BPL_TEST2_Perfusion - demo

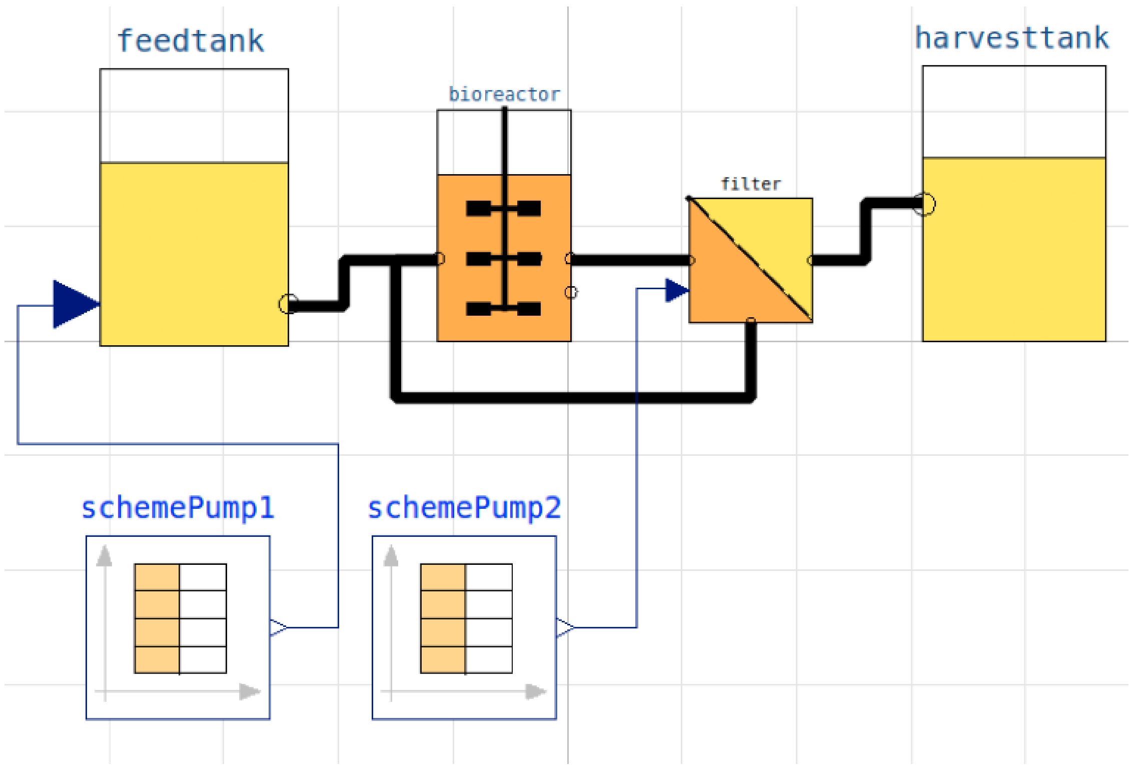
This notebook explore perfusion cultivation in comparison with ordinary continuous cultivation (chemostat) and use comparable settings to earlier notebook. Further you see here examples of interaction with the simplified commands par(), init(), simu() etc as well as direct interaction with the FMU which 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/uni
ts

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]
```

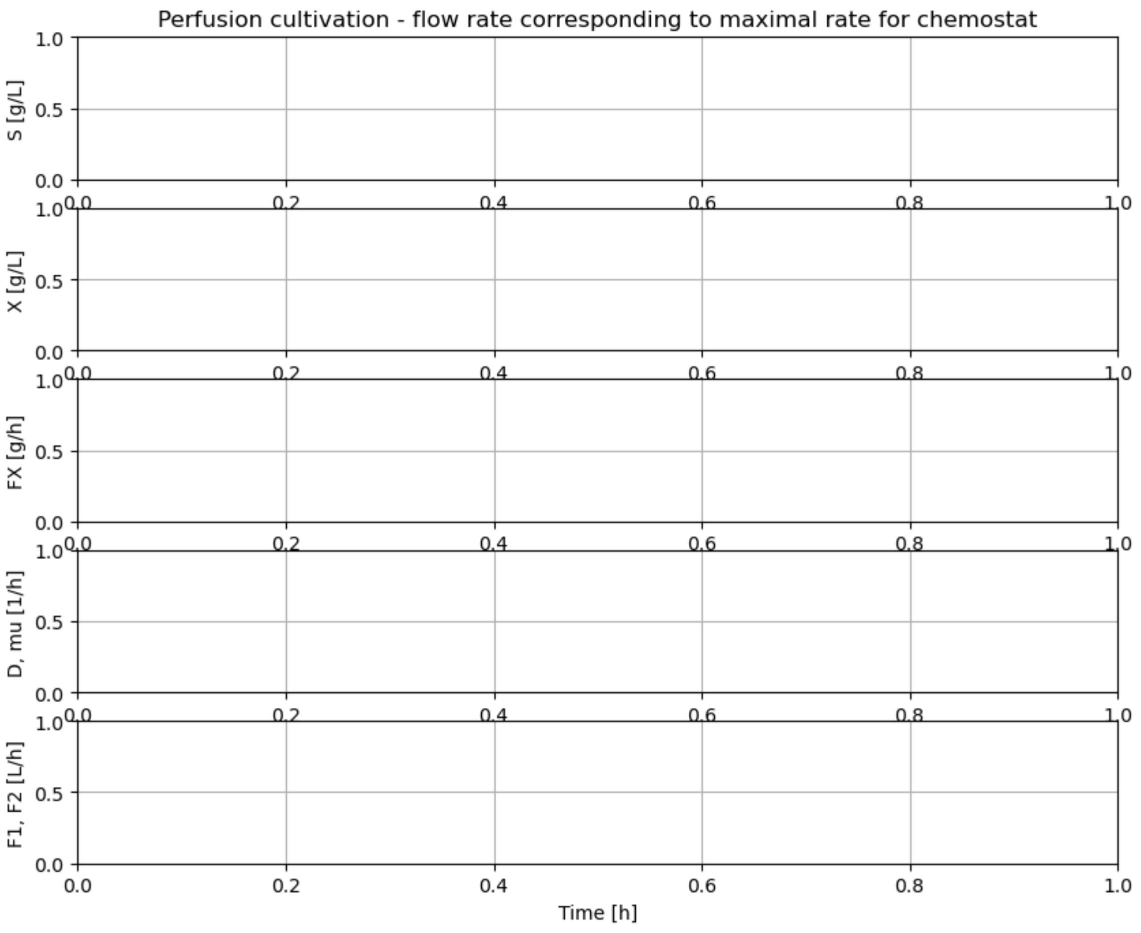
In [3]:
```
process_diagram()
```



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_0=1.0, VX_0=1.0)                                                # Process ir
eps = parDict['filter_eps']                                            # Pump schec
```

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

init(VS_0=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)
```

```
---------------------------------------------------------------------------
FMICallException                          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=96
0, pump2_t4=960)
      8 newplot(title='Perfusion cultivation - flow rate corresponding to maxim
al rate for chemostat')
----> 9 simu(10)

File \\VBoxSvr\Modelica\GitHub\Colab\BPL_TEST2_Perfusion\BPL_TEST2_Perfusion_fm
py_explore.py:566, in simu(simulationTime, mode, options, diagrams)
    563    start_values = {parLocation[k]:parDict[k] for k in parDict.keys()}
    565    # Simulate
--> 566    sim_res = simulate_fmu(
    567        filename = fmu_model,
    568        validate = False,
    569        start_time = 0,
    570        stop_time = simulationTime,
    571        output_interval = simulationTime/options['ncp'],
    572        record_events = True,
    573        start_values = start_values,
    574        fmi_call_logger = None,
    575        output = list(set(extract_variables(diagrams) + list(stateDict.ke
ys()) + key_variables))
    576    )
    578    simulationDone = True
    580 elif mode in ['Continued', 'continued', 'cont']:

File ~\miniconda3\envs\fmpylab\lib\site-packages\fmpy\simulation.py:758, in sim
ulate_fmu(filename, validate, start_time, stop_time, solver, step_size, relativ
e_tolerance, output_interval, record_events, fmi_type, start_values, apply_defa
ult_start_values, input, output, timeout, debug_logging, visible, logger, fmi_c
all_logger, step_finished, model_description, fmu_instance, set_input_derivativ
es, remote_platform, early_return_allowed, use_event_mode, initialize, terminat
e, fmu_state)
    756     result = simulateME(model_description, fmu, start_time, stop_time,
solver, step_size, relative_tolerance, start_values, apply_default_start_value
s, input, output, output_interval, record_events, timeout, step_finished, valid
ate)
    757 elif fmi_type == 'CoSimulation':
--> 758     result = simulateCS(model_description, fmu, start_time, stop_time,
relative_tolerance, start_values, apply_default_start_values, input, output, ou
tput_interval, timeout, step_finished, set_input_derivatives, use_event_mode, e
arly_return_allowed, validate, initialize, terminate)
    760 if fmu_instance is None:
    761     fmu.freeInstance()

File ~\miniconda3\envs\fmpylab\lib\site-packages\fmpy\simulation.py:1270, in si
mulateCS(model_description, fmu, start_time, stop_time, relative_tolerance, sta
rt_values, apply_default_start_values, input_signals, output, output_interval,
timeout, step_finished, set_input_derivatives, use_event_mode, early_return_all
owed, validate, initialize, terminate)
   1268                 break
   1269         else:
-> 1270             raise e
   1271 else:
   1273     t_input_event = input.nextEvent(time)

File ~\miniconda3\envs\fmpylab\lib\site-packages\fmpy\simulation.py:1256, in si
mulateCS(model_description, fmu, start_time, stop_time, relative_tolerance, sta
rt_values, apply_default_start_values, input_signals, output, output_interval,
timeout, step_finished, set_input_derivatives, use_event_mode, early_return_all
owed, validate, initialize, terminate)
   1254 try:
   1255     if time + output_interval <= stop_time:
-> 1256         fmu.doStep(currentCommunicationPoint=time, communicationStepSiz
e=output_interval)
   1257         n_steps += 1
   1258         time = n_steps * output_interval

File ~\miniconda3\envs\fmpylab\lib\site-packages\fmpy\fmi2.py:580, in FMU2Slav
e.doStep(self, currentCommunicationPoint, communicationStepSize, noSetFMUStateP
riorToCurrentPoint)
    579 def doStep(self, currentCommunicationPoint, communicationStepSize, noSe
tFMUStatePriorToCurrentPoint=fmi2True):
--> 580     self.fmi2DoStep(self.component, currentCommunicationPoint, communic
ationStepSize, noSetFMUStatePriorToCurrentPoint)

File ~\miniconda3\envs\fmpylab\lib\site-packages\fmpy\fmi2.py:215, in _FMU2._fm
i2Function.<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 [ ]:
```
# 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))
```

In [ ]:
```
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
```

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

init(VS_0=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)
```

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(ma
```

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

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

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

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

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

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

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 decrase 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 simplifed 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 [6]:
```
disp('culture')
```

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

In [7]:
```
describe('mu')
```

```
Error: Information available after first simution
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [7], line 1
----> 1 describe('mu')

File \\VBoxSvr\Modelica\GitHub\Colab\BPL_TEST2_Perfusion\BPL_TEST2_Perfusion_fm
py_explore.py:400, in describe(name, decimals)
    397    print(cstrProdMax.__doc__,':',cstrProdMax(), '[ g/h ]')
    399 else:
--> 400    describe_general(name, decimals)

File \\VBoxSvr\Modelica\GitHub\Colab\BPL_TEST2_Perfusion\BPL_TEST2_Perfusion_fm
py_explore.py:691, in describe_general(name, decimals)
    689          print(description, ':', value)
    690    else:
--> 691       print(description, ':', np.round(value, decimals), '[',unit,']')
    693 else:
    694    description = model_get_variable_description(name)

File <__array_function__ internals>:200, in round_(*args, **kwargs)

File ~\miniconda3\envs\fmpylab\lib\site-packages\numpy\core\fromnumeric.py:376
3, in round_(a, decimals, out)
   3754 @array_function_dispatch(_around_dispatcher)
   3755 def round_(a, decimals=0, out=None):
   3756     """
   3757     Round an array to the given number of decimals.
   3758 
   (...)
   3761     around : equivalent function; see for details.
   3762     """
-> 3763     return around(a, decimals=decimals, out=out)

File <__array_function__ internals>:200, in around(*args, **kwargs)

File ~\miniconda3\envs\fmpylab\lib\site-packages\numpy\core\fromnumeric.py:333
7, in around(a, decimals, out)
   3245 @array_function_dispatch(_around_dispatcher)
   3246 def around(a, decimals=0, out=None):
   3247     """
   3248     Evenly round to the given number of decimals.
   3249 
   (...)
   3335 
   3336     """
-> 3337     return _wrapfunc(a, 'round', decimals=decimals, out=out)

File ~\miniconda3\envs\fmpylab\lib\site-packages\numpy\core\fromnumeric.py:54,
in _wrapfunc(obj, method, *args, **kwds)
     52 bound = getattr(obj, method, None)
     53 if bound is None:
---> 54     return _wrapit(obj, method, *args, **kwds)
     56 try:
     57     return bound(*args, **kwds)

File ~\miniconda3\envs\fmpylab\lib\site-packages\numpy\core\fromnumeric.py:43,
in _wrapit(obj, method, *args, **kwds)
     41 except AttributeError:
     42     wrap = None
---> 43 result = getattr(asarray(obj), method)(*args, **kwds)
```

```
     44 if wrap:
     45     if not isinstance(result, mu.ndarray):

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

In [8]:
```
# List of components in the process setup and also a couple of other things like
describe('parts')
```

```
['bioreactor', 'bioreactor.culture', 'D', 'feedtank', 'filter', 'harvesttank',
'liquidphase', 'MSL', 'schemePump1', 'schemePump2']
```

In [9]:
```
describe('MSL')
```

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

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

```
System information
 -OS: Windows
 -Python: 3.9.16
 -Scipy: not installed in the notebook
 -FMPy: 0.3.15
 -FMU by: JModelica.org
 -FMI: 2.0
 -Type: CS
 -Name: BPL_TEST2.Perfusion
 -Generated: 2023-09-02T07:46:00
 -MSL: 3.2.2 build 3
 -Description: Bioprocess Library version 2.1.2 prel
 -Interaction: FMU-explore for FMPy version 0.9.8
```

In [ ]: