BPL_TEST2_Batch_calibration script with FMPy

The key library FMPy and scipy and more are installed.

After the installation a small application BPL_TEST2_Batch_calibration 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-py312_24.3.0-0-Linux-x86_64.sh
!chmod +x Miniconda3-py312_24.3.0-0-Linux-x86_64.sh
!bash ./Miniconda3-py312_24.3.0-0-Linux-x86_64.sh -b -f -p /usr/local
import sys
sys.path.append('/usr/local/lib/python3.12/site-packages/')
```

```
--2024-05-16 09:56:26--  https://repo.anaconda.com/miniconda/Miniconda3-py312_24.3.0-0-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: 143351488 (137M) [application/octet-stream]
Saving to: ‘Miniconda3-py312_24.3.0-0-Linux-x86_64.sh’

Miniconda3-py312_24 100%[===================>] 136.71M   152MB/s    in 0.9s

2024-05-16 09:56:27 (152 MB/s) - ‘Miniconda3-py312_24.3.0-0-Linux-x86_64.sh’ saved [143351488/143351488]

PREFIX=/usr/local
Unpacking payload ...

Installing base environment...

Preparing transaction: ...working... done
Executing transaction: ...working... done
installation finished.
```

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

```
Channels:
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - conda


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-24.5.0               |  py312h06a4308_0         1.2 MB
    frozendict-2.4.2           |  py312h06a4308_0          36 KB
    openssl-3.0.13             |       h7f8727e_1         5.2 MB
    ------------------------------------------------------------
                                           Total:         6.5 MB

The following NEW packages will be INSTALLED:

  frozendict         pkgs/main/linux-64::frozendict-2.4.2-py312h06a4308_0

The following packages will be UPDATED:

  conda                              24.3.0-py312h06a4308_0 --> 24.5.0-py312h06a4308_0
```

```
    openssl                                 3.0.13-h7f8727e_0 --> 3.0.13-h7f8727e_1


  Downloading and Extracting Packages:
  openssl-3.0.13       | 5.2 MB    | :   0% 0/1 [00:00<?, ?it/s]
  conda-24.5.0         | 1.2 MB    | :   0% 0/1 [00:00<?, ?it/s]

  frozendict-2.4.2     | 36 KB     | :   0% 0/1 [00:00<?, ?it/s]

  openssl-3.0.13       | 5.2 MB    | :   0% 0.002997347135570501/1 [00:00<01:08, 69.09s/it]

  frozendict-2.4.2     | 36 KB     | : 100% 1.0/1 [00:00<00:00,  2.41it/s]
  openssl-3.0.13       | 5.2 MB    | : 100% 1.0/1 [00:00<00:00,  1.41it/s]
  conda-24.5.0         | 1.2 MB    | : 100% 1.0/1 [00:00<00:00,  1.41it/s]



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

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

```
conda 24.5.0
Python 3.12.2
```

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

```
Channels:
 - conda-forge
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - fmpy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _libgcc_mutex-0.1          |      conda_forge           3 KB  conda-forge
    _openmp_mutex-4.5          |            2_gnu          23 KB  conda-forge
    abseil-cpp-20211102.0      |       h27087fc_1         1.1 MB  conda-forge
    anyio-4.3.0                |     pyhd8ed1ab_0         100 KB  conda-forge
    argon2-cffi-23.1.0         |     pyhd8ed1ab_0          18 KB  conda-forge
    argon2-cffi-bindings-21.2.0|  py312h98912ed_4          34 KB  conda-forge
    arrow-1.3.0                |     pyhd8ed1ab_0          98 KB  conda-forge
    arrow-cpp-14.0.2           |       h374c478_1        11.7 MB
    asttokens-2.4.1            |     pyhd8ed1ab_0          28 KB  conda-forge
    async-lru-2.0.4            |     pyhd8ed1ab_0          15 KB  conda-forge
    attrs-23.2.0               |     pyh71513ae_0          53 KB  conda-forge
    aws-c-auth-0.6.19          |       h5eee18b_0          99 KB
    aws-c-cal-0.5.20           |       hff2c3d7_3          43 KB  conda-forge
    aws-c-common-0.8.5         |       h166bdaf_0         198 KB  conda-forge
    aws-c-compression-0.2.16   |       hf5f93bc_0          18 KB  conda-forge
    aws-c-event-stream-0.2.15  |       h6a678d5_0          50 KB
    aws-c-http-0.6.25          |       h5eee18b_0         200 KB
    aws-c-io-0.13.10           |       h2666983_0         140 KB  conda-forge
    aws-c-mqtt-0.7.13          |       h5eee18b_0          67 KB
    aws-c-s3-0.1.51            |       hdbd6064_0          71 KB
    aws-c-sdkutils-0.1.6       |       hf5f93bc_1          52 KB  conda-forge
    aws-checksums-0.1.13       |       hf5f93bc_5          51 KB  conda-forge
    aws-crt-cpp-0.18.16        |       h6a678d5_0         213 KB
    aws-sdk-cpp-1.10.55        |       h721c034_0         2.5 MB
    babel-2.14.0               |     pyhd8ed1ab_0         7.3 MB  conda-forge
    beautifulsoup4-4.12.3      |     pyha770c72_0         115 KB  conda-forge
    blas-1.1                   |         openblas           1 KB  conda-forge
    bleach-6.1.0               |     pyhd8ed1ab_0         128 KB  conda-forge
    blinker-1.8.2              |     pyhd8ed1ab_0          14 KB  conda-forge
    bokeh-3.4.1                |     pyhd8ed1ab_0         4.5 MB  conda-forge
    boost-cpp-1.82.0           |       hdb19cb5_2          11 KB
    bottleneck-1.3.8           |  py312hc7c0aa3_0         137 KB  conda-forge
    brotli-1.0.9               |       h166bdaf_9          20 KB  conda-forge
```

```
    brotli-bin-1.0.9           |       h166bdaf_9          20 KB  conda-forge
    cached-property-1.5.2      |       hd8ed1ab_1           4 KB  conda-forge
    cached_property-1.5.2      |     pyha770c72_1          11 KB  conda-forge
    certifi-2024.2.2           |     pyhd8ed1ab_0         157 KB  conda-forge
    click-8.1.7                |unix_pyh707e725_0          82 KB  conda-forge
```

```
#!conda install matplotlib --yes
```

```
Channels:
 - defaults
 - conda-forge
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - matplotlib


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    matplotlib-3.8.4           |  py312h06a4308_0           8 KB
    ------------------------------------------------------------
                                           Total:           8 KB

The following NEW packages will be INSTALLED:

  matplotlib         pkgs/main/linux-64::matplotlib-3.8.4-py312h06a4308_0

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

  certifi            conda-forge/noarch::certifi-2024.2.2-~ --> pkgs/main/linux-64::certifi-2024.2.2-py312h06a4308
  conda              conda-forge::conda-24.5.0-py312h7900f~ --> pkgs/main::conda-24.5.0-py312h06a4308_0



Downloading and Extracting Packages:

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

```
#!conda install scipy --yes
```

```
Channels:
 - defaults
 - conda-forge
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - scipy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    libmamba-1.5.8             |       hfe524e5_2         1.9 MB
    libmambapy-1.5.8           |  py312h2dafd23_2         327 KB
    pybind11-abi-5             |       hd3eb1b0_0          14 KB
    scipy-1.13.0               |  py312h2809609_0        23.7 MB
    ------------------------------------------------------------
                                           Total:        25.9 MB

The following packages will be UPDATED:

  libmamba                             1.5.8-hfe524e5_1 --> 1.5.8-hfe524e5_2
  libmambapy                      1.5.8-py312h2dafd23_1 --> 1.5.8-py312h2dafd23_2
  pybind11-abi                               4-hd3eb1b0_1 --> 5-hd3eb1b0_0
  scipy                         1.12.0-py312h2809609_0 --> 1.13.0-py312h2809609_0
```

```
Downloading and Extracting Packages:
scipy-1.13.0         | 23.7 MB   | :   0% 0/1 [00:00<?, ?it/s]
libmamba-1.5.8       | 1.9 MB    | :   0% 0/1 [00:00<?, ?it/s]

libmambapy-1.5.8     | 327 KB    | :   0% 0/1 [00:00<?, ?it/s]


scipy-1.13.0         | 23.7 MB   | :   0% 0.0006595458738429316/1 [00:00<03:07, 187.22s/it]


pybind11-abi-5       | 14 KB     | : 100% 1.0/1 [00:00<00:00,  8.28it/s]

libmambapy-1.5.8     | 327 KB    | :   5% 0.048856710044491096/1 [00:00<00:02,  2.62s/it]
libmamba-1.5.8       | 1.9 MB    | :   1% 0.008399983593782044/1 [00:00<00:15, 15.94s/it]


scipy-1.13.0         | 23.7 MB   | :  21% 0.2090760420082093/1 [00:00<00:00,  1.12it/s]

libmambapy-1.5.8     | 327 KB    | : 100% 1.0/1 [00:00<00:00,  5.18it/s]

scipy-1.13.0         | 23.7 MB   | :  56% 0.5619330845141777/1 [00:00<00:00,  2.16it/s]
libmamba-1.5.8       | 1.9 MB    | : 100% 1.0/1 [00:00<00:00,  2.77it/s]
```

```
!conda install xlrd --yes
```

```
Channels:
 - defaults
 - conda-forge
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - xlrd


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    xlrd-2.0.1                 |     pyhd3eb1b0_1          97 KB
    ------------------------------------------------------------
                                           Total:          97 KB

The following NEW packages will be INSTALLED:

  xlrd               pkgs/main/noarch::xlrd-2.0.1-pyhd3eb1b0_1



Downloading and Extracting Packages:

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

```
!conda install openpyxl --yes
```

```
Channels:
 - defaults
 - conda-forge
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - openpyxl


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    et_xmlfile-1.1.0           |  py312h06a4308_1          12 KB
```

```
    openpyxl-3.1.2              |  py312h5eee18b_0         710 KB
    ------------------------------------------------------------
                                               Total:         722 KB

The following NEW packages will be INSTALLED:

  et_xmlfile         pkgs/main/linux-64::et_xmlfile-1.1.0-py312h06a4308_1
  openpyxl           pkgs/main/linux-64::openpyxl-3.1.2-py312h5eee18b_0



Downloading and Extracting Packages:
openpyxl-3.1.2       | 710 KB    | :   0% 0/1 [00:00<?, ?it/s]
openpyxl-3.1.2       | 710 KB    | :   2% 0.02253139942488823/1 [00:00<00:05,  5.73s/it]
et_xmlfile-1.1.0     | 12 KB     | : 100% 1.0/1 [00:00<00:00,  6.72it/s]


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

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

- FMU - BPL_TEST2_Batch_linux_om_me.fmu
- Setup-file - BPL_TEST2_Batch_fmpy_explore.py

```
%%bash
git clone https://github.com/janpeter19/BPL_TEST2_Batch_calibration
```

```
Cloning into 'BPL_TEST2_Batch_calibration'...
```

```
%cd BPL_TEST2_Batch_calibration
```

```
/content/BPL_TEST2_Batch_calibration
```

BPL_TEST2_Batch_calibration - demo

Author: Jan Peter Axelsson

This notebook shows the possibilities for calibration of the model BPL_TEST2_Batch using scipy.optimize.minimize() routine. There are several different methods to choose between. In this notebook we work with simulated data.

The text-book model of batch cultivation we simulate is the following where S is substrate, X is cell concentration, and V is volume of the broth

$$\frac{d(VS)}{dt} = -q_S(S) \cdot VX$$
$$\frac{d(VX)}{dt} = \mu(S) \cdot VX$$

and where specific cell growth rate μ and substrate uptake rate q_S are

$$\mu(S) = Y \cdot q_S(S)$$
$$q_S(S) = q_S^{max} \frac{S}{K_s + S}$$

where Y is the yield, q_S^{max} is the maximal specific substrate uptake rate and K_s is the corresponding saturation constant.

The parameter estimation is done with optimization methods that only require evaluation of the missmatch between simulation with given parameters and data. At start the allowed range for each parameter is given. The method used for optimization is SLSQP but can easily be changed [1].

In the near future the FMU may provide first derivative gradient informaion, that will make it possible to choose corresponding method of minimize() for improved performance. This possibility is related to the upgrade to the FMI-standard ver 3.0 for the Modelica compiler.

The Python package PyFMI [2] that is the base for FMU-explore has a simplified built-in functionality for parameter estimation that also use scipy.optimize.minimize(). However, there is estimatation functionaly but the purpose seems to only address smaller examples. There is for instance no support to handle models that takes sub-models from libraries and necesssary changes of default parameters not to be estimated. Therefore we here define a Python function evaluate() that facilitate the formulation of the parameter estimation and also bring flexibility to choice of optimization method, default Nelder-Mead.

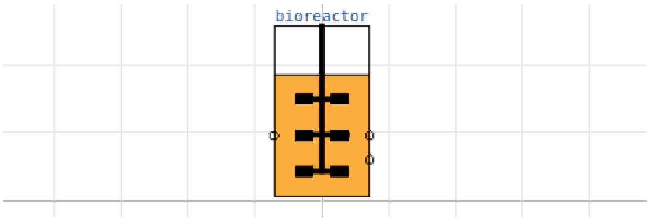
```
run -i BPL_TEST2_Batch_fmpy_explore.py
```

```
Linux - run FMU pre-compiled OpenModelica 1.21.0

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()
```

```
# Adjust the size of diagrams
plt.rcParams['figure.figsize'] = [15/2.54, 12/2.54]
```

```
process_diagram()
```

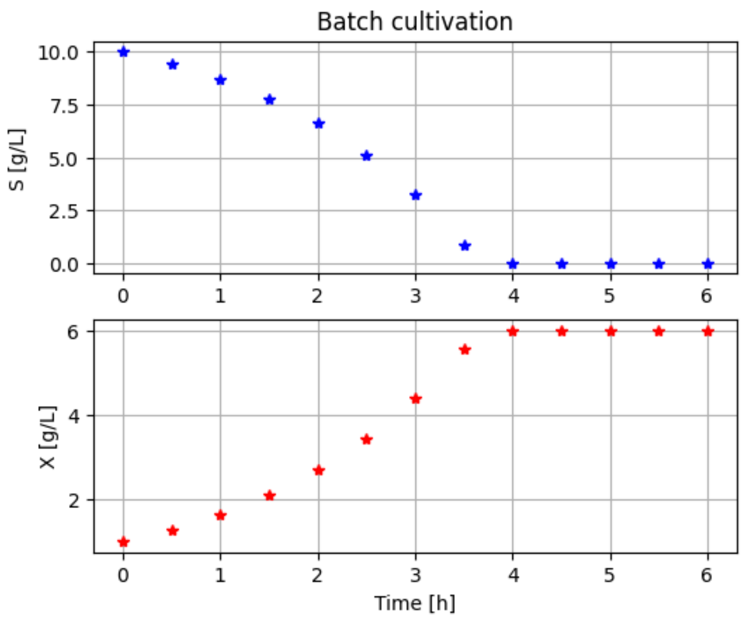
```
No processDiagram.png file in the FMU, but try the file on disk.
```



1 Generate data later used for parameter estimation

```
import pandas as pd
```

```
# Data generated
simulationTime = 6.0
par(Y=0.50, qSmax=1.00, Ks=0.1)
init(V_start=1.0, VS_start=10, VX_start=1.0)
newplot(plotType='Demo_2')
simu(simulationTime, options=opts_data)
```




```
# Store data in a DataFrame for later use
data = pd.DataFrame(data={'time':sim_res['time'], 'X':sim_res['bioreactor.c[1]'], 'S':sim_res['bioreactor.c[2]']})
data
```

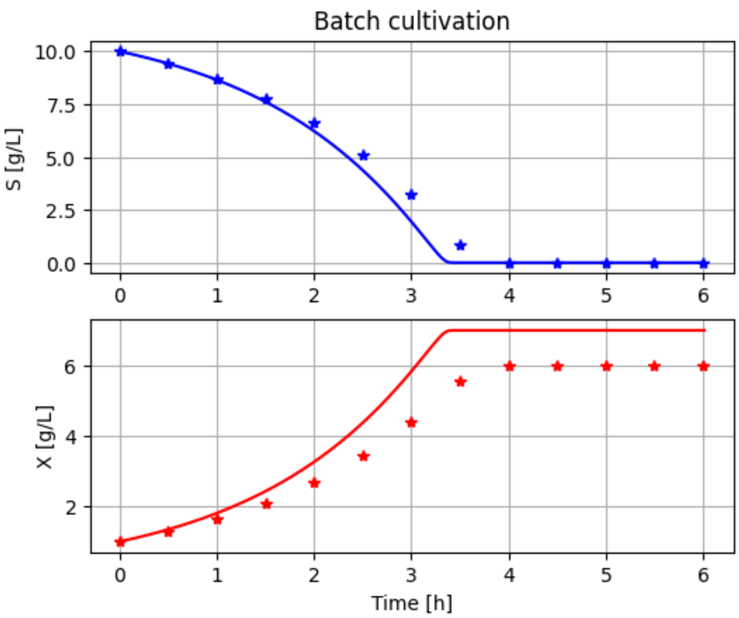
	time	X	S
0	0.0	1.000000	1.000000e+01
1	0.5	1.280777	9.438447e+00
2	1.0	1.640089	8.719823e+00
3	1.5	2.099634	7.800732e+00
4	2.0	2.686794	6.626412e+00
5	2.5	3.435509	5.128981e+00
6	3.0	4.385357	3.229286e+00
7	3.5	5.559274	8.814516e-01
8	4.0	6.000000	4.618858e-07
9	4.5	6.000000	-7.022022e-11
10	5.0	6.000000	-2.610415e-12
11	5.5	6.000000	1.020722e-14
12	6.0	6.000000	-1.037244e-17

2 Simulation with initial guess of parameters compared with data

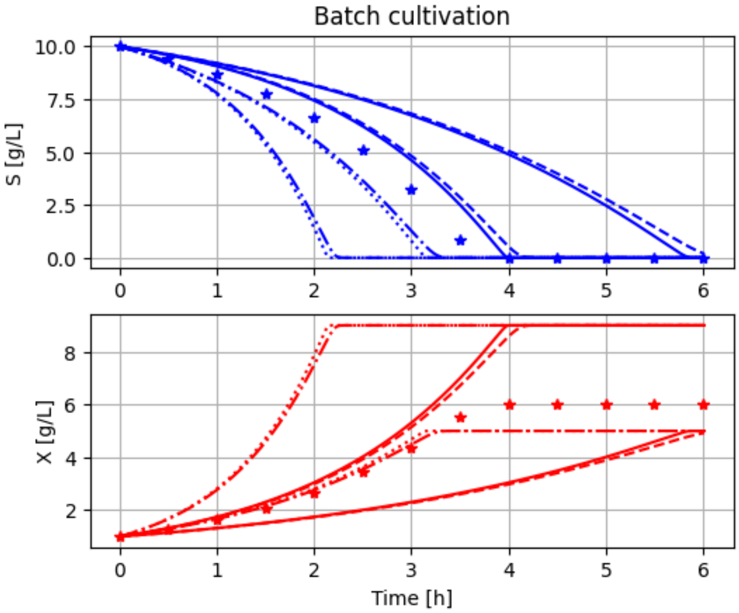
Here we define the parameters that should be estimated and specify allowed ranges. Nominal parameters are chosen as the mid-point of the allowed parameter range.

Simulation with these nominal parameter set and compare with data give an idea of who well the model fit data.

```
# Parameters to be estimated using parDict names and their bounds
parEstim = ['Y', 'qSmax', 'Ks']
parBounds = [(0.4, 0.8), (0.7, 1.3), (0.05, 0.20)]
parEstim_0 = [np.mean(parBounds[k]) for k in range(len(parBounds))]


# Simulation with nominal parameters
newplot(plotType='Demo_1')
par(Y=parEstim_0[0], qSmax=parEstim_0[1], Ks=parEstim_0[2])
simu(simulationTime)

# Show data
ax1.plot(data['time'], data['S'],'b*')
ax2.plot(data['time'], data['X'],'r*')
plt.show()
```



```
# Simulation over the parameter ranges given
newplot(plotType='Demo_1')
for Y_value in parBounds [0]:
    for qSmax_value in parBounds[1]:
        for Ks_value in parBounds[2]:
            par(Y=Y_value, qSmax=qSmax_value, Ks=Ks_value)
            simu(simulationTime)

# Show data
ax1.plot(data['time'], data['S'],'b*')
ax2.plot(data['time'], data['X'],'r*')
plt.show()
```



Simulation over the diferent parameter combinations of the parameter bounds shows that data is "covered" and we have good hope to find a parameter combination that fits data well.

3 Parameter estimation

Here we use the scipy.optimize.minimize() procedure which contain a family of different methods [1]. The default method is Nelder-Mead and is robust for fitting a model to data. Further we have chosen to work with bounds for the parameters to be estimated and the initial guess is chosen as the middle point in parameter space.

```
# Optimization routine import
import scipy.optimize


# Parameters to be estimated using parDict names and their bounds
extra_args = (parEstim, data, fmu_model, simulationTime, parDict, parLocation)


# Modified evaluation function tailored for Python optimization algorithms
def objective(x, parEstim, data=data, fmu_model=fmu_model, simulationTime=simulationTime,
              parDict=parDict, parLocation=parLocation):
    """The parameter list is tailored for scipy optimization algorithms interface,
       where the first parameter x is an array with parameters that are tuned
       and evalauted and parEstim is a list of the names of these parameters.
       The code can be made 20-30% faster, but loner, using pyfmi-commands directly."""

    # Update parameters and simulate
    for i, p in enumerate(parEstim): par(**{p:x[i]})
    simu(simulationTime, options=opts_data)

    # Calculate loss function V
    V={}
    V['X'] = np.linalg.norm(data['X'] - np.interp(data['time'], sim_res['time'], sim_res['bioreactor.c[1]']))
    V['S'] = np.linalg.norm(data['S'] - np.interp(data['time'], sim_res['time'], sim_res['bioreactor.c[2]']))

    return V['X'] + V['S']
```

```
import time


# Run minimize()
start_time = time.time()
result = scipy.optimize.minimize(objective, x0=parEstim_0, args=extra_args,
                                 method='Nelder-Mead', bounds=parBounds, options={"disp":True})
print('CPU-time =', time.time()-start_time)
```

```
Optimization terminated successfully.
         Current function value: 0.045511
         Iterations: 39
         Function evaluations: 75
CPU-time = 1.915511131286621
```

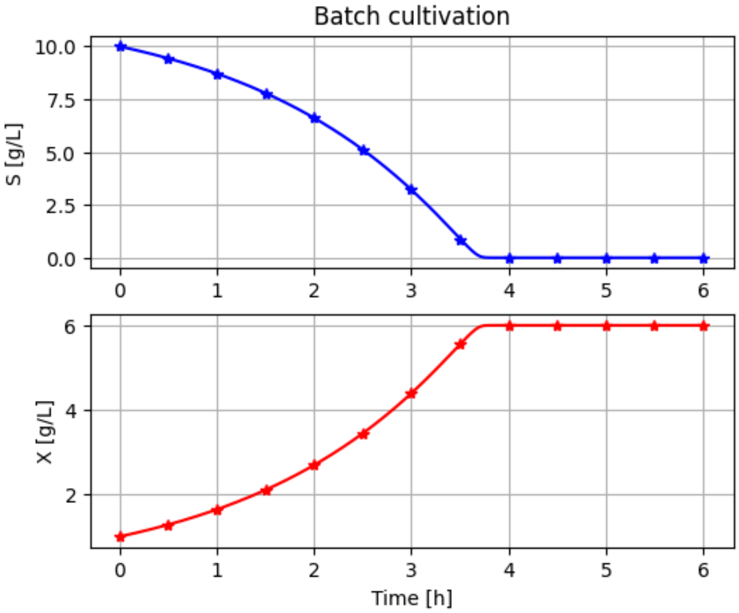
```
result
```

```
       message: Optimization terminated successfully.
       success: True
        status: 0
           fun: 0.045510651189744136
             x: [ 5.001e-01  1.007e+00  1.405e-01]
           nit: 39
          nfev: 75
 final_simplex: (array([[ 5.001e-01,  1.007e+00,  1.405e-01],
                       [ 5.001e-01,  1.007e+00,  1.405e-01],
                       [ 5.001e-01,  1.007e+00,  1.405e-01],
                       [ 5.001e-01,  1.007e+00,  1.405e-01]]), array([ 4.551e-02,  4.552e-02,  4.556e-02,
4.559e-02]))
```

The estimated parameters result.x are very close to the original values and no surprise.

4 Simulation with estimated parameters compared with data

```
newplot(plotType='Demo_1')
par(Y=result.x[0], qSmax=result.x[1], Ks=result.x[2])
simu(simulationTime)

# Show data
ax1.plot(data['time'], data['S'],'b*')
ax2.plot(data['time'], data['X'],'r*')
plt.show()
```



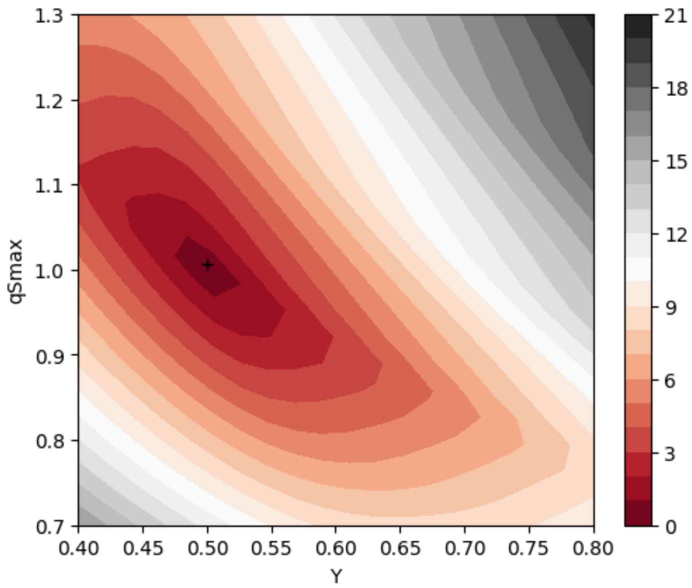

```
# The estimated parameters are
for i in range(len(parEstim)): print(parEstim[i],':', result.x[i])
```

```
Y : 0.5000549816703198
qSmax : 1.0066481313036082
Ks : 0.14048278753421406
```

5 Analysis of the loss function

The problem is small and analysis of the loss function brings some insight. From the diagram above showing parameter sweep over combinations min- and max-parameters we see that the parameter K_s has little influence. Let use set that a fixed value and then plot the loss function in the parameters Y and $qSmax$. We do this by go through all the parametera combinations and evaluate each of them.

```
# Sweep through Y and qSmax variation and store the value of the loss-function for each
nY = 20
nqSmax = 20
V = np.zeros((nY, nqSmax))

Y = np.linspace(parBounds[0][0],parBounds[0][1],nY)
qSmax = np.linspace(parBounds[1][0],parBounds[1][1],nqSmax)

for j in range(nY):
    for k in range(nqSmax):
        V[k,j] = objective([Y[j], qSmax[k], 0.1], parEstim)

# Contour plot
plt.figure()
plt.clf
plt.subplot(1,1,1)
plt.contourf(Y, qSmax, V, 20, cmap='RdGy')
plt.plot(result.x[0], result.x[1],'k+')
plt.colorbar()
plt.ylabel('qSmax')
plt.xlabel('Y')
plt.show()
```



We see the following in the contour diagram of the loss function simplified:

- The minima is unique in the range of parmaters we study. This is good news.
- The contour plot is ellipsoid and rather narrow. The more narrow the ellipsoid the more difficult and more time it takes to converge to the minima.
- The direction of the ellipsoid axis indicate the correlation you may get between the two parameters during the minimization process.

Note that the form of the contour plot change with the parameters (and initial values) of the actual proces. You can see the impact by changing the parameters in "cell # 4" where data is generated and then just choose to run that cell and the cells below. No need to restart the notebook.

6 Summary

A choice was made to work with allowed ranges of parameters to be estimated and a start value was defined as the center point in this parameter space. There are only three methods available in optimize.minimize() that can handle bounds on parameters.

An evaluate() function was created that define how the difference beween simulation and data is measured. The function is rather transparent and easy to modify and you may want to change weight on the loss in S and X, for instance. Here they have so far equal weight.

The FMU-explore workspace dictionaries partDict[] and parLocation[] are useful also here and simplify the code for the evaluation() function. But we also use the detailed PyFMI-functions to administrate and set parameters of the actual simulation.

The call optimize.minimize() has several parameters and can easily be modified, for instance change of method. For fitting a model to data Nelder-Mead is ao a robust and good choice, but can be somewhat slow.

The estimated parameters were close to perfect!

The contour plot of the simplified loss function shows that the minima is unique and should not be difficult too difficut to obtain. More narrow elliptical contour plots would indicate difficulties. Multiple local minima would also be a problem.

7 References

[1] Scipy Reference guide on optimize.minimize() here

[2] Andersson, C., Åkesson, J., Fuhrer C. : "PyFMI: A Python package for simulation of coupled dynamic models with the functional mock-up interface", Centre for Mathematical Sciences, Lund University, Report LUTFNA-5008-2016, 2016.

Appendix

```
describe('parts')
```

```
['bioreactor', 'bioreactor.culture']
```

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

```
MSL: 3.2.3 - used components: none
```

```
system_info()
```

```
System information
 -OS: Linux
 -Python: 3.10.12
 -Scipy: 1.11.4
 -FMPy: 0.3.20
 -FMU by: OpenModelica Compiler OpenModelica 1.21.0
 -FMI: 2.0
 -Type: ME
 -Name: BPL_TEST2.Batch
 -Generated: 2024-03-05T08:05:44Z
 -MSL: 3.2.3
 -Description: Bioprocess Library version 2.2.0
 -Interaction: FMU-explore for FMPy version 1.0.0
```