Introduction to Lsd

Major features and plan

Marco VALENTE[1,2]

[1]LEM, S. Anna School of Advanced Studies, Pisa

[2]University of L'Aquila

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Plan

- **Introduction**: goals and plan of the course

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Plan

- **Introduction**: goals and plan of the course
- **Definitions**: a normal form of a simulation model.

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Plan

- **Introduction**: goals and plan of the course
- **Definitions**: a normal form of a simulation model.
- **Introduction to Lsd**: equations, structures and configurations of models.

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Plan

- **Introduction**: goals and plan of the course
- **Definitions**: a normal form of a simulation model.
- **Introduction to Lsd**: equations, structures and configurations of models.
- **Tutorials**: implementation of increasingly complex example models.

Outline

Objective: learning how to use simulations implemented in Lsd to make research in Economics

Plan

- **Introduction**: goals and plan of the course
- **Definitions**: a normal form of a simulation model.
- **Introduction to Lsd**: equations, structures and configurations of models.
- **Tutorials**: implementation of increasingly complex example models.
- **Methodology**: How to develop and assess scientific knowledge by means of simulation models.

Why using simulations?

The methodology for using simulations in social sciences is hotly debated. But is also highly relevant, since it is at the base of all technical and theoretical issues.

Why using simulations?

The methodology for using simulations in social sciences is hotly debated. But is also highly relevant, since it is at the base of all technical and theoretical issues.

At the end of the course we will provide a detailed definition of the methodological protocol required by simulations, both in social sciences and in “hard” fields. However, we anticipate the fundamental point of the proposed protocol.

Methodological assumptions

1. Research models must **not** replicate the reality, necessarily.

Methodological assumptions

1. Research models must **not** replicate the reality, necessarily.
2. Research models must **explain** reality.

Methodological assumptions

A few comments are required:

1. We talk about research simulation models. Other applications may differ.

Methodological assumptions

A few comments are required:

1. We talk about research simulation models. Other applications may differ.
2. The pivotal concept is what we mean by *reality*. Economic events are ill-defined, lacking direct measures, and never replicate through history.

Methodological assumptions

A few comments are required:

1. We talk about research simulation models. Other applications may differ.
2. The pivotal concept is what we mean by *reality*. Economic events are ill-defined, lacking direct measures, and never replicate through history.
3. *Explaining* needs a formal definition, too. We will provide one, showing that it is both intuitive and rigourous.

Methodological assumptions

We will sustain that the scientific use of simulation models consist in two logical steps:

- Find explanations of interesting simulated events, as if analysing the record of a virtual history

Methodological assumptions

We will sustain that the scientific use of simulation models consist in two logical steps:

- Find explanations of interesting simulated events, as if analysing the record of a virtual history
- Evaluate whether the same explanations, *mutatis mutandis*, apply to the real world cases, too.

Simulation models vs programs

Simulation models and simulation programs are distinct entities.

Simulation models vs programs

Simulation models and simulation programs are distinct entities.

Simulation models as abstract, logical constructs, much like a theorem or a mathematical system of equations. They are defined by logical/mathematical operations located in time.

Simulation models vs programs

Simulation models and simulation programs are distinct entities.

Simulation models as abstract, logical constructs, much like a theorem or a mathematical system of equations. They are defined by logical/mathematical operations located in time.

A simulation model may be implemented with a variety of tools: mental experiments, pencil and paper, Excel spreadsheet, etc. Complex models require computer programs, but, in these cases, the most challenging task is to develop the software tools necessary to make sense of massive amounts of data.

Topics of the course

Using a standard programming language the most difficult task is *not* the coding of the model. Rather it is the coding of ancillary tools necessary to declare the model's elements, assign initial values, export results, etc.

Topics of the course

Using a standard programming language the most difficult task is *not* the coding of the model. Rather it is the coding of ancillary tools necessary to declare the model's elements, assign initial values, export results, etc.

Using LSD, conversely, the modeller focuses only on the model, and the system automatically generates professional tools to control and access any aspect of the model.

Topics of the course

During the course we will approach the following topics:

- **Design**: decide what the model should be like to contribute to a research project.

Topics of the course

During the course we will approach the following topics:

- **Design**: decide what the model should be like to contribute to a research project.
- **Implementation**: turning an abstract idea into a working simulation program.

Topics of the course

During the course we will approach the following topics:

- **Design**: decide what the model should be like to contribute to a research project.
- **Implementation**: turning an abstract idea into a working simulation program.
- **Interpretation**: extracting knowledge from simulation models.

Topics of the course

In the rest of this introductory talk we will address the following issues:

1. Define a normal form for simulation models.

Topics of the course

In the rest of this introductory talk we will address the following issues:

1. Define a normal form for simulation models.
2. Describe the LSD overall structure.

Topics of the course

In the rest of this introductory talk we will address the following issues:

1. Define a normal form for simulation models.
2. Describe the LSD overall structure.
3. Indicate the steps required to use a simulation model.

Definition of simulation models

A simulation model is defined independently from the medium implementing it. We need a definition of simulation model such that we can perfectly identify the results produced by running the model.

Definition of simulation models

A simulation model is defined independently from the medium implementing it. We need a definition of simulation model such that we can perfectly identify the results produced by running the model.

Simulation model: generic definition of how X_t is computed:

$X_t = f(X_{t-k}, Y_t, \alpha)$

Definition of simulation models

A simulation model is defined independently from the medium implementing it. We need a definition of simulation model such that we can perfectly identify the results produced by running the model.

Simulation model: generic definition of how X_t is computed:
$X_t = f(X_{t-k}, Y_t, \alpha)$

Simulation run: sequence(s) of values across simulation time steps. $\{X_1, X_2, ..., X_t, ..., X_T\}$

Definition of simulation models

Notice that we choose to refer to *time driven* simulation models, as opposed to *event driven* models.
The two styles of modelling are equivalent, since one can be turned into the other.

Definition of simulation models

The definition of simulation model we provide is meant to satisfy two requirements:

1. **Univocal**. The definition must be unambiguous, making impossible to include implementations of the same model but generating different results.

Definition of simulation models

The definition of simulation model we provide is meant to satisfy two requirements:

1. **Univocal**. The definition must be unambiguous, making impossible to include implementations of the same model but generating different results.
2. **User friendly**. It must be as close as possible to (one of) the way(s) people usually refer to models in natural language.

Definition of simulation models

In the following we list the elements composing a simulation model, providing definitions such that no ambiguity is left about the simulation results produced with the model.

Variables

Variables are labels, or symbols, that at ***each time step*** are associated to one and only one numerical value.
The numerical value of a variable is computed executing an *equation*, defined as any computational elaboration of other values.
$X_t = f_X(...)$

Parameters

Parameters are labels associated to numerical values.
Parameters do not change value of their own accord.

α

Functions

Functions are, like variables, numerical values computed as result of an equation. However, the values generated by functions are not associated to time steps, but are computed on request during the execution of other equations.

$X = f(...)$

Objects

In almost any case a model is designed to contain many copies, or instances, of variables, parameters and functions. They share the same label and properties (i.e. equations) but are distinguished and independent from one another.
In mathematical format we normally use vectors, using the same label with different indexes:

$$\vec{X} = \{X^1, X^2, ..., X^i, ..., X^n\}$$

Objects

However, in “hierarchical” models, vector-based representations are extremely annoying. For example, consider a variable supposed to refer to a firm (among many), operating in a market (among many) which is part of a country (among many). Such a variable would need three indexes to be accessed, for the firm, market and country it refers to.

Moreover, troubles emerge when we deal with dynamic models. Adding a new firm requires to extend all the vectors referring to this entity.

Programming languages have developed a more powerful concept, that includes vectors, but it is far more general: objects.

Objects

Objects are containers, storing together different types of elements that are, somehow, forming an identifiable unit. Programming using objects is far simpler (and less dangerous) than using vectors. Moreover, objects are particularly useful for simulations, since the unit representing an object can easily be associated to the real-world entity that the model refers to.

Definition of simulation models

Object-based representations are equivalent to a matrix-based representation.

		Object-based			
		$ObOne^1$	$ObOne^2$	...	$ObOne^N$
Vector-based	$\vec{X}$	X^1	X^2	...	X^N
	$\vec{Y}$	Y^1	Y^2	...	Y^N
	$\vec{\alpha}$	α^1	α^2	...	α^N
	$Ob\vec{T}wo$	$ObTwo^1$	$ObTwo^2$	...	$ObTwo^N$

Definition of simulation models

Object-based representations are equivalent to a matrix-based representation.

		Object-based			
		$ObOne^1$	$ObOne^2$	...	$ObOne^N$
Vector-based	$\vec{X}$	X^1	X^2	...	X^N
	$\vec{Y}$	Y^1	Y^2	...	Y^N
	$\vec{\alpha}$	α^1	α^2	...	α^N
	$Ob\vec{T}wo$	$ObTwo^1$	$ObTwo^2$	...	$ObTwo^N$

Object-based representations are far more flexible than vectors, easily expressing, for example, the equivalent of nested matrices and matrices with different number of rows in each column.

Model Structure

In summary, we can call the **structure** of a model the set of the following elements:

1. **Variables**. Symbols associated to a single value at each time step, computed according to a specified equation.
2. **Parameters**. Symbols associated to values not changing of their own accord.
3. **Functions**. Symbols providing values computed by an equation on request by other equations (independently from the time).
4. **Objects**. Units containing a set of other elements.

Model Configuration

The structure of a model is an abstract description of the model, defining how the values of a generic time step t can be computed on the base of the values inherited from time step $t - 1$.

Model Configuration

The structure of a model is an abstract description of the model, defining how the values of a generic time step t can be computed on the base of the values inherited from time step $t - 1$.

The structure of a model is still an ambiguous description, since the same structure will, in general, produce different results depending on the numerical values assigned at $t = 0$. Let's see which numerical values for each type of element can affect the results.

Model Configuration

The first numerical value is the **number of objects**, since it also determines the number of other elements.

Notice that the assignment of objects' numbers may be quite elaborated, with different number of entities for different "branches" of the model.

Model Configuration

Obviously, every parameter must be assigned an initial value. But also, possibly, some variables and functions may require one or more values.

Model Configuration

Obviously, every parameter must be assigned an initial value.
But also, possibly, some variables and functions may require one or more values.
Consider the equation
$X_t = Y_{t-1} + \alpha$

Model Configuration

Obviously, every parameter must be assigned an initial value. But also, possibly, some variables and functions may require one or more values.
Consider the equation
$X_t = Y_{t-1} + \alpha$
At time $t = 1$, the very first step of the simulation, the equation becomes:
$X_1 = Y_0 + \alpha$
Y_0 cannot be produced by the model, since 1 is the first time step. Consequently, the modeller that must assign to Y a *lagged* (or past) value for Y as part of the initialization of the model.

Model Configuration

An equation may also require more than one lag. Consider, for example, the following equation:

$X_t = Y_{t-3} + \alpha$

For the first 3 time steps the model requires the values of Y_{-2}, Y_{-1} and Y_0, and therefore the user must assign three lagged values to Y in order to avoid ambiguities.

Model Configuration

An equation may also require more than one lag. Consider, for example, the following equation:
$X_t = Y_{t-3} + \alpha$
For the first 3 time steps the model requires the values of Y_{-2}, Y_{-1} and Y_0, and therefore the user must assign three lagged values to Y in order to avoid ambiguities.

Functions also may require “lagged” values, though they do not refer to previous time steps, but to previous calls, or executions, of the function’s equation.

Definition of simulation models

A simulation model is therefore perfectly defined once we describe in detail the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model

Definition of simulation models

A simulation model is therefore perfectly defined once we describe in detail the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model
- **Configuration:**
  - **Structure**: list of variables, parameters, functions and objects

Definition of simulation models

A simulation model is therefore perfectly defined once we describe in detail the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model
- **Configuration:**
  - **Structure**: list of variables, parameters, functions and objects
  - **Initialization**: number of objects, values for parameters, lagged values for variables and functions

Definition of simulation models

A simulation model is therefore perfectly defined once we describe in detail the following elements:

- **Equations**: set of routines to compute values for each variable and function in the model
- **Configuration:**
  - **Structure**: list of variables, parameters, functions and objects
  - **Initialization**: number of objects, values for parameters, lagged values for variables and functions
  - **Sim. settings**: num. of time steps, num. of simulation runs, pseudo-random sequences, visualization and saving options.

LSD simulation models

LSD allows users to generate a simulation program defining only the elements of a simulation model according to the format proposed above.

Furthermore, LSD provides programs complete with interfaces, debugger, graphics etc. allowing the full analysis of the results.

Using models in practice

Though we will discuss the methodology of simulations for research at the end of the course, it is worth mentioning practical suggestions that we will derive. That is, we briefly review what type of activities a researcher is required to perform.

These are somehow similar to that of a programmer, though the researcher has a further requirement: understand how its model works and present its content (model and result) to scrutiny.

Using models in practice

1. **Purpose**: decide what is phenomenon to explain (NO DESCRIPTION!).

Using models in practice

1. **Purpose**: decide what is phenomenon to explain (NO DESCRIPTION!).
2. **Design**: define the list and nature of the elements part of the model.

Using models in practice

1. **Purpose**: decide what is phenomenon to explain (NO DESCRIPTION!).
2. **Design**: define the list and nature of the elements part of the model.
3. **Implement the structure**: write the code to implement the model in the chosen language.

Using models in practice

1. **Purpose**: decide what is phenomenon to explain (NO DESCRIPTION!).
2. **Design**: define the list and nature of the elements part of the model.
3. **Implement the structure**: write the code to implement the model in the chosen language.
4. **Implement the configuration**: assign initialization to the structure.

Using models in practice

1. **Purpose**: decide what is phenomenon to explain (NO DESCRIPTION!).
2. **Design**: define the list and nature of the elements part of the model.
3. **Implement the structure**: write the code to implement the model in the chosen language.
4. **Implement the configuration**: assign initialization to the structure.
5. **Interpret/document**: individuate the relevant results produced by the model, and explain them convincingly.

Using models in practice

1. **Purpose**: decide what is phenomenon to explain (NO DESCRIPTION!).
2. **Design**: define the list and nature of the elements part of the model.
3. **Implement the structure**: write the code to implement the model in the chosen language.
4. **Implement the configuration**: assign initialization to the structure.
5. **Interpret/document**: individuate the relevant results produced by the model, and explain them convincingly.
6. **Revise and extend**: modify 1, 2, 3, 4 and 5, and expand the model when satisfied.

Using models in practice

The success of a project depends crucially on the adoption of a **gradual** approach. A model should initiallly implement a tiny part of the elements one eventually wants to place in the model. Only when the implemented part is tested and its behaviour fully understood, then a new module may be added.

Writing in one shot the whole model, is a sure recipe of failure. Most likely, the model will be so full of bugs, that fixing them will be impossible. Furthermore, even in the case that the bugs are removed, the model behaviour will have no chance of being interpreted and understood.

LSD goals

A *simulation model*, even though perfectly defined, is still a long way from being *simulation program*, as much stating a theorem is quite different from proving it.

LSD goals

A *simulation model*, even though perfectly defined, is still a long way from being *simulation program*, as much stating a theorem is quite different from proving it.
A simulation program requires a long list of computationally complicated *technical* code. Such code does not concern the model directly, but make possible for the model to run on the computer, and its results to be readable by humans.

LSD goals

The technical code consists of any piece of code required to execute a simulation and analyse its result, though not being strictly part of the model. For example, this code must ensure that:

- The simulation time step is correctly initialized and updated.

LSD goals

The technical code consists of any piece of code required to execute a simulation and analyse its result, though not being strictly part of the model. For example, this code must ensure that:

- The simulation time step is correctly initialized and updated.
- The variables are computed at the correct moment within the time step.

LSD goals

The technical code consists of any piece of code required to execute a simulation and analyse its result, though not being strictly part of the model. For example, this code must ensure that:

- The simulation time step is correctly initialized and updated.
- The variables are computed at the correct moment within the time step.
- Users can supply the appropriate type of initialization.

LSD goals

The technical code consists of any piece of code required to execute a simulation and analyse its result, though not being strictly part of the model. For example, this code must ensure that:

- The simulation time step is correctly initialized and updated.
- The variables are computed at the correct moment within the time step.
- Users can supply the appropriate type of initialization.
- Results are collected and presented in suitable ways.

LSD goals

The technical code consists of any piece of code required to execute a simulation and analyse its result, though not being strictly part of the model. For example, this code must ensure that:

- The simulation time step is correctly initialized and updated.
- The variables are computed at the correct moment within the time step.
- Users can supply the appropriate type of initialization.
- Results are collected and presented in suitable ways.
- Errors are caught before crashing the program, and adequate information is provided.

LSD goals

The technical code consists of any piece of code required to execute a simulation and analyse its result, though not being strictly part of the model. For example, this code must ensure that:

- The simulation time step is correctly initialized and updated.
- The variables are computed at the correct moment within the time step.
- Users can supply the appropriate type of initialization.
- Results are collected and presented in suitable ways.
- Errors are caught before crashing the program, and adequate information is provided.
- Unexpected results can be reproduced, investigated and clarified.
- ...

LSD goals

LSD's aims is to allow users to provide only a *simulation model*'s configuration and equations. The system automatically generates any technical code required for a professional *simulation program*.

LSD goals

LSD's aims is to allow users to provide only a *simulation model*'s configuration and equations. The system automatically generates any technical code required for a professional *simulation program*.

LSD users define independently the different parts of the model, and the system assembles them in a program generating fast and flexible simulation runs, or detailed error messages.

LSD goals

LSD's aims is to allow users to provide only a *simulation model*'s configuration and equations. The system automatically generates any technical code required for a professional *simulation program*.

LSD users define independently the different parts of the model, and the system assembles them in a program generating fast and flexible simulation runs, or detailed error messages.

Crucially, the elements in the model are considered as separated modules, that the system assembles as required. It is therefore extremely simple to develop, assess and revise a model implemented with LSD.

LSD's equations major features

Coding a LSD models can be done only by writing the code for the model's equations. The closest metaphor to a LSD model is a system of discrete equations, one for each variable.

LSD's equations major features

Coding a LSD models can be done only by writing the code for the model's equations. The closest metaphor to a LSD model is a system of discrete equations, one for each variable.

The equations of a LSD models are defined as independent chunks of code, with references to the model made only via the labels of required elements.

LSD's equations major features

Coding a LSD models can be done only by writing the code for the model's equations. The closest metaphor to a LSD model is a system of discrete equations, one for each variable.

The equations of a LSD models are defined as independent chunks of code, with references to the model made only via the labels of required elements.

At run time the *Lsd Simulation Manager* re-arranges the equations calling them as necessary to perform the necessary computation.

LSD's equations major features

LSD is endowed with an **automatic scheduling**. The modeller needs not to consider when a particular equation is executed within a time step. The *LSM* automatically (and constantly) re-arranges the order as implied by the equations.
For example, consider the following "model":

$$\begin{aligned} X_t &= F_X(Y_t) \\ Y_t &= F_Y(X_{t-1}) \end{aligned}$$

The system interprets the equations so that Y must be updated before X. Changing order of execution entails simply to change the index for the lags within the equations' code.

LSD's equations major features

In general a model contains many copies of each element, say many Market's may contain each many Firm's etc. In a simulation run it must be ensured that each copy of a variable makes use of the "correct" copies of the elements required in its equations. As, for example, $Q^i = f(K^i, p^j)$.

LSD's equations major features

In general a model contains many copies of each element, say many Market's may contain each many Firm's etc. In a simulation run it must be ensured that each copy of a variable makes use of the "correct" copies of the elements required in its equations. As, for example, $Q^i = f(K^i, p^j)$.

LSD is an object-based language, and therefore it is impossible (besides impractical) to use indexes. By default, LSD finds automatically the "correct" copy of the elements to use in an equation, using a **automatic tagging** system to retrieve required elements.

LSD's equations major features

Note that both features make completely effortless the modification of a model, the extension adding new elements, and the re-use of its parts.

LSD's equations major features

Note that both features make completely effortless the modification of a model, the extension adding new elements, and the re-use of its parts.

In practice, a LSD model is made of individual modules (the equations) which are related only by the labels of the elements required for the computation.

LSD's equations major features

The language for expressing the equations can be considered as composed by multiple layers.

LSD's equations major features

The language for expressing the equations can be considered as composed by multiple layers.

Lacking any specific indication the system makes use of the internal scheduler and retriever. This system covers most of the cases and makes the code extremely simple and readable.

LSD's equations major features

The language for expressing the equations can be considered as composed by multiple layers.

Lacking any specific indication the system makes use of the internal scheduler and retriever. This system covers most of the cases and makes the code extremely simple and readable.

A second layer entails the use of specific LSD commands overruling the default behaviour to express frequently used operations. E.g. "pick at random one of the elements in this set".

LSD's equations major features

The language for expressing the equations can be considered as composed by multiple layers.

Lacking any specific indication the system makes use of the internal scheduler and retriever. This system covers most of the cases and makes the code extremely simple and readable.

A second layer entails the use of specific LSD commands overruling the default behaviour to express frequently used operations. E.g. "pick at random one of the elements in this set".

Finally, LSD is built on standard C++ so that any command or external library compatible with GNU C++ can be used within the model.

LSD technical components

LSD is distributed with a companion program called *Lsd Model Manager* which performs the following tasks:

1. Organize the projects and manage the required files.

LSD technical components

LSD is distributed with a companion program called *Lsd Model Manager* which performs the following tasks:

1. Organize the projects and manage the required files.
2. Assist in the writing of the equations.

LSD technical components

LSD is distributed with a companion program called *Lsd Model Manager* which performs the following tasks:

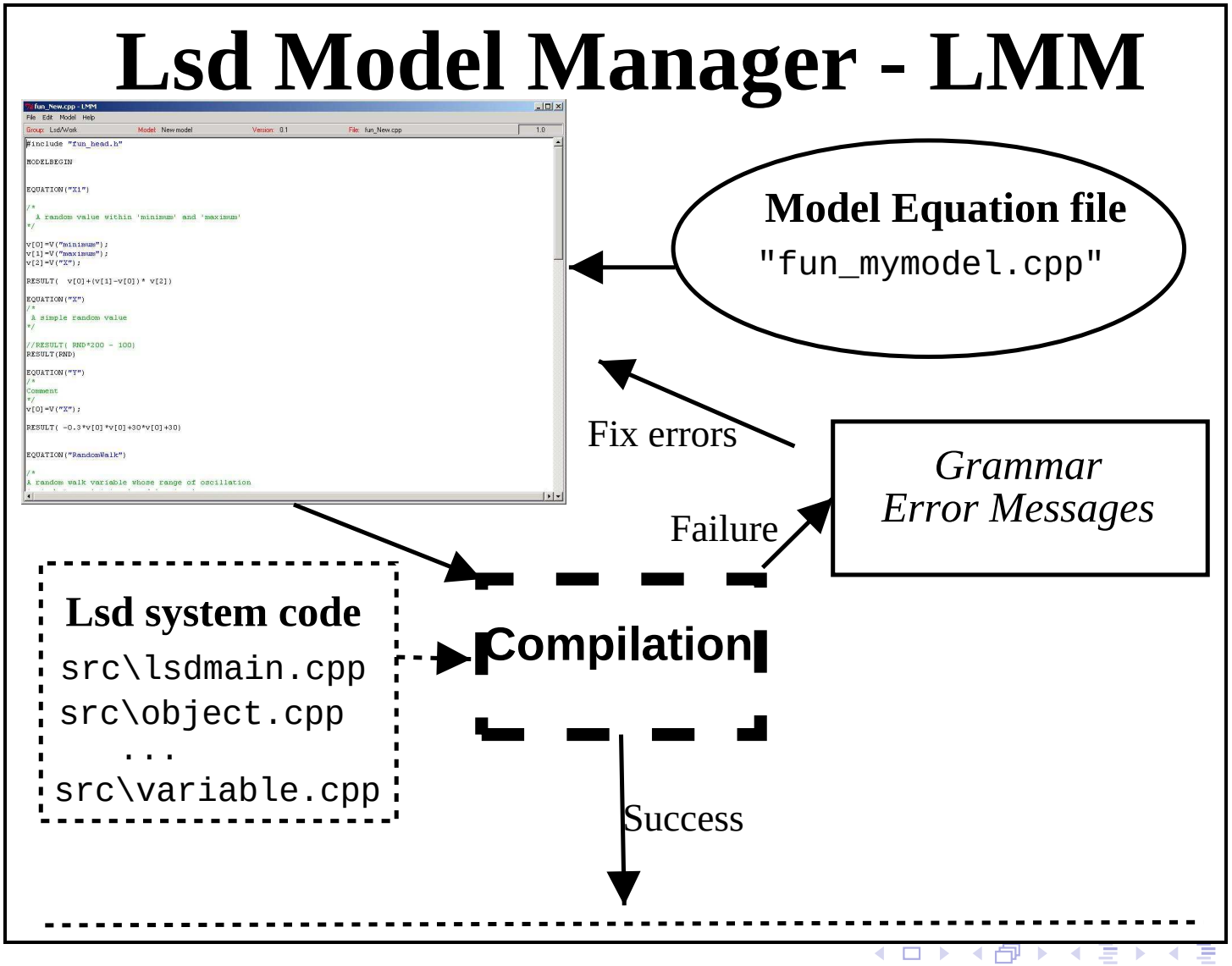
1. Organize the projects and manage the required files.
2. Assist in the writing of the equations.
3. Manage the compilation process.

LSD technical components

LSD is distributed with a companion program called *Lsd Model Manager* which performs the following tasks:

1. Organize the projects and manage the required files.
2. Assist in the writing of the equations.
3. Manage the compilation process.
4. Provide indications on grammar errors in the equations' code.

LSD technical components



LSD technical components

On success LMM generates an executable called *LSD Model Program* embodying the equations of the model and offering every operation concerning the model:

1. Define, save and load model configurations.

LSD technical components

On success LMM generates an executable called *LSD Model Program* embodying the equations of the model and offering every operation concerning the model:

1. Define, save and load model configurations.
2. Run single or multiple simulations.

LSD technical components

On success LMM generates an executable called *LSD Model Program* embodying the equations of the model and offering every operation concerning the model:

1. Define, save and load model configurations.
2. Run single or multiple simulations.
3. Analyse the results, at run-time and at the end of the simulation.

LSD technical components

On success LMM generates an executable called *LSD Model Program* embodying the equations of the model and offering every operation concerning the model:

1. Define, save and load model configurations.
2. Run single or multiple simulations.
3. Analyse the results, at run-time and at the end of the simulation.
4. Investigate the model state before, during or after a run.

LSD technical components

On success LMM generates an executable called *LSD Model Program* embodying the equations of the model and offering every operation concerning the model:

1. Define, save and load model configurations.
2. Run single or multiple simulations.
3. Analyse the results, at run-time and at the end of the simulation.
4. Investigate the model state before, during or after a run.
5. Catch and report on errors at run time, keeping data produced until the stop.

LSD technical components

On success LMM generates an executable called *LSD Model Program* embodying the equations of the model and offering every operation concerning the model:

1. Define, save and load model configurations.
2. Run single or multiple simulations.
3. Analyse the results, at run-time and at the end of the simulation.
4. Investigate the model state before, during or after a run.
5. Catch and report on errors at run time, keeping data produced until the stop.
6. Document a model with its own interfaces, or exporting reports in HTML or Latex format

LSD technical components