# Assignment ODE's
# Genetic oscillator

Computational Methods for Scientific Applications, 2023

*Reproducibility is an important principle in the scientific method. The results obtained in e.g. an experiment should be achieved again when the study is replicated. In this assignment you will reproduce parts of the results in a research article in the area systems biology. In the article, a deterministic and a stochastic model are studied, both describing a "circadian clock". You may not be able to understand everything in the article, but you will be able to understand enough for this mini project.*

## Background

Our environment contains many cyclic processes, such as the diurnal rhythm of light and darkness. Many organisms use molecular mechanisms known as "circadian clocks", to adapt to such changes. It simultaneously promotes the expression of a negative element, which in turn represses the positive element. The cycle completes itself upon degradation of the negative element and re-expression of the positive element. In a research paper (Vilar, Kueh, Barkai, Leibler)[1] a deterministic and a stochastic model are studied, both describing a "circadian clock". The article has been important in the field of systems biology. They use a circadian clock model containing two proteins: an activator protein $A$ and a repressor protein $B$. The corresponding activator and repressor genes are denoted by $D_A$ and $D_R$, respectively. ($D'_A$ and $D'_R$ denote the same genes with $A$ and $R$, respectively, bound to them.) See the article for a more detailed explanation.

[1] *José M. G. Vilar, Hao Yuan Kueh, Naama Barkai, Stanislas Leibler : Mechanisms of noice-resistance in genetic oscillators*, PNAS April 30, 2002 vol. 99 no. 9 page 5988-5992. The article is available in Studium.

## Assignment

a) Reproduce the result in Figure 2, a and b in the article, i.e. simulate the circadian rhythm using Python, and study how the activator and repressor protein, $A$ and $R$ respectively, vary during 400 hours. You can use the built-in ODE solver in SciPy to solve the problem.
The mathematical model is described in equation [1] in the article, and parameter values and initial values are described under Figure 1.

(There are lots of "stuff" to deal with in this problem, many variables and parameters. A tip is to name them in ways that are easy to interpret and that correspond to the problem formulation. Also, to pass all the parameters in a long list into the right-hand-side function is a bit awkward. Try to think of a way to make the call to the function it a little bit easier.)

b) Some ODE's are so called *stiff* ODE's. One definition[2] of stiffness is:

> equations where certain implicit methods (...) perform better, usually tremendously better, than explicit ones.

It is possible to change numerical method in `solve_ivp`. Change the method to a so-called 'implicit' solver, especially beneficial for 'stiff ODE's' (read the reference page for `solve_ivp`). Implicit methods are more complicated and therefore typically more expensive (takes more time) then explicit methods. But, when the problem is a stiff ODE, the implicit will actually be quicker then explicit ones. Change solver and decide whether the ODE is stiff or not. Which method would you preferably chose in his case?

(How to measure execution time, see Lab 1, part 3.)

[2] *C.F. Curtiss, J.O. Hirschfelder (1952). Integration of stiff equations.* Proceedings of the National Academy of Sciences. Vol 38, pp. 235-243.