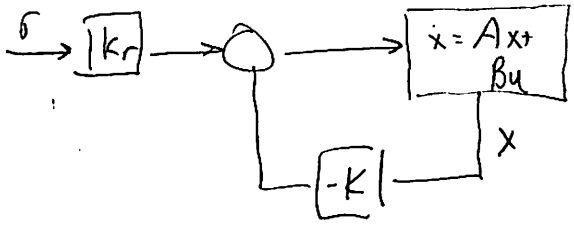
# Lecture 4.2: Reachability & State Feedback

Linear control system design

$$\dot{x} = Ax + Bu$$
$$u = -Kx + k_r r$$




**Thm** If $(A, B)$ is reachable, then we can place eigenvalues of $A - BK$ to any desired values.

Pf (special case)

Suppose $\tilde{A}$ and $\tilde{B}$ are in reachable canonical form

$$\tilde{A} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \\ 1 & 0 & & 0 \\ & 1 & & \vdots \\ & & \ddots & 0 \\ & & & 1 \end{bmatrix} \qquad \tilde{B} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Can show that $\lambda_A(s) = s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n$

Let $\tilde{K} = [\tilde{k}_1 \ \tilde{k}_2 \ \cdots \ \tilde{k}_n]$ be a "control gain matrix"

$$A - BK = \begin{bmatrix} & A & \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \tilde{k}_1 & \cdots & \tilde{k}_n \end{bmatrix} = \begin{bmatrix} -a_1 - \tilde{k}_1 & -a_2 - \tilde{k}_2 & \cdots & -a_n - \tilde{k}_n \\ 1 & 0 & \cdots & 0 \\ & 1 & & \vdots \\ & & \ddots & 0 \end{bmatrix}$$

Closed loop $\Rightarrow \lambda_{A-BK} = s^n + (k_1 - a_1)s^{n-1} + \cdots + (k_n - a_n)$

Suppose desired eigenvalues are $\lambda_1, \lambda_2, \ldots, \lambda_n$

$$\lambda_d(s) = (s-\lambda_1)(s-\lambda_2)\cdots(s-\lambda_n) = s^n + p_1 s^{n-1} + \cdots + p_{n-1} s + p_n$$

$\therefore$ Choose $k_1 = p_1 - a_1$, $k_2 = p_2 - a_2$, $\ldots$, $k_n = p_n - a_n$

## Converting systems to reachable canonical form

Change of coordinates: $z = Tx$, $T \in \mathbb{R}^{n\times n}$, $T$ invertible

$$\dot{z} = T\dot{x} = T(Ax + Bu) = \underbrace{TAT^{-1}}_{\tilde{A}} z + \underbrace{TB}_{\tilde{B}} u$$

$\tilde{A} = TAT^{-1}$ is called a <u>similarity transformation</u> and gives the dynamics matrix in new coordinates $z = Tx$

Q: Given $A$, can we find $z = Tx$ such that $\tilde{A}$ is in reachable can. form?

A: Let $\hat{A} = TAT^{-1}$, $\hat{B} = TB$

$$\tilde{W}_r = \begin{bmatrix} \tilde{B} & \tilde{A}\tilde{B} & \cdots & \tilde{A}^{n-1}\tilde{B} \end{bmatrix} = \begin{bmatrix} 1 - a_1 & a_1^2 - a_2 & * \\ 0 & 1 & -a_1 & * \\ \vdots & 0 & \ddots & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(complicated)

$\tilde{B} = TB$

$\tilde{A}\tilde{B} = TAT^{-1}TB = TAB$

$\tilde{A}^2 B = \cdots = TA^2 B$

Can compute this knowing $\lambda_A(s) = s^n + a_1 s^{n-1} + \cdots$

$$\Rightarrow \quad \tilde{W}_r = T \underbrace{\begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}}_{W_r}$$

$W_r \leftarrow$ Can compute from $A, B$

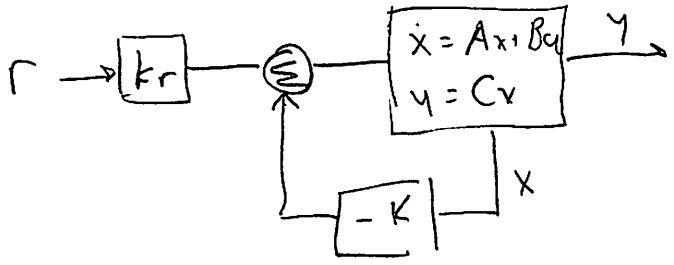
$$\Rightarrow \quad T = \tilde{W}_r W_r^{-1}$$

### Remarks

1. $T$ exists if $(A, B)$ reachable (since $W_r$ full rank)
   ⇒ any reachable system can be converted to reachable canonical form WLOG
2. This proves Thm 6.1, 6.2 and 6.3 (eigenvalue placement)

## Reference tracking

$$\dot{x} = Ax + Bu$$
$$y = Cx$$




Given $r$, how do we design $u = -Kx + k_r r$ so that $y(t) \to r$ as $t \to \infty$

Note: For $r = 0$, controller become $u = -Kx$ and it suffices to choose $K$ such that $\dot{x} = (A - BK)x$ is stable.

Suppose $r \neq 0$. Then

$$\dot{x} = (A - BK)x + k_r r \xrightarrow[\text{equil}]{\text{at}} (A - BK)\tilde{x}_e + Bk_r r = 0 \quad \text{(new eq pt)}$$
$$y = Cx \qquad\qquad y_e = C\tilde{x}_e$$

Solve for $\tilde{y}_e$: $\tilde{y}_e = C\tilde{x}_e = C(-(A-BK)^{-1} B k_r r)$

If we want $\tilde{y}_e = r$ then $-C(A-BK)^{-1} B k_r = 1$

$$\therefore \quad k_r = -\frac{1}{C(A-BK)^{-1}B}$$

## Remarks

1. Is $\tilde{x}_e = -(A-BK)^{-1} B K_r r$ a stable eq pt?

$z = x - \tilde{x}_e \qquad \dot{z} = \dot{x} = Ax + Bu = Ax - BKx + Bk_r r$

$= (A-BK)(z + \tilde{x}_e) + Bk_r r = (A-BK)z + \underbrace{Bk_r r + Bk_r r}_{0}$

$= (A-BK)z$ $\therefore$ stable by construction

## Example: Predator Prey

$$\frac{d}{dt}\begin{bmatrix} H \\ L \end{bmatrix} = \begin{bmatrix} (r+u)H\left(1-\frac{H}{k}\right) - \frac{aHL}{c+H} \\ b\frac{aHL}{c+H} - dL \end{bmatrix} \qquad \begin{aligned} a &= 3.2 \quad b = 0.6 \quad c = 50 \\ d &= 0.56 \quad k = 125 \quad r = 1.6 \end{aligned}$$

$$y = L$$

Goal: Set $y = L_d$, desired # of lynxes

$$\dot{x} = f(x,u) \qquad f(x_e, u_e) = 0 \qquad x_e \approx \begin{bmatrix} 20.6 \\ 29.5 \end{bmatrix} \quad u_e = 0$$

Change coords to eq pt and linearize

$$z = x - x_e \qquad \dot{z} = \begin{bmatrix} 0.13 & -0.93 \\ 0.57 & 0 \end{bmatrix} z + \begin{bmatrix} 17.2 \\ 0 \end{bmatrix} u$$

$$v = u - u_e$$

$$w = y - y_e \qquad w = \begin{bmatrix} 0 & 1 \end{bmatrix} z$$

$$r = y_d - y_e$$

Design feedback controller

$$A - BK = \begin{bmatrix} 0.13 - 17.2k_1 & -0.93 - 17.2k_2 \\ 0.57 & 0 \end{bmatrix}$$

$$\lambda(s) = s^2 + \text{trace}(A-BK)s + \det(A-BK)$$

$$= s^2 + (0.13 - 17.2k_1)s + 0.57(0.93 + 17.2k_2)$$

$$\lambda_d(s) = (s-0.1)(s-0.2) = s^2 - 0.3s + 0.02$$

$$\Rightarrow \quad k_1 = 0.025 \qquad k_2 = -0.052 \qquad \text{"feedback gains"}$$

## Predator Prey, ctd

Design feedforward gain

$$k_r = \frac{1}{C(A-BK)^{-1}B} = 0.002$$

"feedforward gain"
"reference gain"

Full control law

$$u = u_e + v = u_e - Kz + k_r r$$
$$= u_e - K(x - x_e) + k_r(y_d - y_e)$$

## Remarks

1. When $y_d = y_e$, $u = u_e - K(x - x_e) = -K(x - x_e)$ [Mon]
   ($u_e$: 0 for chosen $x_e$)
2. When $y_d \neq 0$, $k_r(y_d - y_e)$ "fights" $-K(x-x_e)$ term
   - $-K(x-x_e)$ tries to bring system back to $y = y_e$
   - $k_r(y_d - y_e)$ pushes back to hold $y = y_d$
   - $k_r$ depends on $K$ (more stabilizing ⇒ more pushing req'd)
3. Control law works <u>near</u> $x_e$, but may not work far away (depending on nonlinearities). In practice, tends to work amazingly well

RMM 22 Oct 2012 (6)

Printed by Richard Murray



```
% L4_2_predprey.m - MATLAB source code for Lecture 4.2
% RMM, 23 Oct 2012
%
% Required files: predprey.m, predprey_rh.m

% Define global functions for use in the feedback
global predprey_K predprey_xd predprey_ud;

% Set paramater values for the simulation
alpha = 12.5;           % multiplier for hare population
beta = 25;              % multiplier for lynx population
gamma = 0.8;            % multiplier for time scale

r = 2 * gamma;                  % birth rate of hares
d = 0.7 * gamma;                % death rate for lynxes
b = 0.3 * beta / alpha;         % birth coefficient for lynxes
k = 10 * alpha;                 % carrying capacity for hares
a = 8 * gamma * alpha / beta;   % interaction coefficient
c = 4 * alpha;                  % saturation coefficient

% Compute the equilibrium point (Note: parameters must be set in predprey.m)
f = inline('predprey(0, x)', 'x');
xeq = fsolve(f, [20, 30])'; He = xeq(1); Le = xeq(2);

% Generate the linearization around the equilibrium point
% App, Bpp computed from symbolic expressions (using Mathematica)
App = [
  -((a*c*k*Le + (c + He)^2*(2*He - k)*r)/((c + He)^2*k)), -((a*He)/(c + He));
  (a*b*c*Le)/(c + He)^2, -d + (a*b*He)/(c + He)
];
Bpp = [He*(1 - He/k); 0];
Cpp = [0 1];

% Assign the desired eigenvalues for the system
Kpp = place(App, Bpp, [-0.1; -0.2]);

% Compute the reference gain
krpp = -1 / (Cpp * ((App - Bpp * Kpp) \ Bpp));

% Now set the parameters for our feedback function
predprey_K = Kpp;
predprey_xd = xeq;

%%
%% Predator prey tracking
%%

figure(3); clf;

% Set initial reference point to equilibrium value
ref = 0; predprey_ud = krpp * ref;
x0 = [20, 30];
T1 = 40;
[t, x] = ode45('predprey', [0 T1], x0, [], -r, d, b, k, a, c);
plot(t, x(:,1), '-', t, x(:,2), '--'); hold on;
plot([0, T1], [ref + Le, ref+Le], '-k');

% Change the reference point and simulate a bit more
ref = +5; predprey_ud = krpp * ref;
```



```
x1 = x(length(t), :); T2 = T1 + 40;
[t, x] = ode45('predprey', [T1 T2], x1, [], -r, d, b, k, a, c);
plot(t, x(:,1), '-', t, x(:,2), '--'); hold on;
plot([T1, T2], [ref + Le, ref+Le], '-k');

% Change the reference point and simulate a bit more
ref = +10; predprey_ud = krpp * ref;
x2 = x(length(t), :); T3 = T2 + 40;
[t, x] = ode45('predprey', [T2 T3], x2, [], -r, d, b, k, a, c);
plot(t, x(:,1), '-', t, x(:,2), '--'); hold on;
plot([T2, T3], [ref + Le, ref+Le], '-k');

% Change the reference point and simulate a bit more
ref = 0; predprey_ud = krpp * ref;
x3 = x(length(t), :); T4 = T3 + 40;
[t, x] = ode45('predprey', [T3 T4], x3, [], -r, d, b, k, a, c);
plot(t, x(:,1), '-', t, x(:,2), '--'); hold on;
plot([T3, T4], [ref + Le, ref+Le], '-k');

% Change the reference point and simulate a bit more
ref = -5; predprey_ud = krpp * ref;
x4 = x(length(t), :); T5 = T4 + 40;
[t, x] = ode45('predprey', [T4 T5], x4, [], -r, d, b, k, a, c);
plot(t, x(:,1), '-', t, x(:,2), '--'); hold on;
plot([T4, T5], [ref + Le, ref+Le], '-k');

axis([0, 200, 15, 45])
xlabel('Time (years)');
ylabel( 'Population', 'Rotation', 90);

lgh = legend('Hare', 'Lynx');
legend(lgh, 'boxoff');
print -dpng 'predprey-track.png'
```



```
% predprey.m - updated predator prey model for CDS 101/110
% RMM, Oct 07
%
% This file contains a simple model for a predator prey system.  The
% model is developed following the treatment in the book by Murray
% (the other one).
%
% The states for these equations are
%
%     x(1)      # of rabbits (prey)
%     x(2)      # of foxes (predator)
%
% The equations allow for a quadratic term in both the predator and
% prey species that cause the population to fall off if you get too
% many of a given species.  These terms are turned off by default.
%

function dxdt=predprey(t, x, flags, r, d, b, k, a, c)

% Extract the states
x1 = x(1);
x2 = x(2);

% Set up the parameter values [these should be passed as arguments...]
if (nargin < 4) r = 1.6; end;
if (nargin < 5) d = 0.56; end;
if (nargin < 6) b = 0.6; end;
if (nargin < 7) k = 125; end;
if (nargin < 8) a = 3.2; end;
if (nargin < 9) c = 50; end;

% Check to see if a feedback law has been defined (when r < 0)
if (r < 0)
  % Assume that the user has defined a function that we can call
  % Add this value to the nominal value of the parameter (renegated)
  r = -r + predprey_rh(t, x, -r);
end

dx1 = r * x1 * (1 - x1/k) - a * x2 * x1/(c + x1);
dx2 = b *a *x2 * x1 / (c + x1) - d * x2;

dxdt = [dx1; dx2];
```



```
% predprey_rh.m - feedback function for predator prey system
% RMM, 13 Sep 06
%
% This function computes a simple linear control law, with saturation, for
% the predator prey system

function rh=predprey_rh(t, x, rnom)
global predprey_K predprey_xd predprey_ud;

% Compute the feedback control law
rh = -predprey_K * ([x(1); x(2)] - predprey_xd) + predprey_ud;

% Saturate the control input between 0 and 4*rnom
if (rh < 0) rh = 0; end;
if (rh > 4*rnom) rh = 4*rnom; end;
```

# Integral Feedback

Issue: controllers we have design require accurate models to obtain perfect tracking. Eg,

$$k_r = -\frac{1}{C(A-BK)^{-1}B}$$

Alternative approach: use integral feedback to compensate for (model) uncertainty

Key idea: define new state $z = \int(y-r)$ and require $z \to 0 \Rightarrow y \to r$

$$\frac{d}{dt}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ y - r \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ Cx - r \end{bmatrix} = \tilde{A}x + \tilde{B}_1 u + \tilde{B}_2 r$$

Choose feedback law

$$u = -Kx - k_i z + k_r r$$

System equilibrium point:

$$x_e = -(A-BK)^{-1}B(k_r r - k_i z_e)$$

$z_e$ unspecified (but will force $\dot{z} = 0 \Rightarrow y = r$)

## Remarks

1. Suppose we set $k_r$ incorrectly (even $k_r = 0$). Still get stability of system to $x=0$, $z=0 \Rightarrow y \to r$
2. Will see integral feedback again when we study PID