BPL_TEST2_Perfusion script with FMPy

The key library FMPy is installed.

After the installation a small application BPL_TEST2_Perfusion is loaded and run. You can continue with this example if you like.

```
!lsb_release -a # Actual VM Ubuntu version used by Google
```

```
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
```

```
%env PYTHONPATH=
```

```
env: PYTHONPATH=
```

```
!wget https://repo.anaconda.com/miniconda/Miniconda3-py312_24.3.0-0-Linux-x86_64.sh
!chmod +x Miniconda3-py312_24.3.0-0-Linux-x86_64.sh
!bash ./Miniconda3-py312_24.3.0-0-Linux-x86_64.sh -b -f -p /usr/local
import sys
sys.path.append('/usr/local/lib/python3.12/site-packages/')
```

```
--2024-05-15 08:14:37--  https://repo.anaconda.com/miniconda/Miniconda3-py312_24.3.0-0-Linux-
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.191.158, 104.16.32.241, 2606:4700::
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.191.158|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 143351488 (137M) [application/octet-stream]
Saving to: ‘Miniconda3-py312_24.3.0-0-Linux-x86_64.sh’

Miniconda3-py312_24 100%[===================>] 136.71M  89.5MB/s    in 1.5s

2024-05-15 08:14:39 (89.5 MB/s) - ‘Miniconda3-py312_24.3.0-0-Linux-x86_64.sh’ saved [14335148

PREFIX=/usr/local
Unpacking payload ...

Installing base environment...

Preparing transaction: ...working... done
Executing transaction: ...working... done
installation finished.
```

```
!conda update -n base -c defaults conda --yes
```

```
Channels:
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - conda


The following packages will be downloaded:
```

```
    package                    |            build
    ---------------------------|-----------------
    conda-24.5.0               |  py312h06a4308_0         1.2 MB
    frozendict-2.4.2           |  py312h06a4308_0          36 KB
    openssl-3.0.13             |       h7f8727e_1         5.2 MB
    ------------------------------------------------------------
                                           Total:         6.5 MB

The following NEW packages will be INSTALLED:

  frozendict         pkgs/main/linux-64::frozendict-2.4.2-py312h06a4308_0

The following packages will be UPDATED:

  conda                                  24.3.0-py312h06a4308_0 --> 24.5.0-py312h06a4308_0
  openssl                                     3.0.13-h7f8727e_0 --> 3.0.13-h7f8727e_1



Downloading and Extracting Packages:
openssl-3.0.13       | 5.2 MB    | :   0% 0/1 [00:00<?, ?it/s]
conda-24.5.0         | 1.2 MB    | :   0% 0/1 [00:00<?, ?it/s]

frozendict-2.4.2     | 36 KB     | :   0% 0/1 [00:00<?, ?it/s]
openssl-3.0.13       | 5.2 MB    | :   0% 0.002997347135570501/1 [00:00<00:50, 50.71s/it]

frozendict-2.4.2     | 36 KB     | :  44% 0.43853215920344746/1 [00:00<00:00,  2.77it/s]

frozendict-2.4.2     | 36 KB     | : 100% 1.0/1 [00:00<00:00,  2.77it/s]
conda-24.5.0         | 1.2 MB    | : 100% 1.0/1 [00:00<00:00,  1.84it/s]



Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

```
!conda --version
!python --version
```

```
conda 24.5.0
Python 3.12.2
```

```
!conda install -c conda-forge fmpy --yes # Install the key package
```

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

```
#!conda install -c conda-forge matplotlib --yes
```

```
#!conda install -c conda-forge scipy --yes
```

```
#!conda install -c conda-forge openpyxl --yes
```

```
#!conda install -c conda-forge xlrd --yes
```

Notes of BPL_TEST2_Perfusion

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.

Now specific installation run a simulation and notebook for that Start with connecting to Github. Then upload the two files:

- FMU - BPL_TEST2_Perfusion_linux_om_me.fmu
- Setup-file - BPL_TEST2_Perfusion_fmpy_explore.py

```
%%bash
git clone https://github.com/janpeter19/BPL_TEST2_Perfusion
```

```
Cloning into 'BPL_TEST2_Perfusion'...
```

```
%cd BPL_TEST2_Perfusion
```

```
/content/BPL_TEST2_Perfusion
```

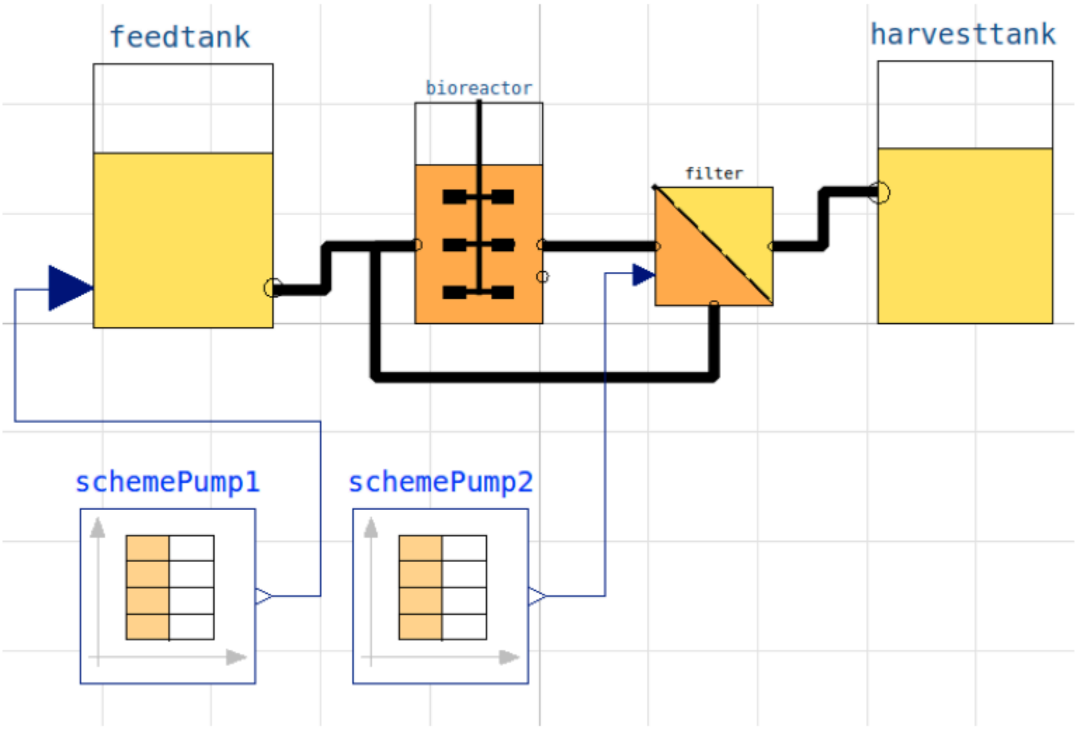
```
run -i BPL_TEST2_Perfusion_fmpy_explore.py
```

```
Linux - run FMU pre-comiled OpenModelica 1.21.0

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()
```

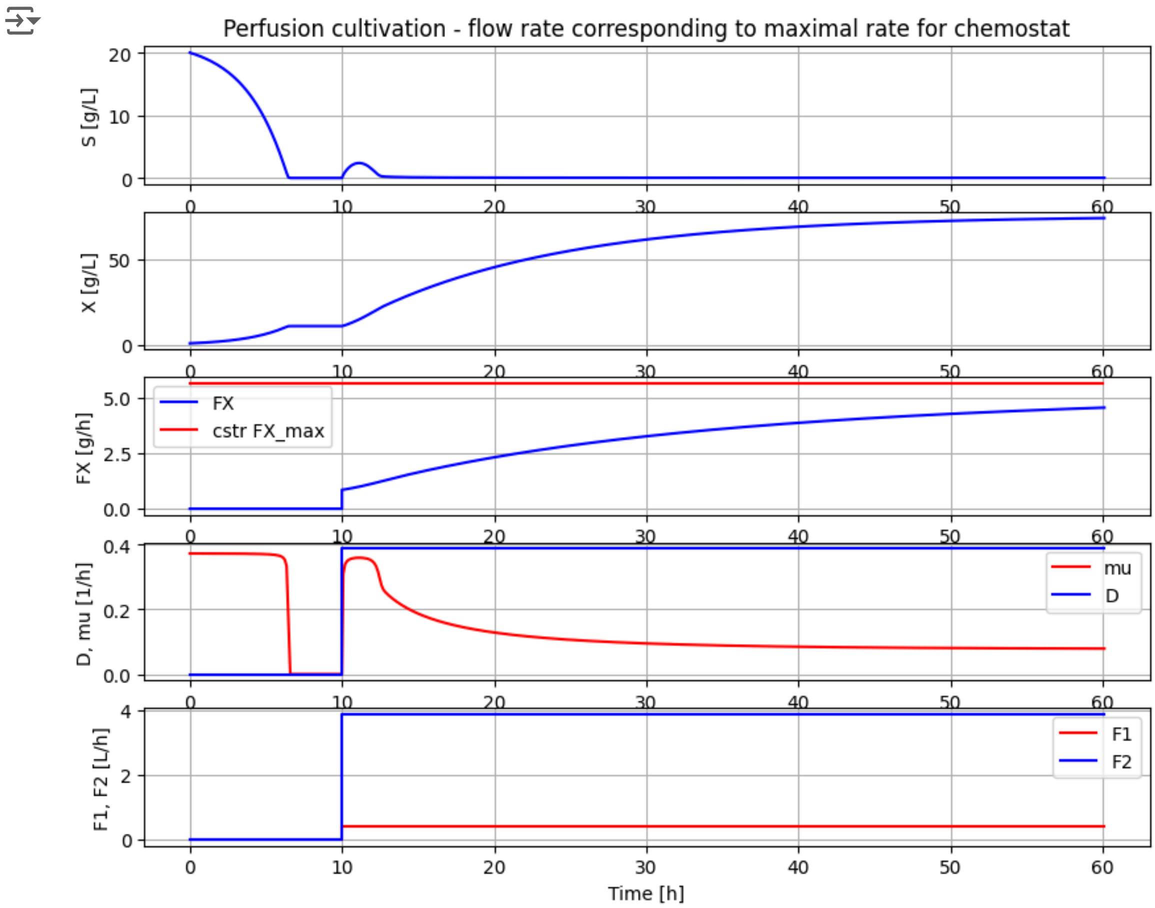
```
%matplotlib inline
plt.rcParams['figure.figsize'] = [25/2.54, 20/2.54]
```

```
process_diagram()
```

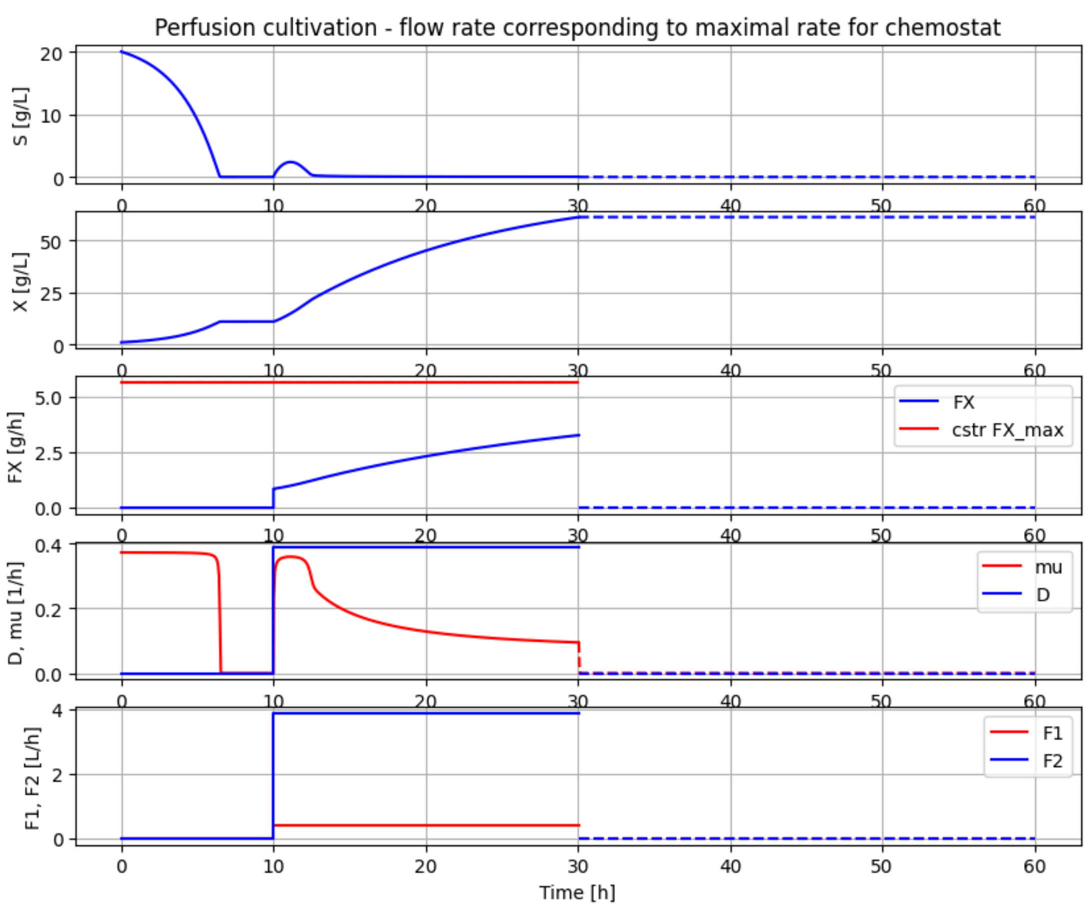
```
No processDiagram.png file in the FMU, but try the file on disk.
```



```
# 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 substrate conc
init(V_start=1.0, VX_start=1.0)                               # Process initial conditions
eps = parDict['filter_eps']                                   # Pump schedule parameter


# 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 - recycle flow 10 times pe
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_t4=960)

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




```
# Simulation of process with flow rate close to wash-out for chemostat

init(VS_start=20)                                   # Process initial
par(pump1_t1=10, pump2_t1=10)                       # Pump schedule - recycle flow 10 times pe
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_t4=960)

newplot(title='Perfusion cultivation - flow rate corresponding to maximal rate for chemostat')
simu(30)
simu(30,'cont')
```




```
# 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))
```

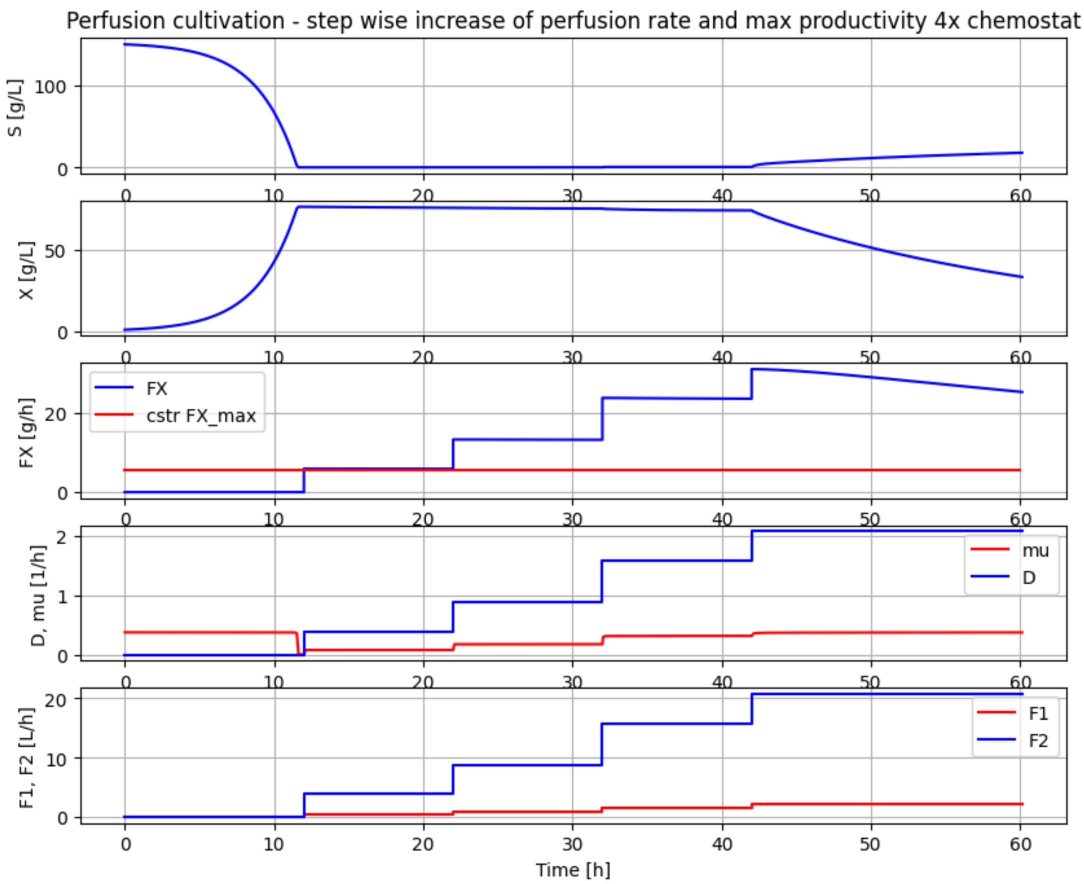
```
Conc factor of perfusion filter = 1.186
```

```
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), np.round(max(c_data[151:]),3))
```

```
Conc factor variation 1.186 1.186
```

```
# 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 rate.

init(VS_start=150)                                  # Process initial varied

par(pump1_t1=12, pump2_t1=12)                       # Pump schedule - recycle flow 10 times pe
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 max productivity 4x
simu(60)
```




```
# Simulation without a plot and just to check typical values at high production rate
#simu(40)
#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.round(max(c_data[190:]),
```

```
#describe('cstrProdMax')
```

```
# 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)
```

```
25.2
```

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

```
4.2
```

```
# Finally we check the filter flow rates at time 40 hour - note the negative sign for outlfow
model_get('filter.inlet.F')
```

```
20.749999999999996
```

```
model_get('filter.filtrate.F')
```

```
-2.0749999999999997
```

```
model_get('filter.retentate.F')
```

```
-18.674999999999997
```

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.

```
# List of components in the process setup and also a couple of other things like liquidphase and D
describe('parts')
```

```
['bioreactor', 'bioreactor.culture', 'D', 'feedtank', 'filter', 'harvesttank', 'schemePump1',
```

```
describe('MSL')
```

```
MSL: 3.2.3 - used components: RealInput, RealOutput, CombiTimeTable, Types
```

```
system_info()
```

```
System information
 -OS: Linux
 -Python: 3.10.12
 -Scipy: not installed in the notebook
 -FMPy: 0.3.20
 -FMU by: OpenModelica Compiler OpenModelica 1.21.0
 -FMI: 2.0
 -Type: ME
 -Name: BPL_TEST2.Perfusion
 -Generated: 2024-03-05T09:10:09Z
 -MSL: 3.2.3
 -Description: Bioprocess Library version 2.2.0
 -Interaction: FMU-explore for FMPy version 1.0.0
```