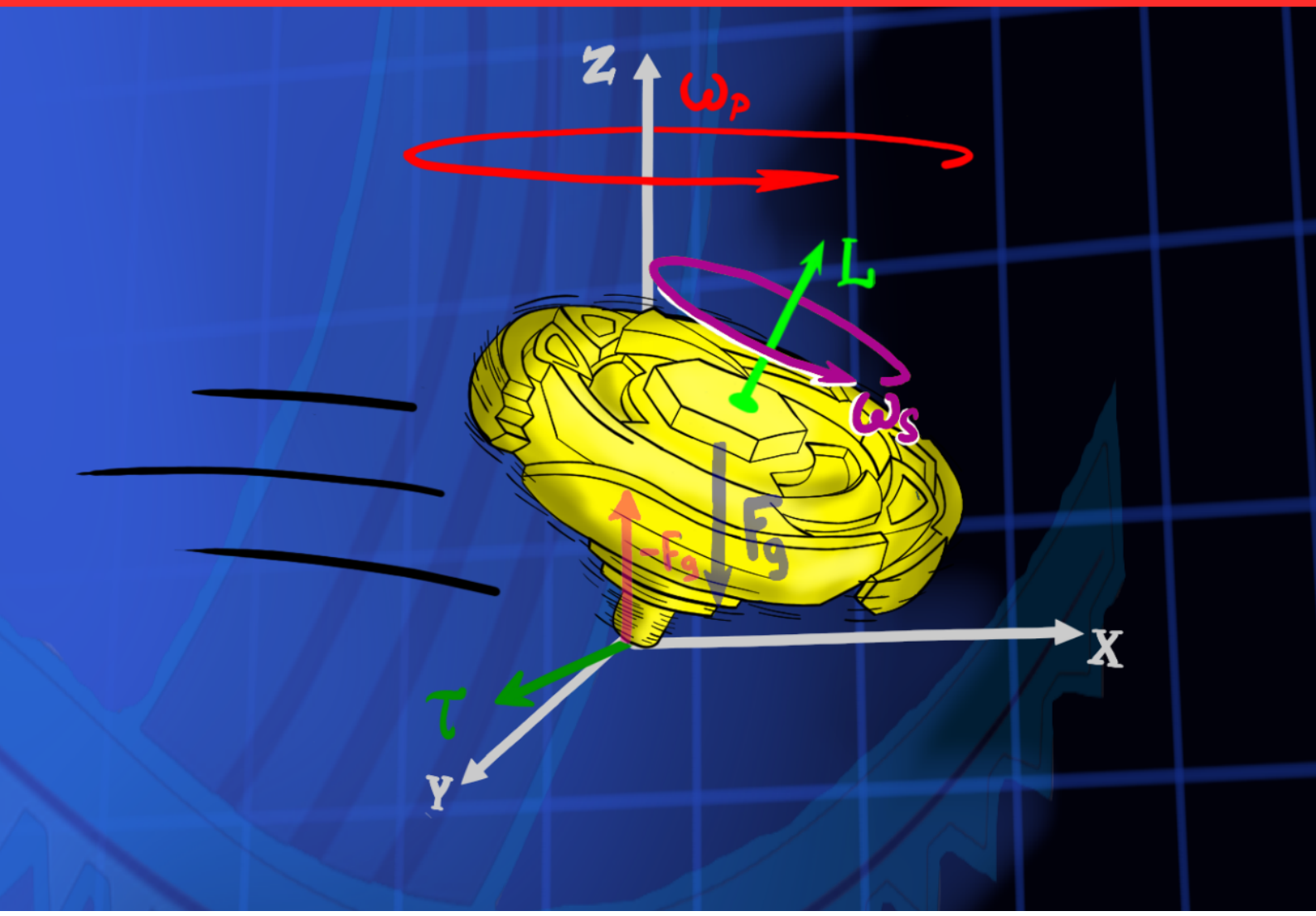
Modelleringsprojekt - TNM085

Simulering av en snurra

MT3a: Erik Asp, Jakob Gunnarsson, Julius Kördel och Nicholas Frederiksen

12 Mars 2018



Sammanfattning

Detta projekt resulterade i en visualisering av ett fysikaliskt system bestående av acceleration, hastighet och position för ett objekt. Detta objekt är en snurra vilken roterar runt sin egen axel och beroende på underlaget förflyttar sig i den bana underlaget tillåter. Visualiseringen grundas i en differentialekvation för att säkerställa att det visualiserade resultatet är av en tillräckligt korrekt grad och representerar verkligheten på ett rättvist sätt.

Innehåll

Figurer

Symboler

Kommande lista presenterar symboler vilka förekommer i detta dokument och deras betydelse.

μ	Motståndskonstant för luft
ω	Vinkelhastighet hos snurran
ω_1	Vinkelhastigheten hos objektet innan kollision
ω_2	Vinkelhastigheten hos objektet efter kollision
ω_p	Vinkelhastigheten för snurrans vertikala axel kring rummets vertikala axel
τ	Momentet på snurran från masscentrum
$\mathbf{n}$	Normalen mellan de olika objekten i kollisionspunkten
$\mathbf{r}_{\perp}^{AP}$	Vektorn från masscentrum hos objekt A till kollisionspunkten
$\mathbf{r}_{\perp}^{BP}$	Vektorn från masscentrum hos objekt B till kollisionspunkten
$\mathbf{v}_1^A$	Hastigheten hos objekt A innan kollision
$\mathbf{v}_1^B$	Hastigheten hos objekt B innan kollision
$\mathbf{v}_1^{AB}$	Skillnaderna mellan hastigheterna för objekt A och B
$\mathbf{v}_2^A$	Hastigheten hos objekt A efter kollision
$\mathbf{v}_2^B$	Hastigheten hos objekt B efter kollision
θ	Vinkeln mellan snurrans vertikala axel och marken
θ_1	Vinkeln θ under nuvarande h
θ_2	Den nya θ efter nuvarande h
φ_{n-1}	Den gamla vinkeln från föregående beräkning
φ_n	Den nya nuvarande vinkeln
a	Accelerationen vid translation
a_R	Acceleration för momentaxeln R
b	Motståndskonstant för ytan vid translation
b_{yta}	Motståndskonstant för ytan vid rotation

e	Konstant mellan 0 till 1 som beskriver elasticiteten hos en kollision
F_v	Kraften vilken bidrar till initialhastigheten
F_{tot}	Totala kraften verkande på snurran
F_{yta}	Bromsande kraft vilken verkar på snurran
h	Steglängden vid simuleringen
I	Tröghetsmomentet för en kon
I_A	Tröghetsmomentet hos objekt A
I_B	Tröghetsmomentet hos objekt B
L	Rörelsemängdsmomentet hos snurran
l	Längden på snurran
M	Moment genererat av källa
m	Massa för snurra
m_A	Massan hos objekt A
m_B	Massan hos objekt B
M_I	Rörelse Moment hos snurran
M_{luft}	Bromsande moment på grund av luftmotstånd
M_{yta}	Bromsande moment mot ytan
R	Längden från marken till snurrans masscentrum
r	Radien hos snurrans topp
v	Hastigheten vid translation
v_R	Hastighet för momentaxeln R

Kapitel 1

Inledning

1.1 Inledning

I sin barndom har många lekt med olika typer av snurror. Ur detta föddes idén att återuppleva den glädje snurrorna en gång gav genom att visualisera en. I detta projekt ska en modell för beteendet hos en snurra framtas. Genom denna modell ska simulering för snurrans rotation och translation utföras. Fenomen som också kommer att undersökas för att bidra till en mer verklighetstrogen visualisering är precession(vobbel), kollision och att få snurran att välta.

Som en referens till visualiseringen användes det hjälpmedel i form av riktiga snurror. Dessa är av den typ vilken önskades att simuleras. En sådan snurra visas i 1.1, där snurran har en rotation kring sin egen axel med påbörjat vobbel. Dock syns ingen translation eller rotation i bilden då rörelse är svår att visualisera i en bild.

Figur 1.1: Exempel på en snurra

1.2 Syfte

Syftet med detta projekt är att genom framtagande av differentialekvationer för acceleration, hastighet och position, genomföra en simulation av ett fysikaliskt system. Denna simulation skall sedan vara den grund vilken används för att visualisera systemet med hjälp av valfri metod och med valfritt

medel.

Målen är att kunna visualisera en snurra med translation och rotation i en halvsfärsutformad bana, att snurran ska kunna välta samt att utreda ifall en implementation av vobbel är möjligt. Sedan i mån av tid även kunna visualisera kollision mellan två snurror.

Kapitel 2

Metod

2.1 Simulering

Simuleringen vilken utgör grunden för systemet är genomförd i MATLAB och den stegmetod som använts är Eulers stegmetod [1]. Den tar i detta fall ett steg av längden 1/60. Alltså tas 60 steg per sekund för att matcha uppdateringsfrekvensen hos skärmen. Detta medför att varje gång den visuellt uppdateras har fysiken och matematiken bakom gjort detsamma. Vilket ska säkerhetsställa att visualiseringen följer simuleringen.

Den allmänna formeln för en algoritm enligt eulermetod kan skrivas enligt 2.1. Denna metod behandlar första ordningens differentialekvation och är en enstegsmetod. Detta innebär att den använder en startpunkt och en punkt framåt sedan tas tangenten där emellan för att förutspå hur förändringen kommer ske i nästa tidssteg.

$$x_{n+1} = (1 + h * \lambda)x_n \tag{2.1}$$

2.2 Modell

Modellen är uppbyggd av flera delar. Dessa är rotation, translation, vobbel och kollision.

2.2.1 Rörelser

För framtagandet av differentialekvationer har bindningsgraf och kraftekvation använts. Dessa har använts för att uttrycka påverkande parametrar i systemet och genom parametrarnas ekvationer funnit uttryck med den gemensamma variabeln hastighet.

För vinkelhastighet har en bindningsgraf med rörelse, motstånd mot yta och luftmotstånd tagits fram, med koefficienter enligt [2] för ytan och [3] för luftmotståndet. Translationshastigheten är baserad på en kraftekvation i vilken luftmotstånd ej har inkluderats. Värden för ytans friktionskonstanter i de olika delarna varierar då rörelsen är annorlunda för rotation och translation.

2.2.2 Underlag

Modellen för hur underlaget påverkar translationen har utökats med projektets gång. Från början var det endast en plan yta där en initialhastighet med en inbromsande verkan på grund av motstånd i

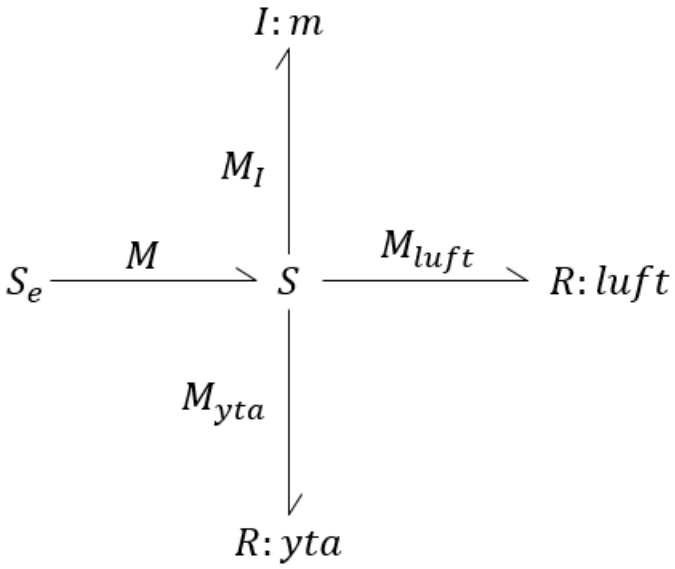
ytan[2] applicerats på snurran. Nästa yta var ett lutande plan vilket fungerade likt en plan yta fast med en konstant acceleration.

Innan snurran rörde sig längs en halvsfärisk bana utreddes fallet för en halvpipa, vilket är ett specialfall när snurran i den halvsfäriska banan är utan initialhastighet.

För att beskriva rörelsen i halvsfär med en initialhastighet ska en modell för en sfäriska pendel användas [4].

2.2.3 Rotation

Bindningsgrafen vilken representerar vinkelhastighet visas i 2.1.



Figur 2.1: Bindningsgraf för hastigheterna

Ur bindningsgrafen härledes 2.6. Den relevanta bakgrunden för de olika elementen och deras ekvationer synes i 2.2 till 2.5. I 2.6 ses 2.2 omskrivet med hjälp av 2.3 till 2.5. Detta är sedan förenklat i 2.7 och 2.8. För samtliga ekvationer ses förklaringar för symboler i Symbollistan.

$$s : M - M_I - M_{yta} - M_{luft} = 0, \tag{2.2}$$

$$I : m : \omega = \frac{1}{m} * \int M_I \mathrm{d}\tau, \tag{2.3}$$

$$R : Yta : M_{yta} = b_{yta} * \omega, \tag{2.4}$$

$$R : Luft : M_{luft} = \frac{2 * \pi * \mu * r * l * \omega}{3 * m}, \tag{2.5}$$

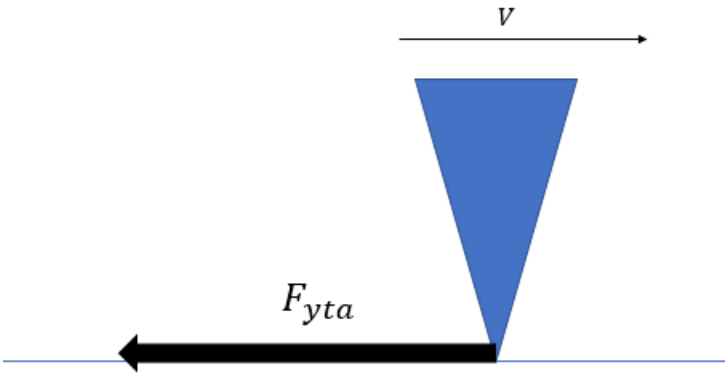
$$\frac{3}{10} * m * r^2 * \omega(t) - \omega'(t) * m - \omega(t) * b_{yta} - \frac{2 * \pi * \mu * r * l * \omega(t)}{3 * m} = 0 \tag{2.6}$$

$$(\frac{3}{10} * r^2 - \frac{b_{yta}}{m} - \frac{2 * \pi * \mu * r * l)}{3 * m}) * \omega = w'(t) \tag{2.7}$$

$$\lambda_{rot} = \frac{3}{10} * r^2 - \frac{b_{yta}}{m} - \frac{2 * \pi * \mu * r * l}{3 * m} \tag{2.8}$$

2.2.4 Translation

En skiss med krafternas friläggning på snurran vid translation ses i 2.2. Där initialkraften är tillräckligt stor för att generera en starthastighet vilken slutar verka när initialhastigheten är nådd. Det finns ett motstånd i ytan den färdas på vilket medför att den resulterande hastighet är retarderande. Ekvationerna härledda från 2.2 ses i 2.9 till 2.11.



Figur 2.2: Kraftekvation för translation

$$F_{tot} = F_v - F_{yta}, \tag{2.9}$$

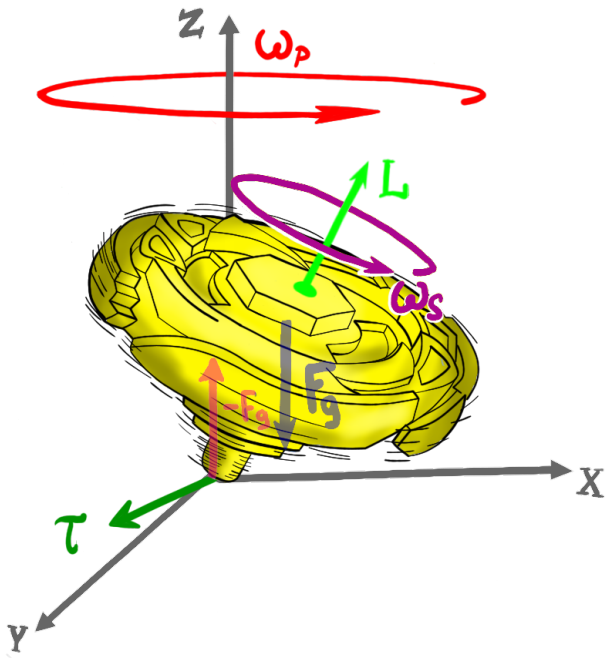
$$m * a(t) = 0 - b * v(t), \tag{2.10}$$

$$a(t) = -\frac{b}{m} * v(t), \tag{2.11}$$

2.2.5 Vobbel

Utöver att vinkelhastigheten minskar med tiden minskar också snurrans förmåga att hålla sig upprätt. Förutom det udda fallet att snurran är perfekt balanserad kommer den att ramla ihop innan det att den slutar snurra. För att implementera denna egenskap i modellen utforskades ett fysikaliskt fenomen kallat precession. Precession hos snurror är en mekanisk funktion där rotationsaxeln pendlar i rummet. Vilket kan ses som en avvikelse av snurrans grundrörelse av första ordningen.

Målet var att hitta ett samband mellan vinkelhastigheten (ω) och vinkeln mellan snurrans vertikal axel och marken. Vinkeln i fråga benämns med θ och är noll när snurran står rakt vertikal mot marken. Som utgångspunkt användes formeln för rörelsemängdsmoment, där snurrans tröghetsmoment [5] och vinkelhastighet ω ingår. Tröghetsmomentet för snurran uppskattas till tröghetsmomentet för en kon. Rörelsemängdsmomentet kan ses som den kraft som hindrar snurran från att ramla ihop. När θ blir större än noll sker ännu en rotation utöver den runt snurrans egen axel, en rotation (ω_p) kring den lodräta axeln i rummet. En skiss över snurrans axlar och rotationer kan ses i 2.3. Då vinkelhastigheten blir lägre minskar också rörelsemängdsmomentet enligt 2.12.



Figur 2.3: Skiss över krafter och axlar kring snurran

$$L = I * w \tag{2.12}$$

Ett försök till att beskriva vinkeln mellan snurran och marken med skalära ekvationer påbörjades. Ekvationerna 2.13 skrevs om till 2.14 och 2.15 skrevs om till 2.16. För att få uttryck som beskriver vinkeln θ med avseende på tid. För att uttrycka ω_p användes ekvationen 2.17 [6].

$$\tau(t) = \frac{\Delta L}{\Delta t} \tag{2.13}$$

$$\tau(t) = \frac{Iw(t+h) - Iw(t)}{h} \tag{2.14}$$

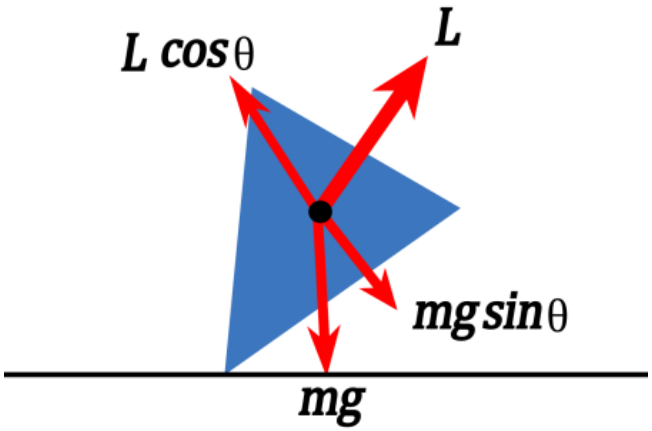
$$\tau(t) = mgR\sin\theta(t) \tag{2.15}$$

$$\theta(t) = \sin^{-1}(\frac{\tau}{mgR}) \tag{2.16}$$

$$\omega_p(t) = \frac{\tau(t)}{mgR} \tag{2.17}$$

Ett andra försök påbörjades med samma mål som det första försöket. Skillnaden var att denna gång beskriva modellen med vektorer och matriser. En beskrivning av ekvationerna på vektorform kan ses i bilaga A.

För att beskriva relationen mellan snurrans olika rotationer generellt kan Lagranges ekvationer [7] användas. Detta utreds inte i stor utsträckning i detta projekt. Istället skedde ett tredje försök för att reda ut vinkeln θ. I detta tredje försök tacklades problemet från en ny vinkel. Utgångspunkten var fortfarande ekvationen för rörelsemängdsmoment men att direkt ställa upp en kraftekvation mellan rörelsemängdsmomentet och gravitationskraften enligt 2.4.



Figur 2.4: Kraftekvation för rörelsemängdsmoment

I detta tredje försök användes ett nytt sätt för att beskriva tröghetsmomentet. Genom att sätta ihop tröghetsmomentet för z-axeln och y-axeln med beroende på vinkel enligt 2.18. Ett uttryck för det moment vilket drar snurran mot marken togs också fram enligt 2.19.

$$I(t) = \cos\theta(t) * I_z + \sin\theta(t) * I_y \tag{2.18}$$

$$\tau(t) = R * (\sin(\theta(t)) * mg - \cos(\theta(t)) * L) \tag{2.19}$$

För att använda momentet till att beskriva θ behöver det ske vissa omskrivningar. Den nya beskrivningen av tröghetsmomentet 2.18 användes tillsammans med momentet i 2.19 för att ta fram accelerationen enligt 2.20.

$$a(t) = \frac{\tau(t)}{I(t)} \tag{2.20}$$

Accelerationen multiplicerades sedan med tiden i form av steglängden h för att få hastighet enligt 2.21. Hastigheten multiplicerades med radien för snurran i 2.22 för att få vinkelhastigheten $\dot{\theta}$.

$$v_R(t) = a(t) * h \tag{2.21}$$

$$\dot{\theta} = v_R(t) * r \tag{2.22}$$

För att uppdatera vinkeln adderas den nuvarande vinkeln θ med vinkelhastigheten $\dot{\theta}$ som har multiplicerats med steglängden enligt 2.23. Denna nya vinkel användes sedan för att uppdatera alla variabler vilka beror av θ.

$$\theta_2 = \theta_1 + \dot{\theta} * h \tag{2.23}$$

2.2.6 Kollision

Att två snurror kolliderar med varandra ingår också i en heltäckande fysikalisk modell för snurror. När två snurror kolliderar med en viss hastighet och vinkelhastighet kommer dessa att påverkas beroende på både den egna snurrans egenskaper samt den andras. För att beräkna dessa har formeln för två roterande kolliderande objekt använts.[8]

$$j = \frac{-(1+e)\mathbf{v}_1^{AB} \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}(\frac{1}{m_A} + \frac{1}{m_B}) + \frac{(\mathbf{r}_\perp^{AP} \cdot \mathbf{n})^2}{I_A} + \frac{(\mathbf{r}_\perp^{BP} \cdot \mathbf{n})^2}{I_B}} \tag{2.24}$$

I 2.24 beräknas en vektor j som senare ska användas för att beräkna de nya hastigheter som objekten besitter efter kollisionen. De olika variablerna i formeln beror på både snurrornas samt kollisionens egenskaper. Variabeln e sätts till 1 eftersom det är en helt elastiskt stöt. Massorna av snurrorna beräknas innan modellen används vilket också tröghetsmomenten görs.

$$I = \frac{3}{10} * m * r^2 \tag{2.25}$$

Tröghetsmomenten beräknas med hjälp av 2.25. Resterande variabler och vektorer beräknas precis när kollisionen sker. Dessa kommer antingen att beräknas med hjälp av programmet eller beräknas innan för en förutbestämd kollision. Efter j har beräknats kan de nya hastigheterna beräknas.

$$\mathbf{v}_2^A(t) = \mathbf{v}_1^A(t) + \frac{j \cdot \mathbf{n}}{m_A} \tag{2.26}$$

$$\mathbf{v}_2^B(t) = \mathbf{v}_1^B(t) - \frac{j \cdot \mathbf{n}}{m_B} \tag{2.27}$$

I 2.26 och 2.27 beräknas de nya hastigheterna. För att dessa funktioner ska fungera ska objekt A ha större hastighet än objekt B. De nya vinkelhastigheterna beräknas på liknande sätt.2.28 visar hur båda nya vinkelhastigheterna beräknas med hjälp av vektorerna som hittats samt tröghetsmomentet.

$$\omega_2(t) = \omega_1(t) + \frac{\mathbf{r}_\perp^{AP} \cdot j\mathbf{n}}{I_A} \tag{2.28}$$

2.3 Visualisering

För att visualisera snurran behövdes först ett 3D-objekt skapas. Objektet skapades i programmet 3ds max. Objektet som ritades var gjord för att efterlikna leksaken som projektet inspirerades av. Trots detta har modellen fortfarande ungefär formen av en kon och det är detta som har räknats med i alla uträkningar.

Animationen och simuleringen av snurran har genomförts i OpenGL och spelmotorn Unity. OpenGL användes i början för att simulera enkla transformationer och senare användes Unity för att simulera mer krävande transformationer.

2.3.1 OpenGL

För att implementera differentialekvationer i OpenGL användes renderingsloopen som löps igenom om och om igen. Innan renderingen i programmet startade angavs värden för snurran vilka skulle användas såsom radie, massa, friktionskonstanter etc. I OpenGL används inte vinkelhastigheten för att visa upp hur mycket den roterar utan vilken vinkel den har roterat till från start vid just den tidpunkten. Värdet som fås ut av differentialekvationen är en vinkelhastighet, därför behövs denna skrivas om.

Innan vinkelhastigheten kunde skrivas om behövde den först fungera ensam. Istället för att skriva in alla värden i en vektor som i MATLAB uppdaterades istället vinkelhastigheten varje gång i renderingsloopen, där de gamla värdena byttes ut mot nya för att spara datorkraft och tid. De enda vinkelhastigheterna som fanns samtidigt var den nuvarande hastigheten och hastigheten som beräknades förra gången i loopen.

$$\varphi_n = \varphi_{n-1} + h * \omega(t) \tag{2.29}$$

I 2.29 visas hur den nuvarande vinkeln beräknas med hjälp av den gamla vinkeln, steglängden samt hastigheten.

2.3.2 Unity

Även om OpenGL är ett bra verktyg för att implementera en enkel grafisk tillämpning är det begränsat när det handlar om att förflytta sig mycket samt kunna använda flera objekt som ska matcha varandra. För att underlätta denna implementation användes spelmotorn Unity där inbyggda funktioner för rotation och translation finns. Det första som genomfördes var att kunna tillämpa rotationen även i detta program. Till skillnad från OpenGL behövdes endast hur mycket den skulle rotera implementeras istället för vilken vinkel den befann sig på precis vid den tidpunkten.

Nästa del av modellen vilken implementerades var translationen i en bana. Denna del var enklare att implementera eftersom Unitys inbyggda funktion för translation endast behövde de koordinater snurran skulle befinna sig på vid en viss tidpunkt. På grund av detta blev det lättare att skriva in ekvationerna från MATLAB likt hur de gjorts för rotation.

Kapitel 3

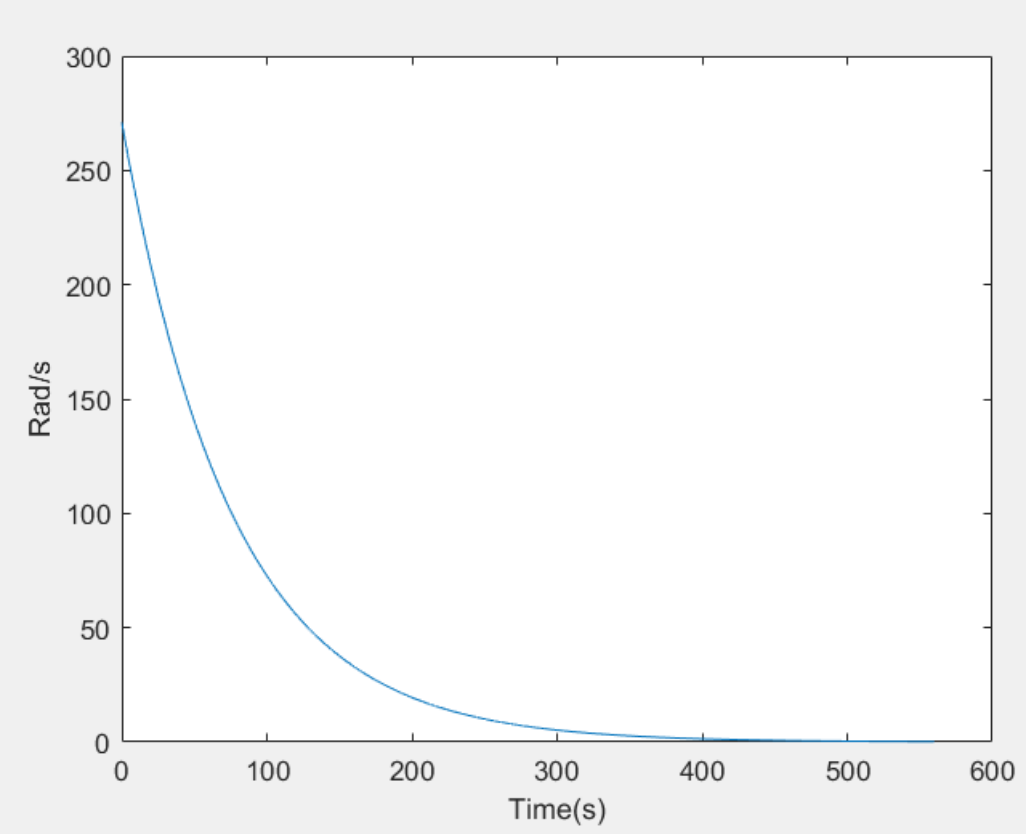
Resultat

3.1 Modeller

3.1.1 Rotation

Ur 2.7 gives att vinkelaccelerationen kan uttryckas med vinkelhastigheten och konstanter. Vilket är det önskade formatet för att använda Eulers stegmetod. Den simuleringsekvation vilken har använts kan då uttryckas enligt 3.1 vilket använder omskrivningen från 2.8. Denna simulering är genomförd med en initialhastighet på 273 rad/s [9] och avtar sedan med tiden mot noll, detta ses i 3.1. Observera att det tar väldigt lång tid för vinkelhastigheten att gå mot noll. Dock kommer snurran att ramla innan den hastigheten nås på grund av att rörelsemängdsmomentet är för litet.

$$\omega_{n+1} = (1 + h * \lambda_{rot})\omega_n \tag{3.1}$$

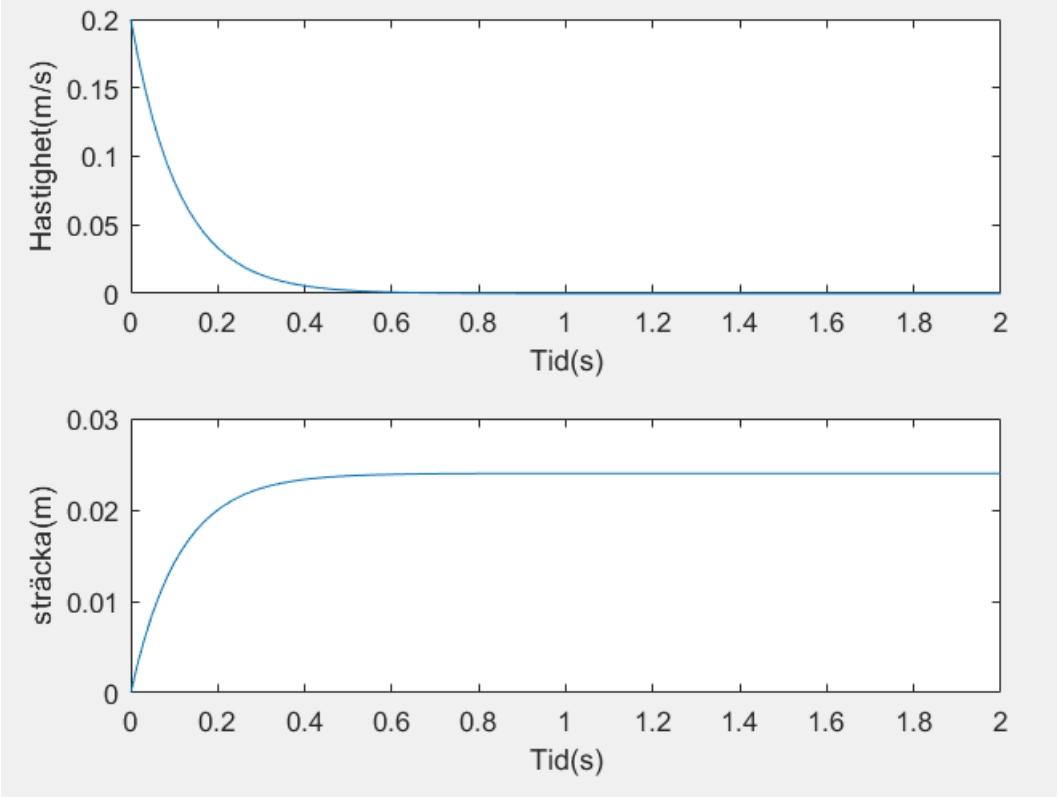


Figur 3.1: Vinkelhastigheten över tiden.

Rotationen hos snurran skapar en synvilla på grund av uppdateringsfrekvensen och den höga vinkelhastigheten, vilka tillsammans får snurran att se ut som att den momentant stannar och byter rotationsriktning [10]. Även fast den fortfarande har en stor vinkelhastighet i samma riktning.

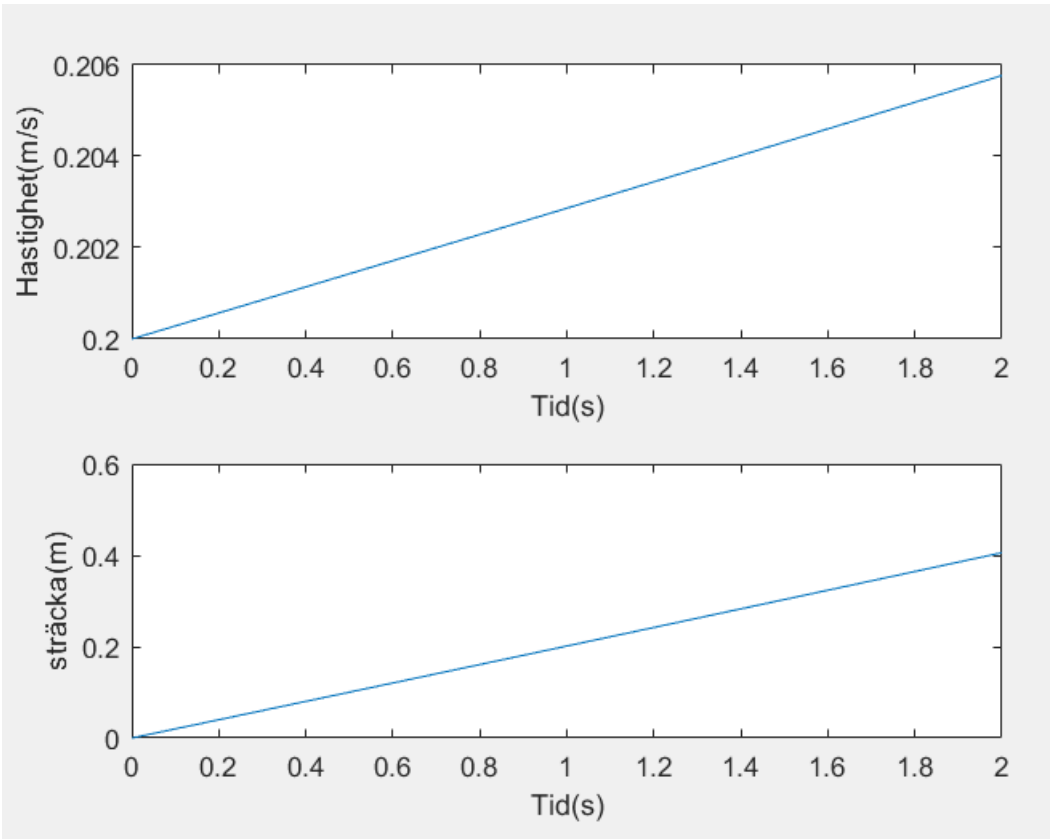
3.1.2 Translation

Hastigheten och förflyttningen över tiden på en platt yta med initialhastighet ses i 3.2. Med hastigheten i den övre och sträckan i den undre.



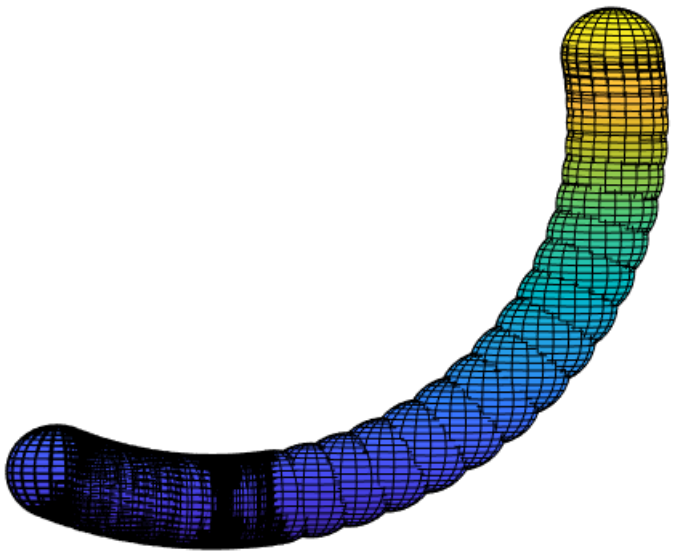
Figur 3.2: Hastigheten och translationen över tiden på ett plan

I 3.3 visas enligt samma formalia som i 3.2 fast för ett lutande plan utan initialhastighet.



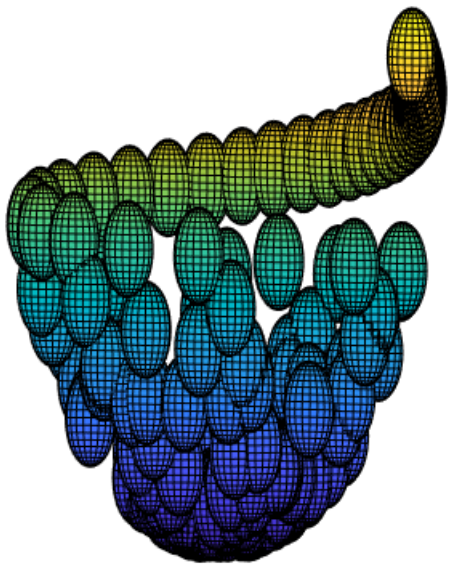
Figur 3.3: Hastigheten och translationen över tiden i ett lutande plan

Resultatet av modellen för en halvpipa visas i 3.4 där mörkare delar motsvarar där den varit ofta. Vilket är centrerat kring centrum då den pendlande rörelsen avtar.



Figur 3.4: Rörelsen för snurran i en halvpipa

Resultatet av snurrans rörelse i en halvsfär ses i 3.5. I figuren visas snurrans förflyttning med gult i starten och mörkblått i slutet.



Figur 3.5: Translationen hos snurra i en halvsfärisk bana

3.1.3 Vobbel

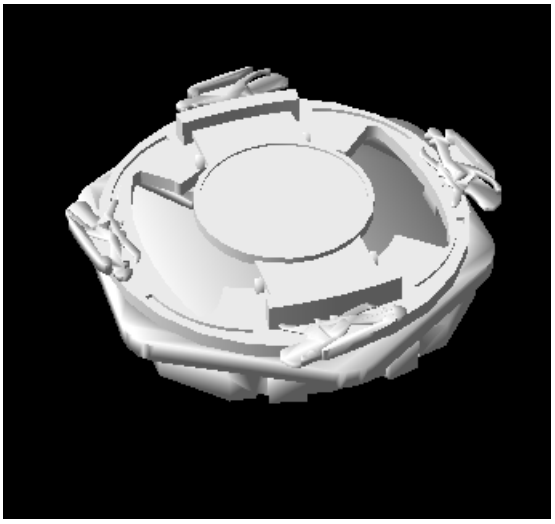
Implementation av försök ett visade att denna modell för vobbel inte påverkade snurran. Den fortsatte snurra utan några hinder. Försök två gav samma resultat som försök ett. Implementation av försök tre i OpenGL gav att snurran ramlar ihop efter en viss tid, runt en minut. Dock saknades något tydligt vobbel och det är fortfarande skillnad mellan modellens beteende och en riktig snurra.

3.2 Grafisk tillämpning

Den grafiska tillämpningen för detta projekt har genomförts i OpenGL och Unity. I projektets tidiga skede användes OpenGL för att simulera rotationen eftersom rotationen är en enkel transformation och objektet behöver ej förflyttas.

3.2.1 OpenGL

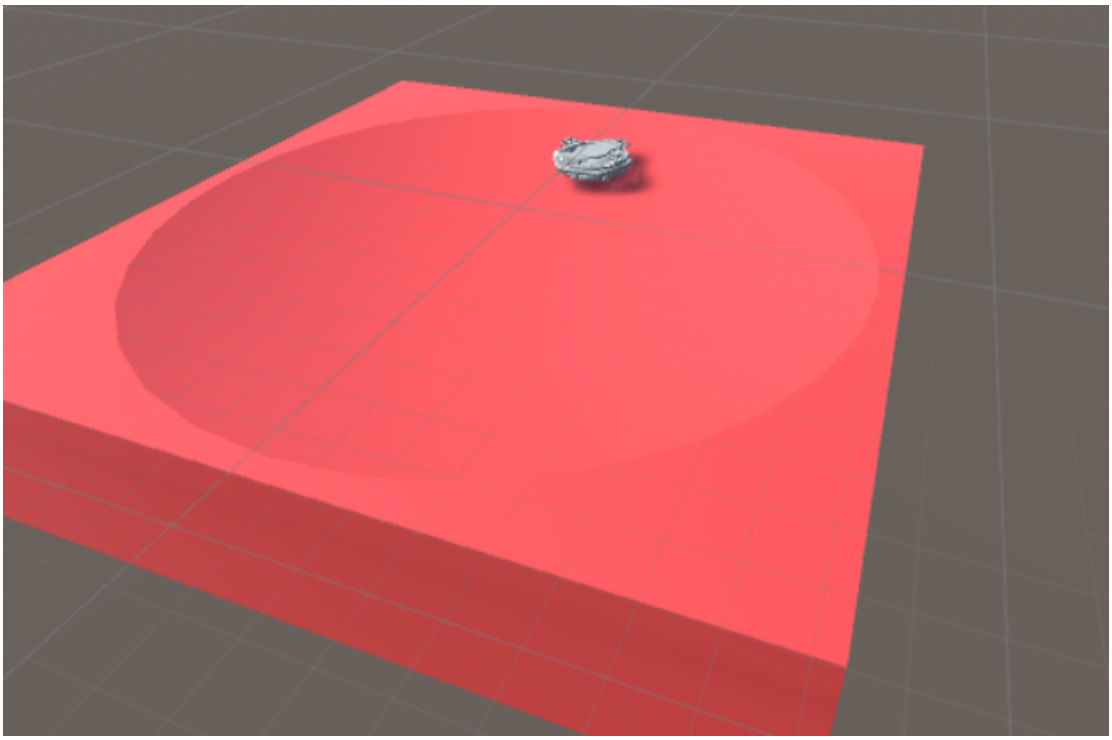
I 3.6 visas hur snurran representerades i OpenGL.



Figur 3.6: Roterande snurra i OpenGL

3.2.2 Unity

Det enda som ändrades i Unity var att snurran skulle kunna gå snyggt i en bana och för bra synlighet gjordes banan lite plattare än en halvsfär. Detta gör att snurran ser ut att närma sig mitten snabbare än vad den egentligen skulle göra. I 3.7 visas hur snurran simuleras i Unity.



Figur 3.7: Snurra simulerad i en bana

I Unity var det även tänkt att kollisionen skulle implementeras. Implementationen skulle ske med hjälp av Unitys funktioner som ska starta när ett objekt kolliderar med ett annat. Dock skapade detta mycket problem när det skulle implementeras och fungerade inte. Tiden för att fixa detta och få det att fungera fanns inte och därför finns ingen fungerande implementation av kollision även om ekvationer finns.

Kapitel 4

Diskussion

Modellen för friktion mot yta är satt med en friktionskoefficient för friktionen mellan de två typerna av material. Denna typ av modell är korrekt för när snurran är i rörelse men den täcker ej fallet när snurran börjar utan en translationshastighet. Detta är avsiktligt eftersom snurran har en starthastighet.

Det finns flera olika modeller för luftmotstånd med varierande komplexitet och lämplighet för denna simulering. Modellen vilken har använts för denna simulering ses i 2.5. Denna modell valdes för att metoden hanterar linjära samband vilka är enklare att implementera och tillräckligt god för att representera verklighet.

Att simulera rörelsen för ett föremål i en halvsfär med hjälp av en halvsfärisk pendel är diskuterbart. Det ger inte en korrekt bild av verkligheten att representera ett okopplat föremål med en modell för ett föremål kopplat via en pendels arm. Anledningen till att denna modell ändå har valts är för att visionen för hur snurran skulle kunna röra sig i en krökt bana valdes över korrektheten i detta fall. Felet mot verkligheten syns i att snurrans rörelse i banan tar orimligt lång tid för denna att nå halvsfärens botten. Att den rör sig i en cirkelformad rörelse i banan förekommer i verkligheten också. Dock i simulationen är detta fenomen inkorrekt i antal varv och hastighet.

För att få modellen av snurran att bete sig som en snurra gör i verkligheten krävs en modell över vobblet. Att den börjar snurra mindre rakt och tillslut ramlar ihop när vinkelhastigheten minskar. Detta utreddes i tre olika försök som har beskrivits i denna rapport. Tyvärr blev inget av försöken något som gruppen ville implementera i modellen. Första försöket där kända ekvationer för hur en snurra beter sig användes slutade med att snurran aldrig började vobbla eller föll ihop. Utan den fortsatte att snurra utan utan några störningar. Det andra försöket där ekvationerna gjordes om på vektorform för att beskriva relationerna i tre dimensioner gav samma resultat som det första försöket. Det tredje försöket där det gjordes egna kraftekvationer gav ett resultat något likt det verkliga systemet. Dock blandas rotationskrafter och translationskrafter rakt av utan någon direkt efterforskning. Tyvärr gav denna implementering inget vobbel, istället ramlar bara snurran ihop när när rörelsemängdsmomentet blir tillräckligt litet.

Anledningen till att Lagranges ekvationer [7] inte utforskades mer i detalj är helt enkelt för att det är svårt. Ekvationerna beskriver det allmänna sambandet mellan tre olika rotationer som snurran utsätts för. Att bryta ut vinkeln θ ur dessa ekvationer var inget gruppen ville ge sig på. Därför påbörjades försök tre istället.

Implementationen av kollisionen sågs från början vara möjlig, men efter flertal försök i Unity med funktioner som startas vid kollision av två objekt genomförts hanns inte bli färdigt. Funktionerna som implementerats startade aldrig och anledningen till detta kan vara att 3D-objekten inte hade en giltig form, att funktionerna implementerats på fel sätt eller att objekten hanterats fel i Unity.

Kapitel 5

Slutsats

Vi lyckades inte implementera och lösa samtliga problem vilka vi hade i vision att lyckas med. Men genom att prova på och misslyckats har förståelsen inför problemen ökats. Med de funktioner vilka visualiseras och har stöd i simuleringen är resultatet på en tillräckligt god nivå för att representera verklighet.

Att skapa translation i en halvsfärisk bana med hjälp av t.ex. fysikalisk motor i 3d och spelmotorer, hos ett objekt är enkelt. Dock att skapa en translation med bas i matematik och fysik var svårare än väntat. Det är också svårare att skapa något vilket är tänkt att representera hur det bör fungera och inte bara hur det kan se ut. Den halvsfäriska pendeln följer en verklighetstrogen bana men ställer kravet på att banan som snurran åker på måste vara av en perfekt sfär och eventuella krockar med snurran och banan bortses.

Arbetet i vidare sammanhang kan fokusera på att implementera vobblet, kollision och lutning beroende på underlaget vilket ej hanns med i denna tidsram. Grunden för matematiken för vobblet och kollision är framtagna medans lutning beroende på underlag ej är det.

Litteraturförteckning

[1] L. Ljung, T. Glad Modellbygge och simulering, Studentlitteratur, 2004, ISBN: 9789144024431

[2] www.tribology-abc.com,*Coefficient of friction*, www.tribology-abc.com,u.å.,[2018-02-12] Hämtad från:
http://www.tribology-abc.com/abc/cof.htm

[3] mordechai9, *Air resistance on rotating cylinder*,www.physicsforums.com,2010-04-28,[2018-01-22] Hämtad från:
https://www.physicsforums.com/threads/air-resistance-on-rotating-cylinder.333995/

[4] Richard Fitzpatrick *Spherical Pendulum*,The University of Texas at Austin,2011-03-31, [2018-02-28] , Hämtad från:
http://farside.ph.utexas.edu/teaching/336k/Newtonhtml/node82.html

[5] Wikipedia *List of moments of inertia*,Wikipedia,2018-01-27,[2018-03-01] Hämtad från:
`https://en.wikipedia.org/wiki/List_of_moments_of_inertia`

[6] Precession of Spinning Top, [2018-03-09] Hämtad från:
`http://hyperphysics.phy-astr.gsu.edu/hbase/top.html#top`

[7] Michael Hart, *Spinning Tops*, University of Surrey, 2004, [2018-03-09] Hämtad från:
http://www.maths.surrey.ac.uk/explore/michaelspages/Spin.htm

[8] Hecker, Chris *Physics, Part 3: Collision Response*,definition six,1997-03,[2018-03-09] Hämtad från:
http://chrishecker.com/images/e/e7/Gdmphys3.pdf

[9] DRACO, *BEYBLADE RPM (Revolutions Per Minute)*,worldbeyblade.org,2009-08-16 [2018-03-01] Hämtad från:
https://worldbeyblade.org/Thread-BEYBLADE-RPM-Revolutions-Per-Minute

[10] Robbie Gonzalez, *Why Do Wheels Sometimes Appear To Spin Backwards?*,io9,2018-01-27,[2018-03-01] Hämtad från:
https://io9.gizmodo.com/why-do-wheels-sometimes-appear-to-spin-backwards-1593807400

Bilaga A

Här presenteras ekvationer som användes för försök två. Implementationen sker i samma ordning som i försök ett.

$$I = \begin{bmatrix} \frac{1}{10}Mh^2 + \frac{3}{20}MR^2 & 0 & 0 \\ 0 & \frac{1}{10}Mh^2 + \frac{3}{20}MR^2 & 0 \\ 0 & 0 & \frac{3}{10}MR^2 \end{bmatrix} \tag{A.1}$$

$$\omega_z = \begin{bmatrix} \frac{\theta - x}{h} & \frac{x-\theta}{h} & 0 \\ \frac{\theta - x}{h} & \frac{\theta - x}{h} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} * \left[\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \right] \tag{A.3}$$

$$\begin{aligned} \tau_x &= I_{xx}\dot{\omega}_x - (I_{yy} - I_{zz})\omega_y\omega_z \\ \tau_y &= I_{yy}\dot{\omega}_y - (I_{zz} - I_{xx})\omega_z\omega_x \\ \tau_x &= I_{zz}\dot{\omega}_z - (I_{xx} - I_{yy})\omega_x\omega_y \end{aligned}$$

$$\bar{\tau}_{ext} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{A.4}$$

$$\dot{\omega} = \frac{\omega(i+h) - \omega(i)}{h} \tag{A.5}$$

$$\tau_z = I_{zz}\dot{\omega}_z = I_{zz}(\frac{\omega_z(i+h) - \omega_z(i)}{h}) => \bar{\tau}_{ext} = \begin{bmatrix} 0 \\ 0 \\ \tau_z \end{bmatrix} \tag{A.6}$$

$$\theta = sin^{-1}(\frac{\bar{\tau}_{ext}}{mgr}) \tag{A.7}$$

$$\omega_p = mgr(I\omega_s)^{-1} \tag{A.8}$$