VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ŘÍZENÍ MODELU ROBOTICKÉHO HADA

CONTROL OF A ROBOTIC SNAKE MODEL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Martin Levek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Tomáš Hůlka
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Martin Levek**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Tomáš Hůlka**
Akademický rok:	2021/22

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 diplomové práce:

Řízení modelu robotického hada

Stručná charakteristika problematiky úkolu:

Úkolem studenta bude navrhnout řízení modelu robotického hada, to poté implementovat v simulovaném prostředí s náhodně generovanými překážkami. Testovací úlohou pro model bude detekovat překážky v prostředí, vyhnout se jim a dosáhnout zadaného cíle.

Cíle diplomové práce:

Stručná rešerše problematiky.
Návrh a implementace řízení simulačního modelu robotického hada.
Otestování funkčnosti na zadané testovací úloze – navigace v prostředí s překážkami.

Seznam doporučené literatury:

HRDINA, Jaroslav, et al. "CGA-based robotic snake control." Advances in Applied Clifford Algebras 27.1 (2017): 621-632.

BING, Zhenshan, et al. "Perception-action coupling target tracking control for a snake robot via reinforcement learning." Frontiers in Neurorobotics 14 (2020): 79.

HŮLKA, Tomáš, et al. "Optimization of snake-like robot locomotion using ga: Serpenoid design." Mendel. Vol. 26. No. 1. 2020.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Cílem této práce bylo řízení modelu robotického hada v neznámém prostředí s náhodně generovanými překážkami. Robotický had je tvořen devíti segmenty. Ty jsou sériově spojeny aktuátory, pomocí kterých se mění úhel natočení mezi sousedními segmenty. Otestování funkčnosti proběhlo v co-simulaci mezi MATLAB Simulinkem a MSC Adamsem. V testovací úloze se had musel proplazit mezerou mezi dvěma stěnami a bez kolize dosáhnout cíle. Byla sledována závislost chování hada na nastavení parametrů pohybu.

ABSTRACT

The goal of this work was control of a robotic snake model in an unknown environment with randomly generated obstacles. The robotic snake is made of nine segments. These segments are serially connected with actuators, which are used to change the angle between neighboring segments. Its functionality was tested in a co-simulation between MATLAB Simulink and MSC Adams. The snake's task was to slither through a gap between two walls and reach the target without a collision. The correlation between the snake's behavior and the set movement parameters was observed.

KLÍČOVÁ SLOVA

Robotický had, biologicky inspirovaná robotika, dynamika více těles, algoritmus hledání nejkratší cesty, řízení pohybu, simulace, co-simulace.

KEYWORDS

Robotic snake, biologically inspired robotics, multi-body dynamics, shortest path algorithm, motion control, simulation, co-simulation.

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

LEVEK, Martin. *Řízení modelu robotického hada* [online]. Brno, 2022 [cit. 2022-05-20]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/139072. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Tomáš Hůlka.

PODĚKOVÁNÍ

Děkuji svému vedoucímu práce, panu Ing. Tomáši Hůlkovi, za pomoc při psaní této diplomové práce a práce bakalářské, na kterou tato navazuje.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 ……

Martin Levek

OBSAH

1 ÚVOD

V oblasti robotiky se inženýři často inspirují biologií. Robotičtí hadi, tvoření segmenty, sériově spojenými aktuátory, jsou jedním z mnoha typů biologicky inspirovaných robotů. Hadi, díky své unikátní tělesné stavbě, vyvinuli velmi adaptivní způsoby pohybu. Na rozdíl od mnoha ostatních organismů, nemají žádné končetiny, což jim nebrání v překonávání překážek, právě naopak. Jsou schopni pohybu v obtížných terénech, úzkých prostorech nebo plavání. Rozložení váhy po celé délce těla jim umožnuje pohyb po měkkém povrchu. Díky flexibilnímu tělu se můžou pohybovat po větvích či sloupech, nebo manipulovat s objekty.

Z inženýrského pohledu mají také následující výhody. Tělo tvořené segmenty je redundantní, robot je tedy schopen pohybu i v případě závady [1]. Klouby nemusí být v kontaktu s okolím, je tedy možné dosáhnout vzduchotěsnosti, což umožňuje pohyb kapalinami nebo znečištěným prostředím [2].

Roboti tohoto typu, díky své flexibilitě, mohou potenciálně najít využití v mnoha situacích – pro inspekci obtížně dosažitelných míst, jako jsou trubky, k průzkumu, jak pro vojenské, tak civilní aplikace, u záchranných misí, například při prohledávání trosek po zemětřesení nebo ve zdravotnictví [3].

V některých případech postačí řízení manuální, jindy je vhodné, aby byl robot částečně nebo plně autonomní. Manuálně řízení roboti mohou být používáni v různých situacích, například na místech, kde člověk nemůže být fyzicky přítomný, ať už důsledkem nedostatku prostoru, nebezpečného prostředí nebo jiných důvodů. Příkladem jsou roboty asistované chirurgické operace, kde je robot využit pro svou přesnost. Chirurg se navíc nemusí nacházet přímo na místě operace [4]. Při operaci může jít o vteřiny, proto může schopnost řízení operačního robota z jiné lokace, místo převozu pacienta, zachraňovat životy. Robotické ramena v podobě hada jsou využívána v endoskopii [3].

Autonomní roboti, vybaveni senzory a řízeni umělou inteligencí, mohou být využiti z důvodu ušetření nejen fyzické, ale i mentální práce člověka. Jeden operátor je schopen dohlížet na více robotů. Zatímco ovládání pohybu robota na čtyřech kolech je relativně jednoduché manuálně, komplikovanější typy pohybu, jako jsou pohyby s končetinami nebo plazivé pohyby robotických hadů, jsou obtížné, ne-li nemožné ovládat ručně.

Tato práce se zabývá řízením modelu robotického hada. Navazuje na bakalářskou práci s názvem Návrh modelu robotického hada [1]. Byl zde zkoumán mechanismus pohybu hada, konkrétně pohyb bočním vlněním po rovné podložce. Byl simulován v prostředí MSC Adams, kde byla dokázána nutnost koleček, které nahrazují funkci šupin, demonstrován pohyb rovně a se zatáčením, vliv poruchy aktuátoru a byla provedena podrobná analýza pohybu (rychlosti, zrychlení…).

Získané vědomosti a 3D model (s jistými modifikacemi) jsou využity k řízení pohybu robotického hada v neznámém prostředí s rovnou podložkou a náhodně generovanými překážkami. Cesta mezi překážkami je nalezena s pomocí metody hledání

nejkratší cesty – algoritmu A*. Had se pohybuje mezi body této cesty. Princip řízení zatáčení spočívá v minimalizaci rozdílu mezi požadovaným a skutečným směrem pohybu. Řízení probíhá v MATLAB Simulinku, jde tedy o co-simulaci s Adamsem, ve kterém se simulují fyzické interakce.

2 TEORETICKÉ ZÁKLADY

Řízení modelu robotického hada kombinuje znalosti z mnoha oborů. Už jen z názvu vyplývá několik z těchto oborů. Prvním je řízení (automatizace, regulace). Dalším je vědecké modelování neboli simulace. Kombinace robotiky a biologie, také nazývaná biologicky inspirovaná robotika. Dalším oborem je například optimalizace – specificky algoritmy hledání nejkratší cesty.

2.1 Biologicky inspirovaná robotika

Jedním z cílů inženýrů je stavět lepší roboty, kteří jsou co nejlépe adaptovaní k práci v jejich prostředí. Skutečná zvířata a rostliny jsou díky milionům let evoluce velmi dobře přizpůsobena životu na Zemi. Proto se v robotice často inspirujeme tělesnou stavbou, chováním nebo organizací nervového systému skutečných živočichů. Nesnažíme se nutně o co nejdokonalejší kopii biologie (Obr. 1). Bereme na vědomí, jaká úroveň realismu je nutná k získání optimálních výsledků – od mlhavé podobnosti, po co nejpřesnější emulaci (alespoň takovou, jakou technologie a finance dovolí). Biologie je tedy využita jako počáteční bod, následně ale může být ignorována v prospěch funkčnosti [5, 6]. V některých případech se snažíme dosáhnout maximální věrohodnosti (alespoň na pohled) – například u lidských robotů, kde musíme překonat takzvaný „uncanny valley“, který popisuje vztah mezi podobností robota člověku a emocionální reakcí, kterou v lidech vyvolává [7].

Kromě robotiky je biologická inspirace zjevná i v jiných oborech, například u umělých neuronových sítí nebo optimalizačních algoritmů (genetický algoritmus [8], optimalizace mravenčí kolonií [9]…).

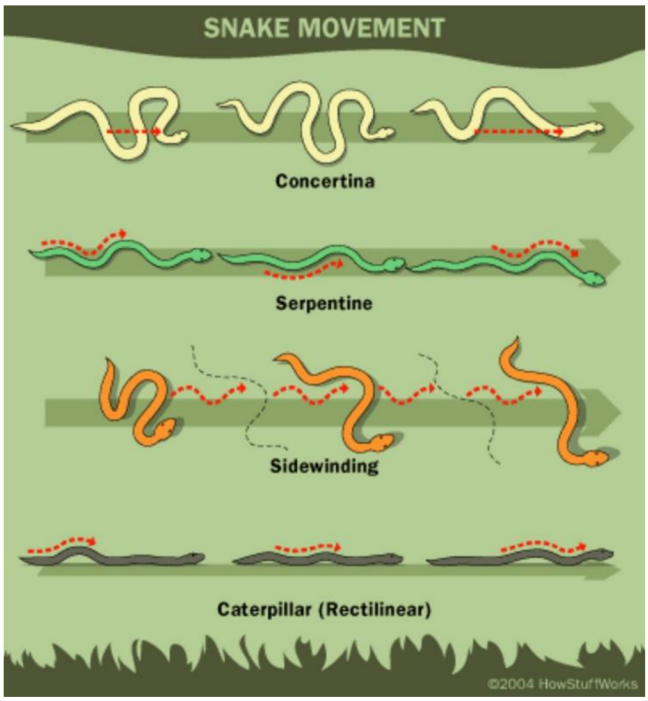
Obr. 1: Robotický had [10]

2.2 Pohyb hada

Hadi jsou charakterističtí svou unikátní tělesnou stavbou – konkrétně tím, že nemají nohy. Díky tomuto faktu se u nich vyvinulo několik metod pohybu. Podle hada a situace ve které se nalezne, volí vhodný způsob pohybu (nebo jejich kombinaci). Pokud jde o hadí biologii, v této práci je omezena pouze na části relevantní k imitaci jejich pohybu (Obr. 2) [11].

Nejpodstatnější částí hadí kostry jsou obratle. Existují dva typy – tělní a ocasní, rozdíl je v tom, že ocasní nemají žebra. Obratle jsou spojeny takovým způsobem, že téměř znemožňují kroucení a zároveň umožňují velký stupeň ohebnosti. Funkci obratlů budou v našem modelu plnit segmenty, ze kterých je robotický had sestaven. Samozřejmě jich nebude zdaleka tolik, biologický had může mít až 600 obratlů [11].

Další zásadní částí hada, umožňující jeho pohyb, jsou šupiny. Díky jejich tvaru a umístění na těle (rovnoběžně se směrem pohybu) způsobují, že had má rozdílné koeficienty tření v jiných směrech. A to – nízké tření při pohybu vpřed a vyšší při pohybu do stran a dozadu [1, 12].



Obr. 2: Pohyb hada [13]

2.2.1 Boční vlnění (serpentine movement / lateral undulation)

Boční vlnění, také známé pod anglickými názvy jako serpentine movement nebo lateral undulation. Je nejběžnějším způsobem hadího pohybu, který si většina lidí představí, když se mluví o hadech. Tento pohyb je použit v našem modelu. Had se pohybuje ve tvaru písmene S (Obr. 3). „Vlny“ pohybu, vytvořené stáhnutím svalů na vnitřní straně ohybu a natažením svalů na straně vnější, postupují od hlavy k ocasu [13, 14, 15, 16, 17].

Zatímco skutečný had je tento pohyb schopný zdokonalit adaptací na konkrétní povrch a odrážením se od překážek, jako kamenů a větví, náš model robotického hada využívá pouze tření povrchu. Koeficient tření povrchu je zásadní. Pokud je příliš nízký – had se pouze vlní na místě [1]. Některé jiné metody pohybu jsou potom vhodnější. Tato metoda je také použitelná nejen na souši, ale i ve vodě [14, 16].

Obr. 3: Boční vlnění [14]

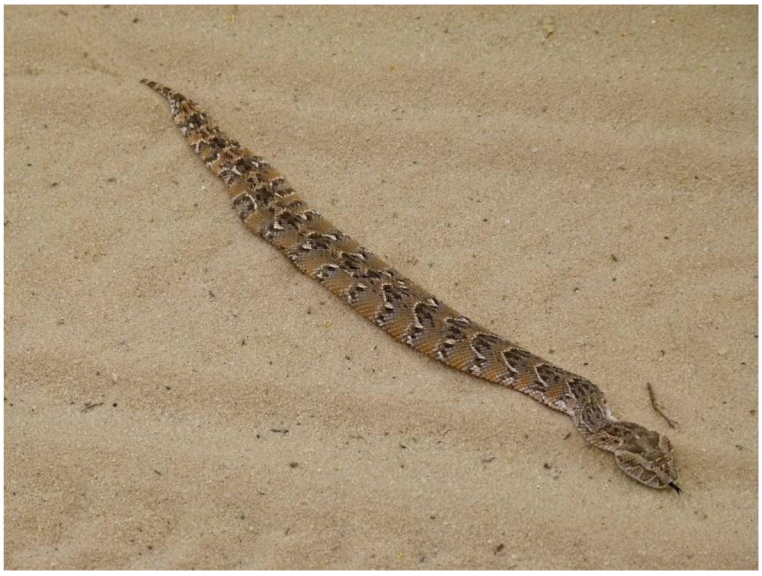
2.2.2 Kráčení do stran (sidewinding)

Podobná metodě bočního vlnění, je ale modifikovaná pro kluzké a sypké povrchy. Kromě vlnění do stran se had zároveň zvedá ze země. Navíc se v místech kontaktu s povrchem, na rozdíl od bočního vlnění, neklouže. Místo plazení by se tento pohyb dal popsat jako valení. Tato metoda pohybu zanechává velmi charakteristickou stopu (Obr. 4). Svůj název získala díky faktu, že se při použití této metody had pohybuje do strany (téměř kolmo) [13, 14, 15, 16, 17].

Obr. 4: Kráčení do stran [18]

2.2.3 Housenkovitý pohyb (rectilinear/caterpillar)

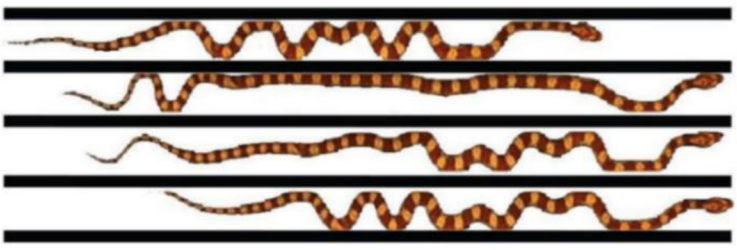
Jde o pomalý přímočarý pohyb (Obr. 5). Je preferovaným pohybem velkých hadů, protože neplýtvá energií. Využívá šupin na spodní straně těla. Šupiny se na některých místech odklopí směrem od těla, čímž se na nich had lehce nadzvedne. Tělo je tedy zvlněné ve vertikálním směru. Při jejich sklopení se posune směrem dopředu. Vzhledem k tomu, že se had nehýbe do stran, je tento pohyb užitečný v úzkých prostorech, kde pro jiné typy pohybu není dostatek místa. Je také vhodný pro plížení, protože je velmi tichý. Použití tohoto pohybu u robotického hada by bylo obzvláště obtížné, vzhledem k nutnosti ovládání jednotlivých šupin [13, 14, 15, 16, 17].



Obr. 5: Housenkovitý pohyb [19]

2.2.4 Harmonikový pohyb (concertina movement)

Tato metoda pohybu je pomalá, a ne příliš efektivní na horizontálním rovném povrchu. Je ale užitečná při pohybu vertikálním (například po stromech) nebo v tunelech (Obr. 6). Jak název napovídá, skládá se ze dvou kroků – natažením hlavy ve směru pohybu, kde had najde vhodné místo k uchycení, a přitáhnutím zadní části těla [13, 14, 15, 16, 17].



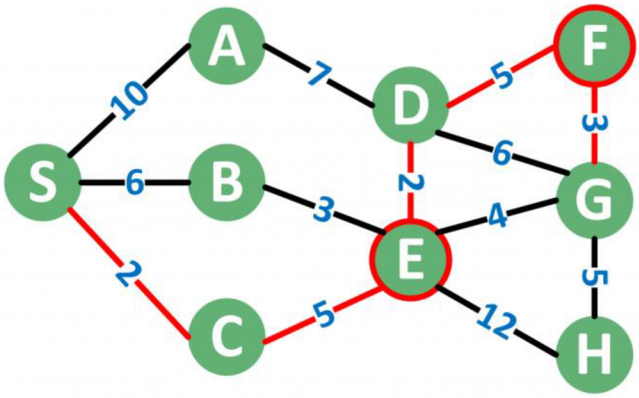
Obr. 6: Harmonikový pohyb [20]

Existují i další metody pohybu, navíc hadi často využívají kombinací několika metod zároveň. Adaptují se podle prostředí, povrchu, požadované rychlosti, snahy šetřit energií a tak dále. Někteří hadi dokonce dokážou plachtit nebo skákat [13, 14, 15].

Boční vlnění je jedinou z těchto čtyř metod pohybu, která nevyžaduje pohyb ve vertikálním směru. Vzhledem k tomu že náš model robotického hada [1] je schopen pouze pohybu ve směru horizontálním, byla využita tato metoda pohybu. Funkce šupin byla nahrazena kolečky, které se vlivem styku s podložkou při pohybu dopředu a dozadu otáčí a při pohybu do stran smýkají. Oproti šupinám jsou kolečka vhodnější pro couvání, které je pro skutečného hada obtížné.

2.3 Algoritmus hledání nejkratší cesty

Algoritmy hledání nejkratší cesty hledají cestu grafem od počátečního uzlu do uzlu cílového s minimální cenou. Uzly jsou spojeny hranami s určitými váhami, suma těchto vah je cena cesty (Obr. 7). Neexistuje jediný optimální algoritmus, schopný řešení všech variací problému nejkratší cesty. Některé algoritmy jsou přizpůsobené práci se statickým grafem, jiné pro práci s grafem dynamickým, některé využívají heuristiky k ušetření výpočetního času, nemusí tak ale nalézt optimální cestu [21, 22].



Obr. 7: Nejkratší cesta [23]

2.3.1 Algoritmus A*

Patří do kategorie best-first search algoritmů, které jsou charakteristické tím, že první prohledávají „nejslibnější“ uzly. Využívá heuristiku, díky tomu je výpočet poměrně rychlý. Algoritmus pracuje se dvěma seznamy – OPEN a CLOSED, do kterých zapisuje, které uzly plánuje prohledat a které už prohledány byly. Uzlům je vypočtena hodnota *f*. Ta je součtem ceny dosáhnutí uzlu – *g* a odhadem ceny k dokončení cesty – *h*. Cena může reprezentovat finanční náklady, vzdálenost, množství paliva… K odhadu ceny se může použít vzdušná vzdálenost, Manhattan norma a mnohé další, volba záleží na konkrétním případě. První jsou prozkoumány „nejslibnější“ uzly – s nízkou hodnotou *f*. Uzly také

obsahují informaci o svém rodičovi, aby po dosažení cíle bylo možné rekonstruovat cestu, kterou se do něj dostal. Algoritmus se řídí následujícím pseudokódem [24]:

1) Inicializovat list OPEN.
2) Inicializovat list CLOSED.
3) Vložit počátečního uzel do seznamu OPEN.
4) Dokud (while) list OPEN není prázdný:
    a. Vybrat uzel s nejmenší hodnotou f v seznamu OPEN.
    b. Odstranit tento uzel ze seznamu OPEN.
    c. Expandovat uzel, jeho potomkům nastavit expandovaný uzel jako rodiče.
    d. Pro (for) každého potomka:
        i. Pokud je potomek cílem – zastavit hledání.
        ii. Vypočítat potomkovi f ($f = g + h$).
        iii. Pokud se tento potomek již nachází v seznamu OPEN s nižší hodnotou f – přeskočit potomka.
        iv. Pokud se tento potomek již nachází v seznamu CLOSED s nižší hodnotou f – přeskočit potomka.
        v. Přidat potomka do seznamu OPEN (přepsat, pokud je již přítomný s vyšší hodnotou).
    e. Vložit uzel do seznamu CLOSED.
5) Rekonstruovat cestu pomocí rodičů.

Obr. 8: Algoritmus A* – zelený start, červený cíl, žlutá cesta, tmavě modré prozkoumané uzly, světle modré uzly ze seznamu OPEN, šedé překážky, černé uzly neprozkoumané [25]

Robotický had tento algoritmus využívá k hledání cesty mezi překážkami. Mapa je „rozsekána“ na čtverce, které jsou dále považovány za uzly grafu (Obr. 8). Graf je v modelu zapsán v podobě matice. Uzly jsou reprezentovány jednotlivými hodnotami této matice. Hrany mezi uzly nejsou definovány přímo. Do seznamu OPEN mohou být algoritmem přidány pouze hodnoty sousední, hrany se tedy nachází mezi sousedními uzly. Hodnoty v této matici se mohou rovnat nule nebo jedné. Hodnota jedna značí překážku, *f* tohoto uzlu tedy bude rovno nekonečnu a uzel nebude vybrán ze seznamu OPEN. Uzel bez překážky je značen nulou, při expanzi je tedy *g* zvýšeno o vzdálenost mezi rodičem a potomkem, *h* je vzdáleností mezi potomkem a cílem. Výsledná cesta, tvořená uzly, je přepočítána na souřadnice a využita pro řízení hada.

2.3.2 Varianty algoritmu A*

Výpočetní čas by se dal potenciálně ještě zkrátit využitím různých variant (případně jejich kombinací) algoritmu A*. Jednou z nich je jump point search (JPS). Tato varianta je použitelná pouze za podmínky, že graf má podobu mřížky (grid) s jednotnou cenou na hranách, což je v našem případě splněno. Rozdíl oproti dříve popsanému algoritmu A* je

v tom, že JPS není při expandování omezen pouze na sousední uzly. Může prozkoumat uzly v řádku, sloupci nebo na diagonále za sousedním uzlem (Obr. 9). Tímto „skokem“ (jump) se může vyhnout expandování mnoha uzlů, čímž se šetří výpočetní čas [25].

Obr. 9: JPS – zelený start, červený cíl, žlutá cesta, tmavě modré prozkoumané uzly, světle modré uzly ze seznamu OPEN, šedé překážky, černé uzly neprozkoumané [25]

Některé modifikace algoritmu, konkrétně způsobu výpočtu *f*, můžou podstatně zrychlit výpočet, ale za cenu nalezení neoptimální cesty. Tyto algoritmy jsou často využívány ve hrách, kde je nalezení „dostatečně dobrého výsledu“ v krátkém čase většinou důležitější než cesta optimální, nalezená v čase delším [26]. U řízení robota v reálném čase může být rychlost výpočtu také preferovaná.

Vážený A* (zkráceně WA*, jako weighted A*) počítá f jako $f = g + wh$, kde w je váhou s hodnotou větší než jedna. Čím je w bližší jedné, tím se výsledná cesta bude blížit cestě optimální, ale za cenu delšího výpočetního času. Váha nemusí být všude stejná, například u přístupu nazývaného dynamické vážení se upravuje podle hloubky hledání. Tato heuristika může být vhodná, když víme, že v průběhu cesty bude pravděpodobně cesta přepočítána [27].

WA* může být upraven na variantu nazývanou anytime weighted A* (zkráceně AWA*), kde se počítají postupně lepší a lepší cesty snižováním hodnoty *w*. Algoritmus může být kdykoli (anytime) přerušen, v takovém případě je rekonstruována cesta ze stavu výpočtu ve chvíli zastavení. Výsledkem je buď cesta optimální, pokud *w* dosáhlo jedné, nebo cesta neoptimální, u předčasného přerušení. Pro nalezení cesty optimální se plýtvá výpočetním časem, výhodou ale je, že pokud nastane situace „lepší něco než nic“, může být použito dosud nejlepší nalezené řešení [27].

Existuje velké množství variant s různými výhodami a nevýhodami, snažící se řešit nedostatky základního algoritmu A*, za cenu vytvoření nebo zhoršení problémů jiných. Jednou z nich je bidirectional A*, který hledá cestu nejen od startu do cíle, ale i z cíle do startu, po čemž se tyto dvě hledání potkají „někde mezi“ [27]. Dále existují například varianty šetřící paměť, jako simplified memory bounded A* (SMA*), která má pevně omezené využití paměti, na rozdíl od základního A*, u kterého využití paměti roste exponenciálně se složitostí úlohy [28]. Iterative deepening A* (IDA*) šetří paměť omezením maximální hloubky hledání a jejím iterativním prohlubováním [29]. Pro aplikaci v reálném čase byly vyvinuty algoritmy, jako například real-time A*, zkráceně RTA* (Korf, 1990), DTA* (Russell & Wefald, 1991), BPS (Hansson & Mayer, 1990) nebo k-best (Pemberton, 1995) [27]. Kromě variant algoritmů A* samozřejmě existuje mnoho dalších algoritmů, schopných řešení problémů hledání nejkratší cesty.

2.4 Multi-body dynamics

Systém dynamiky více těles (anglicky multi-body dynamics, zkráceně MBD, nebo flexible multi-body system, zkráceně FMS) obsahuje tělesa, která mohou být tuhé nebo pružné, a vazby, které omezují jejich vzájemný pohyb (snižují počet stupňů volnosti). U těles také může dojít ke kontaktu s jejich prostředím nebo s jinými tělesy [30]. Analýzou pomocí MBD se může studovat pohyb systému, na který působí síly a momenty. Jde tedy o dopřednou dynamiku, jejímž opakem je dynamika inverzní, která předem zná výsledný pohyb a snaží se zjistit, jaké síly (momenty) ho můžou způsobit [31]. Počítá se dynamická odezva na vnější síly, omezení a počáteční podmínky. Výsledkem je průběh pohybu, deformace a napětí. Inverzní dynamika je pomocí MBD řešitelná také [30].

Stupně volnosti popisují minimální počet parametrů, nutný k definování pozice a orientace entity (tělesa). Ve 3D prostoru je maximální počet stupňů volnosti šest – translační pohyb ve třech osách a rotační pohyb s třemi osami rotace, ve 2D jsou 3 – translační pohyb ve dvou osách a jeden pohyb rotační [32].

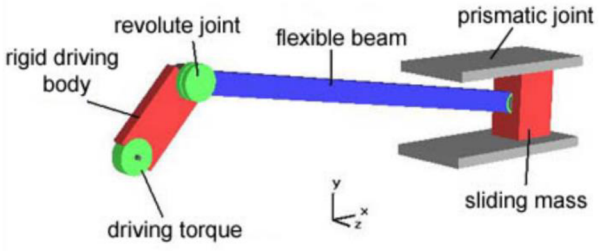
Tělesa jsou definována svou polohou, orientací v prostoru a tenzorem setrvačnosti, který obsahuje informace o hmotnosti a momentech setrvačnosti. Tenzor setrvačnosti je odvozen z kinetické energie a momentu hybnosti [33]. Tyto hodnoty musí být vztažené k určitému referenčnímu rámci. Tyto rámce mohou být plovoucí (floating), korotační (corotational) nebo inerciální (inertial, používané jako globální) [30].

Existuje několik druhů vazeb. Běžnými vazbami mezi komponenty jsou spoje otočné/rotační, kulové/sférické, hranolové/prismatické, rovinné nebo také vodící šrouby,

ozubená kola či vačky [30]. Každá vazba je charakteristická tím, jaké kinematické omezení tělesu přidává, nebo jinými slovy – kolik stupňů volnosti odebírá. Kinematické omezení (stupně volnosti) jsou relativní k jiným tělesům.

Zde jsou podrobněji vysvětlené některé z nejběžnějších vazeb (ve 3D). Rotační vazba umožňuje pouze jednu rotaci relativně k jinému tělesu, což znamená, že je odebráno pět stupňů volnosti. Hranolová vazba pohyb omezuje na translaci v jedné ose, relativně k jinému tělesu, stupňů volnosti je tedy odebráno pět. Kulová vazba znemožňuje translační pohyb v jednom bodě na tělese, relativně k tělesu jinému, povoluje ale všechny rotace, má tedy tři stupně volnosti [32].

Ve většině systémů je alespoň jedno těleso, jemuž jsou odebrány všechny stupně volnosti, takové těleso je potom „uzemněno“, v případě naší simulace je tímto tělesem podložka, po které se had pohybuje. Jiným příkladem je klikový mechanismus (Obr. 10), který má „uzemněný“ píst, tři rotační a jednu hranolovou vazbu. Celý klikový mechanismu má jeden stupeň volnosti.



Obr. 10: Model klikového mechanismu v MBD [32]

Systém je převeden na soustavu pohybových rovnic. Každé těleso je popsáno vlastní rovnicí pohybu a omezujícími podmínkami [32]. Při simulaci se tato soustava rovnic řeší po časových krocích.

Software využívající MBD je jedním z možných nástrojů pro analýzu dynamických systémů, obzvlášť pokud při simulaci dochází k velkým translačním či rotačním pohybům, jako například u klikového mechanismu (Obr. 10). Ve srovnání s jinými metodami, jako je třeba metoda konečných prvků, zkráceně MKP, je na úlohy tohoto typu MBD vhodnější. Pro výpočet je nutné programu zadat informace o tělesech. Jejich tvar, polohu a rozložení hmotnosti, ze kterých software sám vypočítá polohy těžišť, v případě nerovnoměrných hustot, tělesa navíc rozdělí do více menších těles. Působící zatížení (nebo požadovaný pohyb, v případě inverzní dynamiky), což mohou být síly, momenty, gravitační zrychlení a tak dále. Vazby, které mají vlastnosti jako tření, tuhost nebo tlumení. Kontakty, které je nutno nastavit, protože v MBD mezi sebou tělesa nekolidují, pokud kontakty nejsou definovány. A počáteční podmínky, jako rychlost [1].

Výsledky simulace mohou být využity při konstrukci a optimalizaci skutečných dynamických systémů, jako u: vozidel (automobily, vlaky, letadla…), robotů (biologicky inspirovaní robotičtí hadi [1], robotické manipulátory…) a mnoho dalších. Výsledky optimalizace mohou být například rozměry, konfigurace či materiály součástek systému, splňující omezující podmínky a minimalizující cenovou funkci. Další využití se nachází

v regulaci. A to díky schopnosti MBD předpovídat chování systému, způsobené změnou řídících veličin, nebo naopak, vypočítat potřebnou změnu řídících veličin, tak aby bylo dosaženo požadovaného výsledku [30].

K řešení řízení našeho modelu robotického hada byl použit software MSC Adams, který používá MBD [31]. Toto je jediným z mnoha možných způsobů řešení. Jinými jsou například dříve zmíněná MKP, nebo úplně jiný přístup, pomocí konformní geometrické algebry (CGA) [34].

MBD model je možné kombinovat s jinými výpočetními metodami, jako je MKP, která je kromě dynamiky schopna řešit deformace, přenos tepla, akustiku, optiku, elektromagnetismus a tak dále. Funguje tak, že rozdělí složitý problém na menší prvky (elementy), u kterých je řešení poměrně jednoduché. S vyšším počtem prvků se za cenu výpočetního času zvyšuje přesnost výpočtu [35].

3 VLASTNÍ ŘEŠENÍ

K simulaci fyzikálních interakcí byl využit modifikovaný model robotického hada z předešlé práce, který obsahuje devět segmentů, devět koleček a podložku [1]. Tyto simulace proběhly v MSC Adams. Řízení zatáčení bylo provedeno s pomocí zpětné vazby, která počítá rozdíl mezi požadovaným a skutečným směrem pohybu. Řízení hada probíhalo v MATLAB Simulinku, šlo tedy o co-simulaci mezi dvěma programy. Výsledky co-simulací, s různým nastavením parametrů pohybu, byly porovnány a analyzovány. Na jejich základě bylo řízení dále vylepšeno a otestováno.

3.1 Použitý software

V rámci práce byla použita dvojice programů. Prvním je MATLAB, rozšířený Simulinkem, a MSC Adams. Tyto dva programy simulují pohyb a řízení hada společně, jde tedy o co-simulaci.

MATLAB je programovací a výpočetní platforma. Má vlastní programovací jazyk. Využívá se k analýze dat, výtvoru modelů a algoritmů. Jeho funkčnost se dá rozšířit pomocí toolboxů a Simulinku. Simulink nabízí prostředí s blokovými diagramy, které je vhodné pro simulaci dynamických systémů. Také je možné jeho propojení s dalším programem, který byl využit k simulaci pohybu robotického hada a jeho řízení – Adams od MCS Software. Byla použita, v době tvorby této práce, nejaktuálnější verze, což je MATLAB R2021b [36, 37].

MSC Adams je program pro řešení dynamiky více těles pomocí simulace. Můžeme v něm vytvořit 3D model, určit jeho fyzikální vlastnosti, omezit stupně volnosti, aplikovat síly a momenty. Se schopnostmi rozsáhlé analýzy výsledků simulace a řešení nelineárních dynamických systémů ve zlomku času oproti využití metody konečných prvků je ideální volbou pro tuto práci. Byla použita nejnovější verze Adams View Student Edition 2021.1. Jak název napovídá, jde o studentskou licenci, program má tedy omezenou funkčnost oproti verzi plné – je možné simulovat maximálně 20 těles, maximální počet segmentů našeho hada byl tedy omezen na 9 [38].

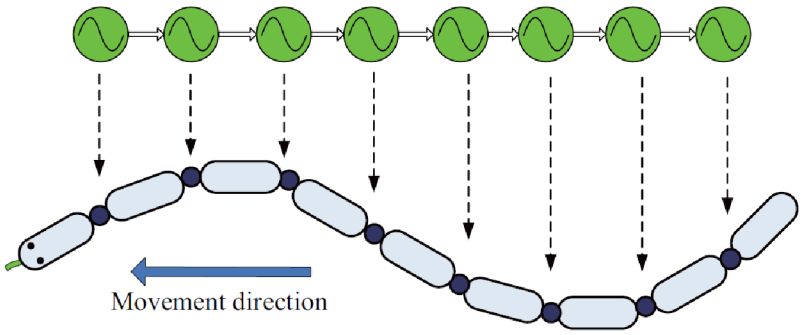
3.2 Pohyb robotického hada

Zatímco pohyb biologického hada je možný díky jeho svalům, náš robotický model je poněkud méně sofistikovaný (ačkoliv umělé svaly založené na elektroaktivních polymerech již existují [5]). Náš had je tvořen segmenty. Mezi páry sousedních segmentů jsou servomotory (reprezentované vazbami a pohyby v Adams modelu), řídící jejich vzájemné natočení. Každý segment má také kolečko, simulující funkci šupin skutečného hada. Tyto kolečka nejsou poháněná, točí se „na volno“, vlivem styku s povrchem. Stejně jako šupiny, způsobují že pohyb dopředu (a dozadu) vytváří menší tření než pohyb do

stran. Nutnost koleček byla dokázána simulací v předešlé práci, na kterou tato navazuje [1].

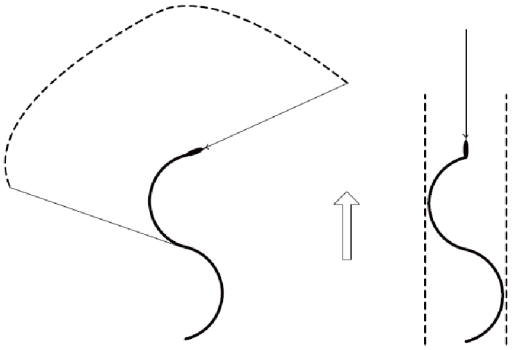
3.3 Řízení hada

Pohyb bočním vlněním je vytvářen pomocí řízení vzájemného natočení sousedních párů segmentů. Pro jednoduchý pohyb dopředu je použita funkce sinus, která je směrem k ocasu zpožděná (Obr. 11). *Amplituda* sinu určuje maximální natočení. Pro zatáčení se jednoduše přičte (nebo odečte) *bias*. To samozřejmě nemůže být provedeno skokově pro všechny segmenty zároveň. *Bias* se musí měnit postupně. Začne se u prvních dvou segmentů, a pokračuje se se zpožděním směrem k ocasu, tak aby zadní segmenty projely (přibližně) stejnou cestou. Vyhýbá se tak kolizím a skluzu (je snaha ho minimalizovat, nějaký skluz je nevyhnutelný).



Obr. 11: Řízení hada [39]

Problém unikátní pro řízení hada oproti mnoha jiným designům, je to, že není zcela zjevné jakým směrem je had natočený. U řízení auta stačí jeden pohled na polohu vozidla a natočení kol, a je známé, jakým směrem se bude pohybovat. U zvlněného těla hada musí být výpočet poněkud kreativnější. Není nutné se starat o všechny segmenty zároveň, pokud je řízení provedeno správně, zadní segmenty se pohybují skoro stejnou cestou jako segmenty před nimi, pouze s jistým zpožděním. Sleduje se tedy pouze směr segmentu prvního, je ale nutné provedení korekce. I při jednoduchém pohybu rovně se první segment natáčí o desítky stupňů. Tyto natočení však nejsou náhodné. Jde o periodickou funkci sinus. Je tedy vypočten úhel natočení prvního segmentu z jeho polohy. Následně je od něj odečtena hodnota pomocné funkce sinus, která má stejnou periodu jako funkce řídící otáčení segmentů. Tímto se kompenzuje vliv vlnění a je možné lépe řídit směr pohybu (Obr. 12) [39].



Obr. 12: Směr pohybu [39]

Dále je nutné vypočítat požadovaný směr pohybu. Z rozdílu požadovaného a skutečného směru pohybu je vypočítán *bias*, který řídí zatáčení, a směry tedy vyrovnává. Než je tento *bias* poslán zpětnou vazbou do natočení segmentů jsou ale ještě nutné dva kroky – saturace a zpomalení změny. Změna nemůže být okamžitá, vytvořila by tak nerealistický rychlý pohyb. Saturace udává maximální (a minimální) hodnotu *biasu*, což je nutné z konstrukčních důvodů a také proto, aby had nenarazil sám do sebe.

3.4 Generování a reprezentace překážek

Překážky nejsou v samotném 3D modelu. Model obsahuje pouze rovnou desku a robotického hada. Pro výpočet cesty není používán model přímo, místo toho je v MATLABu reprezentován mapou. Mapa je uložena ve formě matice. Na souřadnicích, kde nic není, je hodnota nulová, překážky mají hodnotu jedna. Tato mapa může být náhodně generována skriptem.

Souřadnice v mapě a v modelu nejsou stejné. Zatímco v modelu je poloha těles určena přesnými souřadnicemi, v mapě je reprezentována indexy v matici. Matice nemůže mít libovolnou velikost (jemnost), je omezena pamětí a výpočetním časem. Vzhledem k tomu, že v MATLABu se indexuje od jedné, je nutné vhodně zvolit posunutí mapy vůči skutečným souřadnicím. Pokud se například plánuje pohyb hada do záporných souřadnic, musí být index (1, 1) umístěn dostatečně daleko před nulou souřadnicového systému. Ačkoli by bylo možné mapu posunout za běhu programu, rozhodně by se tomu mělo snažit vyhnout. To že indexy jsou pouze kladná celá čísla také znamená, že při přechodu mezi skutečnými souřadnicemi a mapou musí proběhnout transformace. Pokud je vůči sobě posunutý počátek souřadnicového systému a index (1, 1) musí se posun přičíst (nebo odečíst, navíc může být posun ve dvou osách různý), také se musí násobit (nebo dělit) rozlišením. Pokud například jedna hodnota matice reprezentuje čtverec s délkou hrany sto milimetrů, musí se indexy vynásobit stem. Při přechodu ze souřadnic na indexy se samozřejmě musí zaokrouhlit na celé číslo.

Po doplnění překážek se přidá *padding*, tak aby cesta nebyla příliš blízko u překážek a vyhnulo se kolizím. Velikost *paddingu* se volí podle dvou faktorů – jak malou mezeru mezi hadem a překážkou je možné tolerovat a jaký je minimální rádius na kterém

je had schopen zatočit. Oba tyto faktory dále záleží na nastavení pohybu hada – *amplituda* vlnění a maximální (minimální, pro záporné hodnoty) *bias*, který řídí zatáčení.

Pokud jde o kontrolu kolizí, je možné srovnání skutečné cesty hada a souřadnic překážek pomocí skriptu, případně mohou být cesty a překážky vykresleny.

3.5 Rozdíly mezi simulací a skutečnou aplikací

Při modelování a simulování je snaha o zjednodušení reality – vypuštění nepodstatných faktorů a co možná nejpřesnější zachycení relevantních interakcí. Co to znamená pro tuto úlohu – jak se simulace liší od skutečné aplikace řízení robotického hada navrhovanou metodou? Kromě zjevné odpovědi, že probíhá ve virtuálním světě, ne reálném, je hlavní rozdíl v tom, že cesta není počítána v průběhu simulace. Díky vhodné volbě algoritmu hledání nejkratší cesty – variantou A*, je možné velmi dobře napodobit cestu kterou by had objevoval v reálném čase. Tento algoritmus obsahuje heruistiku, která má několik následků. Prvním je, že (až na triviální případy) většinou nenajde optimální cestu. Druhým následkem je podstatně rychlejší výpočet, což je zásadní, obzvláště u výpočtu v reálném čase. Třetím je fakt, že tato konkrétní heuristika způsobuje, že cesta vypočítaná se znalostí celé mapy je velmi podobná, ne-li stejná, jako cesta vypočítaná pomocí iterací s postupným odhalováním překážek, při přiblížení hada dostatečně blízko k detekci. Největším rozdílem těchto cest je nejspíš to, že při odhalovaní překážek by se had mohl dostat do slepé uličky. Takový případ by mohl vyžadovat, aby byl had schopný couvat. V případě simulace schopnost couvat implementována tedy není, protože by k této situaci nemělo dojít.

Proč tedy není hledání cesty provázeno v průběhu simulace? Hlavním důvodem je výpočetní čas. Při skutečné aplikaci neexistuje jiná možnost, než počítat v reálném čase (pokud mapu neznáme předem). Při každé nové detekci překážky by bylo nutno zkontrolovat, jestli neleží na cestě, pokud ano, došlo by k výpočtu cesty nové. V simulaci je ale možné provést výpočet pouze jednou. Navíc při simulaci není počítač zatěžován pouze řízením, ale také simulací fyzických interakcí. Při postupném vytváření modelu se prochází mnoha iteracemi, kdyby byla simulace příliš náročná, výpočetní čas by byl nepřijatelně dlouhý.

3.6 MATLAB skripty

Informace o MATLAB skriptech a o Simulinku jsou rozděleny do dvou kapitol. Zde jsou popsány skripty a funkce, které nejsou používány v průběhu co-simulace.

Mapa s náhodnými překážkami je generována pomocí *generate_map.m*, *padding* je přidán skriptem *pad_map.m*. Případně je možné mapu vytvořit manuálním vyplněním matice vhodnými hodnotami. Tato mapa a souřadnice startu a cíle jsou vstupními argumenty do funkce *A_star.m*, která vrací cestu v podobě matice (2 sloupce pro souřadnice x a z).

Než je tato cesta použita v co-simulaci, mohou z ní být odstraněny některé body. Když je cesta rovná, například z indexu (1, 1) do (1, 10), není nutné aby cesta obsahovala všechny body mezi: (1, 2), (1, 3)… Právě naopak – když je cíl vzdálenější, řízení je „hladší“. Díky vlnění hlavy kolmo ke směru pohybu hada se úhel požadovaného směru pohybu mění také. Tento efekt je minimalizován na větší vzdálenosti. Je použita funkce *trim_path.m*, jejíž vstupem je celá cesta a výstupem cesta „ořezaná“. Dalším potenciálním způsobem odstranění tohoto problému by bylo přepočítání polohy hlavy hada – kompenzace bočního vlnění.

Potenciální problém s tímto „ořezáváním“ je, že když had nezatočí dostatečně rychle, může se dostat mimo cestu a pohybovat se podél ní. Pokud je nastaven dostatečně velký *padding*, tohle by neměl být problém. V případě že „ořezání“ použito není, může nastat jiný problém způsobený nedostatečně rychlým zatáčením. Pokud had „přejede“ bod na cestě, ale nedostane se dostatečně blízko na to, aby se tento bod považoval za dosažený, může se k němu pokusit vrátit, není-li tato situace ošetřena v programu, který hada řídí.

Získaná cestu je přepočtena z indexů do skutečných souřadnic, k tomu je použita funkce *transform.m*. Takto upravená cesta je vložena do bloku v Simulinku (popsáno v jedné z následujících kapitol) a co-simulace je připravena ke spuštění.

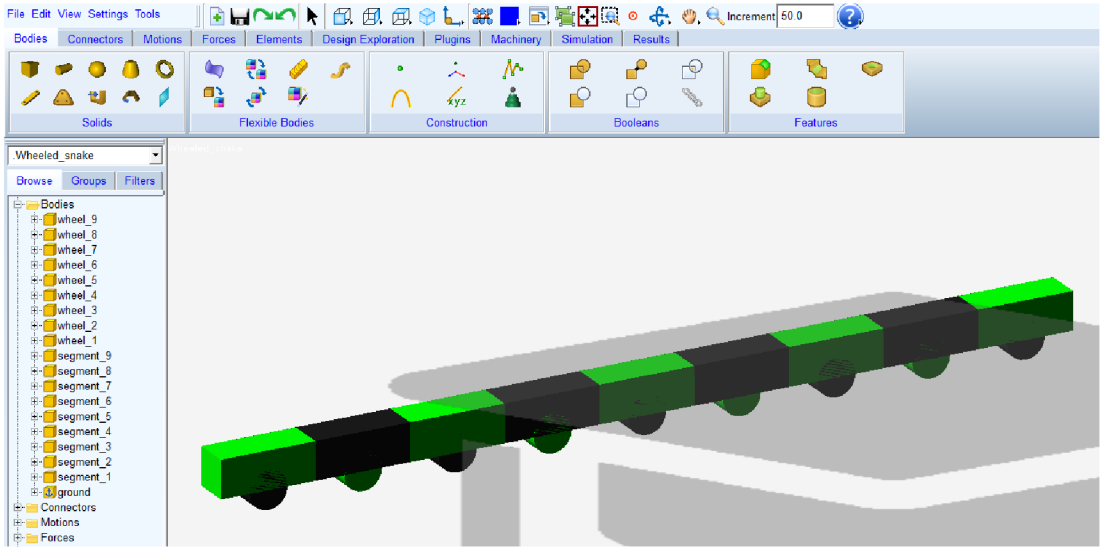
Zpětnou transformací *trans_back.m* je možné cestu převést zpět do indexů (s jistým zaokrouhlením). Pokud je takto přepočtena cesta získaná ze simulace, je možné použít funkci *check_collision.m*, která je schopna zjistit, jestli došlo ke kolizi s překážkou. Další možností je vykreslení cesty pomocí funkce *plot()*.

3.7 Popis Adams modelu

Co-simulace řízení modelu robotického hada probíhá ze Simulinku. Spojení těchto dvou programů funguje tak, že v Simulinku je používán blokový diagram, jako při normální simulaci. V Adamsu je vytvořen a exportován model. Ten je poté vložen do Simulinku ve formě bloku. Simulink a Adams jsou mezi sebou schopni komunikovat přes vstupy a výstupy tohoto bloku.

3.7.1 3D model

Model je možné vytvořit v Adamsu nebo ho importovat, po vymodelování v programech jako Inventor nebo SolidWorks. Tento relativně jednoduchý model, tvořený pouze kvádry (segmenty a podložka) a válci (kolečka), byl vytvořen přímo v Adamsu.



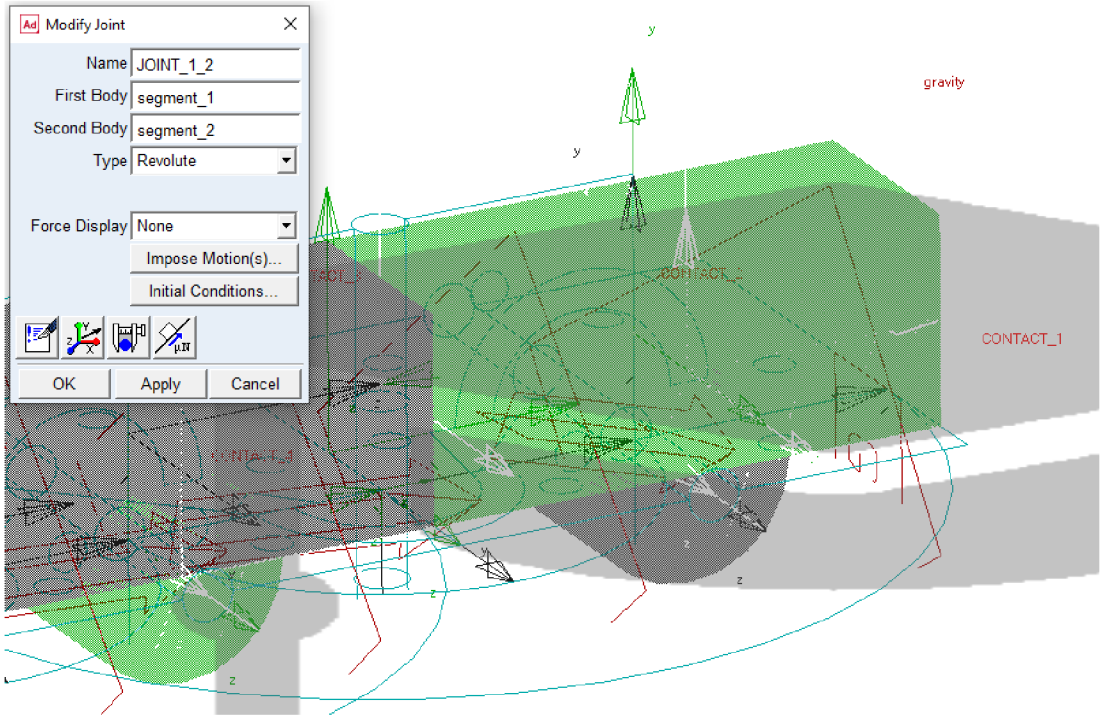
Obr. 13: Tělesa – 9 segmentů, 9 koleček a podložka

V záložce Bodies/Solids se použije RigidBody:Box a RigidBody:Cylinder (Obr. 13). Při vytváření těles se automaticky generují body, jejichž poloha je použita pro řízení hada a kontrolu kolizí. Tělesa se zobrazují v levém sloupci, kde můžou kliknutím pravého tlačítka myši být dále modifikována – změna materiálu (použit defaultní materiál – konstrukční ocel), barvy, rozměrů…

Vzhledem k omezení studentské verze programu, která je kromě názvu identifikovatelná také díky vodoznaku, je možné vytvoření maximálně dvaceti těles. Každý segment musí mít kolečko a je nutné vytvořit podložku. Proto je co-simulace omezena na 9 segmentů.

3.7.2 Vazby

Vazby omezují pohyby těles – snižují počet stupňů volnosti.

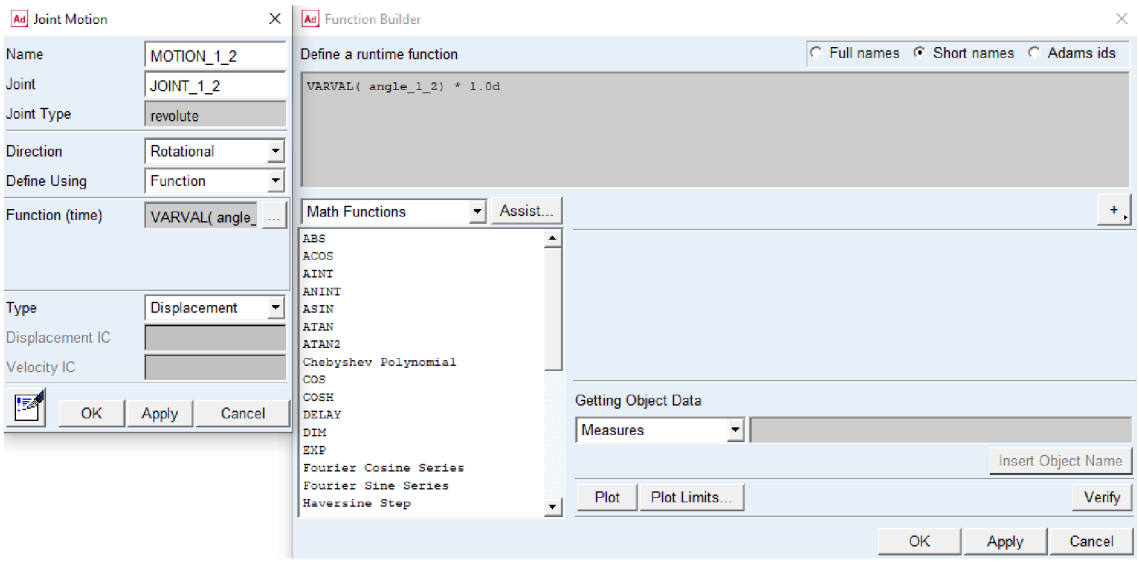


Obr. 14: Rotační vazby mezi sousedními segmenty a mezi segmenty a kolečky

Model obsahuje dva druhy vazeb (Obr. 14). Spojení sousedních segmentů a připojení koleček k jejich segmentům. Oba druhy jsou vazby rotační. Nacházejí se v záložce Connectors, s ikonou Create a Revolute joint. Vazby spojující sousední segmenty rotují kolem svislé osy (Y). Vazby koleček rotují kolem osy horizontální a zároveň kolmé vůči směru pohybu (ve výchozí pozici osa Z).

3.7.3 Pohyby

Stejně jako vazby, pohyby jsou pouze rotační. V záložce Motions jsou vytvářeny pomocí Rotational Joint Motions. Kolečka jsou roztáčena pouze díky kontaktu s podložkou, je tedy nutné pohyby definovat pouze mezi segmenty (na stejných místech jako rotační vazby).

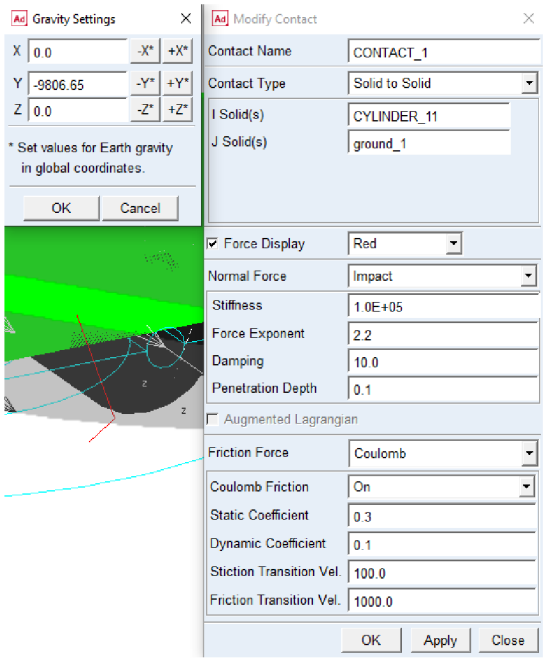


Obr. 15: Pohyby v rotačních vazbách mezi sousedními segmenty

Je vidět, že natočení mezi segmenty (na obrázku mezi prvním a druhým) je definované funkcí (Obr. 15). Stavová proměnná *angle_1_2* byla vytvořena v záložce Elements. Pomocí *VARVAL()* se zapisují stavové proměnné ve funkcích. Je násobena *1.0d*, což znamená 1°, natočení je tedy ve stupních. Tato proměnná je použita jako vstup ze Simulinku do Adamsu. Pomocí těchto úhlů (*angle_1_2*, *angle_2_3*,... *angle_8_9*) je řízeno natočení mezi segmenty. Výstupy z Adamsu do Simulinku jsou polohy bodů na segmentech. Z těchto výstupů je vypočítána zpětné vazba, kterou je had řízen.

3.7.4 Síly a kontakty

V Adamsu mezi sebou tělesa nekolidují, pokud jejich kontakt není definován (proto je možné hada modelovat jako jednoduché kvádry, které jsou umístěny těsně za sebou).

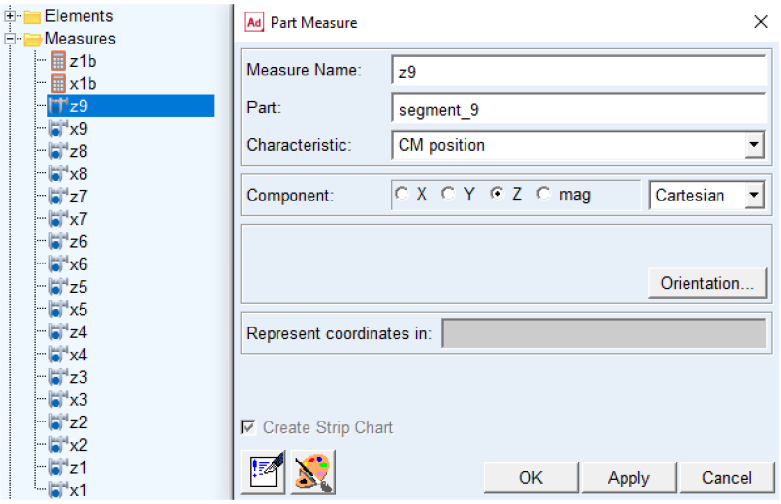


Obr. 16: Gravitační zrychlení a kontakty mezi kolečky a podložkou

V záložce Forces je použita položka Create a Contact k vytvoření kontaktů mezi kolečky a podložkou (Obr. 16). V této záložce se také dá vytvořit gravitační zrychlení, které je ale možné nastavit již při vytvoření nového modelu.

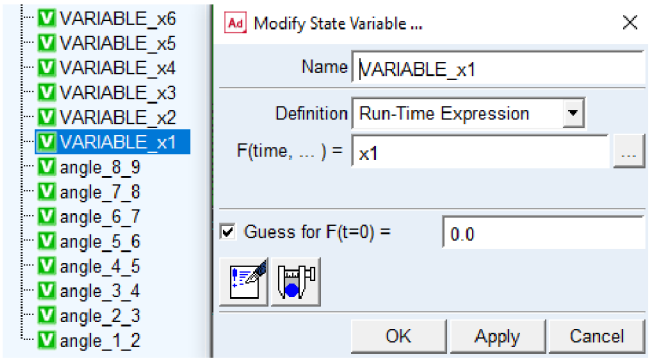
3.7.5 Stavové proměnné

Jak bylo dříve zmíněno – je za potřebí získat polohu segmentů jako výstup z Adams modelu. Výstupy musí být stavové proměnné, které jsou definované funkcemi. Do funkce ale není možné vložit polohu přímo. Nejdřív je nutné vytvořit měření.



Obr. 17: Měření polohy bodů na segmentech

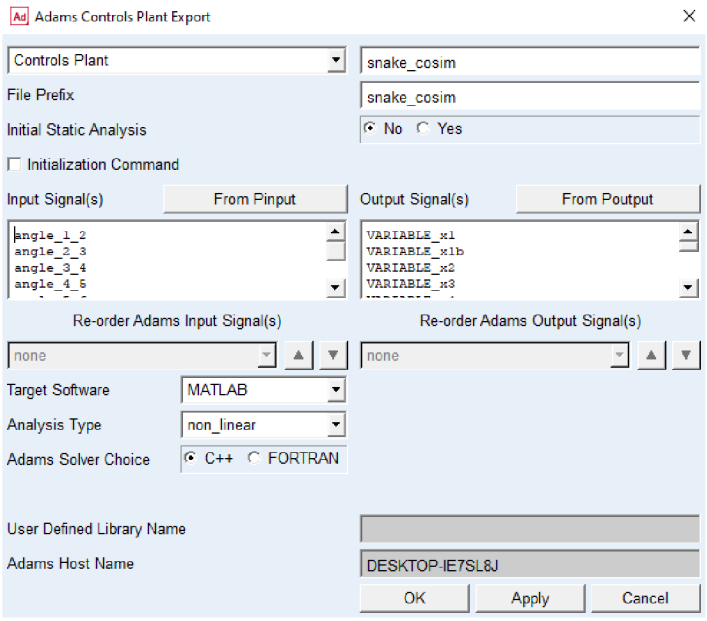
Kliknutím pravého tlačítka myši na těleso a volbou položky Measure, je možné vytvořit měření polohy bodu na tomto tělese (Obr. 17). Jsou sledovány pozice v ose X a Z (*x1*, *z1*,… *x9*, *z9*). Zvoleny jsou CM position – polohy středů hmotností pro všechny segmenty. U prvního segmentu jsou ale sledovány dva body (*x1*, *z1*, *x1b*, *z1b*) – z jejich poloh jsme schopni vypočítat vektor určující směr natočení prvního segmentu, který je zásadní informací pro řízení. Na výstup se tyto souřadnice dostávají skrze stavové proměnné (Obr. 18).



Obr. 18: Stavové proměnné

3.7.6 Propojení co-simulace

Hotový Adams model je možné exportovat přes záložku Plugins.



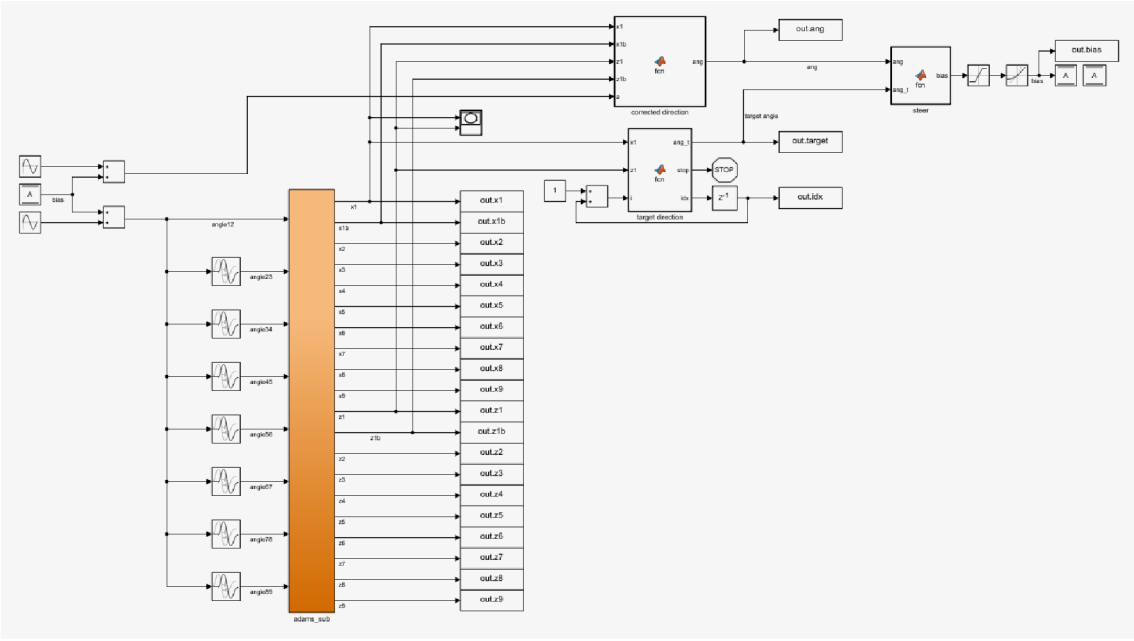
Obr. 19: Propojení co-simulace pomocí stavových proměnných

Byly vybrány vstupní (*angle_1_2*,… *angle_8_9*) a výstupní signály (*VARIABLE_x1*,… *VARIABLE_z9*), zvolen cílový software – MATLAB a model byl exportován (Obr. 19). V pracovní složce bylo vygenerováno několik nových souborů.

Propojení MATLABu a Adamsu je možné následujícími kroky: Spustíme MATLAB, ve kterém nastavíme pracovní složku, do které jsme exportovali. Do konzole napíšeme *adams_sys* a stiskneme enter. Otevře se nám blokové schéma v Simulinku, ve kterém zkopírujeme oranžový blok. Ten vložíme do vlastního blokového schématu, ze kterého co-simulace následně spouštíme.

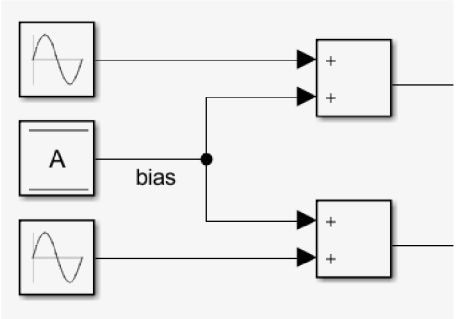
3.8 Simulink model

Na obrázku (Obr. 20) je vidět oranžový blok *adams_sub* s 8 vstupy (úhly natočení segmentů) a 20 výstupy (polohy bodů na segmentech). Tento blok reprezentuje Adams, ve kterém spolu se Simulinkem, probíhá co-simulace.



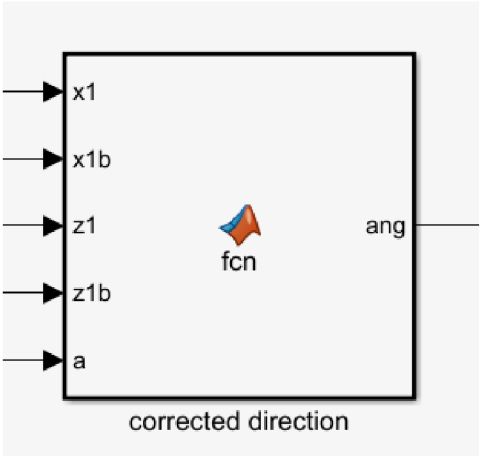
Obr. 20: Blokový diagram v Simulinku

V této kapitole je po částech vysvětlen celý blokový diagram, počínaje vstupem (Obr. 21).



Obr. 21: Vstup

Zde můžeme vidět dva sčítací bloky, do každého vstupuje jiná funkce sinus. Spodní sinus blok je po sečtení (a zpoždění) vstupem do bloku *adams_sub*. Je s ním řízeno natočení mezi segmenty. Horní sinus blok je jedním ze vstupů do funkčního bloku, ve kterém je vytvářena korekce směru natočení prvního segmentu. Jeho parametry byly určeny experimentálně. Paměťový blok A je využit jako zpětná vazba (není využita standartní zpětná vazba, aby se vyhnulo chybě: algebraic loop error). Jde o *bias*, kterým je řízeno zatáčení hada.



Obr. 22: Korekce

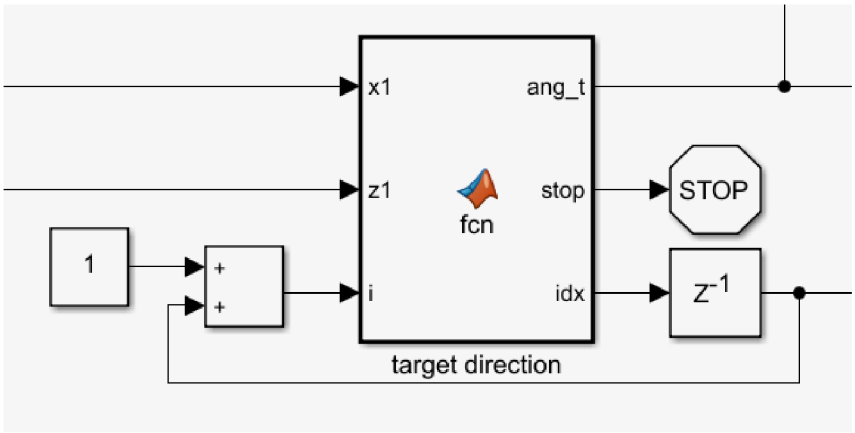
Funkční blok *corrected direction* (Obr. 22) má na vstupech souřadnice bodů z prvního segmentu a sinusovou funkci z (Obr. 21).

```
1   function ang = fcn(x1, x1b, z1, z1b, a)
2   % determines direction the snake is facing
3
4   xz = [x1 - x1b,  z1 - z1b];
5   ang = acos(dot(xz, [1, 0]) / norm(xz));
6
7   if xz(2) < 0
8       ang = 2*pi - ang;
9   end
10
11  ang = ang * 360 / (2 * pi);
12  ang = ang - a;
13
14  if ang > 360
15      ang = ang - 360;
16  elseif ang < 0
17      ang = ang + 360;
18  end
19  end
```

Obr. 23: Kód korekce

Výstupem je odhad úhlu (ve stupních) směru pohybu (Obr. 23). Tento úhel je počítán od osy X. Vzhledem k tomu, že Y je osa svislá, a používá se pravidlo pravé ruky, osa Z při pohledu z vrchu dolů směřuje k nám a osa X doprava. To znamená, že kladný úhel je po směru hodinových ručiček.

Zatáčení hada je řízeno na základě rozdílu mezi požadovaným se skutečným směrem pohybu. Když je směr pohybu správný, rozdíl je nulový a *bias*, řídící zatáčení je tedy také nulový. Požadovaný směr je určen v bloku *target direction* (Obr. 24).



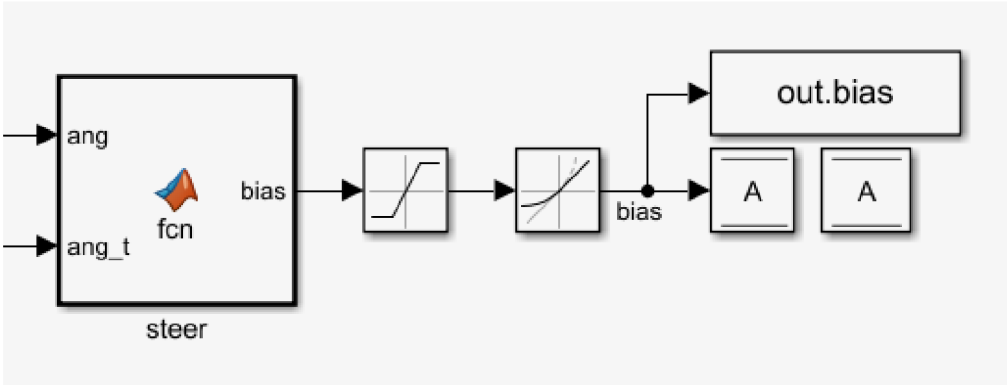
Obr. 24: Směr k cíli

Z obrázku je vidět, že kromě souřadnic prvního segmentu do bloku vstupuje také *i*, což značí index v matici, ve které jsou souřadnice cesty, která je uložena v samotném bloku, jako matice *path*. Zpětná vazba (se zpožděním jednoho kroku) na indexu funguje jako paměť. Suma s konstantou je nutná pro inicializaci.

```
function [ang_t, stop, idx] = fcn(x1, z1, i)
% determines angle to next path coordinates

path = [500,  0;
        500,  2000;
        1000, 2000;
        1000,-2000;
        1500,-2000;
        2000, 0];

xz = [path(i,1) - x1,  path(i,2) - z1];
ang_t = acos(dot(xz, [1, 0]) / norm(xz));
stop = 0;

if xz(2) < 0
    ang_t = 2*pi - ang_t;
end

ang_t = ang_t * 360 / (2 * pi);
idx = i - 1;

if norm(xz) < 150
    idx = idx + 1;
    sz = size(path);
    if idx == sz(1)
        stop = 1;
    end
end
end
```

Obr. 25: Kód směru k cíli

Prostřední výstup způsobuje okamžité ukončení co-simulace. To se stane, pokud je dosaženo posledních souřadnic cesty – cíle. Výstup *ang_t* je samozřejmě dříve zmíněný požadovaný směr pohybu (Obr. 25).



Obr. 26: Zatáčení

Když jsou známé oba úhly, je možné zjistit jejich rozdíl, což se provádí v bloku *steer* (Obr. 26). Také se zde zajišťuje, aby se had otáčel směrem menšího úhlu (Obr. 27).

```
1   function bias = fcn(ang, ang_t)
2   % determines bias for steering
3
4   bias = ang_t - ang;
5
6   if bias < -180
7       bias = bias + 360;
8   elseif bias > 180
9       bias = bias - 360;
10  end
11  end
```

Obr. 27: Kód zatáčení

Než je jeho výstup poslán zpětnou vazbou, musí být omezena maximální hodnota *biasu* a maximální rychlost jeho změny (*change*), aby se vyhnulo nerealistickým rychlým pohybům.

3.9 Co-simulace

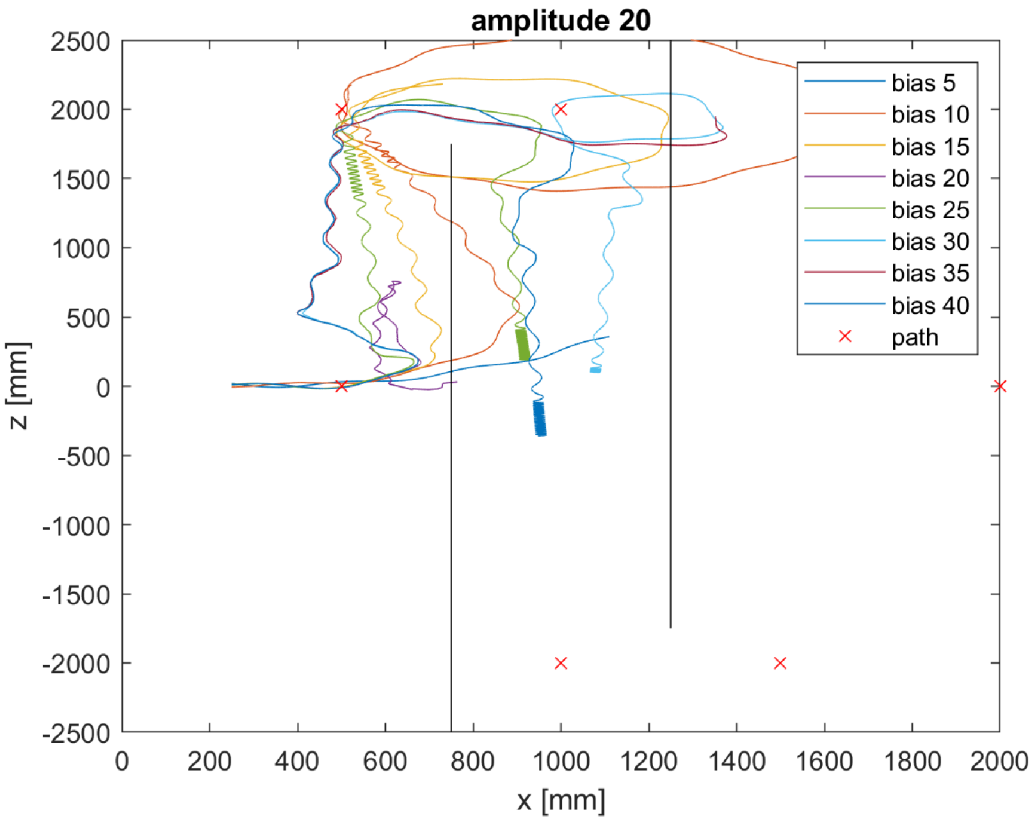
Před spuštěním je nutné nastavení parametrů (*variables.m*), vložení cesty do bloku *target direction*, vytvořené s pomocí *generate_map.m*, *pad_map.m*, *A_star.m*, *trim_path.m*, a *transform.m* (v tomto pořadí). Některé kroky mohou být vynechány nebo provedeny manuálně. Skript *snake_cosim.m* propojuje MATLAB a Adams, obsahuje cestu k pracovní složce a název počítače, pokud v těchto informacích dojde ke změně, je nutné je ve skriptu přepsat. Po otevření *control.slx*, se nastaví *stop time* simulace, a vše je nachystané pro spuštění. Je ještě možná volba dalších parametrů, jako například maximální délky kroku atd., v této práci je využité defaultní nastavení, není-li řečeno jinak.

Mapa použitá v následujících co-simulacích má dvě překážky – stěny ve směru kolmém k přímé cestě ze startu do cíle, mezi kterými had musí projet. Jsou od sebe vzdáleny půl metru. Po přiblížení se k první stěně a její detekci (pohyb po ose X) se musí had otočit do směru kladných hodnot Z a ujet přibližně dva metry. Poté se dostat mezi stěny, posunout se o čtyři metry (do záporného Z) a konečně zamířit k cíli, ležícím na ose X. Cesta obsahuje ostré zatáčky a úzké prostory, vhodné k otestování schopností hada.

Je mnoho parametrů, jejichž vliv na výsledný pohyb je možné sledovat. Jedny z nejvlivnějších jsou *amplituda* a *bias*. Dalšími parametry jsou *frekvence*, *zpoždění*, *rychlost změny*, *vzdálenost detekce*, *korekce*, *rozlišení* (mapy) a *padding*. Nemluvě o modelu v Adamsu, kde je dále možné měnit věci jako koeficient tření, materiál hada, počet a rozměry segmentů, počet, tvar a rozmístění koleček… Vzhledem k poměrně dlouhé době výpočtu simulace, jsme časově omezeni ve zkoumání vlivu všech těchto parametrů.

3.10 Výsledky co-simulace

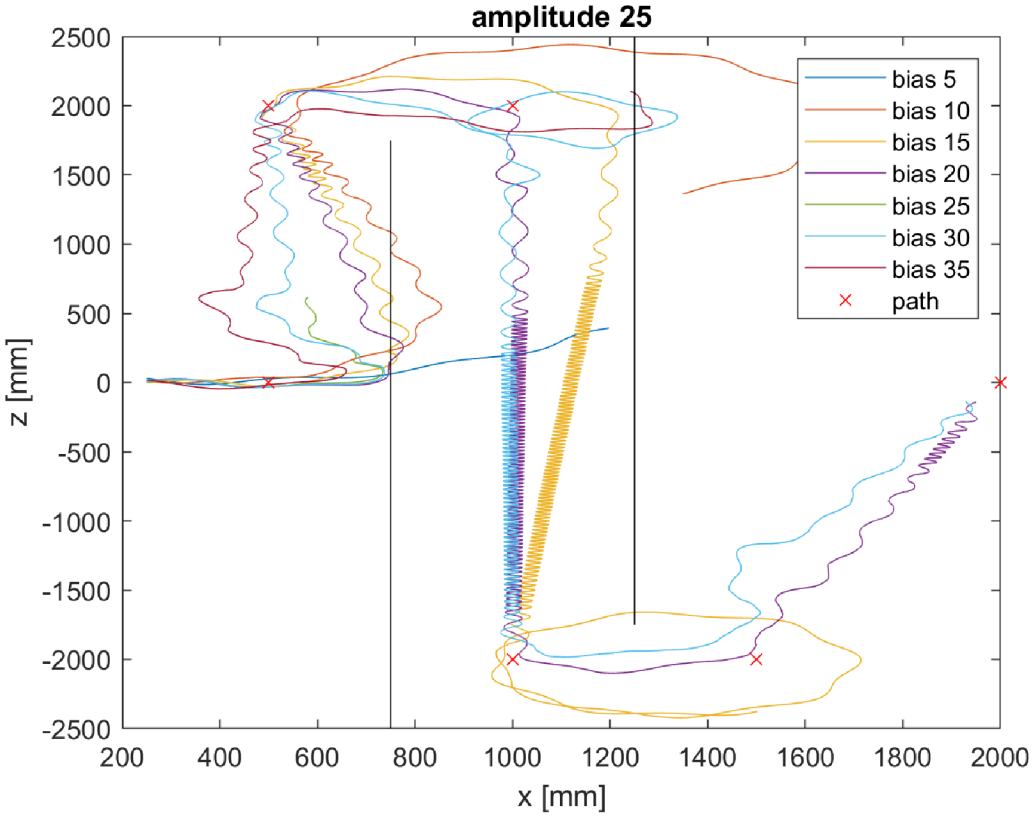
Byly srovnány různé kombinace *amplitudy* a *biasu*. Protože natočení sousedních segmentů nemůže být libovolně veliké, jejich maximální součet byl 60°. Od *amplitudy* 20° a *biasu* 5° do 40° a 20° po 5° krocích. V případě zjevného problému (kolize s překážkou, točení se v kruhu) byly některé co-simulace zastaveny předčasně z důvodu ušetření výpočetního času. Parametry co-simulací jsou zapsány následujícím způsobem – a<*amplituda*>b<*bias*>. Například *amplituda* 30° a *bias* 20° je a30b20.



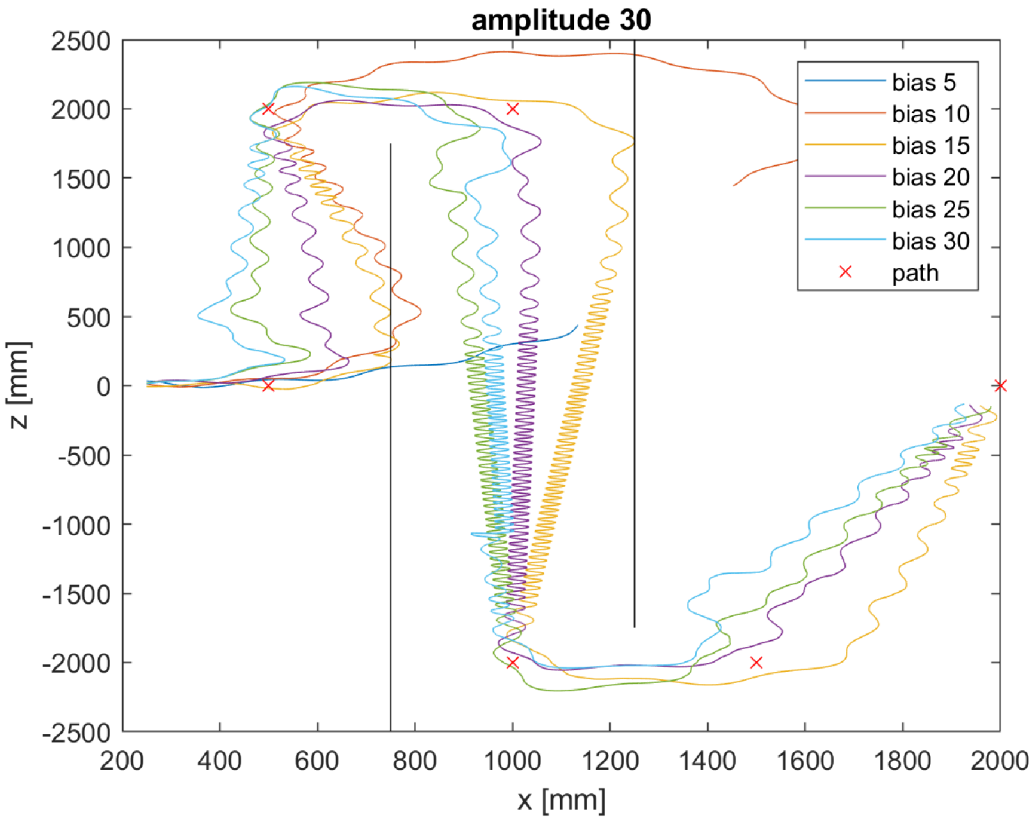
Obr. 28: Amplituda 20

Zde (Obr. 28) vidíme cesty s *amplitudou* 20°. Překážky jsou černé, body cesty, kterých se had postupně snaží dosáhnout, jsou vyznačeny červenými křížky. Co-simulace potřebuje alespoň 7 sekund na inicializaci (díky 7 zpožděním po 1 sekundě), sledujeme tedy její průběh až poté, co první segment dosáhne 250 mm na ose X. Protože různé parametry způsobují různou rychlost pohybu, musíme použít vzdálenost, ne čas.

Všechny co-simulace byly ukončeny před dosažením cíle. Cesty a20b25 a a20b40 měly šanci cíle úspěšně dosáhnout bez kolize, bohužel se na rovném úseku začaly pohybovat velmi pomalu (tento jev je vysvětlen později) a výpočty byly ukončeny předčasně. Ostatní nastavení *biasu* způsobily kolizi s překážkou a některé „přejely“ bod a dostaly se do smyčky. Jejich úspěch byl pravděpodobně velmi nejistý a na jiné mapě by nemusel být opakován. Žádné z těchto nastavení není vhodné.



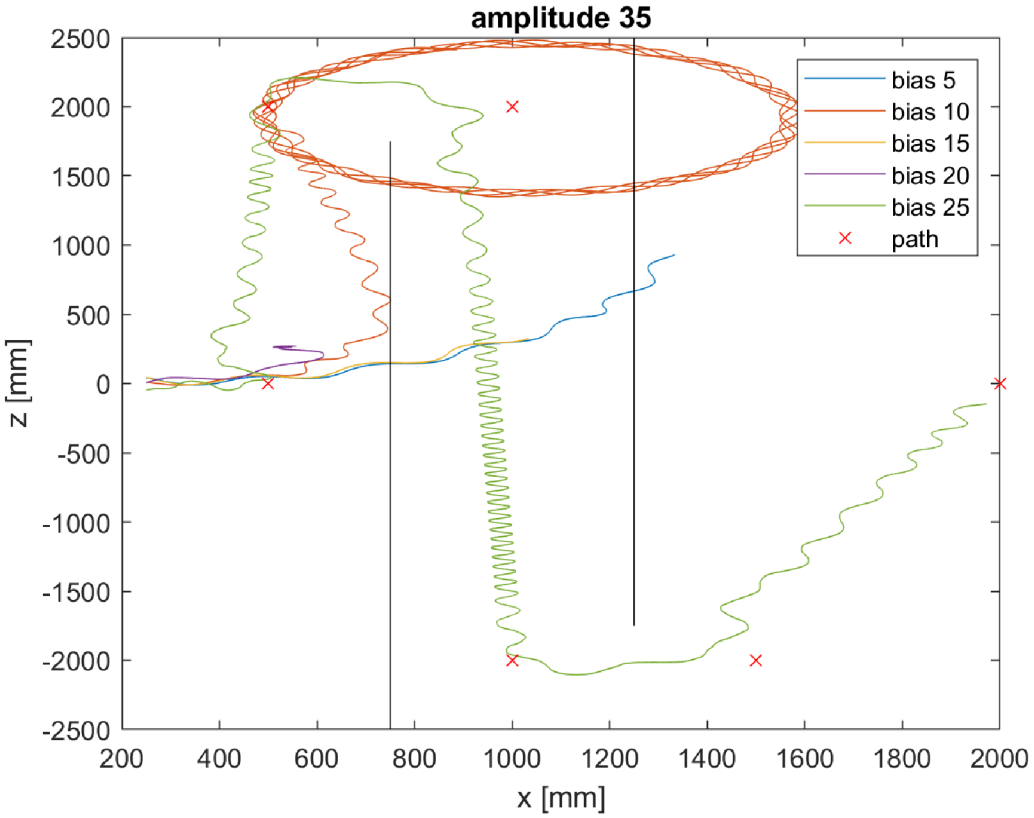
Obr. 29: Amplituda 25



Obr. 30: Amplituda 30

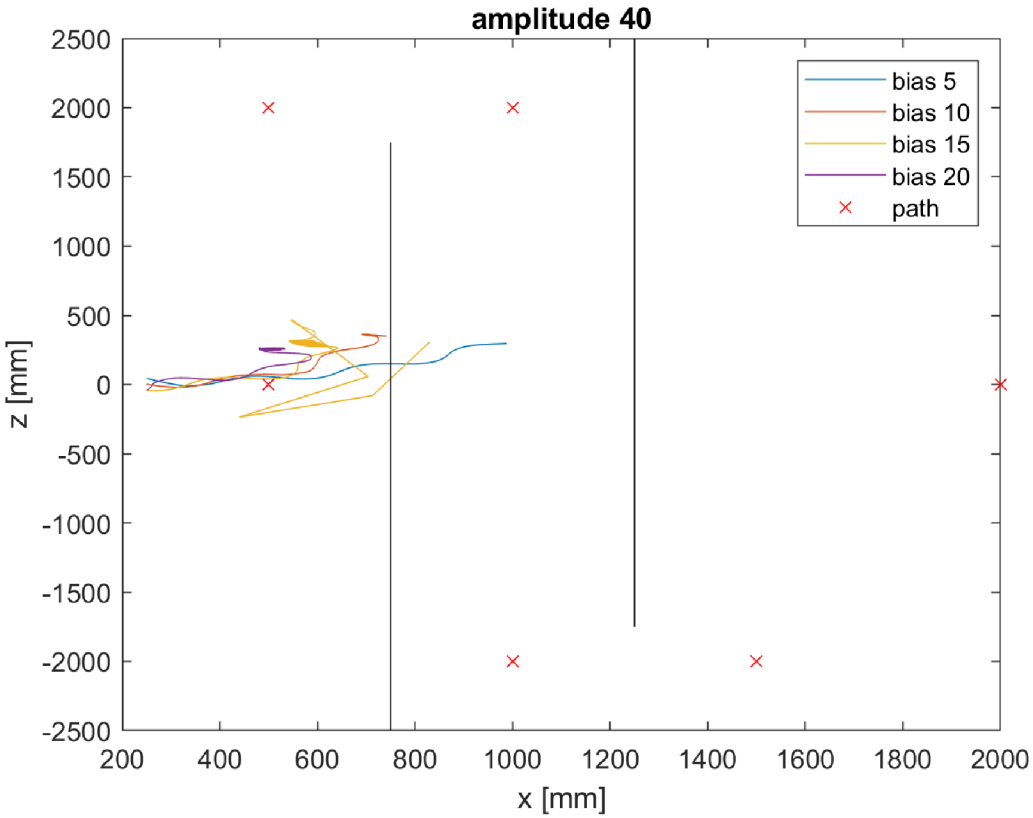
S *amplitudou* 25° cíle dosáhly a25b20 a a25b30 (Obr. 29), obě ale narazily do překážek. U a25b25 opakovaně docházelo k chybě ve stejném čase. Příčina této chyby nebyla odhalena. Žádné z těchto nastavení není vhodné.

U *amplitudy* 30° získáváme úspěšné dokončení cesty bez kolizí (Obr. 30) s nastavením a30b20, a30b25 a a30b30. Cesta a30b15 byla téměř bez kolizí, zatímco a30b5 a a30b15 nezatáčely zdaleka dostatečně rychle. Bohužel stále dochází k dramatickému zpomalení při pohybu rovně, i když méně než u nižší *amplitudy*.



Obr. 31: Amplituda 35

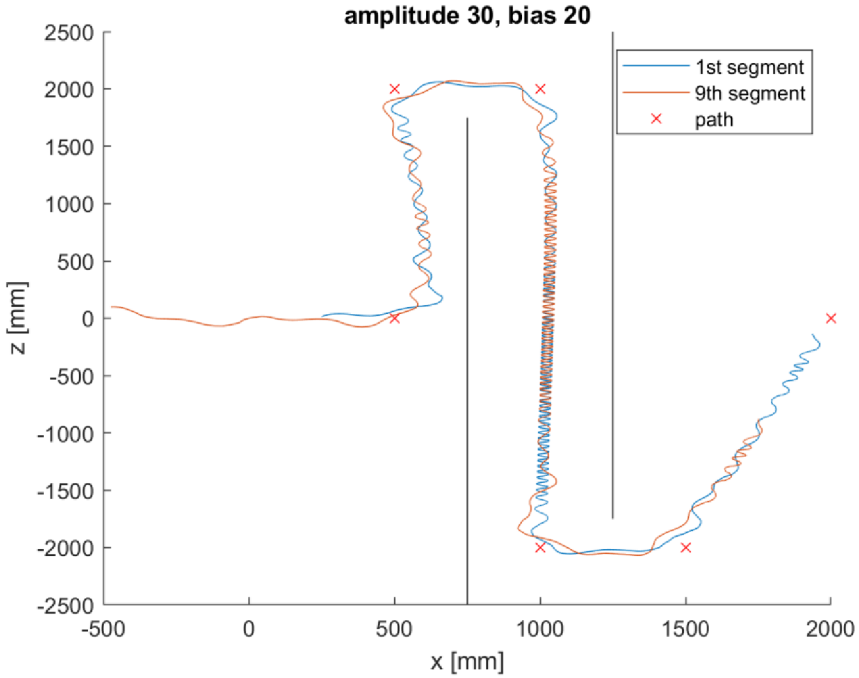
Se zvýšením *amplitudy* z 25°na 30° se výsledky zlepšily, na 35° se zhoršily (Obr. 31), což naznačuje že optimální nastavení se nachází kolem *amplitudy* 30°. Do cíle úspěšně dorazila pouze a35b25. U a35b20 se had začal smýkat na místě. V ostatních případech nezatáčely dostatečnou rychlostí.



Obr. 32: Amplituda 40

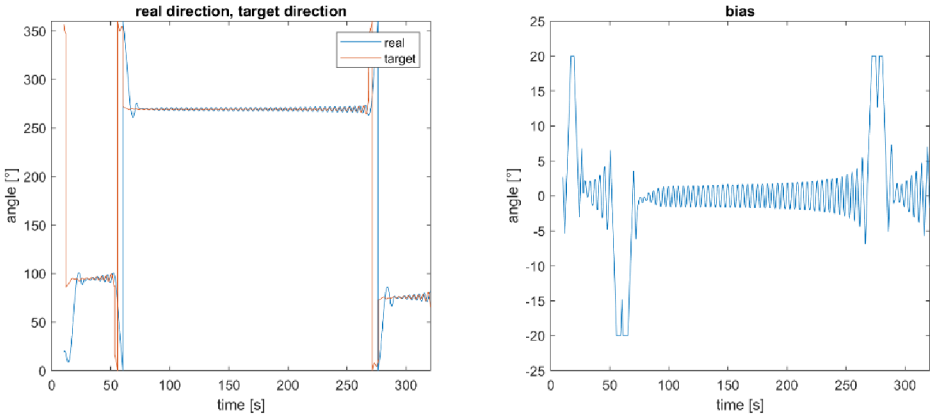
Je vidět, že *amplituda* 40° je příliš vysoká (Obr. 32). Žádný *bias* nebyl schopen úspěšně dosáhnout cíle. *Bias* 5° a 10° v žádné co-simulaci nedosáhnul ani druhého bodu. Zde došlo ke smýkání na místě a kolizi s překážkou, z důvodu příliš pomalého zatáčení. U a40b15 toto smýkání zjevně způsobilo chybu v co-simulaci. Není jisté, jak by se tyto nastavení chovaly u skutečného robotického hada, ale vzhledem k tomu, jak dobrých výsledků bylo dosaženo kolem *amplitudy* 30°, se s těmito parametry není nutné dále zabývat.

Podrobnější analýzou jednoho z úspěšných nastavení se došlo k vylepšení modelu – odstranění zpomalování na rovině. Pro tento účel byly zvoleny parametry a30b20.



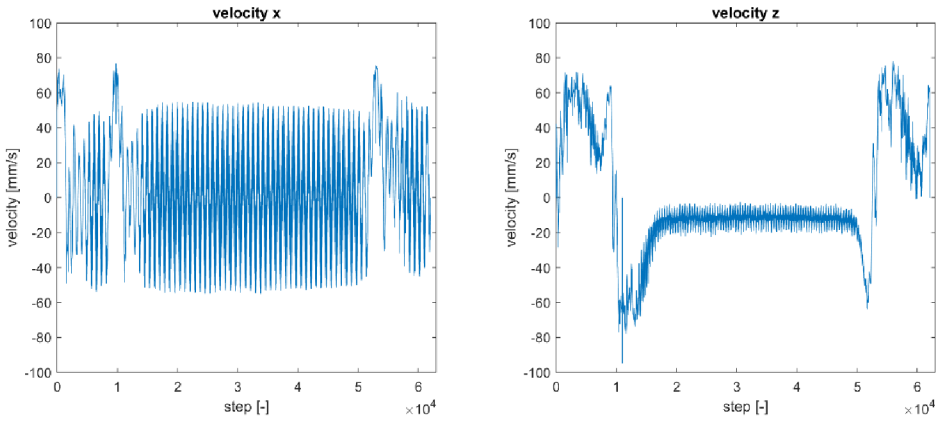
Obr. 33: Cesta a30b20

Nejprve je nutné se ujistit, zda nedošlo ke kolizi s jiným než prvním segmentem (Obr. 33). Je vidět, že segmenty mají různé cesty. Ideálně by následující segmenty projely stejnou cestou jako segment první, což bohužel není v praxi možné. Přesto, že se trajektorie liší, ke kolizi nedochází (nejsou vykresleny všechny segmenty, byly ale zkontrolovány).



Obr. 34: Zatáčení a30b20

Vlevo (Obr. 34) je vidět skutečný směr hada (po korekci) a požadovaný směr pohybu. Chování hada při zatáčení je v pořádku. Pohledem na průběh hodnot v čase mezi 75 a 265 sekundami (rovný úsek mezi překážkami) si můžeme všimnout, že hodnoty nejsou konstantní (na rovině by ideálně být měly). *Bias* (vpravo) se počítá z jejich rozdílu, což znamená, že také není konstantní. Toto je důvodem dříve zmíněného jevu, který pohyb rovně značně zpomaluje (Obr. 35).



Obr. 35: Rychlost a30b20

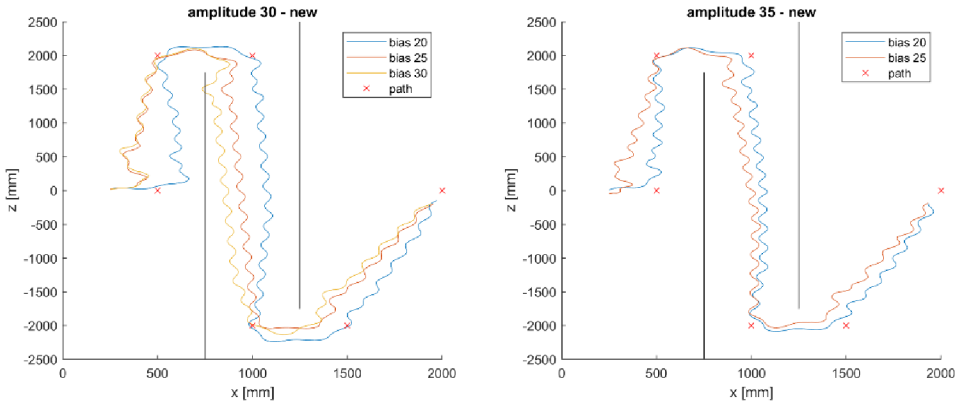
Dlouhý rovný úsek je ve směru osy Z, z rychlosti v ose X tedy mnoho užitečných informací získáno nebylo. Je vidět, že rychlost v ose Z dosahuje přibližně 60 mm/s, rychle ale padá pod 20 mm/s, jakmile se had dostal do rovného úseku.

Z těchto dat je zjevné, že je nutné *bias* udržovat na rovině (pokud možno) nulový. Není možné dosáhnout dokonalé korekce, místo toho byla provedena drobná úprava bloku *steer*, kde se počítá rozdíl mezi skutečným a požadovaným směrem.

```
1   function bias = fcn(ang, ang_t)
2   % determines bias for steering
3
4   bias = ang_t - ang;
5
6   if bias < -180
7       bias = bias + 360;
8   elseif bias > 180
9       bias = bias - 360;
10  end
11
12  if abs(bias) < 5
13      bias = 0;
14  end
15  end
```

Obr. 36: Nové zatáčení

Pokud je *bias* mezi -5° a 5°, změní se na nulu (Obr. 36). Tato jednoduchá podmínka (if, 12-14 řádek kódu) řeší problém zpomalování při pohybu rovně.



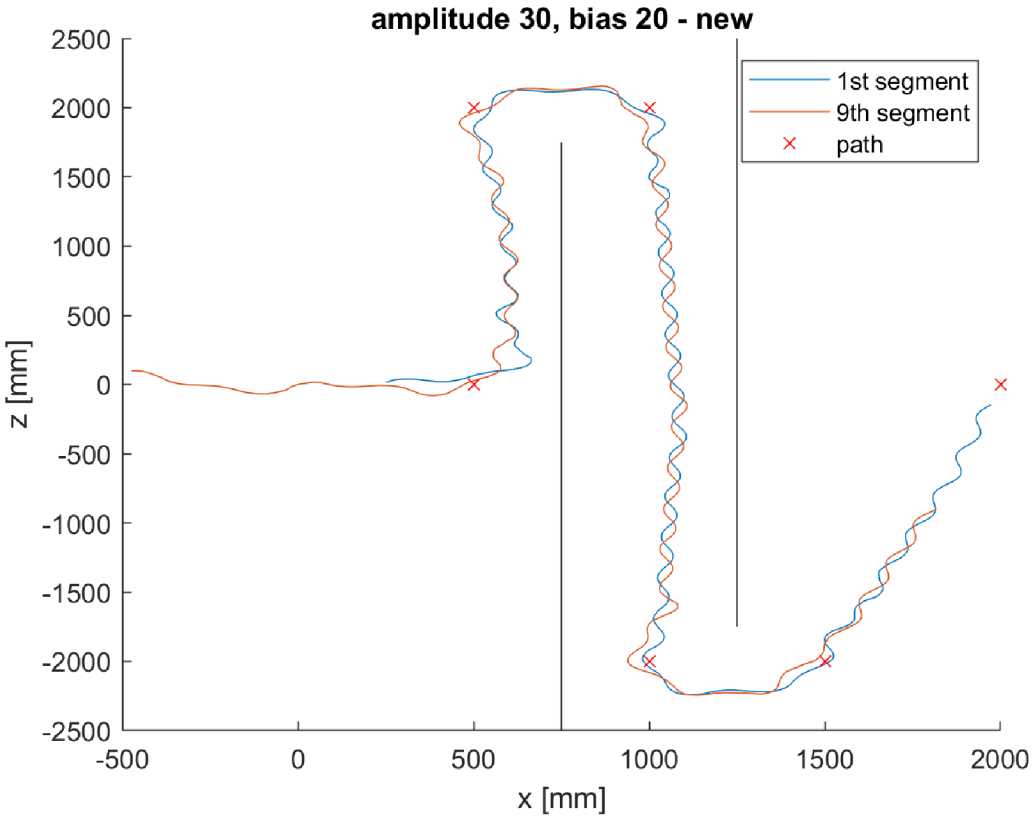
Obr. 37: Nové cesty amplitudy 30 a 35

U nastavení s nejlepšími výsledky byly provedeny nové co-simulace (Obr. 37) s upraveným blokem *steer*. Nové výsledky jsou zapisovány s koncovkou n. Až na a30b30n, který narazil do překážky, všechny úspěšně dosáhly cíle. Je vidět, že na rovině již nedochází ke zpomalení.

Parameters	Time [s]
a30b20	310
a30b25	291
a30b30	273
a35b25	261
a30b20n	144
a30b25n	132
a35b20n	143
a35b25n	147

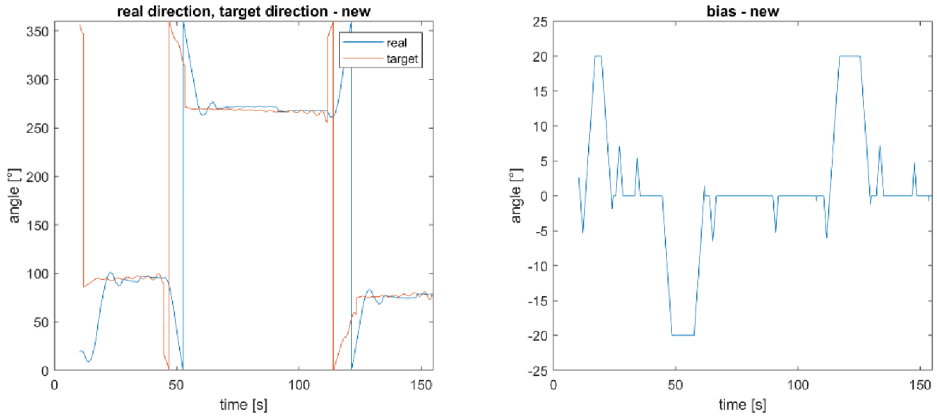
Tab. 1: Časy

Časy cesty jsou u nových co-simulací přibližně poloviční (Tab. 1). Opět byly vykresleny průběhy výstupů pro *amplitudu* 30 a *bias* 20 (Obr. 38).



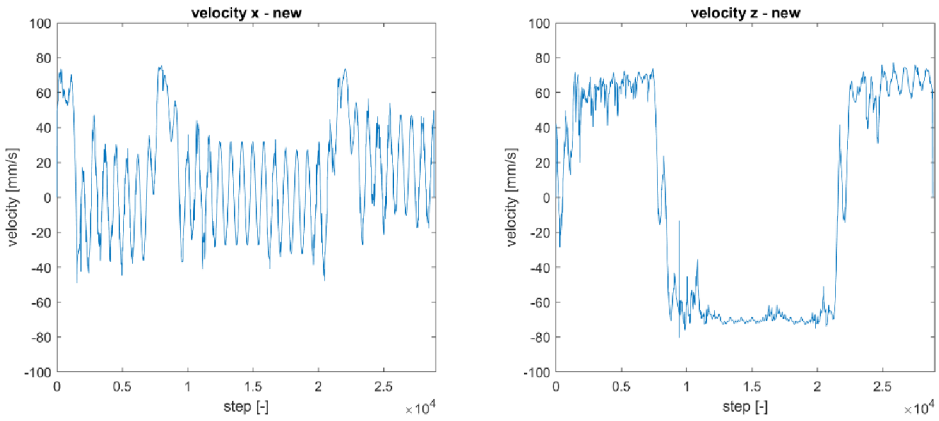
Obr. 38: Nová cesta a30b20

Nedochází ke kolizím s žádným z devíti segmentů. Na rovné části cesty je mezi 0 a -500 mm na ose Z vidět upravení směru pohybu po přesáhnutí odchylky 5°.



Obr. 39: Nové zatáčení a30b20

Průběhu *biasu* (Obr. 39) je v rovné části většinu času konstantní, na rozdíl od jeho průběhu u a30b20.



Obr. 40: Nová rychlost a30b20

Oproti a30b20 se u a30b20n (Obr. 40) rychlost v rovných úsecích udržuje kolem hodnoty 60 mm/s, zatímco před změnou padala pod 20 mm/s. Zdá se tedy, že úpravou bloku *steer* bylo dosaženo požadovaného efektu.

4 ZÁVĚR

Tato práce se zabývala řízením modelu robotického hada v neznámém prostředí s náhodně generovanými překážkami. K tomu bylo využito co-simulace mezi MATLAB Simulinkem, ve kterém probíhalo řízení, a MSC Adamsem, ve kterém byly simulovány fyzické interakce hada s prostředím.

Byl vytvořen mechanismus řízení zatáčení, fungující na principu minimalizace rozdílu mezi požadovaným a skutečným směrem pohybu. Had se pohybuje mezi body, tvořícími cestu, která je vypočítána algoritmem hledání nejkratší cesty – variantou A*.

Po realizaci třiceti co-simulací s různými parametry, byl přibližně určen vhodný rozsah nastavení parametrů. Také byl nalezen problém v modelu, způsobující zpomalení (až zastavení) při pohybu hada po rovině. Díky analýze průběhu co-simulace byl model vhodně upraven a problém odstraněn. Většina co-simulací (v novém modelu) s různými parametry úspěšně dosáhla cíle bez kolize s překážkami.

U a30b30n kolize s překážkou nastala, zatímco u ostatních velmi podobných nastavení, bylo cíle dosaženo úspěšně. Zdá se, že hraje roli, jak je had natočený ve chvíli detekce bodu. Pokud je natočen ve směru zatáčení k následujícímu bodu, může se stát, že zatočí příliš rychle. Naopak při natočení směrem od následujícího bodu, by mohlo dojít ke kolizi, vlivem příliš pomalého zatočení. Potenciálním řešením by bylo dynamické měnění vzdálenost detekce, podle natočení prvního segmentu hada. Pokud by byl segment natočen ve směru následujícího bodu na cestě, vzdálenost detekce by se snížila, dala hadovi čas se dostat dále od překážky a vyhnout se tak kolizi. Když se mluví o vzdálenosti detekce, nejedná se o vzdálenost, na kterou jsou hadovy senzory schopny detekovat překážky. Jde o vzdálenost, na kterou když se had přiblíží k bodu na vypočítané cestě, bod se považuje za dosažený.

Jedním z nedostatků modelu je poměrně dlouhý výpočetní čas. Co-simulace trvající několik minut simulačního času může zabrat několik hodin výpočtů. Vzhledem k velkému množství parametrů, by zkoumání jejich vlivů na chování robotického hada trvalo velmi dlouho. V této práci byly srovnány různé hodnoty *amplitudy* a *biasu*. Dále by bylo možné pozorovat změny způsobené různým nastavením zpoždění natočení segmentů, vzdálenosti detekce překážek, frekvence sinusové funkce řídící natočení, rychlosti změny ve zpětné vazbě, rozlišení mapy nebo podobou samotného hada. Různým počtem, rozměry a váhou segmentů, množstvím, umístěním, tvarem a povrchem koleček. Například – způsobují široká kolečka zbytečně velký skluz? Při zatáčení se část segmentu na vnější straně zatáčky pohybuje po delší trajektorii než část na straně vnitřní. Díky tomuto faktu nevyhnutelně vzniká skluz. Kdyby byly nahrazeny dvěma užšími kolečky, tento problém by mohl být minimalizován. Podobných otázek může být položeno mnoho, jejich zodpovězení ale může chvíli zabrat.

Další otázkou je, jak by se výsledky simulace přenesly na fyzický model v reálné aplikaci. Zde jsou některé potenciální problémy. Přesné souřadnice bodů nutné k řízení

hada v co-simulaci byly získány přímo z Adamsu. Fyzický robotický had by potřeboval schopnost poměrně přesně určit svou polohu.

Dalším problémem je detekce překážek. Vzhledem k vlnitému způsobu pohybu, by se senzor na pevno připojený k hlavě hada (po případě k jiným částem hada) neustále natáčel do stran, což by ztěžovalo zjištění přesné polohy překážek. Jedna možnost řešení je čistě výpočetní – pro natočení by se provedla korekce, podobně jako u určování skutečného směru pohybu. Další možností je použít konstrukci a řízení hada, které by první segment (hlavu) udržovalo natočený ve směru pohybu. Tento segment by samozřejmě nemohl být v kontaktu se zemí. Případně by se mohl natáčet samotný senzor.

Co-simulace proběhly pouze na jedné podložce, reálný terén může mít různé typy povrchu. Také není vždy rovný, mohlo by být vhodné model vylepšit schopností pohybu ve vertikálním směru. Buď využitím kloubů s větším počtem stupňů volnosti nebo natočením os rotace aktuátorů mezi sousedními segmenty různými směry. Díky takové konstrukci jsou někteří robotičtí hadi jsou schopni například pohybu v trubicích nebo lezení po sloupech.

Při přechodu ze simulačního do fyzického prostoru jistě existuje mnoho dalších potenciálních problémů. Zde jich bylo zmíněno pouze několik.

5 ZDROJE

[1] LEVEK, Martin. *Návrh modelu robotického hada* [online]. Brno, 2020 [cit. 2022-05-10]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/125073. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Tomáš Hůlka.

[2] SHUGEN. Analysis of creeping locomotion of a snake-like robot. *Advanced Robotics*. 2001, roč. 15, č. 2, str. 205-224. DOI: https://www.doi.org/10.1163/15685530152116236.

[3] HŮLKA, T., MATOUŠEK, R., DOBROVSKÝ, L., DOSOUDILOVÁ, M. a NOLLE, L. Optimization of Snake-like Robot Locomotion Using GA: Serpenoid Design. *MENDEL*. Srpen 2020, roč. 26, č. 1, str. 1-6. DOI: https://doi.org/10.13164/mendel.2020.1.001.

[4] BING, Z., et al. Perception-action coupling target tracking control for a snake robot via reinforcement learning. *Frontiers in Neurorobotics*. 2020, roč. 14, č. 79. DOI: https://doi.org/10.3389/fnbot.2020.591128.

[5] BAR-COHEN, Yoseph a BREAZEAL, Cynthia. *Biologically inspired intelligent robots*. Proc. SPIE 5051, Smart Structures and Materials 2003: Electroactive Polymer Actuators and Devices (EAPAD). 28. července 2003. DOI: https://doi.org/10.1117/12.484379.

[6] BEER, Randall D. *Biologically inspired robotics*. Scholarpedia. 2009. DOI: https://doi.org/10.4249/scholarpedia.1531.

[7] MORI, Mashiro. *The Uncanny Valley: The Original Essay by Masahiro Mori*. IEEE Spectrum [online]. 2012 [cit. 2022-05-10]. Dostupné z: https://spectrum.ieee.org/the-uncanny-valley.

[8] THENGADE, Anita a DONDAL, Rucha. Genetic Algorithm – Survey Paper. *IJCA Proc National Conference on Recent Trends in Computing, NCRTC*. Leden 2012, roč. 5, č. 1. Dostupné z: https://www.researchgate.net/publication/268010749_Genetic_Algorithm__Survey_Paper.

[9] DORIGO, M., BIRATTARI, M. a STUTZLE T. Ant colony optimization. *IEEE Computational Intelligence Magazine*. Listopad 2006, roč. 1, č. 4, str. 28-39. DOI: https://www.doi.org/10.1109/MCI.2006.329691.

[10] GONZALEZ, Carlos. *7 Bio-Inspired Robots that Mimic Nature*. Machine Design [online]. 18. srpna 2017 [cit. 2022-05-10]. Dostupné z: https://www.machinedesign.com/mechanical-motion-systems/article/21835853/7-bioinspired-robots-that-mimic-nature.

[11] WALLACH, Van a PETERS, James A. *snake* [online]. Encyclopedia Britannica, 10. září 2021 [cit. 2022-05-10]. Dostupné z: https://www.britannica.com/animal/snake.

[12] HSU, Jeremy. *Study Shows How Snakes Slither*. LIVE SCIENCE [online]. 8. června 2009 [cit. 2022-05-10]. Dostupné z: https://www.livescience.com/3639-study-shows-snakes-slither.html.

[13] PERRY, Lacy. *How Snakes Work*. HowStuffWorks [online]. [cit. 2022-05-10]. Dostupné z: https://animals.howstuffworks.com/snakes/snake.htm.

[14] *Snake Locomotion*. Reptile Shows of New England [online]. [cit. 2022-05-10]. Dostupné z: https://www.reptileshowsofnewengland.com/snake-locomotion/.

[15] *Snake Locomotion*. Animals Network [online]. [cit. 2022-05-10]. Dostupné z: https://animals.net/snake-locomotion/.

[16] CARTER, Lou. *How Do Snakes Move? (4 Snake Movement Types Explained)*. Snakes for Pets [online]. 15. prosince 2020 [cit. 2022-05-10]. Dostupné z: https://www.snakesforpets.com/how-do-snakes-move/.

[17] SAWE, Benjamin. *How Do Snakes Move?*. World Atlas [online]. 28. října 2019 [cit. 2022-05-10]. Dostupné z: https://www.worldatlas.com/articles/how-do-snakes-move.html.

[18] *The Amazing Sidewinder*. ZACKANDSCOTTKARMACHAMELEONS [online]. 4. března 2016 [cit. 2022-05-10]. Dostupné z: https://zackandscottkarmachameleons.wordpress.com/2016/03/04/the-amazing-sidewinder/.

[19] DUPONT, Bernard. *Young Puff Adder*. Wikimedia Commons [online]. 3. března 2012 [cit. 2022-05-10]. Dostupné z: https://commons.wikimedia.org/wiki/File:Young_Puff_Adder_%28Bitis_arietans%29_%287000378947%29.jpg.

[20] VIRGALA, I., KELEMEN, M., BOŽEK, P., BOBOVSKÝ, Z., HAGARA, M., PRADA, E., OŠČÁDAL, P. a VARGA, M. Investigation of Snake Robot Locomotion Possibilities in a Pipe. *Symmetry*. 2020, roč. 12, č. 6, str. 939. DOI: https://doi.org/10.3390/sym12060939.

[21] MADKOUR, A., AREF, W., REHMAN, F., RAHMAN, M. a BASALAMAH, S. *A Survey of Shortest-Path Algorithms* [online]. Univerzita Purdue, Západní Lafayette, USA a Univerzita Umm Al-Qura, Makkah, KSA. 2017 [cit. 2022-05-10]. Dostupné z: https://arxiv.org/pdf/1705.02044.pdf.

[22] HENZINGER, M., KLEIN, P., RAO, S. a SUBRAMANIAN, S. Faster Shortest-Path Algorithms for Planar Graphs. *Journal of Computer and System Sciences*. 1997, roč. 55, č. 1, str. 3-23. ISSN 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1493.

[23] SRYHENI, Said. *Shortest Path to Certain Nodes in a Graph*. Baeldung [online]. 19. října 2020 [cit. 2022-05-10]. Dostupné z: https://www.baeldung.com/cs/shortest-path-to-nodes-graph.

[24] *A* Search Algorithm*. GeeksforGeeks [online]. 13. dubna 2022 [cit. 2022-05-10]. Dostupné z: https://www.geeksforgeeks.org/a-search-algorithm/.

[25] HOFKAMP, Albert. *Jump Point Search: Fast A* Pathfinding for Uniform Cost Grids*. gamedev.net [online]. 31. října 2015 [cit. 2022-05-10]. Dostupné z: https://www.gamedev.net/tutorials/programming/artificial-intelligence/jump-point-search-fast-a-pathfinding-for-uniform-cost-grids-r4220/.

[26] PATEL, Amit. *Pathfinding*. Amit's A* Pages [online]. 21. dubna 2022 ©2022 [cit. 2022-05-10]. Dostupné z: https://theory.stanford.edu/~amitp/GameProgramming/.

[27] HANSEN, Eric A. a ZHOU, Rong. Anytime Heuristic Search. *Journal of Artificial Intelligence Research 2007*. El Segundo, CA, USA: AI Access Foundation. Leden 2007, roč. 28, č. 1, str. 267-297. ISSN 1076-9757.

[28] RUSSELL, S. Efficient memory-bounded search methods. *Neumann, B. (ed.). Proceedings of the 10th European Conference on Artificial intelligence*. Vídeň, Rakousko: John Wiley & Sons, New York, NY. 1992. str. 1-5.

[29] KORF, Richard E., REID, Michael a EDELKAMP, Stefan. Time complexity of iterative-deepening-A∗. *Artificial Intelligence*. Červen 2001, roč. 129, č. 1-2, str. 199-218. ISSN 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(01)00094-7.

[30] WASFY, Tamer a NOOR, Ahmed K. Computational strategies for flexible multibody systems. *Applied Mechanics Reviews*. Listopad 2003, roč. 56, č. 6, str. 553-613. DOI: https://doi.org/10.1115/1.1590354.

[31] *Multibody Dynamics*. MSC Software [online]. ©2022 [cit. 2022-05-10]. Dostupné z: https://www.mscsoftware.com/application/multibody-dynamics.

[32] *Multibody system*. ACADEMIC [online]. ©2000-2022 [2022-05-10]. Dostupné z: https://en-academic.com/dic.nsf/enwiki/4728912.

[33] WITTENBURG, Jens. *Dynamics of Multibody Systems*. Springer Berlin Heidelberg New York. ISBN 978-3-540-73913-5.

[34] HRDINA, J., NÁVRAT, A., VAŠÍK, P. et al. CGA-based robotic snake control. *Adv. Appl. Clifford Algebras*. 2017, roč. 27, č. 1, str. 621-632. DOI: https://doi.org/10.1007/s00006-016-0695-5.

[35] ZIENKIEWICZ, Olek C., TAYLOR, Robert L. a ZHU, J. Z. *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann. 31. srpna 2013. ISBN 978-0-08-095135-5.

[36] *MATLAB*. MathWorks [online]. ©1994-2022 [cit. 2022-05-10]. Dostupné z: https://www.mathworks.com/products/matlab.html.

[37] *SIMULINK*. MathWorks [online]. ©1994-2022 [cit. 2022-05-10]. Dostupné z: https://www.mathworks.com/products/simulink.html.

[38] *Adams*. MSC Software [online]. ©2022 [cit. 2022-05-10]. Dostupné z: https://www.mscsoftware.com/product/adams.

[39] WU, X., a MA, S. Head-navigated locomotion of a snake-like robot for its autonomous obstacle avoidance. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3. prosince 2010, str. 401-406. DOI: https://www.doi.org/10.1109/IROS.2010.5648818.

6 SEZNAM ZKRATEK, OBRÁZKŮ A TABULEK

AWA* – anytime weighted A* (kdykoli vážený A*)
BPS – Bayesian problem solver (Bayesovský řešič problému)
CGA – conformal geometric algebra (konformní geometrická algebra)
DTA* – do the right thing A* (udělej správnou věc A*)
FMA – finite element method (metoda konečných prvků)
FMS – flexible multi-body system (flexibilní systém více těles)
IDA* – iterative deepening A* (iterativně prohlubující A*)
JPS – jump point search (hledání skokovým bodem)
MBD – multi-body dynamics (dynamika více těles)
MKP – metoda konečných prvků (anglicky FMA – finite element method)
RTA* – real-time A* (A* ve skutečném čase)
SMA* – simplified memory bounded A* (zjednodušený pamětí omezený A*)
WA* – weighted A* (vážený A*)

7 SEZNAM PŘÍLOH

A_star.m
A_star_license.txt
add_obs_pad.m
add_to_open.m
control.slx
exp_node.m
generate_map.m
check_collision.m
min_f.m
node_idx.m
pad_map.m
snake_cosim.adm
snake_cosim.m
trans_back.m
transform.m
trim_path.m
variables.m
Wheeled_snake.bin