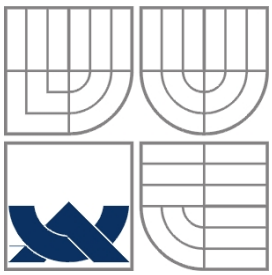


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

REALIZACE JEDNODUCHÉHO DYNAMICKÉHO MODELU POMOCÍ ODE

SIMPLE DYNAMIC MODEL REALIZATION BASED ON ODE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETR ZETKA**
AUTHOR

VEDOUCÍ PRÁCE **ING. STANISLAV VĚCHET, PH.D.**
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2007/08

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Zetka Petr

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

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:

Realizace jednoduchého dynamického modelu pomocí ODE

v anglickém jazyce:

Simple dynamic model realization based on ODE

Stručná charakteristika problematiky úkolu:

Cílem práce je návrh a realizace dynamického modelu v prostředí Open-Dynamic-Engine (ODE). Bude realizován model jednoduché dynamické soustavy. Stejným model bude poté realizován v jiném prostředí. Takto navržený a realizovaný model v různých prostředí poslouží ve srovnávacích experimentech pro porovnání vlastností obou simulačních prostředí a ke zhodnocení vlastností ODE.

Cíle bakalářské práce:

Prostudujte problematiku simulací pomocí ODE
Navrhněte vhodnou dynamickou soustavu
Tuto soustavu realizujte pomocí ODE a také pomocí dalšího vybraného simulačního prostředí
Sestavte množinu testovacích dat a proveďte srovnání obou simulačních prostředí

Seznam odborné literatury:

Kosei D.: Robot programming with Open Dynamics Engine, Morikita Publishing Co, Ltd., Tokyo, 2007, ISBN:978-4627846913

www.ode.org

Vedoucí bakalářské práce: Ing. Stanislav Věchet, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2007/08.

V Brně, dne 27. 02. 2008

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT

Šeda

doc. RNDr. Ing. Miloš Šeda, Ph.D.
Ředitel ústavu

V.z. R. Doupovec

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Petr Zetka

Bytem: Cihlářská 18, Brno 602 00

Narozen/a (datum a místo): 24.5.1986, Brno

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta strojního inženýrství

se sídlem Technická 2896/2, 616 69 Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

….....

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: Realizace jednoduchého dynamického modelu pomocí ODE

Vedoucí/ školitel VŠKP: Ing. Stanislav Věchet, Ph.D.

Ústav: Ústav automatizace a informatiky

Datum obhajoby VŠKP:

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů ……..
- □ elektronické formě – počet exemplářů ……..

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: ……..

…….. ……
Nabyvatel Autor

ABSTRAKT

Cílem práce je návrh a realizace modelu dynamické soustavy pomocí knihovny Open Dynamic Engine a v dalším simulačním prostředí. Po provedení praktických experimentů sestavit množinu testovacích dat. Porovnat výsledky experimentů a zhodnotit vlastnosti ODE.

ABSTRACT

The aim of presented work is the design and realization of a model of dynamic system with the usage of Open Dynamic Engine library with comparison to another simulation tool. The set of experimental data was created after practical experiments. Finally, the ODE analysis was made.

KLÍČOVÁ SLOVA

Simulace, Open Dynamic Engine, Matlab Simulink.

KEYWORDS

Simulation, Open Dynamic Engine, Matlab Simulink.

PODĚKOVÁNÍ

Děkuji Ing. Stanislavu Věchetovi, Ph.D. za odborné vedení, cenné připomínky a rady, potřebné pro vypracování Bakalářské práce.

Obsah:

1 ÚVOD

Cílem práce bylo navrhnout a realizovat model dynamické soustavy s použitím ODE a simulačního nástroje Matlab Simulink, s realizovanými modely provést praktické experimenty a sestavit množinu testovacích dat. Porovnat výsledky praktických experimentů a zjistit vhodnost použití ODE pro další práci.

Pro provedení této práce bylo nutné prostudovat problematiku simulací pomocí ODE. ODE (Open Dynamics Engine) je knihovna pro simulování dynamiky tuhých těles.

Nejprve byla zvolena vhodná dynamická soustava pro demonstrování funkčnosti ODE. Zvolenou soustavou je dvoukolový dopravní prostředek. Model této soustavy byl realizován pomocí ODE. Při realizaci modelu pomocí ODE je nezbytnou součástí nastavení parametrů prostředí pro reálné chování modelu. Stejná soustavu byla realizována i v Matlabu. Po vytvoření modelu dynamické soustavy byl navržen regulátor a začleněn do modelu soustavy.

Po zhotovení modelů byly provedeny praktické experimenty. V experimentech byla sledována schopnost regulátoru regulovat soustavu po vyvedení z rovnovážné polohy. Vychýlení z rovnovážné polohy bylo prováděno v ODE a v Matlabu nakloněním těla soustavy. Experimenty byly provedeny pro různě nastavené parametry regulátorů a kroky simulace. V rámci experimentů byly vytvořeny grafy: naklonění těla soustavy v závislosti na čase, akční moment v závislosti na čase a posun soustavy v závislosti na čase. Úkolem experimentů bylo sestavit množinu testovacích dat, pomocí které budeme srovnávat ODE a Matlab.

Při porovnání výsledků experimentů byla porovnávána kvalita regulace, vyjádřená kvadratickým kritériem kvality regulace.

Na závěr práce byla zhodnocena použitelnost ODE pro simulování dynamických systémů.

2 CÍLE PRÁCE

Hlavním cílem práce bylo sestavení dynamického modelu nestabilní soustavy pomocí ODE a simulačního prostředí Matlab, dále tyto modely vzájemně porovnat. Celkově lze tento cíl rozdělit na několik cílů dílčích a to především:

Prvním cílem bylo prostudovat problematiku simulací pomocí ODE, především vytváření modelu. Seznámit se s programovacím jazykem python. Dále se seznámit s používáním programu Matlab, zejména nadstavby Matlabu Simulink pro vytváření modelu a simulaci.

Druhým cílem bylo navrhnout vhodný model dynamické soustavy pro demonstrování funkčnosti ODE. Dynamickou soustavou je dvoukolový dopravní prostředek. Tuto soustavu popsat dynamickými rovnicemi. Z dynamických rovnic vytvořit matematický popis soustavy, který je potřeba pro realizaci simulačního modelu.

Dalším cílem bylo realizovat model soustavy pomocí ODE a s použitím simulačního prostředí Matlab. Při realizaci v ODE vhodně nastavit parametry prostředí pro reálné chování modelu. V Matlabu realizovat model pomocí funkčních bloků. Po realizaci soustavy navrhnout a implementovat regulátor.

Posledním cílem bylo provést praktické experimenty a sestavit množinu testovacích dat pro ODE a Matlab. Pomocí grafů porovnat obě simulační prostředí a zhodnotit použitelnost ODE pro simulování dynamických soustav.

3 MOŽNÉ PŘÍSTUPY K REALIZACI SIMULACE

Poslední dobou vznikla potřeba sledovat chování soustavy se známým matematickým popisem, toto s příchodem výkonnějších počítačů dalo vzniknout počítačové simulaci jako samostatné disciplíny [1].

Simulace je napodobení procesu nebo objektu (obecně systému) pomocí matematického popisu (modelu). Simulace umožňuje pomocí změny vstupních nebo jiných podmínek zkoumat změny a varianty chování systému a předpovídat tak jeho reálnou činnost. Počítač je ideálním prostředkem pro provádění simulací vzhledem k tomu, že simulování reálných dějů vyžaduje díky množství ovlivňujících faktorů a jejich neurčitosti obrovskou výpočetní sílu [2].

Simulace má tedy za cíl získat nové poznatky, které z nejrůznějších důvodů nejsme schopni získat přímo, zkoumáním původního systému či experimentováním s ním[3].

Z uvedených informací se nabízí otázka: jak daleko se odlišuje chování vytvořeného simulačního modelu od chování skutečné soustavy [1]. Tato otázka úzce souvisí s tím, čím se v této práci zabýváme. V práci budeme zkoumat odlišnost výsledků vybrané fyzikální knihovny od jiného zvoleného simulačního nástroje (u kterého už víme s jakou „přesností“ pracuje).

Dnes je možné simulovat prakticky cokoliv, co umíme popsat matematickými rovnicemi. Modelování patří k tradičním postupům v některých technických disciplínách, například v kybernetice nebo v teorii automatického řízení. S rozvojem levných a dostupných počítačů v posledních desetiletích pronikla simulace a počítačové modelování do většiny technických věd a stala se důležitou součástí i v biologii, meteorologii, geologii a dokonce v ekonomii a ve vědách sociálních[4].

Při výběru nástroje pro simulaci můžeme volit mezi fyzikálním engine (viz. 3.1) a simulačním nástrojem (viz. 3.4). Pokud zvolíme fyzikální engine, potřebujeme grafický engine (viz. 3.2) pro vykreslení modelu a dále znalost programovacího jazyka (viz. kap. 3.3) pro práci s knihovnami. Obsluha fyzikálního engine je sice náročnější, avšak většinou jsou pro nekomerční a někdy i komerční účely zdarma.

Simulační nástroje naproti tomu obsahují uživatelské prostředí, jehož obsluha je jednoduchá a rychle naučitelná.

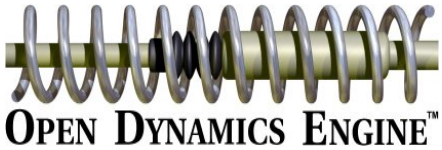
3.1 Fyzikální engine

ODE je knihovna pro fyzikální simulaci, která byla v této práci použita. Pro srovnání zde uvedu další použitelné knihovny pro tuto práci.

3.1.1 Open Dynamics Engine

ODE je open source, vysoce výkonná knihovna pro simulování dynamiky tuhých těles. Je to platforma se snadno použitelným C/C++ API. Obsahuje rozvinuté typy vazeb a integrovanou detekci kolizí s třením. ODE je plně použitelná pro simulování vozidel, objektů ve virtuálním prostředí a virtuální zvířata. Je v současné době používána v mnoha počítačových hrách, 3D vývojových nástrojích a simulačních nástrojích[10].

Výhodou ODE je použitelnost na nejběžnějších operačních systémech Linux, Mac, Windows a kompatibilita s OpenGL. Je volně ke stažení s dostupným zdrojovým kódem. Nevýhodou je, že ODE neobsahuje uživatelské prostředí a nezabývá se vizualizací.



Obr. 1 Logo ODE[10].

3.1.2 Havok physics

Havok Physics poskytuje komplexní SDK podporu pro oblasti: detekce kolizí, MOPP™

technology, dynamické a omezené řešení, dynamiku vozidel, převod dat a nástroje pro podporu vizualizace, vizuální ladění pro diagnostiku ve hrách[11].

Výhodou firmy Havok je, že kromě Havok physics nabízí také produkty zabývající se vizualizací a animací. Od koupi společností Intel je přístupný bez licence pro nekomerční účely. Nevýhodou je, že nejsou přístupné zdrojové kódy.

Obr. 2 Logo Havok[11].

3.1.3 Newton Game Dynamics

Newton Game Dynamics má integrované řešení pro real time simulace fyziky prostředí. API poskytuje vytváření scén, detekci kolizí, dynamické reakce. Newton Game Dynamics není jen pro tvorbu her, ale je také použitelný pro různé real-time physics simulace [12].

Výhodou je použitelnost na operačních systémech Linux, Mac, Windows. Nevýhodou je, že nelze použít bez licence a nejsou přístupné zdrojové kódy.

Obr. 3 Logo Newton game dynamics [12].

3.1.4 AGEIA PhysX

AGEIA PhysX poskytuje všechny základní funkce: detekci kolizí, použití vazeb. Na rozdíl od ostatních engine využívá pro zvýšení výkonu HW akcelerace fyziky pomocí speciální karty s procesorem PPU (Physics Processing Unit)[13].

Výhodou je, že po odkoupení firmou NVidia je volně ke stažení. Nevýhodou je, že zdrojové kódy jsou přístupné pouze pod licencí.

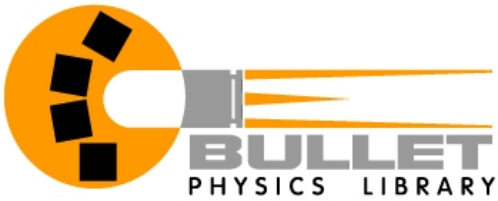
Obr. 4 Logo AGEIA PhysX[13].

3.1.5 Bullet Physics Library

Bullet je profesionální open source knihovna k výpočtu dynamiky tuhých těles s integrovanou detekcí kolizí a multi-threaded 3D. Knihovna je používána různými profesionálními vývojáři her na PC a herních konzolí[14]. Bullet je také součástí programu Blender.

Výhodou je, že je volně ke stažení pro nekomerční použití a pro komerční použití pod Zlib licencí. Podporuje COLLADA Physics formát souborů a může být použita na nejčastějších operačních

systémech Linux, Mac, Windows. Nevýhodou je nedostupnost zdrojových kódů a nepříliš obsáhlá dokumentace.



Obr. 5 Logo Bullet Physics library[14].

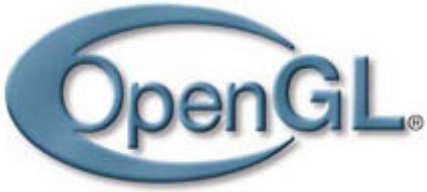
3.2 Grafický engine

Zvolení fyzikální knihovny v této práci vyžaduje použití doplňkových knihoven např.: pro vykreslení modelu potřebujeme grafickou knihovnu. Nejčastěji používanými grafickými engine jsou OpenGL a DirectX. Bližší popis knihoven je v následujících kapitolách.

3.2.1 OpenGL

OpenGL(Open Graphics Library) je průmyslový standard specifikující multiplatformní rozhraní (API) pro tvorbu aplikací počítačové grafiky. Používá se při tvorbě počítačových her, CAD programů, aplikací virtuální reality či vědeckotechnické vizualizace apod. OpenGL je použitelné pod různými operačními systémy[5].

Výhodou je použitelnost na nejběžnějších operačních systémech Linux, Mac, Windows a kompatibilita s ODE. Za nevýhodu lze považovat, že neobsahuje uživatelské prostředí a realizace simulačního modelu se vytváří psaním programového kódu.



Obr. 6 Logo OpenGL[15].

3.2.2 DirectX

DirectX je programátorská knihovna obsahující nástroje pro tvorbu grafiky v počítačových hrách a dalších multimediálních aplikacích, vytvořená firmou Microsoft pro použití pod operačním systémem Windows[6].

Výhodou může být, že je stále používanější než OpenGL. Nevýhodou je podpora pouze operačního systému Windows a stejně jako OpenGL neobsahuje uživatelské prostředí.

Obr. 7 Logo DirectX 10[16].

3.3 Programovací jazyky

Při použití fyzikálního engine ODE se nabízejí dvě základní varianty programovacích jazyků, kterými jsou Python a C++.

3.3.1 Python

Python je interpretovaný objektově orientovaný programovací jazyk, který se může využít v mnoha oblastech vývoje softwaru. Nabízí významnou podporu k integraci s ostatními jazyky a nástroji a přichází s mnoha standardními knihovnami. Mnoho vývojářů se s jeho používáním cítí produktivnější než s jinými programovacími jazyky[19].

Výhodou pythonu je, že je lehce naučitelný, uživatelsky přívětivý a intuitivní jazyk. Další výhodou je, že Python je multiplatformní jazyk a je součástí základní instalace ve většině distribucí systému Linux. Subjektivní nevýhodou může být, že jeho používání není tolik zaběhlé jako u C++ a odlišnost syntaxe od nejpoužívanějších jazyků.

3.3.2 C++

C++ je objektově orientovaný programovací jazyk, který byl vyvinut rozšířením jazyka C. Podporuje několik programovacích stylů (paradigmat) jako je procedurální programování, objektově orientované programování a generické programování, není tedy jazykem čistě objektovým. V současné době patří C++ mezi nejrozšířenější programovací jazyky[7].

Výhoda C++ je ve velké rozšířenosti jazyka. Subjektivní nevýhodou může být, že není tak intuitivní a lehce naučitelný jako Python.

3.4 Simulační nástroje

Alternativou ke zvolenému fyzikálnímu engine je použití již hotového programu pro simulaci. Nástroj pro simulaci byl v práci použit pro srovnání simulace realizované pomocí fyzikálního engine. Trh obsahuje velké množství simulačních nástrojů, mezi nejznámější patří např.: Matlab Simulink a Autodesk Inventor a tyto nástroje jsou dále více rozebrány.

3.4.1 Matlab Simulink

Matlab je program pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, paralelní výpočty, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. Matlab je nástroj jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací.

Simulink je rozšíření Matlabu pro simulaci a modelování dynamických systémů, který využívá jádra Matlabu pro numerické řešení nelineárních diferenciálních rovnic. Poskytuje uživateli možnost rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat a rovnic[17].

Výhodou Matlabu je, že obsahuje uživatelské prostředí, které je lehce ovladatelné. Matlab byl

implementován na všech významných platformách (Windows, Linux, Solaris, Mac). Nevýhodou je, že je to komerční produkt.

3.4.2 Autodesk Inventor

Sada aplikací Autodesk Inventor Simulation Suite kombinuje snadno použitelnou a pevně integrovanou simulaci pohybu a analýzu namáhání. Aplikace usnadňuje uživatelům validaci digitálních prototypů a předpovědi, jak bude návrh fungovat v reálných podmínkách, dříve než je produkt vyroben. Autodesk Inventor Simulation Suite zahrnuje všechny hlavní funkce aplikace Autodesk Inventor Suite a navíc následující sady funkcí: dynamická simulace, analýza napětí[18].

Výhodou Inventoru je, že obsahuje uživatelské prostředí. Nevýhodou je, že je to komerční produkt a podpora pouze operačního systému Windows.

4 POUŽITÉ PŘÍSTUPY

Z rešerše vyplynulo použití knihovny ODE, používání knihovny bude popsáno v kapitole (viz. 4.1). Důvody pro použití jsou: přístupné zdrojové kódy, aktivní vývojová komunita, mnoha existujícími projekty ověřená funkčnost tohoto engine, v neposlední řadě návaznost na další práce FSI UAI, viz. [23][24].

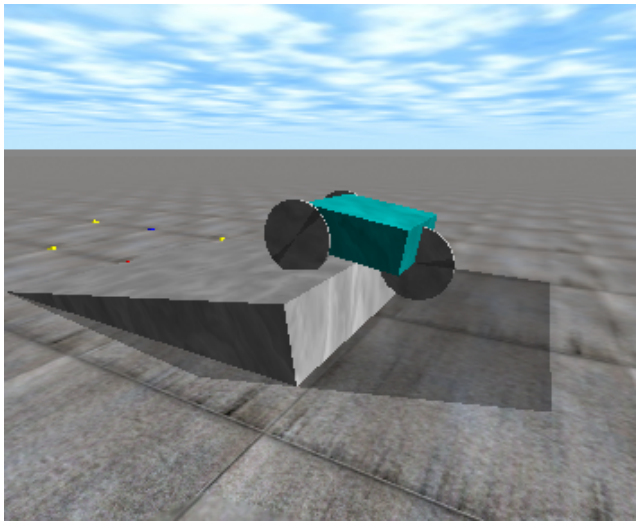
Při výběru grafického engine byly rozhodující: kompatibilita s ODE a práce na různých operačních systémech. Z těchto důvodů byla zvolena knihovna OpenGL. Bližší popis knihovny je v kapitole (viz. 4.2)

Programovací jazyk Python byl vybrán pro snadné používání, rychle naučitelnou syntaxi a také pro návaznost na práce FSI UAI[23][24].

Simulační nástroj Matlab Simulink(viz. kap. 4.4) byl vybrán pro jeho univerzálnost, časté použití v technické praxi a snadné navrhování modelu.

4.1 Použitá knihovna pro dynamickou simulaci ODE

ODE v této práci vypočítává dynamiku těles a detekci kolizí. Obsahuje některá základní tělesa a vazby pro realizaci modelu soustavy. Tento fakt nám velmi usnadní práci při vytváření modelu soustavy, protože namísto programování celého modelu postačí volat metody obsažené v ODE. V této práci budeme používat PyODE. PyODE je fyzikální knihovna, která má metody pro výpočet stejné jako ODE, pouze má některé části upravené tak, aby programovací jazyk pro vytváření modelu byl python.



Obr. 8 Vozidlo vytvořené v ODE[8].

Při vytváření modelu v ODE je nutné nejdříve vytvořit objekt svět a v tomto objektu nastavit konstanty tíhového zrychlení, ERP (error reduction parameter) je parametr, který udává jak velká síla bude působit proti oddělení dvou těles ve vazbě. CFM (constraint force mixing) je parametr, který udává jak měkké nebo tvrdé budou omezení ve vazbách[20].

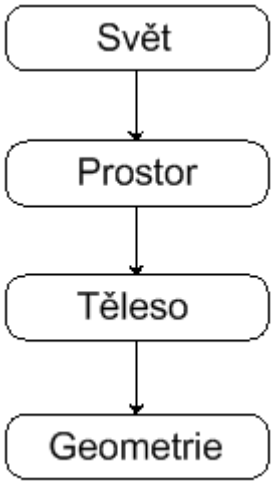
Následně vytvořit prostor ve kterém budou definované kolize a jednotlivá tělesa soustavy, která se přiřadí danému prostoru. Tělesa mají parametry hmotnost případně hustota, ze které se hmotnost vypočítá a geometrický tvar potřebný po detekci kolizí. Dále se vytvořená tělesa propojí vazbami.

Pro simulování vytvořeného modelu je zapotřebí zadat čas a časový krok simulace. Při nastavování kroku simulace musíme pamatovat na to, že malý krok simulace zpřesní výsledky ale zpomalí výpočet, naopak velký krok zrychlí výpočet ale výsledky jsou méně přesné.

V ODE se používá Coulombovský model tření[20]. Tření je možné nastavit jednotné pro

všechny kontaktní vazby, nebo pro každé těleso jiné.

Detekce kolizí se realizuje pomocí funkce near_callback. Při volání funkce near_callback, když dojde ke kolizi dvou těles, vytvoří se mezi tělesy kontaktní vazba(viz. 4.1.2).

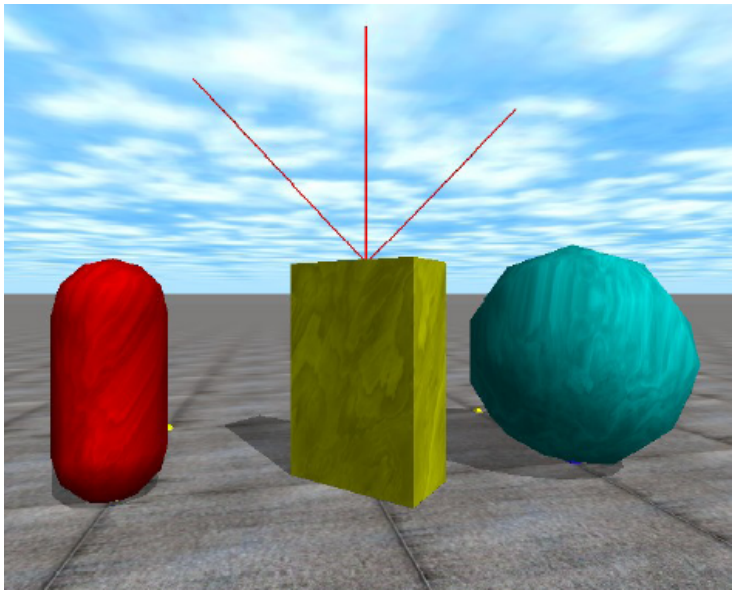


Obr. 9 Struktura v ODE[20] .

- Svět je množina všech statických a dynamických těles.
- Prostor je místo, ve kterém je definována detekce kolizí.
- Těleso je objekt, jehož dynamika se počítá během simulace.
- Geometrie určuje tvar tělesa pro výpočet kolizí.

4.1.1 Geometrie těles

Geometrie rozhoduje při kolizi dvou těles o tvaru a rozměrech těles, která jsou v kontaktu. Nesouvisí s tvarem, který se při vizualizaci vykreslí. V ODE jsou integrované geometrie jako kvádr, kapsle, válec, rovina, paprsek, koule, transform (umožňuje spojit dvě geometrie do sebe), trojúhelníková síť. Kromě druhu geometrie je potřeba zadat rozměry tělesa.



Obr. 10 Druhy geometrie zleva kapsle, kvádr, koule[20].

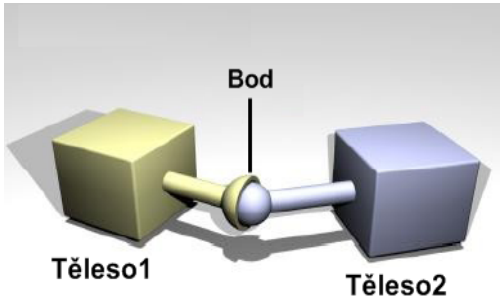
4.1.2 Vazby

S použitím vazeb, které jsou integrované v ODE můžeme propojovat tělesa. ODE obsahuje

základní vazby (pevnou, kontaktní, sférickou, rotační, rotační s dvojitou rotací, posuvnou a univerzální).

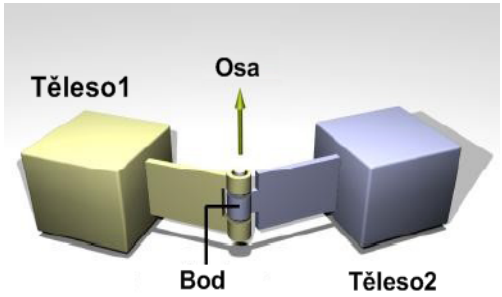
Pevná vazba je jedna z nejpoužívanějších vazeb. Vazba, která přichytí jedno těleso k druhému a neumožňuje posun ani rotaci.

Kontaktní vazba je speciální vazba používaná v ODE pro vytvoření kontaktu mezi dvěma tělesy při kolizi. Sférická vazba umožňuje rotaci okolo tří os, avšak neumožňuje posuv ve vazbě.



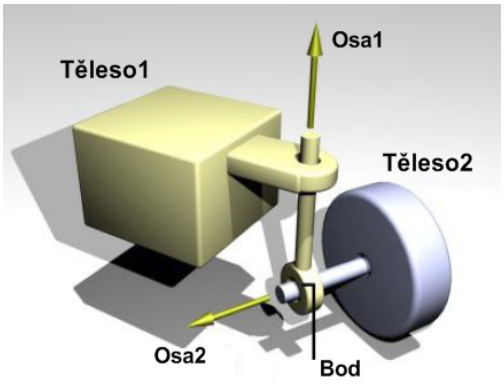
Obr. 11 Sférická vazba[20].

Rotační vazba dovoluje rotaci kolem jedné osy, ale nedovoluje posuv.



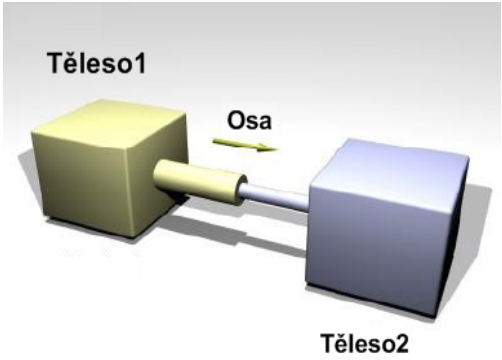
Obr. 12 Rotační vazba[20].

Rotační vazba s dvěma stupni volnosti umožňuje rotaci kolem dvou os, avšak neumožňuje posuv. Vazba se často využívá při vytváření modelu vozidel.



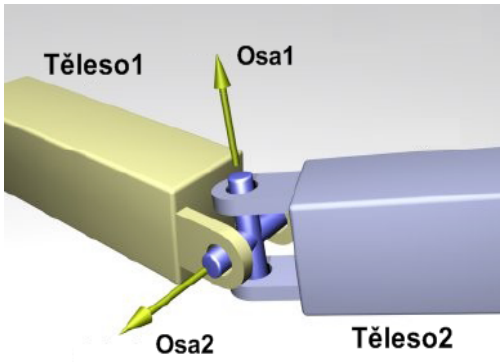
Obr. 13 Rotační vazba s dvěma stupni volnosti[20].

Posuvná vazba umožňuje posuv v jedné ose, ale neumožňuje rotaci.



Obr. 14 Posuvná vazba[20].

Univerzální vazba umožňuje rotaci kolem dvou os.



Obr. 15 Univerzální vazba[20].

4.2 Knihovna pro vizualizaci modelu OpenGL

Knihovnu jsme vybrali pro kompatibilitu s ODE, možnost použití na nejpoužívanějších operačních systémech.

OpenGL (Open Graphics Library) byla navržena firmou SGI (Silicon Graphics Inc.) jako aplikační programové rozhraní API (Application Programming Interface) k akcelerovaným grafickým kartám. Byla navržena tak, aby byla použitelná na různých typech grafických akcelerátorů a aby ji bylo možno použít i v případě, že na určité platformě žádný grafický akcelerátor není nainstalován. V současné době lze knihovnu OpenGL použít na různých verzích unixových systémů (včetně Linuxu a samozřejmě IRIXu), OS/2 a na platformách Microsoft Windows[21].

V OpenGL lze během zadávání příkazů pro vykreslování průběžně měnit vlastnosti vykreslovaných primitiv (barva, průhlednost) nebo celé scény (volba způsobu vykreslování, transformace) a toto nastavení zůstane zachováno do té doby, než ho explicitně změníme. Výhoda tohoto přístupu spočívá především v tom, že funkce pro vykreslování mají menší počet parametrů a že jedním příkazem lze globálně změnit způsob vykreslení celé scény[21].

OpenGL nám umožní vizualizovat model vytvořený v ODE, abychom mohli sledovat jeho chování. Pro naši práci budeme používat knihovnu pyOpenGL, což je OpenGL ve kterém se programuje pomocí jazyka python.

Knihovna GLUT (OpenGL Utility Toolkit) byla vytvořena jako doplněk ke grafické knihovně OpenGL. Definuje a implementuje aplikační rozhraní pro tvorbu oken a jednoduchého grafického uživatelského rozhraní, přičemž je systémově nezávislá. Do knihovny jsou také přidány funkce pro vykreslování bitmapového a vektorového písma v několika základních řezech a vykreslení plných těles (Cube, Sphere, Dodecahedron, Icosahedron, Octahedron, Tetrahedron, Torus, Teapot)[21].

Knihovna GLU (OpenGL Utilities) je nadstavba OpenGL. Umožňuje využívat tesselátory (rozložení nekonvexních polygonů na trojúhelníky), evaluátory (výpočet souřadnic bodů ležících na parametrických plochách), vykreslovat kvadriky (koule, válce, kužely a disky)[21].

4.3 Použitý programovací jazyk Python

Tento programovací jazyk jsme vybrali pro snadné používání, rychle naučitelnou syntaxi a také proto, že s ním pracuje celý tým.

Python navrhl v roce 1990 Guido van Rossum. Vyvíjí se jako open source projekt, který zdarma nabízí instalační balíky pro většinu běžných platforem (Unix, Windows, Mac OS); ve většině distribucí systému Linux je Python součástí základní instalace. Výkon aplikací napsaných v Pythonu je dobrý, protože výkonově kritické knihovny jsou implementovány v jazyce C, s kterým Python výborně spolupracuje. I samotný jazyk je na tom v porovnání s jinými interpretovanými jazyky dobře. Je např. 3 až 5 násobně rychlejší než PHP[9].

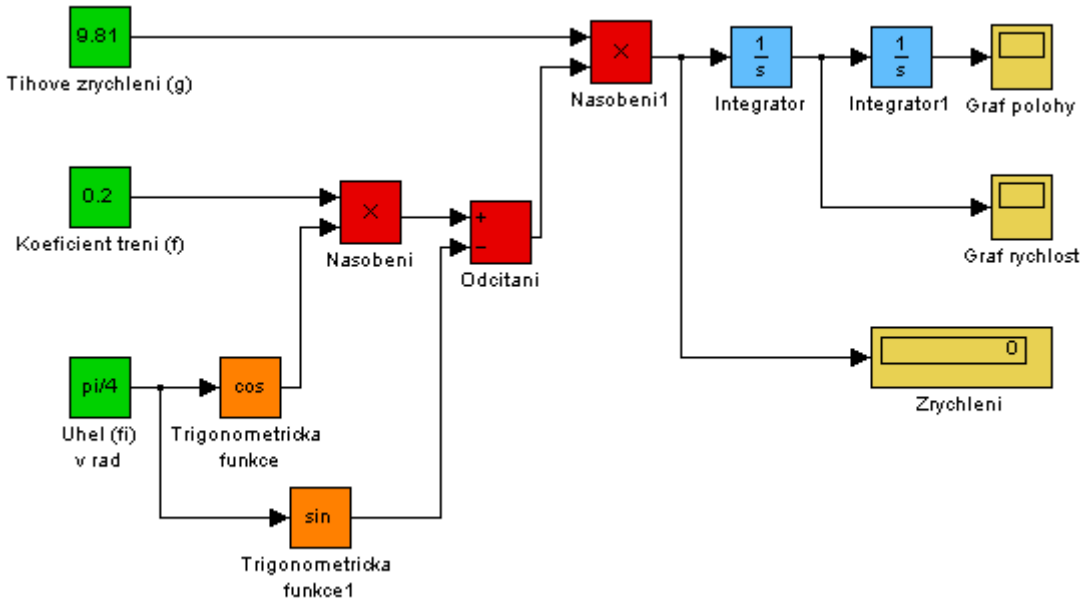
Svou jednoduchostí bývá zařazován mezi skriptovací jazyky, ale je vhodný i pro rozsáhlé aplikace. Python je používán jako programovací jazyk v knihovnách pyODE a pyOpenGL.

Pro tuto práci budeme používat verzi Pythonu 2.5.

4.4 Vybraný simulační nástroj Matlab Simulink

Matlab jsme vybrali pro jeho univerzálnost, časté použití v technické praxi a snadné navrhování modelu.

Pomocí simulinku a jeho grafického editoru lze vytvářet modely lineárních, nelineárních, v čase diskrétních nebo spojitých systémů pouhým přesouváním funkčních bloků. Nový přístup k řešení diferenciálních rovnic dovoluje simulovat i rozsáhlé "stiff" systémy rychle, přesně a s efektivním využitím paměti počítače[17].



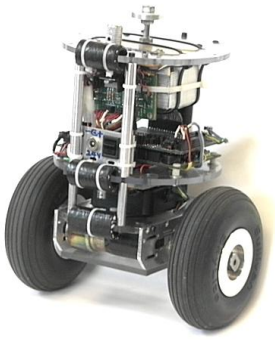
Obr. 16 Model krychle smýkající se po nakloněné rovině vytvořený v Simulinku.

Při práci v simulinku je nutné nastavit: metodu řešení(např.: ode45), krok simulace, čas zahájení a ukončení simulace. Při sestavení modelu vybereme funkční bloky z knihovny (library browser) a propojíme je. V knihovně jsou bloky rozřazené do kategorií (continuous, discrete, function and tables, math, nonlinear, signal and system, sinks, sources).

Spojité bloky jsou pro vytvoření spojitých dynamických modelů. Diskrétní bloky pro

vytvoření diskrétních dynamických modelů. Funkce a tabulky nabízí např.: interpolaci mezi hodnotami tabulkového zadávání průběhů a přepočítá vstupní signál pomocí zadaného polynomu. Matematické bloky jsou pro realizaci algebraické části modelu. Nelineární bloky obsahují typické nelinearity např.: Saturace, Switch, Releová nelinearita. Signálové a systémové (signal and system), bloky ke spojování a změně struktury signálů. Značky jsou bloky pro zpracování výsledků. Zdroje jsou bloky pro zdroje signálů[22].

5 SESTAVENÍ POHYBOVÝCH ROVNIC SOUSTAVY

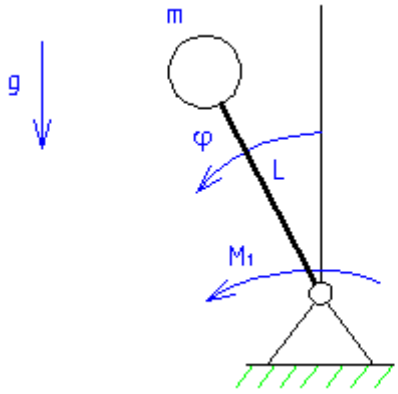


Obr. 17 Balancující robot nBot[27]

Soustava, která byla modelována, byl dvoukolový dopravní prostředek. Existuje řada konkrétních implementací dvoukolových robotů, realizovaných na stejném matematickém základě jako soustava která byla v této práci modelována, např.: balancující robot nBot[27].

Balancující robot nBot byl součástí experimentu řízení inverzního kyvadla. První návrh robota byl tříkolý robot pro vyvažování balonu na tyči. Základní myšlenka vytvoření dvoukolového balancujícího robota spočívá v zavedení řídicího momentu, jehož velikost je regulována dle aktuálního náklonu[27].

Soustava vychází z inverzního kyvadla, přesněji je to inverzní kyvadlo s dvěma stupni volnosti.

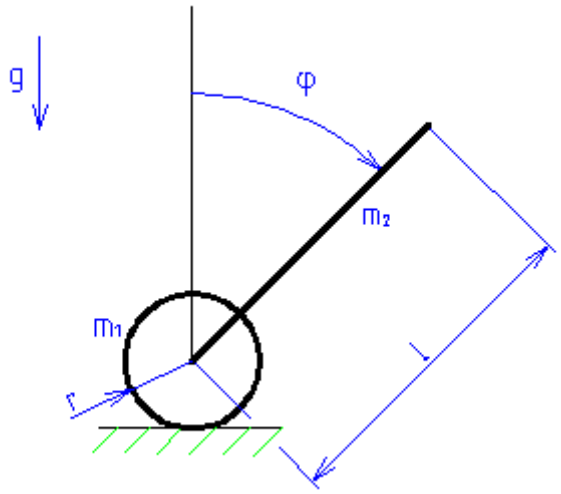


Obr. 18 Inverzní kyvadlo s jedním stupněm volnosti

Kde
m -hmotnost kyvadla
g - tíhové zrychlení
φ - úhel naklonění kyvadla
L - délka kyvadla
M_1 - řídící moment

5.1 Dynamické rovnice soustavy

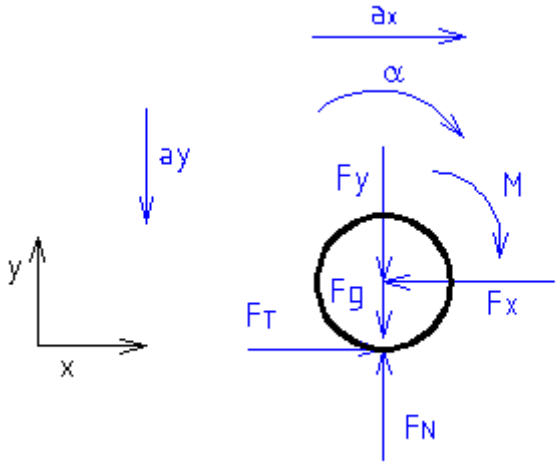
Pro vytvoření modelu dynamické soustavy v Matlabu byla potřebná znalost matematického popisu soustavy.



Obr. 19 Nestabilní dopravní prostředek

V modelu, který byl realizován, neuvažujeme valivé tření ani tření v čepech, deformaci kol ani kyvadla. Dynamické rovnice byly sestavovány pro pohyb v jedné ose.

Pro získání dynamických rovnic bylo potřeba nejdříve soustavu uvolnit. To bylo uděláno tak, že jsme uvolnili tělesa, která byla spojená vazbami. Vyznačili jsme působení sil na tělesa soustavy, síly ve vazbách byly zakresleny u druhého tělesa s opačnou orientací. Síly byly rozloženy do os souřadnicového systému. Pro posun v každé ose a rotaci kolem osy byla napsána dynamická rovnice.

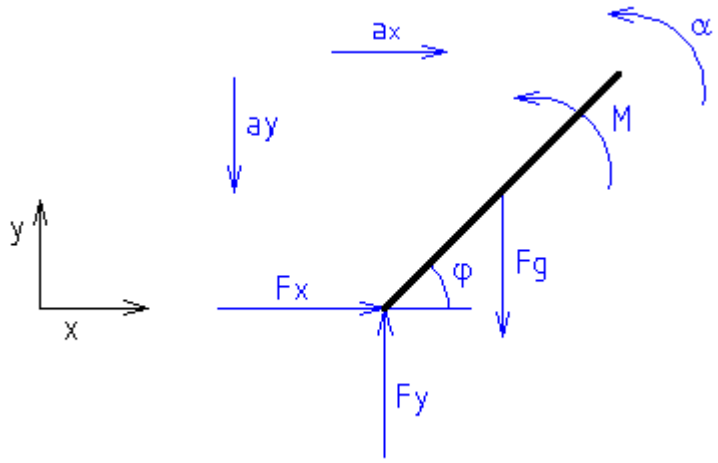


Obr. 20 Uvolnění prvního tělesa soustavy

$$\begin{aligned} X:&\ m\cdot a_x = F_T - F_x \\ Y:&\ -m\cdot a_y = F_N - F_y - F_g \\ X:&\ -I\cdot\alpha = -M + F_T\cdot r \end{aligned} \tag{5.1}$$

Kde
m – hmotnost soustavy
a_x – zrychlení v ose x
F_T – tečná síla
F_x – síla ve vazbě v ose x
a_y – zrychlení v ose y
F_N – normálová síla

F_y – síla ve vazbě v ose y
F_g – tíhová síla
I – moment setrvačnosti
α – úhlové zrychlení
M – řídící moment
r - poloměr kola



Obr. 21 Uvolnění druhého tělesa soustavy

$$X: m \cdot a_x = F_x$$
$$Y: -m \cdot a_y = -F_y - F_g \qquad (5.2)$$
$$M: I \cdot \alpha = M - \frac{F_g \cdot l}{2} \cdot \cos(\varphi)$$

Kde
l – délka kyvadla
φ – úhel naklonění kyvadla

Hmotnost (m) použitá v rovnicích je součtem hmotností kol (m_1) a kyvadla (m_2). Moment setrvačnosti (I) použitý v rovnicích je součtem momentů setrvačnosti kola v ose (I_1)a na konci kyvadla (I_2).

Z rovnic bylo zjištěno, že máme sedm neznámých parametrů a šest rovnic, proto bylo potřeba vyjádřit jeden parametr. Rovnicí bylo vyjádřeno zrychlení v ose x (a_x) pomocí tečného úhlového zrychlení (α_T) a poloměru (r).

$$a_x = \alpha_T \cdot r \quad [26] \qquad (5.3)$$

5.2 Pohybové rovnice soustavy

Pohybové rovnice byly vytvořeny vyřešením dynamických rovnic. Po vyřešení dynamických rovnic byly získány rovnice pro výpočet zrychlení v ose x a úhlové zrychlení.

$$a = \frac{2 \cdot M}{(3 \cdot m \cdot r)} \qquad (5.4)$$

$$\alpha = \frac{M}{I} - \frac{F_g \cdot (\frac{l}{2}) \cdot (\cos(\varphi))}{I} \tag{5.5}$$

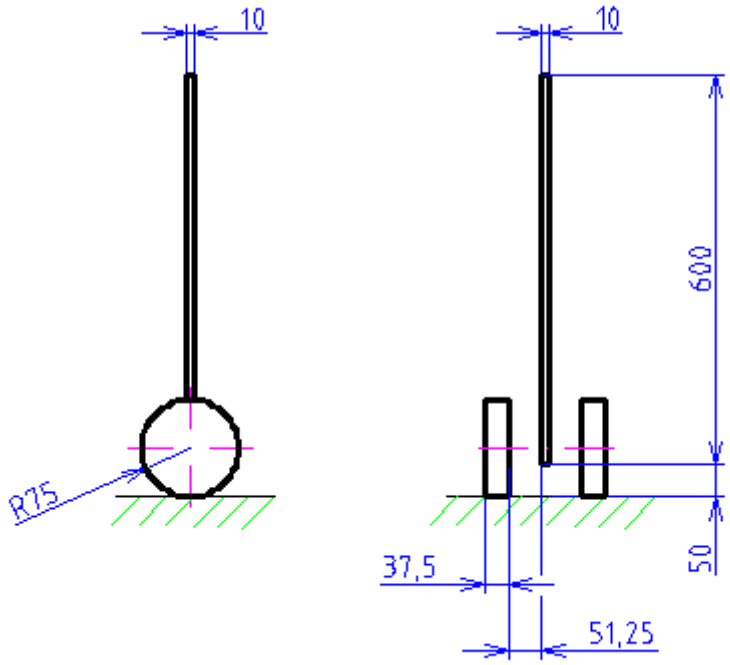
Pomocí integrace bylo dosaženo ze zrychlení v ose x (a_x) rychlost v ose x (v_x) a další integrací polohu v ose x (x). U úhlového zrychlení (α) bylo dosaženo integrací úhlovou rychlost (ω) a po druhou integraci natočení (φ).

$$x = \int (\int a_x dt)\, dt \quad [26] \tag{5.6}$$

$$\varphi = \int (\int \alpha\, dt)\, dt \quad [26] \tag{5.7}$$

6 SESTAVENÍ SIMULAČNÍHO MODELU SOUSTAVY

V této kapitole jsme se zabývali realizací dynamické soustavy. Soustava, která byla realizována pomocí ODE a nadstavbou Matlabu Simulink je nestabilní dvoukolový dopravní prostředek. Skládá se z kyvadla a dvou kol. Po realizaci soustavy začleníme do soustavy PID regulátor.



Obr. 22 Nákres dvoukolového dopravního prostředku

6.1 Vytvoření modelu v ODE

Před započetím práce na realizaci modelu soustavy bylo nutné nainstalovat kromě programovacího jazyku Python 2.5 a knihovny PyODE 1.2.0 také doplňkové knihovny PyOpenGL[28], Matplotlib 0.9.0[29] a Numpy 1.0.2[30].

Při praktické realizaci modelu dynamické soustavy byla pod dohledem vedoucího práce vytvořena jednotná verze struktury zdrojového kódu, používaná v pracích FSI UAI [23][24].

Model soustavy byl rozdělen do unit podle pravidel objektového programování (main1pokus, sTaskToSim, sSim, sPaintSim, vytvor_teleso).

Hlavní unita (main1pokus) při spuštění provádí volání metody pro načtení modelu a spuštění simulace se zadaným časem simulace. Unita, obsahující hlavní cyklus simulace ve kterém se provádí výpočet dynamiky modelu a nastavení parametrů regulátoru, je sTaskToSim. Unita vytvor_teleso obsahuje funkce pro vytváření těles např.: kvádr, válec, koule. O vykreslení těles a práci s knihovnou OpenGL se stará unita sPaintSim. Unita sSim obsahuje funkce pro výpočet kolizí, sestavení modelu a nastavení časového kroku simulace. Vytvoření těles a zavedení vazeb se provádí voláním funkcí z unity vytvor_teleso.

Při pracování s knihovnou ODE bylo potřeba nejdříve importovat tuto knihovnu. Zároveň můžeme importovat knihovny OpenGL pro vykreslení modelu (GL, GLU, GLUT) a knihovny pro práci s grafy (pylab) a knihovna pro práci s matematickými funkcemi např.: trigonometrické funkce.

Při realizaci soustavy s použitím knihovny ODE bylo potřeba vytvořit objekt svět (world), to uděláme přiřazením ode.world do proměnné world. Objekt prostor pro práci s kolizemi byl vytvořen tak, že do proměnné space přiřadíme ode.space.

Pomocí následujícího kódu byl vytvořen objekt svět a prostor.

```
world = ode.World()
space = ode.Space()
```

Po vytvoření objektu svět bylo potřeba vytvořit funkci makeNearCallback pro detekci kolizí. Tato funkce zjišťuje, jestli dvě tělesa kolidují a pokud ano, tak mezi nimi vytvoří kontaktní vazbu(viz. 4.1.2). Funkce nastaví koeficient tření a pružnost při kolizi.

Nyní byly vytvořeny funkce, které vytvářejí tělesa a geometrie pro detekci kolizí. Pro realizaci soustavy byla vytvořena tělesa kvádr pro tělo soustavy a dva válce pro kola.

Funkce create_box vytváří kvádr o zadaných rozměrech a hmotnosti, dále přiřadí tělesu tvar a barvu pro vykreslení tělesa. Potom byla vytvořena geometrie tělesa pro počítání kolizí o tvaru kvádr a stejných rozměrech. Přiřadíme tělesu jméno, které budeme potřebovat při vytváření tělesa v unitě sTaskToSim.Následující kód vytvoří těleso tvaru kvádr se zadanými rozměry x, y, z a hmotnosti. Zadaná hmotnost je rovnoměrně rozložena.

```
def create_box (space, world, X, Y, Z, hmotnost):
    body = ode.Body(world)
    M = ode.Mass()
    M.setBoxTotal(hmotnost, X, Y, Z)
    body.setMass(M)
    geom = ode.GeomBox(space, (X, Y, Z) )
    geom.setBody(body)
    geom.shape = "box"
    geom.color = [128, 128, 128]
    geom.boxsize = (X, Y, Z)
    return body, geom
```

Funkce create_cylinder vytvoří těleso válec o zadaném poloměru, výšce a hmotnosti. Potom vytvoří geometrii tělesa pro počítání kolizí tvaru válec a stejném poloměru a výšce. Stejně jako u kvádru přiřadíme tělesu jméno. Postup vytvoření tělesa válec opakujeme ještě jednou, abychom docílili vytvoření obou kol soustavy

Dále byla vytvořená tělesa vazbami propojená vazbami. V našem případě tělo soustavy propojeno rotačními vazbami s koly. Při vytváření rotačních vazeb zadáváme tělesa, která chceme spojit vazbou, bod, kterým prochází osa a normálu osy, kolem které bude možná rotace. Vytváření rotační vazby ukazuje následující kód.

```
vazba = ode.HingeJoint(self.world)
vazba.attach(self.listOfBodies[0],self.listOfBodies[1])
vazba.setAnchor((0, 0.075, -0.1975))
vazba.setAxis((0,0,1))
self.listOfJoints.append(vazba)
```

Pro vytvoření geometrie statického objektu např.: podlahy zadáme normálu plochy, v našem případě (0,1,0). U statických objektů se nepočítá dynamika tělesa, proto není potřeba zadávat hmotnost. Zavedení objektu podlahy ukazuje následující kód.

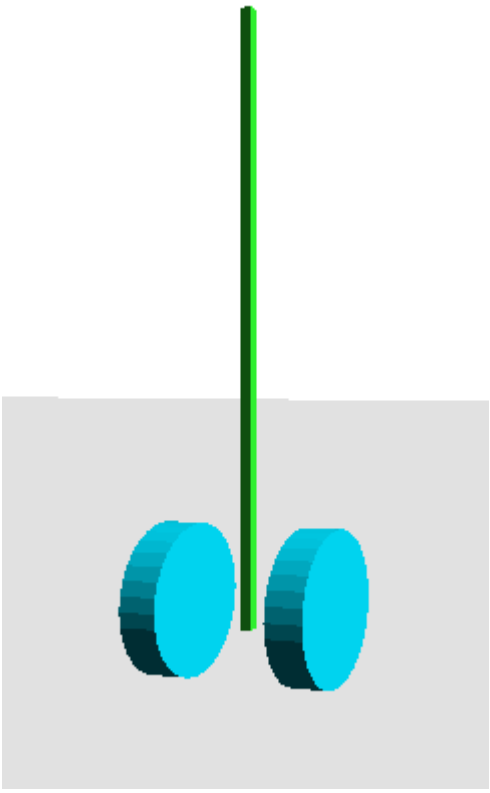
```
geom = ode.GeomPlane(self.space, (0, 1, 0), 0)
```

Pro reálné chování simulace je nutné vhodně nastavit nejen proměnné v unitě sSim: počet snímků za vteřinu (fps), ze kterého se následně vypočítá časový krok simulace (dt), ale také tíhové zrychlení. Parametry ERP a CFM lze nastavit na libovolnou hodnotu mezi 0 a 1.

PID regulátoru byl vytvořen pomocí standardní rovnice regulátoru. Akční moment byl vypočítán v závislosti na natočení, proto je nutné zjistit globální natočení těla soustavy. Funkce matrix2axis, použitá v unitě sTaskToSim pro přepočet matice rotace na globální natočení těla soustavy, byla převzata z[25]. Parametry v PID regulátoru nastavujeme v unitě sTaskToSim a byly

zjištěny experimentálně Ziegler-Nicholsovou metodou.

Vizualizace modelu byla provedena pomocí knihovny OpenGL.



Obr. 23 Pierot vykreslený pomocí OpenGL

K Vykreslení grafů byla použita knihovna PyLab. Proměnné, které chceme vykreslit, musí být tvaru pole. Následující kód vykreslí graf natočení těla soustavy v závislosti na čase. Podobně by vypadal kód pro vykreslení dalších dvou grafů. Rozdíl by byl pouze v proměnných které se mají vykreslit, v názvu grafu a popisu os.

```
plot(time, angleplot, "r-", linewidth=1)
title('NATOCENI')
xlabel('Cas [s]')
ylabel('Uhel [deg]')
grid(True)
```

6.2 Vytvoření modelu v Matlab Simulink

V Simulinku se sestavuje model na základě matematického popisu. Model byl vytvořen z funkčních bloků, které jsou k dispozici v knihovně (browse library). Bloky se vloží do okna modelu a dle pohybových rovnic pospojují.

Při vytváření modelu v simulinku bylo nutné nastavit metodu výpočtu, časový krok simulace (lze zvolit aby se nastavil automaticky), čas začátku a konce simulace. Metoda výpočtu simulace byla zvolena ode3.

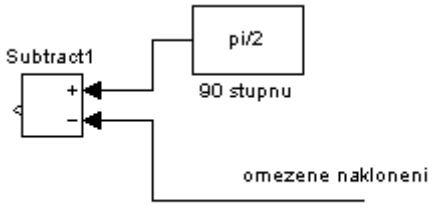
Simulink obsahuje bloky pro základní matematické operace, zdroje signálu (v našem případě konstanty), výstup (v našem případě grafy), pro integrace a derivace, atd.

V našem vytvořeném modelu potřebujeme vložit vstupní parametry, proto zavedeme bloky s konstantami. Vstupní konstanty budou poloměr a hmotnost kol, délka a hmotnost kyvadla a počáteční natočení. Počáteční natočení je měřeno od podlahy, proto kyvadlo má v rovnovážné poloze počáteční úhel 90°.

Kromě bloků s konstantami budeme potřebovat bloky pro základní matematické operace, které

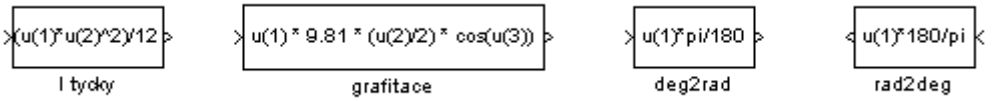
simulink obsahuje.

Protože chceme docílit toho, aby se soustava regulovala do stabilní polohy, to je poloha, kdy je osa těla soustavy kolmá k podlaze, je zapotřebí přepočítat naklonění těla soustavy z 90° na 0°. To provedeme odečtením současného naklonění v radiálech od 90° v radiálech.



Obr. 24 Bloky pro přepočet úhlu naklonění

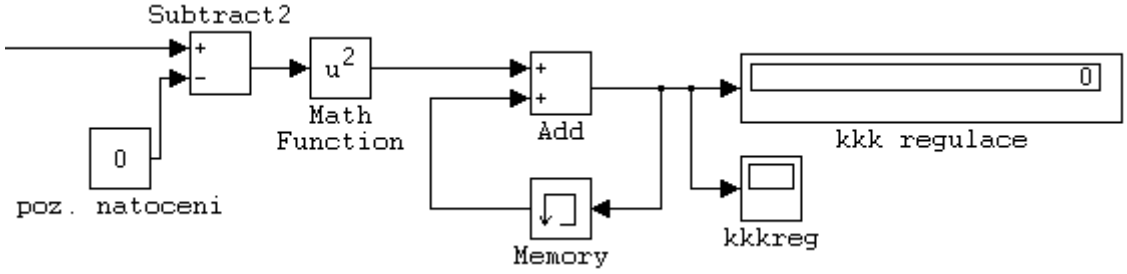
V modelu soustavy jsme použili bloky s námi napsanými funkcemi viz Obr. 25 zleva to jsou: výpočet momentu setrvačnosti kyvadla, výpočet momentu způsobeného gravitační silou, přepočet úhlu ve stupních na radiály a přepočet z radiálů na stupně.



Obr. 25 Bloky s funkcemi

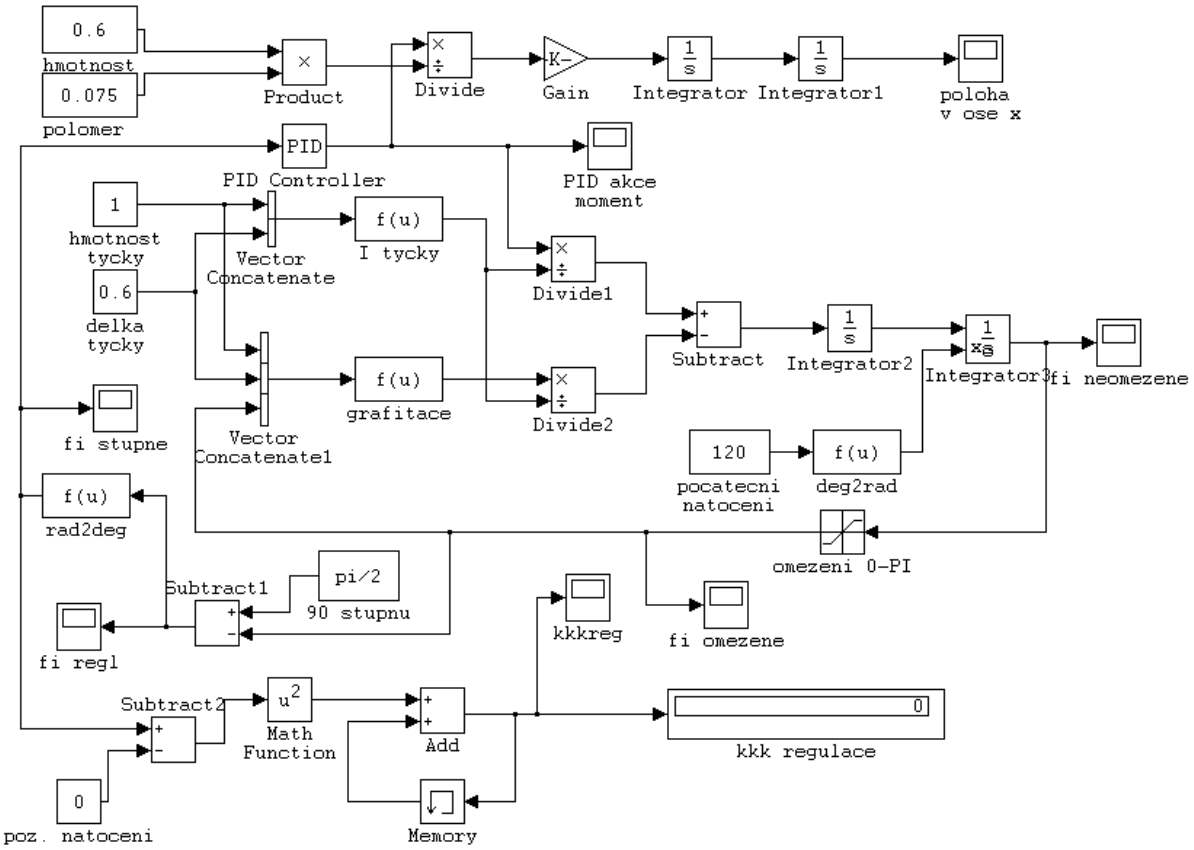
Protože neuvažujeme natočení soustavy větší než 90° a menší než 90° zavedli jsme blok (saturation), v kterém lze nastavit omezení shora i zdola. Blok integrátor funguje stejně jako matematická integrace. S použitím bloku integrátor lze integrací zrychlení vypočítat rychlost a integrací rychlosti vypočítat polohu.

Pro vytvoření výstupu ze simulace použijeme bloky scope, což je graf závislosti vstupní proměnné na čase. Pro zapojení regulátoru do simulačního modelu jsme použili blok PID regulátor. Parametry v PID regulátoru jsme zjistili experimentálně Ziegler-Nicholsovou metodou. Kvadratické kriterium kvality regulace se počítá z aktuálního natočení těla soustavy. Bloky pro výpočet kvadratického kriteria kvality regulace lze pozorovat na Obr. 26.



Obr. 26 Výpočet kvadratického kritéria regulace

Výstupem modelu soustavy jsou grafy: natočení těla soustavy, akční moment regulátoru, poloha soustavy v závislosti na čase a kvadratické kriterium kvality regulace. Celý model soustavy realizovaný v prostředí Matlab Simulink můžeme vidět na Obr. 27.



Obr. 27 Realizovaný model v Simulinku

7 PRAKTICKÉ EXPERIMENTY

Při provádění praktického experimentu byla zkoumána schopnost regulovat úhel naklonění soustavy na požadovanou hodnotu, vyvedenou z klidové polohy. Kvalitu regulace hodnotíme kvadratickým kriteriem kvality regulace. Kvalita regulace je tím vyšší, čím je menší kvadratické kriterium kvality regulace. V modelu vytvořeném pomocí ODE i v modelu vytvořeném v prostředí Matlab je vychýlení z rovnovážné polohy provedeno nakloněním těla soustavy na počátku simulace.

Cílem experimentu bylo shromáždit množinu testovacích dat pro srovnání vlastností knihovny ODE a prostšedí Matlab. V rámci experimentů byly vykresleny tři druhy grafů. První je natočení těla soustavy v závislosti na čase, druhý akční moment regulátoru v závislosti na čase, třetí poloha soustavy v závislosti na čase.

Experiment můžeme rozdělit do tří částí. V první části byly zjištěny parametry PID regulátoru pro časové kroky simulace [s] 0,01; 0,02; 0,05 a následně zjištěna kvalita regulace soustavy. V druhé části byla porovnávána kvalita regulace modelu soustavy vytvořeném pomocí ODE a soustavy, realizované v prostředí Matlab, pro shodné nastavení parametrů PID regulátoru. V třetí části byly zjištěny takové parametry PID regulátoru, které zajišťovaly nejlepší kvalitu regulace soustavy v modelu vytvořeném pomocí ODE.

7.1 Přehled dosažených výsledků

První část experimentu byla provedena pro kroky simulace [s] 0,01; 0,02; 0,05. Úkolem bylo zjištění kvality regulace obou modelů. Cílem této části experimentů bylo navrhnout takové parametry PID regulátoru, aby průběhy regulace soustavy v modelech, vytvořených pomocí ODE a v prostředí Matlab, byly navzájem co nejvíce podobné. Budeme navrhovat parametry PID regulátoru pro časové kroky simulace [s] 0,01; 0,02; 0,05.

Pro časový krok simulace 0,01s, byly vykresleny grafy (viz. Příloha 1) a zjištěno kvadratické kriterium kvality regulace v ODE = 1594,03 Matlab = 1744,44.

Pro časový krok simulace 0,02s, byly vykresleny grafy (viz. Příloha 2) a kvadratické kriterium kvality regulace bylo zjištěno v ODE = 474,91 Matlab = 924,98.

Pro časový krok simulace 0,05s, byly vykresleny grafy (viz. Příloha 3) a kvadratické kriterium kvality regulace bylo zjištěno v ODE = 590,73 Matlab = 427,65.

Z této části experimentů bylo zjištěno, že v ODE i v Matlabu se kvalita regulace zvyšuje, tedy že se snižuje kvadratické kritérium kvality regulace se zvětšujícím se časovým krokem simulace.
V ODE bylo potřeba se změněním časového kroku simulace změnit některé parametry regulátoru.
V Matlabu byly parametry regulátoru pro všechny časové kroky simulace shodné. Parametry PID regulátoru ODE a Matlabu byly při časových krocích simulace 0,01 a 0,02 shodné kromě derivačního členu který byl v ODE 1,4 a v Matlabu 0,01. Při použití časového kroku simulace 0,05 bylo potřeba z důvodu zregulování soustavy do klidové polohy v ODE snížit proporcionální člen na 0,2, derivační člen na 0,2 a integrační člen na 0,001, také bylo nutné zvýšit čas simulace z 0,4s na 0,6s. Čas simulace modelu, vytvořeného v prostředí Matlab, byl ve všech časových krocích simulace 0,5s.

Druhá část experimentu byla provedena pro shodné nastavení parametrů regulátorů modelů při nastaveném konstantním časovém kroku simulace 0,01s. V této části experimentů budeme porovnávat kvadratické kritérium kvality regulace modelu, sestaveného pomocí ODE a modelu, vytvořeného v prostředí Matlab.

Pro parametry PID regulátoru: p = 0,4; i = 0,0; d = 0,0 shodné pro oba modely byly vykresleny grafy (viz. Příloha 4) a zjištěno kvadratické kriterium kvality regulace v ODE = 16638,74, Matlab = 17846,66.

Pro parametry PID regulátoru: p = 0,4; i = 0,0; d = 0,1 shodné pro oba modely byly vykresleny grafy (viz. Příloha 5) a zjištěno kvadratické kriterium kvality regulace v ODE = 4310,55, Matlab = 3332,30.

Pro parametry PID regulátoru: p = 0,2; i = 0,0; d = 0,1 shodné pro oba modely byly vykresleny grafy (viz. Příloha 6) a zjištěno kvadratické kriterium kvality regulace v ODE = 4189,17, Matlab = 7662,52.

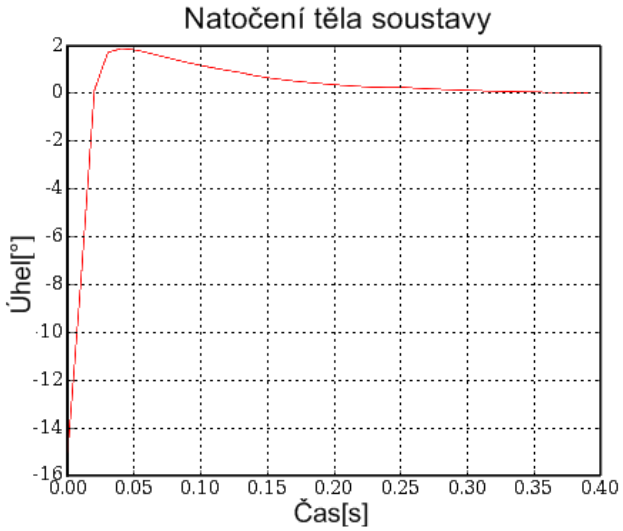
Pro parametry PID regulátoru: p = 0,2; i = 0,001; d = 0,1 shodné pro oba modely byly

vykresleny grafy (viz. Příloha 7) a zjištěno kvadratické kriterium kvality regulace v ODE = 5148,44, Matlab = 7629,86.

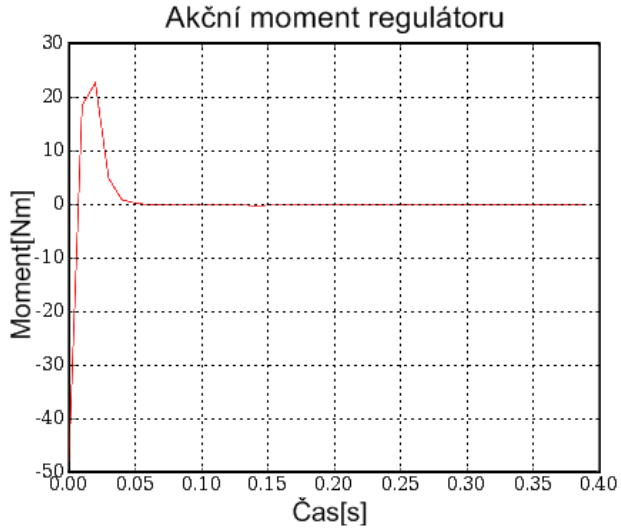
V této části experimenu bylo zjištěno, že model realizovaný v Matlabu je citlivější na změnu derivačního členu regulátoru než model realizovaný pomocí ODE, což lze usoudit z grafů v Příloze 4 a 5. Při nastavení regulátoru s proporciálním členem p = 0,4 pro oba modely byly zjištěny nejvyšší hodnoty kvadratického kriteria kvality regulace a to pro Matlab = 17846,66 a pro ODE = 16638,74. Nejnižší hodnota kvadratického kritéria kvality regulace pro Matlab = 3332,3 byla zjištěna při nastavení regulátoru s proporcionálním členem p = 0,4 a derivačním členem d = 0,1, pro ODE = 4189,17 při nastavení regulátoru s proporcionálním členem p = 0,1 a derivačním členem d = 0,1. Při zavedení derivačního členu do regulátoru bylo sníženo kvadratické kriterium kvality regulace pro Matlab z 17846,66 na 3332,30 u ODE z 16638,74 na 4310,55. Dále bylo zjištěno, že integrační člen má velmi malý vliv na změnu kvadratického kritéria kvality regulace, protože u modelu, vytvořeného v prostředí Matlab se kvadratické kriterium regulace nepatrně snížilo a u modelu, vytvořeného pomocí ODE se kvadratické kriterium kvality regulace zvýšilo.

Třetí část experimentu byla provedena pouze pro model, vytvořený pomocí ODE pro časové kroky simulace [s] 0,01; 0,02; 0,05. Úkolem bylo navrhnout parametry regulátoru s nejlepší kvalitou regulace, což znamená nejmenší kvadratické kriterium kvality regulace.

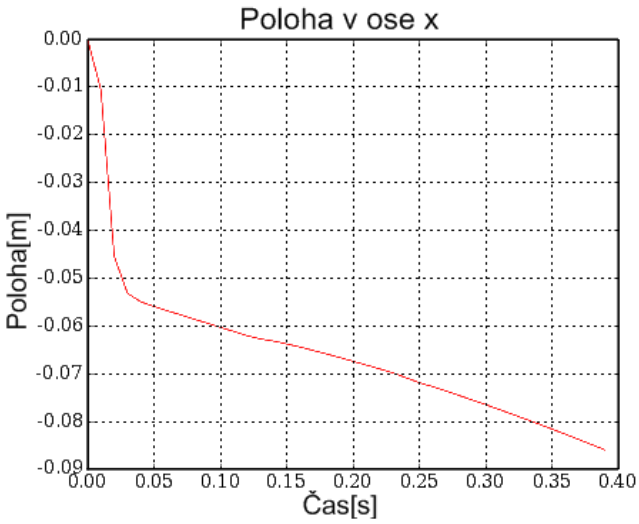
Následující grafy byly vytvořené s časem simulace 0,4s a s časovým krokem simulace 0,01s. Velikost počátečního natočení soustavy je 15° a nastavení parametrů PID regulátoru: p = 0,3; i = 0,005; d = 2,9. Kvadratické kritérium kvality regulace bylo zjištěno 311,25.



Obr. 28 Graf natočení těla soustavy v závislosti na čase

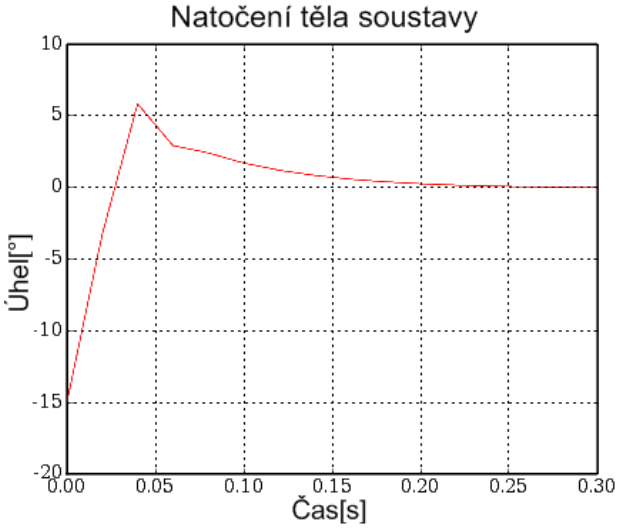


Obr. 29 Graf akčního momentu regulátoru v závislosti na čase

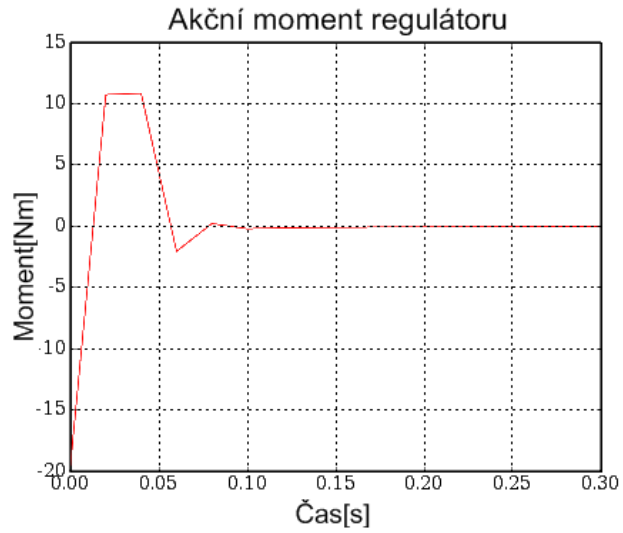


Obr. 30 Graf polohy soustavy v závislosti na čase

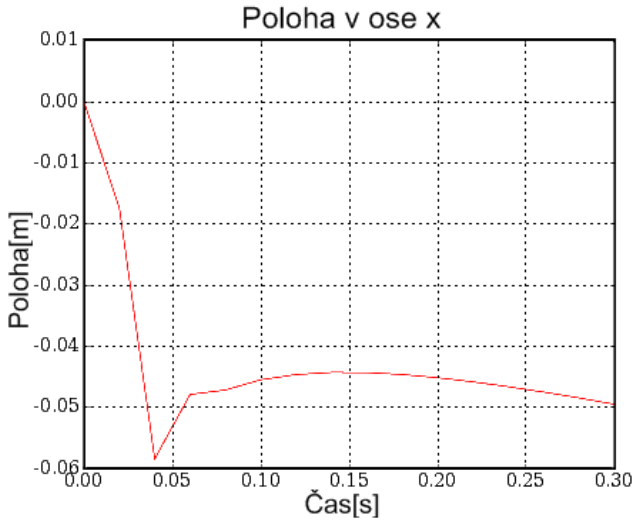
Následující grafy byly vytvořené s časem simulace 0,3s a s časovým krokem simulace 0,02s. Velikost počátečního natočení soustavy je 15° a nastavení parametrů PID regulátoru: p = 0,3; i = 0,001; d = 1,0. Kvadratické kritérium kvality regulace bylo zjištěno 289,83.



Obr. 31 Graf natočení soustavy v závislosti na čase

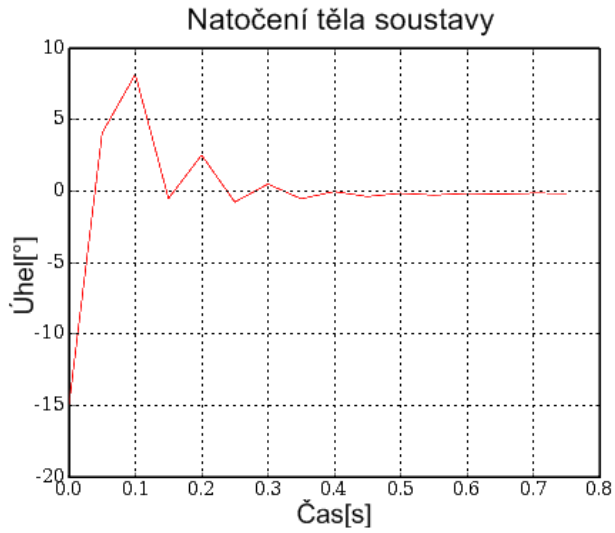


Obr. 32 Graf akčního momentu regulátoru v závislosti na čase

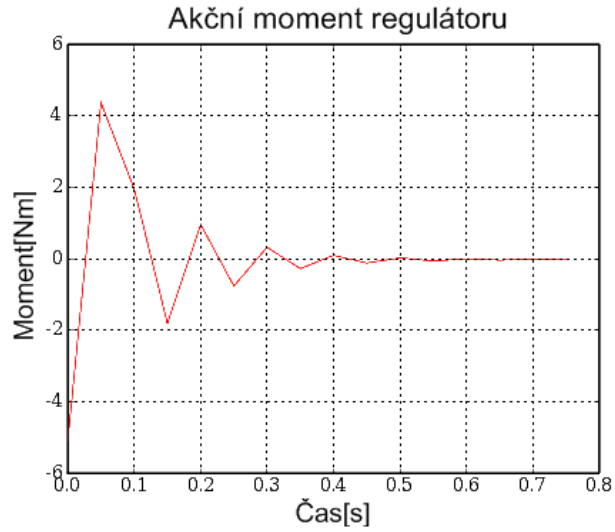


Obr. 33 Graf polohy soustavy v závislosti na čase

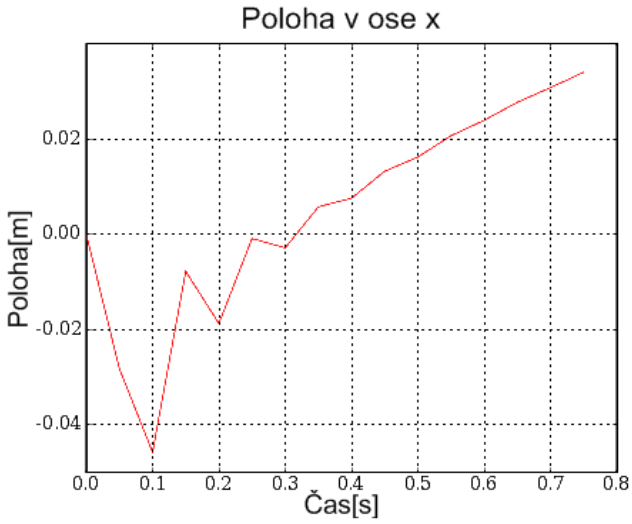
Následující grafy byly vytvořené s časem simulace 0,8s a časovým krokem simulace 0,05s. Velikost počátečního natočení soustavy je 15° a nastavení parametrů PID regulátoru: p = 0,14; i = 0,001; d = 0,2. Kvadratické kritérium kvality regulace bylo zjištěno 316,19.



Obr. 34 Graf natočení těla soustavy v závislosti na čase



Obr. 35 Graf akčního momentu regulátoru v závislosti na čase



Obr. 36 Graf polohy soustavy v závislosti na čase

Z této části experimentů bylo zjištěno, že kvadratické kritérium kvality regulace je oproti

předchozím částem experimentu znatelně nižší a se změnou časového kroku simulace se výrazně nemění. Kvadratické kritérium kvality regulace je 311,25; 289,83; 316,19 pro časové kroky simulace (0,01s; 0,02s; 0,05s). Nejvyšší kvadratické kritérium kvality regulace bylo zjištěno pro časový krok simulace 0,05s. Při změně časového kroku simulace bylo nutné změnit parametry regulátoru, ale počáteční úhel natočení těla soustavy je pro všechny časové kroky simulace stejný 15°.

8 ZÁVĚR

Cílem práce bylo sestavení dynamického modelu nestabilní soustavy pomocí knihovny ODE a simulačního prostředí Matlab. Pro demonstrování funkčnosti ODE byla navrhnuta dynamická soustava dvoukolového dopravního prostředku. Tato soustava byla popsána dynamickými rovnicemi a z dynamických rovnic byl vytvořen matematický popis soustavy, nutný pro realizaci modelu soustavy.

Byla provedena realizace modelu soustavy nejprve pomocí ODE a následně v simulačním prostředí Matlab. Spolu s realizací modelu soustavy byl vytvořen a implementován regulátor pro regulaci úhlu naklonění soustavy. Pod dohledem vedoucího práce byla vytvořena jednotná verze struktury zdrojových kódů, používaná v pracích FSI UAI [23][24].

Byly provedeny praktické experimenty, jejichž součástí bylo vytvoření grafů (úhlu naklonění soustavy v závislosti na čase, akčního momentu v závislosti na čase a polohy soustavy v závislosti na čase). Výsledkem experimentů bylo zjištění kvality regulace pomocí kvadratického kriteria kvality regulace pro různá nastavení modelu soustavy (časový krok simulace, parametry PID regulátoru, čas simulace, počáteční úhel natočení soustavy) pro model, realizovaný pomocí ODE i pro model, vytvořený pomocí programu Matlab. Na základě porovnání výsledků experimentů modelu, vytvořeného pomocí ODE a modelu, vytvořeného v prostředí Matlab, bylo zjištěno, že knihovna ODE je použitelná pro simulaci dynamických soustav.

Výhody modelu, realizovaného pomocí knihovny ODE jsou reálnější chování (knihovna počítá: detekci kolizí, tření, pružnost ve vazbách a při kolizi), možnost vizualizace po přidání knihovny OpenGL, zjednodušení vytváření modelu (znalost základních těles a vazeb).

Nevýhody jsou nutná znalost programovacího jazyka, nutnost přídavných knihoven pro vykreslení grafů a vizualizaci.

SEZNAM POUŽITÉ LITERATURY

[1] ŠŤASTNÝ, Jiří. *Simulace systémů.* Brno 2002. 98 s. Pozn.: Učební text

[2] MINAŘÍK, Matrin. *Simulace* [online]. [cit.2008-4-16]. Dostupné z: <http://www.stud.fme.vutbr.cz/~yminar02/VSA.htm>

[3] PETERKA, Jiří. Simulace vs. emulace. *Computerworld* [online]. 1992. č. 10/92 [cit.2008-4-16]. Dostupné na WWW: <http://www.earchiv.cz/a92/a210c120.php3>

[4] WIKIPEDIA. Dotaz: Počítačová simulace [online]. [cit.2008-4-18] Dostupné z: <http://cs.wikipedia.org/wiki/Počítačová_simulace>

[5] WIKIPEDIA. Dotaz: OpenGL [online]. [cit.2008-4-18]. Dostupné z: <http://cs.wikipedia.org/wiki/OpenGL>

[6] WIKIPEDIA. Dotaz: DirectX [online]. [cit.2008-4-18]. Dostupné z: <http://cs.wikipedia.org/wiki/DirectX>

[7] WIKIPEDIA. Dotaz: C++ [online]. [cit.2008-4-18]. Dostupné z: <http://cs.wikipedia.org/wiki/C%2B%2B>

[8] WIKIPEDIA. Dotaz: Open Dynamics Engine [online]. [cit.2008-4-18]. Dostupné z: <http://en.wikipedia.org/wiki/Open_Dynamics_Engine>

[9] WIKIPEDIA. Dotaz: Python [online]. [cit.2008-4-18]. Dostupné z: <http://cs.wikipedia.org/wiki/Python>

[10] SMITH, Russell. *Open Dynamics Engine* [online]. 2000, 28th of May 2007 [cit.2008-4-18]. Dostupné z: <http://www.ode.org>

[11] Havok [online]. 28th of April 2009 [cit.2009-5-6]. Dostupné z: <http://www.havok.com>

[12] Newton Game Dynamics [online]. [cit.2008-4-18]. Dostupné z: <http://www.newtondynamics.com>

[13] NVIDIA PhysX [online]. [cit.2008-4-18] Dostupné z: <http://www.nvidia.com/object/nvidia_physx.html>

[14] Physics Simulation Forum [online]. [cit.2008-4-18]. Dostupné z: <http://www.bulletphysics.com>

[15] OpenGL [online]. [cit.2008-4-18]. Dostupné z: <http://www.opengl.org>

[16] Microsoft DirectX [online]. [cit.2008-4-18]. Dostupné z: <http://www.techmixer.com/microsoft-directx-10-final-directx-101>

[17] Humusoft [online]. [cit.2009-5-6]. Dostupné z: <http://www.humusoft.cz/produkty/matlab/index.php?lang=cz&p1=1&p2=1>

[18] Autodesk [online]. [cit.2008-4-19]. Dostupné z: <http://www.autodesk.cz>

[19] PyCZ [online]. [cit.2008-4-19]. Dostupné z: <http://www.py.cz>

[20] STUDER, Greg. *Open Dynamics Engine (ODE) For Evolutionary Robotics* [PDF dokument]. Informatics University of Sussex, January 2007 [cit.2008-5-12]. Dostupné z: <http://www.informatics.sussex.ac.uk/users/gms21/ode_presentation_2007.pdf>

[21] TIŠNOVSKÝ, Pavel. Grafická knihovna OpenGL. *Root.cz* [online]. 2003, č. 1 [cit.2008-5-12]. Dostupný na WWW: <http://www.root.cz/clanky/graficka-knihovna-opengl-1>

[22] BRETTSCHNEIDER, Zbyněk. <Brettsz@fel.cvut.cz> *Matlab – Simulink* [PPT dokument]. Praha, Fakulta elektrotechniky ČVUT [cit.2008-5-15]. Dostupný z: <http://www.powerwiki.cz/attach/ProStudenty/Simulink.ppt>

[23] SERIŠ, R. *Využití ODE pro Sestavení Dynamického Modelu Čtyřnohého Robotu*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. Vedoucí bakalářské práce Ing. Vít Ondroušek.

[24] SCHREIBER, P. *Návrh automatického generátoru prostředí pro mobilní robot*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. Vedoucí bakalářské práce Ing. Stanislav Věchet, Ph.D.

[25] PyODE Rotation Matrices [online]. 16th of January 2009 [cit.2009-5-6]. Dostupné z: <http://nothing.co.nz/Programming/PyODERotationMatrices>

[26] JANDORA, Radek. *Kinematika hmotného bodu* [online]. [cit.2008-5-22]. Dostupné z: <http://www.sweb.cz/radek.jandora/f01.htm>

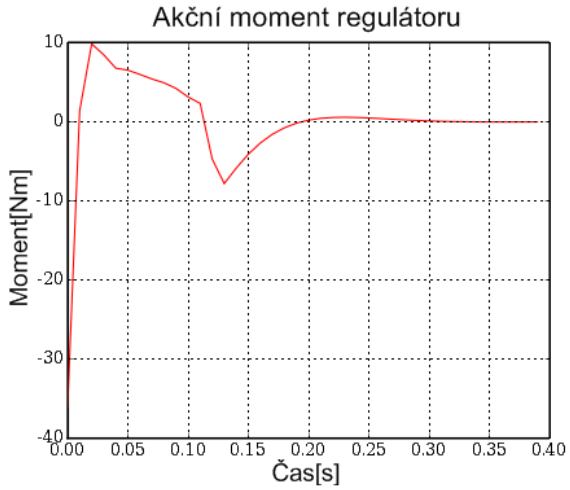
[27]ANDERSON, David. *nBot Balancing Robot* [online]. [cit.2008-3-22]. Dostupné z: <http://www.geology.smu.edu/~dpa-www/robo/nbot>
[28]PyOpenGL [online]. [cit.2009-5-26]. Dostupné z : <http://pyopengl.sourceforge.net>
[29]Matplotlib [online]. [cit.2009-5-26]. Dostupné z: <http://matplotlib.sourceforge.net>
[30]Numpy [online]. [cit. 2009-5-26]. Dostupné z: <http://numpy.scipy.org>

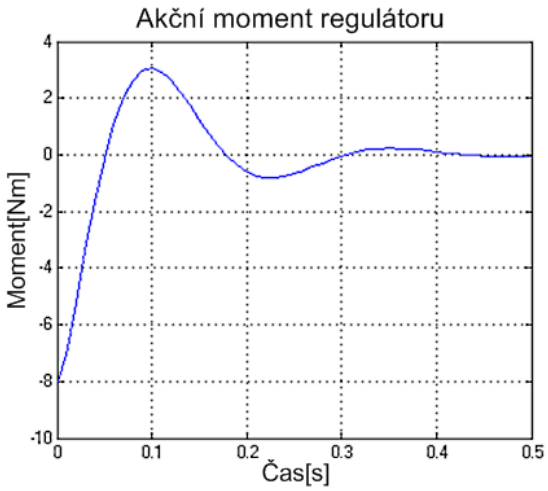
SEZNAM PŘÍLOH

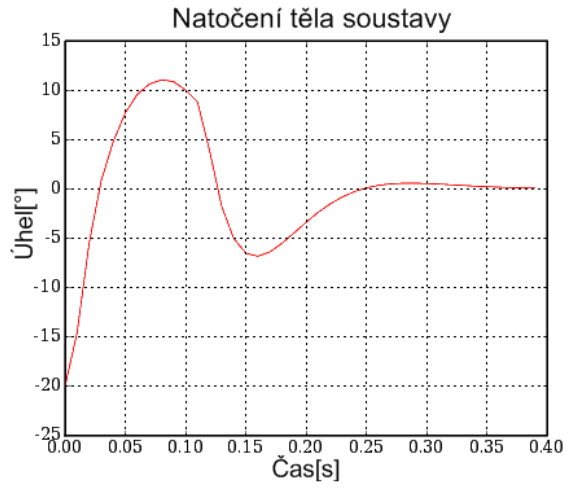
PŘÍLOHY

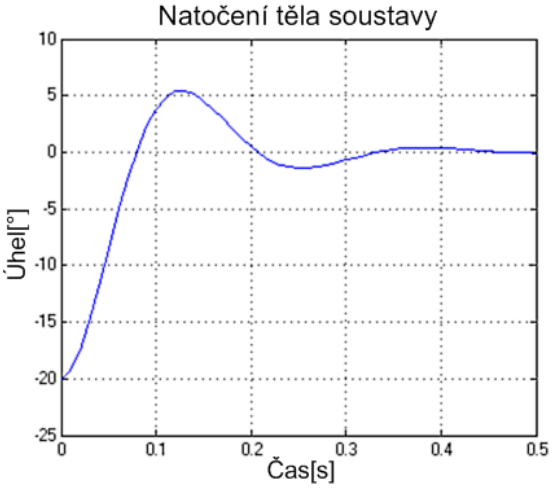
Příloha 1

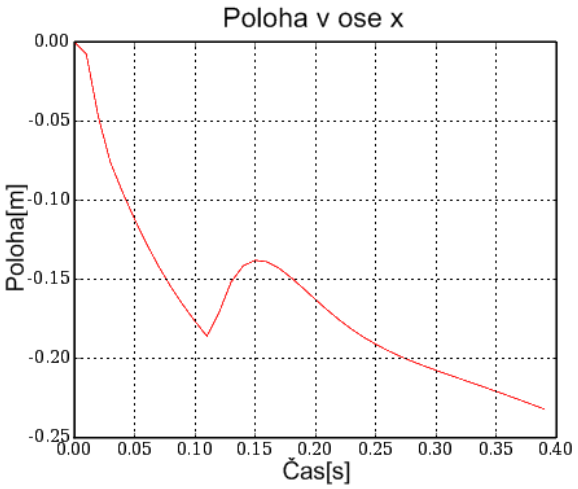
Následující grafy (zleva ODE, Matlab) byly vytvořeny s časem simulace v ODE 0,4s, v Matlabu 0,5s a krokem simulace 0,01s. Počáteční natočení soustavy bylo pro oba modely shodné 20°. Při nastavení PID regulátoru modelu v ODE: p = 0,4; i = 0,019; d = 1,4 a v Matlabu: p = 0,4; i = 0,019; d = 0,01.

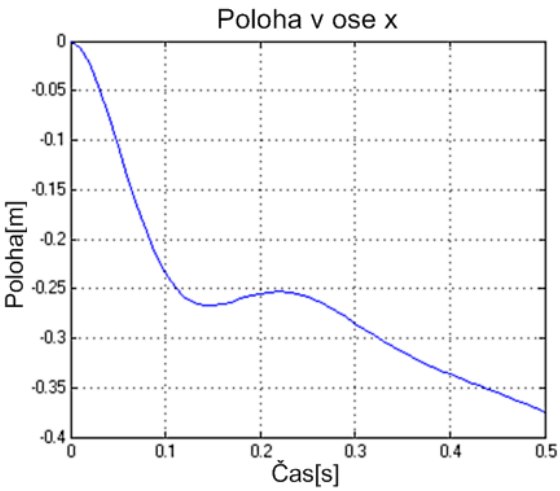






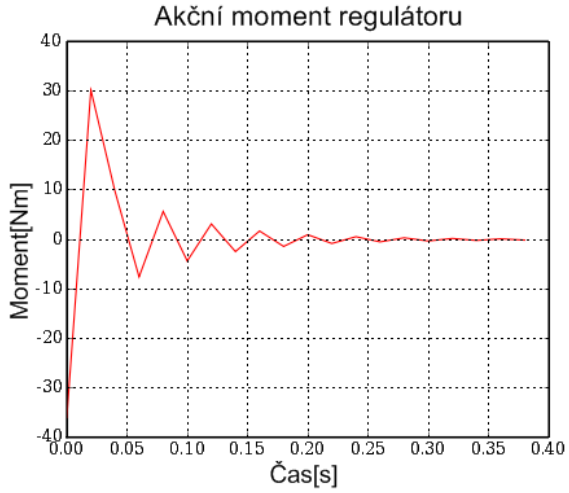


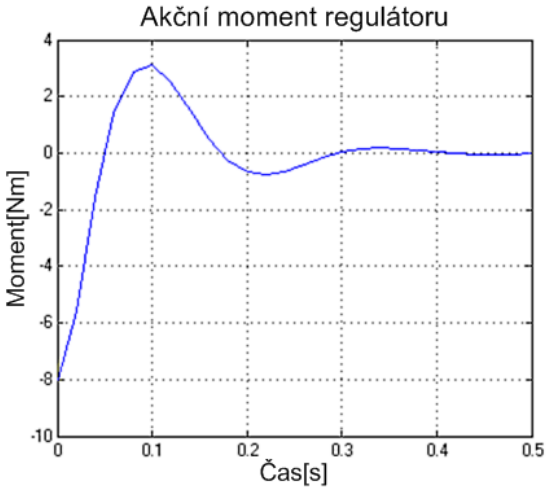


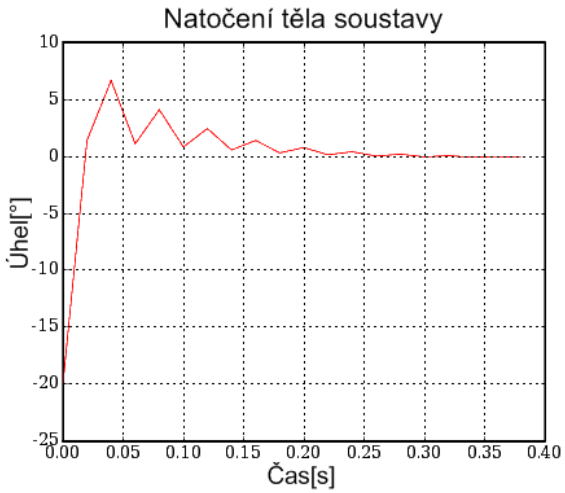


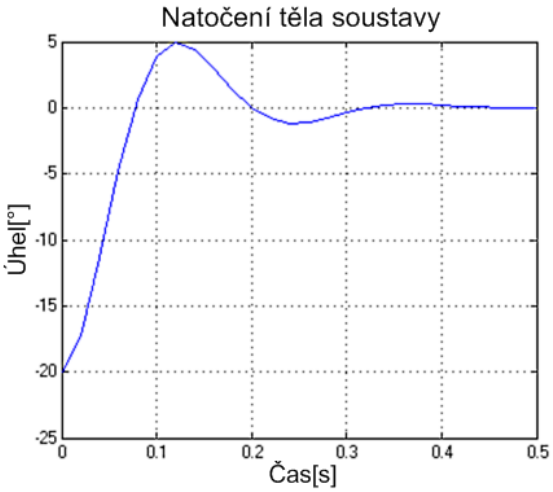
Příloha 2

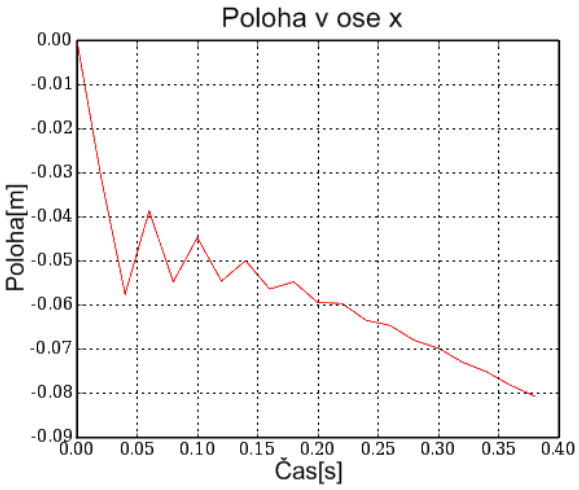
Následující grafy (zleva ODE, Matlab) byly vytvořeny s časem simulace v ODE 0,4s, v Matlabu 0,5s a krok simulace 0,02s. Počáteční natočení soustavy bylo pro oba modely shodné 20°. Při nastavení PID regulátoru modelu v ODE: p = 0,4; i = 0,019; d = 1,4 a v Matlabu: p = 0,4; i = 0,019; d = 0,01.

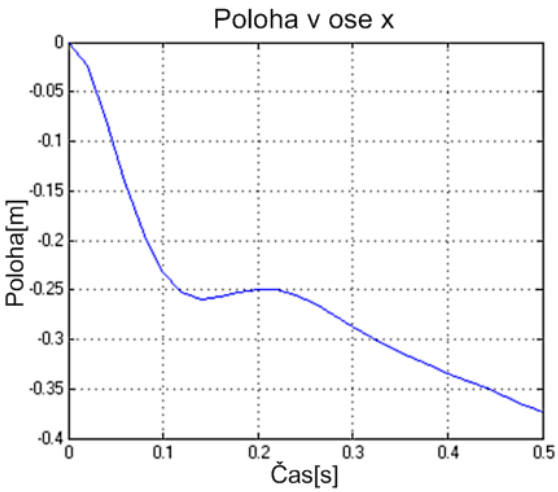






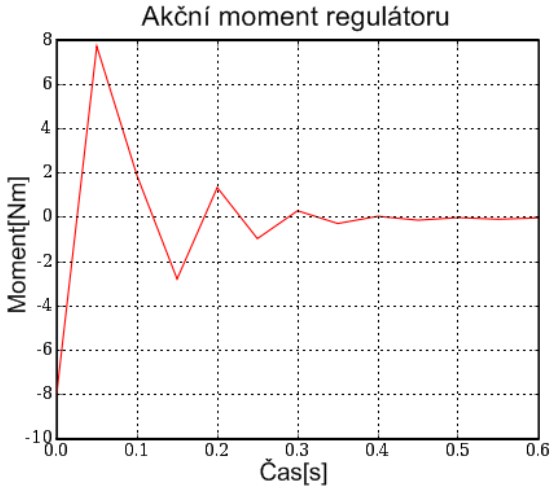


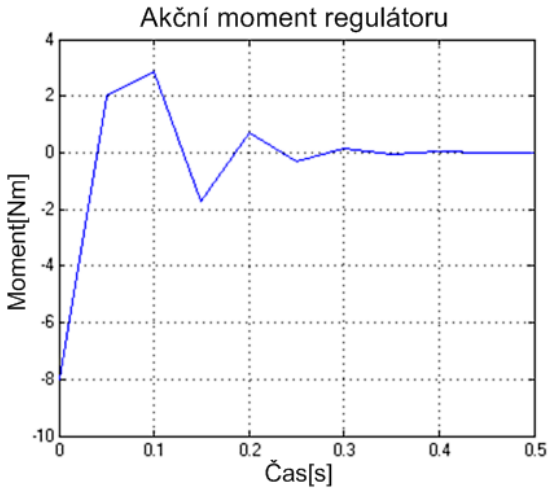


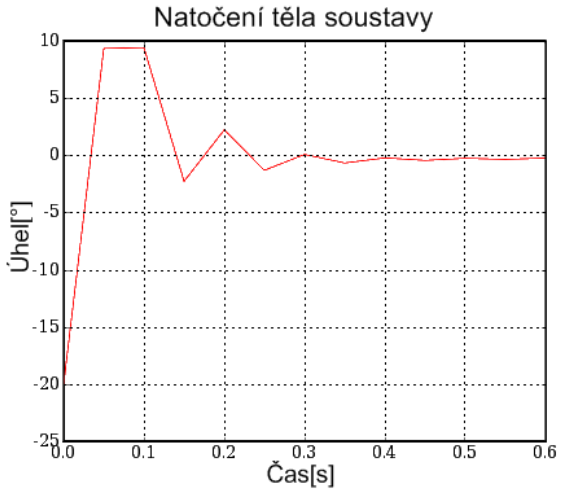


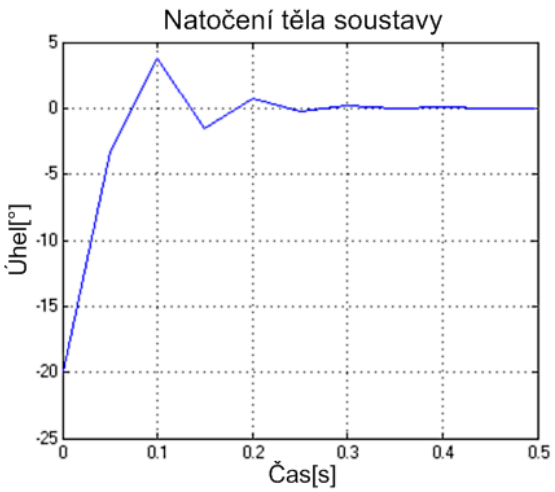
Příloha 3

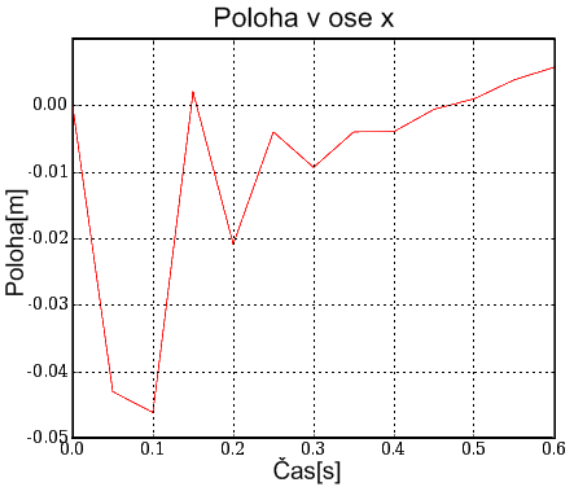
Následující grafy (zleva ODE, Matlab) byly vytvořeny s časem simulace v ODE 0,6s, v Matlabu 0,5s a krok simulace 0,05s. Počáteční natočení soustavy bylo pro oba modely shodné 20°. Při nastavení PID regulátoru modelu v ODE: p = 0,2; i = 0,001; d = 0,2 a v Matlabu: p = 0,4; i = 0,019; d = 0,01.

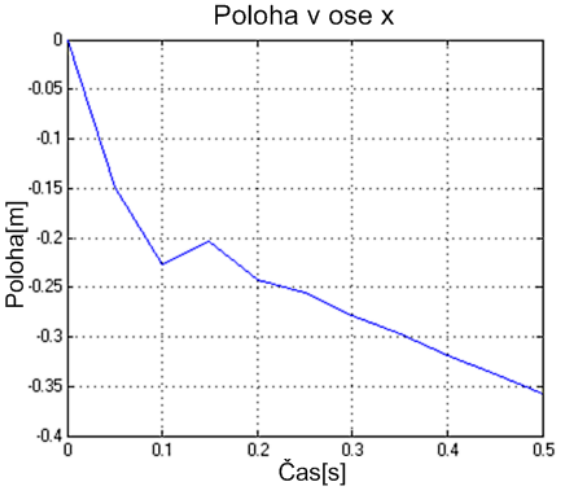






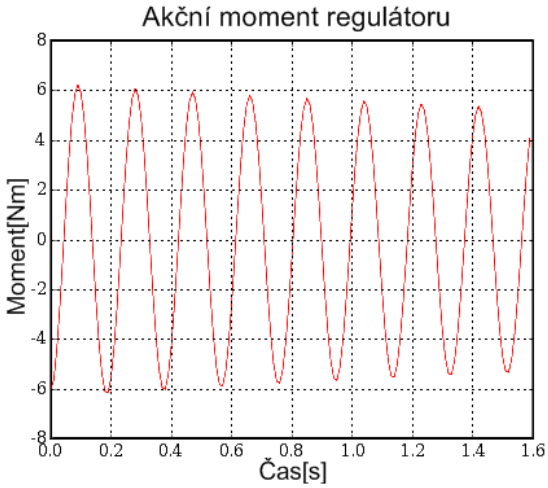


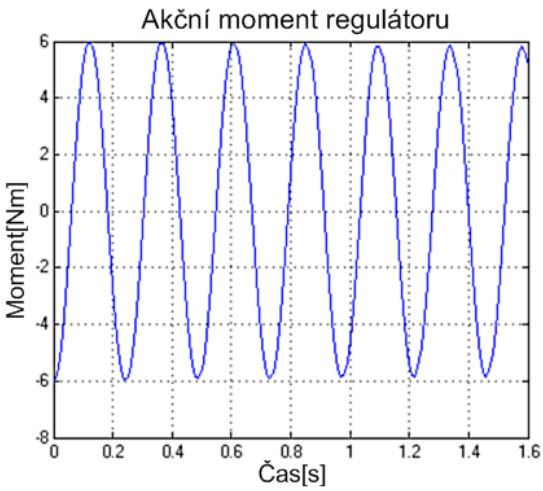


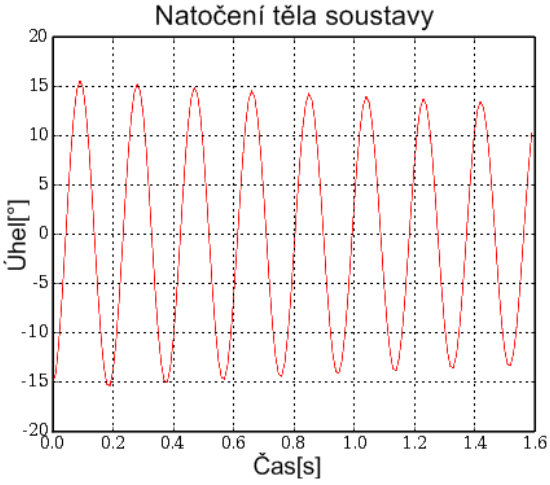


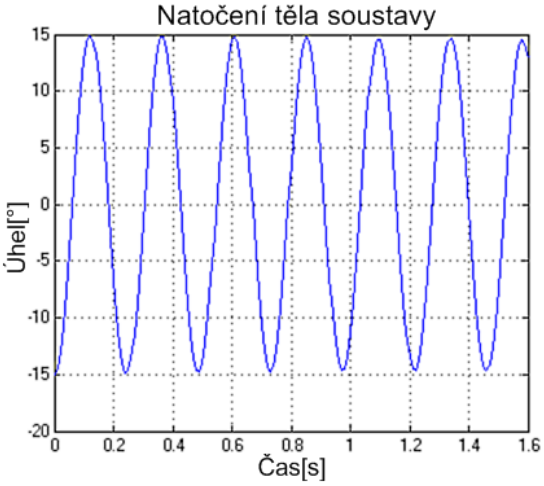
Příloha 4

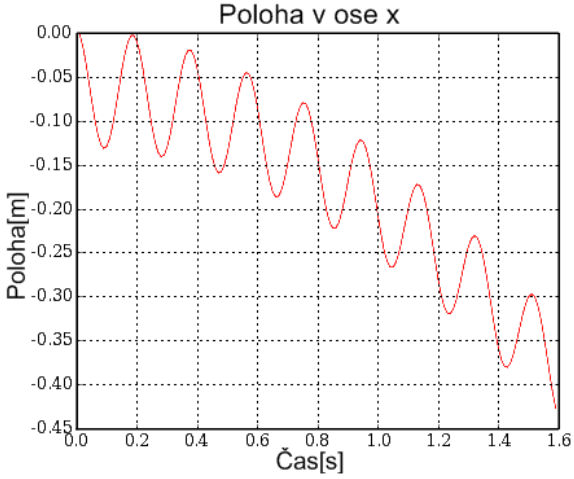
Následující grafy (zleva ODE, Matlab) byly vytvořeny s časem simulace v ODE 1,6s, v Matlabu 1,6s a krok simulace 0,01s. Počáteční natočení soustavy bylo pro oba modely shodné 15°. Při nastavení PID regulátoru modelu v ODE: p = 0,4; i = 0,0; d = 0,0 a v Matlabu: p = 0,4; i = 0,0; d = 0,0.

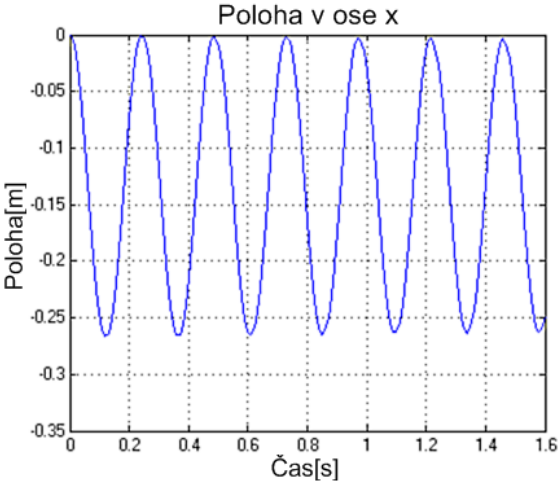






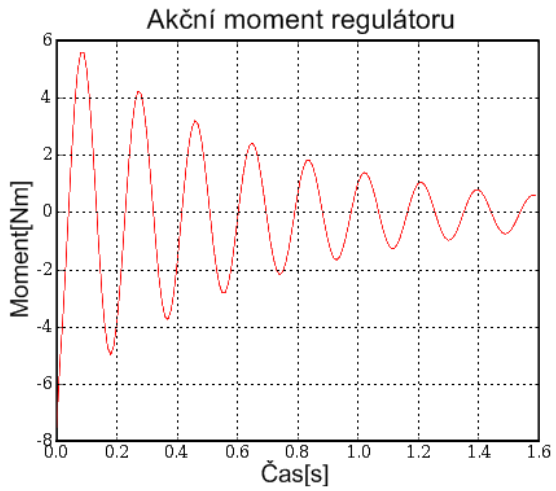


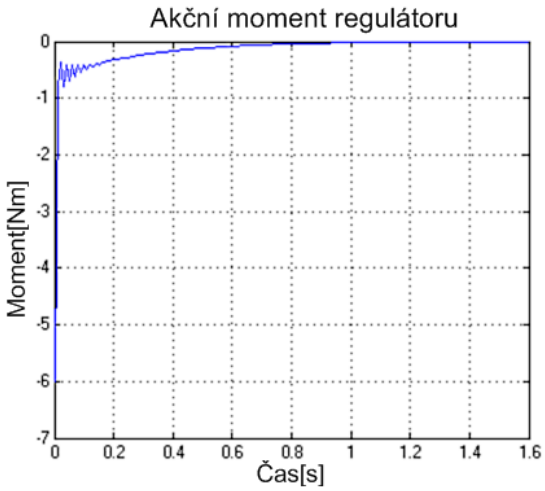


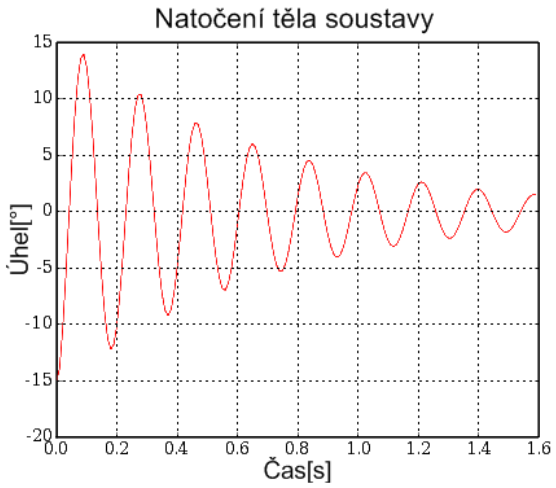


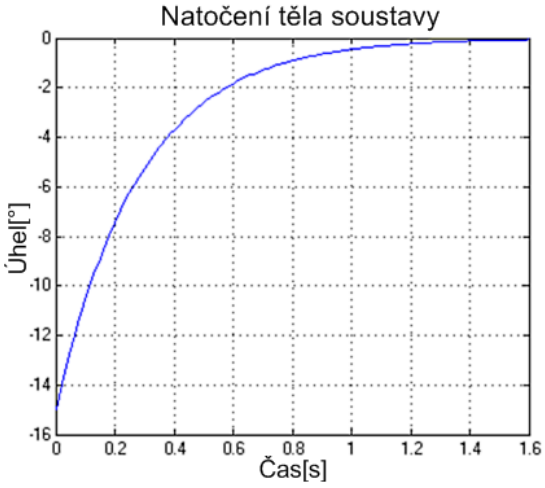
Příloha 5

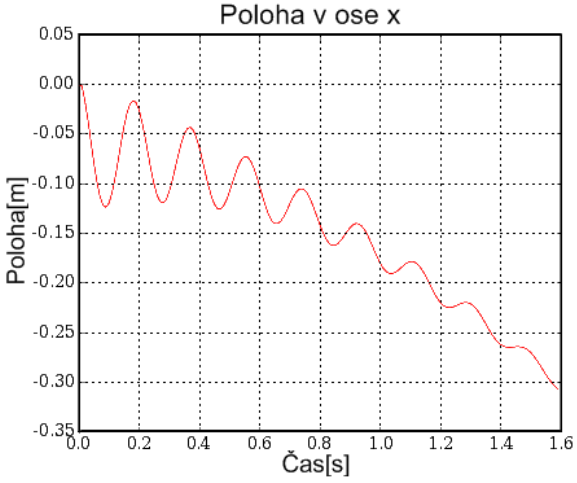
Následující grafy (zleva ODE, Matlab) byly vytvořeny s časem simulace v ODE 1,6s, v Matlabu 1,6s a krok simulace 0,01s. Počáteční natočení soustavy bylo pro oba modely shodné 15°. Při nastavení PID regulátoru modelu v ODE: p = 0,4; i = 0,0; d = 0,1 a v Matlabu: p = 0,4; i = 0,0; d = 0,1.

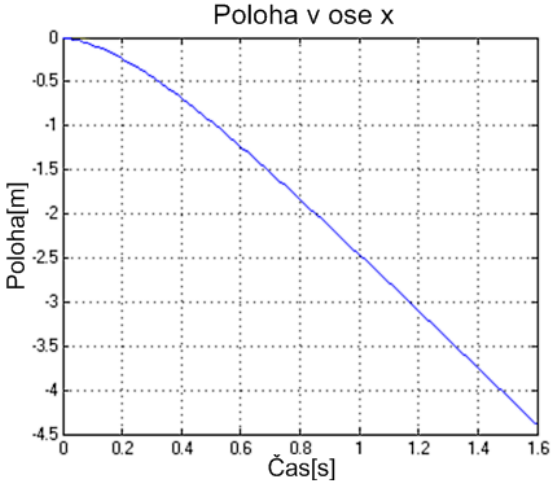






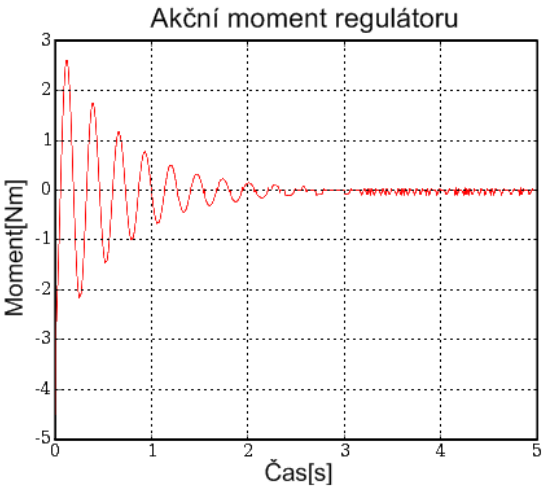


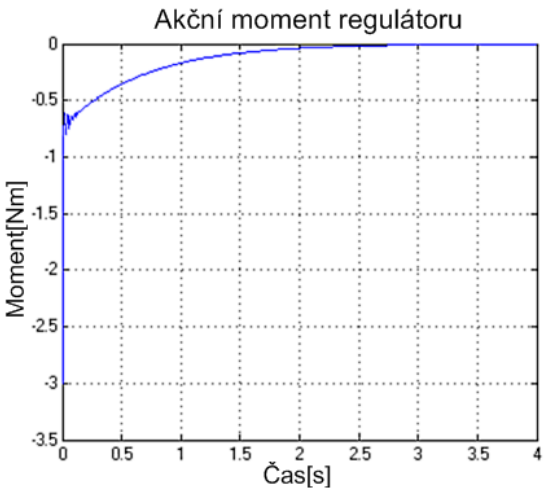


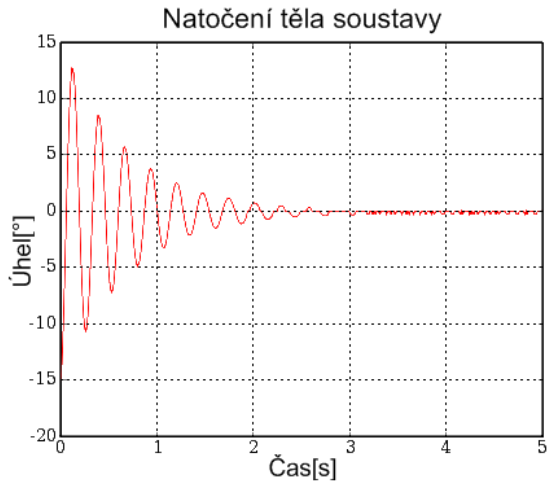


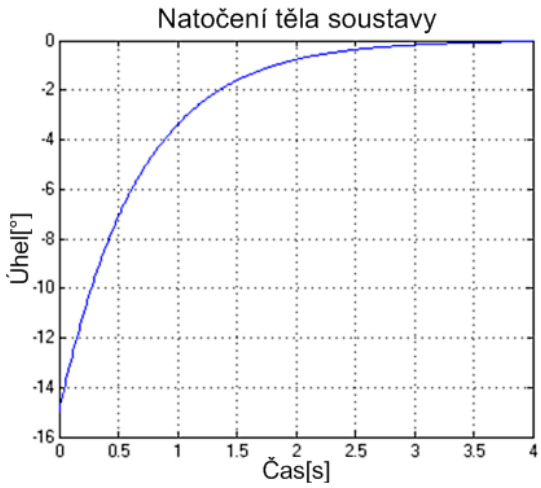
Příloha 6

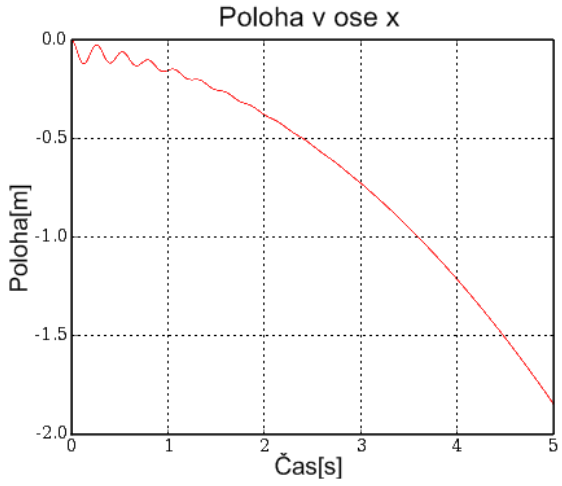
Následující grafy (zleva ODE, Matlab) byly vytvořeny s časem simulace v ODE 5,0s, v Matlabu 4,0s a krok simulace 0,01s. Počáteční natočení soustavy bylo pro oba modely shodné 15°. Při nastavení PID regulátoru modelu v ODE: p = 0,2; i = 0,0; d = 0,1 a v Matlabu: p = 0,2; i = 0,0; d = 0,1.

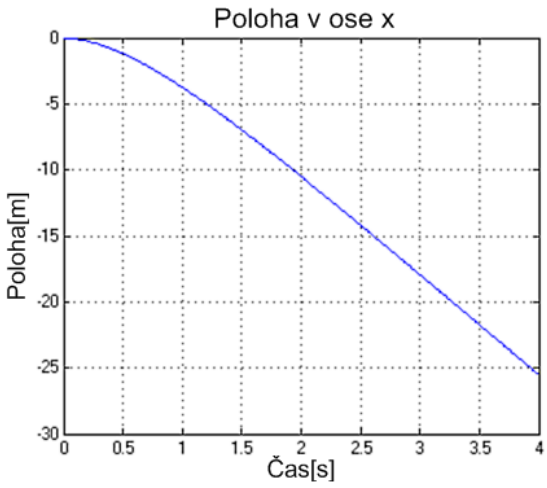












Příloha 7

Následující grafy (zleva ODE, Matlab) byly vytvořeny s časem simulace v ODE 5,0s, v Matlabu 4,0s a krok simulace 0,01s. Počáteční natočení soustavy bylo pro oba modely shodné 15°. Při nastavení PID regulátoru modelu v ODE: p = 0,2; i = 0,001; d = 0,1 a v Matlabu: p = 0,2; i = 0,001; d = 0,1.