

## ✓ BPL\_TEST2\_Perfusion script with PyFMI

The key library PyFMI 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-py310_23.1.0-1-Linux-x86_64.sh
```

```
!chmod +x Miniconda3-py310_23.1.0-1-Linux-x86_64.sh
```

```
!bash ./Miniconda3-py310_23.1.0-1-Linux-x86_64.sh -b -f -p /usr/local
```

```
import sys
```

```
sys.path.append('/usr/local/lib/python3.10/site-packages/')
```

```

--2025-01-14 09:31:53-- https://repo.anaconda.com/miniconda/Miniconda3-py310_23.1.0-1-Linux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.191.158, 104.16.32.241, 2606:4700::6810:bf9e,
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.191.158|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 74403966 (71M) [application/x-sh]
Saving to: 'Miniconda3-py310_23.1.0-1-Linux-x86_64.sh'

```

```
Miniconda3-py310_23 100%[=====>] 70.96M 90.2MB/s in 0.8s
```

```
2025-01-14 09:31:54 (90.2 MB/s) - 'Miniconda3-py310_23.1.0-1-Linux-x86_64.sh' saved [74403966/74403966]
```

```

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 3.10.16
```

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

```
🔄
```

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

## ✓ 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\_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_explore.py
```

```
📂 Linux - run FMU pre-compiled OpenModelica
```

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

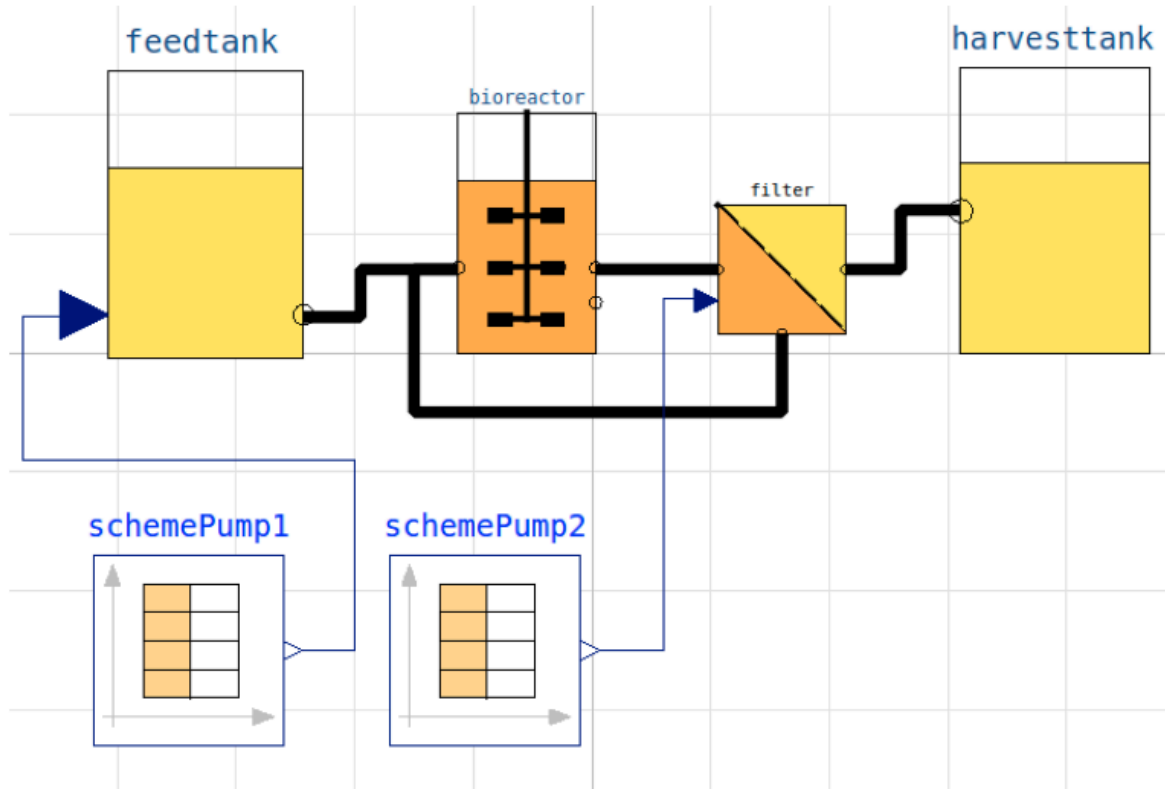
Brief information about a command by `help()`, eg `help(simu)`  
Key system information is listed with the command `system_info()`

```
# Filter out DeprecationWarnings for 'np.float as alias' is needed - wish
import warnings
warnings.filterwarnings("ignore")
```

```
%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)
par(filter_eps=0.10, filter_alpha_X=0.02, filter_alpha_S=0.10)
par(S_in=30.0)
init(V_start=1.0, VX_start=1.0)
eps = parDict['filter_eps']

# Culture
# Filter
# Inlet substrate conc
# Process initial conditions that are cc
# Pump schedule parameter

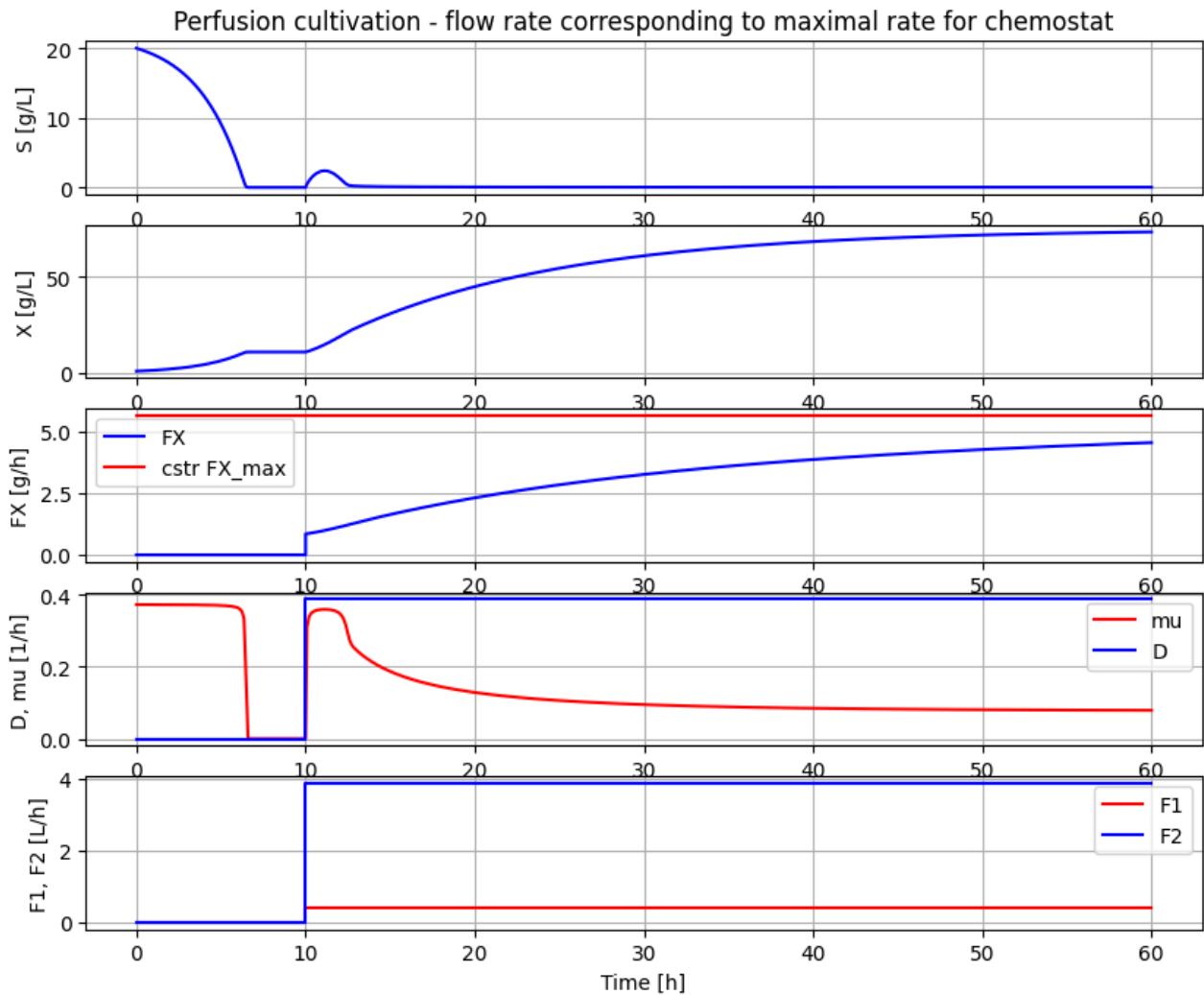
# Simulation of process with flow rate close to wash-out for chemostat

init(VS_start=20)
par(pump1_t1=10, pump2_t1=10)
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)

# Process initial
# Pump schedule - recycle flow 10 times perfusion f

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

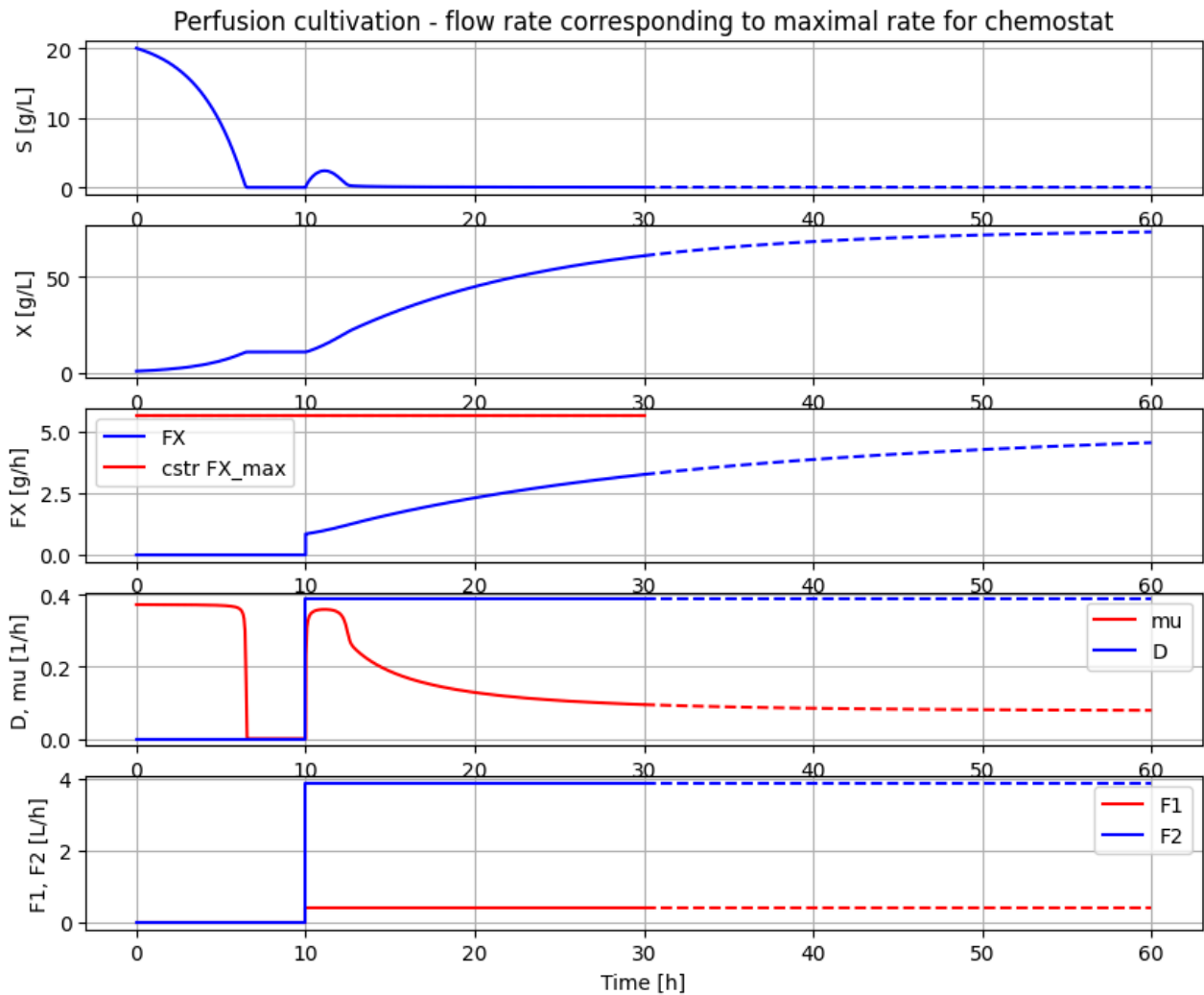
↳ Could not find cannot import name 'dopri5' from 'assimulo.lib' (/usr/local/lib/python3.10/site-packages/)  
 Could not find cannot import name 'rodas' from 'assimulo.lib' (/usr/local/lib/python3.10/site-packages/a  
 Could not find cannot import name 'odassl' from 'assimulo.lib' (/usr/local/lib/python3.10/site-packages/  
 Could not find ODEPACK functions.  
 Could not find RADAR5  
 Could not find GLIMDA.



# 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 perfusion f
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')
```



Note the inability of the OpenModelica FMU to handle `simu('cont')` properly.

```
# 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

```
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 1.179 1.179

```
# 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 rate.
```

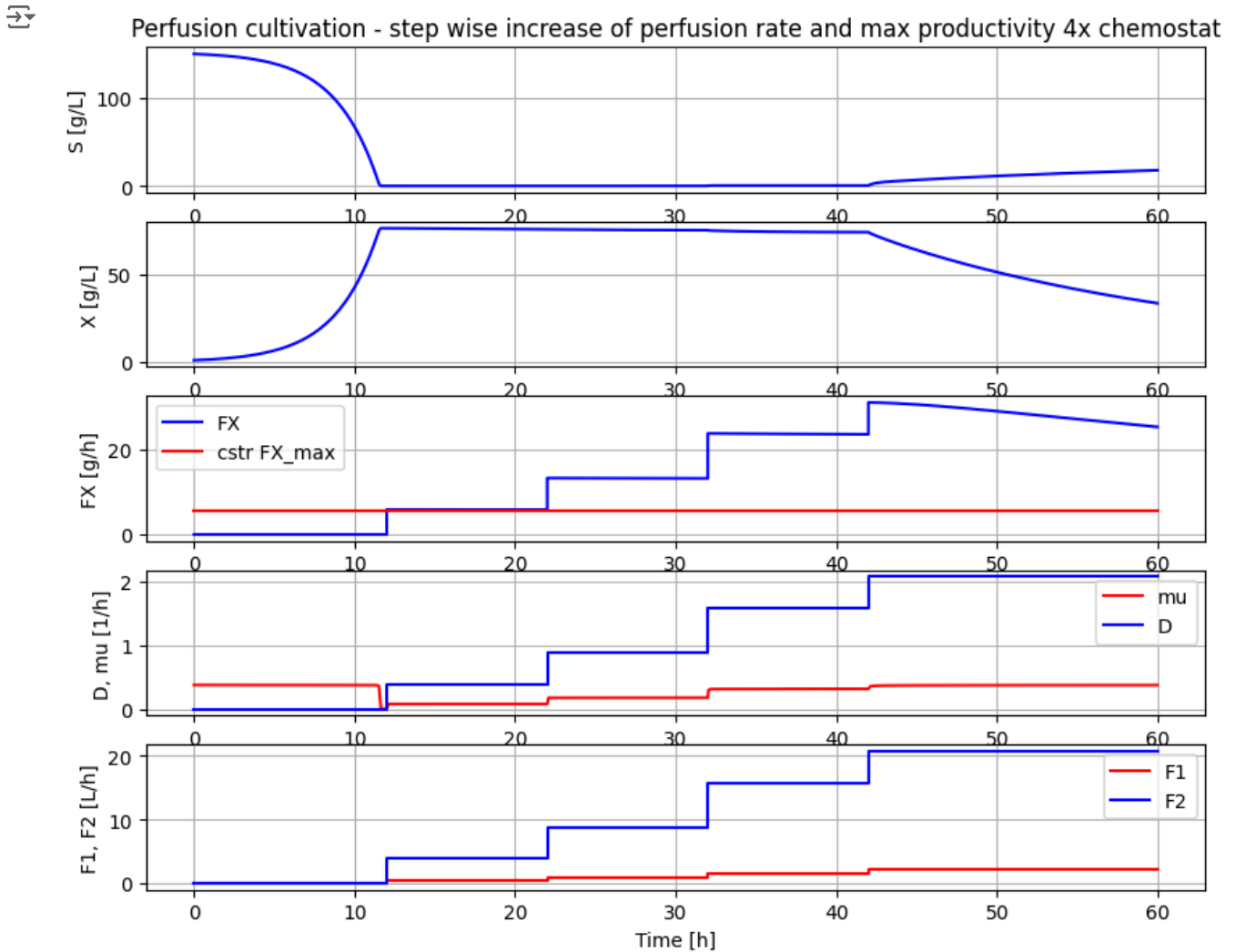
```
init(VS_start=150)
```

```
# Process initial varied
```

```
par(pump1_t1=12, pump2_t1=12)
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)
```

```
# Pump schedule - recycle flow 10 times perfusion f
```

```
newplot(title='Perfusion cultivation - step wise increase of perfusion rate and max productivity 4x chemosta
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:]),3))
```

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

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

```
# The maximal biomass productivity before washout is obtained around 40 hours
np.round(model.get('harvesttank.inlet.F')[0]*model.get('harvesttank.inlet.c[1]')[0],1)
```

```
23.5
```

```
# 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 outflow
model.get('filter.inlet.F')[0]
```

```
15.749999999999998
```

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

```
-1.575
```

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

↻ -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 concentration 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 decrease 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', 'schemePum

```
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
```

```
-PyFMI: 2.16.1
```

```
FMU: C:\Users\janpeter\AppData\Local\Temp\1\133-54705
```