Biomath Communications

www.biomathforum.org/biomath/index.php/conference

Simulation of Gradostat Using System Simulation Software Tools

Milen Kolev Borisov
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
milen_kb@math.bas.bg

Abstract

System simulations are used for modeling simple and complex processes that can be expressed in terms of ordinary differential equations, differential algebraic equations and discrete event simulation. There are many software tools for system simulations and all of them represent the physical system in a graphical form using many components connected together.

A gradostat is an array of interconnected chemostats (bioreactors) and it is very suitable for modeling and simulation by system simulation tools. In this paper we demonstrate how to model and run system simulations of 3-vessel gradostat.

Keywords: Gradostat, System simulation, FMI (Functional Mock-up Interface).

1 Introduction

A gradostat is a type of bioreactor which is consisted of an array of interconnected chemostats (culture vessels), with the nutrition (substrate) and microorganisms flowing between them (see Fig. 1). Devised by Lovitt and Wimpenny [7], the gradostat is suitable for studying the growth of microorganisms in a nutrient gradient, which occurs in abundance in nature. For example [13], the surface films in dental plaque represent growth along such a gradient, as does growth along the banks of a stream or along a seacoast. In a water column, sunlight replaces the nutrient as an essential source needed for growth of microorganisms. On the other hand, the flow

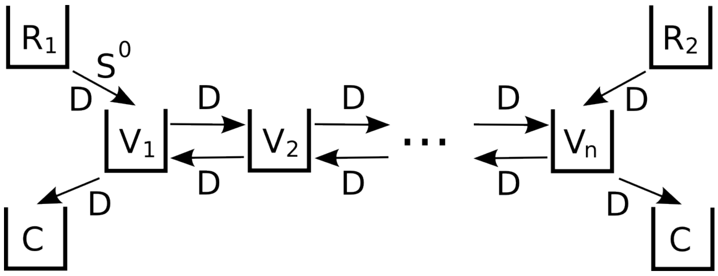
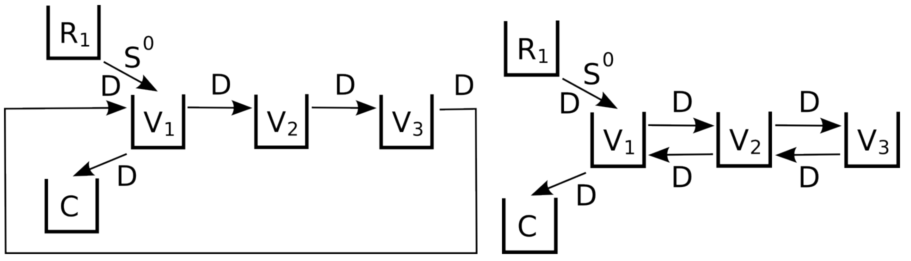


Figure 1: The standard n-vessel gradostat. V_i are the culture vessels which contain microorganisms, R_1 is a reservoir containing a medium (substance) and substrate at concentration S^0, R_2 is a reservoir containing only a medium, C is an overflow vessel, D denotes the dilution rate. Note that, in the absence of microorganisms, at steady state, the gradostat has a linear gradient of the substrate concentration.



(a) Cyclic gradostat

(b) "Dead-end" gradostat

Figure 2: Examples of gradostats

characteristics of a gradostat make it a simple analog of aquatic ecosystems such as river estuaries [4].

Fig. 1 presents the most popular connection pattern of a gradostat. In [13] we can see that much more imaginative connection patterns are possible. For example, a cyclic gradostat and "dead-end" gradostat (see Fig. 2). Note that the inflow into each vessel balances the outflow.

System simulations are used for modeling real dynamical processes that can be expressed in terms of ordinary differential equations, differential algebraic equations and discrete event simulation. There are many system simulation software tools and nearly all of them are suitable for modeling and simulation of gradostats. Most of them also support the FMI

(Functional Mock-up Interface) standard for model exchange between them, which gives us an opportunity to create much more reusable simulation software tools.

In this paper we demonstrate how to model and run system simulations of a 3-vessel gradostat model using the FMI standard.

2 System Simulations

System simulations allow the user to build models of simple or complex processes that can be expressed in terms of mathematical equations. The software products used in system simulations offer the user libraries of components, which are used as building blocks. The component complexity could vary widely depending on the domain, the purpose of the component and the accuracy with which the real-world behavior has to be modeled. Some of the products allow the user to create his own components. A list with some software tools for system simulations is introduced in the Appendix.

The commonality between all system simulation software tools is that they model the real system in a graphical form using many components connected together. An example of such kind of model of an air conditioner is shown in Fig. 3. Note that we can create and simulate this model without knowing anything about the physical or mathematical model of the components (compressor, pipes, valves, etc.).

3 Gradostat

As we said the gradostat is an array of interconnected culture vessels, with the substrate and microorganisms flowing between them. In the gradostat, the density of the microorganisms in each vessel is a result of the balance among immigration, emigration and reproduction. An example of a 3-vessel gradostat is shown in Fig. 4.

Let's define a simple mathematical model of an n-vessel gradostat, which we will use for our system simulations in the next sections. In that model, for each vessel V_i we define a set of two differential equations that describe the dynamics of x_i - microorganisms and s_i - substrate.

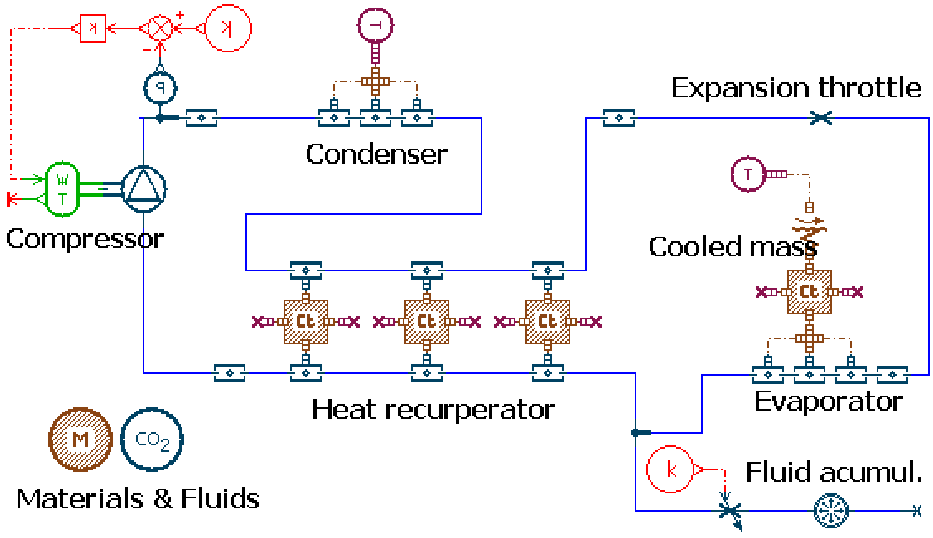


Figure 3: An air conditioner model in AMESim

$$
\begin{array}{rcl}
s_i' & = & (s_{i-1} + s_{i+1})Dr - 2s_iDr - \frac{1}{\gamma}\mu(s_i)x_i \\
x_i' & = & (x_{i-1} + x_{i+1})Dr - 2x_iDr + \mu(s_i)x_i
\end{array} \quad (1)
$$

where:

- $i = 0 \ldots n$ and n is the number of vessels

- $s_1 = S$, $s_{n+1} = x_0 = x_{n+1} = 0$ – boundary conditions applying to V_1 and V_n. The first vessel V_1 receives substance with substrate in concentration S, but without microorganisms (i.e. $x_0 = 0$), from its reservoir R_1. The last vessel V_n receives only the substance without substrate and microorganisms from its reservoir R_n.

- $\mu(s) = \frac{m\,s}{k+s}$ – specific growth rate

- $s_i(t)$ – substrate concentration in vessel V_i

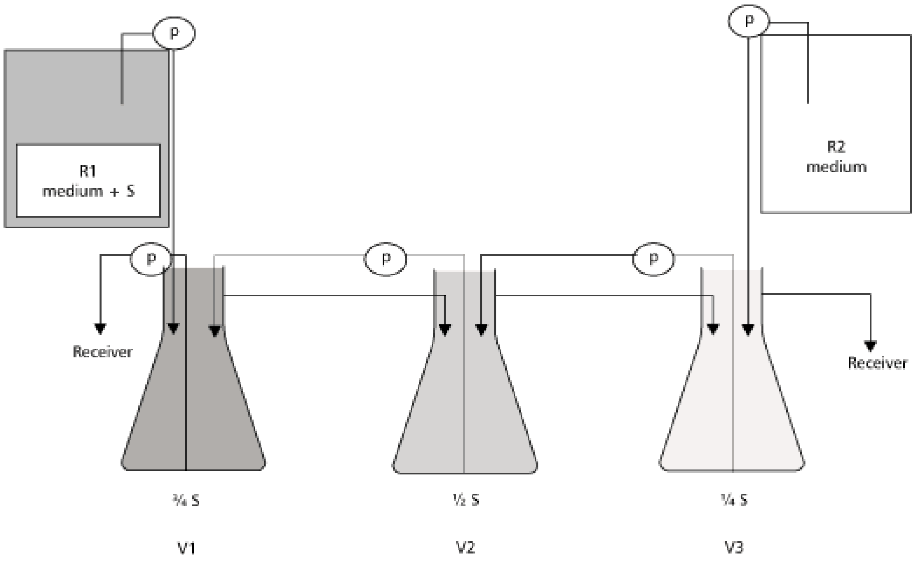


Figure 4: 3-vessles gradostat [4], where:
V_1, V_2, V_3 – are the culture vessels contain microorganisms; R_1, R_2 – are the reservoirs contain substrate and medium (substance); P – are the pumps drives the flow from V_3 to V_1 while the opposite movement is driven by overflow; S – the amount of substrate in R_1.

- $x_i(t)$ – microorganisms concentration in vessel V_i

- Dr – flow rate between vessels, reservoirs and receivers

- m – maximum growth rate

- k – half-saturation constant

- γ – yield coefficient of microorganisms ($\gamma \leq 1$)

The first set of equations of (1) models the dynamics of the substrate s_i in vessel V_i. The amount of substrate in each vessel increases by the inflow from neighbor vessels V_{i-1} and V_{i+1} (first term), and decreases by the outflow (second term) and consumption (third term). The yield coefficient γ indicates how much substrate is eaten up when new microorganisms are formed (i.e. the amount of biomass formed per mass of a given substrate).

The second set of equations of (1) describes the dynamics of microorganism x_i in vessel V_i. The first term indicates that the microorganism density increases as migrants come in from the left V_{i-1} and right V_{i+1} vessels. The second term shows that microorganisms are lost to each one of the two neighbor vessels at a rate of Dr. The last term indicates population growth as a result of reproduction.

One of the most interesting predictions of the gradostat theory is the large impact of flow rate on microorganism distribution [4]. At low flow rates, microorganism distribution is expected to follow the concentration of the limiting substrate, i.e. microorganisms should concentrate in the foremost vessels, which received more substrate, and also microorganism concentration should correlate to substrate concentration. As one increases the flow rate, however, microorganism concentration should become uncorrelated to substrate concentration. That is, at high flow rates vessels at the ends of the array become sinks for the microorganisms, because the dilution rate exceeds the growth rate. Microorganisms would then concentrate in the middle vessels where the dilution rate is lower, generating a triangular distribution.

In the next section we will test the theory prediction presented above by using the system simulation software *Simulink*. Note that the main idea of the paper is not to show the correctness of the Gradostat theory, but to show how to use system simulation for modeling and simulation of a gradostat.

4 System Simulation of Gradostat

If you want to do a system simulation of some real system, a good practice is to stick to the following steps:

1. Define the components of the real system

2. Find a suitable system simulation software tool

3. Find or create the components
4. Create the model by connecting the components
5. Set the parameters of the model
6. Run the simulation of the model
7. Interpret the results

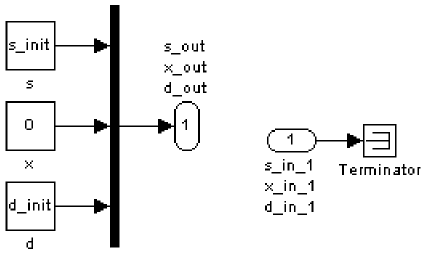
In our case the real system that we want to simulate is the gradostat from the previous section. If we have a close look at its scheme in Fig. 4 we will see that the real system (i.e. gradostat) is built using three components: vessels, reservoirs and receiver (see Table 1). So the first step is done.

Vessel	**Reservoir**	**Receiver**
State variables: s, x	State variables: -	State variables: -
Parameters: k, m, γ	Parameters: S, Dr	Parameters: -
Inputs: 2 ports	Inputs: -	Inputs: 1 port
Outputs: 2 ports	Outputs: 1 port	Outputs: -
Equations: for s' and x' see (1)	Equations: -	Equations: -

Table 1: Gradostat components and their properties

The next step is to find a suitable system simulation software tool. There are many such tools (see Appendix) and most of them can be used for our purpose. In this example we have chosen *Simulink,* which is one of the most popular simulation software products.

Simulink does not contain our needed components, so we should create them. To do that we will use components (blocks) from the standard *Simulink Library* like *Integrator block*, *Function block*, *Source block*, *Sink block* and others. The Reservoir and Receiver components are simple and we have used only Source and Sink blocks to implement them (see Fig. 5). The Vessel is a more complex component, therefore for its implementation we have used also *Integrator* and *Function blocks* (see Fig. 6). Note that our components in *Simulink* can be created also by using S-Function [10], but in that case we will have to write source code in the *C* programming language.



(a) Reservoir (b) Receiver

Figure 5: Implementation of Reservoir and Receiver components in *Simulink*

Now that we have the components of the gradostat, we can proceed to the next step, i.e. to create the model of the gradostat by connecting them together. It is an easy task and at the end we have a model of a 3-vessel gradostat, which is shown in Fig. 7.

Before running the simulation of the model, we should set the parameters of the model. For our example we will use the values of the parameters taken from [4] (see Table 2). In *Simulink* the values of the parameters of the components are set using dialogs like the one in Fig. 8.

Now we are ready to run the simulation of the 3-vessel gradostat. *Simulink* allows us to use different kinds of solvers with fixed or variable step, with different tolerances and algorithms (for more information about *Simulink* solvers see the topic "Choose a Solver" from *the Simulink* Documentation [9]). For our example we will use the default step-variable solver (see Fig. 9).

The result of the simulations with different values of the flow rate are shown in Fig. 10. As we can see the simulation results confirm the prediction of the gradostat theory from the previous section. On the first figure when the flow rate is low (i.e. $Dr = 1.0$) microorganisms are concentrated in the vessel receiving more substrate (i.e. V_1). On the other hand on the second figure when the flow rate is hight ($Dr = 6.0$) then microorganisms are concentrated in the middle vessel (i.e. V_2).

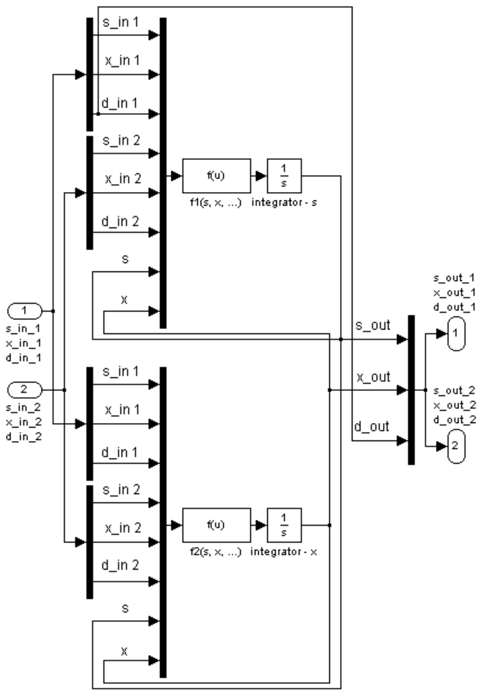


Figure 6: Implementation of Vessel component in *Simulink*

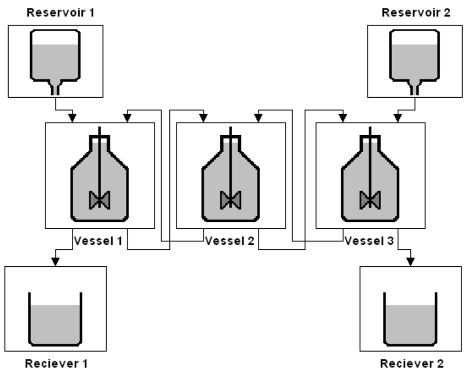
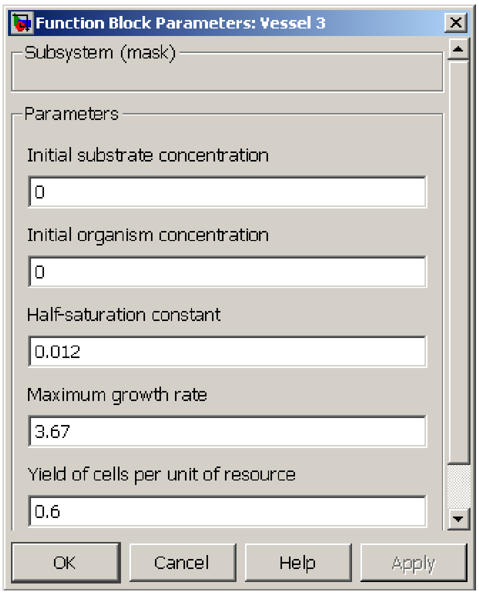


Figure 7: The model of 3-vessels gradostat in *Simulink*

Once we have the components of the gradostat, it is easy to create a model of a n-vessel gradostat, where $n = \{1, 2, ...\}$. Also, these components

Component	Parameter	Values
Reservoir 1	S-substrate concentration	2.2
	Dr - rate of transport	$\{1.0, 6.0\}$
Reservoir 2	S-substrate concentration	0.0
	Dr - rate of transport	$\{1.0, 6.0\}$
Vessel 1	s_{init} - initial substrate concentration	0.0
	x_{init}- initial microorganisms density	0.5
Vessel 2	s_{init} - initial substrate concentration	0.0
Vessel 3	x_{init}- initial microorganisms density	0.0
Vessel 1	m - maximum growth rate	0.012
Vessel 2	k - half-saturation constant	3.67
Vessel 3	γ - yield coefficient of microorganisms	0.6

Table 2: Parameters of the model of 3-vessel gradostat


Function Block Parameters: Vessel 3

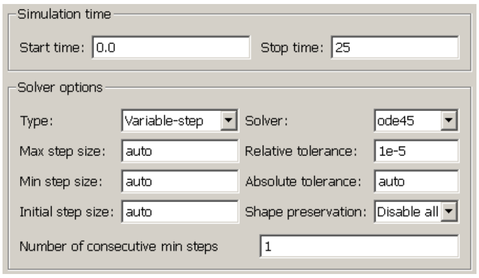
Subsystem (mask)

Parameters

Initial substrate concentration	0
Initial organism concentration	0
Half-saturation constant	0.012
Maximum growth rate	3.67
Yield of cells per unit of resource	0.6

Figure 8: Dialog for setting the values of the parameters of the last Vessel

can be distributed to other users of *Simulink* who, without knowing anything about their implementation (i.e. about the mathematical equations


Simulation time

Start time: 0.0 Stop time: 25

Solver options

Type: Variable-step Solver: ode45

Max step size: auto Relative tolerance: 1e-5

Min step size: auto Absolute tolerance: auto

Initial step size: auto Shape preservation: Disable all

Number of consecutive min steps 1

Figure 9: Configure Simulink Solver

at the components), can create their own models of the gradostat with a different number of vessels and connection patterns like those in Fig. 2.

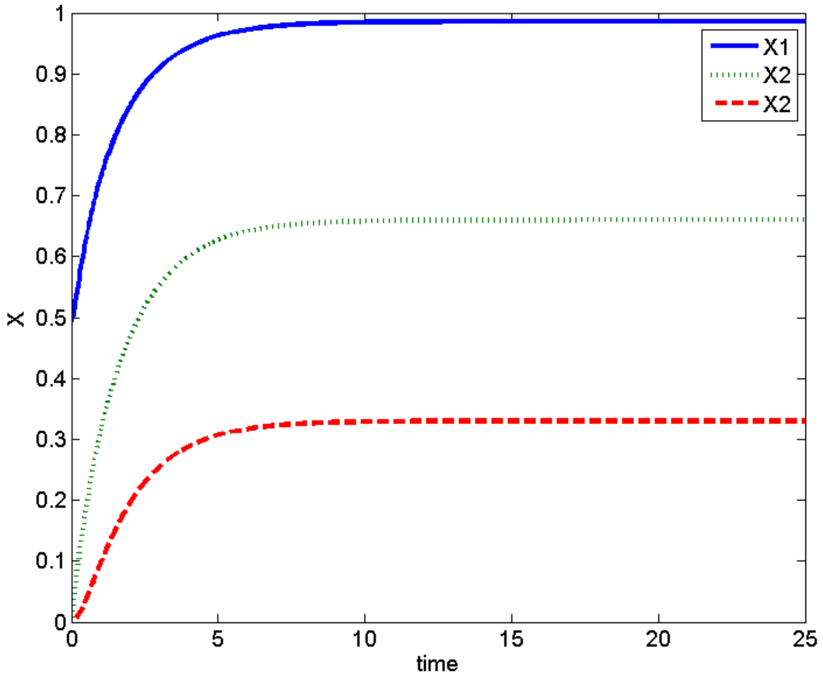
5 FMI (Functional Mock-up Interface)

One of the disadvantages of system simulation software is that when we create components using a specific software product, they can be used only with that particular simulation software tool. For example our components (Vessel, Receiver, Reservoir) can be used only in *Simulink*. To solve this problem we will use the Functional Mock-up Interface (FMI), which is an independent standard for model exchange between different simulation software tools. As of today, FMI is supported by over 35 simulation software tools. For more information about it, see the main site of the FMI project [12].

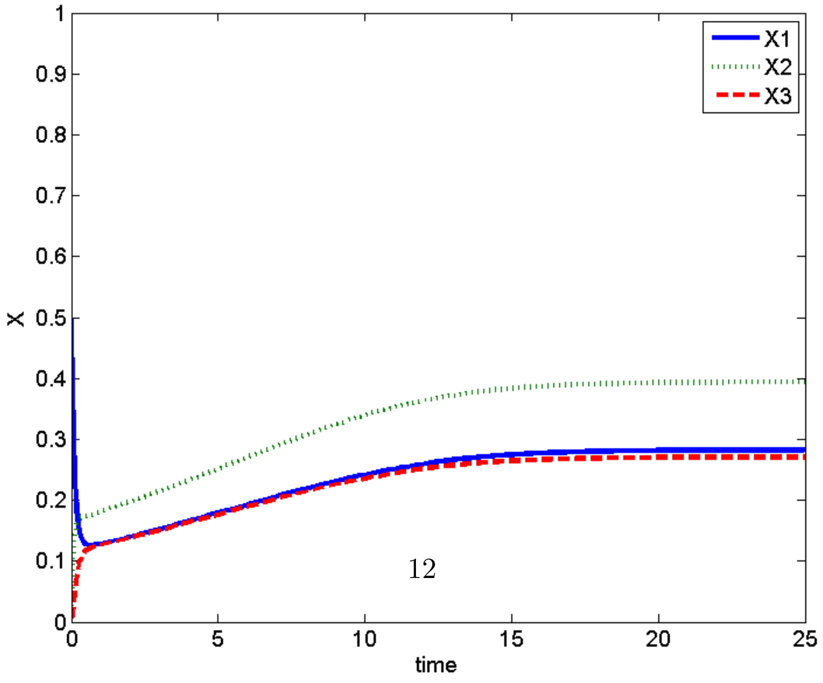
In brief, the idea of the FMI is to describe models defined by differential, algebraic and discrete equations and to provide an interface to evaluate these equations as needed in different simulation software tools, as well as in embedded control systems, with explicit or implicit integrators and with fixed or variable step size. Note that FMI regards the components as models, which is not so strange as each component is a model of something, e.g. the Vessel component is a model of the dynamics of the real Vessel.

Each model that supports the FMI standard is called FMU (Functional Mock-up Unit). FMU is distributed in a zip file with the extension **.fmu* which contains several files:

- an *xml*-file which contains the definition of all variables in the model and other model information.



(a) a low flow rate ($Dr = 1.0$)



(b) a hight flow rate ($Dr = 6.0$)

- all necessary model equations are provided with a small set of easy-to-use C-functions which can either be provided in source files (**.c)* and/or in binary format (**.dll* files). Binary formats for different platforms can be included in the same *fmu*-file.
- other files with additional model data like tables, maps, documentations, etc.

In order to create an FMU of our gradostat components we have to use some of the system simulation tools that support export of their components to FMU. In our case we are using *JModelica*, which is based on the *Modelica* modeling language [3], which was created especially for modeling dynamical systems (for more information about *JModelica* see the Appendix). The reason for not using *Simulink* is that the standard distribution of *Simulink* does not support export to FMU.

Firstly, we create our gradostat components writing source code in the *Modelica* language. Fig. 11 shows the source code of the Vessel component (i.e. the contents of the modelica *Vessel.mo* file). Likewise, we write the source code of the Reservoir and Receiver components. To compile the modelica files to FMU we use *Python API* of *JModelica.* The *Python* script which compiles and creates the FMU from the modelica files of the gradostat components is shown in Fig. 12. Running the script we get three FMU files (i.e. Vessel.fmu, Reservoir.fmu and Receiver.fmu) which we can import into any simulation software tool that supports FMU and after that we can use them to create a model of our gradostat. In our example we will do it in the simulation software tool *SimulationX* (see the Appendix).

Fig. 13 shows a model of the gradostat in *SimulationX* which is built using the FMU files which we created with *JModelica* and imported into *SimulationX.* As we can see, the graphical representation of the model in *SimulationX* is similar to the one in *Simulink* (see Fig. 7 and Fig. 13).

We will test the model using linear ramp values for the flow rate Dr:

	start value	end value	start time	end time
$Dr =$	1.0	1.0	0	50
	1.0	6.0	50	150
	6.0	6.0	150	200

The result of the simulation is shown in Fig. 14 where from the graphic we can see that for $Dr \approx 3.6$ the concentration of the microorganisms in

```
model Vessel
        // Parameters & default values
        parameter Real s_init = 0;
        parameter Real x_init = 0;
        parameter Real k = 0.012;
        parameter Real m = 3.67;
        parameter Real g = 0.6;

        // State variables
        Real s(start = s_init);
        Real x(start = x_init);

        // Input variables
        input Real s1_in, x1_in, Dr1_in;
        input Real s2_in, x2_in, Dr2_in;

        // Output variable
        output Real s1_out, x1_out, Dr1_out;
        output Real s2_out, x2_out, Dr2_out;

    equation
        der(s) = s1_in*Dr1_in + s2_in*Dr2_in
            - (Dr1_in + Dr2_in)*s
            - (1/g)*((m*s)/(k+s))*x;
        der(x) = x1_in*Dr1_in + x2_in*Dr2_in
            - (Dr1_in + Dr2_in)*x
            + ((m*s)/(k+s))*x;

        s1_out = s; x1_out = x; Dr1_out = Dr1_in;
        s2_out = s; x2_out = x; Dr2_out = Dr2_in;
end Vessel;
```

Figure 11: *Modelica* source code of the Vessel component

```
from pymodelica import compile_fmu

compile_fmu("Receiver", "Receiver.mo")
compile_fmu("Reservoir", "Reservoir.mo")
compile_fmu("Vessel", "Vessel.mo")
```

Figure 12: Python script for the creation of FMU models of the gradostat components

the first and second vessels is the same (i.e. $x_1 = x_2$). To check this we run the simulation, but this time with a constant flow rate $Dr = 3.6$ (see Fig. 15).

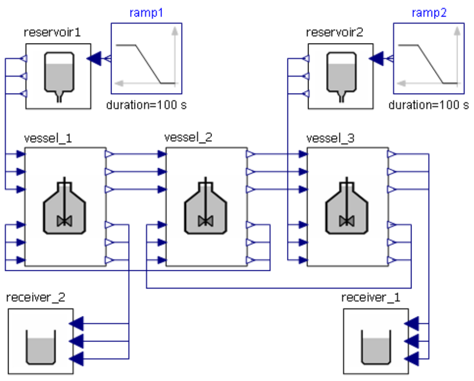


Figure 13: 3-vessels gradostat model in *SimulationX*

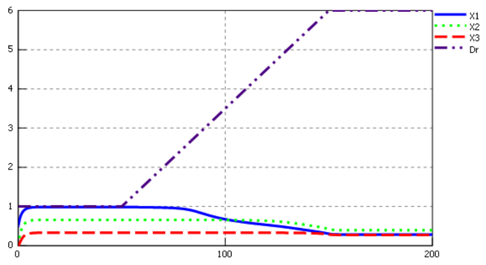


Figure 14: Result of the simulation of 3-vessels gradostat in *SimulationX*; a linear ramp for the flow rate *Dr*

6 Summary

System simulations allow the user to build models of complex systems. They are used and are usually very useful for modeling mechanical, thermal, electrical, hydraulic, pneumatic and other systems. In this paper we demonstrated that they are also useful for modeling and simulation of gradostat. The main advantage of the models of gradostats such as the ones in Fig. 7 and Fig. 13 is that they are easy to use, understand and modify even by people who are not so familiar with the mathematical part of the gradostat theory.

Another advantage of system simulation is the existence of the FMI standard for model exchange between different system simulation software tools. FMI give us an opportunity to create much more reusable compo-

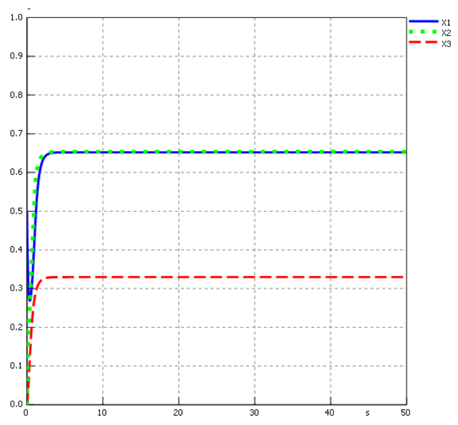


Figure 15: Result of the simulation of 3-vessels gradostat in *SimulationX*; $Dr = 3.6$

nents which can be used in different simulation software tools. Therefore, our future plans are to create a library with FMI components for simulation of gradostat and complex biological models which are suitable for system simulations.

Appendix

Some popular software tools for system simulations:

- **Simulink** [8] – developed by MathWorks, it is a data flow graphical programming language tool for modeling, simulating and analyzing multidomain dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in control theory and digital signal processing for multidomain simulation and Model-Based Design. It is a commercial software.

- **SimulationX** [5] – is a multi-domain system simulation software for virtual prototyping of physical systems. It is developed and sold commercially by ITI GmbH, based in Dresden, Germany. Scientists and engineers in industry and education use the tool for the design, analysis and virtual testing of complex mechatronics systems.

- **AMESim** [2] – is a graphical tool for modeling multi-domain systems using the Bond graph approach. It is developed by LMS, and is used primarily in the automotive industry amongst others for fuel injector simulations, fuel cell simulations, air-conditioning simulations and complete vehicle simulations. It is commercial simulation software.

- **MapleSim** [6] – is a multi-domain modeling and simulation tool developed by Maplesoft. MapleSim generates model equations, runs simulations, and performs analyses using the symbolic and numeric mathematical engine of Maple. Models are created by dragging components from a library and dropping them into a central workspace, resulting in a model that represents the physical system in a graphical form. Maplesoft began the development of MapleSim partly in response to a request from Toyota to produce physical modeling tools to aid in their new model-based development process. It is a commercial software tool, too.

- **OpenModelica** [11] – is an open-source Modelica-based modeling and simulation environment intended for industrial and academic usage. Its long-term development is supported by a non-profit organization – the Open Source Modelica Consortium (OSMC). The goal with the OpenModelica effort is to create a comprehensive Open Source Modelica modeling, compilation and simulation environment based on free software distributed in binary and source code form for research, teaching, and industrial usage.

- **JModelica** [1] – is a free and open-source platform based on the Modelica modeling language for modeling, simulation, optimization and analysis of complex dynamic systems. The platform is maintained and developed by Modelon AB in collaboration with academic and industrial institutions, notably Lund University and the Lund Center for Control of Complex Systems (LCCC). The platform has been used in industrial projects with applications in robotics, vehicle systems, energy systems and polyethylene production.

References

[1] Modelon AB. Jmodelica. www.jmodelica.org, November 2014.

[2] Siemens AG. Amesim. www.plm.automation.siemens.com/en_us/products/lms/imagine-lab/amesim/index.shtml, November 2014.

[3] Modelica Association. Modelica modeling language. www.modelica.org, November 2014.

[4] C. T. Codeco and J. P. Grover. Population dynamics of pseudomonas sp. along a spatial gradient of phosphate: an experimental approach for spatial ecology. *Revista Brasileira de Biologia*, 60(4), 2000.

[5] ITI GmbH. Simulationx. www.simulationx.com, November 2014.

[6] Waterloo Maple Inc. Maplesim. www.maplesoft.com/products/maplesim, November 2014.

[7] R. W. Lovitt and J. W. T. Wimpenny. The gradostat: A tool for investigating microbial growth and interactions in solute gradients. *Society of General Microbiology Quarterly*, (6):80, 1979.

[8] MathWorks. Simulink. www.mathworks.com/products/simulink, November 2014.

[9] MathWorks. Simulink: Choose a solver. www.mathworks.com /help/simulink/ug/choosing-a-solver.html, November 2014.

[10] MathWorks. Simulink: S-function basics. www.mathworks.com /help/simulink/s-function-basics.html, November 2014.

[11] Open Source Modelica Consortium (OSMC). Openmodelica. www.openmodelica.org, November 2014.

[12] Modelica Association Project. Functional mock-up interface. www.fmi-standard.org, November 2014.

[13] H. Smith and P. Waltman. *The Theory of the Chemostat: Dynamics of Microbial Competition.* Cambridge University Press, 1995.