SPEM (S-system Parameter Estimation Method) Vignette

Yang Xinyi, Jennifer E. Dent, Christine Nardini

1 Goal

SPEM (S-system Parameter Estimation Method) package allows for the computation of parameters in the n-gene S-system from time series data.

2 Introduction

Biological systems are composed of interacting components [**?**]. The process of the expression and the interactions of these components are nonlinear. S-systems have a power law formalism which is general enough to capture the base of the observed response, making S-systems fit for the representation of biological process.

An S-system that represents the dynamics of n genes is described as:

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{n} X_j^{g_{i,j}} - \beta_i \prod_{j=1}^{n} X_j^{h_{i,j}}, i = 1, ..., n \tag{1}$$

where, X_i is the expression level of the i-th gene and n is the total number of genes in the network. α and β represent constant scale parameters, indicating the intensity of the relationship between gene X_i and other genes in the network. $g_{i,j}$ and $h_{i,j}$ are exponential parameters (referred to as kinetic orders), indicating the action of whether gene i activates or inhibits gene j, respectively.

S-systems have a fixed structure, in which each parameter has a well-defined meaning to map onto the gene regulatory network. In addition, as S-systems are based on power-laws they therefore, after log transformation, become easy to mathematically manipulate, and thus, easy to analyze.

In this SPEM package, we aim to reconstruct gene networks from time-series expression data using the S-system model. The input dataset should be as an ExpressionSet data container, describing, in assayData, expression data for n genes (rows) and m time points (columns), along with a vector of length m, which records the exact values of time points, thus showing the sample intervals in phenoData. SPEM will calculate the parameters α, β, G and H (where G and H are matrices containing the kinetic orders) of the S-system function set that best fits the dataset (note that multiple solutions may exist).

In our network, interactions between genes will be $+$, $-$ or 0, which means the result is a directed and signed network. A non-zero interaction represents an edge in the network. Figure **??** shows a very simple example to explain the relationship between predicted parameters and networks. The figure shows a four-gene pathway. Parameters given as the same format of outputs of SPEM are shown on the left-hand-side of the figure, while the network's function set format is shown on the top-right. In particular, G and H matrices are used to reconstruct the graphical representation, or interaction matrix of the network. If elements g_{ij} or h_{ij} are non-zero, there will be an activation or inhibition regulation from gene j to gene i, respectively. As α is dependent on the initial (user-given or randomly generated) value for β, α and β do not appear in the reconstructed example network. In fact, α and β represent constant scale parameters, indicating the intensity of the relationship between gene X_i and other genes in the network [**?**], giving weights to interactions needed only during the reconstruction of the system. This means that the resulting topology of the system, and indeed the dynamics, are independent of α and β, and G and H contain the final parameters that indicate the activatory or inhibitory action of one gene on another.

Therefore, when using experimental data, highly affected by noise, it is advisable to have more than one instance of SPEM to ensure the solution is reliable. In our experience, 10 is a largely safe number of runs. Average or rounding to the 1st decimal point are reasonable ways to define g_{ij} and h_{ij} for the final network.

In addition, we offer two additional parameters: "Initial beta" and "error" at the end of the input to give further information during the calculating process. SPEM uses non-linear optimization (Rsolnp [**?**]) to solve the reconstruction problem. As in all optimization procedures, SPEM needs a random value to start with, this value for each regression is "Initial beta", and SPEM minimizes the objective function which is a positive value: "error". This means the smaller the error is, the closer we have got to our goal.

Whilst users can refer to our publication for more information about S-systems and the SPEM algorithm [**?**], here Figure **??** briefly describes the process of the SPEM algorithm. SPEM's outputs are the parameters in the S-system function. SPEM works directly on the logarithmic transition and its inverse matrix but solves the bottleneck described in [**?**, **?**] where multiple solutions can exist in noisy data. Additionally, our

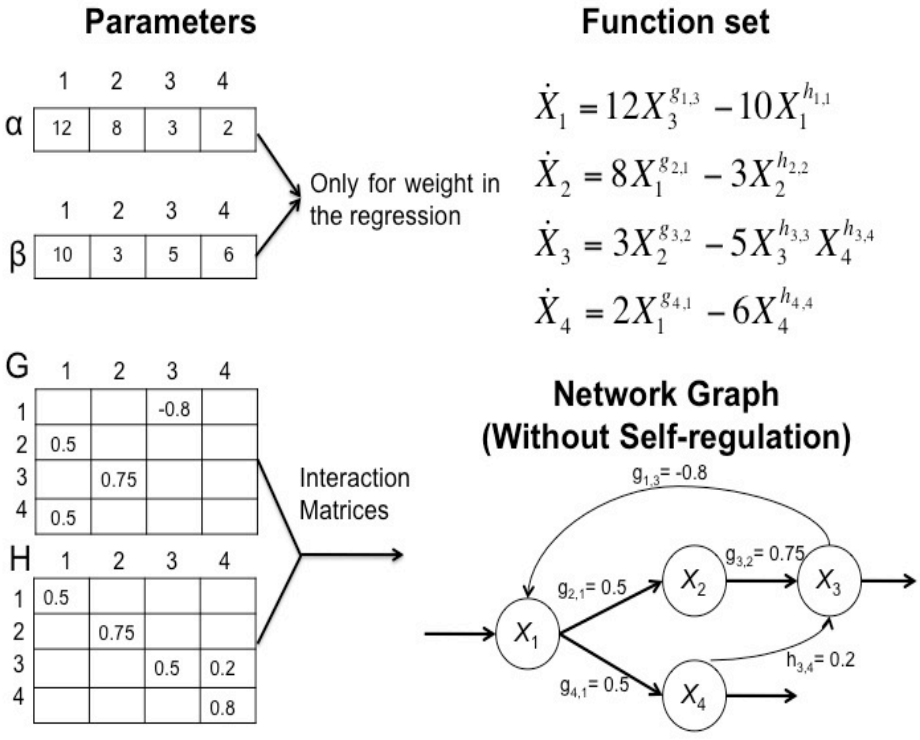

Figure 1: Reconstruction of an S-system network: The left-hand-side of the figure shows the predicted α, G, β and H parameters. The top-right gives the function format of the system with the predicted parameters. The bottom-right shows the graphical representation of the network, according to the predicted parameters. Each non-zero element g_{ij}, h_{ij} gives an activation or inhibition from gene j to gene i, respectively. Elements that represent self-regulation (arrows with no joining properties, i.e., g_{ii} and h_{ii}) are not shown in the graph.

approach provides slope calculation (lacking in the 2 cited methods) and can calculate more than one gene at a time, implying numerous inputs for users.

Results of SPEM show the interactions between genes and their predicted rates of change. The output can therefore be used to build hypotheses that lead to the design of experiments in wet labs. By outputting the full system, users are also able to investigate the characteristics of the system (short and long term); useful for informing further study.

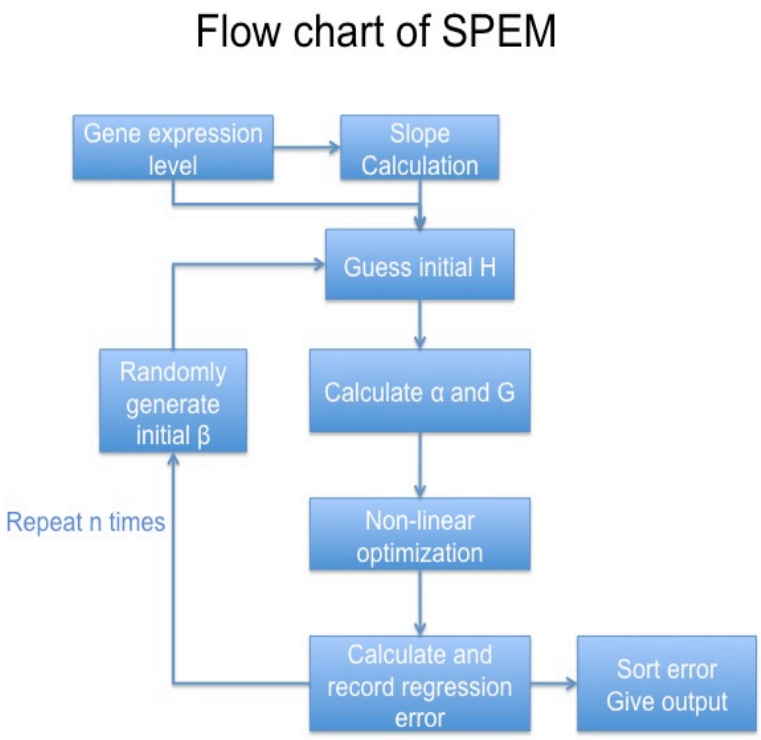

Figure 2: Main flow chart of SPEM

3 Example Dataset and Preparation of Your Own Data

In this package we offer the SOS dataset obtained from Uri Alon's lab [**?**] and prepared in ExpressionSet class. To give some details, the SOS response is a general DNA repair system in bacteria which allows survival after DNA damage [**?**]. As reported by existing literature, LexA and RecA are hub genes in the regulatory SOS pathway (they connect many other genes) and are bound to the interaction sites of other genes as master repressors. When DNA is damaged, RecA will sense the damage and mediates LexA auto cleavage. Figure **??** shows the pathway of SOS response. The SOS dataset used in SPEM is taken from real experiment expression data in *Escherichia coli*. The dataset contains 8 genes (uvrD, lexA, umuD, recA, uvrA, uvrY, ruvA and polB) taken from Experiment 3 (UV light intensities, 4:20 Jm^{-2}) and available to users at http://www.weizmann.ac.il/mcb/UriAlon/.

SPEM can be applied to other real biological time series datasets for which users must prepare their own datasets in ExpressionSet datatype. Here, we give the detailed

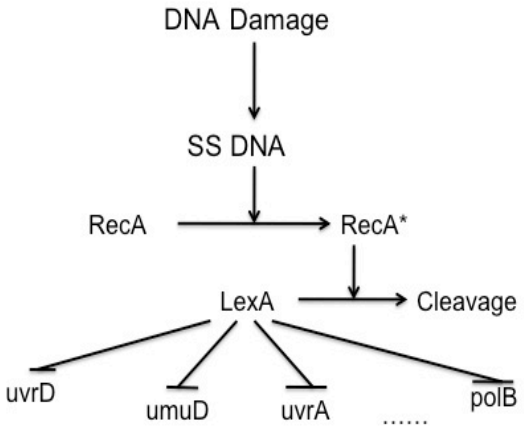

Figure 3: Pathway of SOS response in E. *coli*. LexA and RecA are hub genes in this pathway. When DNA is damaged, RecA senses the damage and mediates LexA auto cleavage.

steps to show how to build a toy dataset with 3 genes (G1, G2, and G3) with positive values generated at random to represent expression levels, and 4 time points (0, 2, 4, 6).

1) Install and load package: Biobase.

```
> library("Biobase")
```

2) Set expression data as matrix form with genes in rows and time points in columns.

```
> toy_expression_data<-matrix(data=abs(rnorm(12)),nrow=3,ncol=4,
+         dimnames=list(paste("G",c(1:3),sep=''),
+         paste("tp",c(0,2,4,6),sep="_")))
```

3) Prepare phenoData, which contains two parts for the data table: a) time points sample and, b) varMetadata, according to the following steps.

a) Time points sample. Prepare the time points sample in a data.frame, with the number of time points equal to the number of rows. Time point details should be given in the first column, labelled "time". Other columns of the same length can be added to give further information if desired. The example here has two columns "index" and "label".

```
> toy_timepoints_data<-data.frame(index=c(1:4),
+         label=paste("tp",c(0,2,4,6),sep='_'),
+         time=c(0,2,4,6),row.names=paste("tp",c(0,2,4,6),sep='_'))
```

b) Prepare varMetadata. To make your dataset more informative, you might need to add some explanation for your time points data labels. varMetadata is also a data.frame with only one column labeled as "labelDescription". All your column labels in the time point data should be row names of the varMetadata. Elements in the varMetadata are your label explanations.

```
> toy_varMetadata<-data.frame(labelDescription=c("Index number","Label Detail",
+         "Time points values"),row.names=c("index","label","time"))
> toy_varMetadata
```

```
      labelDescription
index       Index number
label       Label Detail
time  Time points values
```

c) Integrate time points data and varMetadata to generate the phenoData.

```
> toy_phenoData<-new("AnnotatedDataFrame",
+         data=toy_timepoints_data,varMetadata=toy_varMetadata)
```

4) Integrate expression data and phenoData to generate your own dataset.

```
> toy_ExpressionSet<-new("ExpressionSet",
+         exprs=toy_expression_data,phenoData=toy_phenoData)
```

4 Inputs and Outputs of SPEM

4.1 Input parameters

`TS_eSet`: Time series data in ExpressionSet class. assayData: Matrix with n metabolite in row and m time points in column. phenoData: phenoData type. The sample data.frame should include the label "time", which represents the values of time points.

`n`: Positive integer, SPEM will guess initial β n times

`sparsity`: A positive number. In order to force the interaction matrix to be sparse (biologically relevant, see [**?**] for further details), interactions with absolute value smaller than "sparsity" will be set to zero.

`lbH,ubH,lbB,ubB`: Lower boundary value, upper boundary value for h and β, respectively.

4.2 Output parameters

`alpha, g, beta, h`: Parameters of the reconstructed S-system.

`IniBeta`: Guess of the initial β value (generated randomly by SPEM itself).

`error`: Regression error.

4.3 Functions

`SPEM`: Main function, calculates parameters from entire time series matrix.

`s_diff`: Calculates slope matrix from time points and time series matrix.

`row_optimize`: Calculates parameters from time series matrix row by row.

The example data in SPEM is an 8-gene dataset, representing the SOS pathway in *Escherichia coli* [**?**]. The main function in this package is `SPEM`. Users can calculate the entire parameter matrix in one run. In addition we also offer a sub function called `row_optimize`, which can calculate parameters row by row if desired. Further more, we provide a function called `s_diff` to calculate the slope of time series data. Details are shown below.

4.4 Use SPEM to predict entire parameter matrix.

Load the package.

```
> library(SPEM)
> library(Rsolnp)
> library(Biobase)
```

Load the SOS pathway data.

```
> data(sos)
```

Set parameters.

```
> n<- 1
> sparsity<- 0.2
> lbH<- -3
> ubH<- 3
> lbB<- 0
> ubB<- 10
```

Calculate results.

result<-SPEM(sos,n,sparsity,lbH,ubH,lbB,ubB)

Note that initial β is generated randomly, so the exact result may differ in every round. If the user wants to increase the stability of the results, a bigger n can be set to guess β more times.

4.5 Use `s_diff` to calculate the slope of a time series matrix.

`SPEM` offers slope calculation in the package, using a curve-fitting algorithm (`predict`) to decrease the noise level as a first step. In order to improve the accuracy, 10 times the number of original time points are then inserted, at the same time lag, to the fitted curve. The slope around the original time points is then calculated according the time points that have been added to either side of the original measurement. Slope calculation function is named `s_diff`.

Calculate the slope. Note that the SPEM can calculate the slope matrix in one round.

Slope<- s_diff(sos)

4.6 Use `row_optimize` to estimate a single row of time series data.

Calculate "Slope" and "Initial beta". The user can either input a vector for slope or use `s_diff` to calculate it. The vector S should be the slope of gene row to be calculated.

Set parameters.

```
> Slope<- s_diff(sos)
> S<-Slope[1,]
> sparsity<- 0.2
> lbH<- -3
> ubH<- 3
> lbB<- 0
> ubB<- 10
> beta<-runif(n=1,min=lbB,max=ubB)
```

Calculate results.

result<-row_optimize(sos,S,beta,sparsity,lbH,ubH,lbB,ubB)

4.7 How to reconstruct the network from the result.

After running `SPEM`, the entire G and H matrix is outputted. These two matrices are used to reconstruct the network. If `row_optimize` is run for each gene, it must be run row by row and each result combined to obtain the G and H matrices in full.

Users can simply combine G and H matrices in an interaction matrix A where $A_{ij} = max(g_{ij}, h_{ij})$. This produces a square matrix, A, where A_{ij} represents the interaction from gene j to gene i.