# BPL_YEAST_AIR_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_YEAST_AIR_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/')
```

```
fmilib

Package charset-normalizer conflicts for:
charset-normalizer==2.0.4=pyhd3eb1b0_0
conda==4.12.0=py37h06a4308_0 -> requests[version='>=2.18.4,<3'] -> charset-nor
requests==2.27.1=pyhd3eb1b0_0 -> charset-normalizer[version='>=2.0.0,<2.1.0']

Package gmp conflicts for:
mpfr -> gmp[version='>=6.2.1,<7.0a0']
gmp
suitesparse -> mpfr[version='>=4.1.0,<5.0a0'] -> gmp[version='>=6.2.1,<7.0a0']

Package six conflicts for:
conda-content-trust==0.1.1=pyhd3eb1b0_0 -> six
six==1.16.0=pyhd3eb1b0_1

Package ruamel_yaml conflicts for:
ruamel_yaml==0.15.100=py37h27cfd23_0
conda==4.12.0=py37h06a4308_0 -> ruamel_yaml[version='>=0.11.14,<0.17']

Package wheel conflicts for:
wheel==0.37.1=pyhd3eb1b0_0
pip==21.2.2=py37h06a4308_0 -> wheel

Package lxml conflicts for:
pyfmi -> lxml
lxml
```

```
Package assimulo conflicts for:
assimulo
pyfmi -> assimulo[version='>=3.0']The following specifications were found to b

  - feature:/linux-64::__glibc==2.27=0
  - feature:|@/linux-64::__glibc==2.27=0
  - brotlipy==0.7.0=py37h27cfd23_1003 -> libgcc-ng[version='>=7.3.0'] -> __gli
  - cffi==1.15.0=py37hd667e15_1 -> libgcc-ng[version='>=7.5.0'] -> __glibc[ver
  - conda-package-handling==1.8.1=py37h7f8727e_0 -> libgcc-ng[version='>=7.5.0
  - cryptography==36.0.0=py37h9ce1e76_0 -> libgcc-ng -> __glibc[version='>=2.1
  - gmp -> libgcc-ng[version='>=7.5.0'] -> __glibc[version='>=2.17']
  - libffi==3.3=he6710b0_2 -> libgcc-ng[version='>=7.3.0'] -> __glibc[version=
  - libgcc-ng==9.3.0=h5101ec6_17 -> __glibc[version='>=2.17']
  - libstdcxx-ng==9.3.0=hd4cf53a_17 -> __glibc[version='>=2.17']
  - libxml2 -> libgcc-ng[version='>=9.3.0'] -> __glibc[version='>=2.17']
  - libxslt -> libgcc-ng[version='>=9.3.0'] -> __glibc[version='>=2.17']
  - metis -> libgcc-ng[version='>=7.5.0'] -> __glibc[version='>=2.17']
  - mpfr -> libgcc-ng[version='>=7.5.0'] -> __glibc[version='>=2.17']
  - ncurses==6.3=h7f8727e_2 -> libgcc-ng[version='>=7.5.0'] -> __glibc[version
  - numpy -> libgcc-ng[version='>=7.3.0'] -> __glibc[version='>=2.17']
  - numpy-base -> libgcc-ng[version='>=7.3.0'] -> __glibc[version='>=2.17']
  - openssl==1.1.1n=h7f8727e_0 -> libgcc-ng[version='>=7.5.0'] -> __glibc[vers
  - pycosat==0.6.3=py37h27cfd23_0 -> libgcc-ng[version='>=7.3.0'] -> __glibc[v
  - pyfmi -> libgcc-ng[version='>=7.5.0'] -> __glibc[version='>=2.17']
  - python==3.7.13=h12debd9_0 -> libgcc-ng[version='>=7.5.0'] -> __glibc[versi
  - readline==8.1.2=h7f8727e_1 -> libgcc-ng[version='>=7.5.0'] -> __glibc[vers
  - ruamel_yaml==0.15.100=py37h27cfd23_0 -> libgcc-ng[version='>=7.3.0'] -> __
  - sqlite==3.38.2=hc218d9a_0 -> libgcc-ng[version='>=7.5.0'] -> __glibc[versi
  - tk==8.6.11=h1ccaba5_0 -> libgcc-ng[version='>=7.5.0'] -> __glibc[version='
  - xz==5.2.5=h7b6447c_0 -> libgcc-ng[version='>=7.3.0'] -> __glibc[version='>
  - yaml==0.2.5=h7b6447c_0 -> libgcc-ng[version='>=7.3.0'] -> __glibc[version=
```

```
!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-2.116           | 13 KB     | : 100% 1.0/1 [00:00<00:00,  8.81it/s]
blas-devel-3.9.0     | 12 KB     | : 100% 1.0/1 [00:00<00:00, 26.18it/s]
libopenblas-0.3.21   | 10.1 MB   | : 100% 1.0/1 [00:02<00:00,  2.34s/it]
libgomp-12.2.0       | 455 KB    | : 100% 1.0/1 [00:00<00:00,  8.82it/s]
libcblas-3.9.0       | 13 KB     | : 100% 1.0/1 [00:00<00:00, 23.39it/s]
libgcc-ng-12.2.0     | 936 KB    | : 100% 1.0/1 [00:00<00:00,  4.49it/s]
llvm-openmp-14.0.6   | 4.4 MB    | : 100% 1.0/1 [00:00<00:00,  2.92it/s]

openblas-0.3.21      | 10.8 MB   | : 100% 1.0/1 [00:03<00:00,  3.04s/it]
liblapacke-3.9.0     | 13 KB     | : 100% 1.0/1 [00:00<00:00, 27.60it/s]
_openmp_mutex-4.5    | 6 KB      | : 100% 1.0/1 [00:00<00:00, 24.08it/s]
openssl-1.1.1q       | 2.1 MB    | : 100% 1.0/1 [00:00<00:00,  2.24it/s]
libblas-3.9.0        | 13 KB     | : 100% 1.0/1 [00:00<00:00, 25.94it/s]
liblapack-3.9.0      | 13 KB     | : 100% 1.0/1 [00:00<00:00, 26.65it/s]
_libgcc_mutex-0.1    | 3 KB      | : 100% 1.0/1 [00:00<00:00, 28.86it/s]
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
```

## Notes of BPL_YEAST_AIR_Fedbatch

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

- FMU - BPL_YEAST_AIR_Fedbatch_linux_jm_cs.fmu
- Setup-file - BPL_YEAST_AIR_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_YEAST_AIR_Fedbatch
```

```
    Cloning into 'BPL_YEAST_AIR_Fedbatch'...
```

```
%cd BPL_YEAST_AIR_Fedbatch
```

```
    /content/BPL_YEAST_AIR_Fedbatch/BPL_YEAST_AIR_Fedbatch/BPL_YEAST_AIR_Fedbatch
```

## BPL_YEAST_AIR_Fedbatch - demo

This notebook demonstrate yeast fedbatch cultivation. We look at impact of changes in the glucose feeding. We also take a look at tuning of the DO-control system. Both liquid- and gasphase are included in the model. The culture growth and metabolism are formulated in relation to to the respiratory capacity [1] and the model is exapanded to describe also the gas phase as well as the culture heat production [2]. The model was derived mainly from continuous culture data but proved to capture dynamic aspects well of ethanol production and consumption [3].

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 is to demonstrate how simulations and varyiing conditions can provide some process insight that can support the experimetnal 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 ctrol-c and ctrl-p 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.

```
run -i BPL_YEAST_AIR_Fedbatch_DOcontrol_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 simulatic
 - 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()
```

```
%matplotlib inline
plt.rcParams['figure.figsize'] = [36/2.54, 30/2.54]
```

## About the process model

We can get information about the process, liquid- and gas-phase by the command describe(). 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.

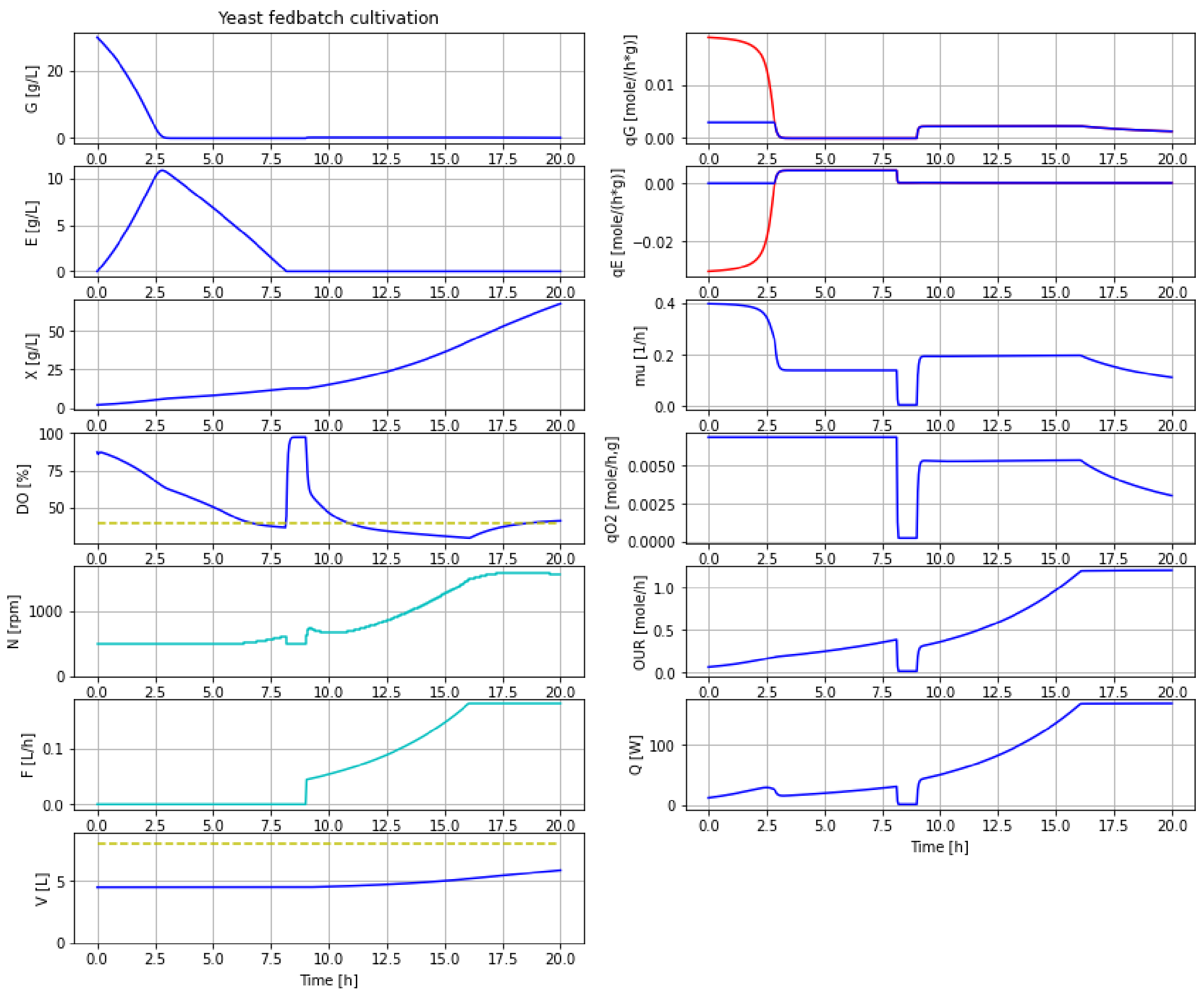
```
describe('culture'); print(); describe('liquidphase'); print(); describe('gasphase'
```

```
    Saccharomyces cerevisae - default parameters for strain H1022

    Reactor broth substances included in the model

    Cells   index       =  1 - molecular weight =  24.6 Da
    Glucose index       =  2 - molecular weight =  180.0 Da
    Ethanol index       =  3 - molecular weight =  46.0 Da
    Dissolved O2 index  =  4 - molecular weight =  32.0 Da
    Dissolved CO2 index =  5 - molecular weight =  44.0 Da

    Reactor gasphase substances included in the model

    N2 etc index  =  1 - molecular weight =  28.0 Da
    O2 index      =  2 - molecular weight =  32.0 Da
    CO2 index     =  3 - molecular weight =  44.0 Da
    Ethanol index =  4 - molecular weight =  46.0 Da
```

The model of the process has parameters both for culture, gas_liquid_transfer, as well as feeding procedure. The paramters that are available for changes you find by the command disp() and you get a long list and you change by them by command par(). The model has even more parameters in the background but not made available for interaction.

## First simulations - adjusting start of substrate feeding

```
# Culture parameters and others at default values
par(qO2lim=0.0069)

# Process initial conditions
init(V_0=4.5, VG_0=4.5*30, VX_0=4.5*2, VE_0=4.5*0)

# Feed profile
par(t_start=9, F_start=0.044, mu_feed=0.20, F_max=0.18)

# DO-control parameters
par(samplePeriod=1/60, K=10, Ti=0.5, I_0=500)

# Simulate and plot
newplot(title='Yeast fedbatch cultivation', plotType='Overview')
simu(20)
```



Now we can get value of broth volume as well as the headspace and values are the last ones in the simulation

```
describe('bioreactor.V')
```

```
    Reactor broth volume : 5.892 [ L ]
```

```
describe('bioreactor.V_gasphase')
```

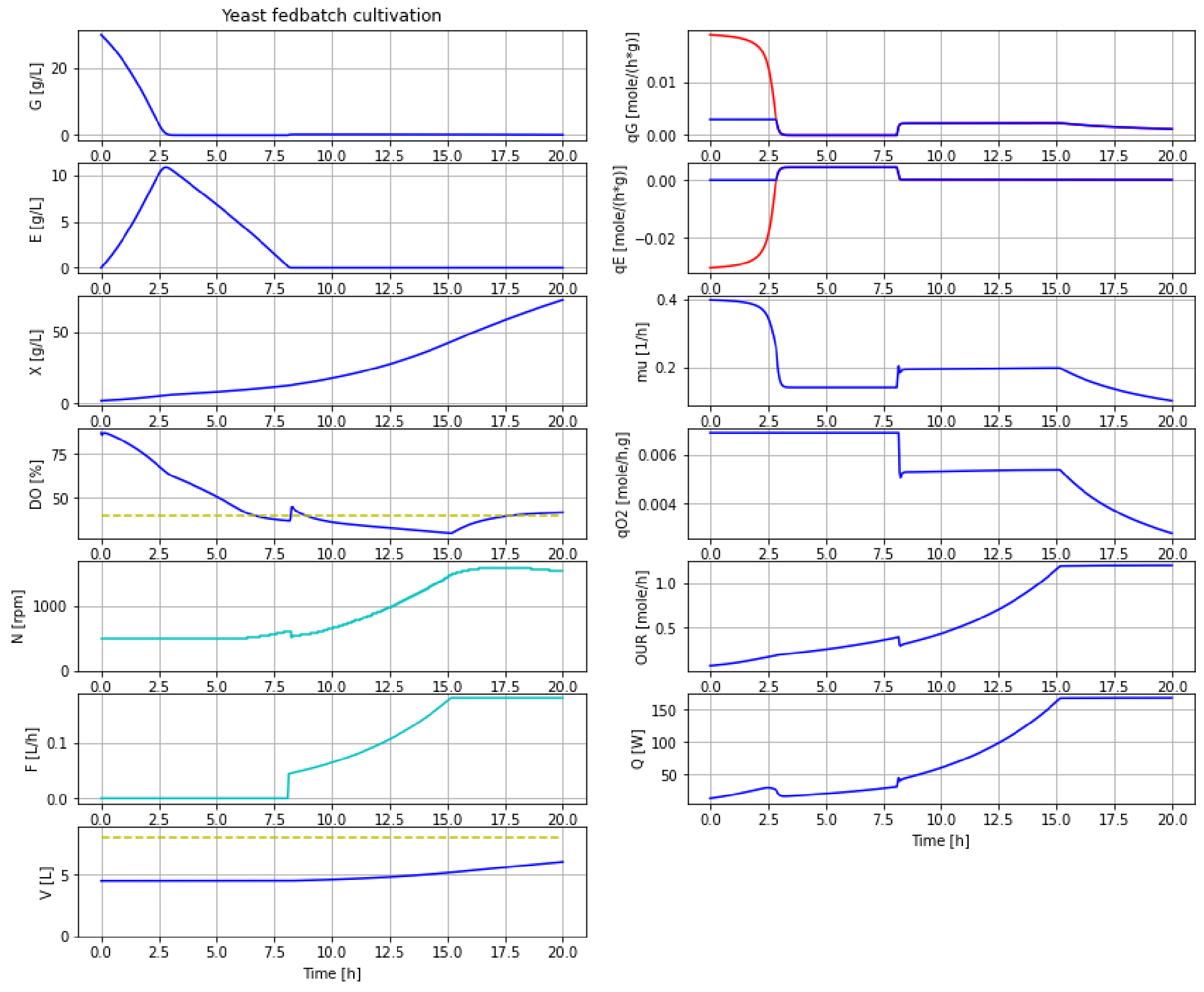
```
    Volume of the gas phase : 2.108 [ L ]
```

```
# Take a look at the parameters available to adjust the dosage scheme
disp('dosage', decimals=4)
```

```
    mu_feed : 0.2
    F_0 : 0.0
    t_start : 9.0
    F_start : 0.044
    F_max : 0.18
```

```
# Let us start the feeding just after the batch phase has ended and keep other para
```

```
par(t_start=8.1)

# Simulate and plot
newplot(title='Yeast fedbatch cultivation', plotType='Overview')
simu(20)
```

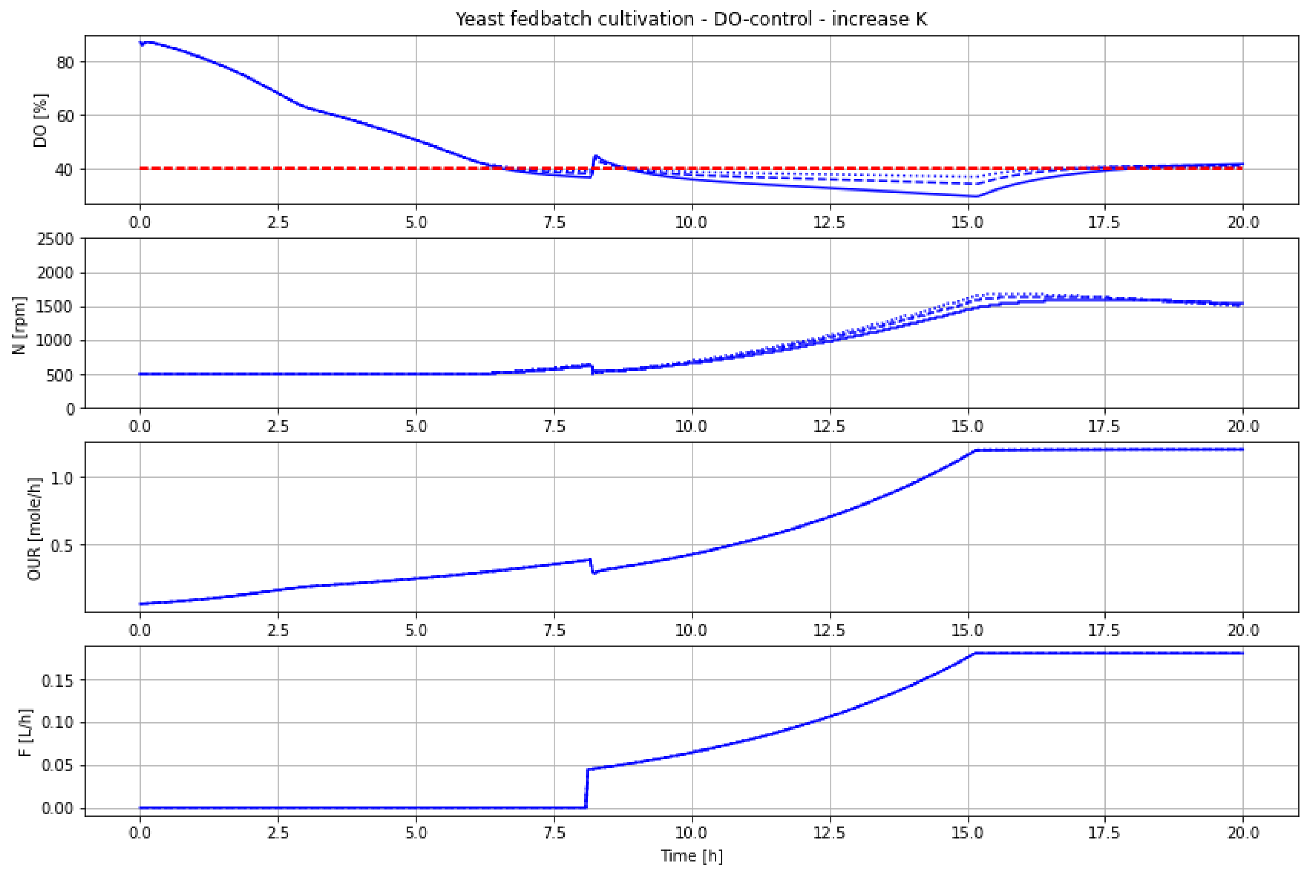


The increase of DO to about 50 % at end of batch phase should be possible to detect easily. This simulation is more realistic and we use these settings from now on.

## DO-control - tuning of PI-regulator parameters

Let us focus on the DO-control system and choose a more limited plotType. We study the impact of PI control parameters and see if we can decrease the control error without looing stability.

```
# Let us take a closer look at the DO-control system and try to make control error
newplot(title='Yeast fedbatch cultivation - DO-control - increase K', plotType='Foc
for value in [10, 20, 40]: par(K=value); simu(20)

# Reset K to the original value
par(K=10)
```



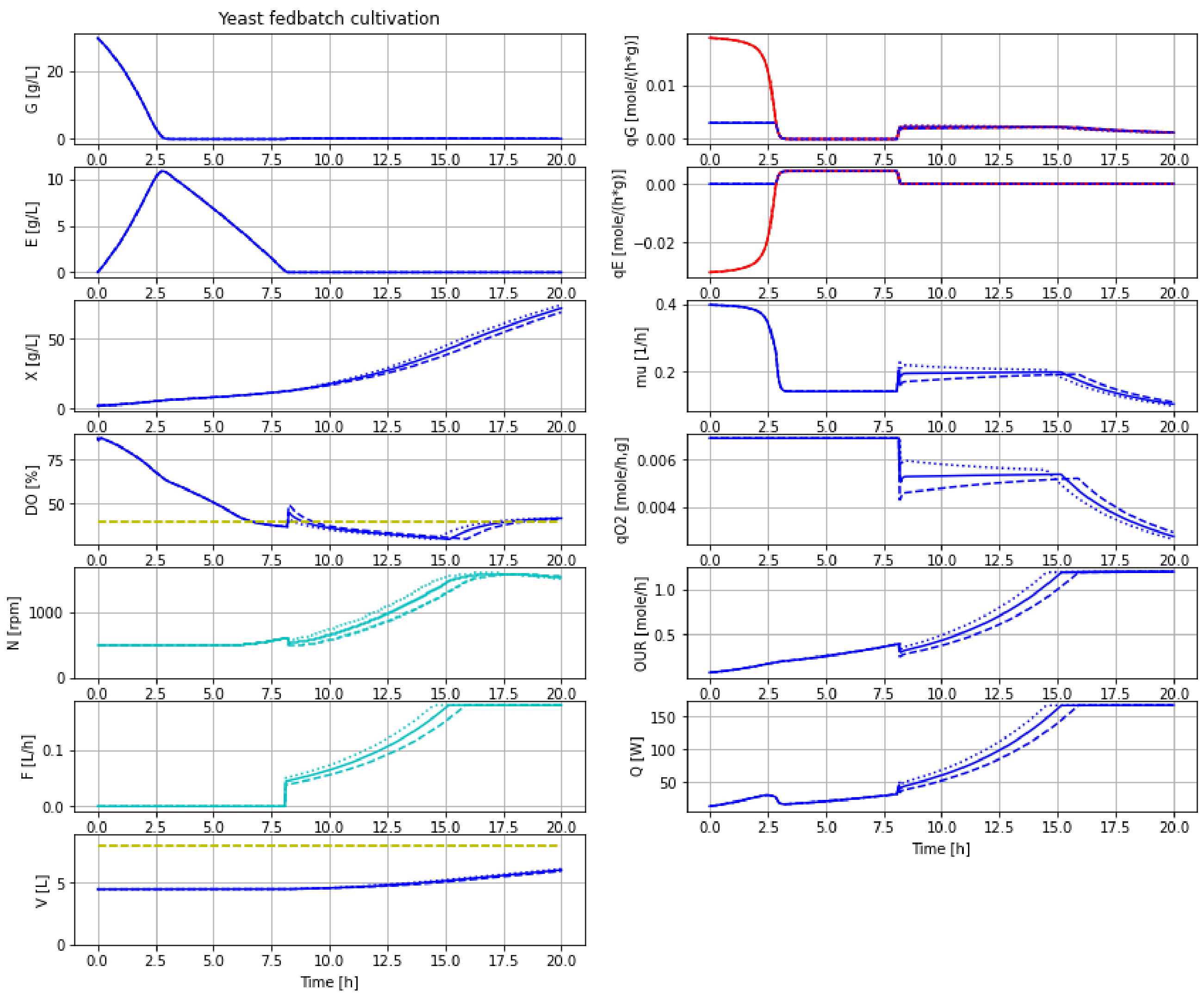
We see that by a higher control gain K the DO-control error get smaller and the stability of the control system is maintained.

**Exercise** I leave for you to study the impact variation of the Ti-parameter. Just make a new cell below. Then copy and paste the cell above and change parameter to Ti.

## Sensitivity to changes in feed-profile

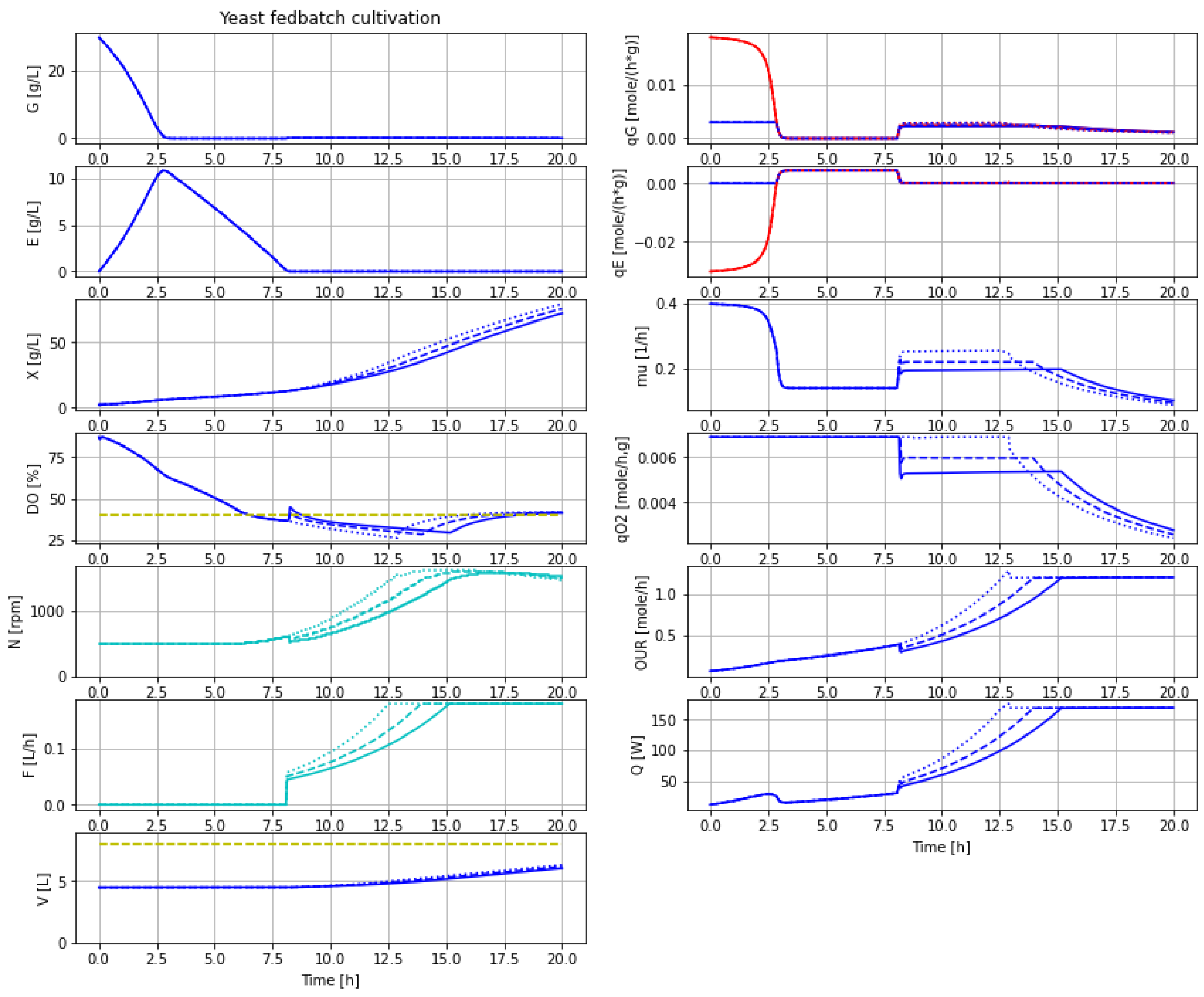
Now, let us focus on investigating impact of changes in the feed-profile. The goal is to increase the produced cell mass without accumulation of by-product ethanol. Simulation can bring some insight into how behaviour of the differen variables change when by-product is formed. This

```
# Let us check the sensitivity to changes in the feed profile design
newplot(title='Yeast fedbatch cultivation', plotType='Overview')
for value in [0.044, 0.038, 0.050]: par(F_start=value); simu(20)
```

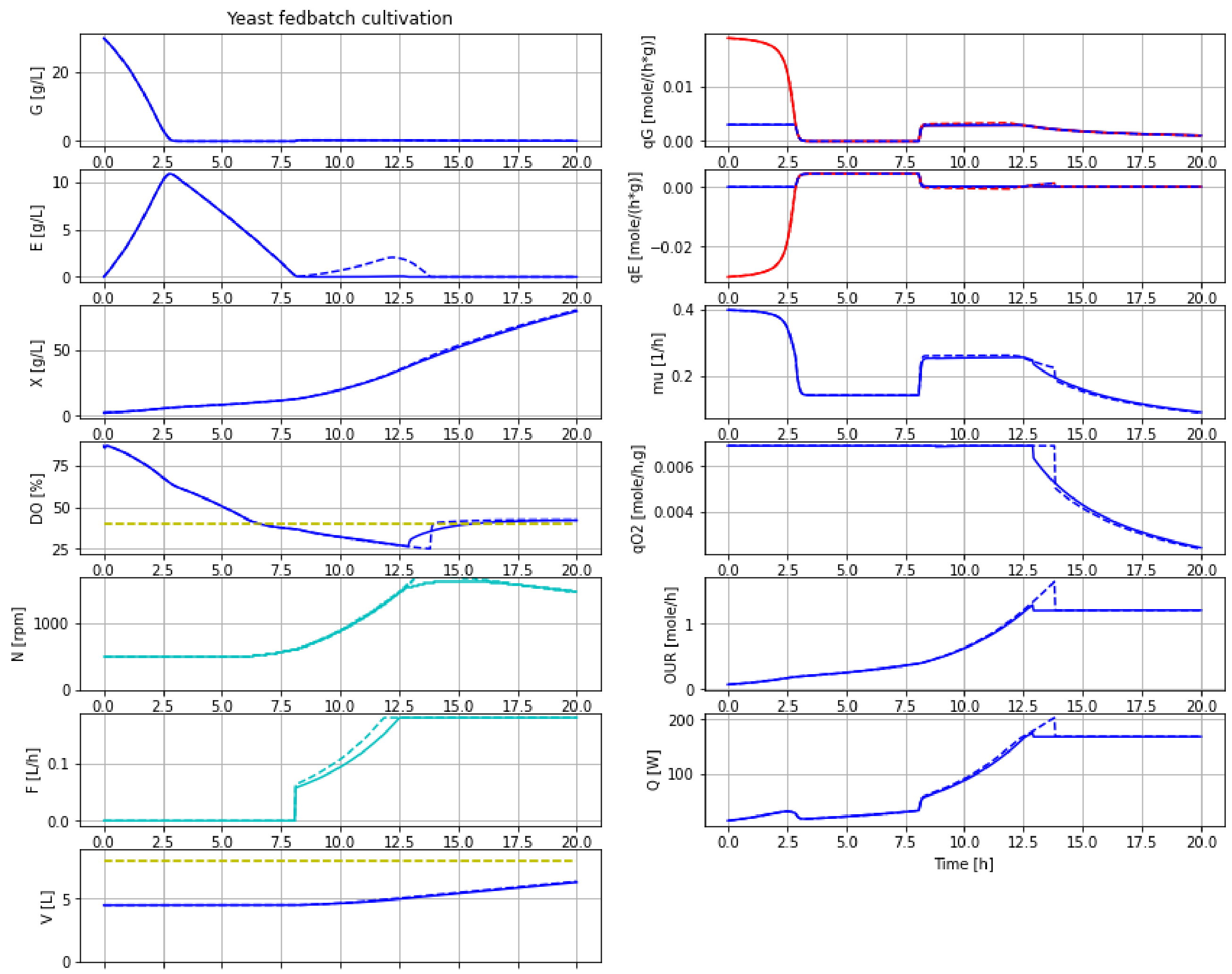


The variation in F_start has an impact and we see that the actual growth rate during fedbatch phase do converge to the set growth rate of the feed, but i takes more than 5 hours.

```
# Let us investigate a feedprofile that is closer to the maximal capacity
newplot(title='Yeast fedbatch cultivation', plotType='Overview')
par(F_start=0.044, mu_feed=0.20); simu(20)
par(F_start=0.050, mu_feed=0.22); simu(20)
par(F_start=0.057, mu_feed=0.26); simu(20)
```



```
# And let us see what happens if the feedprofile exceed the culture capacity
newplot(title='Yeast fedbatch cultivation', plotType='Overview')
par(F_start=0.057, mu_feed=0.26); simu(20)
par(F_start=0.063, mu_feed=0.28); simu(20)
par(F_start=0.044, mu_feed=0.20)
```



Note that with the feedprofile that exceed culture respiratory capacity, ethanol is accumulated during time 8-12.5 hours. When the feedprofile then is constant from time 12.5 hours and on, then the accumulated ethanol is consumed over about an hour. This leads to a higher oxygen demand and heat production during this time. The specific cell growth rate is also slightly higher during this period.

**Exercise** You can investiate the impact of changing the maximal feedrate F_max. Make sure that the DO level do not get too low.

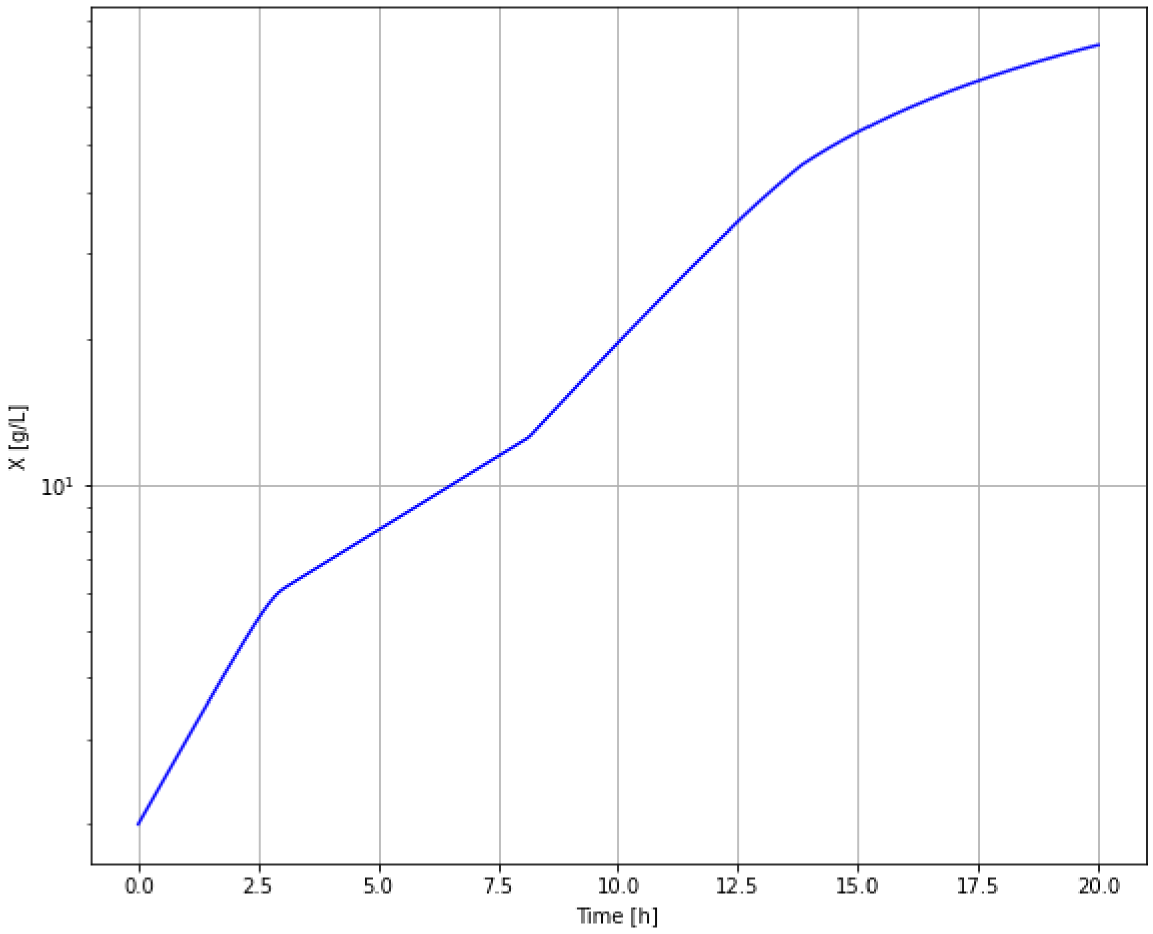
## Make your own diagrams

There are a couple of pre-defined plotType for the application that you make by the command newplot(). The command result in a list "diagrams" that desccrige the commands that make the plot when you call simu() or you just want to look at the last simulation again with a changed plotType using show().

You can also in Jupyter notebook directly define the list "diagrams" and then that will be used for subsequent calls of simu() or show(). When you have made a diagram that you want to reuse many times you can bring it into the python-setup file and edit the newplot() commmand and add a new plotType.

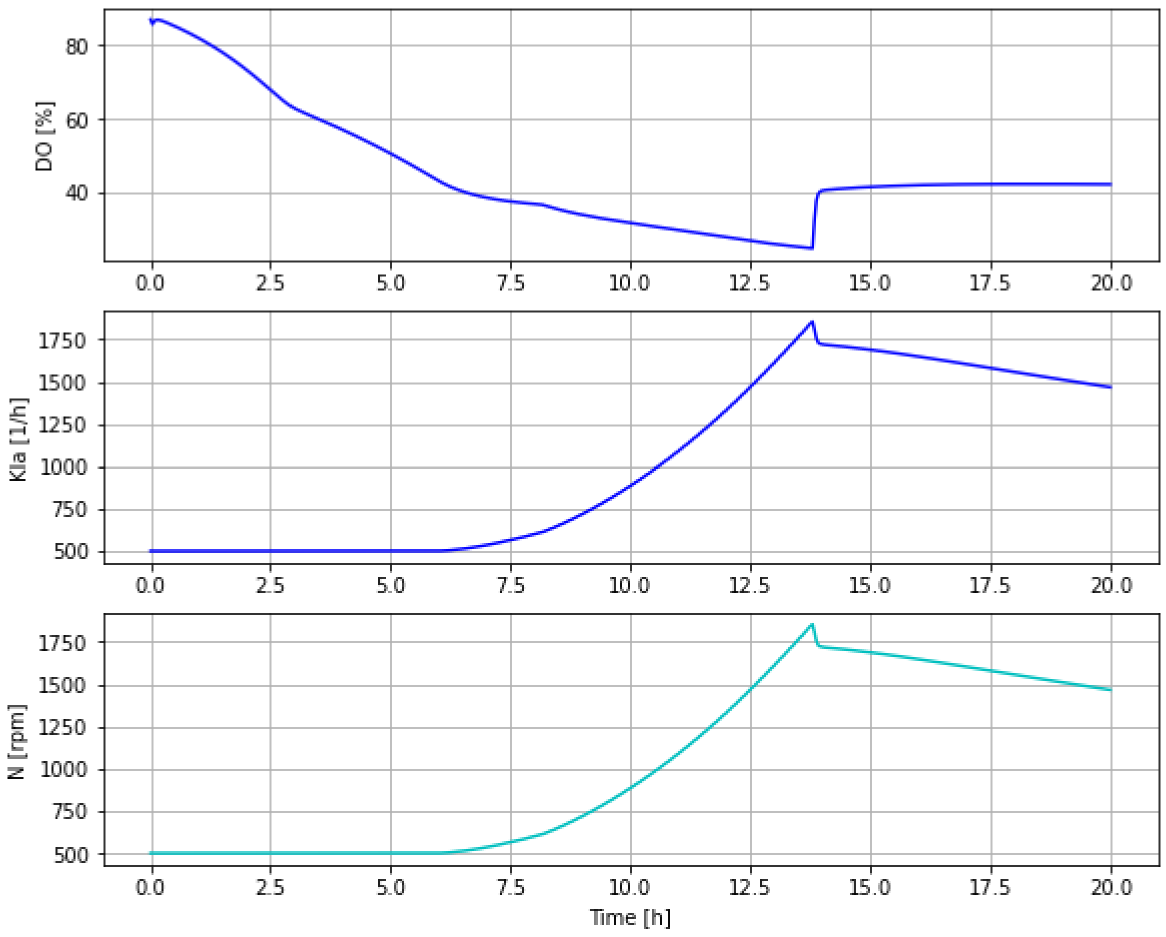
Below a few simple examples that show how to do a diagram directly i the notebook

```
# First decrease the diagram size
plt.rcParams['figure.figsize'] = [24/2.54, 20/2.54]


# Improvise and make your own diagram - cell concentration in a logaritmic plot
plt.figure()
ax1 = plt.subplot(1,1,1)
ax1.set_ylabel('X [g/L]')
ax1.set_xlabel('Time [h]')
ax1.grid()

setLines()
diagrams.clear()
diagrams.append("ax1.semilogy(sim_res['time'], sim_res['bioreactor.c[1]'], color='b
show()
```



```
# - study the variation of Kla together with DO and N during cultivation
plt.figure()
ax1 = plt.subplot(3,1,1); ax2 = plt.subplot(3,1,2); ax3 = plt.subplot(3,1,3)
ax1.set_ylabel('DO [%]'); ax1.grid()
ax2.set_ylabel('Kla [1/h]'); ax2.grid()
ax3.set_ylabel('N [rpm]'); ax3.grid()
ax3.set_xlabel('Time [h]')

setLines()
```

```
diagrams.clear()
diagrams.append("ax1.plot(sim_res['time'], sim_res['DOsensor.out'], color='b', line
diagrams.append("ax2.plot(sim_res['time'], sim_res['bioreactor.gas_liquid_transfer.
diagrams.append("ax3.plot(sim_res['time'], sim_res['bioreactor.N'], color='c', line
show()
```

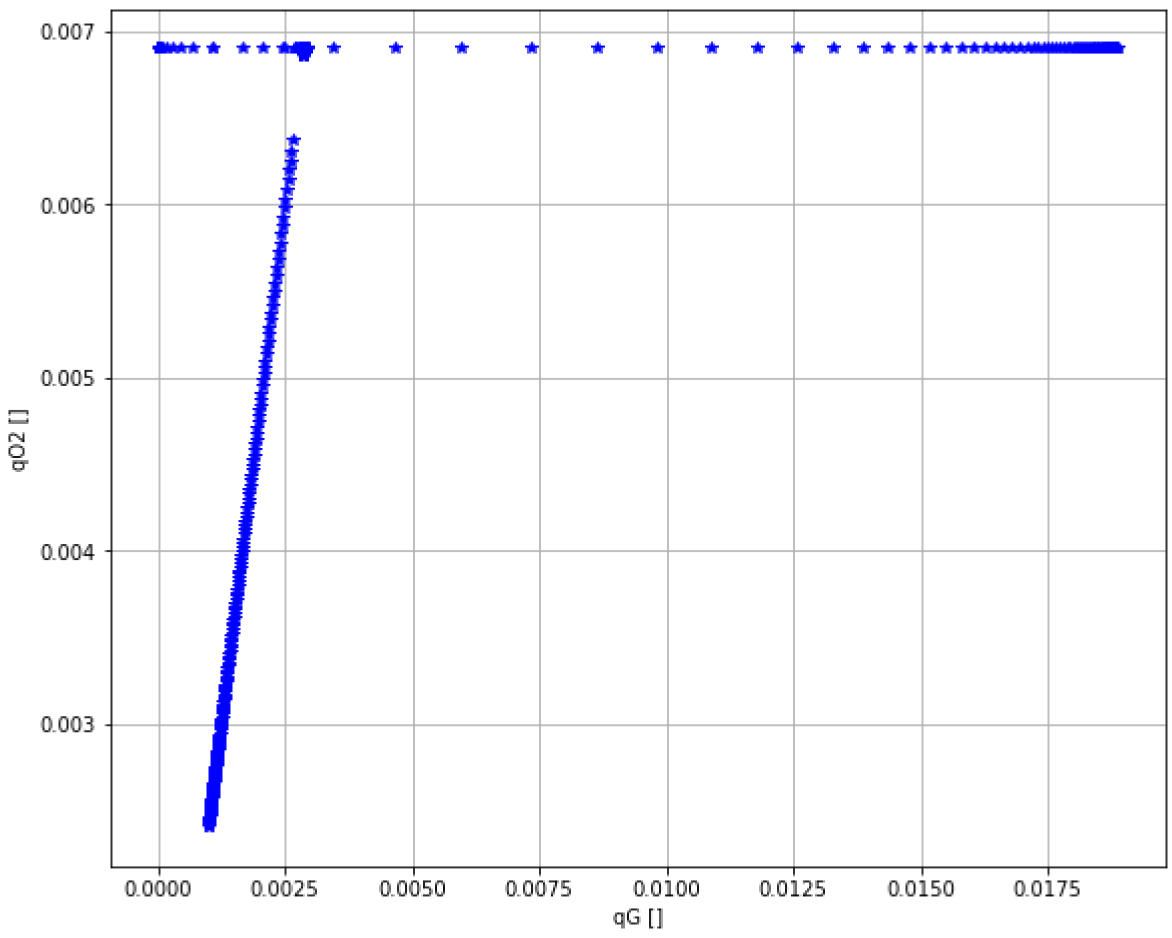


The relation is Kla = alpha_O2*N and we see the value of the parameter should be around 1.0, and we check below

```
disp('bioreactor.gas_liquid_transfer.alpha_O2')
```

```
    alpha_O2 : 1.0
```

```
# - study the relation qO2 vs qG(G)
plt.figure()
ax1 = plt.subplot(1,1,1)
ax1.set_ylabel('qO2 []')
ax1.set_xlabel('qG []')
ax1.grid()

setLines()
diagrams.clear()
diagrams.append("ax1.plot(sim_res['bioreactor.culture.qGm'], sim_res['bioreactor.cu
par(F_start=0.057, mu_feed=0.26)
simu(20)
```



During the cultivation we have a number of data points for qG and qO2 at the same time, during different conditions. What we see in the diagram is that qO2 increase with qG until qG reach a level of just above 0.0025 and then qO2 saturats for highter qG. This what expect to see.

We also see that for lower qG we have also qO2 values at saturation level. This points correspond to a situation where ethanol is consumed with the remaining respiratory capacity. Glucose is consumed by priority.

## Summary

- We have first seen an overview diagram of a typical yeast fedbatch cultivation where the feed started about an hour after the batch phase was finished. A new simulation was made where the feed started directly after detection of lack of substrate.
- We also took a look at the DO-control system and saw that we could decrease the control error by increasing the PI-controller gain. Stability of the control system remained.
- Then we tested variations in the feed dosage scheme and investigated the possibilities to increae the production.
- We also saw what happens if the feed dosage exceed the culture respiratory capacity and what to look for during the experimental work.
- Finally we saw some examples of how to improvise new diagrams.

## References

[1] Sonnleitner, B and O. Käppeli "Growth of *Sacharomyces cerevisiae* is controlled by its limited respiratory capacity: formulationa and verification of a hypothesis", Biotech. Bioeng., 1986.

[2] von Stockar, U., Gustafsson, L., Larsson, C., Marison, I., Tissot, P. and Gnaiger E. "Thermodynamic considerations in constructing energy balances for cellular growth", Biochimica et Biophysics Acta, vol 1183, p 221-240, 1993.

[3] Axelsson, J. P. "Experimental techniques and data analysis to dtermine baker's yeast ethanol dynamics", Anal. Chim. Acta, vol 213, p 151-163, 1988.

## Appendix

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

```
['airFlow_setpoint', 'airtube', 'atmosphere', 'bioreactor', 'bioreactor.cultur
```

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

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

```
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_YEAST_AIR.Fedbatch_DOcontrol
 -Generated: 2022-10-17T07:33:47
 -MSL: 3.2.2 build 3
 -Description: Bioprocess Library version 2.1.0
 -Interaction: FMU-explore ver 0.9.5
```

Colab paid products - Cancel contracts here

✓ 0s completed at 09:37