ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

ÚSTAV MECHANIKY, BIOMECHANIKY A MECHATRONIKY

KINEZIOLOGIE HORNÍ KONČETINY

BAKALÁŘSKÁ PRÁCE

PAVEL BAŠTÁŘ

Vedoucí bakalářské práce: prof. RNDr. Matěj Daniel, Ph.D.

2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Baštář** Jméno: **Pavel** Osobní číslo: **473722**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Kineziologie horní končetiny

Název bakalářské práce anglicky:

Upper extremity kinesiology

Pokyny pro vypracování:

1. Anatomie horní končetiny
2. Horní končetina jako mechanismus
3. Model horní končetiny při házení a uchopování předmětů
4. Model energetické náročnosti hodu
5. Analýza výsledků
6. Diskuse a závěr.

Seznam doporučené literatury:

Gregory S. Rash and Robert Shapiro, A Three-Dimensional Dynamic Analysis of the Quarterback's Throwing Motion in American Football. JOURNAL OF APPLIED BIOMECHANICS, 1995,11, 443-459, 1995.
Kiran Y. Jadhav, Sangeetha Prasanna Ram, Biomechanical Analysis of Human Upper Limb, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 6(1), 2017.
Charles J. Jordan, Laith M. Jazrawi, Joseph D. Zuckerman, Biomechanics of the shoulder. In Basic biomechanics of the musculoskeletal system Eds. Margareta Nordin; Victor H Frankel Publisher: Philadelphia : Wolters Kluwer/Lippincott Williams & Wilkins, ©2012.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. RNDr. Matej Daniel, Ph.D., České vysoké učení technické v Praze, Fakulta strojní

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **12.04.2021** Termín odevzdání bakalářské práce: **16.08.2021**

Platnost zadání bakalářské práce: _____

prof. RNDr. Matej Daniel, Ph.D. podpis vedoucí(ho) práce	doc. Ing. Miroslav Španiel, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání Podpis studenta

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně. Dále prohlašuji, že jsem všechny použité zdroje správně a úplně citoval a uvádím je v přiloženém seznamu použité literatury.

Nemám závažný důvod proti zpřístupňování této závěrečné práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V …….. dne …… podpis

Poděkování

Chtěl bych velice poděkovat prof. RNDr. Matěji Danielovi Ph.D. za odborné vedení mé práce, věcné připomínky, dobré rady, skvělý přístup a vstřícnost při konzultaci a vypracování mé bakalářské práce.

Abstrakt

Cílem předložené práce je analyzovat výhodnost poměru délek předloktí a paže ve vztahu k délce hodu a určit, zda vývoj člověka směruje k upřednostnění délky hodu.

Teoretická část nás seznamuje s anatomií a kinetikou jednotlivých komponent, které mají spojitost s pohybem horní končetiny přes hlavu v sagitální rovině. Obsahem praktické části je věnování se zejména metodě výpočtu pohybových rovnic pro pohyb horní končetiny pomocí programu Python PyDy multibody dynamics a ověření pomocí Lagrangeovy rovnice druhého druhu. V závěru práce analyzujeme a zhodnocujeme výsledky, zároveň ověřujeme platnost naší hypotézy.

Obsah

1 Úvod

Z biomechanického hlediska víme, jaký je způsob pohybu dolní končetiny, protože se jedná primárně o chůzi a běh. Na toto téma bylo již publikováno velké množství studií. Na rozdíl od toho, primární biomechanickou funkci horní končetiny zatím nevíme. Z vývoje se předpokládá, že primární funkcí bylo používání předmětů (nástrojů). Zároveň je horní končetina využívaná i pro další činnost, kterou je házení. Doposud však nejsou známé žádné další práce, které by se zabývaly vztahem mezi házením a antropometrií horní končetiny.

2 Mechanika házení

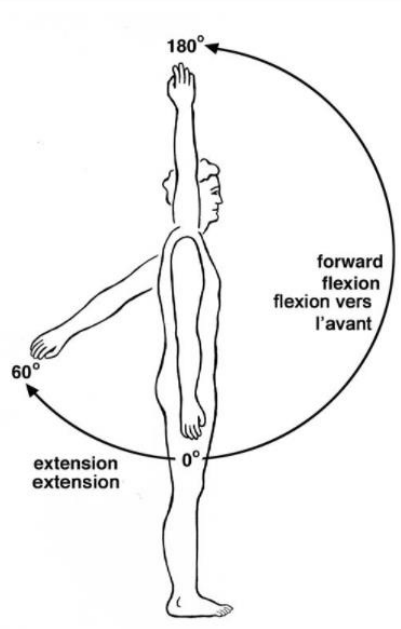
2.1 Anatomie horní končetiny

2.1.1 Rameno (articulatio humeri)

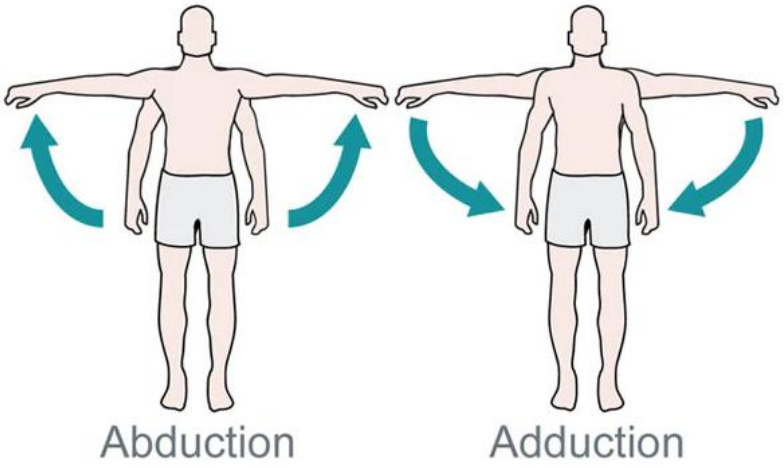
Rameno spojuje horní končetinu s trupem, působí ve spojení s loktem a díky tomu umožňuje prostorovou efektivní funkci. Skládá se z glenohumerálního, akromioklavikulárního, sternoclavikulárního a scapulothorakálního kloubu a dalších svalových struktur, které na ně působí a díky tomu je ramenní kloub považován za nejpohyblivější a nejdynamičtější kloub v těle. Jelikož nemá žádné kostní omezení, umožňuje nám velmi široký rozsah v pohybu, ovšem na úkor stability, kterou ale zajišťují jiné svalové a vazivové struktury. [1]

Aby mohlo docházet ke složitějším pohybům, které jsou nezbytné pro normální funkci ramenního komplexu, jsou zapotřebí čtyři artikulace s přidruženými komponenty, díky kterým jsme schopni větší mobility. [1]

Rozsah pohybu ramen se měří z hlediska flexe, extenze (=pohyb kosti pažní směrem od nebo k hrudníku v sagitální rovině), abdukce (= pohyb v koronální rovině) a tzv. intermálně – externí rotace (= axiální rotace kosti pažní, tzv. humeru s paží drženou v addukované poloze). Abdukce v koronární rovině je omezena kostnatým nárazem větší tuberosity na akromion. [1]



Obrázek 1 – Flexe a extenze ramene [2]



Obrázek 2 – Abdukce a addukce ramene [3]

Sternoclavikulární kloub se skládá ze zvětšeného mediálního konce klavikuly a superolaterálního aspektu manubria spojujícího horní končetinu přímo s hrudníkem, podřadně je přítomna i malá fazeta, která je spojena s prvním žebrem. Považujeme tento kloub za skutečný synoviální, který má sedlovitý tvar a obsahuje fibrokartilaginózní artikulární disk nebo meniskus. Přední a zadní sternoclavikulární vazy jsou umístěny v předních a zadních translacích, zatímco kostoklavikulární vaz stojí vzhůru a je považován za omezující fakt u sternoclavikulárního pohybu. Meziklavikulární vaz spojuje superomediální aspekt klíční kosti a napomáhá snadnějšímu omezení kloubu. Zadní část meziklavikulárního vazu též pomáhá s předním omezením sternoklavilárního kloubu. Interklavikulární vaz je uvolněný pokud je paže zvednutá. [1]

Akromioclavikulární kloub leží mezi laterálním koncem klíční kosti a akromionem lopatky. Tento kloub je vystaven velkému zatížení, které je přenášeno z hrudního svalstva na horní končetiny. Je to také synoviální kloub, ale má planární konfiguraci. Kloubní povrch je pokryt fibrocartilagem. Kloub uzavírá slabá vláknitá kapsle, která je posílena akromioklavikulárním vazivem. Akromioklavikulární vaz působí tak, že omezuje jak osovou rotaci, tak zadní translaci klíční kosti. Většinu vertikální stability kloubu zajišťují tzv. koraklavikulární vazy, které zavěšují lopatku z klavikuly. Koraklavikulární vazy se skládají z posteromediálního koroidu a anterolaterálně směrovaných lichoběžníkových vazů. Díky odlišné struktuře má různé biomechanické funkce. [1]

Důležitou součástí pro celkovou funkci ramenního komplexu je glenohumerální kloub. Humerální hlava je směřována dozadu. Větší a menší tuberosity leží laterálně od kloubního povrchu proximálního humeru, který slouží jako místo připojení svalstva. Proximální humerus se kloubí s glenoidní jamkou, která má mírný sklon, díky tomu má významný stupeň geometrické stability a díky tomu pomáhá odolávat nižší subluxaci anebo dislokaci. Glenoidní fossa je mělká a je schopna pojmout pouze přibližně jednu třetinu průměru hlavice humeru. Kostnatá architektura je umocněna chrupavčitým povrchem, který je centrálněji slabší, což mírně zvyšuje hloubku. Pohyb v glenohumerálním kloubu je tedy téměř rotační. Vzhledem k nedostatku kostního omezení je stabilita zajištěna kapsulárními, vazivovými a svalovými strukturami, které obklopují glenohumerální kloub. Glenohumerální kloubní pouzdro má významný stupeň inherentní laxity s povrchem, který je dvakrát větší než u hlavice humeru. Umožňuje tedy široký rozsah pohybu. Kapsle se přichytává přímo, výjimečně je připevněna na základě korakoidu. Kapsle má stabilizační roli, zejména zadní pouzdro má zásadní význam pro udržení stability. Glenohumerální a Coracohumerální vazy (svrchní, prostřední, spodní) jsou rozhodující pro stabilitu a funkci ramen. [1]

2.1.2 Loket (cubitus)

Loket je komplexní kloub, který funguje jako opora systému páky předloktí, který odpovídá za umístění ruky v prostoru. [4]

Loketní kloub umožňuje dva typy pohybu. Prvním je flexe – extenze a druhým je pronace – supinace. Humerounar a humeroradial umožňuje flexi a extenzi lokte. Proximální a radioulnární artikulace umožňuje pronaci a supinaci předloktí a je klasifikována jako trochoidní kloub. Komplex loketního kloubu, je-li považován za celek je trochleoginglymoidní kloub. Trochlea má tvar cívky a skládá se ze středních a bočních pysků. Kloubní povrch je pokryt hyalinní chrupavkou tvořící oblouk. Přední zkosení distálního humeru (30 stupňů) pomáhá zajistit stabilitu loketního kloubu v plné extenzi. Morrey (1986) poznamenal, že sklon ke vzniku zlomenin se v této oblasti vysvětluje tím, že část většího sigmoidálního zářezu není podporována silnější subchondrální kostí. Čtyři pětiny hlavy jsou pokryty hyalinní chrupavkou, pětině chybí kloubní chrupavka a silná subchondrální kost, což tedy vysvětluje zvýšený sklon ke vzniku zlomenin právě v této oblasti. [4]

2.1.3 Zápěstí (carpi) a ruka (manus)

Zápěstí je soubor kostí a struktur měkkých tkání, které spojují ruku s předloktím. Tento kloubní komplex je schopen podstatného pohybového oblouku, který posiluje funkci rukou a prstů. Zápěstí funguje kinematicky tím, že umožňuje změny umístění a orientace ruky vzhledem k předloktí a naopak. Ačkoliv funkcí všech kloubů horní končetiny je polohovat ruku v prostoru tak, aby mohla vykonávat činnosti každodenního života, zdá se tedy, že zápěstí je klíčem k funkci ruky. Stabilita zápěstí je nezbytná pro správnou funkci digitálních, flexorových a extenzorových svalů a poloha zápěstí ovlivňuje schopnost prstů maximálně se ohýbat a během předpětí účinně uchopit. [5]

Ruka je vysoce komplexní a mnohostranný mobilní přizpůsobivý orgán, protože odpovídá tvaru předmětů, které mají být uchopeny, zdůrazňuje nebo dokonce gestikuluje myšlenku, která má být vyjádřena (Tubiana, 1984). Ruka je konečným článkem mechanického řetězu, který začíná u ramene. Mobilita ramene, lokte a zápěstí, vše v různých rovinách, umožňuje ruce pohybovat se v prostoru relativně snadno a dosáhnout tak na všechny ostatní části těla. Ruka je velmi funkčně přizpůsobivá díky strukturálnímu základu, který obsahuje 19 kostí a 14 kloubů. Komplex zápěstního kloubu se skádá z vícenásobných artikulací, osmi karpálních kostí s distálním poloměrem a struktur v ulnokarpálním prostoru metakarpaly. Struktura měkkých tkání obklopují zápěstní kosti, které zahrnují šlachy, které procházejí přes zápěstí nebo se k nim připojují. Vazivová struktura spojuje zápěstní kosti k sobě navzájem. Osm karpálních kostí je rozděleno na proximální a distální řadu. Distální, karpální řada tvoří relativně nepohyblivou příčnou jednotku, která se kloubí s metakarpály za vzniku karpometakarpálních kloubů. Všechny čtyři kosti v distální řadě těsně přiléhají k sobě a jsou drženy pohromadě silnými nitrosytnými vazy. Pohyblivější je proximální řada, která se skládá z scaphoidu, lunátu a triquistra. Osmá karpální kost, pisiform, je sesamoidní kost, která mechanicky vylepšuje nejsilnější „motor“ zápěstí. Mezi proximálními a distálními řadami karpálních kostí je midcarpální kloub a mezi sousedními kostmi těchto řad jsou interkarpální klouby. [5]

2.2 Mechanika házení

Schopnost házet spočívá ve vypuštění předmětu do letu pomocí jedné nebo obou paží. V této práci budou brány v úvahu pouze jednostranné hody, typu přes rameno. Na rozdíl od více simultánních akcí paží v tlaku, vrhání zahrnuje sekvenční působení segmentů těla postupujících od větších, pomaleji se pohybujících trupových akcí k rychlejším, distálním akcím relativně menších ramenních a ručních segmentů. V závislosti na náročnosti sportu představují cíle hodu rychlost a přesnost v různých kombinacích. Cílem vrhače je provést sekvenci společných akcí, které při uvolnění vytvoří požadovanou rychlost a směr ruky, čímž se tato rychlost a směr předá předmětu uvolněnému rukou. Pohybový vzor používaný pro většinu házecích dovedností obvykle trvá méně než jednu sekundu od začátku akce do uvolnění předmětu. Ve skutečnosti ale analýza hodu přes paži odhalila, že celý švih ramene nahoru a dopředu trvá před uvolněním méně než 400 ms a výsledná rychlost míče s ohledem na konstantní prostorovou referenci se může zvýšit z méně než 6 m/s při 100 ms před vypuštěním na více než 34 m/s při uvolnění. Časté opakovaní takto vysokorychlostního hodu by mohlo způsobit vážné napětí na zapojených svalech, kostech a kloubech. [1]

2.2.1 Kinetika ramene

Četné svaly působí na různé části ramenního komplexu a zajišťují jak mobilitu, tak stabilitu dynamiky. Dynamická stabilizace probíhá několika možnými mechanismy, včetně pasivního svalového napětí, nebo prostřednictvím bariérového účinku staženého svalu a tlakové síly způsobené svalovou kontrakcí. Abychom porozuměli svalové funkci a přenosu sil, je potřeba vzít v úvahu velikost a aktivitu daného svalu. S ohledem na mnohočetné artikulace přítomné v ramenním komplexu může jakýkoliv daný sval ovlivnit několik různých kloubů a v závislosti na poloze horní končetiny se jeho vztah k jakémukoliv členu může změnit, čímž se změní i jeho účinek na tento kloub a výsledné síly nebo pohyby vytvořené. [1]

Zevní vrstvu tvoří svaly deltového svalu a prsního svalu. Tzv. deltoid tvoří normální zaoblený obrys ramene a trojúhelníkový tvar s předními, středními a zadními hlavami. Pod vnější vrstvou leží svalstvo rotátorové manžety: supraspinatus, infraspinatus, subscapularis a teres minor. Tyto čtyři svaly slouží k abdukci a rotaci humeru a působí jako důležité glenohumerální stabilizátory, jak pasivním svalovým napětím, tak dynamickou kontrakcí. Supraspinatus pochází ze supraspinatus fossa lopatky a vkládá se

do větší tuberosity proximálního humeru. Infraspinatus a teres minor fungují jako vnější rotátory humeru. Subscapularis leží na okraji lopatky a zasahuje do menší tuberosity proximálního humeru, funguje jako důležitý vnitřní rotátor humeru. Také spolu se středními a dolními glenohumerálními vazy působí jako přední stabilizátor glenohumerálního kloubu. Šlacha dlouhé hlavy bicepsu leží v glenohumerálním kloubu. Biceps funguje tak, že ohýbá předloktí a zvedá pažní kost. Dlouhá hlava bicepsu také hraje roli při udržování glenohumerální stability. Trapézový sval se nachází v krční oblasti a slouží ke zvedání, zatahování a otáčení lopatky. [1]

Elektromyografie umožňuje kvantifikaci svalové aktivity za dynamických podmínek. To umožňuje nahlédnout do úrovně svalové aktivity, ale přímo neindikuje generované síly. Úplné pochopení vyžaduje znalost momentové paže (= měřeno jako vzdálenost mezi okamžitým středem otáčení kloubu a vzdáleností svalového tahu) a fyziologického průřezu zapojeného svalu (= měřeno jako svalový objem dělený jeho délkou). V komplexu ramen je každý pohyb spojen s pohybem ve více kloubech a neustále se měnícími vztahy svalů. Rameno vzhledem k nedostatku kostní stability v glenohumerálním kloubu (síla je generována jedním svalem – primárním agonistou) vyžaduje aktivaci antagonistického svalu, takže nedojde k dislokační síle (Simon, 1994). Antagonist obvykle tohoto procesu dosahuje excentrickou kontrakcí, při které se sval prodlouží při aktivním stahování nebo produkcí neutralizační síly stejné velikosti, ale v opačném směru. Tento vztah je označován jako silový pár. Kolem glenohumerálního kloubu existuje pár sil v koronální rovině (= mezi deltoidem a dolní částí rotátorové manžety) a v příčné rovině mezi svalem sub-scapularis anteriorly a svalem zadní manžety routátotu svalu infraspinatus a teres minor. Relativní pohyb je způsoben nerovnováhou mezi agonistou a antagonistou, která vytváří točivý moment. Svaly ramenního pletence jsou seskupeny podle relativní důležitosti při nejzákladnějším pohybu komplexu ramene. První skupina zahrnuje deltoid, lichoběžník, supraspinattis a serratus anterior. Druhá skupina se skládá ze střední části lichoběžníku, infraspinátu a dlouhé hlavy bicepsu. Třetí skupinu tvoří zadní hlava deltového svalu, klavikulární hlavice pectoralis a hlavní část lichoběžníku. Čtvrtá a poslední skupina zahrnuje hrudní hlavu pectoralis major latisssimus dorsi a dlouhou hlavu tricepsu. Elektromyografické studie ukázaly, že supraspinatus i deltoid jsou aktivní v celém rozsahu. S rostoucí abdukcí se snižuje svalová síla nebo svislá síla. Když se porovná čistá abdukce s čistou přední elevací, jsou stejné

základní vztahy pozorovány u rotátorové manžety, která stabilizuje glenohumerální kloub, zatímco deltoid poskytuje potřebný točivý moment. [1]

2.2.2 Kinetika lokte

Flexe a extenze lokte probíhá v oblasti humeroradial a humeroulnar. Normální rozsah flexe-extenze je od 0 do 146 stupňů s funkčním rozsahem 30 až 130 stupňů. Normální rozsah pronace-supinace předloktí je v průměru od 71 stupňů pronace do 81 stupňů supinace (Morrey, 1981). Většina aktivit se ovšem provádí ve funkčním rozsahu 50 stupňů pronace až 50 stupňů supinace. Osa rotace flexe-extenze se vyskytuje kolem těsného okolí bodů měřícího 2 až 3 mm v nejširším rozměru. Pronace a supinace probíhá primárně v humeroradiálních kloubech s předloktím rotujícím kolem podélné osy procházející středem capitelu, radiové hlavy a distální ulnární kloubní plochou. Osa je šikmá ve vztahu k anatomické ose poloměru a ulny. Během pronace a supinace se radiální hlava otáčí uvnitř prstencového vazu a distální poloměr se otáčí kolem distální ulny v oblouku. Carret a kol. (1976) studovali centrum rotace na proximálních a distálních, radioulnárních kloubech s předloktím v různých stupních pronace a supinace. Zjistili, že proximální okamžité centrum rotace se mění s rozdíly v zakřivení radiální hlavy mezi jednotlivci. [4]

Valgusová poloha lokte v plném natažení se běžně označuje jako úhel přenášení. Nosný úhel je definován jako úhel mezi anatomickou osou loketní kosti a pažní kosti. Měřený je dle orientace ulny s ohledem na humerus nebo naopak v plné extenzi. Tento úhel je u dětí menší než u dospělých, stejně jako je větší u žen než u mužů. Pokud je úhel nošení definován jako úhel abdukce-addukce ulny vůči humeru pomocí Eulerových úhlů k popisu pohybu paže, úhel přenášení se zmenšuje, když se flexe kloubu mění v extrémní flexi na varus. Valgusovým silám odolává především přední pás komplexu mediálního kolaterálního ligamentu (MCL). Komplex MCL se skládá z předního, zadního, příčného vazu. Přední vaz se napíná v prodloužení, zatímco zadní vaz se napíná ve flexi. K tomu dochází právě proto, že komplex MCL nevzniká ve středu osy rotace lokte. Přední pás MCL komplex pochází z dolního povrchu mediálního epikondylu distálního humeru a vkládá se podél mediálního okraje olecranonu. S neporušeným předním pásem neposkytuje radiální hlava významnou dodatečnou odolnost vůči valgusovému napětí. S transponovaným nebo narušeným předním pásem se však radiální hlava stává primárním omezením valgusového napětí a zdůrazňuje jeho funkci sekundárního stabilizátoru v loktech s neporušeným MCL (Palmer, Glisson, Werner, 1982).

Byla zaznamenána zvýšená valgusová laxnost po radiální excizi hlavy (Coleman, Blair, Shurr, 1987). Komplex LCL se skládá z radiálního kolaterálního vazu, který pochází z laterálního epikondylu a přechází povrchově na prstencový vaz, který je napojený na supinátorový hřeben ulna a připojený komplex LCL. Počátek komplexu LCL leží ve středu osy rotace lokte, což vysvětluje jeho konzistentní délku v celém oblouku extenze-flexe. Struktury omezující extenzi lokte zahrnují oleubranový proces a přední pás komplexu MCL. Pasivní odolnost vůči pronaci-supinaci je z velké části poskytována antagonistickou svalovou skupinou, nikoliv vazivovými strukturami (Braune, Flugel, 1842). Jiné studie také ukázaly, že kvadrátový vaz poskytuje omezení rotace předloktí (Spinner, Kaplan, 1970). Podélnou stabilitu předloktí zajišťuje jak interosseální membrána, tak trojúhelníková fibrocartilage (Lee, 1992). [4]

Primární flexor lokte brachialis vzniká z předního aspektu humeru a zasouvá se do předního aspektu proximální ulny. Biceps vzniká prostřednictvím šlachy s dlouhou hlavou ze supraglenoidního tuberkulu a šlachy s krátkou hlavou z korakoidního procesu lopatky. Brachio – radialis, který pochází z postranních dvou třetin distálního humeru a vkládá se na distální aspekt poloměru v blízkosti radiálního styloidu, je aktivní při rychlých flexních pohybech lokte a při zvedání váhy, při pomalé flexi. Primární extensor loktu, tricepsu je složen ze tří samostatných hlavic. Dlouhá hlava pochází z infraglenoidního tuberkulu a mediální a laterální hlavy ze zadního aspektu humeru. Tři hlavy se spojují a vytváří se jedna šlacha. Anconeus sval, který vzniká z posterolaterálního aspektu distal humerus je také akivní v prodloužení. Tento sval je aktivní při zahájení a udržení extenze, zatímco triceps má největší pracovní sílu ze všech loketních extenzorů. Svaly zapojené do supinace předloktí zahrnují supinátor, bicepsy a laterální epikondylární extenzory zápěstí a prstů. Primárním svalem, zapojeným do supinace je biceps brachii. Mezi svaly zapojené do pronace patří prona-tor quadratus a pronator teres. Pronator quadratus je primárním nositelem předloktí bez ohledu na jeho polohu. Muži byli při testování pevnosti lokte trvale o 40 % silnější než ženy. [4]

Elektromyografie pomáhá při definování muskulatury loktů během každodenních činností v životě. Biceps brachii je při flexi lokte aktivní pouze minimálně, aktivita není ovlivněna rotací předloktí během flexe. Anconeus je aktivní ve všech polohách a je považován za dynamický stabilizátor kloubů. [4]

Ewald a kol. (1977) určili, že tlaková síla loketního kloubu byla osmkrát větší než hmotnost držená nataženou rukou. An Morrey (1991) přišel s tím, že během namáhavého zvedání závaží se výsledná síla v ulnohumerálním kloubu pohybuje od jednoho do trojnásobku tělesné hmotnosti. Přenos síly radiální hlavou je nejvyšší mezi 0 a 30 stupni ohybu a je větší v pronaci než v supinaci. Síla generovaná v loketním kloubu je největší, když je zahájena flexe. Zvýšená síla flexe a snížené loketní síly jsou patrné u lokte při 90 stupňů flexe. Zajímavé je, že směr vektoru výsledné síly v lokti se mění o více než 180 stupňů v celém prodloužení flexe. Klinicky by tato změna ve výsledném vektoru měla být vzata v úvahu při zvažování vnitřní fixace zlomenin distálního humeru a také při zvažování totální náhrady kloubu. Síla generovaná v lokti byla při určitých aktivitách prokázána až trojnásobkem tělesné hmotnosti (An et al.,1981). Kontaktní oblasti lokte se vyskytují na čtyřech místech, dva jsou umístěny na olecranonu a dva na koronoidě. [4]

Pokud je loket ohnutý o 90 stupňů, důležitými svaly jsou brachialis a biceps a síla je vytvářená šlachami těchto svalů. Vzdálenost mezi středem otáčení loketního kloubu a šlach těchto svalů je cca 5 cm. Hmotnost předloktí (2 kg) vytváří gravitační sílu (W) 20 N. Síla svalů potřebná k udržení lokte v ohnuté poloze se vypočítá z rovnovážné rovnice pro okamžiky. Malé zatížení působící na ruku dramaticky zvyšuje reakční sílu loketního kloubu. [4]

2.2.3 Kinetika zápěstí a ruky

Zápěstí je komplikovaný kloubní komplex skládající se z více kloubů osmi karpálních kostí s distálním poloměrem, strukturami TFCC a metakarpálů. Karpální kosti se obvykle dělí na proximální a distální řadu. Pohybu zápěstí zahrnují flexi – extenzi a radiálně – ulnární odchylky. Stabilitu během radiálně – ulnární odchylky zajišťuje systém dvojitého-V, tvořený palmárním vnitřním vazem a paprskovými a ulnolunátními vazy. Proximální a distální karpální řady vytvářejí bimuskulární, biarticulární řetězec, který se při stlačení zhroutí. Stabilitu zajišťuje přesná opozice kloubních povrchů a složité vnitřní a vnější vazivové vazby. Extensor carpi ulnaris, extensor pollicis longus působí jako dynamický kolaterální systém, který zajišťuje stabilitu zápěstí při funkčních pohybech rukou. Poloha zápěstí ovlivňuje schopnost prstů maximálně se ohnout a natáhnout a efektivně uchopit. [5]

TFCC hraje významnou roli při tlumení tlakových zatížení napříč zápěstním kloubem. Flexor carpi ulnaris je nejsilnější motor zápěstí a má tendenci umístit zápěstí do polohy flexe a ulnární odchylky. Prstové paprsky ruky jsou uspořádány do tří oblouků, jeden podélný a dva příčný. Porucha nebo zhroucení obloukového systému v důsledku poranění kosti, revmatického onemocnění nebo ochrnutí vnitřních svalů ruky může přispět k vážnému postižení a deformaci. [5]

Ruka je hlavním nástrojem dotyku. Kombinace citlivosti a motorické funkce dávají ruce její velký význam. Lichoběžníkový, kapitační a druhý a třetí metakarpál svými těsně přiléhajícími artikulacemi tvoří nehybnou jednotku ruky. Čtvrtému a pátému metakarpu je povoleno mírné množství palmárního posunu, což je pohyb nezbytný pro uchopení. Nejdůležitější pohyb palce, kdy je abdukce spojená s rotací v CMC kloubu a pohybuje palcem směrem ke špičce malíčku. Paprsky prstů jsou ovládány koordinovaným působením vnějšího a vnitřního svalového systému. Činnost každého paprsku není zcela nezávislá na sousedním paprsku. Klouby MCP jsou stabilizovány především radiálními a ulnárními kolaterálními vazy a také příčným intermetakarpálním vazem, který navzájem spojuje dlaňové dlahy. Digitální kladkový systém opěrky šlachy digitálního ohýbače je nezbytný pro udržení relativně konstantního momentového ramene pro ohýbače prstů a pro minimalizaci zvýšení napětí mezi šlachou a pochvou. Druhá a čtvrtá prstencová kladka hrají v tomto ohledu obzvláště důležitou roli. Šlacha flexor superficialis má celkově větší odchylku než flexor profundus. Exkurze flexorů je větší než extenzory. Exkurze šlachy vnějšího svalu je obecně větší než exkurze vnitřních šlach. V každém kloubu v důsledku narušení systému kladky je vyžadována zvláštní odchylka, což vede k nedostatečnému vychýlení a následné slabosti distálnějších kloubů. Síla flexorů prstů je více než dvakrát větší než extenzory. Důležitá je také relativní délka metakarpálů a falangů a paprsků prstů jako celku. Poloha palce a vztah mezi rukou a předloktím jsou nejdůležitějšími rozdíly mezi silovým úchopem a přesným ovládáním. [5]

2.2.4 Biomechanika házení přes rameno

Přehoz přes rameno je pojmenován podle polohy ruky vůči paži a ramenům při uvolnění předmětu. Vše bude uváděno pro pravorukého vrhače. V biomechanické literatuře se k analýze mechaniky házení přistupovalo téměř výhradně z kinematiky, nikoliv z kinetického hlediska. Ačkoliv techniky dynamické analýzy a matematického modelování nebyly obecně použity při studiu rychlých, komplexních, vícerozměrných

pohybů hodu přes paži, tyto techniky mohou poskytnout cenné informace v blízké budoucnosti. Mezitím byla kinematografie a elektromyografie hlavními technikami získanými popisnou analýzou tohoto vzoru házení. Metoda, se kterou se často setkáváme při určování účinnosti pohybů házení, zahrnuje měření počáteční rychlosti a směru promítaného předmětu při jeho uvolnění. Měření rychlosti předmětu při uvolnění lze získat pomocí technik filmové analýzy (Cooper a Glassow, 1976) nebo pomocí stopek (Safrit a Pavis, 1968), velocimetru (Roberts, 1972) nebo upraveného radaru (Sanders, 1978). [6]

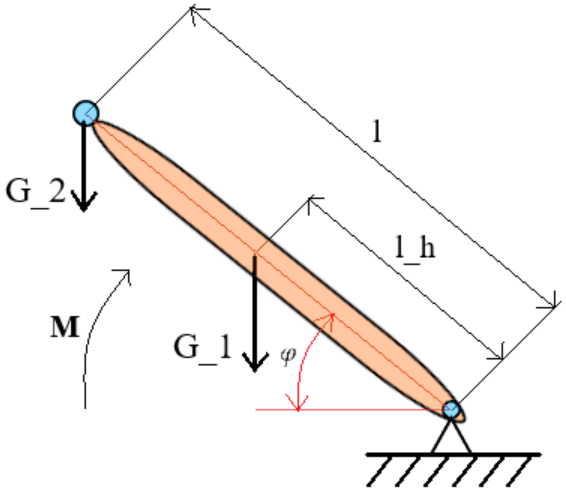
Úsilí o identifikaci příčin poranění vrhací paží je stále na hypotetické úrovni, pokračuje hledání ještě konkrétnějších vztahů mezi zraněními a kloubními pohyby, ke kterým dochází při házení. Kinetické analýzy vysokorychlostních, komplexních, trojrozměrných vrhacích akcí dosud neposkytly dostatečné údaje o kloubních momentech síly nebo točivých momentů způsobujících zranění, aby bylo možné vysvětlit konkrétní zranění. Tato oblast studia však nabízí příslib. Mezitím by další kinematické analýzy (prostřednictvím kinematografie a elektromyografie) ve spojení s klinickou studií vzorů házení u mladých vrhačů a u dospělých mohly zaplnit mezery v našich současných znalostech házení. Pokud jde o otázku, zda může být typ zkušeností Malé ligy, či nikoli, výhodný pro paži mladého vrhače, zatím není k dispozici konečná odpověď. Důkazy na jedné straně jasně spojují mnoho konkrétních zranění s opakujícími se vysokorychlostními házeními, jiné sportovní a rekreační aktivity však také zahrnují schopnost házet. Možná je jedním z řešení prevence zranění tzv. umírnění, to znamená, že nadměrné házení mimo herní situace by mělo být omezeno. [7]

3 Cíl práce

Cílem této práce je vyvrátit nebo potvrdit hypotézu, zda se struktura horní končetiny vyvíjí tak, aby bylo možné maximalizovat vzdálenost hodu.

4 Metodika výpočtu

4.1 Model hodu bez loketního kloubu



Obrázek 4 – Horní končetina bez lokte v obecné poloze

Budeme vycházet z momentové rovnice paže, kde suma momentů sil působící na těleso se musí rovnat setrvačnosti.

$$I \cdot \ddot{\varphi} = M - G_1 \cdot l_T \cdot \cos\varphi - G_2 \cdot l \cdot \cos\varphi \qquad (1)$$

, kde φ je úhel natočení $[°]$

$\ddot{\varphi}$ je úhlové zrychlení → $\ddot{\varphi} = \frac{d\dot{\varphi}}{d\varphi} \cdot \dot{\varphi}$ $[°.s^{-2}]$

I je setrvačnost tělesa → $I = \frac{1}{3} m_1 l^2 + m_2 l^2$ $[kg.m^2]$

G_1 je tíha paže, přičemž m_1 je hmotnost paže → $G_1 = m_1 \cdot g$ $[N]$

G_2 je tíha hozeného předmětu, přičemž m_2 je hmotnost předmětu → $G_2 = m_2 \cdot g$ $[N]$

M je moment síly generovaný v rameni $[N.m]$

l_T je vzdálenost těžiště od ramene $[m]$

l je délka paže $[m]$

Po dosazení za úhlové zrychlení získáme tvar, u kterého lze jednoduše provést integraci.

$$I \cdot \dot{\varphi} \cdot d\dot{\varphi} = (M - G_1 \cdot l_T \cdot \cos\varphi - G_2 \cdot l \cdot \cos\varphi) \cdot d\varphi \qquad (2)$$

Integraci provedeme v mezích od 0 do φ_H (úhel paže při vypuštění předmětu).

$$\int_0^{\dot{\varphi}_H} I \cdot \dot{\varphi} \cdot d\dot{\varphi} = \int_0^{\varphi_H} (M - G_1 \cdot l_T \cdot \cos\varphi - G_2 \cdot l \cdot \cos\varphi) \cdot d\varphi \qquad (3)$$

A dostaneme výsledný tvar rovnice.

$$\frac{1}{2} I \cdot \dot{\varphi}_H^2 = M \cdot \varphi_H - (m_1 l_T + m_2 l) \cdot g \cdot \sin\varphi_H \qquad (4)$$

Nyní už víme, jak se paže chová v závislosti velikosti síly na úhlu natočení. My ale potřebujeme vědět vzdálenost dohozu. Proto si vypočítáme vzdálenost dohozu v závislosti na úhlu a rychlosti výhozu. Dostáváme následující rovnici, kterou využijeme i pro výpočty délek hodu v následujících kapitolách.

$$D = \frac{v_0^2 \cdot \sin(2 \cdot (90 - \varphi_H))}{g} \qquad (5)$$

, kde D je vzdálenost dohozu [m]

φ_H je úhel výhozu [°]

v_0 je počáteční rychlost míčku při vypuštění [8]

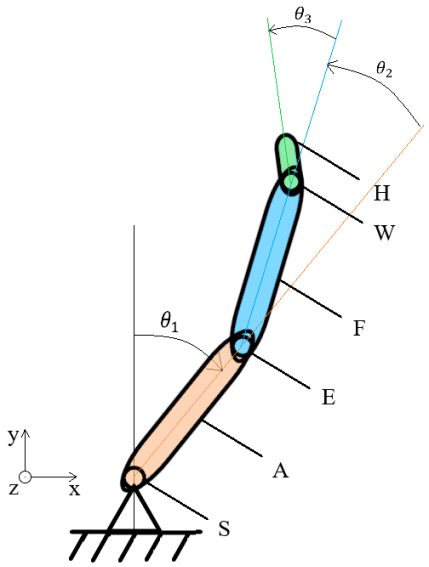
Po dosazení do předchozí rovnice dostáváme výsledný tvar.

$$D = \frac{2 \cdot \sin\left(2 \cdot (90 - \varphi_H)\right)}{g \cdot \left(\frac{1}{3} m_1 + m_2\right)} \cdot [M \cdot \varphi_H - (m_1 l_T + m_2 l) \cdot g \cdot \sin\varphi_H] \qquad (6)$$

Pro názorné předvedení dané rovnice, použijeme vykreslení do *x, y* a *z* souřadnicového systému, kde za osu *x* dosadíme vzdálenost hodu D $[m]$, za *y* úhel paže při vyhození φ_H [°] a za z moment působící v rameni M $[N.m]$.

4.2 PyDy multibody dynamics

V rámci naší studie, jsme pohyb horní končetiny omezili do 2D rozměru v sagitální rovině. Uvažovali jsme tři stupně volnosti: pohyb v (ramenním kloubu) articulatio humeri, (loketním kloubu) articulatio cubiti a v (zápěstí) articulatio radiocarpalis. Síly, které vytváří jednotlivé svaly, jsme nahradili pomocí momentů, které působí mezi jednotlivými segmenty těla. Následující obrázek znázorňuje model a jednotlivé segmenty.



Obrázek 3 – Souřadnicový systém horní končetiny pro PyDy

V referenčním souřadnicovém systému jsou důležité čtyři souřadnicové systémy a tři body. Oranžová vztažná soustava paže A je v articulatio humeri S spojena čepovým kloubem s inerciální vztažnou soustavou základny a otáčí se relativně vůči vertikální pomocné přímce o úhel θ_1. Modrá vztažná soustava předloktí F je přes čepový kloub E připevněna k paži a otáčí se vůči ní o úhel θ_2. Zelená vztažná soustava zápěstí H je spojena čepovým kloubem k distální části předloktí a rotuje relativně o úhel θ_3 vzhledem k ose předloktí. Důležité je, že všechny rotace jsou kolem osy z a jsou relativní k orientaci předchozího tělesa.

Délky paže a předloktí jsou definovány proměnnými l_A a l_F. Dále jsou důležité tři body A_0, F_0 a H_0, které se nacházejí na spojnici distálního a proximálního kloubu každého segmentu ve vzdálenostech d_A, d_F a d_H. Gravitace je směrována v záporném směru na ose y a na každý hmotný střed působí silou o velikosti $m_A g$, $m_F g$ a $m_H g$. Síly působící ve svalech při kontrakcích jsou nahrazeny třemi adekvátními momenty síly: v rameni T_S,

v lokti T_E a v zápěstí T_W. Momenty působící mezi jednotlivými segmenty vyvolávají akci, ale i stejně velkou, opačně orientovanou reakci na segment spjatý s proximálním kloubem.

Nejprve jsme naimportovali potřebné funkce ze SymPy, což nám umožňuje sestrojit časově proměnné vektory ve třech vztažných soustavách.

```
from __future__ import print_function, division
from sympy import symbols, simplify
from sympy.physics.mechanics import dynamicsymbols, ReferenceFrame,
Point
```

SymPy má obsáhlou knihovnu stylů pro vypisování rovnic. Využili jsme tedy funkci *init_vprinting* pro zápis matematických rovnic v klasickém tvaru.

```
from sympy.physics.vector import init_vprinting
init_vprinting(pretty_print=True)
```

K importu některých obrázků byl pro znázornění použit IPython.

```
from IPython.display import Image
```

Dalším důležitým krokem je definovat tři vztažné soustavy pro paži, předloktí a ruku. Tyto vztažné soustavy obsahují informace, jak je každý segment spjatý se sousedním. K popisu vzájemných orientací jsme použili relativní úhly, úhlové rychlosti a úhlová zrychlení. Poslední vztažnou soustavou je inerciální soustava rámu.

```
inertial_frame = ReferenceFrame('I')
arm_frame = ReferenceFrame('A')
forearm_frame = ReferenceFrame('F')
hand_frame = ReferenceFrame('H')
```

Následně jsme potřebovali vědět, jak jsou vztažné soustavy vůči sobě orientované. Proto jsme definovali tři zobecněné souřadnice $\theta_1(t)$, $\theta_2(t)$ a $\theta_3(t)$ pro úhly paže, předloktí a zápěstí, které jsou proměnné v čase.

```
theta1, theta2, theta3 = dynamicsymbols('theta1, theta2, theta3')
```

Jako první jsme nastavili orientaci paže vzhledem k inerciální vztažné soustavě rámu. Chceme, aby se paže otáčela o úhel θ_1 relativně vůči rámu. Je vhodné použít metodu *ReferenceFrame.orient()* a jako první zadat základní rám, typ orientace *Axis* a n-tici obsahující úhel rotace a vektor kolem kterého dochází k rotaci, v našem případě vektor inerciálního vztažného systému *Z*.

```
arm_frame.orient(inertial_frame, 'Axis', (theta1, inertial_frame.z))
```

Nyní použijeme úhel θ_2 pro definování vzájemného pohybu paže a předloktí. Úhel θ_3 pak pro předloktí a ruku.

```
forearm_frame.orient(arm_frame, 'Axis', (theta2, arm_frame.z))
hand_frame.orient(forearm_frame, 'Axis', (theta3, forearm_frame.z))
```

Následně jsme zapsali matici směrových kosinů, která se vztahuje k vztažné soustavě zápěstí a zjednodušíme její zápis pomocí *simplify()*

```
simplify(hand_frame.dcm(inertial_frame))
```

K odvození pohybových rovnic systému nadefinujeme rychlosti hmotných středů každého tuhého tělesa. Určili jsme si body, ve kterých probíhá rotace: rameno, loket a zápěstí, abychom následně zjednodušil popis polohy těžiště každého segmentu.

Rameno je základním kloubem a ostatní body budou definovány vzhledem k němu.

```
shoulder = Point('S')
```

Loket *E* je vzhledem k rameni definován vektorem v distálním směru, který má délku l_A.

```
arm_length = symbols('l_A')
elbow = Point('E')
```

Referenční bod a vektor zapíšeme funkcí *Point.set_pos()*

```
elbow.set_pos(shoulder, arm_length * arm_frame.y)
```

Všechny ostatní pozice pak pomocí *Point.pos_from()*

```
elbow.pos_from(shoulder)
```

Pokud chceme znát pozici ramena v prostoru vůči inerciální vztažné soustavě, existuje pro to funkce:

```
elbow.pos_from(shoulder).express(inertial_frame).simplify()
```

Podobným způsobem jsme udělali zápis pro zápěstí vzhledem k lokti.

```
forearm_length = symbols('l_F')
wrist = Point('W')
wrist.set_pos(elbow, forearm_length * forearm_frame.y)

wrist.pos_from(shoulder).express(inertial_frame).simplify()
```

Dále musíme definovat relativní vzdálenosti d_A, d_F a d_H středů hmotností A_0, F_0 a H_0 od distálních konců segmentů.

```
arm_com_length = symbols('d_A')
arm_mass_center = Point('A_o')
arm_mass_center.set_pos(shoulder, arm_com_length * arm_frame.y)

forearm_com_length = symbols('d_F')
forearm_mass_center = Point('F_o')
forearm_mass_center.set_pos(elbow, forearm_com_length *
forearm_frame.y)

hand_com_length = symbols('d_H')
hand_mass_center = Point('H_o')
hand_mass_center.set_pos(wrist, hand_com_length * hand_frame.y)
```

Důležitým bodem je vyjádření rovnic kinematiky. Máme tři úhly $\theta_1(t)$, $\theta_2(t)$ a $\theta_3(t)$, se kterými musíme provázat odpovídající zobecněné úhlové rychlosti $\omega_1(t)$, $\omega_2(t)$ a $\omega_3(t)$ přes derivace $\dot{\theta}_1(t) = \omega_1(t)$, $\dot{\theta}_2(t) = \omega_2(t)$ a $\dot{\theta}_3(t) = \omega_3(t)$. Toto je důležité, aby výsledné pohybové rovnice byly odvozeny ve formě diferenciálních rovnic prvního řádu. Po odečtení levých stran vzniknou tyto rovnice $\omega_n(t) - \dot{\theta}_n(t) = 0$

```
omega1, omega2, omega3 = dynamicsymbols('omega1, omega2, omega3')

kinematical_differential_equations = [omega1 - theta1.diff(),
                                      omega2 - theta2.diff(),
                                      omega3 - theta3.diff()]
```

Nyní už můžeme zapsat zobecněné rychlosti pro popis úhlových rychlostí vztažných soustav vůči inerciální vztažné soustavě.

```
arm_frame.set_ang_vel(inertial_frame,omega1*inertial_frame.z)
forearm_frame.set_ang_vel(arm_frame,omega2*inertial_frame.z)
hand_frame.set_ang_vel(forearm_frame,omega3*inertial_frame.z)
```

Nakonec jsou zapotřebí lineární rychlosti těžišť každého ze segmentu, počínaje ramenem, které má rychlost rovnu 0.

```
shoulder.set_vel(inertial_frame, 0)
```

Point.v2pt_theory() je způsob jak vypočítat rychlost jakéhokoliv bodu ve vztažné soustavě dokonale tuhých těles, pokud je známa lineární rychlost jiného bodu a úhlová rychlost soustavy. Tuto skutečnost využijeme při výpočtu počáteční rychlosti kriketového balónku při vypuštění.

```
arm_mass_center.v2pt_theory(shoulder, inertial_frame, arm_frame)
arm_mass_center.vel(inertial_frame).express(inertial_frame).simplify()

elbow.v2pt_theory(shoulder, inertial_frame, arm_frame)

forearm_mass_center.v2pt_theory(elbow, inertial_frame, forearm_frame)
forearm_mass_center.vel(inertial_frame).express(inertial_frame).simpli
fy()

wrist.v2pt_theory(elbow, inertial_frame, forearm_frame)
hand_mass_center.v2pt_theory(wrist, inertial_frame, hand_frame)
hand_mass_center.vel(inertial_frame).express(inertial_frame).simplify()
```

Dalším velkým krokem je potřeba poskytnout programu informace o hmotnostech a setrvačnostech příslušných těles. Každé z tuhých těles má svou hmotnost, která působí proti lineárním zrychlením a setrvačnost, která působí proti rotačním zrychlením.

Nejprve naimportujeme výsledky z předchozího kódu.

```
from __future__ import print_function, division
```

Budeme též potřebovat funkci pro generování veličin setrvačnosti a RigidBody pro zápis nezbytných informací o každém tuhém tělese.

```
from sympy.physics.mechanics import inertia, RigidBody
```

Nyní přejdeme od importování knihoven k informacím o hmotnostech. Každé těleso bude mít v našem případě konstantní hmotnost, takže pro každé těleso vytvoříme symbol.

```
arm_mass, forearm_mass, hand_mass = symbols('m_A, m_F, m_H')
```

Vycházíme z předpokladu, že se horní končetina pohybuje ve 2D, rotuje tedy pouze kolem osy *Z* a že všechny segmenty jsou symetrické vůči rovinám *XZ* a *YZ*. Takže potřebujeme pouze jednu proměnnou pro každé těleso.

```
arm_inertia, forearm_inertia, hand_inertia = symbols('I_Az, I_Fz,
I_Hz')
```

Funkce *inertia()* je praktická pro vytváření setrvačných Dyadik (tenzory závislé na bázi). V našem případě, když víme, že rotace kolem os *X* a *Y* je rovna nule, tak budeme mít pouze jednu vstupní proměnnou pro setrvačnost kolem osy *Z*.

```
arm_inertia_dyadic = inertia(arm_frame, 0, 0, arm_inertia)
```

Setrvačnost obecně ukládáme v podobě Dyadik. Pokud chceme vidět jak je setrvačnost vyjádřena, použijeme metodu *to_matrix()*.

```
arm_inertia_dyadic.to_matrix(arm_frame)
```

Také budeme potřebovat vědět k jakému bodu je setrvačnost definována. V našem případě definujeme všechny setrvačnosti v okolí těžiště a zapíšeme pomocí PyDy do n-tice setrvačnosti Dyadik a Point.

```
arm_central_inertia = (arm_inertia_dyadic, arm_mass_center)
```

Setrvačnost pro předloktí a ruku zapíšeme obdobně.

```
forearm_inertia_dyadic = inertia(forearm_frame, 0, 0, forearm_inertia)
forearm_central_inertia = (forearm_inertia_dyadic,
forearm_mass_center)

hand_inertia_dyadic = inertia(hand_frame, 0, 0, hand_inertia)
hand_central_inertia = (hand_inertia_dyadic, hand_mass_center)
```

Pro kompletní specifikaci tuhého tělesa je potřeba určit polohu těžiště, vztažnou soustavu, a setrvačnost.

```
arm = RigidBody('Arm', arm_mass_center, arm_frame,
                arm_mass, arm_central_inertia)
forearm = RigidBody('Forearm', forearm_mass_center, forearm_frame,
                forearm_mass, forearm_central_inertia)
hand = RigidBody('Hand', hand_mass_center, hand_frame,
                hand_mass, hand_central_inertia)
```

Nyní je potřeba zahrnout do výpočtu všechna zatížení. V našem případě uvažujeme tři tíhové síly působící na každý segment a mezi segmenty pak momenty sil, které způsobují rotaci. Budeme tedy specifikovat vektory pro každé zatížení a body, potažmo vztažnou soustavu, na které působí.

Nejprve potřebujeme gravitační konstantu g.

```
g = symbols('g')
```

Síly jsou vázané vektory (působí na určitý bod). Potřebujeme sílu o velikosti mg v záporném směru osy y inerciální vztažné soustavy.

```
arm_grav_force_vector = arm_mass * g * (-inertial_frame.y)
```

Vytvoříme n-tici, která bude reprezentovat vázaný vektor působící na těžiště paže.

```
arm_grav_force = (arm_mass_center, arm_grav_force_vector)
```

Stejně tak aplikujeme i na předloktí a ruku.

```
forearm_grav_force_vector = forearm_mass * g * (-inertial_frame.y)
forearm_grav_force = (forearm_mass_center, forearm_grav_force_vector)

hand_grav_force_vector = hand_mass * g * (-inertial_frame.y)
hand_grav_force = (hand_mass_center, hand_grav_force_vector)
```

Kroutící momenty použijeme pro zjednodušení účinku působení svalů na pohyb segmentů vůči sobě. Musíme zadat tři vektory kroutících momentů, které představují celkový moment síly působící na každé tuhé těleso. Nejprve ale musíme zadat tři proměnné, které budou reprezentovat velikost společných momentů: T_s, T_e a T_w.

```
shoulder_torque, elbow_torque, wrist_torque = dynamicsymbols('T_s,
T_e, T_w')
```

Podobně jako u vektorů vázaných sil musíme určit vztažnou soustavu a vektor všech momentů působící na dané tuhé těleso. Vnější kroutící momenty působící na paži znázornit jako vektor kombinující moment v rameni a v lokti (nesmíme zapomenout na 3. Newtonův pohybový zákon).

```
arm_torque_vector = shoulder_torque * inertial_frame.z - elbow_torque
* inertial_frame.z
```

Pro pozdější použití si můžeme uložit n-tici vztažné soustavy a vektoru točivého momentu paže.

```
arm_torque = (arm_frame, arm_torque_vector)
```

Podobně aplikujeme na předloktí a ruku.

```
forearm_torque_vector = elbow_torque * inertial_frame.z - wrist_torque
* inertial_frame.z
forearm_torque = (forearm_frame, forearm_torque_vector)

hand_torque_vector = wrist_torque * inertial_frame.z
hand_torque = (hand_frame, hand_torque_vector)
```

Nyní máme nadefinované všechny potřebné komponenty pro odvození pohybových rovnic pomocí Kaneovy metody.

- Lineární rychlost každého těžiště.
- Úhlová rychlost každého tuhého tělesa.
- Všechny vnější síly působící na mechanismus.

Použijeme funkci *KanesMethod*, která poskytuje výpočet obyčejných diferenciálních rovnic prvního řádu s ohledem na výše uvedené veličiny.

Pro získání kompaktního výsledku bude zapotřebí naimportovat funkce KanesMethod a trigsimp z knihovny SymPy.

```
from sympy import trigsimp
from sympy.physics.mechanics import KanesMethod
```

Kaneova metoda potřebuje vědět následující: zobecněné úhly a úhlové rychlosti, rovnice kinematiky, zatížení, segmenty a vztažné soustavy.

Nejprve si vytvoříme seznam zobecněných úhlů a úhlových rychlostí.

```
coordinates = [theta1, theta2, theta3]
speeds = [omega1, omega2, omega3]
```

Pomocí inerciální vztažné soustavy, souřadnic, rychlostí a rovnic kinematiky můžeme vytvořit funkci *KanesMethod*.

```
kane = KanesMethod(inertial_frame, coordinates, speeds,
kinematical_differential_equations)
```

Uvedeme šestici sil a tři segmenty soustavy.

```
loads = [arm_grav_force,
         forearm_grav_force,
         hand_grav_force,
         arm_torque,
         forearm_torque,
         hand_torque]

bodies = [arm, forearm, hand]
```

Pohybové rovnice již můžeme vypočítat pomocí funkce *kanes_equations*, která má na vstupu šestici sil a trojici segmentů, přičemž na výstupu vrací obyčejné diferenciální rovnice prvního řádu v Kaneově formě:

$$F_r + F_r^* = 0$$

což odpovídá základním Newton-Eulerovým rovnicím:

$$\boldsymbol{F} = \boldsymbol{ma}$$

$$\boldsymbol{T} = \boldsymbol{I}\alpha$$

```
fr, frstar = kane.kanes_equations(bodies, loads)
trigsimp(fr + frstar)
```

Hlavním cílem je mít pohybové rovnice prvního řádu:

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}, t)$$

Pohybové rovnice jsou lineární, pokud jde o derivace zobecněných rychlostí. *KanesMethod* automaticky převádí rovnice do užitečnější formy pro další krok numerické simulace.

$$\boldsymbol{M}(\boldsymbol{y}, t)\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, t)$$

Poznámka:

$$\boldsymbol{g} = \boldsymbol{M}^{-1}(\boldsymbol{x}, t)\boldsymbol{f}(\boldsymbol{x}, t)$$

z čehož vyplývá, že $\boldsymbol{g}$ lze vypočítat analyticky ale pro složitější rovnice je lepší použít numerický řešič. Jednoduše si vygenerujeme matice $\boldsymbol{M}$ a $\boldsymbol{f}$ pro pozdější využití.

K matici hmotnosti $\boldsymbol{M}$ můžeme přistupovat pomocí metody *mass_matrix* a zahrnout do ní také rovnice kinematiky. Toto zvládneme poměrně jednoduše funkcí *trigsimp*.

```
mass_matrix = trigsimp(kane.mass_matrix_full)
```

Pravá strana, je vektorová funkce $\boldsymbol{f}$ všech neinerciálních sil (gyroskopických, vnějších coriolisových atd.):

```
forcing_vector = trigsimp(kane.forcing_full)
```

Když máme vyřešené rovnice symbolicky, tak je potřebujeme transformovat do funkcí Pythonu, které lze použít pro numerickou integraci. Numerická integrace je nutná k vyřešení problému s obyčejnou počáteční hodnotou a umožní nám sledovat vývoj proměnných v čase.

K nastavení číselných hodnot a integraci budeme potřebovat některé funkce od NumPy, ze SciPy pro změnu funkci pro numerickou integraci ODE (Ordinary differential equation). Pro transformaci rovnic ze symbolického tvaru do numerického, použijeme z knihovny *PyDy ODE* generátor funkcí. V neposlední řadě naimportujeme funkce pro vykreslování grafů z *matplotlib*.

```
from numpy import deg2rad, rad2deg, array, zeros, linspace
import numpy as np
from scipy.integrate import odeint
from pydy.codegen.ode_function_generators import generate_ode_function
from matplotlib import pyplot as plt
```

Následujícím krokem je shromáždění všech proměnných v pohybových rovnicích do seznamu. Budeme potřebovat konstanty, souřadnice, rychlosti a zadané vstupy.

```
constants = [arm_length,
             arm_com_length,
             arm_mass,
             arm_inertia,
             forearm_length,
             forearm_com_length,
             forearm_mass,
             forearm_inertia,
             hand_com_length,
             hand_mass,
             hand_inertia,
             g]
```

Integrátory ODE jako *scipy.integrate.odeint*, potřebují funkci, která numericky vyhodnotí pravou stranu ODE prvního řádu. Máme k dispozici symbolickou matici hmotností a vektor sil. Funkce *generate_ode_function* generuje funkci ze symbolických výrazů potřebnou pro *odeint*. Pro vytvoření funkce vložíme pravou stranu.

```
right_hand_side = generate_ode_function(forcing_vector, coordinates,
                                        speeds, constants,
                                        mass_matrix=mass_matrix,
                                        specifieds=specified)
```

Počáteční hodnoty rychlostí nastavíme na nulu.

```
x0 = zeros(6)
```

Následně nastavíme počáteční hodnoty úhlů.

```
x0[0] = deg2rad(-90)
x0[1] = deg2rad(-180)
```

Funkce pravé strany potřebuje doplnit hodnoty všech konstant a hodnoty momentů sil.

```
numerical_constants = array([0.4,  # arm_length [m]
                             0.2,  # arm_com_length [m]
                             5.0,  # arm_mass [kg]
                             0.1,  # arm_inertia [kg*m^2]
                             0.3,  # forearm_length [m]
                             0.15,  # forearm_com_length
                             3.0,  # forearm_mass [kg]
                             0.05,  # forearm_inertia [kg*m^2]
                             0.1,  # hand_com_length [m]
                             1.0,  # hand_mass [kg]
                             0.01,  # hand_inertia [kg*m^2]
                             9.81],  # acceleration due to gravity
[m/s^2])
```

Můžeme vykreslit první tři sloupce *y* a *t*, abychom viděli, jak se mění úhly v čase.

```
plt.plot(t, rad2deg(y[:, :3]))
plt.xlabel('Time [s]')
plt.ylabel('Angle [deg]')
plt.legend(["${}$".format(c) for c in coordinates])
plt.show()
```

Pro zobrazení vizualizace musíme naimportovat funkce z knihoven PyDy, které nám dají tvary aj.

```
from pydy.viz.shapes import Cylinder, Sphere
import pydy.viz
from pydy.viz.visualization_frame import VisualizationFrame
from pydy.viz.scene import Scene
from IPython.display import Image
```

Nejprve musíme vytvořit tvary, které se spojí s tuhým tělesem. Začneme vytvořením koule, která bude reprezentovat kloubový spoj.

```
shoulder_shape = Sphere(color='black', radius=0.1)
elbow_shape = Sphere(color='black', radius=0.1)
wrist_shape = Sphere(color='black', radius=0.1)
```

Nyní vytvoříme vizualizační soustavu pro každý segment. Objekt *VisualizationFrame* dává vztažné soustavě a bodu tvar, protože nezáleží, zda se koule otáčejí, můžeme je jednoduše připojit k inerciální vztažné soustavě.

```
shoulder_viz_frame = VisualizationFrame(inertial_frame, shoulder,
shoulder_shape)
elbow_viz_frame = VisualizationFrame(inertial_frame, elbow,
elbow_shape)
wrist_viz_frame = VisualizationFrame(inertial_frame, wrist,
wrist_shape)
```

Několik bodů budeme potřebovat pro geometrické středy každého ze segmentu. Pro zjednodušení přiřadíme každé vztažné soustavě tvar válce.

```
arm_center = Point('a_c')
forearm_center = Point('f_c')
hand_center = Point('h_c')

arm_center.set_pos(shoulder, arm_length / 2 * arm_frame.y)
forearm_center.set_pos(elbow, forearm_length / 2 * forearm_frame.y)
hand_center.set_pos(wrist, hand_com_length * hand_frame.y)
```

Bude užitečné mít snadný přístup ke konstantám pro určení délek válců.

```
constants_dict = dict(zip(constants, numerical_constants))
```

Vazbou připojíme několik válců ke třem tuhým tělesům.

```
arm_shape = Cylinder(radius=0.08, length=constants_dict[arm_length],
color='blue')
arm_viz_frame = VisualizationFrame('Arm', arm_frame, arm_center,
arm_shape)

forearm_shape = Cylinder(radius=0.08,
length=constants_dict[forearm_length], color='green')
forearm_viz_frame = VisualizationFrame('Forearm', forearm_frame,
forearm_center, forearm_shape)

hand_shape = Cylinder(radius=0.08, length=2 *
constants_dict[hand_com_length], color='red')
hand_viz_frame = VisualizationFrame('Hand', hand_frame, hand_center,
hand_shape)
```

Vytvoříme podmínky, kde vztažná soustava je inerciální vztažnou soustavou a základem je rameno.

```
scene = Scene(inertial_frame, shoulder)
```

Nyní přidáme soustavy, které chceme zobrazit během vizualizace jako seznam.

```
scene.visualization_frames = [shoulder_viz_frame,
                              elbow_viz_frame,
                              wrist_viz_frame,
                              arm_viz_frame,
                              forearm_viz_frame,
                              hand_viz_frame]
```

Poskytneme vizualizaci seznam symbolů, konstant a jejich číselné hodnoty.

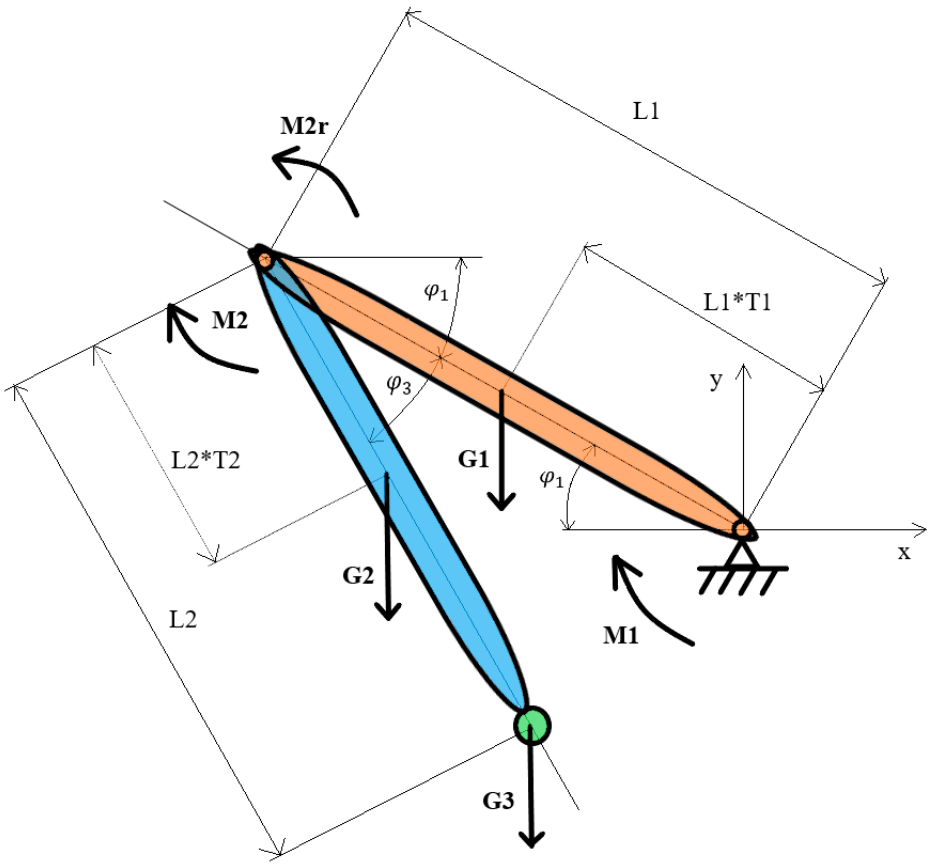
```
scene.states_symbols = coordinates + speeds
scene.constants = constants_dict
scene.states_trajectories = y
```

Nakonec zobrazíme vizualizaci následujícím způsobem:

```
scene.display()
```

4.3 Model hodu s loketním kloubem LR 2. druhu

Pro odvození pohybových rovnic horní končetiny jsme se rozhodli využít Lagrangeovy rovnice 2. druhu se zavedením relativních souřadnic $\varphi_1(t)$ a $\varphi_3(t)$, které jsou proměnné v čase. Zobrazení horní končetiny v relativních souřadnicích $\varphi_1(t)$ a $\varphi_3(t)$ v obecné poloze s tíhami G_1, G_2, a G_3, které působí na paži, předloktí a ruku s míčkem. Do pohybu uvádí končetinu moment síly M1, který působí mezi trupem a paží, M2, který působí mezi paží a předloktím na předloktí, a M2r, který je reakcí od momentu M2 a působí mezi paží a předloktím na paži. Rozměry končetiny jsou L1 a L2 což jsou velikosti paže a předloktí, dále pak L1*T1 a L2*T2 definují vzdálenost působiště tíhové síly na paži a předloktí, přičemž T1 a T2 je bezrozměrné číslo mezi 0 a 1.



Obrázek 9 – Horní končetina v obecné poloze pro LR 2. druhu

Základním postulátem pro využití Lagrangeovy rovnice 2. druhu je postupné dosazení a výpočet všech členů, které následující rovnice obsahuje.

$$\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{q}^j}\right) - \frac{\partial E_k}{\partial q^j} = Q_j \tag{7}$$

Za E_k dosadíme kinetickou energii systému, za q relativní souřadnici a za Q zobecněnou sílu. Za obě strany rovnice dosadíme v libovolném pořadí a dáme do rovnosti, přičemž s dvěma stupni volnosti budeme očekávat dvě parciální diferenciální rovnice 2. řádu. Nejdříve si vyjádříme kinetickou energii systému E_k.

$$E_k = \frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}I_{1S1}\dot{\varphi}_1^2 + \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2) + \frac{1}{2}I_{2S2}\dot{\varphi}_3^2 + \frac{1}{2}m_3(\dot{x}_3^2 + \dot{y}_3^2) \tag{8}$$

Následně musíme zapsat rovnice kinematiky a dosadit do předchozí rovnice

$$\begin{gathered} x_1 = -l_1 T_1 \cos(\varphi_1) \\ \dot{x}_1 = l_1 T_1 \sin(\varphi_1)\,\dot{\varphi}_1 \\ y_1 = l_1 T_1 \sin(\varphi_1) \\ \dot{y}_1 = l_1 T_1 \cos(\varphi_1)\,\dot{\varphi}_1 \\ x_2 = -l_1 \cos(\varphi_1) + l_2 T_2 \cos(\varphi_2) \\ \dot{x}_2 = l_1 \sin(\varphi_1)\,\dot{\varphi}_1 - l_2 T_2 \sin(\varphi_1 + \varphi_3)\,(\dot{\varphi}_1 + \dot{\varphi}_3) \\ y_2 = l_1 \sin(\varphi_1) - l_2 T_2 \sin(\varphi_2) \\ \dot{y}_2 = l_1 \cos(\varphi_1)\,\dot{\varphi}_1 - l_2 T_2 \cos(\varphi_1 + \varphi_3)\,(\dot{\varphi}_1 + \dot{\varphi}_3) \\ x_3 = -l_1 \cos(\varphi_1) + l_2 \cos(\varphi_2) \\ \dot{x}_3 = l_1 \sin(\varphi_1)\,\dot{\varphi}_1 - l_2 \sin(\varphi_1 + \varphi_3)\,(\dot{\varphi}_1 + \dot{\varphi}_3) \\ y_3 = l_1 \sin(\varphi_1) - l_2 \sin(\varphi_2) \\ \dot{y}_3 = l_1 \cos(\varphi_1)\,\dot{\varphi}_1 - l_2 \cos(\varphi_1 + \varphi_3)\,(\dot{\varphi}_1 + \dot{\varphi}_3) \end{gathered} \tag{9}$$

$$\begin{aligned} E_k = \frac{1}{2}\dot{\varphi}_1^2 (I_{1S1} + m_1 l_1^2 T_1^2 + m_2 l_1^2 + m_3 l_1^2) + \frac{1}{2}(\dot{\varphi}_1^2 + 2\dot{\varphi}_1\dot{\varphi}_3 + \dot{\varphi}_3^2) \\ \cdot (I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 \cdot (m_2 T_2 + m_3) \\ \cdot \cos\varphi_3 \cdot (\dot{\varphi}_1^2 + \dot{\varphi}_1\dot{\varphi}_3) \end{aligned} \tag{10}$$

Následné výpočty budou probíhat v závislosti pro který segment sestavujeme pohybovou rovnici. Proto můžeme přejít k pravé straně rovnice a vyjádřit si zobecněný silový účinek například pomocí metody virtuálních prací.

$$Q\partial\varphi_j = M_1\partial\varphi_1 - G_1\partial y_1 + M_2\partial\varphi_3 - M_2\partial\varphi_1 - G_2\partial y_2 - G_3\partial y_3 \tag{11}$$

Nyní již máme obecný postup, společný pro obě souřadnice, hotový a můžeme přejít k sestavování konkrétních rovnic pro proměnné $\varphi_1(t)$ a $\varphi_3(t)$.

Nejprve provedeme derivaci kinetické energie podle derivace obou proměnných a následně podle času tím dostaneme první člen do Lagrangeovy rovnice.

$$\begin{aligned}\frac{\partial E_k}{\partial \dot{\varphi}_1} = \dot{\varphi}_1(I_{1S1} + m_1 l_1^2 T_1^2 + m_2 l_1^2 + m_3 l_1^2) \\ + (\dot{\varphi}_1 + \dot{\varphi}_3)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 \\ \cdot (m_2 T_2 + m_3)\cos\varphi_3\,(2\dot{\varphi}_1 + \dot{\varphi}_3)\end{aligned} \tag{12a}$$

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{\varphi}_1}\right) = \ddot{\varphi}_1(I_{1S1} + m_1 l_1^2 T_1^2 + m_2 l_1^2 + m_3 l_1^2) \\ + (\ddot{\varphi}_1 + \ddot{\varphi}_3)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 \\ \cdot (m_2 T_2 + m_3)\left[cos\,\varphi_3 \cdot (2\ddot{\varphi}_1 + \ddot{\varphi}_3)\right. \\ \left. - sin\,\varphi_3\left(2\dot{\varphi}_1\dot{\varphi}_3 + {\dot{\varphi}_3}^2\right)\right]\end{aligned} \tag{12b}$$

$$\begin{aligned}\frac{\partial E_k}{\partial \dot{\varphi}_3} = (\dot{\varphi}_1 + \dot{\varphi}_3)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 \\ \cdot (m_2 T_2 + m_3)\cos\varphi_3\,\dot{\varphi}_1\end{aligned} \tag{13a}$$

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{\varphi}_3}\right) = (\ddot{\varphi}_1 + \ddot{\varphi}_3)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 \\ \cdot (m_2 T_2 + m_3)(\ddot{\varphi}_1\,cos\,\varphi_3 - \dot{\varphi}_1\dot{\varphi}_3\,sin\,\varphi_3)\end{aligned} \tag{13b}$$

Druhý člen rovnice je derivací kinetické energie podle každé proměnné.

$$\frac{\partial E_k}{\partial \varphi_1} = 0 \tag{14}$$

$$\frac{\partial E_k}{\partial \varphi_3} = l_1 l_2 \cdot (m_2 T_2 + m_3) \cdot \sin\varphi_3 \cdot (\dot{\varphi}_1^2 + \dot{\varphi}_1\dot{\varphi}_3) \tag{15}$$

Tímto je levá strana rovnice vyřešená. Můžeme tak přejít na rozepsání silových účinků pomocí rovnic kinetiky, aby na levé straně byly všechny členy závislé na proměnných.

$$\begin{aligned}Q\partial\varphi_j = M_1\partial\varphi_1 + M_2\partial\varphi_3 - M_2\partial\varphi_1 - m_1 g l_1 T_1 \cos(\varphi_1)\,\partial\varphi_1 \\ - m_2 g[l_1\cos(\varphi_1)\,\partial\varphi_1 \\ - l_2 T_2\cos(\varphi_1 + \varphi_3)\,(\partial\varphi_1 + \partial\varphi_3)] \\ - m_3 g[l_1\cos(\varphi_1)\,\partial\varphi_1 \\ - l_2\cos(\varphi_1 + \varphi_3)\,(\partial\varphi_1 + \partial\varphi_3)]\end{aligned} \tag{16}$$

Výsledný tvar pravé strany pro každou proměnnou získáme tím, že z předešlé rovnice vytkneme elementy požadované proměnné a element druhé proměnné budeme považovat rovný nule.

$$\begin{aligned} Q_1 = M_1 - M_2 &- m_1 g l_1 T_1 \cos(\varphi_1) \\ &- m_2 g [l_1 \cos(\varphi_1) - l_2 T_2 \cos(\varphi_1 + \varphi_3)] \\ &- m_3 g [l_1 \cos(\varphi_1) - l_2 \cos(\varphi_1 + \varphi_3)] \end{aligned} \tag{17}$$

$$Q_3 = M_2 - m_2 g l_2 T_2 \cos(\varphi_1 + \varphi_3) - m_3 g l_2 \cos(\varphi_1 + \varphi_3) \tag{18}$$

Levou a k ní náležitou pravou stranu dáme do rovnosti a vznikne nám výsledný tvar pohybové rovnice pro horní končetinu.

$$\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{\varphi}_1}\right) - \frac{\partial E_k}{\partial \varphi_1} = Q_1 \tag{19}$$

$$\begin{aligned} \ddot{\varphi}_1 (I_{1S1} + m_1 l_1^2 T_1^2 + m_2 l_2^2 + m_3 l_1^2) &\\ + (\ddot{\varphi}_1 + \ddot{\varphi}_3)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 &\\ \cdot (m_2 T_2 + m_3)[(2\ddot{\varphi}_1 + \ddot{\varphi}_3)\cos\varphi_3 &\\ - (2\dot{\varphi}_1 \dot{\varphi}_3 + \dot{\varphi}_3^2)\sin\varphi_3] &\\ = M_1 - M_2 - (m_1 T_1 + m_2 + m_3) g l_1 \cos(\varphi_1) &\\ + (m_2 T_2 + m_3) g l_2 \cos(\varphi_1 + \varphi_3) & \end{aligned} \tag{20}$$

$$\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{\varphi}_3}\right) - \frac{\partial E_k}{\partial \varphi_3} = Q_3 \tag{21}$$

$$\begin{aligned} (\ddot{\varphi}_1 + \ddot{\varphi}_3)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 &\\ \cdot (m_2 T_2 + m_3)(\ddot{\varphi}_1 \cos\varphi_3 + \dot{\varphi}_1^2 \sin\varphi_3) &\\ = M_2 - g l_2 (m_2 T_2 + m_3)\cos(\varphi_1 + \varphi_3) & \end{aligned} \tag{22}$$

Následně dvě rovnice druhého řádu převedeme na čtyři rovnice prvního řádu. Pro převod musíme provést substituci proměnných následujícím způsobem.

$$\begin{aligned} \varphi_1 = \alpha_1;\ \dot{\varphi}_1 = \alpha_2;\ \ddot{\varphi}_1 = \dot{\alpha}_2 \\ \varphi_3 = \alpha_3;\ \dot{\varphi}_3 = \alpha_4;\ \ddot{\varphi}_3 = \dot{\alpha}_4 \end{aligned} \tag{23}$$

A vyjádříme rovnice.

$$\dot{\alpha}_1 = \alpha_2 \tag{24}$$

$$\dot{\alpha}_3 = \alpha_4 \tag{25}$$

$$\begin{aligned}
&\dot{\alpha}_2(I_{1S1} + m_1 l_1^2 T_1^2 + m_2 l_1^2 + m_3 l_1^2) \\
&\quad + (\dot{\alpha}_2 + \dot{\alpha}_4)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 \\
&\quad \cdot (m_2 T_2 + m_3)[(2\dot{\alpha}_2 + \dot{\alpha}_4)\cos\alpha_3 \\
&\quad - (2\alpha_2\alpha_4 + \alpha_4^2)\sin\alpha_3] \\
&\quad = M_1 - M_2 - (m_1 T_1 + m_2 + m_3) g l_1 \cos(\alpha_1) \\
&\quad + (m_2 T_2 + m_3) g l_2 \cos(\alpha_1 + \alpha_3)
\end{aligned} \tag{26}$$

$$\begin{aligned}
&(\dot{\alpha}_2 + \dot{\alpha}_4)(I_{2S2} + m_2 l_2^2 T_2^2 + m_3 l_2^2) - l_1 l_2 \\
&\quad \cdot (m_2 T_2 + m_3)(\dot{\alpha}_2 \cos\alpha_3 - \alpha_2^2 \sin\alpha_3) \\
&\quad = M_2 - g l_2 (m_2 T_2 + m_3)\cos(\alpha_1 + \alpha_3)
\end{aligned} \tag{27}$$

Nyní je zapotřebí převést rovnice do tvaru, který dokážeme vypočítat. Python obsahuje knihovnu SciPy, která je určená pro numerické výpočty. My budeme potřebovat nástroj pro numerickou integraci, například scipy.integrate.solve_ivp, ten ovšem vyžaduje zadání rovnic v přesně stanoveném tvaru: [9]

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, t) \tag{28}$$

S počátečními podmínkami

$$\boldsymbol{y}(0) = y_0 \tag{29}$$

Výsledný tvar pro numerickou integraci viz příloha 1

Pro symbolické odvození Lagrangeových rovnic druhého druhu a převod do tvaru pro výpočet jsme použili výpočetní program Python a jeho knihovnu SymPy, pro následný výpočet rovnic knihovnu SciPy.

4.4 Inerciální vlastnosti lidského těla

Váhy segmentů horní končetiny určíme z následujících rovnic:

$$m_1 = 0{,}0312 * M - 0{,}0027 * H + 0{,}25 \quad (30a)$$

$$m_2 = 0{,}01445 * M - 0{,}00114 * H + 0{,}3185 \quad (30b)$$

$$m_3 = 0{,}0036 * M + 0{,}00175 * H - 0{,}1165 + 0{,}15 \quad (30c)$$

Délky končetin jsou dlouhé v průměru $L_1 = 0{,}35m$ a $L_2 = 0{,}3m$ pro člověka vysokého 180cm. [10] [11]

Jedntolivé segmenty horní končetiny budeme uvažovat jako válce rotující kolem těžiště, tudíž budeme brát v potaz též následující momenty setrvačnosti:

$$I_{1s1} = \frac{1}{12} m_1 l_1^2 \quad (31a)$$

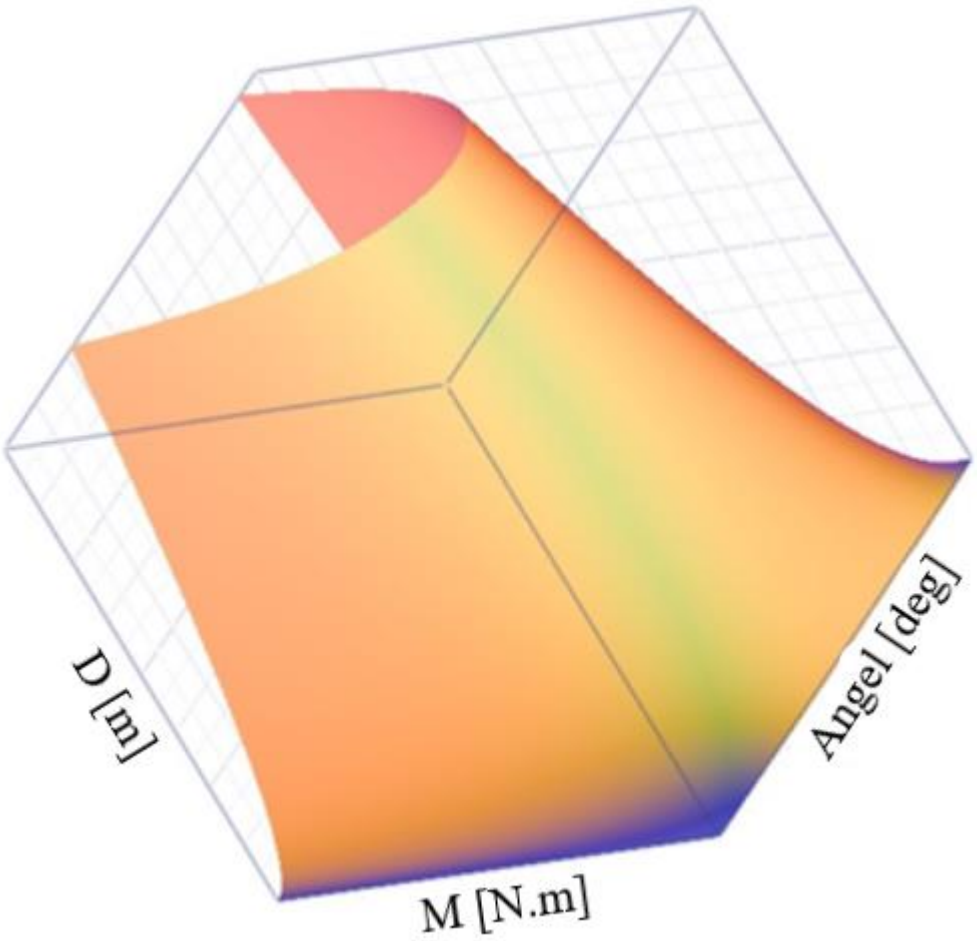
$$I_{2s2} = \frac{1}{12} m_2 l_2^2 \quad (31b)$$

5 Výsledky

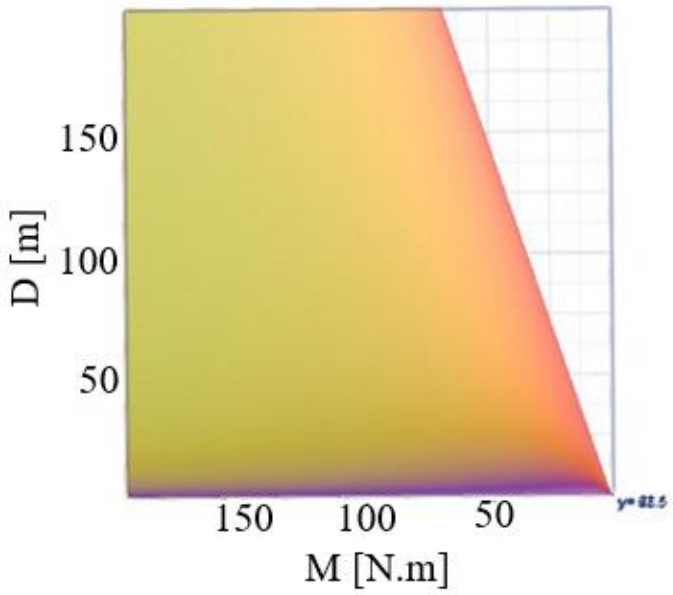
5.1 Model hodu bez loketního kloubu

Pro názorné předvedení dané rovnice, použijeme vykreslení do x, y a z souřadnicového systému, kde za osu x dosadíme vzdálenost hodu D $[m]$, za y úhel paže při vyhození φ_H [°] a za z moment působící v rameni M $[N.m]$.

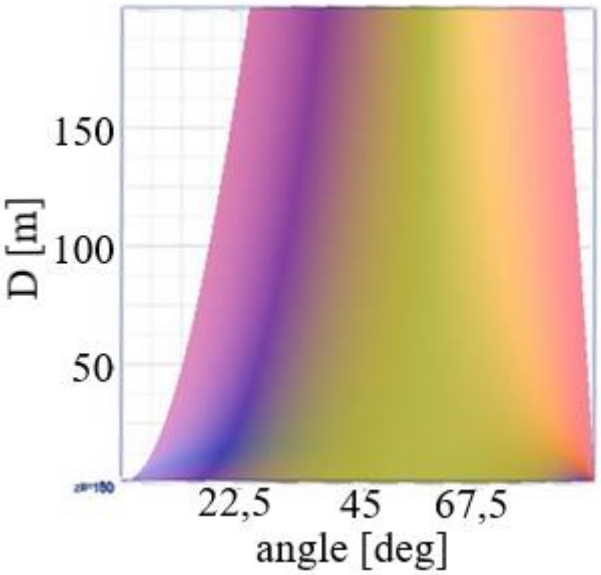
Z grafů je patrné, že nezáleží pouze na úhlu paže při vyhození ale zároveň na momentu, který čím déle působí, tím větší úhlovou rychlost, potažmo výslednou počáteční rychlost hozeného předmětu, generuje. Maximum křivky, kde $y(\varphi_H) = 52° \rightarrow$ $úhel\ vyhození$ 38°, je lineární závislostí momentu M na vzdálenosti hodu D.



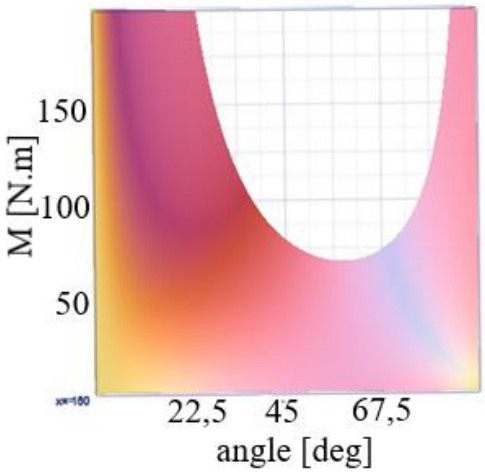
Obrázek 5 – Pohled na 3D graf hodu horní končetiny bez lokte



Obrázek 6 – Pohled na osu vzdálenosti hodu a momentu působící v rameni



Obrázek 7 - Pohled na osu vzdálenosti hodu a úhlu paže při vyhození



Obrázek 8 – Pohled na osu momentu a úhlu paže při vyhození

5.2 PyDy multibody dynamics

Základní nastavení pro určení momentů nejbližších skutečnosti, jsme určovali v závislosti na světovém rekordu zapsaném v Guinessově knize rekordů v hodu kriketovým míčkem (150 g), který činí 128,6m a byl vytvořen Robertem Percivalem. V našem případě, když neuvažujeme rotaci trupu a hod v prostoru, tak bude reálný dohoz nižší. Cíl dohodu v našem případě bude 70 m. [12]

Váhy segmentů horní končetiny určíme z následujících rovnic:

$$m_1 = 0{,}0312 * M - 0{,}0027 * H + 0{,}25 \quad (31a)$$

$$m_2 = 0{,}01445 * M - 0{,}00114 * H + 0{,}3185 \quad (31b)$$

$$m_3 = 0{,}0036 * M + 0{,}00175 * H - 0{,}1165 + 0{,}15 \quad (31c)$$

, kde m_i jsou hmotnosti jednotlivých segmentů a míčku

M je hmotnost člověka

H je výška člověka. [13]

Pro názornost určím člověka vysokého 180 cm a těžkého 80 kg.

$$m_1 = 2{,}26\ kg$$

$$m_2 = 1{,}2692\bar{9}\ kg$$

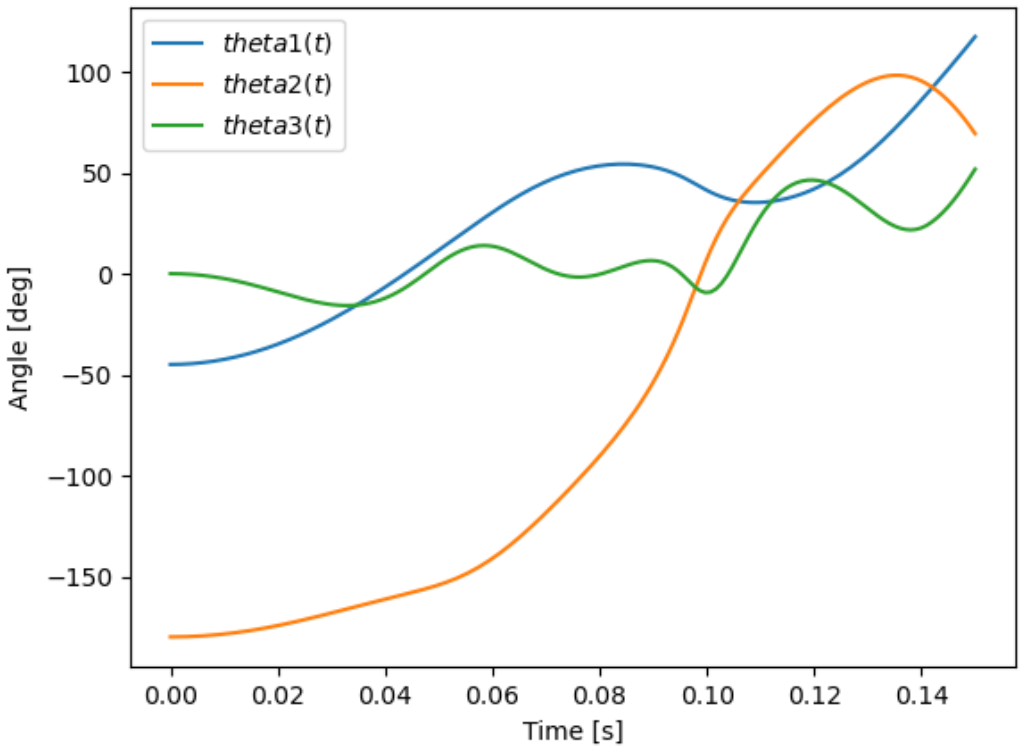
$$m_3 = 0{,}6364\bar{9}\ kg$$

Délky končetin jsou v průměru $L_1 = 0{,}35m$ a $L_2 = 0{,}3m$ pro člověka vysokého 180cm. [10] [11]

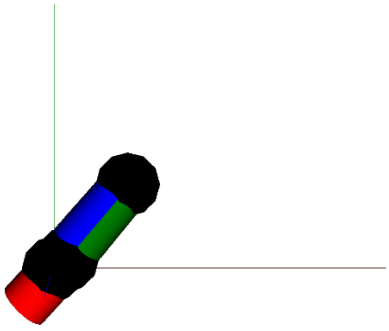
Tabulka 5.1 – 1 – Základní vstupní parametry

L1 [m]	L2 [m]	M1 [N.m]	M2 [N.m]	M3 [N.m]
0,35	0,3	130	30	10

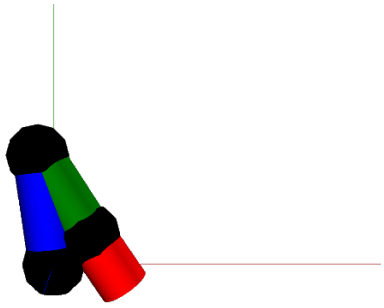
V čase 78 ms došlo k vypuštění, vzdálenost dohozu činila 68,02m. Na následujících obrázkách můžeme vidět průběh úhlů a model pohybu paže v čase při hodu.



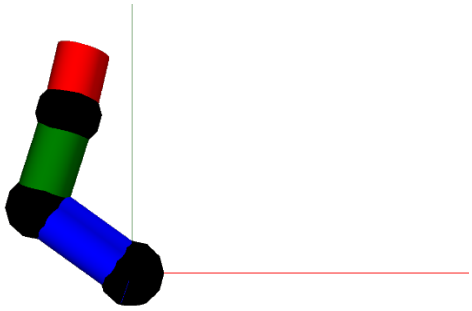
Obrázek 10 - Grafické znázornění průběhu úhlů v čase.



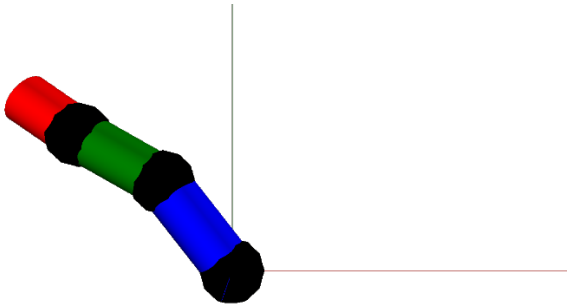
Obrázek 11 – Pozice ruky v čase 25 ms



Obrázek 12 – Pozice ruky v čase 50 ms



Obrázek 13 – Pozice ruky v čase 75 ms



Obrázek 14 – Pozice ruky v čase 100 ms

Tyto hodnoty budeme brát za skutečné, ke kterým dochází při hodu a budeme uvažovat, že úpony svalů a jejich síla je neměnná, tedy momenty jsou konstantní a budeme měnit pouze délky jednotlivých segmentů.

Pro následný výpočet využijeme delší krok zvětšování velikosti jednotlivých segmentů, kvůli úspoře výpočetního času.:

$L_1 \in \langle 0{,}1; 1\rangle\ [m]$

$L_2 \in \langle 0{,}1; 1\rangle\ [m]$

Časový krok 0.001 s

Časový interval $\langle 0; 1\rangle\ [s]$

Maximální vzdálenost hodu v čase vypuštění $t = 87\ ms$ byla 143,48 m pro $L_1 = 0{,}4\ m$ a $L_2 = 0{,}4\ m$

Orientačně tedy víme, že ideální velikosti jsou $L_1 = 0{,}4\ m$ a $L_2 = 0{,}4\ m$, tudíž poměr 1:1 Pro zpřesnění nastavíme menší velikostní krok pro výpočet, ale zároveň menší interval jednotlivých velikostí, kvůli úspoře výpočetního času.

$L_1 \in \langle 0{,}1; 0{,}5 \rangle\ [m]$

$L_2 \in \langle 0{,}1; 0{,}5 \rangle\ [m]$

Časový krok 0.001 s

Krok zvětšování paže = 0,05 m

Časový interval $\langle 0; 0{,}2 \rangle\ [s]$

Maximální vzdálenost hodu 146,79 m byla v čase vypuštění $t = 89\ ms$ pro $L_1 = 0{,}4\ m$ a $L_2 = 0{,}45\ m$

5.3 Lagrangeovy rovnice 2. druhu

Výsledky, zda změna poměru délek ruky vedla ke zlepšení dohozu ověříme pomocí pohybových rovnic odvozených z Lagrangeovy rovnice druhého druhu.

Pro ověření dosadíme do rovnic generalizované parametry z PyDy modelu, tedy:

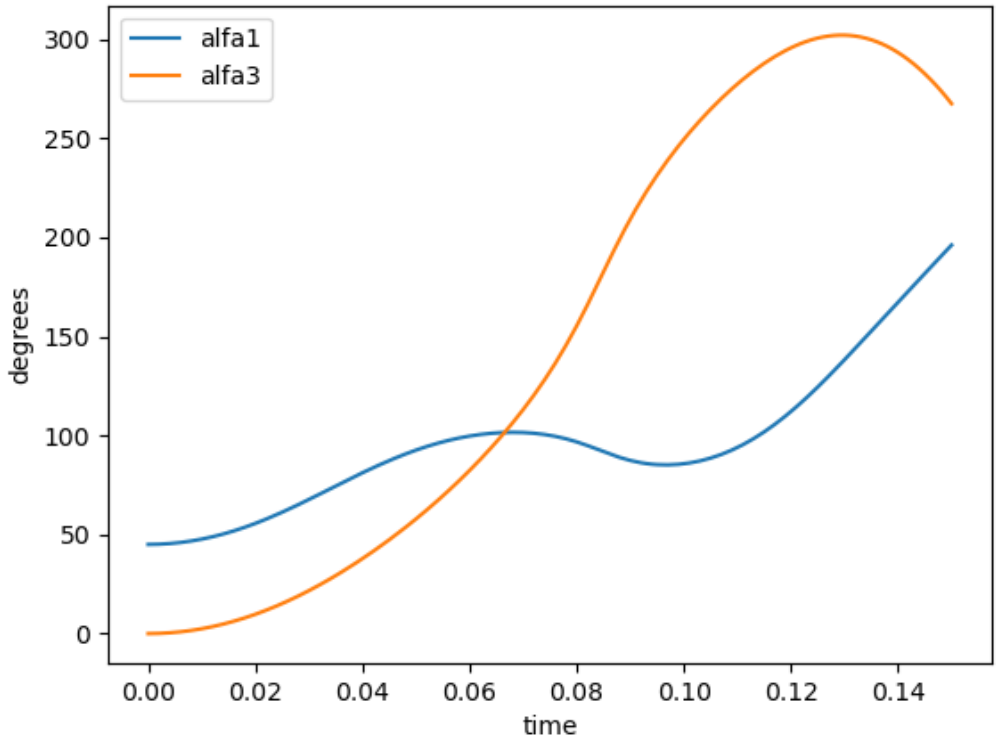
L1 = 0,35m

L2 = 0,3m

M1 = 130 N.m

M2 = 30 N.m

Vzdálenost dohozu činila 37,97 m v čase 95 ms. Průběhy úhlů v čase můžeme vidět na obrázku 15.



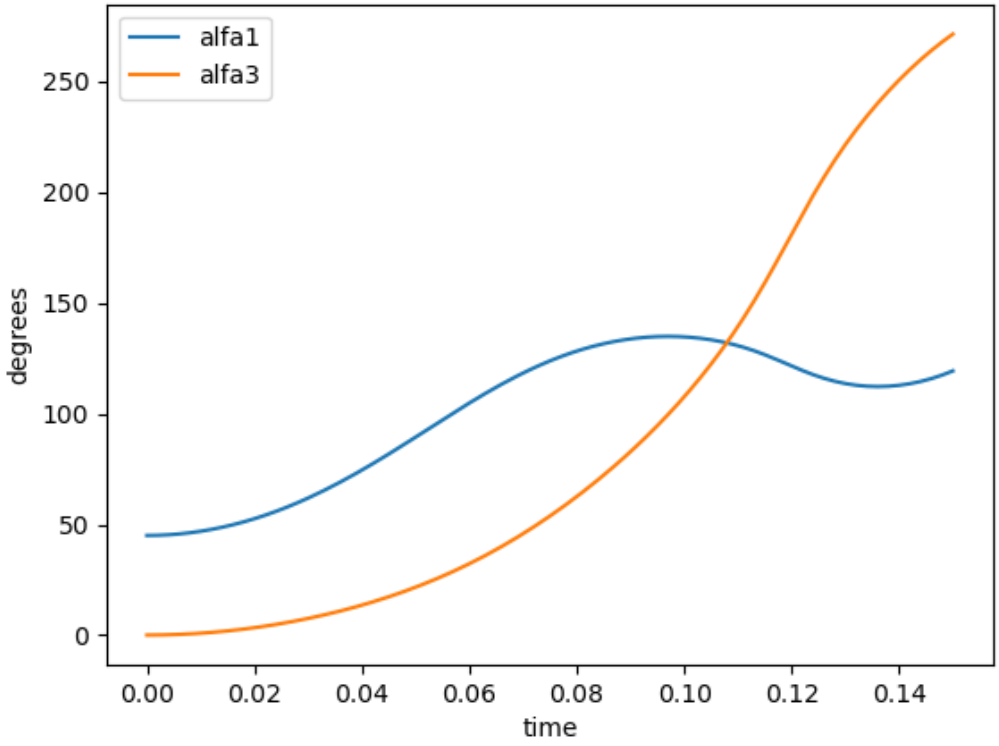
Obrázek 15 – Vývoj úhlů paže a předloktí pro LR 2. druhu, základní nastavení

Nyní nastavíme parametry optimalizovaných délek segmentů končetiny.

$L_1 = 0{,}4\ m$

$L_2 = 0{,}45\ m$

Přičemž délka hodu se skutečně prodloužila a to na 40.98 m v čase 128 ms od počátku pohybu ruky.



Obrázek 16 – Vývoj úhlů paže a předloktí pro LR 2. druhu, ověření optimalizace

Graf závislosti úhlu na čase pro alfa1 a alfa3

6 Diskuse

Chtěli jsme dokázat, zda je lidská horní končetina optimalizována k házení. Z našeho zjednodušeného modelu nám vyšlo, že ideální délka je 40 cm pro paži a 45 cm pro předloktí, z toho je patrné že ideální poměr délek paže ku předloktí je 1:1,125. Přičemž publikace od Dr. Özlem UZUN a kol., jejímž cílem bylo identifikace pohlaví podle parametrů horní končetiny, tvrdí že u 400 vysokoškolských studentů a studentek ve věku 18 až 25, u kterých proběhlo měření, byla průměrná délka pravé mužské paže 38,6±2,9 cm a předloktí plus polovina ruky 38,155±2,715. Průměrná délka pravé ženské paže pak 34,89±2,29 cm a předloktí plus polovina ruky 34,16±2,98 cm. Poměry délek pravé paže ku předloktí vychází pro muže 1:0,99 a pro ženy 1:0,98. [14] [15] [16]

Oproti reálnému hodu, náš model vycházel pouze z fixního momentu, přičemž skutečnost je taková, že v případě kontrakce svalů se mění i svalová síla, která generuje moment. Dále jsme nepředpokládali složitější pohyb v prostoru. Zanedbali jsme například pohyb ramene, trupu, boků aj., který mají velký podíl při hodu, zejména kvůli generování další rotace a švihové síly, která se též ve vzdálenosti hodu projeví.

Tato práce je pilotní studií, která ukazuje, že člověk svým vývojem je blízko ideálního poměru pro maximalizaci délky hodu, ale pro dokončení bychom potřebovali zlepšit naše chápání ruky a těla jako komplexního mechanismu, nebo porovnat délky končetiny v průběhu vývoje člověka. Pomocí videoanalýzy bychom mohli získat reálný průběh pohybu a momentů sil, mj. je házení spojeno s rotací trupu, což by se projevilo na antropometrii páteře u kosterních pozůstatků. [17]

7 Závěr

Cílem práce bylo prokázat, že lidská horní končetina je optimalizována pro házení. Pro tento účel byla vytvořena série matematických modelů, které v jednoduchém případě bylo možné řešit analyticky, nebo následně numericky. V případě prvního modelu pomocí programového balíčku PyDy, v případě druhého modelu pak formulací Lagrangeových rovnic pomocí symbolického balíčku SymPy a následný výpočet pomocí NumPy.

Naše výsledky ukazují, že optimální délky hodu je možné dosáhnout v případě délky paže a předloktí v poměru 1:1,125. Tyto rozměry jsou v souladu s rozměrem paže a předloktí, které pozorujeme u člověka moderního typu.

Naše výsledky poukazují na to, že délky jednotlivých částí jsou optimalizovány právě k hodu.

8 Bibliografie

[1] DELLA VALLE, Craig J., Andrew S. ROKITO, Maureen G. BIRDZELL a Joseph D. ZUCKERMAN. Biomechanics of the Shoulder. NORDIN, Margareta a Victor H. FRANKEL. *Basic Biomechanics of the Musculoskeletal*. Fourth edition. Philadelphia: Lippincott Williams and Wilkins, 2012, s. 319-339. ISBN 9781609133351.

[2] *Shoulder Flexion: 4 Important Muscles to Memorize For NASM* [online]. [cit. 2021-08-15]. Dostupné z: https://cz.pinterest.com/pin/86131411607550189/

[3] What's the Difference Between Abduction and Adduction? (Biomechanics). *Machine Design* [online]. 2014 [cit. 2021-08-15]. Dostupné z: https://www.machinedesign.com/markets/medical/article/21831782/whats-the-difference-between-abduction-and-adduction-biomechanics

[4] JAZRAWI, Laith M., Andrew S. ROKITO, Maureen G. BIRDZELL a Joseph D. ZUCKERMAN. Biomechanics of the Elbow. NORDIN, Margareta a Victor H. FRANKEL. *Basic Biomechanics of the Musculoskeletal*. Fourth edition. Philadelphia: Lippincott Williams and Wilkins, 2012, s. 341-357. ISBN 9781609133351.

[5] BARR, Ann E. a Jane BEAR-LEHMAN. Biomechanics of the Wrist and Hand. NORDIN, Margareta a Victor H. FRANKEL. *Basic Biomechanics of the Musculoskeletal System*. Fourth edition. Philadelphia: Lippincott Williams and Wilkins, 2012, s. 359-387. ISBN 9781609133351.

[6] S. FLEISIG, Glenn, Steven W. BARRANTINE, Rafael F. ESCAMILLA a James R. ANDREWS. Biomechanics of Overhand Throwing with Implications for Injuries. *American Sports Medicine Institute*. Birmingham (Alabama): American Sports Medicine Institute, 1996, , 421-437. Dostupné z: doi:0112-1642/96/0006.0421/508.50/0

[7] E. ATWATER, Anne. Biomechanics of overarm throwing movements and of throwing injuries. *Exercise and Sport Sciences Reviews*. 1979, **1979**, 43-85. Dostupné z: doi:10.1249/00003677-197900070-00004

[8] Vrh šikmý. *Encyklopedie fyziky* [online]. [cit. 2021-08-15]. Dostupné z: http://fyzika.jreichl.com/main.article/view/68-vrh-sikmy

[9] Integration (scipy.integrate): Ordinary differential equations (solve_ivp). *SciPy documentation* [online]. 2021 [cit. 2021-08-13]. Dostupné z: https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html?fbclid=IwAR3S6QHlF4IRLZI27-D3z_GDp2SzU0LCK3S5yPMZlecsruXtx8ThFzFUfFk#ordinary-differential-equations-solve-ivp

[10] SIZE GUIDE V3. *UPPER LIMB CO* [online]. Melbourn [cit. 2021-08-13]. Dostupné z: https://upperlimbco.com/pronosupinator/size-guide-v3/

[11] GHOLAMREZA, Hassanzadeh. Determination of Stature from Upper Arm Length in Medical Students. *RESEARCH PAPERS*. Tehran, 2014, **11**(3), 135-139.

[12] Longest throw of a cricket ball. *Guiness world records* [online]. London [cit. 2021-08-13]. Dostupné z: https://www.guinnessworldrecords.com/world-records/64457-longest-throw-of-a-cricket-ball

[13] ZACIORSKIJ, V. M., A. S. ARUIN a V. N. SELUJANOV. *Biomechanika dvigatělnogo apparata čeloveka*. FiS, Moskva, 1981, **118**, 38-39.

[14] UZUN, Özlem. Evaluation of upper extremity anthropometric measurements in terms of sex estimation. *ResearchGate*. 2018, , 42-50. ISSN 2320-6012. Dostupné z: doi:10.18203/2320-6012.ijrms20175709

[15] UZUN, Özlem. Estimation of Stature from Upper Extremity Anthropometric Measurements. *JCDR*. 2019, (13), 9-15. Dostupné z: doi:10.7860/JCDR/2019/38372.12475

[16] RASHID, Zuli'zam. Comparison of Malaysian and SAE J833 Anthropometric Proportions for Vehicle Package Design. *Advanced Engineering Forum*. 2013, (10), 337-344. Dostupné z: doi:10.4028/www.scientific.net/AEF.10.336

[17] THOMAS ROACH, Neil. Upper body contributions to power generation during rapid, overhand throwing in humans. *Journal of Experimental Biology*. ResearchGate, 2014, , 2139-2149. Dostupné z: doi:10.1242/jeb.103275

Přílohy

1 Rovnice pro numerickou integraci

alfa1_dot = *alfa2*

alfa2_dot = *(I2s2*M1 - I2s2*M2 - I2s2*T1*g*l1*m1*np.cos(alfa1) +*
*I2s2*T2*g*l2*m2*np.cos(alfa1 + alfa3) -*
*2*I2s2*T2*l1*l2*m2*np.sin(alfa3)*alfa2*alfa4 -*
*I2s2*T2*l1*l2*m2*np.sin(alfa3)*alfa4**2 - I2s2*g*l1*m2*np.cos(alfa1) -*
*I2s2*g*l1*m3*np.cos(alfa1) + I2s2*g*l2*m3*np.cos(alfa1 + alfa3) -*
*2*I2s2*l1*l2*m3*np.sin(alfa3)*alfa2*alfa4 - I2s2*l1*l2*m3*np.sin(alfa3)*alfa4**2 +*
*M1*T2**2*l2**2*m2 + M1*l2**2*m3 - 2*M2*T2**2*l2**2*m2 +*
*M2*T2*l1*l2*m2*np.cos(alfa3) + M2*l1*l2*m3*np.cos(alfa3) - 2*M2*l2**2*m3 -*
*T1*T2**2*g*l1*l2**2*m1*m2*np.cos(alfa1) - T1*g*l1*l2**2*m1*m3*np.cos(alfa1) -*
*T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2**2 -*
*2*T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2*alfa4 -*
*T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa4**2 +*
*T2**2*g*l1*l2**2*m2**2*np.cos(alfa1 + 2*alfa3)/2 -*
*T2**2*g*l1*l2**2*m2**2*np.cos(alfa1)/2 - T2**2*g*l1*l2**2*m2*m3*np.cos(alfa1)*
*+ T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa2**2/2 -*
*T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 -*
*2*T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 -*
*T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 +*
*T2*g*l1*l2**2*m2*m3*np.cos(alfa1 + 2*alfa3) +*
*T2*g*l1*l2**2*m2*m3*np.cos(alfa1) +*
*T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa2**2 -*
*T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 -*
*2*T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 –*
*T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 - g*l1*l2**2*m2*m3*np.cos(alfa1) +*
*g*l1*l2**2*m3**2*np.cos(alfa1 + 2*alfa3)/2 - g*l1*l2**2*m3**2*np.cos(alfa1)/2 +*
*l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa2**2/2 -*
*l1*l2**3*m3**2*np.sin(alfa3)*alfa2**2 -*
*2*l1*l2**3*m3**2*np.sin(alfa3)*alfa2*alfa4 -*

*l1*l2**3*m3**2*np.sin(alfa3)*alfa4**2)/(I1s1*I2s2 + I1s1*T2**2*l2**2*m2 + I1s1*l2**2*m3 + I2s2*T1**2*l1**2*m1 + I2s2*T2**2*l2**2*m2 - 2*I2s2*T2*l1*l2*m2*np.cos(alfa3) + I2s2*l1**2*m2 + I2s2*l1**2*m3 - 2*I2s2*l1*l2*m3*np.cos(alfa3) + I2s2*l2**2*m3 + T1**2*T2**2*l1**2*l2**2*m1*m2 + T1**2*l1**2*l2**2*m1*m3 + T2**2*l1**2*l2**2*m2**2*np.sin(alfa3)**2 + T2**2*l1**2*l2**2*m2*m3 + 2*T2*l1**2*l2**2*m2*m3*np.sin(alfa3)**2 - 2*T2*l1**2*l2**2*m2*m3 + l1**2*l2**2*m2*m3 + l1**2*l2**2*m3**2*np.sin(alfa3)**2)*

alfa3_dot *= alfa4*

alfa4_dot *= (I1s1*M2 + I1s1*T2*g*l2*m2*np.cos(alfa1 + alfa3) + I1s1*T2*l1*l2*m2*np.sin(alfa3)*alfa2**2 + I1s1*g*l2*m3*np.cos(alfa1 + alfa3) + I1s1*l1*l2*m3*np.sin(alfa3)*alfa2**2 - M1*T2**2*l2**2*m2 + M1*T2*l1*l2*m2*np.cos(alfa3) + M1*l1*l2*m3*np.cos(alfa3) - M1*l2**2*m3 + M2*T1**2*l1**2*m1 + 2*M2*T2**2*l2**2*m2 - 3*M2*T2*l1*l2*m2*np.cos(alfa3) + M2*l1**2*m2 + M2*l1**2*m3 - 3*M2*l1*l2*m3*np.cos(alfa3) + 2*M2*l2**2*m3 + T1**2*T2*g*l1**2*l2*m1*m2*np.cos(alfa1 + alfa3) + T1**2*T2*l1**3*l2*m1*m2*np.sin(alfa3)*alfa2**2 + T1**2*g*l1**2*l2*m1*m3*np.cos(alfa1 + alfa3) + T1**2*l1**3*l2*m1*m3*np.sin(alfa3)*alfa2**2 + T1*T2**2*g*l1*l2**2*m1*m2*np.cos(alfa1) - T1*T2*g*l1**2*l2*m1*m2*np.cos(alfa1 - alfa3)/2 - T1*T2*g*l1**2*l2*m1*m2*np.cos(alfa1 + alfa3)/2 - T1*g*l1**2*l2*m1*m3*np.cos(alfa1 - alfa3)/2 - T1*g*l1**2*l2*m1*m3*np.cos(alfa1 + alfa3)/2 + T1*g*l1*l2**2*m1*m3*np.cos(alfa1) + T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2**2 + 2*T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2*alfa4 + T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa4**2 - T2**2*g*l1*l2**2*m2**2*np.cos(alfa1 + 2*alfa3)/2 + T2**2*g*l1*l2**2*m2**2*np.cos(alfa1)/2 + T2**2*g*l1*l2**2*m2*m3*np.cos(alfa1) - T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa2**2 -*

```
T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa2*alfa4 -
T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa4**2/2 +
T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 +
2*T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 +
T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 - T2*g*l1**2*l2*m2**2*np.cos(alfa1
- alfa3)/2 + T2*g*l1**2*l2*m2**2*np.cos(alfa1 + alfa3)/2 -
T2*g*l1**2*l2*m2*m3*np.cos(alfa1 - alfa3)/2 + T2*g*l1**2*l2*m2*m3*np.cos(alfa1
+ alfa3)/2 - T2*g*l1*l2**2*m2*m3*np.cos(alfa1 + 2*alfa3) -
T2*g*l1*l2**2*m2*m3*np.cos(alfa1) + T2*l1**3*l2*m2**2*np.sin(alfa3)*alfa2**2
+ T2*l1**3*l2*m2*m3*np.sin(alfa3)*alfa2**2 -
2*T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa2**2 -
2*T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa2*alfa4 -
T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa4**2 +
T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 +
2*T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 +
T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 - g*l1**2*l2*m2*m3*np.cos(alfa1 -
alfa3)/2 + g*l1**2*l2*m2*m3*np.cos(alfa1 + alfa3)/2 -
g*l1**2*l2*m3**2*np.cos(alfa1 - alfa3)/2 + g*l1**2*l2*m3**2*np.cos(alfa1 +
alfa3)/2 + g*l1*l2**2*m2*m3*np.cos(alfa1) -    g*l1*l2**2*m3**2*np.cos(alfa1 +
2*alfa3)/2 + g*l1*l2**2*m3**2*np.cos(alfa1)/2 +
l1**3*l2*m2*m3*np.sin(alfa3)*alfa2**2 + l1**3*l2*m3**2*np.sin(alfa3)*alfa2**2 -
l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa2**2 -
l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa2*alfa4 -
l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa4**2/2 +
l1*l2**3*m3**2*np.sin(alfa3)*alfa2**2 +
2*l1*l2**3*m3**2*np.sin(alfa3)*alfa2*alfa4 +
l1*l2**3*m3**2*np.sin(alfa3)*alfa4**2)/(I1s1*I2s2 + I1s1*T2**2*l2**2*m2 +
I1s1*l2**2*m3 + I2s2*T1**2*l1**2*m1 + I2s2*T2**2*l2**2*m2 -
2*I2s2*T2*l1*l2*m2*np.cos(alfa3) + I2s2*l1**2*m2 + I2s2*l1**2*m3 -
2*I2s2*l1*l2*m3*np.cos(alfa3) + I2s2*l2**2*m3 +
T1**2*T2**2*l1**2*l2**2*m1*m2 + T1**2*l1**2*l2**2*m1*m3 +
T2**2*l1**2*l2**2*m2**2*np.sin(alfa3)**2 + T2**2*l1**2*l2**2*m2*m3 +
2*T2*l1**2*l2**2*m2*m3*np.sin(alfa3)**2 - 2*T2*l1**2*l2**2*m2*m3 +
l1**2*l2**2*m2*m3 + l1**2*l2**2*m3**2*np.sin(alfa3)**2)
```

2 PyDy multibody dynamics

```
from __future__ import print_function, division
from sympy import symbols, simplify
from sympy.physics.mechanics import dynamicsymbols, ReferenceFrame,
Point
from sympy.physics.vector import init_vprinting
from sympy.physics.mechanics import inertia, RigidBody
from sympy import trigsimp
from sympy.physics.mechanics import KanesMethod
from numpy import deg2rad, rad2deg, array, zeros, linspace
import numpy as np
from scipy.integrate import odeint
from pydy.codegen.ode_function_generators import generate_ode_function
from matplotlib import pyplot as plt
from pydy.viz.shapes import Cylinder, Sphere
import pydy.viz
from pydy.viz.visualization_frame import VisualizationFrame
from pydy.viz.scene import Scene
from IPython.display import Image
import math
from sympy.utilities.lambdify import lambdify, implemented_function
Image('figures/human_balance_diagram.png')
init_vprinting(pretty_print=True)

inertial_frame = ReferenceFrame('I')
arm_frame = ReferenceFrame('A')
forearm_frame = ReferenceFrame('F')
hand_frame = ReferenceFrame('H')

theta1, theta2, theta3 = dynamicsymbols('theta1, theta2, theta3')

arm_frame.orient(inertial_frame, 'Axis', (theta1, inertial_frame.z))
forearm_frame.orient(arm_frame, 'Axis', (theta2, arm_frame.z))
hand_frame.orient(forearm_frame, 'Axis', (theta3, forearm_frame.z))

simplify(hand_frame.dcm(inertial_frame))

shoulder = Point('S')

arm_length = symbols('l_A')
elbow = Point('E')
elbow.set_pos(shoulder, arm_length * arm_frame.y)

elbow.pos_from(shoulder)
elbow.pos_from(shoulder).express(inertial_frame).simplify()

forearm_length = symbols('l_F')
wrist = Point('W')
wrist.set_pos(elbow, forearm_length * forearm_frame.y)

wrist.pos_from(shoulder).express(inertial_frame).simplify()

arm_com_length = symbols('d_A')
arm_mass_center = Point('A_o')
arm_mass_center.set_pos(shoulder, arm_com_length * arm_frame.y)

forearm_com_length = symbols('d_F')
forearm_mass_center = Point('F_o')
forearm_mass_center.set_pos(elbow, forearm_com_length *
```

```
forearm_frame.y)

hand_com_length = symbols('d_H')
hand_mass_center = Point('H_o')
hand_mass_center.set_pos(wrist, hand_com_length * hand_frame.y)

omega1, omega2, omega3 = dynamicsymbols('omega1, omega2, omega3')

kinematical_differential_equations = [omega1 - theta1.diff(),
                                      omega2 - theta2.diff(),
                                      omega3 - theta3.diff()]

arm_frame.set_ang_vel(inertial_frame,omega1*inertial_frame.z)
forearm_frame.set_ang_vel(arm_frame,omega2*inertial_frame.z)
hand_frame.set_ang_vel(forearm_frame,omega3*inertial_frame.z)

shoulder.set_vel(inertial_frame, 0)

arm_mass_center.v2pt_theory(shoulder, inertial_frame, arm_frame)
arm_mass_center.vel(inertial_frame).express(inertial_frame).simplify()

elbow.v2pt_theory(shoulder, inertial_frame, arm_frame)

forearm_mass_center.v2pt_theory(elbow, inertial_frame, forearm_frame)
forearm_mass_center.vel(inertial_frame).express(inertial_frame).simpli
fy()

wrist.v2pt_theory(elbow, inertial_frame, forearm_frame)
hand_mass_center.v2pt_theory(wrist, inertial_frame, hand_frame)
hand_mass_center.vel(inertial_frame).express(inertial_frame).simplify(
)

arm_mass, forearm_mass, hand_mass = symbols('m_A, m_F, m_H')

arm_inertia, forearm_inertia, hand_inertia = symbols('I_Az, I_Fz,
I_Hz')

arm_inertia_dyadic = inertia(arm_frame, 0, 0, arm_inertia)

arm_inertia_dyadic.to_matrix(arm_frame)

arm_central_inertia = (arm_inertia_dyadic, arm_mass_center)

forearm_inertia_dyadic = inertia(forearm_frame, 0, 0, forearm_inertia)
forearm_central_inertia = (forearm_inertia_dyadic,
forearm_mass_center)

hand_inertia_dyadic = inertia(hand_frame, 0, 0, hand_inertia)
hand_central_inertia = (hand_inertia_dyadic, hand_mass_center)

arm = RigidBody('Arm', arm_mass_center, arm_frame,
                arm_mass, arm_central_inertia)
forearm = RigidBody('Forearm', forearm_mass_center, forearm_frame,
                forearm_mass, forearm_central_inertia)
hand = RigidBody('Hand', hand_mass_center, hand_frame,
                hand_mass, hand_central_inertia)

g = symbols('g')

arm_grav_force_vector = arm_mass * g * (-inertial_frame.y)
arm_grav_force = (arm_mass_center, arm_grav_force_vector)
```

```
forearm_grav_force_vector = forearm_mass * g * (-inertial_frame.y)
forearm_grav_force = (forearm_mass_center, forearm_grav_force_vector)

hand_grav_force_vector = hand_mass * g * (-inertial_frame.y)
hand_grav_force = (hand_mass_center, hand_grav_force_vector)

shoulder_torque, elbow_torque, wrist_torque = dynamicsymbols('T_s,
T_e, T_w')

arm_torque_vector = shoulder_torque * inertial_frame.z - elbow_torque
* inertial_frame.z
arm_torque = (arm_frame, arm_torque_vector)

forearm_torque_vector = elbow_torque * inertial_frame.z - wrist_torque
* inertial_frame.z
forearm_torque = (forearm_frame, forearm_torque_vector)

hand_torque_vector = wrist_torque * inertial_frame.z
hand_torque = (hand_frame, hand_torque_vector)

coordinates = [theta1, theta2, theta3]
speeds = [omega1, omega2, omega3]

kane = KanesMethod(inertial_frame, coordinates, speeds,
kinematical_differential_equations)

loads = [arm_grav_force,
         forearm_grav_force,
         hand_grav_force,
         arm_torque,
         forearm_torque,
         hand_torque]

bodies = [arm, forearm, hand]

fr, frstar = kane.kanes_equations(bodies, loads)

trigsimp(fr + frstar)

mass_matrix = trigsimp(kane.mass_matrix_full)
forcing_vector = trigsimp(kane.forcing_full)

constants = [arm_length,
             arm_com_length,
             arm_mass,
             arm_inertia,
             forearm_length,
             forearm_com_length,
             forearm_mass,
             forearm_inertia,
             hand_com_length,
             hand_mass,
             hand_inertia,
             g]

specified = [shoulder_torque, elbow_torque, wrist_torque]

right_hand_side = generate_ode_function(forcing_vector, coordinates,
                                        speeds, constants,
                                        mass_matrix=mass_matrix,
```

```
                                        specifieds=specified)

wrist.v2pt_theory(elbow, inertial_frame,
forearm_frame).express(inertial_frame)

x0 = zeros(6)
#x0[:3] = deg2rad(2.0)
x0[0] = deg2rad(-45)
x0[1] = deg2rad(-180)

m = 80
print('hmotnost člověka = ', m)
h = 180
print('výška člověka = ', h)
m1 = 0.0312*m-0.0027*h+0.25
print("hmotnost paže = ", m1)
m2 = 0.01445*m-0.00114*h+0.3185
print('hmotnost předloktí = ', m2)
m3 = 0.15+0.0036*m+0.00175*h-0.1165
print('hmotnost ruky + míčku = ', m3)
l3 = 0.1

frames_per_sec = 1000
final_time = 0.2
L_max = 10
L_min = 1
L_step =0.25
L_size = (L_max-L_min)/L_step+1
A = np.zeros((10, 10, int(final_time * frames_per_sec)))
g = 9.81

for l1f in range(L_min, L_max +1, 1):
    #[7]
    l1 = l1f/20

    for l2f in range(L_min, L_max + 1, 1):
        #[9]
        l2 = l2f/20
        numerical_constants = array([l1,  # arm_length [m]
                                     0.458*l1,  # arm_com_length [m]
                                     m1,  # arm_mass [kg]
                                     1 / 12 * m1 * l1 ** 2,  #
arm_inertia [kg*m^2]
                                     l2,  # forearm_length [m]
                                     0.434*l2,  # forearm_com_length
                                     m2,  # forearm_mass [kg]
                                     1 / 12 * m2 * l2 ** 2,  #
forearm_inertia [kg*m^2]
                                     l3,  # hand_com_length [m]
                                     m3,  # hand_mass [kg]
                                     1/12*m3*l3**2,  # hand_inertia
[kg*m^2]
                                     9.81],  # acceleration due to
gravity [m/s^2]
                                    )

        t = linspace(0.0, final_time, int(final_time *
frames_per_sec))

        numerical_specified = [np.ones(t.size), 10 * np.ones(t.size),
0 * np.ones(t.size)]
```

```
        args = {'constants': numerical_constants,
                'specified': numerical_specified}

        def torquesInTime(x, t):
            if x[0] < np.pi / 2:
                return [130, 30, 2]
            else:
                return [0, 0, 0]

        y = odeint(right_hand_side, x0, t, args=(torquesInTime,
numerical_constants))

        for z in range(0, int(final_time * frames_per_sec)):
            th1 = y[z][0]
            th2 = y[z][1]
            th3 = y[z][2]
            th1dot = y[z][3]
            th2dot = y[z][4]
            th3dot = y[z][5]

            if th1 > math.pi/2:
                D_1 = 0
            else:
                D_1 = 1

            if th2 > math.pi:
                D_2 = 0
            else:
                D_2 = 1

            y3dot = -l1 * th1dot * np.sin(th1) - l2 * (np.sin(th1) *
np.cos(th2) + np.sin(th2) * np.cos(th1)) * (th1dot + th2dot)
            x3dot = -l1 * th1dot * np.cos(th1) - l2 * (-np.sin(th1) *
np.sin(th2) + np.cos(th1) * np.cos(th2)) * (th1dot + th2dot)

            if y3dot < 0 or x3dot > 0:
                D_3 = 0
            else:
                D_3 = 1

            v_02 = (x3dot ** 2 + y3dot ** 2) ** (0.5)
            fi_h_1 = math.atan(abs(y3dot) / abs(x3dot))

            if fi_h_1 > 1e-10:
                fi_h = math.atan(abs(y3dot) / abs(x3dot))
            else:
                fi_h = 0

            D = D_1 * D_2 * D_3 * v_02 ** 2 * np.sin(2 * fi_h)/g
            A[l1f-1, l2f-1, z] = D
            #print(l1f-1, l2f-1, 't =', z, 'ms ', ' l1 =', l1, 'm ', '
l2 =', l2, 'm ', ' D =', D, 'm')
            #th1, th2, th3, y3dot, x3dot, fi_h, v_02
print(A)
ind = np.unravel_index(np.argmax(A, axis=None), A.shape)
print(A[ind])
print(ind)

plt.plot(t, rad2deg(y[:, :3]))
plt.xlabel('Time [s]')
```

```
plt.ylabel('Angle [deg]')
plt.legend(["${}$".format(c) for c in coordinates])
plt.show()

shoulder_shape = Sphere(color='black', radius=0.1)
elbow_shape = Sphere(color='black', radius=0.1)
wrist_shape = Sphere(color='black', radius=0.1)

shoulder_viz_frame = VisualizationFrame(inertial_frame, shoulder,
shoulder_shape)
elbow_viz_frame = VisualizationFrame(inertial_frame, elbow,
elbow_shape)
wrist_viz_frame = VisualizationFrame(inertial_frame, wrist,
wrist_shape)

arm_center = Point('a_c')
forearm_center = Point('f_c')
hand_center = Point('h_c')

arm_center.set_pos(shoulder, arm_length / 2 * arm_frame.y)
forearm_center.set_pos(elbow, forearm_length / 2 * forearm_frame.y)
hand_center.set_pos(wrist, hand_com_length * hand_frame.y)

constants_dict = dict(zip(constants, numerical_constants))

arm_shape = Cylinder(radius=0.08, length=constants_dict[arm_length],
color='blue')
arm_viz_frame = VisualizationFrame('Arm', arm_frame, arm_center,
arm_shape)

forearm_shape = Cylinder(radius=0.08,
length=constants_dict[forearm_length], color='green')
forearm_viz_frame = VisualizationFrame('Forearm', forearm_frame,
forearm_center, forearm_shape)
hand_shape = Cylinder(radius=0.08, length=2 *
constants_dict[hand_com_length], color='red')
hand_viz_frame = VisualizationFrame('Hand', hand_frame, hand_center,
hand_shape)

scene = Scene(inertial_frame, shoulder)

scene.visualization_frames = [shoulder_viz_frame,
                              elbow_viz_frame,
                              wrist_viz_frame,
                              arm_viz_frame,
                              forearm_viz_frame,
                              hand_viz_frame]

scene.states_symbols = coordinates + speeds
scene.constants = constants_dict
scene.states_trajectories = y

scene.display()
```

3 SymPy odvození Lagrangeových rovnic 2. druhu

```
from sympy import *
l1, l2, l3, T1, T2, m1, m2, m3, I1s1, I2s2, g, M1, M2, v_02, fi_h, t =
symbols('l1 l2 l3 T1 T2 m1 m2 m3 I1s1 I2s2 g M1 M2 v_02 fi_h t')
fi1 = Function('fi1')(t)
fi3 = Function('fi3')(t)
eqn = fi1
fi1dot=eqn.diff(t)
eqn = fi3
fi3dot=eqn.diff(t)
eqn = fi1
fi1dot2=eqn.diff(t, t)
eqn = fi3
fi3dot2=eqn.diff(t, t)
eqn = -l1*T1*cos(fi1)
x1dot=eqn.diff(t)
eqn = l1*T1*sin(fi1)
y1dot=eqn.diff(t)
#print('y1dot =', y1dot)
eqn = -l1*cos(fi1)+l2*T2*cos(fi1+fi3)
x2dot=eqn.diff(t)
eqn = l1*sin(fi1)-l2*T2*sin(fi1+fi3)
y2dot=eqn.diff(t)
#print('y2dot =', y2dot)
eqn = -l1*cos(fi1)+l2*cos(fi1+fi3)
x3dot=eqn.diff(t)
eqn = l1*sin(fi1)-l2*sin(fi1+fi3)
y3dot=eqn.diff(t)
#print('y3dot =', y3dot)

#print('x3dot = ', x3dot)
#print('y3dot = ', y3dot)

x1doty1dot = x1dot**2+y1dot**2
x2doty2dot = x2dot**2+y2dot**2
x3doty3dot = x3dot**2+y3dot**2

v_02 = x3doty3dot**(0.5)
#print('v_02 =', v_02)
fi_h = 1/tan(y3dot/x3dot)
#print('fi_h =', fi_h)
D=v_02*sin(2*(pi/2-fi_h))/g
#print('D =', D)

Ek =
(x1doty1dot*m1)/2+(I1s1*fi1dot**2)/2+(m2*x2doty2dot)/2+(I2s2*fi3dot**2
)/2+(m3*x3doty3dot)/2
eqn = Ek
Ekfi1dot=eqn.diff(fi1dot)
eqn = Ek
Ekfi3dot=eqn.diff(fi3dot)
eqn = Ekfi1dot
Ekfi1dotdt=eqn.diff(t)
eqn = Ekfi3dot
Ekfi3dotdt=eqn.diff(t)
eqn = Ek
```

```
Ekfi1=eqn.diff(fi1)
eqn = Ek
Ekfi3=eqn.diff(fi3)
Q1L=Ekfi1dotdt-Ekfi1
Q3L=Ekfi3dotdt-Ekfi3
Q1P=M1-M2-m1*g*l1*T1*cos(fi1)-m2*g*(l1*cos(fi1)-T2*l2*cos(fi1+fi3))-
m3*g*(l1*cos(fi1)-l2*cos(fi1+fi3))
Q3P=M2+m2*g*l2*T2*cos(fi1+fi3)+m3*g*l2*cos(fi1+fi3)
rovnice1=Q1L-Q1P
rovnice2=Q3L-Q3P
fi1dot22=solve(rovnice1, fi1dot2)
fi3dot22=solve(rovnice2, fi3dot2)
#print('fi1dot22 =', fi1dot22)
#print('fi3dot22 =', fi3dot22)
fi1dot22 = (M1 - M2 - T1*g*l1*m1*cos(fi1) - T2**2*l2**2*m2*fi3dot2 +
T2*g*l2*m2*cos(fi1 + fi3) - 2*T2*l1*l2*m2*sin(fi3)*fi1dot*fi3dot -
T2*l1*l2*m2*sin(fi3)*fi3dot**2 + T2*l1*l2*m2*cos(fi3)*fi3dot2 -
g*l1*m2*cos(fi1) - g*l1*m3*cos(fi1) + g*l2*m3*cos(fi1 + fi3) -
2*l1*l2*m3*sin(fi3)*fi1dot*fi3dot - l1*l2*m3*sin(fi3)*fi3dot**2 +
l1*l2*m3*cos(fi3)*fi3dot2 - l2**2*m3*fi3dot2)/(I1s1 + T1**2*l1**2*m1 +
T2**2*l2**2*m2 - 2*T2*l1*l2*m2*cos(fi3) + l1**2*m2 + l1**2*m3 -
2*l1*l2*m3*cos(fi3) + l2**2*m3)
fi1dot22 = fi1dot22.subs(fi3dot2, (M2 - T2**2*l2**2*m2*fi1dot2 +
T2*g*l2*m2*cos(fi1 + fi3) + T2*l1*l2*m2*sin(fi3)*fi1dot**2 +
T2*l1*l2*m2*cos(fi3)*fi1dot2 + g*l2*m3*cos(fi1 + fi3) +
l1*l2*m3*sin(fi3)*fi1dot**2 + l1*l2*m3*cos(fi3)*fi1dot2 -
l2**2*m3*fi1dot2)/(I2s2 + T2**2*l2**2*m2 + l2**2*m3))
fi1dot222 = fi1dot22-fi1dot2
fi1dot2T = solve(fi1dot222, fi1dot2)
print('fi1dot2T =', fi1dot2T)
fi3dot22 = (M2 - T2**2*l2**2*m2*fi1dot2 + T2*g*l2*m2*cos(fi1 + fi3) +
T2*l1*l2*m2*sin(fi3)*fi1dot**2 + T2*l1*l2*m2*cos(fi3)*fi1dot2 +
g*l2*m3*cos(fi1 + fi3) + l1*l2*m3*sin(fi3)*fi1dot**2 +
l1*l2*m3*cos(fi3)*fi1dot2 - l2**2*m3*fi1dot2)/(I2s2 + T2**2*l2**2*m2 +
l2**2*m3)
fi3dot22 = fi3dot22.subs(fi1dot2, (M1 - M2 - T1*g*l1*m1*cos(fi1) -
T2**2*l2**2*m2*fi3dot2 + T2*g*l2*m2*cos(fi1 + fi3) -
2*T2*l1*l2*m2*sin(fi3)*fi1dot*fi3dot - T2*l1*l2*m2*sin(fi3)*fi3dot**2
+ T2*l1*l2*m2*cos(fi3)*fi3dot2 - g*l1*m2*cos(fi1) - g*l1*m3*cos(fi1) +
g*l2*m3*cos(fi1 + fi3) - 2*l1*l2*m3*sin(fi3)*fi1dot*fi3dot -
l1*l2*m3*sin(fi3)*fi3dot**2 + l1*l2*m3*cos(fi3)*fi3dot2 -
l2**2*m3*fi3dot2)/(I1s1 + T1**2*l1**2*m1 + T2**2*l2**2*m2 -
2*T2*l1*l2*m2*cos(fi3) + l1**2*m2 + l1**2*m3 - 2*l1*l2*m3*cos(fi3) +
l2**2*m3))
fi3dot222 = fi3dot22-fi3dot2
fi3dot2T = solve(fi3dot222, fi3dot2)
print('fi3dot2T =', fi3dot2T)
```

4 SymPy integrace diferenciálních rovnic

```
import math
from scipy import integrate
import numpy as np
import matplotlib.pyplot as plt

m = 80
print('hmotnost člověka = ', m)
h = 180
print('výška člověka = ', h)
m1 = 0.0312*m-0.0027*h+0.25
#0.0312*m-0.0027*h+0.25
print("hmotnost paže = ", m1)
m2 = 0.01445*m-0.00114*h+0.3185
#0.01445*m-0.00114*h+0.3185
print('hmotnost předloktí = ', m2)
m3 = 0.15+0.0036*m+0.00175*h-0.1165
#0.15+0.0036*m+0.00175*h-0.1165
print('hmotnost ruky + míčku = ', m3)
l1 = 0.35
l2 = 0.3
T1 = 0.458
T2 = 0.434
g = 9.81
I1s1 = m1*l1*l1/12
print(I1s1)
I2s2 = m2*l2*l2/12
print(I2s2)
pol1 = I1s1 + m1*l1*T1*T1 + m2*l1*l1 + m3*l1*l1
pol2 = I2s2 + m2*l2*l2*T2*T2 + m3*l2*l2
pol3 = l1*l2*(m2*T2+m3)
pol4 = g*l2*(m2*T2 + m3)
pol5 = (m1*T1+m2+m3)*g*l1

t_start = 0
t_stop = 0.3
t_step = 301
M1NmL = 1e-1
M1NmH = 201
M2NmL = 200
M2NmH = 251
M_step = 1e-1

A = np.zeros((51, 31, int(t_stop * 1000)))

for M1 in range(100, 151, 1):
    #[0.2/1000, 0.3/1000]
    for M2 in range(20, 51, 1):
        #[0.2/1000, 0.3/1000]
        def f(t, r):
            alfa1, alfa2, alfa3, alfa4 = r
            alfa1_dot = alfa2
            alfa2_dot = (I2s2*M1 - I2s2*M2 -
I2s2*T1*g*l1*m1*np.cos(alfa1) + I2s2*T2*g*l2*m2*np.cos(alfa1 + alfa3)
- 2*I2s2*T2*l1*l2*m2*np.sin(alfa3)*alfa2*alfa4 -
I2s2*T2*l1*l2*m2*np.sin(alfa3)*alfa4**2 - I2s2*g*l1*m2*np.cos(alfa1) -
I2s2*g*l1*m3*np.cos(alfa1) + I2s2*g*l2*m3*np.cos(alfa1 + alfa3) -
2*I2s2*l1*l2*m3*np.sin(alfa3)*alfa2*alfa4 -
I2s2*l1*l2*m3*np.sin(alfa3)*alfa4**2 + M1*T2**2*l2**2*m2 + M1*l2**2*m3
```

```
- 2*M2*T2**2*l2**2*m2 + M2*T2*l1*l2*m2*np.cos(alfa3) +
M2*l1*l2*m3*np.cos(alfa3) - 2*M2*l2**2*m3 -
T1*T2**2*g*l1*l2**2*m1*m2*np.cos(alfa1) -
T1*g*l1*l2**2*m1*m3*np.cos(alfa1) -
T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2**2 -
2*T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2*alfa4 -
T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa4**2 +
T2**2*g*l1*l2**2*m2**2*np.cos(alfa1 + 2*alfa3)/2 -
T2**2*g*l1*l2**2*m2**2*np.cos(alfa1)/2 -
T2**2*g*l1*l2**2*m2*m3*np.cos(alfa1) +
T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa2**2/2 -
T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 -
2*T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 -
T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 +
T2*g*l1*l2**2*m2*m3*np.cos(alfa1 + 2*alfa3) +
T2*g*l1*l2**2*m2*m3*np.cos(alfa1) +
T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa2**2 -
T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 -
2*T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 -
T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 -
g*l1*l2**2*m2*m3*np.cos(alfa1) + g*l1*l2**2*m3**2*np.cos(alfa1 +
2*alfa3)/2 - g*l1*l2**2*m3**2*np.cos(alfa1)/2 +
l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa2**2/2 -
l1*l2**3*m3**2*np.sin(alfa3)*alfa2**2 -
2*l1*l2**3*m3**2*np.sin(alfa3)*alfa2*alfa4 -
l1*l2**3*m3**2*np.sin(alfa3)*alfa4**2)/(I1s1*I2s2 +
I1s1*T2**2*l2**2*m2 + I1s1*l2**2*m3 + I2s2*T1**2*l1**2*m1 +
I2s2*T2**2*l2**2*m2 - 2*I2s2*T2*l1*l2*m2*np.cos(alfa3) + I2s2*l1**2*m2
+ I2s2*l1**2*m3 - 2*I2s2*l1*l2*m3*np.cos(alfa3) + I2s2*l2**2*m3 +
T1**2*T2**2*l1**2*l2**2*m1*m2 + T1**2*l1**2*l2**2*m1*m3 +
T2**2*l1**2*l2**2*m2**2*np.sin(alfa3)**2 + T2**2*l1**2*l2**2*m2*m3 +
2*T2*l1**2*l2**2*m2*m3*np.sin(alfa3)**2 - 2*T2*l1**2*l2**2*m2*m3 +
l1**2*l2**2*m2*m3 + l1**2*l2**2*m3**2*np.sin(alfa3)**2)
            alfa3_dot = alfa4
            alfa4_dot = (I1s1*M2 + I1s1*T2*g*l2*m2*np.cos(alfa1 +
alfa3) + I1s1*T2*l1*l2*m2*np.sin(alfa3)*alfa2**2 +
I1s1*g*l2*m3*np.cos(alfa1 + alfa3) +
I1s1*l1*l2*m3*np.sin(alfa3)*alfa2**2 - M1*T2**2*l2**2*m2 +
M1*T2*l1*l2*m2*np.cos(alfa3) + M1*l1*l2*m3*np.cos(alfa3) - M1*l2**2*m3
+ M2*T1**2*l1**2*m1 + 2*M2*T2**2*l2**2*m2 -
3*M2*T2*l1*l2*m2*np.cos(alfa3) + M2*l1**2*m2 + M2*l1**2*m3 -
3*M2*l1*l2*m3*np.cos(alfa3) + 2*M2*l2**2*m3 +
T1**2*T2*g*l1**2*l2*m1*m2*np.cos(alfa1 + alfa3) +
T1**2*T2*l1**3*l2*m1*m2*np.sin(alfa3)*alfa2**2 +
T1**2*g*l1**2*l2*m1*m3*np.cos(alfa1 + alfa3) +
T1**2*l1**3*l2*m1*m3*np.sin(alfa3)*alfa2**2 +
T1*T2**2*g*l1*l2**2*m1*m2*np.cos(alfa1) -
T1*T2*g*l1**2*l2*m1*m2*np.cos(alfa1 - alfa3)/2 -
T1*T2*g*l1**2*l2*m1*m2*np.cos(alfa1 + alfa3)/2 -
T1*g*l1**2*l2*m1*m3*np.cos(alfa1 - alfa3)/2 -
T1*g*l1**2*l2*m1*m3*np.cos(alfa1 + alfa3)/2 +
T1*g*l1*l2**2*m1*m3*np.cos(alfa1) +
T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2**2 +
2*T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa2*alfa4 +
T2**3*l1*l2**3*m2**2*np.sin(alfa3)*alfa4**2 -
T2**2*g*l1*l2**2*m2**2*np.cos(alfa1 + 2*alfa3)/2 +
T2**2*g*l1*l2**2*m2**2*np.cos(alfa1)/2 +
T2**2*g*l1*l2**2*m2*m3*np.cos(alfa1) -
T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa2**2 -
T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa2*alfa4 -
T2**2*l1**2*l2**2*m2**2*np.sin(2*alfa3)*alfa4**2/2 +
```

```
T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 +
2*T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 +
T2**2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 -
T2*g*l1**2*l2*m2**2*np.cos(alfa1 - alfa3)/2 +
T2*g*l1**2*l2*m2**2*np.cos(alfa1 + alfa3)/2 -
T2*g*l1**2*l2*m2*m3*np.cos(alfa1 - alfa3)/2 +
T2*g*l1**2*l2*m2*m3*np.cos(alfa1 + alfa3)/2 -
T2*g*l1*l2**2*m2*m3*np.cos(alfa1 + 2*alfa3) -
T2*g*l1*l2**2*m2*m3*np.cos(alfa1) +
T2*l1**3*l2*m2**2*np.sin(alfa3)*alfa2**2 +
T2*l1**3*l2*m2*m3*np.sin(alfa3)*alfa2**2 -
2*T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa2**2 -
2*T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa2*alfa4 -
T2*l1**2*l2**2*m2*m3*np.sin(2*alfa3)*alfa4**2 +
T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2**2 +
2*T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa2*alfa4 +
T2*l1*l2**3*m2*m3*np.sin(alfa3)*alfa4**2 -
g*l1**2*l2*m2*m3*np.cos(alfa1 - alfa3)/2 +
g*l1**2*l2*m2*m3*np.cos(alfa1 + alfa3)/2 -
g*l1**2*l2*m3**2*np.cos(alfa1 - alfa3)/2 +
g*l1**2*l2*m3**2*np.cos(alfa1 + alfa3)/2 +
g*l1*l2**2*m2*m3*np.cos(alfa1) - g*l1*l2**2*m3**2*np.cos(alfa1 +
2*alfa3)/2 + g*l1*l2**2*m3**2*np.cos(alfa1)/2 +
l1**3*l2*m2*m3*np.sin(alfa3)*alfa2**2 +
l1**3*l2*m3**2*np.sin(alfa3)*alfa2**2 -
l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa2**2 -
l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa2*alfa4 -
l1**2*l2**2*m3**2*np.sin(2*alfa3)*alfa4**2/2 +
l1*l2**3*m3**2*np.sin(alfa3)*alfa2**2 +
2*l1*l2**3*m3**2*np.sin(alfa3)*alfa2*alfa4 +
l1*l2**3*m3**2*np.sin(alfa3)*alfa4**2)/(I1s1*I2s2 +
I1s1*T2**2*l2**2*m2 + I1s1*l2**2*m3 + I2s2*T1**2*l1**2*m1 +
I2s2*T2**2*l2**2*m2 - 2*I2s2*T2*l1*l2*m2*np.cos(alfa3) + I2s2*l1**2*m2
+ I2s2*l1**2*m3 - 2*I2s2*l1*l2*m3*np.cos(alfa3) + I2s2*l2**2*m3 +
T1**2*T2**2*l1**2*l2**2*m1*m2 + T1**2*l1**2*l2**2*m1*m3 +
T2**2*l1**2*l2**2*m2**2*np.sin(alfa3)**2 + T2**2*l1**2*l2**2*m2*m3 +
2*T2*l1**2*l2**2*m2*m3*np.sin(alfa3)**2 - 2*T2*l1**2*l2**2*m2*m3 +
l1**2*l2**2*m2*m3 + l1**2*l2**2*m3**2*np.sin(alfa3)**2)
            return [alfa1_dot, alfa2_dot, alfa3_dot, alfa4_dot]
        sol = integrate.solve_ivp(f, (t_start, t_stop), (math.pi/4, 0,
0, 0), t_eval=np.linspace(t_start, t_stop, t_step))
        A =sol.y
        for x in range(0, t_step):
            [alfa1, alfa2, alfa3, alfa4] = A[:, x]
            x3dot = (l1 * np.sin(alfa1) * alfa2 - l2 * (alfa2 + alfa4)
* np.sin(alfa1 + alfa3))
            y3dot = (l1 * np.cos(alfa1) * alfa2 - l2 * (alfa2 + alfa4)
* np.cos(alfa1 + alfa3))
            if x3dot<0:
                v_02 = -((x3dot**2+y3dot**2)**(0.5))
            else:
                v_02 = ((x3dot**2+y3dot**2)**(0.5))
            if alfa1>math.pi:
                D_1 = 0
            else:
                D_1 = 1
            fi_h = math.degrees(math.atan(abs(y3dot) / abs(x3dot)))
            D = D_1 * v_02**2 * np.sin(math.radians(2 * fi_h)) / g

            A[M1-100, M2-20, x] = D
```

```
            #print('alfa1 =', math.degrees(alfa1) , 'alfa2 =',
math.degrees(alfa2), 'alfa3 =', math.degrees(alfa3), 'alfa4 =',
math.degrees(alfa4))
            #print('M1 =', M1, 'M2 =', M2, 't =', x, 'x3dot =',
x3dot)
            #print('M1 =', M1, 'M2 =', M2, 't =', x, 'y3dot =', y3dot)
            #print('M1 =', M1, 'M2 =', M2, 't =', x, 'v_02 =', v_02)
            #print('M1 =', M1, 'M2 =', M2, 't =', x, 'fi_h =', fi_h)
            #print('M1 =', M1, 'M2 =', M2, 't =', x, 'D =', D)
print(A)
ind = np.unravel_index(np.argmax(A, axis=None), A.shape)
print(A[ind]*2)
print(ind)
```