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.

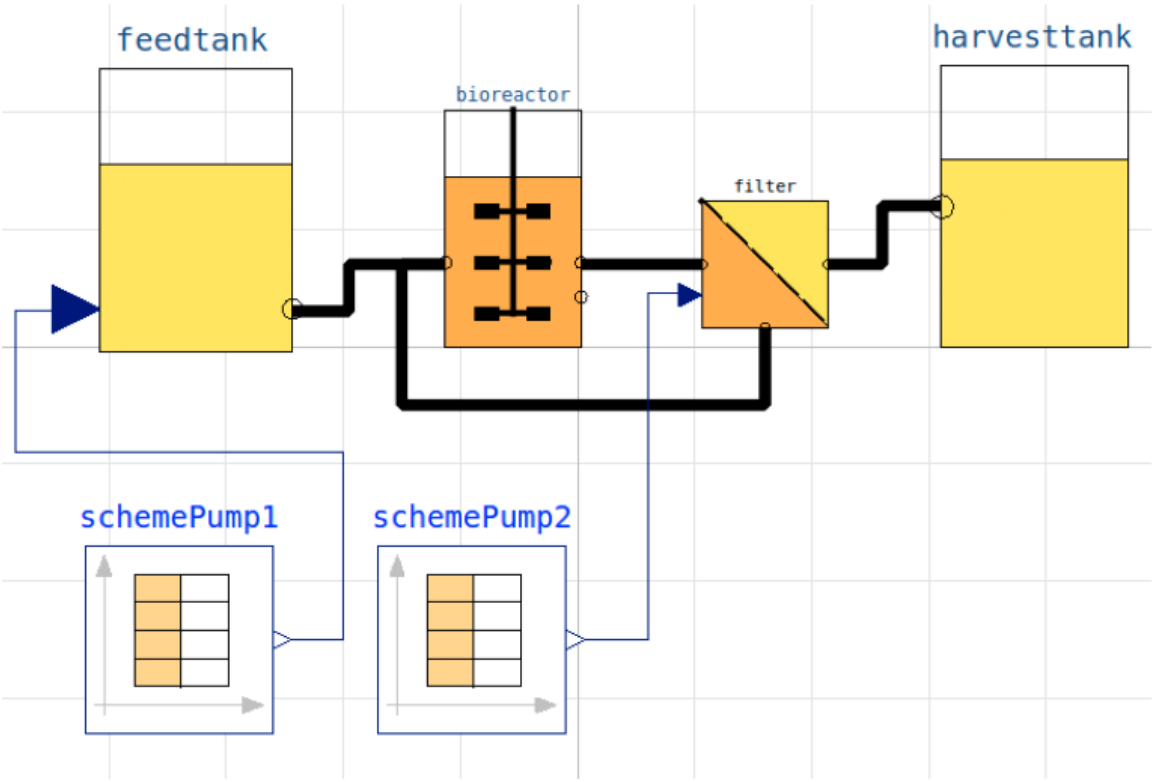
In [1]:
```
run -i BPL_TEST2_Perfusion_explore.py
```

```
Linux - run FMU pre-compiled OpenModelica

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 simul
ation
 - describe()  - describe culture, broth, parameters, variables with value
s/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()
```

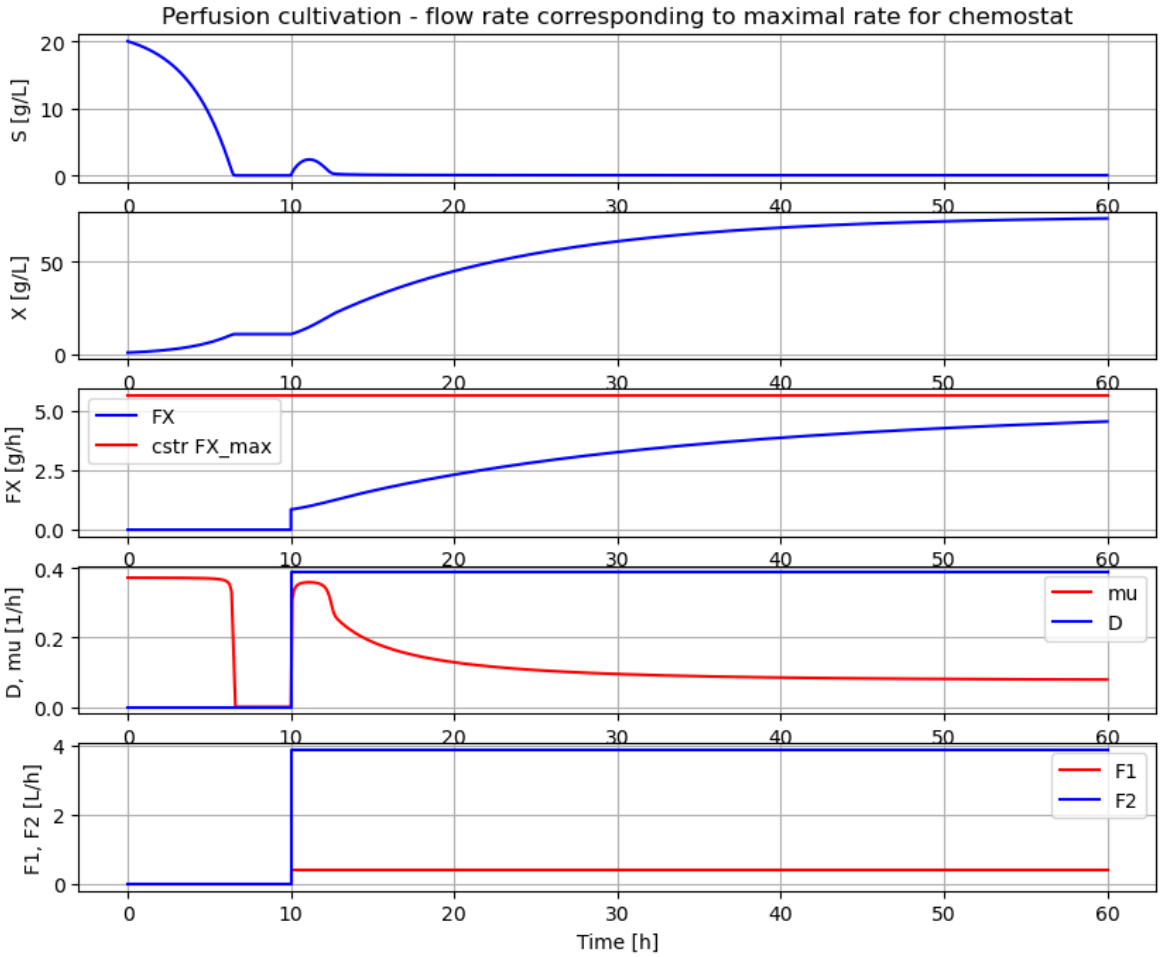
```
No processDiagram.png file in the FMU, but try the file on disk.
```



In [4]:
```
# Process parameters used throughout
par(Y=0.5, qSmax=0.75, Ks=0.1)                                  # Cul
par(filter_eps=0.10, filter_alpha_X=0.02, filter_alpha_S=0.10)  # Fil
par(S_in=30.0)                                                  # Inl
init(V_start=1.0, VX_start=1.0)                                 # Pro
eps = parDict['filter_eps']                                     # Pum
```

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

init(VS_start=20)                                       # Process initia
par(pump1_t1=10, pump2_t1=10)                           # Pump schedule
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,

newplot(title='Perfusion cultivation - flow rate corresponding to maximal
simu(60)
```

```
Could not find cannot import name 'dopri5' from 'assimulo.lib' (/home/janp
eter/miniconda3/envs/pyfmi/lib/python3.12/site-packages/assimulo/lib/__ini
t__.py)
Could not find cannot import name 'rodas' from 'assimulo.lib' (/home/janpe
ter/miniconda3/envs/pyfmi/lib/python3.12/site-packages/assimulo/lib/__init
__.py)
Could not find cannot import name 'odassl' from 'assimulo.lib' (/home/janp
eter/miniconda3/envs/pyfmi/lib/python3.12/site-packages/assimulo/lib/__ini
t__.py)
Could not find ODEPACK functions.
Could not find RADAR5
Could not find GLIMDA.
```



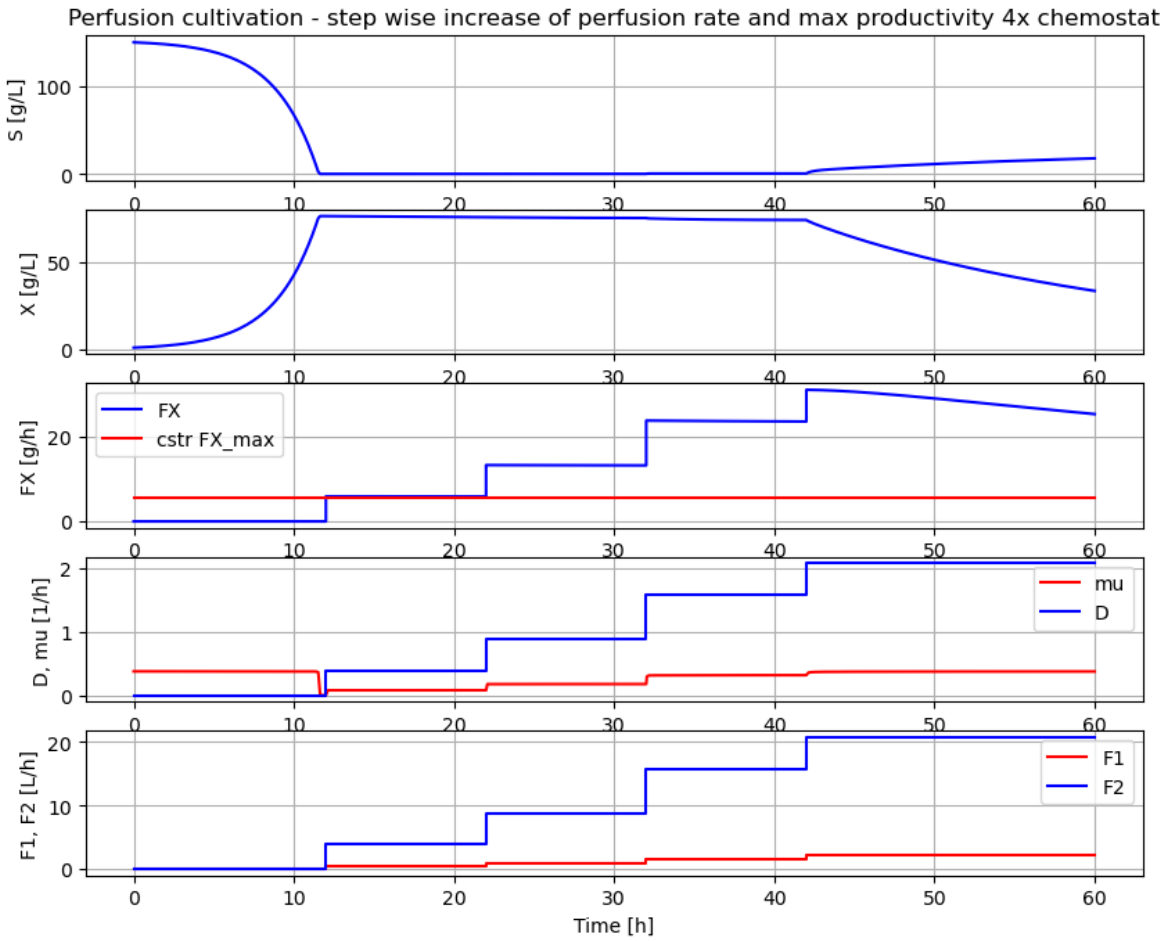
In [6]:
```
# Concentration factor of the filter
c=model.get('filter.retentate.c[1]')[0]/model.get('filter.inlet.c[1]')[0]
print('Conc factor of perfusion filter =', np.round(c,3))
```

```
Conc factor of perfusion filter = 1.179
```

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.ro
```

```
Conc factor variation 1.179 to 1.179
```

In [8]:
```
# Simulation of process with step-wise increase of pefusion rate until wa
# This means that re-circulation rate change at the same time as the perf

init(VS_start=150)                                          # Process initia

par(pump1_t1=12, pump2_t1=12)                               # Pump schedule
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 ra
simu(60)
```



In [9]:
```
# Simulation without a plot and just to check typical values at high prod
simu(38)
c_data=sim_res['filter.retentate.c[1]']/sim_res['filter.inlet.c[1]']
print('Conc factor variation', np.round(min(c_data[190:]), 3), 'to', np.r
```

```
Conc factor variation 1.162 to 1.179
```

In [10]:
```
describe('cstrProdMax')
```

```
Calculate from the model maximal chemostat productivity FX_max : 5.625 [
g/h ]
```

In [11]:
```
# The maximal biomass productivity before washout is obtained aroudn 40 h
np.round(model.get('harvesttank.inlet.F')[0]*model.get('harvesttank.inlet
```

Out[11]: np.float64(23.6)

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

Out[12]: np.float64(4.2)

In [13]:
```
# Finally we check the filter flow rates at time 40 hour - note the negat
model.get('filter.inlet.F')[0]
```

Out[13]: np.float64(15.749999999999998)

In [14]:
```
model.get('filter.filtrate.F')[0]
```

Out[14]: np.float64(-1.575)

In [15]:
```
model.get('filter.retentate.F')[0]
```

Out[15]:
```
np.float64(-14.174999999999999)
```

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 [16]:
```
disp('culture')
```

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

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

```
Cell specific growth rate variable : 0.314 [ 1/h ]
```

In [18]:
```
# List of components in the process setup and also a couple of other thin
describe('parts')
```

```
['bioreactor', 'bioreactor.culture', 'D', 'feedtank', 'filter', 'harvestta
nk', 'schemePump1', 'schemePump2']
```

In [19]:
```
describe('MSL')
```

```
MSL: 4.1.0 - used components: RealInput, RealOutput, CombiTimeTable, Types
```

In [20]:
```
system_info()
```

```
System information
 -OS: Linux
 -Python: 3.12.9
 -Scipy: not installed in the notebook
 -PyFMI: 2.18.0
 -FMU by: OpenModelica Compiler OpenModelica 1.26.0~dev-200-gcb3254b
 -FMI: 2.0
 -Type: FMUModelME2
 -Name: BPL.Examples_TEST2.Perfusion
 -Generated: 2025-07-28T07:59:36Z
 -MSL: 4.1.0
 -Description: Bioprocess Library version 2.3.1
 -Interaction: FMU-explore version 1.0.0
```

In [21]:
```
!lsb_release -a
```

```
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.3 LTS
Release:        24.04
Codename:       noble
```