State Space Models

We begin by writing the state space model for a an *individual (Ai)* axis

```
syms I I_dot theta_ddot_g V_s R_m L_m V_emf theta_dot_m tau_m G theta_dot_g b J K_e K_i

u_i = V_s
```

u_i = V_s

```
x_i = [
    I
    theta_dot_g
]
```

x_i =

$$\begin{pmatrix} \mathrm{I} \\ \dot{\theta}_g \end{pmatrix}$$

```
x_dot_i = [
    I_dot
    theta_ddot_g
]
```

x_dot_i =

$$\begin{pmatrix} \dot{I} \\ \ddot{\theta}_g \end{pmatrix}$$

```
A_i = [
    -R_m/L_m    -K_e/(L_m*G);
    K_i/(J*G)   -b/J
]
```

A_i =

$$\begin{pmatrix} -\frac{R_m}{L_m} & -\frac{K_e}{G\,L_m} \\ \frac{K_i}{G\,J} & -\frac{b}{J} \end{pmatrix}$$

```
B_i = [
    1/L_m;
    0
]
```

B_i =

$$\begin{pmatrix} \frac{1}{L_m} \\ 0 \end{pmatrix}$$

State Space Models for Individual Axis

```
syms I_pan  I_dot_pan  theta_ddot_g_pan  V_s_pan R_m_pan
L_m_pan V_emf_pan theta_dot_m_pan tau_m_pan G_pan theta_dot_g_pan b_pan J_pan K_e_pan K_i_pan
syms I_tilt I_dot_tilt theta_ddot_g_tilt V_s_tilt R_m_tilt
L_m_tilt V_emf_tilt theta_dot_m_tilt tau_m_tilt G_tilt theta_dot_g_tilt b_tilt J_tilt K_e_tilt

s      = [ I      I_dot      theta_ddot_g      V_s      R_m
    L_m V_emf theta_dot_m      tau_m      G      theta_dot_g      b      J      K_e K_i ];
s_pan  = [ I_pan  I_dot_pan  theta_ddot_g_pan  V_s_pan  R_m
    L_m V_emf theta_dot_m_pan  tau_m_pan  G_pan  theta_dot_g_pan  b_pan  J_pan  K_e K_i ];
s_tilt = [ I_tilt I_dot_tilt theta_ddot_g_tilt V_s_tilt R_m
    L_m V_emf theta_dot_m_tilt tau_m_tilt G_tilt theta_dot_g_tilt b_tilt J_tilt K_e K_i ];

x_pan  = subs(x_i, s, s_pan)
```

x_pan =

$$\begin{pmatrix} I_{\mathrm{pan}} \\ \dot{\theta}_{g,\mathrm{pan}} \end{pmatrix}$$

```
x_tilt = subs(x_i, s, s_tilt)
```

x_tilt =

$$\begin{pmatrix} I_{\mathrm{tilt}} \\ \dot{\theta}_{g,\mathrm{tilt}} \end{pmatrix}$$

```
x_dot_pan  = subs(x_dot_i, s, s_pan);
x_dot_tilt = subs(x_dot_i, s, s_tilt);

u_pan  = subs(u_i, s, s_pan);
u_tilt = subs(u_i, s, s_tilt);

A_pan  = subs(A_i, s, s_pan)
```

A_pan =

$$\begin{pmatrix} -\frac{R_m}{L_m} & -\frac{K_e}{G_{\mathrm{pan}}\, L_m} \\ \frac{K_i}{G_{\mathrm{pan}}\, J_{\mathrm{pan}}} & -\frac{b_{\mathrm{pan}}}{J_{\mathrm{pan}}} \end{pmatrix}$$

```
A_tilt  = subs(A_i, s, s_tilt)
```

A_tilt =

$$\begin{pmatrix} -\frac{R_m}{L_m} & -\frac{K_e}{G_{\mathrm{tilt}}\,L_m} \\ \frac{K_i}{G_{\mathrm{tilt}}\,J_{\mathrm{tilt}}} & -\frac{b_{\mathrm{tilt}}}{J_{\mathrm{tilt}}} \end{pmatrix}$$

```
B_pan  = subs(B_i, s, s_pan)
```

B_pan =

$$\begin{pmatrix} \frac{1}{L_m} \\ 0 \end{pmatrix}$$

```
B_tilt  = subs(B_i, s, s_tilt)
```

B_tilt =

$$\begin{pmatrix} \frac{1}{L_m} \\ 0 \end{pmatrix}$$

We can now combine these into one large model

```
x = [ x_pan; x_tilt ]
```

x =

$$\begin{pmatrix} I_{\mathrm{pan}} \\ \dot{\theta}_{g,\mathrm{pan}} \\ I_{\mathrm{tilt}} \\ \dot{\theta}_{g,\mathrm{tilt}} \end{pmatrix}$$

```
x_dot = [ x_dot_pan; x_dot_tilt ]
```

x_dot =

$$\begin{pmatrix} \dot{I}_{\mathrm{pan}} \\ \ddot{\theta}_{g,\mathrm{pan}} \\ \dot{I}_{\mathrm{tilt}} \\ \ddot{\theta}_{g,\mathrm{tilt}} \end{pmatrix}$$

```
A_sym = [
    A_pan    zeros(2);
    zeros(2) A_tilt
]
```

A_sym =

$$\begin{pmatrix} -\frac{R_m}{L_m} & -\frac{K_e}{G_{\mathrm{pan}}\,L_m} & 0 & 0 \\ \frac{K_i}{G_{\mathrm{pan}}\,J_{\mathrm{pan}}} & -\frac{b_{\mathrm{pan}}}{J_{\mathrm{pan}}} & 0 & 0 \\ 0 & 0 & -\frac{R_m}{L_m} & -\frac{K_e}{G_{\mathrm{tilt}}\,L_m} \\ 0 & 0 & \frac{K_i}{G_{\mathrm{tilt}}\,J_{\mathrm{tilt}}} & -\frac{b_{\mathrm{tilt}}}{J_{\mathrm{tilt}}} \end{pmatrix}$$

```
B_sym = [
    B_pan              zeros(size(B_pan));
    zeros(size(B_pan)) B_tilt
]
```

B_sym =

$$\begin{pmatrix} \frac{1}{L_m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{L_m} \\ 0 & 0 \end{pmatrix}$$

```
% Defining outputs to be the velocities
C = [
    0 1 0 0;
    0 0 0 1
];
D = 0;


u = [ u_pan; u_tilt ]
```

u =

$$\begin{pmatrix} V_{s,\mathrm{pan}} \\ V_{s,\mathrm{tilt}} \end{pmatrix}$$

Adding Real Parameters

```
G_tilt_real = 15/48; % measured by counting gear teeth on the system
G_pan_real = G_tilt_real;

% Use moment of inertia calculated in 'inertia.mlx'
J_tilt_real = double(J_tilt_out)
```

J_tilt_real = 0.0213

```
J_pan_real = vpa(J_pan_out)
```

J_pan_real = $0.0433130440995\sin(\theta_{\mathrm{tilt}})^2 + 0.059756512932$

```
% Sines are hard to work with in state space models, we therefore simply
% take the largest value of the moment of inertia.
J_pan_real = double(subs(J_pan_real, theta_tilt, pi/2))
```

```
J_pan_real = 0.1031
```

```


% Motor constants (K_e = K_i in SI units)
K_i_real = 0.509;
K_e_real = K_i_real;
R_m_real = 7.101;
L_m_real = 3.4E-3;

% Friction coefficients
b_pan_real  = 0 % TODO: system identification
```

```
b_pan_real = 0
```

```
b_tilt_real = 0 % TODO: system identification
```

```
b_tilt_real = 0
```

```



% substitute into the model
s      = [ b_pan      b_tilt      L_m      G_pan
    b_pan J_pan      K_e_pan  R_m      L_m_tilt G_tilt      b_tilt J_tilt      K_e      K_i
s_real = [ b_pan_real b_tilt_real L_m_real G_pan_real
    b_pan J_pan_real K_e_real R_m_real L_m_real G_tilt_real b_tilt J_tilt_real K_e_real K_i_rea

A = double(subs(A_sym, s, s_real))

eig(A)

B = double(subs(B_sym, s, s_real))
C
D
```

```
D = 0
```

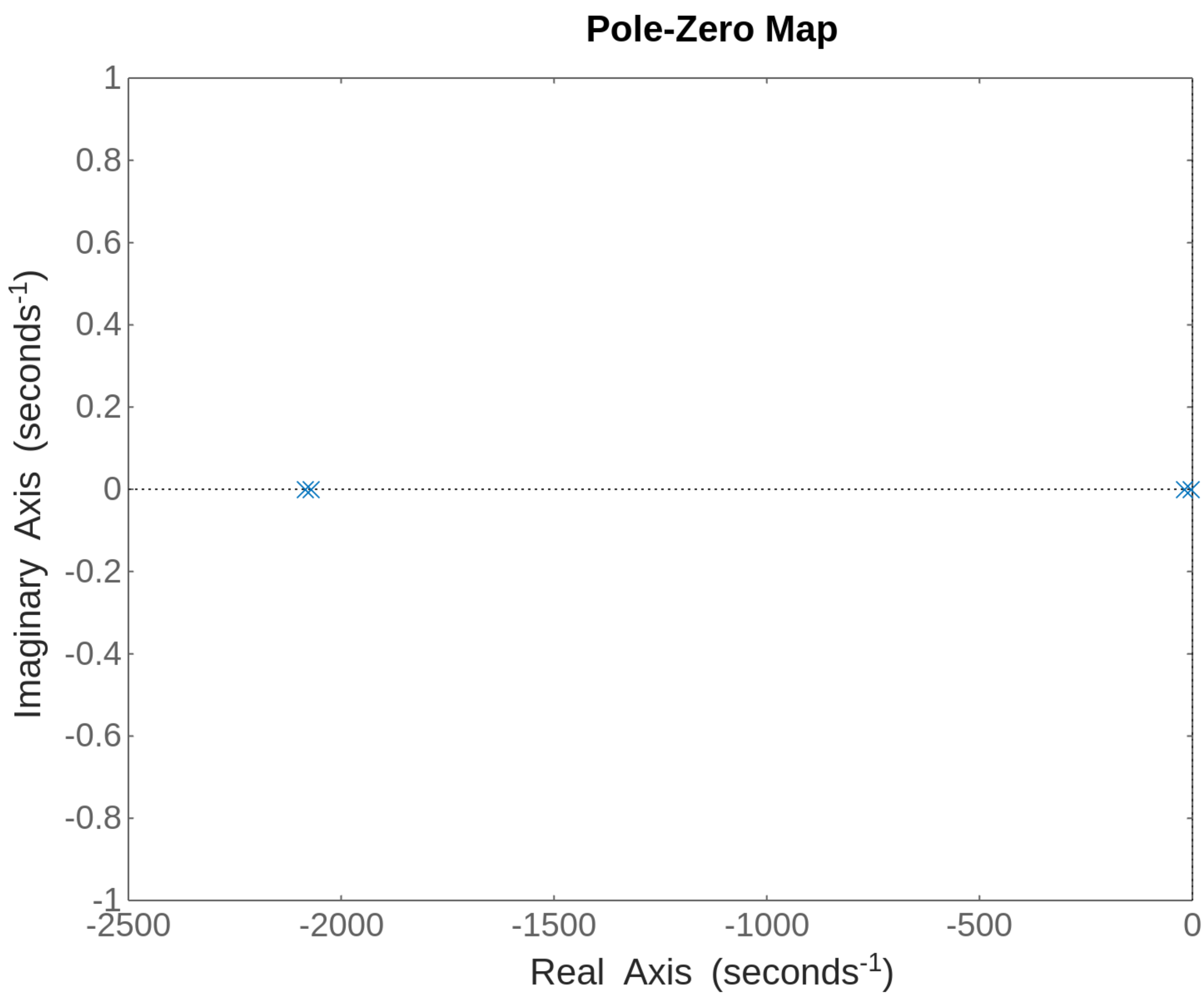
```
y = C*x + D*u
```

y =

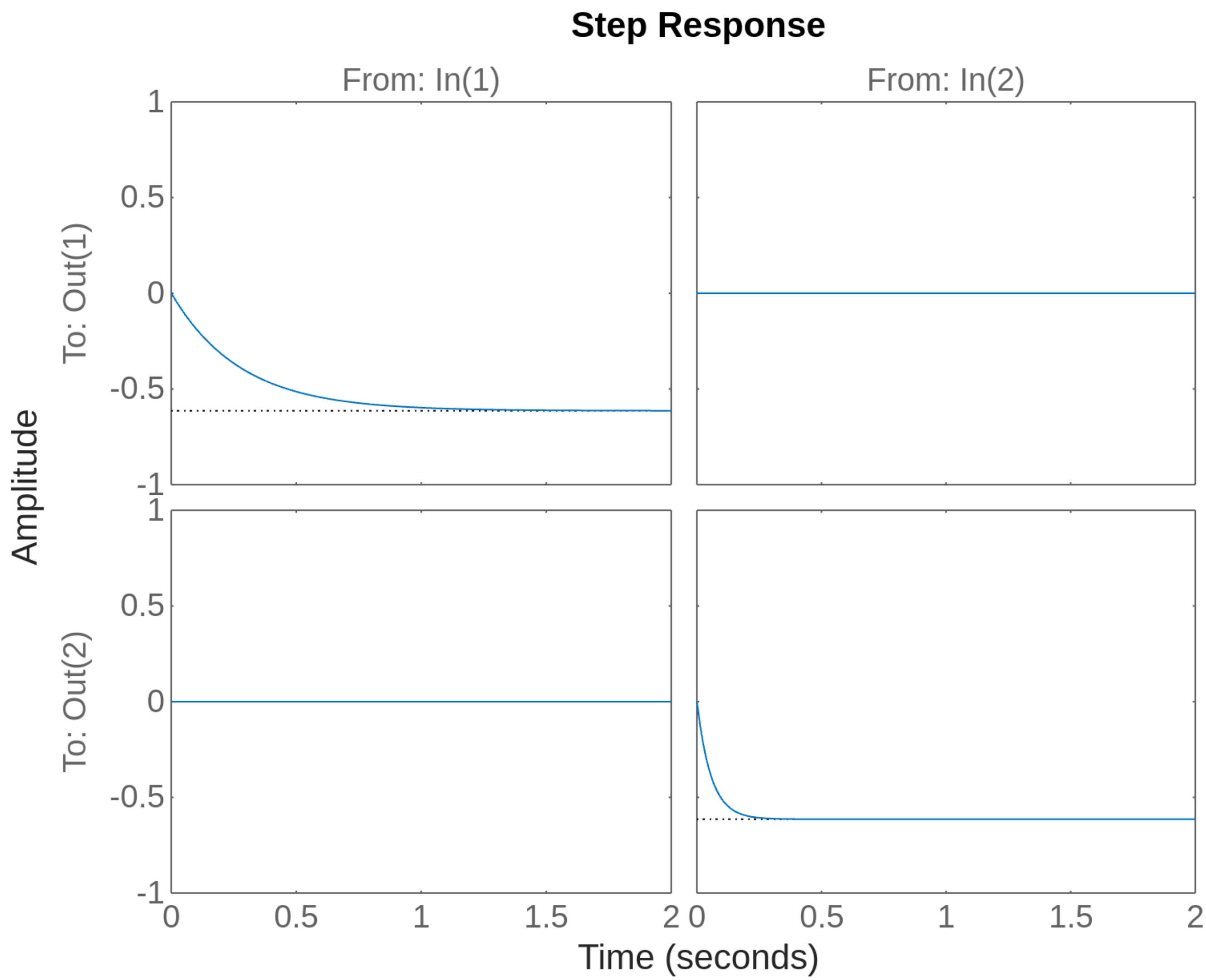
$$\begin{pmatrix} \dot{\theta}_{g,\mathrm{pan}} \\ \dot{\theta}_{g,\mathrm{tilt}} \end{pmatrix}$$

```



sys = ss(A, -B, C, D);
```

```
pzmap(sys)
```




```
step(sys)
```



Controlability

```
Con = [B A*B];
rank(Con)
```

```
ans = 4
```

The system is controlable as the controlability has rank equal to the number of system state variables

Regulator - State Feedback

```
t_s = 1;
alpha = 5;
sigma = -log(alpha/100)/t_s
```

```
sigma = 2.9957
```

```
A_e = [
```

```
        A zeros(4,2);
        C zeros(2,2);
    ]
B_e = [
        B;
        zeros(2,2);
]
C_e = [
        C zeros(2,2);
]
```

Place Poles

```
poles = [-0 -0 -1 -1 -2 -2] - 80;


F_e = place(A_e, -B_e, poles)
eig(A_e + B_e * F_e)
```

```
ans = 6×1 complex
 -82.0000 + 0.0000i
 -82.0000 + 0.0000i
 -81.0000 + 0.0000i
 -81.0000 - 0.0000i
 -80.0000 + 0.0000i
 -80.0000 + 0.0000i
```

```


latex(vpa(F_e, 3))
```

```
ans =
'\left(\begin{array}{cccccc} 6.27 & -2.61 & -1.5e-10 & -3.18e-10 & -114.0 & -1.29e-8\\ -2.6e-11 & -2.67e-10 & 6.27 &
```

Split into F and FI

```
F = F_e(:, 1:4)
FI = F_e(:, 5:6)
eig(A + B*F)
```

```
ans = 4×1 complex
10^2 ×
  -1.2150 + 0.7014i
  -1.2150 - 0.7014i
  -1.2150 + 0.7014i
  -1.2150 - 0.7014i
```

Observer

Observer matrix

```
O = [C; C*A; C*A^2; C*A^3];
rank(O)
```

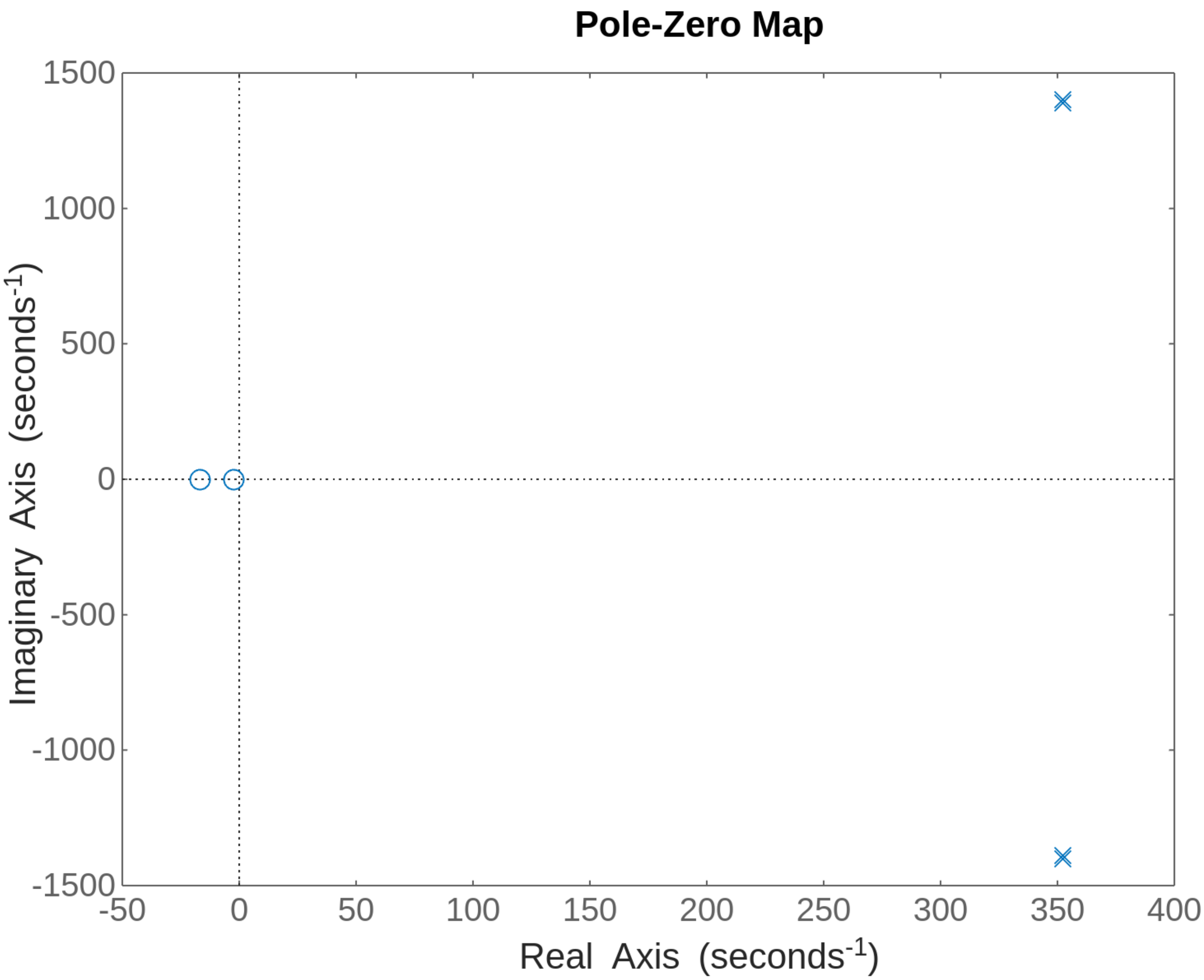
```
ans = 4
```

The system poles where placed at

```
poles
```

The oberver poles must be at around seven times more to the left

```
c = 7;
observer_poles = poles(:, 3:6)*c

L = place(A', -C', observer_poles)'
eig(A + L*C)

observer = ss(A+B*F+L*C, -L, F, zeros(2));

pzmap(observer)
```


Pole-Zero Map
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
1500
1000
500
0
-500
-1000
-1500
-50
0
50
100
150
200
250
300
350
400