Proceedings of the 26th Annual International Conference of the IEEE EMBS
San Francisco, CA, USA • September 1-5, 2004

An Object-Oriented Toolbox for Linear and Nonlinear System Identification

David T. Westwick,[†] Robert E. Kearney[‡]
[†]Dept. Elec. & Comp. Eng., Univ. Calgary, 2500 University Dr. NW, Calgary, AB, T2N 1N4, Canada
[‡] Dept. of Biomedical Eng., McGill University, 3775 University St., Montreal, PQ, H3A 2B4, Canada

Abstract— **Nonlinear system identification techniques are often used to construct mathematical models of physiological systems. There is a wide variety of model structures, many of which require specialized techniques for their identification. Unfortunately, this variety may present a significant barrier to any non-specialists who wish to use these methods. This paper describes a MATLAB^{TM} based toolbox that addresses these problems by providing a common environment supporting a robust set of tools for the identification, simulation, and manipulation of important model structures. The design philosophy and its implementation are described. An illustrative example, constructing a parallel cascade model of the dynamic stiffness of the human ankle, is used to demonstrate many of the features of the toolbox.**

Keywords— **MATLAB, physiological systems, block structured models, Volterra series, Wiener kernels.**

I. Introduction

System identification embodies the construction of mathematical models of dynamic systems based on measurements of their inputs and outputs. It originated as a discipline in control engineering, where low-order parametric linear models are often used to approximate the plant to be controlled, as this usually leads to relatively simple controller designs. Slow time variations and weak nonlinearities in the plant can often be handled using adaptive methods.

System identification is also used in biomedical engineering. In the study of physiological systems, the goal is more often to construct a model that is detailed enough to provide insight into how the system operates. Since many physiological systems are highly nonlinear, physiological system identification often involves models that are nonlinear and/or time-varying.

Physiological systems are often represented using some type of Volterra system[1], i.e. a model that can be represented by a Volterra series, since these models can represent a very large class of systems.

Methods for the identification of models in the Volterra class continue to be developed. The first practical methods, known collectively as "white noise analysis" [2], were based on Wiener's orthogonalization of the Volterra series, that assumed a Gaussian, white-noise input. The development of efficient exact least squares solutions, summarized in [3], eliminated the requirement for a white noise input. As with the Wiener kernel methods, these techniques are limited to relatively low-order nonlinearities. Higher-order nonlinearities are more readily modeled using block structured models: interconnections of dynamic linear systems and memoryless nonlinearities[4]. Unfortunately, each separate block structure requires its own identification method. To further complicate matters, many model structures have several different identification methods (see [3] for a summary).

Supported by the Natural Sciences and Engineering Research Council

This wide variety of model structures and identification methods provides the investigator with an extensive toolkit. Unfortunately, this richness may also hinder the adoption of nonlinear system identification methods in the wider experimental community. Indeed, the vast majority of articles that report using any given nonlinear system identification technique include the method's originator among the authors. This suggests that these techniques are only being used by those that developed them.

We suggest that one of the factors contributing to this lack of widespread adoption is the complexity of the methods themselves. Until recently, the only references describing these methods were in the research literature, where the methods were presented separately, each using a separate notation, and often with subtle differences in terminology. We have attempted to address this in [3], by presenting several, most recently developed, nonlinear system identification methods using a common notation, terminology, and theoretical framework.

Although [3] presents the theoretical derivations in a unified manner, it does not address the problems inherent in actually using the algorithms. Each algorithm is controlled by one or more tuning parameters. Successful identification often depends on setting one or more of these parameters correctly. This plethora of tuning parameters, and the variability between different approaches, presents a high barrier to entry. Thus, the practice of nonlinear system identification is usually left to the specialists that developed the methods.

In this paper, we will describe a MATLAB^{TM} (The MathWorks Inc., Natick MA) based toolbox designed for nonlinear system identification. Our goal in designing the toolbox was to provide implementations of the identification algorithms described in [3] with a consistent user interface. Additionally, the toolbox should provide reasonable default settings for all tuning parameters, while giving users the ability to modify any of these parameters, and easy access to the information necessary to do so intelligently. Section II will provide an overview of the design of the NLID toolbox and its user interface. Section III will demonstrate a typical application of this toolbox to experimental data. Section IV will summarize the the paper.

II. The NLID Toolbox

The key philosophy underlying the toolbox is that each signal and each model should be an object that contains all of the information relevant to its creation. Thus, a signal will have both domain and range values. A model will contain both the elements of the model, complete with their dimensions and characteristics, and information regarding the method used to identify the model, together with the settings of any tuning parameters required by the identification algorithm. Each standard operation has a single syntax that is independent of the type of object that it applies to. However, since the objects contain information regarding their contents, the result of applying an operation will depend on the object type. Plotting a signal produces a plot of the signal, plotting a block structured model results in plots of all of the elements in the model arranged like a block diagram of that model.

A. NLID *Toolbox Classes*

The NLID toolbox has been implemented using the object oriented programming (OOP) framework provided by MATLAB. As detailed below, the design of the toolbox takes advantage of several key features of OOP: encapsulation, inheritance and method overloading.

The NLID toolbox defines classes, listed in Table I, that describe signals or system models. Objects in these classes are data structures comprising several labeled fields, termed properties. Several properties are used to specify the object's domain: the domain name, units, increment and starting value. For a signal object, these could be time, milliseconds, the sampling period, and 0, respectively. Similarly, the objects have several properties that describe the data: the data values, their units, and the dimensions of the data matrix.

In addition to these properties, objects that represent system models have a "parameter" property which contains the information used to control their identification. The first of these parameters identifies algorithm used to identify the model. This is followed by a list of parameters describing the model structure, which is independent of the identification algorithm. For a Volterra series, these would include the memory length and the maximum kernel order. Finally, there is an algorithm dependent list of parameters. If the identification method is changed, these are replaced with a list of parameters appropriate to the newly selected algorithm.

B. *Standard Operations*

A set of standard NLID toolbox methods, summarized in Table II, are overloaded for all model classes. In addition, many standard MATLAB operators, including sub-array referencing techniques, basic arithmetic operations and data plotting functions have been overloaded so that they can operate on objects in NLID defined classes. An example demonstrating their application is provided in Sec. III.

C. *Flexibility*

One of the design goals underlying the NLID toolbox is the desire for flexibility. Thus, adding new model types, and adding new identification algorithms for existing model types, is relatively easy. For example, adding a new identification algorithm requires only slight changes to the corresponding class and its `nlident` method:

1. Add an option for the new algorithm to the object's `method` parameter.
2. Add code that appends any algorithm specific tuning parameters to the object's parameter list, but only when the new method is selected.
3. Write a wrapper around the new algorithm that extracts the identification data from `nldat` objects, and that extracts the identification parameters from the model object.
4. Modify the `nlident` method so that it transfers control to the new algorithm when it has been selected.

Adding a new model structure is somewhat more involved. However, due to the modular nature of the toolbox, either operation can be done without affecting the functioning of the toolbox as a whole. In particular, backward compatibility will generally be maintained.

TABLE I
SUBSET OF THE CLASSES PROVIDED BY THE NLID TOOLBOX.

Class	Description
`irf`	Linear impulse response
`lnbl`	Wiener (LN) system
`lnlbl`	LNL model
`narmax`	NARMAX model
`nlbl`	Hammerstein (NL) system
`nldat`	Generic signal class.
`polynom`	Polynomial nonlinearity
`randv`	Realization of a random variable or stochastic process
`signalv`	Deterministic signal
`vkern`	Volterra kernel
`vseries`	Volterra series – contains one or more vkern objects.
`wbose`	Wiener-Bose model
`wkern`	Wiener kernel
`wseries`	Wiener series – contains one or more wkern objects.

D. *Availability*

The NLID toolbox is free software, as it has been released under the Free Software Foundation's General Public License (GPL). It can be downloaded from:
`www.bmed.mcgill.ca/reklab/nlid_tools`.

III. Example

This section presents an application of the NLID toolbox to data from a previous study [5], which contains a description of the experimental protocol. Briefly, subjects lay supine with their left foot attached to a rotary hydraulic actuator by a custom fitted fiber-glass boot. A broadband position signal

TABLE II

OVERLOADED METHODS PROVIDED BY THE NLID TOOLBOX.

Method	Description
`nlident`	Identify a model from input/output data.
`nlmtst`	Run a test/demo script using a given model type
`nlsim`	Simulate the response of a system to a given input.
`nlid_resid`	Plots error in model output
`smo`	Smooth a signal or nonparametric model
`vaf`	Compare two models/signals and report percent variance accounted for

was applied using an electro-hydraulic actuator configured as a position servo. Ankle torque, position, and the EMG over the Tibialis Anterior muscle were recorded. Fig. 1 shows 4 seconds of the position and torque measurements.

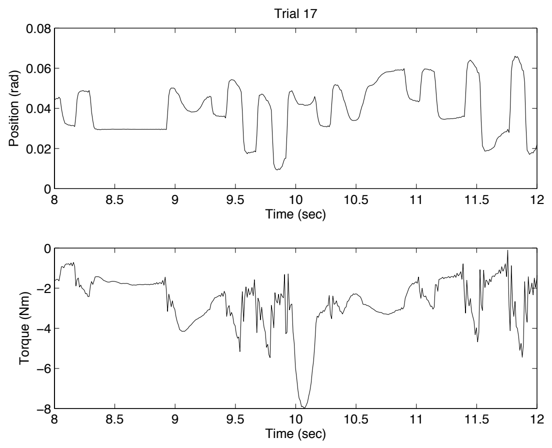

Fig. 1. Extract from a typical experimental trial showing 4 seconds of position and torque.

As has been shown previously [6], the system can be modeled using a parallel cascade structure comprising 2 paths: an a-causal linear system representing the intrinsic stiffness of the ankle, and a differentiator followed by a Hammerstein system, representing the stretch reflex.

Initially, the position and torque signals were sampled at a frequency of 100 Hz, and stored in vectors `pos` and `trq`, respectively. The first step in the analysis is to place the data in `nldat` objects.

```
>> Pos = nldat(pos,'Domainincr',0.01);
>> Torque = nldat(trq,'Domainincr',0.01);
```

The domain of the objects named `Pos` and `Torque` has an increment of $1/100$ seconds, representing the 100 Hz sampling rate. The sampling rate should be specified since it is used to scale the output of all dynamic systems: impulse responses, Volterra and Wiener kernels, etc. Other information may be included as well. For example:

```
>> set(Pos,'comment','Trial 17',...
       'ChanNames',{'Input Position'});
```

These properties do not affect any computations, but they can be used for record keeping and will appear when the data are displayed or plotted.

An initial estimate of the intrinsic pathway can be obtained by fitting a 2-sided IRF, whose memory length is shorter than the 40 ms delay expected in the reflex pathway, between the position and torque.

```
>> IdData = cat(2,Pos,Torque);
>> h1 = irf(IdData,'nsides',2,'nlags',3);
```

The object `IdData` is the concatenation of `Pos` and `Torque`, and is used as the input/output data in the subsequent fitting operation. The identified IRF is specified to be 2-sided, with a memory length of ± 3 samples (i.e. ± 30 ms).

This initial estimate of the intrinsic stiffness IRF is shown in Fig. 2, which was generated by the `plot` operator:

```
>> plot(h1);
```

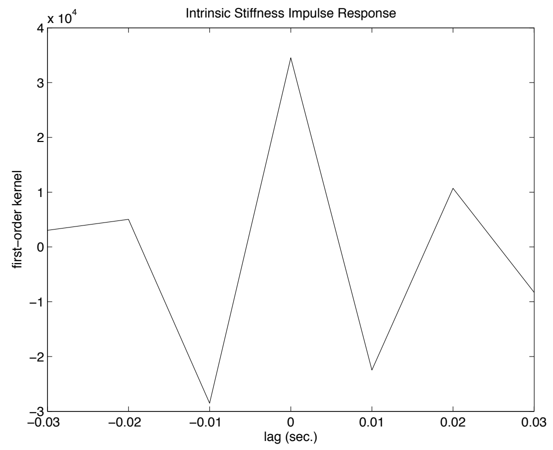

Fig. 2. Initial estimate of the IRF of the intrinsic ankle stiffness.

An initial estimate of the reflex pathway can then be fitted between the ankle velocity, and the residue remaining after the output of the intrinsic pathway has been removed from the torque record. First, compute the ankle velocity,

```
>> Vel = ddt(Pos);
```

and the residue,

```
>> TqInt = nlsim(h1,Pos);
>> Resid = Torque - TqInt;
```

Finally, identify a Hammerstein model between the velocity and the residue.

```
>> IdData = cat(2,Vel,Resid);
>> reflex = nlbl(IdData,'nlags',50,...
      'ordermax',8);
```

The syntax used to identify the Hammerstein model (an object of type `nlbl`, or a nonlinear-linear block structure), is essentially the same as that used to identify the IRF that was used to model the intrinsic stiffness. The only difference is in the selection of options. Since all of the options have "sensible" default settings, it is often possible to get a reasonable initial estimate simply using the syntax:

```
>> reflex = model_type(IdData);
```

where `model_type` is the model structure being fitted to the data in `IdData`.

Note that the `nlbl` model includes a `method` parameter, that selects the identification algorithm used to fit the model. To get a list of available methods, and of the tuning parameters for the currently selected method, use the command:

```
>> get(reflex,'parameters')
  Param object
  Method = sls
       Description: Identification Method
       Values: [  hk, sls ]
.......
```

This is followed by a list of parameters that will be used in the next identification: the memory length of the linear element, the order of the nonlinearity, and several parameters specific to the separable least squares (SLS) algorithm used to identify the model [7]. Setting the method property to `hk` selects an iterative algorithm [4] instead of the default SLS. The list of tuning parameters, and their acceptable ranges and default settings, would then include the parameters required by the `hk` algorithm.

As before, the identified model, as shown in Fig. 3, may be displayed with

```
>> plot(reflex);
```

Note that this is exactly the same command that was used to plot both the data, and the stiffness impulse response.

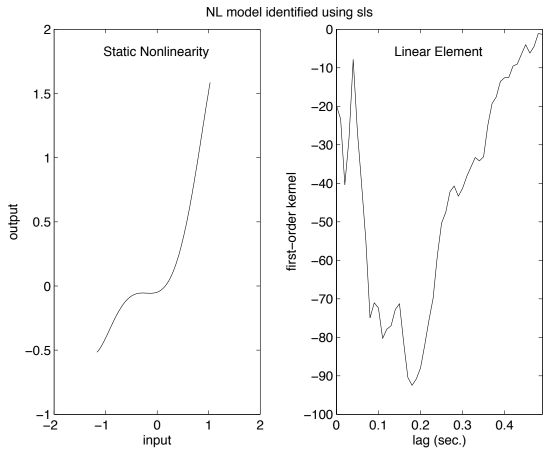

Fig. 3. Initial estimate of the reflex stiffness pathway, modeled as a Hammerstein cascade.

The `nlsim` command, used to compute the output of the linear path, can also be applied to the reflex pathway,

```
>> TqRef = nlsim(reflex,Vel);
```

and then the in-sample accuracy of the parallel cascade model can be evaluated,

```
>> TqTot = TqInt + TqRef;
>> vaf(Torque, TqTot)
```

In practice [6], one would then refine this model iteratively. The reflex contribution would be removed from the torque record, the intrinsic pathway re-estimated, and then used to update the estimate of the reflex pathway. This procedure would be repeated until the model converges. Typically, this will occur within 2-3 iterations.

IV. Conclusions

We have developed an object-oriented toolbox for the identification of nonlinear systems. The object-oriented design simplifies the incorporation of new model structures and identification algorithms. Thus, little maintenance will be required to keep the toolbox abreast of the latest developments.

More significantly, the toolbox provides a consistent interface to a wide variety of model structures and their identification algorithms. Because the model structures encapsulate all of the information relevant to their creation, a common syntax can be used to create, display, simulate, identify and validate any of the model structures included in the toolbox. Novice users can obtain adequate results using the `nlident` method and the default settings for all tuning parameters. More experienced users can modify these parameters and fine-tune the identification procedure. Thus, the toolbox can be used by both novice and experienced users, and may ultimately bring the techniques of nonlinear system identification into more widespread use than is currently the case.

References

[1] Pearson R.K. Doyle III, F.J. and Ogunnaike, *Identification and Control using Volterra Models*, Springer-Verlag, Berlin, 2002.

[2] P.Z. Marmarelis and V.Z. Marmarelis, *Analysis of Physiological Systems*, Plenum Press, New York, 1978.

[3] D.T. Westwick and R.E. Kearney, *Identification of Nonlinear Physiological Systems*, IEEE Press / Wiley, Piscataway, NJ, 2003.

[4] I.W. Hunter and M.J. Korenberg, "The identification of nonlinear biological systems: Wiener and Hammerstein cascade models," *Biol. Cybern.*, vol. 55, pp. 135–144, 1986.

[5] D. Ludvig, M. Baker, I. Cathers and R. E. Kearney, "Task-Dependence of Ankle Stretch Reflex," *Proc. IEEE EMBS Conf.*, vol. 25, pp 1487–1490, 2003.

[6] R.E. Kearney, R.B. Stein, and L. Parameswaran, "Identification of intrinsic and reflex contributions to human ankle stiffness dynamics," *IEEE Trans. Biomed. Eng.*, vol. 44, no. 6, pp. 493–504, 1997.

[7] D.T. Westwick and R.E. Kearney, "Separable least squares identification of nonlinear Hammerstein models: Application to stretch reflex dynamics," *Ann. Biomed. Eng.*, vol. 29, no. 8, pp. 707–718, 2001.