BioSystems 169–170 (2018) 20–25

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Short Communication

A portable structural analysis library for reaction networks

Yosef Bedaso[a], Frank T. Bergmann[b], Kiri Choi[a], Kyle Medley[a], Herbert M. Sauro[a,*]

[a] Department of Bioengineering, William H. Foege Building, Box 355061, Seattle, WA 98195-5061, USA
[b] BioQuant/COS, Heidelberg University, Heidelberg, Germany

ARTICLE INFO

Keywords:
Simulation
Structural analysis
Software
Systems biology

ABSTRACT

The topology of a reaction network can have a significant influence on the network's dynamical properties. Such influences can include constraints on network flows and concentration changes or more insidiously result in the emergence of feedback loops. These effects are due entirely to mass constraints imposed by the network configuration and are important considerations before any dynamical analysis is made. Most established simulation software tools usually carry out some kind of structural analysis of a network before any attempt is made at dynamic simulation. In this paper, we describe a portable software library, `libStructural`, that can carry out a variety of popular structural analyses that includes conservation analysis, flux dependency analysis and enumerating elementary modes. The library employs robust algorithms that allow it to be used on large networks with more than a two thousand nodes. The library accepts either a raw or fully labeled stoichiometry matrix or models written in SBML format. The software is written in standard C/C++ and comes with extensive on-line documentation and a test suite. The software is available for Windows, Mac OS X, and can be compiled easily on any Linux operating system. A language binding for Python is also available through the pip package manager making it simple to install on any standard Python distribution. The bulk of the source code is licensed under the open source BSD license with other parts using as either the MIT license or more simply public domain. All source is available on GitHub (https://github.com/sys-bio/Libstructural).

1. Background

One of the most fundamental processes in living organisms is the chemical reaction where molecules combine, decompose, change configuration or exchange molecular subunits. A network of chemical reactions will obey mass-conservation resulting in properties of the network that are independent of the underlying reaction kinetics. In this paper, we describe a new portable software library that provides many facilities for analyzing the topological properties of reaction networks as a result of mass-conservation.

When describing multiple reactions in a network, it is convenient to represent the stoichiometric coefficients in a compact form called the **stoichiometry matrix**, **N** (Reder, 1988). This matrix is a *m* row by *n* column matrix where *m* is the number of species and *n* the number of reactions. The columns of the stoichiometry matrix correspond to the distinct chemical reactions in the network, the rows to the molecular species, one row per species. The intersection of a row and column in the matrix indicates the stoichiometric coefficient.

The stoichiometry matrix represents the connectivity of the network and contains important information on the network's structural characteristics. These characteristics fall into two groups, relationships among the species as indicated by dependencies in the rows of the stoichiometry matrix and relationships among the reaction rates due to dependencies among the columns (Sauro and Ingalls, 2004). In this paper, we will describe a software library called `libStructural` that provides a wide variety of functions to analyze both row and column dependences in a stoichiometry matrix. libStructural is not concerned with constraint-based modeling (Bordbar et al., 2014) or metabolic flux analysis (Morales et al., 2016). Instead it focuses on the structure of the stoichiometry matrix.

1.1. Moiety conservation laws

One of the characteristics of biological network models is the conservation of certain molecular subgroups, termed moieties (Reich and Selkov, 1981). A typical example of a conserved group in a model is the conservation of the adenine nucleotide moiety, i.e. the total amount of ATP, ADP, and AMP is constant during the evolution of the model.

Determining the conservation laws is important for several reasons. One practical advantage is that the system equations in the form of ordinary or stochastic differential equations can be reduced in size thus making numerical analysis more efficient. This fact is exploited in many

* Corresponding author.
E-mail addresses: bedasoy@uw.edu (Y. Bedaso), frank.bergmann@bioquant.uni-heidelberg.de (F.T. Bergmann), kirichoi@uw.edu (K. Choi), hsauro@u.washington.edu (H.M. Sauro).

https://doi.org/10.1016/j.biosystems.2018.05.008
Received 8 January 2018; Received in revised form 30 April 2018; Accepted 28 May 2018
Available online 30 May 2018

modern modeling platforms including but not limited to SBW (Sauro et al., 2003), Copasi (Hoops et al., 2006), PySCeS (Olivier et al., 2005), VCell (Loew and Schaff, 2001), JWS Online (Peters et al., 2017), libRoadRunner (Somogyi et al., 2015), and the SB Toolbox (Schmidt and Jirstrand, 2006). A recent review of software provision in this area can be found in Dandekar et al. (2012). In addition to reducing the size of the model, reduction of the number of differential equations means that the model's Jacobian matrix is non-singular (Sauro and Ingalls, 2004), an important requirement for a number of numerical methods including steady-state analysis and bifurcation analysis. The conservation laws are also important for theoretical reasons because a non-singular Jacobian is required for metabolic control analysis, stability analysis and frequency analysis (Ingalls, 2004). Finally, conservation laws have very practical implications for perturbation studies and targeted gene knockouts (Eisenthal and Cornish-Bowden, 1998). In such circumstances, the conservation laws provide hard limits to how species levels can change, at least over the time scale of the conservation law. The conservation relationships can also lead to implicit regulatory effects in a network as exemplified by the work of Markevich et al. (2004).

The `libStructural` library supports the computation of **L**, $\mathbf{L_0}$, and **Γ** matrices (Reich and Selkov, 1981). In addition, the library will reorder the stoichiometry matrix rows including row labels as appropriate.

1.2. Steady-state flux constraints

Whereas the rows of the stoichiometry matrix indicate dependencies among the species, the columns of the stoichiometry matrix indicate dependencies among the reaction rates (van der Heijden et al., 1994). The `libStructural` library supports the computation of **K**, $\mathbf{N_{DC}}$, and $\mathbf{N_{IC}}$ matrices (Reich and Selkov, 1981). In addition, the library will reorder the stoichiometry matrix columns including column labels as appropriate. Fig. 1 summarizes the partitioning of the matrix into the various subdivisions.

1.3. Elementary modes

Elementary modes (Zanghellini et al., 2013) are the simplest pathways within a metabolic network that can sustain a steady-state and at the same time are thermodynamically feasible (Ataman and Hatzimanikatis, 2015). Depending on the size of the metabolic network, the number of elementary modes can range from no modes to billions of modes. The full set of elementary modes represents the complete metabolic potential of a given metabolic network and as a result is of interest to the metabolic analysis and engineering communities.

The `libStructural` library computes elementary modes via a refactored Metatool component (Kamp and Schuster, 2006) and includes both integer and double variants as well as a refactored gEFM library (Ullah et al., 2016).

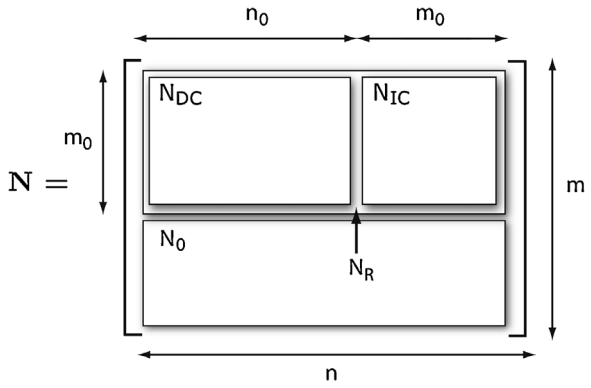

Fig. 1. Partitioned reordered stoichiometry matrix: n = number of reactions; m = number of species; $\mathbf{N_{DC}}$ = partition of linearly dependent columns; $\mathbf{N_{IC}}$ = partition of linearly independent columns; $\mathbf{N_R}$ = reduced stoichiometry matrix; $\mathbf{N_0}$ partition of linearly dependent rows.

2. Results

2.1. Software implementation

The core library for `libStructural` was originally developed by Frank Bergmann. With the development of the C/C++ version of libRoadRunner, `libStructural` was subsequently integrated into libRoadRunner (Somogyi et al., 2015). In this paper, we describe the separate and reusable `libStructural` library.

The core of `libStructural` is written in ISO C/C++ to achieve maximum portability and interoperability. The software can be used on Windows, Mac OS X, and Linux operating systems. Network models can be supplied either directly as a raw stoichiometry matrix or indirectly as an SBML model (Hucka et al., 2003). For SBML support we use the libSBML library (Bornstein et al., 2008). In order to maintain information on the row and column reorderings during the calculations, all row and columns of matrices can be labeled. The library relies heavily on LAPACK (http://www.netlib.org/lapack/), a standard library for linear algebra that is used to carry out householder reflections for the QR factorization (Golub and Van Loan, 1996).

The library itself is split into two parts. One part is used to wrap and expose certain LAPACK functions and to implement other commonly used linear algebra results not directly supported by LAPACK. These include methods that can compute orthonormal null space vectors or generate the reduced echelon forms using Gauss–Jordan matrix reduction. The second part implements the stoichiometric network specific methods. These include conservation analysis such as computing the link and gamma matrices (conservation matrix), returning the reordered row stoichiometry matrix and total amounts in each conservation law. In addition, the columns of the stoichiometric matrix are also analyzed to generate the independent and dependent fluxes, including the **K** matrix. Documentation is provided through readthedocs (https://libstructural.readthedocs.io). The source code is licensed under a combination of the modified BSD license, MIT (gEFM), and public domain (`libMetatool`).

For computing elementary modes there are a wide variety of published software tools. Rather than write our own we decided to reuse existing software. We examined a wide variety of existing tools, and we found only two, Metatool (Kamp and Schuster, 2006) and gEFM (Ullah et al., 2016) that could be easily reused within our C/C++/Python framework.

Metatool 4.3 is written in standard C and has a liberal open license. It is therefore very easy to implement across different computing platforms. We refactored the code to convert Metatool 4.3 into a reusable library we call `libMetatool`. This allows Metatool to be linked to any programming language. The refactoring was done such that `libMetatool` can be used independently of libStructural. The one major change to Metatool when refactoring was to use 64-bit integer types for all calculations. This was done to improve the numerical stability of the algorithms used by the integer version Metatool when dealing with large models. Since the original Metatool source code is in the public domain, the refactored Metatool source code is similarly unrestricted. In order to deal with models with fractional stoichiometries, we also distribute the double version of Metatool and a refactored gEFM.

We also incorporated gEFM as a library which is a more modern distribution that uses a completely different algorithm to Metatool. The code was refactored by adding an API (Application Programming Interface) and improved error handling and memory protection. The addition of gEFM allows comparisons to be made between Metatool and gEFM. For gEFM, we also provide a copy results to file method that can be used for analyzing large models. This was to avoid returning very large arrays directly to the Python console from gEFM. gEFM can be