SYSDYNET - A MATLAB App and Toolbox for Dynamic Network Identification [⋆,⋆⋆]

Paul M.J. Van den Hof, **Shengling Shi**, **Harm H.M. Weerts**, **Xiaodong Cheng**, **Karthik R. Ramaswamy**, **Arne G. Dankers**, **H.J. (Mannes) Dreef**, **Stefanie J.M. Fonken**, **Tom R.V. Steentjes** and **Job B.T. Meijer**

Control Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands. (e-mail: p.m.j.vandenhof@tue.nl).

Abstract: Identification in interconnected systems requires the handling of phenomena that go beyond the classical open-loop and closed-loop type of identification problems. Over the last decade a comprehensive theory has been developed for addressing identification problems in linear dynamic networks, formulated in a module framework, where the network structure is characterized by a directed graph in which nodes are signals and links are transfer functions. The resulting methods and approaches have been collected in a MATLAB App and Toolbox, supported by an attractive graphical user interface that provides an interactive workflow for manipulating the structural properties of dynamic networks, applying basic network operations like immersion and module invariance testing, and for investigating network/module generic identifiability and selecting appropriate predictor model inputs and outputs. The workflow supports the allocation of external excitation signals (actuation) and measured node signals (sensing) so as to achieve generic identifiability and provide consistent estimation of target modules. The Toolbox includes algorithms for actual network simulation and identification.

Keywords: System identification, identifiability, dynamic networks, interconnected systems.

1. INTRODUCTION

Increasing attention for dynamic systems operating in a network of interconnected subsystems, have led to the development of decentralized and distributed control systems. While data-driven modeling questions have primarily been addressed in open-loop (non-controlled) experimental circumstances, the need for identification tools that can effectively exploit the structural topology of the interconnections, while going beyond the standard feedback controlled situation, has become apparent. Starting from the early publications Gonçalves and Warnick (2008); Materassi and Innocenti (2010); Van den Hof et al. (2013) a comprehensive theory has been developed for data-driven modeling problems in dynamic networks. This includes questions of identifying the full network, possibly including the topology (interconnection structure), identifying a local part of the network, selection of and handling of different sensing and actuation schemes, and questions of identifiability of either the full network or of a particular subsystem. Methods and algorithms for addressing all of these questions are incorporated in the MATLAB App and Toolbox, under construction, that is being presented here.

While different representations of dynamic networks are available in the literature, see e.g. Verhaegen et al. (2022); Kivits and Van den Hof (2023), for the MATLAB App and Toolbox presented here we will focus on the so-called module framework (Van den Hof et al., 2013), in which a dynamic network is characterized by the equation

$$w(t) = G(q)w(t) + \underbrace{H(q)e(t)}_{v(t)} + R(q)r(t) \tag{1}$$

where w is a column vector of internal node variables, with scalar-valued $w_1 \cdots w_L$ being individual time series, and t is the (discrete) time-variable; q is the forward shift operator, $qw(t) = w(t+1)$.
$G(q)$, $H(q)$ and $R(q)$ are rational (transfer) matrix, with $G(q)$ $(L \times L)$ being hollow (zero diagonal), indicating which node gets input from which other nodes in the network. $H(q)$ $(L \times p)$ is a stable and (left) stably invertible transfer matrix, modelling the (non-measured) disturbances on the network, e a p-dimensional white noise vector process, $R(q)$ $(L \times K)$ a dynamic transfer matrix representing the mapping from external excitation signals $r_1 \cdots r_K$ to internal node variables w.

[⋆] Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.
[⋆⋆]The App and Toolbox are available for download from the landing page www.sysdynet.net.
[1] The authors gratefully acknowledge implementation contributions from Wim Liebregts and Ilja van Oort, as well as support from Gareth Thomas, VersionBay Inc.

The current version of the App mainly works with *network structures*. A network structure is a triple $(\mathcal{T}_G, \mathcal{T}_H, \mathcal{T}_R)$ representing the Boolean adjacency matrices of a network (G, H, R), in a directed network graph that has node signals as nodes/vertices and modules as links/edges. The presence of a module realizing a connection from node w_i to node w_j, is then represented by $[\mathcal{T}_G]_{ji} = 1$. Similarly this applies to $\mathcal{T}_H$ and $\mathcal{T}_R$ with additional e-signals, respectively r-signals, as nodes in the graph. A prototype dynamic network is depicted in Figure 1. Information on the network structure allows us to address

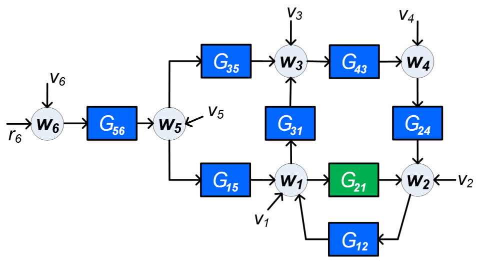


Fig. 1. Prototype dynamic network in a module representation, with in green a possible target module to be identified.

questions of (generic) identifiability, and allows us to construct predictor inputs and outputs for the (consistent) identification of either a full network or of a single module in the network.

2. APP FUNCTIONALITY

2.1 Overall menu structure

The Menu Bar in the top line of the window contains the main functionalities of the App.

- **File**: Load and Save a network structure from/to a file;
- **Actions**: Undo / redo actions;
- **View**: switch between different views of the network structure (in/excluding external signals and disturbance filters);
- **Highlight**: highlight different properties of the nodes and modules in a network structure;
- **Help**: User support.

The main technical operations are present in the Menu Items:

- **Edit**: Edit the network structure and assign properties to nodes and modules;
- **Operations**: Apply some basic operations and tests to a network structure, including immersion and the Parallel Path and Loop Test;
- **Identifiability**: Analyse and synthesize (generic) identifiability, e.g. by allocating external excitation signals;
- **Predictor Model**; Analyse and construct predictor models for identifying a single module;

These four operations each have their own individual windows which are described in the next subsections.

In all main windows, the graph of the network structure is displayed on the central display and standard plotting functions are available for zooming in/out and shifting the network plot. Highlighting functions are available for indicating measured nodes (purple), selected nodes (green) and target modules (purple), as well as for highlighting module properties like feedthrough terms (light blue), known modules (black) and switching modules (pink). Below the central network display, a communication panel is available where messages to the user and results of tests, as well as suggestions to the user are being displayed.

2.2 Edit Window

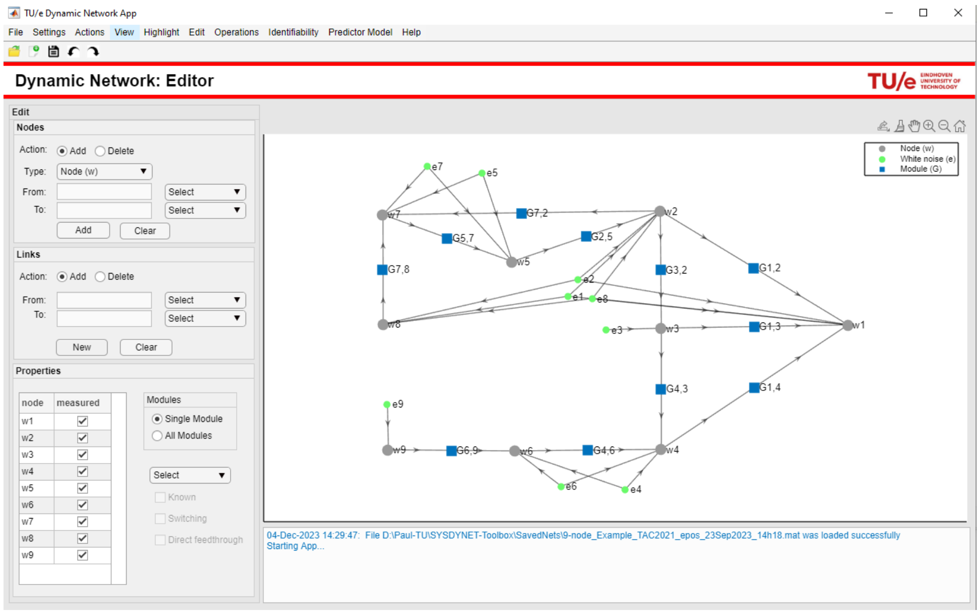
In the Edit Window, the user can interactively edit the network structure/topology and its properties. This includes

- Adding/removing nodes and links (modules);
- Adding/removing noise disturbances and external excitation signals;
- Nodes can be assigned the status "measured" or "unmeasured", depending on whether the node is equipped with a sensor;
- Modules can be assigned the following properties:
  - *Known*: indicating whether the dynamics of this module is known or unknown to the user;
  - *Switching*: indicating whether the dynamics is fixed (non-switching) or switching between different settings.
  - *Direct feedthrough*: indicating whether the transfer function is proper or strictly proper.

For a given network structure, these properties can be highlighted in the network graph by selecting the Menu Item "Highlight". The network edit functions are available through the left panels in the window, but can also be realized by right mouse clicks in the network graph.

2.3 Operations Window

- **Invariant modules and immersion**
  Immersion is the construction of a new network, where a selected set of nodes are maintained and unselected nodes are removed, while the time series of all maintained nodes remain invariant. This network operation is closely related to a so-called Kron reduction, i.e. Gaussian elimination of the unselected nodes. The invariant modules test, tests which modules remain invariant after immersing the unselected nodes in the network. This test can be done separate from, typically before, the actual immersion.
- **Parallel path and loop (PPL) test**
  A selected target module with input w_i and output w_j remains invariant after immersion, if the parallel path and loop test is satisfied. This test verifies whether (a) every path from w_i to w_j passes through a node that is in the set of selected nodes, and (b) every loop around w_j passes through a node that is in the set of selected nodes. The results are presented in the plotted network graph. The test is described in Dankers et al. (2016).
- **Canonical noise model** (Shi et al. (2023b))
  This operation transforms the network to a network where only the selected nodes have a direct contribution from disturbances, and the unselected nodes are disturbance-free. In this transformation the time series of the selected node variables, as well as the



node	measured
w1	✓
w2	✓
w3	✓
w4	✓
w5	✓
w6	✓
w7	✓
w8	✓
w9	✓

Action: [✓ Add] Delete

Modules: [✓ Single Module] All Modules

```
04-Dec-2023 14:29:47:  File D:\Paul-TU\SYSDYNET-Toolbox\SavedNets\9-node_Example_TAC2021_epos_23Sep2023_14h18.mat was loaded successfully
Starting App...
```

Fig. 2. Edit Window for manipulating structural network properties, in so-called "Full View" including the disturbance correlations.

modules in network matrix G, remain invariant. The transformation only changes the noise model H.

2.4 Identifiability Window

In the Identifiability window the generic identifiability of a specified structured network class is evaluated. Generic identifiability is considered for a network class where all unknown modules/links are freely parametrized and all known modules/links are fixed. Generic identifiability concerns the property that a network or module can uniquely be retrieved from the information that is available.

When writing the network equations for the node signals w in an explicit form:

$$w(t) = T_{wr}(q)r(t) + \underbrace{T_{we}(q)e(t)}_{\breve{v}(t)} \tag{2}$$

network identifiability of a network model set $\mathcal{M} := \{(G(q,\theta), H(q,\theta), R(q,\theta)), \theta \in \Theta\}$ assesses whether for two models $M^{(1)}, M^{(2)} \in \mathcal{M}$, with corresponding characteristics $(T_{wr}^{(1)}, \Phi_{\breve{v}}^{(1)}(\omega))$ and $(T_{wr}^{(2)}, \Phi_{\breve{v}}^{(2)}(\omega))$ it holds that:

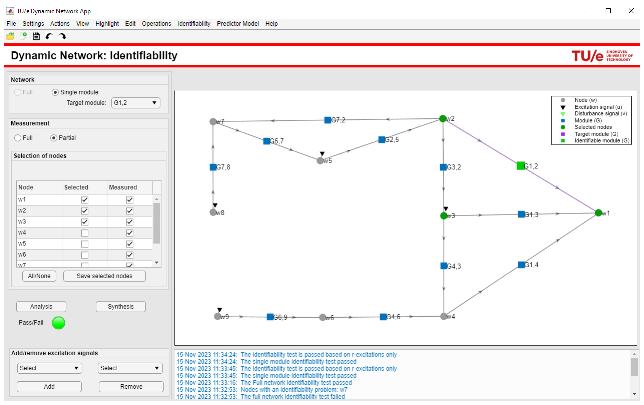
$$\left.\begin{array}{l} T_{wr}^{(1)}(q) = T_{wr}^{(2)}(q) \\ \Phi_{\breve{v}}^{(1)}(\omega) = \Phi_{\breve{v}}^{(2)}(\omega) \text{ for all } \omega \end{array}\right\} \implies \{M^{(1)} = M^{(2)}\}. \tag{3}$$

Rather than investigating whether this implication holds for *all* models $M^{(1)}, M^{(2)} \in \mathcal{M}$, generic identifiability is focusing on this property for *almost all* models in the set. This implies that the property can be tested on the basis of the network structure only, and does not require the numerical values of all parameters in the respective modules.

Generic identifiability is typically dependent on

- the presence and location of external excitation signals r;
- the topology (interconnection structure) of the network;
- the presence of a priori known modules;
- the selection of node variables that are considered to be available for the identifiability study; typically these are the measured node variables.

Generic identifiability of a full network and of a single module can be investigated. In the latter situation a target module G_{ji} needs to be selected, and the right-hand side of implication (3) is replaced by $\{G_{ji}^{(1)}(q) = G_{ji}^{(2)}(q)\}$. While in the *full measurement case* all node signals w in the network are considered in expressions (2)-(3), in the *partial measurement case* these expressions are considered for a selected subset of node signals. The Identifiability



Network: Full [✓ Single module] — Target module: G1,2

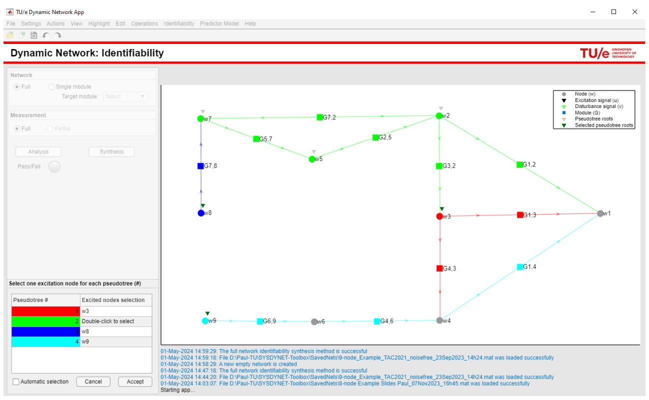
Measurement: Full [✓ Partial]

Node	Selected	Measured
w1	✓	✓
w2	✓	✓
w3	✓	✓
w4		✓
w5		✓
w6		✓
w7		✓

```
15-Nov-2023 11:34:24:  The identifiability test is passed based on r-excitations only
15-Nov-2023 11:34:24:  The single module identifiability test passed
15-Nov-2023 11:33:45:  The identifiability test is passed based on r-excitations only
15-Nov-2023 11:33:45:  The single module identifiability test passed
15-Nov-2023 11:33:16:  The Full network identifiability test passed
15-Nov-2023 11:32:53:  Nodes with an identifiability problem: w7
15-Nov-2023 11:32:53:  The full network identifiability test failed
```

Fig. 3. Identifiability window - Analysis: Analyzing single module identifiability of module G_{21} on the basis of partial measurements.

window has options for either **Analysis** or **Synthesis** of identifiability conditions. The Analysis option typically provides a yes/no answer dependent on whether the identifiability conditions are satisfied. The Synthesis option provides suggestions to the user for additional actions, e.g. adding external excitation signals at particular locations, for satisfying the (generic) identifiability conditions. Generic identifiability of a target module typically requires



Select one excitation node for each pseudotree (#)

Pseudotree #	Excited nodes selection
🟥 1	w3
🟩 2	Double-click to select
🟦 3	w8
🟦 4	w9

```
01-May-2024 14:59:29: The full network identifiability synthesis method is successful
01-May-2024 14:59:18: File D:\Paul-TU\SYSDYNET-Toolbox\SavedNets\9-node_Example_TAC2021_noisefree_23Sep2023_14h24.mat was loaded successfully
01-May-2024 14:58:29: A new empty network is created
01-May-2024 14:47:18: The full network identifiability synthesis method is successful
01-May-2024 14:44:20: File D:\Paul-TU\SYSDYNET-Toolbox\SavedNets\9-node_Example_TAC2021_noisefree_23Sep2023_14h24.mat was loaded successfully
01-May-2024 14:03:07: File D:\Paul-TU\SYSDYNET-Toolbox\SavedNets\8-node Example Slides Paul_07Nov2023_16h45.mat was loaded successfully
Starting app...
```

Fig. 4. Identifiability window - Synthesis: allocating external signals so as to warrant identifiability of the full network in the full measurement case, on the basis of a pseudotree-covering of the network graph (Cheng et al. (2022)). If a root of each pseudotree is excited by either an r or a v signal, generic network identifiability is guaranteed. In the example this holds for excitations on nodes w_3, w_8, w_9 and $(w_2 \cup w_5 \cup w_7)$.

a sufficient number of external signals to be present in the network. Sufficient excitation can be provided by either excitation signals r, or disturbance signals v. If single module identifiability is guaranteed by excitation of r-signals only, then an *indirect method* for single module identification can be applied in the Predictor Model window. The user is informed about this in the user communication panel. If both $r-$ and $e-$signals are exploited for single module identifiability, a *direct method* for identification is the natural choice.

Full network identifiability analysis has been introduced in Weerts et al. (2018b). Based on the graph-based results of Hendrickx et al. (2019), the notion of generic identifiability was developed. The single module identifiability results based on full measurements were developed in Shi et al. (2022) and for partial measurement in Shi et al. (2023b). The synthesis algorithm for allocating external excitation signals for full network identifiability, is based on covering the graph of the network with pseudotrees/SIMUGs, and requiring an external signal at a root of each pseudotree/SIMUG. The corresponding algorithms were developed in Cheng et al. (2022) and Dreef et al. (2022), effectively taking account of a priori known modules.

2.5 Predictor Model Window

In the Predictor Model Window, predictor models can be synthesized and analyzed, for consistently estimating a selected target module in the network, utilizing knowledge of the underlying structure/topology of the network. A predictor model is characterized by the following equation:

$$w_{out}(t) = \bar{G}(q)w_{in}(t) + \bar{H}(q)\xi_{\mathcal{Y}}(t) + \bar{T}(q)r_{in}(t), \qquad (4)$$

where node signals can appear both in the output w_{out} and the input w_{in}. This is typically possible for handling confounding variables [2]. In the Predictor Model Window, the structure/topology of the matrices in (4) are considered in terms of their corresponding adjacency matrices, as well properties of the present links/modules, such as parametrized/known and proper/strictly proper.

There are three types of identification methods for which predictor models can be constructed:

(1) the local **direct method** (Ramaswamy and Van den Hof, 2021), that is based on a predictor model with w-nodes as predictor inputs and w-nodes as predicted outputs, and appropriate handling of external excitation signals. This method can end up with a MIMO (multi-input, multi-output) predictor model.
(2) the **multi-step method** (Fonken et al., 2023), that is based on a similar predictor model, but that uses a nonparametric step to estimate innovation signals first, which are then used as measured inputs in a parametric estimation. This method reflects an alternative way of handling confounding variables and always ends up with a MISO (multi-input, single-output) predictor model with w-nodes as predictor inputs and a w-node as predicted output.
(3) the **indirect method** (Gevers et al., 2018; Shi et al., 2022), that is based on a predictor model with r-nodes as inputs and w-nodes as predicted outputs. It requires post-processing of the identified predictor model in order to arrive at a target module estimate.

Besides the central network graph, there are two main panels in the predictor model window:

- In the **left window panel**, predictor models are being constructed (synthesized), according to particularly chosen algorithms, while constructed predictor models can be `Accepted` and consequently stored in the Stored Predictor Models (right upper) panel.
- In the **right window panel**, a selected predictor model from the Stored Predictor Models panel can be manually edited, and analyzed in terms of its consistency conditions.

Consistent identification of a single module in a network requires the satisfaction of three types of conditions:

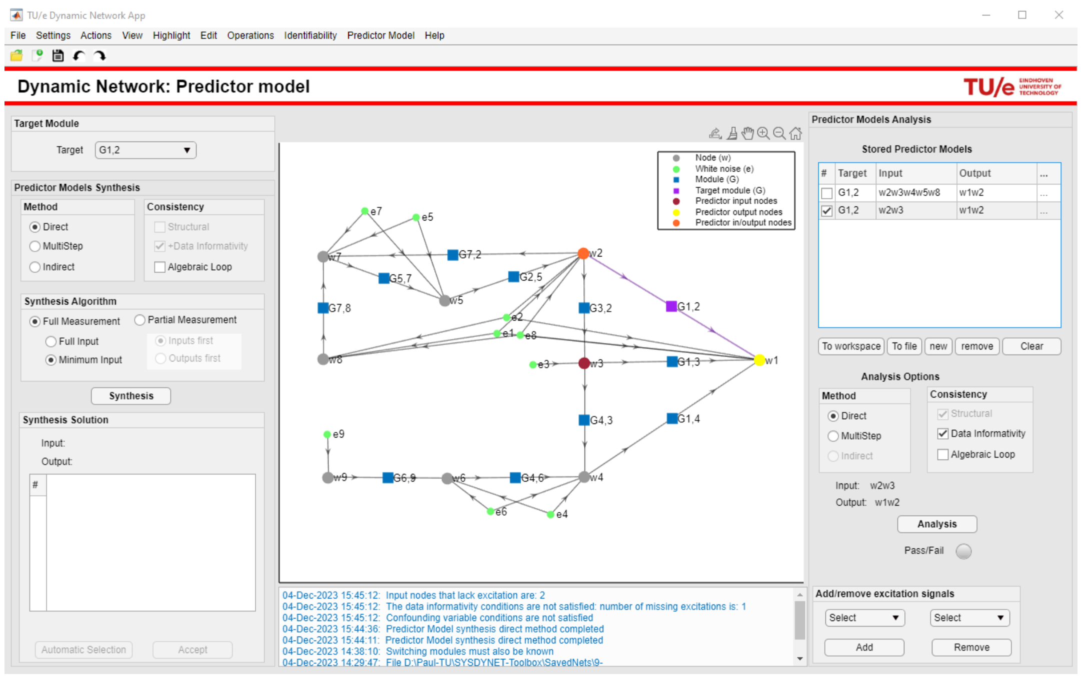
(1) **Structural conditions** on the network topology, that encompass:
    (a) Conditions for module invariance, covered by the parallel path and loop condition;
    (b) Conditions on the absence of confounding variables between particular sets of nodes;
(2) **Data Informativity** conditions, requiring sufficient external signals to be present in the network;
(3) Absence of **Algebraic Loops** in particular parts of the network (not for the indirect method).

When synthesizing a predictor model, satisfying the conditions of which the check boxes are checked, is guaranteed. For condition 2 this means that the predictor model has the structural capability to satisfy data-informativity. It does not imply that these external signals are indeed present. Adding external excitation signals can be done in the analysis panel.

Synthesis algorithms

For the **direct method** there are four synthesis algorithms between which the user can choose for constructing a predictor model. Two algorithms are based on the **Full**

[2] Confounding variables are unmeasured signals that affect both the input and the output of an estimation problem.



#	Target	Input	Output	...
☐	G1,2	w2w3w4w5w8	w1w2	...
✓	G1,2	w2w3	w1w2	...

```
04-Dec-2023 15:45:12:  Input nodes that lack excitation are: 2
04-Dec-2023 15:45:12:  The data informativity conditions are not satisfied: number of missing excitations is: 1
04-Dec-2023 15:45:12:  Confounding variable conditions are not satisfied
04-Dec-2023 15:44:36:  Predictor Model synthesis direct method completed
04-Dec-2023 15:44:11:  Predictor Model synthesis direct method completed
04-Dec-2023 14:38:10:  Switching modules must also be known
04-Dec-2023 14:29:47:  File D:\Paul-TU\SYSDYNET-Toolbox\SavedNets\9-
```

Fig. 5. Predictor model window. For estimating target module G_{12} with the direct method, a predictor model is selected with inputs (w_2, w_3) and outputs (w_1, w_2). Inputs are colored red, outputs yellow, and nodes that appear both as input and output are colored orange. In the left panel predictor models constructed/synthesized through different algorithms that guarantee that the main structural properties for consistency are satisfied. In the right upper panel constructed predictor models are stored and can be edited. In the right lower panel predictor models can be analyzed, e.g. by verifying whether data-informativity conditions are satisfied. The workflow supports the online addition/removal of external excitation signals from the network.

Measurement situation, i.e. irrespective of the measured status of nodes, all nodes in the network are assumed to be available for the predictor model. And two algorithms are implemented for the **Partial Measurement** case, where only the nodes that have the "measured" status can be taken into account. The first three algorithms are presented in Ramaswamy and Van den Hof (2021), while the fourth one ("Outputs first") is presented in Shi et al. (2023a).
For the **multi-step method** there is one algorithm that is a partial measurement algorithm, in the sense that all measured nodes are taken into consideration of the first step of the algorithm, and the predictor that is constructed is the predictor model for the final parametric step.
For the **indirect method** there is one algorithm implemented that is based on full measurements and a MISO (multi-input, single output) predictor model.

Selected predictor models are visualized in the network graph by coloured nodes for the inputs and outputs.

Predictor model analysis

In the **Stored Predictor Models** panel, different predictor models can be stored, edited, and selected for analyzing whether they satisfy the properties for consistent estimation of the target module. If data informativity requires the addition of external excitation signals, it is reported in the communication window where excitation signals are missing. The user can then manually add/remove excitation signals to/from the network.
Selected predictor models can be saved to workspace or stored in a file, as a `nwpredmodel` object (see Section 3).

3. DATA STRUCTURES

The data that is typically related to a dynamic network involves different types of signals. In comparison with a "classical" open-loop or closed-loop multivariable identification problem it involves possibly multiple external excitation signals, allocated at assigned locations in the network. Additionally, the network predictor models may include signals that appear as both input and output in the predictor model. This causes the currently available data structures for data and model objects in MATLAB's System Identification Toolbox (The Mathworks, Inc., 2021) to be less suitable. Inspired by these data and model objects, dedicated data structures are introduced in the current toolbox as MATLAB classes:

- **LabelledAdjStructure**: Network structure object that stores the network topology and its properties, see section 2.2;
- **nwdata**: Network data object to encapsulate all node/excitation data of networked systems and their properties, analogous to MATLAB's **iddata** object;
- **nwpredmodel**: Network predictor model class, which specifies the mapping between nodes and excitations of a network structure to input and outputs of a predictor model (4), including the specification of known/parametrized terms, and estimated values of parameters.
- **nwmodel**: Network model class: special case of the network predictor model class, but where a full network is represented, including all nodes and external

signals, as in (1). It is used in full network identification and in network simulation.

4. ADDITIONAL TOOLBOX FUNCTIONS

The methods and tools from the App are also available as m-files in the MATLAB command window. Additional command line functionality that is not (yet) supported by the graphical App:

- **nwsimulate**: Simulation of all node signals in a network model when given external signals (r- and e signals), applied to a network in the `nwmodel` class.
- **nwidfullSLS**: Identification of a full network, based on an `nwdata` and `nwmodel` object, using the Sequential Least Squares method, as presented in Weerts et al. (2018a).
- **nwidPEM**: Identification of a single module of a full network with the local direct (PEM) method (Ramaswamy and Van den Hof (2021)).
- **nwidsingleMultiStep**: Identification of a single module with the multistep method (Fonken et al. (2023)).

5. FUTURE EXTENSIONS

Identification algorithms, including topology estimation algorithms, will be further extended, as well as integrated in the App through the introduction of an *Identification Window*. Extension of the network model setup from the current *module framework* with a second framework determined by *diffusive couplings* between nodes, is also foreseen. This latter step turns the directed graph of a network into a non-directed graph, see Kivits and Van den Hof (2023).

6. CONCLUSIONS

A MATLAB App and Toolbox has been presented for system identification in dynamic networks, represented in a module framework. It allows to address data-driven modeling problems in interconnected dynamic systems. Currently the focus is on the preparation phase of the identification problem, addressing the construction of a suitable predictor model on the basis of knowledge of the structure/topology of the network, and addressing the question of generic identifiability of either a single module or a full network. First algorithms for simulating and estimating a full network or a single module have been added as command line instructions. More algorithms for actual data-driven modeling and model validation will be added in a future release.

REFERENCES

Cheng, X., Shi, S., and Van den Hof, P.M.J. (2022). Allocation of excitation signals for generic identifiability of linear dynamic networks. *IEEE Trans. Automatic Control*, 67(2), 692–705.

Dankers, A.G., Van den Hof, P.M.J., Heuberger, P.S.C., and Bombois, X. (2016). Identification of dynamic models in complex networks with prediction error methods: Predictor input selection. *IEEE Trans. on Automatic Control*, 61(4), 937–952.

Dreef, H.J., Shi, S., Cheng, X., Donkers, M.C.F., and Van den Hof, P.M.J. (2022). Excitation allocation for generic identifiability of linear dynamic networks with fixed modules. *IEEE Control Systems Letters (L-CSS)*, 6, 2587–2592.

Fonken, S.J.M., Ramaswamy, K.R., and Van den Hof, P.M.J. (2023). Local identification in dynamic networks using a multi-step least squares method. In *Proc. 62th IEEE Conf. on Decision and Control (CDC)*, 431–436. IEEE, Singapore.

Gevers, M., Bazanella, A., and Vian da Silva, G. (2018). A practical method for the consistent identification of a module in a dynamical network. *IFAC-PapersOnLine*, 51-15, 862–867.

Gonçalves, J. and Warnick, S. (2008). Necessary and sufficient conditions for dynamical structure reconstruction of LTI networks. *IEEE Trans. Automatic Control*, 53(7), 1670–1674.

Hendrickx, J., Gevers, M., and Bazanella, A. (2019). Identifiability of dynamical networks with partial node measurements. *IEEE Trans. Autom. Control*, 64(6), 2240–2253.

Kivits, E.E.M. and Van den Hof, P.M.J. (2023). Identification of diffusively coupled linear networks through structured polynomial models. *IEEE Trans. Automatic Control*, 68(6), 3513–3528.

Materassi, D. and Innocenti, G. (2010). Topological identification in networks of dynamical systems. *IEEE Trans. Automatic Control*, 55(8), 1860–1871.

Ramaswamy, K.R. and Van den Hof, P.M.J. (2021). A local direct method for module identification in dynamic networks with correlated noise. *IEEE Trans. Automatic Control*, 66(11), 5237–5252.

Shi, S., Cheng, X., De Schutter, B., and Van den Hof, P.M.J. (2023a). Signal selection for local module identification in linear dynamic networks: A graphical approach. *IFAC-PapersOnLine*, 56-2, 2407–2412.

Shi, S., Cheng, X., and Van den Hof, P.M.J. (2022). Generic identifiability of subnetworks in a linear dynamic network: the full measurement case. *Automatica*, 137(110093).

Shi, S., Cheng, X., and Van den Hof, P.M.J. (2023b). Single module identifiability in linear dynamic networks with partial excitation and measurement. *IEEE Trans. Automatic Control*, 68(1), 285–300.

The Mathworks, Inc. (2021). *Matlab's System Identification Toolbox.*

Van den Hof, P.M.J., Dankers, A.G., Heuberger, P.S.C., and Bombois, X. (2013). Identification of dynamic models in complex networks with prediction error methods - basic methods for consistent module estimates. *Automatica*, 49(10), 2994–3006.

Verhaegen, M., Yu, C., and Sinquin, B. (2022). *Data-driven identification of networks of dynamic systems.* Cambridge University Press, Cambridge, UK.

Weerts, H.H.M., Galrinho, M., Bottegal, G., Hjalmarsson, H., and Van den Hof, P.M.J. (2018a). A sequential least squares algorithm for ARMAX dynamic network identification. *IFAC-PapersOnLine*, 51-15, 844–849.

Weerts, H.H.M., Van den Hof, P.M.J., and Dankers, A.G. (2018b). Identifiability of linear dynamic networks. *Automatica*, 89, 247–258.