#### **System Dynamics**

- In this class, we shall introduce a new *Dymola* library designed to help us with modeling *population dynamics* and similar problems that are described as *pure mass flows*.
- The system dynamics methodology had been introduced in the late sixties by *J.W.Forrester* as a tool for organizing partial knowledge about models of *systems from soft-sciences*, such as *biology*, *bio-medicine*, and *macro-economy*.



#### **Table of Contents**

- From bond graphs to system dynamics
- Exponential growth
- Levels and rates
- Gilpin model
- Laundry list
- System dynamics modeling recipe
- Larch bud moth
- <u>Influenza</u>



# From Bond Graphs to System Dynamics I

• Remember how we have been modeling convective flows (mass flows) using bond graphs.



December 13, 2012



# From Bond Graphs to System Dynamics II

• If we weren't interested, where the energy came from, we could leave the pumps out, and replace them by flow sources.



$$e_1 = e(t)$$
  $f_1 = f(t)$   $der(q) = f_1 - f_2$   $f_2 = f(t)$   $e_2 = e(t)$   
 $e = C(q)$ 

December 13, 2012

© Prof. Dr. François E. Cellier



# From Bond Graphs to System Dynamics III

• If we furthermore aren't interested in the efforts at all, the effort equations can be left out, and all the bonds become activated, i.e., turn into signal flows.







# From Bond Graphs to System Dynamics IV

• *Forrester* introduced a graphical representation tailored to exactly this situation.



December 13, 2012

© Prof. Dr. François E. Cellier



# **Exponential Growth Model**

• Let us start by implementing the simple exponential growth model that had been introduced two classes ago:



File Edit Simulation Plot Animation Commands Window Help

#### Levels

 Levels represent the state variables of the system dynamics modeling methodology.



December 13, 2012

#### © Prof. Dr. François E. Cellier



- 🗆 ×

\_ 8 ×

#### **Rates I**

• *Rates* represent the *state derivatives* of the *system dynamics modeling methodology*.





#### **Rates II**

• For convenience, rates with multiple additive inputs are also provided.



Notice that this is an *OutPort* that was drawn in reverse for optical reasons.

Whereas mass can be envisaged to flow from the source to the level at a rate controlled by the rate valve, the flow of information, however, is from the rate to the source.



#### **Rates III**

• Also available are rate gauges with built-in limiters, e.g. valves that let flow pass in one direction only.





#### Levels II

• Also available are levels with overflow protection and with protection against pumping from an already empty storage.





# The Gilpin Model I

• We now have everything that we need to create a *system dynamics* version of the *Gilpin model*.



#### © Prof. Dr. François E. Cellier

Mathematical Modeling of Physical Systems

#### **The Gilpin Model II**

Experiment Setup	Experiment Setup
General Iranslation Output Debug Compiler Realtime	General Iranslation Dutput Debug Compiler Realtime
	General Iranslation Qutput Debug Compiler Beakime         Format         Textual data format         Double precision         Store         State variables         Input variables         Output variables         Protected variables         Protected variables         Protected variables         Durbut variables         Protected variables         Durbut variables         Durbut selection         Equidistant time grid         Store variables at events         Mumber of Fervaluations         Store variables at events         Mumber of GRID points         If advalues at events         Mumber of f (successful) steps         Store variables at events
	Number of step events : 0 Minimum integration stepsize : 1.07e-010 Maximum integration stepsize : 0.781
	Maximum integration order : 5 Calling terminal section "dsfinal.txt" creating (final states)
OK Store in model Cancel	OK Store in model Cancel

December 13, 2012

© Prof. Dr. François E. Cellier



### **The Gilpin Model III**



December 13, 2012

© Prof. Dr. François E. Cellier

## The Gilpin Model IV

• What have we gained by representing the *Gilpin model* in the *system dynamics formalism*?

Absolutely nothing!

*Systems dynamics* has been invented as a tool for graphically capturing *partial knowledge* about a poorly understood system, generating a model that can be successively augmented, as new knowledge becomes available.

 $\Rightarrow$ 

*System dynamics* is not particularly useful for implementing an already fully developed model, such as the *Gilpin model*.



#### **The Laundry List**

• *System dynamics*, just like any other decent modeling methodology, always starts out with the set of variables to be used in the model, especially the *levels* and the *rates*.

Levels	Rat Inflows	es Outflows
Population	Birth Rate	Death Rate
Money	Income	Expenses
Frustration	Stress	Affection
Love	Affection	Frustration
Tumor Cells	Infection	Treatment
Inventory on Stock	Shipments	Sales
Knowledge	Learning	Forgetting

Birth Rate:	<ul> <li>Population</li> <li>Material Standard of Living</li> <li>Food Quality</li> <li>Food Quantity</li> <li>Education</li> <li>Contraceptives</li> <li>Religious Beliefs</li> </ul>	For each of the <i>rates</i> , a list of the most influential variables is created. This is called a <i>laundry list</i> .
-------------	--	--



# **System Dynamics Modeling Recipe**

- We always start out by choosing the *levels* to be included in the model. These must be quantities that can be *accumulated*.
- For each *level*, we define one or several *additive inflows* and one or several *additive outflows*. These are the *rates*.
- For each *rate*, we define a *laundry list* comprised of the set of *most influential factors*.
- For each of these *factors*, equations are generated that relate these factors back to the *levels*, the *rates*, and *other factors*. These equations are created by using as much *physical insight* as possible. *Algebraic loops* are to be avoided.



# **The Larch Bud Moth Model I**

- We shall now attempt to come up with a better model for describing the *population dynamics* of the *larch bud moth* making use of the *systems dynamics* modeling methodology.
- We stipulate that the *insect/tree interaction* is the dominant influencing factor regulating the population dynamics of the larch bud moth. We assume that the influence of the *parasites* is of second order small, and can be neglected.
- We shall try to come up with a model based primarily on physical insight.
- Curve fitting shall be used, but limited to local measurable properties only.



# The Larch Bud Moth Model II

- The insects breed only once per year. They lay their eggs onto the branches of the larch trees in August. The eggs then remain in a state of extended embryonic diapause until the following spring.
- Hence it makes sense to use a *discrete-time model*, i.e., describe the population dynamics of the larch bud moth by a set of *difference equations*.
- To this end, a *discrete level* model is being offered as part of *Dymola's SystemDynamics library*.



#### **Discrete Levels**

• *Discrete levels* are another form of *state variables* for the *system dynamics modeling methodology*.





The *when-clause* is executed for the first time at *time=h*, then once every *h time units*.

December 13, 2012

© Prof. Dr. François E. Cellier



# The Larch Bud Moth Model III

• There are two discrete state variables: the *number of eggs*, and the *raw fiber*, which the insect larvae use as their food.



Since both the eggs and the needle mass are being replaced every year, the old eggs and old fibers simply all go away. Thus, the outflows are equal to the levels.



# The Larch Bud Moth Model IV

- During the fall, the eggs are preyed upon by several species of *Acarina* and *Dermaptera*.
- During the winter, the eggs are parasitized by a species of *Trichogramma*.
- The surviving eggs are ready for hatching in June.
- The overall effects of *winter mortality* can be summarized as a simple constant.

Small\_larvae = (1.0 – winter\_mortality) · Eggs

winter\_mortality = 57.28%



#### **Gain Factors**

• Gain factors are modeled as follows:

Gain-SystemDynamics.Gain - [Icon]       Else Eds Smiddom Bot Animation Commands Window Help       Image: Imag	Gain - SystemDynamics.Gain - [Modelica Text]	
k = k Components ElyunDynamics.Gain ↓ ↓ k = k name Modeling V Smudation		?×
	Comment Parameters k 1.0 - wintermortality + OK Info Cancel	

December 13, 2012



# **The Larch Bud Moth Model V**

- Whether or not the small larvae survive, depends heavily on luck or mishap. For example, if the branch on which the eggs have been laid dies during the winter, the young larvae have no food.
- This is called the *incoincidence factor*.

Large\_larvae = (1.0 – incoincidence) · Small\_larvae

• However, the incoincidence factor is not constant. It depends heavily on the raw fiber contents of the biomass of the tree.



Start Presentation

#### **Linear Regression**

• A *linear regression model* was used to determine the *incoincidence factor* from measurements:

 $incoincidence = 0.05112 \cdot rawfiber - 0.17932$ 

Sincar - SystemDynamicsLinear - [Icon]       IX         ■ Ele Edt Smidston Bot Animation Commands Window Help       IX         Image: Smidston Bot Animation Commands Window Help       IX         Packages       IX         Image: Smidston Bot Animation Commands Window Help       IX         Image	□ Linear - SystemDynamics.Linear - [Modelica Text]       □ □ ×         □ File Edit Simulation Plot Animation Commands Window Help       □ □ ×         □ □ ×       □ □ ×         □ □ ×       □ □ ×         □ □ ×       □ □ ×         □ □ ×       □ ∞         □ □ ×       □ ∞         □ □ ×       □ ∞         □ □ ×       □ ∞         □ ∞
y A	<pre>Packages Packages Gain Gain Gain y = m*u + b;</pre>
	Line: 1 Modeling

December 13, 2012

# The Larch Bud Moth Model VI

• Hence we can model the population of large larvae using two "linear regression" models in series, followed by a two-input product model:

incoincidence = 0.05112 · rawfiber – 0.17932 coincidence = (-1.0) · incoincidence + 1.0 Large\_larvae = coincidence · Small\_larvae

December 13, 2012



# The Larch Bud Moth Model VII

• In similar ways, we can model the entire *egg life cycle*:

Small\_larvae = (1.0 - winter\_mortality) · Eggs Large\_larvae = (1.0 - incoincidence) · Small\_larvae Insects = (1.0 - starvation) · (1.0 - weakening) · Large\_larvae Females = sex\_ratio · Insects New\_eggs = fecundity · Females

• The animal population is further decimated, either because the *large larvae* don't have enough food (*starvation*), or because they were sick already before (*physiological weakening*).



# The Larch Bud Moth Model VIII

- Notice that we essentially created a *physical model* of the entire *egg life cycle*.
- Curve fitting is only used locally to identify linear regression models of measurable physical quantities.

incoincidence = 0.05112 · rawfiber – 0.17932 weakening = 0.124017 · rawfiber – 1.435284 fecundity = -18.475457 · rawfiber + 356.72636

• The *sex ratio* is constant, whereas *starvation* depends on *food demand* and *tree foliage (food supply)*:

*sex\_ratio* = 0.44

starvation =  $f_1$  (foliage, food\_demand)

food\_demand = 0.005472 · Large\_larvae

December 13, 2012

© Prof. Dr. François E. Cellier



### **The Larch Bud Moth Model IX**

• In similar ways, we can model the *life cycle of the trees*:

*New\_rawfiber = recruitment · rawfiber* 

where:

 $recruitment = f_{2}(defoliation, rawfiber)$   $defoliation = f_{3}(foliage, food_demand, starvation)$   $foliage = specific_foliage \cdot nbr_trees$   $specific_foliage = -2.25933 \cdot rawfiber + 67.38939$   $nbr_trees = 511147$ 

December 13, 2012



#### **The Larch Bud Moth Model X**



# The Larch Bud Moth Model XI

- We started out by deciding on the *formalism* itself, i.e., we decided that we were going to use discrete rather than continuous levels.
- We then identified the *number of levels*, i.e., the number of quantities that can be independently accumulated. In our case, we decided on using the eggs and the raw fiber as the two state variables.
- We then identified *life cycles* for the two levels.
- We limited *curve fitting* to identifying locally verifiable relationships between variables, which in our case turned out to be linear regression models.
- This provided us with an *almost complete model*. There are only three *laundry lists*:  $f_1$ ,  $f_2$ , and  $f_3$  that require further analysis.



## **Functional Relationships**

• The *SystemDynamics library* offers three *partial blocks* for capturing functional relationships, one for functions with a single input, one for functions with two inputs, and one for functions with three inputs.





# **The Larch Bud Moth Model XII**

• This block was used to create a model for the starvation:





#### **The Larch Bud Moth Model XIII**



# **The Larch Bud Moth Model XIV**

• The *equation window* of the main model looks as follows:



• Notice that *no global curve fitting* was ever applied to this model.


# **The Larch Bud Moth Model XV**

• We are now ready to compile and simulate the model.

🖶 Messages - Dymola 📃 🔲	X Experiment Setup	🖨 Messages - Dymola
Syntax Error         Translation         Dialog Error         Simulation           Translation of LarchBudMoth.LBK         DAE having 113 scalar unknowns and 113 scalar equations.         STATISTICS           Driginal Model         Number of components: 34         Variables: 143         Constants: 0           Parameters: 30 (88 scalars)         Unknowns: 113 (113 scalars)         Differentiated variables: 0           Equations: 113         Nontrivial: 91         Differentiated variables: 0	General     Iranslation     Qutput     Debug     Compiler     Bealtime       Experiment	Syntax Error       Translation       Dialog Error       Simulation         Log-file of program ./dymosim       (generated: Wed Jan 31 16:02:05 2007)         dymosim started       "dsin.txt" loading (dymosim input file)         "dsin.txt" loading (dymosim input file)         "LEM.mat" creating (simulation result file)         Integration started at T = 0 using integration method DASSL         (DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))         Integration terminated successically as 1 - 10         CPU-time for integration       : 0.06 seconds
Translated Model Constants: 0 Free parameters: 83 scalars Parameter depending: 5 scalars Inputs: 0 Outputs: 0 Continuous time states: 0 Time-varying variables: 40 scalars Alias variables: 73 scalars Assumed default initial conditions: 2 LogDefaultInitialConditions=true; gives more information Number of mixed real/discrete systems of equations: 0 Sizes of linear systems of equations: {} Sizes after manipulation of the linear systems: {} Sizes after manipulation of the nonlinear systems: {} Sizes after manipulation of the nonlinear systems: {} Number of numerical Jacobians: 0 Finished // experiment StopTime=26 Finished	Integration Algorithm Dassl Tolerance 0.0001 Fixed Integrator Step 0 OK Store in model Cancel	CPU-time for one GRID interval. 0.12 milli seconds Number of result points : 550 Number of GRID points : 501 Number of (successful) steps : 412 Number of f-evaluations : 790 Number of H-evaluations : 936 Number of Jacobian-evaluations: 378 Number of Jacobian-evaluations: 378 Number of (u) time events : 26 Number of (u) time events : 0 Number of state events : 0 Number of state events : 0 Number of step events : 0 Minimum integration stepsize : 4e-006 Maximum integration order : 1 Calling terminal section "dsfinal.txt" creating (final states)

December 13, 2012



#### **The Larch Bud Moth Model XVI**



• The model reproduces the *observed limit cycle behavior* of the larch bud moth population beautifully, both in terms of *amplitude* and *frequency*.

Since <u>no global curve fitting</u> was applied to the model, this is an indication that the important relationships were modeled correctly.



# The Influenza Model I

- Let us create yet one more model today, describing the spreading of an *influenza epidemic* in a *community of 10,000 souls*.
- Since influenza can be contracted at any time, we shall use *continuous levels* for this model.
- People, once *infected* with this particular variant of the disease, take four weeks before they come down with any symptoms. This is called the *incubation period*. Yet, they are already *contagious* during that period.
- Once they are *sick*, they remain sick for two weeks.
- Once they have recovered from the disease, they are *immune* to this particular stem for 26 weeks. Thereafter, they may contract the disease anew.



# The Influenza Model II

- Let us now choose our *level variables*.
- We can identify four types of people:
  - Non-infected people.
  - Infected healthy people.
  - Sick people.
  - Immune people.
- We shall use these four variables as our levels.
- Clearly, there are only three *state variables*, since the sum of the four is always 10,000, i.e., we can always compute the fourth from the other three, but as long as we don't insist that we must choose our initial conditions independently, this doesn't cause any problem.



# **The Influenza Model III**

- The four *level variables* are placed in a loop.
- They are fed by four *rate variables*:
  - Contraction rate.
  - Incubation rate.
  - Recovery rate.
  - Re-activation rate.
- We shall use these four variables as our rates.





### **The Influenza Model IV**

- The *contraction rate* can be computed as the product of the percentage of contagious population multiplied with the number of contacts per week multiplied with the probability of contracting the disease on a single contact.
- The *incubation rate* can be computed as the quotient of the infected population and the time to breakdown.
- The *recovery rate* can be computed as the quotient of the sick population and the duration of the symptoms.
- The *re-activation rate* can be computed as the quotient of the immune population and the immune period.



#### **The Influenza Model V**



3

# The Influenza Model VI

• We want to take into account that the numbers in each level are supposed to be *integers*.

Reference     Second       Control     Second       Cond	Rate_equation - Influenza.Rate_equation - [Modelica Text]         File       Edit         Simulation       Plot         Animation       Commands         Window       Help
C Gan C Leas GOucotterd C Leadudtath C Standard D Oradian D Dataton	
y = f(u1, u2)	Packages block Rate_equation extends SystemDynamics.Interfaces.Nonlin_2;
	Flu Flu Kace_equacion;
B Modero V Smillion	Line: 1 Hodeling Simulation

By default, the *integer function* will schedule events in *Dymola*. As this is not useful here, we use the *noEvent clause* to prevent these unnecessary event iterations from happening.

December 13, 2012



# The Influenza Model VII

• One additional problem concerning the contraction rate needs to be taken care of.



• It could theoretically happen that the model tries to infect more people than the total uninfected population. This must be prevented.



# The Influenza Model VIII

• The equation window of the main model looks as follows:



• At *time = 8 weeks*, we introduce one single influenza patient into the general population of our community.



### **The Influenza Model IX**

• The model can now be compiled.

Amessages - Dymola	Experiment Setup	🖨 Messages - Dymola
Syntax Error Translation Dialog Error Simulation	General Iranslation Output Debug Compiler Realtime	Syntax Error Translation Dialog Error Simulation
Translation of Influenza.Flu: DAE having 103 scalar unknowns and 103 scalar equations. STATISTICS Original Model Number of components: 30 Variables: 133 Constants: 0 Parameters: 30 (30 scalars) Unknowns: 103 (103 scalars) Differentiated variables: 4 scalars Equations: 104	Experiment Name Flu Simulation interval Start time 0 Stop time 52 Output interval C Interval length 0 C Number of intervals 500	Log-file of program ./dymosim (generated: Wed Jan 31 16:32:15 2007) dymosim started "dsin.txt" loading (dymosim input file) "Flu.mat" creating (simulation result file) Integration started at T = 0 using integration method DASSL (DAE multi-step solver (dass1/dass1rt of Petzold modified by Dynasim)) Warning message from dymosim At time T = 3.753153e+001 in current integration interval T_interval = 3.744000e+001 3.754400e+001 a large amount of work has been expended
Nontrivial : 88 Translated Model Constants: 0 Free parameters: 26 scalars Parameter depending: 4 scalars Inputs: 0 Outputs: 0 Continuous time states: 4 scalars Time-varying valables: 16 scalars Alias variables: 80 scalars Number of mixed real discrete systems of equations: 0 Sizes of linear systems of equations: {} Sizes after manipulation of the linear systems: {} Sizes after manipulation of the nonlinear systems: {} Number of numerical Jacobians: 0 Finished Finished	Integration Algorithm Dassi Tolerance 0.0001 Fixed Integrator Step 0 OK Store in model Cancel	<pre>(about 500 steps) in the integrator. Probably the communciation interval is too large or the system is stiff. Integration terminated successfully at 1 = 52 CPU-time for integration 1 = 0.42 milli-seconds CPU-time for one GRID interv 1 = 0.342 milli-seconds Number of result points 1 = 503 Number of GRID points 2 = 503 Number of F-evaluations 2 = 7292 Number of F-evaluations 2 = 7804 Number of H-evaluations 2 = 7804 Number of Jacobian-evaluations 9818 Number of (model) time events 1 0 Number of state events 1 1 Number of state events 2 0 Miniber of state events 2 0 Miniber of state events 2 0 Miniber of state events 3 0 Miniber of sta</pre>



### **The Influenza Model X**

• Simulation results:



• Within only 6 weeks, almost the entire population of the community has been infected with the disease. The epidemiology of the disease is just as bad as that of the chain letter!



#### Conclusions

- We have now improved our skills for developing *softscience models* in an organized fashion that stays as close to the underlying physics as can be done.
- *System dynamics* was introduced as a methodology that allows us to formulate and capture partial knowledge about any soft-science application, knowledge that can be refined as more information becomes available.
- *Systems dynamics* is the most widely used modeling methodology in all of soft sciences. Tens of thousands of scientists have embraced and used this methodology in their modeling endeavors.



## References

- Cellier, F.E. (1991), <u>Continuous System Modeling</u>, Springer-Verlag, New York, <u>Chapter 11</u>.
- Fischlin, A. and W. Baltensweiler (1979), "Systems analysis of the larch bud moth system, Part 1: The larch – larch bud moth relationship," *Mitteilungen der Schweiz. Entomologischen Gesellschaft*, **52**, pp. 273-289.
- Cellier, F.E. (2007), <u>*The Dymola System</u>* <u>*Dynamics Library*</u>, Version 2.0.</u>

