From the Lotka-Volterra model to bi-stability: computer practical.

Rob J. de Boer, Theoretical Biology, Utrecht University.

The phase portraits in the handouts were made with a computer program called GRIND, for GReat INtegrator Differential equations (`http://tbb.bio.uu.nl/rdb/grind.html`). This afternoon you will work with an R-script called `grind.R` that is easier to install, and can perform very similar phase plane analysis (`http://tbb.bio.uu.nl/rdb/grindR.html`). Thanks to the R-packages `deSolve` and `rootSolve` developed by Karline Soetaert and colleagues [7, 8, 10], it was relatively easy to copy most of GRIND's capabilities into R. People liking R may also like this simple interface to phase plane analysis. Thanks to the `FME` package, also developed by Karline Soetaert and colleagues [9], it was also relatively easy to extend GRIND with non-linear parameter estimation. This resulted in an R-script `grind.R` defining five easy-to-use functions:

- `run()` integrates a model numerically and provides a time plot or a trajectory in the phase plane,
- `plane()` draws nullclines and can provide a vector field or phase portrait,
- `newton()` finds steady states (using the Newton-Raphson method) and can provide the Jacobian with its eigenvalues and eigenvectors.
- `continue()` continues a steady state along a parameter, thus providing a bifurcation diagram.
- `fit()` fits a model to data by estimating its parameters, and depicts the result in a timeplot.

The `run()` function calls `ode()` from the `deSolve` library, the `fit()` function calls `modFit` from the `FME` library, and `newton()` and `continue()` call `steady()` from the `rootSolve` library. Typing `?ode`, etcetera, provides help on the options of these library functions. The following section is a tutorial illustrating the usage of the `grind.R`'s phase-plane functions. Go for the exercise as soon as you haved completed the tutorial. The full manual of `grind.R` is available on the website `http://tbb.bio.uu.nl/rdb/practicals/grindR/`.

The best way to get started is to download our example analyzing the Lotka Volterra model. We will work in the RStudio environment. If you work on your own laptop you will need to install the Soetaert libraries by using `Install Packages` in the `Tools` menu of RStudio. (Experts may prefer to type `install.packages(c("deSolve", "FME", "rootSolve")` in the R-console). Download the R-scripts `grind.R` and `lotka.R` from the website `http://tbb.bio.uu.nl/rdb/practicals/grindR/` in a local directory, and open both of them via the `File` menu. It may anyway be useful to set the working directory to the folder where your R-codes are stored (`Set working directory` in the `Session` menu of RStudio). Files with then be opened and saved in that directory.

First "Source" the `grind.R` file (button in right hand top corner) to define the functions, then "Run" the model with its parameter and state definitions from the `lotka.R` script. In the R-console below the `lotka.R` panel, one can then type the function calls given in the example session below. Once you have a picture that you like, you may copy the lines creating that figure into the `lotka.R` window for later usage. (Use "Run" or "Control Enter" to execute lines from the `lotka.R` panel into the console).

Lotka Volterra model

The ODEs of the model are defined in the simple notation defined for the `deSolve` package. The following is an example of the Lotka Volterra model, here defined by the function `model()`. This R-script is available as the file `lotka.R` on the website `tbb.bio.uu.nl/rdb/practicals/grindR/`:

```
model <- function(t, state, parms) {
  with(as.list(c(state,parms)), {

    dR <- r*R*(1 - R/K) - a*R*N
    dN <- c*a*R*N - delta*N

    return(list(c(dR, dN)))
  })
}

p <- c(r=1,K=1,a=1,c=1,delta=0.5) # p is a named vector of parameters
s <- c(R=1,N=0.01)                # s is the state
```

where the two lines below the function define the parameter values in the vector `p`, and the initial state of the variables in the vector `s`. Note that the function returns a list of derivatives `(dR,dN)`. The names `model`, `s`, and `p` are the default designations for the model, state, and parameter values in all `grind.R` functions. This example should be self explanatory as it just defines the Lotka Volterra model $\mathrm{d}R/\mathrm{d}t = rR(1 - R/K) - aRN$, and $\mathrm{d}N/\mathrm{d}t = caRN - \delta N$, with its parameter values and initial state as R-vectors, `p <- c(r=1,K=1,a=1,c=1,d=1,delta=0.5)`, and `s <- c(R=1,N=0.01)`, respectively. The following tutorial is an example session illustrating the usage of the first four `grind.R` functions, by simulating and analyzing this Lotka Volterra model:

```
run()                                  # run the model and make a timeplot
plane()                                # make a phase plane with nullclines
plane(xmin=-0.001,ymin=-0.001)         # include the full axis in the phase plane
plane(tstep=0.5,portrait=T)            # make a phase portrait
plane()                                # make a clean phase plain again
p["K"] <- 0.75                         # change the parameter K from 1 to 0.75
plane(add=T)                           # add the new nullclines
s["R"] <- 0.1                          # change the initial state to (R=0.1,N=0.01)
run(traject=T)                         # run the model and plot a trajecory
newton(c(R=0.5,N=0.5),plot=T)          # find a steady state around (R=0.5,N=0.5)
f <- newton(c(R=0.5,N=0.5))            # store this steady state in f
continue(f,x="K",xmax=2,y="N")         # continue this steady state while varying K
p["K"] <- 0.5                          # set K to the value at which N goes extinct
plane(vector=T)                        # make a phase plane for this value of K
```

Cattle in the Sahel

Overgrazing by cattle in arid areas is known to lead to desertification. In the Sahel zone one may find both barren areas and vegetated areas in the same region. This bistability has been studied with models having saddle-node bifurcations [2–4]. We here study a toy model describing the dynamics of water uptake in arid zones [2, 4]. The main idea is that in areas with little vegetation coverage most of the (sometimes heavy) rainfall fails to penetrate into the soil, and is rapidly washed off into rivers and disappears. In areas with somewhat more vegetation water is better captured, but vegetation also consumes water by growth. Models for vegetation growth in arid areas can remain simple because the availability of water in the soil is typically the major limiting factor. This main idea is translated into the following model:

$$\frac{\mathrm{d}W}{\mathrm{d}t} = R\Big(w_0 + \frac{V}{k_2 + V}\Big) - r_W W - \frac{gVW}{k_1 + W} \ ,$$

$$\frac{\mathrm{d}V}{\mathrm{d}t} = i + \frac{cgVW}{k_1 + W} - dV \ ,$$

where R is the rainfall (in mm d^{-1}), V is the vegetation biomass (in g m^{-2}), and W is the amount of water in the soil (mm). The model has two saturation constants, k_1 (mm) defines the amount of water at which the vegetation grows at half its maximal rate, and k_2 (g m^{-2}) is the vegetation cover at which the penetration of water into the soil is $R(w_0 + 1/2) = 1.4$ mm d^{-1}. Parameters of this model have been estimated by Rietkerk and Van de Koppel [4] and HilleRisLambers *et al.* [2]:

Name	interpretation	value	dimension
R	rainfall	2	mm d^{-1}
c	conversion of water to plant biomass	10	g $mm^{-1}m^{-2}$
d	death rate of vegetation	0.25	d^{-1}
g	maximum water uptake	0.05	mm g^{-1} m^2 d^{-1}
k_1	half saturation constant	5	mm
k_2	half saturation constant	5	g m^{-2}
r_W	soil water loss due to evaporation	0.2	d^{-1}
w_0	water infiltration in absence of vegetation	0.2	—
i	immigration of seeds/plants	0.01	g m^{-2} d^{-1}

In the absence of a vegetation there is about 2 mm water in the soil. The death rate of the vegetation is partly due to normal turnover and partly due to grazing by cattle. In the absence of cattle the vegetation turnover is about $d = 0.05$ d^{-1}, and the vegetation reaches a carrying capacity of approximately 450 g m^{-2}. Assume that a single cow per farm corresponds to a grazing rate of 0.025 d^{-1}. Buying cows in the model should increase d, e.g., `p["d"]<-0.075`, and selling cows is the reverse. The model and its parameters are available on the website `http://tbb.bio.uu.nl/rdb/practicals/toy/` under the name `cattle.R`:

```
model <- function(t, state, parms) {
  with(as.list(c(state,parms)), {
    dW <- R*(w0 + V/(k2 + V)) - rW*W -  g*V*W/(k1 + W)
    dV <- i + c*g*V*W/(k1 + W) - (d + eps*t)*V
    return(list(c(dW, dV)))
  })
}

p <- c(R=2,c=10,d=0.25,g=0.05,i=0.01,k1=5,k2=5,rW=0.2,w0=0.2,eps=0)
s <- c(W=5,V=100)
plane(0,10,0.01,1000,log="y")
```

which defines the two ODEs, and a logarithmic axis for the vegetation because the vegetation grows exponentially and covers a wide range of densities. For `eps=0` this model is identical to the ODEs defined above, setting `eps>0` allows one to increase the herd size over time.

a. The first line of the script draws nullclines for a herd size of 8 cows per farm, i.e., for `d=0.25`. Why is the water nullcline starting at 2 mm of water in the soil? Why is the vegetation nullcline ending at 5 mm of water in the soil?

b. Identify the stability of all steady states and give each of them a biological interpretation.

c. Make a phase portrait by the `plane(...,portrait=T)` command. Which of the two variables has the fastest time scale, and is that reasonable?

d. Study the effect of increasing and decreasing the herd size by buying and selling cows. Make a sketch of the expected vegetation cover as a function of the death and grazing rate d.

e. Make a bifurcation diagram by finding the three steady states (using `newton()`) and following each of them using d as a bifurcation parameter (using `continue()`). For instance:

```
fhig <- newton(c(W=5,V=50),plot=T)
fmid <- newton(c(W=5,V=2),plot=T)
flow <- newton(c(W=2,V=0.1),plot=T)
continue(flow,x="d",xmax=0.5,y="V",ymin=0.01,ymax=1000,log="y")
continue(fmid,x="d",xmax=0.5,y="V",ymin=0.01,ymax=1000,log="y",add=T)
continue(fhig,x="d",xmax=0.5,y="V",ymin=0.01,ymax=1000,log="y",add=T)
```

f. Perform a similar analysis for the effect of the rainfall. What is the effect on the vegetation of a few years of low, or high, rainfall, and how does this depend on the herd size?

g. Scheffer *et al.* [6] give a short overview of the current approaches to predict catastrophic bifurcations in noisy biological systems (see also the book by Scheffer [5]). Have a look at the paper and study the boxes on the underlying theory.

h. Apply their approach to the Sahel model by first adding noise to the rainfall, using the `grind.R`-option `after`, that is executed after each time-step, and is here calling the R-function `rnorm(n,mean,stdev)` for drawing random numbers:

```
after <- "parms[\"R\"]<-abs(rnorm(1,2,0.2))"
run(after=after,traject=T)
```

Next study the behavior of the model by very slowing changing the herd size starting in the upper steady state. This can be done by setting the parameter `eps` to a non-zero value:

```
s <- c(W=5,V=100)
plane(0,10,0.01,1000,log="y")
f <- newton(plot=T)
p["eps"] <- 0.0001
run(1000,state=f,after=after,traject=T)
```

i. Do you obtain an early warning signal? What if the herd size changes even more slowly?

j. Do you receive an early warning signal in another variable of the model?

k. In the Scheffer *et al.* [6] paper they explain in a box how autocorrelations can be used as a predictor. You can compute and visualize autocorrelations by saving the data from a run: `data <- run(table=T)`, and then call something like:

`plot(data$V[1:nrow(data)-1],data$V[2:nrow(data)],type="p")`, and/or
`cor(data$V[1:nrow(data)-1],data$V[2:nrow(data)])`.

l. Do you think you would be able to predict a catastrophic bifurcation, and what would be the best approach to detect this?

m. If you know about the eigenvalues of the Jacobian matrix, study them in the neighborhood of the bifurcation point by calling `newton()`, and predict in what variable to expect to see most the early warning signals. Later this afternoon you could read the paper by Boerlijst *et al.* [1] to learn when early warning signals are expected to be present.

June 4, 2017

References

[1] **Boerlijst, M. C., Oudman, T., and De Roos, A. M.**, 2013. Catastrophic collapse can occur without early warning: examples of silent catastrophes in structured ecological models. PLoS. One. **8**:e62033.

[2] **HilleRisLambers, R., Rietkerk, M., Van den Bosch, F., Prins, H. H. T., and De Kroon, H.**, 2001. Vegetation pattern formation in semi-arid grazing systems. Ecology **82**:50–61.

[3] **Noy-Meir, I.**, 1975. Stability of grazing systems: an application of predator-prey graphs. J. Ecology **63**:459–483.

[4] **Rietkerk, M. and Van de Koppel, J.**, 1997. Alternate stable states and threshold effects in semi-arid grazing systems. Oikos **79**:69–76.

[5] **Scheffer, M.**, 2009. Critical Transitions in Nature and Society. Princeton U.P., Princeton.

[6] **Scheffer, M., Bascompte, J., Brock, W. A., Brovkin, V., Carpenter, S. R., Dakos, V., Held, H., Van Nes, E. H., Rietkerk, M., and Sugihara, G.**, 2009. Early-warning signals for critical transitions. Nature **461**:53–59.

[7] **Soetaert, K.**, 2009. rootSolve: Nonlinear root finding, equilibrium and steady-state analysis of ordinary differential equations. R package 1.6.

[8] **Soetaert, K. and Herman, P. M.**, 2009. A Practical Guide to Ecological Modelling. Using R as a Simulation Platform. Springer. ISBN 978-1-4020-8623-6.

[9] **Soetaert, K. and Petzoldt, T.**, 2010. Inverse modelling, sensitivity and monte carlo analysis in R using package FME. Journal of Statistical Software **33**:1–28.

[10] **Soetaert, K., Petzoldt, T., and Setzer, R. W.**, 2010. Solving differential equations in R: Package desolve. Journal of Statistical Software **33**:1–25.