VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

SROVNÁNÍ NÁSTROJŮ MATLAB A PYTHON S OHLEDEM NA VYUŽITÍ V MECHATRONICE

COMPARISON OF MATLAB AND PYTHON WITH RESPECT TO MECHATRONICS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Jakub Rabčan

AUTHOR

VEDOUCÍ PRÁCE — Ing. Václav Sova

SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Jakub Rabčan**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Václav Sova**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Srovnání nástrojů MATLAB a Python s ohledem na využití v mechatronice

Stručná charakteristika problematiky úkolu:

Interaktivní vývojové prostředí se skriptovacím programovacím jazykem MATLAB je dominantním nástrojem v oblasti inženýrských výpočtů. Cena tohoto nástroje je ovšem značná.
Python je vysokoúrovňový skriptovací programovací jazyk, který může být volně šířen a užíván. Existuje pro něj mnoho různých knihoven, které usnadňují práci. Z pohledu mechatronického inženýra to mohou být především knihovny Numpy, Scipy a Matplotlib, které umožňují provádět vědecké a inženýrské výpočty s maticemi a pokročilé vykreslování dat. Cílem práce bude srovnání těchto dvou nástrojů s ohledem na využití v Mechatronice.

Cíle bakalářské práce:

1) stručné představení MATLABu a Pythonu
2) srovnání obou nástrojů v oblastech:
- řešení soustav lineárních rovnic
- vykreslování dat, tvorba grafů
- řešení soustav nelineárních rovnic
- řešení obyčejných diferenciálních rovnic a jejich soustav
- modelování a práce s LTI systémy ve spojité i diskrétní podobě
- návrh a použití FIR a IIR filtrů
- vzájemná výměna dat mezi Matlabem a Pythonem
- tvorba GUI
3) srovnání bude provedeno formou několika demonstračních úloh

Seznam doporučené literatury:

ZAPLATÍLEK Karel a Bohuslav DOŇAR. MATLAB: tvorba uživatelských aplikací. 1. vyd. Praha: BEN - technická literatura, 2004. ISBN 80-7300-133-0.

DOWNEY Allen B., Think Python. 1st ed. Sebastopol, CA: O'Reilly, 2012. ISBN 9781449330729.

WALTERS Gregory. The Python quick syntax reference. New York, N.Y.: Apress, 2014. ISBN 9781430264798.

MathWorks [online]. [cit. 2016-11-01]. Dostupné z: https://www.mathworks.com/

Python [online]. [cit. 2016-11-01]. Dostupné z: https://www.python.org/

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Táto bakalárska práca sa zaoberá porovnaním nástrojov Matlab a Python. Cieľom tejto bakalárskej práce je porovnanie týchto nástrojov v rôznych oblastiach s ohľadom na využitie v mechatronike. A nasledovne k týmto oblastiam budú vytvorené demonštračné úlohy.

Kľúčové slová

Matlab, Python, filtrácia, LTI systém, diferenciálne rovnice, sústavy rovníc, gui, samo spustiteľne aplikácie.

Abstract

This bachelor thesis deals with the comparison of instruments Matlab and Python. The aim of this bachelor thesis is comparison of these tools in the diferent areas with regard to utilization in mechatronics. And demonstration task will then be created for these areas.

Keywords

Matlab, Python, filtering, LTI system, differential equations, systems of equations, gui, standalone application.

Bibliografická citácia

RABČAN, J. *Srovnání nástrojů MATLAB a Python s ohledem na využití v mechatronice.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 39s. Vedoucí bakalářské práce Ing. Václav Sova.

Čestné prehlásenie

Prehlasujem, že som túto bakalársku prácu na tému „Porovnanie nástrojov Matlab a Python s ohľadom na využitie v mechatronike“ vypracoval samostatne s použitím odbornej literatúry a prameňov uvedených v zozname, ktorý tvorí prílohu tejto práce.

V Brne dňa 26.5.2017 .....

Jakub Rabčan

Poďakovanie

Ďakujem Ing. Václavovi Sovovi, za vedenie mojej bakalárskej práce, za mnohé rady a pripomienky. A následne ďakujem mojim rodičom a celej rodine za to, že tu pri mne stáli a podporovali ma.

Obsah

Úvod

V dnešnej dobe sa už technológie dostali na takú vývojovú úroveň, že nie je možné, aby človek robil zložité výpočty bez toho, aby použil nejaký programovací jazyk. Na svete máme širokú škálu programovacích jazykov. Rôznorodosť je tak veľká, že ich môžeme jednoducho rozdeliť na objektovo orientované, procedurálne, funkcionálne a logické programovanie. Ďalším takým rozdelením môže byť podľa ceny ich licencií. To znamená, že niektoré programovacie jazyky sú príliš drahé, ale nájdu sa aj také, ktoré sú distribuované pod voľnou licenciou. A existuje mnoho iných kritérií pomocou ktorých ich môžeme rozdeliť.

V oblasti inžinierskych výpočtov je najviac preferovaným a dominantným nástrojom Matlab. Jedna sa o skriptovacie a interaktívne vývojové prostredie, ktoré ma neskutočne veľa užitočných nástrojov. Jeho nevýhoda spočíva vo vysokých cenách licencií. Na druhej strane tu mame Python. Je to vysoko úrovňový skriptovací jazyk, ktorý voľne dostupný v rôznych verziách. Existuje veľa knižníc s nástrojmi. Ale najčastejšie sa používajú Numpy, Scipy a matplotlib. Tvoria základ v inžinierskych výpočtoch.

V nasledujúcich kapitolách sa budeme zaoberať jednotlivými oblasťami ako napríklad licencie, to znamená, aké licencie sú používané pre Matlab a Python, aká je ich cena a čo dovoľujú a pod. Alebo akým spôsobom môžeme používať aplikácie vytvorené v jednom z týchto jazykov, bez toho, aby sme mali príslušný program nainštalovaný. Budeme sa aj zaoberať nástrojmi, ktoré sa používajú napríklad na filtrovanie, aké sú ich rozdiely, výhody a tak ďalej, až kým sa nedostaneme k cieľu, ktorý je porovnať tieto dva jazyky.

1. Matlab a Python

1.1. Python

Python je jednoduchý na používanie, ktorého popularita sa v posledných rokoch zvyšuje. Jeho autorom je Guido van Rossum a bol vyvinutý pred viac ako desiatimi rokmi. Meno jazyka nebolo odvodené od názvu hada, ale autor ho nazval podľa britského populárneho seriálu „Monty Python's Flying Circus" , ale napriek tomu sa názov asociuje práve s hadom ako so seriálom.

Jednoduchá syntax pythonu bola odvodená z veľkej časti z ABC. Bol to jednoduchý vyučovací jazyk, ktorý bol vyvinutý v osemdesiatych rokoch. Python bol navrhnutý tak, aby bol dobre čitateľný. To znamená, že ma jednoduché vizuálne rozmiestnenie a často používa anglické kľúčové slová tam, kde iné jazyky používajú interpunkciu.

Taktiež python je multi-paradigmový jazyk, čo značí, že namiesto toho, aby bol programátor nútený používať určitý typ štýlu programovania, dovoľuje používanie viacerých. Tiež podporuje objektovo orientované, štruktúrované aj funkcionálne programovanie. Ďalšou dôležitou vlastnosťou je, že sa dá jednoducho rozširovať a je multiplatformový [1].

Python poskytuje peknú rovnováhu medzi praktickým a koncepčným. Predstavuje to, že v podstate už od výkladu teórie pythonu začiatočník môže robiť pekné veci, bez toho, aby sa zamotal a stratil v problémoch kompilácie a prepojenia.

Predovšetkým bol utvorený na riešene skutočných problémov. A z toho dôvodu pri vývoji boli použité rôzne funkcie z programovacích jazykov ako C++, Java, Modula-3 a Schémy. Okrem toho ma veľmi širokú škálu knižníc s modulmi, ktoré môžu byť použité na tvorbu všetkých druhov úloh, od webového programovania až po grafiku. Tieto knižnice si môžeme jednoducho stiahnuť a nainštalovať[2].

1.2. Matlab

Matlab je veľmi populárne programovacie prostredie, ktoré používajú milióny vedcov a inžinierov po celom svete na rôzne analýzy a dizajn systémov a produktov, ktoré pomáhajú premene celého sveta. Názov Matlabu bol vytvorený skrátením slov MATrix LABoratory. Matlab bol pôvodne napísaný tak, aby poskytoval ľahký prístup k maticovému softvéru vytvorené projektami LINPACK a EISPACK, ktoré boli súčasťou programovacieho jazyka Fortran. Tento program bol zložitý na programovanie, tak pred sedemdesiatimi rokmi bol vytvorený matlab. Vyvinul ho profesor Cleve Moler. Dnes je vedený pod firmou MathWorks.

Matlab rovnako ako python je multi-paradigmové numerické výpočtové prostredie, ktoré ponúka výkonný programovací jazyk pre numerické výpočty a vývoj algoritmov. Má tisíce vstavaných funkcií v rôznych oblastiach, ako napríklad na tvorbu grafov, analýzu

dát atď. Podobne ako python je multiplatformový jazyk, ktorý podporuje rôzne platformy ako sú Windows, Linux atď. [3].

Taktiež obsahuje samostatný doplnkový balík, Simulink, pridáva grafickú simuláciu založenú na viacerých doménach a model založený na modelovaní pre dynamické a vstavané systémy [4].

2. Licencie

Tieto oprávnenia rozhodujú o tom, kto môže daný softvér používať, kopírovať a šíriť. Väčšinovo je to vlastník, komu to licencie dovoľujú. Každý kto používa tieto diela bez povolenia, môže byť vystavený žalobe.

Softvér, ktorý je chránený autorskými právami a licencovaním pod softvérovou licenciou, rozdeľujeme na dve protichodné kategórie. Prvou kategóriou je open-source/slobodná licencia, ktorá umožňuje používateľom väčšiu slobodu. Na druhej strane tu máme closed source/ uzatvorená licencia, v ktorej prevažujú zákazy pred dovoľujúcimi vecami.

2.1. Licencie v pythone

Python a jeho doplnkové knižnice sú distribuované pod rôznymi licenciami. V tejto kapitole si predstavíme tie najznámejšie a najviac používané licencie.

2.1.1. BSD licencia[5]

BSD licencia je permisívna licencia, je veľmi populárna a je jedna z najpoužívanejších licencií pre slobodný softvér. Bežne sa používa pre Berkel Software Distribution, podľa tejto firmy bola pomenovaná a ešte sa používa pre unixový operačný systém.

Táto licencia je veľmi voľná. To znamená, že ma nízke nároky oproti iným bežným licenciám ako je napríklad GPL. Obsahovo je krátka, pri ktorej používateľ je povinný uvádzať iba meno autora, používanie mena autora je tiež právne ošetrené a dotyčný sa musí ešte zrieknuť zodpovednosti za dielo.

Je vhodná pre tých, ktorí nemajú potrebu riešiť licenčné podmienky, úpravu kódu inými užívateľmi alebo nemajú problém s komerčným využitím ich vlastného softvéru. Jednou z možnosti je tiež používanie diela s BSD licenciu pod inou licenciu, alebo ho môžu ponúkať za poplatok.

Môžete sa stretnúť s tromi verziami BSD licencie. Prvá je pôvodná štvorbodová, ktorá je však nekompatibilná s GPL. Potom máme novú, upravenú trojbodovú, ktorá je už kompatibilná s GPL. Ako poslednú dvojbodovú jednoduchú verziu nazývanú aj ako FreeBSD. Pre oveľa viac informácii sa dozvieme na [6] alebo [7].

2.1.2. MIT licencia[8]

Ďalšou takou známou licenciou, ktorá sa tiež dosť často používa je MIT licencia. Jej názov je skratkou od názvu „Massachusetts Institute of Technology“. Vytvorili ju právnici z tohto inštitútu. Ako podklad brali licenciu BSD, ktorú upravili tak, aby bola jednoduchá na čítanie a pochopenie.

MIT licencia sa zaraduje do kategórie slobodných softvérových licencií. Licencia je taká krátka, že môže slúžiť ako licenčná šablóna. Na vyplnenie licenčnej šablóny stačí zmeniť len oznámenie o autorských právach. Tato licencia udeľuje právo na sublicenciu, čo úplne chýba v licencií BSD. To znamená, že ak je sublicenčná, potom každý distribútor má právo udeliť licenciu softvéru vrátane jeho časti priamo tretím stranám.

Užívateľovi umožňuje neobmedzene právo kopírovať, používať a upravovať program a redistribuovať modifikované verzie programu. Pripúšťa tiež vziať kód s programom pod touto licenciou a vydať nový program pod inou licenciou. Stačí ak tvorca programu uvedie meno pôvodného autora a pridá informácie o licencií. A ešte sa musí vzdať práva na podanie žaloby na správcu kódu. Pre oveľa viac informácii sa dozvieme na [6] alebo [7].

2.1.3. PSF licencia[9]

Licencia je odvodená od názvu neziskovej spoločnosti „Python Software Fundantion“. Táto spoločnosť vlastní práva vlastníctva za programovací jazyk Python. Spravuje licenciu python verzie 2.1 a novšie. Vlastní a chráni ochrane známky spojenie s pythonom.

Je to slobodná softvérová licencia, ktorá dovoľuje testovať, reprodukovať, verejne distribuovať, analyzovať alebo inak používať python samostatne alebo jeho novšej verzií.

Podmienky používania sú jednoduché. Licenčná zmluva a sprava o autorských právach musí byť zachovaná v pythone. Pri distribuovaní diela verejnosti vytvoreného v pythone, musia byť do diela zahrnuté všetky zmeny vykonané v pythone. Nezodpovedá za náhodné, špeciálne škody alebo straty spôsobené modifikáciou, distribúciu alebo iným používaním pythonu. Tiež nedovoľuje používanie licencie v rámci ochrannej známky na podporu alebo propagáciu produktov alebo služieb.

2.1.4. GPL licencia[10]

Táto licencia je najpopulárnejšou z pomedzi ostatných slobodných softvérových licencií. Bola pôvodne napísaná pre projekt na vytvorenie kompletného operačného systému, ktorý by bol slobodným softvérom. Napísal ju Richard Stallman. Tradičné softvérové licencie slúžia potrebám firmy. Ich cieľom je zvyčajne maximalizovať zisk z licencovania softvéru. GPL ma úplne iné ciele. Snaží sa maximalizovať množstvo slobodného softvéru, ktorý je k dispozícií vo verejných zariadeniach.

Podmienky GPL sú dostupné každému, ktorý získa dielo s textom tejto licencie. Každý, kto dodržiava podmienky licencie má dovolené modifikovať, kopírovať a ďalej rozširovať toto dielo. Tieto služby môžu byť poskytované zadarmo alebo GPL umožňuje predávať za ľubovoľnú cenu diela pod touto licenciou. Distribútor nesmie pridať ďalšie obmedzenia, ktoré sú v rozpore s podmienkami GPL licencie. A tiež platí, že ak chceme mať tento slobodný softvér, iba pod podmienkou, že všetky odvodené diela, ktoré budú z neho vytvorené a distribuované, musia mať licenciu pre všetkých pod tou istou licenciou. Ak použijeme pre svoj softvér len malú časť kódu distribuovaného pod GPL licenciou, tak celý kód musíme distribuovať pod touto licenciou. Pre oveľa viac informácii sa dozvieme na [6] alebo [7].

2.2. Licencie Matlabu[11]

Licencie matlabu nie sú tak výrazne, a tak rozsiahle ako licencie, ktoré boli použité v pythone. Všetky pravá, ktoré poskytujú sú určované firmou Mathworks. Môžeme ich rozdeliť podľa toho, na čo sa budú používať.

Čiže na študentskú, už z názvu vyplýva, že je určená pre študentov a nie je určená na zisk alebo príjem generujúci činnosť. Ďalej tu máme licenciu na doma, ktorá je určená čisto iba na súkromné účely. Ešte sú tu školská a štandardná licencia.

Tieto licencie už nezaraďujeme do open source, ale už ich musíme zaradiť do kategórie closed source. To znamená, že si musíme za dane práva zaplatiť. V prípade matlabu sa jedná o dosť veľkú sumu. Táto suma sa líši od toho, akú licenciu si chceme zaobstarať. Ako príklad uvedieme študentskú licenciu. Ak si chceme kúpiť licenciu, len na matlab tak cena je 35€ a za každý doplnok navyše si priplácame 20€. Existujú však rôzne akcie a výhodne balíčky, kde za výhodnú cenu môžeme dostať matlab a niekoľko doplnkov navyše. Cena sa tiež odvíja od toho či ich chceme pre školstvo alebo pre vlastné firmy a aj množstva, ktoré potrebujeme.

Všetky tieto licencie sa odvíjajú od jednej pôvodnej, ku ktorej sa pridávajú alebo odoberajú potrebné podmienky a pravá. Sú to tiež multiplatformové licencie.

3. Tvorba samo spustiteľných aplikácii

Táto časť sa bude venovať samo spustiteľným aplikáciám. Sú to také aplikácie, ktoré nepotrebujú iné vývojové prostredie na to, aby sa dali spustiť. Ich výhoda spočíva v ich názve, to znamená, že ak užívateľ vytvoril aplikáciu alebo program a bol by tento program zverejnený, tak by určite chcel, aby ju mohol spustiť každý, kto sa ku tejto aplikácií dostane bez toho si musel zaobstarať potrebný program na ich spustenie.

3.1. Matlab compiler[12]

Matlab compiler umožňuje automaticky konvertovať programy vytvorené v Matlabe do samostatných aplikácií alebo iných softvérových komponentov a zdieľať ich s ostatnými užívateľmi. Aplikácie a komponenty vytvorené pomocou compileru nepotrebujú Matlab nato, aby sa dali spustiť. Na druhej strane na otvorenie týchto programov bez Matlabu potrebujeme mať nainštalovaný takzvaný „runtime-engine“ v matlabe označovaný ako Matlab Runtime[13]. Je to samostatná sadá zdieľaných knižníc, ktoré umožňuje otvorenie kompilovaných aplikácií a komponentov v počítačoch, ktoré nemajú nainštalovaný Matlab. Matlab Runtime je dodávaný spoločne s aplikáciami alebo je voľne stiahnuteľný na stránkach mathworksu. Kompilovaná aplikácia je určená k voľnej distribúcií bez nutnosti vlastnenia licencií pre Matlab. Tiež môžeme zhotoviť doplnky pre Microsoft Excel a integrovať ich do tabuliek programu Excel.

Existuje rozšírenie pre Matlab compiler, je označované po názvom Matlab compiler SDK. Tento doplnok nám umožňuje kompilovať do iných softvérových komponentov ako napríklad sú C/C++ zdieľané knižnice, JAVA triedy, .NET komponenty a balíčkov pre Python[14].

Generované samo spustiteľné aplikácie sa môžu distribuovať na akýkoľvek cieľový počítač, ktorý ma rovnaký operačný systém ako počítač, na ktorom boli skompilované. Napríklad ak by boli tieto aplikácie zhotovené na prístroji s Windows, tak aj prístroj, na ktorý sa to inštaluje musí mať nainštalovaný Windows. Vytvorenú aplikáciu a jej súčasti si môžeme pozrieť v prílohe pod názvom Matlab/Samo spustiteľne aplikácie.

3.2. Pyinstaller[15]

Na vytvorenie samo spustiteľných aplikácií v Pythone existuje niekoľko spôsobov. Jednou z možností je pomocou Pyinstaller. Je to samostatná knižnica nástrojov, ktorá zbalí aplikáciu Pythonu a všetky jej potrebné závislosti do troch súborov. Užívateľ takto vytvorenú aplikáciu môže spustiť bez inštalácie Pythonu alebo akéhokoľvek modulu. Pyinštaller ma vytvorenú podporu iba na Python 2.7 a na 3.3-3.5 verzie.

Taktiež bol testovaný na operačnom systéme Windows, Linux, Mac OS X, ale nie je multiplatformový. To znamená, že môžeme vytvoriť aplikáciu, ktorá podporuje napríklad systém Windows iba ak tato aplikácia bola vytvorená pomocou pyinstalleru v systéme Windows. Bol úspešne použitý s AIX, Solaris a FreeBSD, ale nie je proti nim testovaný. Pyinstaller je distribuovaný pod licenciu GPL, ale s výnimkou, ktorá vám umožňuje používať ho na vytváranie komerčných produktov.

Pyinstallller jednoducho prečíta a analyzuje kód v skripte. Zhromažďuje všetky informácie o tom, aké funkcie boli použité. Potom výtvory kópie súborov, ktoré obsahujú tieto funkcie a následne ich umiestní do jedného priečinku. Väčšinovo sú samo spustiteľné aplikácie vytvorené pomocou jednoduchého príkazu „*pyinstaller mojskript.py“*. Tento príkaz zadáme do príkazového riadku otvoreného v súbore, kde

chceme tuto aplikáciu vytvoriť. Vytvorenú aplikáciu a jej súčasti si môžeme pozrieť v prílohe pod názvom Python/Samo spustiteľne aplikácie.

3.3. Cx_freeze[16]

Ďalším možným spôsobom ako vytvoriť samo spustiteľné aplikácie v Pythone je pomocou Cx_freeze. Je to súbor skriptov a modulov na zmrazovanie skriptov do samo spustiteľných aplikácií, v podstate rovnakým spôsobom ako py2exe a py2app. Na rozdiel od týchto dvoch nástrojov, cx_freeze je multiplatformový a mal by pracovať na každej platforme na akej pracuje samotný Python. Podporuje iba 2.7 a tiež 3.4-3.6 verzie Pythonu. Cx_freeze je distribuovaný pod licenciou PSF.

Nepodporuje vytváranie do jednoduchého súboru napr. exe, kde všetky knižnice sú vložené do jedného spustiteľného súboru. Ale vytvára priečinok obsahujúci spustiteľný súbor programu spolu so zdieľanými knižnicami ako napríklad .dll a .so.

Na vytvorenie aplikácií postupujeme tak, že si najprv zhotovíme skript pod názvom „*setup.py*“, kde si nastavíme meno súboru, verziu, možnosti a executables. Môžeme nastavenie vidieť v ukážkovom skripte:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 18 21:39:38 2017

@author: Jakub Rabčan
"""

import os
import sys
from cx_Freeze import setup, Executable
base = None
if sys.platform == "win32":
    base = "Win32GUI"

os.environ['TCL_LIBRARY'] = r"C:\Users\HP ProBook
455\Anaconda3\DLLs\tcl86t.dll"
os.environ['TK_LIBRARY'] = r"C:\Users\HP ProBook 455\Anaconda3\DLLs\tk86t.dll"

includes = ["tkinter"]
package = ["os"]

setup(
    name = "guiop",
    version = "1.0",
    options = {"build_exe": {"includes": includes, "packages": package}},
    executables = [Executable("guiop.py", base=base)])
```

Potom stačí do príkazového riadku zadať príkaz „*python setup.py build*“ a vytvorí sa súbor s aplikáciou.

3.4. Py2exe a Py2app[17][18]

Okrem už spomínaných nástrojov na vytvorenie samo spustiteľných aplikácií existujú aj Py2exe a Py2app. Sú distribuované pod licenciou MIT. Nie sú multiplatformové, ale každý funguje na inom operačnom systéme. Py2exe je určená na systém Windows a Py2app je určená pre systém Mac OS X. Tieto nástroje už nemajú podporu na verzie pythonu 3.5 a 3.6.

Pri obidvoch je postup rovnaký, na začiatok si vytvoríme setup.py skript v ktorom si všetko nastavíme. No, a potom zadáme jednoduchý príkaz „*python setup.py py2exe(py2app)“*.

4. Filtrovanie dát

Meraním dát alebo rôznych signálov a vhodným spracovaním získame informácie o vlastnostiach objektov alebo veličín. Získané informácie obsahujú aj nežiadúce údaje, ktorých by sme sa potrebovali zbaviť. Úlohou filtrácie je odstránenie nadbytočných dát, potlačiť rušivý vplyv nežiadúcich zložiek a vyzdvihnúť požadované zložky pomocou vhodného algoritmu.

Táto kapitola rozoberá akými metódami môžeme filtrovať rôzne dáta pri používaní nástrojov Matlab a Python. Ďalej sa bude venovať tomu, aké aplikácie môžeme použiť na filtrovanie. A ako posledné sa porovná rýchlosť filtrovania dát pri používaní nástrojov Matlabe a Pythone.

4.1. Filtrovanie v Matlabe

Matlab je všestranný programovací jazyk, ktorý tiež môžeme použiť na filtrovanie dát alebo signálu. Základný Matlab na filtráciu poskytuje základné funkcie ako sú *filter* a *filter2*. *Filter,* tato funkcia je 1D digitálny filter, ktorá filtruje na základe zadaných parametrov prenosovej funkcie. Funkcia *filter2* je 2D digitálny filter, použije filter konečnej impulzovej odozvy na maticu údajov X podľa koeficientov matici H. Je ekvivalentná k funkcií conv2.

Okrem týchto funkcií existuje dodatočná knižnica, ktorá poskytuje rôzne nástroje pre filtráciu. Tato knižnica sa vola Signal Processing System Toolbox[19]. Je kolekcia nástrojov postavených na počítačovom prostredí Matlab. Nástrojová sadá podporuje širokú škálu operácií spracovania signálov, od generovania priebehu tvaru k návrhu a implementácií filtrov, parametrického modelovania a spektrálnej analýzy.

Signál Processing System Toolbox ponuka funkcie, ktoré podporujú cely rad metód na návrh filtrov. Najznámejšie návrhy filtrov sú IIR a FIR. Hlavnou výhodou používania IIR filtrov oproti FIR filtrov, je že zvyčajne spĺňajú daný súbor špecifikácií s oveľa nižšími nárokmi filtrov ako príslušný FIR filter. Hoci IIR majú nelineárnu fázu, spracovávanie dát v rámci Matlabu sa bežne vykonáva „off-line“, to znamená, že celá dátová sekvencia

je k dispozícií pred filtrovaním. To umožňuje nezvyčajný prístup filtrovania nulovej fázy(cez *filtfilt funkciui*), ktorý eliminuje nelineárne fázové skreslenie IIR filtra.

Klasické IIR filtre sú Butterworth, Chebyshey typ I a II, elliptic a Bessel. Tieto funkcie poskytujú návrch a analýzu analógových a digitálnych filtrov, okrem Bessel ten je podporovaný iba pre analógový prípad, v konfigurácií dolnej a hornej priepusty a pásmových priepust.

Okrem IIR filtrov tu máme aj FIR filtre, ktoré sa často používajú na filtrovanie. FIR filtre majú niekoľko výhod:

- Môžu mať presnú lineárnu fázu
- Sú vždy stabilné
- Metódy návrhu sú vo všeobecnosti lineárne
- Môžu byť efektívne realizované v hardvéry
- Priebehy spúšťania filtra majú konečné trvanie

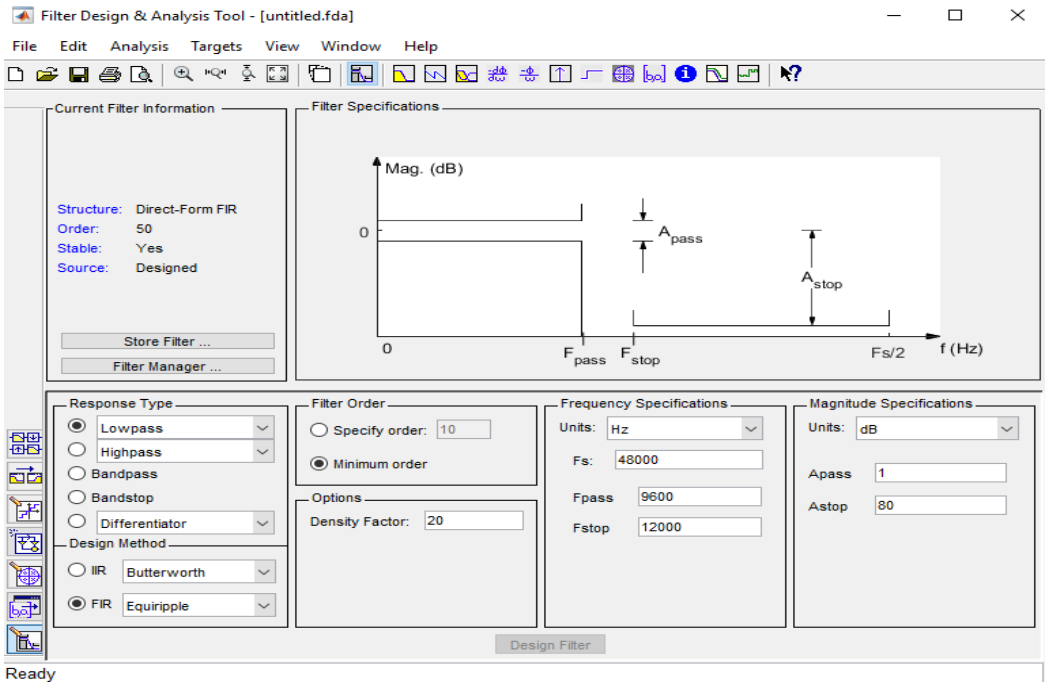
Hlavnou nevýhodou týchto filtrov, je to, že častokrát vyžadujú oveľa vyššie poradie filtrov než IIR filtre na dosiahnutie danej úrovne výkonu. Oneskorenie je často omnoho väčšie ako pri IIR filtrov s rovnakým výkonom. Všetky FIR nástroje navrhujú iba lineárne fázové filtre, okrem funkcie cfirpm. Najznámejšie filtre sú *fir1*, *fir2*, kaiserord tieto funkcie využívajú metódu „windowing“. Potom tu máme aj ostatné, ktoré využívajú aj iné metódy cfirpm, rcosdesingn atď.

Na ukážku si v prílohe pod Matlab/Filtracia/fir_filter, kde je ukázané, ako sa používa návrch filtra pomocou príkazu fir1(návrh FIR filtra založený na oknách – dolná priepust) a filtrovanie pomocou filter. Tiež tam nájdeme(Matlab/Filtracia/butter_filter) aj ukážku pre IIR filter. Na návrch filtra bol použitý Butteworth filter - bandpass, ktorý môžeme vidieť v pomocnej funkcii „*butter_bandpas*“ (Matlab/Filtracia/butter_bandpas), a tiež odfiltrovanie pomocou funkcie filter.

Okrem už spomínaných funkcií Signal Processing toolbox, obsahuje aj funkcie na decimáciu, interpoláciu a prevzorkovanie dát. Taktiež tu nájdeme aj predom vytvorenú aplikáciu na filtrovanie ako FDAtool.

4.1.1. FDATool[19]

Je výkonné používateľské rozhranie na navrhovanie a analýzu filtrov. Umožňuje rýchlo navrhnúť digitálne FIR alebo IIR filtre nastavením špecifických výkonností filtra, importovaním filtrov z pracovného priestoru Matlab alebo priamo špecifikovaním filtračných koeficientov. Tiež poskytuje nástroje na analýzu filtrov, ako sú grafy magnitudy a fázovej odozvy a zero-pole grafy. Taktiež umožňuje vygenerovať kód pre MCU a FPGA. Ale je pre FPGA potrebne mať HDL kóder a pre MCU treba embedded coder. Pre fixet-point simulácie je potrebné mať DPS System Toolbox. Vzhľad tejto aplikácie si môžeme pozrieť na obrázku 4.1.



Obr. 4.1. FDATool

4.2. Filtrovanie v Pythone

Podobne ako Matlab, tak i Python dokáže pracovať s nástrojmi na filtrovanie. Samotný Python neobsahuje prostriedky na filtrovanie, ale na druhej strane existujú knižnice, ktoré nám to zaručujú, ako napríklad Scipy.signal[20]. Je balík nástrojov na filtrovanie, tiež ma funkcie na návrh filtrov a niekoľko interpolačných algoritmov B-splainov pre jednorozmerné a dvojrozmerné údaje.

Python a jeho knižnica scipy.signal ponúka na analýzu a návrh analógových a digitálnych filtrov podobné funkcie, aké obsahuje Matlab, nič výrazného nechýba. Tiež ponúka funkcie pre decimáciu, interpoláciu a prevzorkovanie dát. Základnou funkciu na filtrovanie je *lfilter*. Táto funkcia je 1D filter. Filtruje dáta, získané pri návrhu IIR a FIIR filtrov. Obsahuje aj funkcie pre 2D filtráciu, ktoré však nie sú súčasťou knižnice scipy.signal ale nájdeme ich pod knižnicou scipy.ndimage.

Pri návrhu IIR filtrov využíva rovnaké metódy, aké využíva na návrh filtrov Matlab. To znamená, že majú rovnaké funkcie, ktoré môžeme nájsť pod rovnakým menom a ešte majú rovnaké poradie parametrov, ako majú funkcie v Matlabe. U FIR filtrov tieto funkcie fungujú na rovnaký spôsob, ale už ich nenájdeme pod rovnakým názvom a môžu sa odlišovať aj poradím parametrov.

Tiež na ukážku je v prílohách skript pre FIR filter (Python/Filtracia/fir filter, kde na návrh filtra bol použitá *firwin* funkcia (využíva metódu okien) a pre IIR filter (Python/Filtracia/butterworth filter), tu sa použila funkcia *butter* (Butterworth návrh filtra). A celé sme to filtrovali pomocou príkazu lfilter (1D filter).

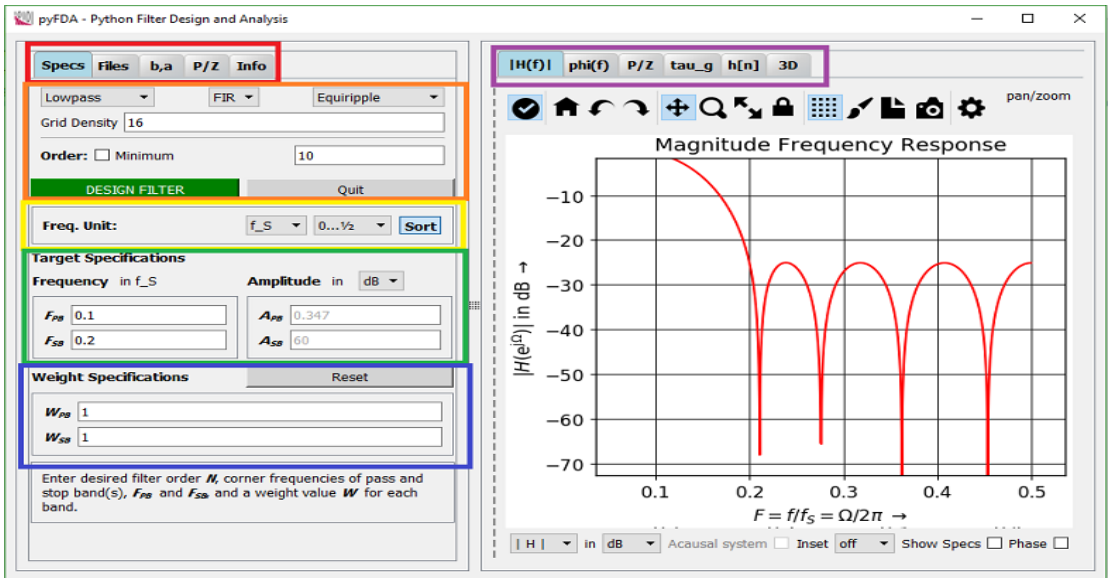
Tak ako Matlab aj Python obsahuje grafické rozhrania, ktoré slúžia na filtrovanie. Ako napríklad môžeme použiť pyFDA.

4.2.1. pyFDA[21]

Je obdoba Matlabovskéj aplikácie FDATool, avšak nie je tak pokročilý. Je to nástroj založený na grafickom rozhraní Qt pre analýzu a návrh diskrétnych filtrov. Obsahuje aj fixpoint návrh filtrov pre FPGA.

Je distribuovaný pod licenciou MIT. Podporuje 2.7 a 3.3 až 3.5 verzie Pythonu. Tiež nájdeme podporu pre všetky operačné systémy. Na prácu používa knižnice ako je numpy, scipy, matplotlib.

Inštalácia je jednoduchá prostredníctvom príkazu „pip install pyfda“. Po otvorení sa nám zobrazí okno, ktoré ma jednoduchú grafickú štruktúru. Vzhľad okna si môžeme všimnúť na obrázku 4.2. V červenom obdĺžniku sú uvedené nástroje pre nastavenie filtrov, na ukladanie, import a export koeficientov a informácie. V oranžovom si nastavujeme návrh filtrovania. Jednotky frekvencie sú v žltom. Nastavovanie cieľových a hmotnostných špecifikácii je v zelenom a modrom. Fialový rámček obsahuje nástroje na vyobrazenie rôznych grafov ako sú amplitúdová a fázová odozva, pole/zero graf a ďalšie.



Obr. 4.2 pyFDA

4.3. Porovnanie rýchlosti filtrovania

Porovnávala sa rýchlosť filtrácií medzi Matlabom a Pythonom, ktorý z týchto programov pracuje s filtrami rýchlejšie. Porovnávanie sa uskutočnilo tak, že sme si zadali veľký objem dát a niekoľko opakovaní pre FIR filter a IIR filtrov. Ako FIR filter sme použili návrh metódou okien a pre IIR filter sme použili Butteworth návrh. Oba filtre boli nastavené na dolnú priepust 29. rádu. Vytvorili sme si zašumenú funkciu, do ktorej sme

vložili 6200000 vzoriek a 30- krát sme opakovali. Po uskutočnení filtrácie sa zistilo, že rýchlosť akou filtruje Python je pri FIR filtri 15 sekúnd a pri IIR filtri je 11 sekúnd. Keď sa porovnával Matlab sa zistila rýchlosť pre FIR filter 10 sekúnd a pre IIR filter 21 sekúnd. Zistilo sa, že priemerne filtruje Python rýchlejšie ako filtruje Matlab. Ukážku nájdeme v prílohe Matlab/Filtracia/FIR_IIR_rychlost (Python/Filtracia/FIR IIR rychlost).

5. LTI systém

LTI, sú trieda systémov používaných v signáloch a systémoch, ktoré sú lineárne aj časovo nemenné. Lineárne systémy sú také, v ktorých platí princíp superpozície. To znamená, že platí aditivita, vstupom pre súčet dvoch signálov bude rovnaký ako súčet výstupu pre tieto signály jednotlivo. A homogenitu, výstup pre násobok iného vstupu bude rovný rovnakému násobku pre tento vstup. Časovo nemenné sú systémy, v ktorých výstup pre konkrétny vstup sa nemení v závislosti od toho, kedy bol tento vstup aplikovaný. Tieto vlastnosti robia LTI systémy ľahko reprezentatívne a graficky pochopiteľné.

LTI systémy delíme na spojité a diskrétne LTI systémy. Diskrétne systémy sa zaoberajú diskrétnymi signálmi, ktoré sú dané funkciami definovanými, len v diskrétnych časových okamihoch a tvorí tak postupnosť funkčných hodnôt. Systém diskrétny v čase je LTI ak vzťah medzi vstupom a výstupom môžeme popísať jednoduchou lineárnou diferenčnou rovnicou s konštantnými koeficientami.

Spojité systémy sa zaoberajú spojitými signálmi, ktoré sú dané spojitými funkciami alebo čiastočne spojitými funkciami spojitého času. Systém spojitý v čase je LTI ak vzťah medzi jeho vstupom a výstupom môžeme popísať lineárnou diferenciálnou funkciu s konštantnými koeficientami [22].

5.1. Matlab a LTI

Jednou z možností ako riešiť LTI systémy je pomocou programu Matlab, ktorý tiež obsahuje rôzne funkcie pre riešenie LTI systémov. Na riešenie Matlab využíva rozširujúcu knižnicu Control System Toolbox[23].

Táto knižnica poskytuje algoritmy a aplikácie na systematickú analýzu, navrhovanie a ladenie lineárnych riadiacich systémov. Systém sa môže špecifikovať ako prenosová funkcia, state-space, zero-pole-gain alebo model frekvenčnej odozvy. Nájdeme tu grafy ako step a bode. Tieto grafy umožňujú analyzovať a vizualizovať správanie systému v časových a frekvenčných doménach.

Control System Toolbox umožňuje vytvárať a analyzovať SISO a MIMO modely. SISO model je taký model, ktorý má jeden vstup a jeden výstup. MIMO model je taký model, ktorý ma viacnásobný vstup s viacerými výstupmi. Vie tiež pracovať s spojitými a diskrétnymi funkciami. Na vytvorenie diskrétnych systémov používame rovnaké funkcie ako pri spojitých ale stým rozdielom, že musíme špecifikovať vzorkovací časový

interval pre každý model, ktorý vytvoríme. Taktiež vlastni funkcie na prevod s spojitého systému na diskrétny systém a naopak, napríklad *c2d, d2c, d2d*.

Okrem samotných funkcií môžeme použiť aplikáciu Linear System Analizer zjednodušuje analýzu lineárnych, časovo invariantných systémov. Pomocou aplikácie môžeme zobraziť a porovnať diagramy odpovedi systémov SISO a MIMO alebo viacerých modelov súčasne. Dokáže generovať grafy časovej a frekvenčnej odozvy, aby sme mohli skontrolovať parametre kľúčových odoziev, ako je čas nábehu, maximálne prekročenie a hranice stability.

Pred tým ako sa pokúsime spustiť aplikáciu musíme mať už vopred vytvorený model vo workspace. Potom môžeme otvoriť pomocou jednoduchého príkazu „*linearSystemAnalyzer"*. Aplikácia je pôvodne nastavená, tak že ako prví graf vyobrazí step ale pomocou nástrojov na paneli úloh si môžeme jednoducho meniť, zobrazovať a analyzovať všetky grafy, ktoré sa používajú pri LTI systémov.

Ukážkový príklad na LTI systém. Nájdeme medzi prílohami Matlab/LTI system/LTIsystem. V tomto skripte riešime LTI systém pomocou stavového modelu, ktorý si následne zobrazíme v grafe step, bode a pomocou aplikácie. Potom ten systém diskretizujeme, a tiež vykreslíme step graf.

5.2. Python a LTI

Rovnako ako Matlab i Python môže byť využívaný na riešenie LTI systémov. Na riešenie týchto systémov využíva knižnicu scipy.signal[20]. O tejto knižnici sme sa bavili v kapitole Filtrácia. Okrem nástrojov na filtrovanie obsahuje funkcie, pomocou ktorých môžeme riešiť tieto systémy. Ani v tejto časti Python nezaostáva za Matlabom, to znamená, že tiež poskytuje riešenie LTI systému pomocou stavového modelu, prechodovej funkcie, ale aj pomocou zero-pole-gain. Tiež tu nájdeme nástroje na simuláciu výstupu LTI systému (*lsim*- simuluje výstup LTI systému).

Takisto zahrňuje funkcie na analýzu správania v časových a frekvenčných doménach ako sú *step, bode, impulse* atď. Avšak, oproti Matlabu, ktorý má tieto funkcie lepšie prepracované, pretože umožňujú aj vykresľovanie do grafov pričom už spomínane funkcie v Pythone vykresľovať do grafu. Získame iba hodnoty, ktoré môžeme vyniesť do grafov pomocou nástrojov na vykreslenie.

Python umožňuje tiež riešenie spojitých a diskrétnych systémov. Zvláštnosťou je, že pri diskrétnych systémoch sú špeciálne vytvorené funkcie, ktoré sú ekvivalentné ku funkciám pre spojitý systém. Toto v Matlabe nenájdeme, ako sme už spomínali Matlab má iba jeden typ nástrojov, kde pri diskrétnych pridáme iba vzorkovací čas. Tiež obsahuje aj príkazy na transformáciu zo spojitých na diskrétne a opačne alebo transformácie medzi reprezentujúcimi stavmi ako napríklad prenosovú funkciu môžeme zmeniť na stavový model (*tf2ss*).

Hlavné funkcie, ktoré sa používajú na riešenie LTI systémov sú *lti* (pre spojite) a *dti* (pre diskrétne). Predstavujú triedy, ktoré v sebe majú zabudované funkcie na riešenie

prenosovej funkcie, stavový model alebo zero-pole-gain. Stačí zadať parametre, ktoré by sme zadávali do jednotlivých funkcií. Samo to rozozná o ktorú funkciu sa jedna, a potom podľa nej spočíta riešenie.

Okrem scipy.signal máme možnosť použiť knižnicu python-control[24]. Táto knižnica obsahuje funkcie pre návrh a analýzu systémov spätnej väzby. Tiež obsahuje balík (control.matlab), ktorý poskytuje veľa funkcií, ktoré sú dostupné v Matlab Control Toolbox. Využíva knižnice numpy a scipy. A je distribuovaná pod BSD licenciou.[33]

I pre Python je vytvorený ukážkový príklad v Python/LTI system/lti, kde tiež riešime stavový model ako pre spojitý, tak i pre diskrétny systém. A analyzovaním modelu pomocou step, bode získame hodnoty, ktoré následne vykreslíme v grafe.

6. Riešenie diferenciálnych rovníc

Na technické a vedecké účely sa vytvárajú diferenciálne rovnice ako matematické idealizácie reality, aby sa zjednodušili zložité javy, ktoré sa vyskytujú vo fyzickom svete. Napríklad experimenty so skutočným objektom sú nahradené počítačovou simuláciou pohybu objektu podliehajúceho rôznym silám. V praxi sú problémy s diferenciálnou rovnicou riešené aproximačnými metódami.

6.1. Riešenie v Matlabe

Keď sa povie diferenciálna rovnica, každého hneď napadne ako čo najlepšie vyriešiť takúto rovnicu. Ak si vyberieme Matlab, zistíme, že nám ponúka širokú škálu riešiteľov na riešenie diferenciálnych rovníc. Ale najprv než so všetkým začneme si musíme uviesť akým spôsobom môžeme zapísať diferenciálne rovnice v Matlabe. Medzi spôsoby zadávania patrí aj pomocou *inline* funkcie, túto funkciu použijeme, ak sa jedna o jednoduchú diferenciálnu rovnicu. Pri sústave diferenciálnych rovníc sa používa zápis v maticovom tvare. Kde na jednu stranu si vyjadríme diferenciálny člen (napr. y‘) a na druhú stranu dáme všetko ostatné. Diferenciálne rovnice vyššieho rádu ako jedna musia byť pomocou substitúcie previesť na sústavu diferenciálnych rovníc prvého radu. Zápis týchto rovníc všimnúť na ukážke kódu, kde rovnica je zapísaná v samostatnom skripte:

```
% mame jednoduchu dif. rovnicu my'' + by' + ky = F
%rovnicu druheho radu premenime
%pomocou substitucie na sustavu dif. rovnic prveho radu
function dy = fn(t,y,m,b,k,F)
dy(1,1) = y(2,1);
dy(2,1) = -(b/m)*y(2,1) - (k/m)*y(1,1) + F/m;
```

Keď sme si definovali ako si môžeme zapísať diferenciálnu rovnicu, tak teraz môžeme pristúpiť na jej vyriešenie. Už sme spomenuli že Matlab má veľa riešiteľov diferenciálnych rovníc. Takým najobyčajnejším riešiteľom je pravé dsolve, je to základná funkcia, ktorá dokáže vyriešiť jednoduché rovnice, vyššieho rádu a aj sústavy rovníc.

Pri riešení obyčajných diferenciálnych rovníc v Matlabe, rozdeľujeme riešiteľov podľa toho či sa jedna o stiff systém[24], non stiff systém alebo plne implicitné diferenciálne rovnice. Na riešenie stiff systému Matlab využíva riešiteľov ako sú *ode15s, ode23s, ode23t, ode23tb*[25]. A naopak na riešenie pre non-stiff systému (opak stiff) používame *ode45, ode23, ode113* [25]. Tieto funkcie sa od seba odlišujú tým, že aký spôsob využijú pri riešení. Preto ako prvá voľba riešenia ODE by mala byt metódou ode45 ak zlyhá alebo je nefiktívny a je tu podozrenie že sa jedna o stiff systém tak našou voľbou by mala byť funkcia ode15s. Riešitelia používajú metódu zero-crosing detenction (v preklade kríženie nulou).

Okrem úž spomínaného Matlab vie riešiť aj diferenciálne rovnice vyjadrene implicitne pomocou ode15i. Tiež môžeme nájsť funkcie, ktoré dokážu riešiť diferenciálne algebrické rovnice. Rovnako vie riešiť parciálne diferenciálne rovnice a nesmieme zabúdať, že vyriešiť komplexne diferenciálne rovnice. Viac sa môžeme dočítať na[26]

Ukážkové príklady, ktorých sa zaoberaním riešenie diferenciálnych rovníc v Matlabe a porovnávanie riešenia pri stiff systému nájdeme v prílohe Matlab/Diferencialne rovnice.

6.2. Riešenie v Pythone

Python tiež obsahuje riešiteľov na riešenie diferenciálnych rovníc ale ich počet je menši oproti Matlabu. Môžeme jednoduchú diferenciálnu rovnicu zapísať pomocou funkcie *lambda,* ako v Matlbe pomocou *inline* funkcie. Pri uvažovaní riešenia sústav diferenciálnych rovníc funguje na rovnakom princípe ako Matlab, čiže pomocou matíc. To iste sa dá povedať aj pri diferenciálnych rovniciach vyššieho rádu. Tiež ich musíme upraviť pomocou substitúcie na diferenciálne rovnice prvého radu. Zápis v Pythone si môžeme pozrieť na kúsku kódu, kde sme použili rovnakú rovnicu ako pri Matlabe:

```
# mame jednochu dif. rovnicu  my** + by* + ky = F
# rovnicu druheho radu premenime
# pomocou substitucie na sústavu dif. rovnic prveho radu
import numpy as np

def fn(t,y):
    dy = np.zeros(2)
    dy[0] = y[1]
    dy[1] = -(b/m)*y[1] - (k/m)*y[0] + F/m
    return dy
```

Python na riešenie diferenciálnych rovníc využíva knižnicu scipy.integrate[28]. Je to funkcia, ktorá obsahuje nástroje na riešenie integrálov a diferenciálnych rovníc. Prvou voľbou pri riešení obyčajných diferenciálnych rovníc je pomocou funkcie odeint. Je to základný riešiteľ, ktorý na riešenie využíva *lsoda*[29][30]. Je to riešiteľ umiestnení v kolekcií ODEPACK. Je to kolekcia Fortrian riešiteľov pre diferenciálne rovnice. Jednou z výhod pri riešení ODE je to, že nemusíme rozlišovať či sa jedna o stiff alebo o non stiff systém. Automaticky zistí, o ktorý typ systému sa jedná.

Okrem príkazu *odeint* existuje trieda *ode*, ktorá v sebe zahŕňa riešiteľov obyčajných diferenciálnych rovní. Kde si pomocou príkazu set_integrator nastavíme parametre integrátoru, ako sú riešiteľ, typ systému... Táto trieda už nemá takú výhodu, akú má odeint. To znamená, že už si musíme dať pozor aký typ systému sa jedná. Ak riešime non-stiff systém, tak si nastavíme metodu BDF alebo ak riešime stiff systém, tak si nastavíme metódu Adams. Trieda ode nám ponúka možnosť riešenia obyčajných rovníc pomocou „vode“. Môžeme riešiť komplexne ODE, ak sú v analytickom tvare používame *zvode* [29][30] ak nie sú, tak použijeme „dvode“[30]. Ďalej nám ponuka možnosť riešenia pomocou metódy Runge-kutta („dopri5 a dopri853“).

Okrem už spomínaného riešenia komplexných čísel existuje špeciálne vytvorená trieda pre komplexné ODE *complex_ode*. Funguje rovnako ako *ode*, ale stým rozdielom, že pre mapuje systém rovní na reálnu hodnotu pred použitím integrátorov.

Namiesto scipy.integrate môžeme použiť knižnicu sympy[31], ktorá obsahuje nástroje na riešenie obyčajných diferenciálnych rovníc. Tiež umožňuje riešiť aj parciálne a algebrické diferenciálne rovnice. Môžeme tu nájsť funkciu *dsolve*, ktorá sa je podobná tej, ktorá je v Matlabe.

Prílohe Python/Diferencialne rovnice nájdeme ukážkový príklad na riešenie sústavy diferenciálnej rovníc, ktorú tvoria 3 tyče spojene 2 pružinami a jedna tyč je tlmená. V príklade zisťujeme výchylku a rýchlosť danej tyče. Tiež v prílohe v tom istom súbore nájdeme aj príklad na porovnanie riešenia stiff systému pomocou rôznych riešiteľov. Kde v detaile pozorujeme veľkosť použitého kroku.

7. Tvorba GUI

Grafické používateľské rozhranie alebo GUI odkazuje na všeobecnú myšlienku ikon, tlačidiel atď., ktoré sú vizuálne prezentované používateľovi ako „front-end“ softvérovej aplikácie. GUI uľahčuje a zjednodušuje užívateľovi používanie aplikácie, pričom si zachová svoju komplexnosť. GUI ma v dnešnej dobe veľké použitie pre elektronické zariadenia, ktoré používajú rôzne aplikácie a operačné systémy.

7.1. GUI v Matlabe

V Matlabe môžeme GUI vytvoriť dvomi spôsobmi. Prvým možným spôsobom je pomocou vlastného programovania, to znamená, že všetky potrebné objekty, ako sú tlačidlá posuvníky atď., použite v našom programe si vytvoríme pomocou funkcii, ktoré sú súčasťou Matlabu. Ako je napríklad *uicontrol*[32]. Pri vytváraní GUI aplikácií programovacou cestou aa umožňuje väčšia kontrola nad samotným kódom, to znamená, presne vieme ako čo funguje a funkcia obsahuje iba to, čo sme do nej zadali bez toho, aby sme sa v nej stratili. Na druhej strane nám prináša aj nevýhody. Na svoje naprogramuje vyžaduje dlhší čas. Okrem toho prináša do programu zložitosť a je potrebné poznať, aké funkcie sa podieľajú na vytvorenie GUI.

Druhým takým možným spôsobom ako vytvoriť GUI v Matlabe je pomocou GUIDE [33]. Je grafické vývojové interaktívne vývojove prostredie, obsahujúce všetky grafické objekty typu uicontrol (tlačidla, posuvníky...), ktorými môžeme ovládať beh aplikácie. Automaticky generuje základný zdrojový kód pre ovládanie. Vzhľad vytvoreného GUI ukladá GUIDE do súboru s príponou .fig a jeho zdrojový kód do súboru s príponou .m.

Oproti klasickému programovaniu nám GUIDE poskytuje veľmi rýchlu tvorbu aplikácii a celý ten proces uľahčuje. Pravdou je, že pri veľmi zložitom programe sa môžeme veľmi ľahko a rýchlo stratiť.

Okrem už spomínaných spôsobov existuje aj pomocou App designer. Funguje na princípe ako GUIDE. Tiež máme okno, v ktorom si poukladáme potrebné objekty. Obsahuje rovnaké objekty ako ma GUIDE, ale je obohatený o viac objektov, ktoré už GUIDE nemá. Vo viacerých veciach a smeroch je oproti GUIDE prepracovanejší i generovaný kód je ľahšie čitateľný. V podstate by sa dalo povedať, že sa jedná o náhradu za GUIDE a postupom času by mohol byť úplne nahradený nástrojom App designer[34].

Ako príklad na tvorbu GUI sme riešili teleso na pružine s tlmením, kde si zadáme vstupné parametre, ako sú hmotnosť telesa, tuhosť pružiny, doba riešenia a vychýlenie telesa. Tento príklad bol riešený pomocou GUIDE a nájdeme ho v prílohe pod názvom Matlab/GUI/pruzinka2. V súbore GUI sú tri skripty, fun1-vnom je zadaná diferenciálna rovnica telesa, pruzinka2.fig-predstavuje GUI okno s grafickými objektami a pruzinka2-skript, kde sú zadefinované funkcie jednotlivých objektov.

7.2. GUI v Pythone

I Python môžeme použiť na vypracovanie GUI. Python oproti Matlabu obsahujú širokú škálu frameworkov, ktoré sú určené na tvorbu GUI. Môžeme ich rozdeliť na jedno-platformové a na multi-platformové. Medzi jedno-platformové zaraďujeme venster atď., ktoré sú používané iba na jednej platforme. Frameworky ako sú Tkinter, PyQt a mnoho ďalších, spadajú do kategórie multi-platformové [35].

Potom tu nájdeme GUI design tools. Obsahujú nástroje na GUI grafickým štýlom, podobne ako v Matlabe si otvoríme okno, do ktorého si potom naukladáme rôzne objekty a automaticky vygeneruje kód. Medzi tieto frameworky môžeme zaradiť Qt Designer[36], PAGE[37]... Väčšinovo tieto nástroje potrebujú na svoj chod knižnice z kategórie multi-platformové, ale nájdu sa aj také, ktoré fungujú samostatne.

Tiež nájdeme aj neaktívne a neudržiavané, tieto frameworky sa prestali modernizovať a vylepšovať. A boli už nahradené inými nástrojmi. Patria tu napríklad PyUi alebo PyMT.

Najznámejším nástrojom na tvorbu GUI je Tkinter [38], ktorý je priamo zabudovaný do Pythonu, čiže netreba ho už sťahovať a inštalovať. Je distribuovaný firmou Scriptics. Keďže je súčasťou Pythonu, tak je rovnako multiplatformový. Rovnako ako Matlab i tento modul obsahuje veľa funkcií, ktorými definujem rôzne objekty ako sú tlačidlá, grafy. GUI tvory pomocou programovacieho spôsobu, čo je trochu komplikovanejší spôsob a náročnejší čas. Ale vynahrádza to jednoduchou stavbou funkcii, čo znamená že zadávanie parametrov a vlastností objektov je v celku jednoduché. Na používanie nám

stačí na importovať modul. Na kúsku kódu vidíme ako sa zadáva tlačidlo, posuvník a vkladanie textu. Pomocou príkazu grid alebo pack rozmiestníme objekty v okne.

```
self.sld1 = tk.Scale(self,from_=0.1,to = 10, resolution=0.1, orient =
tk.HORIZONTAL,variable= self.m)

self.lab2 = tk.Label(self,text="Tuhost k:")
self.lab2.grid(row = 0,column = 2

self.b =tk.Button(self, text="Start", command =self.po,bg = "gray")
self.b.pack(side="top",fill ="x")
```

Ďalší známy nastroj, ktorý môžeme použiť na tvorbu GUI je PyQt [39] . Je kompatibilný so všetkými populárnymi operačnými systémami. Je dostupný pod GPL licenciu. Má rovnaké funkcie ako ma Tkinter, s tým rozdielom, že sa trochu odlišujú v názve a aj zadávanie vlastností, parametrov a tvorba objektov je iné ako u Tkintru. Na porovnanie si môžeme pozrieť zadávanie objektov na nasledujúcom kóde:

```
self.le = QtWidgets.QLineEdit()
self.le.setText("100")
self.b1 = QtWidgets.QPushButton('Start')
self.sl = QtWidgets.QSlider(Qt.Horizontal)
self.sl.setMinimum(-5.)
self.sl.setMaximum(5.)
```

a vkladanie týchto objektov:

```
v_box = QtWidgets.QVBoxLayout()
v_box.addWidget(self.le)
v_box.addWidget(self.sl)
```

I v Pythone na príklad GUI sme riešili teleso zavesené na pružine s tlmením. GUI sme vytvorili pomocou Tkintru (Python/GUI/pruzinka). Okrem Tkintru sme na tvorbu použili PyQt, kde sme už riešili jednoduché vykresľovanie (Python/GUI/guivpyqt).

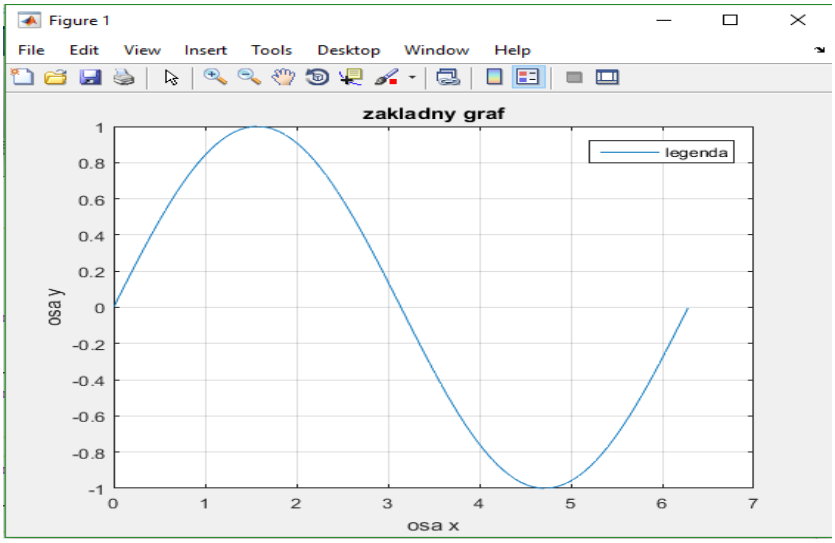
8. Grafy

Vykresľovanie do grafu nám pomáha pri vizualizácií údajov získaných meraním alebo výpočtom. Tiež nám pomáha identifikovať a zdôrazniť rôzne vlastnosti, ktoré boli získane v priebehu merania a výpočtu.

Pri používaní Matlabu zistíme, že ma neskutočne veľa príkazov na tvorbu rôznych druhov grafov. Taktiež disponuje pokročilou grafikou v oblasti 2D a 3D grafov. Umožňuje nám vytvoriť rôzne druhy grafov ako sú napríklad stĺpcové, polárnych súradniciach, logaritmické a mnoho ďalších. Tiež nájdeme aj nástroje určené na animáciu. Matlab nám

umožňuje upravovať grafy, pomocou rôznych nástrojov, ako sú úprava čiar, nastavovanie si os a mnoho ďalších.

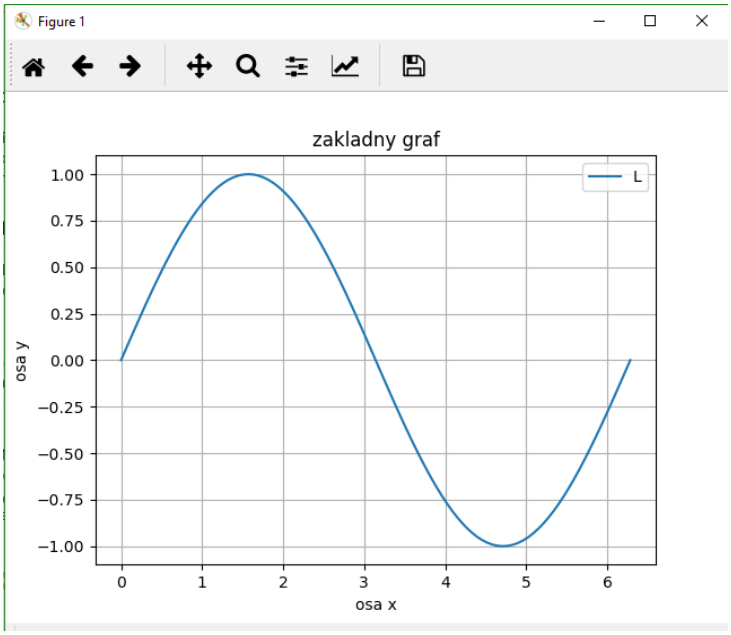
Základný príkaz na tvorbu grafu je *plot* patrí medzi najpoužívanejšie príkazy na tvorbu grafu. Umožňuje nám tvoriť 2D grafy pre 3D existuje podobná funkcia plot3. Po zadaní príkazu sa nám otvorí dialógové okno, v ktorom bude vyobrazený graf (obrázok 8.1). V tomto okne môžeme nájsť samotný graf, ale aj veľa možných nástrojov na úpravu, ktoré sa nachádzajú nad vykresleným grafom. Samotný príkaz plot nám daný plot bude prekresľovať, ale ak by sme ich chceli zachovať použijeme funkciu hold on [33].



Obr. 8.1 Graf v matlabe

Samotný Python nemá v sebe zabudované funkcie na tvorbu grafov. Nato využíva určené knižnice. Najviac využívanou a tvorí základ pri tvorbe grafov je knižnica Matplotlib[40]. Je to balík funkcii určených pre tvorbu 2D a 3D grafov. Podporuje interaktívne a neinteraktívne vykresľovanie a môže uložiť obrázky vo viacerých výstupných formátoch (PNG, PS a iné). Ponúka nám širokú škálu typov grafu ako sú histogramy, koláčový graf a mnoho ďalších. Tiež nám ponúka veľa funkcii na úpravu grafov. Okrem iného je veľmi prispôsobivý, flexibilný a ľahko sa používa. Matplotlib bol navrhnutý, tak aby bol rovnako použiteľný ako Matlab s možnosťou použitia v Pythone. Preto môžeme nájsť niektoré funkcie pod rovnakým menom ako Matlabe. Na druhej strane obsahuje aj iné funkcie na vykresľovanie grafov, ktoré Matlab nemá ako je *axhline*, ktorý pridá vodorovnú čiaru cez os.

Základný príkaz je *plot*, funguje na rovnakom princípe ako plot v Matlabe. Rozdiel je v tom, že tento príkaz automatický neotvára okno s grafom. Na otvorenie musíme použiť príkaz show. Okno s grafom môžeme vidieť na obrázku 8.2. Graf vypadá rovnako ako v Matlabe, len počet nástrojov v okne je oveľa menší, tvorí ich len základ, a tiež ich nájdeme nad grafom.



Obr. 8.2 Graf v Pythone

Okrem knižnice Matplotlib môžeme použiť aj knižnice PyQtgraph[41], Vispy a Chaco. PyQtgraph poskytuje základne funkcie na 2D kreslenie (rozptylové grafy, histogramy atď), tiež nájdeme aj funkcie pre 3D grafy. Okrem iného nám poskytuje rýchlu aktualizáciu údajov pre videá a animácie. Obsahuje funkcie kúskovanie viacrozmerných obrázkov v ľubovoľných uhloch. Ma oveľa viac funkcií pre vedecké a strojárske aplikácie ako sú ImageView, ScatterPlotWidget , nastroj na analýzu ... Funguje na každej platforme, ktorá podporuje numpy a PyQt.

Keď porovnáme PyQtgraph s knižnicou Matplotlib zistíme, že nie je pre vykresľovanie tak prepracovaný ako Matplotlib, ale na druhej strane funguje oveľa rýchlejšie. Matplotlib je skôr zameraný na tvorbu grafiky v publikácii, zatiaľ čo pyqtgraph je určený na použitie v aplikáciách na získavanie a analýzu dát. Matplotlib je intuitívnejší pre užívateľov, ktorý už pracovali s Matlabom. Naopak PyQtgraph je intuitívnejší pre programátorov Python/Qt Okrem týchto knižníc na vykresľovanie v Pythone sa môžeme stretnúť aj s knižnicou Vispy. Vispy je nová 2D/3D knižnica na vykresľovanie, založená na OpenGL. Ponuka priame vizualizácie dát v reálnom čase a rýchlu interaktívnu vizualizáciu 3D modelov.

Jednoduché ukážkové príklady nájdeme v prílohách Matlab/Graf/plt a pre Python/Graf/plot. Príklady spôsob vykresľovania do grafu, jednoduchú úpravu a porovnanie grafov medzi Matlabom a Python- pomocou Matplotlib a PyQtgraph.

9. Import a export dát

Niekedy pri práci s dátami v Matlabe alebo v Pythone je potrebne si tieto dáta uložiť do iných dátových súborov, aby mohli byť používané na inej aplikácií alebo formáte. Na druhej strane niekedy je potrebné zas získať dáta z iných dátových typov, aby ich bolo možné používať v Matlabe alebo Pythone.

Matlab poskytuje bohatú sadu príkazov, ktoré podporujú import a export dát z mnohých rôznych formátov. Ako napríklad dátový formát, kde zaraďujeme MAT, CSV, TXT..., tiež vie pracovať s formátom typu tabuľkových súbormi (XLS, WK1), s formátmi vedeckých údajov (CDF, FITS a HDF). A tiež nesmieme zabudnúť, že dokáže pracovať s obrázkovými formátmi (TIFF, PNG, HDF, BMP), s zvukovým formátom (AU,SND, WAV) a nakoniec s video formátom (AVI).

Na prácu s High-Level Data v Matlabe môžeme použiť funkcie load na import dát a save na export. Ak chceme ukladať dáta pomocou funkcie save, musíme si uvedomiť do akého formátu sa budú dáta ukladať. Pretože predvolením nastaveným je vkladanie do matlabovského súboru MAT. Ak potrebujeme iný typ, tak si jednoducho prenastavíme. Pri funkcii load nie je potrebné nastavovanie dátového typu, vie automaticky rozoznať o aký dátový typ sa jedná. A keďže vie pracovať s Low-level Data, to znamená, že podporuje binárny prístup ukladanie dát a na prácu využíva funkcie ako sú fwrite, fread atď.

Okrem iného podporuje prenos dát z externých zdrojov. Tento prenos dát môžeme uskutočniť pomocou rozhrania RS-232 alebo pomocou podpory protokolu FTP. Tiež môžeme uskutočniť prenos pomocou systému DDE v prostredí operačného systému Windows[42].

Na import a export daťa môžeme tiež použiť Python, ktorý využíva knižnice ako je numpy, csv [43], pickle[44] a iné. Pri práci s dátami zistíme, že dokáže pracovať s rovnakým dátovým typom ako Matlab (txt, csv..). Dokonca vie, tiež pracovať Matlabovským dátovým súborom (MAT). Pre prácu so zvukovým formátom a video formátom (wav, avi ...) používa knižnicu Pymedia[45]. Na prácu s excelovským súborom môžeme využiť knižnicu pyexcel[46].

Pythone môžeme tiež použiť funkcie ako load a save, ale tieto funkcie pracujú iba s npy formátom (binárny súbor, ktorý používa Numpy). Ak by sme chceli pracovať s iným musíme si nájsť príslušní príkaz, napríklad pri txt súboru použijeme loadtxt a savetxt (Numpy). Tiež vie pracovať s low-level dátami pomocou funkcií ako sú napríklad read a write. Umožňuje prenos pomocou RS-232, FTP a DDE.

Porovnávali sme rýchlosť zápisu v Matlabe a Pythone pomocou príkazu fprintf a write. Formátovane ACSII data sme zapisovali do textového súboru a ich počet bol 500 tisíc a predstavovali ich jednotlivé čísla. Po uskutočnení operácie sme zistili, že rýchlosť zápisu v Matlabe pomocou fpintf je 18 sekúnd a rýchlosť zápisu v Pythone je 3 sekundy. Ukázalo sa, že Python je oveľa rýchlejší v ukladaní formátovaných ASCII dát do textového súboru. Potrebný skript, ktorý bol použitý nájdeme v Matlab/Import a export

dat/imp_rýchlosť a pre Python/ Import a export dat/imp_rychlost, a tiež v tom súbore nájdeme ukážkový príklad na import a export dát pod názvom imp_exp.

10. Sústavy rovníc

10.1. Lineárne rovníc[47][48]

Sústava lineárnych rovníc je jedným z najdôležitejších problémov výpočtov. Sústavy rovníc sú reprezentované a riešené ako matice: Ax=B. Rovnice si upravíme, tak aby všetky neznáme parametre boli na ľavej strane a ostatne na pravé, potom tieto rovnice poskladáme do matíc.

Sústavy lineárnych sa riešia v Matlabe najjednoduchším spôsobom a to pomocou operátorov / a \ (delenie z pravá a z ľavá). K týmto operátorom existujú aj príslušne funkcie mldivide (\) a mrdivide(/). V Pythone na riešenie sústavy rovníc, tiež môžeme použit operátor / ale delí po prvkoch. Avšak môžeme ich vyriešiť pomocou príkazu solve z knižnice scipy alebo numpy.

Druhým takým spôsobom, ktorý je pre Matlab a Pyhon rovnaký, je pomocou inverznej matice k štvorcovej matici A. Pre obdĺžnikovú maticu sa použije pseudo inverzia. Tiež oba programy môžu riešiť sústavy pomocou rozkladu matíc ako je napríklad LU rozklad, Cholesky atď. Dokážu riešiť sústavy lineárnych rovníc ako je matica A pásová, trojuholníková alebo Hermitianová. Tiež môžeme riešiť pomocou metódy najmenších štvorcov. Okrem obyčajných lineárnych rovníc vedia vypočítať zložitejšie rovnice ako sú Sylvester, Ricati a Lyapunov rovnice. Existujú aj komplexnejšie metódy riešenia lineárnych rovníc pre veľké, symetrické, štvorcové a riedke matice ako je napríklad metóda nekonjugovaných gradientov.

Ukážkový príklad je uložený v prílohe pod názvom Matlab/Sustavy/Linearne rovnice/dynamika a Python/Sustavy/Linearne rovnice/dynamika, kde sa rieši sústava telies, ktoré sú úplne uvoľnené a rovnice zapísane do matice, ktorá sa následne vyrieši.

10.2. Nelineárne rovnice

Keď sme sa už bavili o sústavách lineárnych rovníc, tak by sme si mohli spomenúť, akým spôsobom môžeme riešiť sústavy nelineárnych rovníc. Než začneme s riešením si musíme rovnice upraviť tak ,aby sa všetky rovnali nule. Potom si vytvoríme funkciu, ktorá bude obsahovať tieto rovnice. Potom pomocou základného príkazu fsolve, ktorý je rovnaký pre Matlab a Python, ale odlišujú sa zadávaním parametrov. Python fsolve nájdeme ho v knižnici scipy.optimize. Tiež nájdeme aj funkcie na riešenie veľkých sústav, a tiež funkcie na výpočet koreňov sústav nelineárnych rovníc.

Ukážkový príklad je uložený v prílohe pod názvom Matlab/Sustavy/Nelinearne rovnice/nelirovnice a Python/Sustavy/Nelinearne rovnice/nelirovnice, kde sa rieši jednoduchá sústava nelineárnych rovníc.

11. Záver

Cieľom tejto bakalárskej bolo porovnanie dvoch silných programovacích jazykov Matlab a Pythonu v oblastiach, ktoré sú dôležite a používajú vo výpočtoch v Mechatronike.

Venovali sme sa porovnaniu licencií, ktoré sa používajú pre Python a Matlab. Pri skúmaní sa zistilo, že Python a jeho doplnkové knižnice sú distribuované pod viacerými licenciami, ktoré zaraďujeme do slobodných softvérových licencií. Oproti Pythonu ma Matlab, ktorý iba par, ktoré sú odvodené od jednej pôvodnej. Zaraďujeme ich medzi platené licencie.

Tiež sme rozoberali akým spôsobom môžeme vytvoriť samo spustiteľné aplikácie bez použitia Matlabu a Pythonu. Na kompiláciu takýchto aplikácií Matlab využíva Matlab Compiler. Naopak u Pythonu na tvorbu takýchto aplikácií je viacero spôsobov ako napríklad pomocou Pyinstaller, Cx_freeze, Py2exe a Py2app.

Pri filtrácií sme sa zaoberali funkciami, ktoré môžeme použiť pri filtrácií dát. Matlab a Python obsahujú nástroje pre návrh a analýzu spojitých a digitálnych filtrov ničím výrazným sa nelíšia. Dokonca pri IIR filtrov majú spoločné filtre s rovnakým názvom. Zaoberali sme sa tým, aké aplikácie môžeme použiť na filtrovanie. Ďalej sme skúmali rýchlosť filtrovania pri FIR a IIR filtrov a výsledok bol, že Python je s porovnaním s Matlabom rýchlejší.

Tiež sme porovnávali riešenia LTI systémov. Matlab využíva na riešenie používa control system toolbox a na riešenie spojitých a diskrétnych LTI systémov v Pythone môžeme použiť scipy.signal, ale tiež môžeme python.control. Ničím výrazným sa neodlišujú.

Pre diferenciálne sústavy rovníc v Matlabe nájdeme riešiteľov pre stiff a non-stiff systém, pre diferenciálne rovnice zadane implicitne atď. Python oproti Matlabu má menej riešiteľov, hlavne pri riešení stiff a non-stiff sa používajú tie isté funkcie, len sa nastaví typ systému. Medzi hlavný rozdiel je, že Matlabovsky riešitelia využívajú zero-crosing detection pričom v Pythonu to nenájdeme.

Na tvorbou GUI v Matlabe používame sadu funkcií alebo pomocou grafického rozhrania GUIDE a App designer. Na rozdiel od Matlabu ma Python neskutočne veľa knižníc, ktoré môžeme použiť na tvorbu GUI. Medzi základne knižnice zaraďujeme Tkinter a PyQt.

Keď sa pozreli na tvorbu grafov nič výrazného sa nenašlo, takže čo vieme vykresliť v Matlabe tak tiež to dokážeme vykresliť v Pythone, pretože obsahuje knižnicu matplotlib, ktorá navrhnutá tak aby sme mohli použiť Matlabovské funkcie na tvorbu grafov v pythone.

Rovnako platí aj pre import a export dát, obidva programy vedia pracovať s high level a low level dátami. Porovnávali sme rýchlosť zápisu dát do textového súboru a zistili sme, že Python v tomto smere je oveľa rýchlejší ako Matlab. Keď sa pozrieme na riešenie lineárnych a nelineárnych sústav rovníc zistíme, že tam rozdiely nie sú. Oba programy dokážu riešiť rovnakými metódami.

Prínosom celej tejto práce pre mňa je, že som získal nový prehlaď informácií ohľadom používania Matlabu a Pythonu a utvrdil som si to, čo som doposiaľ o tomto programe vedel.

12. Zoznam použitej literatúry

[1] WIKIPEDIA *Python (programovací jazyk)* [online]. [cit 2017-05-05] Dostupne z: https://sk.wikipedia.org/wiki/Python_(programovací_jazyk)

[2] DOWNEY, A.; ELKNER, J.; MEYERS, Ch. *How to Think Like a Computer Scientist: learnig with Python* 1st ed. Wellesley: Grean Tea Press, 2002. 288p. ISBN 0-9716775-0-6

[3] HUMUSOFT *Matlab* [online]. [cit 2017-05-05]. Dostupne z: http://www.humusoft.cz/matlab/details/

[4] HUMUSOFT *Simulink* [online]. [cit 2017-05-05], Dostupne z: https://www.mathworks.com/products/simulink.html

[5] WIKIPEDIA *BSD Licenses* [online], [cit. 2017-05-07]. Dostupne z: https://en.wikipedia.org/wiki/BSD_licenses

[6] ROSEN, Lawrence E. *Open source licensing: software freedom and intellectual property law* 1st ed. New Jersey: Prenctice Hall, 2005. 432p. ISBN 0-13-148787-6

[7] LAURENT, St., M. Andrew, Understanding Open Sourece and Free Software Licensing, 1st ed. Sebastopol: O'Reilly, 2004, 207p. ISBN 978-0596-00581-8

[8] Python Software Fundation *PSF* [online], [cit. 2017-05-07]. Dostupne z: https://www.python.org/psf/mission/

[9] WIKIPEDIA *MIT License* [online], [cit. 2017-05-07]. Dostupne z: https://en.wikipedia.org/wiki/MIT_License

[10] WIKIPEDIA *GNU GPL License* [online], [cit. 2017-05-07]. Dostupne z: https://sk.wikipedia.org/wiki/GNU_General_Public_License

[11] MATHWOKS *Pricing and Licensing* [online], [cit. 2017-05-07]. Dostupne z: https://www.mathworks.com/pricing-licensing.html

[12] MATHWORKS *Matlab Compiler* [online], [cit. 2017-05-09]. Dostupne z: http://www.mathworks.com/help/releases/R2014a/pdf_doc/compiler/compiler.pdf

[13] MATHWORKS *Matlab Runtime*[online], [cit. 2017-05-09]. Dostupne z: https://www.mathworks.com/products/compiler/mcr.html

[14] MATHWORKS *Matlab Compiler SDk* [online], [cit. 2017-05-09]. Dostupne z: https://www.mathworks.com/products/matlab-compiler-sdk.html

[15] PyInstaller Development Team *PyInstaller Manual* [online], [cit. 2017-05-09]. Dostupne z: https://pythonhosted.org/PyInstaller/index.html#

[16] TUININGA, A. *Cx_frezee* [online], [cit. 2017-05-09]. Dostupne z: http://cx-freeze.readthedocs.io/en/latest/index.html#

[17] *Py2exe* [online], [cit. 2017-05-09]. Dostupne z: http://www.py2exe.org

[18] *Py2app*[online], [cit. 2017-05-09]. Dostupne z: https://py2app.readthedocs.io/en/latest/#

[19] MATHWORKS *Signal Procesing Toolbox* [online], [cit. 2017-05-10]. Dostupne z: https://www.mathworks.com/help/pdf_doc/signal/signal_tb.pdf

[20] SCIPY *Signal processing* [online], [cit. 2017-05-10]. Dostupne z: https://docs.scipy.org/doc/scipy-0.18.1/reference/tutorial/signal.html

[21] GITHUB *pyFDA* [online], [cit. 2017-05-10]. Dostupne z: https://github.com/chipmuenk/pyFDA

[22] WIKIPEDIA *LTI systém* [online], [cit. 2017-05-12]. Dostupne z: https://en.wikipedia.org/wiki/Linear_time-invariant_theory

[23] MATHWORKS *Control system toolbox* [online], [cit. 2017-05-12]. Dostupne z: https://www.mathworks.com/help/pdf_doc/control/get_start.pdf

[24] MURRAY, M. Richard *Python control systems library* [online], [cit. 2017-05-12]. Dostupne z: http://python-control.readthedocs.io/en/latest/

[25] *Stiff system* [online], [cit. 2017-05-12]. Dostupne z: http://www.biomatematica.it/urbino2002/programmi/Italy1.pdf

[26] SHAMPINE, L. F.; Reichelt, W. M. *The Matlab ode suite* [online], [cit. 2017-05-12]. Dostupne z: https://www.mathworks.com/help/pdf_doc/otherdocs/ode_suite.pdf

[27] HARMAN, L. T.; Dabney, J.; Richert N. *Advanced Engineerigs Mathemathics with Matlab* 2nd ed. California: Brooks/Cole 1999. 750p. ISBN 0-534-37164-7

[28] SCIPY *Integration and ODEs* [online], [cit. 2017-05-12]. Dostupne z: https://docs.scipy.org/doc/scipy/reference/integrate.html

[29] HINDMARSH, C. Allan *ODEPAK* [online], [cit. 2017-05-12]. Dostupne z: https://computation.llnl.gov/casc/odepack/

[30] SCIPY *ode* [online], [cit. 2017-05-12]. Dostupne z: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html#scipy.integrate.ode

[31] SYMPY *Solvers* [online], [cit. 2017-05-12]. Dostupne z: http://docs.sympy.org/latest/modules/solvers/solvers.html#ordinary-differential-equations-odes

[32] ZAPLATÍLEK, K.; DOŇAR, B. *Matlab tvorba uživatelských aplikací* 1. vyd. Praha: BEN-technická literatúra 2004. 216 s. ISBN 80-7300-133-0

[33] MARHAND, P. ; HOLAND, O. , T. *Graphics and GUIs with Matlab* 3rd ed. New York: CHAPMAN & HALL/CRC 2000. 518p. ISBN 1-58488-320-0

[34] MATHWORKS *App designer* [online], [cit. 2017-05-13]. Dostupne z: https://www.mathworks.com/products/matlab/app-designer.html

[35] PYTHON *GUI programing in Python* [online], [cit. 2017-05-13]. Dostupne z: https://wiki.python.org/moin/GuiProgramming

[36] QT *Qt designer* [online], [cit. 2017-05-13]. Dostupne z: http://doc.qt.io/qt-4.8/designer-manual.html

[37] *PAGE* [online], [cit. 2017-05-13]. Dostupne z: http://page.sourceforge.net/html/index.html

[38] FREDRICH *An introduction to Tkinter* [online], [cit. 2017-05-13]. Dostupne z: https://users.tricity.wsu.edu/~bobl/cpts481/an-introduction-to-tkinter.pdf

[39] *PyQt* [online], [cit. 2017-05-13]. Dostupne z: https://www.tutorialspoint.com/pyqt/pyqt_tutorial.pdf

[40] TOSI, S. *Matplotlib for Python developers* 1st ed. Birmingham: Packt publishing 2009. 308p. ISBN 978-1-847197-90-0

[41] CAMPAGNOLA, L. *PyQtgraph documentation* [online], [cit. 2017-05-13]. Dostupne z: https://media.readthedocs.org/pdf/pyqtgraph/develop/pyqtgraph.pdf

[42] ZAPLATÍLEK, K.; DOŇAR, B. *Matlab začíname se signály* 1. vyd. Praha: BEN-technická literatúra 2006. 272 s. ISBN 80-7300-200-0

[43] Python Software Fundation *CSV* [online], [cit. 2017-05-15]. Dostupne z: https://docs.python.org/3/library/csv.html

[44] Python Software Fundation *Pickle* [online], [cit. 2017-05-15]. Dostupne z: https://docs.python.org/2/library/pickle.html

[45] *Pymedia* [online], [cit. 2017-05-15]. Dostupne z: http://pymedia.org

[46] *Pyexcel* [online], [cit. 2017-05-15]. Dostupne z: https://github.com/pyexcel/pyexcel

[47] BLAHO, Michal, Matlab lineárna algebra *Posterus* [online] 2009. č. 9 [cit. 2017-05-16] Dostupný na WWW: http://www.posterus.sk/?p=2717 ISSN 1338-0087

[48] SCIPY *Linear algebra* [online], [cit. 2017-05-12]. Dostupne z: https://docs.scipy.org/doc/scipy/reference/linalg.html

Zoznam skratiek

AIX	Advanced Interactive eXecutive
AU	Audio – zvukový formát
BMP	Bitmap
BSD	Berkeley Software Distribution
CDF	Cumulative distribution function
CSV	Comma-separated values
DDE	Dynamic Data Exchange
FDATool	Filter Design and Analysis Tool
FIR	Finite impulse response
FITS	Flexible Image Transport System
FPGA	Field programmable gate array
FTP	File transfer protocol
GPL	General Public License
GUI	Graphical User Interface
HDF	Hierarchical Data Format
IIR	Infinite impulse responce
LTI	Linear Time Invariant
MAT	Matlab file
MIMO	Multiple-input and multiple-output
MIT	Massachusetts Institue of Technology
ODE	Ordinary Differential Equations
PAGE	Python Automatic GUI generator
PSF	Python Software Foundantion
PYFDA	Python Filter Design and Analysis Tool
SISO	Single-input and single-output
SND	zvukový formát pre Macintosh
TIFF	Tagged Image File Format
WAV	Waveform audio file forma

Prílohy

Všetky prílohy sú dostupne na DVD:

- **Matlab:**
  - Samo spustiteľne súbory
    - Pruzinka- obsahuje súbory s aplikáciou a potrebnými funkciami vytvorené pri kompilovaní
  - Diferencialne rovnice
    - difrovnica- riešenie dif. rovnice
    - fun – funkcia, v nej sú zadane dif. rovnice
    - stiff_system – porovnanie riešenia stiff systému
  - Filtrácia dát
    - butter_bandpass – pomocna funkcia pre navrch filtra
    - butter_bandpass_filter – pomocna funkcia pre filtrovanie
    - butter_filter – príklad na IIR filter
    - fir_filter- príklad na FIR filter
    - FIR_IIR.rychlost- porovnanie rýchlosti filtrovania
  - Graf
    - plt- príklad na vykresľovanie
  - GUI
    - fun1 – obsahuje zadefinovanú dif. rovnicu telesa na pružine
    - pruzinka2.fig – okno obsahujúce grafické objekty
    - pruzinka2 – obsahuje funkcie pre GUI
  - Import a export dát
    - imp_exp – príklad na ukladanie a čítanie dát
    - imp_rychlost – porovnanie rýchlosti zápisu dát
  - LTI systém
    - LTIsystem – príklad na riešenie LTI systému
  - Sústavy rovníc
    - Lineárne rovnice
      - dynamika – riešenie lineárnych rovníc
    - Nelineárne rovnice
      - nelirovnice – riešenie nelineárnych rovníc
      - nerov- zadefinovaná sústava nelineárnych rovníc
- **Python**
  - Samo spustiteľne súbory
    obsahuje súbory s aplikáciou a potrebnými funkciami vytvorené pri kompilovaní
  - Diferencialne rovnice
    - difrovnice- riešenie dif. rovnice
    - stiff_system – porovnanie riešenia stiff systému
  - Filtrácia dát
    - butter_filter – príklad na IIR filter
    - fir_filter- príklad na FIR filter

        - FIR_IIR.rychlost- porovnanie rýchlosti filtrovania
    - Graf
        - plot- príklad na vykresľovanie pomocou matplotlib a pyqtgraph
    - GUI
        - pruzinka – obsahuje GUI vytvorené Tkintrom
        - guivpyqt – obsahuje GUI vytvorené PyQt
    - Import a export dát
        - imp_exp – príklad na ukladanie a čítanie dát
        - imp_rychlost – porovnanie rýchlosti zápisu dát
    - LTI systém
        - lti – príklad na riešenie LTI systému
    - Sústavy rovníc
        - Lineárne rovnice
            - dynamika – riešenie lineárnych rovníc
        - Nelineárne rovnice
            - nelirovnice – riešenie nelineárnych rovníc