The Journal of Open Source Education

StateMint: A Set of Tools for Determining Symbolic Dynamic System Models Using Linear Graph Methods

Cameron Devine[1], **Joseph L. Garbini**[1], **and Rico A. R. Picone**[2, 1]

1 University of Washington, Department of Mechanical Engineering **2** Saint Martin's University, Department of Mechanical Engineering

DOI: 10.21105/jose.00044

Software

- Review
- Repository
- Archive

Submitted: 29 December 2018
Published: 09 April 2019

Abstract

StateMint is a set of software tools that reduce sets of dynamic equations and their constraints to a state-space model and related dynamic system model formulations. These tools are especially useful for students of system dynamics, many of whom can become lost in this algebraic reduction. StateMint includes a Mathematica package, a Python package, and a web interface that is built as a layer on top of the Python package.

Introduction

When deriving a system's state-space model—that is, the vector state (differential) equation and the vector output (algebraic) equation—in the manner of Rowell and Wormley[1] (Rowell & Wormley, 1997), one begins by forming scalar equations for each lumped-parameter element, describing its dynamics; we call these "elemental equations." We assume power enters and exits each element through a finite number of ports, usually one. Let the total number of ports in a given system be N, which is equal to the number of elemental equations. N constraint equations that describe the topology of the system can be defined by the interconnection of the elements. A set of $2N$ differential and algebraic equations and $2N$ unknown variables result. If properly constructed (Rowell & Wormley, 1997), N of the unknown variables can be immediately eliminated through direct substitution. Finally, the set of equations can be reduced to a system of first-order differential equations in state and input variables and their time-derivatives, alone.

Statement of Need

In our experience,[2] a student manually reducing the set of equations, will often make some minor mistake during the tedious process of the last two steps above. This is typically of a "book keeping" variety that, if it teaches the student anything, it is not system dynamics. Instead, the student can be easily discouraged and confused about where they have made their mistake. Furthermore, the tedious nature of the reduction can dampen a student's desire to explore interesting problems. The software tools presented here automate this algebraic reduction. They allow students to focus on understanding the process of dynamic system modeling and to experience a wider breadth of problems.

[1] This method is also described in the tutorial.
[2] Collectively, we have almost 30 years of experience teaching system dynamics.

Software Tools

Originally, utilizing the advanced symbolic mathematics capabilities of Mathematica, a package was written to perform these algebraic reductions. However, it requires students to install and learn the basics of Mathematica. So a web interface was designed to allow students to use this tool without any knowledge of programming by allowing equations to be entered with notation similar to that of many scientific calculators. To support this interface, a Python package was written with the same functionality as the Mathematica package and is deployed as an Amazon AWS Lambda function for use by the web interface. This interface can be accessed by any device with an internet connection and a web browser.

Web Interface

The web interface (statemint.stmartin.edu) has text boxes for entering equations and variables. A special form of the constraint equations is required, as described in the tutorial, based on the text of Rowell and Wormley (Rowell & Wormley, 1997). Once entered, the equations are sent to the Lambda function and the dynamic system model is returned. The results are then displayed as rendered math or source code in any of the following languages: LaTeX, Matlab, Python, and Mathematica. Examples and documentation are built-in, allowing the user to learn the interface as they use it. The user input can be shared, downloaded, and saved for later use or modification. Because this interface utilizes Amazon AWS serverless resources, required maintenance and costs are minimized. An automated installer for independent deployment of the website is also included in the StateMint repository.

Python Package

The Python package uses the SymPy library (Meurer et al., 2017) to symbolically reduce the set of elemental and constraint equations to the state and output equations. The main function, `StateMint.Solve`, accepts the input variables, state variable elemental equations, other elemental equations, constraint equations, and output variables. This function returns an object that includes the resulting system as a state-space model and (when applicable) a transfer function. Auxiliary functions, `StateMint.to_numpy.array` and `StateMint.to_numpy.matrix` are included to convert the SymPy symbolic matrices to NumPy (Oliphant, 2015) arrays or matrices respectively. These functions accept a symbolic matrix and a dictionary mapping system parameters to numeric values. The StateMint package is documented at statemint.readthedocs.io and works for both linear and nonlinear systems.

A detailed example of how to use the Python StateMint package is included in the StateMint repository.

Mathematica Package

The Mathematica package `StateMint` can be installed as described in the documentation. The central function of the package is `stateEquations`, which uses an algorithm similar to that of the Python package, above, to derive the state equations. It takes as arguments lists of elemental equations, constraint equations, primary variables, and input variables

and returns the vector state equation, state variables, and the time-derivative of the state variables.

The `outputEquations` function derives the output equations given output expressions in terms of primary and secondary variables (including inputs). The function accepts lists of input variables, state variables, elemental and constraint equations, and output expressions.

The functions `stateEquations` and `outputEquations` yield what are in general *nonlinear* state and output equations. Linear state and output equations are typically written in a standard vector form described by matrices `A`, `B`, `C`, and `D` (and sometimes `E` and `F`) (Rowell & Wormley, 1997). The `linearizeState` function accepts lists of input variables, state variables, and the time-derivatives of the state vector (from `stateEquations`) and returns the `A`, `B`, and `E` matrices. Similarly, `linearizeOutput` returns the `C`, `D`, and `F` matrices.

A detailed example of how to use the Mathematica StateMint package is included in the StateMint repository. This package is best used by those who are already familiar with Mathematica, or for more complex problems where Mathematica may perform better than SymPy.

Acknowledgments

The authors would like to acknowledge the work of Gavin Basuel who designed the user experience for the web interface and helped with HTML/CSS development.

References

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., et al. (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. doi:10.7717/peerj-cs.103

Oliphant, T. E. (2015). *Guide to numpy* (2nd ed.). USA: CreateSpace Independent Publishing Platform.

Rowell, D., & Wormley, D. N. (1997). *System dynamics: An introduction*. Prentice Hall.