Modelling of the E. coli Lac operon and parameter estimation

Student numbers: 584833 and 583875

March 2023

1 Modelling of the Lac operon

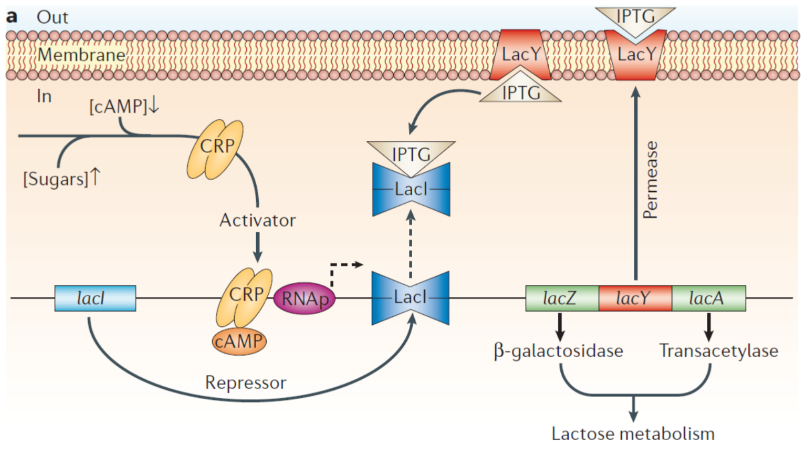

Figure 1: The Lac operon model in E. coli (Smits et al. 2006)

a.

The model of the Lac operon is shown in Figure 1. From this figure, we identify the important variables of the system to be:

- Extracellular IPTG (and other sugars like lactose and allolactose)
- Intracellular IPTG (and other sugars like lactose and allolactose)
- The inhibitor LacI
- The permease LacY
- β-galactosidase (encoded by lacZ)
- Transacetylase (encoded by lacA)

As a note, we will refer to β-galactosidase and transacetylase as, respectively, LacZ and LacA, while the genes encoding these proteins are written lacZ and lacA.

b.

The variables listed in 1.a can be organised as in the reaction diagram given in figure 2. In this model, we decided to group LacA and LacZ as both function in the metabolism (degradation) of intracellular lactose. We therefore give production and degradation constants k_3 and k_{dAZ} to the grouped LacA and LacZ. Furthermore, we assume that LacI inhibits LacA, LacZ and LacY in the same manner, with inhibition constant K_I. Also, LacA, LacZ and LacY are produced at the same rate, as they are encoded on a same operon. Therefore, the grouped LacA and LacZ are produced at a rate $2k_3$, while LacY is produced at a rate k_3. The other constants are self-explanatory.

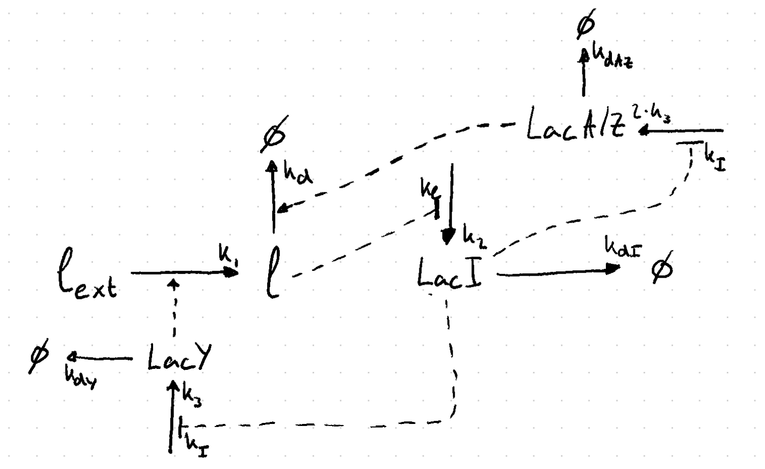

Figure 2: Reaction diagram of the Lac operon model

These set of reactions can be mathematically modelled as follows:

$$\dot{l} = k_1 l_{ext} LacY - l k_d LacAZ \tag{1}$$

$$La\dot{c}I = \frac{k_2}{1 + (\frac{l}{K_l})^m} - k_{dI} LacI \tag{2}$$

$$La\dot{c}Y = \frac{k_3}{1 + (\frac{LacI}{K_I})^n} - k_{dY} LacY \tag{3}$$

$$La\dot{c}AZ = \frac{2k_3}{1 + (\frac{LacI}{K_I})^n} - k_{dAZ} LacAZ \tag{4}$$

c.

We now simulate this mathematical model on MATLAB. To do so, we set the initial conditions (of LacY, LacA, LacZ, LacI and intracellular lactose) at arbitrary values. Additionally, we start the simulation with no extracellular lactose input. The system then approaches a steady state. At t = 50, extracellular lactose is input into the system ($l_{ext} = 1$). The overall evolution is shown in the graph below.

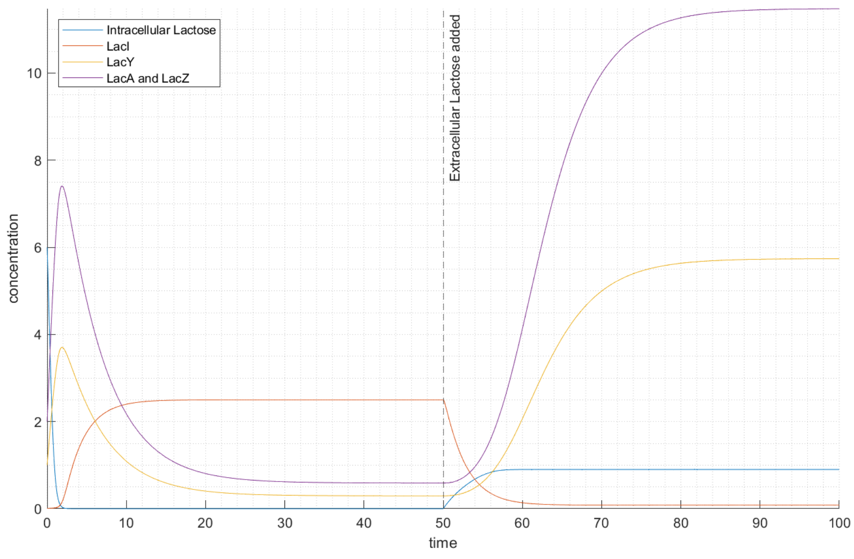

Figure 3: Evolution over time of the Lac operon response to extracellular lactose

In the figure above, we see that the initial intracellular lactose is rapidly degraded due to the large spike in LacA, LacZ and LacY concentrations. Once the lactose is metabolised, LacI is expressed and ensures that LacA, LacZ and LacY are no longer produced, leading the system to a steady "off" state. When extracellular lactose is added (and remains at a constant level of $l_{ext} = 1$), we see that LacI expression diminishes and no longer inhibits LacA, LacZ and LacY, effectively leading the system to a new steady "on" state, allowing it to continuously metabolise the entering lactose.

Note

From this point on, to simplify the analysis, the model will be reduced to the following mathematical equations.

$$\dot{l} = \beta l_{ext} LacY - \gamma l \tag{5}$$

$$\dot{LacY} = \delta + p\frac{l^4}{l^4 + l_0^4} - \sigma LacY \tag{6}$$

d.

By augmenting the system using $\dot{x} = f(x,t,p)$ and $\dot{S} = AS + B$, where A is the Jacobian of the system, and B is the matrix of partial derivatives of f with respect to the vector of parameters $(\beta, \gamma, \delta, \sigma, l_0, p)$, we can solve the $\dot{S}$ differential equation to find the sensitivity matrix. By normalising it, we obtain:

$$S = \begin{pmatrix} 1.0000 & -1.000 & 1.0000 & -1.0000 & 0.0039 & -0.0155 \\ 0.0620 & -0.0620 & 1.0000 & -1.0000 & 0.0039 & -0.0155 \end{pmatrix} \tag{7}$$

High and low absolute value of the element S_{ij} means that the parameter j has a respectively large and small influence on the variable i of the system. Moreover, since the sensitivity matrix is normalised, the largest impact possible is represented by the value ± 1.0000, the sign of the value corresponding to the direction of the impact. As a note, we used l_{ext}, l and $LacY = 1$ to compute the A and B matrices.
Interestingly, we can see that several parameters reach the largest impact possible on the two variables of the Lac operon system. Indeed, β, γ, δ and σ have the largest impact on the rate of formation of intracellular lactose. In addition, δ and σ also have the same impact, but on the rate of formation of LacY. We also notice that the initial concentration of intracellular lactose (l_0) has the smallest impact on the overall system.

e.

The nullclines of the model, with $l_{ext} = 2.5$ are shown in the figure below.

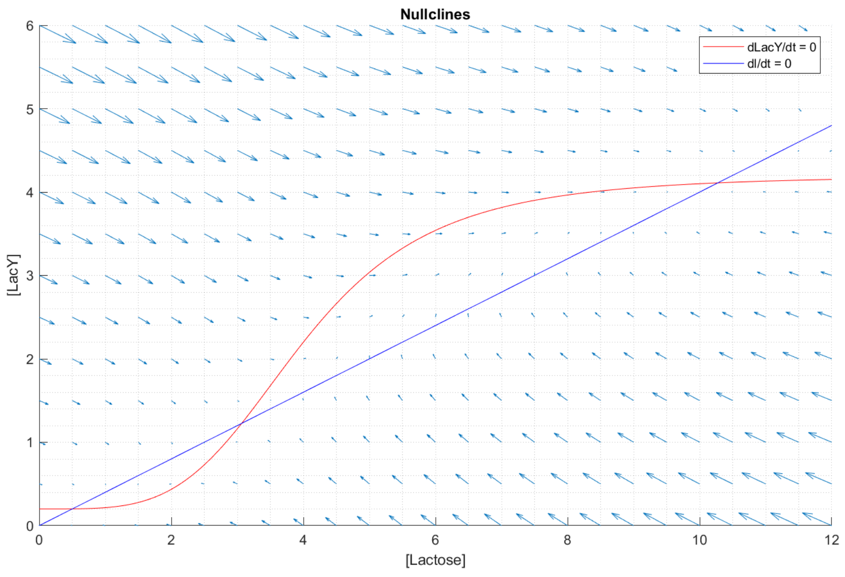

Figure 4: Nullclines of the reduced Lac operon model

In particular, we observe that the nullclines intersect in 3 distinct points, representing 3 different steady states, which are either stable or unstable.

f.

By examining equation (6), we see that as $LacY \to \infty$, with a reasonably low concentration of lactose, $\dot{LacY} < 0$. This implies that above the LacY nullcline, LacY is decreasing. This is also shown by the vector field of figure 4.

g.

By examining equation (5), as in f., as $l \to \infty$, with a reasonably low concentration of LacY, $\dot{l} < 0$. This implies that to the right of the l nullcline, l is decreaing. Again, this result is shown by the vector field of figure 4.

Results from f. and g., notably the vector field seen in figure 4, allow to qualitatively show the stability of the three different steady states. Indeed, the first and third steady states ($(l, LacY) = (0.22, 0.48)$ and $(l, LacY) = (4.10, 10.27)$), are surrounded by vectors spiralling towards them. This shows that these steady states are stable foci. The other steady state ($(l, LacY) = (1.22, 3.07)$) is surrounded by outward vectors; this is an unstable saddle point of the system.

h.

The minimal model analysed above is simulated in MATLAB (see Additional Resources for scripts). The temporal evolution are simulated for different initial conditions ($l = 8, LacY = 3; l = 3.2, LacY = 1.3; l = 3, LacY = 1.2; l = 2, LacY = 1$), with $l_{ext} = 2.5$. The results are shown in section i.

i.

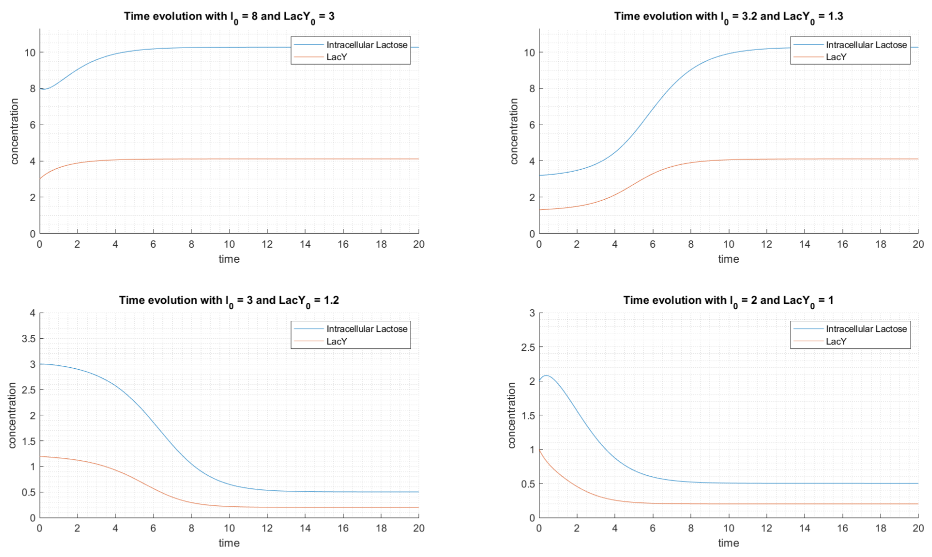

Figure 5: Temporal evolution of the minimal Lac operon model for different initial conditions

Interestingly, wee see here the different stable steady states mentioned above. Indeed, depending on the initial conditions, the system will converge either to $(l, LacY) = (0.22, 0.48)$ or $(l, LacY) = (4.10, 10.27)$, depending on which stable focus is mathematically closer.

j., k., l.

To investigate how the final value of LacY varies as a function of l_{ext}, we generate the following bifurcation diagram (MATLAB script in Additional Resources).

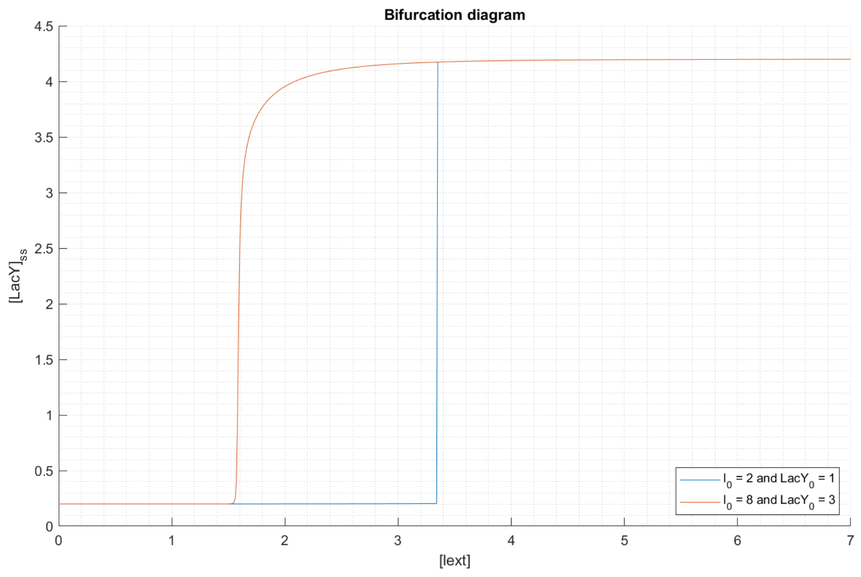

Figure 6: Bifurcation diagram of the Lac operon response

In particular, we observe bistability between the two critical values of $l_{ext} = 1.51$ and $l_{ext} = 3.36$. Between these two values, the steady state value of LacY depends on the initial conditions l_0 and $LacY_0$.

m.

Biologically, this bistability ensures that the bacteria don't toggle uncontrollably between two different genetic states at intermediary values of extracellular lactose. Indeed, we see in Figure 6 that when extracellular lactose is increasing, the bacteria has a "buffer" region before it jumps to a high production of LacY (blue line). On the contrary, when extracellular lactose is decreasing, the bacteria again crosses the buffer region in the other direction, before stopping the high production of LacY (orange line). In this manner, the bacteria doesn't switch between states at a specific l_{ext} in both directions, which would cause many fluctuations.

2 Deterministic Parameter Estimation

Knowing the mathematical model for a biological system, and recorded data (considered to be perfect, without noise), we will estimate the parameters of the model:

$$\dot{Act} = k_1 s + k_2 y_p - k_3 Act \tag{8}$$

$$\dot{y_p} = \frac{(y_T - y_p) k_4 Act}{k_{m4} + (y_T - y_p)} - \frac{y_p k_5 E}{k_{m5} + y_p} \tag{9}$$

a.

We prove mathematically that the system is a positive feedback by computing the Jacobian of the system. In particular, to investigate feedback, we are interested in the terms $\frac{d\dot{Act}}{dy_p}$ and $\frac{d\dot{y_p}}{dAct}$. By computing these derivatives, we find:

$$\frac{d\dot{y_p}}{dAct} = \frac{(y_T - y_p) k_4}{k_{m4} + (y_T - y_p)} \tag{10}$$

$$\frac{d\dot{Act}}{dy_p} = k_2 \tag{11}$$

As k_2, k_4 and k_{m4} are positive constants, and the term $y_T - y_p$ will always be positive as there can never be more phosphorylated y_p than the total y_T, both feedback terms are positive, proving that the system is a positive feedback system.

b.

As $\dot{Act}$ and $\dot{y_p}$ are rates, their units are *Concentration/Time*. Thus, the units of the parameters are the following:

- Concentrations (for example in M): s, k_{m4} and k_{m5}
- Time^{-1} (for example in s^-1): k_1, k_2, k_3, k_4 and k_5

c.

As in the previous sensitivity analysis, we augment the system using $\dot{x} = f(x, t, p)$ and $\dot{S} = AS + B$, where the matrix A is the Jacobian matrix, and the B matrix is the partial derivatives of f with respect to the vector of parameters $(s, k_1, k_2, k_3, k_4, k_5, k_{m4}, k_{m5})$, solve the $\dot{S}$ equation and normalise the obtained sensitivity matrix to find:

$$S = \begin{pmatrix} 0.3960 & 0.0248 & 0.2970 & -0.3960 & 0.1770 & -0.3575 & -0.1265 & 0.5500 \\ 0.6885 & 0.0430 & 0.5164 & -0.6885 & 0.3219 & -0.6500 & -0.2299 & 1.0000 \end{pmatrix} \tag{12}$$

By looking at the sensitivity matrix, we can conclude that the parameter k_{m5} has the largest impact of the overall system, specifically on the rate of formation of y_p. We also notice that k_1 is the parameter that impact the overall system the least.

d.

The true data is shown below.

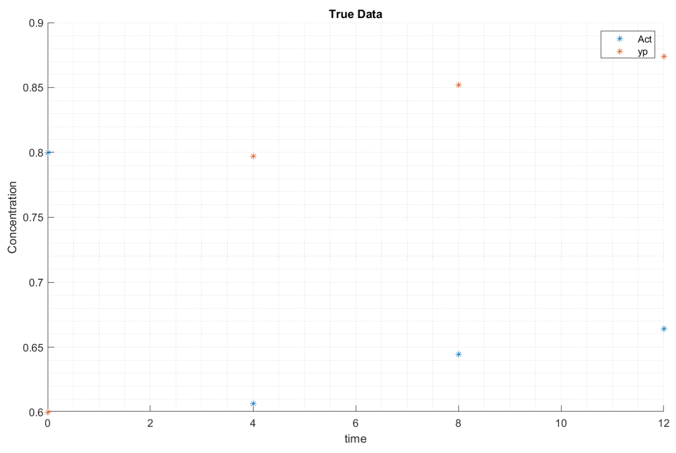

Figure 7: "Recorded data" used for the parameter estimation

e.

We are interested in minimising the residual between the true data and model using the non linear least square method $lsqnonlin(f(x))$ provided in MATLAB, where $f(x)$ is the function to be minimised in the least square's sense. The function to be provided to $lsqnonlin$ is therefore $R = residual(b)$, where $residual(b) = exp - Y$, with exp being the true data and Y the approached estimate.

f.

With all parameters known, except for s, we estimate it using the least squares method described above. To do so, we set the lower and upper bounds of s to be 0.05M and 0.8M respectively. In 8 iterations, the MATLAB script converges to an estimate $\hat{s} = 0.099916M$, while the actual value was $s = 0.1M$. With this estimation, we obtain the following results:

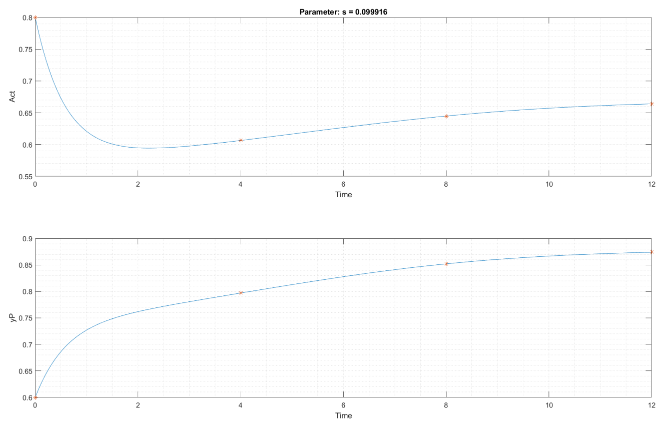

Figure 8: Model evolution over time with estimated s

In particular, we observe that the estimation of s yields a very close simulation to the actual "recorded" data. Indeed, on the last iteration, the residual compared to the true data is only of $s - \hat{s} = 8.4e - 5M$. This is expected as the true data has no noise: it is deterministic. With such a system, we await a very low R value: indeed, $R = SSE = 1.3274e - 07$.

g.

We now assume that 3 parameters are unknown: s, k_{m4} and k_{m5}. For this simulation, we set the lower and upper bounds of s to be 0.05M and 0.8M. For both k_{m4} and k_{m5}, these boundaries are 0.05M and 0.1M. By running the parameter estimating script (given in Additional Resources), in 21 iterations the process converges to the following estimates: $\hat{s} = 0.1M$, $\hat{k}_{m4} = 0.0500305M$ and $\hat{k}_{m5} = 0.05M$. Using these estimates, we obtain the still extremely precise model below, with $R = SSE = 1.2630e - 07$, again because the system is deterministic.

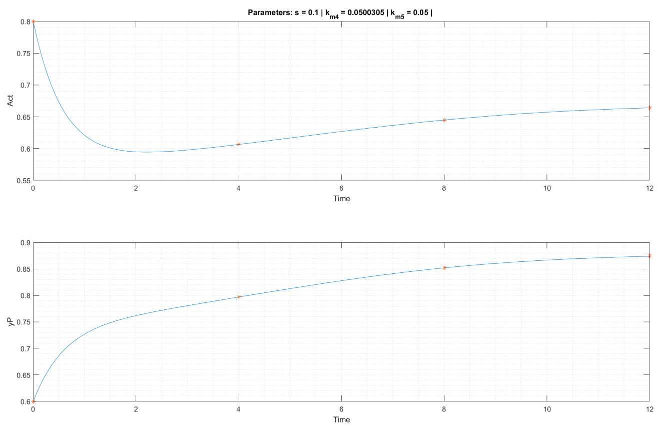

Figure 9: Model evolution over time with estimated s, k_{m4} and k_{m5}

Firstly, we notice that the number of iterations increases with the number of estimates to compute. However, compared to the real values of $s = 0.1M$, $k_{m4} = 0.05M$ and $k_{m5} = 0.05M$, we still obtain very accurate estimates (the residuals being $s - \hat{s} = 0M$, $k_{m4} - \hat{k}_{m4} = 3.05e - 5M$ and $k_{m5} - \hat{k}_{m5} = 0M$). We notice in particular that with three parameters to estimate, with only 4 data points, the model remains very accurate.

h.

We again attempt to estimate the three same parameters of subsection g. but with wrong bounds: we choose upper and lower bounds for k_{m4} being $0.5M$ and $0.2M$ respectively. In 14 iterations, the process converges to the following result.

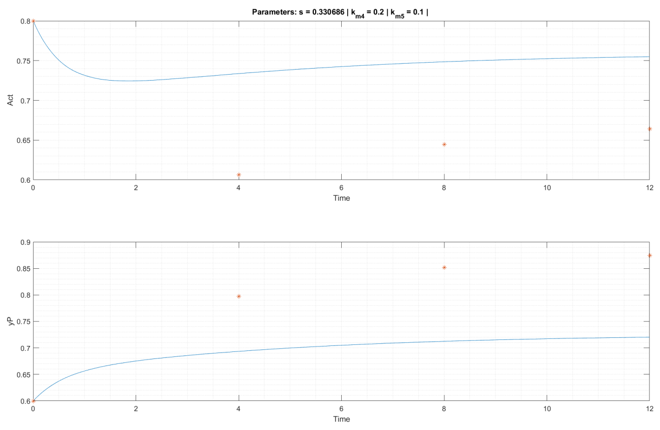

Figure 10: Model evolution over time with estimated s, k_{m4} and k_{m5}, with wrong boundaries for k_{m4}

We immediately notice that the model is simply wrong, with a high value of $R = SSE = 0.0890$ (with residuals being: $s - \hat{s} = 0.230686M$, $k_{m4} - \hat{k}_{m4} = 0.15M$ and $k_{m5} - \hat{k}_{m5} = 0.05$). The importance of the upper and lower bounds is therefore crucial, and it is preferred to have larger boundaries, leading to increased number of iterations, but accurate estimates.

i.

We finally assume that all the 8 parameters are unknown: s, k_1, k_2, k_3, k_4, k_5, k_{m4} and k_{m5}. For this simulation, we set the lower and upper bounds of s to be 0.05M and 0.8M. For all the k_i, these boundaries are 0.1M and 1.5M. For both k_{m4} and k_{m5}, the lower and upper bounds are 0.05M and 0.1M. By running the parameter estimating script, in 89 iterations, the process converges to the following estimates: $\hat{s} = 0.253632M$, $\hat{k}_1 = 0.336042M$, $\hat{k}_2 = 0.603839M$, $\hat{k}_3 = 0.918692M$, $\hat{k}_4 = 0.666263M$, $\hat{k}_5 = 0.642655M$, $\hat{k}_{m4} = 0.061838M$ and $\hat{k}_{m5} = 0.0804506M$. Using these estimates, we obtain the model below.

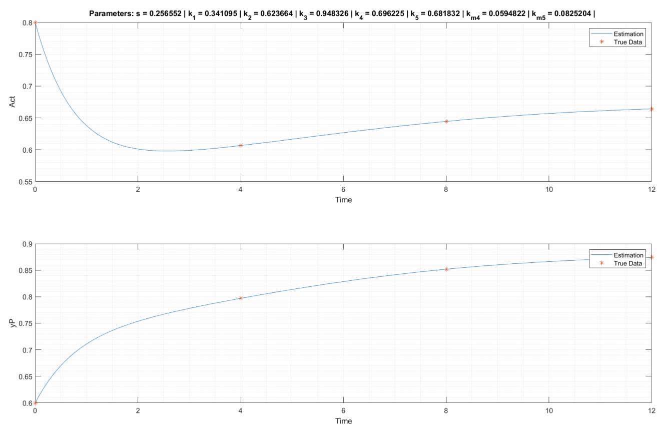

Figure 11: Model evolution over time with all parameters unknown

As discussed previously, we notice that the number of iterations increases with the number of estimates to compute. Indeed, the number of iterations goes from 8 when only one parameter is estimated, to 89 with 8 parameters. In addition, compared to the real values of $s = 0.1$, $k_1 = 1.0$, $k_2 = 0.8$, $k_3 = 1.2$, $k_4 = 1.0$, $k_5 = 1.0$, $k_{m4} = 0.05$ and $k_{m5} = 0.05$, we obtain bad estimates; the residuals are:

- $s - \hat{s} = 0.153632M$
- $k_1 - \hat{k}_1 = 0.663958M$
- $k_2 - \hat{k}_2 = 0.196161M$
- $k_3 - \hat{k}_3 = 0.281308M$
- $k_4 - \hat{k}_4 = 0.333737M$
- $k_5 - \hat{k}_5 = 0.357345M$
- $k_{m4} - \hat{k}_{m4} = 0.011838M$
- $k_{m5} - \hat{k}_{m5} = 0.0304506M$

We notice however that with eight parameters to estimate and only 4 data points, even if the estimated parameters differ a lot from the real values, the model remains close to the measured deterministic data. Indeed, the model has $R = SSE = 7.8935e-08M$.

Such a result can be explained by the small number of data points. Indeed, with only 4 data points and 7 parameters to estimate, there is a lot of freedom to reach the data points with different model estimations. However, with an increased number of data points, the freedom of choice for the estimated parameter values would be reduced to obtain a unique, accurate estimation.

j.

We now run the same estimation (unknown parameters) but with wrong initial conditions for $Act(0)$ and $y_p(0)$ given to the algorithm. By doing so, we obtain the following results:

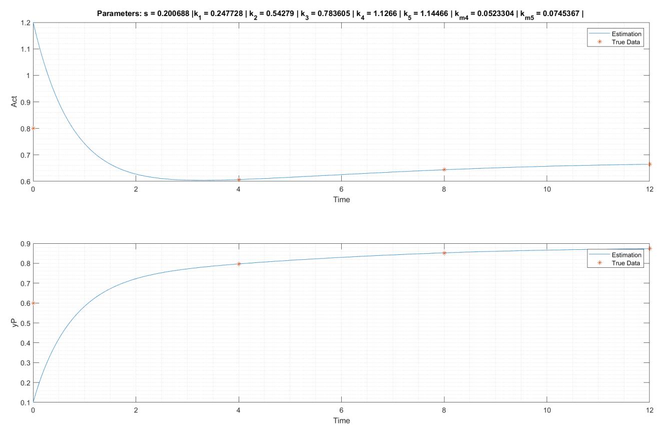

Figure 12: Model evolution over time with all parameters unknown and wrong initial conditions

We firstly notice that the model "obeys" the wrong initial conditions given, and therefore has a large $R = SSE = 0.41$. Thereafter, it still approaches the data points very well, but with parameter estimations that are far off the real values. As explained above, this is due to the fact that with many parameters to estimate, the algorithm has a large freedom to approach a low number of fixed data points. We therefore can conclude that when many parameters are to be estimated and a low amount of data points are available, it is difficult to obtain accurate estimates, even though the "wrong" estimates also approach the true data.

To further illustrate this point, we again consider only one parameter s to be unknown, with the other parameters set at the actual values, and model the system with wrong initial conditions:

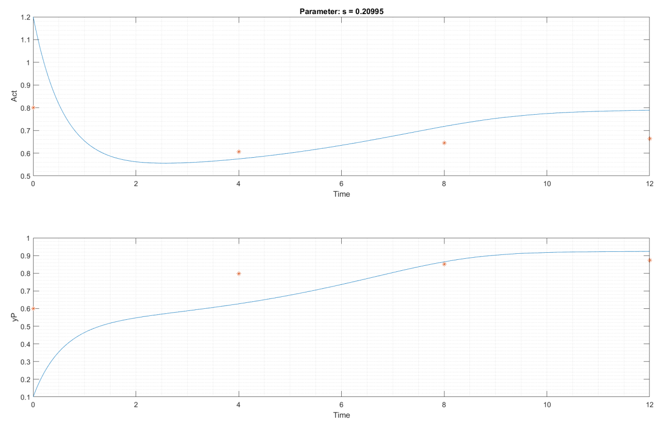

Figure 13: Model evolution over time with estimated s and wrong initial conditions

We indeed see here that a lower degree of freedom of the algorithm and wrong initial conditions yields an estimation $\hat{s} = 0.20995M$ (absolute error of $s - \hat{s} = 0.10995$M) that does not follow the data points as precisely anymore ($R = SSE = 0.4635$).

3 Parameter Estimation with noise

In this final part, we study the same system as the one one before, without dealing with a deterministic system but a noisy one, and study how the parameter estimation behaves.

a.

To corrupt the experimental data, we use the MATLAB function *randn* to model a random noise with a mean μ and standard deviation σ ($\sigma = sqrt(var)$), such as $y_{noise} = y_{true} + \sigma * randn(shape) + \mu$. With a variance of 0.01, we get:

```
data = load("PosFeed_Expdata");
tspan = data.tspan;
exp = data.exp;

% Adding noise (var = 0.01, sd = 0.1)
for j = 1:length(tspan)
    exp(j, 1) = exp(j, 1)+randn(1,1)*0.1;
    exp(j, 2) = exp(j, 2)+randn(1,1)*0.1;
end
```

b.

By estimating the eight parameters using the noisy data with the true initial conditions $Act(0) = 0.8$ and $yp(0) = 0.6$ and the same boundaries and initial guess of the parameter values as problem 2i., we obtain the following estimation, with $R = SSE = 0.03$, in 32 iterations:

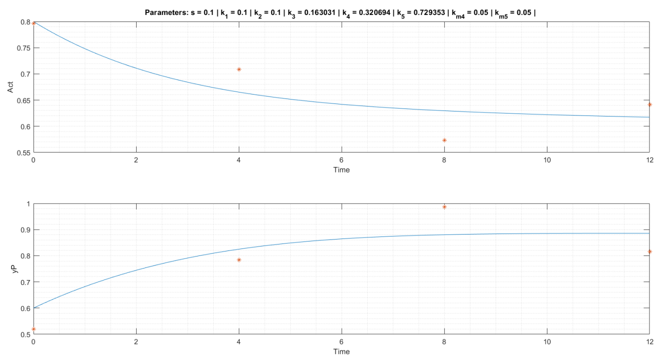

Figure 14: Model evolution over time with all parameters unknown and noisy data with $\sigma = 0.1$

Interestingly, after having run several noisy experiments, we notice that the number of iterations is not fixed. Indeed, the program needed between 45 and 85 iterations to find an estimation. By selecting one of the obtained results, compared to the real values of $s = 0.1$, $k_1 = 1.0$, $k_2 = 0.8$, $k_3 = 1.2$, $k_4 = 1.0$, $k_5 = 1.0$, $k_{m4} = 0.05$ and $k_{m5} = 0.05$, we obtain the following residuals (see values of estimates in the figure above):

- $s - \hat{s} = 0M$
- $k_1 - \hat{k}_1 = 0.9M$
- $k_2 - \hat{k}_2 = 0.7M$
- $k_3 - \hat{k}_3 = 1.036969M$
- $k_4 - \hat{k}_4 = 0.679306M$
- $k_5 - \hat{k}_5 = 0.270647M$
- $k_{m4} - \hat{k}_{m4} = 0M$
- $k_{m5} - \hat{k}_{m5} = 0M$

In addition, we notice that both with and without noise, with eight parameters to estimate as well as 4 data points, the estimated parameters differ a lot from the real values. However, conversely to the model without noise that remained close to the measured data, the noisy model does not reach the data points as accurately. Indeed, this is shown by comparing the 2 R values; $R_{noise} = SSE_{noise} = 0.03$ whereas $R = SSE = 7.8935e - 08$

Such a result is explained by the fact that the system without noise is deterministic, whereas by adding noise, it is non-deterministic.

c.

By increasing the variance of the noise to 0.04 ($\sigma = 0.2$), the following estimations are obtained, with $R = SSE = 0.454$, in 89 iteration.

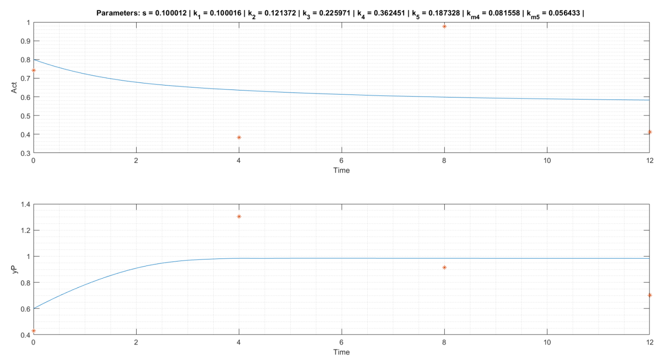

Figure 15: Model evolution over time with all parameters unknown and noisy data with $\sigma = 0.2$

As in the previous noisy experiment, the number of iterations needed to estimate the model is not fixed. In addition, the number of iterations is slightly higher for the noisier system. Indeed, the algorithm needed between 50 and 90 to find an estimation, compared to the 45 to 85 previously observed. By selecting one of the obtained results, compared to the real values of $s = 0.1$, $k_1 = 1.0$, $k_2 = 0.8$, $k_3 = 1.2$, $k_4 = 1.0$, $k_5 = 1.0$, $k_{m4} = 0.05$ and $k_{m5} = 0.05$, we obtain the following residuals (see values of estimates in the figure above):

- $s - \hat{s} = 1.2e - 5M$
- $k_1 - \hat{k}_1 = 0.899984M$
- $k_2 - \hat{k}_2 = 0.678628M$
- $k_3 - \hat{k}_3 = 1.074029M$
- $k_4 - \hat{k}_4 = 0.637549M$
- $k_5 - \hat{k}_5 = 0.812672M$
- $k_{m4} - \hat{k}_{m4} = 0.031558M$
- $k_{m5} - \hat{k}_{m5} = 0.006433M$

As expected, the estimated parameters differ from the real values. Moreover, since the system is noisier, the model with higher noise was less accurate than the previous one. Indeed, the two R values are $R_{noisier} = SSE_{noisier} = 0.454$ whereas $R_{noisy} = SSE_{noisy} = 0.03$.

d.

Process noise is noise due to inherent errors in the modelling of a system. Indeed, biological experiments rely on complex and living systems which will never reproduce identical experiments each time they are run. This leads to differences compared to mathematical models, which will always have a determined response. Therefore, process noise can for example be considered as estimated parameters or missing dynamics in the mathematical description of the model, which can lead to growing errors compared to the true system as the differential equations of the model are solved. To account for this type of noise, we can add additional parameters modelling process noise. For example, to model process noise relating the parameter k_1 of equation (7), we can write:

$$\dot{Act} = (\alpha k_1 + \epsilon)s + k_2 y_p - k_3 Act \tag{13}$$

4 Additional Resources

Script for the Lac operon modelling (1.a-c)

```
clc
clear

%% Lactose input at lext = 0; arbitrary initial conditions
% Initial cdts and params
x0 = init_cond();
par = param();
% Timespan for simulation
tspan = [0 50];
options = [];

[t1,x] = ode45(@diff_eq,tspan,x0,options,par);
l = x(:, 1);
LacI = x(:, 2);
LacY = x(:, 3);
LacAZ = x(:, 4);

% Simulation of the Lac operon without lactose input

% figure(1)
% plot(t1, l, t1, LacI, t1, LacY, t1, LacAZ)
% legend('Intracellular Lactose', 'LacI', 'LacY', 'LacA and LacZ')
% xlabel('time')
% ylabel('concentration')

%% Lactose input at lext = 1; initial cdts are steady state of
   previous
x0 = [l(end), LacI(end), LacY(end), LacAZ(end)]; %x0 = [lin; LacI0;
    LacY0; LacAZ0];
par.lext = 1;

[t2,x] = ode45(@diff_eq,tspan,x0,options,par);
l_1 = x(:, 1);
LacI_1 = x(:, 2);
LacY_1 = x(:, 3);
LacAZ_1 = x(:, 4);

% Simulation of the lac operon with lactose input

% figure(2)
% plot(t2, l_1, t2, LacI_1, t2, LacY_1, t2, LacAZ_1)
% legend('Intracellular Lactose', 'LacI', 'LacY', 'LacA and LacZ')
% xlabel('time')
% ylabel('concentration')

%% Merge graphs

l_merge = cat(1,l,l_1);
LacI_merge = cat(1,LacI,LacI_1);
LacY_merge = cat(1,LacY,LacY_1);
LacAZ_merge = cat(1,LacAZ,LacAZ_1);
t_merge = cat(1,t1,t1(end)+t2);

% Plotting entire simulation
figure(3);
hold on
grid minor
```

```
plot(t_merge, l_merge, t_merge, LacI_merge, t_merge, LacY_merge,
   t_merge, LacAZ_merge)
xline(t1(end),Label = "Extracellular Lactose added",LineStyle="--")
legend('Intracellular Lactose', 'LacI', 'LacY', 'LacA and LacZ','',
   Location = 'northwest')
xlabel('time')
ylabel('concentration')
ylim([0,inf])
set(gcf,'Position',[100 100 1000 600])
saveas(gcf,'Results/complete_evo.png')
hold off

%% Functions
function dxdt = diff_eq(t,x,par)
% Variables
l = x(1);
LacI = x(2);
LacY = x(3);
LacAZ = x(4);

% Differential equations
l_dot = par.k1*par.lext*LacY - l*par.kd*LacAZ;
LacI_dot = par.k2/(1+(l/par.Kl)^par.n)-par.kdI*LacI;
LacY_dot = par.k3/(1+(LacI/par.KI)^par.m)-par.kdY*LacY;
LacAZ_dot = 2*par.k3/(1+(LacI/par.KI)^par.m)-par.kdAZ*LacAZ;

dxdt = [l_dot;LacI_dot;LacY_dot;LacAZ_dot];
end


function par = param()
par.k1 = 0.9;       % Absorption rate of lext to l
par.k2 = 1;         % Formation of LacI
par.k3 = 3;         % Formation of LacY, LacA and LacZ
par.kd = 0.5;       % Degradation of l
par.kdI = 0.4;      % Degradation of LacI
par.kdY = 0.2;      % Degradation of LacY
par.kdAZ = 0.2;     % Degradation of LacA and LacZ
par.Kl = 0.03;      % Inhibition of LacI (smaller value -> bigger
   inhinition)
par.KI = 0.05;      % Inhibition of LacY, LacA and LacZ (smaller
   value -> bigger inhinition)
par.lext = 0.0;     % lext constant
par.n = 1;          % LacI_dot Hill coefficient
par.m = 1;          % LacY_dot and LacAZ_dot Hill coefficient
end

function x0 = init_cond()
lin = 6;
LacI0 = 0.01;
LacY0 = 1;
LacAZ0 = 2*LacY0;

x0 = [lin; LacI0; LacY0; LacAZ0];
end
```

Script for the Lac operon modelling (1.d-i)

```
clc
clear

%% Minimal Model Simulation
% Initial cdts and params
x0 = init_cond();
l_list = [8 3.2 3 2];
LacY_list = [3 1.3 1.2 1];
lext_list = 0:0.01:7;
par = param();

% Timespan for simulation
dt     = 1e-2 ;
tlast = 20.000 ;
iterations = fix(tlast/dt) ;
tspan = dt*(0:iterations-1) ;
options = [];

%% Nullclines
plot_NullCline(6,12);
vectorField(par,6, 12);
title('Nullclines')
xlabel('[LacY]')
ylabel('[Lactose]')
legend('dLacY/dt = 0','dl/dt = 0')
view([90 -90])
set(gcf,'Position',[100 100 1000 600])
saveas(gcf,'Results/nullclines.png')
hold off

%% Temporal evolution with varying init cdt
figure()
hold on
grid minor
for ii = 1:min(length(l_list), length(LacY_list))
    subplot(min(length(l_list), length(LacY_list))/2,2,ii)
    [last_l, last_LacY] = plot_t_evo(tspan, set_x0(l_list(ii),
       LacY_list(ii)), options, par,1);
    ylim([0 max(max(last_l, last_LacY), max(l_list(ii), LacY_list(
       ii)))+1])
end

set(gcf,'Position',[10 10 1500 900])
saveas(gcf,'Results/temporal_evo.png')
hold off

%% Bifurcation
final_LacY83 = [];
final_LacY21 = [];

for i = 1:length(lext_list)
    par.lext = lext_list(i);

    [final_l_83, final_LacY_83] = plot_t_evo(tspan, set_x0(8, 3),
       options, par,0);
    [final_l_21, final_LacY_21] = plot_t_evo(tspan, set_x0(2, 1),
       options, par,0);

    final_LacY83 = [final_LacY83 final_LacY_83];
    final_LacY21 = [final_LacY21 final_LacY_21];

end

% Values of lext for which bistability is no longer observed
for iii = 1:length(lext_list)
    if iii~=1 && abs(final_LacY83(iii)-final_LacY21(iii)) < 0.001
       && abs(final_LacY83(iii-1)-final_LacY21(iii-1)) > 0.001
        largest_lext = lext_list(iii)
    elseif iii~=length(lext_list) && abs(final_LacY83(iii)-
       final_LacY21(iii)) < 0.001 && abs(final_LacY83(iii+1)-
       final_LacY21(iii+1)) > 0.001
        smallest_lext = lext_list(iii)
    end
end

% Plotting of bifurcation diagram
figure()
hold on
grid minor
plot(lext_list, final_LacY21, lext_list, final_LacY83)
legend("l_0 = 2 and LacY_0 = 1", "l_0 = 8 and LacY_0 = 3", Location
     = "southeast")
title("Bifurcation diagram")
ylabel("[LacY]_s_s")
xlabel("[lext]")
set(gcf,'Position',[100 100 1000 600])
saveas(gcf,'Results/bifurcation.png')

%% Sensitivity analysis
[t,S] = ode45(@S_dot,tspan,zeros([14,1]));
S = reshape(S.',2,7,[]);
S(:, :, end)

%% Functions
function dsdt = S_dot(t, S)
beta = 1;
gamma = 1;
delta = 0.2;
sigma = 1;
l0 = 4;
p= 4;
lext = 1;

syms l LacY

l_dotA = beta*lext*LacY-gamma*l;
LacY_dotA = delta+p*l^4/(l^4+l0^4)-sigma*LacY;

A = matlabFunction(jacobian([l_dotA,LacY_dotA],[l LacY]));

syms betaa lextt gammaa deltaa sigmaa pp l00
l=1;
LacY=1;
l_dotB = betaa*lextt*LacY-gammaa*l;
LacY_dotB = deltaa+pp*l^4/(l^4+l00^4)-sigmaa*LacY;

B = matlabFunction(jacobian([l_dotB,LacY_dotB],[betaa lextt gammaa
   deltaa sigmaa pp l00]));

 ds = A(1)*reshape(S, [2, 7]) + B(1, 4, 1, 4);
 dsdt = reshape(ds, [14, 1]);
end

function nl = plot_NullCline(x_lim, y_lim)
par = param();

syms l LacY
LacY_dot = par.delta+par.p*l^4/(l^4+par.l0^4)-par.sigma*LacY;
l_dot = par.beta*par.lext*LacY-par.gamma*l;

figure()
hold on
grid minor
nl1=ezplot(LacY_dot,[0,x_lim,0,y_lim]);
set(nl1,color = 'r');
nl2=ezplot(l_dot,[0,x_lim,0,y_lim]);
set(nl2,color = 'b');
nl=[nl1,nl2];
end

function vectorField(par, x_lim, y_lim)
[LacY ,l] = meshgrid(0:0.5:x_lim, 0:0.5:y_lim);

LacY_dot = par.delta+par.p.*l.^4./(l.^4+par.l0.^4)-par.sigma.*LacY;
l_dot = par.beta.*par.lext.*LacY-par.gamma.*l;

quiver(LacY, l, LacY_dot, l_dot)
xlim([0 x_lim])
ylim([0 y_lim])
end

function dxdt = diff_eq(t,x,par)
% Variables
l = x(1);
LacY = x(2);

% Differential equations
l_dot = par.beta*par.lext*LacY-par.gamma*l;
LacY_dot = par.delta+par.p*l^4/(l^4+par.l0^4)-par.sigma*LacY;

dxdt = [l_dot;LacY_dot];
end

function par = param()
par.beta = 1;
par.gamma = 1;
par.delta = 0.2;
par.sigma = 1;
par.l0 = 4;
par.p = 4;
par.lext = 2.5;
end

function x0 = init_cond()
l_init = 4;
LacY0 = 1;

x0 = [l_init; LacY0];
end

function x0 = set_x0(l_init, LacY0)
```

```
x0 = [l_init, LacY0];
end

function [last_l, last_LacY] = plot_t_evo(tspan, x0, options, par,
   want_plot)
[t,x] = ode45(@diff_eq,tspan,x0,options,par);
l = x(:, 1);
LacY = x(:, 2);

if want_plot == 1
    hold on
    grid minor
    plot(t, l, t, LacY)
    legend('Intracellular Lactose', 'LacY')
    xlabel('time')
    ylabel('concentration')
    title("Time evolution with l_0 = "+x0(1)+" and LacY_0 = "+x0(2)
        )
end

last_LacY = LacY(end);
last_l = l(end);
end
```

Script for the deterministic single parameter estimation (2.f,j)

```
function results =single_parameter_estimation(initParams)
% Estimation of the parameter s by non-linear least squares
   optimization
data = load("PosFeed_Expdata");
tspan = data.tspan;
exp = data.exp;

opts = optimset('TolFun', 1e-12, 'TolX', 1e-12, 'MaxIter', 150, '
   Diagnostics', 'off', 'Display', 'iter');
initParams=[0.8];

loBound = [0.05];
upBound = [0.8];

lsqnonlin(@residual,  log10 (initParams), log10 (loBound), log10 (
   upBound), opts);

function R = residual(b)
a = 10.^b;
[T,Y]= reactionsolve(a);
residual = zeros([length(tspan), 2]);
for i = 1:length(T)
    for ii = 1:length(tspan)
        if T(i) == tspan(ii)
            residual(ii, 1) = exp(ii, 1)-Y(i, 1);
            residual(ii, 2) = exp(ii, 2)-Y(i, 2);
        end
    end
end
R = residual(:);
SSE = sum(R(:).^2)
results=a;

subplot(2,1,1);
plot(T,Y(:,1));
hold on;
grid minor
plot(tspan(:,1),exp(:,1),'*');
xlabel('Time');
ylabel('Act');
hold off;
str = "Parameter: "+string(a);
title(str);

subplot(2,1,2);
plot(T,Y(:,2));
hold on;
grid minor
plot(tspan(:,1),exp(:,2),'*');
xlabel('Time');
ylabel('yP');
hold off;

hold off;
drawnow;
end

function [T,Y] = reactionsolve(a)
s=a(1);
```

```
% exp(tspan = 0)
x0 = [1.2; 0.1];
t = 0:0.01:12;
[T,Y] = ode45(@reaction, t, x0, []);

function dx = reaction(t,x)
Act=x(1);
yp=x(2);

par.ytot=1;
par.E=0.5;

par.k1=1;
par.k2=0.8;
par.k3=1.2;
par.k4=1;
par.k5=1;
par.km4=0.05;
par.km5=0.05;

Act_dot = par.k1*s+par.k2*yp-par.k3*Act;
yp_dot = par.k4*Act*(par.ytot-yp)/(par.km4+par.ytot-yp)-par.k5*par.
   E*yp/(par.km5+yp);

dx = zeros(2,1);
dx(1)=Act_dot;
dx(2)=yp_dot;

end

end

end
```

Script for the deterministic estimation of three parameters (2.g,h)

```
function results = three_parameter_estimation(initParams)
% Estimation of the parameters s, km4 and km5 by non-linear least
   squares optimization
data = load("PosFeed_Expdata");
tspan = data.tspan;
exp = data.exp;

opts = optimset('TolFun', 1e-12, 'TolX', 1e-12, 'MaxIter', 150, '
   Diagnostics', 'off', 'Display', 'iter');
initParams=[0.8;0.05;0.05];

loBound = [0.1 0.05 0.05];
upBound = [0.8 0.1 0.1];

lsqnonlin(@residual,  log10 (initParams), log10 (loBound), log10 (
   upBound), opts);

function R = residual(b)
a = 10.^b;
[T,Y]= reactionsolve(a);
residual = zeros([length(tspan), 2]);
for i = 1:length(T)
    for ii = 1:length(tspan)
        if T(i) == tspan(ii)
            residual(ii, 1) = exp(ii, 1)-Y(i, 1);
            residual(ii, 2) = exp(ii, 2)-Y(i, 2);
        end
    end
end
R = residual(:);
SSE = sum(R(:).^2)
results=a;

subplot(2,1,1);
plot(T,Y(:,1));
hold on;
grid minor
plot(tspan(:,1),exp(:,1),'*');
xlabel('Time');
ylabel('Act');
hold off;
str = "Parameters: " + sprintf('%g | ', a);
title(str);

subplot(2,1,2);
plot(T,Y(:,2));
hold on;
grid minor
plot(tspan(:,1),exp(:,2),'*');
xlabel('Time');
ylabel('yP');
hold off;

hold off;
drawnow;
end

function [T,Y] = reactionsolve(a)
s=a(1);
```

```
km4=a(2);
km5=a(3);

% exp(tspan = 0)
x0 = [0.8; 0.6];
t = 0:0.01:12;
[T,Y] = ode45(@reaction, t, x0, []);

function dx = reaction(t,x)
Act=x(1);
yp=x(2);

par.ytot=1;
par.E=0.5;

% Values of fixed parameters
par.k1 = 1;
par.k2 = 0.8;
par.k3 = 1.2;
par.k4 = 1;
par.k5 = 1;

Act_dot = par.k1*s+par.k2*yp-par.k3*Act;
yp_dot = par.k4*Act*(par.ytot-yp)/(km4+par.ytot-yp)-par.k5*par.E*yp
   /(km5+yp);

dx = zeros(2,1);
dx(1)=Act_dot;
dx(2)=yp_dot;
end
end
end
```

Script for the deterministic and noisy estimation of eight parameters (2.i,j; 3.b,c)

```
function results = seven_parameter_estimation(initParams)

% Estimation of the parameters by non-linear least squares
   optimization
data = load("PosFeed_Expdata");
tspan = data.tspan;
exp = data.exp;

figure()
hold on
grid minor
plot(tspan,exp,'*')
title('True Data')
legend('Act','yp')
xlabel('time')
ylabel('Concentration')
set(gcf,'Position',[100 100 1000 600])
saveas(gcf,'Results/paramest_data.png')
hold off


opts = optimset('TolFun', 1e-12, 'TolX', 1e-12, 'MaxIter', 150, '
   Diagnostics', 'off', 'Display', 'iter');
initParams=[0.1;0.1;0.1;0.1;0.1; 0.1;0.05;0.05];

% 8 parameters estimated s, k1, k2, k3, k4, k5, km4, km5
loBound = [0.1 0.1 0.1 0.1 0.1 0.1 0.05 0.05];
upBound = [0.8 1.5 1.5 1.5 1.5 1.5 0.1 0.1];

lsqnonlin(@residual,  log10 (initParams), log10 (loBound), log10 (
   upBound), opts);

function R = residual(b)

a = 10.^b;
[T,Y]= reactionsolve(a);
residual = zeros([length(tspan), 2]);
for i = 1:length(T)
    for ii = 1:length(tspan)
        if T(i) == tspan(ii)
            residual(ii, 1) = exp(ii, 1)-Y(i, 1);
            residual(ii, 2) = exp(ii, 2)-Y(i, 2);
        end
    end
end
R = residual(:);
SSE = sum(R(:).^2)
results=a;

subplot(2,1,1);
plot(T,Y(:,1));
hold on;
grid minor
plot(tspan(:,1),exp(:,1),'*');
legend('Estimation', 'True Data')
xlabel('Time');
ylabel('Act');
hold off;
```

```
str = "Parameters: " + sprintf('%g | ', a);
title(str);

subplot(2,1,2);
plot(T,Y(:,2));
hold on;
grid minor
plot(tspan(:,1),exp(:,2),'*');
legend('Estimation', 'True Data')
xlabel('Time');
ylabel('yP');
hold off;

hold off;
drawnow;
end

function [T,Y] = reactionsolve(a)

s=a(1);
k1=a(2);
k2=a(3);
k3=a(4);
k4=a(5);
k5=a(6);
km4=a(7);
km5=a(8);

% exp(tspan = 0)
x0 = [0.8; 0.6];
t = 0:0.01:12;
[T,Y] = ode45(@reaction, t, x0, []);

function dx = reaction(t,x)

Act=x(1);
yp=x(2);

par.ytot=1;
par.E=0.5;

Act_dot = k1*s+k2*yp-k3*Act;
yp_dot = k4*Act*(par.ytot-yp)/(km4+par.ytot-yp)-k5*par.E*yp/(km5+yp
    );

dx = zeros(2,1);
dx(1)=Act_dot;
dx(2)=yp_dot;

end

end

end
```