Journal of
Computer Science & Systems Biology

Idowu and Bown, J Comput Sci Syst Biol 2011, 4:4
DOI: 10.4172/jcsb.1000077

Research Article Open Access

Towards an Exact Reconstruction of a Time-Invariant Model from Time Series Data

Michael A. Idowu[1]* and James Bown[2]

[1]*Scottish Informatics Mathematics Biology and Statistics (SIMBIOS) Centre, School of Contemporary Sciences, University of Abertay, Dundee, Scotland, United Kingdom*
[2]*Institute of Arts, Media and Computer Games, University of Abertay Dundee, Scotland, United Kingdom*

Abstract

Dynamic processes in biological systems may be profiled by measuring system properties over time. One way of representing such time series data is through weighted interaction networks, where the nodes in the network represent the measurables and the weighted edges represent interactions between any pair of nodes. Construction of these network models from time series data may involve seeking a robust data-consistent and time-invariant model to approximate and describe system dynamics. Many problems in mathematics, systems biology and physics can be recast into this form and may require finding the most consistent solution to a set of first order differential equations. This is especially challenging in cases where the number of data points is less than or equal to the number of measurables. We present a novel computational method for network reconstruction with limited time series data. To test our method, we use artificial time series data generated from known network models. We then attempt to reconstruct the original network from the time series data alone. We find good agreement between the original and predicted networks.

Keywords: Mathematical modelling; Dynamic systems; Reverse engineering; Inverse problems; System identification and parameter estimation; Time series analysis and Time-invariant models; Network inference algorithm

Introduction

Interaction networks may be used to describe a wide range of biological phenomena, spanning genetic [1] and biochemical networks [2], analysis of clinical orbiomolecular data [3], and personalised medicine [4]. These network-based approaches seek to describe dynamic systems as interaction networks, where the nodes in the network represent the measurables and the edges represent interactions between any pair of nodes. Additionally, networks may have weighted edges where the weights relate to the strength of the interactions [1]. For example, gene networks are commonly represented by directed graphs where the nodes of the graph are genes and the directed edges are causal relationships between genes [1]. In metabolic pathway modelling, the first step involves the development of appropriate functions that describe the behaviour of all the constituents of the system and identification of all components of interest with their symbolic names with (directional) arrows showing which components modulates the ows into, between, and out of components [2]. In such cases, the set of nodes is defined by the measurables; importantly, the edges and their weights, and so the network topology, must be inferred from the data.

Many systems, within and beyond biology, may be described by interaction networks. For example, within biology gene regulatory networks and intra-cellular signalling networks are essentially interaction networks: nodes represent genes and bio-molecular species respectively; edges represent interactions, which may change under different perturbations. Such biological networks are characterised by complex interactions and feedback loops [5]. Beyond biology, our socio-technical infrastructures are readily described by interaction networks, both in terms of their individual characteristics and their inter-network interactions [6]. Our energy, transport, water and telecommunications networks are a complex adaptive system [7] akin to biological interaction networks. These complexities, in biological and engineered systems, make system dynamics challenging to interpret. As a result, system-scale methods that support network identification and parameter estimation methods are important components in developing our understanding of these systems [8]. Simple, effective, reasonable and robust algorithms for constructing underlying network topologies are in demand for understanding the dynamics of complex systems and processes.

Time-evolution of complex processes may be represented in the form of time series data. These time series data are state vectors measured over time in linear order and often at regular intervals. Describing the dynamic properties of the observed time series of a system can help identify and model patterns, predict behaviour, and approximate the underlying process generating the data [3]. Systems that generate time series data may be highly complex and incapable of exact description, so approximating the underlying process through parametric models can often help capture key features of data relevant for the purpose of interest [9].

Here, we propose and present a novel computational, robust method (solution) for constructing weighted interaction networks based on time series data.

The construction of an interaction network from time series data may be recast as seeking to identify a mathematical model that relates a given time point to its successor, consistent with every time step and for all measurables. This model may be expressed in the form of a n by m matrix, for n time points and m measurables. For a given time point t, the mathematical model must relate a measured value x_i at t, x_i (t), to its value in the subsequent time point x_i (t+1). Moreover, this relation is required for all x(1…m) and for all t(0… (n - 1)). This mathematical model, referred to here as a transformation matrix, must be calculated

***Corresponding author:** Michael A. Idowu, Scottish Informatics Mathematics Biology and Statistics (SIMBIOS) Centre, School of Contemporary Sciences, University of Abertay, Dundee, Scotland, United Kingdom DD1 1HG, E-mail: m.idowu@abertay.ac.uk

Received September 08, 2011; **Accepted** November 10, 2011; **Published** November 23, 2011

Citation: Idowu MA, Bown J (2011) Towards an Exact Reconstruction of a Time-Invariant Model from Time Series Data. J Comput Sci Syst Biol 4: 055-070. doi:10.4172/jcsb.1000077

from the data, i.e. the model and model parameters must be inferred from data. This process may be described as an inverse problem, where we must identify the transformation matrix solution to the system describing the time series data.

Finding a time-invariant matrix to the time series inverse problem may require that the number of time points provided be at least equal to the number of measured variables +1. Whenever the available data set is limited in size, the number of time points is less than or equal to the number of measured variables, solution identification becomes more challenging.

The method proposed here requires a minimum of 3 time points to be able to reconstruct a network. As we show later, as the number of time points increases the network reconstruction process becomes more accurate. However, even with (only) 3 time points, we show that the method produces a data-consistent model of a given data set. A model is data-consistent if it is capable of reproducing exactly the original time series data that was used to infer the model.

Any time series data set can be described as a system of ODEs, whose variables are the measurables in the physical system. For example, a gene regulatory network with multiple genes can be represented by a set of nonlinear ordinary differential equations (ODE) with the expression level of each gene as a variable [10]. Reverse engineering of such networks, through gene expression data analysis and reconstruction of gene regulatory networks, involves revealing the underlying network of gene-gene interactions from the measured dataset of gene expression. This process can involve some form of mapping of the observed data to a reconstructed and representative network model inferred from data using reverse engineering techniques [3]. With respect to the identification of the transformation matrix, we show below how variables in the ODEs model map to matrix elements and our algorithm operates on this matrix to reverse engineer interaction network topology and edge weightings. Solving this inverse problem has implications for data modelling in systems biology generally and in particular in personalised medicine through for example protein interaction network modelling.

Our algorithm searches for a solution to the inverse problem, and the algorithm identifies the same solution every time for a given problem. However, if the size of the available dataset is small, i.e., total number of time points < number of measured variables, other potential solutions may still exist. Our approach is different from other forms of optimisation algorithms such as genetic algorithms that are based on finding the best set of model parameters within a search domain, because starting points for the domain search process are decided prior to parameter estimation. Here, no single parameter is fixed prior to estimation and no start point comes into play; the solution is purely based on experimental data. Though some parameters can be set to fixed values, our parameter estimation technique is purely based on and driven by experimental data, without any need for fixed parameters.

We outline how dynamic systems may be described by systems of ordinary differential equations, and present a matrix-based approximation to the solution. We then describe our algorithm to solve the inverse problem through identification of the values of elements in this matrix. We demonstrate the capability of the algorithm to reverse engineer interaction networks with a worked example, provide summary statistics on algorithm performance and comment on the uniqueness of the solutions discovered.

Ordinary Differential Equation Systems

It is common to use ODE formalisms to model and analyse data in complex dynamical systems [10]. Sometimes, simple linear ODE models are sufficient representations of the system in order to identify essential relationships among network components based on time series data analysis. Solutions to systems of first-order ODE may involve matrix factorisation to approximate a matrix exponential [11].

Specifically, the solution of the homogenous equation

$$\frac{dx}{dt} = J.X(t) \qquad (2.1)$$

where X is a column vector representing the dependent components of the system, t is a time variable and J, the relative rate of change of X(t), is a matrix often referred to as jacobian, is given by

$$X(t) = e^{j.t}X_0 \qquad (2.2)$$

Where $X(0) = X_0$ the initial state of the system for a system of linear differential equations, involving a transforming matrix J and the solution $[e^{j*t} * X]$ a function of (J, X, t), where X is a column vector representing the dependent components of the system, and t is a time variable that represents a regular time interval between any two successive states, if J (the transformation matrix inferred from time series data) is derived such that it remains unchanged throughout the system (from the initial condition to steady-state, and after), then the model $X = J*X$ may be said to be time-invariant because J is not dependent on time. In other words, the parameters of J do not change and are not functions of time.

Problem statement

Let X_t and X_{t+1} be the state vectors known from a given time series data, and assume that the time interval t_c is regular and of small magnitude. Then $\dot{X} \approx \frac{X_t+1-X_t}{t_c}$, and the inverse problem is mathematically equivalent to

$$\frac{X_t+1-X_t}{t_c} \approx \dot{X} = J.X \qquad (2.3)$$

There may be more that one J matrix that satisfies the above equation, so the primary objective is to find the best J that describes - with least error - the transformation from any state X_t to the next state X_{t+1}.

Relative rate of change

J, the relative rate of change in $X_{(t)}$, may be described as $J = \frac{\dot{X}}{X}$ (2.4) that is, J is the absolute rate of change X in relation to the present state value X. Since it is true that $\int \frac{\dot{X}}{X} dt = \int \frac{1}{X} dX = InX + c_1$ therefore one may describe J in terms of $\int jdt = j.t + c_2 = InX + c_1 \rightarrow J = \frac{d(J.t)}{dt} = \frac{d(InX)}{dt}$ which means that J itself is that relative rate of change, and it is equivalent to the rate of change in the logarithm of X(t) [12].

The jacobian, in a sense, can be thought as a representation of the rate and extent of change over time for any component and its temporal influence on other components. The jacobian matrix may thus be regarded as a construct for describing system dynamics. The mathematical significance of determined eigenvalues, or chacteristic values, of the jacobian can help inform understanding of the behavior of systems near a stationary point. By observing the eigenvalues of the jacobian matrix in a given neighbourhood, an indication of the stability of the system can be obtained. If the eigenvalues all have a negative real part, the system is said to be stable [13]. A positive real part in any of the eigenvalues means the system may be unstable at that point, exhibiting large transient peaks [13]. Regarding the determinant of the jacobian matrix, the absolute value of the determinant of the (jacobian) transformation matrix indicates the overall rate of change of all the measured variables [14]. A necessary and sufficient condition for the jacobian matrix invertibility is that the magnitude of its determinant > 0 [14].

The algebraic representation of an ODE-based model solution for a time series problem is simple and straightforward.

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \vdots \\ \dot{X}_n \end{bmatrix} = \begin{bmatrix} \frac{\partial X_1}{\partial X_1} X_1 + & \frac{\partial X_1}{\partial X_2} X_2 + & \cdots + & \frac{\partial X_1}{\partial X_m} X_m \\ \frac{\partial X_2}{\partial X_1} X_1 + & \frac{\partial X_2}{\partial X_1} X_2 + & \cdots + & \frac{\partial X_2}{\partial X_m} X_m \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial X_m}{\partial X_1} X + & \frac{\partial X_m}{\partial X_1} X_2 + & \cdots + & \frac{\partial X_m}{\partial X_m} X_m \end{bmatrix} \qquad (2.5)$$

And this is related to the eigenvectors and eigenvalues representation as follows

$$\begin{aligned} X_1(t) &= e^{\lambda_1 t}.[\upsilon_{1,1}].[p_1] + e^{\lambda_2 t}.[\upsilon_{1,2}].[p_2] + \cdots + e^{\lambda_n t}.[\upsilon_{1,m}].[p_m] \\ X_2(t) &= e^{\lambda_1 t}.[\upsilon_{2,1}].[p_1] + e^{\lambda_2 t}.[\upsilon_{2,2}].[p_2] + \cdots + e^{\lambda_n t}.[\upsilon_{2,m}].[p_m] \\ &\quad \cdots \qquad \cdots \qquad \cdots \\ X_n(t) &= e^{\lambda_1 t}.[\upsilon_{m,1}].[p_1] + e^{\lambda_2 t}.[\upsilon_{m,2}].[p_2] + \cdots + e^{\lambda_m t}.[\upsilon_{m,m}].[p_m] \end{aligned}$$

where the parameters λ_i and υ_i represent the eigenvalues and eigenvectors, respectively; and the initial condition, $[X_1(0), X_2(0), \ldots X_n(0)]$ is favourably chosen, i.e., decomposed and approximated, as the linear combination of the eigenvectors of the jacobian matrix using the parameter set $[p_1, p_2, \ldots ..., p_m]$. Each measurable X_i (component within a network) is a function based on the initial condition. The initial condition (first measurement) or any state vectors at any timepoint may be rewritten (decomposed) in a form such that the parameter set $[p_1, p_2, \ldots .., p_m]$ becomes fixed [12].

Complex nonlinearity in systems of nonlinear ODEs

Since most complex systems are nonlinear in nature, nonlinear models are required to describe them fully. Moreover, some systems may require that second-order ODEs be used to formulate a sufficient description of their dynamics due to the complex nature of the processes involved. However, in such cases, the inference of model parameters becomes a much more difficult task. In particular, the appropriate model structure needs to be identified, or at least estimated with a reasonable degree of confidence, before model parameters are estimated.

Here, we suggest that first-order linear ODE solutions based soley on time series data, i.e., with no assumptions made about network structure, may be sufficient to derive valuable information about the most important links among variables if good search algorithms for network inference are employed.

In some cases modellers predetermine network structure prior to parameter estimation, e.g. [15]. By keeping model structure fixed in this way, the inference procedure is restricted and identification of other solutions that do not adhere to the fixed structure constraint is not achieved. Such techniques themselves are thus limited since they are only successful when the constraint imposed on the system is met. Of course, the rationale for fixing the network topology is that it eases identification of potential solutions for underdetermined systems [15]. As a result, many modellers tend to assign some fixed values to subsets of model parameters to ease the fitting process. A good algorithm is one which can accommodate fixed parameters and at the same time does not require these to find an accurate solution.

We propose such an algorithm, effected via a combination of pre-conditioning, regression, analytical techniques, half and or S-system approaches based on time series data. Our rationale for proposing linear ODE solutions to solving inverse problems in time series are as follows:

(1) Second-order differential equations can be recast into first-order (ordinary) differential equations;

(2) Any nonlinear ODE can be cast or approximated into a power-law form called (S-systems or half systems);

(3) Nonlinear half systems are equivalent to systems of log-linear differential equations;

(4) Log-linear differential equations may be used using the same techniques for solving systems of linear differential equations.

Therefore, the development of a robust method for solving systems of linear differential equations is relevant to non-linear problems. Multiple regression and reverse engineering techniques, including the logarithmic inverse function, are important steps we have considered. Here, however, we limit the focus of the paper to time series analysis under systems of linear ordinary differential equations.

Methodology

In a system of linear differential equations an inverse problem may be written as in (2.4): $\dot{X} = J * X$

where $\dot{X}$ and X are known vectors of same length, and J is the unknown matrix that must be identified. Note that there is difference between a general system of n linear differential equations with unknown (jacobian) matrix parameters and a general system of n linear equations with unknown vector parameters. The latter is much simpler to solve due to the reduction in the number of unknown parameters. However, the formulation of the inverse problem remains the same in structure.

A general system of m linear equations with m unknown parameters is of the form b = A.x (3.1)

with the following matrix equations:

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1m}x_m \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2m}x_m \\ & \vdots & & \\ a_{m1}x_1 + & a_{m2}x_2 + & \cdots + & a_{mm}x_m \end{bmatrix} \qquad (3.2\ )$$

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ & \vdots & & \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \qquad (3.3)$$

where b=[b_1,b_2,…,b_m]^T are the constant terms, a_{11},a_{12},…,a_{mm} are the coefficients of the system, and x=[x_1,x_2,….,x_m]^T are the unknown parameters; note [υ]^T denotes the transposed vector [υ]. Finding a solution to the system above involves searching for values in the x vector where all the m equations are simultaneously satisfied and valid.

A general system of m linear differential equations with an unknown m x m jacobian matrix J is of the form $\dot{X} = J.X$ and has the following matrix equation:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \vdots \\ \dot{X}_m \end{bmatrix} = \begin{bmatrix} \frac{\partial X_1}{\partial X_1}X_1 + & \frac{\partial X_1}{\partial X_2}X_2 + & \cdots + & \frac{\partial X_1}{\partial X_m}X_m \\ \frac{\partial X_2}{\partial X_1}X_1 + & \frac{\partial X_2}{\partial X_2}X_2 + & \cdots + & \frac{\partial X_2}{\partial X_m}X_m \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial X_m}{\partial X_1}X_1 + & \frac{\partial X_m}{\partial X_2}X_2 + & \cdots + & \frac{\partial X_m}{\partial X_m}X_m \end{bmatrix} = \begin{bmatrix} \frac{\partial X_1}{\partial X_1} & \frac{\partial X_1}{\partial X_2} & \cdots & \frac{\partial X_1}{\partial X_m} \\ \frac{\partial X_2}{\partial X_1} & \frac{\partial X_2}{\partial X_2} & \cdots & \frac{\partial X_2}{\partial X_m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial X_m}{\partial X_1} & \frac{\partial X_m}{\partial X_2} & \cdots & \frac{\partial X_m}{\partial X_m} \end{bmatrix} . \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \qquad (3.4)$$

where $\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} = X_{(t)}$ is a known state vector recognized as the tth vector of *X* (not *X* at time t), *X* is the derivative vector which may be calculated from two known state vectors (X_t and X_{t+1}) as $\dot{X} \approx \frac{X_{t+1} - X_t}{t_c}$ where t_c is the interval of separation, and *J* is the unknown m x m jacobian matrix. Therefore, at least two state vectors of the same length are required to define an inverse problem in systems of linear differential equations. The following multi-state representation defines an inverse problem involving n+1 different states:

$$\begin{bmatrix} \dot{X}_{1_0} & \dot{X}_{1_1} & \cdots & \dot{X}_{1_{n-1}} \\ \dot{X}_{2_0} & \dot{X}_{2_1} & \cdots & \dot{X}_{2_{n-1}} \\ & & \vdots & \\ \dot{X}_{m_0} & \dot{X}_{m_1} & \cdots & \dot{X}_{m_{n-1}} \end{bmatrix} = \begin{bmatrix} \frac{\partial X_1}{\partial X_1} & \frac{\partial X_1}{\partial X_2} & \cdots & \frac{\partial X_1}{\partial X_m} \\ \frac{\partial X_2}{\partial X_1} & \frac{\partial X_2}{\partial X_2} & \cdots & \frac{\partial X_2}{\partial X_m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial X_m}{\partial X_1} & \frac{\partial X_m}{\partial X_2} & \cdots & \frac{\partial X_m}{\partial X_m} \end{bmatrix} . \begin{bmatrix} X_{1_0} & X_{1_1} & \cdots & X_{1_{n-1}} \\ X_{2_0} & X_{2_1} & \cdots & X_{2_{n-1}} \\ & & \vdots & \\ X_{m_0} & X_{m_1} & \cdots & X_{m_{n-1}} \end{bmatrix} \qquad (3.5)$$

which is equivalent to

$$\begin{bmatrix} \frac{X_{1_1}-X_{1_0}}{t_c} & \frac{X_{1_2}-X_{1_1}}{t_c} & \cdots & \frac{X_{1_n}-X_{1_{n-1}}}{t_c} \\ \frac{X_{2_1}-X_{2_0}}{t_c} & \frac{X_{2_2}-X_{2_1}}{t_c} & \cdots & \frac{X_{2_n}-X_{2_{n-1}}}{t_c} \\ & & \vdots & \\ \frac{X_{m_1}-X_{m_0}}{t_c} & \frac{X_{m_2}-X_{m_1}}{t_c} & \cdots & \frac{X_{m_n}-X_{m_{n-1}}}{t_c} \end{bmatrix} = \begin{bmatrix} \frac{\partial X_1}{\partial X_1} & \frac{\partial X_1}{\partial X_2} & \cdots & \frac{\partial X_1}{\partial X_m} \\ \frac{\partial X_2}{\partial X_1} & \frac{\partial X_2}{\partial X_2} & \cdots & \frac{\partial X_2}{\partial X_m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial X_m}{\partial X_1} & \frac{\partial X_m}{\partial X_2} & \cdots & \frac{\partial X_m}{\partial X_m} \end{bmatrix} . \begin{bmatrix} X_{1_0} & X_{1_1} & \cdots & X_{1_{n-1}} \\ X_{2_0} & X_{2_1} & \cdots & X_{2_{n-1}} \\ & & \vdots & \\ X_{m_0} & X_{m_1} & \cdots & X_{m_{n-1}} \end{bmatrix} \quad (3.6)$$

assuming the states values are captured at regular time interval t_c. The smaller the value of t_c the better the outcome of this derivative approximation; this is because the linear approximation, $e^{tc} \approx 1+t_c$ of $e^{tc} = 1+t_c+\frac{t_c^{\,2}}{2!}+\frac{t_c^{\,3}}{3!}+....$ improves as tc gets smaller and closer to 0 [12]. The solution to this system of linear differential equations:

$$\frac{\begin{bmatrix} \left(X_{1_1}-X_{1_0}\right) & \left(X_{1_2}-X_{1_1}\right) & \cdots & \left(X_{1_n}-X_{1_{n-1}}\right) \\ \left(X_{2_1}-X_{2_0}\right) & \left(X_{2_2}-X_{2_1}\right) & \cdots & \left(X_{2_n}-X_{2_{n-1}}\right) \\ & \cdots & \vdots & \\ \left(X_{m_1}-X_{m_0}\right) & \left(X_{m_2}-X_{m_1}\right) & \cdots & \left(X_{m_n}-X_{m_{n-1}}\right) \end{bmatrix}}{t_c} = \frac{\begin{bmatrix} \frac{\partial X_1}{\partial X_1} & \frac{\partial X_1}{\partial X_2} & \cdots & \frac{\partial X_1}{\partial X_m} \\ \frac{\partial X_2}{\partial X_1} & \frac{\partial X_2}{\partial X_2} & \cdots & \frac{\partial X_2}{\partial X_m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial X_m}{\partial X_1} & \frac{\partial X_m}{\partial X_2} & \cdots & \frac{\partial X_m}{\partial X_m} \end{bmatrix} . \begin{bmatrix} X_{1_0} & X_{1_1} & \cdots & X_{1_{n-1}} \\ X_{2_0} & X_{2_1} & \cdots & X_{2_{n-1}} \\ & & \vdots & \\ X_{m_0} & X_{m_1} & \cdots & X_{m_{n-1}} \end{bmatrix}}{1} \quad (3.7)$$

$$\rightarrow \frac{X_{mn+1}-X_{mn}}{t_c} = J * X_{mn}$$

Is thus

$$\begin{bmatrix} X_{1_1} & X_{1_2} & \cdots & X_{1_n} \\ X_{2_1} & X_{2_2} & \cdots & X_{2_n} \\ & & \vdots & \\ X_{m_1} & X_{m_2} & \cdots & X_{m_n} \end{bmatrix} = [\exp^{\left(\begin{bmatrix} \frac{\partial X_1}{\partial X_1} & \frac{\partial X_1}{\partial X_2} & \cdots & \frac{\partial X_1}{\partial X_m} \\ \frac{\partial X_2}{\partial X_1} & \frac{\partial X_2}{\partial X_2} & \cdots & \frac{\partial X_2}{\partial X_m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial X_m}{\partial X_1} & \frac{\partial X_m}{\partial X_2} & \cdots & \frac{\partial X_m}{\partial X_m} \end{bmatrix} .t_c\right)}] * \begin{bmatrix} X_{1_0} & X_{1_1} & \cdots & X_{1_{n-1}} \\ X_{2_0} & X_{2_1} & \cdots & X_{2_{n-1}} \\ & & \vdots & \\ X_{m_0} & X_{m_1} & \cdots & X_{m_{n-1}} \end{bmatrix} \quad (3.8)$$

Consequently solving an inverse problem in a system of linear differential equations is equivalent to identifying $J, \frac{\partial(X_1,\ldots,X_m)}{\partial(x_1,\ldots,x_m)}$ that fits the data best. This requires optimal estimation of the parameters of *J*; calculating all the entries (parameters) of the matrix of all first-order partial derivatives of vector-valued functions of variable *i* with respect to another variable *j*, where X_i or X_j represents the variable function of component *i* or *j* respectively. Using simple variables, the solution may be rewritten as: $X_{(N+1)} = E.X_{(N)} = e^{J.t_c}.X_{(N)}$ which is on the one hand a representation of system of linear equations, $X_{(N+1)}$=*E.X*($_N$), and on the other hand a direct interpretation of a system of nonlinear equations, $X_{(N+1)} = e^{J.t_c}.X_{(N)}$, implying that $E = e^{J.t_c}$. Consequently E is equivalent to the matrix exponential (function) of the matrix product *J.t*$_c$. The time constant t_c is easily calculated as the difference in time between T_1 at any state in $X_{(N+1)}$ and T_0 its previous state in X_N. E can easily be approximated by our new regression techniques (described below), and these are variant forms of regression for ODE systems. Often regression analysis is used to understand the relation between two or more interrelated variables. In systems of linear differential equations, regression analysis can be used to infer causal relationships or transformations between the model variables and states. So in principle, the state transformation (or the transposive regression) matrix, E, is the matrix exponential of (*J.t*$_c$).

From these definitions, different approaches in analysis may be followed. One which is of worthy mention is the logarithm of E, which in this case is the actual result (*J.t*$_c$) being derived from E such that its exponential equals E.

Approximating a value of E may comprise using a regression technique having the steps:

1. Acquire time series data with the number of time points ≥3;
2. Preprocess the measured state values of one or more components of the system using matrix transposition;
3. Undertake regression analysis of the resultant data using a Moore-Penrose pseudoinverse technique;
4. Postprocess the resultant data using matrix retransposition;

5. Calculate the logarithmic inverse of the retransposed result through application of eigenvectors and eigenvalues techniques;

6. Scale down the resultant data by factor (magnitude) of the time interval used.

The implementation of the steps 2-6 is shown in Sections 3.1.1 and 3.1.2 below. Section 4 considers the provision of time series data (step 1).

Transposive and repressive regression methods

In solving a system of linear differential equations, we define the solution to an inverse problem as one conditioned on the property:

$$\exp^{(J*t_c)} * X_{(t)} = X_{(t+1)}$$

Or simply:

$$E * X_{(t)} = X_{(t+1)}$$

where $E \approx \exp^{(J*t_c)}$, the time interval t_c is assumed to be regular; J is unknown at this point and must be identified, and $X_{(t)}$ and $X_{(t+1)}$ are known arrays of column vectors, each a representation of system states at two successive time points, termed before and after. In this paper we present for the first time two new algorithms to derive J, namely:

1. $(\ddot{T})$ Transposive Regression Algorithm
2. $(\ddot{R})$ Repressive Regression Algorithm

Derivation of the $(\ddot{T})$ transposive regression algorithm:

Steps

(1) $E_1 * X_{(before)} = X_{(after)}$

(2) $X_{(before)^T} * E_1^T = X_{(after)^T}$

(3) $X_{(before)} * X_{(after)}^T * E_1^T = X_{(before)} * X_{(after)}^T$

(4) $E_1^T = [X_{(before)} * X_{(after)}^T]^{-1} * X_{(before)} * X_{(after)}^T$

(5) $E_1 = ([X_{(before)} * X_{(after)}^T]^{-1} * X_{(before)} * X_{(after)}^T)^T$

(6) $E_1 = X_{(before)} * X_{(after)}^T * ([X_{(before)} * X_{(after)}^T]^{-1})^T$

In steps 1-2 recasting the problem by matrix transposition is essential, because each state is represented by a column-vector either in $X_{(before)}$ or $X_{(after)}$, where $X_{(before)}$ is an array of states before the transformation $X_{(before)} = [X_{(0)} X_{(1)} \ldots X_{(t-1)}]$ and $X_{(after)}$ is an array of states after the transformation $X_{(after)} = [X_{(1)} X_{(2)} \ldots X_{(t)}]$

Steps 3-4 illustrate an application of the Moore-Penrose pseudoinverse, a widely known type of matrix pseudoinverse, independently introduced by Moore [16], Bjerhammar [17], and Penrose [18]. Finally, in steps 5-6, retranspositions put E1 in proper order.

Derivation of the $(\ddot{R})$ repressive regression algorithm:

Steps

(1) $E_2 * X_{(before)} = X_{(after)}$

(2) $E_2 * X_{(before)} - X_{(before)} = X_{(after)} - X_{(before)}$

(3) $X_{(before)}^T * (E_2 - I)^T = (X_{(after)} - X_{(before)})^T$

(4) $X_{(before)} * X_{(before)}^T * (E_2 - I)^T = X_{(before)} * (X_{(after)} - X_{(before)})^T$

(5) $(E_2 - I)^T = [X_{(before)} * X_{(before)}^T]^{-1} * X_{(before)} * (X_{(after)} - X_{(before)})^T$

(6) $(E_2 - I) = ([X_{(before)} * X_{(before)}^T]^{-1} * X_{(before)} * (X_{(after)} - X_{(before)})^T)^T$

(7) $(E_2 - I) = (X_{(after)} - X_{(before)}) * X_{(before)}^T * ([X_{(before)} * X_{(before)})^T]^{-1})^T$

(8) $E_2 = (X_{(after)} - X_{(before)}) * X_{(before)}^T * ([X_{(before)} * X_{(before)})^T]^{-1})^T + I$

Here, in step 1 we first repress the equation by subtracting $X_{(before)}$ from both sides of the equation. In steps 2-3 we recast the problem by matrix transposition. In steps 4-5 the Moore-Penrose pseudoinverse is applied. And the re-transposition step is introduced in steps 6-7. The identity matrix (I) is added to both sides of the equation to derive E_2 on the left hand side in step 8.

The search for the jacobian matrix solution

It is not difficult to calculate the jacobian matrix once either E_1 or E_2 is found. There are two different methods to consider:

(1) using eigenvalues and eigenvectors;

(2) using a new approximation technique, presented here for the first time.

The difficulty in finding J is in calculating the principal matrix logarithm of E_1 or E_2; that is, the exact inverse of $\exp^{(J*t_c)}$.

$$\exp^{(J*t_c)} = E_1 \approx E_2$$

$$J = \frac{\log m(E_1)}{t_c} \approx \frac{\log m(E_2)}{t_c}$$

Where logm (. . .) represents the matrix logarithm function.

Application of eigenvalues and eigenvectors

Assuming that E_1 or E_2 is diagonalisable, the following method may be used to obtain the jacobian matrix from E_1 or E_2. First we seek to find the matrix υ of eigenvectors of E_1or E_2 as appropriate, referred to here as E_m for convenience (for either case). Each column of υ is an eigenvector of E_m. We then find eigenvectors of E_m from υ and E_m as $eig_m = \upsilon^{-1} * E_m * \upsilon$. Next we replace each diagonal element of eig_m by its natural logarithm and calculate the natural logarithm of E_m as $\log m(E_m) = \upsilon * \log m(eig_m) * \upsilon^{-1}$. Finally, J is then calculated from $\text{logm}(E_m)$ as follows:

$$J = \frac{\log m(E_m)}{t_c} \approx real(\frac{(\upsilon * \log m(eig_m) * \upsilon^{-1})}{t_c}) \qquad (3.9)$$

Note, only real values for parameters of J are considered.

A new approximation technique for calculating the matrix logarithm

It is generally known that $\exp^{(J*\nabla p)}$ is approximately equal to $I + (J * \nabla p)$ on condition that ∇p is a number and small enough [12]. Here, we will introduce a scaling factor μ to t_c in order to approximate ∇p such that $\nabla p \approx t_c * \mu$. We may calculate the value of J to be:

$$I + J * (t_c * \mu) = [\exp^{J*t_c}]^{\mu} \qquad (3.10)$$

$$J * (t_c * \mu) = [\exp^{J*t_c}]^{\mu} - I \qquad (3.11)$$

$$J = \frac{([\exp^{J*t_c}]^{\mu} - I)}{(t_c * \mu)} \qquad (3.12)$$

$$J = \frac{(E_m{}^{\mu} - I)}{(t_c * \mu)} \qquad (3.13)$$

By substituting a small value for μ in the above equation, e.g. $10^{-4} \le \mu \le 10^{-11}$, we may approximate J. Note, the smaller the magnitude of this value the better the result of approximating J becomes. However, care should be taken not to allow the magnitude of this value to be smaller than this range in order to ensure that it is not approximated to zero internally. Note this new approximation technique is equivalent, in terms of the solution obtained, to (3.9).

Linking jacobian matrices and network models

The inference of model parameters from experimental data sometimes requires multivariate regression to be performed on experimental data with the aim of minimising the residual error between the model and the data, particularly in systems of linear and nonlinear ordinary differential equations (ODE). For example, using the following system of linear ordinary differential equations:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \\ \dot{X}_5 \\ \dot{X}_6 \end{bmatrix} = \begin{bmatrix} \frac{\partial X_1}{\partial X_1} & \frac{\partial X_1}{\partial X_2} & \frac{\partial X_1}{\partial X_3} & \frac{\partial X_1}{\partial X_4} & 0 & \frac{\partial X_1}{\partial X_6} \\ 0 & \frac{\partial X_2}{\partial X_2} & \frac{\partial X_2}{\partial X_3} & \frac{\partial X_2}{\partial X_4} & 0 & \frac{\partial X_2}{\partial X_6} \\ 0 & 0 & \frac{\partial X_3}{\partial X_3} & \frac{\partial X_3}{\partial X_4} & \frac{\partial X_3}{\partial X_5} & 0 \\ 0 & 0 & 0 & \frac{\partial X_4}{\partial X_4} & 0 & \frac{\partial X_4}{\partial X_6} \\ \frac{\partial X_5}{\partial X_1} & 0 & 0 & \frac{\partial X_5}{\partial X_4} & \frac{\partial X_5}{\partial X_5} & \frac{\partial X_5}{\partial X_6} \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial X_6}{\partial X_6} \end{bmatrix} * \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix}$$

one can interprete the jacobian matrix to mean a direct representation of the network interaction and systems dynamics as indicated in (Figure 1).

Results

Method validation

To test our approach for reconstructing network models from time series data, we use artificial data generated from known network models. This data is generated by simulating time series data from those models, and reconstructing the original network from the time series data alone with no knowledge of the generative model. Importantly, the original network is not provided to the reconstruction method - only the time series data, and this provides a source of independent test data not used in method construction. We test our method on 100 randomly generated networks as explained in section 4.5. We consider two data discretisation approaches for generating time series data, namely:

(1) discretisation using simple continuous models;

(2) discretisation using application of eigenvectors and eigenvalues.

Our objectives are:

(1) to discretise (multivariate time series) data with known network models (jacobian matrices);

(2) to ensure that the simulated data is noise-free (noiseless);

(3) to ensure that important features of the data such as correlation between the variables are preserved;

(4) to test and evaluate the performance of our inference algorithms based on minimum number of states (<= number of measured variables +1).

The test methods avoid independent simulation of time series data of any single variable; only multivariate discretisation is used. To demonstrate the performance of our inference algorithms we also avoid using the integral function during the discretisation process. First we describe the discretisation techniques used to generate data. We present an expanded example of the method for each data type. Finally, we present summary statistics of performance of the method for reconstruction of a large number of networks generated at random.

Data discretisation using a simple continuous model

The solution to a system of ordinary differential equation $\dot{X}(t) = JX(t)$ is $X(t) = e^{Jt}X_0$ as noted previously, so in order to discretise at regular time step intervals t (in this example our time step is 0.25 seconds), we define our discretisation process at any state k as :

$$X_{(k)} = e^{Jt}.[e^{J(k-1)t}.X_{(0)}] = e^{Jt}.X_{(k-1)} \qquad (4.1)$$

Or at state k+1

$$X_{(k+1)} = e^{Jt}.[e^{J(k)t}.X_{(0)}] = e^{Jt}.X_{(k)} \qquad (4.2)$$

Where $X(_0), X(_{k-1}), X(_k), X(_{k+1})$ are the vector values at the time points 0, k-1, k, and k+1, respectively. 4.2.1. Example 1. Define J as

$$J = \begin{bmatrix} -0.19242 & -0.17738 & -0.80447 & -1.148 & 0 & 0.10009 \\ 0 & -0.19605 & 0.69662 & 0.10487 & 0 & -0.54453 \\ 0 & 0 & 0.83509 & 0.72225 & -0.43897 & 0 \\ 0 & 0 & 0 & 2.5855 & 0 & -0.60033 \\ -1.4224 & 0 & 0 & -0.66689 & 0.84038 & 0.48997 \\ 0 & 0 & 0 & 0 & 0 & 0.73936 \end{bmatrix}$$

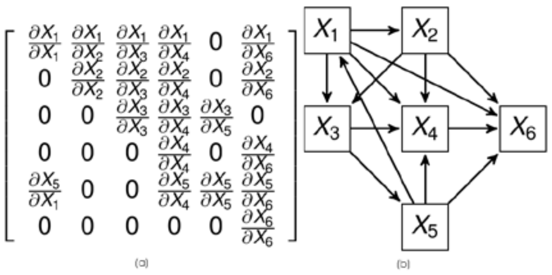
$\begin{bmatrix} \frac{\partial X_1}{\partial X_1} & \frac{\partial X_1}{\partial X_2} & \frac{\partial X_1}{\partial X_3} & \frac{\partial X_1}{\partial X_4} & 0 & \frac{\partial X_1}{\partial X_6} \\ 0 & \frac{\partial X_2}{\partial X_2} & \frac{\partial X_2}{\partial X_3} & \frac{\partial X_2}{\partial X_4} & 0 & \frac{\partial X_2}{\partial X_6} \\ 0 & 0 & \frac{\partial X_3}{\partial X_3} & \frac{\partial X_3}{\partial X_4} & \frac{\partial X_3}{\partial X_5} & 0 \\ 0 & 0 & 0 & \frac{\partial X_4}{\partial X_4} & 0 & \frac{\partial X_4}{\partial X_6} \\ \frac{\partial X_5}{\partial X_1} & 0 & 0 & \frac{\partial X_5}{\partial X_4} & \frac{\partial X_5}{\partial X_5} & \frac{\partial X_5}{\partial X_6} \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial X_6}{\partial X_6} \end{bmatrix}$
X1
X2
X3
X4
X5
X6
(a)
(b)

Figure 1: Structure representation: (a) jacobian matrix of a network; (b) its corresponding network with inter-connected nodes.

and the initial condition, $X_{(0)}$, as:

$$X_{(0)} = \begin{bmatrix} 1.7119 \\ -0.19412 \\ -2.1384 \\ -0.83959 \\ 1.3546 \\ -1.0722 \end{bmatrix}$$

Therefore, the state vector $X_{(1)} = e^{J.t} * X_{(0)}$ is:

$$X_{(1)} = \begin{bmatrix} 2.4189 \\ -0.48906 \\ -2.9903 \\ -1.3564 \\ 0.90592 \\ -1.2898 \end{bmatrix}$$

where t=0.25 seconds. Likewise, state vector $X_{(6)} = e^{J.6*t} * X_{(0)}$ is then calculated and the result is:

$$X_{(6)} = \begin{bmatrix} 19.4655 \\ -6.0087 \\ -16.122 \\ -24.7886 \\ -10.9378 \\ -3.2502 \end{bmatrix}$$

Note that $X(_6)$ could have been calculated from X_1 as: $X(_6) = e^{J.5*t} * X(_1)$. If we defined a time series dataset $DS_1 = [X(_1)\, X(_2)\, X(_3)\, X(_4)\, X(_5)\, X(_6)]$, i.e., a time series data set for the next six states at regular interval of 0.25 seconds after the initial condition, then DS would be:

$$DS = \begin{bmatrix} 2.4189 & 3.4924 & 5.1484 & 7.7765` & 12.0971 & 19.4655 \\ -0.48906 & -0.92226 & -1.5502 & -2.4683 & -3.8477 & -6.0087 \\ -2.9903 & -4.1059 & -5.6107 & -7.7393 & -10.9459 & -16.122 \\ -1.3564 & -2.293 & -4.0204 & -7.245 & -13.3126 & -24.7886 \\ 0.90592 & 0.1048 & -1.2125 & -3.269 & -6.3694 & -10.9378 \\ -1.2898 & -1.5517 & -1.8667 & -2.2457 & -2.7017 & -3.2502 \end{bmatrix}$$

Data discretisation using application of eigenvectors and eigenvalues

It is widely known that the solution set of the system of ordinary differential equations $\dot{X}(t) = J.X(t)$ can be represented by any combination of exponential functions of eigenvalues and their eigenvectors in the form:

$$\begin{bmatrix} X_1(t) \\ X_2(t) \\ \vdots \\ X_m(t) \end{bmatrix} = \begin{bmatrix} e^{\lambda_1 t}.[\upsilon_{11}].[p_1] + e^{\lambda_2 t}.[\upsilon_{12}].[p_2] + \cdots + e^{\lambda_n t}.[\upsilon_{1m}].[p_m \\ e^{\lambda_1 t}.[\upsilon_{21}].[p_1] + e^{\lambda_2 t}.[\upsilon_{22}].[p_2] + \cdots + e^{\lambda_n t}.[\upsilon_{2m}].[p_m] \\ \vdots \\ e^{\lambda_1 t}.[\upsilon_{m1}].[p_1] + e^{\lambda_2 t}.[\upsilon_{m2}].[p_2] + \cdots + e^{\lambda_m t}.[\upsilon_{mm}].[p_m] \end{bmatrix}$$

where the initial condition, X_0, is represented as a linear combination of the eigenvectors of J [12]. Through further analysis, we introduce the matrix form of the initial condition as

$$\begin{bmatrix} X_1(0) \\ X_2(0) \\ \vdots \\ X_m(0) \end{bmatrix} = \begin{bmatrix} \upsilon_{11} & \upsilon_{12} & \cdots & \upsilon_{1m} \\ \upsilon_{21} & \upsilon_{22} & \cdots & \upsilon_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \upsilon_{m1} & \upsilon_{m2} & \cdots & \upsilon_{mm} \end{bmatrix} . \begin{bmatrix} p_1 & \cdots & \cdots & \cdots \\ \cdots & p_2 & \cdots & \cdots \\ \cdots & \cdots & \vdots & \cdots \\ \cdots & \cdots & \cdots & p_m \end{bmatrix} . \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

so that we might present our general matrix form solution to be

$$\begin{bmatrix} X_1(t) \\ X_2(t) \\ \vdots \\ X_m(t) \end{bmatrix} = \begin{bmatrix} \upsilon_{11} & \upsilon_{12} & \cdots & \upsilon_{1m} \\ \upsilon_{21} & \upsilon_{22} & \cdots & \upsilon_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \upsilon_{m1} & \upsilon_{m2} & \cdots & \upsilon_{mm} \end{bmatrix} \cdot \begin{bmatrix} p_1 & \cdots & \cdots & \cdots \\ \cdots & p_2 & \cdots & \cdots \\ \cdots & \cdots & \vdots & \cdots \\ \cdots & \cdots & \cdots & p_m \end{bmatrix} \cdot \begin{bmatrix} e^{\lambda_1 .t} \\ e^{\lambda_2 .t} \\ \vdots \\ e^{\lambda_m .t} \end{bmatrix}$$

where each column vector in

$$\begin{bmatrix} \upsilon_{11} & \upsilon_{12} & \cdots & \upsilon_{1m} \\ \upsilon_{21} & \upsilon_{22} & \cdots & \upsilon_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \upsilon_{m1} & \upsilon_{m2} & \cdots & \upsilon_{mm} \end{bmatrix}$$

is an eigenvector of the jacobian matrix J and the parameter set $\{\lambda_1, \lambda_2, \ldots \lambda_n\}$ is a set of the eigenvalues of J. The parameter e is used to denote the exponential of 1. Note that the parameter set $\{p_1, p_2, \ldots p_m\}$ can easily be calculated by regression analysis [12]. Using the example in the previous section where

$$J = \begin{bmatrix} -0.19242 & -0.17738 & -0.80447 & -1.148 & 0 & 0.10009 \\ 0 & -0.19605 & 0.69662 & 0.10487 & 0 & -0.54453 \\ 0 & 0 & 0.83509 & 0.72225 & -0.43897 & 0 \\ 0 & 0 & 0 & 2.5855 & 0 & -0.60033 \\ -1.4224 & 0 & 0 & -0.66689 & 0.84038 & 0.48997 \\ 0 & 0 & 0 & 0 & 0 & 0.73936 \end{bmatrix}$$ the eigenvalues and eigenvectors of J are calculated to be:

real(eigVec) =

$$\begin{bmatrix} 0.085886 & 0.085886 & -0.30478 & -0.30478 & -0.44332 & 0.14915 \\ -0.054869 & -0.054869 & 0.6724 & 0.6724 & 0.11344 & -0.56798 \\ 0.1344 & 0.1344 & -0.15878 & -0.15878 & 0.32878 & -0.27401 \\ 0 & 0 & 0 & 0 & 0.82485 & 0.21687 \\ -0.74638 & -0.74638 & -0.39895 & -0.39895 & 0.04612 & 0.29707 \\ 0 & 0 & 0 & 0 & 0 & 0.66692 \end{bmatrix}$$

Imag(eigVec) =

$$\begin{bmatrix} 0.32489 & -0.32489 & 0.34644 & -0.3644 & 0 & 0 \\ -0.025758 & 0.25758 & 0 & 0 & 0 & 0 \\ -0.49251 & 0.49251 & 0.11919 & -0.11919 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.3693 & -0.3693 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

eigVec=

$$\begin{bmatrix} 0.0859+0.325i & 0.0859-0.325i & -0.30478+0.346i & -0.30478-0.346i & -0.443 & 0.14915 \\ -0.0549-0.258i & -0.0549+0.258i & 0.6724 & 0.6724 & 0.113 & -0.56798 \\ 0.1344-0.491i & 0.1344+0.491i & -0.15878+0.119i & -0.15878-0.119i & 0.328 & -0.27401 \\ 0 & 0 & 0 & 0 & 0.824 & 0.21687 \\ -0.74638 & -0.74638 & -0.39895+0.36i & -0.39895-0.36i & 0.046 & 0.29707 \\ 0 & 0 & 0 & 0 & 0 & 0.66692 \end{bmatrix}$$

eigVal=

$$\begin{bmatrix} 1.004+0.6191i0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.004-0.6191i & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.36055+0.1235i & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.36055-0.1235i & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.5855 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.73936 \end{bmatrix}$$

Setting the initial condition to

$$X_{(0)} = \begin{bmatrix} X_1(0) \\ X_2(0) \\ X_3(0) \\ X_4(0) \\ X_5(0) \\ X_6(0) \end{bmatrix} = \begin{bmatrix} 1.7119 \\ -0.19412 \\ -2.1384 \\ -0.83959 \\ 1.3546 \\ -1.0722 \end{bmatrix} = eigVec * P * \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

implies that

$$P=\begin{bmatrix} -0.9804-2.2804i & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.9804+2.2804i & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.020446-0.5584i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.020446+0.5584i & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.59519 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1.6076 \end{bmatrix}$$

with this estimated parameter set found through regression analysis. Therefore, we define our second discretisation process at any time point t as :

$$X_{(0)} = \begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \\ X_5(t) \\ X_6(t) \end{bmatrix} = eigVec * P * \begin{bmatrix} e^{(eigVal_{11}*t)} \\ e^{(eigVal_{22}*t)} \\ e^{(eigVal_{33}*t)} \\ e^{(eigVal_{44}*t)} \\ e^{(eigVal_{55}*t)} \\ e^{(eigVal_{66}*t)} \end{bmatrix}$$

Therefore, the timepoint at time t → 0.25 secs becomes:

$$X_{(1)} = \begin{bmatrix} X_1(0.25) \\ X_2(0.25) \\ X_3(0.25) \\ X_4(0.25) \\ X_5(0.25) \\ X_6(0.25) \end{bmatrix} = eigVec * P * \begin{bmatrix} e^{(eigVal_{11}*0.25)} \\ e^{(eigVal_{22}*0.25)} \\ e^{(eigVal_{33}*0.25)} \\ e^{(eigVal_{44}*0.25)} \\ e^{(eigVal_{55}*0.25)} \\ e^{(eigVal_{66}*0.25)} \end{bmatrix} = \begin{bmatrix} 2.4189-0.0i \\ -0.48906 \\ -2.9903+0.0i \\ -1.3564 \\ 0.90592+0.0i \\ -0.2898 \end{bmatrix}$$

We define time series dataset $DS_2 = [X_1\ X_2\ X_3\ X_4\ X_5\ X_6]$ DS_2=

$$\begin{bmatrix} 2.4189-0.0i & 3.4924-0.0i & 5.1484-0.0i & 7.7765-0.0i & 12.0971-0.0i & 19.4655-0.0i \\ -0.48906 & -0.92226+0.0i & -1.5502+0.0i & -2.4683+0.0i & -3.8477+0.0i & -6.0087+0.0i \\ -2.9903+0.0i & -4.1059+0.0i & -5.6107+0.0i & -7.7393+0.0i & -10.9459+0.0i & -16.122+0.0i \\ -1.3564 & -2.293 & -4.0204 & -7.245 & -13.3126 & -24.7886 \\ 0.90592+0.0i & 0.1048+0.0i & -1.2125+0.0i & -3.269+0.0i & -6.3694+0.0i & -10.9378+0.0i \\ -1.2898 & -1.5517 & -1.8667 & -2.2457 & -2.7017 & -3.2502 \end{bmatrix}$$

Reverse engineering results

We now demonstrate how well our reverse engineering (inverse problem solution) algorithms work using the time series created in the last section where

$$\begin{bmatrix} time(\sec s.) \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} = \begin{bmatrix} 0 & 0.25 & 0.5 & 0.75 & 1 & 1.25 & 1.5 \\ 1.7119 & 2.4189 & 3.4929 & 5.1484 & 7.7765 & 12.0971 & 19.4655 \\ -0.1941 & -0.48906 & -0.92226 & -1.5502 & -2.4683 & -3.8477 & -6.0087 \\ -2.1384 & -2.9903 & -4.1059 & -5.6107 & -7.7393 & -10.9459 & -16.122 \\ -0.8396 & -1.3564 & -2.293 & -4.0204 & -7.245 & -13.3126 & -24.7886 \\ 1.3546 & 0.90592 & 0.1048 & -1.2125 & -3.269 & -6.3694 & -10.9378 \\ -1.0722 & -1.2898 & -1.5517 & -1.8667 & -2.2457 & -2.7017 & -3.2502 \end{bmatrix}$$

and we define

$$X_{(\text{before})} = \begin{bmatrix} 1.7119 & 2.4189 & 3.4924 & 5.1484 & 7.7765 & 12.0971 \\ -0.19412 & -0.48006 & -0.92226 & -1.5502 & -2.4683 & -3.8477 \\ -2.1384 & -2.9003 & -4.1059 & -5.6107 & -7.7393 & -10.9459 \\ -0.83959 & -1.3564 & -2.293 & -4.0204 & -7.245 & -13.3126 \\ 1.3546 & 0.90592 & 0.1048 & -1.2125 & -3.269 & -6.3694 \\ -1.0722 & -1.2898 & -1.5517 & -1.8667 & -2.2457 & -2.7017 \end{bmatrix}$$

$$X_{(after)} = \begin{bmatrix} 2.4189 & 3.4924 & 5.1484 & 7.7765 & 12.0971 & 19.4655 \\ 0.48906 & -0.92226 & -1.5502 & -2.4683 & -3.8477 & -6.0087 \\ -2.9903 & -4.1059 & -5.6107 & -7.7393 & -10.9459 & -16.122 \\ -1.3564 & -2.293 & -4.0204 & -7.245 & -13.3126 & -24.7886 \\ 0.90592 & 0.1048 & -1.2125 & -3.269 & -6.3694 & -10.9378 \\ -1.2891 & -1.5517 & -1.8667 & -2.2457 & -2.7017 & -3.2502 \end{bmatrix}$$

and $t_c = 0.25$

4.4.1. Example 1a: Application of the $(\ddot{T})$ Transposive Regression Algorithm. First calculate E_1 from $E_1 = X_{(after)} * X_{(before)} * ([X_{(before)} * X_{(before)}{}^T]^{-1})^T$

$$\vec{E_1} = \begin{bmatrix} 0.95158 & -0.042221 & -0.22248 & -0.42058 & 0.012662 & 0.059982 \\ 0.0012321 & 0.95216 & 0.18909 & 0.057435 & -0.010828 & -0.15008 \\ 0.022125 & -0.00028992 & 1.2306 & 0.28972 & -0.13526 & -0.027802 \\ 0.0000076294 & -0.000045776 & 0.0000076294 & 1.9086 & 0.0000019073 & -0.22946 \\ -0.38654 & 0.0082092 & 0.041029 & -0.18741 & 1.2323 & 0.15822 \\ -0.0000019073 & 0.0000038147 & 0.0000019073 & 0 & 0.00000023842 & 1.203 \end{bmatrix}$$

Then use either of the matrix logarithm techniques, introduced in Section 3.4, to reverse engineer J

$$J = \begin{bmatrix} -0.19248 & -0.17737 & -0.80446 & -1.148 & 0.0000042473 & 0.1001 \\ 0.0000067921 & -0.19605 & 0.69661 & 0.10488 & -0.00000047751 & -0.54451 \\ 0.000033966 & 0.000102 & 0.83509 & 0.72223 & -0.43896 & 0.00010752 \\ 0.000022578 & -0.00013168 & 0.000031848 & 2.5855 & 0.0000061946 & -0.60039 \\ -1.4224 & 0.000018331 & 0.00000762 & -0.6669 & 0.84038 & 0.49001 \\ -0.0000069427 & 0.000014074 & 0.0000044721 & -0.0000014319 & 0.000001096 & 0.73935 \end{bmatrix}$$

4.4.2. Example 1b: Application of the $(\ddot{R})$ Repressive Regression Algorithm. E_2 is calculated from $E_2 = (X_{(after)} - X_{(before)}) * X_{(before)}{}^T * ([X_{(before)} * X_{(before)}{}^T]^{-1})^T + I$.

$\rightarrow$

$$E_2 = \begin{bmatrix} 0.9516 & -0.042213 & -0.22248 & -0.42057 & 0.012661 & 0.059998 \\ 0.0012293 & 0.95216 & 0.1891 & 0.057434 & 0.010828 & -0.15009 \\ 0.022114 & -0.00031662 & 1.2306 & 0.28973 & -0.13527 & -0.027828 \\ 0.0000038147 & 0.000015259 & 0.0000019073 & 1.9086 & -0.00000047684 & -0.22943 \\ -0.38653 & 0.0082054 & 0.041027 & -0.18741 & 1.2323 & 0.15821 \\ 0 & -0.00000095367 & 0.00000023842 & 0.00000011921 & 0.00000011921 & 1.203 \end{bmatrix}$$

Reverse engineering J from E_2 then produces:

$$J = \begin{bmatrix} -0.19241 & -0.17731 & -0.80448 & -1.1479 & -0.0000046917 & 0.10017 \\ -0.00000044025 & -0.19605 & 0.69663 & 0.10487 & 0.00000066244 & -0.54452 \\ -0.0000064513 & -0.000016899 & 0.83509 & 0.72225 & -0.43896 & 0.0000048789 \\ 0.000010899 & 0.000044338 & 0.0000026542 & 2.5855 & -0.0000098578 & -0.6008 \\ -1.4223 & 0.000031253 & -0.000002115 & -0.66688 & 0.84037 & 0.49 \\ 0.00000007547 & -0.0000035556 & 0.0000010714 & 0.00000031275 & 0.00000043957 & 0.73936 \end{bmatrix}$$

Performance evaluation of algorithms

Network structure identification: The performance of the two algorithms introduced in this paper are evaluated in terms of their ability to identify original (unseen) network structures. Before parameter estimation, model structures should be determined. The approximated network structure is easily derivable from the jacobian matrix (derivable from the zero and non-zero entries in the matrix representation of the system of ODEs), regardless of the magnitude of the parameter entries. Therefore, we simplify a weighted jacobian maxtrix into its Boolean form, revealing the network structure (model architecture) in terms of presence and absence of links. These structures are in form of matrices containing only Boolean (0 or 1) entries: an entry of 1 depicts presence of an association (non-zero parameter in the jacobian matrix); otherwise 0 for no association. This initial approximation determines the network topology (see Figure 2).

Network connectivity ratio: Given a (Boolean) matrix representation of the inferred jacobian matrix (network structure), we define network connectivity ratio as $\dfrac{NumOnes}{(NumOfOnes + NumOfZeroes)}$ where NumOfOnes and NumOfZeroes indicate the total number of 1s and 0s in the Boolean matrix respectively.

Metric criteria: Three key properties of network representation are used for performance evaluation:

a) $\dfrac{Miss}{Total}$ ratio - a relative ratio of the number of missing correct links in the inferred matrix to the total number of links inferred; b) $\dfrac{Incorrect}{Total}$ relative ratio in terms of number of incorrect links in the inferred matrix to the total number of links inferred; and c) the norm score based on $\dfrac{Miss}{Total}$ and $\dfrac{Incorrect}{Total}$ ratios - an indication of the degree of closeness of the predicted network to the original network, measured as $\left((\dfrac{Miss}{Total})^2 + (\dfrac{Incorrect}{Total})^2\right)^{\frac{1}{2}}$.

Algorithm performance

We tested our algorithms on 700 simulated datasets with a range of numbers of time points (4-10), all generated from networks with 25% connectivity ratio.

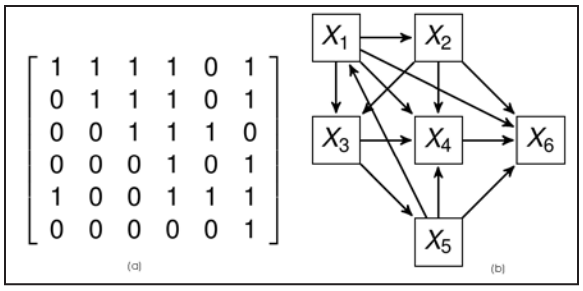

Figure 2: Relation between the derived Boolean representation of the jacobian matrix and the corresponding network topology with inter-connected nodes.

Well-formed jacobian matrices of artificial systems are required to generate data for performance evaluation of our method. A method for constructing nonsingular matrices was used to generate our artificial network data. Based on pameterisation of matrices implied determinants and minors, we were able to randomly generate a large set of new nonsingular jacobian matrices that were used to simulate test data by operating and manipulating products of factors of nonsingular matrices to guarantee that the jacobian matrices produced were not defective. With hundreds of such matrices, we were able to generate a wide range of different artificial data to test our method. The number of non-zero parameters in each of those matrices determined the network connectivity ratio for that system. Here, we fixed this to be 25% of the total parameters in a given matrix. Hence our matrices have 25% connectivity. Finally the results of the predicted networks were then compared to the corresponding information of the original networks recorded in the database.

Based on the performance evaluation criteria $\frac{Miss}{Total}$, $\frac{Incorrect}{Total}$ and the rank (norm) score, we analysed the results of the inferred network structure and measured deviation of network structure size in terms of network connectivity percentage. We assume that an inferred network structure has the potential to be a reasonable or good result if its connectivity ratio is between 20 and 30%. Table 1 shows that at least 60% of the number of dependent variables (here 6) is the minimum number of time points required to obtain a reasonable or good inference result. However, with our methods the reconstructed models are often data-consistent, that is, have the capacity to simulate or reproduce exactly the given dataset, irrespective of the number of time points (even when there are only a few time points). The evaluation criteria have been established to measure variation in performance depending on the number of time points. The results confirm that performance improves with an increase in the number of time points (a decrease in rank score indicates an increase in performance). The whole performance evaluation process is automated and does not need supervision or user intervention.

The network connectivity ratio indicates the estimated number of model parameters identified from (and used to explain) the available data set, which means that the richer the dataset, in terms of number of time points, the better the probability of identifying the correct links (or model parameters) that are suitable or valid, e.g. in (Table 1), the first row indicated that on average at least 13 model parameters (predicted from data to be non-zero) were ascertained to be valid parameters out of a total number of 25, whereas the last row showed that at least 24 model parameters were identified as being valid. All (10) diagonal elements were included as valid entries by default.

Not every identified link is valid, although often the majority of them are. The network connectivity ratio reveals the complexity of the predicted networks, i.e.the total number of correct (and incorrect) links in the predicted network. The $\frac{Miss}{Total}$ value, as previously noted, reflects the number of correct links predicted. As mentioned previously, the norm score based is based on the calculation $\left((\frac{Miss}{Total})^2 + (\frac{Incorrect}{Total})^2\right)^{\frac{1}{2}}$. We assume that the lower the norm score the better the performance of the inference method. A high valued Norm Score might still produce a data-constistent model, but the jacobian matrix of such system would be too sparse for the output structure to be a reasonable representation of the underlying data.

The result shows that performance, in terms of network structure identification, improves with an increase in the number of time points. The rank (norm) score (error ratio) value, a function of $\frac{Miss}{Total}$ and $\frac{Incorrect}{Total}$ ratios, confirms approximately $\frac{(1-0.115)}{1} * 100 = 88.5\%$ success rate in inference of network structure for datasets with size of 10 time points. It is remarkable that with such limited information on data and no information on topology, our inference methods are able to infer completely (100%) the actual network structure at times.

Algorithm #	Size (no. of time points)	Network connectivity ratio (%) - number of non-zero parameters	Miss / total ratio	Incorrect / total ratio	Rank (norm) score
1	4	13	0.94167	0.14167	0.9575
2	4	13	0.94167	0.14167	0.9575
1	5	16	0.63918	0.14145	0.65904
2	5	16	0.63918	0.14145	0.65904
1	6	24	0.48424	0.33622	0.59018
2	6	24	0.48424	0.33622	0.59018
1	7	23	0.34182	0.17576	0.38604
2	7	23	0.34182	0.17576	0.38604
1	8	23	0.23455	0.097917	0.25519
2	8	23	0.23455	0.097917	0.25519
1	9	23	0.14268	0.032794	0.14787
2	9	23	0.14268	0.032794	0.14787
1	10	24	0.1063	0.043982	0.11522
2	10	24	0.10797	0.03455	0.11377

Results of transposive regression (#1) and repressive regression (#2) algorithms for a range of numbers of time points for networks of 10 interacting components. 100 network instances each of seven (7) different datasets are simulated and tested using performance criteria established in section 4.5.

Table 1: Summary statistics of algorithm performances.

In summary, the results show an improvement in method performance with increasing data size. Overall performance is close to optimum, i.e the original network is recovered with approximately 88.5% success rate on average, when the number of time points available is equal to number of measured variables even though this network is unseen to the algorithm and there are many possible data-consistent weighted networks. The main challenge is in keeping both the $\frac{Miss}{Total}$ and $\frac{Incorrect}{Total}$ ratios as low as possible. These two metrics may also be used in robustnessness and sensitivity analysis of any proposed method, keeping in mind that the primary objective of any proposed method is to minimise $\frac{Miss}{Total}$ and $\frac{Incorrect}{Total}$ values. Ultimately, the challenge is to preserve data-consistent model generation while at the same time maximising the likelihood of identifying the original set of links by inference.

Conclusion

Clearly, network structure identification and parameter estimation of dynamical systems are necessary steps in representing system dynamics in terms interaction networks. We demonstrate that algorithmic analysis of time series data may produce data-consistent models. On a promising note, the novel inference algorithms presented in this paper are identified, through simulation experiment and testing, to develop such data-consistent models. As demonstrated in this paper using a worked example, there is a strong theoretical basis for their use in time series data analysis. Moreover, their utility is demonstrated by their performance result under testing conditions using artificial data sets generated in silico.

We assessed the performance of our unsupervised inference algorithm using hundreds of test networks, that is networks that were used to generate randomly valued, independent test data, through two different methods and similar in form (but not values) to the worked example, and importantly the underlying network structure was never provided to the algorithm in this validation. We demonstrated significant improvement in network reconstruction as more data became available, here increasing time series time points from 3 to 10, and showed very good performance as the data size tends to 10 time points. We recognise that 3 data points is a very small data set, but show that even with this limited time series data we are able to reconstruct a data-consistent network. Our algorithms are aimed at simplifying and standardising the methods of finding unique solutions whenever they exist and using those standardised methods to adequately find other potential data-consistent solutions in non-unique scenarios. Of course, as time series data reduces in size, the number of possible networks able to explain the data increases. Note that this ability to work with limited data can be combined with the capacity for the approach to include a priori knowledge, and this knowledge may substantially reduce the solution space. Consequently the approach can blend available knowledge with knowledge gaps to produce data-consistent models of system dynamics.

The application of this algorithm in systems biology extends to any time series data. Where those systems are better described by log-linear processes, our algorithmic approach is still valid. We are especially interested in the impact of cancer drug intervention strategies on the (human) cell signalling network (for example [20]). This cell signalling model describes the PI3K/AKT signalling network and considers the effects of different perturbations on the network response to growth receptor inhibition. By perturbing the system with various mutations, distinct regimes of functioning were observed in the network. Specifically regimes where the system was sensitive to intervention, where inhibition of the input signal led to inhibition of the output signal, and where the system was resistant to intervention, that is where the system was robust to input signal inhibition. Moreover, the transition between sensitivity and resistance was governed by a control parameter derived from the relative balance of the activities of three enzymes and drug interventions that target this balance may effect a shift in therapeutic resistance to therapeutic sensitivity.

Models such as [20] are powerful approaches to understanding biological mechanisms, since they represent the underlying processes involved. However, they are both data hungry, with hundreds of parameters requiring calibration from (typically) cell line data and/ or estimation based on known system-scale behaviours, and resource intensive to construct. Their development is based on provisional knowledge [21] as not all network associations are known. We propose that the data-driven methods developed here may support process-based model development and investigation. For example, given two sets of time series data from a cell line, where one set represents a control condition and another an intervention condition, our algorithm can construct an interaction network for each of the two data sets. A comparison of the constructed networks would reveal the perturbation manifest in the response of the cell line to the intervention introduced. Any significant differences in the networks would suggest areas of the signalling network most impacted on by that intervention. In turn, this may direct theoretical investigations, such as targetting areas in the signalling network that are highly sensitive to perturbation. More fundamentally, our inference algorithm can support process-based model calibration and validation: time series data from the process-based model should generate similar networks to the cell line data. Finally, it may help interpret model dynamics: by analysing time series data generated from the process-based model in different functioning regimes meta-level associations, not limited to the network topology intrinsic to the process-based model, may be elucidated. For example, an intervention may introduce a marked increase in overall association among a subset of measurables - not necessarily in a localised region of the signalling network topology and this can reveal inform understanding of system-scale responses to that intervention [22].

The inference methods presented in this paper are robust and flexible enough to partly incorporate existing knowledge of network structure prior to estimation of model parameters, and this reduces the number of possible networks. It is worth investigating the nature of the factors that contribute to this non-uniqueness in model outcomes. Of note is that for some networks, even when large amounts of data are available, non-uniqueness is still a challenge. In subsequent work we will propose the use of a combinatorial methodological approach to network inference and its application to analysis of pseudo-data generated from real process-based models of biological systems. These combinatorial methods are founded on the work proposed here, and we propose that the methods can serve as fundamental algorithms upon which other variant inference algorithms may be built.

Citation: Idowu MA, Bown J (2011) Towards an Exact Reconstruction of a Time-Invariant Model from Time Series Data. J Comput Sci Syst Biol 4: 055-070. doi:10.4172/jcsb.1000077

References

1. Brazhnik P, de la Fuente A, Mendes P (2002) Gene networks: how to put the function in Genomics. Trends Biotechnol 20: 467-472.
2. Voit E.O (2000) Computational analysis of biochemical systems: A practical guide for bio-chemists and molecular biologists. Cambridge University Press, Cambridge, UK.
3. Chen L, Wang RS, Zhang X (2009) Biomolecular networks methods and applications in systems biology. John Wiley and Sons.
4. Pappalardo F, Zhang P, Halling-Brown M, Basford K, Scalia A, et al. (2008) Computational simulations of the immune system for personalized medicine: state of the art and challenges. Current Pharmacogenomics and Personalized Medicine 6: 260-271.
5. Citri A, Yarden Y (2006) EGF-ERBB signalling: towards the systems level. Nat Rev Mol Cell Biol 7: 505-516.
6. McAdam E, Falconer RE, Bown JL, Crawford, JW (2011) Fungi as metaphors for resource management. Adaptive 2011.
7. Rinaldi JPS, Kelly T (2001) Identifying understanding and analyzing critical infrastructure interdependencies. Control Systems, IEEE 21: 1125
8. Lennart L jung (2008) Perspectives on system identification. Proc 17th IFAC world congress, Seoul, South Korea.
9. Simon P. Burke, John Hunter (2005) Modelling non-stationary economic time series: a multivariate approach. Palgrave Macmillan.
10. Gibson M, Mjolsness E (2004) Modelling the activity of single genes, in computational modelling of genetic and biochemical networks. Bower JM, bolou, MIT press, Cambridge, MA.
11. Liao JC, Boscolo R, Yang YL, Tran LM, Sabatti C, et al. (2003) Network component analysis: reconstruction of regulatory signals in biological systems. Proc Natl Acad Sci USA 100: 15522-15527
12. Strang, Gilbert (1988) Linear algebra and its applications. Third edition. ISBN: 0155510053.
13. Burke JV, Lewis AS, Overton ML (2003) Robust stability and a criss-cross algo-rithm for pseudospectra. IMA Journal of Numerical Analysis 23 (3), 359-375. doi:10.1093/imanum/23.3.359.
14. Sheldon Axler (1995) Down with determinants! American Mathematical Monthly 102: 139-154.
15. Tsai KY, Wang FS (2005) Evolutionary optimization with data collocation for re-verse engineering of biological networks. Bioinformatics 21: 1180-1188.
16. Moore EH (1920) On the reciprocal of the general algebraic matrix. Bulletin of the American Mathematical Society 26: 394395. projecteuclid.org/euclid.bams/1183425340.
17. Bjerhammar, Arne (1951) Application of calculus of matrices to method of least squares; with special references to geodetic calculations. Trans. Roy. Inst. Tech. Stockholm 49.
18. Penrose, Roger (1955) A generalized inverse for matrices. Proceedings of the Cambridge Philosophical Society 51: 406413. doi:10.1017/S0305004100030401.
19. Gardner TS, di Bernardo D, Lorenz D, Collins JJ (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. Science 301: 102-105.
20. Goltsov A, Faratian D, Langdon SP, Bown J, Goryanin I, et al. (2011) Compensatory effects in the PI3K/PTEN/AKT signaling network following receptor tyrosine kinase inhibition. Cell Signal 23: 407-416.
21. Brown KS, Hill CC, Calero GA, Myers CR, Lee KH, et al. (2004) The statistical mechanics of complex signaling networks: nerve growth factor signaling. Phys Biol 1: 184-195.
22. Michael A Idowu, Alexey Goltsov, Hilal S Khalil, Hemanth Tummala, Nikolai Zhelev, et al. (2011) Cancer research and personalised medicine: a new approach to modeling time-series data using analytical methods and Half systems, Current Opinion in Biotechnology, Volume 22, Supplement 1, Page S59, ISSN 0958-1669,10.1016/j.copbio.2011.05.163.