VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

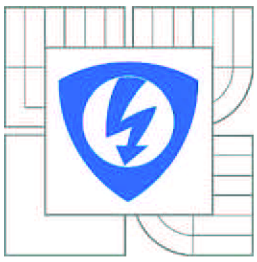
MODELOVÁNÍ DYNAMIKY ČÁSTI TISKAŘSKÉHO STROJE
MODELING OF DYNAMICS OF THE PART OF A PRINTING MACHINE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Jiří Junek

VEDOUCÍ PRÁCE
SUPERVISOR
prof. Ing. Pavel Václavek, Ph.D.

BRNO 2016



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

Magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Jiří Junek ***ID:*** 146031
Ročník: 2 ***Akademický rok:*** 2015/2016

NÁZEV TÉMATU:

Modelování dynamiky částí tiskařského stroje

POKYNY PRO VYPRACOVÁNÍ:
Cílem práce je vytvoření modelu části tiskařského stroje a příprava SW nástrojů v prostředí Matlab-Simulink pro identifikaci parametrů modelu na základě dat změřených na reálném tiskařském stroji. Práce se sestává zejména z následujících úkolů:

1.Vytvořte model odvíjecí/navíjecí části tiskařského stroje v programu Matlab Simulink-SimScape
2.Proveďte identifikaci parametrů modelu na základě změřených dat na reálném tiskařském stroji.
3.Vytvořte uživatelské rozhraní pro ovládání a nastavení modelu s vykreslováním zvolených průběhů.
4.Navrhněte a implementujte automatický algoritmus identifikace parametrů potiskového materiálu po provedení měření na stroji

DOPORUČENÁ LITERATURA:

Jafarboland, M., Zadehbagheri, M. Modeling of Belt-Pulley and Flexible CouplingEffects on Submarine Driven System ElectricalMotors, Journal of Power Electronics, Vol. 11, No. 3, May 2011

Termín zadání: 8. 2. 2016 ***Termín odevzdání:*** 16. 5. 2016

Vedoucí práce: prof. Ing. Pavel Václavek, Ph.D.

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato diplomová práce se zabývá modelováním dynamiky tiskařského stroje firmy SOMA Engineering v toolboxu SimScape simulačního programu Matlab/Simulink. Jsou zde popsány vlastnosti, postup, důležité principy a zákonitosti při modelování v tomto toolboxu. První část práce se zabývá vytvořením modelů, které obsahují jak mechanické, tak i elektrické a řídící části. V druhé části práce je vysvětlen postup vytvoření uživatelského rozhraní k ovládání modelů a vytvoření samostatně spustitelných (tzv. stand-alone) aplikací, které běží nezávisle na prostředí Matlab/Simulink.. V poslední části je vysvětlen princip identifikace parametrů materiálu z naměřených dat, pomocí optimalizačního algoritmu Nelder-Mead. Vytvořené modely a identifikační algoritmus jsou ovládány uživatelským rozhraním.

Klíčová slova

Matlab, SimScape, modelování, dynamika, akauzální, odvíjení, navíjení, stand-alone, identifikace, uživatelské rozhraní, kompilátor, Nelder-Mead

Abstract

This thesis deals with modeling dynamics of printing machine, made by SOMA Engineering, in toolbox SimScape of simulation program Simulink/Matlab. There are descibed properties, progress, important principles and laws of modeling in this toolbox. The first part is focused on creating models, which consist as mechanical as eletrical and control parts. In the second part is explained how to create user interface to control models and creating standalone application, executable without installation of Matlab or Simulink. In last part is explained principle of identification parameters of material from the measured data, using an optimalization algorithm Nelder-Mead. Created models and identification algorithm are controlled via user interface.

Keywords

Matlab, SimScape, modeling, dynamics, acausal, unwind, rewind, standalone, identification, user interface, GUI, compiler

Bibliografická citace:

JUNEK, J. *Modelování dynamiky částí tiskařkého stroje*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 80 s. Vedoucí diplomové práce prof. Ing. Pavel Václavek, Ph.D.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Modelování dynamiky částí tiskařského stroje“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: …… ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce prof. Ing. Pavlu Václavkovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce, také za jeho vstřícnost a ochotu být kdykoliv nápomocen.

V Brně dne: ....

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tato diplomová práce se zabývá modelováním reálného tiskařského stroje Optima firmy SOMA Engineering Lanškroun. Tato firma vyrábí již od roku 1992 speciální jednoúčelová zařízení, stroje na zpracování a potisk materiálu, zejména pak flexotiskové stroje, podélné řezačky, laminátory atd. Pro modelování fyzických částí tohoto stroje byl zvolen toolbox SimScape spustitelný v simulačním nástroji Simulink, matematického nástroje Matlab od firmy Mathworks. Tento toolbox byl zvolen z důvodu velké komplexnosti simulačních bloků, pracujících s reálnými fyzikálními veličinami (točivý moment, úhlová rychlost, síly atd). Bloky představují fyzikální celky jako například hřídele, brzdy, převodovky, nebo i motory (AC/DC, indukční a jiné). Do určité míry je možné je propojovat i se signály samotného Simulinku a to použitím převodníků *PS-S Converter*, *S-PS Converter*. Samotné propojení prvků z rozdílných knihoven SimScape je obtížné, v některých případech až nemožné; například není možné vzájemně propojit bloky z toolboxů SimElectronics a SimDriveline.

Vytvořené modely odvíjecí a navíjecí část flexotiskového stroje OPTIMA mají sloužit firmě SOMA jako podklad a nástroj pro odzkoušení různých parametrů motorů (výkon motorů, rychlostní rampy aj.), nastavení parametrů regulátor tahu a otáček motorů, nebo při zkoušení vlastností nového typu nebo rozměrů potiskového materiálu. V tuto chvíli probíhá odzkoušení nebo dolaďování těchto parametrů testováním na již vyrobeném stroji, nebo se využívá zkušeností programátora z posledního vytvořeného typu stroje.

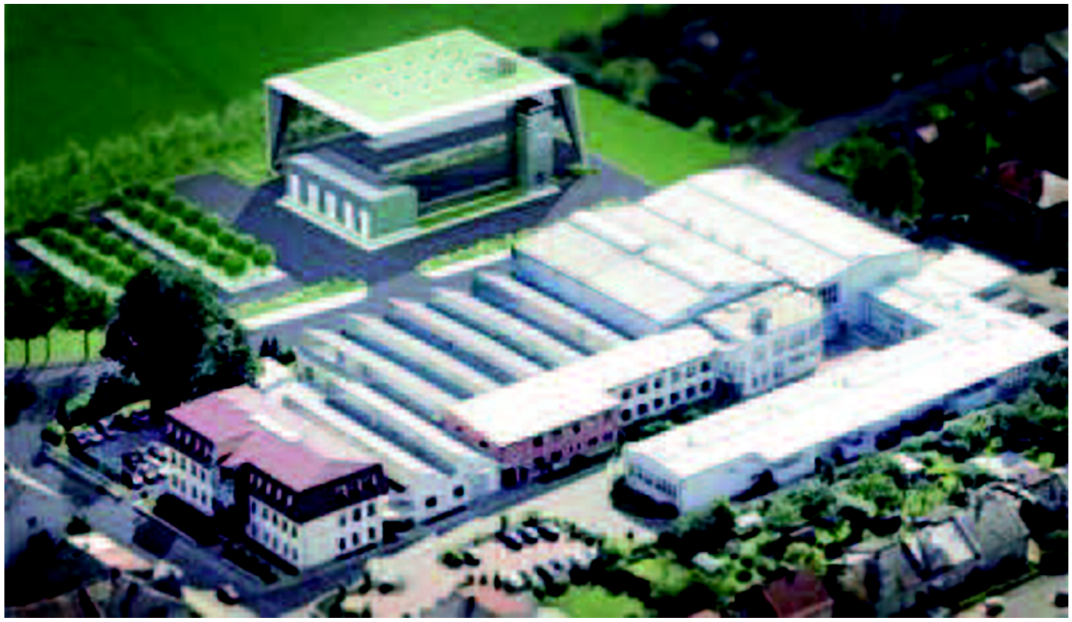
Další částí této diplomové práce je vytvoření samostatně spustitelných aplikací z vytvořených modelů a uživatelských rozhraní, které lze spouštět bez potřeby instalace programovacího prostředku Matlab, resp. Simulink. Pro generování kódu (do jazyka C nebo C++) ze Simulinku je potřeba Simulik Coder, pro generování kódu z uživatelských rozhraní Matlab Compiler. Dále je vysvětleno propojení vytvořených aplikací a modelů, příkazy potřebné pro spuštění, ukončení simulace, výměnu dat apod.

Cílem této práce je vytvořit odpovídající model částí odvíjení a navíjení tiskařského stroje, jehož uživatelské rozhraní zajistí koncovému uživateli intuitivní nastavování a ovládání modelů s vykreslováním zvolených průběhů.

1 SOMA

Společnost SOMA spol. s r.o. vznikla roku 1992 privatizací samostatné divize Tesly Lanškroun. Firma se zabývá vývojem, výrobou a prodejem strojů. Většinu nabídky tvoří vlastní výrobky. Jedná se o stroje k potisku, řezání a laminování obalových materiálů. V současné době má firma SOMA více než 200 zaměstnanců.

Výrobní program je od počátků devadesátých let zaměřen na vývoj a výrobu vysoce kvalitních speciálních jednoúčelových zařízení, strojů na zpracování a potisk materiálu, zejména pak flexotiskových strojů, podélných řezaček, montážních stolic, laminátorů, příčných řezaček a vysekávacích automatů. Společnost prochází trvalou expanzí, byly vybudovány nové produkční, administrativní a technologické budovy. Inovace a moderní design jsou klíčovými faktory úspěchu firmy.



Obr. 1-1 : Sídlo firmy Soma Eng. [1]

Společnost SOMA je silně exportně zaměřenou společností s vlastním know-how. Své stroje vyváží zejména do zemí střední a východní Evropy, postupně proniká na americký, asijský i africký trh. Vysoce kvalitní stroje jsou zárukou dlouhodobého úspěchu. Většina součástek použitých na výrobu strojů jsou vyráběny v podniku SOMA s důrazem na precizní výrobu dílů.[1]

1.1 Flexotiskový stroj Optima

Soma Flex OPTIMA je flexotiskový stroj bez ozubených kol s centrálním protitlakým válcem určený pro kontinuální potisk flexibilních obalových materiálů, papíru nebo laminátů. Stroj je vybaven systémy pro rychlou výměnu zakázky a disponuje robustní a tuhou konstrukcí. Systém masivních odlitků vzájemně propojených tuhými příčníky zabezpečuje optimální dynamické vlastnosti všech jednotek i stroje jako celku.

Obr. 1-2 : Vzhled stroje Optima [3]

Hlavní parametry

Počet barevníků	8
Šíře tisku	620/820 mm
Délka tisku	240/260 – 600 mm
Rychlost	300 m/min
průměr role	800/1000 mm
provedení	hřídelové...
technologie sleevů	ano

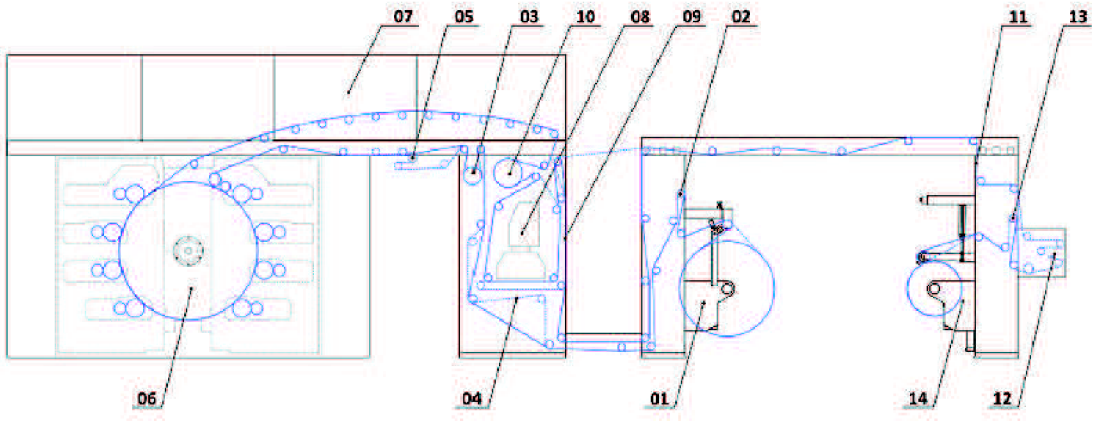
Obr. 1-3 : Hlavní parametry stroje Optima **[3]**

1.1.1 Technická specifikace

Tabulka 1-1 – Detailní parametry stroje Optima

Šíře materiálu	850 mm
Min. šíře materiálu	300 mm
Min. délka tisku	240/260 mm
Tah materiálu	10/40 – 400 N
Průměr dutinek	76,2 / 152,4 mm
Váha role	750/1000 kg
Lihové barvy	Ano
UV barvy	Ano
Hmotnost stroje	27 000 kg

1.1.2 Názorné schéma stroje



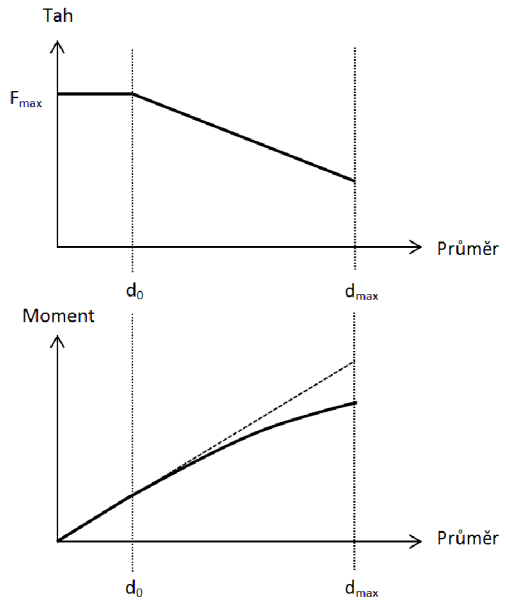
Obr. 1-4 : Názorné schéma stroje [3]

Tisk na tomto stroji probíhá kolem centrálního přítlačného válce (06), kolem kterého obíhá potiskový materiál (modrá křivka) a z druhé strany je prováděn potisk (zde je osm válců – barevníků), dále tento materiál pokračuje sušícím tunelem (07). Poté probíhají finální operace, kontrola soutisku (08) a jiné. V této diplomové práci se budeme hlavně věnovat odvíjecí a navíjecí části stroje (01 a 14).

Všechny důležité části, které se podílejí na kvalitním tisku, jsou konstruovány s velkým důrazem na dynamické vlastnosti. Rám tiskové jednotky je vyroben z litiny a je tvořen dvěma monoblokovými bočnicemi s vloženou deskou jistící centrální válec. Tloušťka bočnic se pohybuje od 25 mm do 80 mm, tloušťka obvodového žebra je 130 mm. Barevníky s válci tiskových a rastrových sleevů jsou ergonomicky řešeny s důrazem na robustnost, přesnost, spolehlivost a jednoruční manipulaci s uzavíracími supporty.

1.2 Naviječí/Odviječí jednotka

Jelikož se tato práce zabývá modelováním naviječí a odviječí části tohoto stroje, popíši zde hlavně tyto dvě části, jejich konstrukci, funkci a účel. V zásadě jsou tyto části určeny k jednoduchému účelu; v případě odviječí jednotky k tomu, aby se potiskový materiál odvíjel rychlostí, kterou určuje hlavní motor, a přitom se materiál neroztrhl, nebo nebyl málo natáhnut. Nedostatečným napnutím materiálu se provede špatný potisk a znehodnotí se tím potiskový materiál. Proto je potřeba udržovat určitý tah materiálu. Tento tah je určen třemi základními parametry: typ materiálu/materiál, tloušťka materiálu a šířka materiálu. Výsledná hodnota tahu se po zjištění těchto materiálových parametrů dosadí do převodní tabulky (dodávané od fy Soma) nebo je určena ze zkušeností operátora. V případě naviječí jednotky je její účel velice podobný, avšak o trochu složitější; naviječí jednotka také musí „kopírovat“ rychlost hlavního motoru, zde se ale regulovaný tah lineárně zmenšuje se zvyšujícím se průměrem naviječé role (viz obr. 1.5), důvodem je zamezení vzniku vzduchových mezery mezi vrstvami materiálu aby byl materiál perfektně navinut.



Obr. 1-5 : Změna tahu při navíjení [4]

Základní parametry navíjecí a odvíjecí části:

- max. průměr rolí 1000 mm
- bezhřídelové upínání rolí pomocí kuželů
- zastavení při dosažení min./max. průměru odvíjené/navíjené role
- přesná regulace tahu motorem a systémem tanečnic
- stranová regulace posuvem odvíjené role
- čidlo pro identifikaci slepeného materiálu

- systém automatického snižování tahu při navíjení podle průměru navíjené role
- pogumovaný přítlačný válec na odklopných ramenech – automatické snižování přítlaku podle průměru navíjeného materiálu
- snímání elektrostatického náboje z povrchu navíjeného materiálu

Jak jsem již zmínil, tyto části stroje jsou určeny při prvním pohledu k jasnému a jednoduchému účelu, avšak pokud jako celek soustavy budeme brát motor, řídící elektroniku motoru, regulátory tahu, tanečnici se systémem válců a roli jako celek, vznikne poměrně rozsáhlý dynamický systém. Role materiálů může mít velice rozdílné parametry; nově vložená role má průměr až 1000 mm a hmotnost až 750 kg, v jiném případě se stroj může rozjíždět s rolí o parametrech 100 mm a hmotností 30 kg. Zde je jasné, že parametry této soustavy se budou velice měnit. Výhodou je, že při nepřerušovaném provozu stroje bude diference parametrů v závislosti na čas celkem malá, naopak při rozjezdu budou počáteční stavy parametrů nezanedbatelné.

Také zde vystupují další parametry, které se budou měnit, a to je např. různá šířka potiskového materiálu, nebo i samotný typ materiálu; v tomto stroji se může tisknou na více materiálů, např. materiály na bázi plastu (LDPE, HPDE, atd.), papír nebo i na vlnitou lepenku. Možnost změny těchto parametrů bude i ve výsledném modelu. Model bude možné aplikovat na jiný výrobní typ stroje Optima – změna výkonu motorů, rychlostní rampy, převodní poměr a mnoho jiných.

2 MODELOVÁNÍ

Modelování je činnost, při kterém se reálný fyzický systém, resp. chování reálného systému, převede pomocí nějakého jazyka/programu na diferenciální rovnice, a ty se poté zpracovávají v počítači. Účelem tohoto procesu je možnost měnit určité parametry nebo vstupy do namodelovaného systému a poté zjistit chování systému při těchto změnách, nebo zjistit chování v určitých kritických situacích. Samotný model je vždy jen přiblížení skutečnosti. K namodelovanému systému vždy přistupujeme tak, že do určité míry odpovídá skutečnému systému. Jak skutečnému systému odpovídá, resp. do jaké míry je tento model věrohodný, je potřeba provést duálními experimenty.

Ziskem správného namodelování je třeba zmenšení nákladů na výrobu stroje – parametry motorů, pevných částí, konstrukce a případné chyby návrhu se mohou zjistit již při samotném vývoji provedením simulace. Tím odpadne potřeba vytvoření nákladného reálného fyzického prototypu. Dalším důvodem může být zlepšení kvality regulace – např. řízení s použití regulátoru s pomocným modelem aj., samozřejmostí je také i zlepšení dalších vlastností nebo parametrů.

Jak jsem již zmínil, model je jen přiblížení skutečnosti a je možné že nastane situace, kdy je reálný systém tak složitý, že model nemusí vždy odpovídat skutečnosti; mohou se zde vyskytovat určité skryté vazby, nezměřitelné nebo nezjistitelné parametry atd.

2.1 Kauzální modelování

Kauzalita znamená příčinnost, vztah mezi příčinou a jejím následkem. V silnějším slova smyslu to znamená, že nic se neděje bez dostatečné příčiny.

Základem je, že v modelovém schématu lze vidět postup spíše matematický, než strukturu reálného systému. Spoje mezi bloky vyjadřují matematické vztahy a těmito bloky jsou převáděny signály. Jednotlivé bloky reálného systému, resp. jednotlivé části chování reálného systému jsou převedeny na matematický popis. Struktura modelu představuje spíše matematickou podstatu problému. Názorným příkladem je například programovací prostředí Simulink; signály (spoje) přenáší jen proměnné mezi bloky, v těchto blocích se provede matematická operace (násobení, integrace, derivace, násobení konstantou atd.). Výsledný fyzikální princip nebo popis se provede spojením těchto bloků. Naopak jiný přístup k tomu má akauzální modelování, viz kapitola níže. [4]

2.2 Akauzální modelování

V moderní době se začíná více aplikovat akauzální zápis modelů, kde jednotlivé bloky (komponenty) nepopisují přímo postup výpočtu, ale spíše definici závislosti fyzikálních veličin, fyzikální celky a principy jako takové. Další změnou oproti kauzálnímu modelování je, že signálem mezi bloky nejsou jednoduché proměnné, ale každý propoj s sebou nese několik proměnných. Tyto proměnné vyjadřují fyzikální veličiny – pokud jde například o propojení dvou bloků reprezentujících rotační pohyb, propoje mezi těmito bloky nesou informaci o úhlové rychlosti, točivém momentu a jsou ovlivněny setrvačnou sílou z jiných bloků. Dále je třeba upozornit, že směr signálu není jednosměrný (jak to je u kauzálního modelu), ale obousměrný – bloky se ovlivňují navzájem. Výsledné schéma především vystihuje fyzikální podstatu modelované reality, než matematický popis jevů.

Tento způsob modelování je výhodný z toho důvodu, že se tvůrce modelu nemusí zajímat do hloubky o všechny fyzikální zákony, které v reálném systému probíhají a matematicky je popisovat, použije ale základní fyzikální princip popisované části. V jednotlivých blocích popisuje přímo rovnice daného fyzikálního elementu, oproti popisování algoritmu výpočtu celkového systému. Propojením těchto komponent propojujeme jednotlivé fyzikální rovnice mezi sebou. Samotný algoritmus výpočtu a skládání diferenciálních rovnic necháváme na simulačním prostředku, lépe řečeno jeho „solveru“ (v českém názvosloví „řešitel“, ale tento výraz není moc obvyklý, proto dále v této práci bude použit termín solver). [5]

Programátor tedy propojením komponent určí strukturu reálného systému a na počítači nechá postavení diferenciálních rovnic a postup výpočtu. Tato vlastnost se velmi hodí při vytváření obsáhlého modelovaného systému, jako v případě této diplomové práce. Zde je použito akauzální modelové prostředí SimScape, které je toolboxem simulačního prostředí Simulink. Akauzální popis mnohem lépe vystihuje fyzikální podstatu modelované reality a simulační modely jsou mnohem přehlednější, a tudíž většinou i méně náchylné k chybám.

2.2.1 Základy akauzálního modelování v SimScape

- Každý zápis zkoumané reality pomocí soustav rovnic by měl mít přesně specifikované proměnné. Pokud je to možné, měl by mít i kromě svého významu také měřitelnou fyzikální jednotku.

- Každý blok vyžaduje určité proměnné, proto zde musí být hlídána určitá „kompatibilita jednotek“, z části je to automatický hlídáno, avšak ne vždy –

například může vzniknout záměna jednotky úhlové rychlosti rad/s za ot/min, tato záměna hlídána není.

- Vhodné je, pro zvětšení přehlednosti modelu, rozdělovat určité fyzické celky nebo fyzikální části do subsystémů.

- I když samotné elementární prvky, resp. jejich popis je triviální, při spojení do větších sítí, nemusí být numerické řešení celku časově nenáročné.

- Propojení bloků mezi knihovnami SimScape (např. SimMechanics a SimEletronics) je až na pár výjimek nemožné, důvodem je použití rozdílných fyzikálních veličiny.

- Možné je vytváření vlastních domén, elementárních prvků nebo fyzikálních celků. [5]

3 SIMULINK – SIMSCAPE

Simulink je nadstavba Matlabu pro simulaci a modelování dynamických systémů, který využívá algoritmy Matlabu pro numerické řešení nelineárních diferenciálních rovnic. Umožňuje snadno a rychle vytvářet modely dynamických soustav ve formě blokových schémat a rovnic. Bloky vyjadřují matematické operace, signálové, lineární/nelineární a jiné operace se signály. [6]

Kromě standardních úloh dovoluje Simulink simulovat i rozsáhlé "stiff" systémy (více v kapitole 5.1 *Stiff systémy*). Pomocí Simulinku a jeho grafického editoru lze vytvářet modely lineárních, nelineárních, v čase diskrétních nebo spojitých systémů. Simulink umožňuje spouštět určité části simulačního schématu na základě výsledku logické podmínky. Možné je vytváření vlastních funkčních bloků a rozšiřovat již tak bohatou knihovnu Simulinku. Hierarchická struktura modelů umožňuje koncipovat i velmi složité systémy do přehledné soustavy subsystémů, prakticky bez omezení počtu bloků. Přenositelnost modelů a schémat mezi různými typy počítačů umožňuje vytvářet rozsáhlé modely, které vyžadují spolupráci většího kolektivu řešitelů na různých úrovních. [6]

3.1 SimScape

SimScape je určený pro modelování reálných fyzikálních systémů. SimScape rozšiřuje Simulink o nástroje pro modelování a simulace tzv. "multi-domain" systémů obsahujících propojení mechanických, elektrických a hydraulických komponent. SimScape využívá nový přístup k modelování fyzikálních systémů, a to akauzální modelování. Hlavním rysem v modelování tímto toolboxem je oproštění popisování reálného systému diferenciálními rovnicemi nebo přenosy; knihovny obsahují základní prvky reálného světa z oblasti mechaniky, elektrotechniky, hydrauliky a nadstavbové knihovny pro modelování termomechanických jevů, magnetických i pneumatických elementů. Z těchto bloků lze sestavit model systému na základě znalosti fyzické struktury pro většinu reálných fyzikálních systémů a jevů, kde vytvořený model odpovídá struktuře systému. Avšak reálný svět je velice rozmanitý a tyto knihovny neobsahují všechny vyskytující se elementy v reálném světě. Možností je vytvoření uživatelských elementů (komponent), což SimScape při dodržení určitých pravidel při psaní podporuje. SimScape obsahuje vlastní jazyk, založený na jazyku Matlabu pro tvorbu uživatelských komponent, rozšiřující knihovnu prvků v dané fyzikální doméně, nebo k tvorbě celých uživatelsky definovaných fyzikálních domén. [7,8]

Bloky se spojují spojnicemi, ale od normálních spojnic používaných v Simulinku se liší tím, že tyto vazby jsou obousměrné, resp. spojnice nemají daný směr

toku – prvky se ovlivňují navzájem, stejně jako v reálném světě. Používané proměnné v těchto blocích reprezentují skutečné fyzikální veličiny (např. rychlost, zrychlení, elektrické napětí a proud, točivý moment atd.). Jak již bylo zmíněno, je snaha oprostit se v modelování od psaní diferenciálních rovnic; po propojení prvků z těchto knihoven a spuštění simulace solver vygeneruje a poskládá diferenciální rovnice. Pomocí speciálních bloků *PS-S/S-PS Coverter* nebo speciálních senzorů převedeme signály z těchto bloků na signály propojitelné se zbytkem bloků Simulinku. Poté tedy můžeme například zobrazit průběh točivého momentu a jiných fyzikálních veličin v bloku *Scope*, nebo s nimi nadále pracovat. [8]

3.1.1 SimMechanics

Knihovna SimScape je určena pro modelování v oblasti, kde se používají reálné fyzikální veličiny jako hmotnost, setrvačnost, síla a translační nebo rotační pohyby. Slouží také k simulaci komplexních zařízení v prostorově souřadnicovém systému. Při použití této knihovny je možné modelovat složité trojrozměrné mechanické soustavy, které se vyskytují ve velkých strojích, automobilech, letadlech apod. Jsou zde obsaženy prvky, které usnadňují modelování modelů složených z tlumičů, kloubů, pružin, šroubů, senzorů a akčních členů, které reprezentují reálný mechanický systém popsaný danými veličinami a silami. Těmito prvky, elementy, lze intuitivně namodelovat mechanický systém a spojovat je s řídícími systémy.

V SimMechanics lze také namodelovat mechanický prvek s určitými rozměry, hmotností, pružností a jiných veličin, jinými slovy mechanickou část z určitého kovu a rozměry. Lze také importovat kompletní modely nakreslené v podporovaných CAD systémech, další funkcí je vestavěná 3D animace umožňující sledovat dynamiku namodelovaných soustav. [7]

Knihovna obsahuje prvky např. :

- Body actuator/senzor

- Planar joint, revolute joint

- Angle driver, linear driver

3.1.2 SimDriveline

Určen pro modelování a simulaci dynamiky mechanických pohonných jednotek automobilů a strojů, vystupují zde veličiny, které používají prvky jako převodovky, rotační hřídele, spojky atd. SimDriveline obsahuje knihovny pro modelování a simulaci jednorozměrných mechanických systémů. To zahrnuje rotační a translační komponenty,

jako jsou šnekové převody, planetové převodovky, brzdy, řemenové kola, spojky, standardní převodovky, modely motorů a pneumatik. Nástroj je optimalizován ke snadnému použití a urychlení výpočtů v mechanice pohonů. [7]

Knihovna obsahuje prvky např. :

- Band brake, double-shoe brake
- Belt drive, belt pulley
- Planet-planet gear, sun-planet gear

3.1.3 SimHydraulics

SimHydraulics je nástroj rozšiřující SimScape o prostředky k modelování a simulaci hydraulických systémů. Umožňuje modelování systémů obsahujících propojení hydraulických a mechanických komponent s použitím přímé analogie s reálnými prvky. SimHydraulics obsahuje hydraulické a mechanické komponenty, jako jsou čerpadla, ventily, akumulátory, potrubí apod. [7]

Knihovna obsahuje prvky např. :

- Pipe bend, Gradual arena change
- Hydraulic pipe, Variable head tank
- Centrifugal pump, Jet pump

3.1.4 SimElectronics

SimElectronics je určen pro modelování a simulaci elektronických a elektromechanických systémů. SimElectronics umožňuje zahrnout analogovou elektroniku a elektromechanické prvky do multi-fyzikálních modelů v SimScape. Knihovny obsahují prvky, jako polovodičové součástky, motory, řídicí obvody motorů, senzory a další součásti elektrických obvodů, které dovolují vytvářet vlastní uživatelské subsystémy. [7]

Knihovna obsahuje prvky např. :

- DC motor, stepper motor
- Diode, Thyristor
- DC/DC Converter, Generic battery

3.1.5 SimPowerSystems

Samostatný, na SimScape nezávislý modul pro simulaci soustav v oboru výkonové elektroniky a energetiky. Rozšiřuje SimScape o nástroje pro modelování a simulaci výroby, přenosu, distribuce a spotřeby elektrické energie. Knihovny obsahují řadu prvků běžně užívaných v energetických soustavách, jako jsou třífázové stroje a elektrické pohony nebo specifické prvky - flexibilní přenosové systémy střídavého proudu nebo prvky větrných elektráren. Je zde také možnost automatických výpočtů harmonické analýzy, činitele harmonického zkreslení, zatížení soustavy a dalších klíčových analýz pro energetické soustavy. [7]

Knihovna obsahuje prvky např. :

- AC/DC Voltage source, Three-Phase source
- Breaker, Series/Parallel RCL Load
- Brushless DC motor, DTC Induction motor

4 VYTVOŘENÍ MODELU

V této kapitole je popis jednotlivých částí odvíjecí jednotky, a vysvětlení postupu převodu těchto částí do modelového schématu.

1 – Odvíjecí motor

2 – Převodovka

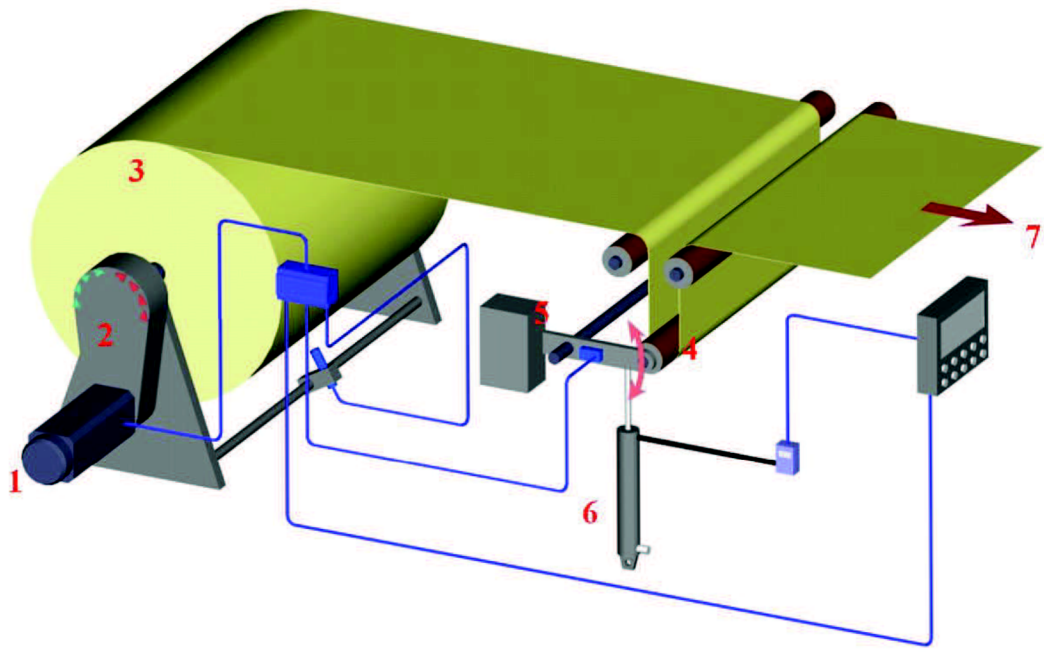
3 – Odvíjecí role

4 – Tanečnice

5 – Měření výchylky tanečnice

6 – Pneumotor pro nastavení tahu

7 – Hlavní motor



Obr. 4-1 : Vzhled a části odvíjecí jednotky [3]

4.1 Odvíjecí motor

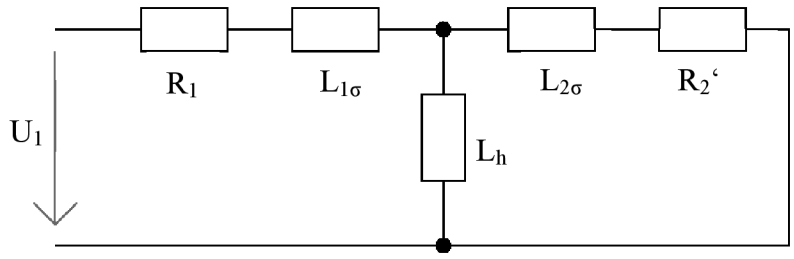
V tomto typu stroje je použit třífázový (3 x 400 V) asynchronní motor typu MCA19S17 od firmy Lenze. V následující tabulce jsou jeho parametry.

Tabulka 4-1 – Parametry motoru MCA19S17

	n_N	M_0	M_n	M_{max}	P_n	I_0	I_n	$U_{n,AC}$	f_n	J
	[r/min]	[Nm]	[Nm]	[Nm]	[kW]	[A]	[A]	[V]	[Hz]	[kg·cm²]
MCA19S17	1700,0	40,0	36,3	180,0	6,4	15,4	13,9	390,0	60,0	72,0

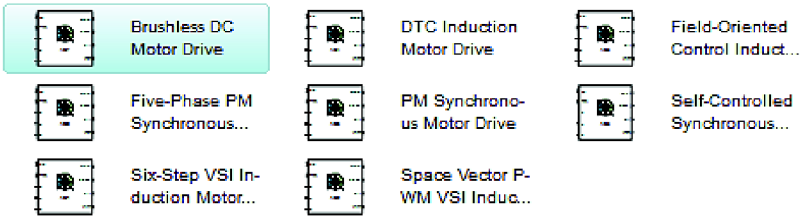
R_1	$R_{UV\ 20\ °C}$	$R_{UV150°C}$	R_2	$L_{1\sigma}$	L	$L_{2\sigma}$	n_{max}	m
[Ω]	[Ω]	[Ω]	[Ω]	[mH]	[mH]	[mH]	[r/min]	[kg]
0,7	1,4	1,9	0,6	2,6	81	3,1	8000,0	48,2

Náhradní schéma motoru :



Obr. 4-2 : Náhradní schéma motoru

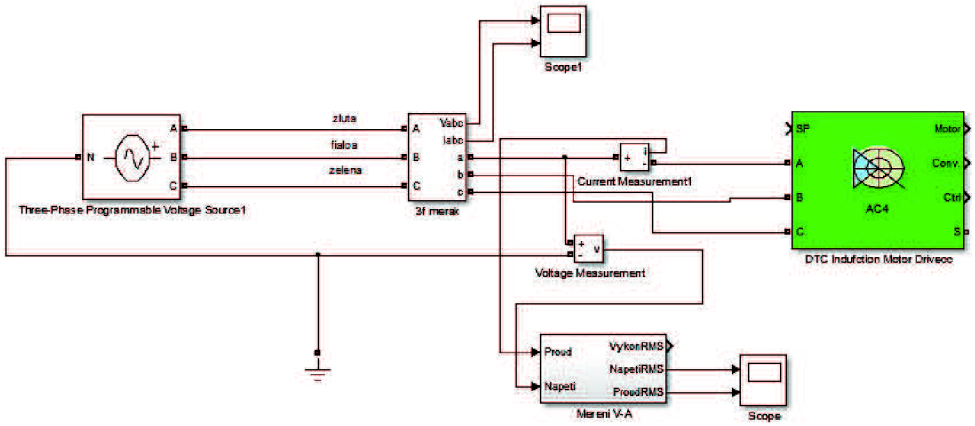
Pro implementaci do simulačního schématu již existují v knihovnách SimScape předpřipravené motory i s řídící elektronikou, přesněji tyto prvky obsahuje knihovna SimPowerSystems. Zde několik základních typů motorů, viz následující obrázek 4-3.



Obr. 4-3 : Výběr motorů v knihovně SimPowerSystems

Na obrázku 4.3 lze vidět několik typů motorů s různými typy řízení; six-step VSI, vector PWM a jiné. V knihovně nalezneme blok odpovídající reálnému typu motoru, vložíme do simulačního schématu a doplníme jeho parametry. V závislosti na použitém bloku máme několik možností řízení motorů - rychlostní a momentové. Dále je možné nastavit typ vstupu do motorů (zpětné vazby), a to buď mechanický vstup – pokud zvolíme tuto možnost můžeme připojovat další bloky z knihovny SimPowerSystems a SimDriveline jako např. hřídele, spojky, převodovky atd. Dalšími možnostmi je rychlostní nebo momentový vstup, do kterých lze zapojit jen Simulinkové bloky – číselná hodnota z bloku „Step“, „Constant“ a podobné.

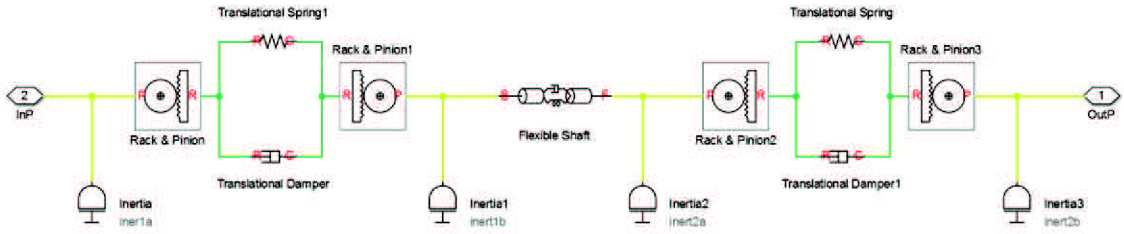
Pro funkčnost musíme připojit elektrický zdroj k motoru z knihovny SimPowerSystems/Eletrical sources. Modelové schéma motoru (bez výstupů a řízení) s napájením je na následujícím obrázku, jsou zde také přidány bloky pro měření napětí a proudu, které mají informativní účel.



Obr. 4-4 : Modelové schéma odvíjecího/sekundárního motoru

4.2 Převodovka sekundárního motoru

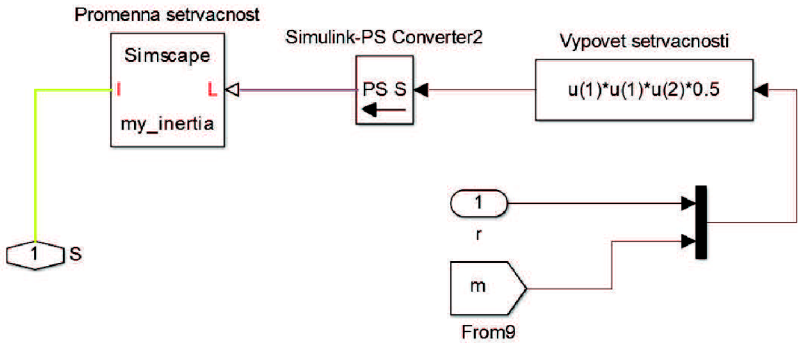
V reálném stroji jsou dvě řemenové (ozubené) převodovky spojené hřídelí, tedy dvoustupňová převodovka. Převodovka takového typu v knihovnách není, musel jsem přejít ke složení převodovky ze samostatných bloků. V modelovém schématu jsem použil blok z kořenové knihovny SimScape/Foundation library blok „Rack & Pinion“, který představuje transformaci rotačního pohybu na translační a obráceně. Z výkresové dokumentace jsem zjistil počet zubů na kolech, délku a šířku řemene a délku propojovací spojky. Prvky „Translational Spring“ a „Translational Damper“ je namodelováno chování, resp. parametry řemenového pásu. Poté jsem zjistil velikosti setrvačností a doplnil je do bloků „Inertia“, které představují setrvačnou sílu rotační části. Výsledné modelové schéma lze vidět na obrázku níže.



Obr. 4-5 : Modelové schéma převodovky

4.3 Odvíjecí role

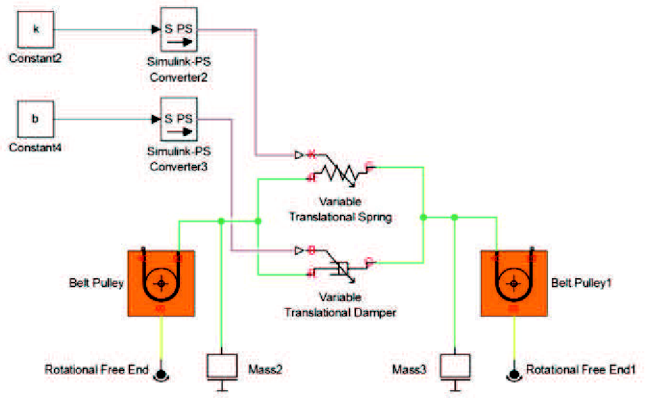
Odvíjený materiál je navinut na dutince o průměru 76,5 mm a délce 800 mm (délka je měnitelná jak v reálném stroji, tak v modelovém schématu), váha této role odpovídá počtu navinutých vrstev, hustotě materiálu a tloušťce vrstev. Některé role nemusejí být ideálně navinuty a vzniká mezi vrstvy vzduchová mezera, proto někdy výpočet hmotnosti z průměru role a hustoty materiálu nemusí být zcela přesný. Dynamika této části je závislá na váze role, resp. její setrvačné síle; váha role se může měnit od desítek kilogramů do stovek kilogramů, tudíž počáteční podmínky této části modelu jsou velice důležité v modelovém schématu. Váha při nepřerušovaném chodu stroje se nemění ve velkém gradientu, materiál je sice odvíjen velikou rychlostí (až 300 m/min), ale vrstvy jsou v řádu desítek mikrometrů. Problém tedy nastává při spouštění stroje pokaždé s jinou váhou role – regulátor rychlosti odvíjecího motoru musí být uzpůsoben tak, aby se motor dokázal rozjet s jakoukoliv váhou role (zatížením motoru). V modelovém schématu je váha přepočítána na setrvačnou sílu, zase tedy použijeme blok jménem „Inertia“. Schéma dle předpokladů má být real-time – parametry jsou v každém kroku simulace aktuální. Hodnoty vah a průměrů rolí jsou zpětně přepočítávány, a tím samozřejmě také i setrvačnost. Jelikož se v knihovnách SimScape nenachází blok, ve kterém lze měnit setrvačnosti při již spuštěné simulaci, bylo potřeba vytvořit nový uživatelský blok „my_inertia“. Postup vytvoření uživatelského bloku je popsán v kap. 4.8 *Vytvoření uživatelské komponenty*.



Obr. 4-6 : Výpočet setrvačné síly a uživatelský blok „my_inertia“

Dalším důležitým krokem je modelování potiskového materiálu a jeho vlastností. U tiskařského stroje je nežádoucí, aby se odvíjený materiál přerušil (nastaven velký tah) nebo nebyl dostatečně natáhnut (nastaven malý tah), přitom regulace tahu byla co nejrychlejší a nejpřesnější. Proto je potřeba nějakým způsobem, a co nejpřesněji namodelovat potiskovaný materiál. Také je potřeba, aby bylo možné parametry materiálů v simulaci nastavovat, jinými slovy spustit simulaci pro několik typů materiálu o různých parametrech.

Materiál je namodelován podobným způsobem jako v případě řemenového pásu v převodovce, a to bloky „Translational Damper“ a „Translational Spring“. Na následujícím obrázku je jen názorné zobrazení části modelu materiálu. Jsou zde dva bloky „Belt Pulley“, které znázorňují válce po kterých putuje materiál (ve stroji jich je několik desítek). Tyto válce jsou na volných ložiskách a jejich rychlost otáčení není nijak řízena, proto je k jejich rotačním portům připojen element „Rotational free end“. Je zde ještě jeden blok jménem „Mass“, který vyjadřuje hmotnost materiálu mezi válci. Dále je potřeba poznamenat, že hodnoty parametrů „Translational Spring“ a „Translational Damper“ jsou vypočítány na základě Youngova modulu materiálu a přepočítány dle skutečných rozměrů materiálu, viz. postup výpočtu níže. [9]



Obr. 4-7 : Model materiálu mezi válci

Výpočet pružnosti [9]:

$$k = E\frac{\mathrm{A}}{l}$$

kde; k – pružnost [Nm]

E – Youngův modul [GPa]

A – řez materiálu (šířka x tloušťka) [m^2]

L – délka materiálu [m]

Výpočet tlumení :

$$b = \sqrt{4 * m * k}$$

kde; b – tlumení [N/(m/s)]

m – hmotnost materiálu [kg]

k – pružnost [Nm]

4.3.1 Youngův modul

Youngův modul, nebo-li modul pružnosti v tahu je definován jako poměr napětí a jím vyvolané deformace materiálu. Viz následující vztah.

$$E = \frac{\sigma}{\varepsilon}$$

kde; E – Youngův modu [Pa]

σ – napětí v tahu [Pa]

ϵ - relativní deformace ($\varepsilon = \frac{\Delta l}{l}$) [-]

Tabulka 4-2 – Hodnoty Youngova modulu, pružnosti a tlumení pro vybrané materiály

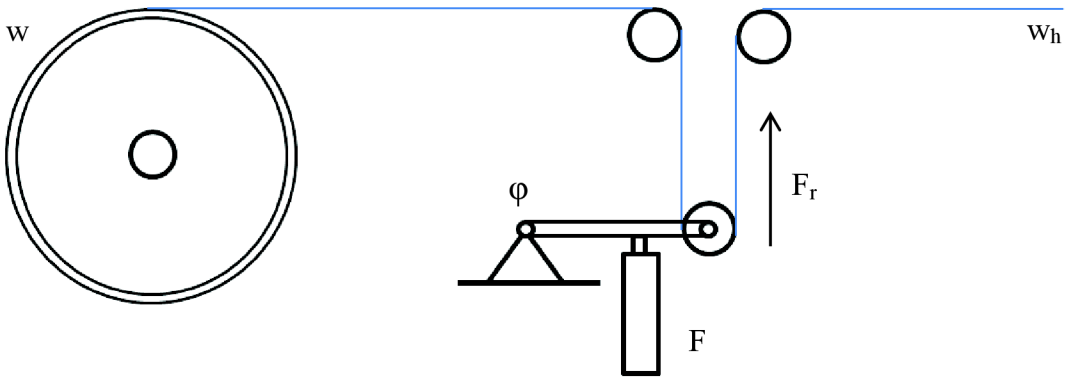
Materiál	Youngův modul [GPa]	k [Nm]	b [N/(m/s)]
LDPE	0,29	24 359	26,060
HDPE	1,55	131 862	60,596
Lepenka	4,96	421600	108,351
Papír	6,54	555 900	124,417
Hliník fólie	71,00	603 500	409,939

4.4 Tanečnice

Tanečnice slouží k regulaci tahu i pro malé změny tahu. Konstrukce se skládá ze tří válců, dva krajní válce jsou napevno uložené, přičemž válec mezi nimi se pohybuje po mechanickém rameni. Potiskový materiál probíhá mezi těmito válci, viz. Obr. 4.6.

Princip je takový, že pokud je potiskový materiál moc napnut (velký tah), materiál přizvedne střední válec nahoru (pokud jde o vertikální tanečnici, u horizontální tanečnice je to velice podobné), změní se výchylka ramene, na kterém je přimontovaný tento střední válec. Tato výchylka je změřena senzorem úhlové výchylky nebo jiným senzorem a poté převedena na diferenci rychlosti odvíjecího motoru ke kompenzaci této výchylky. Pokud nastane tato situace, kdy je střední válec je vychýlen směrem nahoru, musejí se otáčky odvíjecího motoru zrychlit pro dorovnání na požadovaný tah (když zrychlíme, dodáme mu více materiálu - zmenšíme tah), v opačném případě, diference rychlosti je záporná – zpomalujeme rychlost odvíječky a natahujeme materiál. Výhoda tanečnice spočívá v celkem jednoduchém principu a funkčnosti. Jelikož dokáže díky své konstrukci potlačit vysokofrekvenční kmity, které vznikají třením materiálu o válec při vysoké rychlosti, můžeme konstatovat, že funguje jako dolno-frekvenční propust – reaguje jen na větší výchylky ramene a motor není zatěžován kmitáním akčního zásahu regulátoru. Malé výchylky tahu jsou totiž kompenzovány samotnou konstrukcí.

Nevýhodou je konstrukční náročnost na přesnost jednotlivých dílů a horší časová stálost – po delší době vnášejí nepřesnosti ložiska ramene a „zatuhnutí“ pneumatického válce.



Obr. 4-8 : Principiální schéma tanečnice

Zde na obrázku vidíme principiální konstrukci tanečnice. Vyskytují se zde veličiny;

w – otáčky/rychlost odvíjecího motoru

w_h – otáčky/rychlost hlavního motoru

F – hodnota požadovaného tahu (tlaku) v pneumatickém motoru

F_r – reálný tah materiálu

φ – úhel natočení ramene

Rovnováha tanečnice (nulová poloha tanečnice) je dána vztahem

$$F - F_r = 0$$

4.5 Měření výchylky tanečnice

Jak již bylo zmíněno v předchozí kapitole, je měřena výchylka ramene středního válce tanečnice. Měření této výchylky je možné mnoha způsoby a to hlavně v závislosti na konstrukci ramene; např. opt. senzor vzdálenost, odporové měření velikosti vysunutí pneumatického motoru – měření na pneumatickém válci (bez ramene), měření úhlového vychýlení atd. Způsob měření není předmětem této diplomové práce, avšak důležité je zpracování této hodnoty; hodnota výchylky je v měřítku, které je dané použitým převodníkem a také normalizací hodnot, pokud tedy byla provedena. Při znalosti maximální možné fyzické výchylky a rozsahu měřitelných hodnot je možné dostat převodní vztah pro výpočet výchylky tanečnice.

4.6 Pneumotor pro nastavení tahu

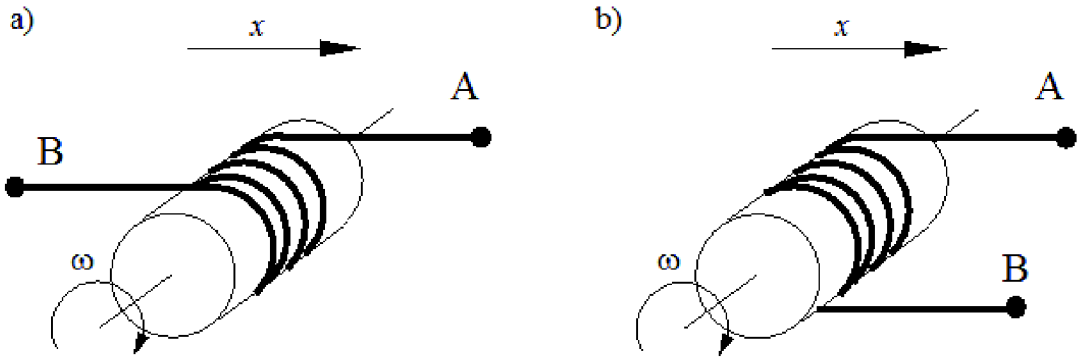
Pro udržování požadovaného tahu je potřeba tuto hodnotu nějak zakomponovat nebo transformovat na potiskový materiál, k tomu slouží pneumotor, který je pevně spojen s ramenem středního válce tanečnice (viz obr. 4-6), tento pneumotor je ovládán vzduchem o určité hodnotě tlaku (v jednotkách Pa). Hodnota tlaku udržovaná v tomto pneumotoru se automaticky přepočítá z nastaveného tahu. A jak lze také na tomto obrázku vidět, pokud je nastavena určitá hodnota tlaku (tahu) na pneumatickém válci, působí tento pneumotor silou F, pokud je materiál napnut silou F_r, která se rovná požadované hodnotě tahu, válec je v nulové poloze (středové poloze) a v tuto chvíli je materiál napnut na požadovanou sílu. Z tohoto faktu vyplývá, že regulace tahu je prováděna regulátorem polohy na nulovou hodnotu výchylky tanečnice.

4.7 Hlavní motor

Tato část není předmětem této diplomové práce, ale je zde zakomponována z důvodu modelování síly tahu materiálu. Materiál je tahán tímto namodelovaným motorem a mezi tímto motorem a odvíjecím motorem vzniká napětí (tah), které je transformováno na pozici tanečnice tak, jak je to ve skutečnosti.

Pro namodelování tahu materiálu hlavním motorem je použit blok „Rope drum“, který reprezentuje buben, nebo-li štoček, na který se namotává pružný materiál, přičemž namotávaný materiál neprokluzuje na bubnu, tyto vlastnosti se přesně hodí pro aplikaci do tohoto modelu. Orientaci pohybu navíjeného materiálu (parametr „Rope windup

type“) lze nastavit dvěma způsoby (jak lze vidět na obr. 4-9), dále také může být nastavena i setrvačná síla bubnu.



Obr. 4-9 : Způsoby napojení bloku „Rope Drum“ []

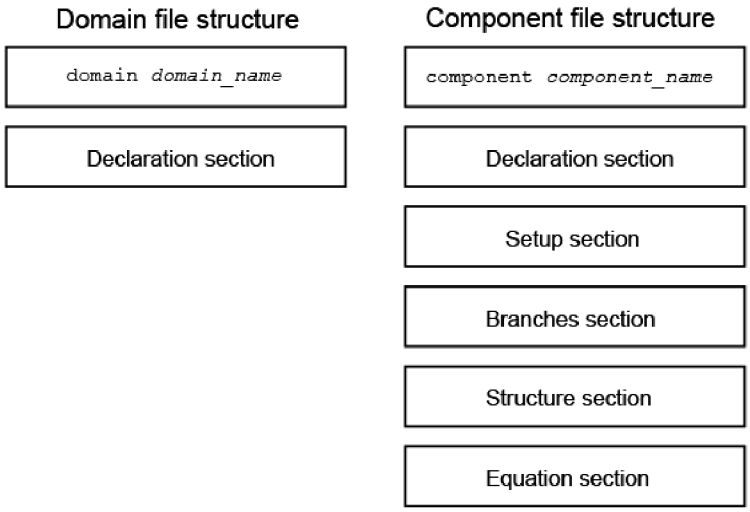
Na *Obr. 4-9a* je namotávaný materiál jen sunut dál – B a A mají stejný směr, na *Obr. 4-9 b* mají A a B opačný směr pohybu.

4.8 Vytvoření uživatelské komponenty

Spojnice mezi bloky a jejich porty reprezentují fyzikální domény (elektrickou, hydraulickou, magnetickou, mechanickou translační/rotační atd.), přes tyto spojnice probíhá výměna energie a dat dané fyzikální domény. Typ fyzikální domény musí odpovídat typům portů které spojujeme.

Podobným jazykem jaký používá Matlab lze vytvořit novou uživatelskou komponentu nebo doménu použitelnou v SimScape. Komponenta může být jakkoliv složitá, ale musí splňovat určitou strukturu a pravidla. Důležitou věcí pro začátek vytváření je struktura a znalost klíčových slov. Používají se standardní klíčová slova používaná i v jiných jazycích, z celkového pohledu vypadá komponenta jako funkce psaná v jazyce C, jen s jinou strukturou.

Prvním krokem je nutné definovat o jaký typ modelu půjde. *Doména* (domain) je model popisující nějakou fyzikální doménu (rotační, tanslační atd.). Za druhé je to *komponenta*, která není obsažena ve standardních knihovnách SimScape. Tyto fyzikální domény odpovídají typům portů, takže lze s vytvořením nové domény vytvářet zcela nové typy SimScape bloků se speciálními typy portů odpovídajícím definici nově vytvořené domény. Na obr. 4-10 je struktura domény a komponenty. Deklarací a naprogramováním nové fyzikální domény se zde nebudu zabývat, zajímavější je popis sestavení nové komponenty. Popis struktury komponenty bude detailně pokračovat níže.



Obr. 4-10 : Struktura domény a komponenty [10]

Komponenta se dělí na části *inputs*, *nodes*, *outputs*, *parameters*, *variables*, *funkction setup* a *equalation* a má přesný sled částí jak jsou vypsány. Prvně musíme rozhodnout, jakou komponentu budeme vytvářet (jakého typu komponenta je, jakými fyzikální veličiny bude popsána atd.), dle toho musíme zvolit typ SimScape portů (*nodes*). Pokud budeme vytvářet komponentu modelující například translační pohyb s dvěma vstupy tohoto typu, nadefinujeme porty takto: [11]

```
1 -   nodes
2 -   A = foundation.mechanical.translational.translational; % A:right
3 -   B = foundation.mechanical.translational.translational; % B:right
4 -   end
```

Tento zápis odpovídá tomu, že v naši nové komponentě budou dva mechanické porty A a B translačního typu, a blok bude typu *mechanical* (mechanický), propojitelný s bloky v knihovně *„Foundation library“*. Znak „%“ v tomto jazyku (zde ho vidíme úplně napravo) neznamená jako v jazyku Matlabu začátek komentáře, ale je zde použit pro nastavování pozice portů a jejich jména, nebo k nadefinování popisků. Popisky se poté zobrazují v *Parametrech komponenty* (obr. 4-11). Pozici portu, vstupu nebo výstupu nastavíme řetězcem *left*, *right*, *top* nebo *bottom*, který poté v tomto případě bude označen písmenem A nebo B - název portu nastavíme řetězcem napsaným před dvojtečku. Tato komponenta tedy bude moci být propojena s ostatními prvky, které mají stejný typ portu jako tyto.

Poté přejdeme k definici vstupů (*inputs*) a výstupů (*outputs*), tyto vstupy a výstupy jsou také SimScape-ové, ale zde již není obousměrný tok jako v typu *nodes* a reprezentují jen jeden signál. Dále musejí mít specifikovanou počáteční hodnotu a jednotku. Jelikož mají přiřazenou nějakou fyzikální jednotku, není je možné propojit

s dalšími bloky ze Simulinku – k transformaci těchto signálu pro propojením s dalšími bloky používaných v Simulinku existují bloky „PS-Simulink Converter“ a „Simulink-PS Converter“.

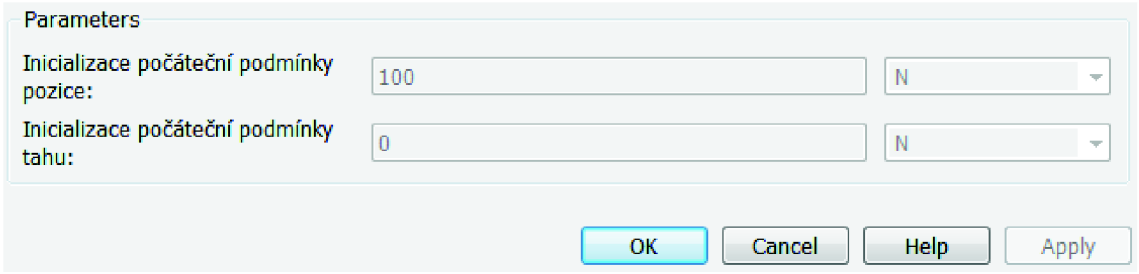
```
inputs
poz_tah = {0, 'N'}; % C:left
end

nodes
...
end

outputs
      pozice = {0 , 'm'}; % P:bottom
      F = {0 , 'N'};    % F:bottom
      end
```

Další částí je nadefinování parametrů (*parameters*), tyto parametry budou vystupovat jako proměnné měnitelné uživatelem; tyto parametry jsou nastavitelné v *Parametrech komponenty* (při dvojkliku na vytvořenou komponentu, obr. 4-11) a je potřeba definovat počáteční hodnotu a veličinu jako v případě vstupů a výstupů. Místo pozice a názvu (jak to bylo v případě *inputs* a *nodes*) zde definujeme popis proměnné, který bude viditelný v *Parametrech bloku* (obr. 4.10). V programu bloku bude vystupovat takto zadefinovaná proměnná pod jménem „poc_pozice“ nebo „poc_tah“.

```
parameters
poc_pozice = { 100, 'N' };      % Inicializace počáteční podmínky
                                   pozice
poc_tah = {0, 'N'};             % Inicializace počáteční podmínky
                                   tahu
end
```



Obr. 4-11 : Parametry vytvořeného bloku

Další v pořadí je část *variables*, tato část je podobná jako v předchozím případě *parameters*, až na to, že tyto proměnné nejsou viditelné ani měnitelné uživatelem v *Parametrech komponenty*, jsou to interní proměnné, většinou používané pro přístup k fyzikálním veličinám vyskytujících se v daných portech nebo také k definici konstant.

Je u nich potřeba nadefinovat směr toku těchto, veličiny se rozdělují do dvou skupin; „across“ a „through“, tento parametr se nastavuje v sekci *function setup*, viz níže. Rozdělení proměnných dle fyzikální domény je na následující Tabulce 4-3.

Through - Proměnné, které jsou měřeny do série s komponentou.

Across - Proměnné, které se měří paralelně ke komponentě.

Tabulka 4-3 - Rozdělení proměnných na Across a Through [12]

Fyzická doména	Across proměnná	Through proměnná
Elektrická	Elektrické napětí	Elektrický proud
Hydraulická	Tlak	Průtok
Magnetická	Magnetomotorická síla	Magnetický tok
Mechanická rotační	Úhlová rychlost	Točivý moment
Mechanická translační	Rychlost	Síla
Pneumatická	Tlak a teplota	Hmotnostní průtok a tepelný tok
Teplotní	Teplota	Tepelný tok

Následující částí je *function setup*, kde se provádí inicializace a nastavení proměnných. Provádí se jen jednou pro každou instanci komponenty v průběhu kompilace modelu.

```
1 - function setup
2 - across( v, A.v, B.v);
3 - through( f, A.f, B.f);
4 - end
```

Poslední částí je sekce *equalation*, která popisuje chování komponenty rovnicemi a vzájemnými vztahy, ve kterých vystupují proměnné *variables*, *parameters* a samozřejmě také *nodes*. Tato část se provádí cyklicky při běhu modelu (oproti části *function setup*). Posledním krokem je vytvoření a sestavení finálního bloku SimScape, k tomu slouží příkaz *ssc_build*. Pro vytvoření uživatelské knihovny (uživatelská komponenta musí být pod novou knihovnou) zadáme příkaz například v následujícím

tvaru – *ssc_build +my_library*, znak „+“ označuje vytvoření nové knihovny a tedy i složky jménem „my_library“. Je potřeba mít při vytváření komponenty v Matlabu nastavenou složku *Current Folder* na takovou, ve které nachází soubor nové komponenty. Po zadání tohoto příkazu se vytvoří nová knihovna pojmenovaná dle toho, jaký řetězec jsme zadali do příkazu *ssc_build* za znak plus. V tuto chvíli máme možnost vkládat tuto nově vytvořenou komponentu do jakéhokoliv modelového schématu. [10]

5 VÝBĚR SPRÁVNÉHO SOLVERU

Vytvořený model se skládá jak z mechanické části (spojky, válce, klece motoru atd.), tak i z elektrické (nápájení motorů) a řídící části (regulátory, ovládání rychlosti motorů, logické podmínky atd.). Tudíž se zde vyskytuje mnoho časových konstant řádově velmi vzdálených od sebe – řídící část má mnohonásobně, až o několik řádů menší časovou konstantu než materiály a mechanické části. Z Shannonova teorému vyplývá, že vzorkovací frekvence musí být minimálně dvakrát tak větší, než je maximální frekvence vyskytující se v systému. Aplikací tohoto teorému na náš případ je, že perioda vzorkování by měla být minimálně dvakrát tak menší, než je nejmenší časová konstanta vyskytující se v modelu.

5.1 Stiff systémy

V případě systému s velmi rozdílnými časovými konstantami dochází k situaci, kdy do celkového řešení v daném časovém intervalu vstupují některé části modelu jen minimálně, takovému systému říkáme *Stiff* systém. V české terminologii nemá toto slovo přesný ekvivalent, ale v některých literaturách je znám pod pojmem „systém se silným tlumením“ (dále bude pužíváno stiff). Dle [13], pokud jsou v Simulinku použity SimScape bloky, pak je tento model vždy stiff, správný postup výběru solveru při této situaci popíší dále. Obecně přesná hranice k určení, zda systém je nebo není stiff, dána není. Avšak existuje zavedený tzv. pojem „stiff poměr“, který lze vypočíst následujícím vztahem; [14]

$$r = \frac{max|Re(\lambda_i)|}{min|Re(\lambda_j)|}$$

kde: i,j = 1, 2, ... , k

k - počet vlastních čísel matice **A**

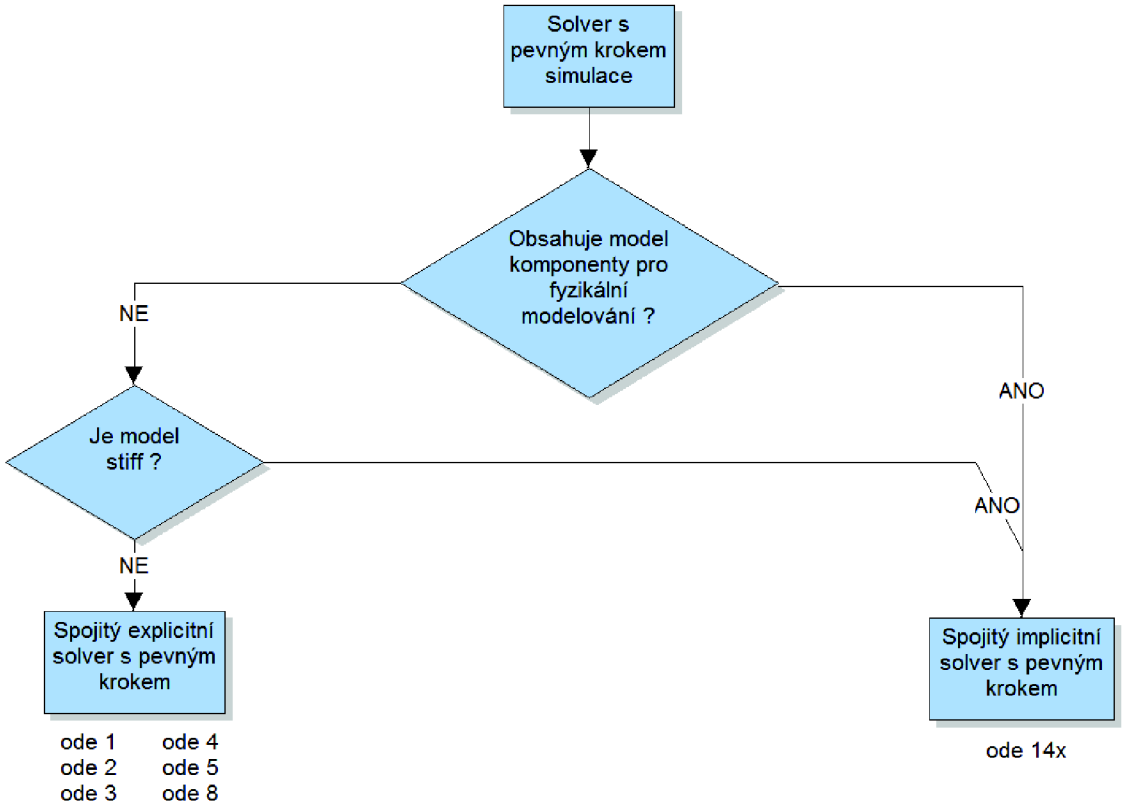
λ - vlastní čísla matice **A** z teorie stavového popisu systému;

$$x' = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$

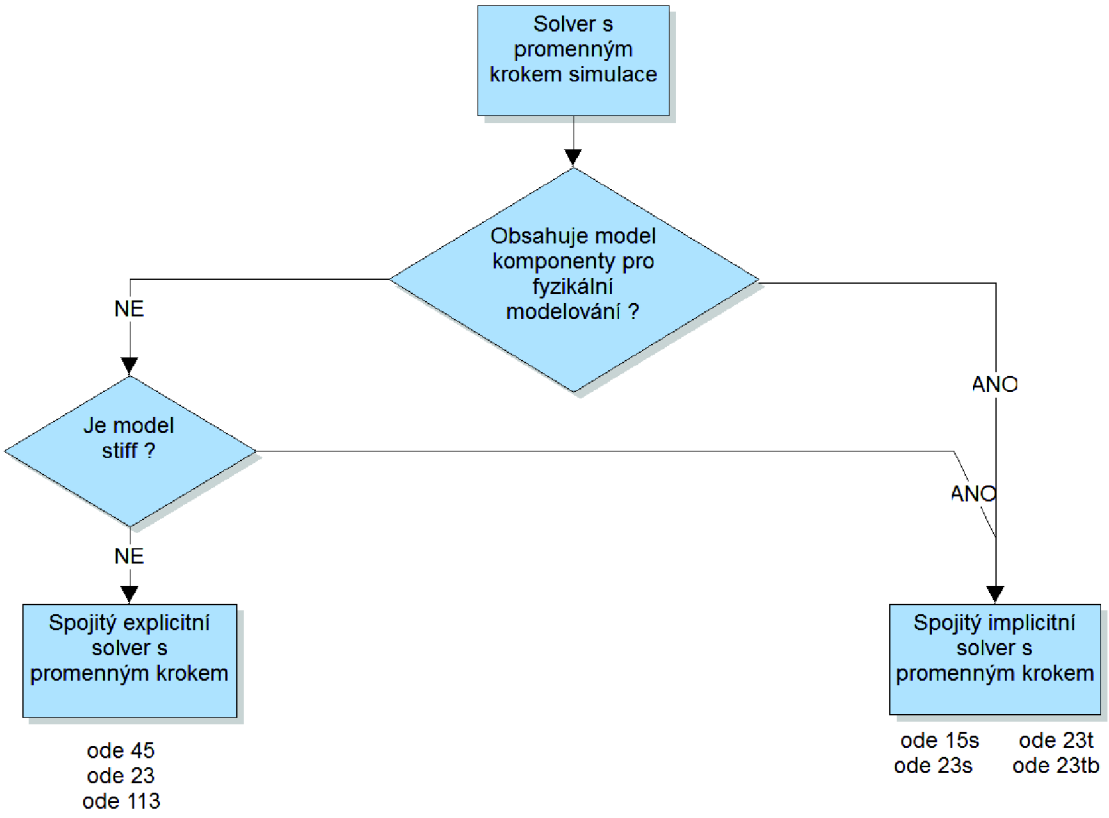
Pokud je hodnota *r* větší než cca 10^3 (jak jsem již zmínil, přesná hranice dána není), pak se již ve většině případů jedná o stiff systém, obecně hodnota *r* vyjadřuje spíše „jak moc je systém stiff“. Nechtěnou vlastností těchto systémů je, že numerické řešení může být nestabilní a explicitní (přímé) numerické metody havarují, i v případě kdy analytické řešení může být stabilní. Nefunkčnost explicitních metod se také stala jednou z definic stiff systémů. Řešením je použití implicitních numerických metod. [14]

5.2 Dostupné solvery v Matlabu

Jelikož jsou v modelu použity bloky z knihoven SimScape, jsme limitováni tím, že vzniklý model bude stiff. Musíme tedy zvolit optimální solver, který dokáže řešit stiff modely. Solvery se dělí na dva základní typy; na solvery s proměnným krokem simulace a na solvery s pevným krokem simulace. Algoritmus, nebo-li spíše postup výběru správného solveru je na obrázcích 5-1 a 5-2. Dalším krokem výběru solveru může být časová náročnost dané výpočetní metody. Je potřeba rozhodnout, zda jsou výsledky potřeba velice přesné na úkor dlouhé výpočetní doby, nebo na druhou stranu postačí menší přesnost za mnohem kratší dobu. Toto rozhodnutí je na dané situaci a náročnosti vytvořeného modelu. Zde je potřeba dát pozor na situaci, kdy zvolíme příliš velký krok simulace - model může být nestabilní a celá simulace zhavarovat, popř. dávat nesmyslné výsledky.



Obr. 5-1 : Výběr solveru při pevném kroku simulace [13]



Obr. 5-2 : Výběr solveru při proměnném kroku simulace [13]

Solvery v simulinku se tedy rozdělují na dva základní typy, s pevným a proměnným krokem simulačního kroku. Výběr správného typu solveru se odvíjí od dynamiky vytvářeného modelu. Pokud model obsahuje přepínače, diskrétní součástky jako např. PWM měnič aj., je potřeba použít solver s pevným krokem. Solver s proměnným krokem simulace použijeme v případě, že modelovaný systém se skládá jen ze spojitých částí, jako například dynamický model pružiny a jím podobné. [13]

6 UŽIVATELSKÉ ROZHRANÍ (GUI)

Grafické uživatelské rozhraní (Graphical User Interface, dále jako GUI) je uživatelské rozhraní sestavené z grafických objektů (komponent) jako jsou tlačítka, textová pole, posuvné seznamy, výběrové menu apod. GUI slouží k zastřešení projektu, zjednodušení procesu editace parametrů a práci s výsledky simulace nebo měření. Poskytuje rozhraní mezi uživatelem a aplikaci podřízeným kódem. Je-li GUI dobře navrženo, mělo by být každému uživateli intuitivně zřejmé, jak jeho jednotlivé součásti fungují. GUI umožní uživateli obsluhovat aplikace bez znalosti příkazů a syntaxe, potřebné pro práci s Matlabem. [16]

GUIDE (Graphical User Interface Development Environment) je vývojové prostředí Matlabu poskytující soustavu nástrojů pro tvorbu GUI. Toto vývojové prostředí nástroje proces návrhu a programování GUI velmi zjednodušují. GUIDE poskytuje nejen sadu návrhových nástrojů, ale také generuje výsledný M-file, který obsahuje kód pro ovládání, inicializaci a spouštění GUI. Tento M-file poskytuje kostru pro implementaci callback funkcí (funkce, které se vykonají při aktivaci některého objektu v GUI). [16]

Návrhové nástroje GUIDE:

- *Alignment Tool* – seřadí označené objekty mezi sebou dle zvoleného stylu
- *Menu editor* – vytváří řádkové menu okna a kontextové menu
- *Tab Order Editor* – mění pořadí, ve kterém jsou prvky vybrané tabulátorem
- *Toolbar Editor* – přidá v GUI pod řádkové menu rychlé zkratky (Tisk, Legend, Colorbar atd.)
- *Editor* – přepnutí do psané formy (otevření M-file pro psaní kódu)
- *Property Inspector* – okno pro nastavování vlastností komponent a objektů
- *Object Browser* – seznam komponent a objektů v daném sekci GUI

6.1 Vytvoření GUI

Kvůli rozsáhlosti schémat a vysokému počtu proměnných parametrů je žádoucí vytvoření uživatelského rozhraní k ovládání simulačních schémat. Pro vytvoření slouží v Matlabu zmíněné GUIDE, zadáním slova *guide* do *Command Window* se spustí průvodce vytvoření nového GUI, nebo otevření již vytvořeného GUI pro následnou úpravu. Po úspěšném ukončení průvodce se vytvoří a zobrazí prázdné okno (Figure), ve kterém lze vkládat různé komponenty – tlačítka (Push button), textová pole (Edit box), výběrová tlačítka (Radio button), texty (Static text) atd. Také se automaticky vytvoří M-file pro psaní kódu obsluhující komponenty.

Kód v tomto M-filu se vytváří také automaticky a ke každému elementu se vytvoří dvě základní funkce - *_Callback* a *_CreateFcn*, kde před podtržítkem je název, nebo-li „tag“ dané komponenty. V každé automaticky vytvořené funkci jsou vstupem tři proměnné – struktury *hObject*, *eventdata* a *handles*. Ve struktuře *hObject* jsou přístupné vlastnosti komponenty, ve které se nacházíme – je to odkaz (struktura) na vlastnosti aktuální komponenty, např. při psaní kódu ve funkci tlačítka jsou ve struktuře *hObject* vlastnosti *Enable*, *String*, *Visible*, *BackgroundColor* aj. Tyto vlastnosti je možné měnit, nastavovat nebo získávat příkazy *set* a *get* – např. příkazem *set(hObject,'String','Run')* změníme text tlačítka na *Run*. Obdobně je to při použití druhého příkazu pro získávání vlastností - *get(hObject,'String')*, tímto příkazem získáme text zobrazený na tlačítku. Struktura *handles* slouží pro přístup k ostatním komponentám, které se vyskytují v tomto GUI. Funguje to obdobně, jen je potřeba znát přesný název elementu – např. *set(handles.edit1,'String','12')* nastavíme text v komponentě nazvané *edit1* na 12, a to i když se nacházíme v jiné komponentě. Jméno komponenty (tagu) získáme dvojklikem na komponentu – zobrazí se *Property Inspector* a v něm nalezneme vlastnost „Tag“.

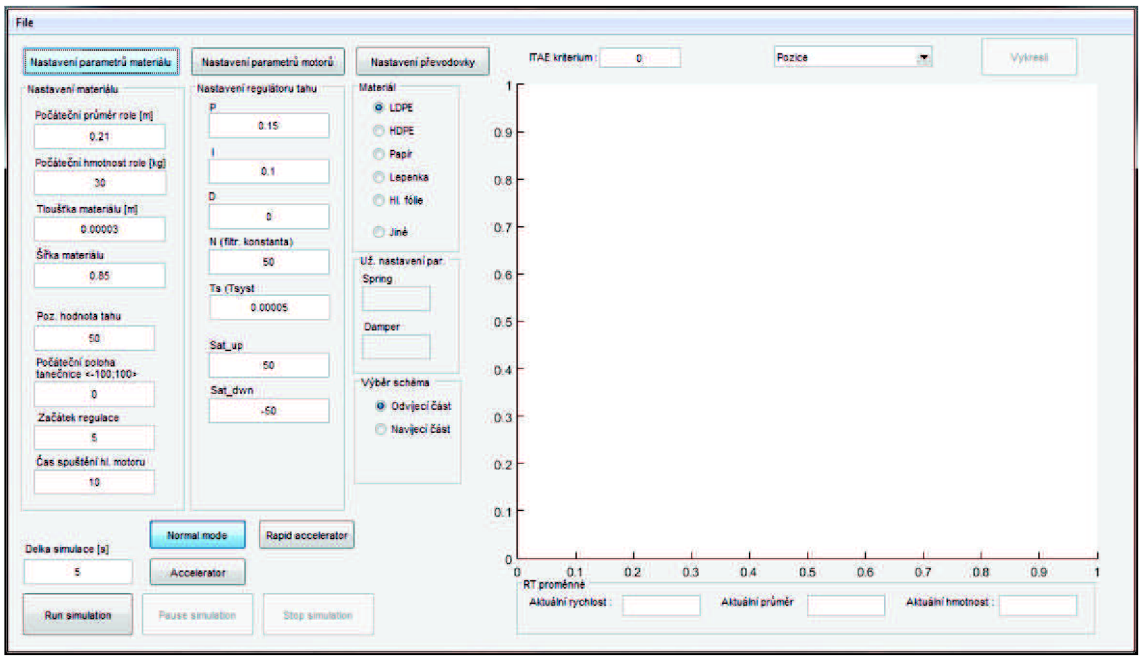
Je potřeba dát pozor při získávání numerické hodnoty z textových polí, jelikož hodnota v tomto poli je text, je nutné získanou hodnotu typu *string* převést na číselnou. K tomu se používají příkazy *str2num* nebo *str2double* (*str2double* se používá častěji, protože je méně náročný na strojový čas oproti *str2num*). Výsledný příkaz pro získání numerické hodnoty z textového pole může vypadat např. takto *str2double(get(handles.edit1,'String'))*.

V GUIDE se nachází mnohem více komponent než textové pole a tlačítko, hojně se využívá *group-radio-buttony* (skupinově-výběrové tlačítka), které slouží pro výběr 1 z N prvků – uživatel vybírá z N možností (N je počet výběrových tlačítek), přičemž je možné vybrat jen jednu možnost. V této práci je to využito např. k výběru potiskového materiálu, nebo ke zvolení simulačního schéma. Dalšími komponenty jsou zaškrtávací políčka (Check Box) pro výběr m z N prvků, kde $m \leq N$, posuvník (Slider)

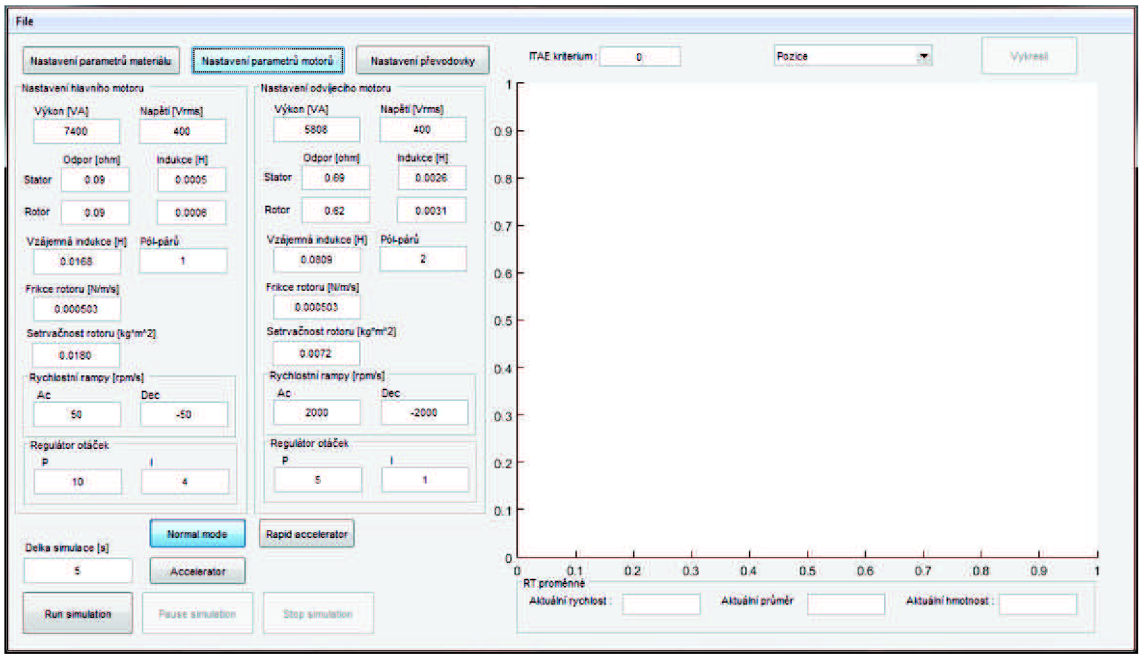
pro nastavování hodnoty v určitém, předem daném rozsahu a v neposlední řadě graf (Axes) pro vykreslování průběhů.

6.2 Ovládání s podporou Matlabu

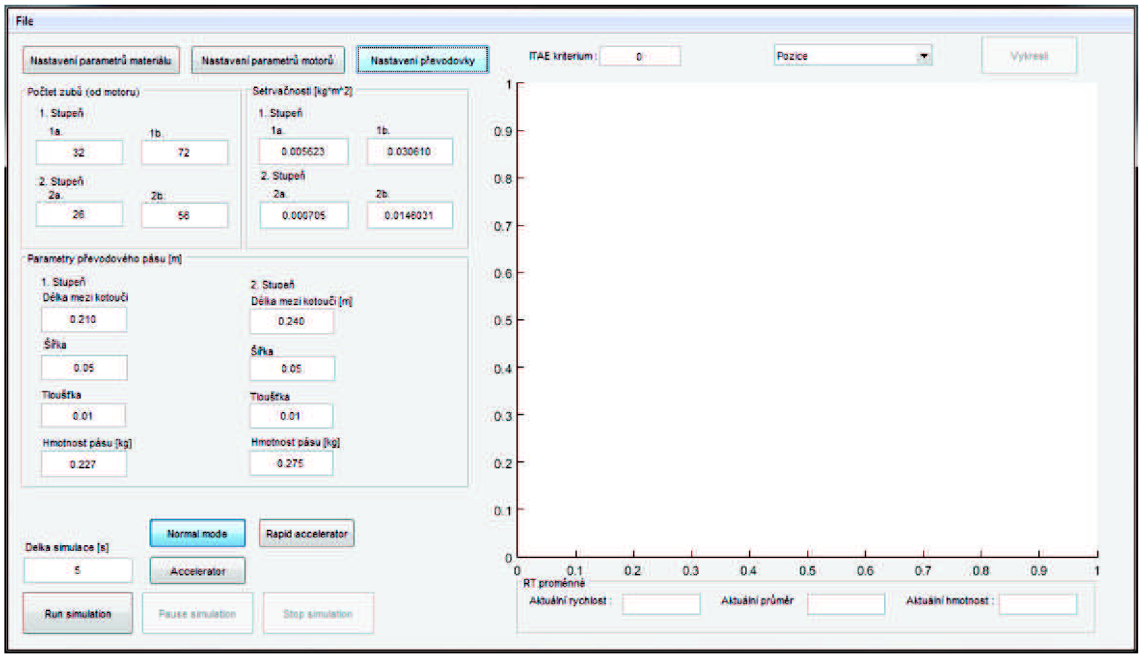
Ovládání s podporou Matlabu je rozsáhlejší oproti ovládání stand-alone aplikace, protože je možné v simulačním schématu měnit víceméně všechny parametry bloků. Jak je na následujících obrázcích 6-1, 6-2 a 6-3 vidět, uživatel má možnost měnit mnoho parametrů, od nastavení potiskového materiálu, regulátoru tahu, převodovky sekundárního (odvíjecího/navíjecího) motoru, po parametry sekundárního a hlavního motoru. Zvolením jednoho výběrového tlačítka v sekci „Výběr schéma“ zvolíme odpovídající modelové schéma pro simulaci.



Obr. 6-1 : Vzhled GUI pro ovládání s podporou Matlabu – Nastavení materiálu



Obr. 6-2 : Vzhled GUI pro ovládání s podporou Matlabu – Nastavení motorů



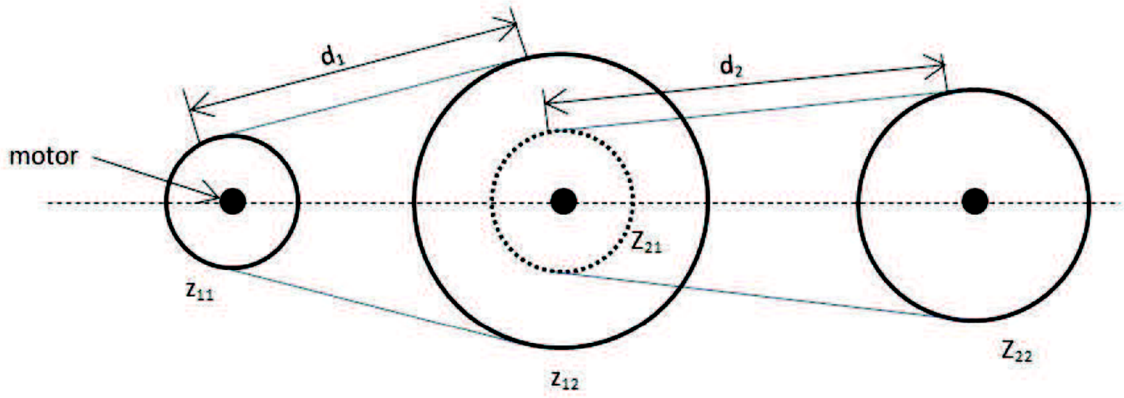
Obr. 6-3 : Vzhled GUI pro ovládání s podporou Matlabu – Nastavení převodovky sek. motoru

Třemi tlačítky umístěnými v levém horním rohu přepínáme mezi nastavením parametrů motorů, materiálu a převodovky sekundárního motoru. Na obrazovce „Nastavení parametrů materiálu“ nalezneme více sekcí; v sekci „Nastavení materiálu“ nastavujeme tloušťku, šířku a počáteční hodnoty role, požadovanou hodnotu tahu materiálu a počáteční polohu tanečnice. V další sekci jménem „Nastavení regulátoru tahu“ nastavujeme parametry pro regulaci tahu. Zde je možné měnit jak parametry

složek regulátoru, tak vzorkovací periodu regulátoru a hodnoty saturačního omezení akčního zásahu. V dalším bloku měníme typ materiálu (LDPE, HDPE atd.), pokud však daný materiál je jiného typu než je zde na výběr, je možné po výběru možnosti „Jiné“ zadat parametry materiálu ručně, tyto zadané hodnoty materiálu budou pravděpodobně získány aplikací „Ovládání identifikace parametrů“, který je popsán níže (kap. 6.4 *Ovládání identifikace parametrů*).

Na druhé obrazovce (obr. 6.2), na kterou se přepneme tlačítkem „Nastavení parametrů motorů“ vidíme dva bloky. V levém bloku nastavujeme parametry hlavního motoru, a to základní parametry týkající se výkonu motoru, napětí a hodnot odporů a indukčností z náhradního schéma motoru. V dolní části tohoto bloku je nastavení rychlostních ramp a regulátoru otáček. Druhý blok je zcela identický, ale pro sekundární motor, při zvolení odvíjecího schématu budeme nastavovat odvíjecí motor, v opačném případě navíjecí motor.

Poslední obrazovkou (obr. 6-3) je nastavení převodovky sekundárního motoru, které vidíme na obrázku 6-4. Postup převedení reálné převodovky do schématu je popsáno v kapitole 4.2 *Převodovka sekundárního motoru*. Aby ovládání korespondovalo s modelových schématem, je potřeba zadat více než jednoduchou numerickou hodnotu převodového poměru, ten je vypočítán automaticky až po vyplnění, popř. změnění těchto hodnot. V horní polovině této obrazovky nastavujeme parametry pevné části převodovky, a to počet zubů na kotoučích a jejich setrvačnosti. Tyto parametry lze jednoduše dohledat ve výkresové dokumentaci. V dolní polovině nastavujeme parametry týkající se řemenového pásu. Na obrázku níže vidíme schématické znázornění převodovky pro jednodušší pochopení zadávaných parametrů. Např. hodnota d_1 je délka pásu mezi kotouči prvního stupně, z_{22} je počet zubů na vzdálenějším kotouči od motoru ve druhém převodovém stupni.

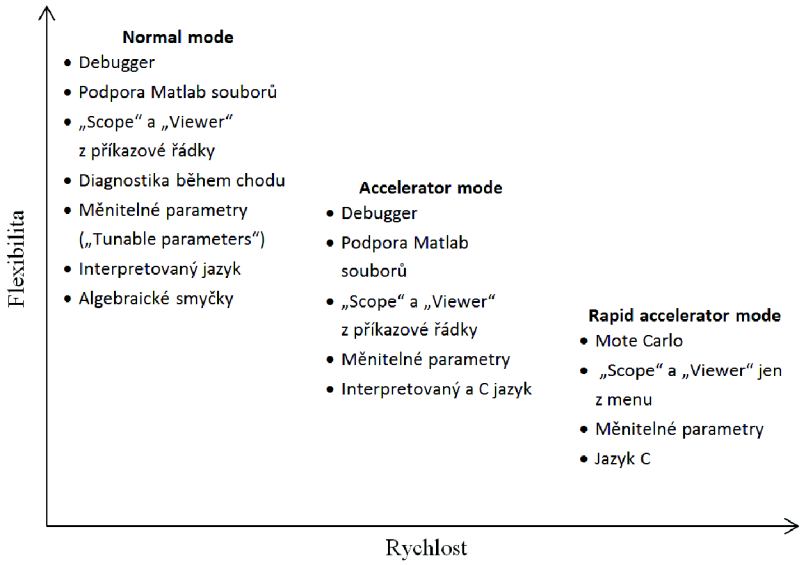


Obr. 6-4 : Schéma převodovky sekundárního motoru

Posledním krokem zbývá výběr schématu, nastavení délky simulace a nakonec spuštění simulace. Je zde také možnost výběru režimu simulace – na výběr jsou tři možnosti – „Normal mode“, „Accelerator“ a „Rapid accelerator“, pokud uživatel nevybere ani jednu možnost, je automaticky zvolen typ Normal. Z názvů je jasné, že Rapid accelerator bude nejrychlejší možností, která je na výběr, zvolení této možnosti je optimální v případě, pokud vyžadujeme, aby simulace proběhla co nejrychleji. Rapid accelerator je nejlepší volbou vzhledem k času vykonání simulace, avšak tento režim má i negativní vlastnosti, viz. v následující kapitole.

6.2.1 Režimy simulace

Při použití jiného typu provádění simulace (režimu) než „Normal“ klesá flexibila a použitelnost některých možností a funkcí k ovládání simulačního procesu. Normální režim nabízí největší flexibilitu týkající se úpravy, změny parametrů (a to i při již spuštěné simulaci), rychlost spuštění simulačního modelu a zobrazování výsledků. Na úkor všech těchto možností je nejpomalejší volbou. Tento typ simulace se nejčastěji používá při jednoduchých modelech, kdy simulace trvá v řádu jednotek sekund a méně. Při použití jiného typu simulace je potřeba nejdříve vygenerování kódu, např. „Rapid accelerator“ využívá převedení simulačního schématu do jazyka C, a v závislosti na složitost modelu proces inicializace a generování kódu zabere obvykle desítky sekund i mnohem více. Mohlo by se tedy stát, že generování kódu by trvalo stejně dlouho nebo i déle než průběh samotné simulace. Proto se typy „Accelerator“ a „Rapid accelerator“ využívají v situacích, kdy běh simulace několikanásobně překročí čas potřebný ke generování kódu. Jedna z hlavních nevýhod „Rapid acceleratoru“ je ta, že nelze již spuštěnou simulaci pozastavit a získat datový výstup ze simulace.



Obr. 6-5 : Závislost flexibility a rychlosti provádění simulačního procesu

Ovládání poskytuje uživateli možnost sledovat průběh a odezvu na nastavení parametrů dříve než skončí simulace. Uživateli to umožní zareagovat na špatně nastavené parametry ve chvíli zjištění chyby, a ne až na konci simulace. Zda parametry, co uživatel nastavil jsou odpovídající a popř. nemají-li za následek pád nebo nestabilitu simulace, má možnost kontrolovat po celou dobu trvání simulace.

Vykreslení požadovaného průběhu uživatel vybere pomocí výběrového menu (Popupmenu) v pravém horním rohu. Na výběr jsou možnosti tah, rychlost motorů, akční zásah regulátoru tahu atd. Po zvolení daného průběhu a následným kliknutí na tlačítko „Vykresli“ se simulace pozastaví, nahraje se výstup ze simulace a vykreslí se zvolený průběh do grafu pod tímto tlačítkem. Po ukončení simulace nebo při pozastavení simulace se načtou dosavadní data do pracovního prostoru (workspace) ve speciálním typu struktury, který se nazývá „Simulink.SimulationOutput“. Tato struktura je specifická hlavně v tom, že oproti regulérní struktuře používané v Matlabu, kde k datům ve struktuře přistupuje pomocí operátoru tečka - např. *Mereni1.senzorA.data*, tak zde přistupujeme pomocí funkce *get*. Pro přístup do této struktury musíme také znát jména proměnných obsažených ve struktuře, tak jak je to i v regulérní struktuře. Příkaz může vypadat např. takto: *get(outputSimulation, 'SenzorA')*, pokud i proměnná *SenzorA* je struktura, tak při uložení do proměnné stejného jména poté k této struktuře přistupujeme jako ke standardní struktuře.

6.2.2 Přenos dat mezi funkcemi komponent

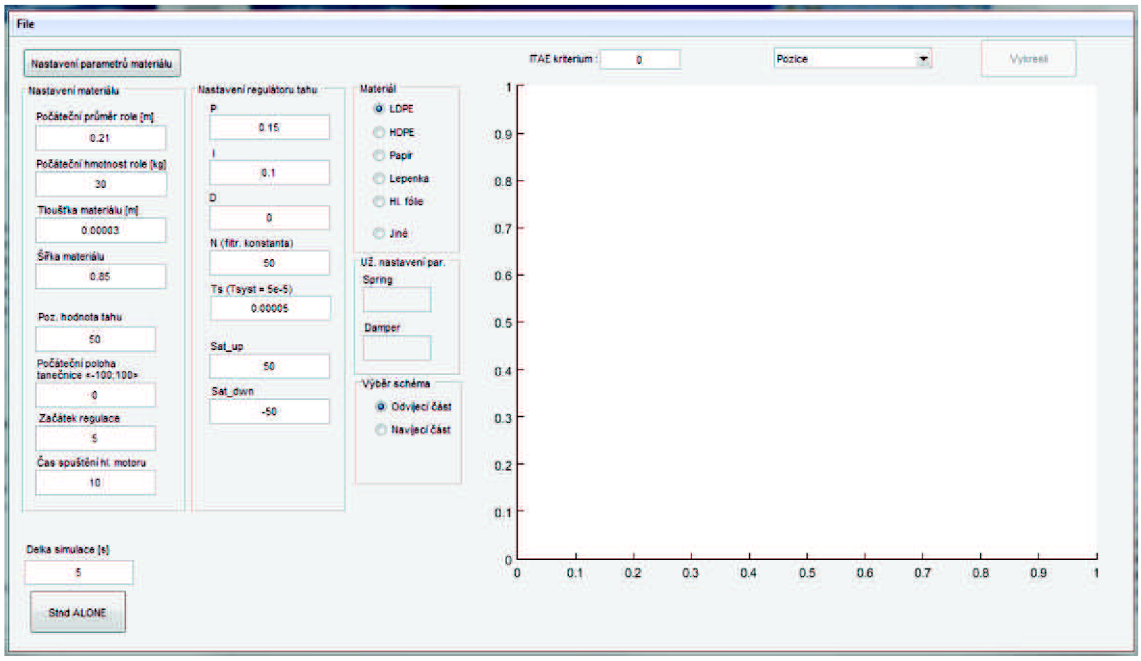
Komponenty v uživatelském rozhraní jsou vlastně funkce a přenos dat mezi funkcemi komponent je někdy obtížné - jelikož lze komponenty (tlačítka atd.) používat v předem neznámém sledu, je nemožné programově zajistit, aby funkce přenášely data mezi sebou jako vstupní proměnné do funkce. Jeden z možných způsobů je použití globálních proměnných, tento způsob je ale nevhodný z několika důvodů, jak už kvůli špatné dohledatelnosti (zjištění, kde se mění jejich hodnota), tak další věcí je nepřehlednost a možná kolidace jmen globálních a lokálních proměnných. Je třeba dát velký důraz na správné pojmenování globálních proměnných, tak aby nenastala situace, že by se ve funkci vyskytovaly dvě proměnné stejně pojmenované, přičemž jedna z proměnných lokální a druhá globální. Globální proměnné je vhodné použít například k inkrementaci počtu opakujícího se eventu, kdy počet takových proměnných není nijak závratný.

Jinou možností je využití pracovního prostoru Matlabu (workspace), který evidentně není přístupný ve funkci komponenty. V této funkci je, jak jsem již zmínil přistup jen k lokálním proměnným, které jsou volány s touto funkcí (*hObject, eventdata, handles*). Pro načtení dat z pracovního prostoru existuje funkce *evalin* (která má i jiné oblasti použití a účel, ale to není v této práci důležité). Je potřeba zajistit a ošetřit, aby se při vyčítání daná proměnná vyskytovala v pracovním prostoru. Pokud vyčítáme

proměnnou, která není uložena v pracovním prostoru, zahlásí funkce chybu (při následném ošetřování havarijních stavů a chyb by se použila nějaká metoda pro odchytávání výjimek, například funkce „try-catch“ nebo aktivací tlačítka po úspěšném uložení proměnné do pracovního prostoru). Tedy např. příkazem *evalin('base','vyst1')* načteme proměnnou jménem *vyst1* z pracovního prostoru, přesněji z hlavního pracovního prostoru (který je viditelný při spuštění Matlabu), použití tohoto pracovního prostoru je určeno použitím klasifikátoru 'base'. Důvodem klasifikátoru je existence více pracovních prostorů, každé simulační schéma totiž může mít (a také má) vlastní pracovní prostor, akorát při výchozím nastavení je pracovní prostor modelu identický s tím hlavním. V opačném případě, pokud je potřeba uložit proměnnou do pracovního prostoru, použijeme funkci *assignin*. Předpis funkce je velice podobný jako v případě vyčítání proměnné – např. *assignin('base','vstup1',vstup1)*, zde je také potřeba určit do jakého pracovního prostoru se má proměnná uložit - vyskytuje se zde zase klasifikátor 'base'. Druhým vstupním parametrem je jméno, pod kterým bude proměnná uložena v pracovním prostoru. Poslední vstupní parametr je proměnná s daty, která se má uložit.

V tomto uživatelském rozhraní je použit časovač s periodou 5 sekund pro vyčítání hodnot v reálném čase (Aktuální rychlost, Aktuální průměr a Aktuální hmotnost) nacházející se v pravém dolním rohu. Tento časovač je také použit k indikaci průběhu simulace. Jak jsem již zmínil, generování kódu při použití jiného režimu simulace než „Normal“ je časově náročné. Uživatel potřebuje mít přehled o stavu simulace - zda simulace byla již spuštěna, jestli byla úspěšně spuštěna, popřípadě jestli ještě probíhá inicializace. Pro tento účel je vytvořen „Wait Bar“ a v něm zobrazeno, v jakém stavu se simulace nachází („Initializing“, „Processing“ nebo „Running“). Aby uživatel měl jistotu, že proces spouštění simulace nezamrzl, je zobrazovaný text každou periodou časovače aktualizován – za stavem procesu spouštění přibývají tečky, po dosažení třech teček za textem se zobrazí opět jen jedna tečka, tento proces se opakuje až do řádného spuštění simulace. Díky této vlastnosti má uživatel jistotu, že proces spouštění probíhá a program pracuje. Wait bar se naplňuje dle procentuální hodnoty celkového postupu simulace, hodnota postupu simulace je jednoduše vypočítána podílem delka_sim/akt_cas.

6.3 Ovládání stand-alone aplikace



Obr. 6-6 : Vzhled GUI pro ovládání stand-alone aplikace

Rozložení, vzhled a algoritmus ovládání je velice podobný jako v předchozím případě. Rozdíl je v tom, že toto uživatelské rozhraní ovládá vygenerovanou stand-alone aplikaci z modelu, nastavení a postup vytvoření této aplikace je vysvětlen v kap. 7.4 *Vytvoření samostatného modelu*. Takto vytvořená aplikace umožňuje externě měnit parametry standardních bloků používaných v simulačním prostředí Simulink. V této práci je ve velkém využíváno bloků z toolboxu SimScape, přičemž tyto bloky nepodporují externí změnu parametrů, a to ani před spuštěním (inicializací) stand-alone aplikace. Změnu parametrů v SimScape blocích lze tedy jen provést v modelovém schématu v Simulinku a poté znova z modelu vygenerovat novou verzi stand-alone alikace. Nastavení možnosti, resp. povolení měnitelných parametrů, aby je bylo možné editovat ve stand-alone aplikaci, lze nalézt v kapitole 7.3, v odborné anglické literatuře lze tyto pojmy nalézt pod názvem „Inline parameters“, popř. „Tunable parameteres“.

Z důvodu nemožné změny parametrů v SimScape blocích, jsou právě tyto parametry uživateli zablokovány měnit. Na druhou stranu výhodou této stand-alone aplikace je velká rychlost výpočtu simulace. Díky těmto vlastnostem je tohle ovládání přizpůsobeno pro ladění parametrů regulátoru tahu, současně s možností nastavovat různé počáteční hodnoty materiálu (počáteční průměr, váha, materiál atd.) a typ materiálu. S tímto faktem je také související skutečnost, že spuštěnou stand-alone aplikaci není možné pozastavit a tudíž vypisovat real-timové proměnné periodicky, tak

jako v předchozím případě. Samozřejmostí je změna délky trvání simulace, viz následující úryvek kódu.

```
1 -     if(get(handles.radiobutton9,'Value'))
2 -       schema = 'odvijecka';
3 -
4 -     elseif(get(handles.radiobutton10,'Value'))
5 -       schema = 'navijecka';
6 -     end
7 -
8 -     cas = str2double(get(handles.edit4,'String'));
9 -     cas = round(cas*10)/10;
10 -    name = [schema, '.exe -tf ',cas, ' &'];
11 -
12 -    system(name)
13 -    warndlg('MATLAB continues after calling EXE');
14 -
15 -    flag = true;
16 -
17 -    while flag
18 -      disp('EXE still running');
19 -      flag = isprocess(schema);
20 -      pause(2)
21 -    end
```

Prvním krokem je potřeba zjistit jaké schéma a délku simulace uživatel zvolil, to je provedeno v první polovině tohoto úryvku kódu. Důležitá část je na řádku 10, kde ze zvolených parametrů poskládáme výsledný tvar volání spustitelného (executable) souboru. Při volání použijeme klíčový parametru „-tf“, kterým nastavujeme délku trvání simulace, následujícím textem za tímto parametrem nastavíme hodnotu tohoto parametru standardním způsobem, používaným obecně v této problematice. Zde je potřeba, aby tento parametr měl tvar desetinného čísla, a to jen s jedním číslem za desetinnou čárkou.

Na konec je potřeba přidat znak „&“ (ampersand), kterým určíme, že aplikace bude spuštěná na pozadí Matlabu. Pokud tento znak nevložíme, Matlab bude při provádění aplikace na tomto řádku ve stavu „Busy“ (zaneprázdněn) a čekat na dokončení aplikace, tudíž nebude možné provádět žádné jiné akce. Po spuštění daného schématu s těmito parametry (příkazem *system*) kontrolujeme v cyklu *while*, zda daná aplikace ještě běží, a to s možností provádět i jiné akce v uživatelském rozhraní. Kontrolování, zda aplikace běží je prováděno funkcí *isprocess*, která je vytvořena jako externí funkce pro prostředí Matlab, dostupná z [17]. Po stisknutí tlačítka a nutné incializaci některých proměnných je potřeba informovat uživatele, že byl spuštěn model a probíhá simulace, to je zajištěno řádkem 13.

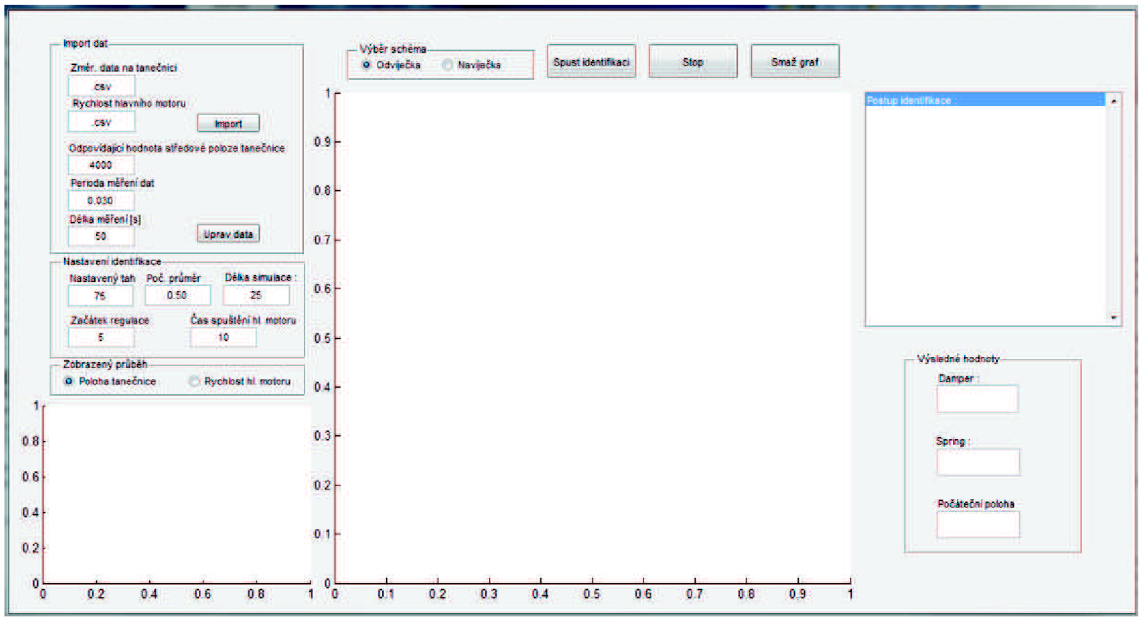
6.3.1 Výměna dat mezi GUI a modelem

Výměna dat mezi stand-alone aplikací a uživatelským rozhraním probíhá přes soubory. V prvním případě, když nastavujeme parametry simulace se po stisknutí tlačítka „Spuštění simulace“ inicializují zadané parametry v textových polích a proběhnou potřebné výpočty dalších parametrů (materiálové konstanty, hmotnost materiálu atd.) a provede se funkce „up_param“. Tato funkce zajistí, že všechny proměnné parametry, potřebné pro správný a bezchybný běh simulace, se uloží ve specifickém tvaru a sledu do textového souboru. Na straně stand-alone aplikace (executable souboru) je aktivováno rozhraní C-API, ve kterém je v jazyku C/C++ napsáno načtení parametrů zapsaných ve zmíněném textovém souboru a přiřazení do příslušných proměnných, specificky pojmenovaných v simulačním schématu. Podrobnější popis je možné nalézt v kapitole 7.4.2 *Nastavení interface*. V druhém případě, po dokončení simulace se vytvoří maticový soubor (s příponou .mat), ve kterém jsou uložena data zvolených průběhů.

Nastavení ukládání výsledných dat z modelu do souboru je možné dvěma způsoby; prvním je v simulačním schématu ke sledovanému signálu připojíme blok „To workspace“. V tomto bloku nastavíme jméno signálu a formát, jakým se bude ukládat. Pokud je žádoucí, aby signál byl jen hodnota o jednom prvku (v této práci např. proměnná „akt_prumer“ - žádána je jedna hodnota, a to ta poslední), nastavíme parametr „Limit data points to last“ na hodnotu *1*, jinak *Inf*. Pokud použijeme tento způsob, přičemž bloků „To workspace“ je více, je nevhodné v každém bloku nastavit „Sructure With Time“, protože bychom zbytečně ukládali čas simulace s každým tímto blokem. Lepší způsob je nastavení ve všech blocích „Structure“ a v nastavení simulace (Model Configuration Parameters) zaškrtnout možnost „Time“ v sekci „Data import/Export“. Druhým způsobem je ve stejné sekci zaškrtnout také „Time“ a navíc volbu „States“, zvolením této možnosti se ukládají všechny signály ze simulačního modelu do souboru. To má za následek velký objem dat a také s tím související obtížné zpracování dat. V obou případech je potřeba zvolit možnost „MAT-file logging“ v sekci „Interface“ v záložce „Code Generation“.

Po ukončení simulace se vytvoří soubor s daty pojmenovaný stejně jako spuštěný model. Data načteme příkazem *load('jmeno_schema.mat')* a uložíme je do nějaké proměnné a poté s nimi můžeme dále pracovat.

6.4 Ovládání identifikace parametrů



Obr č. 6.1 : Vzhled GUI pro ovládání identifikace parametrů

Tento program slouží k identifikaci parametrů materiálu a počáteční podmínky tanečnice. Při použití nového materiálu ve stroji nebo materiálu s jinými parametry se provede měření na stroji a tato změřené data se naimportují do tohoto programu. Formát dat by měl být v tabulkovém souboru, optimálně s příponou „.csv“ (Comma Separated Values File), kde v prvním sloupci je čas a v druhém naměřené data. Měření se provádí na senzoru polohy tanečnice a rychlosti hlavního motoru měřené v dm/min. Data z hlavního motoru jsou důležitá především k tomu, aby simulační schéma bylo řízeno stejným signálem jako při měření na stroji. Při měření dat může nastat situace, kdy měření začne dříve než aktivita stroje – v prvním kroku, po zapnutí stroje se spustí regulace tahu na odvíjecím/navíjecím motoru, poté až za určitou dobu nastane rozběhnutí hlavního motoru (řídícího signálu). K zavedení této skutečnosti do simulačního schématu jsou k tomu určeny textové pole v bloku „Nastavení identifikace“.

Po importu dat následuje potřebná úprava dat, je nutné zadat ustálenou střední hodnotu ze změřených dat, tak aby se v modelu tato ustálená hodnota jevila jako nulová (odpovídala nulové poloze tanečnice). Důvodem je, že v simulačním schématu je použita regulace na nulovou hodnotu (výchylku). Poté je nutné také zadat vzorkovací periodu měření. Při spuštění identifikace tlačítkem „Spusť identifikaci“ se průběh polohy tanečnice a rychlosti hlavního motoru uloží do souboru ve formátu „Microsoft Acces Table Shortcut“ (soubor s příponou .mat) a vzápětí spuštěný model již používá tyto průběhy. Model je tedy řízen průběhem hlavního motoru a výstup polohy tanečnice

v modelu je srovnávám s importovaným průběhem. Z rozdílu z těchto průběhů je vypočítávána chyba Kvadratickým integrálním kritériem, což je kvadrát rozdílu změřené polohy tanečnice na stroji a průběh polohy tanečnice z modelu integrován přes celou dobu simulace. Hodnota z tohoto kritéria je vstupem do minimalizačního algoritmu, známý pod jménem „Nelder-Mead Simplex Method“ popsaný v kap. 8.1. Průběh identifikace se vypisuje v pravém horním rohu do List Boxu. Zde je postupně vypisováno kolikátá probíhá iterace v minimalizačním algoritmu a jaké jsou při této iteraci nastavené hodnoty hledaných parametrů. Identifikují se tři parametry – náhradní schéma materiálu se skládá z pružiny (Linear Spring), tlumiče (Linear Damper) a počáteční polohy tanečnice. Po úspěšné identifikaci se výsledné hodnoty těchto proměnných vypíší do textových polí umístěných v pravém dolním rohu.

Po identifikaci parametrů je možné přistoupit k ladění regulátoru tahu programem popsaným v kap 6.3 *Ovládání stand-alone aplikace*. Pokud ale máme v plánu nastavovat i parametry motorů a s tím spojené například rychlostní rampy, otáčkové regulátory atd., musíme využít program z kap 6.2 *Ovládání s podporou Matlabu*.

7 STAND-ALONE APLIKACE

V této kapitole bude vysvětlen převod vytvořených modelů s uživatelským rozhraním do jazyku spustitelným bez potřeby nainstalovaného Matlabu. Nevýhodou Matlabu je jeho pořizovací cena a tudíž pro většinu firem nedosažitelný programový prostředek. Problém nastává, když je potřeba využít některé funkce z Matlabu v jiném programovacím jazyku, a to např. v jazyku C/C++, ve kterých je někdy obtížné určité funkce naprogramovat. A právě Matlab umožňuje řešit tento problém. Dokáže převádět aplikace nebo funkce do samostatně spustitelných aplikací a knihoven v jazyku C nebo C++. Pro vytvoření takové aplikace je potřeba mít nainstalovaný Matlab Compiler. [18]

Takto vytvořená aplikace lze přenášet multiplatformě a lze spustit bez nainstalovaného Matlabu a jeho komponent. Jednodušší je i distribuce takto zabaleného programu, uživatel jen spustí spustitelný soubor (executable file) a díky propojení modelu a uživatelského rozhraní uživatel intuitivně ovládá simulační schéma. Další výhodou stand-alone aplikace je také uzavřenost (šifrování) kódu a tedy nemožnost změny nebo nahlédnutí do originálního kódu a tím zajištění ochrany duševního vlastnictví.

7.1 Matlab Compiler

Matlab Compiler umožňuje kompilovat uživatelské programy (GUI aj.) vytvořené v Matlabu do samostatně spustitelných aplikací a softwarových komponent a usnadní tím distribuci programů pro koncové uživatele. Aplikace vytvořené v tomto softwarovém nástroji nevyžadují ke svému chodu Matlab, ale využívají tzv. Matlab Compiler Runtime (MCR). Při vytváření aplikace (tento proces se nazývá „building“) můžeme zvolit možnost přibalení k vytvářené aplikaci MCR, který je potřebný ke spuštění vytvořených aplikací na počítači, bez potřeby nainstalované (licencované) verze Matlabu. MCR je volně šiřitelný softwarový nástroj, oproti softwarovému matematickému nástroji Matlab. Uživatel po obdržení aplikace nejdříve nainstaluje MRC dodaný s aplikací, který by se měl pokaždé přibalit k aplikaci, aby nevznikaly rozdíly ve verzi MRC potřebného ke spuštění aplikace a MRC nainstalovaném v počítači. [19]

Matlab Compiler umožňuje vytvářet:

- Samostatné aplikace (Stand-Alone Applications)
- Sdílené knihovny C/C++
- Moduly add-ins do Excelu
- .COM objekty
- .NET aplikace

Matlab Compiler plně podporuje jazyk Matlabu a téměř všechny jeho toolboxy (rozšiřitelné knihovny, funkce atd.). Nicméně, některá omezení zde jsou, týká se to některých toolboxů, které nejsou přizpůsobeny pro generování samospustitelného kódu jako např. symbolický matematický toolbox (Math Toolbox). Dále je to např. většina z předem vytvořených grafických uživatelských rozhraní, která jsou již zahrnuta v prostředí Matlab a s ním spojené toolboxy a v neposlední řadě také některé funkce, které nelze volat přímo z příkazové řádky (Command window).

Kompilované aplikace lze spustit pouze na operačních systémech, na kterých lze spustit Matlab. Vzhledem k tomu, že Matlab Runtime poskytuje plnou podpor jazyka Matlab a také i jazyka Java, spouštění zkompilované aplikace trvá přibližně dlouho jako spuštění Matlabu. Důvodem je nutnost nahrát všechny potřebné prostředky pro spuštění aplikace a run-time knihovny, které jsou velmi obsáhlé.

7.2 Simulink Coder

Simulink Coder vytváří a realizuje C/C++ kód ze simulačních schémat, diagramů, „stateflow“ (stavových) grafů a funkcí. Vygenerovaný zdrojový kód může být použit pro „real-time“ (v reálném čase), „nonreal-time“ aplikace a testování způsobem „hardware-in-the-loop“. Po vygenerování kódu pomocí Simulink Coderu je možné simulační schéma spustit a pracovat s ním bez podpory Simulinku a Matlabu. [20]

Vlastnosti:

- ANSI/ISO C/C++ kód a spustitelné soubory pro diskrétní, spojité, nebo hybridní modely
- Inkrementální generování kódu pro rozsáhlé modely
- Podpora datových typů integer, floating-point a fixed-point

- Single-task, multitask, a vícejádrové generování kódu s nebo bez RTOS (Real-Time Operating System)

- Externí simulační režim pro nastavování parametrů a monitorování signálu

7.3 Tunable a Inline parameters

Pro změnu parametrů stand-alone aplikace se používají soubory C-API, generované Simulink Coderem. Rozhraní C-API poskytuje prostředníka pro možné dotazování a upravování laditelných parametrů. [21]

Ve vygenerovaném kódu z modelu, jsou dva druhy laditelných parametrů:

- Parametr bloku - Hodnota je vyjádřena jako číselný výraz. Parametr zesílení bloku „Gain“ nastavíme na určitou, přesně danou hodnotu. Tento parametr zesílení je laditelný parametr bloku – nevystupuje zde žádná jmenná proměnná. Abychom změnili hodnotu zesílení tohoto bloku, je potřeba použít strukturu modelu.

- Parametr modelu – V pracovním prostoru (workspace) přiřadíme jmenné proměnné určitou hodnotu. Tuto proměnou nastavíme jako hodnotu nějakému bloku, např. hodnotu zesílení bloku „Gain“. Pokud je tato proměnná nastavená jako tunable (musí být objekt Simulink.Parametr), po změně hodnoty této proměnné se změní její hodnota v bloku, a to i když je ve více blocích. Tyto proměnné jsou uloženy ve struktuře *jmeno_modelu_P*, hodnotu proměnné tedy změníme zápisem - *jmeno_modelu_P.jmeno_promenne = 10*.

Typy proměnných :

Inlined

Takové parametry zmenšují potřebu využití RAM a zvyšují efektivitu generovaného kódu. Proces kompilace nepřidělí parametru bloku číselné hodnoty proměnných, ale nastaví parametry jako jazykové proměnné. Lze tedy proměnné nastavené jako *inlined* nalézt ve vygenerovaném kódu stejně jako v modelovém schématu. [21]

Tunable

Nastavení proměnných jako tunable nám zajistí ladění numerických parametrů v generovaném kódu. V kódu jsou reprezentovány číselné parametry bloku a proměnných. Pro ovládání proměnných vygenerovaného kódu se používají tzv. „storage class“, určující typ a použitelnost struktur a parametrů. Tunable parametry se také

používají v modelech spuštěných v Simulinku, kde takto nastavené proměnné mohou být měněny i při již spuštěné simulaci.

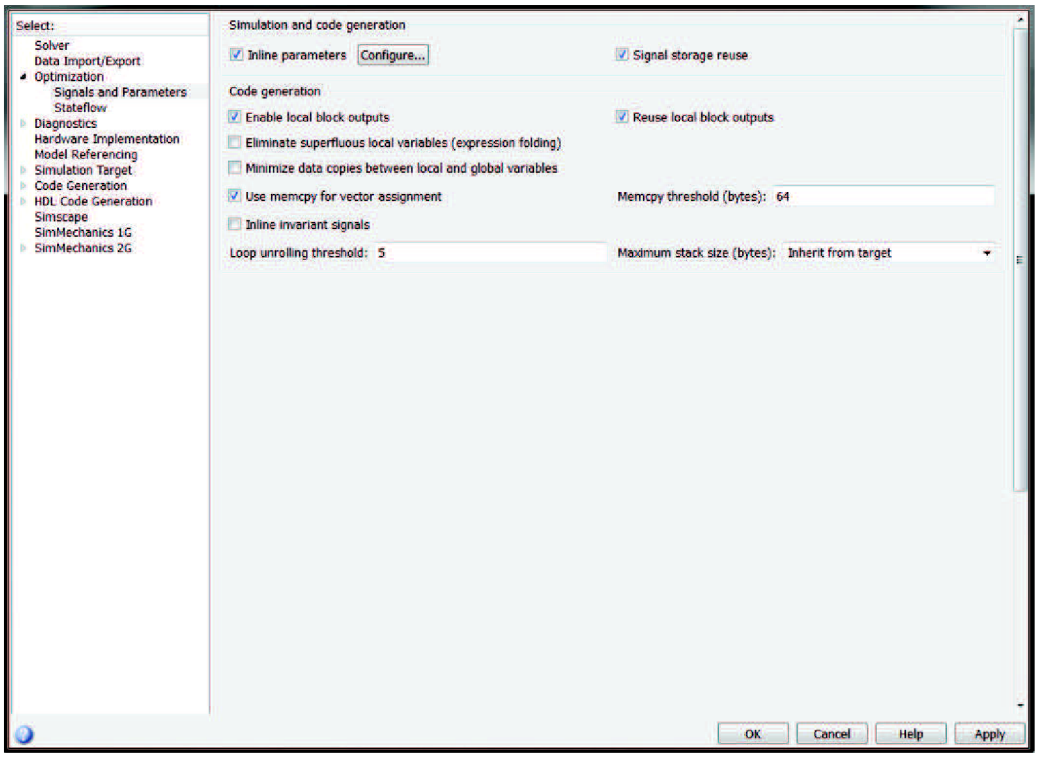
Typy tunable parametrů (StorageClass) [22]:

SimulinkGlobal: Uložení parametru jako pole generované v globální struktuře parametrů *jméno_modelu_P*. Takto nastavené proměnné lze nastavit v generovaném kódu před spuštěním modelu.

ExportedGlobal: Deklaraci a definici parametru v generovaném kódu lze jako samostatnou globální proměnnou. Možné ladění parametru během provádění programu.

ImportedExtern: Import definice parametru z externího kódu. Možná změna parametrů během spuštění.

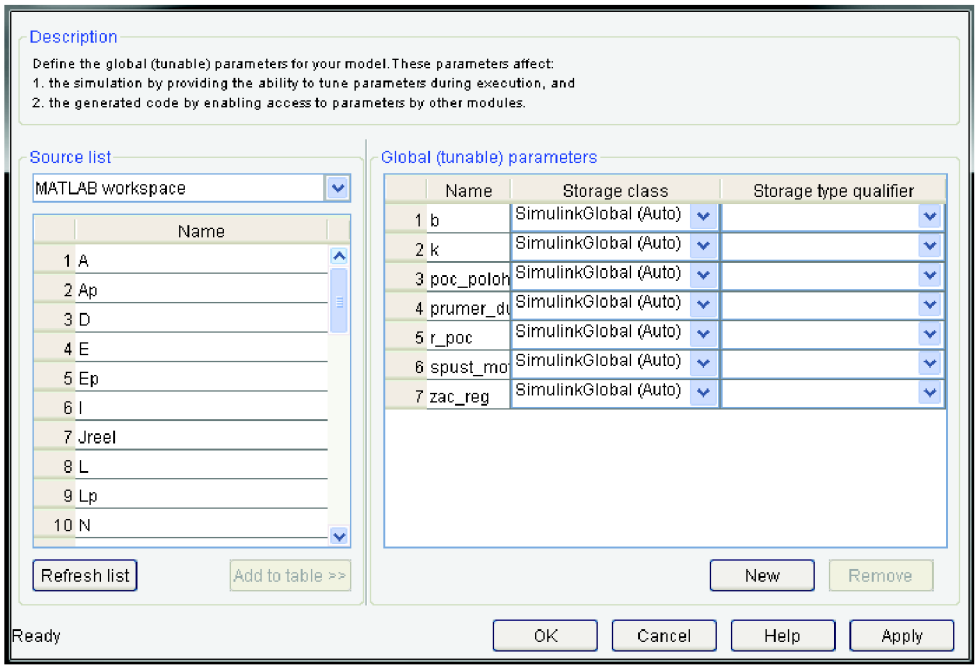
ImportedExternPointer: Import parametru jako ukazatel z externího kódu. Možné ladění parametru během spuštění.



Obr. 7-1 : Zapnutí Inline parametrů v nastavení simulace

Nastavení inline parametrů a výběr, které mají být zvoleny jako inline nalezneme v nastavení simulace v záložce „Optimization“, v podsekci „Signals and Parameters“ (viz obr. 7.1). Po zvolení (zaškrtnutí) volby „Inline parameters“ klikneme na tlačítko

„Configure...“, tím se zobrazí nové okno (obr. 7-2), kde zvolíme z hlavního pracovního prostoru parametry (z výběrového menu zvolíme možnost „MATLAB workspace“, pokud již není nastaveno) a nastavíme vybrané parametry jako inline. Postup přidání parametrů je jednoduchý - zvolíme parametr a poté klikneme na tlačítko „Add to table“. Poté přichází na řadu zvolit typ proměnné (StorageClass, které jsou popsány výše) v prostředním sloupci a nakonec nastavit, zda proměnná má být konstanta či nikoliv, to se provádí v posledním sloupci.



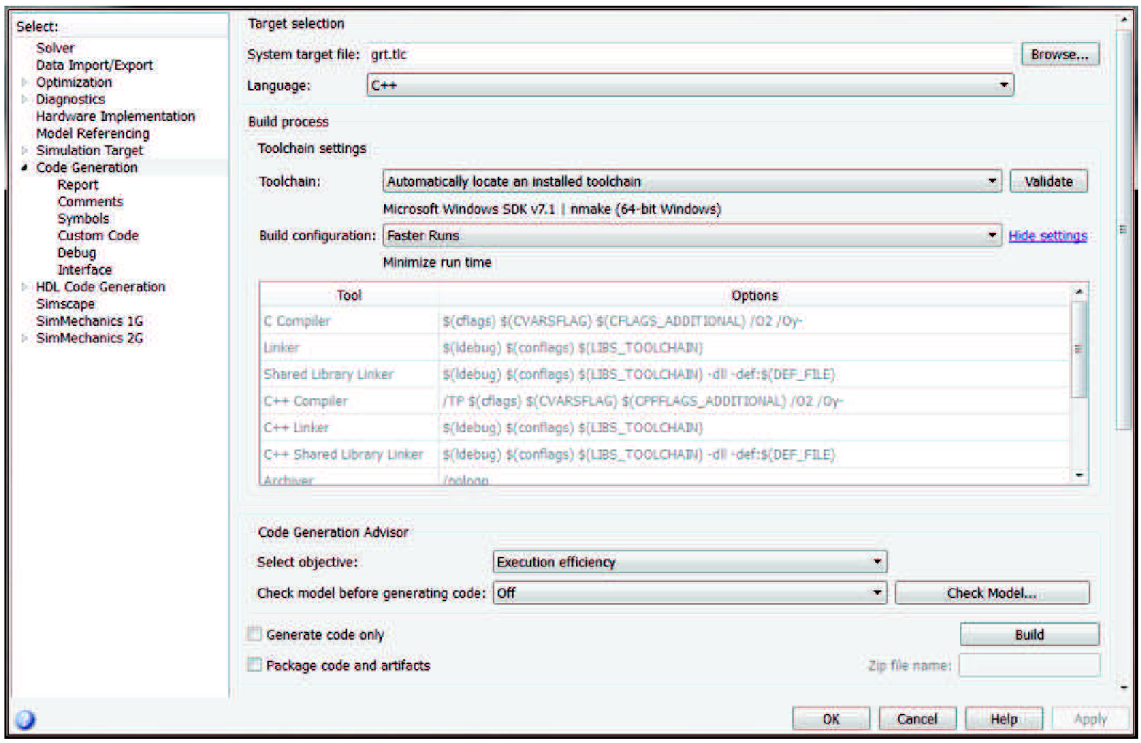
Obr. 7-2 : Nastavení parametrů jako Inline

7.4 Vytvoření samostatného modelu

7.4.1 Nastavení generování kódu

K vytvoření samostatného modelu v simulačním prostředí Simulink slouží záložka v horním řádkovém menu pod jménem „Code“. Předtím než spustíme vygenerování kódu (building) je potřeba nejdříve nastavit průběh a parametry generování kódu. Otevřeme nastavení parametrů simulace (Model Configuration Parameters), v levém bloku naleznene záložku „Code Generation“, zobrazí se následující obrazovka, viditelná na obrázku 7-2. První co nastavíme je „System target file“ (umístěný v okně úplně nahoře), zde nastavujeme typ vygenerované aplikace, pro samostatně spustitelnou aplikaci na standardních platformách zvolíme možnost (tlačítkem Browse) „grt.tlc“, pokud je na výběr více možností, zvolíme „Generic Real-Time Target“. Poté vybereme, zda má být výsledný kód generovaný v jazyku C nebo C++.

Další důležitou věcí, kterou je potřeba nastavit je Toolchain. Toolchain je sbírka nástrojů potřebných pro kompilaci, linkování, a spuštění kódu na zvolené platformě. Toolchain se skládá z několika nástrojů, jako je kompiler, linker aj. Z výběrového menu zvolíme nainstalovaný vývojový systém pro generování samostatného spustitelného souboru, pokud jsi nejsme jisti výběrem, použijeme tlačítko „Validate“. Při zvolení určité možnosti a použití tlačítka „Validate“ zjistíme, zda byl výběr správný – zobrazí se nové okno s možnými výsledky kontroly; „Failed“ (Neúspěšný) nebo „Passed“ (Úspěšný). Pokud se zobrazí „Failed“ (Chyba), volíme následující postup - ve výběru volíme „Automatically locate an installed toolchain“ a znova použijeme tlačítko „Validate“, tím se automaticky zvolí správný naistalovaný toolchain. Je možné, že i při nastavení této možnosti nebude validace úspěšná, v tom případě je potřeba doinstalovat nový toolchain. [23]

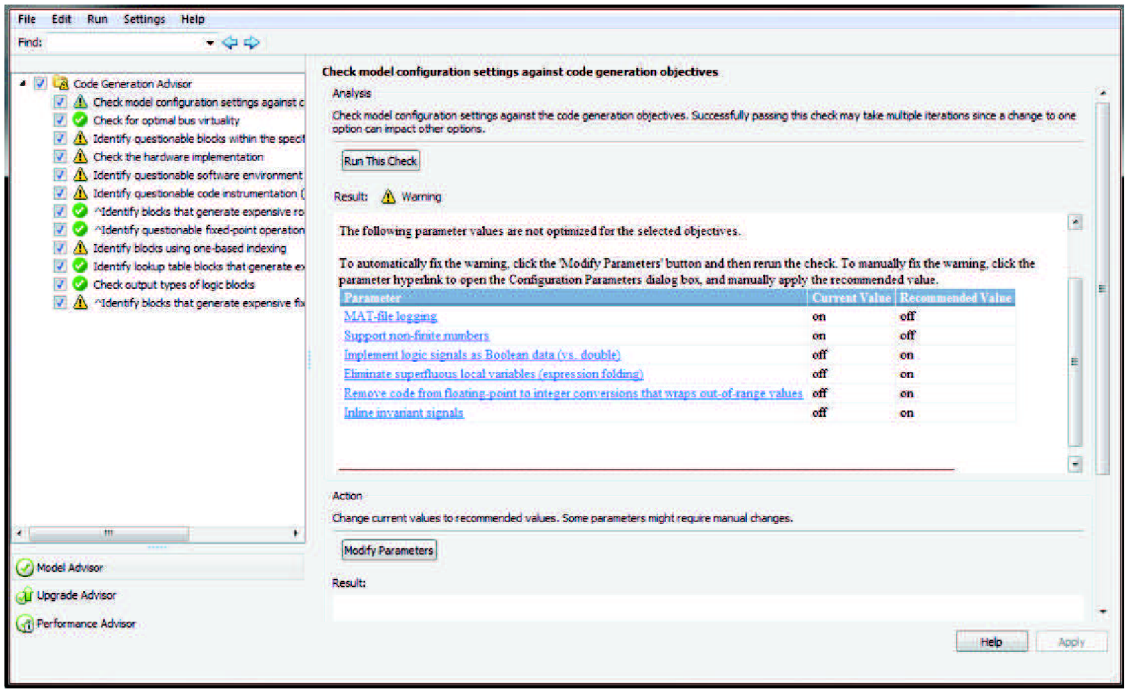


Obr. 7-3 : Vzhled obrazovky nastavení Code Generation

Dalším krokem je volba, zda generovaný kód má být optimalizován na rychlost generování kódu nebo optimalizaci rychlosti modelu (minimalizace délky běhu modelu). Na výběr jsou dvě možnosti „Faster Runs“ a „Faster builds“. Vhodné je zvolit při odlaďování modelu možnost rychlejšího generování kódu a poté po ukončení fáze odlaďování vybrat druhou možnost pro rychlejší chod modelu. V závislosti na rozsáhlosti modelu může building trvat i desítky minut, a to hlavně při vytváření kompletního kódu. V dolní polovině okna nalezneme blok „Code Generation Advisor“, ten slouží k debugování a zvyšování efektivnosti modelu, při prvním pokusu o build

modelu je vysoká pravděpodobnost výskytu chyb. Před prvním spuštění je doporučeno použít tento nástroj v módu „Debuging“, který dokáže zjistit příčiny chyb a dokonce je jednoduchým způsobem odstranit. I zde platí, že tato akce trvá v závislosti na komplexnosti modelu, proto je možné zvolit jen část (subsystém) ke kontrole. Na následujícím obrázku (obr. 7-4) je zobrazen vzhled „Code Generation Advisor“ v módu zvýšení efektivnosti (Execution efficiency). Můžeme také vidět, že lze velice jednoduše upravit parametry nastavení simulačního modelu na doporučenou hodnotu, k tomu slouží tlačítko „Modify parameters“. Jedním stiskem tohoto tlačítka se automaticky nastaví parametry simulace na (dle jeho) správné hodnoty. Avšak musíme dát velký pozor na to, které parametry vyhodnotil jako špatně nastavené, a zda neměníme některé parametry na hodnoty, které nám nevyhovují – např. v tomto případě (na tomto obrázku 7-4) vidíme, že algoritmus vyhodnotil nastavení parametru “MAT-file logging“ na hodnotu „off“. Pokud bychom unáhleně pokračovali, nastavíme parametr sice na doporučenou hodnotu, ale nevhodnou. V našem případě (resp. v této práci) je žádoucí, aby se výsledky ukládaly, na úkor toho, že model nebude efektivně optimalizovaný a tím i rychlý, i když to je právě funkcí tohoto nástroje v tomto módu.

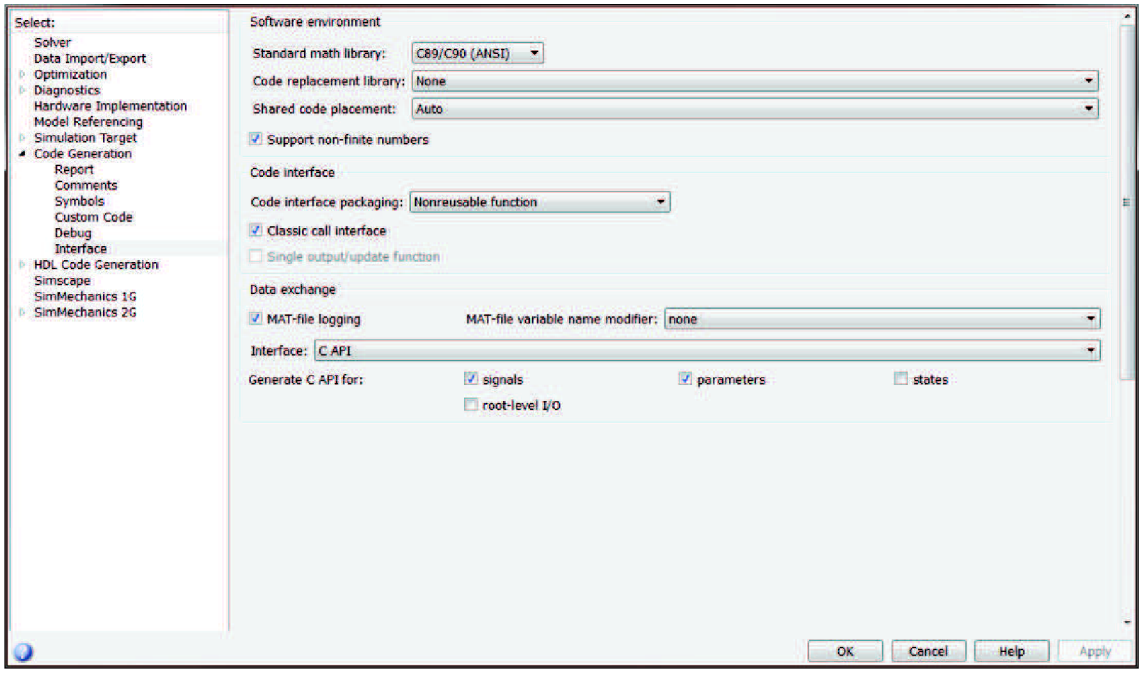
Dalšími nastavitelnými možnostmi na této stránce jsou „Generate code only“. Po zvolení této možnosti se generuje jen samotný kód, bez spustitelného souboru (soubor s příponou .exe). Další možností je „Package code and artifacts“, který zabalí vygenerovaný kód a knihovny do jednoho souboru.



Obr. 7-4 : Code Generation Advisor v módu Execution efficiency

7.4.2 Nastavení interface

Důležitou částí, ve které je potřeba nastavit parametry je „Inteface“ (rozhraní), které také nalezneme v záložce „Code Generation“. Zde se nastavuje rozhraní pro výměnu dat mezi modelem a externím vstupem. Pokud nezvolíme žádný typ rozhraní z výběrového menu (umístěné v bloku s názvem „Data exchange“), nebude žádná možnost měnit proměnné v simulačním modelu, resp. vytvořený model bude po opětovném spuštění končit vždy se stejnými výsledky. V případě této diplomové práce je zvolena možnost C-API, přičemž dalšími možnostmi jsou „External mode“ a „ASAP2“. Poté zvolíme možnosti „Signals“ a „Parameter“ pod tímto výběrovým menu, popř. i jiné. Po tomto kroku bude možné nastavovat proměnné modelu, vytvořeného jako samostatná aplikace. Je tedy potřeba vytvoření vlastního inicializačního kódu. Vlastní inicializační nebo ukončovací kód je možné psát v záložce „Custom Code“, a to v jazyku C nebo C++, dle toho jakou možnost jsme zvolili na stránce „Code generation“, viz. výše).



Obr. 7-5 : Nastavení interface

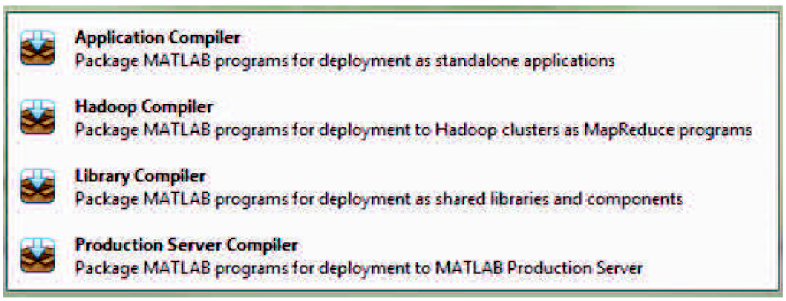
Je potřeba napsat kód pro druhou stranu přenosu dat mezi GUI a modelem - inicializační kód pro načtení dat ze soubor. Ukládání nastavených parametrů do souboru je popsán v kapitole 6.3.1 *Výměna dat mezi GUI a modelem*. Kód je psaný v jazyku C, při vložení vlastního kódu a následným vygenerováním modelu se tento kód zkopíruje doprostřed automaticky vygenerované struktury v souboru *jmeno_modelu.cpp*. Pokud jsou potřebné některé knihovny, ať už standardní nebo vlastní, je nutné je zapsat do „Header files“, popř. „Source files“.

Data s hodnotami parametrů jsou v souboru jsou uloženy ve specifickém tvaru. V tomto programu je potřeba nejdříve otevřít textový soubor (standardně funkcí *fopen*). Dále v cyklu *while* je kontrolován konec souboru pro ukončení tohoto cyklu, při zjištění oddělovacího znaku jsou postupně ukládány parametry (celkově jich lze změnit ve stand-alone aplikaci 15) do jedinečných proměnných. Parametry jsou ukládány a ponechány v těchto proměnných, a to z důvodu lehké dohledatelnosti, přehlednosti a také pro jejich možnou numerickou kombinaci. Důležitý krok přichází v propojení proměnné v kódu s proměnou v modelu. Speciální měnitelné proměnné ve standalone aplikaci, tzv. tunable a inline parameters (postup nastavení byl vysvětlen v kap. 7.3), jsou uloženy ve struktuře *jmeno_modelu_P*. Je nutné nastavit proměnné, které chceme měnit, jako inline. Má to ale jistá omezení – proměnné v SimScape blocích nelze ve stand-alone aplikaci měnit. I proměnné zapsané v blocích „Fcn" nejsou v této struktuře viditelné a nelze měnit, i když jsou nastavené jako tunable. Je tudíž nutné matematický vztah vyjádřený v bloku „Fcn" přestavět do formy skládající se jen z bloků „Gain", „Constant", „Product" atd.

7.5 Vytvoření samostatného uživatelského rozhraní

Pomocí Matlab Compileru lze vytvářet stand-alone aplikace a sdílené knihovny z programů vytvořených v Matlabu, většinou z výpočetních a grafických funkcí včetně vlastních GUI aplikací. Po úspěšné kompilaci je možné distribuovat aplikace, pro jejichž spuštění je nutný jen Matlab Compiler Runtime (MCR), dodávaný s aplikací. Použitím Matlab Compileru v rámci komerční licence lze aplikaci zcela libovolně šířit. V rámci školní licence lze aplikaci šířit pouze bezplatně, přičemž musí být zajištěno její nekomerční (školní/akademické) využití. [24]

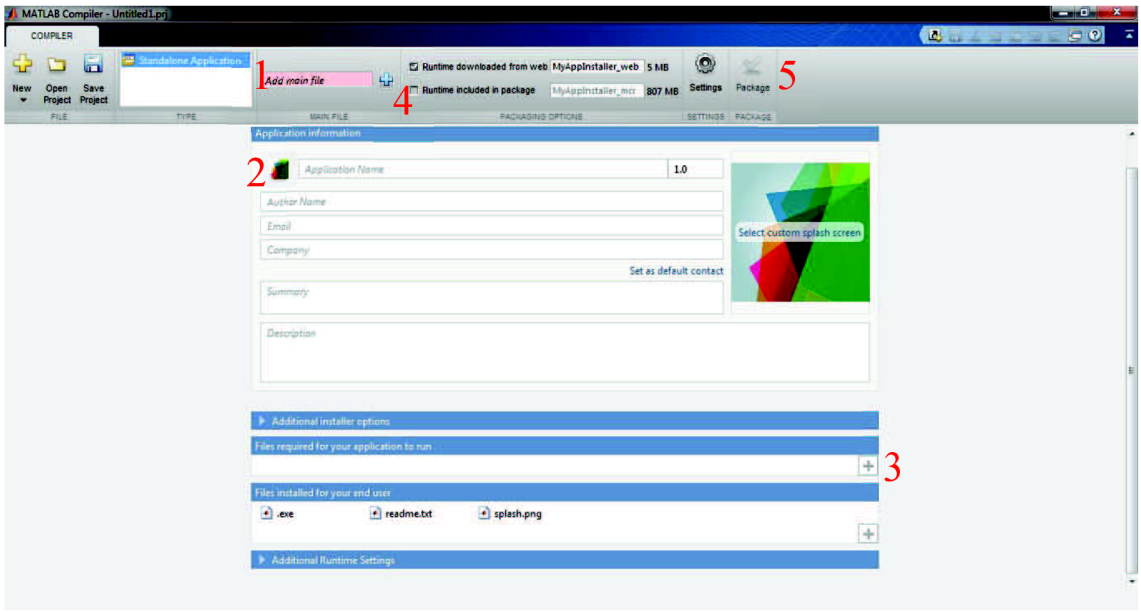
Po vytvoření uživatelských rozhraní, jejich odladění a zajištění odchytávání výjimek (vyřešení havarijních stavů, ošetření běhu při neexistujících datech atd.) přistoupíme k jejich kompilaci. Zadáním *deploytool* do příkazového řádku vyvoláme průvodce pro vytváření standalone aplikací. Zobrazí se následující volba (viz obr 7-3), kde ze zobrazených možností vybereme „Application Compiler".



Obr. 7-6 : Výběr typu aplikace ke kompilaci

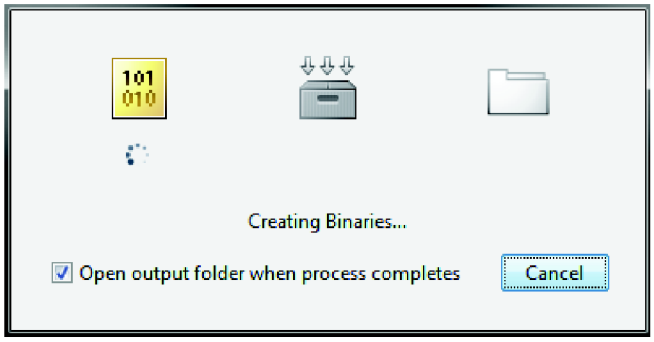
Poté se zobrazí hlavní obrazovka pro nastavení kompilace (obr. 7-7). Jako první krok je vhodné projekt uložit, pro budoucí opravy nebo úpravy nebudeme muset znovu nastavovat celou kompilaci. Nejdůležitějším krokem je nastavení hlavního souboru (1), tedy M-file vytvoření při založení GUI (kódová část). Část ve které jsme vytvářeli grafické rozhraní bude přidáno až poté a vysvětleno níže. Poté je nutné zvolit jméno výsledné aplikace (2), verzi aplikace a popř. i další informace – jméno autora, email, společnost, shrnutí účelu a funkce aplikace a její popis. Lze i změnit ikonu aplikace.

Pokud GUI používá některé externí soubory nebo funkce napsané v samostatných M-filech, je potřeba je zabalit spolu s hlavním souborem GUI, kliknutím na „plus“ (3). Tím se otevře se souborový průvodce, ve kterém zvolíme soubory potřebné pro správný běh aplikace. Při tomto kroku musíme bedlivě zvolit všechny soubory, jinak aplikace nebude pracovat správně, případně při spuštění havaruje. Hned pod tímto je úsek nazvaný „Files installed for your end user“, zde můžeme přidávat soubory, které budou pro konečného uživatele zobrazeny a bude si je moct prohlédnout nebo přečíst. Tato možnost je v této práci využita také pro přibalení potřebných souborů pro externí chod programu - tím je myšleno zkompilovaný simulační model a k tomu potřebné soubory. Dále také soubory s naměřenými daty určené k importu do GUI (průběh polohy tanečnice a rychlost hlavního motoru), které slouží k nahlédnutí a zjištění požadovaného formátu pro uživatele, současně s těmito daty je možný test chodu programu. Do této části se také mohou přidat soubory typu dokumentace, návody k používání atd.



Obr. 7-7 : Hlavní obrazovka pro nastavení kompilace

V tuto chvíli je nejdůležitější část nastavena. Následujícím krokem je zvolit způsob distribuce MRC, na výběr jsou dvě možnosti. Přibalení k aplikaci odkaz pro stáhnutí MRC z internetu, což je ta méně náročná volba vzhledem k velikost výsledné aplikace určené pro koncové uživatele. Druhou možností je přibalení již kompletního MRC, které stačí jen na počítači koncového uživatele nainstalovat, avšak vzhledem k velikosti celého MRC je tato volba volena při možnosti fyzického předání pomocí přenosného média. Po nastavení povinných parametrů se zvýrazní ikona pro kompilaci aplikace (5). Následuje proces kompilování.



Obr. 7-8 : Proces kompilování

8 IDENTIFIKACE PARAMETRŮ

Tato kapitola se věnuje identifikaci parametrů z naměřených dat na stroji. Účelem je zjištění vlastností/parametrů materiálu, než jsou běžně používané, nebo o jiných materiálových rozměrech. Vlastní algoritmus identifikace parametrů je minimalizace kriteriální funkce v n-rozměrném prostoru algoritmem zvaným „Nelder-Mead Method". Vlastní identifikace bude probíhat na třech parametrech, a to dva parametry náhradního modelu materiálu, skládající se z paralelní kombinace lineární pružiny (Linear Spring) a lineárního tlumiče (linear Damper) a třetího parametru představující počáteční pozici tanečnice. Po úspěšné identifikaci pak lze takto získané parametry zavést do simulačního schématu a pokračovat v ladění parametrů regulátoru tahu, popř. testování výkonů a parametrů odvíjecího nebo navíjecího motoru, k čemuž slouží uživatelské rozhraní „Ovládání s pomocí Matlabu", popř. při ladění pouze parametrů regulátoru tahu lze použít rychlejší způsob, a to „Ovládání stand-alone".

8.1 Nelder-Mead Simplex Method

Tato metoda je jedna z nejpoužívanějších přímých numerických metod k nalezení extrému (globálního minima) vícerozměrného neomezeného prostoru. Nelder-Mead tedy spadá do obecné skupiny přímých metod a byla navržena Johnem Nelder a Rogerem Mead v roce 1965. Při použití v podstatě nejsou potřeba žádné teoretické výpočty, tato skutečnost může být právě příčinou velké oblíbenosti této metody. Metoda Nelder-Mead minimalizuje skalární hodnotu nelineární funkce o ‚n' reálných proměnných při použití pouze funkčních hodnot, a to bez jakékoliv znalosti její derivační funkce. [25]

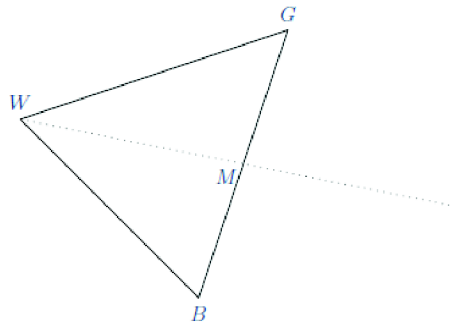
Metoda využívá koncepci simplex, což je speciální mnohostěn o ‚n+1' vrcholů v ‚n' rozměrech. Simplex je n-rozměrné zobecnění trojúhelníku. Metoda tedy pracuje s ‚n+1' body v ‚n' rozměrném prostoru a je založena na postupné modifikaci simplexu. V každém kroku dochází k jednoduchým transformacím simplexu pro nalezení bodu s nižší hodnotou, než je stávající hodnota vrcholů simplexu. [26,27]

Během jedné iterace může dojít k:

- převrácení (reflexion)
- protažení(expansion)
- zkrácení (contraction)
- sražení(shrink)

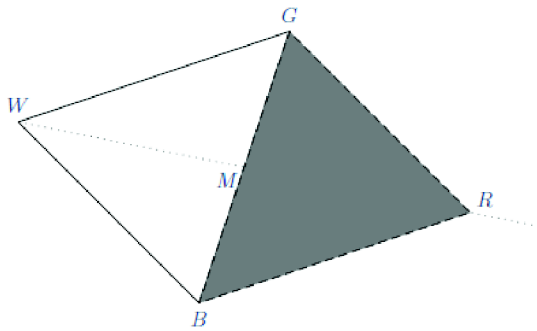
8.1.1 Algoritmus metody

Vytvoříme simplex ve dvourozměrném prostoru – trojúhelník WBG, kde W je jeho nejhorší bod, resp. bod s největší hodnotou kriteriální funkce.



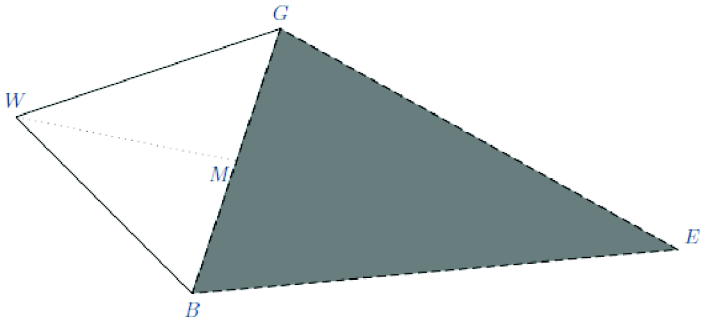
Obr. 8-1 : Základní simplex WGB [26]

Provedeme převrácení trojúhelníku (určíme bod R), je-li hodnota kritéria v bodě R lepší než v bodě A, zjistili jsme, že pokračujeme správným směrem a provádíme protažení (obr. 8.3). Pokud je hodnota v bodě R stejná jako v bodě A, provedeme zkrácení (obr. 8.4).



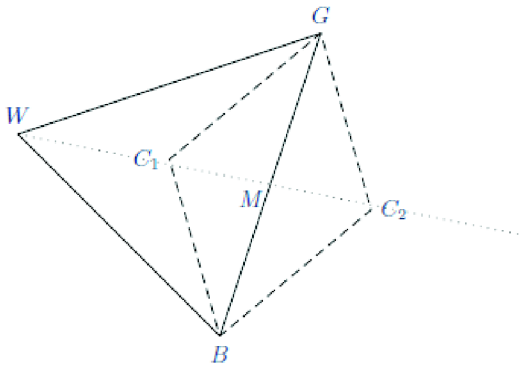
Obr. 8-2 : Převrácení, vznik simplexu RGB [26]

Pokud tedy víme, že pokračujeme správným směrem, zvolíme bod E, který je ve stejném směru jako R, ale jeho délka bude E = 2R-M. Srovnáme hodnoty kriteriální funkce v bodech R a E a zvolíme lepší (menší) hodnotu. Bod s menší hodnotou je novým vrcholem trojúhelníku. Nový simplex bude RGB nebo EGB.



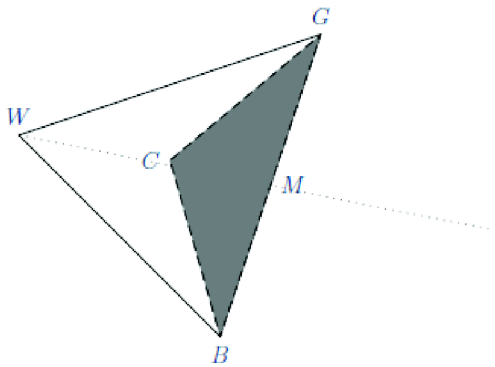
Obr. 8-3 : Protaženi, vznik simplexu EGB [26]

Pokud jsme zjistili, že hodnota v bodě R je stejná jako v bodě A, provádíme zkrácení (viz. obr 8.4). Zjistíme hodnotu kritéria v bodech C_1 a C_2, vybere bod s menší hodnotou a vytvoříme nový simplex C_1GB nebo C_2GB.



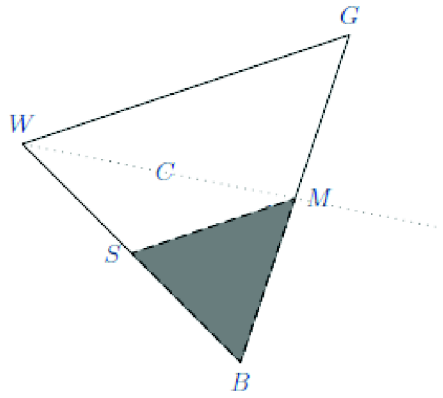
Obr. 8-4 : Zkrácení, vznik simplexu C_1GB nebo C_2GB [26]

Po předchozím kroku zkrácení vznikne např. nový simplex CGB (který vidíme na následujícím obrázku 8-5). V tomto kroku zjistíme hodnoty v bodech B a G, pokud v některém z těchto bodů zjistíme větší hodnotu kriteriální funkce než je v bodě C, provedeme sražení, (obr. 8.6).



Obr. 8-5 : Zkrácení, vznik simplexu CGB [26]

Na následujícím obrázku je příklad sražení – v bodě B byla zjištěna menší hodnota kritéria než v bodě C a byl vytvořen nový simplex BSM



Obr. 8-6 : Sražení, vznik simplexu BSM [26]

8.2 Identifikace v Matlabu

Minimalizační algoritmus Nelder-Mead je v Matlabu již zakomponován a je známý pod funkcí *fminsearch*, v této kapitole jsou popsány její vlastnosti, možnosti, způsob volání a použitelnost. Jak jsem již zmínil, tuto funkci využijeme pro identifikaci třech parametrů za použití simulačního schématu vytvořeného v této práci, ale s několika nutnými úpravami. Uživatelské rozhraní potřebné pro snadné ovládání koncovým uživatelem je popsáno a vysvětleno v kapitole 6.4 *Ovládání identifikace parametrů*. Předpokládejme tedy, že všechny potřebné věci, jako model, měřená data atd. máme již k dispozici a zde si jen vysvětlíme problematiku samotného algoritmu minimalizace kriteriální funkce. Dokumentace funkce *fminsearch* je dostupná na [26].

Prvním krokem je potřeba zjistit kolik, a jaké parametry chceme identifikovat. V této práci je to již specifikované, ale obecně je tento algoritmus nejpoužívanější k nalezení minima funkce pro dvě proměnné, kde ve většině případech dobře a celkem velice rychle konverguje ke správnému výsledku. Použitelný je také pro tři proměnné, může se ale někdy stát, že algoritmus diverguje a poté je potřebný restart této metody a popř. nastavení jiných počátečních hodnot. Pro více než tři proměnné je tento algoritmus nedoporučený. Avšak na druhou stranu, výhodou této metody je jednoduchá implementace a fakt, že nevyžaduje vyhodnocení derivační funkce. Kriteriální funkce je v této práci Kvadratické integrální kritérium – což je kvadrát rozdílu změřené polohy tanečnice na stroji a průběh polohy tanečnice z modelu integrován přes celou dobu simulace. [27]

Základní předpis funkce je následující; *x = fminsearch(function,x0,options)*, kde první vstup *function* je „problém“, tedy funkce nebo model pro který budeme provádět minimalizaci. V této funkci musejí vystupovat hledané proměnné, jejich počet se musí shodovat s počtem proměnných ve druhé vstupu *x0*, ve kterém nastavujeme počáteční hodnoty identifikovaných proměnných (uložené v řádkové vektoru). Posledním vstupem je nastavení minimalizačního algoritmu. Je možné nastavovat tyto parametry;

Display – způsob vypisování výstupu do Command Line; ‘line‘ – zobrazuje výstup po každé iteraci, ‘off‘ – žadný výstup, ‘notify‘ – zobrazí výstup jen pokud algoritmus nekonverguje.

FunValCheck – Kontroluje, zda hodnoty kriteriální funkce jsou platné; ‘on‘ – zobrazuje chybu, když funkce vrátí hodnotu, která je komplexní nebo NaN (not a number), ‘off‘ – nezobrazí chybu.

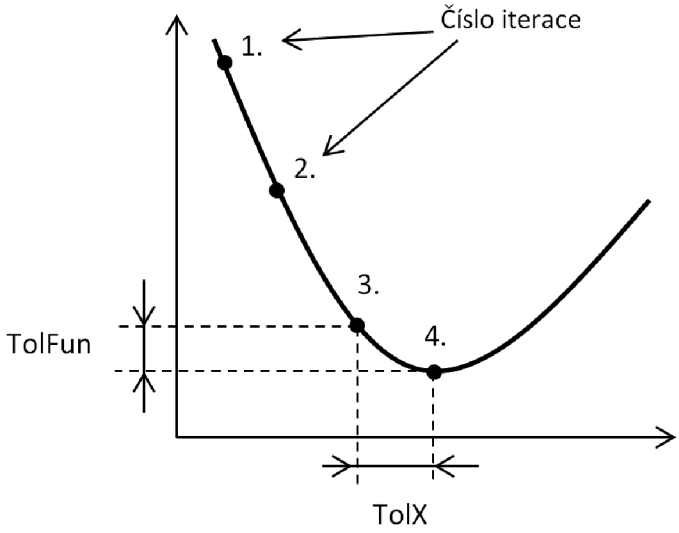
MaxIter – Maximální povolený počet iterací, které proběhnou, doporučená hodnota je 200*pocet_promennych.

MaxFunEvals – Maximální povolený počet výpočtů funkce, doporučená hodnota je 200*pocet_promennych.

TolFun/*TolX* - ukončovací tolerance změny hodnoty krit. funkce/parametrů, zadaná hodnota musí být pozitivní a skalár. Výchozí hodnota je 1E-4, *fminsearch* se ukončí, pokud jsou splněny obě tolerance - TolFun i TolX.

8.2.1 Tolerance a ukončovací podmínky

Počet provedených iterací minimalizační funkce, mimo jiné, z velké části závisí na nastavení ukončovacích podmínek. Tyto kritéria zahrnují prahové odchylky (tolerance), pokud změna parametrů nebo výstupu kriteriální funkce oproti předchozí hodnotě je menší než nastavená hodnota tolerancí *TolFun* a *TolX* (grafické znázornění je na obr. 8.7), je optimalizační proces zastaven a poslední nastavené parametry jsou ty správné, hledané. Při těchto hodnotách parametrů je nalezeno globální minimum kriteriální funkce. Na nastavení těchto hodnot tolerancí je potřeba dát velký důraz, pokud jsou nastavené na vysokou hodnotu, získané parametry jsou nepřesné, v druhém případě, při nastavení velmi malých hodnot tolerance se mohou provádět zbytečné iterace a výsledný algoritmus nemusí v požadovaném čase konvergovat ke konečnému výsledku.



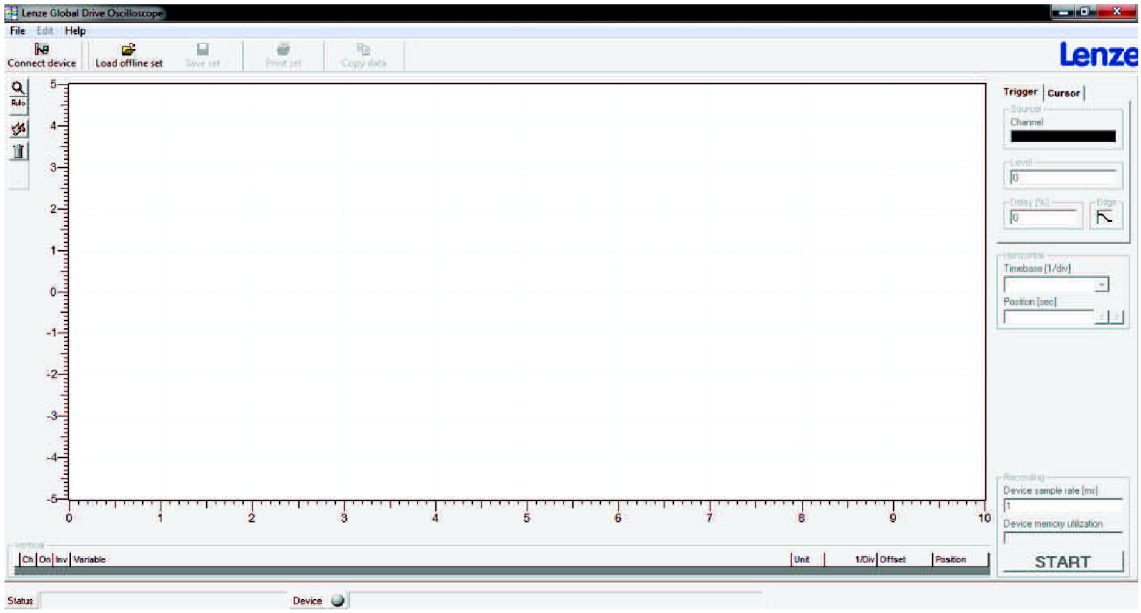
Obr. 8-7 : Grafické znázornění pojmů TolX a TolFun

Po ukončení optimalizačního procesu je potřeba zjistit výsledek identifikace; zda byla ukončena správně, popř. jaká ukončovací podmínka byla dosažena a jaké jsou výsledné parametry. Další možnosti zápisu funkce *fminsearch* je dle počtu výstupních proměnných. Základní tvar je „*x = fminsearch(fun,x0)*“, kde výstup z takto zapsané funkce je vektor, ve kterém jsou uložené zjištěné parametry. Pokud však chceme proces identifikace zautomatizovat (např. použitím v GUI), je potřeba získat více informací. Možný zápis je např. *[x,fval,exitflag] = fminsearch(...)*, kde výstup *x* již známe. Dalším je *fval* – v této proměnné je hodnota kriteriální funkce při výsledných parametrech v *x*, numerická hodnota v *exitflag* určuje, jakým způsobem byla ukončena funkce *fminsearch*. Pokud *exitflag = 1*, funkce *fminsearch* úspěšné dokonvergovala při daných hodnotách v *x*, pokud *exitflag = 0*, bylo dosaženo maximálního počtu iterací nebo volání funkce, pokud *exitflag = -1*, proces optimalizace byl externě ukončen.

Funkce *fminsearch* často dokáže zvládnout i diskontinuity, a to zejména v případě, pokud se nevyskytují v blízkosti řešení. *Fminsearch* pracuje pouze s reálnými čísly, to znamená, že parametry se musí skládat pouze z reálných čísel, samotná funkce také vrací pouze reálná čísla. Komplexní proměnné musejí být rozděleny do reálné a imaginární části.

9 MEŘENÍ NA REÁLNÉM STROJI

Měření bylo prováděno na odvíjecí jednotce stroje Optima O10_14 820 – 8 EG. Vyčítání dat zajišťoval program Global Drive Oscilloskope (GDO) od firmy Lenze. Program komunikuje se přes sběrnici RS232 nebo RS485, vyčítání je omezeno velikostí mezipaměti řídící jednotky motoru. Lze zvolit jakýkoliv počet parametrů s periodou vyčítání minimálně 30 µs nebo násobky minimální periody. Při překročení povolené velikosti mezipaměti je potřeba zmenšit periodu vyčítání nebo počet vyčítaných proměnných. Prostředí GDO je velice intuitivní a ovládání je velice jednoduché. Zjištění názvu požadované hodnoty k měření jsem zjistil u programátora programovatelných řídících automatů. Pokud jde o samotné řízení motorů, jméno proměnných nalézneme v datasheetu od firmy Lenze (Software Package – Template DancerControl), který je určen pro řízené pohonů ve velkoformátových tiskařských strojích. [15]



Obr. 9-1 : Vzhled pracovního prostředí GDO firmy Lenze

Měření na stroji se lišila v počátečních podmínkách; poloze tanečnice (krajní polohy), tím i rozdílný počáteční tah materiálu a požadovaný tah. Dalšími proměnnými parametry byly finální rychlosti, různé průběhy rychlostí, chybové stavy (přetržení materiálu) atd.

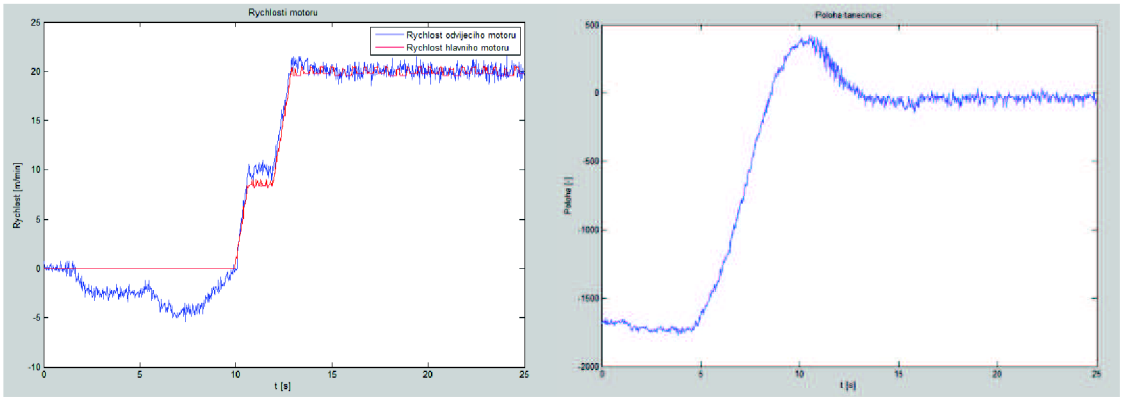
Naměřená data bylo potřeba upravit. Rychlost hlavního motoru byla převedena z dm/min na m/min, některé měřené hodnoty byly normovány na velikost vnitřních proměnných a musely být zpětně převedeny na původní hodnoty. Důvodem normování je, že měření analogové hodnoty. Rozsah parametrů je přizpůsoben velikosti vnitřnímu

rozsahu proměnných, hodnoty jsou normovány tak, aby výsledný rozsah byl 0-16384 (resp. signál je rozdělen na 16384 úrovní). Pro jednoduchost a přehlednost jsou zde vyobrazeny jen průběhy rychlostí a polohy tanečnice dvou měření.

9.1 Měření č.1

Potiskový materiál je ve všech případech stejný - LDPE, tloušťka 30 µm, šířka 850 mm a průměr dutinky 76 mm. Zde je změřen standardní stav při běžném rozjíždění stroje. Regulace tahu začala v čase 5 s, to je způsobeno tím, že stroj po zapnutí je pár vteřin v klidovém stavu. Pokud je materiál hodně povolen (nedotýká se válce tanečnice), je třeba vyčkat, než se materiál přitiskne k válci a poté je teprve možné měřit výchylku tanečnice. Hlavní motor se rozjel v čase 10 s. Ustálené nulové polohy tanečnice bylo dosaženo zhruba v 15 sekundách od spuštění měření, resp. 10 vteřin po zapnutí regulátoru tahu.

- Nastavený tah 75 N
- Maximální rychlost 20 m/min
- Počáteční průměr role 210 mm



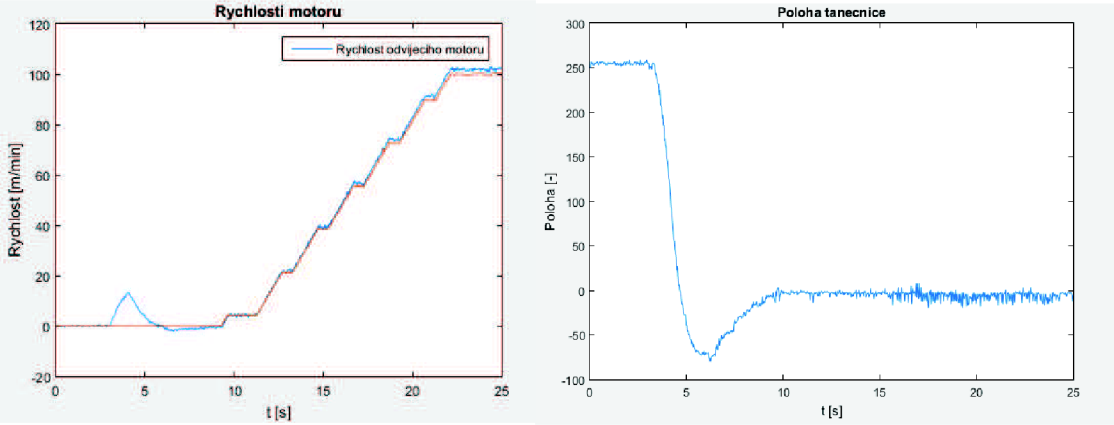
Obr. 9-2 : Průběh rychlostí a polohy tanečnice v měření č.1

9.2 Měření č.2

Při tomto měření byla počáteční poloha tanečnice v druhé (horní) krajní poloze, než při předchozích měření. Což znamená, že materiál byl natáhnut více, než byla požadovaná hodnota tahu. Tento fakt lze vidět na průběhu polohy tanečnice na obr. 9-2, kde průběh začíná v horní polovině grafu. Na obr. 9.3 je průběh rychlosti odvíjecího motoru na počátku v kladných hodnotách – motor se točil dopředu, aby dodat více materiálu a

povolil napnutí. V tomto měření je zajímavé to, že v čase cca 43 sekund byl potiskový materiál přerušen.

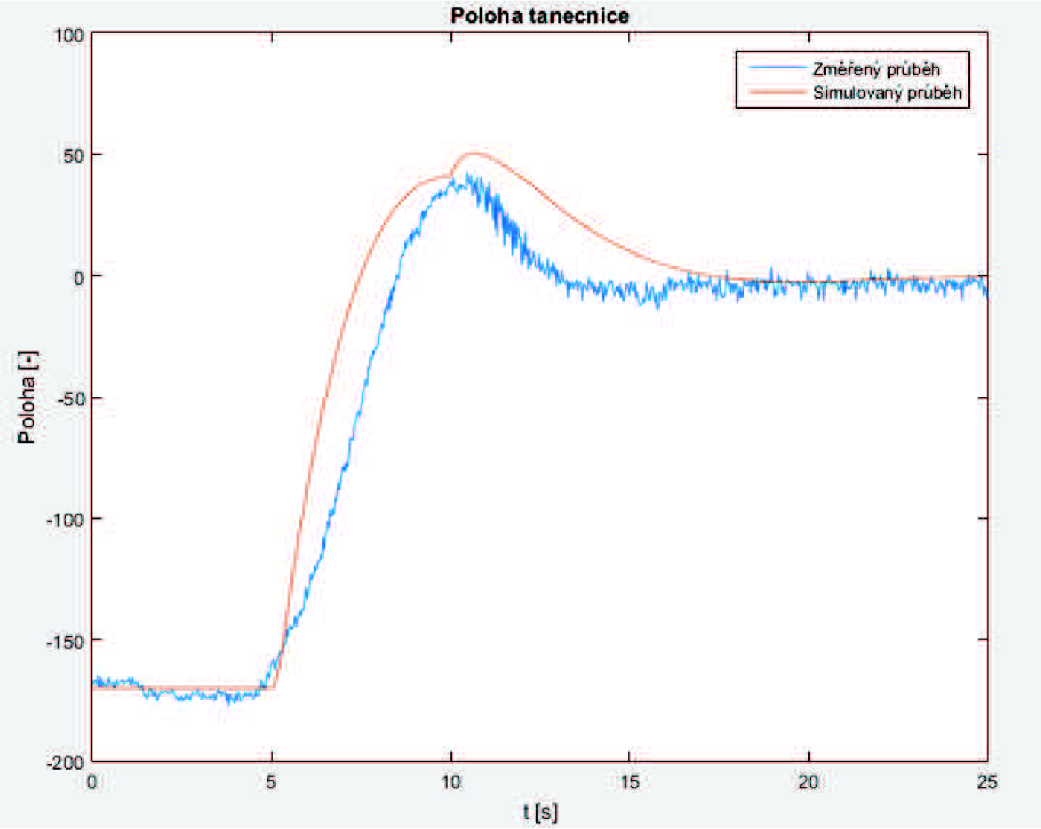
- Nastavený tah 20 N
- Maximální rychlost 100 m/min
- Počáteční průměr role 207 mm



Obr. 9-3 : Průběh rychlostí a polohy tanečnice v měření č.2

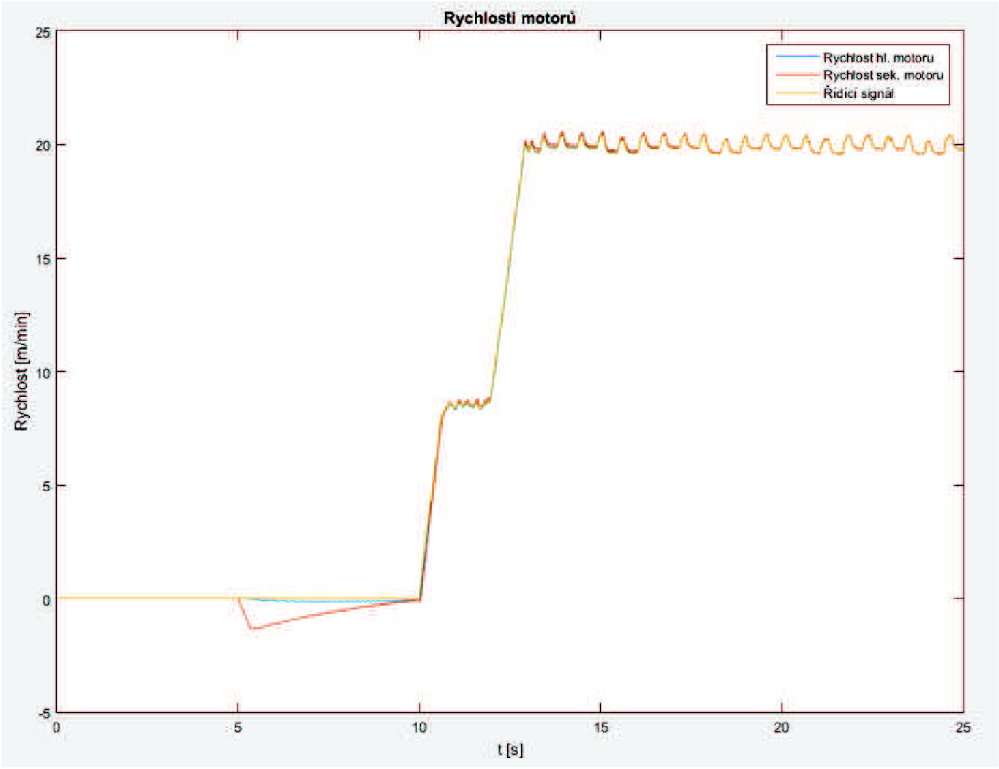
10 SROVNÁNÍ VÝSLEDKŮ

Ověření přesnosti modelu bylo provedeno na obou měřeních. Vstupem do modelového schématu byla rychlost hlavního motoru (obr. 9.2a a 9.3a). Na následujících grafech jsou srovnány průběhy rychlosti sekundárního (v tomto případě odvíjecího) motoru a polohy tanečnice.

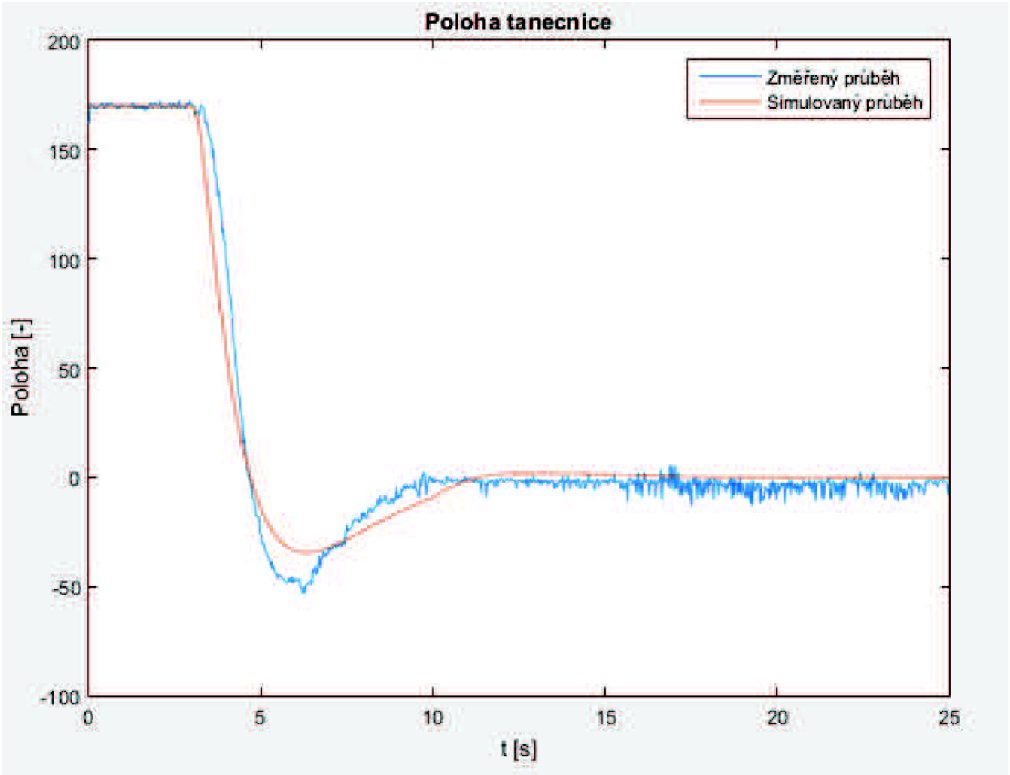


Obr. 10-1 : Srovnání průbehů polohy tanečnice v prvním měřením

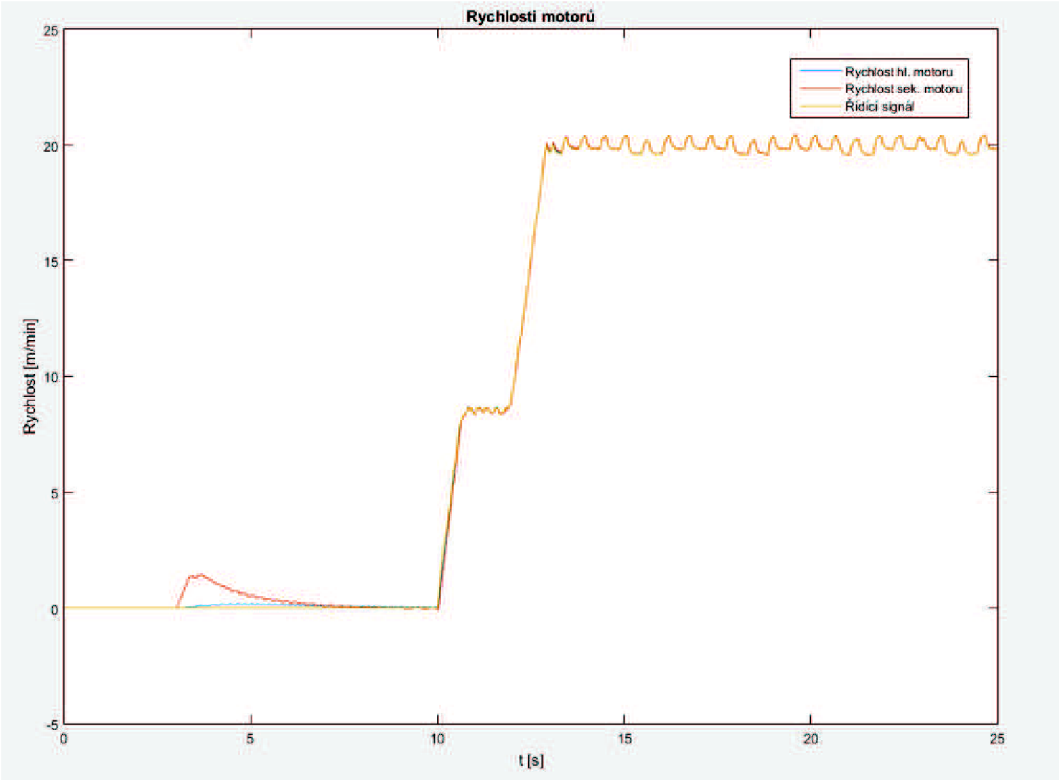
Srovnáním těchto průběhů vidíme, že průběhy se k sobě blíží, ale dynamika není zcela přesná. V čase 10 s by měl simulovaný průběh klesat exponenciální křivkou, přitom se poloha tanečnice ještě zvyší. To je způsobeno tím, že přesně v tomto čase se rozjíždí hlavní motor a sekundární motor na tuto situaci musí rychle zareagovat. Na následujícím obrázku 9-2 tuto skutečnost vidíme, sekundární motor v čase 5 s až 10 s reguluje polohu na nulovou hodnotu (v tomto případě napíná materiál). Pokud nestihne vyregulovat dostatečně rychle (než se rozjede hlavní motor), vzniká chyba regulace, která způsobuje zmíněný překmit v poloze tanečnice.



Obr. 10-2 : Průbehů rychlostí motorů v prvním měřením



Obr. 10-3 : Srovnání průbehů polohy tanečnice v druhém měřením



Obr. 10-4 : Průbehů rychlostí motorů v druhém měřením

Při srovnání průběhů v druhém měření dojdeme k podobnému závěru jako v předchozím případě – průběhy rychlosti sekundárního motoru jsou skoro identické z pohledu dynamického chování (srovnání obr. 10-4 a 9.3a) Průběh polohy tanečnice jsou podobné, avšak dynamika simulačního schématu z tohoto hlediska není zcela identická s reálným strojem

11 ZÁVĚR

Tato diplomová práce se zabývá principy akauzálního modelování v toolboxu SimScape simulačního programu Simulink a vytvořením uživatelského rozhraní pro ovládání vytvořených modelů. Cílem práce bylo vytvořit model odvíjecí a navíjecí části tiskařského stroje v prostředí SimScape tak, aby parametry modelu a chování co nejlépe vystihovalo reálný fyzický systém. Většina parametrů byla zjištěna z výkresové dokumentace stroje Optima O10_14 820 – 8 EG. Další parametry (např. materiálové vlastnosti aj.) byly zjištěny dostupnými a následně ověřenými zdroji a to hlavně pomocí internetu.

V této práci je popsáno akauzální modelování (které je poměrně novým a stále se rozvíjejícím přístupem), jaké výhody přináší a jak se liší oproti často používanému kauzálnímu modelování. Výsledkem této práce je také zjištění, že využití akauzálního modelování v průmyslu je možné a v budoucnu bude pravděpodobně stále více využíváno. Jeho výhodou je velice přehledné schéma modelu a jednoduchá záměna nebo úprava jednotlivých bloků. Naopak nevýhodou toolboxu SimScape je, že nelze externě měnit parametry těchto bloků při převedení schématu do stand-alone aplikace, a to ani před inicializací. Změnu parametrů v těchto blocích lze tedy provést jen v modelovém schématu v Simulinku a poté znova z modelu vygenerovat novou verzi stand-alone alikace.

Při vytváření tohoto modelu bylo potřeba vytvořit tři nové komponenty; blok reprezentující proměnnou setrvačnost, převodník rotačního pohybu na translační s proměnným průměrem a mechanický systém tanečnice (dancer_roller).

Vytvořená modelová schémata jsou poměrně rozsáhlá a komplexní, k tomu je potřebný odpovídající výpočtový čas. Použití motorů z toolboxu SimScape se tento čas ještě navyšuje. Použité motory jsou řízeny PWM měniči, které pracují na frekvenci 20 kHz. Dle Shannonova teorému by měla být frekvence vzorkování minimálně dvakrát tak větší, než maximální frekvence vyskytující se v systému. Musíme tedy volit frekvenci vzorkování simulačního modelu nejméně dvojnásobek této frekvence (40 kHz). Přepočtem na periodu vzorkování získáváme maximální hodnotu periody vzorkování 25 µs. Při nastavení maximální periody vzorkování a délky simulace 25 s se výpočtový čas pohybuje v jednotkách minut a více. Převedením schémat do stand-alone aplikací se tento čas zkrátí nejméně o polovinu. Díky tomuto faktu je obsluhování schémat dlouhodobější činností, nicméně použití a zjištění vlastností rozšiřujícího toolboxu SimScape bylo hlavní náplní této práce.

Modely a aplikace k jejich ovládání byly vytvořeny ve dvou formách. Uživatelské rozhraní „Ovládání s podporou Matlabu“ (a model) nebylo zkompilováno do samostatně spustitelné aplikace, protože jak již z názvu vyplývá, ovládání a průběh simulace probíhá v prostředí Matlabu. V tomto ovládání je možné nastavovat parametry i SimScape bloků, tudíž je mnohem pestřejší v počtu nastavitelných parametrů. Ostatní vytvořené modely jsou zastřešeny pod uživatelským rozhraním a zkompilované do jazyku C, spustitelné bez potřeby Matlabu a Simulinku, uživatel tedy vidí a ovládá jen uživatelské rozhraní. Ostatní věci, týkající se modelu a jeho struktury jsou pro něho skryty. Uživatelské rozhraní je intuitivní a ošetřeno na všechny možné havarijní stavy, ať už důsledkem neexistujících dat nebo při špatném zadání hodnot do textových polí. Uživatel je vždy informován o stavu prováděného procesu, stavu identifikace parametrů, aktuálního času simulace apod.

Srovnání naměřených a simulovaných průběhů je provedeno v kapitole 10. Získané průběhy z modelu odpovídají do určité míry naměřeným průběhům na stroji. Avšak některé parametry, které byly převedeny do modelu nejsou očividně přesně identifikovány. Důvodem může být výskyt skrýtých vazeb, které nelze změřit ani zjistit.

LITERATURA

[1] *SOMA Egineering: Profil společnosti* [online]. [cit. 2016-05-16]. Dostupné z: http://www.soma-eng.com/cs/profil-spolecnosti/firemni-hodnoty

[2] KAŠPÁROVA, L.: *Tisk z výšky; flexotisk*. Opava, 2011/2012. Střední škola průmyslová a umělecká, Opava.

[3] *Flexotiskový stroj s centrálním protitlakým válcem: Průvodní dokumentace a návod k použití stroje OPTIMA O10_17 820 – 8 WG*. Lanškroun, 2015, 205 str.

[4] KOFRÁNEK, J., MATEJÁK, M., PRIVITZER, P., TRIBULA, M.: *Kauzální nebo akauzální modelování : Dřinu lidem nebo dřinu strojům*. In: Konference Matlab [online]. Univerzita Karlova, Praha, 2008, s. 1-15 [cit. 2016-05-16]. Dostupné z: http://dsp.vscht.cz/konference_matlab/MATLAB08/prispevky/058_kofranek.pdf

[5] KOFRÁNEK, J., PRIVITZER, P., MATEJÁK, M., TRIBULA, M.: *Akauzální modelování: Nový přístup pro tvorbu simučních her*. [online]. Univerzita Karlova, Praha, 2008, s. 1-15 [cit. 2016-05-16]. Dostupné z: http://dsp.vscht.cz/konference_matlab/MATLAB08/prispevky/058_kofranek.pdf

[6] *Humusoft*: *Matlab* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.humusoft.cz/matlab/

[7] *Mathworks: SimScape* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/products/simscape/

[8] *Automa: Modelovani elektromechanické soustavy v prostředí Matlab - Simulink*. Praha: Automa-časopis pro automatizační techniku, s. r. o., 2008, 2008(8-9) [cit. 2016-05-16]. ISSN 1210-9592.

[9] KOCKOVÁ, H., HYNČÍK, L. *Modelování Matlabem 2: Doprovodný učení text ke cvičením*. Plzeň, 2007.

[10] *Mathworks: Simscape File Types and Structure* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/help/physmod/simscape/lang/about-simscape-files.html

[11] *Mathworks: Define Relationship Between Component Variables and Nodes* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/help/physmod/simscape/lang/defining-relationship-between-component-variables-and-nodes.html

[12] *Mathworks: Basic Principles of Modeling Physical Networks* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/help/physmod/simscape/ug/basic-principles-of-modeling-physical-networks.html

[13] *Mathworks: Choose a Solver* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/help/simulink/ug/types-of-solvers.html

[14] URBÁNEK, J. *Analýza citlivosti a nejistot v environmentálních distribučních modelech polutantů s využitím moderních IT*. Brno, 2008. Diplomová práce. Masarykova Univerzita, Fakulta informatiky, 81 s. Vedoucí práce Prof. RNDr. Jiří Hřebíček, CSc.

[15] *Global DriveSoftware Package: Winder Template DancerControl*. Lenze, 2003. 88 str.

[16] BLAŠKA, J., KRUMPHOLC, M., SEDLÁČEK, M.: *Využití grafického uživatelského rozhraní Matlabu ve výzkumu a výuce měření*. In: Konference Matlab [online]. České vysoké učení technické v Praze, Praha, 2008, s. 1-6 [cit. 2016-05-16]. Dostupné z: http://dsp.vscht.cz/konference_matlab/MATLAB08/prispevky/058_kofranek.pdf

[17] GANDHI, Varun. *ISPROCESS* [online]. 2013 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/matlabcentral/fileexchange/42805-isprocess

[18] VLASÁK, J. *Tvorba samostatně spustitelných aplikací v Matalbu*. Pardubice, 2010. Bakalářská práce. Univerzita Pardubice, Fakulta elektrotechniky a informatiky, 35 s. Vedoucí práce doc. Ing. František Dušek, CSc.

[19] *Mathworks : MATLAB Compiler* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/products/compiler/?refresh=true

[20] *Mathworks: Simulink Coder* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/products/simulink-coder

[21] *Mathworks: Tuning block parameters and model parameters* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/help/xpc/ug/tuning-block-parameters-and-model-parameters.html

[22] *Mathworks: Optimization pane signals and Parameters* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/help/simulink/gui/optimization-pane-signals-and-parameters.html

[23] *Mathworks: Register and Use Toolchain To Build Executable* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://www.mathworks.com/help/rtw/examples/adding-a-custom-toolchain.html

[24] JIRKOVSKÝ, J. *Matlab: Matematické výpočty, analýza dat a tvorba aplikací* [online]. Praha [cit. 2016-05-16]. Dostupné z: http://docplayer.cz/3811276-Matlab-matematicke-vypocty-analyza-dat-a-tvorba-aplikaci-jaroslav-jirkovsky-jirkovsky-humusoft-cz-www-humusoft-cz-info-humusoft.html

[25] LAGARIAS, J. C., REEDS J. A., WRIGHT M. H.,WRIGHT P. E.: Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions [online]. SIAM Journal of Optimization, Vol. 9 Number 1, pp. 112-147, 1998, s 1-6 [cit. 2016-05-16]. Dostupné z: http://people.duke.edu/~hpgavin/ce200/Lagarias-98.pdf

[26] BLAHA P. Modelování a identifikace: Přednáškové slide. Vysoké učení technické v Brně, 2015.

[27] ŠOLC, F., VÁCLAVEK P., VAVŘÍN P. Řízení a regulace 2. Vysoké učení technické v Brně, 2011.