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 A IMPLEMENTACE ŘÍDICÍHO ALGORITMU 3D TISKÁRNY

MODELLING AND IMPLEMENTATION OF CONTROL SYSTEM FOR 3D PRINTER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Martin Ševčík
AUTHOR

VEDOUCÍ PRÁCE — Ing. Miroslav Jirgl, Ph.D.
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Martin Ševčík ***ID:*** 161251
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Modelování a implementace řídicího algoritmu 3D tiskárny

POKYNY PRO VYPRACOVÁNÍ:

Úkolem je popsat, namodelovat, implementovat a provést PIL simulaci vybraných interpolačních algoritmů zajišťující CNC řízení pro 3D tiskárnu využívajícího S-křivky na základě provedené rešerše pomocí demonstračního přípravku obsahující kit STM32F4.

1. Proveďte rešerši interpolačních algoritmů pro CNC řízení a algoritmů implementujících S-křivky.
2. Namodelujte soustavu (3-osé CNC zařízení) v dostatečné komplexitě a vybrané algoritmy řízení v prostředí Matlab.
3. Implementujte namodelované algoritmy do kitu STM32F4, jenž je součástí řídicího systému 3D tiskárny.
4. Proveďte PIL a SIL simulaci jednotlivých interpolačních algoritmů v kombinaci s algoritmy implementujícími S-křivky.
5. Porovnejte výsledky a zhodnoťte.

DOPORUČENÁ LITERATURA:

GJELAJ, A. Intelligent CNC Programming of Machine Tools: Artificial Techniques. LAP Lambert Academic Publishing, 2016. 136 s. ISBN 978-3-659-88662-1.

Tyagi, A. MATLAB and SIMULINK for Engineers. OUP India, 2011. 492 s. ISBN 9780198072447.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 13.5.2019

Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.
Konzultant:

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

Abstrakt

Obsahem práce je modelování a implementace CNC řídícího algoritmu. V diplomové práci byl proveden popis problematiky řízení počítačem řízených strojů, rešerše interpolačních algoritmů pro řízení CNC strojů, namodelování vybraného algoritmu řízení s rychlostními rozběhovými profily s tvarem S-křivky a simulace vybraných algoritmů a rychlostních profilů v prostředí Matlab a Simulink.

Klíčová slova

Interpolační algoritmus, 3D tiskárna, krokový motor, S křivka, simulace, STM32F411RE, MIL, SIL, PIL

Abstract

The content of the thesis is the modeling and implementation of the CNC control algorithm. This master thesis contains the description of issues of computer-controlled machines, the research of interpolation algorithms for control CNC machines and the modeling of the selected control algorithm with motor S-curve shaped speed profiles and simulation of chosen algorithms with S-shaped speed profiles in Matlab & Simulink.

Key words

Interpolation algorithm, 3D printer, stepper motor, S-curve, simulation, STM32F411RE, MIL, SIL, PIL

Bibliografická citace mé práce:

Citace tištěné práce:

ŠEVČÍK, Martin. Modelování a implementace řídicího algoritmu 3D tiskárny. Brno, 2019. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/121398. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Miroslav Jirgl.

Citace elektronického zdroje:

ŠEVČÍK, Martin. Modelování a implementace řídicího algoritmu 3D tiskárny [online]. Brno, 2019 [cit. 2019-08-04]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/121398. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Miroslav Jirgl.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Modelování a implementace algoritmu 3D tiskárny 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: **5. srpna 2019**

……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Miroslavu Jirglovi, Ph.D. a stínovému vedoucímu Ing. Jakubu Armovi. za metodickou a odbornou pomoc a cenné rady při zpracování mé diplomové práce a dále děkuji všem, kteří se mnou práci konzultovali, zejména doc. Ing. Arnoštu Ševčíkovi, CSc.

V Brně dne: **5. srpna 2019**

……
podpis autora

Obsah

Úvod

Od počátku věků lidé hledali způsob, jak si svou práci ulehčit. Začali jednoduchými nástroji, které se vyvíjely do stále sofistikovanějších strojů. Velkým milníkem byl vynález parního stroje a s ním příchod první průmyslové revoluce, která masově odstartovala používaní strojů v průmyslu. Výroba se zrychlila a nároky na pracovníka se snížily, takže i člověk s minimem znalostí a zkušeností byl schopen díky strojům vyrobit věci, které by dříve nedokázal. Pro nás je však daleko důležitější jiný milník, a sice ten, který je nejčastěji datován od roku 1969 a přichází se zavedením počítačů do průmyslu. Také se označuje jako třetí průmyslová revoluce.

Od toho roku se začaly používat programovatelné logické automaty a průmyslové počítače a byl tak položen základ k automatizaci výrobních procesů v téměř všech průmyslových odvětvích. Stroje, které jsou takto řízené, se označují zkratkou CNC z anglických slov „computer numeric control“, v češtině se jim říká počítačem řízené stroje. Jejich obrovskou výhodou je jejich univerzálnost, kdy pouhou změnou programu nebo vstupních dat je možné na jednom stroji vyrobit nespočet různých dílů. CNC stroj pracuje nepřetržitě, nikdy se neunaví a výrobky, které z něj vycházejí, jsou téměř identické.

Počítačem řízené stroje dnes zvládají téměř všechny výrobní procesy, jako jsou frézování, řezání, vrtání, ohýbání, lisování a jiné. Spousta takových strojů je multifunkčních a jejich zaměření není pouze jednostrané, mohou tak kombinovat více výrobních procesů, čímž se šetří čas i peníze.

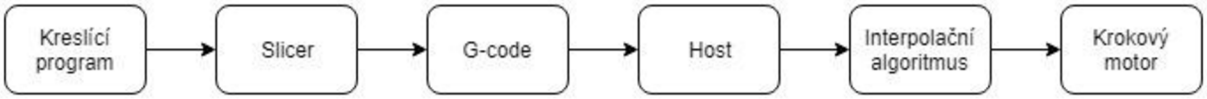
CNC stroje nemívají žádné ručně ovládané prvky, ty jsou nahrazeny speciálním řídícím softwarem, který na základě vstupních dat ovládá motory a ventily, tak aby bylo dosaženo požadovaného výsledku.

Výstupem této diplomové práce by mělo být odsimulování řídícího algoritmu pro řízení 3D tiskárny, což je jednoúčelový typ počítačem řízeného stroje, který slouží převážně k výrobě plastových prototypů. Požadovaného výrobku se dosahuje postupným nanášením vrstev roztaveného plastu, který se spojuje s předchozími vrstvami, respektive s heat bedem (pracovní podložkou) a po vychladnutí vytvoří celistvý model. 3D tiskárna se skládá z rámu, krokových motorů, převodů, vyhřívané podložky, extrudéru (pohyblivá tryska s topnou spirálou a chladičem, která je určena k vytlačování roztaveného plastu) a řídící jednotky.

1 Úvod do CNC problematiky

1.1 Proces zpracování dat pro CNC stroj

Proces obvykle začíná u technického výkresu součástky, což je základní část našeho procesu. Po zkreslení výkresu v nějakém z 3D kreslících programů se vhodným nástrojem převede na tzv. G-code a M-code. Tyto kroky probíhají v počítači mimo samotné CNC zařízení. Vše ostatní se však už provádí přímo ve stroji, a tam je často limitující výpočetní výkon embeded procesorů a pamětí. Dalším krokem tedy je přečíst G-code a na jeho základě vypočítat (aproximovat) trajektorii pohybu nástroje, nebo jak je tomu v případě 3D tiskárny, trajektorii pohybu extrudéru. Posledním krokem je z těchto trajektorií vypočítat pomocí algoritmů ideální profily pro řízení motorů. V případě 3D tiskáren se jedná výhradně o hybridní krokové motory, které se vyznačují především velkým momentem a vysokým rozlišením.



Obr. 1 - Diagram procesu zpracování dat pro CNC

1.2 Kreslící programy

Předtím, než se bude moci cokoliv vytisknout, musí být stanoven 3D model daného objektu. Na internetu je k dispozici celá řada stránek, kde je možné si model stáhnout. Pokud však nenajdeme model, který potřebujeme, je třeba si ho vytvořit. K tomu nás slouží 3D modelovací programy, označované zkratkou CAD (computer-aided design).

Je důležité, aby kreslící software dokázal vyexportovat hotový model ve formátu STL, kompatibilním se slicerem (česky: plátkovačem), což jsou téměř všechny novější programy. Programy můžeme rozdělit na neplacené a placené.

Neplacené programy mají obvykle značně omezené možnosti, především, pokud se jedná o verzi odvozenou z placené verze. Pro nenáročné používání jsou však naprosto vyhovující. Z vlastní zkušenosti mohu doporučit program Sketch Up, který si našel oblibu u mnoha lidí z celého světa a má i svou vlastní databázi modelů a programových rozšíření, díky kterým budete schopni nakreslit téměř cokoliv.

Příklady neplacených verzí modelovacích softwarů jsou:

- Google Sketch Up
- Open SCAD
- TinkerCAD
- Onshape
- Kompas 3D Home edition

Placené CAD programy jsou vhodné spíše pro každodenní použití, kde obrovská paleta funkcí dokáže velmi usnadnit práci.

Příklady placených verzí modelovacích softwarů jsou:

- Solid works
- AutoCAD
- Rhinoceros
- Kompas 3D

1.3 Slicer

3D tiskárna obvykle neumí rozluštit soubory formátu .stl (stereolithography), proto musí být použit ještě jeden mezičlánek, který se nazývá slicer a s jeho pomocí převést dtat do formátu .g (G-code). Jde o software, který celistvý model rozřeže horizontálně na tenké vrstvy, po nichž se bude následně postupně tisknout. V této části se také vytváří G-code, který říká 3D tiskárně, kdy a jak pohybovat extrudérem a kolik hmoty při tom vytlačit, aby byl co nejpřesněji zpětně vytvořen požadovaný model.

Příklady slicerů jsou:

- Cura
- Slic3r
- Simplify3D
- Skeinforge

1.4 G code

G-code je programovací jazyk pro CNC stroje, který se skládá z instrukcí, které popisují pohyb a akce, které mají být během pohybu prováděny. Je používán v téměř všech počítačem řízených strojích, od soustruhů, přes laserové řezačky, až po 3D tiskárny a plottery. Standard, podle kterého se G-code řídí, se nazývá ISO 6983, ale některé země mohou používat jiné standardy. ISO 6983 definuje strukturu programovacího jazyka, jeho příkazy a jejich význam. Každý z příkazů se skládá z písmena a čísla, který ho přesně definuje a většina příkazů má navíc volitelné parametry. Jazyk dostal jméno právě podle toho, že nejpoužívanějším počátečním písmenem je G. Příkazy začínající písmenem G popisují pohyb a tvar. Dalším častým počátečním písmenem u příkazů je písmeno M, které reprezentuje akce spojené s konkretním typem stroje (výměna nástrojů, regulace chlazení a ohřevu, nastavení a kalibrace os).

Skutečnost je však taková, že výrobci často používají své vlastní varianty G-codu, které jsou sice založené na standardu ISO 6983, ale s drobnými obměnami, protože požadavky výrobců jsou mnohdy natolik odlišné, že nemohou být pokryty jediným standardem. Navíc standard některé oblasti psaní G-codu, jako je například psaní komentářů, vůbec nedefinuje. V praxi jsou

tedy příkazy pro pohyb, začínající písmenem G, obvykle převzaty ze standardu, ale více specifické příkazy pro ovládání akcí a dodatečných funkcí stroje, začínající písmenem M, se často liší. To však dělá G-cody nepřenositelné mezi stroji od různých výrobců, naštěstí výrobce ke stroji obvykle přidává nástroj, který dokáže konvertovat standardní zápis G-codu, například z CAD programu, do formátu kompatibilního s jeho strojem.

Pro tento projekt byla zvolena sada příkazů kompatibilní s programem Marlin, což je open source firmware původně vytvořený pro 3D tiskárny značky RepRap. Je odvozen z programů Sprinter a grbl a stal se novým open source programem 12. srpna 2011. Marlin je licencován pod GPLv3 a je tedy volně šiřitelná pro všechny typy aplikací. Je například používán v komerčně prodávaných 3D tiskárnách od firem Ultimaker, Printerbot nebo Prusa Reasearch. [2]

1.4.1 Použitá sada příkazů

Tabulka G-kódu:

Základní příkazy G-kódu		
číslo	popis	parametry
G0	Rychlé polohování bez extruze	
G1	Lineární interpolace	G1 X10 Y20 E0.5 - pojede na pozici 10, 20 a vytlačí při tom 0.5mm filamentu
G2	Kruhová interpolace ve směru hod. ručiček	
G3	Kruhová interpolace proti směru hod. ručiček	
G4	Časová prodleva	
G17	Volba pracovní roviny X-Y	
G18	Volba pracovní roviny X-Z	
G20	Nastaví jednotky na palce	
G21	Nastaví jednotky na mm	
G28	Najetí na domovskou pozici	
G90	Nastavení absolutního polohování	
G91	Nastavení relativního polohování	
G92	Nastavení pozice	G92 bez určení os vyresetuje osy na nulu

Tab. 1 - Základní příkazy G-kódu [2]

Tabulka M-kódu:

Základních příkazů M-codu		
číslo	popis	parametry
M1	Stop	
M2	Sleep	
M17	Zapne krokové motory	
M18	Vypne krokové motory	
M80	ATX napájení ON	
M81	ATX napájení OFF	
M82	Nastaví extruder do absolutního módu	
M83	Nastaví extruder do relativního módu	
M84	Vypne přídrž motorů na všech osách	Nedoporučuje se během tisku
M92	Nastavení počtu kroků na mm	M92 X102.26 Y102.26 Z1850.5 E550.3
M105	Vrátí aktuální teplotu hotbedu a hotendu	
M106	Spustí větrák na nastavené otáčky	M106 S127 - ventilátor na 50% (max 255)
M109	Nastaví teplotu hotendu	M109 S190 - hodnota ve stupních Celsia
M114	rátí aktuální pozici os	
M190	Nastaví teplotu hotbedu	M190 S50 - teplota ve stupních Celsia
M204	Nastaví akceleraci v osách X a Y	
M206	Posune začátek tisku	M206 X40 Y40 - home na 40, 40
M220	Změní rychlost tisku	M220 S150 - rychlost na 150%
M221	Změní poměr extruze k osám X a Y	M221 S90 - 90% rychlosti extrudéru
M301	Nastaví konstanty PID regulátoru	M301 P20 I5.1 D200.5
M400	Čeká na dokončení všech pohybů	
M503	Vypíše aktuální nastavení	
M500	Uloží aktuální nastavení do EEPROMu	

Tab. 2 - Základní příkazy M-kódu [2]

Příklad použití G-kódu:

G0 X12 ; přesune extrudér o 12mm na X-ové ose

G0 F1500 ; nastaví rychlost posuvu na 1500mm za minutu

G1 X90.6 Y13.8 ; posune extrudér na pozici 90.6mm na X-ové ose a 13.8mm na Y-ové ose

1.5 Host

Tiskárna může fungovat samostatně, pokud do ní vložíme data ve vhodném formátu například na SD kartě nebo může být ovládána z počítače, kontroléru či jiné platformy, jako je Raspberry PI nebo i tablet s Androidem, pomocí tzv. hostu (česky: hostitele). Host je tedy ovládací program, který posílá G-code, připravený slicerem, do 3D tiskárny. Obecně platí, že při použití hostu se dosahuje lepších výsledků než při tisku přímo ze softwaru tiskárny.

Mezi zástupce tohoto softwaru patří:

Pronterface – open-source host od firmy Kliment

OctoPrint – open-source host pro Raspberry PI

Není vyjímkou, že se některé slicery jsou zároveň i hosty, jako například:

Cura – open-source od Ultimaker

Simplify3D – zpoplatněný software

A některé hosty mají v sobě implementovány slicery:

Repetier Host – closed-source host od Repetier Software s integrovanými slicery (Slic3r, Cura a Skeinforge)

1.5.1 Repetier Host

Umožňuje poměrně intuitivní ovládání tiskárny, generování, ukládání a editaci G-codu s 3D náhledem vygenerovaného kódu, vkládání modelů, jejich základní úpravy, jako jsou rotace, zvětšení, zmenšení či kopírování, utomatické či manuální umístění na tiskové ploše atd.

Při použití slicovacího modulu slic3r, umožňuje tisk více extrudery s různými rozměry trysek. Integrovaný Cura slicr nemá možnost rozšíření moduly jako při použití kompletního Cura-SW.

V poslední verzi mimo standardního připojení tiskárny přes seriový port, umí tisknout přímo po LAN či WLAN pomocí Raspberry PI. [3]

2 Rešerše interpolačních algoritmů pro CNC řízení využívající S-křivky

2.1 Interpolační algoritmus

Interpolační algoritmus je v oblasti 3D tisku velmi důležitou a zároveň nezbytnou součástí, protože vše, co jsme doposud dostali je kupa řádků kódu s písmeny G nebo M na začátku. Přesně tedy zatím známe počáteční a koncový bod pohybu extrudéru a to, po jaké trajektorii se mezi nimi máme pohybovat. Jenže náš stroj je řízený krokovými motory a ty mají konečný počet kroků, které mohou mezi počátečním a koncovým bodem vykonat. Je tedy nutné vypočítat vnitřní body trajektorie a zjistit kdy a o kolik kroků se má který motor pootočit, a právě k tomu nám slouží interpolační algoritmy.

Interpolační algoritmy se nejčastěji používají v počítačové grafice a slouží k výpočtu, pokud možno nejpřesnější aproximace při vykreslování objektů na monitor. Představme si třeba, že v nějakém kreslícím editoru nakreslíme přímku, která nebude rovnoběžná s hranami monitoru. V tom případě se stane, pokud se nám nepodaří trefit úhel 45 stupňů, že body přímky budou vycházet mezi pixely a neboť monitor má také konečný počet pixelů, tak celou prací interpolačního algoritmu je se rozhodnout, kterému ze dvou sousedních pixelů nechá přímku procházet. Tady můžeme vidět analogii s 3D tiskárnou, protože vše, co potřebujeme vědět je, jestli má motor zůstat v klidu nebo už se má pootočit o další krok.

Interpolační algoritmy se od sebe liší především dvěma parametry. Přesností aproximace, která je nejčastěji jeden krok nebo půl kroku, ale vyšší přesnost nelze z konstrukčního hlediska u krokových motorů dosáhnout. Druhým je výpočetní náročnost, ta nejvíce závisí na tom, v jakém oboru hodnot počítáme. Zkoušíme se vyhnout zejména reálným číslům a složitým matematickým operacím. Oba parametry jsou na sobě závislé, takže u rychlejšího algoritmu je zpravidla nižší přesnost aproximace.

Další dělení algoritmů je závislé na tom, jakou křivku jsou schopny interpolovat (aproximovat). Tady rozlišujeme 2 typy interpolačních algoritmů. Lineární, který dokáže pracovat s úsečkou a kruhový, který si poradí s interpolací kružnice. Některé algoritmy zvládají oba typy interpolací.

Interpolačních algorimů je hned několik a k těm nejznámějším patří:

- DDA algoritmus,
- Bresenhamův algoritmus,
- B-spline,
- NURBS.

2.1.1 DDA algoritmus

Je jedním z prvních algoritmů používaných v počítačové grafice. Je založen na přičítání konstantních přírůstků k oběma souřadnicím, počínaje počátečním bodem úsečky. Obecně mohou mít přírůstky libovolnou velikost, záleží na velikosti kroku a sklonu úsečky.

DDA je výchozí algoritmus pro Bresenhamův. Hlavní rozdíl je v datovém typu čísel - zde se pohybujeme v reálných číslech (tudíž používáme buď double nebo float datové typy) a algoritmus je o poznání pomalejší. Na pochopení se ale zdá být jednodušší.

Pro pochopení algoritmu musíme znát analytické vyjádření přímky:

$$y = mx + q$$

2-1

kde m je směrnice přímky a lze ji určit následovně:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

2-2

kde y_1 … počáteční bod úsečky na ose y

y_2 … koncový bod úsečky na ose y

x_1 … počáteční bod úsečky na ose x

x_2 … koncový bod úsečky na ose x

Rastrový rozklad úsečky je založen na vzorkování úsečky s konstantním krokem vždy podle jedné z os x nebo y a to podle svého sklonu, který je vyjádřen směrnicí m. Je-li:

$|m| > 1$, *pak rozdíl y-ových souřadnic je větší, a proto bude vzorkována s krokem 1 na ose y*

$|m| < 1$, *pak rozdíl x-ových souřadnic je větší, a proto bude vzorkována s krokem 1 na ose x*

$|m| = 1$, *pak krok na obou osách bude 1*

Osa, na které se vzorkuje s krokem 1 se nazývá hlavní, druhá osa je vedlejší.

Na vedlejší ose je tedy krok menší než 1 a lze ho vyjádřit matematicky z obecného vztahu 2-2.

$$m = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

2-3

V případě, že $|m| < 1$, je následující krok na ose x: $x_{i+1} = x_i + 1$, po dosazení do rovnice dostaneme:

$$m = \frac{y_{i+1} - y_i}{1}$$

2-4

kde po vyjádření následujícího kroku y_{i+1} dostaneme:

$$y_{i+1} = y_i + m$$

2-5

V druhém případě, když je $|m| > 1$, je následující krok na ose y: $y_{i+1} = y_i + 1$ a pro zjištění velikosti kroku na ose x dosadíme do rovnice 2-3 a dostaneme:

$$m = \frac{1}{x_{i+1} - x_i}$$

2-6

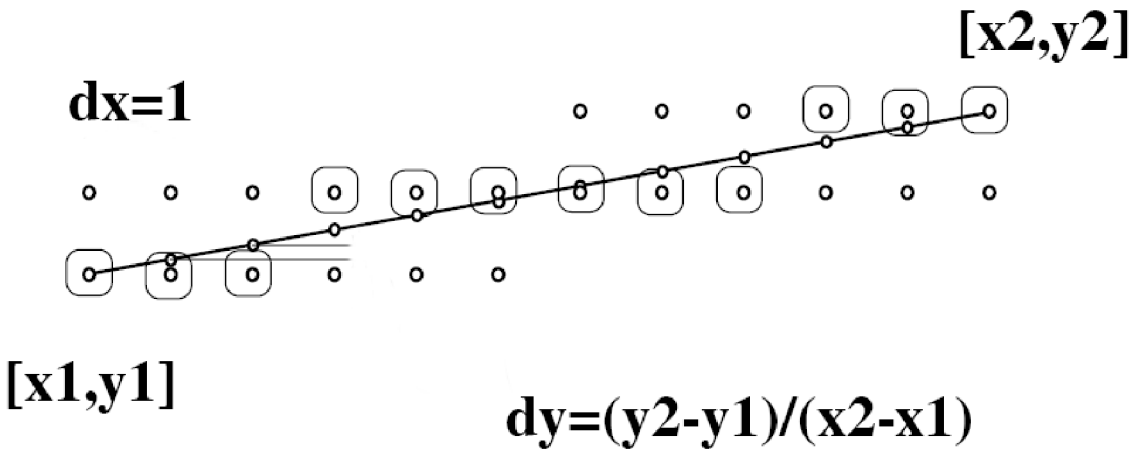
kde po vyjádření následujícího kroku x_{i+1} dostaneme:

$$x_{i+1} = x_i + 1/m$$

2-7

DDA algoritmus je tedy cyklický a v každém cyklu se přičítá celočíselný krok na hlavní osu a neceločíselný přírůstek na vedlejší osu, který je roven směrnici m, resp. 1/m. Souřadnice vedlejší osy je nutné počítat jako reálnou hodnotu, která se zaokrouhluje až při jejím použití.

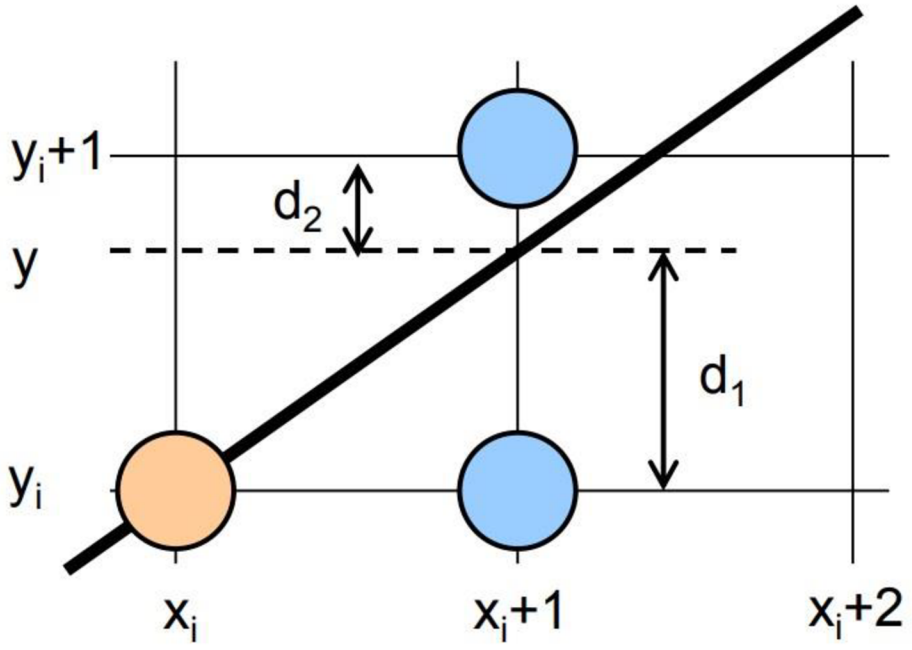
Algoritmus je jednoduchý a přehledný, ale výpočetně náročnější, kvůli práci s desetinnými čísly. DDA algoritmus zvládá lineární i kruhovou interpolaci (s upraveným algoritmem). [4] [5]



Obr. 2 - DDA lineární interpolace [4]

2.1.2 Bresenhamův algoritmus

Pravděpodobně nejefektivnější algoritmus pro vykreslení úsečky. Vychází z DDA algoritmu, narozdíl od něj se pohybuje ale pouze v celých číslech (int), což je nesporná výhoda oproti všem ostatním řešením.



Obr. 3 - Bresenhamova lineární aproximace [4]

Na obrázku č. 3 je nakreslena část rastru s hlavní osou x. Rastr si můžeme představit jako možné kroky krokového motoru, kde průsečíky čar reprezentují možné pozice extrudéru. Po nakreslení levého koncového bodu úsečky je třeba zvolit, zda další bod na souřadnici x+1 bude stále na pozici se stejnou souřadnicí y, nebo se souřadnicí o krok větší, tedy y+1. Algoritmus se rozhoduje podle toho, která z možných sousouřadnice y se více blíží skutečné hodnotě y na dané úsečce. Předpokládejme, že jsme nyní na pozici o souřadnicích [x, y]. Dva další možné pixely leží na pozicích [x + 1, y] a [x + 1, y + 1]. Vyjdeme-li z obecné rovnice přímky y = mx + b, kde m je směrnice a b posun na ose y, získáme souřadnici y následovně:

$$y = m(x + 1) + b$$

2-8

Pak lze obecně vyjádřit:

$$d_1 = y - y_i$$

2-9

$$d_2 = y_{i+1} - y$$

2-10

Rozdíl těchto dvou vzdáleností vyjádříme jako:

$$\Delta d = d_1 - d_2$$

2-11

Na základě znaménka proměnné Δd můžeme pak snadno určit, zda setrvat na stávající y-ové souřadnici nebo se posunout o další krok.

Záporná hodnota Δd, znamená, že skutečný bod leží blíže bodu o souřadnici y_i – tedy stávající souřadnici, kdežto kladná hodnota Δd, značí, že je potřeba se posunout o jeden krok na souřadnici y_{i+1}. To byla lineární interpolace

Bresenhamův algoritmus se však kromě lineární používá i pro kruhovou interpolaci a v jiných úpravách zvládá také interpolaci elipsy.

Kruhová interpolace (obr. 4) pomocí Bresenhamova algoritmu má následující postup:

Opět jde o velice rychlý algoritmus, protože počítá pouze s celými čísly, ale také proto, že pomocí algoritmu je možné určit pouze souřadnice jedné osminy kružnice (tzv. oktantu) a zbytek určit pomocí symetrie.

Kružnice je jednoznačně definována středem $[x_s, y_s]$ a poloměrem r nebo pomocí třech bodů ležících na kružnici $[x_1, y_1]$, $[x_2, y_2]$, $[x_3, y_3]$.

Pokud je řídící osa x a vedlejší y, tak změna na ose x je o 1 krok a změna na ose y je menší než 1 celý krok. Podmínkou konce algoritmu je $x \geq y$.

Základní funkcí pro výpočet je F(x,y):

$$x^2 + y^2 - r = 0$$

2-12

Představme si, že máme bod na kružnici o souřadnicích $[x_i, y_i]$. Následující bod tedy může mít souřadnice:

$[x_{i+1}, y_i]$ nebo $[x_{i+1}, y_{i+1}]$

Bod přesně mezi uvedenými možnostmi je $[x_i+1, y_i-1/2]$ a říká se mu midpoint (česky: středový bod) a podle něj se někdy nazývá celý algoritmus jako “Midpoint algoritmus”.

Dosazením midpointu do základní rovnice kružnice (rovnice 2-12) dosatneme pro $F(x_i+1, y_i-1/2)$:

$$p_i = (x_i+1) + (y_i-1/2) - r^2$$

2-13

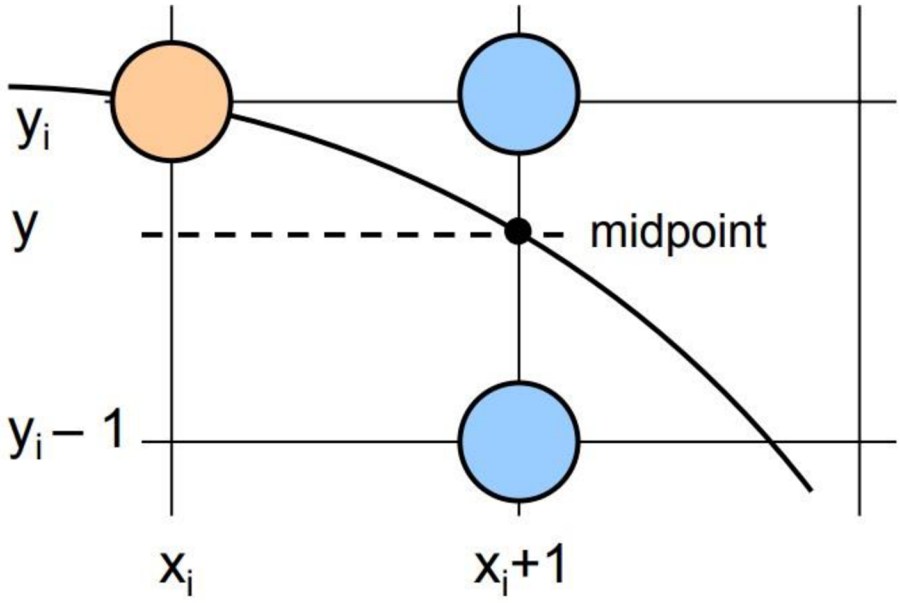
Pokud $p_i \leq 0$, tak další bod interpolované kružnice bude mít stejnou y-ovou souřadnici.

Pokud $p_i > 0$, tak následující bod bude mít souřadnici $y_i - 1$. [4] [10]

Algoritmus můžeme vyjádřit i iterativně ve tvaru:

$$p_{i+1} = p_i + 2x_i + 3 + (y_i - 1/2)^2 + (y_i + 1 - 1/2)^2$$

2-14



Obr. 4 - Bresenhamova kruhová interpolace [4]

2.1.3 B-spline

Křivka je definována n+1 kontrolními body a řádem křivky k.

Výhodou je lokální propagace na rozdíl od globální propagace u Beziérovy křivky. To znamená, že na tvaru křivky se podílejí pouze vybrané řídící body

Může být použit pro aproximaci otevřené i uzavřené křivky. [9]

Rovnice B-spline křivky:

$$x(u) = \sum_{i=0}^{i=n} N_{i,k}(u)x_i \qquad pro\ 0 \leq u \leq n-k+2$$

2-15

Matematická definice B-spline křivky (n=5, k=3)

$$x(u) = N_{0,3}(u)x_0 + N_{1,3}(u)x_1 + N_{2,3}(u)x_2 + N_{3,3}(u)x_3 + N_{4,3}(u)x_4 + N_{5,3}(u)x_5$$
$$\text{pro } 0 \leq u \leq 4$$

2-16

Vlastnosti B-spline křivky:

- je $C^{(k-2)}$ kontinuální
- je tvořena (n-k+2) segmenty
- pouze určité kontrolní body ovlivňují jednotlivé segmenty křivky
- dané kontrolní body ovlivňují 1 nebo 2 nebo …k segmenty (první kontrolní bod ovlivňuje pouze první segment, druhý bod ovlivňuje první a druhý segment…)

Nevýhodou této interpolace je, že nemusí přímo procházet zadanými body

Základní funkce B-splinu:

$$N_{i,k}(u) = \frac{(u-t_i)N_{i,k-1}(u)}{t_{i|k-1}-t_i} + \frac{(t_{i+k}-u)N_{i+1,k-1}(u)}{t_{i|k}-t_{i+1}}$$

2-17

$$t_i(0 \leq i \leq n+k) \quad \ldots uzlové\ hodnoty$$
$$t_i = 0 \quad pokud\ i < k$$
$$t_i = i-k+1 \quad pokud\ k \leq i \leq n$$
$$t_i = n-k+2 \quad pokud\ i > n$$

$$N_{i,k}(u) = 1 \quad pokud\ t_i \leq u \leq t_{i+1}, jinak\ je\ rovno\ 0$$

2.1.4 NURBS

Non-uniform rational B-spline (česky: Neuniformní racionální B-spline), někdy se humorně říká, že tato zkratka znamená „Nobody understands rational B-spline“. Jak název naznačuje, tak algoritmus vychází z klasického B-splinu, ale na rozdíl od něj má možnost měnit váhy h jednotlivých kontrolních bodů. NURBS křivku můžeme vypočítat podle následujícího vztahu:

$$x(u) = \frac{\sum_{i=0}^{i=n} h_i N_{i,k}(u) x_i}{\sum_{i=0}^{i=n} h_i N_{i,k}(u)} \qquad pro\ 0 \leq u \leq n - k + 2$$

2-18

Čím větší váhu daný kontrolní bod má, tím více je k němu křivka přitahována. Pokud jsou všechny váhy rovny 1, pak je křivka stejná jako B-spline. NURBS může být použit k modelování kružnice a jejich částí, k lineární interpolaci, a dokonce i k modelování paraboly, hyperboly a elipsy vhodnou změnou vah u daných bodů. Je to tedy univerzální nástroj použitelný pro interpolaci všech typů křivek. Z toho důvodu je používaný jako standard v mnoha CNC strojích. Nevýhodou je větší výpočetní náročnost složitých výpočtů a stejně jako u B-splinu vypočtená křivka nemusí procházet přímo zadanými body. [9]

2.2 Krokový motor

Krokový motor patří mezi synchronní točivé stroje s nespojitým pohybem. Je napájený stejnosměrnými elektrickými impulsy, které v cívkách statoru generují magnetické pole, jenž přitáhne magnety s opačným pólem na rotoru. Vhodným zapojením a řízením proudu v cívkách vytvoříme rotující magnetické pole, které otáčí rotorem.

Podle požadovaných vlastností krokového motoru volíme jeho typ a druh řízení.

Z důvodů přechodových magnetických jevů je rychlost krokového motoru omezena na několik stovek kroků za sekundu, při jejím překročení může dojít ke ztrátě kroků.

Krokové motory jsou nejčastěji používanými pohony v 3D tiskárnách a to především proto, že pracují bez nákladných snímačů otáček nebo polohy, jsou jednodušší a tím i provozně spolehlivější a levnější. Řízení se typicky provádí v přímé větvi bez zpětné vazby. Mají také vysokou životnost a jsou bezúdržbové.

Tomuto typu řízení, bez zpětné vazby a bez snímačů polohy a natočení, se říká řízení v otevřené smyčce a používá se téměř u všech typů krokových motorů. Kromě finanční úspory je to však zároveň i velká nevýhoda krokových motorů např. oproti servům, protože nemáme žádnou zpětnou vazbu o tom, že krokový motor opravdu daný pohyb vykonal. Aby nedocházelo při rychlých změnách rychlosti, potažmo zrychlení, nebo směru pohybu otáčení rotoru ke ztrátám kroků, používají se tzv. rychlostní profily, které si představíme v kapitole 2.3. Pomocí těchto profilů se plynule zvyšují a snižují otáčky motoru a tím se minimalizuje ryv, který nejčastěji ztráty kroků způsobuje. [7]

2.2.1 Typy řízení krokového motoru

Pro krokový motor motor existují dvě základní metody řízení – unipolární a bipolární.

Unipolární řízení

Při unipolárním řízení prochází proud v jednom okamžiku právě jednou cívkou. Motor s tímto buzením má nejmenší odběr, ale také poskytuje nejmenší kroutící moment. Výhodou tohoto řešení je jednoduché zapojení řídící elektroniky, kde stačí jen jeden tranzistor na každou cívku.

Bipolární řízení

Při bipolárním řízení prochází proud vždy dvěma protilehlými cívkami. Ty jsou zapojené tak, že mají navzájem opačně orientované magnetické pole. Motor v tomto režimu poskytuje větší kroutící moment, ovšem za cenu vyšší spotřeby.

Další dělení řízení krokových motorů je podle počtu fází, které jsou při něm ovládány současně. Máme tedy jednofázové a dvojfázové řízení, ale taky stále populárnější třífázové a pětifázové.

Jednofázové řízení znamená, že magnetické pole je generované pouze jednou, respektive dvěma cívkami v případě bipolárního řízení. U dvojfázového to jsou potom dvě, respektive čtyři cívky atd. S rostoucím počtem řízených fází se zvětšuje rozlišení, ale zvyšuje se spotřeba elektrické energie.

Při řízení v celokrokovém režimu mohou mít rozlišení až 200 kroků na otáčku, úhel kroku je 1,8°.

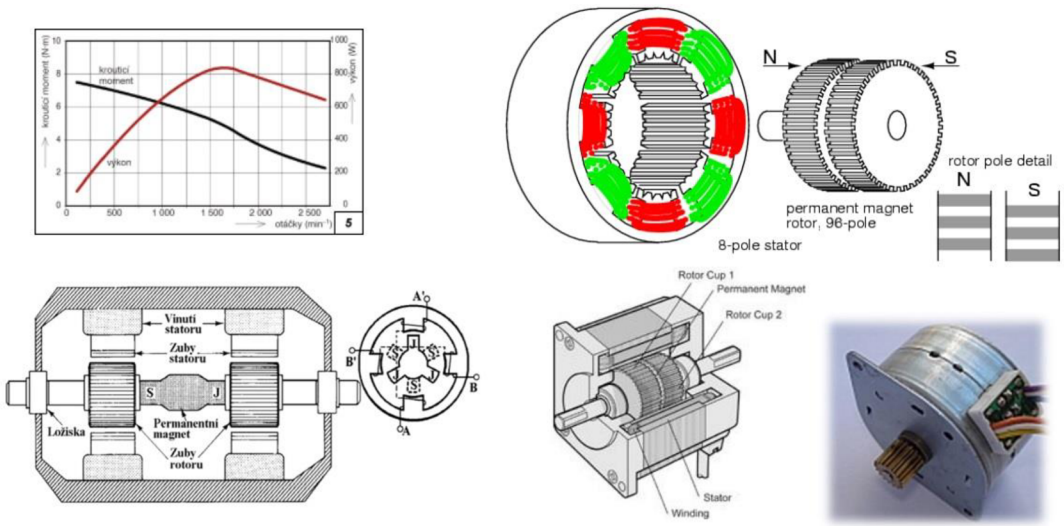
Při řízení v půlkrokovém režimu mohou mít rozlišení až 400 kroků na otáčku, úhel kroku je 0,9°.

Existuje i takzvané mikrokrokování, kde se mění velikosti proudů vstupujících do jednotlivých cívek, kde teoreticky můžeme dosáhnout libovolného natočení, v praxi se však používá maximálně 32 mikrokroků mezi jedním krokem, kde se dosahuje úhlu 3,4′ na jeden mikrokrok. Při navýšení počtu mikrokroků může dojít v nehomogenitě magnetického pole a natočení krokového motoru a dochází k nepřesnostem v krokování, při navýšení počtu kroků při zachování maximální přesnosti se proto častěji používá krokový motor v celokrokovém režimu s převodovkou. [6]

2.2.2 Hybridní krokový motor

Při výběru krokových motorů pro 3D tiskárnu můžeme volit ze tří typů. Motory s pasivním rotorem (s permanentními magnety), motory s aktivním rotorem (s proměnnou reluktancí) a hybridní motory. Protože však první dva typy mají buď nízké rozlišení, nebo malý moment, používají se v těchto aplikacích výhradně hybridní motory, které jsou sice dražší, ale mají dobré vlastnosti v oblasti rychlosti, momentu a rozlišení.

Hybridní krokový motor (obr. 5) má na rotoru umístěny dva permanentní magnety, na jejichž koncích jsou umístěny feromagnetické nástavce. Zuby na obou discích jsou vzájemně pootočeny o polovinu zubu, aby bylo dosaženo vyššího rozlišení na otáčku.



Obr. 5 - Hybridní krokový motor [6]

2.3 Rychlostní profily pro řízení krokového motoru

Požadavky na téměř každý počítačem řízený stroj jsou v celku podobné. Chceme, aby dosahoval co největší přesnosti a rychlosti a přitom, aby vibrace a rázy pohonných ústrojí byly co nejmenší. Velkou měrou se na naplnění těchto požadavků promítá volba rychlostních profilů motorů. U 3D tiskáren, plotrů a všeobecně u všech strojů, kde jsou použity krokové motory, je správná volba těchto profilů nezbytná, protože při určité změně zrychlení/zpomalení v čase, tato veličina se nazývá ryv, může dojít k nevratné ztrátě kroku. Ryv je tedy fyzikální veličina, která se vypočítá jako změna zrychlení v čase a charakterizuje nám, jak velké rázy se přenášejí do systému při akceleraci a brždění. Tvarem rozběhového profilu můžeme tuto hodnotu značně omezit. Máme 3 základní tvary profilů, které se nazývají podle tvaru průběhu rychlosti v čase: obdélníkový, trapezoidní a profily, které tvarem připomínají písmeno S, a říká se jim S křivky.

2.3.1 Obdélníkový

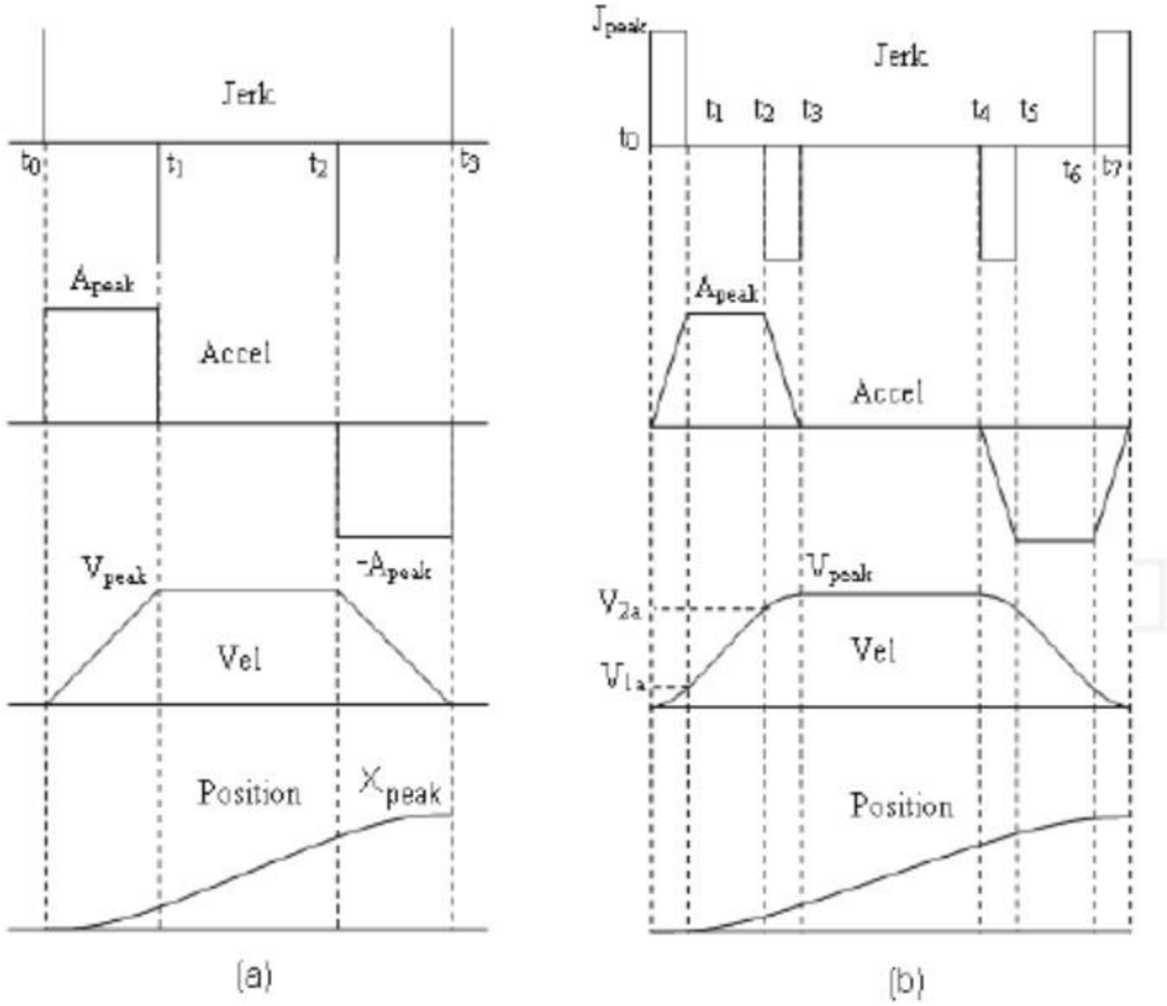
Je to nejjednoduší profil a jedná se prakticky o ON/OFF regulaci, která sepne motor na začátku pohybu bez jakékoliv dodatečné regulace rychlosti.

Jeho reálné využití je pouze u CNC strojů, kde se motor nerozbíhá pod zátěží a poté se otáčí s konstantní rychlostí, např. soustruh.

Tento řídící profil se skokovou změnou rychlosti je pro řízení krokových motorů bez zpětné vazby, kde požadujeme rychlost a přesnost naprosto nevhodný, zrychlení a ryv mají tvar jednotkového impulsu a je tedy skoro jisté, že při tomto typu řízení dojde ke ztrátě kroků.

2.3.2 Trapézoidní (lichoběžníkový)

Z tvaru lichoběžníkového profilu je zřejmé, že má linerní nárůst rychlosti a konstantní zrychlení. Můžeme tedy říct, že motor s tímto řídícím profilem poměrně rychle dosáhne požadované rychlosti, ale v okamžiku jejího dosažení, kdy spadne zrychlení skokově z maxima na nulu, dochází ke skokovým změnám zrychlení. Podobný skok se opakuje také v časech t0, t2 a t3 (viz obr. 6), kdy dochází ke změnám rychlosti. Tyto časově nespojité změny zrychlení způsobují, že ryv dosahuje obrovských, teoreticky nekonečných hodnot, které mohou způsobovat ztráty kroků u krokových motorů a také způsobují vibrace stroje, kde je poté nutné čekat na jejich ustálení, aby bylo dosaženo požadované polohy.

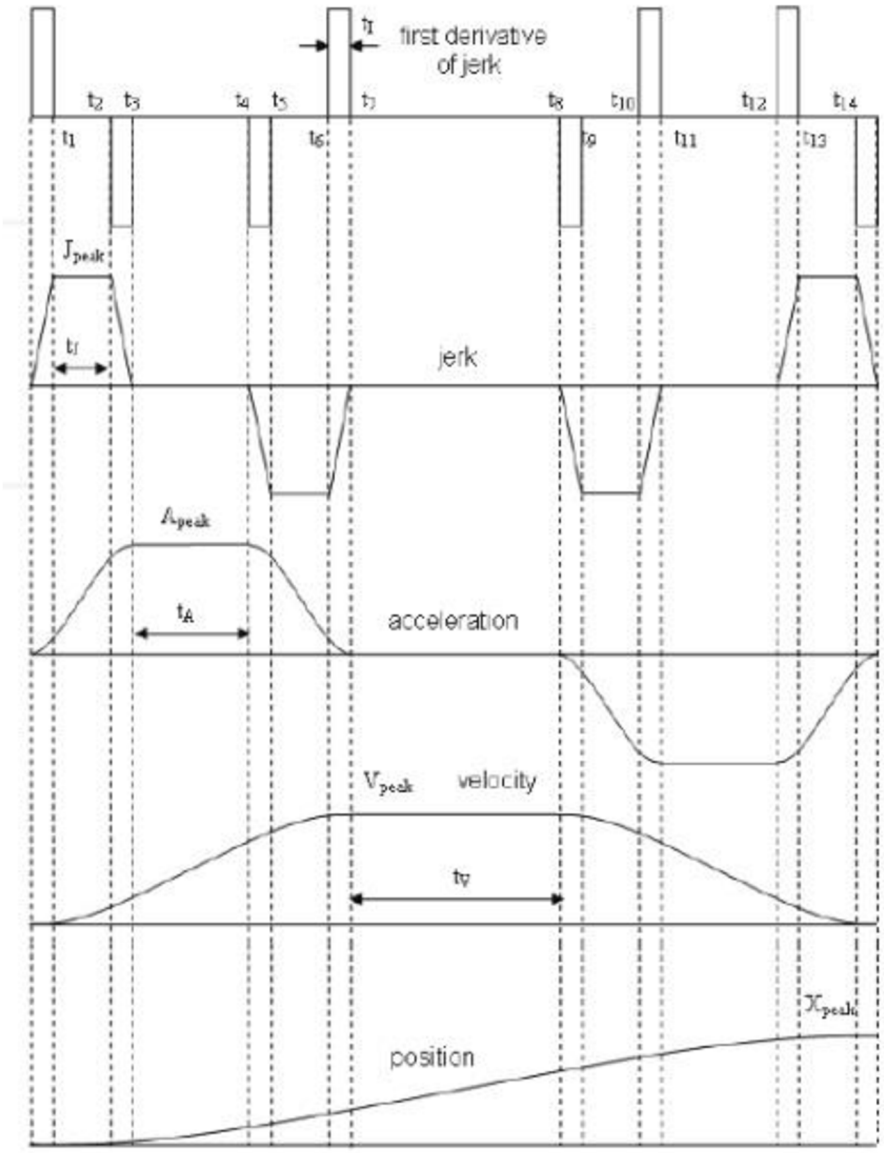


Obr. 6 - a) Trapézový a b) lichoběžníkový rychlostní profil [8]

2.3.3 S křivka

Problémy s mechanickými rázy a vibracemi můžeme úplně eliminovat použitím tzv. S-křivky třetího řádu, kde se rozběhové a zastavovací profily skládají z konvexních a konkávních parabol, kde vrcholy parabol jsou buď v nulové, nebo v konstantní rychlosti a uprostřed mezi nimi jsou inflexní body. Přechody v rámci rychlostního jsou tedy naprosto hladké a tvar zrychlení, resp. zpomalení v čase má pak lichoběžníkový tvar. Tím pádem už víme, že ryv má konečnou hodnotu a rázy do systému se tímto značně eliminují. Výsledkem použití tohoto rychlostního profilu je, že časy náběhu a doběhu můžou být sice delší, ale zato nedochází k vibracím a rázům a tím ke ztrátě kroků.

Hladšího průběhu se dá dosáhnout zvyšováním řádu S-křivky, např. při použití polynomu čtvrtém řádu jakožto rychlostního profilu docílíme trapezoidního průběhu ryvu, jak můžeme vidět na obrázku 7.



Obr. 7 - S křivka čtvrtého řádu [8]

Tento typ rychlostního profilu je náročnější na výpočet a je proto potřeba výkonnějších kontrolérů, ale v dnešním průmyslu už jsou už obvykle průmyslové počítače a embedded systémy dostatečně výkonné na to, aby zvládly náročné výpočty, a proto se s tímto typem rozběhového profilu můžeme setkat čím dál častěji.

Společně s S-křivkami se často používá i tzv. Look-ahead algoritmus, který prochází tvar trajektorie několik segmentů dopředu a zkoumá pod jakým úhlem je další segment natočen oproti předchozímu a podle toho pouze vhodně upravuje přechodovou rychlost krokového motoru ve zlomech trajektorie, aby na konci každého segmentu nemusel motor zastavovat a poté se znovu rozbíhat z nuly. [8]

2.3.4 Výpočet S-křivky

1) Pomocí diferenciálních rovnic

Diferenciální rovnice mají největší výpočetní náročnost, protože počítají okamžitou rychlost a zrychlení doslova v každém časovém okamžiku.

Rovnice pro zrychlení/zpomalení

$$v = \int_{t_0}^{t_1} a(t)dt \tag{2-19}$$

$$a = \int_{t_0}^{t_1} j(t)dt \tag{2-20}$$

kde v je okamžitá rychlost
a je okamžité zrychlení
t_0 je počáteční čas zrychlení/zpomalení
t_1 je koncový čas zrychlení/zpomalení
j je ryv

Z uvedených rovnic a z toho, co už jsme si o S křivkách řekli je patrné, že jedinou konstantou bude ryv, a sice jeho maximální povolená hodnota. O jeho velikost se při akceleraci a zpomalení v každém dalším časovém okamžiku zvětší resp. zmenší okamžité zrychlení a o hodnotu okamžitého zrychlení se zvětší resp. zmenší okamžitá rychlost. Integrací okamžité rychlosti pak lze vypočítat polohu extrudéru podle vztahu:

$$p = \int_{t_0}^{T} v(t)dt \tag{2-21}$$

kde v je okamžitá rychlost
p je okamžitá poloha
t_0 je počáteční čas
T je koncový čas

Kromě zrychlení a zpomalení má rychlostní profil obvykle také část s konstantní (maximální) rychlostí, kde jsou zrychlení a ryv rovny nule. [13]

2) Polynomialním výpočtem

Průběh S křivky lze vypočítat i pomocí polynomů třetího řádu.

Spojitý průběh:

$$P_T = P_0 + V_0 T + 1/2 A_0 T^2 + 1/6 J T^3 \quad \text{2-22}$$

$$V_T = V_0 + A_0 T + 1/2 J T^2 \quad \text{2-23}$$

$$A_T = A_0 + JT \quad \text{2-24}$$

Diskrétní průběh:

$$P_T = P_T + V_T + 1/2 A_T + 1/6 JT \quad \text{2-25}$$

$$V_T = V_T + A_T + 1/2 JT \quad \text{2-26}$$

$$A_T = A_T + JT \quad \text{2-27}$$

kde P_0 je počáteční pozice

V_0 je počáteční rychlost

A_0 je počáteční zrychlení

P_T je pozice v čase T

V_T je rychlost v čase T

A_T je zrychlení v čase T

J je ryv

3) Pomocí tabulky

Tabulka je v podstatě pole pozic nebo časových hodnot a pro každou hodnotu je vypočtena korespondující hodnota rychlosti a zrychlení v daném místě nebo čase. Tyto hodnoty jsou vypočteny předem a při jejich použití při řízení krokového motoru stačí najít příslušnou hodnotu v tabulce č. 3. [12] [14]

Konstrukce tabulky rychlostních profilů:

- Musíme mít rozumný počet hodnot (podle času zrychlení a zpomalení)
- Tabulka obsahuje hodnoty ve vzestupném pořadí
- Profil musí být symetrický během zrchlení a zpomalení, pokud chceme použít Profile table interpolaci

Existují dvě formy interpolací pomocí tabulky OneToOne a Profile table.

OneToOne interpolace používá tabulku pro každý vzorek během celého pohybu, zatímco u Profile table interpolace je profil rozdělen na 3 samostatné části. Fáze zrychlení, Fáze s konstatní rychlostí a Fáze zpomalení.

V první fázi řízení postupuje podle předem propočítaných hodnot, v první části lineárně zvyšujeme a poté lineárně snižujeme zrychlení, dokud není dosaženo požadované rychlosti. V druhé fázi je pohon držen v konstantních otáčkách a řídící software pravidelně kontroluje, kdy nastane tzv. bod zlomu, od kterého se má začít zpomalovat. Souřadnice bodu vypočítáme jako konečnou pozici mínus vzdálenost zpomalení. V třetí fázi pohon zpomaluje po vypočítaném profilu tak, aby v konečném bodě měl nulovou rychlost. Podobně jako v první fázi zpomalení nejřív lineárně narůstá a poté po stejné křivce klesá. Někdy se v první a třetí fázi můžeme také vyskytovat oblast s konstantním zrychlením, resp. zpomalením. [13]

Ukázka části tabulky rychlostního profilu aproximovaného S-křivkou:

Tabulka pro aproximaci S-křivky					
Vzorek	Ryv	Zrychlení	Rychlost	Poloha	Zaokrouhlená rychlost
0	0	0	0,0	0,0	0
1	0,2	0,2	0,3	0,4	0
2	0,2	0,4	0,8	1,5	1
3	0,2	0,6	1,5	3,3	2
4	0,2	0,8	2,4	6,1	2
5	0,2	1	3,5	10,2	4
6	0,2	1,2	4,8	15,6	5
7	0,2	1,4	6,3	22,6	6
8	0,2	1,6	8,0	31,5	8
9	0,2	1,8	9,9	42,3	10
10	0,2	2	12,0	55,3	12
11	0,2	2,2	14,3	70,8	14
12	0,2	2,4	16,8	88,8	17
13	0,2	2,6	19,5	109,6	20
14	0,2	2,8	22,4	133,5	22
15	0,2	3	25,5	160,5	26

Tab. 3 - Tabulka pro aproximaci S-křivky

3 Matematický model systému

Pro odzkoušení algoritmů byla vytvořena simulace v program Matlab a Simulink, která napodobuje chování pohybů 3D tiskárny. [15]

3.1 Model hybridního krokového motoru

Model vždy vychází z matematického popisu. Pro dvoufázově řízený krokový motor máme následující vztahy:

$$u_i = p \cdot \omega \cdot n \cdot \phi_M \cdot \sin(p \cdot \varphi) \tag{3-1}$$

kde: u_i je napětí indukované statorem
ω je úhlová rychlost rotoru
n je počet otáček
ϕ_M je magnetický tok
p je počet pólů
φ je úhlové natočení

Napěťová rovnice fáze A:

$$U - R \cdot i_A + L \cdot \frac{di_A}{dt} - M \cdot \frac{di_B}{dt} - p \cdot \omega \cdot n \cdot \phi_M \cdot \sin(p \cdot \varphi) = 0 \tag{3-2}$$

Napěťová rovnice fáze B:

$$U - R \cdot i_B + L \cdot \frac{di_B}{dt} - M \cdot \frac{di_A}{dt} - p \cdot \omega \cdot n \cdot \phi_M \cdot \sin(p \cdot (\varphi - \lambda)) = 0 \tag{3-3}$$

kde: U je napětí na svorkách motoru
R je odpor vinutí
iA je proud fází A
iB je proud fází B
L je vlatní indukčnost
M je vzájemná indukčnost
λ je úhel natočení mezi póly statoru

Momentová rovnice krokového motoru:

$$J \cdot \frac{d^2\varphi}{dt} + B \cdot \frac{d\varphi}{dt} + m_z - p \cdot n \cdot \phi_M \cdot [i_A \cdot \sin(p \cdot \varphi) + i_B \cdot \sin(p \cdot (\varphi - \lambda))]$$

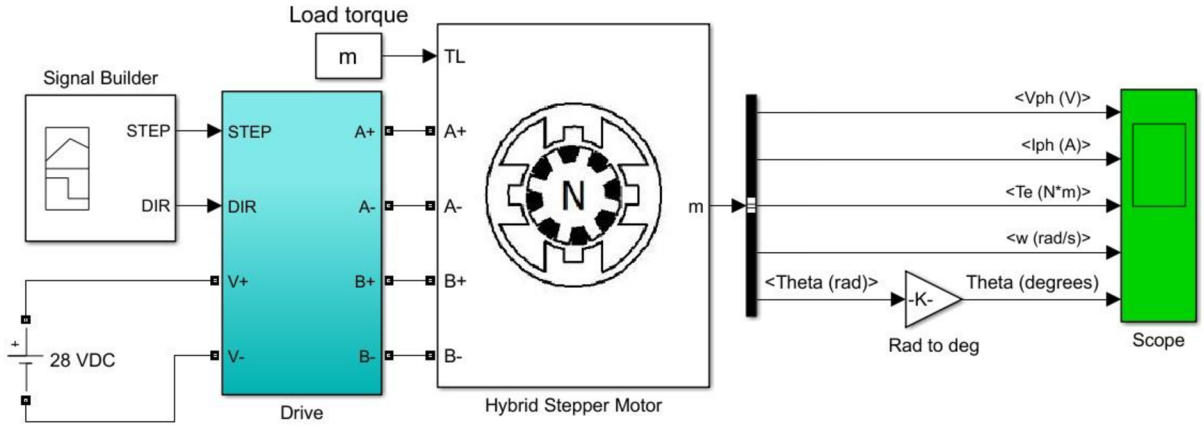
3-4

,kde: J je moment setrvačnosti

B je koeficient viskózního tření

m_Z je zátěžný moment

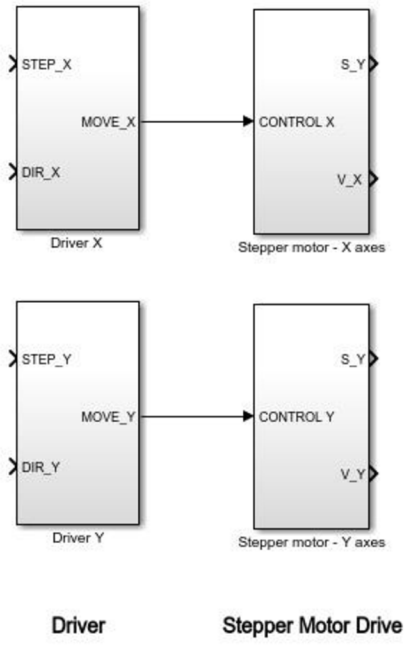
Na základě konzultace s vedoucím práce bylo rozhodnuto, že pro náš problém nejsou podstatné vnitřní parametry motoru, a proto byl použit pouze zjednodušený model, který vychází z již vytvořeného bloku krokového motoru a bloku pro jeho regulaci v program Matlab a Simulink (viz obr. 8).



Obr. 8 - Blok driveru a krokového motoru z programu Matlab a Simulink, ze kterého vychází použitý model

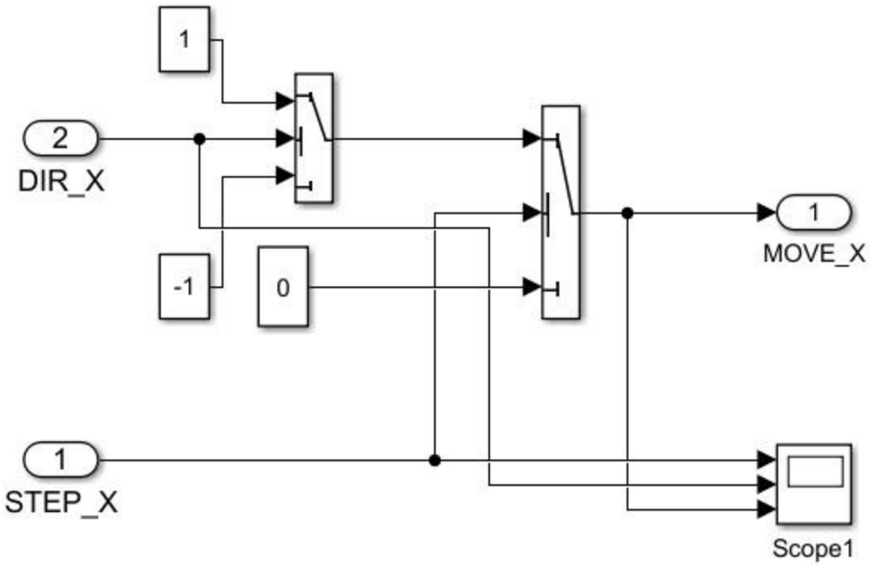
Podstatné pro tuto diplomovou práci byly pouze dva parametry motoru počet kroků na otáčku a vzdálenost, kterou urazí extrudér za jednu otáčku krokového motoru. Tyto hodnoty byly zadány takto: StepsPerRevolution = 200 a DistancePerRevolution = 10mm.

Na obrázku 9 můžeme vidět, jak vypadají bloky driveru a krokového motoru ve vytvořeném modelu v programu Matlab & Simulink.



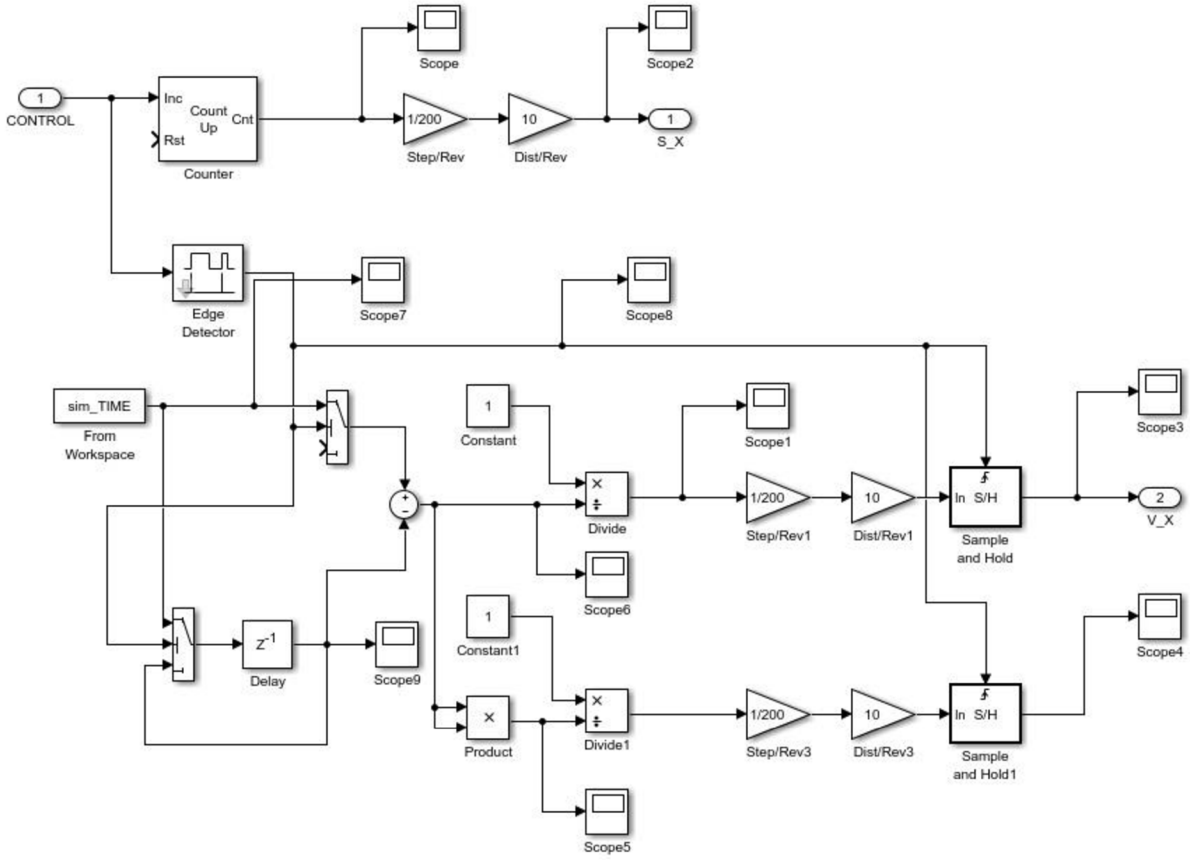
Obr. 9 - Zjednodušený model krokového motoru pro simulaci pohybu

Na obrázku 10 je systém rozhodování řízení na základě impulzů STEP a DIR uvnitř bloku driveru.



Obr. 10 - Blok driveru

Blok motoru je k vidění na obrázku 11. Kde v horní větvi můžeme vidět čítač kroků a jejich přepočet na vzdálenost a v dolní potom výpočet (průměrné) rychlosti z uražené vzdálenosti a časových diferencí.



Obr. 11 - Blok krokového motoru

4 Popis řídícího algoritmu

4.1 Čtení G-kódu

Pro odzkoušení vybraného algoritmu bude dostačující implementovat pouze G-kód pro pohyb extrudéru po přímce a po kružnici.

Implementované příkazy:

Tabulka implementovaných příkazů G-kód		
číslo	popis	parametry
G0	Rychlé polohovíní bez extruze	
G1	Lineární interpolace	G1 X10 Y20 E0.5 - pojede na pozici 10, 20 a vytlačí při tom 0.5mm filamentu
G2	Kruhová interpolace ve směru hod. ručiček	
G3	Kruhová interpolace proti směru hod. ručiček	

Tab. 4 - Implementované příkazy G-kódu

Čtení textového souboru s G-kódem probíhá pomocí program, který prochází celý soubor po jednotlivých řádcích, přičemž každý řádek musí začínat písmenem N a pořadovým číslem, pak následuje mezera a po ní písmeno G nebo M. Neboť však v Matlabu není potřeba ovládat prvky, jako je teplota extrudéru nebo zapínání a vypínání motorů, byly prozatím implementovány pouze příkazy v tabulce 4. Tyto čtyři příkazy jsou naprosto dostačující pro modelování trajektorie.

Ukázka části kódu pro vyčítání G-kódu z textového souboru:

```
% N
      if g_line(1) == 'N'
            col = 1;
            g_array(row, col) = str2double(g_line(2:g_spaces(1)-1));
        else
            error('Wrong G-code syntax - missing N');
        end
% G and M
        if g_line(g_spaces(1)+1) == 'G'
            col = 2;
            if length(g_spaces) == 1
                g_array(row, col) = str2double(g_line(g_spaces(1)+2:end));
            else
                g_array(row, col) = str2double(g_line(g_spaces(1)+2:g_spaces(2)-1));
            end
        elseif g_line(g_spaces(1)+1) == 'M'
            col = 3;
            if length(g_spaces) == 1
                g_array(row, col) = str2double(g_line(g_spaces(1)+2:end));
            else
                g_array(row, col) = str2double(g_line(g_spaces(1)+2:g_spaces(2)-1));
            end
        else
                error('Wrong G-code syntax - missing G or M');
        end
```

4.2 Interpolace

Interpolace byla provedena dvěma způsoby – pomocí Bresenhamova algoritmu a pomocí DDA algoritmu. Před vykonáním řídícího algoritmu si musíme zvolit jeden z nich.

4.2.1 Interpolace – Bresenhamův algoritmus

Pro interpolaci byl použit Bresenhamův algoritmus a to jak pro lineární, tak pro kruhovou interpolaci, protože pracuje pouze s celými čísly a je proto rychlejší než například interpolační algoritmus DDA, který je sice tomu Bresenhemovu do značné miry podobný, ale při rozhodování, jestli se má motor pootočit o další krok nebo setrvat v aktuální pozici, využívá desetiných čísel.

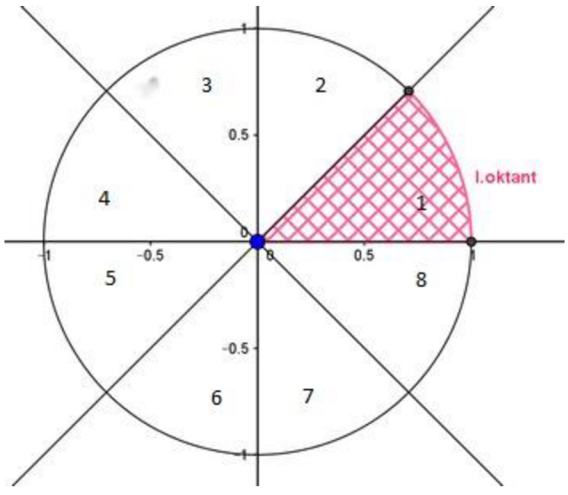
Dalším, v poslední době velmi používaným algoritmem je NURBS, k jeho implementaci však bohužel zatím nebylo v této práci nashromážděno dostatek validních materiálů a informací, ze kterých by bylo možné algoritmus sestavit. [11]

Ukázka části kódu lineární Bresenhamovy interpolace:

```
while 1

    B_linear(k, 1) = x1;
    B_linear(k, 2) = y1;

    if x2==x1 && y2==y1
        break;
    end

    e2 = 2*err;

    if e2 >= dy
        err = err + dy;
        x1 = x1 + sx;
    end

    if e2 <= dx
        err = err + dx;
        y1 = y1 + sy;
    end

    k = k + step;

end
```

Ukázka části kódu kruhové Bresenhamovy interpolace:

```
while ((x ~= x2)||(y ~= y2))

        k = k+1;

        if (x>=0 && y>0)

            if(abs(x)>=abs(y))
                % 1. oktant
                incY = -1;
                incX = 1;
                if (((x+1/2)*(x+1/2)+(y-1)*(y-1)-r*r)<0)
                    x = x + incX;
                end
                y = y + incY;
            elseif(abs(x)<abs(y))
                % 2. oktant
                incX = 1;
                incY = -1;
                if (((x+1)*(x+1)+(y-1/2)*(y-1/2)-r*r)>0)
                    y = y + incY;
                end
                x = x + incX;
            end
        elseif (x<0 && y>=0)
            ...

            ...
        end

        B_circular2(k, 1) = x + offset(1);
        B_circular2(k, 2) = y + offset(2);
        B_circular2(k, 4) = 0.5;

end
```



Obr. 12 - Rozložení oktantů v kruhu

Celý algoritmus je rozdělený na oktanty a podle nich se nastavují znaménka na X-ové a Y-ové ose. Jedna osa se vždy přičítá automaticky, podle toho, v jakém oktantu se právě pohybujeme. Poté se zkouší, jestli už je potřeba posunout i druhou osu nebo ještě setrvat na stejných souřadnicích.

4.2.2 Interpolace – DDA algoritmus

V programu byl implementován i DDA interpolační algoritmus, který bude sloužit pro srovnání výpočetní náročnosti oproti Bresenhemově algoritmu a pro otestování funkčnosti simulovaného systému v simulinku.

Ukázka části kódu lineární DDA interpolace:

```
dx = x2-x1;
dy = y2-y1;

if abs(dx) > abs(dy)
    steps = abs(dx);
else
    steps = abs(dy);
end

Xinc = dx/steps;
Yinc = dy/steps;

while i <= steps

    DDA_linear(k, 1) = x1;
    DDA_linear(k, 2) = y1;

    x1 = x1 + Xinc;
    y1 = y1 + Yinc;

    i = i+1;
    k = k+1;

end
```

4.3 Look-ahead algoritmus

Ještě dříve, než se z interpolovaných dat začne tvořit pohybový-rychlostní profil pro krokové motory, je potřeba se podívat na následující segmenty, jestli je potřeba zastavit až na nulovou rychlost nebo jen zpomalit. To se provádí pomocí tzv. look-ahead algoritmu, který každé křivce přiřadí úhel, který svírá s počátkem souřadnicového systému a následně danému úseku přiřadí počáteční a koncovou rychlost. Na základě těchto rychlostí se poté vytváří vhodný rychlostí profil uvnitř segmentu.

Mohou nastat 3 případy:

1. Pokud na sebe dva po sobě jdoucí úseky navazují a jsou pod stejným úhlem, tak extrudér pokračuje dále se stejnou rychlostí.
2. Pokud dva po sobě jdoucí úseky svírají úhel menší než +/-90 stupňů, tak se velikost rychlosti vypočítá poměrově, přičemž 0° je 100% a 90° je 0% z maximální rychlosti. Tady by bylo možná na zváženou, jestli by tato závislost neměla být spíše kvadratická, než přímoúměrná, ale toto je možné upravit kdykoliv na základě výsledků z praktické části.
3. Pokud úhel bude větší než +/-90 stupňů, tak se motory vždy úplně zastaví a poté se znovu rozjíždějí v dalším segmentu z nulové rychlosti.

Ukázka kódu look-ahead algoritmu:

```
%direction vector
ux = k2x - k1x;
uy = k2y - k1y;

%angle
if uy == 0 && ux > 0
      angle = 0;
elseif uy == 0 && ux < 0
      angle = 180;
elseif ux < 0
      angle = -(atan(uy/ux)*180/pi);
else
      angle = atan(uy/ux)*180/pi;
end

%add angle in interpolation matrix
interpolation{1,n}(m,3) = angle;

angle_dif = abs(interpolation{1,n}(1,3)-interpolation{1,n-1}(1,3));

if angle_dif > 90
      interpolation{1,n}(m,4) = 0;
else
      interpolation{1,n}(m,4) = 1-angle_dif/90;
end
```

4.4 Modelování S křivky

S-křivka byla namodelována několika různými způsoby a opět si můžeme v programu zvolit (stejně jako u interpolačních algoritmů) jaký typ výpočtu nebo aproximace rychlostního profilu použijeme.

V programu je možnost zvolit z těchto typů výpočtu/aproximace S křivek:

1. Pomocí polynomů
2. Pomocí tabulky

4.4.1 Pomocí polynomiálního výpočtu

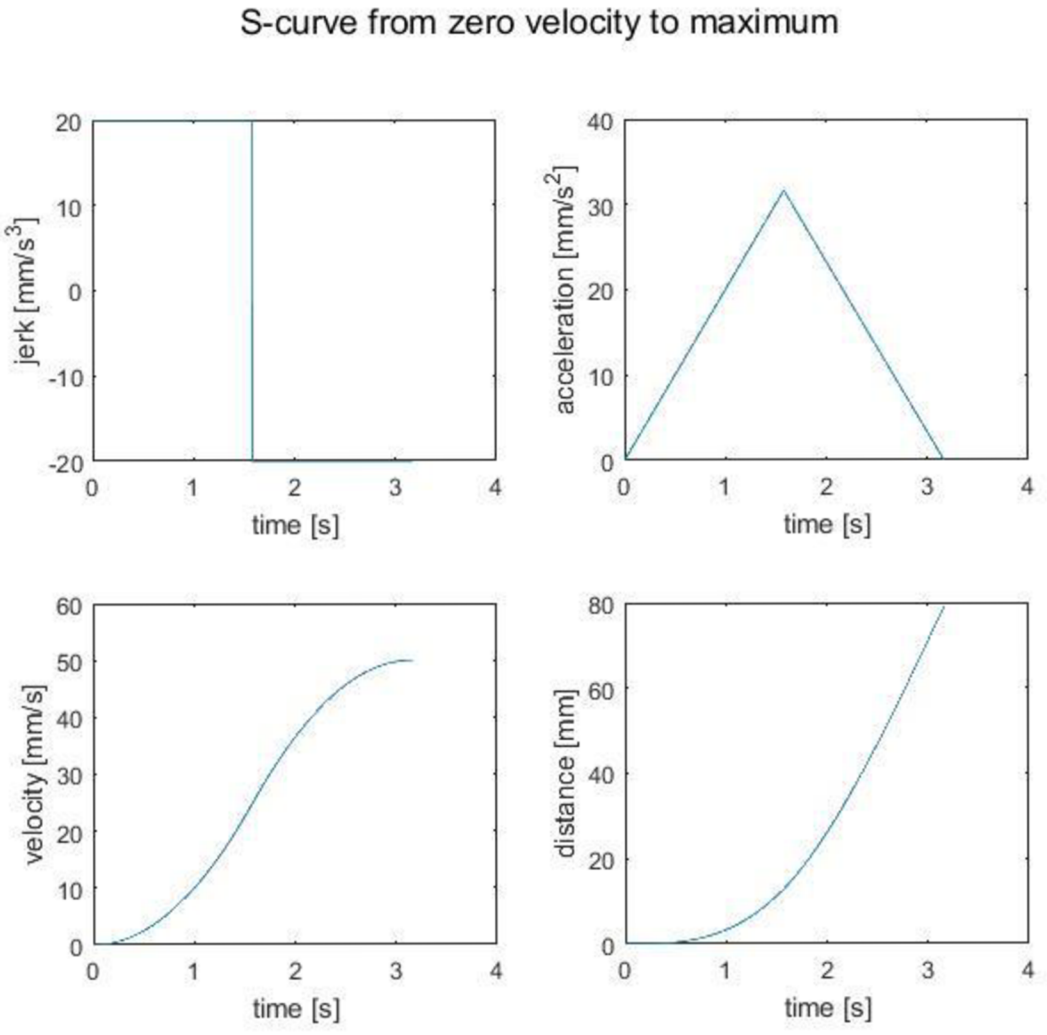
Při modelování S křivky pomocí diferenciálních rovnic se vycházelo ze vztahů uvedených v kapitole 2.3.4, jako časová diference byla zvolena konstanta, která je i zároveň časovou diferencí v simulaci, aby nedocházelo ke ztrátě kroku v důsledku podvzorkování. Hodnota časové konstanty je t = 1e-6 s.

Ukázka části kódu pro výpočet S-křivky pomocí polynomů:

```
j_max = 20; %mm/s3
v_max = 50; %mm/s
% Time step
t = 1e-5; %s

T_full = (sqrt(4*(v_max-v0)*abs(j_max)))/abs(j_max);

T_middle = T_full/2;

T_length = T_full/t;

for i = 1:length(T)

    if T_middle > t_compare
        j_max = abs(j_max);
    else
        j_max = -abs(j_max);
    end

    if (i == 1)
        a_acc(i) = a0 + j_max*t;
        v_acc(i) = v0 + a_acc(i)*t;
        s_acc(i) = s0 + v_acc(i)*t;
    else
        a_acc(i) = a_acc(i - 1) + j_max*t;
        v_acc(i) = v_acc(i - 1) + a_acc(i)*t;
        s_acc(i) = s_acc(i - 1) + v_acc(i)*t;
    end

    t_compare = t_compare + t;
end
```

V úkázce kódu můžeme vidět inicializaci potřebných proměných. Zde je asi nejdůležitější velikost ryvu, která byla nastavena na hodnotu j_max = 20mm/s^3. V další části můžeme vidět výpočet velikosti časového úseky, za který je extrudér schopen zrychlit z nulové rychlosti na rychlost nastavenou, což je v našem případě v_max = 50 mm/s.

Na obrázku č. 13 je pak zobrazen průběh ryvu, zrychlení, rychlosti a dráhy v závilosti na čase. Celkový čas průběhu je 3.162s.

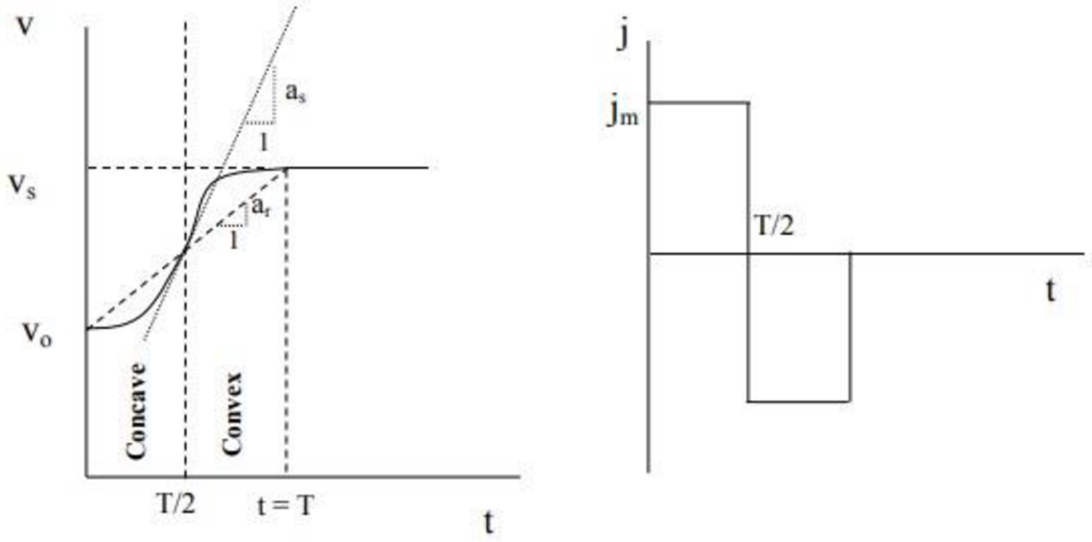


Obr. 13 - Akcelerační část S-křivky z nulové rychlosti na maximum

4.4.2 Pomocí tabulky

Tato část v podstatě vychází z následujících vztahů (4.1) až (4.9) v této podkapitole. Hodnoty celého průběhu S křivky jsou vypočteny předem a následně se jen dosazují do interpolovaných segmentů. V tabulce jsou vypočteny hodnoty absolutního času a okamžité rychlosti v jednotlivých krocích. Tabulka je pouze jedna a to pro akceleraci. Pro zpomalení je potřeba tabulku ‘otočit‘. Tedy brát hodnoty od poslední k první a ještě od nich odečítat celkový čas náběhu.

Na obrázku 14 můžeme vidět význam jednotlivých parametrů a průběh rychlosti, zrychlení a ryvu v čase.



Obr. 14 - S křivka - průběh rychlosti, zrychlení a ryvu v čase [20]

Rovnice pro výpočet hodnot S-křivky:

$$j_m = 2\,a_s/T \tag{4.1}$$

Konkávní část průběhu:

$$s(t) = v_o t + j_m t^3/6 \tag{4.2}$$

$$v(t) = v_o + j_m t^2/2 \tag{4.3}$$

$$t = \mathrm{sqrt}\,(2\,\Delta v/j_m) \tag{4.4}$$

$$s(T/2) = s_h = [v_0 + a_s\,2\,/(6\,j_m)]\,a_s/\,j_m \tag{4.5}$$

Konvexní část průběhu:

$$s(t) = v_0 t + a_s t^2/2 - j_m t^3/6 \tag{4.6}$$

$$v(t) = v_0 + a_s t - j_m t^2/2 \tag{4.7}$$

$$t = [a_s - \mathrm{sqrt}(a_s{}^2 - 2\,j_m\,\Delta v]/j_m \tag{4.8}$$

$$s(T/2) = [v_h + a_s{}^2/(3\,j_m)]\,a_s/j_m \tag{4.9}$$

, kde:

j_m … maximální ryv

a_s … maximální zrychlení

T … celkový čas náběhu

s(t) … dráha v čase t

v(t) … rychlost v čase t

v_0 … počáteční rychlost

S(T/2) … délka dané části

Kód pro výpočet look-up tabulky potřebné pro řízení krokových motorů pomocí S-křivky:

```
LU_jmax = 20;                          % maximal jerk
LU_vmax = 50;                          % maximal speed

LU_vh = LU_vmax/2;                     % speed in half of the curve

LU_t1 = sqrt((2*(LU_vh))/LU_jmax); % time of the concave part
LU_T = LU_t1*2;                        % full time of acceleration

LU_as = (LU_T*LU_jmax)/2;              % maximal acceleration
LU_sh = ((LU_as^2)/(6*LU_jmax))*(LU_as/LU_jmax); % distance of concave part
LU_sh_step = round(LU_sh*20);          % number of steps in concave part

LU_t = 1e-6;


for i = 1:LU_sh_step

        LU_table(i,1) = i;

        while LU_s < i
            LU_s = ((LU_jmax*LU_time^3)/6)*20;
            LU_time = LU_time + LU_t;
        end

        LU_table(i,2) = LU_time - LU_t;
        LU_table(i,3) = LU_jmax*(LU_table(i,2)^2)/2;
        LU_table(i,4) = LU_s;

end



LU_sf = (LU_vh + (LU_as^2)/(3*LU_jmax))*(LU_as/LU_jmax);
LU_sf_step = round(LU_sf*20); % number of steps in convex part


for i = 1:LU_sf_step

        LU_table(i+LU_sh_step,1) = i+LU_sh_step;

        while LU_s < i
            LU_s = (LU_vh*LU_time + (LU_as*LU_time^2)/2 -
            (LU_jmax*LU_time^3)/6)*20;
            LU_time = LU_time + LU_t;
        end

        LU_table(i+LU_sh_step,2) = LU_table(LU_sh_step,2) + LU_time - LU_t;
        LU_table(i+LU_sh_step,3) = LU_vh + LU_as*(LU_time - LU_t) -
        (LU_jmax*(LU_time - LU_t)^2)/2;
        LU_table(i+LU_sh_step,4) = LU_s;

end
```

Ukázka části výpočtené look-up tabulky:

LU_table		
Step	Time	Velocity
1	0,24662	0,60822
2	0,31072	0,96549
3	0,35569	1,26515
4	0,39149	1,53262
5	0,42172	1,77845
6	0,44814	2,00830
7	0,47177	2,22567
8	0,49324	2,43289
9	0,51299	2,63162
10	0,53133	2,82312
11	0,54848	3,00831
12	0,56462	3,18798
13	0,57989	3,36271
14	0,59439	3,53303
15	0,60822	3,69933
16	0,62145	3,86196
17	0,63413	4,02125
18	0,64633	4,17744
19	0,65808	4,33076
20	0,66943	4,48141
21	0,68041	4,62958
22	0,69104	4,77540
23	0,70136	4,91903
24	0,71138	5,06060
25	0,72112	5,20021
26	0,73062	5,33798
27	0,73986	5,47399
28	0,74889	5,60833
29	0,75770	5,74108
30	0,76631	5,87231

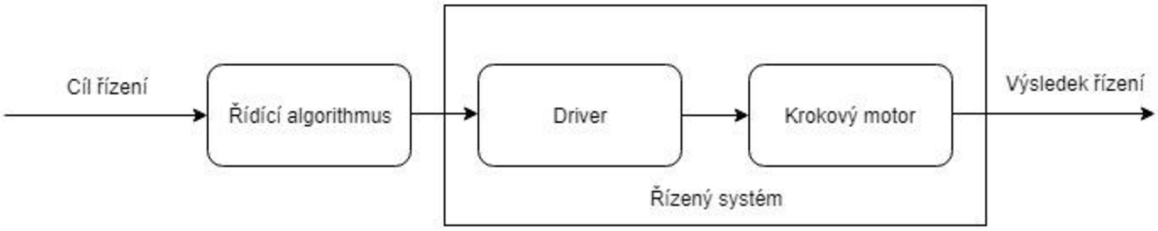
Tab. 5 - Look-up tabulka

5 Simulace

V předchozí kapitole byl namodelován systém dvou osého CNC zařízení a algoritmy pro jeho řízení. V této kapitole bude provedena simulace pro ověření funkčnosti řídícího algoritmu. Boudou provedeny celkem tři typy simulací:

1. MIL = model in the loop,
2. SIL = software in the loop,
3. PIL = processor in the loop.

Ve skutečnosti jsou však 3D tiskárny, respektive její krokové motory, řízeny v tzv. přímé vazbě.



Obr. 15 - Řízení v přímé vazbě

Z obrázku 15 je patrné, že v přímé vazbě je absence zpětné vazby a tedy se nejedná v pravém slova smyslu o „smyčku“ (loop).

Algoritmus vytvořený v této diplomové práci už s tímto počítal. Řídící algoritmus se tedy provede jen jednou na začátku, vypočítá všechny potřebné pohyby krokových motoru na základě vstupních G-kódů, uloží je a poté jen v určitých časech posílá do driveru informace o směru pohybu a počtu kroků pro konkrétní krokový motor.

Řízení bez zpětné vazby má značné nevýhody, ale u krokových motorů řízených pomocí rychlostních profilů tvaru S-křivky nedochází k velkým rázům a tedy ke ztrátě kroků, proto si tento typ řízení můžeme u 3D tiskárny dovolit.

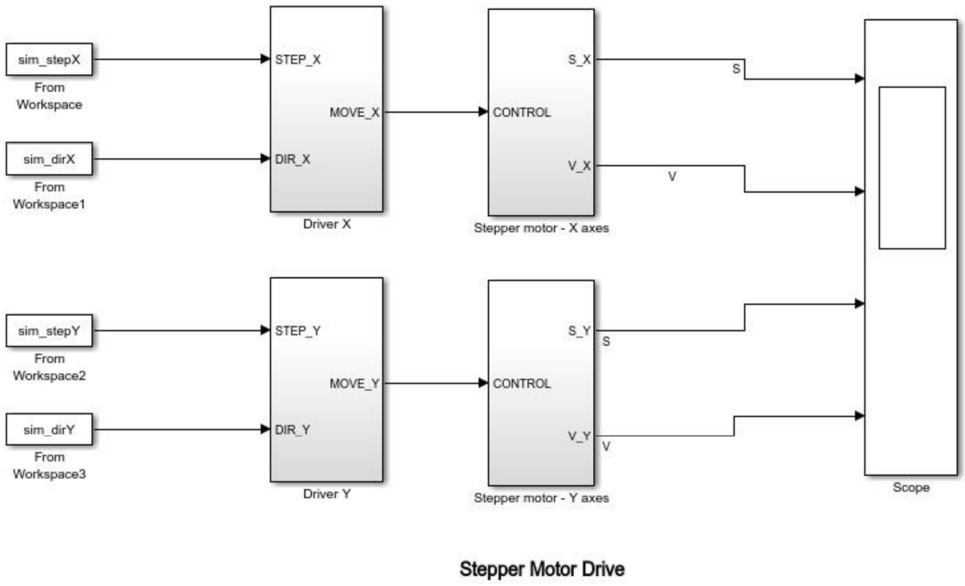
5.1 MIL

Matematický model i řídicí systém jsou simulovány na jednom počítači v prostředí Matlab a Simulink. Tento typ simulace slouží k ověření proveditelnosti a optimalizaci technologie, definování nutných vstupních a výstupních signálů a návrhu struktury řízení. [17]

Při tomto typu simulace byl řídící algoritmus spuštěn přímo v prostředí Matlab/workspace a pomocí algoritmů z předchozí kapitoly se vypočítaly řídící vektory. Poté byl spuštěn model soustavy v prostření Simulink, přičemž výstupní hodnoty se předaly přes celkem pět vektorů do Simulinku.

Výstupní vektory z řídícího algoritmu/vstupní vektory do Simulinku:

- sim_stepX – obsahuje kroky na X-ové ose,
- sim_stepY – obsahuje kroky na Y-ové ose,
- sim_dirX – obsahuje směr pohybu na X-ové ose,
- sim_dirY – obsahuje směr pohybu na Y-ové ose,
- sim_TIME – obsahuje absolutní čas.



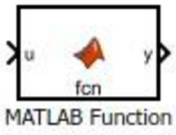
Obr. 16 - MIL model soustavy

Na obrázku 16 můžeme vidět model soustavy pro MIL simulaci. Model driveru, který na základě signálů z řídícího algoritmu posílá informace o krocích do krokového motoru v určitých časových okamžicích a model samotného motoru, který počítá jednotlivé kroky a přes konstanty související s vlastnostmi motoru (počet kroků na otáčku a vzdálenost na otáčku) počítá uraženou dráhu. Z dráhy se pak vypočítává rychlost ze změny dráhy v čase.

5.2 SIL

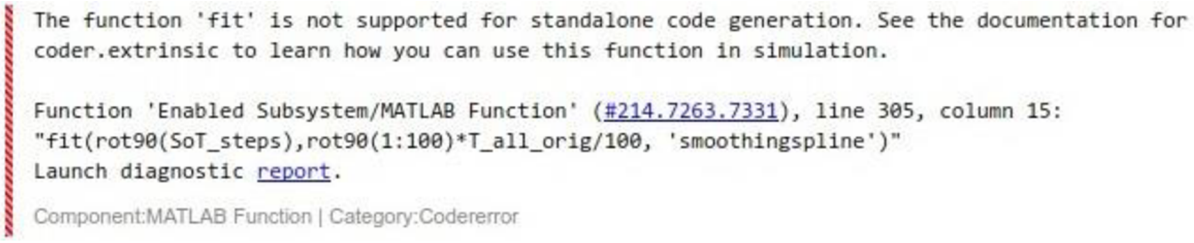
SIL simulace nevyužívá žádný speciální hardware nebo software. Řídící software i simulovaný model jsou na jednom počítači. Simulace by měla ověřit funkčnost algoritmu a zjistit s jakou periodou vzorkování je možné danou soustavu řídit. Na základě této hodnoty se pak může zvolit vhodný HW. [18]

Kvůli tomuto typu simulace musel být řídící algoritmus převeden z Matlabu do samostatné funkce v prostředí Simulink. K tomuto účelu se nejlépe osvědčila tzv. MATLAB Function.



Obr. 17 - MATLAB Function

Ani ta však nedokázala provést všechny funkce, které byly použity pro řízení v předchozím typu simulace. Tady je jedna z chyb, které tato funkce vracela. Šlo o chybu funkce fit, pomocí které se aproximoval počet stepů za čas a obráceně při výpočtu rychlostního profilu s tvarem S-křivky.



Obr. 18 - Chyba funkce fit v bloku MATLAB Function

Z toho důvodu bylo nutné funkci fit nahradit tabulkou a v podstatě se nejedná ani o zjednodušení, které by jakýmkoliv způsobem ovlivňovalo výsledek, protože v tabulce jsou všechny kroky od 1 až do 1600, což je dokonce více, než je maximální počet kroků potřebný k dosažení maximální rychlosti z klidového stavu. Pro snažší srovnání je simulace provedena jen pro první segment pohybu, na kterém je dobře vidět pohyb po S křivce v obou simulovaných osách. Stejný segment byl použit i pro MIL a PIL simulaci.

Dále bylo nutné zajistit, aby se funkce vykonala pouze jednou a vypočítala hodnoty pro řízení modelu. Pro dosažení tohoto cíle byla MATLAB Function uzavřena do subsystému, který se spouští pouze na impuls. Impuls byl tedy nastaven do jedničky v čase 0 a po jednom vzorku simulace byl opět shozen na nulu, to zajistilo jednorázové spuštění. Dalšího cíle, tedy odsimulování modelu a řízení v jedné simulaci s tímto typem řízení, se však dosáhnout nepodařilo. Bylo totiž nutné proměnné někde uchovat a do simulace modelu je posílat postupně.

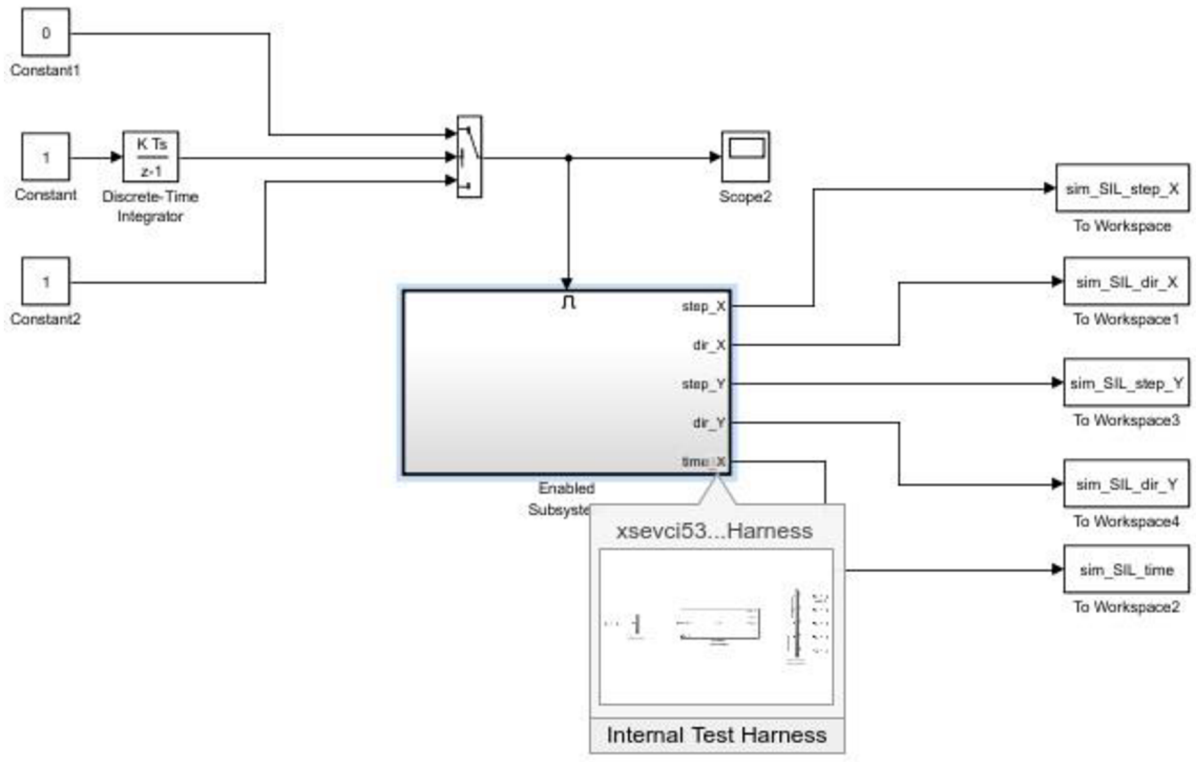
Nakonec zvítězilo řešení, kde jsou hodnoty ze SIL simulace odeslány do prostředí Matlab workspace a následně je spuštěna druhá simulace, tentokrát simulace soustavy, která s těmito hodnotami pracuje.

Není to úplně ideální řešení, ale podle mého názoru je to při řízení v přímé vazbě přinejmenším řešení dostatečné, protože řídící algoritmus nejdříve spočítá kolik má který motor udělat kroků, kterým směrem a v jakém čase a až poté začne posílat příkazy driveru a ten ovládá krokové motory na základě těchto příkazů bez jakékoliv zpětné vazby. Jedná se tedy vlastně o dvě samostatné simulace – první simuluje výpočet hodnot pro řízení a ta druhá simuluje řízení krokových motorů a pohyb extrudéru.

SIL simulace byla provedena dvěma způsoby. Poprvé přes tzv. Test Harness prostředí, které se používá pro testování a editaci modelu v Simulinku a podruhé přes SIL Block model, který je vytvářen přímo ze subsystému, a po vytvoření je jím tento subsystém nahrazen – toto (druhé) řešení bylo zvoleno proto, protože obdobným způsobem byla provedena i PIL simulace.

Test Harness

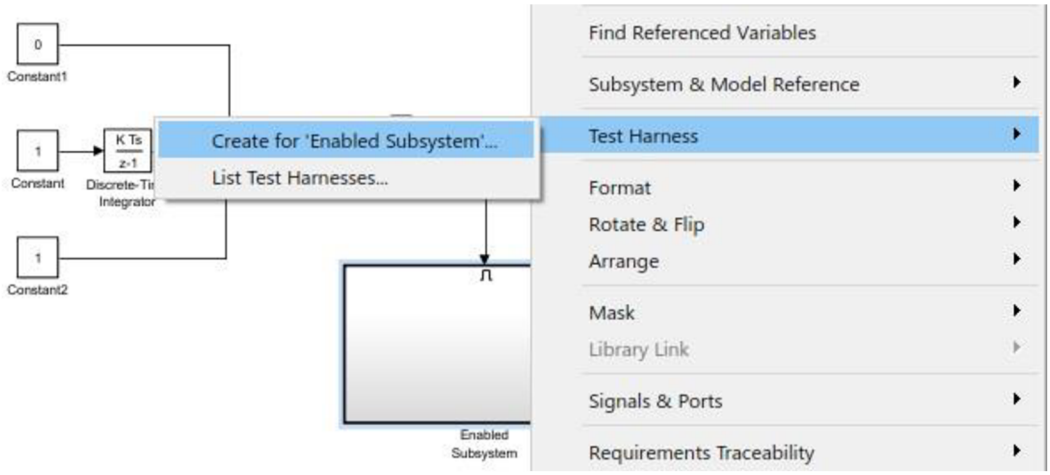
Na obrázku č. 19 můžeme vidět simulaci se subsystémem, uvitř kterého je umístěn blok s MATLAB funkcí se SIL simulací.



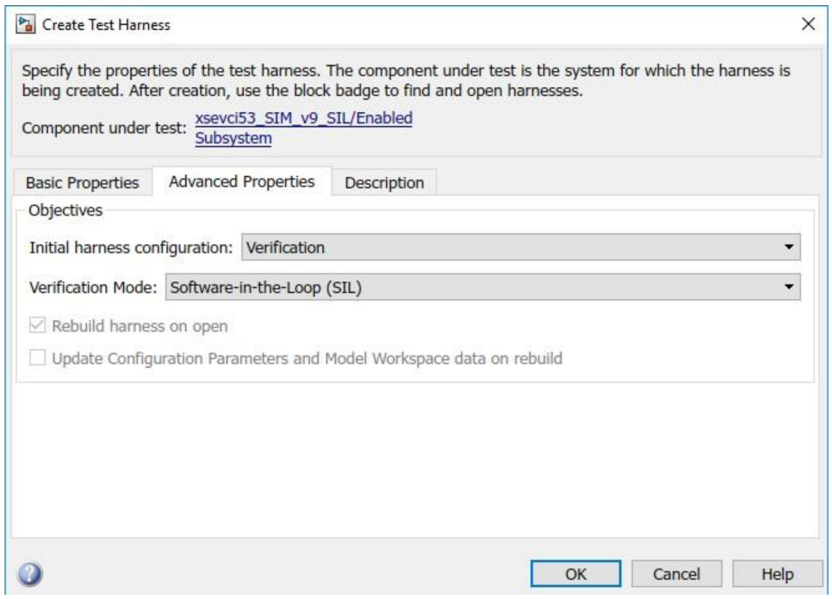
Obr. 19 - SIL simulace řídícího algoritmu

Pro vytvoření SIL bloku v simulaci bylo nutné projít následující kroky:

1. V Simulinku v Configuration Parameters v záložce Code Generation nastavit vhodný embedded compiler.
2. Poté pravým tlačítkem kliknou na subsystem a vybrat záložku Test Harness a Create for <jméno subsystému> (obr. 20).
3. A nakonec ve vyskakovacím okně nastavit v záložce Advanced Properties parameter Initial harness configuration jako Verification a Verification mode jako Software in the Loop (SIL) (obr. 21).

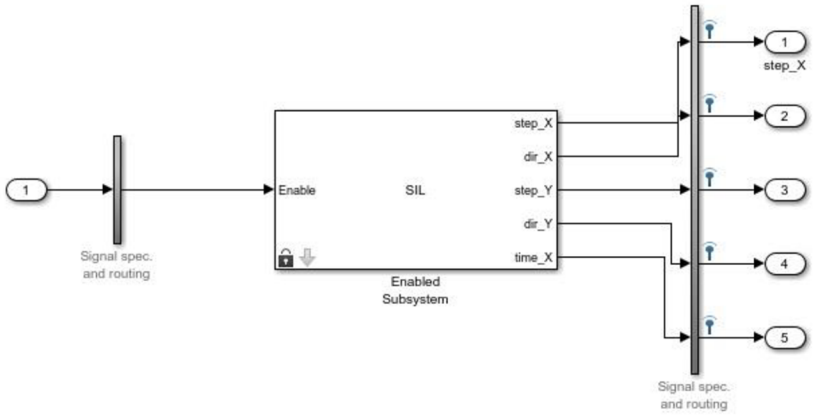


Obr. 20 - Vytoření Test Harness pro SIL simulace



Obr. 21 - Konfigurace Test Harness

Na obrázku č. 22, pak můžeme vidět, jak vypadá vytvořený SIL block v prostředí Test Harness nahrazující subsystému.



Obr. 22 - SIL block v Test Harness vytvořený uvnitř subsystému

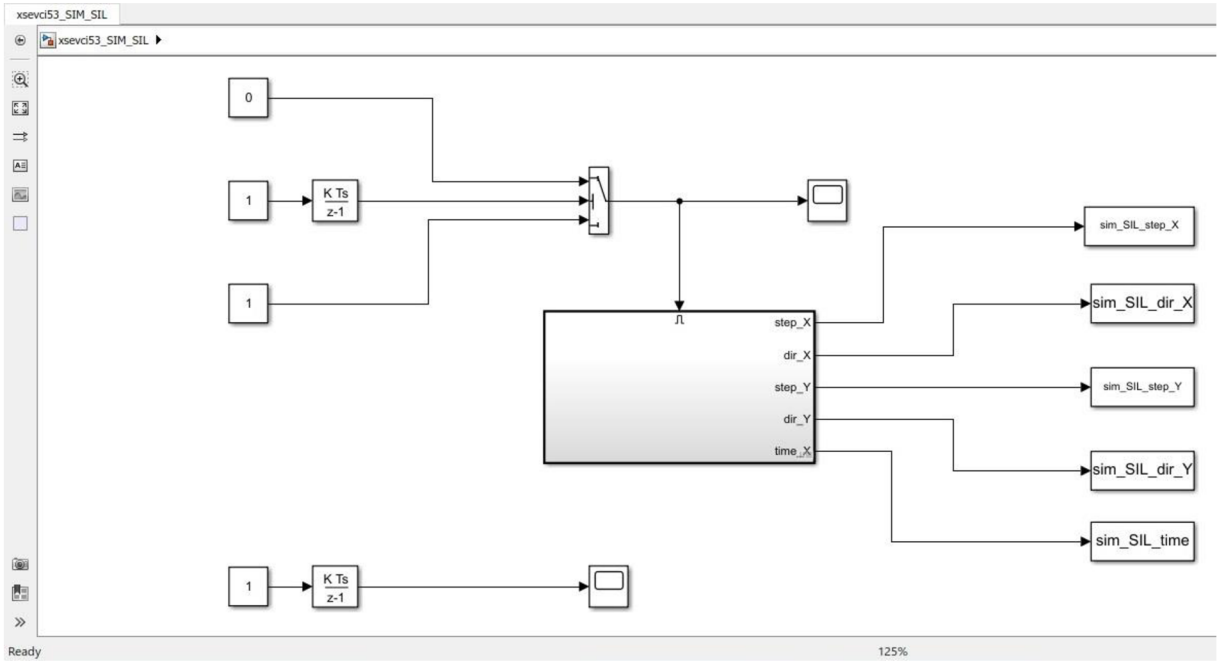
SIL Block model

Výchozí simulace je naprosto stejná jako v předchozím případě. Další kroky už jsou však odlišné a pro vytvoření simulace bylo potřeba provést následující:

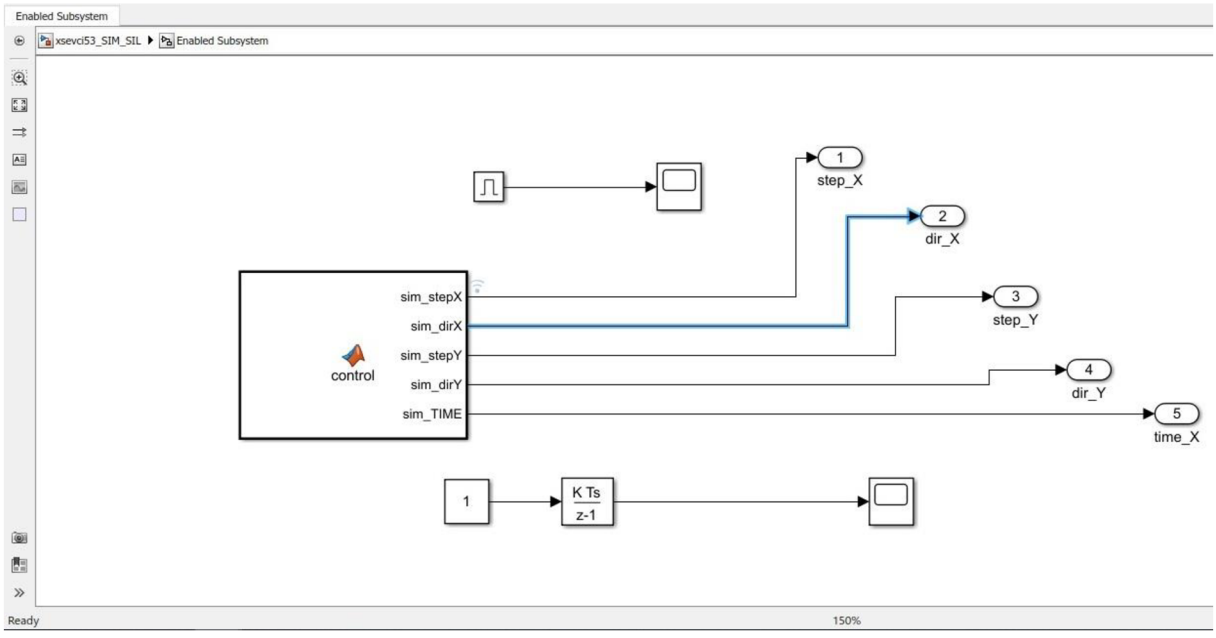
1. Model Configuration Parametrs
   1.1. Hardware Implementation
      1.1.1. Hardware board – vybrat odpovídající typ desky – STM32 F411RE
   1.2. Code Generation
      1.2.1. Toolchain settings – zvolit toolchain – GNU Tools for ARM Embedded Processors
      1.2.2. Interface – v sekci Support enviroment – zatrhnout variable-size signals
      1.2.3. Verification – a nakonec zvolit simulaci SIL

Po nastavení parametrů je nutné kliknout pravým tlačítkem na subsystém > C/C++ Code > Deploy this Subsystem to Hardware a ve vyskakovacím okně je potřeba stisknout tlačítko Build a SIL blok se následně vygeneruje v novém okně. Starý blok je pak potřeba smazat a na jeho místo vložit nově vygenerovaný SIL blok. Simulace se pak spouští standardně tlačítkem pro start simulace.

Na obrázku 23 můžeme vidět výchozí simulaci se subsystémem před vygenerováním SIL bloku. Na obrázku 24, je pak zobrazen vnitřek subsystému.

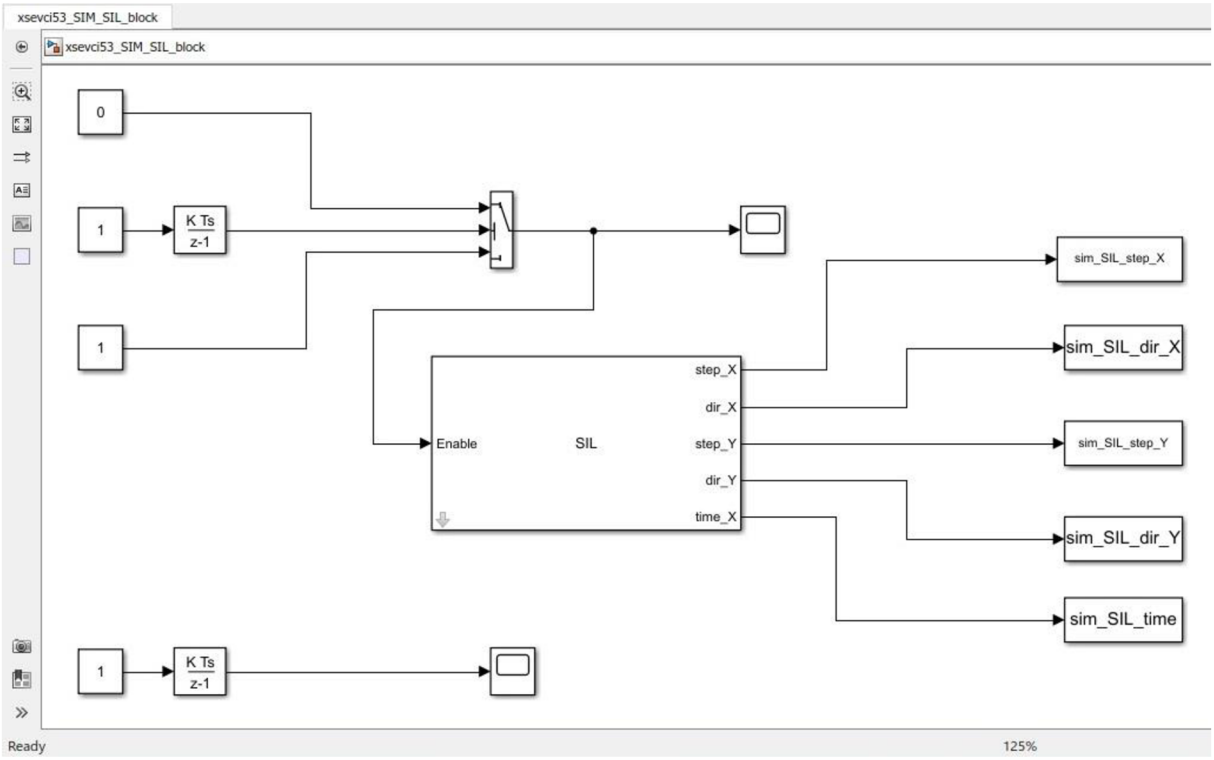


Obr. 23 – SIL simulace výpočtu řídícího algoritmu před generováním SIL bloku



Obr. 24 - Obr. 23 - SIL simulace výpočtu řídícího algoritmu - uvnitř subsystému

Na obrázku č. 25 už je vygenerovaný SIL blok vložen v simulaci namísto původního subsystému.

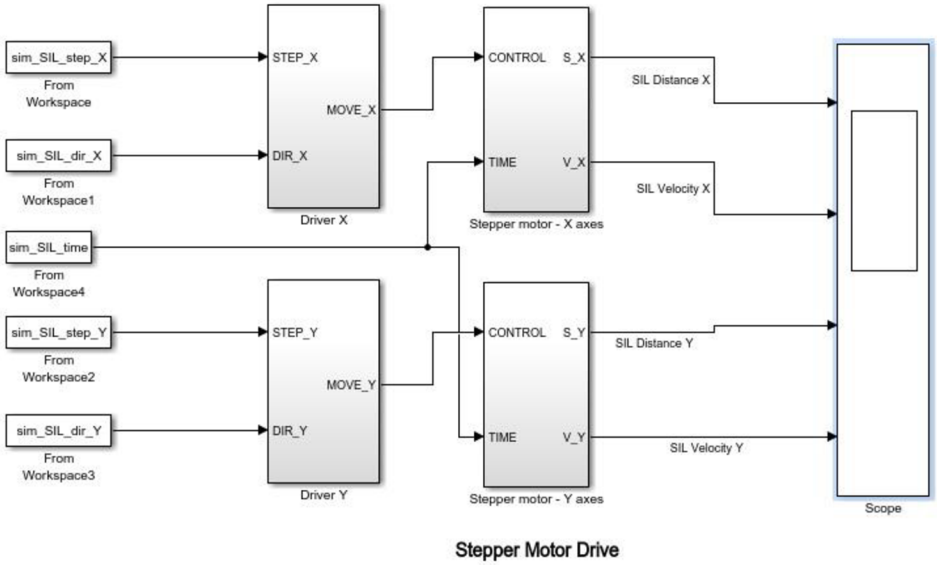


Obr. 25 - SIL simulace výpočtu řídícího algoritmu po vygenerování SIL bloku

Jak se očekávalo, tak pro oba typy testovaných SIL simulací dostáváme naprosto stejné výsledky, proto byl nakonec v programu pro přehlednost ponechán jen druhý případ, který je typově shodný s provedením následující PIL simulace.

Výsledný formát řídícího algoritmu je stejný jako v případě MIL simulace. Máme tedy dva vektory pro každý ze dvou krokových motorů a jeden časový vektor. První vektor obsahuje jednotlivé kroky krokového motoru a k nim přiřazené absulutní časy, kdy k nim má dojít. Druhý vektor pak slouží k ovládání směru pohybu krokového motoru a třetí vektor obsahuje časové hodnoty, které se používají k výpočtu rychlosti z uražené dráhy.

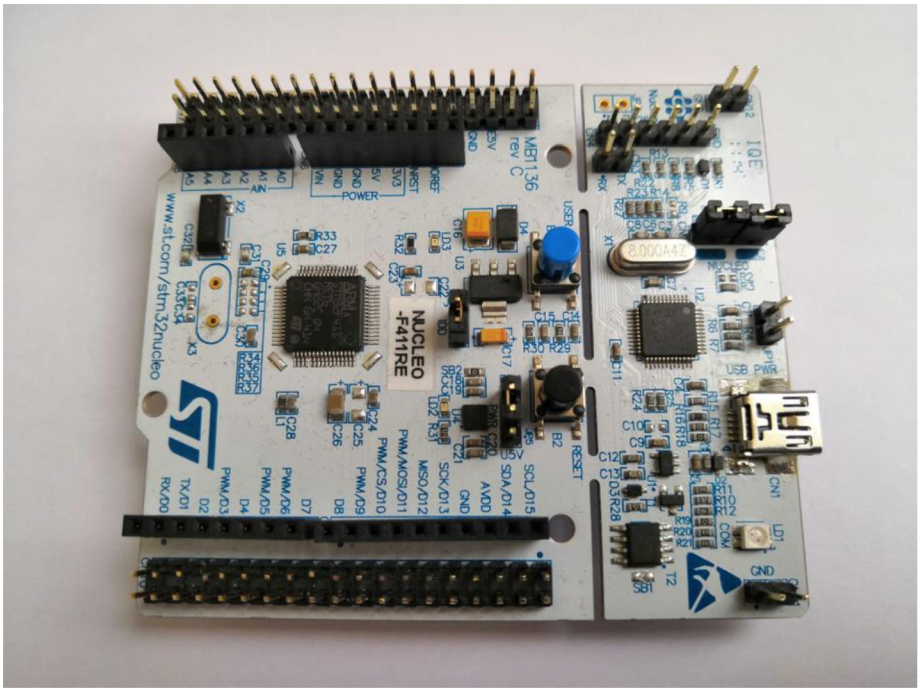
Tyto vektory se odesílají do SIL simulace modelu, která je obdobná MIL simulaci modelu, až na jiné vstupní vektory. Výsledky simulace budou zobrazeny a porovnány v následující kapitole.



Obr. 26 - SIL simulace modelu soustavy

5.3 Nahrání programu do vývojového kitu STM32F Nucleo – F411RE

V této práci byla použita běžne dostupná vývojová deska s označením STM32 Nucleo – F411RE (obr. 27) od firmy STMicroelectronics.



Obr. 27 - STM32 Nucleo – F411RE

Desku lze připojit k počítači pomocí mini-USB a po nainstalování příslušných podpůrných balíčků a knihoven lze na desku nahrát námi vytvořený řídící software přímo z prostření Matlab a Simulink.

Support packages potřebné ke komunikaci a správnému fungování vývojového kitu:

- Simulink Coder Support Package for STMicroelectronics Nucleo Boards
  Tato knihovna umožňuje vytvářet model v Simulinku a následně vygenerovat a nahrát kód do podporovaných desek s označením Nucleo. Tento podpůrný balíček obsahuje knihovnu obsahující Simulinkové bloky pro virtuální přístup k periferiím na procesoru a na desce.
- STM32CubeMX
  Tento nástroj umožňuje konfiguraci STM32 mikrokontrolérů a mikroprocesorů a generování korespondujícího C kódu pro ARM Cortex.
- STM32-MAT/TARGET
  Tento software umožňuje nahrání aplikace z programu Matlab a Simulink do STM32 MCU. Dále zajišťuje aplikacím simulovaným v STM32 MCU propojení s modelem v Simulinku.

Matlab, přesněji řečeno Java, stahování a instalacím těchto balíčků velmi často brání a vrací nejrůznější chybová hlášení. Nejinak tomu bylo i při vypracování této diplomové práce. Pro úspěšné nainstalování všech podpůrných balíčků bylo nutné jít do chybového logu Javy, najít o jakou chybu se při stahování jedná a dopátrat se vhodného řešení problému. Při této práci se vyskytlo takových chyb hned několik, ale ta nejzásadnější byla, že Java nedokázala stáhnout software třetí strany.

Toto je část výpisu z chybového logu:

...

Exiting with status 0

(III 06, 2019 18:46:21) End - Successful.

(III 06, 2019 18:46:21) Assembling product list...

(III 06, 2019 18:46:21) Starting local product/component search in download directory

(III 06, 2019 18:46:21) Searching for archives...

(III 06, 2019 18:46:21) Reading C:\Users\Martin\Downloads\MathWorks\SupportPackages\R2016b\archives

(III 06, 2019 18:46:21) 59 files found in C:\Users\Martin\Downloads\MathWorks\SupportPackages\R2016b\archives

(III 06, 2019 18:46:21) Reading C:\Users\Martin\Downloads\MathWorks\SupportPackages\R2016b

(III 06, 2019 18:46:21) 1 files found in C:\Users\Martin\Downloads\MathWorks\SupportPackages\R2016b

(III 06, 2019 18:46:21) Archive search complete. 60 total files found.

(III 06, 2019 18:46:21) Completed local product/component search

(III 06, 2019 18:46:21) Exiting with status 0

(III 06, 2019 18:46:21) End - Successful.

(III 06, 2019 18:46:41) **Download Error: There was a problem downloading the third-party software. To resolve this issue, contact Technical Support**

...

Caused by: javax.net.ssl.SSLException: Received fatal alert: protocol_version

...

Tuto chybu se podařilo odstranit instalací podpůrného balíčku z následující internetové stránky https://www.mathworks.com/support/bugreports/1741173 a za pomoci návodu nakopírovat soubory na příslušné cesty v Matlabu.

S veškerým potřebným a úspěšně nainstalovaným softwarem, kompatibilním s danou verzí Matlabu a Windows, pak už nebyl větší problém nahrát program simulace do vývojového kitu přímo z prostředí Simulink a dokončit tak i PIL simulaci.

5.4 PIL

Matematický model soustavy je simulován v programu (Matlab & Simulink) v počítači a výpočet řídícího vektoru pro řízení krokových motorů je proveden na reálné hardwarové platformě. Nejsou použity žádné vstupní nebo výstupní porty. Všechna komunikace mezi modelem a hardwarem probíhá pomocí USB nebo průmyslové komunikace. Úkolem této simulace je především ověření výpočetního výkonu procesoru a otestování kritických situací, jako jsou například různé chybové stavy. [19]

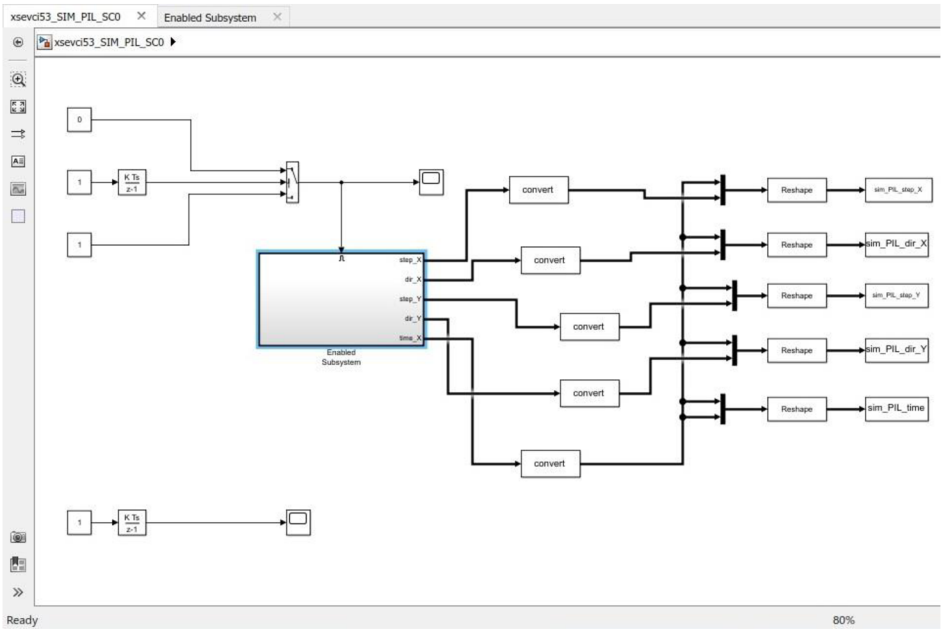
Koncept dvou samostatných simulací pro řízení a pro model soustavy je obdobný jako v případě SIL simulace. Ovšem funkce simulující řídící algoritmus, musela být do značné míry upravena, protože v průběhu nahrávání programu do vývojového kitu bylo emiricky zjištěno, že vygenerovaný program v jazyce C je moc velký, a že počet dat potřebných pro jeho zpracování a výsledné tabulky řídících vektorů se nevejdou do RAM paměti. Z toho důvodu byl odstraněn všechen nadbytečný kód, který sloužil k zobrazování mezivýpočtů a debuggingu, dále byly smazány některé proměnné a především bylo nutné přetypovat zbylé proměnné a tabulky z double na float (v prostředí Matlab se float označuje jako single) a v případě některých i na typ bool (který se v prostředí Matlab označuje jako logical).

Další změnou oproti, SIL simulaci, byla změna konfigurace simulace řízení.

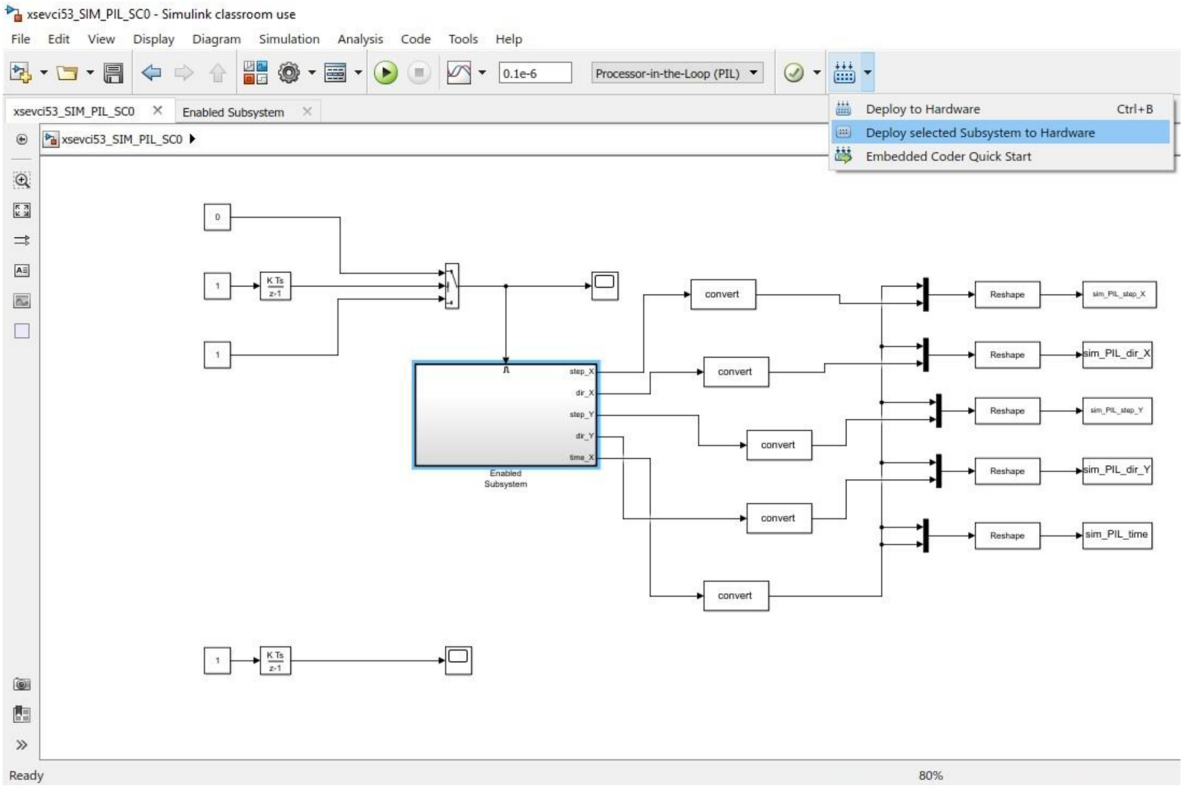
1. Model Configuration Parametrs
   1.1. Hardware Implementation
      1.1.1. Hardware board – vybrat odpovídající typ desky – STM32 F411RE
      1.1.2. Hardware board settings
         1.1.2.1. Target Hardware Resources – zde je potřeba kliknout na PIL a nastavit COM port, na kterém je deska připojena k PC
   1.2. Code Generation
      1.2.1. Toolchain settings – zvolit toolchain – GNU Tools for ARM Embedded Processors
      1.2.2. Interface – v sekci Support environment – zatrhnout variable-size signals
      1.2.3. Verification – a nakonec přepnout simulaci na PIL

A obdobně jako v případě druhého typu SIL simulace je po nastavení všech parametrů vytvořit PIL blok a sice kliknutím pravým tlačítkem myši na subsystém > C/C++ Code > Deploy this Subsystem to Hardware. Nebo lze i subsystém označit, na pracovní ploše stisknout tlačítko Deploy, umístěné v pravé části horního kontrolního panelu (obr. 29). A následně ve vyskakovacím okně zmáčknout tlačítko Build (obr. 30). Tím se vygeneruje PIL blok v novém okně a starý blok je pak potřeba smazat a na jeho místo vložit vygenerovaný PIL blok (obr. 31).

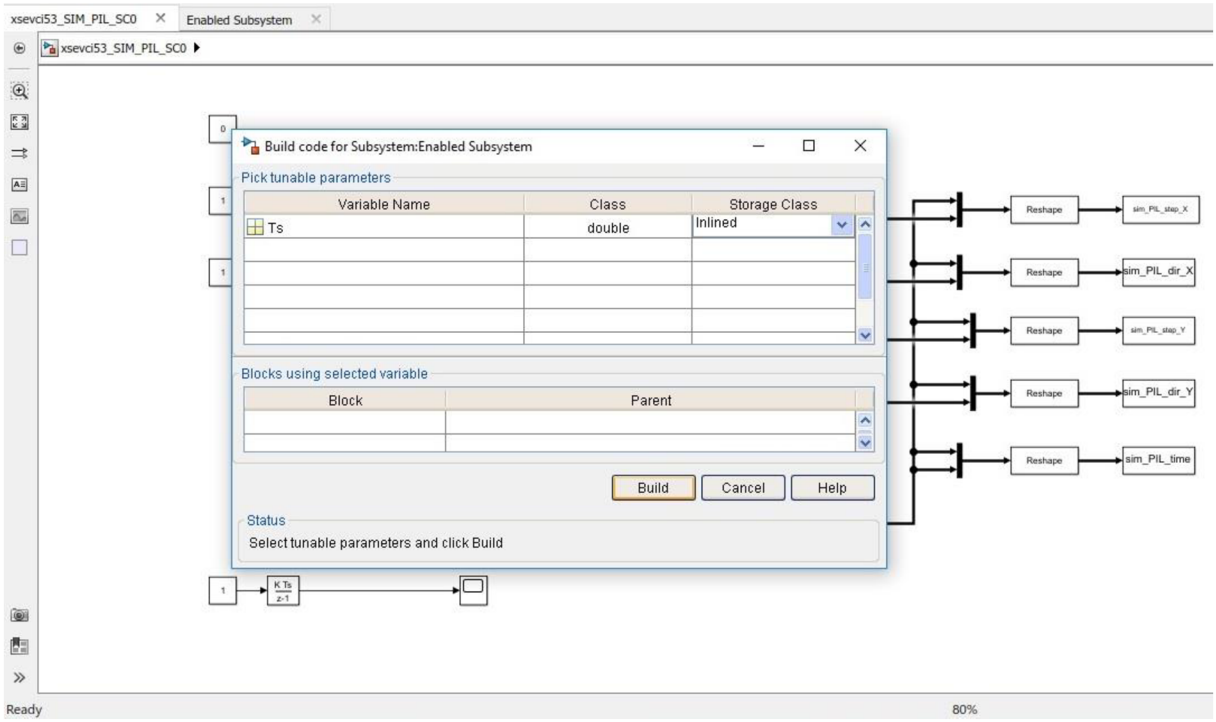
Simulace se opět spustí tlačítkem Play. Při stisknutí tlačítka ‘Play‘ se nejdříve celá simulace zkompiluje, poté se spustí a provádí se až do chvíle, než narazí na PIL blok. Tam se zastaví, nahraje kód do procesoru a předá mu všechny potřebné parametry a čeká na odpověď, po jejím přijetí se pak provede i zbytek simulace a výsledné řídící cektory se uloží do workspaceu.



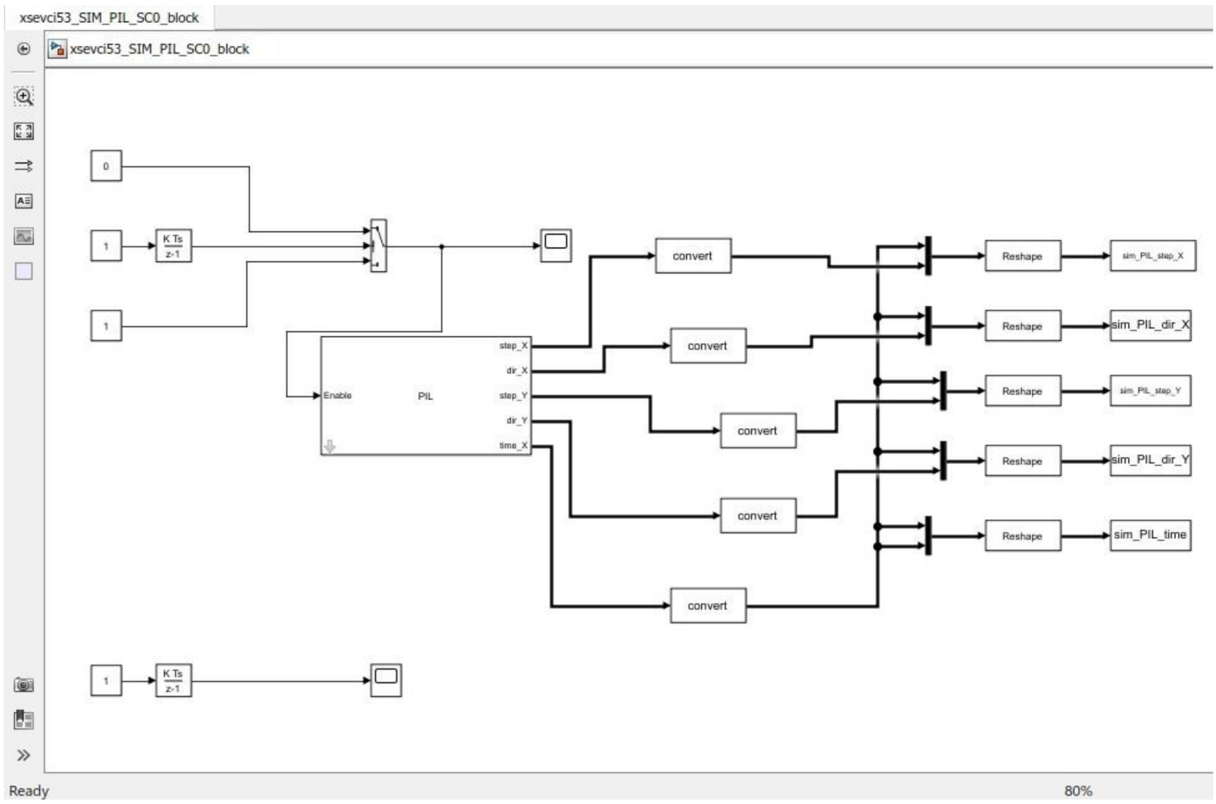
Obr. 28 - PIL simulace výpočtu řídícího algoritmu před vygenerováním PIL bloku



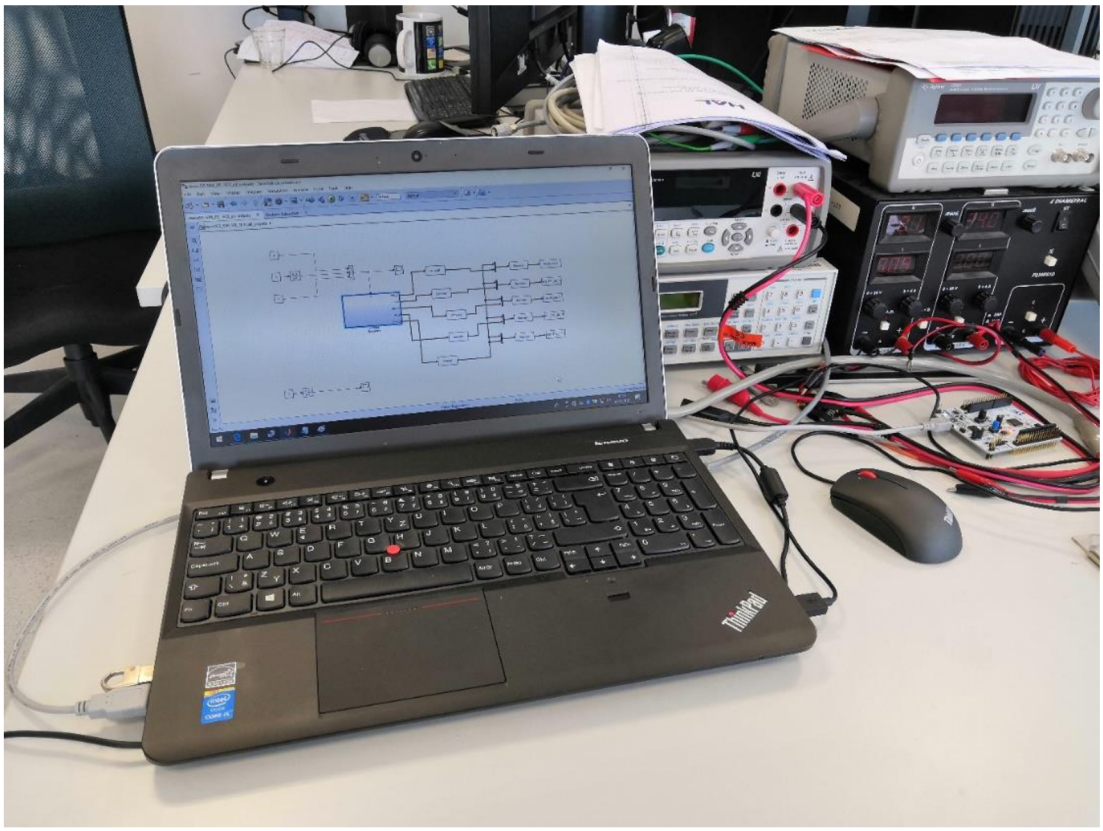
Obr. 29 - PIL simulace výpočtu řídícího algoritmu - vytvoření PIL bloku ze subsystému



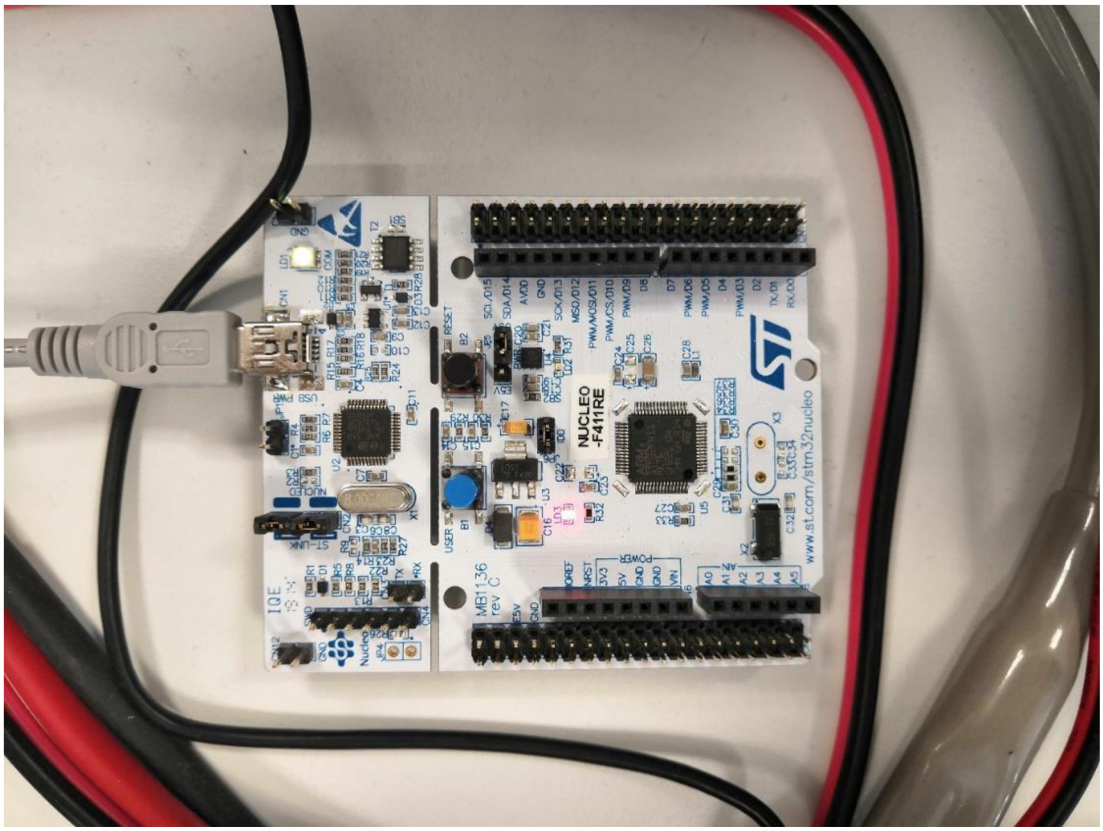
Obr. 30 - PIL simulace výpočtu řídícího algoritmu - generování PIL bloku



Obr. 31 - PIL simulace výpočtu řídícího algoritmu po vygenerování PIL bloku

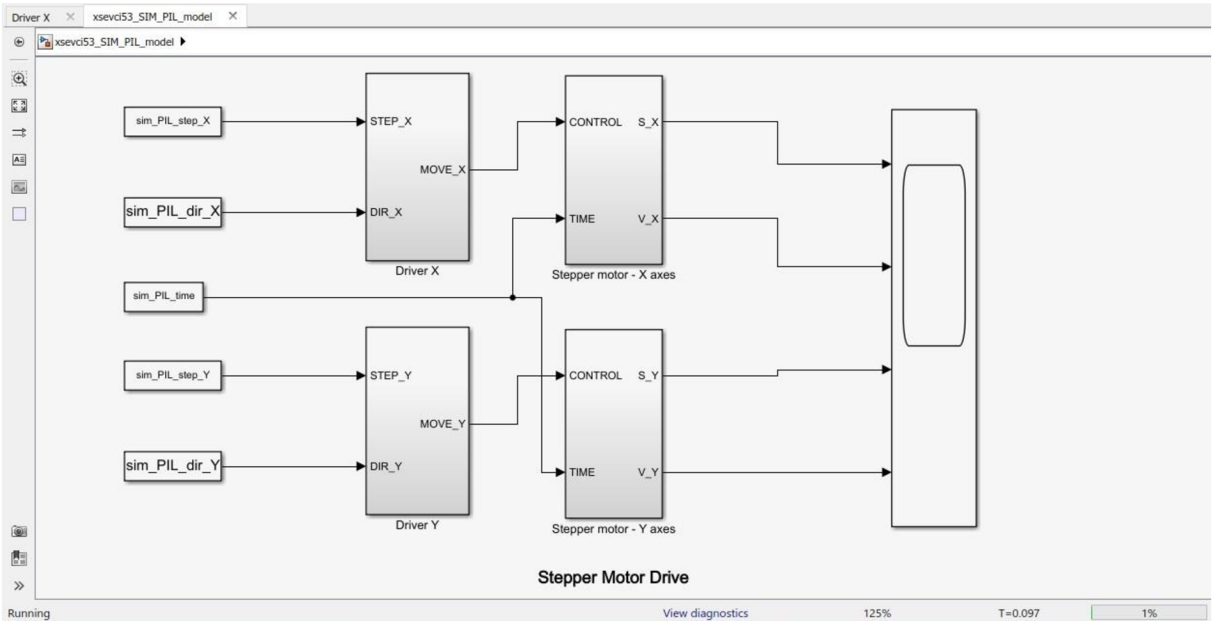


Obr. 32 - Pracoviště při provádění PIL simulace



Obr. 33 - STM32 Nucleo F411RE připojené přes mini USB k PC během PIL simulace

Pro vektory získané touto simulací byl upraven model soustavy (obr. 34) totožný se SIL modelem, pouze byly upraveny jména vstupních proměnných.



Obr. 34 - PIL simulace modelu soustavy

PIL simulace byla úspěšně provedena pro výpočet S křivky pomocí tabulky, protože tento způsob je výpočetně daleko méně náročnější než výpočet pomocí polynomiálních výpočtů, který se ani po zjednodušení nevešel do paměti RAM. Algoritmus pro výpočet polynomů v tom tvaru jak je napsán nebylo možné zredukovat natolik, aby mohl být nahrán do zvoleného hardwaru.

Algoritmus výpočtu řídících vektorů pomocí tabulky se pomocí úprav, popsaných na začátku této podkapitoly, podařilo zmenšit natolik, že program bylo možné bez větších problémů nahrát do vývojového kitu STM32 Nucleo F411RE a provést požadovanou simulaci.

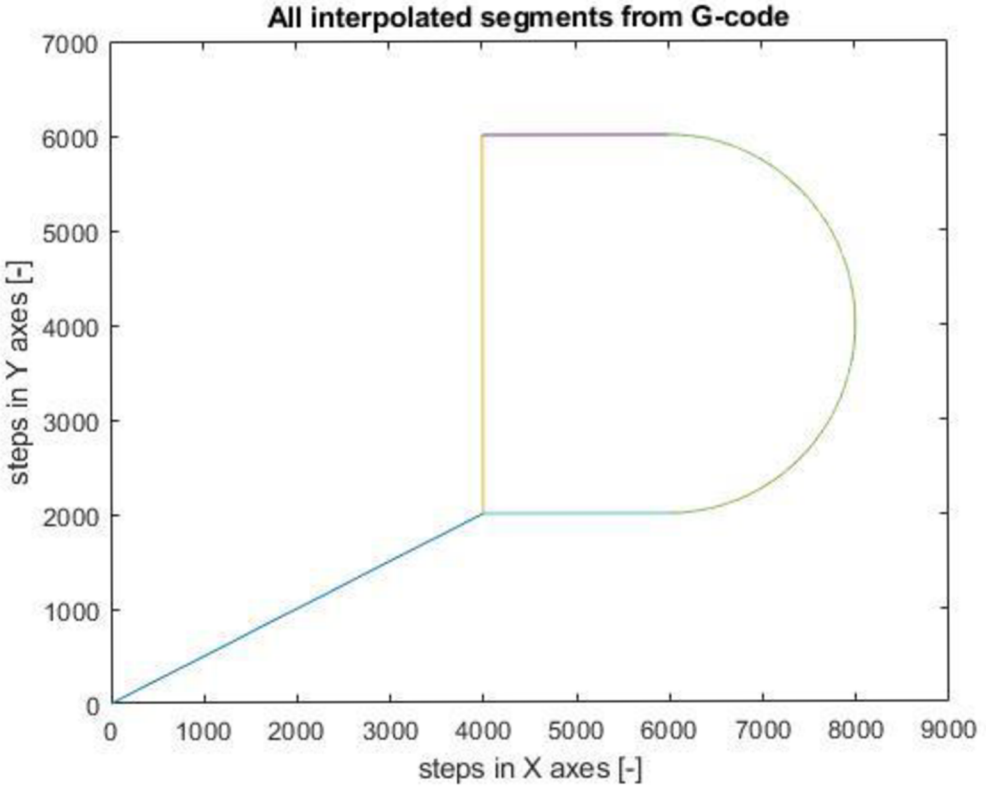
Výsledky simulace pohybu krokových motorů z vypočtených řídících vektorů budou k vidění v následující kapitole.

6 Výsledky simulace

Než přejdeme k výsledkům samotných simulací, tak je na místě si ukázat, co vlastně vstupuje do řídícího algoritmu. Jedná se o tento G kód, jehož význam by nám měl být zřejmý z kapitoly 1.4.1.

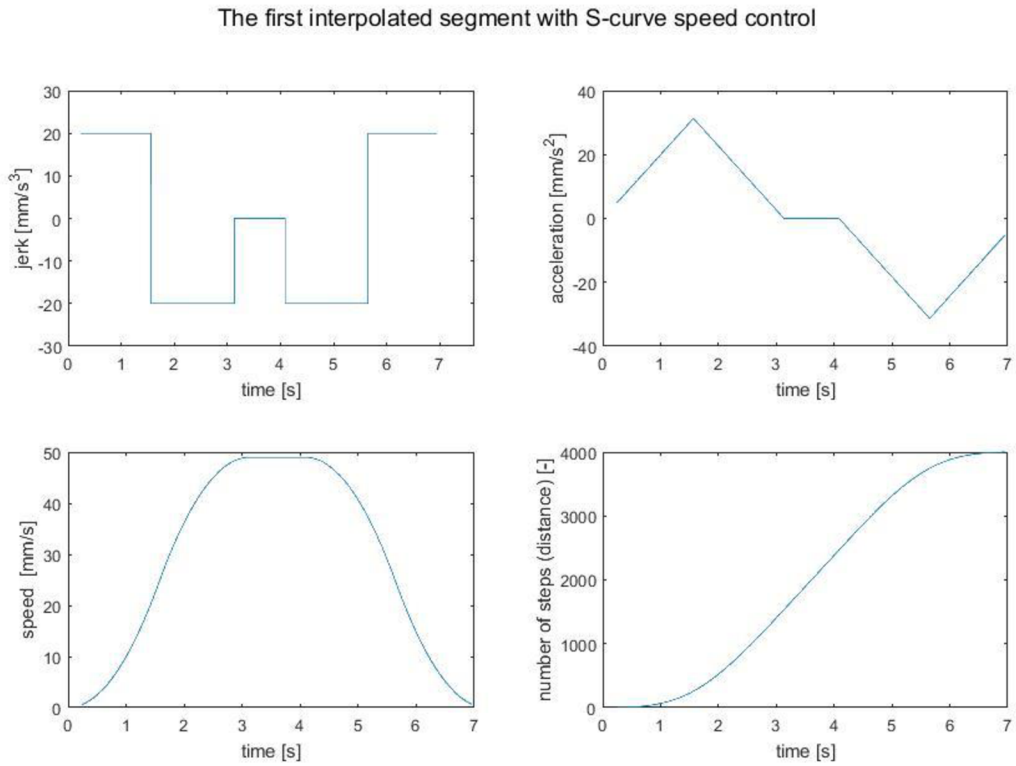
N01 G01 X200 Y100
N02 G01 Z0 F2000
N03 G01 X200 Y300
N04 G01 X300
N05 G02 Y100 I0 J-100
N06 G01 X100
N07 G01 Z10
N08 G00 X0 Y0 Z0

Jeho kompletní vykreslení můžeme vidět na obrázku č. 35. V simulacích nás však bude zajímat pouze první řádek, tedy tmavě modrý segment v obrázku, jdoucí z bodu [0,0] do [4000,2000]. Na něm jde totiž dobře vidět průběh S-křivky pro řízení obou krokových motorů.



Obr. 35 - vykreslení všech interpolovaných segmentů

Vykreslení tohoto segmentu bylo nejdříve provedeno pomocí funkce plot přímo v pracovním prostředí programu Matlab. V obrázku č. 36 tak najdeme průběh všech důležitých veličin (ryvu, zrychlení, rychlosti a dráhy) v čase, které nás při řízení pomocí rychlostního profilu tvaru S-křivky, na tomto segmentu, zajímají. Toto jsou ideální průběhy daných závislostí a těch se budeme snažit v simulacích dosáhnout.



Obr. 36 - Průběh ryvu, zrychlení, rychlosti a dráhy v čase na prvním interpolovaném segmentu

Na obrázcích č. 38 a 39 vidíme výsledky MIL simulace, kde vlevo nahoře je dráha, kterou urazil extrudér na ose X, vlevo dole je jakou rychlostí se při tom pohyboval a v pravém sloupci je to stejné jen pro Y-ovou osu.

Vykreslené křivky byly naprosto totožné pro Bresenhamovu i DDA interpolaci, z toho důvodu nejsou v samostatných grafech. Jediný rozdíl mohl být v rychlosti výpočtu, protože Bresenhamův algoritmus by měl být pro výpočet rychlejší, ale tato hodnota nebyla v simulaci měřena. Kde však můžeme vidět nepatrný rozdíl je mezi polynomiálním a tabulkovým vykreslením S křivek. To je dáno tím, že průběh pomocí polynomů je počítán pomocí časových změn a parametrem pro ukončení je tedy celkový čas náběhu a ten se určuje z tabulky STEPS2TIME, respektive z funkce fit, která je extrapolací jen omezeného počtu hodnot. A aby náhodou nedošlo k tomu, že by tento segment byl delší a rychlost tak překročila maximální nastavenou hodnotu, je celkový čas náběhu vynásoben konstantou 0.99, která zajišťuje jednak to, že nastavená rychlost nikdy není překročena, a že počet kroků výsledného náběhu je vždy nepatrně menší než počet kroků, pro který je simulace počítána. Proto je průběh vždy hladký a nemůže dojít k tomu, že by na zlomu akcelerační a decelerační křivky došlo k rázu.

Při počítání průběhu pomocí tabulky se jedná de facto o dosazování předem vypočtených hodnot na příslušná místa do průběhu a to se odvíjí vždy od daného počtu kroků, proto je tento způsob přesnější a více se podobá ideálnímu průběhu.

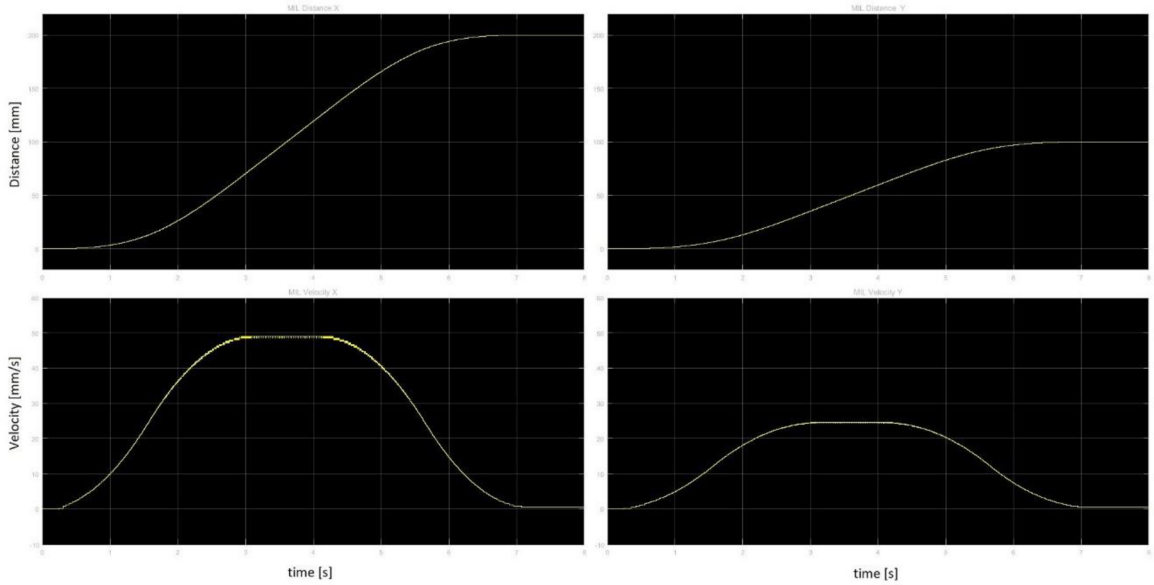
Dalším způsobem bylo i počítání S-křivky spojitě pomocí diferenciálních rovnic, ale to je velmi výpočetně náročné a ze své podstaty je pohyb krokových motorů sám o sobě diskrétní (nespojitý), takže počítat jej spojitě nebo pro tzv. nekonečně malé časové okamžiky, by nemělo smysl.

Výsledky SIL simulace (obr. 40 a 41) dopadly úplně stejně, a když byly vykresleny spolu s MIL simulací do jednoho grafu, tak nebylo poznat, že se jedná o dva samostatné průběhy. Příčinou je to, že obě simulace jsou simulovány na stejném hardwareu, pomocí stejných algoritmů a stejných programů, proto i jejich výsledky bývají velmi často totožné.

Posledním provedeným typem simulace byla Processor-in-the-loop simulace a její výsledky pro výpočet řídícího vektoru pomocí tabulky jsou zobrazeny v obr. 42. Je patrné, že výsledný pohyb při všech třech simulacích pro vypočtené hodnoty pomocí tabulky, je totožný, z čehož vyplývá, že vypočítané vektory ve všech třech případech jsou shodné. Jediné čím se tedy tato simulace liší od ostatních, je čas výpočtu, který v procesoru uvnitř vývojového kitu nebyl změřen, ale v reportech byl čas od spuštění simulace, přes odeslání dat a nahrání programu do procesoru až po vygenerování vypočtených vektorů. Tento čas se v případě počítače pohybuje v řádu desetin sekundy (při druhém spuštění, kdy už je program zkompilován, je to cca 0.75s) a v případě PIL simulace, s použitím procesoru na vývojovém kitu, v řádech desítek sekund (při druhé spuštění se celkový čas pohyboval cca kolem 17s).

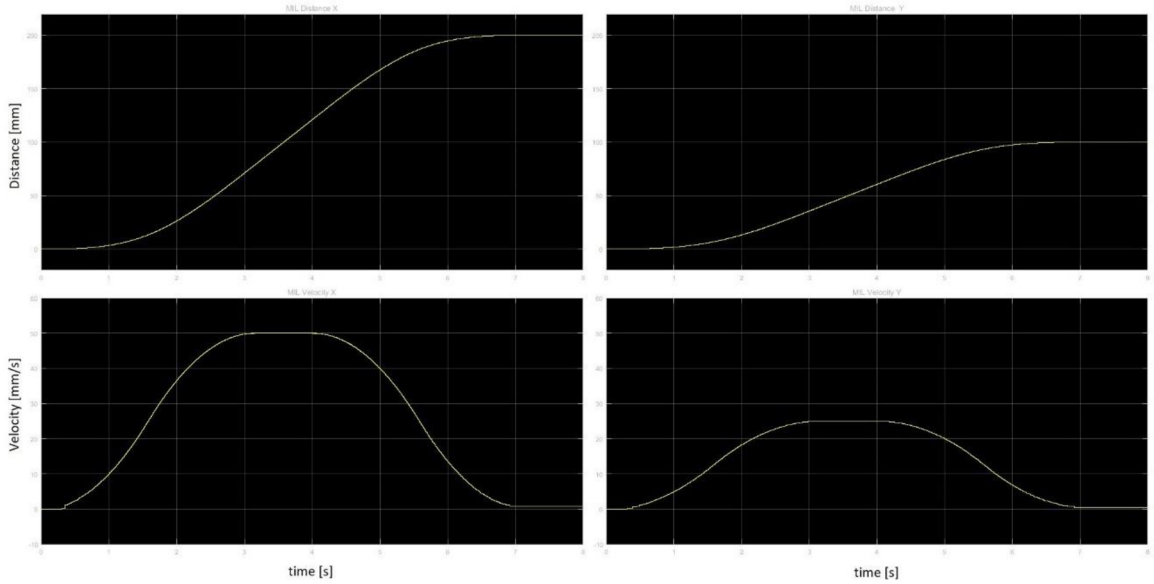
Před optimalizací programu, pro procesor externího hardwareu, se dokonce stalo, že systém musel simulaci přerušit, protože nedostal do stanoveného času odpověď z vývojového kitu.

Výsledky MIL simulace – Bresenham/DDA + S-křivka pomocí polynomů



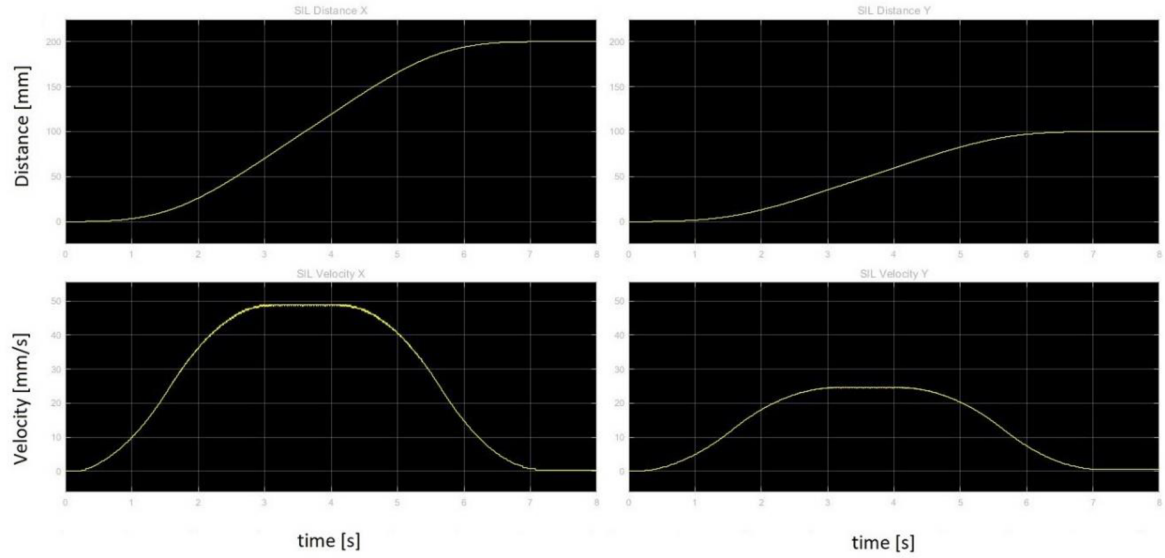
Obr. 37 - MIL simulace - interpolace Bresenham/DDA + S-křivka pomocí polynomů

Výsledky MIL simulace – Bresenham/DDA + S-křivka pomocí tabulky



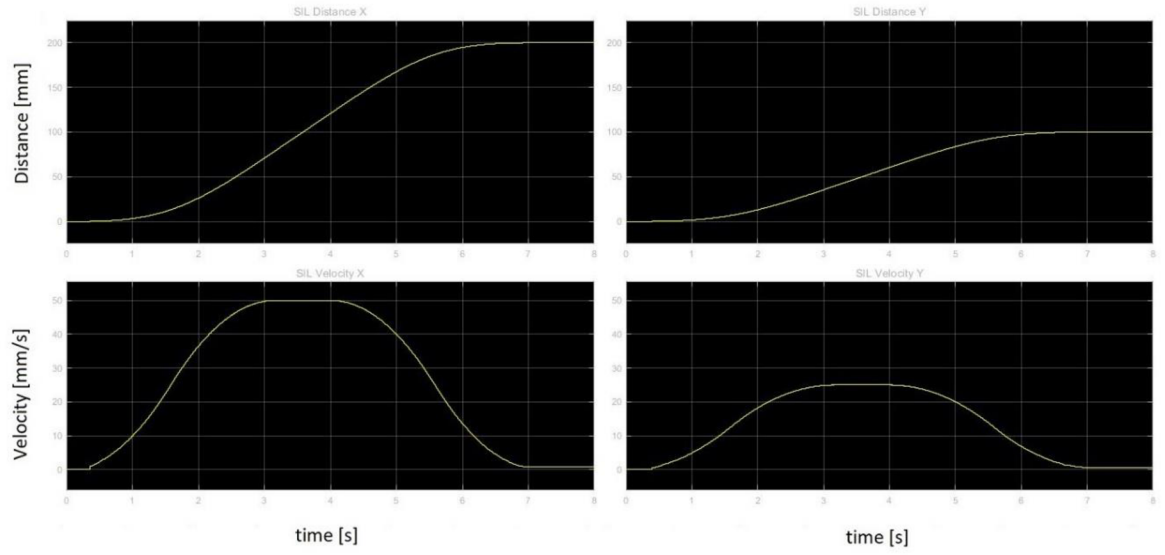
Obr. 38 - MIL simulace - interpolace Bresenham/DDA + S-křivka pomocí tabulky

Výsledky SIL simulace – Bresenham/DDA + S-křivka pomocí polynomů



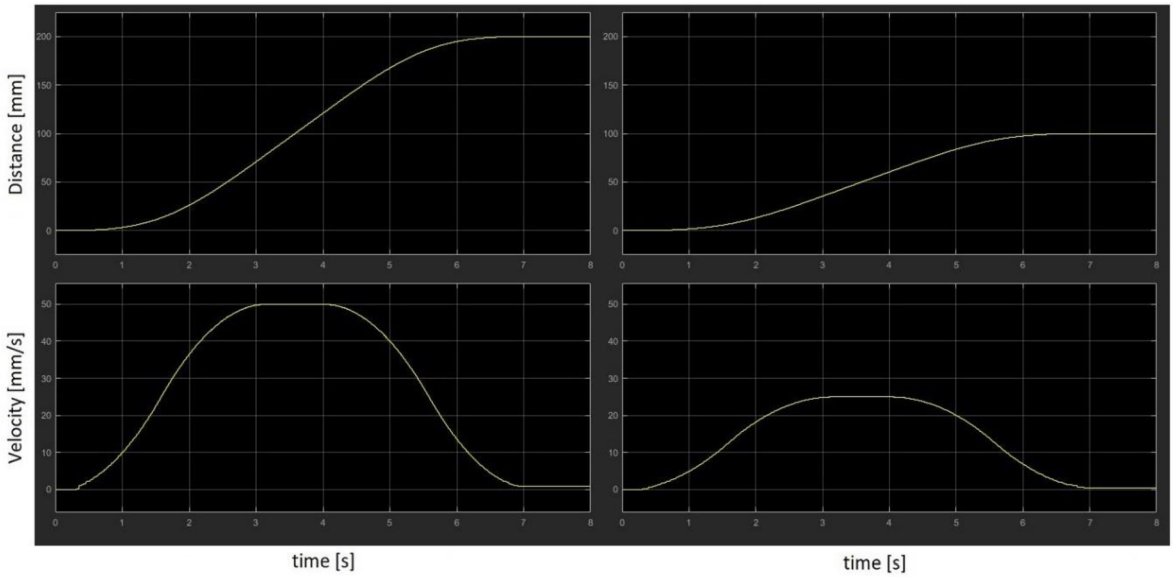
Obr. 39 - SIL simulace - interpolace Bresenham/DDA + S-křivka pomocí polynomů

Výsledky SIL simulace – Bresenham/DDA + S-křivka pomocí tabulky



Obr. 40 - SIL simulace - interpolace Bresenham/DDA + S-křivka pomocí tabulky

Výsledky PIL simulace – Bresenham/DDA + S-křivka pomocí tabulky



Obr. 41 - PIL simulace - interpolace Bresenham/DDA + S-křivka pomocí tabulky

Závěr

Tato diplomová práce se zabývala především hledáním vhodného interpolačního algoritmu a rychlostního profilu pro pohyb motoru pro návrh CNC řídícího algoritmu. Výstupem by však měl být algoritmus pro řízení specifického druhu CNC stroje a sice 3D tiskárny. S ohledem na tento fakt k tomu bylo v celé práci přistupováno.

V diplomové práci byl zpracován úvod do dané problematiky, provedena rešerše interpolačních algortimů pro CNC řízení a algoritmů implementujících S-křivky. Byla také namodelována soustava v prostředí Matlab a Simulink a také implementovány namodelované algoritmy do kitu STM32F4, jež je součástí řídícího systému 3D tiskárny. Dále byly provedeny MIL, SIL a PIL simulace jednotlivých interolačních algoritmů v kombinaci s algoritmy implementujích S-křivky a nakonec porovnány výsledky jednotlivých simulací.

V první kapitole byl popsán úvod do problematiky CNC strojů z hlediska softwaru. Byl zde popsán celý řetězec zpracování dat pro 3D tiskárnu od technického výkresu až po vytvoření G-kódu a jeho odeslání do stroje.

V navazující kapitole pak byla provedena řešerše interpolačních algoritmů. Byly zde popsány čtyři nejčastěji používané algoritmy, kterými jsou DDA, Bresenhemův, B-spline a NURBS. Z nich byl vybrán pro implementaci Bresenhamův algoritmus. Pro interpolaci bodů mezi dvěmi zadanými body používá pouze celá čísla, na rozdíl od ostatních algoritmů a je z nich tedy nejrychlejší a pro srovnání byl implementován i DDA algoritmus.

V třetí kapitole byl matematicky popsán model krokového motoru, který má simulovat pohyby 3D tiskárny v prostředí Matlab a Simulink.

Ve čtvrté kapitole pak byl proveden popis řídícího algorithmu. Zde bylo implementováno čtení nejdůležitějších G-kódu z textového souboru, z nich byla provedena následná identifikace typu trajektorie a počátečních a koncových bodů. Pro interpolaci mezi těmito body byl implementován Bresenhamův lineární a kruhový algoritmus a DDA algoritmus. Následně se pomocí look-ahead algoritmu nastavily počáteční a koncové rychlosti na jednotlivých segmentech a nakonec se mezi těmito rychlostmi, vždy pro daný segment, našel ideální průběh rychlosti pomocí S-křivek.

V páté kapitole pak byl vytvořen zjednodušený model krokového motoru a jeho řízení a v něm byly simulovány algoritmy vytvořené v předchozí kapitole. Nejdříve jako „Model in the loop“, poté jako „Software in the loop“ a nakonec i jako „Processor in the loop“.

V poslední kapitole pak byly shrnuty výsledky provedených simulací a zobrazeny průběhy dráhy a rychlosti krokových motorů pro všechny testované případy v závislosti na čase.

V průběhu této práce jsem se seznámil s funkcí 3D tiskáren a veškerého softwaru potřebného k jejich správnému fungování.

Při studiu materiálů byl kladen důraz především na interpolační algoritmy a na rozběhové a zastavovací rychlostní profily hybridních krokových motorů, používaných pro ovládání pohybu extrudéru při tisku. Všechny tyto načerpané poznatky pak byly využity při vypracování této diplomové práce.

V navazujících pracech by bylo vhodné věnovat větší pozornost algoritmu NURBS, který je sice výpočetně náročnější, ale s rozvojem techniky a zvyšujícím se výpočetním výkonem embedded procesorů, jsou v dnešní době stále více používanější, zejména pro svou všestranost a možnost aproximovat tvary plynulou křivkou, která nemá ostré zlomy a přechody trajektorie pohybu.

Seznam použitých zdrojů

[1] GJELAJ, A. Intelligent CNC Programming of Machine Tools: Artificial Techniques. LAP Lambert Academic Publishing, 2016. 136 s. ISBN 978-3-659-88662-1.

[2] Marlin – open source 3D printer firmware. Home | Marlin Firmware [online]. Copyright © 2018 under the terms of the [cit. 27.11.2018]. Dostupné z: http://marlinfw.org/docs/basics/introduction.html

[3] RepRap/cs - RepRap. Ovládání tiskárnty, G-code [online]. Dostupné z: https://reprap.org/wiki/RepRap/cs

[4] Základní rasterizační algoritmy, přednáška, počítačová grafika, PC, UPCE-FEI-KST [online]. Copyright © [cit. 2.12.2018]. Dostupné z: https://temp.buchtic.net/skola/ipogr/prednasky/IPOGR_2010_P04_Alg-RasterizacniAlgoritmy_4s.pdf

[5] Interpolační algoritmy DDA a Bresenham, University information system MENDELU [online]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=6339

[6] Ing. HLINOVSKÝ, Vít, CSc., Elektrické pohony pro automatizaci a robotiku, Moodle Fakulta elektrotechnická, ČVUT FEL, Praha [online]. Dostupné z: https://moodle.fel.cvut.cz/pluginfile.php/51890/mod_resource/content/1/7%20_%20Krokový%20motor.pdf

[7] Ing. RYDLO, Pavel. KROKOVÉ MOTORY A JEJICH ŘÍZENÍ. Liberec, 2000. TECHNICKÁ UNIVERZITA V LIBERCI. Dostupné také z: https://www.google.cz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=2ahUKEwiH2pfFnbHfAhXSaVAKHc3hBtEQFjAIegQIBxAC&url=http%3A%2F%2Fforum.mcontrollers.com%2Fdownload.php%3Fid%3D1054%26sid%3Dacf2d6b4878bd6900d9981e443a17d07&usg=AOvVaw1sxBAGiCIyYhaiwMwMgWK3.

[8] Kim-Doang Nguyen, Teck Chew Ng a I-Ming Chen. On Algorithms for Planning S-curve Motion Profiles. International Journal of Advanced Robotic Systems [online]. Robotics Research Center, Nanyang Technological University, Singapore, 2008, [cit. 2018-12-11]. Dostupné z:

https://www.researchgate.net/publication/221786319_On_Algorithms_for_Planning_S-curve_Motion_Profiles

[9] DR.P.V.MADHUSUDHAN. Lecture series on Computer Aided Design. Department of Mechanical Engineering, IIT Delhi. [online]. [cit. 2018-12-06]. Dostupné z: https://www.youtube.com/watch?v=OkncKzflw8I

[10] Colin FLANAGAN, The Bresenham Line-Drawing Algorithm [online]. Dostupné z: https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html

[11] Alois ZINGL, The Beauty of Bresenham's Algorithm, Vienna, Austria, Srpen 2016 [online]. Dostupné z: http://members.chello.at/easyfilter/bresenham.html

[12] Xia YUBIN and Wang WEI, S-Curve Acceleration/Deceleration Algorithm Based on Reference, School of Computer Science and Engineering, Beihang University, Beijing 100191, China. Dostupné z: http://maxwellsci.com/print/rjaset/v4-131-134.pdf

[13] Motion Control Overview. ADVANCED Motion Controls © 2018 [online].

[cit. 2018-12-20]. Dostupné z: https://www.a-m-c.com/experience/technologies/motion-control/overview/

[14] Creating Motion Profile Tables, Copyright © 2018, Kollmorgen. Dostupné z: http://curvegen.kollmorgen.com/webhelp/workbench/English/Content/UsersManual/Motion%20Profile%20Table%20Advanced.htm?TocPath=Workbench%20User%20Manual%7CCreating%20Motion%7CMotion%20Profile%20Table%7C_____1

[15] Tyagi, A. MATLAB and SIMULINK for Engineers. OUP India, 2011. 492 s. ISBN 9780198072447

[16] KENJO, Takashi a Akira SUGAWARA. Stepping motors and their microprocessor controls. Oxford: Clarendon Press, 2003. Monographs in electrical and electronic engineering. ISBN 0-19-859326-0.

[17] Simulace MIL – Model in the loop. REX controls [online]. Plzeň: REX Controls, s.r.o. [cit. 2019-04-10]. Dostupné z: https://www.rexcontrols.cz/simulace-mil

[18] Simulace SIL – Software in the loop. REX controls [online]. Plzeň: REX Controls, s.r.o. [cit. 2019-04-15]. Dostupné z: https://www.rexcontrols.cz/simulace-sil

[19] Simulace PIL – Processor in the loop. REX controls [online]. Plzeň: REX Controls, s.r.o. [cit. 2019-04-17]. Dostupné z: https://www.rexcontrols.cz/simulace-pil

[20] Edward RED, Kinematics and dynamics of advanced mechanisms, such as robots, with computer simulation of mechanism motion. [cit. 2019-07-20]. Dostupné z: http://www.et.byu.edu/~ered/ME537/Notes/Ch5.pdf

Seznam použitých zkratek

CNC – Computer numerical control (Počítačové číslicové řízení)

GUI – Grafické uživatelské rozhraní

MIL – Model in the loop

SIL – Software in the loop

PIL – Processor in the loop

Seznam použitých obrázků

Seznam použitých tabulek

Seznam příloh

Příloha 1: CD - Řídící algoritmus a simulace