LSD- Laboratory for Simulation Development

Lecture on *Random Walks*

-

LSD modular code and object structure

Objective of the lecture

This lesson discusses the relation between the model structure, as defined in the LSD browser, and the equations, written in LMM.

Objective of the lecture

This lesson discusses the relation between the model structure, as defined in the LSD browser, and the equations, written in LMM.

In general LSD modelers do not specify in the equations' code the location and nature of the elements (e.g. variables and parameters) used for the computation of the equation code.

Objective of the lecture

This lesson discusses the relation between the model structure, as defined in the LSD browser, and the equations, written in LMM.

In general LSD modelers do not specify in the equations' code the location and nature of the elements (e.g. variables and parameters) used for the computation of the equation code.

The code refers to the elements by their labels, relying on the LSD system to associate, at run time, the actual values required in the equation's computation.

Objective of the lecture

This lesson discusses the relation between the model structure, as defined in the LSD browser, and the equations, written in LMM.

In general LSD modelers do not specify in the equations' code the location and nature of the elements (e.g. variables and parameters) used for the computation of the equation code.

The code refers to the elements by their labels, relying on the LSD system to associate, at run time, the actual values required in the equation's computation.

In the process the lesson will also provide tips on several aspects of LSD interfaces and internal working.

LSD modular features

The separation of data from the code greatly simplifies the writing and re-use of code:

LSD modular features

The separation of data from the code greatly simplifies the writing and re-use of code:

- Avoid mistakes due to wrong referencing, e.g. of writing `myvector[j,i]` instead of `myvector[i,j]`.

LSD modular features

The separation of data from the code greatly simplifies the writing and re-use of code:

- Avoid mistakes due to wrong referencing, e.g. of writing `myvector[j,i]` instead of `myvector[i,j]`.
- Avoid adapting the same computational content to different model structures, such as replacing `myvector[i,j]` with `myvector[i,j,h]`

Random Walk

In this lesson we consider different models sharing an identical computational content (i.e. equations) and will see how different model structures produces intuitively different configurations.

Random Walk

In this lesson we consider different models sharing an identical computational content (i.e. equations) and will see how different model structures produces intuitively different configurations.

We consider a model representing a *random walk* expressed by two equations:

$$\begin{aligned} X_t &= X_{t-1} + RE \\ RE &= U(min, Max) \end{aligned}$$

where $U(m, M)$ generates a (pseudo-)random values as if they were drawn from a uniform function in the range set by the parameters *min* and *Max*. The Uniform function, as many other functions, is provided by the system in the LMM (Use: *Insert Lsd Script*).

Random Walk

Code for variable X.

```
EQUATION("X")
/*
A variable moving as a random walk
*/

v[0]=V("RE");
v[1]=VL("X",1);

v[2]=v[0]+v[1];
RESULT(v[2] )
```

Random Event

Code for variable *RE*.

```
EQUATION("RE")
/*
A random event
*/
v[0]=V("min");
v[1]=V("Max");
v[3]=UNIFORM(v[0],v[1]);
RESULT(v[3] )
```

Structure 1

As initial model structure let's consider the model that may be expressed formally as:

$$\begin{aligned} X_t &= X_{t-1} + RE \\ RE_t &= U(min, Max) \end{aligned}$$

That is, we define a model composed by a single object containing the two variables and the two parameters.

Structure 1

To express such a model structure use the LSD model browser generated with the equations. Then follow these steps:

1. create an object, call it **Obj1**.

Structure 1

To express such a model structure use the LSD model browser generated with the equations. Then follow these steps:

1. create an object, call it **Obj1**.
2. Move the Browser to show the content of the newly created object.

Structure 1

To express such a model structure use the LSD model browser generated with the equations. Then follow these steps:

1. create an object, call it **Obj1**.
2. Move the Browser to show the content of the newly created object.
3. Insert in the object all the elements: variables *X* (1 lag) and *RE* (0 lag); parameters *min* and *Max*.

Structure 1

To express such a model structure use the LSD model browser generated with the equations. Then follow these steps:

1. create an object, call it **Obj1**.
2. Move the Browser to show the content of the newly created object.
3. Insert in the object all the elements: variables *X* (1 lag) and *RE* (0 lag); parameters *min* and *Max*.
4. Using **Data/Initial Values** set $X_0 = 0$ and $min = -10$ and $Max = 10$.

Structure 1

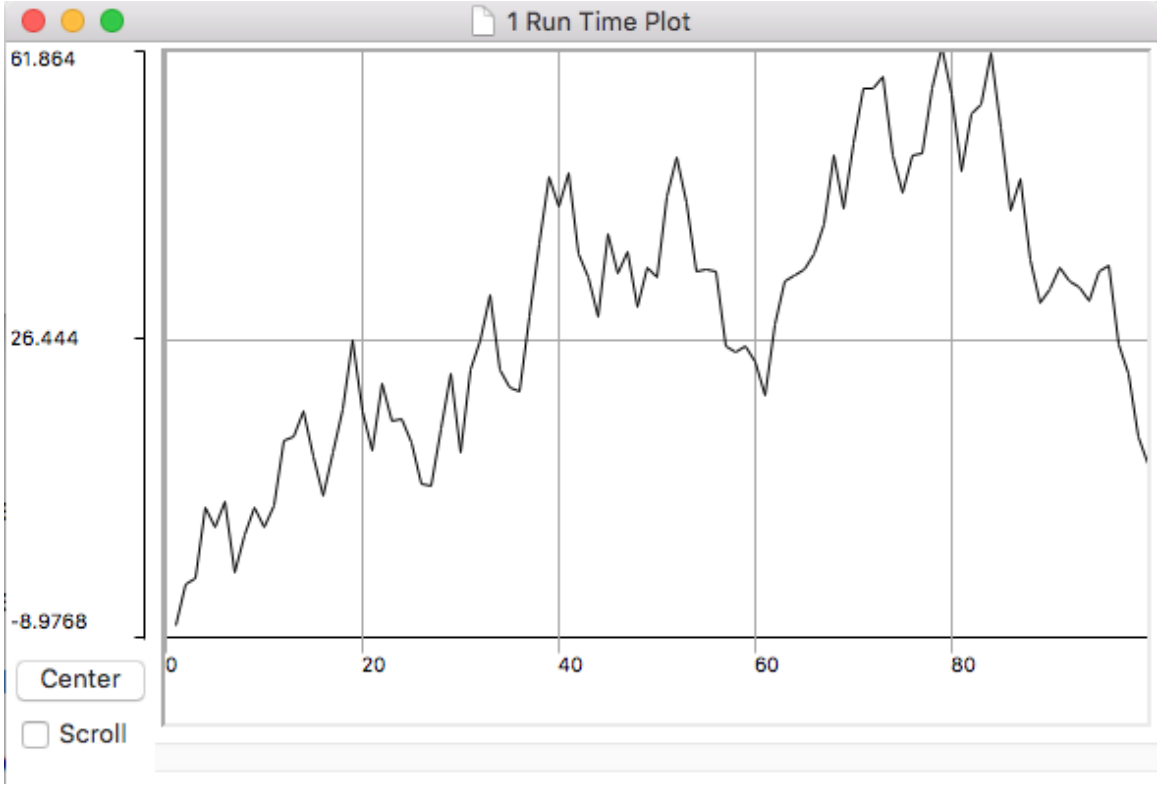
To express such a model structure use the LSD model browser generated with the equations. Then follow these steps:

1. create an object, call it **Obj1**.
2. Move the Browser to show the content of the newly created object.
3. Insert in the object all the elements: variables *X* (1 lag) and *RE* (0 lag); parameters *min* and *Max*.
4. Using **Data/Initial Values** set $X_0 = 0$ and $min = -10$ and $Max = 10$.
5. Double-click on **X** and check on the option **Run Time Plot**.

Beware of using the same spelling in the model configuration as in the equations' code In case of errors correct the equations or the element in the structure. In the latter case double-click on the misspelled element, click on **Properties** and edit the label..

Pseudo random numbers

Running the simulation you will obtain *exactly* the following Run Time Plot window



Pseudo random numbers

Though the pattern generated by the simulation is undoubtedly based on randomness, it clearly cannot be really random, since *every* model generates exactly the same values.

Pseudo random numbers

Though the pattern generated by the simulation is undoubtedly based on randomness, it clearly cannot be really random, since *every* model generates exactly the same values.

Moreover, if you repeat the simulation you generate, again, the same identical result.

Pseudo random numbers

Though the pattern generated by the simulation is undoubtedly based on randomness, it clearly cannot be really random, since *every* model generates exactly the same values.

Moreover, if you repeat the simulation you generate, again, the same identical result.

To replicate the simulation you *cannot* simply use **Run** if you already executed a simulation. The reason is that the values stored in the model (e.g. for X) refer to the last time step of the simulation, which generally differ from those defined to start a simulation. Therefore to repeat a simulation run you need, firstly, the reload a fresh configuration (menu **File**/**Reload**), and then execute a new simulation (menu **Run**/**Run**).

Pseudo random numbers

Computers, as deterministic machines, cannot produce random values, but can simulate random series using algorithms generating sequences of values respecting the requirements for randomness.

Pseudo random numbers

Computers, as deterministic machines, cannot produce random values, but can simulate random series using algorithms generating sequences of values respecting the requirements for randomness.

Users using the same sequence of (pseudo)random values will therefore produce exactly the same results. To generate different results it is sufficient to use a different sequence.

Pseudo random numbers

Computers, as deterministic machines, cannot produce random values, but can simulate random series using algorithms generating sequences of values respecting the requirements for randomness.

Users using the same sequence of (pseudo)random values will therefore produce exactly the same results. To generate different results it is sufficient to use a different sequence.

Before starting a simulation run, use menu **Run/Sim.Settings** and change the value in the entry labeled **Initial seed**. This will produce completely different random values.

A vector of equations

The second model structure we consider is the same structure as before replicated in many different copies.

A vector of equations

The second model structure we consider is the same structure as before replicated in many different copies.

Using the standard formalism we may express the model as follows:

$$\begin{aligned} X_t^i &= X_{t-1}^i + RE_t^i \\ RE_t^i &= U(min^i, Max^i) \end{aligned}$$

where each element is indexed with i.

A vector of equations

To generate the new model follow these steps:

1. Load a fresh configuration of `Sim1` and save it as `Sim2`, to avoid overwriting the first configuration.

A vector of equations

To generate the new model follow these steps:

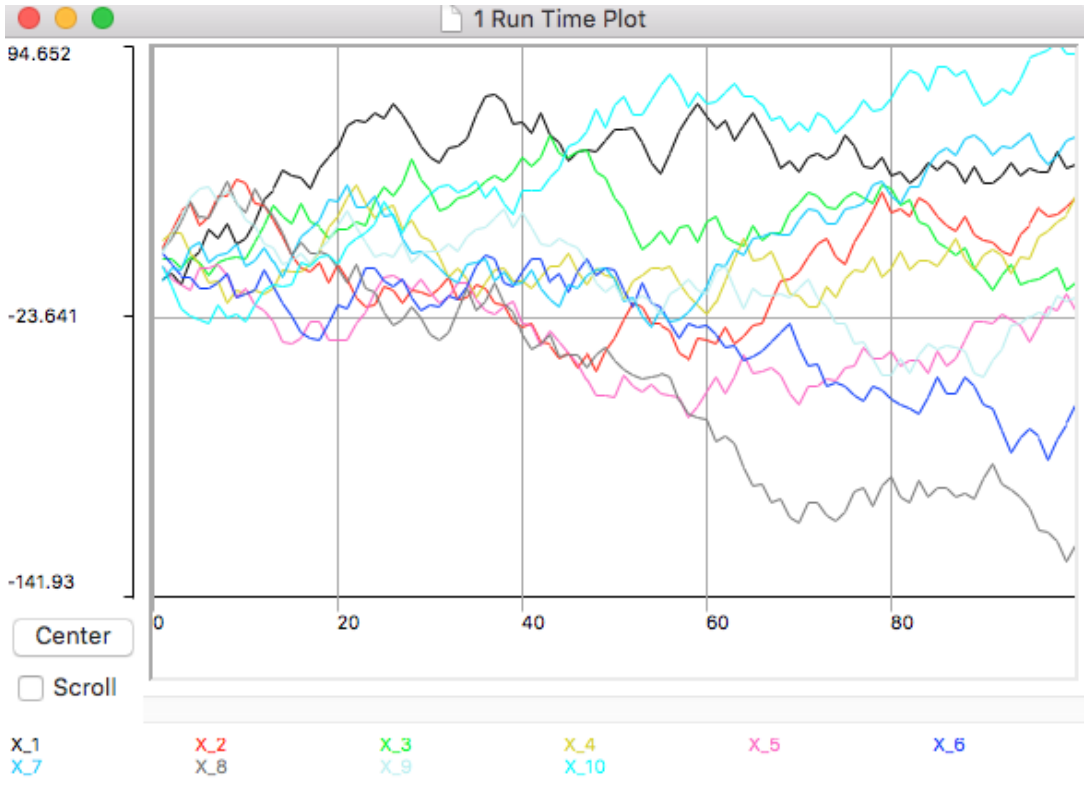
1. Load a fresh configuration of `Sim1` and save it as `Sim2`, to avoid overwriting the first configuration.

2. Use menu **Data/Set Number of Objects** and set to 10 the copies of **Obj1**.

The new model will generate the following result.

Pseudo random numbers

Simulation with 10 copies.



A vector of equations

This configuration of the model exploits the same computational structure, since the code for the equations has not been modified.

A vector of equations

This configuration of the model exploits the same computational structure, since the code for the equations has not been modified.

This is possible because the LSD simulation manager automatically associate each copy of a variable (e.g. X) to a specific copy of the elements required for its computation (in our case RE).

A vector of equations

This configuration of the model exploits the same computational structure, since the code for the equations has not been modified.

This is possible because the LSD simulation manager automatically associate each copy of a variable (e.g. X) to a specific copy of the elements required for its computation (in our case RE).

The modeler needs not to specify the association of the necessary elements in the equation, since LSD automatically deduce that the “correct” elements to use are those in the same object of the variable computed.

A vector of equations

Note that even using the same seed (random sequence) none of the 10 series will be identical to the result from the configuration using a single object.

A vector of equations

Note that even using the same seed (random sequence) none of the 10 series will be identical to the result from the configuration using a single object.

The reason is that the same series of random values will be spread over 10 series. For example, the second random value of the sequence is used, in `Sim1`, as the random event at time $t = 2$, while in `Sim2` it is used for the variable in the second object at time $t = 1$.

A vector of equations with global parameter

In `Sim2` all the 10 series use the same values for the parameters *min* and *Max*. It is a good practice to remove redundancies in simulation models, minimizing the possibility for errors, such as setting different values by mistake.

A vector of equations with global parameter

In `Sim2` all the 10 series use the same values for the parameters *min* and *Max*. It is a good practice to remove redundancies in simulation models, minimizing the possibility for errors, such as setting different values by mistake.

We can therefore consider the implementation of a model where the parameters are the same for all the different series, a model that in formal terms would be expressed as:

$$\begin{aligned} X_t^i &= X_{t-1}^i + RE_t^i \\ RE_t^i &= U(min, Max) \end{aligned}$$

where parameters *min* and *Max* are **not** indexed, meaning they are common to all the copies of *RE*.

A vector of equations with global parameter

To implement this configuration load a fresh configuration of `Sim2` and save it as `Sim3` and then makes the following operations.

1. Move the Browser to show the content of **Obj1**.

A vector of equations with global parameter

To implement this configuration load a fresh configuration of `Sim2` and save it as `Sim3` and then makes the following operations.

1. Move the Browser to show the content of **Obj1**.
2. Use menu entry **Model/Insert new parent** and type the label **Obj0**. This will create a new object “above” **Obj1** descending from **Root**.

A vector of equations with global parameter

To implement this configuration load a fresh configuration of `Sim2` and save it as `Sim3` and then makes the following operations.

1. Move the Browser to show the content of **Obj1**.
2. Use menu entry **Model/Insert new parent** and type the label **Obj0**. This will create a new object “above” **Obj1** descending from **Root**.
3. Move again the Browser to show the content of **Obj1**, since now the Browser points to the newly created object.

A vector of equations with global parameter

To implement this configuration load a fresh configuration of `Sim2` and save it as `Sim3` and then makes the following operations.

1. Move the Browser to show the content of **Obj1**.
2. Use menu entry **Model/Insert new parent** and type the label **Obj0**. This will create a new object “above” **Obj1** descending from **Root**.
3. Move again the Browser to show the content of **Obj1**, since now the Browser points to the newly created object.
4. Double-click on parameter *min*. In the resulting window click on **Properties**. Use option **Move to another Object** and select **Obj0** as object of destination.

A vector of equations with global parameter

To implement this configuration load a fresh configuration of `Sim2` and save it as `Sim3` and then makes the following operations.

1. Move the Browser to show the content of **Obj1**.
2. Use menu entry **Model/Insert new parent** and type the label **Obj0**. This will create a new object “above” **Obj1** descending from **Root**.
3. Move again the Browser to show the content of **Obj1**, since now the Browser points to the newly created object.
4. Double-click on parameter *min*. In the resulting window click on **Properties**. Use option **Move to another Object** and select **Obj0** as object of destination.
5. Repeat the same operation for *Max*.

A vector of equations with global parameter

You can now execute a simulation run with `Sim3`. It generates exactly the same results as using `Sim2`, although by a different computational process managed automatically by the LSD simulation manager.

A vector of equations with global parameter

You can now execute a simulation run with `Sim3`. It generates exactly the same results as using `Sim2`, although by a different computational process managed automatically by the LSD simulation manager.

Differently from the execution of `Sim2`, using `Sim3` the system cannot find the parameters *min* and *Max* in the same object of the copy of *RE* under computation.

A vector of equations with global parameter

You can now execute a simulation run with `Sim3`. It generates exactly the same results as using `Sim2`, although by a different computational process managed automatically by the LSD simulation manager.

Differently from the execution of `Sim2`, using `Sim3` the system cannot find the parameters *min* and *Max* in the same object of the copy of *RE* under computation.

When this is the case, the LSD simulation manager automatically starts a search for the required element. The search starts from the object containing the variable under computation, and then moves, firstly, in its descendants (if any) and then its parent, exploring potentially the whole model structure until the element is found.

Vector of vectors

The automatic search for elements to use in computations is extremely efficient and allows seamlessly to express fairly elaborated computational structures.

Vector of vectors

The automatic search for elements to use in computations is extremely efficient and allows seamlessly to express fairly elaborated computational structures.

As an example, consider the replication of `Sim3` in two distinct settings. Formally, you may express the model as follows:

$$\begin{aligned} X_t^{j,i} &= X_{t-1}^{j,i} + RE_t^{j,i} \\ RE_t^{j,i} &= U(min^j, Max^j) \end{aligned}$$

where j refers to one of two different setting. In practice, the configuration needs to express two independent "branches" formed each by a single **Obj0** containing a set of objects **Obj1**.

Vector of vectors

To express this configuration you need not change the computational content (the equations) but only adapt the model structure:

Vector of vectors

To express this configuration you need not change the computational content (the equations) but only adapt the model structure:

1. Load a fresh configuration `Sim3` and save it as `Sim4`.

Vector of vectors

To express this configuration you need not change the computational content (the equations) but only adapt the model structure:

1. Load a fresh configuration `Sim3` and save it as `Sim4`.
2. Select menu **Data/Set Number of Objects** and define 2 copies of **Obj0**. Upon confirmation of the default options the system generates the two branches with the same number of descending objects **Obj1**.

Vector of vectors

To express this configuration you need not change the computational content (the equations) but only adapt the model structure:

1. Load a fresh configuration `Sim3` and save it as `Sim4`.
2. Select menu **Data/Set Number of Objects** and define 2 copies of **Obj0**. Upon confirmation of the default options the system generates the two branches with the same number of descending objects **Obj1**.
3. Move the browser to show **Obj0**.

Vector of vectors

To express this configuration you need not change the computational content (the equations) but only adapt the model structure:

1. Load a fresh configuration `Sim3` and save it as `Sim4`.
2. Select menu **Data/Set Number of Objects** and define 2 copies of **Obj0**. Upon confirmation of the default options the system generates the two branches with the same number of descending objects **Obj1**.
3. Move the browser to show **Obj0**.
4. Select **Data/Initial Values**. Assign to the parameters *min* and *Max* in the second copy the values -1 and 1, respectively.

Vector of vectors

To express this configuration you need not change the computational content (the equations) but only adapt the model structure:

1. Load a fresh configuration `Sim3` and save it as `Sim4`.
2. Select menu **Data/Set Number of Objects** and define 2 copies of **Obj0**. Upon confirmation of the default options the system generates the two branches with the same number of descending objects **Obj1**.
3. Move the browser to show **Obj0**.
4. Select **Data/Initial Values**. Assign to the parameters *min* and *Max* in the second copy the values -1 and 1, respectively.
5. Double-click on variable *X* in the Browser and check on the option **Save: save these series for later analysis**.

Vector of vectors

We now have effectively two models as in `Sim3` running in parallel. The equation for each *RE* will make use of the appropriate copy of parameters depending on the branch it is part of, thanks to the automatic LSD system of retrieving elements.

Vector of vectors

We now have effectively two models as in `Sim3` running in parallel. The equation for each *RE* will make use of the appropriate copy of parameters depending on the branch it is part of, thanks to the automatic LSD system of retrieving elements.

To verify the results open the module **Data/Analysis of Results**. The series available are marked with two digits, referring respectively to the two layers of objects in the model.

Vector of vectors

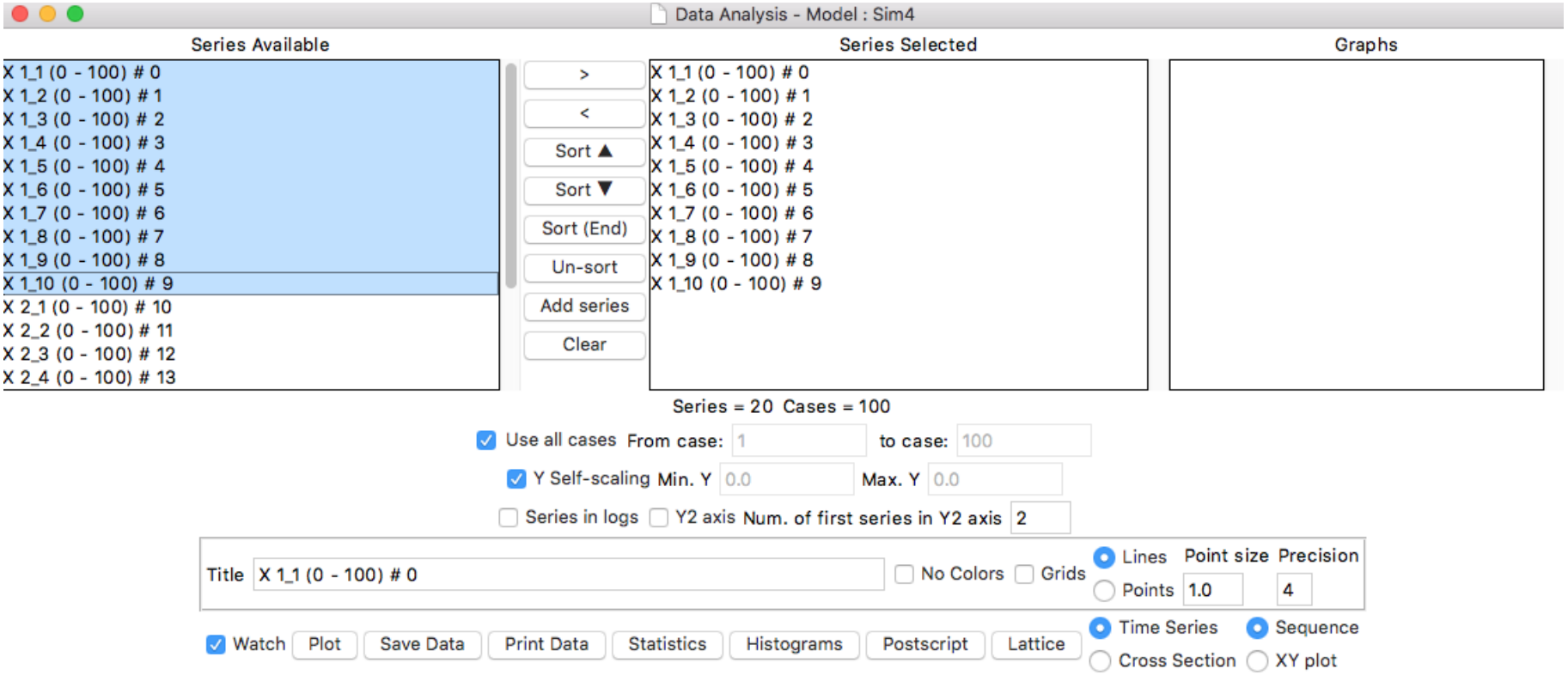
We now have effectively two models as in `Sim3` running in parallel. The equation for each *RE* will make use of the appropriate copy of parameters depending on the branch it is part of, thanks to the automatic LSD system of retrieving elements.

To verify the results open the module **Data/Analysis of Results**. The series available are marked with two digits, referring respectively to the two layers of objects in the model.

Select the series separately from the two branches, and you will see that the same overall dynamics will extend over different ranges, as implied by the different values of parameters.

Pseudo random numbers

Analysis of result for parallel and independent settings.



Functions

We had previously centralized *min* and *Max* since their value is common to the whole branch. Similarly, we can think of centralizing the computation of the random events (*RE*), since these values are generated by the same procedure, and need not to be stored for analysis.

Functions

We had previously centralized *min* and *Max* since their value is common to the whole branch. Similarly, we can think of centralizing the computation of the random events (*RE*), since these values are generated by the same procedure, and need not to be stored for analysis.

The model would then formally become:

$$\begin{aligned} X_t^{j,i} &= X_{t-1}^{j,i} + RE^j \\ RE^j &= U(min^j, Max^j) \end{aligned}$$

where *RE* lacks a time subscript indicating that it does not generate a value at each time step t, but needs to supply fresh values every time it is applied.

Functions

As in all the previous experiments we need not to modify the equations, but only manipulate the model structure.

Functions

As in all the previous experiments we need not to modify the equations, but only manipulate the model structure.

1. Load a fresh configuration `Sim4` and save it as `Sim5`.

Functions

As in all the previous experiments we need not to modify the equations, but only manipulate the model structure.

1. Load a fresh configuration `Sim4` and save it as `Sim5`.
2. Move the Browser to show **Obj1**.

Functions

As in all the previous experiments we need not to modify the equations, but only manipulate the model structure.

1. Load a fresh configuration `Sim4` and save it as `Sim5`.
2. Move the Browser to show **Obj1**.
3. Double-click on **RE**.

Functions

As in all the previous experiments we need not to modify the equations, but only manipulate the model structure.

1. Load a fresh configuration `Sim4` and save it as `Sim5`.
2. Move the Browser to show **Obj1**.
3. Double-click on **RE**.
4. Click on **Properties**. Move the element to **Obj0**.

Functions

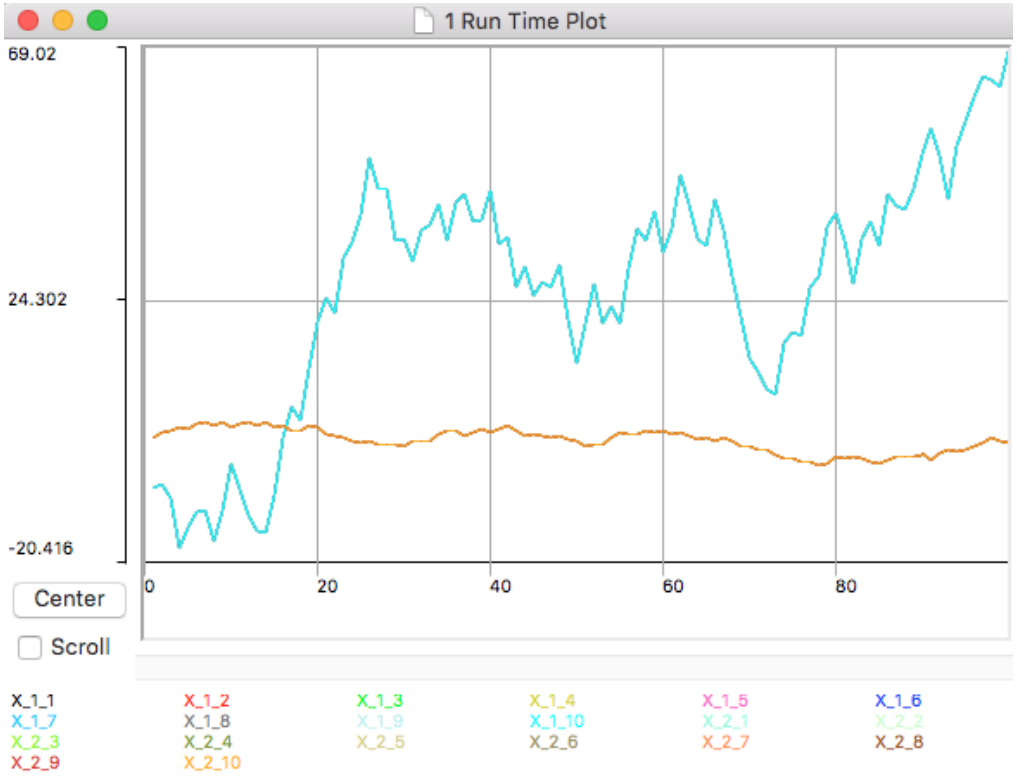
As in all the previous experiments we need not to modify the equations, but only manipulate the model structure.

1. Load a fresh configuration `Sim4` and save it as `Sim5`.
2. Move the Browser to show **Obj1**.
3. Double-click on **RE**.
4. Click on **Properties**. Move the element to **Obj0**.

The new configuration shares the same copy of *RE* for the whole branch. But this generates the **wrong results**. Try to execute a simulation run and it will produce different results from `Sim4`.

Pseudo random numbers

All 10 series for each branch produce *identical* results, and hence overlapping graphical series.



Functions

Moving the variable *RE* to the centralized object for a branch of the model means that it will generate one, and only one, random value at each time step.

Functions

Moving the variable *RE* to the centralized object for a branch of the model means that it will generate one, and only one, random value at each time step.

The different copies of *X* will all be able to reach the value of *RE* stored in their **Obj0**, but this value will be identical for all the copies of *X*, hence generating identical random walks.

Functions

Moving the variable *RE* to the centralized object for a branch of the model means that it will generate one, and only one, random value at each time step.

The different copies of *X* will all be able to reach the value of *RE* stored in their **Obj0**, but this value will be identical for all the copies of *X*, hence generating identical random walks.

If we want that the same computation is replicated every time it is requested, and not only once at each time step, we need to define it as a **function**. A function is like a variable, but its values cannot be stored for analysis since it can produce multiple (or none) values at each time step.

Functions

To fix the `Sim5` double click on variable **RE** and then click on properties. In the resulting window select the option **Function** and confirm.

Functions

To fix the `Sim5` double click on variable **RE** and then click on properties. In the resulting window select the option **Function** and confirm.

Replicating now the simulation will generate exactly the same results as using `Sim4`.

Large models

As a final exercise we use the model for large scale testing. That is, we expand the dimension of the model so as to generate massive amounts of data in order to appreciate aggregate statistical properties.

Large models

As a final exercise we use the model for large scale testing. That is, we expand the dimension of the model so as to generate massive amounts of data in order to appreciate aggregate statistical properties.

We will run the model including 100,000 distinct instances of random walks over 1,000 time steps. LSD can execute simulations under a stripped down mode, renouncing any graphical representation, and hence being as fast as a pure C++ compiled code.

Large models

1. Load a fresh configuration `Sim5` and save it as `Sim6`.

Large models

1. Load a fresh configuration `Sim5` and save it as `Sim6`.
2. Open **Data/Set Number of Objects**. Set to 1 the number of **Obj0**. Set to 100,000 the number of **Obj1**.

Large models

1. Load a fresh configuration `Sim5` and save it as `Sim6`.
2. Open **Data/Set Number of Objects**. Set to 1 the number of **Obj0**. Set to 100,000 the number of **Obj1**.
3. Move the Browser to show **Obj1**.

Large models

1. Load a fresh configuration `Sim5` and save it as `Sim6`.
2. Open **Data/Set Number of Objects**. Set to 1 the number of **Obj0**. Set to 100,000 the number of **Obj1**.
3. Move the Browser to show **Obj1**.
4. Double-click on **X** and deselect the option **Run Time Plot**. Ensure that the option for saving is on.

Large models

1. Load a fresh configuration `Sim5` and save it as `Sim6`.
2. Open **Data/Set Number of Objects**. Set to 1 the number of **Obj0**. Set to 100,000 the number of **Obj1**.
3. Move the Browser to show **Obj1**.
4. Double-click on **X** and deselect the option **Run Time Plot**. Ensure that the option for saving is on.
5. In menu **Run/Sim.Settings** set the number of **Simulation Steps** to 1,000.

Large models

1. Load a fresh configuration `Sim5` and save it as `Sim6`.
2. Open **Data/Set Number of Objects**. Set to 1 the number of **Obj0**. Set to 100,000 the number of **Obj1**.
3. Move the Browser to show **Obj1**.
4. Double-click on **X** and deselect the option **Run Time Plot**. Ensure that the option for saving is on.
5. In menu **Run/Sim.Settings** set the number of **Simulation Steps** to 1,000.
6. Run the simulation. It will take a minute or two to compute the 10^8 computations. Press on **Fast** in the **Log** window to skip the messages on time steps completed, further reducing the execution time.

Large models

At the end of the simulation open the module **Data/Analysis of Results**. Select all the series available and place them in the **Series Selected** box (**Tip**: right-click on one of the **X** series and see the **Help** for details).

Large models

At the end of the simulation open the module **Data/Analysis of Results**. Select all the series available and place them in the **Series Selected** box (**Tip**: right-click on one of the **X** series and see the **Help** for details).

Don't try to plot the time series graph, this will freeze the graphical engine of the LSD windowing system for several minutes. Rather, select the option **Cross Section** (bottom right), and click on the button **Histograms**.

Large models

The resulting box will ask for a few options. Accept all the default values, or click on **Help** for details.

Large models

The resulting box will ask for a few options. Accept all the default values, or click on **Help** for details.

The graph produced on confirmation represents a frequency histogram of evenly spaces classes of values between the minimum and the maximum values from all the series at the last time step of the simulation run.

Large models

The resulting box will ask for a few options. Accept all the default values, or click on **Help** for details.

The graph produced on confirmation represents a frequency histogram of evenly spaces classes of values between the minimum and the maximum values from all the series at the last time step of the simulation run.

As expected, the graph produces a neat Gaussian function, centered on the expected value of 0.

Pseudo random numbers

Histogram of end-of-simulation frequency classes.