Simulations with MatCont

Fabio Della Rossa

Analyze the Rosenzweig-MacArthur model

$$\begin{aligned}\dot{x}_1 &= rx_1(1-\frac{x_1}{K}) - \frac{ax_1}{b+x_1}x_2 \\ \dot{x}_2 &= e\frac{ax_1}{b+x_1}x_2 - mx_2\end{aligned}$$

where x_1 , and x_2 are the prey and predator densities, $r = m = 2\pi$, $a = 4\pi$, $K = e = 1$ and $b \in \{0.2, 0.5, 1.1\}$.

- Show that the model is positive, i.e. $(x_1(0), x_2(0)) \geq 0 \Rightarrow (x_1(t), x_2(t)) \geq 0$.
- Analyze the model dynamic in absence of predators ($x_2 = 0$) and in absence of preys ($x_1 = 0$).
- Locate the equilibria of the system, and discuss their stability through linearization.
- Let b = 0.2, and sketch the trajectories of the system in the neighborhood of the equilibria.
- Discuss the existence of limit cycles.
- Sketch a possible full state portrait.
- Verify the obtained results using MatCont, and repeat the analysis for the different values of b.

Let now assume that the predation half saturation constant varies with a seasonality (this can happen due to a different ability of the preys to hide themselves from the predators), i.e.

$$b = b_0(1 + \varepsilon \sin \frac{\pi}{2} t).$$

Simulate the system with `MatCont`[1] and show that the asymptotic behaviour of the system is the one reported in the following table, for different values of (b_0, ε):

ε \ b_0	0.2	0.5	1.1
0	periodic	stationary	extinction
0.1	quasi-periodic	periodic	extinction
0.7	chaotic	chaotic	periodic

[1] Notice that `MatCont` can only analyze autonomus systems, so we need to generate the sinosoidial forcing by means of the oscillatior

$$\begin{aligned}\dot{x}_3 &= x_3 - \omega x_4 - (x_3^2 + x_4^2)x_3 \\ \dot{x}_4 &= \omega x_3 + x_4 - (x_3^2 + x_4^2)x_4\end{aligned}$$

with $[x_3(0), x_4(0)] = [1, 0]$, and substitute $\sin \omega t$ with variable x_3.

In particular

- show the projection of the attractor in the space $(x_1, x_2, \sin\frac{\pi}{2}t)$
- in the case $(b_0, \varepsilon) = (0.2, 0.7)$ verify the sensitivity from initial condition by plotting two temporal series of x_1 starting from close initial values.
- in the cases $(b_0, \varepsilon) = (0.2, 0.7)$, and $(b_0, \varepsilon) = (0.5, 0.4)$, analyse the Poincarè section[2]and compute the Lyapunov Exponents[3]of the attractor.

---

[2] To compute the Poincarè section of the attractor we need to simulate the system using the event detection feature of theODE package. Opening the system file generated by MatCont, define a new function that changes sign by crossing the Poincarè Section. For example:

```
function [T,Y,TE,YE,IE] = poincare_section(odefun,event,tspan,y0)
t0=tspan(1);
t1=tspan(2);
options=[];
[T,Y] = ode45(odefun,[t0,t1/10],y0,options); % Leave the transient
options=odeset('Events',event);
[T,Y,TE,YE,IE] = ode45(odefun,[t1/10,t1],Y(end,:),options);
figure, line(YE(:,1),YE(:,2),'linestyle','none','marker','.','markersize',10)

function [value,isterminal,direction]=events(t,x,KK,RR,AA,B0,EE,DD,epsilon)
value=x(3);
isterminal=0;
direction=1;

function dydt = fun_eval(t,kmrgd,KK,RR,AA,B0,EE,DD,epsilon)
dydt=...;
```

[3] A function that computes the Lyapunov exponents:

```
function [Texp,Lexp]=lexp(odefun,jacobian,tspan,y0)
stept=0.2;
ioutp=100;
n1=length(y0); n2=n1*(n1+1);
nit = round(diff(tspan)/stept); % Number of steps
% Memory allocation
y=zeros(n2,1); cum=zeros(n1,1);
Lexp=zeros(n1,nit); Texp=zeros(1,nit);
% Initial values
rhs_ext=@(t,x) [odefun(t,x);reshape(jacobian(t,x)*reshape(x(n1+1:n2),n1,n1),n2-n1,1)];
y=[y0(:);reshape(eye(n1),n1^2,1)];
t=tspan(1);

% Main loop
for ITERLYAP=1:nit
[T,Y] = ode45(rhs_ext,[t t+stept],y); % Solutuion of extended ODE system
t=t+stept; y=Y(size(Y,1),:); % Take the last computed point
[Q,R]=qr(reshape(y(n1+1:n2),n1,n1)); % Construct new orthonormal basis
y(n1+1:n2)=Q(:);
cum=cum+log(abs(diag(R))); % Compute lyapunov coefficient
lp=cum/(t-tspan(1)); % normalize exponent
Lexp(:,ITERLYAP)=lp; Texp(ITERLYAP)=t;
if (mod(ITERLYAP,ioutp)==0)
fprintf('t=%6.4f ',t); fprintf('%10.6f ',lp); fprintf('\n');
end;
end;
figure, plot(Texp,Lexp)
```