[[1]](#footnote-1)

APMonitor Model Initialization from SBML Format in IEEE TRANSACTIONS Format

(December 2018)

Matthew W. Bradley, *Student BioEn 6760*

*Abstract*—This paper lays presents current efforts on initializing APMonitor models directly from an SBML formatted biomodel. SBML is a widely used modelling format for biological processes as diverse as genetic circuits to modelling bone regrowth and insulin production in the body. APMonitor is a dynamic optimization suite developed for the chemical industry. APMonitor. APMonitor’s suite includes modes for parameter estimation, and optimization as well as simulation. It’s strengths in these areas can nicely complement the rich abilities of SBML models by providing a specialized set of tools for ODE or PDE based models that are good candidates for optimization e.g. synthetic production of insulin. The paper presents the methods used in conversion, and some test cases.

*Index Terms*—APMonitor, SBML, conversion tool, signal processing, dynamic optimization.

# INTRODUCTION

W

hile synthetic biology has made great strides in recent years, there are still barrier to translating genetic circuits into products. Outside of the lack of knowledge of all possible parts and their interactions important barriers include unwieldy complexity and variability in performance [1]. While the passage of time is likely to solve the lack of knowledge of biological systems, the development of more tools to control variability and simplify models is needed. APMonitor is one such possible tool.

# Background/Aims

Synthetic biology has already been heralded as "The Next Pillar of Medicine" [2] because of the ability to insert recombinant DNA into non-human systems i.e. bacteria, and produce human proteins with therapeutic importance, such as insulin. This method is advantageous because it produces a bio compatible therapeutic without using dangerous chemicals. The challenges described in the Introduction however limit the scope of applications. APMonitor and its Python implementation, Gekko as described below are excellent tools for the efficient modelling and optimization of production from such synthetic methods.

## Aims

The aims of this project consist in two parts: i.) create a library of functions that constructs a Gekko model in a .py format from an SBML file and ii.) transfer these functions into a downloadable module on github that allows the user to set additional options unique to Gekko/APMonitor. Aim i was tested by simulation of Gardener’s 2000 genetic toggle switch model [3] and aim ii was to be tested by setting a biomodels to a model-predictive control application.

## SBML

SBML is an XML-based language for modelling biological processes that is a free and open standard [4]. It’s latest release version is Level 3 Version 1 and has capabilities of representing metabolic networks, cell signaling pathways, regulatory networks, infectious diseases and genetic logic circuits. It was developed in late 1999 to 2000 and has since been suggested as a standard for representing computational models in systems biology today [5]. There are many features of SBML as it has evolved over the last 20 years, and it is especially rich in events analysis. For the scope of this paper events were largely ignored and the basic skeleton of the model i.e. it’s parameters, species, rate laws and rules were focused on.

## APMonitor

APMonitor or Advanced Process Monitor is a modeling language for differential algebraic equations. Its optimization suite is coupled with large-scale solvers for linear, quadratic, nonlinear, and mixed integer programming [6]. There are 9 different modes of operation steady-state simulation, model parameter update, real-time optimization, dynamic simulation, moving horizon estimation, nonlinear control, sequential dynamic simulation, sequential dynamic estimation and sequential dynamic optimization. When running the model APMonitor compiles the model to byte-code and performs reduction based on analysis of the sparsity structure. For differential and algebraic equation systems, orthogonal collocation on finite elements is used to transcribe the problem into a purely algebraic system of equations.

## Gekko

Gekko is the Python module implementation of APMonitor [7]. It contains the same modes and features as APMonitor, it is a wrapper that allows the user to more easily build the model and run it. When Gekko initiates the solving of a model, it is written into an APMonitor format, and once solved it is written into a .json format that is how the variables are accessed.

# MEthods

Both APMonitor and SBML have many unique features, and the overlap of features they do share are frequently not in the same format. Functions were developed to read and store the bare-bones structure of the SBML model and algorithms were then implemented to transfer these elements into a Gekko model in Python. The function library was developed in Notepad++ though any editor environment would have sufficed.

## Biomodels

Three biomodels were obtained for the purposes of developing, building and testing the conversion tool. Two biomodels were obtained from the European Bioinformatics Institutes’ Biomodels Database [8,9], and the genetic toggle switch model was created in iBioSim[10].

## Reading and Storing Data

LibSBML python package [11] was used to read SBML files into the Python coding environment. LibSBML is a conversion api developed by the team that designed SBML file formats and is available as a PyPI package. A new class, Bionetwork\_Model was created to store the data that was contained in the model. The initiator for this class accepted a file path name to the SBML model.

## Bionetwork\_Model Class

The Bionetwork\_Model class has several attributes, most of which are pandas DataFrame objects to store the information to later be used in models and in writing the model. The attributes are: CV, MV, FV, SV, species, variables, parameters, rules, kinetic, and stoich\_table.

Briefly the first four are control variables, manipulated variables, fixed variables and state variables that correspond to APMonitor variable classes of the same name. They are initiated to be an empty set, and populated only when the change\_var\_type method (described below) is used. The species and variables attributes are initialized to the same Pandas DataFrame that contains all the species id’s, names and initial values. The species DataFrame is duplicated into the variables DataFrame so that there is always a DataFrame that contains all the variables (the species DataFrame) while the variable DataFrame will be updated to reflect which species are kept as a variable and not a special class of variables. The parameters attribute is also a pandas DataFrame and it is used to record the id, name, and value of the parameter if it is constant. The rules, kinetic and stoich\_table attributes store intermediates (initial assignments and rate laws) and equations (rate rules), intermediates (rate laws) and stoichiometry for use in creating differential equations using law of mass action respectively.

There are two main methods incorporated into the Bionetwork\_Model class which are change\_var\_type and write\_model. The first method will switch a given variable ID from a variable (i.e. it deletes it from the variable attribute) and creates a MV, CV, FV or SV class object. The change\_var\_type allows the user to specify the variable ID, variable type and any key word argument (kwarg) that is recognized by APMonitor as belonging to that class of variable. These kwargs are stored as attributes of the class. The second major method is to write the model. This is comprised of several sub methods that are concerned with writing the data stored from the SBML model in Pandas DataFrames into strings that are written into a new Python DataFrame. This method allows the user to specify all APMonitor options that are included with the latest release of the Gekko package.

## Writing the Model

As mentioned above, the write\_model method allows the user to specify all the options of a Gekko/APMonitor model. From here on out Gekko can be considered interchangeable with APMonitor Several algorithms had to be implemented to loop over the DataFrames where the information was stored and to join them to strings that would express the values and equations in the appropriate way to construct a valid Gekko model. This was accomplished by breaking the write\_model method and subdividing it into seven sub-methods.

The seven sub-methods consisted of initializing the Gekko model with a time series, writing variables, writing parameters, writing intermediates, writing equations, writing global options and writing the solve command. These different sub-methods were then called or directly implemented in the write\_model method. Some of the elements of these are worth noting.

The first one of special interest is the parameters function. In this case it is interesting because of two features of SBML models: i.) parameters used in equations can be defined outside of the list of parameters in the SBML model, and ii.) parameters can be a constant, or they can be assigned a value at the beginning of calculation. In the case of i.) these parameters are tricky to deal with because if they are defined inside of a reaction definition and so they are local parameters, and could share an overlapping namespace. These local parameters had to be renamed in the format <reaction\_id>\_<local\_parameter>. The second case meant that they were not initiated as Gekko model parameters at all, but rather as intermediates.

The second feature of special interest actually applies to any equation that was written as the math XML formatting is different than the formatting for math in Python and even more different than formatting for Gekko models. In this case only exponents were dealt with, in the manner as can be seen with the github repository turned in with this report. The other challenging thing about writing equations is that they came from many different sources. Intermediates were defined from rate laws given in reactions and from initial assignment rules given in rules, and differential equations were made from stoichiometry information and also from rate rules given in rules.

## Simulating the Model

Once the tool was used to create a .py file from the SBML file, Spyder IDE was used to run the simulations and matplotlib was used to plot the results. There is also a command to be able to view and plot any of the results but it appears to have a bug as reported on the Gekko github issues page.

# Results

The models constructed, along with example scripts to generate those models can be found in the github repository turned in along with this report. The attempt to validate aim i.) by simulating the genetic toggle switch is shown in Fig. 1. As can be seen while some elements of the model were implemented, a successful simulation was not obtained. This could possibly be due to a failure to correctly calculate intermediates as evidenced by Fig. 2. Since the differential equations are constructed by multiplying the rate of reactions (intermediates) by their stoichiometry and all the intermediates are 0 (based off of debugging steps this is the case) then the concentrations will never change. Indeed we see that this is the case from Fig. 1.

Because of this result no attempt has been made to validate a model-predictive control application of an SBML model. This is because the application needs to have event-based mechanics in order to simulate the perturbances that disrupt the controlling mechanism from its setpoint in order for it to respond. Given that intermediates weren’t working in a simpler problem no attempt was made to move on to a more complex problem.

A screenshot of a cell phone

Description automatically generated

Fig. 2. Simulation of Gardener’s Genetic Toggle Switch. This graph shows the concentration of the promoters and complexes with the promoters. They do not change at all over time which is not expected and could be why there is no change in the species concentrations from Fig. 1.

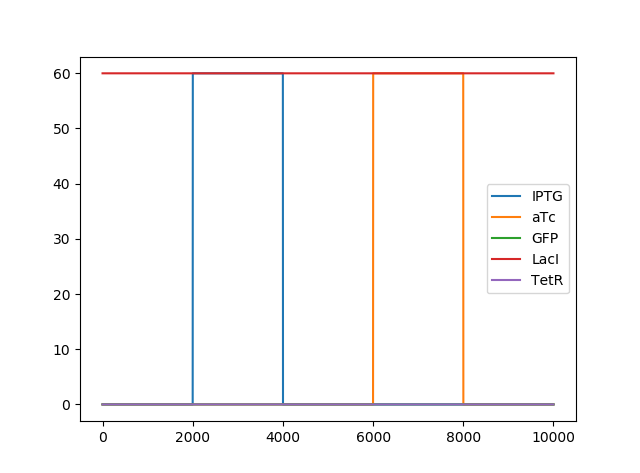


Fig. 1. Simulation of Gardener’s Genetic Toggle Switch. This was an attempt at getting the events to work correctly, which succeeded only in part. The concentration of aTc and IPTG show the correct response but there is no change in any of the other concentrations.

# Conclusion

The aims of this project were to: i.) create a library of functions that constructs a Gekko model in a .py format from an SBML file and ii.) transfer these functions into a downloadable module on github that allows the user to set additional options unique to Gekko/APMonitor. Aim i was tested by simulation of Gardener’s 2000 genetic toggle switch model [3] and aim ii was to be tested by setting a biomodels to a model-predictive control application. Neither of these aims were demonstrably shown to be accomplished.

What has been accomplished is laying the groundwork for a conversion tool between SBML and APMonitor. The encouraging result is that the model compiled by the conversion tool did indeed run without encountering an error and because it is in an object-oriented programming format it can be applied to any other model. The scripts produced by the conversion tool can set all the options that are available to APMonitor models. The missing piece is debugging what is preventing the intermediates from moving away from 0. The formulas match those generated when I double-check them by hand so it could possibly be a feature of using a sequential solver or some other aspect of the APMonitor model. This could potentially be resolved with collaboration with the developers of these tools.

Though the usefulness of APMonitor to synthetic biology was not demonstrated here, there are reasons to remain optimistic that it can help simplify model analysis and be a useful tool to analyze the dynamics of the system. Though it is most suited for larger-scale, i.e. non-stochastic modelling, its data-reconciliation mode may be useful in determining parameters of how different promoters and activators behave.

References

1. R. Kwok, *"Five Hard Truths for Synthetic Biology"*, Nature, vol. 463, no. 7279, pp. 288-290, 2010.
2. M. Fischbach, J. Bluestone and W. Lim, *"Cell-Based Therapeutics: The Next Pillar of Medicine"*, Science Translational Medicine, vol. 5, no. 179, pp. 179, 2013.
3. Gardener
4. Bornstein, B. J., Keating, S. M., Jouraku, A., and Hucka M. (2008) LibSBML: An API Library for SBML.*Bioinformatics*, 24(6):880–881, doi:10.1093/bioinformatics/btn051
5. Hucka, M.; Finney, A.; Bornstein, B. J.; Keating, S. M.; Shapiro, B. E.; Matthews, J.; Kovitz, B. L.; Schilstra, M. J.; Funahashi, A.; Doyle, S. M.; Kitano, M. J. (2004). "Evolving a lingua franca and associated software infrastructure for computational systems biology: The Systems Biology Markup Language (SBML) project". Systems Biology. 1 (1): 41–53. doi:10.1049/sb:20045008
6. Hedengren, J. D. and Asgharzadeh Shishavan, R., Powell, K.M., and Edgar, T.F., Nonlinear Modeling, Estimation and Predictive Control in APMonitor, Computers and Chemical Engineering, Volume 70, pg. 133–148, 2014, DOI: 10.1016/j.compchemeng.2014.04.013
7. Beal, L.D.R., Hill, D., Martin, R.A., and Hedengren, J. D., GEKKO Optimization Suite, Processes, Volume 6, Number 8, 2018, doi: 10.3390/pr6080106.
8. Restif O, Grenfell BT. Vaccination and the dynamics of immune evasion.

J R Soc Interface 2007 Feb; 4(12): 143-153

1. Fisher WG, Yang PC, Medikonduri RK, Jafri MS.

NFAT and NFkappaB activation in T lymphocytes: a model of differential activation of gene expression. Ann Biomed Eng 2006 Nov; 34(11): 1712-1728.

1. iBioSim 3: A Tool for Model-Based Genetic Circuit Design. Leandro Watanabe, Tramy Nguyen, Michael Zhang, Zach Zundel, Zhen Zhang, Curtis Madsen, Nicholas Roehner, and Chris Myers. ACS Synthetic Biology Article ASAP DOI: 10.1021/acssynbio.8b00078
2. J. O. Williams, “Narrow-band analyzer,” Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993.

**This template has been modified from the IEEE Template for**

**Transactions specifically for TMI by the TMI Editorial Office, February 2016.**

1. Submitted Dec. 14, 2018. The submitter is not funded for this research.

   M. W. Bradley is a PhD candidate at University of Utah, UT 84112 USA (e-mail: Bradley\_m@ sci.utah.edu).

   . [↑](#footnote-ref-1)