Numerical Test Rig for Large-Scale and Interconnected Dynamical Systems

submitted
Project Laboratory
Networked and Cooperative Control
by

cand. ing. Francisco Llobet
born on December 18, 1986
resident:
Amalienstr. 87
80799 München

cand. ing. Jose Rivera
born on December 18, 1986
resident:
Amalienstr. 87
80799 München

Institute of
Automatic Control Engineering
Technische Universität München

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Sandra Hirche

Supervisor: F. Deroo, S. Erhart, A. Gusrialdi, H. Mangesius
Beginn: 09.05.2011
Submitted 04.07.2011

Abstract

The goal of this project was to develop a test rig for large-scale and interconnected dynamical systems. The result is MTIDS or Matlab Toolbox for Interconnected Dynamical Systems, which is a mash-up that wraps different toolboxes used for graph analysis and dynamic systems simulation together. MTIDS allows the definition and analysis of graphs, where each node has a specific dynamic assign to it. The template based design of nodes' dynamics allows great flexibility for the creation of complex interconnected dynamical systems with the possibility of implementing clusters/layers. MTIDS is an open-source project under the GNU GPL v2 license. This document presents a general description of MTIDS and instructions for its use.

Contents

Chapter 1

Introduction

In this first Chapter the motivation behind the MTIDS project is explained and the project's goal and framework is presented.

1.1 Motivaion

Large-scale interconnected dynamical system are everywhere: biological systems, power and water systems, the brain neurons, social interaction networks, economic markets, etc. In a canonical form all of this systems can be thought as a bunch of nodes with local dynamics that interact with each other, e.g. a graph. Different topologies of the graph, may lead to different behavior. An example of various large scale interconnected systems can be seen in Figure 1.1.

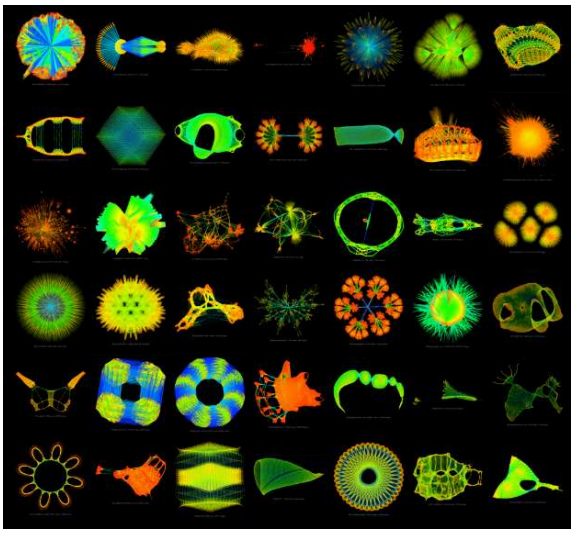

Figure 1.1: Visualization of various large scale systems using the sfdp algorithm © Dr. Yifan Hu of AT&T Labs

There are many tools available for the analysis of interconnected dynamical systems,

for example in power systems you have PSSE and Power Factory. However, this simulation programs are normally very system specific and in most cases it takes a long time to learn how to use them correctly. The difficulties are specially noticed while testing control concepts, where small changes on the topology of the grid or control concept could lead to a painful redesign of your simulation set up. You may actually end up spending the most of your time in the implementation of a simulation. A more general and easy to use solution for the simulation of interconnected dynamical systems is needed.

1.2 Idea and Goal

MTIDS (Matlab Toolbox for Interconnected Dynamical Systems) is a project that aims to design an easy to use and flexible toolbox to make the simulation of large scale dynamical systems easier for students and researchers. The **goal** is to produce a mash-up that wraps different toolboxes used for graph analysis and dynamic systems simulation together into a framework.

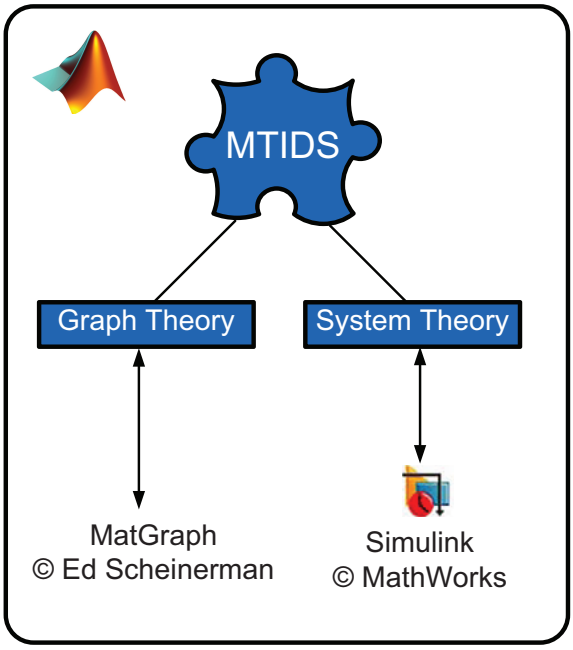

Figure 1.2: MTIDS: Matlab Toolbox for Interconnected Dynamical Systems

As we can see in Figure 1.2 MTIDS runs in the MATLAB environment and is basically a GUI that allows the interaction of tools used in graph theory and control theory. For graph theory we use Matgraph a toolbox design by Prof. Scheinerman of the John Hopkins University and for dynamical simulations we use Simulink.

1.3 Framework

The current framework of MTID is made out of three basic components. A GUI (**mtids.m**) an export to simulink function (**exportSimulink.m**) and an import from Simulink function (**importSimulink.m**).

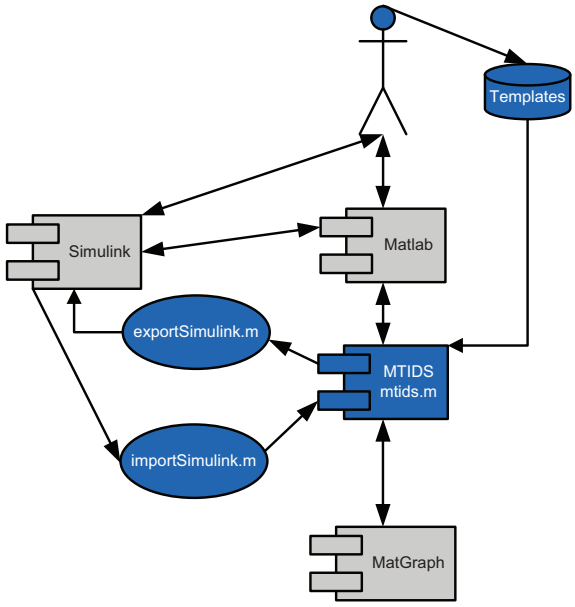

Figure 1.3: MTIDS: Components diagram

In Figure 1.3 we can see that the most important component is the user, specially its head. The better you are at producing templates and interacting with matlab and simulation the more functional MTIDS is going to be for you. In a nutshell MTIDS works as follows:

- GUI (mtids.m) runs inside Matlab
- GUI interacts with Matgraph: create, modify and visualize graphs
- System Inteconnector (SI): exportSimulink.m and importSimulink.m called from GUI to interact with Simulink
- Templates done by User in Matlab/Simulink.
- Simulations done in Simulink

Chapter 2

Graph Theory

Chapter 3

System Theory

In this Chapter we explain the design and simulation capabilities that MTIDS offers for interconnected dynamical systems.

3.1 Export to Simulink

In order to export the model created in the MTIDS GUI to simulink: **Simulation → Export to Simulink...** (see Figure 3.2)

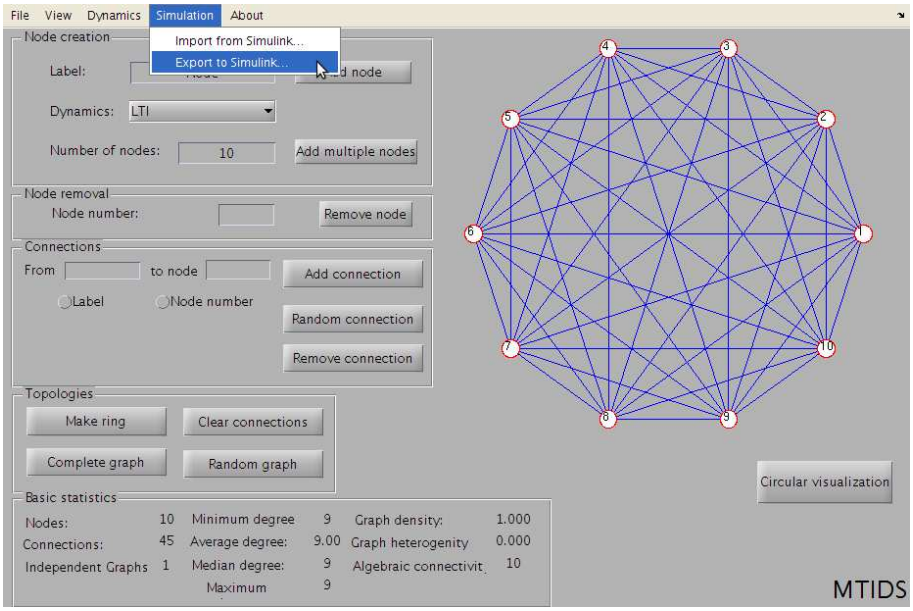

Figure 3.1: MTIDS: export model to Simulink. Example is a complete graph with 10 LTI nodes.

The MTIDS GUI then calls the function exportSimulink.m which builds a Simulink model based with the following information:

- name: Model name
- template: list of nodes' dynamics templates
- templateList: List of templates that are availible in mtids
- A: Adjacense matrix,
- xy: Position of the nodes
- labs: List of nodes' names

The result is a Simulink model in the MTIDS format. To adjust the size of the model to the Simulink window size: **View** → **Fit System To View**

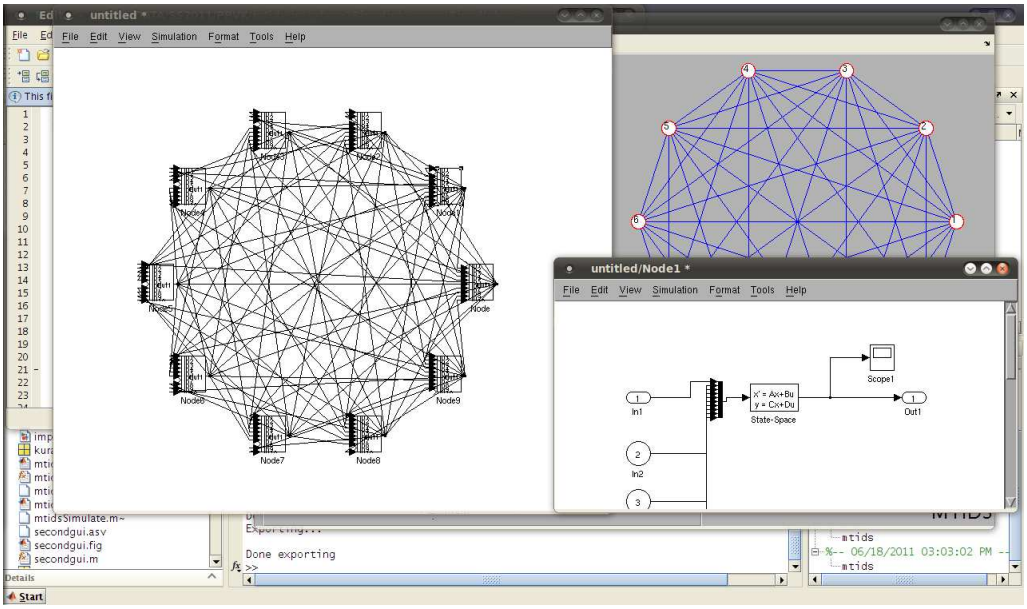

Figure 3.2: Interconnected system Simulink Model. Example is a complete graph with 10 LTI nodes.

In Figure 3.2, the nodes are odered in a circle, the topology of the system defined in MTIDS remains. The circle arrangement is a design descition, made, in order to allow a better acces to the nodes. Each one of the nodes is a subsystem block. The dynamic of the nodes is defined inside the subsystem block. Moreover, each node has a single out-port and N in-ports, where N is the number of nodes the whole system has. Each node on the system has its own in-port on each node, compare with a zoom on the first node of our example in Figure 3.3. Notice that the port corresponding to the nodes number should always be free, e.g. the first node has an unconnected in-port 1, the second node an unconnected in-port 2, etc. In case that a loop of a node with itself is required we recommend doing this inside the subsystem/node.

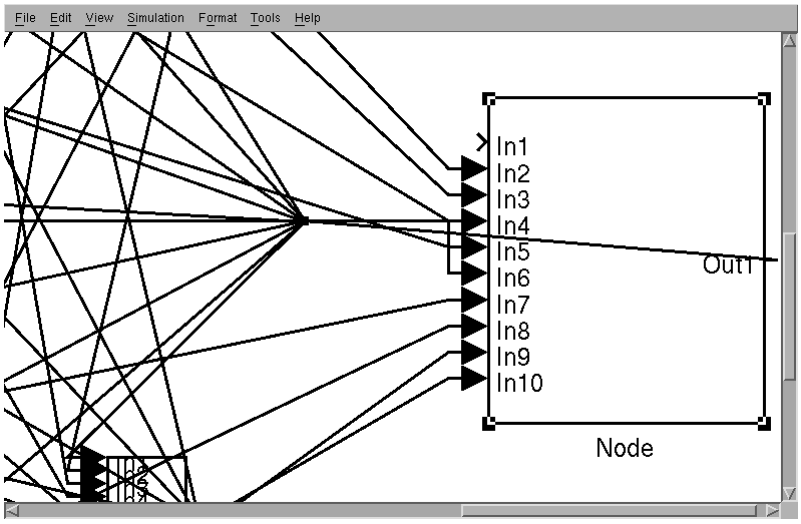

Figure 3.3: Node inputs and output: Each node has an in-port for every node in the model and an out-port. Example is the first node of a complete graph with 10 LTI nodes.

Another important aspect to point out is that when simulating the system, simulink will issue a warning for unconnected in-ports and set their input to the subsystem to 0, this means that an **unconnected ports enter the subsystem (local dynamics of the node) with a value of zero** during simulations.

3.2 Nodes' Dynamics

As mentioned in the past section 3.1 each node defined in MTIDS is exported to simulink as a subsystem. The exportSimulink.m function uses predefined dynamic templates, which are modified accourding to the in MTIDS defined topology. Next we explain how to define your own templates and show how to use 2 preloaded templates: the LTI template and the kuramoto template.

3.2.1 Build your own Template

The real power of MTIDS depends on you ability to build templates.

In Figure 3.4 we can see the components that each template should have: an in-port connected to a mux and an out-port. Between the mux and the outport you can design your own custom dynamics. The imput to the system comes from mux as vector composed by the values send from other nodes/subsystems. The output of your system should also be aggregated to a vector and routed to the out-port. The sepparation of the different inputs and outputs is left to the system designers. With a well thought architecture complex systems are very easy to achieve, please refer

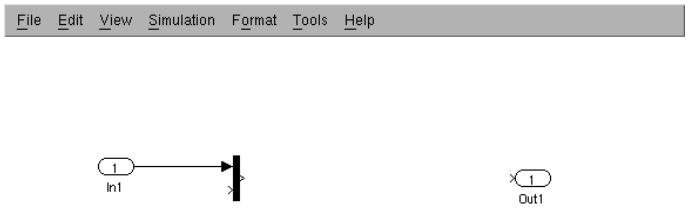

Figure 3.4: Scratch template for node's dynamics.

to the LTI and kuramoto template examples.

The dynamic of a node can be as simple as a junction that only reroutes the incomming signals or more complicated to include controlers and systems inside of it. It is this feature that allows the implementation of clusters or layered systems, see subsection 3.3.

3.2.2 The LTI template

The LTI template is an example that defines a linear time invariant dynamic for nodes. The mathematical model of a simple interconnected LTI system is written as:

$$\begin{pmatrix} \dot{x_1} \\ \vdots \\ \dot{x_N} \end{pmatrix} = \begin{pmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{N1} & \cdots & A_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} + \begin{pmatrix} B_1 & \cdots & B_N \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix} \tag{3.1}$$

or

$$\dot{x} = Ax + Bu \tag{3.2}$$

The local view of a single subsystem/node (here for the first node) is:

$$\dot{x_1} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} & B_1 & \cdots & B_N \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \\ u_1 \\ \vdots \\ u_N \end{pmatrix} \tag{3.3}$$

this can be reformulated

$$\dot{x_1} = A_{11}x_1 + \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} & B_1 & \cdots & B_N \end{pmatrix} \begin{pmatrix} 0 \\ x_2 \\ \vdots \\ x_N \\ u_1 \\ \vdots \\ u_N \end{pmatrix} \tag{3.4}$$

In order to keep things simple, we define the local output of all nodes as:

$$y_N = x_N \tag{3.5}$$

To implement this dynamic as a template we make use of the State-Space block.

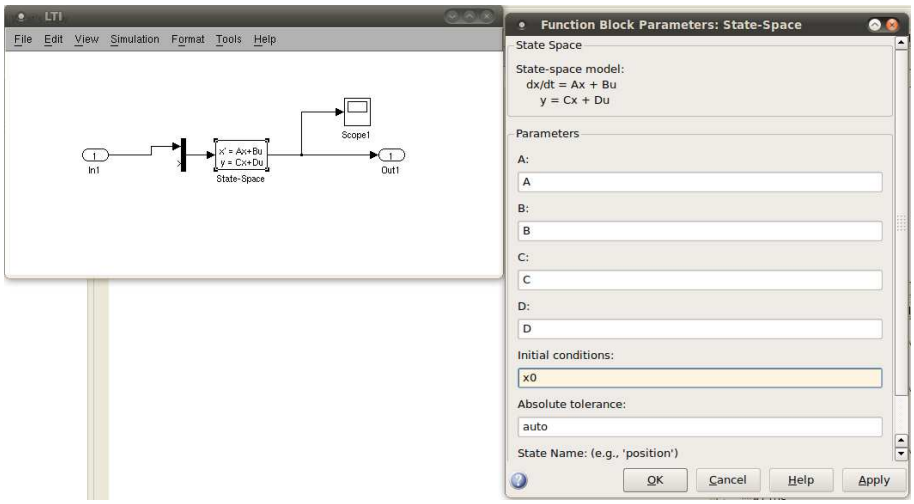

Figure 3.5: LTI template

This template shown in 3.5 can be used in MTIDS to create interconnected LTI systems with different topologies. One way to avoid the work of parametrizing the A,B,C and D matrix of every node sepparately is to define the same matrix for all subsystems. One can use the formula 3.4 And define the same A, B, C, and D matrix for every node. An example could be

$$\begin{aligned} A &= 1 \\ B &= [1 \cdots 1]^{1\times N} \\ C &= 1 \\ D &= [0 \cdots 0]^{1\times N} \end{aligned}$$

As the input of the system is in-port of the node's itself is automatically set to zero we can define this matrizes for all nodes without getting any errors.

For more complicated LTI models we recomend creating manydifferent LTI templates with different behavior.

3.2.3 The Kuramoto Template

The kuramoto template is an example for a non-linear dynamic of a node.

3.3 Layering/Clustering Nodes

3.4 Working in Simuling

3.5 Import from Simulink

Beware to define nodes dynamics...

Chapter 4

Conclusion and Future Development

4.1 Conclusion

4.2 Future Development

List of Figures