Scripting Modelica Models using Python

Bernt Lie*, Finn Haugen

Telemark University College, Porsgrunn, P.O. Box 203, N-3901 Porsgrunn, Norway; *Bernt.Lie@hit.no

Simulation Notes Europe SNE 23(3-4), 2013, 161 - 170
DOI: 10.11128/sne.23.tn.10212
Received: March 10, 2013 (Selected SIM 2012 Postconf. Publ.);
Revised Accepted: November 15, 2013;

Abstract. Modelica has evolved as a powerful language for encoding models of complex systems. In control engineering, it is of interest to be able to analyze dynamic models using scripting languages such as MATLAB and Python. This paper illustrates some analysis and design methods relevant in control engineering through scripting a Modelica model of an anaerobic digester model using Python, and discusses advantages and shortcomings of the Python+Modelica set-up.

Introduction

Modelica is a modern language for describing large scale, multidisciplinary dynamic systems (Fritzson, 2011), and models can be built from model libraries or the user can develop her own models or libraries using a text editor and connect the submodels either using a text editor or a visual tool. Several commercial tools exist, such as Dymola, MapleSim, Wolfram SystemModeler, etc. Free/research based tools also exist, e.g. OpenModelica and JModelica.org. More tools are described at www.modelica.org.

For most applications of models, further analysis and post processing is required, including e.g. model calibration, sensitivity studies, optimization of design and operation, model simplification, etc. Although Modelica is a rich language, the lack of tools for analysis has been a weakness of the language as compared e.g. to MATLAB, etc. Two commercial products are thus based on integrating Modelica with Computer Algebra Systems (MapleSim, Wolfram SystemModeler), while for other tools the analysis part has been more cumbersome (although Dymola includes possibilities for model calibration, an early but simple way of controlling models from MATLAB, etc.).

A recent development has been the FMI standard, which promises to greatly simplify the possibility to script e.g. Modelica models from MATLAB or Python (FMI Toolbox for MATLAB; PyFMI for Python). Several Modelica tools now offer the opportunity to export models as FMUs (Functional Mock-up Units), whereupon PyFMI can be used to import the FMU into Python. Or the FMU can be directly generated from PyFMI. PyFMI is integrated into the JModelica.org tool. More extensive integration with Python is under way for other (free) tools, too.

Python 2.7 with packages Matplotlib, NumPy, and SciPy offer many tools for analysis of models; a simple installation is EPD Free , but many other installations exist.

It is of interest to study whether the combination of (free software) releases of Modelica and Python can serve as useful tools for control analysis and design studies, and what limitations currently limit the spread of such a package. This paper gives an overview of basic possibilities for doing model based control studies by scripting Modelica models from Python. As a case study, a model of an anaerobic digester for converting cow manure to biogas is presented in Section 1. Section 2 presents various examples of systems and control analysis carried out by Python scripts using the model encoded in Modelica. Finally, the results are discussed and some conclusions are drawn in Section 3.

1 Case Study

1.1 Functional description

Figure 1 illustrates the animal waste conversion systems at Foss Biolab in Skien, Norway, which converts cow manure into biogas utilizing Anaerobic Digestion (AD). In this case study, we consider the reactor only (blue box), where the *Feed* is decribed by a volumetric feed rate $\dot{V}_f[L/d]$ (control input) with a given concentration $\rho_{S_{VS,f}}$ of volatile solids (disturbance).

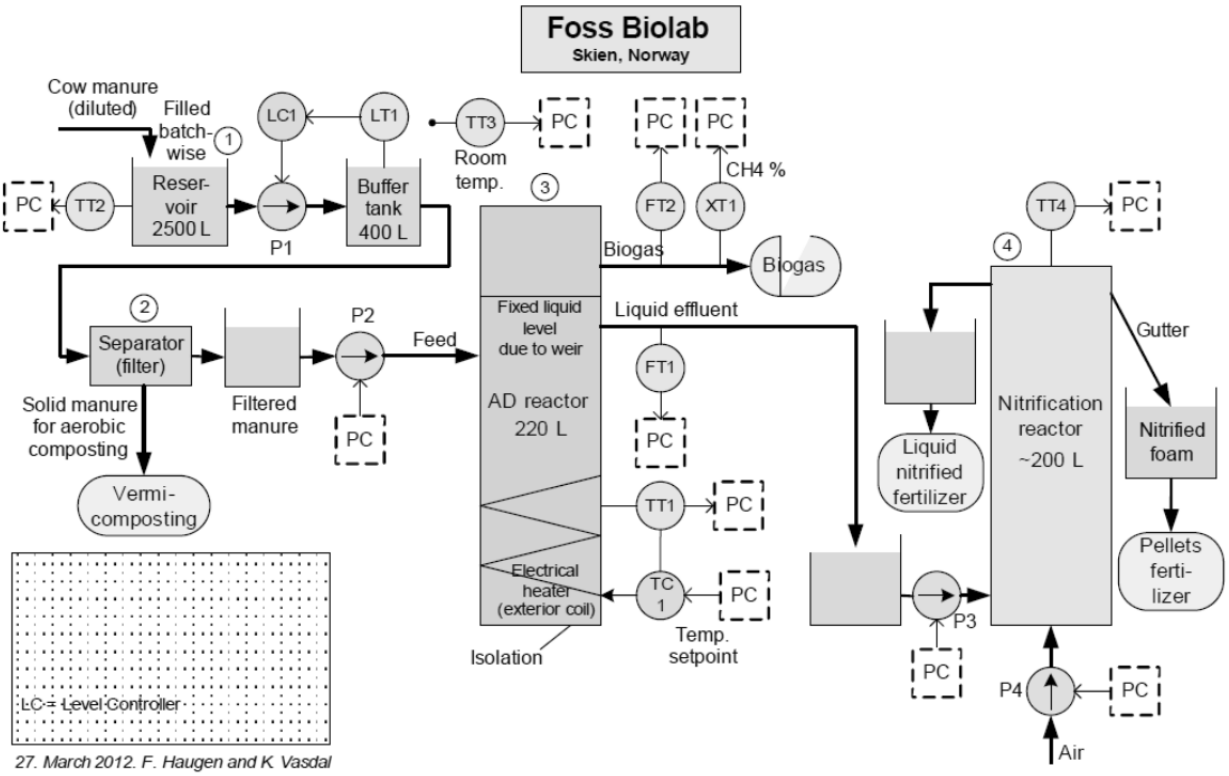

Figure 1: System for converting cow manure to biogas at Foss Biolab, Skien, Norway (Figure by F. Haugen and K. Vasdal).

The 'liquid' level of the reactor is made constant by the use of a weir system, and it is possible to control the reactor temperature T accurately using electric heating (potential control input). The main product considered here, is the mass flow rate of methane out of the reactor, $\dot{m}_{CH_{4,X}}$ (controlled variable).

1.2 Model summary

A model of the reactor is presented in Haugen et al. (2012); in this paper, the same model is used but with a modified notation. The operation of the bio reactor is described by four states $j \in \left\{\rho_{S_{bvs}}, \rho_{S_{vfa}}, \rho_{X_a}, \rho_{X_m}\right\}$:

$$\frac{d}{dt}\rho_j = \frac{1}{\theta_j}\frac{\dot{V}_f}{V}\left(\rho_{j,f} - \rho_j\right) + R_j$$

where V is constant due to perfect level control, the residence time correction $\theta_{S_j} = 1$ and θ_{X_j} may differ from 1, and furthermore:

$$R_{S_{bvs}} = -Y_{S_{bvs}/X_a} R_a$$

$$R_{S_{vfa}} = Y_{S_{vfa}/X_a} R_a - Y_{S_{vfa}/X_m} R_m$$

$$R_{X_a} = R_a - k_a^d \rho_{X_a}$$

$$R_{X_m} = R_m - k_m^d \rho_{X_m}$$

with

$$R_a = \mu_a \rho_{X_a}$$

$$R_m = \mu_m \rho_m$$

$$\mu_a = \frac{\hat{\mu}_a}{1 + K_{S_{bvs}} \dfrac{1}{\rho_{S_{bvs}}}}$$

$$\mu_m = \frac{\hat{\mu}_m}{1 + K_{S_{vfa}} \dfrac{1}{\rho_{S_{vfa}}}}$$

$$\hat{\mu}_a = \hat{\mu}_m = \hat{\mu}_{35} + \alpha_{\hat{\mu}}(T - 35)$$

with units °C for T.

The production (exit) rate of methane is given by

$$\dot{m}_{CH_4,X} = R_{CH_4} V$$

$$R_{CH_4} = Y_{CH_4/X_m} R_m.$$

Feed concentrations of states are given as

$$\rho_{S_{bvs},f} = b_0 \rho_{S_{vs},f}$$

$$\rho_{S_{vfa},f} = a_f \rho_{S_{bvs},f}.$$

Nominal operating conditions for the system are given in Table 1.

Quantity	Value	Unit	Comment
$\boldsymbol{\rho_{S_{bvs}}(0)}$	5.81	g/L	Initially dissolved substrate biodegradable volatile solids
$\boldsymbol{\rho_{S_{vfa}}(0)}$	1.13	g/L	Initially dissolved substrate volatile fatty acids
$\boldsymbol{\rho_{X_a}(0)}$	1.32	g/L	Initially concentration of acetogenic bacteria
$\boldsymbol{\rho_{X_m}(0)}$	0.39	g/L	Initially concentration of methanogenic bacteria
$\boldsymbol{\dot{V}_f}$	50	L/d	Volumetric feed flow of animal waste/manure
$\boldsymbol{T}$	35	℃	Reactor temperature
$\boldsymbol{\rho_{S_{vs,f}}}$	32.4	g/L	Feed concentration of volatile solids

Table 1: Nominal operational data for biogas reactor at Foss Biolab.

Model parameters are given in Table 2.

1.3 System and control problems

A number of control problems are relevant for this system:

- simulation of the system for validation,
- study of model sensitivity wrt. uncertain parameters,
- tuning model parameters to fit the model to experimental data,
- state estimation for computing hidden model states,
- operation of control system,
- optimal control and model predictive control,
- etc.
-

Only a selected few of these problems are considered in the sequel.

Parameter	Value	Unit	Comment
$\boldsymbol{V}$	250	L	Reactor volume
$\boldsymbol{\theta_{X_a} = \theta_{X_m}}$	2.9	-	Correction of residence time for bacteria due to nonideal flow
$\boldsymbol{Y_{S_{bvs}/X_a}}$	3.9	$\frac{gbvs}{gX_a}$	(Inverse) yield: consumption of bvs per growth of bacteria
$\boldsymbol{Y_{S_{vfa}/X_a}}$	1.76	$\frac{gvfa}{gX_a}$	(Inverse) yield: production of vfa per growth of bacteria
$\boldsymbol{Y_{S_{vfa}/X_m}}$	31.7	$\frac{gvfa}{gX_m}$	(Inverse) yield: consumption of vfa per growth of bacteria
$\boldsymbol{Y_{CH_4/X_m}}$	26.3	$\frac{gCH_4}{gX_m}$	(Inverse) yield: production of methane per growth of bacteria
$\boldsymbol{K_{S_{bvs}}}$	15.5	g/L	Half-velocity constant for bvs substrate
$\boldsymbol{K_{S_{vfa}}}$	3.0	g/L	Half-velocity constant for vfa substrate
$\boldsymbol{\hat{\mu}_{35}}$	0.326	d^{-1}	Maximal growth of rate at $T = 35$℃
$\boldsymbol{a_{\hat{\mu}}}$	0.013	$\frac{1}{℃d}$	Temperature sensitivity of maximal growth rate, valid $T \in [20,60]$℃
$\boldsymbol{k_a^d = k_m^d}$	0.02	d^{-1}	Death rate constants for acetogenic and methanogenic bacteria
$\boldsymbol{b_0}$	0.25	$\frac{gbvs}{gvs}$	Fraction biodegradable volatile solids in volatile solids feed
$\boldsymbol{a_f}$	0.69	$\frac{gvfa}{gbvs}$	Fraction volatile fatty acids in biodegradable volatile solids feed

Table 2: Nominal model parameters for biogas reactor at Foss Biolab.

2 Control Relevant Analysis

2.1 Basic Modelica description

The following Modelica encoding in file `adFoss.mo` describes the basic model:

```
Model adFossModel
// Simulation of Anaerobic Digestion Reactor at Foss
//Biolab
// Author: Bernt Lie
// Telemark University College, Porsgrunn, Norway
// August 31, 2012
// Parameter values with type and descriptive text
parameter Real V = 250 "reactor volume,
  L";
parameter Real theta_X = 2.9 "residence
  time correction for bacteria,
```

```
dimensionless";
parameter Real Y_Sbvs_Xa = 3.9 "Yield, g
 bvs/g acetogens";
parameter Real Y_Svfa_Xa = 1.76 "Yield, g
 vfa/g acetogens";
parameter Real Y_Svfa_Xm = 31.7 "Yield, g
 vfa/g methanogens";
parameter Real Y_CH4_Xm = 26.3 "Yield, g
 methane/g methanogens";
parameter Real K_Sbvs = 15.5 "Half-
 velocity constant for bvs, g/L";
parameter Real K_Svfa = 3.0 "Half-
 velocity constant for vfa, g/L";
parameter Real muhat_35 = 0.326 "Maximal
 growth rate at T=35 C, 1/d";
parameter Real alpha_muhat = 0.013 "Tem-
 perature sensitivity of
max growth rate, 1/(C d)";
parameter Real k_d = 0.02 "Death rate
 constants for bacteria, 1/d";
parameter Real b0 = 0.25 "Fraction biode-
 gradable volatile solids in
volatile solids feed, g bvs/g vs";
parameter Real af = 0.69 "Fraction vola-
 tile fatty acids in bvs feed,
g vfa/g bvs";
// Initial state parameters:
parameter Real rhoSbvs0 = 5.81 "initial
 bvs substrate, g/L";
parameter Real rhoSvfa0 = 1.13 "initial
 vfa, g/L";
parameter Real rhoXa0 = 1.32 "initial
 acetogens, g/L";
parameter Real rhoXm0 = 0.39 "initial
 methanogens, g/L";
// Setting initial values for states:
Real rhoSbvs(start = rhoSbvs0, fixed =
 true);
Real rhoSvfa(start = rhoSvfa0, fixed =
 true);
Real rhoXa(start = rhoXa0, fixed = true);
Real rhoXm(start = rhoXm0, fixed = true);
// Miscellaneous variables
Real rhoSbvs_f "feed concentration of
 bvs, g/L";
Real rhoSvfa_f "feed concentration of
 vfa, g/L";
Real rhoXa_f "feed concentration of
 acetogens, g/L";
4
Real rhoXm_f "feed concentration of meth-
 anogens, g/L";
Real R_Sbvs "generation rate of Sbvs,
 g/(L*d)";
Real R_Svfa "generation rate of Svfa,
 g/(L*d)";
Real R_Xa "generation rate of Xa,
 g/(L*d)";
Real R_Xm "generation rate of Xm,
 g/(L*d)";
Real R_CH4 "generation rate of CH4,
 g/(L*d)";
Real R_a "reaction rate acetogenesis,
 g/(L*d)";
Real R_m "reaction rate methanogenesis,
 g/(L*d)";
Real mu_a "growth rate acetogenesis,
 1/d";
Real mu_m "growth rate methanogenesis,
 1/d";
Real muhat_a "maximal growth rate aceto-
 genesis, 1/d";
Real muhat_m "maximal growth rate methan-
 ogenesis, 1/d";
Real mdot_CH4x "mass flow methane produc-
 tion, g/d";
// Defining input variables:
input Real Vdot_f "volumetric feed flow -
 - control variable, L/d";
input Real T "reactor temperature -- pos-
 sible control input, C";
input Real rhoSvs_f "feed volatile solids
 concentration -- disturbance, g/L";
equation
// Differential equations
der(rhoSbvs) = Vdot_f/V*(rhoSbvs_f -
 rhoSbvs) + R_Sbvs;
der(rhoSvfa) = Vdot_f/V*(rhoSvfa_f -
 rhoSvfa) + R_Svfa;
der(rhoXa) = Vdot_f/V/theta_X*(rhoXa_f -
 rhoXa) + R_Xa;
der(rhoXm) = Vdot_f/V/theta_X*(rhoXm_f -
 rhoXm) + R_Xm;
// Feed
rhoSbvs_f = rhoSvs_f*b0;
rhoSvfa_f = rhoSbvs_f*af;
rhoXa_f = 0;
rhoXm_f = 0;
// Generation rates
R_Sbvs = -Y_Sbvs_Xa*R_a;
R_Svfa = Y_Svfa_Xa*R_a - Y_Svfa_Xm*R_m;
R_Xa = R_a - k_d*rhoXa;
R_Xm = R_m - k_d*rhoXm;
R_a = mu_a*rhoXa;
```

```
R_m = mu_m*rhoXm;
mu_a = muhat_a/(1 + K_Sbvs/rhoSbvs);
mu_m = muhat_m/(1 + K_Svfa/rhoSvfa);
muhat_a = muhat_35 + alpha_muhat*(T-35);
muhat_m = muhat_a;
// Methane production
mdot_CH4x = R_CH4*V;
R_CH4 = Y_CH4_Xm*R_m;
end adFossModel;
```

2.2 Basic Python script

The following Python script `adFossSim.py` provides basic simulation of the Anaerobic Digester reactor at Foss Biolab starting at the nominal operating point, and performing some step perturbations for the inputs:

```
# Python script for simulating Anaerobic Digester at
#Foss Biolab
# script: adFossSim.py
# author: Bernt Lie, Telemark University College,
#Porsgrunn, Norway
# location: Telemark University College, Porsgrunn
# date: August 31, 2012
# Importing modules
# matplotlib, numpy
import matplotlib.pyplot as plt
import numpy as np
# JModelica
from pymodelica import compile_fmu
from pyfmi import FMUModel
# Flattening, compiling and exporting model as fmu
adFoss_fmu = compile_fmu("adFossModel",
  "adFoss.mo")
# Importing fmu and linking it with solvers, etc.
adFoss = FMUModel(adFoss_fmu)
# Creating input data
t_fin = 100
adFoss_opdata =
  np.array([[0,50,35,32.4],[10,50,35,32.4]
  ,[10,45,35,32.4],
[30,45,35,32.4],[30,45,38,32.4],[60,45,38
  ,32.4],
[60,45,38,40],[t_fin,45,38,40]])
adFoss_input = (["Vdot_f", "T",
  "rhoSvs_f"], adFoss_opdata)
# Carrying out simulation
adFoss_res = adFoss.simulate(final_time =
  t_fin, input = adFoss_input)
# Unpacking results
rhoSbvs = adFoss_res["rhoSbvs"]
rhoSvfa = adFoss_res["rhoSvfa"]
rhoXa = adFoss_res["rhoXa"]
rhoXm = adFoss_res["rhoXm"]
mdot_CH4x = adFoss_res["mdot_CH4x"]
Vdot_f = adFoss_res["Vdot_f"]
T = adFoss_res["T"]
rhoSvs_f = adFoss_res["rhoSvs_f"]
t = adFoss_res["time"]
# Setting up figure with plot of results
plt.figure(1)
plt.plot(t,rhoSbvs,"-r",t,rhoSvfa,"-
  g",t,rhoXa,"-k",t,rhoXm,"-
  b",linewidth=2)
plt.legend((r"$\rho_{S_{bvs}}$
  [g/L]",r"$\rho_{S_{vfa}}$ [g/L]",
r"$\rho_{X_a}$ [g/L]",r"$\rho_{X_m}$
  [g/L]"),ncol=2,loc=0)
plt.title("Anaerobic Digestion at Foss
  Biolab")
plt.xlabel(r"time $t$ [d]")
plt.grid(True)
plt.figure(2)
plt.plot(t,mdot_CH4x,"-r",linewidth=2)
plt.title("Anaerobic Digestion at Foss
  Biolab")
plt.ylabel(r"$\dot{m}_{CH_4}$ [g/d]")
plt.xlabel(r"time $t$ [d]")
plt.grid(True)
plt.figure(3)
plt.plot(t,Vdot_f,"-r",t,T,"-
  g",t,rhoSvs_f,"-b",linewidth=2)
plt.axis(ymin=30,ymax=55)
plt.title("Anaerobic Digestion at Foss
  Biolab")
```

Running this Python script leads to the results in figures 2-4:

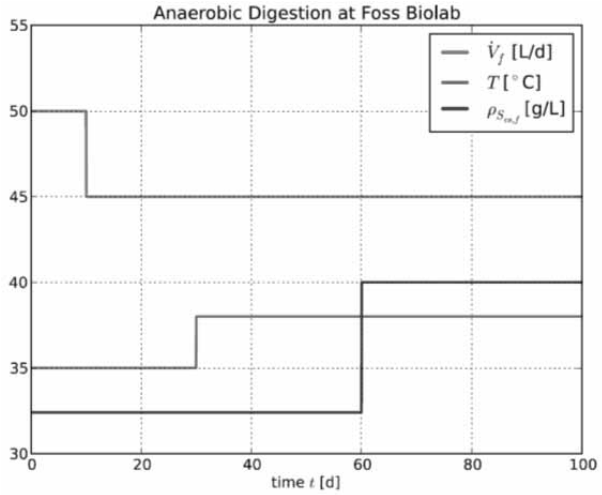

Figure 2: Nominal evolution of inputs at Foss Biolab, with perturbation.

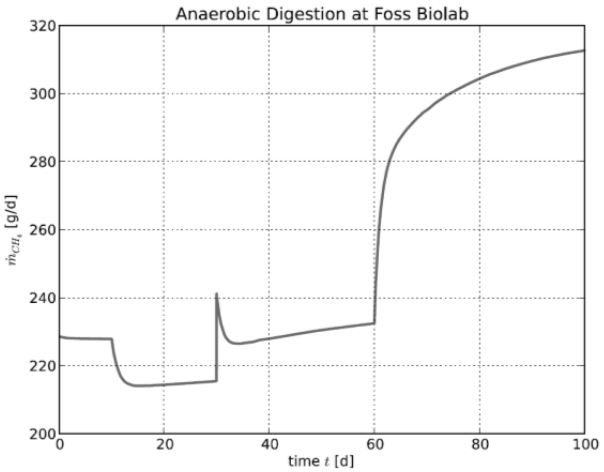

Figure 3: Nominal production of methane gas at Foss Biolab, with perturbation.

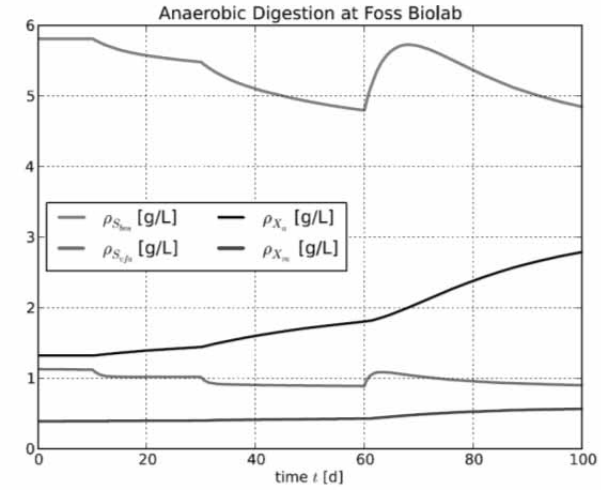

Figure 4: Nominal evolution of states at Foss Biolab, with perturbation.

2.3 Uncertainty analysis

Suppose the value of parameters b_0 and a_f are uncertain, but that we 'know' they lie in intervals $b_0 \in 0.25 \times [0.9,1.1]$ and $a_f \in 0.69 \times [0.9,1.1]$. We can study the uncertainty of the model by running a number N_{MC} of Monte Carlo simulations were we draw values at random from these two ranges – e.g. assuming uniform distribution. The following modifications of the Python code will handle this problem, excerpt of script `adFossSimMC.py`:

```
# Python script for Monte Carlo study of Anaerobic Digester
at Foss Biolab
# script: adFossSimMC.py
# author: Bernt Lie, Telemark University College, Porsgrunn,
  Norway
# location: Telemark University College, Porsgrunn
# date: August 31, 2012
# Importing modules
# matplotlib, numpy, random
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as nr
...
# Carrying out simulation
adFoss_res = adFoss.simulate(final_time =
  t_fin, input = adFoss_input)
...
# Setting up figure with plot of results
plt.figure(1)
plt.plot(t,rhoSbvs,"-r",t,rhoSvfa,"-
  g",t,rhoXa,"-k",t,rhoXm,"-b",linewidth=2)
plt.legend((r"$\rho_{S_{bvs}}$
  [g/L]",r"$\rho_{S_{vfa}}$ [g/L]",
r"$\rho_{X_a}$ [g/L]",r"$\rho_{X_m}$
  [g/L]"),ncol=2,loc=0)
plt.title("Anaerobic Digestion at Foss Bi-
  olab")
plt.xlabel(r"time $t$ [d]")
plt.grid(True)
...
# Monte Carlo simulations
Nmc = 20
b0nom = adFoss.get("b0")
afnom = adFoss.get("af")
for i in range(Nmc):
b0 = b0nom*(1 + 0.1*(nr.rand()-0.5)*2)
af = afnom*(1 + 0.1*(nr.rand()-0.5)*2)
adFoss.set(["b0","af"],[b0,af])
# Carrying out simulation
adFoss_res = adFoss.simulate(final_time =
  t_fin, input = adFoss_input)
# Unpacking results
rhoSbvs = adFoss_res["rhoSbvs"]
rhoSvfa = adFoss_res["rhoSvfa"]
rhoXa = adFoss_res["rhoXa"]
rhoXm = adFoss_res["rhoXm"]
mdot_CH4x = adFoss_res["mdot_CH4x"]
t = adFoss_res["time"]
# Setting up figure with plot of results
plt.figure(1)
plt.plot(t,rhoSbvs,":r",t,rhoSvfa,":g",t,rho
  Xa,":k",t,rhoXm,":b",
linewidth=1.5)
plt.figure(2)
plt.plot(t,mdot_CH4x,":m",linewidth=1.5)
plt.show()
```

The results are as shown in figures 5 and 6.

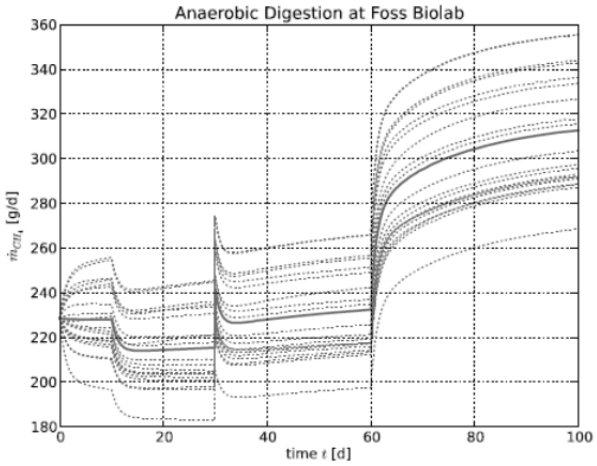

Figure 5: Monte Carlo study of methane production at Foss Biolab, with variation in b_0 and a_f.

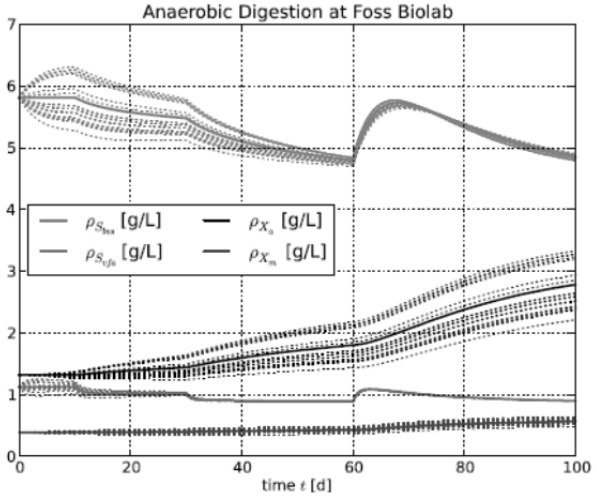

Figure 6: Monte Carlo study of evolution of states at Foss Biolab, with variation in b_0 and a_f .

2.4 Wash-out and recovery of reactor

Suppose that the reactor gets 'washed out' by accidentally applying too high a feed rate $\dot{V}_f$, e.g. $\dot{V}_f = 120L/d$, while T and $\rho_{S_{vs,f}}$ are as in Table 1. It is of interest to see whether the original production can be recovered. Figures 7–9 indicates the behavior over a period of more than 4 years ($1500\ d$) of operation.

As seen, although increasing $\dot{V}_f$ initially leads to a significant increase in the methane production, the bacteria are washed out of the reactor leading to a dramatic fall in the methane production. Furthermore, it takes an inordinate long time to recover after a wash-out if the input is simply set back to the original flow rate.

The steady state values at wash-out ($t = 400\ d$) can be found to be

$$\rho_{S_{bvs},wash-out} = 8.0999999985826001$$

$$\rho_{S_{vfa},wash-out} = 3.96169944436781$$

$$\rho_{X_a,wash-out} = 1.3193454767561001 \times 10^{-9}$$

$$\rho_{X_m,wash-out} = 0.13282069444970099$$

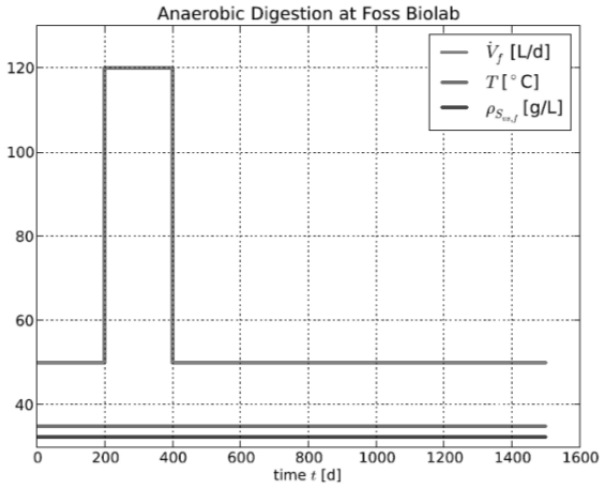

Figure 7: Evolution of inputs at Foss Biolab leading to wash-out/recovery.

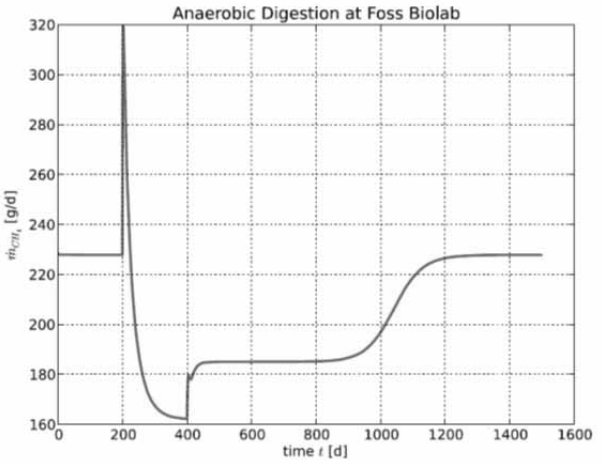

Figure 8: Production of methane gas at Foss Biolab during wash-out/recovery.

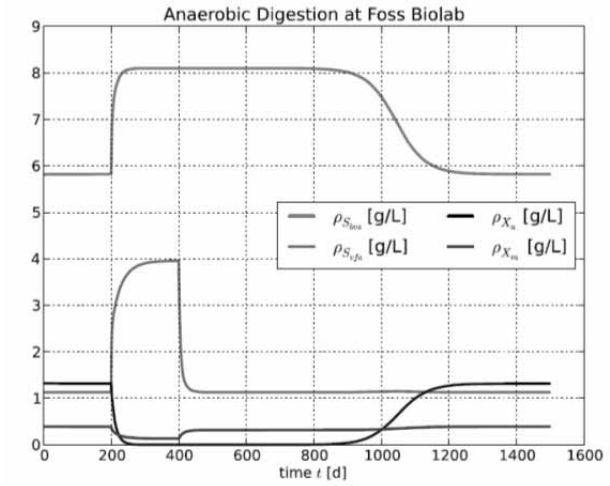

Figure 9: Evolution of states at Foss Biolab during wash-out/recovery.

2.5 Optimal recovery of methane production

The accidental wash-out of bacteria is a serious problem in the operation of Anaerobic Digesters. It is thus of interest to see whether it is possible to recover the operation in an optimal way. We consider the possibility of recovering the operation in the 1100 d horizon spent to wash-out the bacteria, …g. 7 –9. We thus seek to maximize the production of methane, but without using too much feed of animal waste. The following criterion is thus sought *maximized*:

$$J = \int_0^{T_h} \left(\dot{m}_{CH_4,X} - c_{\dot{V}} \dot{V}_f \right) dt$$

where $c_{\dot{V}}$ is a cost parameter. We add the following constraints to make sure that the solution has physical meaning.

$$\rho_j \geq 0$$

$$\dot{m}_{CH_{4,X}} \geq 0$$

$$\dot{V}_f \in [0{,}120] L/d.$$

We assume that the temperature T and the disturbance $\rho_{S_{vs,f}}$ are as in Table 1.

To solve this problem, we use the Modelica extension class *optimization* in JModelica.org. In Modelica, the criterion function is *minimized*, so the criterion in Modelica needs to be $-J$ where J is as above. The essence of the Modelica code for this problem is as given below:

```
optimization adFossOpt(objective =
  J(finalTime), startTime=0,
  finalTime=T_h)
// Optimal recovery of Anaerobic Digestion Reactor at
  //Foss Biolab
// Author: Bernt Lie
// Telemark University College, Porsgrunn, Norway
// September 2, 2012
// Instantiating model adf from class adFossModel
adFossModel adf;
// Additional parameters
parameter Real T_h = 1100 "time horizon
  in optimization criterion, d";
parameter Real cost_V = 1 "relative cost
  of animal waste";
parameter Real Vdot_max = 120 "maximal
  allowed feed rate, L/d";
parameter Real T_nom = 35 "nominal reac-
  tor temperature, C";
parameter Real rhoSvs_f_nom = 32.4
  "nominal feed concentration
of volatile solids, g/L";
// Defining cost function
Real J(start=0, fixed=true);
// Defining input variable:
input Real Vdot_f(free=true,
  min=0,max=Vdot_max) "max feed flow,
  L/d";
equation
// Passing on inputs to model instance
adf.Vdot_f = Vdot_f;
adf.T = T_nom;
adf.rhoSvs_f = rhoSvs_f_nom;
// Computing cost function
der(J) = - adf.mdot_CH4x + cost_V*Vdot_f;
constraint
// Constraining states
adf.rhoSbvs >= 0;
adf.rhoSvfa >= 0;
adf.rhoXa >= 0;
adf.rhoXm >= 0;
// Constraining methane production
adf.mdot_CH4x >=0;
end adFossOpt;
```

With $c_{\dot{V}}$, the result is as in figures 10 – 12. With $c_{\dot{V}}$, the result is highly oscillatoric time evolutions.

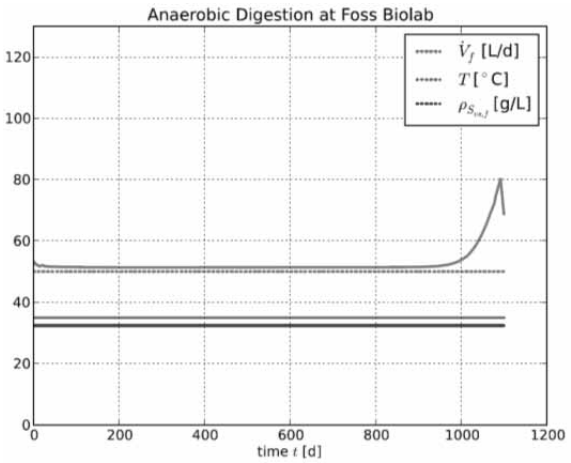

Figure 10: Evolution of optimal input $\dot{V}_f$ at Foss Biolab after wash-out (solid lines), with initial guess (dotted lines).

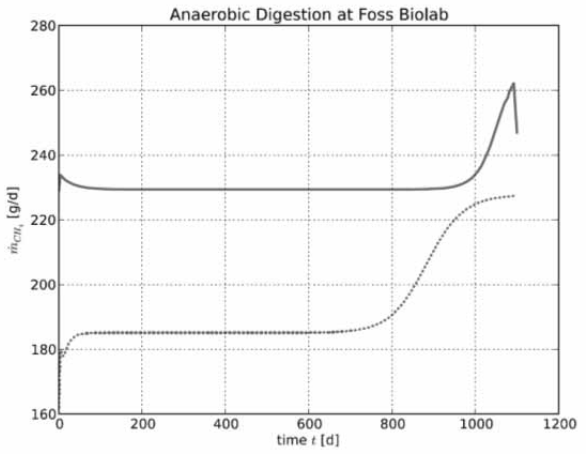

Figure 11: Evolution of optimally recovered methane production at Foss Biolab after wash-out (solidlines), with initial guess (dotted lines).

3 Discussion and Conclusions

Comparing Python to MATLAB for use in control studies reveals clear advantages and clear disadvantages for Python. Python is a free tool, and a rich programming language. However, there is (currently) no control toolbox for Python, the various packages and sub packages are not so well documented, and the quality of some tools are far from perfect. Yet, the combination of Python and Modelica/PyFMI offers ample opportunities for analysis of models and control studies.

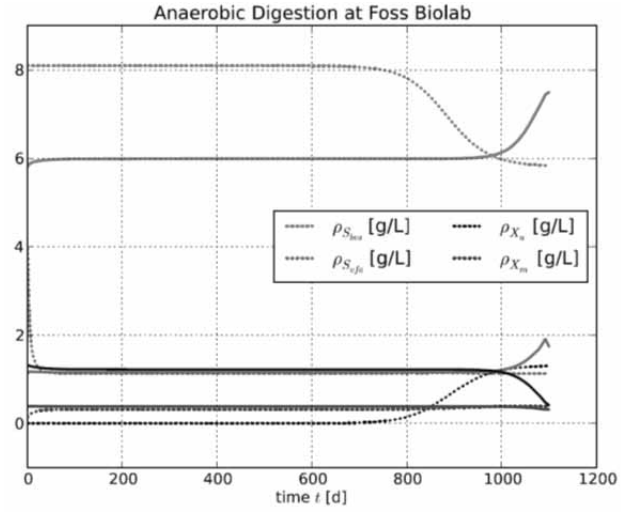

Figure 12: Evolution of optimally recovered states at Foss Biolab after wash-out (solid lines), with initial guess (dotted lines).

This paper illustrates this by showing how natural models can be encoded in Modelica, and how easy Modelica models can be accessed from Python using e.g. PyFMI. Furthermore, it is shown how natural and powerful Python is as a scripting language, e.g. for doing uncertainty/sensitivity analysis of dynamic models. Finally, a simple optimal control problem illustrates on-going research and development in extending the Modelica language using JModelica.org; similar extensions of the Modelica language are also studied in e.g. Bachmann et al.(2012). And yet, in this paper, only the most rudimentary use of Modelica and Python has been touched upon.

Currently, some key problems with the Python+Modelica combination are:

- There is no equivalent of MATLAB's Control Toolbox. This is such a shortcoming that many control engineers will not seriously consider the Python + Modelica combination. Some work at CalTech aims to resolve this problem by developing a Python control toolbox, but there does not appear to be a clear timeline for such a toolbox. Within the Modelica groups, some on-going work addresses this by developing a Linear Systems library within Modelica.

- Although there are a number of powerful (and free) optimization solvers, it is not trivial to integrate these into Python, and those which already have simple Python installers are often poorly documented and/or uses non-standard array packages. A minimal package should include LP, QP, NLP, and NLS solvers of high quality, and they should be equally simple to install in the main OS platforms.
- The FMI is a very positive initiative, and well suited to scripting using either Python or MATLAB. More work is needed in order to make FMI export from the various tools more standardized.
- The initiative of extending Modelica with optimization (and model calibration) possibilities is very interesting for the control society. It would be even more interesting if some standards evolve.

The evolution of alternatives to MATLAB + SIMULINK is very interesting, and Python + Modelica holds promise to be such a tool. There are advantages with commercial tools such as MATLAB+ SIMULINK and similar tools for Modelica such as MapleSim and Wolfram SystemModeler, but inacademia with limited resources for buying software, free software is of interest.

Acknowledgement

This contribution is a post-conference publication from SIMS 2012 Conference (53rd SIMS Conference, Reykjavík, Iceland, October 4 - 6, 2012). The contribution is a (partly) modified publication from the paper published in the Proceedings of SIMS 2012, published by Orkustofnun, National Energy Authority Iceland, ISBN: 978-9979-68-318-6, electronically available at http://www.scansims.org/sims2012/SIMS2012.pdf.

References

[1] Bachmann B, Ochel L, Ruge V, Gebremedhin M, Fritzson P, Nezhadali V, Eriksson L, Sivertsson M. (2012). Parallel Multiple-Shooting and Collocation Optimization with OpenModelica. Modelica 2012. *Proceedings of the 9th International Modelica Conference*; Munich.

[2] Fritzson, P. Introduction to Modeling and Simulation of Technical and Physical Systems withModelica R. *IEEE*. 2011; Hoboken.

[3] Haugen F, Bakke R, and Lie B. Mathematical Modelling for Planning Optimal Operation of a Biogas Reactor for Dairy Manure. Presented at the *IWA World Congress on Water, Climate and Energy (IWA-WCE);* 2012; Dublin.