

Systémy riadenia letu kvadrioptéry

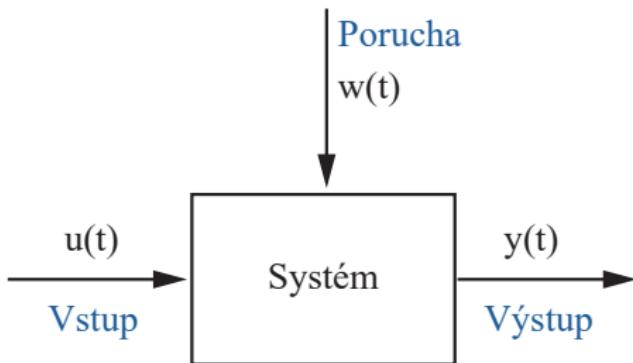
Kaskádové PID slučky

prof. Ing. Gergely Takács, PhD.

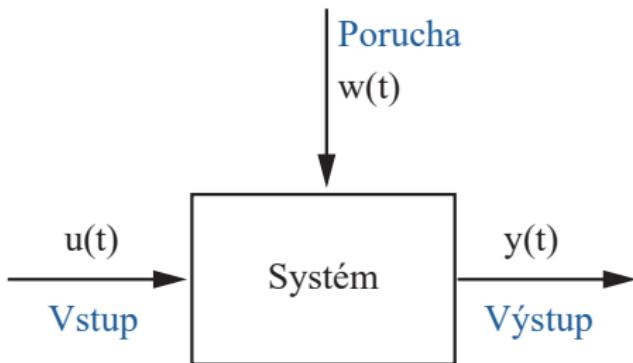


[https://github.com/gergelytakacs/
drone-control-lecture/](https://github.com/gergelytakacs/drone-control-lecture/)

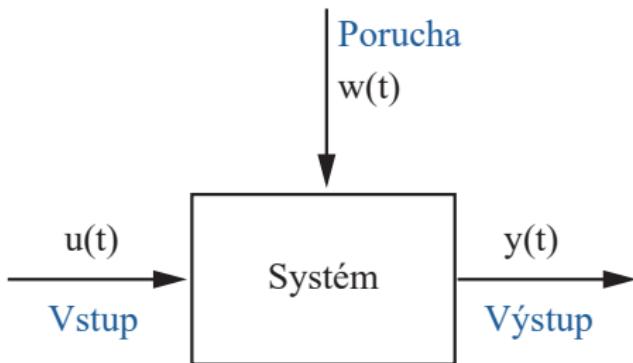
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), v tomto prípade celá kvadrioptéra
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM/RC do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie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



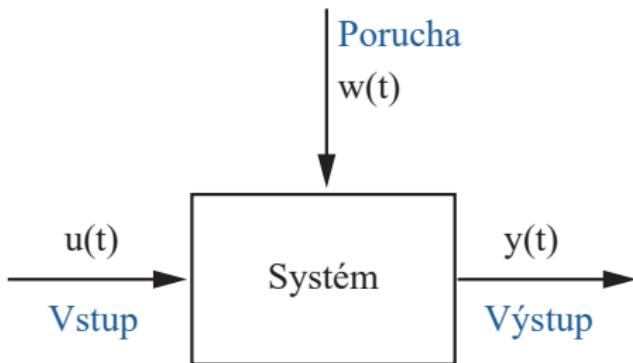
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), v tomto prípade celá kvadrioptéra
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM/RC do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie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



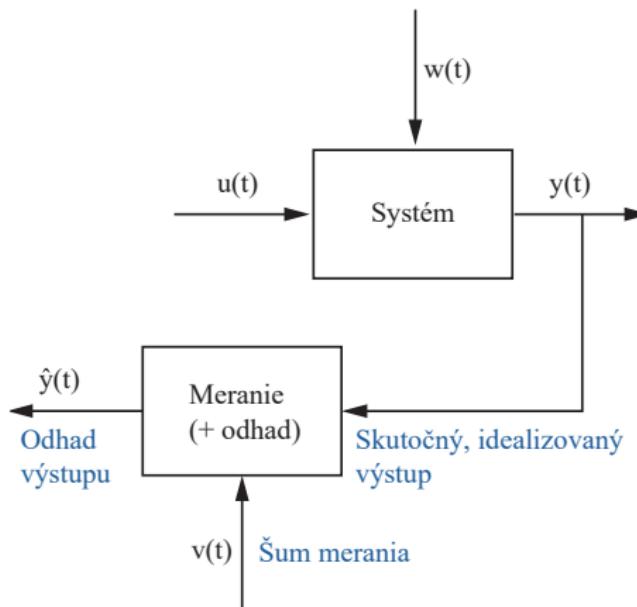
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), v tomto prípade celá kvadrioptéra
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM/RC do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie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



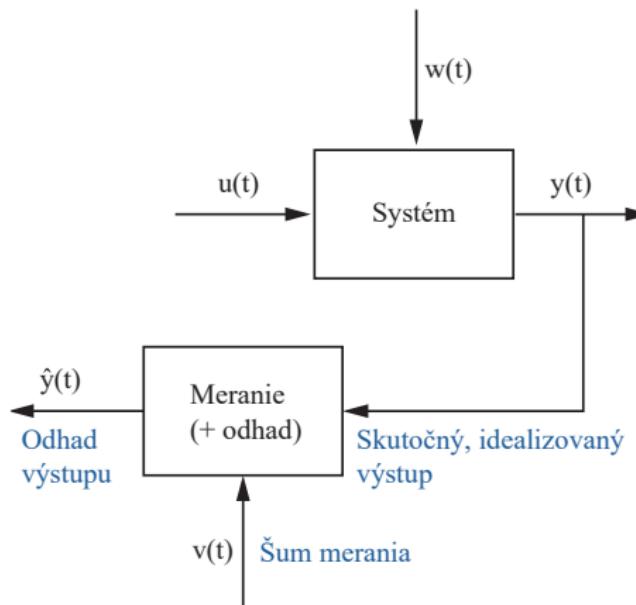
- Riadená sústava, systém alebo proces (*angl.*: plant, system, process), v tomto prípade celá kvadrioptéra
- Vstup $u(t)$ (*angl.*: input) sú akčné zásahy, napr. PWM/RC do motorov
- Výstup $y(t)$ (*angl.*: output) je meraná, tzv. manipulovaná veličina (*angl.*: manipulated variable); napr. klopenie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



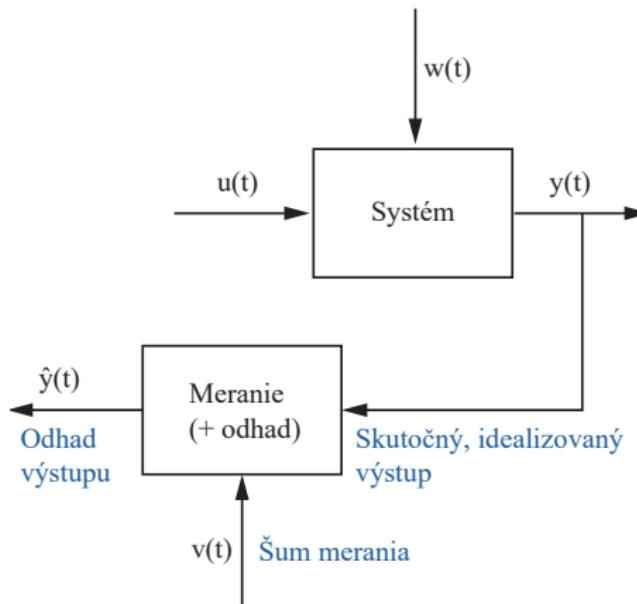
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



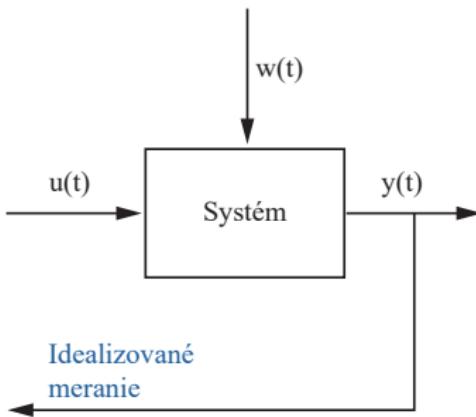
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



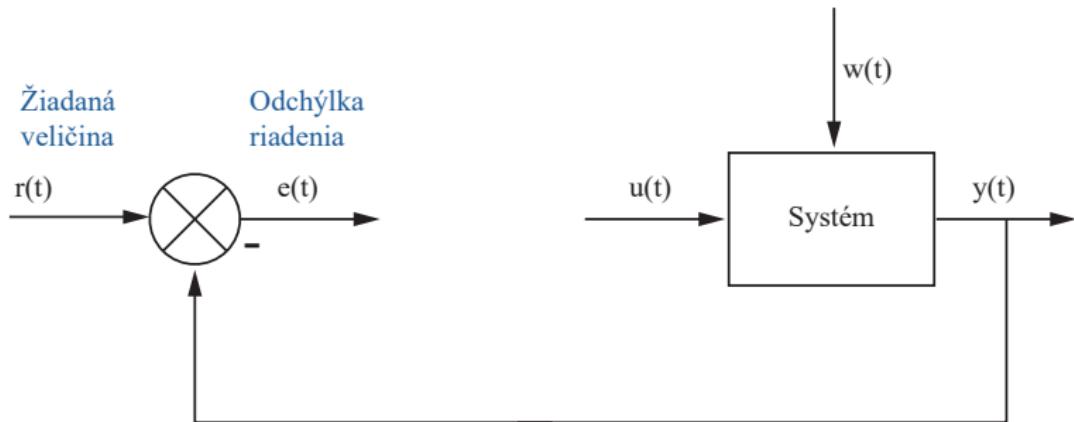
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



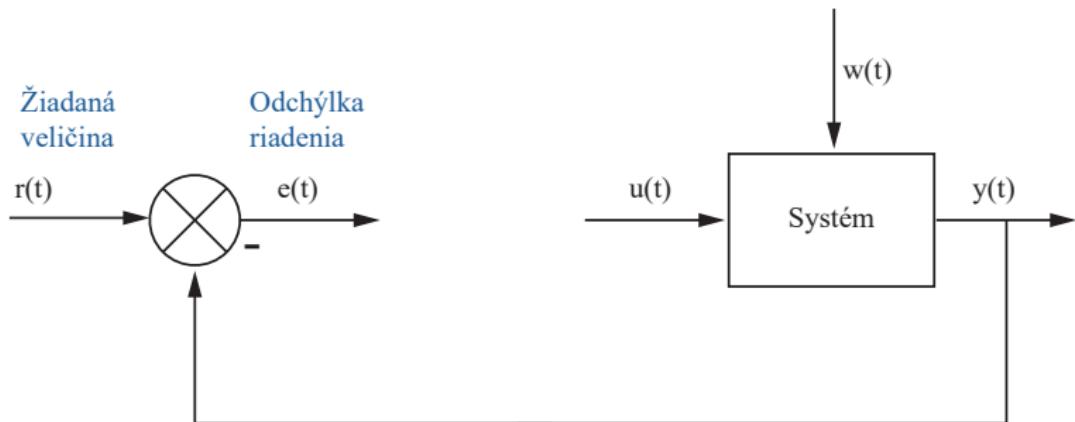
- Výstupy $y(t)$ sú idealizované
- Poruchy, teda šum merania $v(t)$ a dynamika snímača vplýva na výsledok
- Merania môžeme korigovať, resp. nemerané veličiny odhadnúť $\hat{y}(t)$
- Pre jednoduchosť často predpokladáme ideálne meranie $y(t) = \hat{y}(t)$



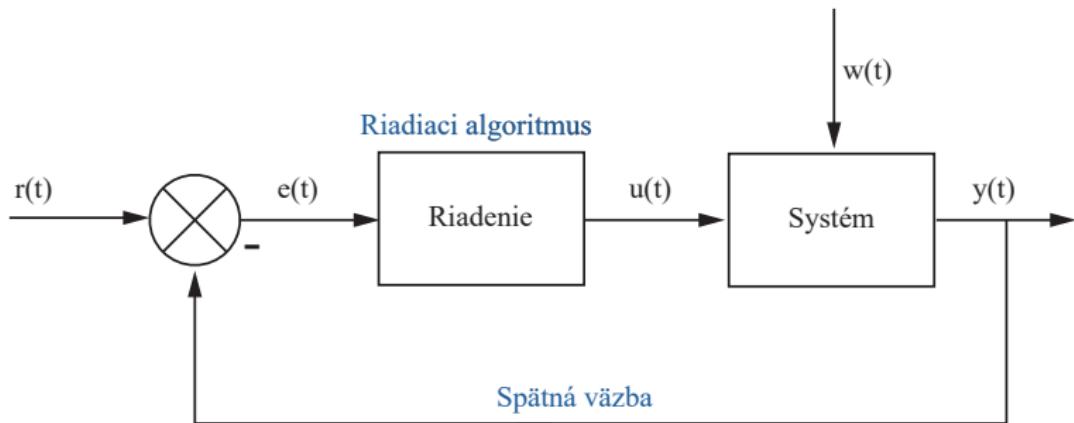
- Žiadaná hodnota alebo referencia (*angl.*: setpoint, reference) $r(t)$ vyjadruje na akú hodnotu chceme dostať výstup, potom
- Rozdiel medzi žiadanou hodnotou a výstupom $e(t) = r(t) - y(t)$ je odchýlka riadenia (*angl.*: error)



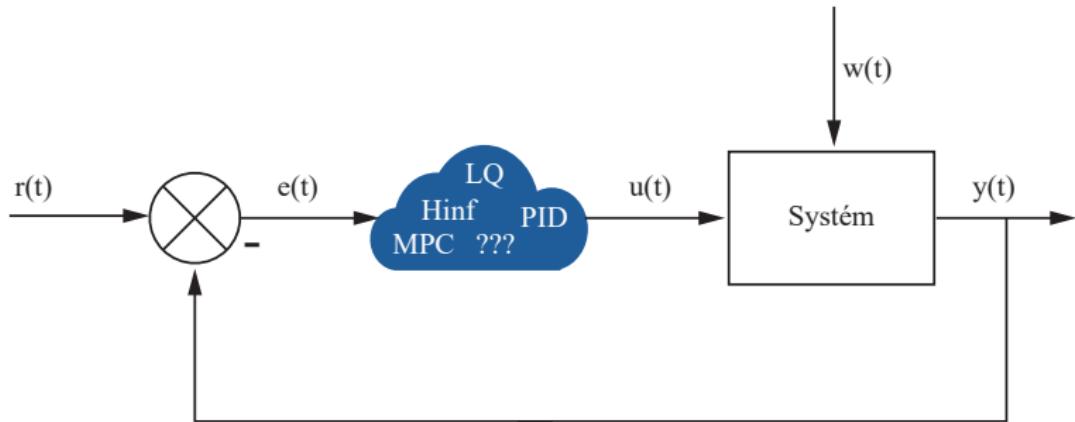
- Žiadaná hodnota alebo referencia (*angl.*: setpoint, reference) $r(t)$ vyjadruje na akú hodnotu chceme dostať výstup, potom
- Rozdiel medzi žiadanou hodnotou a výstupom $e(t) = r(t) - y(t)$ je odchýlka riadenia (*angl.*: error)



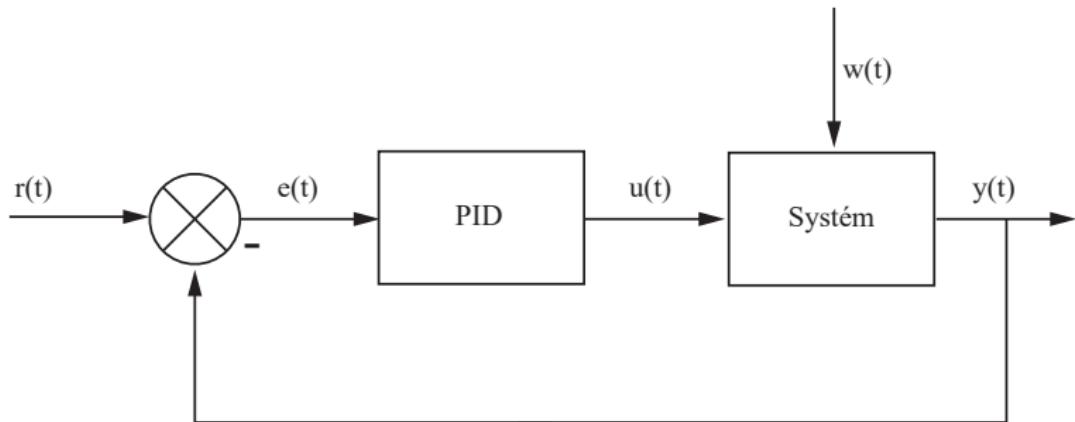
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. riadenie rýchlosťi parného stroja)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich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



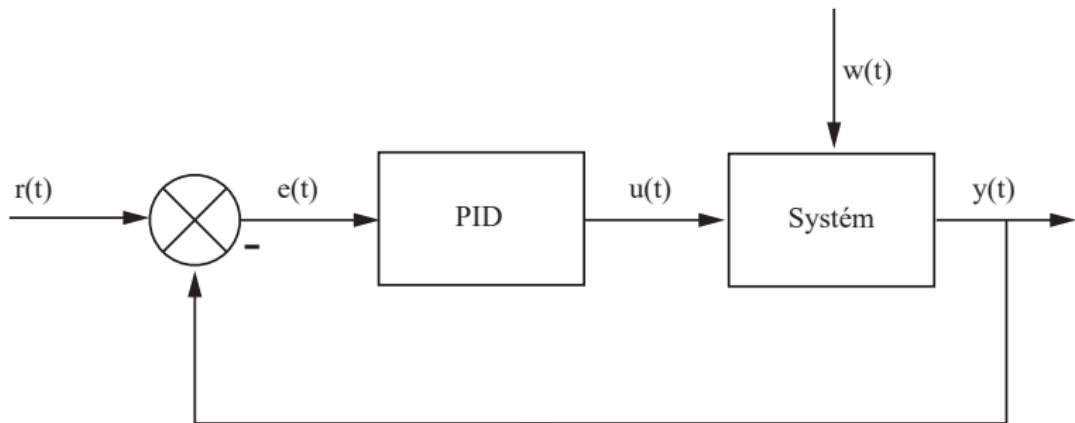
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. riadenie rýchlosťi parného stroja)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich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



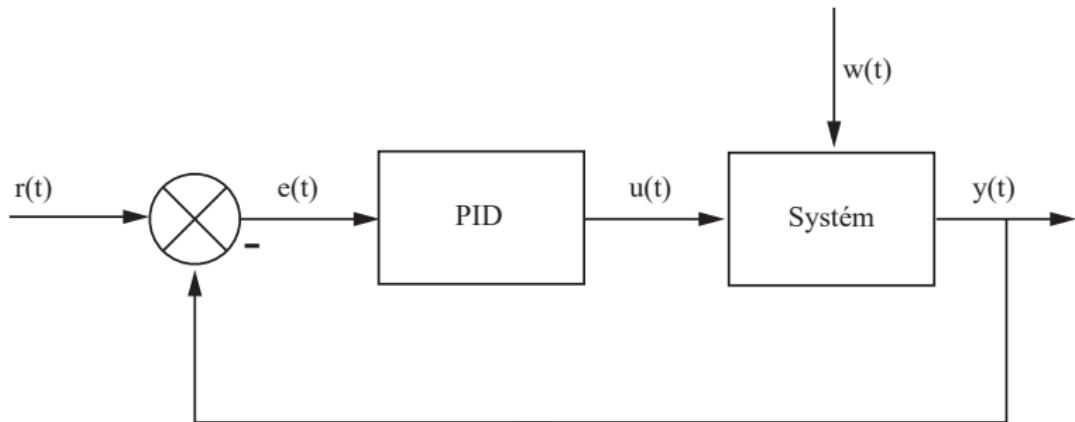
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. riadenie rýchlosťi parného stroja)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich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



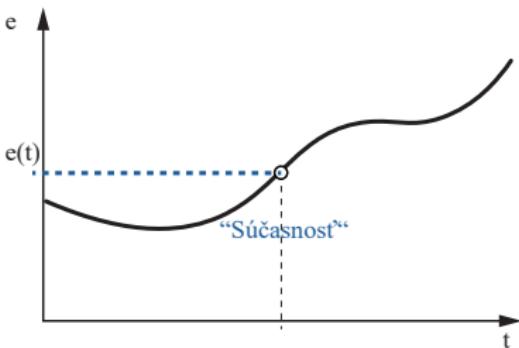
- Riadenie je logika ktorá prepočíta odchýlku riadenia $e(t)$ na vstupy $u(t)$
- Môže to byť ľubovoľné, do konca aj analógové a mechanické (e.g. riadenie rýchlosťi parného stroja)
- Dobrým kompromisom medzi komplexnosťou (pochopenie, návrh) a implementovateľnosťou je PID riadenie
- Až 97% riadiacich 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. riadenie rýchlosťi parného stroja)
- 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.



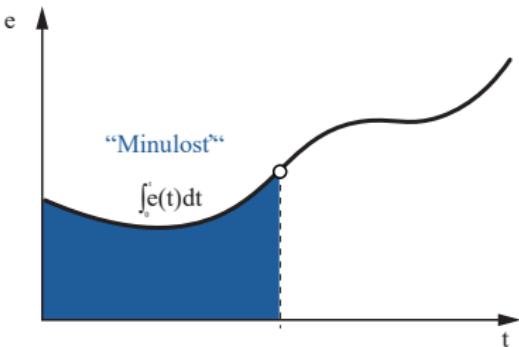
- **Súčasnosť** = proporcionálna zložka: okamžitá odchýlka $e(t)$, ale majme možnosť na ladenie, tak $K_P e(t)$.
- **Minulosť** = integračná zložka: celková odchýlka $e(t)$ v minulosti, ale majme možnosť na ladenie, tak $K_I \int_0^t e(t) dt$.
- **Budúcnosť** = derivačná zložka: projektovaná odchýlka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



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

²pozor, neintuitívne...

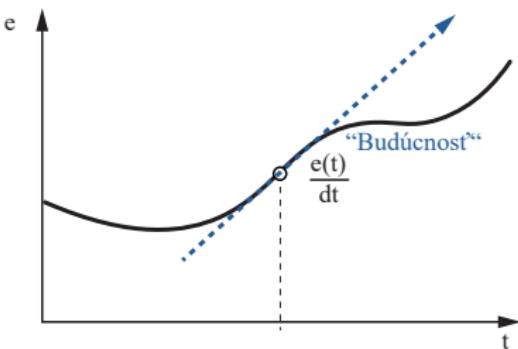
- **Súčasnosť** = proporcionálna zložka: okamžitá odchýlka $e(t)$, ale majme možnosť na ladenie, tak $K_P e(t)$.
- **Minulosť** = integračná zložka: celková odchýlka $e(t)$ v minulosti, ale majme možnosť na ladenie, tak $K_I \int_0^t e(t) dt$.
- **Budúcnosť** = derivačná zložka: projektovaná odchýlka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



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

²pozor, neintuitívne...

- **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**

²pozor, neintuitívne...

- **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](#)

²pozor, neintuitívne...

- **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

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

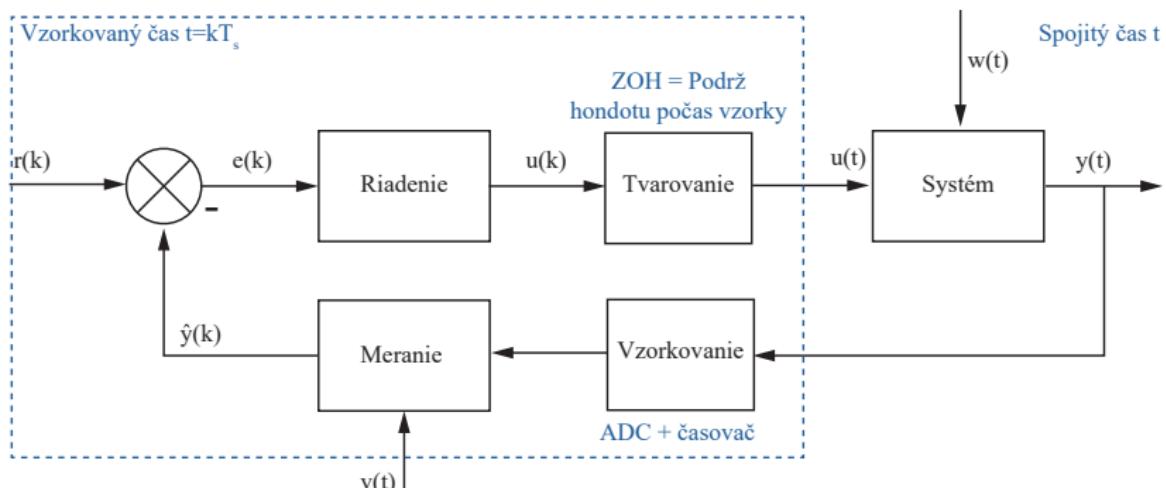
alebo tzv. štandardná forma¹ (máme intuitívnejšie časové konštanty, ale zosilnenie je nezávislé²)

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

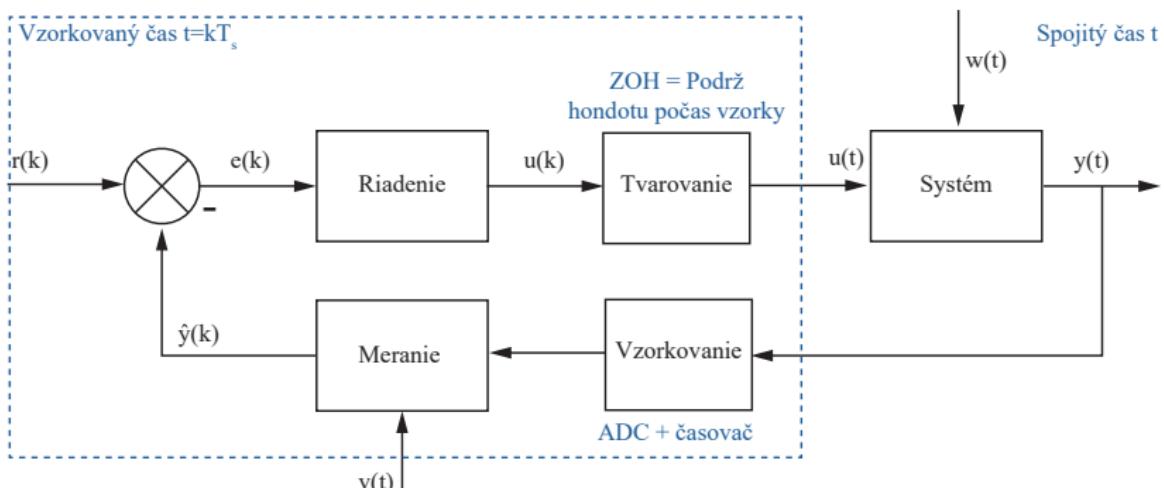
¹V PX4 môžete prepínať pre rate controller [PX4 Autopilot 2021b](#)

²pozor, neintuitívne...

- 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 spojitý čas na $t = kT_s$. Periódna je voliteľná, ale
 - ▶ musí zachytiť dominantnú dynamiku riadeného dejha,
 - ▶ a výpočtová realizácia musí stíhať.



- Systém, resp. proces zostáva spojity³
- 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)

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

³Samozrejme máme aj nespojité procesy a systémy

- Systém, resp. proces zostáva spojity³
- 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)

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

³Samozrejme máme aj nespojité procesy a systémy

- Systém, resp. proces zostáva spojity³
- Koncepty ako integrácia, derivácia musíme numericky approximovať, t.j. ak T_s je vzorkovací čas, naše PID bude

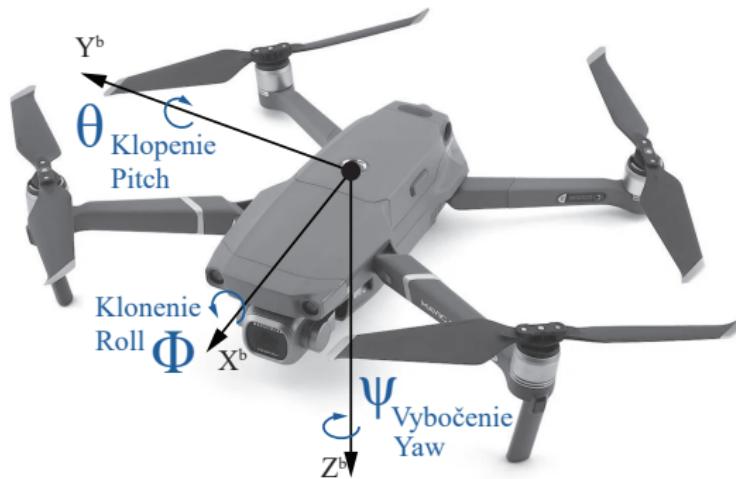
$$u(k) = K_P e(k) + K_I \underbrace{\sum_0^t e(k) T_s}_{\text{"obdlzniky"}} + K_D \underbrace{\frac{e(k) - e(k-1)}{T_s}}_{\text{"vstupanie"}} \quad (3)$$

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

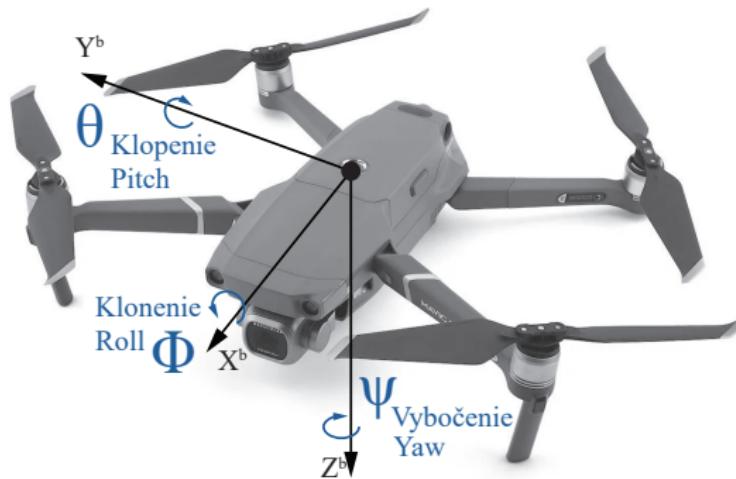
$$u(k) = u(k-1) + \left(K_P + K_I T_s + \frac{K_D}{T_s} \right) e(k) + \left(-K_P - 2 \frac{K_D}{T_s} \right) e(k-1) + \frac{K_D}{T_s} e(k-2) \quad (4)$$

³Samozrejme máme aj nespojité procesy a systémy

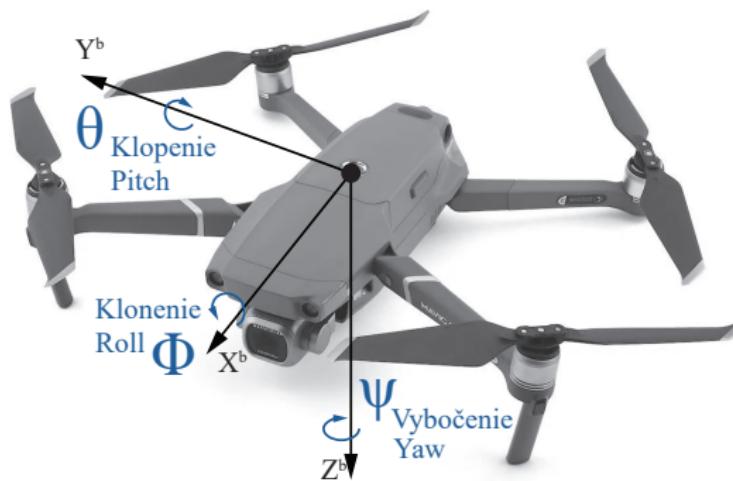
- Nestabilná (*angl.*: unstable) - duh...
- Nelineárna (*angl.*: nonlinear) - uhly, aerodynamika, atď'.
- Nedostatočne riadený (*angl.*: underactuated) - 4 vstupy, 6 DOF výstup
 - 3× posunutie a 3× otočenie [Douglas 2018]



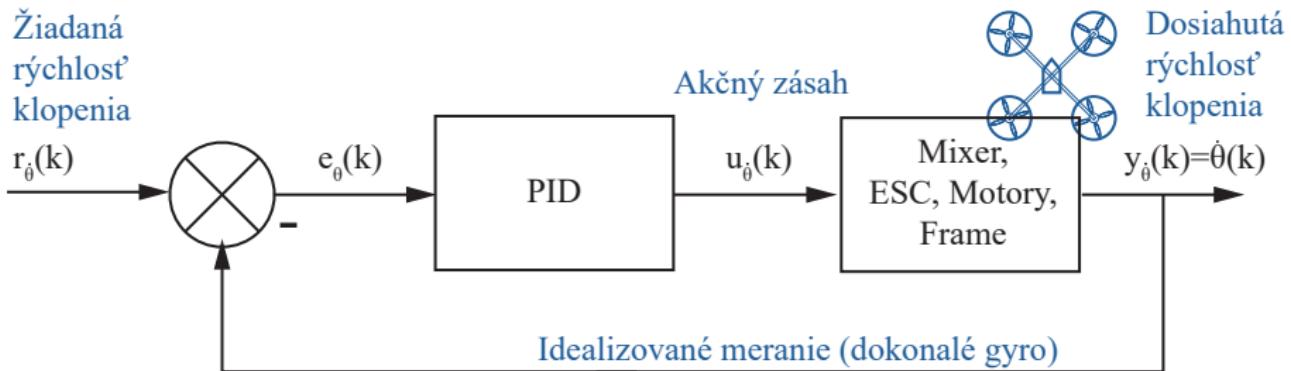
- Nestabilná (*angl.*: unstable) - duh...
- Nelineárna (*angl.*: nonlinear) - uhly, aerodynamika, atď.
- Nedostatočne riadený (*angl.*: underactuated) - 4 vstupy, 6 DOF výstup
 - 3× posunutie a 3× otočenie [Douglas 2018]



- Nestabilná (*angl.*: unstable) - duh...
- Nelineárna (*angl.*: nonlinear) - uhly, aerodynamika, atď.
- Nedostatočne riadený (*angl.*: underactuated) - 4 vstupy, 6 DOF výstup
 - 3× posunutie a 3× otočenie [Douglas 2018]

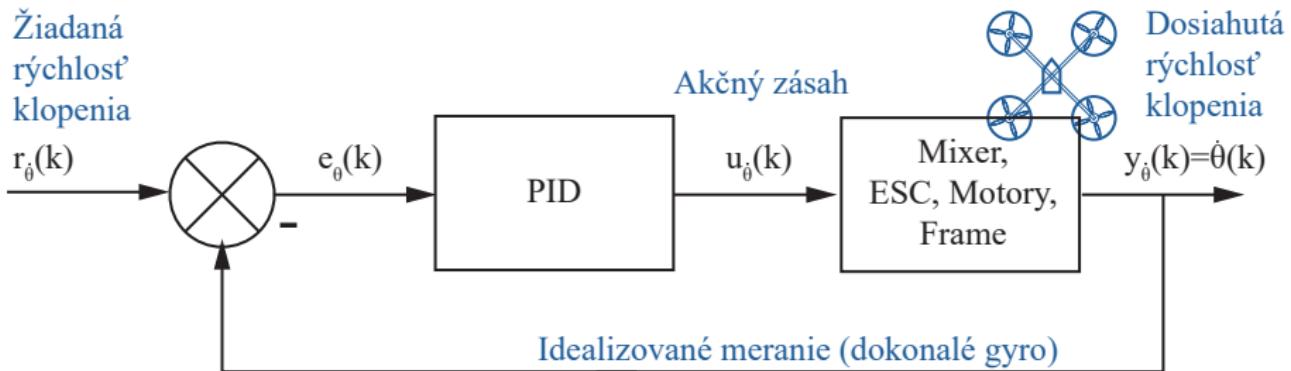


- Tvárame sa, že poznáme žiadanú uhlovú rýchlosť orientácie $r_{\dot{\Theta}}$, $r_{\dot{\phi}}$ a $r_{\dot{\psi}}$ resp r_h (vert. rýchlosť) a pre jednoduchosť sústredme len na rýchlosť zmeny klopenia ($\dot{\Theta}$).
- Sme na najnižšej úrovni, t.j. riadenie uhlovej rýchlosťi (angl.: rate controller) — je to zároveň aj najdôležitejšia slučka, a máme 4 slučky [Hall 2018; PX4 Autopilot 2021b] .

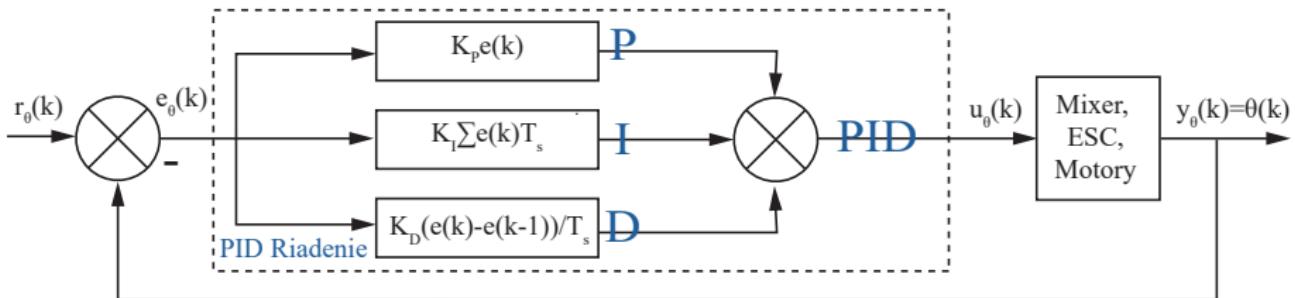


Rýchlosť zmeny uhlov

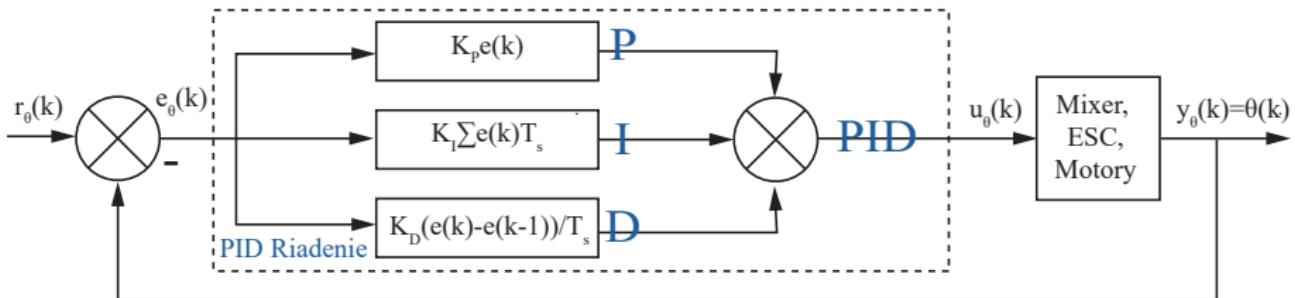
- Tvárame sa, že poznáme žiadanú uhlovú rýchlosť orientácie $r_{\dot{\Theta}}$, $r_{\dot{\phi}}$ a $r_{\dot{\psi}}$ resp r_h (vert. rýchlosť) a pre jednoduchosť sústredme len na rýchlosť zmeny klopenia ($\dot{\Theta}$).
- Sme na najnižšej úrovni, t.j. riadenie uhlovej rýchlosť (*angl.*: rate controller) — je to zároveň aj najdôležitejšia slučka, a máme 4 slučky [Hall 2018; PX4 Autopilot 2021b].



- Slučky sú nezávislé len pre malé uhly, lebo keď zvýšime letovú výšku h pri väčších e.g. Θ tak zvýšime aj vertikálnu aj horizontálnu rýchlosť [Douglas 2018] !
- 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



- Slučky sú nezávislé len pre malé uhly, lebo keď zvýšime letovú výšku h pri väčších e.g. Θ tak zvýšime aj vertikálnu aj horizontálnu rýchlosť [Douglas 2018] !
- 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



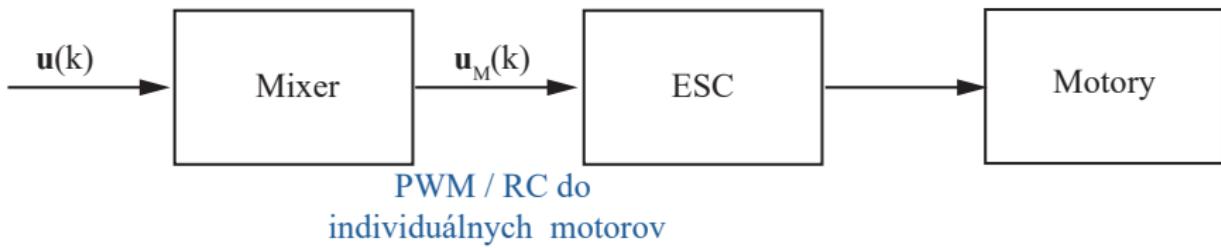
- Aj keď slučky PID na tejto úrovni definujú žiadanú zmenu $u(k)$, musíme implementovať pridelenie riadenia (*angl.*: control allocation) $u_M(k)$ (PWM, RC impulzy) do individuálnych motorov cez elektronické riadenie rýchlosťi (*angl.*: electronic speed control, ESC).
- V najjednoduchšom prípade máme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

$$u_{M1}(k) = u_h + u_\phi + u_\theta + u_\Sigma \quad (5)$$

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

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

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



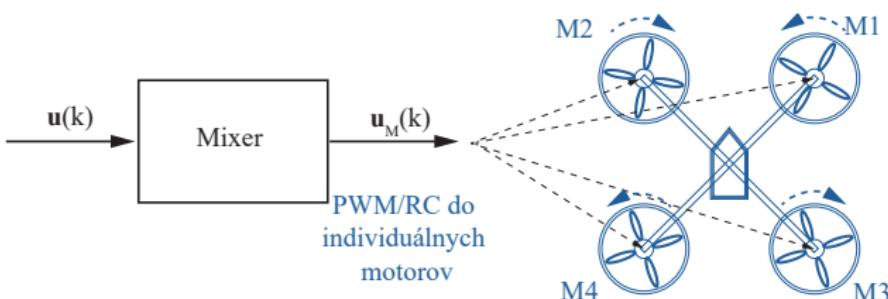
- Aj keď slučky PID na tejto úrovni definujú žiadanú zmenu $u(k)$, musíme implementovať pridelenie riadenia (*angl.*: control allocation) $u_M(k)$ (PWM, RC impulzy) do individuálnych motorov cez elektronické riadenie rýchlosťi (*angl.*: electronic speed control, ESC).
- V najjednoduchšom prípade máme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

$$u_{M1}(k) = u_h + u_\phi + u_\theta + u_\Sigma \quad (5)$$

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

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

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



- z čoho v maticovej forme

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

- 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] .

- z čoho v maticovej forme

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

- 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].

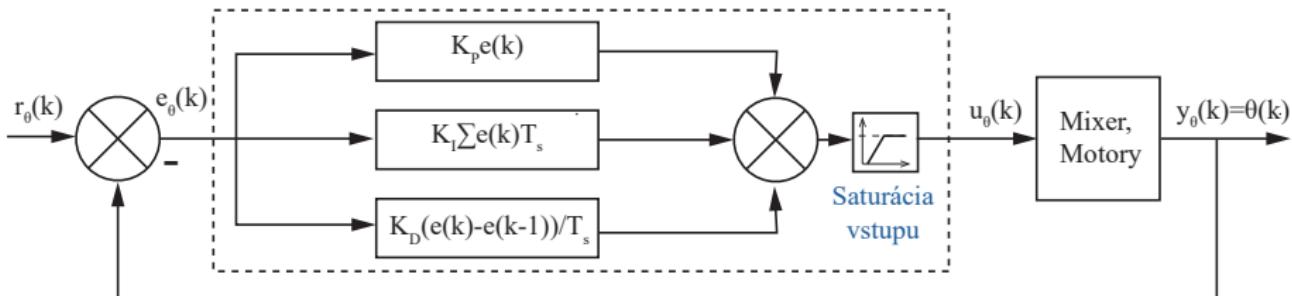
- z čoho v maticovej forme

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

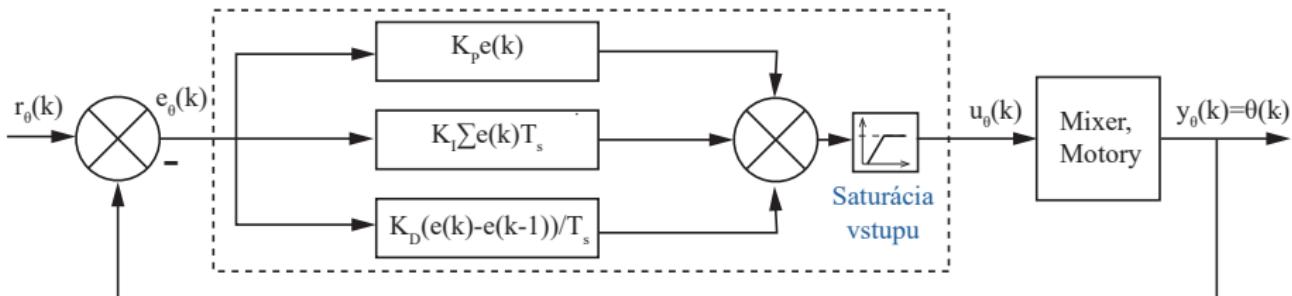
- Mapa $u_M(k) = Pu(k)$ kde P matica pridelenia riadenia (*angl.*: control allocation matrix). Nemusí obsahovať iba jednotky, môže byť aj neštvorcová ale našťastie transformácia je stále lineárna [Bresciani a kol. 2020].
- Môžeme priamo namapovať maticu efektivity akčných členov (*angl.*: actuator effectiveness 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?

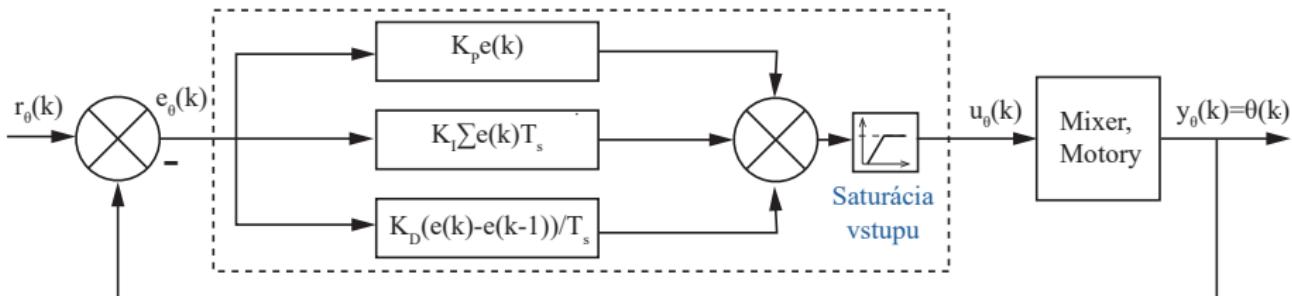
- Akčné zásahy majú svoje ohraničenia (angl.: constraints)
- Ako donútime ich dodržanie? Saturáciou (orezávaním) hodnôt, ktoré skutočne vypočítava PID.
- Saturácia vnáša nelinearitu, vplýva na výkon riadenia aj stabilitu.
- Aj iné veličiny môžeme saturovať, napr. žiadane hodnoty. Ako by sme ohraničili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



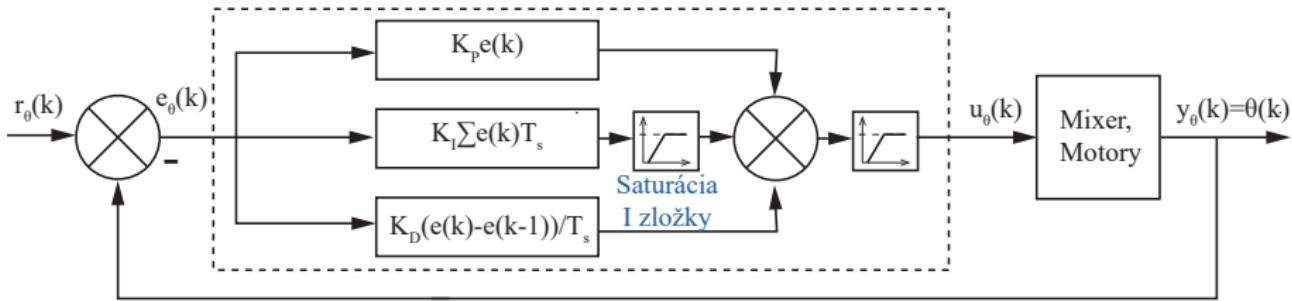
- Akčné zásahy majú svoje ohraničenia (angl.: constraints)
- Ako donútime ich dodržanie? Saturáciou (orezávaním) hodnôt, ktoré skutočne vypočítava PID.
- Saturácia vnáša nelinearitu, vplýva na výkon riadenia aj stabilitu.
- Aj iné veličiny môžeme saturovať, napr. žiadane hodnoty. Ako by sme ohraničili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



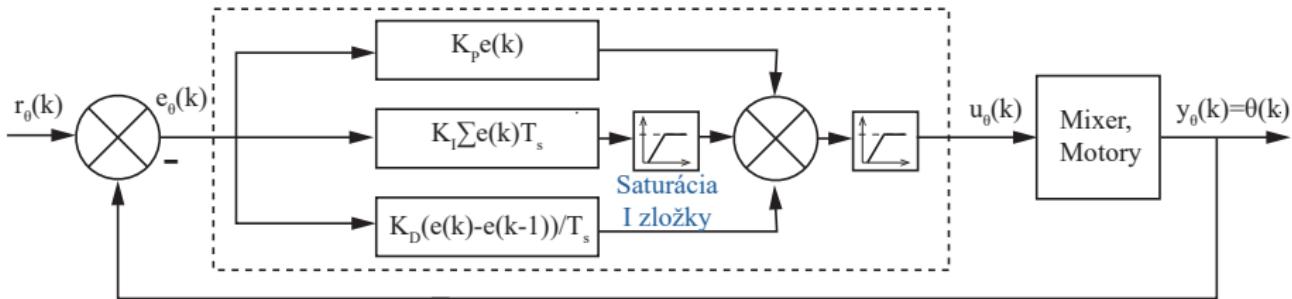
- Akčné zásahy majú svoje ohraničenia (angl.: constraints)
- Ako donútime ich dodržanie? Saturáciou (orezávaním) hodnôt, ktoré skutočne vypočítava PID.
- Saturácia vnáša nelinearitu, vplýva na výkon riadenia aj stabilitu.
- Aj iné veličiny môžeme saturovať, napr. žiadane hodnoty. Ako by sme ohraničili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



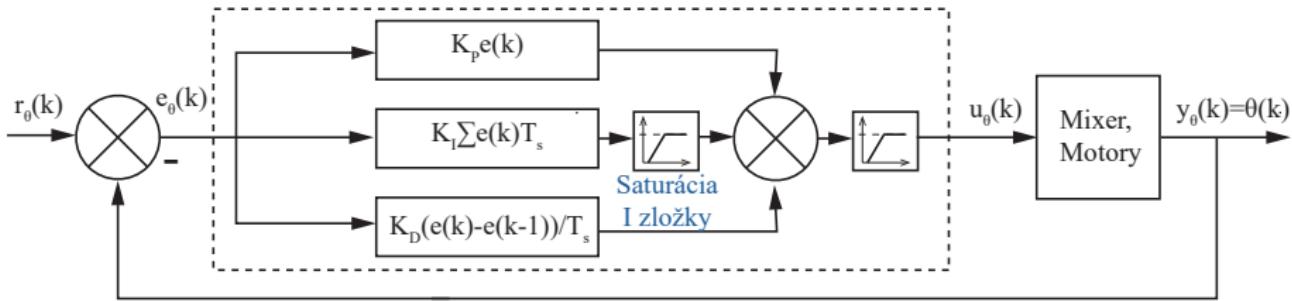
- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind)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



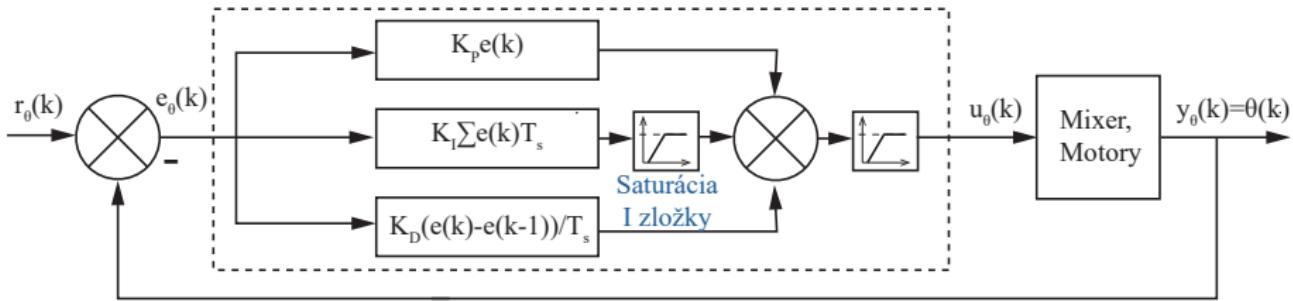
- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind)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

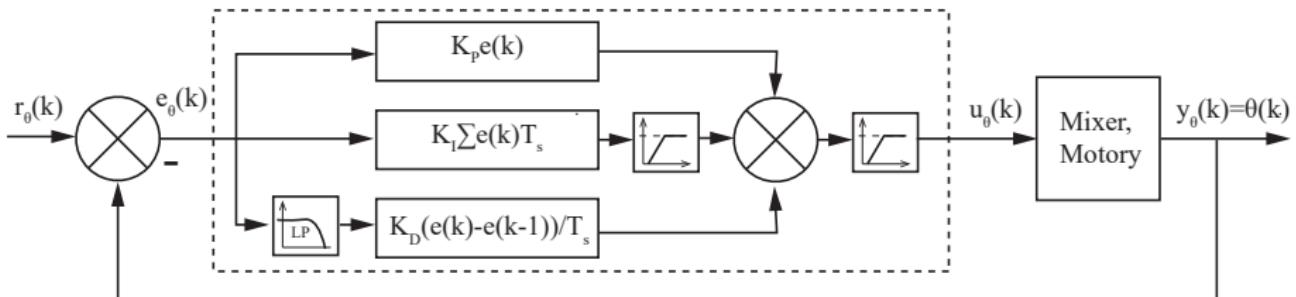


- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadať (angl.: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (angl.: unwind)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

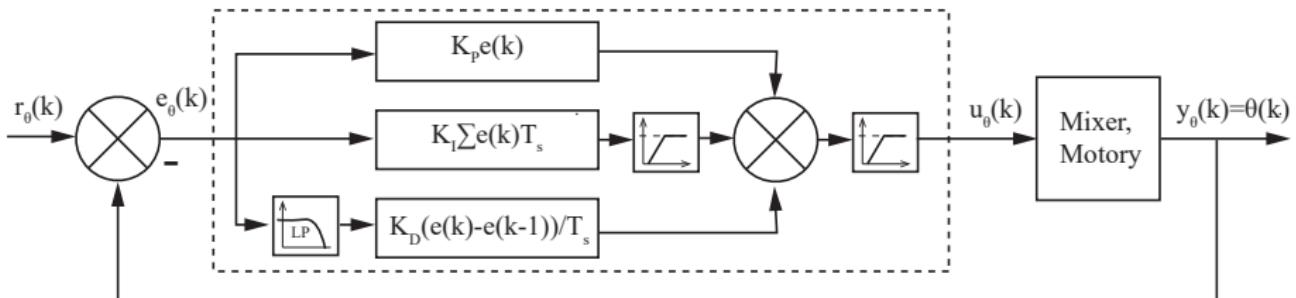


- Integračná zložka ráta odchýlku v minulosti, preto ak akčné členy sú už na hraniciach možností - začína sa nahromadovať (*angl.*: windup).
- Akonáhle sa vrátia akčné zásahy pod ohraničenia, nahromadená I zložka stále bude tlačiť systém na hranice možností, musí sa to chvíľu "uvolňovať" (*angl.*: unwind) a tým pádom prestrelíme (*angl.*: overshoot) žiadane hodnoty
- To je saturácia, resp nahromadenie 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 [\[Hall 2018\]](#).

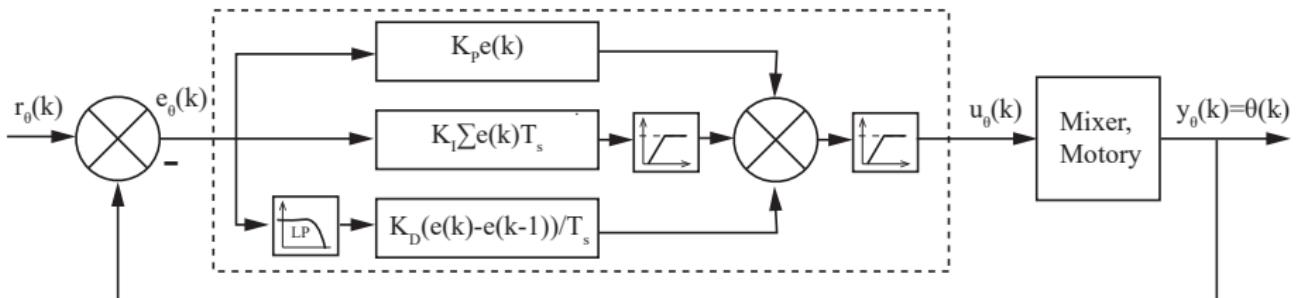
- Šum zo snímačov môže propagovať cez výpočet odchýlky riadenia do derivačnej zložky. Vysokofrekvenčné zložky potom navýšia D zložku (čo je derivácia impulzu?).
- Riešenie: Odchýlku riadenia pustíme cez dolnopriepustný (angl.: low-pass) filter (LPF)
- Aj ArduCopter (20 Hz LPF) aj PX4 Autopilot používajú [Hall 2018; PX4 Autopilot 2021a]



- 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 používajú [Hall 2018; PX4 Autopilot 2021a]



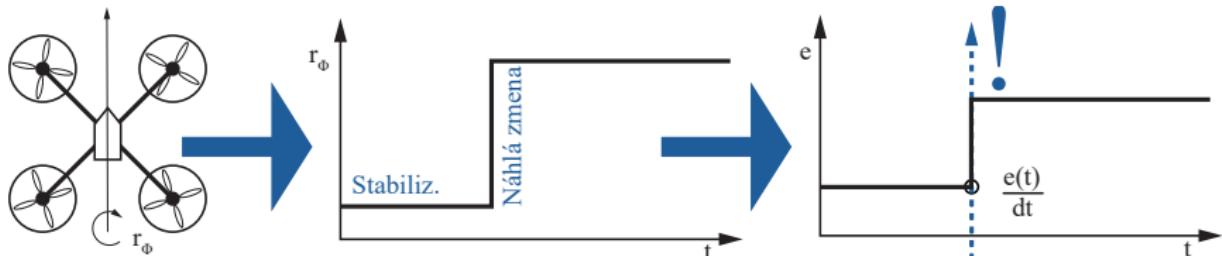
- Šum zo snímačov môže propagovať cez výpočet odchýlky riadenia do derivačnej zložky. Vysokofrekvenčné zložky potom navýšia D zložku (čo je derivácia impulzu?).
- Riešenie: Odchýlku riadenia pustíme cez dolnopriepustný (*angl.:* low-pass) filter (LPF)
- Aj ArduCopter (20 Hz LPF) aj PX4 Autopilot používajú [Hall 2018; PX4 Autopilot 2021a]



- Dron je stabilizovaný, pilot/ ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadaná 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 žiadané hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia) ⁴
 - ▶ Vyhladenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "pocit" je riadiť stroj

⁴ArduCopter 50 Hz → 400 Hz [Hall 2018]

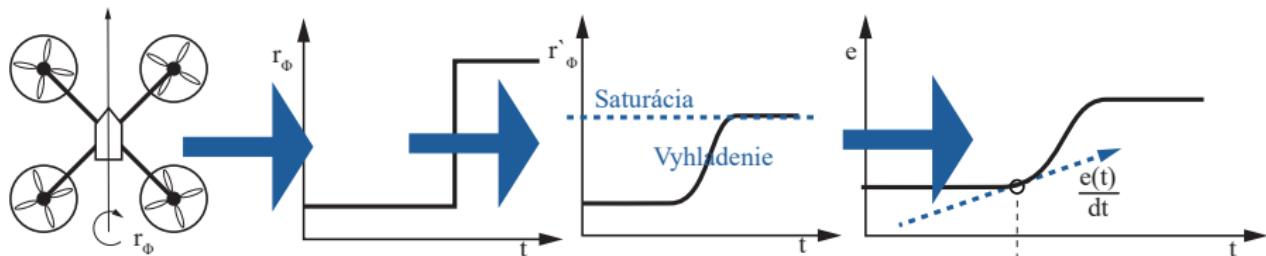
- Dron je stabilizovaný, pilot/ ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadaná hodnota je skokový signál. Derivácia skoku je... D zložka a tým aj vstup do akčných členov vystrelí!
- Potrebujeme tvarovať vstupy do regulácie (*angl.: input shaping*), t.j. tvarovať žiadané hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia)⁴
 - ▶ Vyhladenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "počit" je riadiť stroj



⁴ArduCopter 50 Hz → 400 Hz [Hall 2018]

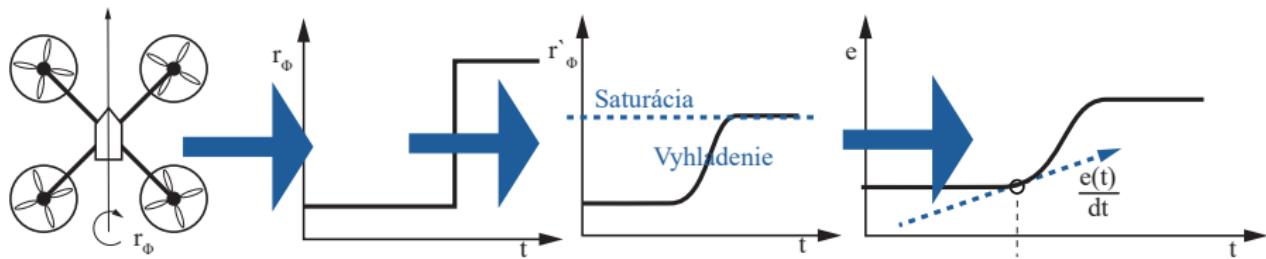
Kopnutie derivačnej zložky: Tvarovanie

- Dron je stabilizovaný, pilot/ ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadaná 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 žiadané hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia) ⁴
 - ▶ Vyhladenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "pocit" je riadiť stroj



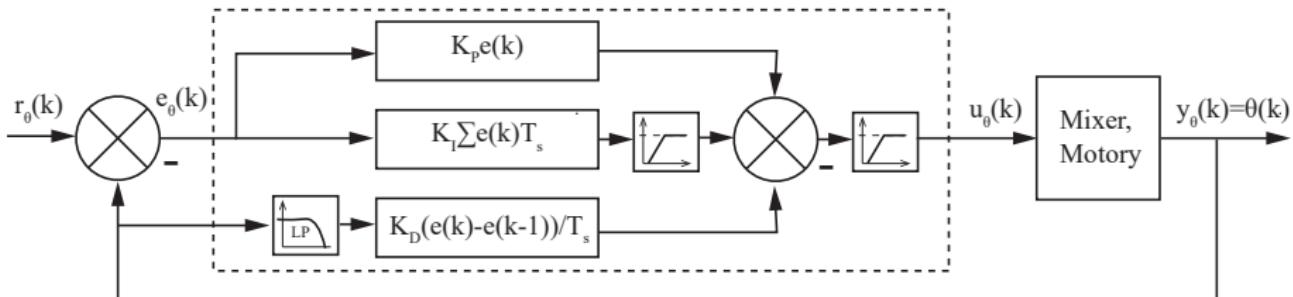
⁴ ArduCopter 50 Hz → 400 Hz [Hall 2018]

- Dron je stabilizovaný, pilot/ ROS náhle chce $r_\phi = +30^\circ$ klonenie. Čo sa stane s riadením?
- Žiadaná 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 žiadané hodnoty:
 - ▶ Pomalšie vzorkovanie na rýchlejšie (interpolácia) ⁴
 - ▶ Vyhľadenie filtráciou, saturácie
 - ▶ Ak hovoríme o manuálnom pilotovaní, tvarovanie určuje aký "pocit" je riadiť stroj

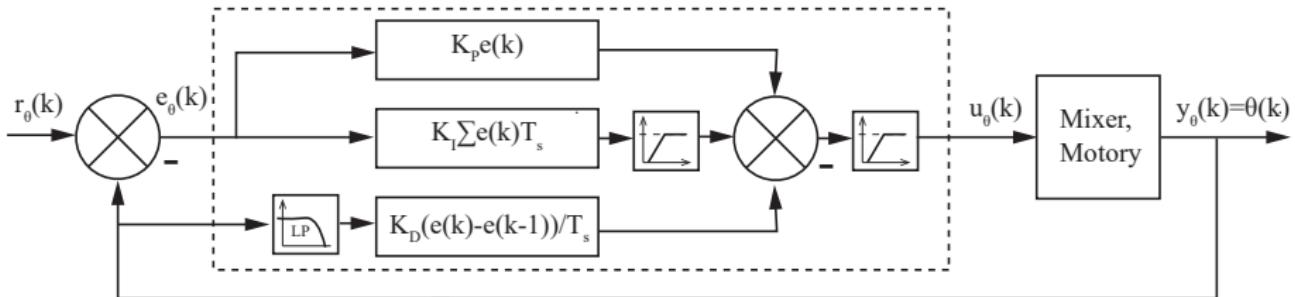


⁴ ArduCopter 50 Hz → 400 Hz [Hall 2018]

- Môžeme celkovo obísť zmenu žiadanej hodnoty a tým aj odchýlky $e_\theta(k)$ tak, že derivujeme výstup $y_\theta(k)$
- Riešenie preferuje PX4



- Môžeme celkovo obísť zmenu žiadanej hodnoty a tým aj odchýlky $e_\theta(k)$ tak, že derivujeme výstup $y_\theta(k)$
- Riešenie preferuje PX4



- Skvelý príklad v MATLAB/Simulink na didaktické účely
- Vyvinuté na základe prednášok na low-level riadenie dronov MIT [Karaman a kol. 2016]
- Nemá chutné vizualizácie, ale zase ľahšie zasahujete do riadenia (ako do ArduPilot, PX4 etc.)
- Má aj simulačnú časť ale FW sa dá nahrať aj na komerčné drony⁵



⁵Parrot Rolling Spider, Parrot Mambo — žiaľ sa už nevyrába

- Skvelý príklad v MATLAB/Simulink na didaktické účely
- Vyvinuté na základe prednášok na low-level riadenie dronov MIT [Karaman a kol. 2016]
- Nemá chutné vizualizácie, ale zase ľahšie zasahujete do riadenia (ako do ArduPilot, PX4 etc.)
- Má aj simulačnú časť ale FW sa dá nahrať aj na komerčné drony⁵



⁵Parrot Rolling Spider, Parrot Mambo — žiaľ sa už nevyrába

- Skvelý príklad v MATLAB/Simulink na didaktické účely
- Vyvinuté na základe prednášok na low-level riadenie dronov MIT [Karaman a kol. 2016]
- Nemá chutné vizualizácie, ale zase ľahšie zasahujete do riadenia (ako do ArduPilot, PX4 etc.)
- Má aj simulačnú časť ale FW sa dá nahrať aj na komerčné drony⁵

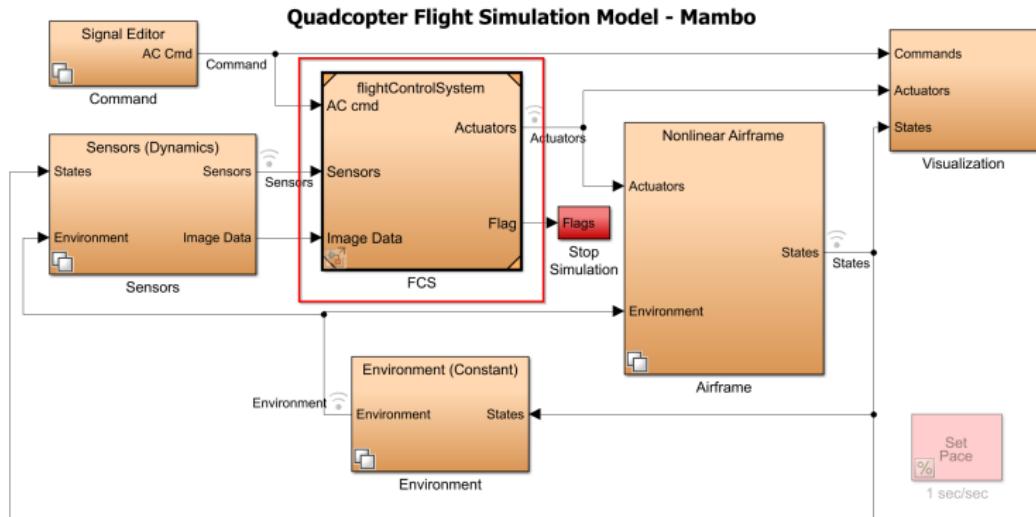


⁵Parrot Rolling Spider, Parrot Mambo — žiaľ sa už nevyrába

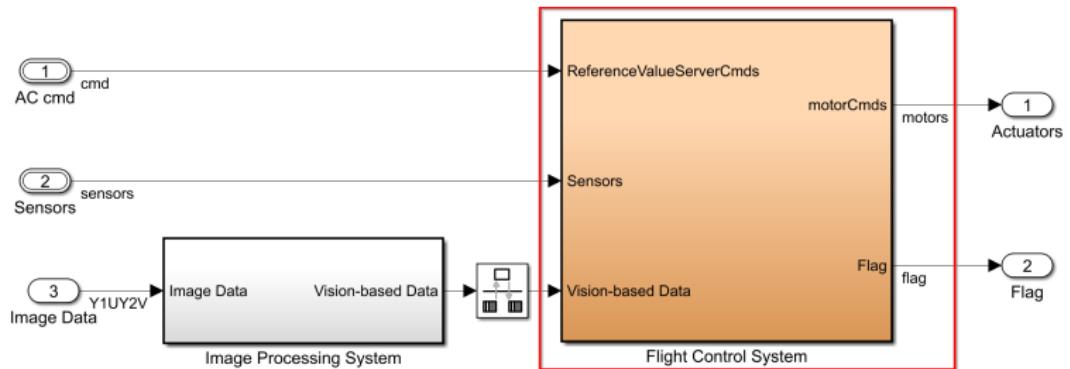
- Skvelý príklad v MATLAB/Simulink na didaktické účely
- Vyvinuté na základe prednášok na low-level riadenie dronov MIT [Karaman a kol. 2016]
- Nemá chutné vizualizácie, ale zase ľahšie zasahujete do riadenia (ako do ArduPilot, PX4 etc.)
- Má aj simulačnú časť ale FW sa dá nahráť aj na komerčné drony⁵

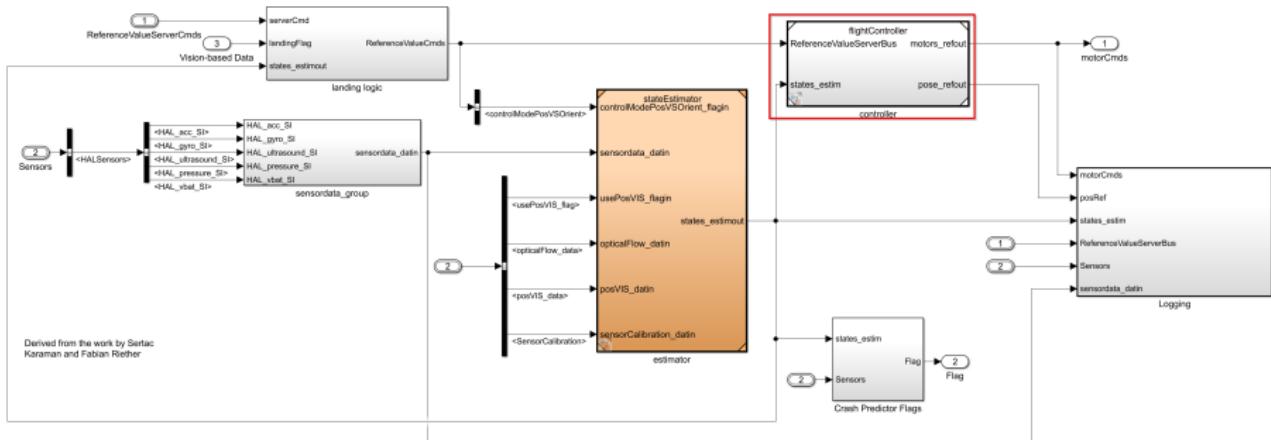


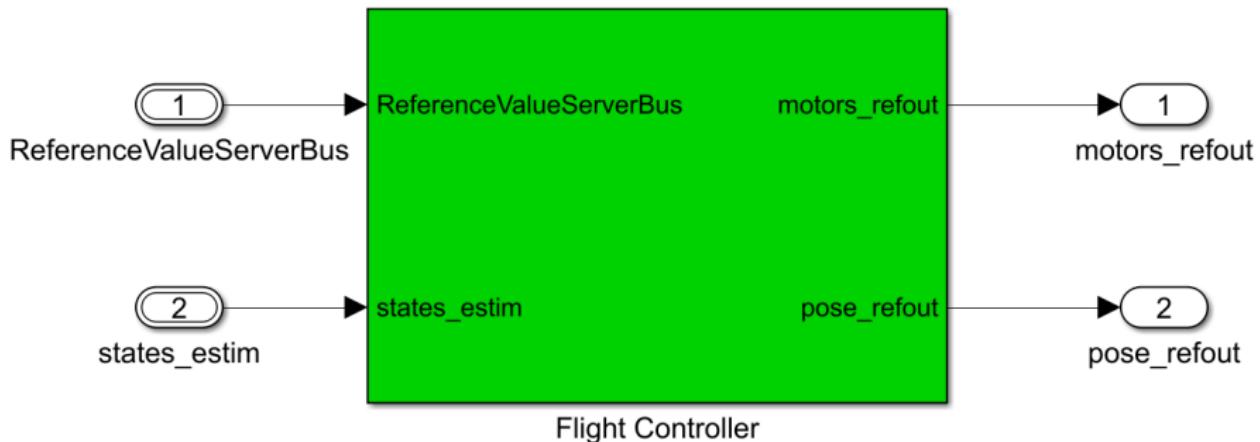
⁵Parrot Rolling Spider, Parrot Mambo — žiaľ sa už nevyrába

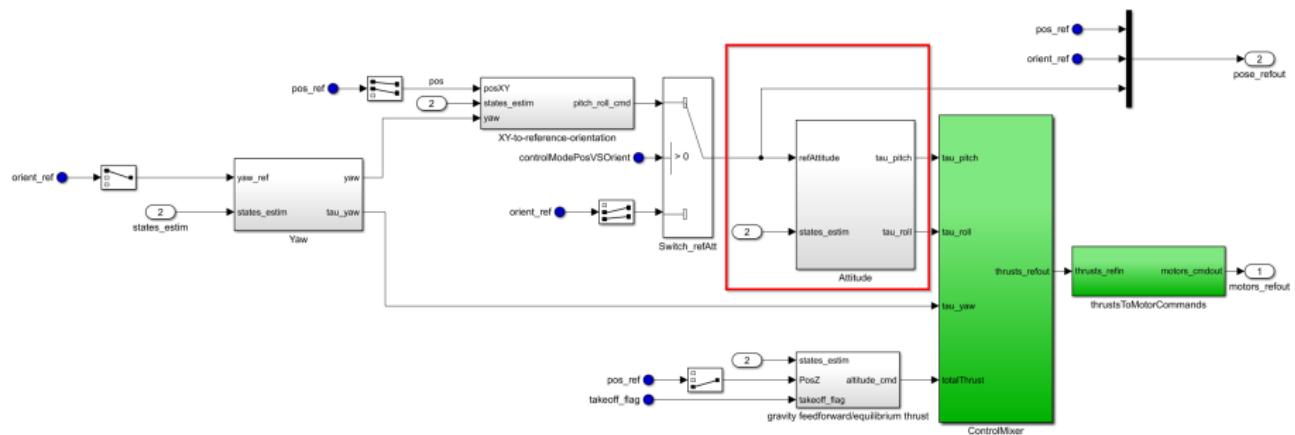


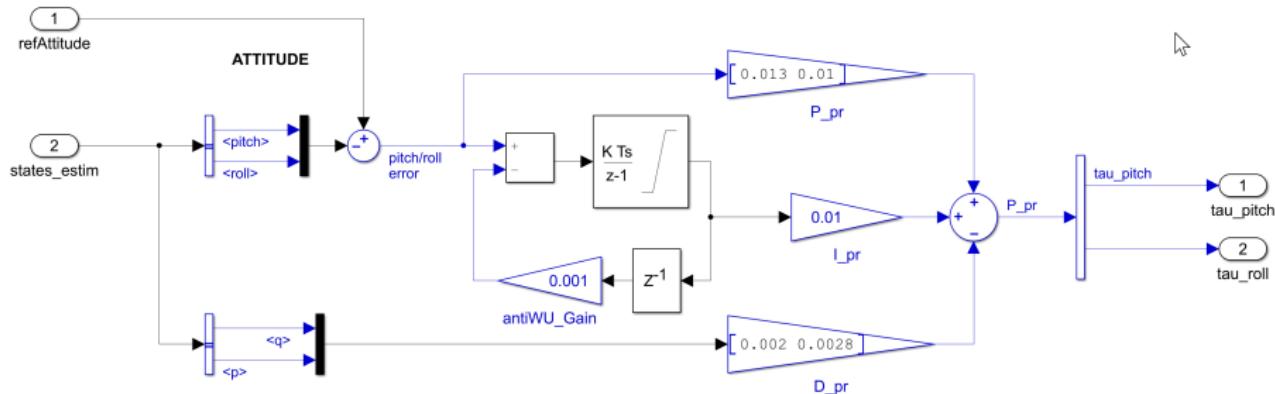
Copyright 2013-2018 The MathWorks, Inc.

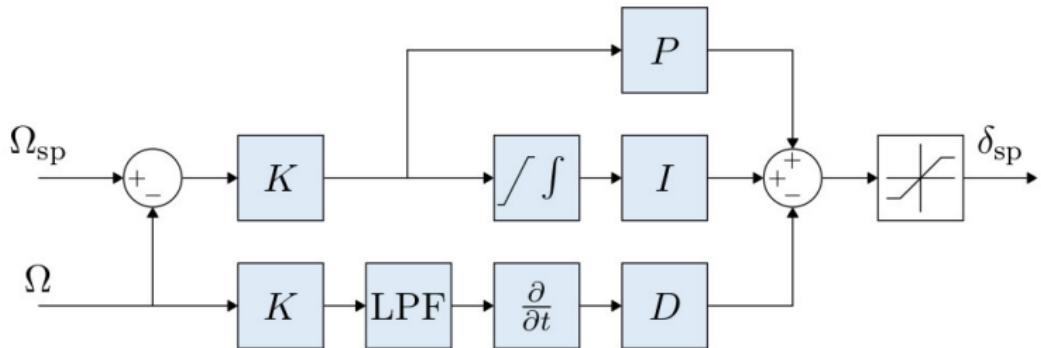












ArduCopter V2.9 STABILIZE Roll, Pitch & Yaw PID's

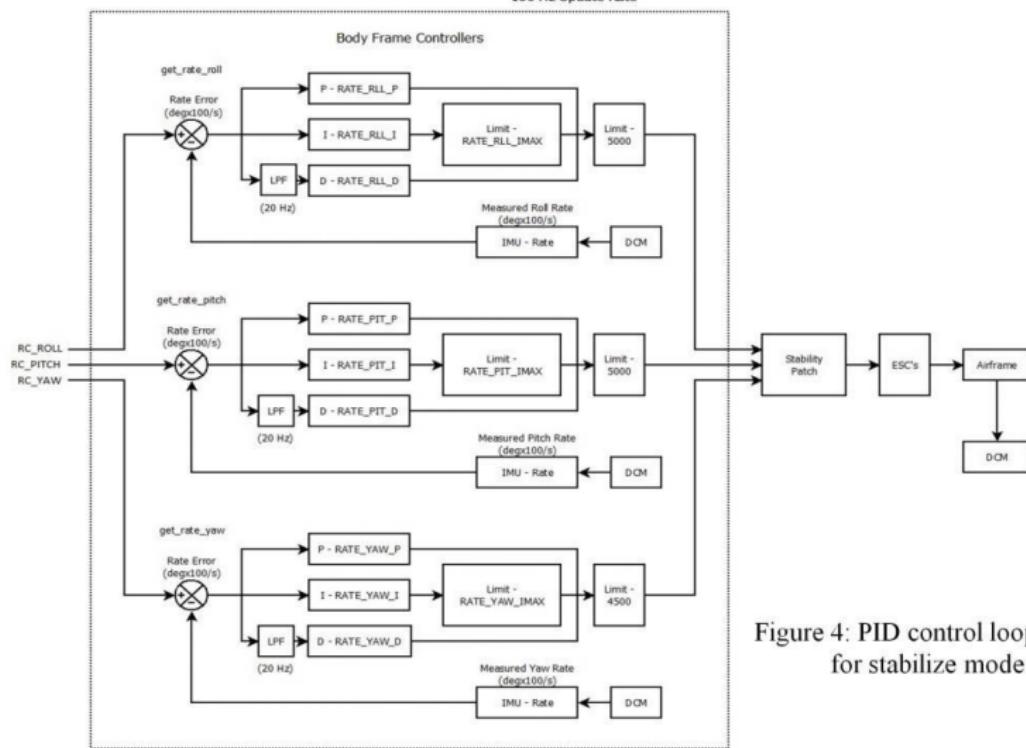


Figure 4: PID control loop for stabilize mode

- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v lokálnych súradničiach telesa (*angl.*: body frame)
- Pre RC dron by to aj stačilo s "plynom" (*angl.*: throttle) $h \rightarrow \tau$ na ovládanie ľahu (*angl.*: thrust) motorov⁶ [Boland 2015]
- Riadiť orientáciu (*angl.*: attitude) len na základe zmeny uhl. rýchlosťi (*angl.*: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow r_{\dot{\Theta}}$ ⁷
- Táto slučka je často nazvaná ako stabilizačné riadenie (*angl.*: stabilize) alebo orientačné riadenie (*angl.*: attitude control)⁸
- Nezabudnime na štvrtú kaskadovanú slučku: výška \rightarrow rýchlosť zmeny výšky resp. vertikálna rýchlosť (*angl.*: climb rate) \rightarrow zrýchlenie \rightarrow motor

⁶Síce rate-control je tiež možné [Boland 2015] !

⁷Naopak môžeme vyniechať rate controller, ako v MATLAB/Simulink príklade

⁸predošlá je (*angl.*: rate)

- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v lokálnych súradničiach telesa (*angl.*: body frame)
- Pre RC dron by to aj stačilo s “plynom” (*angl.*: throttle) $h \rightarrow \tau$ na ovládanie ľahu (*angl.*: thrust) motorov⁶ [Boland 2015]
- Riadiť orientáciu (*angl.*: attitude) len na základe zmeny uhl. rýchlosťi (*angl.*: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow r_{\dot{\Theta}}$ ⁷
- Táto slučka je často nazvaná ako stabilizačné riadenie (*angl.*: stabilize) alebo orientačné riadenie (*angl.*: attitude control)⁸
- Nezabudnime na štvrtú kaskadovanú slučku: výška \rightarrow rýchlosť zmeny výšky resp. vertikálna rýchlosť (*angl.*: climb rate) \rightarrow zrýchlenie \rightarrow motor

⁶Síce rate-control je tiež možné [Boland 2015] !

⁷Naopak môžeme vyniechať rate controller, ako v MATLAB/Simulink príklade

⁸predošlá je (*angl.*: rate)

- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v lokálnych súradničiach telesa (*angl.*: body frame)
- Pre RC dron by to aj stačilo s "plynom" (*angl.*: throttle) $h \rightarrow \tau$ na ovládanie ľahu (*angl.*: thrust) motorov⁶ [Boland 2015]
- Riadiť orientáciu (*angl.*: attitude) len na základe zmeny uhl. rýchlosťi (*angl.*: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow r_{\dot{\Theta}}$ ⁷
- Táto slučka je často nazvaná ako stabilizačné riadenie (*angl.*: stabilize) alebo orientačné riadenie (*angl.*: attitude control)⁸
- Nezabudnime na štvrtú kaskadovanú slučku: výška \rightarrow rýchlosť zmeny výšky resp. vertikálna rýchlosť (*angl.*: climb rate) \rightarrow zrýchlenie \rightarrow motor

⁶Síce rate-control je tiež možné [Boland 2015] !

⁷Naopak môžeme vyniechať rate controller, ako v MATLAB/Simulink príklade

⁸predošlá je (*angl.*: rate)

- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v lokálnych súradničiach telesa (*angl.*: body frame)
- Pre RC dron by to aj stačilo s "plynom" (*angl.*: throttle) $h \rightarrow \tau$ na ovládanie ľahu (*angl.*: thrust) motorov⁶ [Boland 2015]
- Riadiť orientáciu (*angl.*: attitude) len na základe zmeny uhl. rýchlosťi (*angl.*: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow r_{\dot{\Theta}}$ ⁷
- Táto slučka je často nazvaná ako stabilizačné riadenie (*angl.*: stabilize) alebo orientačné riadenie (*angl.*: attitude control)⁸
- Nezabudnime na štvrtú kaskadovanú slučku: výška \rightarrow rýchlosť zmeny výšky resp. vertikálna rýchlosť (*angl.*: climb rate) \rightarrow zrýchlenie \rightarrow motor

⁶Síce rate-control je tiež možné [Boland 2015] !

⁷Naopak môžeme vyniechať rate controller, ako v MATLAB/Simulink príklade

⁸predošlá je (*angl.*: rate)

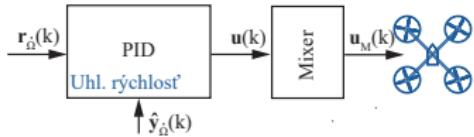
- riadenie orientácie, tj. uhly $\Omega = \Phi, \Theta, \Sigma$ a uhlové rýchlosťi $\dot{\Omega}$ v lokálnych súradničiach telesa (*angl.*: body frame)
- Pre RC dron by to aj stačilo s "plynom" (*angl.*: throttle) $h \rightarrow \tau$ na ovládanie ľahu (*angl.*: thrust) motorov⁶ [Boland 2015]
- Riadiť orientáciu (*angl.*: attitude) len na základe zmeny uhl. rýchlosťi (*angl.*: rate) by bolo dosť neintuitívne, potrebujeme prepočítať $r_\Theta \rightarrow r_{\dot{\Theta}}$ ⁷
- Táto slučka je často nazvaná ako stabilizačné riadenie (*angl.*: stabilize) alebo orientačné riadenie (*angl.*: attitude control)⁸
- Nezabudnime na štvrtú kaskadovanú slučku: výška \rightarrow rýchlosť zmeny výšky resp. vertikálna rýchlosť (*angl.*: climb rate) \rightarrow zrýchlenie \rightarrow motor

⁶Síce rate-control je tiež možné [Boland 2015] !

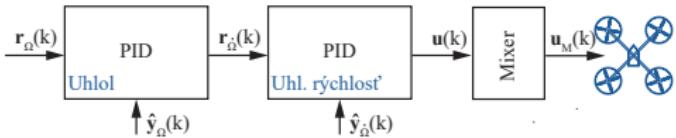
⁷Naopak môžeme vyniechať rate controller, ako v MATLAB/Simulink príklade

⁸predošlá je (*angl.*: rate)

- 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).

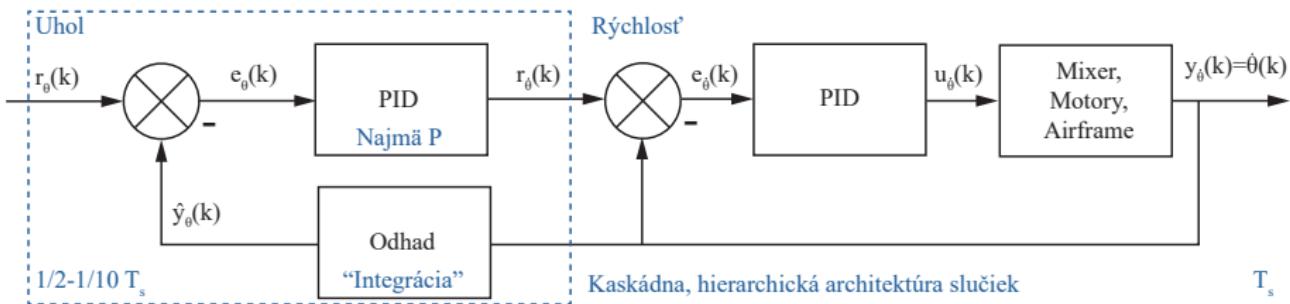


- Nadradené slučky sú pomalšie, vytvára to istý "filter", t.j. nemôžeme rýchlejšie ovládať rýchlosť ako polohu (cca. o rád, min polovicu pomalšie⁹) [Hall 2018; PX4 Autopilot 2021a]
- Pri PID skôr P¹⁰, lebo D reaguje príliš agresívne na šum.
- Do slučky dopracujeme doprednú väzbu (*angl.*: feedforward) ktorá zrýchli odozvu regulácie. "Whatever works" - netreba mystifikovať.

⁹ArduCopter 40 Hz vs 400 Hz, PX4 250 vs. 1000 Hz [Hall 2018; PX4 Autopilot 2021a]

¹⁰ArduCopter a PX4 Autopilot používa P regulátor [PX4 Autopilot 2021a; ArduPilot 2021]

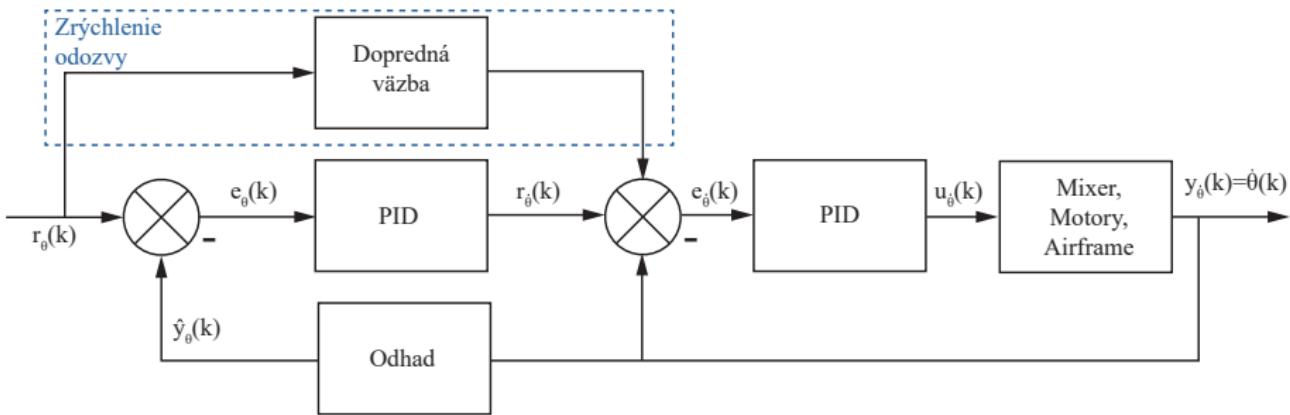
- Nadradené slučky sú pomalšie, vytvára to istý "filter", t.j. nemôžeme rýchlejšie ovládať rýchlosť ako polohu (cca. o rád, min polovicu pomalšie⁹) [Hall 2018; PX4 Autopilot 2021a]
- Pri PID skôr P¹⁰, lebo D reaguje príliš agresívne na šum.
- Do slučky dopracujeme doprednú väzbu (*angl.*: feedforward) ktorá zrýchli odozvu regulácie. "Whatever works" - netreba mystifikovať.



⁹ArduCopter 40 Hz vs 400 Hz, PX4 250 vs. 1000 Hz [Hall 2018; PX4 Autopilot 2021a]

¹⁰ArduCopter 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 pomalšie⁹) [Hall 2018; PX4 Autopilot 2021a]
- Pri PID skôr P¹⁰, lebo D reaguje príliš agresívne na šum.
- Do slučky dopracujeme doprednú väzbu (*angl.*: feedforward) ktorá zrýchli odozvu regulácie. "Whatever works" - netreba mystifikovať.

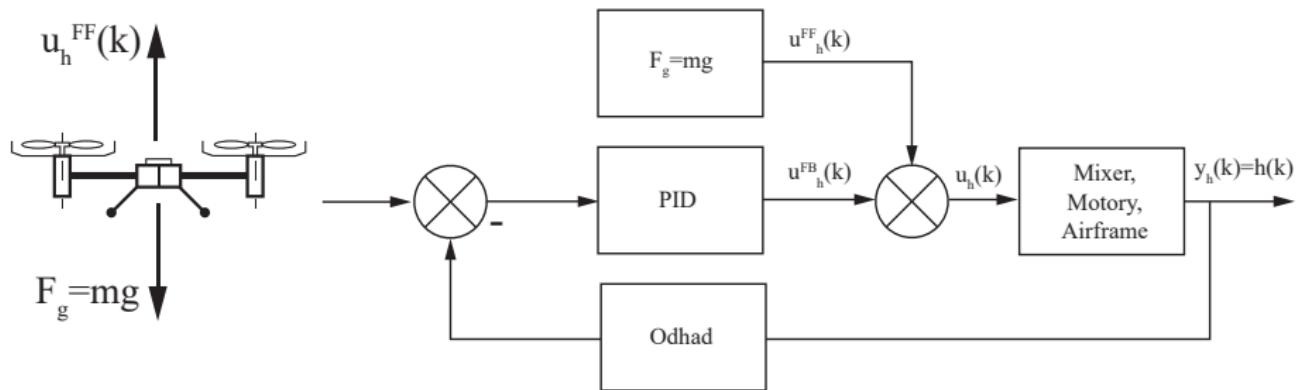


⁹ArduCopter 40 Hz vs 400 Hz, PX4 250 vs. 1000 Hz [Hall 2018; PX4 Autopilot 2021a]

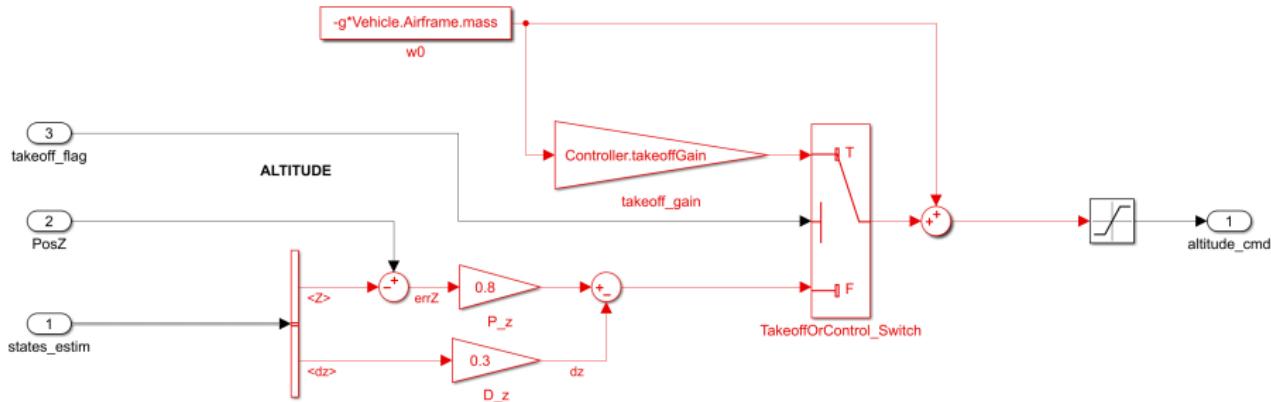
¹⁰ArduCopter a PX4 Autopilot používa P regulátor [PX4 Autopilot 2021a; ArduPilot 2021]

- Dopredná väzba - keď poznáme stálu časť akčného zásahu, prečo to rovno nepoužívať?
- Napr. pri stabilizácii letovej výšky máme $u_h = mg$ a zvyšok rieši regulátor
- Didaktický príklad v Simulinku uvažuje PD regulátor výšky + feedforward

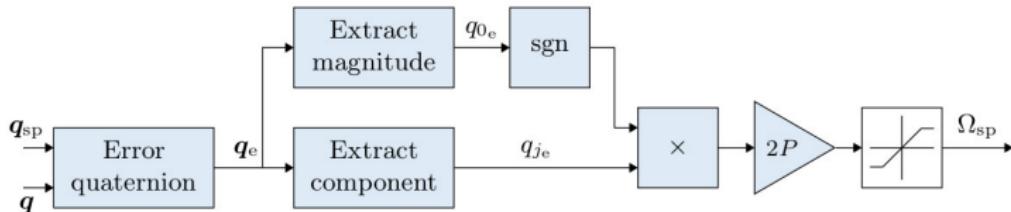
- Dopredná väzba - keď poznáme stálu časť akčného zásahu, prečo to rovno nepoužívať?
- Napr. pri stabilizácii letovej výšky máme $u_h = mg$ a zvyšok rieši regulátor
- Didaktický príklad v Simulinku uvažuje PD regulátor výšky + feedforward



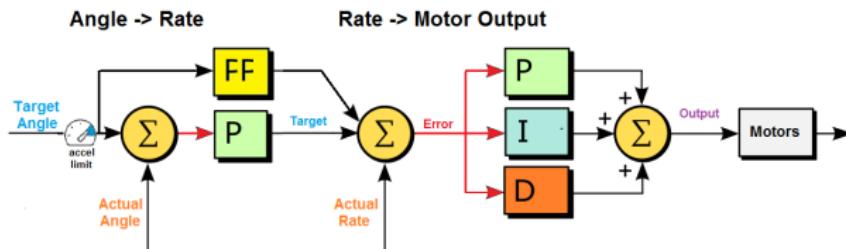
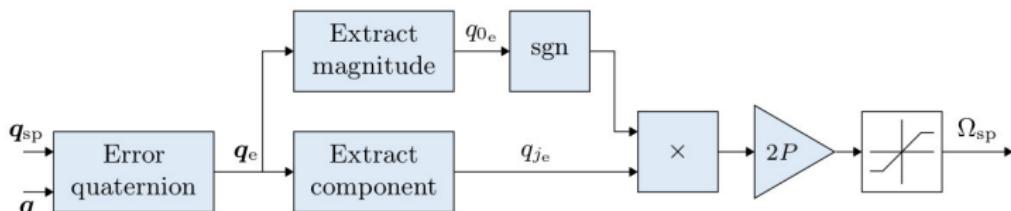
- Dopredná väzba - keď poznáme stálu časť akčného zásahu, prečo to rovno nepoužívať?
- Napr. pri stabilizácii letovej výšky máme $u_h = mg$ a zvyšok rieši regulátor
- Didaktický príklad v Simulinku uvažuje PD regulátor výšky + feedforward



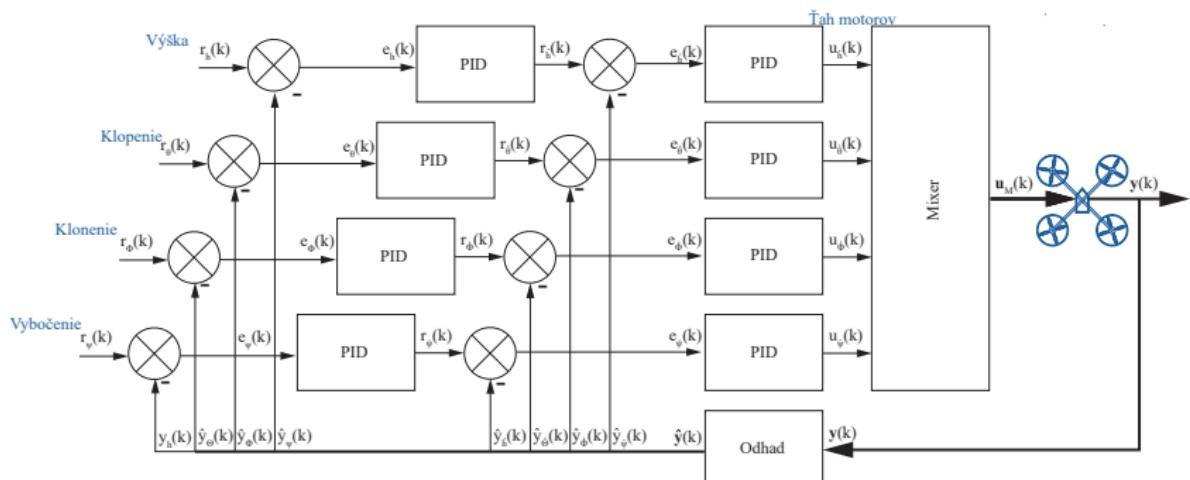
- PX4 používa kvaternióny (pár slov o tom neskôršie), ale je to iba P regulátor
- ArduCopter v podstate taktiež tam má P regulátor + FF



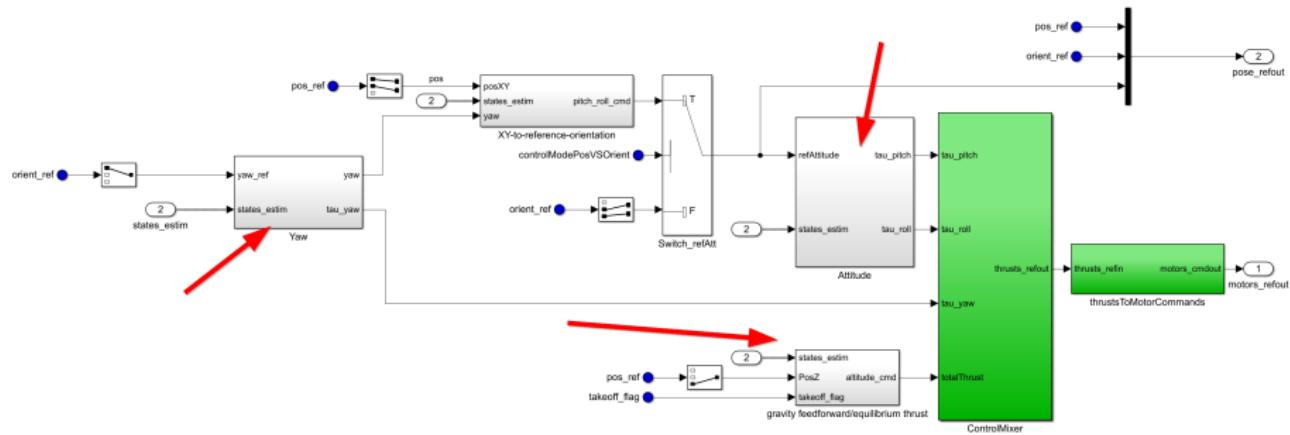
- PX4 používa kvaternióny (pár slov o tom neskôršie), ale je to iba P regulátor
- ArduCopter v podstate taktiež tam má P regulátor + FF



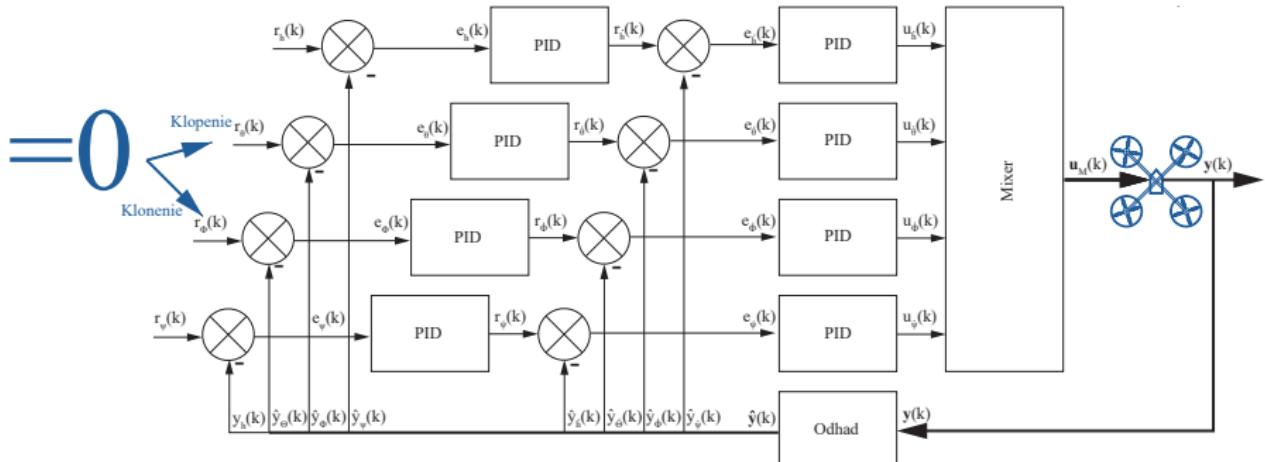
- Máme 4 nezávislých slučiek
 - Ako to vyzerá v MATLAB príklade?
 - Čo sa stane ak chceme dron stabilizovať? Majme 0-vé Eulerove uhly!
 - Určite to drží na jednom mieste?
 - Potrebujeme nenulové klopenie a klonenie, a referenciu na základe polohy!



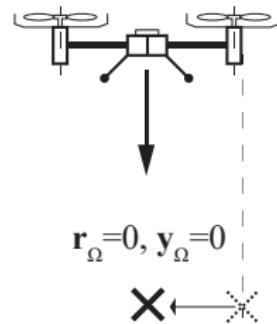
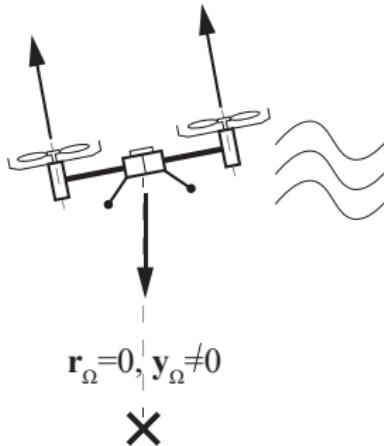
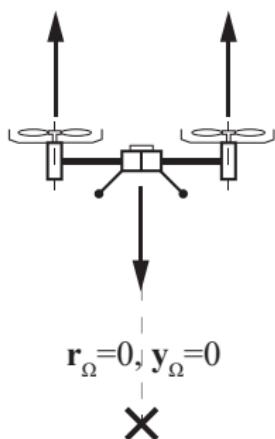
- Máme 4 nezávislých slučiek
- Ako to vyzerá v MATLAB príklade?
- Čo sa stane ak chceme dron stabilizovať? Majme 0-vé Eulerove uhly!
- Určite to drží na jednom mieste?
- Potrebujeme nenulové klopenie a klonenie, a referenciu na základe polohy!



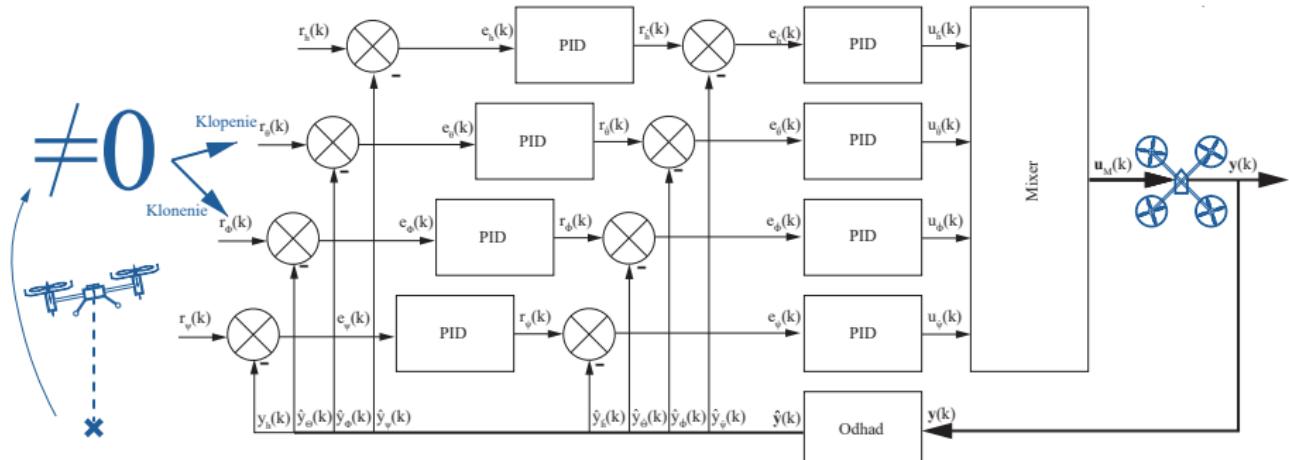
- Máme 4 nezávislých slučiek
- Ako to vyzerá v MATLAB príklade?
- Čo sa stane ak chceme dron stabilizovať? Majme 0-vé Eulerove uhly!
- Určite to drží na jednom mieste?
- Potrebujeme nenulové klopenie a klonenie, a referenciu na základe polohy!



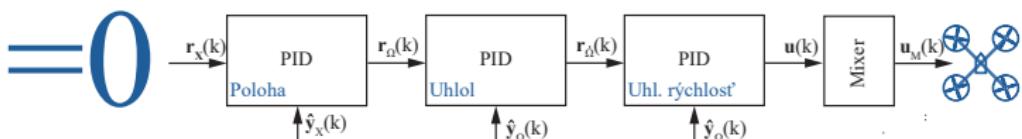
- Máme 4 nezávislých slučiek
- Ako to vyzerá v MATLAB príklade?
- Čo sa stane ak chceme dron stabilizovať? Majme 0-vé Eulerove uhly!
- Určite to drží na jednom mieste?
- Potrebujeme nenulové klopenie a klonenie, a referenciu na základe polohy!



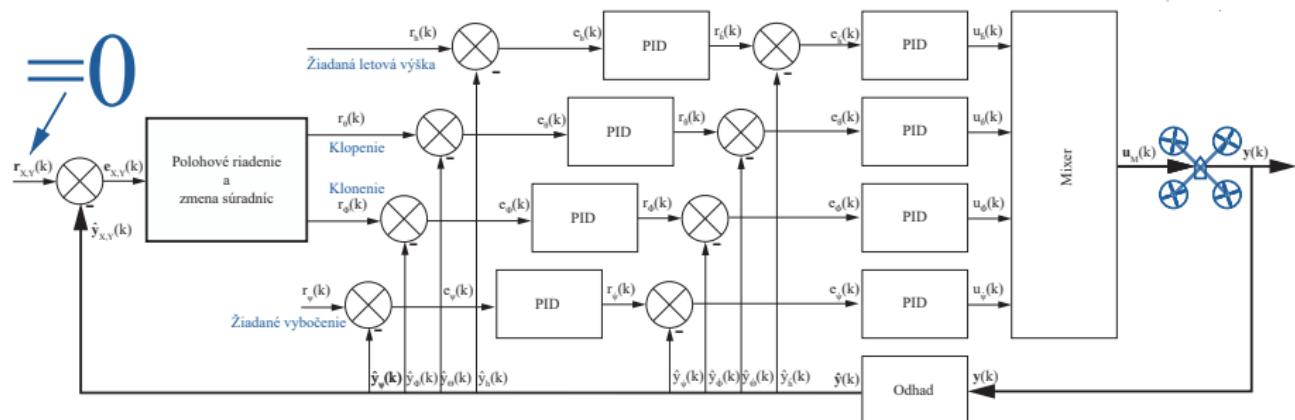
- Máme 4 nezávislých slučiek
- Ako to vyzerá v MATLAB príklade?
- Čo sa stane ak chceme dron stabilizovať? Majme 0-vé Eulerove uhly!
- Určite to drží na jednom mieste?
- Potrebujeme nenulové klopenie a klonenie, a referenciu na základe polohy!



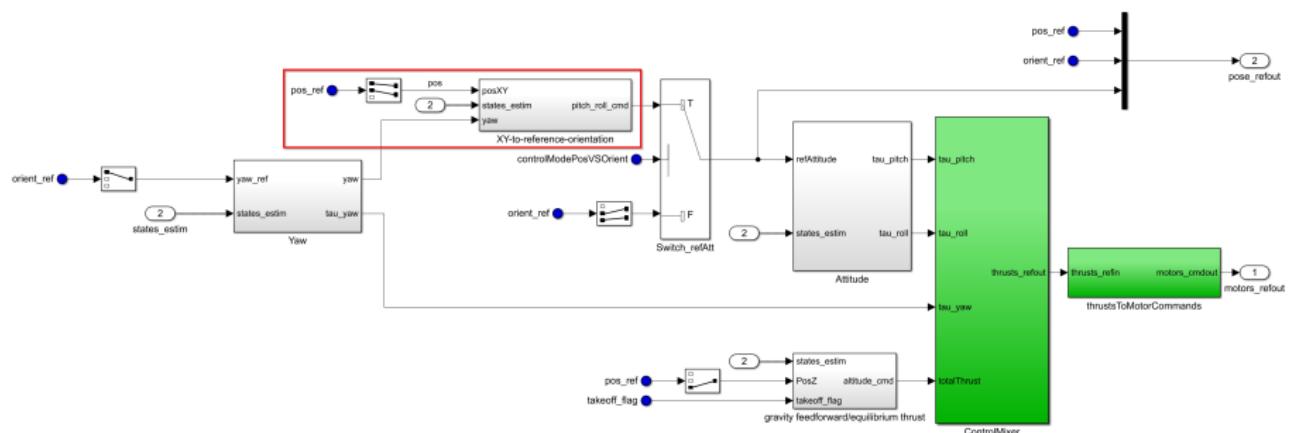
- Úlohou je držať polohu na základe globálnych súradníc (GS) napr. GPS. Pri minimalistickom prevedení (po zmene súradníc!) môžeme priamo premeniť polohu na uhol
- Ako by vyzeral náš kaskádovaný riadiaci systém?
- Môže to tak priamo fungovať?



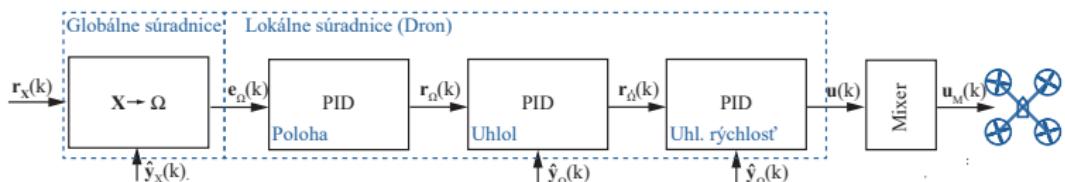
- Úlohou je držať polohu na základe globálnych súradníc (GS) napr. GPS. Pri minimalistickom prevedení (po zmene súradníc!) môžeme priamo premeniť polohu na uhol
- Ako by vyzeral náš kaskádovaný riadiaci systém?
- Môže to tak priamo fungovať?



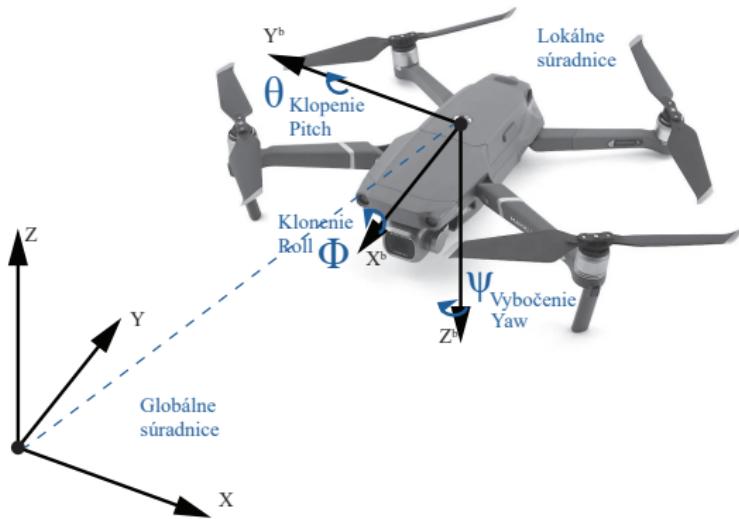
- Úlohou je držať polohu na základe globálnych súradníc (GS) napr. GPS. Pri minimalistickom prevedení (po zmene súradníc!) môžeme priamo premeniť polohu na uhol
- Ako by vyzeral náš kaskádovaný riadiaci systém?
- Môže to tak priamo fungovať?



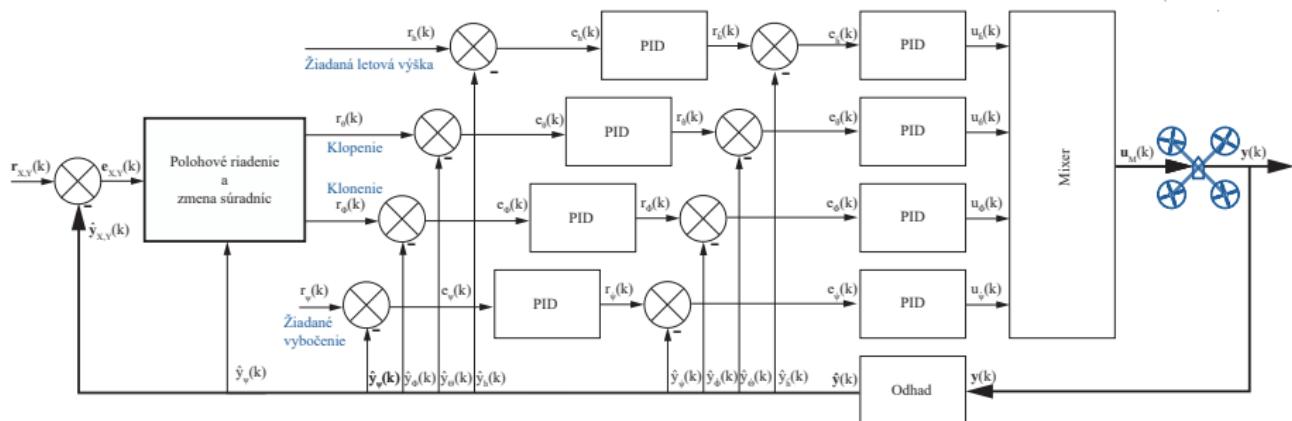
- Aj pri priamom riadení polohy musíme zmeniť súradnice
- Klopenie a klonenie určíme na základe GS súradníc, t.j. premeníme $X, Y \rightarrow \Theta, \Phi\dots$
- ... ale potrebujeme na výpočet aj želané/aktuálne vybočenie, t.j. $X, Y, \Psi \rightarrow \Theta, \Phi,$



- Aj pri priamom riadení polohy musíme zmeniť súradnice
- Klopenie a klonenie určíme na základe GS súradníc, t.j. premeníme $X, Y \rightarrow \Theta, \Phi\dots$
- ... ale potrebujeme na výpočet aj želané/aktuálne vybočenie, t.j. $X, Y, \Psi \rightarrow \Theta, \Phi,$



- Aj pri priamom riadení polohy musíme zmeniť súradnice
- Klopenie a klonenie určíme na základe GS súradníc, t.j. premeníme $X, Y \rightarrow \Theta, \Phi\dots$
- ... ale potrebujeme na výpočet aj želané/aktuálne vybočenie, t.j. $X, Y, \Psi \rightarrow \Theta, \Phi,$



- Môžeme napr. používať trigonometriu, teda rotačné matice (*angl.*: direction cosine matrix, DCM) na premenu z globálnych (G) do lokálnych (L) súradníc
- Najjednoduchší ale funkčný príklad, stabilizácia klonenia a klopenia. Majme odchýlku polohy $e_{X,Y}^G = [X, Y]^T$ a aktuálne vybočenie Ψ , potom [Ben-Ari 2017]

$$e_{\Theta,\Phi}^L = \begin{bmatrix} \cos \Psi & -\sin \Psi \\ \sin \Psi & \cos \Psi \end{bmatrix} e_{X,Y}^G \quad (10)$$

- V príklade v MATLAB/Simulink je to prakticky identické

- Môžeme napr. používať trigonometriu, teda rotačné matice (*angl.*: direction cosine matrix, DCM) na premenu z globálnych (G) do lokálnych (L) súradníc
- Najjednoduchší ale funkčný príklad, stabilizácia klonenia a klopenia. Majme odchýlku polohy $\mathbf{e}_{X,Y}^G = [X, Y]^T$ a aktuálne vybočenie Ψ , potom [Ben-Ari 2017]

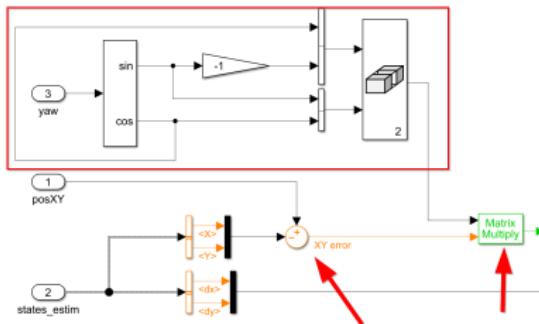
$$\mathbf{e}_{\Theta, \Phi}^L = \begin{bmatrix} \cos \Psi & -\sin \Psi \\ \sin \Psi & \cos \Psi \end{bmatrix} \mathbf{e}_{X,Y}^G \quad (10)$$

- V príklade v MATLAB/Simulink je to prakticky identické

- Môžeme napr. používať trigonometriu, teda rotačné matice (*angl.: direction cosine matrix, DCM*) na premenu z globálnych (G) do lokálnych (L) súradníc
- Najjednoduchší ale funkčný príklad, stabilizácia klonenia a klopenia. Majme odchýlku polohy $e_{X,Y}^G = [X, Y]^T$ a aktuálne vybočenie Ψ , potom [\[Ben-Ari 2017\]](#)

$$e_{\Theta, \Phi}^L = \begin{bmatrix} \cos \Psi & -\sin \Psi \\ \sin \Psi & \cos \Psi \end{bmatrix} e_{X,Y}^G \quad (10)$$

- V príklade v MATLAB/Simulink je to prakticky identické

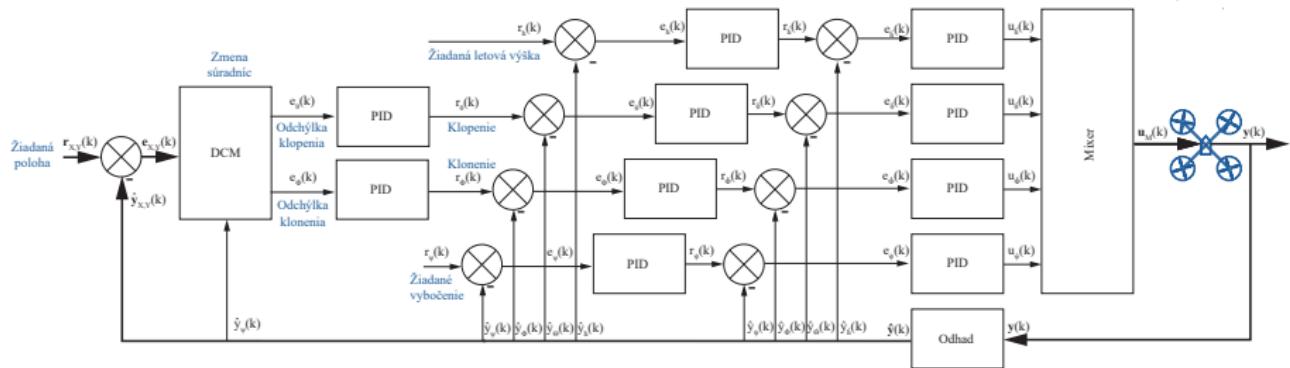


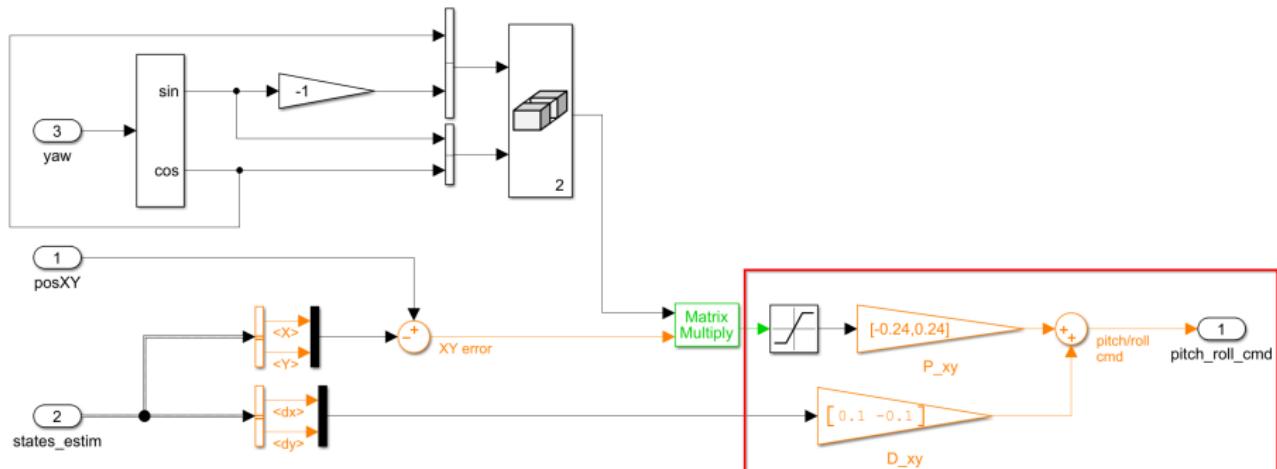
- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [Ben-Ari 2017]
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [Ben-Ari 2017]
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny

- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [Ben-Ari 2017]
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [Ben-Ari 2017]
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny

- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [Ben-Ari 2017]
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [Ben-Ari 2017]
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny

- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [\[Ben-Ari 2017\]](#)
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [\[Ben-Ari 2017\]](#)
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny

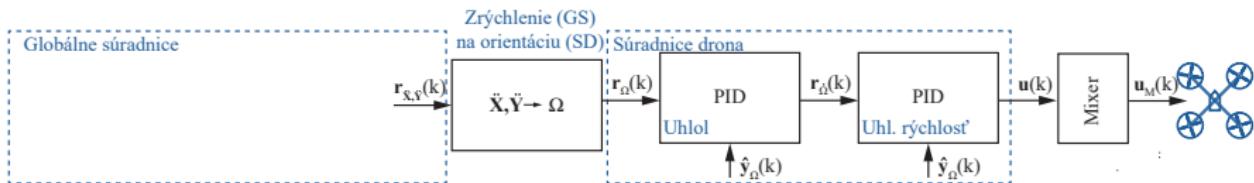




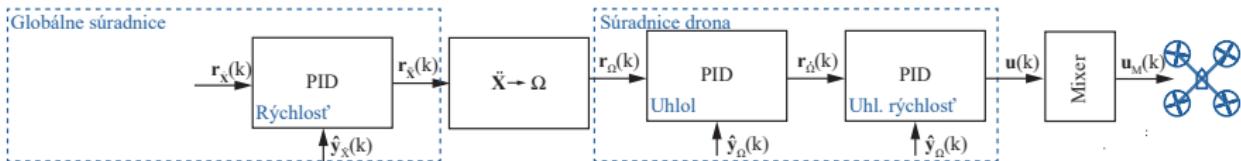
- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto $X, Y \rightarrow \Theta, \Phi$ máme $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).

Nepriame riadenie polohy: Riadenie rýchlosťi

- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto $X, Y \rightarrow \Theta, \Phi$ máme $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).

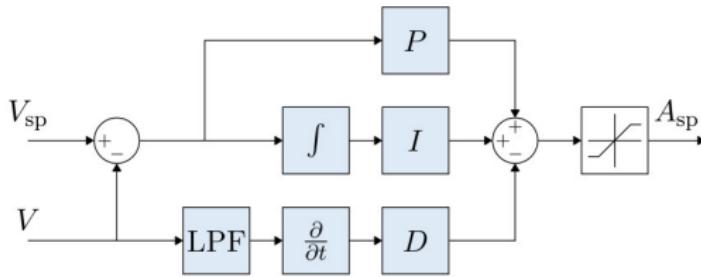


- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto $X, Y \rightarrow \Theta, \Phi$ máme $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).



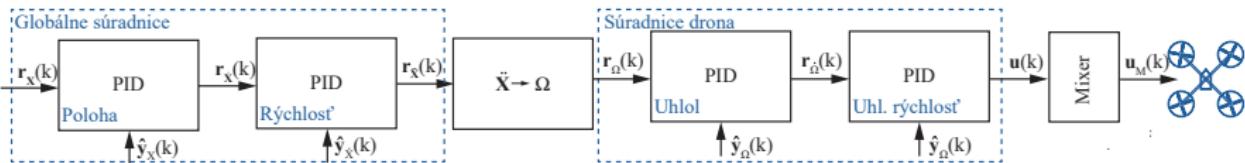
Nepriame riadenie polohy: Riadenie rýchlosťi

- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto $X, Y \rightarrow \Theta, \Phi$ máme $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. [Saha 2020](#)).

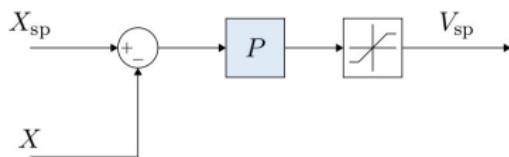
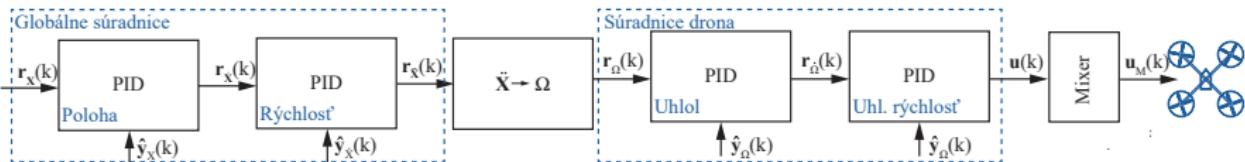


Nepriame riadenie polohy: Polohová slučka

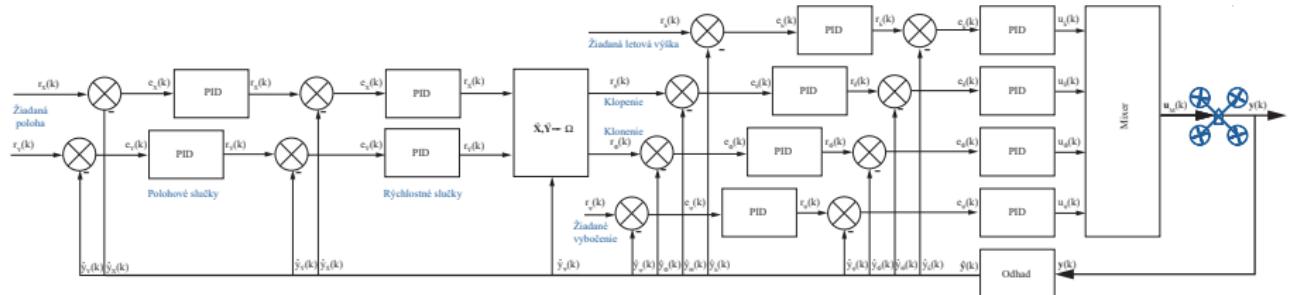
- Potrebujeme ešte jednu slučku — polohovú, ktorá premení polohu na rýchlosť
- Väčšinou je to P riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).

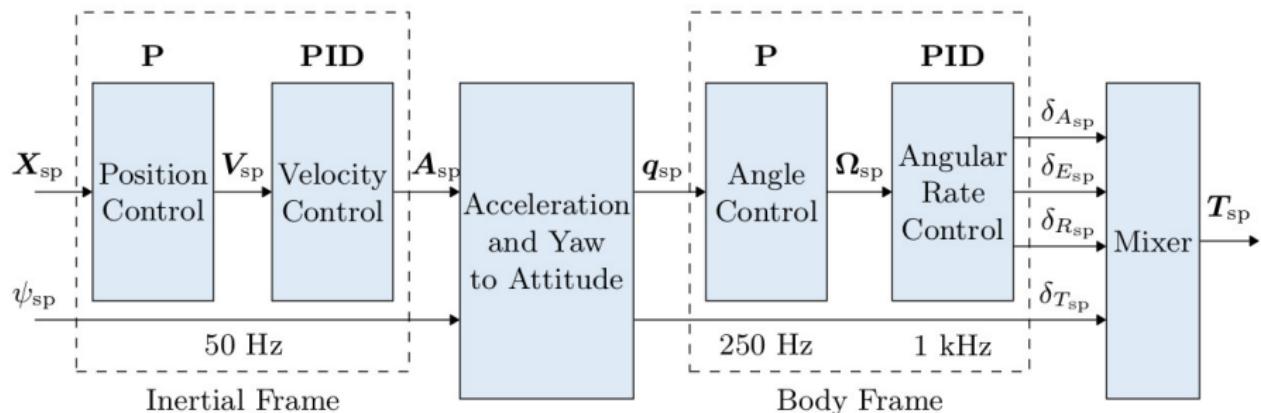


- Potrebujeme ešte jednu slučku — polohovú, ktorá premení polohu na rýchlosť
- Väčšinou je to P riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).



Celková riadiaca architektúra

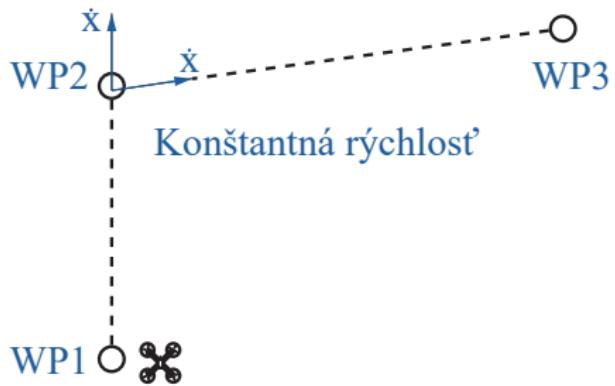




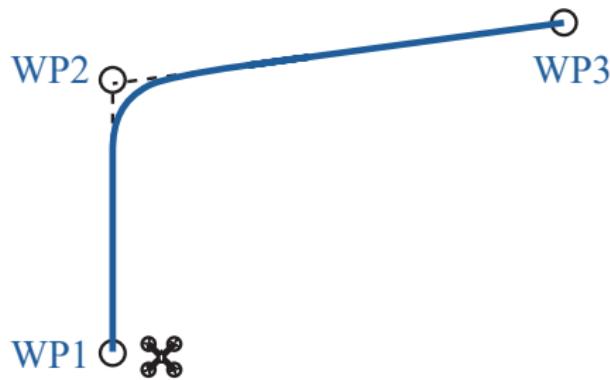
- Majme trasu WP1 do WP2, všetko je v poriadku.
- Pridajme WP3 a rozmýšľajme čo sa deje pri WP2. Je možné preletieť nad WP2?
- Bud' musíme úplne sa zastaviť alebo nemôžeme priamo preletieť — ináč by sme potrebovali nekonečne veľké zrýchlenia



- 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



- Napr. v Mission Planner nájdeme všetky Eulerove uhly + výšku, polohu, rýchlosť



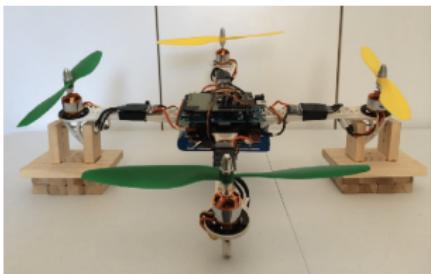
- Heuristika¹¹ vs. pseudo-heuristický prístup: Ziegler-Nicholsova (ZN) metóda: [Ziegler a kol. 1942]
- Nastavme K_I , K_D zložky nulové a zvyšujme K_P , kým nedostaneme na hranu stability s konzistentnými osciláciami. Tam sme našli $K_{P\max}$ a períoda oscilácií bude T_{\max} .
- Opakujeme pre každú slučku, poprípade pre každú os (dá sa aj priviazať dron)
- Na základe týchto hodnôt môžeme z tabuľky na základe pseudoheuristiky vybrať ladenie.

¹¹Metóda FAFO, z (angl.: fuck around, find out)

- Heuristika¹¹ vs. pseudo-heuristický prístup: Ziegler-Nicholsova (ZN) metóda: [Ziegler a kol. 1942]
- Nastavme K_I , K_D zložky nulové a zvyšujme K_P , kým nedostaneme na hranu stability s konzistentnými osciláciami. Tam sme našli $K_{P\max}$ a períoda oscilácií bude T_{\max} .
- Opakujeme pre každú slučku, poprípade pre každú os (dá sa aj priviazať dron)
- Na základe týchto hodnôt môžeme z tabuľky na základe pseudoheuristiky vybrať ladenie.

¹¹Metóda FAFO, z (angl.: fuck around, find out)

- Heuristika¹¹ vs. pseudo-heuristický prístup: Ziegler-Nicholsova (ZN) metóda: [Ziegler a kol. 1942]
- Nastavme K_I , K_D zložky nulové a zvyšujme K_P , kým nedostaneme na hranu stability s konzistentnými osciláciami. Tam sme našli $K_{P\max}$ a períoda oscilácií bude T_{\max} .
- Opakujeme pre každú slučku, poprípade pre každú os (dá sa aj priviazať dron)
- Na základe týchto hodnôt môžeme z tabuľky na základe pseudoheuristiky vybrať ladenie.



¹¹Metóda FAFO, z (angl.: fuck around, find out)

- Heuristika¹¹ vs. pseudo-heuristický prístup: Ziegler-Nicholsova (ZN) metóda: [Ziegler a kol. 1942]
- Nastavme K_I , K_D zložky nulové a zvyšujme K_P , kým nedostaneme na hranu stability s konzistentnými osciláciami. Tam sme našli $K_{P\max}$ a períoda oscilácií bude T_{\max} .
- Opakujeme pre každú slučku, poprípade pre každú os (dá sa aj priviazať dron)
- Na základe týchto hodnôt môžeme z tabuľky na základe pseudoheuristiky vybrať ladenie.

	K_P	T_I	T_D	K_I	K_D
P	$0.5K_{P\max}$	-	-	-	-
PI	$0.45K_{P\max}$	$0.80T_{\max}$	-	$0.54K_{P\max}/T_{\max}$	
PD	$0.8K_{P\max}$	-	$0.125T_{\max}$	-	$0.10K_{P\max}$
PID	$0.6K_{P\max}$	$0.5T_{\max}$	$0.125T_{\max}$	$1.2K_{P\max}/T_{\max}$	$0.075K_{P\max}$

¹¹Metóda FAFO, z (angl.: fuck around, find out)

- **Výhody:** Jednoduché
- **Nevýhody:** tlačí to quad na hranu stability, určite nebude “optimálne” pre všetky prípady a od 1942 máme lepšie metódy. Nastavené na maximálne odstránenie porúch, náchylné na agresívne riadenie s prestrelom Bequette 2010
- Môžeme podobné experimentálne dáta získať aj zo skokovej odozvy
- ArduPilot (ArduCopter) — “AutoTuning” Napodobňuje túto heuristiku (ale nie je to ZN) [Tridgell 2021]
 - ▶ Zvyšuje P parameter pomaly
 - ▶ Akonáhle deteguje osciláciu, zníži P parameter
 - ▶ Opakuje pre iné zložky/slučky
- PX4 dlho nemal vstavanú “autotune” funkciu [Bolton a kol. 2018] , momentálne využíva na odhad SISO linearizovaný model 2. rádu (bez prepojenia medzi osami) [PX4 Autopilot 2021b]

- **Výhody:** Jednoduché
- **Nevýhody:** tlačí to quad na hranu stability, určite nebude “optimálne” pre všetky prípady a od 1942 máme lepšie metódy. Nastavené na maximálne odstránenie porúch, náchylné na agresívne riadenie s prestrelom [Bequette 2010](#)
- Môžeme podobné experimentálne dáta získať aj zo skokovej odozvy
- ArduPilot (ArduCopter) — “AutoTuning” Napodobňuje túto heuristiku (ale nie je to ZN) [\[Tridgell 2021\]](#)
 - ▶ Zvyšuje P parameter pomaly
 - ▶ Akonáhle deteguje osciláciu, zníži P parameter
 - ▶ Opakuje pre iné zložky/slučky
- PX4 dlho nemal vstavanú “autotune” funkciu [\[Bolton a kol. 2018\]](#), momentálne využíva na odhad SISO linearizovaný model 2. rádu (bez prepojenia medzi osami) [\[PX4 Autopilot 2021b\]](#)

- **Výhody:** Jednoduché
- **Nevýhody:** tlačí to quad na hranu stability, určite nebude “optimálne” pre všetky prípady a od 1942 máme lepšie metódy. Nastavené na maximálne odstránenie porúch, náchylné na agresívne riadenie s prestrelom [Bequette 2010](#)
- Môžeme podobné experimentálne dáta získať aj zo skokovej odozvy
- ArduPilot (ArduCopter) — “AutoTuning” Napodobňuje túto heuristiku (ale nie je to ZN) [\[Tridgell 2021\]](#)
 - ▶ Zvyšuje P parameter pomaly
 - ▶ Akonáhle deteguje osciláciu, zníži P parameter
 - ▶ Opakuje pre iné zložky/slučky
- PX4 dlho nemal vstavanú “autotune” funkciu [\[Bolton a kol. 2018\]](#), momentálne využíva na odhad SISO linearizovaný model 2. rádu (bez prepojenia medzi osami) [\[PX4 Autopilot 2021b\]](#)

- **Výhody:** Jednoduché
- **Nevýhody:** tlačí to quad na hranu stability, určite nebude “optimálne” pre všetky prípady a od 1942 máme lepšie metódy. Nastavené na maximálne odstránenie porúch, náchylné na agresívne riadenie s prestrelom [Bequette 2010](#)
- Môžeme podobné experimentálne dáta získať aj zo skokovej odozvy
- ArduPilot (ArduCopter) — “AutoTuning” Napodobňuje túto heuristiku (ale nie je to ZN) [\[Tridgell 2021\]](#)
 - ▶ Zvyšuje P parameter pomaly
 - ▶ Akonáhle deteguje osciláciu, zníži P parameter
 - ▶ Opakuje pre iné zložky/slučky
- PX4 dlho nemal vstavanú “autotune” funkciu [\[Bolton a kol. 2018\]](#), momentálne využíva na odhad SISO linearizovaný model 2. rádu (bez prepojenia medzi osami) [\[PX4 Autopilot 2021b\]](#)

- **Výhody:** Jednoduché
- **Nevýhody:** tlačí to quad na hranu stability, určite nebude “optimálne” pre všetky prípady a od 1942 máme lepšie metódy. Nastavené na maximálne odstránenie porúch, náchylné na agresívne riadenie s prestrelom [Bequette 2010](#)
- Môžeme podobné experimentálne dáta získať aj zo skokovej odozvy
- ArduPilot (ArduCopter) — “AutoTuning” Napodobňuje túto heuristiku (ale nie je to ZN) [\[Tridgell 2021\]](#)
 - ▶ Zvyšuje P parameter pomaly
 - ▶ Akonáhle deteguje osciláciu, zníži P parameter
 - ▶ Opakuje pre iné zložky/slučky
- PX4 dlho nemal vstavanú “autotune” funkciu [\[Bolton a kol. 2018\]](#), momentálne využíva na odhad SISO linearizovaný model 2. rádu (bez prepojenia medzi osami) [\[PX4 Autopilot 2021b\]](#)

- Cohen Coon je založené na ZN, pridá aj oneskorenie sústavy [Joseph a kol. 2018] (nie je typicky problém pri quadoch)
- Chien, Hrones a Reswickova (CHR) metóda, taktiež založené na ZN. Vyladené na maximalizáciu rýchlosťi odozvy bez prestrelu (alebo 20%) [Hambali a kol. 2014] .

- Cohen Coon je založené na ZN, pridá aj oneskorenie sústavy [Joseph a kol. 2018] (nie je typicky problém pri quadoch)
- Chien, Hrones a Reswickova (CHR) metóda, taktiež založené na ZN. Vyladené na maximalizáciu rýchlosťi odozvy bez prestrelu (alebo 20%) [Hambali a kol. 2014] .

- 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

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] Moti Ben-Ari. *A Tutorial on Euler Angles and Quaternions*. Online. Version 2.0.1 [cited 5.12.2021]; Available from <https://www.weizmann.ac.il/sci-tea/benari/sites/sci-tea.benari/files/uploads/softwareAndLearningMaterials/quaternion-tutorial-2-0-1.pdf>. 2017.
- [4] B. W. Bequette. *Process Control: Modeling, Design, and Simulation*. Prentice Hall PTR, 2010.

- [5] Ryan Boland. *Embedded Programming for Quadcopters*. Online. [cited 2.12.2021]; Available from <https://www.youtube.com/watch?v=CHSYgLfhwUo&t=3s>. 2015.
- [6] Barry Bolton a PX4 Dev Team. *Feature Request: AutoTune*. Online. PX4 GitHub [cited 3.12.2021]; Available from <https://github.com/PX4/PX4-Autopilot/issues/8472>. 2018.
- [7] 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.
- [8] Brian Douglas. *Drone Simulation and Control*. Online. From the series Matlab Tech Talks [cited 3.12.2021]; Available from <https://www.mathworks.com/videos/series/drone-simulation-and-control.html>. 2018.

- [9] 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.
- [10] Leonard Hall. *Practical 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.
- [11] Najidah Hambali a kol. “Process controllability for flow control system using Ziegler-Nichols (ZN), Cohen-Coon (CC) and Chien-Hrones-Reswick (CHR) tuning methods”. In: *2014 IEEE International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*. 2014, s. 1–6.
- [12] Elisha Joseph a O OlaiyaO. “Cohen-Coon PID Tuning Method: A Better Option to Ziegler Nichols-Pid Tuning Method”. In: *Computer Engineering and Intelligent Systems* 9 (2018), s. 33–37.

- [13] Sertac Karaman, Fabian Riether a Eric Wieser Jerome Bouvard. *16.30 Feedback Control Systems — An MIT Feedback Control Systems Class that Teaches with Palm-size Drones*. Online. [cited 5.12.2021]; Available from <http://fast.scripts.mit.edu/dronecontrol/>. 2016.
- [14] 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.
- [15] PX4 Autopilot. *Multicopter PID Tuning Guide (Advanced/Detailed)*. Online. [cited 30.11.2021]; Available from <https://docs.px4.io/master/en/config/autotune.html>. 2021.
- [16] Dola Saha. *Precise Unmanned Aerial Vehicle (UAV) Flight Control*. Online. 2020 Fall ECE553 Resources [cited 3.12.2021]; Available from https://www.albany.edu/faculty/dsaha/teach/2020Fall_ECE553/. 2020.

- [17] [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.
- [18] [Wikipedia](#). *PID controller*. [online]. [cited 25.11.2021]; Available from https://en.wikipedia.org/wiki/PID_controller. 2021.
- [19] [J. G. Ziegler a N. B. Nichols](#). “Optimum settings for automatic controllers”. In: *Transactions of the ASME* 64.- (1942), s. 759–768.