15th Triennial World Congress, Barcelona, Spain

DYNAMIC MODELLING USING GENETIC PROGRAMMING

Mark Hinchliffe and Mark Willis

m.p.hinchliffe@ncl.ac.uk, mark.willis@ncl.ac.uk
Advanced Control Group, Department of Chemical and Process Engineering
University of Newcastle, Newcastle upon Tyne NE1 7RU, UK

Abstract: In this contribution we demonstrate how a Single Objective Genetic Programming (SOGP) and a Multi-Objective Genetic Programming (MOGP) algorithm can be used to evolve accurate input-output models of dynamic processes. Having described the algorithms, two case studies are used to compare their performance with that of Filter-Based Neural Networks (FBNNs). For the examples given, the models generated using GP have comparable prediction performance to the FBNN. However, performance with respect to additional modelling criteria can be improved using the MOGP algorithm. *Copyright © 2002 IFAC*

Keywords: Genetic algorithms, dynamic modelling, multi-objective optimisation

1. INTRODUCTION

Previous work (McKay *et al.*, 1997, 2000) has shown that Genetic Programming (GP) is able to evolve accurate input-output models of steady-state process systems. However, it is often necessary for the engineer to be able to accurately predict the dynamic response of the process. Therefore, the aim of this work is to extend the use of the GP methodology to the modelling of dynamic process systems.

A study of the literature shows that there are several ways to modify the GP framework so that it can be applied to dynamic modelling:

1) Use a GP algorithm to evolve sets of differential equations, e.g. see Cao et al (1999). This can be time-consuming, as it requires multiple sets of differential equations to be solved.
2) Use a terminal set containing feedback loops and recursive nodes (Bettenhausen and Marrenbach, 1995). This technique has been successfully applied to the modelling of biotechnological processes, although the computing requirements again appear to be rather high.
3) The output may be represented as a function of input and output values shifted back in time (time series approach). GP has been used to develop time series models in a number of research areas. Notable references include, Fleming and Rodríguez-Vázquez (1998) who applied a multi-objective GP algorithm to the development of Non-linear Auto-Regressive Moving Average eXogeneous (NARMAX) models for gas turbine engine identification and Kulkarni *et al.* (1999) who used GP to develop ARX models of industrial processes including a CSTR and a heat exchanger.

The time series form of model is used in this work. The novelty of the work is that we not only show how a SOGP may be applied to dynamic systems modelling but also extend the work to consider the use of a MOGP. Furthermore, using two case studies, we demonstrate the modelling capabilities of our algorithms in comparison to that of a FBNN.

2. SOGP ALGORITHM

We adopt a multi-basis function (MBF) approach where each population member is a linear sum of a number of non-linear basis functions,

$$\hat{y}_k = a_0 + \sum_{j=1}^{m} a_j g_j(u_{k-1}, \hat{y}_{k-1}, q^{-1}) \qquad (1)$$

The g_j are m basis functions (m was chosen as uniformly random integer in the range [1 10]), which are functions of the process input, u, and the process output, y. q^{-1} is the back-shift operator (for example, $q^{-1}y_k=y_{k-1}$) and is used to indicate that a time series of values are used. a_j are constants, a_0 is a bias or offset term and k represents the current time sample. In order to develop models that can be used for long-term prediction, we assume that the actual process output values $(y_{k-1},\ldots,y_{k-\tau})$ are unknown and cannot be used as model inputs. The predicted output values $(\hat{y}_{k-1},\ldots,\hat{y}_{k-\tau})$ generated by the model are used

instead. This form of prediction is sometimes referred to as 'pure' prediction (Henson and Seborg, 1997) as the method only requires the process inputs in order to predict the output over the entire data set. This is especially important if the model is to be used for carrying out process simulations, as the output must be assumed unknown. As equation (1) is linear in the parameters, the constants ($a_0,\ldots,a_n$) can be optimised using the method of recursive least squares (RLS). It is possible that the input-output data could be analysed before any runs were undertaken in order to estimate system time delays and determine appropriate orders for the auto-regressive and input terms required by the model. However, this would mean that GP was not operating as an automated modelling tool, making few *a priori* assumptions about the underlying structure of the model. A more elegant approach is to provide the algorithm with building blocks that allow the number of process lags to be adjusted as the run proceeds. This can be accomplished by including the back-shift operator, q^{-1}, in the function set. Dynamic models can then be created from a smaller terminal set consisting solely of the process input(s) and model output shifted by a single time sample. Nesting of back-shift operators enables the algorithm to build the necessary time-shifted input and output terms without having to precisely define the number of lags at the start of the run. The GP algorithm settings and parameters are summarised in Table 1.

Table 1. GP algorithm settings and parameters.

Function set	+,-,/,*,^,SQRT,SQR,EXP, LOG,q0,q1,q2,q3
Terminal set	u scaled in range [0 1], y Constants in range [-10 10]
Crossover	0.7
Mutation	0.2
Direct reproduction	0.1
Generation gap	90%
Fitness measure	RMS error
Selection method	Linear ranking.
Maximum tree size	500 characters.

The function set contains the primitives +, -, /, *, ^ (power), SQRT (square root), SQR (square), EXP (exponential) and LOG (logarithm). The q1, q2 q3 represent the back-shift operators q^{-1}, q^{-2} and q^{-3} and u and y are the input and output terminals, u_{k-1} and $\hat{y}_{k-1}$ respectively. As it is convenient to simply assign a back-shift operator to every input/output terminal appearing in a newly generated model equation, it is necessary to include an operator that does not perform a time-shift. This ensures that there is a uniform distribution of time-shifts in the initial population and allows model terms with a single process lag to be produced (e.g. q0(u1)=u_{k-1}). Apart from the terminal and function sets, the other algorithm features and settings are the same as those used for steady-state modelling, including the use of high-level crossover which enables the algorithm to exchange whole basis functions and adapt the total number of functions present in each population member. Low-level crossover provides a mechanism for subtrees to be transferred between individuals. (see Hinchliffe, 2001 for further details)

3. FILTER-BASED FEEDFORWARD NEURAL NETWORKS

The simplest way to use standard feedforward artificial neural networks for dynamic modelling is to use a time history of input variables, $u_{k-1},u_{k-2},\ldots u_{k-n}$, as inputs to the network. This means that each network input consists of a process input variable shifted back in time (Bhat and McAvoy, 1989). This is similar to the Finite Impulse Response (FIR) approach to dynamic modelling and therefore has the disadvantage that a long time history of inputs may be required to enable the network to accurately capture the dynamics of the process. This means that a large number of inputs may be required giving rise to a complex network with a large number of parameters to be optimised. This will make network training more time consuming due to the increased complexity of the problem.

These problems can be avoided by using a FBNN (e.g. see Turner *et al.* 1996), where the hidden layer neurons are augmented with first order transfer functions or "filters" with a gain of one. The process of training filter-based neural networks requires the optimisation of the filter constants as well as the weights and bias values. This was carried out using a Levenberg-Marquardt training algorithm.

4. CASE STUDY 1

The following system equation was used to generate input-output data for the purpose of testing the modelling capabilities of the SOGP and MOGP algorithms,

$$y_k = \frac{y_{k-1}y_{k-2}(2.5+y_{k-1})}{1+y_{k-1}^2+y_{k-2}^2} + u_{k-11} \qquad (2)$$

The input signal, u, was generated as a multi-level pseudo random signal of 400 data points in the range [0 5]. Two hundred of these data points were used for training and the remaining 200 data points were used for model validation. Due to the probabilistic nature of the GP algorithm, and because the training of FBNNs can be affected by the random initialisation of network weights, multiple runs of each algorithm were carried out so that a fair comparison of the two algorithms can be made. Each algorithm was run 20 times. The GP algorithm had a population size of 100 and was run for 50 generations. The neural network runs used an 'early stopping' criteria to stop training when the RMS error on the validation data increased. The FBNN algorithm was run using network architectures ranging from 1 to 15 hidden layer neurons. The GP model with the lowest validation RMS error is shown in Table (2). Equation (3) shows the first basis function in the form it was stored by the GP algorithm (with some simplification to improve readability).

```
-0.4848*q1(q2(q1(q2(q1(q3(u)
+q0(-9.3343e-2))))+3.8604))-1.871          (3)
```

The GP algorithm has combined a series of back-shift operators in order to model the system time delay (the u_{k-11} term is constructed from a combination of six back-shift operators).

Table 2. SOGP model with lowest validation RMS error.

Basis Functions	Parameters
$-0.4848u_{k-11} - 3.698$	-1.068
$-0.4848u_{k-12} - 5.562$	-0.3091
$(\hat{y}_{k-1} + \hat{y}_{k-2})(\hat{y}_{k-2} - \exp((u_{k-4} - u_{k-3})\hat{y}_{k-2} - \hat{y}_{k-3}))$	-0.01907
$-0.4848\hat{y}_{k-2} + 0.4848\hat{y}_{k-1}$	0.3091
$\hat{y}_{k-2} - 3.860$	0.2659
$\hat{y}_{k-2} + \hat{y}_{k-3}$	0.04660
0.1482	-2.549
Bias	-4.259

In this case study, it is difficult to make an unbiased comparison between the neural network and the GP algorithm. The FBNN does not have the ability to directly model the system time delay (unless the appropriate time shifted value of the process input is supplied) and will be at a disadvantage when compared to the GP algorithm. The easiest way to overcome this problem is to present the network with input data delayed by the correct sample delay. However, this will hand the advantage to the neural network, as the GP algorithm has to use a combination of back-shift operators and u_{k-1} terminals to identify the time delay. One solution is to carry out two batches of network runs, with and without the time delay removed, and observe the relative performance of the GP algorithm.

Table 3 provides a numerical summary of the results obtained. As expected, the neural network with time delay compensation (FBNN#2) easily outperforms the network that does not have the time delay removed (FBNN#1). The GP algorithm produced a wide range of prediction errors, with the worst values lying in the same region as those generated by FBNN#1 and the lowest errors approaching the accuracy of FBNN#2. Although the GP algorithm was able to achieve performance comparable to FBNN#2 in a fraction of the runs, the algorithm is also capable of producing unacceptably poor results.

Table 3. Validation data prediction errors (1-9-1 and 1-13-1 refer to the number of neurons in each network layer).

	Min	Mean	Max
SOGP	0.0047	0.0238	0.0622
FBNN#1 (1-9-1)	0.0390	0.0477	0.0549
FBNN#2 (1-13-1)	0.0033	0.0064	0.0106

This should be expected, as the algorithm has the difficult task of evolving a suitable model structure from elementary building blocks. However, the fact that the process time delay does not have to be explicitly accounted for gives the GP algorithm a distinct advantage over neural networks.

5. MOGP ALGORITHM

Process model development is a task that may require a number of other factors or 'objectives' to be considered before the final solution is reached. Examples of possible objectives include measures of model parsimony, such as the number of model parameters and the maximum number of process lags. Additional or alternative measures of prediction error could be considered, for instance, residual variance, one-step ahead and long-term prediction errors. Model validation criteria such as residual correlation tests and statistical information criteria could also be incorporated. An advantage of using GP for model development is that the algorithm can be easily modified to incorporate additional measures of model performance. The most significant difference between the MOGP and SOGP algorithms is that the former makes use of a Pareto based ranking scheme with fitness sharing.

Pareto-based techniques have become increasingly popular in recent years. This is especially true for real world scientific and engineering applications to which 90% of Pareto based multi-objective evolutionary algorithm (MOEA) applications are applied (Van Veldhuizen, 2000). The family of solutions to a multi-objective optimisation problem is said to be Pareto-optimal if, for each individual, an improvement in performance in one objective dimension cannot be achieved without degrading performance with respect to other objectives.

Unfortunately, for real problems, the set of Pareto-optimal solutions for a particular problem may be very large. It will therefore be difficult to effectively sample all regions of the trade-off surface using a GP algorithm that has a relatively small population size. This problem can be overcome by using the goal based Pareto ranking method proposed by Fonseca and Fleming (1995). This enables the user to specify desired levels of performance in each objective domain and direct the search towards the required region of the search space.

Although evolutionary algorithms such as GP are capable of simultaneously exploring different regions of the solution space, genetic drift may cause the algorithm to eventually converge around one region of the trade-off surface. To counteract the effects of this phenomenon and promote diversity, niche induction methods may be used. One such method is that of fitness sharing where individuals that are closer to each other mutually decrease each other's fitness. Consequently, individuals that are more isolated are given a greater chance of reproducing.

Further details of the MOGP algorithm may be found in Hinchliffe, 2001.

6. MODEL RESIDUALS

The residuals of a model represent the difference between the predicted and actual values of the process output. Consequently, the presence of any information remaining in the residuals is an indication that the proposed model may be inadequate in some way. The existence of such information can be investigated by using a number of techniques. For example, some tests measure the correlations between residuals and inputs(s) and the autocorrelation of the residuals. Other approaches include checks for normally distributed residuals and the number of zero crossings (changes of sign) of the residual sequence. These techniques are usually applied after the model structure and associated parameters have been identified. A potential advantage of using a MOGP approach is that performance with respect to the tests will be taken into account throughout the model evolution process. Although any combination of these criteria could be used as objectives within a MOGP framework, the correlation tests outlined by Billings and Voon (1986) were a used. The tests are as follows,

$$\textbf{(a)}\ \phi_{\varepsilon\varepsilon}(\tau)=\delta(\tau)\ \ \textbf{(b)}\ \phi_{u\varepsilon}(\tau)=0\ \ \textbf{(c)}\ \phi_{\varepsilon\varepsilon u}(\tau)=0 \qquad (4)$$
$$\textbf{(d)}\ \phi_{(u^2)'\varepsilon}(\tau)=0\ \ \textbf{(e)}\ \phi_{(u^2)'\varepsilon^2}(\tau)=0 \qquad \forall\tau$$

ϕ_{xy} signifies the correlation between variables x and y, τ is the time-shift and δ is the Kronecker delta function. The first two tests are the standard autocorrelation and cross correlation functions used in linear system identification. The remaining higher order tests are designed to detect missing non-linear terms by examining the correlations between odd and even powers of the inputs and residuals. The correlation tests are usually performed at the 95% confidence level. This means that the residuals will contain no linear or non-linear structure if the absolute value of each test statistic is not greater than $1.96/\sqrt{N}$, where N is the number of data points. The correlation test objective value (Φ) for a MOGP model can then be found by taking the sum of the test values that exceed the 95% confidence limit. Defining $\phi_1=\phi_{\varepsilon\varepsilon}, \phi_2=\phi_{u\varepsilon}, \phi_3=\phi_{\varepsilon\varepsilon u}, \quad \phi_4=\phi_{(u^2)'\varepsilon}$ and $\phi_5=\phi_{(u^2)'\varepsilon^2}$ yields,

$$\Phi=\sum_{i=1}^{5}\sum_{j=0}^{\tau_{max}}\left(|\phi_i(j)|-1.96/\sqrt{N}\right) \quad \text{if } i=1, j\neq 0 \qquad (5)$$

The maximum time-shift, τ_{max}, was set to twenty process lags for all of the correlation tests carried out in this work. In practice, the results of the tests must be analysed carefully to ensure that τ_{max} is not less than the maximum lag required for an accurate model.

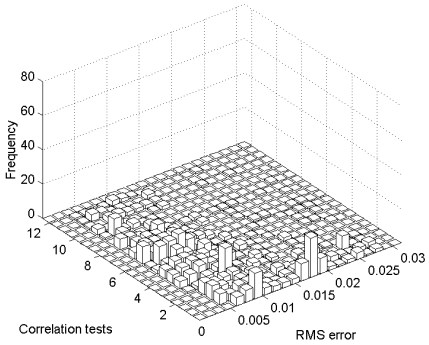

Fig. 1. Distribution of final generation objective values for SOGP.

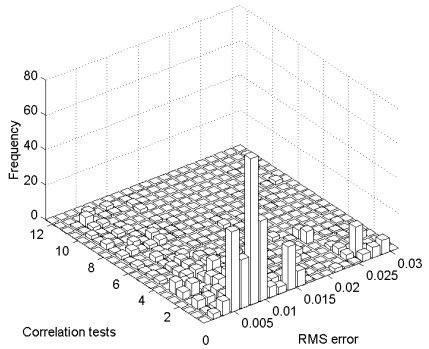

Fig. 2. Distribution of final generation objective values for MOGP.

The distributions of the objective values taken from the final generations of each set of algorithm runs are shown in Figures (1) and (2). It can be seen the SOGP algorithm was able to generate models that have low RMS errors but do not necessarily perform adequately in terms of the correlation tests. The goal based ranking scheme employed by the MOGP algorithm has enabled the algorithm to evolve more solutions in the desired region of the search space. The distribution of the MOGP objective values shows that the algorithm has generated a larger number of individuals that have low RMS errors and have acceptable correlation test performance. Before the 'best' model can be chosen from the non-dominated set of candidate solutions, the performance on the validation data must also be taken into consideration.

Table 4. non-dominated individuals obtained using MOGP with preference information.

Model no.	RMS	Correlation tests	Validation RMS
1	0.002309	3.3707	0.003507
2	0.002353	3.1662	0.003579
3	0.002388	1.3484	0.003234
4	0.002423	0.7742	0.003450
5	0.002720	0.6392	0.003447
6	0.002825	0.4598	0.003691
7	0.002859	0.2848	0.003645
8	0.002908	0.1603	0.003838
9	0.003434	0.0667	0.004079
10	0.003468	0.0158	0.004172
11	0.003938	0	0.004588

Table 4 shows objective values and validation RMS errors for the non-dominated set of MOGP models. Model 3 provides the most accurate prediction on the validation data, with an RMS error of 0.00323 (compared to the best validation RMS of 0.00475 for SOGP). Arguably, model 4 provides a better compromise solution. The model has slightly higher training and validation RMS error values than model 3, but has a significantly lower correlation test score. Figure 3. shows the results of the five correlation tests for this model (+/- 95% confidence limits are shown as horizontal dashed lines).

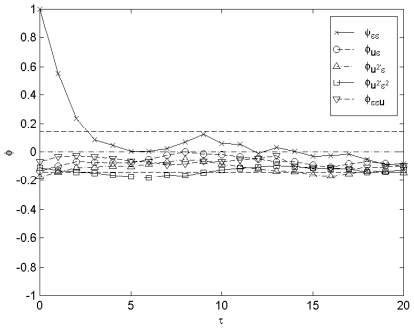

Fig. 3. Correlation test plot for model 4 (RMS=0.00242, Φ=0.774).

The correlation objective value is greater than zero because of the first test, which tests for the presence of autocorrelated residuals. It is conjectured that autocorrelation is indicated as the model contains past outputs of the *model* and not past process outputs. This approach may introduce correlations in the residuals and make it more difficult to completely satisfy all of the tests.

7. CASE STUDY 2 – COOKING EXTRUDER

The aim of this exercise is to develop an input-output model for the degree of starch gelatinisation in an industrial cooking extruder. There are four inputs available for model development: feed flowrate (Q_f), feed moisture content (M_f), screw speed (ω) and the feed temperature (T_f). A data set comprising 400 points was used for training purposes and a further 195 data points were used for model validation. The distributions obtained using the FBNN and SOGP algorithms are given in Fig.4. Although the minimum, mean and maximum RMS values are slightly lower for the neural network, a one-sided Kolmogorov-Smirnov test (at the 95% confidence level) indicates that the difference between the distributions is not significant.

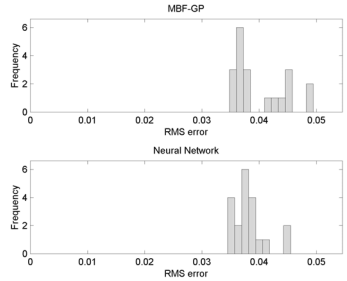

Fig. 4 Validation RMS error distributions.

Fig. 5. compares the performance of the neural network and SOGP algorithms in terms of the computational effort required to achieve a given validation RMS error (the error bars indicate +/- one standard deviation). At higher RMS error values, the SOGP algorithm outperforms the neural network. This could be because the RLS optimisation routine enables the SOGP algorithm to raise the performance of the initial population members to a reasonable level without consuming a particularly large amount of processing power. The neural network may initially produce poor predictions due the large number of model parameters, which all have to be initialised probabilistically.

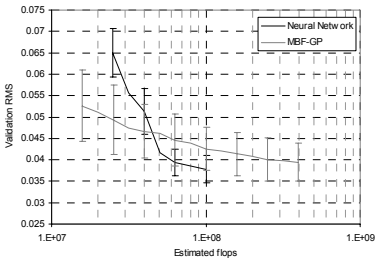

Fig. 5 Comparison of computational effort required by FBNN and SOGP algorithms**.**

As the validation RMS error decreases, the neural network begins to outperform the GP algorithm, requiring fewer FLOPS to achieve the same RMS error. The difference between the algorithms continues to increase as the validation RMS errors are reduced. This could be because neural network training is essentially a parameter optimisation exercise. The GP algorithm has to explore a wide range of different model structures, performing parameter optimisation on each candidate solution, and is therefore unlikely to be as efficient as the neural network. One of the disadvantages of using neural networks is that a wide range of network architectures has to be investigated in order to obtain the best set of model predictions. If the computational effort required to carry out these additional runs is taken into consideration, the difference between the algorithms is less significant and GP becomes a more attractive possibility.

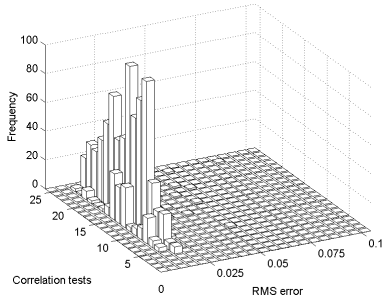

Fig. 6. Distribution of final generation objective values for SOGP.

To determine the advantage gained by using the MOGP 20, further runs were performed with a population size of 100 individuals for 100 generations.

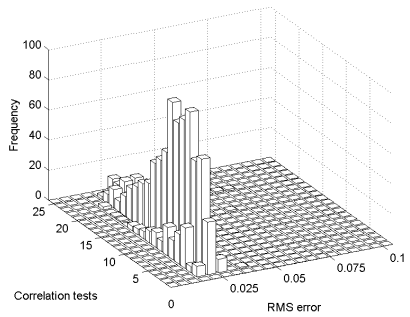

Fig. 7. Distribution of final generation objective values for MOGP.

The distributions of the individuals evolved by the algorithms are compared in Figure (6) and (7). It can be seen the SOGP algorithm was able to generate models that have low RMS errors but do not necessarily perform adequately in terms of the correlation tests. The goal based ranking scheme employed by the MOGP algorithm has enabled the algorithm to evolve more solutions with lower correlation objective values. The non-dominated solutions have comparable RMS errors to the FBNN. However, when the performance with respect to the correlation tests is considered, the MOGP algorithm outperforms the neural network with a mean objective value of 1.6 compared to 12.3 for the neural network.

7. DISCUSSION AND CONCLUSIONS

This paper has demonstrated, using two examples, how performance with respect to additional modelling criteria can be improved using a MOGP algorithm. A disadvantage of using the correlation tests is that the additional processing time becomes significant as the number of process inputs increases. This is because the cross correlation tests have to be repeated for each process input. One alternative is to use tests based only on the residuals and the process output. Billings and Zhu (1994) demonstrated how such tests could be used to reduce the number of correlations that have to be evaluated.

While GP may not provide a significant increase in model accuracy when compared to more established techniques such as neural networks, the algorithm has more of an advantage when applied to multi-objective problems. The parallel nature of GP means that the algorithm can evolve a set of candidate solutions with varying levels of performance in each objective. Real world engineering problems typically involve a number of criteria that must be satisfied before a successful solution can be achieved. Consequently, there has been a large increase in the application of MOEAs to engineering problems and it is likely that this trend will continue.

8. REFERENCES

Bettenhausen, K.D., Marrenbach, P.(1995). Self-Organising Modelling of Biotechnological Batch and Fed-batch Fermentations, EUROSIM '95.

Bhat, N. and McAvoy, T.J.(1989).Use of neural nets for dynamic modelling and control of chemical process systems. ACC, 1342 – 1347.

Billings, S. A. and W. S. F. Voon, (1986), Correlation based model validation tests for non-linear models, International Journal of Control. **44**, 235-244.

Billings, S.A. and Zhu, Q.M. (1994). Nonlinear Model Validation Using Correlation Tests. International Journal of Control, **60**(6): 1107-1120

Cao, H., Yu, J., Kang, L., Chen, Yuping, Chen, Yongyan (1999). The kinetic evolutionary modelling of complex systems of chemical reactions. Computers & Chemistry. **23**, 143-152.

Fonseca, C. M., Fleming, P. J., (1995). Multi-objective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction, GALESIA.

Henson, M. A., Seborg, D. E., (1997), Nonlinear Process Control, Prentice Hall.

Hinchliffe, M.P. (2001) Dynamic Modelling using Genetic Programming, PhD Thesis Uni. of Newcastle, UK.

Kulkarni, B. D., Tambe, S. S., Dahule, R. K., Yadavalli, V. K.(1999) Consider Genetic Programming for Process Identification. Hydrocarbon Processing **78**: 89-97.

McKay, B., Willis, M.J., Barton, G.(1997). Steady-state Modelling of Chemical Process Systems Using Genetic Programming. Computers and Chemical Engineering, **21**, 981-996.

McKay, B., Willis, M.J., Searson, D.P., Montague, G.A. (2000) Nonlinear Continuum Regression: an evolutionary approach. Trans. Inst. M.C. **22**, 125-140.

Rodriguez-Vazquez, K., Fleming, P.J.(1998). Multi-objective Genetic Programming for Gas Turbine Engine Model Identification. UKACC CONTROL'98.

Turner, P., Montague, G.A. Morris, A.J. (1996) 'Non-linear and direction dependant dynamic process modelling using neural networks', IEE Process Control Theory and Applications. **143**, 44-48.

Van Veldhuizen, D.A. Lamont, G.B. (2000), Multiobjective Evolutionary Algorithms Analyzing the State-of-the-Art, Evolutionary Computation **8**(2): 125-147