Design a Controller: PD Control

Mechanical system of inverted pendulum

$$\Theta(s) = G(s)\boxed{F(s)}$$

$$G(s) = \frac{1}{-\alpha s^2 + \beta}$$

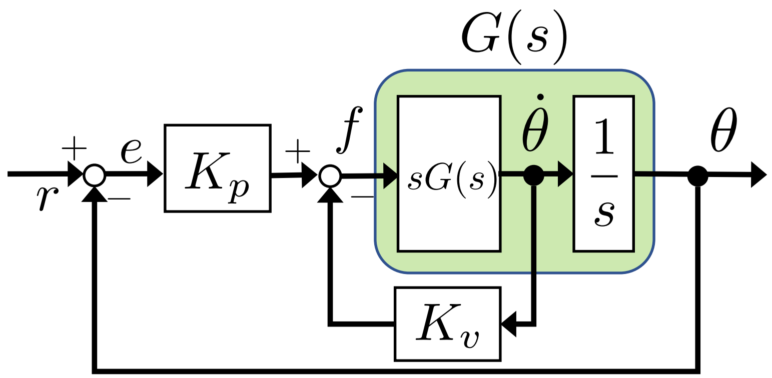
Control law: PD controller

$$f = -\{K_p(r-\theta) - K_v\dot{\theta}\}$$

$$\begin{aligned} F(s) &= -\{K_p(R(s) - \Theta(s)) - sK_v\dot{\Theta}(s)\} \\ &= -K_p(R(s) - \Theta(s)) + sK_v\dot{\Theta}(s) \end{aligned}$$

$$\Theta(s) = G(s)\{-K_p(R(s) - \Theta(s)) + sK_v\Theta(s)\}$$

$$\{1 - (K_p + sK_v)G(s)\}\Theta(s) = -G(s)K_pR(s)$$



Closed-loop transfer function

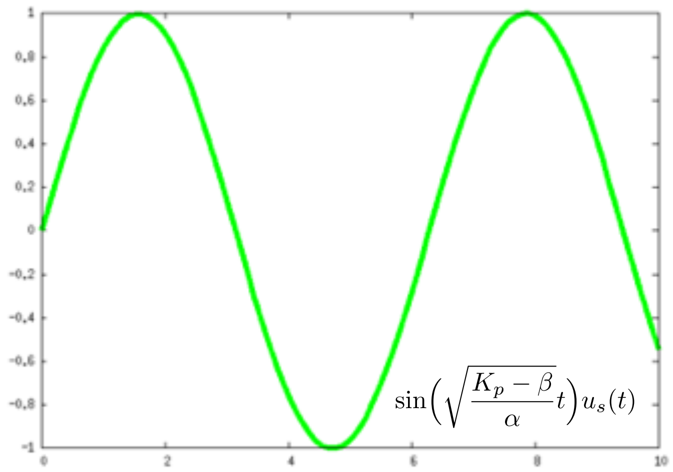
$$\frac{\Theta(s)}{R(s)} = \frac{-K_p}{\frac{1}{G(s)} - K_p - sK_v} = \frac{-K_p}{\boxed{-\alpha s^2 + \beta - K_p - sK_v}}$$

Characteristic equation = 0

$$\alpha s^2 + sK_v - (K_p - \beta) = 0 \quad \Rightarrow \quad s = \frac{-K_v \pm \sqrt{K_v^2 - 4\alpha(K_p - \beta)}}{2\alpha}$$

Selection of Gain Parameters#1: Stability Limitation

$$s = \frac{-K_v \pm \sqrt{K_v^2 - 4\alpha(K_p - \beta)}}{2\alpha}$$



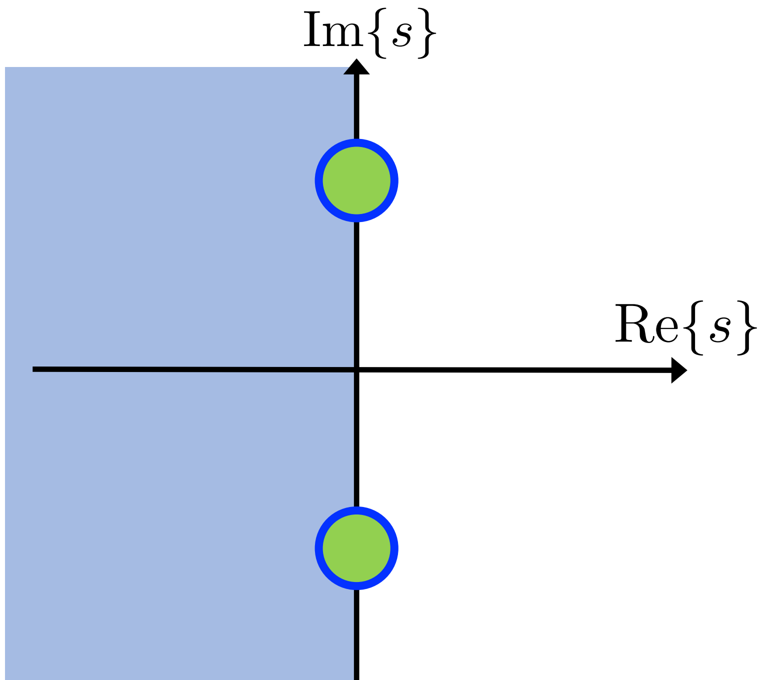
$$\boxed{Kv = 0} \rightarrow s = \pm\sqrt{\frac{-(K_p - \beta)}{\alpha}}$$

$K_p \leq \beta$: Unstable

$\boxed{K_p > \beta \text{ : Stability Limit}}$

$$s = \pm\sqrt{\frac{-(K_p - \beta)}{\alpha}}$$

$$s = \pm j\sqrt{\frac{K_p - \beta}{\alpha}}$$

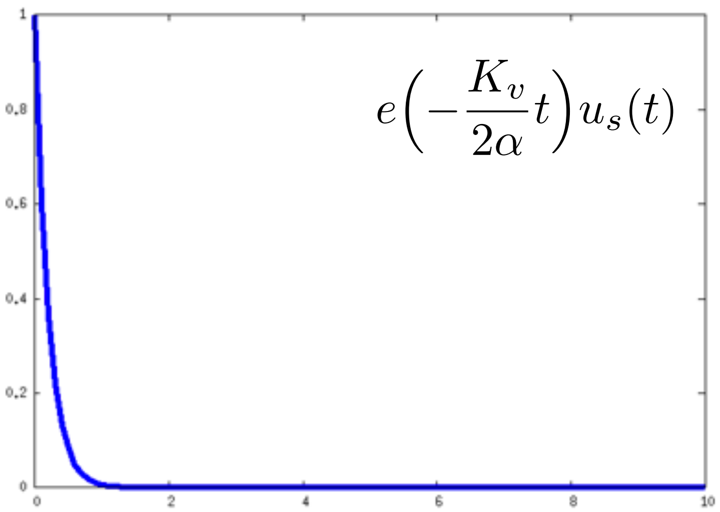


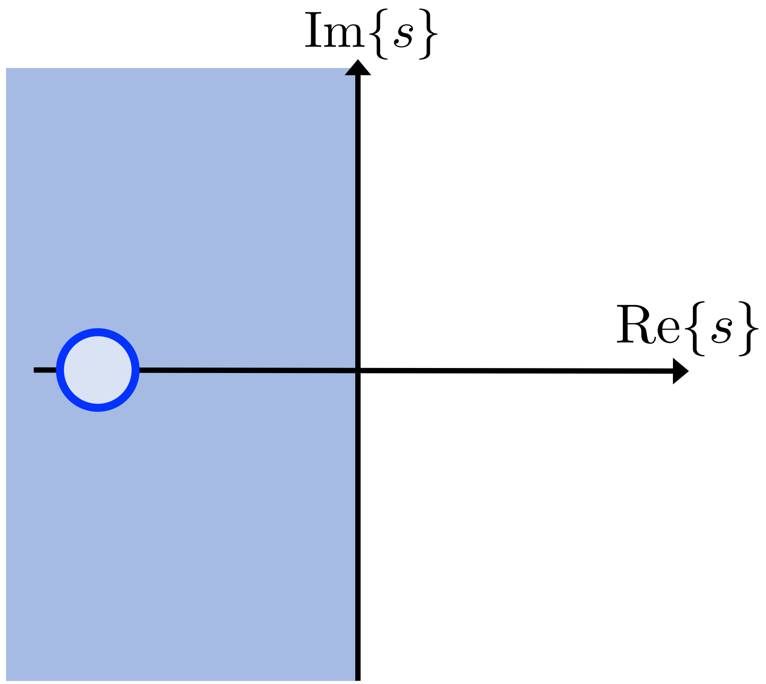
Selection of Gain Parameters#2: Critically Damped

$$s = \frac{-K_v \pm \sqrt{K_v^2 - 4\alpha(K_p - \beta)}}{2\alpha}$$

$$\sqrt{()} = 0 \rightarrow \boxed{K_v^2 = 4\alpha(K_p - \beta)}$$

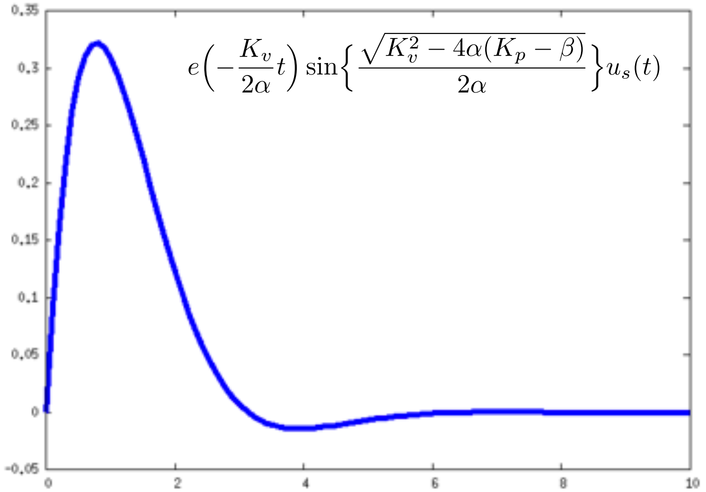
$$s = -\frac{K_v}{2\alpha} < 0$$



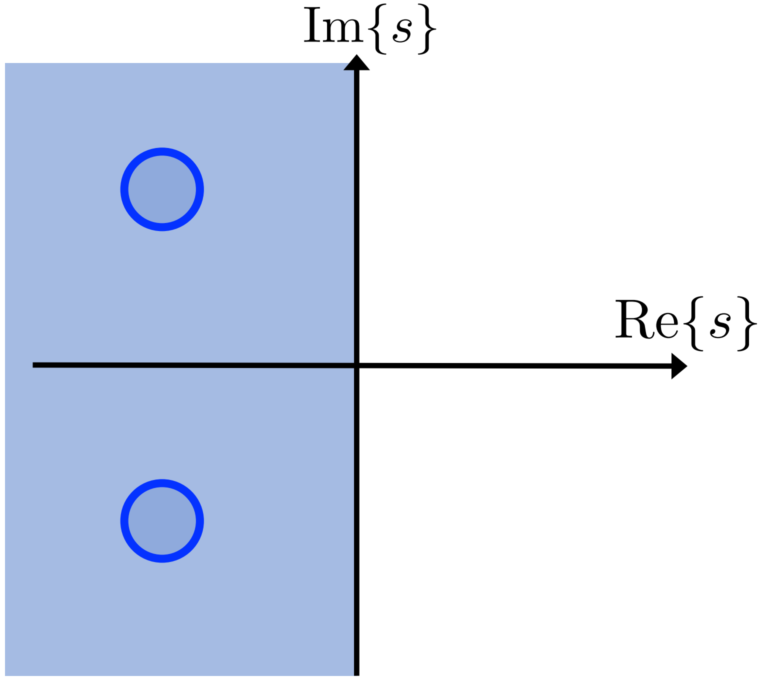


Selection of Gain Parameters #3: Underdamped

$$s = \frac{-K_v \pm \sqrt{K_v^2 - 4\alpha(K_p - \beta)}}{2\alpha}$$



$$\sqrt{()} < 0 \rightarrow \boxed{K_v^2 < 4\alpha(K_p - \beta)}$$



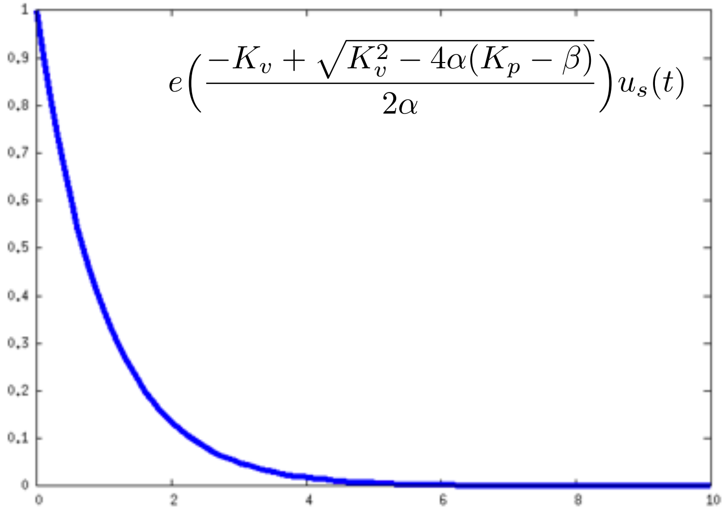
$$s = \frac{-K_v \pm j\sqrt{-\{K_v^2 - 4\alpha(K_p - \beta)\}}}{2\alpha} \left(-\frac{K_v}{2\alpha} < 0\right)$$

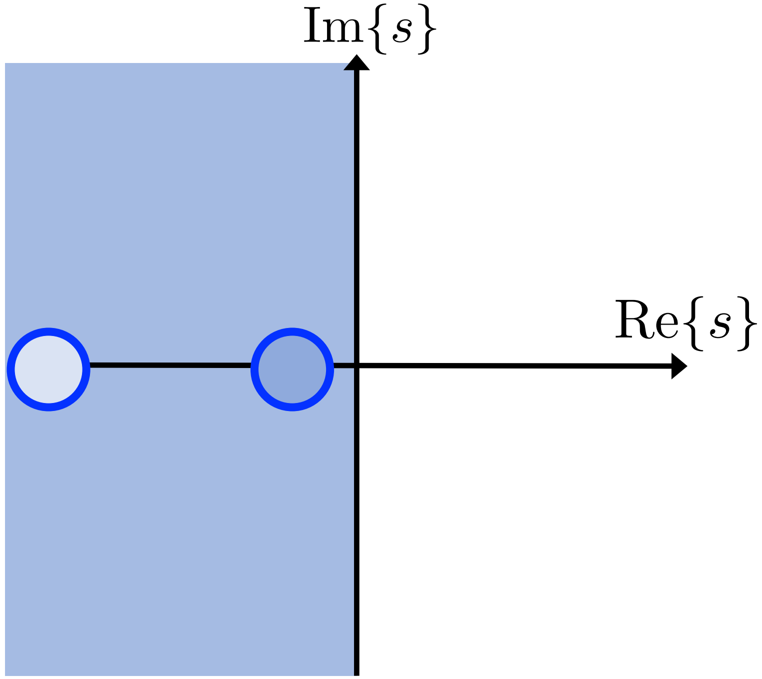
Selection of Gain Parameters #4: Overdamped

$$s = \frac{-K_v \pm \sqrt{K_v^2 - 4\alpha(K_p - \beta)}}{2\alpha}$$

$$\sqrt{()} > 0 \rightarrow \boxed{K_v^2 > 4\alpha(K_p - \beta)}$$

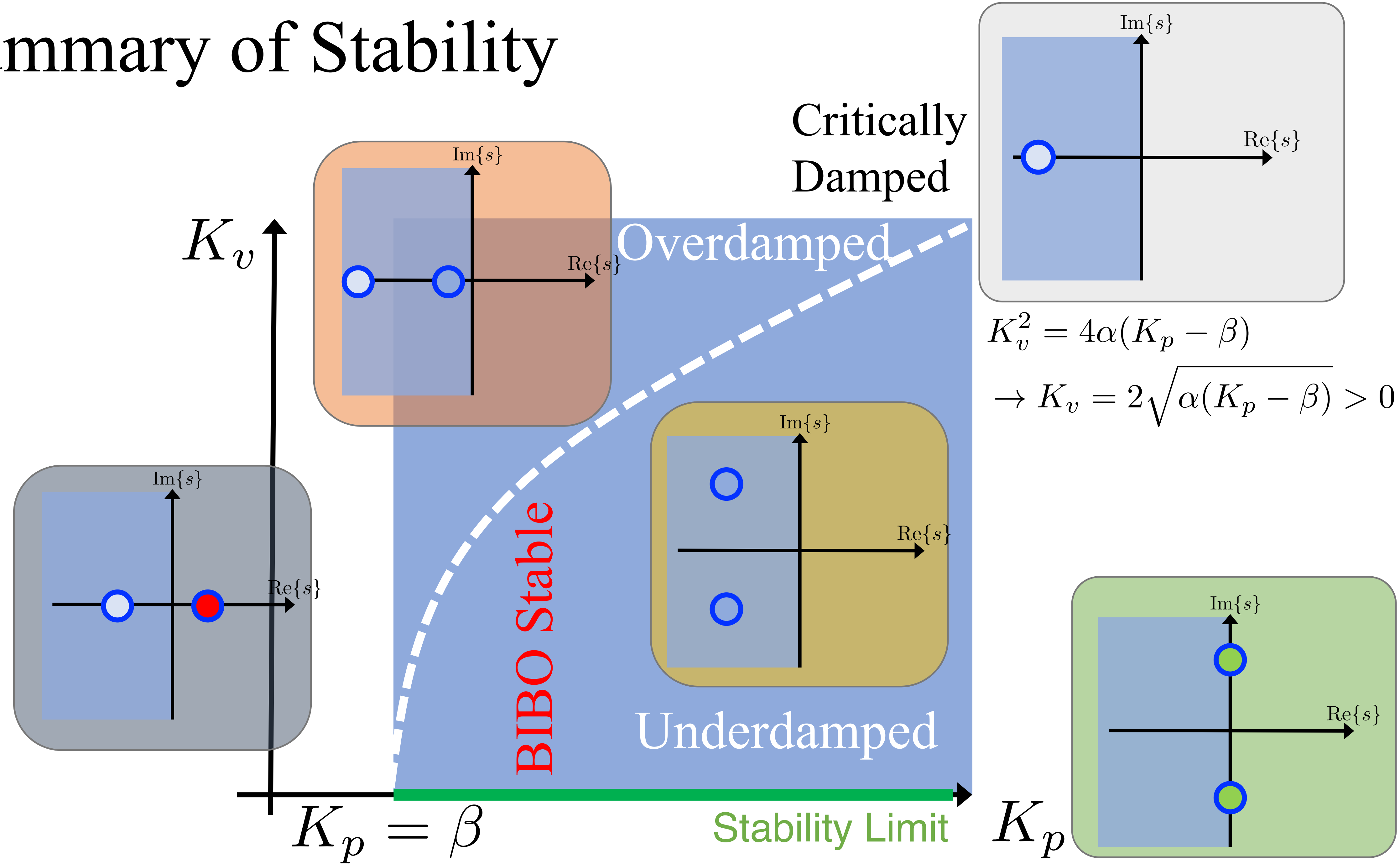
$$s = \frac{-K_v \pm \sqrt{K_v^2 - 4\alpha(K_p - \beta)}}{2\alpha} < 0 \ (K_v > \sqrt{K_v^2 - 4\alpha(K_p - \beta)})$$





Summary of Stability

Critically Damped



$K_v^2 = 4\alpha(K_p - \beta)$

$\rightarrow K_v = 2\sqrt{\alpha(K_p - \beta)} > 0$

Topics: Linear Control

I. Purpose of “Control”

II. Procedure of Control System Design

III. Control System Design for Inverted Pendulum

IV. Python: Stabilization of Inverted Pendulum

Python script #18-1 InvertedPendulum_odeint.py

```
from scipy.integrate import odeint
import numpy as np
import math
import sys

# import original modules
import video_InvertedPendulum as vip

m_th = 0.10#1.0      # mass theta [kg]
m_x  = 0.50#2.0      # mass x [kg]
I    = 0.01#0.00558389#1.0      # inertia 1 [kg m^2]
l_g  = 0.5      # length of pendulum [m]
g    = 9.80665 # gravitational accelaration[m/s^2]

K_p = float(sys.argv[1])#-100.0
K_v = float(sys.argv[2])#-89.6

theta_d = 0.0
dtheta_d = 0.0

params  = [m_th, m_x, I, l_g, g] # parameters
gains   = [K_p, K_v]             # gains
targets = [theta_d, dtheta_d]    # targets

# initial conditions(x0, dx0)
max_t = 5.0 # max_time [s]
dt = 0.01    # dt [s]

alpha = (I*(m_x+m_th))/(m_th*l_g) + m_x*l_g
beta  = (m_x + m_th)*g
```

Calculation for alpha and beta

"space"

$ python InvertedPendulum_odeint_odeint.py [Kp] [Kv]

sys.argv[0]

sys.argv[1] sys.argv[2]

video_InvertedPendulum.py: make animation (given script should be moved/copied to the same directory)

Python script #18-2 InvertedPendulum_odeint.py

```
if K_v*K_v - 4*alpha*(K_p-beta) < 0:
    s1_re = -K_v/(2*alpha)
    s2_re = -K_v/(2*alpha)

    s1_im =  math.sqrt(-(K_v*K_v - 4*alpha*(K_p-beta)))/(2*alpha)
    s2_im = -math.sqrt(-(K_v*K_v - 4*alpha*(K_p-beta)))/(2*alpha)
else:
    s1_re = -K_v/(2*alpha) + math.sqrt(K_v*K_v - 4*alpha*(K_p-beta))/(2*alpha)
    s2_re = -K_v/(2*alpha) - math.sqrt(K_v*K_v - 4*alpha*(K_p-beta))/(2*alpha)

    s1_im = 0.0
    s2_im = 0.0

#S = [math.sqrt(beta/alpha), 0] # Poles of the system (no inputs)
S = [s1_re, s1_im, s2_re, s2_im] # Poles of the system (no inputs)

sqr = 4*alpha*(K_p-beta)
if sqr < 0:
    sqr = -sqr

print('alpha={},beta={}'.format(alpha, beta))
print('K_v^2={}, 4alpha(K_p+beta)={}, sqr={}'.format(K_v*K_v,4*alpha*(K_p-beta), math.sqrt(sqr)))
```

Calculation for poles

Calculation for critical conditions

Python script #18-3 InvertedPendulum_odeint.py

```python
def Control(p):
    x, dx, theta, dtheta = p

    out = - K_p*(theta_d-theta) - K_v*(dtheta_d-dtheta)

    return out

def InvertedPendulum(p, t):
    x, dx, theta, dtheta = p

    if theta > math.pi:
        theta = theta - 2*math.pi
    elif theta < -math.pi:
        theta = theta + 2*math.pi

    M_11 = m_x + m_th
    M_12 = m_th*l_g*math.cos(theta)
    M_21 = m_th*l_g*math.cos(theta)
    M_22 = I + m_th*l_g*l_g

    #define matrix
    M = np.matrix([[M_11, M_12],[M_21, M_22]])
    N = np.matrix([[-m_th*l_g*math.sin(theta)*dtheta*dtheta],[0]])
    G = np.matrix([[0],[-m_th*g*l_g*math.sin(theta)]])
    F = np.matrix([[Control(p)],[0]])

    IM = np.linalg.inv(M) # calc Inverse matrix
    A = (-1)*IM.dot(N+G-F) # F is right hand side of equations

    ddx, ddtheta = A

    return [dx, ddx, dtheta, ddtheta]
```

PD control

$$f = -\{K_p(r-\theta) - K_v\dot{\theta}\}$$

Motion equation

$$\boldsymbol{F} = (f, 0)^T$$

$$\ddot{\boldsymbol{\theta}} = -\boldsymbol{M}(\boldsymbol{\theta})^{-1}\{\boldsymbol{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{F}\}$$

```python
t = np.arange(0.0, max_t, dt)
x0 = [0.0, 0.0, 0.35*math.pi, 0.0]
p = odeint(InvertedPendulum, x0, t)

vip.video(p, dt, max_t, params, gains, targets, S)
```

ODE calculation

visualization