

GREY-BOX SYSTEM IDENTIFICATION WITH VARIABLE-STRUCTURE NEURAL NETWORKS

KARL MATHIA
Renaissance Sciences Corp.
1351 Alma School Road, Suite 265
Chandler, Arizona 85224
kmathia@rscusa.com

ABSTRACT

'Grey-box' models take advantage of physical insight about the dynamical system in question. This reduces uncertainty and facilitates the remaining system identification process. Here a grey-box system identification (GBSI) method for nonlinear dynamical systems is presented, which first establishes a linear model (the 'white box') using a priori knowledge and conventional tools, and then adds a neural network (the 'black box') for learning the residual nonlinear dynamics. What distinguishes this from other GBSI methods is the Variable Structure Learning algorithm, which trains both the neural network connection weights and structure, i.e. number of hidden neurons. The method is tested with two different neural networks using a nonlinear reference function and a realistic aircraft model.

INTRODUCTION

System identification (SI) refers to the design of a sufficiently accurate model of dynamical systems. 'Grey-box' system identification (GBSI) takes advantage of physical insight about the dynamical system in question, for example the model structure and/or certain parameter values. This reduces uncertainty and facilitates identifying the residual, unknown dynamics. Here a nonlinear GBSI method is proposed, which first establishes a linear model (the 'white box') using a priori knowledge and conventional tools, and then synergetically adds an artificial neural network, or ANN (the 'black box'), for learning the residual nonlinear and possibly unmodelled linear dynamics. What distinguishes this from other GBSI methods, for example [1][5][8][12], is the Variable Structure Learning algorithm, which not only trains the ANN connection weights but also the network structure, i.e. number of hidden neurons [9]. The method is tested with two different ANNs using a nonlinear reference function and a high-performance aircraft model.

The GBSI process involves three steps:

- Create a linear time-invariant model (structure and parameters) based on available a priori knowledge.
- Identify the unknown model parameters using conventional linear SI methods and tools.
- Train an ANN (weights and structure) to model the residuals dynamics.

It is often difficult to choose a good structure for the black-box model, due to the uncertainty about complexity and structure of the unknown dynamics. A black-box component with a self-adapting structure is therefore an appealing concept. Here a Variable Structure Learning neural network automates the network design process, resulting in an ANN structure of sufficient complexity for approximating a given system. A single-layer multilayer perceptron (MLP) provides the basic network structure, and the number of hidden units is adjusted to achieve the desired accuracy [9]. In a sense this realizes the well-known universal approximator theorem, which proves that a single-layer MLP can approximate most functions to any degree of accuracy, provided a sufficient number sigmoidal hidden neurons [2][4]. The performance of the neural GBSI with an MLP is compared to that with a radial basis function (RBF) network as the black-box model.

GREY-BOX SYSTEM IDENTIFICATION

Framework. A dynamical system is a physical process exhibiting a certain relationship between input (controllable external signals), noise (uncontrollable external signals), and output (observable signals). See Figure 1. The output of a dynamical system is a function not only of current but also past inputs, and possibly past outputs. The resulting dynamics can be complex, and there is no standard SI recipe. Instead, SI is a repetitive experimental process that requires [6][7]

- observed input/output data;
- a set of candidate models;
- a model adaptation algorithm.

SI generally includes two main steps: model structure selection and parameter specification. For the GBSI method four steps are needed, two per model, although ANN training automates the last two.

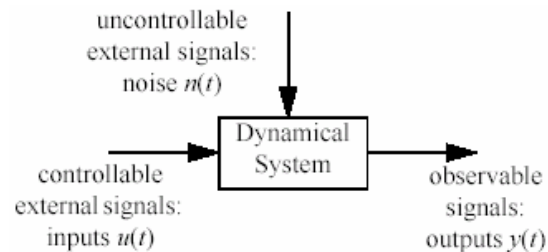


Figure 1. Dynamical system and input/output signals.

System Identification Problem. The system identification problem is usually defined as follows. Given the time series of past observations of input and output signals,

$$u_t = [u(1), u(2), \dots, u(t-1), u(t)] \quad (1)$$

$$y_t = [y(1), y(2), \dots, y(t-1), y(t)] \quad (2)$$

find the relationship $g(\cdot)$,

$$y(t+1) = g(u_t, y_t) + e_m(t) \quad (3)$$

between past observations (u_t and y_t) and future outputs $y(t+1)$. Note that u and y are often vector-valued. The SI objective is to achieve the desired model accuracy by minimizing the residual model error $e_m(t)$.

Multilayer Perceptron and Variable Structure Learning. The well-known Multilayer Perceptron (MLP) with one hidden layer is illustrated in Figure 2. The MLP implements a nonlinear mapping from input vector x to output vector y (the latter is a scalar in Figure 2):

$$y = g(a^o) = g(W g(a^h) + w) = g(W g(Vx + v) + w). \quad (4)$$

Here $g(\cdot)$ is a vector of neuron transfer functions (hyperbolic tangents), V and W are connection weight matrices, v and w are bias weight vectors, and a^h and a^o are the activation values of hidden and output neurons, respectively.

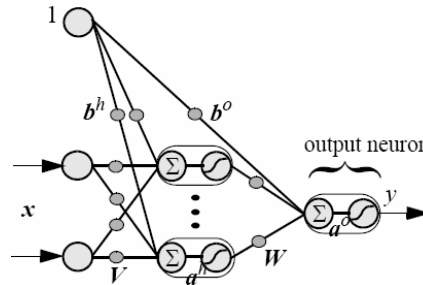


Figure 2. Multi-layer perceptron with one hidden layer. Neurons are shown as summation and transfer function nodes.

The VSL algorithm combines both constructive and pruning concepts for neural network training. Utilizing the popular error backpropagation algorithm, it sequentially adjusts the connection weights and then adjusts the MLP structure by adding hidden neurons until a prespecified success criterion, i.e. the output error limit E_{min} , is reached. Finally the 'least significant' hidden neurons are removed while maintaining that success criterion. VSL generally converges in both weight space and structure (function) space. Five VSL rules determine if and when to add or prune hidden neurons. The rules are based on the VSL metric, the usual mean squared error E and its rate of change [9].

- **Rule 1:** the constructive phase is in progress if E is greater than a prespecified minimum E_{min} and if decreasing. Action: continue weight training.

- **Rule 2:** the ANN structure is insufficient or weight training is stuck in a local minimum if E converges to a value greater than E_{min} . Action: add neuron to hidden layer.
- **Rule 3:** constructive VSL phase is successful if E drops below the desired minimum value E_{min} . Action: terminate constructive VSL phase and initiate VSL pruning phase.
- **Rule 4:** the pruning phase is in progress if $E < E_{min}$. Action: prune the least significant neuron and re-train all weights.
- **Rule 5:** the pruning phase is completed when E increases to $E > E_{min}$. Action: restore last ANN with $E < E_{min}$ and terminate VSL.

NEURAL GREY-BOX MODELS

The proposed neural GBSI method demonstrates that ANNs are well-suited for nonlinear system identification (SI). The method is based on conventional grey-box SI, which maximizes the usage of a priori knowledge and physical insight about the dynamical system in question. An ANN black-box model is used to identify the residual dynamics.

Model Architecture. The architecture of a neural grey-box model (GBM) fm is shown in Figure 3, defined by

$$\mathbf{y}_m(t) = \mathbf{f}_m(\mathbf{x}_{LTI}(t), \mathbf{u}(t)) \quad (5)$$

$$= \mathbf{f}_{LTI}(\mathbf{u}(t)) + \mathbf{f}_{ANN}(\mathbf{x}_{LTI}(t), \mathbf{u}(t)) \quad (6)$$

$$= \mathbf{y}_{LTI}(t) + \mathbf{y}_{ANN}(t) \quad (7)$$

The goal is to reduce the GBM error

$$\mathbf{e}_m(t) = \mathbf{y}(t) - \mathbf{y}_m(t). \quad (8)$$

to the desired model accuracy. The GBM includes the linear-time invariant (LTI) model component \mathbf{f}_{LTI} , and the ANN component \mathbf{f}_{ANN} . The input $\mathbf{u}(t)$ is applied to both the unknown nonlinear system \mathbf{f} and GBM. The state vector \mathbf{x}_{LTI} is an additional input to \mathbf{f}_{ANN} . The overall goal is to minimize the model error $\mathbf{e}_m(t)$. In general all signals and functions are vector-valued, indicated by bold letters.

The linear model \mathbf{f}_{LTI} has the usual form [11]

$$d\mathbf{x}_{LTI}(t)/dt = \mathbf{A} \mathbf{x}_{LTI}(t) + \mathbf{B} \mathbf{u}(t) \quad (9)$$

$$\mathbf{y}_{LTI}(t) = \mathbf{C} \mathbf{x}_{LTI}(t) + \mathbf{D} \mathbf{u}(t). \quad (10)$$

The free SI parameters are \mathbf{A} and \mathbf{B} . All states \mathbf{x}_{LTI} are observable, i.e. $\mathbf{C} = \mathbf{I}$ (identity matrix). Most physical systems are proper, thus we assume $\mathbf{D} = 0$. Note: ANN input and output can be scaled down and up, respectively, to an appropriate range.

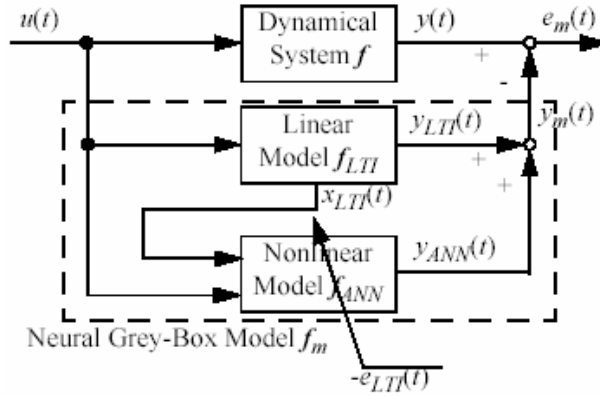


Figure 3. Neural grey-box system identification.

GBM Design Process. The SI process, i.e. the GBM design process, includes the steps summarized in Section I. First, using available *a priori* knowledge, the structure for f_{LTI} is chosen. Second, also based on physical insight, the system parameters A and B are initialized where possible. Due to model imperfections the linear model error will be

$$e_{LTI}(t) = y(t) - y_{LTI}(t). \quad (11)$$

Third, f_{ANN} is connected in parallel with f_{LTI} and trained to reduce the linear model error by approximating $-e_{LTI}$:

$$y_{ANN}(t) = -y(t) + y_{LTI}(t) = -e_{LTI}(t). \quad (12)$$

This is indicated in Figure 3. The ANN is trained with observed input/output data pairs while the already completed linear model is operating. The GBSI steps 1 through 3 may be repeated until the desired model accuracy is accomplished.

Linear models are sometimes established at several operating points and then scheduled depending on the operating conditions. In classical linear SI this approach approximates linear time-varying (LTV) systems, where the number of operating points determines the LTV model accuracy. Model ‘scheduling’ is a well-known tool in the linear case [11]. The proposed GBSI method can reduce the number of necessary operating points without compromising model accuracy. If the operating conditions change significantly as is often the case, for example, in aerospace applications, a GBM can also be designed at selected operating point. The scheduling of nonlinear time-invariant GBMs simulates nonlinear time-variant systems.

Comparison with RBF Networks. The GBM performance with MLP is compared against GBMs with Radial Basis Function (RBF) networks. While the MLP structure is adjusted during VSL, the RBF network structure was preselected and fixed during training. RBF networks are known as good function

approximators in high-dimensional spaces. Their feedforward architecture comprises three completely different layers: an input layer with source nodes, a high-dimensional hidden layer, and a linear output layer. The hidden layer implement a nonlinear vector function $\mathbf{b}(\cdot)$, where each node processes a subset of the input vector using a radial basis functions $\mathbf{b}(\cdot)$. Each output node computes the inner product of $\mathbf{b}(\cdot)$ and the weight vector \mathbf{w} . For the application in Figure 3 the RBF network output is given by [3]:

$$\mathbf{y}_{\text{RBF}}(t) = \mathbf{f}_{\text{RBF}}(\mathbf{u}, \mathbf{x}_{\text{LTI}}) = \mathbf{w}^T \mathbf{b}(\mathbf{u}, \mathbf{x}_{\text{LTI}}). \quad (13)$$

Note that each function in $\mathbf{b}(\cdot)$ processes only a specified ‘window’ of the (input) domain. Generally RBF networks offers two learning variables, i.e. the centers of the radial basis functions (centroids) and the connection weights [3]. Here the centroids were fixed.

EXAMPLES

Two examples evaluate and demonstrate the proposed GBSI method, a scalar, analytical function and a realistic dynamical model of a F4 jet fighter.

Scalar Function. Nonlinear system: here the dynamical system f in Figure 3 is governed by the difference equation [10]

$$x_{k+1} = f(x_k, u_k) = a_1 x_k / (1 + a_2 x_k^2) + u_k^3, \quad (14)$$

with k representing discrete time. Equation 14 is our ‘*a priori knowledge*’ about the system in question, with $a_1=a_2=1$. The ‘real’ system used for simulation has $a_1=0.9$ and $a_2=1.1$. The control signal is

$$u_k = \sin(T_S 2\pi k/25) + \sin(T_S 2\pi k/10), \quad (15)$$

where T_S is the sampling period (absent in [10]).

Linear model: the linear model f_{LTI} (Figure 3) is

$$x_{k+1} = A_k x_k + B_k u_k, \quad k = 1, 2, 3, \dots \quad (16)$$

$$y_k = x_k \quad (17)$$

The system parameters A and B are established at several operating points x^* and u^* using the usual partial derivatives:

$$A^* = df(x_k, u_k)/dx = (1 - x_k^2)/(1 + x_k^2), \quad \text{at } x_k = x^*, \quad (18)$$

$$B^* = df(x_k, u_k)/du = 3 u_k^2, \quad \text{at } u_k = u^*. \quad (19)$$

Model evaluation: the GBM was evaluated with a VSL-trained MLP (Figure 4) and a fixed-size RBF network (Figure 5). The figures show that the RBF performs slightly better, which could be of interest for certain applications. The better approximation performance comes at a higher computational cost: the RBF network needs significantly more neurons, and requires more computational resources per neuron.

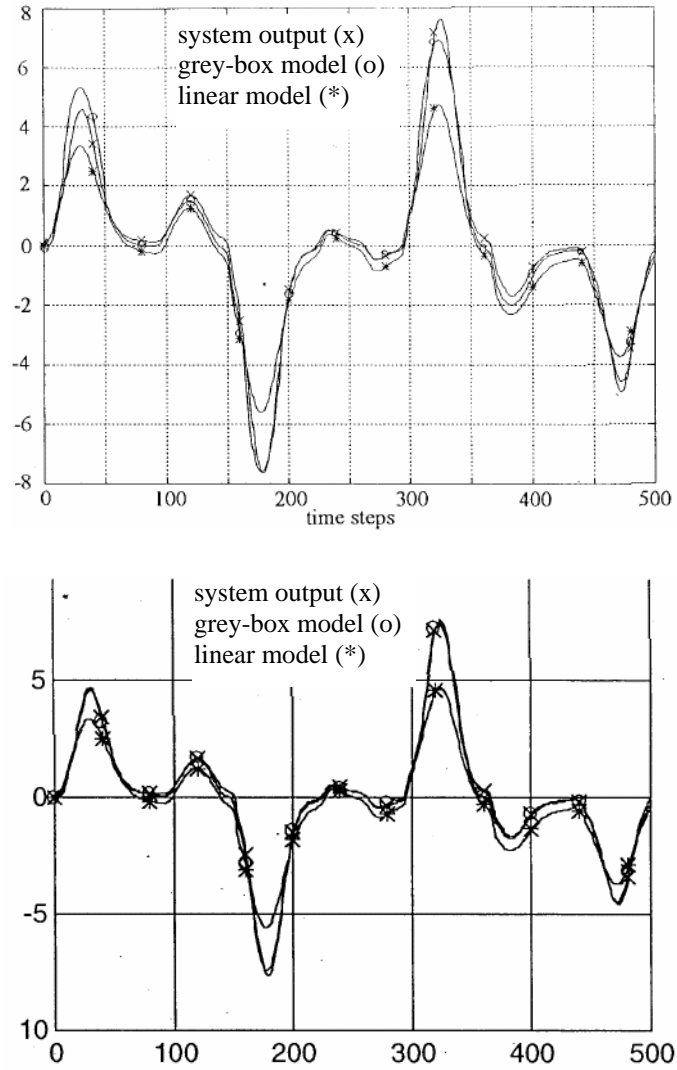


Figure 5. GBM test with an fixed-size RBF network.

High-Performance Aircraft. Nonlinear system: here the dynamical system f in Figure 3 is a nonlinear dynamical model of a high-performance F4 fighter jet. A common dynamical model for subsonic and transonic fixed-wing aircraft was used, which consists of 35 linear and nonlinear equations for simulating the complete six degrees of freedom (DOFs) dynamics. The proposed GBSI method was evaluated by modeling the (most challenging) lateral aircraft dynamics, i.e.

2 DOFs with the four system states lateral velocity, roll angle, roll angle rate, and side slip angle. The two control inputs were aileron and rudder angles.

Linear model: the linear model f_{LTI} (Figure 3) has the structure as shown in Equation 9 and Equation 10, where the state vector $x \in \mathcal{R}^4$ and $u \in \mathcal{R}^2$ and the matrices A , B , C , D have appropriate dimensions.

Model evaluation: Figure 6 shows the actual, simulated and GBM dynamics of four states: lateral velocity [knots], roll angle [deg], roll angle rate [deg/s], side slip angle [deg]. The x-axis represents continuous time in seconds. The figure shows that the neural GBM produced results nearly identical with the F4 reference model and emphasizes a superior performance over the linear model in particular for the roll angle.

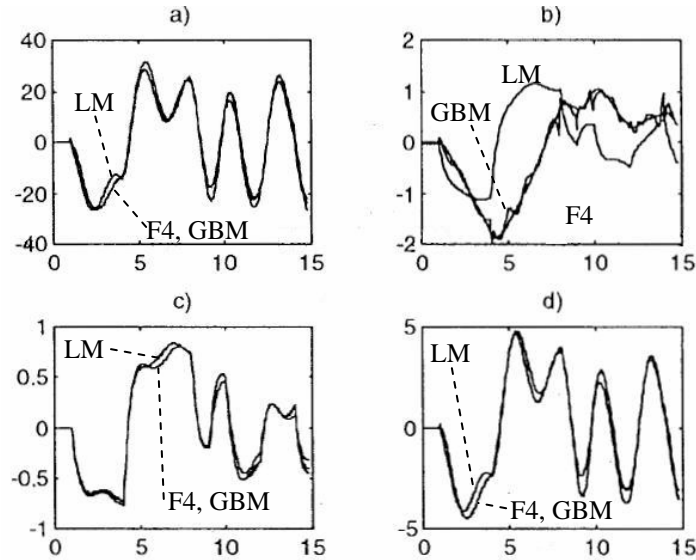


Figure 6. GBSI of F4 aircraft using linear model (LM) and grey-box model (GBM): a) lateral velocity, b) roll angle, c) roll angle rate, d) side slip angle.

CONCLUSIONS

A neural grey-box system identification (GBSI) method for nonlinear dynamical systems was proposed, which first establishes a linear model (the ‘white box’) using a priori knowledge and conventional tools, and then adds a neural network (the ‘black box’) for learning the residual nonlinear dynamics. What distinguishes this from other GBSI methods is the Variable Structure Learning algorithm used for neural network training. We conclude that the proposed method

- is suitable for modeling a variety of nonlinear dynamics;

- works best when the linear model is established separately and prior to the black-box model;
- is generally able to approximate many nonlinear systems to the desired degree of accuracy;
- has been shown to converge to an appropriate network size and connection weights for approximating a system to a desired degree of accuracy.

As such, neural GBSI is suitable for various applications, and is a primary candidate for integration into adaptive neurocontrol system architecture.

REFERENCES

- [1] C. Archambeau, J. Delbecq, C. Veraart, M. Verleysen, "Prediction of visual perceptions with artificial neural networks in a visual prosthesis for the blind," *Artificial Intelligence in Medicine*, Vol. 32, pp.183-194, Elsevier, 2004.
- [2] G. Cybenko, "Approximation by Superposition of A Sigmoidal Function," *IEEE Transaction on Information Theory*, Vol. 39, pp. 930-945, 1989.
- [3] S. Haykin, *Neural Networks - A Comprehensive Foundation*, IEEE Press/Macmillan College, New York, 1994.
- [4] K. Hornik, M. Stinchcombe, H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359-366, 1989.
- [5] G.G. Lendaris and K.Mathia, "Using A Priori Knowledge to Prestructure ANNs", *Australian Journal of Intelligent Information Processing Systems*, Vol. 1, No. 1, pp. 25-30, May 1994.
- [6] L. Ljung, *System Identification - Theory for the User*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.
- [7] L. Ljung, *System Identification Toolbox User Guide*, The MathWorks, Natick, Massachusetts, 1991.
- [8] L. Ljung and J. Sjoberg, "A System Identification Perspective for Neural Networks," *Proceedings of the IEEE Signal Processing Workshop*, pp. 423-435, 1992.
- [9] K. Mathia, "A Variable Structure Learning Algorithm for Multilayer Perceptrons," published in C.H. Dagli et al. (editors), *Intelligent Engineering Systems Through Artificial Neural Networks*, Vol. 14, pp. 93-100, ASME Press, New York, 2004.
- [10] K. Narendra and K. Parthasaraty, "Identification and Control of Dynamical Systems using Neural Networkd," *IEEE Transactions on Neural Networks*, March 1990.
- [11] K. Ogata, *Modern Control Engineering*, Prentice-Hall, Upper Saddle River, New Jersey, 1997.
- [12] J. Sjoberg et al., "Nonlinear Black-Box Modeling in System Identification," *Automatica*, Vol. 31, No. 12, pp. 1691-1724, 1995.