Stabilisierung eines Pendels auf einem Scara Roboter

29. November 2012

1 Das Inverse Pendel

Das inverse Pendel ist ein Pendel mit dem Schwerpunkt oberhalb der Achse. Das Pendel befindet sich in seinem höchsten Punkt in einer instabilen Ruhelage. Das inverse Pendel ist eine der Standardaufgaben der Regelungstechnik für die Stabilisierung einer instabilen Regelstrecke.

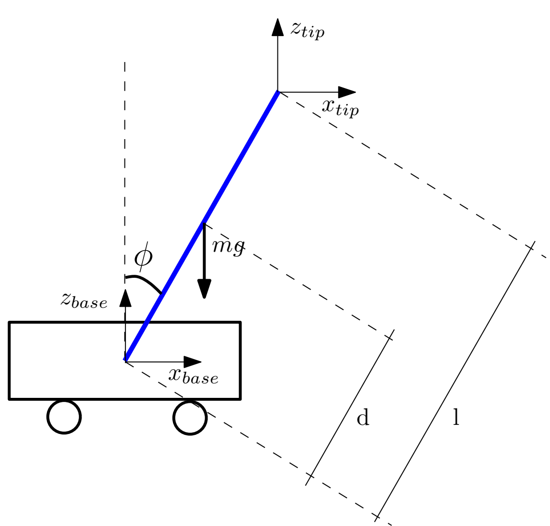

Abbildung 1: *Inverse Pendel*

Ein Standardbeispiel für ein inverses Pendel ist ein Wagen mit einem darauf montierten inversen Pendel (Bild 1). Die Pendelbewegung kann durch die horizontale Bewegung des Wagens beeinflusst werden.

Der Dynamische vom Pendel kann durch die Gleichung 1 beschrieben werden. Die Winkelbeschleunigung ist Abhängig von der x Koordinate an der Basis vom Pendel.

Das Pendel Modell wurde durch diese Gleichung implementiert.

$$\ddot{\phi} = \frac{d(-\ddot{x_{base}} \cos\phi + g \sin\phi)}{d^2 + r^2} - f_p \dot{\phi} \tag{1}$$

Wenn J die Trägheit vom Pendel und m seine Masse sind, ist $r = \sqrt{\frac{J}{m}}$.

f_p ist einen Reibungsfaktor.

Aus diesem Modell kann man die Istwerte von x_{base} Koordinaten und vom Winkel ϕ.

Der Istwert der Position vom Pendels Endpunkt wird dann durch die folgende Gleichung berechnet:

$$x_{tip} = x_{base} + l \sin\phi \tag{2}$$

Der Winkel kann auch durch die folgende Gleichung berechnet werden:

$$\phi = \arctan \frac{\ddot{x_{tip}}}{g} \tag{3}$$

Diese Gleichung ermöglicht, eine Beziehung zwischen ϕ und $\ddot{x_{tip}}$ zu finden.

1.1 Das Inverse Pendel 3-D

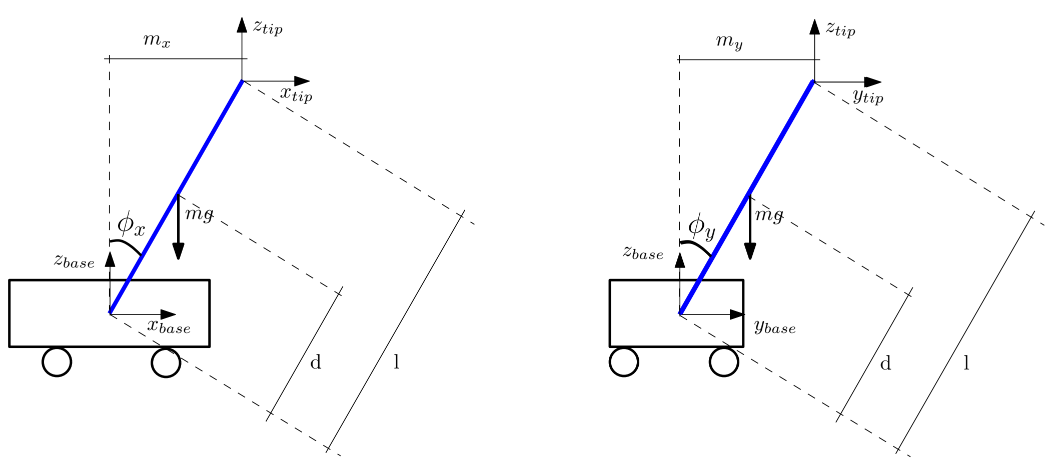

Abbildung 2: *Inverse Pendel 3-D*

Zur Verfügung steht ein Sensor, der durch vier Hall Sensoren den Neigungswinkel eines Pendels in 3-D liefern kann. Das Pendel ist mit einem Magnet gestattet. Das Modell vom Inversen Pendel wurde daher auf den 3-D Fall generalisiert. In Bild 3 wird das Messprinzip vom Sensor gezeigt. Die vier Sensoren (S1, S2, S3, S4) messen einen Wert zwischen 0 und 1, abh‰ngig von wo sich das Pendel befindet.

Das System bekommt dann zwei Werte m_x und m_y. In diesem Fall haben wir die folgenden Gleichungen:

$$\sqrt{m_x^2 + m_y^2} = l \sin \phi \tag{4}$$

$$m_x = l \sin \phi \cos \alpha \tag{5}$$

$$m_y = l \sin \phi \sin \alpha \tag{6}$$

Bei der Regelungskonzept wurde es entschieden, die zwei Winkel ϕ_x und ϕ_y und die x und y Koordinaten des Pendelstips durch die Gleichungen 2 und 3 unabhängig zu regeln.

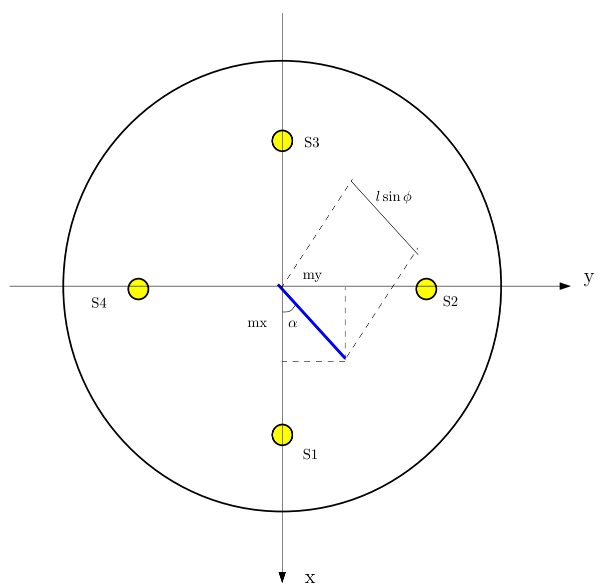

Abbildung 3: *Inverse Pendel Sensor Beschreibung*

2 Der SCARA Roboter

Der SCARA Roboter erledigt Montageaufgaben. Er kann dabei zwischen klassischer Bahnregelung und Kraft- Positionsregelung umschalten.

Ein SCARA-Roboter besitzt in der Regel vier Achsen und vier Freiheitsgrade. Sämtliche Achsen sind als serielle Kinematik ausgeführt, d.h. der Koordinatenursprung der folgenden Achse ist abhängig von der Position der vorhergehenden. Bei einem SCARA-Roboter sind die erste und zweite Achse rotatorischer Natur, die dritte und die vierte Achse sind vielfach aus einem Bauelement hergestellt (der Kugelrollspindel), und erlauben eine rotatorische und eine Linearbewegung. Das Werkzeug des Roboters wird am unteren Ende der Z-Achse montiert.

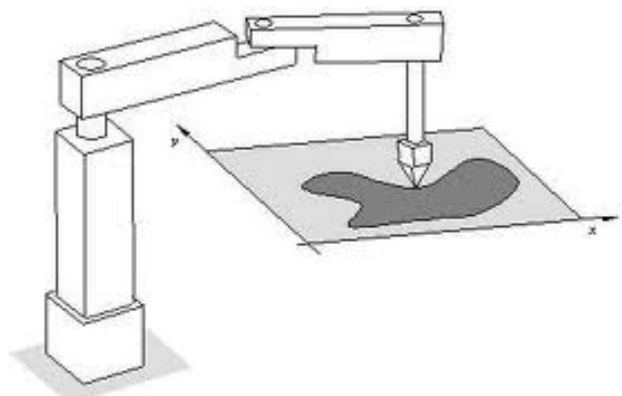

Abbildung 4: *SCARA Roboter*

2.1 Direkt Kinematik

Der SCARA Roboter besteht aus vier Achsen und wird mit verschiedenen Transformationen beschrieben. Aus diesen Matrizen ergibt sich eine Vorwärtstransformation. Die inverse Transformation wird durch eine geometrische Beschreibung hergeleitet. Diese Transformationen werden benötigt, um die Koordinaten zwischen kartesischen und Maschinenkoordinaten umzuwandeln.

Durch die vier Matrizen kann der Industrieroboter zu einer kinematischer Kette zusammen gebaut werden. Die vier Matrizen resultieren aus der Anzahl von Achsen. Bei jedem Koordinatensystem wird eine neue Transformationsmatrize gebildet.

Achse 1:
Bei der ersten Transformation handelt es sich um eine Rotation um die z_0 Achse. Außerdem verschiebt man das Koordinatensystem um die Länge l_1 in der positiven x_0 Achse.

$$T_1^0 = \begin{bmatrix} cos(\phi_1) & -sin(\phi_1) & 0 & l_1 cos(\phi_1) \\ sin(\phi_1) & cos(\phi_1) & 0 & l_1 sin(\phi_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Achse 2:
Bei der ersten Transformation handelt es sich um eine Rotation um die z_1 Achse. Außerdem verschiebt man das Koordinatensystem um die Länge l_2 in der positiven x_1 Achse.

$$T_2^1 = \begin{bmatrix} cos(\phi_2) & -sin(\phi_2) & 0 & l_2 cos(\phi_2) \\ sin(\phi_2) & cos(\phi_2) & 0 & l_2 sin(\phi_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Achse 3:
Bei der ersten Transformation handelt es sich um eine Verschiebung der Länge z in der positiven z_2 Achse.

$$T_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Achse 4:
Bei der ersten Transformation handelt es sich um eine Verdrehung des Endeffektor.

$$T_4^3 = \begin{bmatrix} cos(\phi_4) & -sin(\phi_4) & 0 & 0 \\ sin(\phi_4) & cos(\phi_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Kinematische Kette:
Aus diesen vier Matrizen entsteht eine neue Matrize. Durch diese können die Koordinaten für den Endefekktors herausgelesen werden.

$$T_4^0 = T_1^0 T_2^1 T_3^2 T_4^3 \tag{11}$$

Diese Matrix beschreibt die xyz Koordinaten und die Lage α. Dies ist der Winkel, um welchen der Endeffektor gedreht werden kann.

$$\begin{bmatrix} x \\ y \\ z \\ \alpha \end{bmatrix} = \begin{bmatrix} l_1 cos(\phi_1) + l_2 cos(\phi_1 + \phi_2) \\ l_1 sin(\phi_1) + l_2 sin(\phi_1 + \phi_2) \\ -d_3 \\ \phi_1 + \phi_2 - \phi_4 \end{bmatrix} \tag{12}$$

2.2 Inverse Kinematik

Die inverse Trafomastion kann bei einem SCARA Roboter noch händisch berechnet werden. Jeder Punkt kann mit dem Roboterarm links oder rechts angefahren werden.

$$c^2 = x^2 + y^2 \tag{13}$$

$$c^2 = \frac{l_1^2 + l_2^2 - x^2 - y^2}{2 l_1 l_2} \tag{14}$$

Aus der Formel 13 und 14 berechnet sich der Winkel θ:

$$cos(\theta) = \frac{l_1^2 + l_2^2 - x^2 - y^2}{2 l_1 l_2} \tag{15}$$

$$180 = \theta + \phi_2 \tag{16}$$

$$cos(\theta) = cos(180^\circ - \phi_2) = -cos(\beta) \tag{17}$$

Somit können die Winkel errechnet werden:

$$\phi_2 = \pm arccos(\frac{-l_1^2 - l_2^2 + x^2 + y^2}{2l_1 l_2}) \tag{18}$$

$$\phi_1 = arctan(\frac{y}{x}) \pm arccos(\frac{-l_1^2 - l_2^2 + x^2 + y^2}{2l_1\sqrt{x^2 + y^2}}) \tag{19}$$

Über die verfahrene Höhe z und den Winkel ϕ_4 kann auf d_3 und d_4 zurückgeschossen werden.

$$d_3 = z \tag{20}$$

$$\phi_4 = \phi_1 + \phi_2 - d_4 \tag{21}$$

2.3 Jacobi Matrix

Das Jacobimatrix $J(q)$ definiert die folgende Beziehung zwischen dem Vektor der Gelenkegeschwindigkeiten $\dot{q}$ und dem Vektor der kartesischen Geschwindigkeiten $\dot{x}$:

$$\dot{x} = J(q)\dot{q} \tag{22}$$

Der SCARA hat ein $6x4$ Jacobimatrix, da er nur vier Freiheitsgraden hat. Da die Gelenke 1, 2 und 4 Drehgelenke und Gelenk 4 prismatisch sind,und da $O_4 - O_3$ parallel zu z_3 Axis ist, hat der Jacobimatrix die Form:

$$J = \begin{bmatrix} z_0 x(O_4 - O_0) & z_1 x(O_4 - O_1) & z_2 & 0 \\ z_0 & z_1 & 0 & z_3 \end{bmatrix} \tag{23}$$

Die Ursprünge der Referenzsysteme sind so definiert:

$$O1 = \begin{bmatrix} l_1 cos(\phi_1) \\ l_1 sin(\phi_1) \\ 0 \end{bmatrix} \tag{24}$$

$$O2 = \begin{bmatrix} l_1 cos(\phi_1) + l_2 cos(\phi_1 + \phi_2) \\ l_1 sin(\phi_1) + l_2 sin(\phi_1 + \phi_2) \\ 0 \end{bmatrix} \tag{25}$$

$$O4 = \begin{bmatrix} l_1 cos(\phi_1) + l_2 cos(\phi_1 + \phi_2) \\ l_1 sin(\phi_1) + l_2 sin(\phi_1 + \phi_2) \\ d_3 - d_4 \end{bmatrix} \tag{26}$$

Die Jacobi Matrix wird dann:

$$J = \begin{bmatrix} -l_1 sin(\phi_1) - l_2 sin(\phi_1 + \phi_2) & -l_2 sin(\phi_1 + \phi_2) & 0 & 0 \\ l_1 cos(\phi_1) + l_2 cos(\phi_1 + \phi_2) & l_2 cos(\phi_1 + \phi_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} \tag{27}$$

2.4 Massenmatrix

In der analytische Mechanik wird die Massenmatrix benutzt, um Bewegungsgleichungen mit generalisierten Koordinaten zu lösen. In diesem Fall ist die Massenmatrix eine diagonale Matrix.

$$M = \begin{bmatrix} J_1 i_1^2 & 0 & 0 & 0 \\ 0 & J_2 i_2^2 & 0 & 0 \\ 0 & 0 & J_3 i_3^2 & 0 \\ 0 & 0 & 0 & J_4 i_4^2 \end{bmatrix} \tag{28}$$

und $i_{1\ldots4}$ sind die Überetzungsfaktoren für jeden Gelenk.

3 Regelungskonzept

Bei einer Roboterregelung gelten generell folgende Anforderungen:

- Der Regler muss in jedem Fall stabil sein
- Stöörungen und Lastmomente sollen komplett ausgeregelt werden
- Die Geschwindigkeit soll begrenzt werden können
- Der Regler sollte in jeder Position des Roboters gleich straff sein

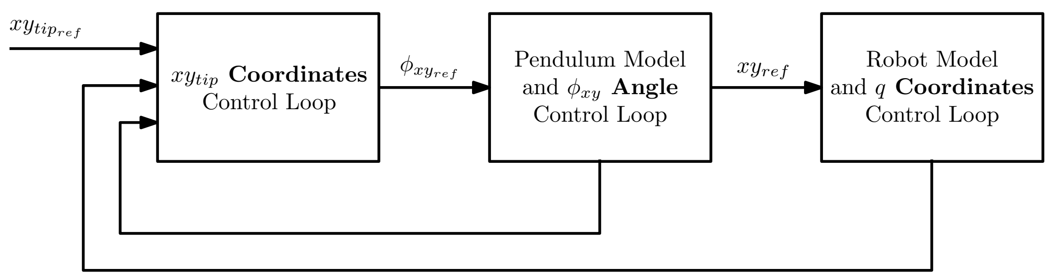

Abbildung 5: *SCARA Regler Schema*

Das Ziel ist, ein in dem SCARA Roboter montierte Pendel stabilisieren zu können. Man will die Position den Endepunkt vom Pendel regeln. Daher wird einen externen Reglerkreis gebaut, der den Wert x_{tip} regeln kann. Der Ausgang von diesem Block ist der Sollwinkel für den Pendel, der auf Null geregelt werden muss. Dieser Sollwert wird im Pendel Regelungskreis benutzt und die Ausgang von diesem Regler sind die kartesischen Koordinaten des End Effektors, die die Sollwerte für den Roboter sind.

Der Roboter Modell bekommt die Sollwerte der kartesischen Koordinaten und regelt die Gelenkekoordinaten q durch den innersten Regelungskreis.

Die Ausgänge vom Pendelregelungskreis ud Roboterregelungskreis werden vom x_{tip} benutzt, um den x_{tip} Istwert zu rechnen.

3.1 Regelungsstruktur und Regelungsparameter

Für eine straffe und schnelle Regelung muss eine Kaskadenregelung eingesetzt werden. Dazu müssen die inneren Zustände bekannt sein. Die Position der

Motoren ist durch den Encoder bekannt und die Geschwindigkeit kann durch Ableiten der Position gefunden werden.

Der Positionsollwert vom gesamten System ist die Position in Kartesischen Koordinaten vom äussersten Punkt des Pendels.

Die Parameter der Reglern, beziehungsweise k_p und k_v wurden nach den folgenden Formeln erhalten:

$$k_p = \omega_0^2 \tag{29}$$

$$k_v = 2D\omega_0 \tag{30}$$

, wo $\omega_0 = 2\pi f$ ist die ungedämpfte Eigenfrequenz und f ist die Frequenz des Systems. D ist der Dämpfungsfaktor und ist in diesem Fall an 0.7 gesetzt.

Die Parameter vom Regler sind auf 1 ms Regeltakt optimiert. Der Regeltakt muss immer grösser als die mögliche Totzeitensein, um eine stabile Regelung zu garantieren.

	$PDxy_{tip}$	$PD\phi_{xy}$	PDq
ω_0	1	10	600
k_p	1	100	36e4
k_v	1.4	14	840

Tabelle 1: *Regler Parameter Einstellungen*

3.2 Das Pendel Modell

Im Bild 6 wird der innerste Regelungskreis beschrieben. Der Winkelssollwert wird mit dem Istwert vom Pendel Modell vergliechen und dieser Fehler ist der Eingang vom PD Regler.

Die Regler Parameter wurden durch die Gleichungen 31 und 32 berechnet. Der Dämpfungsfaktor ist in diesem Fall $D = 1$, und die ungedämpfte Eigenfrequenz $\omega_0 = 50$.

Der Ausgang vom Regler ist eine Beschleunigung. Durch den inversen Pendelmodell kann man die kartesische Beschleunigung kriegen und diese Beschleunigung ist der Eingang vom Pendel.

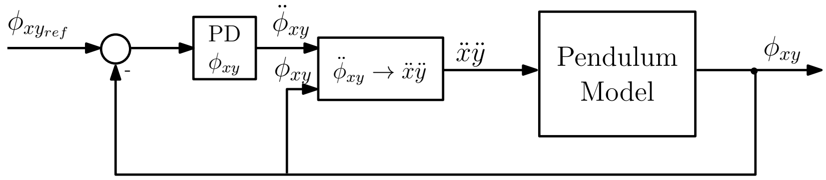

Abbildung 6: *Pendel ϕ_{xy} Regelung Schema*

Im Bild 7 wird einen äusseren Regelungskreis geschlossen. Dieser Kreis regelt die Position vom Pendels Endpunkt xy_{tip}. Der Positionsfehler wird in einem PD REgler bearbeitet. Auch diese Regler Parameter wurden durch die Gleichungen 31 und 32 berechnet. Der Dämpfungsfaktor ist in diesem Fall $D = 1$, und die ungedämpfte Eigenfrequenz $\omega_0 = 5$.

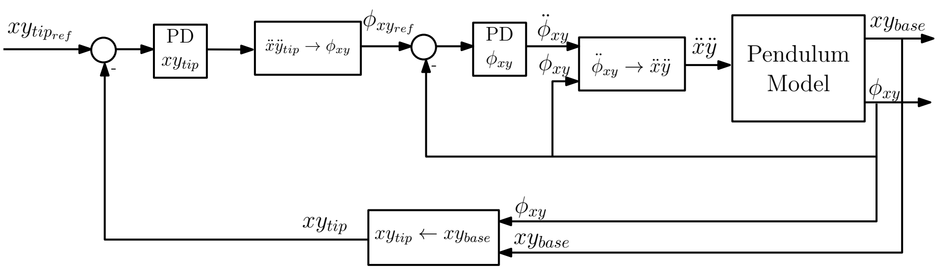

Abbildung 7: *Pendel xy_{tip} Regelung Schema*

Die Gleichung 3 ermöglicht, den Winkelsollwert für den innersten Kreis zu kriegen und die Gleichung 2 ermöglicht schlussendlich die Istposition vom Endpunkt durch die Istposition vom Basispunkt zu rechnen.

3.3 SCARA Roboter Modell und Gelenkregelung

Ein *Path Planner* generiert die Sollwerte der kartesischen Koordinaten vom Roboter. Diese Koordinaten werden durch die Inverse Kinematik auf Gelenkekoordinaten übersetzt. Der Fehler zwischen Gelenkekoordinaten Sollwert und Istwert bestehen die Eingänge vom PD Regler.

Der Dämpfungsfaktor ist in diesem Fall $D = 1$, und die ungedämpfte Eigenfrequenz $\omega_0 = 2\pi f = 50$. Der PD Regler wurde implementiert,so dass:

$$k_p = \omega_0^2 \tag{31}$$

$$k_v = 2D\omega_0 \tag{32}$$

Die Ausgänge vom Regler sind die Gelenkbeschleunigungen. Die Massenmatrix ermöglicht, aus diesen Beschleunigungen, die generalisierte Kräfte vom Roboter zu rechnen.

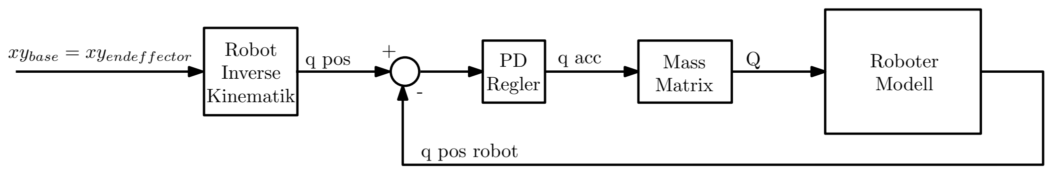

Abbildung 8: *SCARA Roboter Regelung Schema*

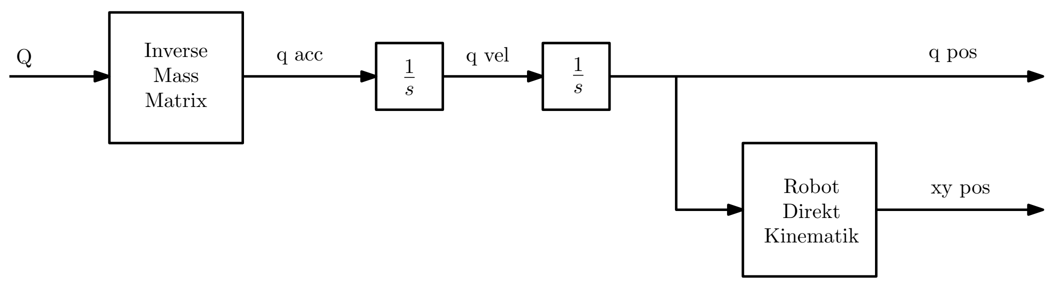

Abbildung 9: *SCARA Roboter Schema*

Im Roboter Modell ermöglicht die *Inverse Massenmatrix* die Gelenkbeschleunigungen vom Roboter zu rechnen. Durch zwei Integratoren kriegt man die Gelenkkoordinaten und durch die *Direkte Kinematik* kann man schliesslich die kartesischen Koordinaten vom Roboter berechnen.

4 Simulink Implementierung

Das gesamte Simulink Schema ist in Bild 10. Man kann die drei Reglerkreise, um die Position x_{tip}, den Winkel vom Pendel ϕ und die Gelenkekoordinaten q zu regeln.

Ein Sample Time von 1 ms wurde eingestellt und die Parameter der Reglern wurden auf diesen Wert optimiert.

Man kann in Matlab im Fenster **Simulation/Configuration Parameters** den Sample Time einstellen.

Alle Blöcke werden dann dieses Wert übernehmen (Der Wert -1 in jedem Block Properties Fenster "heisst ïnherited").

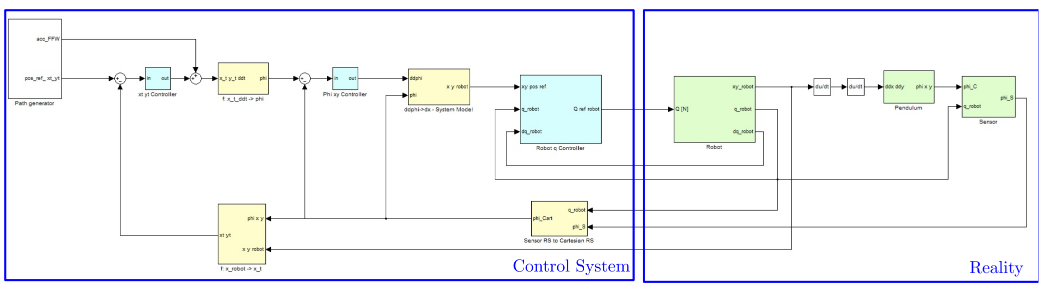

Abbildung 10: *Simulink gesamte Schema*

4.1 Pendel Modell und Regelung

Die Gleichung 1 wurde in Simulink für die x und y Koordinaten wie in Bild 11 implementiert. Bilder 12 und 13 zeigen die Implementierung von den zwei Reglerkreise, die ϕ_{xy} Regelung und die xy_{tip} Regelung.

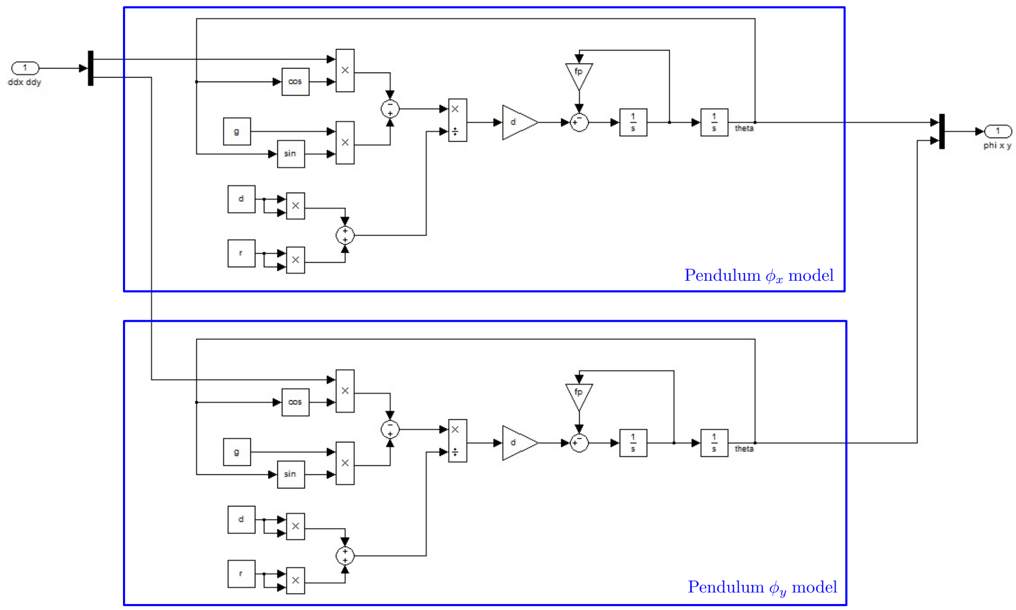

Abbildung 11: *Simulink Pendel Modell*

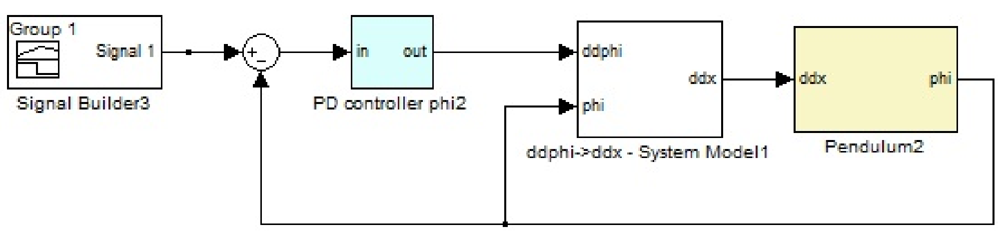

Abbildung 12: *Simulink Pendel ϕ_{xy} Regelung*

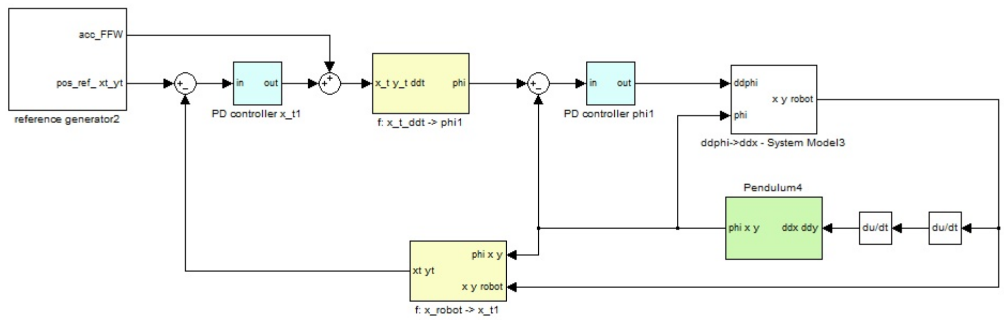

Abbildung 13: *Simulink Pendel xy_{tip} Regelung*

4.2 SCARA Roboter Modell und Gelenkeregelung

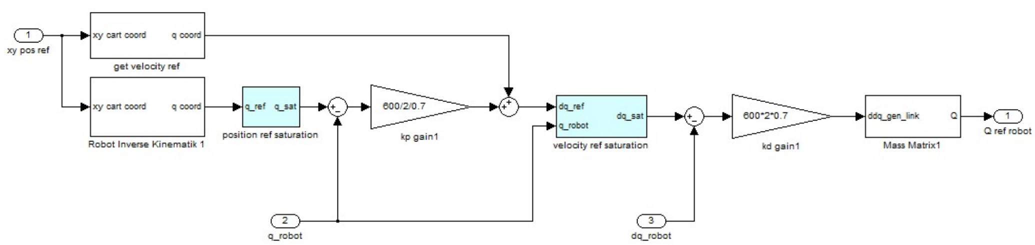

Abbildung 14: *Simulink SCARA Regelung Schema*

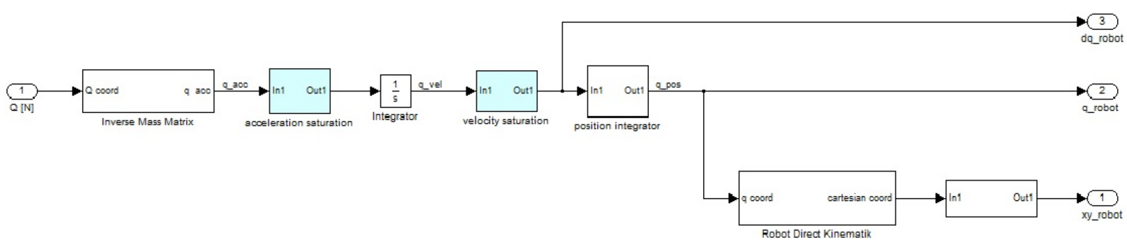

Abbildung 15: *Simulink SCARA Roboter Schema*

Der Roboter kriegt einen Positionssollwert, der dann abgeleitet wird. Das

```
// Velocity limits near the end of the work area

dq_max = sqrt(2*a_max*(q_max-q_ist));
dq_min = -sqrt(2*a_max*(q_ist-q_min));

if dq_in > dq_max
    dq_out = dq_max;
elseif dq_in < dq_min
    dq_out = dq_min;
else
    dq_out = dq_in;
end
```

ist nicht kritisch, da die Ableitung auf einem Sollwert und nicht auf einem gemessenen Wert gemacht wird.

In Roboter Regler wurden auch Positions und Geschwindigkeitslimitierungen implementiert.

Wenn der Roboter in die Limitierung fährt, führt das normalerweise im besten Fall zu einem Notaus und im schlechtesten Fall zu einem Crash.

In der Regelung muss auf allen Ebenen verhindert werden, dass dieser Fall eintrifft und unzulässige Sollwert generiert werden. Ein einfaches Überwachen der Sollposition genügt da nicht, da der Roboter ja mit sehr hohen Geschwindigkeiten diese Grenze überfahren kann.

Die sauberste Methode, unzulässige Sollwerte zu begrenzen ist die Limitierung der Geschwindigkeit in Richtung der Grenzen des Arbeitsraums. Diese Limitierung wird dirch den "velocity limitsÄlgorithmus implementiert.

Die Inverse und direkte Kinematik wurden Embedded Matlab Function implementiert.

```
// Scara inverse kinematik equations

q_coord(2) = acos((-l1^2-l2^2+x^2+y^2)/(2*l1*l2)); % links gefahren

if(q_coord(2)>=0)
    q_coord(1)=atan(y/x)-acos(((x^2+y^2)+l1^2-l2^2)/(2*l1*sqrt(x^2+y^2)));
else
    q_coord(1)=atan(y/x)+acos(((x^2+y^2)+l1^2-l2^2)/(2*l1*sqrt(x^2+y^2)));
end

q_coord(3) = -z;
q_coord(4) = q_coord(1)+q_coord(2)-alpha;
```

```
// Scara direct kinematik equations

phi1 = q(1);
phi2 = q(2);
d3 = q(3);
phi4 = q(4);

cart_coord(1) = cos(phi1)*l1 + cos(phi1+phi2)*l2 ;
cart_coord(2) = sin(phi1)*l1 + sin(phi1+phi2)*l2;
cart_coord(3) = -d3;

cart_coord(4) = phi1 + phi2 - phi4;
```

```
// from sensor ref system to cart coord ref system

phi_cx = phi_sx*cos(q1+q2) - phi_sy*sin(q1+q2);
phi_cy = phi_sx*sin(q1+q2) + phi_sy*cos(q1+q2);
```

4.3 Sensor und Koordinaten Transformation

Die vonm Sensor gemessene Werte beziehen sich auf den Sensor Referenz System. Um die Werte in Kartesischen Koordinaten zu kriegen, muss man die koordinaten Transformation implementieren. *phi_cx* und *phi_cy* sind die Winkeln in kartesischen Koordinaten. Sie sind abhängig von den ersten zwei Gelenkekoordinaten vom Roboter $q1$ und $q2$ und von den gemessenen Werte *phi_sx* und *phi_sy* in Sensor Koordinaten.

5 Simulink Simulation

Ein Beispiel wurde durchgeführt, um die Leistungen der Regelung zu evaluieren. Die Daten vom System befinden sich in den folgenden Tabellen.

5.1 Daten vom Pendel

Mass [kg]	0.2
Length [m]	0.1
Inertia [kgm^2]	0.0006
Friction coefficient [$Nsm^{-1}rad^{-1}$]	0.1

Tabelle 2: *Pendel Daten*

5.2 Daten vom Scara Roboter

Motor Constant	[Nm/A]	0.5
Gear Ratio	[-]	100
Ratio Output Voltage to Current	[V/A]	-4.7619
Ratio Output Voltage to $\dot{Q}$	[V/Nm]	-0.0265
Encoder Ticks	[lines]	4096
Max Voltage of V-Signal	[V]	10
Max Current	[A]	2.1
Max Speed	[rad/s]	3.14159
Controller f_0	[Hz]	50
Controller Damping	[-]	1
Inertia	[kgm^2]	1.6964e-4

Tabelle 3: *Scara Motor 1 Daten*

Motor Constant	[Nm/A]	0.47
Gear Ratio	[-]	100
Ratio Output Voltage to Current	[V/A]	-4.7619
Ratio Output Voltage to $\dot{Q}$	[V/Nm]	-0.0265
Encoder Ticks	[lines]	4096
Max Voltage of V-Signal	[V]	10
Max Current	[A]	1
Max Speed	[rad/s]	6.2832
Controller f_0	[Hz]	50
Controller Damping	[-]	1
Inertia	$[kgm^2]$	6.7475e-5

Tabelle 4: *Scara Motor 2 Daten*

Motor Constant	[Nm/A]	0.47
Gear Ratio	[-]	-1.5
Ratio Output Voltage to Current	[V/A]	4.7619
Ratio Output Voltage to $\dot{Q}$	[V/Nm]	-0.0265
Encoder Ticks	[lines]	4096
Max Voltage of V-Signal	[V]	10
Max Current	[A]	2.1
Max Speed	[rad/s]	314.159
Controller f_0	[Hz]	50
Controller Damping	[-]	1
Inertia	$[kgm^2]$	7e-5

Tabelle 5: *Scara Motor 3 Daten*

Motor Constant	[Nm/A]	0.31
Gear Ratio	[-]	-16.2022
Ratio Output Voltage to Current	[V/A]	4.7619
Ratio Output Voltage to $\dot{Q}$	[V/Nm]	-0.0265
Encoder Ticks	[lines]	4096
Max Voltage of V-Signal	[V]	10
Max Current	[A]	0.5
Max Speed	[rad/s]	31.4159
Controller f_0	[Hz]	50
Controller Damping	[-]	1
Inertia	$[kgm^2]$	3e-5

Tabelle 6: *Scara Motor 4 Daten*

5.3 Ergebnisse

In Bild 16 ist die Sollposition vom Pendels Endepunkt mit der Istposition vergliechen. Der Fehler zwischen den zwei Grössen ist klein und das System hat eine Überschleifung von 10%.

Der Winkel vom Pendel hat auch sehr kleine Werte und wird schnell auf Null stabilisiert. Die Gelenkekoordinaten und kartesischen Koordinaten vom Roboter sind stark geregelt, und der Fehler zwischen Soll- und Istwert sind fast Null.

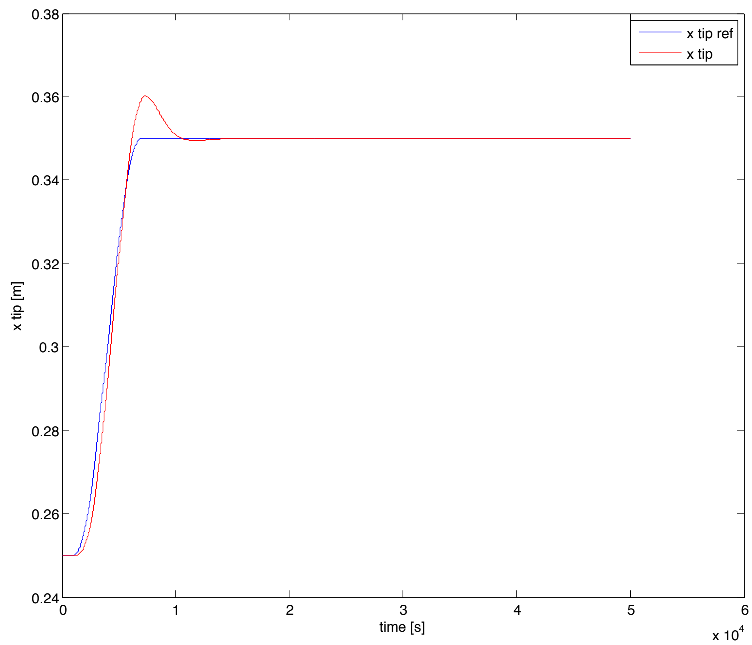

Abbildung 16: *x_t tracking*

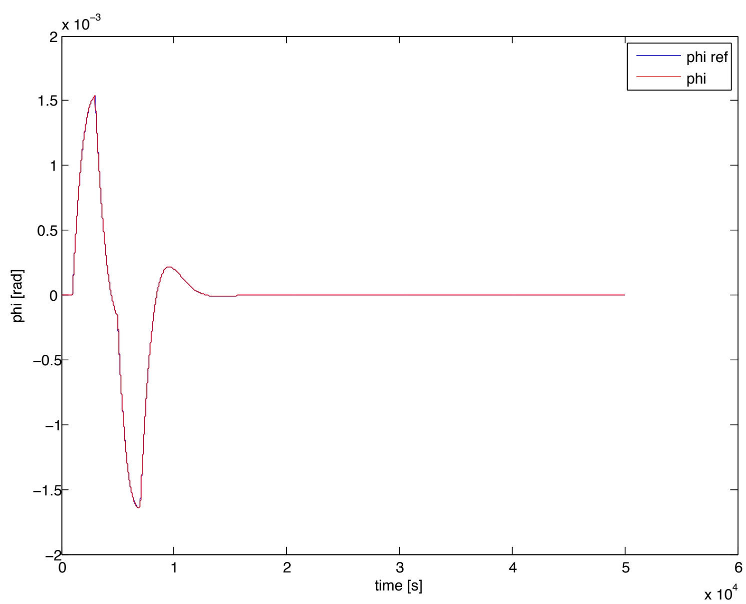

Abbildung 17: *ϕ tracking*

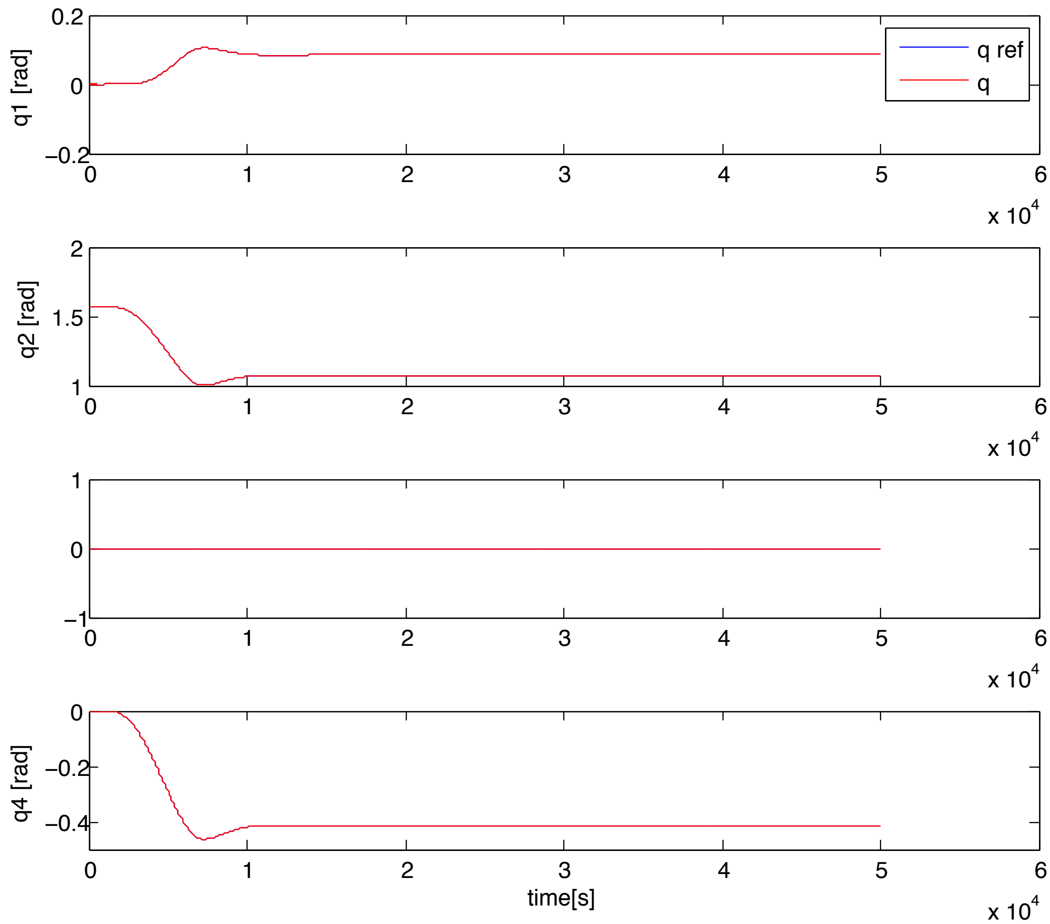

Abbildung 18: *Roboter q Koordinaten tracking*

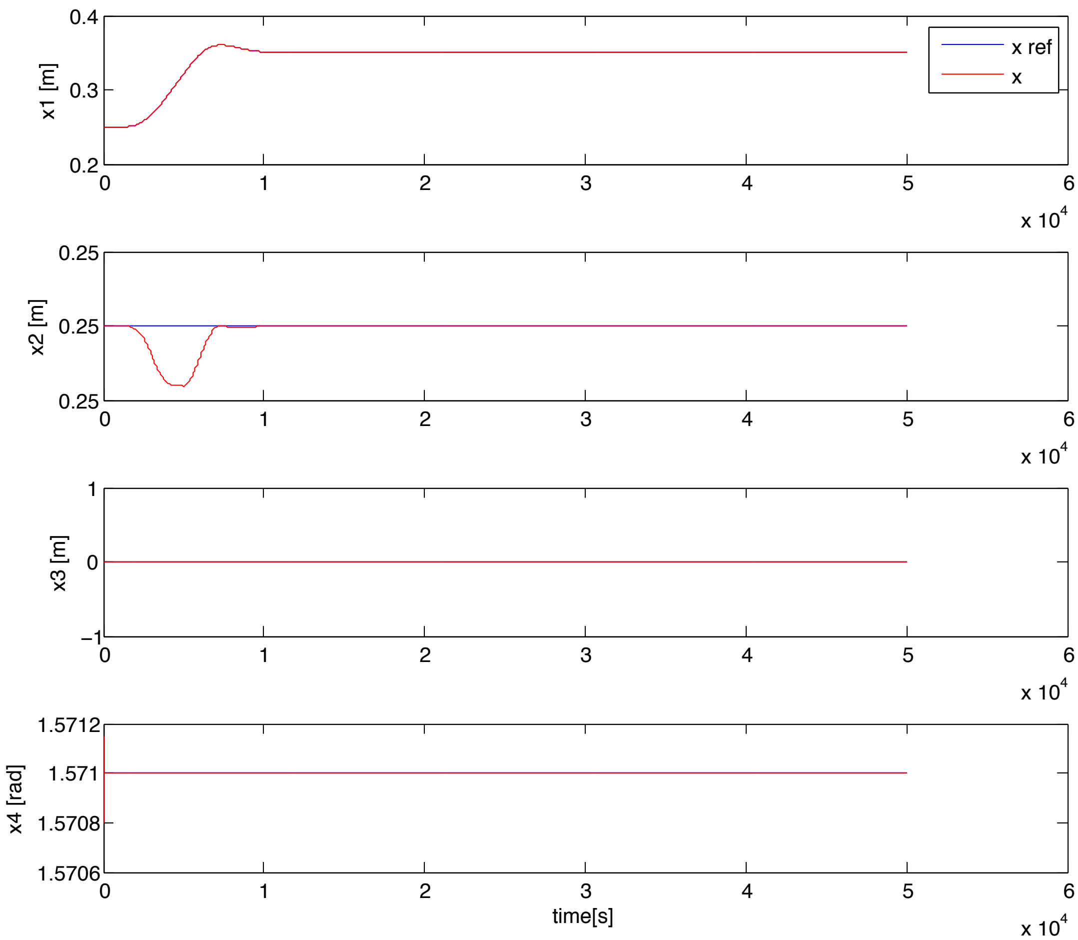

Abbildung 19: *Roboter x kartesischen Koordinaten tracking*