16th IFAC Symposium on System Identification
The International Federation of Automatic Control
Brussels, Belgium. July 11-13, 2012

A *Mathematica* Toolbox for Signals, Models and Identification

Håkan Hjalmarsson * **Jonas Sjöberg** **

* *ACCESS Linnaeus Center, Electrical Engineering, KTH – Royal Institute of Technology, SE100 44 Stockholm, Sweden*
** *Department of Signals and Systems, Chalmers University of Technology, SE41296 Göteborg, Sweden*

Abstract: In this contribution we provide a status report for the *Mathematica* toolbox that is described in **?**. The toolbox covers a comprehensive set of functions for handling deterministic and stochastic signals and models. On top of this the toolbox provides signal processing and system identification methods ranging from non-parametric to parametric, and from linear models to a wide class of non-linear models. Algorithms are tailored to be able to efficiently handle large scale data sets and models as well as symbolic computations. This allows theory to be handled alongside practice, implying that the toolbox provides an environment suitable both for education and data processing. In regards to system identification, one of the novel features is graphical support for building block-based nonlinear models. Another novel feature is that modeling errors can be propagated through applications.

Keywords: signal processing, system identification, stochastic modeling, deterministic modeling, linear systems nonlinear systems

1. INTRODUCTION

A number of potent software packages for system identification are available today, e.g. **????**, offering state-of-the art numerical algorithms for a number of identification methods. There are also a number of signal processing environments available on the market. However, so far the use of symbolic calculations in such packages has been quite restricted. In system identification symbolic calculations can, e.g., be used to compute the predictor gradients needed for the numerical optimization of the model parameters and to compute the Cramér-Rao bounds as an explicit function of model parameters. In fact, by combining numerical and symbolical algorithms an integrated environment can be obtained covering both theoretical concepts and data processing. This is very useful from an educational point of view, but is also of use for researchers in the field.

These driving forces lie behind the signals, models and identification package for *Mathematica* that is described in this paper. The package is still under development so this paper provides a status report. However, the overall aim is to develop an environment that encompasses analysis tools for multivariable deterministic and stochastic signals and systems theory, as well as state-of-the art algorithms for system identification and signal processing in a seamless way. Thus it should be useful all the way from standard courses on signals and linear systems to identification of non-linear industrial processes. In the sections that follow we outline the available functionalities and how they can be used.

2. BASIC STRUCTURE

The package is object oriented, meaning that both signals and models are treated as objects equipped with certain properties that determine the behaviour of the object and how it interacts with other objects. For example, a time-domain signal is embedded in an object called `SignalObject`. It possesses a number of properties: `Function`, the function that defines the actual samples of the signal in the time-domain. As the function may contain symbolic elements, `Variable` defines which symbol in `Function` that is the time-variable. `DataSamplingPeriod` gives the sampling period, `Domain` determines the time-interval over which the signal is defined and `SignalLabels` contain labels for the elements of the signal (which can be matrix-valued). `Assumptions` contain assumptions regarding symbolic variables, e.g. which ones are real-valued. Properties are inherited, or transformed, as operations on a signal are performed. For example when two signals are added, `Function` of the resulting signal becomes the sum of the `Function`s for the two signals. In the remaining part of this section, we outline the main characteristics of the package.

2.1 Object categories

There are three categories of objects: data records, signals and models, all of which can be multivariable. Data records are used for raw data and can be represented in several ways. Signal objects can be seen as extensions of data records, able to represent signals over the entire time axis and with a number of associated properties. A signal can be represented in the time-domain (as a `SignalObject`), as a z-transform (as a `ZObject`) or, if the signal has

978-3-902823-06-9/12/$20.00 © 2012 IFAC

10.3182/20120711-3-BE-2027.00273

a finite domain, as a Discrete Fourier Transform (as a `FourierObject`).

Models can be represented in a number of formats. Firstly, the general `ImpulseResponseModel`, `TransferResponseModel` and `FrequencyResponseModel` correspond to the three signal formats discussed in the previous paragraph. These formats can handle non-causal models. For linear causal models there are additional formats: `LinearStateSpace` and a number of formats adapted to common model structures in system identification (ARX, ARMAX, Box-Jenkins,...). For non-linear models there are a number of ready-made formats, e.g. NLFIR, NLARX, NLARMAX, NLOE, NLBJ, Wiener and Hammerstein and combinations of these, but there is also support for more general tailor-made non-linear state-space models.

2.2 Object transformations

Transformations between equivalent representations is straightforward. For example, if `s` is a SignalObject, `ZObject[s]` transforms `s` to the z-transform domain.

Non-linear models can be linearized and then converted to any of the general formats for linear systems.

2.3 Primitives

The function of an object can be accessed by applying an argument to the object. For example if `s` is a SignalObject `s[t]` gives the signal's value at time `t` and `ZObject[s][z]` gives the z-transform of `s` evaluated at `z`. If `G` is a `TransferResponseModel`, `G[Exp[I omega]]` gives the frequency response of the model. Thus the underlying function can be manipulated by any function available in *Mathematica*.

In addition, objects can be directly manipulated by a number of common operations. For example, with `s1` and `s2` being two `SignalObjects`, `s1+s2` generates a new `SignalObject` with the signal values being the sum of those of the signals. With `G1` and `G2` being two `TransferResponseModel`, `G1.G2` generates a new `TransferResponseModel` corresponding to `G2` in series with `G1`. Other primitives include parallell interconnection of systems and inversion of systems and signals.

3. A SELECTION OF FUNCTIONALITIES

3.1 Model properties

Model properties that can be computed are transmission zeros, model poles, transfer function, impulse response, the McMillan degree and the causal- and anti-causal parts.

3.2 Model and stochastic signal interaction

For a spectrum Φ (which can be represented as a `SignalObject` or a `ZObject`), the positive real part can be computed as well as a stable spectral factor.

The correlation function and spectrum of the output of a system excited by a signal with a given spectrum can be computed. Also the cross-correlation function and the cross-spectrum between outputs and inputs can be computed.

3.3 Estimation of signal properties

Standard second-order properties of stationary signals, such as the correlation function and the spectrum, can be estimated from data. Smoothing can be applied, and a number of smoothing windows are available. The user can also define new smoothing windows. The resulting objects are `SignalObject`, `ZObject` or `FourierObject` and are thus fully integrated in the environment.

3.4 Non-parametric model estimation

Standard non-parametric model estimation methods are available, such as Empirical Transfer Function Estimate and Spectral Analysis. These models are represented as `ImpulseResponseModel`, `TransferResponseModel` or `FrequencyResponseModel` objects and are thus fully integrated in the environment.

3.5 Nonlinear identification

The number of model structures one can choose beteewen increase considerably when one takes the step from linear to nonlinear identification. To help the user to handle this increase of complexity, the package structures nonlinear models in a logical way in analogy with linear black-box models. Also, a substantial effort has been devoted to create tools which enhance the possibilities to illustrate and evaluate nonlinear models.

Parameter estimation

The tool supports prediction error identification of parameters in linear and nonlinear models. There are defaults for many of the user choices, but thanks to the symbolic computational possibilities the user can modify the options readily. For example, the user can specify any criterion of their choice and the criterion can be parameterized so that the noise distribution is estimated together with the other model parameters.

Nonlinear Model Structures

The tool can handle, basically, any model structure which can be expressed in state-space form

$$x(t+1) = f(x(t), u(t), v(t)\theta) \quad (1)$$

$$y(t) = g(x(t), u(t), v(t), \theta) \quad (2)$$

where $y(t)$ is the ouput signal, $u(t)$ the input signal, and $v(t)$ the disturbance signal. $x[t]$ is the state vector. The only requirement is that the derivative of the output exists so that derivative based criterion minimization can be applied.

Tailored model structures A tailored model means that the user specifies the model equations on the form (2). The user can specify the functions f and g freely, based on any prior insights.

Black box model structures Nonlinear counterparts of the linear black-box models are supported, namely NLFIR, NLARX, NLARMAX, NLBJ, NLAR and NLARMA models. They are described by a pseudo-regressor and a nonlinear mapping.

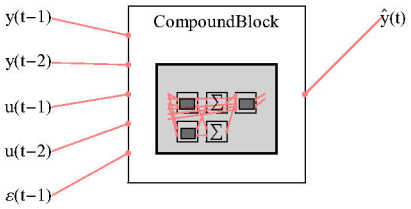

Fig. 1. A NLARMAX model.

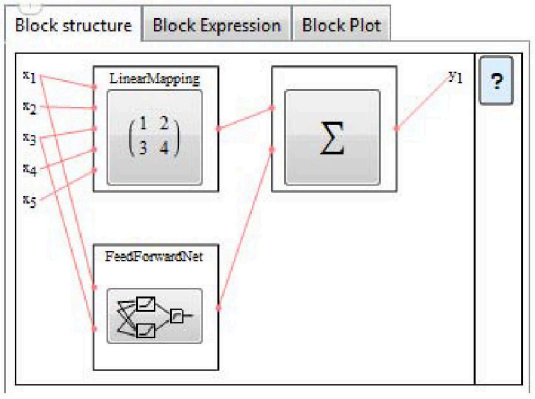

Fig. 2. This is an example of what can be placed inside a compound block, a nonlinear mapping consisting of a couple of blocks.

Model illustration Figure 1 shows how nonlinear black-box models are displayed. The mapping block for the nonlinear models contains a button giving more information about the mapping. Figure 2 shows an example of what the mapping may consist of.

The nonlinear models are estimated starting from linear models, ie, the mapping initially consists of a linear block. Block nonlinearities can be added and the linear model is used to obtain good initial parameter estimates of the nonlinear model. After a block has been added all parameters need to be estimated.

Special model structures Well known special nonlinear model structures, such as Wiener models Hammerstein models, and the combinations of them, Wiener-Hammerstein and Hammerstein-Wiener models, are supported and the algorithms were used to produce the results published in **?**.

Figure 3 shows a Hammerstein-Wiener model consisting of two SISO nonlinearities and a linear model inbetween. By pressing any of the three parts the user obtains information of that particular part of the model. In Figure 3 a nonlinear part of the model is chosen and its block structure is shown. Figure 4 shows a plot of the nonlinearity together with the data points.

If the linear part is chosen one can choose between many ways to illustrate a linear system and in Figure 5 the Bode plot of the system has been chosen.

Linearization

Nonlinear models can be linearized at any given state and value of the input signal.

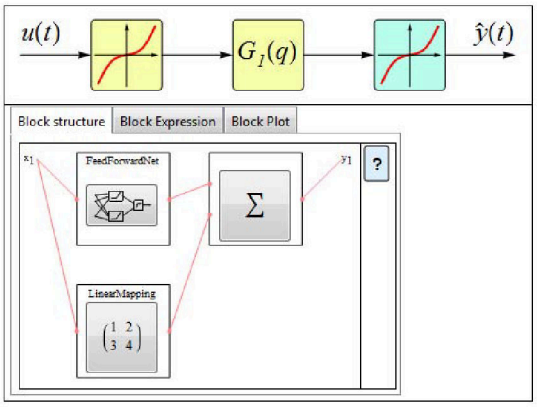

Fig. 3. Illustration of a Hammerstein-Wiener model, the block structure of the second nonlinearity is depicted.

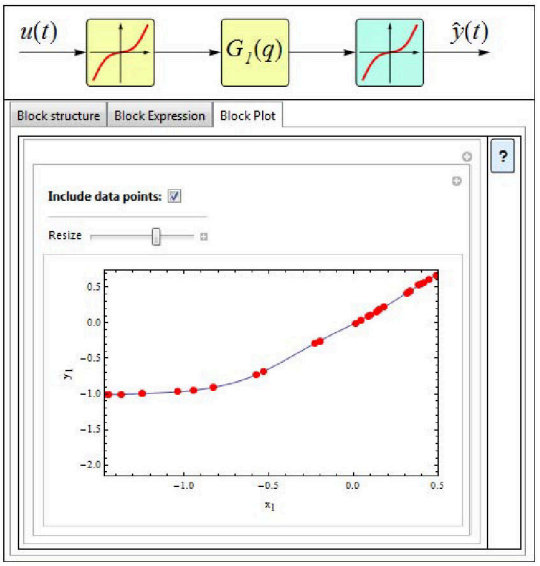

Fig. 4. A Hammerstein-Wiener model where the function of the second nonlinear block is depicted.

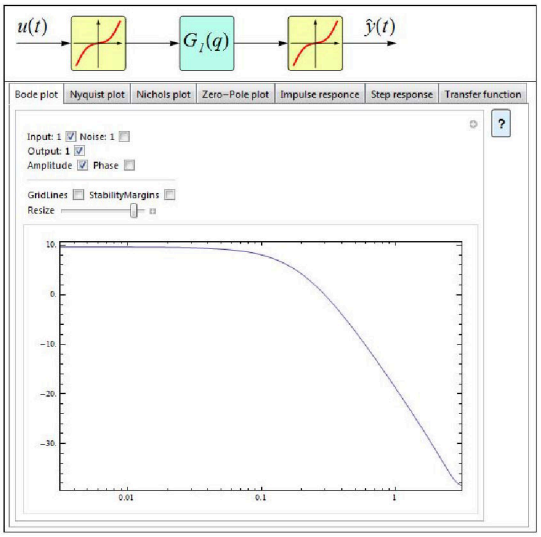

Fig. 5. Illustration of the linear part of a Hammerstein-Wiener model.

Summary of Functionalities

In short, the package supports a wide range of algorithms for identification of linear and nonlinear models of dynamical systems, which can be used for simulation, prediction, and for control design. It also supports numerical and symbolic processing of deterministic and stationary stochastic processes. The toolbox works with discrete time signals. Below we list available functionalities.

Deterministic signals. Double sided z-transform and Fourier transforms.

Stochastic signals. Sample correlations, spectral analysis

Deterministic systems. Multi-input/multi-output, frequency function, poles and transmission zeros, impulse response. Transformations between system representations

Stochastic systems. Positive real part, spectral factorization, system realization, output covariance and input/output cross-correlation functions, propagation of model uncertainty

Linear parametric model estimation. Prediction error estimation of standard model structures, e.g. FIR, ARX, ARMA, ARMAX, OE, BJ. Tailor made model structures. Subspace identification. Non-parametric identification using spectral analysis

Non-linear parametric model estimation. Prediction error estimation of standard model structures, e.g. NLFIR, NLARX, NLARMA, NLARMAX, NLOE, NLBJ, Wiener, Hammerstein, Wiener-Hammerstein, Hammerstein-Wiener. Tailor made block based model structures.

4. SOME APPLICATIONS

Below we illustrate the use of the package with some simple applications.

4.1 Spectral factorization of a sum of spectra

Here two first order systems are defined

```
G1 =TransferResponseModel[
      {TransformFunction → {{z/(z − 0.5)}},
      ROC → {{{0.5, Indeterminate}}}}]
G2 =TransferResponseModel[
      {TransformFunction → {{z/(z − 0.8)}},
      ROC → {{{0.8, Indeterminate}}}}]
```

Above, ROC is the model property that defines the region of convergence for the z-transform. The spectra of the outputs of these two systems when they have zero mean white noise with unit variance are obtained by the operations

$$\Phi 1 = \text{Spectrum[G1]}$$
$$\Phi 2 = \text{Spectrum[G2]}$$

which results in

$$\text{ZObject}\,[\{\text{ROC} \to \{\{\{0.5, 2.\}\}\},\ \text{SignalLabels} \to \{\{\Phi_{y_1 y_1}\}\},\ \text{TransformFunction} \to \left\{\left\{\frac{1}{1.25 - \frac{0.5}{z} - 0.5z}\right\}\right\}\right\}\right]$$

and

$$\text{ZObject}\,[\{\text{ROC} \to \{\{\{0.8, 1.25\}\}\},\ \text{SignalLabels} \to \{\{\Phi_{y_2 y_2}\}\},\ \text{TransformFunction} \to \left\{\left\{\frac{1}{1.64 - \frac{0.8}{z} - 0.8z}\right\}\right\}\right\}\right]$$

Performing the operation

$$\Phi = \Phi 1 + \Phi 2$$

gives the spectrum when the outputs from $G1$ and $G2$ are added, assuming that the white noise inputs are independent. The result is

$$\text{ZObject}\,[\{\text{ROC} \to \{\{\{0.8, 1.25\}\}\},\ \text{SignalLabels} \to \{\{\Phi_{y_1 y_1} + \Phi_{y_2 y_2}\}\}, \text{TransformFunction} \to \left\{\left\{\frac{1}{1.64 - \frac{0.8}{z} - 0.8z} + \frac{1}{1.25 - \frac{0.5}{z} - 0.5z}\right\}\right\}\right\}\right]$$

A stable minimum phase spectral factor of this spectrum is now obtained by

$$H = \text{SpectralFactorization}[\Phi]$$

and the result is

$$\text{TransferResponseModel}\,[\{\text{ROC} \to \{\{\{0.8, \text{Indeterminate}\}\}\},\ \text{SignalLabels} \to \{\{\{\}\}, \{\{\}\}\},\ \text{TransformFunction} \to \left\{\left\{\frac{-0.902279z + 1.4408z^2}{0.4 - 1.3z + z^2}\right\}\right\}\right\}\right]$$

We can verify that this is a spectral factor by computing the spectrum of the output from H and comparing with Φ

```
ΦH = Spectrum[H];
Plot[Abs[{Φ[Exp[Iω]][[1, 1]], ΦH[Exp[Iω]][[1, 1]]}],
{ω, 0, Pi}, PlotRange → {0, 30}]
```

The result is shown in Figure 6. As seen, the spectra overlaps since the two spectra are identical.

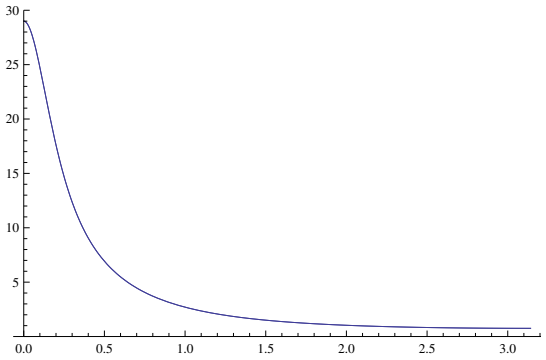

Fig. 6. Comparison of original spectrum and spectrum from spectral factorization (totally overlapping).

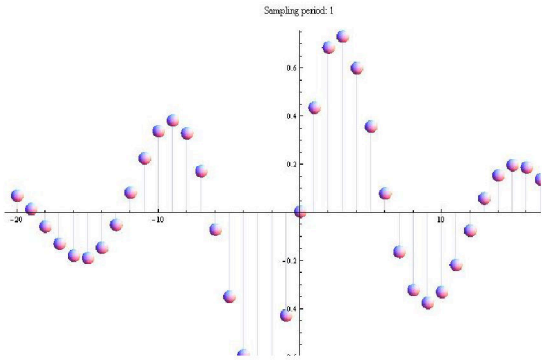

Fig. 7. Signal.

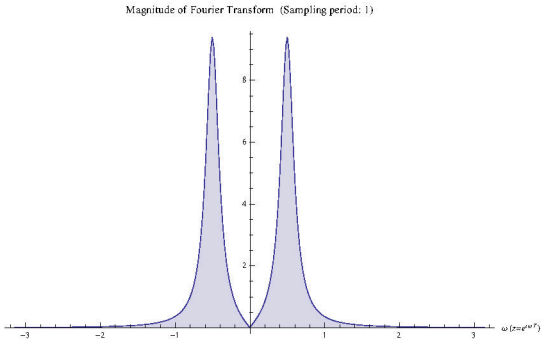

Fig. 8. Magnitude plot of the frequency contents.

4.2 The effect of signal truncation

Here we consider the deterministic signal seen in Figure 7, with infinite domain in both time directions.

In the package it is defined as

r =SignalObject[{Function → {{0.9^Abs[n]Sin[0.5n]}},
Domain → {{{−Infinity, Infinity}}}}];

The frequency contents is obtained by

SignalPlot[ZObject[r]]

see Figure 8. We can truncate the signal duration by changing the domain

rtrunc =SignalObject[{Function → {{0.9^Abs[n]Sin[0.5n]}},
Domain → {{{−-10, 10}}}}];

Issuing

SignalPlot[*ZObject*[*rtrunc*]]

results in the plot of the magnitude of the frequency contents seen in Figure 9. As seen, the truncation has lead to a widening of the main lobe as well as the appearance of a number of side-lobes. This can readliy be explained by theory.

4.3 Non-parametric model estimation

Consider the system

$$y_t = \frac{1}{1 - 0.5z^{-1}} u_t + e_t$$

where u_t is the input and e_t the measurement noise, both being zero mean white noise sequences with variances 1 and 0.1, respectively.

A non-parametric estimate of G, using a Hann window of length 5, is obtained by

{Ghat, Phivvhat} = TransferResponseEstimate[y, u, {{Hann[5][n]}}];

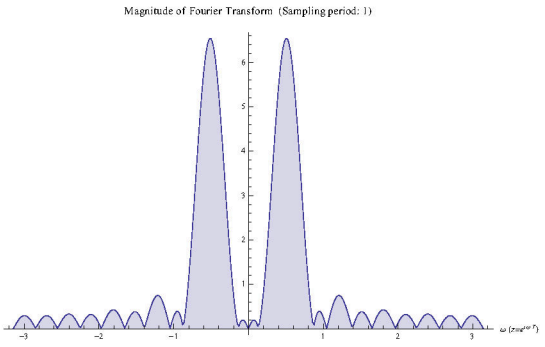

Fig. 9. Magnitude plot of the frequency contents of the truncated signal.

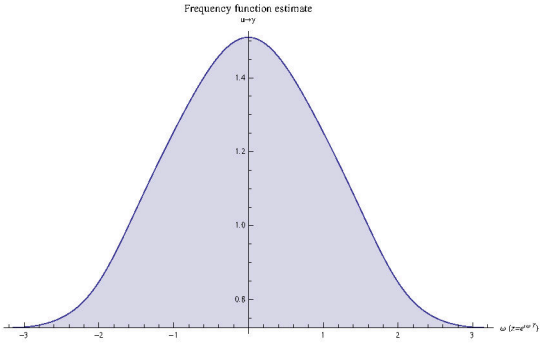

Fig. 10. Magnitude plot of the frequency function of the non-parametric estimate of G

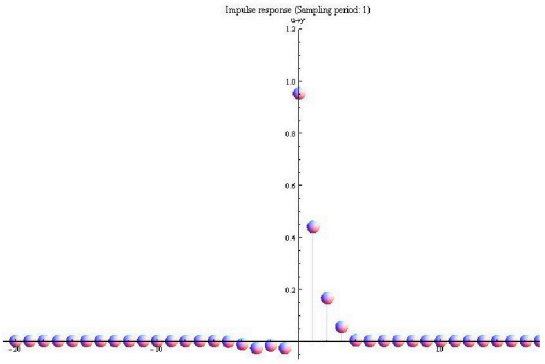

Fig. 11. Impulse response of non-parametric estimate of G

A magnitude plot of the non-parametric frequency function estimate is given in Figure 10. The impulse response can be examined by transforming the model according to

ghat = ImpuseResponseEstimate[Ghat];

It is shown in Figure 11. Perhaps not generally known is that non-parametric model estimates typically are non-causal. This holds for this model where a few of the impulse response coefficients for negative lags are non-zero.

5. EXPERIENCES FROM THE IMPLEMENTATION

At the same time as the combination of numerical and symbolic algorithms open up new perspectives in regards to which functionalities that can be provided, one lesson from this project is that the implementation time grows significantly.

Firstly, more elaborate checks of what the user supplies as arguments to a function are required, and different

processing, sometimes using entirely different algorithms, is required dependent on the input. One example is the computation of the transfer response of a model. For linear state-space models, with numerical state space matrices $(A, B; C, D)$, a numerically stable and computationally efficient algorithm based on the Hessenberg form **?** is used, but when the model contains symbolic parameters the starightforward, but numerically worse, expression

$$G(s) = C(sI - A)^{-1}B + D$$

has to be used.

Another complicating factor in the development is that it is not straightforward to foresee the result of symbolic manipulations such as simplifications of expressions. Post-processing after such manipulations thus have to be made with care.

A final observation is that testing time of implementations also grows significantly as compared with pure numerical algorithms. The reason is simply that there are just so many more possibilities that have to be tested.

6. CONCLUSIONS

There is a range of interesting avenues for further developments when it comes to using symbolic tools in signal processing and system identification. For example, *Mathematica* offers capabilities to maximize and solve multinomial problems.

An interesting research topic is how to balance symbolic and numerical computations. It is usually more computationally expensive to make a symbolic solution but once this is available, it is easy to compute the solution for any parameter.

REFERENCES

Garnier, H., Gilson, M., and Laurain, V. (2009). The CONTSID toolbox for Matlab: extensions and latest developments. In *Proc. 15 IFAC Symposium on System Identification*. Saint- Malo.

Laub, A. (1981). Efficient multivariable frequency response computations. *IEEE Transactions on Automatic Control*, 26(2), 407–408.

Ljung, L., Singh, R., Zhang, Q., Lindskog, P., and Juditski, A. (2009). Developments in Mathworks system identification toolbox. In *Proc. 15 IFAC Symposium on System Identification*. Saint- Malo.

Sjöberg, J. and Schoukens, J. (2012). Initializing Wiener-Hammerstein models based on partitioning of the best linear approximation. *Automatica*, 48(2), 353–359.

Sjöberg, J. and Hjalmarsson, H. (2009). A system, signals and identification toolbox in Mathematica with symbolic capabilities. In *15th IFAC Symposium on System Identification*. Saint-Malo, France.

Wills, A., Mills, A., and Ninness, B. (2009). A matlab software environment for system identification. In *Proceedings of the 15th IFAC Symposium on System Identification, Saint- Malo, France*.

Young, P. (2009). The Captaion toolbox. In *Proc. 15 IFAC Symposium on System Identification*. Saint- Malo.