1 Matrixmultiplikation

Seien A un B zwei (3×3) Matrizen dann ist $C = (c_{ij})$

$C = A \cdot B \Leftrightarrow c_{ij} := a_{ik} b_{kj}$ Summationskonvention: doppelter Index bedeutet Summe

Die Multiplikation ist nur definiert, wenn Spaltenanzahl von A gleich Zeilenanzahl von B ist!
Diese Verknüpfung erlaubt 2 Deutungen:

1.1 Skalare Produkte

Bei fixem i und j durchläuft k bei A den Spaltenindex (greift also den i-ten Zeilenvektor heraus) und bei B den Zeilenindex (greift also den j-ten Spaltenvektor heraus).
Seien $\vec{a}_1$, $\vec{a}_2$, $\vec{a}_3$ die Zeilenvektoren von A und $\vec{b}_1$, $\vec{b}_2$, $\vec{b}_3$ die Spaltenvektoren von B, also

$$A = \begin{pmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{pmatrix} \quad B = \begin{pmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{pmatrix} \Rightarrow A \cdot B = \begin{pmatrix} \vec{a}_1 \cdot \vec{b}_1 & \vec{a}_1 \cdot \vec{b}_2 & \vec{a}_1 \cdot \vec{b}_3 \\ \vec{a}_2 \cdot \vec{b}_1 & \vec{a}_2 \cdot \vec{b}_2 & \vec{a}_2 \cdot \vec{b}_3 \\ \vec{a}_3 \cdot \vec{b}_1 & \vec{a}_3 \cdot \vec{b}_2 & \vec{a}_3 \cdot \vec{b}_3 \end{pmatrix}$$

Beachte, dass das skalare Produkt zweier Vektoren als Matrixmultiplikation gedeutet werden kann. Hat man zwei Spaltenvektoren $\vec{a}$, $\vec{b}$, so ist das skalare(innere) Produkt

$\vec{a} \cdot \vec{b} := \vec{a}^T \cdot \vec{b}$ wobei links das skalare Produkt gemeint ist, rechts die Matrixmultiplikation

Manchmal ist auch das äußere(dyadische) Produkt zweier Vektoren von Interesse:

$\vec{a} \otimes \vec{b} := \vec{a} \cdot \vec{b}^T$ ergibt eine 3 mal 3 Matrix; rechts ist die Matrixmultiplikation gemeint

1.2 Linearkombinationen

Nochmals die Definition:

$C = A \cdot B \Leftrightarrow c_{ij} := a_{ik} b_{kj}$ Summationskonvention: doppelter Index bedeutet Summe

Lassen wir in obiger Formel i alle Werte durchlaufen, dann wird aus c_{ij} der j-te Spaltenvektor $\vec{c}_j$ der Produktmatrix, aus a_{ik} der k-te Spaltenvektor $\vec{a}_k$ von A, wobei über k summiert wird, also

$$\vec{c}_j = \sum_k \vec{a}_k b_{kj} \underbrace{=}_{Summenkonvention} \vec{a}_k b_{kj}$$

Also in der Produktmatrix steht in der j-ten Spalte die Linearkombination der Spaltenvektoren von A, wobei die die Koeffizienten der Linearkombination in der j-ten Spalte von B stehen! Also

$$\begin{pmatrix} \vec{a}_1 & \vec{a}_2 & \vec{a}_3 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_4 \\ b_2 & b_5 \\ b_3 & b_6 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{3} \vec{a}_i \, b_i & \sum_{i=4}^{6} \vec{a}_i \, b_i \end{pmatrix}$$

Sonderfälle sind “Selektionsmatrizen”:

$$\begin{pmatrix} \vec{a}_1 & \vec{a}_2 & \vec{a}_3 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \vec{a}_3 & \vec{a}_2 & \vec{a}_1 \end{pmatrix}$$

Beachte: Durch diese Selektion werden die erste und dritte Spalte vertauscht.
Diese Deutung als Linearkombination werden wir weiter unten in (1) benötigen!

2 Rotation um eine beliebige Achse durch den Ursprung

Statt uns zu überlegen, um welche Winkel wir ein gegebenes Orthonormalsystem drehen müssen, damit eine der Achsen in unsere Drehachse zeigt, führen wir einfach eine Koordinatentransformation durch - wir konzentrieren uns auf den $\mathbb{R}^3$:

Beachte außerdem die Definition einer **mathematisch positiven Drehung**:
Wir schauen in Richtung des Drehachsenvektors und drehen nach **rechts**!

Wir kennen schon die Drehmatrix um die x-Achse $R_x(\alpha)$:

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

(Drehmatrizen um die Standardachsen x-y-z sind leicht herzuleiten indem man sie unbekannt ansetzt und auf die Standardbasisvektoren anwendet - wobei man das Ergebnis kennt!)

Zurück zu unserer Aufgabe: $\vec{e}_1$, $\vec{e}_2$, $\vec{e}_3$ seien unsere Standardbasisvektoren in Spaltenschreibweise. $\vec{e}_1'$ sei der normalisierte Vektor in Richtung der Drehachse. Wir konstruieren ein neues Orthonormalsystem:

$$\vec{e}_2' = \frac{\vec{e}_1 \times \vec{e}_1'}{\|\vec{e}_1 \times \vec{e}_1'\|} \qquad \text{und} \qquad \vec{e}_3' = \vec{e}_1' \times \vec{e}_2'$$

Es gelten folgende Gleichungen (mit Summationskonvention):

$$\vec{e}_i' = a_{ji}\,\vec{e}_j \qquad (1) \qquad \underbrace{\Longleftrightarrow}_{Matrixschreibweise} \qquad (\vec{e}_1',\, \vec{e}_2',\, \vec{e}_3') = (\vec{e}_1,\, \vec{e}_2,\, \vec{e}_3) \cdot A$$

$$\vec{e}_k = b_{jk}\,\vec{e}_j' \qquad (2) \qquad \underbrace{\Longleftrightarrow}_{Matrixschreibweise} \qquad (\vec{e}_1,\, \vec{e}_2,\, \vec{e}_3) = (\vec{e}_1',\, \vec{e}_2',\, \vec{e}_3') \cdot A^{-1}$$

(überlege: Warum ist A invertierbar?)
Die Zahlen b_{jk} stellen die Zahlen der invertierten Matrix von $A = a_{ji}$ dar!

Da beide Basissysteme Orthonormalsysteme sind gilt:

$$\vec{e}_i \cdot \vec{e}_j = \vec{e}_i' \cdot \vec{e}_j' = \delta_{ij} \quad \text{(Kronecker Delta)}$$

Wir multiplizieren nun obige Gleichung (1) mit $\vec{e}_k$ bzw. Gleichung (2) mit $\vec{e}_i'$, dann ergibt sich

$$\vec{e}_i' \cdot \vec{e}_k = a_{ji}\delta_{jk} = a_{ki} \quad \textbf{Berechnungsmethode} \qquad (3)$$

$$\vec{e}_i' \cdot \vec{e}_k = b_{jk}\delta_{ij} = b_{ik} \qquad (4)$$

Einerseits wissen wir aus (3) wie die Matrix A zu berechnen ist und da $b_{ik} = a_{ki}$ gilt in Matrixschreibweise

$$\boxed{B = A^{-1} = A^T}$$

Um den Vektor $\vec{x}$ um $\vec{e}_1'$ um α zu rotieren liegt jetzt unsere Vorgangsweise klar vor uns:

1. Wir transformieren $\vec{x}$ ins gestrichene Koordinatensystem:
$$x_k \vec{e}_k = \underbrace{x_k\, b_{jk}}_{x_j'}\, \vec{e}_j' \Rightarrow x_j' = x_k\, b_{jk} \Rightarrow \vec{x}\,' = B \cdot \vec{x} = A^T\, \vec{x}$$
2. Rotieren dort mit $R_x(\alpha)$
3. Wir machen die Transformation rückgängig mit $\left(A^T\right)^T = A$
$$\boxed{\vec{x}_\alpha = \underbrace{A\, R_x(\alpha)\, A^T}_{T}\, \vec{x}}$$
Dies wäre die Formel für Spaltenvektoren, für Zeilenvektoren müssen wir linke und rechte Seite transponieren (Reihenfolge der Multiplikation dreht sich um):
$$\vec{x}_\alpha^T = (\underbrace{A\, R_x(\alpha)\, A^T}_{T}\, \vec{x})^T = \vec{x}^T\, \underbrace{A\, R_x(-\alpha)\, A^T}_{T_Z}$$

Die Nomenklatur gegennüber dem Skriptum ist geändert :
$\vec{e}_i' \rightarrow$ `e_ip` "p" for primed, in German "Strich"

```
(%i1)  load("vect")$
```

Task to perform

```
(%i3)   axis:[2,-2,1]  /* Rotationaxis */$ X:[0.5,0,0.5]   /* Point to rotate - row vec */$
(%i4)  rotAngle:%pi/3 /* amount of rotation around axis  */$
```

Helpers and Settings

```
(%i6)  row2col(v):=matrix([v[1]],[v[2]],[v[3]])$ unitVec(v):= 1/sqrt(v . v)*v$
(%i7)  R_x(%alpha):=matrix([1,0,0],
                           [0,cos(%alpha),-sin(%alpha)],
                           [0,sin(%alpha),cos(%alpha)])$
(%i8)  R_x:R_x(rotAngle),numer$
(%i11) e_1:[1,0,0]$e_2:[0,1,0]$e_3:[0,0,1]$
(%i12) standardBase:[e_1,e_2,e_3];
```

(%o12) $[[1, 0, 0], [0, 1, 0], [0, 0, 1]]$

```
(%i15) e_1p:unitVec(axis), numer$e_2p:unitVec(express(e_1p ~ e_1))$e_3p:express(e_1p ~ e_2p)$
(%i16) primedBase:[e_1p,e_2p,e_3p]$
```

Generate Matrix A like procedure in the text

```
(%i17) A:genmatrix (lambda ([k, i], primedBase[i] . standardBase[k] ), 3, 3)$
```

Generate the Transformation-Matrix

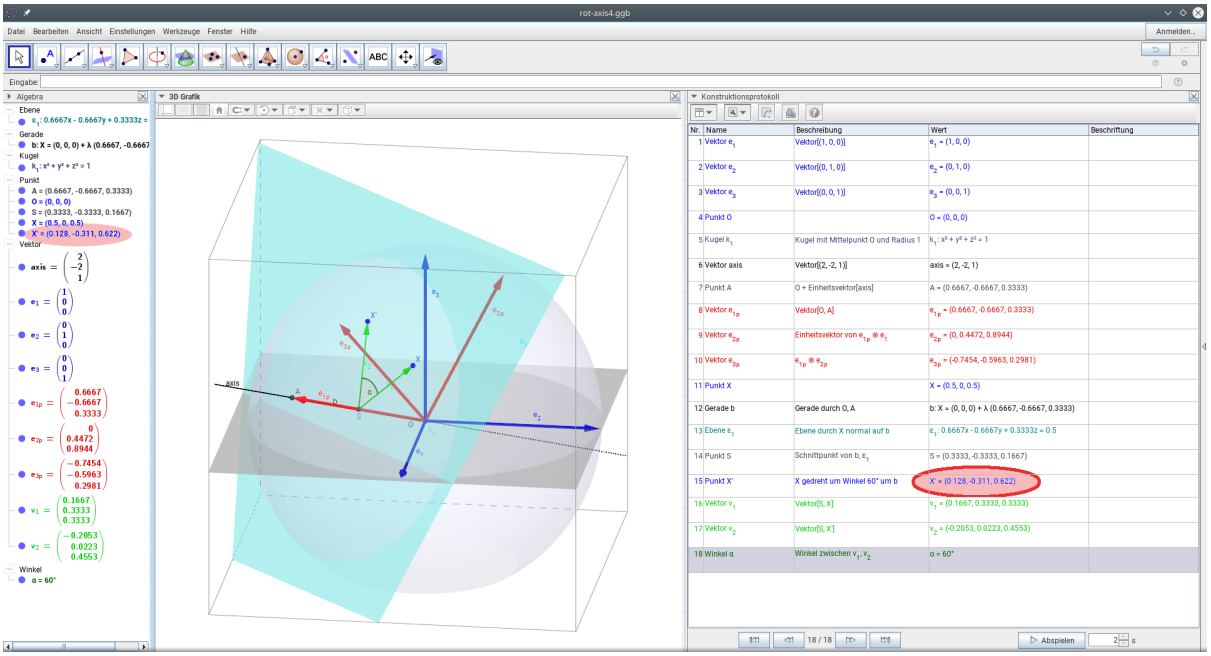
```
(%i18) T:A . R_x . transpose(A)$
```

Column vector was transformed to row vector for space purpose only - compare with Geogebra!

```
(%i19) X_p:transpose(T . row2col(X));
```

(%o19) $(0.1279915320718538 \quad -0.3110042339640731 \quad 0.6220084679281461)$

Hier jetzt die Geogebra-3d Zeichnung dazu



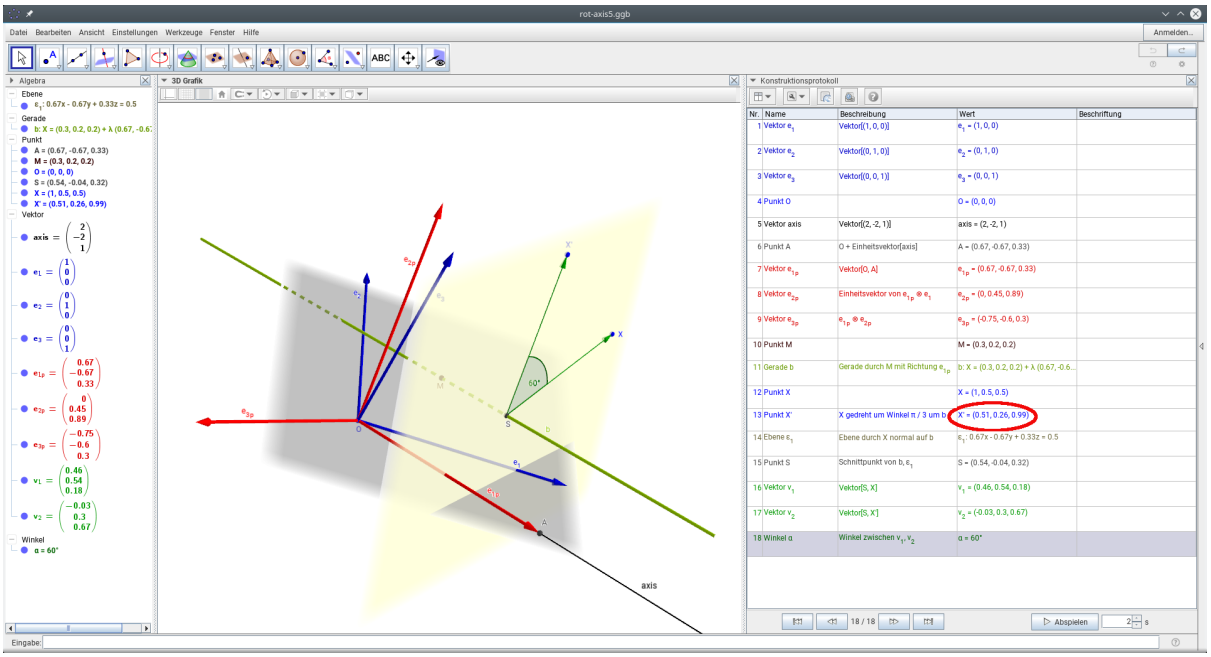
Gehen wir das Konstruktionsprotokoll durch:

1. 1-4 Standardbasisvektoren und Ursprung festlegen
2. 5 Einheitskugel festlegen (jeder Endpunkt eines Orthonormalsystems muss darauf befinden!)
3. 6-7 Endpunkt A von $\vec{e}_1'$ (Rotationsachse) festlegen
4. 8-10 gestrichenes Orthonormalsystem festlegen (Kreuzprodukt mit `Cross[e_{1p}, e_1]`)
5. 11 X festlegen (zu rotierender Punkt um Achse "axis")
6. 12 Rotationsachsen-Gerade b festlegen
7. 13 Ebene ε_1 festlegen: b ist Normale und geht durch X
8. 14 Schneide ε_1 mit b (x'-Achse) : C
9. 15 Drehe X um b um 60°: X' – hier kann man sich von der Richtigkeit unserer Rechnung überzeugen!
10. 16-18 Bestimme den Winkel α zwischen X und X' (Probe)

Hier noch der Link zur Geogebra3d-Datei zum Experimentieren

3 Rotation um eine beliebige Achse außerhalb des Ursprungs

Am einfachsten ist die Sache in Geogebra:
Alles bleibt gleich - nur bestimmen wir einen Punkt M, durch den wir die Rotationsachse durchlegen - das Konstruktionsprotokoll ist analog wie oben. Sowohl M wie auch X sind "freie Punkte" und können in der Konstruktion beliebig verschoben werden!



Die mathematische Theorie um X' zu berechnen ist etwas aufwendiger:
Wir können diesen Fall auf obigen zurückführen, indem wir eine Verschiebung durchführen. Mit Matrizen macht man das indem man auf homogene Koordianten umstellt:

$$(x_1, x_2, x_3) \to (x_1, x_2, x_3, 1)$$

damit lässt sich eine Translation als Matrizenmultiplikation darstellen - die Matrix V verschiebt jeden Vektor um (v_x, v_y, v_z) wie man leicht feststellt:

$$V = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \Rightarrow V \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + v_x \\ y + v_y \\ z + v_z \\ 1 \end{pmatrix} \qquad V^{-1} = \begin{pmatrix} 1 & 0 & 0 & -v_x \\ 0 & 1 & 0 & -v_y \\ 0 & 0 & 1 & -v_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

damit ist der Algorithmus klar: wir verschieben die Lage der Drehachse in den Ursprung (mit V^{-1}) führen dann die Prozedur für "Drehachse im Ursprung" durch und anschl. verschieben wir mit V zurück!

Damit ergibt sich die Transformationsmatrix $T = V\, A\, R_x(\alpha)\, A^T\, V^{-1}$

$$\boxed{\vec{x}_\alpha = \underbrace{V\, A\, R_x(\alpha)\, A^T\, V^{-1}}_{T}\, \vec{x}} \quad \text{wobei } A = \left(\begin{array}{ccc|c} & A_{3\times 3} & & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \quad \text{analog die anderen Matrizen!}$$

Jetzt versuchen wir das in wxMaxima hinzubekommen, außer Verschiebungsmatrix V und Punkt M und anderes X bleibt ja alles gleich!

```
(%i1)  load("vect")$

        Task to perform
(%i3)   axis:[2,-2,1]$  M:[0.3,0.2,0.2]$/* direction and point of Rotationaxis */;
(%i4)  X:[1,0.5,0.5]   /* Point to rotate - row vec */$
(%i5)  rotAngle:%pi/3 /* amount of rotation around axis  */$

        Helpers and Setting
(%i7)  row2col4(v):=matrix([v[1]],[v[2]],[v[3]],[1])$ unitVec(v):= 1/sqrt(v . v)*v$
(%i8)  R_x(%alpha):=matrix([1,0,0],
                           [0,cos(%alpha),-sin(%alpha)],
                           [0,sin(%alpha),cos(%alpha)])$
(%i9)  extend3DimMat(M,v):=block([unit:[0,0,0,1]],
       M:addrow(addcol(M,v), unit))$
(%i10) mk_Trans_Mat(v):=extend3DimMat(diagmatrix(3,1),v)$
(%i11) mk_extended_Mat(M):= extend3DimMat(M,[0,0,0])$
(%i12) R_x:R_x(rotAngle),numer$
(%i15) e_1:[1,0,0]$e_2:[0,1,0]$e_3:[0,0,1]$
(%i16) standardBase:[e_1,e_2,e_3]$
(%i19) e_1p:unitVec(axis), numer$e_2p:unitVec(express(e_1p ~ e_1))$e_3p:express(e_1p ~ e_2p)$
(%i20) primedBase:[e_1p,e_2p,e_3p]$
(%i21) A:genmatrix (lambda ([k, i], primedBase[i] . standardBase[k] ), 3, 3)$

        3-dimensional Transformationmatrix (around the origin)
(%i22) T3_origin: A . R_x . transpose(A)$

        Transformation with explicit Translation - correct result in cart. coordinates
(%i23) X_p:  transpose(T3_origin . (X - M)+M);

(%o23)  (0.5124146010868906  0.256645291237259  0.9884613803007367)

        Do it all with 1 4-dim Transformation - result in homogenous coordinates
(%i24) T4:mk_Trans_Mat(M) . mk_extended_Mat(T3_origin) . mk_Trans_Mat(-M)$
(%i25) X_p:transpose(T4 . row2col4(X));

(%o25)  (0.5124146010868906  0.2566452912372591  0.9884613803007368  1.0)
```

4 Rotation um eine beliebige Achse(2)

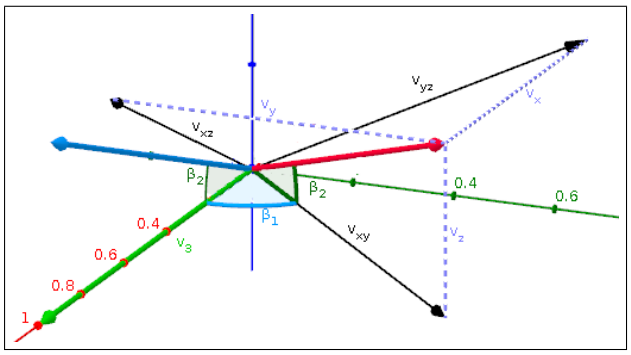
Eine ander Möglichkeit um die Transformationsmatrix zu berechnen, wäre die Zerlegung in Achsenrotationen. Also durch M gehe eine Rotationsachse $\vec{v} = (v_x, v_y, v_z)^t$ mit $|\vec{v}| = 1$. Die Rotation soll einen Winkel α betragen. (Frage: Ist die aus den Rotationen gewonnene Matrix C ident mit der Matrix A^T aus Abschnitt 2 bzw. 3 ?)

Wichtig dabei sind die Projektionen von $\vec{v}$ auf die Koordinatenebenen π_i mit $\vec{v}_{xy} = (v_x, v_y, 0)^t$ und die entsprechendnen beiden anderen. Die Idee ist $\vec{v}$ (roter Pfeil) auf eine Koordinatenachse zu drehen, die Rotation durchführen und zurückdrehen. Nehmen wir als "Zielachse" die x-Achse, dann können wir mit obiger Rechnung direkt vergleichen (rot - x-Achse, grün - y-Achse, blau - z-Achse).

Überlege, dass es mehrere Möglichkeiten gibt dieselbe Achsenrotation in 3 Rotationen zu zerlegen. Außerdem muss $\vec{v}_{xy}$ und damit β_1 nicht existieren ($\vec{v}$ schaut dann in z-Richtung). Die Beschreibung einer Rotation mit Euler-Winkel (hier eigentlich Tait–Bryan Winkel) ist daher nicht unbedingt "das gelbe vom Ei", weil keine Eindeutigkeit (Bijektivität) gewährleistet ist. Wir werden noch andere Möglichkeiten später aufzeigen.

In der Navigation hat man meist das umgekehrte Problem: Ein Flugzeug fliegt einen bestimmten Kurs (x-Achse) horizontal, es soll ein neuer Kurs mit Abweichung β_1 ("Gier-Winkel") geflogen werden, die

"Nase" des Flugzeugs soll sich um β_2 nach oben richten und das Flugzeug soll sich um den Winkel α (Rollwinkel) um die eigene Achse drehen. (Roll-Nick-Gier-Winkel, englisch roll-pitch-yaw angle)



1. Wir drehen im Uhrzeigersinn (math. negativ $\Leftrightarrow$ $v_2 > 0$) um die z-Achse um β_1 (ergibt blauen Pfeil) - dadurch wird $|\vec{v}_{xy}|$ die x-Koordinate des blauen Pfeils!
2. Wir drehen entgegen den Uhrzeigersinn (math. positiv $\Leftrightarrow$ $v_3 \geq 0$) um die y-Achse um β_2 (ergibt grünen Pfeil)
3. Jetzt drehen wir um die x-Achse um α und anschl. machen wir die vorigen Transformationen wieder rückgängig

$$T = \underbrace{R_z^{-1}(\beta_1) \cdot R_y^{-1}(\beta_2)}_{C^{-1}=C^t} \cdot R_x(\alpha) \cdot \underbrace{R_y(\beta_2) \cdot R_z(\beta_1)}_{C}$$

ev. Translation hinzufügen (wie in wxMaxima gemacht)!

```
(%i1)  load("vect")$
```

Task to perform

```
(%i3)  axis:[2,-2,1]$  M:[0.3,0.2,0.2]$/* direction and point of Rotationaxis */;
(%i4)  X:[1,0.5,0.5]   /* Point to rotate - row vec */$
(%i5)  rotAngle:%pi/3 /* amount of rotation around axis  */$
```

Helpers and Setting

```
(%i7)  row2col(v):=matrix([v[1]],[v[2]],[v[3]])$ unitVec(v):= 1/sqrt(v . v)*v$
```

Rotationmatrices and Standardbasis

```
(%i8)  R_x(%alpha):=matrix([1,0,0],
                           [0,cos(%alpha),-sin(%alpha)],
                           [0,sin(%alpha),cos(%alpha)])$
(%i9)  R_y(%alpha):=matrix([cos(%alpha),0,sin(%alpha)],
                           [0,1,0],
                           [-sin(%alpha),0, cos(%alpha)])$
(%i10) R_z(%alpha):=matrix([cos(%alpha),-sin(%alpha),0],
                           [sin(%alpha),cos(%alpha),0],
                           [0,0,1])$
(%i13) e_1:[1,0,0]$e_2:[0,1,0]$e_3:[0,0,1]$
```

Working Horse - determine a set of angles (one of many)

```
(%i14) mk_Transform_Mat(v, angle):=block([%beta1:0, %beta2:0, signBeta1:1, signBeta2:1, C:diagmatrix(3,1) ],
         if ((v[1] = 0) and (v[2] = 0)) then float(R_z(angle))
         else
         block(
            %beta1:float(acos(unitVec([v[1],v[2],0]) . e_1)),
            %beta2:float(acos(unitVec(v) . unitVec([v[1],v[2],0]))),
            if v[2] > 0 then signBeta1: -1,
            if v[3] < 0 then signBeta2: -1,
            C:R_y(signBeta2 * %beta2) . R_z(signBeta1 * %beta1),
            transpose(C) . float(R_x(angle)) . C
         ) )$
(%i15) T:mk_Transform_Mat(axis, rotAngle);
```

$$(\%o15)\quad \begin{pmatrix} 0.7222222222222222 & -0.5108973568170347 & -0.4662391580785149 \\ 0.06645291237259002 & 0.7222222222222222 & -0.6884613803007368 \\ 0.6884613803007369 & 0.466239158078515 & 0.5555555555555554 \end{pmatrix}$$

```
(%i16) X_p:  T .  (X-M)+M;
```

$$(\%o16)\quad \begin{pmatrix} 0.5124146010868906 \\ 0.2566452912372587 \\ 0.9884613803007369 \end{pmatrix}$$

5 Rotation um eine beliebige Achse(3)

Eine andere Möglichkeit der Darstellung einer Drehung (brsonders in der Physik verwendet) ist die durch einen Vektor

$$\vec{\omega} = \theta\hat{n} \quad \text{mit} \quad |\hat{n}| = 1$$

$\hat{n}$ ist dabei die Drehachse, θ entspricht dem Drehwinkel.

Bevor wir die Rodrigues(z)-Formel anschreiben noch eine kurze Vorbemerkung zur Matrixschreibweise eines Vektorprodukts:

$$(\vec{a} \times \vec{b})_i = \underbrace{\varepsilon_{ijk}\, a_j}_{c_{ik}}\, b_k = c_{ik}\, b_k \Rightarrow \vec{a} \times \vec{b} = [\vec{a}]_\times\, \vec{b}$$

Mit obiger Definition ist die Matrix $[\vec{a}]_\times$ zu konstruieren:

$$[\vec{a}]_\times = c_{ik} = \varepsilon_{ijk}\, a_j = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

Diese Matrix erbt die Antisymmetrie vom Levi-Civita Symbol und die Hauptdiagonale verschwindet überall.

Es gilt dann natürlich

$$[\vec{a}]_\times^2\, \vec{b} = \vec{a} \times (\vec{a} \times \vec{b})$$

Nun zur Rodrigues-Formel - (1) in Matrixschreibweise und darunter in Vektorschreibweise

$$R(\hat{n}, \theta) = I + \sin\theta[\hat{n}]_\times + (1 - \cos\theta)[\hat{n}]_\times^2 \tag{1}$$
$$\vec{v}_{rot} = \vec{v}\cos\theta + (\hat{n} \times \vec{v})\sin\theta + \hat{n}(\hat{n} \cdot \vec{v})(1 - \cos\theta) \tag{2}$$
$$\vec{v}_{rot} = \vec{v}_\perp \cos\theta + (\hat{n} \times \vec{v})\sin\theta + \vec{v}_\parallel \tag{3}$$

Hier die Durchführung und "Kontrolle" in wxMaxima:

```
(%i1)  load("vect")$
```

Task to perform

```
(%i3)  axis:[2,-2,1]$  M:[0.3,0.2,0.2]$/* direction and point of Rotationaxis */;
(%i4)  X:[1,0.5,0.5]   /* Point to rotate - row vec */$
(%i5)  rotAngle:%pi/3 /* amount of rotation around axis  */$
```

Helpers and Setting

```
(%i7)  row2col(v):=matrix([v[1]],[v[2]],[v[3]])$ unitVec(v):= 1/sqrt(v . v)*v$
(%i8)  %omega:float(%pi/3)*row2col(unitVec(axis))/* amount and axis of rotation as single vector  */;
```

$$(\%o8) \quad \vec{\omega} = \begin{pmatrix} 0.6981317007977317 \\ -0.6981317007977317 \\ 0.3490658503988658 \end{pmatrix}$$

```
(%i9)  mk_CrossProdMat(v):=block([M:zeromatrix(3,3)],
        M[1,2]: -v[3], M[2,1]: v[3],
        M[1,3]:  v[2], M[3,1]: -v[2],
        M[2,3]: -v[1], M[3,2]: v[1],
        M
       )$
(%i10) n_x:mk_CrossProdMat(float(unitVec(axis) ))$
(%i11) T:float(diagmatrix(3,1) + sin(rotAngle)*n_x + (1 - cos(rotAngle))*(n_x . n_x))$
(%i12) X_p:  T .  (X-M)+M;
```

$$(\%o12) \quad \begin{pmatrix} 0.5124146010868906 \\ 0.256645291237259 \\ 0.9884613803007367 \end{pmatrix}$$

Zuerst gilt es zu zeigen, dass ersten beiden Formeln denselben Sachverhalt darstellen, also

$$R(\hat{n},\theta)\,\vec{v} = \vec{v}_{rot}$$

nachdem der Sinusterm offensichtlich derselbe ist, kürzen wir ihn gleich raus und erhalten

$$\vec{v} + (1-\cos\theta)\,\hat{n}\times(\hat{n}\times\vec{v}) = \vec{v}\cos\theta + \hat{n}(\hat{n}\cdot\vec{v})(1-\cos\theta)$$

$$\cancel{(1-\cos\theta)}\left[\vec{v} + \hat{n}\times(\hat{n}\times\vec{v})\right] = \hat{n}(\hat{n}\cdot\vec{v})\cancel{(1-\cos\theta)} \tag{4}$$

Mit der Graßmann-Identität

$$\vec{a}\times(\vec{b}\times\vec{c}) = (\vec{a}\cdot\vec{c})\vec{b} - (\vec{a}\cdot\vec{b})\vec{c} \tag{5}$$

folgt die Behauptung!

Wir zeigen, dass obige Identität gilt (wir berechnen die j-te Komponente des Ergebnisvektors):

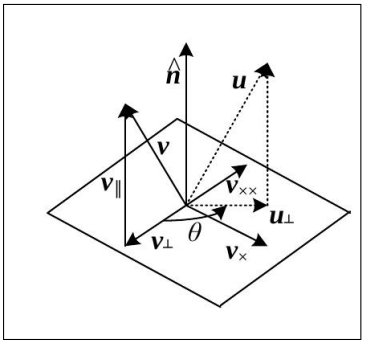
$$\varepsilon_{jk\ell}\,a_k\,\varepsilon_{\ell mn}\,b_m\,c_n = (\delta_{jm}\delta_{kn} - \delta_{jn}\delta_{km})\,a_k\,b_m\,c_n = a_k\,b_j\,c_k - a_k\,b_k\,c_j$$

wobei wir die ε-δ-Beziehung(Wikipedia) benutzt haben.

Jetzt zur Herleitung der Formel von Rodrigues:

Aus Gleichung (4) oben folgt unmittelbar

$$\vec{v} + \vec{v}_{\times\times} = \vec{v}_{\parallel} \quad \text{und da gilt} \quad \vec{v}_{\parallel} = \vec{v} - \vec{v}_{\perp} \Rightarrow \vec{v}_{\times\times} = -\vec{v}_{\perp}$$



Oder man argumentiert über die Rotation:
$\hat{n}\times\vec{v} = \vec{v}_{\times}$ ist die Rotation von $\vec{v}_{\perp}$ um 90°.

$$\vec{v}_{\times} = \hat{n}\times\vec{v} = [\hat{n}]_{\times}\vec{v} \qquad \vec{v}_{\times\times} = \hat{n}\times\vec{v}_{\times} = [\hat{n}]_{\times}^2\vec{v} = -\vec{v}_{\perp}$$
$$\vec{v}_{\parallel} = \vec{v} - \vec{v}_{\perp} = (I - [\hat{n}]_{\times}^2)\vec{v}$$

$\vec{v}_{\perp}$ und $\vec{v}_{\times}$ spannen ein kartesisches Koordinatensystem auf und da gilt
$|\vec{u}_{\perp}| = |\vec{v}_{\perp}| = |\vec{v}_{\times}| \Rightarrow$

$$\vec{u}_{\perp} = \cos\theta\,\vec{v}_{\perp} + \sin\theta\,\vec{v}_{\times} = -\cos\theta\,[\hat{n}]_{\times}^2\vec{v} + \sin\theta\,[\hat{n}]_{\times}\vec{v}$$
$$\vec{v}_{rot} = \vec{u} = \vec{u}_{\perp} + \vec{v}_{\parallel} = -\cos\theta\,[\hat{n}]_{\times}^2\vec{v} + \sin\theta\,[\hat{n}]_{\times}\vec{v} + (I - [\hat{n}]_{\times}^2)\vec{v}$$

Zusammenfassen der letzten Zeile liefert die Behauptung.

Beachte: $|\theta| \ll 1$ geht die Rodrigues-Formel über in:

$$\sin\theta \approx \theta \quad \wedge \quad \cos\theta \approx 1 \Rightarrow R(\hat{n},\theta) = I + \theta[\hat{n}]_{\times} \Rightarrow \boxed{\vec{v}_{rot} = \vec{v} + \omega\times\vec{v}}$$

Eine interessante Möglichkeit unsere Formel zu bestätigen ist folgender Sachverhalt:
Eine Rotation um θ ist dasselbe wie k-Rotationen um θ/k:

$$R(\hat{n},\theta) = \lim_{k\to\infty}\left[R\left(\hat{n},\frac{\theta}{k}\right)\right]^k = \lim_{k\to\infty}\left(I + \frac{\theta}{k}[\hat{n}]_{\times}\right)^k = \exp([\omega]_{\times}) = I + \frac{\theta[\hat{n}]_{\times}}{1} + \frac{\theta^2[\hat{n}]_{\times}^2}{2} + \frac{\theta^3[\hat{n}]_{\times}^3}{3!} + \ldots$$

mit

$$[\hat{n}]_{\times}^{k+2} = -[\hat{n}]_{\times}^k$$

folgt unmittelbar wieder die Rodrigues-Formel!

6 Einheits-Quaternionen(4)

Quaternionen lassen sich als Verallgemeinerung von $\mathbb{C}$ als auch von $\mathbb{R}^3$ auffassen:

$$q := q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3 = (q_0, q_1, q_2, q_3) = \underbrace{q_0}_{Skalarteil} + \underbrace{\vec{q}}_{Vektorteil} = (q_0, \vec{q})$$

Die Addition ist isomorph zu der in $\mathbb{R}^4$:

$$p + q = (p_0 + q_0) + (\vec{p} + \vec{q}) = (p_0 + q_0, \vec{p} + \vec{q})$$

Für die Multiplikation standen die kartesischen Einheitsvektoren (mit dem Kreuzprodukt im Hinterkopf) und $\mathbb{C}$ Pate:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

daraus folgt sofort die Nichkommutativität

$$\mathbf{ij} = \mathbf{k} = -\mathbf{ji} \quad \text{und analoge Ausdrücke}$$

Berechnet man das Produkt zweier Quaternionen ist dies ein längerer Ausdruck, der allerdings mit Skalar- und Vektorprodukt etwas kompakter geschrieben werden kann:

$$pq = (p_0 + \mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3)\,(q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3) = \ldots = (p_0q_0 - \vec{p}\cdot\vec{q}, p_0\vec{q} + q_0\vec{p} + \vec{p}\times\vec{q})$$

Wieder sieht man, dass durch das Vektorprodukt die Kommutativität verletzt wird. Untersucht man die beiden Rechenoperationen genauer ergibt sich als Struktur ein Nicht-Abelscher Ring.

Mit der komplexen Konjugation $q^* := q_0 - \vec{q}$ lässt sich eine Norm $|q|$ festlegen:

$$q^*q = (q_0, -\vec{q})(q_0, \vec{q}) = q_0^2 + \vec{q}\cdot\vec{q} = q_0^2 + q_1^2 + q_2^2 + q_3^2 = qq^* =: |q|^2$$

Man kann mit obiger Multiplikation leicht zeigen:

$$(pq)^* = q^*p^* \Rightarrow |pq|^2 = (pq)\,(pq)^* = (pq)\,(q^*p^*) = p|q|^2p^* = |p|^2|q|^2$$

Mit der Definition des Inversen ergibt sich

$$q^{-1} := \frac{q^*}{|q|^2} \Rightarrow qq^{-1} = q^{-1}q = 1$$

Für Einheitsquaternionen gilt also:

$$q^{-1} = q^* \quad \text{und mit der Beziehung} \quad |q|^2 = 1 = q_0^2 + \vec{q}\cdot\vec{q} = q_0^2 + \|\vec{q}\|^2$$

daher gibt es für jedes q ein $\theta \in [0, \pi]$ und einen Einheitsvektor $\hat{n} \in \mathbb{R}^3$ sodass gilt:

$$q = (s, \vec{q}) = (\cos\theta, \sin\theta\hat{n})$$

Damit können wir jetzt den Rotationsoperator festlegen:

$$\boxed{R_q(\vec{v}) = q\,\vec{v}\,q^{-1} = q\,\vec{v}\,q^* = (s, \vec{q})\,(0, \vec{v})\,(s, -\vec{q})} \qquad (s, \vec{q}) = (\cos\theta, \sin\theta\hat{n})$$

Wir zeigen, dass $R_q(\vec{v})$ eine Rotation des Vektors $\vec{v}$ um die Achse $\hat{n}$ mit Betrag 2θ darstellt:

$$\begin{aligned}
R_q(\vec{v}) &= (s, \vec{q})\,(0, \vec{v})\,(s, -\vec{q}) = (s, \vec{q})\,(\vec{v}\vec{q}, s\vec{v} + \vec{q}\times\vec{v}) = \\
&= (s\vec{v}\vec{q} - \vec{q}\cdot(s\vec{v} + \vec{q}\times\vec{v}), s(s\vec{v} + \vec{q}\times\vec{v}) + (\vec{v}\vec{q})\vec{q} + \vec{q}\times(s\vec{v} + \vec{q}\times\vec{v})) = \\
&= (0, s^2\vec{v} + 2s\vec{q}\times\vec{v} + (\vec{v}\vec{q})\vec{q} + \vec{q}\times\vec{q}\times\vec{v}) = \text{ für den letzten Ausdruck verwenden wir (7)} \\
&= (0, \vec{v}(s^2 - \vec{q}\cdot\vec{q}) + (\vec{v}\vec{q})\vec{q} + 2s\vec{q}\times\vec{v})
\end{aligned}$$

Mit $(s, \vec{q}) = (\cos\theta, \sin\theta\hat{n})$ ergibt sich die Rodrigues-Formel (4) für 2θ!

Auf weitergehende Betrachtungen sei hier verzichtet (Differentiation, Integration, Exponential- und Logarithmusfunktion, algebraische Eigenschaften, usw.) .

Fazit:
Sind Translationen im Spiel haben die Matrizen leichte Vorteile, da sie ebenfalls über homogene Koordinaten als Matrizen darstellbar sind, ansonsten sind Quaternionen wegen ihrer reichhaltigen algebraischen Struktur beinahe unschlagbar.