An **R** library for nonlinear black-box system identification: manual with examples

Helon Vicente Hultmann Ayala[a,b,*], Marcos Cesar Gritti[c], Leandro dos Santos Coelho[b,c]

[a]*Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro (PUC-Rio)*
Marques de Sao Vicente, 225, Zip code 22453-900, Rio de Janeiro, RJ, Brazil
[b]*Industrial and Systems Engineering Graduate Program (PPGEPS), Pontifical Catholic University of Parana (PUCPR)*
Imaculada Conceicao, 1155, Zip code 80215-901, Curitiba, PR, Brazil
[c]*Department of Electrical Engineering, Federal University of Parana (UFPR)*
Cel. Francisco Heraclito dos Santos, 100, Zip code 81531-980 Curitiba, PR, Brazil

Abstract

In the present manual we provide installation instructions, working examples with code, and the outputs using the **narmax** toolbox. The raw codes of the examples depicted in this manual can be found in `https://github.com/helonayala/narmax/examples`. The example data are real-world acquired data which are freely available in the internet, which facilitates reproduction using concrete real-world examples.

1. Installation

The package code is attached to the paper in a zip file. It has been tested on the cloud in the freely available service `https://rstudio.cloud/`. The following steps should be made to install the package and run the examples.

Step 1: **Prepare the environment**: Install package dependencies and devtools updated version.

```
R> install.packages(c("devtools","stringr","latex2exp","ggplot2",
 + "progress","roxygen2","tidyr")) R> devtools::install_github("r-
lib/devtools")
```

*Corresponding author. Tel.: +552135271162; Fax: +552135271165
Email addresses: `helon@puc-rio.br` (Helon Vicente Hultmann Ayala), `cesargritti@gmail.com` (Marcos Cesar Gritti), `leandro.coelho@pucpr.br` (Leandro dos Santos Coelho)

Step 2: **Open the project**: Simply click on the `narmax.Rproj` file to open the project file so the environment is configured to build the package.

Step 3: **Build the package**: Hit CTRL + SHIFT + D to build the documentation and CTRL + SHIFT + B to build the package (in Mac OS, hit use instead of CTRL). If successful the package functions will be available after a `library(narmax)` call.

Step 4: **Navigate the examples**: To run the examples detailed in the next Section, simply navigate to the `examples/` folder, **set it as the working directory** (otherwise the filename extensions for reading the data wont't work), and source the desired files.

The package can be downloaded on the internet, please check GitHub for the most recent version:
`https://github.com/helonayala/narmax`

2. Dowloading the data

In order to run the examples, the users should firstly download the third-party data that is freely available on the internet and then dump them in the `data/` folder

- Example 2: click on the link below to download it

  `ftp://ftp.esat.kuleuven.be/pub/SISTA/data/thermic/heating_system.dat.gz`

  Description is available at

  `ftp://ftp.esat.kuleuven.be/pub/SISTA/data/thermic/heating_system.txt`

- Example 3: click on the links below to download it

  `https://www.york.ac.uk/depts/maths/data/ts/ts22.dat`

  `https://www.york.ac.uk/depts/maths/data/ts/ts23.dat`

  Description can be found in [3].

- Example 4: click on the links below to download it

  `http://www.nonlinearbenchmark.org/FILES/BENCHMARKS/WIENERHAMMERSTEINPROCESS/WienerHammersteinFiles.zip`

  Description can be found in the link below.

  `http://www.nonlinearbenchmark.org/#Tanks`

Number	Model used	R script file	Description
1	NARMAX	`ex01_billings.R`	Example 3.7 in [1], simulated system for identification.
2	ARMAX	`ex02_heating.R`	Heating system, introduced in [2] for data-driven modeling.
3	NARMAX	`ex03_box_jenkins.R`	Box-Jenkins gas furnace [3].
4	NARMAX	`ex04_coupled_drives.R`	Coupled drives system [4].

Table 1: Summary of the examples of the package usage described in this paper.

3. Using the package

In the following we provide four examples of the package usage, together with R code. The list of examples is provided in Table 1. The code for each example is given in the `/examples` folder.

4. Example 1: textbook exercise

Consider the following system

$$y(k) = 0.5y(k-1)+u(k-2)+0.1u^2(k-1)+0.5e(k-1)+0.1u(k-1)e(k-2)+e(k) \quad (1)$$

given as Example 3.7 in [1]. We artificially generate $y(k)$ by setting the input $u(k)$ as a Gaussian white sequence (0 mean and 1 variance), and the system noise $e(k)$ as a zero mean 0.04 variance signal.

```
R> N <- 400
R> u <- rnorm(N, mean = 0, sd=1)
R> e <- rnorm(N, mean = 0, sd=0.04^2)
R> y <- rep(0, length(u))
R> for (k in 3:N) {
y[k] <- 0.5 * y[k-1] + u[k-2] + 0.1 * (u[k-2]^2) +
0.5 * e[k-1] + 0.1 * u[k-1] * e[k-2] + e[k]
}
```

We define the model structure with $na = nb = nc = l = 2$ and we use the ELS-FROLS algorithm to select and estimate the terms of the system as in Equation 1 using only input and output data. For this full model structure, we have a total of $\binom{n+l}{l} = 28$ terms, where $n = na + nb + nc$. Using the toolbox we define the model as

```
R> mdl <- narmax(ny = 2, nu = 2, ne = 2, nl = 2)
R> print(mdl)
```

```
narmax(ny = 2, nu = 2, ne = 2, nl = 2)

Not estimated
```

The code output confirms the model order and informs that it has not been estimated yet. We can estimate the model using the ELS-FROLS with the following command

```
R> mdl <- estimate(mdl, y, u, rho_p = 1e-2, rho_n = 1.9e-6)
R> print(mdl)
narmax(ny = 2, nu = 2, ne = 2, nl = 2)

Term          Coefficient
u(k-2)              1.0001
y(k-1)              0.5002
u(k-2)u(k-2)        0.1000
e(k-1)              0.4766
```

Note that in order to estimate the model, we define the `estimate()` function arguments in the following order: model object, output and input data, and process and noise terms thresholds for terminating the model construction procedure. We also see that the coefficients are in adherence with Equation 1. Having an estimated model at hand, we can perform predictions.

```
R> P0 <- predict(mdl, y, u, K = 0)
Running narmax prediction ...
Done. R2 =
1.0000
R> P1 <- predict(mdl, y, u, K = 1)
Running narmax prediction ...
Done. R2 = 1.0000
```

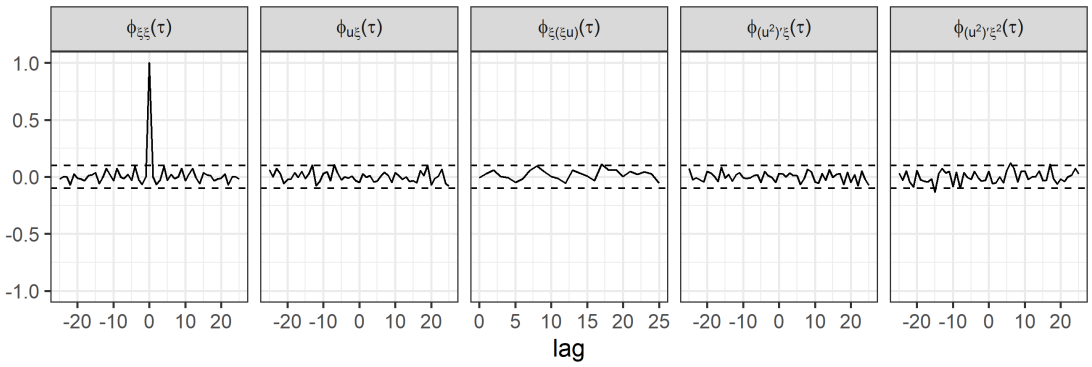

Figure 1: Statistical analysis of the residuals for Example 1. It is possible to see that the tests are adequately satisfied.

The input argument for `predict()` that controls which type of prediction (OSA or FR) is `K`, where 0 and 1 set FR and OSA predictions respectively. In the examples given here the output for the predictions nomenclature also follow the same convention (`P0` and `P1`). The performance of the prediction in terms of R^2 is given right below each function call for prediction. The correlation based tests can be plotted by simply typing `R> P1$xcorrel`, and the output is given in Figure 1.

5. Example 2: heating system

The present case study is a heating system whose data is available at the DaISy (Database for the Identification of Systems) [5]. The input commands a 300 Watt lamp suspended above a steel plate whose temperature is the output measured. This case study has been used in [6] for the model validation.

First we read the data and divide the dataset for estimation and validation. In this example we have 801 samples, so we use the first 400 samples (stored in `ye,ue`)) to construct the model and the rest to validate it.

```
R> data <- matrix(scan("../data/heating_system.dat"),
nrow=801,byrow=TRUE)
R> u <- data[,2]
R> y <- data[,3]
R> ye <- y[1:400]
R> ue <- u[1:400]
```

Now we define the model order as

```
R> mdl <- armax(ny= 4,nu = 4,ne = 10)
armax(ny = 4, nu = 4, ne = 10)

Not estimated
```

and estimate its parameters through

```
R> mdl <- estimate(mdl,ye,ue)
R> print(mdl)
armax(ny = 4, nu = 4, ne = 10)

Term         Coefficient
-y(k-1)            -1.8730
-y(k-2)             0.5438
```

```
-y(k-3)              0.6032
-y(k-4)             -0.2732
u(k-1)               0.0939
u(k-2)               0.2126
u(k-3)               0.0013
u(k-4)              -0.2892
e(k-1)              -0.6480
e(k-2)              -0.4008
e(k-3)               0.0202
e(k-4)               0.0927
e(k-5)              -0.0467
e(k-6)               0.0547
e(k-7)               0.0595
e(k-8)              -0.0811
e(k-9)              -0.0386
e(k-10)              0.1087
```

We have increased the model order until the correlation based tests in were satisfied. All the information related to the predictions can be obtained by

```
R> Pe1 <- predict(mdl, ye, ue, K = 1) # one-step-ahead
Running armax prediction ...
Done. R2 = 1.0000
R> Pa0 <- predict(mdl, y, u, K = 0)   # free-run
Running armax prediction ...
Done. R2 = 0.9904
```

Then the plot of the residuals, prediction versus measured data, and correlation-based tests can be obtained respectively by the code below. The command for checking the predictions is depicted below and in Figure 2.

```
R> Pa0$ploty
```

The error can also be plotted accordingly, by issuing the following line.

```
R> Pa0$plote
```

The output is shown in Figure 3 shows. We can see that the error increases in the validation phase (last 400 samples).

The correlation tests can be issued by the command below. They confirm that the model is valid as the statistical analysis shows in Figure 4. In this

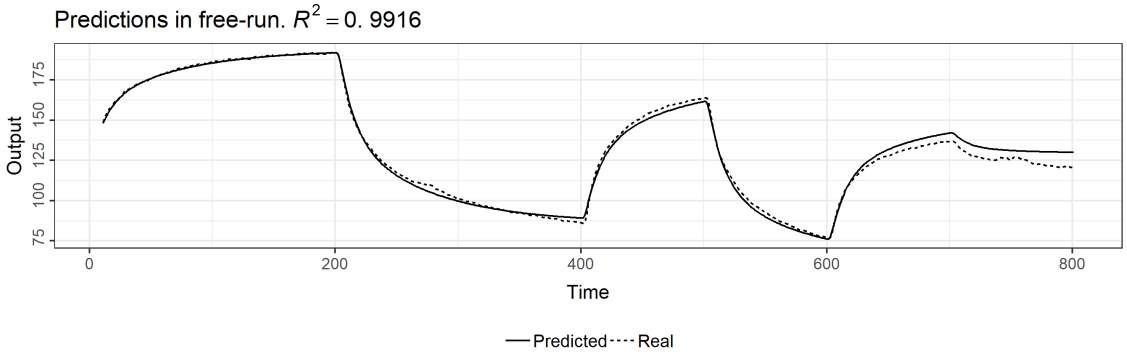

Figure 2: Predictions in FR simulation for all measured data for the model created in Example 2. It is possible to see that the predictions yield a reasonable R^2.

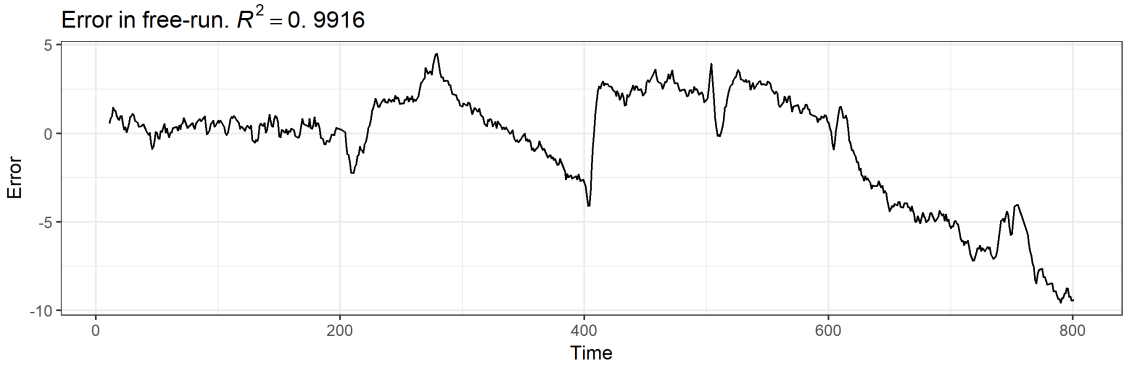

Figure 3: Errors in FR simulation the model created in Example 2.

example we have increased the model orders until the tests were satisfied. The results of the tests indicate that the dynamics present in this set of measurements has been adequately captured by the model.

```
R> Pe1$xcorrel
```

6. Example 3: Box-Jenkins gas furnace

This example has been extensively used in the nonlinear identification literature for testing new methods, see e.g. [7], [8]. The input is the gas flow rate in a gas furnace, whose CO2 concentration in the outlet is the measured desired output. The first step here is to read the data and separate its part used for estimation. The whole dataset contains 296 samples, and we use the first 250 for estimation.

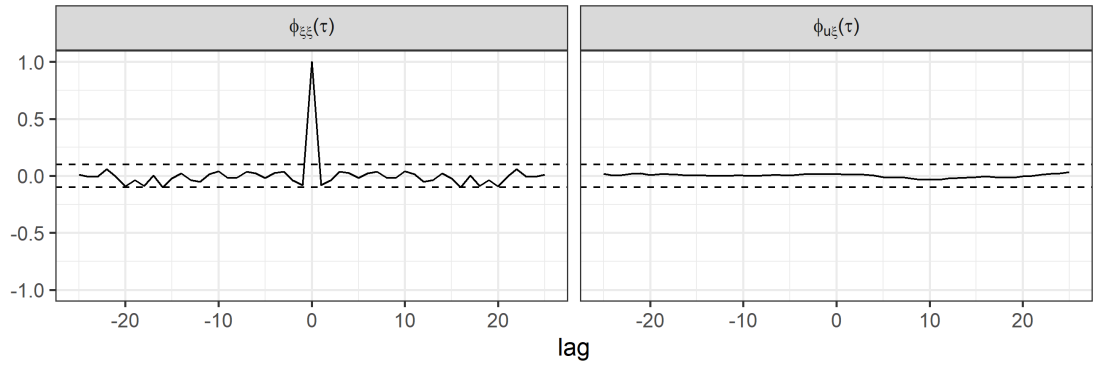

Figure 4: Correlation tests in Equation **??** in the linear systems case for the model created in Example 2. The order of the noise terms were increased until the tests in (c) were satisfied.

```
R> u <- scan("../data/ts22.dat")
R> y <- scan("../data/ts23.dat")
R> ye <- y[1:250]
R> ue <- u[1:250]
```

Now we test two model structures. To do so, we create two variables that contain each of them as below.

```
R> mdl1 <- narmax(ny = 1, nu = 1, ne = 1, nl = 2)
R> mdl2 <- narmax(ny = 5, nu = 5, ne = 1, nl = 2)
```

The estimation is performed using the `estimate()` function, as follows.

```
R> mdl1 <- estimate(mdl1, ye, ue, rho_p = 1e-5, rho_n = 1e-6)
R> mdl2 <- estimate(mdl2, ye, ue, rho_p = 1e-5, rho_n = 1e-6)
```

Now we can run the model predictions in OSA and FR for each model. We do this for the estimation and all dataset, respectively, as below.

```
R> Pe1 <- predict(mdl1, ye, ue, K = 1)
Running narmax prediction ...
Done. R2 = 0.9859
R> Pa1 <- predict(mdl1, y,  u,  K = 0)
Running narmax prediction ...
Done. R2 = 0.8174
R> Pe2 <- predict(mdl2, ye, ue, K = 1)
Running narmax prediction ...
```

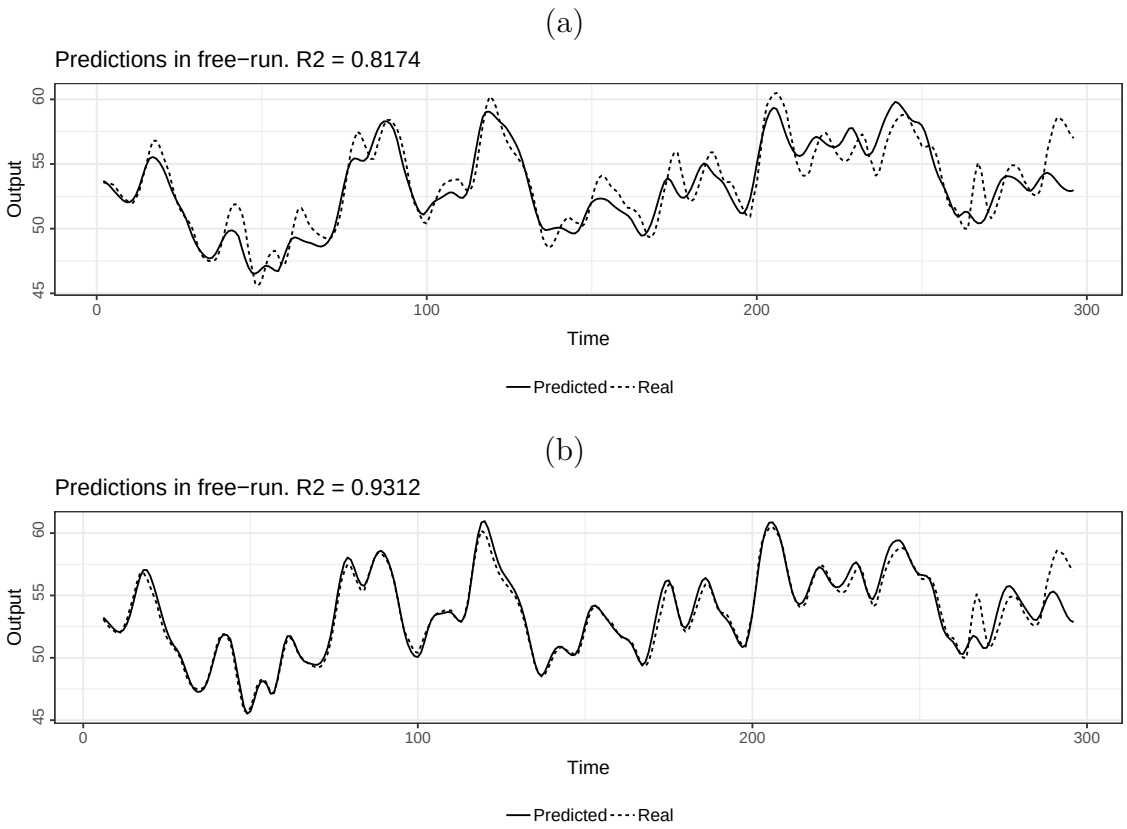

Figure 5: Predictions in FR for the models created in Example 3. (a) mdl1 and (b) mdl2. It is possible to confirm that by increasing the model orders the prediction was improved.

```
Done. R2 = 0.9977
R> Pa2 <- predict(mdl2, y,  u,  K = 0)
Running narmax prediction ...
Done. R2 = 0.9312
```

We can see that, while the performance in OSA is very similar for both models, the first model performs poorly in FR. The plots for both models in FR are shown in Figure 5 and we obtain them by running the following code.

```
R> Pa1$ploty
R> Pa2$ploty
```

The correlation-based tests also reveal that mdl1 fails to capture the dynamics present in the dataset. We obtain the plot for both models as shown in the code below. Again, by increasing the orders of the model we improved the model validation metrics as depicted in Figure 6.

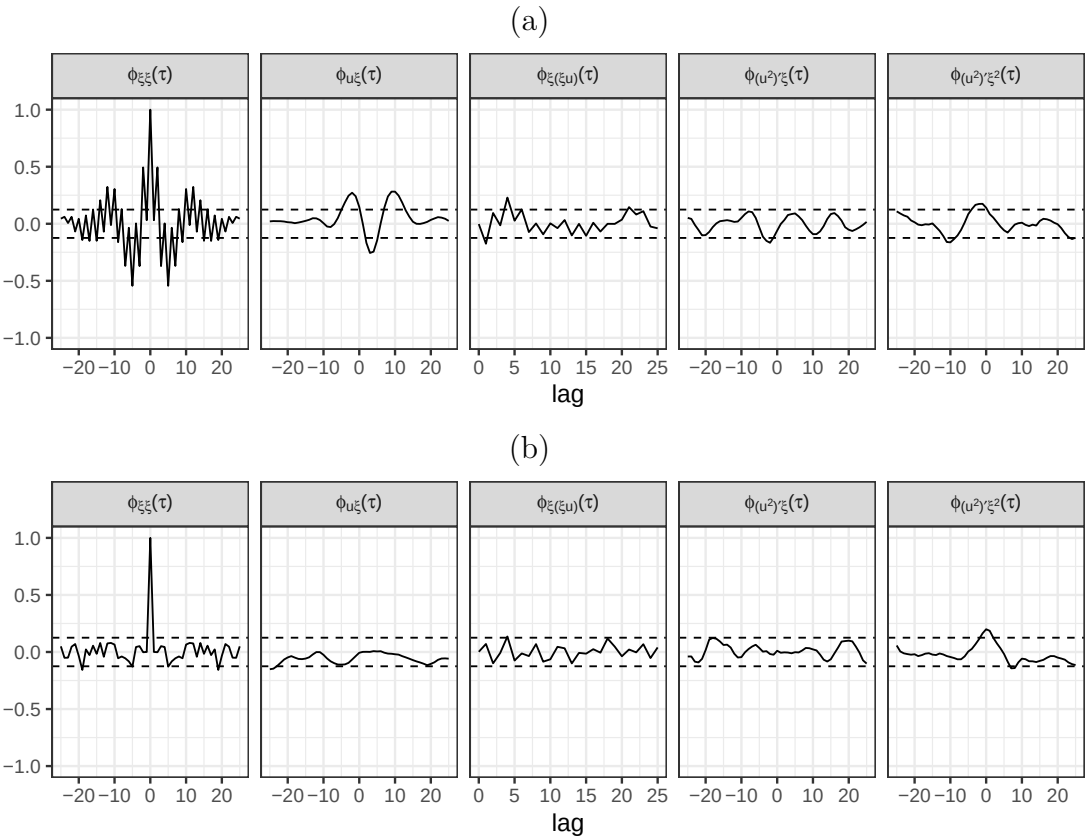

Figure 6: Statistical analysis of the residuals for the models created in Example 3 - Box-Jenkins gas furnace. (a) mdl1 and (b) mdl2. Increasing the complexity of the model improved the requirements given by the correlation-based tests.

7. Example 4: Coupled Drives

The coupled drives system is equipped by two motors connected by a flexible belt and a pulley fixed by a spring. The benchmark was introduced by [4] and as the data is freely available in the internet it has been increasingly used to test new identification algorithms, see e.g. [9], [10]. Both motors are actuated with the same input and the velocity is measured as the output with a sensor insensitive to the direction of the movement. The firs step is to load the data and define a NARMAX model with a suitable order. After some trial and error, we get the following model structure and ELS-FROLS estimation parameters.

```
R> data <- read.csv('../data/DATAUNIF.csv',stringsAsFactors=FALSE,header = TRUE)
R> u <- data$u12
R> y <- data$z12
```

```
R> mdl <- narmax(ny = 10, nu = 10, ne = 2, nl = 2)
R> mdl <- estimate(mdl, y, u, rho_p = 1e-3, rho_n = 1e-5)
```

Now that the model has been created, we can check the predictions, as done below.

```
R> P1 <- predict(mdl, y, u, K = 1)
Running narmax prediction ...
Done. R2 = 0.9986
R> P0 <- predict(mdl, y, u, K = 0)
Running narmax prediction ...
Done. R2 = 0.9590
```

Again the plots for the predictions and the correlation-based tests can be shown with the following commands. The outputs generated are presented in Figure 7.

```
R> P0$ploty
R> P1$xcorrel
```

References

[1] S. A. Billings, Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains, John Wiley & Sons Ltd., West Sussex, United Kingdom, 2013 (2013).

[2] G. Dullerud, R. Smith, Sampled-data model validation: An algorithm and experimental application, International Journal of Robust and Nonlinear Control 6 (9-10) (1996) 1065–1078 (1996).

[3] G. E. P. Box, G. M. Jenkins, Time series analysis, forecasting and control, Holden Day, San Francisco, USA, 1970 (1970).

[4] T. Wigren, J. Schoukens, Three free data sets for development and benchmarking in nonlinear system identification, in: European Control Conference, Zurich, Switzerland, 2013, pp. 2933–2938 (2013).

[5] B. L. R. De Moor, Daisy: Database for the identification of systems[Online; accessed 16-January-2019] (2019).
URL `http://homes.esat.kuleuven.be/~smc/daisy/`

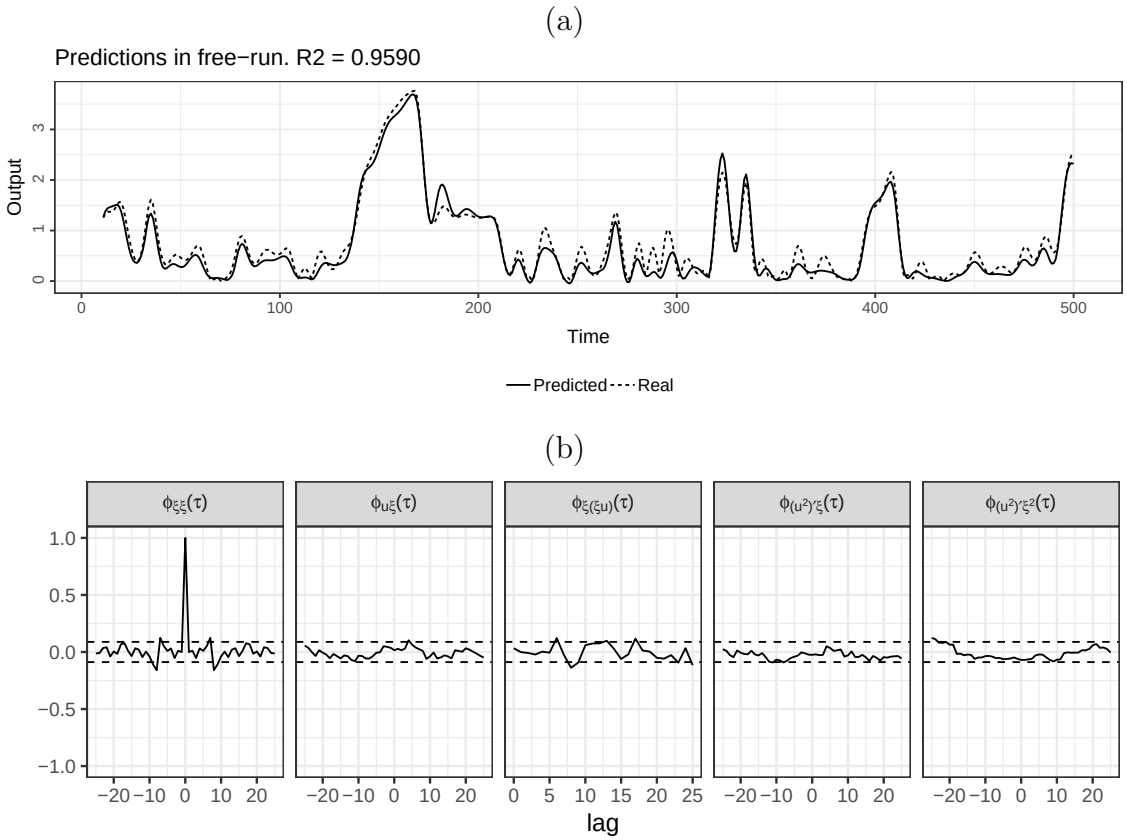

Figure 7: Outputs for the model created in Example 4 - coupled drives exchanger: (a) predictions in FR simulation for all measured data and (b) the correlation tests in Equation **??**. We can see here that in spite that the predictions are accurate in FR, it is still possible to improve the model with the same dataset, as the few excursions in the first test show.

[6] G. Dullerud, R. Smith, Sampled-data model validation: An algorithm and experimental application, International Journal of Robust and Nonlinear Control 6 (9-10) (1996) 1065–1078 (1996).

[7] M. Sugeno, T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, IEEE Transactions on Fuzzy Systems 1 (1) (1993) 7–13 (1993).

[8] H. V. H. Ayala, L. S. Coelho, Cascaded evolutionary algorithm for nonlinear system identification based on correlation functions and radial basis functions neural networks, Mechanical Systems and Signal Processing 68-69 (2016) 378 – 393 (2016).

[9] F. Sabahi, M. R. Akbarzadeh-T, Extended fuzzy logic: Sets and systems, IEEE Transactions on Fuzzy Systems 24 (3) (2016) 530–543 (2016).

[10] P. Wachel, Wiener system modelling by exponentially weighted aggregation, International Journal of Control 90 (11) (2017) 2480–2489 (2017).