CHALMERS | GÖTEBORGS UNIVERSITET


# Rotationer och kvaternioner

*Examensarbete för kandidatexamen i matematik vid Göteborgs universitet*

Damiano Ognissanti
Edvin Wedin
Niklas Andersson

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2012

# Rotationer och kvaternioner

*Examensarbete för kandidatexamen i matematik inom matematikprogrammet vid Göteborgs universitet*

Damiano Ognissanti  Edvin Wedin

*Examensarbete för kandidatexamen i tillämpad matematik inom matematikprogrammet vid Göteborgs universitet*

Niklas Andersson

Handledare: 张根凯, Genkai Zhang
Examinator: Carl-Henrik Fant
Omslagsbild: John Tenniel, 1865 [2]

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2012

**Sammanfattning**

Rotationer i 3-dimensionella rum lärs vanligtvis ut i form av rotationsmatriser. Ett alternativ till rotationsmatriser är kvaternioner, vilka kan ses som en utvidgning av de komplexa talen. Genom att representera vektorer som kvaternioner kan rotationer beräknas med en formel kortare och enklare att memorera än mosvarande rotationsmatris. I rapporten härleder vi denna formel och diskuterar olika sätt att representera rotationer. Avslutningsvis ligger en numerisk del där beräkningshastighet och stabilitet jämförs mellan kvaternioner och matriser.

**Abstract**

Rotations in 3 dimensions are commonly taught by means of matrices. An alternative is using quaternions, which can be seen as an extension of the complex numbers. By representating 3-dimensional vectors as imaginary quaternions, rotations can be computed by shorter formulas, easy to remember compared to its corresponding rotation matrix. In this report we will derive those formulas and discuss different ways to represent rotations. At the end is a numerical part, where computational speed and stability is compared between matrices and quaternions.

# Notationer

| Beteckning | Betydelse |
|---|---|
| $\mathbb{N} = \{0, 1, 2, 3, ...\}$ | Mängden av alla naturliga tal. |
| $\mathbb{Z} = \{... - 2, -1, 0, 1, 2, ...\}$ | Mängden av alla heltal |
| $\mathbb{Z}_+ = \mathbb{N}\backslash\{0\}$ | Mängden av alla positiva heltal |
| $\mathbb{Q} = \{\frac{a}{b} : a, b \in \mathbb{Z}, b \neq 0\}$ | Mängden av alla rationella tal. |
| $\mathbb{R}$ | Mängden av alla reella tal. |
| $\mathbb{C} = \{a + bi : a, b \in \mathbb{R}, i^2 = -1\}$ | Mängden av alla komplexa tal. |
| $\mathbb{H} = \left\{ \begin{matrix} s + ai + bj + ck, s, a, b, c \in \mathbb{R}, \\ i^2 = j^2 = k^2 = ijk = -1 \end{matrix} \right\}$ | Mängden av alla kvaternioner. |
| $\mathbb{H}_{Im} = \left\{ \begin{matrix} 0 + ai + bj + ck, a, b, c \in \mathbb{R}, \\ i^2 = j^2 = k^2 = ijk = -1 \end{matrix} \right\}$ | Mängden av alla rent imaginära kvaternioner. |
| $\mathbb{H}_1 = \{q \in \mathbb{H} : \|\|q\|\| = 1\}$ | Mängden av alla enhetskvaternioner. |
| $\mathbb{Z}_n = \{\mathbb{Z}/n, n \in \mathbb{Z}_+\}$ | Mängden av heltal modulo n |
| $q = s + ai + bj + ck$ | $q$ betecknar kvaternionen med skalärdel $s$ och vektordel $(a, b, c)$. |
| $\mathbf{v} = (v_1, ...v_n)$ | $\mathbf{v}$ betecknar en vektor i $\mathbb{R}^n$ som har komponenterna $v_1, v_2, ..., v_n$. |
| $\mathbf{v} \bullet \mathbf{w}$ | $\bullet$ betecknar skalärprodukten (inre produkten) mellan vektorerna $\mathbf{v}$ och $\mathbf{w}$. |
| $\mathbf{v} \times \mathbf{w}$ | $\times$ betecknar kryssprodukten mellan vektorerna $\mathbf{v}$ och $\mathbf{w}$. |
| $R_v(\theta)$ | Rotationsmatris för rotation $\theta$ radianer kring vektorn $\mathbf{v}$ |
| $[s, (a, b, c)]$ | Ett annat skrivsätt för kvaternionen $q = s + ai + bj + ck$. |
| $SO(n)$ | Mängden av alla ortogonala $n \times n$-matriser med determinant 1. |
| $GL_n(\mathbb{K})$ | Mängden av alla inverterbara $n \times n$-matriser med element i $\mathbb{K}$. |
| $A, B, ...$ | Matriser betecknas med versaler |

# Innehåll

# Förord

Utöver individella tidsloggar så har även en gruppdagbok förts där arbetsprocessen dokumenterats. Varje avsnitt av rapporten har huvdsaklingen tre olika arbetsmoment: inläsning, bearbetning av material och dokumentation. Undantaget från detta är numerikdelen, där simuleringar tillkommer.
Arbetsfördelningen över de olika styckena i rapporten har delats upp enligt nedan:

- Förord: Niklas
- Inledning
  - Dokumentation: Niklas, Edvin
  - Huvudansvarig: Edvin
- Kvaternioner
  - Inläsning: Niklas
  - Bearbetning av material: Niklas, Damiano, Edvin
  - Dokumentation: Niklas
  - Huvudansvarig: Niklas
- Rotationer
  - Bearbetning av material: Niklas, Damiano, Edvin
  - Dokumentation: Damiano (3.1, 3.2.3-3.2.6, 3.3), Niklas (3.2.1-3.2.2)
  - Huvudansvarig: Damiano
- Topologi
  - Bearbetning av material: Niklas, Damiano, Edvin
  - Dokumentation: Edvin
  - Huvudansvarig: Edvin
- Gimbal Lock
  - Bearbetning av material: Niklas, Damiano, Edvin
  - Dokumentation: Edvin
  - Huvudansvarig: Edvin
- Numerik
  - Utveckling av MATLAB-program:
    - Simulering 1: Niklas, Damiano, Edvin
    - Simulering 2: Niklas
  - Bearbetning av material: Niklas, Damiano, Edvin
  - Dokumentation: Niklas
  - Huvudansvarig: Niklas
- Resultat
  - Bearbetning: Niklas, Damiano, Edvin
  - Dokumentation: Niklas
  - Huvudansvarig: Niklas
- Diskussion
  - Bearbetning: Niklas, Damiano, Edvin

  - Dokumentation: Niklas, Edvin, Damiano
  - Huvudansvarig: Niklas, Edvin
- Slutsatser
  - Bearbetning: Niklas, Damiano, Edvin
  - Dokumentation:Niklas, Damiano, Edvin
  - Huvudansvarig: Niklas, Damiano, Edvin
- Cliffordalgebror
  - Inläsning: Niklas
  - Bearbetning av material: Niklas
  - Dokumentation: Niklas
  - Huvudansvarig: Niklas

# 1 Inledning

Kvaternioner kan ses som en utvidgning av de komplexa talen. Deras räkneregler och koppling till rotationer upptäcktes av Olinde Rodrigues [3] 1840, men slog inte igenom förän William Rowan Hamilton, oberoende av Rodrigues, upptäckte kvaternionerna 1843. Innan matrisalgebran utvecklades användes kvaternioner för att representera vektorer i rummet[6]. Idag är matriser och vektorer långt vanligare än kvaternioner i tillämpningar, men just för att representera rotationer uppvisar matrisframställningen brister som inte återfinns hos kvaternionerna. Kvaternioner har länge använts inom rymdfarten[8] och förekommer även inom datoranimering och robotik. Gemensamt för dessa områden är att de är beroende av snabba och stabila datoriserade rotationsberäkningar.

## 1.1 Syfte

Syftet med det här projektet är att studera och förstå rotationer i rummet, både ur ett tillämpat och rent matematiskt perspektiv. Att väl kunna modellera och simulera rotationer är viktigt inom många områden. Trots sin underliggande gruppstruktur bjuder rotationerna dock motstånd när man försöker räkna med dem; såväl datorer som människor tenderar föredra att räkna med tal. Vi vill därför ge exempel på och jämföra olika parametriseringar av rotationerna och utröna deras egenheter, även ur ett numeriskt perspektiv. Speciellt vill vi undersöka hur kvaternioner kan användas för att representera rotationer, detta då de ofta beskrivs som mycket lämpliga ur beräkningssynpunkt.

## 1.2 Avgränsningar

Rotationer utgör en grupp och det finns många sätt att använda dess element. Man kan till exempel studera sätt att interpolera en mängd av dem, lösa rotationsekvationer eller sätta dem samman till matriser. Vi har dock, utöver att beskriva mängderna rotationer och kvaternioner, huvudsakligen begränsat oss till att studera sammansättning, operationen under vilken rotationerna är en grupp. En annan, lätt förbisedd, avgränsning är att vi bara jobbar med rotationer i tre dimensioner, detta då det inte i alla högre dimensioner går att räkna med kvaternioner. Slutligen har vi inte studerat de generaliseringar och vidareutvecklingar av kvaternioner som kan göras då detta skulle vara att frångå vårt huvudspår rotationerna. Ett undantag är Cliffordalgebrorna, en större klass algebror av vilka kvaternionerna utgör ett specialfall, vilka behandlas i ett appendix.

Endast två simuleringar har genomförts, och dessa involverar endast vridningar på formen axel-vinkel.

## 1.3 Metod

Arbetet är främst en literaturstudie men inbegriper även enstaka egna resultat.Huvuddelen av arbetstiden har ägnats åt det rent matematiska, men även simuleringar har ingått som viktig del. Simuleringarna har syftat till att jämföra kvaternion- och matrisrepresentationer av rotationer. Vi har jämfört tidsåtgång och stabilitet; tidsåtgången är viktig när man till exempel vill ha snabba datoranimationer och stabiliteten då man arbetar med väldigt långa serier rotationer, exempelvis innom robotik.

## 1.4 Rapportens upplägg

Rapporten inleds med definitioner för kvaternioner och rotationer samt egenskaper hos dessa. Därefter presenteras de två studerade parametriserningarna av rotationerna och hur de kan skrivas med kvaternioner och matriser. De olika parametriseringarnas topologi diskuteras i två avsnitt innan simuleringsavsnittet, i vilket kvaternioner och matriser jämförs numeriskt. Till sist knyts de olika delarna ihop med en diskussionsdel.

# 2 Kvaternioner

Det första avsnittet ägnas åt att beskriva vad kvaternioner är och de mest grundläggande begreppen i teorin kring dessa. Begrepp som konjugat, norm, m.m. definieras för att kunna användas i senare kapitel. Dessutom presenteras olika framställningar av kvaternioner. Vi börjar med att precisera vad vi menar med en kvaternion:

**Definition 2.1.** *(Hamiltons definition av kvaternioner) Med en kvaternion $q$ avses ett tal på formen*

$$q = s + ai + bj + ck \tag{1}$$

*där $s, a, b, c \in \mathbb{R}$. $s$ kallas realdelen av $q$ och $a, b, c$ är imaginärdelarna av $q$. i,j,k kallas de imaginära enheterna, och uppfyller $i^2 = j^2 = k^2 = ijk = -1$. Mängden av alla kvaternioner betecknas $\mathbb{H}$*

Ibland används termerna *skalärdel* och *vektordel*, och då skrivs $q = (s, \mathbf{v})$, där $s$ är skalärdelen för $q$ och $\mathbf{v}$ är vektordelen, $\mathbf{v} = (a, b, c)$.

Först behandlas den kanske mest naturliga härledningen, nämligen att skapa kvaternionerna genom en utvidgning av de komplexa talen.

## 2.1 Utvidgning av de komplexa talen

De komplexa talen fås som bekant från de reella genom att man inför talet $i$ som uppfyller $i^2 = -1$ och betraktar alla linjärkombinationer $x + yi$, där $x, y \in \mathbb{R}$.

När man inför kvaternionerna kan det göras på liknande sätt. Låt $j$ beteckna den imaginära enheten, $j^2 = -1$, och betrakta alla tal på formen $\alpha + \beta j$, där $\alpha, \beta \in \mathbb{C}$.

Om $s + ai$ och $b + ci$ är två komplexa tal är den kvaternion $q$ som fås med ovanstående definition $q = (s + ai) + (b + ci)j = s + ai + bj + cij$, vilket följer av räkneregler för komplexa tal. Inför nu beteckningen $ij = k$ och definiera $ij = -ji$ så kan $q$ skrivas som

$$q = (s + ai) + (b + ci)j = s + ai + bj + ck$$

vilken är den form som Hamilton använde när han först definierade kvaternioner.

Det följer från definitionen att $i^2 = j^2 = -1$, och det konstateras enkelt att $k^2 = (ij)^2 = ijij = i(-ij)j = -i^2j^2 = -(-1) \cdot (-1) = -1$. Vidare är $(ij)k = kk = -1$, vilket stämmer överens med definitionen.

Nästkommande avsnitt definierar viktiga begrepp som används när man arbetar med kvaternioner.

## 2.2 Grundläggande egenskaper

För addition av två kvaternioner $q_1 = (s_1, \mathbf{v}_1)$ och $q_2 = (s_2\mathbf{v_2})$ gäller, analogt med komplexa tal:

$$q_1 + q_2 = (s_1 + s_2, \mathbf{v}_1 + \mathbf{v}_2) = (s_1 + s_2) + (a_1 + a_2)i + (b_1 + b_2)j + (c_1 + c_2)k$$

Produkter av de imaginära enheterna får man genom omskrivning av de formler som ingår i definitionen av en kvaternion. Ett exempel är som följer: antag att produkten $jk$ ska beräknas. Enligt definitionen av en kvaternion gäller att $ijk = -1$.Vi kan då multiplicera båda leden med $-i$ och få $(-i)(ijk) = -(-i) \Leftrightarrow jk = i$.

På samma sätt härleds följande tabell över produkter av de imaginära enheterna $i, j, k$.

| · | **i** | **j** | **k** |
|---|---|---|---|
| **i** | -1 | k | -j |
| **j** | -k | -1 | i |
| **k** | j | - i | -1 |

Tabell 1: Cayley-tabell över $i, j, k$ med operationen multiplikation

Ur tabellen fås t.ex. att $ij = k$, men $ji = -k$, vilket visar att kvaterioner inte kommuterar.

Eftersom kvaternioner är nära besläktade med de komplexa talen så är det rimligt att anta att man kan multiplicera två kvaternioner på samma sätt som komplexa tal. Detta fungerar utmärkt att göra – det enda man behöver göra är att tillämpa den distributiva lagen och komma ihåg att kvaternioner inte är kommutativa. Vi visar med ett numeriskt exempel:

$$(1+2j)\cdot(2i+3k) = 1\cdot 2i + 1\cdot 3k + 2j\cdot 2i + 2j\cdot 3k$$

$$= 2i + 3k + 4ji + 6jk = 2i + 3k + 4(-k) + 6\cdot i = 2i + 3k - 4k + 6i = -4i - k.$$

I ovanstående beräkning har tabell 1 använts för beräkning av produkter mellan $i, j$ och $k$.

Det finns en formel för produkten av två kvaternioner, vilken visas i följande lemma:

**Lemma 2.2.** *För produkten av två kvaternioner* $q_1 = s_1 + a_1 i + b_1 j + c_1 k = (s_1, \mathbf{v}_1), q_2 = s_2 + a_2 i + b_2 j + c_2 k = (s_2, \mathbf{v}_2)$ *gäller att*

$$\begin{aligned} q_1 q_2 = (s_1, \mathbf{v}_1)\cdot(s_2, \mathbf{v}_2) = \\ (s_1 s_2 - a_1 a_2 - b_1 b_2 - c_1 c_2) + (s_1 a_2 + a_1 s_2 + b_1 c_2 - c_1 b_2) i + \\ + (s_1 b_2 + b_1 s_2 + c_1 a_2 - a_1 c_2) j + (s_1 c_2 + a_1 b_2 - b_1 a_2 + c_1 s_2) k = \\ = (s_1 s_2 - (\mathbf{v}_1 \bullet \mathbf{v}_2), s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \end{aligned}$$

Beviset är enbart en beräkning. Andra likheten följer av att man beräknar produkten för två kvaternioner m.h.a. distributiva lagen och kommer ihåg att behålla rätt ordning för de imaginära enheterna. Sista likheten följer av definitionen för kryss- och skalärprodukt.

Vi noterar att kvaternioner kan ses som vektorer i $\mathbb{R}^4$ genom att låta kvaternionen $q = s + ai + bj + ck \in \mathbb{H}$ representeras av vektorn $(s, a, b, c) \in \mathbb{R}^4$. Detta innebär att mängden av alla kvaternioner tillsammans med operationen addition, dvs. $(\mathbb{H}, +)$, är ett vektorrum, och samma norm som i $\mathbb{R}^4$ kan användas i $\mathbb{H}$.

**Definition 2.3. Konjugatet** *till kvaternionen* $q = s + ai + bj + ck$ *definieras som kvaternionen* $s - ai - bj - ck$ *och betecknas* $\bar{q}$.

**Definition 2.4. Normen** *av kvaternionen* $q$ *definieras som det reella talet* $||q|| = \sqrt{q\bar{q}} = \sqrt{s^2 + a^2 + b^2 + c^2}$. *Man brukar vissa fall använda beteckningen* $N(q) = q\bar{q} = \bar{q}q = ||q||^2$.

**Definition 2.5.** *Låt* $q$ *vara en kvaternion. Om* $||q|| = 1$ *kallas* $q$ *för en* **enhetskvaternion**. *Mängden av alla enhetskvaternioner betecknas* $\mathbb{H}_1$.

**Lemma 2.6.** *Om* $q_1 = s_1 + a_1 i + b_1 j + c_1 k$ *och* $q_2 = s_2 + a_2 i + b_2 j + c_2 k$ *är två kvaternioner så gäller följande:*

1. $\overline{q_1 + q_2} = \bar{q_1} + \bar{q_2}$
2. $\overline{q_1 q_2} = \overline{q_2 q_1}$
3. $N(q_1 q_2) = N(q_1) N(q_2)$

*Bevis.* Beviset är enbart en rutinberäkning och lämnas som övning åt läsaren □

**Definition 2.7.** *Låt* $q \neq 0$ *vara en kvaternion.* **Inversen** *till* $q$ *betecknas* $q^{-1}$ *och är den entydigt bestämda kvaternion som uppfyller* $qq^{-1} = q^{-1}q = 1$

**Sats 2.8.** *För en kvaternion* $q \neq 0$ *gäller att* $q^{-1} = \frac{\bar{q}}{N(q)}$

*Bevis.* Vi har

$$qq^{-1} = q \cdot \frac{\bar{q}}{N(q)} = \frac{q\bar{q}}{N(q)} = \frac{N(q)}{N(q)} = 1$$

och

$$q^{-1}q = \frac{\bar{q}}{N(q)} \cdot q = \frac{\bar{q}q}{N(q)} = \frac{N(q)}{N(q)} = 1$$

Således gäller att $q^{-1} = \frac{\bar{q}}{N(q)}$ är såväl höger- som vänsterinvers till $q$. □

**Korollarium 2.9.** *Om $q$ är en enhetskvaternion så är $q^{-1} = \bar{q}$*

*Bevis.* Detta följer direkt av sats 2.8. För en enhetskvaternion gäller att $N(q) = 1$, så $q^{-1} = \frac{\bar{q}}{N(q)} = \frac{\bar{q}}{1} = \bar{q}$ □

**Definition 2.10.** *En* **kropp** *är en mängd $\mathbb{K}$ med två operationer, addition och multiplikation, som uppfyller följande villkor*

*1)* $x + y = y + x$, $\forall x, y \in \mathbb{K}$ *(kommutativa lagen för addition)*

*2)* $x + (y + z) = (x + y) + z$, $\forall x, y, z \in \mathbb{K}$ *(associativa lagen för addition)*

*3)* $x(yz) = (xy)z$, $\forall x, y, z \in \mathbb{K}$ *(associativa lagen för multiplikation)*

*4)* $x(y + z) = xy + xz \ \wedge \ (x + y)z = xz + yz$ $\forall x, y, z \in \mathbb{K}$ *(distributiva lagarna)*

*5)* $\exists 0 : x+0 = 0+x = x$, $\exists 1 : 1 \cdot x = x \cdot 1 = x$, $\forall x \in \mathbb{K}$ *(existens av nollelement och identitet)*

*6)* $\forall x \in \mathbb{K} \ \exists(-x) \in \mathbb{K} : x + (-x) = (-x) + x = 0$ *(invers vid addition).*

*7)* $\forall x \in \mathbb{K} \setminus \{0\} \ \exists x^{-1} \in \mathbb{K} : x \cdot x^{-1} = x^{-1} \cdot x = 1$ *(invers vid multiplikation).*

**Definition 2.11.** *Om $\mathbb{K}$ är en kropp där $ab = ba$ för alla $a, b \in \mathbb{K}$ kallas $\mathbb{K}$ för en* **kommutativ kropp**. *I annat fall kallas $\mathbb{K}$ för en* **skevkropp** *eller* **divisionsring**.

**Sats 2.12.** *$\mathbb{H}$ är en divisionsring.*

*Bevis.* Låt $q_1 = s_1 + a_1 i + b_1 j + c_1 k, q_2 = s_2 + a_2 i + b_2 j + c_2 k$ och $q_3 = s_3 + a_3 i + b_3 j + c_3 k$ vara kvaternioner. För att visa att $\mathbb{H}$ är en divisionsring måste vi visa att $\mathbb{H}$ är en kropp och att $q_1 q_2 \neq q_2 q_1$ för några $q_1 q_2 \in \mathbb{H}$. Först visas att $\mathbb{H}$ är en kropp, vilket görs genom att kontrollera att punkterna 1-7 i definition 2.10 är uppfyllda.

**Kommutativa lagen för addition** och **associativa lagen för addition** följer direkt av det faktum att $(\mathbb{H}, +, \cdot)$ kan ses som ett vektorrum isomorft med $\mathbb{R}^4$.

**Associativa lagen för multiplikation:** Enligt definitionen av kvaternioner gäller

$$i(jk) = i \cdot i = -1 = kk = (ij)k$$

så $i(jk) = (ij)k$.

Analogt visas att alla kombinationer av de imaginära enheterna är associativa. Men då är även alla linjärkombinationer av baselementen associativa, varför alla kvaternioner är det.

**Distributiva lagarna** ärvs från $\mathbb{C}$ .

**Existens av nollelement** är nollkvaternionen, vilket följer av att man kan se $(\mathbb{H}, +, \cdot)$ som ett vektorrum isomorft med $\mathbb{R}^4$.

**Neutralt elementet m.a.p. multiplikation** är kvaternionen 1, ty $1 \cdot q = q \cdot 1 = q$. Detta visas enkelt med hjälp av definitionen för multiplikation av kvaternioner.

**Invers vid addition** existerar, eftersom $(\mathbb{H}, +) \cong \mathbb{R}^4$. **Invers vid multiplikation** för alla nollskilda kvaternioner existerar enligt sats 2.8

Alltså är $(\mathbb{H}, +, \cdot)$ en kropp. Vidare är exempelvis $ij = k$ men $ji = -k$, dvs. vi har inte att $ab = ba$ för alla kvaternioner. Detta visar att $\mathbb{H}$ en divisionsring. □

## 2.3 Kvaternioner som komplexa 2x2-matriser

Betrakta avbildningen $\mathbb{H} \to GL_2(\mathbb{C}) : (s + ai + cj + dk) \mapsto \begin{pmatrix} s + ai & b + ci \\ -b + ci & s - ai \end{pmatrix}$

Avbildningen skapar en relation mellan komplexa matriser på formen ovan och kvaternionerna och är en isomorfi. Elementen $1, i, j, k \in \mathbb{H}$ identifieras med matriserna $E, I, J, K \in GL_2\mathbb{C}$ definierade enligt:

$$E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, I = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}, J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, K = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \tag{2}$$

Beräknas nu produkten $I \cdot J$ med matrismultiplikation fås

$$I \cdot J = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} = K$$

Det är känt att produkten av kvaternionerna $i$ och $j$ ska bli $k$, och för att avbildningen ovan ska vara meningsfull behöver någon operation motsvarande multiplikation av kvaternioner införas. Detta visar att för $i$ och $j$, samt dess representationer i $GL_2(\mathbb{C})$ fungerar det med vanlig matrismultiplikation. På samma sätt verifierar man att alla kombinationer av produkter mellan bilderna $E, I, J, K \in GL_2(\mathbb{C})$ av $1, i, j, k \in \mathbb{H}$ blir motsvarande resultat som produkter mellan $1, i, j, k$. Detta innebär att multiplikation av kvaternioner i $\mathbb{H}$ motsvaras av matrismultiplikation i $GL_2(\mathbb{C})$.

**Sats 2.13.** *Om*

$$Q = \begin{pmatrix} s + ai & b + ci \\ -b + ci & s - ai \end{pmatrix}$$

*är matrisframställningen av kvaternionen $q = s + ai + bj + ck$ gäller att $det(Q) = ||q||^2$.*

*Bevis.* Beräkning av determinanten ger direkt det önskade resultatet:

$$det \begin{pmatrix} s + ai & b + ci \\ -b + ci & s - ai \end{pmatrix} = (s + ai)(s - ai) - (b + ci)(-b + ci) = s^2 + a^2 + b^2 + c^2 = ||q||^2$$

□

**Sats 2.14.** *Låt $Q$ och $q$ vara som i sats 2.13 ovan. Då gäller att $q^{-1} \mapsto Q^{-1}$, dvs matrisrepresentationen av $q^{-1}$ är samma som matrisinversen till matrisrepresentationen av $q$.*

*Bevis.* Även detta är enbart en beräkning. En $q$ har inversen $q^{-1} = \frac{\bar{q}}{q\bar{q}} = \frac{1}{q\bar{q}}(s - ai - bj - ck)$. Denna kvaternion kan skrivas på matrisform som

$$\frac{1}{s^2 + a^2 + b^2 + c^2} \begin{pmatrix} s - ai & -b - ci \\ b - ci & s + ai \end{pmatrix}$$

vilket känns igen som matrisinversen till

$$\begin{pmatrix} s + ai & b + ci \\ -b + ci & s - ai \end{pmatrix}$$

Att bestämma inversen till en kvaternion om man har den på matrisform är alltså samma sak som att bestämma inversen till matrisen. Inversen till kvaternionen existerar precis då dess norm är nollskild, vilket är ekvivalent med att determinanten för dess matrisframställning är nollskild. □

# 3 Representationer av rotationer

Detta kapitel behandlar rotationers representationer och räkneregler. Först måste självklart ordet rotation definieras, sedan representationssätten beskrivas och slutligen räknereglerna härledas. För att definiera rotation måste vi dock först veta vad en kongruensavbildning är.

**Definition 1.** *Med en* **kongurensavbildning** *menas en funktion av planet eller rummet på sig självt, som bevarar alla avstånd och vinklar (ej orienterade vinklar). Egenskaper som är oförändrade i kongruensavbildningar är rätlinjighet, parallellitet och delningsförhållandet. Om orienteringen bevaras kallas avbildningen direkt, i motsatt fall indirekt. [10]*

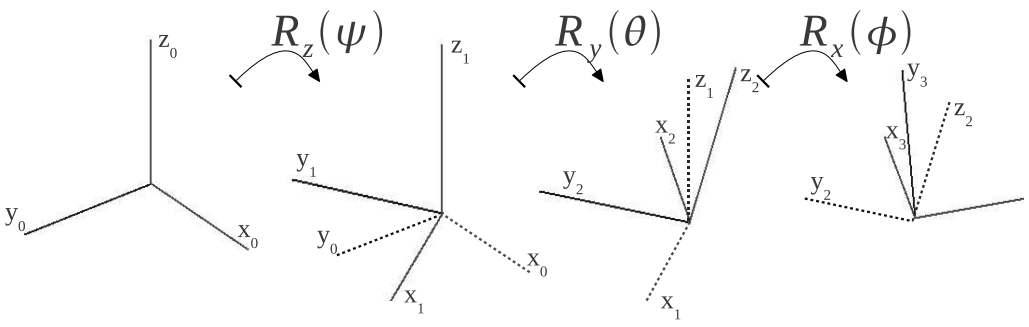
**Definition 2.** *En rotation är en direkt kongruensavbildning kring en mängd av fixpunkter. [10]*

Ett alternativt sätt att säga detta är att en rotation är en avbildning kring en axel genom origo av ett par vektorer sådan att dessa behåller skalärprodukt och kryssprodukt.

## 3.1 Eulervinklar och axel-vinkel - två parametriseringar av rotationer

Eulervinklar är ett sätt att parametrisera rotationer, infört av Leonhard Euler. En rotation beskrivs som en hopsättning av tre oberoende vinklar kring varsin fixerad axel. De tre rotationerna man utför brukar namnges till gir, stigning och roll eftersom man kan föreställa sig ett flygplan som lyfter: Först girar flygplanet på marknivå, då vrids z-axeln, sedan stiger flygplanet mot himlen vilket innebär att man vrider den nya y-axeln. Slutligen kan flygplanet utföra en roll och då vrids x-axeln den nu har. Eftersom rotationer ej är kommutativa krävs det att rotationerna utförs i den exakta ordningen.

Det finns alternativa sätt att representera rotationerna på. Man kan exempelvis välja att utföra en $z$-$x$-$z$ vridning (vanligt bland fysiker), men i detta avsnitt används $z$-$y$-$x$. Notera att vi använder beteckningen $\langle \phi, \theta, \psi \rangle$ där vridningarna är kring $x$-$y$-$z$ axlarna. Vridningarna läses alltså från höger till vänster. Så t.ex. $\langle 0, 90°, 0 \rangle [1, 0, 0] = [0, 0, -1]$ [13]




Figur 1: Rotation med Eulervinklar; heldragna linjer är de nya koordinataxlarna och streckade de gamla

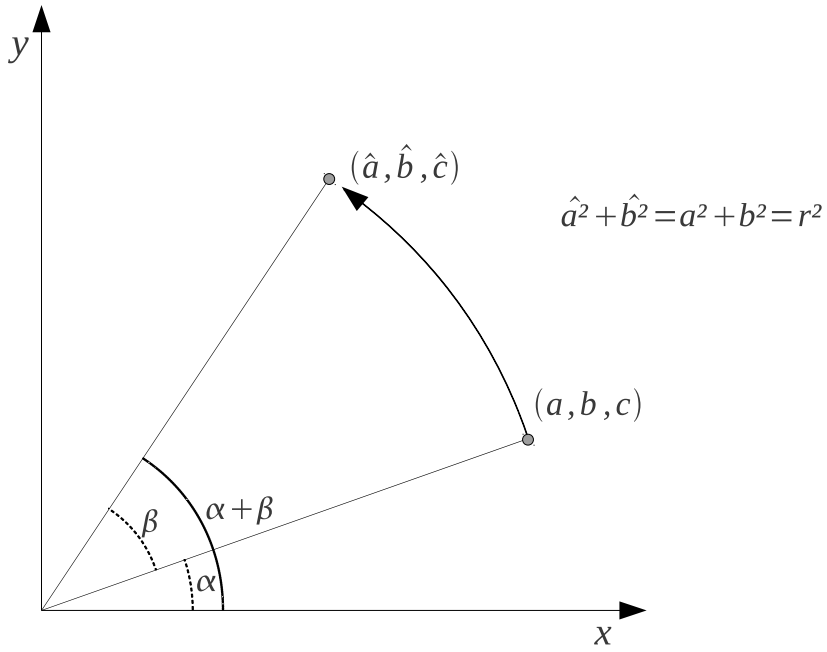
Ett annat sätt att se på en rotation i tre dimensioner är den så kallade axel-vinkel-parametriseringen. Metoden innebär att rotationen ses som en vridning kring en enda axel med en vinkel. Att axel-vinkelparametriseringen faktiskt kan ge alla vridningar i rummet bevisades av Leonhard Euler år 1775 [7]. Denna parametrisering har många fördelar, vilka kommer att lyftas fram i senare kapitel.

## 3.2 Sätt att representera rotationer

Nedan följer fyra sätt att skriva rotationer på, deras räkneregler ges och deras egenskaper tas upp. Först beskrivs Eulervinklar och axel-vinkel på matrisform och sedan på kvaternionform.

### 3.2.1 Representation av Eulervinklar på matrisform

Betrakta följande problem: en given punkt $p = (a, b, c) \in \mathbb{R}^3$ ska roteras $\beta$ radianer kring z-axeln till den nya punkten $\hat{p} = (\hat{a}, \hat{b}, \hat{c})$. Det är uppenbart att z-koordinaten inte förändras eftersom rotationen sker kring z-axeln, varför $\hat{c} = c$, och det räcker således att undersöka vad som händer med koordinaterna $a$ och $b$ i det plan parallellt med xy-planet som vi roterar i.




Figur 2: Punkten $p = (a, b, c)$ ska vridas $\beta$ radianer till punkten $\hat{p} = (\hat{a}, \hat{b}, \hat{c})$

Med hjälp av grundläggande trigonometri kan vi sätta upp följande uttryck för $p$ och $\hat{p}$.

$$\begin{aligned} a &= r \cdot cos(\alpha) \\ b &= r \cdot sin(\alpha) \\ \hat{a} &= r \cdot cos(\alpha + \beta) \\ \hat{b} &= r \cdot sin(\alpha + \beta) \end{aligned}$$

Nu används additionsformlerna för sinus och cosinus varpå uttrycken för $a$ respektive $b$ sätts in:

$$\begin{aligned} \hat{a} = r \cdot cos(\alpha + \beta) = r(cos(\alpha)cos(\beta) - sin(\alpha)sin(\beta)) = \\ r \cdot cos(\alpha)cos(\beta) - r \cdot sin(\alpha)sin(\beta) = a \cdot cos(\beta) - b \cdot sin(\beta) \end{aligned}$$

$$\begin{aligned} \hat{b} = r \cdot sin(\alpha + \beta) = r(sin(\alpha)cos(\beta) + sin(\beta)cos(\alpha)) = \\ r \cdot sin(\alpha)cos(\beta) + r \cdot sin(\beta)cos(\alpha) = b \cdot cos(\beta) + a \cdot sin(\beta) \end{aligned}$$

Koordinaterna för $(\hat{a}, \hat{b}, \hat{c})$ är nu en funktion av $(a, b, c)$ samt vinkeln de vrids. Skrivs detta på matrisform fås:

$$\hat{p} = \begin{pmatrix} \hat{a} \\ \hat{b} \\ \hat{c} \end{pmatrix} = \begin{pmatrix} a \cdot cos(\beta) - b \cdot sin(\beta) \\ b \cdot cos(\beta) + a \cdot sin(\beta) \\ c \end{pmatrix} = \begin{pmatrix} cos(\beta) & -sin(\beta) & 0 \\ sin(\beta) & cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} := R_z(\beta)p$$

Alltså gäller att om vektorn $\mathbf{x} = (a, b, c)$ ska vridas $\beta$ radianer kring z-axeln utförs det genom multiplikation med matrisen enligt ovan. Rotationsmatrisen som roterar $\beta$ radianer kring z-axeln betecknas med $R_z(\beta)$.

Helt analogt med ovanstående härleds rotationsmatrisen för rotation kring x- respektive y-axeln. Detta ger:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Mängden av rotationsmatriser i 3 dimensioner är isomorf med en grupp som betecknas $SO(3)$, specialortogonala $3 \times 3$ matriser. Dessa är ortogonala matriser med determinant 1.

Antag att vi vill rotera en punkt $\mathbf{p} = (a, b, c)$ först $\alpha$ radianer kring x-axeln, sedan $\beta$ radianer kring y-axeln och till sist $\gamma$ radianer kring z-axeln. Den sammansatta rotationsmatrisen ges då av matrismultiplikationen

$$R_z(\gamma)R_y(\beta)R_x(\alpha) =$$

$$\begin{pmatrix} cos(\beta)cos(\gamma) & cos(\gamma)sin(\alpha)sin(\beta) - cos(\alpha)sin(\gamma) & sin(\alpha)sin(\gamma) + cos(\alpha)cos(\gamma)sin(\beta) \\ cos(\beta)sin(\gamma) & cos(\alpha)cos(\gamma) + sin(\alpha)sin(\beta)sin(\gamma) & cos(\alpha)sin(\beta)sin(\gamma) - cos(\gamma)sin(\alpha) \\ -sin(\beta) & cos(\beta)sin(\alpha) & cos(\alpha)cos(\beta) \end{pmatrix}$$

### 3.2.2 Representation av axel-vinkelrotation på matrisform

**Sats 3.1. (Axel-vinkel på matrisform)** *Låt $\mathbf{p} = (x, y, z)$ vara en punkt som ska roteras $\theta$ radianer kring en godtycklig normalvektor $\mathbf{n} = (n_1, n_2, n_3)$ till en ny punkt $\hat{\mathbf{p}} = (\hat{x}, \hat{y}, \hat{z})$. Den rotationsmatris $R_{\mathbf{n}}(\theta)$ som uppfyller $\hat{\mathbf{p}} = R_{\mathbf{n}}(\theta)\mathbf{p}$ ges då av*

$$R_{\mathbf{n}}(\theta) =$$

$$= \begin{pmatrix} cos(\theta) + n_1^2(1 - cos(\theta)) & n_1n_2(1 - cos(\theta)) - n_3sin(\theta) & n_1n_3(1 - cos(\theta)) + n_2sin(\theta) \\ n_2n_3(1 - cos(\theta)) + n_3sin(\theta) & cos(\theta) + n_2^2(1 - cos(\theta)) & n_2n_3(1 - cos(\theta)) - n_2sin(\theta) \\ n_3n_1(1 - cos(\theta)) - n_2sin(\theta) & n_3n_2(1 - cos(\theta)) + n_1sin(\theta) & cos(\theta) + n_3^2(1 - cos(\theta)) \end{pmatrix}$$

*Bevis.* Dela upp $\mathbf{p}$ i två komponenter: en vinkelrät mot $\mathbf{n}$ och en parallell med $\mathbf{n}$. Låt $\mathbf{p}_\parallel$ vara parallell med $\mathbf{n}$ och $\mathbf{p}_\perp$ vara vinkelrät med $\mathbf{n}$ så att $\mathbf{p} = \mathbf{p}_\parallel + \mathbf{p}_\perp$

Grundläggande linjär algebra (projektionsformeln) ger att

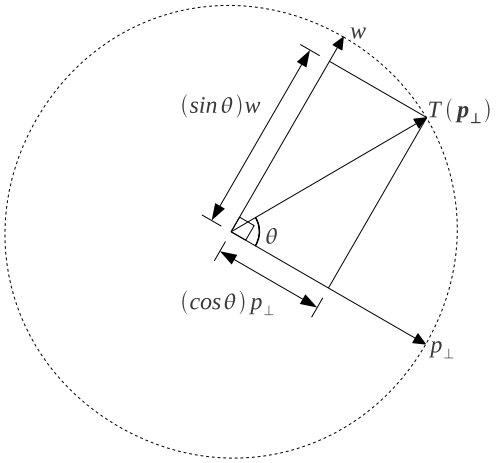
$$\mathbf{p}_\parallel = \frac{\mathbf{p} \cdot \mathbf{n}}{||\mathbf{n}||^2}\mathbf{n} = (\mathbf{p} \cdot \mathbf{n})\mathbf{n}$$

och

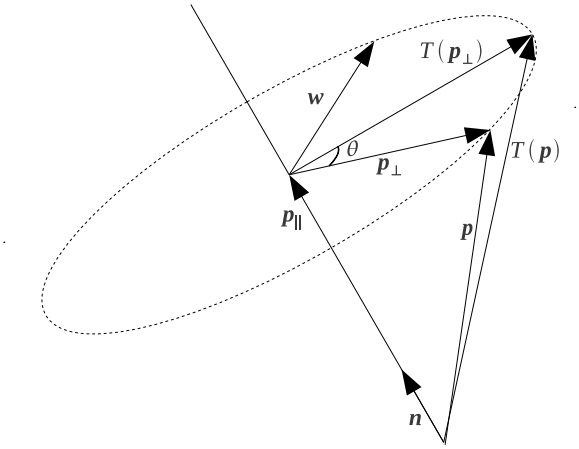
$$\mathbf{p}_\perp = \mathbf{p} - (\mathbf{p} \cdot \mathbf{n})\mathbf{n}$$

Notera att detta ger speciellt att $\mathbf{p} = \mathbf{p}_\parallel + \mathbf{p}_\perp$.

Låt vidare $T$ beteckna operatorn för den sökta rotationen. Vi söker $T(\mathbf{p}) = T(\mathbf{p}_\parallel + \mathbf{p}_\perp) = T(\mathbf{p}_\parallel) + T(\mathbf{p}_\perp)$. Självklart är $T(\mathbf{p}_\parallel) = \mathbf{p}_\parallel$, varför det räcker att hitta $T(\mathbf{p}_\perp)$ För att göra detta betraktas det plan i vilken rotationen sker. Först behövs en bas för det planet. Som första basvektor kan $\mathbf{p}_\perp$ användas, och som andra basvektor väljs $\mathbf{w} = \mathbf{n} \times \mathbf{p}_\perp = \mathbf{n} \times \mathbf{p}$.




Figur 3: Rotation kring en godtycklig axel genom origo sett från sidan av planet.




Figur 4: Rotation kring en godtycklig axel genom origo sett ovanifrån planet.

Genom att betrakta figur 3 och 4 inses att $T(\mathbf{p}_\perp) = cos(\theta)\mathbf{p}_\perp + sin(\theta)\mathbf{w} = cos(\theta)\mathbf{p}_\perp + sin(\theta)(\mathbf{n} \times \mathbf{p})$. Detta ger att

$$\begin{aligned} T(\mathbf{p}) = \mathbf{p}_\parallel + T(\mathbf{p}_\perp) = (\mathbf{p}\cdot\mathbf{n})\mathbf{n} + cos(\theta)\mathbf{p}_\perp + sin\theta(\mathbf{n}\times\mathbf{p}) = \\ = (\mathbf{p}\cdot\mathbf{n})\mathbf{n} + cos(\theta)(\mathbf{p} - (\mathbf{p}\cdot\mathbf{n})\mathbf{n}) + sin\theta(\mathbf{n}\times\mathbf{p}) = \\ (\mathbf{p}\cdot\mathbf{n})\mathbf{n} + cos(\theta)\mathbf{p} - cos(\theta)(\mathbf{p}\cdot\mathbf{nn}) + sin\theta(\mathbf{n}\times\mathbf{p}) = \\ (1 - cos\theta)(\mathbf{p}\cdot\mathbf{n})\mathbf{n} + cos(\theta)\mathbf{p} + sin\theta(\mathbf{n}\times\mathbf{p}) \end{aligned}$$

Följande likheter används för att skriva om ovanstående uttryck:

$$\mathbf{u}\times\mathbf{v} = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix}\mathbf{v}$$

$$(\mathbf{v}\bullet\mathbf{u})\mathbf{u} = \begin{pmatrix} u_1^2 & u_1u_2 & u_1u_3 \\ u_1u_2 & u_2^2 & u_2u_3 \\ u_1u_3 & u_2u_3 & u_3^2 \end{pmatrix}\mathbf{v}$$

Dessa likheter bevisas enkelt enbart genom att använda definitionen av kryssprodukt respektive skalärprodukt. Uttrycket för $T(\mathbf{p})$ kan då skrivas

$$T(\mathbf{p}) = (1 - cos\theta)(\mathbf{p} \bullet \mathbf{n})\mathbf{n} + cos\theta\mathbf{p} + sin\theta(\mathbf{n} \times \mathbf{p}) =$$

$$(1-cos\theta)\begin{pmatrix} n_1^2 & n_1n_2 & n_1n_3 \\ n_1n_2 & n_2^2 & n_2n_3 \\ n_1n_3 & n_2n_3 & n_3^2 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} cos\theta\mathbf{p} + sin\theta \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix} \mathbf{p} = .... =$$

$$= \begin{pmatrix} (1-cos\theta)n_1^2 + cos\theta & (1-cos\theta)n_1n_2 - sin\theta n_3 & (1-cos\theta)n_1n_3 + sin\theta n_2 \\ (1-cos\theta)n_1n_2 + sin\theta n_3 & (1-cos\theta)n_2^2 + cos(\theta) & (1-cos\theta)n_2n_3 - sin\theta n_1 \\ (1-cos\theta)n_1n_3 - sin\theta n_2 & (1-cos\theta)n_2n_3 + sin\theta n_1 & (1-cos\theta)n_3^2 + cos\theta \end{pmatrix} \mathbf{p} =$$

$$= R_{\mathbf{n}}(\theta)\mathbf{p}$$

[11] □

### 3.2.3 Kvaternioner kan ses som rotationer och rotationer som kvaternioner

Detta avsnitt behandlar relationen mellan kvaternioner och rotationer. Först bevisas att kvaternioner faktiskt kan ses som rotationer, på det sätt att det till varje rotation existerar minst en motsvarande kvaternion. Senare tas olika representationer av rotationer upp, och hur dessa kan uttryckas som kvaternioner.

**Sats 3.2.** *För varje $\hat{q} \in \mathbb{H}$ och $p \in Im\mathbb{H}$ är $\hat{q}p\hat{q}^{-1}$ en rotation av punkten $p$.*

*Bevis.* Beviset delas upp i fem delar för att underlätta läsning:

3.2.0) Låt $q = \frac{\hat{q}}{N(\hat{q})} \in \mathbb{H}_1$, då $\hat{q}p\hat{q}^{-1} = qp\bar{q} := \theta_q p$.
3.2.1) $\mathbb{R}^3 \cong Im\mathbb{H}$
3.2.2) $\theta_q p \in Im\mathbb{H}$
3.2.3) $e^p = q = cos(\phi) + \mathbf{n}sin(\phi)$ där $n$ är en enhetsvektor.
3.2.4) $\theta_q p$ är en direkt kongruensavbildning kring fixpunkter (med andra ord bevarar $\theta_q p$ skalärprodukt och kryssprodukt).

3.2.0) $\hat{q} = qN(\hat{q})$ och $\hat{q}^{-1} = \frac{\bar{\hat{q}}}{N(\hat{q})}$ ger att $\hat{q}p\hat{q}^{-1} = qp\bar{q}\frac{N(\hat{q})}{N(\hat{q})} = qp\bar{q} := \theta_q p$.

3.2.1) Eftersom $\mathbb{H} \cong \mathbb{R}^4$ följer att ett rent imaginärt element i $\mathbb{H}$ kan representeras som en vektor i $\mathbb{R}^3$.

3.2.2) Låt $q = s + ai + bj + ck$ och $p = 0 + di + ej + fk$.
Kvaternionmultiplikation ger att:

$$Re(\theta_q p) = -s(ad + be + cf) + a(bf - ce + ds) + b(cd - af + es) + c(ae - bd + fs) = 0$$

så $qp\bar{q} \in H_{Im}$

3.2.3) Om $p = ai+bj+ck \in Im\mathbb{H}$ representeras som en 2×2 matris fås $p = \begin{pmatrix} ai & b+ci \\ -b+ci & -ai \end{pmatrix}$
Diagonalisera $p$ så att $p = UDU^{-1}$.
$|p - E\lambda| = \begin{vmatrix} ai-\lambda & b+ci \\ -b+ci & -ai-\lambda \end{vmatrix} = -(-a^2-\lambda^2) - (-c^2-b^2) = \lambda^2 + a^2 + b^2 + c^2 = 0$
Låt $\phi^2 = a^2 + b^2 + c^2$, så $\lambda = \pm\phi i$. Detta ger egenvektorerna: $v_+ = \begin{pmatrix} \frac{bi-c}{a-\phi} \\ 1 \end{pmatrix}$ och
$v_- = \begin{pmatrix} \frac{bi-c}{a+\phi} \\ 1 \end{pmatrix}$.
Således är $U = \begin{pmatrix} \frac{bi-c}{a-\phi} & \frac{bi-c}{a+\phi} \\ 1 & 1 \end{pmatrix}$ och $U^{-1} = \begin{pmatrix} \frac{(c+bi)}{2\phi} & \frac{\phi-a}{2\phi} \\ \frac{-(c+bi)}{2\phi} & \frac{\phi+a}{2\phi} \end{pmatrix}$.

Alltså är: $e^p = \begin{pmatrix} \frac{bi-c}{a-\phi} & \frac{bi-c}{a+\phi} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} e^{\phi i} & 0 \\ 0 & e^{-\phi i} \end{pmatrix} \begin{pmatrix} \frac{(c+bi)}{2\phi} & \frac{\phi-a}{2\phi} \\ \frac{-(c+bi)}{2\phi} & \frac{\phi+a}{2\phi} \end{pmatrix} =$

$$\begin{pmatrix} cos(\phi) + \frac{a}{\phi} isin(\phi) & \frac{c+bi}{\phi} sin\phi \\ \frac{-c+bi}{\phi} sin\phi & cos(\phi) - \frac{a}{\phi} isin(\phi) \end{pmatrix}$$

Denna matris motsvarar kvaternionen

$$q = cos(\phi) + \frac{p}{\phi} sin(\phi) \in \mathbb{H}_1$$

Om vi skriver $\mathbf{n} = \frac{p}{\phi} = \frac{(a,b,c)}{\phi}$ får vi $e^p = q = cos(\phi) + \mathbf{n} sin(\phi)$, vilket är en generalisering av Eulers identitet.

3.2.4) Låt $p_0, \hat{p}_0 \in Im\mathbb{H}, \quad q, \bar{q} \in \mathbb{H}_1$ och $p_1 = qp_0\bar{q}, \quad \hat{p}_1 = q\hat{p}_0\bar{q}$.
Då gäller att: $p_1\hat{p}_1 = qp_0\bar{q}q\hat{p}_0\bar{q} = qp_0\hat{p}_0\bar{q}$.
Skriv nu $p_0\hat{p}_0 = s + \mathbf{v}$ där s är realdelen och $\mathbf{v}$ de imaginära delarna av kvaternionen, då fås att $qp_0\hat{p}_0\bar{q} = qs\bar{q} + q\mathbf{v}\bar{q} = sq\bar{q} + q\mathbf{v}\bar{q} = s + q\mathbf{v}\bar{q}$. Detta visar på en gång både att skalärprodukten bevaras och att kryssprodukten bevaras i den mening att $p_1 \times \hat{p}_1 = q(p_0 \times \hat{p}_0)\bar{q}$.
Skriv nu $q = \hat{s} + \hat{\mathbf{v}}$. För att visa att avbildningen är kring en mängd av fixpunkter räcker det att studera $q\hat{\mathbf{v}}\bar{q}$.

$$q\hat{\mathbf{v}}\bar{q} = (\hat{s} + \hat{\mathbf{v}})\hat{\mathbf{v}}(\hat{s} - \hat{\mathbf{v}}) = \hat{s}^2\hat{\mathbf{v}} - \hat{s}\hat{\mathbf{v}}^2 + \hat{s}\hat{\mathbf{v}}^2 - \hat{\mathbf{v}}^3 = \hat{s}^2\hat{\mathbf{v}} - \hat{\mathbf{v}}^3 = (\hat{s}^2 - \hat{\mathbf{v}}^2)\hat{\mathbf{v}}$$

Eftersom $q$ är en enhetskvaternion är $\hat{s}^2 = cos^2(\phi)$ och $\hat{\mathbf{v}}^2 = sin^2(\phi) + (\hat{\mathbf{v}} \times \hat{\mathbf{v}}) = sin^2(\phi) + 0$ och kvar är då $(cos^2(\phi) + sin^2(\phi))\hat{\mathbf{v}} = \hat{\mathbf{v}}$. Alltså är avbildningen en rotation och ej en translation. □

**Sats 3.3.** *För varje rotation i rummet finns minst en kvaternion som motsvarar denna.*

Låt $\Omega \in SO(3)$ och definiera en funktion $\aleph : \Omega p \mapsto \theta_q p$. För att bevisa att $\aleph$ existerar krävs först att ordet skevsymmetrisk definieras:

**Definition 3.4.** *Låt A vara en matris, så att $A^T = -A$, A kallas då skevsymmetrisk.*

Om $A_1, A_2$ är skevsymmetriska gäller att $A = A_1A_2 - A_2A_1$ är skevsymmetrisk.

**Definition 3.5.** *En motsvarighet till skevsymmetriska matriser är hos kvaternioner en kvaternion vars konjugat är dess additiva invers. En sådan kvaternion kallas för en skevsymmetrisk kvaternion.*

Notera att en kvaternion är skevsymmetrisk om och endast om den är rent imaginär.

**Sats 3.6.** *Om $b, p$ är skevsymmetriska kvaternioner kommer även $\tau_b^n p$, vara skevsymmetrisk för alla n, där $\tau_b^n p$ definieras av den rekursiva formeln:*

$$\tau_b^0 p = p$$

$$\tau_b^n p = b(\tau_b^{n-1} p) - (\tau_b^{n-1} p)b = \sum_{k=0}^{n} \binom{n}{k} b^{n-k} p(-b)^k$$

*Notera att varje term i $\tau_b^n$ är av grad n (summan av exponenterna på våra $b = n$).*

*Bevis.* Att denna är skevsymmetrisk för alla n fås av induktion. Låt $*$ symbolisera komplexkonjugatet och $(\tau_b^n p)^* = -\tau_b^n p$ vara induktionsantagandet.

För $n = 0$ :

$(\tau_b^0 p)^* = p^* = -p = -\tau_b^0 p$.

Antag att det är sant för $n \geq 0$, då fås för $n + 1$:

$(\tau_b^{n+1} p)^* = (b(\tau_b^n p))^* - ((\tau_b^n p)b)^* = (\tau_b^n p)^* b^* - b^*(\tau_b^n p)^* \overset{ind.}{=} (-\tau_b^n p)(-b) - (-b)(-\tau_b^n p) = -\tau_b^{n+1} p$ □

*Bevis. Sats 3.3*
Beviset delas upp i tre delar för att underlätta läsning:
3.3.0) För varje $2 \times 2$ skevsymmetrisk matris $A$ är $e^A$ en rotation i två dimensioner.
3.3.1) För varje $\Omega \in SO(3)$ finns minst en $3 \times 3$ skevsymmetrisk matris $A$ så att $e^A = \Omega$.
3.3.2) För varje $e^A$, där $A$ är $3 \times 3$ skevsymmetrisk, finns minst ett motsvarande $\theta_q p \in Im\mathbb{H}$ och således finns för varje $\Omega \in SO(3)$ minst ett motsvarande $\theta_q p \in Im\mathbb{H}$.

3.3.0) Alla $2 \times 2$ skevsymmetriska matriser kan skrivas på formen $J\psi = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \psi$, där $J$ är $j$ på matrisform och $\psi \in \mathbb{R}$.
Räkningar som de i 3.2.3) ger att: $e^{J\psi} = \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix} \begin{pmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} =$
$\begin{pmatrix} \frac{e^{i\psi}+e^{-i\psi}}{2} & i\frac{e^{i\psi}-e^{-i\psi}}{2} \\ i\frac{-e^{i\psi}+e^{-i\psi}}{2} & \frac{e^{i\psi}+e^{-i\psi}}{2} \end{pmatrix} = \begin{pmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{pmatrix}$, som är en rotationsmatris i två dimensioner.
Nu inses lätt att om man tar en 3x3 matris som utför en rotation kring en koordinataxel, exempelvis $R_x(\psi)$ kan den skrivas som $exp \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\psi \\ 0 & \psi & 0 \end{pmatrix}$. Eftersom $R_y(\psi)$ och $R_z(\psi)$ bara är permutationer av $R_x(\psi)$ kan slutsatsen dras att rotation i två dimensioner alltid skrivas som $e^A$ för något $3 \times 3$ skevsymmetriskt $A$.

3.3.1) Eftersom $\Omega$ är en rotation kring en godtycklig axel genom origo kan vi välja en bas sådan att rotationsaxeln är en koordinataxel, d.v.s. $\Omega = YRY^T$ för en ortogonal basbytesmatris $Y$ och en rotationsmatris $R$ kring en koordinataxel. Detta kan då skrivas som $\Omega = Ye^BY^T = e^{YBY^T}$, där $B$ är skevsymmetrisk. Eftersom $(YBY^T)^T = (Y(-B)Y^T) = -(YBY^T)$ innebär det att om $B$ är skevsymmetrisk kommer $YBY^T$ också vara det och därför fås $\Omega = Ye^BY^T = e^{YBY^T} = e^A$, där $A$ är $3 \times 3$ skevsymmetrisk.

3.3.2) Låt $q = e^b \in \mathbb{H}_1$, där $b \in Im\mathbb{H}$ och studera $qpq^{-1}$. Då fås att:

$$qpq^{-1} \stackrel{def.}{=} e^b p e^{-b} = \lim_{N\to\infty} (\sum_{n=0}^{N} \frac{b^n}{n!}) p (\sum_{n=0}^{N} \frac{(-b)^n}{n!})$$

För att bilda sig en uppfattning om denna produkt studeras de första tre värdena på N.
$N = 0 : (1)p(1) = p = \tau_b^0 p$
$N = 1 : (1+b)p(1-b) = p + (bp - pb) - bpb = \tau_b^0 p + \tau_b^1 p + O(\frac{b^2}{1})$
$N = 2 : (1 + b + \frac{b^2}{2})p(1 - b + \frac{b^2}{2}) = ... = \tau_b^0 p + \tau_b^1 p + \tau_b^2 p + O(\frac{b^3}{2})$
$N = 3 : (1 + b + \frac{b^2}{2} + \frac{b^3}{6})p(1 - b + \frac{b^2}{2} - \frac{b^3}{6}) = ... = \tau_b^0 p + \tau_b^1 p + \tau_b^2 p + \tau_b^3 p + O(\frac{b^3}{6})$
Alla termer i $\tau_b^n p$ är av grad $n$ och vi får, när vi beräknar $(\sum_{n=0}^{N} \frac{b^n}{n!}) p (\sum_{n=0}^{N} \frac{(-b)^n}{n!})$ alla kombinationer från $\frac{b^0}{0!} p \frac{(-b)^0}{0!}$ till $\frac{b^N}{N!} p \frac{(-b)^N}{N!}$ (alla kombinationer av termer från grad 0 till $2N$). Eftersom varje term vi får är på formen $\frac{b^k}{k!} p \frac{(-b)^{n-k}}{(n-k)!} = \frac{1}{n!}\binom{n}{k} b^k p b^{n-k}$ där $k = 0, ..., n$ har vi $\sum_{n=0}^{N} \frac{\tau_b^n}{n!}$ och sedan resttermer som snabbast växer som $O(\frac{b^{N+1}}{N!})$.
Vi har således att $(\sum_{n=0}^{N} \frac{b^n}{n!}) p (\sum_{n=0}^{N} \frac{(-b)^n}{n!}) = \sum_{n=0}^{N} \frac{\tau_b^n}{n!} p + O(\frac{b^{N+1}}{N!})$ för alla $N$.
Detta innebär att $\theta_q p = e^b p e^{-b} = e^{\tau_b} p$ då $N \to \infty$.
Eftersom $\tau_b p$ motsvarar en skevsymmetrisk matris är $e^{\tau_b} p$ en rotation.
Alltså fås slutligen att om $\Omega \in SO(3)$ och $\hat{p}$ är en punkt i rummet och p dess motsvarande rent imaginära kvaternion så är $\Omega\hat{p} = e^A\hat{p} = e^{\tau_b} p = \theta_q p$ □

### 3.2.4 Representation av axel-vinkelrotation på kvaternionform

Ett sätt att representera en vridning med kvaternioner är att skriva dem på formen $cos(\frac{\theta}{2}) + \mathbf{n} sin(\frac{\theta}{2})$ där $\mathbf{n}$ är axeln vi vrider kring och $\theta$ är vinkeln i radianer.

**Sats 3.7.** *Om $q = cos(\phi) + \mathbf{n} sin(\phi)$ och $p \in Im\mathbb{H}$, kommer $qp\bar{q}$ rotera $p$ $2\phi$ radianer.*

*Bevis.* Vi studerar tre fall:
3.7.1) $p = \mathbf{n}$
3.7.2) $p \perp \mathbf{n}$
3.7.3) $p \not\perp \mathbf{n}$ och $p \neq \mathbf{n}$

3.7.1) $p = \mathbf{n}$ ger bara att $qp\bar{q} = p$, alltså att vi roterar kring en axel, se 3.2.4).
3.7.2) Antag $p \perp \mathbf{n}$:
$qp\bar{q} = (cos(\phi)+\mathbf{n}sin(\phi))p(cos(\phi)-\mathbf{n}sin(\phi)) = cos^2(\phi)p+\mathbf{n}pcos(\phi)sin(\phi)-p\mathbf{n}cos(\phi)sin(\phi)-\mathbf{n}p\mathbf{n}sin^2(\phi)$
Nu är $\mathbf{n}p = -(\mathbf{n} \bullet p) + (\mathbf{n} \times p) = 0 + (\mathbf{n} \times p)$
och $p\mathbf{n} = -(\mathbf{n} \times p)$
och $\mathbf{n}p\mathbf{n} = (\mathbf{n} \bullet (\mathbf{n} \times p)) - (\mathbf{n} \times (\mathbf{n} \times p)) = 0 - (\mathbf{n} \times (\mathbf{n} \times p)) = p$
så vi får $cos^2(\phi)p + 2cos(\phi)sin(\phi)(\mathbf{n} \times p) - sin^2(\phi)p = pcos(2\phi) + (\mathbf{n} \times p)sin(2\phi)$.
Alltså har vi en rotation på $2\phi$ radianer kring axeln $\mathbf{n}$.
3.7.3) Antag nu $p \not\perp \mathbf{n}$ och $p \neq \mathbf{n}$
Vi kan hitta en tredimensionell bas så att den består av $\mathbf{n}$ och två vektorer ortogonala med $\mathbf{n}$ och eftersom vi enligt ovan roterar både $\mathbf{n}$ och vektorerna ortogonala med $\mathbf{n}$ med $2\phi$ radianer kommer vi rotera $p$ med $2\phi$ radianer. [9] □

Sätt $2\phi = \theta$ och få formeln:

$$q = cos(\frac{\theta}{2}) + \mathbf{n}sin(\frac{\theta}{2})$$

Med ovanstående formel är det enkelt att ta fram axeln och vinkeln ur en given enhetskvaternion. Tag endast arcus cosinus av realdelen för att få ut vinkeln och skriv imaginärdelen på vektorform för att få ut axeln.

### 3.2.5 Representation av Eulervinklar på kvaternionform

Eulervinklar kan också skrivas på kvaternionform:

$$\langle \phi, 0, 0 \rangle \mapsto cos(\frac{\phi}{2}) + isin(\frac{\phi}{2})$$
$$\langle 0, \theta, 0 \rangle \mapsto cos(\frac{\theta}{2}) + jsin(\frac{\theta}{2})$$
$$\langle 0, 0, \psi \rangle \mapsto cos(\frac{\psi}{2}) + ksin(\frac{\psi}{2})$$

Detta gäller eftersom ett objekt som bara roteras kring en axel kan ses som ett system i $\mathbb{C}$, eftersom det endast har en realdel och en imaginärdel. Då kan Eulers identitet användas: $e^{i\frac{\phi}{2}} = cos(\frac{\phi}{2}) + isin(\frac{\phi}{2})$.

Om dessa vridningar sätts ihop till en (genom att multiplicera ihop de tre kvaternionerna) fås:

$$\langle \phi, \theta, \psi \rangle \mapsto q = s + ai + bj + ck$$
$$s = cos(\frac{\phi}{2})cos(\frac{\theta}{2})cos(\frac{\psi}{2}) + sin(\frac{\phi}{2})sin(\frac{\theta}{2})sin(\frac{\psi}{2})$$
$$a = sin(\frac{\phi}{2})cos(\frac{\theta}{2})cos(\frac{\psi}{2}) - cos(\frac{\phi}{2})sin(\frac{\theta}{2})sin(\frac{\psi}{2})$$
$$b = cos(\frac{\phi}{2})sin(\frac{\theta}{2})cos(\frac{\psi}{2}) + sin(\frac{\phi}{2})cos(\frac{\theta}{2})sin(\frac{\psi}{2})$$
$$c = cos(\frac{\phi}{2})cos(\frac{\theta}{2})sin(\frac{\psi}{2}) - sin(\frac{\phi}{2})sin(\frac{\theta}{2})cos(\frac{\psi}{2})$$

För att få ut Eulervinklarna från en given kvaternion är det vanligt att man skriver om kvaternionen till en rotationsmatris och sedan skriver om matrisen till Eulervinklar. Detta

behövs dock inte, utan det räcker att studera relevanta element ur matrisen, vilka ger tre formler, en för varje vinkel.

Om vinkeln $\theta \neq \frac{\pi k}{2}, k = 2n+1, n \in \mathbb{Z}$ fås:

$$cos(\theta) = \sqrt{1-(2ac-2sb)^2}$$
$$cos(\psi) = \frac{1-2a^2-2b^2}{cos(\theta)}$$
$$cos(\phi) = \frac{1-2b^2-2c^2}{cos(\theta)}$$

Om vinkeln $\theta = \frac{\pi k}{2}$ blir $cos(\theta) = 0$ och eftersom vi inte får dividera med noll kan vi inte använda ovanstående formler. Då utnyttjas det faktum att vi inte längre kan se skillnad på en vridning av x- och z-axlarna och då är:

$$cos(\theta) = 0$$
$$cos(\phi) = 1-2a^2-2c^2$$
$$cos(\psi) = 1$$

Här måste vi dock ta reda på vilket tecken $\theta$ har, eftersom det inte avslöjas i formeln. Då utnyttjar vi att $sin(\theta) = 2rb - 2ac$ och vi kan då se tecknet på $\theta$. [13]

### 3.2.6 Metod att finna kvaternionen som roterar en punkt till en annan

Det kan i vissa fall vara relevant att finna rotationen mellan två punkter i rummet (om exempelvis en robot vill rotera sin arm från en position till en annan). Att göra detta med kvaternioner är väldigt enkelt.

**Sats 3.8.** *Om $p_1$ och $p_2$ betecknar två punkter i $Im\mathbb{H}$ med samma avstånd från origo kan rotationen mellan dem skrivas som $qp_1\bar{q} = p_2$ där $q = \frac{\hat{q}}{||\hat{q}||}$ och $\hat{q}$ ges av:*
*1) $Re(\hat{q}) = |p_1||p_2| + (p_1 \bullet p_2)$*
*2) $Im(\hat{q}) = p_1 \times p_2$*

*Bevis.* Studera

$$q = cos(\frac{\theta}{2}) + \mathbf{n}sin(\frac{\theta}{2})$$

där $\theta = arccos(\frac{p_1 \bullet p_2}{|p_1||p_2|})$ är vinkeln mellan $p_1$ och $p_2$ och $\mathbf{n} = \frac{p_1 \times p_2}{|p_1||p_2|sin(\theta)}$.
Först används $sin(2\theta) = 2sin(\theta)cos(\theta) \Rightarrow sin(\theta) = \frac{sin(2\theta)}{2cos(\theta)} \Rightarrow sin(\frac{\theta}{2}) = \frac{sin(\theta)}{2cos(\frac{\theta}{2})}$.
Detta ger:

$$q = cos(\frac{\theta}{2}) + \mathbf{n}\frac{sin(\theta)}{2cos(\frac{\theta}{2})} \Rightarrow 2cos(\frac{\theta}{2})q = 2cos^2(\frac{\theta}{2}) + \mathbf{n}sin(\theta)$$

Nu är $cos^2(\frac{\theta}{2}) = \frac{1+cos(\theta)}{2}$ vilket ger:

$$2cos(\frac{\theta}{2})q = (1+cos(\theta)) + \mathbf{n}sin(\theta)$$

Studeras $\mathbf{n}$ fås att:
$\mathbf{n} = \frac{p_1 \times p_2}{|p_1||p_2|sin(\theta)} \Rightarrow \mathbf{n}sin(\theta) = \frac{p_1 \times p_2}{|p_1||p_2|}$

$$2cos(\frac{\theta}{2})q = (1+cos(\theta)) + \frac{p_1 \times p_2}{|p_1||p_2|}$$

Ur den geometriska tolkningen av skalärprodukt fås att:

$p_1 \bullet p_2 = |p_1||p_2|cos(\theta) \Rightarrow cos(\theta) = \frac{p_1 \bullet p_2}{|p_1||p_2|}$

$$2cos(\frac{\theta}{2})q = (1 + \frac{p_1 \bullet p_2}{|p_1||p_2|}) + \frac{p_1 \times p_2}{|p_1||p_2|}$$

Multipliceras allt med $|p_1||p_2|$ fås:

$$\hat{q} := 2cos(\frac{\theta}{2})|p_1||p_2|q = (|p_1||p_2| + p_1 \bullet p_2) + p_1 \times p_2$$

Normera $\hat{q}$ och kvar är enhetskvaternionen $q$ som uppfyller $qp_1\bar{q} = p_2$. [4] □

## 3.3 Sammanfattning av rotationers parametriseringar och representationer

I detta kapitel har fyra sätt att representera vridningar på åskådliggjorts:

- Eulervinklar på matrisform:

$$R_z(\gamma)R_y(\beta)R_x(\alpha) =$$

$$\begin{pmatrix} cos(\beta)cos(\gamma) & cos(\gamma)sin(\alpha)sin(\beta) - cos(\alpha)sin(\gamma) & sin(\alpha)sin(\gamma) + cos(\alpha)cos(\gamma)sin(\beta) \\ cos(\beta)sin(\gamma) & cos(\alpha)cos(\gamma) + sin(\alpha)sin(\beta)sin(\gamma) & cos(\alpha)sin(\beta)sin(\gamma) - cos(\gamma)sin(\alpha) \\ -sin(\beta) & cos(\beta)sin(\alpha) & cos(\alpha)cos(\beta) \end{pmatrix}$$

- Axel-vinkel på matrisform:

$$R_{\mathbf{n}}(\alpha) =$$

$$\begin{pmatrix} cos(\alpha) + n_1^2(1 - cos(\alpha)) & n_1n_2(1 - cos(\alpha)) - n_3sin(\alpha) & n_1n_3(1 - cos(\alpha)) + n_2sin(\alpha) \\ n_2n_1(1 - cos(\alpha)) + n_3sin(\alpha) & cos(\alpha) + n_2^2(1 - cos(\alpha)) & n_2n_3(1 - cos(\alpha)) - n_2sin(\alpha) \\ n_3n_1(1 - cos(\alpha)) - n_2sin(\alpha) & n_3n_2(1 - cos(\alpha)) + n_1sin(\alpha) & cos(\alpha) + n_3^2(1 - cos(\alpha)) \end{pmatrix}$$

- Eulervinklar på kvaternionform:

$$q = r + ai + bj + ck$$

$$r = cos(\frac{\phi}{2})cos(\frac{\theta}{2})cos(\frac{\psi}{2}) + sin(\frac{\phi}{2})sin(\frac{\theta}{2})sin(\frac{\psi}{2})$$

$$a = sin(\frac{\phi}{2})cos(\frac{\theta}{2})cos(\frac{\psi}{2}) - cos(\frac{\phi}{2})sin(\frac{\theta}{2})sin(\frac{\psi}{2})$$

$$b = cos(\frac{\phi}{2})sin(\frac{\theta}{2})cos(\frac{\psi}{2}) + sin(\frac{\phi}{2})cos(\frac{\theta}{2})sin(\frac{\psi}{2})$$

$$c = cos(\frac{\phi}{2})cos(\frac{\theta}{2})sin(\frac{\psi}{2}) - sin(\frac{\phi}{2})sin(\frac{\theta}{2})cos(\frac{\psi}{2})$$

- Axel-vinkel på kvaternionform:

$$cos(\frac{\theta}{2}) + \mathbf{n}sin(\frac{\theta}{2})$$

# 4 Topologiska skilnader mellan $SO(3)$ och $\mathbb{H}_1$

I de kommande avsnitten ska vi närmre undersöka hur rotationsgruppen och gruppen av enhetskvaternioner under multiplikation hänger ihop. Det ska visa sig att de två grupperna, även om de kan verka lika vid första anblick, skiljer sig fundamentalt. Det ska även visa sig hur de olika grupperna, trots sina olikheter, går att koppla samman. Då rotationsgruppen är isomorf med gruppen av ortogonala matriser med determinant 1 under multiplikation, används beteckningen $SO(3)$ konsekvent för att beteckna rotationsgruppen.

## 4.1 Kvaternioner kan läggas på en boll

Som tidigare visats kan rotationer av vektorkvaternioner utföras genom multiplikation med ett par kvaternioner som är varandras invers. Alla nollskilda kvaternioner kan ingå i ett sådant par, men för att underlätta beräkningarna valdes enhetskvaternioner, då dessas invers också är deras konjugat. Tidigare har visats att alla skalärmultipler av en kvaternion representerar samma vridning, så man kan låta varje kvaternion $q$ representeras av enhetskvaternionen $\frac{q}{||q||}$ och därmed få en behändigare mängd att studera. Då $\mathbb{H}_1$ i någon mening är mängden av alla punkter med ett givet avstånd till en bestämd punkt, i detta fall 0, kan den ses som en generaliserad sfär. En sådan sfär betecknas ibland $S^3$, $S$ för sfär och 3 för de tre dimensionerna i sfärens yta. $S^3$ är isomorf med $\mathbb{H}_1$, och de slutsatser som dras om $\mathbb{H}_1$ gäller också för $S^3$.

**Definition 4.1.** *Man säger att en mängd $M$ är **enkelt sammanhängande** om varje sluten kurva i $M$ kontinuerligt kan omformas till varje annan sluten kurva i $M$ och speciellt till varje punkt i $M$.*

**Sats 4.2.** *$\mathbb{H}_1$ är enkelt sammanhängande.*

Ett konstruktivt bevis är som följer:

*Bevis.* I beviset används kvaternioner på formen $q = cos(\frac{\theta}{2}) + \mathbf{n}sin(\frac{\theta}{2})$. Välj en sluten kurva på $\mathbb{H}_1$. Om kvaternionen $-1$ inte ligger på kurvan låter man kontinuerligt $\theta$ gå mot 0. Kurvan kommer då att närma sig kvaternionen 1 och beviset är färdigt. Kvaternionen $-1$ har flera möjliga representationer. Om $-1$ ligger på kurvan, välj $\mathbf{n}$ som gränsvärdet av $\mathbf{n}$ när kurvan närmar sig. Existerar två gränsvärden så tag medelvärdet av gränsvärdena. Existerar inget gränsvärde så rotera hela kurvan runt sfären tills $-1$ inte längre ligger på kurvan. Förfarandet kan sedan fortsätta som ovan. □

## 4.2 Rotationer kan läggas i en boll

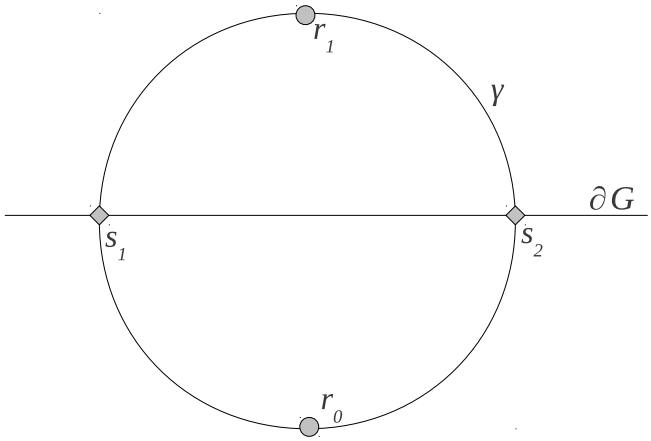
I detta avsnitt undersöks närmre mängden $SO(3)$ av alla rotationer i rummet. Mängden rotationer är lik sfären från föregående avsnitt såtillvida att den är rund, åtminstone i den mening att man efter en serie identiska förflyttningar i mängden kan komma tillbaka till sitt utgångsläge.

Vi ska nu visa hur varje rotation i rummet kan identifieras med en punkt i en delmängd $G$ av $\mathbb{R}^3$ sådan att $G = \{x \in \mathbb{R}^3 : |x| \leq \pi\}$[1]. En rotation bestäms entydigt, som tidigare visats, av en rotationsaxel och en vinkel. Givet en rotation $R_{\mathbf{v}_1}(\alpha)$ bestämd av den positiva vinkeln $\alpha$ kring enhetsvektorn $\mathbf{v}_1$, identifiera $R_{\mathbf{v}_1}(\alpha)$ med en vektor $v \in G$ sådan att $v = \alpha\mathbf{v}_1$. Om $\alpha > \pi$, välj en vinkel $\beta$ sådan att $\alpha + \beta = 2\pi$ och använd att $R_{\mathbf{v}_1}(\alpha) = R_{-\mathbf{v}_1}(\beta)$ och välj vektorn $-\beta\mathbf{v}_1$. Till varje rotation kan alltså på detta sätt väljas en vektor $v$ i bollen $G$. Likheten $R_z(\pi) = R_z(-\pi)$ motiverar att $\pi\mathbf{v}_1$ identifieras med $-\pi\mathbf{v}_1$ för varje $\mathbf{v}_1$. Detta är nyckeln till den olikhet mellan $SO(3)$ och $\mathbb{H}_1$ som detta avsnitt behandlar då det ger följande sats:

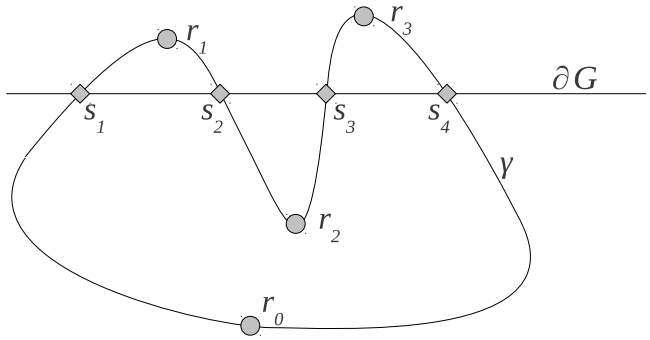
**Sats 4.3.** *Mängden av rotationer i rummet är inte enkelt sammanhängande.*

*Bevis.* Om man kan konstruera en sluten kurva $\gamma$ i $G$ som inte kontinuerligt kan deformeras till varje punkt i $G$ är satsen bevisad. För att kunna omforma $\gamma$ till att enbart omfatta origo måste $\gamma$ också kunna formas om till en liten cirkel $\gamma_0$ kring origo. På samma sätt

måste $\gamma_0$ kunna omformas till $\gamma$ och det räcker således att utreda vilka kurvor $\gamma$ som kan fås kontinuerligt av $\gamma_0$. Kurvor som inte passerar randen $\partial G$ till $G$ är helt ekvivalenta med kurvor i $\mathbb{R}^3$ och kan skapas kontinuerligt från $\gamma_0$ då det lätt inses att $\mathbb{R}^3$ är enkelt sammanhängande. Har man däremot en kurva $\gamma$ och låter en punkt $r_1$ passera medan en annan punkt $r_0$ inte passerar kommer detta att resultera i minst två skärningspunkter $s$, se bild.




För att $r_1$ fortfarande ska vara kopplad till $r_0$ måste $\gamma$ korsa randen minst två gånger om den ska förbli sluten. Låter man sedan $\gamma$ korsa randen fler gånger finner man att antalet skärningspunkter är samma som antalet segment som $\gamma$ delas i av $\partial G$, se bild:




Då $\gamma$ är sluten måste detta antal vara jämnt om $\gamma$ är kontinuerligt omformad från $\gamma_0$. Med detta i åtanke, betrakta nu någon kurva som motsvarar en vridning ett varv kring någon axel. Dessa utgörs av räta linjer genom [0,0,0] mellan två motstående punkter på $\partial G$ och har därigenom endast en skärningspunkt. Då en sådan kurva därigenon inte kan omformas från $\gamma_0$ kan den heller inte omformas till en punkt. Existensen av en sådan kurva bevisar satsen. □

Resonemanget bevisar även följande sats, som kommer till användning i nästa avsnitt.

**Sats 4.4.** *En sluten kurva $\gamma$ korsar bollens rand ett jämnt antal gånger omm den kan deformeras till en punkt.*

Då mängden $SO(3)$ av rotationer inte är enkelt sammanhängande kan ingen operation göra $SO(3)$ till en grupp isomorf med någon grupp över mängden $\mathbb{H}_1$. Man behöver alltså hitta någon annan konstruktion för att koppla ihop enhetskvaternionerna med $SO(3)$.

## 4.3 Rotationerna är dubbelt övertäckta av enhetskvaternionerna

Det som hindrar $G$ från att vara enkelt sammanhängande är, enligt 4.4, att man kan skapa slutna kurvor som skär $\partial G$ ett udda antal gånger. För att $\mathbb{H}_1$ ska kunna användas isomorft

med rotationer behöver $G$ förändras. Detta kan åstadkommas genom att tilldela mängden två "sidor" och säga att en kurva som korsar randen byter sida.

**Definition 4.5.** *Definiera en mängd $\hat{G}$ så att $\hat{G}$ är $\mathbb{Z}_2 \times G$ där $(r, 0)$ identifieras med $(-r, 1)$ för $r \in \partial G \subset G$.*

Alla försök att flytta slutna kurvor med udda antal skärningar från $G$ till $\hat{G}$ misslyckas så länge som $(r, 0)$ och $(-r, 1)$ betraktas som icke intilliggande för alla $r \notin \partial G \subset G$. Att koppla $\hat{G}$ till enhetskvaternionerna är nu möjligt: Associera helt enkelt enhetskvaternioner med positiv skalärdel med den ena sidan, säg 0, och enhetskvaternioner med negativ realdel med 1. Detta motsvarar att "skjuta in" klotet till vilket $\mathbb{H}_1$ är begränsningsarea mellan de två sidorna. I tre dimensioner kan lägga en cirkelskiva på vardera sidan av ett klot, fästa ihop kanterna och få en sfär. På samma sätt som $S^2$ kan delas upp i två cirkelskivor kan på detta sätt $\mathbb{H}_1 = S^3$ delas upp i två klot. Punkterna $[0, 0, 0]$ i de bägge sidorna passas till kbaternionerna 1 respektive -1 i $\mathbb{H}_1$. För att få en naturligare koppling till enhetssfären kan man vrida en av sidorna, säg sidan 1, ett halvt varv kring den reella axeln och på så vis slippa "hoppet" från rotationer $r$ till rotationer $-r$ när man passerar kvaternioner med skalärdel 0.

Överensstämmelsen mellan vår parametriseing av rotationerna och vår parametrisering av enhetskvaternioner som representation av rotationer är nu uppenbar: För att rotera en vinkel $\theta$ kring en axel given av kvaternionvektorn $\mathbf{n}$ valdes kvaternionen $q = cos\frac{\theta}{2} + \mathbf{n}sin\frac{\theta}{2}$. Precis som i $G$ motsvaras vinkeln av avståndet till en mittpunkt, i detta fall till kvaternionen 1 längs sfären $\mathbb{H}_1$, och valet av rotationsaxel av riktningen från mittpunkten. I mängden $\hat{G}$ finns varje rotation representerad på två ställen på sidorna 0 och 1. Detta gäller även för kvaternionerna då $qp\bar{q} = (-q)p(\bar{-q})$ vilket innebär att $q$ och $-q$ motsvarar samma vridning.

Sammanfattningsvis så gäller alltså att även om man inte kan skapa en isomorfi direkt mellan rotationer och kvaternioner så låter det sig göras mellan rotationer och två sammansatta mängder rotationer. Vi kan därmed precisera sats 3.3 som säger oss att det till varje rotation finns minst en kvaternion:

**Sats 4.6.** *Till varje rotation finns exakt två enhetskvaternioner som motsvarar denna. Vi säger att $SO(3)$ är **dubbelt övertäckt** av $\mathbb{H}_1$.*

# 5 Förlust av frihetsgrader hos Eulervinklar

När man väljer att representera rotationer med Eulervinklar medför det några inneboende problem som måste hanteras. Ett av dessa, som inte uppträder med axel-vinkelrepresentationen, är förlust av frihetsgrader för vissa rotationer. I nästan alla fall kan man genom att variera en av de tre vinklarna rotera koordinatsystemet kring tre unika axlar. Det existerar dock fall då då två av dessa axlar sammanfaller, så att man genom att kontinuerligt ändra vinklarna endast kan röra sig i två dimensioner i den tredimensionella mängden $SO(3)$. Det är detta vi kallar förlust av frihetsgrader.

Det finns många sätt att dela upp en rotation i tre rotationer kring fixa axlar, men vilket sätt man än väljer så uppstår problemet någonstans. Först visas detta för den uppdelning som används i avsnittet där vi definierat Eulervinklar och sedan ges ett generellt bevis för alla uppdelningar i tre vinklar. Till sist skissas ett bevis för att fenomenet uppstår även om man introducerar fler än tre axlar.

Betrakta först uppdelningen som i tidigare avsnitt benämnts $z$-$y$-$x$. De sfäriska koordinater som vanligtvis används för att ange geografiska positioner kan användas som hjälp när man försöker visualisera rotationerna som vridningar som sker en i taget. Man kan tänka sig en vektor från jordens kärna ut till punkten där nollmeridianen skär ekvatorn. Man låter sedan vektorn ändra längd- och breddgrad i den ordningen och vrider till sist vektorn kring sig själv. Om ett ortogonalt koordinatsystem där sagd vektor, nordpolen och en punkt på ekvatorn ingår som vrids på detta sätt kommer $\phi$ att ges av vektorns längdgrad, $\theta$ av vektorns breddgrad och $\psi$ av vinkeln för den sista vridningen. Beroende på hur man väljer den sista axeln i koordinatsystemet kan tecknet på vinklarna variera.

Problemen uppstår, om vi ser på det som det presenterats nu senast, när breddgraden är $\pm 90°$ och vektorn alltså roterats till någon av polerna, säg nordpolen. De kommer sig av att man kan representera en vridning vid n som antingen en vridning runt axeln axeln själv eller också som en ändring av den valda längdgraden. Varje rotation får därigenom flera representationer då den första och den sista vridningen helt eller delvis kan ta ut varandra eller förstärka varandra. Det finns två sätt att se på detta och dessa formuleras i varsin sats:

**Sats 5.1.** *Det finns rotationer med överuppräknerligt många representanter.*

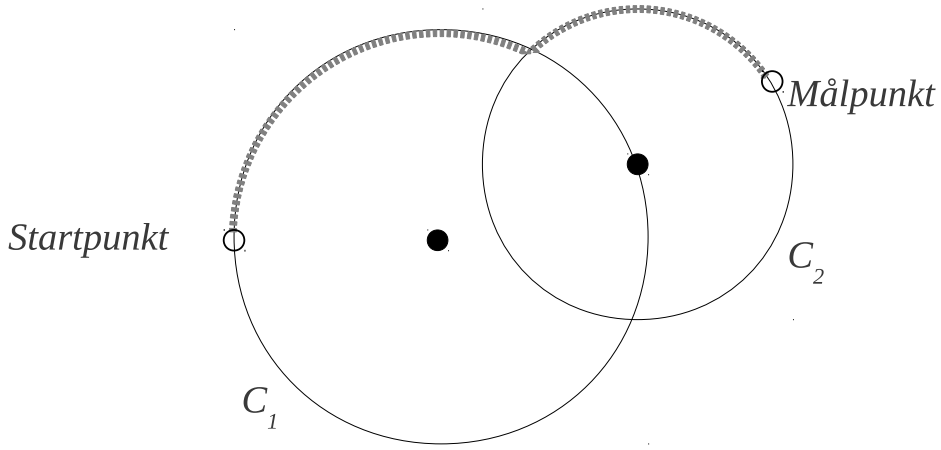
**Sats 5.2.** *Representationerna av två närbelägna rotationer behöver inte sinsemellan vara närbelägna. Per definition innebär detta att det i någon punkt finns en diskontinuitet.*

Sats 5.1 behöver inte orsaka några problem om målet enbart är att ange hur något är roterat. Till exempel kan införas en konvention om att vridningar kring nordpolen har längdgrad 0. Sats 5.2 följer av sats 5.1 och orsakar mer problem. Väljer man att alltid representera en vridning med de tre till beloppet minsta möjliga vinklarna så kan det i termer av jordkordinater formuleras som följer: En punkt på nollmeridianen och den punkt som ligger mitt emot över nordpolen sett har väldigt olika koordinater även om de ligger väldigt nära varandra. För att visa detta matematiskt, betrakta rotationerna $r_1$ och $r_2$ med representationer $E(r_1) = \langle 0°, \alpha, \beta \rangle$ respektive $E(r_2) = \langle 180°, \alpha, -\beta \rangle$ för några $\alpha$ och $\beta$. Låts punkterna närma sig varandra så att $\alpha \to 90°$ fås att $r_1 \to r_2$ trots att $E(r_1) \not\to E(r_2)$. Detta betyder, per definition, att avbildningen $E$ inte är kontinuerlig vilket bevisar satsen .

För att kunna formulera ett allmännare bevis behövs ett lemma:

**Lemma 5.3.** *För att en sammansättning av tre rotationer kring förutbestämda axlar ska kunna ge alla rotationer i $SO(3)$ måste varje axel vara vinkelrät mot den föregående.*

*Bevis.* Såvida inte alla de tre axlarna är linjärt beroende är påståendet ekvivalent med att de tre rotationerna kan ställa dessa tre axlar i varje möjligt läge. För detta krävs att sammansättningen av de två första rotationerna kan ställa den sista rotationsaxeln i varje möjligt läge. Väljs alla axlar som enhetsvektorer kan man betrakta de banor i vilka vektorerna rör sig på enhetssfären då man vrider kring axlarna en i taget. Dessa kommer alla att vara uppbyggda av cirkelsegment. Centrum för alla cirklar av vilka dessa segment är en del kommer att ligga på randen till någon föregående cirkel, förutsatt att alla avstånd mäts längs enhetssfären.




Varje cirkels radie längs enhetssfären blir identisk med en mellanliggande vinkel för två axlar. Figuren visar hur den sista rotationsaxeln rör sig runt de andra två längs cirklar, kallade $C_1$ och $C_2$. Vinkeln mellan första och andra axeln är radien i $C_1$ och vinkeln mellan andra och tredje är radien i $C_2$. Först noteras att cirklarna måste ha samma radie, ty annars existerar en cirkelskiva kring centrum för $C_1$ dit inga vägar kan finnas. Sedan noteras att om en cirkel har en radie större än $\frac{\pi}{2}$ så är den ekvivalent med en mindre cirkel kring sitt centrums antipol på enhetssfären. Därigenom kan alla cirklar sägas ha en radie mindre än eller lika med $\frac{\pi}{2}$. Skulle radierna vara mindre än $\frac{\pi}{2}$ skulle det dock uppstå en icke nåbar cirkelskiva kring antipolen till centrum för cirkeln $C_1$, så likhet måste gälla. □

Med ovanstående lemma kan följande sats bevisas:

**Sats 5.4.** *För alla uppdelningar av rotationer i någon sekvens av tre rotationer med vinklar $\alpha_1, \alpha_2, \alpha_3$ kring förutbestämda axlar $\mathbf{x}_1$, $\mathbf{x}_2$ och $\mathbf{x}_3$ gäller att ingen surjektiv avbildning $E$ : $SO(3) \ni r \mapsto \langle \alpha_1, \alpha_2, \alpha_3 \rangle$ kan vara kontinuerlig.*

*Bevis.* För att bevisa satsen räcker att visa att den inversa avbildningen $E^{-1} : \langle \alpha_1, \alpha_2, \alpha_3 \rangle \to SO(3)$ avbildar en sammanhängande överuppräknerlig mängd vinklar till samma rotation i $SO(3)$. Detta görs genom att visa att det existerar vinklar $\alpha_1$ och $\alpha_2$ sådana att

$$R_{\mathbf{x}_2}(\alpha_2) R_{\mathbf{x}_1}(\alpha_1) = R_{\mathbf{x}_3}(\alpha_1) R_{\mathbf{x}_2}(\alpha_2) \tag{3}$$

eller efter omskrivning

$$R_{\mathbf{x}_2}(\alpha_2) R_{\mathbf{x}_1}(\alpha_1) R_{\mathbf{x}_2}(-\alpha_2) = R_{\mathbf{x}_3}(\alpha_1) \tag{4}$$

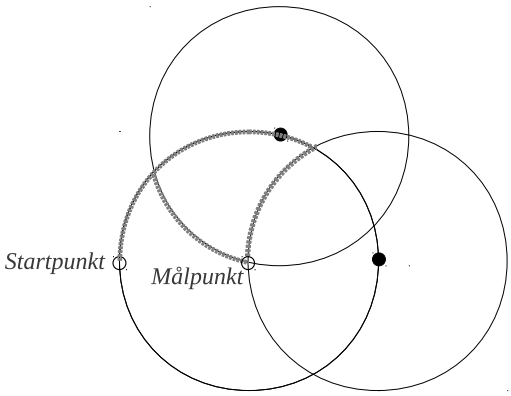
Enligt lemma 5.3 gäller att $\mathbf{x}_1 \perp \mathbf{x}_2$ samt att $\mathbf{x}_2 \perp \mathbf{x}_3$. Detta innebär att ett ortogonalt koordinatsystem kan definieras i vilket $\mathbf{x}_1 = [1,0,0]^T$, $\mathbf{x}_2 = [0,1,0]^T$ och $\mathbf{x}_3 = [cos\theta, 0, sin\theta]^T$ för något $\theta$. Vänsterledet i 4 kan då skrivas

$$\begin{pmatrix} cos(\alpha_2) & 0 & sin(\alpha_2) \\ 0 & 1 & 0 \\ -sin(\alpha_2) & 0 & cos(\alpha_2)) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha_1) & sin(\alpha_1) \\ 0 & -sin(\alpha_1) & cos(\alpha_1)) \end{pmatrix} \begin{pmatrix} cos(\alpha_2) & 0 & -sin(\alpha_2) \\ 0 & 1 & 0 \\ sin(\alpha_2) & 0 & cos(\alpha_2)) \end{pmatrix} = ... =$$
$$\begin{pmatrix} cos(\alpha_1) + cos^2(\alpha_2)(1 - cos(\alpha_1)) & -sin(\alpha_1)sin(\alpha_2) & (1 - cos(\alpha_1))cos(\alpha_2)sin(\alpha_2) \\ sin(\alpha_1)sin(\alpha_2) & cos(\alpha_1) & -sin(\alpha_1)cos(\alpha_2) \\ (1 - cos(\alpha_1))cos(\alpha_2)sin(\alpha_2) & sin(\alpha_1)cos(\alpha_2) & cos(\alpha_1) + sin^2(\alpha_2)(1 - cos(\alpha_1)) \end{pmatrix}$$

Vid isättning av elementen i $\mathbf{x}_3$ i matrisen i sats 3.1 undersöks lätt att varje vridning med vinkeln $\alpha_1$ kring $\mathbf{x}_1$ har samma matris som vänsterledet i ekvationen ovan. Detta får till följd att då $\alpha_2 = \theta$ gäller $E^{-1}(\langle \alpha_1, \alpha_2, \alpha_1 \rangle) = E^{-1}(\langle \alpha_3, \alpha_2, \alpha_3 \rangle)$ för alla $\alpha_1$, $\alpha_3$ och satsen är bevisad. □

Det nyss givna beviset gäller uppdelningar i tre vridningar. För att se att diskontinuiteten uppstår oavsett antalet vridningar utnyttjas att Sats 5.4 kan även ges ett annat, mer visuellt bevis.

*Bevis.* I detta bevis åskådliggörs vridningarna kring de två första axlarna på samma sätt som i Lemma 5.3. Betrakta bilden från beviset igen. Då de två cirklarna måste ha samma radie så kommer $C_2$ oundvikligen att skära centrum för $C_1$. Det betyder att det finns ett läge för den tredje axeln som det finns flera sätt att uppnå, se figur.




Ovanstående resonemang visar existensen av trippler av vinklar som avbildas på samma rotation i $SO(3)$ vilket bevisar satsen. □

Lemma 5.3 kan lätt generaliseras till att säga att för varje uppdelning i $n$ rotationer måste gälla att summan av de mellanliggande vinklarna till de $n-1$ första axlarna måste uppgå till $\pi$. Man kan sedan med ett resonemang liknande beviset ovan inse att det uppstår minst en punkt till vilken den sista axeln kan vridas på oändligt många sätt.

Förevarande stycke motiverar följande sats, vilken får avsluta avsnittet:

**Sats 5.5.** *För alla uppdelningar av rotationer i någon sekvens av något antal rotationer med vinklar $\alpha_1, \alpha_2, ..., \alpha_n$ kring förutbestämda axlar $\mathbf{x}_1$, $\mathbf{x}_2$, ...,$\mathbf{x}_n$ gäller att ingen surjektiv avbildning $E : SO(3) \ni r \mapsto \langle \alpha_1, \alpha_2, ..., \alpha_n \rangle$ kan vara kontinuerlig.*

# 6 Numerik

Vid datorberäkningar med rotationer, precis som i alla numeriska beräkningar, är det huvudsakligen två aspekter som är viktiga:

Den första är beräkningstiden: det måste gå snabbt att utföra beräkningarna. I en datoranimering måste alla beräkningar vara klara innan nästa bild i animeringen ska visas på skärmen. Om inte detta sker snabbt kommer bilden att hacka, eftersom datorn måste vänta på att beräkningarna blir klara innan nästa bild kan presenteras.

Den andra är stabiliteten: om punkten $a$ ska roteras till en annan punkt $b$, då är det väsentligt att det program som används för beräkning av rotationen verkligen genererar punkt $b$ som resultat av rotationen. För att klassa en metod som bra kräver vi att så kommer vara fallet. Hur stort ett acceptabelt fel är beror på tillämpningen, men i kommande simuleringar anses ett fel i storleksordningen $10^{-15} - 10^{-16}$ som godtagbart. Detta motiveras av att en dator vanligtvis räknar med 15-16 gällande siffror, så bättre noggrannhet än så går inte att få.

I beräkningar av rotationer med hjälp av kvaternioner används formeln $p_n = q_n p_{n-1} \overline{q_n}$. Vid beräkning av många rotationer beräknas den produkten av kvaternioner många gånger, och för att kunna få upp hastigheten i beräkningar krävs det att beräkningarna sker på ett effektivt sätt. Dessutom går det åt tid till att skapa rotationsmatriserna respektive kvaternionerna för rotationerna. Vid en jämförelse av olika metoder för att räkna med kvaternioner är det viktigt att alla dessa tider tas i beaktan. Därför kommer till att börja med tre små program köras: i det första programmet jämförs 3 sätt att beräkna $qp\bar{q}$, och i det andra undersöks hur lång tid det tar att givet en rotationsaxel och rotationsvinkel skapa den matris respektive kvaternion som krävs för rotationen. Det tredje programmet kommer göra en simulering som undersöker sammansättningar av en sekvens (identiska) rotationer givna på axel-vinkelparametrisering.

## 6.1 Beräkning av rotationsmatris respektive rotationskvaternion

Följande korta simulering visar hur lång tid det tar att skapa rotationsmatrisen respektive kvaternionen för en rotation givet en normerad rotationsaxel $\mathbf{n}$ och en rotationsvinkel $\theta$. För att bestämma rotationsmatrisen används formel 3.1, och för kvaternionen enbart formeln $q = cos(0.5\theta) + \mathbf{n}sin(0.5\theta)$. En snabb körning i MATLAB ger följande resultat:

```
Skapande av 500000 rotationsmatriser: 2.2452 sekunder.
Skapande av 500000 rotationskvaternioner: 1.5818 sekunder.
```

Det går alltså ca 1.42 gånger långsammare att skapa en rotationsmatris än vad det gör att skapa en kvaternion som utför motsvarande rotation. Detta förklaras genom att när datorn skapar en rotationsmatris är det 9 element som måste beräknas, där varje element kräver ungefär 3 multiplikationer och 2 additioner, samt en sinus- och en cosinusberäkning. Vid skapandet av kvaternioner behöver fortfarande cosinus och sinusvärdet beräknas, men därefter är 3 multiplikationer allt som behövs för att få kvaternionen.

## 6.2 Beräkning av $qp\bar{q}$

Vid beräkning av rotationer i MATLAB via matriser används MATLABs inbyggda rutin för matris-vektor multiplikation. Hur $qp\bar{q}$ beräknas snabbast är inte lika uppenbart, varför vi jämför tre olika metoder i MATLAB att göra detta:

1. En rutin för att beräkna produkten enligt första likheten i lemma 2.2 implementeras. Detta är kanske det mest intuitiva sättet att beräkna produkten på.

2. Här används istället sista likheten i lemma 2.2, dvs $q_1 q_2 = (s_1 s_2 - (\mathbf{v}_1 \bullet \mathbf{v}_2), s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$, och resultatet förenklas till ett uttryck som innehåller så få beräkningar som möjligt.

3. I och med att MATLAB har väldigt optimerade rutiner för matris-vektoroperationer ser vi kvaternioner som komplexa $2 \times 2$-matriser och använder den inbyggda rutinen för

matrismultiplikation för att multiplicera ihop matriserna. På så sätt får resultatet av kvaternionmultiplikationen i form av en $2 \times 2$ komplex matris, ur vilken kvaternionen på formen $s + ai + bj + ck$.

Formeln som används i metod 2 ges av följande faktum: Låt $\mathbf{p} \in \mathbb{R}^3$. Om $p = (0, \mathbf{p})$ är den kvaternion som svarar mot $\mathbf{p}$ och $q = (s, \mathbf{v})$ en rotation som ska appliceras på $\mathbf{p}$ gäller för resultatet av kvaternionmultiplikationen $p_{ny} = (0, \mathbf{p_{ny}}) = qpq^{-1}$ att $\mathbf{p_{ny}} = \mathbf{p} + (2\mathbf{v}) \times (\mathbf{v} \times \mathbf{p}) + s\mathbf{p}$.

En MATLAB-kod som beräknar produkten $qp\bar{q}$ 10000 gånger för var och en av de ovanstående metoderna finns i appendix. En körning av den koden ger följande utskrift:

```
Metod 1: Elapsed time is 0.26907 seconds.
Metod 2: Elapsed time is 1.7953 seconds.
Metod 3: Elapsed time is 0.035778 seconds.
```

Vid en jämförelse av de numeriska resultaten ses att de är identiska på alla 16 decimaler som vi har till vårt förfogande. Baserat på detta kommer simulering 1 att se kvaternioner som komplexa $2 \times 2$-matriser och använda MATLABs inbyggda rutiner för matrismultiplikation. I simulering 2 undersöks om det går snabbare genom att implementera en egenskriven rutin för kvaternionmultiplikation i programmeringsspråket C, som anropas från MATLAB via en MEX-fil.

## 6.3 Simulering 1

Denna simulering undersöker följande problemställning:

Givet en startpunkt $p_0$ och en sekvens rotationsoperatorer $R_1, R_2, ..., R_{n-1}, R_n$ som ska appliceras på $p_0$, vilken metod är snabbast och stabilast för att bestämma sekvensen av punkter $p_1, p_2, ..., p_{n-1}, p_n$? Vi har undersökt två metoder för matriser (M1-M2) och två för kvaternioner (K1-K2) att göra rotationerna med.

**M1** Det första alternativet är att använda punkten från föregående steg. Låt $M_i$ vara matrisen för rotationen i steg $i$. Följden av punkter blir då $p_1 = M_1 p_0, p_2 = M_2 p_1, ..., p_n = M_n p_{n-1}$, där $M_i$ är rotationsmatrisen för rotation $i$.

**M2** Andra alternativet är att multiplicera ihop matriserna och sedan multiplicera resultatet med startpunkten. Detta ger följden $p_1 = M_1 p_0$, ... , $p_n = (M_n M_{n-1} \cdot ... \cdot M_2 M_1) p_0$, där parentesen innebär att produkten av rotationsmatriser beräknas först och sedan multipliceras med $p_0$

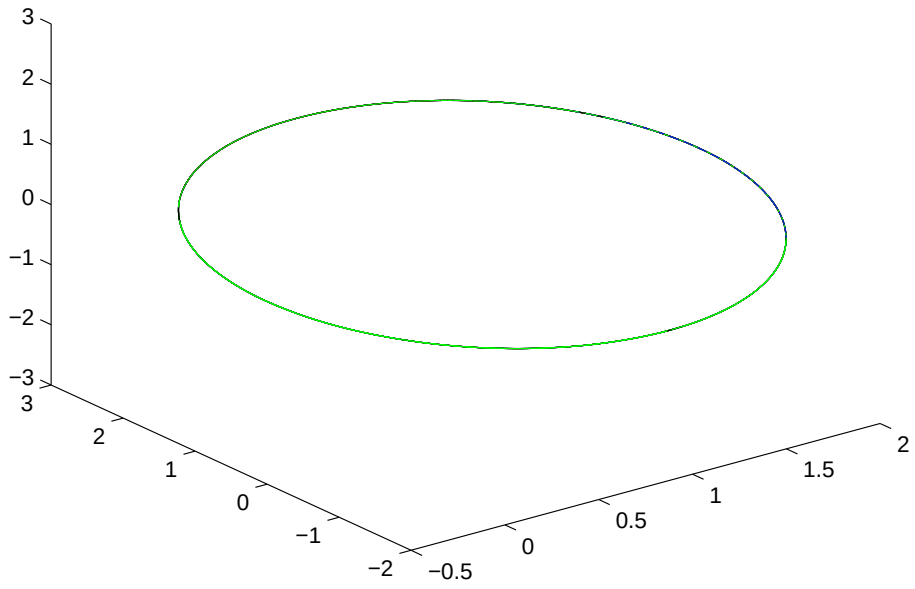
**K1** Det tredje alternativet motsvarar metod M1, fast rotationerna representeras av enhetskvaternioner istället för matriser. Kvaternionen $p_i$ svarar mot kvaternionframställningen av punkten i steg $i$, och $q_i$ är enhetskvaternionen för rotation $i$. Som bekant gäller då sambandet $p_i = q_i p_{i-1} \bar{q}_i$, vilket är det som används i den här metoden.

**K2** Det tredje alternativet motsvarar metod M2. Rekursivt får vi formeln $p_i = (q_i(q_{i-1}...(q_2(q_1 p_0 \overline{q_1})\overline{q_2})...)\overline{q_{i-1}}\overline{q_i}) = ... = (q_i q_{i-1}...q_2 q_2) p_0 \overline{(q_i q_{i-1}...q_2 q_2)}$. Detta innebär att vi först multiplicerar ihop kvaternionerna, och applicerar sedan vridningen på startpunkten.

Notera att det i exakt matematik inte är någon skillnad mellan dessa metoder. Skillnaderna ligger på ett numeriskt plan. Metoder som är ekvivalenta matematiskt kan skilja sig markant i den numeriska världen, och det är detta fenomen som ska undersökas.

I simuleringen kommer punkten $p_0 = (1, 0, 2)$ roteras kring vektorn $\mathbf{v} = (2, 1, 0)$, vilket ger en cirkel som illustreras av figur 5. För detta ändamål har vi skrivit en MATLAB-kod med möjlighet att ändra olika parametrar. Den aktuella punkten kommer roteras 300 varv, med en upplösning av 100 steg per varv. Detta ger att vridningen i varje steg är $\frac{2\pi}{100}$ radianer runt cirkelns mittpunkt.

Vi kommer att betrakta olika aspekter för att kunna dra någon slutsats om vilken metod som är bäst i detta specifika fall. Det enklaste att i varje iteration betrakta är avståndet




Figur 5: Simulering 1 - cirkeln som fås då punkten $p_0$ roteras

mellan den punkt vi har fått fram och mittpunkten. Detta avstånd jämförs sedan med radien för cirkeln, som vi kan bestämma ett exakt uttryck för.

Man inser snabbt att enbart detta inte är tillfredsställande eftersom en rotation som bevarar längden av radien inte nödvändigtvis behöver vara korrekt. Radien bevaras även om punkten roteras så att den inte längre ligger i samma plan som cirkeln. Vi kan alltså som resultat få en punkt som har exakt rätt avstånd till cirkelns mittpunkt, men som ligger lite snett ovanför den riktiga punkten. Att betrakta enbart radien räcker alltså inte för att kunna avgöra vilken metod som är stabilast. Vi inför därför två andra mått att undersöka. Dessa illustreras nedan.

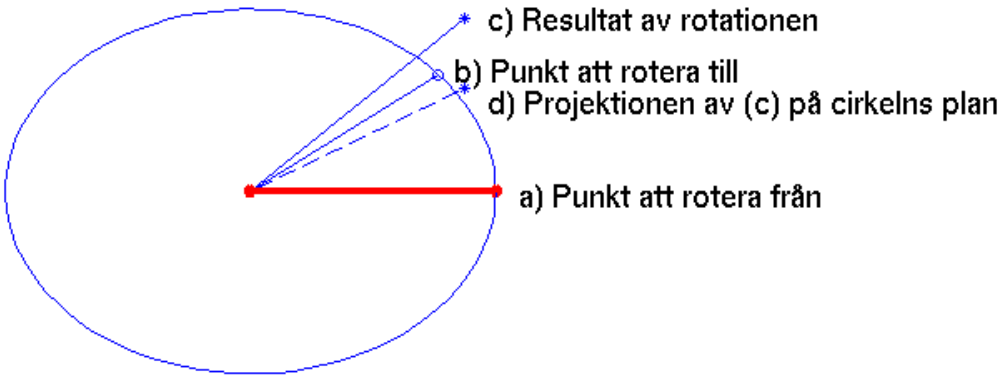
Betrakta figur 6: Målet är att i ett steg rotera punkten $a$ till punkten $b$. På grund av numeriska fel kommer vi inte hamna exakt i punkten $b$, utan i punkten $c$, som ej nödvändigtvis ligger i samma plan som cirkeln. Det vi gör då är att beräkna projektionen av punkten $c$ på cirkelns plan och mäter den vinkel som uppstår. Kalla denna vinkel för $\alpha$.

Projektionen ger oss punkten $d$. Vi beräknar nu vinkeln mellan vektorn som går genom mittpunkten och $b$ med vektorn genom mittpunkten och $d$. Kalla denna vinkel för $\beta$.

Om vi räknar exakt kommer båda dessa vinklar att vara exakt 0, men i och med att vi inte räknar exakt kommer dessa vinklar inte att bli exakt noll. Genom att studera hur stora avvikelser vi får kan vi dra slutsatser om vilken metod som är stabilast för att beräkna rotationer.

För beräkning av projektionen används Householdertransformationer. Householdertransformationer är ett alternativ till den ”vanliga” projektionsformeln som ofta används i läroböcker i linjär algebra. Givet en vektor $\mathbf{v}$ och en punkt $\mathbf{p}$ ($\mathbf{v}, \mathbf{p}$ angivna som kolonnvektorer) ges projektionen $\hat{\mathbf{p}}$ av $\mathbf{p}$ på planet vinkelrätt mot $\mathbf{v_1}$ av $\hat{\mathbf{p}} = (I - \mathbf{v}\mathbf{v}^T)\mathbf{p}$.

Vinkeln $\theta$ mellan två vektorer $\mathbf{v}_1, \mathbf{v}_2$ beräknas via $cos\theta = \frac{\mathbf{v}_1 \bullet \mathbf{v}_2}{||\mathbf{v}_1||_2||\mathbf{v}_2||_2}$, där $||\cdot||_2$ betecknar den vanliga tvånormen.




Figur 6: Simulering - illustration av a,b,c,d

## 6.4 Simulering 2

Eftersom MATLAB är ett långsamt språk att köra rutiner i förväntar vi oss inte att kvaternionerna ska gå snabbare än matrisen om det rutin som används för att multiplicera två kvaternioner är skriven i MATLAB. För att snabba upp koden ytterligare skrivs därför den rutinen istället i språket C, som är betydligt snabbare än MATLAB vad gäller egna script. Denna rutin kan sedan med hjälp av MEX-filer anropas från MATLAB.

Rutinen använder resultatet från lemma 2.2: Om $q_1 = s_1 + a_1 i + b_1 j + c_1 k = (s_1, \mathbf{v}_1)$ och $q_2 = s_2 + a_2 i + b_2 j + c_2 k = (s_2, \mathbf{v}_2)$ är två kvaternioner gäller för produkten av dessa:

$$\begin{aligned} q_1 q_2 = (s_1 s_2 - a_1 a_2 - b_1 b_2 - c_1 c_2) + (s_1 a_2 + a_1 s_2 + b_1 c_2 - c_1 b_2) i + \\ + (s_1 b_2 + b_1 s_2 + c_1 a_2 - a_1 c_2) j + (s_1 c_2 + a_1 b_2 - b_1 a_2 + c_1 s_2) k \end{aligned}$$

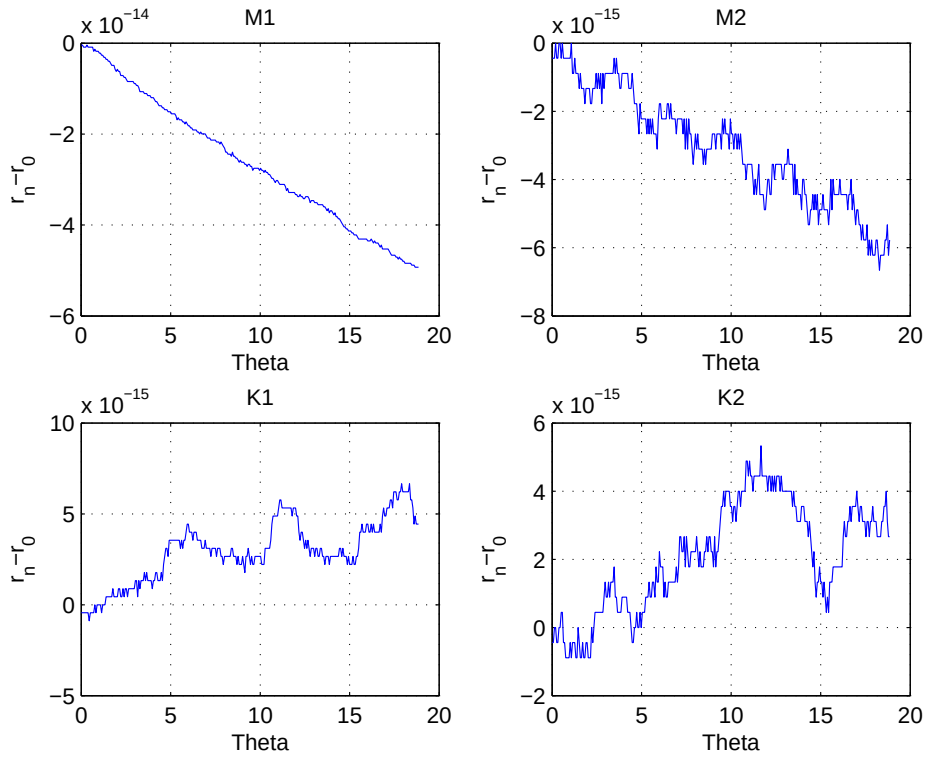
I simulering två betraktas enbart det som tidigare kallats M1 och K1, då dessa på förhand kan motiveras bör vara snabbare, detta då matris-vektormultiplikation kräver färre beräkningar än matris-matrismultiplikation.

Bortsett från det är simulering 1 och 2 identiska. Källkod finns bifogad i appendix.

# 7 Resultat

Följande kapitel sammanställer resultaten för simulering 1 och 2 som presenterats i föregående avsnitt.

## 7.1 Simulering 1

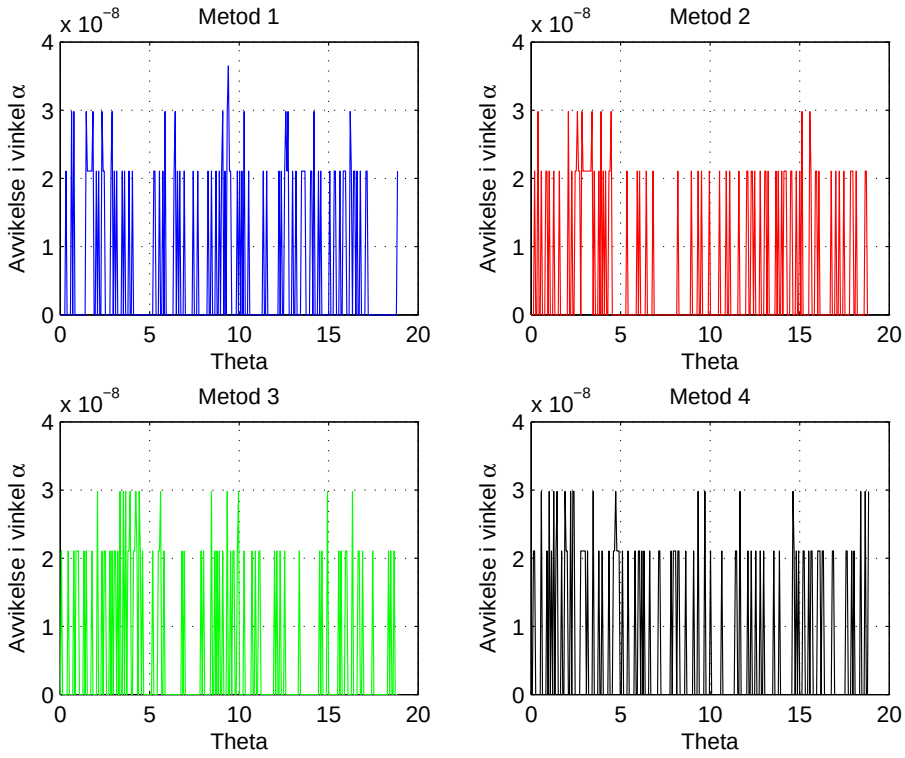



Figur 7: Simulering 1 - $r_n - r_0$

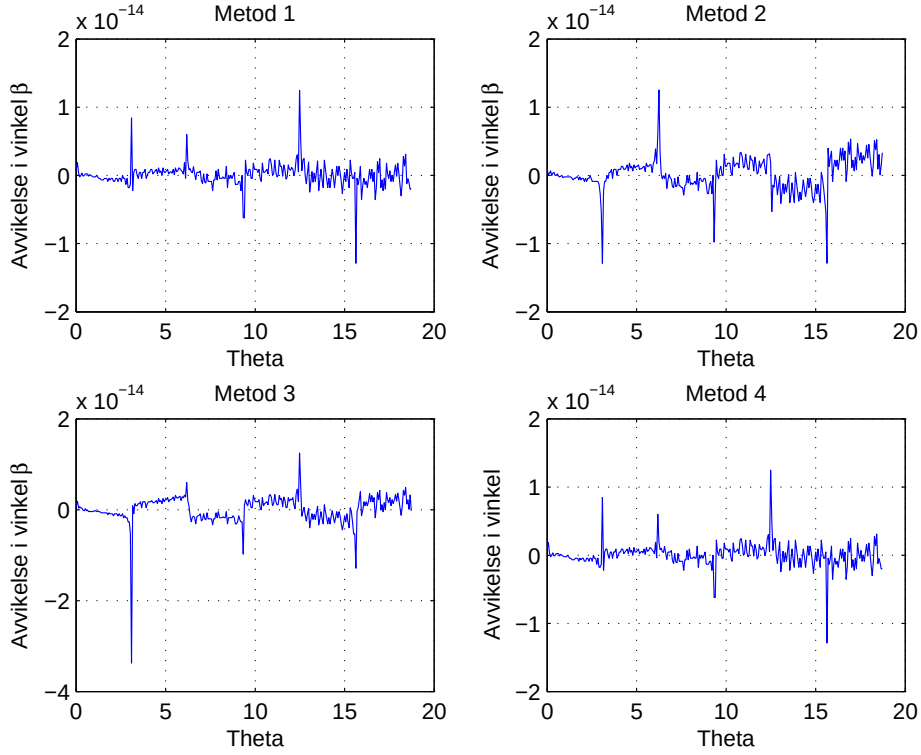
Figur 7 visar hur mycket avståndet mellan den beräknade punkten i steg n och mittpunkten $r_n$ skiljer sig från den verkliga radien $r_0$. Det syns tydligt att i metod M1 och M2 tenderar detta avstånd att krympa, medan de däremot i metod K1 och K2 verkar öka. Detta bekräftar att det är numeriskt känsligt att multiplicera många matriser/kvaternioner med norm $\approx 1$ med varandra. Resultatet av det syns i figur 7, skillnaden $r_n - r_0$ är inte konstant 0. Anledningen är att då en vektor multipliceras med något som inte har norm ett utförs ju inte enbart en rotation, utan även en skalning.

Figur 8 och 9 visar avvikelsen i de vinklar som kallats $\alpha$ och $\beta$. För vinkel $\alpha$ kan kostateras att avvikelsen är av storleksordningen $10^{-8}$ i samtliga fall. Att felet inte är av storleksordning $\epsilon$ kan bero på numerisk instabilitet i rutinen för arccosinus snarare än i rutinerna för rotationerna. Inte heller i beaktandet av vinkel $\beta$ kan någon av metodern sägas vara bättre eller sämre, ty skillnaden mellan de uppmätta $\beta$ och motsvarande exakta $\beta$ är av storleksordning $10^{-14}$.

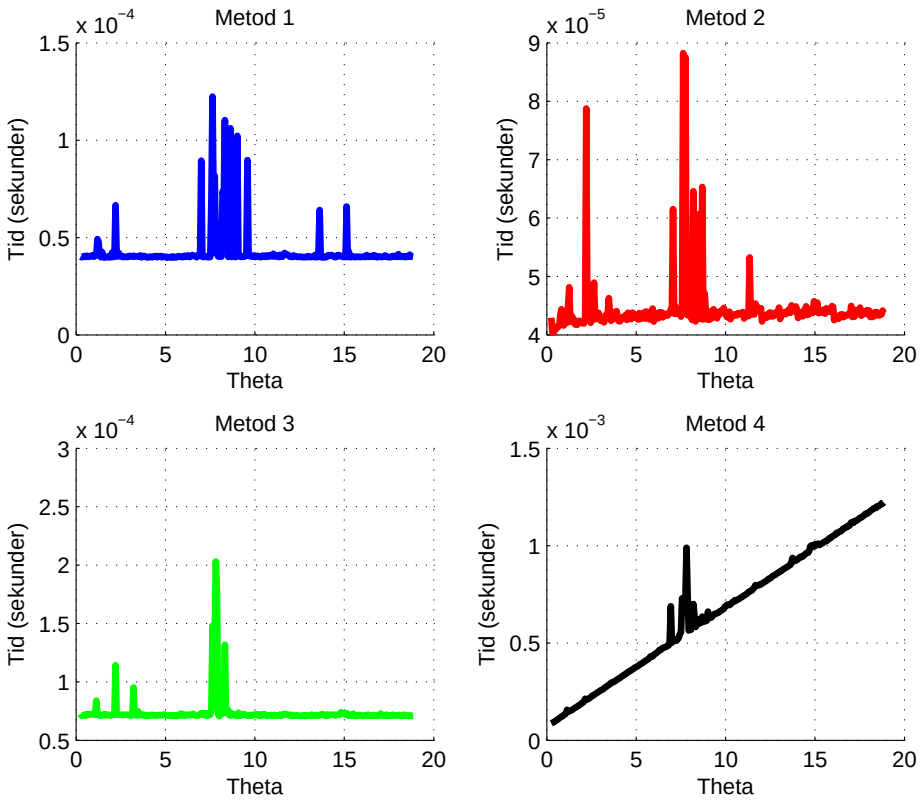
Figur 10 visar det kanske intressantaste i någon mening - tidsåtgången. Vid en första anblick ses att en av metoderna utmärker sig direkt som långsammast - metod K2. Mellan de övriga tre skiljer sig inte tidsåtgången avsevärt. Matriserna är aningen snabbare än kvaternionerna, vilket väckte ideén till simulering 2.




Figur 8: Simulering 1 - Avvikelse i vinkel $\alpha$.




Figur 9: Simulering 1 - Avvikelse i vinkel $\beta$.

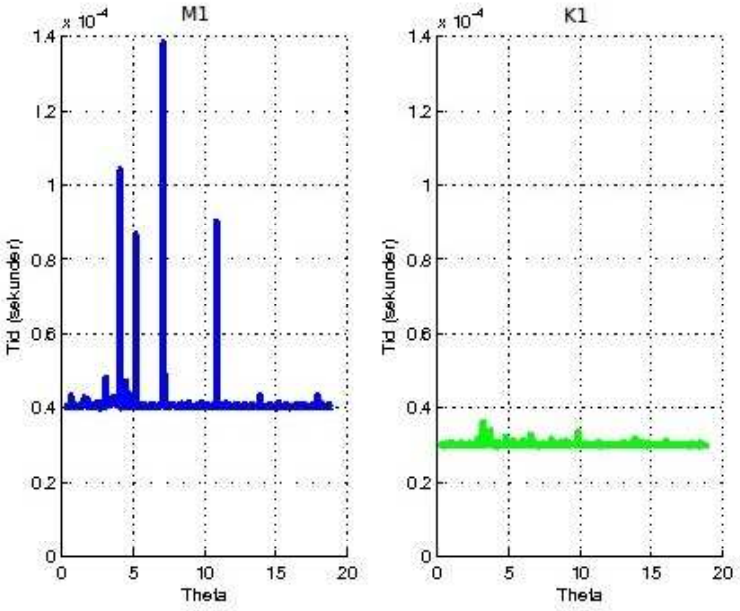



Figur 10: Simulering 1 - Tidsåtgång för de olika metoderna

## 7.2 Simulering 2

När kvaterniomultiplikationen istället utförs av en C-rutin som anropas från MATLAB via en MEX-fil blir resultaten lite annorlunda. Den enda som skiljer sig från simulering 1 är tidsåtgången för K1, varför endast den bilden tas med.

Figuren visar att med kvaternionmultiplikationen implementerad i en C-rutin, som sedan importeras via en MEX-fil, istället för ett MATLAB-script går det snabbare att utföra rotationerna med kvaternioner än med matriser. Detta beror på att MATLAB är ett långsamt språk för att köra egna skript i.




Figur 11: Simulering 2 - Tidsåtgång

# 8 Diskussion

I tidigare avsnitt har vi beskrivit två parametriseringar av rotationer - Eulervinklar och axel-vinkel. Parametriseringarna har representerats på två sätt: rotationsmatriser och kvaternioner. Varje parametrisering och representationer har sina för- respektive nackdelar, och i det här avsnittet diskuteras dessa.

## 8.1 Axel-vinkelparametrisering vs. Eulervinklar

Mängden $SO(3)$ har den egenskapen att den saknar rand. Att dela upp mängden i Eulervinklar innebär att införa en rand till gruppen, vilket på ett intuitivt plan förklarar den diskontinuitet som visas i avsnitt 4. I axel-vinkelparametriseringen finns ingen motsvarande rand vilket kan ses som en fördel för denna parametrisering.

Axel-vinkelrepresentationen har fördelen att den, till skillnad från Eulervinklarna, står i ett-till-ett förhållande till rotationerna, varför man lätt kan växla från den ena till den andra. Eulervinklarna, å sin sida, har fördelen av att i någon mening vara intuitivare. Det går att, utan inblandning av någon parametrisering, givet en rotation och ett utgångsläge hitta de Eulervinklar som beskriver rotationen. Att under samma förutsättningar hitta en axel och en vinkel som ger samma rotation är betydligt svårare.

## 8.2 Matriser vs. kvaternioner

En viktig skillnad mellan rotationsmatriser och kvaternioner är att rotationsmatriser är ett specialfall av matriser. Detta får konsekvenser då vi lämnar den exakta matematiken och tvingas göra avrundningar. Allmänna matriser kan, förutom rotationer, representera till exempel projektioner eller skalningar. Eftersom en rotationsmatris $A$ är lösningen till den kvadratiska ekvationen $AA^T = I$ innehåller $A$ ofta irrationella element; då vi avrundar en rotationsmatris till en matris med ett begränsat antal siffror i varje element riskerar vi att lämna rotationerna och få en annan linjär avbildning. Denna kommer att ligga nära en rotation, men efter många avrundningar kan resultatet bli något annat. Kvaternioner å sin sida blir med transformationen vi använt alltid rotationer, åtminstone i exakt aritmetik. Efter avrundningar resulterar även kvaternioner i avbildningar som endast ungefärligt är rotationer, men då inversen kan beräknas för varje kvaternion ackumuleras inte felen på samma sätt som för matriser. Kvaternioner har också den fördelen att de har ett naturligt samband med axel-vinkelrepresentationen av rotationer. En given enhetskvaternion $q$ roterar med vinkeln $arccos(Re\ q)$ kring axeln $Im\ q$. Något sådant enkelt samband med en parametrisering existerar inte för matriser.

En fördel med matriser är att om objekt även ska translanteras så kan man med hjälp av homogena koordinater och 4x4-matriser skapa en matris som roterar och translanterar på samma gång. Detta görs inte lika lätt med kvaternioner.

Det är tydligt att det är större skillnad mellan Euler- och axel-vinkelparametriseringarna än mellan matris- och kvaternionrepresentationerna i exakt matematik.

## 8.3 Simulering

Simulering 1 visade att det rent tidsmässigt inte var någon större skillnad mellan att utföra axel-vinkel rotationer m.h.a. matriser och kvaternioner. Det gick snabbare att skapa kvaternionen som krävdes för en rotation, men i slutändan tog allt som allt ungefär lika lång tid. Detta har helt och hållet att göra med att den implementering för att beräkna produkten av två kvaternioner var för dålig. Är det hastighet man är ute efter är MATLAB-script inte det man bör använda, vilket även visade sig i simuleringen.

Rutinen som MATLAB använder för att multiplicera en matris med en vektor är i grund och botten en rutin skriven i programmeringsspråket Fortran, som i sin tur bygger på hårt optimerade LAPACK- (Linear Algebra PACKage) och BLAS-rutiner (Basic Linear Algebra Subroutines). Dessa är bland de snabbaste rutinerna som finns för att utföra matris-vektor multiplikation.

Rutinen som användes för beräkning av produkten för två kvaternioner i simulering 1 är däremot en egenskriven rutin, som förvisso bygger på komplexa matriser, men som inte alls är lika optimerad som matris-vektor rutinen. Att tiden kommer ner till samma storleksordning får ses som en lyckad simulering med tanke på de förutsättningarna.

I simulering 2 bekräftades det att rutinen för kvaternionmultiplikation var en förklaring till att det tog lång tid för kvaternionerna i simulering 1. Vi fick då, med en rutin skriven i C, rotationer med kvaternioner att gå snabbare än rotationer med matriser. Detta visar att kvaternioner faktiskt går att få snabbare än matriser, och motiverar varför de är intressanta att studera ur ett numeriskt och inte bara ur ett rent matematiskt perspektiv. Det visar även att det spelar roll vilket programmeringsspråk som används.

Värt att notera är att den matrismultiplikation som använts är den inbyggda i MATLAB. Visserligen bygger den på LAPACK och BLAS-rutiner, men MATLAB skickar inte datan direkt till rutinen, utan undersöker först vad för sorts matris det är, och till vilken rutin datan ska skickas. MATLABs matrismultiplikation skickar datan till olika BLAS-rutiner beroende på hur matrisen ser ut (gles, bandmatris, Hessenberg, etc.). Om matrisen är liten, vilket den är i våra simuleringar, så kan det mycket väl vara så att det tar längre tid att skicka datan till en optimerad rutin än vad det tar att räkna ut matrisen med MATLABs enklaste rutin som bygger på yttreprodukter. Med andra ord vet vi inte om det skulle gå snabbare eller långsammare att göra en MEX-fil för matrismultiplikationer också.

Viktigt att påpeka är även att i en riktig tillämpning där målet är så snabba beräkningar som möjligt används inte MATLAB. MATLAB har den fördelen att det är ett relativt lätt och intuitivt programmeringsspråk att lära sig, men om man är ute efter optimerad kod så bör exempelvis Fortran eller C användas. Det hade varit intressant att göra motsvarande simulering helt i något av de språken, men tiden var för knapp.

Vad gäller felen som vi får i simuleringarna anser vi att samtliga fel är tillräckligt små för att ses som försumbara. Därför väljer vi att lägga störst vikt vid vilken metod som går snabbast. Det skulle gå att studera hur metoderna beter sig när man exempelvis inte känner alla punkter exakt, utan har störningar i vissa punkter/rotationer, men det har ingen tid lagts på.

# 9 Slutsatser

De slutsatser som vi kan dra av det här arbetet är följande:

- En rent imaginär kvaternion exponentierad blir en enhetskvaternion (generalisering av Eulers identitet).
- För varje kvaternion finns en motsvarande rotation och för varje rotation finns exakt två motsvarande kvaternioner.
- Alla uppdelningar av rotationer i delrotationer kring förutbestämda axlar (exempelvis Eulervinklar) lider av förlust av frihetsgrader.
- När vi väljer att representera rotationer med kvaternioner är enligt vår mening axel-vinkel den behändigaste formen att arbeta med.
- Matriser och kvaternioner på formen axel-vinkel ger matematiskt sett exakt samma resultat när man arbetar med dem i praktiken. Skillnaderna kommer in först på ett numeriskt plan, där det går åt mindre utrymme för att lagra en kvaternion och det går (i våra simuleringar) snabbare att räkna med kvaternioner.
- Om det som eftersträvas är hög beräkningshastighet bör ett annat språk än MATLAB användas, exempelvis Fortran eller C.

# 10 Vidare studier

Det här arbetet har enbart behandlat en liten del av allt som finns att göra under titeln kvaternioner och rotationer. För den som tycker ämnet är intressant och kan tänka sig att

göra ett arbete som behandlar avbildningar med hjälp av rotationer ges här några förslag på vidare frågeställningar som kan vara intressanta att undersöka:

- Rotationer kring andra punkter än origo. Hur förändras teorin om de rotationer som utförs inte längre ska ske genom origo, om den ändras över huvud taget.
- Andra avbildningar än rotationer. Hur kan man göra exempelvis translantioner med hjälp av kvaternioner?
- Hur interpolerar vi rotationer? Givet två lägen (position+orientering) i det 3-dimensionella rummet, hur hittar vi den bana att gå längs för att interpolera dessa lägen?
- Hur beter sig matriser och kvaternioner numeriskt sett om man programmerar i ett annat språk än MATLAB och gör rutiner med ändamålet att optimera operationerna så mycket som möjligt?
- Inom området Liegrupper och Cliffordalgebror finns mycket mer att fördjupa sig i än det som vi har gått igenom i det här arbetet.

# Referenser


[1] Mars 2012, Idén att lägga SO(3) i en boll kommer från http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation.

[2] John Tenniel (1865), *Alice with the march hare, hatter and dormouse at the mad tea party*, 2012, http://upload.wikimedia.org/wikipedia/commons/6/69/Alice_par_John_Tenniel_25.png.

[3] Simon Altmann, *Rotations, quaternions and double groups.*

[4] Martin John Baker, *Maths - angle between vectors*, April 2012, http://www.euclideanspace.com/maths/algebra/vectors/angleBetween/index.htm.

[5] Julius Brzezinski, *Linjär och multilinjär algebra*, Matematiska vetenskaper, Chalners Tekniska Högskola, Göteborgs Universitet, 2005.

[6] Michael J. Crowe, *A history of vector analysis.*

[7] Leonhardus Eulerus, *Formulae generales pro translatione quacunque corporum rigidorum*, Novi Commentarii academiae scientiarum Petropolitanae **20** (1775), 189–207.

[8] D. M. Henderson, *(nasa-tm-74839) shuttle program. euler angles, quaternions, and transformation matrices working relationships.*

[9] Michael Henle, *Modern geometries: The analytic approach*, Prentice Hall, Inc., 1997.

[10] William Karush, *Matematisk uppslagsbok*, Wahlström och Widstrand Stockholm., 1970.

[11] Philippe B. Laval, *Rotation about an arbitrary axis*, April 2012, http://science.kennesaw.edu/ plaval/math4490/rotgen.pdf.

[12] Ian R. Porteous, *Clifford algebras and the classical groups*, Cambridge University Press, 1995.

[13] Ken Shoemake, *Animating rotation with quaternion curves*, Siggraph **19** (1985), no. 3, 245–251.


# Figurer



# Tabeller

# A Cliffordalgebror

Kvaternioner utgör en av oändligt många specialfall av en så kallad Cliffordalgebra. Vi ska därför nu beskriva vad en Cliffordalgebra är, vilka egenskaper den har och dess koppling till kvaternionerna. Först behövs dock några begrepp introduceras.

**Definition A.1.** *Låt $K$ vara en kropp. En algebra $A$ över $K$ är en ring $A$ så att*

- $(A, +)$ *är ett vektorrum över* $K$
- *För* $k \in K$ *och* $a, b \in A$ *gäller att* $k(ab) = (ka)b = a(kb)$.

*En algebra över $K$ brukar kallas för en K-algebra*

**Exempel:** Med $K = \mathbb{R}$ och $A = \{\mathbb{C}\}$ skapar vi en algebra över alla reella tal. Dessa utgör ett av oändligt många exempel på en K-algebra.

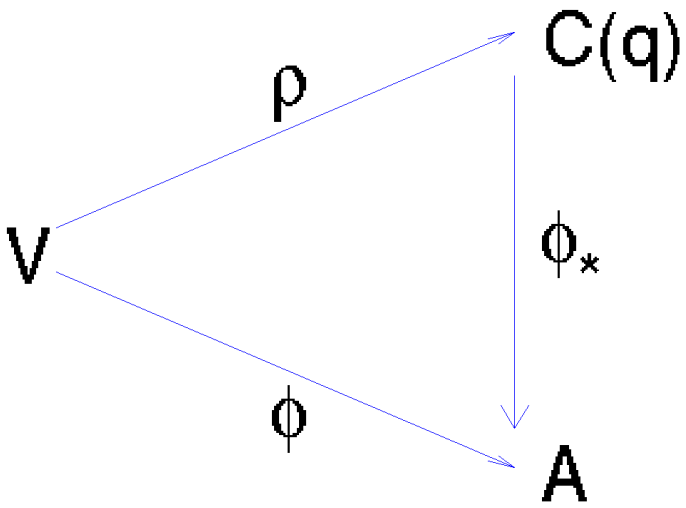
**Definition A.2.** *Med ett* **kvadratiskt rum** *avses ett rum $X$ tillsammans med en kvadratisk form $q(x)$, $x \in X$. Detta brukar betecknas $(X, q)$.[12]*

Som exempel på ett kvadratiskt rum har vi $(\mathbb{R}^2, q)$, där $q(x) = x_1^2 + x_2^2$.

**Definition A.3.** *Låt $(V, q)$ vara ett kvadratiskt rum. Med en* **bilinjär** *form på $V$ menas en funktion $b(x, y)$ sådan att*

$$b(x, y) = q(x + y) - q(x) - q(y) \tag{5}$$

**Definition A.4.** *(Cliffordalgebra) Låt $(V, q)$ vara ett kvadratiskt rum över en kropp $K$. Med en Cliffordalgebra av $(V, q)$ menar man en $K$-algebra $C(q)$ sådan att det finns en injektiv linjär avbildning $\rho : V \to C(q)$ med $\rho(v)^2 = q(v) \cdot 1$, där 1 är ettan i $C(q)$, och följande villkor är uppfyllt: Om $\phi : V \to A$ är en $K$-linjär avbildning av $V$ i en $K$-algebra $A$ sådan att $\rho(v)^2 = q(c) = 1_A$ så existerar exakt en algebrahomomorfism $\phi_* : C(q) \to A$ sådan att nedanstående diagram kommuterar: [5]*




Figur 12: Diagram över avbildningarna $\rho$, $\phi$, $\phi_*$.

Varje Cliffordalgebra har en bas, som vi kan ta reda på med hjälp av nedanstående sats:

**Sats A.5.** *Låt $(V, q)$ vara ett kvadratiskt rum enligt ovan med $dim V = n$. Då existerar $C(q)$ och $dim C(q) = 2^n$. $C(q)$ har som bas över $K$:*

$$1 \ \textit{och} \ \ e_{i_1}, ..., e_{i_1} \tag{6}$$

*där* $1 \leq i_1 < ... < i_k \leq n$ *då är en ortogonalbas för* $(V, b)$. *Om* $q(e_i) = a_i$ *så är* $e_i^2 = a_i$ *och* $e_i e_j = -1$ *då* $i \neq j$.

Beviset är relativt långt och tekniskt och är direkt hämtat från "Linjär och multilinjär algebra" [5].

*Bevis.* Låt $\phi : V \to A$ uppfylla $\bar{x}^2 = q(x)$, där $\bar{x} = \phi(x)$. Då är $\bar{x}\bar{y} + \bar{y}\bar{x} = b(x, y)$ för $x, y \in V$. Detta följer av följande likheter:

$$q(x + y) = q(x) + b(x, y) + q(y) = \bar{x}^2 + b(x, y) + \bar{y}^2$$

$$q(x + y) = (x \bar{+} y)^2 = (\bar{x} + \bar{y})^2 = \bar{x}^2 + \bar{x}\bar{y} + \bar{y}\bar{x} + \bar{y}^2$$

Vi har $\bar{x}^2 + \bar{x}\bar{y} + \bar{y}\bar{x} + \bar{y}^2 = q(x + y) = \bar{x}^2 + b(x, y) + \bar{y}^2$, vilket ger $\bar{x}\bar{y} + \bar{y}\bar{x} = b(x, y)$.

Låt vidare $e_1, ..., e_n$ vara en ortogonalbas för $(V, b)$ över $K$ med$q(e_i) = a_i$. Då är $\bar{e_i}^2 = q(e_i) = a_i$ och $\bar{e_i}\bar{e_j} + \bar{e_j}\bar{e_i} = b(e_i, e_k) = 0$ då $i \neq j$. Detta innebär att varje produkt $\bar{e_{i_1}} \cdot ... \cdot \bar{e_{i_k}}$ kan skrivas om till en produkt med $1 \leq i_1 < ... < i_k < n$ följd av en koefficient ur $K$ som fås genom upprepad användning av $\bar{e_i}^2 = a_i$ och $\bar{e_j}\bar{e_i} = -\bar{e_i}\bar{e_j}$. Låt vidare $I = \{i_1, ..., i_k\} \subseteq \{1, 2, ..., n\}$ och låt $e_I = e_{i_1} \bar{\cdot} ... \bar{\cdot} e_{i_k}$. Genom upprepad användning av $\bar{e_i}^2 = a_i$ och $\bar{e_j}\bar{e_i} = -\bar{e_i}\bar{e_j}$ fås även att

$$e_I e_J = \beta(I, J) \prod_{k \in I \bigcap J} a_k e_{I \div J} \tag{7}$$

där $\beta(I, J) = \prod(i, j)$ med $i \in I$, $j \in J$ och

$$(i, j) = \begin{cases} 1 \; om \; i \leq j \\ -1 \; om \; i > j \end{cases}$$

samt $I \div J = (I \bigcap J) - (I \cup J)$. Vi antar dessutom att $\beta(\emptyset, J) = \beta(I, \emptyset) = 1$.

Observera att

$$\beta(I_1 \div I_2, J) = \beta(I_1, J)\beta(I_2, J) \quad och \quad \beta(I, J_1 \div J_2) = \beta(I, J_1)\beta(I, J_2)$$

Nu kan vi definiera $C(q)$. Först låt $C(q)$ vara ett vektorrum med $2^n$ basvektorer $e_I$ för $I \subseteq \{1, 2, ..., n\}$. Därefter förvandlar vi $C(q)$ till en vektoralgebra genom att definiera en produkt av två godtyckliga elelment ur $C(q)$ som

$$\sum x_I e_I \cdot \sum y_J e_J = \sum x_I y_J \beta(I, J) \prod_{k \in I \cap J} a_k e_{I \div J} \tag{8}$$

där $x_I, y_J \in K$. På det sättet får vi en K-algebra. Den har som etta $e_\emptyset$, ty

$$e_\emptyset e_I = \beta(\emptyset, I) e_{\emptyset \div I} = e_I = e_I e_\emptyset \tag{9}$$

Vi identifierar $e_1, ..., e_n$ med $e_{\{1\}}, ..., e_{\{n\}}$ och får på det sättet får vi $V \subset C(q)$. Vi har

$$e_i^2 = e_i e_i = \beta(\{i\}, \{i\}) a_i e_{\{i\} \div \{i\}} = a_i \tag{10}$$

och

$$e_j e_i + e_i e_j = \beta(\{j\}, \{i\}) e_{\{j\} \div \{i\}} + \beta(\{i\}, \{j\}) e_{\{i\} \div \{j\}} = 0 \tag{11}$$

då $i \neq j$. Alltså är $e_i^2 = a_i$ (vi skriver $e_\emptyset = 1$ och identifierar $a \in K$ med $a \cdot 1$ så att $K \subseteq C(q)$, och $e_j e_i = -e_i e_j$. Om $x = \sum x_i e_i \in V$ så är

$$\tag{12}$$

dvs $x^2 = q(x)$. Slutligen kontrollerar vi att algebran $C(q)$ är associativ:

$$(e_I e_J) e_K = \beta(I, J) \prod_{r \in I \cap J} a_r e_{I \div J} e_K = \beta I, J \beta(I \div J, K) = \prod_{r \in I \cap J} a_r \prod_{r \in (I \div J) \cap K} a_r e_{(I \div J) \div K} \tag{13}$$

och

$$e_I(e_J e_K) = e_I \beta(J,K) \prod_{r \in J \cap K} a_r e_{J \div K} = \beta(J,K)\beta(I, J \div K) \prod_{r \in J \cap K} a_r \prod_{r \in I \cap (J \div K} a_r e_{I \div (J \div K)} \tag{14}$$

Men $(I \div J) \div K = I \div (J \div K)$, $\beta(I,J)\beta(I \div J, K) = \beta(J,K)\beta(I, J \div K)$ och

$$\prod_{r \in I \cap J} a_r \prod_{r \in (I \div J) \cap K} a_r = \prod_{r \in J \cap J} a_r \prod_{r \in I \cap (J \div K)} a_r \tag{15}$$

ty bägge är lika med $\prod_r a_r$, där $r \in (I \cap J) \cup (I \cap K) \cup (J \cap K)$

Om nu $\phi : V \to A$ är K-linjär och $\phi(x)^2 = q(x)$ då $x \in V$ så kan man definiera $\phi_* : C(q) \to A$ så att $\phi_*(e_I) = \phi(e_{i_1})...\phi(e_{i_r})$, där $I = \{i_1, ..., i_r\}$. Dock gäller formel 7 i $A$, så $\phi_*$ är en algebrahomomorfism. Å andra sidan, om man har en algebrahomomorfism $\psi : C(q) \to A$ sådan att $\psi(x) = \phi(x)$ får $x \in V$ så är $\psi(e_i) = \phi_*(e_i) = \phi(e_i)$, vilket ger $\psi(x) = \phi_*(x)$ för varje $x \in C(q)$ därför att $C(q)$ genereras av $e_i$. Detta visar att $\psi_*$ är entydig. [5] □

Eftersom Cliffordalgebror är väldigt generella kan vi definiera egna, och vi får olika Cliffordalgebror beroende på vilken kvadratisk form vi använder som $q$.

**Definition A.6.** *Med en* **Generaliserad kvaternionalgebra** *menar vi Cliffordalgebran med* $q(X,Y) = aX^2 + bY^2, a, b \in K$. *Denna algebra brukar betecknas* $(a,b)_K$.

En generaliserad kvaternionalgebra har dimension 4 och basen $1, e_1, e_2, e_1e_2$. Från ovanstående sats har vi att $e_1^2 = a$, $e_2^2 = b$ och $e_1e_2 = -e_2e_1$. Inför nu beteckningarna $i = e_1$, $j = e_2$ och $k = e_1e_2$. Då är $i^2 = a$,$j^2 = b$, $k^2 = -ab$, $ij = -ji = k$, $jk = -kj = -bi$, $ki = -ik = -aj$.

Vi noterar att om vi låter $(a,b) = (-1,-1)$ så får vi algebran $(-1,-1)_K$, bestående av alla tal på formen $q_0 + q_1 i + q_2 j + q_3 k$ med $i^2 = j^2 = k^2 = -1$, $ij = -ji = k$. Denna algebra betecknas $\mathbb{H}$ och utgör kvaternionalgebran som innehåller Hamiltons kvaternioner.

# B Källkod

## B.1 Beräkning av $qp\bar{q}$

```
clc
v=[1 1 1];
v=v/norm(v);
theta=pi/2;
p=[1 2 3];


n=1e4;
% Metod 1 - quatprod
    quat=[ cos(theta/2) v.*sin(theta/2)];
    quat_inv=[quat(1) -quat(2:4)];

tic
for k=1:n
    p_ny_1=quatprod(quatprod(quat,[0 p]),quat_inv);
end
t=toc;
disp(['Metod 1: Elapsed time is ' num2str(t) ' seconds.'])

% Metod 2 - "formel"
quat=[ cos(theta/2) v.*sin(theta/2)];
    quat_inv=[quat(1) -quat(2:4)];

tic
for k=1:n

    p_ny_2=p+cross(2*quat(2:4),(cross(quat(2:4),p)+quat(1)*p));
end
t=toc;
disp(['Metod 2: Elapsed time is ' num2str(t) ' seconds.'])

% Metod 3 - komplexa 2x2 matriser
   quat=[ cos(theta/2) v.*sin(theta/2)];
    quat_inv=[quat(1) -quat(2:4)];

    quat_matris=[quat(1)+quat(2)*1i quat(3)+quat(4)*1i; -quat(3)+quat(4)*1i qu
    punkt_matris=[0+p(1)*1i p(2)+p(3)*1i; -p(2)+p(3)*1i 0-p(1)*1i];
    quat_matris_invers=[quat(1)-quat(2)*1i -quat(3)-quat(4)*1i; quat(3)-quat(4

tic
for k=1:n

    p_ny_3=quat_matris*punkt_matris*quat_matris_invers;
end
t=toc;
disp(['Metod 3: Elapsed time is ' num2str(t) ' seconds.'])


p_ny_1=p_ny_1(2:4)
p_ny_2
p_ny_3=[imag(p_ny_3(1,1)) real(p_ny_3(1,2)) imag(p_ny_3(1,2))]
```

## B.2 Beräkning av rotationsmatris respektive kvaternion

```
vektor=rand(1,3);
v=vektor/norm(vektor);

theta=rand(1)*pi;

iter=500000;
clc

%Skapa matrisen
tic
for k=1:iter
    M=[];
    c=cos(theta);
    s=sin(theta);
    M=[(1-v(1)^2)*c+v(1)^2   -v(3)*s-v(1)*v(2)*c+v(1)*v(2)  v(2)*s-v(1)*v(3)*c
    v(3)*s-v(1)*v(2)*c+v(1)*v(2)   (1-v(2)^2)*c+v(2)^2     -v(1)*s-v(2)*v(3)*c
    -v(2)*s-v(1)*v(3)*c+v(1)*v(3)  v(1)*s-v(2)*v(3)*c+v(2)*v(3)
(1-v(3)^2)*c+v(3)^2];
end
disp(['Skapande av ' num2str(iter) ' rotationsmatriser: ' num2str(toc) ' sekun

%Skapar kvaternionen
tic
for k=1:iter
    q=[];
    q=[cos(theta*0.5) sin(theta*0.5).*v];
end
disp(['Skapande av ' num2str(iter) ' rotationskvaternioner: ' num2str(toc) ' s
```

## B.3 Simulering 1

```
%   SIMULATION 1 IN THESIS FOR BATCHELOR DEGREE IN MATHEMATICS
%
%   THESIS SUBJECT: QUATERNIONS AND ROTATIONS
%
%   DEVELOPED BY: N. ANDERSSON, D. OGNISSANTI, E. WEDIN
%
%   CHALMERS UNIVERSITY OF THECHNOLOGY/UNIVERSITY OF GOTHENBURG
%
%   SPRING 2012

feature accel off

%Point to rotate
punkt=[1 0 2];
punkt=punkt(:);

%Vector to rotate around
vektor=[2 1 0];
v=vektor/norm(vektor); %Need a unit vector

%Computes center and radius for the circle
mittpunkt=(punkt(:)'*v(:))*v;
mittpunkt=mittpunkt(:);
radie=norm(punkt-mittpunkt);

antal_varv=3;
steg_per_varv=100;

Antal_iterationer_per_punkt=100; %To make an average of times in each step

vinklar=linspace(0,antal_varv*2*pi,antal_varv*steg_per_varv);
vinkel_i_varje_steg=vinklar(2)-vinklar(1);
v_i_v_s=vinkel_i_varje_steg;
pett=zeros(3,antal_varv*steg_per_varv);
pett(:,1)=punkt';
ptva=pett;
ptre=pett;
pfyra=pett;
radie1=zeros(1,antal_varv*steg_per_varv)';
radie2=radie1;
radie3=radie1;
radie4=radie1;
nyquat=[1,0,0,0];

tid1=radie4;
tid2=tid1;
tid3=tid1;
tid4=tid2;

vinkel1=tid4;
vinkel2=vinkel1;
vinkel3=vinkel1;
vinkel4=vinkel1;

vinkelb1=vinkel1;
```

```
vinkelb2=vinkelb1;
vinkelb3=vinkelb1;
vinkelb4=vinkelb1;

index=1;


co=cos(v_i_v_s);
si=sin(v_i_v_s);

lv=length(vinklar);

Antal_iterationer_per_punkt_invers=1/Antal_iterationer_per_punkt;
for theta=vinklar;
    index=index+1;

    %Kommentera bort ånedanstende rad öfr snabbare kod.
    disp([num2str(100*index/(lv+1)) '% avklarat']);

    if(mod(theta+v_i_v_s,2*pi)>pi)
        theta2=2*pi-mod(theta+v_i_v_s,2*pi);
    else
        theta2=mod(theta+v_i_v_s,2*pi);
    end

    %Method 1

%     t=cputime;
    tic
    for k=1:Antal_iterationer_per_punkt
       Matris=[];
       p_temp=[];
       Matris(:,1)= [(1-v(1)^2)*cos(v_i_v_s)+v(1)^2
       -v(3)*sin(v_i_v_s)-v(1)*v(2)*cos(v_i_v_s)+v(1)*v(2)
       v(2)*sin(v_i_v_s)-v(1)*v(3)*cos(v_i_v_s)+v(1)*v(3)];

       Matris(:,2)=[ v(3)*si-v(1)*v(2)*cos(v_i_v_s)+v(1)*v(2)
                  (1-v(2)^2)*cos(v_i_v_s)+v(2)^2
                  -v(1)*sin(v_i_v_s)-v(2)*v(3)*cos(v_i_v_s)+v(2)*v(3)];

       Matris(:,3)=[-v(2)*sin(v_i_v_s)-v(1)*v(3)*cos(v_i_v_s)+v(1)*v(3)
                    v(1)*si-v(2)*v(3)*cos(v_i_v_s)+v(2)*v(3)
                    (1-v(3)^2)*cos(v_i_v_s)+v(3)^2];
        p_temp=pett(:,index-1);
        punktny=Matris*p_temp;
    end

    tid1(index-1)=(toc)*Antal_iterationer_per_punkt_invers;
    pett(:,index)=punktny(:);
     radie1(index)=norm(punktny(:)-mittpunkt(:));

    a=punktny(:)-mittpunkt(:);


    %Projection with Householder
    c=(eye(3)-v(:)*v(:)')*punktny(:);

```

```
113      vinkel1(index+1)=acos((a(:)'*c(:))/(norm(a)*norm(c)));
114      vinkelb1(index+1)=acos(a'*(punkt(:)-mittpunkt(:))/...
115                          (norm(a)*norm(punkt(:)-mittpunkt(:))))-theta2;
116
117      %Metod 2
118
119     tic
120
121      for k=1:Antal_iterationer_per_punkt
122          % äNollstller M till identiteten ö fr att det ska vara
123          % samma ööäfrutstningar i alla iterationer
124          Matris=eye(3);
125          Matris=[ ...
126          (1-v(1)^2)*co+v(1)^2    -v(3)*si-v(1)*v(2)*co+v(1)*v(2) ...
127          v(2)*si-v(1)*v(3)*co+v(1)*v(3);
128          v(3)*si-v(1)*v(2)*co+v(1)*v(2)    (1-v(2)^2)*co+v(2)^2   ...
129          -v(1)*si-v(2)*v(3)*co+v(2)*v(3);
130          -v(2)*si-v(1)*v(3)*co+v(1)*v(3)   v(1)*si-v(2)*v(3)*co+v(2)*v(3) ...
131          (1-v(3)^2)*co+v(3)^2];
132
133            Matris=Matris^(index-1);
134          punktny=Matris*punkt;
135      end
136
137      tid2(index-1)=(toc)*Antal_iterationer_per_punkt_invers;
138      ptva(:,index)=punktny(:);
139      radie2(index)=norm(punktny(:)-mittpunkt(:));
140
141      a=punktny(:)-mittpunkt(:);
142
143
144      %Projection with Householder transformation
145      c=(eye(3)-v(:)*v(:)')*punktny(:);
146
147      vinkel2(index+1)=acos((a(:)'*c(:))/(norm(a)*norm(c)));
148      vinkelb2(index+1)=acos(a'*(punkt(:)-mittpunkt(:))/...
149                          (norm(a)*norm(punkt(:)-mittpunkt(:))))-theta2;
150
151
152      %Method 3
153      p=ptre(:,index-1);
154
155  %Kvaternionen a+bi+cj+dk kan representeras av den komplexa matrisen:
156  % [(a+bi   ,  (c+di)]
157  % [(-c+di),  (a-bi)
158
159  tic
160      for k=1:Antal_iterationer_per_punkt
161           %äNollstller öfr att det ska vara lika i alla iterationer
162          q_matris=zeros(2);
163          q_matris_invers=zeros(2);
164          p_matris=zeros(2);
165          quat=zeros(1,4);
166          quat_inv=zeros(1,4);
167
168          quat=[cos(0.5*v_i_v_s),v.*sin(0.5*v_i_v_s)];
169          quat_inv=[quat(1),-quat(2:end)];
```

```
170         q_matris=[quat(1)+quat(2)*1i quat(3)+quat(4)*1i; ...
171                     -quat(3)+quat(4)*1i quat(1)-quat(2)*1i];
172 
173         q_matris_invers=[quat_inv(1)+quat_inv(2)*1i ...
174                         quat_inv(3)+quat_inv(4)*1i;
175                         -quat_inv(3)+quat_inv(4)*1i...
176                         quat_inv(1)-quat_inv(2)*1i];
177 
178         p_matris=[ p(1)*1i p(2)+p(3)*1i; -p(2)+p(3)*1i -p(1)*1i];
179 
180         punktny=q_matris*p_matris*q_matris_invers;
181     end
182 t=toc;
183     tid3(index-1)=(t)*Antal_iterationer_per_punkt_invers;
184     punktny=[imag(punktny(1,1)) -real(punktny(2,1)) imag(punktny(2,1))];
185     ptre(:,index)=punktny(:);
186     radie3(index)=norm(punktny(:)-mittpunkt(:));
187     a=(punktny(:)-mittpunkt(:));
188 
189     %Projection with Householder transformation
190     c=(eye(3)-v(:)*v(:)')*punktny(:);
191 
192     vinkel3(index+1)=acos((a(:)'*c(:))/(norm(a)*norm(c)));
193     vinkelb3(index+1)=acos(a'*(punkt(:)-mittpunkt(:))/ ...
194                     (norm(a)*norm(punkt(:)-mittpunkt(:))))-theta2;
195 
196     %Method 4
197 
198 %      t=cputime;
199   p=punkt;
200 
201     tic
202     for k=1:Antal_iterationer_per_punkt
203         p_matris=zeros(2);
204         q=zeros(1,4);
205         q=[cos(0.5*v_i_v_s),v.*sin(0.5*v_i_v_s)];
206         p_matris=[ p(1)*1i p(2)+p(3)*1i; -p(2)+p(3)*1i -p(1)*1i];
207         ny_q=eye(2);
208 
209         for j=1:index-1
210             ny_q=ny_q*q_matris;
211         end
212         ny_q_inv=[real(ny_q(1,1))-imag(ny_q(1,1))*1i ...
213                 -real(ny_q(1,2))-imag(ny_q(1,2))*1i; ...
214                 real(ny_q(1,2))-imag(ny_q(1,2))*1i ...
215                 real(ny_q(1,1))+imag(ny_q(1,1))*1i];
216         punktny=ny_q*p_matris*ny_q_inv;
217 
218     end
219     punktny=[imag(punktny(1,1)),real(punktny(1,2)),imag(punktny(1,2))];
220     tid4(index-1)=(toc)*Antal_iterationer_per_punkt_invers;
221     pfyra(:,index)=punktny(:);
222     radie4(index)=norm(punktny(:)-mittpunkt(:));
223 
224     a=punktny(:)-mittpunkt(:);
225 
226     %Projection with Householder transformation
```

```
227      c=(eye(3)-v(:)*v(:)')*punktny(:);
228
229      vinkel4(index+1)=acos((a(:)'*c(:))/(norm(a)*norm(c)));
230      vinkelb4(index+1)=acos(a'*(punkt(:)-mittpunkt(:))/...
231                      (norm(a)*norm(punkt(:)-mittpunkt(:))));%-theta2;
232  end
233
234
235
236  %% Plotting section
237  figure(1)
238  clf
239  % min_axis=-5e-14;
240  % max_axis=5e-14;
241  subplot(2,2,1)
242  plot(vinklar,radie1(2:end)-radie)
243  grid on
244
245  clc
246
247  title('M1')
248  xlabel('Theta')
249  ylabel('r_n-r_0')
250  % axis([0 max(vinklar) min_axis max_axis])
251
252  subplot(2,2,2)
253  plot(vinklar,radie2(2:end)-radie)
254  grid on
255  title('M2')
256  xlabel('Theta')
257  ylabel('r_n-r_0')
258  % axis([0 max(vinklar) min_axis max_axis])
259
260  subplot(2,2,3)
261  grid on
262  plot(vinklar,radie3(2:end)-radie)
263  grid on
264  title('K1')
265  xlabel('Theta')
266  ylabel('r_n-r_0')
267
268  % axis([0 max(vinklar) min_axis max_axis])
269
270  subplot(2,2,4)
271  plot(vinklar,radie4(2:end)-radie)
272  grid on
273  title('K2')
274  xlabel('Theta')
275  ylabel('r_n-r_0')
276
277  % axis([0 max(vinklar) min_axis max_axis])
278  % saveas(gcf,'fig1.png')
279
280  %%
281  figure(2)
282  clf
283  subplot(2,2,1)
```

```
284 plot(vinklar,real(vinkel1(3:end)),'b')
285 grid on
286 title('M1')
287 xlabel('Theta')
288 ylabel('Avvikelse i vinkel \alpha');
289
290 subplot(2,2,2)
291 plot(vinklar,real(vinkel2(3:end)),'r')
292 grid on
293 title('M2')
294 xlabel('Theta')
295 ylabel('Avvikelse i vinkel \alpha');
296
297 subplot(2,2,3)
298 plot(vinklar,real(vinkel3(3:end)),'g')
299 grid on
300 title('K1')
301 xlabel('Theta')
302 ylabel('Avvikelse i vinkel \alpha');
303
304 subplot(2,2,4)
305 plot(vinklar,real(vinkel4(3:end)),'k')
306 grid on
307 title('K2')
308
309 xlabel('Theta')
310 ylabel('Avvikelse i vinkel \alpha');
311
312 % saveas(gcf,'fig2.png')
313
314 %%
315 figure(3)
316 clf
317 plot3(pett(1,:),pett(2,:),pett(3,:),'k')
318 hold on
319 % plot3(ptva(1,:),ptva(2,:),ptva(3,:),'r')
320 % plot3(ptre(1,:),ptre(2,:),ptre(3,:),'g')
321 plot3(pfyra(1,2:end),pfyra(2,2:end),pfyra(3,2:end),'k')
322 % legend('Matris åp punkt','Matris^n åp startpunkt','Kvaternion öfr hela vridni
323 %
324 plot3([0 vektor(1)],[0 vektor(2)],[0 vektor(3)],'k','LineWidth',3)
325 plot3([0 -vektor(1)],[0 -vektor(2)],[0 -vektor(3)],'k','LineWidth',3)
326 plot3(mittpunkt(1),mittpunkt(2),mittpunkt(3),'b*','MarkerSize',10)
327  grid on
328  axis equal
329  axis vis3d
330
331 % saveas(gcf,'../fig3.png')
332
333
334 %%
335 figure(4)
336
337 clf
338 axis equal
339 % subplot(2,2,1)
340 plot(vinklar(5:end),tid1(5:end),'bo','Linewidth',1)
```

```
% hold on
%  plot(vinklar(5:end),tid2(5:end),'r*-','Linewidth',1)
%  plot(vinklar(5:end),tid3(5:end),'g-','Linewidth',1)
%  plot(vinklar(5:end),tid4(5:end),'k--','Linewidth',1)
%  hold on
% semilogy(vinklar(5:end),tid1(5:end),'b','Linewidth',3)
% hold on
%  semilogy(vinklar(5:end),tid2(5:end),'r','Linewidth',3)
%  semilogy(vinklar(5:end),tid3(5:end),'g','Linewidth',3)
%  semilogy(vinklar(5:end),tid4(5:end),'k','Linewidth',3)
% grid on
% legend('Metod 1','Metod 2','Metod 3','Metod 4');
%
% xlabel('Theta')
% ylabel('Tid (sekunder)');
subplot(2,2,1)
hold on
semilogy(vinklar(5:end),tid1(5:end),'b','Linewidth',3)
grid on
xlabel('Theta')
ylabel('Tid (sekunder)');
title('M1')
subplot(2,2,2)
hold on
title('M2')
semilogy(vinklar(5:end),tid2(5:end),'r','Linewidth',3)
grid on
xlabel('Theta')
ylabel('Tid (sekunder)');
subplot(2,2,3)
hold on
title('K1')
semilogy(vinklar(5:end),tid3(5:end),'g','Linewidth',3)
grid on
xlabel('Theta')
ylabel('Tid (sekunder)');
subplot(2,2,4)
hold on
title('K2')
semilogy(vinklar(5:end),tid4(5:end),'k','Linewidth',3)
grid on
xlabel('Theta')
ylabel('Tid (sekunder)');

% saveas(gcf,'../fig4.png')

%%
figure(5)
clf
subplot(2,2,1)
plot(vinklar(1:end-2),vinkelb1(3:end-2),'b')
grid on
title('M1')
xlabel('Theta')
ylabel('Avvikelse i vinkel \beta');
subplot(2,2,2)

```

```
398 plot(vinklar(1:end-2),vinkelb2(3:end-2),'b')
399 title('M2')
400 xlabel('Theta')
401 ylabel('Avvikelse i vinkel \beta');
402 grid on
403 subplot(2,2,3)
404
405 plot(vinklar(1:end-2),vinkelb3(3:end-2),'b')
406
407 title('K1')
408 xlabel('Theta')
409 ylabel('Avvikelse i vinkel \beta');
410 grid on
411 subplot(2,2,4)
412 plot(vinklar(1:end-2),vinkelb1(3:end-2),'b')
413 title('K2')
414
415 xlabel('Theta')
416 ylabel('Avvikelse i vinkel \beta');
417 grid on
418
419 xlabel('Theta')
420 ylabel('Avvikelse i vinkel');
421
422 % saveas(gcf,'../fig5.png')
```

## B.4 Simulering 2

### B.4.1 MATLAB

```
%   SIMULATION 2 IN THESIS FOR BATCHELOR DEGREE IN MATHEMATICS
%
%   THESIS SUBJECT: QUATERNIONS AND ROTATIONS
%
%   DEVELOPED BY: N. ANDERSSON
%
%   CHALMERS UNIVERSITY OF THECHNOLOGY/UNIVERSITY OF GOTHENBURG
%
%   SPRING 2012

feature accel off

%Point to rotate
punkt=[1 0 2];
punkt=punkt(:);

%Vector to rotate around
vektor=[2 1 0];
v=vektor/norm(vektor); %Need a unit vector

%Computes center and radius for the circle
mittpunkt=(punkt(:)'*v(:))*v;
mittpunkt=mittpunkt(:);
radie=norm(punkt-mittpunkt);

antal_varv=3;
steg_per_varv=100;

Antal_iterationer_per_punkt=100; %To make an average of times in each step

vinklar=linspace(0,antal_varv*2*pi,antal_varv*steg_per_varv);
vinkel_i_varje_steg=vinklar(2)-vinklar(1);
v_i_v_s=vinkel_i_varje_steg;
pett=zeros(3,antal_varv*steg_per_varv);
pett(:,1)=punkt';
ptva=pett;
ptre=pett;
pfyra=pett;
radie1=zeros(1,antal_varv*steg_per_varv)';
radie2=radie1;
radie3=radie1;
radie4=radie1;
nyquat=[1,0,0,0];

tid1=radie4;
tid2=tid1;
tid3=tid1;
tid4=tid2;

vinkel1=tid4;
vinkel2=vinkel1;
vinkel3=vinkel1;
vinkel4=vinkel1;
```

```
54
55  vinkelb1=vinkel1;
56  vinkelb2=vinkelb1;
57  vinkelb3=vinkelb1;
58  vinkelb4=vinkelb1;
59
60  index=1;
61
62
63  co=cos(v_i_v_s);
64  si=sin(v_i_v_s);
65
66  lv=length(vinklar);
67
68  Antal_iterationer_per_punkt_invers=1/Antal_iterationer_per_punkt;
69  for theta=vinklar;
70      index=index+1;
71      disp([num2str(100*index/(lv+1)) '% avklarat']);
72
73      if(mod(theta+v_i_v_s,2*pi)>pi)
74          theta2=2*pi-mod(theta+v_i_v_s,2*pi);
75      else
76          theta2=mod(theta+v_i_v_s,2*pi);
77      end
78
79      % Matriser
80
81      tic
82      for k=1:Antal_iterationer_per_punkt
83         Matris=[];
84         p_temp=[];
85         Matris(:,1)=[ (1-v(1)^2)*cos(v_i_v_s)+v(1)^2
86                       -v(3)*sin(v_i_v_s)-v(1)*v(2)*cos(v_i_v_s)+v(1)*v(2)
87                       v(2)*sin(v_i_v_s)-v(1)*v(3)*cos(v_i_v_s)+v(1)*v(3)];
88
89         Matris(:,2)=[ v(3)*si-v(1)*v(2)*cos(v_i_v_s)+v(1)*v(2)
90                     (1-v(2)^2)*cos(v_i_v_s)+v(2)^2
91                     -v(1)*sin(v_i_v_s)-v(2)*v(3)*cos(v_i_v_s)+v(2)*v(3)];
92
93         Matris(:,3)=[-v(2)*sin(v_i_v_s)-v(1)*v(3)*cos(v_i_v_s)+v(1)*v(3)
94                      v(1)*si-v(2)*v(3)*cos(v_i_v_s)+v(2)*v(3)
95                      (1-v(3)^2)*cos(v_i_v_s)+v(3)^2];
96          p_temp=pett(:,index-1);
97          punktny=Matris*p_temp;
98      end
99
100     tid1(index-1)=(toc)*Antal_iterationer_per_punkt_invers;
101     pett(:,index)=punktny(:);
102     radie1(index)=norm(punktny(:)-mittpunkt(:));
103
104     a=punktny(:)-mittpunkt(:);
105
106     %Projection with Householder
107     c=(eye(3)-v(:)*v(:)')*punktny(:);
108
109     vinkel1(index+1)=acos((a(:)'*c(:))/(norm(a)*norm(c)));
110     vinkelb1(index+1)=acos(a'*(punkt(:)-mittpunkt(:))/...
```

```
111                         (norm(a)*norm(punkt(:)-mittpunkt(:))))-theta2;
112
113     % Kvaternioner
114     p=ptre(:,index-1);
115
116 tic
117     for k=1:Antal_iterationer_per_punkt
118
119         quat=[cos(0.5*v_i_v_s),v.*sin(0.5*v_i_v_s)];
120         quat_inv=[quat(1),-quat(2:end)];
121        p_kvat=[0 p(:)'];
122         punktny=q_mult(q_mult(quat,p_kvat),quat_inv);
123     end
124     t=toc;
125     tid3(index-1)=(t)*Antal_iterationer_per_punkt_invers;
126     punktny=punktny(2:4);
127     ptre(:,index)=punktny(:);
128     radie3(index)=norm(punktny(:)-mittpunkt(:));
129     a=(punktny(:)-mittpunkt(:));
130
131     %Projection with Householder transformation
132     c=(eye(3)-v(:)*v(:)')*punktny(:);
133
134     vinkel3(index+1)=acos((a(:)'*c(:))/(norm(a)*norm(c)));
135     vinkelb3(index+1)=acos(a'*(punkt(:)-mittpunkt(:))/...
136                     (norm(a)*norm(punkt(:)-mittpunkt(:))))-theta2;
137
138 end
139
140
141
142 %% Plotting section
143 figure(1)
144 clf
145 subplot(1,2,1)
146 plot(vinklar,radie1(2:end)-radie)
147 grid on
148 title('Metod 1')
149 xlabel('Theta')
150 ylabel('Avvikelse i radie');
151
152 subplot(1,2,2)
153 grid on
154 plot(vinklar,radie3(radie3>0)-radie)
155 grid on
156 title('Metod 3')
157 xlabel('Theta')
158 ylabel('Avvikelse i radie');
159
160
161 %%
162 figure(2)
163 clf
164 subplot(1,2,1)
165 plot(vinklar,real(vinkel1(3:end)),'b')
166 grid on
167 title('Metod 1')
```

```
168 xlabel('Theta')
169 ylabel('Avvikelse i vinkel \alpha');
170
171 subplot(1,2,2)
172 plot(vinklar,real(vinkel3(3:end)),'g')
173 grid on
174 title('Metod 3')
175 xlabel('Theta')
176 ylabel('Avvikelse i vinkel \alpha');
177
178
179 %%
180 figure(4)
181
182 clf
183 axis equal
184 subplot(1,2,1)
185 hold on
186 semilogy(vinklar(5:end),tid1(5:end),'b','Linewidth',3)
187 grid on
188 xlabel('Theta')
189 ylabel('Tid (sekunder)');
190 title('Metod 1')
191 axis([0 20 0 14e-5])
192 subplot(1,2,2)
193 hold on
194 title('Metod 3')
195 semilogy(vinklar(5:end),tid3(5:end),'g','Linewidth',3)
196 grid on
197 axis([0 20 0 14e-5])
198 xlabel('Theta')
199 ylabel('Tid (sekunder)');
200
201 %%
202 figure(5)
203 clf
204 subplot(1,2,1)
205 plot(vinklar(1:end-2),vinkelb1(3:end-2),'b')
206 grid on
207 title('Metod 1')
208 xlabel('Theta')
209 ylabel('Avvikelse i vinkel \beta');
210
211 subplot(1,2,2)
212
213 plot(vinklar(1:end-2),vinkelb3(3:end-2),'b')
214
215 title('Metod 3')
216 xlabel('Theta')
217 ylabel('Avvikelse i vinkel \beta');
218 grid on
```

### B.4.2 MEX

MEX-fil:

```
#include <math.h>
#include "mex.h"

// MEX-file, used in simulation 2 in Batchelor Degree in mathematics Thesis
// Thesis title: "Quaternions and Rotations"
// Created by Niklas Andersson

void mexFunction(int nlhs, mxArray * plhs[],
                 int nrhs, const mxArray * prhs[])
{
  double *q1, *q2,*p_q_ut;
  int q1_rows,q1_cols,q2_rows,q2_cols;


  q1_rows = mxGetM(prhs[0]);
  q1_cols = mxGetN(prhs[0]);
  q2_rows = mxGetM(prhs[1]);
  q2_cols = mxGetN(prhs[1]);

  plhs[0] = mxCreateDoubleMatrix(q1_cols, q1_rows, mxREAL);

  p_q_ut = mxGetPr(plhs[0]);         // Allocate space to output

  q1 = mxGetPr(prhs[0]);          // q1
  q2 = mxGetPr(prhs[1]);          // q2

  *(p_q_ut + 0) = (*(q1+0))*(*(q2+0))-(*(q1+1))*(*(q2+1))-
              (*(q1+2))*(*(q2+2))-(*(q1+3))*(*(q2+3));

  *(p_q_ut + 1) = (*(q1+0))*(*(q2+1))+(*(q1+1))*(*(q2+0))+
              (*(q1+2))*(*(q2+3))-(*(q1+3))*(*(q2+2));

  *(p_q_ut + 2) = (*(q1+0))*(*(q2+2))-(*(q1+1))*(*(q2+3))+
                 (*(q1+2))*(*(q2+0))+(*(q1+3))*(*(q2+1));

  *(p_q_ut + 3) = (*(q1+0))*(*(q2+3))+(*(q1+1))*(*(q2+2))-
              (*(q1+2))*(*(q2+1))+(*(q1+3))*(*(q2+0));

}
```