CHALMERS

GÖTEBORGS UNIVERSITET

Rotationer, spinorer och spinn

Hur man kan visualisera spinn

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Lars Wickström
Liqin Xu
Patrik Agné
Simon Jonsson

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

Rotationer, spinorer och spinn

Hur man kan visualisera spinn

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet

Liqin Xu Patrik Agné

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk fysik vid Chalmers

Lars Wickström

Kandidatarbete i matematik inom civilingenjörsprogrammet Kemiteknik med fysik vid Chalmers

Simon Jonsson

Handledare: Andreas Rosén
Examinator: Maria Roginskaya Ulla Dinger

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

Populärvetenskaplig presentation

I detta arbete ska vi diskutera begreppet spinn, vad som inom matematik kallas $\mathbf{Z}_2$-minnet hos rotationer. Det visar sig för en del system, bland annat inom kvantmekaniken, att när det har genomgått en full 360° rotation har det ännu inte återställts till sitt grundläge, utan är nu i motsatt konfiguration, och det är först när systemet fått rotera ytterligare 360°, till sammanlagt 720°, som systemet återställts. Spinn är inte bara ett fenomen som dyker upp inom matematiken, utan är ett naturligt fenomen och har en plats i världen vi befinner oss i.

”Bälttricket” kan hjälpa oss få insikt i detta fenomen. Det visar sig att om man fäster ena änden av ett bälte i ett bordsben och håller den andra änden i sin hand och roterar änden man håller i ett varv kommer man märka att detta vridna bälte inte går att få ”ovridet” genom att flytta runt bältet; det enda sätt det kan återgå till att bli ovridet är att rotera bältet ett helt varv till.

Ett annat sätt att betrakta fenomenet är att utgå från att rotation alltid är kring en axel, och försöka göra en avbildning av rotationer som ett typ av rum där vardera punkt utgör en viss rotation. Låt vardera axel vara en linje där samtliga linjer har en punkt de alla går igenom och låt hur långt ut på varje linje man är i förhållande till denna punkt utgöra den vinkel man roterat, detta kommer utgöra ett klot med ”radie” 180°. Men då sammanfaller systemen, dvs 180° rotation runt en axel, eller $-180°$ runt samma axel motsvarar samma rotation.

Detta fenomen uppkommer som sagt bland annat i kvantmekaniken. För att kunna modellera dem väl behöver man ett nytt sätt att föreställa sig vad som sker. Det behövs ett nytt typ av objekt med egenskapen att de, precis som kvantmekaniska system, inte återställts efter 360° rotation utan istället 720°. De matematiska objekt som har denna egenskap kallas ”spinorer” och upptäcktes av Elie Cartan 1913, ett bra tag innan behovet inom kvantmekaniken uppkom.

Ett problem med spinn och spinorer är att det är svårt att faktiskt visualisera hur det ser ut. I detta arbete har vi därför skapat en datoranimation för att visualisera fenomenen.

Vi har bland annat utvecklat datoranimationer där en användare kan rotera runt ett vanligt koordinatsystem, och jämföra den med lägen inom ett visst klot. Dessa lägen i klotet relaterar vi till kvaternioner, vilka är en samling av skalärer och ”bivektorer”. Bivektorer är en speciell typ av vektorer men som istället för att motsvara linjer är de mer som plan.

Det finns även ett program som istället låter dig manipulera en punkt i klotet, där nu koordinatsystemet istället kommer få rotera efter hur punkten befinner sig. Man kommer se att när man dragit runt koordinataxlarna ett helt varv kommer inte punkten i klotet ha återkommit, utan två varv krävs. Ett tredje program, kanske mer exotiskt än de tidigare, låter användaren se hur kvaternionen roterar spinorer, där man kan se att det kommer krävas just två varv för att systemet skall återställas. Bland annat används kvaternionerna i mekatronik och stelkroppsmekanik just för att de beskriver roterande system så enkelt och så troget till verkligheten.

I vårt projekt beskriver vi hur man bygger upp en Cliffordalgebra, en algebra som består av olika typer av ”multivektorer”, i vilka bivektorerna är en del i av. De används inte enbart i kvantmekaniken utan man kan bygga upp dem för att göra uträkningar i speciell relativitetsteori. Även en del klassisk mekanik blir förenklad med Cliffordalgebran.

Sammanfattning

Ett viktigt begrepp inom kvantmekaniken är spinn. Vissa kvantmekaniska system har egenskapen att vid en full rotation har systemet inte återställts utan befinner sig istället i motsatt konfiguration relativt startläget. Detta är vad man menar med spinn. Spinn är dock känt för att vara svårt att visualisera. I detta arbete har vi skapat en datoranimation för att visa hur spinn uppkommer och beter sig. Vi har använt programspråket MATLAB för att göra detta. För att kunna förstå denna datoranimation måste man dock först ha grundläggande förståelse för spinn. I detta arbete har vi därför gjort en genomgång av den matematiska teorin bakom spinn. Vi börjar med att förklara begreppen yttre algebra och Cliffordalgebra. Sedan introducerar vi kvaternioner och förklarar deras koppling till spinn. Vi går därefter igenom begreppen spinorer och spinorrum som är nödvändiga för att beskriva spinn i fysiken. Vi avslutar arbetet med att förklara hur koden är uppbyggd och hur den är kopplad till spinn.

Abstract

A central concept in quantum mechanics is spin. Certain quantum systems have the property that, at a full rotation, the system has not been reset, but is in the opposite configuration relative to the starting position. It is previously known that spin is hard to visualize. In this paper we have for this reason created a computer animation to show how spin arises and how it behaves. We have used the programming language MATLAB to do this. To be able to understand this animation it is necessary to have a basic understanding of spin. A central part of this paper will therefore consist of a review of the mathematical theory behind spin. We start by explaning the mathematical concepts of exterior algebra and Clifford algebra. Then we introduce quaternions and explain their connections to spin. Furthermore we explain the mathematical concepts of spinors and spinor space to describe spin. Finally we end the paper with explaining how the code is written and how it relates to spin.

Innehåll

Förord

Bidragsrapport, Lars:
Lars har till stor del skrivit all den kod som presenteras i rapporten, och skrev därför även texten som beskrev koden. Han har även skrivit den delen av introduktionen till Spinorer som beskrev kvantdynamiken hos partiklar med spinn, och även arbetat med den populärvetenskapliga rapporten. Vidare har Lars hjälpt till med bevisföring samt vart med i diskussioner om konceptuella frågor.

Bidragsrapport, Simon:
Simon har bidragit till teoridelen om yttre algebra och Cliifordalgebra. Han har även till stora delar själv skrivit teoridelen om kvaternioner och deras koppling till rotationer. Simon har även skrivit stora delar av spinoravsnitten, om dimensionerna, och även analysen utav de två representationerna av $\mathbb{H}$ fram till beviset för kompletta matrisalgebror.

Bidragsrapport, Patrik:
Patrik har skrivit stora delar av sektionen om yttre algebra och Cliffordalgebra och bidragit till ritning av figurerna. Han har skrivit sammanfattning/abstract och delar av den populärvetenskapliga presentationen. Han har också skrivit delarna om basbyten, homomorfa matrisrepresentationer och tagit fram rotationsmatrisen för godtycklig axel och vinkel i VR. Han har även bidragit till att planera arbetet och organiseringen av arbetet.

Bidragsrapport, Liqin:
Liqin har skrivit delar av teoridelen om yttre algebra och Cliffordalgebra och ritat de flesta figurerna. Hon har även bidragit till delarna om dimensionerna, komplexa matriser och kopplingen mellan våra representationer av Cliffordalgebran. Hon har även skött dagboken.

En loggbok har förts över de enskilda medverkandes prestationer.

1 Vad vi kommer att göra

Syftet med detta arbete är att skapa ett interaktivt datorprogram där man kan se relationer mellan rotationer av vektorer och spinorer, men även hur rotationer i sig påverkar vektorerna respektive spinorerna. Vidare är syftet också att kunna visualisera fenomenet spinn och att matematiskt kunna presentera varför rotationer har det interna minne vi kallar spinn.

Detta projekt kommer att vara tvådelat. Första delen kommer handla om rotationer i 3 dimensioner. Vi kommer här undersöka de topologiska egenskaperna hos mängden av rotationer för $\mathbb{R}^3$ och även påvisa det så kallade $\mathbf{Z_2}$-minnet som finns hos rotationsgruppen $SO(3)$. Detta minne är vad man menar med spinn. Vi kommer även skapa en datoranimation för att visualisera spinn.

Andra delen av projektet kommer att behandla spinorer. Spinorer är objekt som samexisterar med ett tillhörande vektorrum. Spinorer tillämpas inom kvantmekaniken och är därför viktiga att lära sig mer om. Vi kommer även att göra en datoranimation för att visa hur vektorrummet roterar tillsammans med spinorrummet.

1.1 Vad handlar egentligen spinn om?

Spinn som fenomen existerar för vektorrum med dimension högre än 2. Det roterar gradvis till två rotationer med två topologiskt urskiljbara homotopklasser, en till 2π och en till 4π. Dessa två olika klasser ger spinortransformationer av motsatt tecken. Som vi nämnde tidigare är bälttricket ett berömt exempel för att illustrera den övergripande spinnteorin. Ena änden av ett bälte är fastsatt och den andra änden roterar fritt. Bältet vrids när den fria änden roterar ett varv, för att återgå till att ej längre vara vridet måste den fria änden rotera två varv runt samma axel med moturs orientering.

1.2 Struktur och förkunskap

Vi kommer att lära oss förstå spinn genom geometrisk algebra. Först går vi igenom yttre algebra och Cliffordalgebra. Genom Cliffordalgebran kommer vi sedan att definiera kvaternioner. Kvaternioner är objekt som används för att rotera i 3 dimensioner. Sedan kommer vi att hitta en homomorfi mellan de kvaternioner som ger upphov till rotationer och rotationsgruppen $SO(3)$.

Vi kommer sedan att undersöka spinorer genom matrisrepresentationer av Cliffordalgebran i $\mathbb{R}^3$. Vi kommer även att undersöka rotationer av spinorer och även visa att de är objekt som är oberoende av representationen från Cliffordalgebran.

För att få en lättläst och (förhoppningsvis) njutbar läsning av detta arbete behöver man grundläggande kunskaper i linjär algebra och abstrakt algebra. Att ha grundläggande förståelse för kvantmekanik är nyttigt men inte nödvändigt.

1.3 Metod och material

Som vi sett är syftet med detta arbete dels att genomföra en teoretisk genomgång av begreppet spinn och spinorer, dels att göra en datoranimation där vi visar hur spinn uppkommer. För att kunna ge en teoretisk genomgång av begreppet spinn måste man ha en god förståelse för spinn. Vi har därför läst igenom och diskuterat litteraturen kring spinn och spinorer, och skapat en datoranimation för att kunna visualisera spinn. Vidare har vi fått stor hjälp från handledare Andreas Rosén för att förstå teorin. För att skapa datoranimationerna har vi använt programspråket MATLAB.

2 Yttre algebra och Cliffordalgebra

Vi inleder detta arbete med att diskutera yttre algebra. För att kunna göra detta börja vi med att definiera ett vektorrum, V, med någon kropp, K. I detta arbete kommer vi att utgå från att alla vektorrum är euklidiska. Yttre algebran $(\wedge V, +, \wedge, 1)$ till V är det 2^n dimensionella vektorrummet $\wedge V := \wedge^0 V \oplus \wedge^1 V \oplus \wedge^2 V ... \oplus \wedge^n V$. Ett element i detta rum skrivs då: $v = v_0 + v_1 + v_2 + ... + v_n \in \wedge V$ och vi säger att v är en multivektor av grad n, där $v_i \in \wedge^i V$. Speciellt är $\wedge^0 V$ rummet av skalärer, $\wedge^1 V$ rummet av vektorer, $\wedge^2 V$ rummet av bivektorer och $\wedge^3 V$ rummet av trivektorer.

Nu definierar vi yttreprodukten.

Definition Yttreprodukt (referens: Andreas Rosén (handledare), skriftlig kommunikation, 14/5 2019) *Låt $\{e_1, ..., e_n\}$ utgöra en bas för vektorrummet V och låt $v_j \in V$ vara vektorer sådana att $v_j = \sum a_{i,j} e_i$. Vi beräknar då yttreprodukten av dessa vektorer som*

$$v_1 \wedge v_2 \wedge ... \wedge v_n = \begin{vmatrix} e_1 & a_{1,1} & ... & a_{1,n} \\ e_2 & a_{2,1} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ e_n & a_{n,1} & ... & a_{n,n} \end{vmatrix} := \sum_{i=1}^{n} \begin{vmatrix} a_{1,1} & \dots & a_{i,n} \\ \vdots & \ddots & \vdots \\ a_{i,1} & \dots & a_{n,n} \end{vmatrix} e_1 \wedge ... \wedge e_n. \tag{1}$$

Vi har att följande regler för yttreprodukten:

$$1. v \wedge u = -u \wedge v \ (antikommutativitet); \tag{2}$$

$$2. (v \wedge u) \wedge w = v \wedge (u \wedge w) \ (associativitet). \tag{3}$$

2.1 Cliffordprodukt

Följande sats lägger grunden till det som vi nedan kallar Cliffordprodukten.

Sats: Lagranges identitet[1] *Låt V vara ett skalärproduktsrum. Vektorerna v_1 och v_2 uppfyller då att*

$$|\langle v_1, v_2 \rangle|^2 + |v_1 \wedge v_2|^2 = |v_1|^2 |v_2|^2. \tag{4}$$

Där $\langle a, b \rangle$ är skalärprodukten mellan två vektorer a och b, och $|v_i|$ anger längden på vektorn v_i.

Intuitionen bakom denna sats hittas i Cauchy-Schwarz och Hadamards olikheter.

Vi kommer ihåg att om V är ett skalärproduktsrum och vektorerna $u, v \in V$ vet vi från Cauchy-Schwarz olikhet att

$$|\langle v_1, v_2 \rangle| \leq |v_1||v_2|. \tag{5}$$

Hadamards olikhet säger istället att

$$|v_1 \wedge v_2| \leq |v_1||v_2|. \tag{6}$$

Vi får likhet i Hadamards olikhet om och endast om $\langle v_1, v_2 \rangle = 0$, om vektorerna är ortogonala.

[1] För bevis av denna sats, se (1)

Vi får likhet i Cauchy-Schwarz olikhet om och endast om $\langle v_1, v_2\rangle = |v_1||v_2|$, om vektorerna är parallella. Det finns alltså ett inverst förhållande mellan skalärprodukten och yttreprodukten.

I euklidiska rum definierar vi skalärprodukten som det unika talet $0 \leq \theta \leq \pi$ sådant att

$$\langle v_1, v_2\rangle = |v_1||v_2|\cos(\theta), \tag{7}$$

ekv. (4) kan då skrivas

$$|v_1|^2|v_2|^2\cos^2(\theta) + |v_1 \wedge v_2|^2 = |v_1|^2|v_2|^2. \tag{8}$$

Det följer då att

$$|v_1 \wedge v_2|^2 = |v_1|^2|v_2|^2\sin^2(\theta),\ 0 \leq \theta \leq \pi. \tag{9}$$

Dessa produkter kan vi sätta ihop genom operationen

$$v_1 \vartriangle v_2 = \langle v_1, v_2\rangle + v_1 \wedge v_2 \in \wedge^0 V \oplus \wedge^2 V. \tag{10}$$

Detta är vad vi menar med Cliffordprodukten av vektorerna v_1 och v_2. I fortsättningen kommer vi, på de ställen där det inte finns risk att missförstånd uppstår, skriva $v_1 \vartriangle v_2 =: v_1v_2$.

Definition Cliffordprodukt (1) *Cliffordprodukten är den unika bilinjära produkt på yttre algebran, $\wedge V$. För en parvis ortogonal mängd $\{a_i\}_1^n$ vektorer i V sammanfaller cliffordprodukten med yttre produkten. Vidare uppfyller även cliffordprodukten nedanstående punkter*

$$\begin{aligned} &\bullet\ a^2 = |a|^2, \forall a \in V \\ &\bullet\ a_ia_j = -a_ja_i,\ \text{för } i \neq j \\ &\bullet\ (a_1 \vartriangle a_2) \vartriangle a_3 = a_1 \vartriangle (a_2 \vartriangle a_3) \end{aligned} \tag{11}$$

Nu följer definitionen av Cliffordalgebra.

Definition Cliffordalgebra *Med ΔV menar vi Cliffordalgebran $(\wedge V, +, \vartriangle)$ definierad av Cliffordprodukten på rummet av multivektorer i V.*

Något som kan verka onödigt nu, men som kommer bli relevant senare i arbetet när vi kommer till kvaternioner, är uppdelningen i udda och jämna multivektorer. Vi definierar de jämna och udda multivektorerna som

$$\wedge^{ev} V := \wedge^0 V \oplus \wedge^2 V \oplus \wedge^4 V \oplus ...,\ \wedge^{od} V := \wedge^1 V \oplus \wedge^3 V \oplus \wedge^5 V \oplus ..., \tag{12}$$

så $\wedge V = \wedge^{ev} V \oplus \wedge^{od} V$.

Låt $\Delta^{ev}\mathrm{V}{:=}\wedge^{ev}\mathrm{V}$, $\Delta^{od}\mathrm{V}{:=}\wedge^{od}\mathrm{V}$ och

$$\Delta V = \Delta^{od} V \oplus \Delta^{ev} V. \tag{13}$$

Vi ser särskilt att om $u, v \in \Delta^{ev} V$ så följer det att $u \vartriangle v \in \Delta^{ev} V$, och alltså är sluten. Medan $\Delta^{od} V$ inte är sluten under Cliffordprodukt.

2.2 Geometrisk tolkning av multivektorer i Cliffordalgebran för $\mathbb{R}^3$

Vi ska nu närmare studera strukturen hos Cliffordalgebran för $\mathbb{R}^3$.[2] Låt$\{e_1, e_2, e_3\}$ utgöra en ON-bas för $V = \mathbb{R}^3$. Vi kan se basvektorerna som de traditionella $\{x, y, z\}$ respektive. Vi kallar dessa vektorer för grad 1 objekt och i Cliffordalgebra uttrycker vi dem $\Delta^1\mathbb{R}^3$.

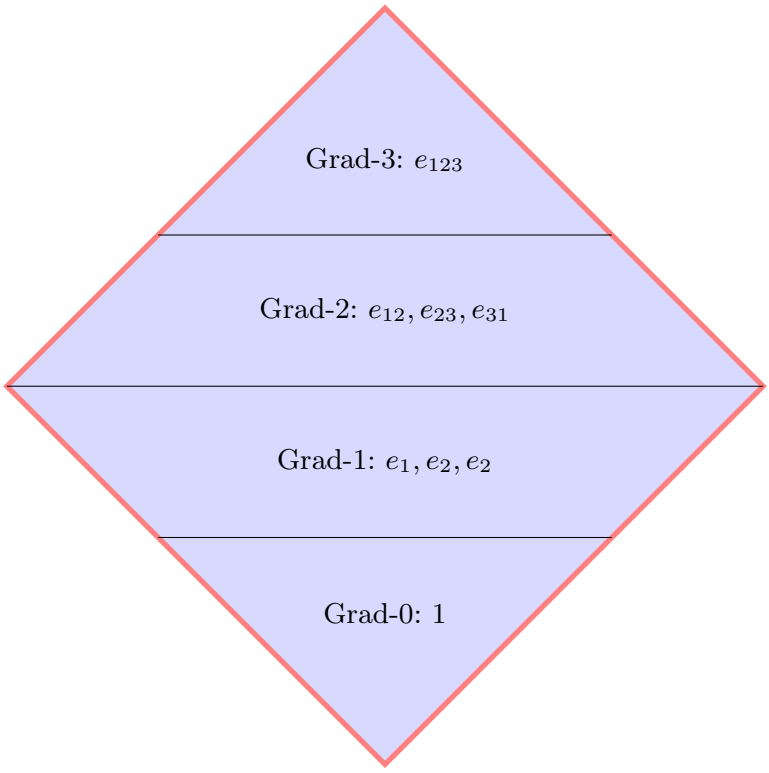
En skalär representerar en punkt och kallas för grad 0 objekt. I Cliffordalgebra blir de $\Delta^0\mathbb{R}^3$.

Från basvektorerna ska vi nu bilda de så kallade bivektorerna $\{e_1e_2, e_2e_3, e_3e_1\}$ genom Clifford-produkten. Bivektorerna ligger i respektive plan i ett 3D rum. e_1e_2 i xy-planet, e_2e_3 i yz-planet och e_3e_1 i zx-planet. Vi ritar dem som parallellogram med moturs orientering (se 4, 5 och 6 i Appendix). Vi kallar dessa grad 2 objekt och beskriver dem som $\Delta^2\mathbb{R}^3$ på Cliffordalgebraiskt vis.

Nu återstår bara att skapa det sista ”grundelementet” $e_1e_2e_3$. Vi kallar detta objekt trivektorn. I Cliffordalgebra uttrycker vi den som $e_{123} \in \Delta^3\mathbb{R}^3$.

Denna bildar vi genom att ta en av bivektorerna, säg e_1e_2, och ”förlänger” den i e_3s riktning. På samma sätt tar vi bivektorn e_2e_3 och förlänger i e_1 riktning, samt tar bivektorn e_3e_1 och förlänger den i e_2 riktning. Vi bildar på så sätt en så kallad parallellepiped med positiv orientering. (se 7 i Appendix).

Vi sammanfattar ovan 4 typer av objekt (en skalär med grad-0, tre vektorer med grad-1, tre bivektorer med grad-2 och en trivektor med grad-3) som 8 grundelement i ett vektorrum $\{1, e_1, e_2, e_3, e_1e_2, e_2e_3, e_3e_1, e_1e_2e_3\}$. Detta illustrerar vi i figur 1.



Figur 1: Cliffordalgebrans struktur i $\mathbb{R}^3$

[2]Det här beskrivningen bygger på (4)

3 Rotationer i $\mathbb{R}^3$

I detta avsnitt begränsar vi oss till att $V = \mathbb{R}^3$. Vi kommer här titta på rotationer av vektorer i V med hjälp av både rotationsmatriser men även med hjälp av kvaternionerna, som kan definieras från den jämna Cliffordalgebran. Vi hittar en homomorfi mellan kvaternionerna och rotationsmatriser, sedan går vi vidare och visar $\mathbf{Z}_2$-minnet hos rotationer. Gruppen av rotationsmatriser skrivs som $SO(3) := \{T : V \to V; T^\top T = I, \det T = 1\}$. Där $^\top$ är transponatet till matrisen.

3.1 Kvaternioner

Vi skall nu använda oss utav Cliffordalgebran för att definiera kvaternionerna. Kvaternionerna kan ses som en utvidgning av det komplexa talplanet $\mathbb{C}$. I det komplexa talplanet kan man enkelt rotera komplexa tal, $z \in \mathbb{C}$, geonom multiplikation av en fas, $e^{i\theta}$, sådan att

$$z \to z' = e^{i\theta} z = z e^{i\theta} = e^{i\frac{\theta}{2}} z e^{i\frac{\theta}{2}}.$$

Vi vill nu, på liknande sätt, använda kvaternioner för att rotera vektorer i $\mathbb{R}^3$. Kvaternionerna, som vi nedan hänvisar till som $\mathbb{H}$, är fyrdimensionella additiva objekt med följande egenskaper för sin specifika produkt

$$i^2 = j^2 = k^2 = ijk = -1. \tag{14}$$

Vi kan konstruera samma egenskaper med hjälp av den jämna Cliffordalgebran, $\Delta^{ev}V$, genom att låta

$$\begin{aligned} -e_2 e_3 &= i, \\ e_1 e_3 &= j, \\ -e_1 e_2 &= k, \end{aligned}$$

så vi får, med Cliffordprodukt, samma egenskaper som kvaternionerna. Vi representarar $\mathbb{H}$ som $\mathbb{H} = (\Delta^{ev}V, +, \triangle)$. Vi kan även se att $\mathbb{H}$ är en associativ divisionsalgebra, alltså att varje icke-noll kvaternion är inverterbar. För $q = a + bi + cj + dk$, där $\bar{q} = a - bi - cj - dk$ betecknar konjugatet i $\mathbb{H}$ har vi att $\bar{q}q = a^2 + b^2 + c^2 + d^2 = |q|^2$. qs invers är då $q^{-1} = \bar{q}/|q|^2$.

Vi vill hitta ett sätt att rotera vektorer i $\mathbb{R}^3$ i ett godtyckligt plan $[j]$ med hjälp av $\mathbb{H}$.
Betrakta först en godtycklig rotation $T \in SO(3)$ och en vinkel φ. Från B.4 följer att det alltid finns en egenvektor till T med egenvärde 1. Välj en ON-bas $\{e_i\}_{i=1}^3$ sådan att $T(e_i) = e_i$. Vi får då att matrisen för T i basen $\{e_i\}$ är

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & cos\varphi \end{bmatrix}. \tag{15}$$

För en rotationsmatris gäller

$$T = (T_1\ T_2\ T_3) = (Te_1\ Te_2\ Te_3). \tag{16}$$

Så varje kolumn i T svarar mot hur respektive basvektor har roterat.
Vi hittar nu en kvaternion q så att vi får samma rotation för respektive basvektorer,

$$\begin{aligned} qe_1q^{-1} &= e_1, \\ qe_2q^{-1} &= \cos\varphi e_2 + \sin\varphi e_3, \\ qe_3q^{-1} &= -\sin\varphi e_2 + \cos\varphi. \end{aligned}$$

Låt $q = \exp(-\varphi j/2) = cos(\varphi/2) - j\sin(\varphi/2)$. Där $j \in \Delta^2 V$ och $|j| = 1$. För e_1 får vi då

$$\begin{aligned} e_1 &= qe_1q^{-1} \\ &= \big[cos(\varphi/2) - j\sin(\varphi/2)\big] e_1 \big[cos(\varphi/2) + j\sin(\varphi/2)\big] \\ &= \cos^2\varphi e_1 - \cos\frac{\varphi}{2}\sin\frac{\varphi}{2}(je_1 - e_1 j) - \sin^2\frac{\varphi}{2} j e_1 j = e_1. \end{aligned}$$

Detta gäller vid fallet att $je_1 = e_1 j$, samt ekvivalent $-je_1 j = e_1$, vilket enligt definitionen för Cliffordprodukten (11) ger att $j = \pm e_{23}$. Vi undersöker nu om detta även gäller för e_2 och e_3. Om detta gäller har vi hittat en kvaternion för den givna matrisen.

$$\begin{aligned}
&qe_2q^{-1} = \cos\varphi e_2 + \sin\varphi e_3 \\
&= \big[cos\frac{\varphi}{2} - j\sin\frac{\varphi}{2}\big]e_2\big[cos\frac{\varphi}{2} + j\sin\frac{\varphi}{2}\big] \\
&= \cos^2\frac{\varphi}{2}e_2 + \cos\frac{\varphi}{2}\sin\frac{\varphi}{2}(e_2j - je_2) - \sin^2\frac{\varphi}{2}je_2j \\
&= \big[\cos^2\frac{\varphi}{2} - \sin^2\frac{\varphi}{2}\big]e_2 + \cos\frac{\varphi}{2}\sin\frac{\varphi}{2}(2e_2j) \\
&= \cos\varphi e_2 + \sin\varphi e_2 j.
\end{aligned}$$

Här har vi använt de trigonometriska identitetetrna

$$\cos^2\varphi - \sin^2\varphi = \cos 2\varphi, \quad 2\cos\varphi\sin\varphi = \sin 2\varphi.$$

Vidare får vi för e_3,

$$\begin{aligned}
&-\sin\varphi e_2 + \cos\varphi e_3 = qe_3q^{-1} \\
&= \big[\cos\frac{\varphi}{2} - j\sin\frac{\varphi}{2}\big]e_3\big[\cos\frac{\varphi}{2} + j\sin\frac{\varphi}{2}\big] \\
&= \cos^2\frac{\varphi}{2}e_3 + \cos\frac{\varphi}{2}\sin\frac{\varphi}{2}(e_3j - je_3) - \sin^2\frac{\varphi}{2}je_3j \\
&= \big[\cos^2\frac{\varphi}{2} - \sin^2\frac{\varphi}{2}\big]e_3 + \cos\frac{\varphi}{2}\sin\frac{\varphi}{2}(2e_3j) \\
&= \cos\varphi e_3 + \sin\varphi e_3 j.
\end{aligned}$$

Vi ser här nu att $e_3 j = -e_2$ samt att $e_2 j = e_3$, vilket ger att $j = e_23$. Vi har alltså nu hittat en representation för matriser i $SO(3)$ som kvaternioner. För att vi skall få en homomorfi krävs det på grund av konvention att $q = \exp(-\varphi j/2)$ (1). Om vi valt det omvända, $q = \exp(\varphi j/2)$ hade vi fått $j = e_{32}$. Orienteringen på denna bivektor hade då varit motsatt riktningen av rotationen, därför väljer vi $q = \exp(-\varphi j/2)$.

Sats (Rotationer i $\mathbb{R}^3$)**.** *Låt V vara ett tredimensionellt högerorienterat euklidiskt rum med orientering J. En rotation av $v \in V$ med en vinkel ϕ moturs i planet $[j]$ beskrivs genom*

$$v \to qvq^{-1}.$$

Där $q = exp(\frac{b}{2}) \in spin(V) := \{q \in \Delta^{ev}V; |q|^2 = 1\}$ och $b = -\phi j,\ \ j \in \Delta^2 V; |j|^2 = 1$.

Vi kan observera här att den grupp av kvaternioner som ger upphov till rotationer, $q \in spin(V)$ är isomorf med enhetssfären i fyra dimensioner, S^3. Detta ses tydligast genom att för $q = a + bi + cj + dk \in spin(V)$ gäller att $|q|^2 = \bar{q}q = a^2 + b^2 + c^2 + d^2 = 1$ Uttrycken $spin(V)$ och S^3 kommer nedan användas synonymt. Vi kallar ett $q \in spin(V)$ för en rotor.

Vi undersöker nu strukturen hos dessa rotationer. Antag att vi har två olika kvaternioner, q_1 och q_2, som båda ger samma rotation.

$$q_1 v q_1^{-1} = q_2 v q_2^{-1}. \tag{17}$$

Genom att multiplicera från vänster med q_2^{-1} och från höger med q_1 ges

$$q_2^{-1}q_1 v = v q_2^{-1}q_1. \tag{18}$$

Alltså kommuterar v med $q_2^{-1}q_1$.

Vi vet att skalärer är kommutativa och att addition är en kommutativ operation. Det vi behöver undersöka nu är om rena kvaterioner, bivektorer, kommuterar med vektorer.

Vi vet att Cliffordprodukten är antikommutativ för vektorer enligt ekv. (11).

$$e_i e_{jk} = \begin{cases} -e_{jk} e_i \text{ om } i = j \text{ eller } i = k \\ e_{jk} e_i \text{ om } i = j = k \text{ eller } k \neq i \neq j \end{cases} \tag{19}$$

Vad vi kan se är att om vektorn e_i är en del av bivektorn, t.ex $e_1 e_{31}$, så gör den ett ojämnt antal "hopp" och resulterar därför i antikommutativitet, och vice versa om e_i inte ligger i bivektorn. Men vi kan hitta q_2^{-1} och q_1 så att vi får en bivektordel som inte kommuterar måste bivektordelen vara 0. Det enda som återstår är då en skalär. Då spin(V) är sluten under Cliffordprodukten så måste $q_2^{-1} q_1 \in \text{spin}(V)$ och därmed

$$q_2^{-1} q_1 = \pm 1 \rightarrow q_1 = \pm q_2. \tag{20}$$

Vi har alltså för en given rotation T finns det exakt två kvaternioner $\pm q \in spin(V)$ som svarar mot rotationen. Dessa två är antipodala kvaternioner på S^3. Vi kan nu, med samma metod som ovan, få fram rotationsmatrisen T för ett godtckligt $q = \exp(-\theta j/2)$, där
$j = ae_{32} + be_{13} + ce_{21},\ a^2 + b^2 + c^2 = 1$ ges av

$$\begin{bmatrix} \cos^2(\theta/2)+\sin^2(\theta/2)(a^2-b^2-c^2) & 2ab\sin^2(\theta/2)-c\sin(\theta) & 2ac\sin^2(\theta/2)+b\sin(\theta) \\ 2ab\sin^2(\theta/2)+c\sin(\theta) & cos^2(\theta/2)+\sin^2(\theta/2)(-a^2+b^2-c^2) & 2b\sin^2(\theta/2)-a\sin(\theta) \\ 2ac\sin^2(\theta/2)-b\sin(\theta) & 2bc\sin^2(\theta/2)+a\sin(\theta) & \cos^2(\theta/2)+\sin^2(\theta/2)(-a^2-b^2+c^2) \end{bmatrix}. \tag{21}$$

Härledning för denna finns i appendix B.3.

Vi vill nu hitta en invers så att vi, för en given rotationsmatris, kan få fram rotationsaxel och vinkel. Vi kan givietvis få fram den givna kvaternionen från rotationsmatrisen genom att jämföra med ekv. (21). Men detta kräver mycket arbete. Istället kan vi definiera Cliffordspåret $Tr_C(R)$ (1) som

$$Tr_C(R) := \sum_i e_i \triangle R(e_i). \tag{22}$$

Vi får då

$$\begin{aligned} Tr_C(R) = e_1 \triangle (R_{11}e_1 + R_{21}e_2 + R_{31}e_3) + \\ e_2 \triangle (R_{12}e_1 + R_{22}e_2 + R_{32}e_3) + \\ e_3 \triangle (R_{13}e_1 + R_{23}e_2 + R_{33}e_3) = \\ Tr(R) + e_{12}(R_{21} - R_{12}) + e_{13}(R_{31} - R_{13}) + e_{23}(R_{32} - R_{23}). \end{aligned} \tag{23}$$

Där $Tr(R)$ representerar det vanliga spåret hos matrisen, $Tr(A) := \sum_n a_{nn}$. Alltså sumeringen av diagonalelementen.

Anta att vi nu vet rotationsaxeln. Låt $\{\tilde{e_i}\}$ vara en ON-bas för V, där $j = \tilde{e_2} \wedge \tilde{e_3}$ är bivektorn parallel med planet för rotationen. Rotationsmatrisen i basen $\{e_i'\}$ kommer då att bli

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{24}$$

Cliffordspåret kommer nu, med ekv. (23) att bli

$$Tr_C(R) = 1 + 2(\cos\phi + j\sin\phi). \tag{25}$$

Cliffordspåret är invariant till val av bas enligt följande lemma.

Lemma. *För en linjär avbildning T har vi att Cliffordspåret är invariant till val av bas. Alltså för två baser $\{e_i\}$ och $\{\tilde{e_i}\}$ Har vi att*

$$Tr_C(T_{e_i}) = Tr_C(T_{\tilde{e_i}}).$$

Där indexet då representerar avbildningen i de två baserna.

Bevis. Låt $\{\tilde{e}_i\}$ vara en ON-bas till V där $\tilde{e}_i = \sum_i a_{j,i}e_j$ där e_j är basvektorer i standardbasen. Basbytesmatrisen A är ortogonal. Vidare gäller att

$$\begin{aligned} \langle \tilde{e}_i, \tilde{e}_k \rangle = \langle \sum_j a_{j,i}e_j, \sum_p a_{p,k}e_p \rangle \\ = \sum_j a_{j,i} \sum_p a_{p,k} \langle e_j, e_p \rangle = \sum_j a_{j,i}a_{j,k} = \delta_{ik}. \end{aligned} \tag{26}$$

Cliffordspåret i basen $\tilde{e}_j$ blir nu

$$\begin{aligned} Tr_C(T) &= \sum_i \sum_j a_{j,i}e_j \triangle T(\sum_k a_{k,i}e_k) \\ &= \sum_i \sum_j a_{j,i}e_j \triangle \sum_k a_{k,i}T(e_k) \\ &= \sum_i \sum_j \sum_k a_{j,i}a_{k,i}e_j \triangle T(e_k). \end{aligned} \tag{27}$$

Då summation är kommutativ spelar det ingen roll vilken summa vi tar först. Vidare, då A är ortogonal, är även A^T. Vi får då

$$\begin{aligned} Tr_C(T) &= \sum_j \sum_k \sum_i a_{j,i}a_{k,i}e_j \triangle T(e_k) \\ &= \sum_j \sum_k \delta_{jk}e_j \triangle T(e_k) \\ &= \sum_j e_j \triangle T(e_j) = Tr_C(T). \end{aligned} \tag{28}$$

□

Vi kan nu, genom att termjämföra ekv. (23) och (25), få fram ϕ och j som

$$\begin{aligned} \phi = \arccos\left(\frac{Tr(R)-1}{2}\right), \\ j = \frac{1}{\sin(\phi)}\Big[e_{12}(R_{21}-R_{12}) + e_{13}(R_{31}-R_{13}) + e_{23}(R_{32}-R_{23})\Big]. \end{aligned} \tag{29}$$

3.2 Kontinuerliga rotationer och spinn

Än så länge har vi att för varje rotation finns det exakt 2 punkter på S^3 som motsvarar en rotation. Vi skall nu undersöka vad som händer med ett system när vi roterar det kontinuerligt.

3.2.1 Parametrisering av S^3

Då S^3 är en tredimensionell mångfald inbäddad i 4D är det är svårt att visualisera den på ett bra sätt. Det som vi gör här är att vi lyfter alla punkter på S^3 till ett klot i $\triangle^2 V$. Betrakta funktionen

$$\begin{aligned} p : \Delta^2 V &\to S^3, \\ p : b &\to q. \end{aligned} \tag{30}$$

Där $b \in \Delta^2 V$. Vi skriver det som $b = \phi j$, där $\phi = |b| \mod 4\pi$ och $j = \frac{b}{|b|}$. Vidare så har p formen $p(b) = \exp(b/2) = \cos(\phi/2) + j\sin(\phi/2)$.

Randen, där $|b| = 2\pi + 4\pi k,\ k = 0,\ 1,\ 2,\ ....$ Alla punkter är identifierade med samma kvaternion $q = -1$. Vi ser här att för $|b| = 4\pi k$ är $p(b) = 1$. Vidare för $|b| = \pi + 2\pi k$ så har vi att $p(b) = b$. Vi har att p en bijektiv avbildning för $|b| < 2\pi$ på grund av injektiviteten hos exponentialfunktionen.

3.2.2 S^3 enkelt sammanhängande

För en mångfald som är enkelt sammanhängande betyder det att alla slutna kurvor är homotopa, vidare nollhomotopa.

Vi kommer här att bevisa att S^3 är enkelt sammanhängande. Betrakta en sluten, kontinuerlig kurva på S^3 som inte går genom -1,

$$\begin{aligned} \gamma : [0,\ 1] &\to S^3, \\ \gamma(0) = \gamma(1) &= \hat{q}, \end{aligned} \tag{31}$$

där $\hat{q}$ är en fix punkt. Vi lyfter γ till bivektorklotet $\Delta^2 V$ genom ekv. (30). Vi har nu en kurva, $p^{-1}(\gamma(t))$ i $\Delta^2 V$. Vi vill hitta en homotopi mellan denna kurva och dess startpunkt, $\hat{q}$. Låt

$$\begin{aligned} [0,\ 1] \times [0,\ 1] \to \Delta^2_{2\pi} V : (s,t) \to \Gamma(s,t), \\ \Gamma(s,t) = p^{-1}(\gamma(t)) + s(p^{-1}(\hat{q}) - p^{-1}(\gamma(t))) \end{aligned} \tag{32}$$

vara en yta genom $\Delta^2 V$. Vi ser att $\Gamma(0,t) = p^{-1}(\gamma(t))$, medan $\Gamma(1,t) = p^{-1}(\hat{q})$. Alltså, Γ kan kontinuerligt deformeras från $p^{-1}(\gamma(t))$ till $p^{-1}(\hat{q})$. För fallet när kurvan går genom sydpolen, -1, kan vi låta den undvika sydpolen innan vi lyfter den med p genom att t.ex låta kurvan gå i en liten halvcirkel runt sydpolen.

Alla slutna kurvor på S^3 är därmed homotopa. Fundamentalgruppen är den grupp av skilda ekvivalensklasser för en given mängd. I detta fallet de slutna kurvor som ej är homotopa. Fundamentalgruppen för S^3 är då $\pi_1(S^3) = \{1\}$.

3.2.3 Fundamentalgruppen för $SO(3)$

Vi vill nu undersöka slutna rotationskurvor i $SO(3)$ för att kunna påvisa $\mathbf{Z}_2$-minnet.

Anta på samma sätt som i ekv. (31) en kontinuerlig kurva genom $SO(3)$,

$$\begin{aligned} T : t \in [0,\ 1] &\to SO(3), \\ T(0) = T(1) &= I. \end{aligned} \tag{33}$$

På samma sätt som vi lyfte en kurva på S^3 till $\Delta^2 V$ vill vi nu lyfta $T(t)$ till S^3. Vi har en homomorfi från kvaternioner på S^3 till matriser i $SO(3)$

$$\begin{aligned} \hat{p} : S^3 &\to SO(3), \\ T &\to q. \end{aligned} \tag{34}$$

Formen för denna homomorfi kan t.ex ges från den allmäna rotationsmatrisen i ekv. (21). Skillnaden mellan lyftet i ekv. (34) och lyftet i ekv. (30) är att för varje $T \in SO(3)$ finns $\pm q \in S^3$. Vi har alltså surjektivitet men inte injektivitet. Vi kan dock få en typ av lokal injektivitet genom att bara betrakta en av dessa kvaternioner. Då de två kvaternionerna, som motsvarar samma $T \in SO(3)$, är antipodala kommer deras vägar aldrig korsas om vi gör en liten förflyttning av T. Vi får alltså en lokal isomorfi från $S^3 \to SO(3)$ när vi betraktar kontinuerliga rotationer och bara betraktar ett $q \in S^3$. Vi låter nu $\hat{p}^{-1}$ vara inversen. Vi har då att

$$\begin{aligned} \hat{p}^{-1} \circ T(t): \ & SO(3) \to S^3, \\ & \hat{p} \circ T(0) = 1. \end{aligned} \tag{35}$$

Så kurvan på S^3 börjar på 1 och kommer nu att röra sig över sfären. När t närmar sig 1 finns det nu två olika punkter den kan sluta på. Då $\hat{p}^{-1} \circ T(I) = \pm 1$ så kommer kurvan att sluta på antingen 1 eller -1. Det finns alltså två skilda vägar som inte tillhör samma ekvivalensklass. Den ena, som är sluten på S^3 och därmed motsvarar nollhomotop, är samma system som vi började med. Den kurva som går från nordpol till sydpol motsvarar inte längre samma system, även om det ser likadant ut. Denna kurva är homotop med en kurva som motsvarar rotation av 2π radianer runt en valfri axel, vilket vi kan se genom lyftet p i ekv. (30). Den kurva som är sluten är vidare homotop med en kurva som går ner till -1 och sedan upp på andra sidan av S^3. Alltså en rotation av 4π radianer runt en given axel. Det följer från dessa två skilda kurvor att fundamentalgruppen för rotationsgruppen i tre dimensioner är

$$\pi_1(SO(3)) = \{1, \ -1\} := \mathbf{Z}_2. \tag{36}$$

Detta avslutar nu avsnittet om rotationer i $\mathbb{R}^3$. Vi har visat hur vi väldigt enkelt kan rotera vektorer med hjälp av kvaternioner. Men vi kan inte längre använda $SO(3)$ för att beskriva rotationer av objekt som har detta $\mathbf{Z}_2$-minne. Vi kommer nu att flytta fokus från vektorer till spinorer.

4 Spinorer

$\mathbf{Z}_2$-minnet återkommer, som vi nämnt, till exempel inom kvantmekaniken. En elektron vars spinntillstånd har roterat 2π radianer är inte längre i sitt ursprungliga grundtillstånd utan har istället hamnat i motsatt tillstånd.

Vad vi menar med tillstånd och rotation här är ett kvantmekaniskt tillstånd, och en rotation här är en rotation i tillståndsrummet (dock är denna relaterad till en ”fysikalisk rotation”, som beter sig mer normalt, med en period på 2π): $|\psi\rangle = \psi^+ |+\rangle + \psi^- |-\rangle$, där ψ^+, ψ^- är komplexa tal, och $|+\rangle, |-\rangle$ är tillstånden ”spinn upp” respektive ”spinn ned”, ett kvantmekaniskt tillstånd motsvarar en sannolikhet i den mening att exempelvis om man mäter ett system i tillstånd $|\psi\rangle$ i ”spinn upp/ned” riktningen är $\langle +|\psi\rangle$ sannolikheten att man får ett mätvärde i ”spinn upp”, där $+\psi$ är en seskvilinjär skalärprodukt mellan det kvantmekaniska tillståndsrummet (beskrivs med $|\rangle$, en ”cket”) som är ett Hilbertrum, och dess duala rum (beskrivs med $\langle|$, en ”bra”, tillsammans utgör de ”bracket” notationen).

I kvantmekaniken finns ”operatorer”, objekt som kan ändra ett tillstånd, och om en operator A är hermitesk är $\langle a|A|a\rangle$ ett väntevärde som motsvarar en observabel som man kan mäta. Tidsutvecklingen hos ett kvantmekaniskt system är beroende av dess hamiltonian, H, som alltid är hermitesk och om hamiltonianen är tidsoberoende fås evolutionen som en operator $exp(-iHt/\hbar)$ (specifikt är tidsutvecklingen hos ett system dess tillstånd vid specifika tidspunkter, och blir $exp(-iHt/\hbar)\,|t_0\rangle$, där $|t_0\rangle$ är tillståndet i tid $|t_0\rangle$, för det tillstånd som intresserar oss). Det fall som intresserar oss är när en elektron befinner sig i ett externt statiskt (tidsoberoende) magnetfält, och hamiltonianen kan då skrivas som $H = \omega S_z$, där $\omega = |e|B/(m_e c)$.

Tidsevolutionen ges då av $exp(-i\omega S_z/\hbar)\,|\psi\rangle = exp(-i\omega t S_z/\hbar)(\psi^+ |+\rangle + \psi^- |-\rangle) = \psi^+ e^{(-i\omega/\hbar)S_z} |+\rangle + \psi^- e^{(-i\omega t/\hbar)S_z} |-\rangle = \psi^+ e^{-i\omega t/2} |+\rangle + \psi^- e^{i\omega t/2} |-\rangle$. Detta är eftersom $|+\rangle, |-\rangle$ är spinntillstånd i z-led, och är därmed egentillstånd (motsvarar egenvektorer) till S_z operatorn. Den har egenvärden $\pm\hbar/2$ ($exp(A)$ för operatorn A betraktas i stort på samma sätt som om A vore en matris, i detta fall kan operatorn S_z till och med beskrivas med en matris. Med andra ord eftersom $|+\rangle, |-\rangle$ är egentillstånd till S_z med egenvärden $\pm\hbar/2$ är de även egentillstånd till $exp(S_z)$ med egenvärden $e^{\pm\hbar/2}$). Om man betraktar tidsutvecklingen för $|\psi\rangle$ ser man i exponentialen $\pm i\omega t/2$. Division med 2 innebär att vi får en period $T = 4\pi/\omega$, vilket är en annan precession än den som skulle vara om man mäter på egenvärdet $\langle\psi|S_z|\psi\rangle$, det vill säga om man mäter väntevärdet på spinn i z-led, som har period $T = 2\pi/\omega$. Det kommer senare i texten uppdagas för läsaren att tillståndet vi bemärkte med $|\psi\rangle$ i inledningen är en spinor, vars natur kommer gås igenom i texten.

Så med ”motsatt tillstånd” här menas inte övergång i stil med ”spinn upp” $\rightarrow$ ”spinn ned” utan istället $|\psi\rangle \rightarrow -|\psi\rangle$, eller ”spinn upp” $\rightarrow$ ”minus spinn upp”. Denna övergång är dock mätbar, med exempelvis neutroninterferometri.

Om läsaren vill fördjupa sig i ämnet rekommenderas varmt (3).

$\mathbf{Z}_2$-minnet hos elektroner medför att $SO(3)$ inte kan beskriva en elektrons spinn. Vad som krävs är en representation av $spin(V)$, ρ, sådan att $\rho(-1) = -I$. Vi kommer i detta avsnitt gå in på just detta. Vi definierar en representation ρ från den jämna Cliffordalgebran till algebran av matriser för ett tillhörande spinorrum. Spinorrummet är alltså ett rum som existerar med ett tillhörande vektorrum. Vi kommer senare visa att spinorrummet är oberoende av hur vi representerar Cliffordalgebran. De olika representationerna är relaterade via basbyte.

4.1 Gruppprepresentation

Vi kommer nu att börja titta på spinorummet. Vi definierar först en gruppprepresentation av $spin(V)$ som en slät grupphomomorfi från $spin(V)$ till matriser för ett linjärt rum

$$\begin{aligned} \rho : \mathrm{spin}(V) &\to \mathcal{L}(S), \\ \rho(-q) &= -\rho(q). \end{aligned} \tag{37}$$

Vi får en injektiv representation så att för varje matris i bilden av ρ finns det exakt en rotor i spin(V). På samma sätt som bivektorer kan användas för att rotera vektorer kan de även användas för rotation av spinorer. För en representation med egenskapen att $\rho(-q) = -\rho(q)$ kallar vi S för ett spinorrum. Spinorerna är då de objekt som matriserna i $\mathcal{L}(S)$ opererar på.

Vi kommer att ta fram matrisrepresentationer av Cliffordalgebran då $spin(V) \subset \mathbb{H} \subset \Delta V$ och så kommer vi, genom våra representationer av ΔV, även att få en grupprepresentation för $spin(V)$.

4.2 Dimensionsanalys

Vi kommer återigen att begränsa oss till att $V = \mathbb{R}^3$. Vi har då att $\dim_{\mathbb{R}}(\Delta V) = 8$. Vi kommer här att ta fram representationer till tvådimensionella komplexa matriser. Algebran för dessa matriser betecknas $\mathcal{L}(\mathbb{C}^2)$. Vi vill hitta en isomorfi mellan Cliffordalgebran och matriserna för spinorrumet. För att kunna täcka hela matrisalgebran behöver vi att dimensionerna är samma, i detta fallet är både definitionsmängd och värdemängd 8 dimensionellt. Skillnaden är dock att vi går från en reell algebra till en komplex. Genom att låta trivektorn $e_{123} := J$ har vi att $J^2 = -1$. Vidare kan vi skriva en multivektor i Cliffordalgebran som $w = w_0 + w_1$, där $w_0 \in \Delta^{ev}V$ och $w_1 \in \Delta^{od}V$. Vi låter nu $w_1 = Jw_2$, där $w_2 \in \Delta^{ev}V$. Det vi ser nu som att ΔV är isomorft med de komplexa kvaternionerna $\mathbb{H}_c$. Vi skriver en komplex kvaternion som $q_c = q_1 + Jq_2$, där q_1 och q_2 båda är reella kvaternioner enligt representationen vi gjorde i avsnitt 3.1. Vi har nu en komplex algebra $\mathbb{H}_c$ sådan att $\dim_{\mathbb{R}}(\mathbb{H}_c) = 8 = \dim_{\mathbb{R}}(\mathcal{L}(\mathbb{C}^2))$. Vi kan nu hitta en isomorfi mellan de komplexa kvaternionerna och matriser i $\mathcal{L}(\mathbb{C}^2)$.

4.3 Matrisrepresentationer av Cliffordalgebran

Här kommer vi att ta fram matrisrepresentationer av Cliffordalgebran och visa att de olika representationerna inte påverkar strukturen hos spinorrummet utan förser oss endast med ett annat perspektiv av det.

4.4 Matriser för Cliffordalgebran

I detta avsnitt kommer vi att visa hur vi kan representera underalgebror hos Cliffordalgebran med matriser. En algebrahomomorfi är en homomorfi $\rho : \Delta V \to \mathcal{L}(S)$, sådan att för $w \in \Delta V$,

$$\rho(w)^2 = \rho(w^2) = \rho(\langle w \rangle^2) = \langle w \rangle^2 I.$$

4.4.1 Komplexa matriser som reella matriser

Vi vill först hitta ett sätt att beskriva komplexa matriser som reella matriser.

Vi tittar först på det endimensionella fallet för $\mathbb{C}$

$$\varphi_1 : \mathcal{L}(\mathbb{C}) \longrightarrow \mathcal{L}(R^2) = \Delta R^2 \Longrightarrow \varphi_1(a + bi) := \begin{bmatrix} a & b \\ -b & a \end{bmatrix}, a, b \in R^2. \tag{38}$$

Det går lätt att visa att φ_1 är en injektiv homomorfi, det är dock ingen isomorfi då den ej är surjektiv.

Vi går nu vidare genom att utvidga φ_1 till komplexa 2 x 2-matriser. Vi får för $z_i = a_i + ib_i$,

$$
\begin{aligned}
&\varphi_2 : \mathcal{L}(\mathbb{C}^2) \to \mathcal{L}(\mathbb{R}^4), \\
\begin{bmatrix} z_1 & z_2 \\ z_3 & z_4 \end{bmatrix} = \begin{bmatrix} a_1 + ib_1 & a_2 + ib_2 \\ a_3 + ib_3 & a_4 + ib_4 \end{bmatrix} &\to \begin{bmatrix} \varphi_1(a_1 + ib_1) & \varphi_1(a_2 + ib_2) \\ \varphi_1(a_3 + ib_3) & \varphi_1(a_4 + ib_4) \end{bmatrix} \\
&= \begin{bmatrix} a_1 & b_1 & a_2 & b_2 \\ -b_1 & a_1 & -b_2 & a_2 \\ a_3 & b_3 & a_4 & b_4 \\ -b_3 & a_3 & -b_4 & a_4 \end{bmatrix}.
\end{aligned} \tag{39}
$$

Funktionen $\varphi_2 : \mathcal{L}(\mathbb{C}^2) \to \mathcal{L}(\mathbb{R}^4)$ är en injektiv homomorfi.

Vi kan även hitta en isomorfi för vektorerna som dessa matriser verkar på sådan att

$$
\begin{aligned}
&\varrho : \mathbb{C}^2 \to \mathbb{R}^4, \\
\begin{bmatrix} z \\ w \end{bmatrix} = \begin{bmatrix} a_1 + ib_1 \\ a_2 + ib_2 \end{bmatrix} &\to \begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{bmatrix} \in R^4.
\end{aligned}
$$

4.4.2 Representation av kvaternioner som komplexa matriser

Vi ska nu ta fram de två representationer för kvaternionerna vi kommer att använda i detta arbete. Vi vill hitta en isomorfi $\rho : \mathbb{H}_c \to \mathcal{L}(\mathbb{C}^2)$, där är $\mathbb{H}_c$ enligt ovan isomorft med $\Delta\mathbb{R}^3$.

Vi kommer först att betrakta representationer av reella kvaternioner $\mathbb{H}$, så att för $q = a + bi + cj + dk \in \mathbb{H}$, där $a,\ b,\ c,\ d \in \mathbb{R}$. q kan skrivas som $q = a + bi + cj + dij = (a + bi) + (c + di)j = z + wj$. Vi kan nu använda oss utav φ_1 för att ta fram en liknande homomorfi för kvaternionerna. Från (2), får vi en representation

$$
\begin{aligned}
&\rho_1 : \mathbb{H} \to \mathcal{L}(C^2), \\
&\rho_1(z + wj) := \begin{bmatrix} z & w \\ -\bar{w} & \bar{z} \end{bmatrix}.
\end{aligned} \tag{40}
$$

Där är $\bar{w}, \bar{z}$ respektiv komplex konjugation av w och z.

Från (1) får vi nu ytterligare en representation för $\mathbb{H}$,

$$
\begin{aligned}
&\rho_2 : \mathbb{H} \to \mathcal{L}(\mathbb{C}^2), \\
&\rho_2(q = a + bi + cj + dk) = \begin{bmatrix} a - ci & d - bi \\ -d - bi & a + ci \end{bmatrix}.
\end{aligned} \tag{41}
$$

För att nu få en isomorfi behöver vi att dessa representationer kan utvidgas till de komplexa kvaternionerna $q = a + bi + cj + dk$, där $a,\ b,\ c,\ d \in \Delta^0 V \oplus \Delta^3 V$ alltså är tal med den komplexa struktur J som ansattes i 4.2. Låt nu $\rho : \mathbb{H} \to \mathcal{L}(\mathbb{C})^2$ vara en homomorfi. Vi komplexifierar avbildningen ρ till $\tilde{\rho} : \mathbb{H}_c \to \mathcal{L}(\mathbb{C}^2)$ sådan att för den komplexa strukturen J har vi att $\tilde{\rho}(J) = iI$, där $i \in \mathbb{C}$. En kvaternion $\mathbb{H}_c \ni q = q_1 + Jq_2$ representeras nu genom $\tilde{\rho}(q) = \rho(q_1) + i\rho(q_2)$. Det följer av de homomorfa egenskaperna hos ρ att detta är en giltlig komplexifiering. Det följer från appendix B.2 att ρ_1 och ρ_2 är homomorfier. Vi utvidgar nu dessa till att även representera de komplexa kvaternionerna. Vi har nu två surjektiva homomorfier från ΔV till $\mathcal{L}(\mathbb{C}^2)$. Vad som är kvar att visa är injektivitet.

Lemma (Injektivitet hos representationer)**.** *En representation $\rho : \mathbb{H} \to \mathcal{L}(\mathbb{C}^2)$ som är en homomorfi är också injektiv.*

Bevis. Låt $\rho : \mathbb{H}_c \to \mathcal{L}(\mathbb{C}^2)$ vara en homomorfi som ovan. För att ρ skall vara injektiv gäller att $\rho(q) = 0$ om och endast om $q = 0$. Då representationen $q = a + bi + cj + dk$ är linjär följer att,

$$\rho(q) = \rho(a + bi + cj + dk) = a\rho(1) + b\rho(i) + c\rho(j) + d\rho(k) = 0.$$

Vi multiplicerar nu från vänster med konjugatet $\rho(\bar{q})$,

$$\begin{aligned} \rho(\bar{q})\rho(q) &= \rho(\bar{q}q) \\ &= \rho(a^2 + b^2 + c^2 + d^2) \\ &= (a^2 + b^2 + c^2 + d^2)\rho(1) = \rho(\bar{q})0 = 0. \end{aligned}$$

Detta gäller endast om $a^2 + b^2 + c^2 + d^2 = 0 \Leftrightarrow a = b = c = d = 0$. Vi ser nu att om ρ är en homomorfi gäller för $\rho(q_1) = \rho(q_2) \Leftrightarrow \rho(q_1 - q_2) = 0 \Leftrightarrow q_1 = q_2$, vilket visar injektivitet. □

Vi har nu alltså två isomorfier som båda representerar Cliffordalgebran till V.

4.4.3 Koppling mellan representationerna

Vi ska här visa relationerna mellan två representationer av Cliffordalgebran. Vi kommer hitta en automorfi mellan representationerna som relaterar dessa via basbyte.

Vi har våra två representationer av kvaternionerna.

$$\begin{aligned} \rho_1 : \mathbb{H}_c &\to \mathcal{L}(\mathbb{C}), \\ \rho_2 : \mathbb{H}_c &\to \mathcal{L}(\mathbb{C}), \\ \rho_1(a + bi + cj + dk) &= \begin{bmatrix} a + bi & c + di \\ -c + di & a - bi \end{bmatrix}, \\ \rho_2(a + bi + cj + dk) &= \begin{bmatrix} a - ci & d - bi \\ -d - bi & a + ci \end{bmatrix}. \end{aligned} \tag{42}$$

Vi undersöker nu relationerna mellan dessa. Båda två avbildar 1 på I och -1 på $-I$. $\mathcal{L}(\mathbb{C}^2)$ är därmed fortfarande operatorer till ett tillhörande spinorrum. Vi vill nu hitta en koppling mellan dessa två representationer. Vi kan definiera en automorfi $\phi : \mathcal{L}(\mathbb{C}) \to \mathcal{L}(\mathbb{C})$ sådan att

$$\phi(\rho_1(q)) = \rho_2(q). \tag{43}$$

För $\rho_1(q) = M \in \mathcal{L}(\mathbb{C}^2)$ har vi då att $\phi(M) = \rho_2 \circ \rho_1^{-1}(M) = \rho_2 \circ \rho_1^{-1}(\rho_1(q)) = \rho_2(q)$, vilket är vad vi letar efter. Denna automorfi är väldefinierad då inversen existerar. Låt oss nu observera strukturen hos dessa två representationer,

$$\begin{aligned} \rho_1(1) &= I = \rho_2(1), \\ \rho_1(-1) &= -I = \rho_2(-1), \\ \rho_1(i) &= \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} = \rho_2(-j), \\ \rho_1(j) &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \rho_2(k), \\ \rho_1(k) &= \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix} = \rho_2(-i). \end{aligned} \tag{44}$$

Så vi har att för $q \in \mathbb{H}$,

$$\begin{aligned} \pm 1 &\to \pm 1, \\ i &\to -j, \\ j &\to k, \\ k &\to -i. \end{aligned} \tag{45}$$

Vi försöker nu hitta ett $q_s = a + bi + cj + dk$, så $q \to q_s q q_s^{-1}$ som ger samma struktur som ekv. (45) ovan.

$$q_s i = -j q_s \Leftrightarrow ai - b - ck + dj = -aj + bk + c - di, \tag{46}$$

detta ger $c = -b$, $d = -a$.

$$q_s j = -k q_s \Leftrightarrow aj + bk + b + ai = -ak - bj - bi - a, \tag{47}$$

detta ger $a = -b$. Vi har därmed $q_s = a - ai + aj - ak$ normerar vi detta får vi $a = 1/2$. ρ_1 och ρ_2 är därmed relaterade med en rotation i $\mathbb{H}$. Vi har alltså

$$\rho_1(q_s q q_s^{-1}) = \rho_1(q_s)\rho_1(q)\rho_1(q_s^{-1}) = \rho_2(q). \tag{48}$$

Matrisen $\rho_1(q_s)$ ser ut som

$$\rho_1(q_s) = T = \frac{1}{2}\begin{bmatrix} 1-i & 1-i \\ -1-i & 1+i \end{bmatrix}. \tag{49}$$

Vi har nu en automorfi ϕ sådan att

$$\phi(\rho_1(q)) = T\rho_1(q)T^{-1} = \rho_2(q). \tag{50}$$

Vi ser här att dessa två representationer är relaterade via ett basbyte enligt appendix B.1. Det visar sig faktiskt att alla representaioner för $\mathcal{L}(\mathbb{C}^2)$ är relaterade via ett basbyte. Detta summeras i satsen nedan.

Sats (Automorfier i $\mathcal{L}(S)$ då $\dim_{\mathbb{C}}(S) = 2$)**.** *Algebran för matriser till ett vektorrum S av komplex dimension 2 är komplett, i den meningen att för varje automorfi $\phi : \mathcal{L}(S) \to \mathcal{L}(S)$, $\exists\, T \in \mathcal{L}(S)$ så att $\phi(X) = TXT^{-1}$, $\forall X \in \mathcal{L}(S)$. T är unik upp till multiplikation med skalär.*

Bevis. (1) Detta bevis är tvådelat. Låt först, $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, $X = v\begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} v_1 & 0 \\ v_2 & 0 \end{bmatrix}$ och $E = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$.

Eftersom E inte är inverterbar är inte heller $\phi(E)$ det, med andra ord kolumnerna i $\phi(E)$ är linjärt beroende. Låt nu $\{e_i\}$ vara en ON-bas för S så att e_1 är parallell med kolumnerna i $(\phi(E))$, vidare låt $e_2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

Vi ser att

$$XE = 0 \Rightarrow \phi(XE) = \phi(X)\phi(E) = 0,$$

då är kolumnerna i $\phi(E) \parallel e_1$. Detta är ekvivalent med att $\phi(X)e_1 = 0$.

Vi har nu att

$$\begin{aligned} \phi(X) &= \phi(X)I = \\ &= \phi(X)(e_1 e_1^\top + e_2 e_2^\top) = \phi(X)e_2 e_2^\top. \end{aligned}$$

Låter vi nu

$$Av := \phi(X)e_2.$$

Får vi genom multiplikation från vänster av $e_2^\top$,

$$\phi(X) = Av\begin{bmatrix} x_1 & x_2 \end{bmatrix}.$$

Med liknande argumentation kan vi visa att för en matris $U = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u^\top = \begin{bmatrix} 0 & 0 \\ u_1 & u_2 \end{bmatrix}$.

$$\phi(U) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} u^\top B,$$

där $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$.

Vidare gäller att

$$\phi(vu^\top) = \phi(v \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} u^\top)$$
$$= \phi(X)\phi(U) = Av \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} u^\top B$$
$$= \lambda A v u^\top B,$$

där $\lambda = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$. Låt nu $T := \lambda A$. Vi har då $\forall u, v \in S : \phi(vu^\top) = Avu^\top B$.

Låt oss nu titta närmare på

$$\phi(v_1 u_1^\top v_2 u_2^\top) = \phi(v_1 \langle u_1, v_2 \rangle u_2^\top) = \langle u_1, v_2 \rangle \phi(v_1 u_2^\top)$$
$$= \langle u_1, v_2 \rangle T v_1 u_2^\top B. \tag{51}$$

Samtidigt har vi att

$$\phi(v_1 u_1^\top v_2 u_2^\top) = \phi(v_1 u_1^\top)\phi(v_2 u_2^\top) = T v_1 u_1^\top B T v_2 u_2^\top B.$$

Vi ser nu att

$$\langle u_1, v_2 \rangle T v_1 u_2^\top B = A v_1 u_1^\top B T v_2 u_2^\top B.$$

Vi flyttar in $\langle u_1, v_2 \rangle$,

$$T v_1 u_1^\top v_2 u_2^\top B = T v_1 u_1^\top B T v_2 u_2^\top B.$$

Här kan vi deducera att $BT = I \Leftrightarrow B = T^{-1}$ alltså för matriser på formen $vu^\top$,

$$\phi(vu^\top) = T v u^\top T^{-1}.$$

För godtyckliga matriser kan vi skriva dessa som en linjärkombination utav matriser på formen $vu^\top$. Vi kan sedan använda linjäriteten hos automorfin och får då att

$$\phi(X) = TXT^{-1} \quad \forall X \in \mathcal{L}(S).$$

Att de är unika ses lättast genom

$$T_1 X T_1^{-1} = T_2 X T_2^{-1} \Leftrightarrow T_2^{-1} T_1 X = X T_2^{-1} T_1.$$

Med X godtycklig har vi att $T_2^{-1}T_1$ kommuterar med samtliga matriser. Eftersom de 2x2 komplexvärda matriserna är isomorfa med $\mathbb{H}_c$ innebär det att det finns en motsvarande komplexvärd kvaternion $q_{T_2^{-1}T_1}$ som kommuterar med samtliga komplexvärda kvaternioner. De enda kvaternioner som kommuterar med samtliga är skalära tal (eftersom att i j och k ej kommuterar). Detta medför att $q_{T_2^{-1}T_1} = \beta$, där är β ett komplext tal. Om vi nu återgår isomorft till matriserna fås $T_2^{-1}T_1 = \beta I$, alltså $T_1 = \beta T_2$. □

Vad vi kommit fram till här är att spinorrummet är ett geometriskt rum inbäddat i Cliffordalgebran. Vi kan hitta olika representationer för det med dessa olika förser en bara med olika perspektiv på spinorerna.

Detta avslutar nu teoridelen. Vi har visat hur vi kan rotera vektorer i $\mathbb{R}^3$ med hjälp av kvaternioner. Vidare har vi även visat $\mathbf{Z}_2$-minnet hos $SO(3)$ och argumenterat för att vi behöver en representation för att beskriva objekt som har detta minne. Spinorerna har då detta $\mathbf{Z}_2$-minne som inte kan observeras genom vanliga vektorer. Vi har tagit fram representationer för operatorer på spinorerna.

I vårt interaktiva program har vi tagit fram visualiseringar av vektorummet och spinorrummet och kommer att kunna demonstrera detta fenomen tydligt.

5 Kodningen bakom VR, BR och SR

Under projektet har vi producerat tre program, VR, BR och SR (står för Vektorrymd, Bivektorrymd samt Spinorrymd). VR och BR är visualiseringar av de parametriseringar mellan bivektorrymden och vektorrymden

$$BR \to \mathcal{L}(VR), \quad b \to T, \quad Tv = e^{b/2} v e^{-b/2}.$$

(se även (30) samt avsnittet om kvaternioner och rotationer i $\mathbb{R}^3$ 3.1.) De är skapade i pedagogiskt syfte. Användaren skall kunna visuellt få demonstrerat för sig att $SO(3)$ har ett $\mathbf{Z}_2$-minne i den meningen att när ett vektorsystem roterat ett helt varv kommer inte en punkt i bivektorrymden som kontinuerligt följt detta system ha återställts. Bivektorn befinner sig då istället på randen, $|b| = 2\pi$. Alla b som uppfyller detta motsvarar, enligt ekv. (30), samma kvaternion -1, $b \to e^{b/2}$.

SR visualiserar representationen mellan bivektorrymden och spinorrymden

$$BR \to \mathcal{L}(SR), \quad b \to \tilde{T}, \quad \tilde{T}\psi = \rho(e^{b/2})\psi.$$

Där ψ är en spinor. Man kan se i programmet som visualiserar spinorer att efter att en punkt i bivektorrymden förflyttats till sin rand kommer spinorer inte ha återställts, utan har istället hamnat i motsatt konfiguration: $\psi \to -\psi$.

BR och SR behöver programmet quaternion.m för att fungera. SR behöver även platonic_solid.m.

I samtliga program finns en Bivektor-Spinor- samt Vektorrymd uppvisade som grafiska element. Bivektorrymden består av ett klot med radie 2π, där motsvarar punkterna de enhetskvaternioner som finns i S^3, och relateras till de rena bivektorerna med $q = exp(b/2)$. Det finns däri ett litet rött klot som representerar en punkt i bivektorklotet. Den sitter till en början i origo men när programmet används kommer den att börja flytta runt. Tre ”skuggbilder”, mer transparenta punkter projicerade på de tre axlarna kommer att synas för att ge bättre insikt i vart i klotet punkten befinner sig. Eftersom denna avbildning har ett modulus på 4π är den röda punkten i programmet inställd på att då den börjar hamna utanför 4π så ska den flyttas innanför klotet men speglas så att den behåller den sista flyttningen som användaren gjorde på systemet (dvs. $b \to -(4\pi - |b|) \cdot b/(|b|)$. Det finns även ett sfäriskt skal inritad i området $|b| = \pi$ som motsvarar ekvatorn för kvaternionerna i S^3, för att användarna skall få bättre koll på vilka kvaternioner som motsvarar den givna rotationen. Spinorrymden är väldigt lik Bivektorrymde i den mening att den har en sfär och ekvator men det finns istället fyra punkter, en för varje spinor som hade utgört en bas i fyra dimensioner. Dessa är utritade som platonska kroppar. Istället för skuggbilder är kropparna projicerade ned på $z = -10$ och $x = 10$ -planen för att ge en tydligare bild av deras lägen. Vektorrymden består av tre vektorer som representerar en bas i $\mathbb{R}^3$, som efter grafiken uppdateras kommer att börja rotera.

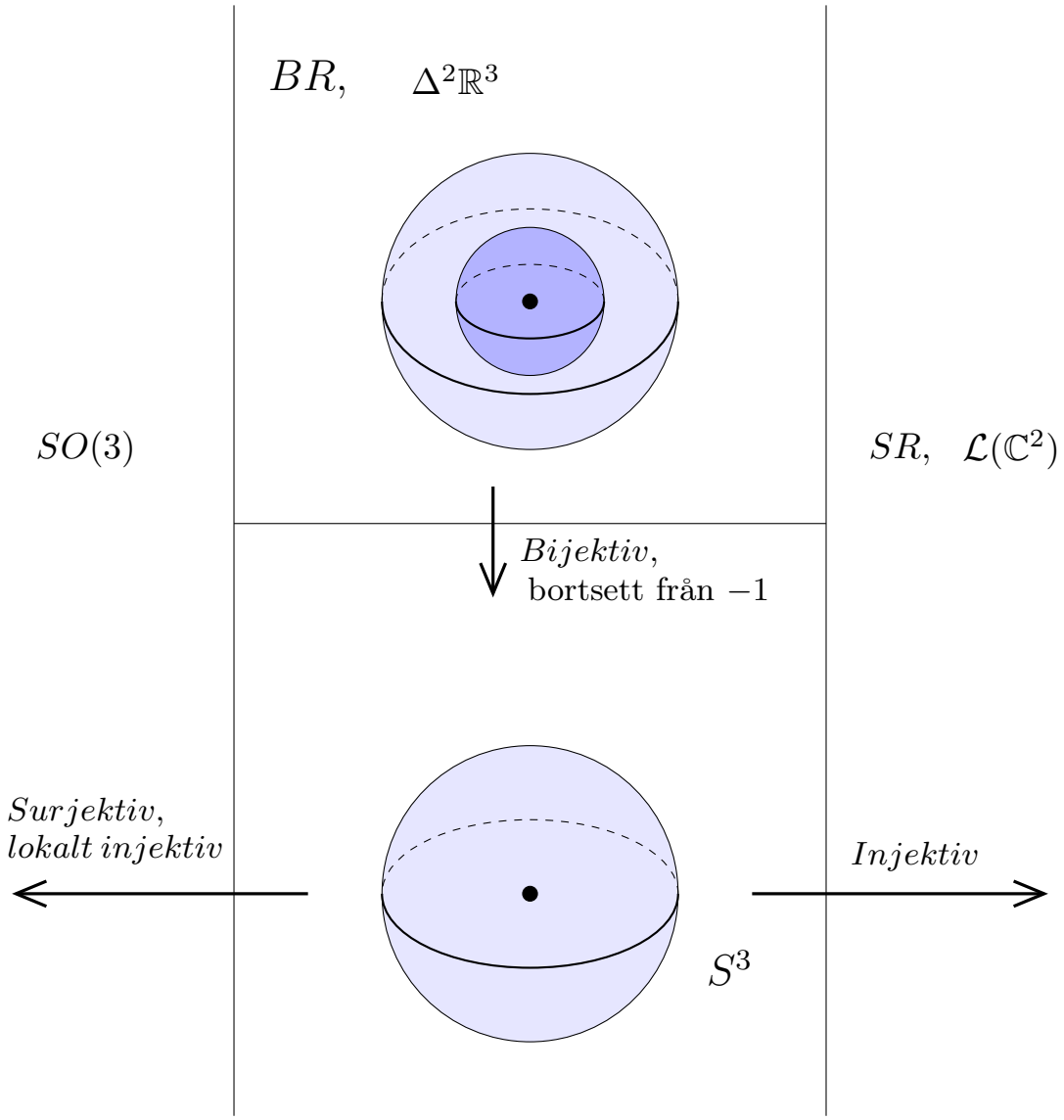
I samtliga program finns även interaktionsobjekt, objekt som en användare kan interagera med för att göra ändringar i grafiken. Dessa utgörs av planytor, tre olika plan som motsvarar projektioner/skärningar av $x - y, z - x$ och $y - z$-planen.

I programmet VR är dessa interaktionsplan projektioner av vektorrymden, där samtliga vektorer finns projicerade. Om interaktionspunkten kommer tillräckligt nära någon vektor kommer vektorn rotera så den pekar på interaktionspunkten. Samtliga vektorer kommer då att förflyttas och hela systemet ändras, vilket även kommer att flytta punkten i Bivektorrymden. Denna kommer att lägga sig i den punkt som motsvarar den axel och det vinkelutslag som hade roterat vektorrymden in i dess befintliga läge i en rotation. Detta läge fås genom användning av Cliffordspåret (29). Eftersom Bivektorrymden är 2-1 finns det egentligen fler sätt att göra dessa. Men om vi sätter kriterierna att flyttningarna ska vara kontinuerliga och att $b = (0, 0, 0)$ motsvarar vektorns grundsystem finns bara ett sätt.

I BR är interaktionsplanen skärningsplan av Bivektorrymden genom origo, parallellt med $x - y, z - x.y - z$ planen. De objekt man kan interagera med i dessa är röda punkter, som motsvarar klotpunkten i Bivektorrymden. Om interaktionspunkten kommer tillräckligt nära någon av planens klotpunkter kommer klotpunkten att ”dras” till denna punkt, vilket kommer att ändra punkten i Bivektorrymden. Detta kommer sedan uppdatera grafiken i VR på så sätt att systemet roterar in till motsvarande läge.

BR och VR visar $\mathbf{Z}_2$-minnet i den mening att när vektorsystemet roterat ett fullt varv och återställts till samma läge kommer inte klotpunkten att infinna sig i grundposition, utan det är först efter två varv som klotpunkten återställs. Om man till exempel enbart roterar i z-axeln ($x-y$ planet) kommer klotpunkten att befinna sig i $(0, 0, 2\pi)/(0, 0, -2\pi)$ (dessa motsvarar samma punkt), medan vektorsystemet då är i grundläget.

SR har samma interaktionsmoment som BR. Skillnaden är att spinorer, istället för vektorer, kommer förflyttas. Det som kommer visa sig här är att spinorerna inte befinner sig i grundläge då klotpunkten rört sig 2π från sin grundposition, istället kommer samtliga spinorer att finnas i de antipodala lägena till sitt grundläge (representationer av operatorer till spinorrymden har egenskapen att $\rho(-1) = -I$).



Figur 2: Avbildning mellan olika vektorrum

Litteraturförteckning

[1] Rosén, Andreas *Geometric Multivector Analysis*. Birkhäuser, Cham (forthcoming) 2019.

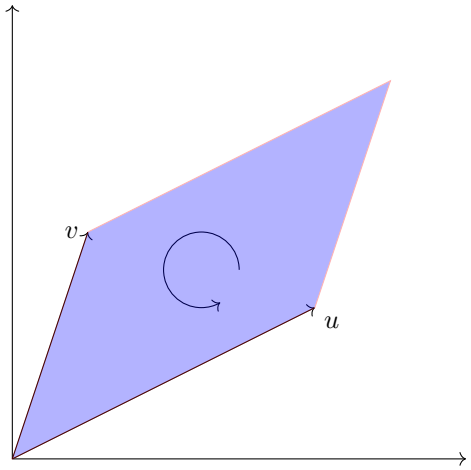
[2] Tapp, Kristopher. *Matrix Groups for Undergraduates*. AMERICAN MATHEMATICAL SOCIETY 2005.

[3] Sakurai, Jun John. *Modern Quantum Mechanics,2nd Edition*. Addison-Wesley Publishing Company 1994.

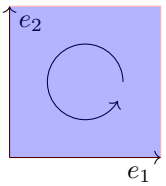
[4] Mathoma.
`Geometric Algebra in 3D - Fundamentals. Hämtad 2019-02-17 från` https://www.youtube.com/watch?v=ElLl6gzNbFE&list=PLpzmRsG7u_gqaTo_vEseQ7U8KFvtiJY4K&index=9 `Publicerades 27 december 2016.`

A Visualisering av yttre algebran

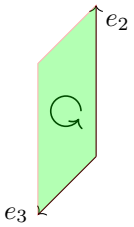
Figurer



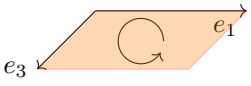
Figur 3: Visualisering av yttreprodukten av vektorer u och v



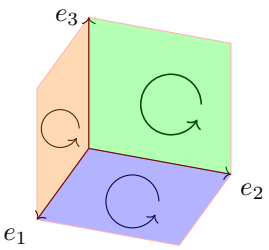
Figur 4: e_1e_2 i xy-plan



Figur 5: e_2e_3 i yz-plan



Figur 6: e_3e_1 i zx-plan



Figur 7: Yttreprodukten av tre linjärt oberoende vektorer.

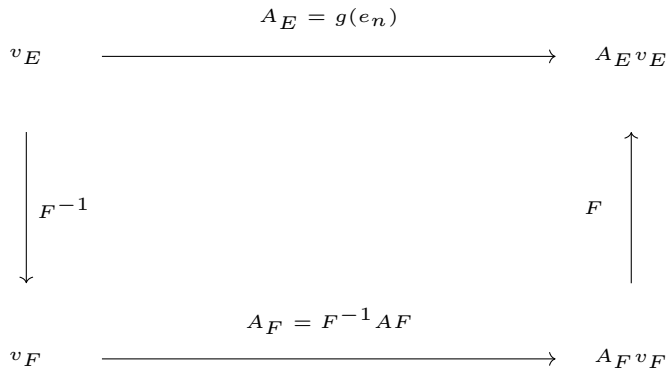
B Kompletterande Teori

B.1 Basbyten

I linjär algebra lär vi oss att till en linjär avbildning av ett vektorrum V $g : V \to V$ finns en matris A sådan att $g(v) = Av_E$, för en vektor $v \in V$, där $A = (g(e_1) \;\; g(e_2) \;\; ... \;\; g(e_n))$ och v_E är koordinaterna i basen $E = (e_1 \;\; e_2 \;\; ... \;\; e_n)$. Om vi nu låter $F = (f_1 \;\; f_2 \;\; ... \;\; f_n)$ vara en annan bas för V, sådan att $Fv_F = v_E$, gäller att $g(v) = A_F v_F$, där $A_F = (g(f_1) \;\; g(f_2) \;\; ... \;\; g(f_n))$. Vidare följer att vi kan skriva A_F som

$$A_F = F^{-1}AF$$

Detta förtydligas i nedanstående diagram.



Figur 8: Basbytesdiagram

B.2 Homomorfa matrisrepresentationer

Vi visar nedan att de representationer för kvaternionera som vi valde i 4.4.2 faktiskt är homomorfier.

Bevis. Låt $\phi : \mathbb{H} \to \mathcal{L}(\mathbb{C}^2)$,

Genom ϕ(a+bi+cj+dk)$=\begin{bmatrix} a+bi & c+di \\ -c+di & a-bi \end{bmatrix}$, där a,b,c,d $\in \mathbb{R}$.

För att visa att detta är en homomorfi måste vi visa att den bevarar multiplikation och addition. För att visa att den bevarar addition, låt

$$\phi((a+bi+cj+dk)+(e+fi+gj+hk)) =$$

$$\phi((a+e)+(b+f)i+(c+g)j+(d+h)k)=\begin{bmatrix} (a+e)+(b+f)i & (c+g)+(d+h)i \\ -(c+g)+(d+h)i & (a+e)-(b+f)i \end{bmatrix}=$$

$$= \begin{bmatrix} a+bi & c+di \\ -c+di & a-bi \end{bmatrix} + \begin{bmatrix} e+fi & g+hi \\ -g+hi & e-fi \end{bmatrix} = \phi(a+bi+cj+dk)+\phi(e+fi+gj+hk).$$

Vilket visar att den bevarar addition. För att visa att den bevarar multiplikation låt

$$\phi((a+bi+cj+dk)(e+fi+gj+hk)) =$$

$$\phi((ae-bf-cg+dh)+(af+be+ch-dg)i+(ag-bh+ce+df)j+(ah+bg-cf+de)k) =$$

$$\begin{bmatrix} (ae-bf-cg+dh)+(af+be+ch-dg)i & (ag-bh+ce+df)+(ah+bg-cf+de)i \\ -(ag-bh+ce+df)+(ah+bg-cf+de)i & (ae-bf-cg+dh)-(af+be+ch-dg)i \end{bmatrix}$$

$$= \begin{bmatrix} a+bi & c+di \\ -c+di & a-bi \end{bmatrix} \begin{bmatrix} e+fi & g+hi \\ -g+hi & e-fi \end{bmatrix} = \phi(a+bi+cj+dk)\phi(e+fi+gj+hk).$$

Vilket visar att den bevarar multiplikation. Alltså är ϕ en homomorfi.

□

Med samma argument som ovan kan man även se att

$\psi : \mathbb{H} \to \mathcal{L}(\mathbb{C}^2)$ är en homomorfi, där

$$\psi(a+bi+cj+dk) = \begin{bmatrix} a-ci & d-bi \\ -d-bi & a+ci \end{bmatrix}.$$

B.3 Rotationsmatris för godtycklig axel och vinkel i VR

Från linjär algebra lär vi oss att en rotation bestäms av vart den tar basvektorerna. Om vi vill lära oss hur rotationsmatrisen för en godtycklig axel och godtycklig vinkel räcker det alltså att vi kollar på basvektorerna. Detta kan vi göra med hjälp av kvaternioner. Vi har att

$R = (Re_1\ Re_2\ Re_3)$, där R är vår rotationsmatris och

$$\begin{aligned} Re_1 &= qe_1q^{-1}, \\ Re_2 &= qe_2q^{-1}, \\ Re_3 &= qe_3q^{-1}. \end{aligned}$$

Där $q = exp(\frac{-j\phi}{2})$, $j = ae_{32} + be_{13} + ce_{21}$, $a,\ b,\ c \in \mathbb{R}$ och $a^2+b^2+c^2 = 1$. Om vi räknar på detta får vi att

$qe_3q^{-1} = (\cos(\theta/2) - j\sin(\theta/2))e_3(\cos(\theta/2) + j\sin(\theta/2)) =$

$=(\cos(\theta/2)e_3 + (ae_{23}\sin(\theta/2) - be_{13}\sin(\theta/2) + ce_{12}\sin(\theta/2))e_3(\cos(\theta/2) + j\sin(\theta/2)) =$

$=(\cos(\theta/2)e_3 + a\sin(\theta/2)e_2 - b\sin(\theta/2)e_1 + c\sin(\theta/2)e_{123})(\cos(\theta/2) - a\sin(\theta/2)e_{23} + b\sin(\theta/2)e_{13} - c\sin(\theta/2)e_{12}) =$

$= \cos^2(\theta/2)e_3 + a\cos(\theta/2)\sin(\theta/2)e_2 - b\cos(\theta/2)\sin(\theta/2)e_1 - c\cos(\theta/2)\sin(\theta/2)e_{312} + a\cos(\theta/2)\sin(\theta/2)e_2 - a^2\sin^2(\theta/2)e_3 + ab\sin^2(\theta/2)e_{213} + ac\sin^2(\theta/2)e_1 - b\cos(\theta/2)\sin(\theta/2)e_1 + ab\sin^2(\theta/2)e_{123} - b^2\sin^2(\theta/2)e_3 + bc\sin^2(\theta/2)e_2 + c\cos(\theta/2)\sin(\theta/2)e_{123} + ac\sin^2(\theta/2)e_1 + bc\sin^2(\theta/2)e_2 + c^2\sin^2(\theta/2)e_3 =$

$= \cos^2(\theta/2)e_3 + \sin^2(\theta/2)(-a^2 - b^2 + c^2)e_3 + 2a\cos(\theta/2)\sin(\theta/2)e_2 - 2b\cos(\theta/2)\sin(\theta/2)e_1 + 2bc\sin^2(\theta/2)e_2 + 2ac\sin^2(\theta/2)e_1 =$

$=(2\text{ac}\sin^2(\theta/2) - b\sin(\theta))e_1 + (2bc\sin^2(\theta/2) + a\sin(\theta))e_2 + (\cos^2(\theta/2) - a^2\sin^2(\theta/2) - b^2\sin^2(\theta/2) + c^2\sin^2(\theta/2))e_3.$

Om man gör motsvarande beräkningar för e_1 och e_2 får man att

$qe_1q^{-1} = (\cos^2(\theta/2) + a^2\sin^2(\theta/2) - b^2\sin^2(\theta/2) - c^2\sin^2(\theta/2))e_1 + (2ab\sin^2(\theta/2) - c\sin(\theta))e_2 + (2ac\sin^2(\theta/2) + b\sin(\theta))e_3,$

$qe_2q^{-1} = (2ab\sin^2(\theta/2) + c\sin(\theta))e_1 + (\cos^2(\theta/2) - a^2\sin^2(\theta/2) + b^2\sin^2(\theta/2) - c^2\sin^2(\theta/2))e_2 + (2b\sin^2(\theta/2) - a\sin(\theta))e_3.$

Vi får då att

$$R = \begin{bmatrix} \cos^2(\theta/2)+\sin^2(\theta/2)(a^2-b^2-c^2) & 2ab\sin^2(\theta/2)-c\sin(\theta) & 2ac\sin^2(\theta/2)+b\sin(\theta) \\ 2ab\sin^2(\theta/2)+c\sin(\theta) & cos^2(\theta/2)+\sin^2(\theta/2)(-a^2+b^2-c^2) & 2b\sin^2(\theta/2)-a\sin(\theta) \\ 2ac\sin^2(\theta/2)-b\sin(\theta) & 2bc\sin^2(\theta/2)+a\sin(\theta) & \cos^2(\theta/2)+\sin^2(\theta/2)(-a^2-b^2+c^2) \end{bmatrix}$$

B.4 En rotation i $\mathbb{R}^3$ är alltid kring en axel

Bevis. Detta påstående innebär att för en allmän rotation kring origo finns alltid en linje (rotationsaxel) som inte har förflyttat sig med avseende på grundläget. Detta är bevisat om man kan visa att samtliga element $R \in SO(3)$ har en egenvektor v för vilken $Rv = v$. Detta är identiskt med att påstå att samtliga matriser i $SO(3)$ har ett egenvärde $\lambda = 1$, eller att $det(R-I) = 0$. För att visa detta använder vi att eftersom $R^\top = R^{-1}$ gäller $R-I = R-RR^\top = R\cdot(I-R^\top) = R\cdot(I-R)^\top$ vi har även för rotationsmatriser att $det(R) = 1$, samt att för godtyckliga A, B är $det(AB) = det(A)\cdot det(B)$ och $det(A^\top) = det(A)$. Med dessa beräknas

$det(R-I) = det(R\cdot(I-R)^\top) = det(R)\cdot det(I-R) = det((-1)\cdot(R-I)) = (-1)^3\cdot det(R-I) = -det(R-I).$

Eftersom $det(R-I) = -det(R-I)$ måste $det(R-I) = 0$. □

C KOD

C.1 VR

```
clf;clear all;
%introduktion
%{
%Detta program 'ar skapat i samband med ett kandidatprojekt,
%och 'ar menat att vara en del i ett paket av tre
%(VR,BR och SR, d'ar VR står f'or vektorrymden,
% BR f'or bivektorrymden och SR f'or spinorrymden).

%f'or att kunna k'ora BR och SR kr'avs quaternion.m
%SR kr'aver 'aven platonic_solid.m

%Vardera program skall gå att anv'anda utan tillgång till de andra, och går
%att anv'anda utan djupare kunskap i matematiken om bara f'or n'ojes skull.

%Denna inledning 'ar skriven f'or att ge insikt i vad de olika
%programmen g'or, vad de 'ar till f'or, hur man anv'ander dem, och hur de 'ar
%uppbyggda.
%Den 'ar skriven f'or att f'ors'oka så långt det går enhetligt
%beskriva samtliga tre program samtidigt, så
%att det inte finns behov av att l'asa inledningen i någon av de andra.

%Mer specifika detaljer relaterade till varje kods uppbyggnad; vad de olika
%momenten g'or och varf'or de skrivs på det s'att de g'ors skall skrivas n'ara
%tillh'orande kodstycke.

%Kunskaper kring programmering tenderar att variera, så jag har
%f'oredragit att ta upp "f'or många detaljer" 'an f'or få i texten, så koden
%skall vara enkel att f'orstå 'aven utan tidigare kunskaper.
%Jag 'ar sj'alv inte mycket av en kodare, och genom att beskriva så
%mycket av logiken som m'ojligt 'ar det f'orhoppningsvist enklare f'or en mer
%van att g'ora f'or'andringar denne tycker beh'ovs.


%BR och VR 'ar en visuell demonstration av hur kvaternionerna
%dubbelt t'acker gruppen SO(3), som beskriver rotationer i 3D.
%SR 'ar en visualisering av spinorer, som har egenskapen att de inte
%återst'allts efter 360 graders, utan 720 graders rotation
%(specifikt 'ar att f'or en spinor-representation p 'ar p(-1)=-I)

%Programmens strukturer 'ar på många s'att lika;
%Den f'orsta koden i varje program har jag satt ut parametrar som 'ar ditsatta
%så att anv'andaren skall kunna justera efter datorprestanda och personliga preferenser
%sedan de matematiska/logiska objekten,
```

```
%Darefter instantieras diverse geometriska objekt satt i sina
%"grundtillstånd".

%I vardera program finns tre plan ned på vilka ett rum projiceras
%(I BR och SR projiceras "Bivektorrymden", i VR projiceras "
    Vektorrymden").
%Dessutom kommer i vardera plan en punkt instantieras
%(en blå stj'arna ('*')) som i samtliga texter kommer kallas "
    interaktionspunkten",
%f'or att s'arskilja den från punkten i BR, som kommer kallas "
    klotpunkten".
%Planen kommer kallas "interaktionsplanen". Planen kommer ofta
    beskrivas
%som att de "tillh'or" det rum som projicerades ned på det
%(exempelvis h'ar i VR kan texten "vektorrummets plan" f'orekomma).
%Bland de objekt som finns visuellt representerade i planen finns
%komponenter som kan interagera med interaktionspunkter,
%vilka kommer kallas 'interaktionsobjekt'.
%interaktionspunkten går att flytta, och
%genom att komma tillr'ackligt n'ara ett interaktionsobjekt
%kommer det att flytta sig så den hamnar så n'ara
%interaktionspunkten den kan.
%interaktionsplanen 'ar alltid x-y, z-x, och y-z plan.
%Det 'ar v'art att anm'arka att jag valt att kalla ett plan z-x.

%I samtliga program finns ett "vektorrum" och ett "bivektorrum"
%vektorrummets visuella komponenter 'ar tre stycken vektorer,
%som skall representera ett koordinatystem
%(x-axel, y-axel och en z-axel) och i en bredare bem'arkelse, vektorer i
%tre dimensioner.

%Bivektorrummet 'ar ett klot med radie 2*pi
%som 'ar en visuell representation av enhetskvaterniornerna, vilka utg'or
%en tredimensionell sf'ar inb'addad i fyra dimensioner, och
%om b=[b1 b2 b3] 'ar en punkt i B(0,2*pi) kan den relateras till en
    specifik
%kvaternion via b->exp((b1*i+b2*j+b3*k)/2)
%Klotets fr'amsta komponent 'ar en punkt (kallad klotpunkt, kommer ibland
     i kod understryka
%att den 'ar r'od, f'or att s'arskilja den från interaktionspunkten, och n'a
    r jag diskuterar
%den som en kvaternion eller vektor kommer jag kalla den b)
%bollen har f'or 'ovrigt ett modulus på 4*pi (exp((4*pi*j)/2)=1), så n'ar
    b kommer "'over" 4*pi
%kommer den f'orflyttas till ett motsvarande l'age på andra sidan klotet.
%f'or att ge en b'attre bild av var i klotet punkten befinner sig har
%"skuggbilder satts ut, vilka beskriver b's lprojektioner på B's axlar.

%I VR 'ar det vektorrummet som 'ar projicerat på interaktionsplanen.
%vektorrummets interaktionsobjekt 'ar de olika vektorernas pilspetsar;
%n'ar interaktionspunkten kommer n'ara pilspetsen kommer dess vektor
%roteras in mot interaktionspunkten,
%varefter hela koordninatsystemet kommer rotera efter samma axel och
%med samma vinkelutslag.
%Varje system av koordinataxlar (x',y',z'), motsvarar en
    rotationsmatris:
```

```
91 %[x' y' z'] (eftersom x' y' z' 'ar ortonormerade, vi har
92 %R[e1 e2 e3]=[R(e1) R(e2) R(e3)]=[x' y' z']
93 %Ifrån denna rotationsmatris kan man få ut en bivektor
94 %genom att betrakta matrisens "clifford trace", vilket vi kan
95 %relatera till en specifik axel och en specifik vinkel, vilket
96 %då 'ar en b-vektor som motsvarar klotpunktens nya l'age i bivektorrymden
       .
97
98 %Det som tydligt m'arks 'ar att 'aven då rotationssystemet roterat 360
       grader
99 %kommer inte b-vektorn ha återgått till [0 0 0], utan det 'ar f'orst
       efter
100 %två rotationer som systemet har återst'allts helt.
101
102
103 %I BR 'ar det bivektorrummet som 'ar projicerat på interaktionsplanen,
       och
104 %interaktionsobjektet 'ar klotpunkten. Då interaktionspunkten kommer n'a
       ra
105 %klotpunkten flyttas klotpunkten till interaktionspunktens l'age.
106 %I varje interaktionsplan finns 'aven en cirkel utritad, som motsvarar
107 %randen av BR.
108 %klotpunkten motsvarar ('ar) en bivektor, vilket vi kan relatera till en
       enhets-
109 %kvaternion q=exp(b/2), d'ar vi nu kan rotera koordinatsystemet
110 %med qvq^-1. Mer detaljrik f'orklaring finns i projektrapporten.
111 %I koden i sig anv'ands ett befintligt program som fanns i quaternion.m,
112 %som jag ej vet hur den 'ar uppbyggd.
113
114 %I SR har vi 'aven spinorrummet, vilket likt BR 'ar ett klot av radie 2*
       pi,
115 %vari det ligger fyra stycken punkter (återigen 'stj'arnor', (*))
116 %som motsvarar spinorer.
117 %Spinorer beskrivs ofta som vektorer i C^2, d'ar det finns en
118 %kvaternionrepresentation p(q)=[z w;-w' z'], som 'verkar' på
119 %spinorerna. I SR går vi 'over till R^4, oh eftersom p bevarar l'angder
120 %kommer "enhetsspinorer" befinna sig i sf'aren, så vi representerar
121 %spinorerna i någon mening som om de vore "kvaternioner".
122 %I SR 'ar det återigen BR som projiceras på interaktionsplanen, med
123 %samma interaktionssystem som i BR, och n'ar bollens r'orelse resulterar
124 %i f'or'andringar av kvaternionens l'age genom det p(q) som beskrevs ovan.
125 %hur kvaternionen roterar vektorrummet 'ar också med
126
127 %f'or att hj'alpa visualiseringen av spinorernas l'agen finns 'aven
128 %projektionsplan som beskriver dess l'agen i xy och zx planen, satta i
129 %deras klot, på nivåerna z=-10 respektive x=10,
130 %Det går 'aven att slå på "skuggbilder" som fungerar på samma s'att som
131 %klotets skuggor, men detta resulterar i 16 punkter i klotet, så det 'ar
132 %I grundl'aget satt till 0, men kan s'attas på med en "shadeon"-konstant
133
134
135 %samtliga interaktioner mellan program och anv'andare
136 %hanteras inom en while loop, som alltid 'ar satt till 'true'.
137 %interaktionspunkterna 'ar "impoint"-objekt (eller i 2019: drawpoint).
138 %Som instantieras med planet det befinner sig i.
139 %Dessa kallas på i loopen med en "get"
```

```
%(F'or den ovane: get/getters och set/setters 'ar
%i programmering typiska funktioner att ge klasser. En "get" låter
%anv'andaren l'asa av variabler/h'amta information som ett objekt har
%(i detta fall har vi en impoint, vars variabel vi plockar 'ar "Position
    "
%Den kan instantieras via h.impoint();, och positionen fås då
%via h.getPosition.))
%interaktionspunktens l'age sparas, och en serie if-satser kollar om
%den nuvarande punkten 'ar n'ara något interaktionsobjekt.
%pointdistance garanterar att de interaktionspunkter som inte vidr'ors
%ej kan interagera med sitt plans interaktionsobjekt, vilket hade gjort
%koden svårnavigerad.

%Det finns en del kriterier utsatta f'or att se till att så få "on'odiga"
%loopar g'ors som m'ojligt f'or att snabba upp programmetn som i nul'aget
%'ar r'att långsamma.
%Om interaktionspunkten kommer n'ara ett objekt, ber'aknar programmet
    vilka operationer
%som skall utf'oras.
%d'arefter har varje programs while loop en serie set-funktioner (en "
    set"/setter
%låter anv'andaren ans'atta ett nytt v'arde i ett objekt,
% exempelvis: set(u,'XData',b(1),'YData',b(2),'ZData',b(3)) s'atter ut
%klotpunkten i dess nya l'age i BR.) vilka uppdaterar alla grafiska
    objekt.
%}

%beskrivning av konstanter som kan vara v'art att modifiera,
%om tex kod går långsamt
%{

%1. 'pauselength' best'ammer, i sekunder, den tid det skall ta mellan
    varje
%iteration av att l'asa av vektorpunktens l'age och positionera
    vektorsystemet och
%klotpunkten efter det.

%2. 'dist' beskriver från vilket avstånd interaktionspunkten skall ha
%från pilspetsarna tills de b'orjar flytta på sig

%3. Pointdistance uts'atter ett krav
%som begr'ansar antalet processer som g'ors i while loopen, genom att
%kontrollera vilket avstånd den nuvarande interaktionspunkten har
%relativt den tidigare iterationspunkten
%dvs om man s'atter pointdistance 0.01 måste interaktionspunkten ha f'o
    rflyttats
%0.01 enheter från det l'age den var sedan senaste grafikuppdatering,
    innan
%programmet kontrollerar huruvida klotpunkten skall flyttas relativt
%interaktionspunktens nuvarande l'age.

%Den g'or 'aven så att interaktionspunkterna
%inte påverkar systemet om inte anv'andaren r'or vid dem. Om man
%s'atter pointdistance till noll finns det ingen f'ors'akring till detta.

%(Jag tror detta 'ar pga att impoint, som genererar
```

```
    interaktionspunkterna,
%verkar flytta sig sj'alv, om 'an v'aldigt lite, i sin egen uppdatering.
%(jag 'ar os'aker på dessa detaljer, kan ju också tex vara relaterat till
%grafikens uppdateringar).)

%startvec best'ammer interaktionspunktens startl'age då programmet k'ors.

%}

pauselength=0.05;
dist=0.4;
startvec=[0 0];
pointdistance=0.05;

%x,y,z 'ar namnen på axlarna hos vektorsystemet
%p,p2 sparar datan hos impoint i while-loopen
%dist kvadreras f'or att f'orenkla matematiken i programmet
%j2,i,theta2 anv'ands f'or att kolla kriterier om vilken sida
%ekvatorn klotpunkten skall befinna sig i bivektorrymden


x=[1;0;0];
y=[0;1;0];
z=[0;0;1];
p=startvec;
p2=startvec;
dist=dist^2;

j2=[0 0 0];
i=0;
theta2=0;

%klotet som beskriver de enhetskvaternioner som verkar
%på vektorrummet,
%{
%en punkt inom klotet (b) relateras till en "ren kvaternion"
%b=[b1 b2 b3]->b1*i+b2*j+b3*k som relateras till en
%enhetskvaternion q via q=exp(b/2).
%}

figure(1)

set(gcf, 'Position', [100, 100, 1000, 500])

subplot(1,2,1);

[m n o]=sphere; %total sphere bdry => q=-1
surf(2*pi*m,2*pi*n,2*pi*o,'facecolor','blue', 'facealpha',.2,'EdgeAlpha
    ',0);
hold on
surf(pi*m,pi*n,pi*o,'facecolor','green', 'facealpha',.4,'EdgeAlpha',0);
hold on
phi=linspace(0,2*pi);

xlabel('x')
ylabel('y')
```

```
242 zlabel('z')
243 plot(2*pi*cos(phi),2*pi*sin(phi),'black') % circle for visualisation
244
245 plot3([-10 10],[0 0], [0 0],'black',[0 0],[-10 10], [0 0],'black',[0
    0],[0 0], [-10 10],'black')
246
247 %{
248 %b representerat inom klotet som en punkt, bem'arkt som en stj'arna ('*')
    ,
249 %samt tre "skuggbilder", projektioner av b på x,y och z axlarna,
250 %i en svagare gråskala, f'or att få b'attre 'oversikt i var i det
251 %tredimensionella rummet b befinner sig
252 %(visualiseringen av bollens l'age står sist i koden, så att om
    punkterna
253 %overlappar, kommer den faktiska bollens l'age vara den som syns)
254 %}
255
256 uxyz=surf([0.35*m;0.35*m;0.35*m],[0.35*n;0.35*n;0.35*n],[0.35*o;0.35*o
    ;0.35*o],'facecolor','red', 'facealpha',0.25,'EdgeAlpha',0);
257
258 u=surf(0.35*m,0.35*n,0.35*o,'facecolor','red', 'facealpha',1,'EdgeAlpha
    ',0);
259
260 grid on
261 axis([-7 7 -7 7 -7 7])
262 axis equal
263
264 title('Bivektorrummet BR','FontSize',14)
265
266 subplot(1,2,2);
267
268 %tre vektorer visualiseras som motsvarar koordinater f'or ett
    tredimensionellt
269 %vektorrum, interaktivt kan roteras av anv'andaren.
270 %aven ett mindre vektorsystem, som visar grundpositionen.
271 %{
272 %dess rotationer kommer relateras till den bivektor b=-phi*j, som genom
     att
273 %verka på grundsystemet (via exp(b/2)v(exp(-b/2) skulle ha roterat
    systemet till dess nul'age.
274 %}
275 vecx=quiver3([0 -0.5],[0 -0.5],[0 -1.2],[x(1) 0.5],[x(2) 0],[x(3) 0],'b
    ','Autoscale','off'); hold on; %generate a rotateable x-y-z
    coordinate system
276 vecy=quiver3([0 -0.5],[0 -0.5],[0 -1.2],[y(1) 0],[y(2) 0.5],[y(3) 0],'r
    ','Autoscale','off');
277 vecz=quiver3([0 -0.5],[0 -0.5],[0 -1.2],[z(1) 0],[z(2) 0],[z(3) 0.5],'g
    ','Autoscale','off');
278
279 xlim([-1.2 1.2])
280 ylim([-1.2 1.2])
281 zlim([-1.2 1.2])
282 title('Vektorrummet VR','FontSize',14);
283
284 pbaspect([1 1 1]);
285
```

```
286
287
288 figure(2);
289
290 set(gcf, 'Position', [100, 100, 600, 200])
291
292 vx=quiver([-2.5 0 2.5],[0 0 0],[x(1),x(3),x(2)],[x(2),x(1),x(3)],'b','
    AutoScale','off'); hold on;%,'b');hold on
293 vy=quiver([-2.5 0 2.5],[0 0 0],[y(1),y(3),y(2)],[y(2),y(1),y(3)],'r','
    AutoScale','off');
294 vz=quiver([-2.5 0 2.5],[0 0 0],[z(1),z(3),z(2)],[z(2),z(1),z(3)],'g','
    AutoScale','off');
295
296 axis([-5 5 -1.5 1.5]);
297 pbaspect([10 3 1]);
298
299
300 txtxy = 'x-y plan';
301 txtzx = 'z-x plan';
302 txtyz = 'y-z plan';
303 text(-3,-1.1,txtxy,'FontSize',12);
304 text(-0.5,-1.1,txtzx,'FontSize',12);
305 text(2,-1.1,txtyz,'FontSize',12);
306 title('interaktiva ytor','FontSize',12);
307
308 h=impoint(gca,startvec);
309 while(true)
310     pause(pauselength)
311     %p sparar nya interaktionsl'aget
312     p=h.getPosition;
313     %om vi inte r'ort oss ett litet avstånd skall ingenting ske
314     if(pointdistance<=norm(p-p2))
315     if(0.1<=norm(p)&&0.1<=norm(p+[2.5 0])&&0.1<=norm(p-[2.5 0]))
316         %kollar om vi hamnat n'ara en specifik vektor
317     if((p(1)-x(1)+2.5)^2+(p(2)-x(2))^2<=dist&&0.1<=x(1)^2+x(2)^2)
318
319         %vi anv'ander det nya l'aget samt vektorn f'or att
320         %bilda en rotationsmatris, som vi anv'ander f'or att rotera
                samtliga
321         %objekt
322          p(1)=p(1)+2.5;
323
324          q=[x(1) x(2)];
325
326          p=p/norm(p);
327          q=q/norm(q);
328
329          %alfa,beta best'ammer rotationsmatrisen
330          alfa=p(1)*q(1)+p(2)*q(2);
331          beta=p(1)*q(2)-p(2)*q(1);
332
333
334          x1=alfa*x(1)+beta*x(2);
335          x2=alfa*x(2)-beta*x(1);
336          x(1)=x1;x(2)=x2;
337
```

```
338
339             y1=alfa*y(1)+beta*y(2);
340             y2=alfa*y(2)-beta*y(1);
341             y(1)=y1;y(2)=y2;
342
343             z1=alfa*z(1)+beta*z(2);
344             z2=alfa*z(2)-beta*z(1);
345             z(1)=z1;z(2)=z2;
346
347         elseif((p(1)-y(1)+2.5)^2+(p(2)-y(2))^2<=dist&&0.1<=y(1)^2+y(2)^2)
348
349             p(1)=p(1)+2.5;
350
351             q=[y(1) y(2)];
352
353             p=p/norm(p);
354             q=q/norm(q);
355
356
357             alfa=p(1)*q(1)+p(2)*q(2);
358             beta=p(1)*q(2)-p(2)*q(1);
359
360             x1=alfa*x(1)+beta*x(2);
361             x2=alfa*x(2)-beta*x(1);
362             x(1)=x1;x(2)=x2;
363             y1=alfa*y(1)+beta*y(2);
364             y2=alfa*y(2)-beta*y(1);
365             y(1)=y1;y(2)=y2;
366
367             z1=alfa*z(1)+beta*z(2);
368             z2=alfa*z(2)-beta*z(1);
369             z(1)=z1;z(2)=z2;
370             %x=x/norm(x);y=y/norm(y);z=z/norm(z);
371
372         elseif((p(1)-z(1)+2.5)^2+(p(2)-z(2))^2<=dist&&0.1<=z(1)^2+z(2)^2)
373
374             p(1)=p(1)+2.5;
375             q=[z(1) z(2)];
376
377             p=p/norm(p);
378             q=q/norm(q);
379
380
381             alfa=p(1)*q(1)+p(2)*q(2);
382             beta=p(1)*q(2)-p(2)*q(1);
383
384             x1=alfa*x(1)+beta*x(2);
385             x2=alfa*x(2)-beta*x(1);
386             x(1)=x1;x(2)=x2;
387
388             y1=alfa*y(1)+beta*y(2);
389             y2=alfa*y(2)-beta*y(1);
390             y(1)=y1;y(2)=y2;
391
392             z1=alfa*z(1)+beta*z(2);
393             z2=alfa*z(2)-beta*z(1);
```

```
394             z(1)=z1;z(2)=z2;
395
396
397             %x=x/norm(x);y=y/norm(y);z=z/norm(z);
398
399         elseif((p(1)-x(3))^2+(p(2)-x(1))^2<=dist&&0.1<=x(3)^2+x(1)^2)
400             q=[x(3)  x(1)];
401              p=p/norm(p);
402              q=q/norm(q);
403
404
405              alfa=p(1)*q(1)+p(2)*q(2);
406              beta=p(1)*q(2)-p(2)*q(1);
407
408             x1=alfa*x(3)+beta*x(1);
409             x2=alfa*x(1)-beta*x(3);
410             x(3)=x1;x(1)=x2;
411
412             y1=alfa*y(3)+beta*y(1);
413             y2=alfa*y(1)-beta*y(3);
414             y(3)=y1;y(1)=y2;
415
416             z1=alfa*z(3)+beta*z(1);
417             z2=alfa*z(1)-beta*z(3);
418             z(3)=z1;z(1)=z2;
419
420         elseif((p(1)-y(3))^2+(p(2)-y(1))^2<=dist&&0.1<=y(3)^2+y(1)^2)
421             q=[y(3)  y(1)];
422             p=p/norm(p);
423             q=q/norm(q);
424
425
426              alfa=p(1)*q(1)+p(2)*q(2);
427              beta=p(1)*q(2)-p(2)*q(1);
428
429             x1=alfa*x(3)+beta*x(1);
430             x2=alfa*x(1)-beta*x(3);
431             x(3)=x1;x(1)=x2;
432
433             y1=alfa*y(3)+beta*y(1);
434             y2=alfa*y(1)-beta*y(3);
435             y(3)=y1;y(1)=y2;
436
437
438             z1=alfa*z(3)+beta*z(1);
439             z2=alfa*z(1)-beta*z(3);
440             z(3)=z1;z(1)=z2;
441
442         elseif((p(1)-z(3))^2+(p(2)-z(1))^2<=dist&&0.1<=z(3)^2+z(1)^2)
443
444             q=[z(3)  z(1)];
445             p=p/norm(p);
446             q=q/norm(q);
447
448
449             alfa=p(1)*q(1)+p(2)*q(2);
```

```
450         beta=p(1)*q(2)-p(2)*q(1);
451
452         x1=alfa*x(3)+beta*x(1);
453         x2=alfa*x(1)-beta*x(3);
454         x(3)=x1;x(1)=x2;
455
456         y1=alfa*y(3)+beta*y(1);
457         y2=alfa*y(1)-beta*y(3);
458         y(3)=y1;y(1)=y2;
459
460         z1=alfa*z(3)+beta*z(1);
461         z2=alfa*z(1)-beta*z(3);
462         z(3)=z1;z(1)=z2;
463
464     elseif((p(1)-x(2)-2.5)^2+(p(2)-x(3))^2<=dist&&0.1<=x(2)^2+x(3)^2)
465         p(1)=p(1)-2.5;
466         q=[x(2) x(3)];
467         p=p/norm(p);
468         q=q/norm(q);
469
470
471         alfa=p(1)*q(1)+p(2)*q(2);
472         beta=p(1)*q(2)-p(2)*q(1);
473
474          x1=alfa*x(2)+beta*x(3);
475          x2=alfa*x(3)-beta*x(2);
476          x(2)=x1;x(3)=x2;
477
478          y1=alfa*y(2)+beta*y(3);
479          y2=alfa*y(3)-beta*y(2);
480          y(2)=y1;y(3)=y2;
481
482          z1=alfa*z(2)+beta*z(3);
483          z2=alfa*z(3)-beta*z(2);
484          z(2)=z1;z(3)=z2;
485
486     elseif((p(1)-y(2)-2.5)^2+(p(2)-y(3))^2<=dist&&0.1<=y(2)^2+y(3)^2)
487         p(1)=p(1)-2.5;
488         q=[y(2) y(3)];
489         p=p/norm(p);
490         q=q/norm(q);
491
492
493          alfa=p(1)*q(1)+p(2)*q(2);
494          beta=p(1)*q(2)-p(2)*q(1);
495
496          x1=alfa*x(2)+beta*x(3);
497          x2=alfa*x(3)-beta*x(2);
498          x(2)=x1;x(3)=x2;
499
500          y1=alfa*y(2)+beta*y(3);
501          y2=alfa*y(3)-beta*y(2);
502          y(2)=y1;y(3)=y2;
503
504          z1=alfa*z(2)+beta*z(3);
505          z2=alfa*z(3)-beta*z(2);
```

```
506             z(2)=z1;z(3)=z2;
507
508         elseif((p(1)-z(2)-2.5)^2+(p(2)-z(3))^2<=dist&&0.1<=z(2)^2+z(3)^2)
509             p(1)=p(1)-2.5;
510             q=[z(2) z(3)];
511             p=p/norm(p);
512             q=q/norm(q);
513
514
515             alfa=p(1)*q(1)+p(2)*q(2);
516             beta=p(1)*q(2)-p(2)*q(1);
517
518             z1=alfa*z(2)+beta*z(3);
519             z2=alfa*z(3)-beta*z(2);
520             z(2)=z1;z(3)=z2;
521
522
523             y1=alfa*y(2)+beta*y(3);
524             y2=alfa*y(3)-beta*y(2);
525             y(2)=y1;y(3)=y2;
526
527             x1=alfa*x(2)+beta*x(3);
528             x2=alfa*x(3)-beta*x(2);
529             x(2)=x1;x(3)=x2;
530         end
531
532         %{
533         %b-bollens l'age fås ut via Clifford Tracet, d'ar
534         %CT(R)=2*(cos(theta/2)+j*sin(theta/2))+1
535         %R 'ar en rotationsmatris, som kan fås ut av att givet en
536         %konfiguration {x',y',z'} 'ar R=[x' y' z']
537         %if satserna 'ar till f'or att l'osa problemet att cosinus inte 'ar
538         %surjektiv på [0,2pi].
539         %det man kan kolla 'ar att eftersom sin(theta/2) byter tecken i områ
                det
540         %theta=k*pi kommer den j-vektor som tas ut d'ar byta tecken i detta
                område
541         %och vi bryr oss enbart om området vid "ekvatorn" abs(b)=pi
542         %(att man får teckenskift vid 2pi 'ar inga problem, eftersom vi 'andå
543         %har en "spegling" i detta område. 0an 'ar också ok, d'ar har vi ocks
                å
544         %tecken'andringar.
545         %så n'ar vi flyttar mellan sidor av ekvatorn tar vi i+1, och
546         %om i 'ar j'amn eller udda best'ammer vilket belopp vi skall v'alja f'or
547         %theta.
548         %}
549         j=[y(3)-z(2) z(1)-x(3) x(2)-y(1)];
550
551         theta=real(acos((x(1)+y(2)+z(3)-1)/2));
552
553             if(not(j(1)==0&&j(2)==0&&j(3)==0))
554                 j=j/norm(j);
555             if(abs(theta2-pi)<=0.5&&not(sign(j(1))==sign(j2(1)))&&not(abs(j
                    (2)^2+j(3)^2-1)<=0.2))
556              i=mod(i+1,2);
557             elseif(abs(theta2-pi)<=0.5&&not(sign(j(2))==sign(j2(2)))&&not(
```

```
            abs(j(1)^2+j(3)^2-1)<=0.2))
558           i=mod(i+1,2);
559         elseif(abs(theta2-pi)<=0.5&&not(sign(j(3))==sign(j2(3)))&&not(
            abs(j(1)^2+j(2)^2-1)<=0.2))
560           i=mod(i+1,2);
561         end
562
563         else
564             j=j2;
565         end
566
567
568         if(i==0)
569             b=theta*j;
570             theta2=theta;
571
572         else
573             b=-(2*pi-theta)*j;
574             theta2=2*pi-theta;
575         end
576         %set funktionerna uppdaterar grafiken
577         set(vx,'udata',[x(1),x(3),x(2)],'vdata',[x(2),x(1),x(3)]);
578         set(vy,'udata',[y(1),y(3),y(2)],'vdata',[y(2),y(1),y(3)]);
579         set(vz,'udata',[z(1),z(3),z(2)],'vdata',[z(2),z(1),z(3)]);
580         set(vecx,'udata',[x(1) 0.5],'vdata',[x(2) 0],'wdata',[x(3) 0]);
581         set(vecy,'udata',[y(1) 0],'vdata',[y(2) 0.5],'wdata',[y(3) 0]);
582         set(vecz,'udata',[z(1) 0],'vdata',[z(2) 0],'wdata',[z(3) 0.5]);
583         set(uxyz,'XData',[b(1)+0.35*m;0.35*m;0.35*m],'YData',[0.35*n;b
            (2)+0.35*n;0.35*n],'ZData',[0.35*o;0.35*o;b(3)+0.35*o])
584
585         set(u,'XData',b(1)+m*0.35,'YData',b(2)+n*0.35,'ZData',b(3)+o
            *0.35)
586         j2=j;
587         theta2=theta;
588
589     end
590     end
591     p2=p;
592 end
```

C.2 BR

```
1  %{
2  %Detta program 'ar skapat i samband med ett kandidatprojekt,
3  %och 'ar menat att vara en del i ett paket av tre
4  %(VR,BR och SR(detta program), d'ar VR står f'or vektorrymden,
5  % BR f'or bivektorrymden och SR f'or spinorrymden).
6
7  %F'or BR och SR kr'avs att man har quaternion.m f'or att programmen skall
8  %kunna k'ora. SR kr'aver 'aven att platonic_solid.m finns.
9
10 %Vardera program skall gå att anv'anda utan tillgång till de andra, och
       går
11 %att anv'anda utan djupare kunskap i matematiken om bara f'or n'ojes skull
       .
12
```

```
%Denna inledning 'ar skriven f'or att ge insikt i vad de olika
%programmen g'or, vad de 'ar till f'or, hur man anv'ander dem, och hur de 'ar
%uppbyggda.
%Den 'ar skriven f'or att f'ors'oka så långt det går enhetligt
%beskriva samtliga tre program samtidigt, så
%att det inte finns behov av att l'asa inledningen i någon av de andra.

%Mer specifika detaljer relaterade till varje kods uppbyggnad; vad de olika
%momenten g'or och varf'or de skrivs på det s'att de g'ors skall skrivas n'ara
%tillh'orande kodstycke.

%Kunskaper kring programmering tenderar att variera, så jag har
%f'oredragit att ta upp "f'or många detaljer" 'an f'or få i texten, så koden
%skall vara enkel att f'orstå 'aven utan tidigare kunskaper.
%Jag 'ar sj'alv inte mycket av en kodare, och genom att beskriva så
%mycket av logiken som m'ojligt 'ar det f'orhoppningsvist enklare f'or en mer
%van att g'ora f'or'andringar denne tycker beh'ovs.


%BR och VR 'ar en visuell demonstration av hur kvaternionerna
%dubbelt t'acker gruppen SO(3), som beskriver rotationer i 3D.
%SR 'ar en visualisering av spinorer, som har egenskapen att de inte
%återst'allts efter 360 graders, utan 720 graders rotation
%speciellt 'ar v'al att f'or en spinor-representation p 'ar p(-1)=-I

%Programmens strukturer 'ar på många s'att lika;
%Den f'orsta koden i varje program har jag satt ut parametrar som 'ar ditsatta
%så att anv'andaren skall kunna justera efter datorprestanda och personliga preferenser
%sedan de matematiska/logiska objekten,
%D'arefter instantieras diverse geometriska objekt satt i sina
%"grundtillstånd".

%I vardera program finns tre plan ned på vilka ett rum projiceras
%(I BR och SR projiceras "Bivektorrymden", i VR projiceras "Vektorrymden").
%Dessutom kommer i vardera plan en punkt instantieras
%(en blå stj'arna ('*')) som i samtliga texter kommer kallas "interaktionspunkten",
%f'or att s'arskilja den från punkten i BR, som kommer kallas "klotpunkten".
%Planen kommer kallas "interaktionsplanen". Planen kommer ofta beskrivas
%som att de "tillh'or" det rum som projicerades ned på det
%(exempelvis h'ar i VR kan texten "vektorrummets plan" f'orekomma).
%Bland de objekt som finns visuellt representerade i planen finns
%komponenter som kan interagera med interaktionspunkter,
%vilka kommer kallas 'interaktionsobjekt'.
%interaktionspunkten går att flytta, och
%genom att komma tillr'ackligt n'ara ett interaktionsobjekt
```

```
58 %kommer det att flytta sig så den hamnar så n'ara
59 %interaktionspunkten den kan.
60 %interaktionsplanen 'ar alltid x-y, z-x, och y-z plan.
61 Det 'ar v'art att anm'arka att jag valt att kalla ett plan z-x.
62
63 %I samtliga rum f'orekommer en "vektor- och bivektorrymd"
64 %Vektorrymden visuella komponenter 'ar tre stycken vektorer,
65 %som skall representera ett koordinatystem
66 %(x-axel, y-axel och en z-axel) och i en bredare bem'arkelse, vektorer i
67 %tre dimensioner.
68
69 %Bivektorrymden 'ar klot med radie 2*pi
70 %som 'ar en visuell representation av enhetskvaterniornerna, vilka utg'or
71 %en tredimensionell sf'ar inb'addad i fyra dimensioner, och
72 %om b=[b1 b2 b3] 'ar en punkt i B(0,2*pi) kan den relateras till en
       specifik
73 %kvaternion via b->exp((b1*i+b2*j+b3*k)/2)
74 %Klotets fr'amsta komponent 'ar en punkt (kallad klotpunkt, kommer ibland
       i kod understryka
75 %att den 'ar r'od, f'or att s'arskilja den från interaktionspunkten, och n'a
       r jag diskuterar
76 %den som en kvaternion eller vektor kommer jag kalla den b)
77 %bollen har f'or 'ovrigt ett modulus på 4*pi (exp((4*pi*j)/2)=1), så n'ar
       b kommer "'over" 4*pi
78 %kommer den f'orflyttas till ett motsvarande l'age på andra sidan klotet.
79 %f'or att ge en b'attre bild av var i klotet punkten befinner sig har
80 %"skuggbilder satts ut, vilka beskriver b's lprojektioner på B's axlar.
81
82 %I VR 'ar det vektorrummet som 'ar projicerat på interaktionsplanen.
83 %vektorrummets interaktionsobjekt 'ar de olika vektorernas pilspetsar;
84 %n'ar interaktionspunkten kommer n'ara pilspetsen kommer dess vektor
85 %roteras in mot interaktionspunkten,
86 %varefter hela koordninatsystemet kommer rotera efter samma axel och
87 %med samma vinkelutslag.
88 %Varje system av koordinataxlar (x',y',z'), motsvarar en
       rotationsmatris:
89 %[x' y' z'] (eftersom x' y' z' 'ar ortonormerade, vi har
90 %R[e1 e2 e3]=[R(e1) R(e2) R(e3)]=[x' y' z']
91 %Ifrån denna rotationsmatris kan man få ut en bivektor
92 %genom att betrakta matrisens "clifford trace", vilket vi kan
93 %relatera till en specifik axel och en specifik vinkel, vilket
94 %då 'ar en b-vektor som motsvarar klotpunktens nya l'age i bivektorrymden
       .
95
96 %Det som tydligt m'arks 'ar att 'aven då rotationssystemet roterat 360
       grader
97 %kommer inte b-vektorn ha återgått till [0 0 0], utan det 'ar f'orst
       efter
98 %två rotationer som systemet har återst'allts helt.
99
100
101 %I BR 'ar det bivektorrummet som 'ar projicerat på interaktionsplanen,
       och
102 %interaktionsobjektet 'ar klotpunkten. Då interaktionspunkten kommer n'a
       ra
103 %klotpunkten flyttas klotpunkten till interaktionspunktens l'age.
```

```
104 %I varje interaktionsplan finns 'aven en cirkel utritad, som motsvarar
105 %randen av BR.
106 %klotpunkten motsvarar ('ar) ju en b-vektor, vilket vi kan relatera till
      en enhets-
107 %kvaternion q=exp(b/2), d'ar vi nu kan rotera koordinatsystemet
108 %med qvq^-1. Mer detaljrik f'orklaring finns i projektrapporten.
109 %I koden i sig anv'ands ett befintligt program som fanns i quaternion.m,
110 %som jag ej vet hur den 'ar uppbyggd.
111
112 %I SR har vi spinorrummet, vilket likt BR 'ar ett klot av radie 2*pi,
113 %vari det ligger fyra stycken punkter (återigen 'stj'arnor', (*))
114 %som motsvarar spinorer.
115 %Spinorer beskrivs ofta som vektorer i C^2, d'ar det finns en
116 %kvaternionrepresentation p(q)=[z w;-w' z'], som 'verkar' på
117 %spinorerna. I SR går vi 'over till R^4, oh eftersom p bevarar l'angder
118 %kommer "enhetsspinorer" befinna sig i sf'aren, så vi representerar
119 %spinorerna i någon mening som om de vore "kvaternioner".
120 %I SR 'ar det återigen BR som projiceras på interaktionsplanen, med
121 %samma interaktionssystem som i BR, och n'ar bollens r'orelse resulterar
122 %i f'or'andringar av kvaternionens l'age genom det p(q) som beskrevs ovan.
123
124 %f'or att hj'alpa visualiseringen av spinorernas l'agen finns 'aven
125 %projektionsplan som beskriver dess l'agen i xy och zx planen, satta i
126 %deras klot, på nivåerna z=-10 respektive x=10,
127
128 %samtliga interaktioner mellan program och anv'andare
129 %i varje kod hanteras inom en while loop, som alltid 'ar satt till 'true
      '.
130 %interaktionspunkterna 'ar "impoint"-objekt (eller i 2019: drawpoint).
131 %Som instantieras med planet det befinner sig i.
132 %Dessa kallas på i loopen med en "get"
133 %(F'or den ovane: get/getters och set/setters 'ar
134 %i programmering typiska funktioner att ge klasser. En "get" låter
135 %anv'andaren l'asa av variabler/h'amta information som ett objekt har
136 %(i detta fall har vi en impoint, vars variabel vi plockar 'ar "Position
      "
137 %Den kan instantieras via h.impoint();, och positionen fås då
138 %via h.getPosition.))
139 %interaktionspunktens l'age sparas, och en serie if-satser kollar om
140 %den nuvarande punkten 'ar n'ara något interaktionsobjekt.
141 %pointdistance garanterar att de interaktionspunkter som inte vidr'ors
142 %ej kan interagera med sitt plans interaktionsobjekt, vilket hade gjort
143 %koden svårnavigerad.
144
145 %Det finns en del kriterier utsatta f'or att se till att så få "on'odiga"
146 %loopar g'ors som m'ojligt f'or att snabba upp programmetn som i nul'aget
147 %'ar r'att långsamma.
148 %Om interaktionspunkten kommer n'ara ett objekt, ber'aknar programmet
      vilka operationer
149 %som skall utf'oras.
150 %d'arefter har varje programs while loop en serie set-funktioner (en "
      set"/setter
151 %låter anv'andaren ans'atta ett nytt v'arde i ett objekt,
152 % exempelvis: set(u,'XData',b(1),'YData',b(2),'ZData',b(3)) s'atter ut
153 %klotpunkten i dess nya l'age i BR.) vilka uppdaterar alla grafiska
      objekt.
```

```
154 %}
155
156 clear all; clf;
157
158
159 %nedan introduceras konstanter som anv'andaren kan v'alja att
160 %justera efter behov, beroende på dennes dators prestandakrav osv.
161 %{
162 %Detta program (BR) verkar vara mindre kr'avande 'an VR
163 %så det finns mer frihet i hur man vill ans'atta konstanterna h'ar, och
164 %jag har inga direkta rekommendationer, om anv'andaren 'ar intresserad
165 %b'or denne testa sj'alv.
166
167 %1. 'pauselength' best'ammer, i sekunder, den tid det skall ta mellan
      varje
168 %iteration av att l'asa av vektorpunktens l'age och positionera
      vektorsystemet och
169 %klotpunkten efter det.
170
171 %2. 'dist' beskriver från vilket avstånd de blå punkterna
172 %aktiverar pilarna så att de kan f'orflyttas
173
174 %3. Pointdistance uts'atter ett krav
175 %som begr'ansar antalet processer som g'ors i while loopen, genom att
176 %kontrollera vilket avstånd den nuvarande interaktionspunkten har
177 %relativt den tidigare iterationspunkten
178 %dvs om man s'atter pointdistance 0.01 måste interaktionspunkten ha f'o
      rflyttats
179 %0.01 enheter från det l'age den var sedan senaste grafikuppdatering,
      innan
180 %programmet kontrollerar huruvida klotpunkten skall flyttas relativt
181 %interaktionspunktens nuvarande l'age.
182
183 %Den g'or 'aven så att interaktionspunkterna
184 %inte påverkar systemet om inte anv'andaren r'or vid dem. Om man
185 %s'atter pointdistance till noll finns det ingen f'ors'akring till detta.
186
187 %(Jag tror detta 'ar pga att impoint, som genererar
      interaktionspunkterna,
188 %verkar flytta sig sj'alv, om 'an v'aldigt lite, i sin egen uppdatering.
189 %(jag 'ar os'aker på dessa detaljer, kan ju också tex vara relaterat till
190 %grafikens uppdateringar).)
191 %}
192 dist=1.5;
193 pauselength=0.05;
194 pointdistance=0.005;
195
196 %nedan instantieras de logiska/matematiska objekten
197
198 %b 'ar punkten i klotrymden som, som relaterar till kvaternionrummet
199 %via q=exp(b/2). H'ar relaterar den till rotationer med T(v)=qvq^(-1)
200 %(punkten kommer i texten kallas "klotpunkten" då dess visuella
      aspekter betraktas
201 % och b om jag skall beskriva den som ett matematiskt objekt)
202
203 b=[0 0 0];
```

```
204 x=[1,0,0];
205 y=[0,1,0];
206 z=[0,0,1];
207
208
209 %koden nedan ritar ut klotet i vilket klotpunkten (b) befinner sig
210
211 figure(1)
212
213 set(gcf, 'Position', [100, 100, 1000, 500])
214
215
216 subplot(1,2,1)
217
218 [m n o]=sphere; %total sphere bdry => q=-1
219 surf(2*pi*m,2*pi*n,2*pi*o,'facecolor','blue', 'facealpha',.3,'EdgeAlpha
    ',0);
220 hold on
221 surf(pi*m,pi*n,pi*o,'facecolor','green', 'facealpha',.5,'EdgeAlpha',0);
222 hold on
223 phi=linspace(0,2*pi);
224 xlabel('x')
225 ylabel('y')
226 zlabel('z')
227 title('Bivektorrymden BR','FontSize',14);
228
229 plot(2*pi*cos(phi),2*pi*sin(phi),'black') % circle for visualisation
230 hold on
231 plot3([-10 10],[0 0], [0 0],'black',[0 0],[-10 10], [0 0],'black',[0
    0],[0 0], [-10 10],'black')
232
233 %klotpunkten ritas ut,
234 %samt tre stycken "skuggbilder"; projektioner av b på klotrummets x,y
    och z axlar
235 %utritade i gråskala med svag ljusinst'allning, f'or att ge mer insikt
236 %i var inom klotrummet bollen befinner sig i.
237 %(bollen står sist i koden så att den inte 'overt'acks av skuggbilderna)
238
239 uxyz=plot3([0 0 0],[0 0 0],[0 0 0],'*','Color',[0.5 0.5 0.5]);
240 u=surf(0.35*m,0.35*n,0.35*o,'facecolor','red', 'facealpha',1,'EdgeAlpha
    ',0);
241
242 grid on
243 axis([-7 7 -7 7 -7 7])
244 axis equal
245 hold on
246
247 subplot(1,2,2);
248
249 %Koden nedan ritar ut vektorsystemet som roteras av
250 %den kvaternion (q) som beskrivs av klotpunkten via q=exp(b/2).
251 vec1=quiver3(0,0,0,x(1),x(2),x(3),'b'); hold on;
252 vec2=quiver3(0,0,0,y(1),y(2),y(3),'r');
253 vec3=quiver3(0,0,0,z(1),z(2),z(3),'g');
254
255 %ett mindre vektorsystem som visar vektorsystemets ursprungsposition
```

```
256 %(b=0, eller [x,y,z]=[1 0 0],[0 1 0], [0 0 1])
257 smallx=quiver3(-0.5,-0.5,-1,0.5,0,0,'b');
258 smally=quiver3(-0.5,-0.5,-1,0,0.5,0,'r');
259 smallz=quiver3(-0.5,-0.5,-1,0,0,0.5,'g');
260
261
262 xlim([-1.2 1.2])
263 ylim([-1.2 1.2])
264 zlim([-1.2 1.2])
265 title('Vektorrymden VR','FontSize',14);
266
267 %kod som 'projicerar' bollrymden på x-y,z-x och y-z planen,
268 %inom vilka tre separata punkter (h'ar kallade interaktionspunkter (de
      blå punkter)
269 %vilka låter dig manipulera klotpunktens (den r'oda) l'age.
270 %Om man f'orflyttar ett plans interaktionspunkt så att den hamnar
271 %tillr'ackligt n'ara klotpunkten (avståndet best'ams av 'dist') kommer
272 %klotpunkten att f'orflyttas till interaktionspunktens plats.
273
274 %cirkeln inritat i planet beskriver avståndet punkten har från
275 %klotets rand i detta plan, dess storlek uppdateras efter
276 %klotpunktens nya l'agen.
277
278 %(koden f'or samtliga plan 'ar i allm'anhet analoga)
279
280 %genom att g'ora subplotten som ett objekt, kan den deklareras
281 %till en "parent" f'or impointen, vilket tillåter mig att best'amma
282 %impointens l'age inuti koden
283 %xyz=subplot(3,4,[9 10]);
284
285 figure(2);
286 set(gcf, 'Position', [100, 100, 600, 200])
287
288 bxyz=plot([b(1)-20 b(3) b(2)+20],[b(2) b(1) b(3)],'red*');hold on
289
290 circz=plot(2*pi*cos(phi)-20,2*pi*sin(phi),'g');
291
292 circy=plot(2*pi*cos(phi),2*pi*sin(phi),'r');
293
294 circx=plot(2*pi*cos(phi)+20,2*pi*sin(phi),'b');
295
296 xlim([-30 30]);
297 ylim([-8 8]);
298 axis([-30 30 -10 10])
299
300 txtxy = 'x-y plan';
301 txtzx = 'z-x plan';
302 txtyz = 'y-z plan';
303 text(-23,-7.5,txtxy);
304 text(-3,-7.5,txtzx);
305 text(17,-7.5,txtyz);
306 title('interaktiva plan','FontSize',12);
307
308 h=impoint(gca,[0 7]);
309
310 p=[0 7];
```

```
311 p2=p;
312
313 dist=dist^2;
314
315
316 %inom while loopen definieras all interaktion mellan programmet
317 %och anv'andare. F'orst initieras programmets get-funktioner f'or
318 %att finna interaktionspunkternas nuvarande l'agen, varefter
319 %if satser kollar om interaktionspunkterna 'ar tillr'ackligt n'ara
    klotpunkten
320 %f'or att klotpunkten skall f'orflyttas, varefter bollens l'age justeras
321 %och vektorsystemet roteras efter den boll i rummet som valts ut.
322
323 bt=b;
324 while(true)
325         pause(pauselength)
326         p=h.getPosition;
327
328         if(pointdistance<=norm(p-p2));
329     if((p(1)-b(1)+20)^2+(p(1)-b(2))^2<=dist)
330         b(1)=p(1)+20;
331         b(2)=p(2);
332         elseif((p(1)-b(3))^2+(p(2)-b(1))^2<=dist)
333                 b(3)=p(1);
334                 b(1)=p(2);
335         elseif((p(1)-b(2)-20)^2+(p(2)-b(3))^2<=dist)
336                 b(2)=p(1)-20;
337                 b(3)=p(2);
338     end
339
340     %Då b's norm (phi) 'ar st'orre 'an 2*pi reflekteras den genom den
        linje som
341    %delar dess riktning med storleken (4*pi-phi)
342
343    %{
344    %Om phi=2*pi+phi', phi'>0 får vi phi->phi2=2*pi-phi', eftersom vi
       har
345    %modulo 2*pi, motsvarar ju denna punkt samma kvaternion som man drog
        till,
346    %fast b skall hållas inom klotet.
347    %}
348
349         if(2*pi<=norm(b))
350         b=-(4*pi-norm(b))*b/norm(b);
351         end
352
353    %if(balldist<(bt(1)-b(1))^2+(bt(2)-b(2))^2+(bt(3)-b(3))^2)
354
355     %uppdaterar storleken hos cirklarna i planen efter b-vektorns nya l
        'age
356
357
358     rx=sqrt((2*pi)^2-b(1)^2);
359     ry=sqrt((2*pi)^2-b(2)^2);
360     rz=sqrt((2*pi)^2-b(3)^2);
361
```

```
362             set(circx,'XData',rx*cos(phi)+20,'YData',rx*sin(phi))
363             set(circy,'XData',ry*cos(phi),'YData',ry*sin(phi))
364             set(circz,'XData',rz*cos(phi)-20,'YData',rz*sin(phi))
365
366             %set funktioner som uppdaterar klotpunktens l'age
367             set(bxyz,'XData',[(b(1)-20) b(3) b(2)+20],'YData',[b(2) b(1) b
                    (3)]);
368             set(u,'XData',b(1)+m*0.35,'YData',b(2)+n*0.35,'ZData',b(3)+o
                    *0.35)
369             set(uxyz,'XData',[b(1) 0 0],'YData',[0 b(2) 0],'ZData',[0 0 b
                    (3)])
370
371             %kvaternionen exp(b/2) tas ut, och RotateVectorQ 'ar en funktion
372             %i quaternion.m som roterar systemet givet en kvaternion
373             q=exp(quaternion(0,b(1)/2,b(2)/2,b(3)/2));
374             x=RotateVectorQ(q,[1 0 0]);
375             y=RotateVectorQ(q,[0 1 0]);
376             z=RotateVectorQ(q,[0 0 1]);
377
378             %set funktioner som uppdaterar vektorrummet
379             set(vec1,'udata',x(1),'vdata',x(2),'wdata',x(3));
380             set(vec2,'udata',y(1),'vdata',y(2),'wdata',y(3));
381             set(vec3,'udata',z(1),'vdata',z(2),'wdata',z(3));
382             bt=b;
383         %end
384         p2=p;
385             end
386 end
```

C.3 SR

```
1  clear all; clf;
2
3  %{
4  %Detta program 'ar skapat i samband med ett kandidatprojekt,
5  %och 'ar menat att vara en del i ett paket av tre
6  %(VR,BR och SR(detta program), d'ar VR står f'or vektorrummet, BR f'or
7  %Bivektorrummet
8  %och SR f'or spinorrummet).
9
10 %F'or att BR och SR skall kunna k'ora kr'avs att man laddar hem quaternion
       .m
11 %F'or SR kr'avs 'aven platonic_solid.m
12
13 %Vardera program skall gå att anv'anda utan tillgång till de andra, och
       går
14 %att anv'anda utan djupare kunskap i matematiken om bara f'or n'ojes skull
       .
15
16 %Denna inledning 'ar skriven f'or att ge insikt i vad de olika
17 %programmen g'or, vad de 'ar till f'or, hur man anv'ander dem, och hur de 'a
       r
18 %uppbyggda.
19 %Den 'ar skriven f'or att f'ors'oka så långt det går enhetligt
20 %beskriva samtliga tre program samtidigt, så
21 %att det inte finns behov av att l'asa inledningen i någon av de andra.
```

```
22
23 %Mer specifika detaljer relaterade till varje kods uppbyggnad; vad de
       olika
24 %momenten g'or och varf'or de skrivs på det s'att de g'ors skall skrivas n'a
       ra
25 %tillh'orande kodstycke.
26
27 %Kunskaper kring programmering tenderar att variera, så jag har
28 %f'oredragit att ta upp "f'or många detaljer" 'an f'or få i texten, så
       koden
29 %skall vara enkel att f'orstå 'aven utan tidigare kunskaper.
30
31 %BR och VR 'ar en visuell demonstration av hur kvaternionerna
32 %dubbelt t'acker gruppen SO(3), som beskriver rotationer i 3D.
33 %SR 'ar en visualisering av spinorer, som påvisar rotationers "Spin 2
       minne".
34
35 %Programmens strukturer 'ar på många s'att lika;
36 %Den f'orsta koden i varje program har jag satt ut parametrar som 'ar
       ditsatta
37 %så att anv'andaren skall kunna justera efter datorprestanda och
       personliga preferenser,
38 %sedan står så många som m'ojligt av de logiska/matematiska objekten,
39 %vektorer osv
40
41 %D'arefter instantieras diverse geometriska objekt satt i sina
42 %"startl'agen".
43
44 %I vardera program finns tre plan ned på vilka ett rum projiceras
45 %(I BR och SR projiceras "Bivektorrymden", i VR projiceras "
       Vektorrummet").
46 %Dessutom kommer i vardera plan en punkt instantieras
47 %(en blå stj'arna ('*')) som i samtliga texter kommer kallas "
       interaktionspunkten",
48 %f'or att s'arskilja den från punkten i BR, som kommer kallas "
       klotpunkten".
49 %Planen kommer kallas "interaktionsplanen" och kommer ofta beskrivas
50 %som att de "tillh'or" det rum som projicerades ned på det
51 %(exempelvis h'ar i VR kan texten "vektorrummets plan" f'orekomma).
52 %Bland de objekt som finns visuellt representerade i planen finns
53 %komponenter som kan interagera med interaktionspunkter,
54 %vilka kommer kallas 'interaktionsobjekt'.
55 %interaktionspunkterna går att flytta i det plan i vilket det 'ar
       definierat
56 %Genom att komma tillr'ackligt n'ara ett interaktionsobjekt
57 %kommer f'oremålet att flytta sig så den hamnar på samma plats som
58 %interaktionspunkten.
59 %interaktionsplanen 'ar alltid x-y,z-x, och y-z planen.
60 %Det 'ar v'art att anm'arka att jag valt att kalla ett plan z-x. Det var
61 %delvis f'or att behålla orientering och så att samtliga grundtillstånd
62 %hos vektorrummets plan har samma form, samt att det finns något
       trevligt
63 %overlapp i meningen att n'ar man i Spin-Algebran ofta n'amner
64 %bivektorn e13 som e31 (eller snarare har vi i,j,k-> -e23,-e31,-e12).
65
66 %"Vektorrummet" och "Bivektorrummet" representeras i samtlig kod
```

```
67 %vektorrummets visuella komponenter 'ar tre stycken vektorer,
68 %som skall representera ett koordinatystem
69 %(x-axel, y-axel och en z-axel) och i en bredare bem'arkelse, vektorer i
70 %tre dimensioner.
71 %Bivektorrummet består av ett klot med radie 2*pi
72 %som 'ar en visuell representation av enhetskvaterniornerna, vilka utg'or
73 %en tredimensionell sf'ar inb'addad i fyra dimensioner, och
74 %om b=[b1 b2 b3] 'ar en punkt i B(0,2*pi) kan den relateras till en
       specifik
75 %kvaternion via b->exp((b1*i+b2*j+b3*k)/2)
76 %Bollens fr'amsta komponent 'ar en punkt (kallad klotpunkt, kommer ibland
       i kod understryka
77 %att den 'ar r'od, f'or att s'arskilja den från interaktionspunkterna, och
       n'ar jag diskuterar
78 %den som en kvaternion eller vektor kommer jag kalla den b)
79 %bollen har f'or 'ovrigt ett modulus på 4*pi (exp((4*pi*j)/2)=1), så n'ar
       b kommer "'over" 4*pi
80 %kommer den f'orflyttas till ett motsvarande l'age på andra sidan klotet.
81 %som beskriver en specifik punkt i klotet.
82 %f'or att ge en b'attre bild av var i klotet punkten befinner sig har
83 %"skuggbilder satts ut, vilka beskriver b's projektioner på
       Bivektorrymdens axlar.
84
85 %I VR 'ar det vektorrummet som 'ar projicerat på interaktionsplanen.
86 %vektorrummets interaktionsobjekt 'ar de olika vektorernas pilspetsar;
87 %n'ar interaktionspunkten kommer n'ara pilspetsen kommer dess vektor
       flyttas
88 %mot punkten på så s'att att vektorns  l'angd inte
89 %'andras, dvs den roterar mot interaktionspunkten,
90 %varefter hela koordninatsystemet kommer rotera efter samma axel och
91 %med samma vinkelutslag.
92 %Varje system av koordinataxlar (x',y',z'), motsvarar en
       rotationsmatris:
93 %[x' y' z'] (eftersom x' y' z' 'ar ortonormerade, vi har
94 %R[e1 e2 e3]=[R(e1) R(e2) R(e3)]=[x' y' z']
95 %Ifrån denna rotationsmatris kan man få ut en vektor i Bivektorrymden
96 %genom att betrakta matrisens "clifford trace", vilket vi kan
97 %relatera till en specifik axel och en specifik vinkel, vilket
98 %då 'ar en b-vektor som motsvarar klotpunktens nya l'age i bollrummet.
99
100 %Det som tydligt m'arks 'ar att 'aven då rotationssystemet roterat 360
       grader
101 %kommer inte b-vektorn ha återgått till [0 0 0], utan det 'ar f'orst
       efter
102 %två rotationer som systemet har återst'allts helt.
103
104
105 %I BR 'ar det bollrummet som 'ar projicerat på interaktionsplanen, och
106 %interaktionsobjektet 'ar klotpunkten. Då interaktionspunkten kommer n'a
       ra
107 %klotpunkten flyttas klotpunkten till interaktionspunktens l'age.
108 %I varje interaktionsplan finns 'aven en cirkel utritad, som motsvarar
109 %randen av BR.
110 %klotpunkten motsvarar ('ar) ju en b-vektor, vilket vi kan relatera till
       en enhets-
111 %kvaternion q=exp(b/2), d'ar vi nu kan rotera koordinatsystemet
```

```
112 %med qvq^-1. Mer detaljrik f'orklaring finns i projektrapporten.
113 %I koden i sig anv'ands ett befintligt program som fanns i quaternion.m,
114 %som jag ej vet hur den 'ar uppbyggd.
115
116 %I SR har vi spinorrummet, vilket likt BR 'ar ett klot av radie 2*pi,
117 %vari det ligger fyra stycken punkter (återigen 'stj'arnor', (*))
118 %som motsvarar spinorer.
119 %Spinorer beskrivs ofta som vektorer i C^2, d'ar det finns en
120 %kvaternionrepresentation p(q)=[z w;-w' z'], som 'verkar' på
121 %spinorerna. I SR går vi 'over till R^4, oh eftersom p bevarar l'angder
122 %kommer "enhetsspinorer" befinna sig i sf'aren, så vi representerar
123 %spinorerna i någon mening som om de vore "kvaternioner".
124 %I SR 'ar det återigen BR som projiceras på interaktionsplanen, med
125 %samma interaktionssystem som i BR, och n'ar bollens r'orelse resulterar
126 %i f'or'andringar av kvaternionens l'age genom det p(q) som beskrevs ovan.
127
128 %f'or att hj'alpa visualiseringen av spinorernas l'agen finns 'aven
129 %"skuggbilder", men eftersom det inneb'ar att spinorrymden har 16
    stycken
130 %visualiserade punkter, 'ar det valbart att s'atta på dem.
131 %det finns 'aven projektioner av spinorerna ned på "golvet" och ena
132 %"v'aggen", f'or att ge mer insikt i deras l'agen. Den 'ar satt som "på" om
133 %anv'andaren inte st'anger av dem.
134
135
136 %samtliga interaktioner mellan program och anv'andare
137 %i varje kod hanteras inom en while loop, som alltid 'ar satt till 'true
    '.
138 %interaktionspunkterna 'ar "impoint"-objekt.
139 %Som instantieras strax efter planet det befinner sig i.
140 %Dessa kallas på i loopen med en "get"
141 %(F'or den ovane: get/getters och set/setters 'ar
142 %i programmering typiska funktioner att ge klasser. En "get" låter
143 %anv'andaren l'asa av variabler/h'amta information som ett objekt har
144 %(i detta fall har vi en impoint, vars variabel vi plockar 'ar "Position
    "
145 %Den kan instantieras via h.impoint();, och positionen fås då
146 %via h.getPosition.))
147 %interaktionspunktens l'age sparas, och en serie if-satser kollar om
148 %den nuvarande punkten 'ar n'ara något interaktionsobjekt.
149 %pointdistance garanterar att de interaktionspunkter som inte vidr'ors
150 %ej kan interagera med sitt plans interaktionsobjekt, vilket hade gjort
151 %programmet svårnavigerat.
152 %}
153
154 %nedan introduceras en del konstanter som anv'andaren kan v'alja att
155 %justera efter behov, beroende på dennes dators prestandakrav osv.
156 %{
157
158 %1. 'pauselength' best'ammer, i sekunder, den tid det skall ta mellan
    varje
159 %iteration av att l'asa av vektorpunktens l'age och positionera
    vektorsystemet och
160 %klotpunkten efter det.
161
162
```

```
163 %2. 'dist' beskriver från vilket avstånd interaktionspunkten skall vara
164 %från någon av klotpunktens projektioner innan r'orelse aktiveras.
165
166 %3. Pointdistance uts'atter ett krav
167 %som begr'ansar antalet processer som g'ors i while loopen, genom att
168 %kontrollera vilket avstånd den nuvarande interaktionspunkten har
169 %relativt den tidigare iterationspunkten
170 %dvs om man s'atter pointdistance 0.01 måste interaktionspunkten ha f'o
    rflyttats
171 %0.01 enheter från det l'age då klotpunkten senast flyttades, innan
172 %programmet kontrollerar huruvida klotpunkten skall flyttas relativt
173 %interaktionspunktens nuvarande l'age.
174 %Den har 'aven funktionen att utan den hade 'aven de interaktionspunkter
175 %i planen du inte befinner dig i hade kunnat påverka systemet.
176 %Jag hade tidigare f'ors'okt med en liknande funktion, ekvivalent
177 %med denna om pointdistance var noll.
178 %Av någon anledning fick jag dock st'orningar, jag tror det kan bero på
179 %impoint, men 'ar definitivt inte s'aker.
180
181 %Med balldist kontrolleras hur långt klotpunkten skall ha r'ort
182 %sig innan grafiken uppdateras. Dess analog i VR 'ar "vecdist"
183
184 %}
185
186 dist=1.5;
187 pauselength=0.05;
188 pointdistance=0.1;
189 balldistance=0.1;
190
191 %shadeon s'atter ut "skuggade" versioner av spinorerna (mer transparenta
    )
192 %som 'ar utsatta på x-y-z axlarna
193 %projectionon s'atter ut projektioner av spinorerna, plana bilder
194 %på z=-10 och x=10 planen.
195 %de syns om konstanten 'ar satt till 1.
196 %shadeon 'ar f'or tillf'allet av, det blir l'att v'aldigt många punkter
    vilket
197 %kan g'ora det svårt att se
198
199 shadeon=0;
200 projectionon=1;
201
202 %de matematiska/logiska objekten
203 %b 'ar punkten i klotet av radie 2*pi, som relaterar till
    kvaternionrummet
204
205 %som en exponential.b1,b2,b3,b4 'ar de vektorer som beskriver
206 %spinorerna som representerade i kvaternionrummet (spinorpunkter)
207 %spinorerna 'ar i sig vektorer i ett 4 dimensionellt rum, som vi h'ar
208 %relaterar till klotet genom att betrakta vardera index som ett bidrag
    till
209 %en kvaternion: q=[q1 q2 q3 q4]-> q1+i*q2+j*q3+k*q4
210
211 %x,y,z 'ar basvektorerna i vektorrymden
212
213 %r best'ammer storleken hos spinorerna
```

```
214
215 b=[0 0 0];
216
217 b1=[0 0 0];
218 b2=[pi 0 0];
219 b3=[0 pi 0];
220 b4=[0 0 pi];
221
222 x=[1;0;0];
223 y=[0;1;0];
224 z=[0;0;1];
225
226 r=0.5;
227
228 shadedist=10;
229
230
231 %f'orst ritas bollrymden (BR) ut
232
233 figure(1)
234
235 set(gcf, 'Position', [100, 100, 1800, 600])
236
237
238 subplot(2,6,[1 2 7 8])
239
240 %koden nedan ritar ut klotet i vilket klotpunkten (b) befinner sig
241
242 [m n o]=sphere; %total sphere bdry => q=-1
243 surf(2*pi*m,2*pi*n,2*pi*o,'facecolor','blue', 'facealpha',.15,'
    EdgeAlpha',0);
244 hold on
245
246 surf(pi*m,pi*n,pi*o,'facecolor','green', 'facealpha',.3,'EdgeAlpha',0);
247 phi=linspace(0,2*pi);
248 plot(2*pi*cos(phi),2*pi*sin(phi),'black') % circle for visualisation
249 hold on
250 plot3([-10 10],[0 0], [0 0],'black',[0 0],[-10 10], [0 0],'black',[0
    0],[0 0], [-10 10],'black')
251
252 %klotpunkten ritas ut
253 %samt tre stycken "skuggbilder"; projektioner av b på klotrummets x,y
    och z axlar
254 %utritade i gråskala med svag ljusinst'allning, f'or att ge mer insikt
255 %i var inom klotrummet bollen befinner sig i.
256 %(bollen står sist i koden så att den inte 'overt'acks av skuggbilderna)
257
258 %uxyz=plot3([0 0 0],[0 0 0],[0 0 0],'*','Color',[0.5 0.5 0.5]);
259
260 uxyz=surf([0.35*m;0.35*m;0.35*m],[0.35*n;0.35*n;0.35*n],[0.35*o;0.35*o
    ;0.35*o],'facecolor','red', 'facealpha',0.2,'EdgeAlpha',0);
261
262 u=surf(0.35*m,0.35*n,0.35*o,'facecolor','red', 'facealpha',1,'EdgeAlpha
    ',0);
263
264 xlabel('x')
```

```
265 ylabel('y')
266 zlabel('z')
267 title('Bivektorrymden BR','FontSize',14);
268 grid on
269 axis([-7 7 -7 7 -7 7])
270 axis equal
271
272 %kodstycket nedan visualiserar genererar spinorrymden och spinorerna
273
274 %klotet d'ar spinorerna 'ar utsatta (Spinorrymden)
275
276 subplot(2,6,[3 4 9 10])
277
278 surf(2*pi*m,2*pi*n,2*pi*o,'facecolor','blue', 'facealpha',0.15,'
    EdgeAlpha',0);
279 hold on
280 surf(pi*m,pi*n,pi*o,'facecolor','green', 'facealpha',.3,'EdgeAlpha',0);
281
282 xlabel('x')
283 ylabel('y')
284 zlabel('z')
285
286 plot(2*pi*cos(phi),2*pi*sin(phi),'black') % circle for visualisation
287
288 plot3([-10 10],[0 0], [0 0],'black',[0 0],[-10 10], [0 0],'black',[0
    0],[0 0], [-10 10],'black')
289
290
291 %spinorerna, representerade som platonska solider inom ett klot
292 %samt skuggbilder
293
294 if(shadeon==1)
295 [Vt,F]=platonic_solid(5,r);
296 Vt=[Vt(:,1)+b1(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b1(2),Vt(:,3);Vt(:,1)
    ,Vt(:,2),Vt(:,3)+b1(3)];
297 u1xyz=patch('Faces',[F;F+20;F+20],'Vertices',Vt,'FaceColor','b','
    FaceAlpha',0.2,'EdgeColor','k','Edgealpha',0.2,'LineWidth',0.1);
298 [Vt,F]=platonic_solid(2,r);
299 Vt=[Vt(:,1)+b2(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b2(2),Vt(:,3);Vt(:,1)
    ,Vt(:,2),Vt(:,3)+b2(3)];
300 u2xyz=patch('Faces',[F;F+8;F+16],'Vertices',Vt,'FaceColor','r','
    FaceAlpha',0.2,'EdgeColor','k','Edgealpha',0.2,'LineWidth',0.1);
301 [Vt,F]=platonic_solid(3,r);
302 Vt=[Vt(:,1)+b3(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b3(2),Vt(:,3);Vt(:,1)
    ,Vt(:,2),Vt(:,3)+b3(3)];
303 u3xyz=patch('Faces',[F;F+6;F+12],'Vertices',Vt,'FaceColor','g','
    FaceAlpha',0.2,'EdgeColor','k','Edgealpha',0.2,'LineWidth',0.1);
304 [Vt,F]=platonic_solid(4,r);
305 Vt=[Vt(:,1)+b4(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b4(2),Vt(:,3);Vt(:,1)
    ,Vt(:,2),Vt(:,3)+b4(3)];
306 u4xyz=patch('Faces',[F;F+12;F+24],'Vertices',Vt,'FaceColor','y','
    FaceAlpha',0.2,'EdgeColor','k','Edgealpha',0.2,'LineWidth',0.1);
307 end
308
309 %projektionsbilder f'or spinorerna
310 %1. blå, dodekahedronens projektion (dodekagon)
```

```
311 %2. r'od, kubens projektion (kvadrat)
312 %3. gr'on, hexaederns projektion (romb)
313 %. gul, isokahedronen (hexagon)
314 if(projectionon==1)
315
316 dodecax=[1/2;sqrt(3)/4;1/2^2;0;-1/2^2;-sqrt(3)/4;-1/2;-sqrt(3)
        /4;-1/2^2;0;1/2^2;sqrt(3)/4];
317 dodecay=[0;1/2^2;sqrt(3)/4;1/2;sqrt(3)/4;1/2^2;0;-1/2^2;-sqrt(3)
        /4;-1/2;-sqrt(3)/4;-1/2^2];
318 d3=[10;10;10;10;10;10;10;10;10;10;10;10];
319
320 v1=[dodecax+b1(1) dodecay+b1(2) -d3;d3 dodecax+b1(2) dodecay+b1(3)];
321
322 Fdodeca=[1 2 3 4 5 6 7 8 9 10 11 12];
323
324 u12xyz=patch('Faces',[Fdodeca;Fdodeca+12],'Vertices',v1,'FaceColor','
        blue'); hold on
325
326 squarex=[-1/2;1/2;1/2;-1/2];
327 squarey=[-1/2;-1/2;1/2;1/2];
328 d1=[10;10;10;10];
329
330 u22xyz=patch('Faces',[1 2 3 4;5 6 7 8],'Vertices',[squarex+b2(1)
        squarey+b2(2) -d1;d1 squarex+b2(2) squarey+b3(3)],'FaceColor','red'
        ); hold on
331
332 rhombusx=[-1/2;0;1/2;0];
333 rhombusy=[0;1/2;0;-1/2];
334 d4=[10;10;10;10];
335
336 v1=[rhombusx rhombusy -d4;d4 rhombusx rhombusy];
337
338 F=[1 2 3 4];
339
340 u32xyz=patch('Faces',[F;F+4],'Vertices',[rhombusx+b3(1) rhombusy+b3(2)
        -d4;d4 rhombusx+b3(2) rhombusy+b3(3)],'FaceColor','green'); hold on
341
342
343 icosax=[0;sqrt(3)/4;sqrt(3)/4;0;-sqrt(3)/4;-sqrt(3)/4];
344 icosay=[1/2;1/4;-1/4;-1/2;-1/4;1/4];
345 d2=[10;10;10;10;10;10];
346
347 u42xyz=patch('Faces',[1 2 3 4 5 6;7 8 9 10 11 12],'Vertices',[icosax+b4
        (1) icosay+b4(2) -d2;d2 icosax+b4(2) icosay+b4(3)],'FaceColor','
        yellow'); hold on
348
349 end
350
351
352 [Vt,F]=platonic_solid(5,r);
353 Vt=[Vt(:,1)+b1(1),Vt(:,2)+b1(2),Vt(:,3)+b1(3)];
354 u1=patch('Faces',F,'Vertices',Vt,'FaceColor','b','FaceAlpha',1,'
        EdgeColor','k','LineWidth',0.1);
355 [Vt,F]=platonic_solid(2,r);
356 Vt=[Vt(:,1)+b2(1),Vt(:,2)+b2(2),Vt(:,3)+b2(3)];
357 u2=patch('Faces',F,'Vertices',Vt,'FaceColor','r','FaceAlpha',1,'
```

```
    EdgeColor','k','LineWidth',0.1);
[Vt,F]=platonic_solid(3,r);
Vt=[Vt(:,1)+b3(1),Vt(:,2)+b3(2),Vt(:,3)+b3(3)];
u3=patch('Faces',F,'Vertices',Vt,'FaceColor','g','FaceAlpha',1,'
    EdgeColor','k','LineWidth',0.1);
[Vt,F]=platonic_solid(4,r);
Vt=[Vt(:,1)+b4(1),Vt(:,2)+b4(2),Vt(:,3)+b4(3)];
u4=patch('Faces',F,'Vertices',Vt,'FaceColor','y','FaceAlpha',1,'
    EdgeColor','k','LineWidth',0.1);

grid on
axis([-7 7 -7 7 -7 7])
axis equal
title('Spinorrymden SR','FontSize',14);


subplot(2,6,[5 6 11 12])


vecx=quiver3([0 -0.5],[0 -0.5],[0 -1.2],[x(1) 0.5],[x(2) 0],[x(3) 0],'b
    ','Autoscale','off'); hold on; %generate a rotateable x-y-z
    coordinate system
vecy=quiver3([0 -0.5],[0 -0.5],[0 -1.2],[y(1) 0],[y(2) 0.5],[y(3) 0],'r
    ','Autoscale','off');
vecz=quiver3([0 -0.5],[0 -0.5],[0 -1.2],[z(1) 0],[z(2) 0],[z(3) 0.5],'g
    ','Autoscale','off');

axis([-1.2 1.2 -1.2 1.2 -1.2 1.2])

pbaspect([1 1 1]);
xlabel('x')
ylabel('y')
zlabel('z')
title('Vektorrymden VR','FontSize',14);


%{
%kod som projicerar klotrummet på x-y,z-x och y-z planen,
%inritade i en gemensam plot, d'ar en punkt (h'ar kallade
    interaktionspunkten (den blå)
%låter dig manipulera klotpunktens (den r'oda) l'age.
%Om man f'orflyttar interaktionspunkten så den hamnar
%tillr'ackligt n'ara klotpunkten (avståndet best'ams av 'dist') kommer
%klotpunkten att f'orflyttas till interaktionspunktens plats.

%cirkeln inritat i planet beskriver avståndet punkten har från
%klotets rand i detta plan, dess storlek uppdateras efter
%klotpunktens nya l'agen.
%}

figure(2);
set(gcf, 'Position', [100, 100, 600, 200])

bxyz=plot([b(1)-20 b(3) b(2)+20],[b(2) b(1) b(3)],'red*');hold on

circz=plot(2*pi*cos(phi)-20,2*pi*sin(phi),'g');
```

```
406 circy=plot(2*pi*cos(phi),2*pi*sin(phi),'r');
407 circx=plot(2*pi*cos(phi)+20,2*pi*sin(phi),'b');
408
409 axis([-30 30 -10 10])
410
411 txtxy = 'x-y plan';
412 txtzx = 'z-x plan';
413 txtyz = 'y-z plan';
414 text(-23,-7.5,txtxy);
415 text(-3,-7.5,txtzx);
416 text(17,-7.5,txtyz);
417 title('interaktiva ytor','FontSize',12);
418
419
420 h=impoint(gca,[0 7]);
421
422 %p sparar impointens l'age i loopen,
423 %p2 anv'ands f'or att kontrollera huruvida interaktionspunkten flyttats
424 %tillr'ackligt f'or att grafiken skall uppdateras
425
426 p=[0 7];
427 p2=p;
428
429 pointdistance=pointdistance^2;
430 dist=dist^2;
431 balldistance=balldistance^2;
432
433
434 %{
435 %inom while loopen definieras all interaktion mellan programmet
436 %och anv'andare. F'orst initieras programmets get-funktioner f'or
437 %att finna interaktionspunkternas nuvarande l'agen, varefter
438 %if satser kollar om interaktionspunkterna 'ar tillr'ackligt n'ara
      klotpunkten
439 %f'or att klotpunkten skall flyttas, vilket relateras till en kvaternion
440 %som vi representerar som en matris som kommer transformera spinorerna,
441 %vars l'agen kan relateras till en boll på samma s'att som man kan
442 %enhetskvaternionerna
443 %}
444 while(true)
445         pause(pauselength)
446         p=h.getPosition;
447         if(pointdistance<=(p(1)-p2(1))^2+(p(2)-p2(2))^2)
448         dist1=(p(1)-b(1)+20)^2+(p(2)-b(2))^2;
449         dist2=(p(1)-b(3))^2+(p(2)-b(1))^2;
450         dist3=(p(1)-b(2)-20)^2+(p(2)-b(3))^2;
451         if(balldistance<=dist1||pointdistance<=dist2||pointdistance<=
              dist3)
452     if((p(1)-b(1)+20)^2+(p(1)-b(2))^2<=dist)
453         b(1)=p(1)+20;
454         b(2)=p(2);
455         elseif((p(1)-b(3))^2+(p(2)-b(1))^2<=dist)
456                 b(3)=p(1);
457                 b(1)=p(2);
458         elseif((p(1)-b(2)-20)^2+(p(2)-b(3))^2<=dist)
459                 b(2)=p(1)-20;
```

```
460                     b(3)=p(2);
461       end
462
463         %{
464       %Om phi=2*pi+phi', phi'>0 får vi phi->phi2=2*pi-phi', eftersom vi
              har
465       %modulo 2*pi, motsvarar ju denna punkt samma kvaternion som man drog
              till,
466       %fast b skall hållas inom klotet.
467       %}
468             if(2*pi<=norm(b))
469             b=-(4*pi-norm(b))*b/norm(b);
470             end
471
472
473       %skippar uppdateringar om b-vektorn inte f'orflyttats
474       %tillr'ackligt (kan vara 'overfl'odig ide...)
475   %     if(balldist<=(bt(1)-b(1))^2+(bt(2)-b(2))^2+(bt(3)-b(3))^2)
476
477
478
479       %kvaternionens nya l'age best'ammer spinorernas
480
481       %d'arefter beskrivs spinorerna som enhetskvaternioner
482       %i den andemening att de kan representeras med deras respektive
483       %deras j,phi, som best'ammer
484       %deras positioner som punkter inom ett kvaternionklot
485       %(dvs)b1=phi1*j1, d'ar de relateras till en
486       %kvaternion via q1=exp(b1/2) (analogt f'or b2,b3,b4)
487
488       %{
489       %Av något sk'al st'allde jag om systemet så matrismultiplikationen
              gjordes
490       %inuti koden, d'ar man plockar ur m
491       %Känner mig manad att återst'alla den representation som anv'andes,
492       %ifall det finns intresse (kontrollering av r'akneregler osv):
493
494       %rho(q)=rho(q1+i*q2+j*q3+k*q4)=
495       %q1*I+q2*Î+q3*J+q4*K =
496       %[q1 0 0 0;      [0 q2 0 0;     [0 0 q3 0;     [0 0 0 q4;
497       % 0 q1 0 0;   + -q2 0 0 0;  +   0 0 0 -q3; +   0 0 q4 0;
498       % 0 0 q1 0;      0 0 0 q2;      -q3 0 0 0;     0 -q4 0 0;
499       % 0 0 0 q1]      0 0 -q2 0]     0 q3 0 0]     -q4 0 0 0]
500       %}
501
502       q=exp(quaternion(0,b(1)/2,b(2)/2,b(3)/2));
503       J1=[q.e(2);q.e(3);q.e(4)];
504       J2=[q.e(1);q.e(4);-q.e(3)];
505       J3=[-q.e(4);q.e(1);q.e(2)];
506       J4=[q.e(3);-q.e(2);q.e(1)];
507
508       phi1=2*acos(q.e(1));
509       phi2=2*acos(-q.e(2));
510       phi3=2*acos(-q.e(3));
511       phi4=2*acos(-q.e(4));
512
```

```
513         b1=phi1*J1;
514         b2=phi2*J2;
515         b3=phi3*J3;
516         b4=phi4*J4;
517
518         %RotateVectorQ 'ar en funktion i quaternion.m som
519         %roterar systemet givet en kvaternion
520             x=RotateVectorQ(q,[1 0 0]);
521             y=RotateVectorQ(q,[0 1 0]);
522             z=RotateVectorQ(q,[0 0 1]);
523
524             %set-funktioner som uppdaterar skuggbildernas l'agen
525
526         %uppdaterar storleken hos cirklarna i planen efter b-vektorns nya l
                'age
527         %(cirklarna skall ge inblick hur n'ara klotpunkten 'ar klotets rand)
528
529         rx=sqrt((2*pi)^2-b(1)^2);
530         ry=sqrt((2*pi)^2-b(2)^2);
531         rz=sqrt((2*pi)^2-b(3)^2);
532
533             set(circx,'XData',rx*cos(phi)+20,'YData',rx*sin(phi))
534             set(circy,'XData',ry*cos(phi),'YData',ry*sin(phi))
535             set(circz,'XData',rz*cos(phi)-20,'YData',rz*sin(phi))
536
537             %set funktioner som uppdaterar klotpunktens l'age
538             %samt interaktionsobjekt (klotpunkt i plan) och skuggbilder
539
540             set(bxyz,'XData',[(b(1)-20) b(3) b(2)+20],'YData',[b(2) b(1) b
                    (3)]);
541             set(u,'XData',b(1)+m*0.35,'YData',b(2)+n*0.35,'ZData',b(3)+o
                    *0.35);
542             set(uxyz,'XData',[b(1)+0.35*m;0.35*m;0.35*m],'YData',[0.35*n;b
                    (2)+0.35*n;0.35*n],'ZData',[0.35*o;0.35*o;b(3)+0.35*o])
543
544
545
546         %Spinorer samt dess 'skuggbilder' och projektioner uppdateras
547
548             if(shadeon==1)
549
550             [Vt,F]=platonic_solid(5,r);
551             Vt=[Vt(:,1)+b1(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b1(2),Vt(:,3)
                    ;Vt(:,1),Vt(:,2),Vt(:,3)+b1(3)];
552             set(u1xyz,'Faces',[F;F+20;F+20],'Vertices',Vt);
553             [Vt,F]=platonic_solid(2,r);
554             Vt=[Vt(:,1)+b2(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b2(2),Vt(:,3)
                    ;Vt(:,1),Vt(:,2),Vt(:,3)+b2(3)];
555             set(u2xyz,'Faces',[F;F+8;F+16],'Vertices',Vt);
556             [Vt,F]=platonic_solid(3,r);
557             Vt=[Vt(:,1)+b3(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b3(2),Vt(:,3)
                    ;Vt(:,1),Vt(:,2),Vt(:,3)+b3(3)];
558             set(u3xyz,'Faces',[F;F+6;F+12],'Vertices',Vt);
559             [Vt,F]=platonic_solid(4,r);
560             Vt=[Vt(:,1)+b4(1),Vt(:,2),Vt(:,3);Vt(:,1),Vt(:,2)+b4(2),Vt(:,3)
                    ;Vt(:,1),Vt(:,2),Vt(:,3)+b4(3)];
```

```
561             set(u4xyz,'Faces',[F;F+12;F+24],'Vertices',Vt);
562
563             end
564
565             if(projectionon==1)
566             set(u12xyz,'Faces',[Fdodeca;Fdodeca+12],'Vertices',[dodecax+b1
                    (1) dodecay+b1(2) -d3;d3 dodecax+b1(2) dodecay+b1(3)]);
567             set(u22xyz,'Faces',[1 2 3 4;5 6 7 8],'Vertices',[squarex+b2(1)
                    squarey+b2(2) -d1;d1 squarex+b2(2) squarey+b3(3)]);
568             set(u32xyz,'Faces',[1 2 3 4;5 6 7 8],'Vertices',[rhombusx+b3(1)
                     rhombusy+b3(2) -d4;d4 rhombusx+b3(2) rhombusy+b3(3)]);
569             set(u42xyz,'Faces',[1 2 3 4 5 6;7 8 9 10 11 12],'Vertices',[
                    icosax+b4(1) icosay+b4(2) -d2;d2 icosax+b4(2) icosay+b4(3)
                    ]);
570
571             end
572
573
574                 [Vt,F]=platonic_solid(5,r);
575                 Vt=[Vt(:,1)+b1(1),Vt(:,2)+b1(2),Vt(:,3)+b1(3)];
576                 set(u1,'Faces',F,'Vertices',Vt);
577                 [Vt,F]=platonic_solid(2,r);
578                 Vt=[Vt(:,1)+b2(1),Vt(:,2)+b2(2),Vt(:,3)+b2(3)];
579                 set(u2,'Faces',F,'Vertices',Vt);
580                 [Vt,F]=platonic_solid(3,r);
581                 Vt=[Vt(:,1)+b3(1),Vt(:,2)+b3(2),Vt(:,3)+b3(3)];
582                 set(u3,'Faces',F,'Vertices',Vt);
583                 [Vt,F]=platonic_solid(4,r);
584                 Vt=[Vt(:,1)+b4(1),Vt(:,2)+b4(2),Vt(:,3)+b4(3)];
585                 set(u4,'Faces',F,'Vertices',Vt);
586
587             set(vecx,'udata',[x(1) 0.5],'vdata',[x(2) 0],'wdata',[x(3) 0]);
588             set(vecy,'udata',[y(1) 0],'vdata',[y(2) 0.5],'wdata',[y(3) 0]);
589             set(vecz,'udata',[z(1) 0],'vdata',[z(2) 0],'wdata',[z(3) 0.5]);
590             end
591             p2=p;
592             end
593
594 end
```