

Západočeská univerzita v Plzni  
Fakulta aplikovaných věd  
Katedra kybernetiky

## DIPLOMOVÁ PRÁCE

Přesné měření rychlosti a zrychlení rotačního pohybu  
na základě signálu z inkrementálního snímače

PLZEŇ, 2013

VLASTIMIL ŠETKA

**ZADÁNÍ DIPLOMOVÉ PRÁCE**  
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Vlastimil ŠETKA**  
Osobní číslo: **A11N0072P**  
Studijní program: **N3918 Aplikované vědy a informatika**  
Studijní obor: **Kybernetika a řídicí technika**  
Název tématu: **Přesné měření rychlosti a zrychlení rotačního pohybu na základě signálu z inkrementálního snímače**  
Zadávací katedra: **Katedra kybernetiky**

Z á s a d y p r o v y p r a c o v á n í :

1. Seznamte se s pokročilými metodami odhadu rychlosti a zrychlení rotačního pohybu na základě standardního signálu z inkrementálního rotačního snímače (IRC) a proveďte jejich podrobnou analýzu.
2. Navrhněte novou metodu zpracování signálu z IRC, která vede na optimální odhad úhlové rychlosti a zrychlení.
3. Navrhněte hardwarový modul pro základní zpracování signálu z IRC na bázi FPGA.
4. Navržené řešení ověřte simulačně v programovém prostředí MATLAB. Na základě výsledků simulace sestavte optimalizovaný algoritmus zpracování signálu z IRC pro účely měření rychlosti a zrychlení.
5. Navržený algoritmus implementujte v jazyce C a ověřte jeho funkčnost na typických rychlostních profilech.
6. Ověřte funkční vlastnosti celého řetězce zpracování signálu na reálném IRC a porovnejte s teoretickými předpověďmi.

Rozsah grafických prací: **dle potřeby**  
Rozsah pracovní zprávy: **35-50 stránek A4**  
Forma zpracování diplomové práce: **tištěná**  
Seznam odborné literatury:

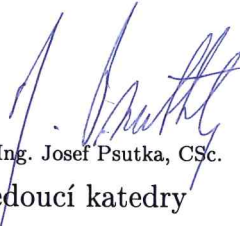
- [1] **A Microprocessor-Controlled High-Accuracy Wide-Range Speed Regulator for Motor Drives (Ohmae et al - IEEE 1982)**
- [2] **Velocity Estimation from Irregular, Noisy Position Measurements (Glad, Ljung - IFAC 1984)**
- [3] **Estimation of Angular Velocity and Acceleration from Shaft Encoder Measurements (Bélanger - IEEE 1992)**
- [4] **A Wide-Range Velocity Measurement Method for Motion Control (Tsuji et al - IEEE 2009)**
- [5] **Improved Digital Tachometer With Reduced Sensitivity to Sensor Nonideality (Kavanagh - IEEE 2000)**
- [6] **Odhad derivací signálu s šumem Kalmanovým filtrem (Schlegel, Večerek - KKY ZČU 2002)**

Vedoucí diplomové práce: **Prof. Ing. Miloš Schlegel, CSc.**  
Katedra kybernetiky

Datum zadání diplomové práce: **24. září 2012**  
Termín odevzdání diplomové práce: **17. května 2013**

  
Doc. Ing. František Vávra, CSc.  
děkan



  
Prof. Ing. Josef Psutka, CSc.  
vedoucí katedry

V Plzni dne 24. září 2012

## PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 16. května 2013

---

vlastnoruční podpis

### Poděkování

Děkuji panu Prof. Ing. Miloši Schlegelovi, CSc. za hodnotné rady a vedení této diplomové práce.

Děkuji svým rodičům za podporu v průběhu celého studia.

## **Anotace**

Diplomová práce se zabývá možnostmi přesného měření rychlosti a zrychlení rotačního pohybu na základě signálu z inkrementálního snímače

Řešení tohoto problému používaná v současné průmyslové praxi neposkytují požadovanou přesnost měření, což neumožňuje uspokojit stále vyšší požadavky na kvalitu regulace. V této práci je navržena a otestována komplexní metoda zpracování signálu z levných inkrementálních snímačů, kterými jsou průmyslové pohony standardně vybaveny.

Navržená metoda využívá předzpracování signálu na programovatelném hradla-vém poli FPGA a výpočetní algoritmus běžící na výkonném průmyslovém po-čítači. Tyto technologie se v poslední době staly cenově dostupnými i pro běžné aplikace. Výpočet rychlosti a zrychlení je realizován na bázi Kalmanova filtru pro optimální odhad stavu vhodně definovaného modelu měřeného signálu.

## **Klíčová slova**

Měření, rychlost, zrychlení, inkrementální snímač, IRC, Kalmanův filtr, FPGA.

## **Anotation**

The diploma thesis analyses possibilities of precise angular velocity and acce-leration measurement using an incremental encoder.

Solutions of this problem available in today's industrial systems provide poor accuracy which is not sufficient for high demands on quality of regulation. The thesis provides design and test results of a complex measurement method for signal processing from a cheap incremental encoders which is a standard equipment of motor actuators.

Proposed method uses signal preprocessing on the Field Programmable Gate Array (FPGA) and computational algorithm running on high performance in-dustrial computer. These technologies are now available at reasonable price for intended applications. Computation of velocity and acceleration is based on the Kalman filter for optimal estimation of feasible measured signal model.

## **Keywords**

Measurement, velocity, acceleration, incremental encoder, Kalman filter, FPGA.

# Obsah

<b>1 Úvod</b>	<b>8</b>
1.1 Současný stav řešené problematiky . . . . .	8
1.2 Cíl práce . . . . .	9
1.3 Obsah práce . . . . .	10
<b>2 Inkrementální snímače</b>	<b>12</b>
2.1 Princip funkce . . . . .	12
2.2 Dekódování výstupního signálu . . . . .	14
2.3 Elektrická rozhraní . . . . .	17
2.4 Chyby IRC snímačů . . . . .	19
<b>3 Metody zpracování signálu z IRC snímače</b>	<b>23</b>
3.1 Základní metody měření dekodovaného IRC signálu . . . . .	24
3.2 Adaptivní metoda měření . . . . .	28
<b>4 Filtr pro odhad polohy, rychlosti a zrychlení</b>	<b>32</b>
4.1 Formulace problému . . . . .	32
4.2 Odhad derivací . . . . .	34
4.3 Časová predikce . . . . .	37
4.4 Numerická implementace výpočtů . . . . .	39
<b>5 Simulace v systému MATLAB</b>	<b>42</b>
5.1 Výkonné třídy – funkční bloky . . . . .	44
5.2 Pomocné funkce . . . . .	49
5.3 Funkce pro generování simulačních dat . . . . .	50
5.4 Funkce pro testování měřicích a filtračních algoritmů . . . . .	53
<b>6 Vyhodnocení vlastností navržených algoritmů</b>	<b>57</b>

6.1	Simulační výsledky . . . . .	58
6.2	Měření na reálném snímači . . . . .	66
<b>7</b>	<b>Návrh integrace do prostředí reálného času</b>	<b>68</b>
7.1	Koncepce řešení . . . . .	68
7.2	Popis implementace algoritmu filtru v C++ . . . . .	70
7.3	Vyhodnocení a optimalizace výpočetní náročnosti . . . . .	74
<b>8</b>	<b>Návrh FPGA logiky pro měření metodou M/T</b>	<b>78</b>
<b>9</b>	<b>Převodník pro připojení IRC snímače k FPGA</b>	<b>80</b>
9.1	Použité součástky . . . . .	81
9.2	Popis obvodového řešení . . . . .	82
9.3	Dosažené vlastnosti a parametry . . . . .	84
9.4	Naměřené parametry . . . . .	84
<b>10</b>	<b>Závěr</b>	<b>87</b>
<b>11</b>	<b>Příloha A – Datový nosič CD</b>	<b>92</b>
<b>12</b>	<b>Příloha B – Schéma a DPS převodníku</b>	<b>93</b>

# 1 Úvod

V oblasti automatického řízení mechatronických systémů vzniká potřeba řízení polohy, rychlosti a zrychlení. Tyto veličiny je třeba měřit, k čemuž se velmi často používají tzv. *inkrementální rotační snímače*, též nazývané *IRC snímače* nebo *enkodéry*. Kvalita měření má přitom zásadní vliv na kvalitu regulace, na kterou jsou kladeny stále vyšší požadavky.

Metody měření rychlosti a zrychlení pomocí IRC snímačů používané v současné průmyslové praxi neumožňují dosáhnout požadované kvality a přesnosti a jsou citlivé na nedokonalosti reálných snímačů. Zejména měření úhlového zrychlení je v praxi považováno za velmi obtížný úkol.

Rychlý vývoj v oblasti výpočetní techniky přinesl v posledních letech vysoký výpočetní výkon i do průmyslových systémů, a zároveň se dostupnějšími a levnějšími staly technologie výkonné programovatelné logiky – obvody FPGA. Vzniká tak prostor pro zásadní zlepšení metod zpracování signálu z IRC snímačů pro plné využití jejich potenciálu, a také pro zlepšení algoritmů výpočtu rychlosti a zrychlení např. použitím metod *optimálního odhadu*. Přitom potřebné technologie jsou dostupné s náklady akceptovatelnými pro široké spektrum aplikací řízení pohonů.

## 1.1 Současný stav řešené problematiky

V úlohách automatického řízení je IRC snímač připojen k číslicovému řídicímu systému (ŘS), který provádí měření, výpočty a zásahy s určitou konstantní vzorkovací periodou  $T_s$ .

Nejjednodušší, a v průmyslové praxi téměř výhradně používané, metody zpracování signálu z IRC snímačů jsou založeny na *počítání počtu pulzů za vzorkovací periodu  $T_s$* . Výhodou je nenáročnost na hardwarovou realizaci, neboť stačí čítač počtu dekodovaných pulzů respektující směr otáčení, jehož hodnotu čteme v každém vzorkovacím okamžiku ŘS. Změna hodnoty čítače za  $T_s$  odpovídá relativní změně polohy za  $T_s$ , tj. rychlosti. Zrychlení lze vypočítat z rozdílu rychlostí v předchozím a aktuálním kroku.

Informace o změně polohy (úhlu natočení) ze snímače, tj. dekodované polohové pulzy, však z principu jeho funkce přichází asynchronně k vzorkování ŘS. Vzniká tak chyba v určení změny polohy, jejíž absolutní velikost je rovna až jednomu celému pulzu. Relativní význam této chyby roste s nižší rychlostí, tedy s nižším počtem pulzů za  $T_s$ . Např. zaznamenanáme-li jen 10 pulzů za  $T_s$ , je relativní chyba takto změřené rychlosti 10 %.

Z pohledu kvality regulace požadujeme obvykle co nejkratší  $T_s$ , aby bylo možné akčními



zásahy rychle reagovat na měřené hodnoty. Zde ovšem platí, že čím kratší  $T_s$ , tím větší chyba měření rychlosti.

Požadavek na přesnější měření malých rychlostí se pak řeší použitím speciálních a drahých snímačů s velkým počtem dílků na otáčku, které ale na druhou stranu nelze použít pro měření velkých rychlostí z důvodu omezení maximální frekvence výstupního signálu.

V praxi jsou výše popsané chyby velmi významné a limitují dosažitelnou kvalitu řízení. Kromě toho mají reálné snímače další nedokonalosti, jejichž vliv je třeba kompenzovat nebo filtrovat. Proto má zásadní smysl zabývat se možnostmi přesnějších metod zpracování IRC signálu. Dokladem toho je, že v posledních desítkách let byla k tomuto problému publikována řada článků. Několik příkladů:

- OHMAE. 1982. A Microprocessor-Controlled High-Accuracy Wide-Range Speed Regulator for Motor Drives [1],
- GLAD. 1984. Velocity Estimation from Irregular, Noisy Position Measurements [2],
- BÉLANGER. 1992. Estimation of Angular Velocity and Acceleration from Shaft Encoder Measurements [3],
- PROKIN. 1994. Extremely Wide-Range Speed Measurement Using a Double-Buffered Method [4],
- KAVANAGH. 2000. Improved Digital Tachometer With Reduced Sensitivity to Sensor Nonideality [5],
- TSUJI. 2009. A Wide-Range Velocity Measurement Method for Motion Control [6].

## 1.2 Cíl práce

Cílem této práce je nalezení vhodné metody zpracování signálu z IRC snímače a následně vhodného algoritmu odhadu polohy, rychlosti a zrychlení, přičemž hlavním požadavkem je dosažení co nejvyšší přesnosti odhadů v reálných podmínkách a také realizovatelnost a dostupnost pro běžné průmyslové aplikace řízení pohybu.

Nalezené metody měření by měly být použité pro velmi široký rozsah hodnot rychlostí a zrychlení, a měly by být odolné proti poruchám způsobeným nedokonalostmi snímačů.

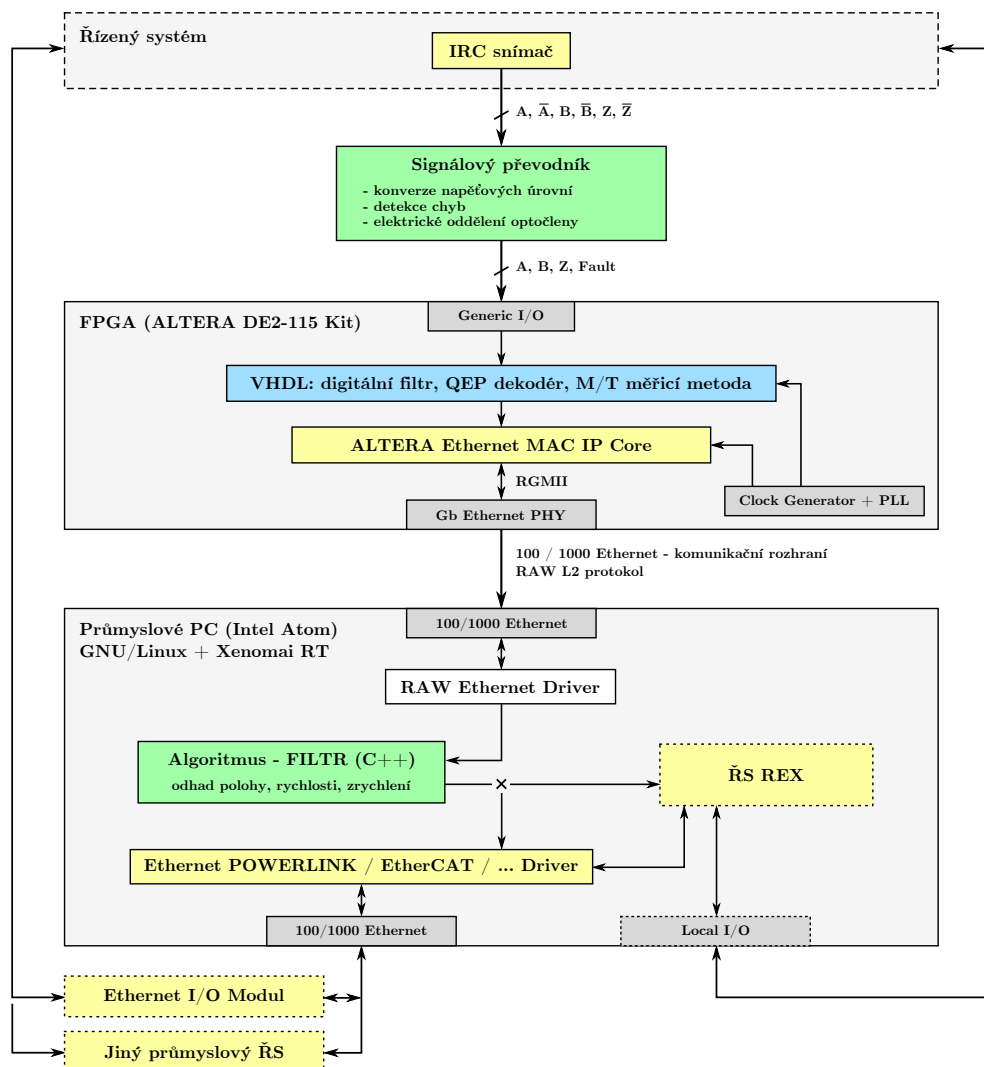
Řešení musí být použitelné v řídicí smyčce, tedy v reálném čase s vzorkováním v řádu 1 ms.

### 1.3 Obsah práce

Řešení problému měření rychlosti a zrychlení, jak si klade za cíl tato práce, je komplexní úkol skládající se z mnoha dílčích problémů, které je třeba poznat a vyřešit.

V této práci je proveden rozbor a návrh řešení všech dílčích úloh. Zásadní částí je úplné řešení a ověření vlastností výpočetního algoritmu pro odhad rychlosti a zrychlení pomocí Kalmanova filtru, včetně optimalizované implementace v jazyce C++ pro použití v prostředí reálného času. Vstupem algoritmu je signál z IRC snímače předzpracovaný navrženou adaptivní měřicí metodou, jejíž implementace na FPGA je zde popsána ve fázi návrhu.

Celkový přehled částí navrženého řešení a jejich souvislostí ukazuje obrázek 1.



Obrázek 1: Blokové schéma celkové situace řešeného problému a jeho částí.

V úvodu je podrobně popsán princip a vlastnosti inkrementálních snímačů (kapitola 2).

Dále jsou popsány známé přístupy ke zpracování signálu z těchto snímačů a jejich chyby měření. Je navržena adaptivní metoda minimalizující chybu v širokém rozsahu rychlostí, která je vhodná pro další zpracování měření algoritmem filtru (kapitola 3).

Hlavní a nejdůležitější částí práce je detailní návrh algoritmu pro odhad skutečné polohy, rychlosti a zrychlení s využitím Kalmanova filtru. Jsou vyřešeny problémy diskretizace a asynchronního vzorkování a navrženy vhodné algoritmy pro numerickou realizaci potřebných výpočtů (kapitola 4). Pro ověření vlastností navrženého řešení s modelovými i skutečnými vstupy bylo vytvořeno simulační prostředí – knihovna funkcí pro systém MATLAB (kapitola 5). Výsledky mnoha realizovaných testů jsou shrnuty v kapitole 6.

Dále byl navržen způsob integrace do prostřední reálného času včetně naprogramování algoritmu filtru v jazyce C++, rozboru výpočetní náročnosti, a realizace významných optimalizací (kapitola 7).

Implementance dekodéru signálu ze snímače, adaptivní měřicí metody, a přenosu přes rozhraní Ethernet na hradlovém poli FPGA je dovedena do fáze návrhu (kapitola 8). Kompletní vývoj řešení v jazyce VHDL je již mimo rozsah této práce.

Součástí práce je též návrh elektronického převodníku pro připojení průmyslových snímačů k vývojové desce s FPGA obvodem. Na vzorku převodníku, který byl vyroben, byly naměřeny časové parametry zkreslení průchodu signálu (kapitola 9).

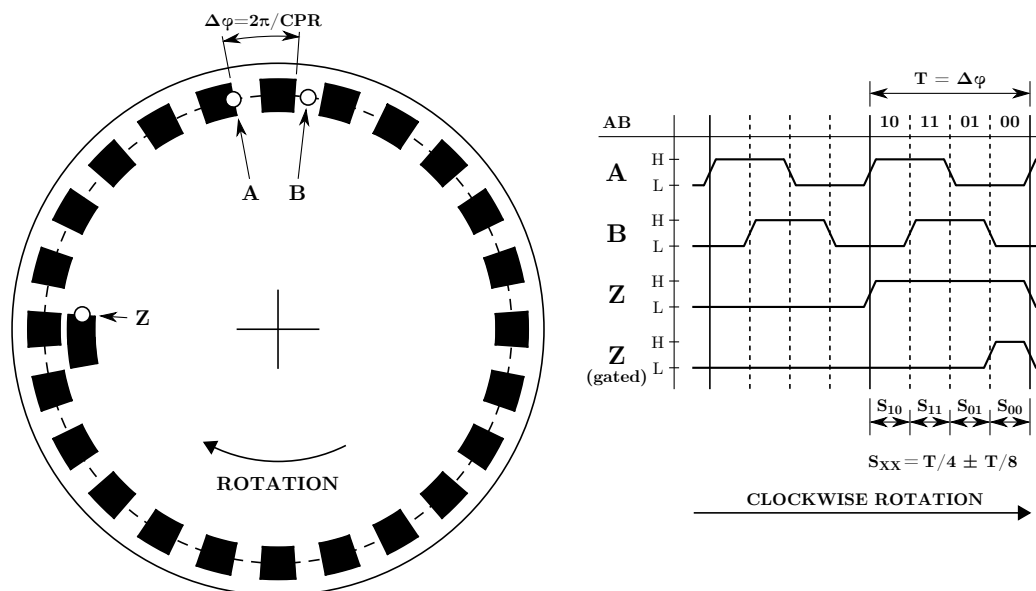
## 2 Inkrementální snímače

### 2.1 Princip funkce

Inkrementální snímače patří do kategorie elektromechanických snímačů polohy. Jejich funkce spočívá v převodu informace o relativní změně polohy na elektrický signál. Nepřímo (výpočtem, zpracováním signálu) pak lze odvodit informaci o rychlosti a zrychlení.

Konstrukční uspořádání inkrementálního snímače může být rotační nebo lineární – podle druhu pohybu, princip funkce je shodný. V dalším textu se zaměříme na snímače rotační, označované zkratkou IRC (*Incremental Rotation enCoder*), v literatuře se lze často setkat též s označením *Shaft Encoder*. U rotačního pohybu pak polohu označujeme jako *úhel*, veličiny rychlost a zrychlení uvažujeme jako úhlové.

Fyzikální princip funkce těchto snímačů je nejčastěji optický. Rotační pohyb je přenášen na skleněný kotouč, na kterém je po obvodu rovnoměrně rozmístěn určitý počet dílků, které se skládají z kombinace transparentní a tmavé plošky. Počet dílků na otáčku je označován CPR (*Counts Per Revolution*), viz např. [8]. Hodnoty CPR u běžných snímačů jsou v rozsahu 100 až 5 000, u vysoce přesných typů až 50 000. Časté jsou hodnoty odpovídající mocninám dvou, např. 512, 1024, 4096. Schematicky je princip znázorněn na obrázku 2.



Obrázek 2: Princip funkce IRC snímače a průběh výstupních signálů.

Tmavé plošky mají funkci clony, která přerušuje světelný paprsek mezi zdrojem světla a světlocitlivým přijímačem (fotodetektorem). Počet přerušení paprsku, tj. počet dílků,

je pak úměrný relativní změně polohy (úhlu).

Aby bylo možné rozlišit směr otáčení, je přijímač zdvojený. Aktivní pozice přijímačů jsou posunuty o polovinu šířky dílku ( $T/4$ ). Dostáváme tak dva výstupní signály označené  $A$  a  $B$ , které jsou vzájemně fázově posunuté o  $90^\circ$ . Dvojice takto posunutých signálů zároveň umožňuje zpřesnit měření, neboť dostáváme dvojnásobný počet hran na jednu otáčku.

Snímače obvykle mají ještě třetí, referenční / index, výstupní signál označený  $Z$  nebo  $I$ . Referenční signál indikuje nulovou polohu jedním impulzem na otáčku. Lze tak odvodit i absolutní polohu. Šířka impulsu může být  $T$ , tj. odpovídající jednomu dílku, nebo může být výstup  $Z$  tzv. hradlovaný (*gated*) jedním nebo oběma signály  $A$  a  $B$ , pak je šířka pulzu  $T/2$  nebo  $T/4$  (viz obrázek 2).

Konkrétní uspořádání optických přijímačů u různých typů inkrementálních snímačů může být odlišné a složitější než bylo popsáno výše. Cílem je omezení jevů, které mají nepříznivý dopad na přesnost snímače, jak bude popsáno dále v kapitole 2.4.

Kromě optických snímačů se lze setkat též s magnetickým principem funkce. Výhodou je bezkontaktní uspořádání – odpadá mechanická vazba mezi hřídelem měřeného objektu a snímačem. Výstupní signály jsou funkčně shodné s optickými snímači, takže pro potřeby této práce se rozdílů nemusíme zabývat.

Konkrétní parametry se zpravidla liší u různých výrobců a konkrétních typů snímačů. Obsah této kapitoly vychází např. z katalogových listů:

- [16] AVAGO TECHNOLOGIES, *HEDM-55xx/560x* & *HEDS-55xx/56xx*,
- [17] MAXON MOTOR, *Encoder HEDL 5540*,
- [18] PEPPERL+FUCHS, *RHI58N, Series 58*,
- [19] AUTONICS, *E40 Series*,
- [20] ENCODER PRODUCTS COMPANY, *Model 755A*.

Kromě výše popsaného typu výstupu ve formě obdélníkového (logického) signálu, existují speciální typy snímačů s výstupem označeným jako *sin/cos*. Signály  $A$  a  $B$  pak mají harmonický průběh, fázový posun  $90^\circ$  je zachován. Výhodou je možnost zvýšení rozlišení tzv. *interpolací*, kdy v jedné periodě harmonického signálu na základě vhodně nastavených rozhodovacích úrovní rozlišujeme větší množství polohových stavů, např. 8, 16 a více.

Tímto se dále nebudeme zabývat, neboť interpolovaný výstup lze z pohledu zpracování signálu považovat za standardní obdélníkový výstup snímače s vyšším CPR.

## 2.2 Dekódování výstupního signálu

Výstupní signály  $A$  a  $B$  jsou vzájemně svázány fázovým posuvem  $90^\circ$ , tj.  $1/4$  úhlové šířky dílku. Takový signál označujeme jako *kvadrurní*, nebo *QEP – Quadrature Encoder Pulses*.

Úloha dekodování signálu spočívá v odvození směru otáčení a polohových pulzů (*edge-pulses*) odpovídajících změně polohy o určitou jednotku ze signálu QEP.

Platí, že v okamžiku, kdy signál na některém z výstupů  $A$  nebo  $B$  přechází z nízké do vysoké logické úrovně nebo naopak (tj. hrana signálu), se kotouč snímače nachází v určité přesně definované poloze – jeden z optických přijímačů mívá rozhraní transparentní a tmavé plošky.

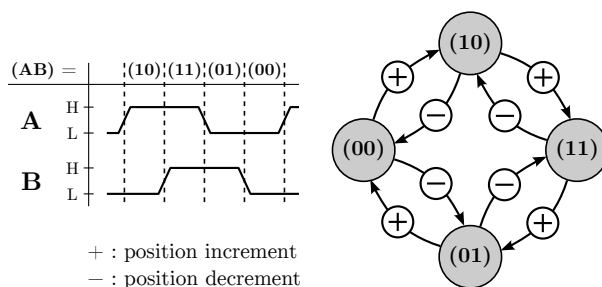
Každá hrana signálu  $A$  nebo  $B$  tedy znamená změnu polohy o čtvrtinu šířky dílku, tj.:

$$\frac{\Delta\varphi}{4} = \frac{1}{4} \frac{2\pi}{\text{CPR}}$$

Změnu polohy v takovém případě odvozujeme ze 4 možných hran –  $A$  vzestupná a sestupná,  $B$  vzestupná a sestupná. Tento režim dekodování je v literatuře označován jako  $4X$  nebo *čtyřnásobná přesnost*. Frekvence generovaných impulzů (*edge-pulses*) je totiž čtyřikrát vyšší než frekvence signálů  $A$  a  $B$ . V některých případech může být užitečné pro dekodování použít obě hrany jen z jednoho signálu, nebo i jen jednu hranu z jednoho signálu. Tyto režimy jsou označovány jako  $2X$  a  $1X$ .

Pro odvození směru otáčení platí:

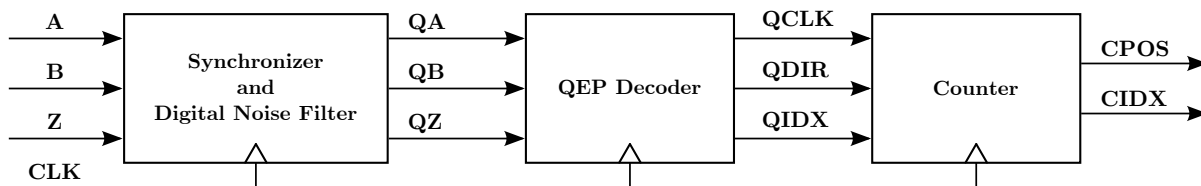
- Vzestupná hrana  $A$  je o  $90^\circ$  v předstihu před  $B \rightarrow$  pohyb v *kladném* směru.
- Vzestupná hrana  $A$  je o  $90^\circ$  opožděna za  $B \rightarrow$  pohyb v *záporném* směru.



Obrázek 3: Statový automat pro dekodování směru otáčení ze signálů  $A$  a  $B$ .

K řešení dílčí úlohy odvození směru otáčení lze přistoupit různými způsoby. Jako nejvhodnější pro další postup a implementaci na FPGA se jeví uvažovat signály  $A$  a  $B$  jako binární dvojici, která může nabývat 4 možných stavů:  $(AB) = (00) / (01) / (10) / (11)$ .

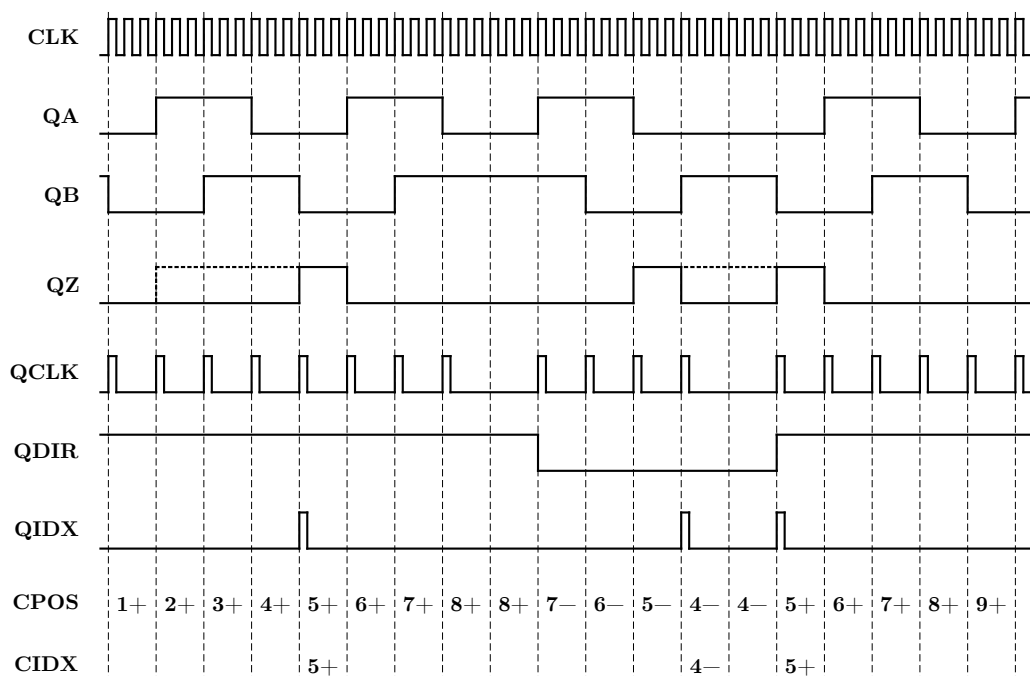
Přechody mezi těmito stavy budeme modelovat jako standardní *stavový automat*, obrázek 3. Tento přístup je popsán např. v aplikačních příručkách integrovaných dekodérů Texas Instruments TMS [14] nebo Microchip dsPIC [15].



Obrázek 4: Ideové blokové schéma základního zpracování signálu z IRC snímače.

Pro další popis definujeme zjednodušené blokové schéma řetězce zpracování signálu, obrázek 4. Vzhledem k budoucí implementaci na FPGA jsou všechny bloky a interní signály uvažovány jako synchronní se systémovým hodinovým signálem CLK.

První blok – digitální filtr – zajišťuje synchronizaci vstupních signálů (A, B, Z) a odstranění případných zákmitů. Následuje blok dekodéru, který obsahuje logiku pro odvození polohových pulzů (QCLK), směru otáčení (QDIR) a index pulzu (QIDX). Následuje čítač, který udržuje informaci o poloze – s každým pulzem na QCLK zvýší nebo sníží (podle QDIR) hodnotu na výstupu CPOS o jedna, výstup CIDX odpovídá hodnotě CPOS při posledním index pulzu. Hodnoty CPOS a CIDX obsahují i informaci o směru.



Obrázek 5: Příklad časových průběhů kvadrurních signálů A, B, Z a výstupu dekodéru.

Příklad časových průběhů všech signálů je na obrázku 5.

Zpracování signálu  $Z$  komplikují různé varianty jeho hradlování, jak bylo popsáno výše. Na obrázku 5 je tečkovaně vyznačena varianta bez hradlování, jinak je uvažováno hradlování na  $(AB) = (00)$ . Pro reálné využití s různými typy snímačů musí být dekodér konfigurovatelný pro hradlování na libovolnou kombinaci  $(AB)$ . Navíc, je-li výstup  $Z$  snímačem hradlován, není obvykle zaručen předstih hrany signálu  $Z$  před hranami signálů  $A$  a  $B$ , a potom QIDX nemusí být synchronní s QCLK.

Index pulzy lze kromě odvození absolutní polohy možné využít také ke kontrole správné funkce dekodéru a samotného snímače. Dobrý návrh dekodéru zajistí, že pulz (QIDX) je generován vždy ve stejné poloze, nezávisle na změnách směru otáčení. Polohy zaznamenané při index pulzech (CIDX) se při správné funkci musí vzájemně lišit o nominální počet pulzů na otáčku ( $CPR \times 4$ ). Můžeme tak kontrolovat, že nedošlo ke ztrátě žádného impulzu, což by se mohlo stát například z důvodu poruchy snímače nebo rušení vstupních signálů.

Pro další obsah této práce je podstatné, že z IRC signálu dekódujeme informaci o poloze, která nabývá diskrétních hodnot, a která může přijít v libovolném čase.

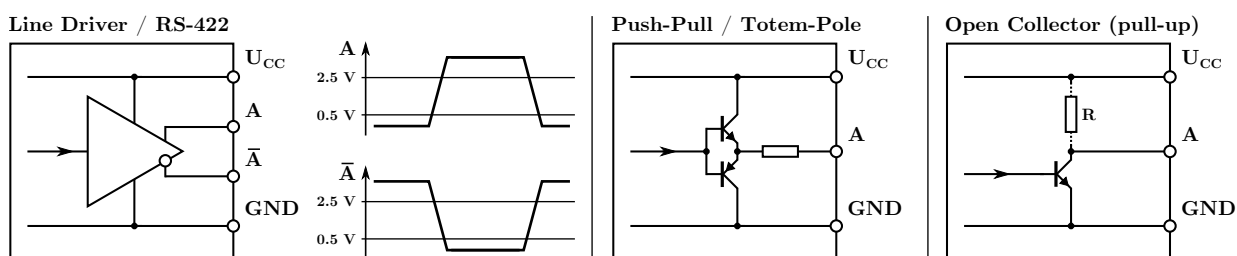
*Dostáváme vlastně měření polohy (úhlu)  $\varphi(t_i)$  v diskrétních časových okamžicích  $t_i$ , přičemž hodnoty  $t_i$  jsou obecně neekvidistantní a předem neznámé.*



## 2.3 Elektrická rozhraní

Dosud jsme výstupní signály snímače uvažovali jako logické, s dvěma možnými stavy – nízká a vysoká logická úroveň. Reálně jsou tyto stavy reprezentovány elektrickými napětovými úrovněmi. U snímačů pro průmyslové aplikace se používá několik různých standardních elektrických rozhraní, která popíšeme v této kapitole. Označení používaná v literatuře:

- Line Driver, RS-422 – diferenciální výstup,
- Push-Pull, Totem-Pole, TTL (Transistor Transistor Logic), HTL (High Threshold Logic) – výstup ve dvojčinném zapojení,
- Open Collector – výstup typu otevřený kolektor.



Obrázek 6: Srovnání různých typů elektrických rozhraní.

Základní rozdělení je do dvou skupin – výstup diferenciální, nejčastěji označovaný podle standardu RS-422, a výstup jednopólový (*single-ended*). Napětové úrovně se pro různé typy snímačů mírně liší. Dále uvedeme podrobnější popis jednotlivých typů rozhraní.

### Line Driver / RS-422

Každý výstupní kanál je tvořen *diferenciálním párem* vodičů, které jsou označeny např. A (pozitivní výstup) a  $\bar{A}$  (negativní výstup). Logickou úroveň nevyhodnocujeme proti společné zemi, ale na základě *rozdílu* napětí mezi A a  $\bar{A}$ . Napětové úrovně:

- napájecí napětí:  $U_{CC} = 5 \text{ V}$ ,
- L (A), H ( $\bar{A}$ ):  $\leq 0,5 \text{ V}$  proti zemi,
- H (A), L ( $\bar{A}$ ):  $\geq 2,5 \text{ V}$  proti zemi,
- maximální zatížení výstupu:  $\pm 20 \text{ mA}$ .

Diferenciální výstup proti ostatním typům obvykle nabízí nejlepší strmost hran a nejvyšší maximální frekvenci. Přitom je zachována vysoká odolnost proti indukovaným rušivým

signálům. Indukované rušení je totiž superponováno na oba vodiče diferenciálního páru se stejnou polaritou, takže neovlivňuje vyhodnocovaný rozdíl napětí.

### **Push-Pull / Totem-Pole / TTL / HTL**

Označení Push-Pull a Totem-Pole mají shodný význam, používají se pro tzv. dvojčinné zapojení výstupního budiče, které umožňuje zatížit výstup v obou logických úrovních. Tím je rozdíl od výstupu typu Open Collector dosaženo přibližně stejné doby trvání vzestupné a sestupné hrany. Výstupní kanál je tvořen jedním vodičem, napětí měříme proti zemi.

Zkratky TTL a HTL pak určují napěťové úrovně:

- TTL – napájecí napětí:  $U_{CC} = 5 \text{ V}$ ; L:  $\leq 0,4 \text{ V}$ ; H:  $\geq U_{CC} - 2,0 \text{ V}$ ,
- HTL – napájecí napětí:  $U_{CC} = 10 - 30 \text{ V}$ ; L:  $\leq 0,4 \text{ V}$ ; H:  $\geq U_{CC} - 3,0 \text{ V}$ ,
- maximální zatížení výstupu:  $\pm 20 \text{ mA}$ .

Velký rozkmit napěťových úrovní HTL poskytuje ochranu proti rušení, ovšem za cenu horších dynamických parametrů než u diferenciálního řešení.

### **Open Collector**

Označení Open Collector se používá pro výstup, který je možné zatížit pouze v nízké logické úrovni. Vnitřní zapojení výstupního budiče obsahuje jediný tranzistor, z jehož kolektoru je vyveden výstup. Někdy je doplněn tzv. *pull-up* rezistor mezi výstupem a napájením, který definuje napěťovou úroveň H při uzavřeném tranzistoru (viz obrázek 6).

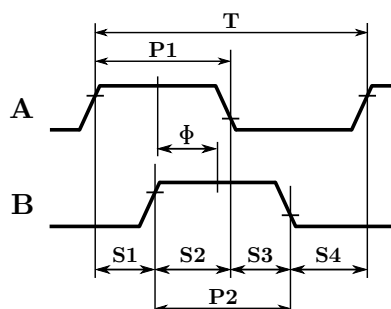
Výhodou tohoto řešení je možnost jednoduchého připojení ke vstupu, který má logické úrovně např. CMOS 3,3 V, i když je snímač napájen vyšším napětím. Nevýhoda je, že může docházet k významnému rozdílu ve strmosti vzestupné a sestupné hrany signálu, tedy ke zkreslení přenášené informace.

Napěťové úrovně:

- napájecí napětí:  $U_{CC} = 5 \text{ V}, 10 - 30 \text{ V}$ ,
- L:  $\leq 0,4 \text{ V}$  (zbytkové napětí),
- maximální zatížení výstupu:  $-30 \text{ mA}$ .

## 2.4 Chyby IRC snímačů

Dosud jsme IRC snímač uvažovali jako ideální prvek. Protože obsahem této práce je vývoj metody pro velmi přesné měření, je třeba porozumět hlavním typům nepřesností reálných snímačů, jejich příčinám a případně možnostem jejich kompenzace při zpracování signálu.



Obrázek 7: Parametry časového průběhu signálů A a B.

Pro popis důsledků těchto chyb definujeme hlavní parametry časových průběhů výstupních signálů (obrázek 7) a označení jejich chyb:

- $T = 360^\circ e$  – chyba periody – *cycle error*,
- $P1, P2 = 180^\circ e$  – chyba délky pulzu – *pulse width error*,
- $\Phi = 90^\circ e$  – chyba fáze – *phase error*,
- $S1, S2, S3, S4 = 90^\circ e$  – chyba délky stavu – *state width error*.

Jednotky nominálních hodnot jsou stupně periodického signálu (*electrical degrees*).

Pro další zpracování signálu je důležité, že *chyba fáze, délky stavu a délky pulzu je u reálných snímačů výrazně vyšší než chyba periody*.

Hlavní faktory, které ovlivňují kvalitu časového průběhu výstupů, jsou:

- změny intenzity záření zdroje světelného paprsku,
- nepřesnosti a vady značek na kotouči,
- posunutí vzájemné polohy kotouče a snímacích prvků,
- elektrické parametry – doby vzestupné a sestupné hrany signálu vztažené k frekvenci,
- mechanické vlivy – excentricita, axiální pohyb kotouče, vibrace.

Jednotlivé typy nepřesností a jejich příčiny podrobněji rozebereme dále.

## Změny intenzity záření zdroje světelného paprsku

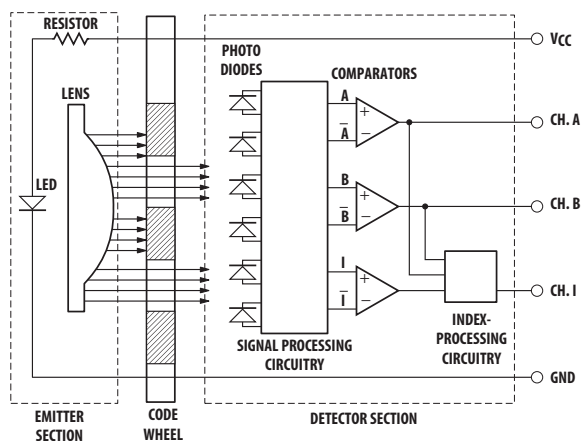
Optický přijímač (fotodetektor) převádí intenzitu dopadajícího záření na spojitý elektrický signál (napětí, proud). Logická interpretace je určena *rozhodovacími úrovněmi*. Při otáčení kotouče s ploškami přerušujícími tok světla mezi zdrojem a detektorem se intenzita záření dopadajícího na detektor nemění skokově. Proto při různé intenzitě záření zdroje odpovídá rozhodovacím úrovním různá mechanická poloha kotouče. Následkem toho se na výstupu mění šířka pulzu a šířky stavů, tj. tyto hodnoty neodpovídají ideálním  $180^\circ$  a  $90^\circ$ .

Jako zdroje světla se nejčastěji používají LED, u kterých dochází ke změnám intenzity záření při změnách teploty a vlivem stárnutí.

Tento jev může být velmi významný, [8] uvádí rozdíl v délce pulzu více než  $60^\circ$  mezi proudem LED 8 mA a 17 mA.

Tento jev je možné poměrně snadno eliminovat použitím více fotodetektorů v tzv. *push-pull* uspořádání [9]. Pro každý výstupní kanál snímače ( $A$ ,  $B$ ) je použita komplementární dvojice fotodetektorů ( $A/A'$ ,  $B/B'$ ) uspořádaných tak, že jeden je vždy zastíněn a druhý osvětlen. Rozhodovací úroveň pak není definována absolutně, ale vyhodnocuje se poměr  $A/A'$ , což právě eliminuje závislost na intenzitě záření zdroje.

Například snímače typu *AVAGO HEDS-55xx* [16] nebo odvozené *MAXON HEDL-55xx* [17] podle katalogových listů tuto metodu využívají.



Obrázek 8: Blokové schéma snímače Avago HEDS-5540. Převzato z [16].

## Nepřesnosti a vady značek na kotouči

Kvalita kotouče IRC snímače, tj. přesnost a rovnoměrnost umístění značek, je zásadním faktorem ovlivňujícím celkovou nepřesnost snímače. Navíc čím je větší počet dílků na otáčku (CPR), a tedy jemnější rozlišení, tím je určitá odchylka polohy značky vzhledem k šířce dílku relativně větší.

*Tyto odchylky ale mají systematický charakter, jsou nezávislé na vnějších vlivech jako je teplota nebo stárnutí. Lze tedy uvažovat o jejich kompenzaci v rámci zpracování signálu.*

Jiný případ jsou taková poškození kotouče, kdy dojde k zakrytí celé transparentní plošky např. vlivem prachu a nečistot. Pak dochází ke ztrátě celého pulzu. Tento problém řeší některé speciální typy snímačů s více fotodetektory pro každý výstupní kanál, přičemž jejich výstup je průměrován (viz [8], [13] – Question 35). Při zakrytí jedné nebo i více plošek pak dojde jen k malému zkreslení délky pulzu.

## Posunutí vzájemné polohy kotouče a snímacích prvků

Radiální odchylka polohy fotodetektorů vůči kotouči vede na chybu fáze. Dojde tedy k odchylce fázového posuvu  $\Phi$  a tím i délky stavů S1 až S4. I malá odchylka polohy může způsobit významnou chybu, [8] uvádí příklad pro detektor s poloměrem kotouče 11,0 mm a posunem o 0,076 mm, kdy je chyba fáze 38.7°e.

*Pokud je posunutí neměnné, jde opět o odchylku, která může být kompenzována v rámci zpracování signálu.*

## Elektrické parametry

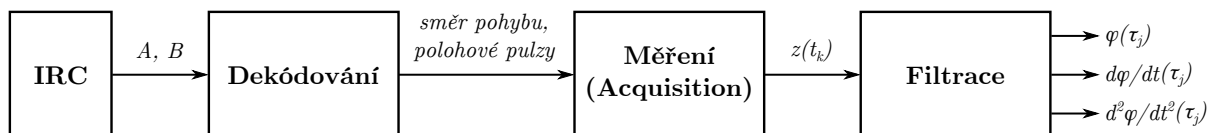
V reálných podmínkách je změna napětí elektrických signálů spojena s přechodovými jevy označovanými jako vzestupná a sestupná hrana signálu, které mají nenulovou dobu trvání.

Vzestupná a sestupná hrana mohou mít různou dobu trvání (*rise time, fall time*), což vede na chybu délky pulzu. Tyto parametry závisí hlavně na typu výstupního budiče (viz předchozí kapitola) a na délce kabelů. Nejlepší parametry z pohledu rychlosti a minimálního zkreslení délky pulzu má diferenciální typ výstupu (Line Driver, RS-422).

## **Mechanické vlivy**

Chyby mohou být způsobeny také řadou mechanických vlivů – excentricita hřídele, vůle v ložiscích, vibrace, nebo axiální pohyb kotouče. Při velkém dynamickém zatížení mohou mít významný vliv i torzní kmity hřídele snímače. Tento typ chyb nelze při zpracování signálu přímo kompenzovat, ale je třeba s jejich přítomností počítat jako s šumem měření a nějakým způsobem je filtrovat.

### 3 Metody zpracování signálu z IRC snímače



Obrázek 9: Řetězec zpracování signálu z IRC pro měření polohy, rychlosti a zrychlení.

Zpracování signálu z IRC snímače zahrnuje řetězec několika navazujících operací, viz obrázek 9. Výstupem jsou odhady polohy, rychlosti a zrychlení v ekvidistantních vzorkovacích časech  $\check{R}S - \tau_j = j \cdot T_s$ . Funkce a význam operací:

- *Dekódování* – bylo podrobně popsáno v předchozí kapitole – z kvadratických signálů A a B dekódujeme informaci o směru a polohový pulz s každou hranou (režim 4X).
- *Měření (Acquisition)* – převod výstupu dekóderu na informaci o poloze obecně označenou  $z(t_k)$ , kde metoda měření určuje, jak budou generovány časové okamžiky  $t_k$ , zda budou ekvidistantní, a zda budou synchronní se vzorkováním  $\check{R}S$ , tedy s  $\tau_j$ .
- *Filtrace* – zde pojmem filtrace myslíme obecnější matematické zpracování – filtr řeší odhad skutečné polohy a její první a druhé derivace z měření  $z(t_k)$ , které je potenciálně zatížené šumem, a případně také predikci z časové báze  $t_k$  na  $\tau_j$ .

V této kapitole popíšeme známé přístupy k řešení problému *měření*, a definujeme adaptivní metodu vhodnou pro další zpracování v bloku *filtrace*.

$L$	[-]	počet dekódovaných IRC pulzů na jednu otáčku
$f_{CLK}$	[Hz]	kmitočet CLK oscilátoru u metod T a M/T
$\varepsilon$	[s]	skutečná perioda IRC pulzů při ustálené rychlosti
$Q_\varepsilon$	[-]	relativní chyba měření periody IRC pulzů
$\omega$	[rad/s]	úhlová rychlost
$Q_\omega$	[-]	relativní chyba měření úhlové rychlosti
$t_k$	[s]	obecné časové okamžiky měření
$\tau_j$	[s]	ekvidistantní časové okamžiky vzorkování $\check{R}S$
$z(t_k)$	[rad]	měření polohy
$\Delta z$	[rad]	změna polohy odpovídající jednomu pulzu

Tabulka 1: Značení a význam použitých veličin.

Dále používané veličiny, jejich značení a význam shrnuje tabulka 1. V příkladech a grafech

je uvažován snímač s 2000 pulzy na otáčku ( $L = 2000$ ,  $CPR = 500$  pro dekódování 4X). Zatím uvažujeme ideální dekódovaný signál v podobě pulzu s každou změnou polohy o  $\Delta z$ .

## Relativní chyba metody

U každé metody určíme relativní chybu měření rychlosti  $Q_\omega$ , a její závislost na rychlosti  $\omega$ . Protože rychlost  $\omega$  je nepřímo úměrná periodě IRC pulzů  $\varepsilon$ , platí rovnost relativních chyb:

$$\varepsilon(\omega) = \frac{2\pi}{\omega L} \Rightarrow Q_\omega(\omega) = Q_\varepsilon(\omega)$$

Měříme-li periodu pulzů  $\varepsilon$  tak, že počítáme počet pulzů za nějaký čas  $T$ , je relativní chyba měření  $Q_\varepsilon$  v nejhorsím případě dána vztahem:

$$Q_\alpha = \frac{\varepsilon}{T} \quad (3.1)$$

### 3.1 Základní metody měření dekódovaného IRC signálu

Úloha měření polohových pulzů je podobná obecnější úloze měření kmitočtu. V obou případech existují 2 základní přístupy, v literatuře označované jako *M Method* a *T Method*:

- *M* – měření počtu pulzů, tj. měření změny polohy, v ekvidistantních časech, synchronně se vzorkováním ŘS, s periodou  $T_s$ .
- *T* – měření časové vzdálenosti pulzů, tj. času mezi ekvidistantními polohami.

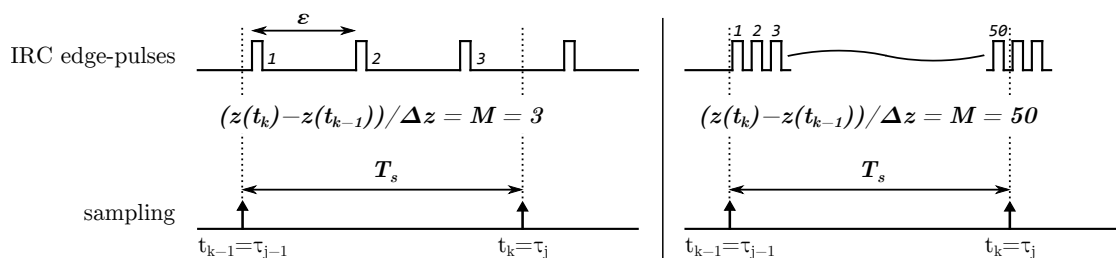
Dále podrobněji rozebereme vlastnosti metod M a T, zejména relativní chybu měření v závislosti na rychlosti. Ukážeme, že tyto metody bez dalších modifikací nejsou obecně dobře použitelné pro uvažovanou aplikaci. Pokročilejší metody z nich ale vycházejí, a v určitých pracovních oblastech vykazují podobné chování.

#### M – měření počtu IRC pulzů za konstantní čas $T_s$

Metoda M je v praxi nejpoužívanější, neboť pro její implementaci stačí čítač počtu dekódovaných pulzů respektující směr otáčení. Hodnotu čítače M čteme v každém vzorkovacím cyklu řídicího systému, tj.  $t_k = \tau_j$ , měření  $z(t_k)$  dostáváme přímo pro požadované časy.

Při malé rychlosti za daný  $T_s$  přijde jen málo IRC pulzů, což způsobuje velkou relativní chybu měření změny  $z(t_k) - z(t_{k-1})$ , jak ilustruje obrázek 10. Zlepšení je možné dosáhnout jen prodloužením  $T_s$ , což ale jde proti obecnému požadavku na co nejrychlejší vzorkování.





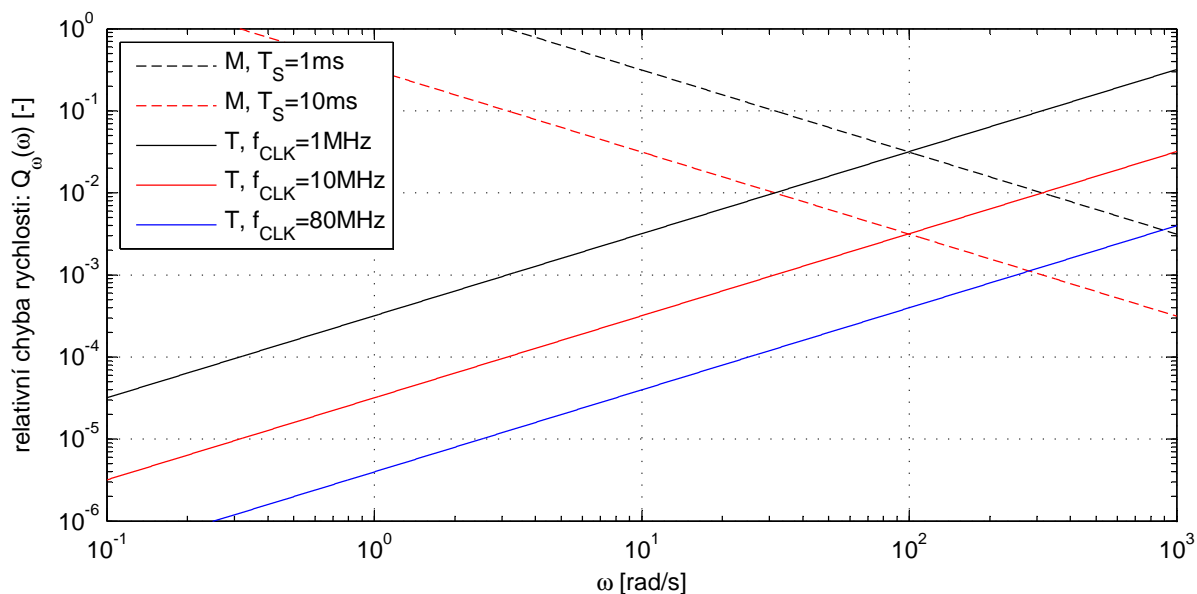
Obrázek 10: Princip funkce metody M pro malé a velké rychlosti.

Relativní chyba metody v závislosti na rychlosti je určena vztahem:

$$Q_{\omega}(\omega) = Q_{\varepsilon}(\omega) = \frac{\varepsilon(\omega)}{T_s} = \frac{2\pi}{\omega L T_s} \quad (3.2)$$

Pro představu je závislost relativní chyby  $Q_{\omega}(\omega)$  na rychlosti  $\omega$  pro konkrétní příklady  $T_s$  vynesena do grafu – obrázek 11. Graf ukazuje i srovnání s dále popsanou metodou T.

Na 1 % chyba klesne až při rychlosti 30 rad/s ( $T_s = 10$  ms), resp. 300 rad/s ( $T_s = 1$  ms).



Obrázek 11: Metody M a T – Srovnání relativní chyby měření rychlosti.

Na tomto místě ještě uvedeme konkrétní příklad vlivu chyby metody – obrázek 12.

Z grafu je patrné, že chyba měření rychlosti se pohybuje od 100 % a více pro malé rychlosti k asi 5 % při vyšších rychlostech. Měření zrychlení je naprosto nepoužitelné.

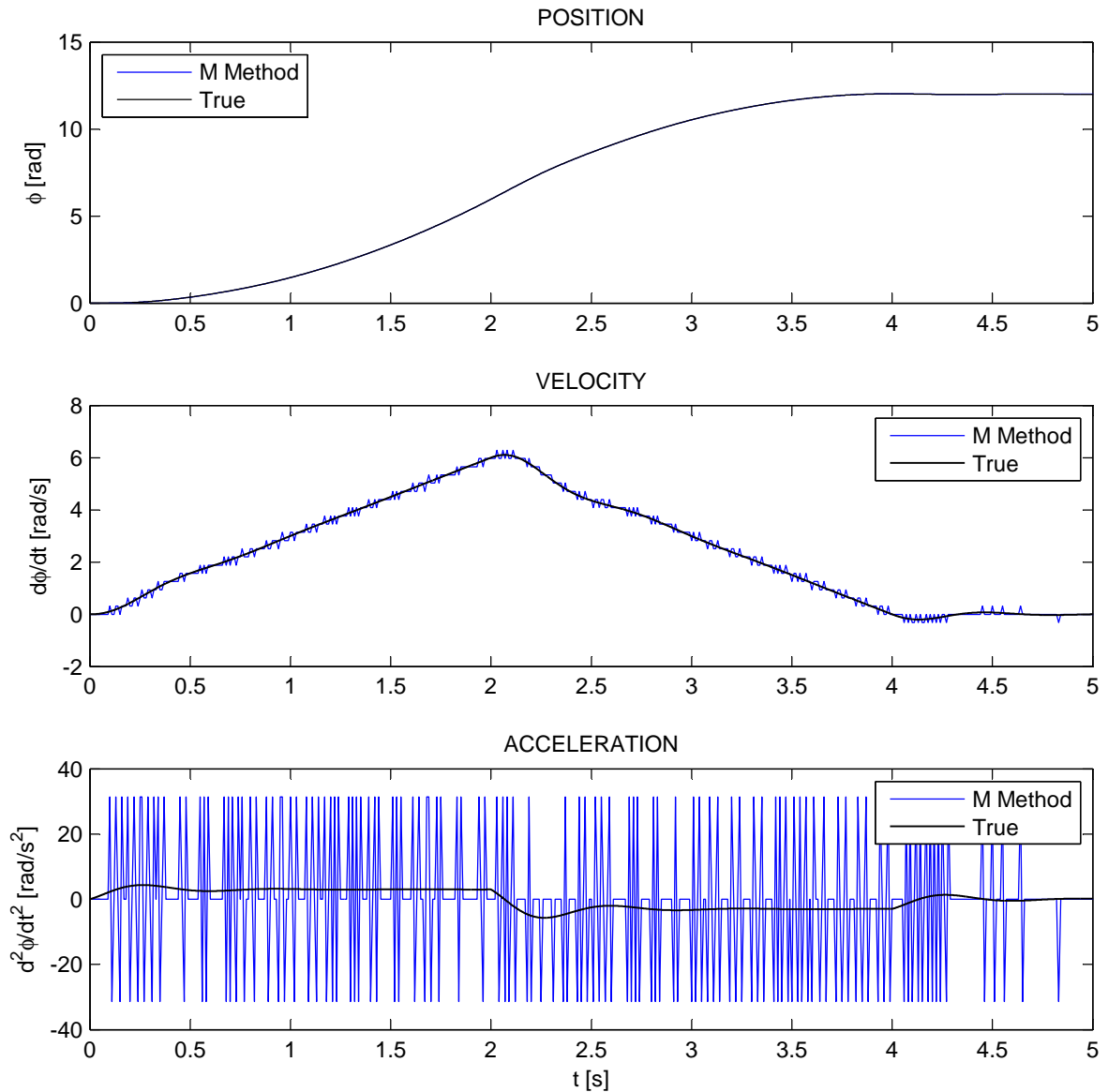
Rychlost a zrychlení počítáme metodou numerických diferencí (*finite difference*):

$$[\varphi(\tau_k) - \varphi(\tau_{k-1})] = z(t_k) - z(t_{k-1}) = M \cdot \Delta z \quad (3.3)$$

$$\dot{\varphi}(\tau_k) = \frac{[\varphi(\tau_k) - \varphi(\tau_{k-1})]}{T_s} \quad (3.4)$$

$$\ddot{\varphi}(\tau_k) = \frac{[\varphi(\tau_k) - \varphi(\tau_{k-1})] - [\varphi(\tau_{k-1}) - \varphi(\tau_{k-2})]}{T_s^2} \quad (3.5)$$

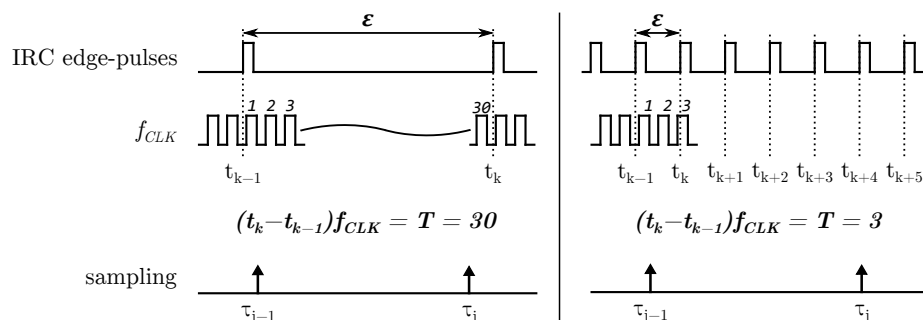
M Method Acquisition, Finite-Difference for Velocity and Acceleration,  $T_s = 10$  ms,  $L = 2000$



Obrázek 12: Metoda M – Příklad měření a vypočtené rychlosti a zrychlení.

## T – měření času mezi následujícími IRC pulzy

Metoda T je založena na měření času mezi každými dvěma po sobě jdoucími IRC pulzy. Čas měříme čítačem T jako počet pulzů z oscilátoru s kmitočtem  $f_{CLK}$ .



Obrázek 13: Princip funkce metody T pro malé a velké rychlosti.

Princip ilustruje obrázek 13. S každým IRC pulzem čteme hodnotu čítače T, tím měříme čas  $(t_k - t_{k-1})$ , za který došlo k ekvidistantní změně polohy o  $\Delta z$ . Okamžiky měření  $t_k$  nejsou synchronní se vzorkováním řídicího systému.

Pro velké rychlosti klesá čas  $\varepsilon$ , tím i odpovídající počet pulzů z  $f_{CLK}$ , a roste relativní chyba měření času  $(t_k - t_{k-1})$ . Je tedy nejlepší volit  $f_{CLK}$  co nejvyšší, což navíc nemá žádné negativní dopady na jiné parametry. Limitem jsou jen možnosti použitého hardwaru.

Relativní chyba metody v závislosti na rychlosti je určena vztahem:

$$Q_\omega(\omega) = Q_\varepsilon(\omega) = \frac{1}{\varepsilon(\omega)} \frac{1}{f_{CLK}} = \frac{\omega L}{2\pi} \frac{1}{f_{CLK}} \quad (3.6)$$

Metoda T tedy má potenciál k dosažení vysoké přesnosti měření cestou zvyšování  $f_{CLK}$ . Při velkých rychlostech se ovšem interval měření  $z(t)$  zkracuje, řádově až k  $1 \mu s$ , takže není reálné v navazujícím bloku *filtrace* stihnout zpracovat všechna měření. Zároveň roste relativní chyba měření (viz graf – obrázek 11).

Vzniká tak prostor pro další zlepšení této metody – v případě velkých rychlostí by měření přes nějaký delší časový interval, tj. „přeskočení“ některých IRC pulzů, relativní chybu snížilo. Přitom se pohybujeme v oblasti časů v řádu 0,1 ms a méně, kde pro reálné aplikace lze očekávat zanedbatelný vliv takto zvýšeného zpoždění měření. Tato myšlenka je rozvinuta v další kapitole, včetně simulace pro konkrétní případy.

### 3.2 Adaptivní metoda měření

V této kapitole rozvineme myšlenku modifikace měřicí metody T tak, aby při vyšších rychlostech neprobíhalo měření rychleji, než jsme jej schopni zpracovávat, a zároveň aby nedocházelo k nárůstu chyby měření.

Dále popsanou metodu publikoval již v roce 1982 T. Ohmae v článku [1], kde je označena jako *M/T Method*. Použitý princip bývá také označován *Time Stamping with Pulse Skip*.

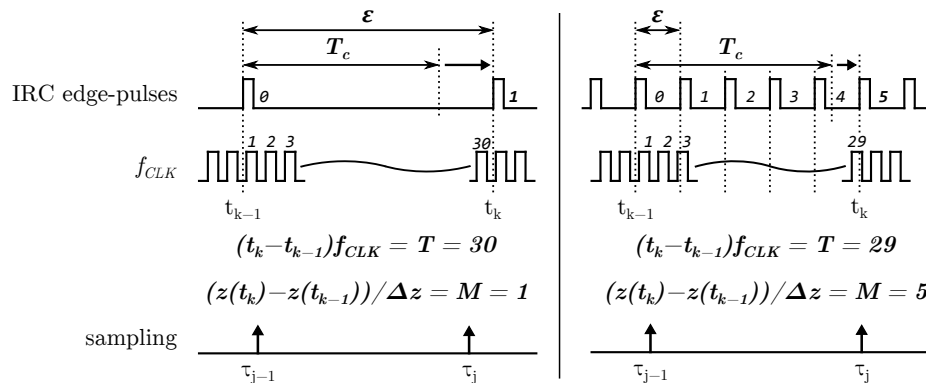
Problémem se zabývala řada dalších autorů, např. [5] (Kavanagh, 2000) – metoda *Constant Sample-time Digital Tachometer*, [6] (Tsuji, 2009) – metoda *Synchronous Measurement Method*. Z pohledu cíle této práce – získání co nejpřesnějších odhadů polohy, rychlosti a zrychlení – tyto metody proti M/T nepřinášejí zlepšení. Problémy, které řeší již na úrovni *měření*, zejména synchronizaci časových bází  $t_k$  a  $\tau_j$ , řešíme až v navazujícím bloku *filtrace*.

#### M/T – adaptivní měření [ $\delta t_k, \delta z(t_k)$ ]

Metoda měření M/T je zde označena jako adaptivní, neboť pro nízké rychlosti je shodná s metodou T, a při zvyšování rychlosti od určité hranice dochází k „přeskakování impulzů“, aby čas od předchozího měření nebyl menší než předem stanovený limit  $T_c$ :

$$t_k - t_{k-1} \geq T_c \quad (3.7)$$

Princip ilustruje obrázek 14. Při vysokých rychlostech, kdy  $\varepsilon < T_c$ , neměříme časovou diferenci mezi sousedními pulzy, tak jako u metody T, ale určitý počet pulzů je „přeskočen“. Měříme časovou diferenci mezi *nultým* pulzem a prvním pulzem následujícím po uplynutí *minimálního času měření*  $T_c$ .



Obrázek 14: Princip funkce metody M/T pro malé a velké rychlosti.

Při přeskočení pulzu vlastně provádíme průměrování několika hodnot, které by poskytla metoda T. Relativní chyba metody M/T je určena vztahy:

$$\begin{aligned} \varepsilon(\omega) \geq T_c & : Q_\varepsilon(\omega) = \frac{1}{\varepsilon} \frac{1}{f_{CLK}} = \frac{\omega L}{2\pi} \frac{1}{f_{CLK}} \\ \varepsilon(\omega) < T_c & : Q_\varepsilon(\omega) = \frac{1}{T_c + \varepsilon} \frac{1}{f_{CLK}} = \frac{1}{T_c + \frac{2\pi}{\omega L}} \frac{1}{f_{CLK}} \end{aligned} \quad (3.8)$$

Algoritmus měření metodou M/T lze popsat takto:

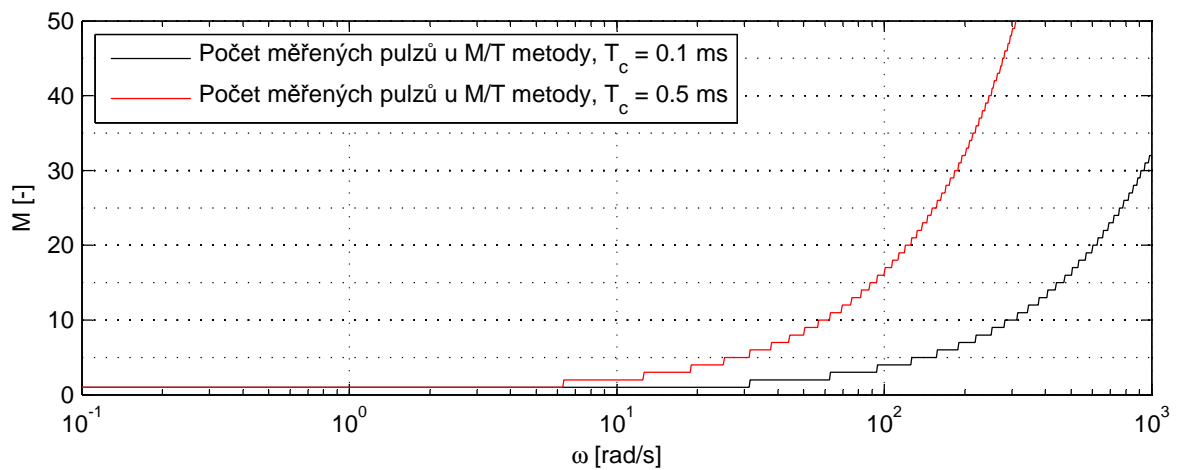
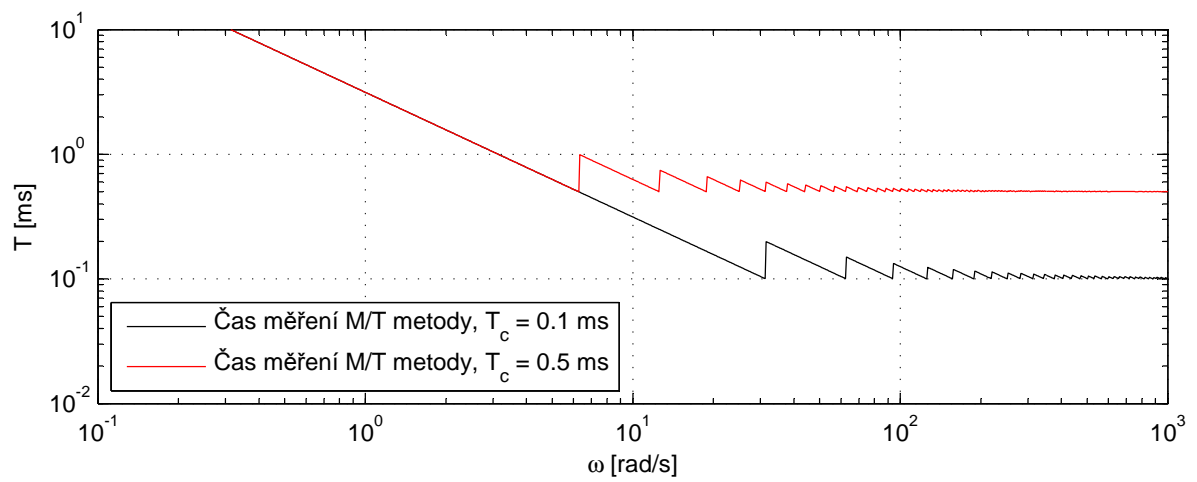
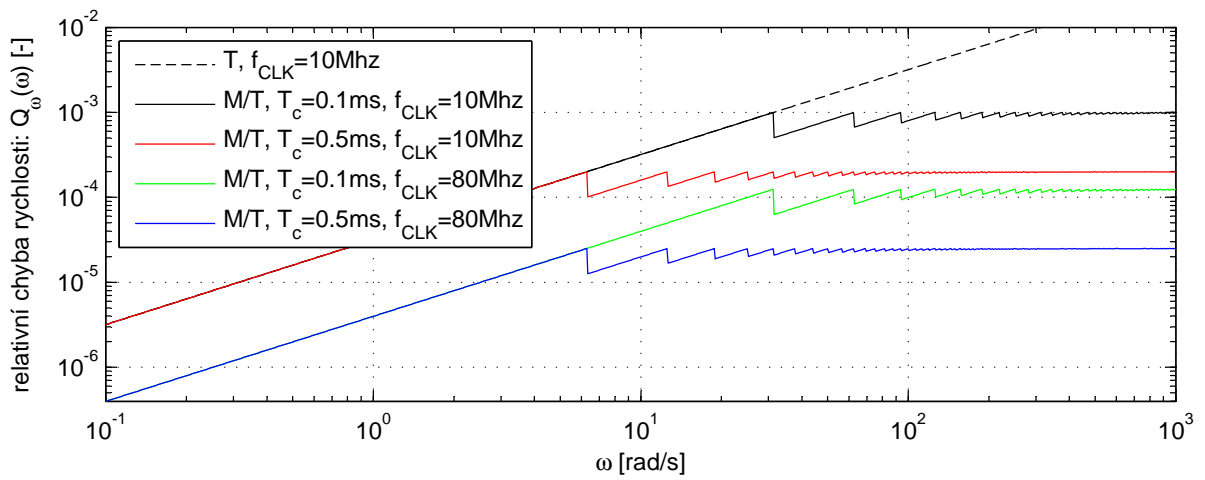
- s *nultou* hranou IRC pulzu vynulujeme čítače M a T,
- čítač M počítá IRC pulzy, čítač T počítá CLK pulzy z oscilátoru s kmitočtem  $f_{CLK}$ ,
- měření ukončíme s první náběžnou hranou IRC pulzu, která přijde po uplynutí času  $T_c$  od *nulté* hrany,
- $\delta t_k = t_k - t_{k-1} = T \cdot f_{CLK}^{-1}$  (časová diference),
- $\delta z(t_k) = z(t_k) - z(t_{k-1}) = M \cdot \Delta z$  (úhlová diference).

Takový algoritmus lze s požadovanými parametry realizovat jedině jako hardware. V původním článku [1] byly použity logické obvody vysoké integrace (LSI). V současné době lze tuto úlohu řešit použitím programovatelného hradlového pole – obvodu FPGA.

Přesnost získané hodnoty  $\delta z(t_k)$  je určena kvalitou IRC snímače, resp. jeho nedokonalostmi popsány v předchozí kapitole. Přesnost  $\delta t_k$  je určená volbou parametru  $T_c$  a kmitočtem  $f_{CLK}$ , který je limitován jen možnostmi použitého hardware.

Závislost relativní chyby  $Q_\omega(\omega)$  na rychlosti  $\omega$  včetně srovnání s metodou T ukazuje první graf na obrázku 15. V celém rozsahu rychlostí je relativní chyba shora omezená, např. pro  $T_c = 0,1$  ms a  $f_{CLK} = 80$  MHz relativní chyba nepřesahuje  $1,3 \cdot 10^{-4}$ .

Druhý graf ukazuje závislost času měření  $\delta t_k$  na rychlosti  $\omega$  – je vidět, že s rostoucí rychlostí čas měření klesá až k hranici  $T_c$ .



Obrázek 15: Metoda M/T – Relativní chyba, čas měření, závislost na parametrech.

Shrnutí vlastností M/T metody:

- v celém spektru  $\omega$  je relativní chyba měření  $Q_\omega(\omega)$  shora omezená,
- maximum  $Q_\omega(\omega)$  klesá s rostoucí  $f_{CLK}$  a s rostoucím  $T_c$  (jen pro velké  $\omega$ ),
- ve všech případech dostáváme měření časové a polohové difference  $[\delta t_k, \delta z(t_k)]$  synchronně s hranou IRC signálu, tedy bez jakéhokoli přídavného zpoždění,
- čas měření  $\delta t_k$  je závislý na rychlosti  $\omega$ ,
- čas měření  $\delta t_k$  je při konstantní rychlosti  $\omega$  také konstantní.

Parametry M/T metody:

- $f_{CLK}$  – Neovlivňuje čas měření, vyšší  $f_{CLK}$  snižuje chybu v celém spektru  $\omega$ .
- $T_c$  – Určuje minimální čas měření, vyšší  $T_c$  zvyšuje fázové zpoždění (phase-lag), ale zároveň snižuje chybu pro velké  $\omega$ .

Vhodné hodnoty parametrů pro reálné aplikace určíme později na základě simulace funkce celého řetězce zpracování signálu včetně bloku *filtrace*.

Cílem bude ověřit, že volba  $f_{CLK}$  v rozsahu cca 25 až 100 MHz, což jsou hodnoty dosažitelné na dostupném hardwaru, poskytne uspokojivé výsledky.

Čas  $T_c$  bude vhodné volit co nejmenší tak, aby blok *filtrace*, který bude pravděpodobně realizován jako softwarový algoritmus, stíhal zpracovávat příchozí měření.

V dalším textu se budeme zabývat návrhem programovatelné logiky pro realizaci M/T algoritmu, a návrhem vhodného typu filtru a jeho numerické realizace.

Blok *filtrace* se bude muset vyrovnat zejména s faktem, že výstup zde navrženého bloku *měření* je časově neekvidistantní.

## 4 Filtr pro odhad polohy, rychlosti a zrychlení

$t_k$	[s]	obecné časové okamžiky měření
$z(t_k)$	[rad]	měření polohy
$\tau_j$	[s]	ekvidistantní časové okamžiky vzorkování řídicího systému
$T_s$	[s]	perioda vzorkování, $T_s = \tau_j - \tau_{j-1}$
$T_c$	[s]	minimální čas měření, volitelný parametr měřicí metody M/T
$T_d$	[s]	čas detekce zastavení ( <i>dead-time</i> ), volitelný parametr
$\hat{x}(t_k)$		filtrační odhad stavu v čase $t_k$ na základě měření $z(t_k)$
$\tilde{x}(\tau_j)$		predikční odhad stavu pro časový okamžik $\tau_j$
$\Delta z$	[rad]	změna polohy odpovídající jednomu pulzu ze snímače

Tabulka 2: Značení a význam použitých veličin.

### 4.1 Formulace problému

V předchozí kapitole jsme definovali adaptivní metodu měření na dekodovaném IRC signálu označenou jako M/T. Tato poskytuje měření polohy (úhlu natočení)  $z(t_k)$  v obecných časových okamžicích  $t_k$ . Přitom platí:

- $T_k = t_k - t_{k-1}$  – perioda měření.
- Metoda M/T zaručuje *minimální čas měření*  $T_c$ ,  $T_k \geq T_c$ , pomocí techniky *přeskakování impulzů*.
- Časový okamžik  $t_k$  je měřen s určitou chybou, jejíž velikost je dána kmitočtem  $f_{CLK}$ .
- Chyba měření  $T_k$  závisí na volbě  $T_c$  a  $f_{CLK}$ , pro  $T_c = 0,1$  ms a  $f_{CLK} = 100$  MHz je relativní velikost chyby v nejhorším případě  $10^{-4}$ , střední hodnota je nulová.
- Chyba měřené polohové difference ( $z(t_k) - z(t_{k-1})$ ) je určena nepřesností samotného snímače. Relativní velikost chyby závisí na mnoha faktorech, a může být v rozsahu asi  $10^{-3}$  až  $5 \cdot 10^{-1}$ , střední hodnota je nulová. Část této chyby může být systematická, a tato by mohla být kompenzována po kalibraci na konkrétní kus snímače.

V této kapitole provedeme návrh *filtru*, jehož úkolem je z měření  $z(t_k)$  získat *odhady* polohy  $\varphi(\tau_j)$  a její první a druhé derivace, tj. rychlosti  $\dot{\varphi}(\tau_j)$  a zrychlení  $\ddot{\varphi}(\tau_j)$ .



Filtr řeší dva dílčí problémy:

- *odhad derivací* měřeného signálu zatíženého šumem,
- *časovou predikci* od  $t_k$  k  $\tau_j$ ,  $t_k \leq \tau_j$ .

Pro další postup definujeme *stav*  $x(t)$  jako vektor skutečné polohy, rychlosti a zrychlení:

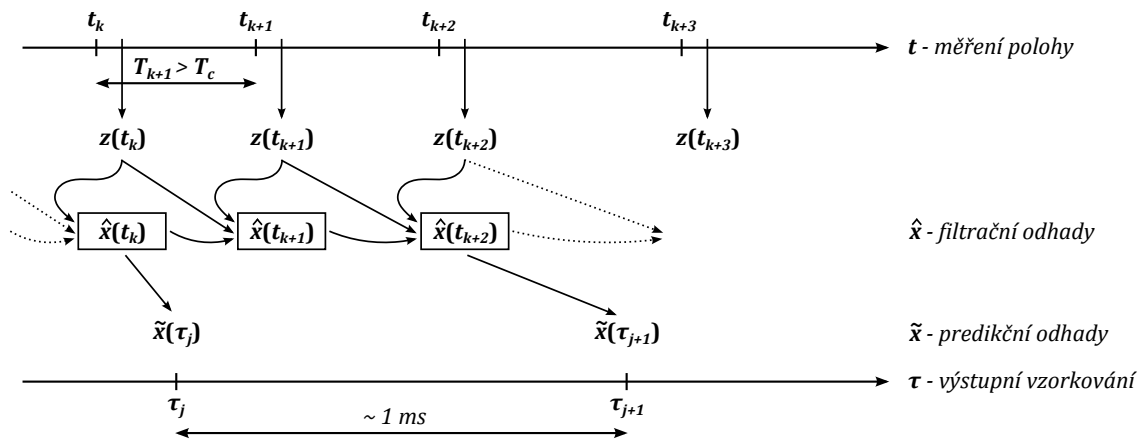
$$x(t) = [ \varphi(t), \dot{\varphi}(t), \ddot{\varphi}(t) ]^T \quad (4.1)$$

Vzhledem k požadované aplikaci v uzavřené smyčce řídicího systému musí filtr pracovat v *on-line* režimu. Odhady pro určitý čas tedy musí být založeny jen na aktuální a předchozí informaci, ne na informaci budoucí.

Návrh rozdělíme na dva bloky:

1. S každým měřením  $z(t_k)$  vypočteme na základě tohoto měření, předchozího měření  $z(t_{k-1})$ , a předchozího odhadu  $\hat{x}(t_{k-1})$  nový *filtrační odhad* stavu  $\hat{x}(t_k)$ .
2. Pro každý vzorkovací okamžik  $\tau_j$  vypočteme z posledního známého odhadu  $\hat{x}(t_k)$  *predikční odhad* stavu  $\tilde{x}(\tau_j)$ .

Časové souvislosti měření, výpočtu odhadů, a výstupního vzorkování ukazuje obrázek 16.



Obrázek 16: Časové souvislosti a struktura výpočtu filtračních a predikčních odhadů.

## 4.2 Odhad derivací

Derivace signálů zatížených šumem není snadno řešitelný problém. Použití metody numerické diference (*finite difference*) nepřináší uspokojivé výsledky, neboť i relativně malý šum měření vede na velkou poruchu derivace. Tento jev jsme ukázali v předchozí kapitole na konkrétním příkladu, kde se výpočet druhé derivace ukázal jako zcela nepoužitelný.

V praxi se používají derivační články, které potlačují vliv těchto poruch, což ovšem vede na nezanedbatelné zhoršení kvality takto získaného odhadu derivace.

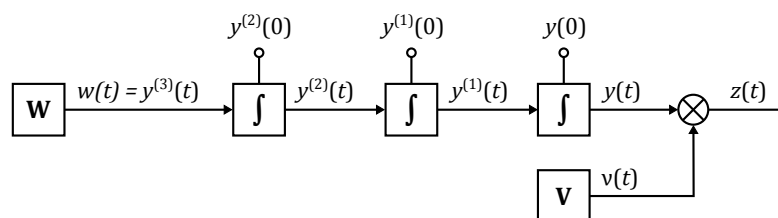
Lepších výsledků je možné dosáhnout použitím Kalmanova filtru. Dále popsané řešení vychází z článku [7] (Schlegel, Večerek, 2002).

Pro aplikaci Kalmanova filtru jako rekonstruktoru stavu lineárního systému, je třeba tento systém – model generátoru signálu – nějak definovat. Odvození provedeme ve spojitém popisu, který později diskretizujeme. Kalmanův filtr navíc kromě odhadu derivací poskytne i odhad samotné polohy očištěné od šumu.

Pro další postup je třeba přijmout určité předpoklady o měřeném signálu:

- Měření  $z(t)$  je součtem užitečného signálu  $y(t)$  a šumu  $\vartheta(t)$ .
- Měření  $z(t)$  je jednorozměrné a spojitě v čase.
- Porucha  $\vartheta(t)$  je spojitý bílý šum s nulovou střední hodnotou a kovariancí  $R$ .
- Užitečný signál  $y(t)$  je spojitý a dostatečně hladký, takže existuje jeho 3. derivace, která je spojitá.
- Užitečný signál  $y(t)$  je *lokálně aproximovatelný polynomem řádu 3*.

Za těchto předpokladů lze generátor signálu  $y(t)$  modelovat jako řetězec 3 sériově spojených integrátorů. Vstupem prvního integrátoru je spojitý bílý šum  $w(t)$ , pomocí kterého je do modelu vnesena neurčitost, která rozšiřuje množinu signálů, které je schopen generovat.



Obrázek 17: Generátor měření  $z(t)$ .

Generátor měřeného signálu  $z(t)$  je pak modelován lineárním stochastickým systémem:

$$\begin{aligned}\dot{x}(t) &= A \cdot x(t) + G \cdot w(t) \\ y(t) &= C \cdot x(t) \\ z(t) &= y(t) + \vartheta(t)\end{aligned}\tag{4.2}$$

kde:

$$x(t) = \begin{bmatrix} \varphi(t) \\ \dot{\varphi}(t) \\ \ddot{\varphi}(t) \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C = [1 \quad 0 \quad 0]$$

$w(t)$  – bílý šum s kovariancí  $Q$ , udává délku intervalu, na kterém lze  $y(t)$  dobře aproximovat polynomem řádu 3;

$\vartheta(t)$  – bílý šum s kovariancí  $R$ , udává míru důvěry v měření  $z(t)$ .

### Kalmanův filtr

Rekonstruktor stavu pro systém (4.2) popíšeme rovnicemi odpovídajícími modelu systému bez šumu doplněnému o inovační zpětnou vazbu (Kalmanův zisk)  $K$ :

$$\begin{aligned}\dot{\hat{x}}(t) &= A \cdot \hat{x}(t) + G \cdot w(t) + K \cdot (y(t) - \hat{y}(t)) \\ \hat{y}(t) &= C \cdot \hat{x}(t)\end{aligned}\tag{4.3}$$

Šumy  $w$  a  $\vartheta$  mají nulovou střední hodnotu, budeme uvažovat:

$$E[w(t)] = 0 \quad \rightarrow \quad G \cdot w(t) \approx 0\tag{4.4}$$

$$E[\vartheta(t)] = 0 \quad \rightarrow \quad y(t) \approx z(t)\tag{4.5}$$

Po dosazení a úpravě dostáváme *dynamickou rovnici rekonstruktoru*:

$$\dot{\hat{x}}(t) = A \cdot \hat{x}(t) + K \cdot (z(t) - C \cdot \hat{x}(t))\tag{4.6}$$

kde vektor  $K$  – Kalmanův zisk v ustáleném stavu – dostaneme řešením Riccatiový rovnice:

$$AP_N + P_N A^\top - P_N C^\top R^{-1} C P_N + G Q G^\top = 0\tag{4.7}$$

kde  $P_N$  je neznámá kovarianční matice stavu  $\hat{x}$  Kalmanova filtru (4.6), matice  $A$ ,  $C$  a  $G$  jsou matice generátoru měřené hodnoty,  $Q$  a  $R$  jsou známé skalární kovariance šumů  $w(t)$  a  $\vartheta(t)$ .

Díky vhodné volbě předpokladů o měřeném signálu bylo možné najít analytické řešení této rovnice, viz [7]. Vychází velmi jednoduchý tvar závislý na parametru  $\alpha$ :

$$K = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \begin{bmatrix} 2e^{\alpha/6} \\ 2e^{\alpha/3} \\ e^{\alpha/2} \end{bmatrix}, \quad \alpha = \ln\left(\frac{Q}{R}\right) \quad (4.8)$$

Parametr  $\alpha$  je určen poměrem kovariancí  $Q$  a  $R$ , které ve skutečnosti neznáme. Protože jde o jediný parametr, není problém jej nastavit experimentálně.

Rovnici Kalmanova filtru lze dále zjednodušit:

$$\dot{\hat{x}}(t) = A_R \cdot \hat{x}(t) + K \cdot z(t) \quad (4.9)$$

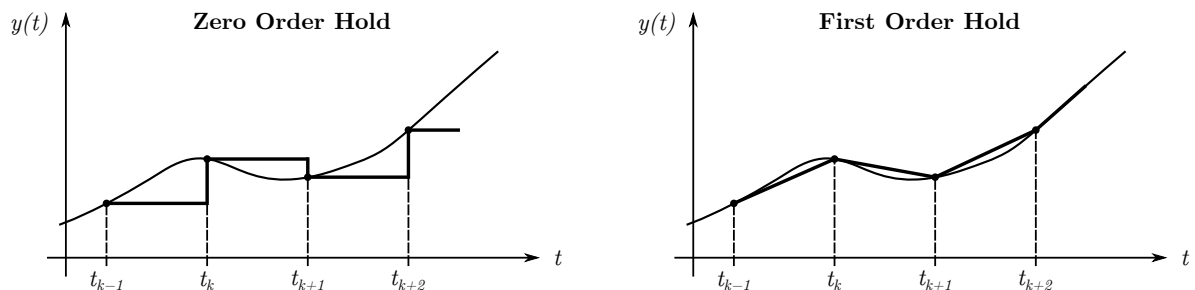
$$A_R = A - KC = \begin{bmatrix} -k_1 & 1 & 0 \\ -k_2 & 0 & 1 \\ -k_3 & 0 & 0 \end{bmatrix} \quad (4.10)$$

## Diskretizace

V předchozím textu jsme dospěli ke spojitému popisu Kalmanova filtru pro výpočet odhadu užitečného signálu a jeho první a druhé derivace, včetně řešení pro Kalmanův zisk  $K$ . Původní problém je ale formulován v diskretních časech, provedeme tedy diskretizaci.

Řešení komplikuje fakt, že intervaly mezi měřeními  $T_k$  jsou variabilní, v řádu 0,1 ms až 10 ms nebo i více při malých rychlostech. Měření tedy může být časově řídké, a filtr musí být schopen v něm obsaženou informaci co nejlépe využít.

Základní metoda ZOH (Zero Order Hold), která aproximuje signál po částech konstantní funkcí se skokovými změnami ve okamžicích  $t_k$  (viz obrázek 18), nebude v tomto případě dávat dobré výsledky. Odhad pro čas  $t_k$  by totiž vůbec nezahrnoval aktuální měření  $z(t_k)$ .



Obrázek 18: Diskretizační metody ZOH a FOH.

Vhodné bude použít metodu FOH (First Order Hold), která aproximuje měřený signál po částech lineární funkcí:

$$z(t \in \langle t_{k-1}, t_k \rangle) = z(t_{k-1}) + [t - t_{k-1}] \cdot \frac{z(t_k) - z(t_{k-1})}{t_k - t_{k-1}} \quad (4.11)$$

Úpravou rovnice Kalmanova filtru (4.9), kdy zatím uvažujeme  $t \in \langle 0, T \rangle$ , dostaneme:

$$\dot{\hat{x}}(t) = A_R \cdot \hat{x}(t) + K \cdot \left[ z(0) + t \cdot \frac{z(T) - z(0)}{T} \right] \quad (4.12)$$

$$\dot{\hat{x}}(t) = A_R \cdot \hat{x}(t) + K \cdot z(0) + Kt \cdot \frac{z(T) - z(0)}{T} \quad (4.13)$$

Diskretizace pomocí FOH a posun do časové oblasti  $\langle t_{k-1}, t_k \rangle$ :

$$\hat{x}(t_k) = \Phi_k \cdot \hat{x}(t_{k-1}) + \Gamma_k \cdot z(t_{k-1}) + \Pi_k \cdot \frac{z(t_k) - z(t_{k-1})}{t_k - t_{k-1}} \quad (4.14)$$

kde:

$$\begin{aligned} T_k &= t_k - t_{k-1} \\ \Phi_k &= e^{A_R T_k} \end{aligned} \quad (4.15)$$

$$\Gamma_k = \int_0^{T_k} e^{A_R(T_k - \tau)} d\tau \cdot K \quad (4.16)$$

$$\Pi_k = \int_0^{T_k} e^{A_R(T_k - \tau)} \tau d\tau \cdot K \quad (4.17)$$

Koeficienty  $\Phi_k$ ,  $\Gamma_k$  a  $\Pi_k$  bude třeba počítat při každém měření, protože  $T_k$  není konstantní. Další postup bude směřovat k nalezení vhodných aproximací a numerických metod pro uspokojivě rychlý výpočet.

### 4.3 Časová predikce

Zbývá vyřešit přechod od aktuálního odhadu stavu  $\hat{x}(t_k)$  k odhadu stavu ve známém budoucím vzorkovacím časovém okamžiku  $\tilde{x}(\tau_j)$ ,  $\tau_j > t_k$ , tj. problém časové predikce.

V tomto případě nepracujeme s žádným měřením. Provádíme prostou predikci stavu lineárního systému (4.2):

$$\tilde{x}(\tau_j) = \Theta_{jk} \cdot \hat{x}(t_k); \quad \Delta_{jk} = \tau_j - t_k; \quad \Delta_{jk} > 0 \quad (4.18)$$

$$\Theta_{jk} = e^{A \cdot \Delta_{jk}} = \begin{bmatrix} 1 & \Delta_{jk} & \frac{1}{2} \Delta_{jk}^2 \\ 0 & 1 & \Delta_{jk} \\ 0 & 0 & 1 \end{bmatrix} \quad (4.19)$$

Zde existuje použitelné algebraické řešení výrazu  $e^{A \cdot \Delta_{jk}}$ , takže bude stačit dosazovat  $\Delta_{jk}$  do známého tvaru matice  $\Theta_{jk}$  (4.19). Výpočet výstupního odhadu tedy bude velmi rychlý. Řešení lze popsat tak, že po dobu  $\Delta_{jk}$  uvažujeme konstantní zrychlení. Při malé rychlosti nebo dokonce při úplném zastavení ovšem  $\Delta_{jk}$  roste, a je-li poslední vypočtené zrychlení nenulové, bude predikovaná poloha a rychlost s rostoucím časem divergovat k nesmyslným hodnotám.

K řešení tohoto problému uvedeme několik úvah a pozorování z dále popsaných simulací.

- Pokud po určitý čas  $\Delta_{jk}$  zatím *nepřišlo* měření  $z(t_k)$ , je to také užitečná informace. Víme, že se v tomto čase skutečná poloha nevychýlila z intervalu odpovídajícímu sousedním značkám snímače. Zde využijeme informaci o *směru* otáčení  $d(t_k)$  při posledním měření pro stanovení *hraničního intervalu*:
  - $d(t_k) = -1 \rightarrow \langle z(t_k) - \Delta z, z(t_k) \rangle$
  - $d(t_k) = +1 \rightarrow \langle z(t_k), z(t_k) + \Delta z \rangle$

- Při selhání predikce, tj. při vybočení predikované polohy z hraničního intervalu, zafixujeme odhad polohy na krajní hodnotě intervalu, která byla překročena.

Odhady rychlosti a zrychlení je pak také třeba nějakým způsobem korigovat. Jako jednoduché a dobře použitelné řešení se ukázalo tyto hodnoty po nějakou dobu ponechat konstantní, a následně je vynulovat, když  $\Delta_{jk} > T_d$  (volitelný parametr).

Je však potřeba odlišit selhání predikce od vlastní dynamiky filtru, který s cílem redukce šumu může generovat odhady polohy vybočující z hraničního intervalu, a přitom „správné“. Jako dobré řešení se ukázalo omezit detekci selhání na  $\Delta_{jk} > 10T_s$ , tedy na situaci, kdy nepřišlo žádné měření po dobu deseti vzorkovacích period.

- *Detekce zastavení*: pokud  $\Delta_{jk} > T_d$ , považujeme systém za *zastavený* nezávisle na tom, jestli bylo detekováno selhání predikce. V predikčním odhadu zafixujeme složku polohy a vynulujeme složky rychlosti a zrychlení. Zároveň bude vhodné vynulovat složky rychlosti a zrychlení i ve filtračním odhadu  $\hat{x}(t_k)$ .

## 4.4 Numerická implementace výpočtů

Výpočet  $\Phi_k$

$$\Phi_k = e^{A_R T_k} = \text{expm}(A_R T_k) \quad (4.20)$$

Numerický výpočet maticové exponenciály lze nejlépe realizovat použitím Padého aproximace s technikou měřítkování a umocňování (*Padé Approximation with Scaling and Squaring*). Tato metoda je popsána např. v [10]. *Padé approximant*  $[k, m]$  funkce  $f(A)$  je racionální funkce – podíl polynomů  $p_{km}$  stupně  $k$  a  $q_{km}$  stupně  $m$ :

$$f(A) \approx r_{km}(A) = [q_{km}(A)]^{-1} p_{km}(A) \quad (4.21)$$

Pro maticovou exponenciální funkci  $f(A) = e^A$  je tvar polynomů znám:

$$p_{km}(A) = \sum_{j=0}^k \frac{(k+m-j)!k!}{(k+m)!(k-j)!j!} A^j, \quad q_{km}(A) = \sum_{j=0}^m \frac{(k+m-j)!k!}{(k+m)!(k-j)!j!} (-A)^j \quad (4.22)$$

Stupně polynomů volíme shodné ( $k = m$ ), v závislosti na požadované přesnosti.

Aproximace  $e^A$  dává dobré výsledky pouze pro malé  $\|A\|$ . Technika měřítkování a umocňování řeší tento problém tak, že matice  $A$  je před výpočtem aproximantu vydělena koeficientem  $2^s$  tak, aby  $\|A\| < 1$ . Výsledek aproximace  $e^{A/2^s}$  je  $s$ -krát umocněn na druhou:

$$e^A = \left(e^{A/2^s}\right)^{2^s}, \quad e^A \approx \left(r_{km}(A/2^s)\right)^{2^s} \quad (4.23)$$

V literatuře byly popsány a podrobně analyzovány různé algoritmy výpočtu z pohledu přesnosti a rychlosti, neboť jde o problém náročný na výpočetní výkon. Zde popsané řešení vychází z algoritmu popsaného v článku [11], kde bylo provedeno srovnání s alternativními řešeními, včetně funkce `expm` ze systému MATLAB [26].

Algoritmus je založen na řešení maticové rovnice  $q_m(A)r_m(A) = p_m(A)$ , přičemž platí  $q_m(A) = p_m(-A)$ . Položíme  $p_m(A) = \sum_{i=0}^m b_i A^i$  a pro liché  $m$  dostáváme:

$$\begin{aligned} p_{2m+1}(A) &= A \left[ b_{2m+1} A^{2m} + \dots + b_3 A^2 + b_1 I \right] + \left[ b_{2m} A^{2m} + \dots + b_2 A^2 + b_0 I \right] \\ &= U + V \end{aligned} \quad (4.24)$$

$$\begin{aligned} q_{2m+1}(A) &= -A \left[ b_{2m+1} A^{2m} + \dots + b_3 A^2 + b_1 I \right] + \left[ b_{2m} A^{2m} + \dots + b_2 A^2 + b_0 I \right] \\ &= -U + V \end{aligned} \quad (4.25)$$

Hodnoty koeficientů  $b_i$  získáme ze vztahu (4.22), kromě toho je potřeba jen vypočítat sudé mocniny  $A$  od  $A^2$  do  $A^{2m}$ .

Volba řádu aproximace  $m$  byla podrobně rozebrána v původním článku. Pro naši aplikaci vychází nejlépe  $m = 9$ . Potom pro měřítkování platí:  $\|A\|_1 < \theta_9 = 2,0978$ .

Algoritmus lze zapsat takto:

```

1  if norm(A, 1) > 2.0978
2    s = ceil(log2(norm(A, 1) / 2.0978))
3    A = A / 2^s
4  else
5    s = 0
6  end
7
8  A2 = A^2, A4 = A2*A2, A6 = A2*A4, A8 = A4^2
9
10 U = A*(b9*A8 + b7*A6 + b5*A4 + b3*A2 + b1*I)
11 V = b8*A8 + b6*A6 + b4*A4 + b2*A2 + b0*I
12
13 E = (-U+V) \ (U+V)
14
15 for k = 1:s; E = E*E; end

```

Tento algoritmus byl otestován v systému MATLAB, bylo provedeno srovnání s vestavěnou funkcí `expm` a s hodnotami vypočtenými s přesností na 100 číslic pomocí *Symbolic Math Toolbox*. Test byl proveden s maticí  $A_R T$  (4.9) pro  $\alpha = 15, 20, 25, 30$  a pro  $T = 10^{-4} \dots 1$ .

Algoritmus je přibližně o 50 % rychlejší než vestavěná funkce `expm` a pro testované případy dává stejné nebo lepší výsledky. Srovnání pro  $\alpha = 25$  ukazuje obrázek 19.

## Výpočet $\Gamma_k$

Argument integrálu je exponenciální funkce závislá jen na  $\tau$ , lze najít analytické řešení:

$$\begin{aligned}
 \Gamma_k &= \int_0^{T_k} e^{A_R(T_k - \tau)} d\tau \cdot K = A_R^{-1} (e^{A_R T_k} - I) K = \\
 &= A_R^{-1} (\Phi_k - I) K
 \end{aligned} \tag{4.26}$$

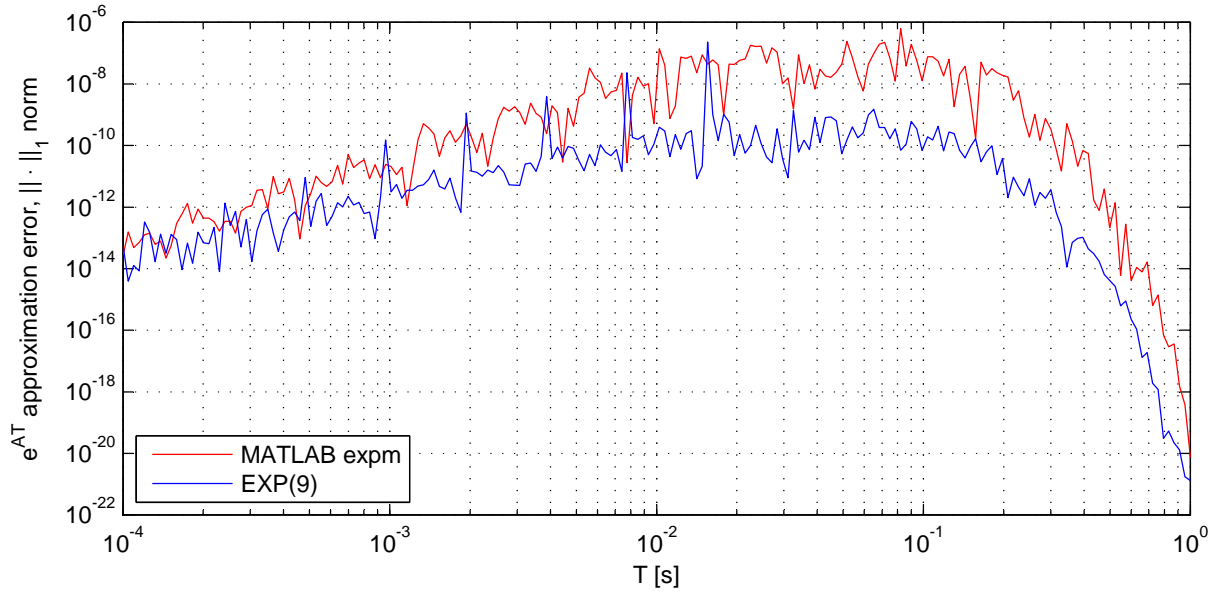
Kromě maticového násobení tedy potřebujeme vypočítat jen inverzi matice  $A_R$ , která je navíc nezávislá na  $T_k$ , a tedy shodná pro všechna měření. Pro numerické řešení může být užitečné užitečné úlohu přepsat jako řešení soustavy rovnic:

```

1  Gamma_k = A_R \ ((Phi_k - eye(3)) * K)

```





Obrázek 19: Chyba aproximace popsaného algoritmu a vestavěné MATLAB funkce, použita matice  $A_R T$ ,  $\alpha = 25$ .

### Výpočet $\Pi_k$

Zde je argument integrálu proti  $\Gamma_k$  navíc násoben  $\tau$ , přesto lze nalézt analytické řešení:

$$\begin{aligned} \Pi_k &= \int_0^{T_k} e^{A_R(T_k-\tau)} \tau d\tau \cdot K = e^{A_R T_k} \cdot \int_0^{T_k} e^{-A_R \tau} \tau d\tau \cdot K = \\ &= \Phi_k A_R^{-1} A_R^{-1} [I - e^{-A_R T_k} (A_R T_k + I)] \end{aligned} \quad (4.27)$$

Přitom inverzi  $A_R^{-1}$  již máme vypočtenou z předchozího kroku. Výraz  $e^{-A_R T_k}$  je vlastně  $\Phi_k$  s negativním argumentem. Tuto hodnotu můžeme efektivně získat při výpočtu  $\Phi_k$  tak, že rovnici  $q_m(A)r_m(A) = p_m(A)$  vyřešíme s přehozeným  $q_m$  a  $p_m$ , protože  $q_m(A) = p_m(-A)$ .

Implementaci Padého aproximace tedy bude vhodné rozšířit tak, aby výpočet vracel maticovou exponenciálu zároveň pro kladný i záporný argument:

```

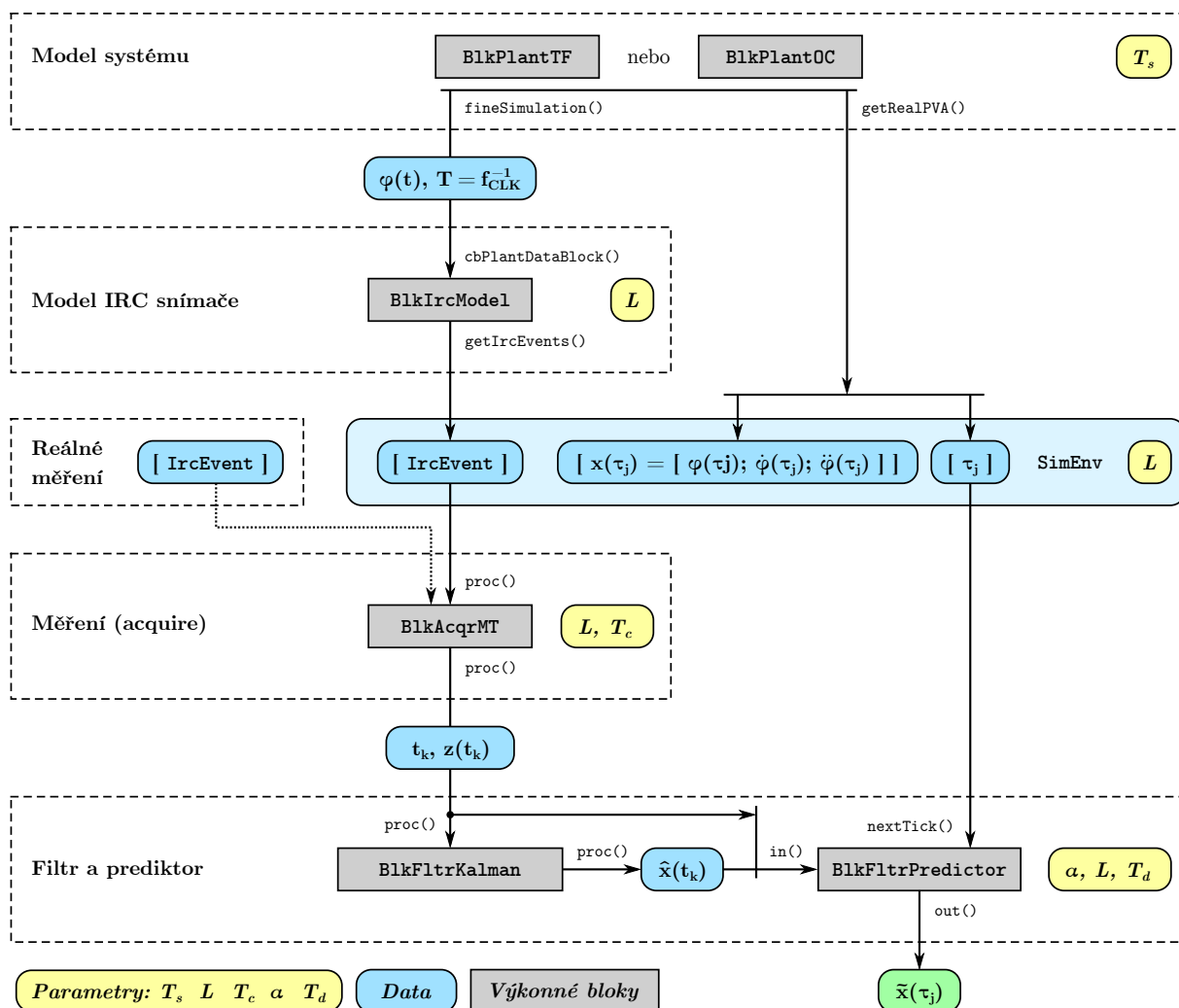
1 ...
2
3 E_Neg = (U+V) \ (-U+V)
4
5 for k = 1:s; E_Neg = E_Neg*E_Neg; end

```

## 5 Simulace v systému MATLAB

Pro otestování funkce navržených algoritmů byla jako součást této práce vytvořena sada skriptů pro MATLAB, která umožňuje simulovat celý řetězec zpracování signálu a vyhodnotit kvalitu získaných výsledků.

Celkovou strukturu simulačního prostředí ukazuje obrázek 20. Dále popíšeme koncepci zvoleného řešení a uvedeme podrobnější popis jednotlivých částí.



Obrázek 20: Struktura realizovaného simulačního prostředí.

Základní struktura simulačního prostředí je následující:

- Simulace měřeného systému a IRC snímače
  - *Model měřeného systému* – generuje „skutečné“ hodnoty polohy, rychlosti a zrychlení v požadovaných vzorkovacích okamžicích.  
Jsou implementovány dvě varianty modelu: 1) přenosová funkce  $u \rightarrow [\varphi, \dot{\varphi}, \ddot{\varphi}]$ ; 2) generátor časově optimální trajektorie pro řízení řetězce 3 integrátorů s omezením na vstup a stav (derivaci zrychlení, zrychlení, rychlost).
  - *Model IRC snímače* – generuje polohové pulzy (*edge-pulses*) na základě informace o změnách polohy v čase dodané z modelu systému, přičemž je použito velmi jemné vzorkování s periodou  $f_{CLK}^{-1}$  ( $10^{-6}$  až  $10^{-8}$  s). Model umožňuje simulovat i nedokonalost snímače – viz popis bloku *BlkIrcModel* dále.
- Simulace měřicí metody M/T a filtru
  - *Měření (acquire)* – algoritmem M/T generuje měření  $z(t_k)$  z polohových pulzů, které mohou být dodány z výstupu simulace měřeného systému nebo z měření na reálném snímači.
  - *Filtr pro odhad derivací a prediktor* – z měření  $z(t_k)$  generuje požadované odhady polohy, rychlosti a zrychlení  $\tilde{x}(\tau_j)$ .

Ve výsledku je tak možné porovnat odhady  $\tilde{x}(\tau_j)$  se „skutečnými“ hodnotami  $x(\tau_j)$ .

Zdrojové kódy lze rozdělit na:

- *Výkonné funkční bloky* – obsahují vlastní algoritmy s jasně ohraničenými vstupy a výstupy, řešené jako objektové třídy (*Class M-File*) – soubory **Blk\*.m**.
- *Pomocné funkce* – např. funkce **expm92** pro výpočet maticové exponenciely.
- *Simulační funkce* – určeny pro přímé spouštění uživatelem z *MATLAB Workspace*, zajišťují inicializaci a propojení výpočetních bloků pro spuštění simulace a případně vizualizaci výsledků v podobě grafů – soubory **xm\*.m** a **xs\*.m**.

V dalším textu je základním způsobem popsána funkce jednotlivých bloků a jsou uvedeny příklady spuštění simulačních funkcí včetně výsledných grafů. Vyhodnocení výsledků získaných ze simulačního systému je pak provedeno v následující kapitole.

Kompletní zdrojové soubory jsou na příloženém CD. Vývoj byl realizován na verzi MATLAB R2009b.

## 5.1 Výkonné třídy – funkční bloky

### BlkBase – Bázová třída pro funkční bloky

Od této třídy dědí všechny ostatní bloky – třídy **Blk\***. Obsahuje funkce pro ladicí výpisy.

Atributy:

- **name** – název bloku pro ladicí výpis, atribut by měl být nastaven v konstruktoru.

Funkce:

- **profile(arg)** – volání s **arg=1** aktivuje sledování času; volání s **arg=0** ukončí sledování času; volání bez parametru vrací reálný čas v sekundách od aktivace sledování.
- **msg(msg)** – vypíše název bloku dle atributu **name**, čas od spuštění sledování funkcí **profile()** (je-li sledování aktivní) a předanou zprávu **msg**.

### BlkPlantTF – Model systému definovaného přenosovou funkcí

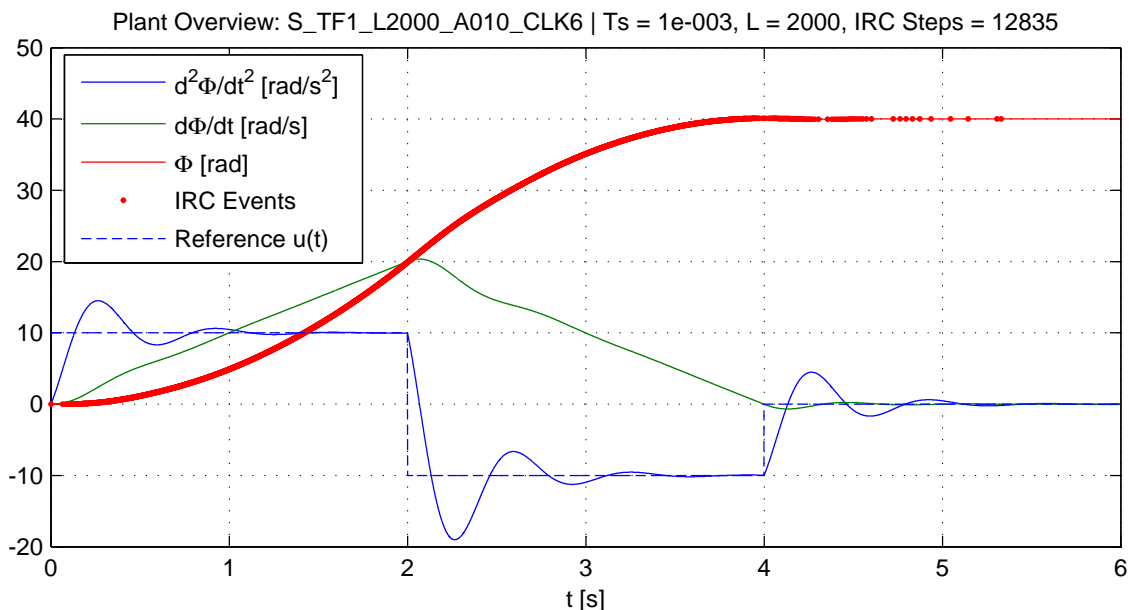
Implementace modelu měřeného systému definovaného přenosovými funkcemi:

$$\frac{\mathcal{L}\{\ddot{\varphi}(t)\}}{\mathcal{L}\{u(t)\}} = F(s), \quad \frac{\mathcal{L}\{\dot{\varphi}(t)\}}{\mathcal{L}\{u(t)\}} = \frac{1}{s}F(s), \quad \frac{\mathcal{L}\{\varphi(t)\}}{\mathcal{L}\{u(t)\}} = \frac{1}{s^2}F(s) \quad (5.1)$$

kde  $u(t)$  je referenční hodnota zrychlení a  $F(s)$  je přenos systému od reference ke zrychlení. Tento model byl pro srovnání převzat z [3], kde byl použit přenos:

$$F(s) = \frac{6s + 100}{s^2 + 6s + 100} \quad (5.2)$$

Příklad časových průběhů polohy, rychlosti a zrychlení pro model s uvedeným přenosem je na obrázku 21. Budicí signál  $u(t)$  je po částech konstantní a je také vyznačen v grafu.



Obrázek 21: Příklad profilu časového profilu zrychlení, rychlosti a polohy z BlkPlantTF.

Funkce:

- `BlkPlantTF(f, t, u)` – inicializace; `f`: přenos  $F(p)$  jako hodnota vrácená funkcí `tf()`; `t`: vektor časových okamžiků  $\tau_j$ ; `u`: vektor referenčních hodnot  $u(\tau_j)$ .
- `x = getRealPVA()` – vypočte skutečné hodnoty výstupů v časech  $\tau_j$  ve formátu `x: [ [  $\tau_1$ ;  $\varphi(\tau_1)$ ;  $\dot{\varphi}(\tau_1)$ ;  $\ddot{\varphi}(\tau_1)$  ] [  $\tau_2$ ;  $\varphi(\tau_2)$ ;  $\dot{\varphi}(\tau_2)$ ;  $\ddot{\varphi}(\tau_2)$  ] .. ]`.
- `fineSimulation(ddt, cb)` – provede simulaci s jemným časovým krokem `ddt`; to může znamenat např.  $10^8$  hodnot na 1 s času, takže z důvodu paměťové náročnosti a kumulace chyb není možné provést celou simulaci v jednom kroku; místo toho se simulace provádí v blocích po intervalech  $t_d \in \langle \tau_i, \tau_{i+1} \rangle$ ; po vypočtení každého bloku se volá předaná `callback` funkce `cb([  $t_1$ ,  $t_2$ , .. ], [  $\varphi(t_1)$ ,  $\varphi(t_2)$ , .. ])`.

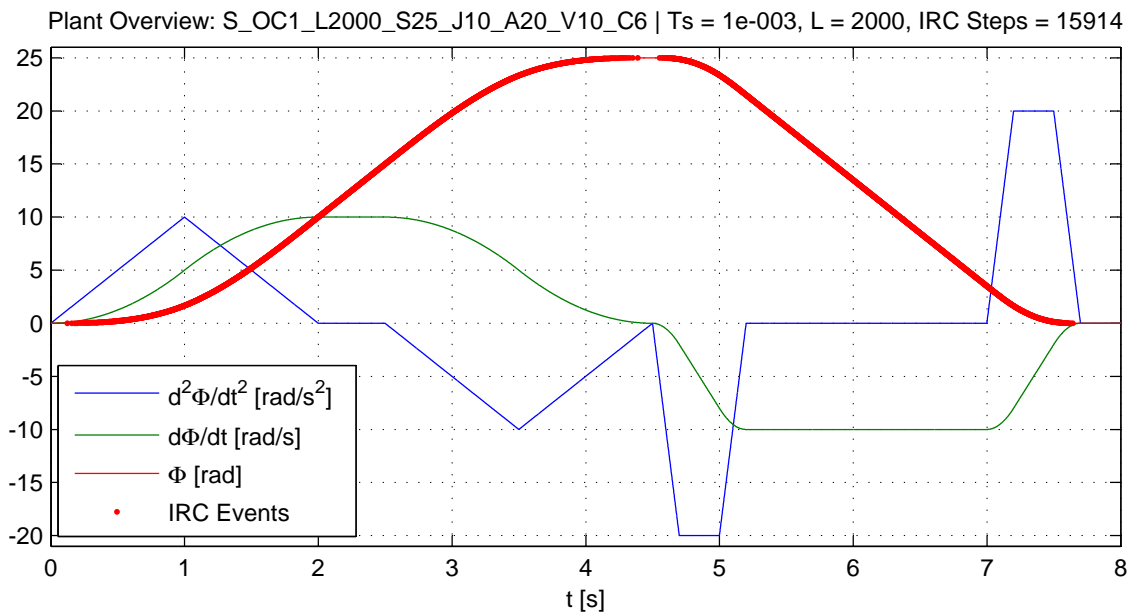
### BlkPlantOC – Model systému generujícího optimální trajektorii s omezením

Implementace modelu měřeného systému jako generátoru optimální trajektorie lineárního systému ve formě řetězce tří integrátorů, jehož stavové proměnné představují polohu, rychlost a zrychlení –  $x(t) = [s(t), v(t), a(t)]^T$ .

Hledaná optimální trajektorie změní v minimálním čase stav z klidové polohy  $s_0$  do klidové polohy  $s_f$ , přičemž vstup systému (derivace zrychlení, *jerk*) je omezen –  $|u(t)| < j_m$  – a dále jsou definována stavová omezení na zrychlení –  $|a(t)| < a_m$  – a rychlost –  $|v(t)| < v_m$ .

Problém byl podrobně analyzován např. v [12]. Bylo ukázáno, že optimální trajektorie je vždy určena vstupem  $u(t)$  po částech konstantním na 7 intervalech  $t_1$  až  $t_7$ , kdy přitom  $u(t_1) = u(t_7) = j_m$ ,  $u(t_3) = u(t_5) = -j_m$ ,  $u(t_2) = u(t_4) = u(t_6) = 0$  (pro  $s_f > s_0$ ).

Příklad časových průběhů polohy, rychlosti a zrychlení získaných tímto modelem je na obrázku 22. Příklad je složen ze dvou pohybů – z polohy  $s_0 = 0$  do  $s_f = 25$ , s omezením  $j_m = 10$ , a zpět s  $j_m = 100$ , pro oba případy je  $a_m = 20$  a  $v_m = 10$ .



Obrázek 22: Příklad profilu časového profilu zrychlení, rychlosti a polohy z BlkPlantOC.

Algoritmus výpočtu časových intervalů a výstupních hodnot byl popsán v práci [27] (Jáger, 2009), ze které zde uvedená implementace vychází.

Funkce:

- `BlkPlantOC(dt, sf, st, mj, ma, mv)` – inicializace; `dt`: perioda vzorkovacích okamžiků  $\tau_i$ ; `sf`: počáteční poloha; `st`: koncová poloha; `mj`: omezení derivace zrychlení (*jerk*); `ma`: omezení zrychlení (*acceleration*), pokud je 0, neuplatní se; `mv`: omezení rychlosti (*velocity*), pokud je 0, neuplatní se.

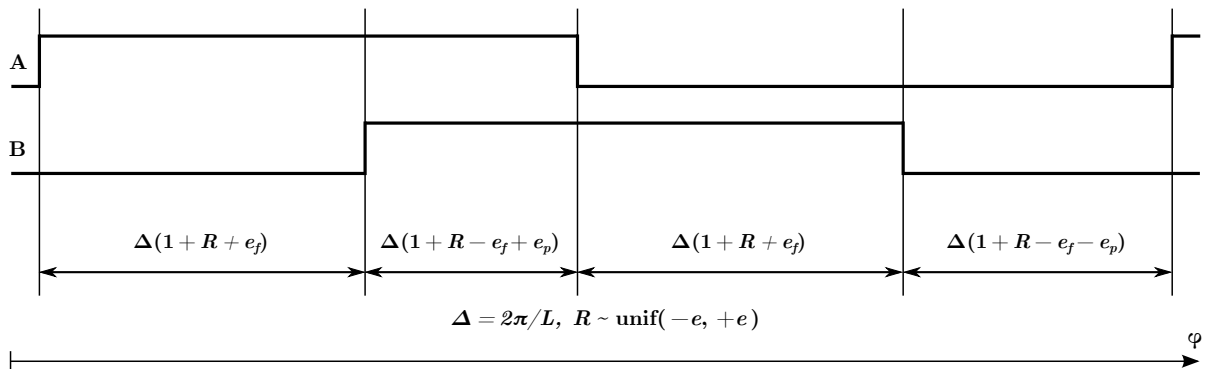
Při inicializaci se vypočtou všechny potřebné parametry optimální trajektorie.

- `getRealPVA()`, `fineSimulation(ddt, cb)` – viz popis třídy `BlkPlantTF`.

## BlkIrcModel – Model IRC snímače

Implementace modelu IRC snímače. Při inicializaci jsou vygenerovány polohy  $L$  značek na intervalu  $\langle 0, 2\pi \rangle$ . Je možné modelovat nepřesnosti snímače, jak byly popsány v kapitole 2.4. *Chyba periody* (šířky značky) je modelována rovnoměrným rozdělením pravděpodobnosti na intervalu  $\delta T \in \langle -e\Delta z, e\Delta z \rangle$ , kde  $e$  je zadaný parametr. *Chyba fáze*  $\delta\Phi = e_f\Delta z$  a *chyba délky pulzu*  $\delta P = e_p\Delta z$  jsou uvažovány konstantní.

Blok umožňuje použít i jednodušší model nepřesnosti jako šumu podle článku [3]. V tom případě je poloha každé značky posunuta od přesné hodnoty o *chybu polohy* s trojúhelníkovým rozdělením pravděpodobnosti na intervalu  $\delta z \in \langle -e_r\Delta z, e_r\Delta z \rangle$ . Tato chyba nemá kumulativní charakter, narozdíl od *chyby periody*  $\delta T$ .



Obrázek 23: Model nepřesností IRC snímače s parametry  $e, e_f, e_p$ .

Funkce:

- `BlkIrcModel(L, e, ef, ep, er)` – inicializace;  $L$ : počet pulzů na otáčku, model je uvažován včetně 4X dekodéru, tj. pro snímač s CPR = 500 bude  $L = 2000$ ;  $e, e_f, e_p, e_r$ : výše popsané parametry chybového modelu, je-li  $e_r$  nenulové, použije se chybový model s šumem a ostatní parametry jsou ignorovány.
- `cbPlantDataBlock(T, ft, fy)` – funkce je volána jako *callback* z bloku `BlkPlant*` z funkce `fineSimulation`;  $ft$ : vektor časových okamžiků  $t_d$ ;  $fy$ : vektor poloh  $\varphi(t_d)$ . V rámci volání této funkce jsou detekovány průchody polohy přes hrany značek, a ukládány odpovídající časové okamžiky  $t_k$  a směry průchodu  $d(t_k) \in \{1, -1\}$ .
- `IrcEvents = getIrcEvents()` – vrací interně uložené detekované IRC pulzy (*edge-pulses*) ve formátu `IrcEvents: [ [  $t_1$ ;  $d(t_1)$  ] [  $t_2$ ;  $d(t_2)$  ] .. ]`.

## BlkAcqrMT – Měřicí systém podle adaptivního M/T algoritmu

Implementace adaptivní měřicí metody M/T popsané v kapitole 3.

Blok zpracovává polohové pulzy – informace, že v čase  $t_i$  došlo k průchodu polohy přes IRC značku ve směru  $d(t_i)$ .

S každým příchozím polohovým pulzem je hodnota čítače  $M$  zvýšena nebo snížena o jedna. Je-li  $t_i - t_k \geq T_c$ , kde  $t_k$  je čas posledního měření, je vygenerováno nové měření  $z(t_k)$ :

- $t_k = t_i$ ,
- $d(t_i) = +1 \rightarrow z(t_k) = M\Delta z$ ,
- $d(t_i) = -1 \rightarrow z(t_k) = (M + 1)\Delta z$ .

Funkce:

- **BlkAcqrMT(L, Tc)** – inicializace; L: počet dekódovaných pulzů na otáčku; Tc: minimální čas měření, volitelný parametr metody M/T.
- **z = proc(IrcEvents)** – získá měření  $z(t_k)$  z dekódovaných IRC pulzů; **IrcEvents**: vektor polohových pulzů ve formátu popsáném u třídy **BlkIrcModel**; formát výstupu **z**: [ [  $t_1$ ;  $z(t_1)$  ] [  $t_2$ ;  $z(t_2)$  ] .. ].

## BlkFltrKalman – Blok filtru – Kalmanův filtr

Kalmanův filtr pro odhad skutečné polohy (očištěné od šumu), rychlosti a zrychlení, tj. odhad stavu  $\hat{x}(t_k)$ . Algoritmus byl podrobně odvozen a popsán v kapitole 4.

S každým příchozím měřením  $[t_k, z(t_k)]$  se provede:

- výpočet koeficientů  $\Phi_k$  (4.20),  $\Gamma_k$  (4.26),  $\Pi_k$  (4.27), k čemuž je vypočítat potřeba maticové exponenciály  $e^{A_R T_k}$  a  $e^{-A_R T_k}$ ,  $T_k = t_k - t_{k-1}$ , pomocí funkce **Expn9\_2()**.
- výpočet nového odhadu stavu  $\hat{x}(t_k)$  podle vztahu (4.14) z aktuálního měření  $z(t_k)$  a hodnot z předchozího kroku  $t_{k-1}, z(t_{k-1}), \hat{x}(t_{k-1})$ .

Při inicializaci se provede výpočet  $K$  (4.8) a  $A_R$  (4.10) podle parametru  $\alpha$ , a inverze  $A_R^{-1}$ .

Funkce:

- **BlkFltrKalman(a, Td)** – inicializace; a: parametr  $\alpha$ ; Td: parametr  $T_d$ , *dead-time*.
- **x = proc(tk, zk)** – z měření  $[t_k, z(t_k)]$  vypočte nový odhad  $\hat{x}(t_k)$ ; formát výstupu **x**: [  $\hat{\varphi}(t_k)$ ;  $\hat{\dot{\varphi}}(t_k)$ ;  $\hat{\ddot{\varphi}}(t_k)$  ].



## BlkFltrPredictor – Blok filtru – Prediktor

Prediktor pro časový posun odhadu z času měření  $t_k$  k času dalšího výstupního vzorkovacího času  $\tau_j$ . Nový výpočet predikce  $\tilde{x}(\tau_j)$  podle (4.19) a (4.18) je třeba uskutečnit při každé změně kteréhokoliv z těchto časů. Zároveň provádíme detekci *selhání predikce a zastavení* podle parametru  $T_d$ , jak bylo popsáno v kapitole 4.3.

Funkce:

- **BlkFltrPredictor(L, Td)** – inicializace; L: počet dekódovaných pulzů na otáčku; Td: parametr  $T_d$ , *dead-time*.
- **in(tk, zk, dk, xk)** – nastaví poslední známé měření a filtrační odhad stavu; tk:  $t_k$ ; zk  $z(t_k)$ ; dk:  $d(t_k)$ ; xk:  $\hat{x}(t_k)$ .
- **nextTick(tj)** – nastaví další známý budoucí vzorkovací čas  $t_j = \tau_{j+1}$ .
- **[x, f] = out()** – přečte predikční odhad stavu  $\tilde{x}(\tau_j)$ , kde  $\tau_j$  je hodnota tj, se kterou byla naposledy volána funkce **nextTick()**; formát výstupu  
x: [  $\tau_j$ ;  $\tilde{\varphi}(\tau_j)$ ;  $\tilde{\dot{\varphi}}(\tau_j)$ ;  $\tilde{\ddot{\varphi}}(\tau_j)$  ]; f: 0 = OK, 1 = dead-time, 2 = selhání predikce.

## 5.2 Pomocné funkce

**[Epos, Eneg] = expm92(A)** – Výpočet kladné a záporné maticové exponenciály

Funkce vypočte maticovou exponencielu pro kladný a záporný argument  $A$  algoritmem popsaným v kapitole 4.4.  $Epos = e^A$ ,  $Eneg = e^{-A}$ .

Výpočet pro kladný a záporný argument zároveň je rychlejší než opakované volání funkce, protože významná část operací se pro oba případy shoduje. Společná část výpočtu je vyčleněna do samostatné funkce **expm90()**.

**[s, U, V] = expm90(A)** – Výpočet maticové exponenciály, společná část

Funkce vypočte konstanty  $b_0 \dots b_9$ , určí *scaling factor*  $2^s$  a sestaví výrazy  $U$  a  $V$  (viz podrobný popis algoritmu).

**graph6(name, ID, SE, X, Tc, a)** – Vykreslení grafů s výsledky měření

Funkce vykreslí 6 grafů – časové průběhy polohy, rychlosti a zrychlení – hodnoty „skutečné“ a měřené, a časové průběhy jejich odchylek. Viz popis **xs10**, **xs11** a **xs12** dále.

### 5.3 Funkce pro generování simulačních dat

Dále popsané funkce `xm_tf1` a `xm_oc1` slouží ke spuštění parametrizované *simulace modelu systému a modelu IRC snímače*. Výstup je zapouzdřen do datové struktury `SimEnv`:

- **x**: „skutečné“ hodnoty polohy, rychlosti a zrychlení z nějakého modelu systému,
- **t**: vzorkovací okamžiky *skutečných hodnot*, ekvidistantní s s periodou 1 ms,
- **e**: pole časových okamžiků a směru pulzů ze simulovaného IRC snímače,
- **L**: počet dekodovaných pulzů ze snímače na jednu otáčku (tj. změnu polohy o  $2\pi$ ),
- **name**: automaticky sestavený název pro identifikaci zdrojových dat např. v grafech, zahrnuje hodnoty všech volitelných parametrů.

Pokud je funkce volána bez výstupního parametru, výsledek se uloží do *Wokrspace* pod automaticky sestaveným názvem `name`. Takto si můžeme snadno připravit simulační data v mnoha variantách pro pozdější testování měřicích algoritmů.

Generování dat jen odděleno od funkcí pro jejich zpracování, protože v některých případech trvá velmi dlouho – desítky sekund až jednotky hodin podle požadované přesnosti. Příčinou je potřeba velmi jemného vzorkování výstupu modelu systému pro simulaci IRC snímače.

#### # `xm_tf1(L, A, CLK, e, ef, ep)` – Systém s přenosovou funkcí

Funkce simuluje měřený systém `BlkPlantTF` s přenosou funkcí:

$$F(s) = \frac{6s + 100}{s^2 + 6s + 100}$$

Vstup systému, který odpovídá zrychlení v ustáleném stavu, je určen takto:

$$u(t) = \{t \in \langle 0, 2 \text{ s} \rangle : -A; t \in \langle 2 \text{ s}, 4 \text{ s} \rangle : +A; t \in \langle 4 \text{ s}, 6 \text{ s} \rangle : 0\}$$

Parametry funkce:

- **L** – počet pulzů IRC snímače na otáčku (po  $4X$  dekodování).
- **A** – amplituda budicího signálu.
- **CLK** – kmitočet vzorkování pro simulaci funkce snímače; určuje přesnost měření časových okamžiků IRC pulzů.
- **e, ef, ep** – volitelné parametry, určují nepřesnost snímače, viz popis bloku `BlkIrcModel`.

## # xm\_oc1(L, S, J, A, V, CLK, e, ef, ep) – Systém s optimální trajektorií

Funkce simuluje měřený systém BlkPlantOC ve třech krocích:

- pohyb z polohy 0 do S s omezením derivace zrychlení J, zrychlení A, rychlosti V;
- pohyb z polohy S do 0 s omezením derivace zrychlení 5J, zrychlení 2A, rychlosti 1.5V;
- konstantní poloha po dobu 500 ms.

Celkový čas závisí na volbě cílové polohy S a omezujících parametrů.

Parametry funkce:

- S, J, A, V – parametry pro generátor optimální trajektorie.
- L, CLK, e, ef, ep – viz popis funkce xm\_tf1.

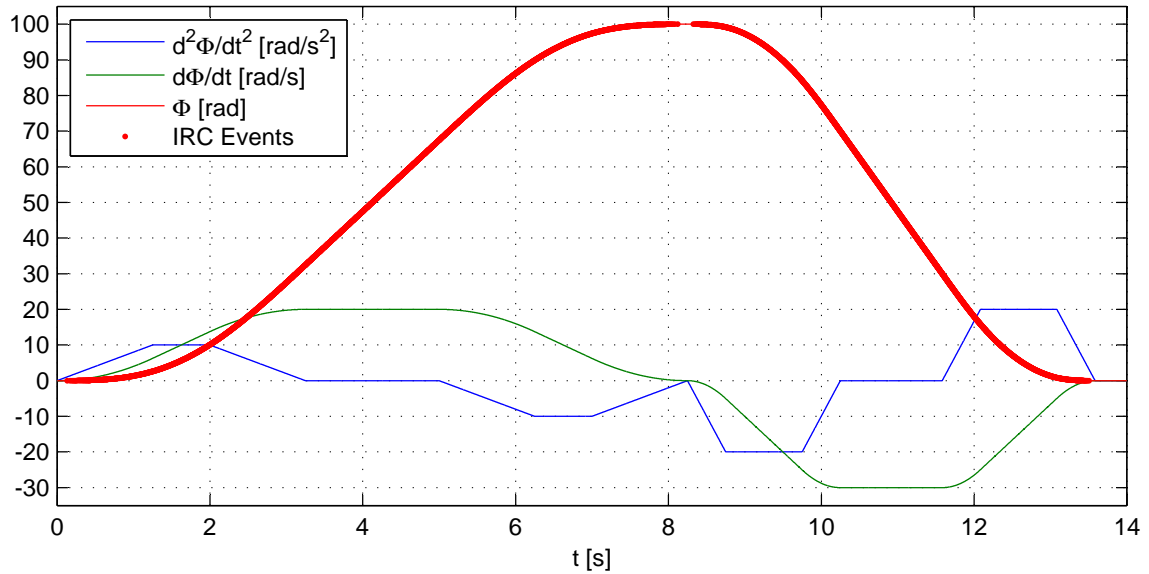
## Příklad použití

První příklad ukazuje použití modelu s optimální trajektorií. Parametry byly zvoleny tak, aby trajektorie obsahovala úseky s konstantním zrychlením a s konstantní rychlostí:

```
>> xm_oc1( 2000, 100, 8, 10, 20, 1e6, 0.008, 0.01, 0.02 )
  [BlkPlantOC] 0.001 Init mj=8.000 ma=10.000 mv=20.000
  [BlkPlantOC] 0.002 Init mj=40.000 ma=20.000 mv=30.000
  [BlkPlantOC] 0.012 Calculating P,V,A real response
  [BlkPlantOC] 1.036 Finished
  [BlkPlantOC] 1.043 Calculating P,V,A real response
  [BlkPlantOC] 1.725 Finished
  [BlkIrcModel] 1.728 Init, L=2000
  [BlkIrcModel] 1.728 ERROR MODEL e=0.008, ef=0.010, ep=0.020
  [BlkPlantOC] 1.734 RUN fineSimulation
  [BlkPlantOC] 2.135 8249 blocks, 8249000 steps total
  [BlkIrcModel] 71.854 Edge-pulse 2000
...
  [BlkIrcModel] 493.972 Edge-pulse 62000
  [BlkPlantOC] 530.071 END fineSimulation
  [BlkIrcModel] 530.095 getData, pulse count: 63661
Workspace Var: SE_OC1_L2000_S100_J8_A10_V20_C6_e8_ef10_ep20
```

Časový průběh polohy, rychlosti a zrychlení tohoto modelu systému ukazuje obrázek 24. Parametry chybového modelu ( $e = 0.008$ ;  $e_f = 0.01$ ;  $e_p = 0.02$ ) byly zvoleny podle katalogových údajů snímače HEDS-5540 [16].

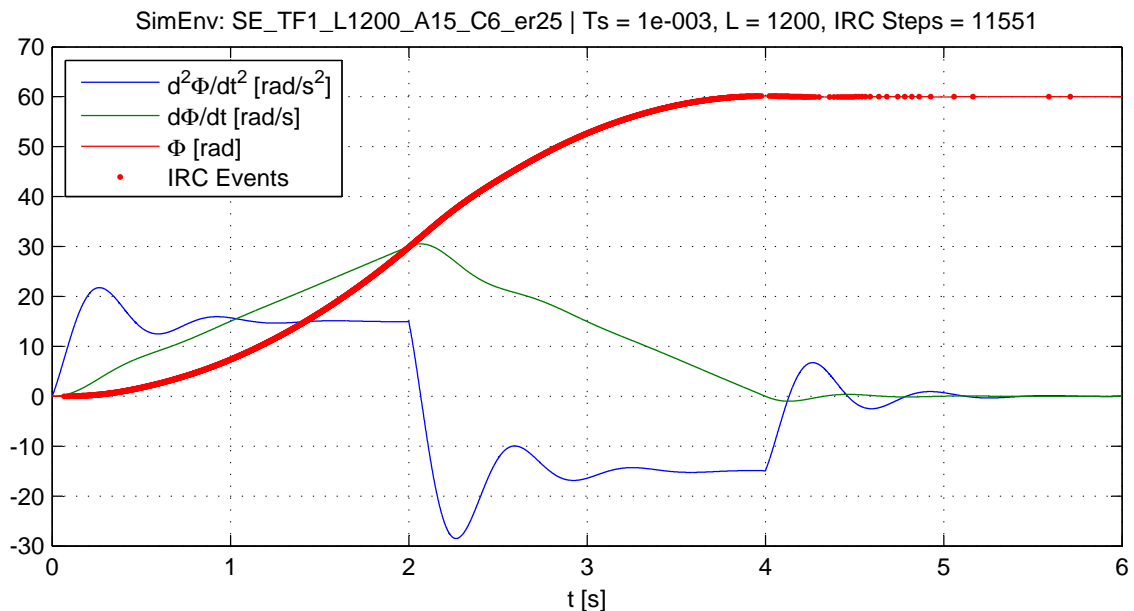
SimEnv: SE\_OC1\_L2000\_S100\_J8\_A10\_V20\_C6\_e8\_ef10\_ep20 | Ts = 1e-003, L = 2000, IRC Steps = 63660



Obrázek 24: Příklad časových průběhů „skutečných“ hodnot veličin vygenerovaných funkcí `xm_oc1()`.

Druhý příklad ukazuje použití modelu s přenosovou funkcí. Parametry byly zvoleny pro srovnání shodně s článkem [3]. Průběhy ukazuje obrázek 25.

```
>> xm_tf1( 1200, 15, 1e6, 0, 0, 0, 0.025 )  
...  
Workspace Var: SE_TF1_L1200_A15_C6_er25
```



Obrázek 25: Příklad časových průběhů „skutečných“ hodnot veličin vygenerovaných funkcí `xm_tf1()`.

## 5.4 Funkce pro testování měřicích a filtračních algoritmů

```
# xs10_MT_Fltr_Graph(SE, Tc, a, Td)
```

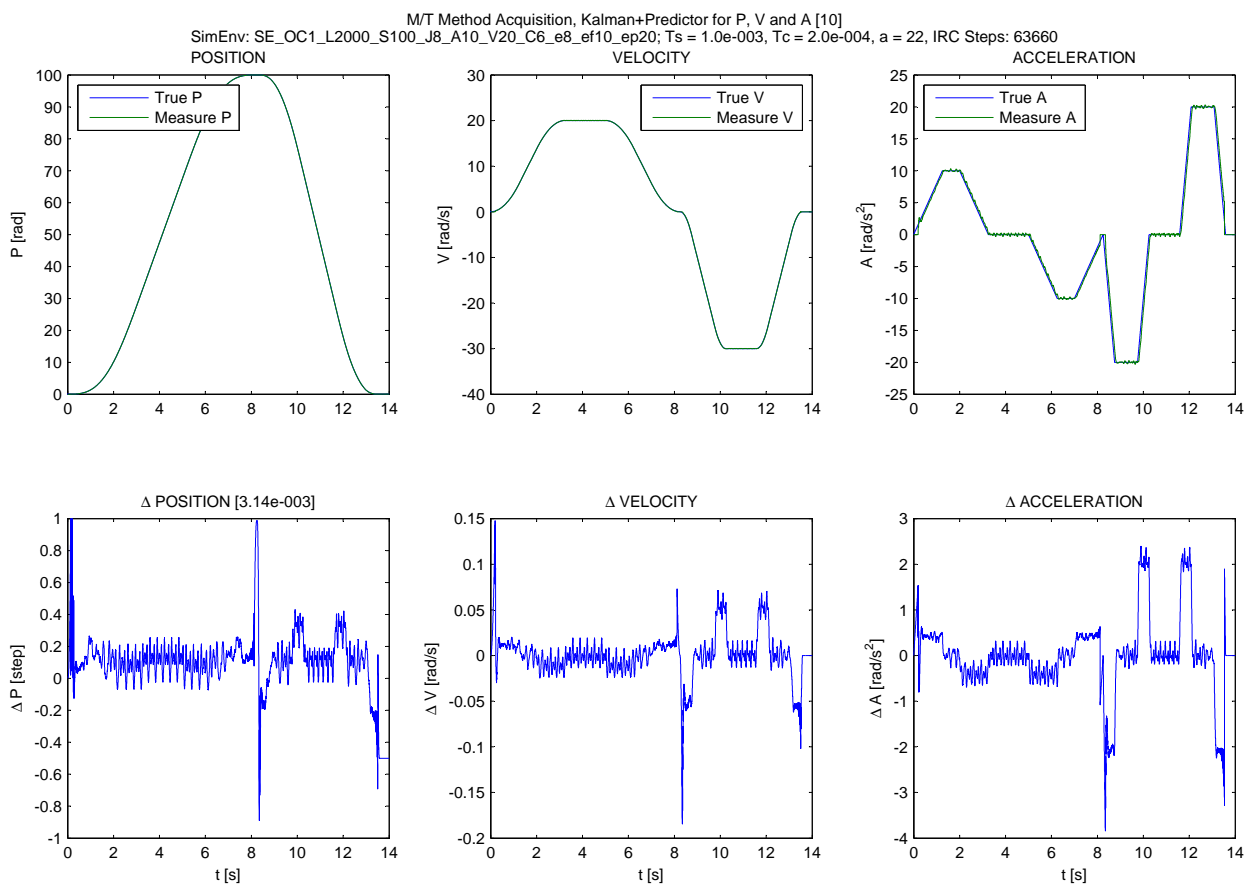
Funkce provede simulaci:

- měření IRC pulzů metodou M/T – dostaneme  $z(t_k)$ ,
- výpočet  $\varphi(\tau_j)$ ,  $\dot{\varphi}(\tau_j)$ ,  $\ddot{\varphi}(\tau_j)$  pomocí Kalmanova filtru a prediktoru.

Parametry: SE: Simulační data *SimEnv*; Tc: parametr *BlkAcqrMT*; a, Td: parametry pro *BlkFltrKalman* a *BlkFltrPredicror*.

Výstupem je graf, který porovnává časový průběh „skutečných“ a měřených hodnot a průběh jejich rozdílu. Obrázek 26 ukazuje výstup funkce spuštěné s parametry:

```
>> ex_xs10_MT_Fltr_Graph( ...
    SE_OC1_L2000_S100_J8_A10_V20_C6_e8_ef10_ep20, 2e-4, 22, 0.03 )
```



Obrázek 26: Příklad výstupu funkce `ex_xs10_MT_Fltr_Graph()`.

```
# xs11_MT_FiniteDiff_Graph(SE, Tc)
```

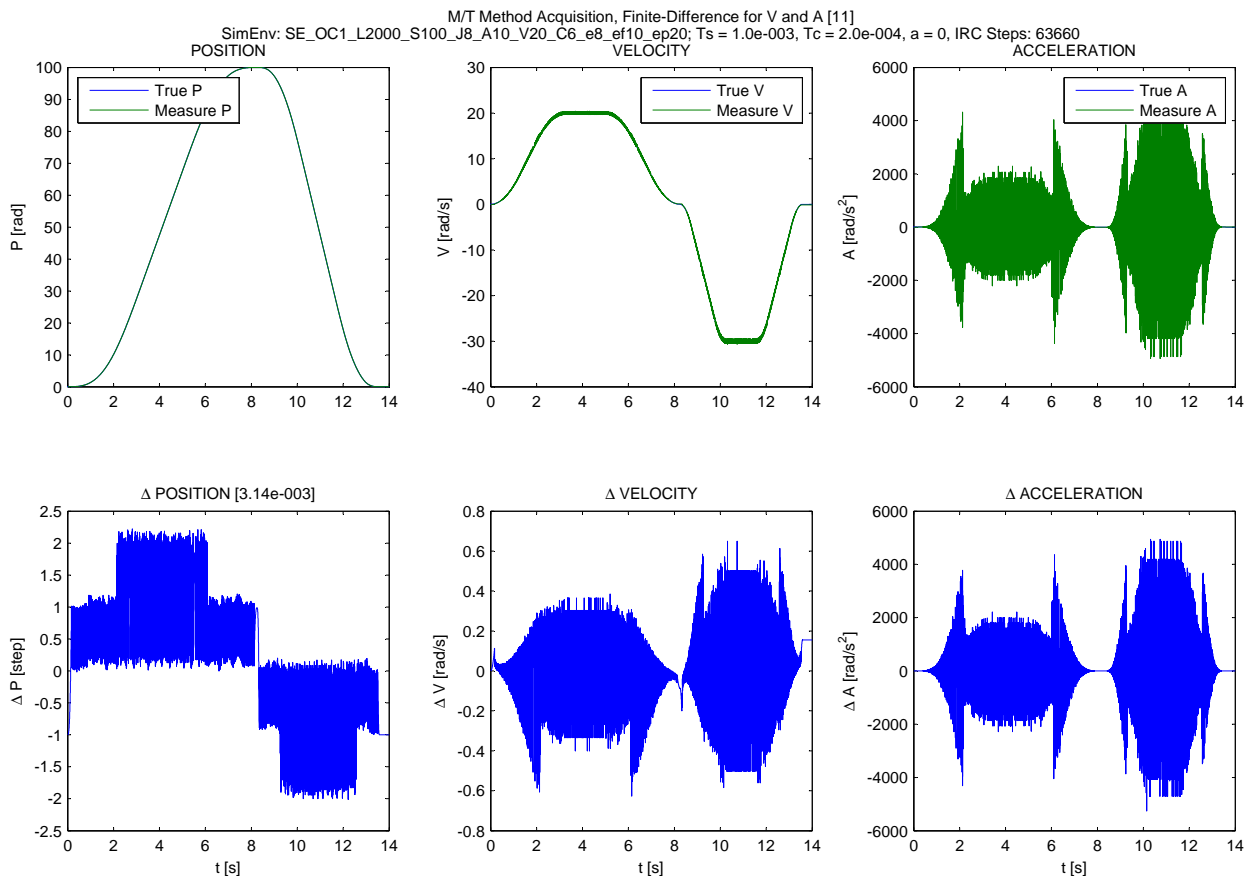
Funkce provede simulaci:

- měření IRC pulzů metodou M/T – dostaneme  $z(t_k)$ ,
- výpočet  $\varphi(t_k), \dot{\varphi}(t_k), \ddot{\varphi}(t_k)$  metodou numerických diferencí,
- měřené hodnoty v časech  $\tau_j$  pro srovnání se „skutečnými“ hodnotami jsou simulovány hodnotami z posledního času  $t_k$ , pro který  $t_k < \tau_j$ .

Parametry: SE: Simulační data *SimEnv*; Tc: parametr *BlkAcqrMT*.

Výstupem je graf v podobě stejné jako u předchozí funkce *xs10*. Obrázek 27 ukazuje výstup funkce spuštěné s parametry:

```
>> ex_xs11_MT_FiniteDiff_Graph( ...
    SE_OC1_L2000_S100_J8_A10_V20_C6_e8_ef10_ep20, 2e-4 )
```



Obrázek 27: Příklad výstupu funkce `ex_xs11_MT_FiniteDiff_Graph()`.

```
# xs12_M_FiniteDiff_Graph(SE, q)
```

Funkce provede simulaci:

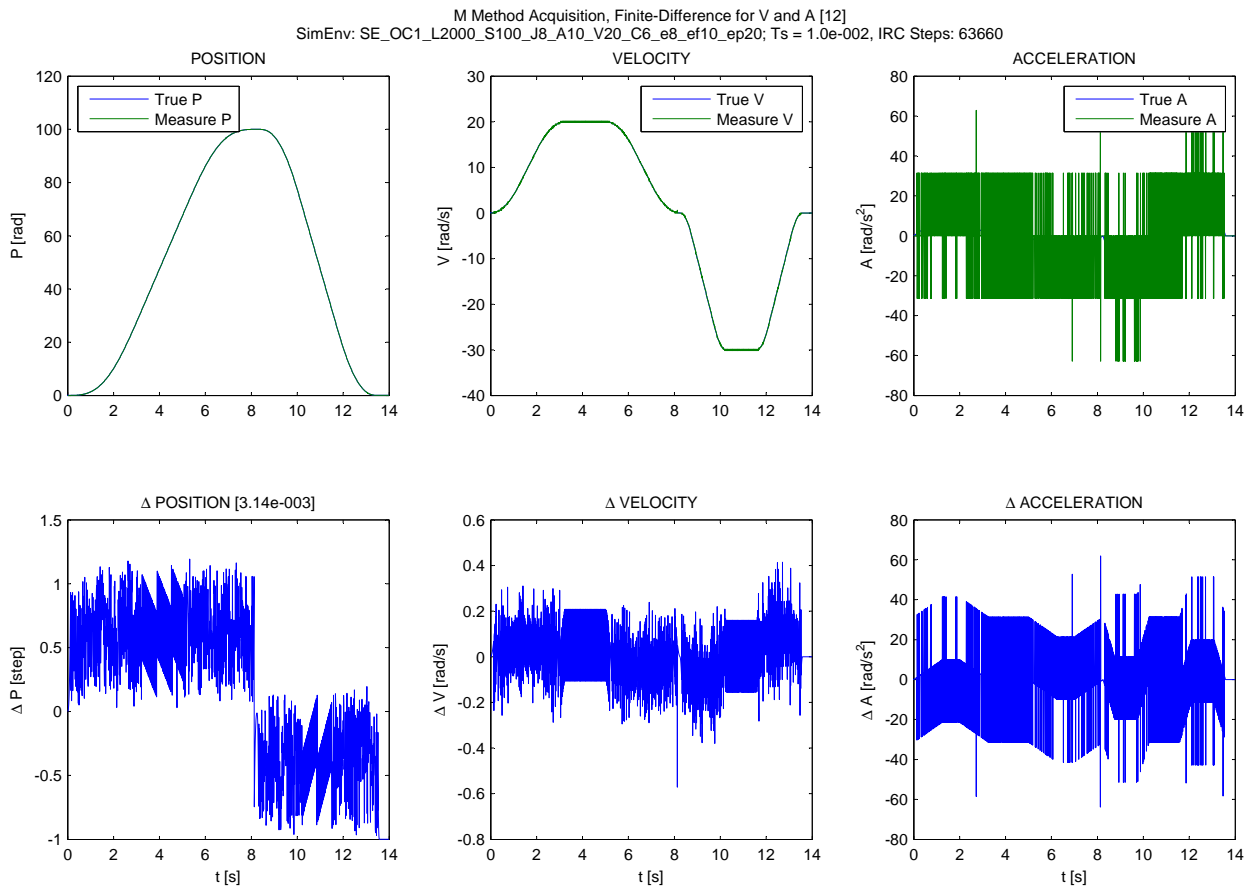
- měření IRC pulzů metodou M – dostaneme  $z(\tau'_j)$ ,
- výpočet  $\varphi(\tau'_j), \dot{\varphi}(\tau'_j), \ddot{\varphi}(\tau'_j)$  metodou numerických diferencí.

Parametry: SE: Simulační data *SimEnv*; q: perioda  $\tau'_j$  je  $q$ -krát delší než perioda  $\tau_j$ .

Pomocí parametru  $q$  lze simulovat delší vzorkovací periodu, než s jakou byla vygenerována simulační data (tj. 1 ms). U metody M se totiž chyba výpočtené rychlosti a zrychlení snižuje s delším vzorkováním. U zde uvedeného příkladu je  $q = 10$ , tedy vzorkování 10 ms.

Výstupem je graf v podobě stejné jako u předchozí funkce *xs10*. Obrázek 28 ukazuje výstup funkce spuštěné s parametry:

```
>> xs12_M_FiniteDiff_Graph( SE_OC1_L2000_S100_J8_A10_V20_C6_e8_ef10_ep20, 10 )
```



Obrázek 28: Příklad výstupu funkce *xs12\_M\_FiniteDiff\_Graph()*.

```
# z = xs30_MT_Dump(SE, Tc)
```

Funkce provede simulaci měření IRC pulzů metodou M/T a výsledek vrátí jako matici z:

```
z = xs30_MT_Dump(SE_TF1_L2000_A10_C6_e8_ef1_ep2, 2e-4)
      [BlkAcqrMT]  0.001 Init, L=2000, Tc=2.0e-004
      [BlkAcqrMT]  1.477 getData, pulse count: 12834, data count: 10356
z =
      6519400          1          1
      8168400          2          1
      9312400          3          1
      ...
      163067400       4200          1
      163106300       4202          1
      ...
      525288899       12733          1
      539190200       12732         -1
```

První sloupec je čas pulzu s jednotkou 10 ns, druhý sloupec je stav čítače pulzů, třetí sloupec je směr posledního pulzu zaznamenaného v daném čase.

```
# xs91_SimEnv_Model_Graph(SE)
```

Funkce vykreslí graf se „skutečnými“ hodnotami polohy, rychlosti a zrychlení v závislosti na čase. V grafu jsou dále vyznačeny časové okamžiky IRC pulzů.

Výstup této funkce byl použit pro vytvoření grafů na výše použitých obrázcích 24 a 25.

```
# xs92_SimEnv_Irc_Stat(SE)
```

Funkce slouží pro ladění výpočetních algoritmů. Zobrazí 15 nejdelších intervalů mezi dvěma po sobě jdoucími pulzy. Seznam je seřazen podle času pulzu. Příklad:

```
>> xs92_SimEnv_Irc_Stat(SE_OC1_L2000_S100_J8_A10_V20_C6)
      [BlkAcqrT]  0.001 Init, L=2000
      [BlkAcqrT]  0.463 getData, pulse count: 63660, data count:
Top 15 time diffs (MAX 0.209):
      t=0.168 dt=0.035
      ...
      t=8.327 dt=0.209
      ...
      t=13.484 dt=0.014
      t=13.505 dt=0.020
```



## 6 Vyhodnocení vlastností navržených algoritmů

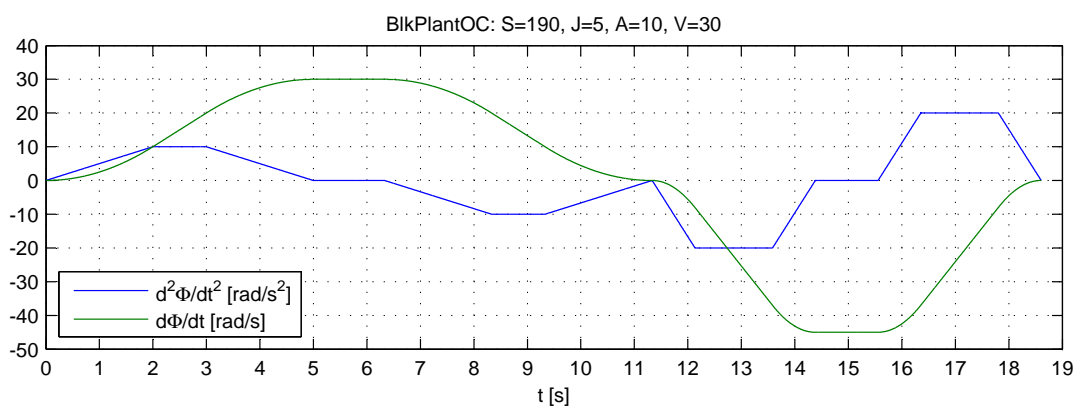
V předchozí kapitole byly na konkrétním příkladu rychlostního profilu a modelu IRC snímače ilustrovány výsledky několika různých algoritmů odhadu polohy, rychlosti a zrychlení:

- (a) měření M/T (adaptivní), odhad pomocí Kalmanova filtru a prediktoru,
- (b) měření M/T (adaptivní), odhad derivací metodou numerických diferencí,
- (c) měření M (počítání pulzů za  $T_s$ ), odhad derivací metodou numerických diferencí.

Časové průběhy odhadů a jejich srovnání se „skutečnými“ hodnotami ukazují obrázky 26, 27 a 28. Je evidentní, že v této práci navržená metoda (a) využívající Kalmanův filtr poskytuje nejkvalitnější odhady polohy, rychlosti a zrychlení. A to i v situaci, kdy byla uvažována výrazná nedokonalost IRC snímače.

V praxi nejpoužívanější metoda (c) na testovaném příkladu poskytuje relativně použitelné měření polohy a rychlosti, ovšem s výrazně větším šumem než metoda (a), a to i při ustálené rychlosti. Navíc musela být proti metodě (a) desetinásobně zvýšena vzorkovací perioda na 10 ms, jinak by byly výsledky výrazně horší. Měření zrychlení je nepoužitelné. Případné další fitrování měření kontaminovaného takovýmto šumem, například dolnoproputním filtrem, by bylo možné jen za cenu neakceptovatelného zhoršení kvality odhadu.

Dále podrobně analyzujeme chování navržené metody (a) v různých situacích, a jeho závislost na volbě parametrů. Rychlostní profil generovaný jako optimální trajektorie s vhodně zvolenými omezeními  $j_m \geq |\ddot{\varphi}|$  (jerk),  $a_m \geq |\dot{\varphi}|$  (zrychlení) a  $v_m \geq |\dot{\varphi}|$  (rychlost) nám umožní otestovat měřicí systém za podmínky: (i) *konstatní rychlosti*; (ii) *konstantního zrychlení*; (iii) *konstantní derivace zrychlení*; a také při přechodu mezi těmito stavy.



Obrázek 29: Profil rychlosti a zrychlení s omezeními  $j_m = [5; 25]$ ,  $a_m = [10; 20]$ ,  $v_m = [30; 45]$  a  $s_f = 190$ .

## 6.1 Simulační výsledky

### Ideální snímač, volba parametru $\alpha$

Pro začátek otestujeme vlastnosti navržené metody pro ideální snímač bez nepřesností. Cílem bude zjistit, jaký charakter má chyba odhadů při malých a velkých rychlostech a zrychleních, a jak závisí na parametru Kalmanova filtru  $\alpha$ .

Pro srovnání byly vygenerovány 3 sady simulačních dat (*SimEnv*), které mají shodný časový průběh a liší se jen měřítkem hodnot polohy, rychlosti a zrychlení. Pro další popis zavedeme značení:

$$(A) \text{ SE\_OC1\_L2000\_S38\_J1\_A2\_V6\_C6} - j_m = 1, a_m = 2, v_m = 6, s_f = 38$$

$$(B) \text{ SE\_OC1\_L2000\_S190\_J5\_A10\_V30\_C6} - j_m = 5, a_m = 10, v_m = 30, s_f = 190$$

$$(C) \text{ SE\_OC1\_L2000\_S950\_J25\_A50\_V150\_C6} - j_m = 25, a_m = 50, v_m = 150, s_f = 950$$

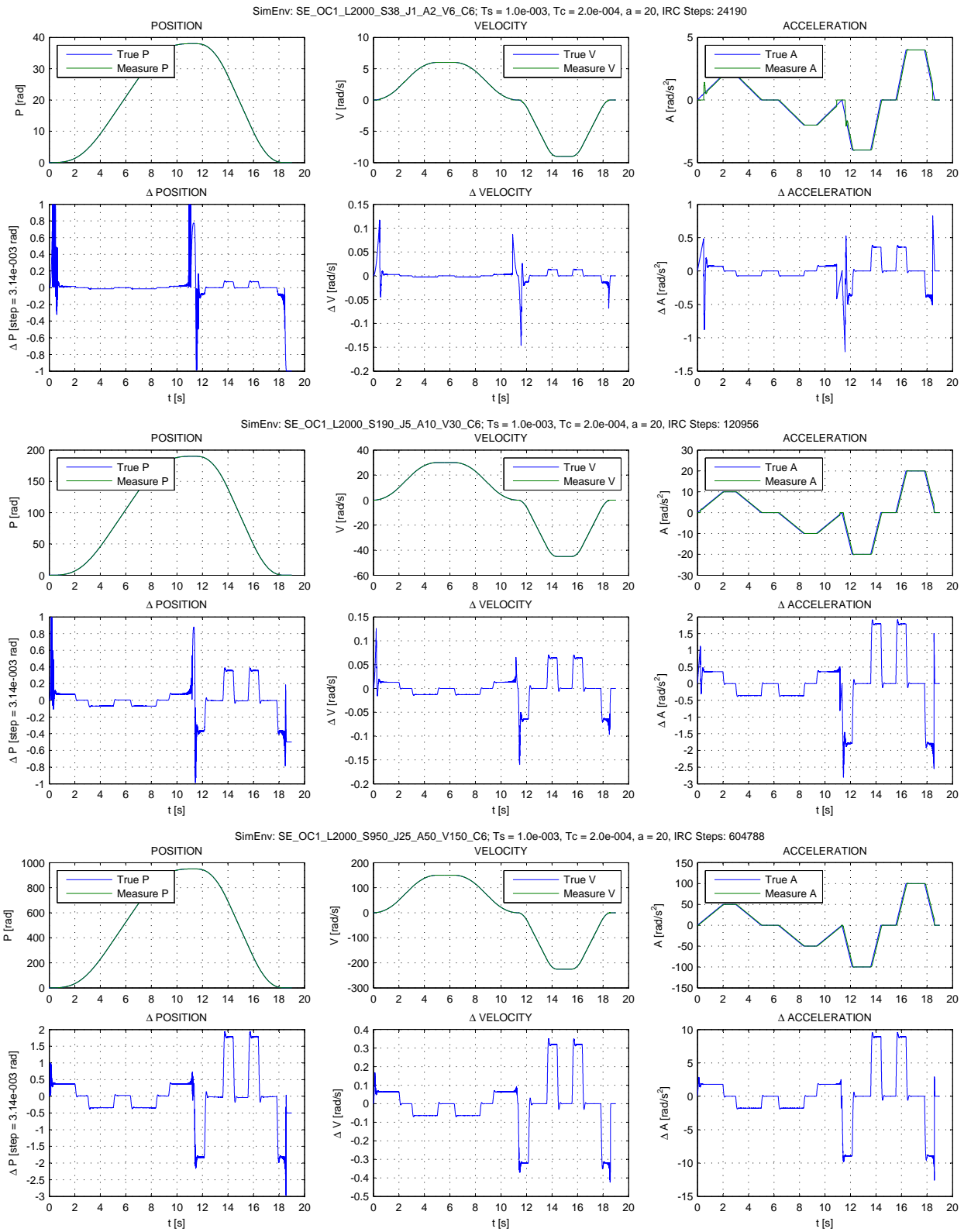
Trajektorie (viz obrázek 29) je generována výše popsanou funkcí `xm_oc1`. Tedy optimální pohyb z polohy 0 do  $s_f$  s omezeními  $j(t) \leq j_m, v(t) \leq v_m, a(t) \leq a_m$ ; a následně rychlejší pohyb zpět z  $s_f$  do 0 s omezeními  $j(t) \leq 5 \cdot j_m, v(t) \leq 2 \cdot v_m, a(t) \leq 1.5 \cdot a_m$ .

časový úsek	ustálený stav	(A)	(B)	(C)	
$t_{11} : 0.00 \dots 2.00$ $t_{13} : 3.00 \dots 5.00$ $t_{15} : 6.33 \dots 8.33$ $t_{17} : 9.33 \dots 11.33$	$\ddot{\varphi} = j = \pm j_m$	1	5	25	rad/s <sup>3</sup>
$t_{12} : 2.00 \dots 3.00$ $t_{16} : 8.33 \dots 9.33$	$\ddot{\varphi} = a = \pm a_m$	2	10	50	rad/s <sup>2</sup>
$t_{14} : 5.00 \dots 6.33$	$\dot{\varphi} = v = +v_m$	6	30	150	rad/s
$t_{21} : 11.33 \dots 12.13$ $t_{23} : 13.58 \dots 14.38$ $t_{25} : 15.56 \dots 16.36$ $t_{27} : 17.81 \dots 18.61$	$\ddot{\varphi} = j = \pm 5.0 \cdot j_m$	5	25	125	rad/s <sup>3</sup>
$t_{22} : 12.13 \dots 13.58$ $t_{26} : 16.36 \dots 17.81$	$\ddot{\varphi} = a = \pm 2.0 \cdot a_m$	4	20	100	rad/s <sup>2</sup>
$t_{24} : 14.38 \dots 15.56$	$\dot{\varphi} = v = -1.5 \cdot v_m$	9	45	225	rad/s

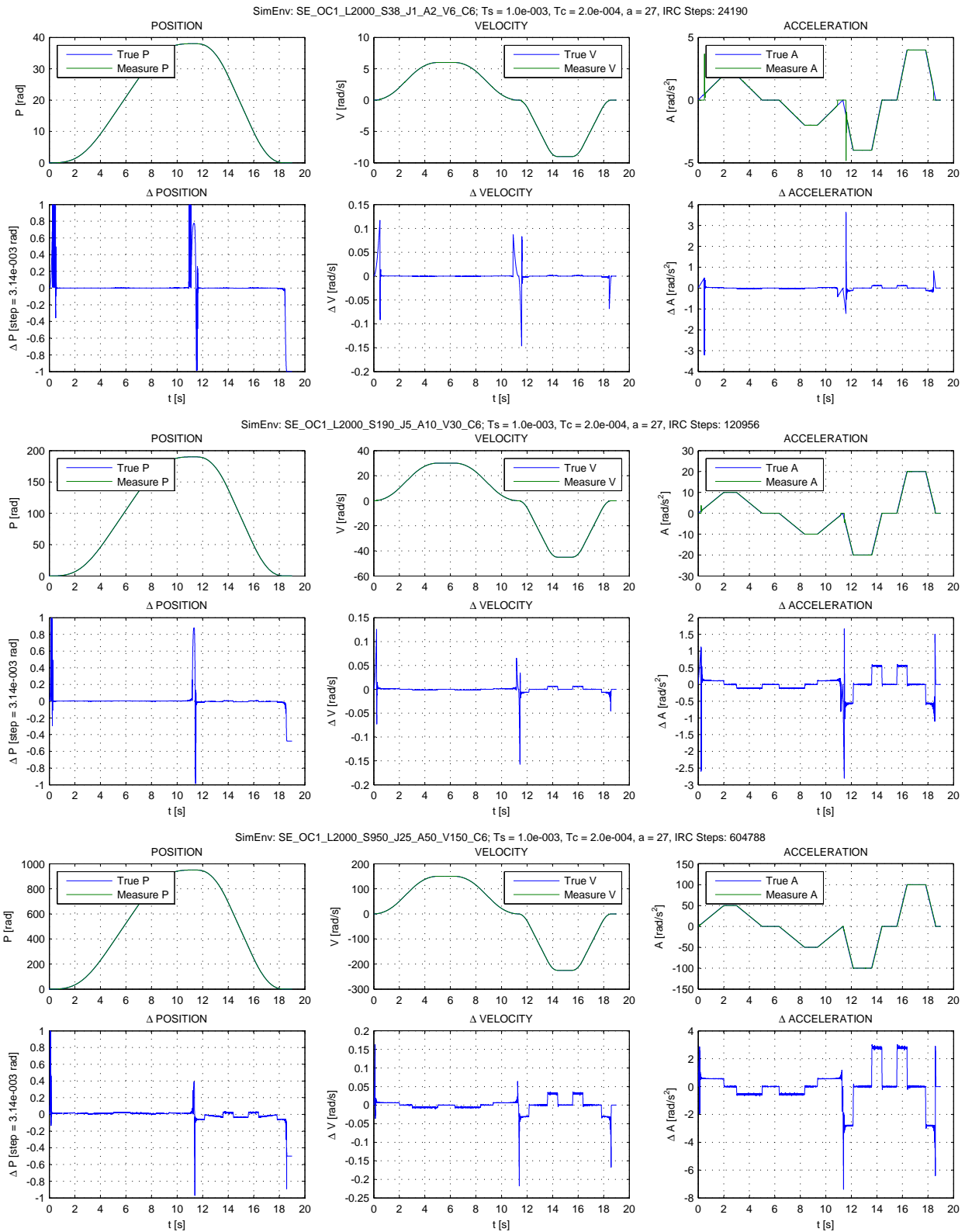
Tabulka 3: Použité rychlostní profily – intervaly ustálených hodnot rychlosti, zrychlení, jerku.

Výsledky simulace pro tato data jsou vyneseny v grafech pro dvě různé hodnoty parametru  $\alpha$ .  $T_c = 0.2$  ms,  $T_d = 30$  ms,  $L = 2000$ . Obrázek 30 pro  $\alpha = 20$ , obrázek 31 pro  $\alpha = 27$ .

Pro simulaci byla záměrně zvolena poměrně nízká hodnota kmitočtu  $f_{CLK} = 1$  MHz, který určuje přesnost měření časových okamžiků v rámci M/T metody. Pro praktické použití uvažujeme hodnoty o jeden až dva řády vyšší.



Obrázek 30: Simulační výsledky pro ideální snímač,  $\alpha = 20$ . Srovnání dat (A) / (B) / (C).



Obrázek 31: Simulační výsledky pro ideální snímač,  $\alpha = 27$ . Srovnání dat (A) / (B) / (C).

*Už po zbežném pohledu na grafy je možné formulovat několik důležitých poznatků:*

- *Chyba odhadů polohy, rychlosti i zrychlení při nějaké volbě  $\alpha$  závisí téměř výhradně na hodnotě derivace zrychlení (jerk). Větší jerk znamená větší chybu odhadů.*
- *Na časovém úseku, kde je konstantní nenulový jerk, je chyba všech odhadů konstantní.*
- *Při libovolně velké konstantní rychlosti nebo zrychlení je střední hodnota chyby všech odhadů nulová, vyskytuje se jen nevýrazný šum.*
- *Velmi problematické jsou malé rychlosti okolo rozběhu nebo zastavení.*
- *Vyšší hodnota  $\alpha$  výrazně redukuje chyby odhadů.*

Vlastnosti šumu odhadů okolo nulové hodnoty při konstantní rychlosti nebo zrychlení podrobněji analyzujeme později, ve vztahu k nepřesnostem IRC snímače a volbě  $f_{CLK}$ .

Parametr  $\alpha$  je možné interpretovat jako „rychlost“ filtru – čím větší, tím rychleji filtr reaguje na změny zrychlení, má menší fázové zpoždění.

Ze simulace pro „nejpomalejší“ trajektorii (A) je ale patrné, že příliš agresivní nastavení  $\alpha = 27$  pro pomalý pohyb způsobuje v okolí rozběhu a zastavení obrovský překmit proti skutečné hodnotě zejména u odhadu zrychlení. Při nastavení delšího času  $T_d = 0.1$  je dokonce několik prvních odhadů zrychlení záporných, přestože skutečné zrychlení je kladné! Jde ovšem o situaci, kdy je mezi měřeními (jednotlivými pulzy ze snímače) prodleva v řádu desítek ms, přičemž výstupní vzorkování odhadů je 1 ms.

Byla realizována řada pokusů o eliminaci tohoto jevu, například snižováním  $\alpha$  pro dlouhé časy měření, ale nepodařilo se nalézt řešení fungující uspokojivě ve všech případech.

## Rozbor vlivu nepřesností snímače

Pro zkoumání vlivu nepřesností snímače na kvalitu odhadů byly vygenerovány 3 sady dat s rychlostním profilem shodným se sadou (B) ( $j_m = 5, a_m = 10, v_m = 30, s_f = 190$ ):

(D) SE\_OC1\_L2000\_S190\_J5\_A10\_V30\_C6\_e0\_ef30\_ep50

$e = 0, e_f = 0.03, e_p = 0.05$  – konstantní chyba fáze 3 % a chyba délky pulzu 5 %; charakter statického vysokofrekvenčního šumu.

(E) SE\_OC1\_L2000\_S190\_J5\_A10\_V30\_C6\_e8\_ef30\_ep50

$e = 0.008, e_f = 0.03, e_p = 0.05$  – stejně jako (D) a navíc chyba periody s rovnoměrným náhodným rozdělením  $\pm 0,8$  %; kumulativní charakter chyby v rámci jedné otáčky.

(F) SE\_OC1\_L2000\_S190\_J5\_A10\_V30\_C6\_er250

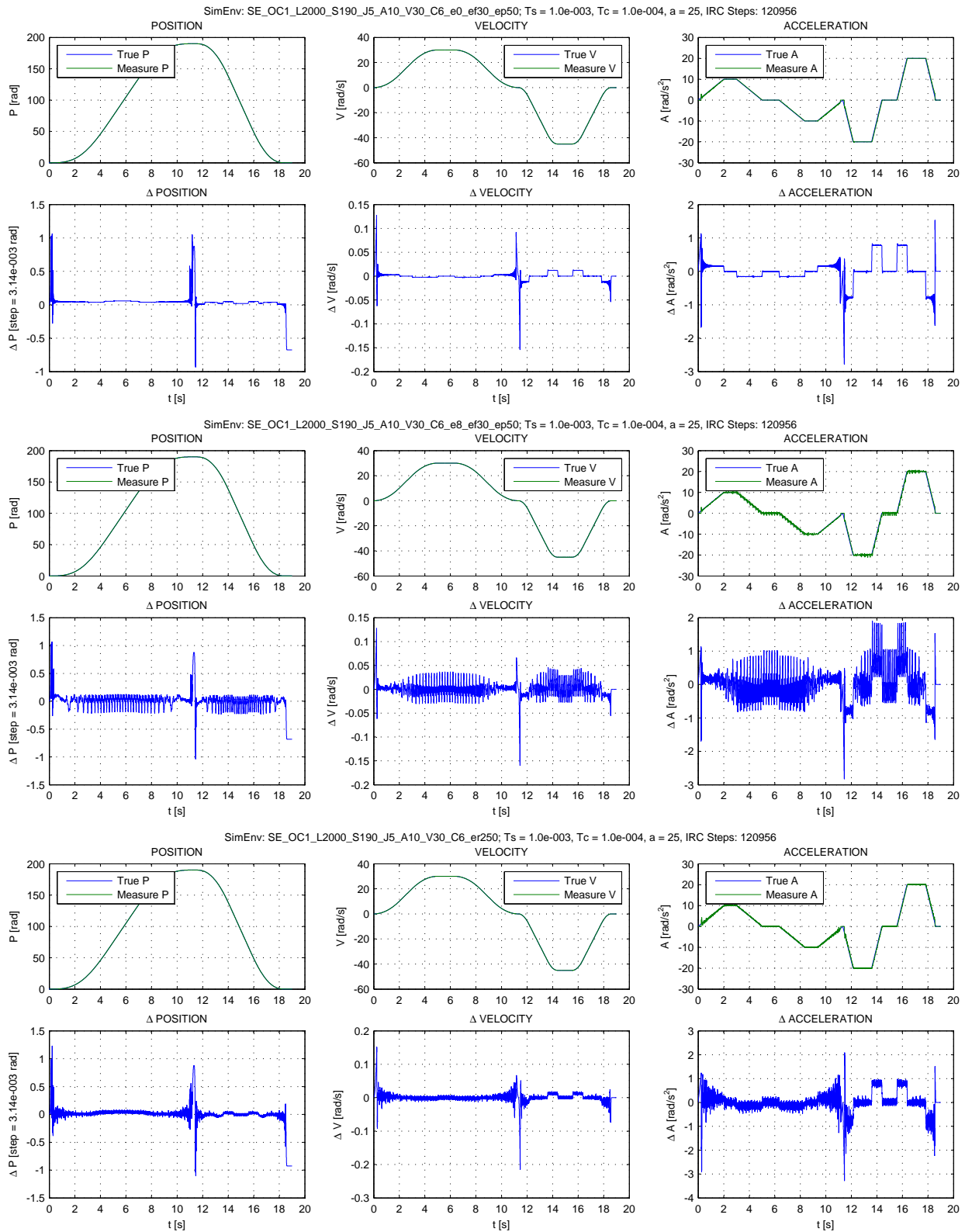
$e_r = 0.25$  – jednodušší model s náhodným posunem polohy každé značky od své nominální polohy podle trojúhelníkového rozdělení na intervalu  $\pm 25$  %; charakter velmi silného náhodného vysokofrekvenčního šumu.

Důsledek takto modelovaných nepřesností na chyby odhadů při  $\alpha$  ukazuje obrázek 32. Dopad modelu (D) je minimální. V případě (E) a (F) jsou odhady pořád použitelné, ale zatížené výrazným šumem. Snížením parametru filtru na  $\alpha = 20$  je možné tento šum velmi dobře redukovat, a to za cenu o něco větší odchylky při nekonstantním zrychlení.

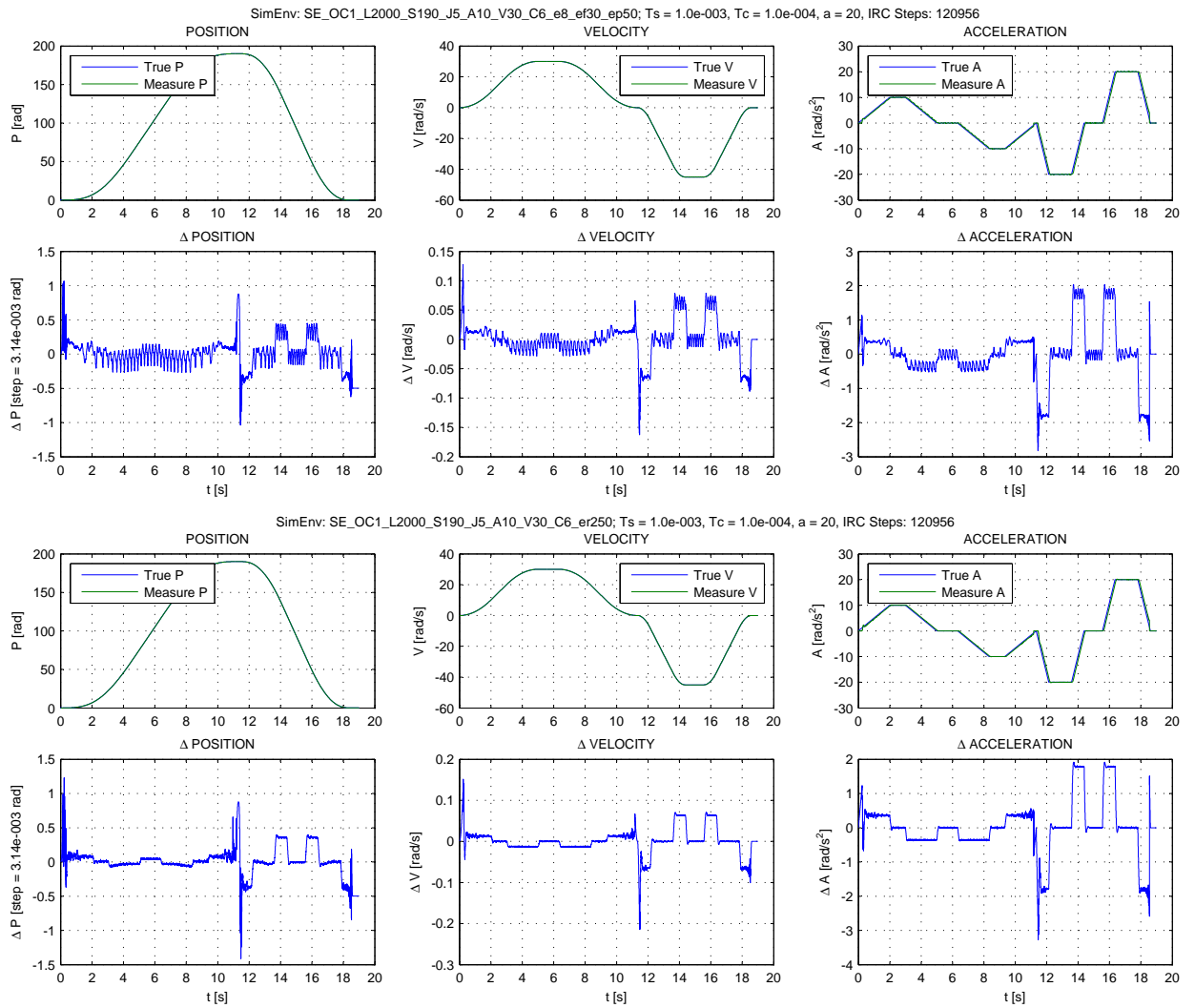
*Volba parametru  $\alpha$  je kompromisem mezi dobrým odfiltrováním šumů a velikostí chyby odhadu při nekonstantním zrychlení. Při konstantní rychlosti nebo zrychlení směřuje střední hodnota chyby k nule. Je-li signál silně poškozen šumem, pak pro udržení uspokojivě malých dynamických chyb odhadů nesmí být derivace zrychlení příliš velká.*

model	$\alpha$	$a = 20\text{rad/s}^2$				$j = -25\text{rad/s}^3$			
		$\Delta v$		$\Delta a$		$\Delta v$		$\Delta a$	
		mean	std	mean	std	mean	std	mean	std
(B)	25	9.44e-6	5.43e-5	2.67e-5	1.62e-3	1.21e-2	1.07e-4	7.77e-1	3.20e-3
	20	8.26e-6	1.92e-5	1.72e-5	2.49e-4	6.38e-2	7.44e-5	1.79e+0	1.05e-3
(D)	25	3.89e-6	7.58e-5	2.05e-4	2.40e-3	1.21e-2	8.91e-5	7.77e-1	2.81e-3
	20	1.28e-5	1.95e-5	6.63e-5	3.17e-4	6.38e-2	8.03e-5	1.79e+0	1.09e-3
(E)	25	1.30e-3	1.21e-2	2.10e-2	3.43e-1	1.66e-2	1.53e-2	8.70e-1	4.45e-1
	20	6.17e-4	6.48e-3	6.63e-3	8.27e-2	6.42e-2	6.09e-3	1.80e+0	8.20e-2
(F)	25	5.03e-5	2.73e-3	3.90e-3	8.18e-2	1.23e-2	1.94e-3	7.82e-1	6.01e-2
	20	8.12e-5	7.66e-4	8.69e-4	9.90e-3	6.38e-2	4.44e-4	1.79e+0	6.11e-3

Tabulka 4: Chyby odhadů v různých částech rychlostního profilu pro různé typy nepřesností snímače.



Obrázek 32: Srovnání důsledků různých typů nepřesností snímače, vše s  $\alpha = 25$ . Srovnání (D) / (E) / (F).



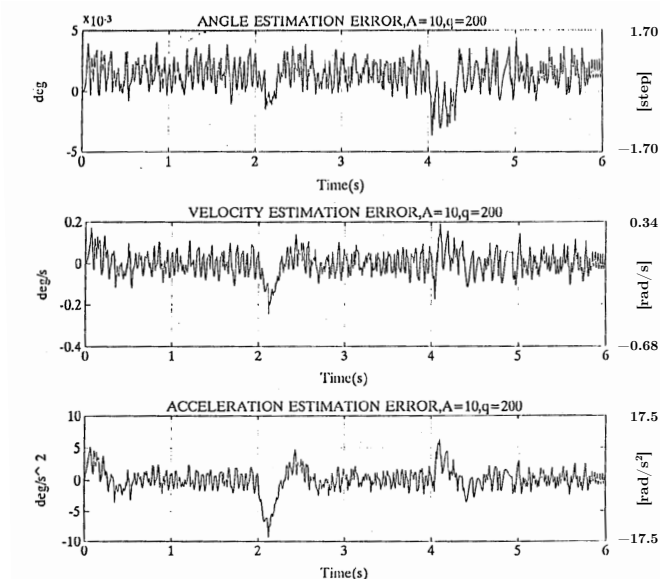
Obrázek 33: Výrazné snížení šumu při snížení na  $\alpha = 20$ . Srovnání (E) / (F).



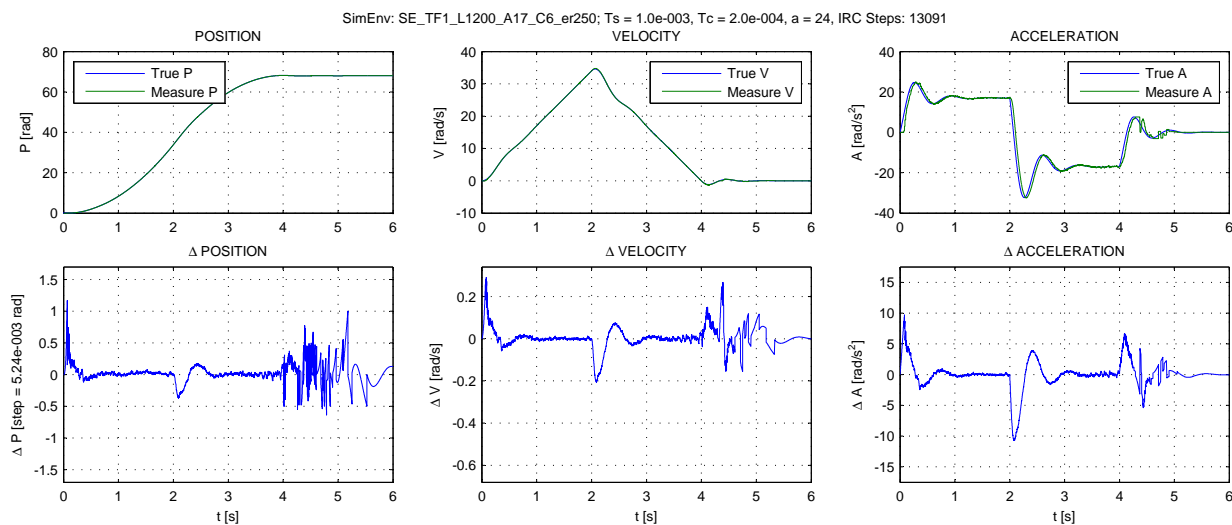
## Srovnání s řešením dle (BÉLANGER, 1992) [3]

Pro zajímavost je ještě uvedeno srovnání s řešením, které je popsáno v článku [3]. Autor použil obdobně navržený Kalmanův filtr pro model se sérií integrátorů. Vstupem Kalmanova filtru však byla ekvidistantní měření počtu pulzů za každých 10 ms.

Srovnání výsledků této metody se zde předloženým řešením pro totožný rychlostní profil, snímač s 1200 pulzy na otáčku, a šum měření modelovaný parametrem  $e_r = 0.25$  ukazují obrázky 34 a 35. Filtr navržený v této práci dává velmi výrazně lepší odhady.



Obrázek 34: Chyby odhadů pro metodu dle článku [3].



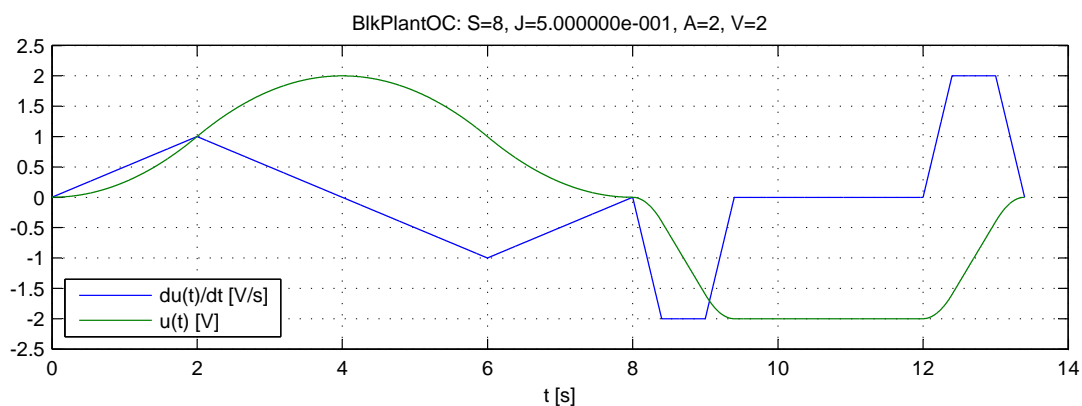
Obrázek 35: Výsledky zde navrženého algoritmu pro parametry shodné s [3].

## 6.2 Měření na reálném snímači

Vyhodnocení chování navrženého filtru v reálných podmínkách bylo provedeno na laboratorním modelu s bezkomutátorovým motorem s vestavěným inkrementálním snímačem s 500 značkami na otáčku. Motor byl řízen servozesilovačem MAXON 205679. Rychlost otáčení hřídele motoru je řízena napětím na analogovém vstupu servozesilovače.

Výstup IRC snímače byl připojen k servozesilovači, a zároveň k převodníku, jehož konstrukci popisuje kapitola 9. Logické výstupy převodníku byly připojeny k logickému analyzátoru, který jejich hodnoty s vzorkovací frekvencí 50 MHz ukládal do počítače.

Pro jednoduchost bylo ovládací napětí generováno programově bez zpětné vazby. Časový profil napětí  $u(t)$ , a tedy i rychlosti, byl obdobně jako při výše popsanych simulacích vytvořen jako optimální trajektorie s omezením na maximální jerk, zrychlení a rychlost. Průběh napětí a jeho derivace ukazuje obrázek 36.

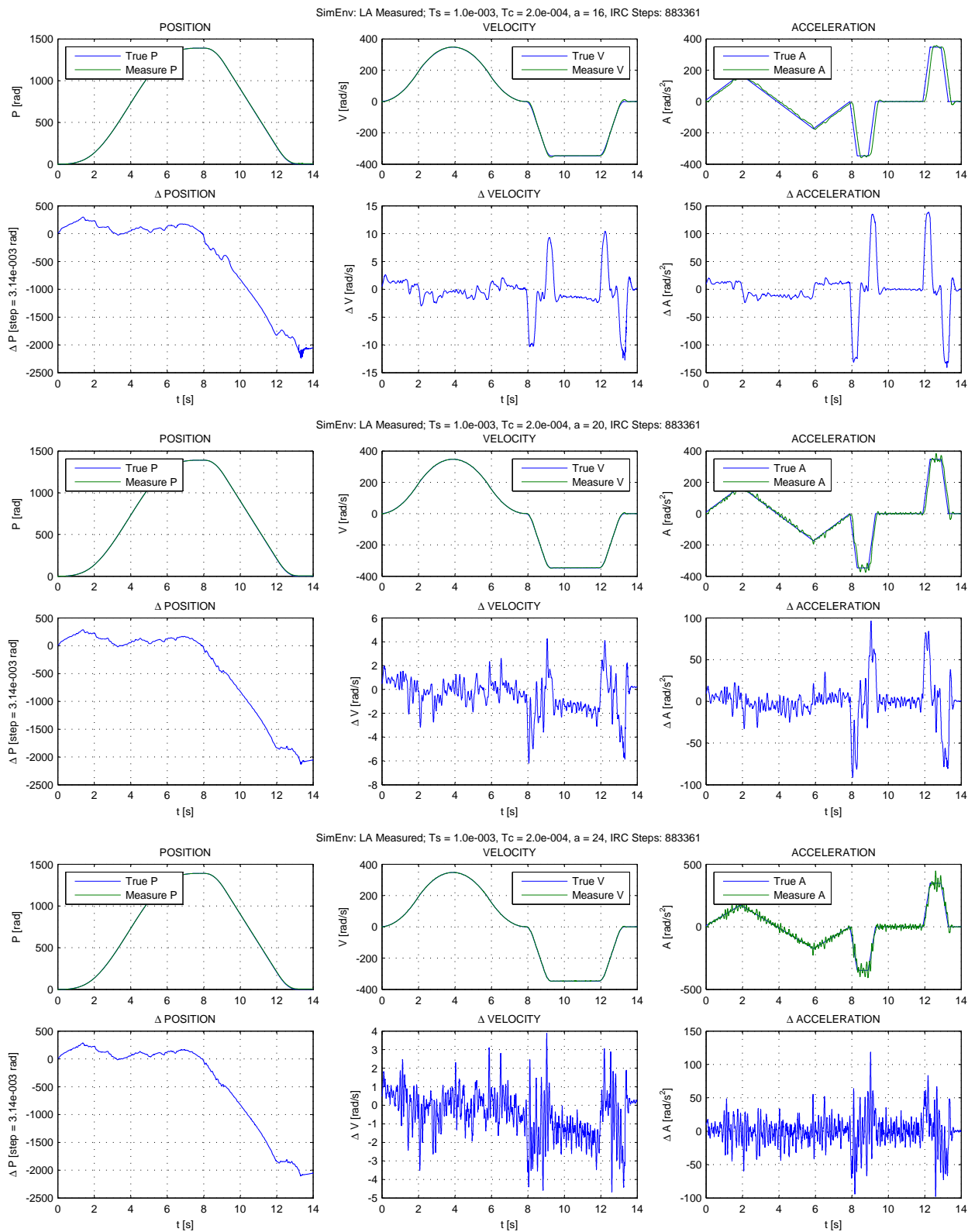


Obrázek 36: Časový profil ovládacího napětí při experimentu na reálném snímači.

V průběhu experimentu docházelo ke značnému kmitání celé aparatury, které jistě působilo jako porucha měřených veličin. Použitý servozesilovač trpí také určitou nesymetrií analogového ovládacího vstupu, takže koncová poloha se liší od počáteční asi o jednu otáčku, ač by se podle rychlostního profilu měly shodovat.

Výsledky zpracování naměřených dat navrženým filtrem – odhady polohy, rychlosti a zrychlení – ukazuje obrázek 37. Srovnání se „skutečnými“ hodnotami je velmi orientační, neboť je ve skutečnosti neznáme, a byly použity hodnoty, které by odpovídaly ideální trajektorii.

Zajímavé je srovnání pro různou volbu  $\alpha = 16 / 20 / 24$  vzhledem k tomu, že měření je velmi silně zatíženo šumem. Je dobře vidět, že nejnižší hodnota  $\alpha$  nejlépe filtruje šum při ustálené rychlosti a zrychlení, i když maxima chyby při velké derivaci zrychlení jsou větší.



Obrázek 37: Výsledky měření na reálném snímači, srovnání pro  $\alpha = 16 / 20 / 24$ .

## 7 Návrh integrace do prostředí reálného času

### 7.1 Koncepce řešení

Součástí této práce je implementace navrženého *filtru* (algoritmu pro odhad polohy, rychlosti a zrychlení z měření IRC snímačem) v programovacím jazyce C++ v podobě optimalizované na běh v reálném čase. Dále je navržen vhodný způsob integrace do real-time prostředí na průmyslovém počítači. Popis návrhu a testovací implementace všech potřebných algoritmů v systému MATLAB byl proveden v přechozích kapitolách.

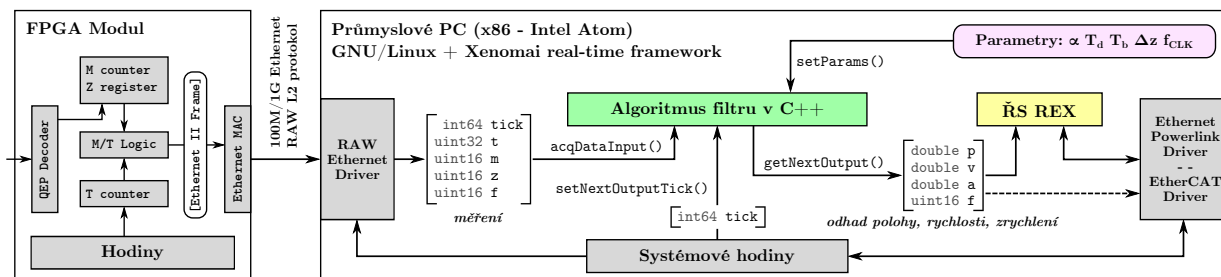
Pro praktické použití těchto algoritmů, tedy pro jejich běh v reálném čase (real-time), je třeba podrobněji prozkoumat a případně optimalizovat tyto aspekty:

- Čas potřebný pro zpracování real-time kritických operací – v našem případě jde o výpočet nových odhadů ve výpočetních blocích Kalmanova filtru a prediktoru, který je třeba provést s každým měřením a musí trvat co nejkratší dobu. V nejhorsím případě se musí „stihnout“ dříve, než přijde další měření. Minimální čas mezi měřeními je dán parametrem měřicího systému  $T_c$ , který volíme v rozsahu 0,1 ms až 0,3 ms.
- Potenciální ztráta přesnosti trvale rostoucích hodnot – algoritmus pracuje s informací o poloze, která v určitých případech může trvale růst nebo klesat (při stálém otáčení jedním směrem), a přitom požadujeme spolehlivou funkci i při dlouhodobém běhu algoritmu po dobu mnoha dnů nebo měsíců. Je třeba se vyrovnat s faktem, že rozlišení čísel s pohyblivou řádovou čárkou (floating point) s vyšší hodnotou klesá. V našem případě je součástí výstupu odhad polohy, kde se toto potenciálně může projevit. Odhad polohy se ale používá i pro interní výpočty v bloku Kalmanova filtru, a proto nelze problém snadno řešit reprezentací hodnoty jako celého čísla s pevným rozlišením. Problém v rámci této práce necháváme otevřený.

Proti algoritmu použitému pro simulační výpočty zavedeme navíc tyto předpoklady:

- Systémový čas ( $\tau_j$ , `tick`) je 64bitové celé číslo s rozlišením 1 ns.
- Vstup z měřicího systému – jedno *měření* – je trojice čísel:
  - `uint32 T` – čítač času, rozlišení  $f_{CLK}^{-1}$ , neresetuje se – pro 100 MHz přeteče přibližně každých 40 s,
  - `uint16 M` – čítač IRC pulzů, nenuluje se, po hodnotě 65535 přetéká na 0,
  - `uint16 F` – pole bitových značek (*flags*), obsahuje informaci o směru otáčení při posledním zaznamenaném pulzu a další signalizaci z měřicího systému.

Pro reálnou aplikaci navrženého algoritmu měření je uvažováno využití real-time prostředí Linux/Xenomai [30] a případné propojením s řídicím systémem REX [31], který běh nad Xenomai též podporuje. Návrh propojení vstupů a výstupů navrženého algoritmu filtru do real-time prostředí ukazuje obrázek 38.



Obrázek 38: Integrace algoritmu do real-time prostředí.

Dekódování signálu z IRC snímače a jeho předzpracování adaptivním M/T algoritmem je realizováno na FPGA modulu.

*Měření* – vstupní data pro filtr – jsou do PC přenášena přes rozhraní Ethernet. Data jsou zapouzdřena přímo do Ethernet rámců (*Ethernet type-II frame*) bez použití vyššího protokolu. Tím je dosaženo jednoduchosti a minimálního zpoždění přenosu.

Problematický faktor je, že hodiny na FPGA modulu a PC nejsou synchronizovány. Pro filtr je klíčová přesnost informace o časové diferencii měření ( $T_k = t_k - t_{k-1}$ ), kterou počítáme jako rozdíl stavu čítače T, takže synchronitou hodin není ovlivněna. Zároveň ale potřebujeme znát čas  $t_k$  podle systémových hodin. Navržené řešení je měřit tento čas jako čas přijetí ethernetového rámce počítačem. Dojde tak sice k chybě v řádu desítek us, ale tento čas je využíván jen pro stanovení intervalu predikce k výstupnímu odhadu, jehož přesnost v tomto měřítku není kritická. Lepší řešení v podobě synchronizace hodin například protokolem *IEEE 1588 Precision Time Protocol* by bylo nesrovnatelně složitější.

*Výstup filtru* – odhady polohy, rychlosti a zrychlení v požadovaných časech ( $\tau_j$ ) mohou být využity buď v řídicím systému běžícím na stejném počítači, nebo mohou být zpřístupněny přes nějaké komunikační rozhraní, například *Ethernet Powerlink* nebo *EtherCAT*.

V rámci této práce byl vytvořen optimalizovaný algoritmus filtru. Implementace je popsána v další kapitole. Úplná realizace integrace do real-time prostředí je již mimo rozsah práce.

Pro testy bylo použito průmyslové PC MOXA V2400 s procesorem Intel Atom:

- CPU: Intel(R) Atom(TM) CPU N270, 1.60 GHz, RAM: 1 GB DDR2,
- OS: Linux Debian 7.0 32bit, kernel 3.2.0-4-rt-686-pae.

## 7.2 Popis implementace algoritmu filtru v C++

Pro implementaci filtru byl zvolen jazyk C++. Potřebné výpočty inverze matice a řešení maticové soustavy lineárních rovnic jsou řešeny s využitím knihovny *Armadillo* (*C++ linear algebra library*) [29], která poskytuje výkonnostně optimalizované a snadno použitelné algoritmy pro standardní operace z oblasti lineární algebry a práce s maticemi. Výhodou je i volná licence *Mozilla Public License 2.0*, která nebrání použití s jiným kódem.

### Organizace zdrojových kódů

Zdrojové soubory jsou uloženy na přiloženém CD v archivu *IRC\_FilterCpp.tgz*.

Vlastní implementace výpočetních algoritmů je tvořena třídami *FilterKalman*, *FilterPredictor*, *IrcFilter*, a funkcí *expm2*. Tyto soubory jsou uloženy ve složce *./source/*.

Pro otestování funkce a rychlosti jsou pak vytvořeny testovací programy T-FILTER (spuštění filtru nad daty ze souboru) a T-PERF (analýza výpočetní náročnosti) uložené ve shodně pojmenovaných složkách. Tyto je možné zkompileovat a spustit pod operačním systémem GNU/Linux. Vše bylo testováno v distribuci Debian GNU/Linux 7.0.

Pro zkompileování zdrojových kódů je třeba mít sestavenou knihovnu *Armadillo* (viz výše), a nainstalované knihovny LAPACK a BLAS (balíky *liblapack-dev* a *libblas-dev*).

### Třída `IrcFilter`

Slouží k propojení výpočetních bloků a přizpůsobení vnějšího rozhraní do podoby vhodné pro použití v real-time prostředí podle obrázku 38.

```
# void setParams(double deltat, double deltaz, double a, double td, double tb)
```

Funkce nastaví všechny volitelné parametry:

- `deltat` – časová jednotka T čítače =  $1/f_{CLK}$ ,
- `deltaz` – polohová jednotka M čítače =  $2\pi/L$ ,
- `a` – návrhový parametr  $\alpha$  pro Kalmanův filtr,
- `td` – „dead-time“  $T_d$  [s],
- `tb` – „bound-check-guard-time“  $T_b$  [s].

```
# void acqDataInput(int64_t tick, uint32_t t, uint16_t m, uint16_t z, uint16_t f)
```

Funkce pro zpracování příchozího *měření*.

- `tick` – systémový čas okamžiku měření,
- `t` – přijatá hodnota čítače T,
- `m` – přijatá hodnota čítače M,
- `z` – přijatá hodnota registru Z,
- `f` – „flags“ – pole bitových značek, prozatím se využívá jen pro informaci o směru.

```
# void setNextOutputTick(int64_t tick)
```

Funkce nastaví nejbližší budoucí systémový čas, pro který budeme chtít znát výstupní odhady, tj. pro něj budeme počítat predikci stavu. Musí být větší, než s jakým byla posledně volána funkce *acqDataInput*.

- `tick` – systémový čas pro nejbližší budoucí výstupní odhad.

```
# void getNextOutput(double * p, double * v, double * a, unsigned int * f)
```

Funkce pro přečtení hodnot výstupního odhadu v čase nastaveném posledním voláním funkce *setNextOutputTick*.

- `*p` – výstupní parametr – poloha,
- `*v` – výstupní parametr – rychlost,
- `*a` – výstupní parametr – zrychlení,
- `*f` – výstupní parametr – „flags“, značky informující o kvalitě výstupních hodnot;  
1 = detekováno selhání predikce, 2 = překročen dead-time, 4 = zatím nepřišlo měření.

### **Třída `FilterKalman`**

Výpočetní blok – Kalmanův filtr. Rozhraní dokumentováno v hlavičkovém souboru.

### **Třída `FilterPredictor`**

Výpočetní blok – prediktor. Rozhraní dokumentováno v hlavičkovém souboru.

## Funkce expm2

```
# void expm2(arma::mat33 A, arma::mat33 * EPos, arma::mat33 * ENeg)
```

Výpočet maticové exponenciály kladného a záporného argumentu. Algoritmus viz 4.4.

- $A$  – argument exponenciály, matice  $3 \times 3$ .
- $*EPos$  – výstupní hodnota  $e^A$ , matice  $3 \times 3$ .
- $*ENeg$  – výstupní hodnota  $e^{-A}$ , matice  $3 \times 3$ .

## Program T-FILTER

Program slouží k otestování správné funkce algoritmu nad vstupy ze souboru. Parametry:

- $-a$  – návrhový parametr  $\alpha$  pro Kalmanův filtr, obvykle 18 – 24,
- $-d$  – dead-time ( $T_d$ ) [s], obvykle 0.03 až 0.15,
- $-l$  – počet pulzů na otáčku ( $L$ ), testovací data generována s  $L = 2000$ ,
- $-s$  – sample-time [s], perioda výstupních odhadů, obvykle 0.001,
- $-t$  – koncový čas [s],
- $-i$  – cesta ke vstupnímu souboru.

Příklad spuštění a ladicího výstupu:

```
~/FilterCpp/T-FILTER$ ./bin/32/t-filter -a 24 -d 0.050 -l 2000 -s 0.001 -t 19 \  
-i ./input/OC1_S190_Tc2.txt > out.txt  
Offline simulation params: a=24.0 Td=0.050 L=2000 Ts=0.0010 T=19.0  
FilterKalman: setParams a=24.0 td=0.050  
FilterKalman: setParams PRECOMPUTE dTx=1.242582e-05 dTy=7.882201e-04  
FilterPredictor: setParams step=0.00314 td=0.050  
Loaded DATA, rows: 98733  
[T000.000] FilterInterface: INIT tick=0 t=0 m=0  
[T000.156] FilterKalman: dt over dead-time, dt=0.156  
[T000.220] FilterPredictor: hit bound_h  
...  
[T018.500] FilterPredictor: hit bound_l  
[T018.524] FilterPredictor: hit bound_l  
[T018.564] FilterPredictor: dead-time, dt=0.051  
TOTAL EXECUTION TIME: 1.1417 s
```



Formát vstupního souboru je textový, každé měření jeden řádek – trojice  $\langle T \rangle \langle M \rangle \langle D \rangle$ .

$T$  (rozsah uint32) – čas od počátku měření s jednotkou  $10^{-8}$  sekundy ( $f_{CLK} = 100$  MHz).

$M$  (rozsah uint16) – stav čítače počt pulzů.

$D$  – směr posledního pulzu – hodnota +1 pro kladný, -1 pro záporný.

Výstupní odhady polohy, rychlosti a zrychlení jsou vypisovány na standardní výstup ve formátu  $\langle čas [s] \rangle \langle poloha [rad] \rangle \langle rychlost [rad/s] \rangle \langle zrychlení [rad/s^2] \rangle \langle flags \rangle$ :

```
4.999 7.497003e+01 3.000337e+01 1.885496e-01 0
5.000 7.500003e+01 3.000336e+01 1.833428e-01 0
5.001 7.503003e+01 3.000337e+01 1.786444e-01 0
5.002 7.506003e+01 3.000335e+01 1.734107e-01 0
```

## Program T-PERF

Program slouží k měření času potřebného pro zpracování příchozího měření v závislosti na parametru  $\alpha = 18 / 24 / 30$  a na času mezi měřeními  $dt$  v rozsahu 0,1 ms až 300 ms. Je měřen procesorový čas potřebný pro vykonání funkce `IrcFilter::acqDataInput`.

Výsledky jsou vypsané ve formátu  $\langle alpha \rangle \backslash t \langle dt [us] \rangle \backslash t \langle čas [us] \rangle$ . Příklad:

```
~/FilterCpp/T-PERF$ ./bin/32/t-perf
18 100 006.4
18 300 008.6
18 1000 009.3
18 3000 008.7
18 10000 009.9
...
18 300000 010.0
24 100 008.7
...
24 300000 010.0
...
```

Podrobnější popis a vyhodnocení výsledků je obsahem další kapitoly.

## 7.3 Vyhodnocení a optimalizace výpočetní náročnosti

### Kritické operace prováděné v reálném čase

Časově kritické jsou všechny operace, které je třeba provést při každém měření, tj. výpočet nového odhadu Kalmanovým filtrem a výpočet nové časové predikce. Nejnáročnější operací je výpočet maticových exponenciál ve funkci `FilterKalman::setInput`:

$$EPos = e^{A_R T_k}, ENeg = e^{-A_R T_k} \quad (7.1)$$

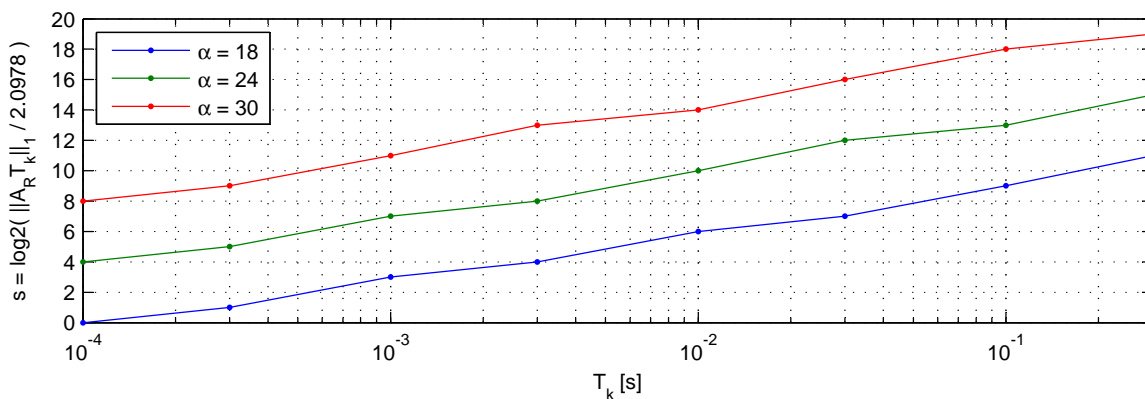
$A_R$  je předem vypočtená matice závislá na parametru  $\alpha$ .  $T_k$  je interval mezi měřeními.

Tento výpočet je realizován algoritmem Padeho aproximace popsaným v kapitole (4.4). Algoritmus využívá techniku měřítkování a umocňování, což znamená že vstupní matice  $A_R T_k$  je vydělena koeficientem  $2^s$  a výsledek výpočtu je pak  $s$ -krát umocněn na druhou.

$$\|A_R T_k\|_1 / 2^s < 2,0978 \rightarrow s = \max(0, \lceil \log_2(\|A_R T_k\|_1 / 2,0978) \rceil) \quad (7.2)$$

Koeficient  $s$  je volen podle vztahu (7.2). Umocnění matice je ovšem výpočetně relativně drahá operace. *Problematická vlastnost tohoto řešení je, že výpočetní náročnost a tím čas zpracování závisí na normě matice, tedy na hodnotě  $T_k$ , která se pro každé měření liší.*

Závislost potřebného počtu umocnění matice (tj. hodnoty  $s$ ) pro reálně použitelný rozsah  $T_k = 0,1$  ms až 300 ms a hodnoty  $\alpha = 18 / 24 / 30$  ukazuje obrázek 39.

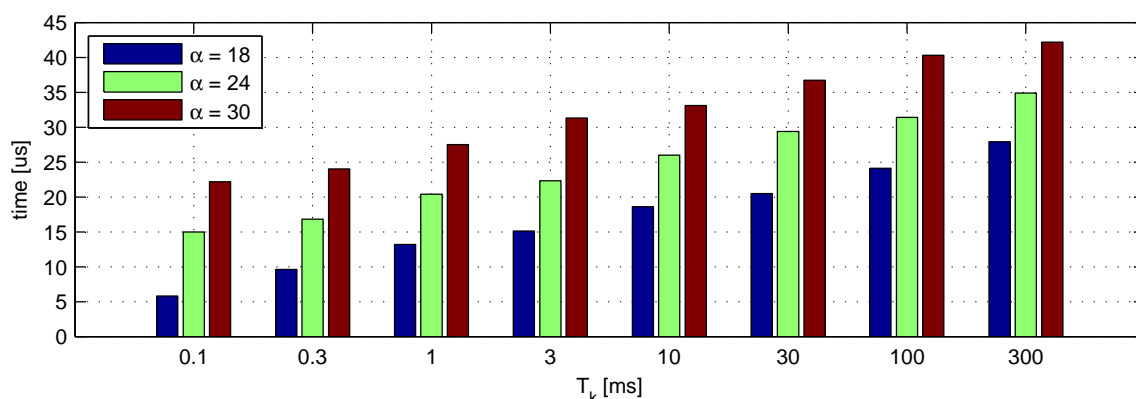


Obrázek 39: Závislost koeficientu umocňování  $s$  na parametrech  $T_k$  a  $\alpha$ .

Testováním algoritmu bylo zjištěno, že právě opakované umocňování matice má zásadní podíl na časové náročnosti výpočtu. Tento jev podrobněji analyzujeme v dalším textu.

## Test časové náročnosti neoptimalizovaného algoritmu

Pro analýzu časové náročnosti tohoto výpočtu byl vytvořen testovací program T-PERF. V zájmu zjednodušení není test určen pro real-time prostředí (např. Xenomai), ale spouští se jako obyčejný proces v operačním systému Linux. Testovaný výpočet tak sdílí procesorový čas s operačním systémem a s ostatními běžícími procesy. Pro omezení tohoto vlivu je výpočet se shodnými parametry vždy opakován 50 000 krát, a čas zprůměrován. Výsledky jsou tak drobně zkresleny k delším časům.



Obrázek 40: Závislost času zpracování měření na parametrech  $\alpha, T_k$  – neoptimalizovaný algoritmus.

Výsledky testovacího programu spuštěného na počítači s procesorem Intel Atom N270 (viz výše) jsou shrnuty na obrázku 40. Graf ukazuje čas potřebný pro zpracování měření, tj. pro vykonání funkce `IrcFilter::acqDataInput`, v závislosti na čase od předchozího měření  $T_k$  a na parametru filtru  $\alpha$ . Je vidět, že závislost času na parametrech přibližně odpovídá závislosti koeficientu umocňování  $s$  podle obrázku 39.

Rozdíly jsou velmi výrazné. Při  $\alpha = 30$ ,  $T_k = 100$  ms – 18 umocnění matice (a to ještě dvakrát, pro *EPos* a *ENeg*) – čas 40 us. Proti tomu při  $\alpha = 18$ ,  $T_k = 0,1$  ms – umocňování není třeba – čas 6 us.

Zásadní kritérium je, že výpočet musí trvat bezpečně kratší čas, než je interval mezi měřeními. V reálné aplikaci ale na stejném počítači poběží ještě další úlohy, pro které musí zůstat dostatečný (ideálně co největší) podíl procesorového času.

Nejhůře vychází situace pro velké  $\alpha$  a malé  $T_k$ . Při  $\alpha = 30$  a  $T_k = 0,1$  ms trvá výpočet 23 us, takže by téměř 25 % procesorového času bylo spotřebováno na měření z jediného IRC snímače! Pro větší  $T_k$  sice čas výpočtu roste, ale závislost má logaritmický charakter. Takže pro  $T_k = 0,3$  ms vzroste čas jen na 24 us, což je již výrazně příznivějších 8 %.

Je vidět, že má smysl hledat možné optimalizace algoritmu, což je obsahem další kapitoly.

## Optimalizovaný algoritmus

Pro optimalizaci se nabízí využít toho, že argument maticové exponenciály má tvar součinu  $A_R T_k$ , kde pouze  $T_k$  se v průběhu měření mění. Přitom platí rovnost:

$$e^{A_R T_k} = e^{A_R (T_k - T_x)} \cdot e^{A_R T_x} = e^{A_R T'_k} \cdot e^{A_R T_x}, \quad T'_k = (T_k - T_x), \quad T'_k > 0 \quad (7.3)$$

Idea je, že výraz  $e^{A_R T_x}$  může být předpočítán předem pro dané konstantní  $\alpha$  a mnoho různých hodnot  $T_x$ . V reálném čase je pak potřeba počítat jen  $e^{A_R T'_k}$ . Budou-li hodnoty  $T_x$  zvoleny vhodným způsobem, s dostatečně jemným dělením, tak bude  $T'_k$  malé, tím norma  $\|A_R T'_k\|_1$  malá, a výpočet rychlý.

V ideálním případě by měly být hodnoty  $T_x$  zvoleny tak, aby pro libovoně  $T_k$  v užitečném rozsahu bylo možné vybrat  $T_x$  takové, že  $\|A_R T'_k\|_1 < 2,0978$ , takže při výpočtu  $e^{A_R T'_k}$  vůbec nebude třeba provádět závěrečné umocňování.

Například pro  $\alpha = 30$ , což je asi nejvyšší použitelná hodnota, to ovšem znamená podmínku:

$$\Delta T_x = T'_{max} = \frac{2,0978}{\|A_R\|_1} = 6,331 \cdot 10^{-4} \text{ ms} \quad (7.4)$$

Pro pokrytí intervalu od 0,1 do 200 ms (horní hranice je dána parametrem filtru  $T_d$ ) by pak ovšem bylo nutné předpočítat 315732 hodnot  $e^{A_R T_x}$ . To není nereálné, ale trvalo by to asi 10 sekund. Případně by mohly být předpočtené hodnoty pro předem vybrané  $\alpha$  načteny z trvalé paměti (pevného disku).

Nabízí se však ještě možnost předpočítat dvě sady hodnot – jednu s „hrubým“ dělením  $\Delta T_y$ , a jednu s „jemným“ dělením  $\Delta T_x$ :

$$\{T_y(iy) = iy \cdot \Delta T_y\} \in \langle \Delta T_y, T_d \rangle, \quad \{T_x(ix) = ix \cdot \Delta T_x\} \in \langle \Delta T_x, \Delta T_y \rangle \quad (7.5)$$

$$\Delta T_x = \frac{2,0978}{\|A_R\|_1}, \quad \Delta T_y = \sqrt{T_d \cdot \Delta T_x} \quad (7.6)$$

Pro náš případ s  $T_d = 200 \text{ ms}$ ,  $\alpha = 30$ :

$$\Delta T_x = 6,331 \cdot 10^{-7} \text{ s}, \quad \Delta T_y = 3,558 \cdot 10^{-4} \text{ s}$$

$$\max(ix) = (\Delta T_y - \Delta T_x) / \Delta T_x = 561, \quad \max(iy) = (T_d - \Delta T_y) / T_y = 561$$

je třeba předpočítat 561 hodnot  $e^{A_R T_x}$  a 561 hodnot  $e^{A_R T_y}$ , což už není problém.

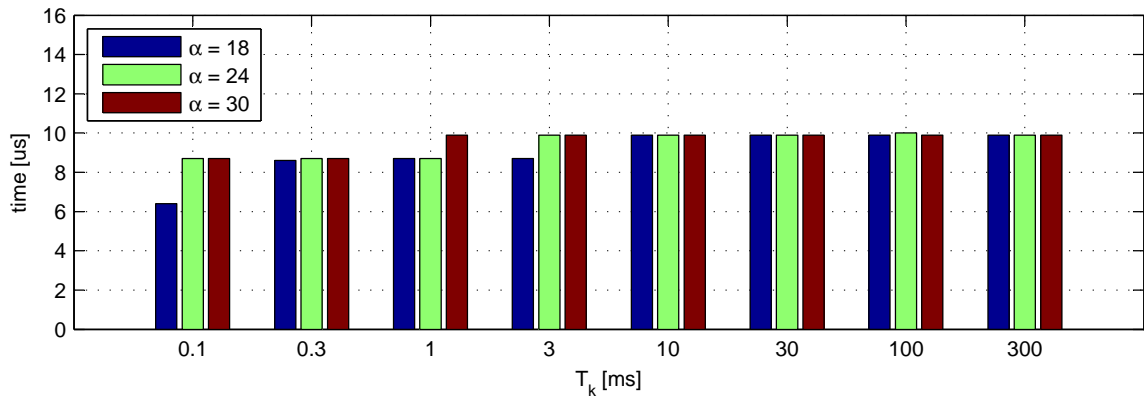
Pro určitou hodnotu  $T_k$  pak bude výpočet finálně realizován takto:

$$iy = \lfloor T_k / \Delta T_y \rfloor, \quad ix = \lfloor (T_k - T_y(iy)) / \Delta T_x \rfloor \quad (7.7)$$

$$e^{A_R T_k} = e^{A_R(T_k - T_y(iy) - T_x(ix))} \cdot e^{A_R T_y(iy)} \cdot e^{A_R T_x(ix)} \quad (7.8)$$

kdy máme zaručeno, že  $\|A_R(T_k - T_y(iy) - T_x(ix))\|_1 < 2,0978$ . Pokud navíc vyjde  $ix = 0$  nebo  $iy = 0$ , tedy nulový čas  $T_x$  nebo  $T_y$ , může se příslušný činitel v (7.8) vypustit.

Takto optimalizovaný algoritmus tedy potřebuje zaručeně nejvýše dvě maticová násobení. Implementace v C++ byla popsáním způsobem upravena. Výsledky testovacího programu T-PERF s tímto optimalizovaným algoritmem ukazuje obrázek 41.



Obrázek 41: Závislost času zpracování měření na parametrech  $\alpha, T_k$  – optimalizovaný algoritmus.

Z grafu je patrné, že se podařilo čas zpracování kritické funkce `IrcFilter::acqDataInput` pro libovolné  $\alpha$  a  $T_k$  shora omezit na 10 us. Za cenu předpočítání dat, které je třeba provést jen jednou při nastavení parametrů. Při  $\alpha = 30, T_d = 300$  ms trvá předpočítání 40 ms.

Tato optimalizace bude mít jistý vliv na přesnost výpočtu maticové exponenciály. Tento aspekt nebyl úplně podrobně prozkoumán. Byly pouze porovnány výsledky neoptimalizovaného a optimalizovaného algoritmu na různých testovacích vstupech. Rozdíly se projeví v odhadech rychlosti a zrychlení, ale naprosto nevýznamně.

## 8 Návrh FPGA logiky pro měření metodou M/T

V této kapitole popíšeme návrh řešení pro zpracování signálu z IRC snímače adaptivní metodou M/T, která byla specifikována v kapitole 3.

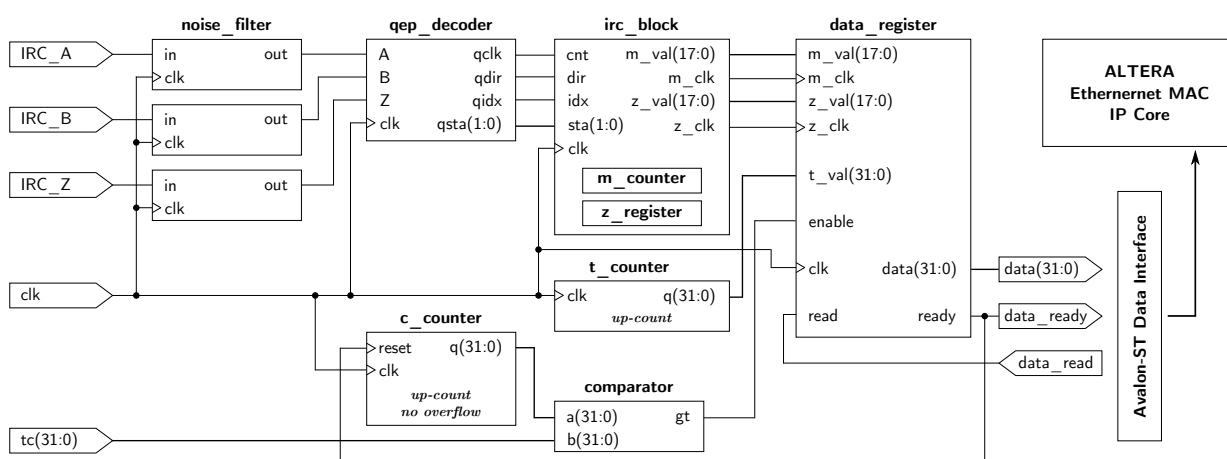
Řešíme vlastně určité předzpracování elektrických signálů A, B, Z, které jsou výstupem snímače, do podoby informace – *měření*, která bude přenášena přes nějaké komunikační rozhraní do počítače, kde bude dále zpracovávána.

Tento problém nelze řešit softwarově – programem na klasickém procesoru. Nejlepší cestou je použití programovatelného hradlového pole – integrovaného obvodu FPGA (*Field Programmable Gate Array*). Tyto obvody jsou v současnosti cenově dostupné pro uvažovanou aplikaci, a umožňují realizovat v podstatě libovolně složitou logickou strukturu.

Pro vývoj programového kódu v jazyce VHDL (*VHSIC Hardware Description Language*) budeme uvažovat využití vývojové desky *TERASIC DE2-115* [34] osazené hradlovým polem *ALTERA Cyclone IV EP4CE115*. Tato deska obsahuje kromě samotného FPGA obvodu řadu periférií včetně dvou rozhraní *Gigabit Ethernet*, která lze použít pro rychlou komunikaci s počítačem.

Pro přizpůsobení elektrických signálů z IRC snímače ke vstupům FPGA obvodu a jeho ochranu před poškozením byl navržen signálový převodník, jehož výstup je možné připojit přímo k vývojové desce DE2-115. Viz kapitola 9.

Blokové schéma navrženého řešení ukazuje obrázek 42.



Obrázek 42: Blokové schéma logiky pro měření M/T metodou pro FPGA implementaci.

## Vstupy a výstupy

- $IRC\_A$ ,  $IRC\_B$ ,  $IRC\_Z$  – kvadraturní signály A a B, index signál Z z IRC snímače.
- $clk$  – společný hodinový signál.
- $tc(31:0)$  – volitelný parametr  $T_c$  v jednotkách počtu hodinových tiků.
- $data\_ready$ ,  $data(31:0)$ ,  $data\_read$  – rozhraní vyčítání naměřených hodnot.

## Koncepce řešení

Vstupní signály jsou filtrovány proti případným zákmitům v blocích *noise\_filter* a následně dekódovány v bloku *qep\_decoder* na polohové pulzy s rozlišením směru. Tento blok zároveň řeší synchronizaci index signálu. Problém dekódování byl podrobně popsán v kapitole 2.

Dekódované pulzy jsou počítány v bloku *irc\_block*. Ten obsahuje 16bitový čítač *m\_counter*, jehož hodnota se s každou hranou na vstupu *cnt* zvýší nebo sníží o jedna podle hodnoty vstupu *dir*. Hrana na vstupu *idx* znamená, že aktuální hodnota čítače *m\_counter* odpovídá nulové značce snímače – tato hodnota je přepsána do 16bitového registru *z\_register*. Na vstup *sta(1:0)* přichází aktuální kvadraturní stav – stav signálů A, B – synchronizovaný se vstupem *cnt*. Výstupní signály *m\_val(17:0)* a *z\_val(17:0)* obsahují kromě 16bitové hodnoty v nejvyšších bitech navíc ještě 1 bit informace o směru a 2 bity o kvadraturním stavu. Na výstupech *m\_clk* a *z\_clk* jsou generovány pulzy při aktualizaci čítače a registru.

Čítač *t\_counter* je volně běžící, jeho hodnota se používá jako časová značka měření. Podle rozdílu přijatých časových značek z po sobě jdoucích měření určíme časovou diferenci mezi měřeními pulzy.

Čítač *c\_counter* měří čas od posledního odeslaného měření. Jeho výstup je porovnáván se vstupní hodnotou *tc* blokem *comparator*, jehož výstup je připojen na vstup *enable* bloku *data\_register*. Je-li čas větší než *tc*, je povoleno odeslání dalšího měření.

Blok *data\_register* sestavuje data k odeslání. Je-li aktivní vstup *enable*, pak hrana na vstupu *m\_clk* spustí sestavení dat – hodnoty T (*t\_counter*), C (*c\_counter*), Z (*z\_register*) a F (16bitové pole značek sestavené z nejvyšších 3 bitů vstupů *m\_val* a *z\_val*, zbytek je vyhrazen pro budoucí použití).

Uvedení popis je dosti zjednodušený, a klade si za cíl jen rychlé seznámení s koncepcí řešení, která by měla být dále rozvíjena směrem k implementaci ve VHDL.

## 9 Převodník pro připojení IRC snímače k FPGA

Standardní průmyslové IRC snímače nedisponují elektrickým výstupním rozhraním vhodným pro přímé připojení k FPGA čipu. Signál přiváděný na vstupy FPGA musí mít obvykle napěťové úrovně CMOS 3,3 V. Proti tomu standardní snímače mají výstupní signál nejčastěji diferenciální podle normy RS-422 s napájecím napětím 5 V nebo signál typu HTL s napájecím napětím 10 až 30 V.

Častým požadavkem může být připojení snímače zároveň k FPGA a k dalšímu zařízení, např. k výkonovému řídicímu bloku motoru. Snímač také vyžaduje napájení z externího zdroje, přičemž při aplikaci ve složitějším řídicím systému by mohlo dojít k problémům s rozdíly elektrických potenciálů tohoto zdroje a zdroje pro FPGA blok. Proto budeme požadovat galvanické oddělení signálů z IRC snímače od vstupů FPGA.

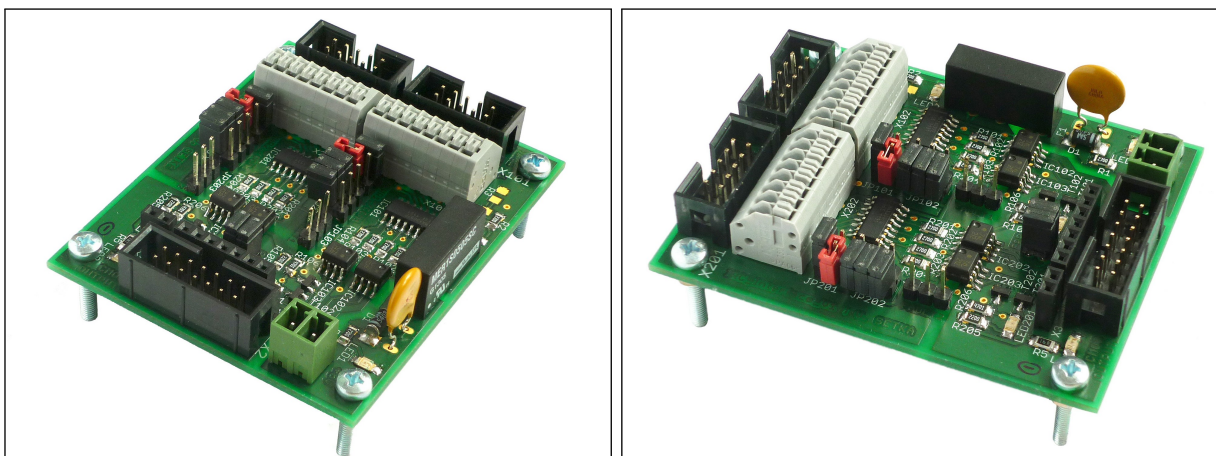
Tato kapitola se zabývá konstrukcí převodníku, který řeší výše popsané problémy. Základní požadavky můžeme formulovat takto:

- schopnost zpracovat vstupní signály typu RS-422, TTL, HTL 10 – 30 V,
- detekce chybné úrovně diferenciálních signálů, např. při zkratu nebo přerušení vedení,
- výstupní logické napěťové úrovně CMOS 3.3 V,
- galvanické oddělení signálů z IRC snímače od vstupů FPGA pomocí optočlenění,
- galvanické oddělení napájení převodníku od vstupních i výstupních signálů,
- minimální zpoždění průchodu signálu (*propagation delay*), optimálně < 500 ns,
- minimální zkreslení délky pulzu, tj. rozdíl mezi zpožděním vzestupné a sestupné hrany (*pulse skew*), optimálně < 50 ns.

Motivace pro vlastní konstrukci spočívá v dosažení vynikajících parametrů v oblasti časového zkreslení signálu. Metoda výpočtů a zpracování signálu pomocí FPGA navržená v této práci stojí na předpokladu co nejpřesnějšího měření elektrického signálu v časové oblasti. Zkreslení na začátku řetězce zpracování signálu – na úrovni elektrického převodníku – není pro metodu zásadní, ale později jej už nelze kompenzovat, a proto má smysl snažit se jej minimalizovat. Komerčně dostupné převodníky zpravidla nejsou primárně navrženy pro takové použití, a požadovaných parametrů nedosahují nebo je vůbec nezaručují.

Vlastní konstrukce převodníku zároveň přinese ověření vlastností zvoleného odbovodného řešení a vybraných součástek pro případnou pozdější konstrukci kompletního rozhraní pro přesné měření signálu z IRC snímačů s integrovaným FPGA.





Obrázek 43: Fotografie realizovaného převodníku.

## 9.1 Použité součástky

### Zpracování vstupních signálů

Pro zpracování diferenciálních signálů podle normy RS-422 je použit integrovaný obvod MAXIM MAX3097E [21]. Je přímo určen pro použití jako vysoce přesný přijímač signálu z IRC snímačů. V jednom pouzdře typu SOIC-16 jsou sdruženy 3 diferenciální přijímače s detekcí chybného vstupního signálu.

Rozdíl napětí mezi komplementárními vstupy menší než 275 mV nebo souhlasné napětí mimo povolený rozsah jsou vyhodnoceny jako chybový stav. Vzájemný zkrat obou vstupů nebo odpojení jednoho z nich při použití terminačního rezistoru znamená téměř nulové napětí mezi vstupy, tedy také chybový stav.

Časové parametry jsou pro uvažovanou aplikaci vynikající: *propagation delay* < 75 ns, *pulse skew* < 10 ns.

Tento integrovaný obvod lze použít i pro zpracování jednopólového (nediferenciálního, *single-ended*) signálu s téměř libovolnými napěťovými úrovněmi. Omezení je na maximální vstupní napětí  $\pm 25$  V proti vstupní zemi převodníku.

V případě jednopólového signálu se negativní vstup diferenciálního páru připojí na pevné napětí odpovídající požadované rozhodovací úrovni. Nastavení rozhodovací úrovně na 2,0 V vyhoví pro většinu snímačů s výstupem TTL nebo HTL. Proto je toto napětí možné připojit na negativní vstupy pomocí propojek přímo na desce zde navrženého převodníku.

## Optické oddělení výstupu

Pro galvanické oddělení výstupu a převod na napěťové úrovni CMOS 3,3 V jsou použity rychlé optočleny typu Avago ACPL-074L [22]. Jedno pouzdro typu SOIC-8 obsahuje 2 optočleny s rychlou LED, detektorem s fotodiodou a zesilovačem s CMOS výstupem. Napájecí napětí výstupního bloku může být 5 V nebo 3,3 V. V našem případě je napájecí napětí 3,3 V přivedeno z FPGA desky.

Pro jeden IRC snímač potřebujeme přenášet 4 výstupní signály – dva kvadrurní A a B, jeden index (absolutní poloha) Z, a pomocný signál indikující chybový stav některého vstupního kanálu. Použity jsou tedy dva optočleny.

Konkrétní typ optočlenu byl vybrán s ohledem na časové parametry. Běžné typy optočlenů s interním fototranzistorem mají čas zpoždění hran v řádu jednotek až desítek  $\mu\text{s}$  a nezaručený maximální rozdíl zpoždění vzestupné a sestupné hrany, což nevyhovuje požadavkům této aplikace. Parametry použitého typu: *propagation delay* < 40 ns, *pulse skew* < 25 ns.

## Napájení převodníku

Převodník vyžaduje pro svou činnost napájecí napětí 5 V. Vzhledem k požadavku na galvanické oddělení je použit integrovaný izolační DC/DC měnič Murata MER1S0505SC [23].

## 9.2 Popis obvodového řešení

Zapojení se skládá ze zdrojové části s DC/DC měničem a ze 2 shodných bloků signálového převodníku. Schéma zapojení a podklady k desce plošných spojů jsou v příloze B.

Vstup zdrojové části je chráněn proti přepólování vratnou pojistkou F1 a diodou D1. Přítomnost napájecího napětí signalizuje LED1. Následuje DC/DC měnič s blokovacími kondenzátory C1, C2, C3. Napětí na výstupu měniče signalizuje LED2.

Snímač je možné připojit standardním konektorem  $2 \times 10$  pinů – X101, nebo svorkovnicí pro připojení libovolných vodičů – X102. Následuje *line receiver* MAX3097E a dva optočleny ACPL-074L. Napájení integrovaných obvodů je blokováno keramickými kondenzátory C101, C102, C103.

IRC snímač může být volitelně napájen přímo z převodníku, a to spojením propojek JP101. K přivedení pevného napětí na negativní vstupy při použití s jednopólovým IRC snímačem slouží propojky JP102.

Na konektoru JP103 je k dipozici chybový signál každého vstupního kanálu (A, B, Z) samostatně. Chybový stav na kterémkoliv vstupním kanálu indikuje LED101.

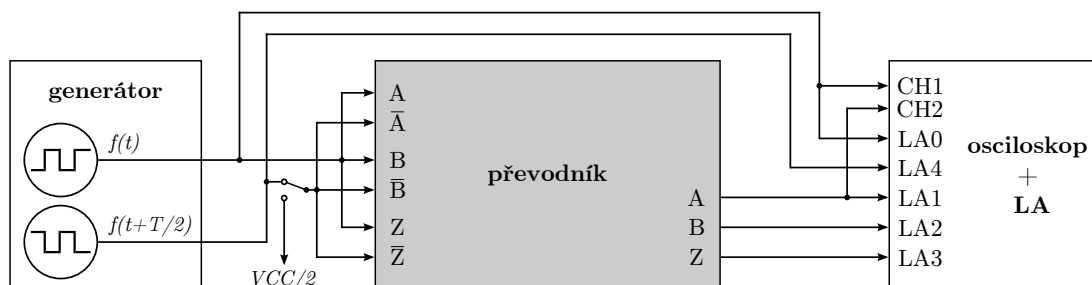
Výstupní část je přizpůsobena použité prototypové FPGA desce, která má na použitém konektoru vyvedeno jen 7 vstupních signálů. Výstup každého bloku obsahuje 3 užitečné signály A, B, Z. Jediný výstupní chybový signál ERR je sdružen z obou vstupních bloků. Pokud využíváme jen jeden vstupní blok, je možné chybový signál z druhého bloku potlačit rozpojením příslušné části propojky JP1. Napájecí napětí výstupní části indikuje LED3.

### 9.3 Dosažené vlastnosti a parametry

- **Napájení**
  - 5 V DC, 200 mA max.
- **Vstupní část**
  - $2 \times 3$  diferenciální vstupy, rozhodovací úroveň mezi  $-200$  mV a  $+200$  mV,
  - kompatibilní se standardem RS-422,
  - maximální vstupní napětí  $\pm 25$  V proti zemi,
  - detekce chybového stavu při rozdílu napětí diferenciálních vstupů  $< 275$  mV, nebo při souhlasném napětí mimo povolený rozsah  $-10$  V až  $+13,2$  V.
- **Výstupní část**
  - napájení 3,3 V DC, 50 mA max, napěťové úrovně CMOS 3,3 V.
- **Izolační vlastnosti**
  - vzájemné galvanické oddělení napájení, vstupní a výstupní části,
  - napájení – vstup: 1 kV DC, vstup – výstup: 560 V DC.
- **Časové parametry**
  - zkreslení délky pulzu – *pulse skew*  $< 40$  ns,
  - zpoždění průchodu signálu – *propagation delay*  $< 120$  ns.

### 9.4 Naměřené parametry

Pro kontrolu parametrů uvedených v předcházející kapitole, které vychází z katalogových údajů použitých součástek, bylo na realizovaném kusu převodníku provedeno měření.



Obrázek 44: Měření parametrů – zapojení.

Zapojení použité pro měření je znázorněno na obrázku 44. Všechny tři vstupy jedné sekce převodníku byly připojeny ke generátoru signálu. Signál z generátoru a výstupy převodníku byly sledovány digitálním osciloskopem s logickým analyzátozem.

Pro testování byly použity tyto budící signály:

- (a) jednopólový signál, obdélníkový průběh 1:1, napětí 0 V – 5 V, 50 kHz, negativní vstupy připojeny na napětí 2,5 V,
- (b) diferenciální signál, obdélníkový průběh 1:1, napětí  $\pm 1,5$  V, 2 MHz,
- (c) diferenciální signál, sinusový průběh, napětí  $\pm 1,5$  V, 2 MHz.

Naměřené průběhy signálů, ze kterých je patrné časové zkreslení průchodu signálu převodníkem, jsou na obrázcích 45, 46 a 47.

### Naměřené hodnoty

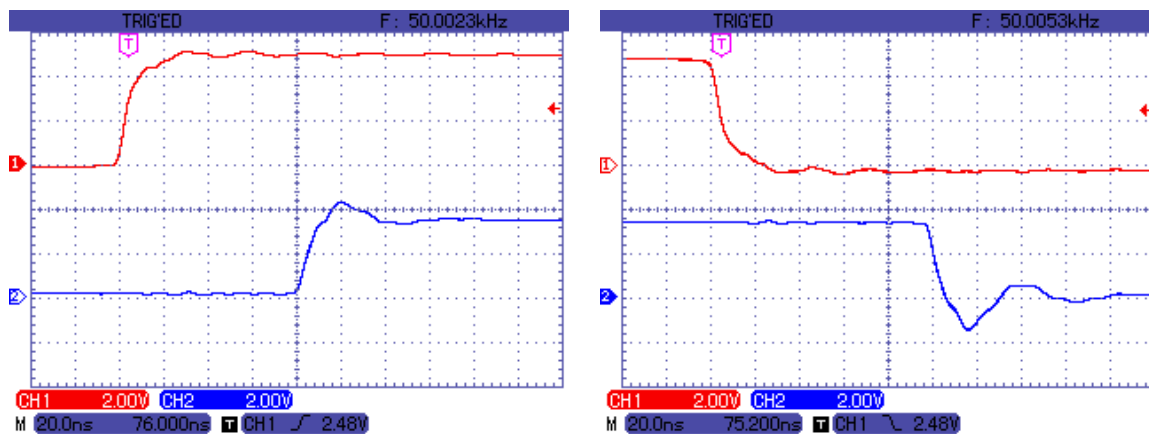
- čas trvání vzestupné hrany na výstupu,  $t_{LH} = 6$  ns,
- čas trvání sestupné hrany na výstupu,  $t_{HL} = 6$  ns,
- zpoždění vzestupné hrany vstup – výstup,  $t_{PD,LH} = 84$  ns,
- zpoždění sestupné hrany vstup – výstup,  $t_{PD,HL} = 102$  ns,
- zkreslení délky pulzu (*pulse skew*),  $|t_{PD,HL} - t_{PD,LH}| = 18$  ns.

Vzhledem k tomu, že naměřené hodnoty jsou na hranici možností použitých měřících přístrojů, je třeba je brát jako přibližné. Skutečné parametry budou o něco lepší.

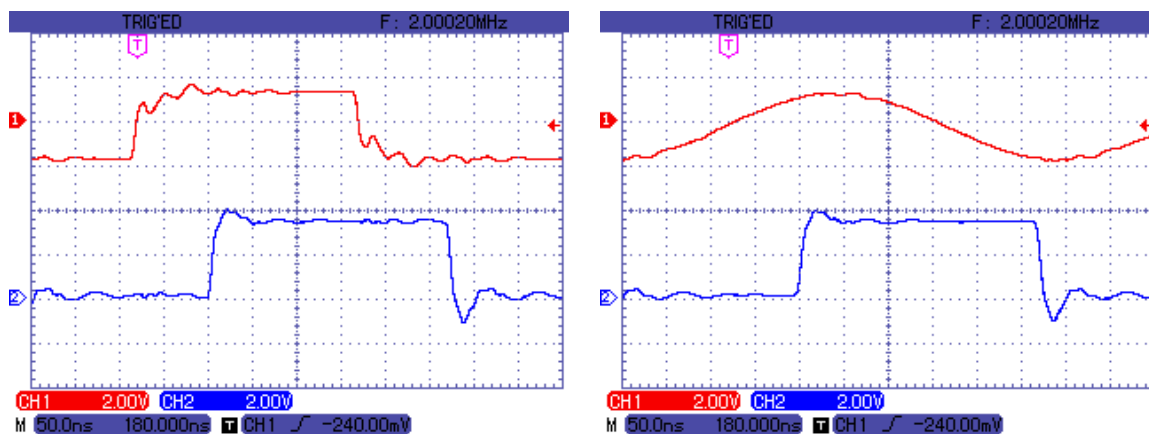
Hodnoty naměřených parametrů odpovídají hodnotám teoreticky odvozeným z katalogových parametrů.

### Použité měřicí přístroje

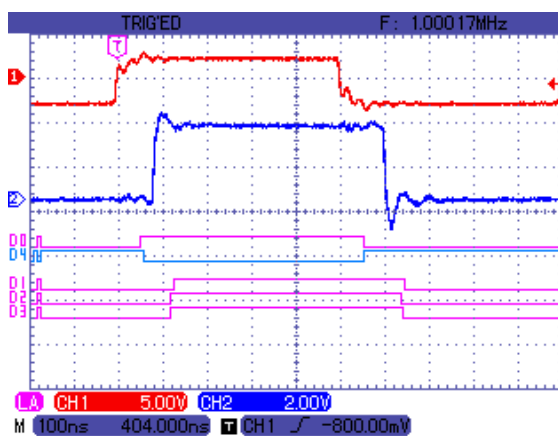
- dvoukanálový DDS generátor – SIGLENT SDG1025; 25 MHz, 125 MSa/s,
- digitální osciloskop a logický analyzátoz – UNI-T UTD4082C; 80 MHz, 2 GS/s.



Obrázek 45: Naměřené průběhy. Vstupní signál (a), vzestupná a sestupná hrana.



Obrázek 46: Naměřené průběhy. Vstupní signály (b) a (c).



Obrázek 47: Naměřené průběhy. Vstupní signál (b), srovnání s logickým analyzátořem.

## 10 Závěr

Diplomová práce přináší řešení problému měření rychlosti a zrychlení rotačního pohybu na základě signálu z inkrementálního snímače, tedy na základě časově nepravidelných měření polohy zatížených šumem. Přestože se s tímto problémem často setkáváme v průmyslové praxi, běžně používané metody neposkytují uspokojivou kvalitu výsledků.

Stěžejní částí a největším přínosem této práce je návrh komplexního řetězce algoritmů zpracování signálu pro odhad rychlosti a zrychlení, přičemž je kladen důraz na praktickou realizovatelnost. Navržené řešení bylo implementováno a detailně otestováno na simulovaných i reálných signálech. Kvalita odhadů rychlosti a zrychlení dalece překonává známé metody, a to i v podmínkách silného zatížení šumem, kdy se navíc projevuje velmi dobrá laditelnost frekvenční charakteristiky filtru jediným uživatelsky volitelným parametrem.

Princip navrženého řetězce zpracování signálu je možné popsat těmito body:

- Přesné měření časových okamžiků pulzů z inkrementálního snímače s přeskokováním při příliš velké rychlosti (*time stamping with pulse skip*) – předzpracování signálu.
- Odhad skutečné polohy a její první a druhé derivace pomocí Kalmanova filtru jako rekonstruktoru stavu modelu zdroje signálu. Ve skutečnosti neznámý model je nahrazen modelem v podobě řetězce tří sériově spojených integrátorů, kdy předpokládáme, že změny polohy v čase lze lokálně aproximovat polynomem třetího řádu.

Problém vzorkování neekvidistantního v čase je vyřešen spojitým návrhem Kalmanova filtru a odvozením diskretizačních vztahů v podobě vhodné pro rychlý numerický výpočet a nalezením vhodných numerických algoritmů.

- Predikce odhadů v čase vstupních pulzů k požadovaným ekvidistantním časům.

Metoda předzpracování signálu ze snímače vyžaduje použití speciálního hardware, který lze ale snadno realizovat na hradlovém poli – programovatelném obvodu FPGA.

Algoritmus navrženého filtru je výpočetně dosti náročný. Byla vytvořena implementace v jazyce C++ a na základě testů byla dále optimalizována, takže na průměrně výkonném průmyslovém počítači (procesor Intel Atom) zabere výpočet filtru v nejnepříznivějším případě kolem 5 % procesorového času.

Dosažené výsledky ukazují na velký potenciál praktického využití. V práci by bylo vhodné dále pokračovat, zejména na integraci výpočetního algoritmu do systému reálného času Linux/Xenomai, a na dokončení implementace kódu pro hradlové pole v jazyce VHDL.

## Reference

### Články:

- [1] OHMAE, T. et al. A Microprocessor-Controlled High-Accuracy Wide-Range Speed Regulator for Motor Drives. *IEEE Transactions on Industrial Electronics*, Vol. IE-29, No. 3, August 1982, s. 207–211.
- [2] GLAD, T., Ljung, L. Velocity Estimation from Irregular, Noisy Position Measurements. *IFAC 9th World Congress*, Budapest, 1984, s. 1069–1073.
- [3] BÉLANGER, Pierre R. Estimation of Angular Velocity and Acceleration from Shaft Encoder Measurements. *IEEE International Conference on Robotics and Automation*, Nice, 1992, s. 585–592.
- [4] PROKIN, M. Extremely Wide-Range Speed Measurement Using a Double-Buffered Method. *IEEE Transactions on Industrial Electronics*, Vol. IE-41, No. 5, October 1994, s. 550–559.
- [5] KAVANAGH, R. C. Improved Digital Tachometer With Reduced Sensitivity to Sensor Nonideality. *IEEE Transactions on Industrial Electronics*, Vol. 47, No. 4, August 2000, s. 890–897.
- [6] TSUJI, T. et al. A Wide-Range Velocity Measurement Method for Motion Control. *IEEE Transactions on Industrial Electronics*, Vol. 56, No. 2, February 2009, s. 510–519.
- [7] SCHLEGEL, M., Večerek, O. Odhad derivací signálu s šumem Kalmanovým filtrem. *Process Control 2002: the 5th international scientific technical conference*, Pardubice, University of Pardubice, 2002. ISBN 80–7149–452–1.
- [8] AGILENT TECHNOLOGIES. *Incremental Encoder Errors: Causes and Methods to Reduce Them, Application Brief M-109*. 1999.  
Dostupné z: <http://www.datasheetarchive.com/indexer.php?file=DSA0014646.pdf>
- [9] LEONARD, Mark. Push-Pull Optical Detector Integrated Circuit. *IEEE Journal of Solid-State Circuits*, 1980, Vol. SC-15, no. 6, December, s. 1087–1089.
- [10] MOLER, C., Van Loan, C. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. *SIAM Review*, 2003, Vol. 45, No. 1, s. 3–49.



- [11] HIGHAM, N. J. The Scaling and Squaring Method for the Matrix Exponential Revisited. *SIAM Journal on Matrix Analysis and Applications*, 2005, Vol. 26, No. 4, s. 1179–1193.
- [12] BLÁHA, L., Schlegel, M., Mošna, J.: Optimal Control of Chain of Integrators with Constraints. *Proceedings of the 17th International Conference on Process Control 09*, Štrbské Pleso, Slovakia, 2009, s. 51–56.

### **Katalogové listy a aplikační poznámky:**

- [13] AVAGO TECHNOLOGIES. *Encoder Questions and Answers, Application Brief M-101*. 2010.  
Dostupné z: <http://www.avagotech.com/docs/5988-5062EN>
- [14] TEXAS INSTRUMENTS. *TMS320x2833x, 2823x Enhanced Quadrature Encoder Pulse (eQEP) Module, Reference Guide*. 2008.  
Dostupné z: <http://www.ti.com/lit/ug/sprug05a/sprug05a.pdf>
- [15] MICROCHIP. *dsPIC33F, Quadrature Encoder Interface (QEI), Data Sheet*. 2007.  
Dostupné z: <http://ww1.microchip.com/downloads/en/DeviceDoc/70208A.pdf>
- [16] AVAGO TECHNOLOGIES. *HEDM-55xx/560x & HEDS-55xx/56xx, Quick Assembly Two and Three Channel Optical Encoders, Data Sheet*. 2012.  
Dostupné z: <http://www.avagotech.com/docs/AV02-1046EN>
- [17] MAXON MOTOR. *Encoder HEDL 5540, Data Sheet*. 2012.  
Dostupné z: [http://www.maxonmotor.com/medias/sys\\_\\_master/8801131036702/12\\_278\\_279\\_280\\_en.pdf](http://www.maxonmotor.com/medias/sys__master/8801131036702/12_278_279_280_en.pdf)
- [18] PEPPERL+FUCHS. *RHI58N, General Purpose Inc. Encoder, Data Sheet*. 2011.  
Dostupné z:  
[http://files.pepperl-fuchs.com/selector\\_files/navi/productInfo/edb/t10714\\_eng.pdf](http://files.pepperl-fuchs.com/selector_files/navi/productInfo/edb/t10714_eng.pdf)
- [19] AUTONICS. *E40 Series Incremental Rotary Encoder, Data Sheet*.  
Dostupné z: [http://www.autonics.com/upload/data/E40\\_eng\\_110822.pdf](http://www.autonics.com/upload/data/E40_eng_110822.pdf)
- [20] ENCODER PRODUCTS COMPANY. *Model 755A, Data Sheet*.  
Dostupné z: <http://www.encoder.com/literature/datasheet-755a-shaft.pdf>

- [21] MAXIM INTEGRATED PRODUCTS. *MAX3097E/MAX3098E Datasheet*. 2000.  
Dostupné z:  
<http://datasheets.maximintegrated.com/en/ds/MAX3097E-MAX3098E.pdf>
- [22] AVAGO TECHNOLOGIES. *ACPL-071L and ACPL-074L Datasheet*. 2011.  
Dostupné z: <http://www.avagotech.com/docs/AV02-0963EN>
- [23] MURATA POWER SOLUTIONS. *MER1 Series DC/DC Converters*. 2010.  
Dostupné z: [http://www.murata-ps.com/data/power/ncl/kdc\\_mer1.pdf](http://www.murata-ps.com/data/power/ncl/kdc_mer1.pdf)
- [24] ON SEMICONDUCTOR. *BSS138LT1 MOSFET Datasheet*. 2011.  
Dostupné z: [http://www.onsemi.com/pub\\_link/Collateral/BSS138LT1-D.PDF](http://www.onsemi.com/pub_link/Collateral/BSS138LT1-D.PDF)

#### Ostatní:

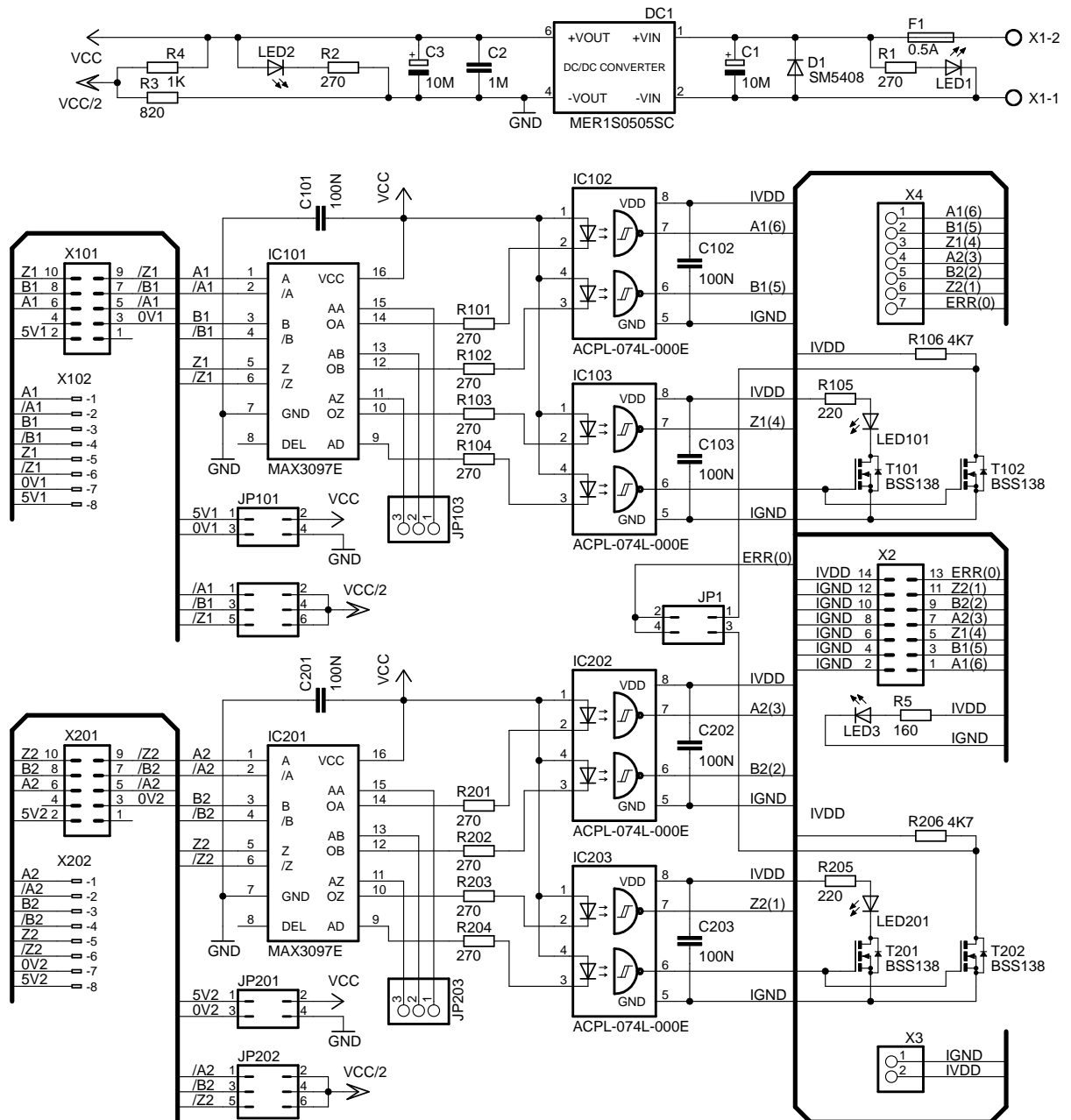
- [25] THE MATHWORKS. *Continuous-Discrete Conversion Methods* [online]  
[cit. 1.4.2013]. Dostupné z:  
<http://mathworks.com/help/ident/ug/continuous-discrete-conversion-methods.html>
- [26] THE MATHWORKS. *Matrix exponential – expm* [online] [cit. 1.4.2013].  
Dostupné z: <http://www.mathworks.com/help/matlab/ref/expm.html>
- [27] JÁGER, A. *Návrh generátoru časově optimální trajektorie pro řetězec tří integrátorů s intervalovým omezením na řízení a souřadnice stavu*. 2009. Bakalářská práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky.
- [28] PRATA, Stephen. *Mistrovství v C++*. 3. aktualizované vydání. Computer Press, 2007. ISBN 978-80-251-1749-1.
- [29] NICTA. *Armadillo: C++ linear algebra library* [online] [cit. 1.4.2013].  
Dostupné z: <http://arma.sourceforge.net/>
- [30] *Xenomai: Real-Time Framework for Linux* [online] [cit. 1.4.2013].  
Dostupné z: <http://www.xenomai.org/>
- [31] REX Controls. *Řídicí systém REX* [online] [cit. 1.4.2013].  
Dostupné z: <http://www.rexcontrols.cz/rex>
- [32] PINKER, J., Poupa, M. *Číslicové systémy a jazyk VHDL*. 1. vydání. BEN, 2006. ISBN 80-7300-198-5.

- [33] Král, J. *Řešené příklady ve VHDL*. 1. vydání. BEN, 2010. ISBN 978-80-7300-257-2.
- [34] TERASIC. *Altera DE2-115 Development and Education Board* [online] [cit. 1.4.2013].  
Dostupné z: <http://de2-115.terasic.com>
- [35] ALTERA. *Triple-Speed Ethernet MegaCore Function, User Guide*. 2013.  
Dostupné z: [http://www.altera.com/literature/ug/ug\\_ethernet.pdf](http://www.altera.com/literature/ug/ug_ethernet.pdf)

## 11 Příloha A – Datový nosič CD

- Text práce ve formátu PDF:
  - *SETKA\_Vlastimil\_Diplomova\_prace\_2013.pdf*
- Simulační prostředí - balík funkcí a skriptů pro MATLAB:
  - *IRC\_Simulations\_MATLAB.zip*
- Zdrojové kódy algoritmu filtru a testovacích programů v C++:
  - *IRC\_FilterCpp.tgz*
- Výběr z použité literatury ve formátu PDF:
  - *Reference/* (složka)

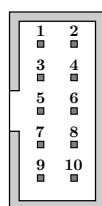
## 12 Příloha B – Schéma a DPS převodníku



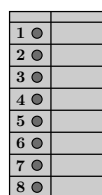
Obrázek B.1: Schéma zapojení.

Pozice	Hodnota	Pouzdro	Počet	Popis
C1, C3	10 $\mu$ F / 16 V	SMD 1206	2	kondenzátor tantalový
C2	1 $\mu$ F / 25 V	SMD 1206	1	kondenzátor keramický Y5V
C101 - C103	100 nF / 16 V	SMD 1206	6	kondenzátor keramický X7R
R1, R2, R101 - R104	270R	SMD 1206	10	rezistor 5%
R3	820R	SMD 1206	1	rezistor 5%
R4	1K	SMD 1206	1	rezistor 5%
R5	160R	SMD 1206	1	rezistor 5%
R105	220R	SMD 1206	2	rezistor 5%
R106	4K7	SMD 1206	2	rezistor 5%
D1	SM5408-TAP	DO213AB	1	dioda
LED1 - LED3	KPT-3216 GREEN	SMD 1206	3	LED zelená
LED101	KPT-3216 RED	SMD 1206	2	LED červená
T101, T102	BSS138	SOT-23	4	tranzistor N-MOSFET
IC101	MAX3097E	SOIC-16	2	RS-422 line receiver
IC102, IC103	ACPL-074L-000E	SOIC-8	4	dual high speed optocoupler
DC1	MER1S0505SC	MER1	1	DC/DC měnič 5 V, 1 W
F1	PSWRX.050	-	1	vratná pojistka 0.5 A
JP1, JP101	PLD-04S	-	3	jumper lišta 2×2
JP102	PLD-06S	-	2	jumper lišta 2×3
JP103	PLS-03S	-	2	jumper lišta 1×3
X1	MC000120	-	1	svorkovnice napájecí 2×3.81 mm
X2	WSL14G	-	1	konektor 2×7 pinů 2.54 mm
X3	PS-02S	-	1	zásuvková lišta 1×2 2.54 mm
X4	PS-07S	-	1	zásuvková lišta 1×7 2.54 mm
X101	WSL10G	-	2	konektor 2×5 pinů 2.54 mm
X102	WAGO 250-408	-	2	svorkovnice

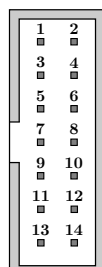
Tabulka B.1: Seznam součástek. Pozice 2xx nejsou uvedeny, jsou shodné s 1xx.

**X101, X201**

1	N.C.	6	A
2	VCC (+5 V)	7	$\bar{B}$
3	GND	8	B
4	N.C.	9	$\bar{Z}$
5	$\bar{A}$	10	Z

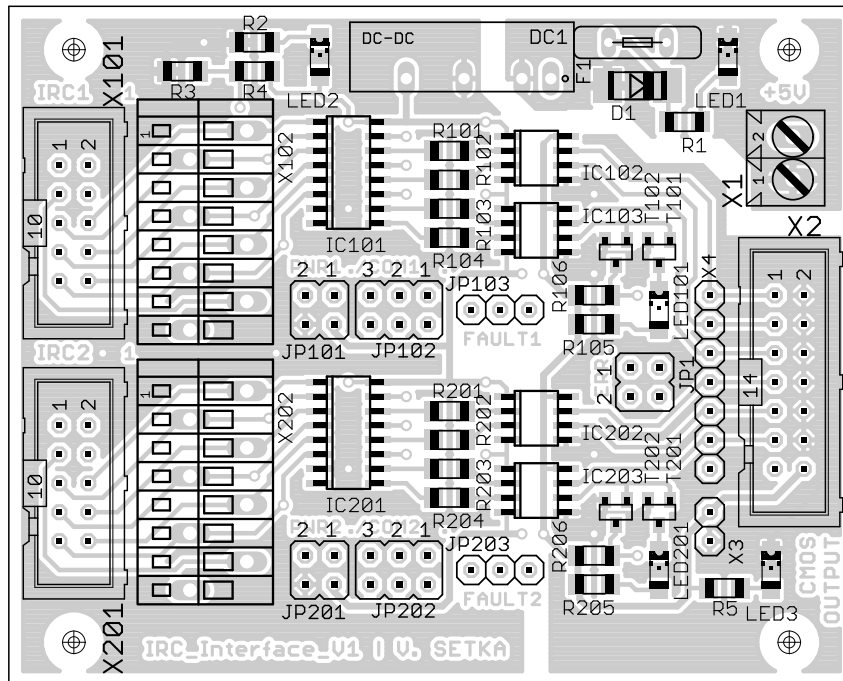
**X102, X202**

1	A	5	Z
2	$\bar{A}$	6	$\bar{Z}$
3	B	7	GND
4	$\bar{B}$	8	VCC (+5 V)

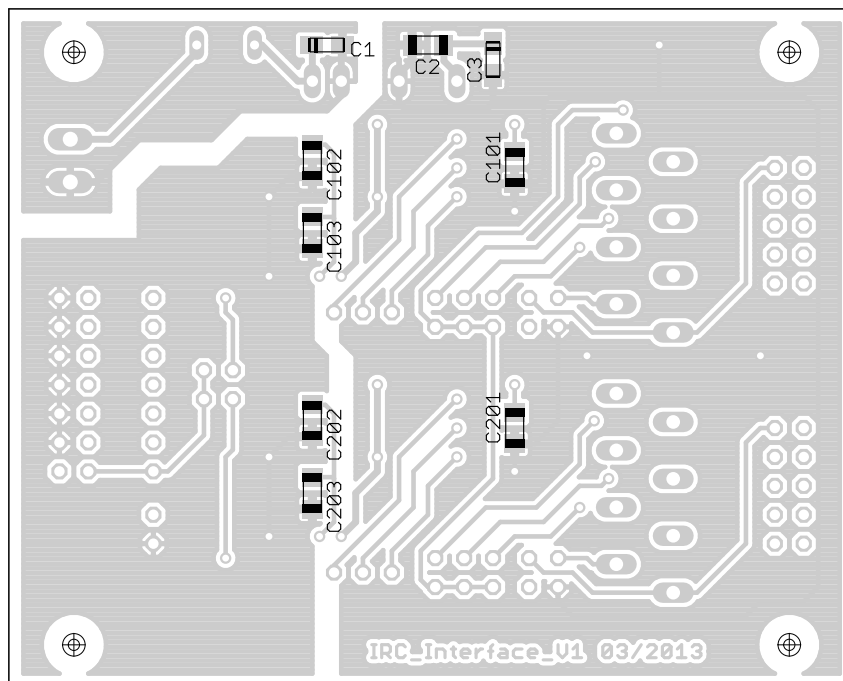
**X2**

1	A1	8	IGND
2	IGND	9	B2
3	B1	10	IGND
4	IGND	11	Z2
5	Z1	12	IGND
6	IGND	13	ERR
7	A2	14	IVDD (3.3 V)

Obrázek B.2: Popis konektorů (označení pinů při pohledu na desku shora).  
X101, X201, X102, X202 – připojení IRC snímačů; X2 – CMOS výstup.

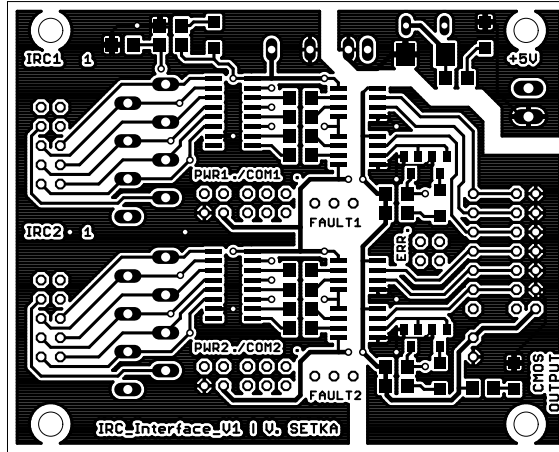


Obrázek B.3: Osazení desky – horní strana.

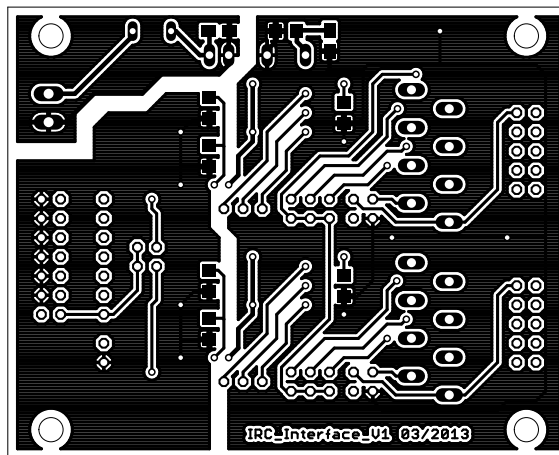


Obrázek B.4: Osazení desky – spodní strana.





Obrázek B.5: Motiv desky – horní strana (M 1:1, 74,3 mm × 59,7 mm).



Obrázek B.6: Motiv desky – spodní strana (M 1:1, 74,3 mm × 59,7 mm).