

# BPL\_TEST2\_Batch\_design\_space - demo

```
In [1]: run -i BPL_TEST2_Batch_explore.py
```

Windows - run FMU pre-compiled JModelica 2.14

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

Brief information about a command by help(), eg help(simu)

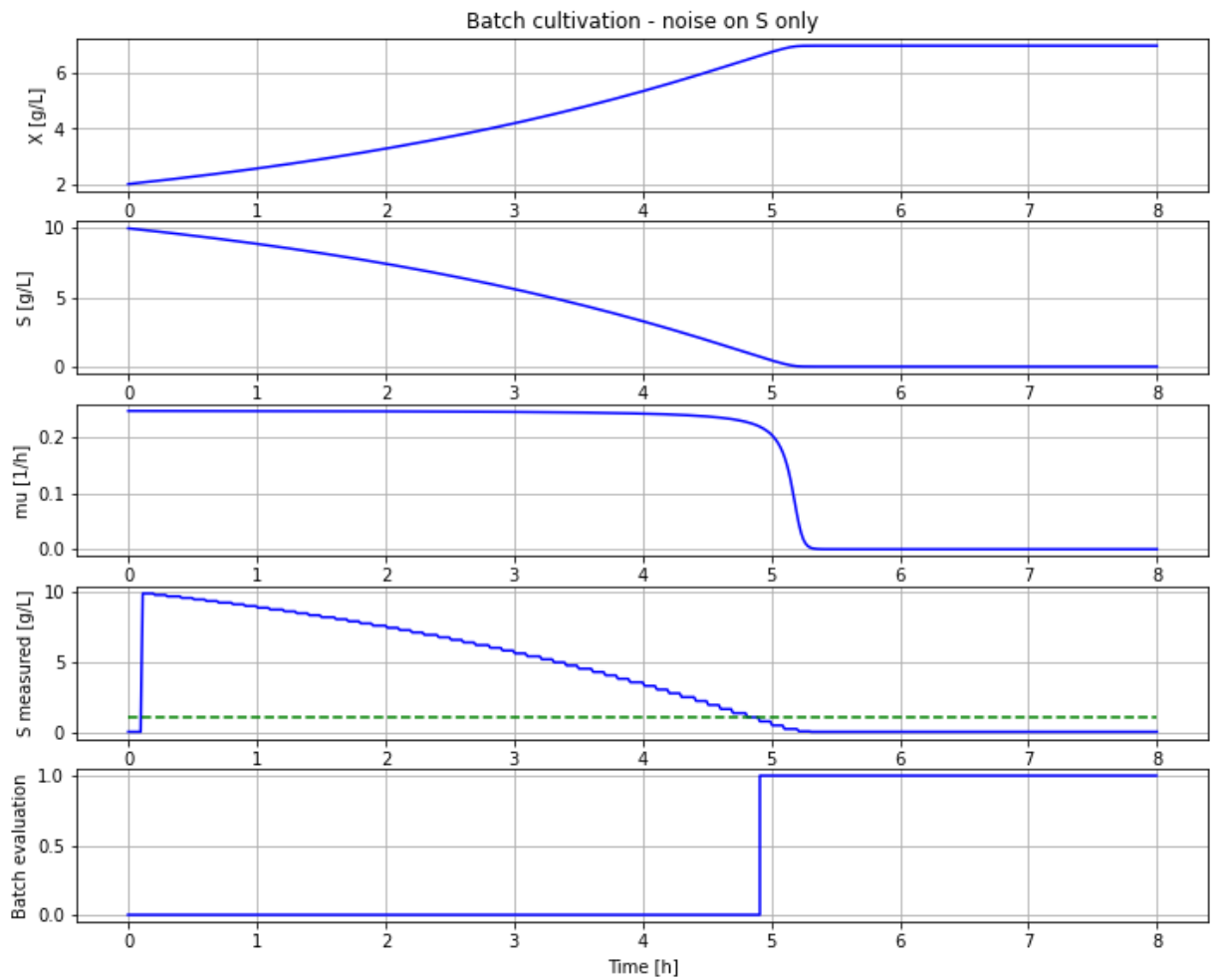
Key system information is listed with the command system\_info()

```
In [2]: plt.rcParams['figure.figsize'] = [30/2.54, 24/2.54]
```

## Batch evaluation

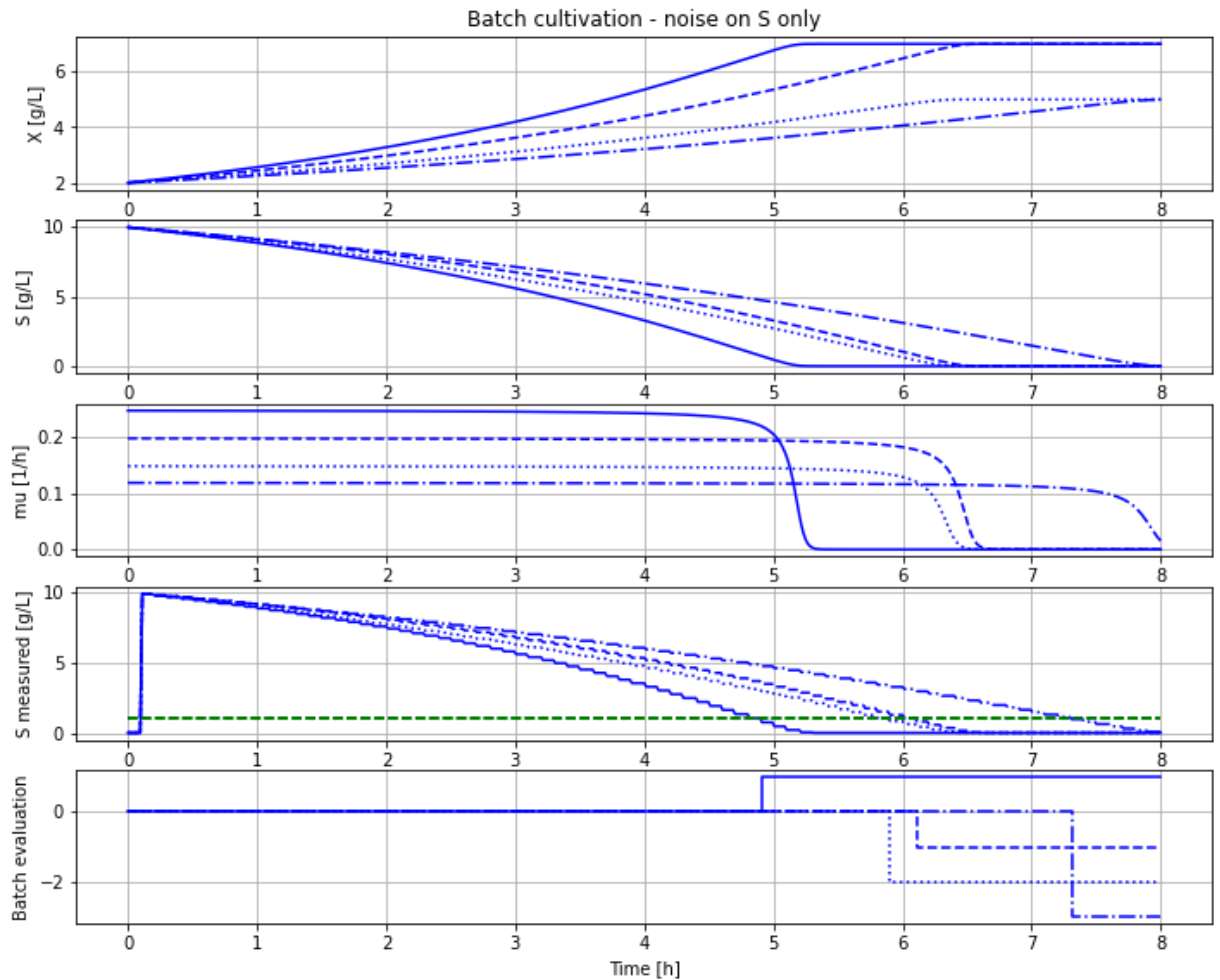
```
In [3]: # Nominal parameters
par(S_min=1.0, time_final_max=6.0, X_final_min=5.0)
init(VX_0=2, VS_0=10)
par(Y=0.5, qSmax=0.5, Ks=0.1)
par(sigma=0.0)
```

```
In [4]: # Simulation of nominal parameters that gives a batch that meed the end criteria
newplot(plotType='TimeSeries_2')
simu(8)
```



In [5]:

```
# Example of process parameter changes and how they meet the end criteria
newplot(plotType='TimeSeries_2')
par(Y=0.50, qSmax=0.50); simu(8) # - pass
par(Y=0.50, qSmax=0.40); simu(8) # - fail criteria time_final < 6.0
par(Y=0.30, qSmax=0.50); simu(8) # - fail criteria X_final > 5.0
par(Y=0.30, qSmax=0.40); simu(8) # - fail both criteria
```



## Batch evaluation under process variation - parameter sweep

```
In [6]: # Define sweep ranges and storage of final data
nY = 20
nqSmax = 20
Y_range = np.linspace(0.3,0.5,nY)
qSmax_range = np.linspace(0.4,0.6,nqSmax)
data = np.zeros([nY,nqSmax,5])
```

```
In [13]: # Run parameter sweep
newplot(plotType='TimeSeries_2_diagrams')
init(VX_0=2, VS_0=10)
par(sigma=0.0, seed=1)

for j in range(nY):
    for k in range(nqSmax):
        par(Y=Y_range[j])
        par(qSmax=qSmax_range[k])
        simu(8)

    # Store final results
    data[j,k,0] = Y_range[j]
    data[j,k,1] = qSmax_range[k]
    data[j,k,2] = sim_res['monitor.time_final'][-1]
    data[j,k,3] = sim_res['monitor.X_final'][-1]
    data[j,k,4] = sim_res['monitor.batch_evaluation'][-1]

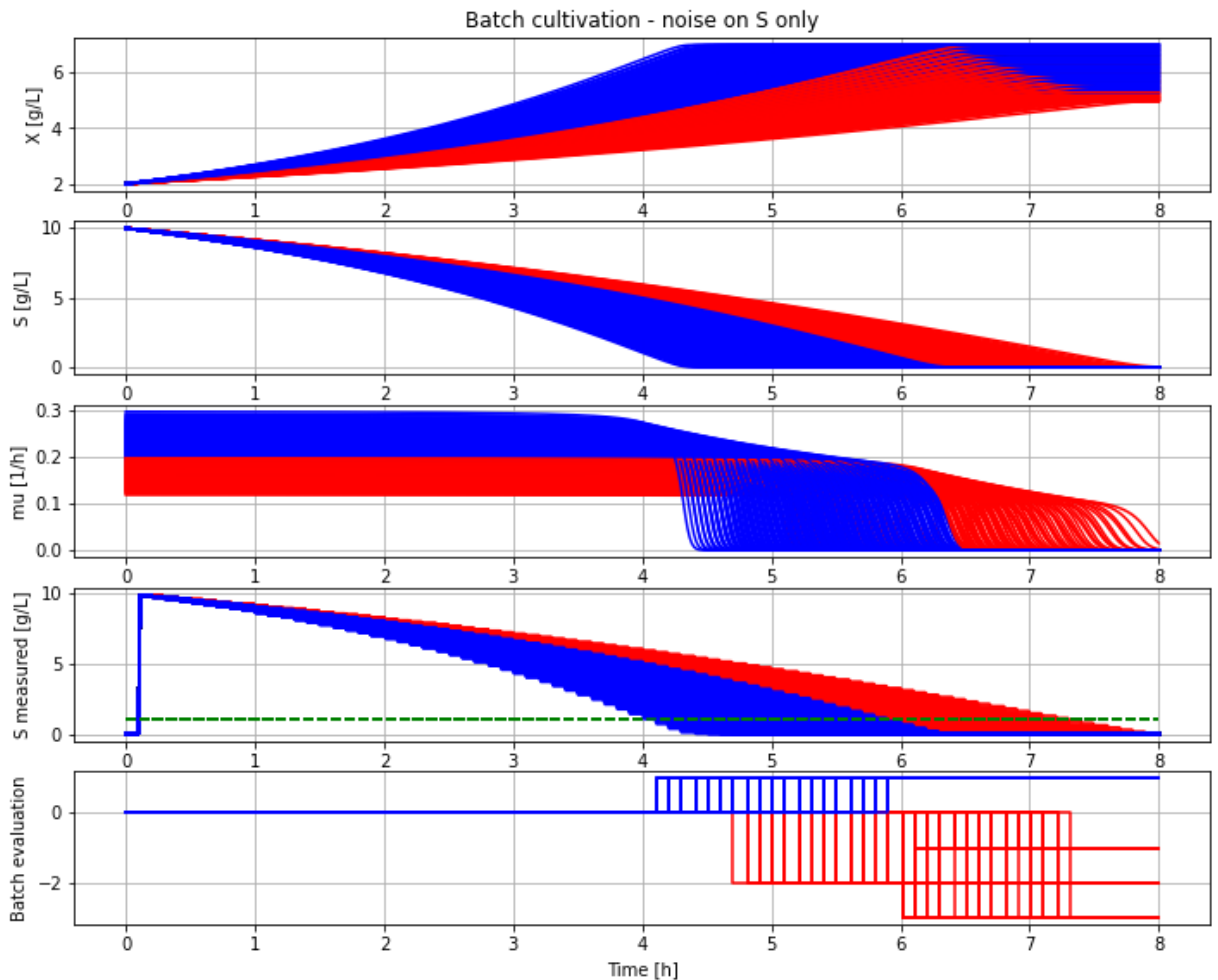
    # Plot simulation results
```

```

if sim_res['monitor.batch_evaluation'][-1] > 0:
    ax1.plot(sim_res['time'], sim_res['bioreactor.c[1]'], 'b-')
    ax2.plot(sim_res['time'], sim_res['bioreactor.c[2]'], 'b-')
    ax3.plot(sim_res['time'], sim_res['bioreactor.culture.q[1]'], 'b-')
    ax4.plot(sim_res['time'], sim_res['sensor.out.c[2]'], 'b-')
    ax4.plot([0, simulationTime], [model.get('monitor.S_min'), model.get('mo
    ax5.step(sim_res['time'], sim_res['monitor.batch_evaluation'], where='post
else:
    ax1.plot(sim_res['time'], sim_res['bioreactor.c[1]'], 'r-')
    ax2.plot(sim_res['time'], sim_res['bioreactor.c[2]'], 'r-')
    ax3.plot(sim_res['time'], sim_res['bioreactor.culture.q[1]'], 'r-')
    ax4.plot(sim_res['time'], sim_res['sensor.out.c[2]'], 'r-')
    ax4.plot([0, simulationTime], [model.get('monitor.S_min'), model.get('mo
    ax5.step(sim_res['time'], sim_res['monitor.batch_evaluation'], where='post

plt.show()

```



In [14]:

```

# Show end results
plt.figure()
ax1 = plt.subplot(1,2,1)
ax2 = plt.subplot(1,2,2)

for j in range(nY):
    for k in range(nqSmax):
        if data[j,k,4] > 0:
            ax1.scatter(data[j,k,0], data[j,k,1], c='b')
        else:
            ax1.scatter(data[j,k,0], data[j,k,1], c='r')
ax1.grid()
#plt.axis([0, 0.8, 0, 0.8])
ax1.set_ylabel('qSmax [g/g,h]')
ax1.set_xlabel('Yxs [g/g]')

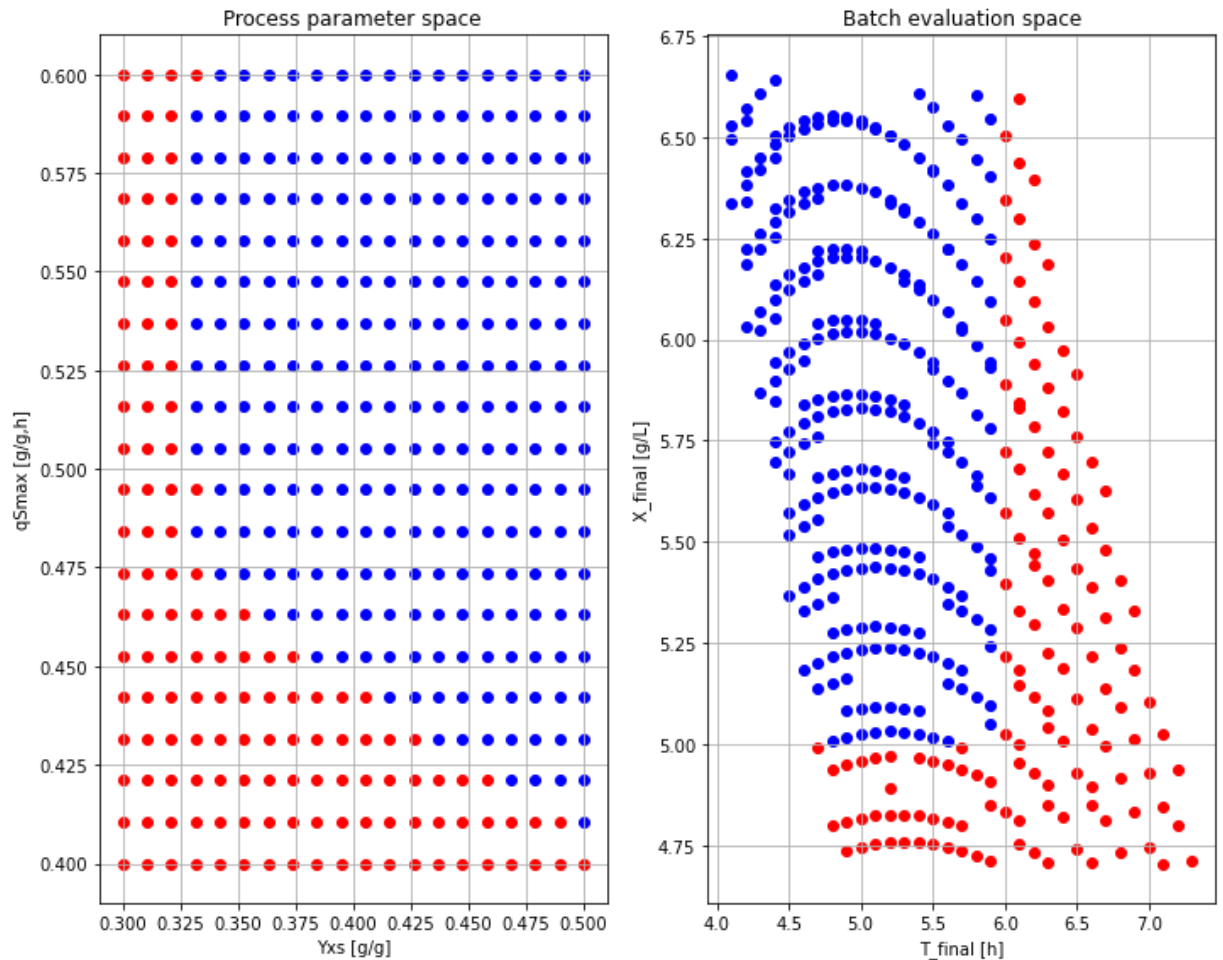
```

```

ax1.set_title('Process parameter space')

for j in range(nY):
    for k in range(nqSmax):
        if data[j,k,4] > 0:
            ax2.scatter(data[j,k,2],data[j,k,3],c='b')
        else:
            ax2.scatter(data[j,k,2],data[j,k,3],c='r')
ax2.grid()
#plt.axis([0, 8, 0, 8])
ax2.set_xlabel('T_final [h]')
ax2.set_ylabel('X_final [g/L]')
ax2.set_title('Batch evaluation space')
plt.show()

```



## Batch evaluation under substrate measurement error

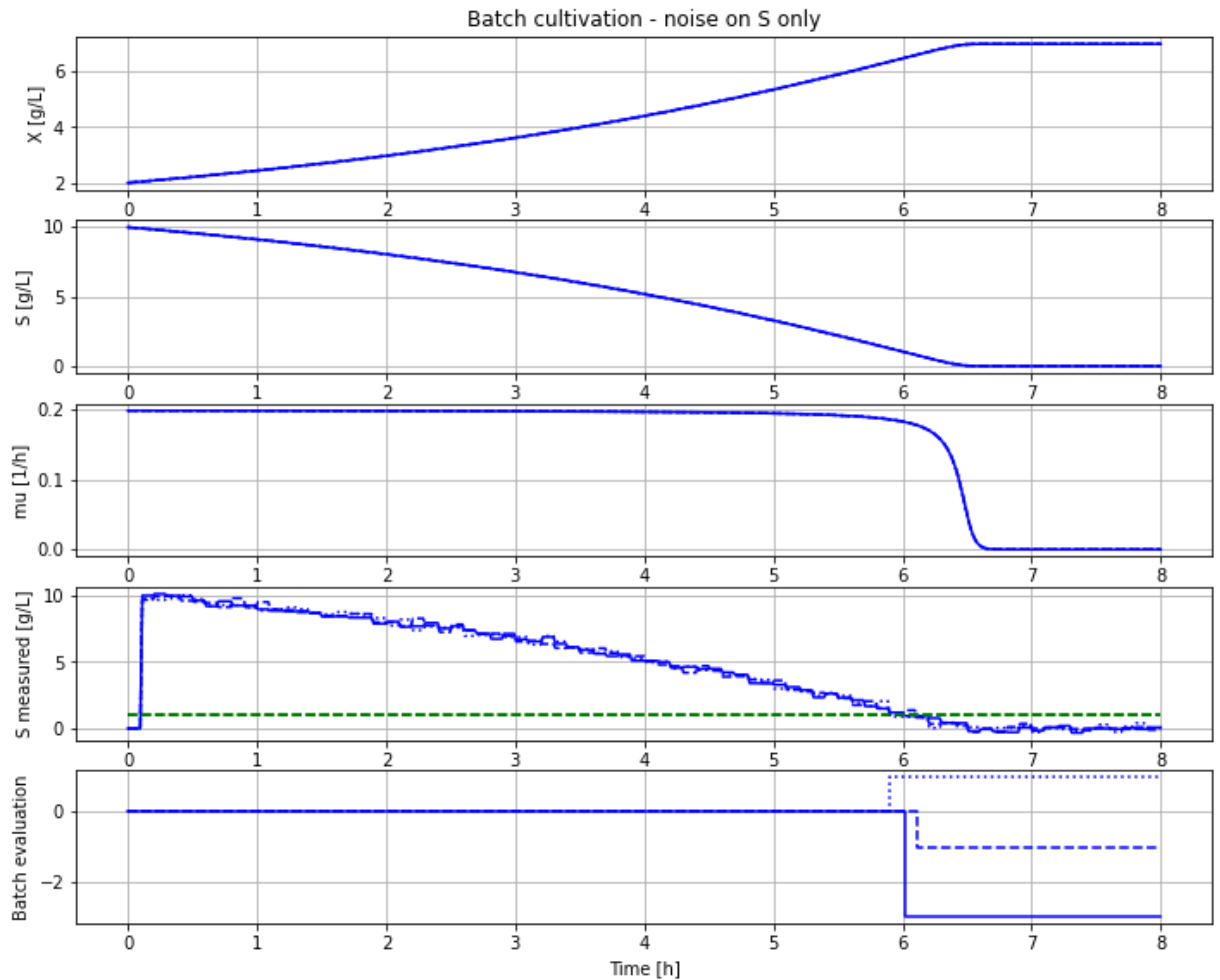
In [22]:

```

# Simulation of nominal parameters that gives a batch that meed the end criteria
newplot(plotType='TimeSeries_2')
par(sigma=0.2)
par(Y=0.5, qSmax=0.4);

for value in [2,3,5]: par(seed=value); simu(8)

```



## Batch evaluation under process variation and measurement error - parameter sweep

```
In [23]: # Define sweep ranges and storage of final data
nY = 20
nqSmax = 20
Y_range = np.linspace(0.3,0.5,nY)
qSmax_range = np.linspace(0.4,0.6,nqSmax)
data = np.zeros([nY,nqSmax,5])
```

```
In [24]: # Run parameter sweep
newplot(plotType='TimeSeries_2_diagrams')
par(sigma=0.2, seed=1)

for j in range(nY):
    for k in range(nqSmax):
        par(Y=Y_range[j])
        par(qSmax=qSmax_range[k])
        simu(8)

    # Store final results
    data[j,k,0] = Y_range[j]
    data[j,k,1] = qSmax_range[k]
    data[j,k,2] = sim_res['monitor.time_final'][-1]
    data[j,k,3] = sim_res['monitor.X_final'][-1]
    data[j,k,4] = sim_res['monitor.batch_evaluation'][-1]

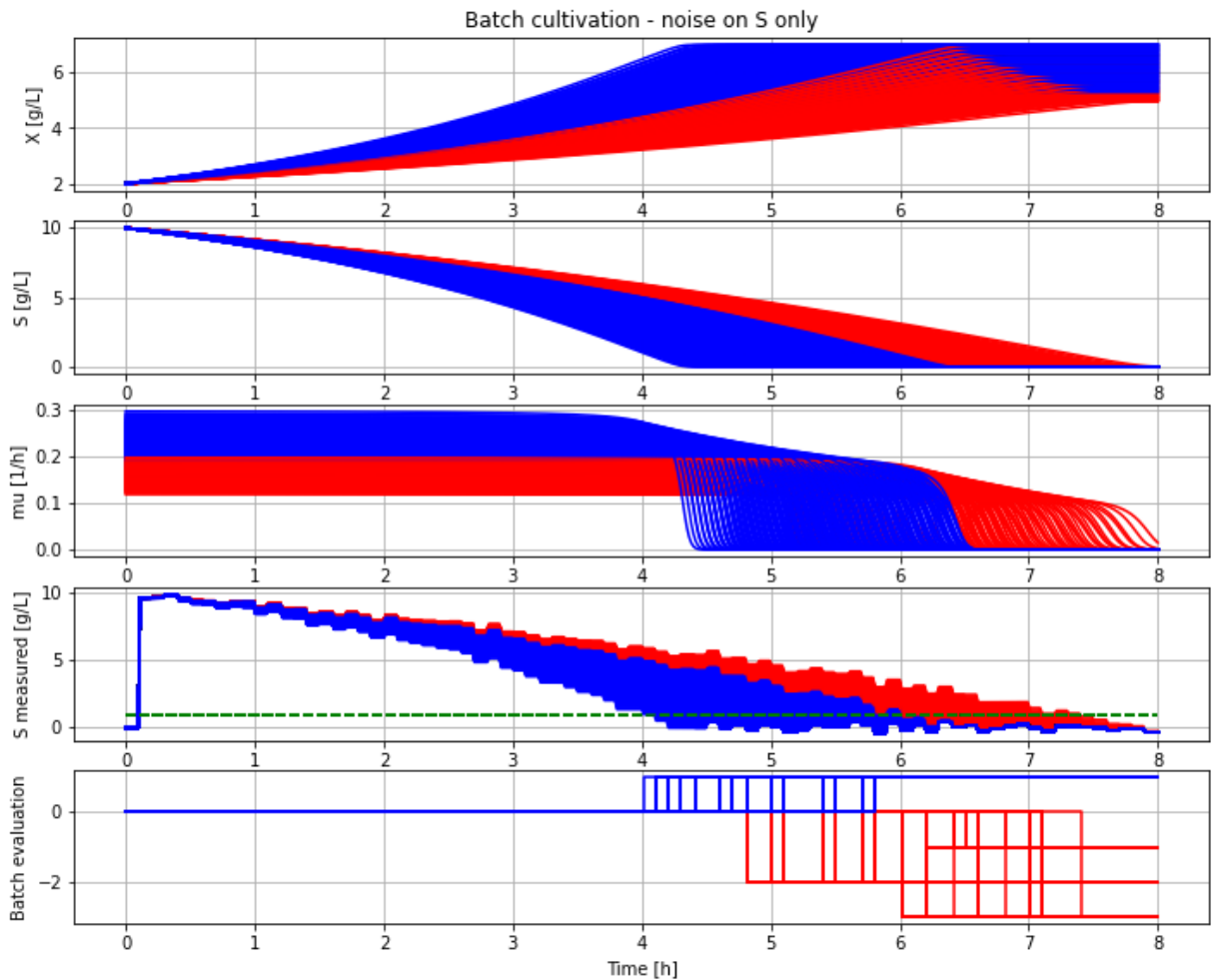
    # Plot simulation results
    if sim_res['monitor.batch_evaluation'][-1] > 0:
```

```

ax1.plot(sim_res['time'], sim_res['bioreactor.c[1]'], 'b-')
ax2.plot(sim_res['time'], sim_res['bioreactor.c[2]'], 'b-')
ax3.plot(sim_res['time'], sim_res['bioreactor.culture.q[1]'], 'b-')
ax4.plot(sim_res['time'], sim_res['sensor.out.c[2]'], 'b-')
ax4.plot([0, simulationTime], [model.get('monitor.S_min'), model.get('mo
ax5.step(sim_res['time'], sim_res['monitor.batch_evaluation'], where='post
else:
ax1.plot(sim_res['time'], sim_res['bioreactor.c[1]'], 'r-')
ax2.plot(sim_res['time'], sim_res['bioreactor.c[2]'], 'r-')
ax3.plot(sim_res['time'], sim_res['bioreactor.culture.q[1]'], 'r-')
ax4.plot(sim_res['time'], sim_res['sensor.out.c[2]'], 'r-')
ax4.plot([0, simulationTime], [model.get('monitor.S_min'), model.get('mo
ax5.step(sim_res['time'], sim_res['monitor.batch_evaluation'], where='post

plt.show()

```



In [25]:

```

# Show end results
plt.figure()
ax1 = plt.subplot(1,2,1)
ax2 = plt.subplot(1,2,2)

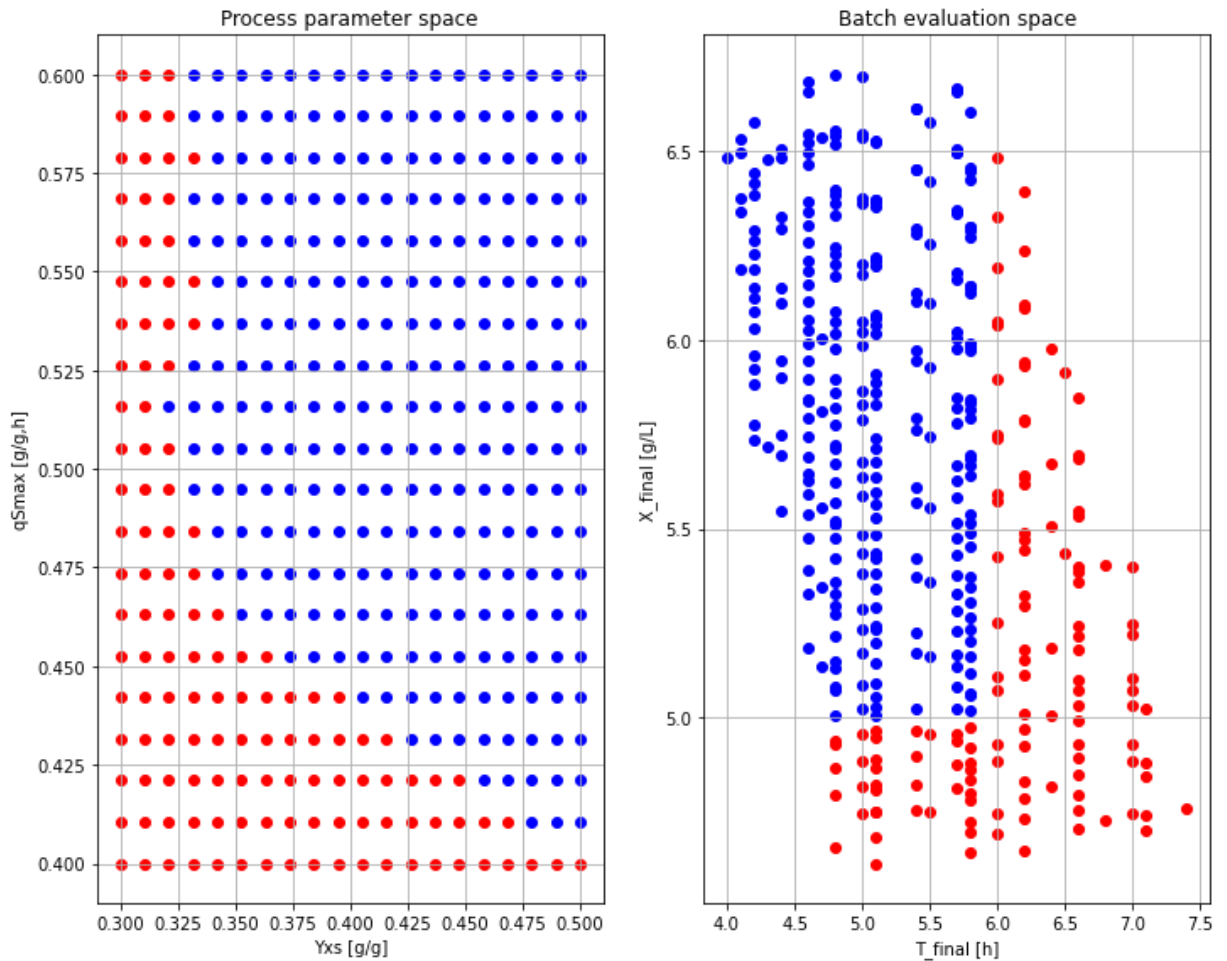
for j in range(nY):
    for k in range(nqSmax):
        if data[j,k,4] > 0:
            ax1.scatter(data[j,k,0], data[j,k,1], c='b')
        else:
            ax1.scatter(data[j,k,0], data[j,k,1], c='r')
ax1.grid()
#plt.axis([0, 0.8, 0, 0.8])
ax1.set_ylabel('qSmax [g/g,h]')
ax1.set_xlabel('Yxs [g/g]')
ax1.set_title('Process parameter space')

```

```

for j in range(nY):
    for k in range(nqSmax):
        if data[j,k,4] > 0:
            ax2.scatter(data[j,k,2],data[j,k,3],c='b')
        else:
            ax2.scatter(data[j,k,2],data[j,k,3],c='r')
ax2.grid()
#plt.axis([0, 8, 0, 8])
ax2.set_xlabel('T_final [h]')
ax2.set_ylabel('X_final [g/L]')
ax2.set_title('Batch evaluation space')
plt.show()

```



## Summary

## References

Axelsson J.P. and A. Elsheikh: "An example of sensitivity analysis of a bioprocess using Bioprocess Library for Modelica", Proceedings MODPROD, Linköping, Sweden 2019, see presentation [here](#).

## Appendix

In [26]:

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

```
['bioreactor', 'bioreactor.culture', 'liquidphase', 'monitor', 'MSL', 'sensor']
```



```
In [27]: describe('MSL')
```

```
MSL: 3.2.2 build 3 - used components: Noise.NormalNoise
```

```
In [28]: system_info()
```

```
System information
```

```
-OS: Windows
```

```
-Python: 3.9.5
```

```
-PyFMI: 2.9.5
```

```
-FMU by: JModelica.org
```

```
-FMI: 2.0
```

```
-Type: FMUModelCS2
```

```
-Name: BPL_TEST2.BatchWithNoise
```

```
-Generated: 2022-09-04T14:31:31
```

```
-MSL: 3.2.2 build 3
```

```
-Description: Bioprocess Library version 2.1.0 beta
```

```
-Interaction: FMU-explore ver 0.9.3
```

```
In [ ]:
```