BPL_IEC_validation script with PyFMI

The key library PyFMI is installed.

After the installation a small application BPL_IEC_validation 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.
!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/')
```

```
--2024-07-19 09:33:34--  https://repo.anaconda.com/miniconda/Miniconda3-py310
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.32.241, 104.16.191.1
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.32.241|:443... conn
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   191MB/s    in 0.4s

2024-07-19 09:33:35 (191 MB/s) - ‘Miniconda3-py310_23.1.0-1-Linux-x86_64.sh’ s

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 3.10.14
```

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

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

```
# Notes of BPL_IEC_validation
```

Now specific installation and the run simulations. Start with connecting to Github. Then upload the two files:

- FMU - BPL_IEC_Column_system_linux_om_me
- Setup-file - BPL_IEC_explore

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

```
Cloning into 'BPL_IEC_validation'...
```

```
%cd BPL_IEC_validation
```

```
/content/BPL_IEC_validation
```

BPL IEC validation

Author: Jan Peter Axelsson

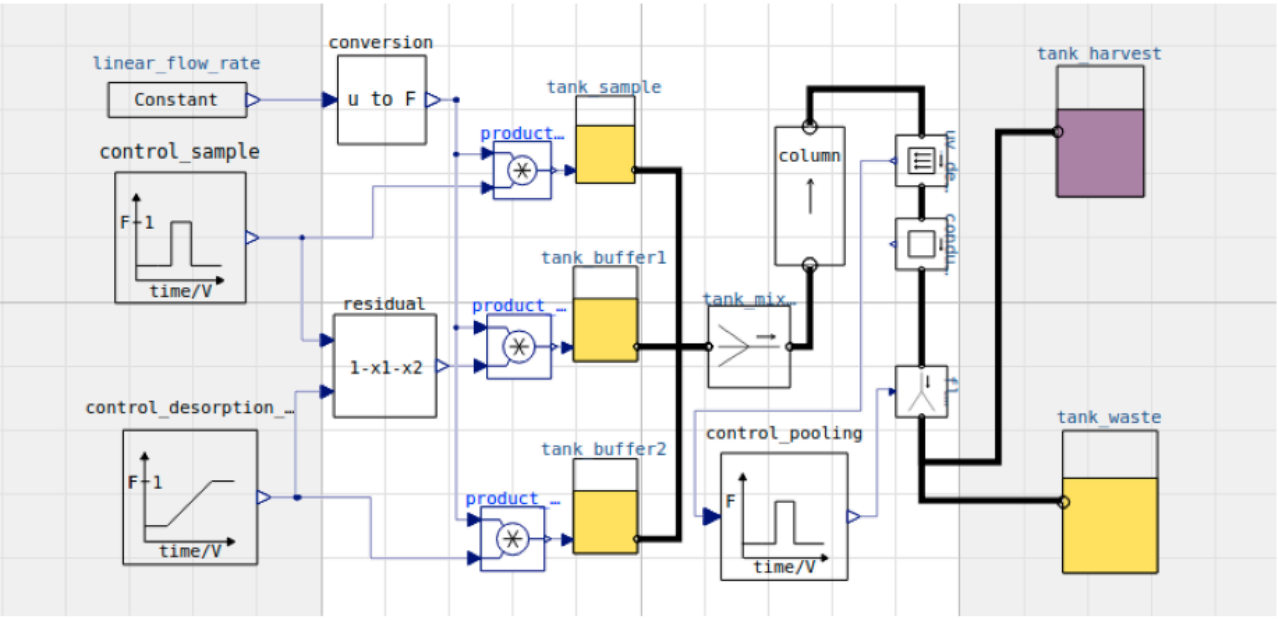
```
run -i BPL_IEC_explore.py
```

```
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 simulati
     - describe()  - describe culture, broth, parameters, variables with values/u

    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()
```

```
plt.rcParams['figure.figsize'] = [30/2.54, 24/2.54]
```

```
# The process diagram is made outside Modelica to illustrate the configuration
process_diagram()
```

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




```
describe('chromatography'); # print(); describe('liquidphase')
```

```
Ion exchange chromatorgraphy controlled with varying salt-concentration. The |
```

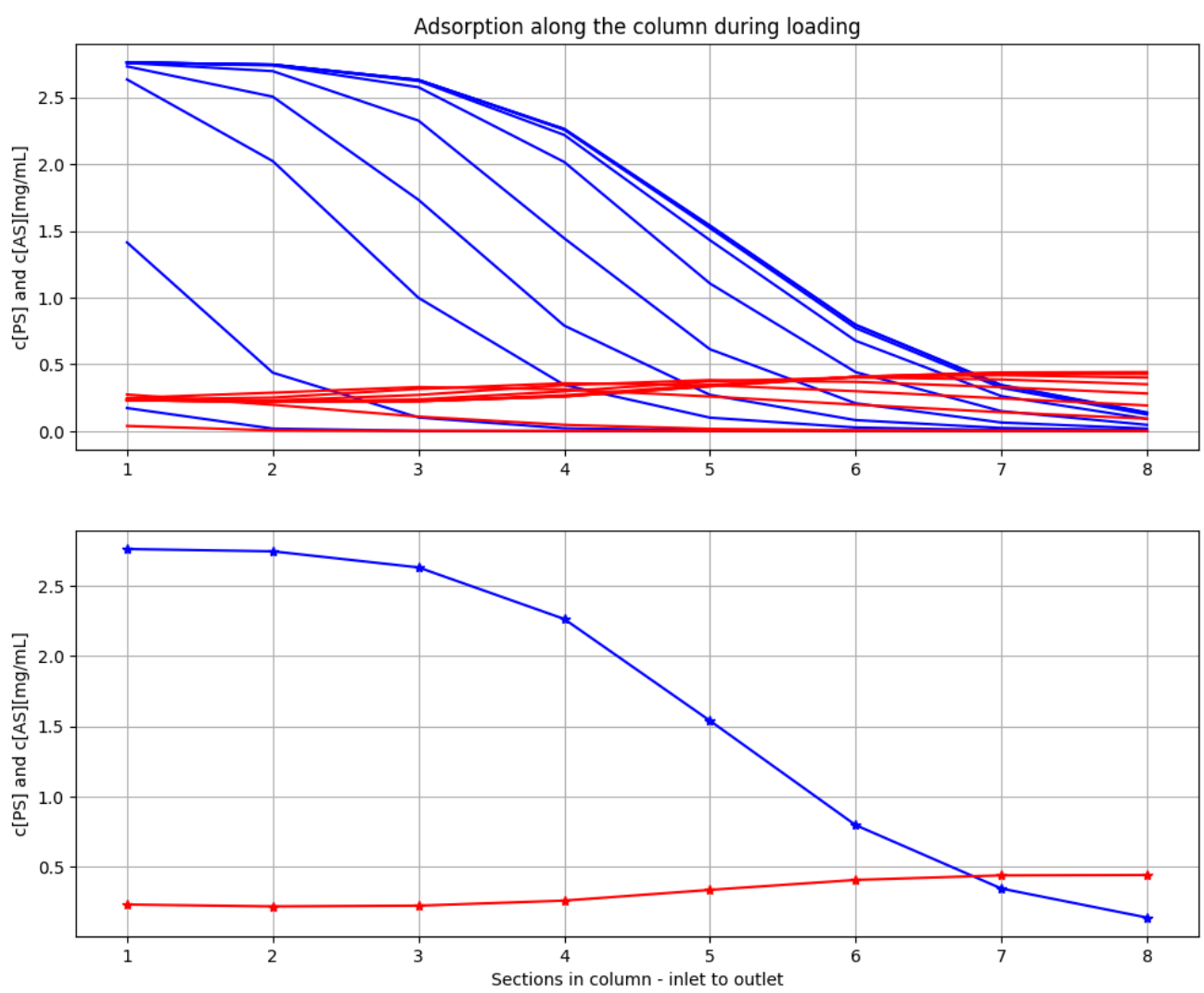
Loading or adsorption

The parameter notation and values are the same as in the referred report. However the flow rate is here denoted F while q in the report. The column is diveded in n=8 sections and set at compilation time. The values are arbitrarly chosen in the report and the focus is on qualitative aspects of the model.

The simplified model describe only the column in terms of volume and does not distinguish a high column with a small diameter from a lower with larger diameter.

The parameters k1, k2, k3, k4 and Q_av are given relative volume and with increased column volume a larger capacity is thus obtained.

```
# Loading of the column - try to reproduce Jonas figure 13.
newplot(title='Adsorption along the column during loading', plotType='Loading')

# Sample
par(P_in=1.0, A_in=1.0, E_in=0)

# Column properties
par(k1=0.3, k2=0.05, k3=0.05, k4=0.3, Q_av=3.0)
par(height=20, diameter=0.714)
par(x_m=0.3)

# Operation
par(E_in_desorption_buffer=8)
par(LFR=12)
#par(scale_volume=False)
par(start_adsorption=0, stop_adsorption=50)
par(start_desorption=150, stationary_desorption=450)
par(start_pooling=220, stop_pooling=450)

# Simulation
simu(100)
```



The results are the same af Figure 13 in [1].

```
# We just check that we had the same volume flow rate as Jonas
describe('F')
```

```
: 0.08
```

```
describe('V')
```

```
Column volume total - derived : 8.008 [ mL ]
```

```
model.get('column.x_m')
```

```
array([0.3])
```

```
model.get('column.V_m')
```

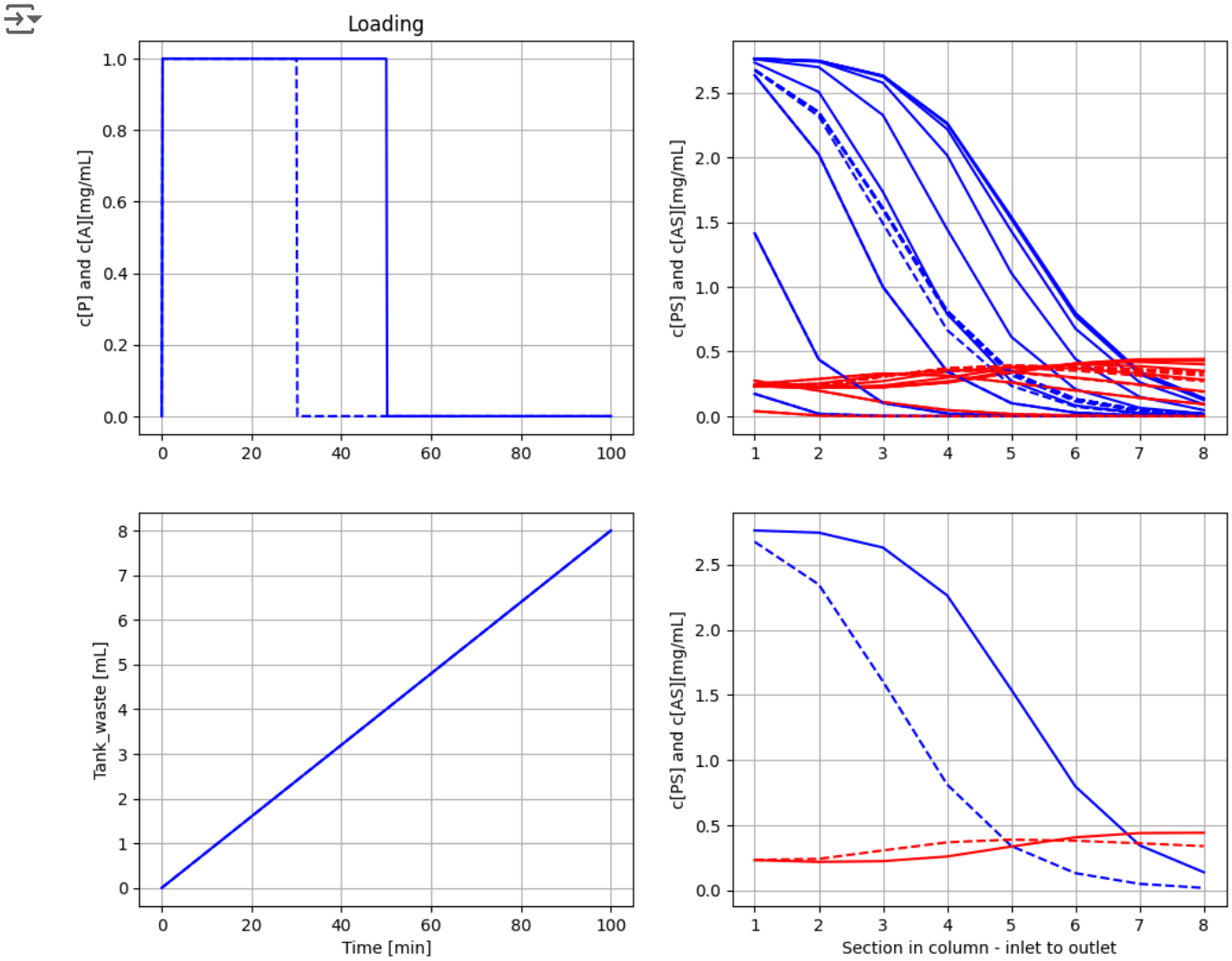
```
array([2.40235705])
```

```
#describe('column.n')
```

```
model.get('column.column_section[1].V_m')
```

```
array([0.30029463])
```

```
# Impact of shorter time for loading and then less material
newplot(title='Loading', plotType='Loading-combined')
show()

# Simulation with changed parameter t2
par(stop_adsorption=30); simu(100)

# Reset changed parameter
par(stop_adsorption=50)
```



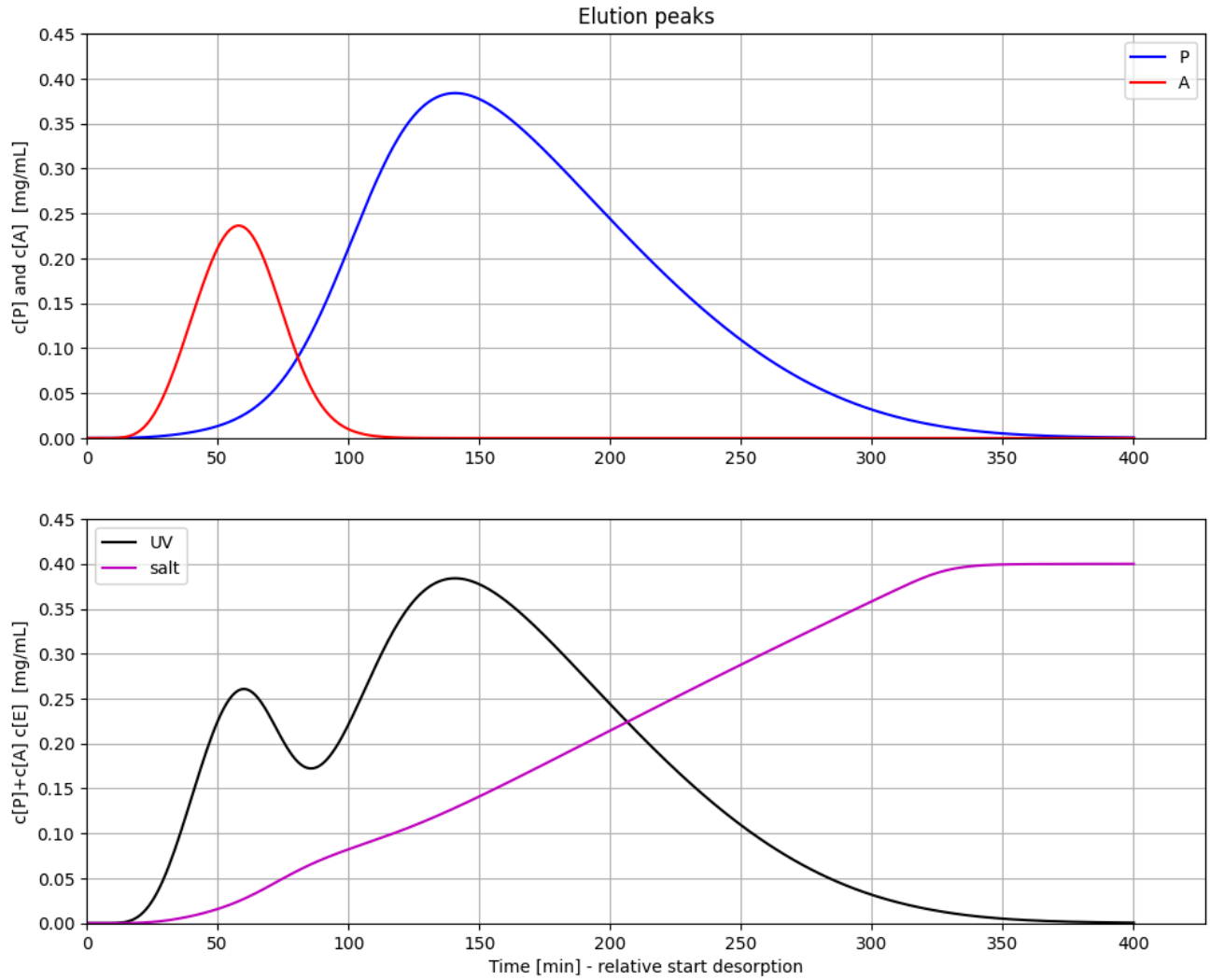
To the left the inlet loading over time. To the right upper concentration along the column at different times and in steady states finally To the right lower concentrations along the column in steady state.

We see that a shorter time and then less material makes less of the column capacity used.

Note that the flow through the column is constant despite change from sample to just buffer 1, and shown in how the volume of the waste tank increase with time.

Elution or desorption

```
# Elution of the column
newplot(title='Elution peaks', plotType='Elution')

# Sample
par(P_in=1, A_in=1.0, E_in=0)

# Operation
par(E_in_desorption_buffer=8)
par(LFR=12.0, start_adsorption=0, stop_adsorption=50, start_desorption=150, stati

# Simulation
simu(550)
```



The results are the same af Figure 14 in [1].

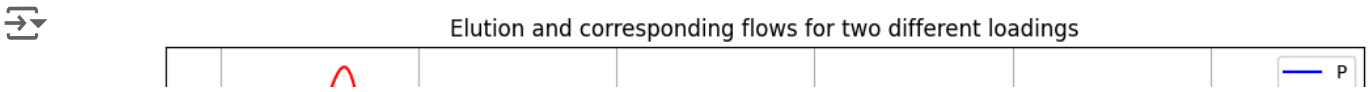
The upper diagrams shows the column outlet concentrations of P and A over time.

The lower diagram shows the sum (or possibly the UV signal) at column outlet as well as the salt concentration. We have some separation between the two peaks.

Note that the salt concentration deviates slightly from the linear increse between time 50 to 100. This is due to ion interaction with P and A in the column. This is phenomenon can also be seen in real data. The ion-salt concentration is scaled with factor 0.05 to get comparable concentrations to P and A.

I have here simulated time 150 of adsorbtion and then started elution. Here is time counted as zero at time of start of elutions. Not sure how long Jonas simulated to get steady state before he did elution.

```
# More complete visualization of the elution phase and the different flows
newplot(title='Elution and corresponding flows for two different loadings', plotT
par(stop_adsorption=50); simu(550)

# Simulation with changed parameter t2
par(stop_adsorption=30); simu(550)

# Reset changed parameter
par(stop_adsorption=50)
```



Here a diagram that shows the peaks at the outlet as shown in the previous diagram. Below the flow rates of the three differnet sorces. Here time is 0 at start of adsorbtion and elution starts at time 150.

Automatic pooling based on UV-mmeasurement is tested in another notebook.

Change of resin properties


```
# Elution of the column
newplot(title='Elution peaks for different resin parameters k1 and k2', plotType=

# Sample
par(P_in=1, A_in=0.0, E_in=0)

# Operation
par(E_in_desorption_buffer=8)
par(LFR=12.0, start_adsorption=0, stop_adsorption=50, start_desorption=150, stati

# Simulations
par(k1=0.05, k2=0.50); simu(550)
par(k1=0.05, k2=0.25); simu(550)
par(k1=0.05, k2=0.05); simu(550)
par(k1=0.25, k2=0.05); simu(550)
par(k1=0.50, k2=0.05); simu(550)

# Adjust diagrams
ax1.set_ylim([0, 0.65])
ax2.set_ylim([0, 0.65])
plt.show()
```