FH Vorarlberg
University of Applied Sciences

Robotics: Hausübung 2

Fachhochschule Vorarlberg
Master Mechatronics

Betreut von

FH-Prof. DI Dr. Robert Merz

Vorgelegt von

Roman Passler

Dornbirn, 25.06.2017

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Aufgabenstellung

Als Aufgabe sollen die Abweichungen der differentiellen Rückwärtskinematik von der exakten Bahn bei einer linearen Trajektorie bei drei verschiedenen Interpolationstakten $t\_ipo = 0.1\ s$, $0.01\ s$, $0.001\ s$ gezeigt werden.

Folgende Schritte sollen gemacht werden:

1. Finden Sie eine geeignete Trajektorie, die zur Gänze im Arbeitsraum liegt und die keine Singularitäten aufweist. Es ist empfehlenswert die Trajektorie so zu wählen, dass Sie auch nicht extrem knapp an einer Singularität vorbei fährt.

   Die Trajektorie soll mindestens 1000 mm lang sein. Vom Start zum Endpunkt sollen sich mindestens zwei Koordinaten der Position und mindestens ein Eulerwinkel ändern.

   Verwenden Sie für $v_c = 250\ mm/s$ und für $a_{max} = 250\ mm/s^2$.

   Vor der Ausführung der Trajektorie befindet sich der Roboter im Startpunkt in Ruhe ($v = 0$). Nach Ausführung der Trajektorie bleibt der Roboter im Endpunkt und verharrt in Ruhe ($v = 0$).

   Halten Sie die von Ihnen verwendeten Euler-Koordinaten $(x, y, z, \alpha, \beta, \gamma)$ des Start- und Endpunktes (in *mm* bzw. deg.) am Anfang Ihrer schriftlichen Ausarbeitung fest.

2. Skizzieren Sie den grundlegenden Ablauf, den Sie zur Berechnung der Ergebnisse verwendet haben (was müssen Sie wie und woraus berechnen?) z. B. Flowchart oder Struktogramm mit den verwendeten Matlabroutinen, etc.

3. Zeichnen Sie für jeden Achswinkel ein Diagramm, in dem Sie den zeitlichen Verlauf des exakten Achswinkels (Berechnung aus der Rückwärtskinematik) mit den drei Verläufen, die Sie durch die differentielle Rückwärtskinematik ermittelt haben, gegenüber stellen (6 Diagramme mit je 4 Kurven).

4. Zeichnen Sie für die Position und Orientierung des Endeffektors (Flansch) die Diagramme, in denen Sie den zeitlichen Verlauf der x-, y- und z-Koordinate, sowie der drei Eulerwinkel der Trajektorie mit den Verläufen, die sich mit der differentiellen Rückwärtskinematik ergeben, gegenüberstellen (6 Diagramme mit je 4 Kurven).

5. Ermitteln Sie den zeitlichen Verlauf der euklidischen Distanz zwischen der exakten Position (nur x, y, z) und den Positionen, die sich durch die differentielle Rückwärtskinematik ergeben (1 Diagramm mit 3 Kurven).

In der Tabelle 1.1 sind 2 Beispiele angeführt, wie Ihre Trajektorie aussehen könnte.

Bei Nr. 1 ändern sich alle 6 Koordinaten vom Anfangs- zum Endpunkt. Bei Nr. 2 ändern sich nur x, y und z, die Eulerwinkel bleiben konstant.

Nr.		Startpunkt	Endpunkt
1	Beispieltrajektorie	1000, 0, 1000, 0, 95, 35	0, 1000, 500, -45, 175, 0
2	Vereinfachte Trajektorie	1000, 0, 1000, 0, 175, 0	0, 1000, 500, 0, 175, 0

Tabelle 1.1: Trajektorien

2 Ausführung

2.1 Trajektorienermittlung

In Tabelle 2.1 ist die empirisch ermittelte Trajektorie dargestellt. Sie verläuft über einen Weg von mehr als 1100 mm.

Nr.		Startpunkt	
1	Eigene Trajektorie	700, -750, 500, 0, 160, -85	1000, 250, 1000, 0, 100, -25

Tabelle 2.1: Trajektorie ermittelt
Quelle: eigene Ausarbeitung

In Abbildung 2.1 ist der gefahrene Roboterpfad dargestellt.

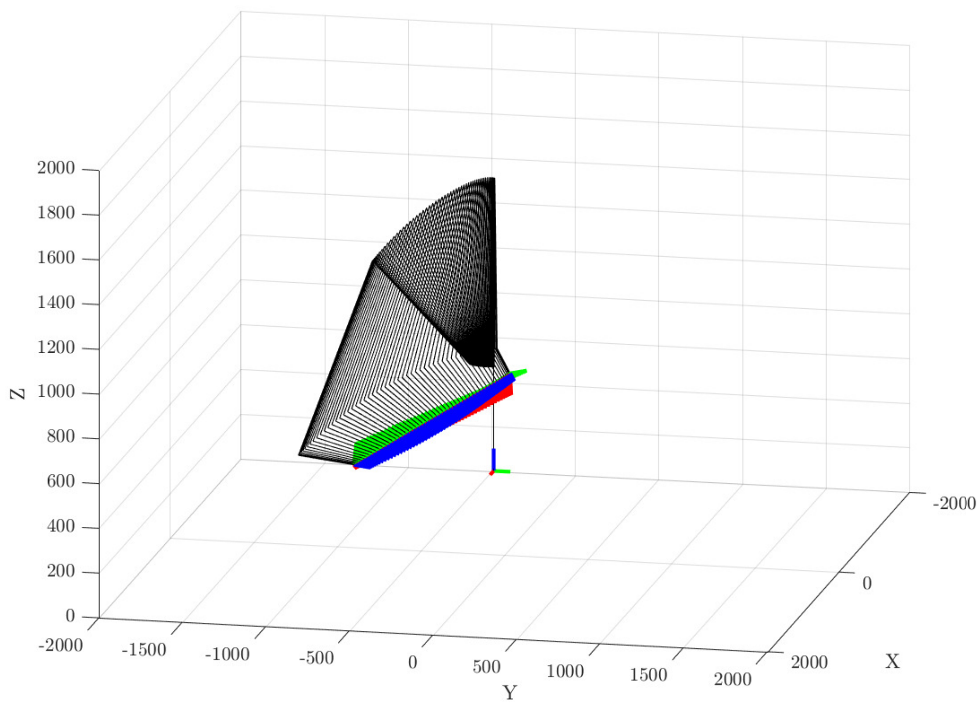

Abbildung 2.1: Roboter Pfad
Quelle: eigene Ausarbeitung

2.2 Grundlegender Ablauf des Matlab Scripts

In Abbildung 2.2 sind die verwendeten Variablen und ihre Beschreibung dargestellt.

Variablen	
ctrTipo	{Zähler für Taktzeit.}
k	{Iterationsvariable für jeden Trajektiorienschritt.}
t_ipo	{Hällt im Arrayverbund die Taktzeiten von 0.1 s, 0.01 s, 0.001 s.}
diffQ	{Differentielle Achswinkel der sechs Achsen für die gesamte Trajektorie.}
ec_diff	{Differentielle Trajektorie des Zielframes.}
ec_diffRe	{Vom analyticalQ zurück gerechnete Trajektorie des Zielframes.}
v0	{Enthällt die Geschwindigkeiten des Zielframes für die gesamte Trajketorie.}
analyticalQ	{Mit der analytisch ermittelten Rückwärtskinematik gerechneten Achswinkel.}
qDot	{Ableitungen der differentiellen Achswinkel für die gesamte Trajektorie.}
ec	{Solltrajektorie.}

Abbildung 2.2: Matlab Script: verwendete Variablen
Quelle: eigene Ausarbeitung

In Abbildung 2.3 ist die Programmlogik dargestellt und beschrieben. Der Matlab Code für die Berechnung ist in Anhang A zu finden.

- ctrTipo $\leftarrow 1$
- ctrTipo $<=$ length(t_ipo)
  - create_lin_seg_list()
    Beschleunigungszeiten und Beschleunigung berechnen.
    create_lin_intvec()
    für jeden Zeitschritt t s, x, a berechnen.
    ec $=$ create_lin_path()
    die Solltrajektorie berechnen.
  - Variablen Speicher allozieren $\rightarrow$ ist schneller!
  - k $\leftarrow 1$
  - k $\leq$ length(ec)
    - tg $=$ xyzabc_2_t(ec $_k$) *Zielframe berechnen von der Trajektorie.*
    - analyticalQ $_k$ $=$ irb4600_rk(tg, robot)
      Analytische Rückwärtskinematik zum Vergleich berechnen.
    - coor_wRe $=$ fk_craig(analyticalQ $_k$, robot) *Zurückrechnen auf den Zielframe*
    - ec_diffRe $_k$ $=$ t_2_xyzabc(coor_wRe, 1)
      Zurückrechnen vom Zielframe auf die Trajektorie.
    - k $== 1$
      - FALSCH: $\varnothing$
      - WAHR:
        - v0 $_k$ $= 0$
        - vNext $_k$ $= 0$
        - diffQ $_k$ $=$ analyticalQ $_k$
          Erste Achswinkel von der Rückwärtskinematik nehmen, da keine Wegmessung in der Simulation integriert ist.
    - coor_w=fk_craig(diffQ $_k$,robot)
      Zielframe von den differentiellen Achswinkeln berechnen.
    - ec_diff $_k$ =t_2_xyzabc(coor_w, 1)
      mithilfe des Zielframes der Differentiellen Rückwärtskinematik, die differentielle Trajektorie bestimmen.
    - k > 1
      - FALSCH: $\varnothing$
      - WAHR:
        - vNext $_k$ $=$ ec $_k$ $-$ ec $_{k+1}$ t_ipo (ctrTipo)
          Differentiell die Geschwindigkeiten ausrechnen für $v_x, v_y, v_z, \dot{\alpha}, \dot{\beta}, \dot{\gamma}$ mittels der Solltrajektorie.
    - omega $=$ deg2rad(abc_t_w(ec $_k$, vNext $_k$ $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$))
      Mithilfe der aktuellen Position in x, y, z und $\dot{\alpha}, \dot{\beta}, \dot{\gamma}$ berechnet sich omega in Radiant.
    - v0 $_k$ $=$ vNext $_{k+1}$ & omega *v0 richtig zusammenstellen aus x, y, z und omega.*
    - qDot $_k$ $=$ rad2deg(irb4600_jakobi(diffQ $_k$)$^{-1}\cdot$ v0 $_k$)
      mithilfe der IRB4600 Jakobi und v0 die Ableitungen der Achswinkel berechnen.
    - diffQ $_{k+1}$ $=$ diffQ $_k$ $+$ qDot $_k$ $\cdot$ t_ipo (ctrTipo)
      differentielle Achswinkel für nächsten Zyklus berechnen.
    - $x_{dif} = ($ ec_diff $(x_k) -$ ec $(x_{k-1}))$
      $y_{dif} = ($ ec_diff $(y_k) -$ ec $(y_{k-1}))$
      $z_{dif} = ($ ec_diff $(z_k) -$ ec $(z_{k-1}))$
      euklAbstand$=\sqrt{x_{dif}^2 + y_{dif}^2 + z_{dif}^2}$
      euklidischer Abstand berechnen.
    - k $++$
  - ctrTipo $++$

Abbildung 2.3: Matlab Script Logik
Quelle: eigene Ausarbeitung

2.3 Verlauf der Achswinkel

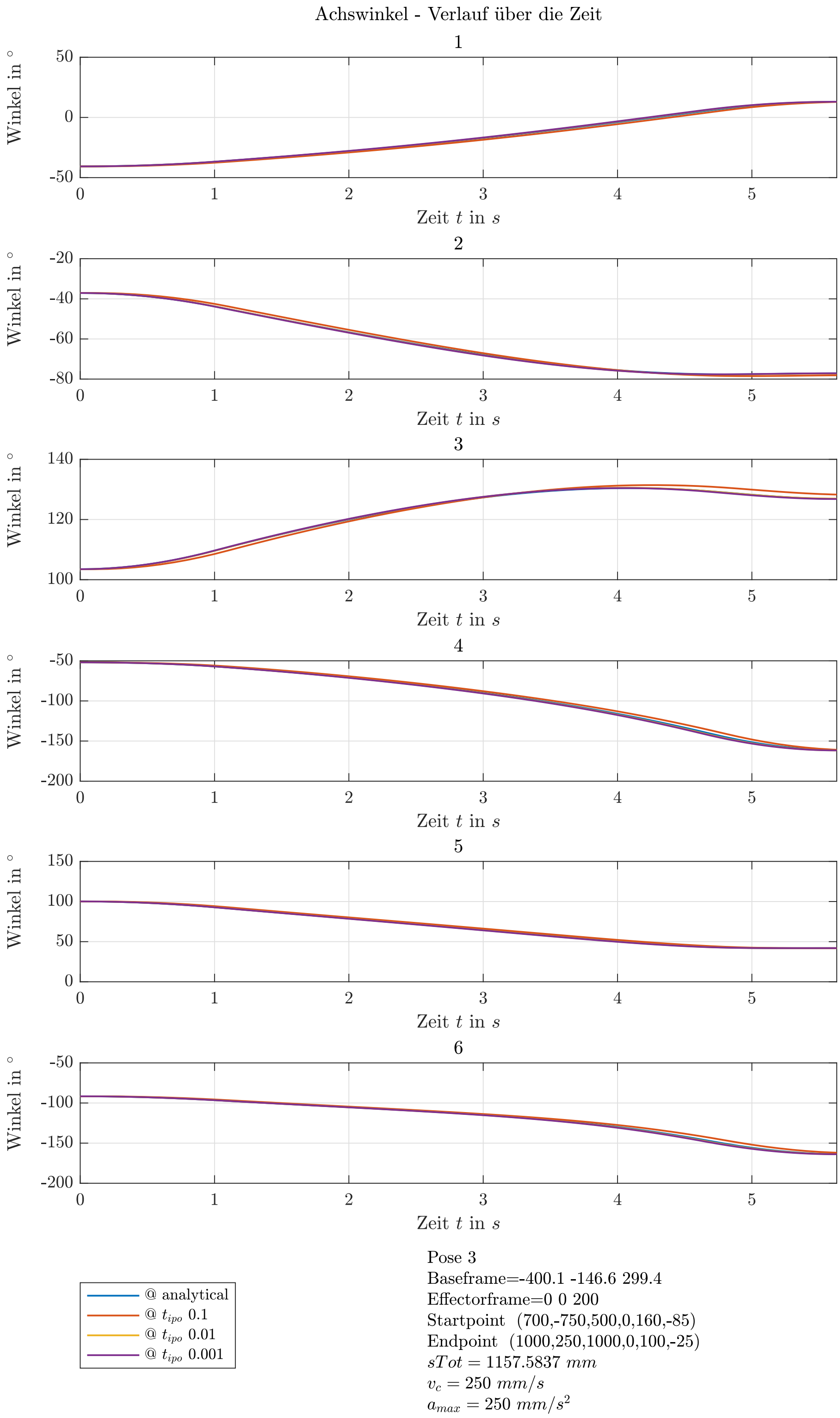

Abbildung 2.4: Achswinkel - Verlauf über die Zeit
Quelle: eigene Ausarbeitung

2.4 Verlauf der Eulerkoordinaten

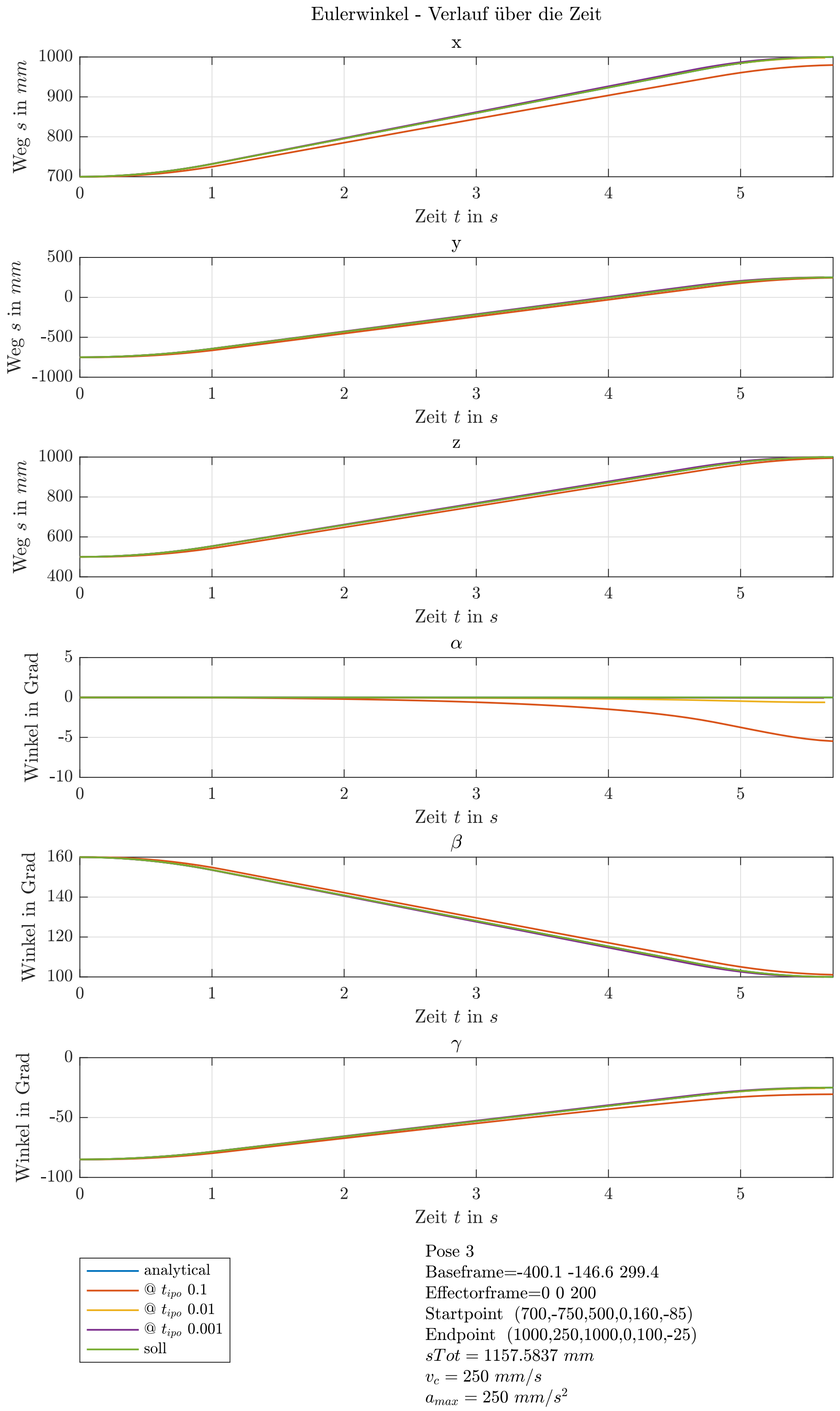

Abbildung 2.5: Eulerkoordinaten - Verlauf über die Zeit
Quelle: eigene Ausarbeitung

2.5 Euklidischer Abstand

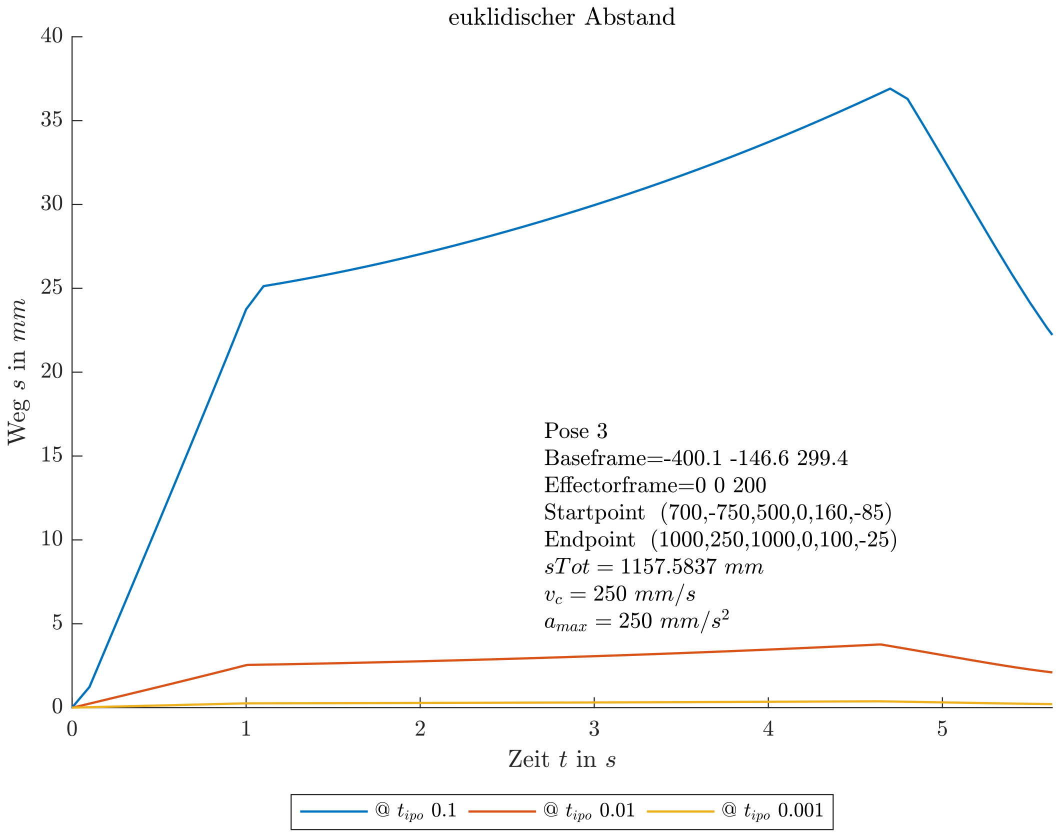

Abbildung 2.6: euklidischer Abstand
Quelle: eigene Ausarbeitung

Anhang

A Matlab Code

```
t_ipo=[0.1 0.01 0.001]';
amax=250;
vc=250;
pose=3;
t_2_xyzabcPose = 2;

e{1}=[700, -750, 500, 0, 160, -85,0];
e{2}=[1000, 250, 1000, 0, 100, -25,vc];

robot = irb4600_robot();
robot.bas = trans(-400.1,-146.6,299.4);
robot.eff = trans(0,0,200);
erease = 0;

for ctrTipo=1:length(t_ipo)
    for ii = 1:length(e)-1
        %% Pfad generieren
        [tx,ax] = create_lin_seg_list(e{ii}(1:6),e{ii+1}
           (1:6),e{ii+1}(7),amax,t_ipo(ctrTipo));
        [t,a,v,s] = create_lin_intvec(tx,ax,t_ipo(ctrTipo)
           );
        ec = create_lin_path(e{ii}(1:6),e{ii+1}(1:6),s);
        % init variables -> should be faster...
        actualEC =  cell(1,length(ec{1}));
        analyticalQ = actualEC;
        realQGG = actualEC;
        qDot = actualEC;
        trajektSoll = actualEC;
        trajektIst = actualEC;
        euklAbstand = zeros(length(ec{1}),1);
        v0 = cell(1,length(ec{1})+1);
        difQ = v0;
        %% Rueckwaertskinematik anwenden
        tic
        for kk = 1:length(ec{1})
            tg = xyzabc_2_t(ec{1}(kk),ec{2}(kk),ec{3}(kk),
               ec{4}(kk),ec{5}(kk),ec{6}(kk));
            analyticalQ{kk} = irb4600_rk(tg,robot.bas,
               robot.eff,pose)';
```

```
            coor_wRe=fk_craig(analyticalQ{kk},robot);
            [ec_diffRe{1}(kk),ec_diffRe{2}(kk),ec_diffRe{3}(kk),ec_diffRe{4}(kk),ec_diffRe{5}(kk),ec_diffRe{6}(kk)] = t_2_xyzabc(coor_wRe, t_2_xyzabcPose);
            if kk == 1
                v0{kk}(:,1) = [0 0 0 0 0 0]';
                vNext = [0 0 0 0 0 0];
                % Die erste Position faken, da keine Wegmessung integriert ist
                difQ{kk} = analyticalQ{kk};
            end
            coor_w=fk_craig(difQ{kk},robot);
            [ec_diff{1}(kk),ec_diff{2}(kk),ec_diff{3}(kk),ec_diff{4}(kk),ec_diff{5}(kk),ec_diff{6}(kk)] = t_2_xyzabc(coor_w, t_2_xyzabcPose);
            trajektSoll{kk}  = [ec{1}(kk) ec{2}(kk) ec{3}(kk) ec{4}(kk) ec{5}(kk) ec{6}(kk)]';
            trajektIst{kk}  = [ec_diff{1}(kk) ec_diff{2}(kk) ec_diff{3}(kk) ec_diff{4}(kk) ec_diff{5}(kk) ec_diff{6}(kk)]';
            % dieses Delta rechnen ist zwar in der Simulation sinnlos,
            % aber bei einer realen Wegmessung sollte es so gemacht
            % werden.
            delta = trajektSoll{kk} - trajektIst{kk};
            actualEC{kk} = trajektIst{kk} + delta;
            if kk > 1
                vNext = (actualEC{kk}-actualEC{kk-1})/t_ipo(ctrTipo);
            end
            % output is degree, but we ned rad!
            omega = deg2rad(abc_t_w(actualEC{kk}, vNext(4), vNext(5), vNext(6)));
            v0{kk}(:,1) = [vNext(1); vNext(2); vNext(3); omega];
            qDot{kk}(:,1) = rad2deg((irb4600_jakobiCross(difQ{kk}, robot.bas, robot.eff))\v0{kk}(:));
            difQ{kk+1}(:,1) = difQ{kk} + qDot{kk}(:,1) * t_ipo(ctrTipo);
            euklAbstand(kk,1)=sqrt((ec{1}(kk)-ec_diff{1}(kk))^2+(ec{2}(kk)-ec_diff{2}(kk))^2+(ec{3}(kk)-ec_diff{3}(kk))^2);
        end
    end
    toc
```

```
    %%
    tic
    euklAbstandList{ctrTipo} = euklAbstand;
    ec_diffReList{ctrTipo} = ec_diffRe;
    ec_diffEcList{ctrTipo} = ec;
    ec_diffList{ctrTipo} = ec_diff;
    realQList{ctrTipo} = analyticalQ;
    qqList{ctrTipo} = difQ;
    tList{ctrTipo} = t;
    toc
    %%
    if ctrTipo > 0
        %break;
    end
end
```

Listing A.1: Matlab Skript