

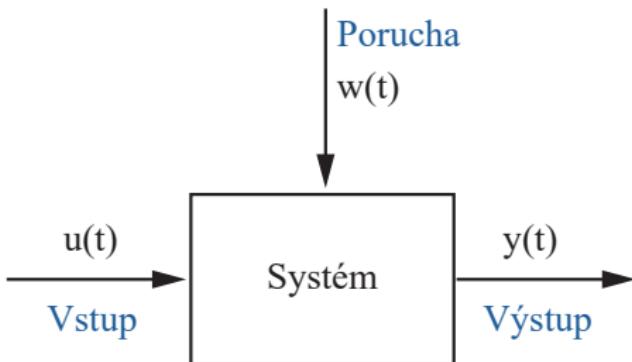
Working title drone control

Riadenie dronov

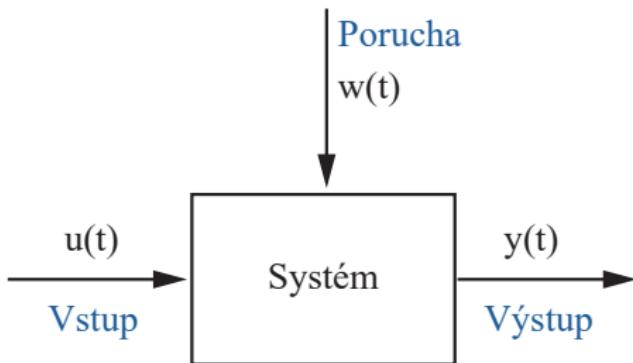
prof. Ing. Gergely Takács, PhD.



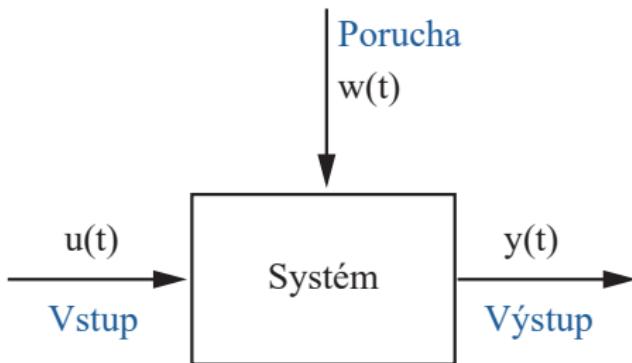
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), napr. motory drona
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie $\theta(t)$ drona
- Porucha (*angl.*: disturbance) $w(t)$ je vplyv vonkajšieho prostredia, napr. vietor



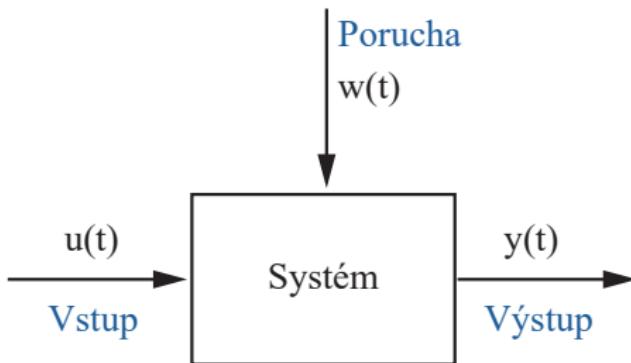
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), napr. motory drona
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie $\theta(t)$ drona
- Porucha (*angl.*: disturbance) $w(t)$ je vplyv vonkajšieho prostredia, napr. vietor



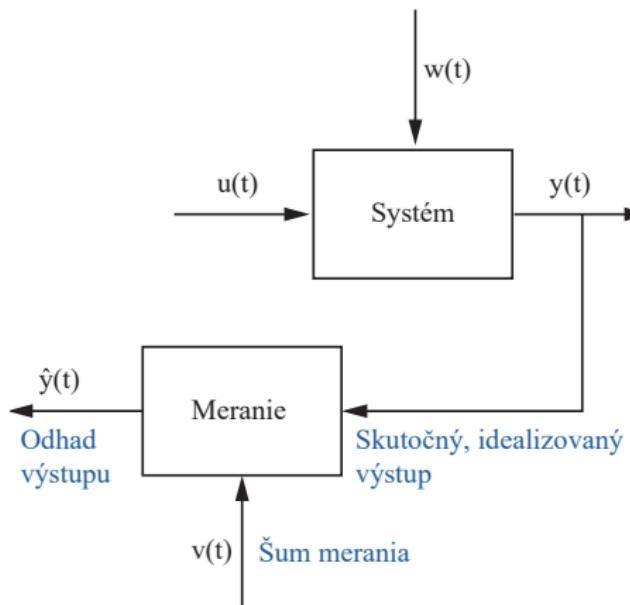
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), napr. motory drona
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie $\theta(t)$ drona
- Porucha (*angl.*: disturbance) $w(t)$ je vplyv vonkajšieho prostredia, napr. vietor



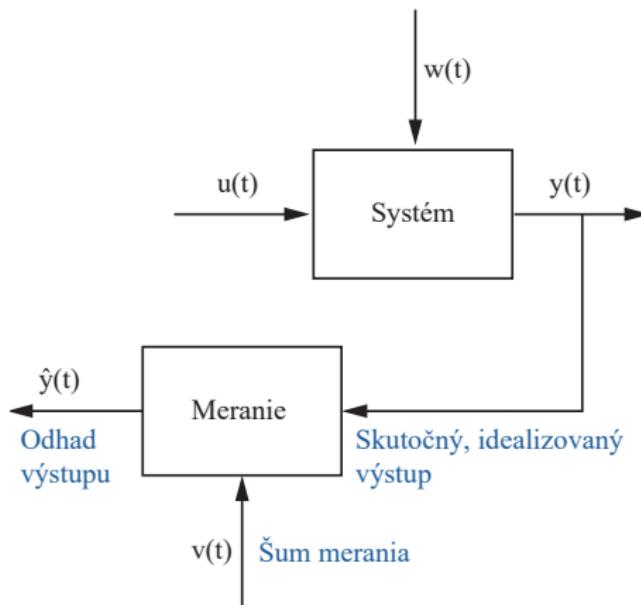
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), napr. motory drona
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie $\theta(t)$ drona
- Porucha (*angl.*: disturbance) $w(t)$ je vplyv vonkajšieho prostredia, napr. vietor



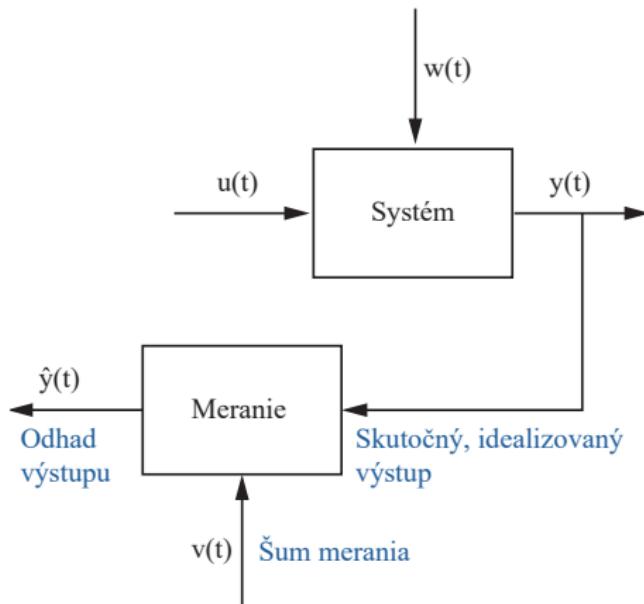
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



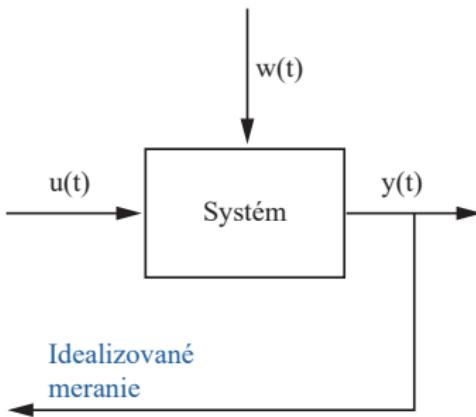
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



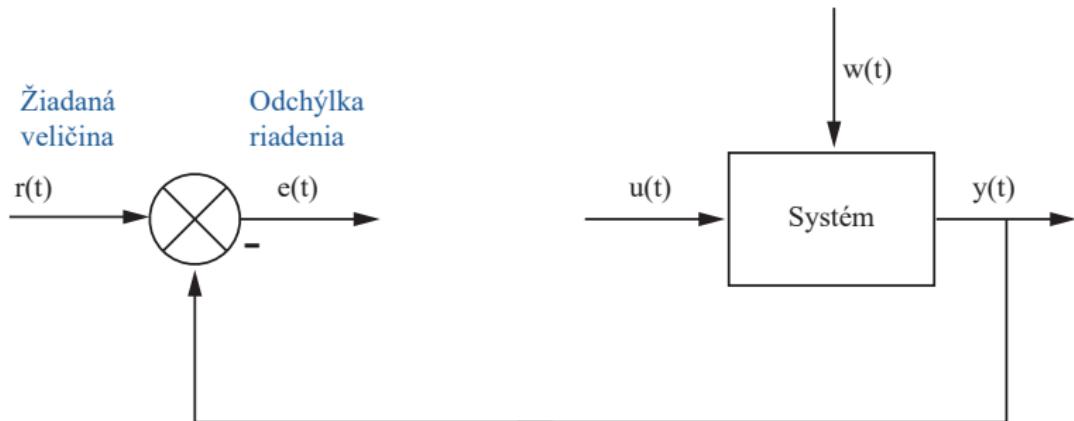
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



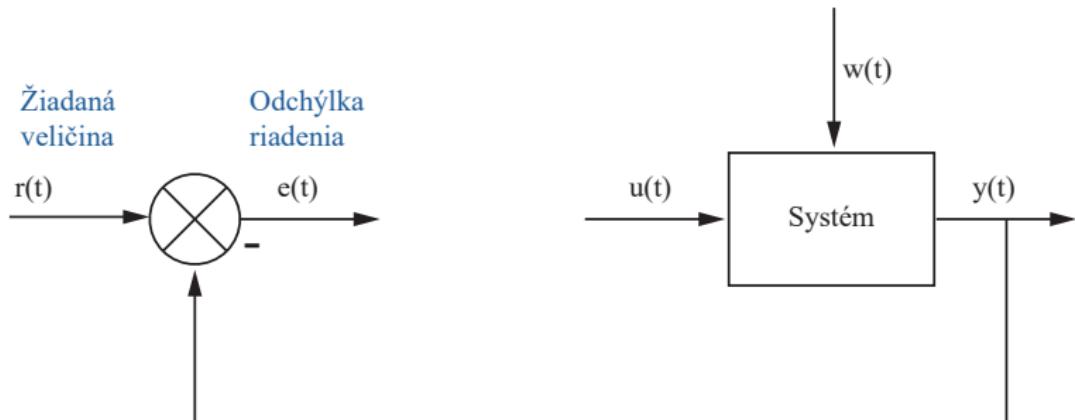
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



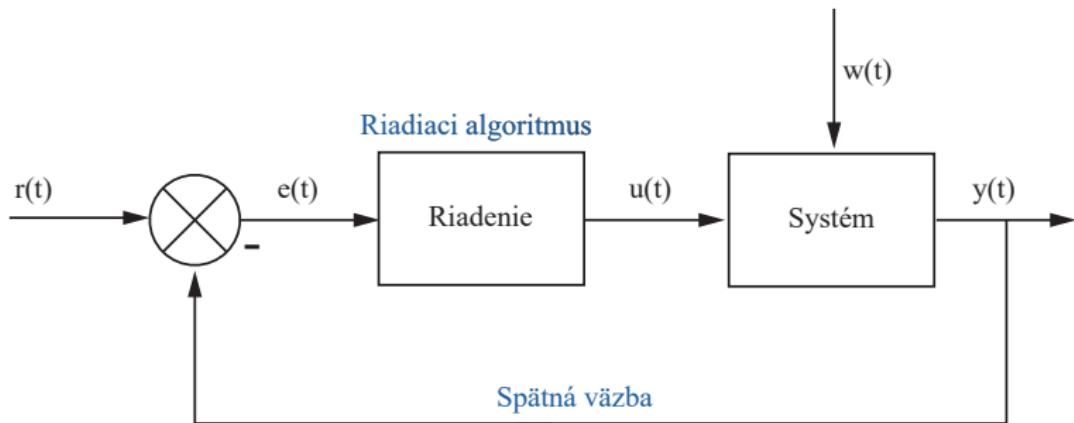
- Žiadaná hodnota alebo referencia (*angl.*: setpoint, reference) $r(t)$ vyjadruje na akú hodnotu chceme dostať výstup, potom
- Rozdiel medzi žiadanou hodnotou a výstupom $e(t) = r(t) - y(t)$ je odchýlka riadenia (*angl.*: error)



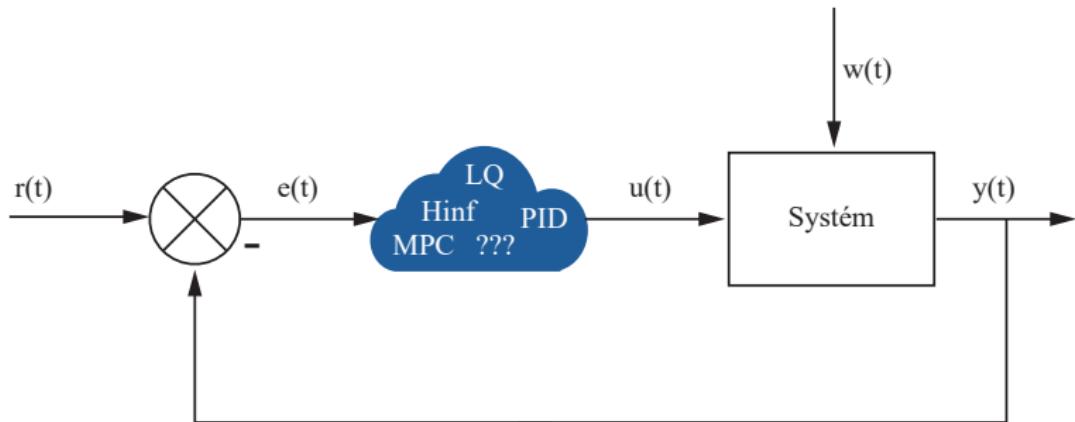
- Žiadaná hodnota alebo referencia (*angl.*: setpoint, reference) $r(t)$ vyjadruje na akú hodnotu chceme dostať výstup, potom
- Rozdiel medzi žiadanou hodnotou a výstupom $e(t) = r(t) - y(t)$ je odchýlka riadenia (*angl.*: error)



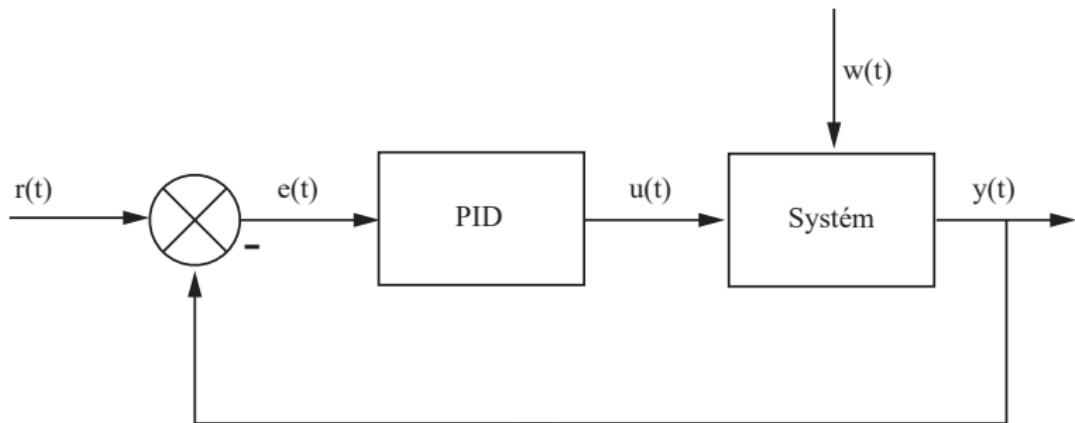
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. parné stroje)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich algoritmov v praxi sú PID... [Åström a kol. 2004]
- ...a väčšina z nich sú slabo naladené.



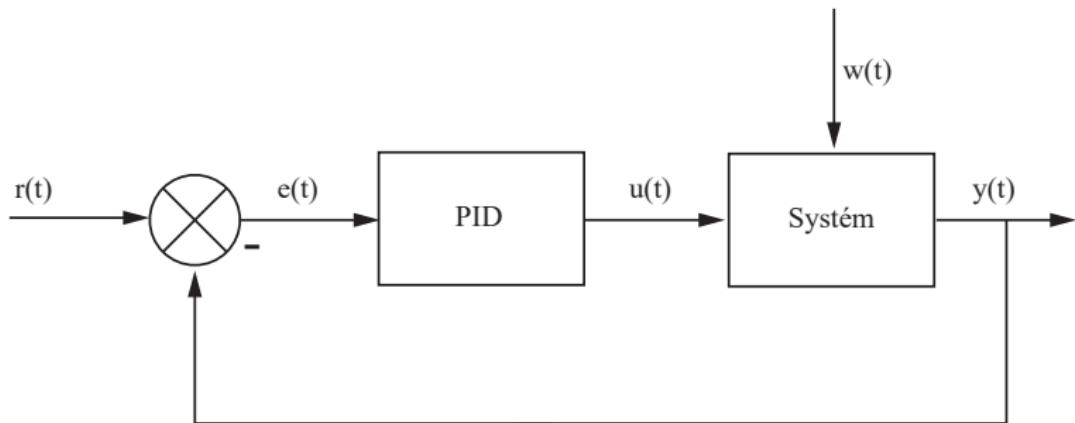
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. parné stroje)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich algoritmov v praxi sú PID... [Åström a kol. 2004]
- ...a väčšina z nich sú slabo naladené.



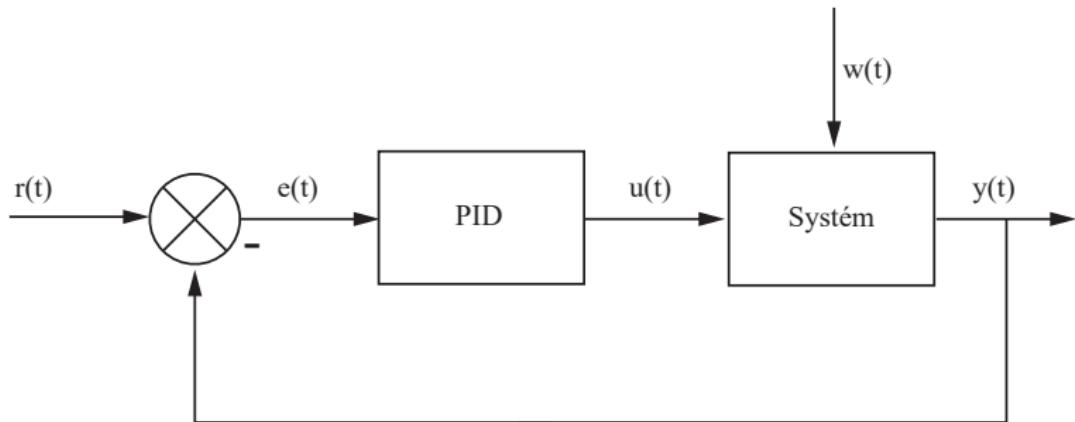
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. parné stroje)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich algoritmov v praxi sú PID... [Åström a kol. 2004]
- ...a väčšina z nich sú slabo naladené.



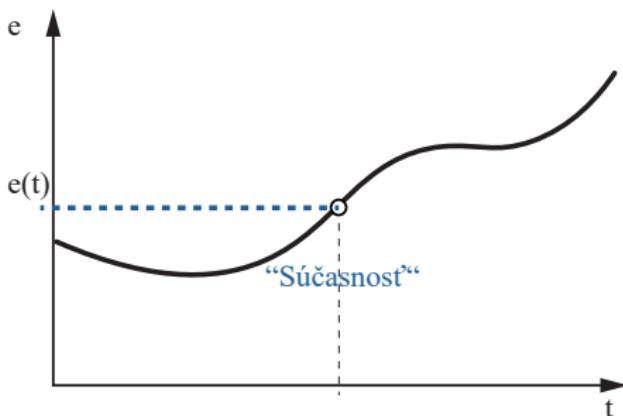
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. parné stroje)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich algoritmov v praxi sú PID... [Åström a kol. 2004]
- ...a väčšina z nich sú slabo naladené.



- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. parné stroje)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich algoritmov v praxi sú PID... [Åström a kol. 2004]
- ...a väčšina z nich sú slabo naladené.

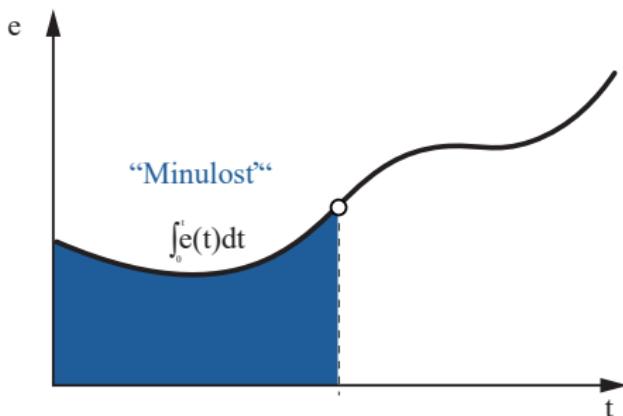


- **Súčasnosť** = proporcionálna zložka: okamžitá odchýlka $e(t)$, ale majme možnosť na ladenie, tak $K_P e(t)$.
- **Minulosť** = integračná zložka: celková odchýlka $e(t)$ v minulosti, ale majme možnosť na ladenie, tak $K_I \int_0^t e(t) dt$.
- **Budúcnosť** = derivačná zložka: projektovaná odchýlka $e(t)$ do budúcnosti, ale majme možnosť na ladenie, tak $K_D \frac{de(t)}{dt}$.



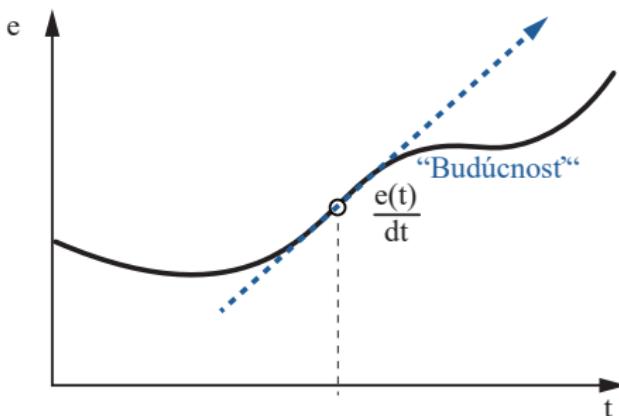
¹V PX4 môžete prepínať pre rate controller PX4 Autopilot 2021b

- **Súčasnosť** = proporcionálna zložka: okamžitá odchýlka $e(t)$, ale majme možnosť na ladenie, tak $K_P e(t)$.
- **Minulosť** = integračná zložka: celková odchýlka $e(t)$ v minulosti, ale majme možnosť na ladenie, tak $K_I \int_0^t e(t)dt$.
- **Budúcnosť** = derivačná zložka: projektovaná odchýlka $e(t)$ do budúcnosti, ale majme možnosť na ladenie, tak $K_D \frac{de(t)}{dt}$.



¹V PX4 môžete prepínať pre rate controller PX4 Autopilot 2021b

- **Súčasnosť** = proporcionálna zložka: okamžitá odchýlka $e(t)$, ale majme možnosť na ladenie, tak $K_P e(t)$.
- **Minulosť** = integračná zložka: celková odchýlka $e(t)$ v minulosti, ale majme možnosť na ladenie, tak $K_I \int_0^t e(t) dt$.
- **Budúcnosť** = derivačná zložka: projektovaná odchýlka $e(t)$ do budúcnosti, ale majme možnosť na ladenie, tak $K_D \frac{de(t)}{dt}$.



¹V PX4 môžete prepínať pre rate controller PX4 Autopilot 2021b

- **Súčasnosť** = proporcionálna zložka: okamžitá odchýlka $e(t)$, ale majme možnosť na ladenie, tak $K_P e(t)$.
- **Minulosť** = integračná zložka: celková odchýlka $e(t)$ v minulosti, ale majme možnosť na ladenie, tak $K_I \int_0^t e(t) dt$.
- **Budúcnosť** = derivačná zložka: projektovaná odchýlka $e(t)$ do budúcnosti, ale majme možnosť na ladenie, tak $K_D \frac{de(t)}{dt}$.

tzv. paralelná (ideálna) forma

$$u(t) = K_P e(t) + K_I \int_0^t e(t) dt + K_D \frac{de(t)}{dt} \quad (1)$$

¹V PX4 môžete prepínať pre rate controller [PX4 Autopilot 2021b](#)

- **Súčasnosť** = proporcionálna zložka: okamžitá odchýlka $e(t)$, ale majme možnosť na ladenie, tak $K_P e(t)$.
- **Minulosť** = integračná zložka: celková odchýlka $e(t)$ v minulosti, ale majme možnosť na ladenie, tak $K_I \int_0^t e(t) dt$.
- **Budúcnosť** = derivačná zložka: projektovaná odchýlka $e(t)$ do budúcnosti, ale majme možnosť na ladenie, tak $K_D \frac{de(t)}{dt}$.
tzv. paralelná (ideálna) forma

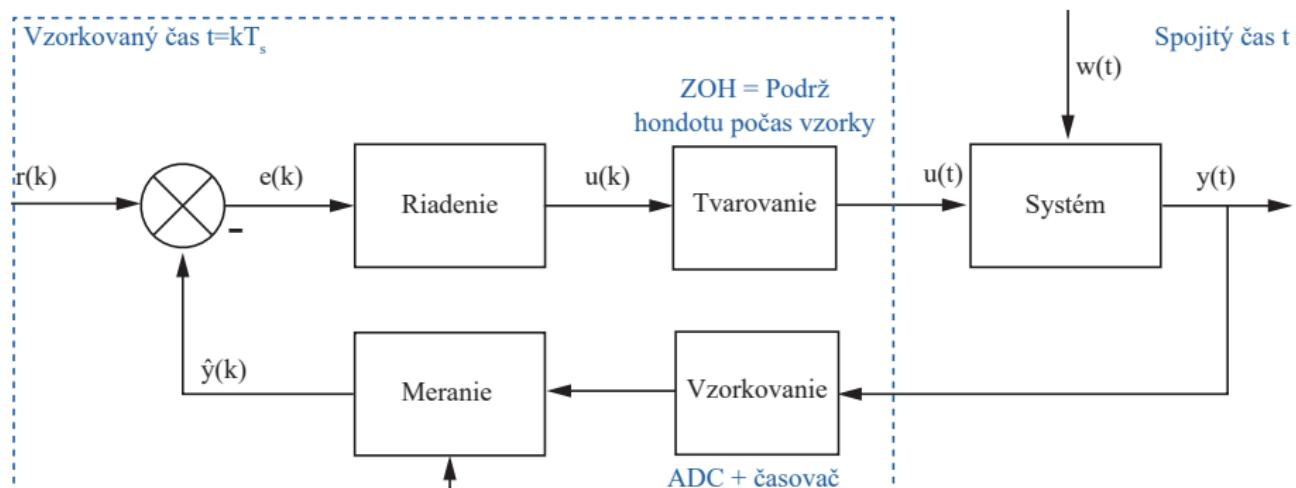
$$u(t) = K_P e(t) + K_I \int_0^t e(t) dt + K_D \frac{de(t)}{dt} \quad (1)$$

alebo tzv. štandardná forma¹ (máme intuitívnejšie časové konštanty, ale zosilnenie je viazané)

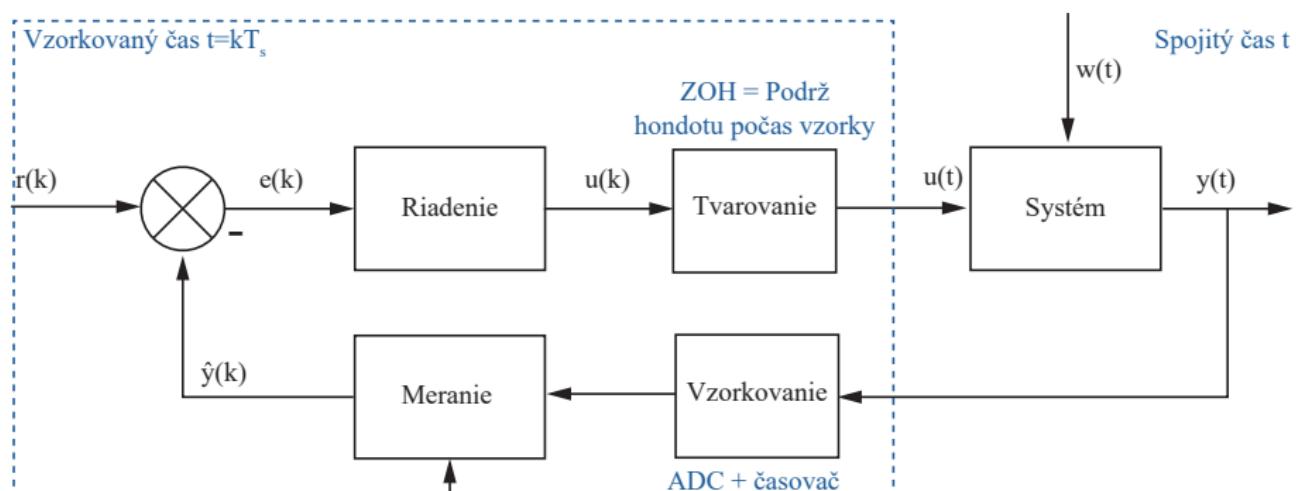
$$u(t) = K_P \left(e(t) + \frac{1}{T_I} \int_0^t e(t) dt + T_D \frac{de(t)}{dt} \right) \quad (2)$$

¹V PX4 môžete prepínať pre rate controller **PX4 Autopilot 2021b**

- Výpočtová realizácia neumožní počítať spojito, preto
 - ▶ Vzorkujeme na strane výstupov — ADC + slučka v hard real-time pomocou časovačov
 - ▶ Tvarujeme na strane vstupov — zero order hold (ZOH) len znamená podržíme hodnotu počas vzorky
- Vzorkovanie mení spojity čas na $t = kT_s$. Periódna je voliteľná, ale
 - ▶ musí zachytiť dominantnú dynamiku riadeného dejha,
 - ▶ a výpočtová realizácia musí stíhať.



- Výpočtová realizácia neumožní počítať spojito, preto
 - ▶ Vzorkujeme na strane výstupov — ADC + slučka v hard real-time pomocou časovačov
 - ▶ Tvarujeme na strane vstupov — zero order hold (ZOH) len znamená podržíme hodnotu počas vzorky
- Vzorkovanie mení spojity čas na $t = kT_s$. Periódna je voliteľná, ale
 - ▶ musí zachytiť dominantnú dynamiku riadeného dejha,
 - ▶ a výpočtová realizácia musí stíhať.



- Systém, resp. proces zostáva väčšinou spojity
- Koncepty ako integrácia, derivácia musíme numericky approximovať, t.j. ak T_s je vzorkovací čas, naše PID bude

$$u(k) = K_P e(k) + K_I \underbrace{\sum_0^t e(k) T_s}_{\text{"obdlzniky"}} + K_D \underbrace{\frac{e(k) - e(k-1)}{T_s}}_{\text{"vstupanie"}} \quad (3)$$

- alebo v prakticky vhodnejšej tzv. inkrementálnej forme pre MCU (bez dôkazu, vid'. Wikipedia 2021)

$$u(k) = u(k-1) + \left(K_P + K_I T_s + \frac{K_D}{T_s} \right) e(k) + \left(-K_P - 2 \frac{K_D}{T_s} \right) e(k-1) + \frac{K_D}{T_s} e(k-2) \quad (4)$$

- Systém, resp. proces zostáva väčšinou spojity
- Koncepty ako integrácia, derivácia musíme numericky approximovať, t.j. ak T_s je vzorkovací čas, naše PID bude

$$u(k) = K_P e(k) + K_I \underbrace{\sum_0^t e(k) T_s}_{\text{"obdlzniky"}} + K_D \underbrace{\frac{e(k) - e(k-1)}{T_s}}_{\text{"vstupanie"}} \quad (3)$$

- alebo v prakticky vhodnejšej tzv. inkrementálnej forme pre MCU (bez dôkazu, vid'. Wikipedia 2021)

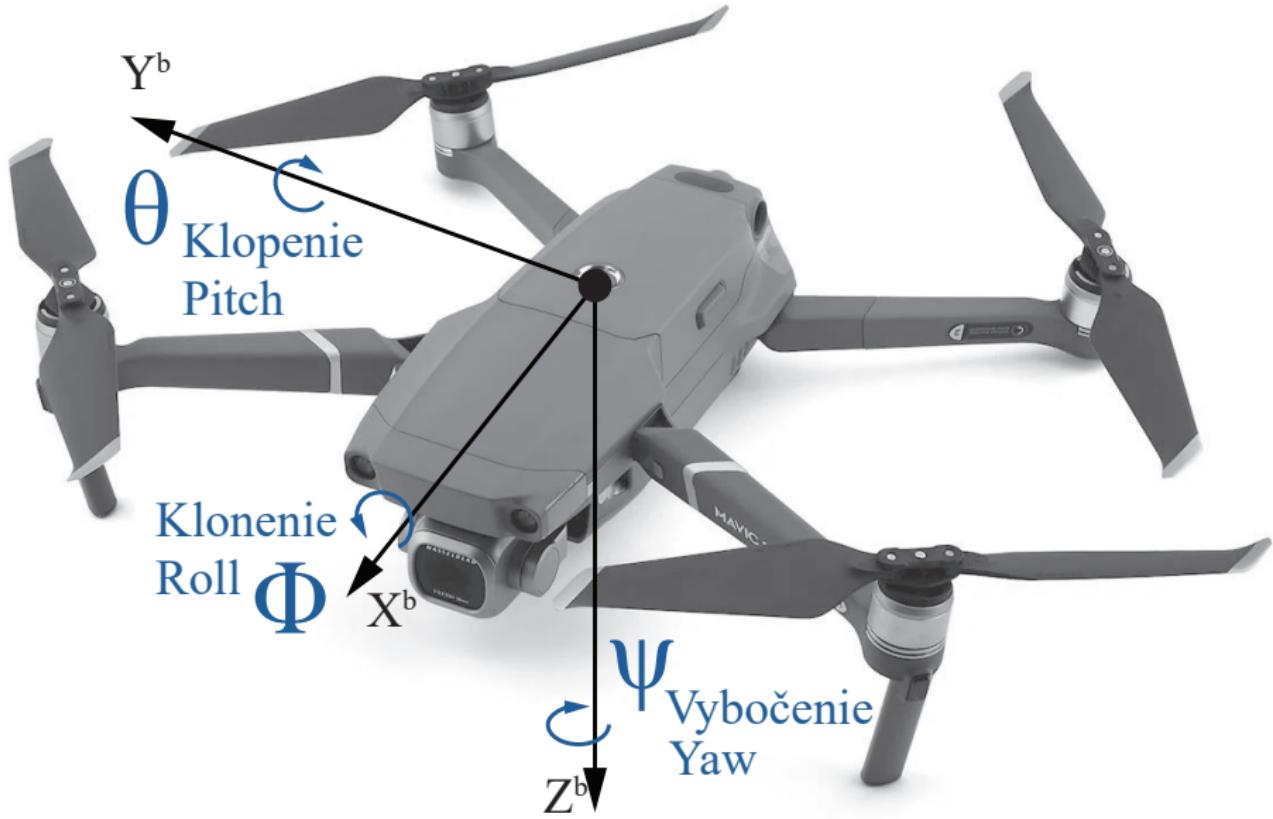
$$u(k) = u(k-1) + \left(K_P + K_I T_s + \frac{K_D}{T_s} \right) e(k) + \left(-K_P - 2 \frac{K_D}{T_s} \right) e(k-1) + \frac{K_D}{T_s} e(k-2) \quad (4)$$

- Systém, resp. proces zostáva väčšinou spojity
- Koncepty ako integrácia, derivácia musíme numericky approximovať, t.j. ak T_s je vzorkovací čas, naše PID bude

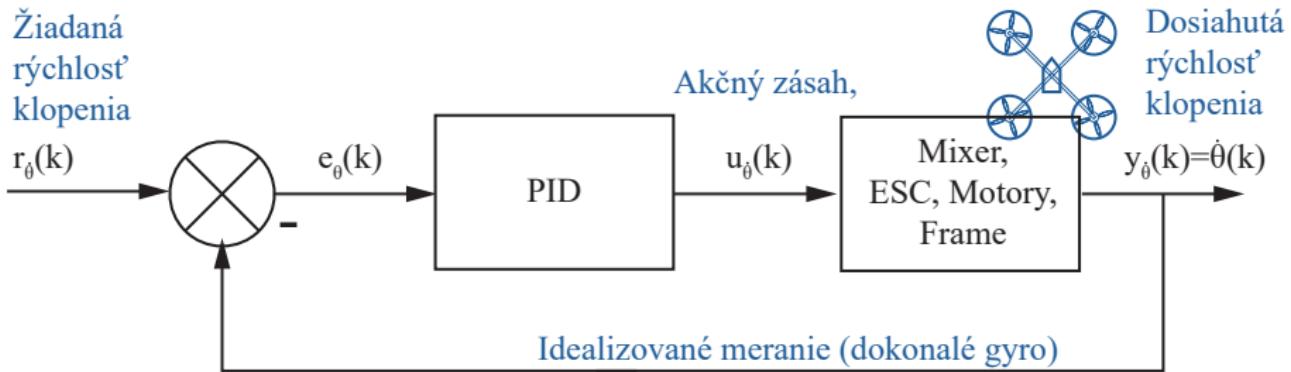
$$u(k) = K_P e(k) + K_I \underbrace{\sum_0^t e(k) T_s}_{\text{"obdlzniky"}} + K_D \underbrace{\frac{e(k) - e(k-1)}{T_s}}_{\text{"vstupanie"}} \quad (3)$$

- alebo v prakticky vhodnejšej tzv. inkrementálnej forme pre MCU (bez dôkazu, vid'. [Wikipedia 2021](#))

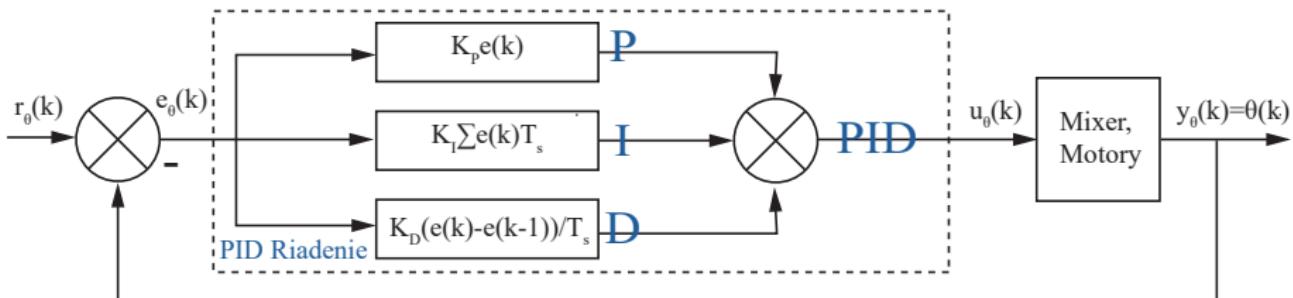
$$u(k) = u(k-1) + \left(K_P + K_I T_s + \frac{K_D}{T_s} \right) e(k) + \left(-K_P - 2 \frac{K_D}{T_s} \right) e(k-1) + \frac{K_D}{T_s} e(k-2) \quad (4)$$



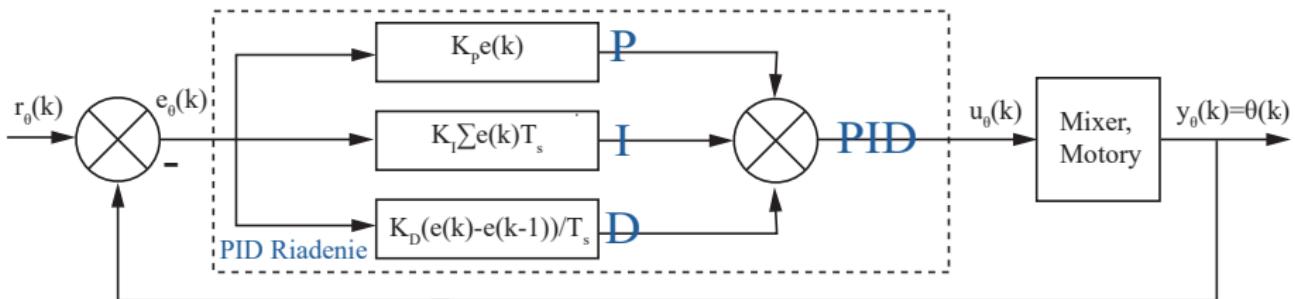
- Tvárame sa, že poznáme žiadanú uhlovú rýchlosť orientácie $r_{\dot{\Theta}}$, $r_{\dot{\phi}}$ a $r_{\dot{\psi}}$ a pre jednoduchosť sústredme len na rýchlosť zmeny klopenia ($\dot{\Theta}$).
- Sme na najnižšej úrovni, t.j. riadenie uhlovej rýchlosť (angl.: rate controller) — je to zároveň aj najdôležitejšia slučka, a máme 3 nezávislých slučiek [Hall 2018; PX4 Autopilot 2021b] .
- Je to aj najrýchlejšia slučka (cca. $f_s=400\text{--}1000\text{ Hz}$ [Hall 2018; PX4 Autopilot 2021b]) Meranie je komplexný problém sám v sebe, napr. low-pass gyro + notch [Bresciani a kol. 2020] alebo odhad



- Tvárame sa, že poznáme žiadanú uhlovú rýchlosť orientácie $r_{\dot{\Theta}}$, $r_{\dot{\phi}}$ a $r_{\dot{\psi}}$ a pre jednoduchosť sústredme len na rýchlosť zmeny klopenia ($\dot{\Theta}$).
- Sme na najnižšej úrovni, t.j. riadenie uhlovej rýchlosť (*angl.*: rate controller) — je to zároveň aj najdôležitejšia slučka, a máme 3 nezávislých slučiek [Hall 2018; PX4 Autopilot 2021b].
- Je to aj najrýchlejšia slučka (cca. $f_s=400\text{--}1000$ Hz [Hall 2018; PX4 Autopilot 2021b]) Meranie je komplexný problém sám v sebe, napr. low-pass gyro + notch [Bresciani a kol. 2020] alebo odhad



- Tvárame sa, že poznáme žiadanú uhlovú rýchlosť orientácie $r_{\dot{\Theta}}$, $r_{\dot{\phi}}$ a $r_{\dot{\psi}}$ a pre jednoduchosť sústredme len na rýchlosť zmeny klopenia ($\dot{\Theta}$).
- Sme na najnižšej úrovni, t.j. riadenie uhlovej rýchlosť (*angl.*: rate controller) — je to zároveň aj najdôležitejšia slučka, a máme 3 nezávislých slučiek [Hall 2018; PX4 Autopilot 2021b].
- Je to aj najrýchlejšia slučka (cca. $f_s=400\text{--}1000$ Hz [Hall 2018; PX4 Autopilot 2021b]) Meranie je komplexný problém sám v sebe, napr. low-pass gyro + notch [Bresciani a kol. 2020] alebo odhad



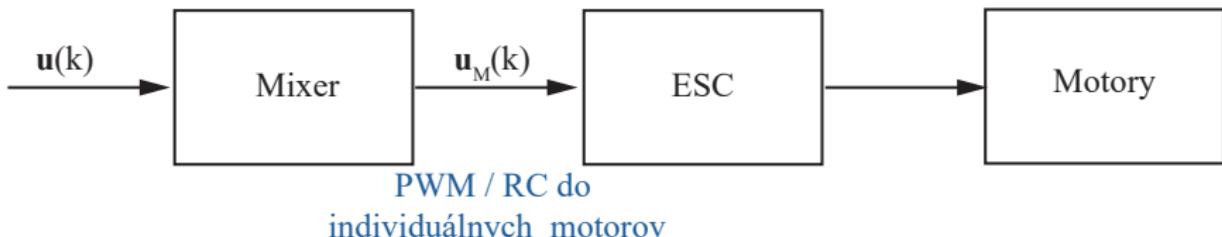
- Aj keď slučky PID na tejto úrovni definujú žiadanú zmenu $u(k)$, musíme implementovať pridelenie riadenia (*angl.*: control allocation) $u_M(k)$ (PWM, RC impulzy) do individuálnych motorov cez elektronické riadenie rýchlosťi (*angl.*: electronic speed control, ESC).
- V najjednoduchšom prípade máme mapujeme pre každý motor 1-4 (alebo viac)

$$u_{M1}(k) = u_\tau - u_\phi - u_\theta - u_\Sigma \quad (5)$$

$$u_{M2}(k) = u_\tau + u_\phi + u_\theta - u_\Sigma \quad (6)$$

$$u_{M3}(k) = u_\tau - u_\phi + u_\theta + u_\Sigma \quad (7)$$

$$u_{M4}(k) = u_\tau + u_\phi - u_\theta + u_\Sigma \quad (8)$$



- Aj keď slučky PID na tejto úrovni definujú žiadanú zmenu $u(k)$, musíme implementovať pridelenie riadenia (*angl.*: control allocation) $u_M(k)$ (PWM, RC impulzy) do individuálnych motorov cez elektronické riadenie rýchlosťi (*angl.*: electronic speed control, ESC).
- V najjednoduchšom prípade máme mapujeme pre každý motor 1-4 (alebo viac)

$$u_{M1}(k) = u_\tau - u_\phi - u_\theta - u_\Sigma \quad (5)$$

$$u_{M2}(k) = u_\tau + u_\phi + u_\theta - u_\Sigma \quad (6)$$

$$u_{M3}(k) = u_\tau - u_\phi + u_\theta + u_\Sigma \quad (7)$$

$$u_{M4}(k) = u_\tau + u_\phi - u_\theta + u_\Sigma \quad (8)$$



$$\begin{bmatrix} u_{M1}(k) \\ u_{M2}(k) \\ u_{M3}(k) \\ u_{M4}(k) \end{bmatrix} = \underbrace{\begin{bmatrix} +1 & -1 & -1 & -1 \\ +1 & +1 & +1 & -1 \\ +1 & -1 & +1 & +1 \\ +1 & +1 & -1 & +1 \end{bmatrix}}_P \begin{bmatrix} u_\tau \\ u_\phi \\ u_\dot{\theta} \\ u_\dot{\Sigma} \end{bmatrix} \quad (9)$$

- Mapa $u_M(k) = Pu(k)$ kde P matica pridelenia riadenia (*angl.: control allocation matrix*). Nemusí obsahovať iba jednotky, môže byť aj neštvorcová ale našťastie transformácia je stále lineárna [Bresciani a kol. 2020].
- Môžeme priamo namapovať maticu efektivity akčných členov (*angl.: actuator effectiveness matrix*) B , kde $P = B^{-1}$. Ak počet akčných zásahov = počtu akčných členov, používame inverziu ináč napr. Moore-Penroseovu pseudoinverziu $P = B^\dagger$ [Bresciani a kol. 2020].



$$\begin{bmatrix} u_{M1}(k) \\ u_{M2}(k) \\ u_{M3}(k) \\ u_{M4}(k) \end{bmatrix} = \underbrace{\begin{bmatrix} +1 & -1 & -1 & -1 \\ +1 & +1 & +1 & -1 \\ +1 & -1 & +1 & +1 \\ +1 & +1 & -1 & +1 \end{bmatrix}}_P \begin{bmatrix} u_\tau \\ u_\phi \\ u_\dot{\theta} \\ u_\dot{\Sigma} \end{bmatrix} \quad (9)$$

- Mapa $u_M(k) = Pu(k)$ kde P matica pridelenia riadenia (*angl.: control allocation matrix*). Nemusí obsahovať iba jednotky, môže byť aj neštvorcová ale našťastie transformácia je stále lineárna [Bresciani a kol. 2020].
- Môžeme priamo namapovať maticu efektivity akčných členov (*angl.: actuator effectiveness matrix*) B , kde $P = B^{-1}$. Ak počet akčných zásahov = počtu akčných členov, používame inverziu ináč napr. Moore-Penroseovu pseudoinverziu $P = B^\dagger$ [Bresciani a kol. 2020].

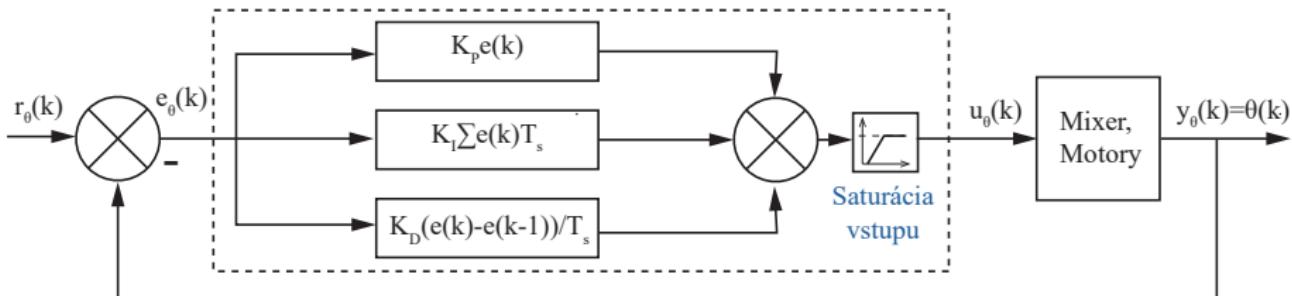


$$\begin{bmatrix} u_{M1}(k) \\ u_{M2}(k) \\ u_{M3}(k) \\ u_{M4}(k) \end{bmatrix} = \underbrace{\begin{bmatrix} +1 & -1 & -1 & -1 \\ +1 & +1 & +1 & -1 \\ +1 & -1 & +1 & +1 \\ +1 & +1 & -1 & +1 \end{bmatrix}}_P \begin{bmatrix} u_\tau \\ u_\phi \\ u_\dot{\theta} \\ u_\dot{\Sigma} \end{bmatrix} \quad (9)$$

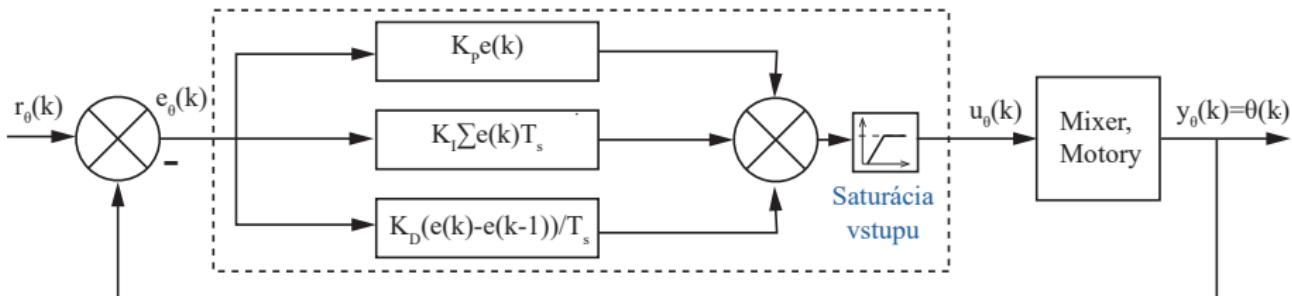
- Mapa $u_M(k) = Pu(k)$ kde P matica pridelenia riadenia (*angl.: control allocation matrix*). Nemusí obsahovať iba jednotky, môže byť aj neštvorcová ale našťastie transformácia je stále lineárna [Bresciani a kol. 2020].
- Môžeme priamo namapovať maticu efektivity akčných členov (*angl.: actuator effectiveness matrix*) B , kde $P = B^{-1}$. Ak počet akčných zásahov = počtu akčných členov, používame inverziu ináč napr. Moore-Penroseovu pseudoinverziu $P = B^\dagger$ [Bresciani a kol. 2020].

- Akčné zásahy majú svoje ohraničenia ((angl.: constraints))
- Ako donútime ich dodržanie? Saturáciou (orezávaním) hodnôt, ktoré skutočne vypočíta PID.
- Saturácia vnáša nelinearitu, vplýva na výkon riadenia aj stabilitu.
- Aj iné veličiny môžeme saturovať, napr. žiadane hodnoty. Ako by sme ohraničili výstup?

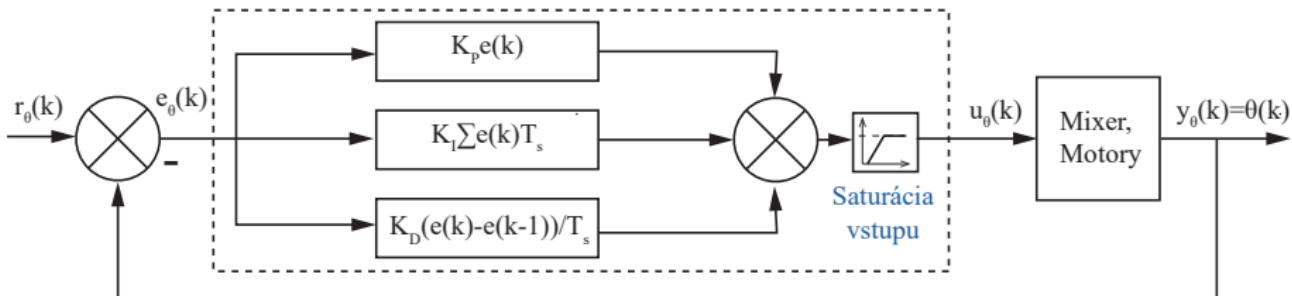
- Akčné zásahy majú svoje ohraničenia ((angl.: constraints))
- Ako donútime ich dodržanie? Saturáciou (orezávaním) hodnôt, ktoré skutočne vypočítava PID.
- Saturácia vnáša nelinearitu, vplýva na výkon riadenia aj stabilitu.
- Aj iné veličiny môžeme saturovať, napr. žiadane hodnoty. Ako by sme ohraničili výstup?



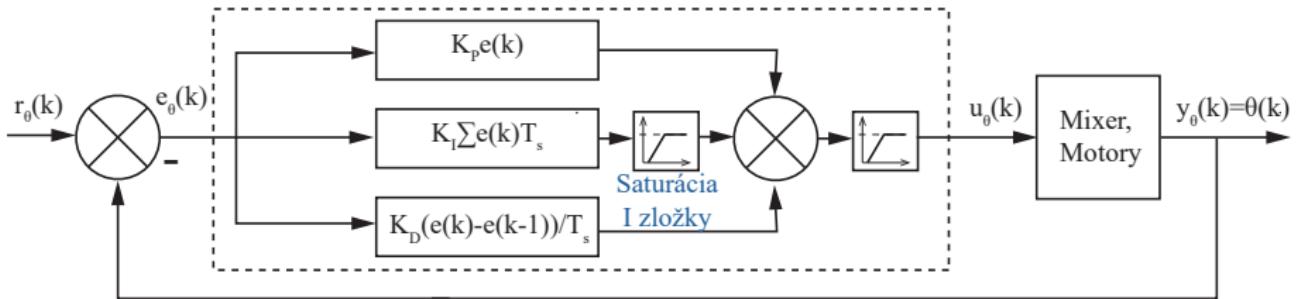
- Akčné zásahy majú svoje ohraničenia ((angl.: constraints))
- Ako donútime ich dodržanie? Saturáciou (orezávaním) hodnôt, ktoré skutočne vypočítava PID.
- Saturácia vnáša nelinearitu, vplýva na výkon riadenia aj stabilitu.
- Aj iné veličiny môžeme saturovať, napr. žiadane hodnoty. Ako by sme ohraničili výstup?



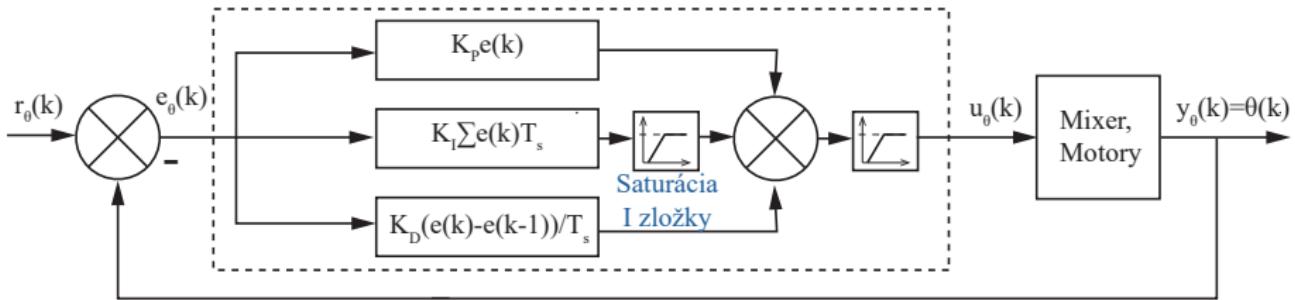
- Akčné zásahy majú svoje ohraničenia ((angl.: constraints))
- Ako donútime ich dodržanie? Saturáciou (orezávaním) hodnôt, ktoré skutočne vypočítava PID.
- Saturácia vnáša nelinearitu, vplýva na výkon riadenia aj stabilitu.
- Aj iné veličiny môžeme saturovať, napr. žiadane hodnoty. Ako by sme ohraničili výstup?



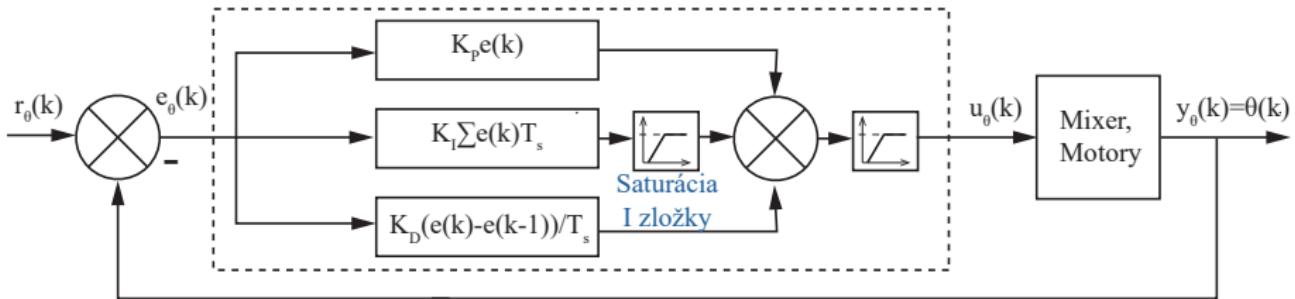
- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadovať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind) a tým pádom prestrelíme (*angl.*: overshoot) žiadané hodnoty
- To je saturácia integračnej zložky (*angl.*: integral windup).
- Môžeme používať rôzne triky, napr. ohraničiť veľkosť integračnej zložky, resp. vypnúť zložku pri určitých podmienkach.



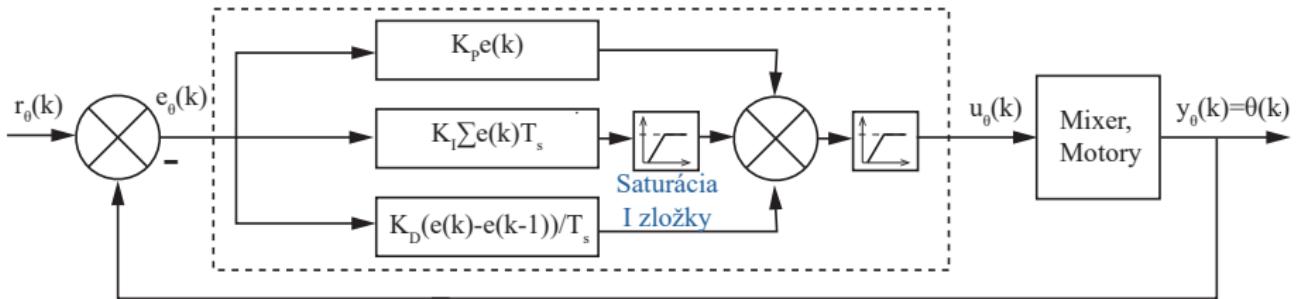
- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind) a tým pádom prestrelíme (*angl.*: overshoot) žiadané hodnoty
- To je saturácia integračnej zložky (*angl.*: integral windup).
- Môžeme používať rôzne triky, napr. ohraničiť veľkosť integračnej zložky, resp. vypnúť zložku pri určitých podmienkach.



- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind) a tým pádom prestrelíme (*angl.*: overshoot) žiadané hodnoty
- To je saturácia integračnej zložky (*angl.*: integral windup).
- Môžeme používať rôzne triky, napr. ohraničiť veľkosť integračnej zložky, resp. vypnúť zložku pri určitých podmienkach.

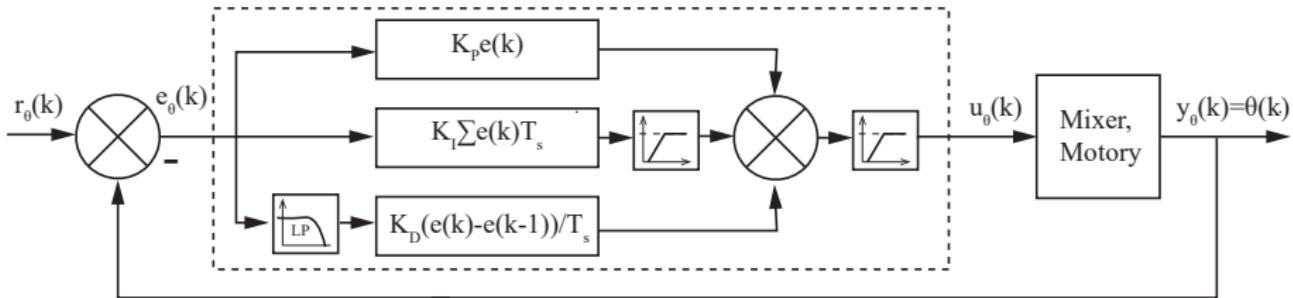


- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind) a tým pádom prestrelíme (*angl.*: overshoot) žiadané hodnoty
- To je saturácia integračnej zložky (*angl.*: integral windup).
- Môžeme používať rôzne triky, napr. ohraničiť veľkosť integračnej zložky, resp. vypnúť zložku pri určitých podmienkach.

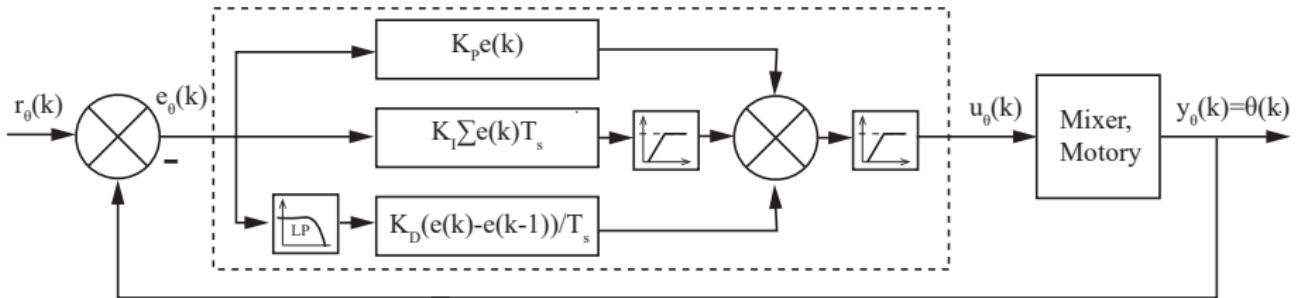


- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadovať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind) a tým pádom prestrelíme (*angl.*: overshoot) žiadane hodnoty
- To je saturácia integračnej zložky (*angl.*: integral windup).
- Môžeme používať rôzne triky, napr. ohraničiť veľkosť integračnej zložky, resp. vypnúť zložku pri určitých podmienkach.
- ArduPilot - Ak akčný člen je akurát saturovaný, podrž hodnotu I zložky. Nižšie to môže ísť, vyššie nie [\[Hall 2018\]](#).
- ArduPilot - Keďže máme kaskádnu konfiguráciu PID slučiek, saturačný znak postupuje cez hierarchiu nižšie a nižšie aby zastavil nahromadenie I zložky [\[Hall 2018\]](#).

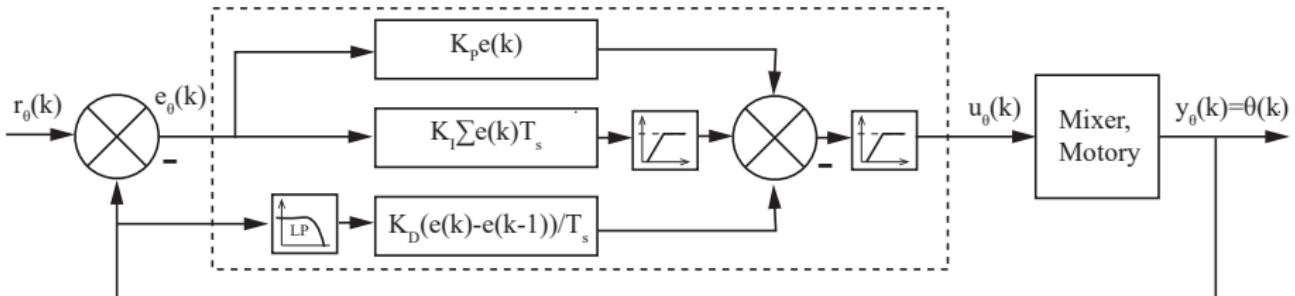
- Šum zo snímačov môže propagovať cez výpočet odchýlky riadenia do derivačnej zložky Vysokofrekvenčné zložky potom navýšia D zložku (čo je derivácia impulzu?)
- Riešenie: Odchýlku riadenia pustíme cez dolnopriepustný (angl.: low-pass) filter (LPF) — Aj ArduCopter (20 Hz LPF) aj PX4 Autopilot používa [Hall 2018; PX4 Autopilot 2021a]
- Náhle zmeny spôsobia “kopnutie” riadenia (angl.: derivative kick). (čo je derivácia impulzu?)
- Môžeme celkovo obísť zmenu žiadanej hodnoty a tým odchýlky $e_\Theta(k)$ tak, že derivujeme výstup $y_\Theta(k)^2$



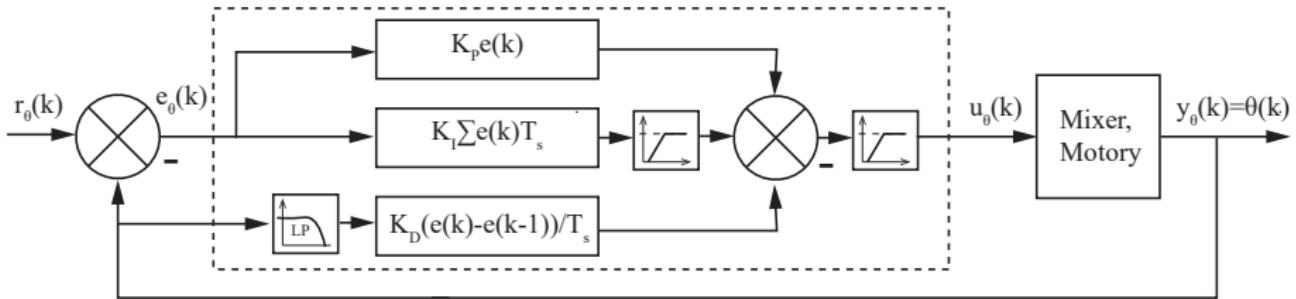
- Šum zo snímačov môže propagovať cez výpočet odchýlky riadenia do derivačnej zložky Vysokofrekvenčné zložky potom navýšia D zložku (čo je derivácia impulzu?)
- Riešenie: Odchýlku riadenia pustíme cez dolnopriepustný (*angl.*: low-pass) filter (LPF) — Aj ArduCopter (20 Hz LPF) aj PX4 Autopilot používa [[Hall 2018](#); [PX4 Autopilot 2021a](#)]
- Náhle zmeny spôsobia “kopnutie” riadenia (*angl.*: derivative kick). (čo je derivácia impulzu?)
- Môžeme celkovo obísť zmenu žiadanej hodnoty a tým odchýlky $e_\theta(k)$ tak, že derivujeme výstup $y_\theta(k)^2$

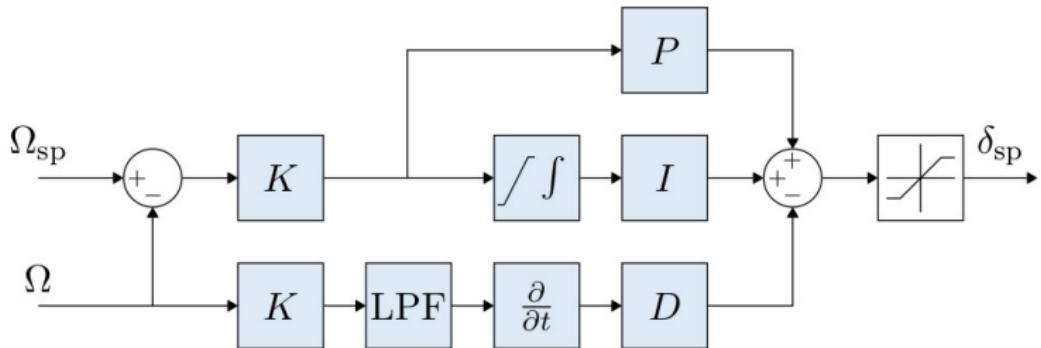


- Šum zo snímačov môže propagovať cez výpočet odchýlky riadenia do derivačnej zložky Vysokofrekvenčné zložky potom navýšia D zložku (čo je derivácia impulzu?)
- Riešenie: Odchýlku riadenia pustíme cez dolnopriepustný (*angl.*: low-pass) filter (LPF) — Aj ArduCopter (20 Hz LPF) aj PX4 Autopilot používa [Hall 2018; PX4 Autopilot 2021a]
- Náhle zmeny spôsobia “kopnutie” riadenia (*angl.*: derivative kick). (čo je derivácia impulzu?)
- Môžeme celkovo obísť zmenu žiadanej hodnoty a tým odchýlky $e_\Theta(k)$ tak, že derivujeme výstup $y_\Theta(k)^2$



- Šum zo snímačov môže propagovať cez výpočet odchýlky riadenia do derivačnej zložky Vysokofrekvenčné zložky potom navýšia D zložku (čo je derivácia impulzu?)
- Riešenie: Odchýlku riadenia pustíme cez dolnopriepustný (angl.: low-pass) filter (LPF) — Aj ArduCopter (20 Hz LPF) aj PX4 Autopilot používa [Hall 2018; PX4 Autopilot 2021a]
- Náhle zmeny spôsobia “kopnutie” riadenia (angl.: derivative kick). (čo je derivácia impulzu?)
- Môžeme celkovo obísť zmenu žiadanej hodnoty a tým odchýlky $e_\theta(k)$ tak, že derivujeme výstup $y_\theta(k)^2$





DCM -> Direction Cosine Matrix (pre 3.2) Transformation to inertial to body axes. Quaternion 4 by 1

- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v "body frame"
- Pre RC dron by to aj stačilo s plynom τ (*angl.*: throttle)³ [Boland 2015]
- Riadiť orientáciu (*angl.*: attitude) len na základe zmeny uhl. rýchlosťi (*angl.*: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow r_{\dot{\Theta}}$

³Síce rate-control je tiež možné [Boland 2015] !

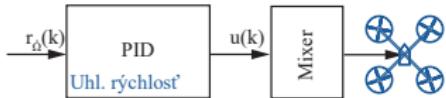
- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v "body frame"
- Pre RC dron by to aj stačilo s plynom τ (angl.: throttle)³ [Boland 2015]
- Riadiť orientáciu (angl.: attitude) len na základe zmeny uhl. rýchlosťi (angl.: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow \dot{r}_\Theta$

³Síce rate-control je tiež možné [Boland 2015] !

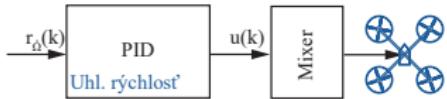
- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v "body frame"
- Pre RC dron by to aj stačilo s plynom τ (*angl.*: throttle)³ [Boland 2015]
- Riadiť orientáciu (*angl.*: attitude) len na základe zmeny uhl. rýchlosťi (*angl.*: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow r_{\dot{\Theta}}$

³Síce rate-control je tiež možné [Boland 2015] !

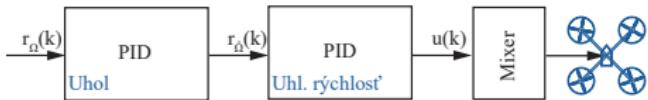
- Tvárme sa, že poznáme žiadané orientácie (napr. RC), napr. klopenie r_Θ a chceme riadiť y_Θ . V skutočnosti riešené s kvaternióny aby sme obišli singularitu Eulerových uhlov pri odhadе [Erasmus 2020]. Ak by sme priamo pilotovali RC , bol by to tzv. stabilizovaný letový mód [Boland 2015].
- Riadenie orientácie môže byť riešené ďalšou, nadradenou regulačnou slučkou - hovoríme o tzv. kaskádnom riadení (*angl.*: nested, cascaded).



- Tvárme sa, že poznáme žiadané orientácie (napr. RC), napr. klopenie r_Θ a chceme riadiť y_Θ . V skutočnosti riešené s kvaternióny aby sme obišli singularitu Eulerových uhlov pri odhadе [Erasmus 2020]. Ak by sme priamo pilotovali RC , bol by to tzv. stabilizovaný letový mód [Boland 2015].
- Riadenie orientácie môže byť riešené ďalšou, nadradenou regulačnou slučkou - hovoríme o tzv. kaskádnom riadení (*angl.*: nested, cascaded).



- Tvárme sa, že poznáme žiadané orientácie (napr. RC), napr. klopenie r_Θ a chceme riadiť y_Θ . V skutočnosti riešené s kvaternióny aby sme obišli singularitu Eulerových uhlov pri odhadе [Erasmus 2020]. Ak by sme priamo pilotovali RC , bol by to tzv. stabilizovaný letový mód [Boland 2015].
- Riadenie orientácie môže byť riešené ďalšou, nadradenou regulačnou slučkou - hovoríme o tzv. kaskádnom riadení (*angl.*: nested, cascaded).

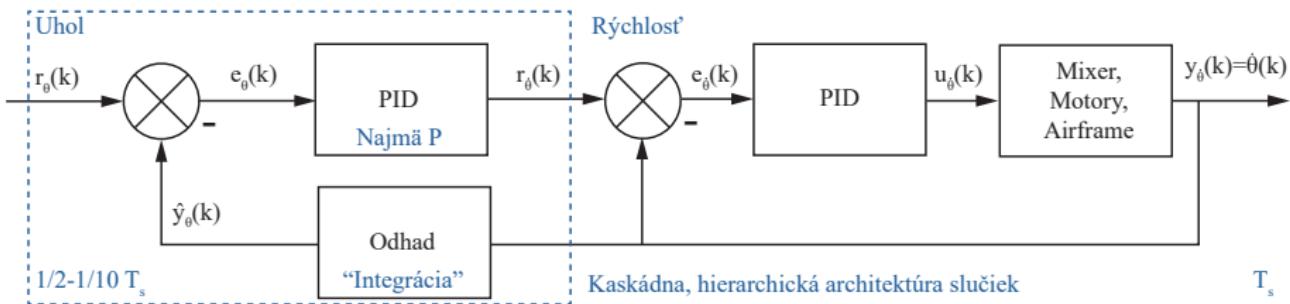


- Nadradené slučky sú pomalšie, vytvára to istý "filter", t.j. nemôžeme rýchlejšie ovládať rýchlosť ako polohu (cca. o rád, min polovicu pomalšie⁴) [Hall 2018; PX4 Autopilot 2021a]
- Pri PID skôr P⁵, lebo reaguje príliš agresívne na šum.
- Do slučky dopracujeme doprednú väzbu (*angl.*: feedforward) ktorá zrýchli odozvu regulácie. "Whatever works" - netreba mystifikovať.

⁴ArduCopter 40 Hz vs 400 Hz, PX4 250 vs. 1000 Hz [Hall 2018; PX4 Autopilot 2021a]

⁵ArduCopter a PX4 Autopilot používa P regulátor [PX4 Autopilot 2021a; ArduPilot 2021]

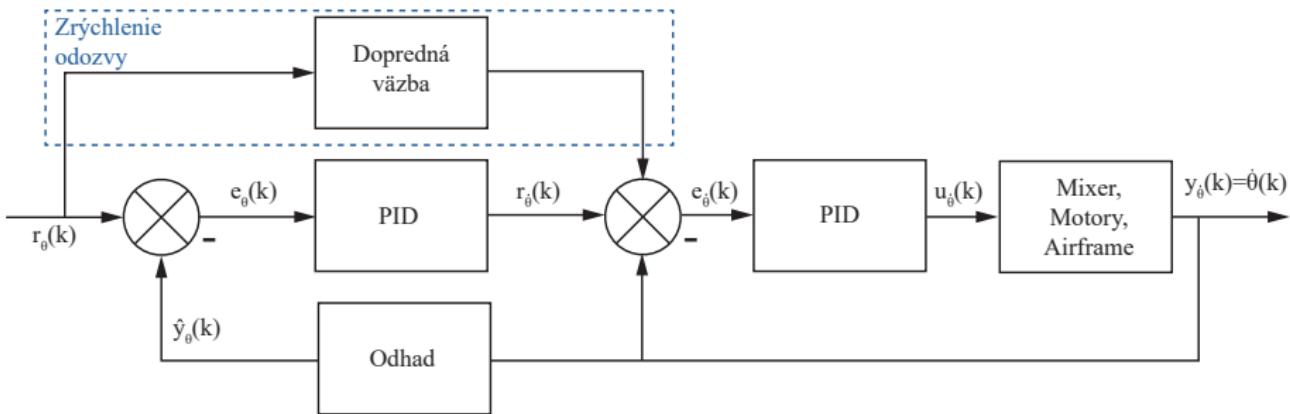
- Nadradené slučky sú pomalšie, vytvára to istý "filter", t.j. nemôžeme rýchlejšie ovládať rýchlosť ako polohu (cca. o rád, min polovicu pomalšie⁴) [Hall 2018; PX4 Autopilot 2021a]
- Pri PID skôr P⁵, lebo reaguje príliš agresívne na šum.
- Do slučky dopracujeme doprednú väzbu (*angl.*: feedforward) ktorá zrýchli odozvu regulácie. "Whatever works" - netreba mystifikovať.



⁴ArduCopter 40 Hz vs 400 Hz, PX4 250 vs. 1000 Hz [Hall 2018; PX4 Autopilot 2021a]

⁵ArduCopter a PX4 Autopilot používa P regulátor [PX4 Autopilot 2021a; ArduPilot 2021]

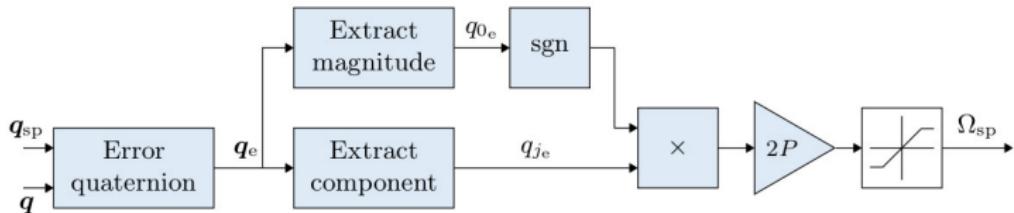
- Nadradené slučky sú pomalšie, vytvára to istý "filter", t.j. nemôžeme rýchlejšie ovládať rýchlosť ako polohu (cca. o rád, min polovicu pomalšie⁴) [Hall 2018; PX4 Autopilot 2021a]
- Pri PID skôr P⁵, lebo reaguje príliš agresívne na šum.
- Do slučky dopracujeme doprednú väzbu (*angl.*: feedforward) ktorá zrýchli odozvu regulácie. "Whatever works" - netreba mystifikovať.



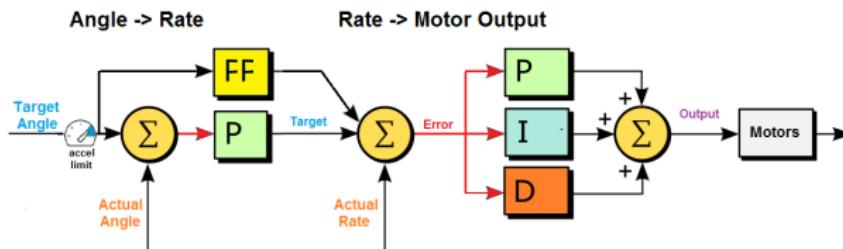
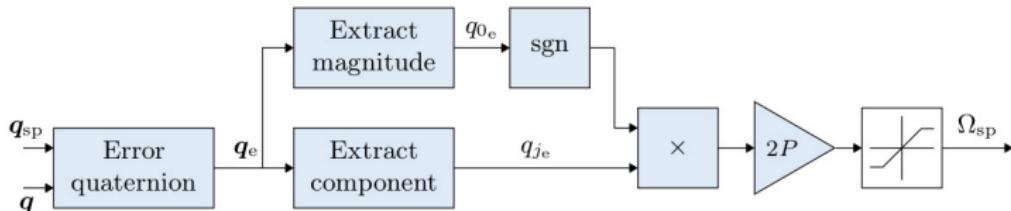
⁴ArduCopter 40 Hz vs 400 Hz, PX4 250 vs. 1000 Hz [Hall 2018; PX4 Autopilot 2021a]

⁵ArduCopter a PX4 Autopilot používa P regulátor [PX4 Autopilot 2021a; ArduPilot 2021]

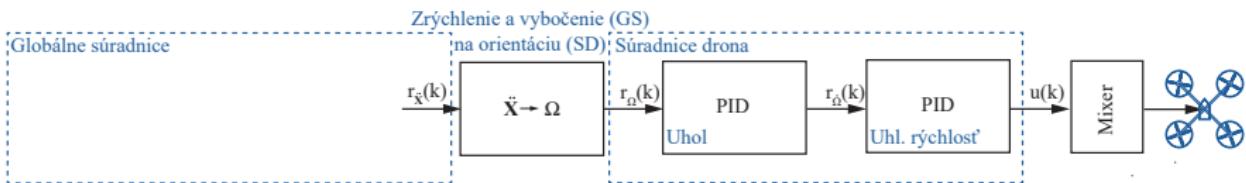
- PX4 používa kvaternióny, ale je to iba P regulátor
- ArduCopter v podstate taktiež tam má P regulátor + FF



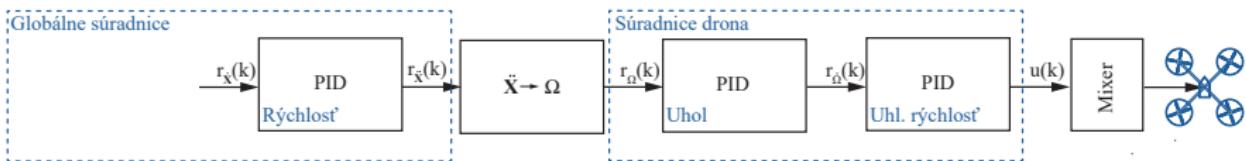
- PX4 používa kvaternióny, ale je to iba P regulátor
- ArduCopter v podstate taktiež tam má P regulátor + FF



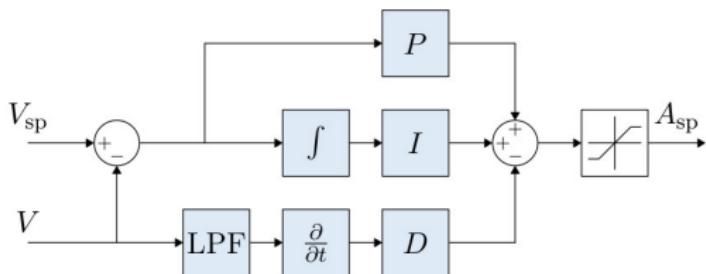
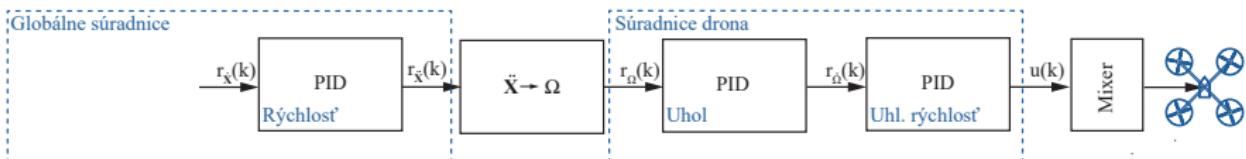
• aaaa



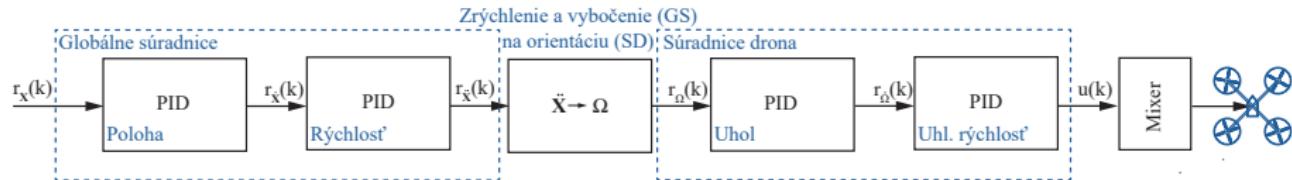
• aaaa



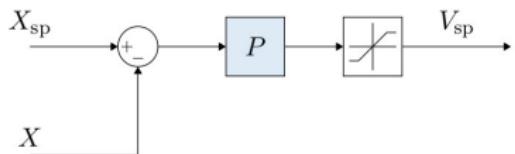
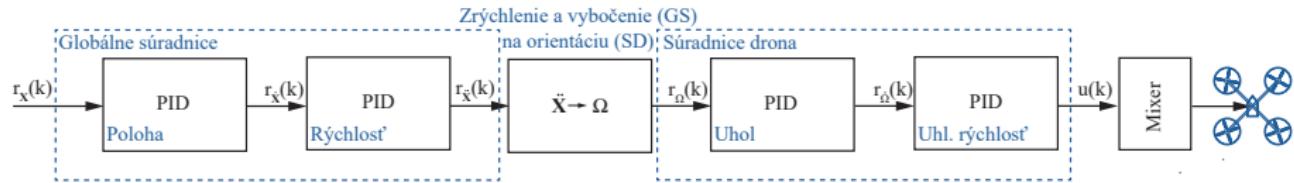
- aaaa



● aaaa



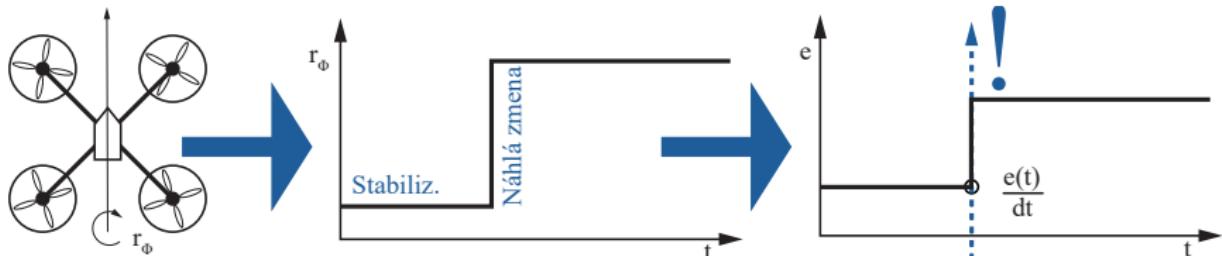
● aaaa



- Dron je stabilizovaný, pilot/ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadana hodnota je skokový signál. Derivácia skoku je... D zložka a tým aj vstup do akčných členov vystrelí!
- Potrebujeme tvarovať vstupy do regulácie (*angl.: input shaping*), t.j. tvarovať žiadane hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia)⁶
 - ▶ Vyhladenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "pocit" je riadiť stroj

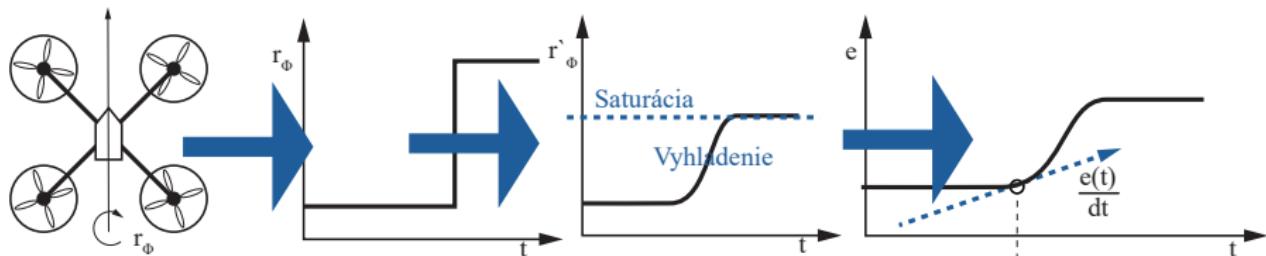
⁶ArduCopter 50 Hz → 400 Hz [Hall 2018]

- Dron je stabilizovaný, pilot/ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadana hodnota je skokový signál. Derivácia skoku je... D zložka a tým aj vstup do akčných členov vystrelí!
- Potrebujeme tvarovať vstupy do regulácie (angl.: input shaping), t.j. tvarovať žiadane hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia)⁶
 - ▶ Vyhladenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "počit" je riadiť stroj



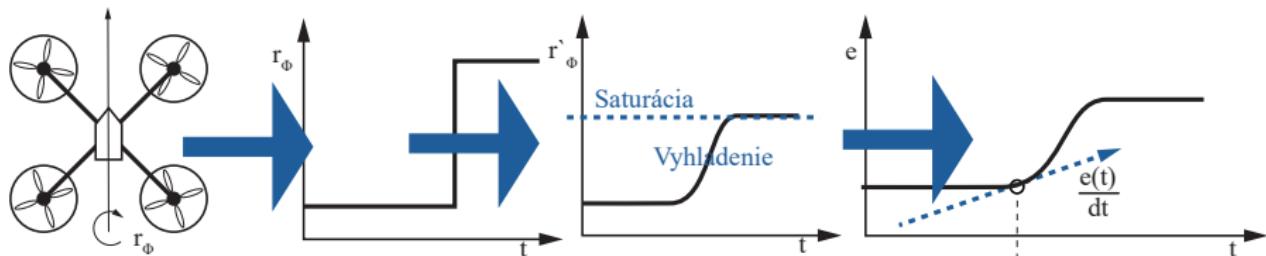
⁶ArduCopter 50 Hz → 400 Hz [Hall 2018]

- Dron je stabilizovaný, pilot/ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadana hodnota je skokový signál. Derivácia skoku je... D zložka a tým aj vstup do akčných členov vystrelí!
- Potrebujeme tvarovať vstupy do regulácie (angl.: input shaping), t.j. tvarovať žiadane hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia) ⁶
 - ▶ Vyhladenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "pocit" je riadiť stroj



⁶ArduCopter 50 Hz → 400 Hz [Hall 2018]

- Dron je stabilizovaný, pilot/ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadana hodnota je skokový signál. Derivácia skoku je... D zložka a tým aj vstup do akčných členov vystrelí!
- Potrebujeme tvarovať vstupy do regulácie (angl.: input shaping), t.j. tvarovať žiadane hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia) ⁶
 - ▶ Vyhľadenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "pocit" je riadiť stroj



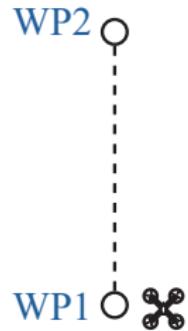
⁶ArduCopter 50 Hz → 400 Hz [Hall 2018]

- Ako naštartujeme riadenie? Aká je odchýlka $e(0)$ pri štarte?
- Kritické pri zmene letových módov [Hall 2018; Bresciani a kol. 2020]
- Pre ArduCopter Hall 2018
 - ▶ Nadradené riadenie poloha vs. rýchlosť (P) — tak aby vstup bol konštantný
 - ▶ Rýchlosť (PID) — $e(0) = 0$, I zložka s konštantným vstupom, zrátať D pri nastavení žiadanej hodnoty

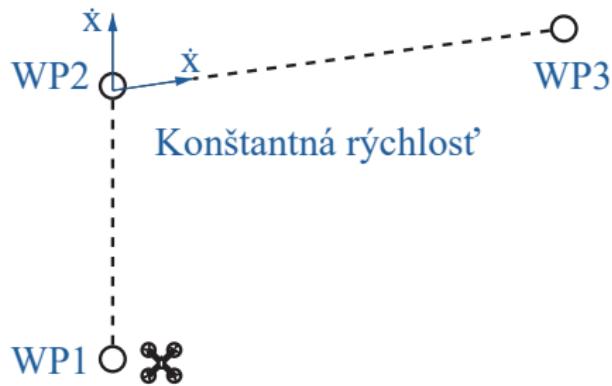
- Ako naštartujeme riadenie? Aká je odchýlka $e(0)$ pri štarte?
- Kritické pri zmene letových módov [Hall 2018; Bresciani a kol. 2020]
- Pre ArduCopter Hall 2018
 - ▶ Nadradené riadenie poloha vs. rýchlosť (P) — tak aby vstup bol konštantný
 - ▶ Rýchlosť (PID) — $e(0) = 0$, I zložka s konštantným vstupom, zrátať D pri nastavení žiadanej hodnoty

- Ako naštartujeme riadenie? Aká je odchýlka $e(0)$ pri štarte?
- Kritické pri zmene letových módov [Hall 2018; Bresciani a kol. 2020]
- Pre ArduCopter Hall 2018
 - ▶ Nadradené riadenie poloha vs. rýchlosť (P) — tak aby vstup bol konštantný
 - ▶ Rýchlosť (PID) — $e(0) = 0$, I zložka s konštantným vstupom, zrátať D pri nastavení žiadanej hodnoty

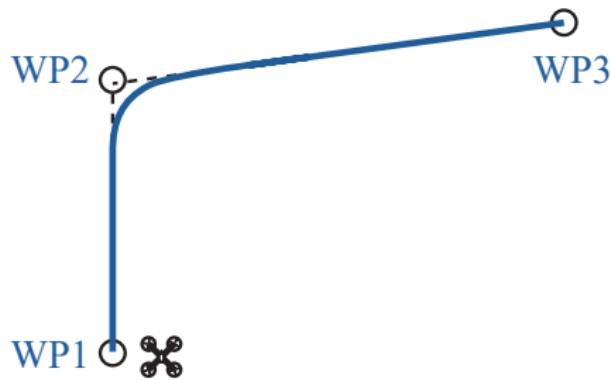
- Majme trasu WP1 do WP2, všetko je v poriadku.
- Pridajme WP3 a rozmýšľajme čo sa deje pri WP2. Je možné preletieť nad WP2?
- Bud' musíme úplne sa zastaviť alebo nemôžeme priamo preletieť — ináč by sme potrebovali nekonečne veľké zrýchlenia



- Majme trasu WP1 do WP2, všetko je v poriadku.
- Pridajme WP3 a rozmýšľajme čo sa deje pri WP2. Je možné preletieť nad WP2?
- Bud' musíme úplne sa zastaviť alebo nemôžeme priamo preletieť — ináč by sme potrebovali nekonečne veľké zrýchlenia



- Majme trasu WP1 do WP2, všetko je v poriadku.
- Pridajme WP3 a rozmýšľajme čo sa deje pri WP2. Je možné preletieť nad WP2?
- Bud' musíme úplne sa zastaviť alebo nemôžeme priamo preletieť — ináč by sme potrebovali nekonečne veľké zrýchlenia



- ArduPilot (ArduCopter) — Napodobňuje heuristiku [Tridgell 2021]
 - ▶ Zvyšuje D parameter pomaly
 - ▶ Akonáhle deteguje osciláciu, zníži D parameter
 - ▶ Opakuje pre iné zložky/slučky
- Intuitívne ale konzervatívne

Notes: [Hall 2018](#)

User (ROS) → Shaping → PID → Actuators Yaw je prioritizovanych nad Pitch Roll, lebo to drzi dron v lufte [\[Erasmus 2020\]](#) 50 Hz -> 400 Hz Min 24 tazsie uchopytelne koncepty pre prezentaciu, dava menej konkretnosti Velocity prioritizuje vertikalnu rychlosť [\[Erasmus 2020\]](#)

Ďakujem za Vašu pozornosť.

- [1] ArduPilot. *Copter Attitude Control*. Online. [cited 29.11.2021]; Available from <https://ardupilot.org/dev/docs/apmcopter-programming-attitude-control-2.html>. 2021.
- [2] K. J. Åström a R. M. Murray. *Analysis and Design of Feedback Systems*. [online]. Book preprint, Chapter 8. [cited 26.11.2021]; Available from https://www.cds.caltech.edu/~murray/courses/cds101/fa04/caltech/am04_ch8-3nov04.pdf. 2004.
- [3] Ryan Boland. *Embedded Programming for Quadcopters*. Online. [cited 2.12.2021]; Available from <https://www.youtube.com/watch?v=CHSYgLfhwUo&t=3s>. 2015.
- [4] Mathieu Bresciani a Matthias Grob. *Overview of multicopter control from sensors to motors*. Online. PX4 Developer Summit Virtual 2020. [cited 1.12.2021]; Available from https://www.youtube.com/watch?v=orvng_11ngQ. 2020.

- [5] **Anton Erasmus.** *An In-depth Look at the Multicopter Control System Architecture.* Online. PX4 Developer Summit Virtual 2020. [cited 1.12.2021]; Available from <https://www.youtube.com/watch?v=nEo4WG14Lgc>. 2020.
- [6] **Leonard Hall.** *Pratical PID implementation and the new Position Controller.* [online]. ArduPilot UnConference 2018, uploaded Feb 22, 2018 [cited 24.11.2021]; Available from <https://www.youtube.com/watch?v=-PC69jcMizA>. 2018.
- [7] **PX4 Autopilot.** *Controller Diagrams — Multicopter Control Architecture.* Online. [cited 29.11.2021]; Available from https://docs.px4.io/master/en/flight_stack/controller_diagrams.html. 2021.
- [8] **PX4 Autopilot.** *Multicopter PID Tuning Guide (Advanced/Detailed).* Online. [cited 30.11.2021]; Available from [MulticopterPIDTuningGuide\(Advanced/Detailed\)](MulticopterPIDTuningGuide(Advanced/Detailed).). 2021.

- [9] [Andrew Tridgell](#). *ArduPilot Log Analysis Seminar*. Online. [cited 10.11.2021]; Available from https://www.youtube.com/watch?v=WcfLTW_qZ08&list=PLC8WVaJJhN4ya_HDxh6qGBT6VbjXR6L5p. 2021.
- [10] [Wikipedia](#). *PID controller*. [online]. [cited 25.11.2021]; Available from https://en.wikipedia.org/wiki/PID_controller. 2021.