Induced Gene Expression with Inducer Half-Live

Rainer Machne

Oct 07, 2021

Assuming that the inducer half-life $t_{1/2} = \frac{\log 2}{\delta}$ is not affected by metabolism, and that the measured fluorescence per OD is linearly related to protein concentration, $f_{\mathrm{OD}} \sim y(t)$, we can describe the measured data as:

$$\begin{aligned} \beta &= \mu + \delta_P \\ I(t) &= I_0 e^{-\delta t} \\ \alpha(t) &= \ell + v\frac{I^n}{I^n + K^n} \\ \frac{\mathrm{d}y}{\mathrm{d}t} = y' &= \alpha - \beta y \end{aligned} \tag{1}$$

with culture growth rate μ and AVS-tagged protein degradation rate δ_P, basal ("leaky") transcription rate ℓ, maximal induced transcription rate v, half-maximal inducer concentration K and a Hill-factor n, and initial inducer concentration I_0 and inducer degradation rate δ.

Note that gene expression functions are purely empirical, aimed at fitting the data at hand. The model simplifies multiple steps of complex formation (inducer and transcription factor, transcription factor on promoter) and subsumes both transcription and translation. Nevertheless, the parameters may roughly reflect dominant biochemical parameters of the real system, eg. K could be representative of the actual dissociation constant between inducer and transcription factor $K = \frac{IP}{P}$, or $n > 1$ may reflect multiple binding sites of the complex on the promoter.

This inhomogenous first-order differential equation can be solved easily for constant inducer, $\delta = 0$, (Appendix I):

$$y(t) = y_0 e^{-\beta t} + \frac{1 - e^{-\beta t}}{\beta}\left(\ell + v\frac{I^n}{I^n + K^n}\right). \tag{2}$$

An analytic solution for the general case, $\delta > 0$, is somewhat harder. The integration routine of the Wolfram alpha web service provides the solution which simply replaces the activation term with the hypergeometric function ${}_2\mathrm{F}_1$:

$$y(t) = y_0 e^{-\beta t} + \frac{1 - e^{-\beta t}}{\beta}\left(\ell + v\ {}_2\mathrm{F}_1\left(1, \frac{\beta}{n\delta}; \frac{\beta}{n\delta} + 1; -e^{n\delta t}\frac{K^n}{I_0^n}\right)\right). \tag{3}$$

${}_2\mathrm{F}_1$ is implemented in the GNU Scientific Library (GSL), but requires certain transformations of the variables (Forrey 1997) to ensure convergence (Appendix II). For our model ${}_2\mathrm{F}_1$ is well defined only for $\delta > 0$ and $I_0 > 0$ and we use eq.(2) for $I_0 = 0$ or $\delta = 0$ (no or constant inducer) and eq.(3) for $\delta > 0$.

We observed numerical stability problems for too small δ, esp. when combined with small I_0. A more stable implementation of ${}_2\mathrm{F}_1$ or the required transformations would be desirable. Alternatively, a scaling to relative time t/δ may avoid the problematic terms. However, the approach worked well for the observed inducer half-lives.

Parameter Fitting Strategy

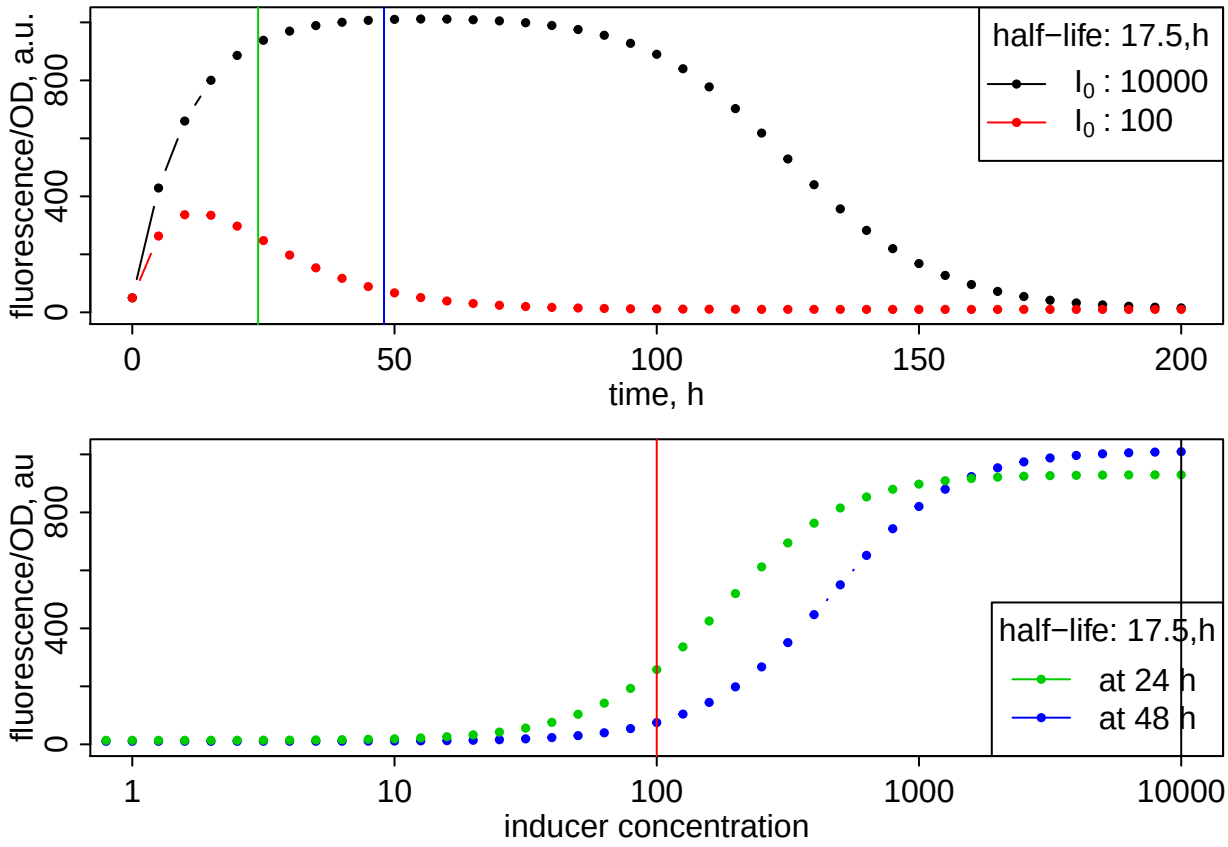
1. Calculate growth rate μ for all experiments,
2. Calculate μ and $\beta = \mu + \delta_P$ from rhamnose experiment after step-down to get ASV-tagged protein degradation rate δ_P,
3. Estimate y_0 from 0 inducer for each promoter.

These parameters should now remain unchanged. Use the correct set for each fit. Consider which parameters are shared between which experiments, eg. all experiments are done with ASV-tagged mVenus, so the protein degration rate δ_P should be the same for all experiments. Leaky transcription rate l and initial concentration y_0 should be the same for all experiments using the same promoter, independent of induction strength. Manually estimate parameter ranges and start values for variable parameters to (a) make physical/biological sense, and (b) roughly cover the range of observed values.

4. Fit v and K from dose-response curves using a rough estimate of inducer half-life δ,
5. Use obtained parameters as initial values and fit half-lives δ from time-series experiments,
6. Iterate the above procedure until all shared parameters converge.

Steps 4-6 could be implemented in an automated loop.

Example



APPENDIX

Appendix I: Induced Transcription with Constant Inducer

The "inhomogenous linear first order differential equation",

$$y'(t) + \beta y(t) = \alpha\,,$$

can be solved by adding a particular solution y_p of the inhomogenous equation and the general solution y_h of the homogenous version of the equation (with $\alpha = 0$).

The general solution is $y_h = Ce^{-\beta t}$, and the particular solution is simply $y_p = \frac{\alpha}{\beta}$, and we get:

$$y(t) = y_p + y_h = \frac{\alpha}{\beta} + Ce^{-\beta t}\,.$$

Assuming an initial condition $y(0) = y_0$ we further get $C = y_0 - \frac{\alpha}{\beta}$, and

$$y(t) = y_0 e^{-\beta t} + \frac{\alpha}{\beta}\left(1 - e^{-\beta t}\right)\,.$$

Appendix II: Induced Transcription with Inducer Half-Life

Here the factor α itself is a function of time t, and can use the "Variation of the Constant" method, where the constant C of the solution of the homogenous equation is replaced by a function $f(t)$, ie.

$$y(t) = f(t)\, e^{-\beta t} \tag{4}$$

Using this in the full equation, we get:

$$f'(t)e^{-\beta t} + f(t)e^{-\beta t}(-\beta) + \beta f(t)e^{-\beta t} = \alpha(t)$$

where two terms on the left-hand side cancel out, and we need to integrate to get $f(t)$:

$$f(t) = \int \alpha(t)e^{\beta t}\mathrm{d}t + C$$

Let's first make it simple, and assume a linear induction with $I \ll K$, with

$$\alpha(t) \approx v\frac{I_0 e^{-\delta t}}{K}\,.$$

This is comparatively easy to integrate and yields

$$f(t) = \frac{v}{\beta - \delta}\frac{I_0}{K}e^{(\beta-\delta)t} + C\,.$$

Again solving for $y(0) = y_0$ we obtain an expression for C and get:

$$y(t) = y_0 e^{-\beta t} + \frac{v}{\beta - \delta}\frac{I_0}{K}\left(e^{-\delta t} - e^{-\beta t}\right)\,.$$

Next, we attempt to integrate the more complex activation function:

$$\alpha(t) = \ell + v\frac{(I_0 e^{-\delta t})^n}{(I_0 e^{-\delta t})^n + K^n}$$
$$f(t) = \int \alpha(t) e^{\beta t} \mathrm{d}t + C$$

At Wolfram Alpha (replacing I_0 by `A` and t by `x`)
`integrate ( (l + v*A^n*e^(-d*n*x)/(A^n*e^(-d*n*x) + K^n))*e^(b*x))` yields
`(e^(b x) (v 2F1(1, b/(d n), b/(d n) + 1, -A^(-n) e^(d n x) K^n) + l))/b + constant`,
where ${}_2\mathrm{F}_1$ is the hypergeometric function, ie.:

$$f(t) = \frac{e^{\beta t}}{\beta}\left({}_2\mathrm{F}_1\left(1, \frac{\beta}{n\delta}; \frac{\beta}{n\delta} + 1; -e^{n\delta t}\frac{K^n}{I_0^n}\right) v + l\right) + C\,. \tag{5}$$

We could not identify the applied integration rule in literature (TODO), but minimally, for $\beta \neq 1$: $\int \frac{e^{\beta t}}{1+e^t}\mathrm{d}t = \frac{e^{\beta t}}{\beta}\,{}_2\mathrm{F}_1(1, \beta; \beta + 1; -e^t)$).

Substituting eq. 5 in eq. 4 and solving for $y(0) = y_0$, we get:

$$y(t) = y_0 e^{-\beta t} + \frac{1}{\beta}\left({}_2\mathrm{F}_1\left(1, \frac{\beta}{n\delta}; \frac{\beta}{n\delta} + 1; -e^{n\delta t}\frac{K^n}{I_0^n}\right) v + l\right)\left(1 - e^{-\beta t}\right).$$

The hypergeometric function ${}_2\mathrm{F}_1$ is rather unusual but often turns up in solutions of differential equations. See eg. Daalhuis' presentation at NIST (https://dlmf.nist.gov/15.17) for more information. It can be solved using the `R` interface to its implementation in the "GNU Scientific Library", specifically the function `gsl::hyperg_2F1`. However, (fast) convergence is only guaranteed for $|z| < 1$, and may fail for certain combinations of the other parameters. For our model $z < 0$ is always true, but $|z|$ can become very large, eg. it is exponential with t, and inversely proportional to I_0. We use the transformation provided by Forrey (1997) for $z < -1$:

```
library(gsl)
Gauss2F1_forrey <- function(a,b,c,z){
  if( abs(z) < 1 ) {
    y <- hyperg_2F1(a, b, c, z)
  } else {
    w <- 1/(1-z)
    A <- w^a*gamma(c)*gamma(b-a)/(gamma(b)*gamma(c-a))*hyperg_2F1(a,c-b,a-b+1,w)
    B <- w^b*gamma(c)*gamma(a-b)/(gamma(a)*gamma(c-b))*hyperg_2F1(b,c-a,b-a+1,w)
    y <- A+B
  }
  y
}
```

An alternative approach was suggested at `stackexchange` by Stephane Laurent which seems more efficient. However, no official reference for this solution was provided.

```
## https://stats.stackexchange.com/questions/33451/computation-of-hypergeometric-function-in-r
## quote: "Based on my experience I also recommend to only evaluate the Gauss
## hypergeometric function for a value of the variable lying in [0,1], and
## use a transformation formula for values in ]-Inf,0]."
## Alternatively, use a more precise implementation at
## http://stla.overblog.com/the-binary-splitting-with-the-r-gmp-package-application-to-gauss-hypergeome
library(gsl)
Gauss2F1_laurent <- function(a,b,c,z){
  if(z>=0 & z<1){
```

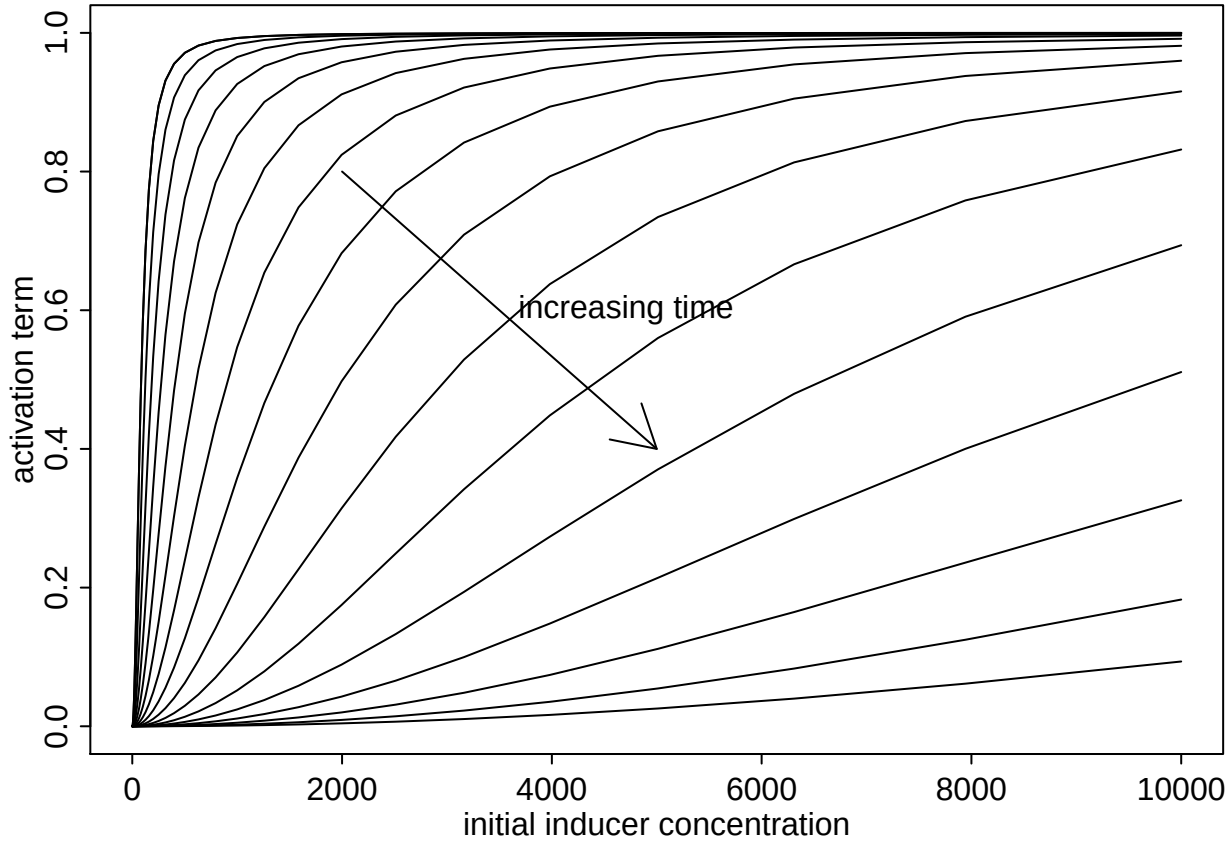
```
    hyperg_2F1(a,b,c,z)
  }else{
    hyperg_2F1(c-a,b,c,1-1/(1-z))/(1-z)^b
  }
}
```

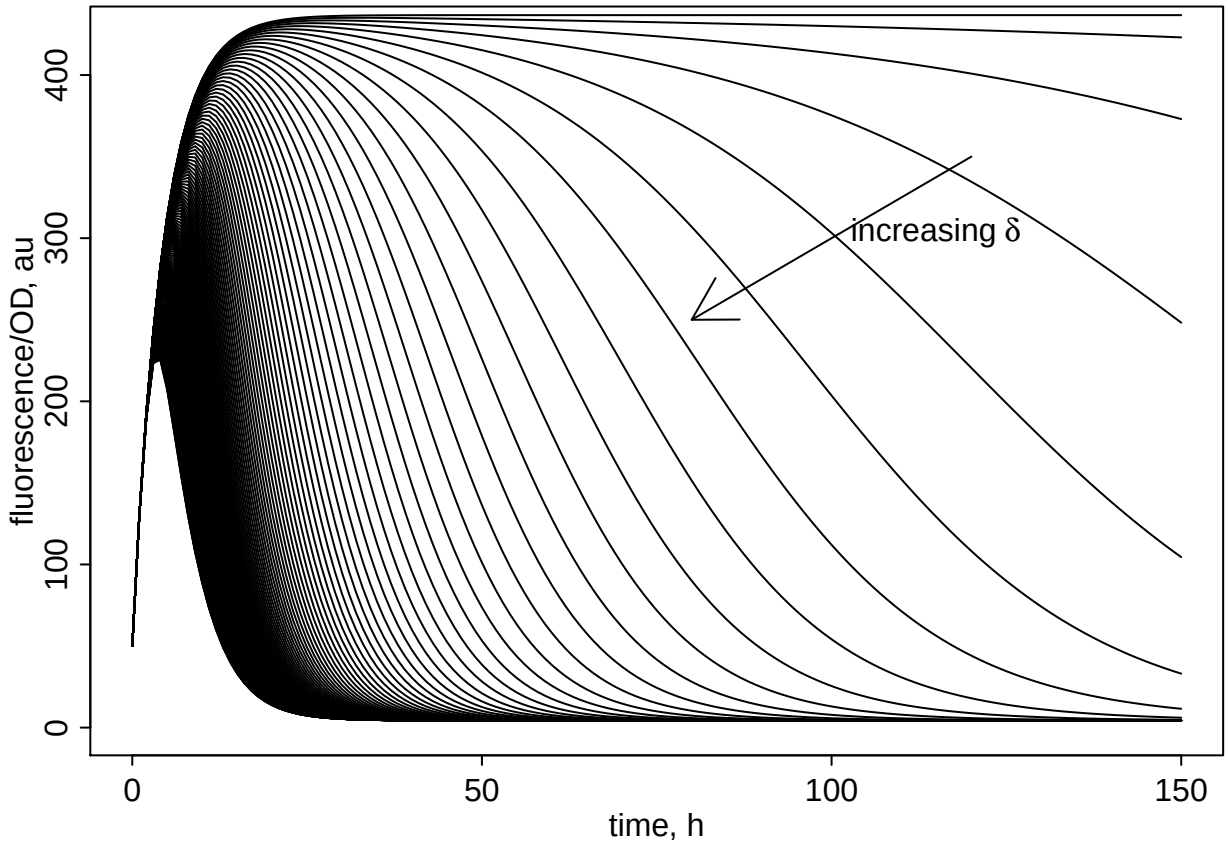
References

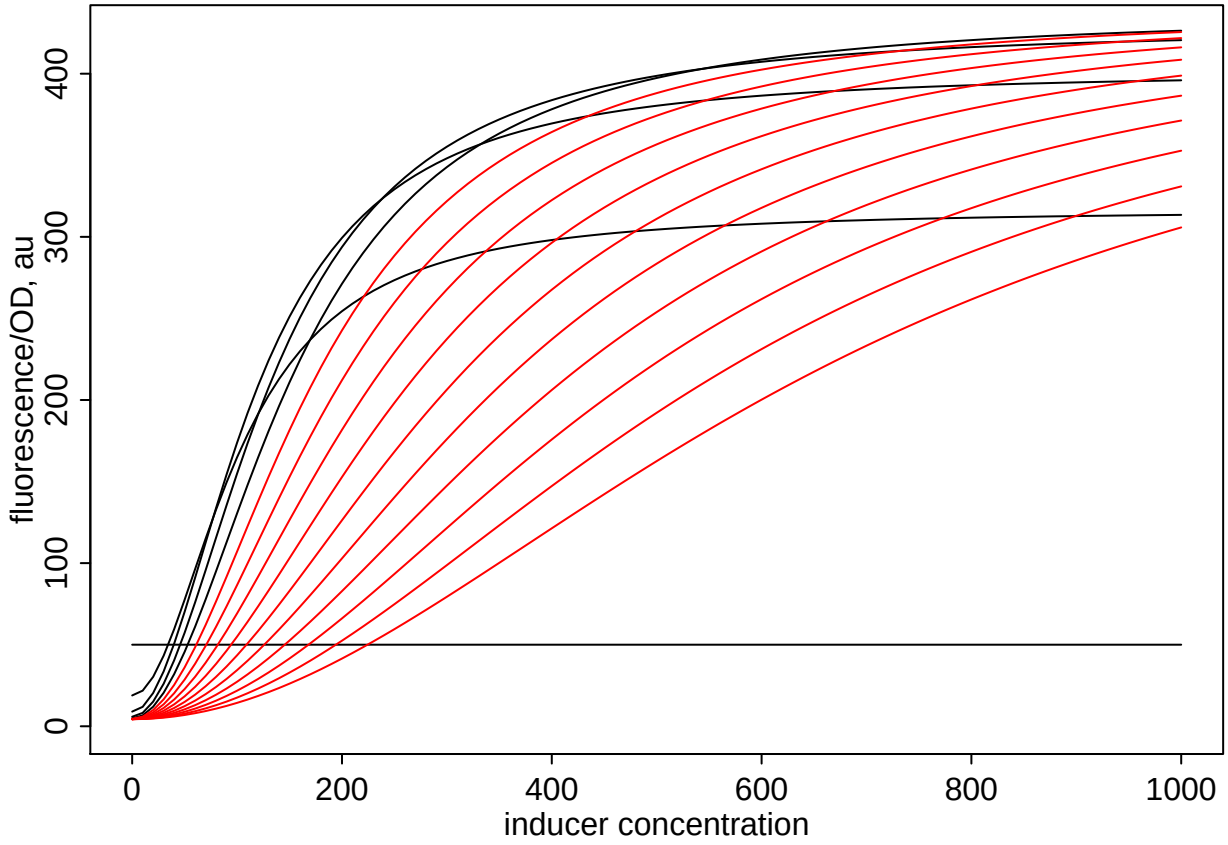
Forrey, Robert C. 1997. "Computing the Hypergeometric Function." *Journal of Computational Physics* 137 (1): 79–100. doi:https://doi.org/10.1006/jcph.1997.5794.

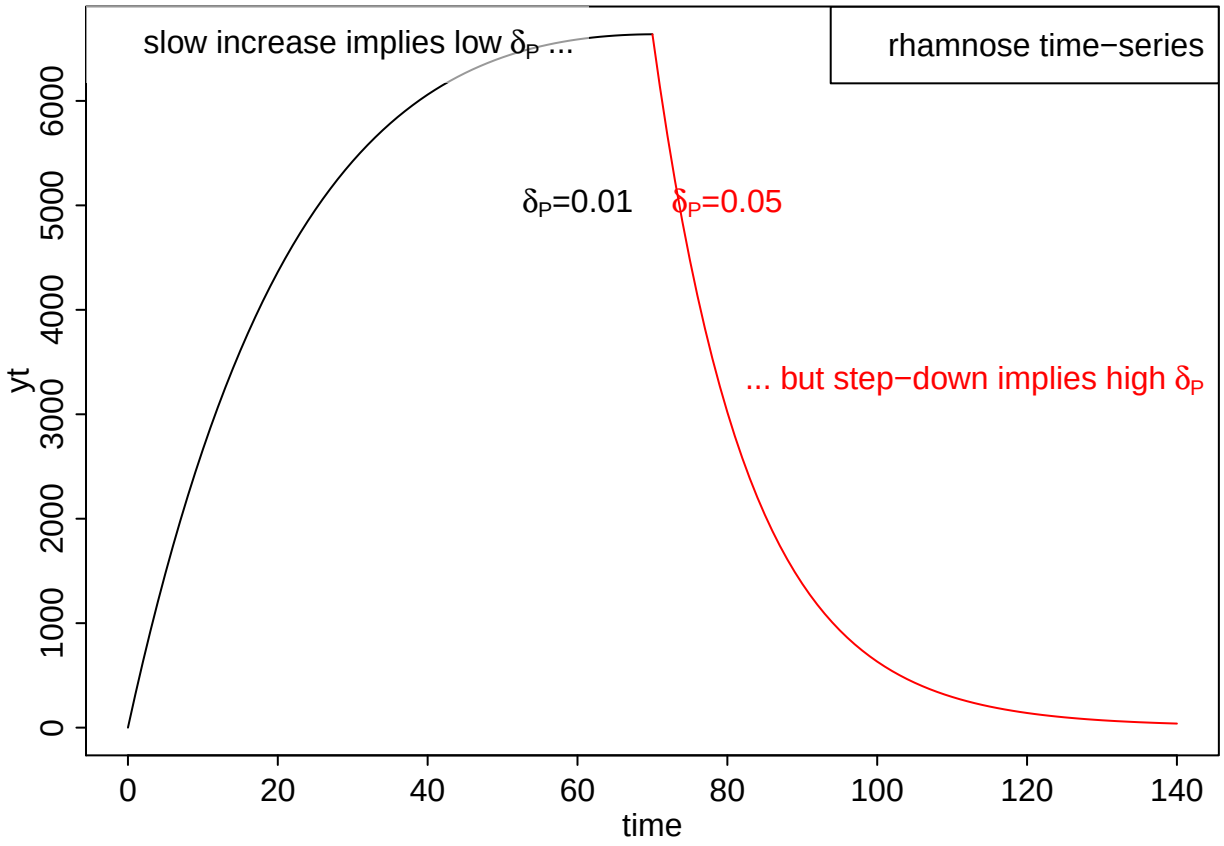
Appendix III: ANALYSIS

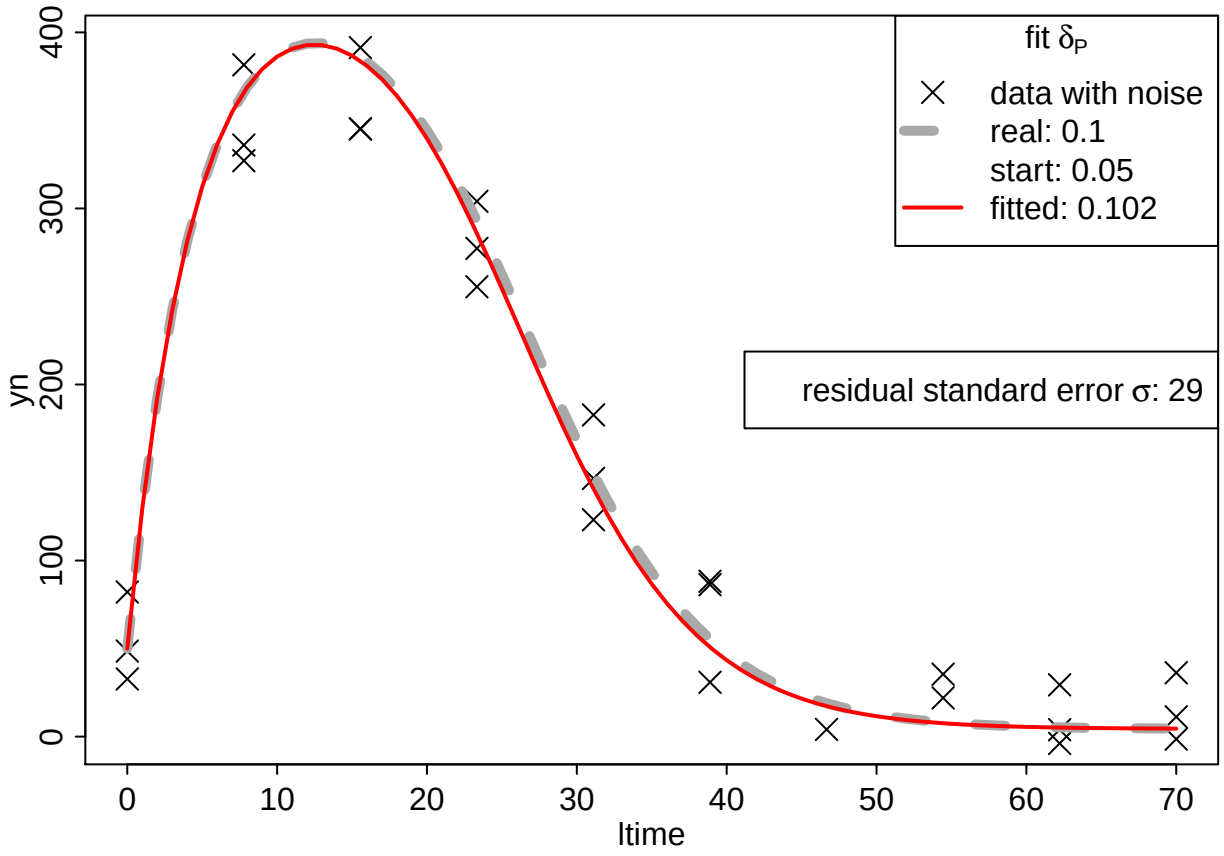
```
source("../scripts/analyse.R")
```


activation term
0.0
0.2
0.4
0.6
0.8
1.0
increasing time
0
2000
4000
6000
8000
10000
initial inducer concentration


fluorescence/OD, au
0
100
200
300
400
increasing δ
0
50
100
150
time, h


fluorescence/OD, au
0
100
200
300
400
0
200
400
600
800
1000
inducer concentration


slow increase implies low δP ...
rhamnose time−series
δP=0.01
δP=0.05
... but step−down implies high δP
yt
0
1000
2000
3000
4000
5000
6000
0
20
40
60
80
100
120
140
time


fit δ_P
data with noise
real: 0.1
start: 0.05
fitted: 0.102
residual standard error σ: 29
yn
ltime
0
100
200
300
400
0
10
20
30
40
50
60
70