

BPL_TEST2_Perfusion - demo

This notebook explores perfusion cultivation in comparison with ordinary continuous cultivation (chemostat) and uses comparable settings to earlier notebooks. 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 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 the process 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

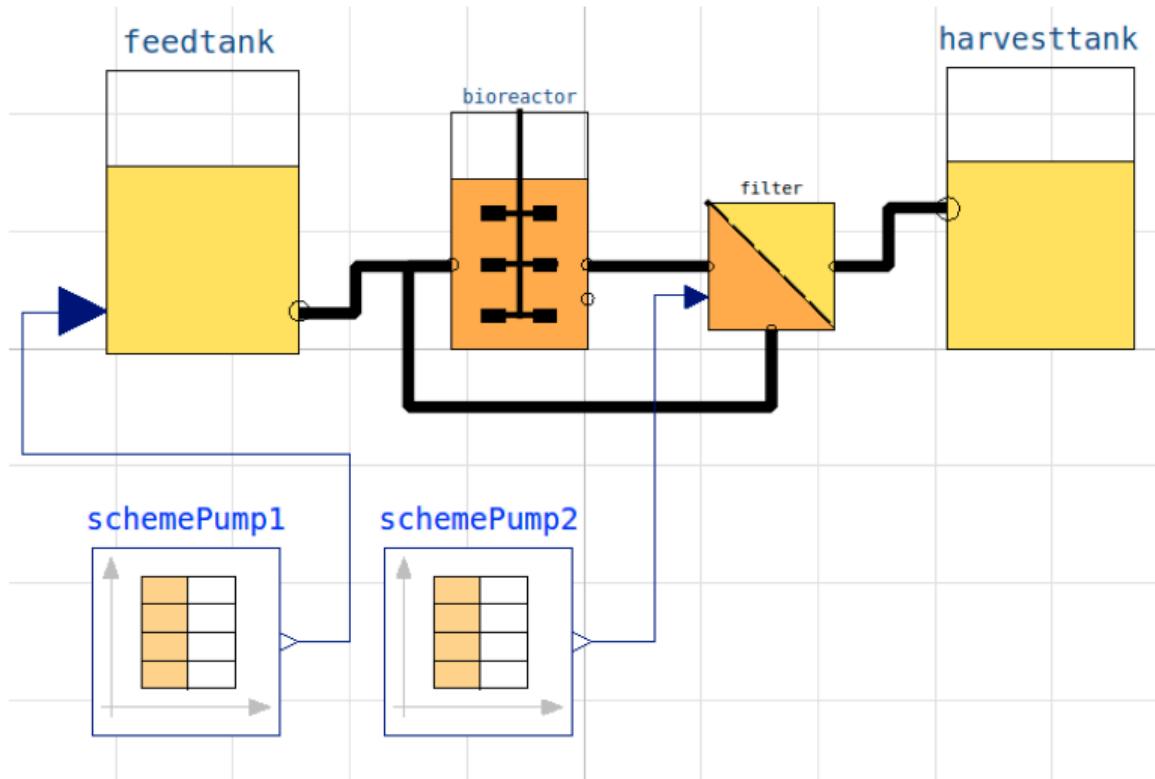
Note that both `disp()` and `describe()` takes values from the last simulation and the command `process_diagram()` brings up the main configuration

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.

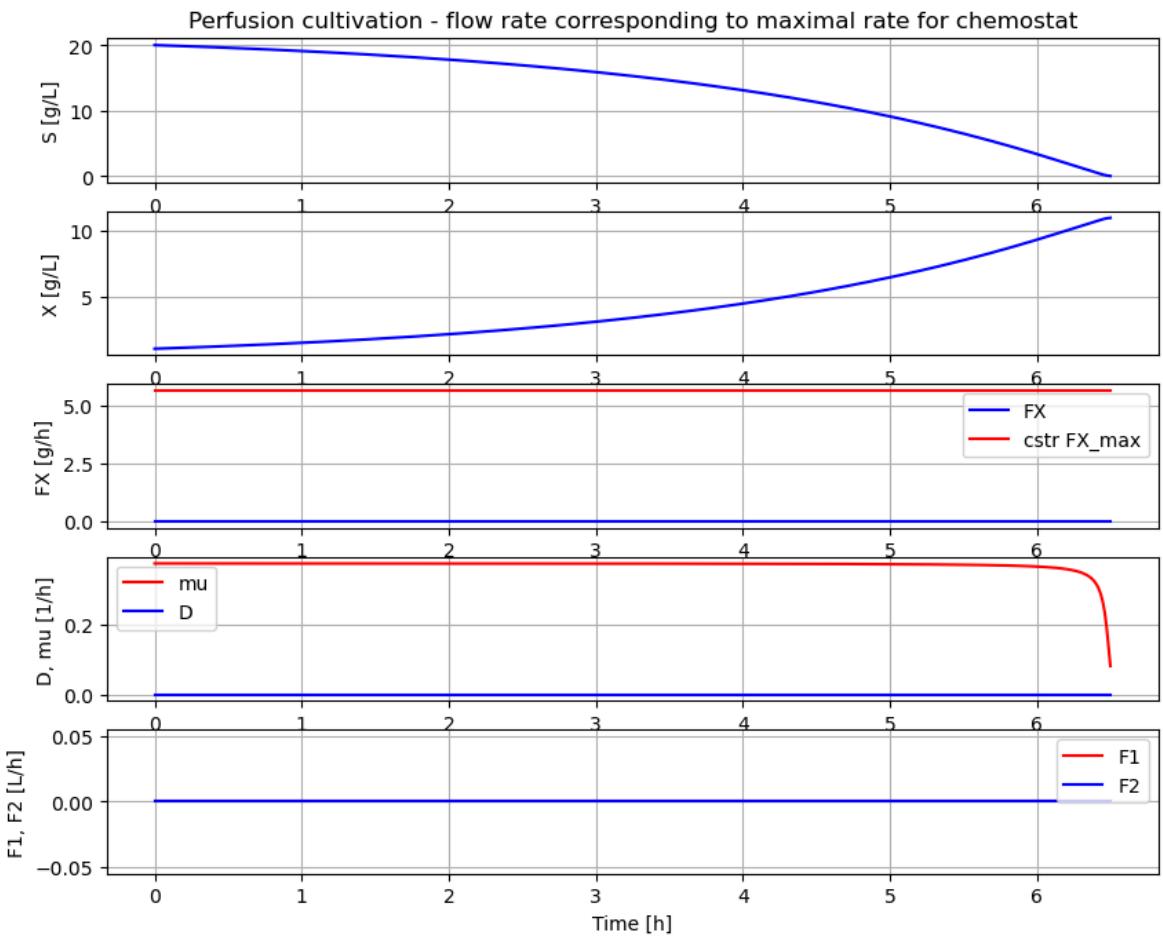


```
In [5]: # Process parameters used throughout
par(Y=0.5, qSmax=0.75, Ks=0.1)                                     # Culture
par(filter_eps=0.10, filter_alpha_X=0.02, filter_alpha_S=0.10)       # Filter
par(S_in=30.0)                                                       # Inlet subs
init(V_start=1.0, VX_start=1.0)                                         # Process in
eps = parValue['filter_eps']                                              # Pump sche
```

```
In [6]: # 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 - recycl
par(pump1_F1=2.5*0.155, pump2_F1=2.5*0.155/eps)                 # Pump schedule - recycl
par(pump1_t2=940, pump2_t2=940, pump1_t3=950, pump2_t3=950, pump1_t4=960, pump2_t4=960) # Pump schedule - recycl

newplot(title='Perfusion cultivation - flow rate corresponding to maximal rate f
simu(6.5)
```



```
In [7]: # 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))
```

Conc factor of perfusion filter = 1.186

```
In [8]: 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
```

Conc factor variation 1.186 to 1.186

```
In [9]: # 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)
```

```

-----  

FMICallException                                     Traceback (most recent call last)  

Cell In[9], 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 \\VBoxSrv\Modelica\GitHub\Colab\BPL_TEST2_Perfusion\BPL_TEST2_Perfusion_fmpy  

_explor.py:579, in simu(simulationTime, mode, options, diagrams, timeDiscreteSta  

tes, stateValue, stateValueInitial, stateValueInitialLoc, parValue, parLocation,  

fmu_model)  

    576     start_values = {parLocation[k]:parValue[k] for k in parValue.keys()}  

    578     # Simulate  

--> 579     sim_res = simulate_fmu(  

    580         filename = fmu_model,  

    581         validate = False,  

    582         start_time = 0,  

    583         stop_time = simulationTime,  

    584         output_interval = simulationTime/options[      ],  

    585         record_events = True,  

    586         start_values = start_values,  

    587         fmi_call_logger = None,  

    588         output = list(set(extract_variables(diagrams)) + list(stateValue.key  

s()) + key_variables))  

    589     )  

    591     simulationDone = True  

    593 elif mode in ['Continued', 'continued', 'cont']:  

File ~\miniconda3\envs\fmpy0325\Lib\site-packages\fmpy\simulation.py:789, in simu  

late_fmu(filename, validate, start_time, stop_time, solver, step_size, relative_t  

olerance, output_interval, record_events, fmi_type, start_values, apply_default_s  

tart_values, input, output, timeout, debug_logging, visible, logger, fmi_call_log  

ger, step_finished, model_description, fmu_instance, set_input_derivatives, remot  

e_platform, early_return_allowed, use_event_mode, initialize, terminate, fmu_stat  

e, set_stop_time)  

    787     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)  

    788 elif fmi_type == 'CoSimulation':  

--> 789     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)  

    791 if fmu_instance is None:  

    792     fmu.freeInstance()  

File ~\miniconda3\envs\fmpy0325\Lib\site-packages\fmpy\simulation.py:1322, in sim  

ulateCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_  

values, apply_default_start_values, input_signals, output, output_interval, timeo  

ut, step_finished, set_input_derivatives, use_event_mode, early_return_allowed, v  

alidate, initialize, terminate, set_stop_time)  

    1320             break  

    1321         else:  

-> 1322             raise exception  

    1324     recorder.sample(time)  

    1326 else:  

File ~\miniconda3\envs\fmpy0325\Lib\site-packages\fmpy\simulation.py:1307, in sim

```

```

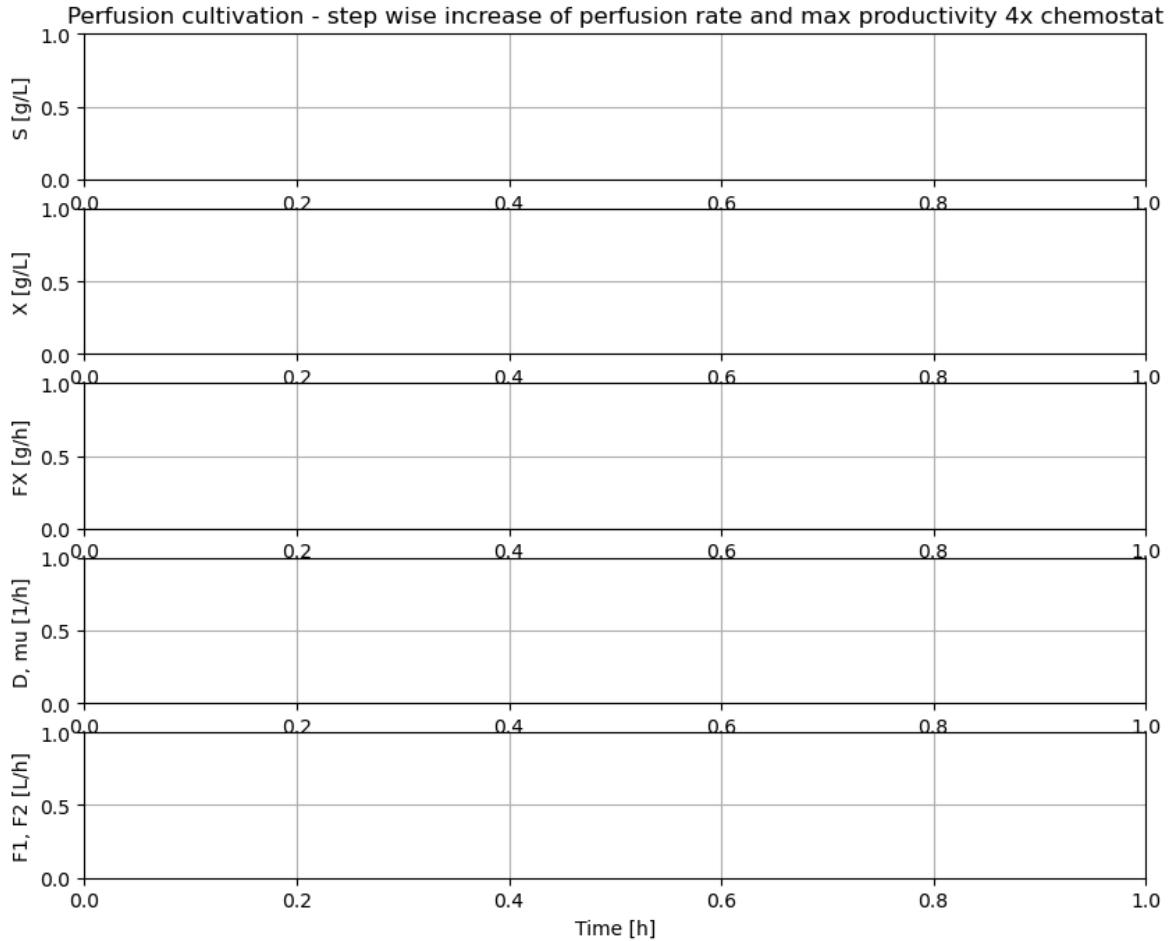
ulateCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_
values, apply_default_start_values, input_signals, output, output_interval, timeo
ut, step_finished, set_input_derivatives, use_event_mode, early_return_allowed, v
alidate, initialize, terminate, set_stop_time)
    1304 input.apply(time, continuous=True, discrete=True, after_event=True)
    1306 try:
-> 1307     fmu.doStep(currentCommunicationPoint=time, communicationStepSize=step_
_size)
    1309     time = next_communication_point
1311 except FMICallException as exception:

File ~\miniconda3\envs\fmpy0325\Lib\site-packages\fmpy\fmi2.py:580, in FMU2Slave.
doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStatePrior
ToCurrentPoint)
    579 def doStep(self, currentCommunicationPoint, communicationStepSize, noSetF
MUStatePriorToCurrentPoint=fmi2True):
--> 580     self.fmi2DoStep(self.component, currentCommunicationPoint, communicat
ionStepSize, noSetFMUStatePriorToCurrentPoint)

File ~\miniconda3\envs\fmpy0325\Lib\site-packages\fmpy\fmi2.py:215, in _FMU2._fmi
2Function.<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

FМИCallException: fmi2DoStep failed with status 3 (error).

```



```
In [ ]: # 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))

In [ ]: #describe('cstrProdMax')

In [ ]: # 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)

In [10]: # Thus perfusion (with this filter) brings a productivity improvement of about
np.round(23.5/5.6,1)

Out[10]: np.float64(4.2)

In [11]: # Finally we check the filter flow rates at time 40 hour - note the negative sign
model_get('filter.inlet.F')

Out[11]: np.float64(0.0)

In [12]: model_get('filter.filtrate.F')

Out[12]: np.float64(-0.0)

In [13]: model_get('filter.retentate.F')

Out[13]: np.float64(-0.0)

```

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 deacrese 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 [14]: disp('culture')
```

```
Y : 0.5
qSmax : 0.75
Ks : 0.1
```

```
In [15]: describe('mu')
```

```
Cell specific growth rate variable : 0.081 [ 1/h ]
```

```
In [16]: # List of components in the process setup and also a couple of other things like
         describe('parts')
```

```
['bioreactor', 'bioreactor.culture', 'D', 'feedtank', 'filter', 'harvesttank', 'M
SL', 'schemePump1', 'schemePump2']
```

```
In [17]: describe('MSL')
```

```
MSL: RealInput, RealOutput, CombiTimeTable, Types
```

```
In [18]: system_info()
```

```
System information
-OS: Windows
-Python: 3.12.11
-Scipy: not installed in the notebook
-FMPy: 0.3.25
-FMU by: JModelica.org
-FMI: 2.0
-Type: CS
-Name: BPL.Examples_TEST2.Perfusion
-Generated: 2025-07-26T09:41:00
-MSL: 3.2.2 build 3
-Description: Bioprocess Library version 2.3.1
-Interaction: FMU-explore for FMPy version 1.0.2
```

```
In [ ]:
```

```
In [ ]:
```