports user-defined activation functions. In addition, user defined activation functions are allowed to contain any parameters to be optimised.

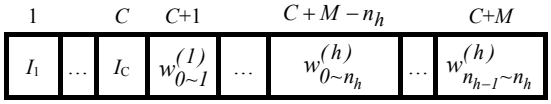

Fig. 1 Chromosome encoding scheme

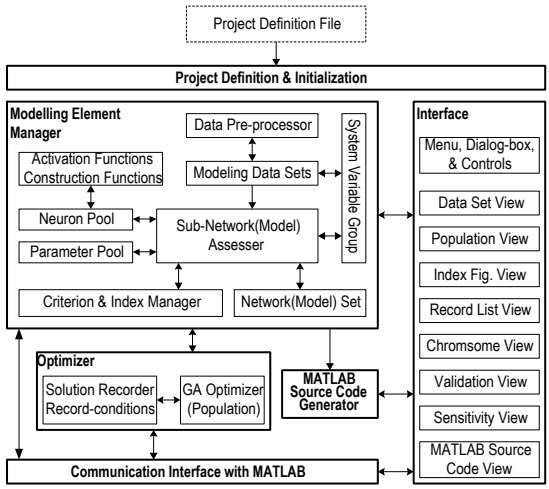

Fig. 2. Software structure

3) A data pre-processing module is developed, which can be used to perform data normalisation.

4) Multiple sets of data may be specified for both network optimisation and validation. Different format of data sets are supported, e.g. MATLAB files (*.m), ASCII text data files (*.txt) and binary data files (*.dat).

5) The C++ software has interfaced with MATLAB, which enables the software to exchange data with a MATLAB workspace and to call MATLAB functions.

6) A MATLAB source code generator is designed. It can output neural models in MATLAB code in order that the results produced in the software can be reproduced in the MATLAB using the generated MATLAB codes.

7) Different "views" are designed to monitor the evolution process in text or graphic form, for examples, to visualize the evolution process, the validation results, and parameter sensitivity.

Fig. 2 summarises the software structure and information flow between different modules.

4. APPLICATION AND RESULTS

The proposed neural modelling method was applied to model the NOx emission in a cyclic thermal power plant with two generating units (Li, Thompson and Peng, 2004). Each unit has a dual fired (oil or coal) drum boiler and produces full load 300 MWe with oil firing or 200 MWe with coal firing. There is one burner box on each corner. There are four burner boxes, one at each corner that will supply the furnace with oil or pf coal. All of the sections in each burner box tilt in unison through ±25°, relative to the horizontal. This is achieved by means of a burner tilt mechanism.

Coal is the major source of fuel. Coal is taken from the bottom of a coal bunker, pulverised and then entrained in a hot primary airflow. Primary air dries and carries the pf coal to the burners. Each corner of the furnace houses a burner box and a separated overfire air (SOFA) box, which admit fuel and secondary air streams into the furnace. These streams are directed at tangents to imaginary firing circles in the centre of the furnace. The tangential firing creates turbulence in the combustion area that ensures the thorough mixing of fuel and air streams necessary for complete combustion. Low momentum burners are employed to achieve a longer flame path, leading to reduced flame temperatures.

Plant data covering about 6-day's operation period were obtained. The following data sampled each minute were available: NOx): Mass flow of fuel m_f (Kg/s); Mass flow of air m_a (Kg/s), specifically, mass flow of primary air m_{pa} and mass flow of secondary air m_{sa} (Kg/s); Tilting position of burners θ (degree).

Table 1 Statistics of the NO_x emissions (ppm)

	Data set 1	Data set 2	Data set 3
Mean	306.80	296.92	298.70
STD	21.92	20.85	17.40

```
[Parameters]
   Parameters 01 = w01: 02, [-1,1]
[Neural Network]
   Input Layer            = [mf, mpa, msa, tilt], 6
   Output Layer           = [NOx]
   Hidden Layer 1         = [5 * dllMLEXP], 5
   Input Preprocessing    = Normalisation {Period 1, Period 2, Period 3}
[Extended Index]
   Criterion      = 0.2 * NMSE1 ( NOx, { Period 1 } | {Period 1} ) + \\
                    0.8 * NMSEL ( NOx, { Period 1 } | {Period 1} )
   Validation 1   = NMSEL( NOx, {Period 2, Period 3 } | { Period 1 } )
   Validation 2   = NMSEL( NOx, {Period 1, Period 2, Period 3 } | {Period
2} )
   Validation 3   = NMSEL( NOx, {Period 1, Period 2, Period 3 } | {Period
3} )
   Evolution Record       = {Validation 1, Validation 2, Validation 3}
```

Fig. 3 Definition of the network set, the criterion and validation indices in the project definition file

The normalized data were split into three sets (data set 1 with 2300 samples, data set 2 with 3000 samples and data set 3 with 2500 samples). The mean value and standard deviations for the NOx emissions in the three data sets are listed in Table 1.

Of the three data sets, data set 1 is used for neural model optimisation, data set 2 and 3 for model validation. A one-hidden layer neural network with activation function $\varphi = e^{(c/(x+b))}$ was developed. In the chromosome representation, all the system inputs and the NOx output with an order of 4 were coded as the neural network inputs, and 5 hidden neurons were also coded into the chromosome. These network inputs and hidden nodes constituted the segment of integer genes in the chromosome. The genetic algorithm was then used to select a subset of these integer genes and to optimise their associated floating genes to generate a solution to the optimisation problem.

The neural network structure defined in the project definition file is shown in Fig. 3. In Fig. 3, field [Parameters] defines the parameters in the activation functions to be optimised. In the file, two parameters, named *w01* and *w02*, are defined within range [-1, 1]. They represent c, b in the activation functions φ. In field [Neural Network] in Fig. 3, input layer contains all system input variables in time, and it is required that at least 6 of these inputs are selected. 'dllMLEXP' stands for φ. NMSE1 and NMSEL in Fig. 3 are build-in functions for calculating normalized mean squares of one-step-

ahead and long-term prediction errors. The software interface for neural modelling of NOx emissions is illustrated in Fig. 4.

The neural model optimised against the performance index defined in Fig. 3 (Model 1) is a 6-5-1 neural network comprising 6 inputs and 5 hidden nodes. The GA parameters are: one population with a size of 400, crossover rate 0.95, and mutation rate 0.05.

Finally, to illustrate the flexibility of this modelling platform, another 6-5-1 neural model was optimised against the NMSE1 only (Model 2), i.e. NMSE1 is set to 0, and NMSEL is set to 0 in Fig. 3. The GA parameters are kept the same.

Table 2 Short -term prediction performance (RMSE) of the two neural models

Model	Data set	50 gen	100 gen
Model 1	Modelling	6.7665	6.7072
	Valid-1	4.8149	4.7035
	Valid-2	4.5262	4.5044
Model 2	Modelling	6.1864	6.1763
	Valid-1	4.2435	4.2488
	Valid-2	4.6641	4.2321

Table 3 Long-term prediction performance (RMSE) of the two neural models

Model	Data set	50 gen	100 gen
Model 1	Modelling	13.4718	13.0780
	Valid-1	16.3923	13.4783
	Valid-2	12.0873	10.9220
Model 2	Modelling	16.6533	17.6698
	Valid-1	14.9589	17.6416
	Valid-2	13.6269	11.8767

Table 4 Average short and long-term prediction performance (RMSE) of the two neural models

Model	Data set	50 gen	100 gen
Model 1	Short term	5.3837	5.3176
	Long term	14.2721	12.5912
Model 2	Short term	5.0183	4.8920
	Long term	15.0784	16.0306

The short-term and long-term prediction performances of the two neural models over the three data sets, when optimised against the two different performance indexes for 50 and 100 generations, are given in tables 2 and 3. The average model performance over the data sets is summarised in table 4. The model performance - RMSE stands for the rooted mean of sum squared predicted errors (RMSE).

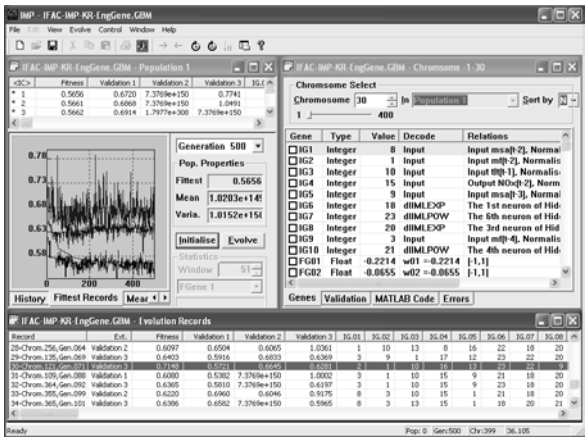

Fig. 4 Layout of the software interface

5. CONCLUSION

A novel GA-based neural modelling platform has been developed. Using this platform, the selection of neural inputs, optimisation of the neural network structure and identification of optimal connection weights can be formularized as one integrated optimisation problem. Both one-step ahead and long-term prediction performances of the neural models are incorporated into one fitness function for GA optimisation. This platform has then applied to model the NOx emissions from a thermal power plant to confirm the effectiveness and flexibility in modelling complex nonlinear dynamic engineering systems.

ACKNOWLEDGEMENTS

Dr K. Li wishes to acknowledge the financial support of the UK Engineering and Physical Sciences Research Council (EPSRC Grant GR/S85191/01).

REFERENCES

Anders, U., O. Korn (1999). Model selection in neural networks. *Neural Networks*, vol. 12, pp. 309-323.

Blanco, A., M. Delgado and M. C. Pegalajar (2001). A real-coded genetic algorithm for training recurrent neural networks. *Neural Networks,* **14**, 93-105.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc.

Li, K., S. Thompson and J. Peng (2004). Modelling and prediction of NO_x emission in a coal-fired power generation plant. *Control Engineering Practice*. **12**, 707-723.

Yu, D. L., J. B. Gomm, D. Williams (2000). Neural model input selection fro a MIMO chemical process. *Engineering Application of Artificial Intelligence*, vol. 13, pp. 15-12, 2000.