Rotasjon av Stive Legemer

Prosjektoppgave i TDAT3024

Michael S. Larsen
Sabine Seljeseth
Marius Torbjørnsen
Sivert Utne

30. Oktober 2020

Innhold

1 Innledning **1**

2 Teori **2**

3 Metode **4**

Oppgave 1 4

Oppgave 2 4

Oppgave 3 5

Oppgave 4 6

Oppgave 5 6

Oppgave 6 6

4 Resultater **7**

Oppgave 1 7

Oppgave 2 7

Oppgave 3 8

Oppgave 5 8

Oppgave 6 8

5 Diskusjon og Konklusjon **11**

Hvorfor vil kulen være stabil? 11

Hvorfor har ikke t-nøkkelen lik energi for alle starthastighetene? 11

Hvorfor er den kinetiske energien ikke helt konstant? 11

Er t-nøkkelen stabil om alle aksene? 11

Hva er årsaken til ulikheten når vi løser likning $\frac{dX}{dt} = F(X) = X\Omega$ numerisk og konkret? 12

Tolkning av vinkelhastighetsgrafer 12

Animasjon av t-nøkkelen 12

Konklusjon 13

Referanser **14**

6 Vedlegg **15**

Vedlegg 1 - exponent_function.m 16

Vedlegg 2 - kinetic_energy.m 16

Vedlegg 3 - moment_of_inertia.m 16

Vedlegg 4 - RK4.m 17

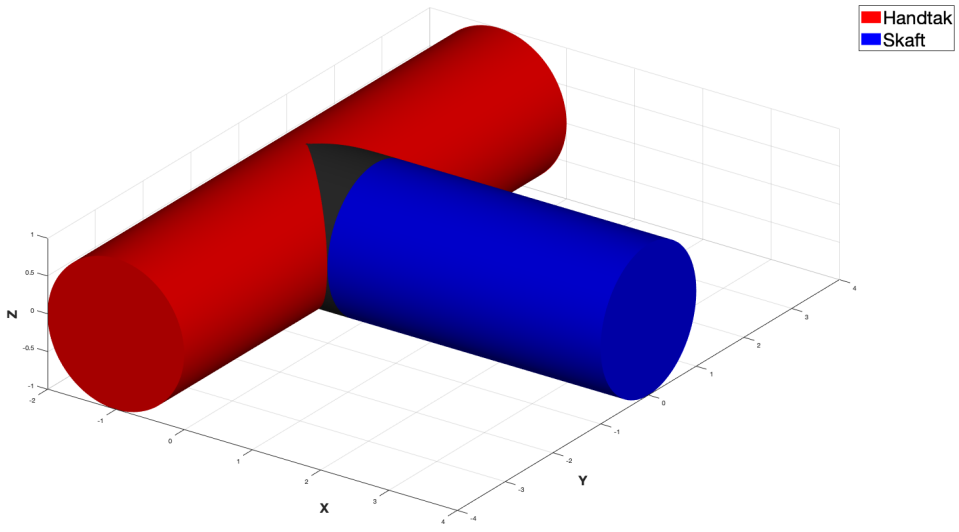
Vedlegg 5 - angular_velocity.m 17
Vedlegg 6 - derivate_of_X.m 17
Vedlegg 7 - euler_method.m 17
Vedlegg 8 - rotating_matrix.m 18
Vedlegg 9 - torque.m 18
Vedlegg 10 - plot_KineticEnergy.m 18
Vedlegg 11 - plot_AngularVelocity.m 18
Vedlegg 12 - draw_T_handle.m 19

7 Erklæring **22**

Michael S. Larsen 22
Sabine Seljeseth 22
Marius Torbjørnsen 22
Sivert Utne 22
Signaturer 22

1 Innledning

Oppgaven som skal løses er å modellere og undersøke bevegelsen til en roterende t-nøkkel. I beregningene vil vi ta utgangspunkt i t-nøkkelen beskrevet under.



Håndtak				**Skaft**			
Radius	=	$1.0cm$	(R_1)	Radius	=	$1.0cm$	(R_2)
Lengde	=	$8.0cm$	(L_1)	Lengde	=	$4.0cm$	(L_2)
Massetetthet	=	$6.7g/cm^3$	(ρ)	Massetetthet	=	$6.7g/cm^3$	(ρ)
Massesenter	=	$(-1.0x,\ 0y,\ 0z)$		Massesenter	=	$(2.0x,\ 0y,\ 0z)$	

For å kunne gjøre disse beregningene tar vi utgangspunkt i et koordinatsystem som roterer sammen med hovedaksene til treghetsmomentet til det roterende legemet. Vi setter opp bevegelseslikningene med hensyn på de roterende aksene og får ut komponentene som et system av 9 koblede differensiallikninger. Vi bruker så numeriske metoder (Euler og Runge Kutta) for å løse dette systemet, og derav beskrive bevegelsen til legemet. Med hjelp av disse resultatene vil vi undersøke Dzhanibekov effekten som vil oppstå når t-nøkkelen roterer, vi vil også se på dette i forhold til en kule som roterer, slik at vi kan sammenligne resultatene av stabil og ustabil rotasjon.

2 Teori

Dzhanibekov effekten, også kjent som tennisracket-teoremet eller mellomakse-teoremet, handler om et objekt med tre forskjellige akser, der hver akse har forskjellig treghetsmoment. Treghetsmomentet angir hvor mye energi som kreves for å rotere eller flytte objekt rundt hver akse [3]. De aksene med høyest og lavest treghetsmoment vil spinne stabilt. Den siste aksen, blir da den mellomliggende aksen og vil spinne ustabilt [5]. Denne effekten oppstår altså når man spinner et objekt rundt dens mellomliggende akse.

Den kinetiske rotasjons energien er gitt ved [4, p. 4],

$$K = \frac{1}{2}\overrightarrow{L} \cdot \overrightarrow{\omega_b} = \frac{1}{2} I \cdot \overrightarrow{\omega_b}^2 \tag{1}$$

Der I er treghetsmomentet og ω_b er vinkelhastighet i de forskjellige retningene.

$$\overrightarrow{L} = XI\overrightarrow{\omega_b} \tag{2}$$

$$\overrightarrow{\omega_b} = (XI)^{-1}\overrightarrow{L} \tag{3}$$

Formelen for L er uavhengig av hvilket koordinat system som blir brukt. Origio er i massesenteret. På grunn av dette velger vi et koordinatsystem som følger aksene til treghetsmomentet. X er da en 3×3 matrise hvor hvert element beskriver retningen til objektet og aksen. [4, p. 3, 5]

Likning 2 og 3 bruker treghetsmomentet I. Treghetsmomentet er gitt ved en 3×3 matrise der hvert element representerer treghetmomentet rundt retningsaksene for objektet.[4, p. 9-10]

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{4}$$

For t-nøkkelen har vi at treghetsmomentet rundt de forskjellige aksene er:

$$I_{xx} = \frac{M_1 R_1^2}{4} + \frac{M_1 L_1^2}{12} + \frac{M_1 R_2^2}{2} \tag{5}$$

$$I_{yy} = M_1 R_1^2 + \frac{M_2 L_2^2}{4} + \frac{M_1 R_1^2}{2} + \frac{M_2 R_2^2}{4} + \frac{M_2 L_2^2}{12} \tag{6}$$

$$I_{zz} = M_1 R_1^2 + \frac{M_2 L_2^2}{4} + \frac{M_1 R_1^2}{4} + \frac{M_1 L_1^2}{12} + \frac{M_2 R_2^2}{4} + \frac{M_2 L_2^2}{12} \tag{7}$$

$$I_{xy} = I_{xz} = I_{yz} = 0 \tag{8}$$

Her brukes verdiene for t-nøkkelen fra innledningen. M_1 og M_2 er henholdsvis massen til håndtaket og skaftet. Som blir regnet ut med:

$$M = V\rho = \pi R^2 L\rho \tag{9}$$

ω brukes for å finne Ω [4, p. 5]

$$\Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{10}$$

Ω blir brukt for å regne ut endringen til X.

$$\frac{dX}{dt} = F(X) = X\Omega \tag{11}$$

Eksponentfunksjonen blir brukt til å finne en tilnærming av X. Denne tilnærmingen blir brukt siden søylene i tilnærmingen er ortogonale, noe den fra likning 11 ikke er. [4, p. 5]

$$\exp(h\Omega) = Id_{3x3} + (1 - \cos|w|h)\frac{\Omega^2}{|w^2|} + (\sin|w|h)\frac{\Omega}{|w|} \tag{12}$$

Likning 12 stemmer kun dersom $X = \exp(h\Omega)$ multiplisert med X^T er lik identitetsmatrisen [4, p. 5]

$$X^T X = I_{d_{3x3}}$$

Eulers metode er en av metodene vi bruker for å løse differensial likninger. Konkret vil vi bruke formelen under i beregningene [4, p. 5],

$$\begin{aligned} W_0 &= X_0 \\ W_{i+1} &= W_i \exp(h\Omega_i) \end{aligned} \tag{13}$$

Ω_i regnes ut i fra likning 3 og 10 i hvert ledd.

Runge-Kutta er en annen metode for å løse differensial likninger. Denne metoden er mer nøyaktig enn Eulers metode. Vi bruker Runge-Kutta av 4. orden, også kalt RK4 [4, p. 7].

$$\begin{aligned} W_0 &= X_0 \\ \overrightarrow{\sigma_1} &= I^{-1} W_i^T \overrightarrow{L} \\ \overrightarrow{\sigma_2} &= I^{-1} \exp\Big(-(h/2)\,\Sigma_1 \Big) W_i^T \overrightarrow{L} \\ \overrightarrow{\sigma_3} &= I^{-1} \exp\Big(-(h/2)\,\Sigma_2 \Big) W_i^T \overrightarrow{L} \\ \overrightarrow{\sigma_4} &= I^{-1} \exp\Big(-h\Sigma_3 \Big) W_i^T \overrightarrow{L} \\ W_{i+1} &= W_i \exp\Big(\frac{h}{6}\big(\Sigma_1 + 2\Sigma_2 + 2\Sigma_3 + \Sigma_4\big) \Big) \end{aligned}$$

Eulers metode har en feil på $O(h^3)$. Dette vil si at feilen endrer seg relativt til steglengden h^3. Mens RK4 har en høyere grad og vil derfor avta raskere med steglengde $0 < h < 1$. [4, p. 6] [1, p. 317]

3 Metode

Oppgave 1

Her vil vi implementere ulike funksjoner vi vil bruke for å kunne regne på bevegelsen av et roterende legeme i rommet.

(a) Eksponentfunksjonen (ligning 12):
Funksjonen brukes til å kunne regne ut en estimering av $X(t+h)$. Fordelen av å bruke denne tilnærming er at den tilfredstiller likningen

$$\exp(h\Omega) \cdot \exp(h\Omega)^T = I$$

der I er identitetsmatrisen. Dette vil si at $X \cdot exp(h\Omega)$ har ortogonale søylevektorer.
Matlabfunksjon: *exponent_function.m* (vedlegg 1)

(b) Funksjon for energien til roterende legeme (ligning 1):
Funksjonen brukes til å dobbeltsjekke om energien er bevart ved en gitt endring i rotasjon over en tidsendring Δt.
Matlabfunksjon: *kinetic_energy.m* (vedlegg 2)

(c) Funksjon for beregning av treghetsmomentet til t-nøkkelen (ligning 4):
For å kunne regne ut endringen i vinkelhastighet over tid må vi også finne legemets treghetsmoment. Spesifikt for denne oppgaven bruker vi en t-nøkkel laget av to sylindere som er beskrevet i innledningen.
Matlabfunksjon: *moment_of_inertia.m* (vedlegg 3)

Oppgave 2

I denne oppgaven skal vi undersøke en kule ved å løse linkingene 3 og 11 eksakt.

Vi får oppgitt initialverdiene

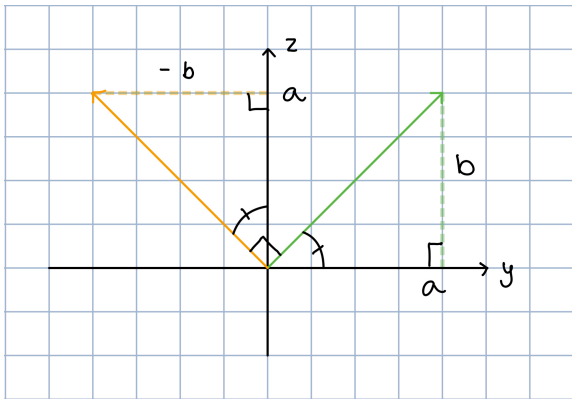
$$\overrightarrow{L} = \begin{bmatrix}1 & 0 & 0\end{bmatrix} \text{ og } X(0) = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \quad I = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

og setter dette inn i ligning 3

$$\overrightarrow{\omega_b} = I^{-1}X^T\overrightarrow{L} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$$

$$\Omega = \begin{bmatrix}0 & \omega_z & \omega_y\\ \omega_z & 0 & -\omega_x\\ -\omega_y & \omega_x & 0\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{bmatrix}$$

Vi vet at X er en matrise som tilhører den spesielle ortogonale gruppen. Altså at $X^TX = I$, om vi tegner disse opp vil X og X^T stå vinkelrett på hverandre. Videre vet vi ut i fra startbetingelsene at vi har en rotasjon rundt x-aksen. Setter vi opp dette i et system får vi følgene



Vi ser ut fra figuren at vinkelen mellom y-aksen og den grønne vektoren er lik vinkelen mellom z-aksen og den oransje vektoren. X-aksen vil peke ut av arket. Dette gir oss en lengde b, videre vil vi ha lengden på disse vektorene som vil være uttrykt med a. Dette gjør at vi kan forenkle X til $X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & -b \\ 0 & b & a \end{bmatrix}$.

$$X' = \begin{bmatrix} 1' & 0' & 0' \\ 0' & a' & -b' \\ 0' & b' & a' \end{bmatrix} = \Omega \cdot X \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & -b \\ 0 & b & a \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -b & a \\ 0 & a & -b \end{bmatrix}$$

Vi får altså fire differensiallikninger:

$$\begin{aligned} a' &= -b \\ -b' &= -a \\ b' &= a \\ a' &= -b \end{aligned}$$

Hvor to vil falle bort siden de er lik de andre, vi står da igjen med to ulike differensiallikninger:

$$\begin{aligned} &1\text{: } a' = -b \\ &2\text{: } b' = a \end{aligned}$$

som vi kan løse for å sette tilbake i X matrisen for å få den konkrete løsningen.

$$\begin{aligned} a' &= -b \\ \frac{da}{dt} &= -b \\ \frac{d^2b}{dt^2} &= -b \\ \frac{d^2b}{dt^2} + b &= 0 \\ r^2 + 1 &= 0 \\ r^2 &= -1 \\ r &= \pm\sqrt{-1} \\ r &= \pm i \end{aligned}$$

$$\begin{aligned} b' &= a \\ b(0) &= 0 \\ \frac{db}{dt} &= a \\ a &= \frac{db}{dt} \end{aligned}$$

$$\begin{aligned} a &= -C_1 \cdot \sin t + C_2 \cdot \cos t \\ b &= C_1 \cdot \cos t + C_2 \cdot \sin t \end{aligned}$$

Oppgave 3

Implementering av Eulers metode: I denne implementeringen bruker vi en tilpasset versjon av algoritmen for å kunne ta i bruk eksponent funksjonen fra 1a) slik at vi kan få ortogonale W_i.

Metoden er da bygget opp slik som likning 13.

Matlabfunksjon: *euler_method.m* (vedlegg 7)

Oppgave 4

Her har vi laget en funksjon som tar inn tidsintervallet, X-en vi har i starten, steglengden h, treghetsmomentet I og dreiemomentet L. For hver gang funksjonen kjører RK4-steget vil den legge til resultatet i en matrise som inneholder alle X-verdiene vi får. Denne brukes senere i oppgaven til å tegne rotasjonen av t-nøkkelen.

Matlabfunksjon: *RK4.m* (vedlegg 4)

Oppgave 5

Bruker implementasjonen av RK4 fra oppgave 4) til å løse systemet i likning 11, der $X(0) = Id_{3\times 3}$, I er lik treghetsmomentet fra oppgave 1c) og vinkelhastighetene er

$$
\begin{aligned}
a)\ \ \overrightarrow{\omega}(0) &= \begin{bmatrix} 1 & 0.05 & 0 \end{bmatrix}^T \\
b)\ \ \overrightarrow{\omega}(0) &= \begin{bmatrix} 0 & 1 & 0.05 \end{bmatrix}^T \\
c)\ \ \overrightarrow{\omega}(0) &= \begin{bmatrix} 0.05 & 0 & 1 \end{bmatrix}^T
\end{aligned}
$$

De oppgitte verdiene for vinkelhastighet ω ved tid 0 har da lengde omtrent lik 1.

Oppgave 6

Her skal vi visualisere resultatene fra oppgave 5, hvor vi tegner grafer for de forskjellige komponentene som en funksjon av tiden. Vi skal deretter tolke disse grafene og forklare hva de representerer. For å visualisere vinkelhastighetene i ω_b bruker vi likning 3. Vi får da plottet vinkelhastighet rundt de forskjellige aksene i ω_b, og kan deretter plotte over tidsintervallet Δt.
Matlabfunksjon: *plot_AngularVelocity.m* (vedlegg 11)

Her vil vi også se på den totale kinetiske enrgien for t-nøkkelen gjennom rotasjonen for å se om energien er bevart.
Matlabfunksjon: *plot_KineticEnergy.m* (vedlegg 10)

Til slutt animerer vi t-nøkkelen for å få et klart bilde over hvordan de forskjellige vinkelhastighetene fra oppgave 5 påvirker rotasjonen.

4 Resultater

På alle oppgavene er det brukt $h = 0.0001$

Oppgave 1

(a) Regner ut X fra likning 12 der $\omega = \begin{bmatrix}1 & 6 & 5\end{bmatrix}$. Får da

$$X = \begin{bmatrix} 1.0000 & -0.0005 & 0.0006 \\ 0.0005 & 1.0000 & -0.0001 \\ -0.0006 & 0.0001 & 1.0000 \end{bmatrix} X^T = \begin{bmatrix} 1.0000 & 0.0005 & -0.0006 \\ -0.0005 & 1.0000 & 0.0001 \\ 0.0006 & -0.0001 & 1.0000 \end{bmatrix}$$

$$XX^T = \begin{bmatrix} 1.000000609908922 & 1.900150004074122 \times 10^{-7} & 3.099899923979417 \times 10^{-7} \\ 1.900150004074122 \times 10^{-7} & 1.000000259875962 & -2.900149160132748 \times 10^{-7} \\ 3.099899923979417 \times 10^{-7} & -2.900149160132748 \times 10^{-7} & 1.000000370214956 \end{bmatrix}$$
$$\approx Id_{3x3}$$

(b) Treghetmomentet I definert i 4 blir regnet ut i fra likningene 5 til 8. Massene M_1 og M_2 for håndtaket og skaftet finner vi med ligning 9

$$\text{Håndtaket: } M_1 = 8\pi\rho \approx 168.39\ gram$$
$$\text{Skaftet: } M_2 = 4\pi\rho \approx 84.19\ gram$$

Vi får da at treghetsmomentet for t-nøkkelen er:

$$I = \begin{bmatrix} 982.271303022409 & 0 & 0 \\ 0 & 722.671030080772 & 0 \\ 0 & 0 & 1578.650308428871 \end{bmatrix}$$

Oppgave 2

Med uttrykket vi fant i metodedelen for oppgave 2

$$a = -C_1 \cdot \sin t + C_2 \cdot \cos t$$
$$b = C_1 \cdot \cos t + C_2 \cdot \sin t$$

Setter vi inn initialbetingelsene $b(0) = 0$ og $a(0) = 1$ for å finne C_1 og C_2.

$$\begin{aligned} C_1 \cdot \cos 0 + C_2 \cdot \sin 0 &= 0 \\ C_1 \cdot 1 + C_2 \cdot 0 &= 0 \\ C_1 &= 0 \end{aligned}$$

$$\begin{aligned} C_1 \cdot \sin t + C_2 \cdot \cos t &= 1 \\ C_1 \cdot \sin 0 + C_2 \cdot \cos 0 &= 1 \\ C_1 \cdot 0 + C_2 \cdot 1 &= 1 \\ C_2 &= 1 \end{aligned}$$

Vi står altså igjen med uttrykkene

$$b = \sin t$$
$$a = \cos t$$

Om vi setter dette inn i uttrykket for X når $t = 1$ får vi

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & a & -b \\ 0 & b & a \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos t & -\sin t \\ 0 & \sin t & \cos t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5403023059 & -0.8414709848 \\ 0 & 0.8414709848 & 0.5403023059 \end{bmatrix}$$

Oppgave 3

Vi bruker verdiene vi fikk i oppgave 2 og løser likningen ved hjelp av implementasjonen av eulers metode på tidsintervallet 0 til 1s.

$$X(1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.540329326917989 & -0.841470981430706 \\ 0 & 0.84155132736037 & 0.540329326917991 \end{bmatrix}$$

Sammenlikner vi dette med resultatet med resultatet fra oppgave 2 får vi en forskjell som er

$$X_{eksakt} - X_{tilnærmet} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -0.270210498491963 \times 10^{-4} & -0.000033771903052 \times 10^{-4} \\ 0 & -0.841479281403767 \times 10^{-4} & -0.270210498514167 \times 10^{-4} \end{bmatrix}$$

Oppgave 5

På alle disse utregningene bruker vi tidsintervallet 0 til 100s. Vi bruker RK4 med følgende start-vinkelhastighet og finner $X(100)$:

(a)

$$\omega(0) = [1, 0.05, 0]^T$$

$$X(100) = \begin{bmatrix} 1.0001 & -0.0068 & -0.0094 \\ 0.0102 & 0.8640 & 0.5134 \\ 0.0046 & -0.5136 & 0.8640 \end{bmatrix}$$

(b)

$$\omega(0) = [0, 1, 0.05]^T$$

$$X(100) = \begin{bmatrix} 0.9713 & 0.0038 & -0.2558 \\ -0.0340 & 0.9995 & 0.0032 \\ 0.2558 & 0.0045 & 0.9718 \end{bmatrix}$$

(c)

$$\omega(0) = [0.05, 0, 1]^T$$

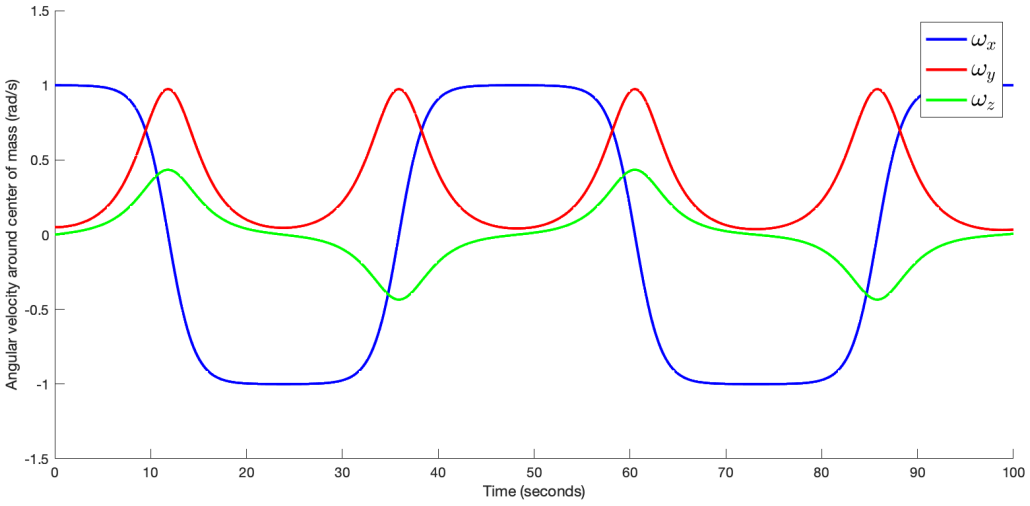
$$X(100) = \begin{bmatrix} 0.9017 & 0.4398 & 0.0589 \\ -0.4399 & 0.9036 & -0.0138 \\ -0.0590 & -0.0135 & 0.9982 \end{bmatrix}$$

Oppgave 6

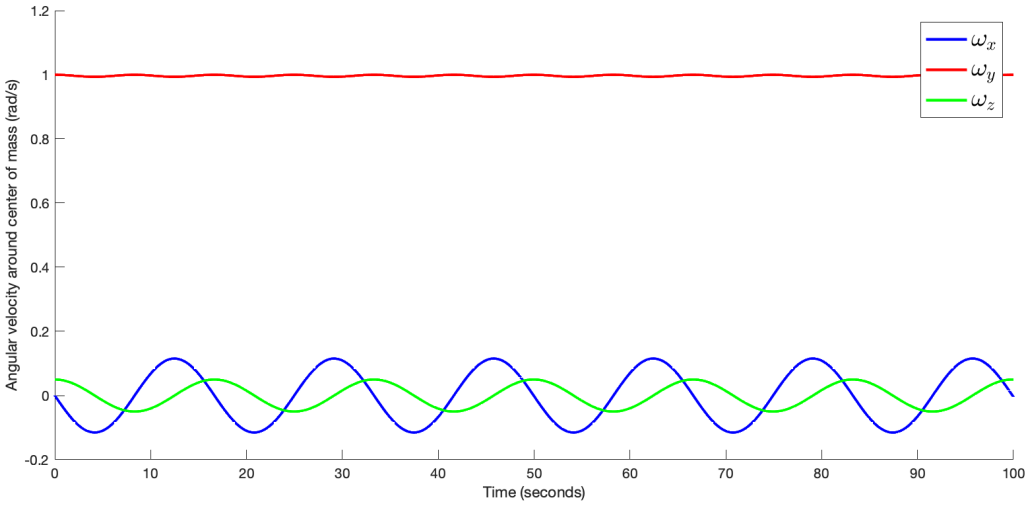
Plotting av vinkelhastighetene fra oppgave 5. Samt den totale kinetiske rotasjonsenergien for t-nøkkelen mens den roterer.

Animasjonene av t-nøkkelen for de forskjellige vinkelhastighetene er lagt ved. Se:
Animasjon: *Animasjon_A.gif* (vedlegg 14)
Animasjon: *Animasjon_B.gif* (vedlegg 15)
Animasjon: *Animasjon_C.gif* (vedlegg 16)

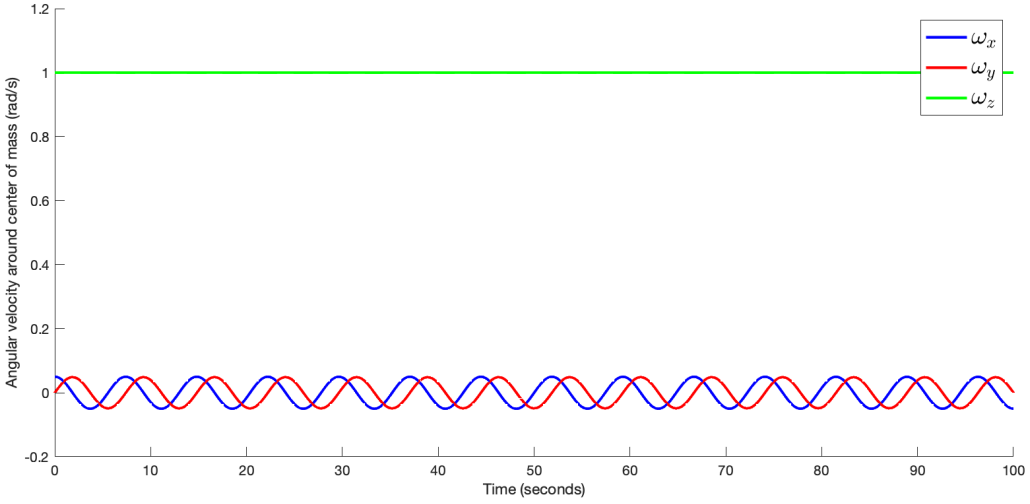
Figur 1: Rotasjon rundt X-aksen



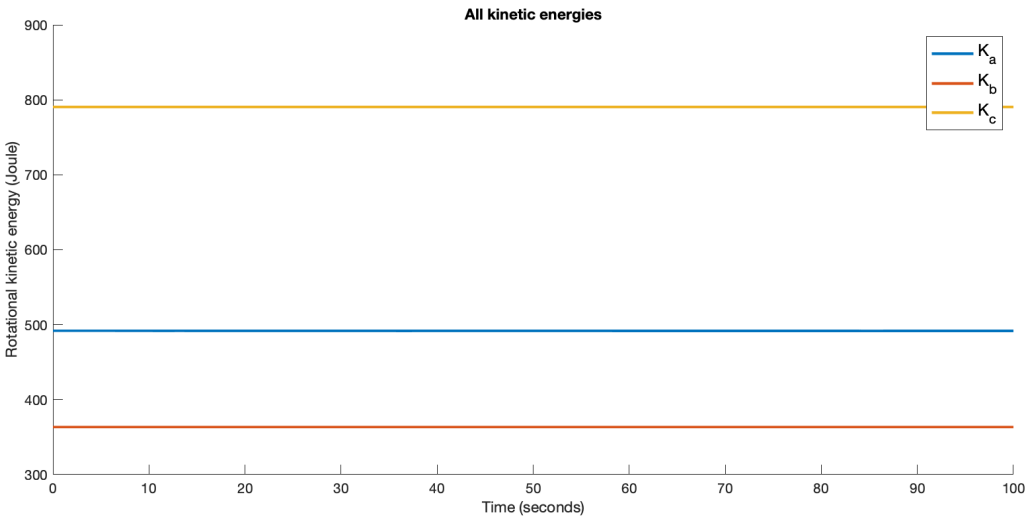
Figur 2: Rotasjon rundt Y-aksen



Figur 3: Rotasjon rundt Z-aksen



Figur 4: Endring i kinetisk energi



5 Diskusjon og Konklusjon

Hvorfor vil kulen være stabil?

I motsetning til t-nøkkelen vi kula ha samme treghetsmoment rundt alle aksene, siden massen er like langt unna alle aksene. Dette betyr at uansett hvordan akse vi velger å rotere kula rundt, vil det kreve den samme mengden energi. Dette gjør at vi ikke vil ha noen mellomliggende akse, slik som t-nøkkelen har, og dermed vil ikke vi oppleve Dzhanibenkov effekten med kula. Den vil rotere stabilt.

Hvorfor har ikke t-nøkkelen lik energi for alle starthastighetene?

Det er lett å tenke at energien T-nøkkelen har burde være lik for alle de ulike starthastighetene vi har gitt den, det kan derfor virke rart at energien vi har funnet for de forskjellige tilfellene ikke er like. Dette kan vi se på Figur 4. Det er her treghetsmomentet kommer inn. Jo høyere treghetsmomentet er for en akse, jo mer energi kreves for å akselerere objektet rundt denne aksen. Dette kan man observere ved f.eks. å sitte i en snurrende kontorstol. Ved å å strekke ut eller eller trekke inn benene er det mulig å endre treghetsmomentet for systemet (stolen med person). Med utstrakte bein snurrer stolen saktere enn ved sammentrukne bein. Du kan altså endre hastigheten din uten å tilføre energi til systemet. Det er derfor vi ser at den kinetiske energien er ulik for de forskjellige starthastighetene rundt aksene, til tross for at størrelsen til hastighetene er like.

Hvorfor er den kinetiske energien ikke helt konstant?

Ifølge Newton's første lov, også kjent som *treghetsloven* vil et objekt som ikke blir påvirket av ytre krefter beholde sin kinetiske energi. I dette forenklede systemet er det ikke tatt hensyn til noen ytre krefter, her kunne det vært relevant å inkludere for eksempel luftmotstand under rotasjonen, varme el. Det betyr at energien for dette systemet er bevart og må derfor være konstant. Denne energien er bestemt av starthastigheten vi gir T-nøkkelen, da vi kan se for oss at T-nøkkeln ligger stille i rommet og så tilfører vi nøkkelen energi ved å gi den hastighet rundt de forskjellige aksene.

Resultatet derimot forteller oss at energien for t-nøkkelen endrer seg noe over tid. Dette er forsåraket av de numeriske metodene vi bruker til å løse systemet.

Er t-nøkkelen stabil om alle aksene?

Ut i fra treghetsmomententet vi finner i oppgave 1c) ser vi at det varierer for de forskjellige aksene.

$$I = \begin{bmatrix} 982.271303022409 & 0 & 0 \\ 0 & 722.671030080772 & 0 \\ 0 & 0 & 1578.650308428871 \end{bmatrix}$$

Vi ser at vi har det største treghetsmomentet er rundt z-aksen og det minste er rundt y-aksen. Dette vil si at massen til t-nøkkelen er distribuert nærmest y-aksen og lengst unna z-aksen. Treghetsmomentet rundt x-aksen er et sted i mellom. Vi vet ifølge *intermediate axis theoremet* at den mellomliggende treghetsmomentet, treghetsmometet til x-aksen i dette tilfelle, vil forårsaket en ustabil rotasjon om aksen. Dette betyr at vi vil ha en stabil rotasjon rundt y- og z-aksen, mens vi vil kunne observere at t-nøkkelen flipper om den roterer rundt x-aksen. Dette kan vi se i grafen i oppgave 6 a) der den vil bevege seg fra 1 til -1. Den er dermed ikke stabil rundt alle aksene.

Hva er årsaken til ulikheten når vi løser likning $\frac{dX}{dt} = F(X) = X\Omega$ numerisk og konkret?

Grunnen til at vi får en forskjell mellom den eksakte løsningen og den numeriske tilnærmingen er at numeriske løsninger ikke er helt eksakte. Vi forventer at denne forskjellen mellom den eksakte løsningen og vår tilnærming er som beskrevet i teoridelen med en feil av $O(h^3)$ ved bruk av Eulers metode.

Tolkning av vinkelhastighetsgrafer

Vinkelhastighetene er relativt til posisjonen til T-nøkkelen, som er til en hver tid beskrevet med X matrisen.

ω_x beskriver hvor fort T-nøkkelen roterer rundt sin x-akse

ω_y beskriver hvor fort T-nøkkelen roterer rundt sin y-akse

ω_z beskriver hvor fort T-nøkkelen roterer rundt sin z-akse

Når vi undersøker grafen fra oppgave 6b) (Figur 2) som i hovedsak har startvinkelhastighet rundt y-aksen. Ser vi at rotasjonen om y-aksen holder seg tilnærmet helt stabilt. De to andre vinkelhastighetene holder seg relativt stabile med litt svingning. Det samme skjer i oppgave 6c) (3) der hovedrotasjonen er om z-aksen.

Når vi derimot ser på grafen fra oppgave 6a) (Figur 1) som har en startvinkelhastighet i hovedsak rundt x-aksen. Ser vi at rotasjonen om alle aksene har en stor endring over tid. Spesielt ser vi at rotasjonen om x-aksen beveger seg fra +1 til -1 rad/s. Om vi sammenlikner med de to andre grafene ser vi at denne ustabile rotasjonen er spesiel for rotasjon om x-aksen.

Grunnen til at rotasjonen om x-aksen gir oss en ustabil rotasjon er at dette er aksen med det mellomliggende treghetsmomentet. Vi observerer derfor at T-nøkkelen beveger seg i henhold til Dzhanibekov effekten, som var forventet ut ifra teorien.

Animasjon av t-nøkkelen

Vi forsøkte å få til en korrekt animasjon av t-nøkkelen mens den roterer, men møtte på flere problemer. Dette besto hovedsakelig i beregning av vinkler, og hvordan rotasjonen rundt aksene påvirker hverandre. Målet var å få korrekt rotasjon rundt hver akse slik at t-nøkkelen havner i riktig posisjon, altså i samme posisjon som den beregnede X-matrisen. Dersom det hadde vært lenger tid, eller dersom vi hadde brukt en annen måte til å animere t-nøkkelen kunne vi kanskje fått animasjonen helt korrekt.

Animsjonene som er vedlagt er korrekt med tanke på rotasjonen rundt y og z-aksen, farget rød og grønn på figurene og i animasjonen. Det som ikke stemmer helt, er rotasjonen rundt x-aksen, farget blått. Mer spesifikt ser vi i animasjonen for oppgave 5 a) (Figur 1, Vedlegg 14) at håndtaket i rødt, som er det vi observerer rotere rundt x-aksen har feil retning dersom rotasjonen rundt z-aksen er mer enn 90 grader, samt at den "hopper"mellom posisjoner ved spesifikke kombinasjoner av vinkler. Dette er altså et resultat av måten vi beregner rotasjonsvinklene og den grafiske rotasjonen rundt aksene, og ikke et problem med de beregnede posisjonene. Dette skjer også for animasjonen for oppgave 5 b) (Figur 2, Vedlegg 15), mens animasjonen for oppgave 5 c) (Figur 3, Vedlegg 16) er korrekt rundt alle aksene. Animasjonene er forøvrig raskere enn den reelle rotasjonen, ett sekund i animasjontiden tilsvarer fem sekunder på figurene og i virkeligheten.

Konklusjon

Etter å ha jobbet med prosjektet har vi fått et bedre innblikk i hvordan treghetsmomentet og massefordelingen kan påvirke rotasjonen til et legme. Vi har sett på hvordan Dzhanibekov effekten oppstår i rotasjonen til t-nøkkelen. I tilegg har vi fått sett på hvordan vi kan bruke numeriske metoder for å løse problemstillinger i hverdagen. Om vi skulle ha gjort noe annerledes hadde vi programmert i python. Hadde vi hatt mer tid kunne vi fått finjustert animasjonen til t-nøkkelens rotasjon. Videre kunne vi ha undersøkt flere stive legemer og hvordan ytre krefter ville påvirket rotasjonene.

Referanser

[1] Sauer, T. 2012. *Numerical Analysis (2. utg.)*. USA: Pearson Education Limited.

[2] *Rotation Matrix*. Wikipedia.com
https://en.wikipedia.org/wiki/Rotation_matrix.
(lest 2020, 28.Oktober).

[3] *Moment of Inertia*. Brilliant.org.
https://brilliant.org/wiki/calculating-moment-of-inertia-of-point-masses/.
(lest 2020, 24.Oktober).

[4] H. J. Rivertz. *ROTASJON AV STIVE LEGEMER* 15. Oktober 2020

[5] *Understanding the Dzhanibekov Effect*. Engineeringclicks.
https://www.engineeringclicks.com/dzhanibekov-effect/
(Lest 2020, 28 Oktober)

6 Vedlegg

Vedlegg nr	Navn	Beskrivelse
1	exponent_function.m	Beregne likning 12
2	kinectic_energy.m	Beregne likning 1
3	moment_of_inertia.m	Beregne likning 4
4	RK4.m	Løse systmer med bruk av Runge kutta
5	angular_velocity.m	Beregne likning 3
6	derivative_of_X.m	Beregne likning 11
7	euler_method.m	Løse likninger med eulers metode 4
8	rotating_matrix.m	Matlabfunksjon for å gjøre ω til Ω
9	torque.m	Beregne dreiemomentet L
10	plot_KineticEnergy.m	Plotte kinetiske energi over tid
11	plot_AngularVelocity.m	Plotte vinkelhastighet over tid
12	draw_T_handle.m	Tegne opp T-nøkkelen
13	Oppgaver.mlx	Livescript med utregningene til prosjektet
14	Animasjon_A.gif	Animasjon av rotasjon rundt x-aksen
15	Animasjon_B.gif	Animasjon av rotasjon rundt y-aksen
16	Animasjon_C.gif	Animasjon av rotasjon rundt z-aksen

Vedlegg 1 - exponent_function.m

```
function X = exponent_function(h, ohm)
I = eye(3);
w = norm([ohm(2,3) -ohm(1,3) ohm(1,2)]);

X = I + (1 - cos(w * h)) * (ohm.^2/w^2) + sin(w * h) * (ohm/w);
end
```

Vedlegg 2 - kinetic_energy.m

```
function K = kinetic_energy(I, w)
% Calculates the total rotational kinetic energy of the object
%    I = moment of inertia around each axis
%    w = angular velocity around center of mass

I_x = I(1,1);
I_y = I(2,2);
I_z = I(3,3);

w_x = w(1);
w_y = w(2);
w_z = w(3);

K = 1/2 * (I_x * w_x^2 + I_y * w_y^2 + I_z * w_z^2);
end
```

Vedlegg 3 - moment_of_inertia.m

```
function I = moment_of_inertia(R1, L1, R2, L2, p)
M1 = pi * R1^2 * L1 * p;
M2 = pi * R2^2 * L2 * p;

I_xx = ((M1*R1^2)/4) + ...
       ((M1*L1^2)/12) + ...
       ((M2*R2^2)/2);

I_yy = (M1*R1^2) + ...
       ((M2*L2^2)/4) + ...
       ((M1*R1^2)/2) + ...
       ((M2*R2^2)/4) + ...
       ((M2*L2^2)/12);

I_zz = (M1*R1^2) + ...
       ((M2*L2^2)/4) + ...
       ((M1*R1^2)/4) + ...
       ((M1*L1^2)/12) + ...
       ((M2*R2^2)/4) + ...
       ((M2*L2^2)/12);

I = [[I_xx   0     0 ];
     [  0  I_yy    0 ];
     [  0    0   I_zz]];
end
```

Vedlegg 4 - RK4.m

```
function [W, X_all] = RK4(time_interval, X0, h, I, L)
W    = X0;
n    = (time_interval(2) - time_interval(1)) / h;

X_all = cell(1,n); % save all instances of X matrix for animation/plotting
for i = 1:n
    W = rk4step(W, h, I, L);
    X_all{i} = W;
end

function W = rk4step(W, h, I, L)
sigma1 = I \ W' * L;
SIGMA1 = rotating_matrix(sigma1);

sigma2 = I \ exponent_function((-h/2), SIGMA1) * W' * L;
SIGMA2 = rotating_matrix(sigma2);

sigma3 = I \ exponent_function((-h/2), SIGMA2) * W' * L;
SIGMA3 = rotating_matrix(sigma3);

sigma4 = I \ exponent_function(-h, SIGMA3) * W' * L;
SIGMA4 = rotating_matrix(sigma4);

W = W * exponent_function(h/6, (SIGMA1 + (2 * SIGMA2) + (2 * SIGMA3) + SIGMA4));
```

Vedlegg 5 - angular_velocity.m

```
function w = angular_velocity(X, I, L)
w = (X * I) \ L;
end
```

Vedlegg 6 - derivate_of_X.m

```
function dX = derivative_of_X(X, omega)
dX = X * omega;
end
```

Vedlegg 7 - euler_method.m

```
function W = euler_method(time_interval, X0, h, I, L)

W = X0;
n = (time_interval(2) - time_interval(1)) / h;

for i = 1:n
   W = eulerstep(W, h, I, L);
end

function W = eulerstep(W, h, I, L)
wb = I \ W' * L;
omega = rotating_matrix(wb);
W = W * exponent_function(h, omega);
```

```
end
end
```

Vedlegg 8 - rotating_matrix.m

```
function ohm = rotating_matrix(w)
w_x = w(1);
w_z = w(2);
w_y = w(3);

ohm = [[  0   -w_y   w_z];
       [ w_y    0   -w_x];
       [-w_z   w_x    0 ]];
end
```

Vedlegg 9 - torque.m

```
function L = torque(X, I, w)
L = X * I * w;
end
```

Vedlegg 10 - plot_KineticEnergy.m

```
function plt = plot_KineticEnergy(X, interval, I, L)
%plot_KineticEnergy plots the total kinetic energy for the T-Handle
%   X = a list of X matricies for each time step
%       over the time interval 'interval'
%   I = moment of inertia for  the T-handle
%   L = torque for the t-handle

n = length(X);
h = (interval(2) - interval(1)) / n;
t = interval(1) + h : h : interval(2);

K = zeros(1, n);
for i = 1:n
    wb = angular_velocity(X{i}, I, L);
    K(i) = kinetic_energy(I, wb);
end

plt = plot(t, K,'linewidth', 2);
xlabel('Time (seconds)');
ylabel('Rotational kinetic energy (Joule)');
end
```

Vedlegg 11 - plot_AngularVelocity.m

```
function plot_AngularVelocity(X, interval, I, L)
%plots the angular velocity around x, y and z axis
%   X = a list of X matricies for each time step
%   over the time interval 'interval'
%   I = moment of inertia for t-handle
%   L = torque for t-handle
```

```
n = length(X);
h = (interval(2) - interval(1)) / n;
t = interval(1) + h : h : interval(2);

wb_x = zeros(1, n);
wb_y = zeros(1, n);
wb_z = zeros(1, n);

for i = 1 : n
    wb = (X{i}*I) \ L;
    wb_x(i) = wb(1);
    wb_y(i) = wb(2);
    wb_z(i) = wb(3);
end

figure
hold on
plot(t, wb_x, 'blue', 'LineWidth', 2)
plot(t, wb_y, 'red', 'LineWidth', 2)
plot(t, wb_z, 'green', 'LineWidth', 2)
hold off
xlabel('Time (seconds)');
ylabel('Angular velocity around center of mass (rad/s)');
legend({'${\omega_{x}}$', '${\omega_{y}}$', '${\omega_{z}}$'}, ...
        'Interpreter','latex', 'FontSize', 18)
end
```

Vedlegg 12 - draw_T_handle.m

```
function draw_T_handle(init_Handle, init_Shaft)
%% Fetch inital values

R_handle = init_Handle(1);
L_handle = init_Handle(2);
c_handle = [init_Handle(3) init_Handle(4) init_Handle(5)];

R_shaft = init_Shaft(1);
L_shaft = init_Shaft(2);
c_shaft = [init_Shaft(3) init_Shaft(4) init_Shaft(5)];

%% Set drawing specifications

color_handle = [255, 1, 29]/255;
color_shaft = [1, 29, 255]/255;
color_connection = [46 52 58]/255;
numberOfEdges = 1000;

%% Draw T-handle in correct position

HANDLE = draw_Sylinder(R_handle, ...
                       L_handle, ...
                       c_handle, ...
                       color_handle, ...
                       numberOfEdges);
hold on
rotate(HANDLE, [1 0 0], 90)
```

```
SHAFT = draw_Sylinder(R_shaft, ...
                      L_shaft, ...
                      c_shaft, ...
                      color_shaft, ...
                      numberOfEdges);
rotate(SHAFT, [0 1 0], 90, [c_shaft(1) L_shaft/2 0])

center_connection = [c_shaft(1), ...
                     c_shaft(2), ...
                     c_shaft(3)-L_shaft/2-R_handle/2];
CONNECTION = draw_Sylinder(R_shaft, ...
                           R_handle, ...
                           center_connection, ...
                           color_connection, ...
                           numberOfEdges);
rotate(CONNECTION, [0 1 0], 90, [c_shaft(1) L_shaft/2 0])

% close end of the sylinders
draw_Circle([c_handle(1), c_handle(2)+L_handle/2, c_handle(3)], ...
                [0 1 0], R_handle, color_handle);
draw_Circle([c_handle(1), c_handle(2)-L_handle/2, c_handle(3)], ...
                [0 1 0], R_handle, color_handle);
draw_Circle([c_shaft(1)+L_shaft/2, c_shaft(2), c_shaft(3)], ...
                [1 0 0], R_handle, color_shaft);
hold off

%% Camera and plot specifications

light('position', [5 -5 5])
lighting gouraud
view([35 25])
axis equal

legend({'Handtak','Skaft'},'fontsize',14)
xlabel('X','fontweight','bold','fontsize',14);
ylabel('Y','fontweight','bold','fontsize',14);
zlabel('Z','fontweight','bold','fontsize',14);
end

%% Sub-Functions
function sylinder = draw_Sylinder(radius, length, center, color, n)
[x, y, z] = cylinder(radius, n);
x = center(1) + x;
y = center(2) + y;
z = center(3) + z * length - length/2;
sylinder = surf(x,y,z, 'facecolor', color,'edgecolor','none');
end

function circle = draw_Circle(center, normal, radius, color)
theta=0:0.01:2*pi;
v=null(normal);
points=repmat(center',1,...
                  size(theta,2))+radius*(v(:,1)*cos(theta)+v(:,2)*sin(theta));
circle = fill3(points(1,:), ...
               points(2,:), ...
               points(3,:), ...
               color, 'edgecolor', 'none');
```

```
end
```

7 Erklæring

Michael S. Larsen

Jeg, Michael S. Larsen, erklærer at jeg har vært med på samarbeidet på dette prosjektet og har bidratt konstruktivt i besvarelsen.

Sabine Seljeseth

Jeg, Sabine Seljeseth, erklærer at jeg har vært med på samarbeidet på dette prosjektet og har bidratt konstruktivt i besvarelsen.

Marius Torbjørnsen

Jeg, Marius Torbjørnsen, erklærer at jeg har vært med på samarbeidet på dette prosjektet og har bidratt konstruktivt i besvarelsen.

Sivert Utne

Jeg, Sivert Utne, erklærer at jeg har vært med på samarbeidet på dette prosjektet og har bidratt konstruktivt i besvarelsen.

Signaturer

Michael S. Larsen — 30/10-20

Michael S. Larsen — Dato

Sabine Seljeseth — 30/10 - 20

Sabine Seljeseth — Dato

Marius Torbjørnsen — 30.10.2020

Marius Torbjørnsen — Dato

Sivert Utne — 30/10-20

Sivert Utne — Dato