System Dynamics using Modelica

Introduction to System Dynamics

- Simulations in biology, ecology or economy:

  highly complex systems

  causal relationships often unclear

  simulation as testbed to check new ideas

  mathematical formulation
  - usually as differential equation
  - often too abstract for users
  - functional relationships often based on empiric data (tables, plots)

  graphical modeling with system dynamics diagramms
  - emphasis on causal relationships
  - only very few general blocks
  - mathematical relations are hidden, equations are defined as parameters
  - several commercial modeling and simulation environments, e. g. Stella, Vensim, Simile

  invented by Jay Forrester around 1955

- Basic building blocks of system dynamics diagrams:

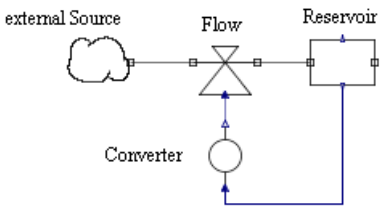
  Reservoir (or stock)
  - corresponds to a state variable
  - needs initial value

  Flow
  - defines rate of change (positive/negative) of a reservoir
  - connects reservoir with another reservoir or external sources/sinks (cloud)
  - is symbolized as a valve

  Converter
  - external parameter or auxiliary variable
  - computed using other values
  - concrete computation is hidden as parameter

  Connector
  - specifies, which variables affect others

  graphical representation



- Physical modeling:

  models built from "physical" components (masses, resistors, valves) instead of

integrators or function blocks

connecting lines represent "physical" connections (flanges, wires, pipes) instead of signals

internal representation uses Modelica language

- object-oriented, equation-based modeling language
- provides means for graphical representation of components and models
- huge free library of components (MSL = Modelica standard library)

simulation

- equations come from components and connections
- automatically combined, simplified (highly non-trivial!) and numerically solved

modeling and simulation environment

- several commercial programs
- open source: OpenModelica

- Modelica library `SystemDynamics.mo`:

  based on SystemDynamics 2.0 by Cellier

  design changed to cope with blocks like Oven

  using library in OpenModelica

  - start `OMEdit`
  - load base library `SystemDynamics.mo` und examples library `SystemDynamicsExamples.mo`
  - both are displayed in `Libraries` pane

  design of base library

  - packages `Reservoirs`, `Flows` and `Converters`
  - predefined components for common equations
  - user-defined components necessary for special mathematical relations
  - definition of such components easy
  - packages `Interfaces` contains supporting auxiliary components

  design of example library

  - `Examples` contains executable models
  - packages `Examples` for executable models, `AuxComponents` for auxiliary components
  - required data sets in package `Resources`

- Simple growth model `Inflow`:

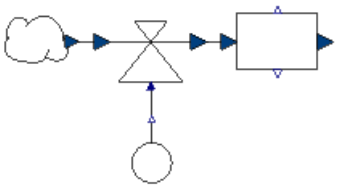
  only one state variable

  growth (inflow)

  - defined by flow (valve symbol)
  - rate variable (i. e. amount/time)

  defined by `ConstantConverter`

  diagram



- Building the model:

  pick up components from library pane and drag them into model pane
  - form `Reservoirs`: `Stock`, `CloudSource`
  - from `Flows`: `Flow`
  - from `Converters`: `ConstantConverter`

  connect components

  set parameter values (after double click on a component)
  - initial value of reservoir (`Stock`): `m0 = 2`
  - inflow rate (`ConstantConverter`): `constValue = 0.5`

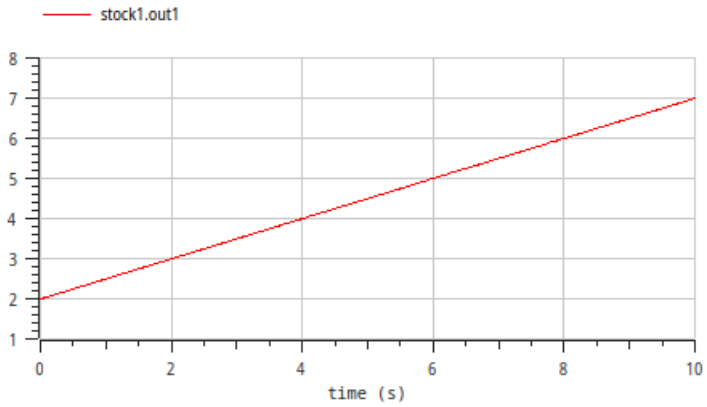
- Running simulation:

  check model

  setup and run simulation
  - `Stop Time` = 10
  - automatically runs simulation
  - → one output window is shown

  change window size (plot window icons "hidden" top-right)

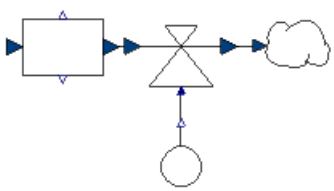
  choose variable: `Inflow.stock1.out1`



  use switch at bottom-right to retrun to model pane

- Model `Outflow`:

  mirror image of `Inflow`



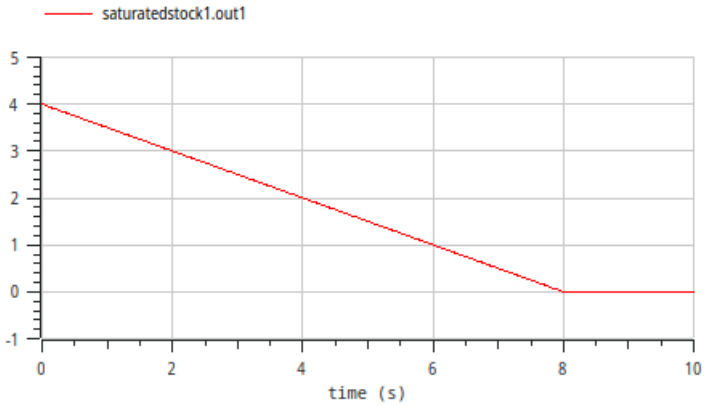
- initial value `m0 = 4`

simulation

- stock value becomes negative
- is a negative level meaningful?

alternative: use `SaturatedStock` with `minLevel` = 0

result



Population models

- Basic model:

  stock describes size N of population

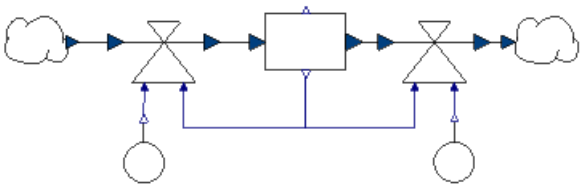
  number of births and deaths proportional to N

  $G = g\,N$
  $T = t\,N$

  rates g, t in `ConstantConverter` blocks

  products with `Mult2Flow` blocks

  model `population1`



  start with N(0) = 10

  result for g = 0.03, t = 0.01: exponential growth

- Limited growth:

  scarcity of resources → death rate grows for large population

  approach: death rate proportional to N

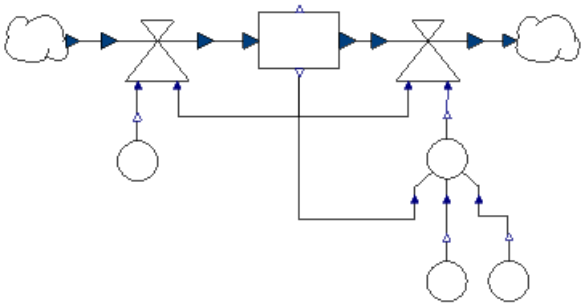
  $t = (N/N_b)\,t_b$

  implement equation with `MultPower3Converter`

  $out = in1^{k1} * in2^{k2} * in3^{k3}$

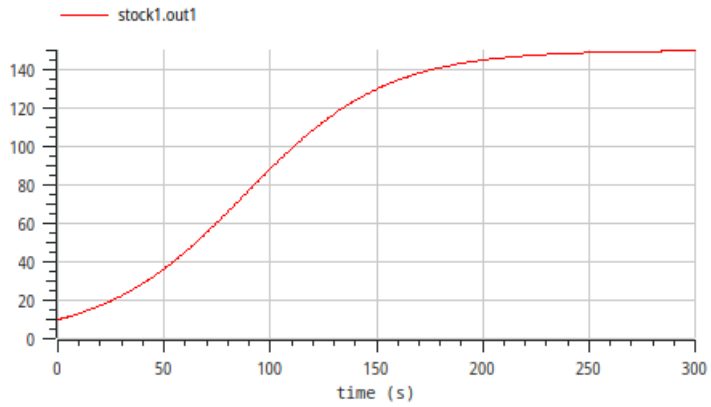
  - N: in1 = stock.out1, k1 = 1
  - t_b: in2 = 0.01 (coming from `ConstantConverter`), k2 = 1
  - N_b: in3 = 50 (coming from `ConstantConverter`), k3 = -1

  complete model `population2`



  result

  - change plot range with plot setup to [0, 150]



- Fixed capacity:

  changes in `population3`

  - death rate stays constant for small N
  - N has upper limit N_k

  idea

  $$t = \frac{t_b}{1 - N/N_k}$$

  no predefined converter for this formula

  replace `MultPower3Converter` by `CapacityConverter` using Modelica code

  ```
  block CapacityConverter
     extends SystemDynamics.Interfaces.GenericConverter3;
  equation
     out1 = in2/(1 - in1/in3);
  end CapacityConverter;
  ```

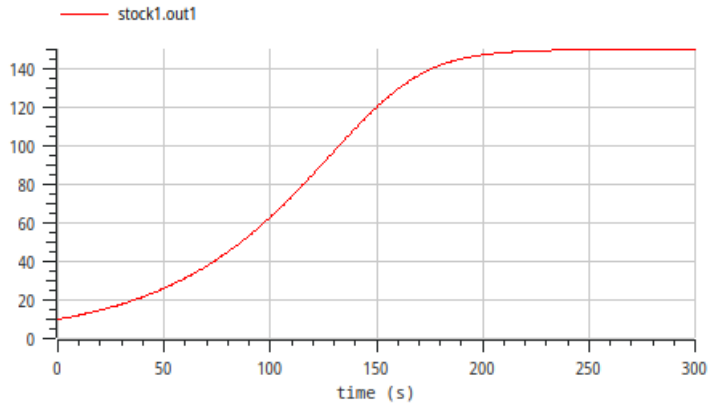
  create new component

  - in OpenModelica: `File/New/New Modelica Class`
  - Name: `CapacityConverter`
  - Extends: `SystemDynamics.Interfaces.GenericConverter3`
  - Insert in class: `SystemDynamicsExamples.AuxComponents`

  set N_k = 225

  result

  - similar to previous version



"curve fits data" does not imply "model mechanism is correct"!

Predator prey systems

- Modeling of predator and prey populations:

  state variables N_b, N_r for number of prey and predator animals

  inflows for births, outflows for deaths

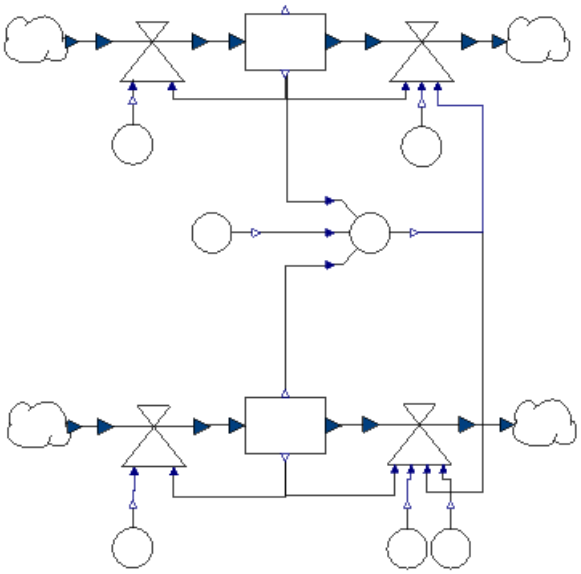
  flows like in population models
  - constant birth and death rates
  - number of births and deaths proportional to size of population

  interaction between the two species
  - F = number of prey hits
  - proportional to N_b and N_r
  - directly increases number of prey deaths T_b
  - decreases number of predator deaths T_r
  - demand B = number of prey animals (per time) needed by a predator to survive

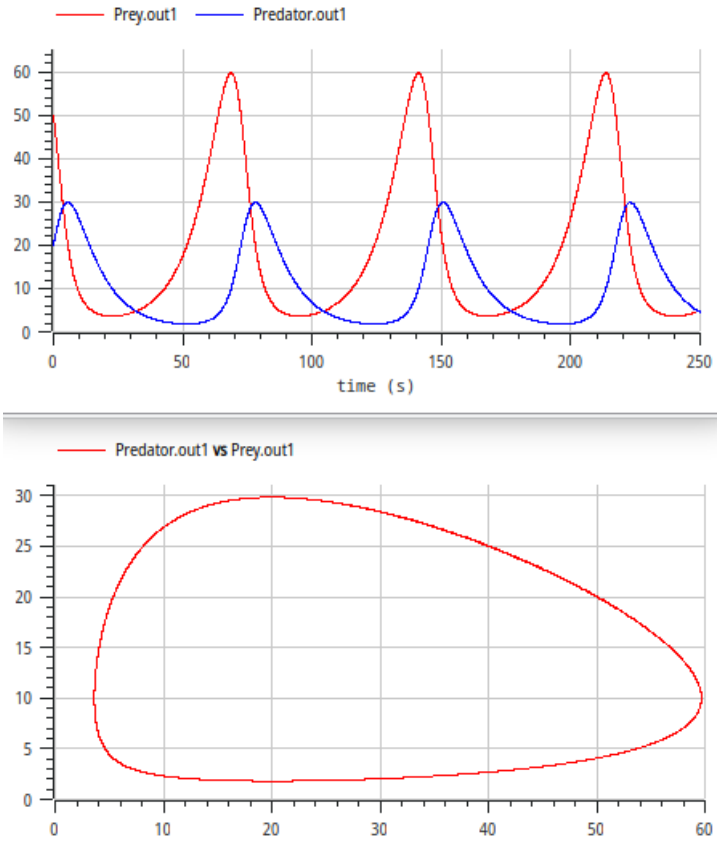
  concrete relations (Lotka-Volterra equations)

$$
\begin{aligned}
F &= fN_bN_r \\
T_b &= t_bN_b + F \\
T_r &= t_rN_r - \frac{F}{B}
\end{aligned}
$$

  model `PredatorPrey1`

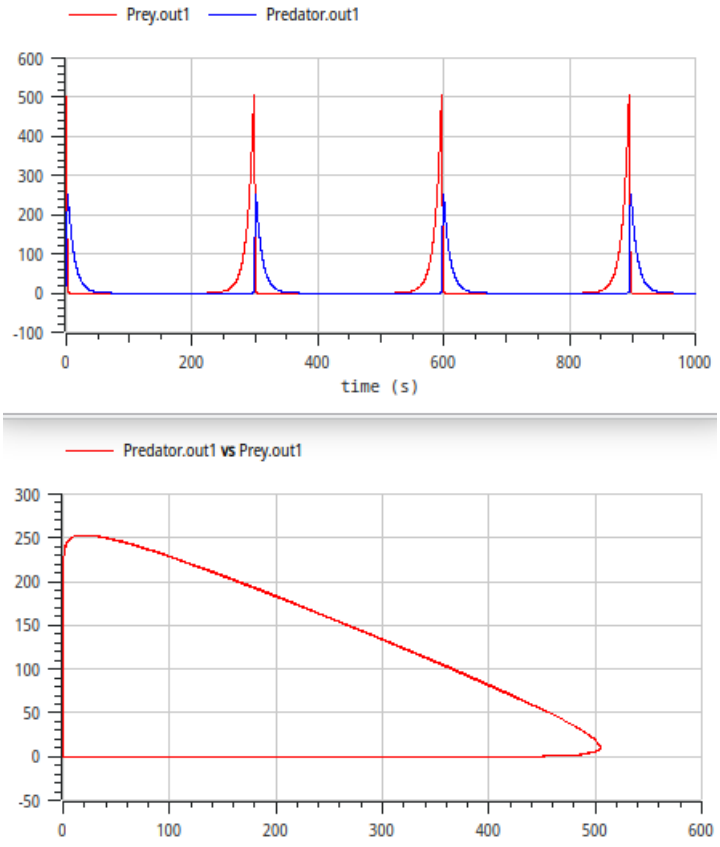


  results



- typical oscillations
- closed loop in phase diagram ≙ constant of motion

increase of initial number of prey animals from 50 to 500



- reduce solver tolerance to 1e-8 !
- N_r rises immediately
- both populations collapse but recover after a long time

- very steep oscillations
- still a closed loop in phase diagram
- very unnatural behaviour!

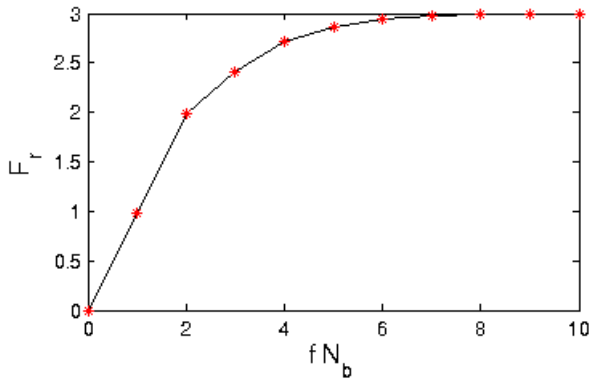
problem

- number of catches per predator $F_r = F / N_r = f N_b$ rises with N_b
- idea: limit F_r to a saturation value $F_{r,max}$

- Functions defined by graphs:

  saturation curve $F_r = sat(f N_b)$ known only qualitatively

  defined explicitely by a set of points and linear interpolation



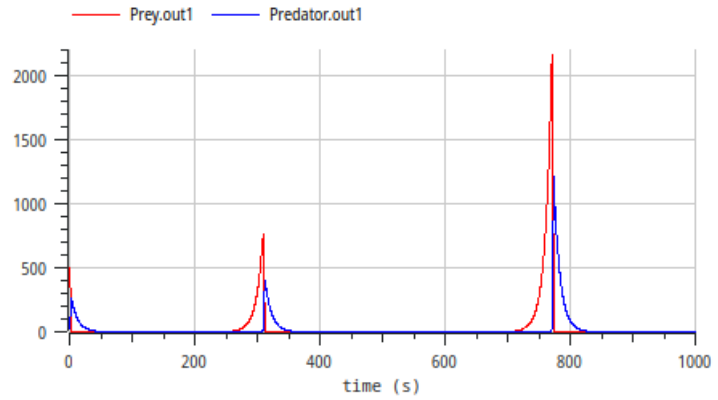
  implemented with component `Mult2GraphConverter`

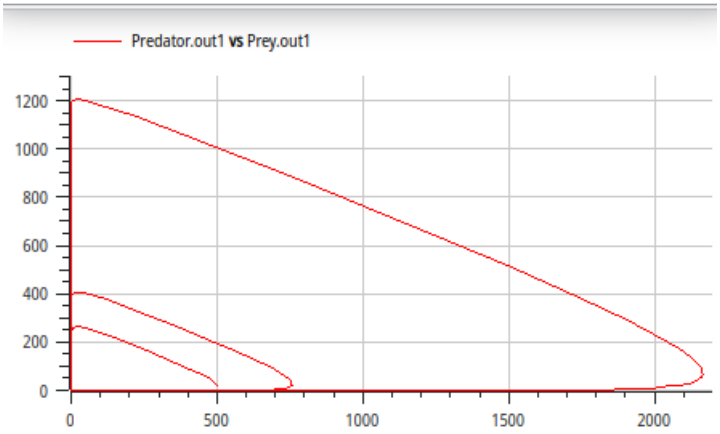
  - multiplies its inputs and applies interpolated function to the product
  - points defined in catchesPerPredator.txt
  - values defined in Modelica as constant array in `SystemDynamicsExamples.Resources.PredatorPrey.cpp`

- Results of `PredatorPrey2B`:

  $N_b(0) = 50$ → same results as before (F_r far from saturation)

  $N_b(0) = 500$





- oscillations get stronger!

reason: T_r becomes negative

$$T_r = \left(t_r - \frac{\mathrm{sat}(f N_b)}{B} \right) N_r$$

happened already for `PredatorPrey1`, but caused limited damage due to constant of motion

Forrester's WORLD2 model

- Modeling the world with WORLD2:

  5 state variables
  - size of (human) population P
  - capital investment CI
  - fraction of capital invested in agriculture CIAF
  - amount of natural resources NR
  - pollution POL

  additional variables to quantify relationships, e. g.
  - quality of life QL
  - per capita food ratio FR
  - capital investment ration CIR

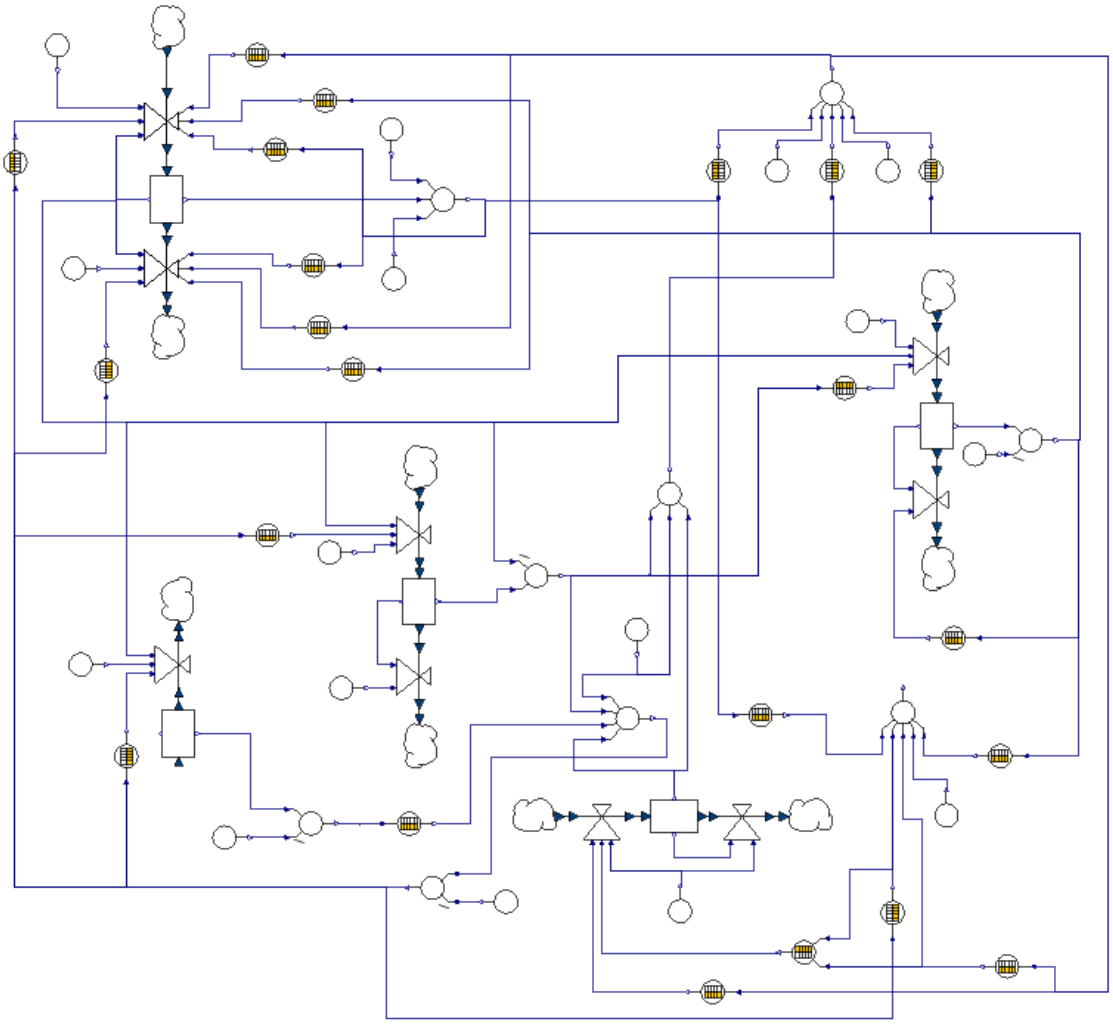
  relations between the variables
  - plausible equations
  - graphical functions based on statistical data
  - qualitative graphical functions

  complete model and simulation results described in: Forrester, World Dynamics (1970)

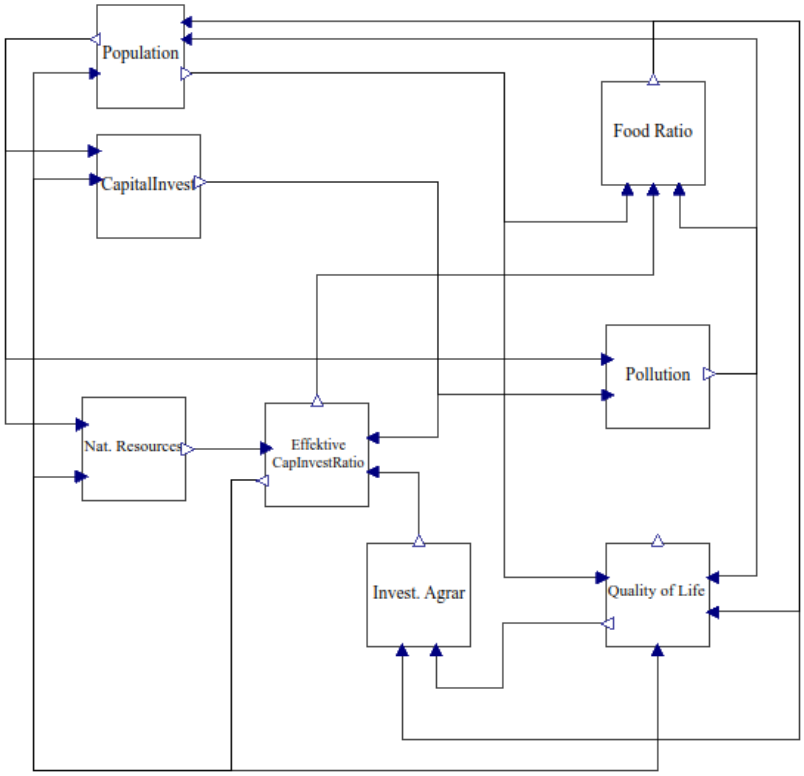
  attracted great international attention

- Complete system dynamics model:

  based on standard components, using only one special component `ECIRConverter`



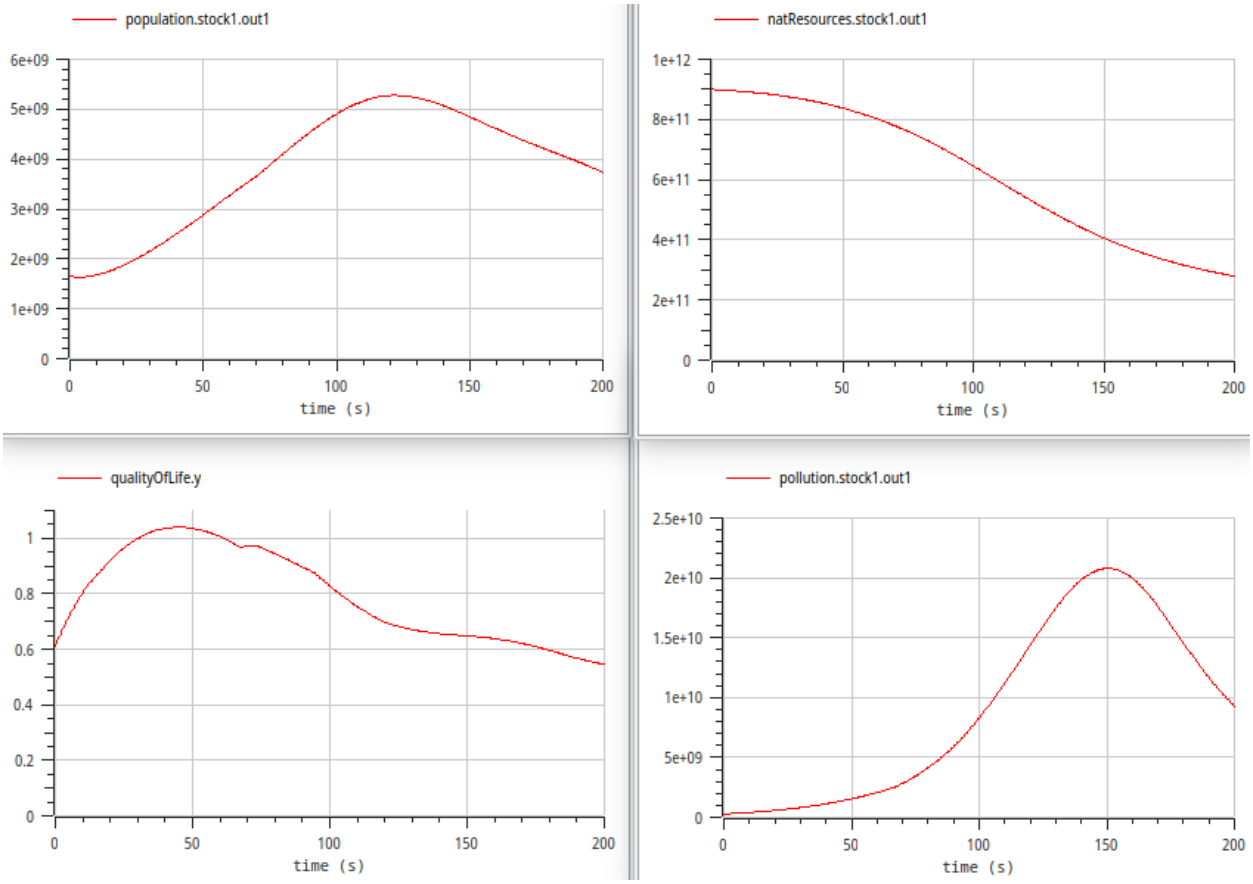
enhancement in Modelica: introduction of submodels



simulated time range 1900 - 2100

parameters adapted to reproduce values of 1970

results of selected variables



- Consequences:

  lots of criticism, scientifically and political

further development: much more complex model WORLD3

qualitatively similar results

published in "Limits to growth" (1972), still highly debated

detailed discussion of WORLD2, implemented in Vensim

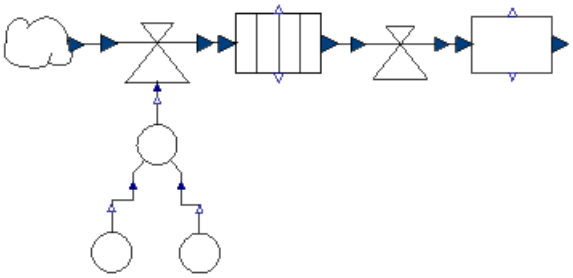
Manufacturing

- Conveyor belt (`Conveyor`):

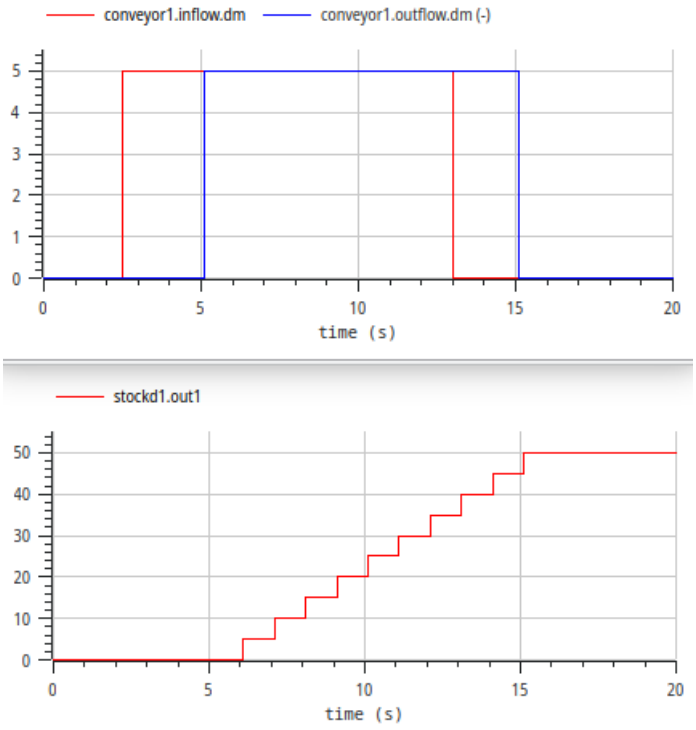
  special stock element introduced in Stella, redesigned in Modelica

  functionality
  - operates at clock ticks $t_n = t_0 + n\,\Delta t$
  - stores input value
  - outputs the value after a given transit time t_{tr}

  example
  - inflow = 5 (per time unit)
  - $t_0 = 0.1$, $\Delta t = 1$, $t_{tr} = 3$



  results



  details
  - discrete simulation at sample times $t_n = 0.1 + n$
  - needs discrete stock component `StockD`
  - input flow is 5 for $2.5 \le t \le 13$ (implemented with `TimeSwitchedConverter`)
  - initial waiting time < Δt is rounded to Δt

- Manufacturing machine (`Oven`):

another discrete stock from Stella

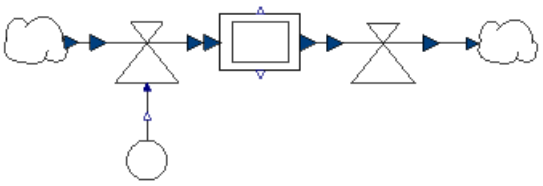
baking tray model: fill tray, bake, unload

parameters `initialLoad`, `capacity`, `cookingTime`

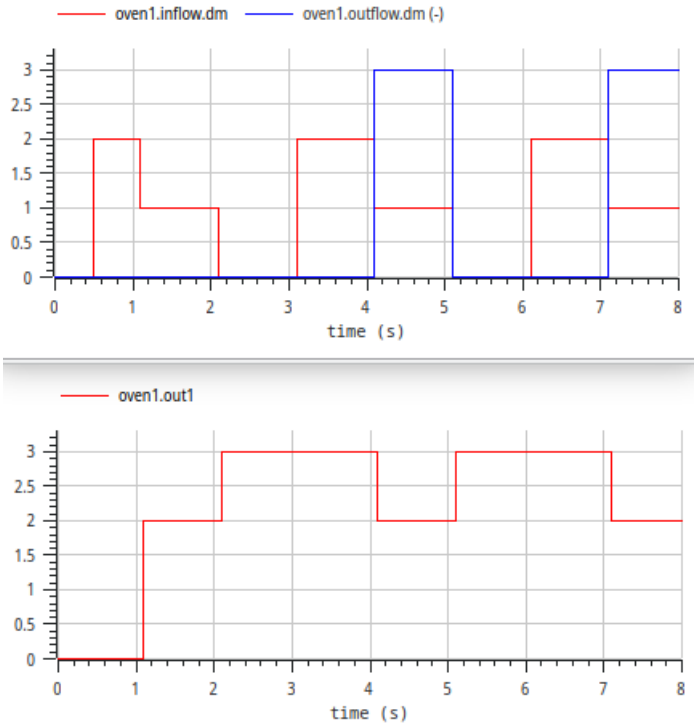
processing phases

- loading: get input, until capacity is reached
- production: wait until `cookingtime` ends
- unloading: output complete content, immediately start with reloading

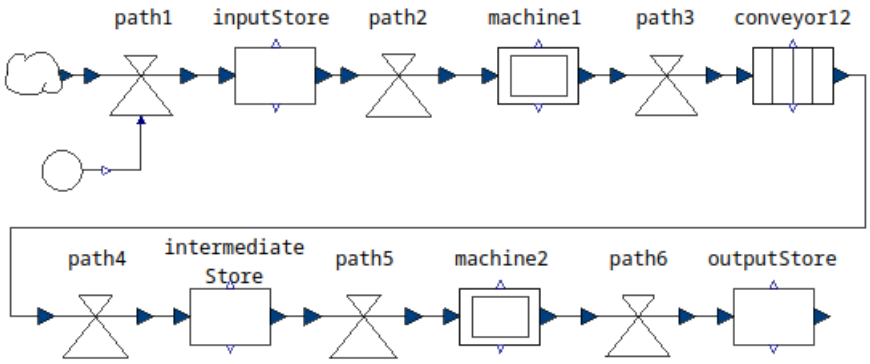
example with `initialLoad = 0`, `capacity = 3`, `cookingTime = 2`



results



- Model of a simple assembly line:



delivery of 2 units per time unit of raw material

stored in `inputStore` with initial stock = 7

`path2` transports 2 parts per time unit

`machine1` has a capacity of 4 and needs 2 time units

`conveyor12` has a transport time of 2 time units

`intermediateStore` for buffering, initially empty

`path5` transports 2 parts per time unit

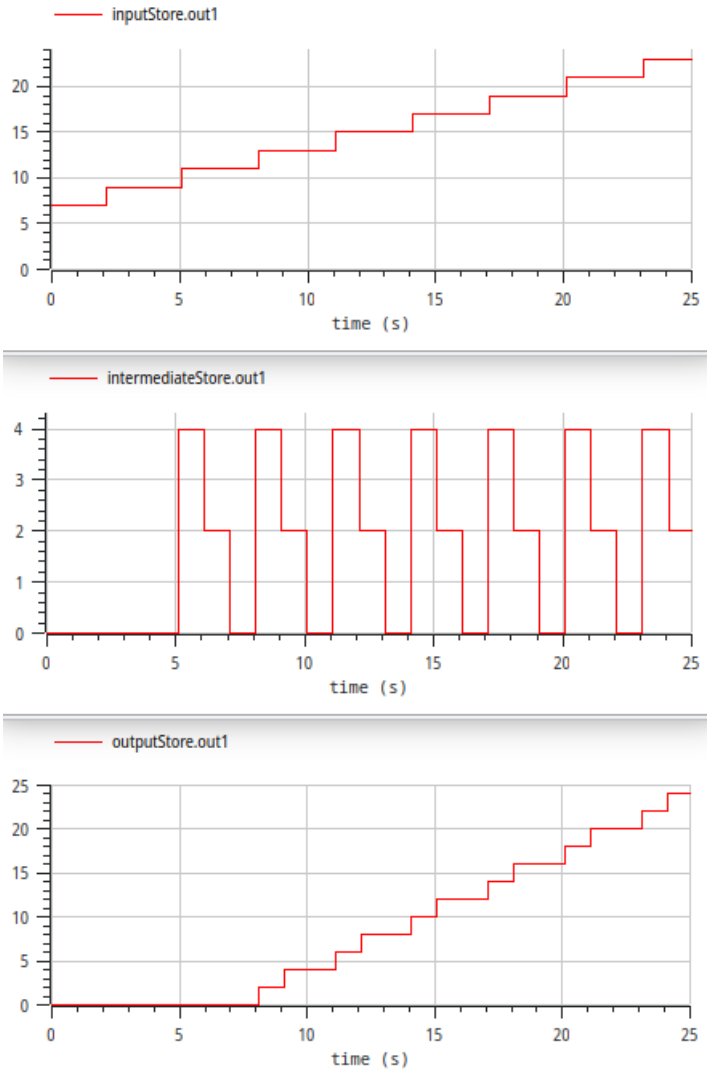
`machine2` has a capacity of 2 and needs 1 time unit

`outputStore` for final products, initially empty

- Simulation results of model `AssemblyLine`:

  expectation: throughput of 2 parts per time unit

  inventory of the stores

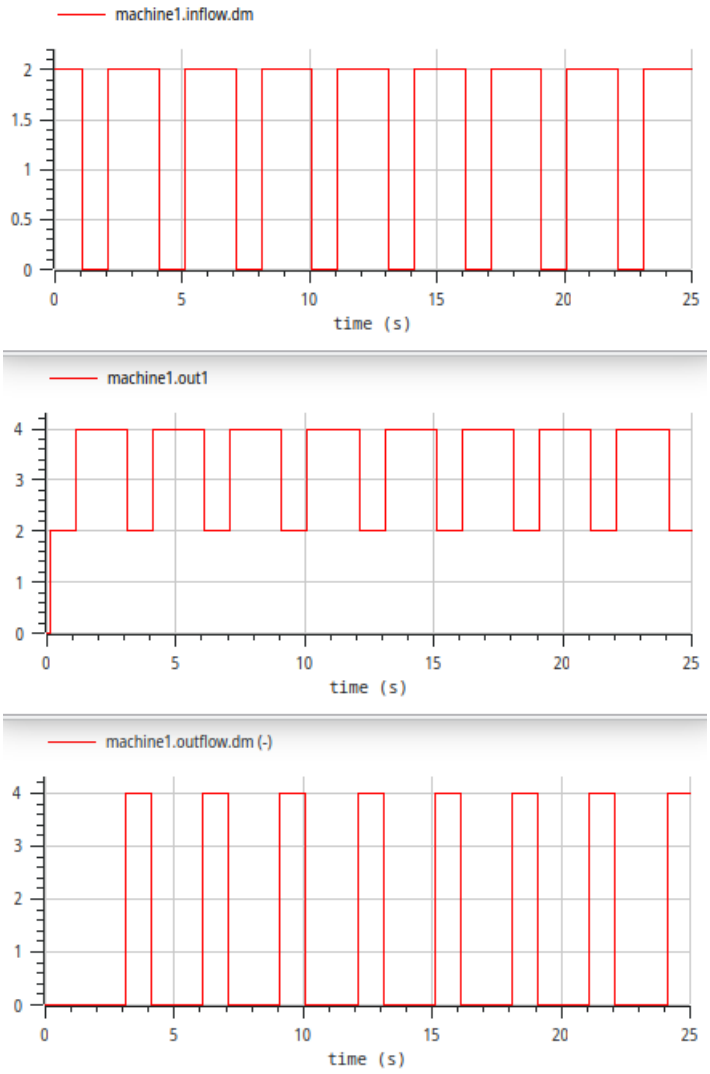


  problem

  - `inputStore` grows continually
  - uneven growth of `outputStore`
  - average throughput: 1.5 parts per time unit
  - all stations allow for a throughput of 2 parts per time unit

  cause

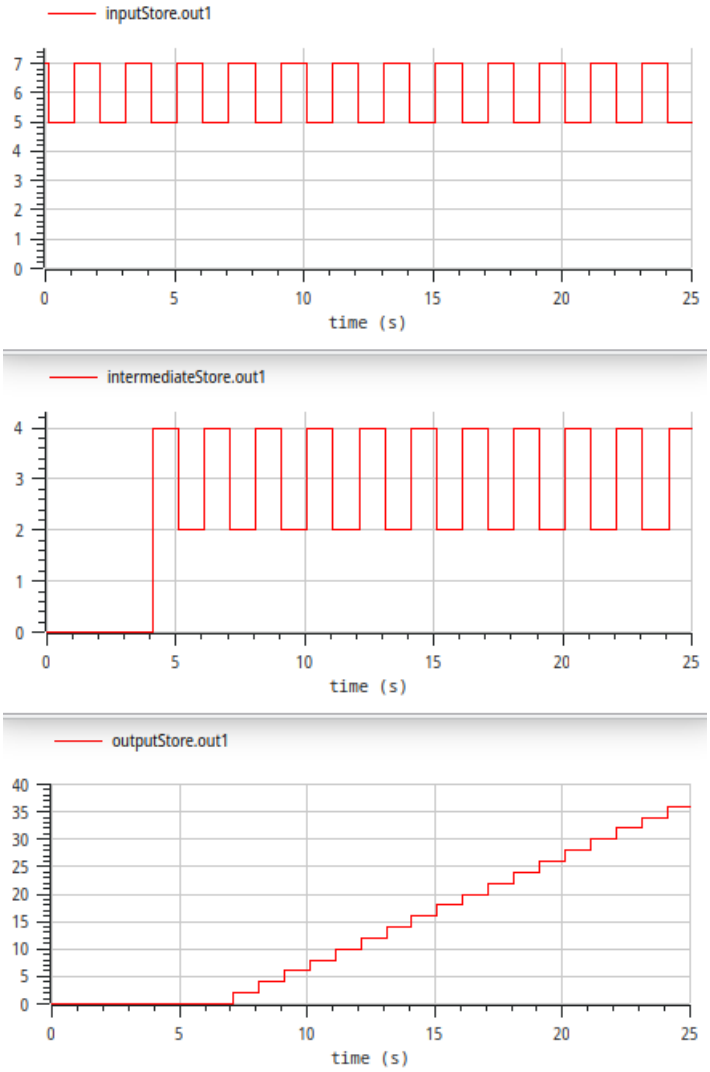
- behavior at `machine1`



- `path2` transports 2 parts per time unit
- can't deliver during production time of `machine1`
- loading of `machine1` needs 4 parts, but only 2 parts can be delivered in one step

solution:

- `path2` must be extended to transport 4 parts per time unit



- simulation shows necessary sizes of stores under optimal conditions

Internals of the SystemDynamics library

- Basic idea from Cellier:

  notation
  - flow line denoted by large arrow
  - signal line denoted by (standard) small arrow

  Connector
  - standard input/output connectors for signal ports
  - input/output connectors for flow with different icons (`MassInPort`/`MassOutPort`)

  Converter
  - block, i. e. dedicated input and output ports

  Flow
  - computes flow dm from inputs
  - distributes dm via `MassOutPort`s to reservoirs

  Reservoir
  - gets dm from `MassInPort`s
  - subtracts and integrates to get stock m

  all ports have static causality (i. e. are designated inputs or outputs)

- Problems with causality:

  model `Outflow`
  - usually flow defines dm
  - stock is (almost) empty → reservoir (`Stock`) defines dm

  model `TestConveyor`
  - reservoir (`Conveyor`) always defines dm
  - value provided by flow is ignored

  model `TestOven`
  - output flow (0 or capacity) given by reservoir (`Oven`)
  - input flow depends on phase
  - loading: given by flow and reservoir together
  - production: given by reservoir (0)

  model `AssemblyLine`
  - inflow of `inputStore` defined by `path1`
  - inflow of `machine1` depends on stock of `inputStore`, value of `path2` and state of `machine1`

- Conception of the connector `MassPort`:

  mass flow `dm` defined as flow variable

  corresponding potential variable `data`

- contains amount defined by reservoir

additional integer variable `info` defines, what to do with `data`

- info = 0 → discard it, dm = dm_{flow}
- info = 1 → use it directly, dm = data
- info = 2 → reservoir is restricted, dm = min(data, dm_{flow})

`info` has fixed causility: output of reservoir, input of flow

two connectors

- ```
  connector MassPortR "mass port of reservoirs"
    flow Real dm;
    Real data;
    output Integer info;
  end MassPortR;
  ```

- ```
  connector MassPortF "mass port of flows"
    flow Real dm;
    Real data;
    input Integer info;
  end MassPortF;
  ```

- Practical considerations:

  several predefined `Flow` and `Converter` models implementing common equations

  defining special equations easy using generic blocks in `Interfaces`

  tabular data not as files, but as constant arrays
  - several tables can be combined in one resource class

Appendix

- Homepage of OpenModelica
- Webpage "System dynamics library in Modelica"
- Modelica library SystemDynamics
- Modelica library SystemDynamicsExamples
- Literature

Literature

1. Introduction to system dynamics with lots of examples:

   Hannon B, Ruth M: Dynamic Modeling. Springer, New York, 2nd edition, 2001.

2. Description of WORLD2:

   Forrester JW: World Dynamics. Pegasus Communications, 1971.

3. Textbook on modeling and simulation with chapter about system dynamics with Modelica (in German):

   Junglas P: Praxis der Simulationstechnik. Europa-Lehrmittel, Haan-Gruiten, 2014.

4. Basic implementation of system dynamics in Modelica:

   Cellier FE: World3 in Modelica: Creating System Dynamics Models in the Modelica Framework. In: Proc. 6th Int. Modelica Conf., Bielefeld, Germany, 2008; p. 393 – 400.

5. More complete implementation of system dynamics in Modelica:

   Junglas, P: Causality of System Dynamics Diagrams, SNE Simulation Notes Europe 26/3 (2016), 147-154.