Semestrální práce

Jiří Maňák

I. Zadání

V rámci zadání této semestrální práce jsem dostal příležitost vylepšit řízení aktuátorů prstů myoelektrické protézy, což je projekt, na kterém se podílím již přes dva roky. Jedná se o problém, který je blízkou analogií semestrální úlohy řízení serva Amira, konkrétně převážně vycházím z loňského zadání úlohy.

V této práci se budu zabývat pouze zjednodušeným řízením, na kterém se bude dále stavět v budoucnu, aby bylo prakticky nasaditelné v koncové aplikaci. Výsledný regulátor z této práce by měl být schopen uřídit použité lineární aktuátory do požadované polohy při zadaných maximálních rychlostech. Z toho důvodu jsme se rozhodli implementovat tři řídicí smyčky

- proudový regulátor
- rychlostní regulátor
- poziční regulátor

II. Fyzický model

Platforma, na které budu tuto práci zpracovávat se skládá z výkonové řídicí desky protézy a jednoho aktuátoru prstu, ke kterému je možné přichytit zbytek mechaniky.

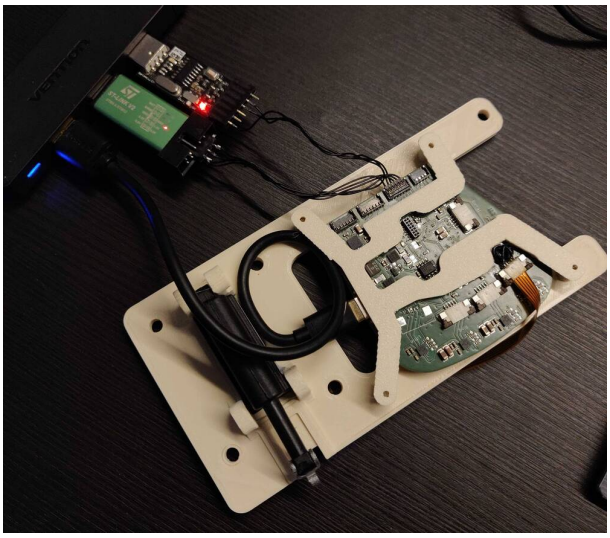


Fig. 1: Vývojová platforma obsahující výkonovou řídicí desku a jeden aktuátor prstu

A. Záchyt dat

Na mikroprocesoru výkonové desky běží vývojová verze firmware, která poskytuje běhové prostředí pro experimentální regulátory. Deska komunikuje s připojeným PC, na kterém je spuštěna ovládací aplikace.

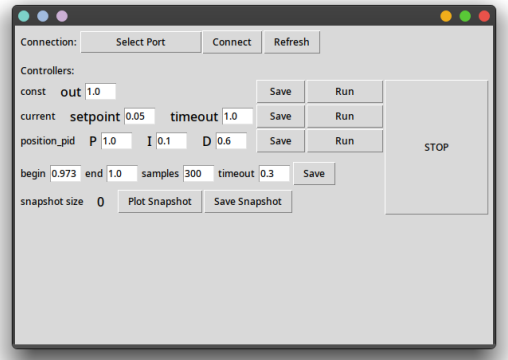


Fig. 2: Uživatelské rozhraní aplikace

Aplikace je schopna spustit vybraný regulátor s danými parametry a následně z přípravku stáhnout zachycené běhové proměnné. Průběh spojení je následující:

1) po stisku tlačítka "Run"v řádku daného regulátoru jsou zachyceny aktuálně nastavené hodnoty parametrů a je připraven příkaz ve formátu JSON. Tento příkaz se serializuje do formátu CBOR a odešle se prostřednictvím transportního protokolu přes sériové rozhraní do desky.
2) deska přijme příkaz, najde příslušný regulátor, inicializuje jej, odpoví na příkaz informacemi o běhovém prostředí a spustí periodicky generovaný interrupt, který implementuje obsluhu sekvence pohybu
3) použitím poziční zpětné vazby je dosažena zadaná výchozí pozice - pole "begin"
4) po dosažení výchozí pozice je spuštěn odpočet 1 s zpoždění před předáním řízení regulátoru, navíc se v posledních 100 ms odpočtu začnou zaznamenávat běhová data, aby bylo možné ne zachycených datech sledovat stav před spuštěním regulátoru
5) po uplynutí odpočtu (v čase 0 s) je spuštěn dříve inicializovaný regulátor, tomu je předána kontrola nad aktuátorem do doby, než nastaví done flag nebo do uplynutí "timeout"sekund od spuštění. Při každém cyklu jsou zaznamenány běhové proměnné
6) po ukončení předchozí řízené fáze se začnou odesílat zachycené běhové proměnné zpět do řídicí aplikace
7) po přijetí všech záchytů jsou data uložena aplikací ve formátu JSON, případně vynesena do grafu

B. Formát záchytu

Příklad uloženého záchytu - klíč `config` obsahuje celý příkaz, který byl použit ke spuštění tohoto běhu, klíč

`env` obsahuje odpověď výkonové desky na tento příkaz a popisuje běhové prostředí, konečně klíč `snapshots` nese samotná data záchytu, ze kterého jsem pro účely příkladu odstranil většinu dat.


```
{
    "config": {
        "ctrl": "current",
        "begin": 0.8,
        "end": 0.2,
        "samples": 300,
        "timeout": 3.0,
        "run": true,
        "args": {
            "setpoint": 0.05,
            "timeout": 2.5
        }
    },
    "meta": {
        "timestamp": 1680623596.7305748,
        "localtime": "Tue Apr
4 17:53:16 2023",
        "port": "/dev/ttyUSB0"
    },
    "env": {
        "ctrl_freq": 100,
        "tick": 373.09
    },
    "snapshots": {
        "time": [
            -0.09000000357627869,
            -0.07999999821186066,
            ...
            2.611999988555908,
            2.621999979019165
        ],
        "setpoint": [...],
        "position": [...],
        "drive": [...],
        "current": [...],
        ...
    }
}
```


Výhodou tohoto formátu je konzistence, snadná čitelnost, možnost reprodukovat testy díky automatickému zachycení kontextu, ve kterém byly historické pokusy provedeny, a také široká kompatibilita s dalšími nástroji, včetně možnosti importu časových posloupností nativně do MATLAB proměnné.

III. Regulace proudu

A. Identifikace

Z odezvy je zřejmé, že se jedná o systém prvního řádu bez nul, což odpovídá zadání a hledanému tvaru přenosové funkce

$$G(s) = \frac{k}{\tau s + 1} \tag{1}$$

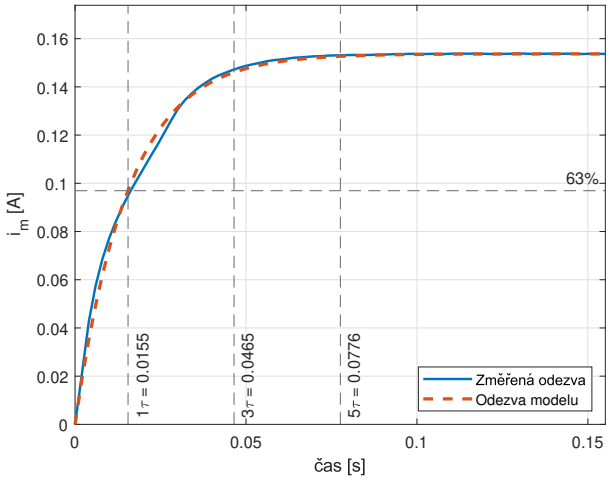


Fig. 3: Přechodová odezva proudu motorem na připojení napájení při zablokovaném táhlu aktuátoru

Zvolil jsem model $g(t) = k(1 - e^{-t/\tau})$ a použil funkci `fit` na datech záchytu, čímž jsem obdržel $\tau = 0.01551s$ a $k = 0.1537A$. Po dosazení dostáváme přenos systému

$$G(s) = \frac{0.1537}{0.01551s + 1} \tag{2}$$

Zachycená přechodová odezva je pravděpodobně ovlivněna nedokonalým zaaretováním táhla aktuátoru, tedy motor není absolutně statický, a vzniká nechtěný pokles a opětovný nárust proudu v průběhu měření odezvy, který způsobený krátkým roztočením motoru.

Navzdory této praktické komplikaci změřená odezva blízce sleduje ideální odezvu systému prvního řádu a na identifikaci to nemá zásadní vliv.

B. Návrh regulátoru

Pro regulaci proudu motorem jsme se rozhodli použít PI regulátor, jehož přenos je

$$C(s) = \frac{K_P s + K_I}{s} \tag{3}$$

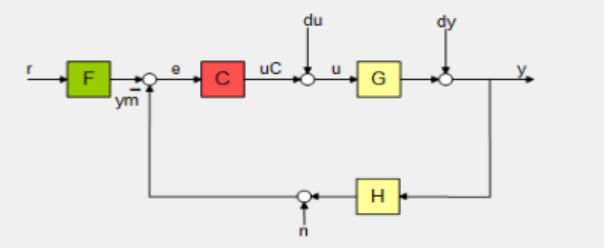


Fig. 4: Zapojení zpětné vazby regulátoru, platí $F = 1$ a $H = 1$

Hodnoty konstant jsem určil s využitím nástroje `sisotool`, výsledky jsou $K_P \approx 21.587$ a $K_I \approx 8869$. Výsledný přenos regulátoru je tedy

$$C(s) = \frac{21.587s + 8869}{s} = \frac{21.587(s + 410.8)}{s} \tag{4}$$

Vzhledem k tomu, že proudový regulátor je na nejnižší úrovni v celém systému, je nutné aby jeho reakce na změny v referenci byla co nejrychlejší, proto jsem se při návrhu snažil minimalizovat dobu náběhu, jelikož jakékoliv zpoždění by se negativně projevilo v dalších krocích řízení.

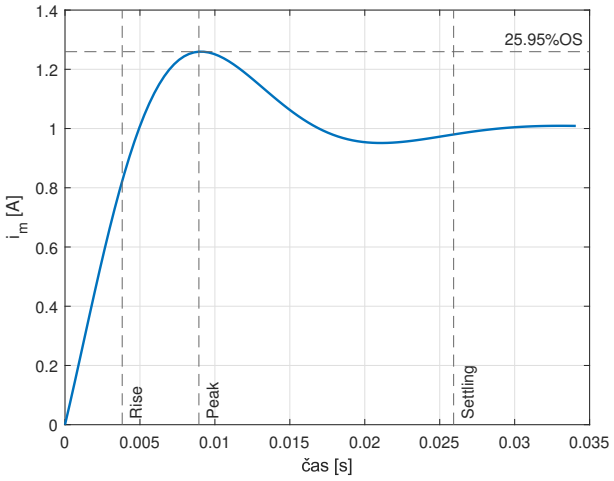


Fig. 5: Odezva na skok zpětnovazebného systému T

Iterativní metodou ladění regulátoru, využitím nástroje PID tuning dostupného v nástrojích ladění `sisotool`, a porovnáváním simulovaných a reálných odezev jsem dospěl k požadavkům

- Response Time 6 ms
- Transient Behavior 0.5 (kompromis mezi robustním a agresivním).

Přenos uzavřené smyčky jsem vypočetl pomocí známého vzorce $T = \frac{C \cdot G}{1+C \cdot G}$ pro zpětnou vazbu. Z CL přenosu můžeme navíc získat Gain a Phase Margin, které také vykazují dobrou záruku stability.

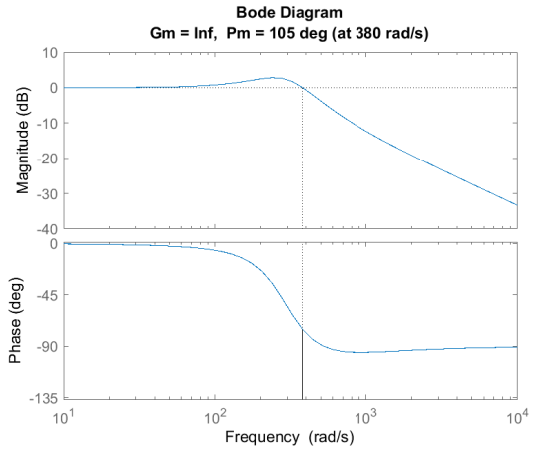
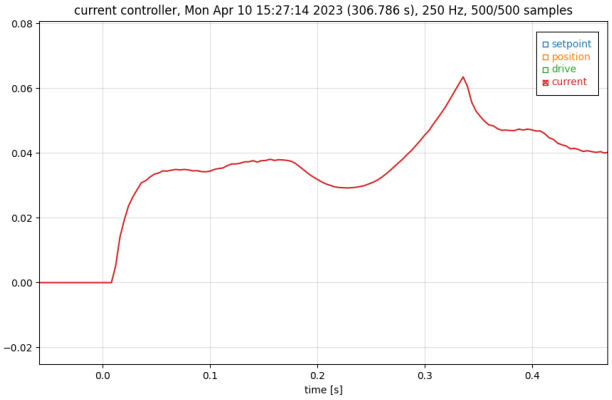


Fig. 6: Bodeho diagram systému T

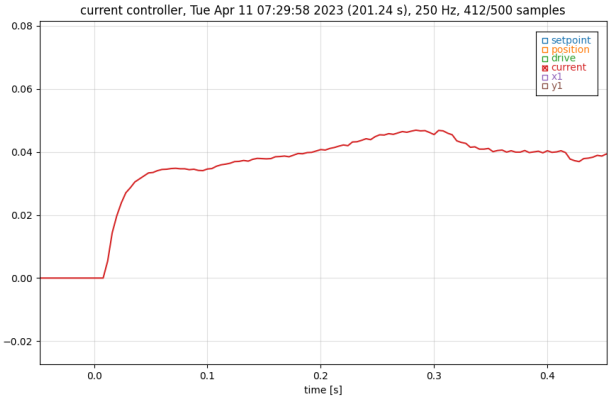
C. Diskretizace a implementace

Přenos regulátoru C jsem diskretizoval využitím bilineární transformace pro $Ts = \frac{1}{500} = 0.002s$ a obdržel předpis diferenční rovnice, kolem které jsem postavil regulátor kopírující blokové schema výše. Koeficienty jsou zaokrouhleny pro lepší orientaci.

$$y[n] = 30x[n] - 13x[n-1] + y[n-1] \tag{5}$$



(a) Regulátor bez omezení $y[n-1]$ složky



(b) Regulátor s omezením $|y[n-1]| \leq 1$

Fig. 7: Posouzení efektivity implementace anti-windup při navozené poruše zastavením pohybu táhla na krátkou dobu

IV. Regulace rychlosti

A. Výpočet rychlosti

Použitý aktuátor neumožňuje měřit rychlost posuvu táhla přímo, ale měří pouze jeho absolutní polohu, takže rychlost je nutné dopočítat. Ze zkušenosti jednoduchý rozdíl poslední a aktuální polohy, diference, v praxi přenáší příliš mnoho šumu, hlavně pro rychlejší vzorkování a malé rozdíly v poloze, což je náš případ.

Experimentálně jsem dospěl ke kompromisu mezi zpožděním a amplitudou šumu rychlosti. Zpracování se skládá ze dvou fází

- IIR filtr druhého řádku typu dolní propust s parametry $f_C = 100\ Hz$ a $Q = 0.7$, který filtruje změřenou polohu táhla na frekvenci vzorkování
- výpočet pseudo-diference z posledních tří vzorků po filtraci, který se odehrává pro každý pátý vzorek

Tímto se sníží frekvence vzorkování pro rychlostní smyčku na pětinu proti proudovému řízení, tedy $f_s = 100\ Hz$,

při identifikaci uvidíme, že systém stejně není schopen reagovat na rychlejší změny v rychlosti.

B. Identifikace

Zde vycházím z popisu systému ze zadání pro servo Amira, konkrétně rovnice

$$\dot{v} = \frac{k_m}{J} i_m - \frac{b}{J} v \tag{6}$$

popisuje mechanické chování systému a jeho odezvu na změny proudu motoru.

Pro účely identifikace jsem šestkrát opakoval záchyt pohybu, vždy se stejným požadavkem na proud $i_{mref} = 20\ mA$, ale pro různé směry pohybu, začáteční polohy a fyzické orientace aktuátoru. Tyto záchyty jsem zprůměroval ve snaze předejít systematické chybě při měření.

Zvolený proud motorem nevyvolá saturaci výstupu proudového regulátoru (tedy napětí na motoru nikdy nedosáhne 100 %) v žádném úseku pohybu, takže při výpočtech můžeme předpokládat $i_m = 20\ mA$, což podle dat ze záchytu platí až na první vzorky zachycující přechodovou odezvu proudového regulátoru po dobu $T_{settling} \approx 25\ ms$.

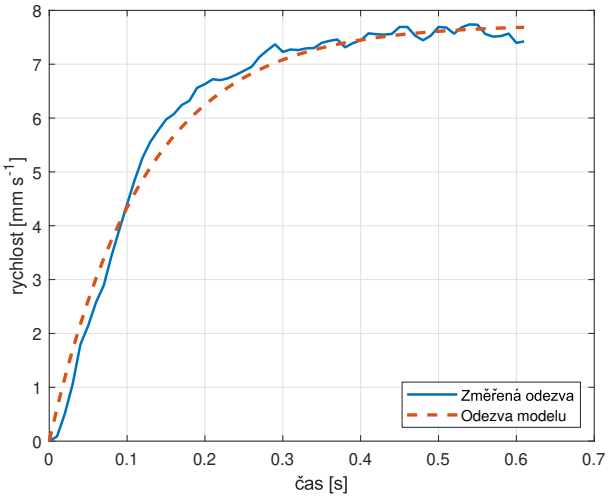


Fig. 8: Odezva rychlosti táhla na změnu proudu motorem

Model jsem zvolil na základě rovnice (6)

$$\dot{v} = a = k_2 i_m - k_3 v \tag{7}$$

přičemž z dat záchytu je možné přímo získat i_m a v, zrychlení a jsem dopočítal jako `a = diff(v) ./ diff(t);`. Konstanty $k_2 = \frac{k_m}{J}$ a $k_3 = \frac{b}{J}$ je nyní možné získat využitím funkce `fit`

```
model = @(k2, k3, x, y) k2*x - k3*y;
fitted = fit([i_m, v], a, model);
```

kde $k_2 = 3187$, $k_3 = 8.238$. Z toho konečně přenos proudu na rychlost

$$G = \frac{3187}{s + 8.238} \tag{8}$$

C. Návrh zpětnovazebního regulátoru

Zvolil jsem PID regulátor s filtrovanou derivační složkou ve snaze zlepšit odezvu na externí poruchy, kde jednoduchý PI regulátor reagoval příliš pomalu. Obecný tvar použitého regulátoru je

$$C = K_P + \frac{K_I}{s} + K_D s \frac{\tau_f s + 1}{\alpha \tau_f s + 1} \tag{9}$$

a opět pomocí `sisotool` jsem navrhl následující realizaci

$$C = \frac{0.004 s^2 + 0.563 s + 7.343}{s^2 + 67.66 s} \tag{10}$$

Postup pro diskretizaci je shodný s proudovým regulátorem, pouze upravíme $T_s = 0.01\ s$.

D. Feed Forward

Z testů vyšlo najevo, že regulace rychlosti pomocí proudu má v našem případě velmi opakovatelný průběh při změnách požadavku na rychlost, což je pravděpodobně zapříčiněno vysokým třením v převodech aktuátoru. Navíc odezva na změny v požadované rychlosti s čistým zpětnovazebním řízením nebyla podle zachycených průběhů tak pohotová jak by mohla být, jelikož akční zásah rostl pozvolně.

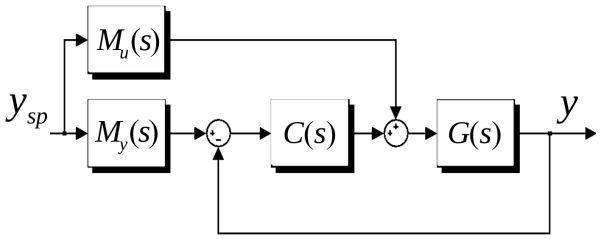
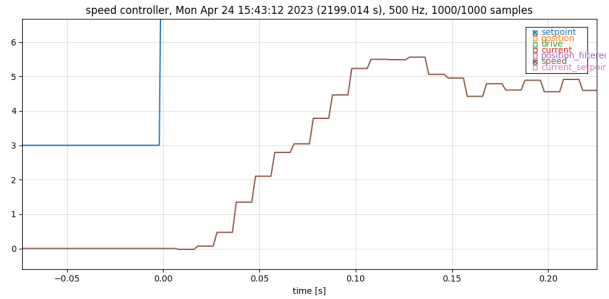


Fig. 9: Architektura celého regulátoru rychlosti

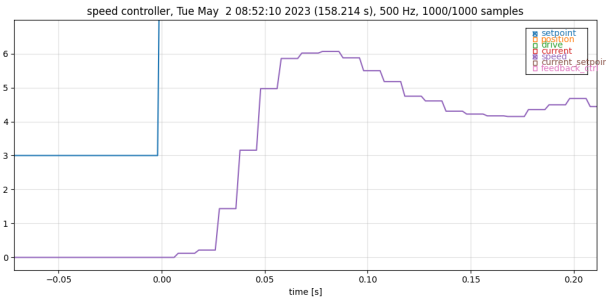
Z těchto důvodů jsem se nezávisle na zadání pokusil o začlenění Feed Forward do již naladěné CL vazby. Zvolil jsem návrh přes model-matching, kde jako požadovaný přenos M_y systému volím systém prvního řádu s časovou konstantou $\tau = 0.03\ s$

$$M_y(s) = \frac{1}{0.03 s + 1} \quad \rightarrow \quad M_u(s) = G^{-1} M_y \tag{11}$$

Zařazením FF se zkrátila doba náběhu na polovinu a navíc je možné omezit saturaci zpětnovazebního regulátoru, což pomáhá se zotavením při externích poruchách, hlavně při úplném zablokování táhla.



(a) zpětnovazební řízení bez FF



(b) zpětnovazební řízení včetně FF

Fig. 10: Porovnání doby náběhu před a po implementaci Feed Forward

E. Výsledky

V této aplikaci není přesné sledování rychlosti hlavním požadavkem, navíc je její řízení komplikováno množstvím šumu a poruch v nepřímém měření rychlosti. Díky tomuto regulátoru hlavně obdržíme další parametr pro ladění celkového chování řízení a usnadní nám navazující regulátory.

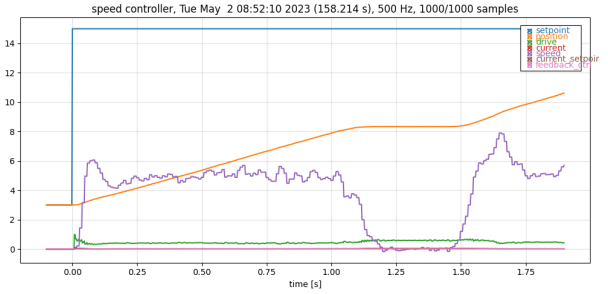


Fig. 11: Odezva na zablokování a následné uvolnění táhla celého regulátoru rychlosti při stálém požadavku rychlosti 5 mm/s. Fialový průběh je rychlost táhla v mm/s.

V. Poziční regulátor

A. Identifikace

Poziční regulátor stavíme nad rychlostním regulátor popsaným v předchozí sekci. Vztah mezi rychlostí a polohou je

$$d(t) = \int_0^t v(\tau)d\tau + d_0 \tag{12}$$

takže základem modelu převodu rychlosti na polohu bude jednoduchý integrátor. Ze záchytů je patrné, že přenos bude obsahovat navíc transportní zpoždění. Zbývá nám určit zesílení K, což je ale směrnice růstu hodnoty na výstupu integrátoru, neboli změřená rychlost pohybu táhla. Celý model je tedy

$$G(s) = v \cdot \frac{1}{s} \cdot e^{-T_d \cdot s} \tag{13}$$

pokud předpokládáme, že rychlost v je konstantní, což platí v záchytu použitém pro identifikaci.

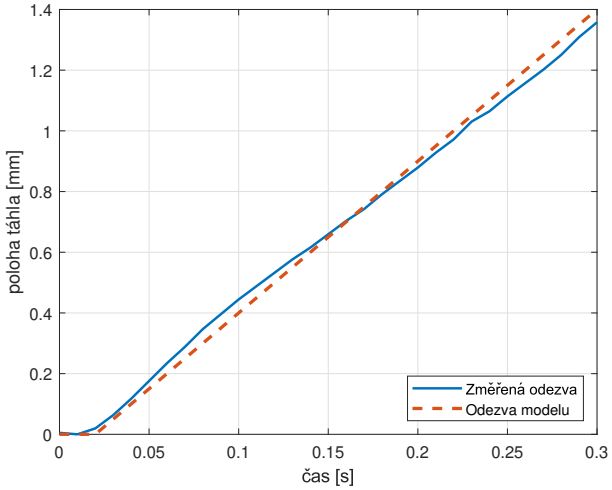


Fig. 12: Odezva polohy táhla při skokovém požadavku rychlosti na 5 mm/s.

Výsledný model této konkrétní realizace je

$$G(s) = \frac{5}{s}e^{-0.02s} \tag{14}$$

B. Zpětnovazební řízení

Vzhledem k relativně jednoduché povaze toho systému jsem zvolil čistou proporcionální zpětnou vazbu, která dosáhne požadované polohy pro hodnoty zesílení v rozsahu 4 až 7 bez kmitání v přijatelném čase.

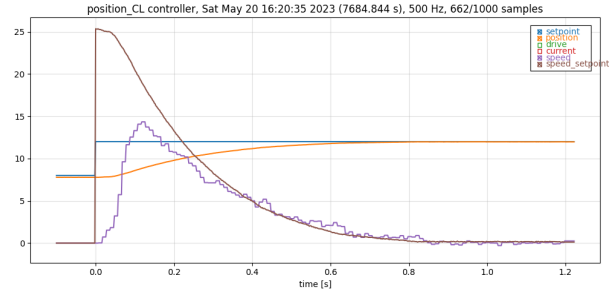


Fig. 13: Odezva pro hodnotu $K = 6$ při požadovaném posunu o 4 mm. Hnědý průběh ukazuje výstup regulátoru, fialový skutečnou rychlost táhla. Modře a žlutě je požadovaná resp. změřená poloha.

C. Přímovazební řízení

Jelikož předpokládáme, že táhlu je umožněn volný pohyb, můžeme také předpokládat, že dokud se nedostane aktuátor do saturace, bude rychlostní regulátor sledovat

požadavek na rychlost bezchybně. Aktuátor se do saturace nedostane, dokud držíme požadavek na rychlost pod maximální rychlostí aktuátoru.

Kvůli těmto předpokladům se nabízí využít čisté přímé vazby pro nastavení polohy. Díky tomu budeme schopni přesně určit zrychlení a zpomalení na začátku a na konci pohybu a navíc můžeme určit maximální rychlost pohybu, což je důležité pokud potřebujeme synchronizovat více aktuátorů.

Implementoval jsem jednoduchý regulátor, který dostane finální polohu a rychlost pohybu jako parametry. Tento regulátor při změně parametrů vypočte čas, za který se pro danou rychlost dostane do požadované polohy a nastaví požadavek na rychlost.

Pro plynulejší přechody jsem použil klouzavý průměr, který z ostrých změn v rychlosti udělá 'spojité' přechody, což nezmění 'plochu pod křivkou průběhu rychlosti', tedy uraženou vzdálenost.

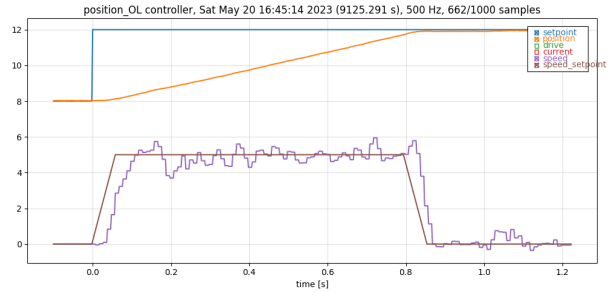


Fig. 14: Odezva při požadovaném posunu o 4 mm při rychlosti 5 mm/s a 60 ms rampách. Hnědý průběh ukazuje výstup regulátoru, fialový skutečnou rychlost táhla. Modře a žlutě je požadovaná resp. změřená poloha.

Při experimentech se ukázalo, že regulátor dosahuje odchylek ± 0.1 mm a nižších, což je postačující pro stávající požadavky.