→ BPL_TEST2_Perfusion script with FMPy ver 0.3.15

The key library FMPy v0.3.15 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 20.04.5 LTS
    Release:
                    20.04
    Codename:
                    focal
%env PYTHONPATH=
    env: PYTHONPATH=
!wget https://repo.anaconda.com/miniconda/Miniconda3-py38_22.11.1-1-Linux-x86_64.sh
!chmod +x Miniconda3-py38_22.11.1-1-Linux-x86_64.sh
!bash ./Miniconda3-py38_22.11.1-1-Linux-x86_64.sh -b -f -p /usr/local
import sys
sys.path.append('/usr/local/lib/python3.8/site-packages/')
    --2023-03-28 08:04:04-- https://repo.anaconda.com/miniconda/Miniconda3-py38_22.11.1-1-Linux-x86_64.sh
    Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.131.3, 104.16.130.3, 2606:4700::6810:8203, ...
    Connecting to repo.anaconda.com (repo.anaconda.com) | 104.16.131.3 | :443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 64630241 (62M) [application/x-sh]
    Saving to: 'Miniconda3-py38_22.11.1-1-Linux-x86_64.sh'
    Miniconda3-py38 22. 100%[===========] 61.64M
    2023-03-28 08:04:05 (128 MB/s) - 'Miniconda3-py38 22.11.1-1-Linux-x86 64.sh' saved [64630241/64630241]
    PREFIX=/usr/local
    Unpacking payload ...
    Installing base environment...
    Downloading and Extracting Packages
    Downloading and Extracting Packages
    Preparing transaction: done
    Executing transaction: done
    installation finished.
!conda update -n base -c defaults conda --yes
```

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
!conda --version
!python --version

conda 23.1.0
Python 3.8.15
```

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

```
Preparing transaction: done
Verifying transaction: done

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

matplotlib-base-3.7. | 6.7 MB | : 100% 1.0/1 [00:00<00:00, 1.21it/s]
```

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

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

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

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

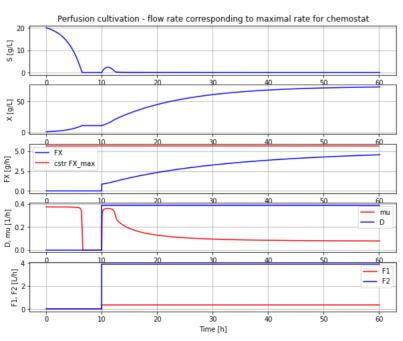
Notes of BPL_TEST2_Perfusion

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.

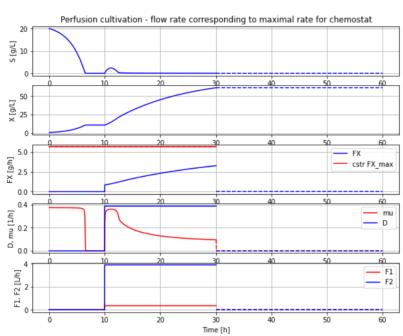
Now specific installation run a simulation and notebook for that Start with connecting to Github. Then upload the two files:

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

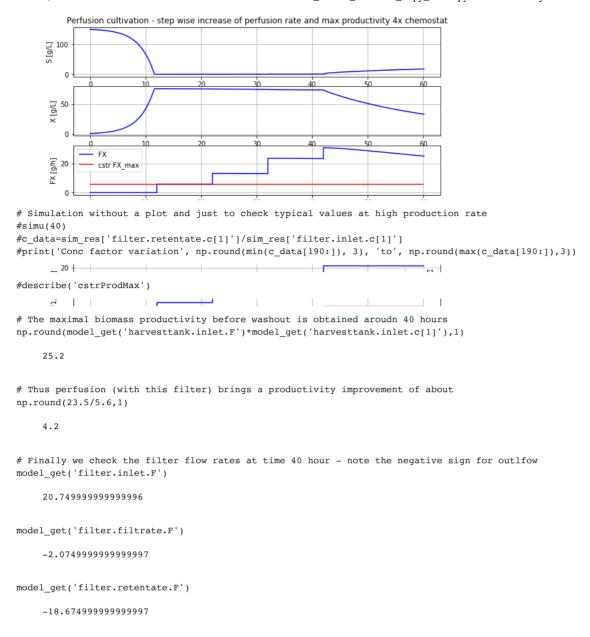
```
git clone https://github.com/janpeter19/BPL TEST2 Perfusion
    Cloning into 'BPL_TEST2_Perfusion'...
%cd BPL TEST2 Perfusion
    /content/BPL TEST2 Perfusion
run -i BPL TEST2_Perfusion_fmpy_explore.py
    Linux - run FMU pre-comiled OpenModelica 1.21.0
    Model for bioreactor has been setup. Key commands:
     - par()
                   - change of parameters and initial values
                   - change initial values only
       init()
     - simu()
                   - simulate and plot
     - newplot()
                   - make a new plot
                   - show plot from previous simulation
     - show()
                   - display parameters and initial values from the last simulation
     - disp()
     - describe() - describe culture, broth, parameters, variables with values/units
    Note that both disp() and describe() takes values from the last simulation
    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 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 0=1.0, VX 0=1.0)
                                                                    # Process initial conditions that are common
eps = parDict['filter eps']
                                                                    # Pump schedule parameter
# 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 - recycle flow 10 times perfusion flow
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
```



```
# 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 = nan
    <ipython-input-18-4726d2554b9f>:2: RuntimeWarning: invalid value encountered in double_scalars
      c=model get('filter.retentate.c[1]')/model get('filter.inlet.c[1]')
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 nan nan
    <ipython-input-19-0b550512bc49>:1: RuntimeWarning: invalid value encountered in true_divide
      c_data=sim_res['filter.retentate.c[1]']/sim_res['filter.inlet.c[1]']
# 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 0=150)
                                                         # Process initial varied
par(pump1_t1=12, pump2_t1=12)
                                                         # Pump schedule - recycle flow 10 times perfusion flow
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 chemostat')
simu(60)
```



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 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.

```
# 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', 'schemePump2']
describe('MSL')
   MSL: 3.2.3 - used components: RealInput, RealOutput, CombiTimeTable, Types
system_info()
```

```
System information
-OS: Linux
-Python: 3.9.16
-Scipy: not installed in the notebook
-FMPy: 0.3.15
-FMU by: OpenModelica Compiler OpenModelica 1.21.0~dev-185-g9d983b8
-FMI: 2.0
-Type: ME
-Name: BPL_TEST2.Perfusion
-Generated: 2023-01-25T11:02:31Z
-MSL: 3.2.3
-Description: Bioprocess Library version 2.1.1-beta
-Interaction: FMU-explore for FMPy version 0.9.7
```

✓ 0s completed at 10:10

• x