VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

POROVNÁNÍ VOLNĚ DOSTUPNÝCH SIMULAČNÍCH NÁSTROJŮ

COMPARISON OF FREELY AVAILABLE SIMULATION TOOLS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ROBIN VYSLOUŽIL**
AUTHOR

VEDOUCÍ PRÁCE **Dr. Ing. PETR PERINGER,**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Vysloužil Robin**
Program: Informační technologie
Název: **Porovnání volně dostupných simulačních nástrojů**
Comparison of Freely Available Simulation Tools
Kategorie: Modelování a simulace

Zadání:

1. Seznamte se s volně dostupnými simulačními nástroji a metodikou jejich porovnávání. Vyberte alespoň 8 takových nástrojů. Zaměřte se na reprezentativní vzorek těchto nástrojů s ohledem na jejich použití ve výuce modelování a simulace. Výběr musí zahrnovat tyto nástroje: *OpenModelica*, *SciLab*, *Dymola*.
2. Analyzujte a zdokumentujte základní vlastnosti vybraných simulačních nástrojů. Navrhněte vhodnou metodiku pro porovnávání charakteristických vlastností simulačních nástrojů. Navrhněte reprezentativní sadu modelů vhodných pro porovnání těchto nástrojů.
3. Nainstalujte vybrané simulační nástroje a implementujte v nich sadu navržených modelů. Proveďte experimenty a vyhodnoťte jejich výsledky. Porovnejte simulační nástroje podle zadaných kritérií.
4. Zhodnoťte dosažené výsledky a navrhněte možná další vylepšení použitých metod.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních dvou bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Peringer Petr, Dr. Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Tato bakalářská práce se zabývá srovnáním volně dostupných simulačních nástrojů. Cílem práce je zdokumentovat a porovnat nástroje implementované nad jazyky Modelica, Matlab a Python. Pro účely srovnání jsou vytvořeny různé simulační modely pro zvolené jazyky. Tyto modely jsou použity pro simulaci a poznatky z běhů simulací jsou následně vyhodnocovány pomocí porovnávací metodiky a jednotlivé nástroje jsou srovnávány.

Abstract

This thesis compares different freely available simulation tools. The aim of the thesis is to document and compare tools implemented in Modelica, MATLAB and Python. Various simulation models are created for each language. The models were simulated and the results from simulation runs were evaluated.

Klíčová slova

Modelica, Matlab, Python, Scipy, Octave, OpenModelica, simulace, srovnání volně dostupných simulačních nástrojů

Keywords

Modelica, Matlab, Python, Scipy, Octave, OpenModelica, simulation, comparison of freely available simulation tools

Citace

VYSLOUŽIL, Robin. *Porovnání volně dostupných simulačních nástrojů.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Dr. Ing. Petr Peringer,

Porovnání volně dostupných simulačních nástrojů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Dr. Ing. Petra Peringera. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Robin Vysloužil
16. května 2019

Poděkování

Rád bych poděkoval panu Dr. Ing. Petru Peringerovi za dobře mířené rady, které vedly k vypracování této práce.

Obsah

Kapitola 1

Úvod

Dnešní doba poskytuje mnoho možností, jak si pro svůj vyvíjený produkt, nebo například jen pouhou hypotézu toho, jak by nějaký systém mohl fungovat, vytvořit model a provést na tomto modelu simulaci [18]. Z takto provedené simulace získáme představu o tom, jak se testovaný produkt chová. To nám společně s vhodně zvoleným simulačním nástrojem může ušetřit nemalé finanční prostředky při uvádění produktu na trh, neboť simulace mohou pomoci odhalit nedostatky, které je potřeba odladit [5], proto je vhodné věnovat čas simulaci dostatečný čas. Simulační nástroje jsou implementovány nad různými jazyky, ale pro účely práce byly zvoleny primárně dvě hlavní rodiny jazyků - MATLAB [17] a Modelica [8]. Zároveň je pro porovnání vybrán jazyk Python s rozšířeními SciPy a NumPy [13], který není příbuzný s výše uvedenými jazyky.

Cílem této bakalářské práce je provést srovnání volně dostupných simulačních nástrojů. Pro účely porovnání budou implementovány simulační modely v různých programovacích jazycích. Tyto modely budou následně porovnány pomocí porovnávací metodiky a získané výsledky srovnání budou zhodnoceny z hlediska vhodnosti použití implementovaných modelů pro zvolené nástroje [14].

V kapitole 2 jsou popsány a vysvětleny důležité pojmy potřebné pro pochopení problematiky modelování a simulací. Dále kapitola pojednává o zvolených programovacích jazycích. Kapitola 3 obsahuje jednoduché rozdělení použitých nástrojů a jejich bližší popis. Kapitola 4 se zabývá popisem implementovaných simulačních modelů pro simulační nástroje. Společně s implementací je v této kapitole vysvětlena porovnávací metodika, která je následně využita v kapitole 5, která obsahuje výsledky porovnání simulačních nástrojů zároveň s popisem způsobilosti použitých modelů pro využité nástroje podle zvolené metodiky.

Kapitola 2

Modelování a simulace

Modelování a simulace jsou dnes již nedílnou součástí průmyslu, vědy a výzkumu. Tato kapitola se věnuje vysvětlení pojmů nutných k pochopení základní problematiky modelování a simulací. Dále lze v kapitole nalézt výčet výhod a nevýhod využití simulací. Poslední podkapitola se věnuje popisu programovacích jazyků využitých pro účely bakalářské práce.

2.1 Problematika modelování a simulací

Jelikož jsou modelování a simulace velmi komplexní disciplíny, je nutné definovat, alespoň v jednoduchosti, základní pojmy, které jsou v rámci této práce využívány.

2.1.1 Systém

Systém [21] je možno definovat jako soustavu částic, které na sebe vzájemně působí. Systémy se mohou dělit na:

- Otevřené a uzavřené – podle jejich interakce s okolním prostředím
- Spojité a diskrétní – podle toho, zda se hodnoty systému mění spojitě, nebo skokově
- Statické a dynamické – podle toho, zda se systém vyvíjí v čase

Pro potřeby simulací jsou využívány především dynamické systémy, u kterých můžeme sledovat jejich vývoj v průběhu simulace [4].

2.1.2 Modelování a model

Modelování je vědecká disciplína, při které popisujeme systém většinou z reálného světa a získáváme tak všechny možné informace, které k tomuto systému máme. Ze získaných poznatků vytváříme model. V rámci zkoumání získaných informací je nutná analýza možných nepochopitelných a nepopsatelných vlastností modelu. Model je tedy jen přiblížení skutečnosti, nikoliv její dokonalý odraz. Modely lze dělit podle jejich chování na [21]:

- spojité – hodnoty modelu se v průběhu simulace mění spojitě. Tyto modely jsou popsatelné diferenciálními rovnicemi
- diskrétní – hodnoty modelu se v průběhu simulace mění skokově. Tyto modely lze popsat například jako konečný, nebo celulární automat

- kombinované – tyto modely obsahují společně spojité a diskrétní složky vedle sebe

V literatuře můžeme nalézt další možné rozdělení modelů [4].

S takto popsanými modely je pak možné vykonávat simulace. Během simulace se pozoruje chování modelu, vyhodnocují se výsledky simulací a případně se mění popis modelu, pokud to vyžaduje simulace, nebo pokud se přijde na nové, nečekané chování.

Model je také možné vytvářet pomocí tzv. nereálného systému. Takovéto systémy jsou často k nalezení v počítačových hrách, případně ve vědeckých oborech, kde jsou získané poznatky o chování systému velmi omezené.

2.1.3 Simulace

Simulace je disciplína zabývající se získáváním informací z modelů pomocí provádění experimentů s nimi. Pro získání validních informací o modelu je potřeba provést simulaci vícekrát a následně porovnat a vyhodnotit výsledné hodnoty.

Před během simulace a po vyhodnocení výsledků je vhodné provést analýzu vhodnosti modelu pro simulaci. Pokud shledáme model nevalidním, není vhodný pro náš záměr a je záhodno jej z dalších simulací vyřadit.

Simulace můžeme rozdělovat podle mnoha kritérii:

1. Podle typu použitého modelu:
   - Spojitá – rozdělení je závislé podle typu použitého modelu, viz 2.1.2
   - Diskrétní
   - Kombinovaná
2. Podle zpracování výsledků:
   - Kvalitativní – v rámci kvalitativní metody nevnímáme model jako komplexní objekt, ale zaměřujeme se pouze na pár vybraných kvalit
   - Kvantitativní – tato metoda zpracování je vhodná pro modely, které neumíme přesně zapsat a popsat
3. Podle rozložení výpočtů:
   - Simulace na jednom stroji – pro účely výpočtu je využit jen jeden výpočetní stroj (procesor, cluster). Toto využití je vhodné pro simulace jednoduššího rázu.
   - Paralelní a distribuovaná simulace – simulace probíhá na více strojích zároveň. Pomocí dílčího rozdělení výpočtů můžeme dosáhnout značného zrychlení v případě složitých simulací, např. lety do kosmu.

Další a mnohá rozdělení simulací můžeme nalézt v literatuře [21]. Pro účely různých druhů simulací jsou rozdělení jiná, nemluvě o dynamickém vývoji tohoto odvětví a tudíž i neustálým změnám v rozdělení simulací.

V této práci se věnuji pouze simulacím spojitým a kombinovaným. Diskrétní simulace, které simulují například systémy hromadné obsluhy, výrobní linky a obchody, v této bakalářské práci řešeny nebudou z důvodu velké odlišnosti a rozsahu práce.

2.2 Postup modelování a simulace

Jak už bylo řečeno v kapitole 2.1.2, modely jsou tvořeny pomocí pozorování systému a získávání poznatků o něm. Pro účely popisu tvoření modelu budeme vycházet z předpokladu, že pozorujeme tzv. reálný systém. Na základě získaných reálií o systému vytvoříme abstraktní model, který je vlastně "zjednodušenina" pozorovaného systému, díky tomu, že pracujeme pouze s daty, která získáme pomocí pozorování. Takto vytvořený model může být popsán například diferenciálními rovnicemi [4].

Z takto popsaného abstraktního modelu dále vytvoříme simulační model. Simulační model je vlastně přepis abstraktního modelu do programovacího jazyka tak, abychom byli schopni s tímto modelem provádět měření a experimenty. Simulační model je tedy program/skript, který počítá výsledky podle zadaných vstupních hodnot, počátečního stavu, parametrů modelu a případných proměnných.

S výsledky získanými ze simulačního modelu získáváme další poznatky o chování abstraktního modelu. Tyto získané výsledky můžeme využít pro úpravu abstraktního modelu, následnou úpravu simulačního modelu podle modelu abstraktního a na závěr opětovné provedení simulace [21].

Výsledky je vhodné vizualizovat tak, abychom s nimi mohli dále pracovat a analyzovat je. V rámci analýzy je potřeba rozlišit, s jakým typem dat pracujeme, a podle toho zvolit metodu, pomocí které budeme analýzu provádět. Mezi běžné druhy analýzy výsledků patří:

- Porovnání získaných výsledků s reálnými naměřenými daty daného systému
- Použití statistických metod pro zpracování výsledků
- Automatické vyhodnocení výsledků simulace podle metodiky zvoleného simulačního nástroje

V rámci simulace můžeme sledovat i hodnoty, které s výsledkem simulace souvisí nepřímo. Mezi tyto hodnoty patří například:

- Strojový čas – čas, po který simulace využívá výpočetní stroj
- Přesnost numerické metody – mění se v závislosti na použité numerické metodě
- Alokace zdrojů při simulaci – využití fyzických a virtuálních zdrojů výpočetního stroje

Část z těchto proměnných budeme sledovat v rámci naší analýzy. Popis sledování lze nalézt v kapitole 4 a následné porovnání se nachází v kapitole 5.

2.3 Výhody a nevýhody využití simulací

Jako všechny procesy, tak i simulace mají své pozitivní i negativní vlastnosti. V jednoduchosti se dá říct, že simulace šetří čas a peníze, na druhou stranu však mohou stát mnoho zdrojů (peněz i času) pro to, aby vůbec byly vytvořeny a provedeny podle reálného podkladu.

Výhody využití simulací

Jak už bylo zmíněno výše, mezi hlavní výhody patří ušetřený čas a finanční prostředky potřebné pro provedení experimentu. Kromě nich však můžeme zmínit ještě bezpečnost, replikovatelnost a rychlost. Bližší popis výhod:

1. Cena zakázky je v dnešní době velmi důležitým faktorem pro potřeby simulací, protože experimenty s reálným systémem mohou být velmi nákladné. Díky využití modelů a simulací je možné výrazně ušetřit na výrobě testovacích kusů. Toto je dobře patrné například v leteckém průmyslu - reálné systémy (letadla nebo jejich komponenty) se zde využívají až pro validaci výsledků předchozích simulací [2].

2. Rychlost je další z výhod pro běh simulací. V závislosti na výkonu výkonu stroje, který simulaci vykonává, můžeme simulaci zpomalovat a zrychlovat dle našich potřeb. Důležitým faktorem je zde výkon výpočetního zdroje, který nám omezuje maximální (minimální) rychlost simulace. Dobrým příkladem pro nutnost zrychlování simulace je sledování pohybu litosferických desek. Tento pohyb je v reálném systému velmi pomalý, ale pokud máme dostatečně výkonný výpočetní stroj, můžeme simulaci zrychlit a dosáhnout výsledků mnohem rychleji.

3. Replikovatelnost je faktor, který velmi úzce souvisí s cenou simulace. Pomocí simulace můžeme provést výpočty na jednom modelu prakticky nekonečněkrát bez potřeby využití reálného systému. Pomocí replikovatelnosti jsme schopni dosáhnout přesnějších výsledků a tím pádem i větší znalosti vlastností objektů [3].

4. Bezpečnost je další faktor, který je velkou výhodou využití simulací. Některé experimenty s reálnými systémy není možné z bezpečnostních důvodů vůbec provádět - například crash testy letadel, simulace chemických a jaderných reakcí, nebo simulace epidemií různých nemocí.

5. V neposlední řadě je to možnost experimentovat se systémy, které můžeme blíže zkoumat právě jen díky provádění simulací s nimi - například působení černých děr [7], kolize vesmírných těles, a podobně.

Tento výčet výhod simulací však není konečný. Je velmi pravděpodobné, že se v blízké době objeví další velká výhoda tohoto procesu. Jelikož výpočetní výkon počítačů neustále roste podle Moorova zákona [23], časem bude možné provádět experimenty s daleko složitějšími modely než doposud. Výsledky simulací se využívají mimo vědu také pro populárně naučné účely, případně pro potřeby využití ve filmovém a herním průmyslu [10].

Nevýhody využití simulací

Stejně tak, jako simulace mohou naše zdroje šetřit, při nesprávné tvorbě modelu, případně nesprávném zavedení simulací, se mohou modely velmi prodražit a prodloužit časy simulací. Se simulacemi se samozřejmě pojí více rizik, než jsou finance a čas:

1. Mezi běžné problémy všech výpočetních operací patří bezesporu velké nároky na výpočetní výkon. S rostoucími nároky na rychlost a přesnost výsledků simulací přímo úměrně roste také nárok na výkon simulačního stroje. Nicméně díky zvětšování výkonu v závislosti na Moorovu zákonu, viz kapitola 2.3, je možné provádět složitější výpočetní operace i na běžných osobních počítačích a ne pouze na sálových superpočítačích, jako tomu bylo dříve.

2. Během simulace může docházet k chybám kvůli nevhodně zvolené numerické metodě, případně kvůli její nepřesnosti řešení. Tento stav dokáže ovlivnit i relativně bezchybně namodelovaný model a znehodnotit tak výsledky z experimentů s ním.

3. Velká časová náročnost v případě, že chceme simulaci provést vícekrát. Tento problém je spjatý i s nároky na výpočetní výkon – je-li výkon dostatečný, můžeme simulaci urychlit a tím i zkrátit čas simulace.

4. Velmi důležitým problémem je problém ověřování validity. Pokud máme chybně namodelovaný model, pak výsledky simulace obsahují také chyby a tedy model není validní. Ověřování validity modelu by mělo proběhnout ještě před provedením simulace, tak, abychom předešli následnému přepracování modelu.

Stejně jako u výhod simulací v kapitole 2.3 není tento výčet konečný. Další velkou nevýhodou je například vysoká složitost vytvoření modelu. A stejně jako u výhod, i zde je velmi pravděpodobné, že se další velké nevýhody budou objevovat s přibývajícím časem.

2.4 Charakteristiky použitých jazyků

Jazyků umožňujících vytváření a specifikaci modelů a provádení experimentů s nimi je na trhu nepřeberné množství [20]. Tyto jazyky se také liší v případech využití - pro různé typy modelů a simulací systémů jsou jinak vhodné. Pro potřeby bakalářské práce jsem si zvolil tři programovací jazyky, z toho dva z kategorie jazyků, které jsou využívány vybranými simulačními nástroji. Nástroje tyto jazyky používají buď v čisté formě, případně jejich modifikované verze určené pro konkrétní nástroj. Těmito jazyky jsou MATLAB[17] a Modelica[8]. Jako třetí programovací jazyk byl pro porovnání zvolen jazyk Python s podporou matematických, vědeckých a simulačních knihoven a knihoven pro tvorbu grafů [13].

Simulačních jazyků, jejich modifikací a nástrojů, které je následně využívají, je nespočet [22]. Z těch nejznámějších to je například FORTRAN, který posloužil jako základ pro jazyk MATLAB, dále pak třeba SIMULA a Smile. Výše zvolené jazyky (MATLAB, Modelica, Python) byly zvoleny kvůli jejich rozšířenosti, díky které existuje mnoho nástrojů, které tuto trojici jazyků využívají.

MATLAB

Jazyk MATLAB je uzavřený skriptovací jazyk, který je vyvíjený a spravovaný firmou MathWorks od roku 1984. Tento jazyk je spjatý se stejnojmenným nástrojem MATLAB [17]. Jazyk je využíván k popisu analýzy dat, k návrhu a implementaci algoritmů a také k vytváření modelů a provádění operací (simulací) s nimi. Jazyk MATLAB se vyznačuje velmi slabou typovou kontrolu a také tím, že se všemi objekty je pracováno jako s maticemi, v případě jednořádkové matice, vektoru, je práce s touto maticí shodná s prací s polem hodnot. Většina vestavěných funkcí je pro tento účel naprogramována a očekává na svém vstupu právě matici. Syntax zápisu kódu v MATLABu připomíná kterýkoliv jiný funkcionální jazyk. Jazyk MATLAB podporuje také objektově orientované programování. Pro účely zpracování a analýzy získaných výsledků dává jazyk možnost vykreslit grafy. Prostředí MATLABu je navrženo tak, že navíc dokáže pracovat s funkcemi napsanými v jazyce C a Fortran. Zároveň má jazyk možnost využít knihovny napsané v jiných jazycích [17], jako jsou například Java, .NET nebo Perl. Výsledky práce s nástrojem MATLAB můžete nalézt v kapitole 5.

Tento programovací jazyk byl vybrán kvůli rozšířenosti na středních a vysokých školách, ale také na různých pracovištích zabývajícími se výpočty, modelováním a simulacemi. Navzdory tomu, že nástroj MATLAB není volně dostupný, na trhu jsou k dostání volně dostupné nástroje, jako například GNU Octave [12] a SciLab [24], které jsou s jazykem

MATLAB do určité míry kompatibilní. Také tyto nástroje jsou ve výběru nástrojů využitých v této bakalářské práci. Dále pro MATLAB existuje rozšíření SIMULINK, které do prostředí MATLABu přidává možnost provádět vícerozměrné grafické simulace.

Popis nástroje MATLAB lze nalézt v kapitole 3.2.1.

Modelica

Jazyk Modelica je objektově orientovaný, modelovací a simulační jazyk, který je vyvíjen neziskovou organizací Modelca Association od roku 1997. Díky tomu, že je tento jazyk volně dostupný, je na trhu mnoho simulačních nástrojů a vývojových prostředí, které pracují právě s tímto jazykem. Pro účely této práce byly vybrány nástroje Dymola [8], OpenModelica [8] a jModelica.org [11]. Vývojové prostředí Dymola není, stejně jako MATLAB, volně dostupné. Pro účely provádění experimentů s tímto vývojovým prostředím je využita licence, pomocí které je možno spustit nástroj Dymola na školním serveru Merlin v rámci VUT FIT.

Modely, které jsou zapsány v jazyku Modelica, mají veškeré své chování popsané rovnicemi. Například rovnice Newtonova zákona ochlazování, se v jazyce Modelica zapíše naprosto totožným způsobem [26]. Výsledné modely jsou pak snáze čitelné i bez komentářů a případných vysvětlivek.

Pro to, aby byl model funkční, je nutné definovat vstupní hodnoty jednotlivých proměnných. Zapsaná rovnice se tedy chová naprosto stejně, jako když počítáme rovnici na papíře. Použítím znaku = tedy nedochází k přiřazení, jako je to běžné u jiných programovacích jazyků, ale slouží zde jako znak pro porovnání - proto tedy ona práce s rovnicemi.

Jazyk je v současnosti velmi využíván v oblasti proudění kapalin a termodynamiky, vzhledem k velmi jednoduchému vytvoření modelu podle předem stanovených rovnic. Dále se pak používá ve větší míře v elektrotechnice a strojírenství [26].

Výsledky běhu simulací nástrojů využívajících jazyk Modelica jsou dostupné v kapitole 5.

Python

Python je volně dostupný interpretovaný programovací jazyk, který má nepřeberné množství využití. Poprvé byl představen v roce 1991. Jazyk Python není v základním pojetí simulačním nebo modelačním jazykem. Tuto funkcionalitu získává teprve s dodatečnými knihovnami. V rámci využití v této bakalářské práci byla vybrána rodina knihoven SciPy. Hlavním přínosem této rodiny knihoven je možnost použití metod integrace, lineární algebry, rychlé Fourierovy transformace a řešení diferenciálních rovnic.

Tento jazyk byl zvolen z důvodu, který byl řečený výše. Python nemá primární použití jako jazyk vhodný pro modelování a simulace. Nicméně s jeho rostoucí popularitou a rostoucí uživatelskou základnou, bude jeho využití růst ještě více. Momentálně má Python velmi široký záběr pokrytí - od funkcionální, přes logické programování, až po tvorbu grafiky, databází, nebo provádění simulací. Jazyk je využíván velkými firmami na trhu, jako jsou například Google, NASA nebo Wikipedia. Velmi často je v dnešní době využíván k implementaci strojového učení [15].

Jako simulační knihovny k tomuto jazyku byla vybrána rodina knihoven SciPy [13] [19]. Pomocí těchto knihoven je následně možné provádět náročnější matematické operace, simulace a vizualizaci dat. Bližší popis této knihovny je k nalezení v kapitole 3.2.8.

Stejně jako u jazyků MATLAB a Modelica jsou výsledky simulací v jazyce Python dostupné v kapitole 5.

Kapitola 3

Popis použitých simulačních nástrojů

Obsahem této kapitoly je rozdělení zvolených simulačních nástrojů, které lze nalézt v kapitole 3.1. Následující podkapitola 3.2 obsahuje bližší popis jednotlivých nástrojů společně s jejich výhodami a zajímavostmi. Vybranými nástroji pro porovnání v rámci bakalářské práce jsou:

- MATLAB
- GNU Octave
- SciLab
- FreeMat
- Dymola
- OpenModelica
- jModelica
- Python + SciPy

Výše uvedené nástroje byly vybrány tak, aby všechny byly volně dostupné a zároveň podporovaly alespoň jeden jazyk ze zvolených v kapitole 2.4. Nicméně dva nástroje z výběru nejsou primárně volně dostupné. Prvním z nich je MATLAB, u kterého je volná dostupnost zaručena využitím 30 denní zkušební verze, která je zdarma ke stažení. Druhým takovým nástrojem je Dymola, u které je dostupnost zaručena díky školní licenci VUT FIT a je dostupná na serveru Merlin. I zde je dostupná zkušební verze zdarma po omezenou dobu.

3.1 Rozdělení použitých simulačních nástrojů

Zvolené simulační nástroje můžeme rozdělit podle mnoha kritérií. V tabulce 3.1 lze nalézt dělení podle základních vlastností, jako jsou jazyk, dostupnost, nebo podporované platformy. Jak je z tabulky patrné, všechny nástroje jsou spustitelné na všech běžných platformách (Windows, Linux, MacOS), kromě nástrojů Dymola a jModelica, které jsou spustitelné pouze v prostředí Windows a Linux. Všechny v tabulce zmíněné nástroje mají

	MATLAB	GNU Octave	SciLab	FreeMat	Dymola	OpenModelica	jModelica	Python + SciPy
Jazyk	MATLAB	MATLAB	MATLAB	MATLAB	Modelica	Modelica	Modelica	Python
Grafické prostředí	Ano	Ano	Ano	Ano	Ano	Ano	Ano[1]	Ne
Platforma	Vše	Vše	Vše	Vše	Windows, Linux	Vše	Windows, Linux	Vše
Podpora jiných jazyků	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano
Možnost online aplikace	Ano	Ano	Ano	Ne	Ne	Ne	Ne	Ano
Dostupnost	Komerční licence	Zdarma	Zdarma	Zdarma	Komerční licence	Zdarma	Zdarma	Zdarma
Opensource[2]	Ne	Ne	Ano	Ano	Ne	Ano	Ano	Ano

Tabulka 3.1: Přehled základních vlastností porovnávaných simulačních nástrojů

také do určité míry vlastní grafické prostředí, až na nástroj SciPy, který je popsán v kapitole 3.2.8.

3.2 Popis použitých nástrojů

V následující kapitole jsou podrobně popsány simulační nástroje využité v rámci bakalářské práce. Kapitola neobsahuje žádné dělení podle vlastností. Pro účely dělení lze použít tabulku 3.1 z kapitoly 3.1.

3.2.1 MATLAB a Simulink

Simulační nástroj MATLAB, je stejně jako jazyk MATLAB, který je popsán v kapitole 2.4, uzavřený nástroj vyvíjený společností MathWorks. Nástroj je mateřským prostředím pro nástroj Simulink.

MATLAB není primárně nástroj určený k modelování, avšak díky velmi širokému rozpětí implementovaných knihoven, tento nástroj také nabízí rozsáhlý záběr možností využití, od matematických výpočtů, přes použití pro ekonomii, až například po využití pro genetické algoritmy ve strojírenství [27].

MATLAB má také vlastní rozhraní pro plnohodnotnou webovou instanci tohoto simulačního programu[3]. Pomocí tohoto webového nástroje je možné svůj kód/model sdílet společně s dalšími uživateli a tak společně pracovat na jednom projektu. Porovnání a zhodnocení webového rozhraní nejsou v této bakalářské práci zapracovány.

Simulink je programovatelná grafická nástavba pro nástroj MATLAB. Tento nástroj je také vyvíjen společností MathWorks. Hlavní využití nástroje je jednoduché vymodelování systému pomocí grafických programovatelných bloků, respektive blokových schemat. Pro tyto účely jsou v rámci nástroje Simulink předem vytvořeny a předdefinovány bloky pro tvorbu dynamických a spojitých modelů. Další výhodou je přímé generování kódu v jazyce C, případně v jazyce C++. Běh simulací s takto vytvořenými modely je možné sledovat v reálném čase.

Nástroj MATLAB bohužel není volně dostupný, nicméně v akademickém prostředí je velmi běžně k dostání, stejně jako v následné praxi. Pro účely bakalářské práce je však využita běžná 30 denní zkušební licence, dostupná zdarma z webových stránek společnosti MathWorks[4], ve verzi MATLAB_R2019a.

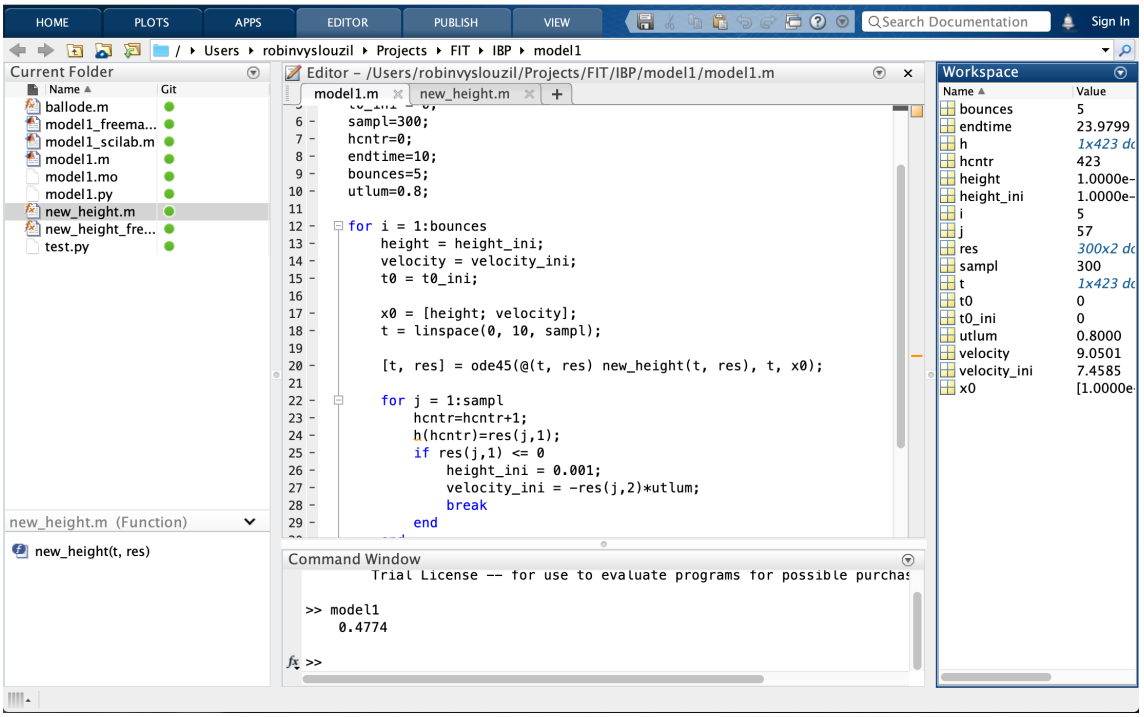
Simulink je, stejně jako MATLAB, uzavřený a komerční nástroj. Jako volně dostupnou alternativu bychom mohli považovat nástroj OpenModelica, který je popsán v kapitole 3.2.7.

[1] Jen s příslušným rozšířením, viz 3.2.6

[2] Volně dostupné zdrojové kódy, například na serveru github.com

[3] https://www.mathworks.com/products/matlab-online.html

[4] https://www.mathworks.com/campaigns/products/trials.html?procode=ML



Obrázek 3.1: Ukázka prostředí MATLAB

3.2.2 GNU Octave

Nástroj Octave je vyvíjený pod veřejnou licencí GNU General Public Licence (GPL) [12]. Syntax jazyka, který je použit k modelování v GNU Octave, je velmi podobný jazyku MATLAB. Hlavním rozdílem v syntaxi je ten, že Octave podporuje autoinkrementaci stejně, tak jako například jazyk C – MATLAB tuto podporu bohužel nemá. Mezi další rozdíly patří drobně odlišné syntaktické zápisy, případně volání jinak pojmenovaných funkcí, například pro matematickou operaci umocňování se v Octave používá zápis `a^b` nebo `a**b` [12], zatímco v MATLABu je možné tuto operaci zapsat pouze jako `a^b` [17].

Hlavní výhodou nástroje GNU Octave je, jak už bylo nastíněno výše, že je volně otevřený a zdarma. Tento nástroj má poměrně širokou komunitu uživatelů, kteří vyvíjí další rozšiřující balíčky pro potřeby simulací, matematiky, statistiky a dalších jiných vědeckých odvětví[5].

Nástroj byl nainstalován bez obtíží podle oficiálního návodu v dokumentaci nástroje [12]. Pro účely ověření funkčnosti byl nástroj nainstalován na všech platformách, které by měl nástroj podporovat. V rámci bakalářské práce byla použita poslední aktuální verze 5.1.0.

3.2.3 SciLab

Další z nástrojů, který je volně dostupnou alternativou MATLABu, je SciLab. Syntax jeho jazyka je zase podobný syntaxu MATLABu, ovšem už ne do takové míry, jako u nástroje GNU Octave [6]. Kód napsaný pro SciLab tedy není volně přenositelný do prostředí MATLABu, ale díky velké podobnosti syntaxe je možné kód refaktorovat pro tento nástroj.

[5] https://octave.sourceforge.io/packages.php

Podobně jako u MATLABu, i nástroj SciLab nabízí možnost využití vlastního webového rozhraní[6]. Přítomnost této možnosti používání bych označil za velkou výhodu oproti jiným volně dostupným nástrojům. Ani pro tento nástroj nebude tato možnost více rozebírána v rámci této bakalářské práce.

Instalace nástroje proběhla bez problému podle návodů v oficiální dokumentaci a byl také proveden test instalace pro všechny podporované platformy. Nástroj byl nainstalován ve verzi 6.0.2.

3.2.4 FreeMat

Nástroj FreeMat je posledním vybraným nástrojem, který primárně podporuje jazyk MATLAB. Tento nástroj je volně dostupný pod licencí GPL a byl vyvíjen kolektivem autorů, jejichž seznam je dostupný na oficiálních stránkach nástroje[16]. Poslední verze nástroje FreeMat vyšla v roce 2013, tudíž je možné ho již považovat za neaktuální. Nicméně jeho komunita je stále aktivní. Do jisté míry je to tím, že zdrojové texty tohoto nástroje jsou volně dostupné na serveru SourceForge[7]. FreeMat se také používá v rámci výuky na VUT FIT, jako náhrada za nástroj MATLAB.

Skladba jazyka použitého v FreeMat je podle oficiální dokumentace přibližně na 95 % kompatibilní s jazykem MATLAB. Oproti MATLABu je zde větší podpora pro import kódu zvenčí a lepší zobrazování 3D vizualizací dat[16].

Instalace probíhá pomocí předpřipravených balíků dostupných z oficiální stránky, které jsou dostupné pro běžné platformy.

3.2.5 Dymola

Nástroj Dymola je uzavřený komerční nástroj vyvíjený firmou Dassault Systèmes. Tento simulační nástroj je založený, jak už bylo řečeno v rozdělení v kapitole 3.2, na jazyce Modelica. Dymola umožňuje jednoduché vytvoření modelu a pak následně provádění simulací s ním. Tento nástroj je velmi využíván ve strojírenství, elektrotechnice a energetice[8].

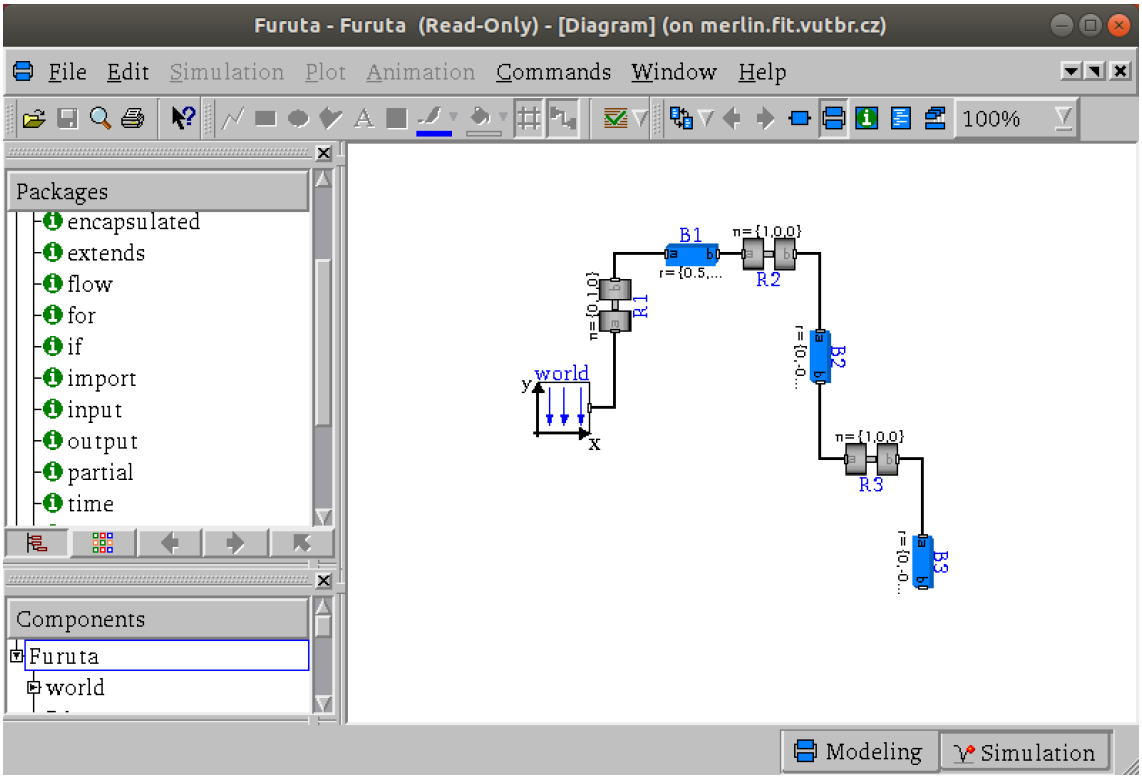
Pro účely práce je využita instalace nástroje Dymola ve verzi 6.0b, která vyšla v roce 2006, na školním serveru Merlin. Aktuální verze Dymola 2019 FD01 vyšla 30. listopadu 2018. Dymola má také k dispozici, stejně jako MATLAB, zkušební časově omezenou verzi zdarma. Nicméně pro účely srovnání mi přišlo zajímavé využít právě školní licenci, která je daleko více přístupná pro případné využití studenty.

3.2.6 jModelica

Jednou z možných alternativ k nástroji Dymola je jModelica. Tento nástroj je volně dostupný, otevřený a stejně jako GNU Octave, který byl popsán v kapitole 3.2.2, vyvíjený pod licencí GPL.

jModelica podporuje přímé generování ze zdrojového kódu napsaného v jazyce Modelica do jazyků C a XML. Tato funkcionalita je velmi přínosná, pokud například chceme využívat nástroj jModelica jen jako modelovací prostředek a následné simulace spouštět v jiném prostředí. Dále také podporuje využití jazyka Python a jeho knihoven. Aplikace má uživatelské rozhraní na bázi příkazové řádky, které je napsáno v jazyce Python. Aplikace sama o sobě grafický editor nemá, lze ho však emulovat pomocí rozšíření do editoru Eclipse [1]. Pro

[6] https://www.scilab.org/cloud/web-application
[7] https://sourceforge.net/projects/FreeMat/files/
[8] https://www.3ds.com/products-services/catia/products/dymola/key-advantages/



Obrázek 3.2: Ukázka prostředí Dymola

řešení diferenciálních rovnic jModelica používá balík Assimulo[9]. Tento balík funkcí je také napsán v jazyce Python. jModelica má volně dostupný repozitář se všemi zdrojovými kódy všech verzí, které byly vydány[10]. Zajímavostí je, že jsou přístupné i zdrojové texty verzí, které ještě nejsou oficiálně vydány. Aktuální verze, a také verze na která byla instalována, je verze 2.4, nicméně v repozitáři je dohledatelná i verze 2.8.x, která teprve na svoji stabilní verzi čeká.

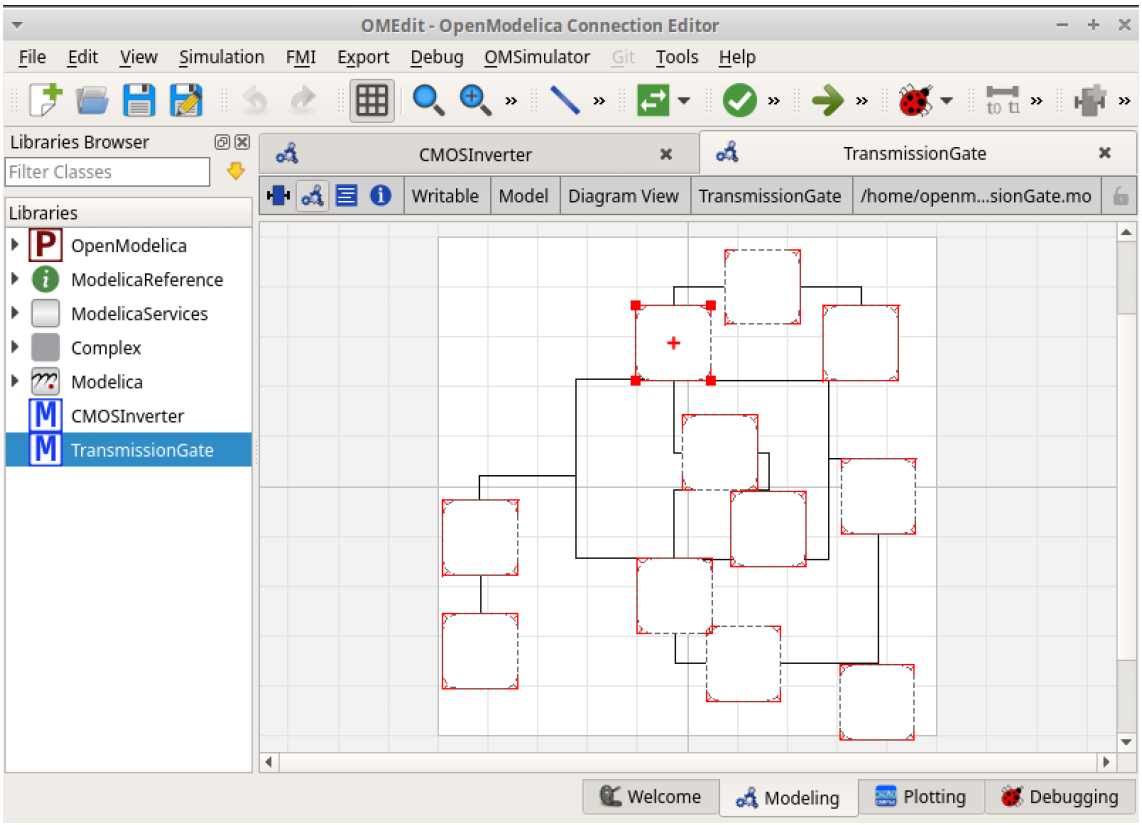
Dříve byla jModelica dostupná pro všechny běžné operační systémy. Od verze 2.0 jsou podporovány již jen operační systémy Linux a Windows a podpora pro MacOS byla zastavena. Verze 1.9 byla tedy poslední, kdy byl systém MacOS oficiálně podporován.

3.2.7 OpenModelica

Simulační nástroj OpenModelica je vyvíjena společenstvím autorů známých pod jménem Open Source Modelica Consortium (OSMC). OpenModelica má jako jeden ze dvou vybraných nástrojů volně dostupné zdrojové kódy[11]. Tato skutečnost nahrává k vývoji vlastních modifikací a uživatelských pluginů.

Obecně se nástroj OpenModelica udává jako jedna z alternativ pro prostředí MATLAB + Simulink. Další takovou alternativou je nástroj SciLab popsaný v kapitole 3.2.3. Dále je jednou z možných alternativ nástroje Dymola. Z obrázků 3.2 a 3.3 je patrné, že i grafické rozdíly v GUI jednotlivých nástrojů nejsou nijak markantní.

[9] https://jmodelica.org/assimulo/
[10] https://svn.jmodelica.org/
[11] https://github.com/OpenModelica



Obrázek 3.3: Ukázka prostředí OpenModelica

OpenModelica, jako jediný nástroj z výběru, nabízí možnost si stáhnout předinstalovanou verzi OpenModelica na virtuálním disku s operačním systémem Linux, konkrétně na distribuci Xubuntu. Zároveň s touto možností OpenModelica podporuje běžné operační systémy, pro které byla také provedena zkušební instalace.

3.2.8 SciPy

Tento simulační nástroj jako jediný není založený na jazycích MATLAB, či Modelica, nýbrž na skriptovacím jazyce Python. Není to simulační nástroj v pravém slova smyslu, jedná se spíše o rozšiřující knihovny jazyka Python, které jsou zdarma a volně dostupné[12] [13]. V rámci této rodiny knihoven jsou například knihovny:

- SciPy – tato knihovna obsahuje numerické algoritmy pro provádění simulací, zpracovávání signálů a následné statistické vyhodnocení simulace
- NumPy – obsahuje metody pro numerické výpočty, práci s číselnými poly a maticemi
- Matplotlib – pomocí této knihovny je možné na základě vstupních dat vykreslovat grafické výstupy ve 2D a i ve 3D

SciPy nemá žádné vlastní grafické prostředí pro tvorbu modelů a následný průběh simulací probíhá proto v běžném IDE pro vývoj kódu v jazyce Python.

Instalace probíhá pomocí správce balíčků pro daný operační systém, případně přes správce balíčků jazyka Python. Knihovny SciPy jsou kompatibilní jak s Python2, tak i s Python3. Nástroj SciPy byl v této bakalářské práci použit ve verzi 1.2.1.

[12] https://github.com/scipy/scipy

Kapitola 4

Návrh metodiky a popis implementovaných modelů

V této kapitole bude představena použitá porovnávací metodika, kritéria hodnocení a následné způsoby jejich vyhodnocení. Tato metodika bude následně použita pro vyhodnocení výsledků získaných z běhu simulací s modely, které jsou také popsány v této kapitole. Zároveň s popisem modelů lze v této kapitole najít i popis jejich implementace a popis nastalých problémů během ní.

4.1 Popis vytvořené metodiky

Předlohou k tvorbě vyhodnocovací metodiky sloužila metodika ke srovnávání nástrojů simulujících dopravní situace na pozemních komunikacích [14]. Původní převzatá metodika sledovala kritéria jako hardware a software náročnost, náročnost na kvalitu vstupních dat a pak další kritéria ve vztahu k použitým nástrojům. Pro účely bakalářské práce byla vybraná metodika modifikována podle potřeb zadání a následně byla vytvořena tato kritéria:

- Strojový čas – čas, po který simulace běží na výpočetním stroji
- Hardwarové a softwarové požadavky – v rámci tohoto kritéria bude hodnocena kompatibilita nástrojů napříč standardními platformami (Windows, Linux, MacOS), případně zde budou uvedeny omezení, která vyplynou z instalace a běhu experimentu
- Obtížnost implementace modelu – u tohoto kritéria lze namítat, že je velmi subjektivní. Obecně lze předpokládat, že pokud budou vybrány základní typy modelů pro různé druhy simulací, bude obtížnost implementace nízká, pokud bude jazyk, nebo nástroj, vhodný pro tuto implementaci. V opačném případě bude obtížnost implementace označena jako vysoká
- Obtížnost instalace – všechny nástroje budou nainstalovány pomocí návodů v oficiálních dokumentacích. Nebude-li tomu tak, pak v rámci hodnocení obtížnosti instalace jsou uvedena úskalí vyvstalá během instalace

V rámci hodnocení nebudeme používat žádnou škálu, k hodnocení "úspěšnosti"simulačního nástroje. Výsledek hodnocení bude spíše doporučení, jestli je daný simulační nástroj, případně jazyk, vhodný pro daný typ úloh.

4.2 Implementované modely

V následující kapitole jsou popsány jednotlivé modely a jejich implementace pro potřeby běhu experimentu v simulačních nástrojích. Společně s implementací jsou popsány zjištěné problémy během implementování a jejich řešení.

Skákající míček

V rámci této implementace je namodelován systém skákajícího míčku, který je z nedokonale pružného materiálu a postupně během skoků ztrácí svou rychlost a výšku odrazu. Míček je vypouštěn z výšky h, postupně nabere rychlost v_p a narazí do země. Takový model lze popsat jednoduchou diferenciální rovnicí, která má znění:

$$h' = v_p \tag{4.1}$$

$$v_p' = -9.81 \tag{4.2}$$

s počáteční podmínkou $h(0) = 10$. Pro účely experimentu jsou další počáteční hodnoty nastaveny takto:

Proměnná	Hodnota	Jednotka
h	10	m
v_p	15	$m.s^{-1}$
koeficient pružnosti	0.8	

Tabulka 4.1: Vstupní hodnoty modelu skákajícího míčku

Po každém dopadu se vypočte nová výška, do které míček vyletí a nová rychlost, kterou míček dopadne na zem. Tyto vypočtené hodnoty se následně použijí pro výpočet další iterace.

Pro implementaci posloužil jako inspirace kód pro nástroj MATLAB převzatý z oficiální dokumentace MathWorks[1]. Pro účely implementace v jazyce MATLAB byl nejprve model naprogramován pro nástroj SciLab a následně kód refaktorován pro každý nástroj zvlášť, podle potřeb jednotlivého nástroje. V rámci řešení bylo nutno předem určit počet odrazů tak, aby se zajistil průběh výpočtu.

Během implementace v jazyku Python se vyskytl problém s detekcí výšky a výpočtu následného odrazu. Původní zvolená metoda `odeint()` byla vyhodnocena jako nevhodná z důvodu nemožnosti detekce nespojitostí ve výpočtu. Po několika neúspěšných pokusech s touto a dalšími metodami byla zvolena metoda `solve_ivp()`, která tyto situace umí do jisté míry detekovat. Bohužel ani použití této metody nezaručilo možnost implementovat model tak, aby mohl být změřen čas výpočtu.

Při implementaci a následném zkoumání výsledků simulace v jazyce Modelica se jev, kdy se míček "propadne pod zem", objevuje také. Zde je to způsobeno zřejmě přesností výpočtu a následnou práci s velmi malými čisly. Nicméně na průběh výpočtu, navzdory implementace v Pythonu, tento jev nemá vliv, protože se míček propadne až po velmi dlouhé době.

[1] https://www.mathworks.com/help/stateflow/ug/modeling-a-bouncing-ball-in-continuous-time.html

Výsledky implementovaného modelu a následné srovnání výsledků simulačních nástrojů lze nalézt v kapitole 5.1. Zvolený model byl implementován pro všechny nástroje popsané v kapitole 3.2, kromě nástroje SciPy z důvodů popsaných výše.

Kruhový test

Kruhový test je implementován podle návodu ze studijní opory pro předmět IMS - Modelování a simulace [21]. Jedná se o implementaci řešení jednoduché soustavy diferenciálních rovnic. Systém, který byl modelován, je zadán soustavou diferenciálních rovnic:

$$x' = y \tag{4.3}$$

$$y' = -x \tag{4.4}$$

a počáteční podmínka rovnice je zadána jako $y(0) = 1$ a $x(0) = 0$.

Během běhu simulace se výsledky postupně přibližují analytickému řešení, které je definováno jako:

$$y(t) = \cos(t) \tag{4.5}$$

Implementace v jazyce Modelica proběhla velmi snadno. Jak bylo popsáno v kapitole 2.4, zápis v tomto jazyce připomíná zápis výpočtu na papír.

Pro implementaci v Pythonu byla využita metoda `odeint()` z knihovny `scipy.integrate`[2], která je vytvořena pro řešení diferenciálních rovnic a jejich soustav. Pro účely využití této metody je potřeba si definovat obslužnou funkci pro výpočet nových stavů integrátorů. Pro implementaci v tomto případě nebyla využita Eulerova metoda podle příkladu ze studijní opory, neboť rodina knihoven SciPy nemá tuto metodu implementovánu a je doporučeno využívat právě metodu `odeint()`.

Tento model byl implementován také v jazyce MATLAB. K výpočtu byla využita standardní funkce pro výpočet diferenciálních rovnic `dsolve()`. Během studování výpisů ze standardního výstupu jsem objevil, že nástroj MATLAB přímo vypočítává i výsledek analytického řešení, ke kterému by měl výpočet konvergovat. Další nástroje zpracovávající jazyk MATLAB toto chování nevykazovaly. Pro implementaci ve zbytku těchto nástrojů byly použity metody `ode()` a `ode45()`.

Tuhý systém

Tuhý systém[3] je systém, který je popsán matematickou rovnicí, kterou lze označit za tuhou [9]. Tuhá rovnice je diferenciální rovnice, která má při řešení určitými numerickými metodami numericky nestabilní řešení. Systémy, které můžeme označit za tuhé, můžeme najít například v rámci chemických reakcí. Tato nestabilita je způsobena vysokou hodnotou délky kroku při řešení rovnice. Pro řešení se většinou používají jednokrokové metody pro výpočty diferenciálních rovnic. Obvykle je k řešení využita některá z varianty metod Runge-Kutta, nebo i jednoduchá Eulerova metoda [21]. Při výpočtu vícekrokovou metodou se využívají například metody Adamsova, nebo Nyströmova. Využití vícekrokových metod je ale velmi vzácné a podle literatury popisováno jako méně stabilní, než jednokrokové metody [9]. Pro účely experimentu byla vybrána rovnice:

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

[3] Z anglického "stiff system"

$$y' = -50(y - \cos(x)) \tag{4.6}$$

s délkou kroku $h = \frac{1.875}{50}$ a počáteční podmínkou $y(0) = 0$. Pokud by byla délka kroku h větší než $\frac{2}{50}$, pak by došlo k nestabilitě a k oscilování okolo kýženého výsledku [9].

V jazyce Python, konkrétně tedy s využitím knihovny SciPy, se během implementace vyskytl problém s volbou zvolené metody výpočtu zvolené diferenciální rovnice. Většina metod pro výpočet diferenciálních rovnic je implementována tak, že předem detekují, zda-li je zadaná rovnice tuhá, či nikoliv. Pokud ji vyhodnotí jako tuhou, pak si metoda sama upraví velikost kroku na tolik, aby se výsledek co nejvíce přiblížil analytickému řešení. Při zvolení metody, která tuto detekci před výpočtem neobsahuje, vypisuje metoda chybu při zjištění tuhosti rovnice a následné oscilace výsledku okolo analytického řešení. Po studiu možných metod byla nakonec vybrána metoda `odeint()`, která obsahuje detekci pro tuhé rovnice a jejich soustavy.

Implementace pro nástroje Dymola, OpenModelica a jModelica proběhla naprosto identicky s předešlými implementacemi, jen byla změněna diferenciální rovnice a vstupní proměnné. Jazyk Modelica má v tomto jednoduchém přístupu implementace velké výhody oproti jazykům MATLAB a Python, kde je implementace o poznání složitější.

Pro implementaci pro nástroje používající jazyk MATLAB byla vybrána metoda `ode15s()`, která je popisována jako jedna z univerzálnějších metod pro řešení tuhých rovnic a jejich sytémů [17]. Metoda si stejně, jako u implementace v Pythonu, sama zvolí délku kroku podle zadané rovnice. Dalšími metodami pro řešení tuhých rovnic v MATLABu jsou `ode23s()`, `ode23t()` a `ode23tb()`. MATLAB obsahuje více metod, které jsou schopny řešit tuhé rovnice, například `ode45()`, nebo `ode113()`, ale tyto metody jsou označovány za pomalé a jsou primárně doporučovány k řešení netuhých diferenciálních rovnic [17].

Výsledky běhů simulace implementovaných modelů lze nalézt v kapitole 5.3. V této kapitole je také porovnání využitých nástrojů.

Kapitola 5

Simulační experimenty a porovnání výsledků

V této kapitole lze nalézt výsledky z experimentů a následné vyhodnocení porovnání simulačních nástrojů. Každý z modelů, které byly popsány v kapitole 4.2, byl postupně implementován ve všech nástrojích uvedených v kapitole 3.2. Naměřené výsledky jsou zatížené velkým statistickým šumem (např. běh procesů na pozadí), avšak pro účely porovnání jsou využitelné. Všechny naměřené časy jsou čisté časy od začátku skriptu až po konec výpočtu a výpis na standardní výstup. Čas pro případnou tvorbu grafů pro ověření výsledků není v těchto měřeních zakomponován. U všech měření byl přeskočen výsledný čas prvního výpočtu z důvodu několikanásobně delšího průběhu.

Čas byl měřen s pomocí knihoven a funkcí jednotlivých nástrojů. U modelů pro nástroje zpracovávajích jazyk MATLAB byla použita dvojce vestavěných funkcí `tic()` a `toc()`. Pro měření v modelech pro nástroj SciPy byla využita knihovna `time` a následné volání funkce `time.time()` na začátku a na konci výpočtu – k měření času byl použit rozdíl těchto hodnot. Při měření v nástrojích Dymola a OpenModelica byl využit přepínač, který automaticky měřil strojový čas výpočtu a následně jej ukládal do proměnné pro další využití. U nástroje jModelica proběhl manuální sběr naměřených během simulací.

5.1 Model systému skákajícího míčku

Prvním implementovaným modelem byl systém skákajícího míčku, který je popsán v kapitole 4.2. Během implementací tohoto modelu je klíčové zajistit detekci nespojitosti průběhu v době odrazu míčku od země, což se ukázalo jako velmi podstatný problém pro implementaci. Na tento jev nejlépe reagovaly nástroje zpracovávající model napsaný v jazyce Modelica.

Ze zvolených modelů pro jazyk MATLAB je čas výpočtu většiny modelů srovnatelný. Zde je nutno poukázat na vyšší čas výpočtu nástroje FreeMat oproti zbytku nástrojů. Tento nástroj je zatížen výpočtovou chybou, která je blíže popsána v kapitole 5.4 v sekci Strojový čas.

Výsledky naměřené pro jednotlivé modely lze nalézt v tabulce 5.1. Výsledný čas pro nástroj SciPy nebyl naměřen z důvodu nemožnosti detekce nespojitostí, který je popsán v kapitole 4.2.

	Rychlost výpočtu [s]
MATLAB	0.030
GNU Octave	0.010
SciLab	0.0086
FreeMat	0.34
Dymola	0.010
OpenModelica	0.014
jModelica	0.11
Scipy	nenaměřeno

Tabulka 5.1: Průměrná rychlost výpočtu v sekundách pro model 1. Zaokrouhleno na dvě platná desetinná místa.

	Rychlost výpočtu [s]
MATLAB	1.65
GNU Octave	0.0040
SciLab	0.0045
FreeMat	0.077
Dymola	0.010[1]
OpenModelica	0.014
jModelica	0.0065
Scipy	0.065

Tabulka 5.2: Průměrná rychlost výpočtu v sekundách pro model 2. Zaokrouhleno na dvě platná desetinná místa.

5.2 Model kruhového testu

Druhým modelem v pořadí, který byl implementován, byl jednoduchý kruhový test popsaný v kapitole 4.2. Výsledky měření výpočtu času tohoto modelu jsou k dispozici v tabulce 5.2. Jak je patrné z výsledků rychlosti výpočtu v tabulce 5.2, rozdíly napříč jednotlivými nástroji jsou velké. Pokud se například podíváme na nástroje napsané v jazyce Modelica (Dymola, OpenModelica, jModelica), je rozdíl v rychlosti mezi jModelica a zbytkem nástrojů o prakticky řádový. Zde je nunto podoktnout, že nástroj Dymola pro původní délku testu vypisoval strojový čas 0s. Po delším zkoumání bylo zjištěno, že Dymola čas zaokrouhluje na setiny vteřin, proto u takto malých výpočtů nebylo možné dosáhnout validního výsledku měření. Toto zaokrouhlení navíc provádí až po nějaké době výpočtu – v našem případě to bylo po více než dvojnásobném simulačním čase. Pokud by nástroj byl schopen dodávat validní výslekdy, byl by bez pochyb nejrychlejší z nástrojů napsaných pro jazyk Modelica a možná i nejrychlejším z výběru nástrojů.

V případě času výpočtu nástroje MATLAB může zaujmout, že výpočet zabral tolik procesorového času. Tato velká časová odlišnost je způsobena využitou metodou `dsolve()`, která je v dokumentaci doporučována pro řešení soustav diferenciálních rovnic [17]. U jiných nástrojů tato metoda nebyla použita, proto jsou časy daleko nižší. U nástrojů SciLab a Octave byla využita metoda `ode()`, která je používána pro výpočet jednotlivých diferenciálních rovnic. V implementaci pro nástroj FreeMat byla použita metoda `ode45()`, protože je to jediná metoda, která je v tomto nástroji dostupná pro řešení diferenciálních rovnic.

[1] v čase 22.76s

	Rychlost výpočtu [s]
MATLAB	0.35
GNU Octave	0.0022
SciLab	0.26
FreeMat	0.12
Dymola	0.010[2]
OpenModelica	0.015
jModelica	0.0052
Scipy	0.00087

Tabulka 5.3: Průměrná rychlost výpočtu v sekundách pro model 3. Zaokrouhleno na dvě platná desetinná místa.

5.3 Model tuhého systému

Posledním měřeným modelem byl tuhý systém taktéž popsaný v kapitole 4.2. V tomto případě bylo rozpětí výsledků měření nejmenší napříč všemi nástroji. V tomto případě mohou být výsledky poněkud zkreslené různými numerickými metodami, jelikož žádný z nástrojů, vyjma nástrojů napsaných pro jazyk Modelica, nepodporují stejné metody pro řešení tuhých diferenciálních rovnic, a proto byly vybrány metody doporučované oficiálními dokumentacemi. Výsledky jsou dostupné v tabulce 5.3.

U tohoto modelu bych chtěl vyzdvihnout kvality nástroje Dymola, kterému výpočet zabral velmi malý procesorový čas. Výpočet zabral pouhých 0.010, avšak tento čas byl detekován až v čase simulace 258 sekund. Pokud bychom přepočítali výsledek na čas simulace $T_{max} = 10$s, pak by rychlost výpočtu byla přibližně 0,00038 sekund. Stejně dobře, i když řádově pomaleji, si vedly i další nástroje zpracovávající modely v jazyce Modelica.

Pro nástroj MATLAB byla využita metoda `ode15s()`, která je podle dokumentace [17] označována za nejpřesnější metodu pro řešení tuhých systémů. Naopak v implementaci pro SciLab byla využita metoda `ode()` s parametrem `"stiff"` a následně volá metodu `lsode()`, která je schopna řešit diferenciální rovnice tuhých systémů. V rámci implementace pro nástroj FreeMat byla i zde použita jediná metoda, pomocí které je FreeMat schopen řešit diferenciální rovnice, a tou metodou byla `ode45()`. Podobně jako u implementace pro nástroj SciLab, byla pro Octave využita metoda `lsode()`, která byla volána s parametrem `"stiff"`.

5.4 Shrnutí výsledků experimentů a porovnání nástrojů

V této kapitole se nachází konečné srovnání volně dostupných simulačních nástrojů podle vytvořené metodiky, která je popsána v kapitole 4.1. Je zde popsáno porovnání instalací, hardwarových a softwarových požadavků, délka strojového času během výpočtu a hodnocení obtížnosti implementace jednotlivých modelů. Dále tato kapitola obsahuje závěrečná doporučení vycházející ze získaných poznatků vyhodnocených podle zvolené metodiky.

[2] v čase 258s

Instalace

Pro účely naplnění tohoto kritéria byly všechny nástroje bez problému nainstalovány podle návodu uvedeném v oficiální dokumentaci na běžných operačních systémech (Windows, Linux, MacOS), které by zvolené nástroje měly podporovat podle návodu. Přehled dostupnosti nástrojů pro jednotlivé operační systémy je obsažen v tabulce 3.1.

Hardwarové a softwarové požadavky

Během práce se zvolenými simulačními nástroji nebyly pozorovány téměř žádná omezení. Jediným větším omezením byla občasná nestabilita nástroje SciLab na operačním systému MacOS. Tato nestabilita byla zjištěna během odlaďování naimplementovaných modelů a ve chvílích, kdy během překladu došlo k závažnější chybě, se program ukončil s chybovou hláškou.

Obtížnost implementace modelů

Obtížnost modelů velmi úzce souvisí s kompatibilitou stejného jazyka napříč různými nástroji [25], vybaveností nástroje programovacími knihovnami a kvalitou oficiální dokumentace. V rámci tohoto kritéria bych chtěl vyzdvihnout jednoduchost a hlavně přenositelnost skritpů v jazyce Modelica mezi nástroji Dymola, OpenModelica a jModelica. Implementované modely byly v tomto případě plně kompatibilní. Obtížnost implementace pro nástroje využívající jazyk Modelica hodnotím jako velmi snadnou a vhodnou pro tvorbu modelů.

Implementace modelů v jazyku MATLAB pro nástroje MATLAB, Octave, SciLab a FreeMat se ukázaly jako nejproblematičtější ze všech implementací, především díky nekompatibilitě nástrojů a velkým rozdílům ve vestavěných funkcích jednotlivých nástrojů. Nejlépe si z této čtveřice vedly nástroje MATLAB a SciLab, které mají velmi přehledně zpracovanou dokumentaci. Naopak velmi propadl nástroj FreeMat, převážně kvůli malé podpoře výpočtu diferenciálních rovnic, neboť FreeMat má k výpočtu dispozici pouze jednu metodu, a dále pak díky nemožnosti definovat funkce v souboru se zpracovávaným skriptem.

Programovací jazyk Python s rozšířením SciPy se prokázal jako velmi schopný nástroj pro modelování systémů. Díky detailně zpracované dokumentaci a velké uživatelské základně nečinilo větší problémy implementovat modely v tomto jazyce. Jediným problémem byl model skákajícího míčku. Během implementace bylo zjištěno, že Python nedokáže vhodně zpracovat nespojitosti v místě odrazu míčku a proto nebyl tento model naimplementován. V tomto případě by bylo asi vhodnější upustit od popisu modelu diferenciálními rovnicemi a uchýlit se spíše k analytickým výpočtům. Nicméně i přes tento problém hodnotím implementaci v jazyce Python jako snadnou.

Strojový čas

Během výpočtů s implementovanými modely dosahovaly nástroje velmi podobných hodnot, zvláště pak pokud bylo možné implementovat výpočet pomocí stejných metod. Přehledy dob výpočtů lze nalézt v tabulkách 5.1, 5.2 a 5.3. Zde je nutno znovu podotknout, že měření bylo zatíženo velkým statistickým šumem a také omezeností podporovaných funkcí jednotlivých nástrojů.

U nástroje Freemat byl zjištěn postupný nárůst měřeného čas s každou novou iterací měření. Můžeme tedy předpokládat, že pokud bychom měření opakovali v řádech tisíců a více opakování, pak by tento nástroj byl ve výpočtu nejpomalejší. Naopak nejrychlejším

nástrojem se stal Dymola, který podle naměřených časů naprosto překonal všechny ostatní simulační nástroje.

Závěrečná doporučení

Ze závěrů zjištěných během studia simulačních systémů, implementace modelů a provádění experimentů s nimi, bych celkově doporučil pro tvorbu modelů využívat nástroje podporující modelovací jazyk Modelica, zejména pro jeho jednoduchost a rychlost. Volba nástroje pak záleží na dostupnosti pro koncového uživatele - pokud má přístup k nástroji Dymola, pak bych volil ten, jinak bych doporučil práci v programu OpenModelica. Tyto nástroje bych doporučil pro tvorbu libovolně velkého modelu.

Z nástrojů pro jazyk MATLAB doporučuji právě nástroj MATLAB, avšak je zde nutno příhlížet k tomu, že MATLAB není primárně modelovací nástroj, proto tvorba modelu v něm může být někdy až zbytečně složitá. MATLAB doporučuji kvůli detailní podpoře, výborně zpracované dokumentaci s příklady a velké uživatelské základně. Pokud bych měl volit volně dostupný nástroj, pak by jím byl SciLab, který je velmi podobný nástroji MATLAB, případně i GNU Octave. Naopak bych pro modelování v jazyku MATLAB výrazně nedoporučil nástroj FreeMat, který nemá dostatečnou podporu pro práci s diferenciálními rovnicemi a při měření strojového času se u něj vyskytovala postupná kumulativní prodleva ve výpočtu.

Jazyk Python s rozšířením SciPy se ukázal jako silný nástroj pro modelování menších systémů. Tento nástroj bych rád doporučil zejména pro využití ve výuce, kdy díky rozšířenosti jazyka Python a zpracované dokumentaci, je velmi snadné koncovým uživatelem (studentem) pochopit implementaci modelu a případně ji rozšířit.

Kapitola 6

Závěr

Cílem této práce bylo získat povědomí o simulačních nástrojích, jak komerčních, tak i volně dostupných, analyzovat je a vytvořit reprezentativní sadu nástrojů vhodných k porovnávání. Pro tyto účely bylo zkoumáno více než 12 simulačních nástrojů, z nichž bylo následně vybráno 8 tak, aby měly společné průniky v přijímaných jazycích - MATLAB, Modelica, Python.

Pro tyto nástroje byly navrženy jednoduché modely, které sloužily k následnému porovnání nástrojů. Předlohy modelů byly za účelem validity vybrány z oficiálních dokumentací, případně z literatury zabývající se daným tématem a byly postupně implementovány pro všech 8 zvolených nástrojů. Z důvodu rozsahu práce se práce věnuje pouze spojitým a kombinovaným modelům.

Během implementace a následných experimentů byla potvrzena nekompatibilita mezi jednotlivými nástroji používajícími jazyk MATLAB, především u metod pro řešení diferenciálních rovnic. Naproti tomu bych chtěl vyzdvihnout jednoduchost práce s jazykem Modelica, který z porovnání vyšel nejlépe, jak z hlediska měření času výpočtu, tak i jednoduchostí implementace vybraných modelů. Jazyk Python, konkrétně tedy rozšíření SciPy, byl do porovnání zařazen z důvodu velké rozšířenosti napříč různými obory. I přes problémy se ukázal jako schopný a jednoduchý nástroj pro modelování a simulaci menších systémů. V případě složitějších systémů bych raději volil nástroj, který zpracovává jazyk Modelica.

Doporučujícím výstupem z této práce je tedy používat pro tvorbu modelů jazyk Modelica v rámci kteréhokoliv podporovaného nástroje, zejména díky jednoduchosti ve tvorbě modelů, velké uživatelské komunitě a dostupné dokumentaci. Osobně bych volil nástroj OpenModelica, který mi přišel ze všech nejvíce uživatelsky přívětivý.

Možným rozšířením této práce by byl hlubší průzkum možností využití simulačních nástrojů, především pro jazyk Modelica, který z konečného porovnání vyšel jako nejlepší. Pro tyto účely by bylo potřeba rozšířit oblast průzkumu také na komerční nástroje, například o nástroj Wolfram SystemModeler, nebo SimulationX. Dalším možným rozšířením by bylo srovnání nástrojů pro modelování diskrétních systémů, následná implementace vybraných diskrétních modelů a vyhodnocení výsledků podle zvolené metodiky. K tomuto rozšíření by bylo vhodné rozšířit sadu zvolených nástrojů, například o nástroje určené pro tvorbu Petriho sítí.

Literatura

[1] *Graphical Editing in JModelica.org*. Diplomová práce, Lund University, Lund, 2012. URL http://sam.cs.lth.se/ExjobGetFile?id=471

[2] Alfaro-Bou, E.; Hayduk, R.; Thomson, R.; aj.: Simulation of aircraft crash and its validation. 1975.

[3] Belanger, J.; Venne, P.; Paquin, J.-N.: The what, where and why of real-time simulation. *Planet Rt*, ročník 1, č. 1, 2010: s. 25–29.

[4] Cellier, F. E.; Kofman, E.: *Continuous system simulation*. New York: Springer, c2006, ISBN 978-0-387-26102-7.

[5] Cohen, E. R.; Feinglass, J.; Barsuk, J. H.; aj.: Cost savings from reduced catheter-related bloodstream infection after simulation-based education for residents in a medical intensive care unit. *Simulation in healthcare*, ročník 5, č. 2, 2010: s. 98–102.

[6] E, U. G.: Comparison of Scilab syntax and Functions to Matlab. infocleaninghouse.com, 2001, [Online; citováno 26. 4. 2019]. URL https://agenda.ct.infn.it/event/319/material/6/0.pdf

[7] Fragile, P. C.; Blaes, O. M.; Anninos, P.; aj.: Global General Relativistic Magnetohydrodynamic Simulation of a Tilted Black Hole Accretion Disk. *The Astrophysical Journal*, ročník 668, č. 1, oct 2007: s. 417–429, doi:10.1086/521092. URL https://doi.org/10.1086%2F521092

[8] Fritzson, P.; Bunus, P.: Modelica – A General Object-Oriented Language for Continuous and Discrete-Event System Modeling. In *IN PROCEEDINGS OF THE 35TH ANNUAL SIMULATION SYMPOSIUM*, 2002, s. 14–18.

[9] Hairer, E.; Wanner, G.: *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, ročník 14. Springer-Verlag Berlin Heidelberg, 01 1996, doi:10.1007/978-3-662-09947-6.

[10] James, O.; von Tunzelmann, E.; Franklin, P.; aj.: Gravitational lensing by spinning black holes in astrophysics, and in the movieInterstellar. *Classical and Quantum Gravity*, ročník 32, č. 6, feb 2015: str. 065001, doi:10.1088/0264-9381/32/6/065001. URL https://doi.org/10.1088%2F0264-9381%2F32%2F6%2F065001

[11] jModelica: jModelica. 2019, [Online; citováno 16. května 2019]. URL https://jmodelica.org/

[12] John W. Eaton, S. H., David Bateman; Wehbring, R.: *GNU Octave version 5.0.1 manual: a high-level interactive language for numerical computations*. CreateSpace Independent Publishing Platform, 2018, ISBN 1441413006.
URL http://www.gnu.org/software/octave/doc/interpreter

[13] Jones, E.; Oliphant, T.; Peterson, P.; aj.: SciPy: Open source scientific tools for Python. 2001–2019, [Online; citováno 16. května 2019].
URL http://www.scipy.org/

[14] Jones, S. L.; Sullivan, A. J.; Cheekoti, N.; aj.: Traffic simulation software comparison study. *UTCA report*, ročník 2217, 2004.
URL http://utca.eng.ua.edu/files/2011/08/02217fnl.pdf

[15] Kirk, M.: *Thoughtful Machine Learning with Python: A Test-driven Approach*. O'Reilly, 2017, ISBN 9781491924136.
URL https://books.google.cz/books?id=DCd4rgEACAAJ

[16] Kyriakis, D.: FreeMat. 2013, online; citováno 12. 5. 2019.
URL http://freemat.sourceforge.net/

[17] MATLAB: *version 9.6.0 (R2019a)*. Natick, Massachusetts: The MathWorks Inc., 2019.

[18] Negrão, C. O.; Hermes, C. J.: Energy and cost savings in household refrigerating appliances: A simulation-based design approach. *Applied Energy*, ročník 88, č. 9, 2011: s. 3051 – 3060, ISSN 0306-2619, doi:https://doi.org/10.1016/j.apenergy.2011.03.013.
URL http://www.sciencedirect.com/science/article/pii/S030626191100170X

[19] Nunez-Iglesias, J.; van der Walt, S.; Dashnow, H.: *Elegant SciPy: The Art of Scientific Python*. O'Reilly Media, Inc., první vydání, 2017, ISBN 1491922877, 9781491922873.

[20] Pawlewski, P.: Multimodal Approach to Modeling of Manufacturing Processes. 12 2014, doi:10.1016/j.procir.2014.01.130.
URL https://www.researchgate.net/publication/270980703_Multimodal_Approach_to_Modeling_of_Manufacturing_Processes

[21] Peringer, P.: *Modelování a simulace - studijní opora. [Online; citováno 7. 4. 2019].*
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php/course/IMS-IT/texts/opora-ims.pdf

[22] Roa, L.; Prado-Velasco, M.: Simulation Languages*, ročník 8. Wiley-Interscience, 04 2006, ISBN 9780471740360, doi:10.1002/9780471740360.ebs1089.*

[23] Schaller, R.: Moore's law. IEEE Spectrum*, ročník 34, č. 6, 1997: s. 52–59, ISSN 0018-9235, doi:10.1109/6.591665, [Online; citováno 19. 4. 2019].*
URL http://ieeexplore.ieee.org/document/591665/

[24] SciLab: SciLab. 2019, [Online; citováno 16. května 2019].
URL https://www.scilab.org/

[25] Sharma, N.; Gobbert, M. K.: A comparative evaluation of Matlab, Octave, FreeMat, and Scilab for research and teaching. UMBC Faculty Collection*, 2010.*

[26] *Tiller, M. M.: Modelica by Example. [Online; citováno 21. 4. 2019]. URL* http://book.xogeny.com/

[27] *Zapletal, M.:* Implementace a testování vybraných optimalizačních metod pro úlohy odhadu parametrů simulačních modelů. *Diplomová práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, Brno, 2016.*

Příloha A

Obsah přiloženého CD

CD přiložené k bakalářské práci obsahuje tyto položky:

- `/models` – složka obsahující implementace pro jednotlivé simulační nástroje
- `/text` – složka obsahující zdrojové testy potřebné pro přeložení a vygenerování textu bakalářské práce
- `README.txt` – soubor s informacemi o instalaci nástrojů a spuštění modelů v těchto nástrojích
- `xvyslo05_BP.pdf` – výsledné znění textu bakalářské práce