Model-Based Controller Design

Optimal linear quadratic (LQ) control

Given the LTI plant model, with directly measurable state x:

$$\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases}; \qquad x \in R^n, u \in R^m, y \in R^l$$

Find the control u to minimize the performance index (cost):

$$J = \frac{1}{2}x^T(t_f)Mx(t_f) + \frac{1}{2}\int_{t_0}^{t_f}[x^T(t)Qx(t) + u^T(t)Ru(t)]dt$$

where $R = R^T > 0,\ Q = Q^T \geq 0,\ M = M^T \geq 0$ are weight matrices

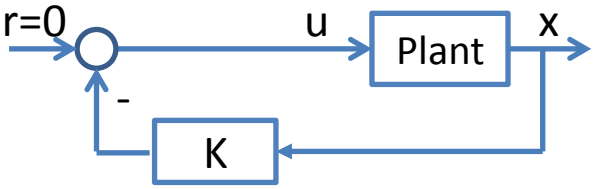
- The optimal LQR control is found by minimizing the Hamiltonian function:

$$H = \frac{1}{2}[x^T(t)Qx(t) + u^T(t)Ru(t)] + \lambda^T(t)[Ax(t) + Bu(t)]$$

where $\lambda(t)$ is the Lagrangian multiplier

So, let $\begin{cases}\frac{\partial H}{\partial u} = 0\\ \frac{\partial H}{\partial x} = \dot{\lambda}(t)\\ \frac{\partial H}{\partial \lambda} = \dot{x}(t)\end{cases} \Rightarrow \begin{cases}Ru(t) + B^T\lambda(t) = 0\\ Qx(t) + A^T\lambda(t) = \dot{\lambda}(t)\\ Ax(t) + Bu(t) = \dot{x}(t)\end{cases} \Rightarrow \begin{cases}u(t) = -R^{-1}B^T\lambda(t)\\ \dot{\lambda}(t) = A^T\lambda(t) + Qx(t)\\ \dot{x}(t) = Ax(t) + Bu(t)\end{cases}$

- For LTI systems $\lambda(t) = P(t)x(t)$



$\Rightarrow$ Optimal LQ control: $u(t) = -K(t)x(t)$

where $K(t) = R^{-1}B^TP(t)$

and $P(t)$ is the solution of the differential Riccati equation (DRE):

$$\dot{P}(t) = A^TP(t) + P(t)A + Q - P(t)BR^{-1}B^TP(t)\ , \qquad P(t_f) = M$$

Optimal Linear Quadratic (LQ) Control

Linear quadratic regulator (LQR)

Given the plant model:

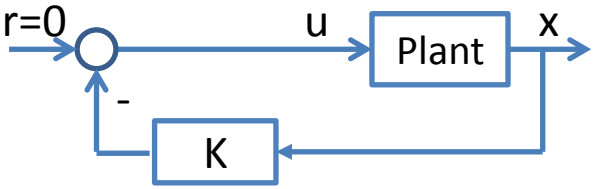
$$\begin{cases}\dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t)\end{cases}; \qquad x \in R^n, u \in R^m, y \in R^l$$

Minimize the performance index (cost), for $t_f = \infty$ (steady-state case):

$$J = \frac{1}{2}\int_{t_0}^{\infty}[x^T(t)Qx(t) + u^T(t)Ru(t)]dt$$

where $R = R^T > 0,\ Q = Q^T \geq 0$ are weight matrices

- For $t_f = \infty$, $P(t) = P$ (constant)



$\Rightarrow$ LQR control: $u(t) = -Kx(t)$

where $K = R^{-1}B^TP$

and P is the solution of the algebraic Riccati equation (ARE):

$$A^TP(t) + P(t)A + Q - P(t)BR^{-1}B^TP(t) = 0$$

Closed-loop system equation ($r = 0$):

$$\begin{cases}\dot{x}(t) = (A - BK)x(t) + Br \\ y(t) = Cx(t) + Dr\end{cases}$$

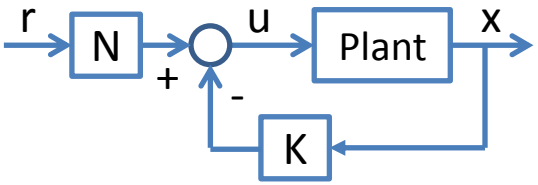
Or, equivalently, the quadruple $[(A - BK), B, C, D]$

Optimal Linear Quadratic (LQ) Control

Linear quadratic regulator (LQR)

Given the plant model:

$$\begin{cases}\dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t)\end{cases}; \quad x \in R^n, u \in R^m, y \in R^l$$



Let $u(t) = -Kx(t) + Nr$

$\Rightarrow$ LQR state feedback control gain: $K = R^{-1}B^T P$

and P is the solution of the algebraic Riccati equation (ARE):

$A^T P(t) + P(t)A + Q - P(t)BR^{-1}B^T P(t) = 0$

Choose N so that the closed-loop DC-gain is equal to one.

Closed-loop system equation:

$$\begin{cases}\dot{x}(t) = (A - BK)x(t) + BNr \\ y(t) = Cx(t) + DNr\end{cases}$$

Or, equivalently, $G(s) = \left[C\left(sI - (A - BK)\right)^{-1}B + D\right]N$

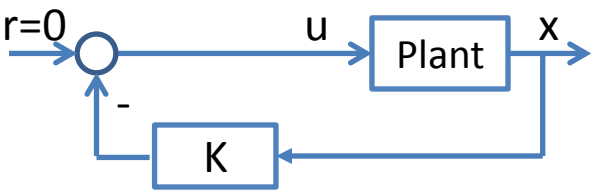
Or, the quadruple $[(A - BK), BN, C, DN]$

dc-Gain: $G(0) = -[C(A - BK)^{-1}B + D]N = 1$

$\Rightarrow$ Feedforward control gain: $N = -[C(A - BK)^{-1}B + D]^{-1}$

Linear quadratic regulator (LQR)

Matlab functions: are(), lqr(), care(), dcgain(),



- For example: P= are(A,B*inv(R)*B',Q);
  [K,P]=lqr(A,B,Q,R);
  [P,L,K]=care(A,B,Q,R,S,E);

Example: Let $\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases}$, with $A = \begin{bmatrix} -0.2 & 0.5 & 0 & 0 & 0 \\ 0 & -0.5 & 1.6 & 0 & 0 \\ 0 & 0 & -14.3 & 85.8 & 0 \\ 0 & 0 & 0 & -33.3 & 100 \\ 0 & 0 & 0 & 0 & -10 \end{bmatrix}$,

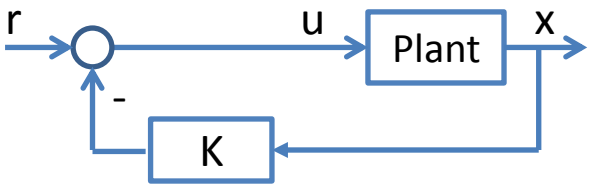
$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 30 \end{bmatrix}$, $C = [1 \quad 0 \quad 0 \quad 0 \quad 0]$, $D = 0$.

- Find the LQR control $u(t) = -Kx(t)$ and plot the closed-loop step response for $R = 1$, $Q = diag\{\rho, 0,0,0,0\}$, and $\rho = 1,5,10,50,100$.
- Also, for ρ=100, plot the closed-loop states

Linear quadratic regulator (LQR)

Matlab code:

```
A=[-0.2,0.5,0,0,0; 0,-0.5,1.6,0,0;
0,0,-14.3,85.8,0; 0,0,0,-33.3,100;
0,0,0,0,-10];  B=[0;0;0;0;30];
C=[1,0,0,0,0]; D=0;  rho=1;
R=1; Q=diag([rho,0,0,0,0]);
for rho=[1,5,10,50,100],
Q(1,1)=rho; [K,P] =lqr(A,B,Q,R),
step(ss(A-B*K,B,C,D)); hold on;
end, hold off;
```



Optimal solution:

P =

0.3563	0.0326	0.0026	0.0056	0.0309
0.0326	0.0044	0.0004	0.0009	0.0056
0.0026	0.0004	0.0000	0.0001	0.0005
0.0056	0.0009	0.0001	0.0002	0.0012
0.0309	0.0056	0.0005	0.0012	0.0088

K =

0.9260	0.1678	0.0157	0.0371	0.2653

$$\boldsymbol{u(t) = -Kx(t) + r}$$

- Or, instead of lqr(), use:

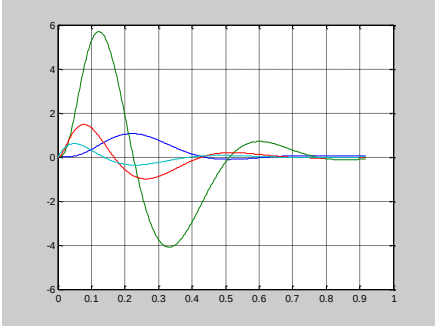
```
P=are(A,B*inv(R)*B',Q),
K=inv(R)*B'*P,
```

- Closed-loop states

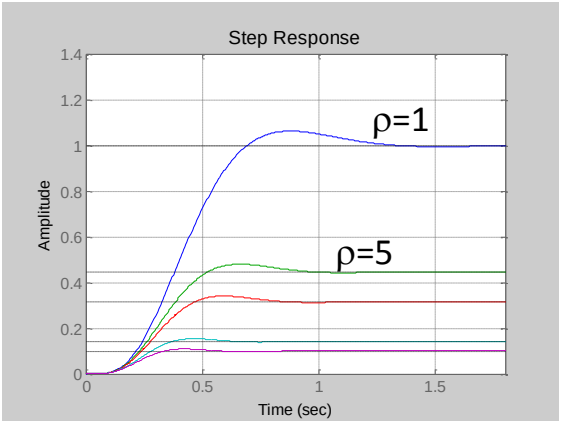
```
Q(1,1)=100; [K,P] =lqr(A,B,Q,R); Ac=A-B*K;
[y,t,x]=step(ss(Ac,B,C,D)); plot(t,x(:,2:5)); grid;
```

⇒

States x2, …, x5



Closed-loop Step Response



LQR for Discrete-Time Systems

Discrete-time LQ control

Given the plant model:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(t) = Cx(t) + Du(t) \end{cases}; \qquad x \in R^n, u \in R^m, y \in R^l$$

Minimize the performance index (cost):

$$J = \frac{1}{2}\sum_{k=0}^{N}[x^T(k)Qx(k) + u^T(k)Ru(k)]$$

where $R = R^T > 0,\ Q = Q^T \geq 0$ are weight matrices

$\Rightarrow$ Discrete LQ control: $u(k) = -Kx(k)$

where $K = R^{-1}B^T P(k)$

and $P(k)$ is the solution of the dynamic/difference Riccati equation (DRE):

$$P(k) = A^T[P(k+1) - P(k+1)BR^{-1}B^T P(k+1)]A + Q \ , \quad P(N) = Q$$

For steady-state case ($N = \infty$)

- P is a constant matrix

$\Rightarrow$ Discrete LQR

$\Rightarrow \quad u(k) = -Kx(k)$

where $K = (R + B^T PB)^{-1}B^T PA$

and $P = A^T[P - PBR^{-1}B^T P]A + Q$

Matlab functions:

dlqr(), dare(),

Selection of Weighting Matrices

Single-input, single-output (SISO) case:

- LQR control (a possible choice for weight matrices)
  - $R = 1, Q = diag(x_{1max}, \dots, x_{nmax})$, where x_{imax} is the maximum expected value of state $x_i(t)$
- Cheap control (control effort is inexpensive)
  - $R = 1, Q = \rho Q_0$, where $Q_0 = {Q_0}^T \geq 0$ and $\rho > 0$ is a large number
- Expensive control (control effort is expensive)
  - $R = 1, Q = \rho Q_0$, where $Q_0 = {Q_0}^T \geq 0$ and $\rho > 0$ is a small number
- Terminal control (terminal state must tend to zero)
  - $Q = 0, M = \rho M_0$, where $M_0 = {M_0}^T \geq 0$ and ρ tends to infinity
  - Here, the DRE changes to:

$$-\dot{P}(t) = A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) \ , \qquad P(t_f) = \rho M_0$$

Equivalently, let $\Pi(t) = P^{-1}(t)$

$$\Rightarrow \quad \dot{\Pi}(t) = \Pi(t)A^T + A\Pi(t) - BR^{-1}B^T \ , \qquad \Pi(t_f) = 0$$

LQR with Degree of Stability α

Plant model: $\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$

Degree of stability α

- All closed-loop poles are to the left of $-\alpha$
- Define a new performance index (cost)

$$J = \frac{1}{2}\int_{t_0}^{\infty} e^{2\alpha t}[x^T(t)Qx(t) + u^T(t)Ru(t)]dt$$

where $R = R^T > 0,\ Q = Q^T \geq 0$ are weight matrices

- Define a new state variable and control $\begin{cases} \xi(t) = e^{\alpha t}x(t) \\ v(t) = e^{\alpha t}u(t) \end{cases}$

New state equation and performance index (cost)

$$\dot{\xi}(t) = (A + \alpha I)\xi(t) + Bv(t)$$

$$J = \frac{1}{2}\int_{t_0}^{\infty}[\xi^T(t)Q\xi(t) + v^T(t)Rv(t)]dt$$

LQR with degree of stability α

$$u(t) = -Kx(t)$$

where $K = R^{-1}B^TP$

and the modified ARE is: $(A + \alpha I)^TP + P(A + \alpha I) + Q - PBR^{-1}B'P = 0$

Matlab code:

```
P=are((A_a*eye(size(A)))',B*inv(R)*B',Q);
k=inv(R)*B'*P;
u=-k*x;
```

or use lqr() with A replaced with (A+αI)

Observer Design

LQR assumptions:

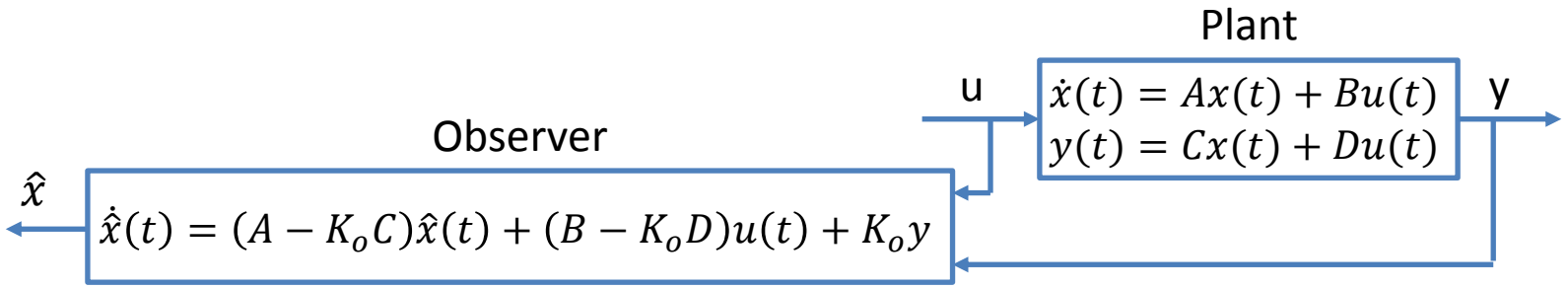
1. Plant model is perfectly known: $\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases}$
2. All states are directly measurable
   - If assumption 1 is not true (the model has uncertainties), use robust control
   - If assumption 2 is not true (only the output is measurable), use an observer
     - System must be observable

Observer (state estimator) design:

- Create a copy of the plant model with corrective term

$$\begin{cases}\dot{\hat{x}} = A\hat{x} + Bu + K_o(y - \hat{y})\\ \hat{y} = C\hat{x} + Du\end{cases}$$

- Choose the observer gain K_o so that all eigenvalues of $(A - K_oC)$ are in LHP



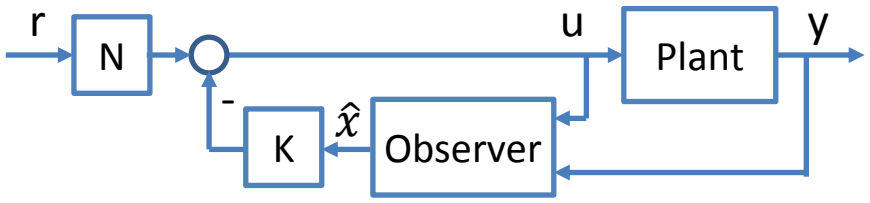
Observer-Based Controller Design

For plant model, (assuming that the state vector x is not measured):

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

<u>An observer-based controller has the form:</u>

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + K_o(y - C\hat{x} - Du) \\ u = -K\hat{x} + Nr \end{cases}$$



- The observer can be written as:

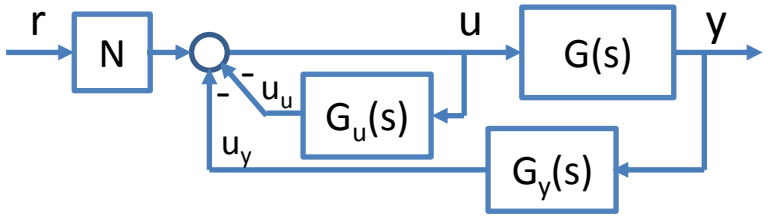
$$\dot{\hat{x}} = (A - K_oC)\hat{x} + (B - K_oD)u + K_oy$$

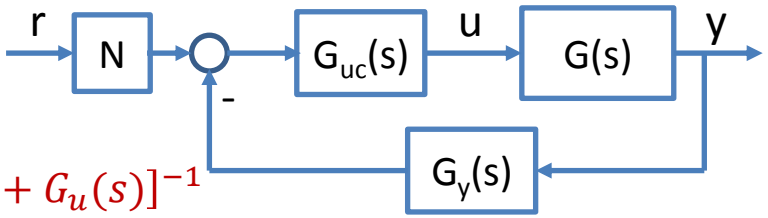
- Or, by superposition:

$$\begin{cases} \dot{\hat{x}}_u = (A - K_oC)\hat{x}_u + (B - K_oD)u \\ \dot{\hat{x}}_y = (A - K_oC)\hat{x}_y + K_oy \\ \hat{x} = \hat{x}_u + \hat{x}_y \end{cases}$$

<u>Observer-based controller, ($\hat{x} = \hat{x}_u + \hat{x}_y$):</u>

$$\Rightarrow \begin{cases} \left.\begin{aligned} \dot{\hat{x}}_u &= (A - K_oC)\hat{x}_u + (B - K_oD)u \\ u_u &= K\hat{x}_u \end{aligned}\right\} \Rightarrow G_u(s) \\ \left.\begin{aligned} \dot{\hat{x}}_y &= (A - K_oC)\hat{x}_y + K_oy \\ u_y &= K\hat{x}_y \end{aligned}\right\} \Rightarrow G_y(s) \\ u = Nr - u_u - u_y \end{cases}$$





$$G_{uc}(s) = [I + G_u(s)]^{-1}$$

Observer-Based Controller Design

Matlab function: reg(),

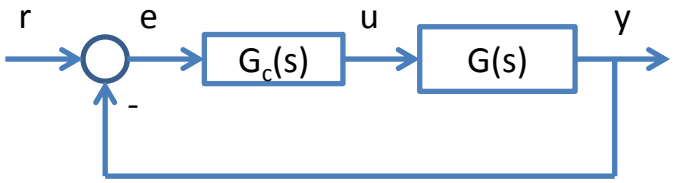
Syntax: [Ac,Bc,Cc,Dc]=reg(A,B,C,D,K,Ko);

Or: Gc=-reg(G,K,Ko)

Example: Let $\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases}$, with $A = \begin{bmatrix}-0.2 & 0.5 & 0 & 0 & 0\\ 0 & -0.5 & 1.6 & 0 & 0\\ 0 & 0 & -14.3 & 85.8 & 0\\ 0 & 0 & 0 & -33.3 & 100\\ 0 & 0 & 0 & 0 & -10\end{bmatrix}, B = \begin{bmatrix}0\\0\\0\\0\\30\end{bmatrix}$,

$C = [1 \quad 0 \quad 0 \quad 0 \quad 0]$, $D = 0$.

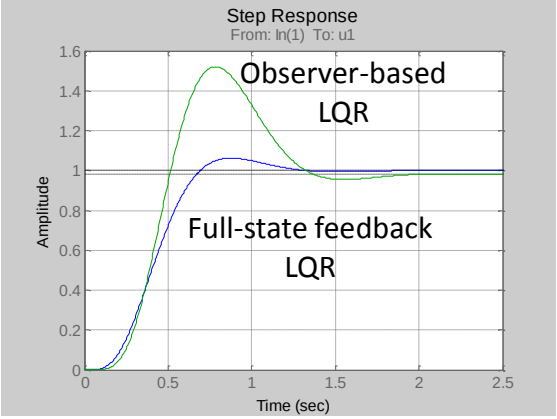
Find the observer-based LQR control $u(t) = -K\hat{x}(t) + r$ for $R = 1$, $Q = diag\{1,0,0,0,0\}$, using an observer gain $K_o = [-8.3 \quad 979.24 \quad -19367.61 \quad 4293.85 \quad 0]^T$, and then plot the closed-loop step response.



Matlab code:

```
A=[-0.2,0.5,0,0,0; 0,-0.5,1.6,0,0;
0,0,-14.3,85.8,0; 0,0,0,-33.3,100;
0,0,0,0,-10];  B=[0;0;0;0;30];
C=[1,0,0,0,0]; D=0; R=1; Q=diag([1,0,0,0,0]);
G=ss(A,B,C,D); [K,P] =lqr(A,B,Q,R);
Ko=[-8.3,979.24,-19367.61,4293.85,0]';
Gc=-reg(G,K,Ko); zpk(Gc),
Gcl1=ss(A-B*K,B,C,D); Gcl2=feedback(Gc*G,1,-1);
step(Gcl1,Gcl2);
```

$$\Rightarrow G_c(s) = \frac{11.4839\ (s+33.34)\ (s+14.3)\ (s+10)\ (s+1.792)}{(s+20.92)\ (s^2 + 30.19s + 328.1)\ (s^2 + 6.845s + 120)}$$



Pole Placement Design Technique

Given the system $\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$ with a state feedback control $u = -Kx + Nr$

- If (A, B) is controllable and x is directly measurable, the closed-loop poles (eigenvalues of $(A - BK)$) can be arbitrarily assigned
  - Imaginary poles, however, must come in conjugate pairs
- The system's open-loop characteristic equation is:

  $$a(s) = det(sI - A) = s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n$$
- The system's closed-loop characteristic equation is:

  $$\delta(s) = det(sI - A + BK) = s^n + d_1(K) s^{n-1} + \cdots + d_{n-1}(K) s + d_n(K)$$
- Let the desired closed-loop poles be: $\mu_i, i = 1, \ldots, n$
  - The desired closed-loop characteristic equation can be written as :

    $$\alpha(s) = \prod_{i=1}^{n} (s - \mu_i) = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1} s + \alpha_n$$

Feedback gain $K = [k_1 \quad \cdots \quad k_n]$ can be found, by equating $\delta(s)$ and $\alpha(s)$, as

$$\delta(s) = \alpha(s) \Rightarrow \begin{cases} d_1(K) = \alpha_1 \\ \vdots \\ d_{n-1}(K) = \alpha_{n-1} \\ d_n(K) = \alpha_n \end{cases} \Rightarrow K = [k_1 \quad \cdots \quad k_n] = ?$$

Pole Placement Control Algorithms

- Bass-Gura algorithm

$$K = [a - \alpha]^T L^{-1} \mathcal{C}^{-1}$$

where $[a-\alpha]^T = [a_1 - \alpha_1 \quad \cdots \quad a_n - \alpha_n]$, $\mathcal{C} = [B \quad AB \quad \cdots \quad A^{n-1}B]$, and $L = \begin{bmatrix} a_{n-1} & a_{n-2} & \cdots & a_1 & 1 \\ a_{n-2} & a_{n-3} & \cdots & 1 & \\ \vdots & \vdots & \ddots & & \\ a_1 & 1 & & & \\ 1 & & & & \end{bmatrix}$

is the non-singular Hankel matrix

- Ackermann's algorithm
  - Can be applied only to SISO systems

$$K = [0 \quad \cdots \quad 0 \quad 1]\mathcal{C}^{-1}\alpha$$

where $\alpha^T = [\alpha_1 \quad \cdots \quad \alpha_n]$, $\mathcal{C} = [B \quad AB \quad \cdots \quad A^{n-1}B]$

- Numerically robust pole-placement algorithm (place)
  - Can be applied to MIMO systems
  - Desired poles must all be distinct

Matlab functions: bass_pp(), acker(), place(),

K=bass_pp(A,B,p_{des})

K=acker(A,B,p_{des})

K=place(A,B,p_{des})

where p_{des} = desired closed-loop poles

Matlab function for Bass-Gura algorithm

```
function K=bass_pp(A,B,p)
n=length(B); a1=poly(p);
alpha=[a1(n:-1:2),1]; C=ctrb(A,B);
a=poly(A); aa=[a(n:-1:2),1]; L=hankel(aa);
K=(a1(n+1:-1:2)-a(n+1:-1:2))*inv(L)*inv(C);
```

Pole Placement Control Design

Example: Consider the multivariable system model $\begin{cases}\dot{x} = Ax + Bu \\ y = Cx + Du\end{cases}$ with

$$A = \begin{bmatrix}0 & 2 & 0 & 0 & -2 & 0\\ 1 & 0 & 0 & 0 & 0 & -1\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 3 & 0 & 0\\ 2 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 0 & 1 & 0\end{bmatrix}, B = \begin{bmatrix}1 & 2\\ 0 & 0\\ 0 & 1\\ 0 & -1\\ 0 & 1\\ 0 & 0\end{bmatrix}, C = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0], D = 0$$

- Design a state feedback control $u = -Kx + Nr$ so the closed-loop poles are at: $\mu_i = -1, -2, -3, -4, -1 \pm j$. Also draw the closed-loop step response.

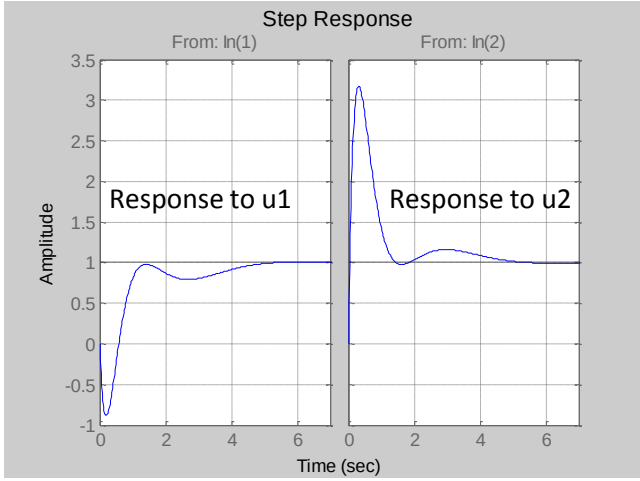
Matlab code:

```
A=[0,2,0,0,-2,0; 1,0,0,0,0,-1; 0,1,0,0,0,0; 0,0,0,3,0,0; 2,0,0,1,0,0; 0,0,-1,0,1,0];
B=[1,2; 0,0; 0,1; 0,-1; 0,1; 0,0]; C=[1,0,0,0,0,0]; D=0;
pdes=[-1,-2,-3,-4,-1+i,-1-i];
K=place(A,B,pdes), N=inv(diag(dcgain(ss((A-B*K),B,C,D)))),
eig(A-B*K), sys_cl=ss((A-B*K),B*N,C,D); dcgain(sys_cl);
step(sys_cl);
```

Control gains:

$$K = \begin{bmatrix}8.0037 & -26.0056 & -26.1122 & 16.3759 & 23.2324 & 28.7704\\ 0.1759 & -3.0018 & -3.1317 & -7.7366 & 2.0396 & 2.6613\end{bmatrix}$$

$$N = \begin{bmatrix}-10.9884 & 0\\ 0 & 13.6388\end{bmatrix}$$



Pole Placement Control Design Technique

- Observer design using pole placement technique
  - Dual of control design problem
    - Matlab functions: place(), acker(),

Example: Given the plant $\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$ with $A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 11 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix},$

$C = [1 \quad 2 \quad 3 \quad 4], D = 0$

- Design an observer with poles at: $s_{1,2,3,4} = -1, -2, -1 \pm i$

Matlab code:

```
A=[0,1,0,0; 0,0,-1,0; 0,0,0,1; 0,0,11,0];
B=[0; 1; 0; -1]; C=[1,2,3,4]; D=0;
pdes=[-1, -2, -1+i, -1-i];
Ko=place(A',C',pdes)', eig(A-Ko*C),
```

Observer gain:

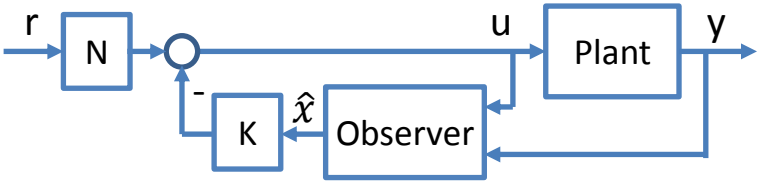
$$K_o = \begin{bmatrix} -0.2203 \\ -0.4750 \\ 0.4238 \\ 1.2247 \end{bmatrix}$$

Pole Placement Control Design Technique

Observer-based controller design using pole placement

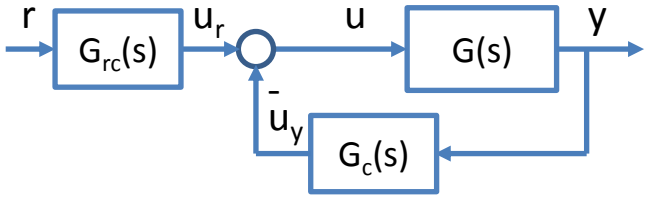
Plant: $\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$

Controller: $\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + K_o(y - C\hat{x} - Du) \\ u = -K\hat{x} + Nr \end{cases}$



Equivalently:

- $G_c(s) = \frac{U_y(s)}{Y(s)} = \begin{bmatrix} A - BK - K_oC + K_oDK & K_o \\ K & 0 \end{bmatrix}$
- $G_{rc}(s) = \frac{U_r(s)}{R(s)} = \begin{bmatrix} A - BK - K_oC + K_oDK & (B - K_oD)N \\ -K & N \end{bmatrix}$



Closed-loop system:

$$\begin{cases} \begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A & -BK \\ K_oC & A - BK - K_oC \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} Nr \\ y = [C \quad -DK] \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + [D]Nr \end{cases}$$

$$\Rightarrow \quad G_{cl}(s) = \left([C \quad -DK] \left[sI - \begin{pmatrix} A & -BK \\ K_oC & A - BK - K_oC \end{pmatrix} \right]^{-1} \begin{bmatrix} B \\ B \end{bmatrix} + [D] \right) N$$

$$\text{DC-gain} = G_{cl}(0) = 1 \quad \Rightarrow \quad N = -\left([C \quad -DK] \begin{bmatrix} A & -BK \\ K_oC & A - BK - K_oC \end{bmatrix}^{-1} \begin{bmatrix} B \\ B \end{bmatrix} + [D] \right)^{-1}$$

Pole Placement Control Design Technique

Observer-based controller design using pole placement

Example: Given the plant $\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$ with $A = \begin{bmatrix} -0.3 & 0.1 & -0.05 \\ 1 & 0.1 & 0 \\ -1.5 & -8.9 & -0.05 \end{bmatrix}$, $B = \begin{bmatrix} 2 \\ 0 \\ 4 \end{bmatrix}$,

$C = [1 \quad 2 \quad 3]$, $D = 0$

- Design an observer-based controller $u = -Kx + Nr$ so the closed-loop poles are at: $p_{1,2,3} = -1, -2, -3$ and the observer poles are at $s_{1,2,3} = -10, -20, -30$

Matlab code:

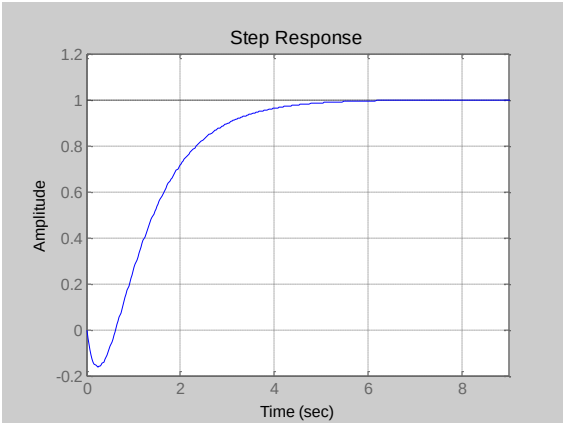
```
A=[-0.3,0.1,-0.05; 1,0.1,0; -1.5,-8.9,-0.05];
B=[2; 0; 4]; C=[1,2,3]; D=0;
pcdes=[-1, -2, -3]; podes=[-10, -20, -30];
K=place(A,B,pcdes), Ko=place(A',C',podes)',
N=inv(dcgain(ss([A,-B*K;Ko*C,A-B*K-Ko*C],[B;B],[C,-D*K],D))),
Gcl=ss([A,-B*K;Ko*C,A-B*K-Ko*C],[B;B]*N,[C,-D*K],D*N);
Step(Gcl);
```

Controller gains:

$K = [3.6496 \quad 5.7189 \quad -0.3873]$,

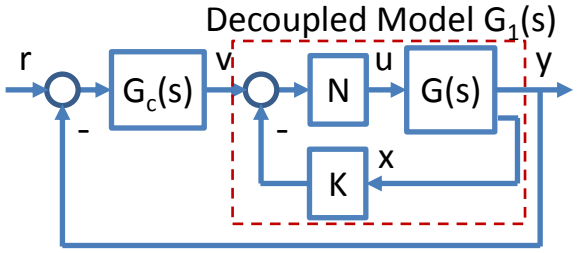
$K_o = \begin{bmatrix} -225.8203 \\ -17.9695 \\ 107.1698 \end{bmatrix}$,

$N = -0.1106$



Decoupling Control of Multivariable Systems

Decoupling control with state feedback



- Plant (A,B,C,D) with m inputs and m outputs
- State feedback control: $u = Nv - Kx$

For each row ${c_j}^T$ of C , let $d_j \in \{0,1,\cdots,n-1\}$ be the smallest integer where ${c_j}^T A^{d_j} B \neq 0$

Decoupler gains: $N = {B_1}^{-1}$, $K = N \begin{bmatrix} {c_1}^T A^{d_1+1} \\ \vdots \\ {c_m}^T A^{d_m+1} \end{bmatrix}$, (if $B_1 = \begin{bmatrix} {c_1}^T A^{d_1} B \\ \vdots \\ {c_m}^T A^{d_m} B \end{bmatrix}$ is nonsingular)

Decoupled model:

$$G_1(s) = diag\left(\frac{1}{s^{d_1+1}}, \cdots, \frac{1}{s^{d_m+1}}\right)$$

Now, design a controller G_{cj} for each G_{1j}:

$$G_c(s) = diag(G_{c1}(s), \cdots, G_{cm}(s))$$

Matlab function: decoupler(),

Syntax: [G1,K,d,N]=decoupler(G)

where G and G1 are the plant and decoupled models,
K and N are the feedback and feedforward gains,
d is the vector of dj's

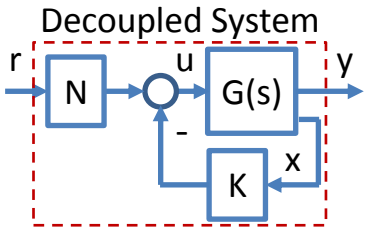
Matlab code for decoupler function:

```
function [G1,K,d,Gam]=decoupler(G)
A=G.a; B=G.b; C=G.c; [n,m]=size(G.b); B1=[]; K0=[];
for j=1:m,
    for k=0:n-1,
        if norm(C(j,:)*A^k*B)>eps, d(j)=k; break; end,
    end,
    B1=[B1; C(j,:)*A^d(j)*B]; K0=[K0; C(j,:)*A^(d(j)+1)];
end,
Gam=inv(B1); K=Gam*K0;
G1=minreal(tf(ss(A-B*K,B,C,G.d))*inv(B1));
```

Decoupling Control of Multivariable Systems

Pole placement decoupling control with state feedback

- State feedback control: $u = Nr - Kx$



Define: $E = \begin{bmatrix} {c_1}^T A^{d_1} B \\ \vdots \\ {c_m}^T A^{d_m} B \end{bmatrix}$ and $F = \begin{bmatrix} {c_1}^T \left(A^{d_1+1} + a_{1,1}A^{d_1} + \cdots + a_{1,d_1+1} I\right) \\ \vdots \\ {c_m}^T \left(A^{d_m+1} + a_{m,1}A^{d_m} + \cdots + a_{m,d_m+1} I\right) \end{bmatrix}$

Pole placement: Assign control gains so that the transfer function of the jth subsystem is equal to the n_j-th-order optimal ITAE standard TF: $T_{n_j}(s) = \frac{1}{s^{n_j} + a_1 s^{n_j-1} + \cdots + a_{n_j-1} s + a_{n_j}}$, $n_j = d_{j+1}$, with parameters:

n	M_p	$\omega_n t_s$	Denominator of $T_n(s)$ with $a_n = \omega_n^n$
1			$s + \omega_n$
2	4.6%	6.0	$s^2 + 1.41\omega_n s + \omega_n^2$
3	2%	7.6	$s^3 + 1.75\omega_n s^2 + 2.15\omega_n^2 s + \omega_n^3$
4	1.9%	5.4	$s^4 + 2.1\omega_n s^3 + 2.4\omega_n^2 s^2 + 2.7\omega_n^3 s + \omega_n^4$
5	2.1%	6.6	$s^5 + 2.8\omega_n s^4 + 5.0\omega_n^2 s^3 + 5.5\omega_n^3 s^2 + 2.4\omega_n^4 s + \omega_n^5$
6	5%	7.8	$s^6 + 2.25\omega_n s^5 + 6.6\omega_n^2 s^4 + 8.6\omega_n^3 s^3 + 7.45\omega_n^4 s^2 + 2.95\omega_n^5 s + \omega_n^6$

Matlab code for std_tf function:

```
function G=std_tf(wn,n)
if n>6, disp('n must be less than 7'); end
M=[1,1,0,0,0,0,0;    1,1.4,1,0,0,0,0;
   1,1.75,2.15,1,0,0,0;    1,2.1,3.4,2.7,1,0,0;
   1,2.8,5,5.5,3.4,1,0;    1,3.25,6.6,8.6,7.45,3.95,1];
G=tf(wn^n,M(n,1:n+1).*(wn*ones(1,n+1)).^[0:n]);
```

Decoupled transfer function: $G_1(s) = diag\left(\frac{a_{1,d_1+1}}{s^{d_1+1} + a_{1,1}s^{d_1} + \cdots + a_{1,d_1+1}}, \cdots, \frac{a_{m,d_m+1}}{s^{d_m+1} + a_{m,1}s^{d_m} + \cdots + a_{m,d_m+1}}\right)$

Control gains: $K = E^{-1}F$, $N = E^{-1}M$, with $M = diag(a_{1,d_1+1}, \cdots, +a_{m,d_m+1})$, (if E is nonsingular)

Pole Placement Control Design Technique

Pole placement decoupling control with state feedback

Matlab functions: std_tf(), decouple_pp(),

Syntax: G=std_tf(wn,n);

[G1,K,d,N]=decouple_pp(G,wn)

Example: For plant $\begin{cases}\dot{x} = Ax + Bu \\ y = Cx + Du\end{cases}$ with

$A = \begin{bmatrix}2.25 & -5 & -1.25 & -0.5 \\ 2.25 & -4.25 & -1.25 & -0.25 \\ 0.25 & -0.5 & -1.25 & -1 \\ 1.25 & -1.75 & -0.25 & -0.75\end{bmatrix}$, $B = \begin{bmatrix}4 & 6 \\ 2 & 4 \\ 2 & 2 \\ 0 & 2\end{bmatrix}$, $C = \begin{bmatrix}0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2\end{bmatrix}$,

$D = \begin{bmatrix}0 & 0 \\ 0 & 0\end{bmatrix}$

- Design an observer-based controller $u = -Kx + Nr$ so the closed-loop poles are defined by the optimal ITAE standard transfer function with $\omega_n = 5$ rad/sec

Matlab code:

```
A=[2.25,-5,-1.25,-0.5; 2.25,-4.25,-1.25,-0.25;
0.25,-0.5,-1.25,-1; 1.25,-1.75,-0.25,-0.75];
B=[4,6; 2,4; 2,2; 0,2]; C=[0,0,0,1; 0,2,0,2]; D=[0,0; 0,0];
G=ss(A,B,C,D); [G1,K,d,N]=decouple_pp(G,5),
step(G1);
```

Matlab code for decouple_pp function:

```
function [G1,K,d,N]=decouple_pp(G,wn)
A=G.a; B=G.b; C=G.c; [n,m]=size(G.b); E=[]; F=[];
for j=1:m,
   for i=0:n-1,
      if norm(C(j,:)*A^i*B)>eps, d(j)=i; break; end,
   end,
   g1=std_tf(wn,d(j)+1); [~,cc]=tfdata(g1,'v');
   F=[F; C(j,:)*polyvalm(cc,A)]; E=[E; C(j,:)*A^d(j)*B];
end,
Gam=inv(E); K=Gam*F; G0=tf(ss(A-B*K,B,C,G.d));
N=Gam*inv(dcgain(G0*Gam)); G1=minreal(G0*N);
```

Controller gains:

$K = \begin{bmatrix}-0.125 & 2.125 & -0.375 & -4.375 \\ 0.625 & -0.875 & -0.125 & 2.125\end{bmatrix}$,

$N = \begin{bmatrix}-7.5 & 1.25 \\ 2.5 & 0\end{bmatrix}$

Decoupled TF:

$G_1(s) = \begin{bmatrix}\frac{5}{s+5} & 0 \\ 0 & \frac{5}{s+5}\end{bmatrix}$