MATLAB Software for Identification of Nonlinear Autonomous Systems – Revision 1

Torbjörn Wigren

Systems and Control, Department of Information Technology, Uppsala University, SE-75105 Uppsala, SWEDEN. E-mail: torbjorn.wigren@it.uu.se.

April 2014

Abstract

This report is intended as a user's manual for a package of MATLAB™ scripts and functions, developed for recursive and batch identification of nonlinear autonomous state space models of order 2. The core of the package consists of implementations of four algorithms for this purpose. There are two least squares batch schemes and two recursive algorithms based on Kalman filtering techniques.

The algorithms are based on a continuous time, structured black box state space model of a nonlinear autonomous system of order 2. The software can only be run off-line, i.e. no true real time operation is possible. The recursive algorithms are however implemented so that true on-line operation can be obtained by extraction of the main algorithmic loops. The user must then provide the real time environment. The software package contains scripts and functions that allow the user to either input live measurements or to generate test data by simulation. The functionality for display of results include scripts for plotting of data and parameters. The estimated model obtained at the end of a run can be simulated and the model output plotted, alone or together with the data used for identification.

Keywords: Identification, Recursive algorithm, Nonlinear systems, Autonomous systems, Periodic signal, State space model, Ordinary differential equation, Software, MATLAB™.

Prerequisites

This report only describes the parts of [1] , [2], [3], [4] and [5] that are required for the description of the software of this package. Hence the user is assumed to have a working knowledge

of the algorithms of these publications and of MATLAB™, see e.g. [6]. This, in turn, requires that the user has a working knowledge of system identification and in particular of recursive identification methods as described in e.g. [7].

Installation

The file SWAutonomous.zip is copied to the selected directory and unzipped. The software is then ready for use.

Note: This report is written exactly with respect to the software, as included in the SWAutonomous.zip file. It may therefore be advantageous to store the originally supplied software for reference purposes.

Error reports

When errors are found, these may be reported in an e-mail to:
torbjorn.wigren@it.uu.se.

1. Introduction

Identification of nonlinear systems is an active field of research today. There are several reasons for this. First, many practical systems show strong nonlinear effects. This is e.g. true for many chemical reactions and for biological systems of many kinds. See e.g. [1], [2], [3], [4], [5] and the references therein for further examples. Another important reason is perhaps that linear methods for system identification are quite well understood, hence it is natural to move the focus to more challenging problems. Furthermore, while there are many identification methods available for identification of nonlinear systems with inputs, there are few identification methods available for identification of nonlinear autonomous systems. Examples of the latter include the algorithms of [2], [3] that are included in this software package and the algorithm of [8] that is based on convex optimization.

This report focuses on the software that implements the nonlinear recursive system identification methods of [2] and [3]. These black box methods estimate continuous time parameters in a general state space model of order two, with a known linear measurement equation. The identification methods belong to the classes of least squares - and Bayesian identification methods. The least squares and Kalman filter algorithms are both exploiting differentiation of the measured signal. These approaches cannot guarantee a resulting model with a stable oscillation - this needs to be checked after the identification has been completed. The extended Kalman filter algorithm embeds a simulated

model of the autonomous system and adjusts the parameters of the model to resemble the data. Therefore, the extended Kalman filter more often results in a model with a stable periodic orbit. On the other hand, the extended Kalman filter is strongly nonlinear and may diverge.

The nonlinear identification algorithms are based on a continuous time black box state space model. This model is structured in that only the second right hand side component of the second order ordinary differential equation (ODE) model is parameterized as an unknown function. This avoids over-parameterization. The restriction imposed on the model structure may seem restrictive. However, it is motivated in [9] that the selected structure can always (locally in the states) model systems with more general right hand sides, a fact that extends the applicability of the method significantly. The selected parameterization of the right hand side function of the ODE is a linear-in-the-parameters multi-variate polynomial in the two states.

Recursive system identification is a software dependent technology. Hence, when publishing new methodology in this field, it is relevant to also provide useful software for application of the presented algorithms. This facilitates a quick practical exploitation of new ideas. The development of the present MATLAB™ software package is motivated by this fact. The software package is developed and tested using a large variety of MATLAB revisions and it does not require any toolboxes. Briefly, the software package consists of scripts for generation or measurement of data, scripts for execution of the algorithms and scripts for generation and plotting of results. There is presently no GUI, the scripts must be run from the command window. Furthermore, input parameters need to be configured in the algorithm scripts and the scripts for data generation and plotting. In case of data generation by simulation, the ODE that defines the data generating system must be specified in standard MATLAB style. The software can only be run off-line, i.e. there is no support for execution in a real time environment. The major parts of the algorithmic loop can however easily be extracted for such purposes.

The report is organized according to the flow of tasks a user encounters when applying the scripts of the package. Before the software is described some basic facts about the ODE model and the scaling method are reviewed.

2. Model

The nonlinear MIMO model to be defined here is used for estimation of an unknown parameter vector from the measured signal

$$z(t) = y(t) + e(t)$$

The starting point for the estimation algorithms is the following second order state space ODE

$$\begin{pmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \end{pmatrix} = \begin{pmatrix} x_2(t) \\ f_2\left(x_1(t), x_2(t), \theta_2\right) \end{pmatrix}.$$

The following polynomial parameterization of the right hand side function is used

$$f_2\left(x_1(t), x_2(t), \theta_2\right) = \sum_{l=0}^{L_2} \sum_{m=0}^{M_2} \theta_{2,l,m} x_1^l(t) x_2^m(t).$$

In order to obtain a discrete time model that is suitable for scaling, the Euler integration method is applied. The result of the discretization is

$$x_1\left(t + T_S\right) = x_1(t) + T_S x_2(t)$$

$$x_2\left(t + T_S\right) = x_2(t) + T_S \sum_{l=0}^{L_2} \sum_{m=0}^{M_2} \theta_{2,l,m} x_1^l(t) x_2^m(t).$$

It can be remarked that the Euler method may require fast sampling in order not to introduce significant discretization errors. This is fortunately a less important effect in system identification applications. The reason is that the minimization algorithm uses the parameters as instruments to fit the model output to the measured data, as expressed by the criterion function. Even if an additional bias would be introduced in the estimated parameters, the input-output properties of the identified model can be expected to describe the data well. Finally, it is noted that above equations can be compactly written as

$$x_1\left(t + T_S\right) - x_1(t) = T_S x_2(t)$$

$$x_2\left(t + T_S\right) - x_2(t) = \varphi_2^T\left(x_1(t), x_2(t)\right)\theta_2$$

$$\frac{1}{T_S}\varphi_2\left(x_1(t), x_2(t)\right) =$$

$$\left(1 \quad \ldots \quad x_2^{M_2}(t) \quad \ldots \quad x_1^{L_2}(t) \quad \ldots \quad x_1^{L_2}(t) x_2^{M_2}(t)\right)^T$$

$$\theta_2 =$$

$$\left(\theta_{2,0,0} \quad \ldots \quad \theta_{2,0,M_2} \quad \ldots \quad \theta_{2,L_2,0} \quad \ldots \quad \theta_{2,L_2,M_2}\right)^T.$$

3. Software package overview

The software package is command driven, i.e. no GUI is available. It consists of a number of MATLAB scripts and functions. These are described in the next subsection.

3.1 Scripts, functions and command flow

The scripts and functions can be divided into three groups:

- *Simulated data generation*. The script of this group defines a dynamic system, that is then used for generation of simulated data. The script is **Generate2DData.m**.
- *Identification*. The scripts of this group perform the actual identification tasks. The scripts and functions are **Scaling.m**, **LeastSquares2D.m**, **LeastSquaresDifferentiated2D.m**, **Kalman2D.m** and **ExtendedKalman2D.m**.
- *Preparation and display of results*. The scripts in this group prepare, compute and display results of the identification process. The scripts are **Plot2DData.m, Generate2DModelData.m**, **IdentifiedModel.m**, **Plot2DModel.m**, **Plot2DModelAndData.m**.

These groups of scripts and functions need to be operated in a particular order to make sense. In case the user intends to use simulated data, this data can be generated by execution of the scripts and functions of the group *Simulated data generation.* The user can then proceed directly to use the groups *Identification* and *Preparation and display of results.*

4. Data input

Since the user has access to all source files, the descriptions below do not describe code related issues and internal variables. Only the parts that are required for the use of the software package are covered. The description builds on the exact source code of the software package. Note that the setup files are to be treated as templates, the user is hence required to modify right hand sides only - no addition or deletion of code should be used in the normal use of the package.

4.1 Simulated data

The generation of simulated data requires that the user:

1. Modifies the underlying ODE model, given by VanDerPool.m in the simulator.
2. Provides further input data in the script **Generate2DData.m**. The parameters that define the data generation are directly written into this script. These parameters define the sampling period, the data length, the dimensions of the system, the type and parameters of the disturbances, as well as the initial value of the ODE.
3. Generates data by execution of **Generate2DData.m**. After the execution of this script, variables with sampling instances and output signals are available in the MATLAB workspace.

Example 1: This and the following examples illustrates the use of the software package for identification of the system

$$\begin{pmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 + 2\left(1 - x_1^2\right)x_2 \end{pmatrix}.$$

The equations describe the Van der Pol oscillator [2]. The MATLAB command window command is

>> Generate2DData

>>

4.2 Display of data

After execution of Generate2Ddata, data can be plotted, by using the **Plot2DData.m** script.

Example 2: The MATLAB command window command is

>> Plot2DData

>>

The following plots are generated:

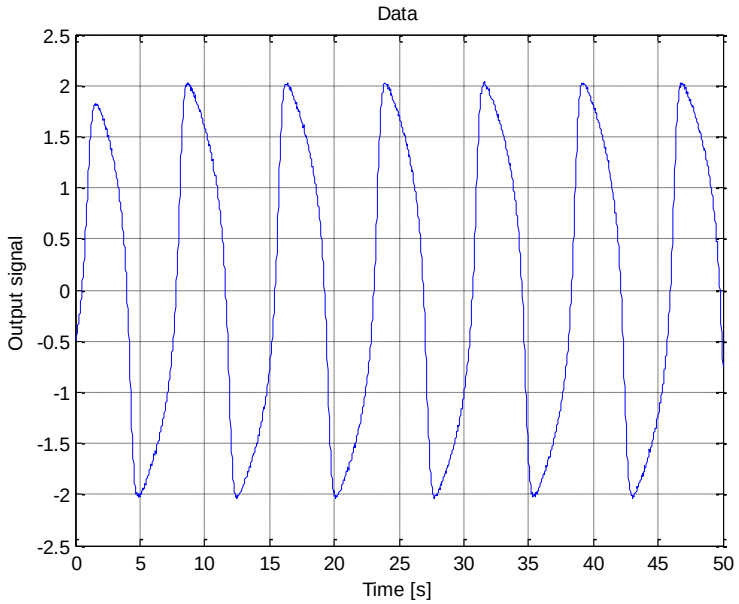

Figure 1: The first result of a Plot2DData command.

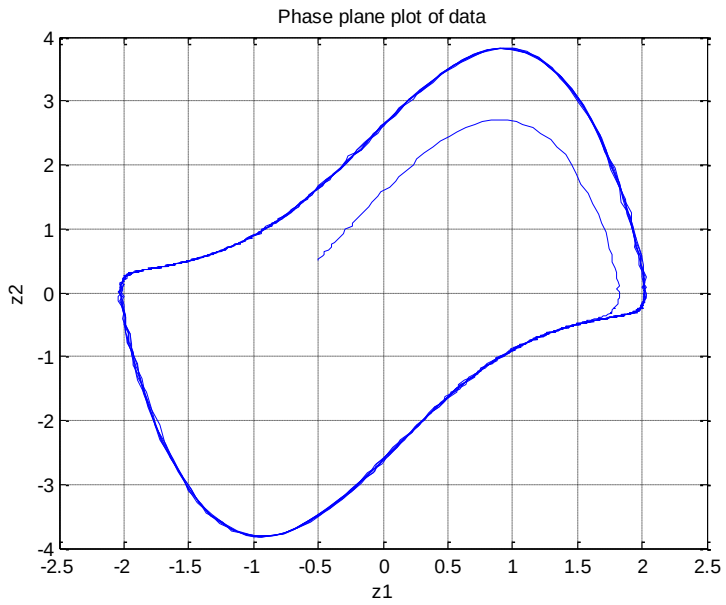

Figure 2: The second result of a Plot2DData command.

4. **Identification**

5.

At this point everything is in place for a first identification run. When describing the procedure the Kalman filter and extended Kalman filter algorithms are used. The procedures for the least squares algorithms are parallel.

5.1 Scaling - optional

As a preparation for the identification run, the user may wish to scale the data in amplitude or time. A detailed discussion on the need for this appears in [5] and [9]. Briefly, it is essential to scale signals so that the nonlinear effect of the model is focused on amplitudes and frequencies of the measured signals. This requires that the user:

1. Modifies the values of timeScaleFactor and dataScaleFactor in **Scaling.m**.
2. Executes the script **Scaling.m**

Example 3: The MATLAB command window command is

```
>> Scaling
>>
```

With timeScaleFactor set to 0.5 and dataScaleFactor set to 2, re-execution of Plot2DData then gives Figure 3 and Figure 4.

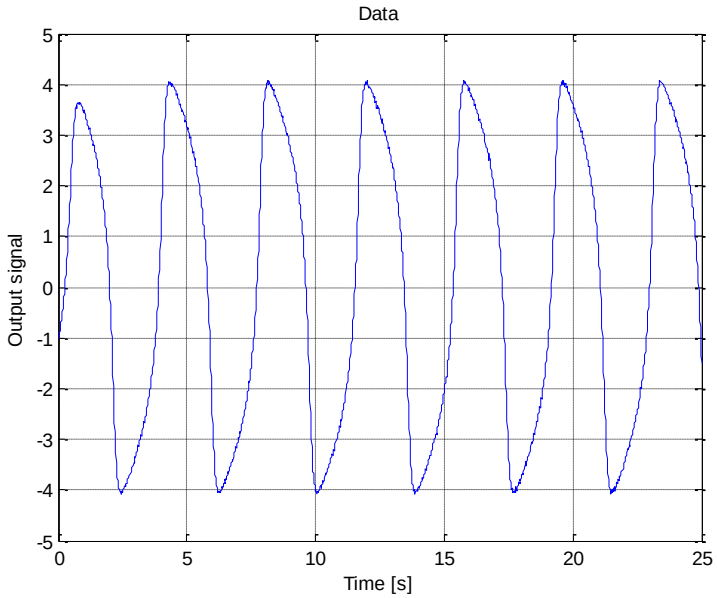

Figure 3: The first result of a Plot2DData command, after scaling.

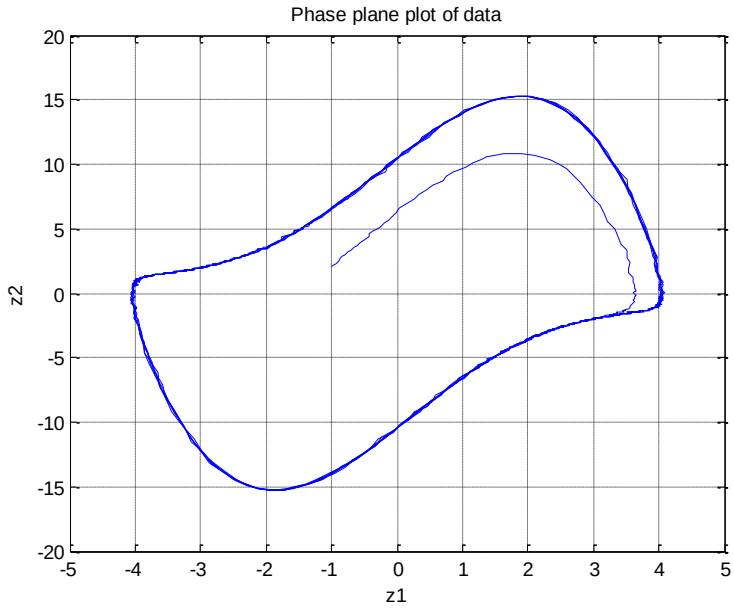

Figure 4: The second result of a Plot2DData command, after scaling. Note that the differentiated signal z2 is affected by both the amplitude and time scaling.

5.2 Recursive identification

The preparation for the identification run requires that the user sets the parameters for the identification algorithm used. This means that one or several of the scripts **Kalman2D.m, ExtendedKalman2D.m, LeastSquares2D.m** or **LeastSquaresDifferentiated2D.m** need to be modified. See [2] and [3] for details. In particular this step requires that the user:

1. Provides further parameter data in the scripts above. These parameters define the dimension of the system, the initial value used in the ODE model, the initial value of the covariance matrix in case of Kalman or extended Kalman filter algorithms, and the values of the systems and noise covariance matrices in case of Kalman or extended Kalman filter algorithms. The reader is referred to [2] and [3] for details on these parameters, as well as on their use.
2. Executes one of the above scripts. This loads the necessary parameters into the MATLAB workspace and performs the system identification task.

Example 4: The Kalman filter algorithm of [2] is assumed to be used here, processing unscaled data according to Figure 1 and Figure 2. To execute the simulation the user executes **Kalman2D.m** in the MATLAB command window, to obtain:

```
>> Kalman2D

theta =

   -0.0085
    1.9812
   -1.0033
    0.0028
    0.0037
   -1.9919

>>
```

The extended Kalman filter algorithm automatically uses theta as initial value, when theta is available. To run the extended Kalman filter scheme, the user executes **ExtendedKalman2D.m** in the command window to obtain

```
>> ExtendedKalman2D

theta =

   -0.0039
    2.0060
   -1.0740
    0.0010
    0.0020
   -2.0812
```

>>

6. Simulation and display of results

6.1 Simulation

First the identified model needs to be simulated using the script **Generate2DModelData.m** . That script in turn, uses the function **IdentifiedModel.m** , which is based on the model structure outlined above and in [2], [3].

Example 5: First unscaled data was re-generated as described in section 4.1. To simulate the identified model the user then executes the following command in the MATLAB command window

```
>> Generate2DModelData
>>
```

6.1 Display of results

In order to plot the parameters the user is required to execute one of the scripts **Plot2DModel.m** or **Plot2DModelAndData.m** in the command window.

Example 6: The command in the MATLAB command window is

```
>> Plot2DModel
>>
```

The following plots are then generated:

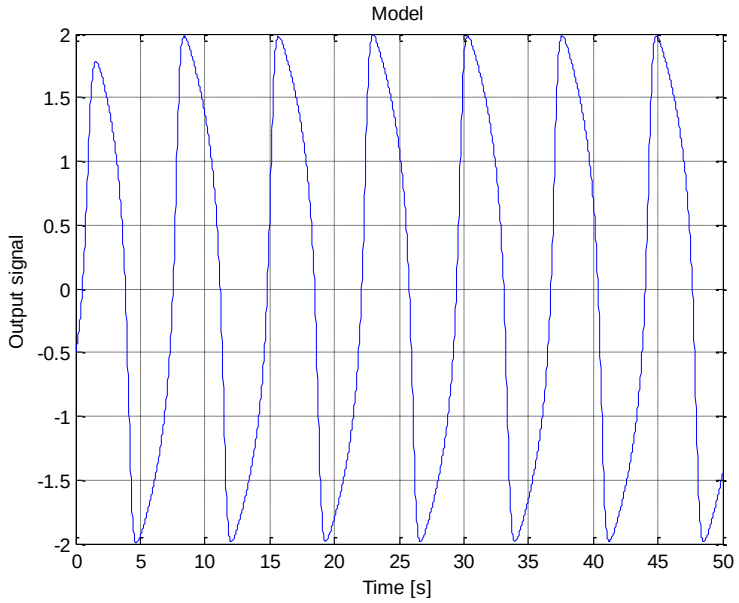

Figure 5: The first result of a Plot2DModelData command.

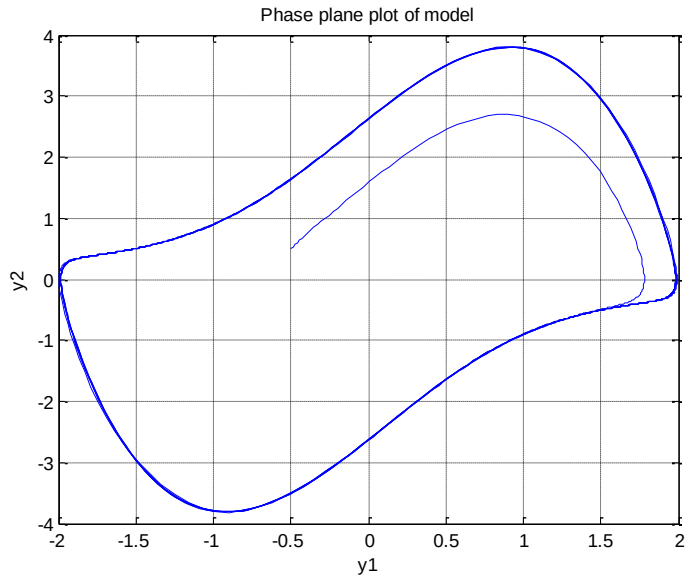

Figure 6: The second result of a Plot2DModelData command.

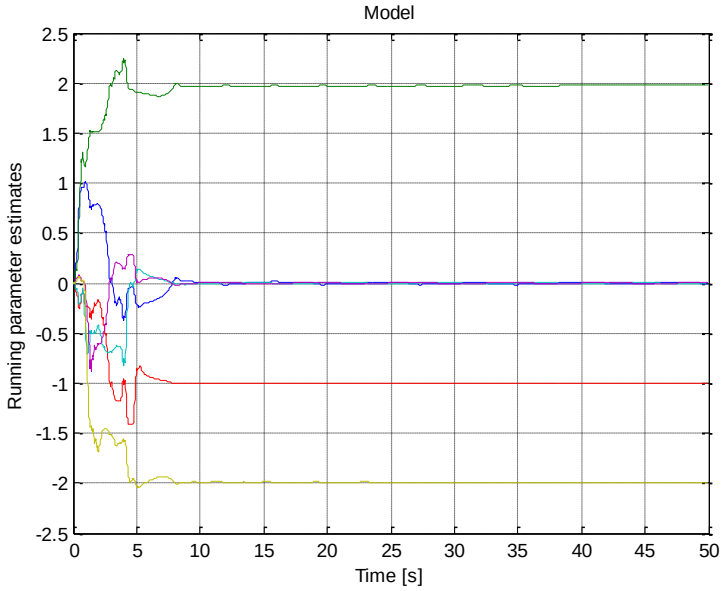

Figure 7: The third result of a Plot2DModelData command.

Example 7: It is also possible to do joint plotting of the data used for identification and the simulated model. The user then executes the following command in the MATLAB command window:

```
>> Plot2DModelAndData
>>
```

This generates the following plots:

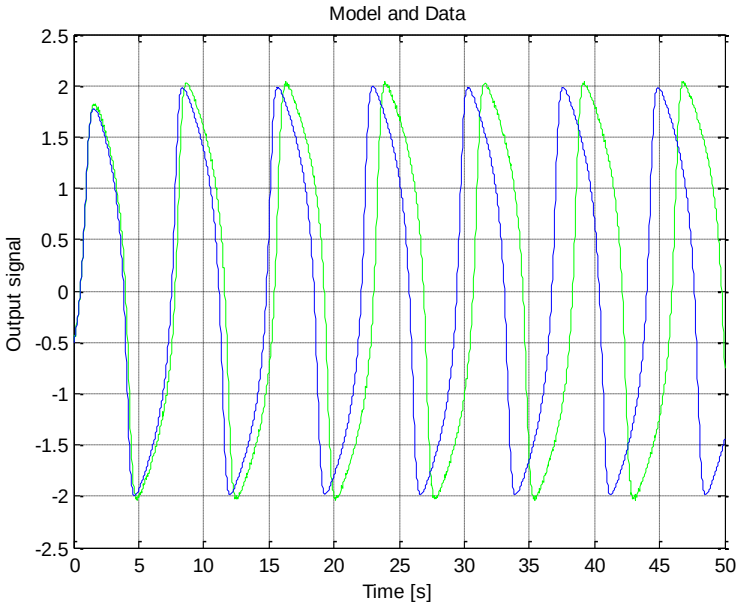

Figure 8: The first result of a Plot2DModelAndData command.

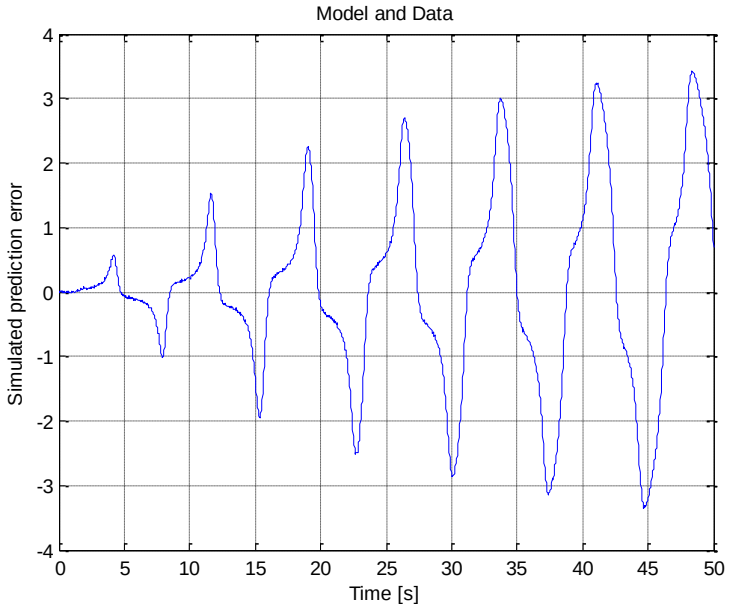

Figure 9: The second result of a Plot2DModelAndData command. The reason why the simulated prediction error is much larger than in Figure 10 is that the Kalman filter is used. That algorithm is not exploiting feedback from a simulated model in the identification process, as does the extended Kalman filter.

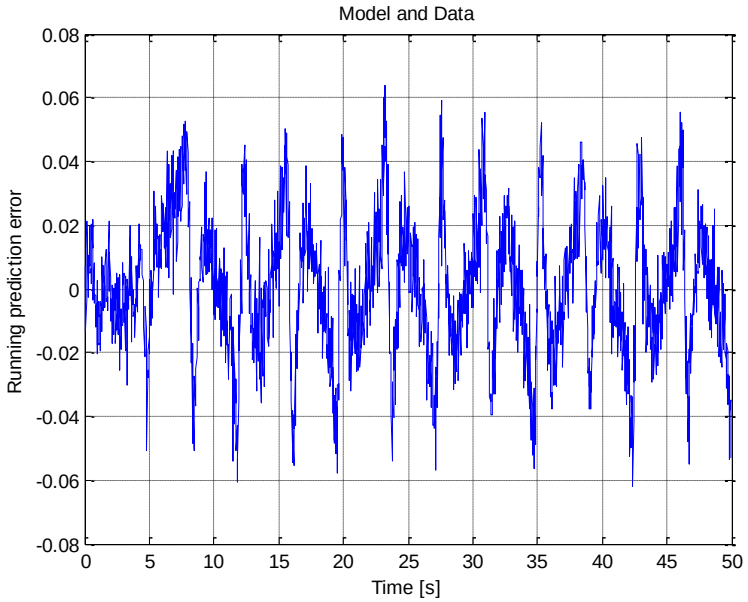

Figure 10: The third result of a Plot2DModelAndData command.

11. Summary

This report describes a software package for identification of nonlinear autonomous systems of order two. Future work in this field, that result in useful MATLAB routines, will be integrated with the presently available functionality. Updated versions of this report will then be made available.

11. References

[1] T. Wigren and T. Söderström, "Second order ODEs are sufficient for modeling of many periodic signals", Technical Reports from the department of Information Technology 025-2003, Uppsala University, Uppsala, Sweden, April, 2003.

[2] T. Wigren, E. Abd-Elrady and T. Söderström, "Harmonic signal analysis with Kalman filters using periodic orbits of nonlinear ODEs", *in Proc. ICASSP 2003*, Hong Kong, China, vol. VI, pp. 669-672, 2003.

[3] T. Wigren, E. Abd-Elrady and T. Söderström, "Least squares harmonic signal analysis using periodic orbits of ODEs", *in Prep. 13:th IFAC Symposium on System Identification*, Rotterdam, The Netherlands, pp.1584-1589, August 27-29, 2003.

[4] T. Wigren and T. Söderström, "A second order ODE is sufficient for modeling of many periodic signals", *Int. J. Contr.* , vol. 78, no. 13, pp. 982-996, Sep., 2005.

[5] T. Wigren, “Model order and identifiability of non-linear biological systems in stable oscillation”, *submitted to IEEE/ACM Trans. on Computational Biology and Bioinformatics,* April, 2014.

[6] D. Hanselmann and B. Littlefield. *Mastering Matlab 5 - A Comprehensive Tutorial and Reference.* Upper Saddle River, NJ: Prentice Hall, 1998.

[7] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification.* Cambridge, Ma: MIT Press, 1983.

[8] I. R. Manchester, M.M. Tobenkin and J. Wang, “Identification of nonlinear systems with stable oscillations”, *Proc. CDC-ECC,* Orlando, FL, pp. 5792-5597, Dec. 12-15, 2011.

[9] T. Wigren, "Recursive prediction error identification and scaling of nonlinear state space models using a restricted black box parameterization", *Automatica,* vol. 42, no. 1, pp. 159-168, 2006.