BPL_CHO_Fedbatch script with PyFMI ver 2.7.4

The key library PyFMI v2.7.4 is installed and downgrading is done Numpy v1.19.1. To simplify this we first install conda.

After the installation a small application BPL_CHO_Fedbatch 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 18.04.6 LTS
Release:        18.04
Codename:       bionic
```

```
%env PYTHONPATH=
```

```
env: PYTHONPATH=
```

```
!wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh
!chmod +x Miniconda3-py37_4.12.0-Linux-x86_64.sh
!bash ./Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -f -p /usr/local
import sys
sys.path.append('/usr/local/lib/python3.7/site-packages/')
```

```
--2022-10-17 08:07:36--  https://repo.anaconda.com/miniconda/Miniconda3-py37_4
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.131.3, 104.16.130.3,
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.131.3|:443... conne
HTTP request sent, awaiting response... 200 OK
Length: 104996770 (100M) [application/x-sh]
Saving to: 'Miniconda3-py37_4.12.0-Linux-x86_64.sh'

Miniconda3-py37_4.1 100%[===================>] 100.13M   135MB/s    in 0.7s

2022-10-17 08:07:37 (135 MB/s) - 'Miniconda3-py37_4.12.0-Linux-x86_64.sh' save

PREFIX=/usr/local
Unpacking payload ...
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible
Solving environment: failed with repodata from current_repodata.json, will ret
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible
Solving environment: \
Found conflicts! Looking for incompatible packages.
This can take several minutes.  Press CTRL-C to abortfailed

UnsatisfiableError: The following specifications were found to be incompatible

Output in format: Requested package -> Available versions
```

```
Package ld_impl_linux-64 conflicts for:
ruamel_yaml==0.15.100=py37h27cfd23_0 -> python[version='>=3.7,<3.8.0a0'] -> ld
ld_impl_linux-64==2.35.1=h7274673_9
charset-normalizer==2.0.4=pyhd3eb1b0_0 -> python[version='>=3.5'] -> ld_impl_l
toolz -> python[version='>=3.5'] -> ld_impl_linux-64
requests==2.27.1=pyhd3eb1b0_0 -> python[version='>=3.6'] -> ld_impl_linux-64
pycosat==0.6.3=py37h27cfd23_0 -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_l
pyopenssl==22.0.0=pyhd3eb1b0_0 -> python[version='>=3.6'] -> ld_impl_linux-64
python==3.7.13=h12debd9_0 -> ld_impl_linux-64
python_abi -> python=3.7 -> ld_impl_linux-64
numpy-base -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_linux-64
pip==21.2.2=py37h06a4308_0 -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_linu
conda-package-handling==1.8.1=py37h7f8727e_0 -> python[version='>=3.7,<3.8.0a0
pyfmi -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_linux-64
lxml -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_linux-64
wheel==0.37.1=pyhd3eb1b0_0 -> python -> ld_impl_linux-64
idna==3.3=pyhd3eb1b0_0 -> python[version='>=3.5'] -> ld_impl_linux-64
cryptography==36.0.0=py37h9ce1e76_0 -> python[version='>=3.7,<3.8.0a0'] -> ld_
pysocks==1.7.1=py37_1 -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_linux-64
tqdm==4.63.0=pyhd3eb1b0_0 -> python[version='>=2.7'] -> ld_impl_linux-64
certifi==2021.10.8=py37h06a4308_2 -> python[version='>=3.7,<3.8.0a0'] -> ld_im
numpy -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_linux-64
setuptools==61.2.0=py37h06a4308_0 -> python[version='>=3.7,<3.8.0a0'] -> ld_im
conda-content-trust==0.1.1=pyhd3eb1b0_0 -> python -> ld_impl_linux-64
pycparser==2.21=pyhd3eb1b0_0 -> python[version='>=3.6'] -> ld_impl_linux-64
brotlipy==0.7.0=py37h27cfd23_1003 -> python[version='>=3.7,<3.8.0a0'] -> ld_im
cffi==1.15.0=py37hd667e15_1 -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_lin
assimulo -> python[version='>=3.7,<3.8.0a0'] -> ld_impl_linux-64
six==1.16.0=pyhd3eb1b0_1 -> python -> ld_impl_linux-64
colorama==0.4.4=pyhd3eb1b0_0 -> python -> ld_impl_linux-64
urllib3==1.26.8=pyhd3eb1b0_0 -> python[version='<4.0'] -> ld_impl_linux-64
```

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

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

Retrieving notices: ...working... done
```

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

```
conda 22.9.0
Python 3.7.13
```

```
!conda install -c conda-forge pyfmi==2.7.4 --yes # Install the key package
```

```
    libcblas-3.9.0              |16_linux64_openblas          13 KB  conda-forg
    libgcc-ng-12.2.0            |      h65d4601_18         936 KB  conda-forge
    libgomp-12.2.0              |      h65d4601_18         455 KB  conda-forge
    liblapack-3.9.0             |16_linux64_openblas          13 KB  conda-forg
    liblapacke-3.9.0            |16_linux64_openblas          13 KB  conda-forg
    libopenblas-0.3.21          |pthreads_h78a6416_3        10.1 MB  conda-forg
    llvm-openmp-14.0.6          |        h9e868ea_0         4.4 MB
    openblas-0.3.21             |pthreads_h320a7e8_3        10.8 MB  conda-forg
    openssl-1.1.1q              |        h166bdaf_0         2.1 MB  conda-forge
```

```
------------------------------------------------------------
                                       Total:       28.9 MB

The following NEW packages will be INSTALLED:

  blas-devel         conda-forge/linux-64::blas-devel-3.9.0-16_linux64_openbla
  liblapacke         conda-forge/linux-64::liblapacke-3.9.0-16_linux64_openbla
  llvm-openmp        pkgs/main/linux-64::llvm-openmp-14.0.6-h9e868ea_0 None
  openblas           conda-forge/linux-64::openblas-0.3.21-pthreads_h320a7e8_3

The following packages will be UPDATED:

  blas                         pkgs/main::blas-1.0-openblas --> conda-forge::b
  ca-certificates    pkgs/main::ca-certificates-2022.07.19~ --> conda-forge::c
  conda              pkgs/main::conda-22.9.0-py37h06a4308_0 --> conda-forge::c
  libblas                      3.9.0-15_linux64_openblas --> 3.9.0-16_linux
  libcblas                     3.9.0-15_linux64_openblas --> 3.9.0-16_linux
  libgcc-ng          pkgs/main::libgcc-ng-11.2.0-h1234567_1 --> conda-forge::l
  libgomp              pkgs/main::libgomp-11.2.0-h1234567_1 --> conda-forge::l
  liblapack                    3.9.0-15_linux64_openblas --> 3.9.0-16_linux
  libopenblas                0.3.20-pthreads_h78a6416_0 --> 0.3.21-pthread

The following packages will be SUPERSEDED by a higher-priority channel:

  _libgcc_mutex          pkgs/main::_libgcc_mutex-0.1-main --> conda-forge::_
  _openmp_mutex          pkgs/main::_openmp_mutex-5.1-1_gnu --> conda-forge::_
  certifi            pkgs/main/linux-64::certifi-2022.9.24~ --> conda-forge/nc
  openssl              pkgs/main::openssl-1.1.1q-h7f8727e_0 --> conda-forge::c



Downloading and Extracting Packages
blas-devel-3.9.0     | 12 KB     | : 100% 1.0/1 [00:00<00:00,  6.58it/s]
libgomp-12.2.0       | 455 KB    | : 100% 1.0/1 [00:00<00:00,  4.89it/s]
openssl-1.1.1q       | 2.1 MB    | : 100% 1.0/1 [00:00<00:00,  2.12it/s]
blas-2.116           | 13 KB     | : 100% 1.0/1 [00:00<00:00, 19.21it/s]
libopenblas-0.3.21   | 10.1 MB   | : 100% 1.0/1 [00:02<00:00,  2.21s/it]
libcblas-3.9.0       | 13 KB     | : 100% 1.0/1 [00:00<00:00, 21.17it/s]
_libgcc_mutex-0.1    | 3 KB      | : 100% 1.0/1 [00:00<00:00, 23.91it/s]
liblapacke-3.9.0     | 13 KB     | : 100% 1.0/1 [00:00<00:00, 20.16it/s]
liblapack-3.9.0      | 13 KB     | : 100% 1.0/1 [00:00<00:00, 20.23it/s]
libgcc-ng-12.2.0     | 936 KB    | : 100% 1.0/1 [00:00<00:00,  4.67it/s]
libblas-3.9.0        | 13 KB     | : 100% 1.0/1 [00:00<00:00, 21.40it/s]
llvm-openmp-14.0.6   | 4.4 MB    | : 100% 1.0/1 [00:00<00:00,  2.13it/s]
_openmp_mutex-4.5    | 6 KB      | : 100% 1.0/1 [00:00<00:00, 16.20it/s]
openblas-0.3.21      | 10.8 MB   | : 100% 1.0/1 [00:02<00:00,  2.93s/it]
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Retrieving notices:    working    done
```

```
!conda install numpy=1.19.1 --yes # Need to downgrade numpy
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local
```

```
    added / updated specs:
      - numpy=1.19.1


  The following packages will be SUPERSEDED by a higher-priority channel:

    ca-certificates    conda-forge::ca-certificates-2022.9.2~ --> pkgs/main::ca-
    certifi            conda-forge/noarch::certifi-2022.9.24~ --> pkgs/main/linu
    conda              conda-forge::conda-22.9.0-py37h89c186~ --> pkgs/main::con
    openssl            conda-forge::openssl-1.1.1q-h166bdaf_0 --> pkgs/main::ope


  Preparing transaction: done
  Verifying transaction: done
  Executing transaction: done
  Retrieving notices: ...working... done
```

BPL_CHO_Fedbatch setup

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

- FMU - BPL_CHO_Fedbatch_linux_jm_cs.fmu
- Setup-file - BPL_CHO_Fedbatch_explore

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

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

```
    Cloning into 'BPL_CHO_Fedbatch'...
```

```
%cd BPL_CHO_Fedbatch
```

```
    /content/BPL_CHO_Fedbatch/BPL_CHO_Fedbatch/BPL_CHO_Fedbatch
```

BPL_CHO_Fedbatch - demo

This notebook deals with CHO fedbatch cultivation and recombinant protein production is included. First we make a check of the model by comparing a simulation result with corresponding published diagram. Then we take a closer look at the start-up strategy to keep the by-product formation low. After that we investigate at a whole cultivation and see the impact of feeding strategy on both cell growth and protein production where a trade-off is needed in this case.

The model used takes its inspiration from the microbial bottleneck models as described in the original papers [1] and [2] and reformulated and studied in [3]. The laboratory cultures used for model validation in [1] did produce MAb (against part of IgG) but no MAb-data was presented. The paper focus on viable and non-viable cell concentrations only. The original model is in section 5 expanded with the classical empirical Luedeking-Piret model recombinant protein production, see chapter 5 in [4]. In this way can get more insight into choice of feeding profile.

Interaction with the compiled model as FMU is mainly through the simplified commands: par(), init(), newplot(), simu() etc. The last simulation is always available in the workspace and called 'sim_res'. The command describe() brings mainly up description infomration from the actual Modelica code from the FMU but is complemented with information given in the dedicated Python setup-file.

The idea here is to demonstrate how simulations and varyiing conditions can provide some process insight that can support the experimental work. I hope that at the end of this session you are ready to formulate your own questions you want to address with simulations - and you can just go on in this notebook! Just press the field "+Code" in the upper left part of notebook interface and you get a new "cell" where you write your own code. You can copy and paste from cells above using ctrl-c and ctrl-v as usual and edit the cell. When your are ready to execute the cell just press the "play button" to the left in the cell or press shift-enter as in "ordinary" Jupyter notebooks.

After a session you may want to save your own notebook. That you can do on your Google Drive account and I refer to Colab instructions for how to do this. It is easy.

Good luck!

```
run -i BPL_CHO_Fedbatch_explore.py
```

```
Linux - run FMU pre-comiled JModelica 2.4

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 simula
 - describe()  - describe culture, broth, parameters, variables with values

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()
<Figure size 708.661x566.929 with 0 Axes>
```

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

1 About the process model

We can get information about the process and liquid phase by the command describe(). Here is no gas-phase included. This command can also be used to bring up information about a specific variable or parameter. However, you should use describe() after a simulation to get the valued used during the simulation.

```
describe('culture'); print(); describe('liquidphase')
```

```
Reactor culture CHO-MAb - cell line HB-58 American Culture Collection ATCC

Reactor broth substances included in the model

Cells viable index =  1 molecular weight =  24.6 Da
Cells dead   index =  2 molecular weight =  24.6 Da
Glucose      index =  3 molecular weight =  180.0 Da
Glutamine    index =  4 molecular weight =  146.1 Da
Lactate      index =  5 molecular weight =  90.1 Da
Ammonia      index =  6 molecular weight =  17.0 Da
Protein      index =  7 molecular weight =  150000.0 Da
```

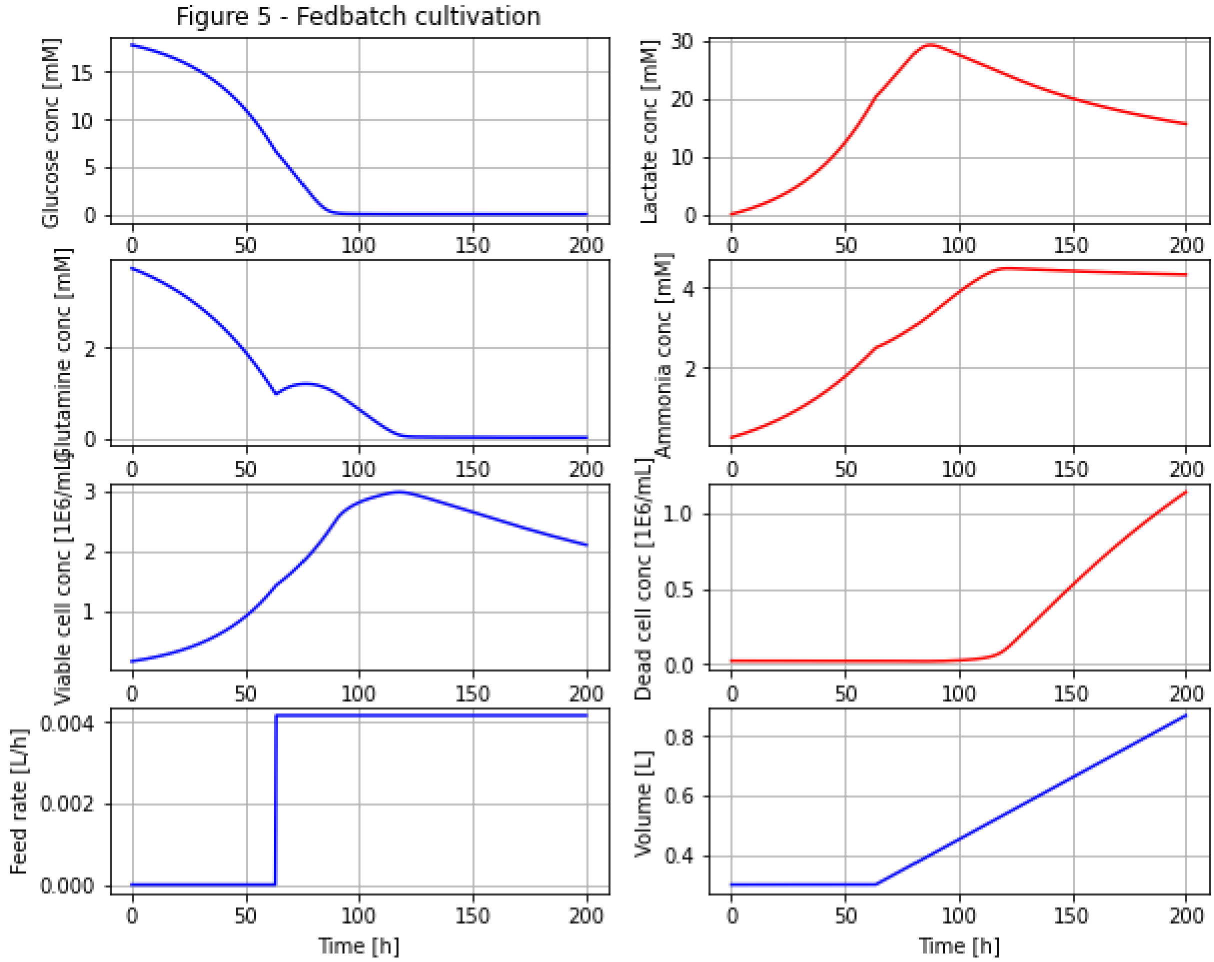
The molecular weight of the recombinant protein (MAb) is somewhat arbitrarly chosen and the value not used in the simulations.

2 Simulation reproducing the original paper

The simulation below reproduce diagrams in Figure 5 in the original paper. There are several simulation in the paper showing how well the model describe different experiments and here I just choose one of them.

```
# Data from Table 1 and 2 for experiment 4 shown in Figure 5 in paper [1]
# -culture parameters taken from Table 5 identified parameters for cultures 1,2,and

# Initial process conditions
V_0=0.30
init(V_0=V_0, VXv_0=V_0*0.172, VXd_0=V_0*0.020)
init(VG_0=V_0*17.83, VGn_0=V_0*3.74, VL_0=V_0*0.12, VN_0=V_0*0.24)

# Feeding
Feed=0.1/24
par(G_in=15, Gn_in=9.3)
par(t0=0, F0=0, t1=63.5, F1=Feed, t2=300, F2=Feed)

# Simulation
newplot(title='Figure 5 - Fedbatch cultivation')
simu(200)
```

```
Simulation interval    : 0.0 - 200.00000000000003 seconds.
Elapsed simulation time: 0.01600714100004552 seconds.
```



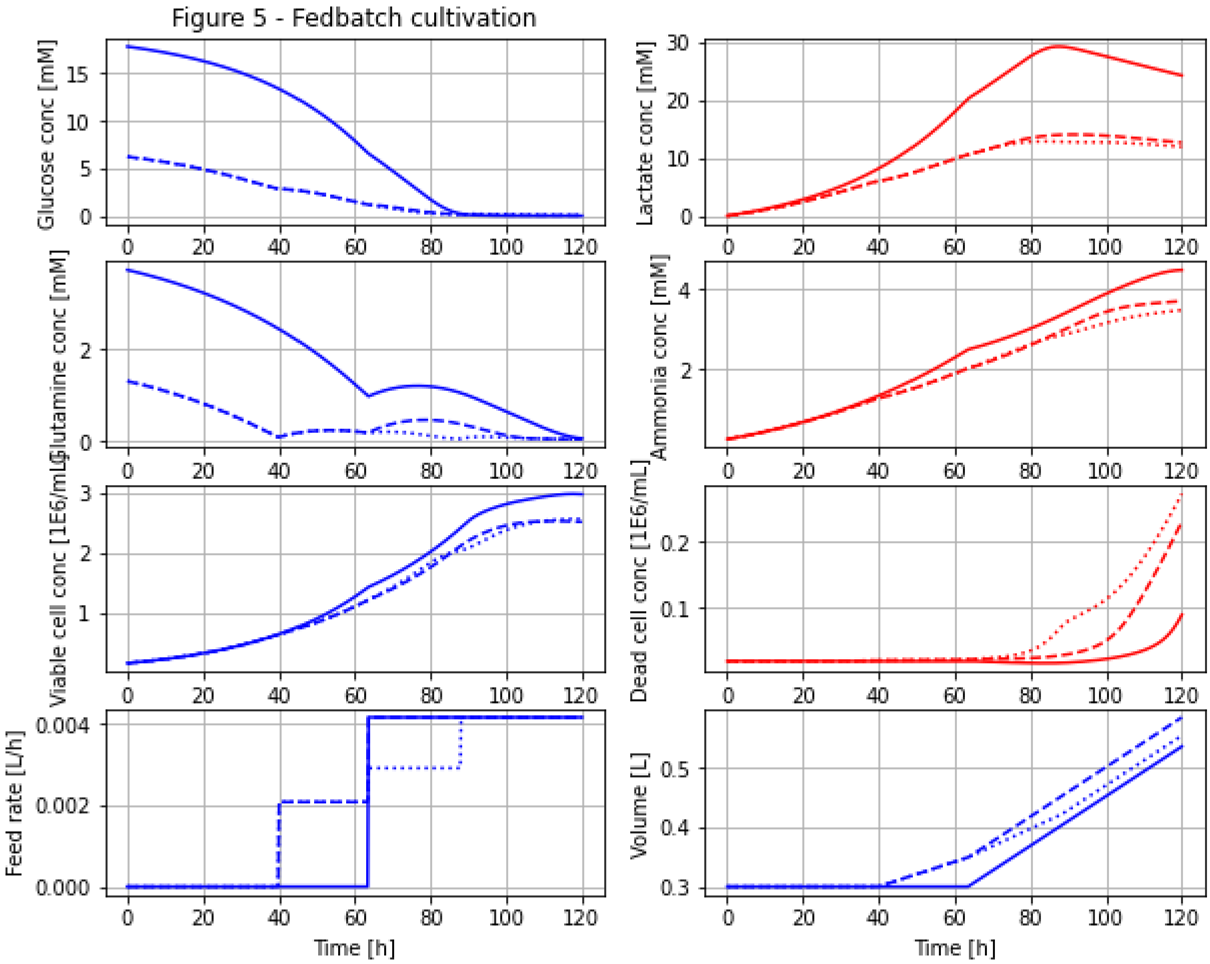
Comment: The simulation results looks very similar to the published diagram Figure 5 in [1]. The model pass this quality check.

3 Simulation of different start-up feeding strategies

```
# Figur 5
V_0=0.30
init(V_0=V_0, VXv_0=V_0*0.172, VXd_0=V_0*0.020)
init(VG_0=V_0*17.83, VGn_0=V_0*3.74, VL_0=V_0*0.12, VN_0=V_0*0.24)

# Feeding
Feed=0.1/24
par(G_in=15, Gn_in=9.3)
par(t0=0, F0=0, t1=63.5, F1=Feed, t2=300, F2=Feed)

newplot(title='Figure 5 - Fedbatch cultivation')
simu(120)

init(VG_0=0.35*V_0*17.83, VGn_0=0.35*V_0*3.74)
par(t0=0, F0=0, t1=40.0, F1=0.5*Feed, t2=63.5, F2=Feed,t3=300, F3=Feed)
simu(120)

init(VG_0=0.35*V_0*17.83, VGn_0=0.35*V_0*3.74)
```

```
par(t0=0, F0=0, t1=40.0, F1=0.5*Feed, t2=63.5, F2=0.7*Feed, t3=88.0, F3=Feed, t4=30
simu(120)

# Reset time table to avoid problems below
par(t1=1001, t2=1002, t3=1003, t4=1004, t5=1005, t6=1006)
```

```
Simulation interval    : 0.0 - 119.99999999999999 seconds.
Elapsed simulation time: 0.013860082000064722 seconds.
Simulation interval    : 0.0 - 119.99999999999999 seconds.
Elapsed simulation time: 0.013566843000035078 seconds.
Simulation interval    : 0.0 - 119.99999999999999 seconds.
Elapsed simulation time: 0.014958736000039607 seconds.
```



Comment: We see that starting the feed a day erlier at lower rate and then increase decreaes lactate formation to half, while the cell conentation is just slightlly lower. With a more careful design of the feedprofile the ammonia formation can be decreased more than shown here.

4 Simulation of optimal feed profile for cell growth

At the end of the original paper section 5 in [1], the derived model is used to find an optimal feeding profile for high final cell concentration. It is stated that protein productivity is assumed to be mainly positively growth associated and therefore optimization of cell concentration is very similar to optimization of protein product. The optimization of feed profile is done with diffrent structures of the feed profile. All of them have a start-time and all of them has a fixed amount of substrate and concentrations in the media are also the same.

- The first optimzation is for a feed profile similar to the experimental, i.e. after start the feed rate remains constant throughout the cultivation. Thus the start time and the actual feed rate are optimized. The result was that the start time was about the same as expermentally but the feed rate was 50% higher, see Figure 7 and Figure 10 in [1].
- The second optimization is for a feed profile with not just one increase but three steps of increase of feed rate. The results is a somwehat higher final cell concentration, see Figure 11.
- The third optimization is for a feed profile with five steps of increase of feed rate. The results is a slightly higher final cell concentration than for three steps, see Figure 12.
- The fourth optimization is for a feed profie with continuous exponential increase of the reed rate. The result is a bit higher final concentration than the previous with five steps, see Figure 13 but not shown in the figure below.

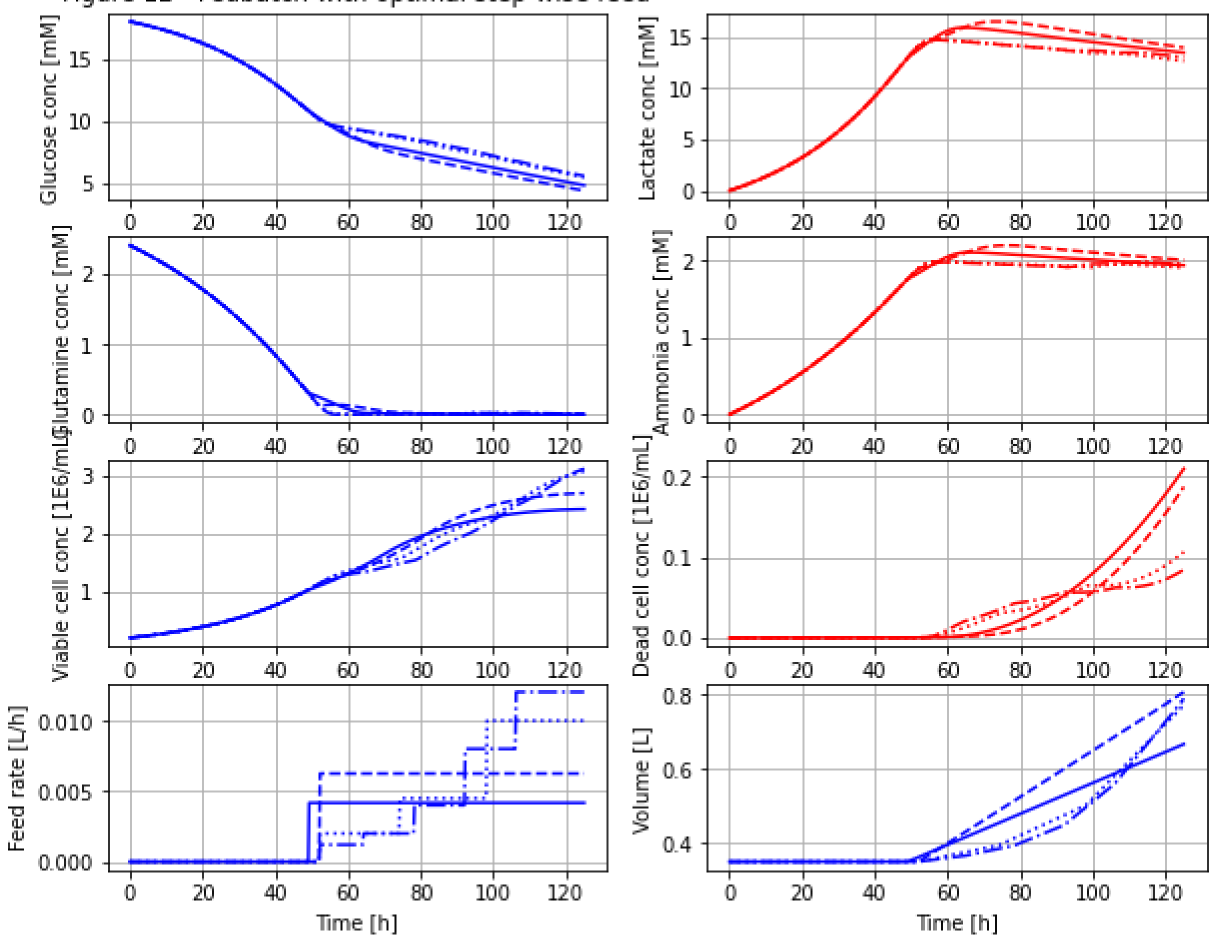
Below we just show the results of the original experimetnal cultivation, compared with results from three and five steps. It is possible to do the optimization in Python with the FMU, but we

```
# Culture parameters taken from Table 5 identified parameters for cultures 1,2,and

# Data chosen
V0=0.35
init(V_0=V0, VXv_0=V0*0.20, VXd_0=V0*0.0)
init(VG_0=V0*18.0, VGn_0=V0*2.4, VL_0=V0*0, VN_0=V0*0)

# Feeding n=1 - experimental and lower feed rate
par(G_in=15, Gn_in=4.0)
par(t0=0, F0=0, t1=49, F1=0.00417)
par(t2=1002, t3=1003, t4=1004, t5=1005)

# Simulation
newplot(title='Figure 12 - Fedbatch with optimal step-wise feed')
simu(125)

# Feeding n=1
par(G_in=15, Gn_in=4.0)
par(t0=0, F0=0, t1=52, F1=0.00625)
par(t2=1002, t3=1003, t4=1004, t5=1005, t6=1006)

# Simulation
simu(125)

# Feeding n=3
par(G_in=15, Gn_in=4.0)
par(t0=0, F0=0, t1=52, F1=0.002, t2=74, F2=0.0045, t3=98.0, F3=0.010)
par(t4=99.0, F4=0.010, t5=106, F5=0.010, t6=150, F6=0.010)

# Simulation
```

```
simu(125)

# Feeding n=5
par(G_in=15, Gn_in=4.0)
par(t0=0, F0=0, t1=52, F1=0.0012, t2=64, F2=0.0020, t3=78.0, F3=0.0040)
par(t4=92.0, F4=0.0080, t5=106, F5=0.012, t6=150, F6=0.012)

# Simulation
simu(125)

# Reset feeding parameters since the table need time in strict increasing value
par(t3=1004, t4=1005, t5=1005, t6=1006)
```

```
Simulation interval    : 0.0 - 125.0 seconds.
Elapsed simulation time: 0.013602624999975887 seconds.
Simulation interval    : 0.0 - 125.0 seconds.
Elapsed simulation time: 0.014723095000022113 seconds.
Simulation interval    : 0.0 - 125.0 seconds.
Elapsed simulation time: 0.015500108999958684 seconds.
Simulation interval    : 0.0 - 125.0 seconds.
Elapsed simulation time: 0.018435351999998314 seconds.
```

Figure 12 - Fedbatch with optimal step-wise feed



Comment: We see that that already the better tuned constant feed rate (dahsed) compared to the experimental (solid) gives higher final cell concnetration.

Breaking up the constant feed rate in three (dotted) and five (dash-dotted) steps with a more gradual increase of the feed rate gives even higher final cell concentration. The difference between n=3 and n=5 is small. The change to continuous exponential feed is even smaller and not shown here.

The results shown here are similar to what is presented in Table 7 in [1] but our simulation are slightly longer and here are small differences in the final cel concentration too. The qualitative result is the same though. The difference we see to the result in the original paper is most likely due to the fact that we here use the full model with 17 parameters while in the paper they have

5 Simulation of different feed profiles to increase recombinant protein production

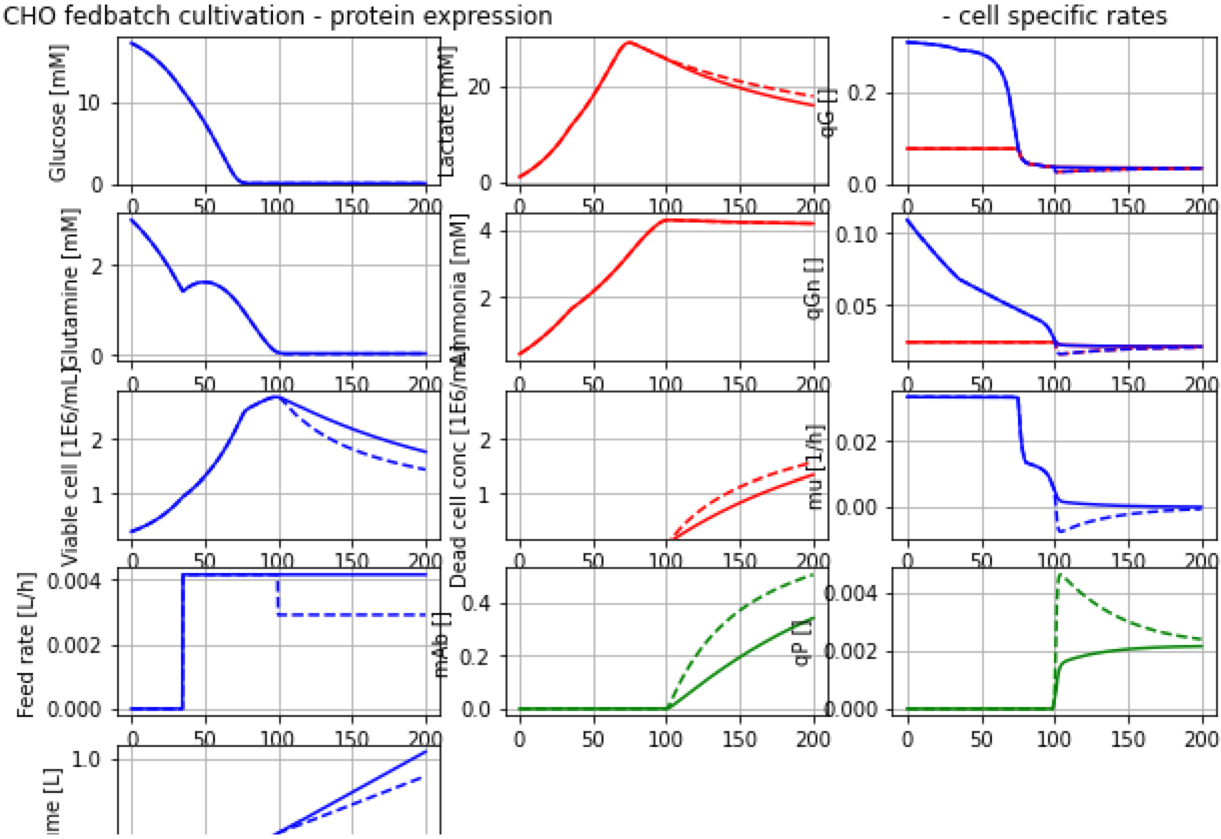
In this section we take a closer look at recombinant protein production. The original model is extended with the empirical model for specific protein production, see chapter 5 in [4]

$$q_P = \alpha \cdot \mu + \beta$$

Here we choose a negative value of growth-associated protein production production α while keeping the non-growth associated β positive. The culture produced recombinant protein in the form of monoclonal antibodies for a specific IgG1 molecule,see section 2 in [1]. However, no experimental results were given. The only information we have is that feed rate was kept constant at a low level during fedbatch production and this choice indicates that the the growth-associated protein production is negative. The consequence of this observation for the feed profile we take a look at there by simulation.

```
# Slide 3
newplot('CHO fedbatch cultivation - protein expression', plotType='Textbook_3')

# Data from Table 1 and 2 for experiment 3
V_0=0.35
init(V_0=V_0, VXv_0=V_0*0.29, VXd_0=V_0*0.010)
init(VG_0=V_0*17.17, VGn_0=V_0*3.02, VL_0=V_0*1.12, VN_0=V_0*0.29)

# Feeding
Feed=0.1/24
par(G_in=15, Gn_in=9.3)
par(t0=0, F0=0, t1=35, F1=Feed, t2=100, F2=Feed, t3=300, F3=Feed)

# Culture parameters
par(alpha=-1.0, beta=0.01)

# Simulation
simu(200)
par(t2=100, F2=0.7*Feed, t3=300, F3=0.7*Feed); simu(200)
par(F2=Feed, F3=Feed)
```

```
Simulation interval    : 0.0 - 200.00000000000003 seconds.
Elapsed simulation time: 0.015894877999926393 seconds.
Simulation interval    : 0.0 - 200.00000000000003 seconds.
Elapsed simulation time: 0.014797330999954283 seconds.
```



Comment: The simulation results show that actually a decease in the feed rate can lead to an increase in recombinant protein produced, although the cell concentration is a bit lower. This is a result due to the fact that growth-associated protein production here is set to a negative value. The main point is that the model can actually capture this phenomena.

6 Summary

In short we have done the following:

- The model was checked by comparing the simulation results with one of the published diagrams [1].
- The common startup-procedure with 3 days batch cultivation can be questioned. We found that by shorten it to 2 days, and giving smaller feed rate day 3, byproduct formation can be kept lower at the prize of just a bit lower cell concentration. Similar idea was shown in section 2.1 in [3].
- In the original paper the experimental feeding strategy was to keep the substrate feed at a constant lower level. The authors made a point of that the optimal feeding strategy should be exponential for maximal cell production. This is an insight derived from the bottle-neck model and they showed that through simulation optimization [1]. However, there was no experimental support to confirm the results. The optimal cell growth feedprofile simulation was just reproduced here.

- To optimize recombinant protein production we must include production in the model. Here we do that with the empirical model that distinguish between growth-associated and non-growth-associated protein production, see chapter 5 in [4]. For a class of CHO-processes the recombinant protein productivity is acutally negatively affected by cell growth. Simulation of the original model extended with such a protein production model shows that keeping the substrate feed rate constant as the cell culture grows, giving less and less feed per cell, actually can give higher protein production than an increaeing feed rate. Simulation confirms this idea. The results gives some possible background to why the constant feed rate was used experimetnally in the original paper [1].

7 References

[1] Amribt, Z., Niu, H. and Bogaerts P.: "Macroscopic modelling of overflow metabolism and model based optimization of hybridoma cell fed-batch cultures.", Biochem. Eng. Journal, 2013.

[2] Niu,H., Amribt, Z., Fickers, P., Tan, W. and Bogaerts P.: "Metabolic pathway analysis and reduction for mammalian cell cultures - towards macroscopic modelling", Chem. Eng. Science, 2013.

[3] Axelsson, J. P.: "Simplified model of CHO-cultivation in Bioproces Library for Modelica - some experience", conference paper 22nd NPCW Lyngby, Denmark, August 22-23, 2019.

[4] Hu, W-S: "Cell culture bioprocess engineering", 2nd edition, CRC Press, 2020.

Appendix

```
disp('culture')
```

```
qG_max1 : 0.297
qG_max2 : 0.038
qGn_max1 : 0.124
qGn_max2 : 0.022
mu_d_max : 0.13
alpha : -1.0
beta : 0.01
```

```
describe('mu')
```

```
Specific cell growth rate variable : 0.008 [ 1/h ]
```

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

```
['bioreactor', 'bioreactor.broth_decay', 'bioreactor.culture', 'dosagescheme',
```

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

```
MSL: 3.2.2 build 3 - used components: RealInput, RealOutput, CombiTimeTable, T
```

```
system_info()
```

```
System information
 -OS: Linux
 -Python: 3.7.14
 -Scipy: not installed in the notebook
 -PyFMI: 2.7.4
 -FMU by: JModelica.org
 -FMI: 2.0
 -Type: FMUModelCS2
 -Name: BPL_CHO.Fedbatch
 -Generated: 2022-10-17T07:45:26
 -MSL: 3.2.2 build 3
 -Description: Bioprocess Library version 2.1.0
 -Interaction: FMU-explore ver 0.9.5
```