Answers,

Exercises 2, Dynamic Systems

Modelling Biological Systems, BIOS13
Jörgen Ripa

1. Lotka-Volterras predator-prey equations

Lotka-Volterras predator-prey equations look like this:

$$\begin{cases} \frac{dn}{dt} = rn - anp \\ \frac{dp}{dt} = anp - \mu p \end{cases},$$

…

a) Write an R script that plots the isoclines of the system. Make it a function file with input parameters according to:

```
LV_isoclines <- function(r,a,mu) {
  # equilibria (we use these coordinates to draw the isoclines):
  ne <- mu/a
  pe <- r/a
  # prey isocline:
  plot(c(0,2*ne), c(pe,pe), type='l', col='blue')
  # predator isocline:
  lines(c(ne,ne), c(0,2*ne), col='red')
}
```

b) Now we will run some simulations. Write an R-script like in the previous exercise, using the `ode` function. The difference now is that we have two state variables instead of one, so the system function has to return (a list of) two derivatives. Something like this:

```
LV_sys <- function(t, np, P) {
    # extract vector content:
    n <- np[1]
    p <- np[2]
    # calculate the two growth rates:
    dndt <- P$r*n - P$a*n*p
    dpdt <- P$a*n*p - P$mu*p
    list(c(dndt, dpdt )) # the result as a vector in a list
}
```

Next, write a script file … (next page)

```
# First load the deSolve package:
library(deSolve)

# define the growth function:
LV_sys <- function(t, np, P) {
    # extract vector content:
    n <- np[1]
    p <- np[2]
    # calculate the two growth rates:
    dndt <- P$r*n - P$a*n*p
    dpdt <- P$a*n*p - P$mu*p
    list(c(dndt, dpdt )) # the result as a vector in a list
}

# set up a vector of time-points for the output:
timevec <- seq( 0, 10, by=0.1 )

# we need a list of parameters:
P <- list( r=2, a=1, mu=2 )

np0 <- c(n=2,p=1)  # initial population sizes

# call the ode function to solve the differnetial equation:
out <- ode( y = np0, func = LV_sys, times = timevec, parms = P)
time <- out[,'time']
n <- out[,'n']
p <- out[,'p']

# plot the output in two panels:
op <- par(mfrow=c(2,1))
# first population sizes over time:
plot( time, n, type='l', col='blue' )
lines( time, p, type='l', col='red' )

# next the phase plane, starting with the isoclines:
LV_isoclines(P$r, P$a, P$mu)
# add the trajectory:
lines( n, p)
# plot a square at the initial condition
points(n[1], p[1], pch=0)

# reset graphics parameters:
par(op)
```

c) Calculate the Jacobian matrix of the Lotka-Volterra predator-prey equations. Write a function in R that takes the three parameters (r, a, μ) as input and returns the Jacobian matrix as output.

According to the lecture, we have (at the equilibrium point, using $c = 1$):

$$J = \begin{pmatrix} 0 & -\mu \\ r & 0 \end{pmatrix}$$

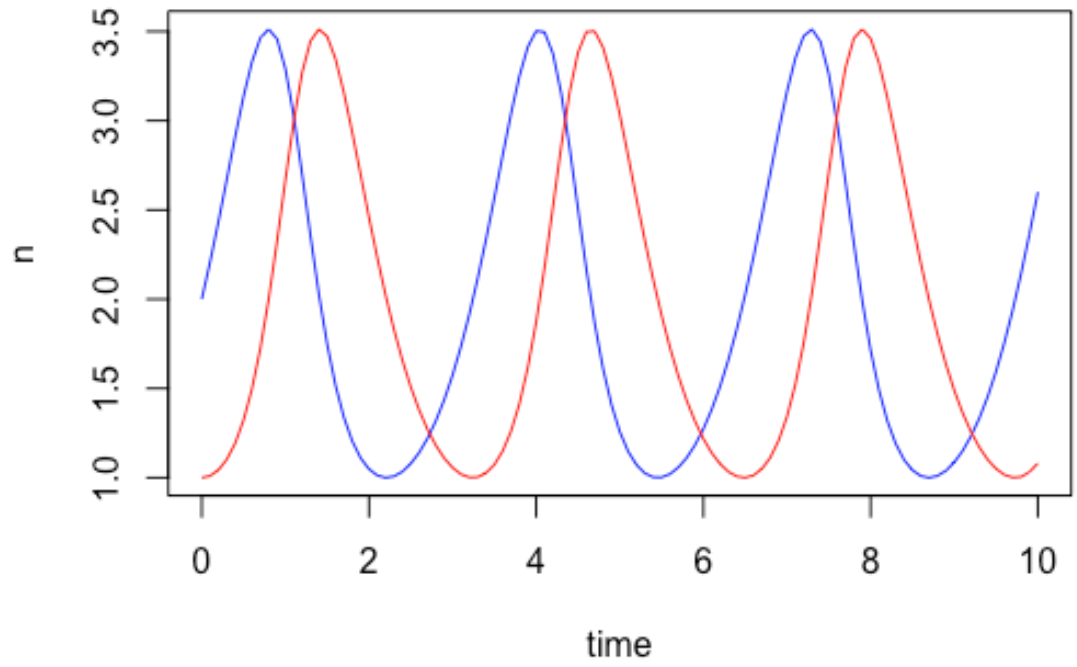
```
LV_Jacobian <- function(r,a,mu) {
  matrix(c(0,r,-mu,0),2,2)
}
```

(we don't use the a parameter)

d) Use your function in c) to calculate the Jacobian for a set of parameter values. Also calculate the cycle period associated with the complex eigenvalues. Is it a good approximation of the length of the actual cycles? (check your plots, rerun your simulation script with the new parameter values if necessary)

```
# make sure yo use the same parameters as in the simulation:
> J <- LV_Jacobian(r=2,a=1,mu=2)
> E <- eigen(J)
# the output E is a list:
> E
$values
[1] 0+2i 0-2i

$vectors
                        [,1]                   [,2]
[1,]  0.0000000-0.7071068i  0.0000000+0.7071068i
[2,] -0.7071068+0.0000000i -0.7071068+0.0000000i

# An approximation of the cycle period:
> 2*pi/Im(E$values[1])
[1] 3.141593
```

Compare to previous graph:



It looks good!

If you have the time:

2. Stabilising Lotka-Volterra

The Lotka-Volterra equations have been criticised for a number of unrealistic assumptions. For instance, the prey will grow to infinite population sizes without the predator. To correct for this, we can introduce density dependence of prey growth, following the logistic equation:

$$\begin{cases} \dfrac{dn}{dt} = r_0 n\left(1-\dfrac{n}{K}\right) - anp \\ \dfrac{dp}{dt} = anp - \mu p \end{cases}$$

a) Calculate the new equilibrium!

$\frac{dp}{dt} = 0 \;\Rightarrow\; p = 0 \quad or \quad n = \frac{\mu}{a}$, as before.

$\frac{dn}{dt} = 0 \;\Rightarrow\; n = 0 \;\; or \;\; r_0\left(1-\frac{n}{K}\right) - ap = 0.$

The last equation can be written $p = \frac{r_0}{a}(1-\frac{n}{K})$, which is a straight line in phase space!

Inserting $n^* = \mu/a$ gives $p^* = \frac{r_0}{a}\left(1-\frac{\mu}{Ka}\right)$.

b) Can you find a condition for predator existence, i.e. a necessary condition for a positive predator equilibrium?

It can be seen from the expression for p^* that $p^* < 0$ if $\mu > Ka$.

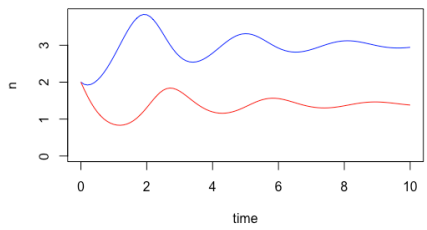
c) Write an R-script, like the one you did before, which plots the isoclines of the new system.

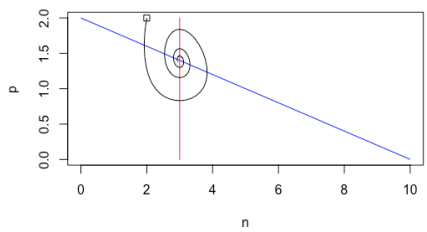
```
LV_isoclines2 <- function(r0,a,mu,K) {
  # prey equilibrium:
  ne <- mu/a
  # prey isocline (we only need two points):
  n <- c(0,K)
  p <- r0/a*(1-n/K)
  plot( n, p, type='l', col='blue')
  # predator isocline:
  lines( c(ne,ne), c(0, r0/a ), col='red')
}
```

d) Write an R-script to simulate this system and do some plotting, like above. Try some different parameter values. Does the equilibrium seem to be stable?

Next page

```r
# First load the deSolve package:
library(deSolve)

# define the growth function:
LV_sys2 <- function(t, np, P) {
  # extract vector content:
    n <- np[1]
    p <- np[2]
    dndt <- P$r*n*(1-n/P$K)  - P$a*n*p
    dpdt <- P$a*n*p - P$mu*p
    list(c(dndt, dpdt ))
}

# set up a vector of time-points for the output:
timevec <- seq( 0, 10, by=0.1 )

# we need a list of parameters:
P <- list( r=2, a=1, mu=3, K=10 )

np0 <- c(n=2,p=2)  # initial population sizes

# call the ode function to solve the differnetial equation:
out <- ode( y = np0, func = LV_sys2, times = timevec, parms = P)
time <- out[,'time']
n <- out[,'n']
p <- out[,'p']

# plot the output
# first population sizes over time:
op <- par(mfrow=c(2,1))
plot( time, n, type='l', col='blue', ylim=c(0,max(rbind(n,p))) )
lines( time, p, type='l', col='red' )

# next the phase plane, starting with the isoclines:
LV_isoclines2(P$r, P$a, P$mu, P$K)
# add the trajectory:
lines( n, p)
# plot a square at initial condition
points(n[1],p[1],pch=0)

par(op)
```





e) Calculate the new Jacobian matrix and its eigenvalues (in R). Conclusions?
First some math:

$$\left.\frac{\partial f_1}{\partial n}\right|_* = \left.\left(r_0\left(1-\frac{n}{K}\right)+r_0 n\left(-\frac{1}{K}\right)-ap\right)\right|_* = -\frac{r_0 n^*}{K} = -\frac{r_0\mu}{Ka}$$

$$\left.\frac{\partial f_1}{\partial p}\right|_* = (-an)|_* = -an^* = -\mu$$

$$\left.\frac{\partial f_2}{\partial n}\right|_* = (ap)|_* = ap^* = r_0\left(1-\frac{\mu}{Ka}\right)$$

$$\left.\frac{\partial f_2}{\partial p}\right|_* = (an-\mu)|_* = an^*-\mu = 0$$

$$J = \begin{pmatrix} -\frac{r_0\mu}{Ka} & -\mu \\ r_0\left(1-\frac{\mu}{Ka}\right) & 0 \end{pmatrix}$$

Next write an R function:

```
LV_Jacobian2 <- function(r0,a,mu,K) {
  J11 <- -r0*mu/K/a
  J12 <- -mu
  J21 <- r0*(1-mu/K/a)
  J22 <- 0
  matrix(c(J11,J21,J12,J22),2,2)
}
```

and use it:

```
> J <- LV_Jacobian2(r=2,a=1,mu=3,K=10)
> E <- eigen(J)
> E
$values
[1] -0.3+2.027313i -0.3-2.027313i

$vectors
                         [,1]                   [,2]
[1,]  0.8257228+0.0000000i  0.8257228+0.0000000i
[2,] -0.0825723-0.5579997i -0.0825723+0.5579997i
```

This time the eigenvalues do have a negative real part, and the equilibrium is stable, as can be seen in the plot on the previous page.