

BPL_TEST2_Perfusion - demo

This notebook explore perfusion cultivation in comparison with ordinary continuous cultivation (chemostat) and use comparable settings to earlier notebook. Further you see here examples of interaction with the simplified commands `par()`, `init()`, `simu()` etc as well as direct interaction with the FMU which is called "model" here. The last simulation is always available in the workspace and called "sim_res". Note that `describe()` brings mainly up from descriptive information from the Modelica code from the FMU but is complemented by some information given in the Python setup file.

```
In [1]: run -i BPL_TEST2_Perfusion_fmpy_explore.py
```

Windows - run FMU pre-compiled JModelica 2.14

Model for bioreactor has been setup. Key commands:

- `par()` - change of parameters and initial values
- `init()` - change initial values only
- `simu()` - simulate and plot
- `newplot()` - make a new plot
- `show()` - show plot from previous simulation
- `disp()` - display parameters and initial values from the last simulation
- `describe()` - describe culture, broth, parameters, variables with values/units

Note that both `disp()` and `describe()` takes values from the last simulation and the command `process_diagram()` brings up the main configuration

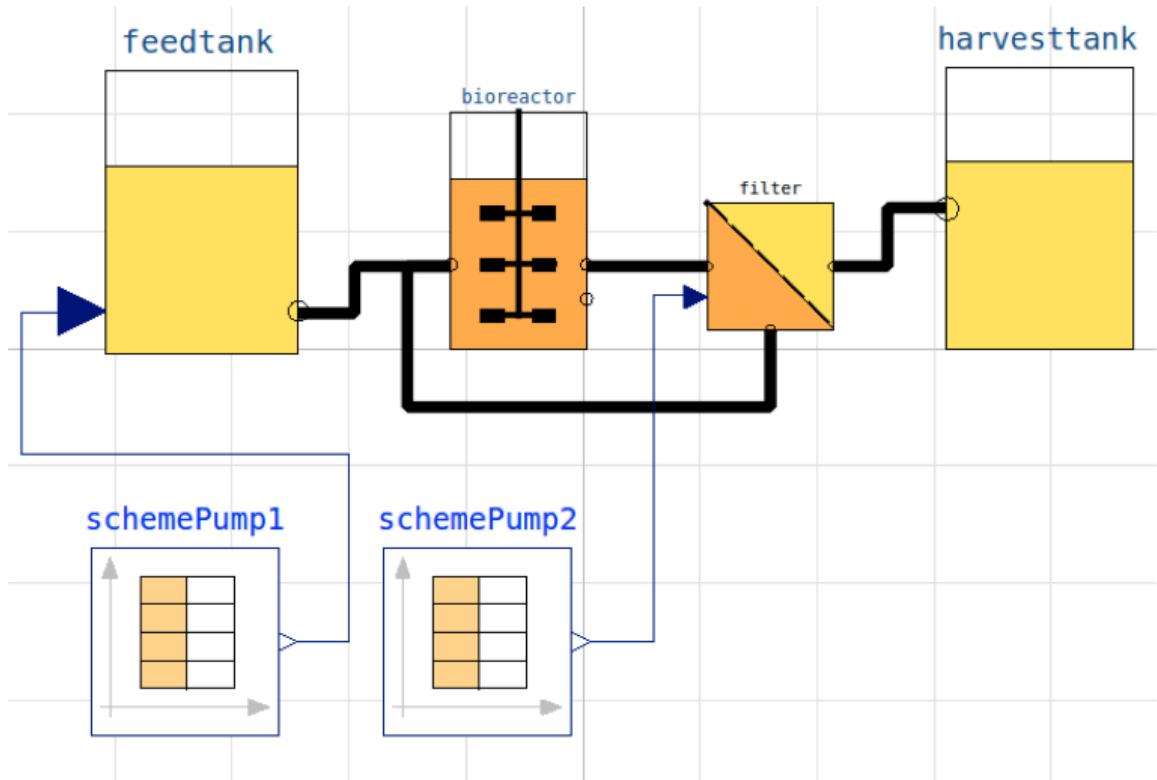
Brief information about a command by `help()`, eg `help(simu)`

Key system information is listed with the command `system_info()`

```
In [2]: %matplotlib inline
plt.rcParams['figure.figsize'] = [25/2.54, 20/2.54]
```

```
In [3]: process_diagram()
```

No processDiagram.png file in the FMU, but try the file on disk.



```
In [4]: # Process parameters used throughout
par(Y=0.5, qSmax=0.75, Ks=0.1) # Culture
par(filter_eps=0.10, filter_alpha_X=0.02, filter_alpha_S=0.10) # Filter
par(S_in=30.0) # Inlet subs
init(V_start=1.0, VX_start=1.0) # Process in
eps = parDict['filter_eps'] # Pump sched
```

```
In [5]: # Simulation of process with flow rate clot to wash-out for chemostat

init(VS_start=20) # Process initial
par(pump1_t1=10, pump2_t1=10) # Pump schedule - recyc
par(pump1_F1=2.5*0.155, pump2_F1=2.5*0.155/eps)
par(pump1_t2=940, pump2_t2=940, pump1_t3=950, pump2_t3=950, pump1_t4=960, pump2_

newplot(title='Perfusion cultivation - flow rate corresponding to maximal rate f
simu(10)
```

```

-----
FMIException                                         Traceback (most recent call last)
Cell In[5], line 9
      6 par(pump1_t2=940, pump2_t2=940, pump1_t3=950, pump2_t3=950, pump1_t4=960,
      7 pump2_t4=960)
      8 newplot(title='Perfusion cultivation - flow rate corresponding to maximal
rate for chemostat')
----> 9 simu(10)

File \\VBoxSvr\\Modelica\\GitHub\\Colab\\BPL_TEST2_Perfusion\\BPL_TEST2_Perfusion_fmpy
_explore.py:567, in simu(simulationTime, mode, options, diagrams)
    564 start_values = {parLocation[k]:parDict[k] for k in parDict.keys()}
    565 # Simulate
--> 567 sim_res = simulate_fmu(
    568     filename = fmu_model,
    569     validate = False,
    570     start_time = 0,
    571     stop_time = simulationTime,
    572     output_interval = simulationTime/options['ncp'],
    573     record_events = True,
    574     start_values = start_values,
    575     fmi_call_logger = None,
    576     output = list(set(extract_variables(diagrams) + list(stateDict.keys
(
    577         )) + key_variables))
    578 )
    579 simulationDone = True
    581 elif mode in ['Continued', 'continued', 'cont']:

File ~\\miniconda3\\envs\\fmpy\\lib\\site-packages\\fmpy\\simulation.py:761, in simulate
_fmu(filename, validate, start_time, stop_time, solver, step_size, relative_toler
ance, output_interval, record_events, fmi_type, start_values, apply_default_start
_values, input, output, timeout, debug_logging, visible, logger, fmi_call_logger,
step_finished, model_description, fmu_instance, set_input_derivatives, remote_pla
tform, early_return_allowed, use_event_mode, initialize, terminate, fmu_state, se
t_stop_time)
    759 result = simulateME(model_description, fmu, start_time, stop_time, so
lver, step_size, relative_tolerance, start_values, apply_default_start_values, in
put, output, output_interval, record_events, timeout, step_finished, validate, se
t_stop_time)
    760 elif fmi_type == 'CoSimulation':
--> 761     result = simulateCS(model_description, fmu, start_time, stop_time, re
lative_tolerance, start_values, apply_default_start_values, input, output, output
_interval, timeout, step_finished, set_input_derivatives, use_event_mode, early_r
eturn_allowed, validate, initialize, terminate, set_stop_time)
    763 if fmu_instance is None:
    764     fmu.freeInstance()

File ~\\miniconda3\\envs\\fmpy\\lib\\site-packages\\fmpy\\simulation.py:1272, in simulat
eCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_valu
es, apply_default_start_values, input_signals, output, output_interval, timeout,
step_finished, set_input_derivatives, use_event_mode, early_return_allowed, valid
ate, initialize, terminate, set_stop_time)
    1270         break
    1271     else:
-> 1272         raise e
    1273 else:
    1275     t_input_event = input.nextEvent(time)

File ~\\miniconda3\\envs\\fmpy\\lib\\site-packages\\fmpy\\simulation.py:1258, in simulat
eCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_valu

```

```

es, apply_default_start_values, input_signals, output, output_interval, timeout,
step_finished, set_input_derivatives, use_event_mode, early_return_allowed, validate,
initialize, terminate, set_stop_time)

```

```

1256 try:
1257     if time + output_interval <= stop_time:
-> 1258         fmu.doStep(currentCommunicationPoint=time, communicationStepSize=
output_interval)
1259         n_steps += 1
1260         time = n_steps * output_interval

```

File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\fmi2.py:580, in FMU2Slave.doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint)

```

579 def doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint=fmi2True):
--> 580     self.fmi2DoStep(self.component, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint)

```

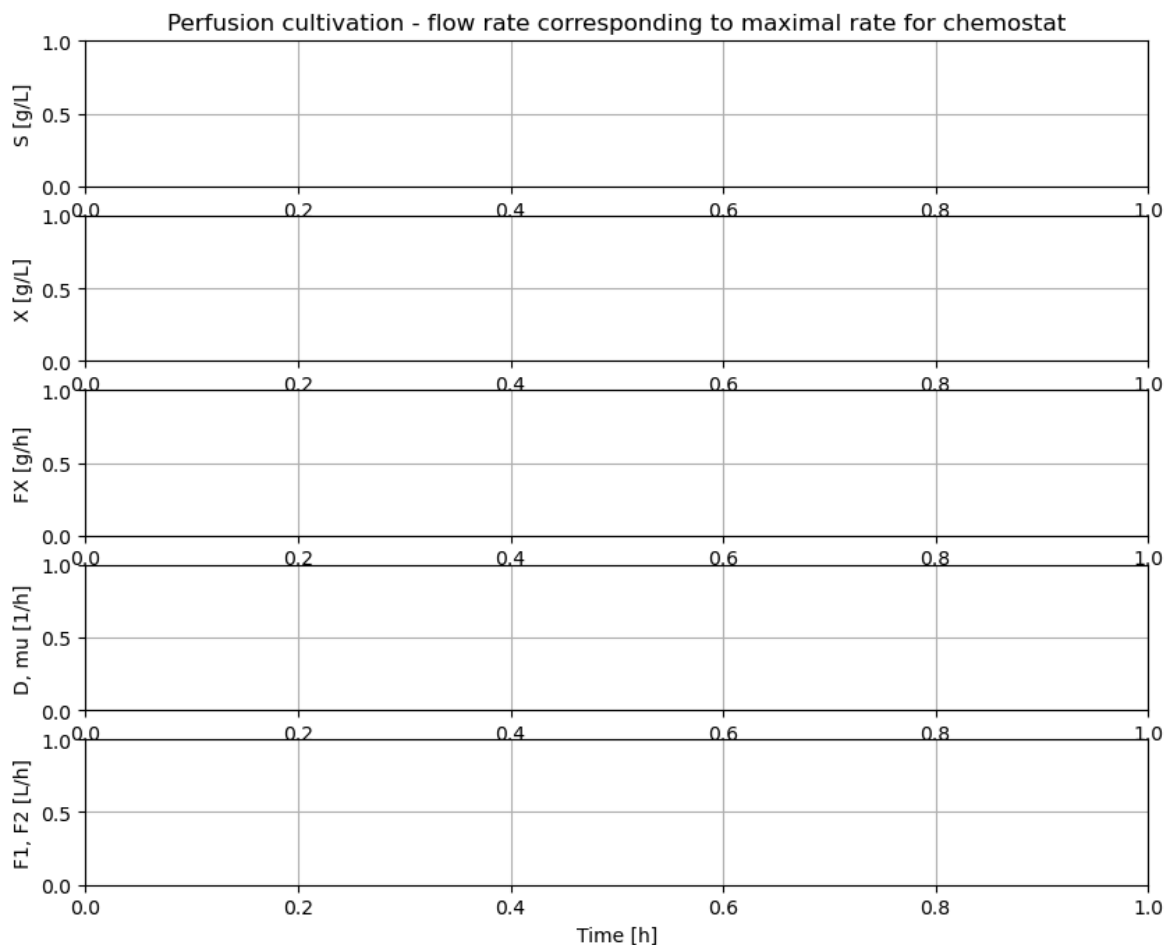
File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\fmi2.py:215, in _FMU2._fmi2Function.<locals>.w(*args)

```

212 if restype == fmi2Status: # status code
213     # check the status code
214     if res > fmi2Warning:
--> 215         raise FMICallException(function=fname, status=res)
217 return res

```

FMICallException: fmi2DoStep failed with status 3 (error).



```

In [6]: # Concentration factor of the filter
c=model_get('filter.retentate.c[1]')/model_get('filter.inlet.c[1]')
print('Conc factor of perfusion filter =', np.round(c,3))

```

Error: Information available after first simulation

Error: Information available after first simulation

TypeError Traceback (most recent call last)

Cell In[6], line 2

```
1 # Concentration factor of the filter
----> 2 c=model_get('filter.retentate.c[1]')/model_get('filter.inlet.c[1]')
      3 print('Conc factor of perfusion filter =', np.round(c,3))
```

TypeError: unsupported operand type(s) for /: 'NoneType' and 'NoneType'

```
In [7]: c_data=sim_res['filter.retentate.c[1]']/sim_res['filter.inlet.c[1]']
        print('Conc factor variation', np.round(min(c_data[151:]), 3),'to', np.round(max
```

NameError Traceback (most recent call last)

Cell In[7], line 1

```
----> 1 c_data=sim_res['filter.retentate.c[1]']/sim_res['filter.inlet.c[1]']
      2 print('Conc factor variation', np.round(min(c_data[151:]), 3),'to', np.ro
und(max(c_data[151:]),3))
```

NameError: name 'sim_res' is not defined

```
In [8]: # Simulation of process with step-wise increase of perfusion rate until wash-out.
        # This means that re-circulation rate change at the same time as the perfusion r

        init(VS_start=150)                                # Process initial varie

        par(pump1_t1=12, pump2_t1=12)                      # Pump schedule - recyc
        par(pump1_F1=2.5*0.155, pump2_F1=2.5*0.155/eps)
        par(pump1_t2=22, pump2_t2=22)
        par(pump1_F2=2.5*0.35, pump2_F2=2.5*0.35/eps)
        par(pump1_t3=32, pump2_t3=32)
        par(pump1_F3=2.5*0.63, pump2_F3=2.5*0.63/eps)
        par(pump1_t4=42, pump2_t4=42)
        par(pump1_F4=2.5*0.83, pump2_F4=2.5*0.83/eps)

        newplot(title='Perfusion cultivation - step wise increase of perfusion rate and
simu(60)
```

```

-----
FMIException                                         Traceback (most recent call last)
Cell In[8], line 16
    13 par(pump1_F4=2.5*0.83, pump2_F4=2.5*0.83/eps)
    15 newplot(title='Perfusion cultivation - step wise increase of perfusion ra
te and max productivity 4x chemostat')
--> 16 simu(60)

File \\VBoxSvr\Modelica\GitHub\Colab\BPL_TEST2_Perfusion\BPL_TEST2_Perfusion_fmpy
_explore.py:567, in simu(simulationTime, mode, options, diagrams)
    564 start_values = {parLocation[k]:parDict[k] for k in parDict.keys()}
    566 # Simulate
--> 567 sim_res = simulate_fmu(
    568     filename = fmu_model,
    569     validate = False,
    570     start_time = 0,
    571     stop_time = simulationTime,
    572     output_interval = simulationTime/options['ncp'],
    573     record_events = True,
    574     start_values = start_values,
    575     fmi_call_logger = None,
    576     output = list(set(extract_variables(diagrams) + list(stateDict.keys
    577 )) + key_variables))
    579 simulationDone = True
    581 elif mode in ['Continued', 'continued', 'cont']:

File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\simulation.py:761, in simulate
_fmu(filename, validate, start_time, stop_time, solver, step_size, relative_toler
ance, output_interval, record_events, fmi_type, start_values, apply_default_start
_values, input, output, timeout, debug_logging, visible, logger, fmi_call_logger,
step_finished, model_description, fmu_instance, set_input_derivatives, remote_pla
tform, early_return_allowed, use_event_mode, initialize, terminate, fmu_state, se
t_stop_time)
    759 result = simulateME(model_description, fmu, start_time, stop_time, so
lver, step_size, relative_tolerance, start_values, apply_default_start_values, in
put, output, output_interval, record_events, timeout, step_finished, validate, se
t_stop_time)
    760 elif fmi_type == 'CoSimulation':
--> 761     result = simulateCS(model_description, fmu, start_time, stop_time, re
lative_tolerance, start_values, apply_default_start_values, input, output, output
_interval, timeout, step_finished, set_input_derivatives, use_event_mode, early_r
eturn_allowed, validate, initialize, terminate, set_stop_time)
    763 if fmu_instance is None:
    764     fmu.freeInstance()

File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\simulation.py:1272, in simulat
eCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_valu
es, apply_default_start_values, input_signals, output, output_interval, timeout,
step_finished, set_input_derivatives, use_event_mode, early_return_allowed, valid
ate, initialize, terminate, set_stop_time)
    1270         break
    1271     else:
-> 1272         raise e
    1273 else:
    1275     t_input_event = input.nextEvent(time)

File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\simulation.py:1258, in simulat
eCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_valu
es, apply_default_start_values, input_signals, output, output_interval, timeout,

```

```
step_finished, set_input_derivatives, use_event_mode, early_return_allowed, validate, initialize, terminate, set_stop_time)
```

```
1256 try:
1257     if time + output_interval <= stop_time:
-> 1258         fmu.doStep(currentCommunicationPoint=time, communicationStepSize=output_interval)
1259         n_steps += 1
1260         time = n_steps * output_interval
```

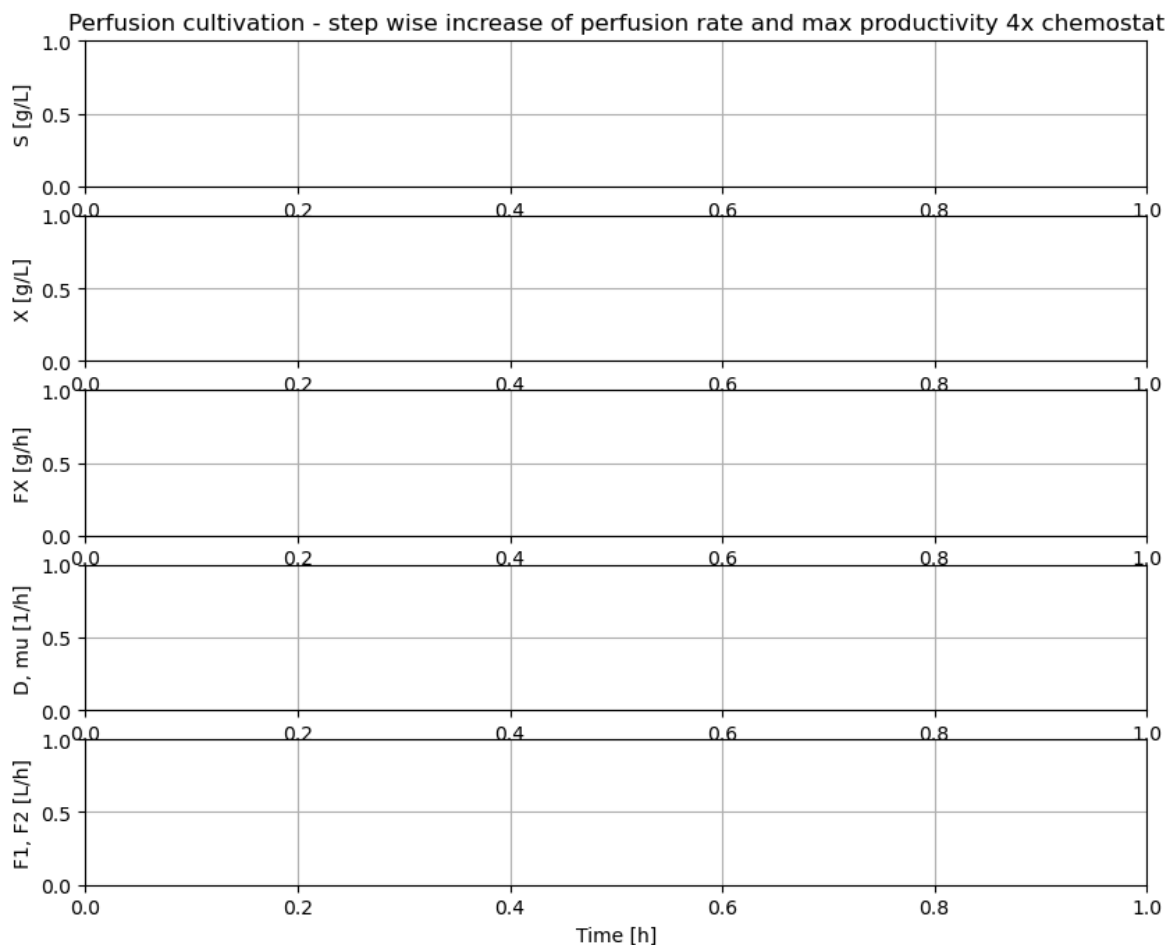
File ~\miniconda3\envs\fmipy\lib\site-packages\fmipy\fmipy2.py:580, in FMU2Slave.doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint)

```
579 def doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint=fmi2True):
--> 580     self.fmi2DoStep(self.component, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint)
```

File ~\miniconda3\envs\fmipy\lib\site-packages\fmipy\fmipy2.py:215, in _FMU2._fmi2Function.<locals>.w(*args)

```
212 if restype == fmi2Status: # status code
213     # check the status code
214     if res > fmi2Warning:
--> 215         raise FMICallException(function=fname, status=res)
217 return res
```

FMICallException: fmi2DoStep failed with status 3 (error).



```
In [9]: # Simulation without a plot and just to check typical values at high production
simu(40)
c_data=sim_res['filter.retentate.c[1]'][304:]/sim_res['filter.inlet.c[1]'][304:]
print('Conc factor variation', np.round(min(c_data[304:]), 3), 'to', np.round(max(c_data[304:]), 3))
```

FMIException

Traceback (most recent call last)

Cell **In[9]**, line 2

```
1 # Simulation without a plot and just to check typical values at high prod
  uction rate
----> 2 simu(40)
      3 c_data=sim_res['filter.retentate.c[1]'][304:]/sim_res['filter.inlet.c
[1]'][304:]
      4 print('Conc factor variation', np.round(min(c_data[304:]), 3), 'to', np.r
ound(max(c_data[304:]),3))
```

File \\VBoxSvr\\Modelica\\GitHub\\Colab\\BPL_TEST2_Perfusion\\BPL_TEST2_Perfusion_fmpy_explore.py:567, in **simu**(simulationTime, mode, options, diagrams)

```
564     start_values = {parLocation[k]:parDict[k] for k in parDict.keys()}
566     # Simulate
--> 567     sim_res = simulate_fmu(
568         filename = fmu_model,
569         validate = False,
570         start_time = 0,
571         stop_time = simulationTime,
572         output_interval = simulationTime/options['ncp'],
573         record_events = True,
574         start_values = start_values,
575         fmi_call_logger = None,
576         output = list(set(extract_variables(diagrams) + list(stateDict.keys
577             ())) + key_variables))
579     simulationDone = True
581 elif mode in ['Continued', 'continued', 'cont']:
```

File ~\\miniconda3\\envs\\fmpy\\lib\\site-packages\\fmpy\\simulation.py:761, in **simulate_fmu**(filename, validate, start_time, stop_time, solver, step_size, relative_tolerance, output_interval, record_events, fmi_type, start_values, apply_default_start_values, input, output, timeout, debug_logging, visible, logger, fmi_call_logger, step_finished, model_description, fmu_instance, set_input_derivatives, remote_platform, early_return_allowed, use_event_mode, initialize, terminate, fmu_state, set_stop_time)

```
759     result = simulateME(model_description, fmu, start_time, stop_time, so
lver, step_size, relative_tolerance, start_values, apply_default_start_values, in
put, output, output_interval, record_events, timeout, step_finished, validate, se
t_stop_time)
760 elif fmi_type == 'CoSimulation':
--> 761     result = simulateCS(model_description, fmu, start_time, stop_time, re
lative_tolerance, start_values, apply_default_start_values, input, output, output
_interval, timeout, step_finished, set_input_derivatives, use_event_mode, early_r
eturn_allowed, validate, initialize, terminate, set_stop_time)
763 if fmu_instance is None:
764     fmu.freeInstance()
```

File ~\\miniconda3\\envs\\fmpy\\lib\\site-packages\\fmpy\\simulation.py:1272, in **simulateCS**(model_description, fmu, start_time, stop_time, relative_tolerance, start_values, apply_default_start_values, input_signals, output, output_interval, timeout, step_finished, set_input_derivatives, use_event_mode, early_return_allowed, validate, initialize, terminate, set_stop_time)

```
1270         break
1271     else:
-> 1272         raise e
1273 else:
1275     t_input_event = input.nextEvent(time)
```



```
File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\simulation.py:1258, in simulateCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_values, apply_default_start_values, input_signals, output, output_interval, timeout, step_finished, set_input_derivatives, use_event_mode, early_return_allowed, validate, initialize, terminate, set_stop_time)
```

```
1256 try:
1257     if time + output_interval <= stop_time:
-> 1258         fmu.doStep(currentCommunicationPoint=time, communicationStepSize=output_interval)
1259         n_steps += 1
1260         time = n_steps * output_interval
```

```
File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\fmi2.py:580, in FMU2Slave.doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint)
```

```
579 def doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint=fmi2True):
--> 580     self.fmi2DoStep(self.component, currentCommunicationPoint, communicationStepSize, noSetFMUStatePriorToCurrentPoint)
```

```
File ~\miniconda3\envs\fmpy\lib\site-packages\fmpy\fmi2.py:215, in _FMU2._fmi2Function.<locals>.w(*args)
```

```
212 if restype == fmi2Status: # status code
213     # check the status code
214     if res > fmi2Warning:
--> 215         raise FMICallException(function=fname, status=res)
217 return res
```

FMICallException: fmi2DoStep failed with status 3 (error).

```
In [ ]: #describe('cstrProdMax')
```

```
In [11]: # The maximal biomass productivity before washout is obtained around 40 hours
np.round(model_get('harvesttank.inlet.F')*model_get('harvesttank.inlet.c[1]'),1)
```

Error: Information available after first simulation

Error: Information available after first simulation

TypeError Traceback (most recent call last)

Cell In[11], line 2

```
1 # The maximal biomass productivity before washout is obtained around 40 hours
----> 2 np.round(model_get('harvesttank.inlet.F')*model_get('harvesttank.inlet.c[1]'),1)
```

TypeError: unsupported operand type(s) for *: 'NoneType' and 'NoneType'

```
In [12]: # Thus perfusion (with this filter) brings a productivity improvement of about
np.round(23.5/5.6,1)
```

Out[12]: 4.2

```
In [13]: # Finally we check the filter flow rates at time 40 hour - note the negative sig
model_get('filter.inlet.F')
```

Error: Information available after first simulation

```
In [14]: model_get('filter.filtrate.F')
```

Error: Information available after first simulation

```
In [15]: model_get('filter.retentate.F')
```

Error: Information available after first simution

Summary

- The perfusion filter had a concentration factor of cells around 1.08 and re-cycling flow was set to a factor 10 higher than the perfusion rate and changed when perfusion rate was change to keep the ratio factor 10.
- The first simulation showed that by cell retention using perfusion filter the process could be run at a perfusion flow rate at the maximal flow rate possible for corresponding chemostat culture and cell concetration increased steadily.
- The second simulation showed that with a proper startup cell concentration, the cell concentration remained constant when perfusion rate increased in a similar way as what we see in a chemostat.
- The second simulation also showed that biomass productivity in this case was increased by a factor 4.2 compared to chemostat.
- If the perfusion rate increased to higher levels washout started but the decrease of cell concentration was slow.

Some of you who read this may have your perfusion experience with CHO-cultures. For such cultures the cell concentration do increase with increase of perfusion rate and there are understood reasons for that. But for this simplified process as well as microbial processes they typically keep cell concentration constant when flow rate is chaged, and that under quite wide conditions. I will try come back to this phenomena in a later notebook.

Appendix

```
In [ ]: disp('culture')
```

```
In [ ]: describe('mu')
```

```
In [ ]: # List of components in the process setup and also a couple of other things like  
describe('parts')
```

```
In [ ]: describe('MSL')
```

```
In [10]: system_info()
```

System information

- OS: Windows
- Python: 3.10.13
- Scipy: not installed in the notebook
- FMPy: 0.3.19
- FMU by: JModelica.org
- FMI: 2.0
- Type: CS
- Name: BPL_TEST2.Perfusion
- Generated: 2024-02-29T19:58:20
- MSL: 3.2.2 build 3
- Description: Bioprocess Library version 2.1.2 prel
- Interaction: FMU-explore for FMPy version 0.9.9

In []: