Adaptive control tutorial

Gaze stabilization in animals

Rapid reflexive responses

Vestibulo-ocular reflex

A Comprehensive Gaze Stabilization Controller Based On Cerebellar Internal Models

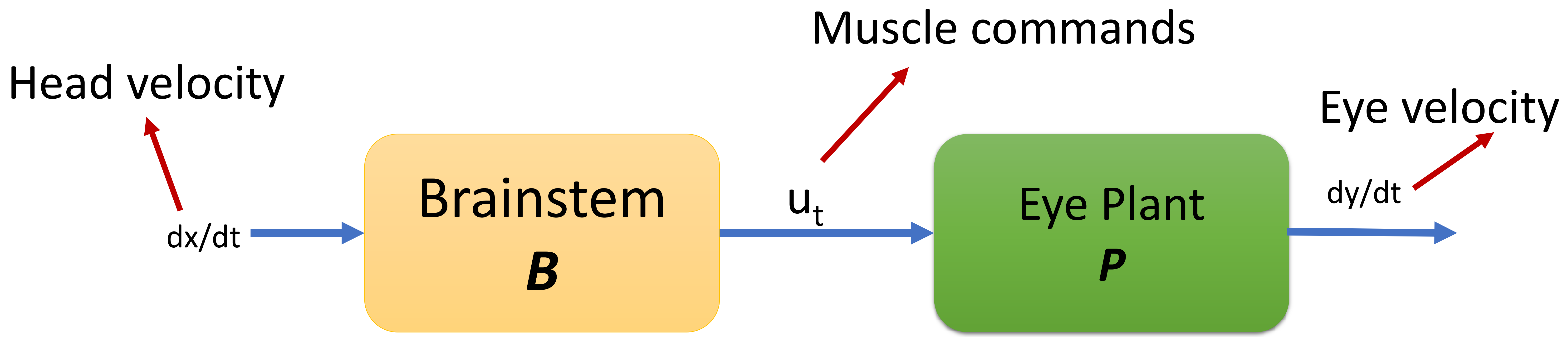
Lorenzo Vannucci, Egidio Falotico, Silvia Tolu, Vito Cacucciolo, Paolo Dario, Henrik Hautop Lund, Cecilia Laschi

The BioRobotics Institute

Scuola Superiore Sant'Anna

DTU

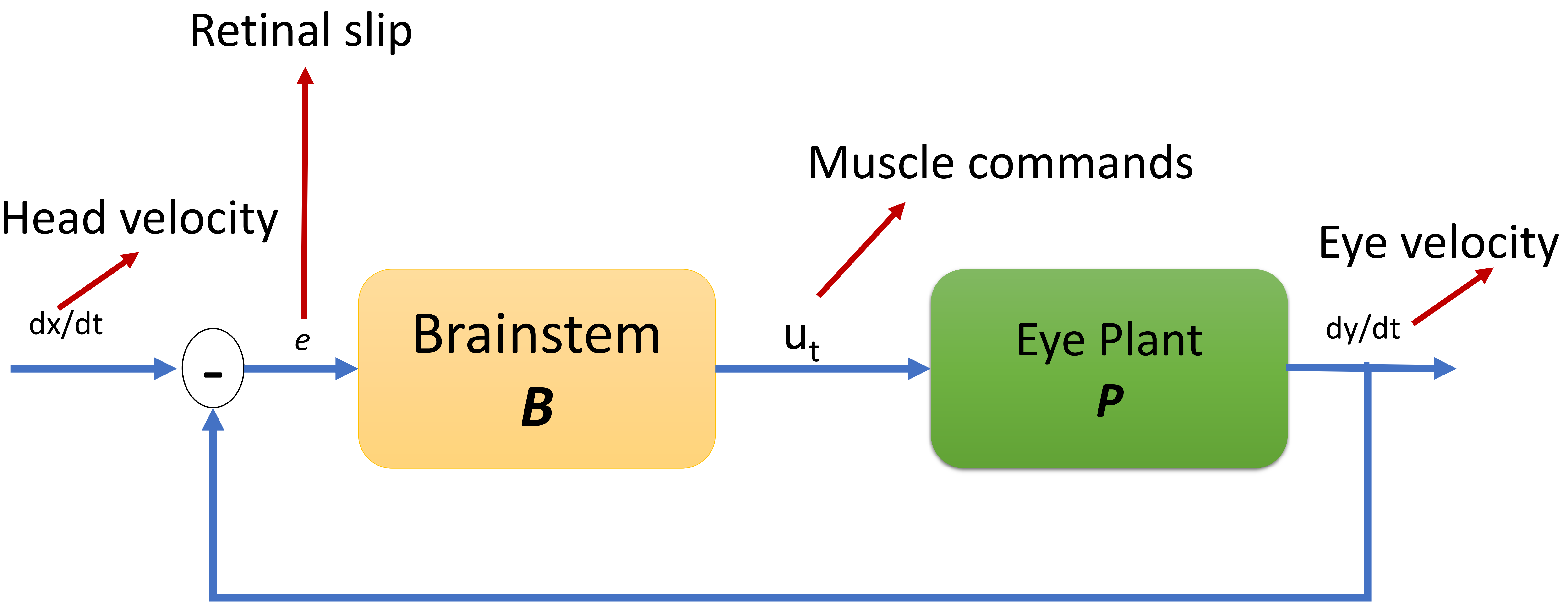
A direct mapping between head-eye velocity



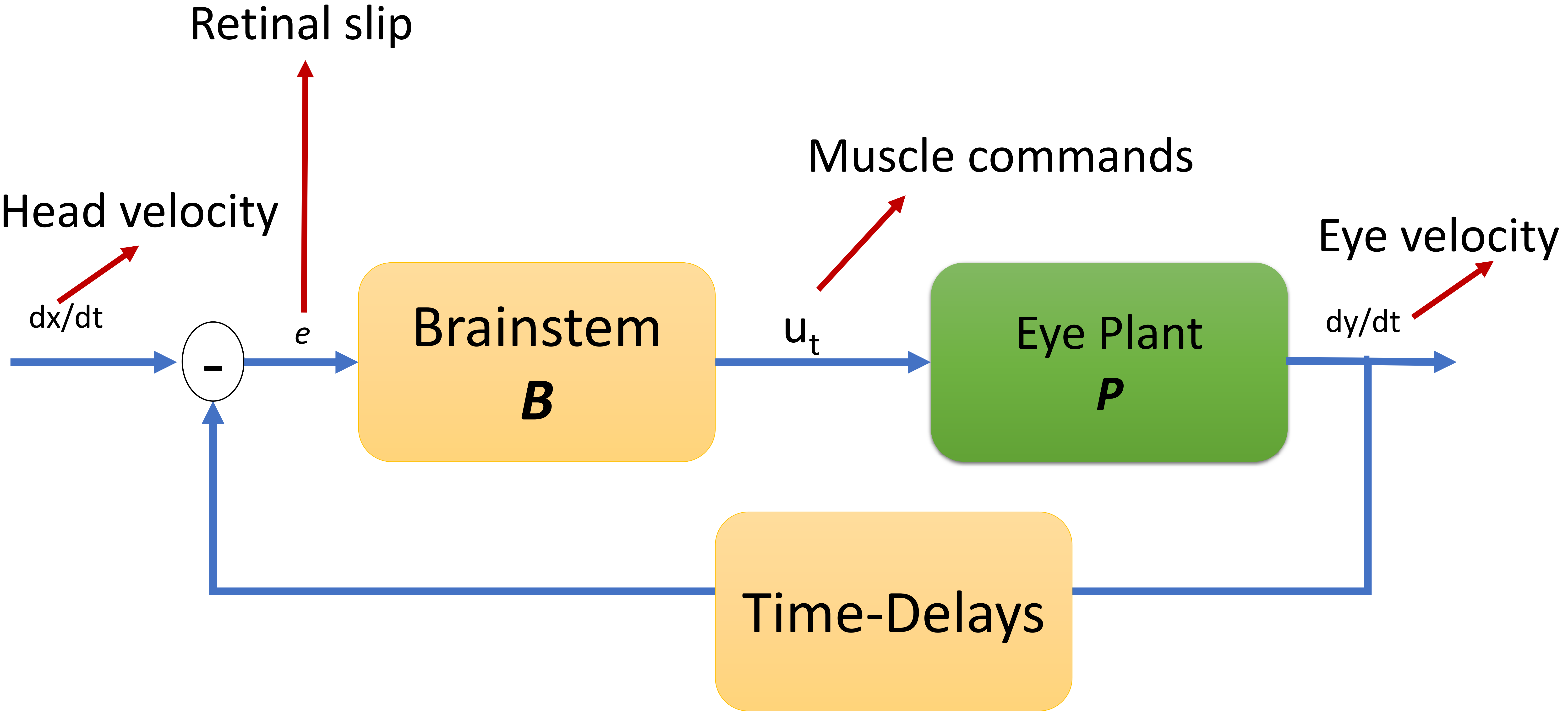
For a given head velocity, there exist a mapping that outputs motor commands
To generate equal and opposite eye velocity to stabilize the visual image

These direct mappings are problematic if there is any undesired effect such as
Noise, changes in the plant characteristics etc.,

Simple Feedback Control

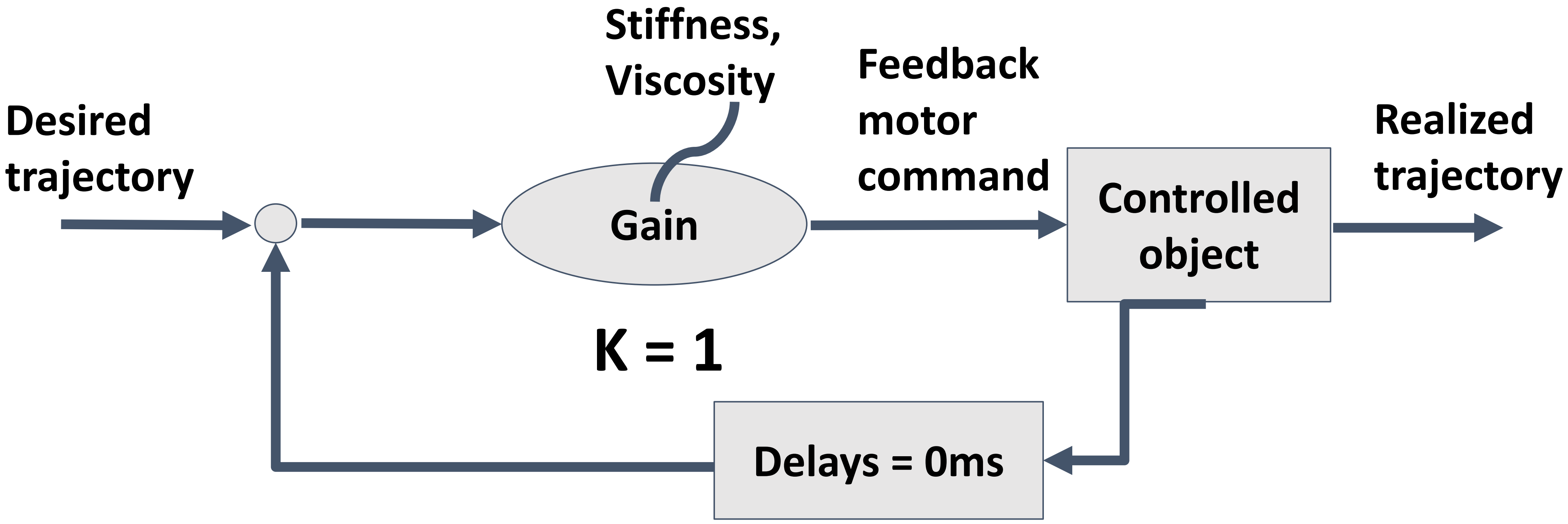


Problems with feedback delays



A simple example of the effect of feedback delays

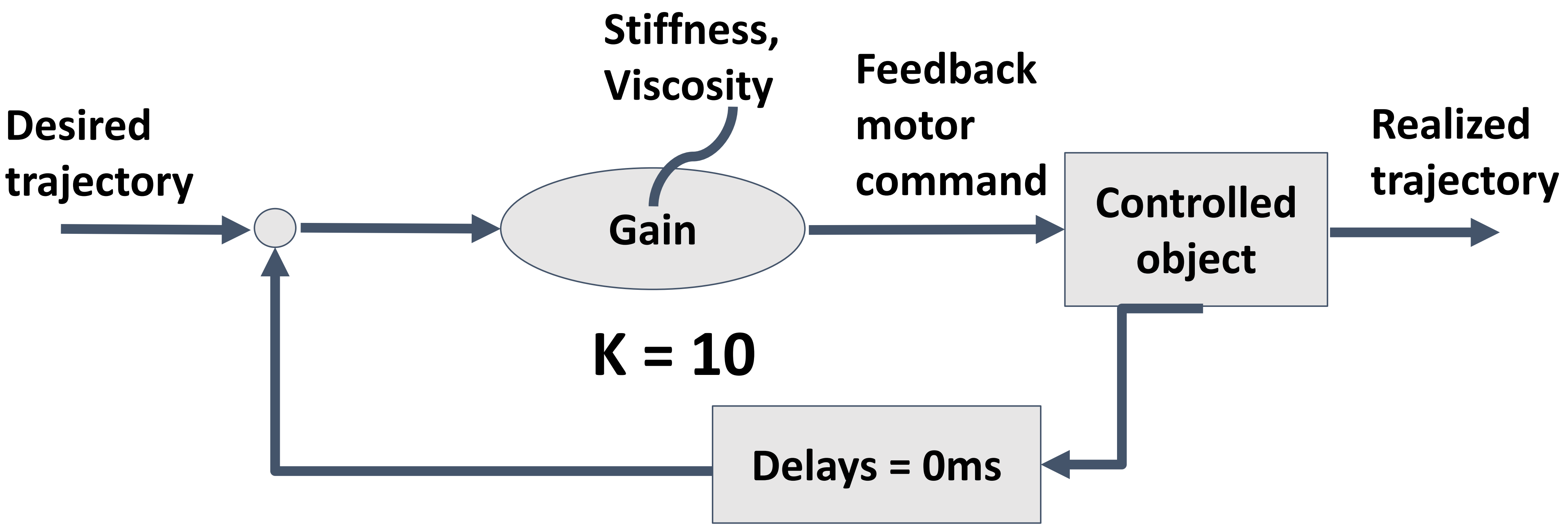
Consider a simple feedback control loop
with proportional feedback gain



Low gain + No delay

A simple example of the effect of feedback delays

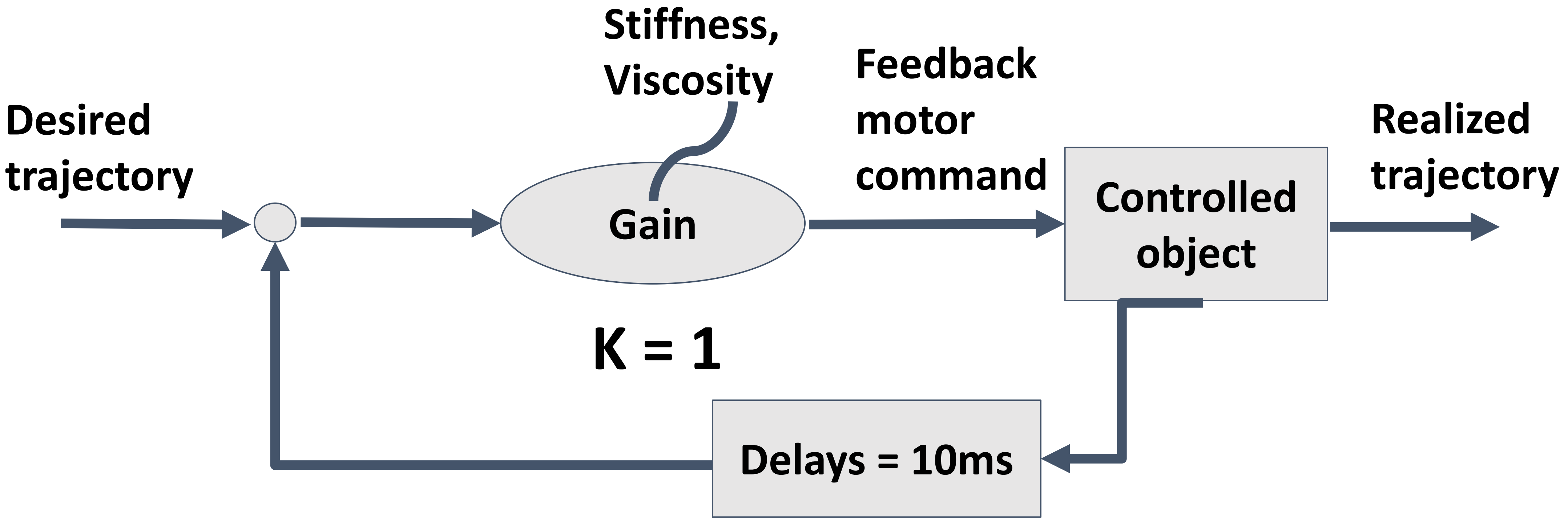
Consider a simple feedback control loop
with proportional feedback gain



High gain + No delay

A simple example of the effect of feedback delays

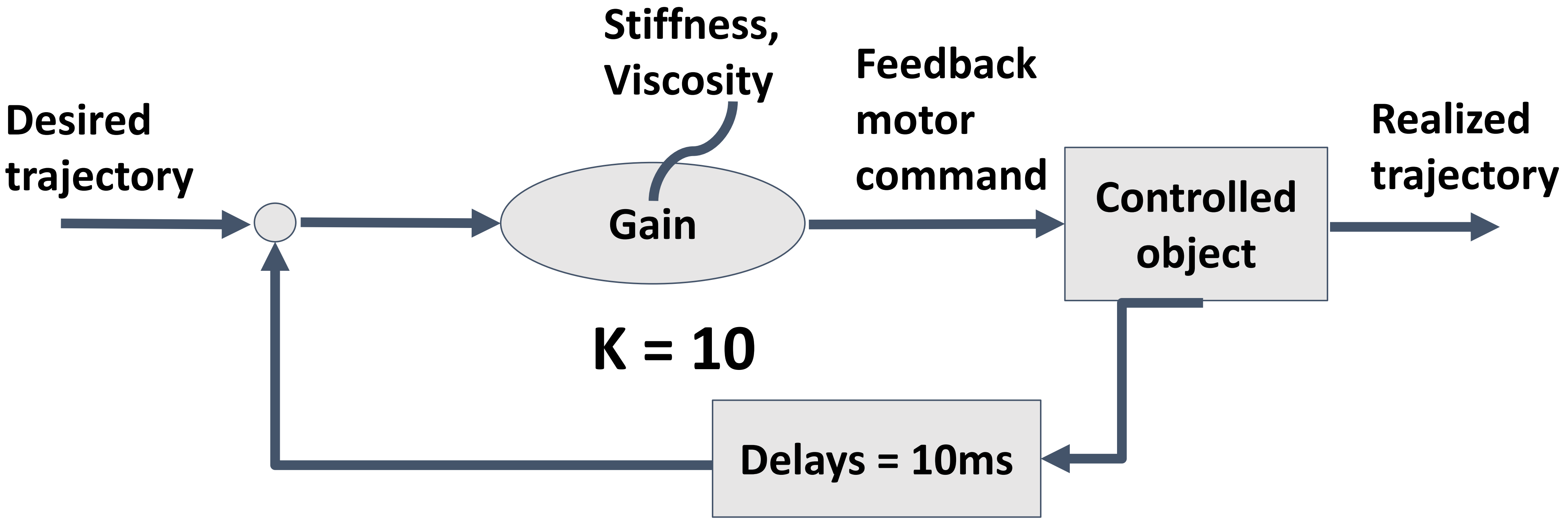
Consider a simple feedback control loop
with proportional feedback gain



Low gain + delay

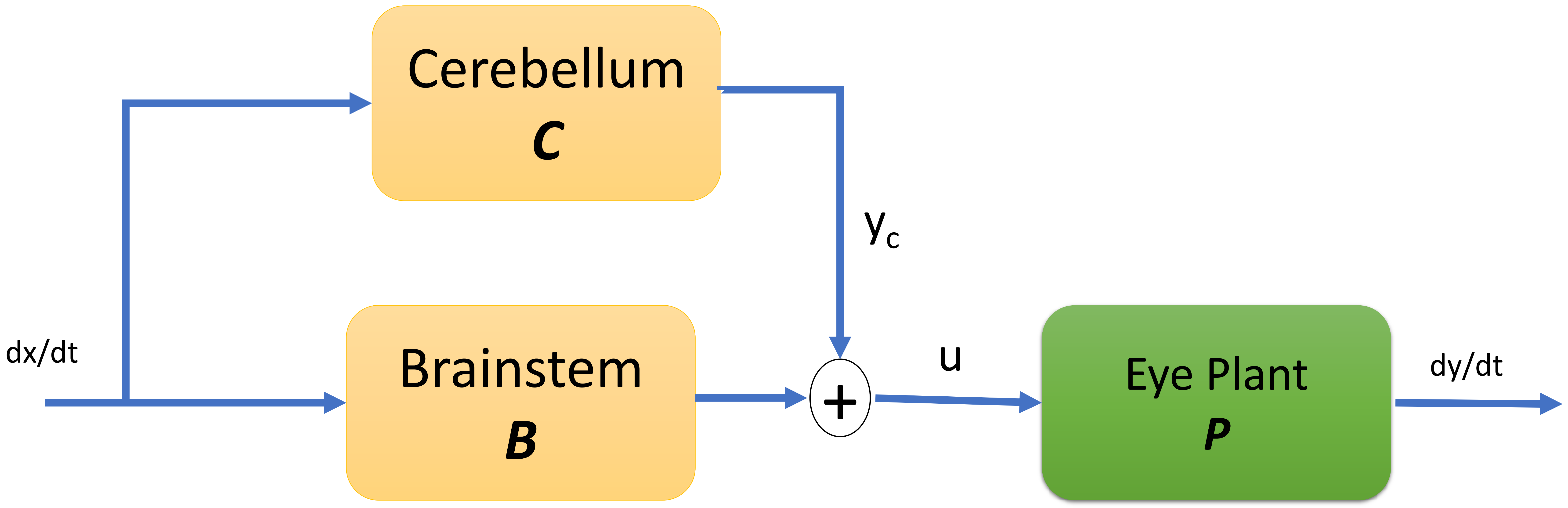
A simple example of the effect of feedback delays

Consider a simple feedback control loop
with proportional feedback gain

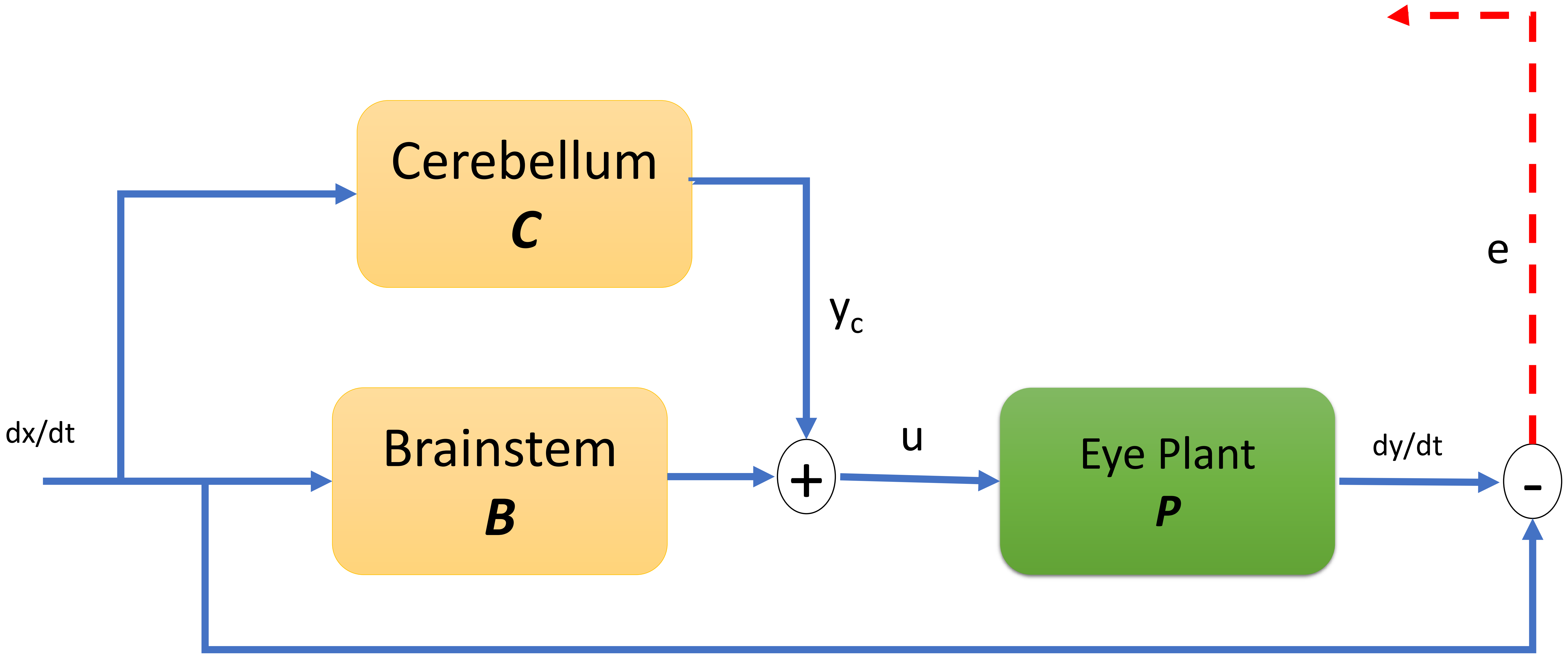


High gain + delay

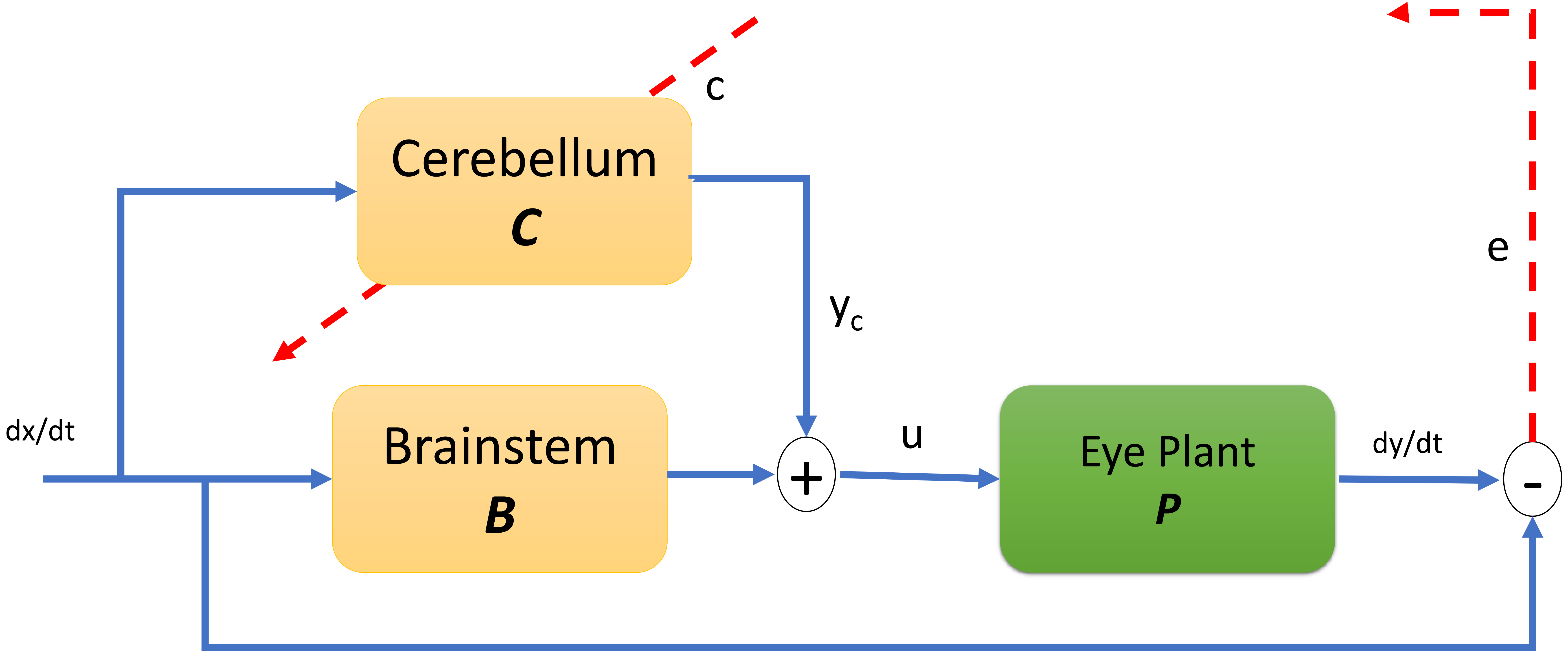
Cerebellum based adaptive control – inverse model



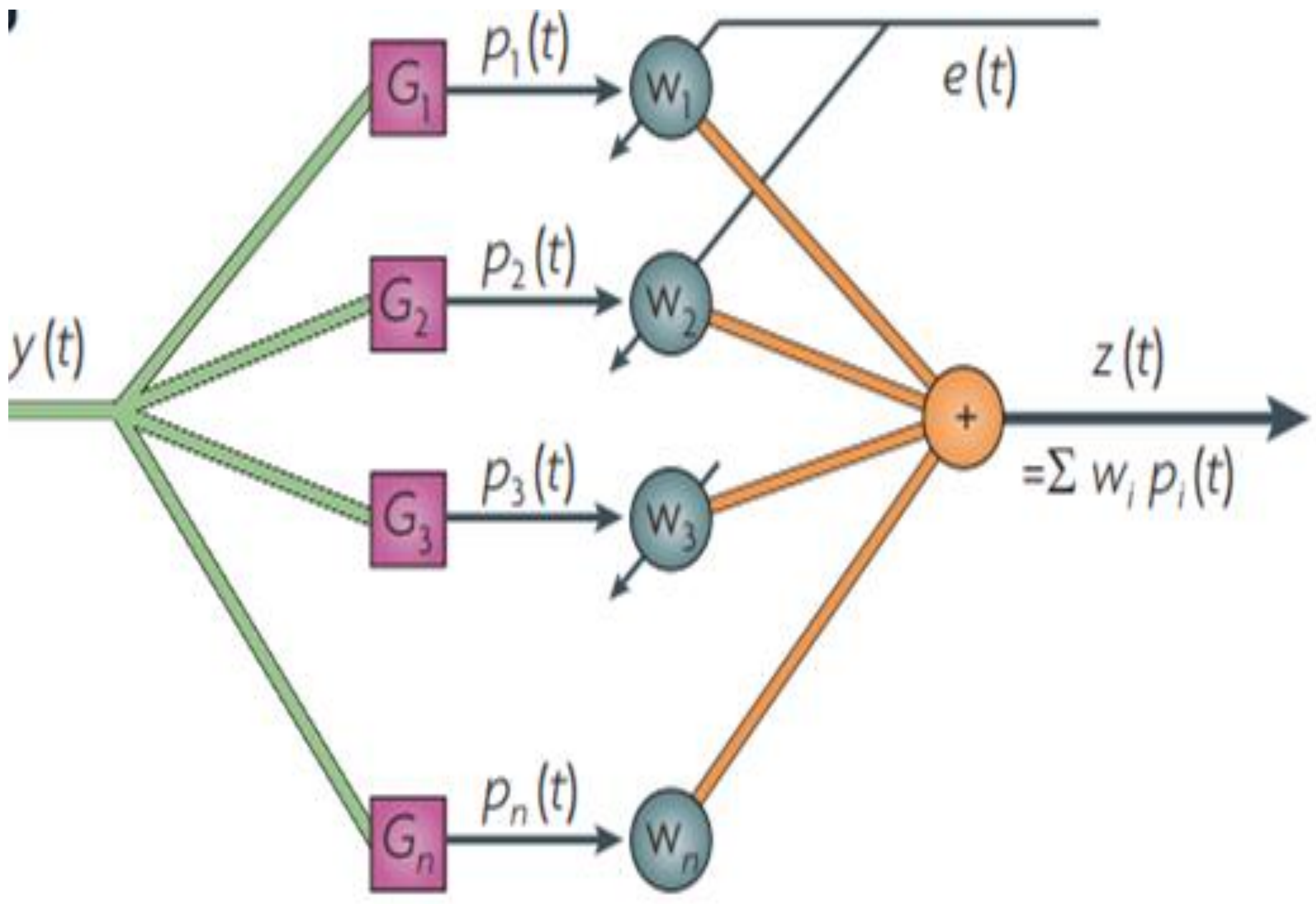
Cerebellum based adaptive control – inverse model



Cerebellum based adaptive control – inverse model



The computational circuit of cerebellum - adaptive filter



1) $p(t) = G \; * y(t)$

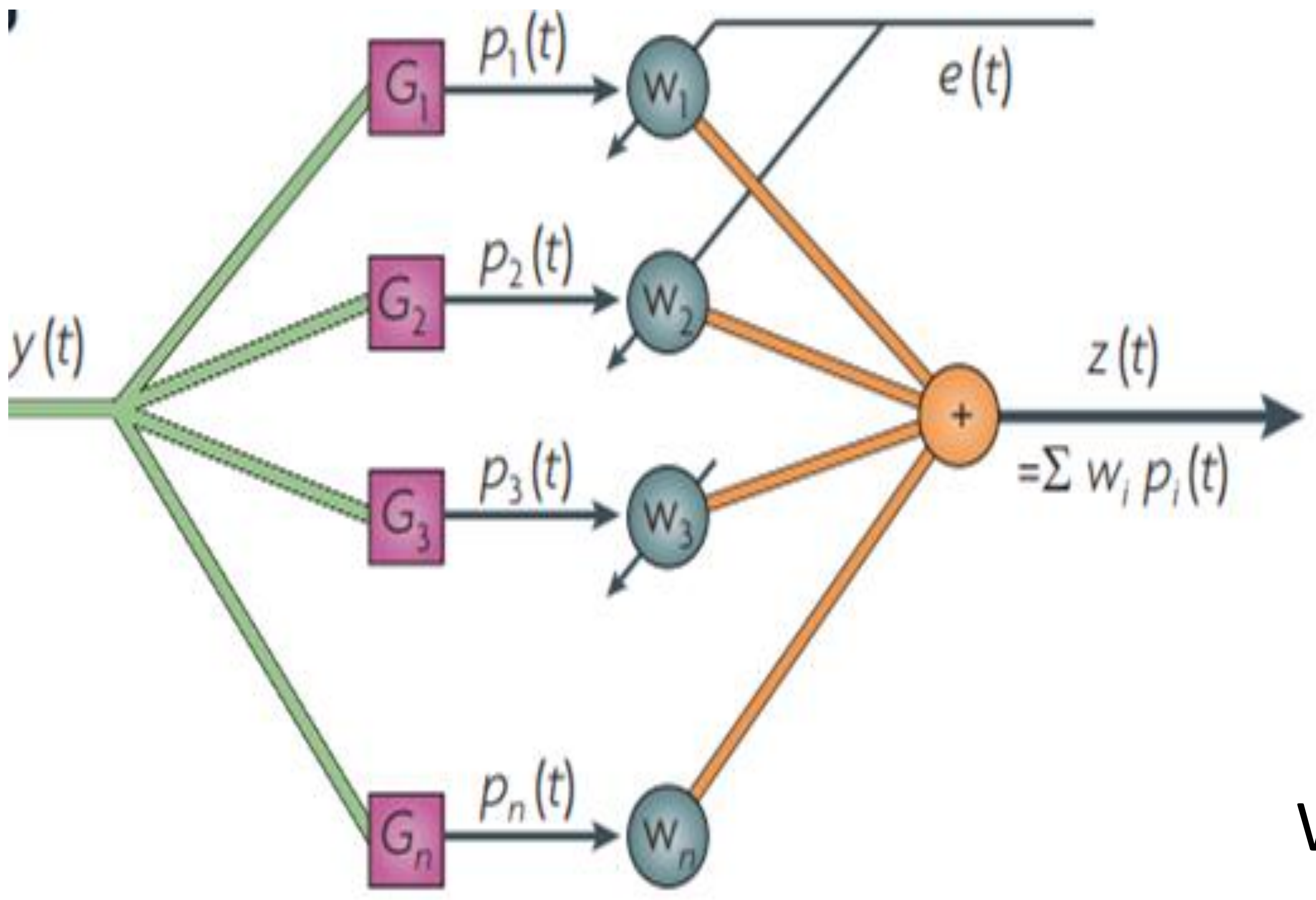
$(n\ X\ 1) = (n\ X\ m) \; * (m\ X\ 1)$
s.t., n >= m

2) $z(t) = W \; * p(t)$

$(k\ X\ 1) = (k\ X\ n) \; * (n\ X\ 1)$
s.t., k <= n

Note: 'W' represents the vector of weights {w1, w2, w3…..,wn}.
'G' represents the input signal weighting

The computational circuit of cerebellum - adaptive filter

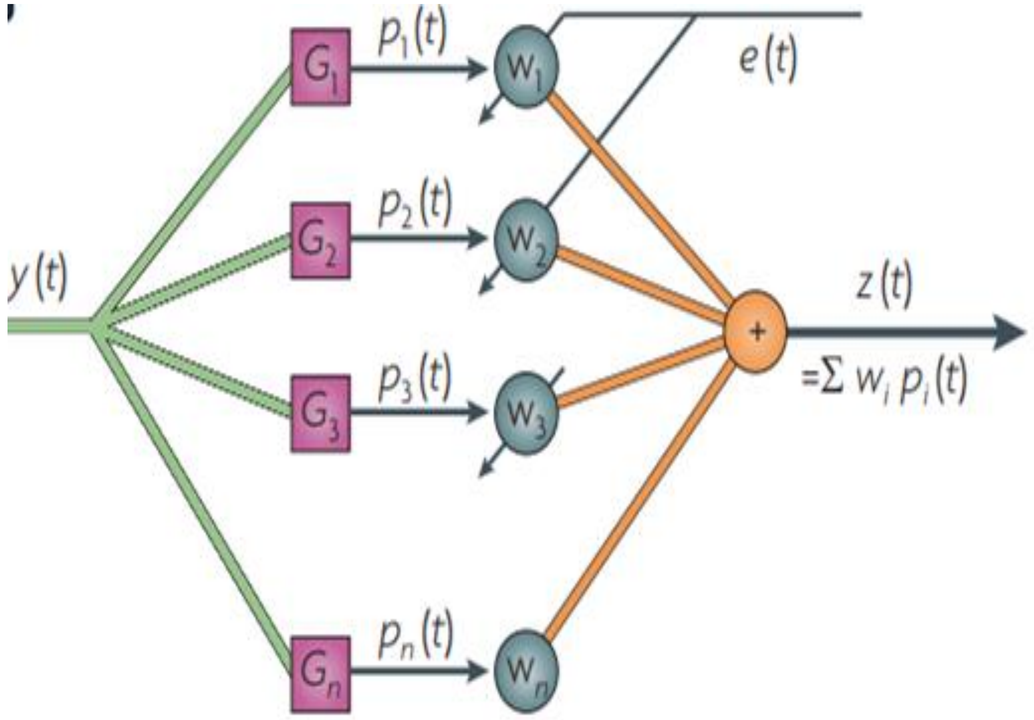


$$learning\ rule$$

$$3)\ \delta w(t)\ \boldsymbol{\alpha}\ -e(t).p(t)$$

Weight update proportional to the negative product of the error received by the cerebellum and the basis activity ‘p’

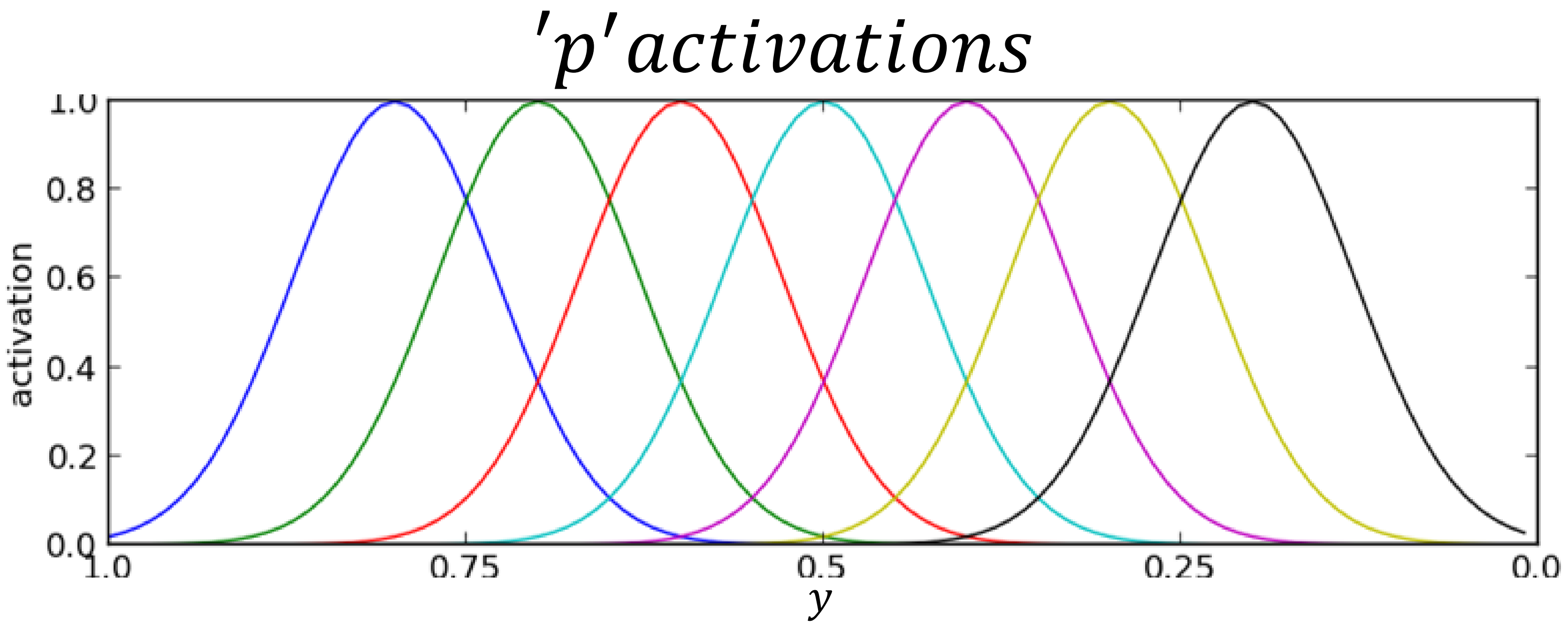
The computational circuit of cerebellum - adaptive filter



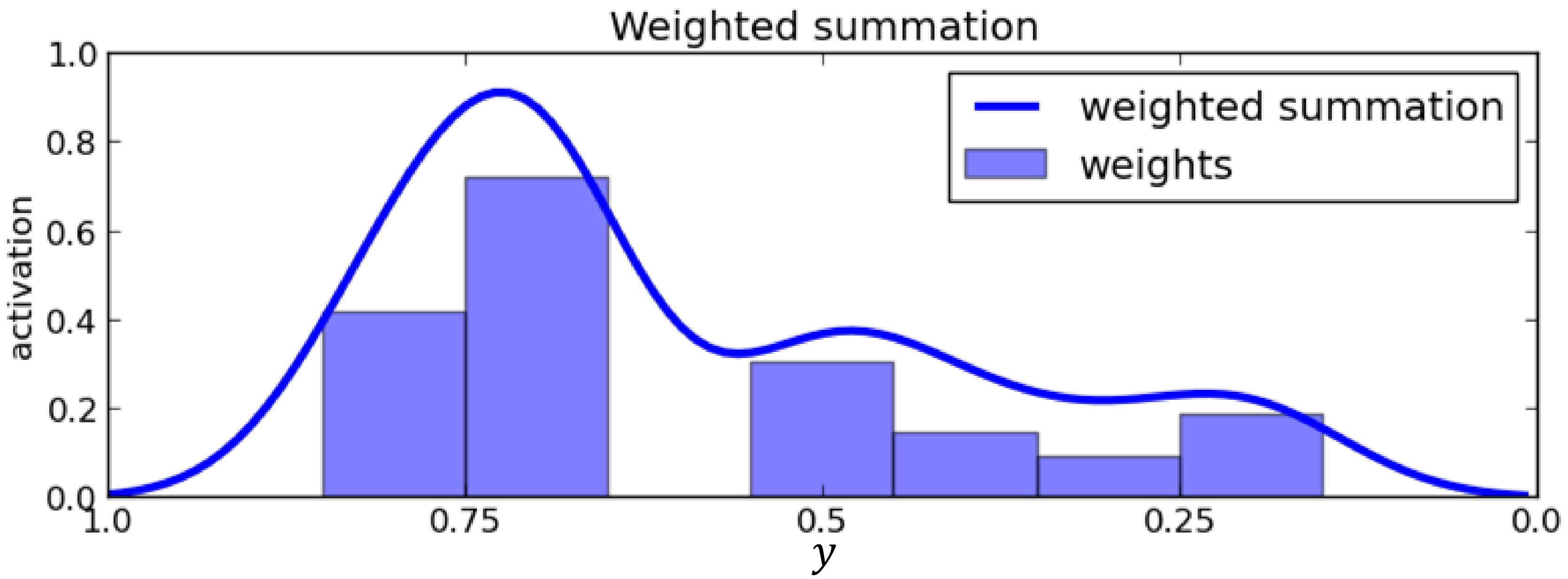
learning rule

$$3)\ \delta w(t) = -e(t).p(t)$$

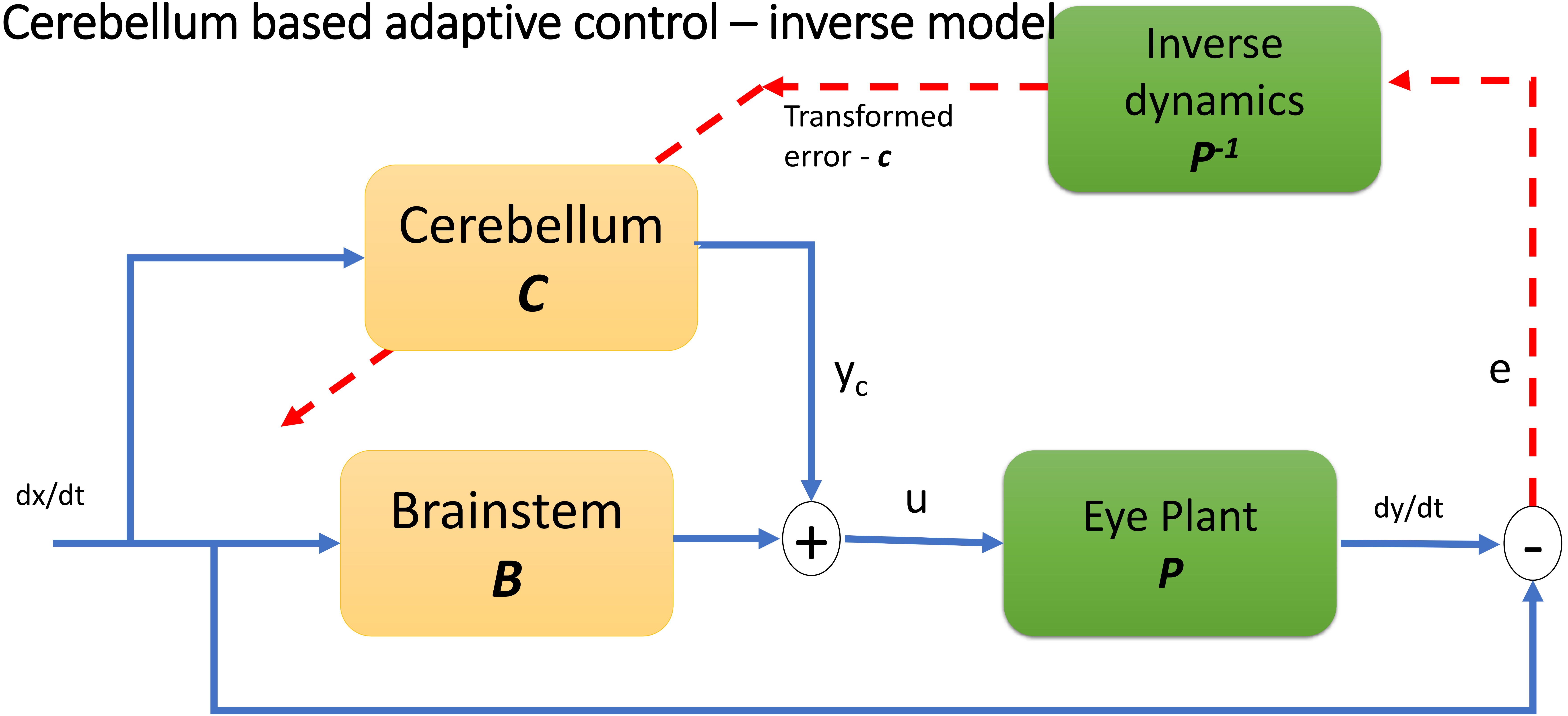
$$p(t) = G * y(t)$$



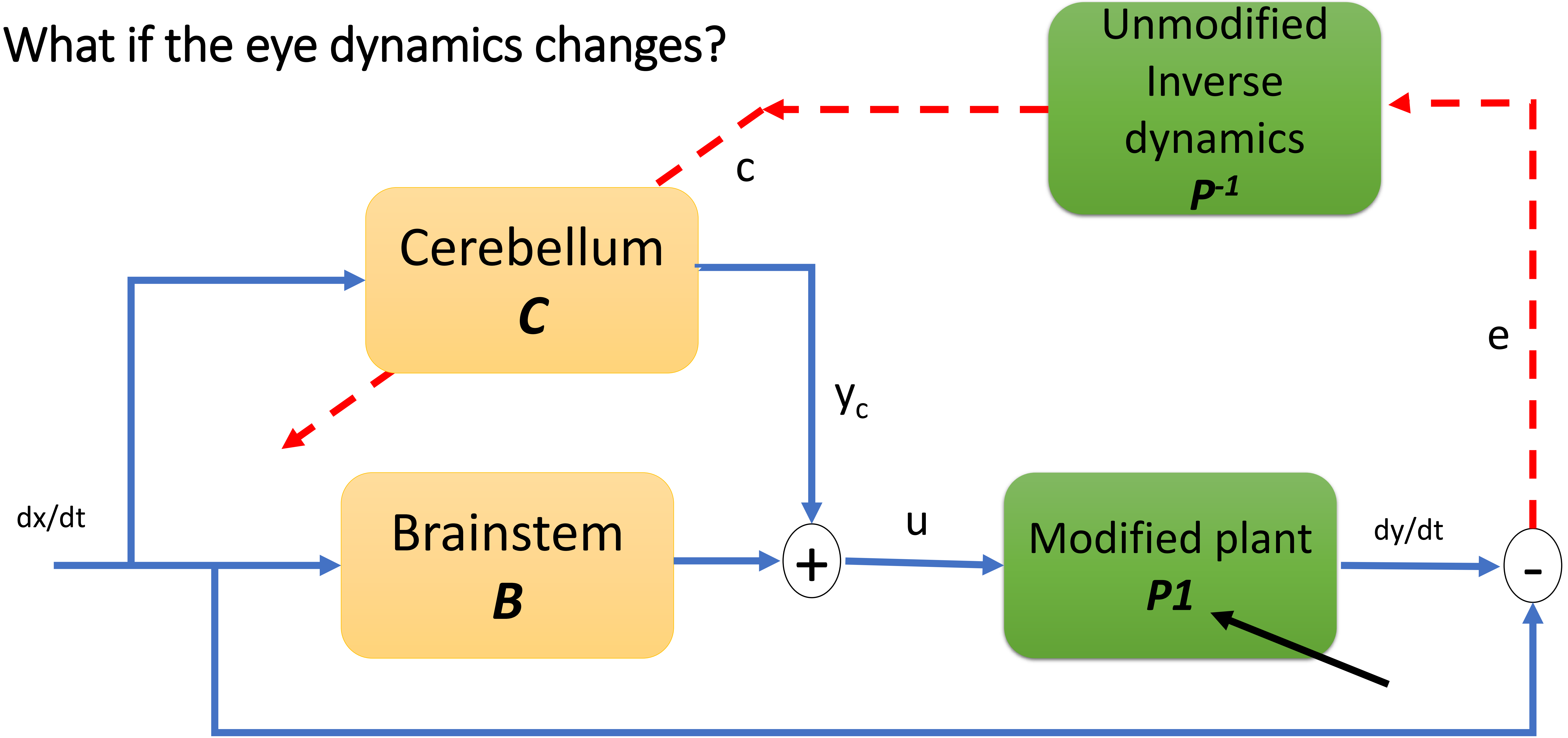
$$z(t) = W * p(t)$$



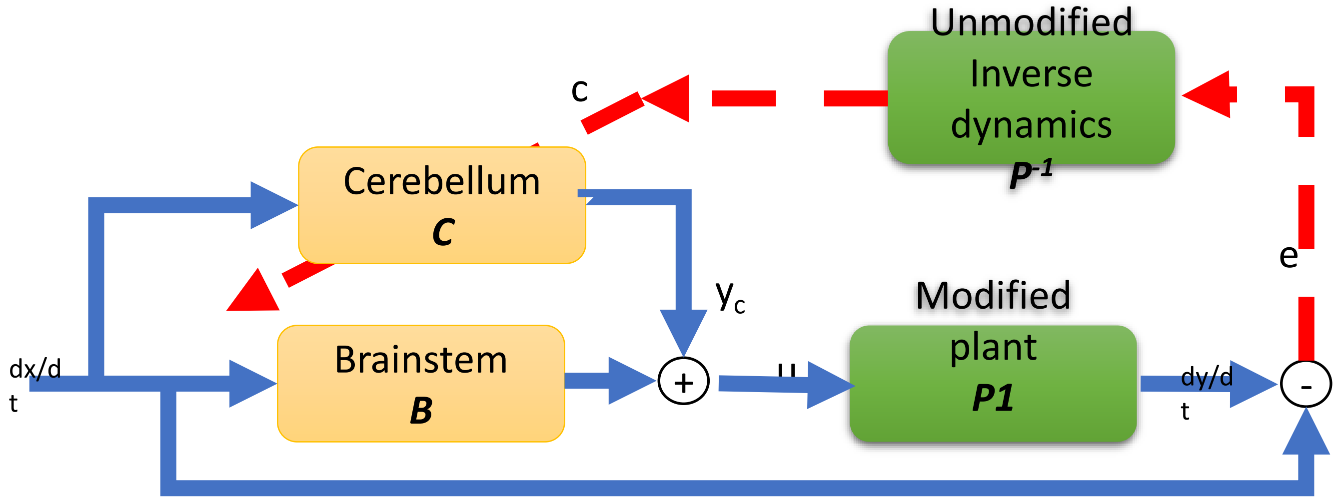
Cerebellum based adaptive control – inverse model

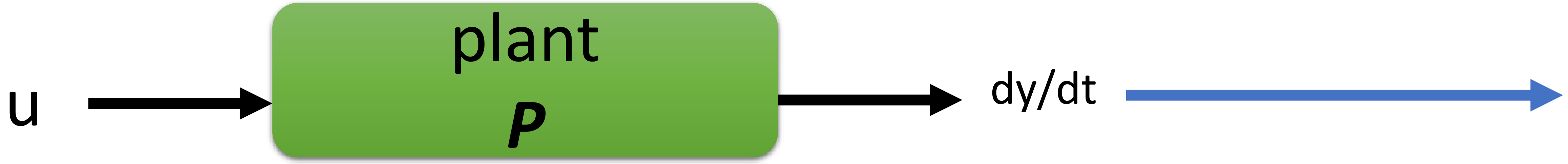


What if the eye dynamics changes?

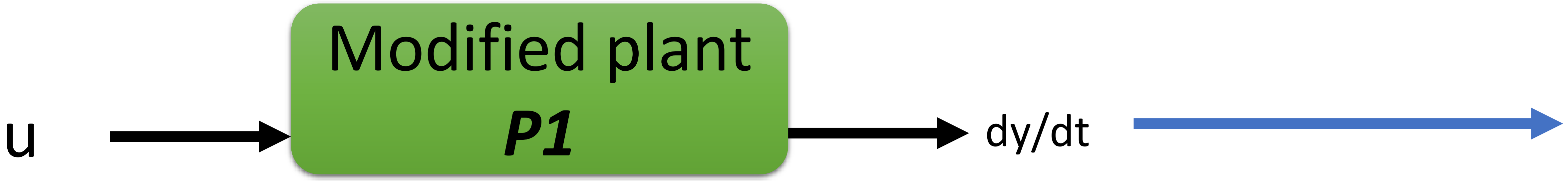


2-degree eye movement (horizontal & vertical)



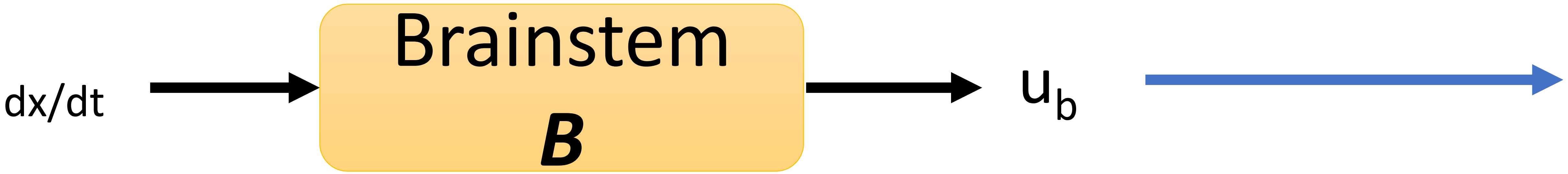


$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



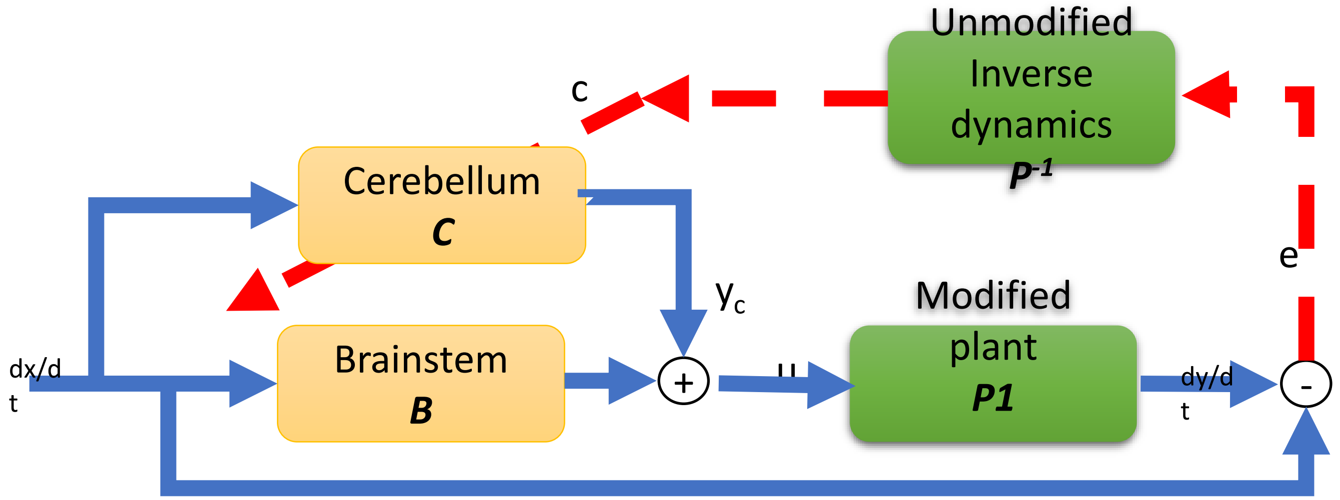
Curl/rotation

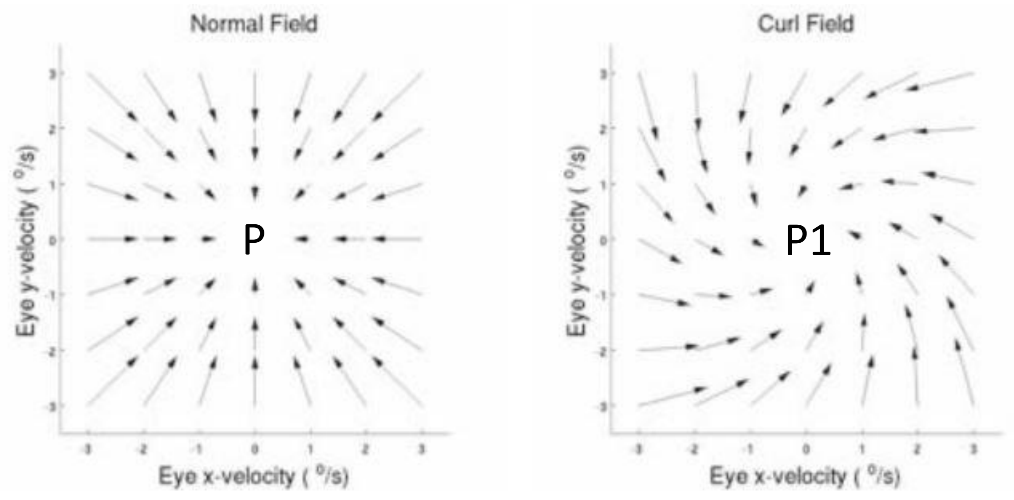
$$P1 = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}$$



$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2-degree eye movement (horizontal & vertical)





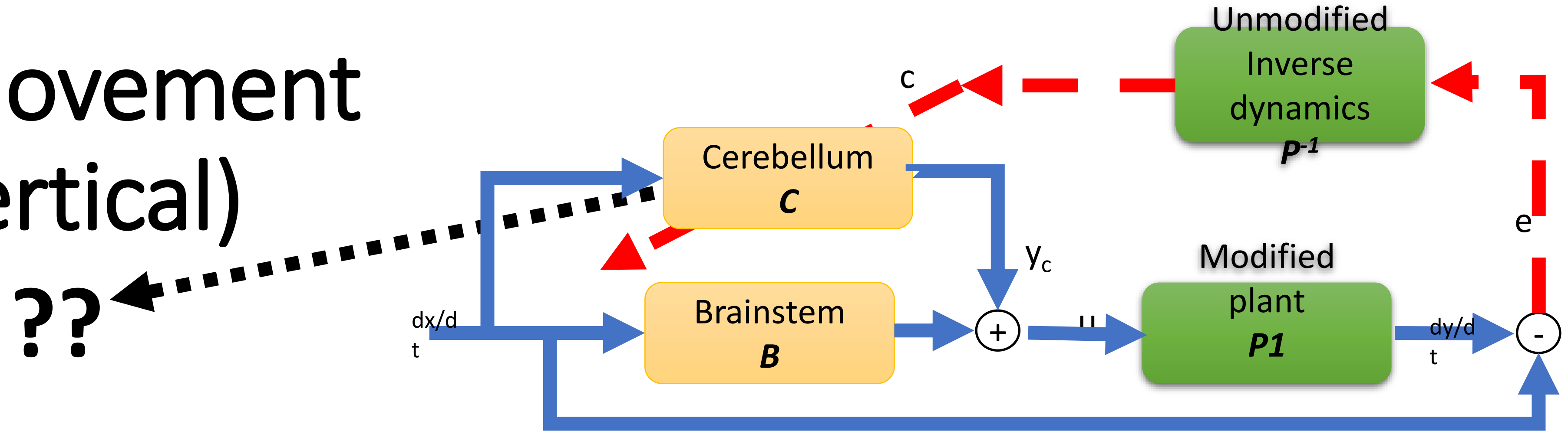
(a) Dynamics transformation

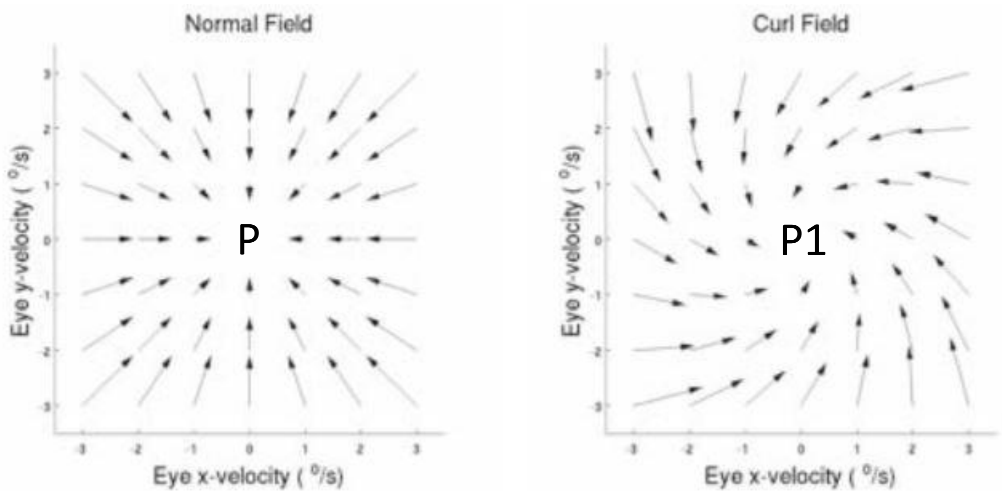
u → plant ***P*** → dy/dt —normal field→ $P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

u → Modified plant ***P1*** → dy/dt —Curl field→ $P1 = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}$

dx/dt → Brainstem ***B*** → u_b → $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

2-degree eye movement (horizontal & vertical)





(a) Dynamics transformation

u → plant *P* → dy/dt →

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

u → Modified plant *P1* → dy/dt →

$$P1 = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}$$

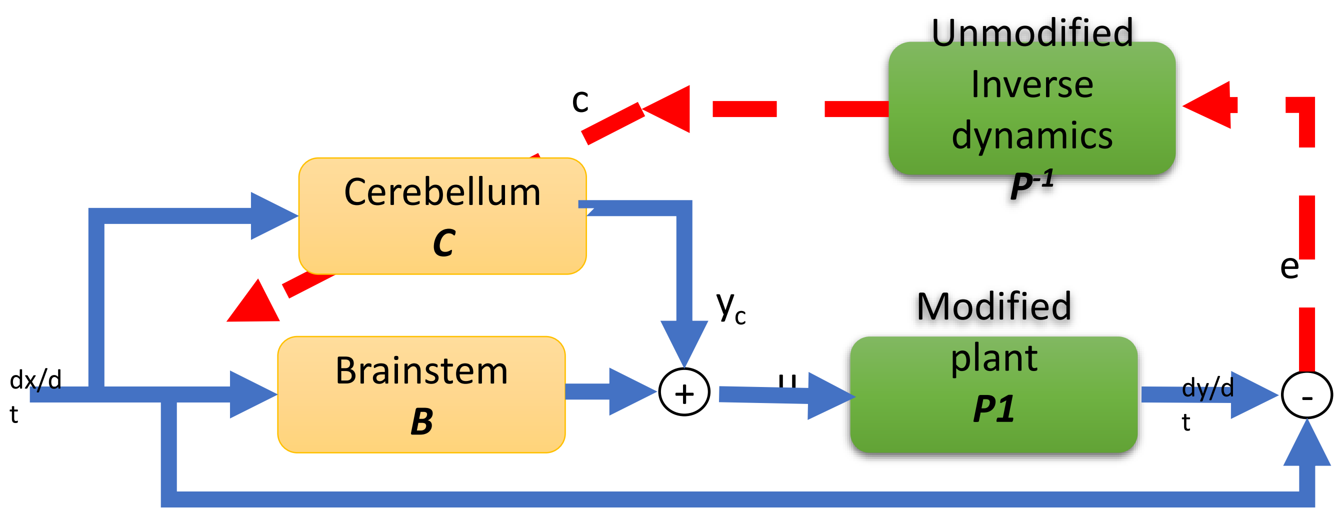
dx/dt → Brainstem *B* → u_b →

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Demonstration

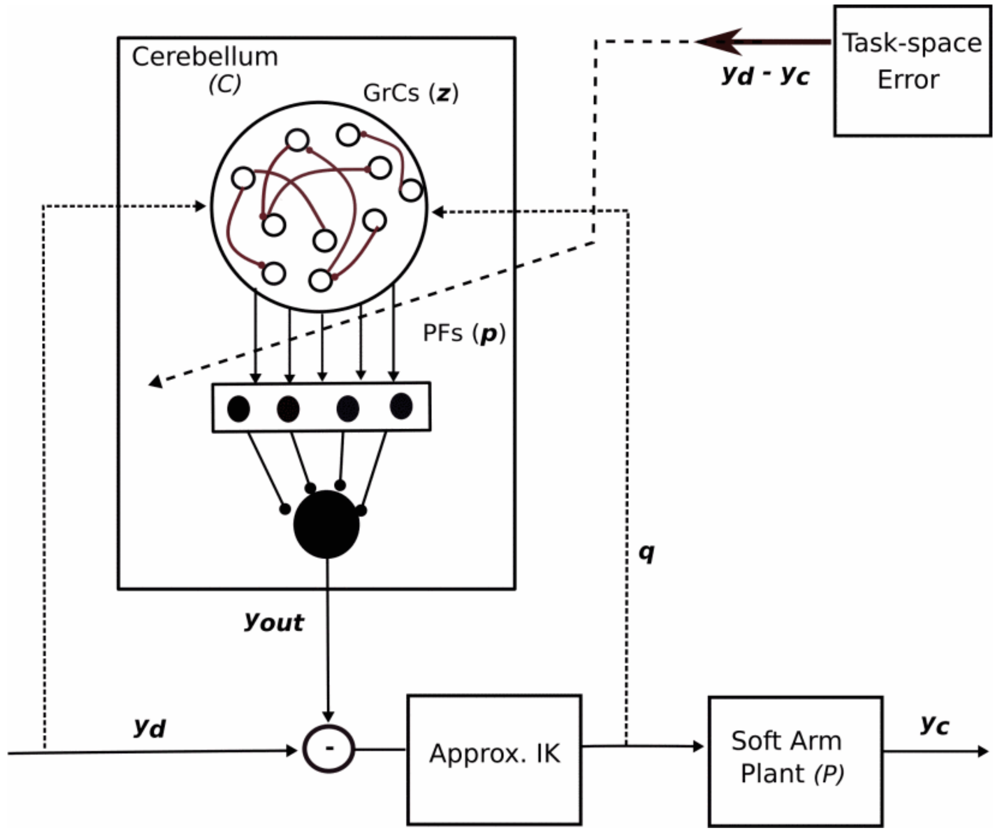
- start a sinusoidal head movement
- Use the adaptive control method to cause compensatory eye movement

algorithm

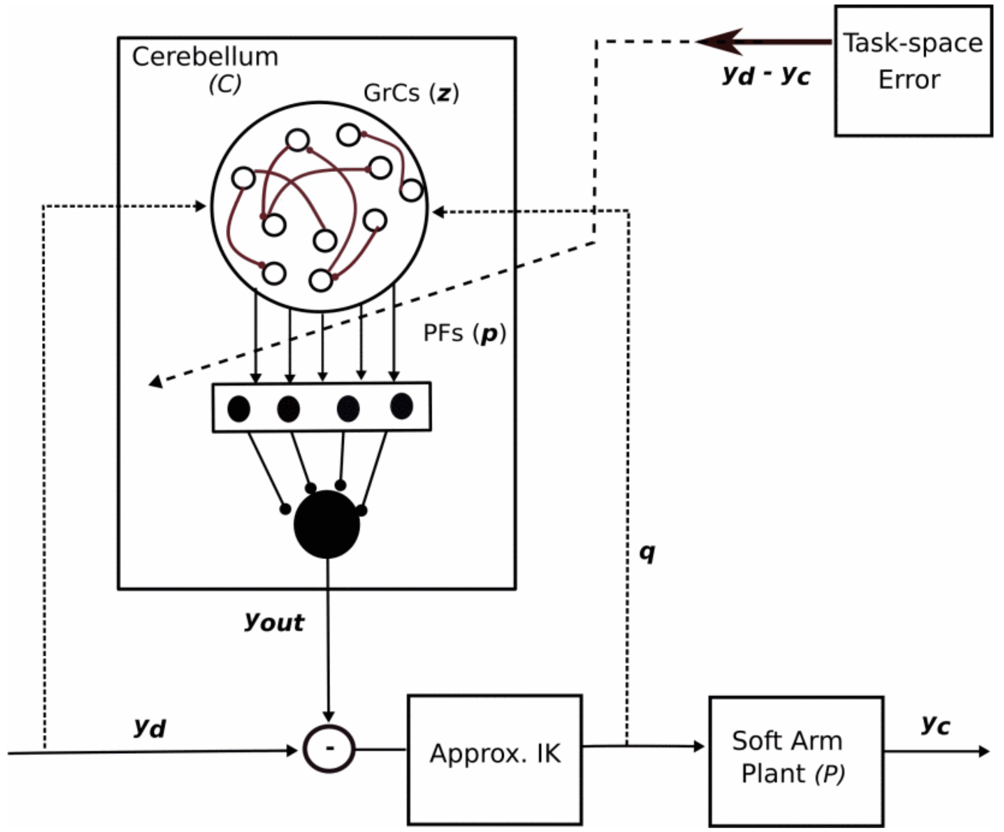


- Initialize cerebellum as an adaptive filter
- Initialize learning rate = lr
- for t = 1:T
    - Sense head velocity: x_dot
    - Compute brain-stem output : u_b
    - Compute cerebellum output : y_c = w*x_dot
    - Total u = u_b + y_c
    - Compute eye velocity : y_dot = P1 * u
    - Compute retinal slip/error: e = x_dot – y_dot
    - Convert retinal slip to cerebellum error : c = P^{-1} * e in general (or) simply c = e for small curl fields
    - Update cerebellum parameters : dw = - lr * c * x_dot

Applications - Soft robot simulation video

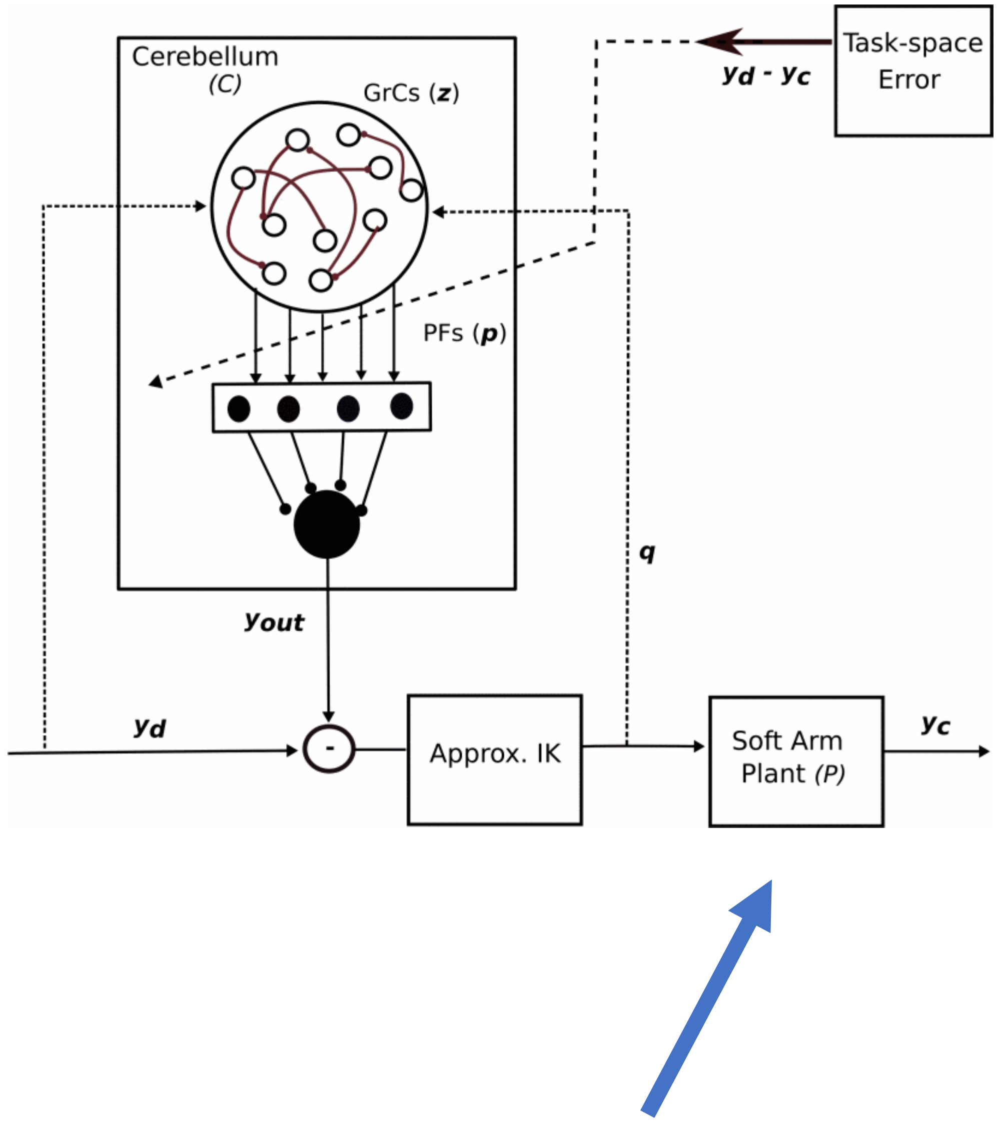


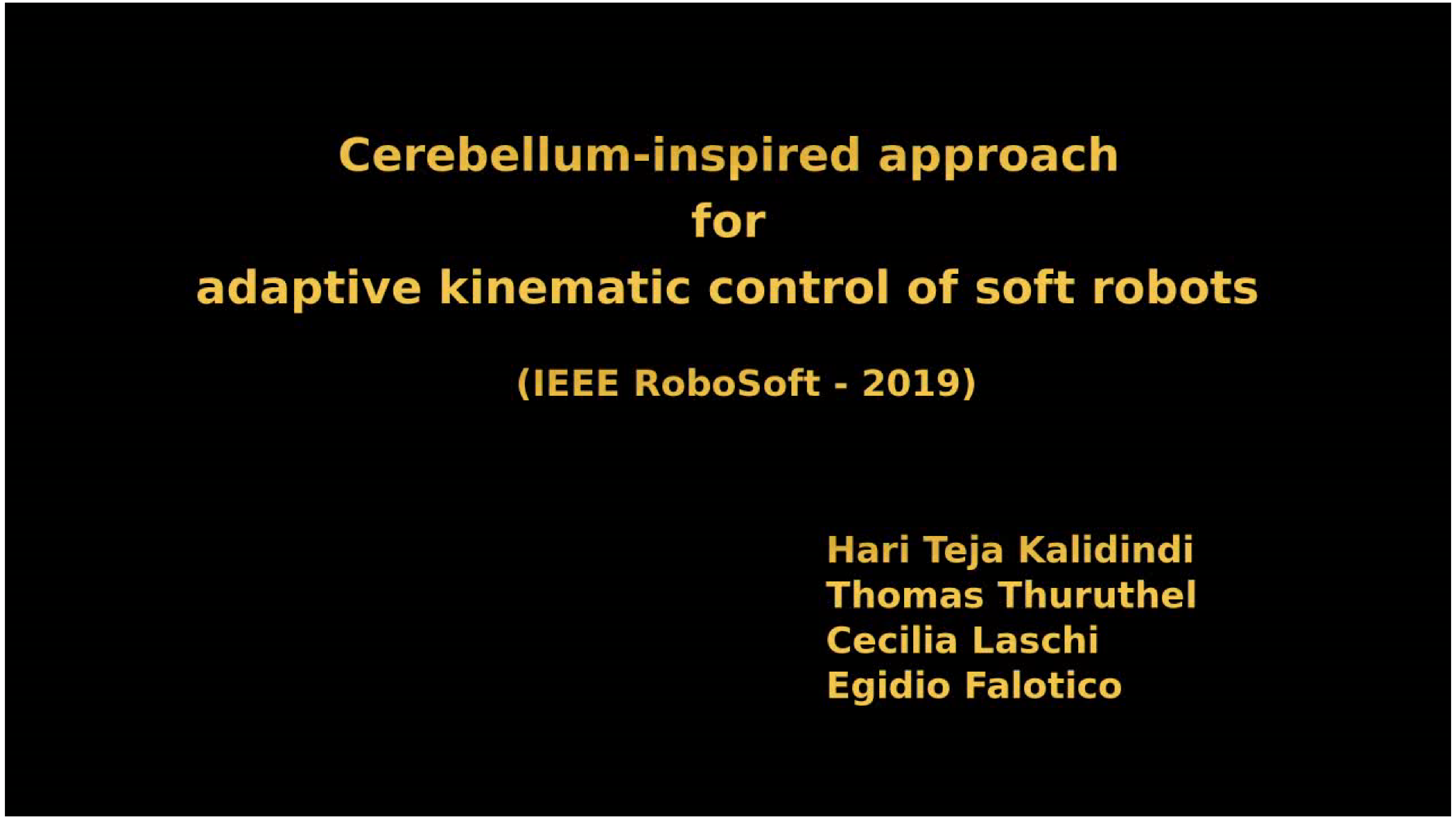
Applications - Soft robot simulation video



Can the cerebellum forward model compensate for changes in the soft arm dynamics?

Applications - Soft robot simulation video





Can the cerebellum forward model compensate for changes in the soft arm dynamics?