Kompetitív Lotka-Volterra modell

Két egymással versengő populáció dinamikájának modellje.

(Pl. egy kovász kultúra és egy invazívabb élesztőgomba kultúra. Állítólag az élesztőgombák kiszorítják a kovászt, valószínűleg azért mert az élesztőgombák gyorsabban szaporodnak.)

```
syms t x y mu real
f = [
    x*(1 - x/2) - x*y
    y*(mu - y) - x*y
    ]
```

f =

$$\begin{pmatrix} -x\left(\frac{x}{2}-1\right)-x\,y \\ y\,(\mu-y)-x\,y \end{pmatrix}$$

```
x = [x;y];

f_fh = matlabFunction(f,'vars',{x,mu});             % later used for ODE solver
J_fh = matlabFunction(jacobian(f,x),'vars',{x,mu}); % compute trace and determinant of Jacobian
f1_fh = matlabFunction(f(1),'vars',[x;mu]);         % to plot vector field
f2_fh = matlabFunction(f(2),'vars',[x;mu]);         % to plot vector field
```

Egy másik reprezentációja a Lotka-Volterra egyenletnek:

$$\begin{cases} \dot{x}_1 = x(a_{11}x_1 + a_{12}x_2 + b_1) \\ \dot{x}_2 = y(a_{21}x_1 + a_{22}x_2 + b_2) \end{cases}$$

Mátrix-vektor formában felírva a következő egyenletet kapjuk:

$\dot{x} = x \odot (Ax + b)$,

ahol az $\odot$ szimbólum a Hadamard (vagyis az elemenkénti) mátrix szorzatot jelöli. Ez a forma azért előnyös mert ebből könnyedén kifejezhető a belső egyensúlyi pont:

$x^* = -A^{-1}b$.

```
A = sym('a',[2,2]), b = sym('b',[2,1]);
```

A =

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

```
EqP = simplify(-A \ b)
```

EqP =

$$\begin{pmatrix} \frac{a_{1,2}\,b_2 - a_{2,2}\,b_1}{a_{1,1}\,a_{2,2} - a_{1,2}\,a_{2,1}} \\ -\frac{a_{1,1}\,b_2 - a_{2,1}\,b_1}{a_{1,1}\,a_{2,2} - a_{1,2}\,a_{2,1}} \end{pmatrix}$$

A következő két sorban $\dot{x} = f(x)$ formából fejezem ki A és b értékeit.

```
A = jacobian( simplify(f ./ x) , x )
```

A =

$$\begin{pmatrix} -\frac{1}{2} & -1 \\ -1 & -1 \end{pmatrix}$$

```
b = simplify( (f - A*x .* x) ./ x )
```

b =

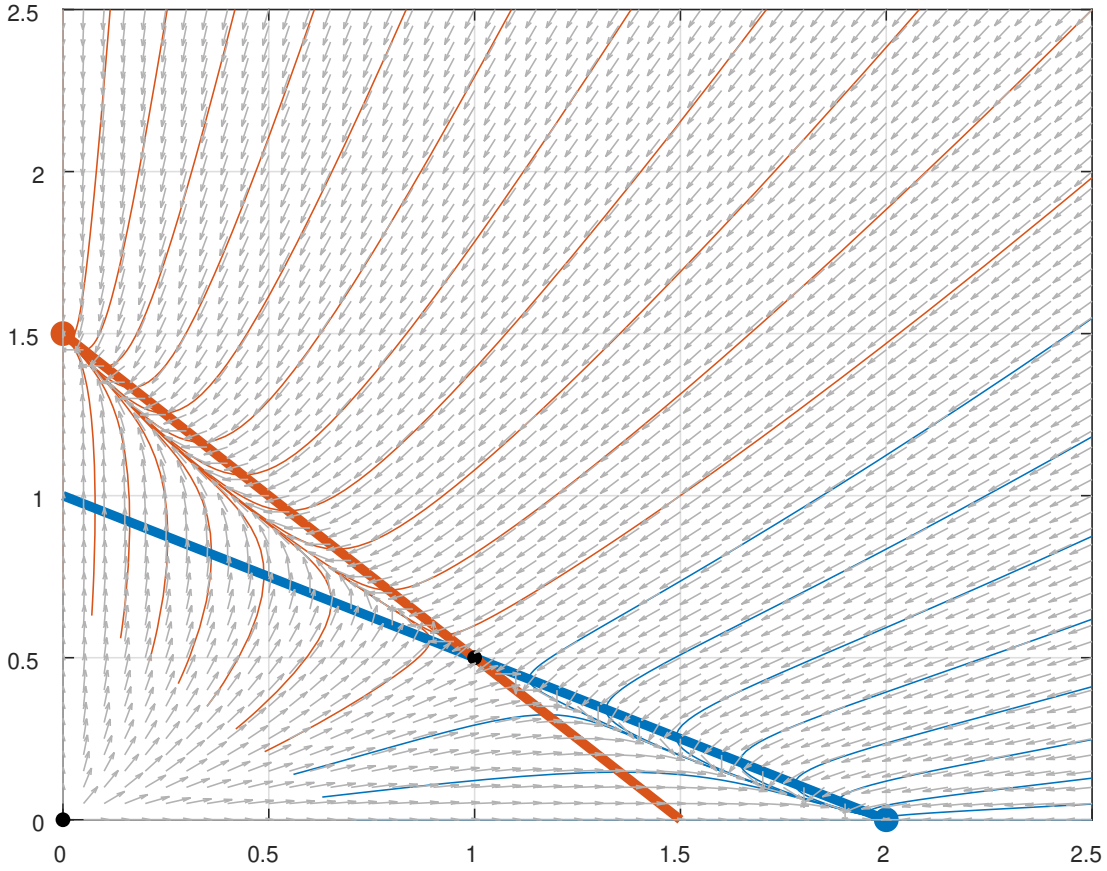
$$\begin{pmatrix} 1 \\ \mu \end{pmatrix}$$

```
A = double(A);
A_ = num2cell(A');
[a11,a12,a21,a22] = deal(A_{:});
b1 = double(b(1));
```

```
LimX = 2.5;
LimY = 2.5;
hX = 0.05;
hY = 0.05;

mu = 1.5;
b2 = mu;

Color_X = [0 0.4470 0.7410];
Color_Y = [0.8500 0.3250 0.0980];

EqX = [-b1/a11 ; 0];
EqY = [0 ; -b2/a22];
EqP = -A \ double(subs(b));

fig = figure(123);
delete(fig.Children);
ax = axes(fig); hold on; grid on; box on

Pl_ode = [];
for alpha = 0:0.1:1
    x0 = [alpha ; 1-alpha]*0.7;
    [t,x] = ode45(@(t,x) f_fh(x,mu),[0 10],x0);
    Pl = plot(x(:,1),x(:,2));
    Pl.UserData.x0 = x0;
    Pl.UserData.xEnd = x(end,:)';
    if isempty(Pl_ode)
        Pl_ode = Pl;
    else
        Pl_ode = [Pl_ode , Pl];
    end
end

for alpha = 0:0.05:1
    x0 = [alpha ; 1-alpha]*5;
    [t,x] = ode45(@(t,x) f_fh(x,mu),[0 10],x0);
```

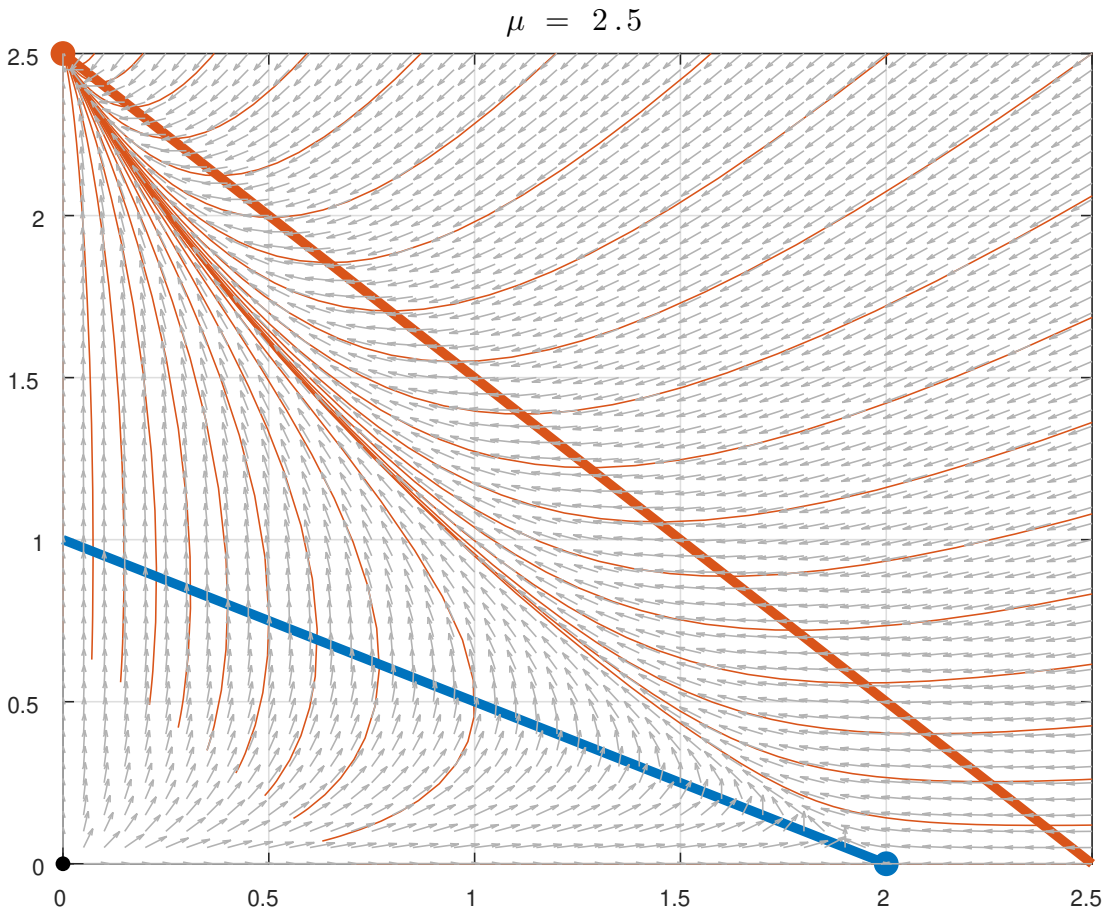
```
    Pl = plot(x(:,1),x(:,2));
    Pl.UserData.x0 = x0;
    Pl.UserData.xEnd = x(end,:)';
    Pl_ode = [Pl_ode , Pl];
end

for Pl = Pl_ode
    if Pl.UserData.xEnd(1) < EqP(1)
        Pl.Color = Color_Y;
    else
        Pl.Color = Color_X;
    end
end

plot(0,0,'k.','MarkerSize',14);
xx = linspace(0,EqX(1),2);
yy = -(b1+a11*xx)/a12;
PlncX = plot(xx,yy,'Color',Color_X,'LineWidth',3);

yy = linspace(0,EqY(2),2);
xx = -(b2+a22*yy)/a21;
PlncY = plot(xx,yy,'Color',Color_Y,'LineWidth',3);

PleX = plot(EqX(1),EqX(2),'.','Color',Color_X,'MarkerSize',25);
PleY = plot(EqY(1),EqY(2),'.','Color',Color_Y,'MarkerSize',25);
PleP = plot(EqP(1),EqP(2),'.k','MarkerSize',14);

% Compute the values of the vector field in a given number of grid points.
[xx,yy] = meshgrid(0:hX:LimX,0:hY:LimY);
f1_val = f1_fh(xx,yy,mu);
f2_val = f2_fh(xx,yy,mu);

% Normalize the vectors of the vector field.
r = sqrt(f1_val.^2 + f2_val.^2);
f1_val = f1_val ./ r;
f2_val = f2_val ./ r;

% Visualize the normalized vector field.
Qv = quiver(xx,yy,f1_val,f2_val,'Color',[1,1,1]*0.7);

Tit = title("$\mu = " + num2str(mu) + "$",'Interpreter','latex','FontSize',14);
xlim([0 LimX])
ylim([0 LimY])
```

$\mu = 1.5$

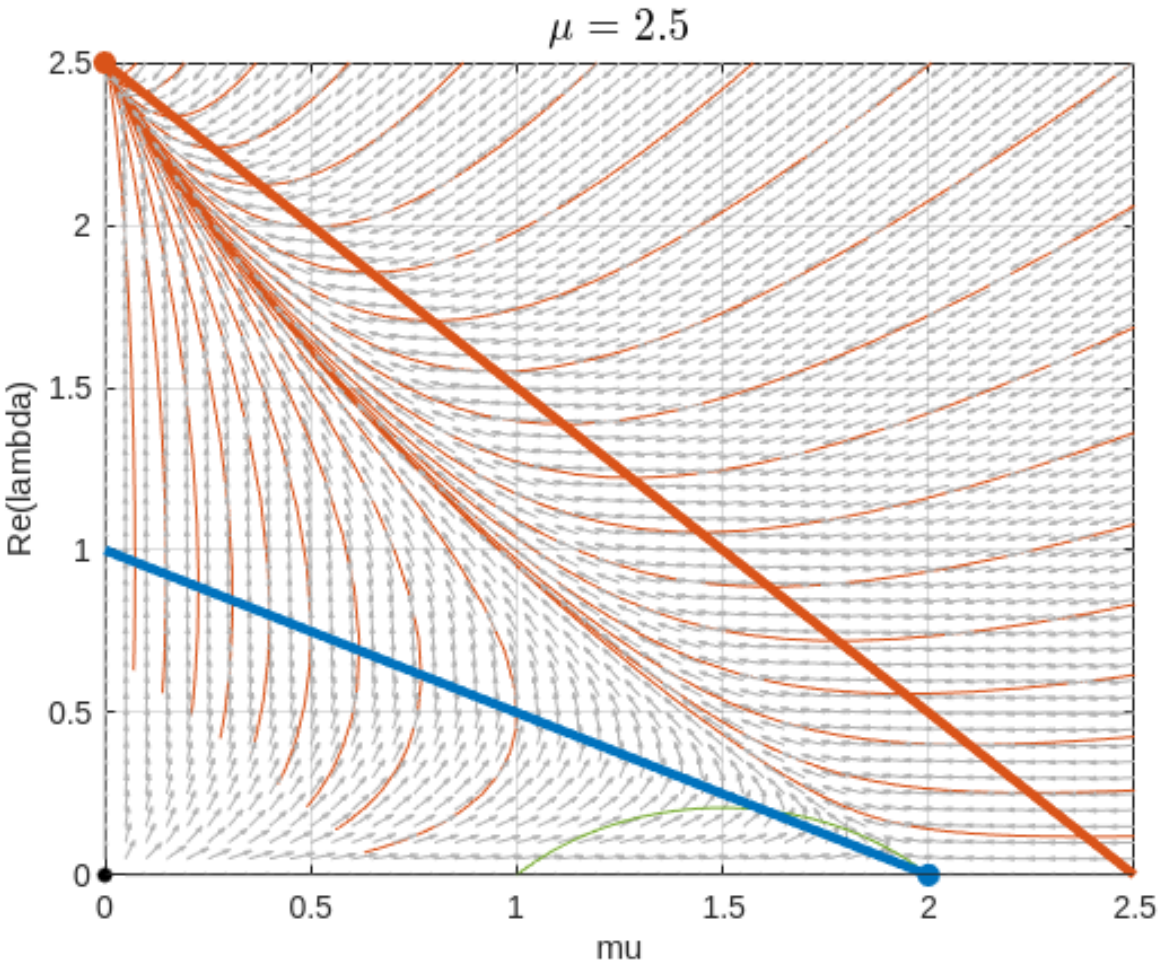



```
mu_vals = 0.5:0.01:2.5;
for i = 1:numel(mu_vals)
    mu = mu_vals(i);
    b2 = mu;

    EqX = [-b1/a11 ; 0];
    EqY = [0 ; -b2/a22];
    EqP = -A \ double(subs(b));

    PlncX.XData = linspace(0,EqX(1),2);
    PlncX.YData = -(b1+a11*PlncX.XData)/a12;

    PlncY.YData = linspace(0,EqY(2),2);
    PlncY.XData = -(b2+a22*PlncY.YData)/a21;

    PleX.XData = EqX(1);
    PleX.YData = EqX(2);

    PleY.XData = EqY(1);
    PleY.YData = EqY(2);

    PleP.XData = EqP(1);
    PleP.YData = EqP(2);
```

```
    if all(EqP > 0)
        PleP.Visible = "on";
    else
        PleP.Visible = "off";
    end

    for Pl = Pl_ode
        [t,x] = ode45(@(t,x) f_fh(x,mu),[0 10],Pl.UserData.x0);
        Pl.XData = x(:,1);
        Pl.YData = x(:,2);
        if x(end,1) < EqP(1)
            Pl.Color = Color_Y;
        else
            Pl.Color = Color_X;
        end
    end

    % Compute the values of the vector field in a given number of grid points.
    [xx,yy] = meshgrid(0:hX:LimX,0:hY:LimY);
    f1_val = f1_fh(xx,yy,mu);
    f2_val = f2_fh(xx,yy,mu);

    % Normalize the vectors of the vector field.
    r = sqrt(f1_val.^2 + f2_val.^2);
    f1_val = f1_val ./ r;
    f2_val = f2_val ./ r;

    % Visualize the normalized vector field.
    Qv.XData = xx;
    Qv.YData = yy;
    Qv.UData = f1_val;
    Qv.VData = f2_val;

    Tit.String = "$\mu = " + num2str(mu) + "$";

    drawnow
    pause(0.1)
end
```




```
mu_vals = 0.5:0.01:2.5;
Eigs = nan(2,numel(mu_vals));
for i = 1:numel(mu_vals)
    mu = mu_vals(i);
    EqP = -A \ double(subs(b));
    J_EqP = J_fh(EqP,mu);
    Eigs(:,i) = eig(J_EqP);
end
Eigs = switch_Eigs(Eigs);

fig = figure(31);
delete(fig.Children)
fig.Visible = 'on';
hold on; box on;
plot3(mu_vals,real(Eigs(1,:)),imag(Eigs(1,:)))
plot3(mu_vals,real(Eigs(2,:)),imag(Eigs(2,:)))
yline(0)
xlabel('mu')
ylabel('Re(lambda)')
zlabel('Im(lambda)')
```




```
function Eigs = switch_Eigs(Eigs)
    n = height(Eigs);
    dim = zeros(1,n);
    for i = 1:width(Eigs)-1
        for d = 1:n
            [~,dim(d)] = min(abs(Eigs(d,i) - Eigs(:,i+1)));
        end
        if any(dim ~= 1:n) && all(sort(dim) == 1:n)
            Eigs(:,i+1:end) = Eigs(dim,i+1:end);
        end
    end
end
```