Home Exam BIOS13 Modelling Biological Systems

Dept. of Biology, Lund University, 2022-2023

2. Protein production (6p)

Assume a cell produces a particular protein at a fixed rate a. The protein leaks out of the cell membrane at a rate depending on the protein concentration within the cell. The dynamics of the protein concentration in the cell can thus be written

$$\frac{dP}{dt} = a - bP$$

where P is protein concentration and b is a 'leaking rate'.

a) What is the equilibrium protein concentration in the cell? (1p)

An equilibrium is defined as 'no change'. In this case when $\frac{dP}{dt} = 0$:

$$\frac{dP}{dt} = a - bP = 0$$
$$a = bP$$
$$P = \frac{a}{b}$$

Answer: The equilibrium is at $P^* = \frac{a}{b}$

b) Show (mathematically) that it is a stable equilibrium. (1p)

We have

$$\frac{dP}{dt} = a - bP = f(P)$$

The derivative of $f(P)$ at P^* determines the stability of P^*:

$$f'(P) = 0 - b = -b$$

The derivative is always negative, and especially so at $P = P^*$, which implies that P^* indeed is stable.

Assume further that the cell also produces *another* protein, Q, which catalyzes the breakdown of P. The production of Q is stimulated by the concentration of P. One can think of Q as a way to regulate the concentration of P. The combined dynamics of P and Q can now be written

$$\begin{cases} \frac{dP}{dt} = a - bP - cPQ \\ \frac{dQ}{dt} = rP - bQ \end{cases}$$

where it is assumed that Q has the same leaking rate b as P.

c) Calculate the equilibrium (P^*, Q^*). (1p)

For this we have to solve the two equations, $\frac{dP}{dt} = 0$ and $\frac{dQ}{dt} = 0$, simultaneously:

I) $a - bP - cPQ = 0$
II) $rP - bQ = 0$

Luckily, equation (II) quite easily gives a useful answer:

$$rP = bQ$$

$$P = \frac{b}{r}Q$$

This can be substituted into equation (I), which gives:

$$a - \frac{b^2}{r}Q - \frac{bc}{r}Q^2 = 0$$

which can be re-arranged as (multiplying with r/bc and moving terms around) :

$$Q^2 + \frac{b}{c}Q - \frac{ra}{bc} = 0$$

which is a standard second order equation with solutions (there are many alternative expressions here, all correct):

$$Q = -\frac{b}{2c} + (-)\sqrt{\frac{b^2}{4c^2} + \frac{ra}{bc}} = \frac{-\frac{b}{c} + \sqrt{\frac{b^2}{c^2} + 4\frac{ra}{bc}}}{2} = \frac{-b + \sqrt{b^2 + 4\frac{rac}{b}}}{2c}$$
$$= \frac{-b^2 + \sqrt{b^4 + 4abcr}}{2bc} = \frac{b^2 - \sqrt{b^4 + 4abcr}}{-2bc}$$

The negative solution can be excluded since Q can not be negative (it is a concentration). Next, we can find P from the solution to (II):

$$P = \frac{b}{r}Q = \frac{b}{r}\left(\frac{-b + \sqrt{b^2 + 4\frac{rac}{b}}}{2c}\right) = \frac{-b^2 + \sqrt{b^4 + 4rabc}}{2cr} = \frac{-b^2}{2cr} + \sqrt{\frac{b^4 + 4rabc}{4c^2r^2}}$$
$$= \frac{-b^2}{2cr} + \sqrt{\frac{b^4}{4c^2r^2} + \frac{ab}{cr}}$$

Again, there are many alternative expressions, all correct.

d) Trajectories of the protein dynamics can be visualized in the system *phase space* (or phase plane). What are the coordinate axes of this phase space? (1p)

The phase space has the system's state variables on the coordinate axes. In this case, P and Q.

e) Write an R script that

i) plots the isoclines in the phase space of the system above (1p)

First we need expressions for the isoclines, which are the points in phase space where one or the other of the state variables has zero growth.
The P isocline (there is only one) is found by solving:

$$a - bP - cPQ = 0,$$

which can be solved as either Q as a function of P:

$$a - bP = cPQ$$

$$Q = \frac{a - bP}{cP} = \frac{a}{cP} - \frac{b}{c},$$

or P as a function of Q:

$$a = bP + cPQ = P(b + cQ)$$

$$P = \frac{a}{b + cQ}$$

The Q isocline is the solution to $rP - bQ = 0$, which we already solved as $P = \frac{b}{r}Q$, or, alternatively, $Q = \frac{r}{b}P$.
See the script below for R-code.

ii) simulates the dynamics and plots the resulting trajectory in the same phase space. (1p)

This is a continous time system, so we need a continuous time solution, such as provided by the ode function, just like we have done in many exercsises. A possible R-script goes:

```
# Setting system parameters first:
pars <- list(a=1,b=1,c=1,r=1)

# Plotting P isocline, using Q as a function of P:
P <- seq(0,5,by=0.1) # a range of P-values
Q <- pars$a/(pars$c*P) - pars$b/pars$c # the corresponding Q-values
plot(P, Q, type='l', col='blue', ylim=c(0,1), xlim=c(0,1.5))

# Q-isocline:
Q <- pars$r/pars$b*P
lines(P,Q,col='red')

# Next, use ode to calculate a trajectory:
library(deSolve)
# The set of differential equations:
PQfunc <- function(t,PQ,pars) {
  P <- PQ[1]
  Q <- PQ[2]
  dPdt <- pars$a - pars$b*P - pars$c*P*Q
  dQdt <- pars$r*P - pars$b*Q
  return(list(c(dPdt,dQdt)))
}

time_vec <- seq(0,10,by=0.1)
PQ0 <- c(P=0,Q=0) # Initial condition
results <- ode(func = PQfunc, times = time_vec, y = PQ0, parms = pars)
# Plot the result in phase space:
lines(results[,2],results[,3],col='black')
```

3. Bumble bee colony growth (5p)

Bumble bees have colonies that start with a single queen in spring. She produces workers, that, as they mature, help produce new workers. At some point, a switching time t_S, the colony switches to produce new queens (and drones) instead. Queen production continues until the end of the season (at time T). The new queens mate and then hibernate until next spring, when they form new colonies.

The growth of the worker population (W) before the switching time can be formulated as

$$\frac{dW}{dt} = rW$$

where r is a positive constant ($r > 0$).

a) Assuming the colony starts with a single worker at $t = 0$, what is the number of workers at the switching time t_S? (1p)

This can be recognised as a case of exponential growth. It thus has the general solution $W(t) = W(0)e^{rt}$, which gives

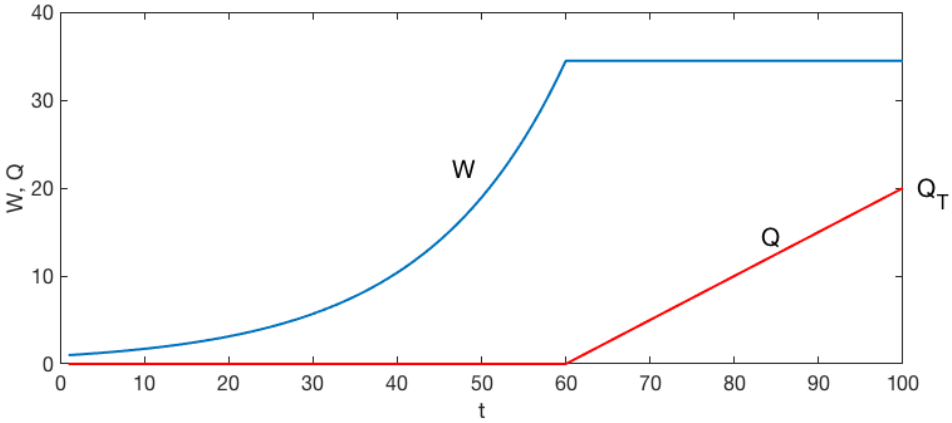
$$W(t_s) = e^{rt_s}$$

(since W(0)=1 according to the assumption).

After the switching time, the number of workers is assumed to remain constant (assume they don't die). The production of queens (Q) depends on the number of workers, following

$$\frac{dQ}{dt} = cW$$

where c is another positive constant. Queen production is thus constant until the end of the season, and the total number of queens in the colony grows linearly with time (following a straight line). An example season is depicted below (using $t_S = 60$, $T = 100$).



b) What is the number of queens at the end of the season, Q_T? Express it as a function of the switching time t_S and the parameters r, c and T. (1p)

Since the queens in this case have a constant growth rate, they will grow following a linear function with the slope $cW(t_s)$ and that starts at zero at $t = t_s$. Such a function

can be written $Q(t) = cW(t_s)(t - t_s)$ (one can check that the derivative $\frac{dQ}{dt}$ is correct and that $Q(t_s) = 0$). Now we can express Q_T as (using $W(t_s)$ from above):

$$Q_T = ce^{rt_s}(T - t_s)$$

c) Colony fitness can be assessed by the number of new queens it can produce. Calculate the optimal switching time that maximizes Q_T, i.e. that maximizes the number of queens at the end of the season. (1p)

An optimum is found by first solving $\frac{dQ_t}{dt_s} = 0$:

$$\frac{dQ_T}{dt_s} = cre^{rt_s}(T - t_s) + ce^{rt_s}(-1) = ce^{rt_s}(r(T - t_s) - 1) =$$

$$= rce^{rt_s}\frac{(r(T - t_s) - 1)}{r} = rce^{rt_s}\left(\frac{rT - rt_s - 1}{r}\right) = rce^{rt_s}\left(T - \frac{1}{r} - t_s\right) = 0$$

$$r(T - t_s) - 1 = 0$$

$$T - t_s = \frac{1}{r}$$

$$t_s = T - \frac{1}{r}$$

Next, we should make sure it is a maximum. This is done either with a sign table or by calculating the second derivative:

$$\frac{d^2Q_T}{dt_s^2} = re^{rt_s}(r(T - t_s) - 1) + e^{rt_s}(-r)$$

Inserting the optimum gives:

$$\frac{d^2Q_T\left(t_s = T - \frac{1}{r}\right)}{dt_s^2} = \cdots = -r < 0$$

Since the second derivative is negative we indeed have a maximum.

d) Extend the model to include worker and queen mortality. Motivate (briefly) your model extension. (1p)

For example (using fixed per capita mortality, the same for workers and new queens):

Before switching: $\frac{dW}{dt} = rW - \mu W = (r - \mu)W$

After switching: $\frac{dW}{dt} = -\mu W$

$\frac{dQ}{dt} = cW - \mu Q$

e) Write a script in R that simulates colony growth (following your extended model in *d*)) and produces a plot similar to the one above. (1p)

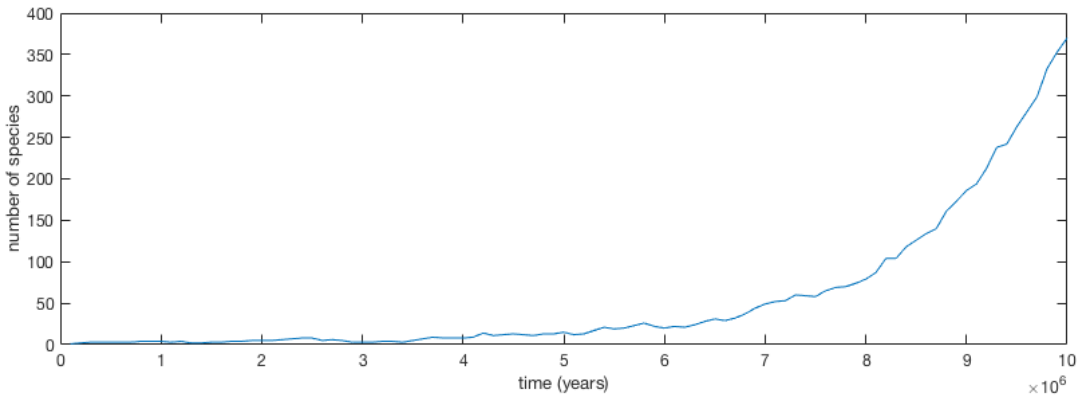
Again, we can use ode for this, although the worker dynamics before the switch are still exponential (with growth rate $r - \mu$) and can be calculated directly as $W(t) = e^{(r-\mu)t}$.

```
rm(list=ls())
# Parameter list:
P <- list(tS=60, r=0.1, c=0.1, mu= 0.05, T=100)

# Dynamics before switch (ts):
t_vec <- seq(0, P$tS, by=0.1)
Wbefore <- exp((P$r-P$mu)*t_vec)

plot(t_vec,Wbefore,type='l',col='blue',xlim=c(0,P$T), xlab='time',ylab='W(blue), Q(red)')

# WQ-dynamics after ts:
WQgrowth <- function(t,WQ,P) {
  W <- WQ[1]
  Q <- WQ[2]
  dWdt <- -P$mu*W
  dQdt <- P$c*W - P$mu*Q
  return(list(c(dWdt,dQdt)))
}

t_vec2 <- seq(P$tS,P$T,by=0.1)
# Use the worker density at t=ts as an initial condition:
WQ0 <- c(W=Wbefore[length(Wbefore)], Q=0)

WQafter <- ode(func = WQgrowth, times = t_vec2, parms = P, y=WQ0)

# Plot workers and queens over time:
Wafter <- WQafter[,'W']
Qafter <- WQafter[,'Q']
lines(t_vec2,Wafter,col='blue')
lines(t_vec2,Qafter,col='red')
```

4. Phylogenesis (4p)

The number of species within a lineage increases with speciation events and decreases with extinction events. For simplicity, assume a species goes extinct with probability P_E within a period of 100,000 years. *If it does not go extinct*, it may speciate (and become two species) with probability P_S during the same time period.

a) Write an R *function* that simulates the number of species N over a period of 10 million years, plots the number of species over time, and *returns* the number of species present after 10 million years. The function should take the extinction probability P_E and speciation probability P_S as input parameters. (2p)



There are a few pitfalls here. First of all we should make sure that extinct species do not speciate. There is also a particular feature in R that causes problems here. In most programming languages, a loop from 1 to 0 does nothing :

```
for ( i in 1:0 ) { … }
```

However, in R such a loop will go backwards:

```
> for(i in 1:0) {print(i)}
[1] 1
[1] 0
```

This comes from the fact the the expression 1:0 (in R) produces a vector [1, 0], which is not the case in for example Matlab. In our coding example here, this may cause problems whenever N is zero (extinction of all species), which has to be treated with care.

You find my R solution on the next page. It is not the most efficient (some of you came up with much faster solutions), but does the job.

```
species <- function(PE,PS) {
  t_max <- 1e7 # 10 milion, length of simulation
  time_step <- 1e5 # 100,000, length of one time-step
  step_count <- t_max/time_step # Total number of time_steps
  N_save <- rep(0,step_count+1) # save also for t=0
  # Start with one species:
  N <- 1
  N_save[1] <- N
  for (i in 1:step_count) {
    # We track the new generation separately,
        # without updating N until the end.
    # This is so because N marks the end of the loop
        # and should not be altered within the loop.
    N_new <- N
    # For each species:
    for (s in 1:N) {
      # it may go extinct:
      if(runif(1)<PE){
        N_new <- N_new-1
      }else{ # if not, possibly speciate:
        if(runif(1)<PS){
          N_new <- N_new+1
        }
      }
    }
    # update N:
    N <- N_new
    N_save[i+1] <- N
    if(N<=0){
      break # This is important! Otherwise the loop on line 16 (for (s in 1:N)) will run
backwards!
    }
  }
    # Plot the dynamics on the right time-scale:
    plot(time_step*(0:step_count),N_save,type='l',xlab='time',ylab='#species')
    return(N) # Don't forget to return the result!
}
# Testing:
species(0.1,0.2)
```

A faster alternative:

```
species <- function(PE,PS) {
  t_max <- 1e7 # 10 milion, length of simulation
  time_step <- 1e5 # 100,000, length of one time-step
  step_count <- t_max/time_step # Total number of time_steps
  N_save <- rep(0,step_count+1) # save also for t=0
  # Start with one species:
  N <- 1
  N_save[1] <- N
  for (i in 1:step_count) {
    survivors <- sum(runif(N)<(1-PE))
    new_species <- sum(runif(survivors)<PS)
    N <- survivors + new_species
    N_save[i+1] <- N
    if(N<=0){
      break # This is important! Otherwise the loop on line 14 may run backwards!
    }
  }
  # Plot the dynamics on the right time-scale:
  plot(time_step*(0:step_count),N_save,type='l',xlab='time',ylab='#species')
  return(N)
}
```

b) Write a script that uses the function above to simulate the evolutionary process above 1000 times and plot a histogram of the number of species after 10 million years. Use $P_E = 0.1$, $P_S = 0.2$.
For this task, it is advisable to turn off the plotting of species over time.
Tip: The R-function hist(v) makes a histogram of the data in vector v.
(1p)

First remove the plot command from the function above.

Next, write a short script which calls the function repreatedly and save the result in a vector. The result is eventually plotted with hist:

```
source('species.R') # define the function
iterations <- 1000
PE <- 0.1
PS <- 0.2
N_final = seq(0,iterations)
for (i in 1:iterations) {
  N_final[i] <- species(PE,PS)
}
hist(N_final,breaks=100)
```

c) It can be assumed that closely related species compete with each other to some extent, and that niche space is limited. Modify (somehow) the function in *a)* such that the probability of speciation is *density dependent*, i.e. assume P_S declines with the number of species. Motivate (briefly) your model choice. (1p)

In principle any function for PS that decreases with N is possible here, just make sure it stays within the limits [0,1] (since it is a probability). As a simple example, one can use an exponentially declining function

$$P_S = P_0 \mathrm{e}^{-kN},$$

where P_0 is the probability of speciation at low values of N and k is a parameter regulating the strength of the density dependence. An alternative could be an accellerating dependence like $P_S = P_0 \mathrm{e}^{-kN^2}$ ('half a Gaussian curve'). This is perhaps more realistic, since the effect is often the strongest as N grows large.

The new R-function (using my first example) is only a slight modification of the first one (next page):

```
species2 <- function(PE,P0,k) {
  t_max <- 1e7 # 10 milion, length of simulation
  time_step <- 1e5 # 100,000, length of one time-step
  step_count <- t_max/time_step # Total number of time_steps
  N_save <- rep(0,step_count+1) # save also for t=0
  # Start with one species:
  N <- 1
  N_save[1] <- N
  for (i in 1:step_count) {
    # We track the new generation separately,
    # without updating N until the end
    # This is so because N marks the end of the loop
    # and should not be altered within the loop.
    N_new <- N
    # Density dependent speciation rate:
    PS = P0*exp(-k*N)
    # For each species:
    for (s in 1:N) {
      # it may go extinct:
      if(runif(1)<PE){
        N_new <- N_new-1
      }else{ # if not, possibly speciate:
        if(runif(1)<PS){
          N_new <- N_new+1
        }
      }
    }
    # update N:
    N <- N_new
    N_save[i+1] <- N
    if(N<=0){
      break # This is important! Otherwise the loop on line 18 will run backwards!
    }
  }
  # Plot the dynamics on the right time-scale:
  plot(time_step*(0:step_count),N_save,type='l',xlab='time',ylab='#species')
  return(N)
}
# Testing:
species2(0.1,0.2,0.01)
```