



Budapesti Műszaki és Gazdaságtudományi Egyetem



Műszaki Mechanika Tanszék

Féregszerű mozgások vizsgálata

Szerző:

Juhász Zoltán

Mechatronikai Mérnök Hallgató, IV. évf.

Budapesti Műszaki és Gazdaságtudományi Egyetem

Konzulens:

Zelei Ambrus

Tudományos segédmunkatárs

MTA-BME Gépek és Járművek Dinamikája Kutatócsoport

Tudományos Diákköri Dolgozat

2011. november

Féregszerű mozgások vizsgálata

Tudományos Diákköri dolgozat

szerző: Juhász Zoltán, mechatronikai mérnök hallgató
konzulens: Zelei Ambrus, tudományos segédmunkatárs

Bevezető

A természet mindig érdekes ötletek tárháza, melyek az évezredek evolúciós fejlődésnek hála gyakran olyan kiforrottak, hogy a mérnököknek nincs szükségük új ötletekre elég "csupán" a természetet utánózni. Számtalan ilyen műszaki megoldást lehet találni a műszaki tudományok szinte bármely területén.

A kigyók és férgek már az állatvilág fejlődésének kezdetin szakaszán jelen voltak és mind a mai napig a föld szinte bármely területén megtalálhatóak, melyben jelentős szerepet játszik mozgásfajtájuk. Mind féreg mind a kígyószerű mozgások jellegzetesége, hogy teljes testük periodikus mozgatásával, összehúzódásokkal-megnyúlásokkal (perisztaltia), vagy tekergőzéssel a talaj vagy a közeg súrlódását kihasználva jutnak előre. Mivel helyváltoztatásuk szinte egyetlen feltétele az őket körülvevő közeggel kialakított kölcsönhatás, ezért vízben, szárazon és a föld alatt is egyaránt otthonosan mozognak, ruganyos hajlékony alakjuk pedig lehetővé teszi, hogy könnyűszerrel küzdenek le bármilyen nehéz terepet.

Ezen munka célja az állatvilágban évmilliók óta jelenlévő csúszó-mászó lények mozgásának vizsgálata numerikus szimulációk útján. A modellalkotás célja nem csupán a természet, ezen mozgásfajtájának megismerés, és előnyeinek ezúton való feltérképezése, hanem esetlegesen alapot szolgáltathat újfajta mozgástípussal rendelkező robotoknak, melyeket hatékonyan lehetne olyan terepeken használni ahol a hagyományos kerekcsatlós mozgás nem valósítható meg, a lábakon történő mozgás pedig helyhiány (pl.: földrengés után épület törmelékek között) miatt nem kivitelezhető, vagy a terep egyenetlensége miatt túl bonyolult szabályozást igényelne, hiszen a lábbakkal szerelt konstrukciók jelentős része folyamatosan instabil helyzetben van, szemben egy többnyire stabil helyzetben lévő hengeres testtel.

Analysing of worm-like locomotion

author: Juhász Zoltán, mechatronikai mérnök hallgató

supervisor: Zelei Ambrus, Research assistant, BME, Department of Applied Mechanics

Prologue

The nature is always a storehouse of great ideas, which are mostly so well planned because of the million years long evolution, that engineers do not need to get new ideas. It's enough to imitate the nature. There are uncountable nature-inspired mechanical solutions in almost every field of the mechanical science.

Snakes and worms were present soon in the early period of life, and they can still be found in almost every region of our planet. In this success their locomotion technique takes a big part. Snake- and worm-like movements have the property, that they move their whole body periodically, getting shorter than longer (peristaltic), or wriggled, to get forward used the friction of the ground or the surrounding medium. They only need getting forward is the friction, so they can move in fluids, on the ground or even below the ground, and their flexible body make them proper to accept heavy terrain too.

The aim of this work the analysing this locomotion technique through numerical simulations. After the modelling it could be a great substructure of a new robot which uses this locomotion technique. This could be used in such situations where the conventional propulsions (wheels, tracks) are not proper, and the movement with legs are too complicated cause of narrow places (in ruins) or the controlling would be extremely complicated, because of the terrain and we should control an every time unstable robot opposite to a stable cylinder like body which lays on the ground.

Tartalomjegyzék

1. Bevezető	1
1.1. Mozgásfajták	2
2. Modellalkotás	4
2.1. Helyváltoztatás értelmezése	4
2.1.1. Testek mozgásának definiálása	4
2.1.2. Maguktól mozgó rendszerek, helyváltoztatás	6
2.2. Talajjal való kapcsolat modellezése	8
2.2.1. Száraz súrlódás	8
2.2.2. Talajjal történő ütközés	13
2.3. A féregmodell	20
2.3.1. A féregtest összeállítása	20
2.3.2. Vezérlések megvalósítása	36
3. Optimalizálás	40
3.1. Genetikus algoritmus	41
3.1.1. A populáció létrehozása	42
3.1.2. Az egyedek osztályozása	43
3.1.3. Gének keresztezése és mutációja	44
3.1.4. A genetikus algoritmus alkalmazhatósága	46
3.2. A pásztázás módszere	46
3.3. Feltételes optimum keresés	49
4. Konklúzió, fejlesztési irányok	52
5. Köszönetnyilvánítás	54
A. A féregmodell programkódja	55
Irodalomjegyzék	64

1. fejezet

Bevezető

Ezen dolgozat témája végtagokkal nem rendelkező férgek, hernyók és kígyók mozgásának vizsgálata numerikus szimulációk útján. Ennek a háromféle élőlénynek (1.1. ábra) a mozgása lényegileg eltér egymástól, ezt a következő fejezetben tárgyaljuk. A dolgoztban tárgyalt témát az inspirálta, hogy ha szétnézünk világunkban, szinte mindenhol találhatók olyan lények, melyek hengeres testük alakjának periodikus megváltoztatásával az őket körülvevő közeg száraz súrlódása révén jutnak előre. Ezen lények evolúciós sikere nem véletlen, hiszen hajlékony alakjuknak köszönhetően szinte bármilyen terepviszonyból fakadó akadályt könnyűszerrel leküzdnek, helyváltoztató mozgás mechanizmusuk pedig lehetővé teszi számukra, hogy egyaránt otthonosan mozogjanak szárazon, vízben és a föld alatt is. A helyváltoztatás hatékonyságán kívül a modellezés mellett szól az is, hogy ilyen élőlények testfelépítése nagyon egyszerű, emiatt könnyen utánozható mérnöki szerkezetekben.

Ezen adottságaikból fakadóan, nem csoda, hogy olyan neves kutatóintézetek is kísérleteznek ilyen lényeket utánzó robotok létrehozásával, mint például a NASA, az MIT, az ETH, a Cleveland-i vagy az Ilmenau-i egyetem [1] [2] [3] [4].

A kutatások sokféle modellel próbálkoznak, emiatt jelenleg nincs egyértelmű irányvonal ezen kutatási területen. Munkám célja egy olyan univerzális modell megalkotása, mely többféle mozgásfajtát képes minél hatékonyabban leutánozni, így optimalizálási eljárások útján megkereshetjük a leghatékonyabb helyváltoztató módot az ilyen hagyományos kerekeket, láncalpakat, lábakat nélkülöző konstrukcióknak.

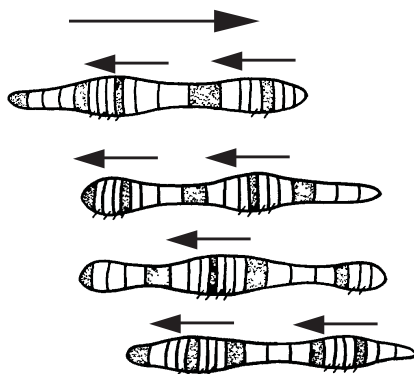


1.1. ábra: féreg, hernyó, kígyó

1.1. Mozgásfajták

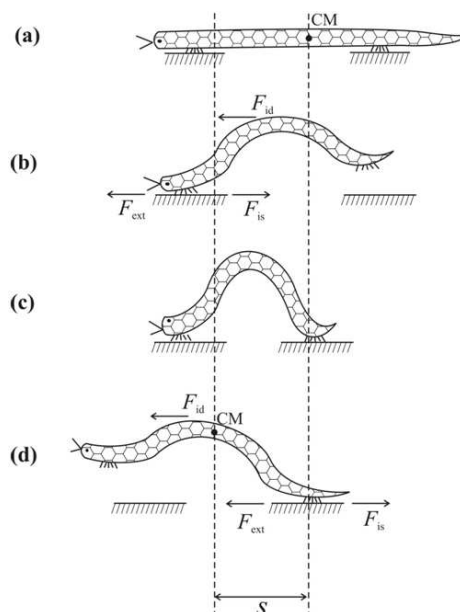
Bármennyire hasonlónak is tűnhet a lábatlan élőlények mozgása mégis számos variáció és ezek megszámlálhatatlan alvariációja figyelhető meg a természetben. Megfigyelések alapján 3 csoportra osztottam a vizsgálni kívánt mozgásokat:

1. **Féregmozgás:** Ezen mozgásfajtát a lineáris hossz és az átmérő periódikus megváltozása (*peristaltia*) jellemzi. A mozgásfajta nagy előnye, hogy egydimenziós, így kiválóan alkalmas szűk helyeken (föld alatt, törmelékek között) történő helyváltoztatáshoz.



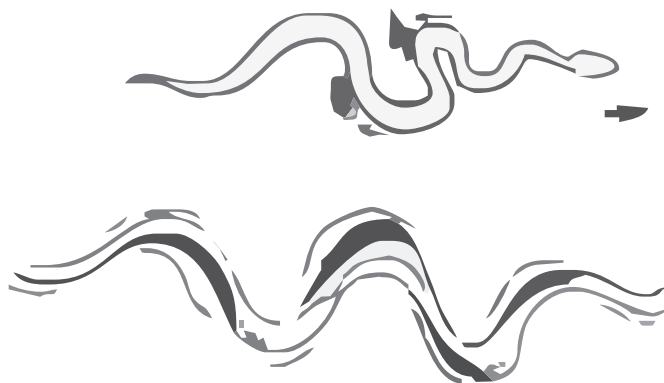
1.2. ábra: Peristaltia

2. **Hernyó mozgás:** Ennek a főként hernyóknál megfigyelhető helyváltoztatási módnak az alapja a test hullámformába történő meggömbölyítése, aminek révén a test végpontja elmozdul, majd ezután a hullám kiegyenesítésekor a test elülső része mozdul el a haladási irányba, míg a hátsó rész fixen marad.



1.3. ábra: Hernyó mozgása

3. **Kígyózó mozgás:** A kígyók sajátossága tekergőző mozgásuk, melynek során testük alakját a talaj síkjában változtatják, így jutnak a súrlódási erő segítségével előre.



1.4. ábra: Kígyó mozgása, és nyoma

A fenti csoportok persze nem biológiailag és törzsfejlődésileg egzakt csoportosítások, csupán támpontot adnak a modellezendő mozgásokhoz, hiszen létezik mindezen mozgások különböző keveréke is, ahol például a hossz is megváltozik, de a függőleges síkban is történik a hernyókra jellemző hullámzás, illetve létezik olyan mozgás is, ahol több hullámcsúcs is végigfut egyszerre a lény testén. A kígyók fajtánként különböző igen összetett mozgásáról pedig már ne is beszéljünk, vagy az ugrásra is képes lárvákról. Emiatt az egyes speciális esetek vizsgálata inkább csak tapasztalatszerzési kitérő, a modell viselkedésének minél jobb megismerését szolgálja ezzel elősegítve az optimális mozgás megtalálását.

2. fejezet

Modellalkotás

2.1. Helyváltoztatás értelmezése

Mivel modellünk célja a minél gyorsabb, és hatékonyabb helyváltoztatási módok keresése, ezért nem mehetünk el amellett, hogy definiáljuk, mit is értünk helyváltoztatáson. Ugyanis erre a látszólag egyszerű fogalomra még a szakirodalmak ([5] referencia listája) is gyakran teljesen eltérő definíciókat tartalmaznak.

A szakirodalomban ennek ellenére találhatók közös vonások:

1. a rendszer megváltoztatja a helyét a térben (itt már a hely fogalmának definiálása is problémás, lehet pl.: a súlypont vagy a rendszer egy kitüntetett pontjának pozíciója)
2. a helyváltoztatást a rendszeren belüli erők okozzák, amit úgy is meg lehet fogalmazni, hogy a rendszer helyváltoztatása a rendszer alakjának változtatásából ered
3. a környezettel való kapcsolat is elengedhetetlen a helyváltoztatáshoz (pl.: a talajról átadódó súrlódási erő)

A helyváltoztatás sok különböző definíciója több kérdést is felvet:

1. Hogyan definiáljuk a mozgó rendszer "helyét"?
2. A rendszer "helyét" milyen térben értelmezzük?
3. hogyan értelmezzük vagyis mérjük a rendszer helyváltoztatását?

Az továbbiakban a 2.1 fejezetben és alfejezeteiben taglalt leírás Joachim Steigenberger helyváltoztatásról szóló cikkén alapszik [5], amely a helyváltoztatásra próbál meg egyértelmű matematikai definíciót adni.

2.1.1. Testek mozgásának definiálása

Az alkalmazott mechanika tudományában makroszkopikus testek viselkedését írjuk le térben és időben. Azonban az [5]-ös irodalom szerint nem feltétlen kell merev testekkel foglalkozni, annak ellenére, hogy ez számunkra a legtermészetesebb. Kikötjük, hogy a

testek önálló individuumok tehát egymásba nem hatolhatnak be, és az $\mathbb{R}^3$ euklideszi térben figyelhetők meg. Ezek alapján a következő definíciót adhatjuk a test fogalmára:

A test egy olyan részecskékből álló β halmaz mely rendelkezik a $\varphi|\beta \rightarrow \mathbb{R}^3$ leképezések Φ halmazával és egy $\mu|\sigma(\beta) \rightarrow \mathbb{R}^+$ pozitív mérőszámmal (tömegnek megfelelő mennyiség) valamint igazak rá a következő tulajdonságok:

1. Minden $\varphi \in \Phi$ injektív, tehát a $\varphi^{-1}|\varphi(\beta) \rightarrow \mathbb{R}^3$ inverz leképezés létezik
2. Minden $\varphi \in \Phi$ leképezés $\varphi(\beta) =: \mathbf{B} \subset \mathbb{R}^3$ képe kompakt halmaz.
3. Bármely két $\varphi, \psi \in \Phi$ leképezés esetén a

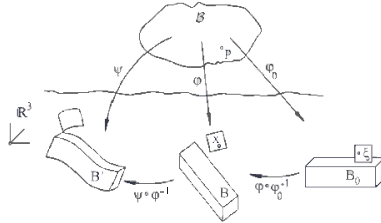
$$\psi \circ \varphi^{-1} =: f|\varphi(\beta) \rightarrow \psi(\beta), \quad (2.1)$$

kompozíció egy iránytartó és szakadásmentes leképezés az $\mathbb{R}^3$ térből saját magába.

4. Minden $\varphi \in \Phi$ -re vett $H \in \mathbb{R}^3$ iránytartó és diffeomorf (bijektív és folytonosan deriválható) leképezés esetén igaz, hogy:

$$H \circ \varphi =: \varphi \in \psi, \quad (2.2)$$

5. Minden $\varphi \in \Phi$ mérhető a $\sigma(\beta)$ σ -algebra függvényében, amelyen μ adott, és a $\mathbf{B} = \varphi(\beta)$ Borel halmazok $\beta(\mathbf{B})$ σ -algebrájának a függvényében.



2.1. ábra: Részecskék halmaza és konfigurációi

Tehát β egy absztrakt halmaz, melynek elemei a részecskék, amik minden $\varphi \in \Phi$ -n egy darab $\mathbb{R}^3$ -on értelmezett ponttal írhatók le, és anyagi pontoknak hívjuk őket. $\mathbf{B}$ tehát egy meghatározott véges kombinációja térfogatoknak, égióknak, felületeknek, íveknek és pontoknak (egymáshoz kapcsolódó vagy független 3-2-1 vagy 0 dimenziós részek) vagyis a fenti definíció a testek és a testekből álló rendszerek fogalmát is kezelni tudja, ami számunkra a féreg modellezésénél elengedhetetlen. Ennek bijektív folytonosan deriválható (diffeomorf) leképezése

$$f := \psi \circ \varphi^{-1}|\beta \rightarrow \beta' = \psi(\beta) \quad (2.3)$$

leírja a konfiguráció megváltozását (2.1. ábra). Ez a megfogalmazás egyszerűen alkalmazható több testből álló rendszerekre is, ahol $\mathbf{B}_1$ és $\mathbf{B}_2$ testekre akkor mondhatjuk, hogy egymással kapcsolatban állnak, ha

$$\mathbf{B}_1 \cap \mathbf{B}_2 \subset \partial \mathbf{B}_1 \cap \partial \mathbf{B}_2 \neq \emptyset \quad (2.4)$$

Egy test leírásakor praktikus egy $\varphi_0 \in \phi$ referencia helyzetet felvenni ($\mathbf{B}_0 := \varphi_0(\beta)$). Térbeli referencia rendszernek célszerű egy ortonormált vektorhármast választani $\mathbf{O} \in \mathbf{B}_0$ -ban. A modellalkotás során a későbbiekben mindig a mozgást végző rendszer (továbbiakban féreg) kezdeti helyzetéhez képest rögzített középpontú természetes koordináta rendszerrel fogunk dolgozni. Ebben a rendszerben minden részecske helyzete leírható a kezdeti állapot

$$\mathbf{B}_0 : \xi \mapsto p = \varphi_0^{-1}(\xi) \mapsto x = \varphi(p) \quad (2.5)$$

transzformációjával, mely megadható egy $\mathbf{h}$ folytonosan differenciálható koordináta transzformációval

$$\mathbf{h} := \varphi \circ \varphi_0^{-1} : \xi \mapsto x = h(\xi), \quad (2.6)$$

ahol ξ és x az adott konfigurációkban az egyes p részecskékhez rendelt helyvektor koordinátáit tartalmazó mátrix.

Vegyünk t valós paramétert egy időintervallumnak (t_0, t_1) , amin $\mathbf{B}_0 \rightarrow \mathbf{B}_t =: h(\mathbf{B}_0, t)$ a test mozgása, ha ez kifejezhető $\varphi_t|_{t \in (t_0, t)}$ -vel és $h(\cdot, t) := \varphi_t \circ \phi_0^{-1}$ -el, mely minden időpillanatban diffeomorfikus, tehát létezik az inverze és folytonos. Ebből az időfüggő koordinátatranszformáció

$$\xi \rightarrow x = h(\xi, t) \quad (2.7)$$

mely leírja a ξ anyagi pont helyzetét a térben és időben (Lagrange féle leírási mód). Ha $\mathbf{B}$ -re és $\mathbf{B}'$ -re fennáll, hogy $\mathbf{B}' = h(\mathbf{B})$, ahol h egy közvetlen egybevágó transzformáció $\mathbb{R}^3$ -on, akkor a testre mondhatjuk, hogy megtartja az alakját. Merev testek esetén ezt a feltételt minden esetben kikötjük. Ezen leírás elbonyolódhat amennyiben az alak és a pozíció függ az időtől továbbá kinematikai és dinamikai jelenségeket is figyelembe kell venni. E probléma egy lehetséges feloldása az $m_o := \mu \circ \varphi_0^{-1}$ tömegeloszlással rendelkező $\mathbf{B}_0$ tömegközéppontjának a meghatározásából ($\xi^* := \frac{1}{M} \int_{\mathbf{B}_0} \xi dm_0(\xi)$), és a tehetetlenségi főtengelek $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$ felhasználásából nyerhető $\mathbf{F}_0^* = \{\xi^*, \mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$ vonatkoztatási rendszer. Bármely más $\mathbf{B}$ helyzethez megadható egy ekvivalens $\mathbf{F}^* = \{\mathbf{x}^*; \mathbf{E}_1^*, \mathbf{E}_2^*, \mathbf{E}_3^*\}$. Az $\mathbf{F}_0^* \rightarrow \mathbf{F}^*$ meghatároz egy egybevágósági transzformációt, mely a test pozícióváltozását írja le a térben és időben. A $t \rightarrow \mathbf{F}_t$ leképezéssel pedig az időben változó alakú test útját is leírhatjuk a térben és időben, azzal a feltétellel, hogy az egyes $\mathbf{F}_t$ rendszerek egymással való kapcsolata a következő képpen van meghatározva: $h(\mathbf{B}_0, t) =: \mathbf{B}_t \rightarrow \mathbf{F}_t$.

Röviden összefoglalva a fentieket [5], elmondható, hogy egy test helyváltoztatását a legegyszerűbben a tömegközéppontjának mozgásával jellemezhetünk, mivel ekkor ahhoz, hogy a rendszer időben leírt pályáját meghatározzuk, nincs szükségünk a test minden részecskéjének a pozíciójára és teljes test aktuális orientációjára. A tömegközéppont elmozdulásának figyelését (3.1) fogjuk alkalmazni a féreg helyváltoztatásának a hatékonyságának a mérésére is a későbbiek során (3.1 részben)

2.1.2. Maguktól mozgó rendszerek, helyváltoztatás

Definíció szerint egy merev test alakja állandó és helyzete a térben az előzőekben taglalt formalizmussal megadható. Newton első törvénye értelmében egy merev test külső erők nélkül képtelen megváltoztatni mozgásállapotát.

Ez azonban testrendszerekről már nem mondható el. Egymással rendszert alkotó testek belső erők révén hathatnak egymásra. Ezek az erők származhatnak rugóktól, csillapításoktól, lineáris és forgó motoroktól, vagy izmoktól, hidraulikus rendszerektől, melyeket jelen modell során tömegnélküli virtuális komponensekként kezelünk. Newton harmadik törvényének értelmében ezek az erők páronként kioltják egymást (akció-reakció hatás), tehát nem okozhatják a tömegközéppont elmozdulását, de elforgathatják a test rendszert tömegközéppontja körül, vagy megváltoztathatják annak alakját.

Minden ilyen szögelfordulás vagy lineáris elmozdulás belső erők eredménye, melyeket a rendszer részét képező aktuátorok fejtenek ki. Ezeket a hatásokat gyűjtőnéven belső vezérlő erőknek hívjuk. Mivel ezek a belső vezérlő erők önmagukban nem képesek elmozdítani a testrendszer tömegközéppontját, ezért mindig szükség van valamilyen külső erőre, mely rendszerint a környezettel való kapcsolat során kialakuló kényszererő. Teljesen önállóan mozgó rendszerről akkor beszélhetünk, ha csak olyan külső erők lépnek fel, melyek valóban szükségesek a rendszer tömegközéppontjának elmozdulásához, semmilyen más külső erő nem játszik szerepet az elmozdulásban. Ezek alapján kijelenthetjük, hogy a helyváltoztatás "lelke" a belső vezérlő erők mozgássá történő konverziója a környezettel való interakció révén. Ez definíciószerűen megfogalmazva: magától mozgó rendszer helyváltoztatásáról beszélünk ha létezik valamilyen belső vezérlő erő, melynek hatására adott idő alatt $(t_0, t_0 + T)$, rajta kívül csak nem vezérlő külső erők lépnek fel, és ez idő alatt a testrendszer egyik pontja sem (a tömegközéppont sem) marad fixen vagy fut be ciklikus pályát

$$x^*(t_0 + T) = x^*(t_0) \wedge x(\xi, t_0 + T) \neq x(\xi, t_0) \quad \forall \xi \in \mathbf{B}_0 - ra. \quad (2.8)$$

Az imént [5] alapján kifejtett tények jól megfigyelhetők a féregmodellen is, ahol a hossz megváltoztatásával, vagy az egyes szelvények egymáshoz képesti szögelfordulásával érünk el haladó mozgást úgy, hogy a tömegközéppont elmozdulásához szükséges külső erő az érintkező pontok és a talaj között fellépő kényszerekből származik.

2.2. Talajjal való kapcsolat modellezése

A talajjal való érintkezés az elmozdulás kialakulásában kulcsszerepet játszik, ezért a modellalkotást a testek talajjal történő érintkezésekor kialakuló kölcsönhatások vizsgálatával kezdtem. Mint azt az előző fejezetben említettem a féreg elmozdulásában a száraz súrlódásnak van jelentős szerepe hiszen ez az a kényszererő ami közvetlenül ki fogja váltani az elmozdulást. A másik fontos jelenség amit meg kell vizsgálnunk a talajjal történő ütközés, hiszen amikor a modell megemeli majd a talajra visszafekteteti anyagi pontjait a mozgás intenzitásától függően kisebb nagyobb ütközésekkel kell számolnunk.

2.2.1. Száraz súrlódás

A száraz vagy más néven Coulomb súrlódás már a középiskolai tanulmányok során megismert fogalom, azonban egyetemi tanulmányaink során először a rezgésstanban [6] kerül újra elő, hiszen kezelése nem ideális kényszer lévén (a kényszererő iránya nem merőleges az érintkező felületekre), valamint a tapadás és megcsúszás eltérő viselkedése miatt számos megoldandó problémát vet fel.

A súrlódási erő tehát csak akkor lép fel ha valamilyen aktív erő működik a testen. Nagysága függ az érintkező felületek minőségétől, melyet egy μ mozgás közbeni (megcsúszás) és egy μ_0 tapadási együtthatóval jellemezhetünk, valamint a nagysága függ még felületre merőleges $\mathbf{F}_n$ nyomóerőtől is, iránya pedig az érintkezési sebességgel ellentétes (2.2. ábra).

Statikus esetben a súrlódási erő egyensúlyt tart az aktív erőkkel, és ekkor a súrlódási erő kisebb vagy egyenlő mint a tapadási súrlódási tényezővel számítható maximális érték:

$$F_s \leq \mu_0 \cdot F_n. \quad (2.9)$$

Amennyiben az aktív erő nagyobb mint a maximális tapadási súrlódási erő, akkor csúszás lép fel, és:

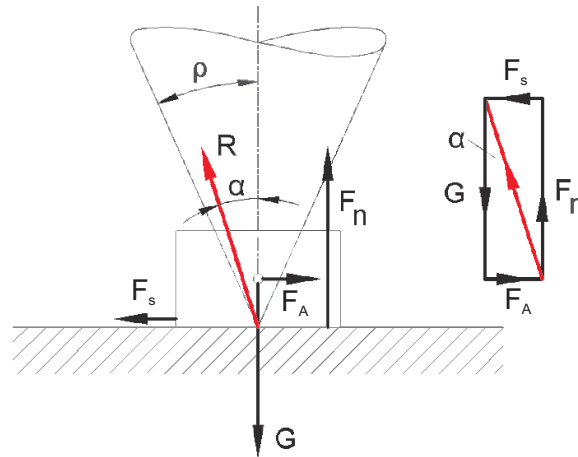
$$F_s = \mu \cdot F_n. \quad (2.10)$$

Láthatjuk, hogy a súrlódási erőnek mind nagysága mind iránya változhat a talajjal való érintkezés körülményeitől függően. Az eredő kényszererő lehetséges értékeit egy súrlódási kúppal szoktuk jellemezni, ahol az erő a paláston belül változhat vagy a paláston lehet tapadási esetben.

Egy adatott értékű $\mathbf{F}_n$, normális erőhöz a fenti képlet szerint a megcsúszás határán $\mathbf{F}_s = \mu_0 \cdot \mathbf{F}_n$ súrlódási erő tartozik. A két erő egymásra merőleges és

$$\tan \rho = \mu_0 \quad (2.11)$$

szöget zár be egymással. Belátható, hogy ha a felületeket elmozdítani akaró erő és a felületeket összeszorító erő eredőjének α szöge kisebb a ρ szögnél, akkor a felületek nem csúsznak el egymáson. Ez igaz akkor, ha az eredő erő egy ρ félkúpszögű kúpon belül marad, akármilyen irányú legyen is a felületeket elmozdítani akaró erő. Ezért a ρ szöget súrlódási félkúpszögnek hívják [7].



2.2. ábra: Súrlódási kúp [7]

Csúszó-mászó lények súrlódási együtthatója

Fontos megjegyezni, hogy a természet a legtöbb végtag nélküli lénynél egy apró trükköt alkalmaz a haladási irányba történő mozgás elősegítésére. Bizonyára sokan ismerik azt a jelenséget, mikor egy buzakalászt veszünk a tenyerünkbe, majd ezután ha a tenyerünket ütemesen nyitjuk és zárjuk, akkor a buzakalász mindig a hegye irányába fog haladni. Ennek oka a kalászon található kis "szőröknek" köszönhető, ugyanis emiatt a kalász súrlódási együtthatója nem azonos a két haladási irányba, így minden esetben tudni fogjuk a kalász merre indul el, holott a tenyerünkkel kifejtett erőhatásnak önmagában nem lenne ilyen irány meghatározó hatása. Ezt a jelenséget megfigyelhetjük a legtöbb féregfajtánál apró szőrök formájában [3], de kigyók pikkelyeinek (2.3. ábra) meghatározott mintája is ezt a jelenséget idézi elő.

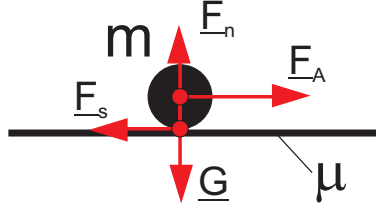


2.3. ábra: Kígyó pikkelyei

Ezen dolgozat modellje azonban mentes az ilyen irányfüggő súrlódási együtthatótól azért, hogy az alkalmazott vezérlőjelekkel szabadon választhassuk ki a haladási irányt, és mindkét haladási irányban teljesen egyenértékű jelenségeket vizsgálhassunk.

A száraz súrlódás numerikus modellezése

A féreg modellezésekor első lépésként egyetlen tömegpont talajjal történő érintkezését modelleztük, amelynek mechanikai modellje a 2.4. ábrán látható.



2.4. ábra: Súrlódási modell

Az anyagi pont súrlódásának modelljét egy szimuláción keresztül teszteltük. Súrlódási modellünkben tömegpont vízszintes talajon történő síkbeli mozgásakor vizsgáljuk a száraz súrlódás hatását.

A szükséges paraméterek:

1. gravitációs gyorsulás
2. az anyagi pont tömege
3. tapadási súrlódási együttható
4. csúszási súrlódási együttható

Továbbá, mivel numerikus megoldásról beszélünk szükségünk van még az iterációs paraméterekre: a lépésszámmra és a lépések közt eltelt időre (időlépés).

A megoldandó feladat tipikus kezdeti érték probléma, azaz adott a tömegpont kezdeti pozíciója és sebessége, valamint megadhatjuk az aktív F erő időbeni lefutását. Az ilyen problémák megoldására a mérnöki gyakorlat általában a közönséges negyedrendű Runge-Kutta eljárást alkalmazza, mely a következőképpen néz ki:

$$y_{n+1} = y_n + h \cdot \frac{a_n + 2 \cdot b_n + 2 \cdot c_n + d_n}{6}, \quad (2.12)$$

$$a_n = f(t_n, y_n), \quad (2.13)$$

$$b_n = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot a_n\right), \quad (2.14)$$

$$c_n = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot b_n\right), \quad (2.15)$$

$$d_n = f(t_n + h, y_n + h \cdot c_n) \quad (2.16)$$

,ahol $f(t, y)$ a megoldandó függvény.

Tehát a következő y_{n+1} érték az előző y_n értékből és a (2.13-2.16) meredekségek súlyozott átlagából valamint az időlépés h nagyságából számítható. A további modellek során is mindig ezen módszert alkalmaztam a numerikus számítások elvégzésére, hiszen a negyedrendű Runge-Kutta eljárás már kellően pontos, de még viszonylag gyorsan futtatható a jelenleg általánosan elérhető számítási kapacitásokkal.

A megoldandó egyenlet a haladás irányától függően (1 - pozitív irány, -1 - negatív irány, 0 - tapadás), melyet minden időpillanatban meghatározunk és eltárolunk két lehetséges variációt ölthet:

- tapadás:

$$F_s = F_A. \quad (2.17)$$

Tehát a test tapad és pontosan akkora erő hat rá amekkora az F_A aktív erő. Azonban, ha az F_s súrlódási erő abszolútértéke nagyobb mint a maximális tapadási súrlódási erő (2.9) akkor abban az időpillanatban vagyunk ahol a test megmozdul, tehát az ebben az időpillanatban érvényes súrlódási erő:

$$F_s = \mu \cdot m \cdot g \cdot \text{sign}(F_A), \quad (2.18)$$

ahol $\text{sign}(F_A)$ az aktív erő iránya, ugyanis a Runge kutta eljárásnak majd az $F_A - F_s$ -ből számítunk gyorsulást.

- csúszás:

Csúszás esetén egyszerűen a (2.18) egyenletből számítjuk ki a súrlódási erőt, és ismét az $F_A - F_s$ -ből számítunk gyorsulást.

A Runge-Kutta eljárás azonban nem képes kielégítően megoldani feladatot ugyanis a becsült meredekségek és a numerikus instabilitások miatt felléphetnek olyan esetek, hogy a haladási irány letapadás nélkül, egy időlépésen belül változik meg. Ezt úgy tudjuk programozástechnikailag kezelni, hogy megvizsgáljuk az előző iteráció eredményének haladási irányát majd az adott iterációs lépésben kapott haladási irányt. Amennyiben előjelváltás történt be kell avatkoznunk, és a tárolt haladási irány át kell állítanunk a tapadási értékre, vagyis 0-ra és a kapott sebességet az aktuális iterációs lépésben le kell nulláznunk.

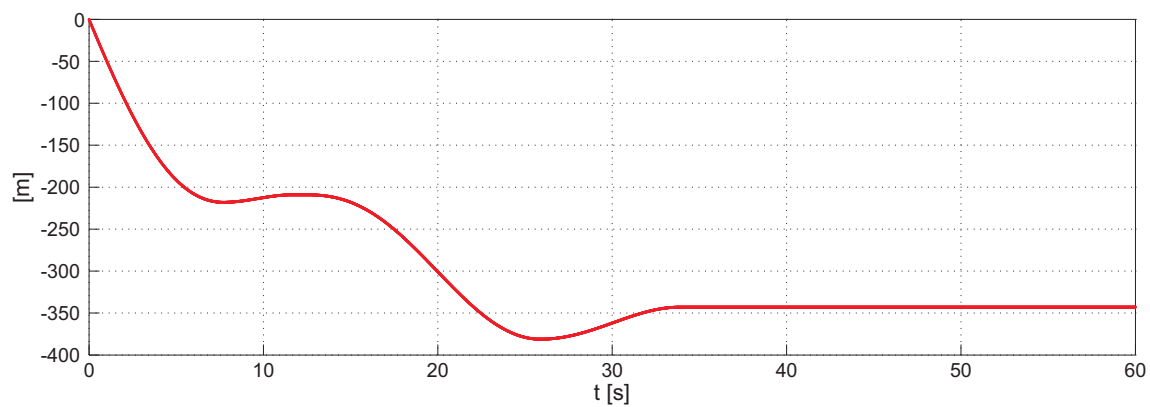
Ezzel az egyszerű módszerrel kiküszöbölhetjük a numerikus megoldási módból fakadó hibákat. A módszer helyességét az alábbi futási eredmény is mutatja, melynek paraméterei:

1. $g = 9,81[\frac{m}{s^2}]$
2. $m = 1[kg]$
3. $\mu = 0,15[-]$
4. $\mu_0 = 0,35[-]$

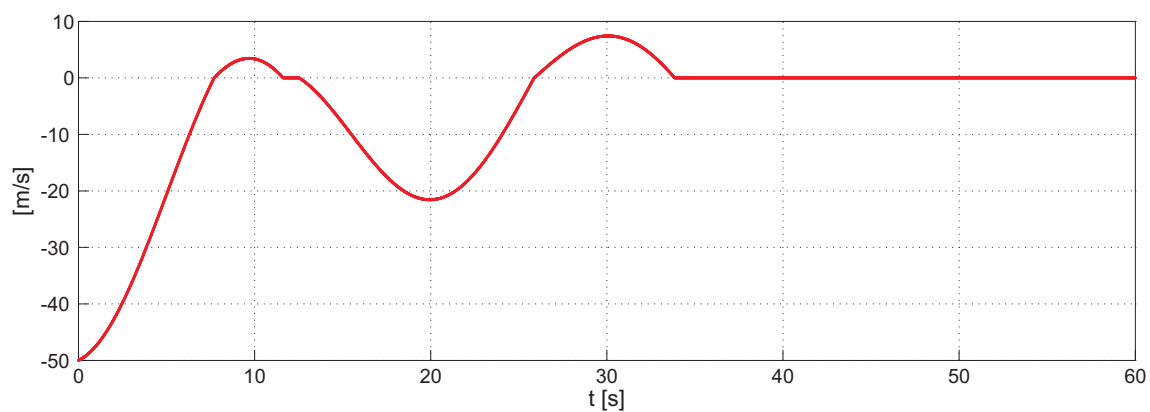
Az aktív erő pedig az alábbi matlab függvénnyel állítható elő:

$$\mathbf{F} = 1/8 * (60 - \mathbf{t}). * \sin(0.3 * \mathbf{t}),$$

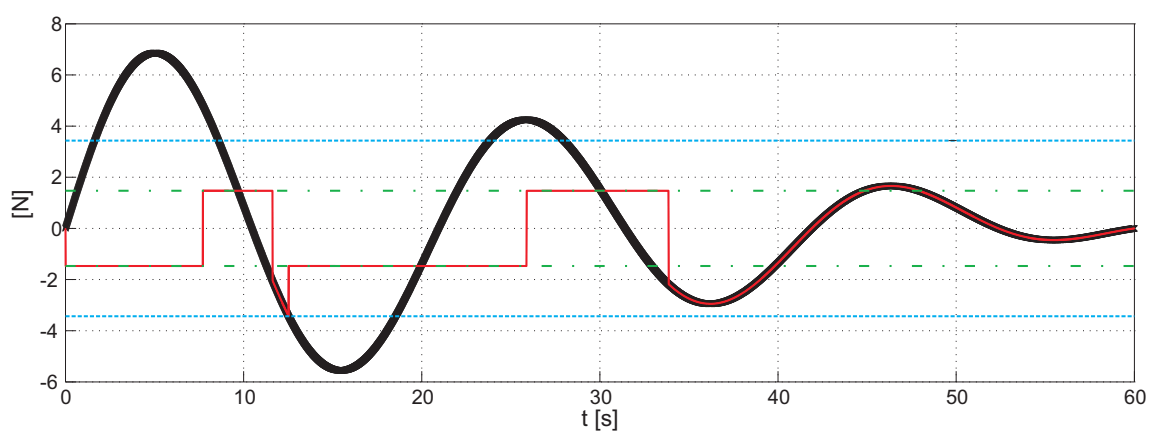
ahol $\mathbf{t}$ egy nullától a a szimulációs idő végéig a választott lépésközzel feltöltött vektor, és a $.*$ jelölés a vektorok elemenkénti összeszorzását jelenti (Hadamard szorzat). Az így kapott $\mathbf{F}$ vektornak minden szimulációs időlépésben van értéke a pontokra illeszthető görbe pedig egy csillapodó szinusz függvény képe.



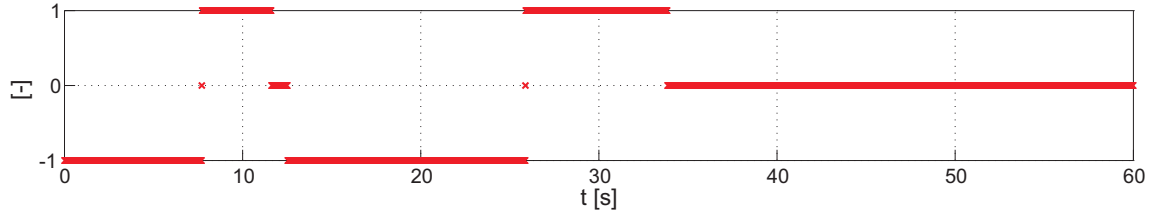
2.5. ábra: Pozíció



2.6. ábra: Sebesség



2.7. ábra: Külső erő és a súrlódási erő



2.8. ábra: Haladás iránya

Az erőket ábrázoló grafikonon (2.7. ábra) a zöld vonalak a csúszási súrlódási együtthatóval számított maximális súrlódási erők, a kék vonal jelöli a tapadási súrlódás maximumát, a fekete vonal az aktív erő, a piros pedig a súrlódási erő. Látható, hogy a modell kellő pontossággal modellezi a súrlódási erő viselkedését.

2.2.2. Talajjal történő ütközés

Ha két test mozgásuk során (elég ha csak az egyik mozog) érintkeznek, lendületük és mozgási energiájuk megváltozik. Ezt a jelenséget a Dinamika című tantárgy [6] utolsó tananyagaként tanítják egyetemünkön. Az ütközés pillanatában lejátszódó fizikai folyamatok nehezen leírhatók, ezért az általános mérnöki feladatokban nem is szokták körültekintően vizsgálni, inkább az ütközés előtti és ütközés utáni állapotokat hasonlítják össze, és a k ütközési tényezőt vizsgálják. Egyetemi tanulmányaim során az ütközést merev testek pontszerű érintkezéseként modelleztük, ahol a merev testek az érintkezési pontban lokális alakváltozást szenvednek. A következő közelítésket mondhatjuk el az ütközés tulajdonságairól:

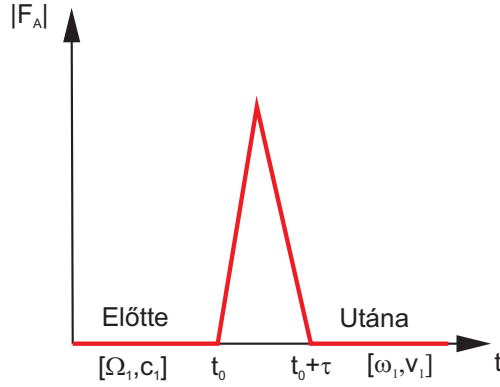
1. a τ ütközési időtartam "kicsi", az $|\mathbf{F}_\mathbf{A}|$ ütközési erő "nagy" (2.9. ábra)
 - (a) a τ ütközési időtartam alatt a testek helyzete közel változatlan
 - (b) az $|\mathbf{F}_\mathbf{A}|$ erő mellett minden más erő elhanyagolható
2. A merev testek lokálisan rugalmas ($k = 1$), lokálisan-rugalmas képlékeny ($0 < k < 1$), vagy képlékeny alakváltozást ($k = 0$) szenvednek.
3. Az ütközési pontban nincs súrlódás.

Jelölési szintaktikánkban az ütközés előtti sebességeket mindig $\mathbf{c}$ -vel az ütközés utáni sebességeket $\mathbf{v}$ -vel jelöljük.

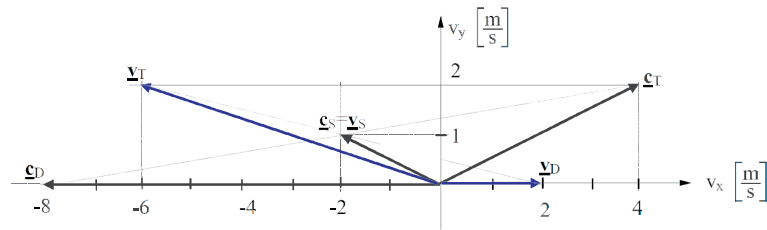
Centrikus ütközés

Az adott test szempontjából centrikus ütközésről akkor beszélünk ha az ütközési sík normálisán a test súlypontja rajta van. A sebességek számításához az impulzusmegmaradás és a k ütközési tényező függvényében vett sebességek normál irányú komponensének kicserélődését használjuk fel. Szemléltetésre Maxwell ábrát használunk, amin hasonló háromszögek révén szemléletesen látható a sebességek számítási módja:

A számítás során használható egyenletek a következők:



2.9. ábra: Erőimpulzus az ütközés során



2.10. ábra: Példa Maxwell ábrára

- Impulzusmegmaradás:

$$m_1 \cdot v_1 + m_2 \cdot v_2 = m_1 \cdot c_1 + m_2 \cdot c_2 = (m_1 + m_2) \cdot c_S, \quad (2.19)$$

ahol a súlyponti sebességek megegyeznek ($v_S = c_S$)

- Ütközési tényező:

$$k = \frac{v_1 - v_S}{c_1 - c_S} = \frac{v_2 - v_S}{c_2 - c_S}. \quad (2.20)$$

A féreg anyagi pontokkal való modellezése során a továbbiakban csak centrikus ütközés számítására lesz szükség.

Az ütközési tényező meghatározása

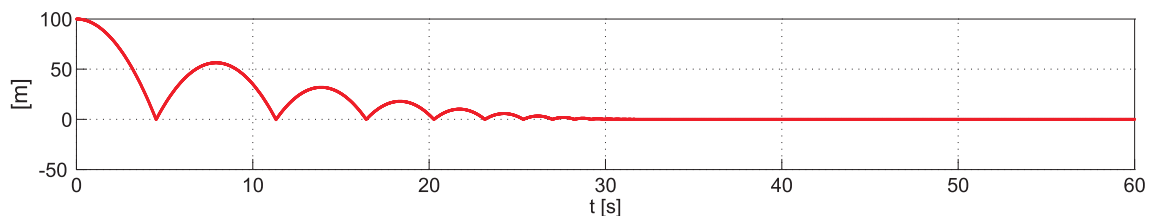
Az ütközési tényezőt alapvetően mérések útján tudjuk meghatározni, mivel az ütközések során lejátszódó pontos jelenségek szimulációja meglehetősen bonyolult feladat. A legegyszerűbb módja az energiák megváltozásából számítani. Például egy földre leejtett teniszlabda kezdeti potenciális energiája $m \cdot g \cdot H$ ahol H a leejtési magasság. Mivel a teniszlabda nem tökéletesen rugalmasan fog visszapattanni a földről, hanem az ütközés közben elszorított alakváltozáskor mozgási energia nyelődik el, ezért a visszapattanási magasság h lesz mely kisebb H -nál. h -val kiszámítható a test potenciális energiája az ütközés utáni maximális magasság elérésnek pillanatában, a kettő hányadosából pedig számítható az ütközési tényező.

$$k = \sqrt{\frac{h}{H}}. \quad (2.21)$$

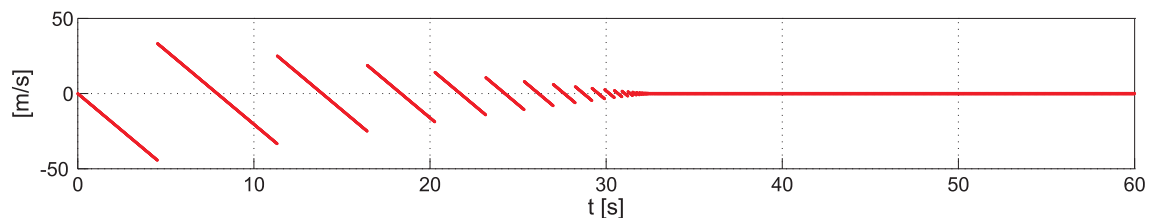
Ez például egy teniszlabda esetében 75% körüli érték.

Az így kapott ütközési tényező segítségével elkészíthetjük a legegyszerűbben programozható ütközési modellt, ahol minden időpillanatban figyeljük az anyagi pont talajtól vett magasságát. Ha ez a magasság zérus vagy az alatti (a diszkrét időlépés miatt ilyen is előfordulhat), továbbá a sebesség talajra merőleges részének iránya a talajba befelé mutat, akkor a talaj síkjára merőleges sebességkomponens előjelét megfordítjuk és az ütközési tényező értékének megfelelően csökkentjük nagyságát.

A legyszerűbb szimulációs eset, ahol a tömegpontot adott magasságból függőlegesen zérus kezdősebességgel leejtjük súrlódásmentes talajra és csak a gravitáció hat rá:



2.11. ábra: Függőleges irányú pozíció



2.12. ábra: Függőleges irányú sebesség

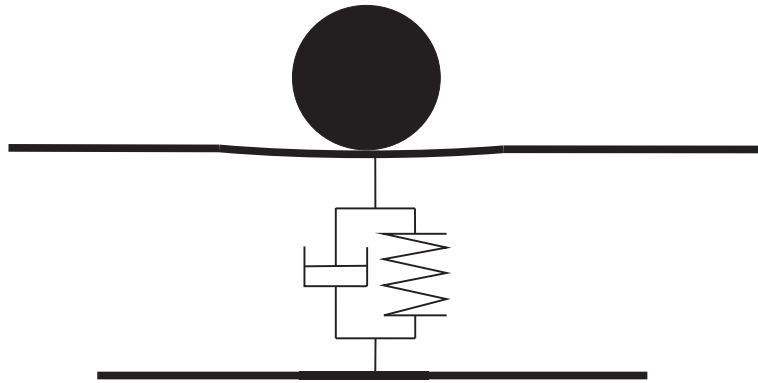
Látható, hogy ezzel a modellel a sebesség szakadásos (2.12. ábra), ami könnyen vezethet numerikus problémákhoz (a szakadási pontokban végtelen nagy a gyorsulás), továbbá az ütközés minden esetben a választott lépésköz időtartamára esik, ami a valóságban hosszabb ideig is tarthat. Mivel a féregmodell haladásánál nagyon kritikus a súrlódási erő hatása, ezért szükséges, hogy a modellünkben a talajjal való érintkezés egy véges hosszúságú ideig történjen az egy időlépésnyi időtartam helyett. Ezen problémák feloldására másfajta ütközési modellt kell választani.

Az ütközés rugalmas modellje

Látható, hogy a sebességben lévő szakadást nem tudjuk megszüntetni ha az ütközés előtti pillanaból direkt módon próbálunk következtetni az ütközés utánra. Szükségünk

van egy olyan modellre, mely képes kezelni az ütközéskor lejátszódó folyamatokat, nevezetesen a deformációból és az ütközésből származó sebességváltozásokat. A pillanat-szerűen lezajló ütközésnél további probléma, hogy a súrlódást nem tudjuk figyelembe venni, ami mint azt már a 2.1 részben is kifejtettük elengedhetetlen a helyváltoztatáshoz.

Ezért a talajt egy rezgéstani megközelítéssel élve (2.13. ábra) egy csillapítási és egy merevségi tényezővel modellezzük, melyek akkor fognak a tömegpontra hatni, amikor az érintkezik a talajjal (vízszintes talaj esetén ez $z \leq z_0$ feltétellel írható le). Az egyszerű modellezhetőség érdekében jelen munkában csak vízszintes talajon való mozgással foglalkoztunk.



2.13. ábra: Talajmodell

Ekkor a rezgéstanból [6] jól ismert egy szabadságfokú lengőrendszer egyenletével kiszámítunk egy $\mathbf{F}_t$ talajerőt, mellyel módosítjuk a z irányú erőkomponenst melyből a Runge-Kutta eljárásnak visszaadott z irányú gyorsulást számítjuk:

$$F_t = -s \cdot z - k \cdot \dot{z}. \quad (2.22)$$

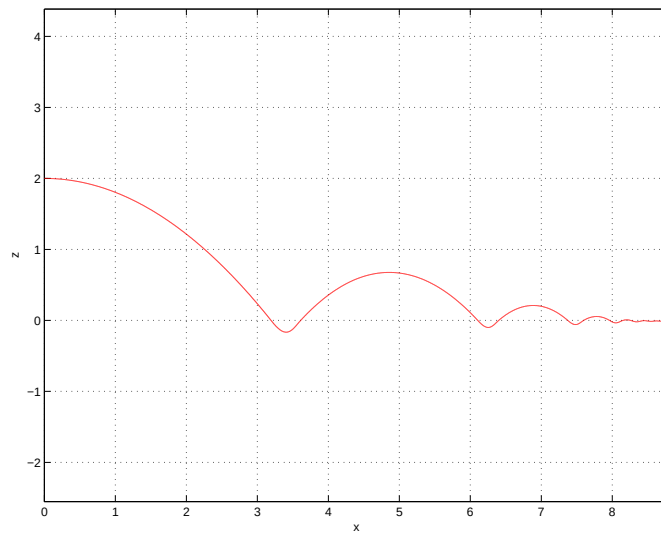
Ez modell képes kezelni az ütközésből fakadó energiaveszteséget, valamint megvalósítja az ütközés folytonosságát, aminek következtében a sebességben nem lesz szakadás és a súrlódási erő is képes működni. A csillapítás és merevség értékét egyfajta szabályzóparaméterként is felfoghatjuk amik biztosítják, hogy a talajból származó geometriai kényszer ($z \geq z_0$) teljesüljön. Hasonló megfontolással kezeljük majd a modellezéshez szükséges geometriai kényszereket is, ezt a 2.3.1 fejezetben a [8, 9, 10] szakirodalom alapján tárgyaljuk.

A modell szemléltetésére vegyünk egy egyszerű példát, ahol a kezdeti indítási feltételek a következők:

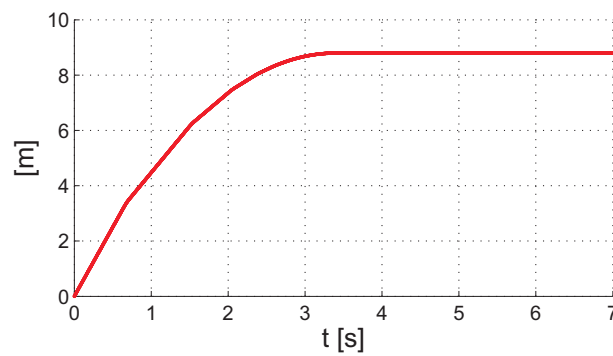
- $m = 1[kg]$
- $g = 9,81[\frac{m}{s^2}]$
- $x = 0[m]$
- $z = 2[m]$

- $\dot{x} = 5\left[\frac{m}{s}\right]$
- $\dot{z} = 0\left[\frac{m}{s}\right]$

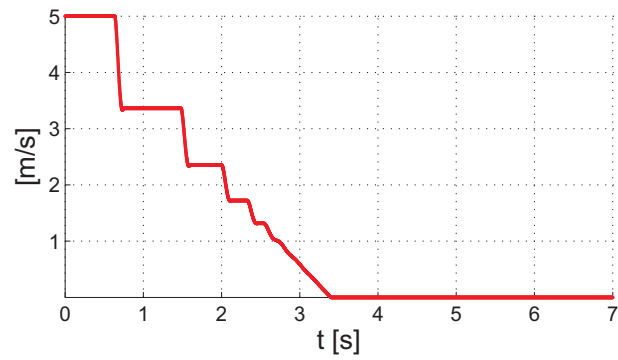
A talaj merevségét és csillapítását úgy választottuk, hogy ne süllyedjen bele túlzottan a tömegpont, és megfigyelhető legyen a pattogás és az ütközési pillanatokban bekövetkező tapadás vagy csúszás. A testre a jelenlegi szimulációban csak a gravitációs erő hat, hogy könnyen értelmezhető grafikonokat kapjunk.



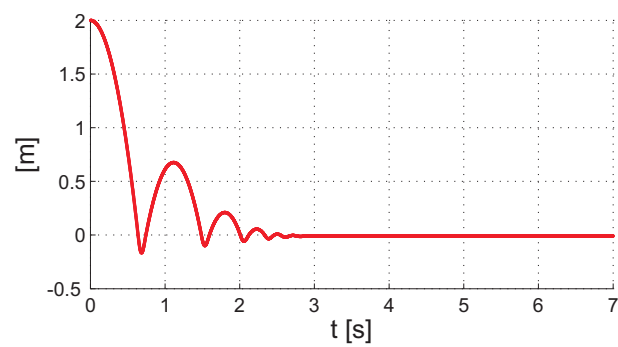
2.14. ábra: A tömegpont pályája



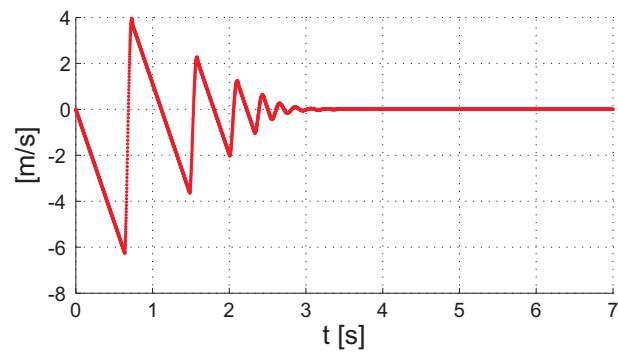
2.15. ábra: x irányú pozíció



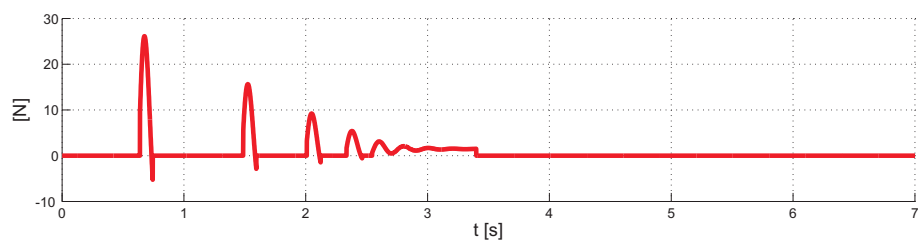
2.16. ábra: x irányú sebesség



2.17. ábra: z irányú pozíció



2.18. ábra: z irányú sebesség



2.19. ábra: x irányú erők

A 2.19. ábrán jól megfigyelhető a súrlódási erő életbelépése a lepattanások pillanataiban, a 2.18. ábrán pedig láthatjuk a sebesség kvázi folytonos változását a talajjal történő érintkezés időtartama alatt. Kvázi folytonos változásról azért beszélünk, mert tökéletesen folytonos változás disztkrét időlépésekkel nem érhető el, azonban mintavételes rendszerről lévén szó az, hogy minden mintavételi időpontban olyan sebességértékek vannak, melyek egymáshoz képesti változása kicsi ezért a modellt alkalmas a féregmodellnél is a talaj szimulációjára, hiszen az ütközések egy időpillanatnál tovább tartanak, így tudunk súrlódási erőt számítani, valamint a sebesség folytonos változása miatt nincsenek végtelen gyorsulásaink, amik instabillá tehetnék volna a számításaink.

2.3. A főregmodell

A 2.2-es fejezetben körüljártuk a főregmodell helyváltoztatásához nélkülözhetetlen környezettel való kapcsolatot, és létrehoztunk egy olyan modellt, ami képes kezelni egyidejűleg pattogó és megcsúszó mozgásokat.

Ahhoz hogy a 1.1-es részben megfogalmazott kritériumnak, azaz univerzális, minél többféle mozgást reprodukálni tudó modellt készítsünk a következő szempontokat kell figyelembe vennünk:

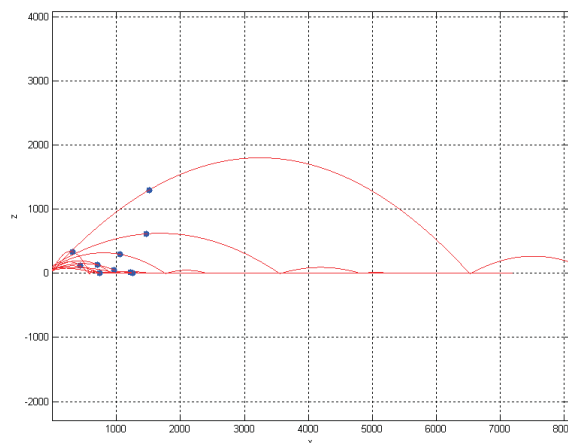
1. Tetszőleges számú szelvény
2. Többféle megvalósítható vezérlés

2.3.1. A főregtest összeállítása

Logikusan továbbhaladva a modellezési folyamatban a következő lépés, hogy tetszőleges számú anyagi pontot tudjunk kezelni. Ez főként programozástechnikai feladat ezért csak röviden ismertetem.

Azt kikötjük a későbbiek során is a főregmodellünkben, hogy homogén felépítésű, tehát minden anyagi pont ugyanolyan tulajdonságokkal rendelkezik. Emiatt annyi a feladatunk, hogy az eddig egy darabbal dolgozó Runge-Kutta eljárásnak vektorokat adunk, melyek megfelelően indexelve tartalmazzák az egyes pontok pozícióját, sebességét, és a rá ható erőket. Kezdeti feltételeket az elemszámtól függő, megfelelően elkészített függvényekkel és véletlenszám generátorokkal adtam meg.

A 2.2.1 súrlódásról szóló rész szimulációjának ábráit 2.5-2.8 elemenként itt is ki-rajzolja a program, tehát egyesével meggyőződhetünk róla, hogy reálisak e a kapott eredmények. De itt egy olyan ábrát tekinthetünk meg, amin az összes anyagi pont pattogó mozgása látható:



2.20. ábra: Sok tömegpont szimulációja

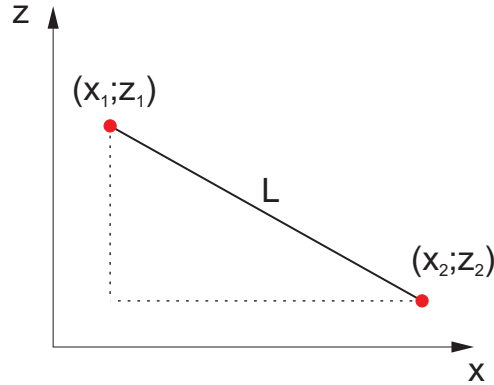
Ezzel modelleztük a főregmodell talajjal érintkező tetszőleges számú pontjait. Azonban mint az a 2.20. ábrán is látható ezek az anyagi pontok egymástól teljesen függetlenek. Következő lépésként kényszereket kell definiálnunk az anyagi pontok közé. Erre

először egy egyszerű időben állandó geometriai kényszert fogunk alkalmazni, melyet az alábbi módon fogalmazhatunk meg tetszőleges számú anyagi pont között:

$$(x_i - x_{i+1})^2 + (z_i - z_{i+1})^2 = L^2, \quad (2.23)$$

ahol i és $i + 1$ két szomszédos anyagi pont indexe L a pontok közti kívánt távolság.

Ez grafikusan ábrázolva egy természetes koordinátákkal megadott két pont közti távolság, amit egyenlővé teszünk a kívánt L hosszal:



2.21. ábra: Geometriai kényszer számítása

Az n index az elemszámot jelöli a (2.23) egyenletben, ami az $\mathbf{x}$ vektor felépítése miatt szükséges. Két dimenzióban modellezünk természetes koordináták segítségével, tehát az $\mathbf{x}$ vektorba minden pontnak tároljuk az $x, z, \dot{x}, \dot{z}$ értékeit. Ezt a vektort minden időpillanatban eltároljuk, így a későbbiekben könnyedén készíthetünk animációkat.

$$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n \ z_1 \ z_2 \ \dots \ z_n \ \dot{x}_1 \ \dot{x}_2 \ \dots \ \dot{x}_n \ \dot{z}_1 \ \dot{z}_2 \ \dots \ \dot{z}_n]^T. \quad (2.24)$$

Ezzel definiáltuk a szükséges geometriai kényszert ahhoz, hogy immáron ne egy rendezetlen tömegponthalmazzal legyen dolgunk, hanem egy megfelelően szelvényekre osztott testrendszerünk legyen, mely n darab talajjal érintkezni képes ponttal és $n - 1$ db szelvényt rendelkezik. Az így kapott modell egy holonom szkleronom rendszer tehát csak geometriai kényszerekkel rendelkezik, melyek nem függenek az időtől. Ez matematikailag megfogalmazva:

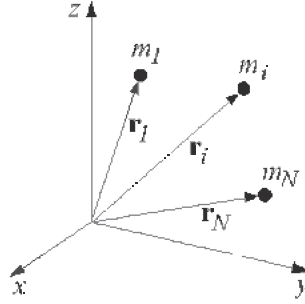
$$f_\alpha(\mathbf{q}) = 0; \quad \alpha = 1 \dots g. \quad (2.25)$$

Ahhoz azonban, hogy egy geometriai kényszereket tartalmazó mozgásegyenlet szimulációját el tudjuk végezni szükséges valamilyen megoldási módszert választanunk.

Az elsőfajú Lagrange egyenlet

Ahhoz, hogy ezt a tetszőleges elemszámú testrendszert mechanikailag kezelni tudjuk szükség van egy olyan módszerre, mely képes kezelni a modularitást. Erre a természetes koordinátákkal felírt elsőfajú Lagrange egyenlet mátrixos alakját választottam.

Csató Bertalannal közösen írt 2010-es TDK munkánk [11] során már részletesen levezettük és alkalmaztuk az elsőfajú Lagrange egyenletet ezért itt most csak a főbb



2.22. ábra: Anholonom, reonom mechanikai modell

lépéseket említtem meg, egy anyagi pontokból álló holonom, reonom esetben, mely a féregmodellünkhöz szükséges kényszerek jellegét lefedi.

Az elsőfajú Lagrange egyenlet levezetésénél Newton második axiómáját használjuk, úgy, hogy a $\mathbf{K}$ kényszererőket és az $\mathbf{F}$ külső aktív erőket szétválasztva írjuk be az egyenletbe:

$$m_i \cdot \ddot{\mathbf{r}}_i = \mathbf{F}_i + \mathbf{K}_i, \quad (2.26)$$

ahol $\mathbf{F}$ az anyagi pontra ható külső erővektor, míg $\mathbf{K}$ a kényszererők vektora. Ezt 1-től n -ig összegezzük minden anyagi pontra és megszorozzuk δr_i -vel:

$$\sum_{i=1}^n (m_i \cdot \ddot{r}_i - \mathbf{F}_i - \mathbf{K}_i) \cdot \delta \dot{r}_i = 0. \quad (2.27)$$

δr_i értékének pedig teljesíteni kell a következő egyenletet:

$$\sum_{i=1}^n \frac{\partial f_\alpha}{\partial r_i} \cdot \delta \dot{r}_i = 0, \quad (2.28)$$

ami a virtuális sebességekkel felírt kényszeregyenlet, ahol f_α a geometriai kényszerekt tartalmazó vektor α szerint indexelve, $\delta \dot{r}_i$ pedig a virtuális sebességek i szerint indexelve. Ezután a $\mathbf{K}$ kényszererőket tartalmazó vektort a Lagrange féle szélsőértékszámítással fejezzük ki, ahol λ_α a geometriai kényszererő nagyságát kifejező Lagrange multiplikátor.

$$\mathbf{K}_i = \sum_{\alpha=1}^g \lambda_\alpha \cdot \frac{\partial f_\alpha}{\partial r_i} \quad (2.29)$$

Behelyettesítve a (2.27) egyenletbe a (2.29) egyenletet a zárójelben lévő rész lehet zérus $\delta \dot{r}_i$ tetszőleges értékeire. Átrendezve ez az elsőfajú Lagrange egyenlet, azonban a számítások során a programozástechnikailag jobban kezelhető mátrixos alakot használjuk:

$$\mathbf{M} \cdot \ddot{\mathbf{q}} + \boldsymbol{\phi}_{\mathbf{q}}^T \cdot \boldsymbol{\lambda} = \mathbf{Q} \quad (2.30)$$

$$\boldsymbol{\varphi}(\mathbf{q}) = 0. \quad (2.31)$$

Az egyes jelölések alaphalmaz és jelentése:

- $\mathbf{M} = \mathbb{R}^{n \times n}$ a tömegmátrix
- $\mathbf{q} = \mathbb{R}^n$ a nem minimális számú természetes koordinátákat tartalmazó vektor
- $\mathbf{Q} = \mathbb{R}^n$ az aktív erők vektora
- $\boldsymbol{\varphi} = \mathbb{R}^g$ a geometriai kényszerek vektora
- $\boldsymbol{\lambda} = \mathbb{R}^g$ Lagrange multiplikátorok vektora (a kényszer erők nagyságával arányos skalárok)
- $\boldsymbol{\phi}_q = \mathbb{R}^{g \times n}$ a geometriai kényszerek helykoordináták szerinti Jacobi mátrixa

A numerikus megoldó stabilizálása

A levezetett elsőfajú Lagrange egyenlet numerikus megoldása során gyakran tapasztaljuk, hogy a kényszeregyenleteink nem teljesülnek megfelelően. Ez azért következik be, mert a számítógép csak véges pontossággal képes tárolni a számokat így például egy egyszerű hossz kényszer (2.23) a numerikus integrálások és deriválások során a folyamatos kerekítési hibák miatt egy idő után már nem ugyanazt az L hosszt fogja adni, hanem egy $L + \delta$ -t, ahol δ a halmozódó kerekítési hiba. Ez pedig a megoldandó hipermátrixot szingulárisá teheti. Emiatt stabilizáló eljárást kell alkalmaznunk a kényszerekre, így biztosítva azok minden időlépésbeni teljesülését.

Az általunk választott stabilizáló eljárás a jól ismert Baumgarte stabilizáció melyet még 1972-ben publikált J. Baumgarte [8]. Ez a módszer kifejezetten a numerikus instabilitások kezelésére lett kidolgozva. Hogy ezt jobban megérthessük nézzük meg egy egyszerű mozgásegyenleten, hogy is tehetjük stabillá az analitikusan stabilitás határán lévő egyenletet numerikus megoldások során is.

Vegyünk egy egyszerű esetet (2.23. ábra), melyben két anyagi pontot egy holonom szkleronom kényszerrel kapcsolunk össze és az egyik pontot egy külső erővel meghúzzuk:



2.23. ábra: Mechanikai modell példája a baumgarte stabilizációra

$$m_1 \cdot \ddot{x}_1 = Q_1, \quad (2.32)$$

$$m_2 \cdot \ddot{x}_2 = Q_2, \quad (2.33)$$

$$\varphi(x) = x_1 - x_2 - L = 0, \quad (2.34)$$

ahol

$$Q_1 = F; \quad Q_2 = 0. \quad (2.35)$$

Könnyen belátható, hogy analitikus esetben ez a kényszer az $x_1 = 0, x_2 = -L, \dot{x}_1 = \dot{x}_2 = 0$ kezdeti feltételekre azonosan nulla tehát a kényszer minden időpillanatban teljesül.

Ahhoz, hogy numerikus módszert alkalmazhassunk célszerű az egyenlet rendszert az elsőfajú Lagrange egyenlet mátrixos alakjára átírni, melyhez szükség van a kényszer egyenlet második idő szerinti deriváltjának előállítására:

$$\dot{\varphi} = \phi_q \cdot \dot{\mathbf{q}}, \quad (2.36)$$

$$\ddot{\varphi} = \dot{\phi}_q \cdot \dot{\mathbf{q}} + \phi_q \cdot \ddot{\mathbf{q}}, \quad (2.37)$$

A második deriváltból a leíró koordináták gyorsulása kifejezhető:

$$\phi_q \cdot \ddot{\mathbf{q}} = -\dot{\phi}_q \cdot \dot{\mathbf{q}}, \quad (2.38)$$

amelyet hozzáírva a mozgásegyenlethez a következő mátrixos alakot kapjuk:

$$\begin{bmatrix} \mathbf{M} & \phi_q^T \\ \phi_q & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ -\dot{\phi}_q \cdot \dot{\mathbf{q}} \end{bmatrix}. \quad (2.39)$$

ahol az $\mathbf{M}$ mátrix m_1 és m_2 -ből álló diagonál mátrix, ϕ_q pedig a (2.34) kényszer helykoordináták szerinti deriváltja. Az $\mathbf{y}$ a természetes koordináták idő szerinti deriváltjainak vektora, $\mathbf{Q}$ pedig Q_1 és Q_2 -ből álló általános erővektor.

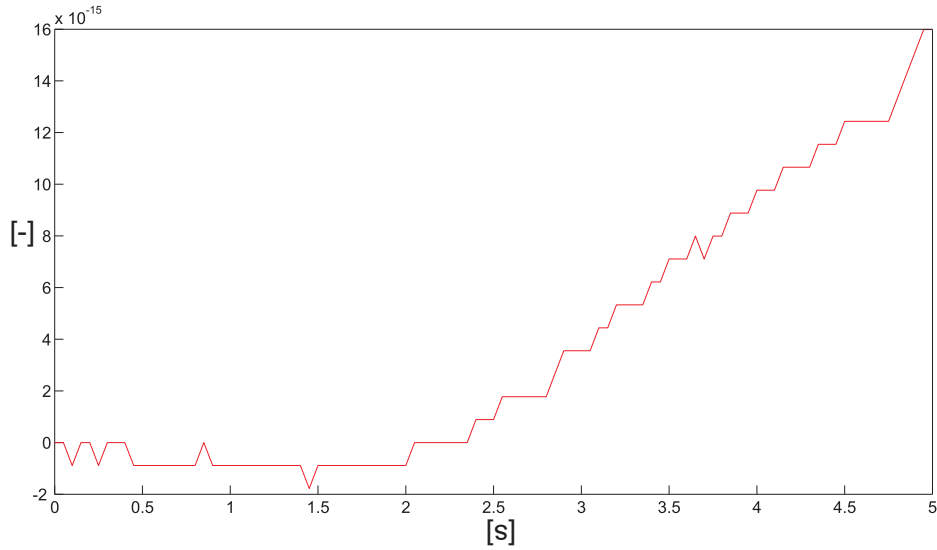
Numerikus megoldónak ismét a Runge-Kutta eljárást (2.12) használjuk, de megjegyzendő, hogy az Euler módszer egyszerűbb de az elérhető pontosság azonos időlépés mellett kisebb.

A szimuláció paraméterei:

- $m_1 = 10[kg]$
- $m_2 = 5[kg]$
- $Q_1 = 10[N]$
- $Q_2 = 0[N]$
- $L = 5[m]$
- $h = 0.05[s]$

A kezdeti feltételek:

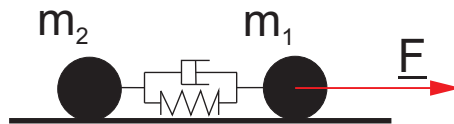
- $x_1 = 0[m]$
- $x_2 = -L[m]$
- $\dot{x}_1 = 0[\frac{m}{s}]$
- $\dot{x}_2 = 0[\frac{m}{s}]$



2.24. ábra: Stabilizáció nélkül a kényszeregyenlet számításakor kapott hiba

Látható, hogy numerikus megoldót stabilizáló eljárás nélkül használva, a 2.24. ábrán is látható a halmozódó kerekítési hibák miatt a kényszerünk már rövid futási időnél ($t_{max} = 5[s]$) és nem túl nagy lépésköznél sem teljesül tökéletesen már egy ilyen egyszerű esetben sem.

Emiatt szükség van a kényszeregyenlet olyan átírására mely mechanikailag közel egyenértékű, numerikusan azonban stabil. Végiggondolva a legegyszerűbb ilyen megközelítés ha a jelenleg merev rúdként definiálható kényszert (2.34) lecseréljük egy csillapításra és egy rugalmas tagra a rendszerben (2.25. ábra). Ezáltal a numerikus számítás hibáiból eredő kvázi "zavarásnak" felfogható jelenségek a számítás előrehaladtával az egy szabadságfokú lengőrendszer közismert csillapodó megoldása révén kioltódnak.



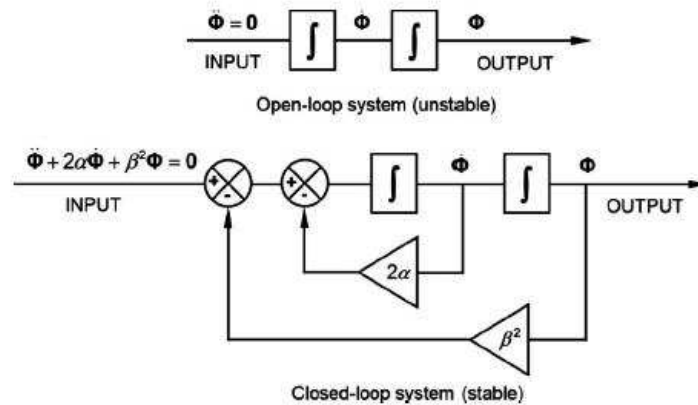
2.25. ábra: Baumgate stabilizáció alapján módosított modell

A kényszert tehát kétszeres idő szerinti deriválással írjuk át gyorsulásra, majd egészítsük ki ezt két taggal mely egy csillapításnak és egy rugalmasságnak fogható fel:

$$\ddot{\varphi} + 2 \cdot \alpha \cdot \dot{\varphi} + \beta^2 \cdot \varphi. \quad (2.40)$$

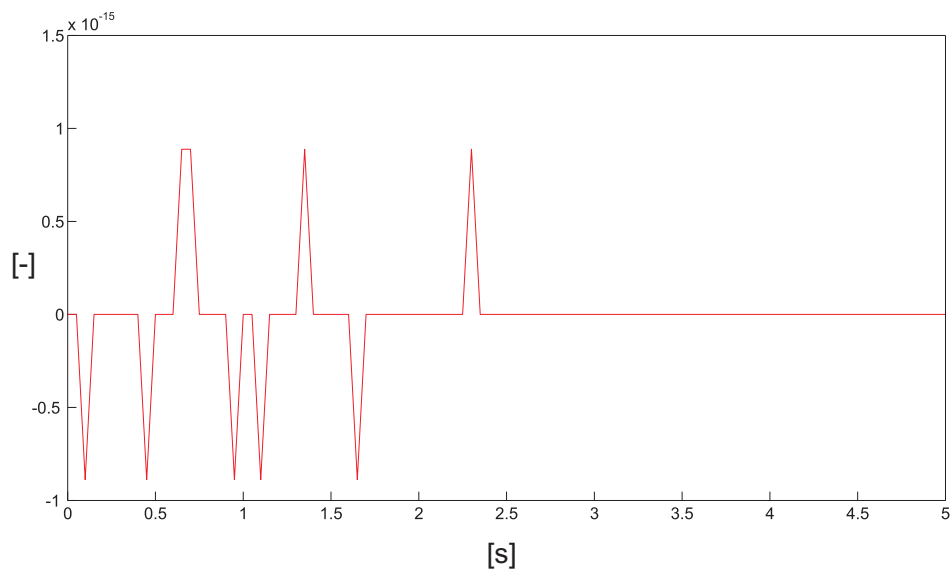
Az α és β tagok úgy viselkednek, mint a csillapítási és rugó merevségi tényezők, tehát ezek értékeivel lehet szabályozni az átírt kényszeregyenlet beállási karakterisztikáját. Ez szabályozástechnikai megfogalmazásban egy PD szabályzó, amit ha beírunk az elsőfajú Lagrange egyenletbe egy zárt szabályozási kört kapunk, amelyet a 2.26. ábra szemléltet a [10] alapján.

$$\begin{bmatrix} \mathbf{M} & \phi_q^T \\ \phi_q & 0 \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ -\dot{\phi}_q \cdot \mathbf{y} - 2 \cdot \alpha \cdot \dot{\varphi} - \beta^2 \cdot \varphi \end{bmatrix} \quad (2.41)$$



2.26. ábra: A baumgarte stabilizáció blokk diagrammja

Ha az így stabilizált rendszerrel figyeljük a kényszeregyenlet időbeni teljesülését (2.27. ábra), akkor láthatjuk, hogy ugyanazon beállítások során sokkal beljebb hozhatjuk a hibát, és a hiba pozitív és negatív csúcsokkal is rendelkezik ami a lengőrendszerként viselkedő stabilizálás következménye.



2.27. ábra: A stabilizált rendszerben a kényszer hibája

Mint említettük a rendszer stabilitási jellemzőit az α és β tagokkal lehet beállítani. Arra azonban, hogy ezen két paraméternek milyen numerikus értéket, vagy egyáltalán milyen lineáris kombinációját kell választani, hogy a kívánt stabilitás teljesüljön, de ne

változzon a rendszer mechanikája kevés ajánlást találni a szakirodalomban, 2011-ben jelent meg az első jelentősebb publikáció [10] a problémával kapcsolatban.

Vezessük be a (2.40) egyenletbe D relatív csillapítási tényezőt [6], hogy a rezgés-tanban tanult csillapodási görbékkel tehessük szemléletessé az egyes α, β kombináció által elérhető karakterisztikákat:

$$\ddot{\varphi} + 2 \cdot D \cdot \beta \cdot \dot{\varphi} + \beta^2 \cdot \varphi, \quad (2.42)$$

$$D = \frac{\alpha}{\beta}. \quad (2.43)$$

A D relatív csillapítási tényező függvényében három esetet tudunk megkülönböztetni:

1. $D < 1$, amit alulcsillapított lengő rendszernek hívunk, ekkor két komplex gyökkel rendelkezik a rendszer karakterisztikus egyenlete. Ez az $\alpha^2 - \beta^2 < 0$ kombinációval érhető el.
2. $D = 1$, ez a kritikus csillapítás esete amikor csak egy gyököt kapunk megoldásként, ekkor $\alpha^2 - \beta^2 = 0$
3. $D > 1$, ami túlcsillapított lengőrendszert eredményez két valós gyökkel, és ekkor $\alpha^2 - \beta^2 > 0$

Az megadott α, β kombinációkat a (2.40) rendszer karakterisztikus egyenletének gyökeiből is megkaphatjuk [10]. A karakterisztikus egyenlet a következő:

$$s^2 + 2 \cdot \alpha \cdot s + \beta^2 = 0, \quad (2.44)$$

ennek gyökei:

$$s = -\alpha \pm \sqrt{\alpha^2 - \beta^2}. \quad (2.45)$$

Belátható, hogy itt is három lehetséges eset kapunk eredményül a gyökjel alatti kifejezés viszonyának függvényében (2.45), melyek a relatív csillapításos analógiával azonos eredményre vezetnek.

Ezek után nézzük meg, hogy milyen módok állnak rendelkezésünkre, hogy ne csak beállási jellegre válasszunk helyes paraméterértékeket, hanem a stabilitásnak is eleget tegyünk.

Ehhez a legegyszerűbb mód, ha megvizsgáljuk a kényszeregyenlet Taylor sorba fej-tett alakját (az első három tagot nézve)[10]:

$$\varphi(t+h) = \varphi(t) + \dot{\varphi}(t) \cdot h + \ddot{\varphi}(t) \frac{h^2}{2}, \quad (2.46)$$

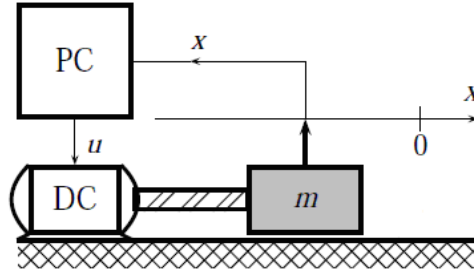
ahol h az időlépés. Feltételezve, hogy a $t+h$ -adik időpillanatban a kényszer teljesül az alábbi egyenletet írhatjuk fel a kényszerre ebben az időpillanatban:

$$\ddot{\varphi}(t) + \frac{2}{h} \cdot \dot{\varphi}(t) + \frac{2}{h^2} \cdot \varphi(t) = 0. \quad (2.47)$$

Ezek alapján $\alpha = \frac{1}{h}, \beta = \frac{\sqrt{2}}{h}$ -nek választható, mely az esetek nagy részében kielé-gíti a stabilitási és mechanikai követelményeinket. Kis időlépések esetében azonban a

csillapítás és a merevség számottevő növekedése merevvé teheti a kényszert, ami instabilitásokhoz vezethet. Emiatt érdemes lehet adaptív időlépések, vagy nagyon finom felbontás esetén másképp is megvizsgálni a rendszer stabilitást.

Már többször melítettük, hogy a Baumgarte stabilizáció felfogható egy egy szabadságfokú lengőrendszerként. Emiatt az átírt kényszer egyenletre analógiaként alkalmazhatunk egy digitális pozíciószabályozási modellt [6] [12].



2.28. ábra: DC motor PD szabályozási modellje

Ezen rendszer mozgásegyenlete és a Baumgarte stabilizációval átírt kényszeregyenlet (2.40) egyenértékűsége nyilvánvaló amennyiben a tömeget egységnyinek vesszük:

$$x''(t) + P \cdot x'(t) + D \cdot x(t) = 0. \quad (2.48)$$

A nem mintavételes rendszernél az egyetlen stabilitási kritérium, hogy P és D paraméterek értékei pozitívak legyenek (Routh-Hurwitz kritérium alapján)[6][12]. Azonban a mintavételes rendszerben a helyzet bonyolultabb. Írjuk át a rendszert mintavételessé a $[t_j, t_j + h]$ intervallumra, valamint dimenziótalanítsunk a h időlépéssel. Az előző időpillanatbeli sebesség, és pozíció is ismert, tehát az egyenlet az alábbi alakot ölti:

$$x(T)'' = -p \cdot x'(j-1) - d \cdot x(j-1), \quad (2.49)$$

$$p = h^2 \cdot P, \quad (2.50)$$

$$d = h \cdot D, \quad (2.51)$$

ahol $T = \frac{t}{h}$. Nevezzük a (2.49) egyenlet egyenlőségjel utáni felét a_j -nek, majd fejezzük ki a mintavételi adatokból a pozíciót és a sebességet is a valós T időpontban:

$$x(T)' = x'(j) + a_j(T - j), \quad (2.52)$$

$$x(T) = x(j) + x'(j) \cdot (T - j) + \frac{1}{2} \cdot a_j(T - j)^2. \quad (2.53)$$

Mátrixba rendezve a (2.49)-(2.53) egyenleteket a mintavételezett rendszer állapottermodelljét kapjuk egy 3 dimenziós mértani sor formájában:

$$\begin{bmatrix} x(j+1) \\ x'(j+1) \\ a_{j+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0.5 \\ 0 & 1 & 1 \\ -p & -d & 0 \end{bmatrix} \begin{bmatrix} x(j) \\ x'(j) \\ a_j \end{bmatrix}. \quad (2.54)$$

A 3×3 -as mátrix a rendszer tulajdonságait leíró $\mathbf{A}$ rendszermátrix. A sajátértékek kiszámításához a $\mu \mathbf{I} - \mathbf{A}$ mátrix determinánsát kifejtjük:

$$\mu^3 - 2\mu^2 + (1 + d + \frac{p}{2})\mu - d + \frac{p}{2} = 0, \quad (2.55)$$

majd elvégezzük rajta a Möbius transzformációt, hogy a $\mu_i < 1$ feltétel helyett a Routh-Hurwitz kritérium legyen alkalmazható:

$$\mu := \frac{\eta + 1}{\eta - 1}. \quad (2.56)$$

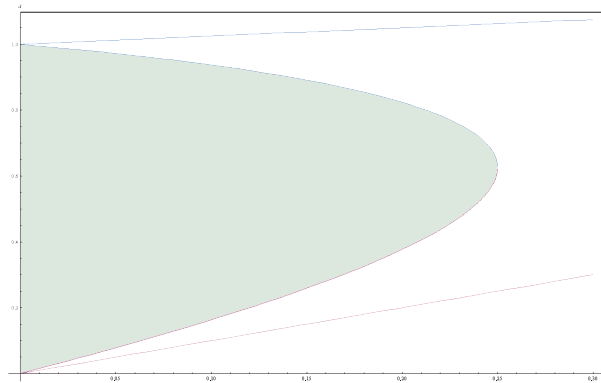
$(\eta - 1)$ -el való beszorzás, majd átrendezés után az alábbi karakterisztikus egyenletet kapjuk:

$$a_3 \cdot \eta^3 + a_2 \cdot \eta^2 + a_1 \cdot \eta + a_0 = 0. \quad (2.57)$$

Ezt mátrixban felírva:

$$\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 1 & -1 & -3 \\ 3 & -1 & -1 & 3 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1 + d + \frac{p}{2} \\ -d + \frac{p}{2} \end{bmatrix}. \quad (2.58)$$

Az a paramétereket Routh-Hurwitz vizsgálat alá vetjük, azaz mindegyikre kikötjük hogy pozitívak, valamint megvizsgáljuk az a paraméterekből összeállított második Hurwitz determinánst is [6]. Ezen vizsgálat eredményeképpen stabilitási térképet tudunk rajzolni:



2.29. ábra: A szabályozás stabilitási térképe

A besatírozott részen belül található a stabil p, d paraméterek, melyekből a (2.50), (2.51) egyenletek alapján meghatározhatjuk p és d -ből α és β értékeit a mintavételi idő h függvényében.

A modell tökéletesítése

Az eddig végrehajtott lépések alapján már rendelkezünk egy tetszőleges anyagi pontból álló geometriai kényszerekkel láncba alakított féregtesttel, mely a talajjal érintkezve képes pattani, tapadni, csúszni.

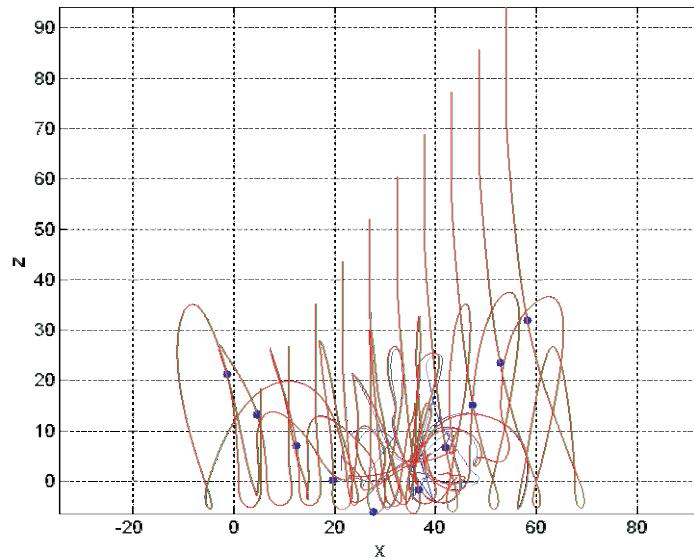
A tapadás és csúszás miatt azonban a Runge-Kutta eljárás által hívott, a mechanikai rendszert leíró függvényben szükségünk van az adott időpillanatbeli gyorsulásra is. Ez azonban a numerikus megoldók sajátossága miatt nem áll rendelkezésünkre, hiszen

ott mindig az előző pillanatbeli pozícióból és sebességből iteráljuk az adott pillanatbeli gyorsulásértékeket, melyet az előzőekben bemutatott elsőfajú Lagrange egyenlet eredményez, azonban mikor ezt számítjuk az általános erővektorba már az adott pillanatban érvényes talajerőket kell megadnunk. Azt viszont, hogy tapadás vagy csúszás történik-e az egyes talajjal érintkező anyagi pontoknál az adott pillanatban érvényes gyorsulásból tudjuk eldönteni. Programozástechnikailag ez úgy néz ki, hogy a féregtest minden egyes anyagi pontjára megvizsgáljuk, hogy érintkezik-e a talajjal, ha igen akkor a letárolt x irányú haladási irány alapján eldöntjük, hogy megcsúszásról, vagy tapadásról van-e szó (2.2.1 fejezet), majd az x és z irányú erőket felhasználva a (2.9), (2.10) egyenletek alapján kiszámítjuk a súrlódási erő értékét.

Azonban látható, hogy ez a gondolatmenet hibás, hiszen az itt használt erők nem tartalmazzák a λ kényszererőket. Ezért a következő eljárással finomítottuk a számítást:

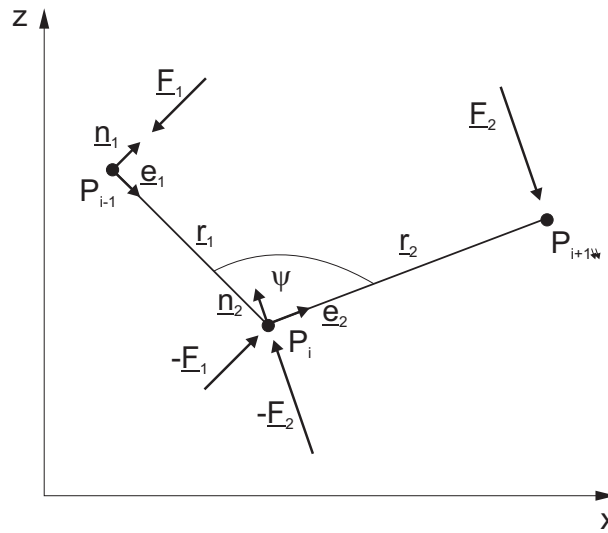
1. Az előző lépésből letárolt Lagrange multiplikátorokból (kényszererők nagysága) és a kényszerek általános kordinták szerinti deriváltjának szorzatából kiszámítjuk a kényszererőket, majd a talajerőket, végül az eredményeket belevéve az elsőfajú Lagrange egyenlet hipermátrixába kiszámítjuk a gyorsulások és a Lagrange multiplikátorok értékét.
2. A kapott eredményekből újra kiszámítjuk a talajerőket, amikből ismét a mozgásegyenlet segítségével gyorsulásokat számítunk.
3. Ezt az eljárást addig folytatjuk, amíg kellően pontos eredményeket nem kapunk. (Tesztek alapján egy iteráció után már nem érünk el számottevő változást a szimuláció eredményében)

A 2.30. ábrán kékkel látható a lambda nélküli számítás, zölddel az előző lépés lambda-jával számított, pirossal pedig az iterációs eljárással számított pálya képe:



2.30. ábra

Az előző képen (2.30. ábra) látható, hogy ugyan a féregmodell tömegpontjai, mostmár a geometriai kényszerek révén rendezett mozgást végeznek, azonban a felrajzolt pályákon látható, hogy a leejtés és pattogás után mint egy gyöngysor úgy gabalyodik össze. Ez a természetben található élőlényeknél nem így van hiszen testük belső rugalmasságokkal és csillapítókkal rendelkezik. Hogy realiztikusabb mozgást kapjunk a féregmodellbe csillapítást és rugalmasságot építettem be. Ezt úgy a legegyszerűbb kezelni, ha az egyes szelvények egymáshoz képesti szögelfordulását meghatároztuk és a csomópontokba torziós rugókat és csillapításokat építünk. Az ezekből párokká számítjuk át (természetes koordináták esetén a nyomatékok is erőként jelentkeznek az általános erő vektorában). Ezeket az erőket hozzáadjuk az általános erő vektorhoz. Erre azonban olyan számítási módszert kell írunk, mely a féreg bármely helyzetében képes pontonként szögértékeket az akutális és az előző szögértékből pedig sögsebességeket számolni, majd a kapott nyomatékokat természetes koordinátákkal megadott erővektorokká alakítani. Erre az alábbi számítási módszer alkalmazható [9], melynek értelmezését a 2.31. ábra segíti:



2.31. ábra: Szelvények által bezárt szög és a torziós rugóból származó erő

A számításokhoz szükségünk van a pontok közti helyvektorra és távolságra:

$$\mathbf{r}_1 = \begin{bmatrix} x_i - x_{i-1} \\ z_i - z_{i-1} \end{bmatrix}, \quad (2.59)$$

$$\mathbf{r}_2 = \begin{bmatrix} x_{i+1} - x_i \\ z_{i+1} - z_i \end{bmatrix}, \quad (2.60)$$

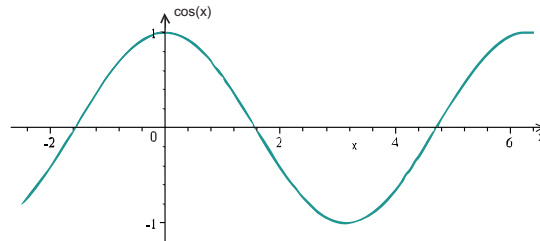
$$L_1 = \sqrt{r_{1x}^2 + r_{1z}^2}, \quad (2.61)$$

$$L_2 = \sqrt{r_{2x}^2 + r_{2z}^2}. \quad (2.62)$$

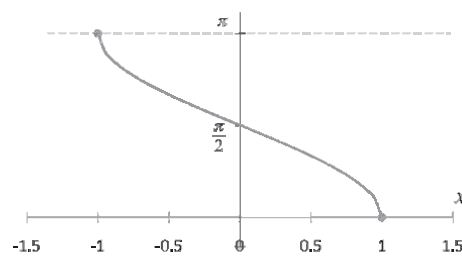
A közbezárt szög meghatározásához elvileg ez elég is kéne, hogy legyen ugyanis a skaláris szorzatból kényelmesen meghatározhatjuk a szög cosinusát, amiből vissza tudnánk számolni a szöget.

$$\mathbf{r}_1 \cdot \mathbf{r}_2 = L_1 \cdot L_2 \cos \psi, \quad (2.63)$$

Azonban a Matlab `acos()` függvénye csak az első periódusra $\psi \in [0, \pi]$ ad valós eredményeket ezen kívül komplex eredményeket kapunk, amik a számításainkhoz nem használhatók.



2.32. ábra: Cosinus függvény



2.33. ábra: Arcus Cosinus függvény

Ez azonban nem alkalmazható a modellben, mivel gyakran előfordulhatnak ettől eltérő szögértékek is. Kutakodva a függvénykészletben találtam az `atan2()` függvényt, mely egy természetes koordinátákkal megadott vektor x tengellyel bezárt szögét adja meg.

Tehát elő kell állítanunk egy olyan vektort, mely pontosan az $\mathbf{r}_1$ és $\mathbf{r}_2$ vektorok által bezárt szög nagyságával azonos szöget zár be az x tengellyel.

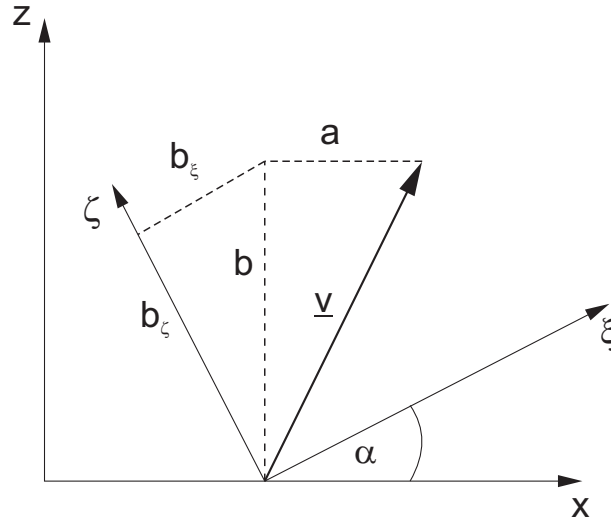
Ehhez előállítjuk a szelvények irányvektorát:

$$\mathbf{e}_1 = \frac{\mathbf{r}_1}{L_1}, \quad (2.64)$$

$$\mathbf{e}_2 = \frac{\mathbf{r}_2}{L_2}. \quad (2.65)$$

Pozitív irányú 90° -os forgatással előállítjuk az irányvektorokból $\mathbf{n}_1$ és $\mathbf{n}_2$ normálvektorokat.

Ahhoz, hogy tudjuk használni az `atan2()` függvényt az $\mathbf{r}_2$ vektort az $\mathbf{r}_1$ vektor által meghatározott irányú koordináta-rendszerben kell kifejezni. Ehhez a 2.34. ábra szerint fogunk összeállítani egy forgatási mátrixot.



2.34. ábra: Forgatási mátrix ábrája

A $\mathbf{v}$ vektort szeretnénk megkapni x, z koordináta rendszerben:

$$\mathbf{v}_{(x,z)} = \begin{bmatrix} a \\ b \end{bmatrix}. \quad (2.66)$$

Ami rendelkezésünkre áll az a $\mathbf{v}_{(\xi,\zeta)}$

$$\mathbf{v}_{(\xi,\zeta)} = \begin{bmatrix} a \cos \alpha + b \sin \alpha \\ -a \sin \alpha + b \cos \alpha \end{bmatrix}. \quad (2.67)$$

Ez mátrix alakba rendezve:

$$\mathbf{v}_{(\xi,\zeta)} = \begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}. \quad (2.68)$$

Amire ebből szükségünk van az tehát a 2×2 -es forgatási mátrix, melyet egyszerűen előállíthatunk, hiszen a ξ tengely egységnyi irány és normál vektora az alábbiak szerint néz ki:

$$\mathbf{e}_\xi = \begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix}, \quad (2.69)$$

$$\mathbf{n}_\xi = \begin{bmatrix} -\sin \alpha \\ \cos \alpha \end{bmatrix}. \quad (2.70)$$

Tehát $\mathbf{R}$ forgatási mátrix egyszerűen előállítható ha a fenti analógiát alkalmazzuk ($\mathbf{v}$ azonos $\mathbf{r}_2$ – vel ξ, ζ tengelyeket pedig $\mathbf{e}_1$ és $\mathbf{n}_1$ feszíti ki):

$$\mathbf{R} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{n}_1 \end{bmatrix}. \quad (2.71)$$

A forgatáshoz vesszük $\mathbf{R}^T$ és $\mathbf{r}_2$ szorzatát, majd a kapott eredményt beírjuk az atan(2) függvénybe, az így kapott szögértéket pedig eltároljuk.

Ez a módszer bonyolultnak tűnhet, hiszen egyszerűbben számíthatnánk a két szelvény x tengellyel bezárt szögeinek különbségéből a bezárt szöget, azonban ebben az

esetben két $\text{atan2}()$ függvényt kellett volna hívni, mely lassabb eljárás, mint a mátrixokkal való transzformáció.

A modell két szelvénye közötti rugómerevségből származó nyomaték a rugómerevség paraméteréből és a szögértékből egyszerűen számítható:

$$M_s = s \cdot \psi. \quad (2.72)$$

A csillapításhoz azonban elő kell állítanunk a szögsebességet is, melyet a letárolt szögértékek segítségével hátralépő differenciálási sémával állíthatunk elő:

$$\dot{\psi} = \frac{\psi(t) - \psi(t-1)}{h}. \quad (2.73)$$

A fenti formula (2.73) ugyan némileg rontja a Runge-Kutta eljárás rendjét, azonban ahhoz, hogy analitikusan tudjuk számítani a szelvények közti szögsebességet, szükségünk van az időben változó hosszúságú szelvények hosszúságának az idő szerinti deriváltjára. Az alakjukat időben változtató testek esetében azonban nem alkalmazhatjuk a merev testekre alkalmazott képleteinket a sebességszámításhoz [6]:

$$\mathbf{v}_B = \mathbf{v}_A + \omega \times \mathbf{r}_{AB} \quad (2.74)$$

Ugyanis a fenti (2.74) egyenlet az időfüggő hossz miatt kiegészül egy taggal, így az alábbi alakot ölti:

$$\mathbf{v}_B = \mathbf{v}_A + \omega \times \mathbf{r}_{AB}(t) + \frac{\partial \mathbf{r}_{AB}(t)}{\partial t}. \quad (2.75)$$

Ebből már ki tudjuk fejezni a szögsebességet, azonban ezt a számítási módszert a jelenlegi modellben nem alkalmaztuk, mivel a kapott eredmények kielégítőek a véges differencia módszerrel is. Azonban a jövőben megvalósuló 3 dimenzós modellben már ezen megfontolások alapján fogjuk számítani a szögsebességet, hiszen 3 dimenzióban a hagyományos geometriai összefüggések nem alkalmazhatóak, valamint feltételezhetően további előnye ennek a számításnak, hogy a baumgarte stabilizáció miatt keletkező pontatlanságokat is csökkentti ez a számítási módszer, hiszen $\frac{\partial \mathbf{r}(t)}{\partial t}$ -t ismerjük az $\mathbf{r}$ vektort pedig szintén előállíthatjuk az szelvényenként ismert hosszából és a szelvények irányvektorából, így az egyes csomópontok koordinátinak számításakor a rendszerbe kerülő numerikus hiba nem kerül bele a szögsebességbe ezáltal pontosabb eredményt kaphatunk.

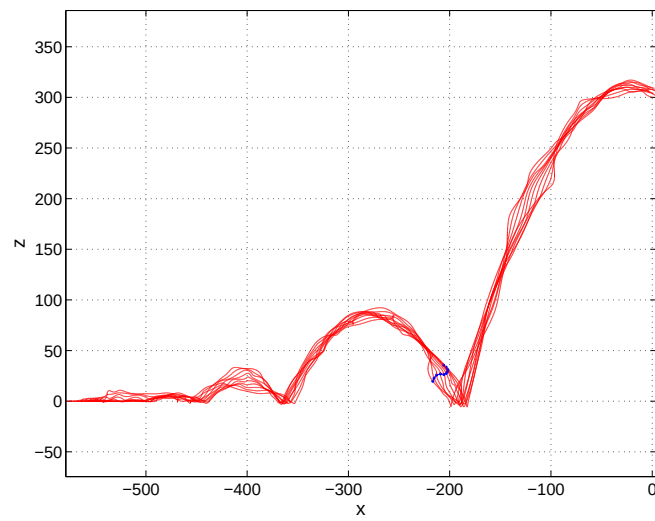
A véges differencia módszer alkalmazása után (2.73) kapott szögsebességből, már egyszerűen számíthatjuk a szelvények közti csillapításból származó nyomatékokat is:

$$M_k = k \cdot \dot{\psi}. \quad (2.76)$$

A két szélső pontra ható erőket egyszerűen megkaphatjuk, ha a nyomatékokat elosztjuk az L_1 és az L_2 hosszal majd megszorozzuk a normálvektorokkal. A középső pontra pedig a mind a két szélső erőt fordított előjellel összegezzük.

$$\mathbf{F}_1 = -\frac{M}{L_1} \cdot \mathbf{n}_1 \quad (2.77)$$

$$\mathbf{F}_2 = -\frac{M}{L_2} \cdot \mathbf{n}_2 \quad (2.78)$$



2.35. ábra: Adott kezdeti feltételekkel elhajított féregmodell

A 2.35. ábrán látható, hogy az így kapott modell már élethűen tudja azt modellezni, ha elhajítjuk a féregtestet.

2.3.2. Vezérlések megvalósítása

Az előző részben összeállított féregmodell olyan mint a filmes és számítógépes animációkban eséseknél ütközéseknél használt rongybaba (Rag-Doll) modell, mely egy olyan testrendszer, ami külső erők hatására jön mozgásba, és a modellezett lényt próbálja ebben az ütközési szituációban utánozni. Ez a modell tökéletesen megfelel ütközések, esések modellezéséhez, hiszen az ott modellezett élőlény saját maga által kifejtett erők elhanyagolhatók a külső behatás nagyságához képest. Azonban ahhoz, hogy a 1.1. bevezetőben kitűzött célt, nevezetesen a helyváltoztatás vizsgálatát megvalósítsuk mindenképpen szükségünk lesz arra, hogy a 2.1.2. részben definiált magától mozgó rendszert létrehozzuk. Ehhez szükséges, hogy a rendszert belső vezérlő erőkkel lássuk el.

Peristaltia

A peristaltia azaz az, hogy a féreg képes legyen az egyes szelvények hosszának megváltoztatására, mindenképpen érdekes lehetőség a haladás megvalósítására, hiszen az így haladó robot képes lenne a keresztmetszeténél alig nagyobb résekben való közlekedésre, mely számos felhasználási lehetőséggel kecsegtet. Hogy a modellünk erre alkalmas legyen nem kell mást tennünk, mint az anyagi pontok közt lévő szelvényeknek megengednünk, hogy változtassák a hosszukat, amit úgy érhetünk el hogy a kényszereket továbbfejlesztjük. Ahhoz, hogy haladást érjünk el a hosszváltozással, természetesen időben rendezett módon kell változtatnunk a szelvények hosszát. Tehát az eddig időben állandó kényszert (2.23) időben változtathatóvá kell tennünk:

$$\varphi_i(\mathbf{q}, t) = (x_i - x_{i+1})^2 + (x_{n+i} - x_{n+i+1})^2 - L(t)^2 = 0. \quad (2.79)$$

Így egy reonom, azaz időben változó geometriai kényszert kapunk tehát a modellünk holonom, reonom rendszerré válik. Ennek kezeléséhez két taggal kell bővítenünk a hipermátrixos egyenletünket (2.41), mely így az alábbi alakot fogja öltetni [9]:

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\phi}_q^T \\ \boldsymbol{\phi}_q & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ -\dot{\boldsymbol{\phi}}_q \cdot \dot{\mathbf{q}} - \frac{\partial^2 \varphi}{\partial t^2} - 2 \cdot \alpha \cdot (\boldsymbol{\phi}_q \cdot \dot{\mathbf{q}} + \frac{\partial \varphi}{\partial t}) - \beta^2 \cdot \varphi \end{bmatrix}. \quad (2.80)$$

Az egyes szelvények időbeni hosszváltozásához szükségünk van egy jól kézbentartható jelre. Ehhez egy szinuszos hullámot választottam, melynek időbeni és pontonkénti lefutását a szimuláció elején letárolom egy kétdimenziós mátrixban, így minden időpillanatban rendelkezésre fog állni az összes szelvény hossza:

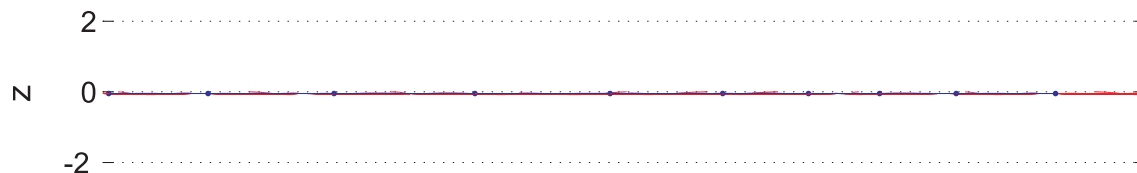
$$\mathbf{L}_i(t) = \text{ampl} \cdot \sin(\text{omega} \cdot t + \text{theta} \cdot i) + \text{const}. \quad (2.81)$$

Az így generált szinuszos jel három független paraméterrel rendelkezik:

1. ampl: a jel amplitudója, mely meghatározza a hosszváltozás mértékét.
2. omega: a jel körfrekvenciája, mely az időbeni lefutás sebességét határozza meg.
3. theta: a fáziseltolás értéke, mely a hullámok hosszával áll kapcsolatban.

Ezzel a három paraméterrel tág határok között változtathatjuk a peristaltia jellegét.

Azonban azt tapasztaljuk, ahogy az a 2.36. ábrán kirajzolt pályán is látható, hogy a modell nem halad. Ennek oka a 2.2.1 sűrűlódásról szóló részben leírt a természet által alkalmazott trükk, az irányfüggő sűrűlódási együttható kihagyása a modellből. Emiatt mivel jelenleg minden pont a talajon van a keletkező sűrűlódási erők kiegyenlítik egymást és a féreg súlypontjának pozíciója nem változik.

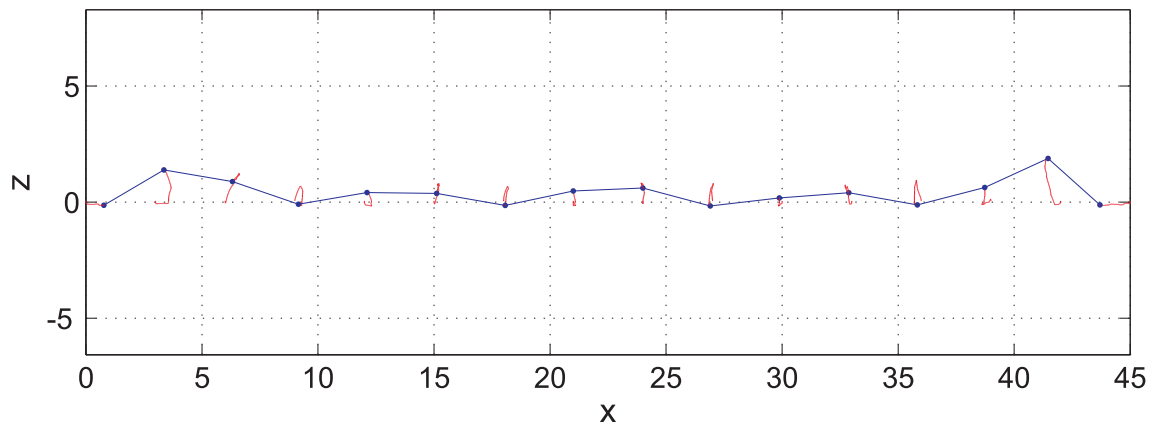


2.36. ábra: Peristaltíával mozgó modell

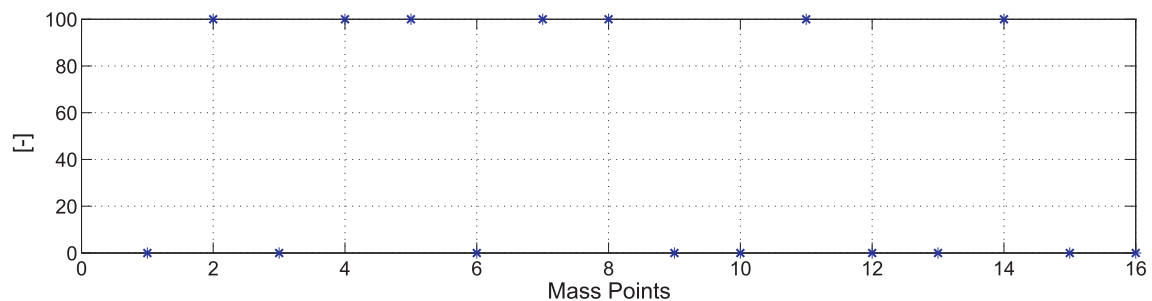
Az irányfüggő sűrűlódási együtthatót továbbra is kihagyom a modellből, és inkább úgy próbálok meg haladást elérni a továbbiakban, hogy a modellbe épített plussz vezérlésekkel képes legyen elemelni az egyes pontokat a talajtól, így a keletkező sűrűlódási erők már képesek lehetnek előre mozgatni a féreg testét. Erre a következőkben két modellt ismertetnék, melyek működését ki is dolgoztam a programkódban.

Nyomatékvezérlés

Ennél a módszernél a modell csomópontjaiba rotációs szervomotorokat képzelünk, melyeket vezérlő jelekkel irányíthatjuk. Megfelelően megválasztott jellel képesek vagyunk a modell periodikus hullámvázát előidézni. Ehhez a szinusz jelnél egyszerűbben értelmezhető négyszögjelet választottam, hiszen a szinuszjelnél a motorokon lévő nyomatékok nehezen értelmezhetők, míg a négyszögjel esetén csak a "van nyomaték" vagy "nincs nyomaték" állapotokat kell megkülönböztetnünk. A szervomotorokat teljesen egyszerűen olyan eszközöknek képzeljük el, melyek jel hatására időzés nélkül képesek az adott nyomatékot kifejteni, tehát a motor csillapításmentes és rugalmatlan. Ez azonban nem okoz irreális rángatózást a modellünkben, hiszen a csomópontokba már építettünk csillapítást és torziós merevséget. Vehetjük tehát úgy is, hogy jelen modellnél a három paraméter a négyszögjel amplitudója, mely meghatározza a motor által kifejtett nyomatékot, és a csillapítás valamint a torziós merevség együtt modellez egy szabályozható felfutási karakterisztikával és nyomatékkal rendelkező szervomotort. A négyszögjel akár csak a perisztaltia okozó jel három paraméterrel szabályozható. A rákapcsolt nyomaték a 2.37. ábrán látható, ahol megfigyelhető, hogy az egyes pontok már el tudnak emelkedni a talajtól.



2.37. ábra: Nyomatékvezérléses féreg modell



2.38. ábra: Nyomaték vezérlő jele

A 2.38. ábrán pedig látható az adott időpillanatban a féreg tömegpontjaiban lévő aktuátorokra kapcsolt jel értéke. Így már képesek vagyunk az egyes tömegpontokat megemelni, azonban a próbálkozások azt mutatták, hogy ezzel a négyszögjeles vezérléssel nehéz rendezett mozgásra bírni a modellt.

Műizmok alkalmazása

A jobban kézbentartható vezérlés érdekében ismét egy a természetből vett ötletet alkalmaztam. Az izmok úgy működnek, hogy mikor megfeszülnek nagy merevséget képvisznek mikor elernyednek minimálisat ezen belül pedig széles határok között változhat a merevség. Ezt a logikát követve felfoghatjuk a tömegpontokba épített torziós rugókat úgy mint izmok, melyek képesek megfeszülni és elernyedni, így a rugó egyensúlyi helyzetét eltolni. Ezt például pneumatikus műizmokkal lehetne megvalósítani. Ennél a módszernél nincs más dolgunk mint definiálni egy olyan vezérlő függvényt, mely minden időpillanatban szolgáltat számunkra egy előfeszítettségi értéket. Mivel minél folytonosabb mozgást szeretnénk megvalósítani ezért ismét a (2.81) egyenletben megfogalmazott szinusz jelet alkalmazzuk valamint, hogy a rugó feszítetlen helyzetének eltolása megvalósulhasson a (2.72) egyenletet az alábbiak szerint módosítjuk:

$$M_s = s \cdot (\psi - \psi_0). \quad (2.82)$$

Az így kapott vezérlési móddal jól láthatóan hatékony hernyószerű haladás valósítható meg:



2.39. ábra: Műizmom modellel mozgatott féreg

A fent bemutatott három különböző vezérlési mód lehetőséget ad az optimális helyváltoztatási stratégia tanulmányozására. A dolgozat következő főfejezete a vezérlés optimalizálásával foglalkozik.

3. fejezet

Optimalizálás

A 1.1. bevezetőben megfogalmazott cél a féregmodell minél hatékonyabb helyváltoztatásra bírása. A modellalkotás során láthattuk, hogy a modell számos paraméterrel bír, melyek nagyban befolyásolják a helyváltoztatás jellegét és minőségét. Ezek közül azok, melyek közvetlenül a féregmodell és nem a környezet vagy a szimuláció paraméterei:

1. A modell elemszám
2. Az anyagi pontok tömege
3. Az anyagi pontokba elhelyezett toziós rugók merevsége (s_{torz})
4. Az anyagi pontokba helyezett csillapítások mértéke (k_{torz})
5. A peristaltia amplitudója
6. A peristaltia frekvenciája
7. A peristaltia fázis eltolása
8. A rotációs szervó motor vezérlőjelének amplitudója
9. Az előbbi frekvenciája
10. és fázis eltolása
11. A műizomra ráadott jel amplitudója ($ampl0$)
12. frekvenciája (ω_0)
13. fázistolása (θ_0)

Látható, hogy a modell 13 paraméterrel rendelkezik, melyek ráadásul nem függetlenek egymástól emiatt véletlenszerű próbálkozással valószínűleg nem találjuk meg az optimális mozgást. Ezért speciális optimalizáló eljárásokat kell alkalmazni.

Azonban a 13 nem független paraméter még így is sok figyelembe véve, hogy egy 15 másodperces szimuláció majd fél perc alatt fut le egy átlagos teljesítményű számítógépen, ezért egyszerűsítésekhez folyamodunk:

1. a mozgást nagyban befolyásoló perisztaltit kikapcsoljuk (a vezérlő jel amplitudóját nullára állítjuk)

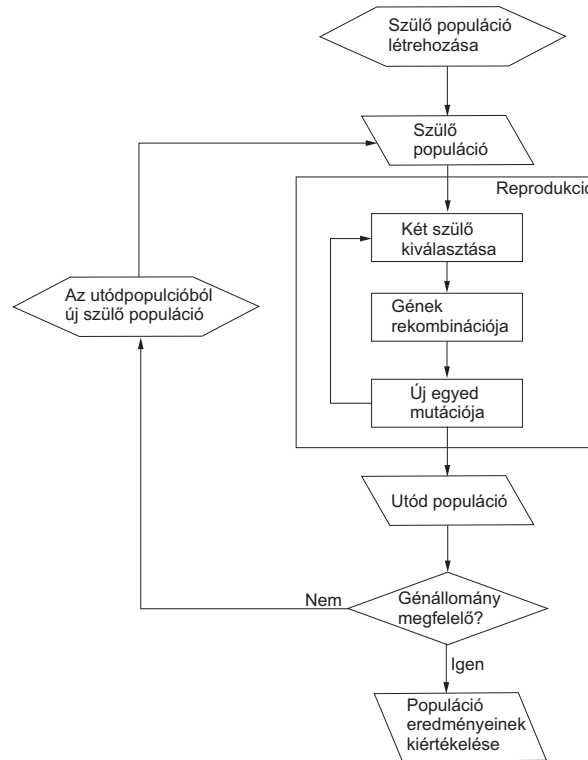
2. jelen munkában nem foglalkozunk a nyomatékkaal történő vezérléssel sem (ezen jel amplitudóját is nullának vesszük)

Tehát egyedül a várhatóan jó eredményeket hozó műizmos mozgatóst optimalizáljuk. Ezzel az egyszerűsítéssel 6-tal csökkentettük a paraméterek számát. Ezenfelül ha rögzítjük a tömeget egy nagyjából reálisnak mondható mértékre, mellyel valószínűleg már lehetne ilyen konstrukciót építeni (legyen ez mondjuk 2 kg csomópontonként) és az elemszámot is egy nem túl rövid, de nem is hosszú méretre (ez legyen 10db) akkor újabb két paramétert húzhatunk le a listánkról, így már csak 5 paramétert kell optimalizálni mely lényegesen kezelhetőbb, és ehhez már elengedő számítási kapacitással rendelkezünk. Egy ilyen optimalizálás során a 15 másodpercesre állított szimulációt akár több ezerszer is le kell futtatnunk, hogy használható eredményeket kapjunk, az újabb paraméterek vizsgálata pedig exponenciálisan növeli a szükséges futtatási időt. Még az optimalizálás során használt 4 magos jelenleg csúcskategóriás számítógép is 20 másodperc alatt futtatott le egyetlen szimulációt, emiatt az optimalizálás során a szűk keresztmetszet nyilvánvalóan az idő.

3.1. Genetikus algoritmus

A mai modern műszaki problémák gyakran kiterjedt problémaköröket fednek le, és számos olyan jelenleg is nehezen kezelhető problémát tartalmaznak, hogy a hagyományos matematikai minimum és maximumkereső problémákkal (pl. Newton-Rhapon iteráció, Ritz-módszer) nem lehet eredményeket elérni. A számítási kapacitások exponenciális növekedése viszont lehetővé tette, hogy a mérnökök olyan módszereket dolgozzanak ki, melyek bonyolult matematika nélkül iterációk, próbálgatások és statisztikai megfontolások révén képesek optimumkeresésre. Az egyik ilyen ma már széles körben alkalmazott módszer a genetikus algoritmus, mely akárcsak a féregmodell, szintén a természetből merített ötlet alapján készült.

A módszer alapötlete az evolúció, mely úgy működik, hogy adott populáció legsikeresebb egyedei tudnak csak párosodni, ezáltal csak a legjobb gének rekombinálódnak, így a populációban az idő előrehaladtával a génállomány és ezáltal a túléléshez szükséges képességek egyre javulnak, míg eléri azt a pontot hogy kialakulnak a legjobb lehetséges egyedek. A módszer alkalmazását a féregmodellen keresztül fogom ismertetni. A módszer működését a 3.1. ábra foglalja össze.



3.1. ábra: A genetikus algoritmus folyamatábrája

3.1.1. A populáció létrehozása

Az előzőekben kifejtettek alapján 5 paraméterét szeretnénk a férgünknek optimalizálni. Ez lesz az egyedek 5 génje mely meghatározza a túléléshez szükséges tulajdonságot, mely most a minél hatékonyabb helyváltoztatás lesz. Tapasztalatom alapján, melyek a 2009-es Csató Bertalannal közösen írt TDK munkánk [13] alapján írt genetikus algoritmuson alapszik, 5 gén optimalizálásához nagy, több száz egyedből álló populációra van szükség. A kezdeti populáció egyedeinek génjeit véletlenszerűen generáljuk adott határokon belül, melyeket olyan közökre választunk, ahol az optimumot sejtjük, vagy ami jó eséllyel még a fizikai valóság határain belül van.

```

s_torz_min=40;
s_torz_max=400;
k_torz_min=40;
k_torz_max=10;
ampl0_min=0.5;
ampl0_max=700;
omega0_min=0.1;
omega0_max=10;
theta0_min=0.1;
theta0_max=10;
  
```

A merevségek határának megválasztása rugókatalógusok [14] alapján történt, ugyanis ezek azok a tartományok, amik még gyárthatóak nem túl nagy méretben és megfelelő súllyal. A vezérlőjel paramétereinek tartományát pedig a választott merevség

és csillapítás értékekhez megfelelően határoltuk be.

A fenti határok között véletlenszerűen határozzuk meg az első populáció génjeinek értékét:

```
for i = 1:1:NEgyed
    Gen(i,2) = s_torz_min+(s_torz_max-s_torz_min)*rand(1);
    Gen(i,3) = k_torz_min+(k_torz_max-k_torz_min)*rand(1);
    Gen(i,4) = ampl0_min+(ampl0_max-ampl0_min)*rand(1);
    Gen(i,5) = omega0_min+(omega0_max-omega0_min)*rand(1);
    Gen(i,6) = theta0_min+(theta0_max-theta0_min)*rand(1);
end
```

3.1.2. Az egyedek osztályozása

Míg a természetben a természetes szelekció, úgy a genetikus algoritmusban egy osztályozó függvény vagy más néven célfüggvény (fitness-function) segítségével döntjük el melyek a legjobb egyedek. Ehhez először minden egyes egyeden le kell futtatnunk a féregmodell szimulációját (a gének által meghatározott paraméterekkel), majd a féreg szimulációja során tömegpontonként, és minden időpillanatban egyesével letárolt koordinátákból meg kell határoznunk a mozgás minőségét. Erre számos megfontolás alapján lehetne osztályozó függvényt írni, azonban a gyors futás érdekében egy meglehetősen egyszerű függvényt alkalmazunk, mely a kezdeti időpillanat tömegpontjainak x koordinátáit és a végső időpont x koordinátáit összegzi, elosztja az elemszámmal, majd kivonja a két időpontban kapott értéket egymásból:

$$\mathbf{Fit} = \left| \frac{\sum_{i=1}^n x_i^{end}}{n} - \frac{\sum_{i=1}^n x_i^{start}}{n} \right|. \quad (3.1)$$

Így minden egyedre kapunk egy skalár számot, mely a tömegközéppont időbeni elmozdulásának mértékét adja. Ez alapján sorba kell rendeznünk az egyedeket, hogy lássuk a populáció mely egyedei jók, melyek rosszabbak.

Sorbarendező algoritmusok

Mivel a nagy létszámú populáció miatt a futási idő várhatóan nagy lesz, illetve a nagy egyedszám miatt a sorbarendezéskor nagy mennyiségű adathalmazzal kell dolgoznunk ezért elengedhetetlen, hogy megvizsgáljuk a sorbarendezés hatékonyságát. A legáltalánosabb sorbarendező algoritmus a buborék módszer, melynek lényege, hogy a sorbarendezendő tömbbön annak méretével azonos mennyiségű iterációt hajt végre, melyben ha talál két olyan szomszédos elemet melyek közül a sorban előrébb lévő kisebb mint a sorban utána következő (jelenleg maximumot keresünk) akkor megcseréli a két elemet.

```
for i = 1:1:n % n-szer kell megcsinalni
    for j = 1:1:n-1 % vegigmegy a tombon
        if B(j,1) < B(j+1,1) %elemcsere
            temp = B(j,1);
            B(j,1) = B(j+1,1);
            B(j+1,1) = temp;
```

```

        end
    end
end

```

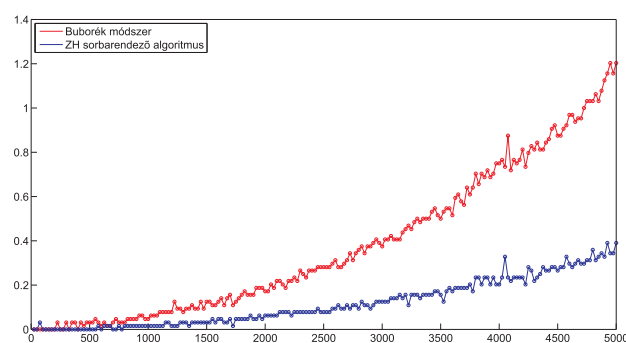
Ez érezhetően nem a leghatékonyabb módszer ezért konzulensemmel kidolgoztunk egy általunk "zh sorbarendező algoritmus"-nak elnevezett módszert. Ennek lényege, hogy létrehozunk egy ugyanakkora méretű tömböt mint amit sorbarendezzünk és ebbe olyan elv szerint rakjuk bele az elemeket, hogy ha találunk olyat, ami a sorbarendeztettek közé illik, akkor egyszerűen eltoljuk az elemeket és annak helyére berakjuk. Így mindig csak a már sorbarendeztett elemeken kell végighaladnunk nem pedig a teljes tömbön. A rendezőalgoritmus kódja:

```

C = zeros(n,1);
C(1,1) = A(1,1);
c = 1;
for i = 2:1:n
    C(c+1,1) = A(i,1);
    for j = 1:1:c
        if A(i,1) > C(j,1)
            for k = c:-1:j
                C(k+1,1) = C(k,1);
            end
            C(j) = A(i,1);
            break
        end
    end
    c=c+1;
end

```

A két rendezőeljárás hatékonyságának összehasonlítása a 3.2. ábrán látható.



3.2. ábra

3.1.3. Gének keresztezése és mutációja

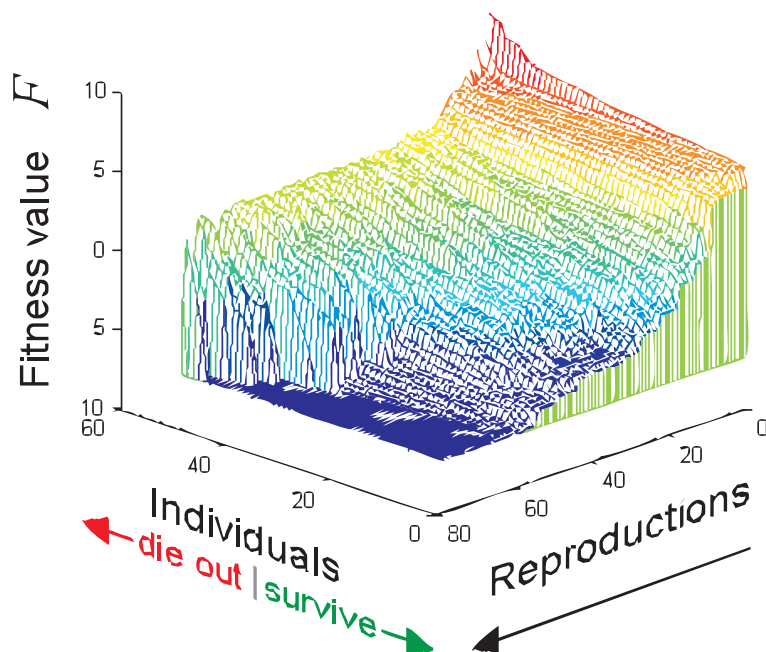
Miután sorbarendeztük a populációt kényelmesen ki tudjuk választani azokat az egyedeket melyek génjeit rekombinálni szeretnénk. Erre a szakirodalomban többféle ajánlás

is található, most azonban egy egyszerű módszert választottam, mely elkülöníti a felső 50%-ot majd közülük véletlenszerűen kiválaszt mindig két egyedet és ezen egyedek génjeit egy nulla és egy közé eső randomgenerált szám alapján súlyozott átlaggal számítva keresztezi. Az így kapott egyedekkel feltöltjük a populáció alsó 50%-át.

Az iteráció során azonban gyakran előfordul olyan eset, hogy úgynevezett szuper egyedeket kapunk, melyek nem valós optimumok, de olyan helyet foglalnak el az adott populációban, hogy a saját génjeiket örökítik tovább, míg olyan egyedek esnek ki a szelekció folyamán ezek rovására, melyek hasznos lehetőségeket tartogattak volna, így a megoldó rossz helyre konvergál.

Ennek kiküszöbölésére ismét a természet nyújt megoldási módszert a mutáció révén. A mutáció megvalósítására is számos lehetőség adódik, itt most csak a főleg optimalizálása során implementált módszert ismertetem, melynek lényege, hogy a mutációs paraméterekkel szabályozhatjuk, hogy a populáció mekkora százaléka mutálódjon. Jelen algoritmusban ez a százalékos arány az iterációk során lineárisan nullára csökken, mivel mutáció tulajdonképpen keregélést jelent a keresett paraméterek (gén) terében, és az iteráció végén már nem keresgélnek, hanem az eredményt minél inkább pontosítani szeretnénk. Ezen felül megadhatjuk még a mutáció valószínűségét, és azt is, hogy mekkora százalékban mutálódhatnak a gének. Ez sajnos három újabb paramétert hoz az optimalizáció algoritmusába, azonban mutáció nélkül a genetikai algoritmusok általában rossz eredményeket adnak.

A 3.3. ábrán egy viszonylag jó konvergencia esetén látható a célfüggvény logaritmusának az alakulása az idő, vagyis az újabb és újabb generációk függvényében (reproductions), és a populációk sorbarendezett egyedeinek függvényében (individuals). Látható, hogy a kezdetben minden egyedhez egy magas célfüggvény érték tartozik, majd az idő, pontosabban a generációk előrehaladtával megjelennek olyan egyedek, amelyek célfüggvény értéke igen kicsi, vagyis ők jól megfelelnek az elérni kívánt célnak, tehát konvergencia figyelhető meg. A jó egyedek mellett pedig továbbra is mindig jelen vannak a nagy célfüggvény értékkel rendelkező egyedek, ezek azok, amelyeknek a génjei erősen mutálódtak.



3.3. ábra: Példa a fitness value-k alakulására (nem a féregmodell futtatásából)

3.1.4. A genetikus algoritmus alkalmazhatósága

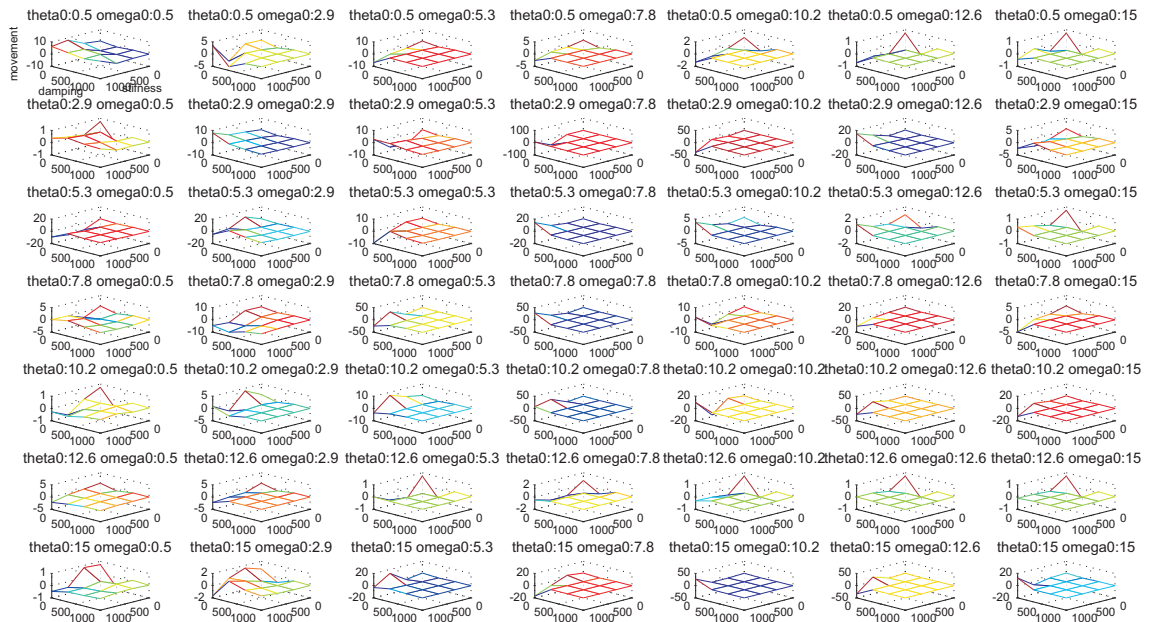
A gének száma nemcsak a jó konvergenciához szükséges populáció méretét befolyásolja, hanem a futási időt is, hiszen kevesebb változóval hamarabb találunk optimális kombinációt. Továbbá a három újabb paraméter (mutációs paraméterek) értékét is optimálisan kell megválasztanunk, melyre csak hozzávetőleges ajánlásokat találni, hiszen problémánként eltérőek a populációk, melyeknek teljesen más konvergálási tulajdonságaik lehetnek. A nagy futási idő miatt jelen munka írásakor még nem rendelkezem eredményekkel a genetikus algoritmusból, mivel túl sok idő lett volna megválasztani a gének helyes érték határait, a populáció szükséges méretét, a szükséges iterációs számot, valamint beállítani a mutációs paramétereket. A módszer nagy hátránya, hogy csak a futás végén tudjuk meg hogy mi történt a rendszerben (rajzoltathatnánk reziduumokat, de ez nagyban lassítaná a futást). Az algoritmus gyorsítására lehetne kísérleteket tenni, például a gének megváltozása és a kialakult hatás függvényében, vagy a gének egymásrahatásának függvényében írt szelektáló és rekombináló függvények írásával, azonban az így nyert futási sebesség kérdéses, a kapott eredményekben pedig további bizonytalanságokat okoz. Emiatt más, gyorsabban eredményre vezető eljárást próbáltam alkalmazni.

3.2. A pásztázás módszere

Bár elsőre kissé merész ötletnek tűnhet mégis a leggyorsabban talán a pásztázás módszerével lehet eredményeket felmutatni az optimalizálás során. Ennek a lényege, hogy paramétereinket adott tartományokon egy általunk megválasztott tetszőlegesen finom felbontással felosztjuk, majd egy egymásba ágyazott iterációval minden lehetséges kombinációt kipróbálunk. Ekkora méretű rendszerrel ez nem tűnhet túl jó módszernek,

hiszen paramétereinket tág határok között változtathatjuk, és öt paraméter adott határonokon belüli 10 elemre történő felosztása esetén 10^5 szimulációra van szükség melyet megszorozva a szükséges 20 másodperces futási idővel hihetetlenül hosszú optimalizációs idővel (23 nap) kell számolnunk. Ehhez hozzávéve azt a tényt, hogy például a merevség optimumát akár 0 és 1000 között is feltételezhetjük, tehát a 10 elemre történő felosztásnál is jó esetben csak tendenciákat tudunk megfigyelni optimumról aligha beszélhetünk. Azonban pont az egyes változások függvényében kirajzolódó tendenciák megfigyelhetősége ennek a módszernek a nagy erőssége. Ugyanis ha kisebb felbontásokkal vizsgáljuk a paramétertartományokat akkor a futási idő exponenciálisan csökken, viszont a megadott felbontásban szereplő minden kombinációra kapunk eredményeket, melyeket grafikonokon ábrázolva különböző tendenciákat figyelhetünk meg és ezek alapján fokozatosan szűkíthetjük a lehetséges értékek tartományát így közelítve a kívánt optimum felé.

Öt paraméternél azonban már nehézkes a grafikonon való ábrázolás. Ennek áthidalására az amplitudó változást mindig új grafikonon ábrázoltam, így a grafikonjaim már "csak" 4 dimenziósak, ami már elég jól analizálható. Azért az amplitudót választottam az új grafikonok rajzolási feltételének, mert a próbálkozások alapján ez volt az a paraméter ami leginkább független volt a többitől, a másik 4 pedig párokban függ egymástól. Egy grafikon úgy épül fel, hogy az x tengely mentén a kis grafikonokon folyamatosan változik az omega0 paraméter, azaz a jel frekvenciája, az y tengely irányában pedig a fentről lefelé növekszik a theta0 paraméter, mely a hullám szélessége. A kis belső grafikonok x tengelyén a merevség, y tengelyén pedig a csillapítás található. Megjegyzendő, hogy az előzőekben ismertetett (3.1) egyenletet abszolút érték nélkül használta, hogy azt is meg tudjuk figyelni, hogy a milyen paramétereknél merre indul el a modell. A kis grafikonoknak kétféle megjelenítési módjával próbálkoztam. Az egyik a Matlab mesh() függvénye:

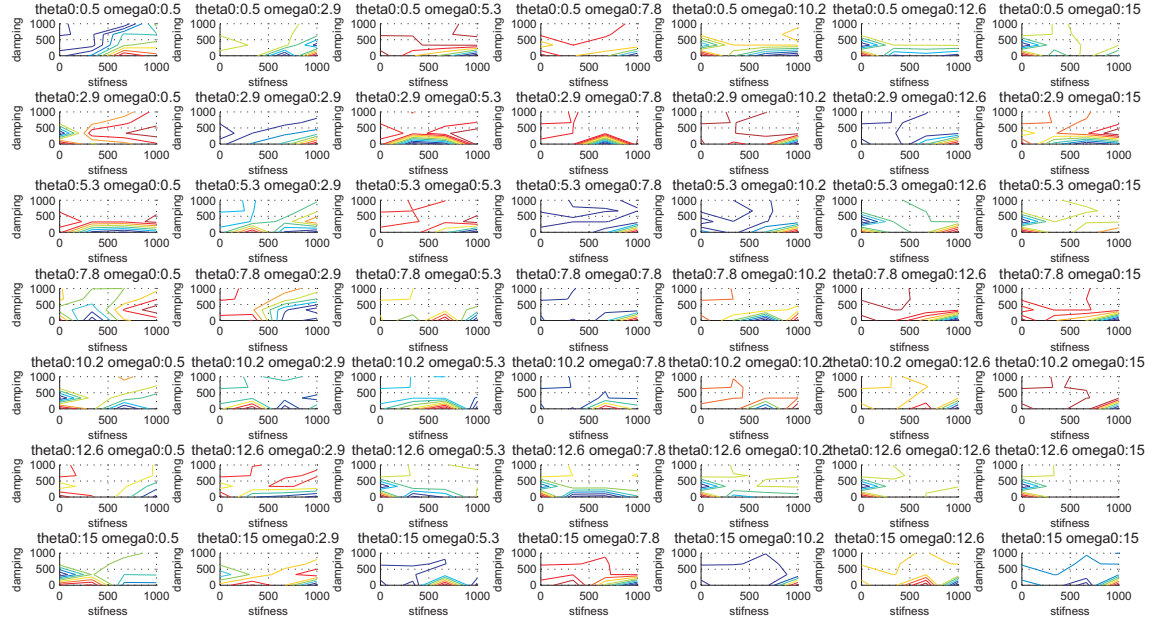


3.4. ábra: 4D paramétertér

Ennek nagy előnye, hogy matlab környezetben a kis grafikonok egyenként forgat-

hatók, és azt is látjuk, hogy mekkora értéke van a "fitness function"-nek (3.1) így az egyes kombinációk jól megfigyelhetők.

A másik ábrázolási mód a contour plot:



3.5. ábra: 4D paramétertér

Ennek előnye, hogy jobban megfigyelhetőek rajta a változások okozta tendenciák.

Ebből a két grafikonfajtából már az első futás során sikerült következtetéseket levonni. Ekkor meglehetősen tág paraméterhatárokat választottam melyeket egyenként 4 részre osztottam:

- $s_{torz} \in [0, 1000]$
- $k_{torz} \in [0, 1000]$
- $ampl0 \in [0.5, 2]$
- $theta0 \in [0.5, 15]$
- $omega0 \in [0.5, 15]$

Az így indított szimuláció nagyjából egy nap alatt futott le. Az eredményeken azonban szembetűnő, hogy kis csillapításoknál és nagy merevségeknél kapunk jó értékeket. Kigyűjtve a jónak látszó kombinációkat és lefuttatva animációval a szimulációt azt is megfigyelhetjük, hogy a nagy merevség és nagy amplitudó párosítások esetén a féreg már összetekeredve gurul, pattan, rendezetlen mozgást végez. Ezek a kombinációk tehát csupán matematikai optimumok, mérnöki nem adnak helyes eredményeket. A szinuszos jel hossz mentén mért terjedési sebességénél ($omega0$) is megfigyelhető, hogy a túl gyors jel a féreg rángatózását idézi elő. A fázistolás ($theta0$) pedig azt befolyásolja, hogy egyszerre hány amplitudo csúcs található a modellen. Ebből is a kisebb értékek tűnnek jobbnak. A nagy merevségeknél a modell jobban halad ugyan mint a kicsiknél,

azonban ezt konstrukciós megfontolások alapján lecsökkentettem, és a későbbiekben kapott optimumnál ugyanolyan eredményesen halad modell, mint a nagy merevségnél.

Ezen megfigyelések alapján még két futásból, ahol már sokkal kisebb tartományokon belül kereshettük az optimumokat találtam egy lehetséges optimumot:

- $s_{torz} \approx 500$
- $k_{torz} \approx 50$
- $ampl \approx 1.3$
- $theta \approx 5$
- $omega \approx 2$

Azért lehetséges optimum, mert a paraméterek egymástól nem teljesen függetlenek, továbbá a hosszváltozás, vagy a tömegváltozás esetén teljesen más optimumokat kaphatunk, így ez csak ennél a beállított féreg konstrukciónál kiválasztott optimum. Látható, hogy ezzel az eljárással viszonylag hamar optimumot kapunk. Ezután megpróbáltam változtatni a többi paramétert ezen optimumnál, hogy lássam milyen hatása van. A hossz növelése egyértelműen regulárisabbá tette a féreg mozgását, egy bizonyos elemszám felett azonban már nem okozott változást, az elemszám csökkentése viszont egyértelműen rontja a haladási tulajdonságokat (5 az a minimális elemszám ahol még megmozdul). Az optimális elemszám 20 körül van a futások alapján. A tömeg növelése egyértelműen lassította a rendszert a túl kicsi tömegeket viszont konstrukciós megfontolásokból nem vettem figyelembe. Látható tehát, hogy a kapott optimum a tömegre és az elemszáma reagál ugyan de a féreg nem veszti el a haladási képességét, ezért konstrukciós kialakításokat is figyelembe véve ez egy jó optimumközeli állapot lehet.

Azonban ahhoz, hogy valódi optimumot találjunk mindenképpen érdemes tovább finomítani az optimumkeresést, ezt a következő fejezet tárgyalja.

3.3. Feltételes optimum keresés

A Matlab Optimization toolboxa rendelkezik beépített optimalizáló eljárásokkal is. Azonban ezek az eljárások matematikai optimumkereső eljárások, emiatt közvetlenül nem tudjuk használni őket, hiszen a modellt mérnöki optimumra szeretnénk optimalizálni. Ezért az olyan eseteket, ahol a rendszer pattog, ugrál esetleg összegömbölyödve gurul vagy irreálisan nagy vezérlőnyomatéokra van szükség nem tekintünk optimumnak. Hiába hatékonyabb haladási módok, egy férget utánozó robot valószínűleg nem lenne képes ilyenfajta helyváltoztatásra. Emiatt feltételes minimumkeresést tudunk alkalmazni, azaz az adott határokon belül keressük az optimális haladást. A matlab Optimization toolbox minimum (pl.:fminsearch) és zérus (pl.:fsolve) kereső eljárásokkal rendelkezik. Mindkét függvény matemaikai optimumot keres, ezért nekünk valahogy le kell korlátoznunk a keresési tartományukat. Ezt az fsolve() esetén úgy valósíthatjuk meg, hogy nem a (3.1) egyenlet eredményét vizsgáljuk, hanem a kapott értéket kivonjuk egy számból, mely a mérnöki optimum határa. A függvény ugyan zérus kereső eljárás azonban ha nem talál zérus helyet akkor a zérushoz legközelebbi állapothoz tartozó paraméter beállításokat fogja visszaadni eredményül. Az fminsearch eljárás pedig

úgy használható, hogy a (3.1) egyenlet negáltját vesszük így keresünk minimumot, és a feltételezett mérnöki optimum határánál kilépünk az eljárásból. A feltételezett optimumot az előző 3.2 fejezet alapján határoztam meg 5 méter körüli értékben, majd ezt egyre megnöveltem amíg még értelmes eredményt kaptam. Megfigyeltem, hogy az előző fejezetben meghatározott értékek voltak azok a beállítások aminél még jelentősebb tranziens nélkül indult a rendszer. Ezen paraméterek:

- $s_{torz} \approx 500$
- $k_{torz} \approx 50$
- $ampl \approx 1.3$
- $theta \approx 5$
- $omega \approx 2$

Ezt az eredményt mind az `fsolve()`, mind az `fminsearch()` eljárás visszatéríti, ha a határt 5 méter körülinek állítjuk. Tovább növelve a futási határt azonban azt tapasztaljuk, hogy a féreg jelentős tranziensekkel indul ugyan, de sokkal jobban halad, valamint csak az $theta$ paraméter a vezérlő jel frekvenciája változik.

Az így talált haladási határérték 10 méter körüli 30 másodperces futás alatt

A hozzá tartozó paraméterbeállítások pedig:

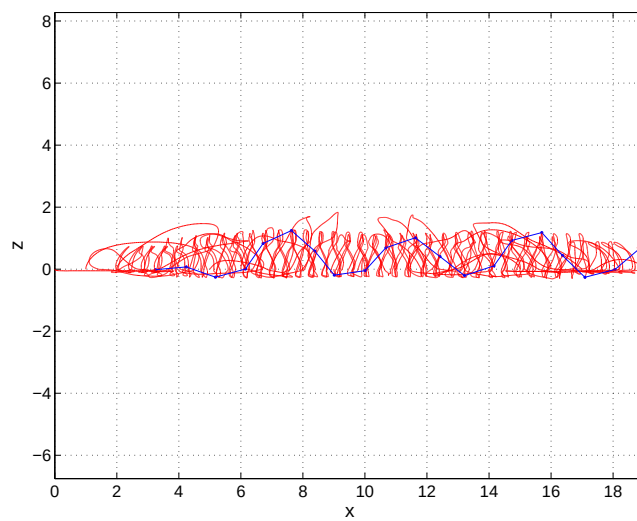
- $s_{torz} \approx 500$
- $k_{torz} \approx 50$
- $ampl \approx 1.3$
- $theta \approx 5.25$
- $omega \approx 2$

Rendszer kezdeti tranziensét úgy tudjuk csökkenteni, hogy az eddig használt vezérlő függvényre (2.81) rászorzunk az alábbi függvénnyel:

$$f = 1 - e^{-a \cdot t}. \quad (3.2)$$

Így a kezdetben vízszintes helyzetben nyugvó féregmodell a vezérlőjel hatására nem összerándulva fogja felvenni a hullámos formáját, hanem szépen fokozatosan, miközben a végső állapotig megtett út hossza alig módosul, hiszen kiküszöböltük a kezdeti rágatózást, amit eddig a belső csillapítások disszipáltak.

Ennél nagyobb távot már csak olyan esetekben tudott megtenni a modell, ahol szelvényei már egymásba gabajodtak. Ezek alapján azt mondhatjuk, hogy az előbb említett paraméter értékekkel érhető el a műizmos vezérlés leghatékonyabb helyváltoztatása.



3.6. ábra: Az optimális paraméterek futási képe 20-as elemszámmal

4. fejezet

Konklúzió, fejlesztési irányok

Láthattuk, hogy munka már jelenlegi stádiumában is számos eredménnyel szolgál. A mechanikai modellezés során feltérképeztük a talajjal való érintkezés két nemlienáris kényszerkapcsolatát, nevezetesen az ütközést és a súrlódást, melyekre sikerült olyan mechanikai és matematikai modelleket készíteni, melyek realisztikusan működnek, valamint a kidolgozott modellek széles körben alkalmazhatóak fizikai modellekre, ahol a talajjal való kapcsolat modellezése kritikus. További előnye ezen modelleknek a könnyű programozhatóság és az, hogy numerikus számítások során jól "kézbentarthatók", tehát stabilak és gyors futattást tesznek lehetővé.

Ezenfelül létrehoztunk egy olyan féregmodellt, mely számos változtatható paraméterrel rendelkezik, melyek közül talán az elemszám változtathatóságának kiemelése a legfontosabb, hiszen itt látható a természetes koordináták előnye a szélesebb körben elterjed Euler szögek alkalmazásával szemben. Egy ekkora modellt, mint amekkorával munkám során foglalkoztam, képtelenség lenne átláthatóan leprogramozni Euler szögekkel, a természetes koordinátás leírás azonban probléma nélkül bővíthető 3 dimenziós modellre. Az eredmények megjelenítése, és az animációk elkészítése pedig ugyancsak kezelhetőbb a természetes koordináták alkalmazásánál. Ezenfelül remekül tükröződik az elsőfajú Lagrange egyenlet mátrixos változatának az előnye, mellyel könnyen kezelhetően lehet merev testrendszerek mozgását leírni, és könnyű modulárisan felépülő rendszert felépíteni, mivel két testet mindössze kényszer egyenletek definiálásával kell összekapcsolni, a mátrixos forma pedig jó algoritmizálhatóságot, emiatt könnyű programozhatóságot és gyors kódot eredményez, ahogy azt a féregmodellnél is láthattuk.

A kód stabilitását pedig egy közkedvelt, de kevésbé publikált eljárás segítségével tettük stabillá, melyre mechanikai ismereteim segítségével kellően szemléletes, de egyszerű példát mutattam, a paraméterválasztást pedig szintén szemléletes mechanikai analógiák segítségével igyekeztem bemutatni az olvasónak.

Emelet a munka kiválóan alkalmas volt többféle optimalizáló eljárás megismerésére, melyek olyan rendszerekre is sikerrel alkalmazhatók, mint a féregmodell, ahol a számos paraméter közötti optimum analitikus úton történő megkeresése valószínűleg nem megoldható.

Azonban rengeteg továbbfejlesztési lehetőség van még a modellen. Jelenleg mindössze sík talajon történő 2 dimenziós haladási esetet vizsgáltunk. Mindenképpen érdekes továbbfejlesztés a talajkényszerek egyenetlen talajra történő adaptálása, mely elsősorban programozástechnikai nehézségeket rejt magában. A 1.1. részben említett

kígyó mozgás pedig azért nem került jelen munkában szimulációra és elemzésre, mert a modell jelenleg csak 2 dimenzióban működik. A háromdimenziós kiterjesztés főleg a kényszerek és a szelvények egymáshoz képesti viszonyából számított erőknél (torziós merevség, csillapítás) jelent módosítást, hiszen 2 dimenzióban ezek az összefüggések viszonylag könnyen leírhatók, míg a 3 dimenziós eset néhány új geometriai problémát vet fel, melyeket a koordináta geometria segítségével könnyedén kezelni lehet.

A modell fejlesztése mellett azonban nem szabad elfeledkeznünk az optimalizálásról sem, hiszen jelenleg egy eléggé leegyszerűsített változatot optimalizáltunk, ezért mindenképpen tovább kell haladni ezen az úton, akár fejlettebb optimumkereső algoritmusok megismerésével, akár elosztott számítógépes rendszerek készítésével [15], melyek nagyobb számítási kapacitást biztosítanak, így hatékonyabban lennének képesek ilyen sokparaméteres rendszerek optimalizálására. Ezzel a peristaltia figyelembevételével meg-jelenő újabb paraméterek is optimálhatóak lennének.

5. fejezet

Köszönetnyilvánítás

A dolgozat az Oktatásért Közalapítvány Tehetséggondozói Alkuratóriumának támogatásával készült a NTP-OKA XXII kódszámú, "A felsőoktatási intézmények karain és tanszékein működő, tehetséggondozó műhelyek és a tudományos diákköri tevékenység keretében működő kutatóműhelyek támogatása" című pályázat keretén belül.

A munka szakmai tartalma kapcsolódik a "Minőségorientált, összehangolt oktatási és K+F+I stratégia, valamint működési modell kidolgozása a Műegyetemen" c. projekt szakmai célkitűzéseinek megvalósításához. A projekt megvalósítását az ÚMFT TÁMOP-4.2.1/B-09/1/KMR-2010-0002 programja támogatja.

A. Függelék

A féregmodell programkódja

Ebben a fejezetben plussz megjegyzések nélkül publikálom a (2.80) hipermátrix egyes elemeinek programkódját. **A modell paraméterei és kezdeti feltételei:**

```
function Par = Param()
% Parameters of the worm
%%
% simulation parameters
Par.n = 20; %number of points
Par.h = 0.01;
Par.itmax = 1500;
%%
%physical parameters
Par.g = 9.81; %[m/s^2] gravity
Par.m = 5; %[kg] mass of the nodes
Par.mu = 0.45; %[-] stict friction on moving
Par.mu0 = 0.55; %[-] stict friction
Par.st = 1000; %[N/m] ground stiffness
Par.kt = 500; %[N*s/m] ground damping
Par.s_torz = 500; %[N*m/rad] stiffness of the nodes
Par.k_torz = 50; %[N*m*s/rad]damping of the nodes

%length of the connecting sticks
Par.L = zeros(Par.n-1,Par.itmax+1);
omega = 1;
theta = pi/4;
ampl = 0;
const = 1; % should be larger than ampl
for i=1:1:Par.n-1
    Par.L(i,:) = ampl*sin( omega*(0:Par.h:Par.h*(Par.itmax))
    + theta*i ) + const; %distance of balls in time
end
%%
% equilibrium point of the torsional springs
Par.phi0 = zeros(Par.n-1,Par.itmax+1);
```

```

theta0 = 5;
omega0 = 3;
ampl0 = 1.4;
const0 = 0; % should be larger than ampl
for i=1:1:Par.n-1
    Par.phi0(i,:) = ampl0*sin( omega0*(0:Par.h:Par.h*(Par.itmax))
        + theta0*i ) + const0; %distance of balls in time
end
%%
%%Baumgarte stabilization
Par.alpha = 100; %baumgarte
Par.beta = 60; %baumgarte

%%
%%Control wave parameters
Par.omega = 1; %speed of the wave
Par.theta = 2; %length of the wave
Par.ampl = 0;
end

function [] = Initialization()

global Par Sol
h = Par.h;
n = Par.n;
itmax = Par.itmax;

Sol.t = 0:h:h*(itmax); % time
Sol.x = zeros(4*n, itmax+1); % coordinates and velocities
Sol.S = zeros(4*n, itmax+1); % friction force
Sol.lambda = zeros(n-1, itmax+1); % lambda lagrange multiplier
Sol.C = zeros(n, itmax+1)+2; % direction of friction force
Sol.phi = zeros(n-2, itmax+1); % relative angle
Sol.M_c = zeros(n, itmax+1); % control torque
end

function [] = SetICs()

global Par Sol
n = Par.n;
L = Par.L;
%%
slant line
phi=1.2; % angle of the first element
dphi = 0;
vx = -0;

```

```

vy = 0;
Sol.phi(:, 1) = dphi*ones(n-2,1);
Sol.x(1,1) = 0;
Sol.x(n+1,1) = 50.05; % height
for i=2:1:n
    Sol.x(i,1) = Sol.x(i-1,1) + L(i-1)*cos(phi+(i-2)*dphi);
    Sol.x(n+i,1) = Sol.x(n+i-1,1) + L(i-1)*sin(phi+(i-2)*dphi);
end
Sol.x(2*n+1:3*n,1) = vx*ones(n,1);
Sol.x(3*n+1:4*n,1) = vy*ones(n,1);
end

```

A tömegmátrix:

```
function M = matrix_M()
```

```

global Par
m = Par.m;
n = Par.n;

```

```

M = m*eye(2*n);
end

```

A kényszereket tartalmazó vektor és deriváltjai:

```
function phi = matrix_phi(x,it)
```

```

global Par
L = Par.L;
n = Par.n;

```

```
phi = zeros(n-1,1);
```

```

for i=1:1:n-1
    phi(i,1) = (x(i)-x(i+1))^2+(x(n+i)-x(n+i+1))^2-L(i,it)^2;
end
end

```

```
function phidot = matrix_phidot( it )
```

```

global Par
L = Par.L;
n = Par.n;
h = Par.h;

```

```
phidot = zeros(n-1,1);
```

```
for i=1:1:n-1
```

```

        phidot(i,1) = - 2*L(i,it)* (L(i,it)-L(i,it-1))/h;
end
end

function phidotdot = matrix_phidotdot( it )

global Par
L      = Par.L;
n      = Par.n;
h      = Par.h;
itmax  = Par.itmax;

phidotdot = zeros(n-1,1);

for i=1:1:n-1
    if it<itmax
        phidotdot(i,1) = - 2*( ((L(i,it)-L(i,it-1))/h)^2
        + L(i,it)*(L(i,it+1)+L(i,it-1)-2*L(i,it))/h^2);
    else
        phidotdot(i,1) = - 2*( ((L(i,it)-L(i,it-1))/h)^2
        + L(i,it)*(L(i,it)+L(i,it-2)-2*L(i,it-1))/h^2);
    end
end
end
end

```

A kényszerek helykoordináták szerint derivált mátrixa és annak időszerinti deriváltja:

```

function Phiq = matrix_Phiq(x)

global Par
n = Par.n;

Phiq = [[ zeros(n-1,1),-eye(n-1)]+eye(n-1,n)
        [ zeros(n-1,1),-eye(n-1)]+eye(n-1,n) ];

for i=1:1:n-1
    Phiq(i,1:n) = 2*Phiq(i,1:n)*(x(i)-x(i+1));
    Phiq(i,n+1:2*n) = 2*Phiq(i,n+1:2*n)*(x(n+i)-x(n+i+1));
end
end

function Phiqdot = matrix_Phiqdot( x )

global Par
n = Par.n;

Phiqdot = [[ zeros(n-1,1),-eye(n-1)]+eye(n-1,n)

```

```

                                [ zeros (n-1,1), -eye (n-1)]+eye (n-1,n) ] ;
for i =1:1:n-1
    Phi qdot (i ,1:n) =      Phi qdot (i ,1:n)*(x (2*n+i)-x (2*n+i +1));
    Phi qdot (i ,n+1:2*n) = Phi qdot (i ,n+1:2*n)*(x (3*n+i)-x (3*n+i +1));
end
end

```

Az átlános erővektor komponensei:

```
function Q = matrix_Q()
```

```

global Par
n = Par.n;
g = Par.g;
m = Par.m;

```

```

Q= -m*g*[ zeros (n,1); ones (n,1) ] ;
end

```

```
function Qi = matrix_Qinner( x, it )
```

```
%%
```

```

global Par Sol
n = Par.n;
h = Par.h;
s_torz = Par.s_torz;
k_torz = Par.k_torz;

```

```
Qi = zeros (2*n,1);
```

```
%parameters of the control moment
```

```

omega = Par.omega;
theta = Par.theta;
ampl = Par.ampl;

```

```
for i =2:1:n-1
```

```
%%
```

```
%Calculating the needed vectors within the nodes
```

```
r1=[x(i)-x(i-1); x(n+i)-x(n+i-1)];
```

```
r2=[x(i+1)-x(i); x(n+i+1)-x(n+i)];
```

```
L1 = norm(r1);
```

```
L2 = norm(r2);
```

```
e1=r1/L1;
```

```
e2=r2/L2;
```

```
n1=rot90(rot90(e1)).*[-1;1];
```

```
n2=rot90(rot90(e2)).*[-1;1];
```

```
R = [e1,n1];
```

```

    r2L = R'*r2;
    Sol.phi(i-1, it) = atan2(r2L(2),r2L(1));
    phidot = (Sol.phi(i-1, it)-Sol.phi(i-1, it-1))/h;
%%
%%calculating the moment of the spring and the damping
M = s_torz*(Sol.phi(i-1, it) - Par.phi0(i-1, it)) + k_torz*phidot;

%%
%%generate a control moment

M_c = ampl*heaviside(square(theta*i+omega*it*h));

F1 = - (M+M_c)/L1*n1;
F2 = - (M+M_c)/L2*n2;
%%
%%Build an inner force vector
    Qi( i-1) = Qi( i-1) + F1(1);
% point i-1; x component
    Qi(n+i-1) = Qi(n+i-1) + F1(2);
% point i-1; z component

    Qi( i) = Qi( i) - F1(1)-F2(1); % point i;
x component
    Qi(n+i) = Qi(n+i) - F1(2)-F2(2); % point i;
z component

    Qi( i+1) = Qi( i+1) + F2(1);
% point i+1; x component
    Qi(n+i+1) = Qi(n+i+1) + F2(2);
% point i+1; z component

    Sol.M_c(i, it) = M_c;
end
end

```

A rendszert mozgását számító függvény:

```

function [fun, S, lambda] = System(x, C, lambda_old, runge_kutta_iter)
%%
%Initialization of the current step
global Par
g = Par.g;
m = Par.m;
n = Par.n;
mu = Par.mu;
mu0 = Par.mu0;

```

```

st =      Par.st;
kt =      Par.kt;
alpha =   Par.alpha;
beta  =   Par.beta;

x_part(:,1) = x(1:n);
z(:,1) = x(n+1:2*n);
xdot(:,1) = x(2*n+1:3*n);
zdot(:,1) = x(3*n+1:4*n);
S = zeros(n,1);
Ft = zeros(n,1);
%%
%Get the matrixes to the Lagrange equation
%Mass matrix
M = matrix_M();
%Matrix of the constraints (the distance of the nodes)
phi = matrix_phi(x,runge_kutta_iter);
%Derivative of the constraints by time
phidot = matrix_phidot(runge_kutta_iter);
phidotdot = matrix_phidotdot(runge_kutta_iter);
size_phi = size(phi,1);
%Derivative of the constraints by coordinates
Phiq = matrix_Phiq(x);
%Derivative of the constraints by coordinates and time
Phiqdot = matrix_Phiqdot(x);
%external force
Qg = matrix_Q();
%internal stiffnes and damping within the nodes and the control force on
%the nodes
Qi = matrix_Qinner(x, runge_kutta_iter);
%%
%Natural coordinates
%x, and z coordinates
q = [x_part; z];
%x, and z velocities
y = [xdot; zdot];
%%
%calculating the friction force and the force of the ground

%inner constraint forces
Q_k=Phiq'*lambda_old;
Fx(:,1)=Qg(1:n)+Q_k(1:n);
Fz(:,1)=Qg(n+1:2*n)+Q_k(n+1:2*n);
for i=1:1:n
    if z(i,1) <= 0
        switch C(i,1)
            case 0

```

```

        S(i,1) = Fx(i,1);
        if abs(S(i,1)) > mu0*m*g
            S(i,1) = mu*m*g*sign(Fx(i,1));
        end
    case 1
        S(i,1) = mu*m*g;
    case -1
        S(i,1) = -mu*m*g;
    end
else
    S(i,1) = 0;
end

if z(i,1) <= 0 %talajrol atadodo ero
    Ft(i,1) = - st*z(i,1) - kt*zdot(i,1);
else
    Ft(i,1) = 0;
end

end
%forces from the contact with the ground
QTalaj = [-S; Ft];
%%
%lagrange equation
%creating the hyper matrixes to the lagrange equation
HypM = [M Phiq'; Phiq zeros(size_phi,size_phi)];
HypQ = [Qg + QTalaj + Qi; - Phiqdot*y - phidotdot - 2*alpha*(Phiq*y + phi
%calculating the acceleration in natural coordinates
qddotlambda = HypM\HypQ;
%get the inner constrain forces
lambda=qddotlambda(2*n+1:3*n-1);
%%
%iterating with the current constraint forces to get the proper accelerati
for iter=1:1:1
    Q_k=Phiq'*lambda;
    Fx(:,1)=Qg(1:n)+Q_k(1:n);
    Fz(:,1)=Qg(n+1:2*n)+Q_k(n+1:2*n);
    for i=1:1:n
        if z(i,1) <= 0
            switch C(i,1)
                case 0
                    S(i,1) = Fx(i,1);
                    if abs(S(i,1)) > mu0*m*g
                        S(i,1) = mu*m*g*sign(Fx(i,1));
                    end
                case 1
                    S(i,1) = mu*m*g;
                case -1

```

```

                                S(i,1) = -mu*m*g;
                        end
                else
                        S(i,1) = 0;
                end

                if z(i,1) <= 0 %talajrol atadodo ero
                        Ft(i,1) = - st*z(i,1) - kt*zdot(i,1);
                else
                        Ft(i,1) = 0;
                end
        end

        QTalaj = [-S; Ft];
        HypM = [M Phiq'; Phiq zeros(size_phi,size_phi)];
        HypQ = [Qg + QTalaj + Qi; - Phiqdot*y - phidotdot - 2*alpha*(Phiq*y +

% qddotlambda = inv(HypM)*HypQ;
qddotlambda = HypM\HypQ;

        lambda=qddotlambda(2*n+1:3*n-1);
end
%%
%return to the Ringe-Kutta
fun = [xdot; zdot; qddotlambda(1:2*n)];

```

Irodalomjegyzék

- [1] Fukunaga A.S., Morookian J.M., Quillin K. et al., EARTHWORMLIKE EXPLORATORY ROBOTS, Technical report, NASA, 1998.
- [2] Paulina Varshavskaya MIT B. and Jorg Conradt ETH Z., WormBot A Mobile Robotic Worm, Technical report, MIT,ETH, 2003.
- [3] Behn C. and Zimmermann K., Straight worms under Adaptive Control and Friction, in *Dynamics & Modelling Interactions*, Budapest, Hungary, 2011.
- [4] Boxerbaum A., Chiel H. and Quinn R., Softworm: A Soft, Biologically Inspired Worm-Like Robot.
- [5] Steigenberger J., Some theory towards a stringent definition of 'locomotion', *Multibody System Dynamics*, 26(1):81–90, 2011.
- [6] Stépán G., Dinamika (BMEGEMMAGM3), Rezgésstan (BMEGEMMAGM4), Robotok Mechanikája (BMEGEMMAM32) jegyzetek.
- [7] Wikipaedia, Száraz súrlódás, URL http://hu.wikipedia.org/wiki/S%C3%BAr1%C3%B3d%C3%A1s#Sz.C3.A1raz_s.C3.BAr1.C3.B3d.C3.A1s, access date: 30/10/2011, (in Hungarian).
- [8] Baumgarte J., Stabilization of constraints and integrals of motion in dynamical systems, *COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING (NORTH-HOLLAND PUBLISHING COMPANY)*, (1):1–16., 1972.
- [9] Jalón J.G.D. and Bayo E., *Kinematic and dynamic simulation of multibody systems: the real-time challenge*, Springer-Verlag, 1994.
- [10] Flores P., Machado M., Seabra E. et al., A Parametric Study on the Baumgarte Stabilization Method for Forward Dynamics of Constrained Multibody Systems, *Journal of Computational and Nonlinear Dynamics*, 6, 2011.
- [11] Bertalan C. and Zoltán J., 20 forint kicsit másképp (TDK Dolgozat), Technical report, Budapest University of Technology, 2010.
- [12] Juhász Z., Robotok Mechanikája (BMEGEMMAM32) Házi Feladat Dokumentáció, 2011.
- [13] Bertalan C. and Zoltán J., Versenyben az idővel maximális gyorsítás a versenypályán (TDK Dolgozat), Technical report, Budapest University of Technology, 2009.

- [14] AsSpec, AsSpec Rugókatalógus, URL <http://www.assocspring.co.uk/catalogue/PDF/Product%20Selector%20UK/SPEC%2009%20UK/Sec%2008%20Torsion%20Selector%20UK.pdf>, access date: 07/12/2011, (in English).
- [15] Juhász Z., Szakmai gyakorlat beszámoló, Technical report, 2011.