Modelling of the E. coli Lac operon and parameter estimation

Student numbers: 584833 and 583875

March 2023

1 Modelling of the Lac operon

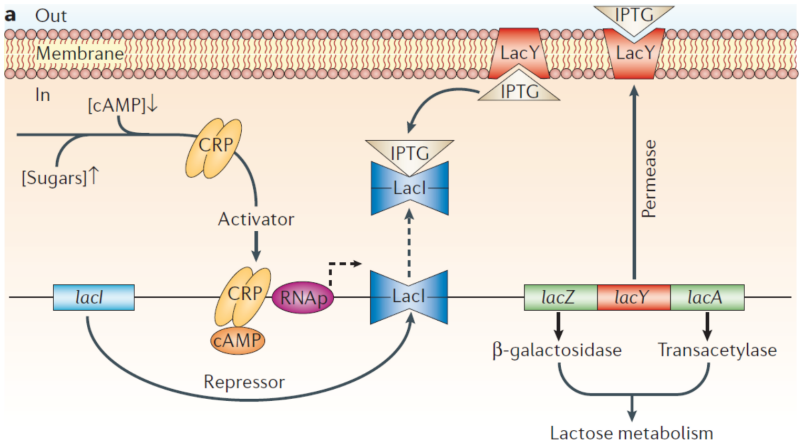

Figure 1: The Lac operon model in E. coli (Smits et al. 2006)

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.

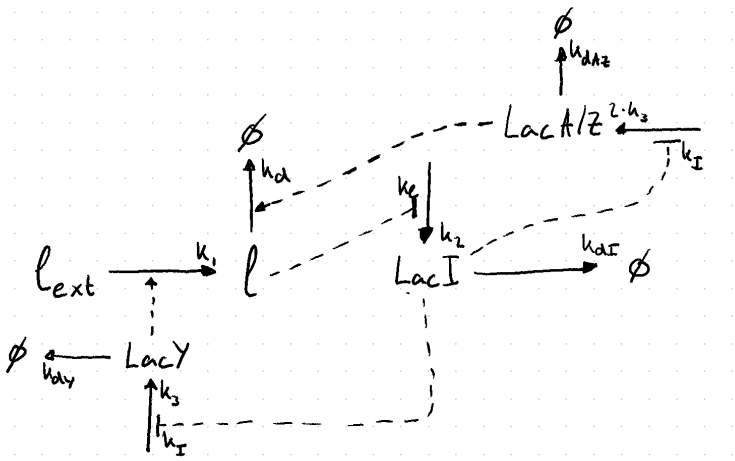

Figure 2: Reaction diagram of the Lac operon model

These set of reactions can be mathematically modelled as follows:

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

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

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

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

c.

We now simulate this mathematical model on MATLAB (see biosys_project.m). 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.

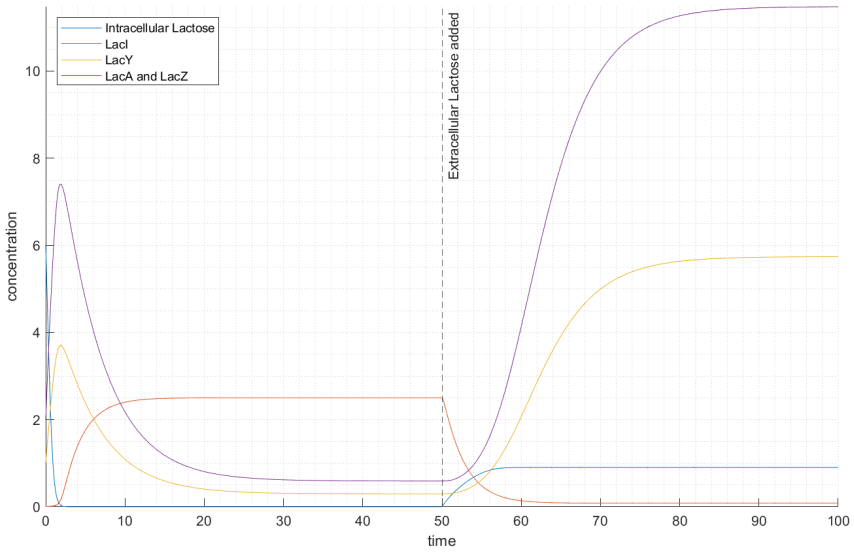

Figure 3: Evolution over time of the Lac operon response to extracellular lactose

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 (see biosys_project_minimal_model.m).

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

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

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 $(\beta, \gamma, \delta, \sigma, l_0, p, l_{ext})$, we can use the Euler method to find the sensitivity matrix. We get $S_{ij} = (f(x_i, t, p_j + \Delta p_j) - f(x_i, t, p_j))/\Delta p_j$, where Δp_j is set as $0.01 p_j$. By normalising the sensitivity matrix, we finally obtain:

$$S = \begin{pmatrix} -0.1989 & -0.2012 & -1.0000 & -0.2012 & -0.0500 & -0.0497 & -0.0796 \\ -0.0009 & -0.0010 & -0.0045 & -0.0014 & -0.0002 & -0.0001 & -0.0004 \end{pmatrix} \tag{7}$$

High and low absolute value of the element S_{ij} means that the parameter j has a respectively large and small influence on the overall system when acting on the variable i of the latter. 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 (rates of formation in our case).

Interestingly, we can see that only one parameter reaches the largest impact possible on the Lac operon system when acting on the variable l. Indeed, δ is the most sensitive parameter of the system when influencing the rate of formation of intracellular lactose. We also notice that the parameters have a smaller impact on the system when they act on $\dot{LacY}$. Indeed, the second row of the sensitivity matrix has a maximum absolute value of 0.0045 whereas the minimum absolute value of the first row is 0.0497. This is expected as the system relies on l^4, and x_1 represents the intracellular lactose rate of formation.

We finally remark that the overall system is least sensitive to the parameter p.

e.

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

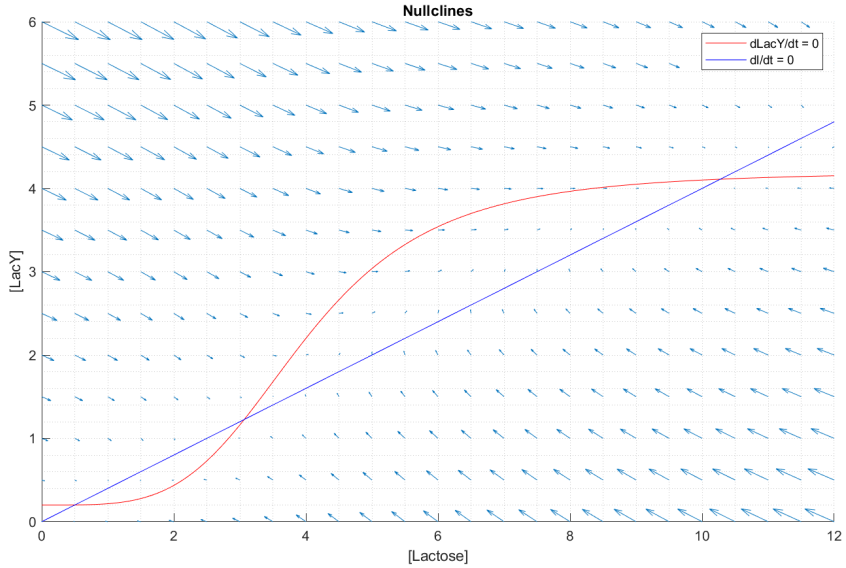

Figure 4: Nullclines of the reduced Lac operon model

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 biosys_project_minimal_model.m). 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.

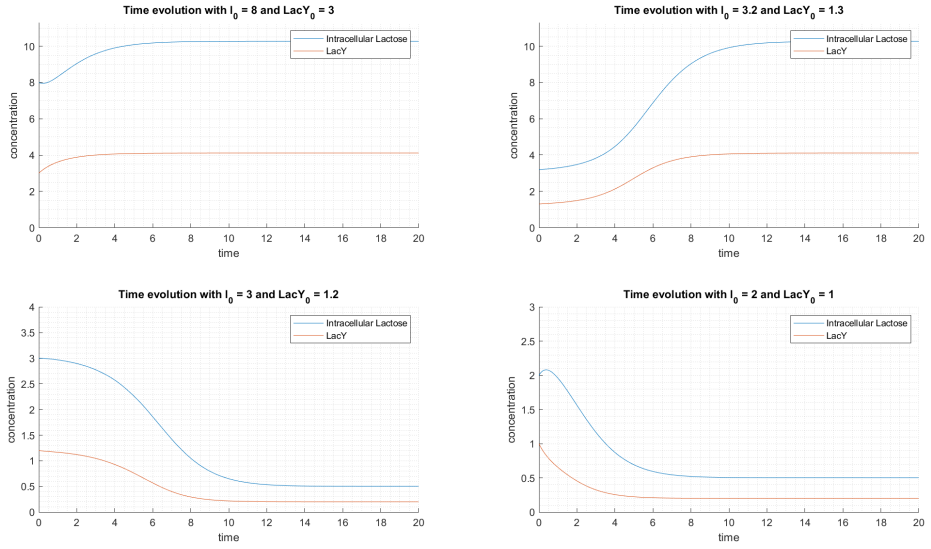

Figure 5: Temporal evolution of the minimal Lac operon model for different initial conditions

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.

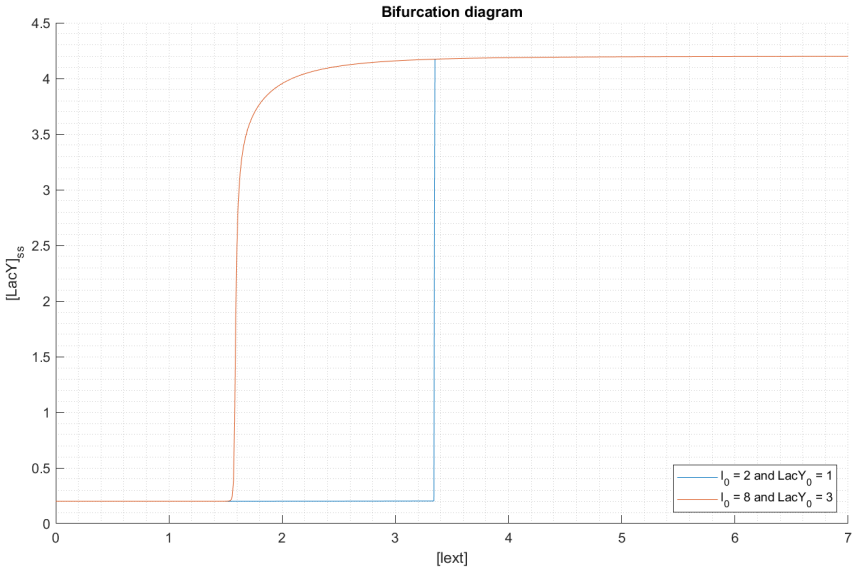

Figure 6: Bifurcation diagram of the Lac operon response

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 $(s, k_1, k_2, k_3, k_4, k_5, k_{m4}, k_{m5})$, use the Euler method and normalise the obtained sensitivity matrix to find:

$$S = \begin{pmatrix} 0.7680 & 0.6144 & 0.7467 & -1.0000 & 0.0264 & -0.0217 & -0.3455 & 0.0153 \\ 0.0324 & 0.0260 & 0.0251 & -0.0394 & 0.0415 & -0.1176 & -0.5287 & 0.1445 \end{pmatrix} \tag{12}$$

By looking at the sensitivity matrix, we can conclude that the parameter k_3 has the largest impact of the overall system, specifically on the rate of formation of Act. We also notice that k_{m5} has the smallest impact on $\dot{Act}$ and k_2 on $\dot{y_p}$.

d.

The true data is shown below.

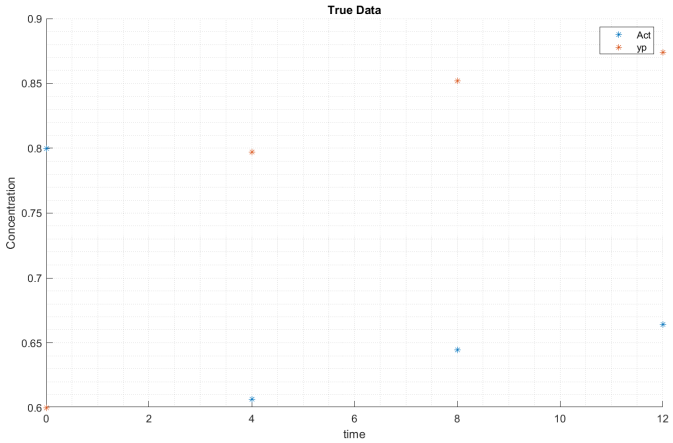

Figure 7: "Recorded data" used for the parameter estimation

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 (see single_parameter_estimation.m) converges to an estimate $\hat{s} = 0.099916M$, while the actual value was $s = 0.1M$. With this estimation, we obtain the following results:

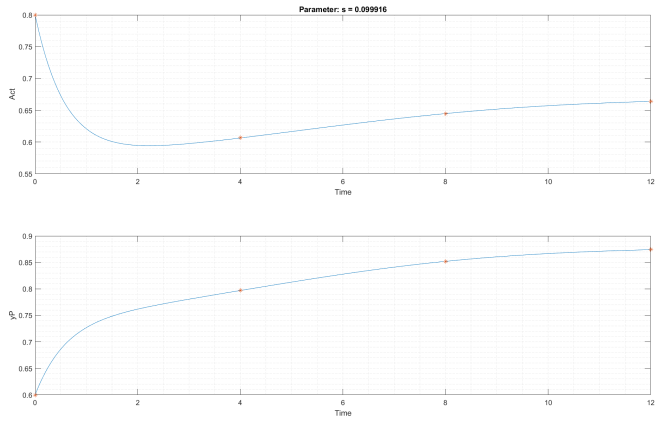

Figure 8: Model evolution over time with estimated s

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 (see three_parameter_estimation.m), 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.

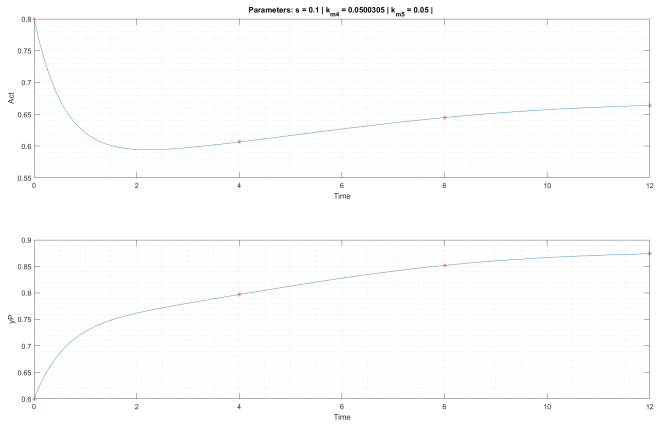

Figure 9: Model evolution over time with estimated s, k_{m4} and k_{m5}

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.

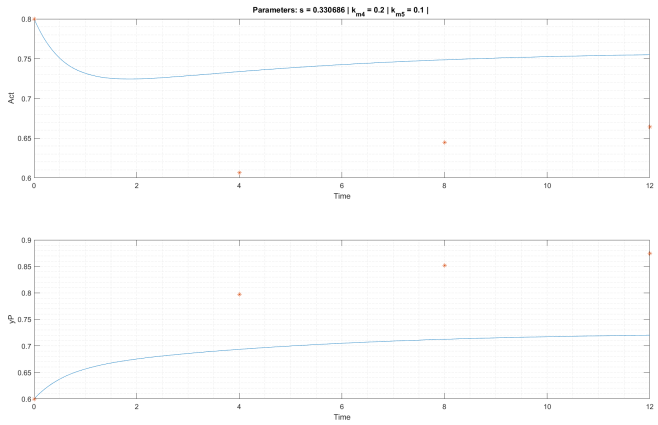

Figure 10: Model evolution over time with estimated s, k_{m4} and k_{m5}, with wrong boundaries for k_{m4}

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 (see eight_parameter_estimation.m), 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.

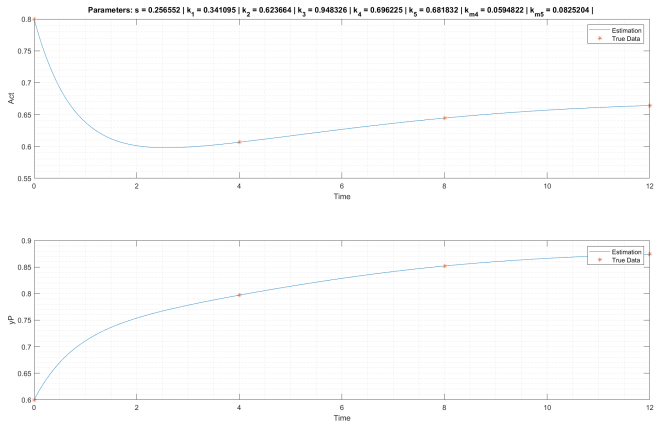

Figure 11: Model evolution over time with all parameters unknown

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 8 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:

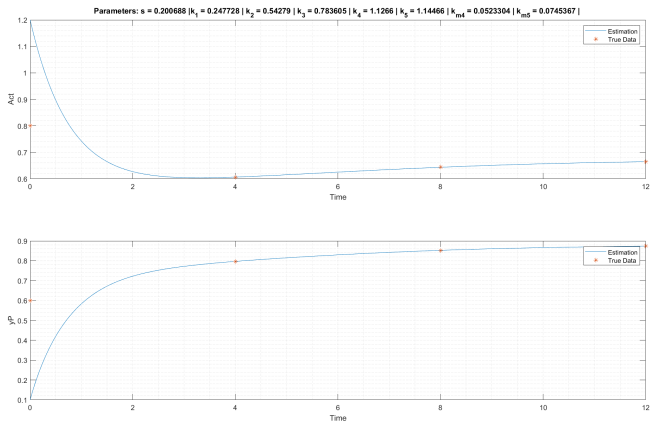

Figure 12: Model evolution over time with all parameters unknown and wrong initial conditions

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:

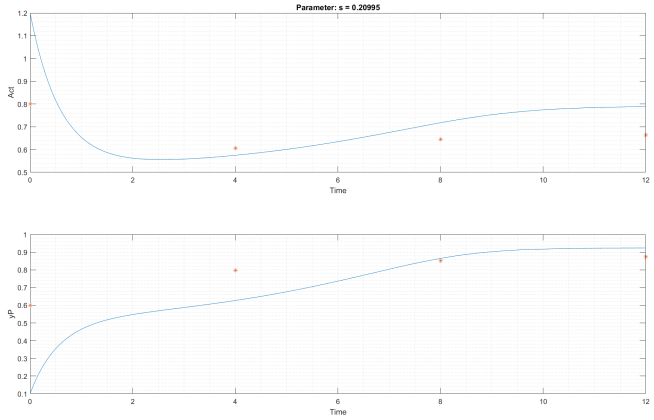

Figure 13: Model evolution over time with estimated s and wrong initial conditions

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 (see see noise_parameter_estimation.m) 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:

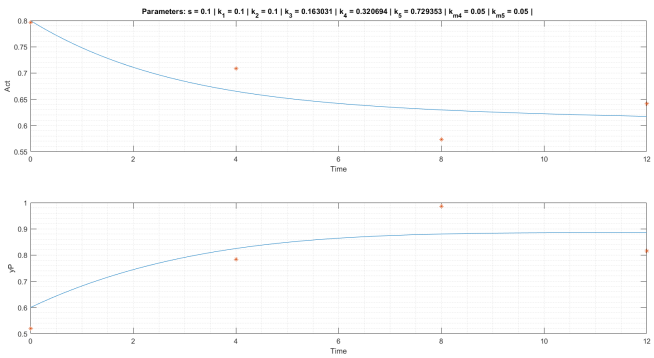

Figure 14: Model evolution over time with all parameters unknown and noisy data with $\sigma = 0.1$

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.

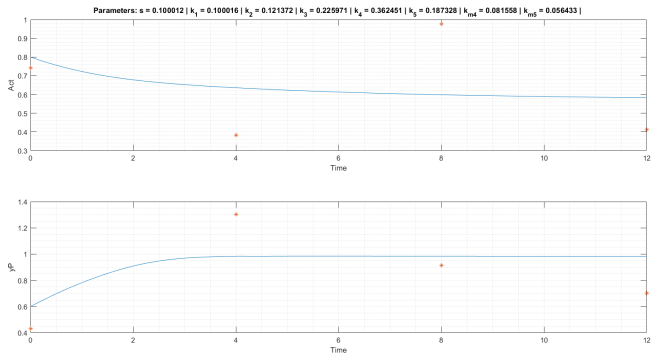

Figure 15: Model evolution over time with all parameters unknown and noisy data with $\sigma = 0.2$

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 errors in 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:

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