**Assignment 8**

**Validation and Parameter Tuning for Network Models**

**Assignment Outcomes:**

Being able to perform automated parameter tuning of a network model against some empirical data. Empirical data are assumed real world data or data indicating requirements as formulated for the model. The aim is to get simulation of a model close to these data by determining appropriate values for the network characteristics for Connectivity, Aggregation, and Timing represented by **ω**, **c(..)**, **γ**, **η**. The closer, the better.

Note that in this context the word *parameter* can refer to any of these characteristics, it is not limited to what usually are called combination function parameters **γ**.

**Pre-Requisites of Assignment:**

* Book 1, Chapter 14 and its slides should be known.

**Basic Requirements for Parameter Tuning (indicated in later steps of the assignment as well):**

* **Empirical Data:** Empirical data for one or more of the states in the model is assumed to be gathered from real world observations and experiments, or based on requirements formulated for the model, which usually also are based on empirical literature. Usually this is only for a small subset of the set of all states in the model, as for many of the states it may be difficult to get data. And the data usually also will only concern a small subset of the time points that the model uses. Different scientific methods like fMRI, body states (heart rate, skin conductance, ..) can be used, or data can be obtained from social media such as emotion levels extracted from posts. Various case studies can be used to develop empirical data, and it can play an important part to get models closer to reality. So, these data relate to one or some of the states of the model, and to some of the time points: therefore usually you will have:
  + The specific **states** for which you have obtained data. These selected states are given to the software environment in the form of a list or row (a one-dimensional matrix) of some length *M*. For example,

stateselection = [2 4 5]

of length *M* = 3 expresses that the data relate to the three indicated states *X*2, *X*4, and *X*5.

* + The **time points** for which you have empirical data. These time points are usually given to the simulation environment in the form of a list or row (a one-dimensional matrix) of some length *N*. For example,

timepoints=[11.5 16]

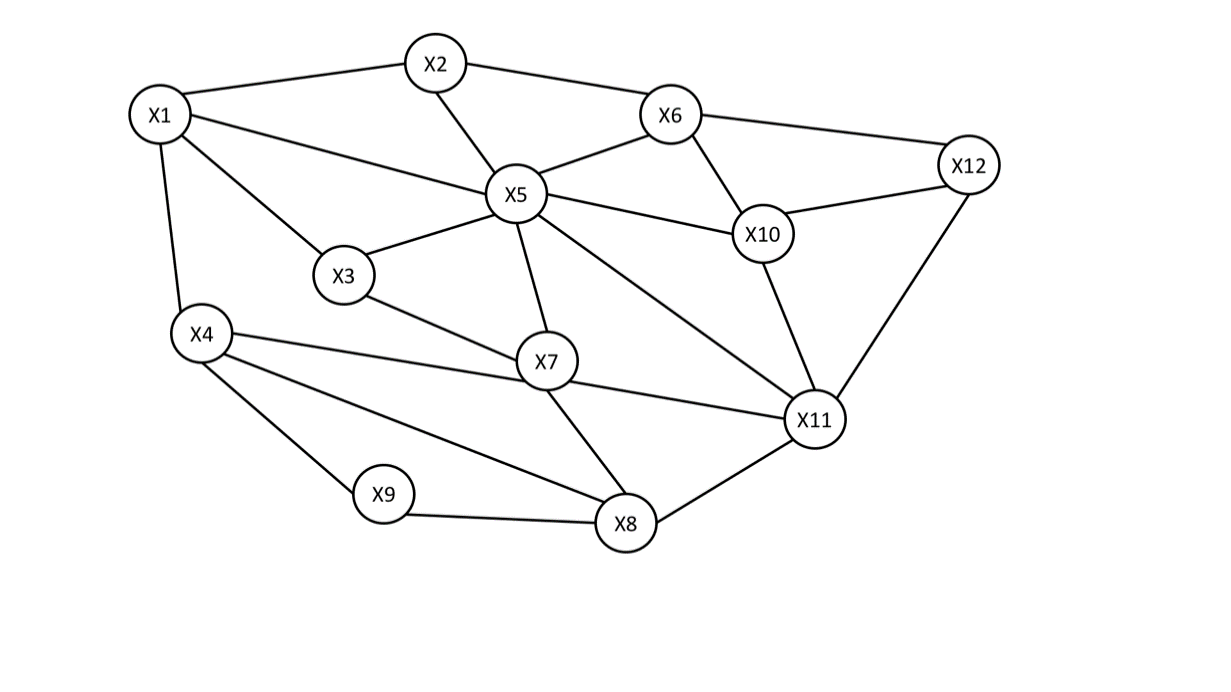
of length *N* = 2 indicates that data is available (only) for the indicated time points 11.5 and 16.

* + The **(empirical)** **data** themselves which have the form of a 2-D matrix with dimensions *M* x *N*, where *M* refers to the length of the state selection list above, and *N* refers to the length the time points list above. For example, here the data will have the form of a 3 x 2 matrix.
* **Choice of parameters**: You also have to indicate the characteristics from the value matrices or initial values list which you want to use as parameters to be tuned. For example, in this assignment it is chosen to address tuning of (some) speed factors and (some) connection weights. But you can extend this to other parameters as well.
* **Parameter intervals:** For each of the chosen parameters you have to specify the **interva**l indicating the range for the considered values for that parameter. For example, the interval [0, 1] or a smaller or bigger interval.

**PART I.**

In this part you will learn to a) simulate the (nonadaptive) example network model and, b) to explore by hand a parameter optimization method. As a first step, you need to simulate the dynamics of a Social Network model for opinion dynamics for 12 persons. The whole network is connected by bi-directional connections as shown in Fig. 1. Use the (value) role matrices and initial values shown in the Appendix, to build the model using the nonadaptive template for Fig. 1 and answer the following.





**Figure 1**- Friendship Graph (Incoming connections)

**Q1. Report the simulation resulting from the model of Fig. 1 using the NOMEnonadaptive template, with the following settings.**

* endtimeofsimulation = 100
* dt = 0.5 (second)

**Q2. Answer the questions a)-h) with the following requirement:**

Here you need to look for one parameter (a speed factor**η**) against some observations. That is, you need to look for the speed factor **η***X*12 of state *X*12 such that the following requirement is fulfilled:

**Requirement**

The values (the curves) of the opinions indicated by nodes *X*11and *X*12should cross each other twice, i.e. at time point *t =* 2 *and t =* 13*.*

Note that this requirement involves two specified time points *t =* 2and *t =* 13 and two specified states *X*11and *X*12 for which the required relations are:

*X*11(2) = *X*12(2) and *X*11(13) = *X*12(13).

1. Check values for**η***X*12by exhaustive search (see Book 1, Chapter 14, Section 14.4) with grain size of 0.05, e.g., like**η***X*12*=* 0.3, 0.35, 0.4, and so on. Take the interval [0, 0.5] as range for the considered parameter values for **η***X*12.
2. Report the whole process by giving an overview of all options that were tried and the resulting deviation for each of them (like in Book 1, Chapter 14, Table 14.3 at page 405); to determine the deviation use the RMSE (Root Mean Square Error) measure, which for this case is the square root of the average of the squares of the deviations found at the two considered time points 2 and 13:

RMSE =

1. What is the new (best) value of the speed factor **η***X*12, and what is the remaining RMSE?
2. Plot the simulated state values for states *X*11and *X*12 by lines in an Excel graph. What are the remaining deviations for the two time points separately? Are these deviations as seen here in the graph in accordance with the RMSE value found in a) and b)?
3. How many parameter values will be checked for **η***X*12 if you set the grain size at 0.01 for the same interval [0, 0.5]? Explain your answer.
4. How does the time step dt influence the simulation and size of the output, if we set dt = 0.1?

**PART II.**

In this part, we aim to learn how to tune in an automated manner any set of network characteristics (in this case the speed factors of all 12 states) in order to get close to some given empirical information. The network model addressed is the same as in Part I. The following is an example of empirical data that is provided for the state values of three selected states *X*1, *X*6, *X*12 and for five time points 2, 5.9, 14.3, 21.1, 29.2, resulting in a 3 x 5 dimensional matrix as shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **2** | **5.9** | **14.3** | **21.1** | **29.2** |
| **1** | 0.90 | 0.82 | 0.72 | 0.67 | 0.63 |
| **6** | 0.54 | 0.52 | 0.49 | 0.47 | 0.46 |
| **12** | 0.03 | 0.13 | 0.24 | 0.29 | 0.33 |

This indicates that te requirements for the network model are that at the given time points, the values of the given states should have the value as indicated in the 3x5 matrix of data points:

**Requirements**

*X*1(2) = 0.90

*X*6(2) = 0.54

*X*12(2) = 0.03

*X*1(5.9) = 0.82

*X*6(5.9) = 0.52

*X*12(5.9) = 0.13

*X*1(14.3) = 0.72

*X*6(14.3) = 0.49

*X*12(14.3) = 0.24

*X*1(12.1) = 0.67

*X*6(21.1) = 0.47

*X*12(21.1) = 0.29

*X*1(29.2) = 0.63

*X*6(29.2) = 0.46

*X*12(29.2) = 0.33

Unlike Q2, this time we will use an automated algorithm from the Matlab optimization tool box to tune some of the characteristics of our network model. The Simulated Annealing algorithm (see Book 1, Chapter 14, Section 14.7) of optimtool, uses an optimisation algorithm selected under ‘Solver’ and is applied to your NOMEtuningnonadaptive tuning template with its name (e.g., NOMEtuningnonadaptivev02) written under ‘Objective function’ to compute a minimal deviation between the simulated and empirical data.

To make it work for the empirical data mentioned above you have to fill in the three items in your NOMEtuningnonadaptive template:

a) timepoints list

b) stateselection list

c) empirical\_data matrix

like:

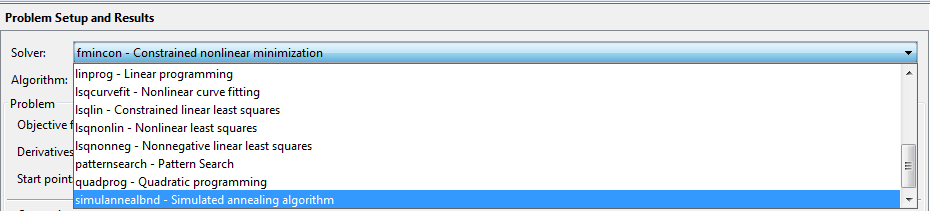
timepoints = [2 5.9 14.3 21.1 29.2]

stateselection = [1 6 12]

empiricaldata = [… (copy the above table here) …]

To run the optimization tool write ‘optimtool’ in the command window of Matlab and press Enter.

Under Solver select the Simulated annealing algorithm as:

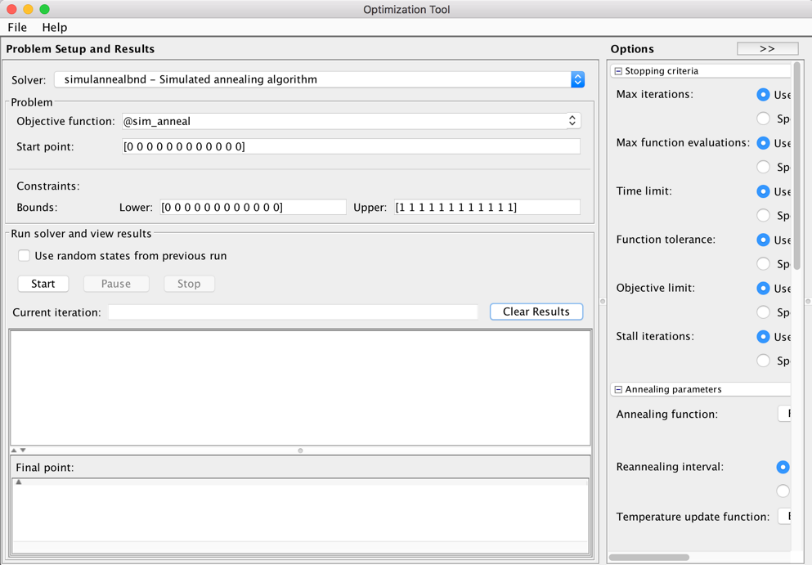


You have to provide options in the window like a) starting point, and intervals for the parameter values b) lower and c) upper bound for values of the parameters, like:

* Objective function = @NOMEtuningnonadaptivev02
* Start point: which parameter values you want to start with
* Lower Bounds = [0 0 0 0 0 0 0 0 0 0 0 0]
* Upper Bounds [1 1 1 1 1 1 1 1 1 1 1 1].

Note that depending on the parameters chosen, the 0 and 1 values can be different, for example for steepness parameters they might be chosen 3 (lower bound) and 15 (upper bound) instead.

Press start (make sure your parameters setting look like as shown in the image below)



**Q.3: Answer the questions below.**

1. Give the tuning matrix **mstuning** where you write the parameters P1 to P12 in the appropriate cells

Then (leaving out the letter P and adding NaN to all empty cells) copy it into the NOMEtuningnonadaptive template between the [ ] of mstuning = [].

1. Run the optimisation from the optimtool interface and report the best set of values found for the chosen parameters (the speed factorsin this case), and the RMSE value in the Final Point Window.
2. Make a graph in Excel like in Fig. 14.10 (lower part) at page 416 in Book 1, displaying the RMSE over the iterations of the tuning process. *Hint: use the data from the Output.xlsx file saved by the NOMEtuningnonadaptive template*
3. Using the best found parameter values, simulate the tuned model in the NOMEnonadaptive template, and make a graph in Excel for the selected states, like in Fig. 14.10 (upper part) at page 416 in Book 1, displaying the simulated values for the states as lines and the data points as dots. *Hint: use the Output.xlsx file written by the NOMEnonadaptive template*
4. Is the RMSE value for the most optimal point found in b) in accordance with what you see in the graph in d)?

**Q.4: Repeat the steps and answer Q3 for the following empirical data given below.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **3.1** | **11.9** | **20.2** | **25.2** | **28** | **30** |
| **3** | 0.92 | 0.85 | 0.81 | 0.79 | 0.78 | 0.77 |
| **5** | 0.75 | 0.74 | 0.73 | 0.71 | 0.71 | 0.70 |
| **9** | 0.40 | 0.55 | 0.61 | 0.64 | 0.65 | 0.65 |
| **10** | 0.25 | 0.28 | 0.30 | 0.31 | 0.32 | 0.32 |
| **11** | 0.18 | 0.32 | 0.39 | 0.41 | 0.42 | 0.43 |

**PART III**

For this part, we simulate the network model described in PART I, by using the combination function **alogistic**(..) for all states instead of scaled sum combination function. Here, the aim is to tune some of the connection weights and a speed factor in order to get close to the given empirical data. The empirical data is a 4x9 matrix, for 4 states and 9 time points. Therefore we will have to run NOMEtuningnonadaptive with the following configuration:

* mcf = [2].
* dt = 0.5
* endtimeofsimulation = 30
* mcfpv:

|  |  |  |
| --- | --- | --- |
| **State** | **mcfpv** | |
| **σ** | **τ** |
| X1 | 1 | 0.14 |
| X2 | 1.4 | 0.14 |
| X3 | 1 | 0.14 |
| X4 | 0.9 | 0.16 |
| X5 | 0.7 | 0.2 |
| X6 | 0.8 | 0.14 |
| X7 | 1.1 | 0.02 |
| X8 | 1 | 0.16 |
| X9 | 1.6 | 0.14 |
| X10 | 1 | 0.17 |
| X11 | 0.8 | 0.14 |
| X12 | 2.2 | 0.15 |

At this point our goal is to tune the incoming connection weights and the speed factor for state *X*5. Examination of **mb** and **mcwv** shows that state *X*5 has seven incoming connections with weights:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *X*5 | 0.5 | 0.4 | 1 | 0.3 | 0.3 | 0.4 | 0.3 |

Our aim is to tune these 7 connection weights of state *X*5 (in this order as parameters P1 to P7) and its speed factor (as parameter P8), such that the simulation results get closest to the empirical data. So, like in previous section, we will use NOMEtuningnonadaptive and we will set the following:

* stateselection = [2 4 7 12]
* timepoints = [2 5 8 11 14 17 20 23 26 30]
* Empirical data = the following table:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **2** | **5** | **8** | **11** | **14** | **17** | **20** | **23** | **26** | **30** |
| **2** | 0.96 | 0.89 | 0.85 | 0.82 | 0.78 | 0.76 | 0.73 | 0.71 | 0.69 | 0.67 |
| **4** | 0.79 | 0.76 | 0.74 | 0.73 | 0.72 | 0.71 | 0.71 | 0.70 | 0.69 | 0.68 |
| **7** | 0.59 | 0.67 | 0.67 | 0.67 | 0.67 | 0.66 | 0.65 | 0.66 | 0.65 | 0.65 |
| **12** | 0.13 | 0.26 | 0.28 | 0.29 | 0.29 | 0.30 | 0.31 | 0.32 | 0.32 | 0.33 |

Copy these data from the table and paste into the “empirical\_data” matrix in Matlab as you did in previous section.

For the optimization, use optimtool:

* Objective function = NOMEtuningnonadaptive
* Give Start point for the 7 connection weights parameters P1 to P7 and the speed factor parameter P8, for example: 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.6
* Put 0 0 0 0 0 0 0 0 as Lower Bounds.
* Put 1 1 1 1 1 1 1 1 in Upper Bounds.
* Press start to activate the tuning

**Q.6: Answer the questions below.**

1. Give the tuning matrices **mcwtuning** and **mstuning.**  
    Next, leaving out the letter P and adding NaN for empy cells, copy them to the NOMEtuning template.
2. Run the optimisation and report the minimal RMSE found.
3. Report the found connection weight values and speed factor for this minimal RMSE.
4. Show the simulation graphs together with the data points using the found weight values and speed factor value.
5. Also make a graph in Excel like in Fig. 14.10 (lower part) at page 416 in Book 1, displaying the RMSE over the iterations of the tuning process.

**Appendix**

**The role matrices for the example network model:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **mb** | | | | | | | |
| **States** | **Incoming connection from** | | | | | | |
| X1 | X2 | X3 | X4 | X5 |  |  |  | |
| X2 | X1 | X5 | X6 |  |  |  |  | |
| X3 | X1 | X5 | X7 |  |  |  |  | |
| X4 | X1 | X7 | X8 | X9 |  |  |  | |
| X5 | X1 | X2 | X3 | X6 | X7 | X10 | X11 | |
| X6 | X2 | X5 | X10 | X12 |  |  |  | |
| X7 | X3 | X4 | X5 | X8 | X11 |  |  | |
| X8 | X4 | X7 | X9 | X11 |  |  |  | |
| X9 | X4 | X8 |  |  |  |  |  | |
| X10 | X5 | X6 | X11 | X12 |  |  |  | |
| X11 | X5 | X7 | X8 | X10 | X12 |  |  | |
| X12 | X6 | X11 | X11 |  |  |  |  | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **mcwv** | | | | | | | |
| **States** | **Incoming Connection Weight** | | | | | | |
| X1 | 0.7 | 0.5 | 0.6 | 0.3 |  |  |  | |
| X2 | 0.6 | 0.5 | 0.6 |  |  |  |  | |
| X3 | 0.8 | 0.4 | 1 |  |  |  |  | |
| X4 | 1 | 0.2 | 0.5 | 0.9 |  |  |  | |
| X5 | 0.5 | 0.4 | 1 | 0.3 | 0.3 | 0.4 | 0.3 | |
| X6 | 0.8 | 0.3 | 0.7 | 0.9 |  |  |  | |
| X7 | 0.9 | 0.4 | 0.1 | 0.4 | 0.2 |  |  | |
| X8 | 0.8 | 0.1 | 1 | 0.3 |  |  |  | |
| X9 | 0.7 | 0.8 |  |  |  |  |  | |
| X10 | 0.7 | 0.7 | 0.1 | 0.8 |  |  |  | |
| X11 | 0.3 | 0.2 | 0.5 | 0.9 | 0.9 |  |  | |
| X12 | 0.2 | 0.7 | 0.2 |  |  |  |  | |

|  |  |
| --- | --- |
| **msv** | |
| **States** | **Speed factor** |
| X1 | 0.1 |
| X2 | 0.1 |
| X3 | 0.1 |
| X4 | 0.05 |
| X5 | 0.05 |
| X6 | 0.05 |
| X7 | 0.5 |
| X8 | 0.1 |
| X9 | 0.1 |
| X10 | 0.01 |
| X11 | 0.1 |
| X12 | 0.5 |

|  |  |
| --- | --- |
| **iv** | |
| **States** | **Initial Value** |
| X1 | 1 |
| X2 | 1 |
| X3 | 0.9 |
| X4 | 0.8 |
| X5 | 0.7 |
| X6 | 0.6 |
| X7 | 0.5 |
| X8 | 0.4 |
| X9 | 0.3 |
| X10 | 0.2 |
| X11 | 0.1 |
| X12 | 0 |

|  |  |
| --- | --- |
| **mcfwv** | |
| **States** | **Combination Function** |
| X1 | 1 |
| X2 | 1 |
| X3 | 1 |
| X4 | 1 |
| X5 | 1 |
| X6 | 1 |
| X7 | 1 |
| X8 | 1 |
| X9 | 1 |
| X10 | 1 |
| X11 | 1 |
| X12 | 1 |

|  |  |
| --- | --- |
| **mcfpv** | |
| **States** | **Combination Function Parameters** |
| X1 | 2.1 |
| X2 | 1.7 |
| X3 | 2.2 |
| X4 | 2.6 |
| X5 | 3.2 |
| X6 | 2.7 |
| X7 | 2.0 |
| X8 | 2.2 |
| X9 | 1.5 |
| X10 | 2.3 |
| X11 | 2.8 |
| X12 | 1.1 |