SELANSI: SEmiLAgrangian Numerical SImulation of Gene Regulatory Networks version 1.0.0 User's guide

Manuel Pájaro [*a], Irene Otero-Muras [†a], Carlos Vázquez [‡b], and Antonio A. Alonso [§a]

[a]Process Engineering Group, IIM-CSIC. Spanish National Research Council. Eduardo Cabello 6, 36208 – Vigo (Spain)
[b]Department of Mathematics, University of A Coruña. Campus Elviña s/n, 15071 – A Coruña (Spain)

[*]mpajaro@iim.csic.es
[†]ireneotero@iim.csic.es
[‡]carlosv@udc.es
[§]antonio@iim.csic.es

Contents

1 SELANSI

- Webpage: `https://sites.google.com/view/selansi`
- Authors: Manuel Pájaro Diéguez, Irene Otero-Muras, Carlos Vázquez Cendón, Antonio Álvarez Alonso.
- e-mail: antonio@iim.csic.es
- Version: 1.0.0
- License: GPLv3
- Copyright: CSIC, Spanish National Research Council

1.1 Software requirements and compatibility

SELANSI is compatible with Matlab under Windows, Linux and MacOS.

SELANSI has been tested with Matlab versions 2011b, 2015b and 2016b.

1.2 Overview

1.2.1 The problem

SELANSI approximates the solution of the Chemical Master Equation (CME) for multidimensional gene regulatory networks(GRN). Given a GRN expressing n proteins $\mathbf{X} = \{X_1, \cdots, X_n\}$, we make use of the results in [Pajaro et al.(2017)], which generalize the one dimensional model proposed by [Friedman et al. (2006)], to approximate the corresponding CME by a multidimensional partial integral differential equation (PIDE). The governing equation describes the temporal evolution of the joint density distribution function $p : \mathbb{R}_+ \times \mathbb{R}^n_+ \to \mathbb{R}_+$

for the n proteins, and reads as follows:

$$\frac{\partial p(t,\mathbf{x})}{\partial t} = \sum_{i=1}^{n} \left(\frac{\partial}{\partial x_i} \left[\gamma_x^i x_i p(t,\mathbf{x})\right] - k_m^i c_i(\mathbf{x}) p(t,\mathbf{x}) \right. \\ \left. + k_m^i \int_0^{x_i} \omega_i(x_i - y_i) c_i(\mathbf{y}_i) p(t,\mathbf{y}_i)\, \mathrm{d}y_i \right) \quad (1)$$

where the amount of each protein type in given by the vector $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{R}_+^n$. In the above expression, $\mathbf{y}_i$ represents the vector state $\mathbf{x}$ whose i position is changed for y_i, namely $(\mathbf{y}_i)_j = x_j$ if $j \neq i$ and $(\mathbf{y}_i)_j = y_i$ if $j = i$. Transcription into the corresponding $mRNA_i$ takes place at a frequency k_m^i, whereas γ_x^i corresponds with the degradation rate for each protein.

The dynamics associated to the translation from $mRNA_i$ into the corresponding protein X_i is incorporated in the model through probability transition functions $\omega_i(x_i - y_i)$ which represent the conditional probability for protein jumping from a state y_i to x_i. Under a burst condition for protein production, where burst size follows an exponential distribution, such functions become:

$$\omega_i(x_i - y_i) = \frac{1}{b_i} \exp(-\frac{x_i - y_i}{b_i}) \quad (2)$$

where $b_i = \frac{k_x^i}{\gamma_m^i}$ are dimensionless frequencies associated to translation that correspond with the mean protein produced per burst (burst size). Burst condition holds whenever messenger RNA degrades mush faster than proteins [Friedman et al. (2006)]. Note however that, as reported in [Pajaro et al.(2017)], the above model provides very reasonable approximations already for degradation rate ratios in the order of $3 - 5$.

Functions $c_i(\mathbf{x})$ ($c_i : \mathbb{R}_+^n \to [\varepsilon_i,\ 1]$) in (1) model network regulation topology as well as the switching dynamics (activation/inactivation) of gene states in the network. When the promoter switching rates between on and off states are much faster than transcription-translation, the corresponding expressions c_i will be defined in terms of Hill-type functions (see Section 3.2.1). Note that the model structure in (1) can accommodate other switching dynamics (see for instance the examples discussed in sections 6.1.2 and 6.3)

Our model provides a reliable description of gene regulatory networks with general kinetics under mild assumptions. Networks under consideration might involve multiple genes with self and cross regulations, in which genes can be regulated by different transcription factors.

A scheme of the transcription-translation mechanisms under study is depicted in Fig. 1:

-The promoters associated with the genes of interest are assumed to switch between active ($DNAi_{\text{on}}$) and inactive ($DNAi_{\text{off}}$) states, with rate constants k^i_{on} and k^i_{off} per unit time, respectively.

-The transition is controlled by a feedback mechanism induced by the binding/unbinding of a given number of X_j-protein molecules, which makes the network self-regulated if $i = j$ or cross-regulated if $j \neq i$.

-Transcription of messenger RNA ($mRNA_i$) from the active $DNAi$ form, and translation into protein X_i occurs at rates (per unit time) k^i_m and k^i_x, respectively. k^i_ε is the rate constant associated with transcriptional leakage.

-The $mRNA_i$ and X_i-protein degradations occur by first order processes with rate constants γ^i_m and γ^i_x respectively.

-Important: We assume that proteins are produced in bursts, i.e. $\gamma^i_m >> \gamma^i_x$. Our model is proven to reliably approximate the CME solution for $\gamma^i_m > 2\gamma^i_x$ [Pajaro et al.(2017)].

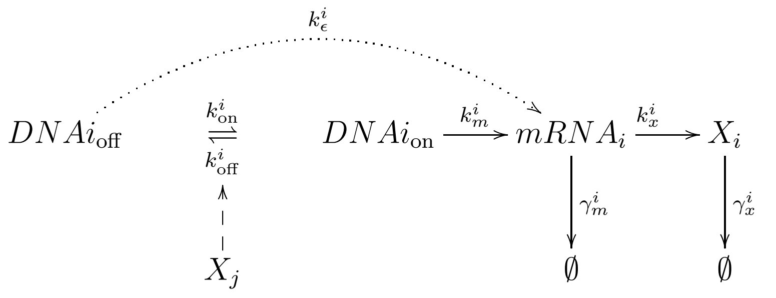

$$\begin{tikzcd}
DNAi_{\text{off}} \arrow[rrrr, dotted, bend left, "k^i_\epsilon"] & \overset{k^i_{\text{on}}}{\underset{k^i_{\text{off}}}{\rightleftharpoons}} & DNAi_{\text{on}} \arrow[r, "k^i_m"] & mRNA_i \arrow[r, "k^i_x"] \arrow[d, "\gamma^i_m"] & X_i \arrow[d, "\gamma^i_x"] \\
 & X_j \arrow[u, dashed] & & \emptyset & \emptyset
\end{tikzcd}$$

Figure 1: Schematic representation of the transcription-translation mechanism under simulation, $i = 1, \ldots, n$.

The chemical reaction steps encoding the transcription-translation mechanism for each gene are described in Table 1.

1. $\emptyset \xrightarrow{k_m^i c_i(\mathbf{x})} mRNA_i$	3. $mRNA_i \xrightarrow{\gamma_m^i} \emptyset$
2. $mRNA_i \xrightarrow{k_x^i} mRNA_i + X_i$	4. $X_i \xrightarrow{\gamma_x^i} \emptyset$

Table 1: Chemical reaction steps encoding the transcription-translation mechanism for each gene $i = 1, \ldots, n$. Reaction steps 1 and 2 represent transcription and translation respectively. Steps 3 and 4 describe degradation of $mRNA_i$ and X_i.

Within this framework, a gene regulatory network mechanism is completely defined by setting:

- Number of genes n.
- Kinetics (input functions $c_i(\mathbf{x})$). SELANSI incorporates predefined input functions for Hill kinetics (see section 3.2.1). Alternatively, the user can easily define his/her own input functions modifying the template provided (see section 3.2.2).
- k_m^i: transcription rate constant per unit time of messenger $mRNA_i$ from active DNA_i (for $i = 1, \ldots, n$).
- k_x^i: translation rate constant per unit time of messenger $mRNA_i$ into protein X_i (for $i = 1, \ldots, n$)..
- γ_x^i: degradation rate constant of protein X_i (for $i = 1, \ldots, n$)..
- γ_m^i: degradation rate constant of messenger $mRNA_i$ (for $i = 1, \ldots, n$)..

A problem is defined by setting:

- Gene regulatory mechanism (as indicated above).
- Initial condition (a probability density function).
- Mesh for the semilagrangian method (discretization of the protein amount and time domains).

Section 3 contains full details on how to proceed to define a new problem with SELANSI.

Once the problem is defined, SELANSI solves the PIDE model using the semilagrangian numerical method [Pajaro et al.(2017)], providing the temporal evolution of the protein probability density function.

1.2.2 Main features of SELANSI

- **Stochastic simulation of multidimensional gene regulatory networks:** Networks under consideration might involve multiple genes with self and cross regulations, in which genes can be regulated by different transcription factors.
- **Generality:** The validity of the method is not restricted to a particular type of kinetics (mass action, Hill, etc). Different kinetic types can be defined by the user (through the input functions c_i in Table 1).
- **Flexibility:** The toolbox offers total flexibility, since the user can decide the size and topology of the gene regulatory network, kinetics, parameters, time horizon for simulation, discretisation, etc.
- **Computational efficiency:** The semilagrangian method implemented in SELANSI is proven to simulate with high efficiency and accuracy the stochastic dynamics of multidimensional gene regulatory networks [Pajaro et al.(2017)]. The method can efficiently afford networks of 3 genes with meshes in the order of 10^2, as it works very efficiently for mesh sizes up to 10^7 points for 8 GB RAM computers. Such number can increase up to 5 coupled genes at affordable computation times for reasonably smooth solutions, with uniform meshes in the order of 30 points in each direction.

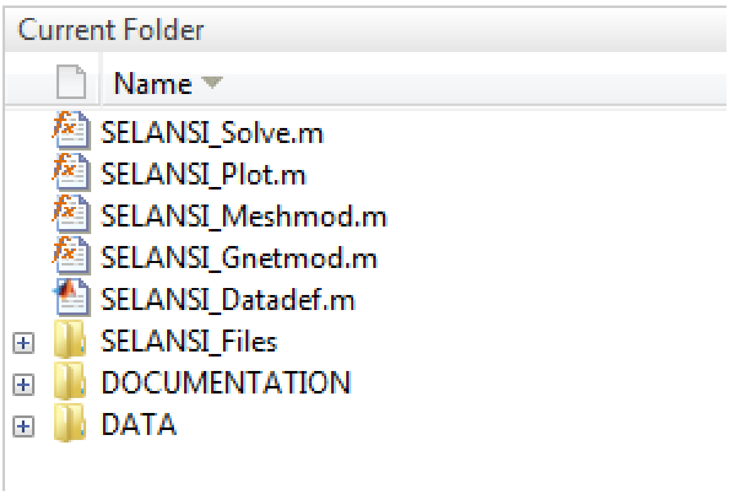
1.2.3 Scheme of the toolbox

The SELANSI directory contains the following folders:

- SELANSI_Files contains internal toolbox code (not to be modified by the user).
- DOCUMENTATION contains the user's manual
- DATA is the folder in which data and simulation results are saved.

The main tasks available in SELANSI (functions in the SELANSI folder) are:

- `SELANSI_Datadef` creates the necessary data to define a new problem to be solved with the semilagrangian method (information is requested to the user through the command window).
- `SELANSI_Gnetmod` modifies default network parameters and feedback mechanism.
- `SELANSI_Meshmod` modifies default mesh for the semilagrangian method and the initial condition.
- `SELANSI_Solve` runs the semilagrangian method to compute the numerical solution.
- `SELANSI_Plot` plots the numerical solution.



```
SELANSI_Solve.m
SELANSI_Plot.m
SELANSI_Meshmod.m
SELANSI_Gnetmod.m
SELANSI_Datadef.m
SELANSI_Files
DOCUMENTATION
DATA
```

Figure 2: Folders and Functions in the SELANSI directory.

1.3 Installation

1. Copy the SELANSI folder in a directory of your choice.
2. **IMPORTANT: SELANSI functions must be called always from the main directory (SELANSI).**

1.4 Questions and troubleshooting

For questions, feedback and troubleshooting, please contact `mpajaro@iim.csic.es`

1.5 About this manual

Basic knowledge on stochastic dynamics, gene regulatory networks and Matlab usage is assumed.

2 Quick start

- Start Matlab.
- Go to the SELANSI directory.
- Solve one of the provided examples running:

  ```
  SELANSI_Solve('Example1D')
  ```

  The input *'Example1D'* is the folder name within the directory DATA where the example parameters are saved.

  You can run any of the examples provided in the DATA folder, running:

  ```
  SELANSI_Solve('examplename')
  ```

  The input *examplename* is the folder name within the directory DATA where the example parameters are saved.

3 Definition of a new problem to solve with SELANSI

There are two options to define a new problem:

1. Using a template: the user can copy an example within the DATA directory, rename it and use it as a template substituting the default data by the new data. The detailed procedure is explained in section 3.1.

2. Using the function `SELANSI_Datadef`: the user runs `SELANSI_Datadef` to introduce the data (number of genes, parameters, initial condition...) defining the new problem. The detailed procedure is explained in section 3.3

3.1 Definition of a new problem using a template

The functions `SELANSI_Gnetmod` and `SELANSI_Meshmod` are used to modify old data saved in a folder inside the directory DATA.

- `SELANSI_Gnetmod` is used to modify the parameters of the network (see 3.1.1).

- `SELANSI_Meshmod` is used to modify the mesh data and initial conditions (see 3.1.2).

To preserve the old data, the user can copy the old folder and change its name to make the modifications in a new folder (for example: "example2Dbis").

Then, the name of the folder is used as an input of the functions `SELANSI_Gnetmod` and `SELANSI_Meshmod`, as explained next.

3.1.1 Network parameters modification

In order to modify the network parameters use:

```
SELANSI_Gnetmod('example2Dbis')
```

the data inside the corresponding folder Reaction_data and the function `IF_FM_user` (if it is the case) will be automatically opened. Changes can be made to these files (see section 3.2.2)

3.1.2 Mesh data and initial condition modification

In order to modify the mesh specifications use:

```
SELANSI_Meshmod('example2Dbis')
```

the files inside the folder Mesh_data and the function `IC_Function` (initial condition) of your example folder will be automatically opened. The user should modify these files to accommodate its needs (see section 3.2.3).

Important: In order to modify an existing model, please use the appropriate functions (`SELANSI_Gnetmod` and `SELANSI_Meshmod`). Do not change the .txt files directly.

3.2 How to modify the default files

In the following sections we describe all files used by SELANSI which should be modified by the user (see Table 2).

3.2.1 Using predefined Hill function: H.txt, K.txt, epsilon.txt

In the SELANSI formalism, the probability of gene i being in the off state by the action of protein X_j is given by:

$$\rho_{ij}(x_j) = \frac{x_j^{H_{ij}}}{x_j^{H_{ij}} + K_{ij}^{H_{ij}}}, \; i, j \in \{1, \cdots, n_{gene}\} \tag{3}$$

File	The user can modify	Section
`H.txt`	Hill coefficients for predefined Hill function	3.3.1.1.
`K.txt`	Hill constants for predefined Hill function	3.3.1.2.
`epsilon.txt`	epsilon coefficients for predefined Hill function	3.3.1.3.
`IF_FM_user`	Arbitrary input function	3.2.2
`IC_Function`	Initial condition	3.2.3
`R_constants.txt`	Network parameters	3.2.4
`Prot_mesh.txt`	Proteins space discretization	3.2.5
`Time_mesh.txt`	Time discretization	3.2.6

Table 2: Files which can be modified by the user. Note that the predefined Hill function is defined in Section 3.2.1.

where H_{ij} is denoted as the Hill coefficient of the regulation of gene i by protein j, and K_{ij} is the corresponding Hill constant, see [Pajaro et al.(2017)].

The Hill coefficients (integers) are collected in a $n_{gene} \times n_{gene}$ matrix, **H** where:

- $H_{ij} = 0$ if protein X_j does not regulate the expression of protein X_i.
- $H_{ij} > 0$ if protein X_j inhibits the expression of protein X_i.
- $H_{ij} < 0$ if protein X_j activates the expression of protein X_i.

The Hill constants (positive real variables defined as the equilibrium binding constants of the transcription factor to its DNA binding site) are collected in a $n_{gene} \times n_{gene}$ matrix, **K** where:

- $K_{ij} > 0$ if protein X_j regulates the expression of protein X_i.
- $K_{ij} = 0$ otherwise.

Using the probabilities ρ_{ij} (probability of gene i being in the off state by the action of protein X_j) given by Eq. (3), the predefined Hill function, for a network with one gene (with self regulation) is of the form:

$$c_1(\mathbf{x}) = \varepsilon_{11}\rho_{11}(x_1) + \varepsilon_{12}(1 - \rho_{11}(x_1)) \tag{4}$$

With $\varepsilon_{11}\rho_{11}(x_1)$ being the rate of transcription due to leakage (no transcription factor is activating) and $\varepsilon_{12}(1 - \rho_{11}(x_1))$ being the rate of transcription rate due to activation by the transcription factor. The coefficient corresponding to the state of full activation is $\varepsilon_{12} = 1$.

For a network with two genes with mutual regulation (and no self regulation), the predefined Hill function corresponding to the regulation of the gene 1 is of the form:

$$c_1(\mathbf{x}) = \varepsilon_{11}\rho_{12}(x_2) + \varepsilon_{12}(1 - \rho_{12}(x_2)) \tag{5}$$

where the coefficient corresponding to the state of full activation is $\varepsilon_{12} = 1$.

The Hill function corresponding to the regulation of the gene 2 is of the form:

$$c_2(\mathbf{x}) = \varepsilon_{21}\rho_{21}(x_1) + \varepsilon_{22}(1 - \rho_{21}(x_1)) \tag{6}$$

where the coefficient corresponding to the state of full activation is $\varepsilon_{22} = 1$.

The general formula for the input function $c_i(x)$ in a n gene network (in which all genes regulate each other, i.e. without any null Hill coefficient) is of the form:

$$c_i(x) = \mathcal{C}_i^n \cdot \boldsymbol{\varepsilon_i} \tag{7}$$

where $\boldsymbol{\varepsilon}_i$ is a (column) vector collecting all epsilon coefficients (for gene i), $\mathcal{C}_i^n$ follows the recursive formula:

$$\mathcal{C}_i^n = \mathcal{C}_i^{n-1} \otimes [\rho_{in}(x_n), 1 - \rho_{in}(x_n)] \tag{8}$$

with $\otimes$ being the Kronecker product and:

$$\mathcal{C}_i^1 = [\rho_{i1}(x_1), 1 - \rho_{i1}(x_1)]\,. \tag{9}$$

In case the network has some null Hill coefficients the formula becomes:

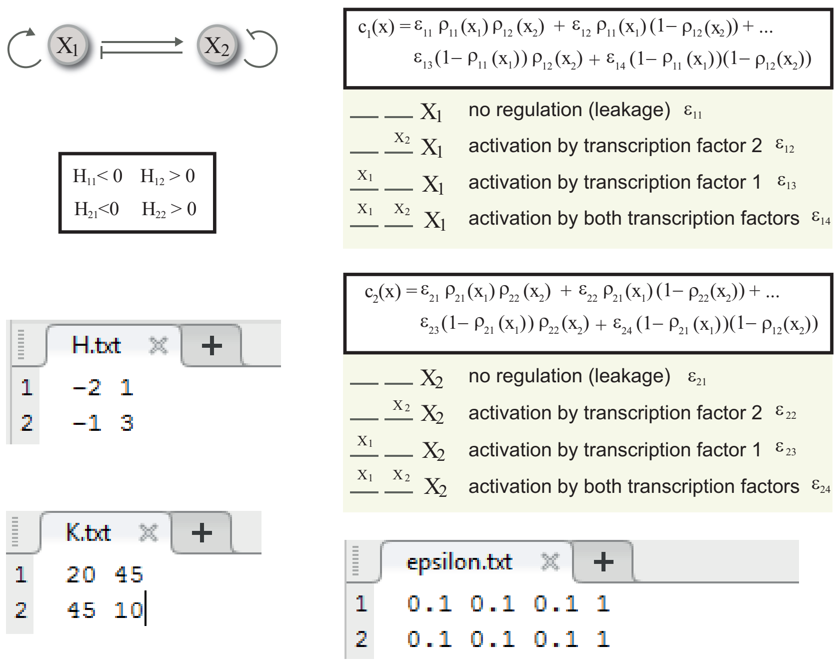
$$\mathcal{C}_i^n = \mathcal{C}_i^{n-1} \otimes F^n \tag{10}$$

with $\mathcal{C}_i^1 = F^1$ and

$$F^j = \begin{cases} [\rho_{ij}(x_n), 1 - \rho_{ij}(x_n)] & \text{if } \; H_{ij} \neq 0 \\ 1 & \text{if } \; H_{ij} = 0 \end{cases} \tag{11}$$

for all $j = 1, \ldots, n$.

Illustrative examples are included in figures 3, 4 and 5.



$H_{11} < 0 \quad H_{12} > 0$

$H_{21} < 0 \quad H_{22} > 0$

$$c_1(x) = \varepsilon_{11}\, \rho_{11}(x_1)\, \rho_{12}(x_2) + \varepsilon_{12}\, \rho_{11}(x_1)(1 - \rho_{12}(x_2)) + \ldots$$
$$\varepsilon_{13}(1 - \rho_{11}(x_1))\, \rho_{12}(x_2) + \varepsilon_{14}(1 - \rho_{11}(x_1))(1 - \rho_{12}(x_2))$$

$\_\_\ \_\_\ X_1$	no regulation (leakage)	ε_{11}
$\_\_\ \overset{X_2}{\_\_}\ X_1$	activation by transcription factor 2	ε_{12}
$\overset{X_1}{\_\_}\ \_\_\ X_1$	activation by transcription factor 1	ε_{13}
$\overset{X_1}{\_\_}\ \overset{X_2}{\_\_}\ X_1$	activation by both transcription factors	ε_{14}

$$c_2(x) = \varepsilon_{21}\, \rho_{21}(x_1)\, \rho_{22}(x_2) + \varepsilon_{22}\, \rho_{21}(x_1)(1 - \rho_{22}(x_2)) + \ldots$$
$$\varepsilon_{23}(1 - \rho_{21}(x_1))\, \rho_{22}(x_2) + \varepsilon_{24}(1 - \rho_{21}(x_1))(1 - \rho_{12}(x_2))$$

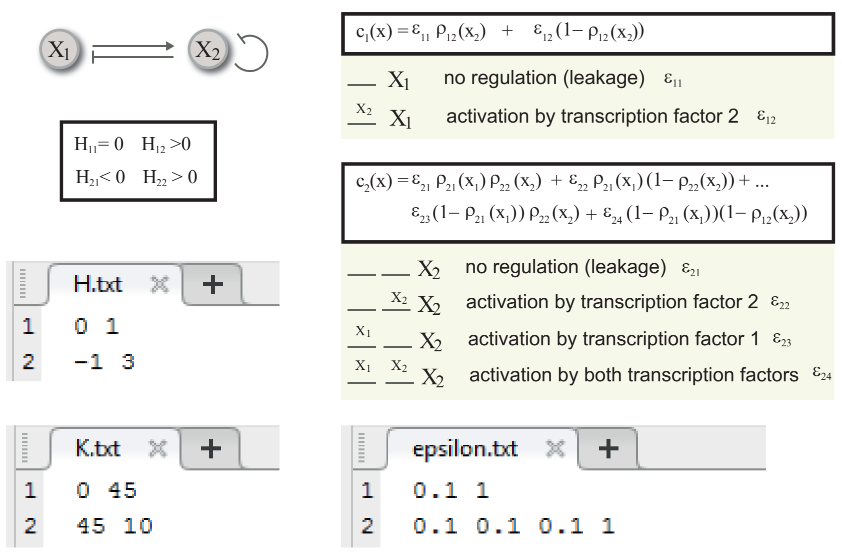
$\_\_\ \_\_\ X_2$	no regulation (leakage)	ε_{21}
$\_\_\ \overset{X_2}{\_\_}\ X_2$	activation by transcription factor 2	ε_{22}
$\overset{X_1}{\_\_}\ \_\_\ X_2$	activation by transcription factor 1	ε_{23}
$\overset{X_1}{\_\_}\ \overset{X_2}{\_\_}\ X_2$	activation by both transcription factors	ε_{24}

H.txt
```
-2 1
-1 3
```

K.txt
```
20 45
45 10
```

epsilon.txt
```
0.1 0.1 0.1 1
0.1 0.1 0.1 1
```

Figure 3: Two-gene example with self and mutual regulation: H matrix, K matrix, vector of epsilon coefficients and expression of the input function corresponding to genes 1 and 2.


X_1
X_2
$H_{11}= 0$ $H_{12} >0$
$H_{21}< 0$ $H_{22} > 0$
X_1 no regulation (leakage) ε_{11}
X_2 X_1 activation by transcription factor 2 ε_{12}
X_2 no regulation (leakage) ε_{21}
X_2 X_2 activation by transcription factor 2 ε_{22}
X_1 X_2 activation by transcription factor 1 ε_{23}
X_1 X_2 X_2 activation by both transcription factors ε_{24}
H.txt
K.txt
epsilon.txt

$$c_1(x) = \varepsilon_{11}\,\rho_{12}(x_2) + \varepsilon_{12}(1-\rho_{12}(x_2))$$

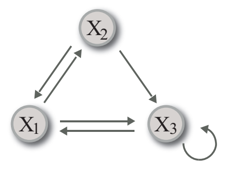
$$c_2(x) = \varepsilon_{21}\,\rho_{21}(x_1)\,\rho_{22}(x_2) + \varepsilon_{22}\,\rho_{21}(x_1)(1-\rho_{22}(x_2)) + \dots$$
$$\varepsilon_{23}(1-\rho_{21}(x_1))\,\rho_{22}(x_2) + \varepsilon_{24}(1-\rho_{21}(x_1))(1-\rho_{12}(x_2))$$

H.txt
```
0 1
-1 3
```

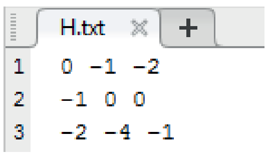
K.txt
```
0 45
45 10
```

epsilon.txt
```
0.1 1
0.1 0.1 0.1 1
```

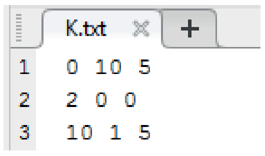
Figure 4: Two-gene example with mutual regulation: H matrix, K matrix, vector of epsilon coefficients and expression of the input function corresponding to genes 1 and 2.



$$\boxed{\begin{array}{lll} H_{11}=0 & H_{12}<0 & H_{13}<0 \\ H_{21}<0 & H_{22}=0 & H_{23}=0 \\ H_{31}<0 & H_{32}<0 & H_{33}<0 \end{array}}$$



```
0 -1 -2
-1 0 0
-2 -4 -1
```



```
0 10 5
2 0 0
10 1 5
```

$$c_1(x) = \varepsilon_{11}\, \rho_{12}(x_2)\, \rho_{13}(x_3) + \varepsilon_{12}\, \rho_{12}(x_2)\, (1-\rho_{13}(x_3)) + \ldots$$
$$\varepsilon_{13}(1-\rho_{12}(x_2))\, \rho_{13}(x_3) + \varepsilon_{14}(1-\rho_{12}(x_2))(1-\rho_{13}(x_3))$$

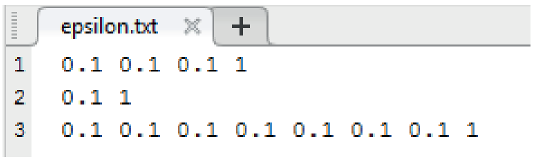
$\underline{\quad}\ \underline{\quad}$	X_1	no regulation (leakage)	ε_{11}
$\underline{\quad}\ \underline{X_3}$	X_1	activation by transcription factor 3	ε_{12}
$\underline{X_2}\ \underline{\quad}$	X_1	activation by transcription factor 2	ε_{13}
$\underline{X_2}\ \underline{X_3}$	X_1	activation by both transcription factors	ε_{14}

$$c_2(x) = \varepsilon_{21}\, \rho_{21}(x_1) + \varepsilon_{22}\, (1-\rho_{21}(x_1))$$

$\underline{\quad}$	X_2	no regulation (leakage)	ε_{21}
$\underline{X_1}$	X_2	activation by transcription factor 1	ε_{22}

$$c_3(x) = \varepsilon_{31}\, \rho_{31}(x_1)\, \rho_{32}(x_2)\, \rho_{33}(x_3) + \varepsilon_{32}\, \rho_{31}(x_1)\, \rho_{32}(x_2)(1-\rho_{33}(x_3)) + \ldots$$
$$\varepsilon_{33}\, \rho_{31}(x_1)\, (1-\rho_{32}(x_2))\rho_{33}(x_3) + \varepsilon_{34}\, \rho_{31}(x_1)\, (1-\rho_{32}(x_2))\, (1-\rho_{33}(x_3)) + \ldots$$
$$\varepsilon_{35}(1-\rho_{31}(x_1))\rho_{32}(x_2)\, \rho_{32}(x_3) + \varepsilon_{36}(1-\rho_{31}(x_1))\rho_{32}(x_2)(1-\rho_{33}(x_3)) + \ldots$$
$$\varepsilon_{37}(1-\rho_{31}(x_1))(1-\rho_{32}(x_2))\, \rho_{33}(x_3) + \varepsilon_{38}(1-\rho_{31}(x_1))(1-\rho_{32}(x_2))(1-\rho_{33}(x_3))$$

$\underline{\quad}\ \underline{\quad}\ \underline{\quad}$	X_3	no regulation (leakage)	ε_{31}
$\underline{\quad}\ \underline{\quad}\ \underline{X_3}$	X_3	activation by transcription factor 3	ε_{32}
$\underline{\quad}\ \underline{X_2}\ \underline{\quad}$	X_3	activation by transcription factor 2	ε_{33}
$\underline{\quad}\ \underline{X_2}\ \underline{X_3}$	X_3	activation by transcription factors 2,3	ε_{34}
$\underline{X_1}\ \underline{\quad}\ \underline{\quad}$	X_3	activation by transcription factor 1	ε_{35}
$\underline{X_1}\ \underline{\quad}\ \underline{X_3}$	X_3	activation by transcription factors 1,3	ε_{36}
$\underline{X_1}\ \underline{X_2}\ \underline{\quad}$	X_3	activation by transcription factors 1,2	ε_{37}
$\underline{X_1}\ \underline{X_2}\ \underline{X_3}$	X_3	activation by transcription factors 1,2,3	ε_{38}

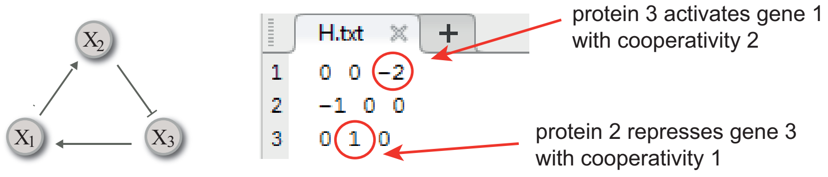


```
0.1 0.1 0.1 1
0.1 1
0.1 0.1 0.1 0.1 0.1 0.1 0.1 1
```

Figure 5: Three gene network: H matrix, K matrix, vector of epsilon coefficients and expression of the input function corresponding to genes 1, 2 and 3.

3.3.1.1. The matrix of Hill coefficients: H.txt

The file `H.txt` contains the matrix of Hill coefficients of the system.



```
0 0 -2
-1 0 0
0 1 0
```

protein 3 activates gene 1 with cooperativity 2

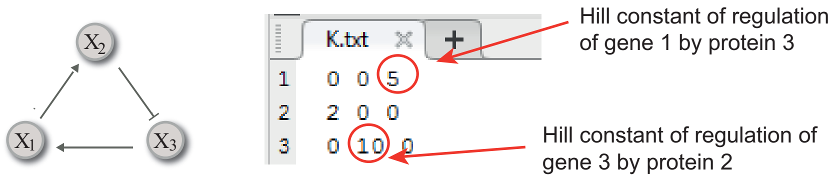
protein 2 represses gene 3 with cooperativity 1

Figure 6: Example of H file for a 3 gene network

Important: The matrix of Hill coefficients determines the connectivity of the system. Note that, when modifying the H coefficients with `SELANSI_Gnetmod`, you can only re-write the values of nonzero entries. To change the network connectivity, you should define a new problem with `SELANSI_Datadef`.

3.3.1.2. The matrix of Hill constants: K.txt

The file `K.txt` contains the matrix of Hill constants of the system.



```
0 0 5
2 0 0
0 10 0
```

Hill constant of regulation of gene 1 by protein 3

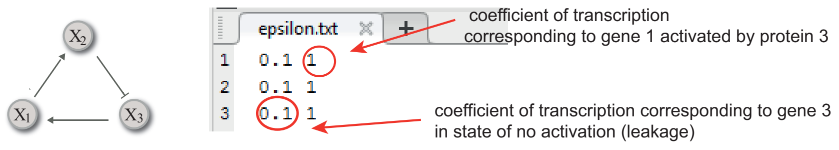
Hill constant of regulation of gene 3 by protein 2

Figure 7: Example of K file for a 3 gene network

Important: The structure (location of nonzero elements) of the matrix of Hill constants is determined by the connectivity of the system (i.e. by the location of nonzero elements in the matrix of Hill coefficients).

3.3.1.3. The matrix of epsilon coefficients: epsilon.txt

The file `epsilon.txt` contains the vectors of epsilon coefficients of the system. The vector in row i corresponds to regulated gene i. The vector in row i has as many entries as possible states of regulation for the gene i. SELANSI generates the epsilon values by default (the user can change the values).


X_2
X_1
X_3
epsilon.txt
coefficient of transcription corresponding to gene 1 activated by protein 3
coefficient of transcription corresponding to gene 3 in state of no activation (leakage)

```
0.1 1
0.1 1
0.1 1
```

Figure 8: Example of epsilon file for a 3 gene network

3.2.2 Feedback mechanism input function: IF_FM_user

The Matlab function `IF_FM_user` is contained in Fig. 9. The inputs of this function are:

- **Xgrid**, a $n_{gene} \times 1$ cell of multidimensional arrays with the dimension of the mesh.
- **ci**, integer index used to choose the desired $c_i(\boldsymbol{x})$, i.e. the input function for the regulated gene $i =$ **ci**.

Lines 8, 16 and 32 of `IF_FM_user` file, see Fig. 9, should not be changed. The user is free to insert the local feedback mechanism parameters and to define the desired input function $c_i(\boldsymbol{x})$. Element-wise operators must be used in order to construct functions $c_i(\boldsymbol{x})$ with correct dimensions.

3.2.3 Initial condition: IC_Function

The Matlab initial condition function, which can be modified by the user is in Fig. 10. The inputs of this function are:

- **x**, a $n_{gene} \times 1$ cell of vectors. Each vector $x\{i\}$ is the mesh of the i protein state space.
- **Xgrid**, a $n_{gene} \times 1$ cell of multidimensional arrays with the dimension of the mesh.

```
function cx=IF_FM_user(Xgrid,ci)
%%%
% cx=IF_FM_user(Xgrid,ci)
% user defined input function
%%%

% Number of genes
n_gene = length(Xgrid); % DO NOT MODIFY

% Feedback mechanism parameters
H=2*eye(n_gene);
K=10*eye(n_gene);
epsilon=0.1*ones(n_gene,1);

% Saving all c_i(x) in a cell
cx_cell=cell(n_gene,1); % DO NOT MODIFY

% We define the functions c_i, note that the variable x is a cell of
% dimension equal to the number of genes and each element x{i} is a
% multidimensional (n_gene) array which the same dimensions of your mesh.
% Use elementwise multiplication (.*) division (./) power (.^) ...
% in your input function expression.
for i=1:n_gene
if H(i,i)>0
cx_cell{i}=(epsilon(i)*Xgrid{i}.^H(i,i)+K(i,i)^H(i,i))./(Xgrid{i}.^H(i,i)+K(i,i)^H(i,i));
elseif H(i,i)<0
cx_cell{i}=(Xgrid{i}.^(-H(i,i))+epsilon(i)*K(i,i)^(-H(i,i)))./(Xgrid{i}.^(-H(i,i))+K(i,i)^(-H(i,i)));
end
end

% cx returns the input function (c_i(x)), i.e. the input function for regulated gene i=ci
cx=cx_cell{ci}; % DO NOT MODIFY
end
```

Figure 9: The feedback mechanism input function in Matlab

The initial condition function defined in `IC_Function` is a multidimensional Gaussian function with mean $x0g$ (line 12 in Figure 10) and standard deviation $sigmag$ (line 13 in Figure 10). The user is free to modify this initial condition, but must use element-wise operators in order to construct the initial condition with correct dimensions. Note that the initial condition is automatically normalized.

3.2.4 Network parameters: R_constants.txt

The file `R_constants.txt` contains a $n_{gene} \times 4$ matrix collecting the network parameters:

$$\begin{pmatrix} k_m^i & k_x^i & \gamma_m^i & \gamma_x^i \end{pmatrix} \tag{12}$$

for $i = 1, \ldots, n_{gene}$ (see Fig. 1).

3.2.5 Proteins space discretization: Prot_mesh.txt

The file `Prot_mesh.txt` is a $n_{gene} \times 3$ matrix collecting the minimum and maximum number of proteins together with the mesh elements for each protein, $n_{x_i} \in \mathbb{N}$:

$$\begin{pmatrix} \min(x_i) & \max(x_i) & n_{x_i} \end{pmatrix} \tag{13}$$

for $i = 1, \ldots, n_{gene}$. The mesh step is $\Delta x_i = (\max(x_i) - \min(x_i))/n_{x_i}$.

3.2.6 Time discretization: Time_mesh.txt

The file `Time_mesh.txt` is a 1×4 vector collecting the following time discretization data:

$$\begin{pmatrix} t_0 & t_{end} & \nu & n_{solsave} \end{pmatrix} \tag{14}$$

where t_0 is the initial time, t_{end} is the final time, $n_{solsave} \in \mathbb{N}$ is the number of time steps for which the solution is saved, and ν is an integer number such that the product $N_{tot} = \nu n_{solsave}$ is the total number of elements of the time mesh. In this way, we can use a finer time mesh for simulation than the one saved (which might be convenient in multidimensional problems for

```matlab
function IC=IC_Function(x,Xgrid)
%%%
% IC=IC_Function(x,Xgrid)
% IMPORTANT: The initial Condition is approached by a
    Gaussian density and must be a
% multidimensional array of dimension equal to Xgrid{i}
    with i=1,...,n_gene.
%%%

% Number of genes
n_gene = length(Xgrid);

% Parameters of the Gaussian density funcion
x0g=0*ones(1,n_gene);  % Mean of the Gaussian density
sigmag=5*ones(1,n_gene); % Standard deviation of the
    Gaussian density

gausker=0;
for i=1:n_gene
    gausker = gausker+((Xgrid{i}-x0g(i))/sigmag(i)).^2;
end
PX0un=exp(-gausker/2);

% Norm of the initial condition
auxnor0=PX0un;
for i=1:n_gene
    auxnor0 = trapz(x{i},auxnor0);
end

% Normalization of the initial condition to be a density
    function
IC=PX0un/auxnor0;
end
```

Figure 10: The initial condition function in Matlab

memory reasons). Note that, for $\nu = 1$ the simulation mesh and the saved mesh coincide (all time simulations are saved).

The time step is $\Delta t = (t_{end} - t_0)/N_{tot}$.

3.2.7 Assessment of discretization parameters and accuracy

The recommended minimum and maximum number of proteins x_i can be computed as $\min(x_i) = 0$ and $\max(x_i) = M\frac{k_m^i k_x^i}{\gamma_m^i \gamma_x^i}$, where M is a number with a typical value $M = 2$. If the computed density does not vanish at the corresponding boundaries, then the user must increase the value of M ($M = 3, 4, \dots$). In most cases $M = 3$ is large enough so that density vanishes at the boundaries.

For time intervals in the order of (0, 50) a number of mesh points in the order of $n_{x_i} = 300$, together with a number of time steps $N_{tot} = 10^4$, should be enough in terms of accuracy of the solution (for the case $H > 0$, N_{tot} can be reduced to 10^3).

In order to evaluate the accuracy of the solution, we recommend to increase both the number of time steps and the mesh points, n_{x_i}, (for example, doubling them) and check the convergence of the resulting solution. In addition, users are encouraged to make use of tables in [Pajaro et al.(2017)] as guidelines for the discretization level. Such tables record convergence rates of the solutions for different representative examples, including comparisons with analytical solutions or SSA simulations.

Note that a high increase in the number of discretization points in the gene directions and time can lead to memory overload, which also depends on the computer capacity.

3.3 Definition of a new problem using SELANSI_Datadef

Within the SELANSI directory, run:

```
SELANSI_Datadef
```

The following information will be requested through the command window:

1. Number of genes

   ```
   Write the number (integer) of genes considered
   ```

   The user must insert the number of genes, n_{gene}, in the network considered (an integer number).

2. Name of the folder

   ```
   Write the folder name for saving data and results:
   ```

   The user must insert a name for the folder where the data (and results) will be saved (for example: `example2D`)

3. Type of input function

   ```
   Choose your feedback mechanism input function
   Insert 1 if you want to use a predefined Hill
      function
   Insert 0 if you define your input function
   ```

   The user can select a Hill type input function for the feedback mechanism (inserting 1), or define other different input functions (inserting 0).

3'. Matrix with Hill coefficients

   ```
   Write the correct Hill coefficients (substitute
      values by default) and SAVE
   Push INTRO at the end to continue
   ```

   If the selected option is 1 (Hill type input function), the user should provide the entries of the matrix of Hill coefficients.

4. Initial condition

   ```
   Write your Initial Condition and SAVE
   Check that it is correct
   Push INTRO at the end to continue
   ```

The file `IC_Function` opens automatically, and the user can modify the initial condition for the semilagrangian numerical method by writing the desired function within the file `IC_Function`, following the indications therein (see section 3.2.3). This file is saved in your folder "example2D" within the directory DATA.

Once the initial condition is defined, the user can modify some files to insert the desired data, depending on the selected input function.

5.A. (Predefined Hill) if the user has chosen the option 1, five .txt files are automatically open: `K.txt`, `epsilon.txt`, `R_constants.txt`, `Prot_mesh.txt` and `Time_mesh.txt`, corresponding to the feedback equilibrium constants, the epsilon values (that collect coefficients of transcription as described in Section 3.2.1), the network rate constants, the spatial (proteins) mesh specifications and the temporal mesh, respectively (see sections 3.2.1 to 3.2.4). All these data are saved in folder "example2D" inside the directory DATA.

```
Write the constant values (substituting values by
    default) and SAVE
Push INTRO at the end to continue
```

The user can modify the values of choice in the corresponding files.

5.B. (User's own input function) if the user has chosen option 0:

- First, the input file `IF_FM_user` is automatically opened. This file is saved in folder "example2D" inside the directory DATA.

```
Write your input function and SAVE
Check that all parameters are correct
Push INTRO at the end to continue
```

  The user should make (and save) the desired changes in the `IF_FM_user` file to define its function (see section 3.2.2).

- Secondly, three .txt files are opened automatically: `R_constants.txt`, `Prot_mesh.txt` and `Time_mesh.txt`, which are the network rate constants, the spatial (proteins) mesh specifications and the time mesh, respectively. All these data are saved in folder "example2D" inside the directory DATA.

```
Write the constant values (substituting
   values by default) and SAVE
Push INTRO at the end to continue
```

The user should make (and save) the desired changes in the corresponding files (see sections 3.2.4, 3.2.5 and 3.2.6).

4 Numerical Simulation with SELANSI

Once the problem is defined, the user can call the semilagrangian solver to simulate the dynamics of the gene regulatory network. To call the solver, run:

```
SELANSI_Solve('examplename')
```

the input *examplename* is a string with the folder name within the directory DATA in which the network parameters were previously saved.

5 Results

Your data are saved in a folder in the directory DATA. Inside this folder, within the Results directory, the results are contained in the file `Solution.mat`. This file contains a Matlab structure with the following elements:

- **solution.T**, is a $n_{solsave} + 1$ vector collecting the times for which the solution is saved.
- **solution.x**, is a $n_{gene} \times 1$ cell of vectors. Each $n_{x_i} + 1$ vector is the mesh of the i protein state space, `solution.x{i}`$= (\min(x_i), \min(x_i) + \Delta x_i, \cdots, \min(x_i) + n_{x_i}\Delta x_i)$, for $i = 1, \cdots, n_{gene}$. Note that, $\min(x_i) + n_{x_i}\Delta x_i = \max(x_i)$.

- **solution.Xgrid**, a $n_{gene} \times 1$ cell of $(n_{x_1} + 1) \times \cdots \times (n_{x_{n_{genes}}} + 1)$ multidimensional arrays with the dimensions of your mesh, which is constructed using the Matlab function `ndgrid`:

```
% Protein space Mesh
Xgrid=cell(dato.n_gene,1);
[Xgrid{1:dato.n_gene}] = ndgrid(x{1:dato.n_gene});
```

- **solution.PTX**, is a $(n_{solsave}+1)\times 1$ cell collecting the solution at some discretization times. So that:

$$\texttt{solution.PTX\{i\}} = P(\texttt{solution.T\{i\}}, \texttt{solution.Xgrid\{1:n\_gene\}}) \tag{15}$$

with $i = 1, \cdots, n_{solsave} + 1$. Note that the solution, `solution.PTX{i}`, has the same dimensions of the mesh, `solution.Xgrid{i}`, for all $i = 1, \cdots, n_{gene}$.

5.1 Plot results

The results saved in each folder inside the directory DATA can be plotted using the function `SELANSI_Plot`:

```
SELANSI_Plot('examplename')
```

The input of this function is the folder name, *examplename*, where the numerical solution has been saved. Moreover, the user must choose the time T at which the solution will be plotted:

```
The solution can be ploted at different times:
initial time, T0=0, choose 0
final time, Tend=5, choose 100
intermediate time, T= T0 + 0.05*nt, choose an integer nt
   such that 1 < nt < 100
```

This time, T, is chosen inserting an integer number between 0 and $n_{solsave}$ (see section 3.2.6) such that:

$$T = t_0 + (\nu\Delta t) \cdot n_{solsave}.$$

6 Examples

In this section we illustrate the use of the SELANSI toolbox through a number of examples. These examples can be directly simulated by `SELANSI_Solve`. used to generate new problems for the same dimensions using the functions `SELANSI_Gnetmod` and `SELANSI_Meshmod`. Moreover these examples can be used to generate new problems for the same dimensions using the functions `SELANSI_Gnetmod` and `SELANSI_Meshmod`.

6.1 Example 1: Gene networks of one gene

We consider the one-gene network with self regulation depicted in Fig. 11 with Hill kinetics. We solve the example for the predefined Hill function in section 6.1.1. In order to illustrate the flexibility of SELANSI to incorporate any kind of kinetics, in section 6.1.2 we consider the same network structure with a kinetics of connectionist type [Munteanu et al.(2017)] using to this purpose the `IF_FM_user` function.

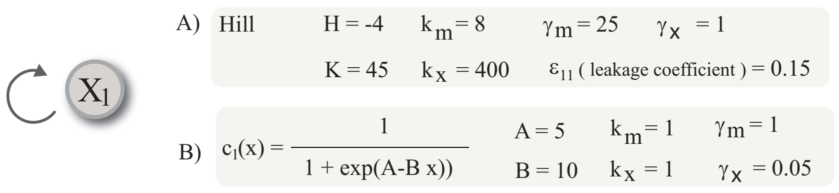


Figure 11: One dimensional gene network with self regulation: A) Hill kinetics. B) Connectionist kinetics.

6.1.1 Example 1D with predefined Hill function "Example1D"

We choose as the current folder of Matlab the SELANSI toolbox directory and we start running the function `SELANSI_Datadef` to create the necessary information to use `SELANSI_Solve`. Once we run `SELANSI_Datadef` the following information is requested (Re.) and inserted (In.) in the Matlab command window:

Re.1
```
Write the number (integer) of genes considered
```

In.1 We insert 1 since is the number of genes considered.
```
    1
```

Re.2
```
You have selected 1 genes
Write the folder name for saving data and results:
```

In.2 We insert a name for the folder were the data will be saved, in this case "Example1D".
```
Example1D
```

Re.3
```
The name for your folder is : Example1D
You can choose your feedback mechanism input
    function
Insert 1 if you want to use a predefined Hill
    function
Insert 0 if you define your input function
```

In.3 We select the predefined Hill function inserting the value 1.
```
1
```

Re.4
```
Write the correct Hill coefficients (substitute
    values by default) and SAVE
 Push INTRO at the end to continue
```

In.4 The H matrix file `H.txt` is open, we replace the default value (1) by -4 and push INTRO.

Re.5
```
Write your Initial Condition and SAVE
Check that it is correct
Push INTRO at the end to continue
```

In.5 The initial condition function `IC_Funtion` is open, but we do not change its expression.

```
Push INTRO
```

Re.6
```
Write the constant values (substituting values by
    default) and SAVE
Push INTRO at the end to continue
```

In.6 The txt files `R_constants.txt`, `Prot_mesh.txt`, `Time_mesh.txt`, `K.txt` and `epsilon.txt` are opened. We modify the default values of each file as follow:

`R_constants.txt`, the default value $\left(\ 2\quad 75\quad 25\quad 1\ \right)$ is replaced by $\left(\ 8\quad 400\quad 25\quad 1\ \right)$ and saved.

`Prot_mesh.txt`, the default value $\left(\ 0\quad 30\quad 300\ \right)$ is replaced by $\left(\ 0\quad 350\quad 1400\ \right)$ and saved.

`Time_mesh.txt`, the default value $\left(\ 0\quad 5\quad 10\quad 100\ \right)$ is replaced by $\left(\ 0\quad 50\quad 500\quad 50\ \right)$ and saved.

`K.txt`, the default value 1 is replaced by 45 and saved.

`epsilon.txt`, the default value for the leakage coefficient $(0.1\quad 1)$ is replaced by $(0.15\quad 1)$ and saved.

Finally, after we have inserted and saved the new values, we push INTRO in the command window.

```
Push INTRO
```

Re.7
```
 Your parameters are saved in C:\...\SELANSI\DATA\
    Example1D
```

At this moment we have already the necessary data to use the function `SELANSI_Solve` to obtain the numerical solution of the network considered. To this aim we write in the Matlab command window the following order:

```
SELANSI_Solve('Example1D')
```

After a few seconds we have the numerical solution saved inside the directory $\cdots$\SELANSI\DATA\Example1D\Results.

In order to show the results we run the function `SELANSI_Plot` which allows us to select the desired time of the solution:

```
SELANSI_Plot('Example1D')
```

The following information is provided:

```
The solution can be ploted at different times:
initial time, T0=0, choose 0
final time, Tend=50, choose 50
intermediate time, T= T0 + 1*nt, choose an integer nt such
   that 1 < nt < 50
```

We must insert an integer between 0 and 50, which correspond to the solution at time equal to the initial time plus the integer selected, we insert 0:

```
0
```

the following figure, which corresponds to the initial condition, is plotted:

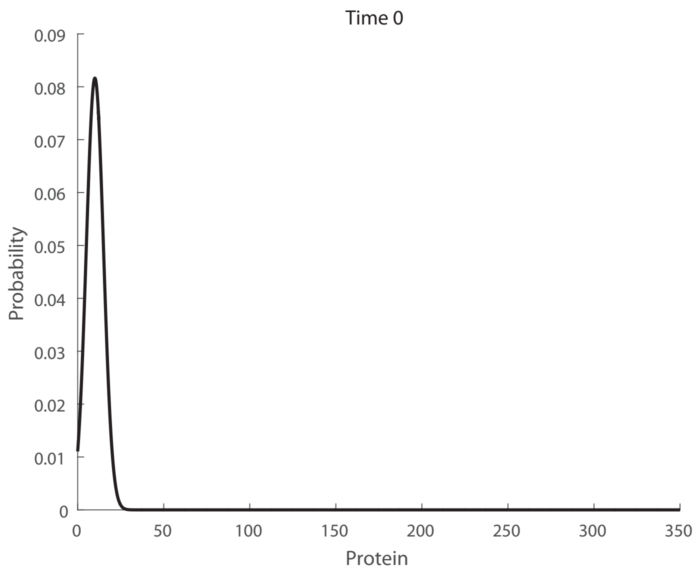


Figure 12: Snapshot of the probability distribution of protein abundance obtained at nt value 0 (the corresponding time are indicated in the title).

The user can run again the file `SELANSI_Plot` and select an intermediate time, for example 25:

```
25
```

the following figure, which corresponds to time 25, is plotted:

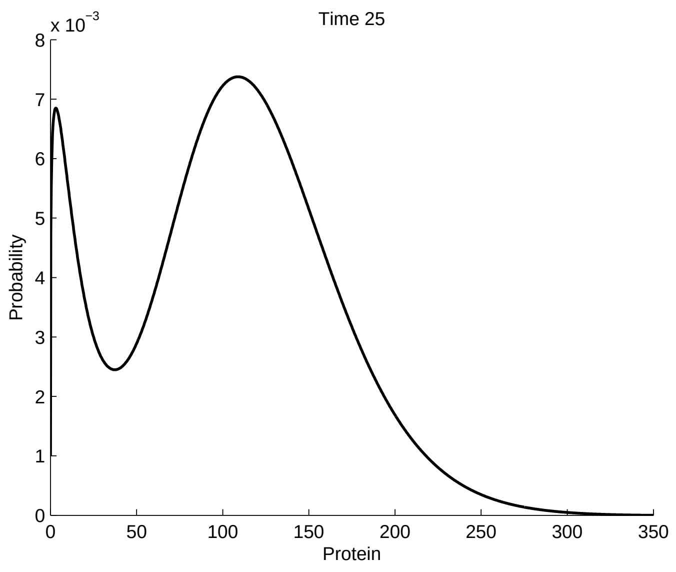


Figure 13: Snapshot of the probability distribution of protein abundance obtained at nt value 25 (the corresponding time are indicated in the title).

Finally we show also the final time considered, inserting 50 once the file `SELANSI_Plot` was run:

```
50
```

the following figure, which corresponds to time 50, is plotted:

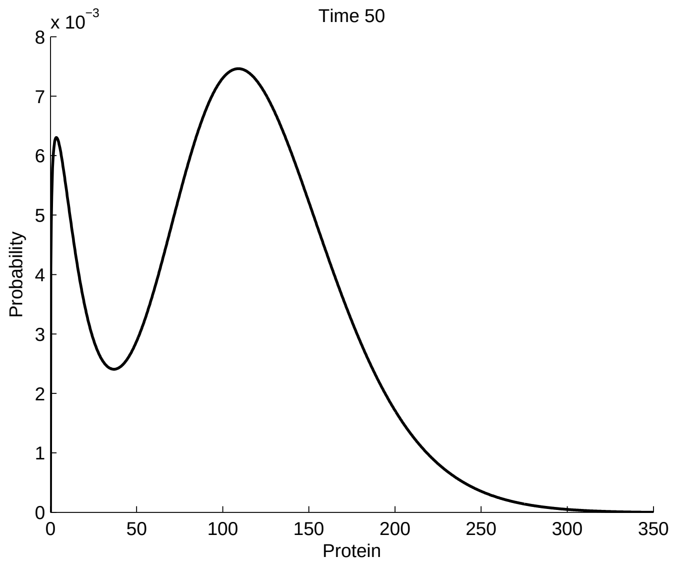


Figure 14: Snapshot of the probability distribution of protein abundance obtained at nt value 50 (the corresponding time are indicated in the title).

6.1.2 Example 1D with IF_FM_user function "Example1D_userIF"

We choose as the current Matlab folder the SELANSI toolbox directory and we start running the function `SELANSI_Datadef` to define the problem to solve with `SELANSI_Solve`. Once we run `SELANSI_Datadef` the following information is requested (Re.) and inserted (In.) in the Matlab command window:

Re.1
```
Write the number (integer) of genes considered
```

In.1 We insert 1 since is the number of genes considered.
```
1
```

Re.2
```
You have selected 1 genes
Write the folder name for saving data and results:
```

In.2 We insert a name for the folder were the data will be saved, in this case "Example1D_userIF".

```
Example1D_userIF
```

Re.3
```
The name for your folder is : Example1D_userIF
You can choose your feedback mechanism input
    function
Insert 1 if you want to use a predefined Hill
    function
Insert 0 if you define your input function
```

In.3 We select define our input function inserting the value 0.

```
0
```

Re.4
```
Write your Initial Condition and SAVE
Check that it is correct
Push INTRO at the end to continue
```

In.4 The initial condition function `IC_Funtion` is open, we change the values of the mean and standard deviation of the Gaussian density to 20 and 1, respectively, and save.

```
Push INTRO
```

Re.5
```
Write your input function and SAVE
Check that all parameters are correct
Push INTRO at the end to continue
```

In.5 The input function `IF_FM_user` is open, we use it as a template to introduce our input function as depicted in Fig. 15.

```
Push INTRO
```

Re.6
```
Write the constant values (substituting values by
    default) and SAVE
Push INTRO at the end to continue
```

```
function cx=IF_FM_user(Xgrid,ci)
%%%
% cx=IF_FM_user(Xgrid,ci)
% user defined input function
%%%

% Number of genes
n_gene = length(Xgrid); % DO NOT MODIFY

% Feedback mechanism parameters
A=5;
B=10;

% Saving all c_i(x) in a cell
cx_cell=cell(n_gene,1); % DO NOT MODIFY

% We define the functions c_i, note that the variable x is
    a cell of
% dimension equal to the number of genes and each element x
    {i} is a
% multidimensional (n_gene) array which the same dimensions
    of your mesh.
% Use elementwise multiplication (.*) division (./) power
    (.^) ...
% in your input function expression.
cx_cell{1}=1./(1+exp(A-B*Xgrid{1}));


% cx returns the input function (c_i(x)), i.e. the input
    function for regulated gene i=ci
cx=cx_cell{ci}; % DO NOT MODIFY
end
```

Figure 15: Input Function file for the connectionist model in Fig. 11 B

In.6 The txt files `R_constants.txt`, `Prot_mesh.txt`, `Time_mesh.txt` are opened. We modify the default values of each file as follow:

`R_constants.txt`, the default value $\begin{pmatrix} 2 & 75 & 25 & 1 \end{pmatrix}$ is replaced by $\begin{pmatrix} 1 & 1 & 1 & 0.05 \end{pmatrix}$ and saved.

`Prot_mesh.txt`, the default value $\begin{pmatrix} 0 & 30 & 300 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 50 & 500 \end{pmatrix}$ and saved.

`Time_mesh.txt`, the default value $\begin{pmatrix} 0 & 5 & 10 & 100 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 300 & 3000 & 100 \end{pmatrix}$ and saved.

Finally, after we have inserted and saved the new values, we push INTRO in the command window.

```
    Push INTRO
```

Re.7
```
    Your parameters are saved in C:\...\SELANSI\DATA\
        Example1D_userIF
```

At this moment we have already the necessary data to use the function `SELANSI_Solve` to obtain the numerical solution of the network considered. To this aim we write in the Matlab command window the following order:

```
SELANSI_Solve('Example1D_userIF')
```

After a few seconds we have the numerical solution saved inside the directory ···\SELANSI\DATA\Example1D_userIF\Results.

In order to show the results we run the function `SELANSI_Plot` which allows us to select the desired time of the solution:

```
SELANSI_Plot('Example1D_userIF')
```

The following information is provided:

```
 The solution can be ploted at different times:
 initial time, T0=0, choose 0
 final time, Tend=300, choose 100
 intermediate time, T= T0 + 3*nt, choose an integer nt such
      that 1 < nt < 100
```

We must insert an integer between 0 and 100, which correspond to the solution at time equal to the initial time plus 3 times the integer selected, we insert 0:

```
0
```

the following figure, which corresponds to the initial condition, is plotted:

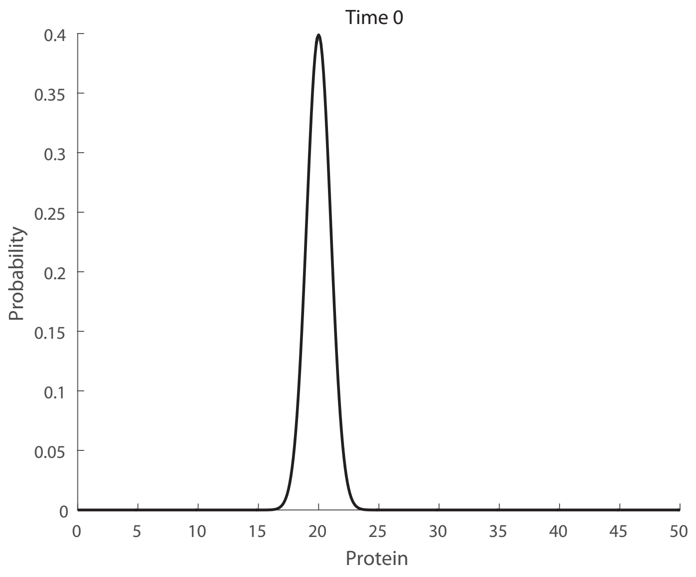


Figure 16: Snapshot of the probability distribution of protein abundance obtained at nt value 0 (the corresponding time are indicated in the title).

The user can run again the file `SELANSI_Plot` and select an intermediate time, for example with $nt = 2$:

```
2
```

the following figure, which corresponds to time 6, is plotted:

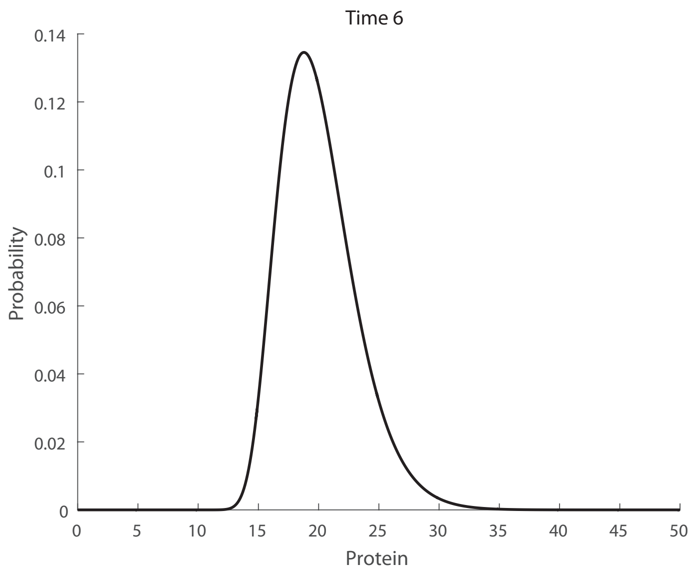


Figure 17: Snapshot of the probability distribution of protein abundance obtained at nt value 2 (the corresponding time are indicated in the title).

Finally we show also the final time considered, inserting 100 once the file `SELANSI_Plot` was run:

```
100
```

the following figure, which corresponds to time 300, is plotted:

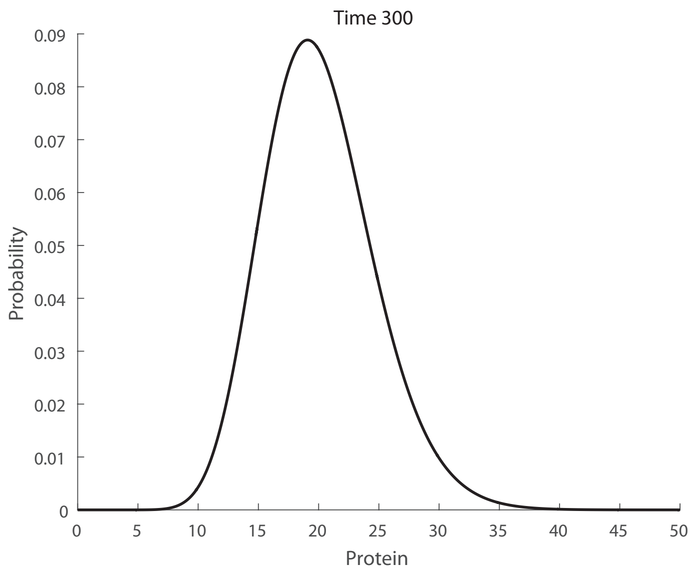


Figure 18: Snapshot of the probability distribution of protein abundance obtained at nt value 100 (the corresponding time are indicated in the title).

6.2 Example 2: Gene networks of two gene "Example2D"

We consider the two gene network in Fig. 19, with two genes mutually repressed, where the protein produced by one gene type inhibits the other gene expression with Hill kinetics.

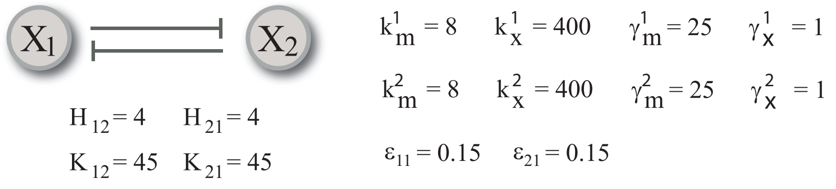

X_1
X_2
$H_{12} = 4$ $H_{21} = 4$
$K_{12} = 45$ $K_{21} = 45$
$k^1_m = 8$ $k^1_x = 400$ $\gamma^1_m = 25$ $\gamma^1_x = 1$
$k^2_m = 8$ $k^2_x = 400$ $\gamma^2_m = 25$ $\gamma^2_x = 1$
$\varepsilon_{11} = 0.15$ $\varepsilon_{21} = 0.15$

Figure 19: Two dimensional gene network with mutual repression

For the simulation of this example the predefined Hill function of section

3.2.1 is used.

We start choosing the SELANSI toolbox directory as the current folder of Matlab. We run the function `SELANSI_Datadef` to create the necessary files to use `SELANSI_Solve`. Once we run `SELANSI_Datadef` the following information is requested (Re.) and inserted (In.) in the Matlab command window:

Re.1
```
Write the number (integer) of genes considered
```

In.1 We insert 2 since is the number of genes considered.
```
2
```

Re.2
```
You have selected 2 genes
Write the folder name for saving data and results:
```

In.2 We insert a name for the folder were the data will be saved, in this case "Example2D".
```
Example2D
```

Re.3
```
The name for your folder is : Example2D
You can choose your feedback mechanism input
    function
Insert 1 if you want to use a predefined Hill
    function
Insert 0 if you define your input function
```

In.3 We select the predefined Hill function inserting the value 1.
```
1
```

Re.4
```
 Write the correct Hill coefficients (substitute
     values by default) and SAVE
Push INTRO at the end to continue
```

In.4 The H matrix file `H.txt` is open, the default value $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 4 \\ 4 & 0 \end{pmatrix}$ and saved. Then we push INTRO.

Re.5
```
Write your Initial Condition and SAVE
Check that it is correct
Push INTRO at the end to continue
```

In.5 The initial condition function `IC_Funtion` is open, but we do not change its expression.

```
Push INTRO
```

Re.6
```
Write the  constant values (substituting values by
    default) and SAVE
Push INTRO at the end to continue
```

In.6 The txt files described in sections 3.2.4, 3.2.5, 3.2.6 and 3.2.1 are opened. We modify the default values of each file as follow:

`R_constants.txt`, the default value $\begin{pmatrix} 2 & 75 & 25 & 1 \\ 2 & 75 & 25 & 1 \end{pmatrix}$ is replaced by $\begin{pmatrix} 8 & 400 & 25 & 1 \\ 8 & 400 & 25 & 1 \end{pmatrix}$ and saved.

`Prot_mesh.txt`, the default value $\begin{pmatrix} 0 & 30 & 300 \\ 0 & 30 & 300 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 350 & 700 \\ 0 & 350 & 700 \end{pmatrix}$ and saved.

`Time_mesh.txt`, the default value $\begin{pmatrix} 0 & 5 & 10 & 100 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 20 & 80 & 50 \end{pmatrix}$ and saved.

`K.txt`, the default value $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 45 \\ 45 & 0 \end{pmatrix}$ and saved.

`epsilon.txt`, the default value $\begin{pmatrix} 0.1 & 1 \\ 0.1 & 1 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0.15 & 1 \\ 0.15 & 1 \end{pmatrix}$ and saved.

Finally, after we have inserted and saved the new values, we push INTRO in the command window.

```
Push INTRO
```

Re.7

```
Your parameters are saved in C:\...\SELANSI\DATA\
    Example2D
```

At this moment we have already the necessary data to use the function `SELANSI_Solve` to obtain the numerical solution of the network considered. To this aim we write in the Matlab command window the following order:

```
SELANSI_Solve('Example2D')
```

After a few minutes we have the numerical solution saved inside the directory ···\SELANSI\DATA\Example2D\Results.

In order to show the results we run the function `SELANSI_Plot` which allows us to select the desired time of the solution:

```
SELANSI_Plot('Example2D')
```

The following information is provided:

```
The solution can be ploted at different times:
initial time, T0=0, choose 0
final time, Tend=50, choose 50
intermediate time, T= T0 + 0.4*nt, choose an integer nt
    such that 1 < nt < 50
```

We must insert an integer between 0 and 50, which correspond to the solution at time equal to the initial time plus the 0.4 times the integer selected, we insert 0:

```
0
```

In Fig. 20, we show the snapshots of the probability distribution of protein abundance obtained at different *nt* values:

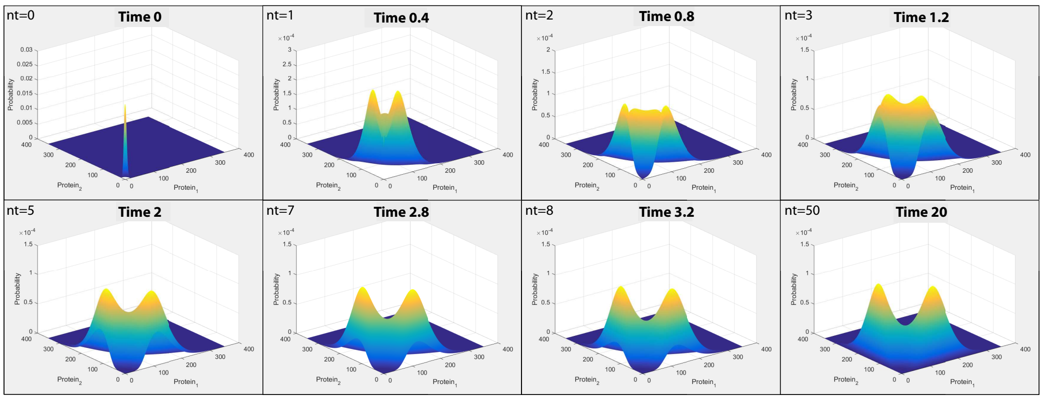
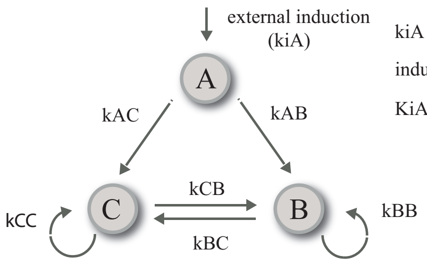


Figure 20: Snapshots of the probability distribution of protein abundance obtained at nt values 0,1,2,3,5,7,8, and 50 (the corresponding times are indicated in the title of each subfigure).

We remark here that for a given nt, `SELANSI_Plot` generates the plots corresponding to the marginal and joint distributions. In this case `SELANSI_Plot` generates three plots corresponding to the marginal distributions of protein 1, protein 2 and the joint probability distribution, respectively.

6.3 Example 3: Gene networks of three gene "Example3D"

We consider the network with external induction as well as self and mutual regulation among three genes as depicted in Fig. 21. In order to illustrate the flexibility of SELANSI to incorporate any kind of kinetics, in this example we use multi-dimensional transcription functions of the type of ratios of polynomials as described in [Gonze(2016)].



kiA = 0.1	kAB = 1	KAB = 100	nAB = 2	$k_m^1 = 1$	$k_x^1 = 1$
induc = 1	kBB = 0.01	KBB = 11	nBB = 2	$k_m^2 = 1$	$k_x^2 = 1$
KiA = 0.1	kCB = 1	KCB = 150	nCB = 4	$k_m^3 = 1$	$k_x^3 = 1$
	kAC = 1	KAC = 0.01	nAC = 2	$\gamma_m^1 = 1$	$\gamma_x^1 = 0.1$
	kBC = 1	KBC = 0.01	nBC = 4	$\gamma_m^2 = 10$	$\gamma_x^2 = 0.05$
	kCC = 0.01	KCC = 0.5	nCC = 2	$\gamma_m^3 = 0.1$	$\gamma_x^3 = 0.05$

$$c_1(x) = (kiA\,(induc/KiA))/(1 + kiA(induc/KiA))$$

$$c_2(x) = (kAB\,(x_1/KAB)^{nAB} + kCB\,(x_3/KCB)^{nCB} + kBB)/(1 + kAB\,(x_1/KAB)^{nAB} + kBB\,(x_2/KBB)^{nBB} + kCB\,(x_3/KCB)^{nCB})$$

$$c_3(x) = (kAC\,(x_1/KAC)^{nAC} + kCC\,(x_3/KCC)^{nCC} + kCB)/(1 + kAC\,(x_1/KAC)^{nAC} + kBC\,(x_2/KBC)^{nBC} + kCC\,(x_3/KCC)^{nCC})$$

Figure 21: Three dimensional gene network with external induction and mutual and self regulation. The kinetics is of the type of ratios of polynomials as described in [Gonze(2016)].

For the simulation of this example we define the considered input function using the `IF_FM_user` file (section 3.2.2).

We chose as the current folder of Matlab the SELANSI toolbox directory. We start running the function `SELANSI_Datadef` to create the necessary files to use `SELANSI_Solve`. Once we have run `SELANSI_Datadef` the following information is requested (Re.) and inserted (In.) in the Matlab command window:

Re.1
```
Write the number (integer) of genes considered
```

In.1 We insert 3 since is the number of genes considered.
```
3
```

Re.2
```
You have selected 3 genes
Write the folder name for saving data and results:
```

In.2 We insert a name for the folder were the data will be saved, in this case "Example3D".

```
Example3D
```

Re.3

```
The name for your folder is: Example3D
You can choose your feedback mechanism input
    function
Insert 1 if you want to use a predefined Hill
    function
Insert 0 if you define your input function
```

In.3 We select define our input function inserting the value 0.

```
0
```

Re.4

```
Write your Initial Condition and SAVE
Check that it is correct
Push INTRO at the end to continue
```

In.4 The initial condition function `IC_Funtion` is open and we change this file by substituting (in lines 12 and 13) the Gaussian mean and standard deviation as follows:

```
1   function IC=IC_Function(x,Xgrid)
2  %%%
3  % IC=IC_Function(x,Xgrid)
4  % IMPORTANT: The initial Condition is approached by a
       Gaussian density and must be a
5  % multidimensional array of dimension equal to Xgrid{i
       } with i=1,...,n_gene.
6  %%%
7
8  % Number of genes
9  n_gene = length(Xgrid);
10
11 % Parameters of the Gaussian density funcion
12 x0g=[5 5 75];  % Mean of the Gaussian density
13 sigmag=[1 1 5]; % Standard deviation of the Gaussian
       density
14
```

```
gausker=0;
for i=1:n_gene
    gausker = gausker+((Xgrid{i}-x0g(i))/sigmag(i))
        .^2;
end
PX0un=exp(-gausker/2);

% Norm of the initial condition
auxnor0=PX0un;
for i=1:n_gene
    auxnor0 = trapz(x{i},auxnor0);
end

% Normalization of the initial condition to be a
    density function
IC=PX0un/auxnor0;
end
```

We save this new initial condition and continue the program execution.

```
    Push INTRO
```

Re.5
```
    Write your input function and SAVE
    Check that all parameters are correct
    Push INTRO at the end to continue
```

In.5 The input function `IF_FM_user` is open, we change its default expression (see 9) by the following file:

```
    function cx=IF_FM_user(Xgrid,ci)
%%%
% cx=IF_FM_user(Xgrid,ci)
% user defined input function
%%%

% Number of genes
n_gene = length(Xgrid); % DO NOT MODIFY

% Feedback mechanism parameters
kAB = 1;
```

```
kBB = 0.01;
kCB = 1;
kAC = 1;
kBC = 1;
kCC = 0.01;

KAB = 100;
KBB = 1;
KCB = 50;
KAC = 0.01;
KBC = 0.01;
KCC = 0.5;

nAB = 2;
nBB = 2;
nCB = 4;
nAC = 2;
nBC = 4;
nCC = 2;

kiA=0.1;
induc=1.1;
KiA=1;


% Saving all c_i(x) in a cell
cx_cell=cell(n_gene,1); % DO NOT MODIFY

% We define the functions c_i , note that the variable
    x is a cell of
% dimension equal to the number of genes and each
    element x{i} is a
% multidimensional (n_gene) array which the same
    dimensions of your mesh.
% Use elementwise multiplication (.*) division (./)
    power (.^) ...
% in your input function expression.

cx_cell{1}=(kiA*(induc/KiA))/(1 + kiA*(induc/KiA));
cx_cell{2}=(kAB*(Xgrid{1}/KAB).^nAB + kCB*(Xgrid{3}/
    KCB).^nCB + kBB)./...
      (1 + kAB*(Xgrid{1}/KAB).^nAB + kBB*(Xgrid{2}/KBB)
          .^nBB + kCB*(Xgrid{3}/KCB).^nCB);
cx_cell{3}=(kAC*(Xgrid{1}/KAC).^nAC + kCC*(Xgrid{3}/
    KCC).^nCC + kCB)./...
      (1 + kAC*(Xgrid{1}/KAC).^nAC + kBC*(Xgrid{2}/KBC)
          .^nBC + kCC*(Xgrid{3}/KCC).^nCC);

% cx returns the input function (c_i(x)), i.e. the
    input function for regulated gene i=ci
cx=cx_cell{ci}; % DO NOT MODIFY
end
```

Once we have save the modified `IF_FM_user` file, we push INTRO to continue.

```
    Push INTRO
```

Re.6
```
      Write the  constant values (substituting values by
           default) and SAVE.
       Push INTRO at the end to continue
```

In.6 The txt files described in sections 3.2.4, 3.2.5 and 3.2.6 are opened. We modify the default values as follow:

`R_constants.txt`, the default value $\begin{pmatrix} 2 & 75 & 25 & 1 \\ 2 & 75 & 25 & 1 \\ 2 & 75 & 25 & 1 \end{pmatrix}$ is replaced by $\begin{pmatrix} 1 & 1 & 1 & 0.1 \\ 1 & 1 & 10 & 0.05 \\ 1 & 1 & 0.1 & 0.05 \end{pmatrix}$ and saved.

`Prot_mesh.txt`, the default value $\begin{pmatrix} 0 & 30 & 300 \\ 0 & 30 & 300 \\ 0 & 30 & 300 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 10 & 100 \\ 0 & 10 & 100 \\ 0 & 150 & 150 \end{pmatrix}$ and saved.

`Time_mesh.txt`, the default value $\begin{pmatrix} 0 & 5 & 10 & 100 \end{pmatrix}$ is replaced by $\begin{pmatrix} 0 & 400 & 160 & 50 \end{pmatrix}$ and saved.

Finally, we push INTRO in the command window.

```
    Push INTRO
```

Re.7

```
    Your parameters are saved in C:\...\SELANSI\DATA\
        Example3D
```

At this moment we have the necessary data to obtain the numerical solution for the network considered using `SELANSI_Solve`. To this aim we write in the Matlab command window the following order:

```
SELANSI_Solve('Example3D')
```

After approximately 2 hours (depending on the computer) we have the numerical solution saved inside the directory $\cdots$\SELANSI\DATA\Example3D\Results.

In order to show the results we run the function `SELANSI_Plot` which allows us to select the desired time of the solution:

```
SELANSI_Plot('Example3D')
```

The following information is provided:

```
The solution can be ploted at different times:
initial time, T0=0, choose 0
final time, Tend=400, choose 50
intermediate time, T= T0 + 8*nt, choose an integer nt such
    that 1 < nt < 50
```

We must insert an integer between 0 and 50, which correspond to the solution at time equal to the initial time plus 8 times the integer selected, we insert 0:

```
0
```

In Fig. 22, we show the snapshots of the probability distribution of protein abundance (joint probability of protein 1 and 2) obtained at different *nt*:

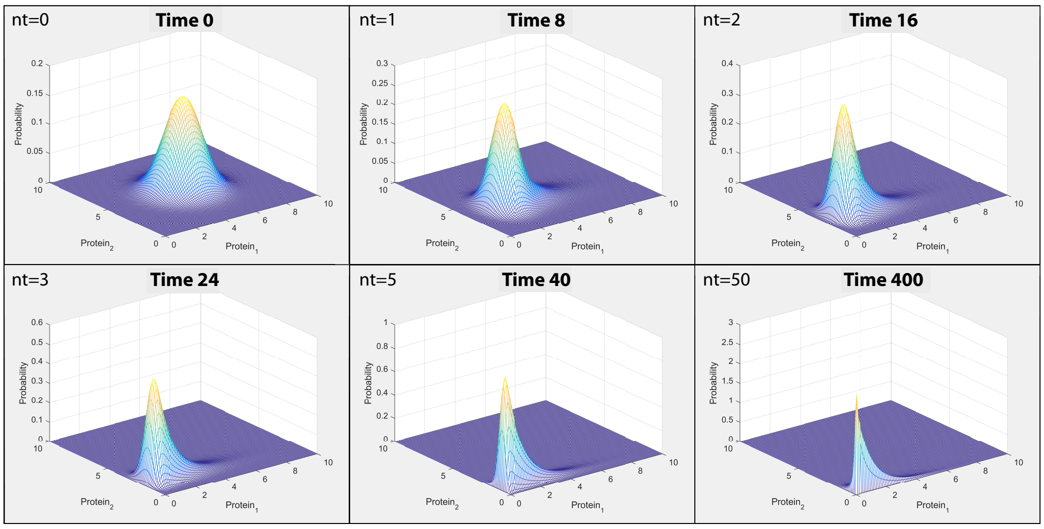


Figure 22: Snapshots of the probability distribution of protein abundance (joint probability distribution of protein 1 and 2) obtained at nt values 0,1,2,3,5, and 50 (the corresponding times are indicated in the title of each subfigure).

We remark here that for a given *nt*, `SELANSI_Plot` generates the plots corresponding to the marginal and joint distributions. In this case `SELANSI_Plot` generates six plots corresponding to the marginal distributions of proteins 1, 2 and 3, and the joint probability distributions 1-2, 1-3 and 2-3.

References

[Pajaro et al.(2017)] Pájaro, M., Alonso, A. A., Otero-Muras, I. and Vázquez, C. (2017). Stochastic Modeling and Numerical Simulation of Gene Regulatory Networks with Protein Bursting. *J. Theor. Biol.* 421: 5170.

[Friedman et al. (2006)] Friedman N, Cai L, Xie XS. Linking stochastic dynamics to population distribution: An analytical framework of gene expression. Phys Rev Lett. 2006;97(16):168302.

[Munteanu et al.(2017)] Munteanu A., Cotterell J., Solé R. V. and Sharpe J. (2014) Design principles of stripe-forming motifs: the role of positive feedback. *Sci. Rep.* 4:5003.

[Gonze(2016)] Gonze, D. and Kaufman, M. (2016) Chemical and enzyme kinetics. *Lecture Notes*