Combining Autosmooth and Alternating Regression for the Estimation of S-System Parameters

Wolfgang Beyer

January 2008

INESC-ID Technical Report 29/2008

Preface

This report is the result of a research internship I did at the KDBIO group at INESC-ID in Lisbon in the summer semester of 2007 supervised by Prof. Susana Vinga (INESC-ID/FCM-UNL) and co-supervised by Prof. Ana Teresa Freitas (INESC-ID/IST) as part of the projects MaGiC (IE02ID01004) from INESC-ID (A.T. Freitas, PI) and DynaMo (PTDC/EEA-ACR/69530/2006) from FCT (S. Vinga, PI). Apart from my supervisors, whom I would like to thank for their constant support and encouragement, I would also like to thank the DynaMo members Prof. Jonas Almeida, Dr. Andreas Bohn, Paula Gaspar, Prof. João Miranda Lemos, Dr. Ana Rute Neves, Rodrigo Piedade, Prof. Helena Santos, Marco Vilela and Prof. Eberhard Voit.

The general research area I was involved in was the mathematical modeling of biochemical systems from time series data of metabolite concentrations. My task was to use the Matlab software environment to combine autosmooth [1], an algorithm that smoothes noisy time series and estimates their slopes, with alternating regression [2], which estimates the parameters of an S-system. I also evaluated the joint performance of the two algorithms.

Abstract

Accurate time course data of metabolite concentrations in living cells or whole organisms generated by methods of modern molecular biology allows new approaches to mathematically model the biochemical processes taking place. This kind of data contains a wealth of information about the structure and dynamics of the biochemical system which generates them. The bottleneck in the analysis of such data is the estimation of parameter values. Because of its non-linearity and high dimensionality this problem is challenging and computationally expensive and a lot of effort has already been put in improving the available methods.

In this report two recently published methods – autosmooth [1] for smoothing and estimation of slopes and alternating regression [2] for the actual parameter estimation - are combined for the task of generating an S-system model from raw metabolic time courses. The results show that these methods indeed show some improvement over previous methods that directly estimate systems of nonlinear differential equations. While the combination of these methods may be helpful to a skilled modeler, the goal of generating a good model from time course data without human intervention is still rather far away.

Index

Preface ... 3
Abstract ... 3
Index ... 5
Introduction ... 7
Systems and Methods ... 8
The S-System modeling framework ... 8
Decoupling ... 8
Smoothing ... 8
Alternating Regression ... 9
Results ... 11
The example network ... 11
Adding noise and filtering ... 12
Performing alternating regression ... 12
The performance ... 13
Case 1: no noise, true derivatives ... 13
Case 2: noisy time series, true derivatives ... 13
Case 3: no noise, slopes via autosmooth ... 13
Case 4: noisy function values, slopes via autosmooth ... 15
Case 5: noisy data, function values and derivatives via autosmooth ... 16
Discussion ... 17
Dealing with complex parameter values ... 17
Using the parameter estimates for predictions ... 18
Mass conservation and other restrictions ... 19
Alternative approaches for parameter estimation ... 20
Conclusion ... 21
References ... 21
Appendix ... 24

Introduction

Systems biology studies the interactions between the components of biological systems, and how these interactions give rise to the function and behavior of that system. An important part of systems biology is the modeling of metabolic networks, which consists of various different tasks. First of all, one needs data that is suitable for building a model upon it. Nowadays modern high-throughput techniques of molecular biology are able to generate data that contains a huge amount of information, but the extraction of this information is not a straightforward task. Especially useful for the generation of metabolic models are time series data of the concentrations of the metabolites involved in a certain pathway. In vivo nuclear magnetic resonance (NMR) measurements are able to produce exactly this kind of data, which contains information about both the material flow and the regulation within the network (see [3] for an example and [4-6] for applications).

The modeler’s task is subsequently to identify the structure of the network and to estimate the model’s parameters. By using the S-systems modeling framework within Biochemical Systems Theory (BST) [7-9] the complex structure identification task can be reduced to a parameter estimation task, albeit the number of parameters and consequently the dimensionality of the parameter search space is increased significantly in that step. S-systems have been proven to be flexible enough to be able to handle the most dissimilar metabolic networks and have been discussed in the literature numerous times already [10-12]. Another advantage of using S-systems is that every single parameter can be directly interpreted in a biological sense, offering insight into the topological structure of the network and into the orders of the chemical reactions involved.

S-system models are nonlinear, therefore the parameter estimation task corresponds essentially to the numerical solution of a system of nonlinear differential equations, which is both difficult and very time-consuming. At the moment this parameter estimation step still poses a significant challenge and is the bottleneck in the whole modeling process. Various attempts to improve the currently available techniques have already been made [13-17] and while they were somewhat successful in simplifying this crucial step in the modeling process and improving the results, a method which delivers satisfactory results under a wide range of circumstances and which can be automatized has not been found yet. Presently parameter estimation of an S-system still requires the knowledge and experience of a specialist and even then success is not guaranteed.

In this work two novel methods that aim to simplify the parameter estimation task for S-systems are combined. First, the raw time series data is smoothed using the autosmooth algorithm [1], which also generates derivatives of the time series. The smoothed time series as well as their derivatives are then used as input data for alternating regression [2], which estimates the corresponding S-system parameters. Various scenarios using synthetic data are analyzed in order to judge the quality of the results generated by the combination of autosmooth and alternating regression and in order to find out whether one can expect this combination of methods to be suited to handle real-life experimental data.

Systems and Methods

The S-System modeling framework

The S-system framework within BST is ideally suited for the modeling of biochemical pathways. BST is based on linearization of a Taylor series in a logarithmic coordinate system and leads to a non-linear power-law approximation in Cartesian space [13]. The generic form of an S-system is the following:

$$\dot{X}_i = \alpha_i \prod_{j=1}^{n} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} X_j^{h_{ij}}$$

where X_i denotes the concentration of metabolite *i*. Its change over time $\dot{X}_i$ is expressed as the difference between a production and a degradation term. These two terms are products of power-law functions and contain two different types of parameters: the non-negative rate constants *α* and *β* and the kinetic orders *g* and *h*, which can be directly interpreted as the kinetic orders of the corresponding chemical reactions. Therefore, the structure of the network can be inferred in a straightforward fashion if the parameter values of the S-system are known. The values of the kinetic orders typically lie between -3 and 3. The use of S-systems for biochemical modeling has been suggested many times before [7-12], so no detailed discussion of them is needed here.

Decoupling

For *n* metabolites the S-system formulation consists of *n* coupled differential equations, whose parameters we want to estimate. If we know the concentrations X_i and the slopes of the concentrations S_i for each metabolite *i* at *N* points in time t_k, we can instead reformulate the problem as a set of *n* x *N* non-linear algebraic equations [13]:

$$S_i(t_k) \approx \alpha_i \prod_{j=1}^{n} X_j^{g_{ij}}(t_k) - \beta_i \prod_{j=1}^{n} X_j^{h_{ij}}(t_k)$$

Parameter estimation for the coupled differential equations involves the computationally expensive numerical integration, whereas this step is not necessary for the analysis of the algebraic equations. We do need to estimate the slopes S_i however, and as we will see, this step is a crucial one, especially when we are dealing with data that is not noise-free since the noise tends to be magnified in the estimation of the slopes.

Smoothing

Before starting with the actual task of parameter estimation, we have to pre-process the input data. The goal is to use a filter for the data that separates the noise from the signal, and that allows us to estimate the slopes S_i with great precision. In [1] Vilela *et al.* are proposing a new smoothing algorithm that is based on Eilers' extension [18] of the Whittaker filter [19]. This filter uses an

information-theoretic error function [20] and is able to deal with time-varying error structures by automatically splitting the input data into different domains or windows if this is deemed necessary.
The Whittaker filter fits a smooth time series *z* to a given series *y* of noisy data points by minimizing the following function:

$$Q = \sum_{i=1}^{N} (y_i - z_i)^2 + \lambda \sum_{i=2}^{N} (\Delta z_i)^2$$

The two additive components quantify the fidelity of the smoothed output to the original data $(y_i - z_i)$ and the smoothness Δz_i of the output. The result is dependent on two parameters: the order of the filter, *d*, and the weighing of its residuals *λ*. The order *d* is an implicit parameter that determines the definition of Δz_i by defining the number of data points that are taken into account for the calculation of this measure of smoothness.
Eilers [18] approach of using cross-validation to optimize the parameters *λ* and *d* is extended further by Vilela *et al.* [1] who are following an approach based on information theory. They optimize *λ* and *d* by minimizing Renyi's second order entropy of the cross-validation error. Additionally, they do not assume that the noise structure of the signal is invariant, because this assumption often is not true for biological time series such as those of metabolic profiles. Therefore, the autosmooth algorithm includes a process for segmenting the time series into windows with the same noise behavior. They divide the signal into increasingly smaller windows, each with its own set of parameters *λ* and *d*, until the smaller windows result in no gain according to an information theoretic cost function anymore. The authors also give a closed form solution for the derivative of the smoothed signal.
The whole autosmooth algorithm has been implemented in Matlab by Vilela *et al.* and is available for download at http://www.bioinformaticstation.org where a standalone of the same code can also be found. This package will be used for the smoothing of the input data as well as the estimation of the slopes in the remainder of this work.

Alternating Regression

The results of the pre-processing with the autosmooth filter will then be used as input data for alternating regression, which performs the actual parameter estimation. Alternating regression has been proposed by Chou *et al.* in [2] where a detailed description and evaluation of the algorithm can be found. Its name derives from the fact that the algorithm cycles between two phases of multiple linear regression, alternating between estimating the parameters of the production term and the degradation term of a single S-system equation at a time. The main advantage of using alternating regression is that it is many times faster than any algorithm that directly estimates systems of nonlinear differential equations simultaneously.

The general procedure of performing alternating regression on the *i*th equation of an S-system is as follows:

- The matrices L_p and L_d, containing the regressors of the production term and the degradation term of X_i, are created:

$$L_p = \begin{bmatrix} 1 & \ln(X_1(t_1)) & \cdots & \ln(X_j(t_1)) & \cdots & \ln(X_n(t_1)) \\ 1 & \ln(X_1(t_2)) & \cdots & \ln(X_j(t_2)) & \cdots & \ln(X_n(t_2)) \\ 1 & \vdots & \cdots & \vdots & \cdots & \vdots \\ 1 & \ln(X_1(t_k)) & \cdots & \ln(X_j(t_k)) & \cdots & \ln(X_n(t_k)) \\ 1 & \vdots & \cdots & \vdots & \cdots & \vdots \\ 1 & \ln(X_1(t_N)) & \cdots & \ln(X_j(t_N)) & \cdots & \ln(X_n(t_N)) \end{bmatrix}$$

Only those X_j that are known to affect the production term (those with a non-zero kinetic order), are included in L_p. L_d is defined analogously, containing only those X_j that affect the degradation term.

- The following two matrices are calculated:

$$C_p = \left(L_p^T L_p\right)^{-1} L_p^T$$
$$C_d = \left(L_d^T L_d\right)^{-1} L_d^T$$

- Select initial estimations for β_i and h_{ij}, the parameters of the degradation term of the equation.
- Start the iteration.
- Compute the N-dimensional vector y_d:

$$y_d = \ln\left(S_i(t_k) + \beta_i \prod_{j=1}^{n} X_j^{h_{ij}}(t_k) \right) \qquad (k = 1,2,\ldots,N)$$

- Estimate the parameters of the production term by calculating vector b_p:

$$b_p = \left[\hat{\alpha}_i, \hat{g}_{ij}(j = 1, 2, \ldots, n)\right]' = C_p y_d$$

- Perform the analogous regression for the degradation side using the parameter estimations determined in the previous step:

$$y_p = \ln\left(\alpha_i \prod_{j=1}^{n} X_j^{g_{ij}}(t_k) - S_i(t_k) \right) \qquad (k = 1,2,\ldots,N)$$

$$b_d = \left[\hat{\beta}_i, \hat{h}_{ij}(j = 1, 2, \ldots, n)\right]' = C_p y_d$$

The components of b_d will be used as estimations for the parameters of the degradation term in the next iteration.

- Calculate the logarithm of the sum of squared errors:

$$\ln(SSE) = \ln\left(\sum_{k=1}^{N} \left(y_d(k) - L_p b_p(k)\right)^2 + \sum_{k=1}^{N} \left(y_p(k) - L_d b_d(k)\right)^2 \right)$$

- Continue the iteration until the termination criteria (ln(SSE) is less than a specified value or number of iterations is greater than a specified value) are satisfied.

The result of alternating regression are the vectors b_p and b_d, which contain the estimated parameter values of the production term (α_i and g_{ij}) and the degradation term (β_i and h_{ij}) respectively. Taken together they contain a set of all parameters for one S-system equation. After performing alternating regression for all equations of an S-system, one therefore has estimations for all the parameters of the S-system.

Results

In order to analyze the performance of the combination of autosmooth and alternating regression for the estimation of S-system parameters, synthetic time course data of a small system is being created. This time course data is then used as input data for autosmooth and alternating regression and their output, estimations for the parameters of the S-system, is compared to the true parameter values. In addition the system is simulated using the estimated parameters and the resulting time courses compared to the true ones. Several combinations of time course data with and without noise are tested and some attempts to adapt alternating regression in a way that improves the quality of the results are made.

The example network

An example network that is representative of a small biochemical network and that has already appeared in other publications [2, 13, 15], is used to generate the synthetic time series data. The example network consists of only four metabolites and as many differential equations, but it contains all the relevant features of a bigger network, so that it is suitable for testing the parameter estimation methods. The equations and initial conditions are:

$$\begin{aligned}
\dot{X}_1 &= 12X_3^{-0.8} - 10X_1^{0.5} & X_1(t_0) &= 1.4 \\
\dot{X}_2 &= 8X_1^{0.5} - 3X_2^{0.75} & X_2(t_0) &= 2.7 \\
\dot{X}_3 &= 3X_2^{0.75} - 5X_3^{0.5}X_4^{0.2} & X_3(t_0) &= 1.2 \\
\dot{X}_4 &= 2X_1^{0.5} - 6X_4^{0.8} & X_4(t_0) &= 0.4
\end{aligned} \qquad \text{(Eq. 1)}$$

The branched pathway qualitatively representing these equations is shown in Figure 1. Material flow is displayed as straight arrows, regulative influences as curved arrows. The figure shows that *X₁* is a precursor of both *X₂* and *X₄*, *X₂* is a precursor of *X₃*, *X₄* activates the degradation of *X₃* and *X₃* inhibits the formation of *X₁*.

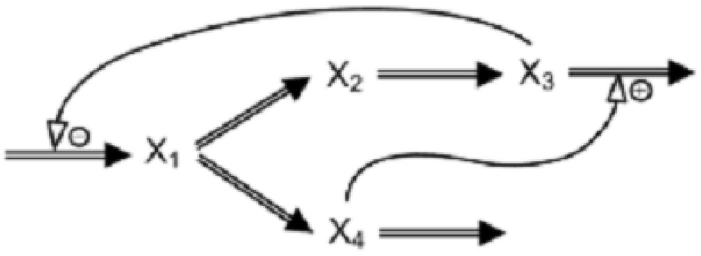

Figure 1 – Branched pathway with four variables and two regulatory signals

Figure 2 shows the metabolite concentrations and their slopes over time.

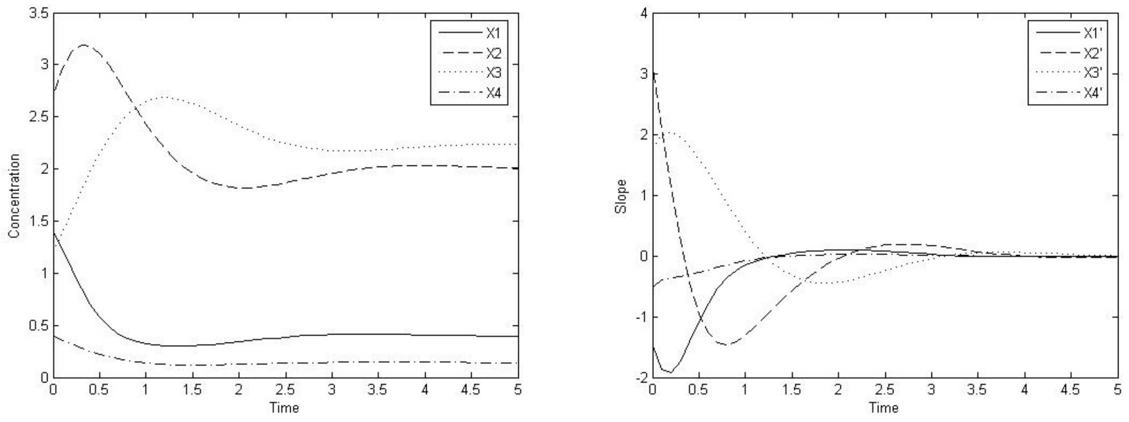

Figure 2 – Time courses and slopes of the four variables of the system in Figure 1

This time course data is obtained by simulating the network with the “Power Law Analysis and Simulation” (PLAS) software package [21] or with Matlab’s `ode45` function that solves initial value problems for ordinary differential equations (ODEs). The metabolite concentrations are being generated for 51 time points, which are equally spaced over the interval [0;5].

Adding noise and filtering

To the noise-free time series data generated in the previous step, varying amounts of randomly distributed Gaussian noise are added using Matlab’s `normrnd` function. Then these noisy time series are filtered using the autosmooth package [1] by Vilela *et al.* Both a stand-alone version of the algorithm as well as a set of Matlab functions are available. They can be downloaded from http://www.bioinformaticstation.org/.

Performing alternating regression

The smoothed data that includes estimated slopes is then used as input data for alternating regression. Alternating regression is performed within Matlab, the functions that are used can be found in the Appendix and at http://kdbio.inesc-id.pt/~svinga/dynamo/beyer/ . These functions are not optimized for speed in any way, so it should be possible to accelerate the calculations substantially. It is assumed that the network topology is already known, which is the same as knowing which parameters in the S-system equations are zero. As a consequence the number of regressors can be reduced by using only those parameters that are known to be non-zero.

Alternating regression generally performs the parameter estimation for the equations of the S-system independently of each other and one equation at a time. This means that mass conservation considerations, which cause dependency between parameters of different equations of the S-system, are not taken into account. As a consequence the parameter values returned by alternating regression may violate the laws of mass conservation. The implementation of alternating regression used for the results in this report does not change this behavior, but it performs the regression of all equations simultaneously, so that the error function can sum the error over all equations and that the parameters of all equations are estimated using the same number of iterations. The termination condition used is two-fold: alternating regression is aborted if the logarithm of the sum of squared errors is less than -15 or after 100000 iterations.

The output of the alternating regression algorithm is a set of values for the parameters of the S-system. These values are then on the one hand compared to the known true values and on the other hand used to simulate the network and the resulting time series compared to the true time series.

The performance

Case 1: no noise, true derivatives

As a first test, alternating regression is being performed on time series data without noise. Moreover, the true slopes are calculated algebraically according to Eq. 1, so the autosmooth algorithm is not needed in this case at all. The results obtained converge to the theoretical values and alternating regression performs fine in this case.

Case 2: noisy time series, true derivatives

In the second case Gaussian noise is added to the time series data. $\sigma(X_1)$=0.04, $\sigma(X_2)$=0.2, $\sigma(X_3)$=0.22 and $\sigma(X_4)$=0.015, which corresponds to a σ of about 10% of the steady state concentration of the respective metabolite. Again the derivatives are calculated according to Eq. 1. The input data is visualized in Figure 3.

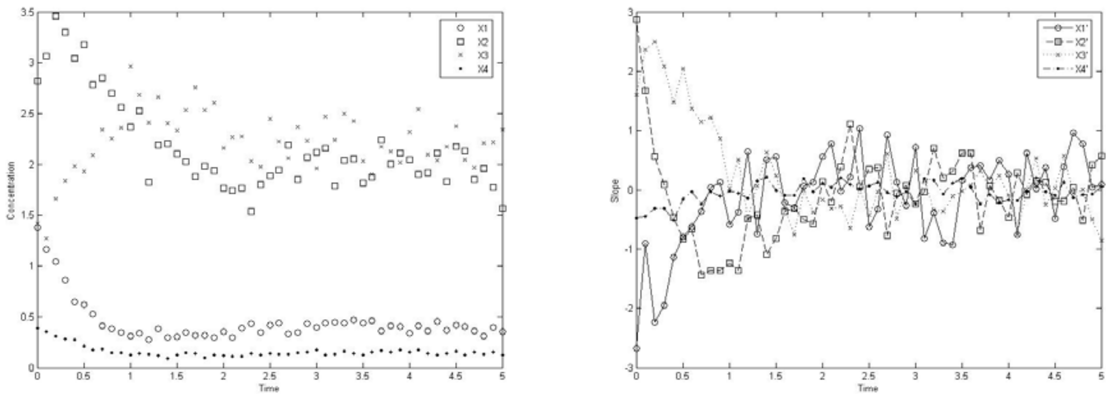

Figure 3 – Case 2 input data: Noisy time course data and slopes calculated from the true S-system equations

There is still no smoothing being performed and again alternating regression is able to calculate the true parameter values.

Case 3: no noise, slopes via autosmooth

In the inverse case to case 2 there is no noise being added to the time series, but the slopes are not anymore calculated according to Eq. 1. This time the autosmooth algorithm is being used to estimate the slopes. In Figure 4 we can see that the difference between the true slopes and the slopes estimated with autosmooth is minimal.

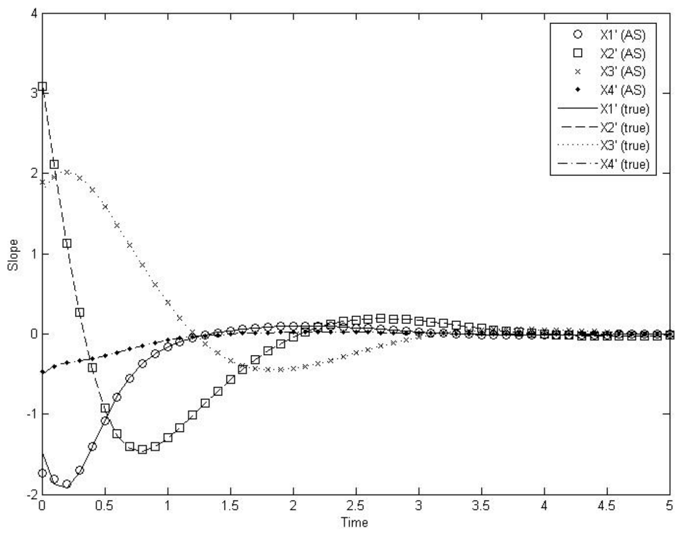

Figure 4 – Comparison of the true slopes with the slopes calculated by autosmooth (no noise)

But even after these minimal changes in the input, the results of alternating regression are already different and look like this:

$$\dot{X}_1 = 10.93 X_3^{-0.99} - 8.66 X_1^{0.61}$$
$$\dot{X}_2 = 8.23 X_1^{0.48} - 3.24 X_2^{0.71}$$
$$\dot{X}_3 = 5.34 X_2^{0.47} - 6.66 X_3^{0.25} X_4^{0.05}$$
$$\dot{X}_4 = 2.00 X_1^{0.52} - 6.01 X_4^{0.82}$$

Even though the numerical parameter values are different, simulating the system with these values results in time courses that are almost indistinguishable from the true time courses (Figure 5).

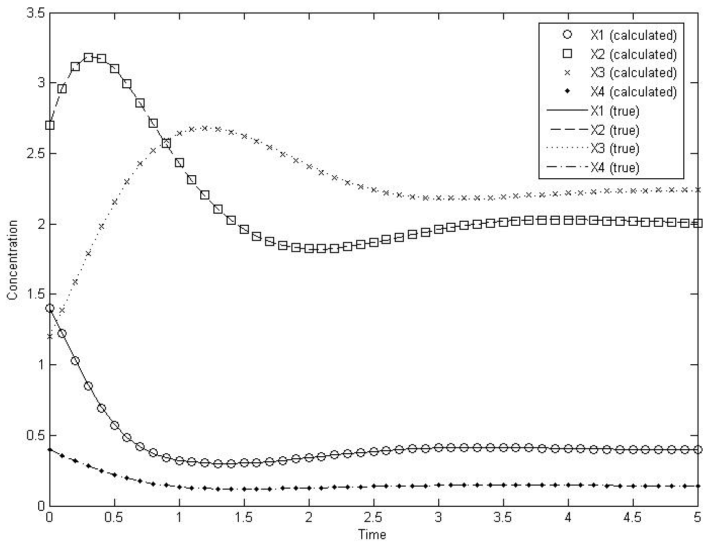

Figure 5 - Comparison of the calculated time courses of case 3 with the true time courses

This phenomenon is typical for S-systems and a result of the fact that there are more parameters than variables (17 versus 4 in our network), so that more than one combination of parameters can explain the same kind of behavior.

Case 4: noisy function values, slopes via autosmooth

In this case, Gaussian noise that corresponds to 5-10% of the steady state value of the respective variable ($\sigma(X_1)$=0.04, $\sigma(X_2)$=0.1, $\sigma(X_3)$=0.11 and $\sigma(X_4)$=0.015), is added to the time series. The autosmooth algorithm is used only for the estimation of the slopes which are used as input data for alternating regression together with the unfiltered raw data for X_1 to X_4. Figure 6 shows both input data and true data.

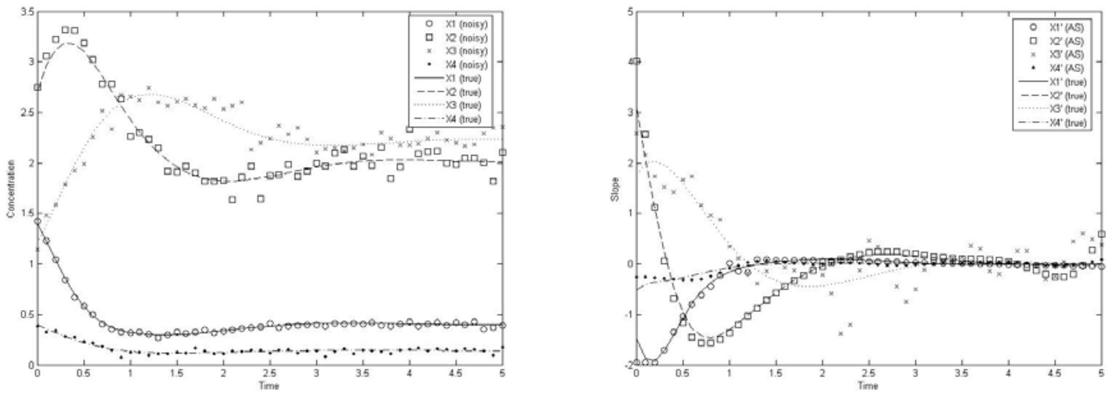

Figure 6 - Case 4 input data: Noisy time course data and slopes calculated with autosmooth

The parameter estimations calculated by alternating regression now consist of complex values.

$$\dot{X}_1 = (1.21 + 76.85i)X_3^{0.03i} - (1.75 + 76.86i)X_1^{-0.03i}$$
$$\dot{X}_2 = (4.13 + 0.04i)X_1^{1.12-0.87i} - (-0.14 + 0.45i)X_2^{1.63-1.54i}$$
$$\dot{X}_3 = (-1.95 + 180.95i)X_2^{-0.02i} - (-1.31 + 180.97i)X_3^{-0.01i}X_4^{0}$$
$$\dot{X}_4 = (-0.16 + 37.21i)X_1^{0} - (0.23 + 37.21i)X_4^{0}$$

In some cases where we get complex results it is possible to get rid of them by running alternating regression with different starting estimations for the parameters in the first iteration. In this case though, starting estimations resulting in real results were not found. Surprisingly, simulating the network which includes complex parameters results in time series, where the imaginary component is very small and which are not very different from the true time courses (Figure 7) if the imaginary part is disregarded.

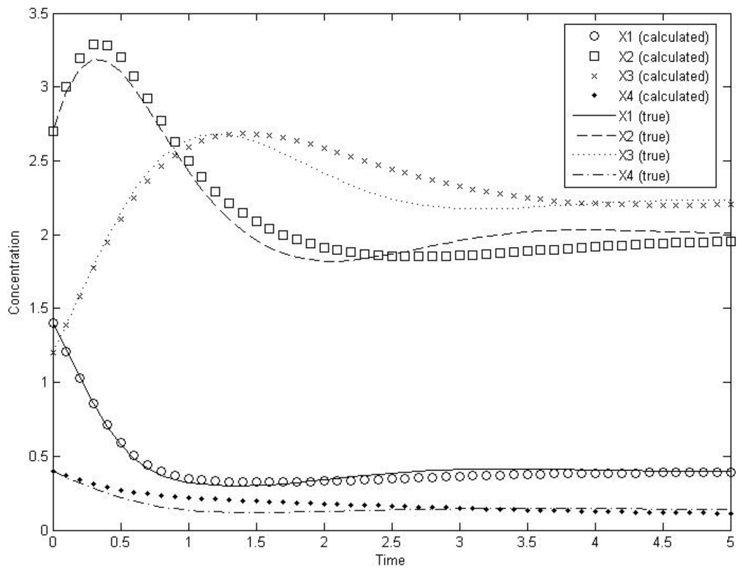

Figure 7 – Comparison of the calculated time courses of case 4 with the true time courses

Case 5: noisy data, function values and derivatives via autosmooth

In this case, instead of passing the raw noisy function values to the alternating regression algorithm, autosmooth is being used for both smoothing the raw data and estimating the slopes. The input data for the autosmooth algorithm is presented in Figure 8.

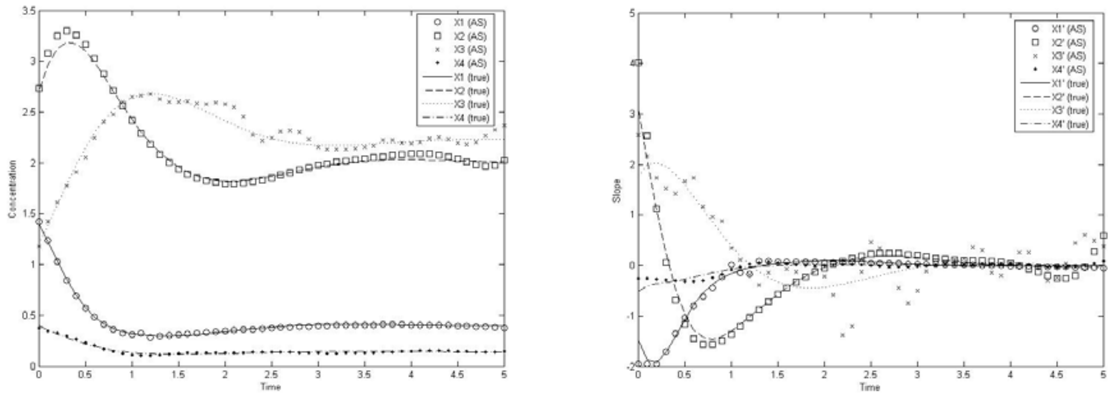

Figure 8 - Case 5 input data: Noisy and autosmoothed time course data and slopes calculated with autosmooth

Again we get complex results that look very similar to those of case 4, but this time the overall mean square error is a little bit smaller.

$$\dot{X}_1 = (2.07 + 61.57i)X_3^{0.06i} - (2.17 + 61.61i)X_1^{-0.05i}$$
$$\dot{X}_2 = (5.33 + 0.35i)X_1^{1.11-0.53i} - (0.21 + 0.54i)X_2^{1.70-0.94i}$$
$$\dot{X}_3 = (-1.70 + 165.64i)X_2^{-0.02i} - (-1.80 + 165.63i)X_3^0 X_4^0$$
$$\dot{X}_4 = (-0.19 + 31.86i)X_1^{-0.01i} - (1.15 + 31.85i)X_4^{0.02i}$$

Figure 9 compares the simulated time courses with the true time courses.

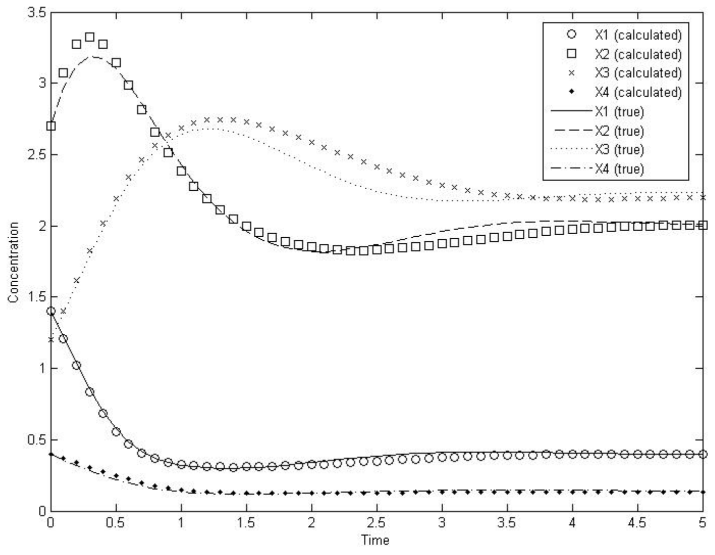

Figure 9 - Comparison of the calculated time courses of case 5 with the true time courses

Discussion

After analyzing how autosmooth and alternating regression perform in each scenario, a comparison of all cases was made. Table 1 summarizes the results obtained in each case:

Data Points	**Slopes**	**Alternating Regression Results**	
		Parameter Values	**Curve Fit**
Exact	Exact	Converge to true values	Perfect
Noisy	Exact	Converge to true values (as long as σ is small enough)	Perfect
Exact	Autosmooth	Converge to different values	Perfect
Noisy	Autosmooth	Converge to complex values	Reasonable
Autosmooth	Autosmooth	Converge to complex values	Reasonable

The results show that S-system parameter estimation with alternating regression is sensitive to noisy data. The noise in the time courses themselves does not play as big a part as the noise in the estimated slopes. For those, accurate estimations are essential. The quality of the results of alternating regression is directly correlated to the quality of the estimations of the slopes.

Dealing with complex parameter values

Since complex rate constants and kinetic orders make no sense from a biochemical point of view, it would be desirable to get rid of them and have only real results. Different strategies can be employed in order to reach this goal.

In the first iteration of the alternating regression algorithm one has to start with initial guesses for the parameters of either the production or the degradation term of the equation. The influence of these starting values on the result should not be disregarded easily. Chou *et al.* already noted in [2] that the convergence of the algorithm depends on the starting values. Because alternating regression is fast, it is possible to try different initial guesses until one of them results in non-complex parameter values. While this strategy is successful in some cases, in the more complicated ones no starting values that resulted in real results were found.

Another option is to disregard any imaginary component of a parameter estimation in the alternating regression algorithm as soon as it appears and to continue the calculation only with the real component of that parameter. The problem is that by ignoring the imaginary part, the algorithm does not converge anymore and no useful results are obtained.

A third possible way to deal with the problem of complex results is to watch the alternating regression algorithm more closely. The origin of the complex results is a step where the logarithm of a sum/difference is taken. This is done for every observation of X_i separately and the results are arranged in a column vector. Now it is possible to check in every iteration of the algorithm whether this vector contains only real numbers and if this is not the case one can simply remove (for this single iteration only) those observations that cause complexity and continue the calculations for this iteration as if we only had those observations that cause no complex results. Unfortunately again, this adaptation of the algorithm causes the loss of convergence to specific values and no results are obtained.

Therefore the problem of complex results remains. If varying the starting parameter estimations does not lead to real results, getting non-complex results is not possible at all with the methods described above. And since complex parameter values make no sense biochemically, a way to find real parameter values is highly desirable.

Using the parameter estimates for predictions

Obviously, if alternating regression returns the true parameter values as results, simulating the resulting system under different starting conditions will show the behavior of the true system under these conditions perfectly. But what happens in those cases where alternating regression calculates parameter values that are not exactly true? Are these systems still able to predict the behavior under different conditions? Figure 10 compares the time courses obtained when simulating the true system with those obtained from simulating the equations using the solution of case 5 under different initial conditions:

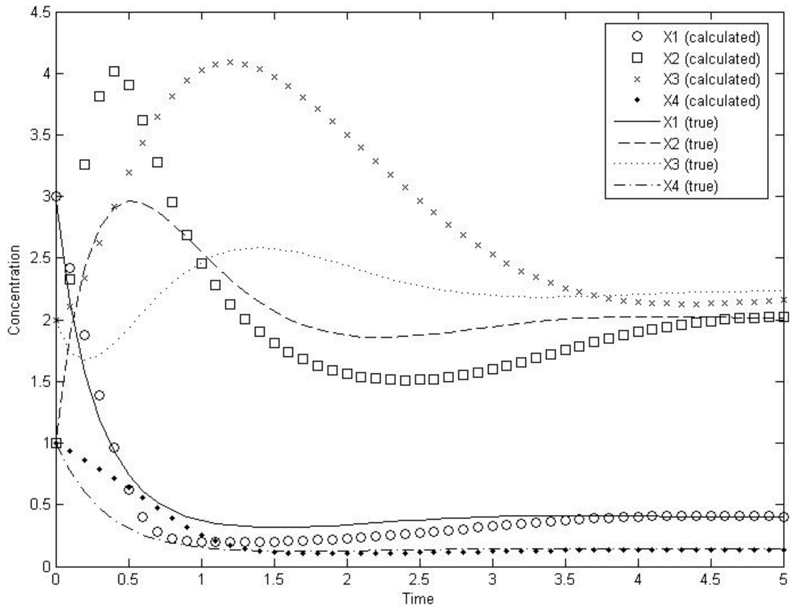

Figure 10 – Comparison of the calculated time courses with the true time courses when choosing different initial conditions

As can easily be seen, the predictive power of the parameter set returned by alternating regression is severely limited. Maybe it can be used to make some qualitative statements about the behavior of the system, but it is far too inaccurate for any quantitative statements. This fact confirms that more accurate (and non-complex) parameter estimations are needed to obtain a valid model, which fulfills its purpose of being able to predict the behavior of the system under conditions that have not been tested yet.

Mass conservation and other restrictions

Because alternating regression estimates the parameters of each equation of the S-system separately, it does not take the law of mass conservation into account. As opposed to the physical reality, in alternating regression there are no restrictions for the parameters of metabolites which are part of a precursor-product relationship. In our example network (Figure 1) X_2 and X_3 form such a relationship. Consequently the degradation term of X_2 is exactly the same as the production term of X_3. But because alternating regression performs the calculations for each equation separately, there is no straightforward way to implement these kinds of restrictions that concern more than one equation of the S-system into the algorithm. It is possible though, to run exactly one iteration for all equations of the S-system before starting the next iteration. In between the individual iterations one can compare the values of those parameters that should have equal values and adjust them. What has been done in this work is the following: if the parameter p_1 should be equal to parameter p_2 according to the (known) network structure, p_1 and p_2 were both set to the average of p_1 and p_2 after each iteration of alternating regression.

Another kind of restriction that is easier to implement, is setting minimum and maximum parameter values. Any kinetic orders typically lie between -1 and 2, so one can set these values as borders for the parameters. If at any time during the alternating regression a kinetic order is smaller than -1 (greater than 2), it is set to -1 (2).

Experimenting with different combinations of restrictions within the alternating regression algorithm yields unexpected results. Some combinations improve the convergence of the algorithm, so that it takes less iterations to reach a stable solution, while others impair the convergence. Adding a single additional restriction may have positive or negative effects on the convergence of the algorithm and one cannot tell beforehand what will happen, so that trial and error is the only strategy to explore the use of restrictions. Additionally, in those cases where restrictions were added successfully, only the speed of convergence improved and not the quality of the results. Because there are a lot of different combination of restrictions that can be employed and because there are different ways to implement them, no final judgment on the usefulness of restrictions can be given here. The feeling that adding some kind of restrictions can significantly improve the results remains, but unfortunately such a combination was not found.

Alternative approaches for parameter estimation

Of course there exist other ways to estimate S-system parameters. Due to the high number of parameters any S-system has, the main strategy has been the use of massive computer power. In [14] a cluster of 1040 Pentium III 933MHz processors was used to estimate the parameters of a differential equation model with five variables. The data was noise-free data and a genetic algorithm was applied to solve the equations. Even though the time required for one algorithmic loop was about 10 hours, this meant an improvement in optimization speed and convergence rate and also increased the number of predictable parameters.

In [13] Voit and Almeida have proposed the decoupling of the system of differential equations of an S-system into decoupled algebraic equations. They estimated the slopes by using a universal function of time, in their case produced by artificial neural networks. The first derivative of these universal functions can be determined symbolically, but they also produce artifacts in higher order derivatives, which led the researchers to pursue other methods for data smoothing and estimation of slopes. The resulting methods are the ones analyzed in this report.

Tucker and Moulton are following a deterministic approach to parameter estimation based on interval analysis in [15]. They test their method on the same S-systems used in [13] and [14]. Their algorithm has a run time of several hours for a system of 4 or 5 equations and gives accurate parameter estimations, but is not yet able to handle noisy time series data. Their improved algorithm presented in [16] does not have this limitation. It uses Newton-flow analysis to identify a one-dimensional attractor containing the true parameter values for a wide range of examples, therefore reducing the parameter search space and allowing parameter estimation for noisy data.

In [17] Gennemark and Wedeling proposed algorithms for both structure identification and parameter estimation. The model structure is built incrementally with a heuristic search algorithm and the parameters are estimated one equation at a time. They show that they can get meaningful results using only small amounts of time course data and without using supercomputers.

Conclusion

The inverse problem of modeling a biochemical system from metabolic time course data is challenging and computationally expensive. After the experimental data is obtained, the two main steps for creating a suitable S-system model are smoothing of the data (including estimation of the slopes) and parameter estimation. These two steps constitute the bottleneck in metabolic modeling nowadays and a lot of effort is put into automation of this work. There is still no “recipe” for modeling and most successful models were created with a lot of manual intervention and trial and error. Nevertheless progress towards automation is made.

Reviewing autosmooth and alternating regression, two recently published methods for data filtering and parameter estimation and combining them for the task of creating a model, shows both the power and the limitation of these methods. They perform very well in a noise-free environment and show their potential. But when tested in a more realistic case that includes noise (but which is still simple compared to real problems), they already exhibit some difficulties. They are fast compared to other methods which directly estimate systems of differential equations and also deliver results that allow one to reconstruct the behavior of the biochemical system, but when trying to predict the system’s behavior under untested conditions, the quality of these predictions is limited.

Combining autosmooth and alternating regression is a good way to estimate the parameters of an S-system, but is still far from being a perfect solution. There still has to be made a lot of progress until the goal of automating model building from metabolic data is reached.

References

1. Vilela M, Borges C, Vinga S, Vasconcelos AT, Santos H, Voit E, Almeida JS: **Automated smoother for numerical decoupling of dynamic models.** submitted.

2. Chou I, Martens H, Voit EO: **Parameter estimation in biochemical systems models with alternating regression.** *Theor Biol Med Model* 2006, 3:25.

3. Neves AR, Ramos A, Nunes MC, Kleerebezem M, Hugenholtz J, de Vos WM, Almeida J, Santos H: **In vivo nuclear magnetic resonance studies of glycolytic kinetics in Lactococcus lactis.** *Biotechnol Bioeng* 1999, 64:200-212

4. Voit E, Neves AR, Santos H: **The intricate side of systems biology.** *PNAS* 2006, 103:9452-9457.

5. Voit EO, Almeida J, Marino S, Lall R, Goel G, Neves AR, Santos H: **Regulation of glycolysis in *Lactococcus lactis*: an unfinished systems biological case study.** *IEE Proc Systems Biol* 2006, 153:286-298.

6. Piedade R, Vinga S, de Miranda Lemos JML, Freitas AT: **Exploring metabolic and genetic control of the glycolytic pathway of Lactococcus lactis. (Extended abstract).** ICMSB'06 International Conference on Molecular Systems Biology 2006.

7. Savageau MA**: Biochemical systems analysis. I. Some mathematical properties of the rate law for the component enzymatic reactions.** *J Theor Biol* 1969, 25:365-369.

8. Savageau MA: **Biochemical systems analysis. II. The steady-state solutions for an n-pool system using a power-law approximation.** *J Theor Biol* 1969, 25:370-379.

9. Savageau MA: **Biochemical systems analysis. III. Dynamic solutions using a power-law approximations.** *J Theor Biol* 1970, 26:215-226.

10. Savageau MA: **Biochemical systems analysis: A study of function and design in molecular biology.** Reading, MA, Addison-Wesley; 1976:xvii, 379.

11. Voit EO, Savageau MA: **Accuracy of Alternative Representations for Integrated Biochemical Systems.** *Biochemistry* 1987, 26:6869-6880.

12. Voit EO: **Computational analysis of biochemical systems: a practical guide for biochemists and molecular biologists.** Cambridge, U.K., Cambridge University Press; 2000:xii, 531.

13. Voit EO, Almeida J: **Decoupling dynamical systems for pathway identification from metabolic profiles.** *Bioinformatics* 2004, 20:1670-1681.

14. Kikuchi S, Tominaga D, Arita M, Takahashi K, Tomita M: **Dynamic modeling of genetic networks using genetic algorithm and S-system.** *Bioinformatics* 2003, 19:643-650.

15. Tucker W, Moulton V: **Parameter Reconstruction for Biochemical Networks Using Interval Analysis.** *Reliable Computing* 2006, 12:389-402.

16. Kutalik Z, Tucker W, Moulton V: **S-system parameter estimation for noisy metabolic profiles using Newton-flow analysis.** *IET Sys Biol* 2007, 1:174-180.

17. Gennemark P, Wedelin D: **Efficient algorithms for ordinary differential equation model identification of biological systems.** *IET Sys Biol* 2007, 1:120-129.

18. Eilers PH: **A perfect smoother.** *Anal Chem* 2003, 75:3631-3636.

19. Whittaker ET: **On a new method of graduation.** *Proc. Edinburgh Math. Soc. 1923*, 41:63-75

20. Principe J, Xu D, Fisher J: **Information-Theoretic Learning.** In *Advances in unsupervised adaptive filtering.* Wiley 1999.

21. Ferreira A: **Power Law Analysis and Simulation.** http://www.dqb.fc.ul.pt/docentes/aferreira/plas.html. 2005

Appendix

These are the Matlab functions and scripts that I have written during the work on this report. You are free to use and modify these functions in any way you like.

ToyNetwork.m

Contains the differential equations of the network and is only called by other functions.

```
% Describes a simple set of S-System equations
%
% Input:
%   t:      time
%   x:      initial values of X1, X2, X3 and X4
%   param:  vector of the parameter values (1 to 8 for production, 9-17 for
%           degradation)
%
% Wolfgang Beyer, 2007

function dx = ToyNetwork3(t,x,param)
dx = zeros(4,1);

p=param(1:8);
d=param(9:17);

dx(1) = p(1)*x(3)^p(2) - d(1)*x(1)^d(2);
dx(2) = p(3)*x(1)^p(4) - d(3)*x(2)^d(4);
dx(3) = p(5)*x(2)^p(6) - d(5)*x(3)^d(6)*x(4)^d(7);
dx(4) = p(7)*x(1)^p(8) - d(8)*x(4)^d(9);
```

Generate_TS.m

Generates time series data of the toy network and adds Gaussian noise.

```
% Generates time series data for the S-system described in ToyNetwork.m
%
% Output:
% TS:     time series data (columns: t, X1, X2, X3, X4)
%
% Wolfgang Beyer, 2007

% Simulate the differential equations
% [0 5]          time interval
% [1.4 ... 0.4] initial values for X1, X2, X3, X4
% [12 ... .8]   parameter values of the S-System
testSoln=ode45(@ToyNetwork,[0 5],[1.4; 2.7; 1.2; 0.4],[],[12 -.8 8 .5 3 .75 2 .5 10
.5 3 .75 5 .5 .2 6 .8]);
new_t=linspace(0,5,51);
new_x=deval(testSoln,new_t);
TS=[new_t' new_x'];

% delete unused variables
clear new_t;
clear new_x;
clear testSoln;

% add Gaussian noise
TS(:,2)=TS(:,2)+normrnd(0,.04,51,1);
TS(:,3)=TS(:,3)+normrnd(0,.1,51,1);
TS(:,4)=TS(:,4)+normrnd(0,.11,51,1);
TS(:,5)=TS(:,5)+normrnd(0,.015,51,1);

% plot the generated data
plot(TS(:,1),[TS(:,2) TS(:,3) TS(:,4) TS(:,5)])
legend('X1','X2','X3','X4');
```

AR_X1.m

Performs alternating regression on equation 1 of the network.

```
% Performs alernating regression on equation 1 of the S-System defined in
ToyNetwork.m
%
% alternating regression is performed as published in
% Chou I, Martens H, Voit EO: Parameter estimation in biochemical systems models
with alternating regression. Theor Biol Med Model 2006, 3:25.
%
% Input:
%   X1:         time course data of X1
%   X3:         time course data of X3
%   Si:         (estimated) slope of X1
%   bd:         initial estimations for the parameters of the degradation term
% Output:
%   Production:     estimations for the parameters of the production term
%   Degradation:    estimations for the parameters of the degradation term
%   SSE:            sum of squared errors
%   Iterations:     number of iterations run
%
% Wolfgang Beyer, 2007

function [Production, Degradation, SSE, Iterations] = X1C(X1, X3, Si, bd)

%Generate matrix of logarithms of regressors
TSLength=length(X1);
Lp=[ones(TSLength,1) log(X3)];
Ld=[ones(TSLength,1) log(X1)];

%Compute matrices Cp and Cd
Cp=inv(Lp'*Lp)*Lp';
Cd=inv(Ld'*Ld)*Ld';

%Output of initial estimates for the parameters of the degradation term
Degradation_Estimates=bd'

%Iterate the actual regression
SSE=1;
Iterations=0;
while (SSE>-15) && (Iterations<100000)

    %Degradation side
    %do regression
    yd=[log(Si+bd(1,1)*X1.^bd(2,1))];
    bp=Cp*yd;
    %calculate error
    SSE=sum((yd-Lp*bp).^2);
    %transform parameter from logarithmic space in to linear space
    bp(1,1)=exp(bp(1,1));

    %Production side
    %do regression
    yp=[log(bp(1,1)*X3.^bp(2,1)-Si)];
    bd=Cd*yp;
    %calculate error
    SSE=log(SSE+sum((yp-Ld*bd).^2));
    %transform parameter from logarithmic space in to linear space
    bd(1,1)=exp(bd(1,1));

    Iterations=Iterations+1;
end

%show theoretical values together with the calculated ones
Production=[[12; -.8] bp];
Degradation=[[10; .5] bd];
```

AR_X2.m

Performs alternating regression on equation 2 of the network.

```
% Performs alernating regression on equation 2 of the S-System defined in
ToyNetwork.m
%
% alternating regression is performed as published in
% Chou I, Martens H, Voit EO: Parameter estimation in biochemical systems models
with alternating regression. Theor Biol Med Model 2006, 3:25.
%
% Input:
%   X1:         time course data of X1
%   X2:         time course data of X2
%   Si:         (estimated) slope of X2
%   bd:         initial estimations for the parameters of the degradation term
% Output:
%   Production:     estimations for the parameters of the production term
%   Degradation:    estimations for the parameters of the degradation term
%   SSE:            sum of squared errors
%   Iterations:     number of iterations run
%
% Wolfgang Beyer, 2007

function [Production, Degradation, SSE, Iterations] = X2C(X1, X2, Si, bd)

%Generate matrix of logarithms of regressors
TSLength=length(X2);
Lp=[ones(TSLength,1) log(X1)];
Ld=[ones(TSLength,1) log(X2)];

%Compute matrices Cp and Cd
Cp=inv(Lp'*Lp)*Lp';
Cd=inv(Ld'*Ld)*Ld';

%Output of initial estimates for the parameters of the degradation term
Degradation_Estimates=bd'

%Iterate the actual regression
SSE=1;
Iterations=0;
while (SSE>-15) && (Iterations<100000)

    %Degradation side
    %do regression
    yd=[log(Si+bd(1,1)*X2.^bd(2,1))];
    bp=Cp*yd;
    %calculate error
    SSE=sum((yd-Lp*bp).^2);
    %transform parameter from logarithmic space in to linear space
    bp(1,1)=exp(bp(1,1));

    %Production side
    %do regression
    yp=[log(bp(1,1)*X1.^bp(2,1)-Si)];
    bd=Cd*yp;
    %calculate error
    SSE=log(SSE+sum((yp-Ld*bd).^2));
    %transform parameter from logarithmic space in to linear space
    bd(1,1)=exp(bd(1,1));

    Iterations=Iterations+1;
end

%show theoretical values together with the calculated ones
Production=[[8;.5] bp];
Degradation=[[3; .75] bd];
```

AR_X3.m

Performs alternating regression on equation 3 of the network.

```
% Performs alernating regression on equation 3 of the S-System defined in
ToyNetwork.m
%
% alternating regression is performed as published in
% Chou I, Martens H, Voit EO: Parameter estimation in biochemical systems models
with alternating regression. Theor Biol Med Model 2006, 3:25.
%
% Input:
%   X2:         time course data of X2
%   X3:         time course data of X3
%   X4:         time course data of X4
%   Si:         (estimated) slope of X3
%   bd:         initial estimations for the parameters of the degradation term
% Output:
%   Production:     estimations for the parameters of the production term
%   Degradation:    estimations for the parameters of the degradation term
%   SSE:            sum of squared errors
%   Iterations:     number of iterations run
%
% Wolfgang Beyer, 2007

function [Production, Degradation, SSE, Iterations] = X3C(X2, X3, X4, Si, bd)

%Generate matrix of logarithms of regressors
TSLength=length(X3);
Lp=[ones(TSLength,1) log(X2)];
Ld=[ones(TSLength,1) log(X3) log(X4)];

%Compute matrices Cp and Cd
Cp=inv(Lp'*Lp)*Lp';
Cd=inv(Ld'*Ld)*Ld';

%Output of initial estimates for the parameters of the degradation term
Degradation_Estimates=bd'

%Iterate the actual regression
SSE=1;
Iterations=0;
while (SSE>-15) && (Iterations<100000)

    %Degradation side
    %do regression
    yd=[log(Si+bd(1,1)*X3.^bd(2,1).*X4.^bd(3,1))];
    bp=Cp*yd;
    %calculate error
    SSE=sum((yd-Lp*bp).^2);
    %transform parameter from logarithmic space in to linear space
    bp(1,1)=exp(bp(1,1));

    %Production side
    %do regression
    yp=[log(bp(1,1)*X2.^bp(2,1)-Si)];
    bd=Cd*yp;
    %calculate error
    SSE=log(SSE+sum((yp-Ld*bd).^2));
    %transform parameter from logarithmic space in to linear space
    bd(1,1)=exp(bd(1,1));

    Iterations=Iterations+1;
end

%show theoretical values together with the calculated ones
Production=[[3;.75] bp];
Degradation=[[5; .5; .2] bd];
```

AR_X4.m

Performs alternating regression on equation 4 of the network.

```
% Performs alernating regression on equation 4 of the S-System defined in
ToyNetwork.m
%
% alternating regression is performed as published in
% Chou I, Martens H, Voit EO: Parameter estimation in biochemical systems models
with alternating regression. Theor Biol Med Model 2006, 3:25.
%
% Input:
%   X1:         time course data of X1
%   X3:         time course data of X4
%   Si:         (estimated) slope of X4
%   bd:         initial estimations for the parameters of the degradation term
% Output:
%   Production:     estimations for the parameters of the production term
%   Degradation:    estimations for the parameters of the degradation term
%   SSE:            sum of squared errors
%   Iterations:     number of iterations run
%
% Wolfgang Beyer, 2007

function [Production, Degradation, SSE, Iterations] = X2C(X1, X4, Si, bd)

%Generate matrix of logarithms of regressors
TSLength=length(X4);
Lp=[ones(TSLength,1) log(X1)];
Ld=[ones(TSLength,1) log(X4)];

%Compute matrices Cp and Cd
Cp=inv(Lp'*Lp)*Lp';
Cd=inv(Ld'*Ld)*Ld';

%Output of initial estimates for the parameters of the degradation term
Degradation_Estimates=bd'

%Iterate the actual regression
SSE=1;
Iterations=0;
while (SSE>-15) && (Iterations<100000)

    %Degradation side
    %do regression
    yd=[log(Si+bd(1,1)*X4.^bd(2,1))];
    bp=Cp*yd;
    %calculate error
    SSE=sum((yd-Lp*bp).^2);
    %transform parameter from logarithmic space in to linear space
    bp(1,1)=exp(bp(1,1));

    %Production side
    %do regression
    yp=[log(bp(1,1)*X1.^bp(2,1)-Si)];
    bd=Cd*yp;
    %calculate error
    SSE=log(SSE+sum((yp-Ld*bd).^2));
    %transform parameter from logarithmic space in to linear space
    bd(1,1)=exp(bd(1,1));

    Iterations=Iterations+1;
end

%show theoretical values together with the calculated ones
Production=[[2;.5] bp];
Degradation=[[6; .8] bd];
```

Call_AR_X1toX4.m

Loads time courses and then calls AR_X1, AR_X2, AR_X3 and AR_X4 one after each other to estimate the parameters of the S-System.

```
% Calls the functions that perform alernating regression on the S-System defined in
ToyNetwork.m
%
% alternating regression is performed as published in
% Chou I, Martens H, Voit EO: Parameter estimation in biochemical systems models
with alternating regression. Theor Biol Med Model 2006, 3:25.
%
% Wolfgang Beyer, 2007

tic

% loads the time series data
load TS_clean.mat
% loads the results of performing autosmooth on the time series data
load TS_clean_AS.mat

%estimated slopes are the ones taken from autosmooth
Si1=result(:,6);
Si2=result(:,7);
Si3=result(:,8);
Si4=result(:,9);

%alternatively one could calculate the true slopes and use them
%uncomment the following lines if you want to use the true slopes

%Si1=[12*TS(:,4).^-.8 - 10*TS(:,2).^.5];
%Si2=[8*TS(:,2).^.5 - 3*TS(:,3).^.75];
%Si3=[3*TS(:,3).^.75 - 5*TS(:,4).^.5.*TS(:,5).^.2];
%Si4=[2*TS(:,2).^.5 - 6*TS(:,5).^.8];

%Set first estimation of degradation term and run alternating regression for X1
bd=[15; 1];
[Prod, Deg, Error, It] = AR_X1(TS(:,2), TS(:,4), Si1, bd)

%Set first estimation of degradation term and run alternating regression for X2
bd=[15; 1];
[Prod, Deg, Error, It] = AR_X2(TS(:,2), TS(:,3), Si2, bd)

%Set first estimation of degradation term and run alternating regression for X3
bd=[15; 1; 1];
[Prod, Deg, Error, It] = AR_X3(TS(:,3), TS(:,4), TS(:,5), Si3, bd)

%Set first estimation of degradation term and run alternating regression for X4
bd=[15; 1];
[Prod, Deg, Error, It] = AR_X4(TS(:,2), TS(:,5), Si4, bd)

%clear unused variables
clear Si1;
clear Si2;
clear Si3;
clear Si4;
clear bd;

toc
```

AR_allX.m

Performs alternating regression on all 4 equations simultaneously.

```
% Performs alernating regression on the whole S-System defined in ToyNetwork.m
%
% alternating regression is performed as published in
% Chou I, Martens H, Voit EO: Parameter estimation in biochemical systems models
with alternating regression. Theor Biol Med Model 2006, 3:25.
%
% Input:
%   X1:         time course data of X1
%   X2:         time course data of X2
%   X3:         time course data of X3
%   X4:         time course data of X4
%   Si1:        (estimated) slope of X1
%   Si2:        (estimated) slope of X2
%   Si3:        (estimated) slope of X3
%   Si4:        (estimated) slope of X4
%   bd1:        initial estimations for the parameters of the degradation term of
X1
%   bd2:        initial estimations for the parameters of the degradation term of
X2
%   bd3:        initial estimations for the parameters of the degradation term of
X3
%   bd4:        initial estimations for the parameters of the degradation term of
X4
%
% Output:
%   Production:     estimations for the parameters of the production term
%   Degradation:    estimations for the parameters of the degradation term
%   SSE:            sum of squared errors
%   Iterations:     number of iterations run
%
% Wolfgang Beyer, 2007

function [Production, Degradation, SSE, Iterations] = X1to4C(X1, X2, X3, X4, Si1,
Si2, Si3, Si4, bd1, bd2, bd3, bd4)

%Generate matrices of logarithms of regressors
TSLength=length(X1);
Lp1=[ones(TSLength,1) log(X3)];
Ld1=[ones(TSLength,1) log(X1)];
Lp2=[ones(TSLength,1) log(X1)];
Ld2=[ones(TSLength,1) log(X2)];
Lp3=[ones(TSLength,1) log(X2)];
Ld3=[ones(TSLength,1) log(X3) log(X4)];
Lp4=[ones(TSLength,1) log(X1)];
Ld4=[ones(TSLength,1) log(X4)];

%Compute matrices Cp and Cd
Cp1=inv(Lp1'*Lp1)*Lp1';
Cp2=inv(Lp2'*Lp2)*Lp2';
Cp3=inv(Lp3'*Lp3)*Lp3';
Cp4=inv(Lp4'*Lp4)*Lp4';
Cd1=inv(Ld1'*Ld1)*Ld1';
Cd2=inv(Ld2'*Ld2)*Ld2';
Cd3=inv(Ld3'*Ld3)*Ld3';
Cd4=inv(Ld4'*Ld4)*Ld4';

%Iterate the actual regression
SSE=1;
Iterations=0;
while (SSE>-15) && (Iterations<100000)

    %Degradation side
    %do regression
    yd1=[log(Si1+bd1(1,1)*X1.^bd1(2,1))];
    yd2=[log(Si2+bd2(1,1)*X2.^bd2(2,1))];
    yd3=[log(Si3+bd3(1,1)*X3.^bd3(2,1).*X4.^bd3(3,1))];
    yd4=[log(Si4+bd4(1,1)*X4.^bd4(2,1))];

    bp1=Cp1*yd1;
    bp2=Cp2*yd2;
    bp3=Cp3*yd3;
    bp4=Cp4*yd4;

    %calculate error
    SSE=sum((yd1-Lp1*bp1).^2+(yd2-Lp2*bp2).^2+(yd3-Lp3*bp3).^2+(yd4-Lp4*bp4).^2);

    %transform parameters from logarithmic space in to linear space
    bp1(1,1)=exp(bp1(1,1));
    bp2(1,1)=exp(bp2(1,1));
    bp3(1,1)=exp(bp3(1,1));
    bp4(1,1)=exp(bp4(1,1));

    %Production side
    %do regression
    yp1=[log(bp1(1,1)*X3.^bp1(2,1)-Si1)];
    yp2=[log(bp2(1,1)*X1.^bp2(2,1)-Si2)];
    yp3=[log(bp3(1,1)*X2.^bp3(2,1)-Si3)];
    yp4=[log(bp4(1,1)*X1.^bp4(2,1)-Si4)];

    bd1=Cd1*yp1;
    bd2=Cd2*yp2;
    bd3=Cd3*yp3;
    bd4=Cd4*yp4;

    %calculate error
    SSE=log(SSE+sum((yp1-Ld1*bd1).^2+(yp2-Ld2*bd2).^2+(yp3-Ld3*bd3).^2+(yp4-
Ld4*bd4).^2));

    %transform parameters from logarithmic space in to linear space
    bd1(1,1)=exp(bd1(1,1));
    bd2(1,1)=exp(bd2(1,1));
    bd3(1,1)=exp(bd3(1,1));
    bd4(1,1)=exp(bd4(1,1));

    Iterations=Iterations+1;
end

%show theoretical values together with the calculated ones
Production=[[12; -.8; 8; .5; 3; .75; 2; .5] [bp1; bp2; bp3; bp4]];
Degradation=[[10; .5; 3; .75; 5; .5; .2; 6; .8] [bd1; bd2; bd3; bd4]];
```

Call_AR_allX.m:

Loads time course data and then calls AR_allX to estimate the parameters of the S-system.

```
% Calls the function that performs alernating regression on the whole S-System
defined in ToyNetwork.m
%
% alternating regression is performed as published in
% Chou I, Martens H, Voit EO: Parameter estimation in biochemical systems models
with alternating regression. Theor Biol Med Model 2006, 3:25.
%
% Wolfgang Beyer, 2007

tic

%load the time series data
load TS_noisy.mat

%load the results of performing autosmooth on the time series data
load TS_noisy_AS.mat

%estimated slopes are the ones taken from autosmooth
Si1=result(:,6);
Si2=result(:,7);
Si3=result(:,8);
Si4=result(:,9);

%alternatively one could calculate the true slopes and use them
%uncomment the following lines if you want to use the true slopes

%Si1=[12*TS(:,4).^-.8 - 10*TS(:,2).^.5];
%Si2=[8*TS(:,2).^.5 - 3*TS(:,3).^.75];
%Si3=[3*TS(:,3).^.75 - 5*TS(:,4).^.5.*TS(:,5).^.2];
%Si4=[2*TS(:,2).^.5 - 6*TS(:,5).^.8];

%Set first estimation of degradation term for each equation
bd1=[15; 1];
bd2=[15; 1];
bd3=[15; 1; 1];
bd4=[15; 1];

%Run alternating regression
[Prod, Deg, Error, It] = AR_allX(TS(:,2), TS(:,3), TS(:,4), TS(:,5), Si1, Si2, Si3,
Si4, bd1, bd2, bd3, bd4)

%simulate the system with the calculated solution
testSoln=ode45(@ToyNetwork,[0 5],[1.4; 2.7; 1.2; 0.4],[],[Prod(:,2); Deg(:,2)]);
new_t=linspace(0,5,51);
new_x=deval(testSoln,new_t);
calc_t=new_t';
calc_x=new_x';

%simulate the true system
testSoln=ode45(@ToyNetwork,[0 5],[1.4; 2.7; 1.2; 0.4],[],[12 -.8 8 .5 3 .75 2 .5 10
.5 3 .75 5 .5 .2 6 .8]);
new_x=deval(testSoln,new_t);

%plot time courses
plot(calc_t, [calc_x new_x']);
legend('X1','X2','X3','X4','X1true','X2true','X3true','X4true');

%clear unused variables
clear Si1;
clear Si2;
clear Si3;
clear Si4;
clear bd1;
clear bd2;
clear bd3;
clear bd4;
clear calc_x;
clear new_x;
clear testSoln
clear calc_t;
clear new_t;

toc
```