Simulations with MatCont

Fabio Della Rossa

Analyze the Rosenzweig-MacArthur model

\[
\begin{align*}
\dot{x}_1 &= r x_1 \left(1 - \frac{x_1}{K}\right) - \frac{a x_1}{b + x_1^3} x_2 \\
\dot{x}_2 &= e^{\frac{a x_1}{b + x_1^3}} x_2 - mx_2
\end{align*}
\]

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.

\[ b = b_0 (1 + \varepsilon \sin \frac{\pi}{2} t), \quad b_0 = 2. \]

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 \((b_0, \varepsilon)\):

<table>
<thead>
<tr>
<th>(\varepsilon)</th>
<th>(b_0)</th>
<th>0.2</th>
<th>0.5</th>
<th>1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Periodic</td>
<td>Stationary</td>
<td>Extinction</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>Quasi-periodic</td>
<td>Periodic</td>
<td>Extinction</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>Chaotic</td>
<td>Chaotic</td>
<td>Periodic</td>
<td></td>
</tr>
</tbody>
</table>

In particular
- show the projection of the attractor in the space \((x_1, x_2, \sin \frac{\pi}{2} t)\)
- in the case \((b_0, \varepsilon) = (2, 0.7)\) verify the sensitivity from initial condition by plotting two temporal series of \(x_1\) starting from close initial values.

\(^1\)Notice that MatCont can only analyze autonomous systems, so we need to generate the sinusoidal forcing by means of the oscillator

\[
\begin{align*}
\dot{x}_3 &= x_3 - \omega x_4 - (x_3^2 + x_4^2) x_3 \\
\dot{x}_4 &= \omega x_3 + x_4 - (x_3^2 + x_4^2) x_4
\end{align*}
\]

with \([x_3(0), x_4(0)] = [1, 0]\), and substitute \(\sin \omega t\) with variable \(x_3\).
• in the cases \((b_0, \varepsilon) = (2, 0.7)\), and \((b_0, \varepsilon) = (0.5, 0.4)\), analyse the Poincaré section and compute the Lyapunov Exponents of the attractor.

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

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

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

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

A function that computes the Lyapunov exponents:

```matlab
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);
  y=zeros(n2,1); cum=zeros(n1,1);
  Lexp=zeros(n1,nit); Texp=zeros(1,nit);
  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),n1ˆ2,1)];
  t=tspan(1);
  for ITERLYAP=1:nit
    [T,Y] = ode45(rhs_ext,[t,t+stept],y); % Solution 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(:,1);
    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('\verb+\n');
    end;
  end;
  figure, plot(Texp,Lexp)
end
```