HOCHSCHULE FURTWANGEN UNIVERSITY HFU

Fakultät Mechanical and Medical Engineering

Hausarbeit

im Fach

Dynamische Auslegung von Werkzeugmaschinen, Roboter und Bewegungsachsen

im

Sommersemester 2024

von

Daniel Medina Velez (277542)

Caleb Kieninger (270001)

Studiengang

Advanced Precision Engineering (APE)

Betreuer:

Prof. Dr.- Ing. Dipl.-Ing. Betriebswirt

Gunter Ketterer

Fakultät Mechanical and Medical Engineering

Villingen-Schwenningen, den 28. März 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Aufgabe 1: Einleitung/Aufgabenstellung

In der Hausarbeit für das Fach „dynamische Auslegung von Werkzeugmaschinen, Roboter- und Bewegungsachsen" wird ein SCARA-Roboter ausgelegt. Gegeben ist ein vierachsiger Scara-Roboter, dargestellt in Grundstellung $\phi_0=0$; $\phi_1=0$; $\phi_2=0$ $\phi_4=0$und H_3 in einer Zwischenstellung. Er besitzt 3 rotatorische und eine rot. - translatorische Achse. Anders als in der originalen Aufgabenstellung liegt das AKS des TCP mit der Z-Achse nach oben und mit der X-Achse nach links. Diese Änderung wird bei der Grundstellung und bei den Montagekräften beachtet.

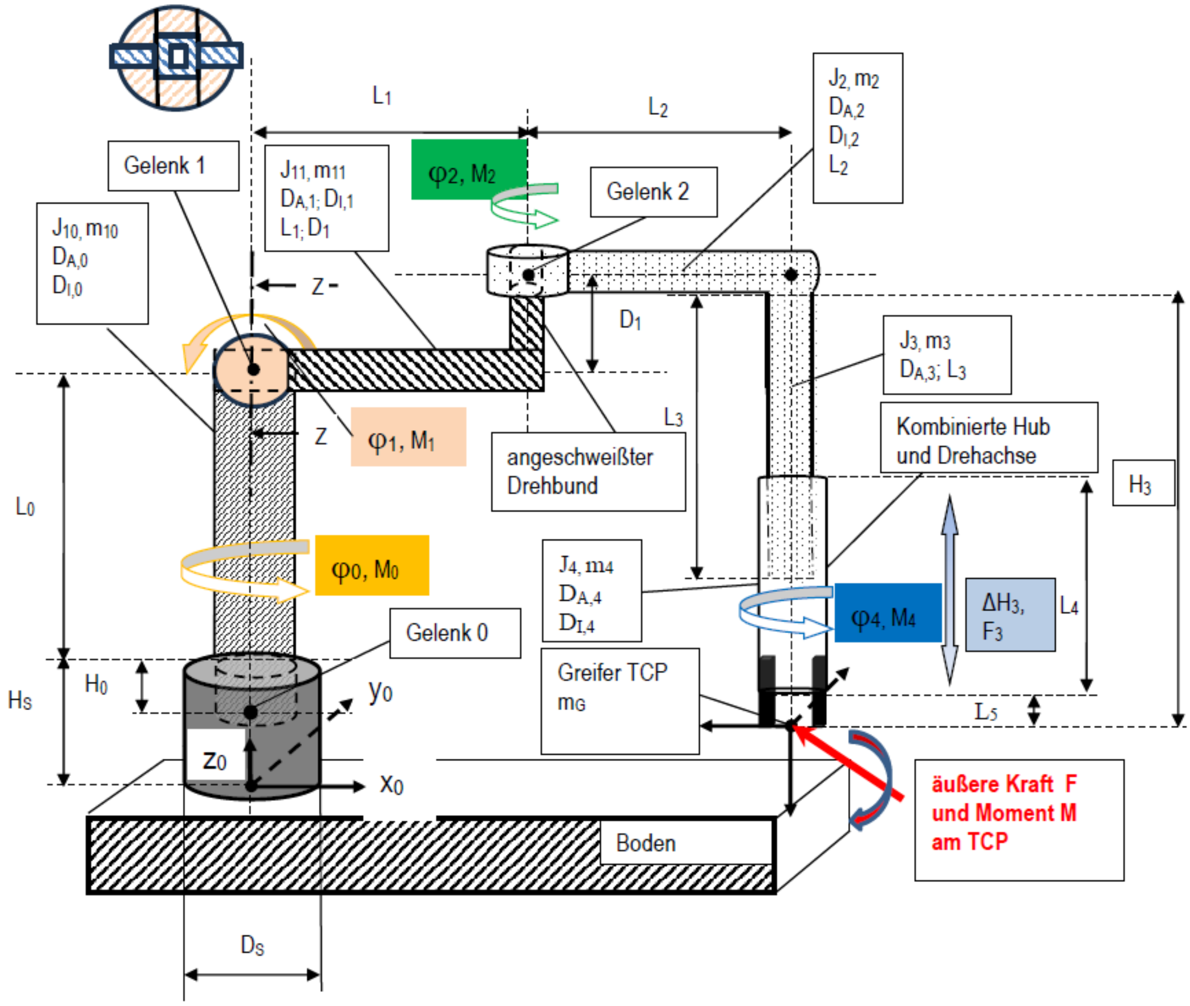


Abbildung 1: Scara-Roboter in Grundstellung – schematische Darstellung (nicht maßstabsgetreu)

Positive Verfahrrichtung:

<u>Als geometrische Maße gelten:</u>
$L_0 = 800$mm; $H_0=100$mm; $L_1 = 400$mm; $L_2 = 500$mm; $L_3 = 630$mm; $L_4 = 600$mm; $L_5 = 50$mm, $D_1=110$mm, $H_S=200$mm, $D_S=300$mm

<u>Dreh-, Schwenk- und Hubbereiche:</u>
Achse 0: $-150° < \phi_0 < 150°$; $D_{A,0} = 200$mm, $D_{I,0} = 150$mm;

Achse 1: $0° < \phi_1 < 90°$; $D_{A,1} = 100mm$, $D_{I,1} = 80mm$
Gelenk: 1: Voll-Kugel mit dem Durchmesser $D_{G,1} = 200mm$

Achse 2: $-170° < \phi_2 < 170°$; $D_{A,2} = 100mm$, $D_{I,2} = 80mm$, $D_{A,3} = 80mm$
Gelenk 2: $D_{I,G2}=100mm$ $D_{A,G2}=140mm$

Achse 4:
Gelenk: Drehachse $-180° < \phi_4 < 180°$;
Hubachse: $\Delta H_3=400mm$ mit: $H_{3min}=700mm \leq H_3 \leq H_{3,max}=1100mm$; $D_{A,4}= 140mm$, $D_{I,4}=80mm$
00mm, $D_S=300mm$

a) Arbeitsraum

In Abbildung 2 ist Roboter in der Grundstellung mit ausgefahrenem Hub. Mit diesen Abmessungen ist der Arbeitsraum ermittelt.

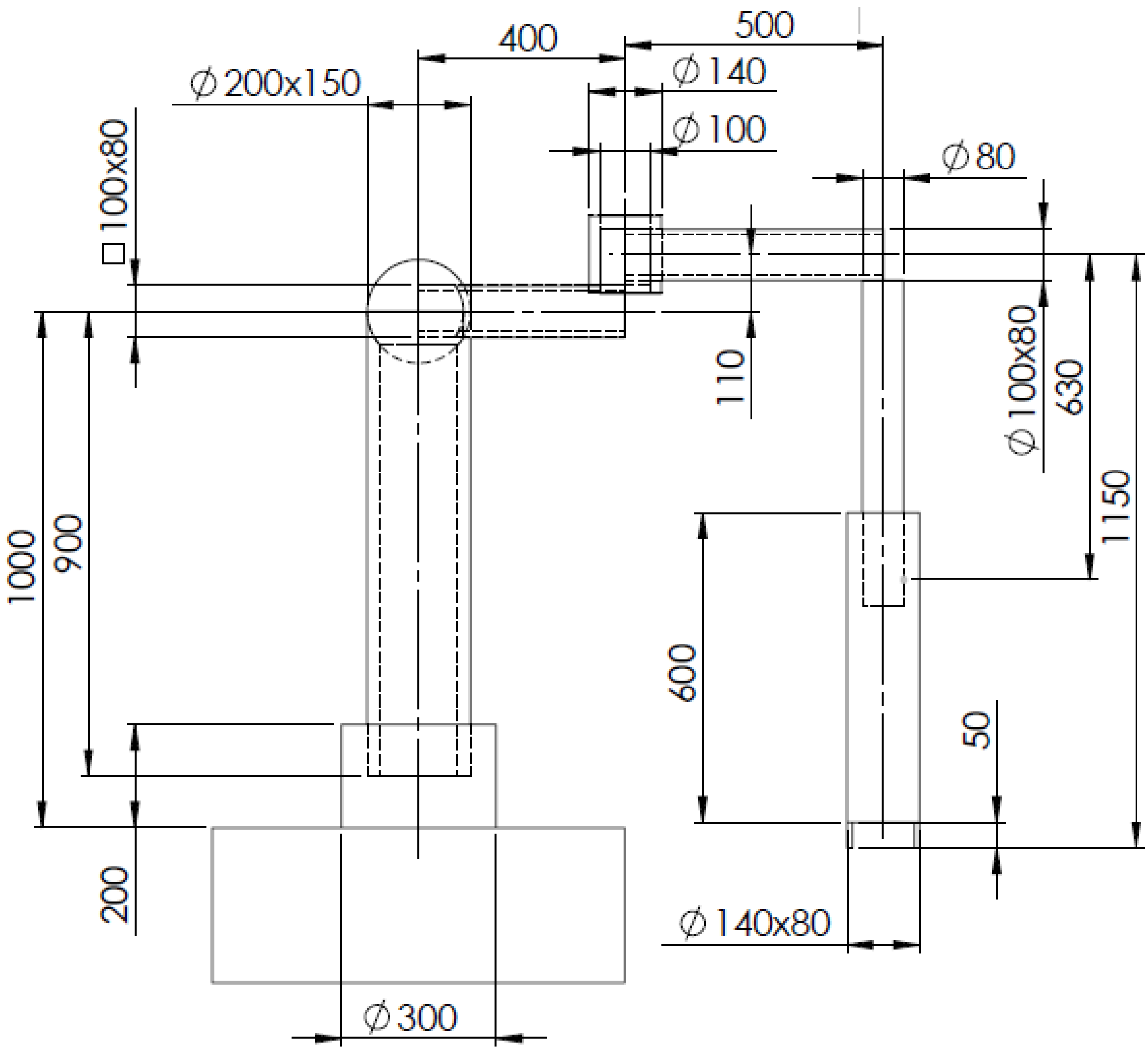


Abbildung 2: Grundstellung

In Abbildung 3 ist der Arbeitsraum in x_0 - z_0 - Ebene dargestellt. Der Arbeitsraum ist „nierenförmig“. Die linke Seite unterschiedet sich von der rechten Seite durch die Winkelbegrenzung in ϕ_0. Es entsteht eine Kollision mit dem Boden. Der Hub sollte um mindestens 40mm verkürzt werden.

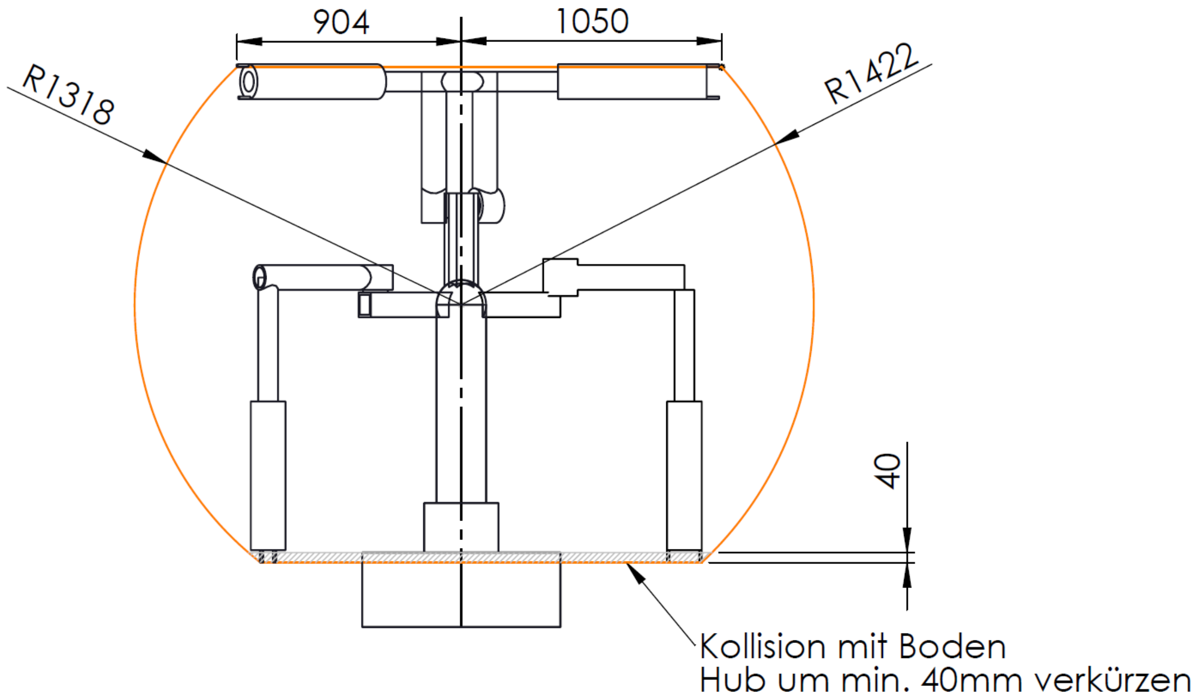


Abbildung 3: Arbeitsraum X0-Z0 Ebene

In Abbildung 4 ist der Arbeitsraum in x_0 - y_0 - Ebene dargestellt. Der TCP kann mit einem Radius von 1389mm fast den kompletten Arbeitsraum Kreisförmig abfahren. Lediglich 2 x 30° nicht. Hier schwenkt der TCP mit R1151mm ein. In Abbildung 5 ist eine mögliche Kollision zu sehen. ϕ_2 sollte auf 154° beschränkt werden, sonst kollidiert die 4. Achse mit dem Sockel.

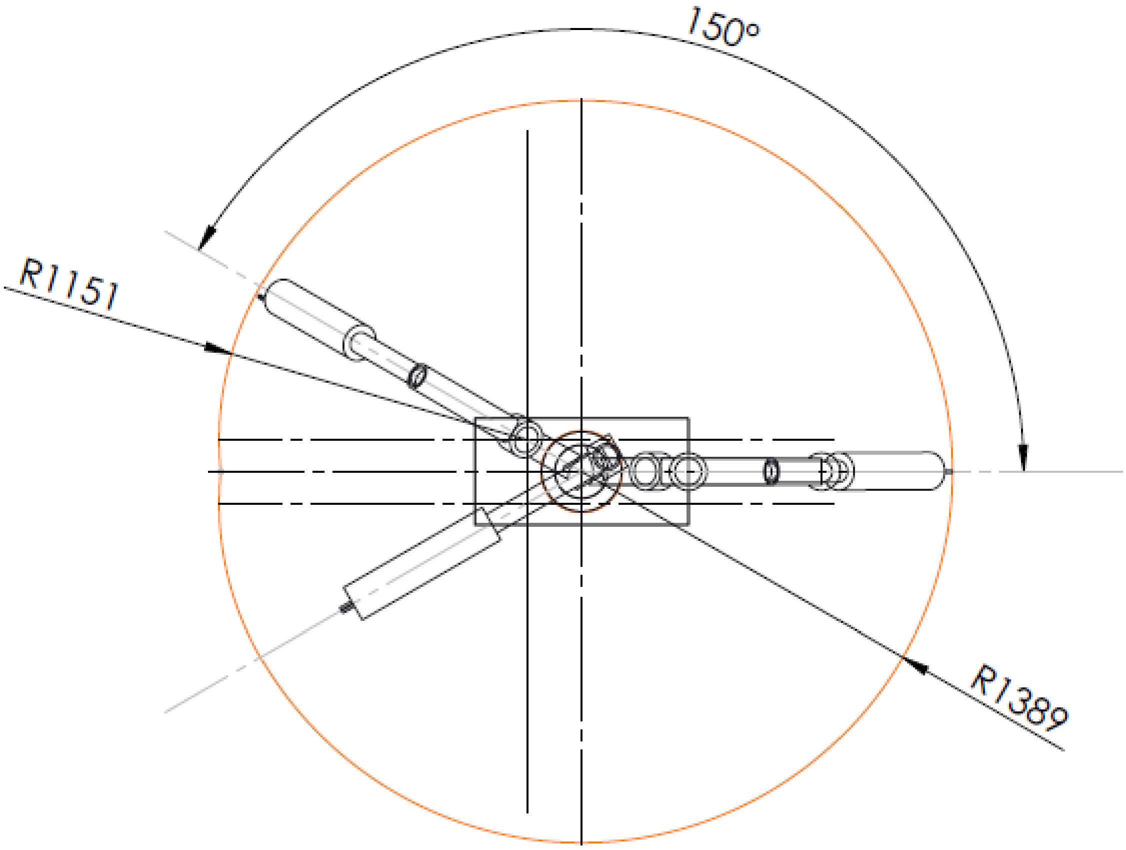
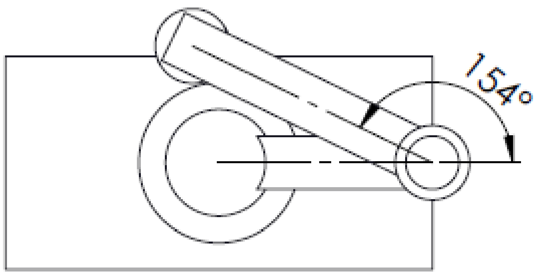


Abbildung 4: Arbeitsraum X0-Y0 Ebene



Kollision bei φ2 >154° mit dem Sockel

φ2 muss auf ± 154° beschränkt werden

Abbildung 5: Kollision Ansicht in X0-Y0 Ebene

In Abbildung 6 ist der Arbeitsraum in der y_0 - z_0 – Ebene dargestellt. Dieser ist symmetrisch um die zentrale Achse. Auch hier kommt es zur Kollision des TCP mit dem Boden. Die Form lässt sich beschreiben als ein Kreis mit abgeflachten Seiten.

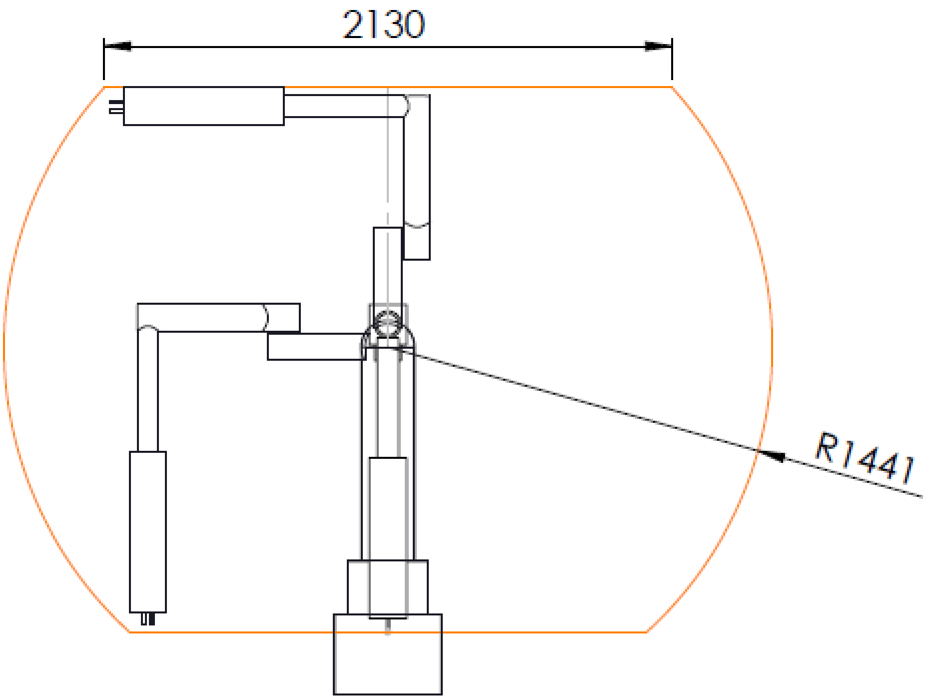


Abbildung 6: Arbeitsraum in Y0-Y0 Ebene

b) Schematisches Ersatzbild und Koordinatensysteme

In der nachfolgenden Abbildung 7 ist das schematische Ersatzschaltbild des Roboters sowie die Koordinatensysteme nach Denavit Hartenberg dargestellt. Es sind zwei Hilfskoordinatensysteme notwendig.

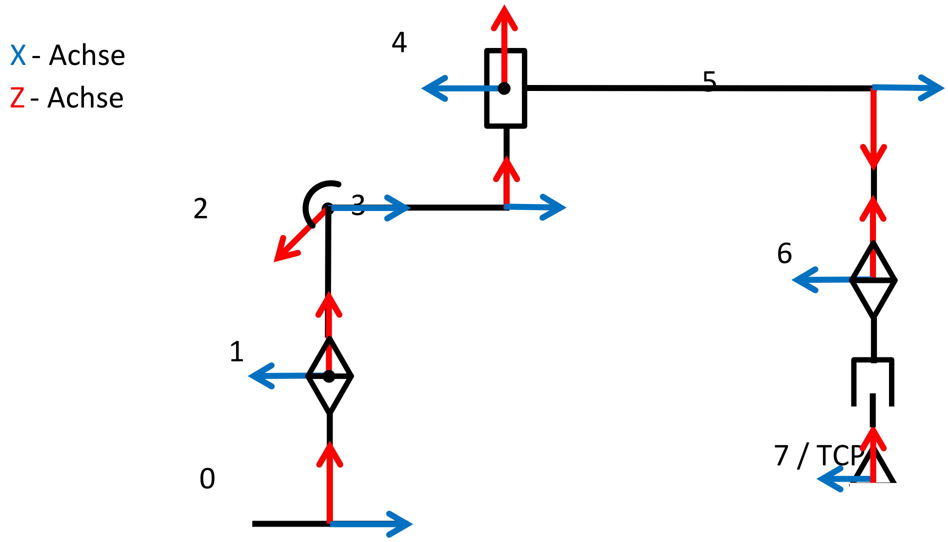


Abbildung 7: Koordinatensystem nach Denavit Hartenberg

c) Einzeltransformationsmatrizen $^0A_1$ – $^6A_7$

Mit Hilfe der Denavit-Hartenberg-Konvention (DH-Konvention) können die in Tabelle 1 gezeigten Parameter bestimmt werden. Die einzelnen AKS lassen sich mit Hilfe von zwei Drehungen (α, ϕ) und zwei Verschiebungen (L, D) eindeutig zu ihrem vorhergehenden AKS herleiten.

Tabelle 1 Denavit-Hartenberg-Konvention

AKS Nr. i	α_i	L_i	D_i	ϕ_i
1	0	0	$H_S - H_0$	π
2	$\pi/2$	0	$H_0 + L_0$	$\pi + \phi_0$
3	$-\pi/2$	L_1	0	ϕ_1
4	0	0	D_1	π
5	π	L_2	0	$\pi + \phi_2$
6	π	0	$(D_{A2}/2) + L_3$	π
7 / TCP	0	0	$-(H_3 - L_3)$	ϕ_4

Mit der folgenden homogenen Transformationsmatrix (siehe Abbildung 8) können die Einzeltransformationen in eine Matrizenschreibweise überführt werden. (Ketterer, Dynamische Auslegung von Werkzeugmaschinen und Roboterachsen / Teil 2 Jacobi-Lagrange Skript)

$$\overline{A} = \mathrm{rot}(\varphi_z)\cdot \mathrm{trans}(D_z)\cdot \mathrm{trans}(L_x)\cdot \mathrm{rot}(\alpha_x) = \begin{bmatrix} C_\varphi & -C_\alpha \cdot S_\varphi & S_\alpha \cdot S_\varphi & L\cdot C_\varphi \\ S_\varphi & C_\alpha \cdot C_\varphi & -S_\alpha \cdot C_\varphi & L\cdot S_\varphi \\ 0 & S_\alpha & C_\alpha & D \\ 0 & 0 & 0 & 1 \end{bmatrix} = \overline{DH} \; .$$

Abbildung 8: Transformationsmatrix

Eingesetzt ergeben sich die Matrizen $^0A_1$ (A01) bis $^6A_7$ (A67). Die einzelnen Matrizen, auch Überführungsmatrizen genannt, bilden jeweils die räumliche Positionierung von einem AKS zum nächsten AKS.

```
A01 =
[-1,  0, 0,          0]
[ 0, -1, 0,          0]
[ 0,  0, 1, H_s - H_0]
[ 0,  0, 0,          1]


A12 =
[-cos(phi_0), 0, -sin(phi_0),         0]
[-sin(phi_0), 0,  cos(phi_0),         0]
```

```
[          0, 1,             0, H_0 + L_0]
[          0, 0,             0,         1]


A23 =
[cos(phi_1),  0, -sin(phi_1), L_1*cos(phi_1)]
[sin(phi_1),  0,  cos(phi_1), L_1*sin(phi_1)]
[         0, -1,           0,              0]
[         0,  0,           0,              1]


A34 =
[-1,  0, 0,   0]
[ 0, -1, 0,   0]
[ 0,  0, 1, D_1]
[ 0,  0, 0,   1]


A45 =
[-cos(phi_2), -sin(phi_2),  0, -L_2*cos(phi_2)]
[-sin(phi_2),  cos(phi_2),  0, -L_2*sin(phi_2)]
[          0,           0, -1,               0]
[          0,           0,  0,               1]


A56 =
[-1, 0,  0,            0]
[ 0, 1,  0,            0]
[ 0, 0, -1, D_A2/2 + L_3]
[ 0, 0,  0,            1]


A67 =
[cos(phi_4), -sin(phi_4), 0,         0]
[sin(phi_4),  cos(phi_4), 0,         0]
[         0,           0, 1, L_3 - H_3]
[         0,           0, 0,         1]
```

d) Gesamttransformation

Die Gesamttransformation wird durch Multiplikation der Einzeltransformationen erzeugt. Durch diese kann die Position und Lage des TCP genau bestimmt werden.

$$^{0}[\mathrm{T}]_{7} = {}^{0}[\mathrm{A}]_{1} * {}^{1}[\mathrm{A}]_{2} * {}^{2}[\mathrm{A}]_{3} * {}^{3}[\mathrm{A}]_{4} * {}^{4}[\mathrm{A}]_{5} * {}^{5}[\mathrm{A}]_{6} * {}^{6}[\mathrm{A}]_{7}$$

Die Gesamttransformation wurde in Matlab durchgeführt. Durch die Matrizenmultiplikation ergibt sich die folgende Gesamttransformationsmatrix.

$$^{0}[\mathrm{T}]_{7} = A07$$

```
A07 =
erste Zeile:
[  cos(phi_4)*(sin(phi_0)*sin(phi_2) - cos(phi_0)*cos(phi_1)*cos(phi_2)) +
```

```
sin(phi_4)*(cos(phi_2)*sin(phi_0) + cos(phi_0)*cos(phi_1)*sin(phi_2)),
cos(phi_4)*(cos(phi_2)*sin(phi_0) + cos(phi_0)*cos(phi_1)*sin(phi_2)) -
sin(phi_4)*(sin(phi_0)*sin(phi_2) - cos(phi_0)*cos(phi_1)*cos(phi_2)),
 -cos(phi_0)*sin(phi_1),
L_1*cos(phi_0)*cos(phi_1) - D_1*cos(phi_0)*sin(phi_1) +
cos(phi_0)*sin(phi_1)*(D_A2/2 + L_3) + cos(phi_0)*sin(phi_1)*(H_3 - L_3) -
L_2*sin(phi_0)*sin(phi_2) + L_2*cos(phi_0)*cos(phi_1)*cos(phi_2)]

Zweite Zeile:
[- cos(phi_4)*(cos(phi_0)*sin(phi_2) + cos(phi_1)*cos(phi_2)*sin(phi_0)) -
sin(phi_4)*(cos(phi_0)*cos(phi_2) - cos(phi_1)*sin(phi_0)*sin(phi_2)),
sin(phi_4)*(cos(phi_0)*sin(phi_2) + cos(phi_1)*cos(phi_2)*sin(phi_0)) -
cos(phi_4)*(cos(phi_0)*cos(phi_2) - cos(phi_1)*sin(phi_0)*sin(phi_2)),
 -sin(phi_0)*sin(phi_1),
L_1*cos(phi_1)*sin(phi_0) + L_2*cos(phi_0)*sin(phi_2) - D_1*sin(phi_0)*sin(phi_1)
+ sin(phi_0)*sin(phi_1)*(D_A2/2 + L_3) + sin(phi_0)*sin(phi_1)*(H_3 - L_3) +
L_2*cos(phi_1)*cos(phi_2)*sin(phi_0)]

Dritte Zeile:
[ sin(phi_1)*sin(phi_2)*sin(phi_4) - cos(phi_2)*cos(phi_4)*sin(phi_1),
cos(phi_2)*sin(phi_1)*sin(phi_4) + cos(phi_4)*sin(phi_1)*sin(phi_2),
cos(phi_1),
H_s + L_0 + D_1*cos(phi_1) - cos(phi_1)*(D_A2/2 + L_3) - cos(phi_1)*(H_3 - L_3) +
L_1*sin(phi_1) + L_2*cos(phi_2)*sin(phi_1)]

Vierte Zeile:
[0, 0, 0, 1]
```

Setzt man nun die gegebenen Zahlenwerte ein und unter Annahme aller Winkel = 0 mit maximalem Hub ergibt sich folgende Matrix:

```
T = 0 A 7
[ -1.0000   -0.0000    0.0000  900.0000]
[ 0.0000    -1.0000    0        -0.0000]
[ 0.0000    0          1.0000  -40.0000]
[ 0         0          0        1.0000]
```

Im Verglich zu AKS 0 ist bei AKS 7 die x- und y- Achse um 180° gedreht. Die z-Achse ist in ihrer Richtung gleichgeblieben. L1 = 400mm + L2 = 500mm = 900mm.
In positive z_0 Richtung: HS=200mm + L_0 = 800mm + D1=110mm = 1110mm
In negativer z_0 Richtung: $D_{A2}/2$ = 50mm + H_{3max} = 1100mm = 1150mm
Somit ergibt sich eine Differenz von -40mm. T07 = A07 ist somit überprüft.

e) Rückwärtstransformation

In der Regel soll ein Roboter eine bestimmte Arbeit ausführen. Das bedeutet, dass der TCP eine vorgegebene Position (X, Y, Z) in einem vorgegebenen Winkel (Rotation um X, Y ,Z) anfahren soll. Die allgemeine Matrixdarstellung des TCP ergibt sich durch:

$$T = \begin{bmatrix} n_x & u_x & a_x & p_x \\ n_y & u_y & a_y & p_y \\ n_z & u_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Um diese Position anfahren zu können, müssen die Gelenkvariablen ϕ_0, ϕ_1, ϕ_2, ϕ_4 und H_3 bestimmt werden. Dies ist mit Hilfe der Rückwärtstransformation möglich.

Bei der Rückwärtstransformation ist ein direktes Auflösen der Gleichungen zum Ermitteln der unbekannten Gelenkvariablen nicht möglich. Diese lassen sich über einen Koeffizientenvergleich ermitteln.

Das Verfahrensprinzip besteht prinzipiell aus vier Punkten: (Ketterer, Dynamische Auslegung von Werkzeugmaschinen und Roboterachsen / Teil 2 Jacobi-Lagrange Skript)

1. Erstellen der Gesamttransformation als Produkt der DH-Matrizen.

2. Suchen von lösbaren Bestimmungsgleichungen für die gesuchten Gelenkvariablen per Koeffizientenvergleich zwischen den Matrizendarstellungen. Z.B. Beziehungen für cos und sin der Drehwinkel, die sich über arctan-Funktionen auflösen lassen.

3. Um weitere Bestimmungsgleichungen auffinden zu können, werden beide Seiten der Matrixgleichungen mit den Inversen der einzelnen DH-Matrizen multipliziert und führt mit diesen Schritt 2 erneut durch.

4. Schritt 3 wird solange wiederholt, bis genügend Bestimmungsgleichungen gefunden wurden.

An diesem Vorgehen orientiert, werden alle möglichen Matrizen Multiplikationen durchgeführt. Die Ergebnisse aus den Command Window werden in einer separaten Exceltabelle ausgewertet und die entsprechenden Elemente der Matrizen gleichgesetzt. (Die Exceltabelle ist auf dem beigelegten Stick. Benennung: Koeffizientenvergleich)

```
%disp('0. Koeff. Vgl.')
Atemp1 = T;
Atemp2 = A07;

%disp('1. Koeff. Vgl.')
Atemp1 = inv(A01)* T;
Atemp2 = A12 * A23 * A34 * A45 * A56 * A67;

%disp('2. Koeff. Vgl.')
Atemp1 = inv(A12) * inv(A01)* T;
Atemp2 = A23 * A34 * A45 * A56 * A67;

%disp('3. Koeff. Vgl.')
Atemp1 = inv(A23) * inv(A12) * inv(A01)* T;
Atemp2 = A34 * A45 * A56 * A67;
```

```
%disp('4. Koeff. Vgl.')
Atemp1 = inv(A34) * inv(A23) * inv(A12) * inv(A01)* T;
Atemp2 = A45 * A56 * A67;

%disp('5. Koeff. Vgl.')
Atemp1 = inv(A45) * inv(A34) * inv(A23) * inv(A12) * inv(A01)* T;
Atemp2 = A56 * A67;
```

Nachfolgend beispielhaft der 2. Koeffizienten Vergleich:

Tabelle 2 2. Koeffizientenvergleich

[(nx*cos(phi_0)) + (ny*sin(phi_0))	=	[cos(phi_1)*sin(phi_2)*sin(phi_4) - cos(phi_1)*cos(phi_2)*cos(phi_4)
[nz	=	[sin(phi_1)*sin(phi_2)*sin(phi_4) - cos(phi_2)*cos(phi_4)*sin(phi_1)
[(nx*sin(phi_0)) - (ny*cos(phi_0))	=	[cos(phi_2)*sin(phi_4) + cos(phi_4)*sin(phi_2)
(ux*cos(phi_0)) + (uy*sin(phi_0))	=	cos(phi_1)*cos(phi_2)*sin(phi_4) + cos(phi_1)*cos(phi_4)*sin(phi_2)
uz	=	cos(phi_2)*sin(phi_1)*sin(phi_4) + cos(phi_4)*sin(phi_1)*sin(phi_2)
(ux*sin(phi_0)) - (uy*cos(phi_0))	=	cos(phi_2)*cos(phi_4) - sin(phi_2)*sin(phi_4)
(ax*cos(phi_0)) + (ay*sin(phi_0))	=	-sin(phi_1)
az	=	cos(phi_1)
(ax*sin(phi_0)) - (ay*cos(phi_0))	=	0
(px*cos(phi_0)) + (py*sin(phi_0))]	=	L_1*cos(phi_1) - D_1*sin(phi_1) + sin(phi_1)*(D_A2/2 + L_3) + sin(phi_1)*(H_3 - L_3) + L_2*cos(phi_1)*cos(phi_2)]
pz - L_0 - H_s]	=	D_1*cos(phi_1) - cos(phi_1)*(D_A2/2 + L_3) - cos(phi_1)*(H_3 - L_3) + L_1*sin(phi_1) + L_2*cos(phi_2)*sin(phi_1)]
(px*sin(phi_0)) - (py*cos(phi_0))]	=	-L_2*sin(phi_2)]

Phi_0:

2. Koeffizientenvergleich; Element 3:3

Zeile 9 wird durch cos(phi_0) geteilt und entsprechend umgestellt:

Phi_0 = arctan(ay / ax)

Phi_0 führt so leider zu keinem guten Ergebnis, da in der Position des Einpressens eine Division mit „0“ zu einem undefinierten Ergebnis führt.

Es wurden keine weiteren verwertbaren Gleichungen gefunden. Phi_0 wird über geometrische Beziehungen berechnet. Es ist zu sehen, dass zwei Stellungen des Roboters möglich sind. Die Berechnung der Winkel erfolgt mit dem Kosinussatz.

Phi_0 entspricht der orangenen Position; phi_01 entspricht der grünen Position

```
phi_0 = atan2(py,px) + acos((L_1^2+px^2+py^2-L_2^2)/(2*L_1*sqrt(px^2+py^2)))
phi_01 = atan2(py,px) - acos((L_1^2+px^2+py^2-L_2^2)/(2*L_1*sqrt(px^2+py^2)))
```

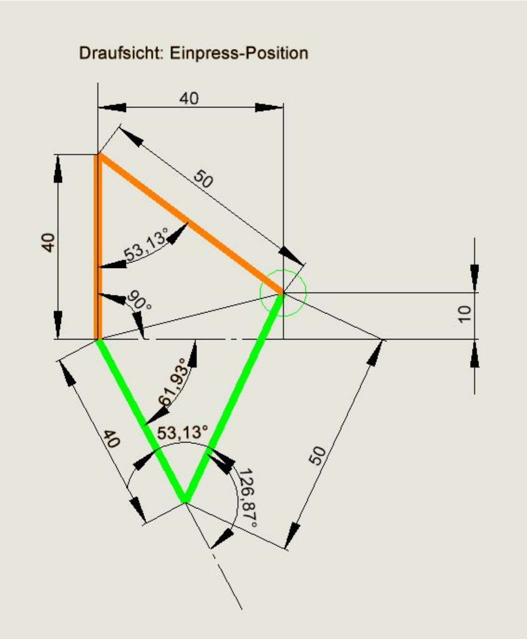


Abbildung 9 Draufsicht Einpressen

Phi_1:

3. Koeffizientenvergleich; Element 3:1

[(nz*cos(phi_1)) - (ny*sin(phi_0)*sin(phi_1)) - (nx*cos(phi_0)*sin(phi_1))	=	0

```
phi_1 = atan2(nz,(nx*cos(phi_0)+ny*sin(phi_0)))
phi_11 = atan2(nz,(nx*cos(phi_01)+ny*sin(phi_01)))
```

phi_11 ergibt die zweite Lösung. Gerechnet mit phi_01

Phi_2:

3. Koeffizientenvergleich; Element 4:1 & 4:2

(sin(phi_1)*(H_0 - H_s)) - (sin(phi_1)*(H_0 + L_0)) - L_1 + (pz*sin(phi_1)) + (px*cos(phi_0)*cos(phi_1)) + (py*cos(phi_1)*sin(phi_0))	=	L_2*cos(phi_2)]
(py*cos(phi_0)) - (px*sin(phi_0))	=	L_2*sin(phi_2)]

Wird nun die zweite durch die erste Zeile geteilt, ergibt sich auf der rechten Seite tan(phi_2).

```
phi_2 = atan2((-px*sin(phi_0) + py*cos(phi_0)),(sin(phi_1)*(-H_s-L_0) - …
L_1+px*cos(phi_0)*cos(phi_1)+py*sin(phi_0)*cos(phi_1)+pz*sin(phi_1)))

phi_21 = atan2((-px*sin(phi_01) + py*cos(phi_01)),(sin(phi_1)*(-H_s-L_0)-…
L_1+px*cos(phi_01)*cos(phi_1)+py*sin(phi_01)*cos(phi_1)+pz*sin(phi_1)))
```

Phi_4:

2. Koeffizientenvergleich; Element 1:1

(nx*cos(phi_0)) + (ny*sin(phi_0))	=	cos(phi_1)*sin(phi_2)*sin(phi_4) - cos(phi_1)*cos(phi_2)*cos(phi_4)

1. Dividieren mit cos(phi_1)
2. Sin(x) * sin(y) – cos(x) * cos(y) = -cos(x+y)
3. Umstellen nach phi_4

```
phi_4 =  pi - phi_2 - acos((nx*cos(phi_0) + ny*sin(phi_0))/cos(phi_1))
phi_41 =  pi - phi_2 - acos((nx*cos(phi_01) + ny*sin(phi_01))/cos(phi_11))
```

H_3:

3. Koeffizientenvergleich; Element 3:4

(cos(phi_1)*(H_0 - H_s)) - (cos(phi_1)*(H_0 + L_0)) + (pz*cos(phi_1)) - (py*sin(phi_0)*sin(phi_1)) - (px*cos(phi_0)*sin(phi_1))	=	D_1 - D_A2/2 - H_3

Durch Umstellen nach H_3:

```
H_3 = (-D_1 + D_A2/2 + cos(phi_1)*(pz-H_s-L_0) - py*sin(phi_0)*sin(phi_1) -…
px*cos(phi_0)*sin(phi_1))*(-1)
```

f) Temperatureinfluss

Nach der Berechnung der Winkel und H_3 könnten diese mit den Zahlenwerten für die Variablen in die Transformationsmatrix eingesetzt werden. Das Ergebnis zeigt die Stellung des TCP im AKS 0.

Die Vorgabe wurde angepasst an die Situation, dass die z-Achse des TCP nach oben zeigt. Das hat Einfluss auf ux, uy zu sowie az. Die Achsen müssen entsprechend der rechten Hand Regel ausgerichtet sein.

```
nx = cos(30/180*pi);
ny = sin(30/180*pi);
nz = 0;

ux = -cos(60/180*pi);
uy = sin(60/180*pi);
uz = 0;

ax = 0;
ay = 0;
az = 1;

px = 400;
py = 100;
pz = 50;
```

```
Stellung TCP morgens=
```

```
[0.87, -0.5,   0, 400.0]
[ 0.5, 0.87,   0, 100.0]
[   0,    0, 1.0,  50.0]
[   0,    0,   0,   1.0]
```

Nun werden die sich ändernden Längen aufgrund der Temperatur berechnet:

```
%neue Längen
alphaT = 12.5 * 10^(-6);
deltaT = 30 - 18;

L_0 = 800*exp(alphaT * deltaT);
L_1 = 400*exp(alphaT * deltaT);
L_2 = 500*exp(alphaT * deltaT);
L_3 = 630*exp(alphaT * deltaT);
L_4 = 600*exp(alphaT * deltaT);
L_5 = 50*exp(alphaT * deltaT);
D_1 = 110*exp(alphaT * deltaT);
D_A2 = 100*exp(alphaT * deltaT);
H_s = 200*exp(alphaT * deltaT);
H_0 = 100*exp(alphaT * deltaT);
```

Das führt mit gleichen Winkeln zu folgender Stellung:

```
Stellung TCP abends =

[0.86603,    -0.5,   0, 400.06]
[    0.5, 0.86603,   0, 100.02]
[      0,       0, 1.0, 50.159]
[      0,       0,   0,    1.0]
```

Die Abweichung wird durch Subtraktion der Stellungen dargestellt:

```
deltapos =

[-5.59339e-11,   1.1687e-10, 0, 0.0600046]
[ -1.1687e-10, -5.59339e-11, 0, 0.0150012]
[           0,            0, 0,  0.159012]
[           0,            0, 0,         0]
```

Es wird die Rückwärtstransformation mit den geänderten Längenmaßen durchgeführt. Dadurch ändern sich die Winkel, jedoch bleibt die Stellung am Abend gleich.

```
Korrigierte TCP Stellung abends =

[0.87, -0.5,   0, 400.0]
[ 0.5, 0.87,   0, 100.0]
[   0,    0, 1.0,  50.0]
[   0,    0,   0,   1.0]
```

Abschließend werden die Achswinkel mit Hub morgens und nach Delta T = 12° dargestellt.

	Winkel und Hub zu Arbeitsbeginn		Winkel und Hub nach Delta T = 12°C	
	Degree	Rad	Degree	Rad
Phi_0	90,00	1,5708	90,01	1,5709
Phi_1	0,00	0	0,00	0
Phi_2	-126,87	-2,2143	-126,88	-2,2145
Phi_4	246,87	4,3087	246,87	4,3087
H_3 [mm]	1010		1010,2	

2 Aufgabe 2: JACOBI Matrix

Die Jacobi–Matrix beschreibt den Zusammenhang zwischen den Gelenkgeschwindigkeiten und der Geschwindigkeit des TCP. Außerdem wird sie für die Ermittlung von Singularitäten und für die Bestimmung des dynamischen Robotermodells nach Lagrange verwendet. Die Dimension der Jacobi–Matrix beträgt 6 x n, wobei n die Anzahl der Gelenkachsen des Roboters ist, das heißt jede Spalte entspricht einer Gelenkachse. Da die vierte Achse eine Hub-Drehachse ist, handelt es sich um eine 6 x 5 Matrix

a) Geometrische Jacobi-Matrix

Unter Ausnutzung von geometrischen Beziehungen und der Kenntnis der Gelenkkoordinatensysteme lässt sich eine Jacobi-Matrix mit geringerem Aufwand als durch die Bildung sämtlicher partieller Ableitungen aufstellen. Sie wird als geometrische Jacobi-Matrix bezeichnet. Die Bestimmung der Jacobi-Matrix ***J*** erfolgt dabei, wie in der allgemeinen Form, spaltenweise. Die separat erstellten Spalten werden nebeneinandergesetzt und bilden die Jacobi-Matrix: (Brillowski, 2004)

$$\mathcal{J} = \mathcal{J}_1 \, \mathcal{J}_2 \cdots \mathcal{J}_n$$

Die Berechnung der Spaltenvektoren von ***J*** ist abhängig vom Gelenktyp.

Rotatorische Achse:

$$\mathcal{J}_{i+1} = \begin{pmatrix} {}^0b_i \quad \text{X} \quad ({}^0p_e - \quad {}^0p_i) \\ {}^0b_i \end{pmatrix} \quad \text{mit i} \ = \ 0, 1, 2, \ldots$$

Translatorische Achse:

$$\mathcal{J}_{i+1} = \begin{pmatrix} {}^0b_i \\ 0 \end{pmatrix} \quad \text{mit i} \ = \ 0, 1, 2, \ldots$$

Der Zusammenhang der DH-Konvention und der Jacobi-Matrix wird im nachfolgenden Beispiel erläutert. Die Vektoren ${}^0b_i$, ${}^0p_i$ und ${}^0p_e$ werden dabei mit Hilfe der Gesamttransformationsmatrix ${}^0A_7$ und der Einzeltransformationsmatrix ${}^0A_1$ erklärt

$$A01 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{H_s - H_0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$A07$: siehe Seite 8

${}^0b_i$ ist der Richtungsvektor der Achsbewegung. Die Drehbewegung geht immer um die Z-Achse. Diese ist für **J**$_i$ in Spalte 3 der Einzeltransformations-matrix zu finden. Für ${}^0b_1$ sieht diese beispielsweise folgendermaßen aus:

$${}^0b_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Der Vektor ${}^0p_i$ beschreibt die Position des Gelenks zum Basiskoordinatensystem. Er wird also aus der 4. Spalte der Einzeltransformationsmatrix gebildet. Für ${}^0p_1$ sieht dieser folgendermaßen aus:

$${}^0p_1 = \begin{pmatrix} 0 \\ 0 \\ \mathrm{H_s - H_0} \end{pmatrix}$$

Der Vektor ${}^0p_e$ stellt die TCP-Stellung zum Basiskoordinatensystem dar und kann aus der Gesamttransformationsmatrix A07 in Spalte 4 ausgelesen werden.

$${}^0p_e = \begin{pmatrix} \mathrm{L\_1*c0*c1 - D\_1*c0*s1 + c0*s1*(D\_A2/2 + L\_3) + c0*s0*(H\_3 - L\_3) - L\_2*s0*s2 + L\_2*c0*c1*c2} \\ \mathrm{L\_1*c1*s0 + L\_2*c0*s2 - D\_1*s0*s1 + s0*s1*(D\_A2/2 + L\_3) + s0*s1*(H\_3 - L\_3) + L\_2*c1*c2*s0} \\ \mathrm{H\_s + L\_0 + D\_1*c1 - c1*(D\_A2/2 + L\_3) - c1*(H\_3 - L\_3) + L\_1*s1 + L\_2*c2*s1} \end{pmatrix}$$

Für **J**$_1$ ergibt sich somit die folgende Gleichung:

$$\mathcal{J}_1 = \begin{pmatrix} {}^0b_i \quad ({}^0p_e - \quad {}^0p_i) \\ {}^0b_i \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathrm{X} \quad \left(({}^0p_e) - \begin{pmatrix} 0 \\ 0 \\ \mathrm{H_s} - \mathrm{H_0} \end{pmatrix} \right) \\ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{pmatrix}$$

Für $\mathbf{J_2}$ $\mathbf{J_3}$ und $\mathbf{J_4}$ wird gleich verfahren, wobei die Einzeltransformationsmatrizen dann von 0 bis 2 (A02), 0 bis 4 (A04) und von 0 bis 6 (A06) gehen. Da das vierte Gelenk eine Dreh-Hubachse ist, muss für $\mathbf{J_5}$ die Formel für translatorische Achsen angewendet werden.

Der folgende Matlab-Code zeigt die Berechnungen der Jacobi Matrix:

```
A01 = T_alle_matrizen{1};
A12 = T_alle_matrizen{2};
A23 = T_alle_matrizen{3};
A34 = T_alle_matrizen{4};
A45 = T_alle_matrizen{5};
A56 = T_alle_matrizen{6};
A67 = T_alle_matrizen{7};

A01 = T_kumulative{1};
A02 = T_kumulative{2};
A03 = T_kumulative{3};
A04 = T_kumulative{4};
A05 = T_kumulative{5};
A06 = T_kumulative{6};
A07 = T_kumulative{7};

% JACOBI Matrix
z = cell(1,7);
p = cell(1,7);
J = zeros(6,5);

for i=1:7

    z{i} = T_kumulative{i}(1:3,3);
    p{i} = T_kumulative{i}(1:3,4);

end

J=[cross(z{1},(p{7}-p{1})), cross(z{2},(p{7}-p{2})), cross(z{4},(p{7}-p{4})),…
cross(z{6},(p{7}-p{6})), z{6}; z{1}, z{2}, z{4}, z{6} {0;0;0}];
```

6x5 Jacobi Matrix mit Variablen:

```
J =
(erste Zeile)
[D_1*sin(phi_0)*sin(phi_1) - L_2*cos(phi_0)*sin(phi_2) - L_1*cos(phi_1)*sin(phi_0)
- sin(phi_0)*sin(phi_1)*(D_A2/2 + L_3) - sin(phi_0)*sin(phi_1)*(H_3 - L_3) -
L_2*cos(phi_1)*cos(phi_2)*sin(phi_0),

-cos(phi_0)*(D_1*cos(phi_1) - cos(phi_1)*(D_A2/2 + L_3) - cos(phi_1)*(H_3 - L_3) +
L_1*sin(phi_1) + L_2*cos(phi_2)*sin(phi_1)),
```

```
sin(phi_0)*sin(phi_1)*(cos(phi_1)*(D_A2/2 + L_3) + cos(phi_1)*(H_3 - L_3) -
L_2*cos(phi_2)*sin(phi_1)) - cos(phi_1)*(L_2*cos(phi_0)*sin(phi_2) +
sin(phi_0)*sin(phi_1)*(D_A2/2 + L_3) + sin(phi_0)*sin(phi_1)*(H_3 - L_3) +
L_2*cos(phi_1)*cos(phi_2)*sin(phi_0)),

0, -cos(phi_0)*sin(phi_1)]

Zweite Zeile:
[L_1*cos(phi_0)*cos(phi_1) - D_1*cos(phi_0)*sin(phi_1) +
cos(phi_0)*sin(phi_1)*(D_A2/2 + L_3) + cos(phi_0)*sin(phi_1)*(H_3 - L_3) -
L_2*sin(phi_0)*sin(phi_2) + L_2*cos(phi_0)*cos(phi_1)*cos(phi_2),

-sin(phi_0)*(D_1*cos(phi_1) - cos(phi_1)*(D_A2/2 + L_3) - cos(phi_1)*(H_3 - L_3) +
L_1*sin(phi_1) + L_2*cos(phi_2)*sin(phi_1)),

cos(phi_1)*(cos(phi_0)*sin(phi_1)*(D_A2/2 + L_3) + cos(phi_0)*sin(phi_1)*(H_3 -
L_3) - L_2*sin(phi_0)*sin(phi_2) + L_2*cos(phi_0)*cos(phi_1)*cos(phi_2)) -
cos(phi_0)*sin(phi_1)*(cos(phi_1)*(D_A2/2 + L_3) + cos(phi_1)*(H_3 - L_3) -
L_2*cos(phi_2)*sin(phi_1)),

0, -sin(phi_0)*sin(phi_1)]

Dritte Zeile:
[ 0,

cos(phi_0)*(L_1*cos(phi_0)*cos(phi_1) - D_1*cos(phi_0)*sin(phi_1) +
cos(phi_0)*sin(phi_1)*(D_A2/2 + L_3) + cos(phi_0)*sin(phi_1)*(H_3 - L_3) -
L_2*sin(phi_0)*sin(phi_2) + L_2*cos(phi_0)*cos(phi_1)*cos(phi_2)) +
sin(phi_0)*(L_1*cos(phi_1)*sin(phi_0) + L_2*cos(phi_0)*sin(phi_2) -
D_1*sin(phi_0)*sin(phi_1) + sin(phi_0)*sin(phi_1)*(D_A2/2 + L_3) +
sin(phi_0)*sin(phi_1)*(H_3 - L_3) + L_2*cos(phi_1)*cos(phi_2)*sin(phi_0)),

 sin(phi_0)*sin(phi_1)*(cos(phi_0)*sin(phi_1)*(D_A2/2 + L_3) +
cos(phi_0)*sin(phi_1)*(H_3 - L_3) - L_2*sin(phi_0)*sin(phi_2) +
L_2*cos(phi_0)*cos(phi_1)*cos(phi_2)) -
cos(phi_0)*sin(phi_1)*(L_2*cos(phi_0)*sin(phi_2) + sin(phi_0)*sin(phi_1)*(D_A2/2 +
L_3) + sin(phi_0)*sin(phi_1)*(H_3 - L_3) + L_2*cos(phi_1)*cos(phi_2)*sin(phi_0)),

0, cos(phi_1)]

vierte Zeile
[0,  sin(phi_0), -cos(phi_0)*sin(phi_1), -cos(phi_0)*sin(phi_1), 0]

Fünfte Zeile:
[0, -cos(phi_0), -sin(phi_0)*sin(phi_1), -sin(phi_0)*sin(phi_1), 0]

Sechste Zeile
[1, 0, cos(phi_1), cos(phi_1), 0]
```

b) Berechnung der Achsmomente

Am TCP liegt eine statische Montagekraft und Montagedrehmoment an. Die Werte hierfür sind:

$$f_{TCP} = \begin{pmatrix} {}^{TCP}f_x \\ {}^{TCP}f_y \\ {}^{TCP}f_z \end{pmatrix} = \begin{pmatrix} 50N \\ -25N \\ -75N \end{pmatrix} \qquad M_{TCP} = \begin{pmatrix} {}^{TCP}M_x \\ {}^{TCP}M_y \\ {}^{TCP}M_z \end{pmatrix} = \begin{pmatrix} 30Nm \\ 0Nm \\ -100Nm \end{pmatrix}$$

Da sich die Montagekraft und das Montagedrehmoment auf den TCP beziehen, muss diese erst auf das WKS umgerechnet werden. Dies wird mit der Gesamtrotationsmatrix realisiert. Hierfür werden für die beiden Vektoren eine vierte Zeile mit einer 0 ergänzt und mit der Gesamtrotationsmatrix multipliziert. Zu beachten ist, dass sich die y- und z-Achse im Vergleich zur Aufgabenstellung um 180° gedreht hat.

```
f_TCP = [50; -25; -75; 0];
M_TCP = [30000;0;-100000;0];

%Umrechnug auf das 0.-Koordinatensystem
f_0 = A07 *f_TCP;
M_0 = A07 * M_TCP;
```

Die externen Kräfte und Momente werden zu einem 6x1-Wrenchvektor (F0n) zusammengefasst:

```
F0n = [f_0(1,1); f_0(2,1); f_0(3,1); M_0(1,1); M_0(2,1); M_0(3,1)];
```

Es erfolgt die Berechnung der Achskräfte und -momente in allgemeiner Form:

```
tau = transpose(J) * F0n;
```

Durch Einsetzen der Winkel und Variablen werden die Achskräfte und Momente berechnet:

```
vpa(subs(tau),8)
```

Achskräfte und Momente:

```
-104240.38 Nmm (Drehmoment erste Achse)
 21662.659 Nmm (Drehmoment zweite Achse)
-81919.873 Nmm (Drehmoment dritte Achse)
 -100000.0 Nmm (Drehmoment vierte Achse)
     -75.0 N (Kraft vierte Achse)
```

c) Singularitäten

Singularitäten sind Roboterstellungen, bei denen Verluste von einem oder mehreren Freiheitsgraden auftreten. Das heißt, dass die Hand des Roboters z.B. nicht mehr in eine bestimmte Richtung bewegt werden kann. Die Jacobi-Matrix kann in singulären Stellungen nicht invertiert werden und der Roboter kann in bestimmte Richtungen keine Bewegungen vornehmen. In der Nähe von Singularitäten können Achsgeschwindigkeiten extrem ansteigen. (Brillowski, 2004)

Die wichtigste Methode, Singularitäten im Arbeitsraum zu finden, besteht darin, dass die Determinante der Jacobi-Matrix zu Null gesetzt wird. (Brillowski, 2004)

$$|\mathcal{J}| = 0$$

Die Determinante kann nur aus quadratischen Matrizen gezogen werden. Das bedeutet, dass die Anzahl der Spalten gleich der Anzahl der Zeilen sein muss. Daher wird eine weitere Spalte ergänzt. Sie wird mit Nullen gefüllt und eine 1 in der Diagonalen. Somit handelt es sich um eine 6x6 Matrix; genannt Jsin.

Anschließend wird die Determinante errechnet:

```
detJsin = simplify(det(Jsin))
```

detJsin = 0 = `(L_2*sin(phi_2)*(D_A2/2 - D_1 + H_3 + D_1*cos(2*phi_1) - (D_A2*cos(2*phi_1))/2 - H_3*cos(2*phi_1) + L_1*sin(2*phi_1)))/2`

Die rechte Seite der Gleichung wird Null, indem `sin(phi_2)` = 0.

- ➔ Phi_2 = 0, da sin(0) = 0

3 Aufgabe 3 Lagrange

3.1. Bewegungsgleichung nach Lagrange

Die Dynamik eines Roboters kann durch die Beziehungen zwischen der Bewegung des Roboters und seinen Gelenkmomenten beschrieben werden. Da es sich um mehrere Eingangs- und Ausgangsgrößen handelt, werden die Gleichungen ebenfalls in Matrixform geschrieben.

Eine generelle Form der Gleichung lautet:

$$\mathrm{Q} = \mathrm{M}(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{V}(\mathrm{q}, \dot{\mathrm{q}}) + \mathrm{G}(\mathrm{q})$$

Dabei ist **Q** der Vektor der verallgemeinerten Kräfte, **M** die konfigurationsabhängige Massenmatrix, **V(q,q̇)** ist ein Vektor nichtlinearer Terme, der durch die Zentripetal- und die Coriolis-Beschleunigung entsteht und **G(q)** ist ein Vektor, der die Gravitationsterme enthält. (Brillowski, 2004)

Im Folgenden wird die Erstellung von Bewegungsgleichungen von Robotern nach Lagrange beschrieben. Begonnen wird mit der kinetischen Energie eines Roboterglieds **i**. Sie kann beschrieben werden als:

$$E_{Kin,i} = \frac{1}{2} * V_{si}^T * m_i * V_{si} + \frac{1}{2} * \omega_i^T * {}^0\mathrm{I}_\mathrm{i} * \omega_i$$

In der Gleichung steht $\mathbf{V_{si}}$ für die Geschwindigkeit und ω_i für die Winkelgeschwindigkeit des Schwerpunktes S von Roboterglied i. Der Term $\mathbf{m_i}$ steht für die Masse von Roboterglied i. Die Größe $\mathbf{{}^0I_i}$ ist die Trägheitsmatrix von Roboterglied i bezogen auf den Schwerpunkt von i und ausgedrückt im Basiskoordinatensystem {0}. Der Tensor $\mathbf{{}^iI_i}$ drückt dagegen die Trägheit von Glied i, bezogen auf dessen Schwerpunkt in den Koordinaten von i aus. Die Transformationsbeziehung erfolgt unter Verwendung der Rotationsmatrix $\mathbf{{}^0{}_iR}$. (Brillowski, 2004)

Die kinetische Energie des gesamten Roboters kann auch mit der Trägheitsmatrix ausgedrückt werden:

$$\mathrm{E_{Kin}} = \frac{1}{2}\dot{\mathrm{q}}^\mathrm{T}\mathrm{M}\dot{\mathrm{q}}$$

Die potentielle Energie eines Robotergliedes i ist definiert als die unter Gravitationsbedingungen aufzubringende Arbeit, um den Massenschwerpunkt von Glied i von einer horizontalen Referenzebene zur aktuellen Position zu bringen. Dann gilt für die Summe der potentiellen Energie aller n Roboterglieder der Zusammenhang: (Brillowski, 2004)

$$\mathrm{E_{Pot}} = -\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{m_i} * \mathrm{g}^\mathrm{T} * \mathrm{r_{si}}$$

Der von der Referenzposition auf den Massenschwerpunkt von Glied i weisende Vektor wird mit $\mathbf{r_{si}}$ bezeichnet. Dabei ist g der Vektor der Gravitationsbeschleunigung, ausgedrückt im Basiskoordinatensystem.

Mit den erstellten Termen kann die Lagrange-Funktion L definiert werden. Sie setzt sich in allgemeiner Form aus der Differenz der kinetischen und der potentiellen Energie eines betrachteten Systems zusammen:

$$L = E_{Kin} - E_{Pot}$$

Mit den aufgestellten Termen lautet die Lagrange-Funktion dann:

$$L = \frac{1}{2}\dot{q}^T M \dot{q} - \left(-\sum_{i=1}^{n} m_i * g^T * r_{si}\right)$$

a) Trägheitsmatrizen für den Gliedschwerpunkt

Im CAD-System SolidWorks wurde der Scara-Roboter nachgebildet und analysiert. Neben der Berechnung mit einer Excel Tabelle (Die Exceltabelle ist auf dem beigelegten Stick. Benennung: Gewichte und Traegheiten) konnten mit SolidWorks das Gewicht, der Schwerpunkt und die Trägheitsmomente überprüft werden. Hier exemplarisch die Berechnung der Achse 4.

Berechnung der Achse 4 (Rohr + Greifer):

Masse Rohr:

$$m = V * \rho = \frac{{D_{A,4}}^2 - {D_{I,4}}^2}{4} * \pi * L * \rho \;\;\Rightarrow\;\; m = \frac{(140mm)^2 - (80mm)^2}{4} * \pi * 600mm * 7.75g/mm^3 = 48{,}2kg$$

Schwerpunkt Rohr:

$$X_s = 0, \qquad Y_s = 0, \qquad Z_s = \frac{L}{2} \;\Rightarrow\; Z_s = \frac{600mm}{2} = 300mm$$

Massenträgheit Rohr:
Rohr, das um eine Querachse rotiert:

$$I_{xx} = I_{yy} = \frac{m}{12} * \left(3 * \left(\frac{{D_{A,4}}^2}{2} - \frac{{D_{I,4}}^2}{2}\right) + L^2\right) \;\Rightarrow\; \frac{48{,}2}{12} * \left(3 * \left(\frac{140^2}{2} - \frac{80^2}{2}\right) + 600^2\right)$$
$$= 1524569{,}75 kg * mm^2$$

Rohr, das um seine Mittelachse rotiert:

$$I_{zz} = \frac{m}{2} * \left(\frac{{D_{A,4}}^2}{2} + \frac{{D_{I,4}}^2}{2}\right) \;\Rightarrow\; \frac{48{,}2}{2} * \left(\frac{140^2}{2} + \frac{80^2}{2}\right) = 156675{,}15\ kg * mm^2$$

Da der Greifer als Punktmasse mit 15kg angenommen wird, hat dieser keine Trägheit um den eigenen Schwerpunkt.

Abstand der Körperschwerpunkte zum Achsschwerpunkt:

Der Achsschwerpunkt liegt auf der z-Achse. Die beiden Körperschwerpunkte liegen a = 350mm entfernt. Der Abstand des Greifer Schwerpunkts zum Achsschwerpunkt lässt sich wie folgt berechnen:

$$z_{s\ Greifer} = \frac{m_{Rohr}}{m_{Rohr} + m_{Greifer}} * a = 266{,}93mm$$

$$z_{s\ Rohr} = \frac{m_{Greifer}}{m_{Rohr} + m_{Greifer}} * a = 83{,}07mm$$

Es werden nun die Trägheiten um den Achsschwerpunkt mit dem Satz von Steiner berechnet.

$$I_{xx,S,Rohr} = I_{yy,S,Rohr} = I_{xx,Rohr} + m * z_{s\ Rohr} = 1857473{,}6\ kg * mm^2$$

$$I_{xx,S,Greifer} = I_{yy,S,Greifer} = I_{xx,Greifer} + m * z_{s\ Greifer} = 1068534{,}2\ kg * mm^2$$

Die Trägheit um z bleibt unverändert, da die z-Achse der Körper der z-Achse des Schwerpunkts entspricht. Nun werden die Trägheiten Ixx, Iyy und Izz der addiert.

$$I_{xx,Achse4} = I_{yy,Achse4} = I_{xx,S,Rohr} + I_{xx,S,Greifer} = 2926007{,}75\ kg * mm^2$$

$$I_{zz,Achse4} = I_{zz} = 156675{,}15\ \mathrm{kg} * \mathrm{mm}^2$$

Bei der Überprüfung mit den im CAD ermittelten Daten wurde gezeigt, dass diese übereinstimmen und somit als korrekt angenommen werden können. Im CAD unterscheidet sich jedoch die y- und die z-Achse.

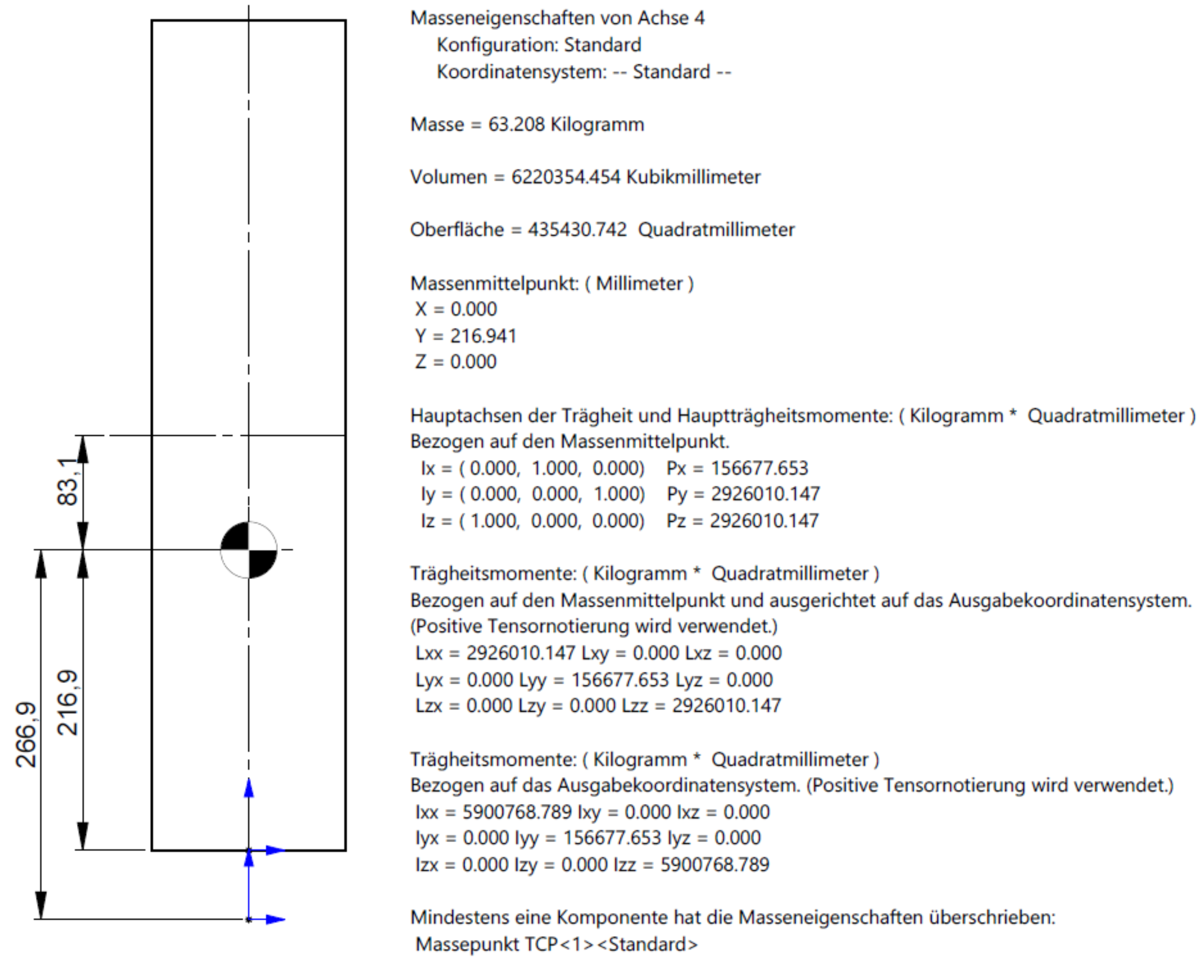


Abbildung 10 Auswertung Trägheit mit SolidWorks

Für die weiteren Berechnungen werden die Daten aus der CAD-Analyse verwendet. Als erstes werden die Trägheitsmatrizen für den Gliedschwerpunkt im Gliedkoordinatensystem aufgestellt. Trägheitstensor in kg*mm². Die Auswertungen der CAD-Analyse ist im Dokument Traegheit Achsen zu finden.

$$ {}^{i}I_i = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} $$

Achse 1:

$$ {}^{1}I_1 = \begin{bmatrix} 4877711 & 0 & 0 \\ 0 & 4877711 & 0 \\ 0 & 0 & 665748 \end{bmatrix} $$

Achse 2:

$$ {}^{2}I_2 = \begin{bmatrix} 235288 & 212609 & 0 \\ 212609 & 1340808 & 0 \\ 0 & 0 & 1407370 \end{bmatrix} $$

Achse 3:

$$^{3}\mathrm{I}_3 = \begin{bmatrix} 2333421 & 0 & 0 \\ 0 & 4280675 & -1441242 \\ 0 & -1441242 & 2021779 \end{bmatrix}$$

Achse 4:

$$^{4}\mathrm{I}_4 = \begin{bmatrix} 2926008 & 0 & 0 \\ 0 & 2926008 & 0 \\ 0 & 0 & 156675 \end{bmatrix}$$

b) Massenträgheitsmoment im Basiskoordinatensystem

Bisher sind die Trägheitstensoren auf Achsenschwerpunkte bezogen. Diese müssen nun im Basiskoordinatensystem auszudrückt werden. Dazu werden die Rotationsmatrizen verwendet.

```
R01 = T_kumulative{1}(1:3,1:3);
R02 = T_kumulative{2}(1:3,1:3);
R03 = T_kumulative{4}(1:3,1:3);
R04 = T_kumulative{6}(1:3,1:3);
```

Anschließend werden mithilfe der Rotationsmatrizen die Trägheitstensoren im Basiskoordinatensystem ausgedrückt. Dies wird in folgender Gleichung berechnet:

```
I01 = R01 * I11 *transpose(R01);
I02 = R02 * I22 *transpose(R02);
I03 = R03 * I33 *transpose(R03);
I04 = R04 * I44 *transpose(R04);
```

c) Translatorische Unter-Jacobi-Matrizen

Im Folgenden müssen die translatorischen Unter-Jacobi-Matrizen gebildet werden. Die allgemeine Schreibweise sieht folgendermaßen aus:

$$\mathrm{J}_{\mathrm{Ti}} = \begin{pmatrix} \frac{\partial r_{si,x}}{\partial \varphi 1} & \cdots & \frac{\partial r_{si,x}}{\partial \varphi n} \\ \frac{\partial r_{si,y}}{\partial \varphi 1} & \cdots & \frac{\partial r_{si,y}}{\partial \varphi n} \\ \frac{\partial r_{si,z}}{\partial \varphi 1} & \cdots & \frac{\partial r_{si,z}}{\partial \varphi n} \end{pmatrix}$$

Der Vektor $^{0}r_{si}$ beschreibt den Ortsvektor zum Schwerpunkt des jeweiligen Achsgliedes im Basiskoordinatensystem. Es werden Vektoren vom AKS zum Schwerpiunkt aus dem CAD erstellt und mit dem jeweiligen Positionsvektor addiert. Die Auswertungen der CAD-Analyse ist im Dokument Traegheit Achsen zu finden.

```
%Ortsvektor vom Schwerpunkt der ersten Achse
%x,y,z in Grundkoordinatensystem
%Schwerpunkte aus dem CAD
r01s = [
        0 ,
        0,
        H_s + L_0/2,
];

%Schwerpunkt der 2. Achse im AKS
% x = 97,58
% y = 0
% z = 13,97

r22s = [
        97.58,
         0,
        13.97,
];

r02s = T_kumulative{2}(1:3,4) + r22s;

%Schwerpunkt der 3. Achse im AKS
% x = 339.11
% y = 0
% z =  -202.38

r33s = [
        339.11,
        0,
        -202.38,
];

r03s = T_kumulative{4}(1:3,4) + r33s;

%Schwerpunkt der 3. Achse im AKS
% x = 0
% y = 0
% z = 83.1-L_4/2-L_5+H_3-L_3

r44s = [
        0 ,
        0 ,
        -83.1-L_4/2-L_5+H_3-L_3,
];

r04s = T_kumulative{6}(1:3,4) + r44s;
```

Durch Differenzieren der Schwerpunktvektoren nach ϕ0, ϕ1, ϕ2, ϕ4 und H3 werden im nächsten Schritt die Unter-Jacobi-Matrizen erzeugt.

```
%UNTER-JACOBI MATRIX

JT1= [
    diff(r01s(1,1), phi_0) diff(r01s(1,1), phi_1) diff(r01s(1,1), phi_2) diff(r01s(1,1), phi_4) diff(r01s(1,1), H_3)
    diff(r01s(2,1), phi_0) diff(r01s(2,1), phi_1) diff(r01s(2,1), phi_2) diff(r01s(2,1), phi_4) diff(r01s(2,1), H_3)
    diff(r01s(3,1), phi_0) diff(r01s(3,1), phi_1) diff(r01s(3,1), phi_2) diff(r01s(3,1), phi_4) diff(r01s(3,1), H_3)
    ];

JT2= [
    diff(r02s(1,1), phi_0) diff(r02s(1,1), phi_1) diff(r02s(1,1), phi_2) diff(r02s(1,1), phi_4) diff(r02s(1,1), H_3)
```

```
    diff(r02s(2,1), phi_0) diff(r02s(2,1), phi_1) diff(r02s(2,1), phi_2) diff(r02s(2,1), phi_4) diff(r02s(2,1), H_3)
    diff(r02s(3,1), phi_0) diff(r02s(3,1), phi_1) diff(r02s(3,1), phi_2) diff(r02s(3,1), phi_4) diff(r02s(3,1), H_3)
    ];

JT3= [
    diff(r03s(1,1), phi_0) diff(r03s(1,1), phi_1) diff(r03s(1,1), phi_2) diff(r03s(1,1), phi_4) diff(r03s(1,1), H_3)
    diff(r03s(2,1), phi_0) diff(r03s(2,1), phi_1) diff(r03s(2,1), phi_2) diff(r03s(2,1), phi_4) diff(r03s(2,1), H_3)
    diff(r03s(3,1), phi_0) diff(r03s(3,1), phi_1) diff(r03s(3,1), phi_2) diff(r02s(3,1), phi_4) diff(r03s(3,1), H_3)
    ];

JT4= [
    diff(r04s(1,1), phi_0) diff(r04s(1,1), phi_1) diff(r04s(1,1), phi_2) diff(r04s(1,1), phi_4) diff(r04s(1,1), H_3)
    diff(r04s(2,1), phi_0) diff(r04s(2,1), phi_1) diff(r04s(2,1), phi_2) diff(r04s(2,1), phi_4) diff(r04s(2,1), H_3)
    diff(r04s(3,1), phi_0) diff(r04s(3,1), phi_1) diff(r04s(3,1), phi_2) diff(r04s(3,1), phi_4) diff(r04s(3,1), H_3)
    ];

JT5= [
    diff(r04s(1,1), phi_0) diff(r04s(1,1), phi_1) diff(r04s(1,1), phi_2) diff(r04s(1,1), phi_4) diff(r04s(1,1), H_3)
    diff(r04s(2,1), phi_0) diff(r04s(2,1), phi_1) diff(r04s(2,1), phi_2) diff(r04s(2,1), phi_4) diff(r04s(2,1), H_3)
    diff(r04s(3,1), phi_0) diff(r04s(3,1), phi_1) diff(r04s(3,1), phi_2) diff(r04s(3,1), phi_4) diff(r04s(3,1), H_3)
    ];
```

d) Rotatorische Unter-Jacobi-Matrizen

Im Folgenden werden die rotatorischen Unter-Jacobi-Matrizen gebildet. Die allgemeine Schreibweise sieht folgendermaßen aus:

$$J_{Ri} = \frac{\partial \omega_i}{\partial \dot{q}} = \begin{pmatrix} \frac{\partial \omega_{1,1}}{\partial \dot{\varphi} 1} & \cdots & \frac{\partial \omega_{1,1}}{\partial \dot{\varphi} n} \\ \frac{\partial \omega_{1,2}}{\partial \dot{\varphi} 1} & \cdots & \frac{\partial \omega_{1,2}}{\partial \dot{\varphi} n} \\ \frac{\partial \omega_{1,3}}{\partial \dot{\varphi} 1} & \cdots & \frac{\partial \omega_{1,3}}{\partial \dot{\varphi} n} \end{pmatrix}$$

Für die weiteren Berechnungen werden die Vektoren der Achsgeschwindigkeiten benötigt. Sie wird anhand der Winkelgeschwindigkeiten berechnet. Diese sind durch Überlegen entstanden. Begonnen bei der ersten Achse. Aufgrund des seriellen Aufbaus ist jeweils die Winkelstellung der vorherigen Gelenken wichtig.

```
omega0 = [
    0,
    0,
    phi_0P,
];

omega1 = [
    sin(phi_0)*phi_1P,
    cos(phi_0)*phi_1P,
    phi_0P,
];

omega2 = [
    sin(phi_0)*phi_1P + cos(phi_0) * sin(phi_1) * phi_2P,
    cos(phi_0)*phi_1P + sin(phi_0) * sin(phi_1) * phi_2P,
    phi_0P+cos(phi_1)*phi_2P,
];

omega4 = [
    sin(phi_0)*phi_1P + cos(phi_0) * sin(phi_1) * (phi_2P + phi_4P),
```

```
    cos(phi_0)*phi_1P + sin(phi_0) * sin(phi_1) * (phi_2P + phi_4P),
    phi_0P + cos(phi_1)*(phi_2P +  phi_4P),
];
```

Der folgende Matlab-Code zeigt die Berechnung der rotatorischen Unter-Jacobi-Matrizen.

```
JR1= [
    diff(omega0(1,1), phi_0P) diff(omega0(1,1), phi_1P) diff(omega0(1,1), phi_2P) diff(omega0(1,1), phi_4P) diff(omega0(1,1), H_3P)
    diff(omega0(2,1), phi_0P) diff(omega0(2,1), phi_1P) diff(omega0(2,1), phi_2P) diff(omega0(2,1), phi_4P) diff(omega0(2,1), H_3P)
    diff(omega0(3,1), phi_0P) diff(omega0(3,1), phi_1P) diff(omega0(3,1), phi_2P) diff(omega0(3,1), phi_4P) diff(omega0(3,1), H_3P)
    ];

JR2= [
    diff(omega1(1,1), phi_0P) diff(omega1(1,1), phi_1P) diff(omega1(1,1), phi_2P) diff(omega1(1,1), phi_4P) diff(omega1(1,1), H_3P)
    diff(omega1(2,1), phi_0P) diff(omega1(2,1), phi_1P) diff(omega1(2,1), phi_2P) diff(omega1(2,1), phi_4P) diff(omega1(2,1), H_3P)
    diff(omega1(3,1), phi_0P) diff(omega1(3,1), phi_1P) diff(omega1(3,1), phi_2P) diff(omega1(3,1), phi_4P) diff(omega1(2,1), H_3P)
    ];

JR3= [
    diff(omega2(1,1), phi_0P) diff(omega2(1,1), phi_1P) diff(omega2(1,1), phi_2P) diff(omega2(1,1), phi_4P) diff(omega2(1,1), H_3P)
    diff(omega2(2,1), phi_0P) diff(omega2(2,1), phi_1P) diff(omega2(2,1), phi_2P) diff(omega2(2,1), phi_4P) diff(omega2(2,1), H_3P)
    diff(omega2(3,1), phi_0P) diff(omega2(3,1), phi_1P) diff(omega2(3,1), phi_2P) diff(omega2(3,1), phi_4P) diff(omega2(3,1), H_3P)
    ];

JR4= [
    diff(omega4(1,1), phi_0P) diff(omega4(1,1), phi_1P) diff(omega4(1,1), phi_2P) diff(omega4(1,1), phi_4P) diff(omega4(1,1), H_3P)
    diff(omega4(2,1), phi_0P) diff(omega4(2,1), phi_1P) diff(omega4(2,1), phi_2P) diff(omega4(2,1), phi_4P) diff(omega4(2,1), H_3P)
    diff(omega4(3,1), phi_0P) diff(omega4(3,1), phi_1P) diff(omega4(3,1), phi_2P) diff(omega4(3,1), phi_4P) diff(omega4(3,1), H_3P)
    ];

JR5= [
    diff(omega4(1,1), phi_0P) diff(omega4(1,1), phi_1P) diff(omega4(1,1), phi_2P) diff(omega4(1,1), phi_4P) diff(omega4(1,1), H_3P)
    diff(omega4(2,1), phi_0P) diff(omega4(2,1), phi_1P) diff(omega4(2,1), phi_2P) diff(omega4(2,1), phi_4P) diff(omega4(2,1), H_3P)
    diff(omega4(3,1), phi_0P) diff(omega4(3,1), phi_1P) diff(omega4(3,1), phi_2P) diff(omega4(3,1), phi_4P) diff(omega4(3,1), H_3P)
    ];
```

e) Kinetische Energie des Roboters

Mit den aufgestellten Größen wird die kinetische Energie des gesamten Roboters bestimmt.

$$E_{Kin} = \frac{1}{2}\dot{q}^{T} M \dot{q}$$

In der Gleichung steht das **M** für die Roboterträgheitsmatrix des gegebenen Scara-Roboters. Sie wird mit folgender Formel berechnet:

$$M = \sum_{i=1}^{n} (J_{Ti}^{T} * M_i * J_{Ti} + J_{Ri}^{T} * {}^{0}I_i * J_{Ri})$$

Im Matlab-Code sieht die Formel folgendermaßen aus:

```
%ROBOTERTRÄGHEITSMATRIX

M = transpose(JT1) * m1 * JT1 + transpose(JR1) * I01 * JR1 + ...
    transpose(JT2) * m2 * JT2 + transpose(JR2) * I02 * JR2 + ...
    transpose(JT3) * m3 * JT3 + transpose(JR3) * I03 * JR3 + ...
    transpose(JT4) * m4 * JT4 + transpose(JR4) * I04 * JR4 + ...
    transpose(JT5) * m4 * JT5 + transpose(JR5) * I04 * JR4;
```

Nun wird die kinetische Energie berechnet. Auf den Scara-Roboter übertragen lautet die Formel wie folgt:

$$E_{kin} = \frac{1}{2} \cdot (\dot{\varphi}_1 \quad \dot{\varphi}_2 \quad \dot{\varphi}_3 \quad \dot{H}_4) \cdot M \begin{pmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \\ \dot{H}4 \end{pmatrix}$$

Die Matlab-Berechnung ergibt folgende Gleichung für die kinetische Energie:

```
Ekin = 1/2 * [phi_0P , phi_1P , phi_2P , phi_4P , H_3P] * M * [phi_0P ; phi_1P ;
phi_2P ; phi_4P ; H_3P];
```

f) Potentielle Energie des Roboters

Um die Lagrange-Gleichung aufzustellen, wird noch die potentielle Energie des Roboters benötigt. Wie anfangs beschrieben wird diese mit folgender Gleichung berechnet:

$$E_{Pot} = -\sum_{i=1}^{n} m_i * g^T * r_{si}$$

Dabei ist **g** der Vektor der Gravitationsbeschleunigung, ausgedrückt im Basiskoordinatensystem:

$$g = (0 \quad 0 \quad -g)^T$$

Der von der Referenzposition auf den Massenschwerpunkt von Glied i weisende Vektor wird mit r0is bezeichnet. Mit den aufgestellten Größen wird die potentielle Energie des gesamten Roboters bestimmt. Der folgende Matlab-Code zeigt die Berechnung der potentiellen Energie.

```
Epot = -1 * ((m1 * [0,0,(-1*g)] * r01s)+...
             (m2 * [0,0,(-1*g)] * r02s)+...
             (m3 * [0,0,(-1*g)] * r03s)+...
             (m4 * [0,0,(-1*g)] * r04s)+...
             (m4 * [0,0,(-1*g)] * r04s));
```

g) Bewegungsgleichung nach Lagrange

Mit den erstellten Termen wird die Lagrange-Funktion L definiert. Sie setzt sich in allgemeiner Form aus der Differenz der kinetischen und der potentiellen Energie eines betrachteten Systems zusammen:

$$\mathrm{L} = \mathrm{E}_{\mathrm{kin}} - \mathrm{E}_{\mathrm{pot}}$$

In der Lagrange Gleichung wurde phi_0 mit phi_0(t) ersetzt. So auch für alle Variablen und Ableitungen davon. Die Matlab-Rückgabe lautet folgendermaßen:

```
Lt(t) =

3082683*phi_2P(t)*phi_4P(t) - (1397*g*m2)/100 + (10119*g*m3)/50 + (831*g*m4)/5 +
(7266839*phi_0P(t)^2)/2 + 4258908*phi_1P(t)^2 + (5260283*phi_2P(t)^2)/2 +
(3082683*phi_4P(t)^2)/2 - 1462577*cos(2*phi_1(t))*phi_0P(t)^2 +
(1462577*cos(2*phi_1(t))*phi_1P(t)^2)/2 + (3022245*cos(4*phi_0(t))*phi_1P(t)^2)/2
- 1462577*cos(4*phi_1(t))*phi_2P(t)^2 - (2769333*cos(4*phi_1(t))*phi_4P(t)^2)/2 +
m4*H_3P(t)^2 + 5260283*cos(phi_1(t))*phi_0P(t)*phi_2P(t) +
3082683*cos(phi_1(t))*phi_0P(t)*phi_4P(t) - H_s*g*m1 - H_s*g*m2 - H_s*g*m3 -
2*H_s*g*m4 - (L_0*g*m1)/2 - L_0*g*m2 - L_0*g*m3 - 2*L_0*g*m4 + 2*L_3*g*m4 +
L_4*g*m4 + 2*L_5*g*m4 - 720621*sin(phi_1(t))*sin(4*phi_0(t))*phi_1P(t)^2 +
(D_1^2*m3*phi_0P(t)^2)/4 + (D_1^2*m3*phi_1P(t)^2)/2 + (D_1^2*m4*phi_0P(t)^2)/2 +
D_1^2*m4*phi_1P(t)^2 + (D_A2^2*m4*phi_0P(t)^2)/8 + (D_A2^2*m4*phi_1P(t)^2)/4 +
(L_1^2*m3*phi_0P(t)^2)/4 + (L_1^2*m3*phi_1P(t)^2)/2 + (L_1^2*m4*phi_0P(t)^2)/2 +
L_1^2*m4*phi_1P(t)^2 + (3*L_2^2*m4*phi_0P(t)^2)/4 + (L_2^2*m4*phi_1P(t)^2)/2 +
(L_3^2*m4*phi_0P(t)^2)/2 + L_2^2*m4*phi_2P(t)^2 + L_3^2*m4*phi_1P(t)^2 -
2925154*cos(3*phi_1(t))*phi_0P(t)*phi_2P(t) -
2769333*cos(3*phi_1(t))*phi_0P(t)*phi_4P(t) -
2769333*cos(4*phi_1(t))*phi_2P(t)*phi_4P(t) +
212609*sin(2*phi_0(t))*phi_0P(t)*phi_1P(t) - 2*g*m4*H_3(t) -
(1462577*cos(2*phi_1(t))*cos(4*phi_0(t))*phi_1P(t)^2)/2 +
1441242*cos(phi_1(t))*cos(2*phi_0(t))*phi_0P(t)*phi_1P(t) +
5260283*sin(phi_1(t))*sin(2*phi_0(t))*phi_1P(t)*phi_2P(t) +
3082683*sin(phi_1(t))*sin(2*phi_0(t))*phi_1P(t)*phi_4P(t) -
(D_1^2*m3*cos(2*phi_1(t))*phi_0P(t)^2)/4 -
(D_1^2*m4*cos(2*phi_1(t))*phi_0P(t)^2)/2 -
(D_A2^2*m4*cos(2*phi_1(t))*phi_0P(t)^2)/8 +
(L_1^2*m3*cos(2*phi_1(t))*phi_0P(t)^2)/4 +
(L_1^2*m4*cos(2*phi_1(t))*phi_0P(t)^2)/2 +
(L_2^2*m4*cos(2*phi_1(t))*phi_0P(t)^2)/4 -
(L_2^2*m4*cos(2*phi_2(t))*phi_0P(t)^2)/4 -
(L_3^2*m4*cos(2*phi_1(t))*phi_0P(t)^2)/2 +
(L_2^2*m4*cos(2*phi_2(t))*phi_1P(t)^2)/2 +
1441242*cos(2*phi_0(t))*cos(2*phi_1(t))*phi_1P(t)*phi_2P(t) +
2925154*sin(2*phi_0(t))*sin(2*phi_1(t))*phi_0P(t)*phi_1P(t) +
2925154*sin(2*phi_0(t))*sin(3*phi_1(t))*phi_1P(t)*phi_2P(t) +
2769333*sin(2*phi_0(t))*sin(3*phi_1(t))*phi_1P(t)*phi_4P(t) -
(D_1*D_A2*m4*phi_0P(t)^2)/2 - D_1*D_A2*m4*phi_1P(t)^2 - D_1*L_3*m4*phi_0P(t)^2 -
2*D_1*L_3*m4*phi_1P(t)^2 + (D_A2*L_3*m4*phi_0P(t)^2)/2 + D_A2*L_3*m4*phi_1P(t)^2 -
D_1*g*m3*cos(phi_1(t)) - 2*D_1*g*m4*cos(phi_1(t)) + D_A2*g*m4*cos(phi_1(t)) +
2*L_3*g*m4*cos(phi_1(t)) - L_1*g*m3*sin(phi_1(t)) - 2*L_1*g*m4*sin(phi_1(t)) +
2*L_1*m4*cos(phi_1(t))*H_3P(t)*phi_1P(t) -
```

```
2*D_1*m4*sin(phi_1(t))*H_3P(t)*phi_1P(t) + D_A2*m4*sin(phi_1(t))*H_3P(t)*phi_1P(t)
+ (L_2^2*m4*cos(2*phi_1(t))*cos(2*phi_2(t))*phi_0P(t)^2)/4 +
2*L_3*m4*sin(phi_1(t))*H_3P(t)*phi_1P(t) - 2*L_2*g*m4*cos(phi_2(t))*sin(phi_1(t))
+ (D_1*D_A2*m4*cos(2*phi_1(t))*phi_0P(t)^2)/2 +
D_1*L_3*m4*cos(2*phi_1(t))*phi_0P(t)^2 -
(D_A2*L_3*m4*cos(2*phi_1(t))*phi_0P(t)^2)/2 -
(D_1*L_1*m3*sin(2*phi_1(t))*phi_0P(t)^2)/2 -
D_1*L_1*m4*sin(2*phi_1(t))*phi_0P(t)^2 +
(D_A2*L_1*m4*sin(2*phi_1(t))*phi_0P(t)^2)/2 +
L_1*L_3*m4*sin(2*phi_1(t))*phi_0P(t)^2 +
2*L_2^2*m4*cos(phi_1(t))*phi_0P(t)*phi_2P(t) +
L_1*L_2*m4*cos(phi_2(t))*phi_0P(t)^2 + 2*L_1*L_2*m4*cos(phi_2(t))*phi_1P(t)^2 +
2*L_2*m4*cos(phi_1(t))*cos(phi_2(t))*H_3P(t)*phi_1P(t) +
L_2^2*m4*sin(phi_1(t))*sin(2*phi_2(t))*phi_0P(t)*phi_1P(t) -
2*L_2*m4*sin(phi_1(t))*sin(phi_2(t))*H_3P(t)*phi_2P(t) +
2*D_1*L_2*m4*sin(phi_2(t))*phi_1P(t)*phi_2P(t) -
D_A2*L_2*m4*sin(phi_2(t))*phi_1P(t)*phi_2P(t) -
2*L_2*L_3*m4*sin(phi_2(t))*phi_1P(t)*phi_2P(t) +
L_1*L_2*m4*cos(phi_2(t))*cos(2*phi_1(t))*phi_0P(t)^2 -
D_1*L_2*m4*cos(phi_2(t))*sin(2*phi_1(t))*phi_0P(t)^2 +
(D_A2*L_2*m4*cos(phi_2(t))*sin(2*phi_1(t))*phi_0P(t)^2)/2 +
L_2*L_3*m4*cos(phi_2(t))*sin(2*phi_1(t))*phi_0P(t)^2 +
2*L_1*L_2*m4*cos(phi_1(t))*cos(phi_2(t))*phi_0P(t)*phi_2P(t) +
2*D_1*L_2*m4*cos(phi_1(t))*sin(phi_2(t))*phi_0P(t)*phi_1P(t) -
2*D_1*L_2*m4*cos(phi_2(t))*sin(phi_1(t))*phi_0P(t)*phi_2P(t) -
D_A2*L_2*m4*cos(phi_1(t))*sin(phi_2(t))*phi_0P(t)*phi_1P(t) +
D_A2*L_2*m4*cos(phi_2(t))*sin(phi_1(t))*phi_0P(t)*phi_2P(t) -
2*L_2*L_3*m4*cos(phi_1(t))*sin(phi_2(t))*phi_0P(t)*phi_1P(t) +
2*L_2*L_3*m4*cos(phi_2(t))*sin(phi_1(t))*phi_0P(t)*phi_2P(t) +
2*L_1*L_2*m4*sin(phi_1(t))*sin(phi_2(t))*phi_0P(t)*phi_1P(t)
```

Mit der Lagrange-Funktion **L**, den Koordinaten **ϕi** und ihren Ableitungen von **ϕi** werden die Lagrange-Gleichungen gebildet. Damit werden die Bewegungsgleichungen für ein Robotersystem erstellt:

$$Q_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}_i}\right) - \left(\frac{\partial L}{\partial \varphi_i}\right)$$

Durch mehrfaches Differenzieren (nach $\partial\varphi_i$, $\partial\dot{\varphi}_i$ und ∂t) sowie die Angaben von Längen, Massen, Schwerpunkte und Massenträgheiten werden folgende Gleichungen für τ_1, τ_2, τ_3, τ_4 und τ_5:

$$\partial\dot{\varphi}_i = phi\_P$$

$$\partial\ddot{\varphi}_\imath = phi\_PP$$

```
tau1(t) =

83395383.0*phi_0PP(t) - 53740000.0*phi_0P(t)*phi_1P(t) +
26870000.0*phi_0PP(t)*sin(2.0*phi_1(t)) + 212609.0*phi_1PP(t)*sin(2.0*phi_0(t)) +
45635745.0*cos(phi_1(t))*phi_2PP(t) + 11390682.0*cos(phi_1(t))*phi_4PP(t) -
5850308.0*phi_1P(t)^2*sin(2.0*phi_0(t)) + 8969644.0*phi_1P(t)^2*sin(4.0*phi_0(t))
- 20141698.0*cos(phi_1(t))^2*phi_0PP(t) - 31600000.0*cos(phi_2(t))^2*phi_0PP(t) -
11700616.0*cos(phi_1(t))^3*phi_2PP(t) - 11077332.0*cos(phi_1(t))^3*phi_4PP(t) -
2882484.0*sin(phi_1(t))*phi_1P(t)^2 +
20141698.0*phi_0P(t)*phi_1P(t)*sin(2.0*phi_1(t)) -
2882484.0*phi_1P(t)*phi_2P(t)*sin(2.0*phi_0(t)) +
31600000.0*phi_0P(t)*phi_2P(t)*sin(2.0*phi_2(t)) +
50560000.0*cos(phi_1(t))^2*cos(phi_2(t))*phi_0PP(t) +
5764968.0*sin(phi_1(t))*phi_1P(t)^2*cos(2.0*phi_0(t))^2 +
11700616.0*cos(phi_1(t))^2*phi_1P(t)^2*sin(2.0*phi_0(t)) +
25280000.0*cos(phi_1(t))*sin(phi_2(t))*phi_1P(t)^2 -
25280000.0*cos(phi_1(t))*sin(phi_2(t))*phi_2P(t)^2 +
36024000.0*sin(phi_1(t))*sin(phi_2(t))*phi_1P(t)^2 -
36024000.0*sin(phi_1(t))*sin(phi_2(t))*phi_2P(t)^2 -
72048000.0*cos(phi_2(t))*phi_0P(t)*phi_1P(t) +
1441242.0*cos(phi_1(t))*phi_1PP(t)*cos(2.0*phi_0(t)) -
42133897.0*sin(phi_1(t))*phi_1P(t)*phi_2P(t) +
21841314.0*sin(phi_1(t))*phi_1P(t)*phi_4P(t) +
31600000.0*cos(phi_1(t))^2*cos(phi_2(t))^2*phi_0PP(t) +
107480000.0*cos(phi_1(t))^2*phi_0P(t)*phi_1P(t) +
25280000.0*cos(phi_1(t))*cos(phi_2(t))*phi_2PP(t) -
35101848.0*sin(phi_1(t))^3*phi_1P(t)*phi_2P(t) -
33231996.0*sin(phi_1(t))^3*phi_1P(t)*phi_4P(t) -
1441242.0*sin(phi_1(t))*phi_1P(t)^2*cos(2.0*phi_0(t)) -
36024000.0*cos(phi_1(t))*sin(phi_2(t))*phi_1PP(t) +
36024000.0*cos(phi_2(t))*sin(phi_1(t))*phi_2PP(t) +
25280000.0*sin(phi_1(t))*sin(phi_2(t))*phi_1PP(t) +
144096000.0*cos(phi_1(t))^2*cos(phi_2(t))*phi_0P(t)*phi_1P(t) -
50560000.0*cos(phi_1(t))^2*sin(phi_2(t))*phi_0P(t)*phi_2P(t) +
63200000.0*cos(phi_2(t))^2*sin(phi_1(t))*phi_1P(t)*phi_2P(t) -
11700616.0*cos(phi_1(t))^2*phi_1P(t)^2*cos(2.0*phi_0(t))*sin(2.0*phi_0(t)) +
72048000.0*cos(phi_1(t))*cos(phi_2(t))*sin(phi_1(t))*phi_0PP(t) +
31600000.0*cos(phi_2(t))*sin(phi_1(t))*sin(phi_2(t))*phi_1PP(t) -
4670258.0*sin(phi_1(t))*phi_1P(t)*phi_2P(t)*cos(2.0*phi_0(t)) -
626700.0*sin(phi_1(t))*phi_1P(t)*phi_4P(t)*cos(2.0*phi_0(t)) +
31600000.0*cos(phi_1(t))*cos(phi_2(t))*sin(phi_2(t))*phi_1P(t)^2 +
5764968.0*cos(phi_1(t))^2*phi_1P(t)*phi_2P(t)*sin(2.0*phi_0(t)) +
5850308.0*cos(phi_1(t))*sin(phi_1(t))*phi_1PP(t)*sin(2.0*phi_0(t)) -
23401232.0*cos(phi_1(t))^2*sin(phi_1(t))*phi_1P(t)*phi_2P(t)*cos(2.0*phi_0(t)) -
22154664.0*cos(phi_1(t))^2*sin(phi_1(t))*phi_1P(t)*phi_4P(t)*cos(2.0*phi_0(t)) -
101120000.0*cos(phi_1(t))*cos(phi_2(t))*sin(phi_1(t))*phi_0P(t)*phi_1P(t) -
72048000.0*cos(phi_1(t))*sin(phi_1(t))*sin(phi_2(t))*phi_0P(t)*phi_2P(t) -
63200000.0*cos(phi_1(t))*cos(phi_2(t))^2*sin(phi_1(t))*phi_0P(t)*phi_1P(t) -
63200000.0*cos(phi_1(t))^2*cos(phi_2(t))*sin(phi_2(t))*phi_0P(t)*phi_2P(t)
```

```
tau2(t) =

93233206.0*phi_1PP(t) + 68280.0*g*cos(phi_1(t)) + 67175.0*g*sin(phi_1(t)) +
3022245.0*phi_1PP(t)*cos(4.0*phi_0(t)) + 1462577.0*phi_1PP(t)*cos(2.0*phi_1(t)) +
15800000.0*phi_1PP(t)*cos(2.0*phi_2(t)) + 212609.0*phi_0PP(t)*sin(2.0*phi_0(t)) +
50560.0*cos(phi_1(t))*H_3PP(t) + 72048.0*sin(phi_1(t))*H_3PP(t) +
425218.0*phi_0P(t)^2*cos(2.0*phi_0(t)) - 26870000.0*phi_0P(t)^2*cos(2.0*phi_1(t))
+ 50560000.0*cos(phi_2(t))*phi_1PP(t) - 2170849.0*phi_0P(t)^2*sin(2.0*phi_1(t)) -
1462577.0*phi_1P(t)^2*sin(2.0*phi_1(t)) - 5850308.0*phi_2P(t)^2*sin(4.0*phi_1(t))
- 5538666.0*phi_4P(t)^2*sin(4.0*phi_1(t)) - 36024000.0*sin(phi_2(t))*phi_2PP(t) -
36024000.0*cos(phi_2(t))*phi_2P(t)^2 -
1462577.0*phi_1PP(t)*cos(4.0*phi_0(t))*cos(2.0*phi_1(t)) +
1441242.0*phi_2PP(t)*cos(2.0*phi_0(t))*cos(2.0*phi_1(t)) -
12088980.0*phi_0P(t)*phi_1P(t)*sin(4.0*phi_0(t)) -
31600000.0*phi_1P(t)*phi_2P(t)*sin(2.0*phi_2(t)) -
11077332.0*phi_2P(t)*phi_4P(t)*sin(4.0*phi_1(t)) +
2925154.0*phi_0PP(t)*sin(2.0*phi_0(t))*sin(2.0*phi_1(t)) -
8775462.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_2P(t) -
8307999.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_4P(t) +
2925154.0*sin(3.0*phi_1(t))*phi_2PP(t)*sin(2.0*phi_0(t)) +
2769333.0*sin(3.0*phi_1(t))*phi_4PP(t)*sin(2.0*phi_0(t)) +
63200.0*g*cos(phi_1(t))*cos(phi_2(t)) +
1441242.0*cos(phi_1(t))*phi_0PP(t)*cos(2.0*phi_0(t)) +
36860283.0*sin(phi_1(t))*phi_0P(t)*phi_2P(t) +
3082683.0*sin(phi_1(t))*phi_0P(t)*phi_4P(t) -
50560000.0*sin(phi_2(t))*phi_1P(t)*phi_2P(t) +
5850308.0*phi_0P(t)^2*cos(2.0*phi_0(t))*sin(2.0*phi_1(t)) +
7900000.0*phi_0P(t)^2*cos(2.0*phi_2(t))*sin(2.0*phi_1(t)) +
1462577.0*phi_1P(t)^2*cos(4.0*phi_0(t))*sin(2.0*phi_1(t)) +
15800000.0*sin(phi_1(t))*phi_0PP(t)*sin(2.0*phi_2(t)) -
1441242.0*sin(phi_1(t))*phi_1PP(t)*sin(4.0*phi_0(t)) +
5260283.0*sin(phi_1(t))*phi_2PP(t)*sin(2.0*phi_0(t)) +
3082683.0*sin(phi_1(t))*phi_4PP(t)*sin(2.0*phi_0(t)) +
63200.0*cos(phi_1(t))*cos(phi_2(t))*H_3PP(t) -
36024000.0*cos(phi_2(t))*phi_0P(t)^2*cos(2.0*phi_1(t)) -
2882484.0*cos(phi_1(t))*phi_0P(t)^2*sin(2.0*phi_0(t)) -
720621.0*cos(phi_1(t))*phi_1P(t)^2*sin(4.0*phi_0(t)) +
25280000.0*cos(phi_2(t))*phi_0P(t)^2*sin(2.0*phi_1(t)) -
36024000.0*cos(phi_1(t))*sin(phi_2(t))*phi_0PP(t) +
25280000.0*sin(phi_1(t))*sin(phi_2(t))*phi_0PP(t) +
5850308.0*phi_0P(t)*phi_1P(t)*cos(2.0*phi_1(t))*sin(4.0*phi_0(t)) -
2882484.0*phi_0P(t)*phi_2P(t)*cos(2.0*phi_1(t))*sin(2.0*phi_0(t)) +
5850308.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_2P(t)*cos(2.0*phi_0(t)) +
5538666.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_4P(t)*cos(2.0*phi_0(t)) -
5764968.0*sin(phi_1(t))*phi_0P(t)*phi_1P(t)*cos(4.0*phi_0(t)) +
10520566.0*sin(phi_1(t))*phi_0P(t)*phi_2P(t)*cos(2.0*phi_0(t)) +
31600000.0*sin(phi_1(t))*phi_0P(t)*phi_2P(t)*cos(2.0*phi_2(t)) +
6165366.0*sin(phi_1(t))*phi_0P(t)*phi_4P(t)*cos(2.0*phi_0(t)) -
72048000.0*cos(phi_1(t))*cos(phi_2(t))*phi_0P(t)*phi_2P(t) +
50560000.0*cos(phi_2(t))*sin(phi_1(t))*phi_0P(t)*phi_2P(t)
```

```
tau3(t) =

36860283.0*phi_2PP(t) + 3082683.0*phi_4PP(t) -
2925154.0*phi_2PP(t)*cos(4.0*phi_1(t)) - 2769333.0*phi_4PP(t)*cos(4.0*phi_1(t)) -
2925154.0*cos(3.0*phi_1(t))*phi_0PP(t) + 36860283.0*cos(phi_1(t))*phi_0PP(t) -
7900000.0*phi_0P(t)^2*sin(2.0*phi_2(t)) + 15800000.0*phi_1P(t)^2*sin(2.0*phi_2(t))
- 36024000.0*sin(phi_2(t))*phi_1PP(t) + 12640000.0*sin(phi_2(t))*phi_0P(t)^2 +
25280000.0*sin(phi_2(t))*phi_1P(t)^2 +
1441242.0*phi_1PP(t)*cos(2.0*phi_0(t))*cos(2.0*phi_1(t)) +
11700616.0*phi_1P(t)*phi_2P(t)*sin(4.0*phi_1(t)) +
11077332.0*phi_1P(t)*phi_4P(t)*sin(4.0*phi_1(t)) +
8775462.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_1P(t) +
2925154.0*sin(3.0*phi_1(t))*phi_1PP(t)*sin(2.0*phi_0(t)) -
63200.0*g*sin(phi_1(t))*sin(phi_2(t)) -
36860283.0*sin(phi_1(t))*phi_0P(t)*phi_1P(t) -
2882484.0*phi_1P(t)^2*cos(2.0*phi_0(t))*sin(2.0*phi_1(t)) +
7900000.0*phi_0P(t)^2*cos(2.0*phi_1(t))*sin(2.0*phi_2(t)) +
5260283.0*sin(phi_1(t))*phi_1PP(t)*sin(2.0*phi_0(t)) +
8775462.0*cos(3.0*phi_1(t))*phi_1P(t)^2*sin(2.0*phi_0(t)) -
63200.0*sin(phi_1(t))*sin(phi_2(t))*H_3PP(t) +
25280000.0*cos(phi_1(t))*cos(phi_2(t))*phi_0PP(t) +
5260283.0*cos(phi_1(t))*phi_1P(t)^2*sin(2.0*phi_0(t)) +
12640000.0*sin(phi_2(t))*phi_0P(t)^2*cos(2.0*phi_1(t)) +
36024000.0*cos(phi_2(t))*sin(phi_1(t))*phi_0PP(t) +
18012000.0*sin(phi_2(t))*phi_0P(t)^2*sin(2.0*phi_1(t)) -
2882484.0*phi_0P(t)*phi_1P(t)*cos(2.0*phi_1(t))*sin(2.0*phi_0(t)) +
5850308.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_1P(t)*cos(2.0*phi_0(t)) +
10520566.0*sin(phi_1(t))*phi_0P(t)*phi_1P(t)*cos(2.0*phi_0(t)) -
31600000.0*sin(phi_1(t))*phi_0P(t)*phi_1P(t)*cos(2.0*phi_2(t)) +
72048000.0*cos(phi_1(t))*cos(phi_2(t))*phi_0P(t)*phi_1P(t) -
50560000.0*cos(phi_2(t))*sin(phi_1(t))*phi_0P(t)*phi_1P(t)


tau4(t) =

3082683.0*phi_2PP(t) + 3082683.0*phi_4PP(t) -
2769333.0*phi_2PP(t)*cos(4.0*phi_1(t)) - 2769333.0*phi_4PP(t)*cos(4.0*phi_1(t)) -
2769333.0*cos(3.0*phi_1(t))*phi_0PP(t) + 3082683.0*cos(phi_1(t))*phi_0PP(t) +
11077332.0*phi_1P(t)*phi_2P(t)*sin(4.0*phi_1(t)) +
11077332.0*phi_1P(t)*phi_4P(t)*sin(4.0*phi_1(t)) +
8307999.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_1P(t) +
2769333.0*sin(3.0*phi_1(t))*phi_1PP(t)*sin(2.0*phi_0(t)) -
3082683.0*sin(phi_1(t))*phi_0P(t)*phi_1P(t) +
3082683.0*sin(phi_1(t))*phi_1PP(t)*sin(2.0*phi_0(t)) +
8307999.0*cos(3.0*phi_1(t))*phi_1P(t)^2*sin(2.0*phi_0(t)) +
3082683.0*cos(phi_1(t))*phi_1P(t)^2*sin(2.0*phi_0(t)) +
5538666.0*sin(3.0*phi_1(t))*phi_0P(t)*phi_1P(t)*cos(2.0*phi_0(t)) +
6165366.0*sin(phi_1(t))*phi_0P(t)*phi_1P(t)*cos(2.0*phi_0(t))
```

```
tau5(t) =

126.4*g + 126.4*H_3PP(t) + 50560.0*cos(phi_1(t))*phi_1PP(t) +
72048.0*sin(phi_1(t))*phi_1PP(t) + 72048.0*cos(phi_1(t))*phi_1P(t)^2 -
50560.0*sin(phi_1(t))*phi_1P(t)^2 -
63200.0*cos(phi_2(t))*sin(phi_1(t))*phi_1P(t)^2 -
63200.0*cos(phi_2(t))*sin(phi_1(t))*phi_2P(t)^2 +
63200.0*cos(phi_1(t))*cos(phi_2(t))*phi_1PP(t) -
63200.0*sin(phi_1(t))*sin(phi_2(t))*phi_2PP(t) -
126400.0*cos(phi_1(t))*sin(phi_2(t))*phi_1P(t)*phi_2P(t)
```

3.2. Simulation des Bewegungsverhaltens

a) Ruckbegrenztes Lagesollprofil

Für die Berechnung des Lagesollprofils benötigt man den Ruck r_{max} die maximale Beschleunigung a_{max}, die maximale Geschwindigkeit v_{max} und den Weg s_{max}. Zudem sind die zulässige Beschleunigung und die zulässige Geschwindigkeit erforderlich.

Mit diesen Daten kann man mithilfe des Ablaufs zur Berechnung des Sollprofils die relevanten Fälle ermitteln. Da in unserem Fall die maximale Beschleunigung größer ist als die zulässige Beschleunigung, muss die Beschleunigung auf die zulässige begrenzt werden. Ebenso muss die maximale Geschwindigkeit auf die zulässige begrenzt werden, da sie größer ist. Da die zulässige Geschwindigkeit nicht kleiner ist als die zulässige Beschleunigung quadriert durch den Ruck, fällt unser Fall unter Fall IIIa nein. Dies gilt für alle Winkel. Für H_3 tritt der Fall IIIb ein.

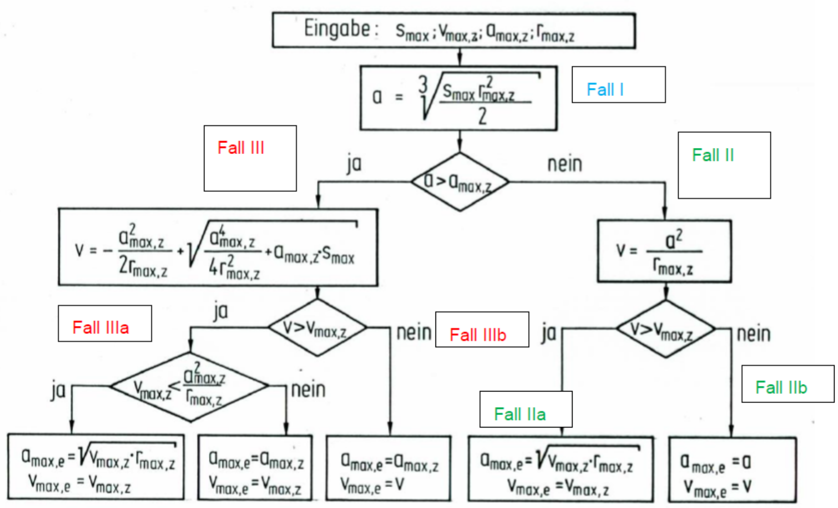


Abbildung 11 Ablauf zur Berechnung des Sollprofils (Ketterer, SERVOMECHANISMEN V1.3)

Berechnung der Zeiten

1. Berechnung von t_{01}:

t_{01} errechnet sich aus $\frac{a_{max}}{r_{max}}$

2. Berechnung von t_{12}:

Da die maximale Geschwindigkeit bei t_3 erreicht wird, gilt:

$$v_{max} = \frac{1}{2} a_{max} t_{01} \cdot 2 + a_{max} t_{12}$$

Umgestellt nach t_{12} ergibt sich:

$$t_{12} = \frac{v_{max} - a_{max} t_1}{a_{max}}$$

3. Symmetrie der Intervalle:

$$t_{01} = t_{23} = t_{45} = t_{67}$$

$$T_{12} = T_{56}$$

4. Berechnung von t_{34}:

$$s_{max} = (\int_0^{t_{01}} r_{max}\ dt + \int_{t_{01}}^{t_{12}} a_{max}\ dt + \int_{t_{12}}^{t_{23}} -r_{max}\ dt) * 2 + v_{max}\ *\ t_{34}$$

So kann man es nach t_{34} umstellen

Definition der Abschnitte

1. Abschnitt t_{01}:

$$r(i) = r_{max},$$

$$a(i) = r_{max} \cdot t(i)$$

$$v(i) = 0.5 \cdot r_{max} \cdot t(i)^2$$

$$s(i) = \frac{1}{6} \cdot r_{max} \cdot t(i)^3$$

2. Abschnitt t_{12}:

$$r(i) = 0$$

$$a(i) = a_{max}$$

$$v(i) = a_{max} \cdot (t(i) - t_1) + 0.5 \cdot r_{max} \cdot t_1^2$$

$$\mathrm{s(i)} = \frac{1}{2} \cdot \mathrm{a_{max}} \cdot (\mathrm{t(i)} - \mathrm{t_1})^2 + 0.5 \cdot \mathrm{r_{max}} \cdot \mathrm{t_1^2} \cdot (\mathrm{t(i)} - \mathrm{t_1}) + \frac{1}{6} \cdot \mathrm{r_{max}} \cdot \mathrm{t_1^3}$$

3. Abschnitt t_{23}:

$$r(i) = -r_{max},$$

$$a(i) = a_{max} \; - \; r_{max} \cdot t(i)$$

$$v(i) = 0.5 \cdot r_{max} \cdot t(i)^2$$

$$s(i) = \frac{1}{6} \cdot r_{max} \cdot t(i)^3$$

4. Abschnitt t_{34}:

$$\mathrm{r(i)} = 0$$

$$\mathrm{a(i)} = 0$$

$$\mathrm{v(i)} = \mathrm{v(i-1)}$$

$$\mathrm{s(i)} = \mathrm{s(i-1)} + \mathrm{v(i)} \cdot \big(\mathrm{t(i)} - \mathrm{t(i-1)}\big)$$

5. Abschnitt t_{45}:

$$\mathrm{r(i)} = -\mathrm{r_{max}}$$

$$a(i) = -r_{max} \cdot \left(t(i) - (t_4)\right)$$

$$v(i) = v(i-1) + a(i) \cdot \left(t(i) - t(i-1)\right)$$

$$s(i) = s(i-1) + v(i) \cdot \left(t(i) - t(i-1)\right) + 0.5 \cdot a(i) \cdot \left(t(i) - t(i-1)\right)^2$$

6. Abschnitt t_{56}:

$$r(i) = 0$$

$$a(i) = -a_{max}$$

$$v(i) = v(i-1) + a(i) \cdot \left(t(i) - t(i-1)\right)$$

$$s(i) = s(i-1) + v(i) \cdot \left(t(i) - t(i-1)\right) + 0.5 \cdot a(i) \cdot \left(t(i) - t(i-1)\right)^2$$

7. Abschnitt t_{67}:

$$r(i) = r_{max}$$

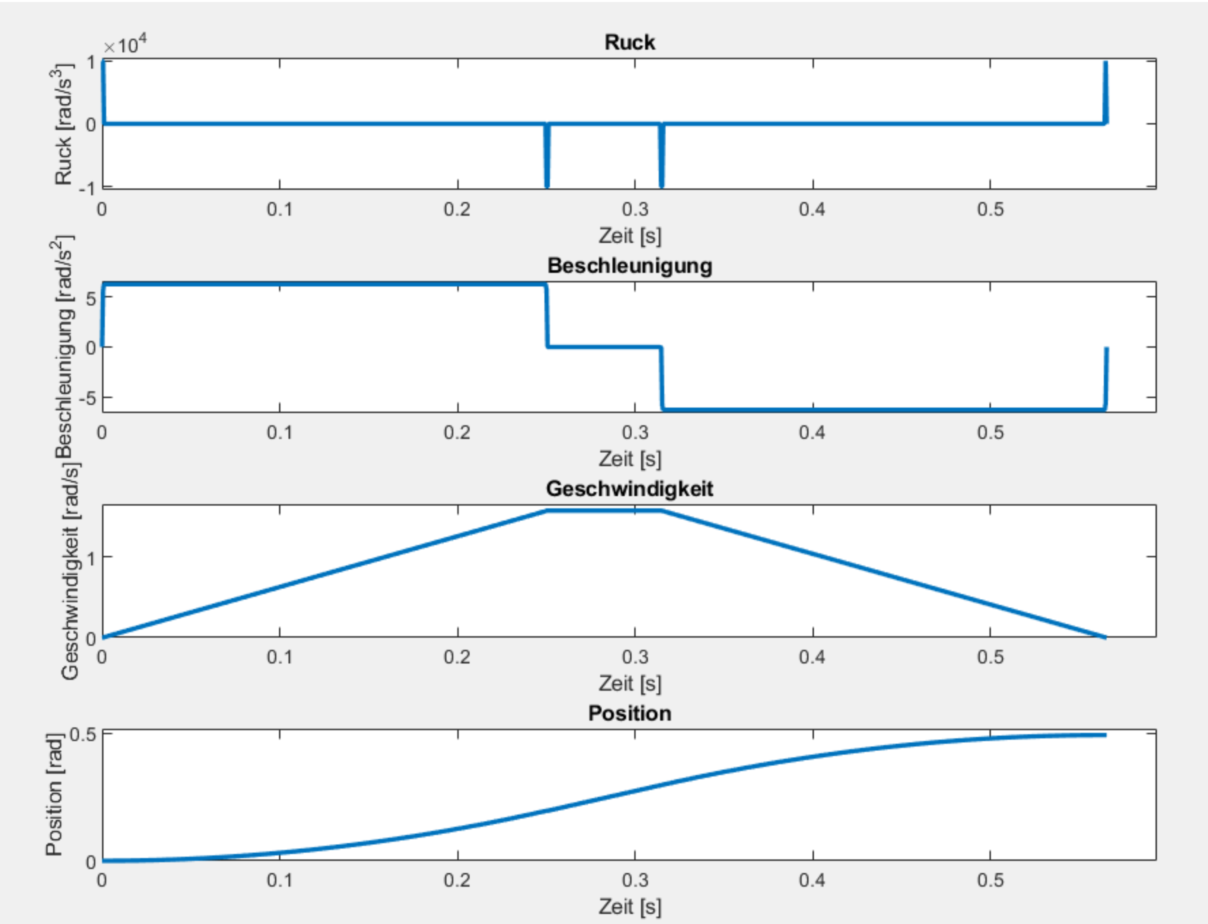
$$a(i) = -a_{max} + r_{max} \cdot \big(t(i) - (t_6)\big)$$

$$v(i) = v(i-1) + a(i) \cdot \big(t(i) - t(i-1)\big)$$

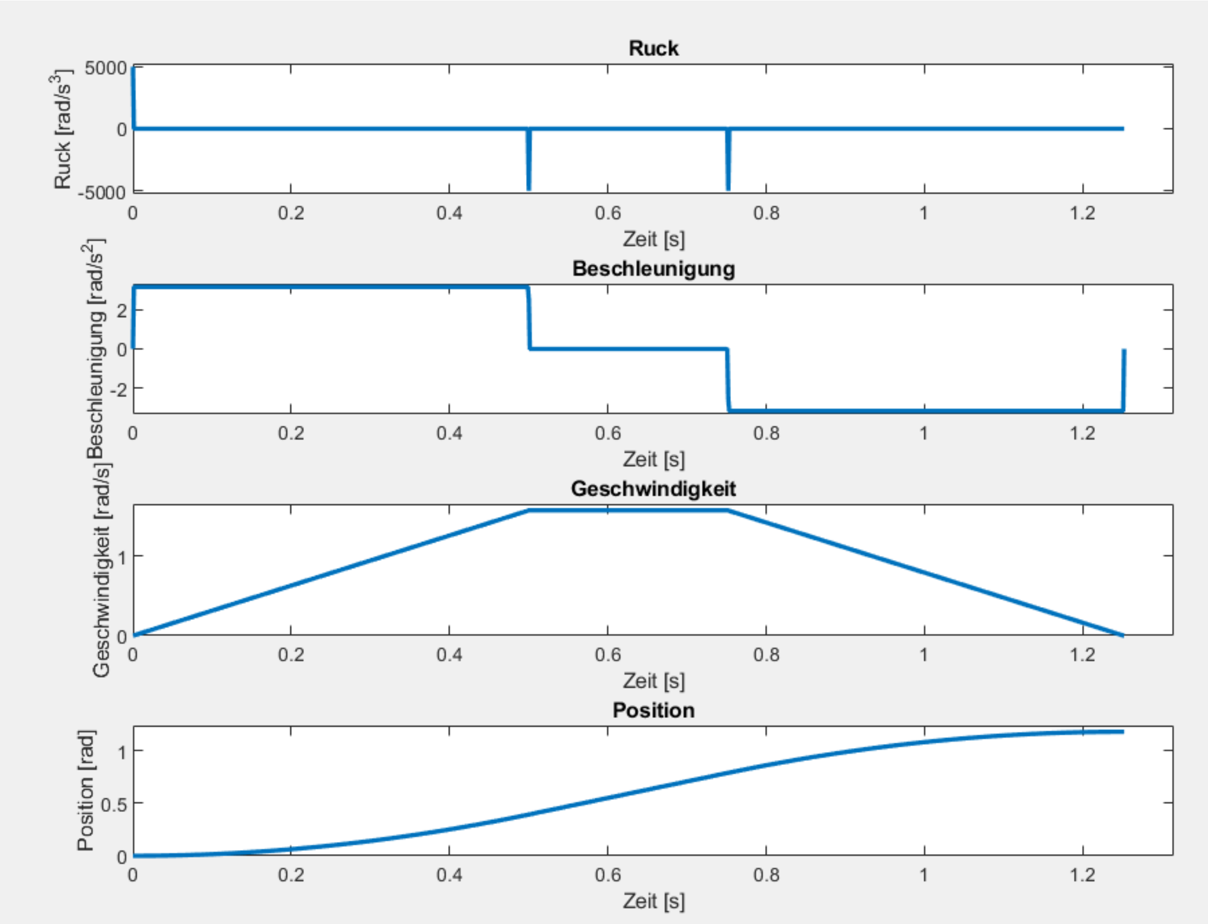
$$s(i) = s(i-1) + v(i) \cdot \big(t(i) - t(i-1)\big) + 0.5 \cdot a(i) \cdot \big(t(i) - t(i-1)\big)^2$$

So hat man eine allgemeine Vorgehensweise für den Fall IIIa (nein). Dies wird auf jeden Winkel und H3 angewendet.

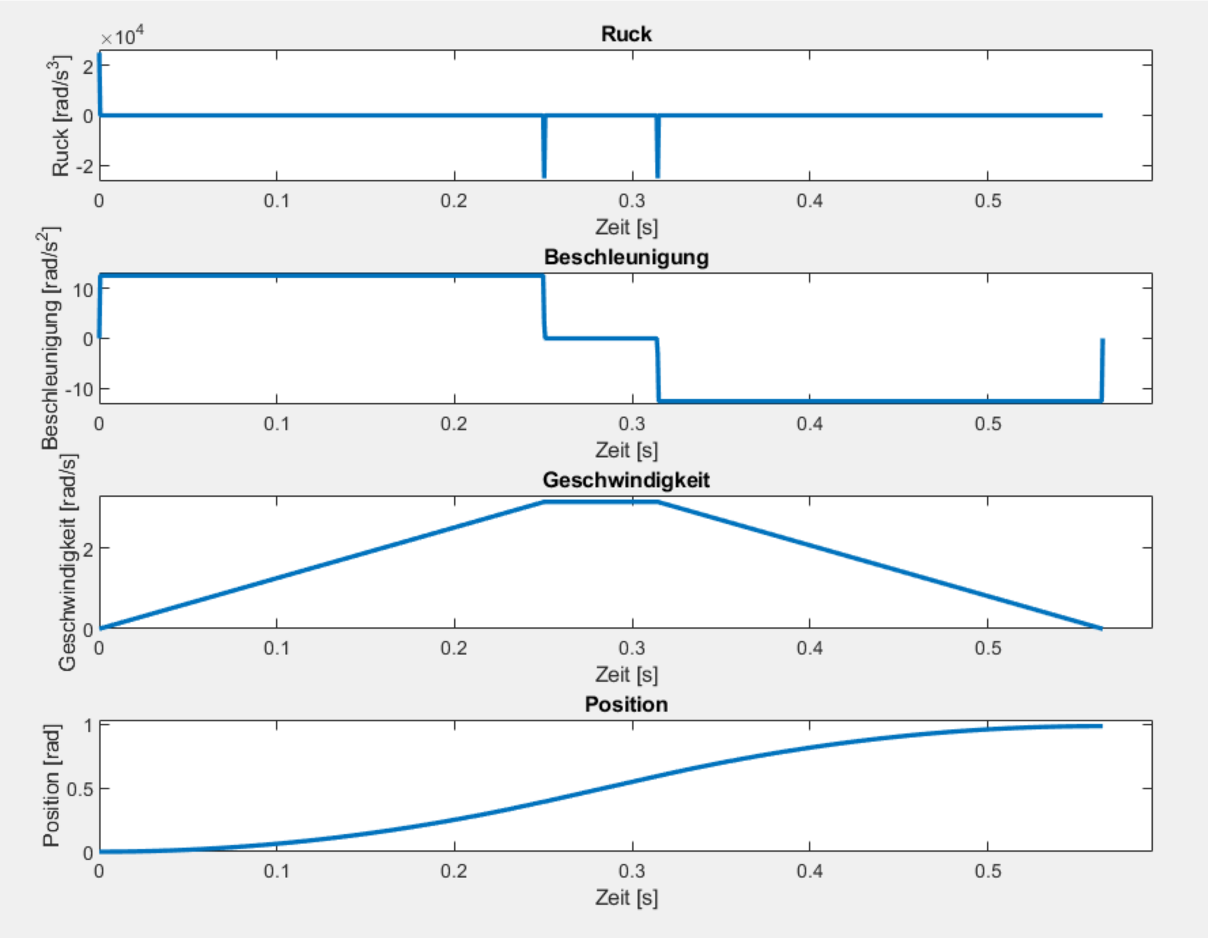
Mit Hilfe eines Plots von MATLAB erhält man für den Winkel Phi 0



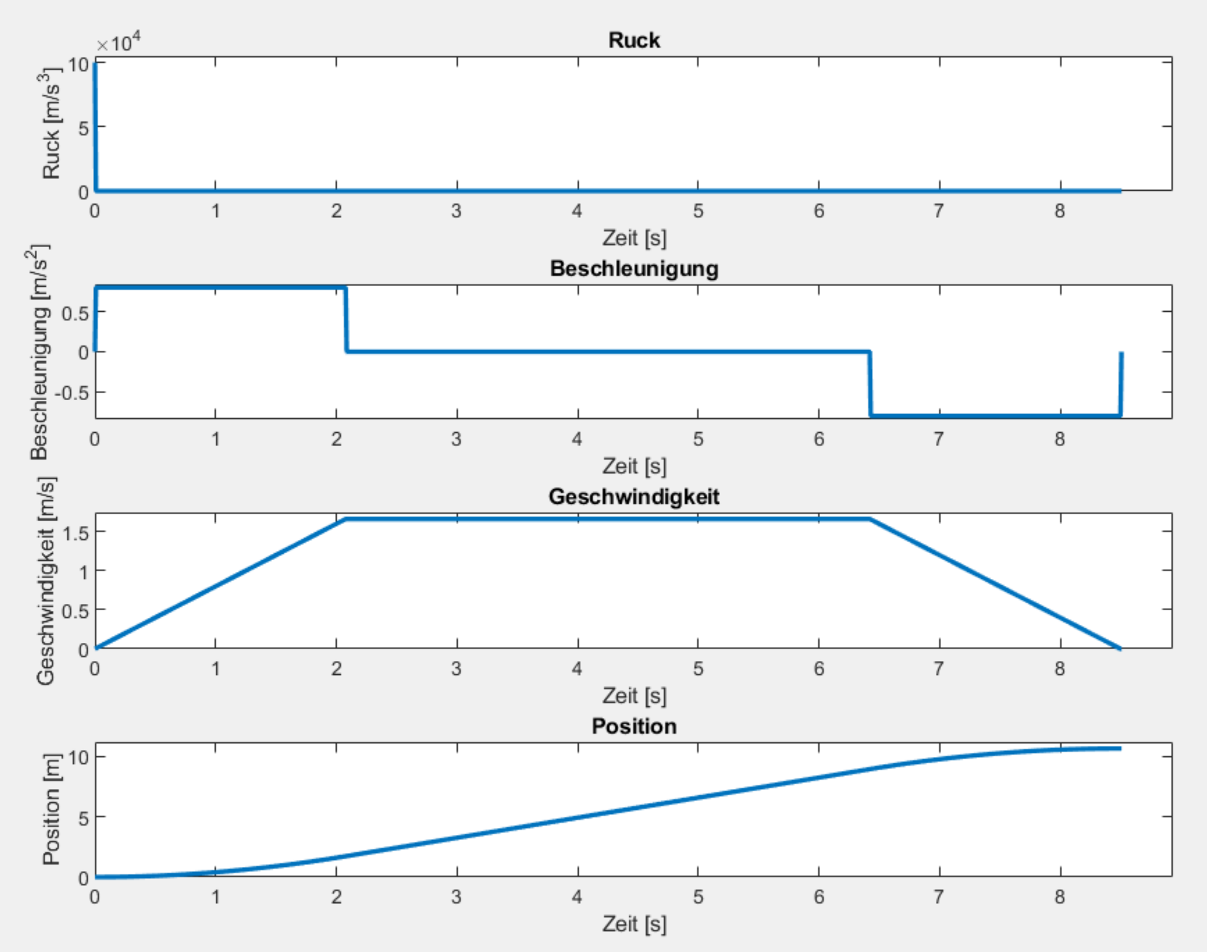
PHI 1:


Ruck
Ruck [rad/s^3]
5000
0
-5000
0
0.2
0.4
0.6
0.8
1
1.2
Zeit [s]
Beschleunigung
Beschleunigung [rad/s^2]
2
0
-2
Zeit [s]
Geschwindigkeit
Geschwindigkeit [rad/s]
1
0
Zeit [s]
Position
Position [rad]
1
0.5
0
Zeit [s]

PHI 3 :


Ruck
×10^4
Ruck [rad/s^3]
2
0
-2
0
0.1
0.2
0.3
0.4
0.5
Zeit [s]
Beschleunigung
Beschleunigung [rad/s^2]
10
0
-10
0
0.1
0.2
0.3
0.4
0.5
Zeit [s]
Geschwindigkeit
Geschwindigkeit [rad/s]
2
0
0
0.1
0.2
0.3
0.4
0.5
Zeit [s]
Position
Position [rad]
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
Zeit [s]

H3 :



4 Literaturverzeichnis

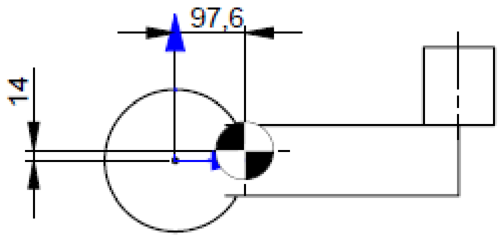
Brillowski, P. D.-I. (2004). *Einführung in die Robotik, Auslegung und Steuerung serieller Roboter.* Aachen: Shaker Verlag.

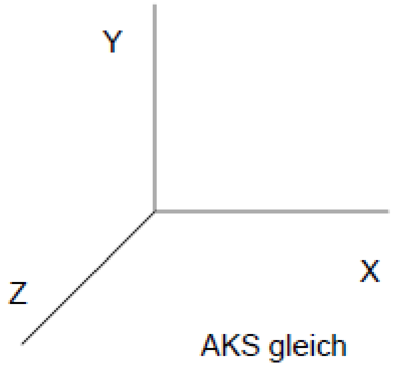
Ketterer, P. D.-I.-I. (kein Datum). *Dynamische Auslegung von Werkzeugmaschinen und Roboterachsen / Teil 2 Jacobi-Lagrange Skript*. Prof. Dr.-Ing Dipl.-Ing. Betriebswirt Gunter.

Ketterer, P. D.-I.-I. (kein Datum). *SERVOMECHANISMEN V1.3*. Prof. Dr.-Ing Dipl.-Ing. Betriebswirt Gunter.

5 Anhang

Achse 1:





AKS gleich

Masseneigenschaften von Achse1
Konfiguration: Standard
Koordinatensystem: -- Standard --

Dichte = 0.00 Kilogramm pro Kubikmillimeter

Masse = 50.32 Kilogramm

Volumen = 6492728.18 Kubikmillimeter

Oberfläche = 463275.21 Quadratmillimeter

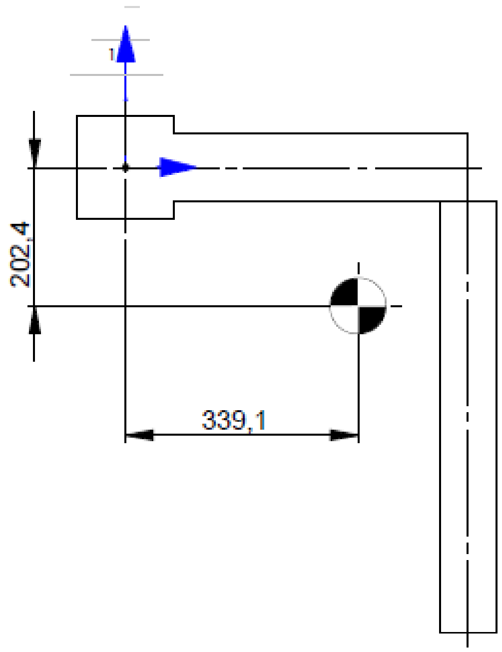
Massenmittelpunkt: (Millimeter)
X = 97.58
Y = 13.97
Z = 0.00

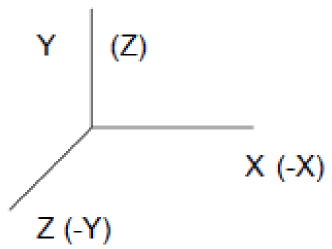
Hauptachsen der Trägheit und Hauptträgheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt.
Ix = (0.98, 0.18, 0.00) Px = 195810.03
Iy = (-0.18, 0.98, 0.00) Py = 1380285.91
Iz = (0.00, 0.00, 1.00) Pz = 1407370.05

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt und ausgerichtet auf das Ausgabekoordinatensystem. (Positive Tensornotierung wird verwendet.)
Lxx = 235288.20 Lxy = 212608.59 Lxz = 0.00
Lyx = 212608.59 Lyy = 1340807.75 Lyz = 0.00
Lzx = 0.00 Lzy = 0.00 Lzz = 1407370.05

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf das Ausgabekoordinatensystem. (Positive Tensornotierung wird verwendet.)
Ixx = 245110.61 Ixy = 281211.81 Ixz = 0.00
Iyx = 281211.81 Iyy = 1819956.99 Iyz = 0.00
Izx = 0.00 Izy = 0.00 Izz = 1896341.71

Achse 2:





Masseneigenschaften von Achse2
Konfiguration: Standard
Koordinatensystem: -- Standard --

Dichte = 0.00 Kilogramm pro Kubikmillimeter

Masse = 44.26 Kilogramm

Volumen = 5711415.44 Kubikmillimeter

Oberfläche = 584964.55 Quadratmillimeter

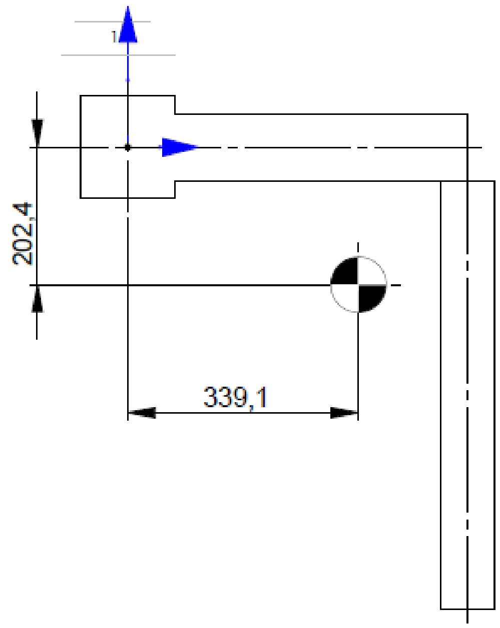
Massenmittelpunkt: (Millimeter)
X = 339.11
Y = -202.38
Z = 0.00

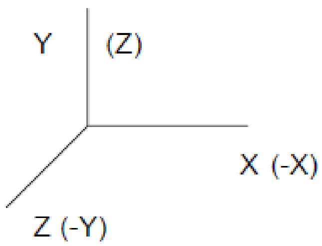
Hauptachsen der Trägheit und Hauptträgheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt.
Ix = (-0.67, 0.74, 0.00) Px = 727959.03
Iy = (-0.74, -0.67, 0.00) Py = 3627241.13
Iz = (0.00, 0.00, 1.00) Pz = 4280675.38

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt und ausgerichtet auf das Ausgabekoordinatensystem. (Positive Tensornotierung wird verwendet.)
Lxx = 2333420.99 Lxy = -1441242.18 Lxz = 0.00
Lyx = -1441242.18 Lyy = 2021779.18 Lyz = 0.00
Lzx = 0.00 Lzy = 0.00 Lzz = 4280675.38

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf das Ausgabekoordinatensystem. (Positive Tensornotierung wird verwendet.)
Ixx = 4146281.92 Ixy = -4478937.23 Ixz = 0.00
Iyx = -4478937.23 Iyy = 7111850.39 Iyz = 0.00
Izx = 0.00 Izy = 0.00 Izz = 11183607.52

Achse 3:





Masseneigenschaften von Achse2
Konfiguration: Standard
Koordinatensystem: -- Standard --

Dichte = 0.00 Kilogramm pro Kubikmillimeter

Masse = 44.26 Kilogramm

Volumen = 5711415.44 Kubikmillimeter

Oberfläche = 584964.55 Quadratmillimeter

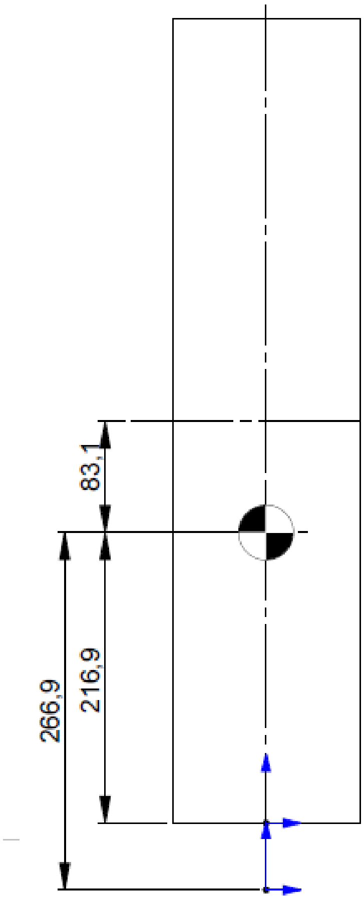
Massenmittelpunkt: (Millimeter)
X = 339.11
Y = -202.38
Z = 0.00

Hauptachsen der Trägheit und Hauptträgheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt.
Ix = (-0.67, 0.74, 0.00) Px = 727959.03
Iy = (-0.74, -0.67, 0.00) Py = 3627241.13
Iz = (0.00, 0.00, 1.00) Pz = 4280675.38

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt und ausgerichtet auf das Ausgabekoordinatensystem. (Positive Tensornotierung wird verwendet.)
Lxx = 2333420.99 Lxy = -1441242.18 Lxz = 0.00
Lyx = -1441242.18 Lyy = 2021779.18 Lyz = 0.00
Lzx = 0.00 Lzy = 0.00 Lzz = 4280675.38

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf das Ausgabekoordinatensystem. (Positive Tensornotierung wird verwendet.)
Ixx = 4146281.92 Ixy = -4478937.23 Ixz = 0.00
Iyx = -4478937.23 Iyy = 7111850.39 Iyz = 0.00
Izx = 0.00 Izy = 0.00 Izz = 11183607.52

Achse 4:



Masseneigenschaften von Achse 4
Konfiguration: Standard
Koordinatensystem: -- Standard --

Masse = 63.208 Kilogramm

Volumen = 6220354.454 Kubikmillimeter

Oberfläche = 435430.742 Quadratmillimeter

Massenmittelpunkt: (Millimeter)
X = 0.000
Y = 216.941
Z = 0.000

Hauptachsen der Trägheit und Hauptträgheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt.
Ix = (0.000, 1.000, 0.000) Px = 156677.653
Iy = (0.000, 0.000, 1.000) Py = 2926010.147
Iz = (1.000, 0.000, 0.000) Pz = 2926010.147

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf den Massenmittelpunkt und ausgerichtet auf das Ausgabekoordinatensystem.
(Positive Tensornotierung wird verwendet.)
Lxx = 2926010.147 Lxy = 0.000 Lxz = 0.000
Lyx = 0.000 Lyy = 156677.653 Lyz = 0.000
Lzx = 0.000 Lzy = 0.000 Lzz = 2926010.147

Trägheitsmomente: (Kilogramm * Quadratmillimeter)
Bezogen auf das Ausgabekoordinatensystem. (Positive Tensornotierung wird verwendet.)
Ixx = 5900768.789 Ixy = 0.000 Ixz = 0.000
Iyx = 0.000 Iyy = 156677.653 Iyz = 0.000
Izx = 0.000 Izy = 0.000 Izz = 5900768.789

Mindestens eine Komponente hat die Masseneigenschaften überschrieben:
Massepunkt TCP<1><Standard>