MATHEMATICAL AND COMPUTER MODELING OF NONLINEAR BIOSYSTEMS I

COMPUTER LABORATORY VIII: Zeeman model for the heartbeat

Ph. D. Programme 2013/2014

PhD STUDIES

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Project co-financed by European Union within the framework of European Social Fund

Zeeman model for the heartbeat

The Zeeman model for the heartbeat reads

$$\begin{cases} \dot{b} = x - x_1 \,, \\ \varepsilon \dot{x} = -(x^3 - Tx + b) \,, \end{cases}$$

where all parameters are positive.

Parameter T is the muscle tension and is related to blood pressure (the higher is blood pressure the higher is the muscle tension).

Parameter x_1 is the average muscle length in the diastole (the period of time when the heart refills with blood after contraction).

Exercise 1 - phase portrait

Implement MATLAB function for drawing phase portrait of the Zeeman model

$$\begin{cases} \dot{b} = x - x_1 \,, \\ \varepsilon \dot{x} = -(x^3 - Tx + b) \,. \end{cases}$$

Requirements

- null-clines, exemplary solutions and arrows showing the direction of the vector field should be plotted;
- input arguments:
  1. the ranges for variables b and x;
  2. number of mesh points (in one direction) in which arrows will be plotted;
  3. model parameters in a structure.

Exercise 1 - solution, step 1

Function returning solution to the model

```
function sol = solutionZeeman( Tend, par, init )

    opt = odeset('RelTol',1e-8,'AbsTol',1e-8);
    sol = ode45(@RHS,[0 Tend],init,opt);

    function dy = RHS(~,x)
       dy=zeros(2,1);
       dy(1)=(x(2)-par.x0);
       dy(2)=-(x(2)^3-par.T*x(2)+x(1))/par.eps;
    end

end
```

Exercise 1 - solution, step 2

Function structure

```
function phasePortrait(bR,xR,N,par)

    figure(1)
    clf
    hold on

%PLOT HERE

    hold off

    xlabel('b')
    ylabel('x')
    xlim(bR)
    ylim(xR)

end
```

Exercise 1 - solution, step 3

Drawing null-clines

```
plot([bR(1) bR(2)],[par.x0 par.x0],'k',...
     'LineWidth',2,'LineStyle','--')

xmesh = linspace(xR(1),xR(2),100);

plot(-xmesh.^3+par.T*xmesh,xmesh,'k',...
     'LineWidth',2,'LineStyle','--')
```

Exercise 1 - solution, step 4

Drawing vector field arrows

```
h1 = (xR(2)-xR(1))/(N-1);
h2 = (bR(2)-bR(1))/(N-1);

[x,y] = meshgrid(bR(1):h2:bR(2),xR(1):h1:xR(2));
[u, v] = vectorField(x,y);
quiver(x,y,u,v,1)

function [u, v] = vectorField(x,y)
   u=(y-par.x0);
   v=-(y.^3-par.T*y+x)/par.eps;
end
```

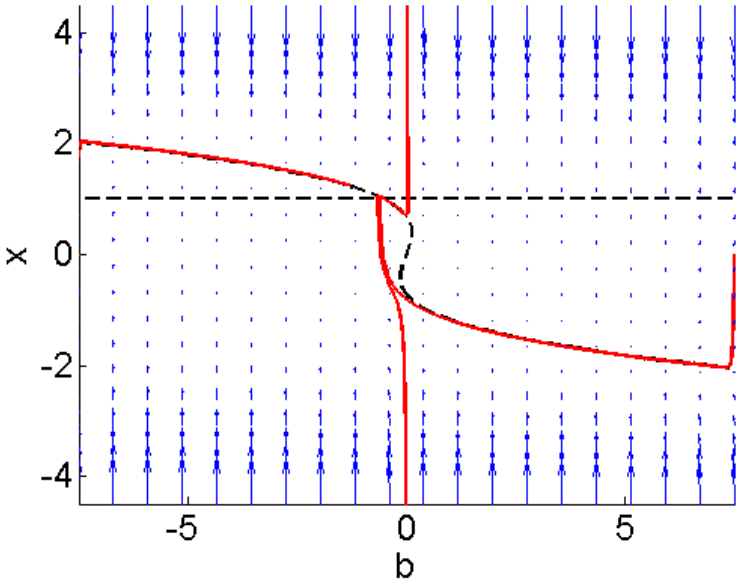
Exercise 1 - solution, step 5

Drawing exemplary trajectories

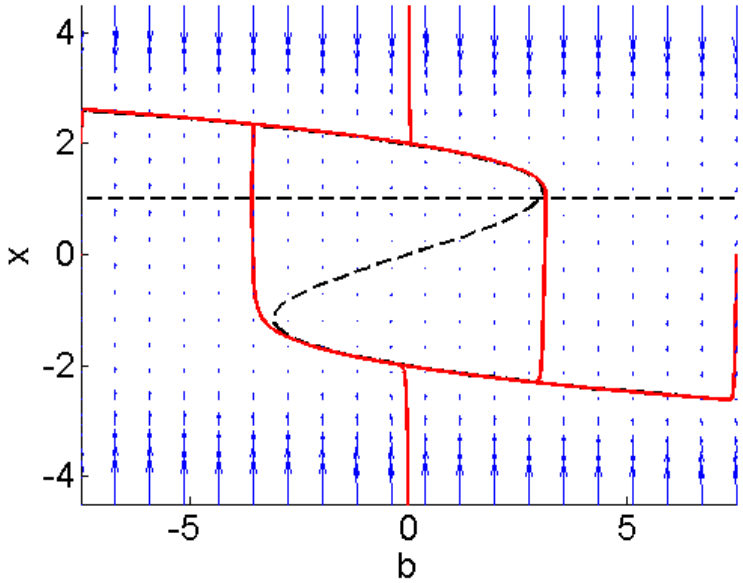
```
for i=1:2

    sol = solutionZeeman( 10, par, [bR(i), 0]);
    plot(sol.y(1,:),sol.y(2,:),'b','LineWidth',2);

    sol = solutionZeeman( 10, par, [0, xR(i)]);
    plot(sol.y(1,:),sol.y(2,:),'b','LineWidth',2);

end
```

Exemplary phase plots

$x_1 = 1,\ \varepsilon = 0.1,\ T = 0.5$



$x_1 = 1,\ \varepsilon = 0.1,\ T = 4$



For low blood pressure (small T) we have asymptotically stable node, i.e. there is **no heartbeat.**

For sufficiently large pressure (large T) we have limit cycle, i.e. there is a **heartbeat.**

Peacemaker

Experiments revealed that heart beats slowly even in a zero pressure conditions (for $T = 0$).

We saw that the Zeeman model fails to predict that behaviour.

We need to modify the model to include the peacemaker

$$\begin{cases} \dot{b} = (x - x_0) + u(x_0 - x_1)\,, \\ \varepsilon \dot{x} = -(x^3 - Tx + b)\,, \end{cases}$$

where $u = 1$ when

- $||(b, x) - (b_0, x_0)|| < \delta$
- $b > b_1$
- $||(b, x) - (b_1, x_1)|| \geqslant \delta$ and $b_0 \leqslant b \leqslant b_1$ and $x^3 - Tx + b > 0$

and $u = 0$ otherwise. δ is additional small parameter and b_0 (b_1) is steady state for $u = 0$ ($u = 1$).

Exercise 2 - solution with peacemaker

Implement MATLAB function for solving the Zeeman model with the peacemaker modification.

Exercise 2 - **bad** solution

```
function sol = solutionZeemanMod( Tend, par, init )

    b0 = -par.x0^3+par.T*par.x0;
    b1 = -par.x1^3+par.T*par.x1;

    sol = ode45(@RHS,[0 Tend],init);

    function dy = RHS(~,x)
       dy=zeros(2,1);
       dy(1)=(x(2)-par.x0)+peacemaker(x)*(par.x0-par.x1);
       dy(2)=-(x(2)^3-par.T*x(2)+x(1))/par.eps;
    end

    function u = peacemaker(y)
       u = (norm(y-[b1; par.x1])>=par.delta)&&(y(1)>=b0)&&...
           (y(1)<=b1)&&((y(2)^3-par.T*y(2)+y(1))>0)...
            || y(1)>b1 || (norm(y-[b0; par.x0])<par.delta);
    end
end
```

Exercise 2 - **correct** solution, part 1

```
function sol = solutionZeemanMod2( Tend, par, init)

    b0 = -par.x0^3+par.T*par.x0;
    b1 = -par.x1^3+par.T*par.x1;

    u = peacemaker(init); t = 0; tstep = 2;
    sol.y = [init]; sol.x = [0];

    opt = odeset('Events', @events);

    function [value,isterminal,direction] = events(~,y)
      value = [norm(y-[b0; par.x0])-par.delta;...
               norm(y-[b1; par.x1])-par.delta;...
               y(1)-b0;...
               y(1)-b1;...
               y(2)^3-par.T*y(2)+y(1)];
      isterminal = zeros(size(value));
      direction = zeros(size(value));
    end
```

Exercise 2 - **correct** solution, part 2

```
while t<Tend
   solTmp = ode45(@RHS,[t t+tstep],init,opt);

   if ~isempty(solTmp.xe)
     c = false;
      for ii = 1:length(solTmp.xe)
         if ii == length(solTmp.xe)
           cS = solTmp.y(:,end);
         else
           cS = deval(solTmp,(solTmp.xe(ii)+solTmp.xe(ii+1))/2);
         end
            u2 = peacemaker(cS);
            if u2~=u && solTmp.xe(ii)>t+1e-6
                u = u2;
                c=true;
                break;
            end
       end
```

Exercise 2 - **correct** solution, part 3

```
if c
    t = solTmp.xe(ii);
    init = solTmp.ye(:,ii);
    ni = solTmp.x<=solTmp.xe(ii);
    sol.x = [sol.x solTmp.x(ni(2:end))];
    sol.y = [sol.y solTmp.y(:,ni(2:end))];
else
    t = solTmp.x(end);
    init = solTmp.y(:,end);
    sol.x = [sol.x solTmp.x(2:end)];
    sol.y = [sol.y solTmp.y(:,2:end)];
end
```

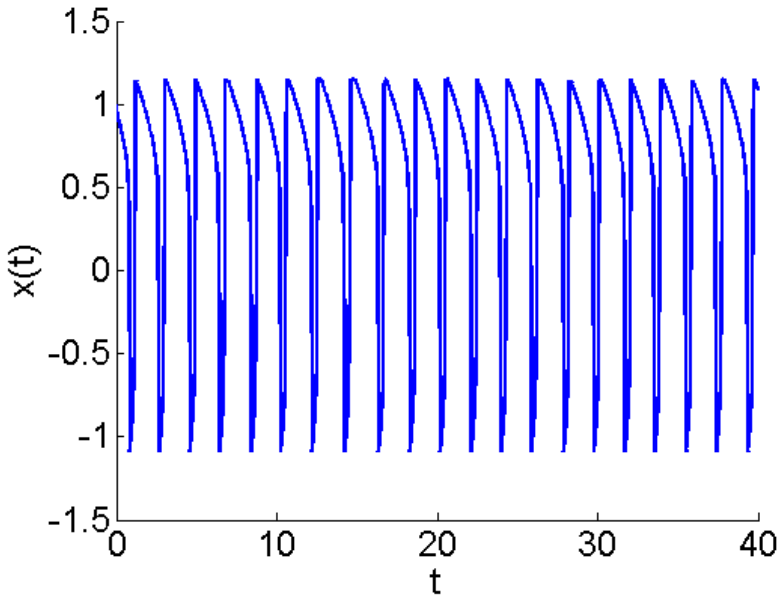
Exercise 2 - **correct** solution, part 4

```
    else
       t = solTmp.x(end);
       init = solTmp.y(:,end);
       sol.x = [sol.x solTmp.x(2:end)];
       sol.y = [sol.y solTmp.y(:,2:end)];
    end
end %end while

     function dy = RHS(~,x)
        dy=zeros(2,1);
        dy(1)=(x(2)-par.x0)+u*(par.x0-par.x1);
        dy(2)=-(x(2)^3-par.T*x(2)+x(1))/par.eps;
     end
     function u = peacemaker(y)
        u = (norm(y-[b1; par.x1])>=par.delta)&&(y(1)>=b0)&&...
            (y(1)<=b1)&&((y(2)^3-par.T*y(2)+y(1))>0)...
             || y(1)>b1 || (norm(y-[b0; par.x0])<par.delta);
     end
end
```

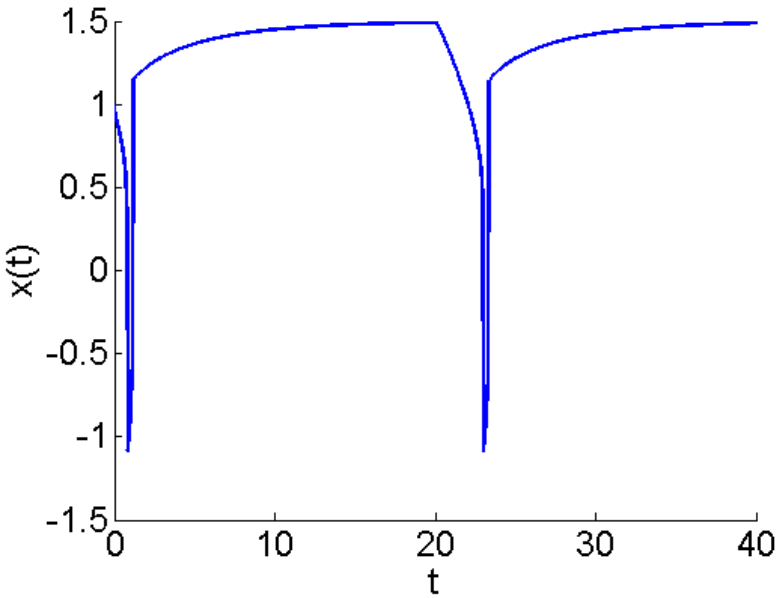
Comparizon of the solutions

Zero tension condition: $T = 0$

Wrong solution



Correct solution

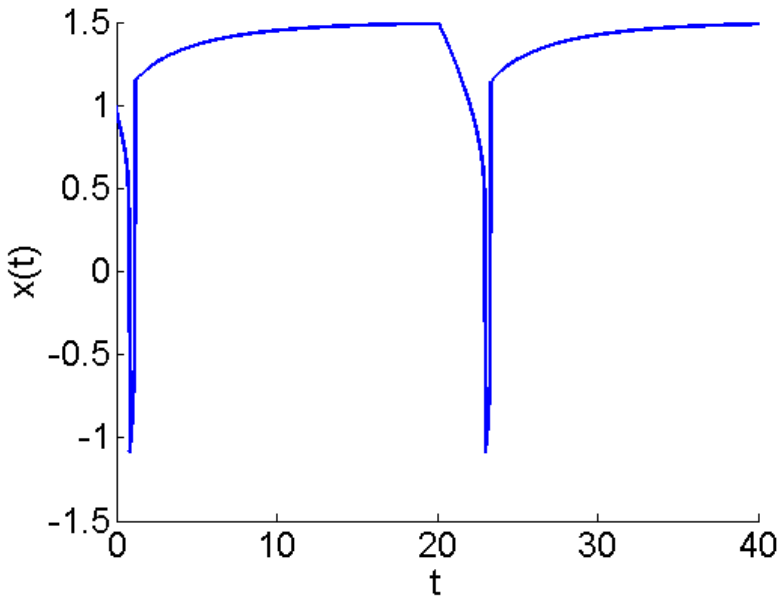


There is a heartbeat, but numerical methods predict different period.

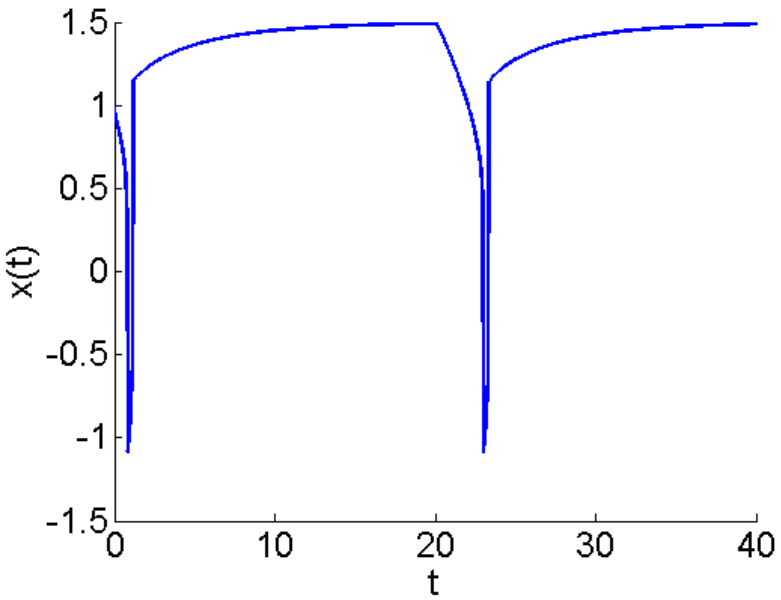
Comparizon of the solutions

Zero tension condition: $T = 0$

Wrong solution
(increased solver accuracy)



Correct solution



There is a heartbeat and numerical methods predict similar period.