PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW

PIAP

Al. Jerozolimskie 202 02-486 Warszawa

Zespół Zrobotyzowanych Systemów Inteligentnych
ZSI

Wykonawca: mgr inż. Zbigniew Pilat

Nr zlecenia: S1656

Tytuł pracy: Analiza możliwości i opracowanie koncepcji realizacji sterowania CP robota 120 kg z układem sterowania Bosch.

Etap 1: Analiza kinematyki robota 120 kg w aspekcie sterowania CP i koncepcja realizacji tego sterowania z układem sterowania Bosch. Przygotowanie publikacji.

Zleceniodawca: praca statutowa PIAP

Pracę rozpoczęto dnia: 1996.02.01. zakończono dnia: 1996.12.31.

Kierownik Ośrodka
mgr inż. Zbigniew Pilat

Dyrektor Pionu
dr inż. Jan Jabłkowski

Praca zawiera:

stron --
rysunków --
fotografii --
tabel --
tablic --
załączników --

Rozdzielnik - ilość egz.:

Egz. 1 ZSI
Egz. 2 OIN
Egz. 3
Egz. 4
Egz. 5
Egz. 6

Nr rej. 7403

PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW

PIAP

Al. Jerozolimskie 202 02-486 Warszawa

Zespół Zrobotyzowanych Systemów Inteligentnych
ZSI

Wykonawca: mgr inż. Zbigniew Pilat

Nr zlecenia: S1656

Tytuł pracy: Analiza możliwości i opracowanie koncepcji realizacji sterowania CP robota 120 kg z układem sterowania Bosch.

Etap 1: Analiza kinematyki robota 120 kg w aspekcie sterowania CP i koncepcja realizacji tego sterowania z układem sterowania Bosch. Przygotowanie publikacji.

Zleceniodawca: praca statutowa PIAP

Pracę rozpoczęto dnia: 1996.02.01. zakończono dnia: 1996.12.31.

Kierownik Ośrodka
mgr inż. Zbigniew Pilat

Dyrektor Pionu
dr inż. Jan Jabłkowski

Praca zawiera:

stron --
rysunków --
fotografii --
tabel --
tablic --
załączników --

Rozdzielnik - ilość egz.:

Egz. 1 ZSI
Egz. 2 OIN
Egz. 3
Egz. 4
Egz. 5
Egz. 6

Nr rej. 7403

Analiza deskryptorowa:

ROBOTY PRZEMYSŁOWE, KINEMATYKA, STEROWANIE CP, OPROGRAMOWANIE

Analiza dokumentacyjna:

Sprawozdanie zawiera analizę kinematyki robota 120kg. Sformułowano i rozwiązano proste i odwrotne zadanie kinematyczne. Zaproponowano sposób opisu narzędzia robota oraz budowę generatora trajektorii liniowej. Przeanalizowano możliwości implementacji opracowanego modelu kinematycznego w układzie sterowania Bosch.

Tytuły poprzednich sprawozdań:

Niniejsze sprawozdanie jest pierwszym i jedynym dokumentem przedstawiającym przebieg realizacji pracy w zleceniu S1656.

SPIS TREŚCI

1. Wstęp

Robot o udźwigu 120/150kg został opracowany w PIAP z myślą o zastosowaniach w przemyśle motoryzacyjnym, przede wszystkim do automatyzacji zadań transportowych, szczególnie do obsługi pras, a także do zgrzewania. Ponieważ są to zadania wymagające często bardzo szybkich ruchów robota, zdecydowano opracować specjalną wersję robota, która będzie się charakteryzowała dużymi osiąganymi przyspieszeniami i opóźnieniami. Po analizie możliwych rozwiązań uznano, że najlepszym będzie opracowanie robota, w którym część manipulacyjna 120/150kg zostanie połączona z gotowym układem sterowania, oferowanym przez doświadczoną, sprawdzona firmę. Wybór padł na model sterownika rho3 firmy Bosch. Zakupiono jeden egzemplarz szafy sterowniczej wraz z kompletnymi torami napędowymi (sterowniki, silniki, mierniki położenia) oraz z oprogramowaniem systemowym realizującym podstawowe funkcje robota, w tym ruch we współrzędnych wewnętrznych oraz pozycjonowanie w trybie PTP. Firma zaoferowała również wykonanie oprogramowania realizującego sterowanie ruchem w trybie CP, jednak za dodatkowa, bardzo dużą opłatą. Równocześnie informacje otrzymane z firmy Bosch stwarzają realne szanse przygotowania własnymi siłami części oprogramowania podstawowego układu sterowania rho3 w zakresie rozwiązania prostego i odwrotnego zadania kinematyczne. Wtedy do realizacji sterowania CP można by było wykorzystać funkcje generatora trajektorii interpolowanych zaimplementowane standardowo w oprogramowaniu systemowym rho3. Praca wymaga w pierwszej kolejności opracowania modelu kinematyki manipulatora 120/150kg, a następnie wykonania i uruchomienia własnego oprogramowania symulacyjnego. Oprogramowanie to zostanie napisane najpierw w języku C (ponieważ w tym języku posiadamy już podobne funkcjonalnie oprogramowanie dla IRb/URP-6). Z uwagi na wymagania Boscha oprogramowanie to, po przetestowaniu, powinno być następnie przepisane na język Pascal i przesłane pocztą elektroniczną do Boscha. Tam, po usunięciu błędów formalnych wynikających ze specyfiki sterownika rho3 i systemu uruchomieniowego Boscha nasze oprogramowanie będzie mogło zostać zintegrowane z oprogramowaniem systemowym rho3. Zwrotnie powinniśmy otrzymać wzorzec nadstawki EPROM do jednostki centralnej z zaszytym oprogramowaniem realizującym sterowanie CP. W PIAP byłyby wtedy przeprowadzone próby funkcjonalne i wykonane ewentualne poprawki.

Niniejsze sprawozdanie obejmuje pierwsza część opisanych wyżej prac, a więc analizę kinematyki manipulatora 120/150kg wraz z opracowaniem prostego i odwrotnego zadania kinematycznego oraz koncepcji realizacji tą droga sterowania CP robota 120/150 z układem rho3 Boscha. Wykorzystując uzyskane wyniki zgłoszono publikacje na ten temat na konferencję MECHATRONIKA,97 - w

załączeniu. Warunkiem realizacji drugiej części jest dostęp do sprawnego zestawu robota 120kg ze sterownikiem Boscha. Zestaw taki jest przygotowywany w PIAP w ramach projektu celowego i po jego zakończeniu będzie mógł być wykorzystany do prób z oprogramowaniem CP. Wyniki prezentowanej pracy będą też mogły posłużyć do realizacji funkcji pozycjonowania CP w wersji URP-120 - z polskim układem sterowania..

2. Kinematyka manipulatora robota 120/150 kg.

Manipulator robota 120/150 kg jest sześcioosiowym system manipulacyjny o strukturze antropomorficznej. W celu rozwiązania kinematyki wprost i wstecz trzeba przede wszystkim opracować model kinematyczny manipulatora. Przy jego tworzeniu [11] przyjmuje się następujące założenia:

- system manipulacyjny zbudowany jest z siedmiu członów ponumerowanych od 0 do 6, gdzie człon 0 jest stałym elementem konstrukcyjnym, przytwierdzonym do podłoża (np. nieruchomą podstawą), natomiast człon 6 (ostatni) jest zakończony tarczką, do której montuje się narzędzie,
- człony połączone są przegubami (tą nazwą będą określane pary kinematyczne zarówno obrotowe jak i przesuwne), które są ponumerowane od 1 do 6, gdzie przegub **i** jest umieszczony między członami **i-1** a **i**,
- w opisie kinematyki zastosowano notację Denavita-Hartenberga,
- z każdym członem związano lokalny, kartezjański układ współrzędnych, przy czym układ związany z członem 0, tzw. bazowy układ współrzędnych, został wybrany arbitralnie, natomiast zasady przyporządkowania układów kolejnych są zgodne z ogólnie stosowanymi regułami,
- pozycję układu **i** w układzie **i-1** opisuje macierz przejścia $A_i[4\times4]$,
- pozycja wewnętrzna systemu manipulacyjnego jest opisana przez strukturę:

  $$INT\_POS = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6), \qquad (1)$$

  gdzie:

  $\theta_1,...,\theta_6$ - współrzędne wewnętrzne opisujące położenie poszczególnych przegubów,
- pozycja zewnętrzna systemu manipulacyjnego jest opisana przez strukturę:

  $$EXT\_POS = (x, y, z, nut, prec, rot) \qquad (2)$$

  gdzie:

  x, y, z - położenie środka układu współrzędnych związanego z ostatnim członem w bazowym układzie współrzędnych,

  $nut, prec, rot$ - trzy kąty Eulera opisujące orientację układu współrzędnych związanego z ostatnim członem względem bazowego układu współrzędnych.

  Pozycja ta może być zapisana w równoważnej postaci macierzowej:

$$T\_POS = \begin{array}{|c|c|c|c|} \hline n_x & o_x & a_x & p_x \\ \hline n_y & o_y & a_y & p_y \\ \hline n_z & o_z & a_z & p_z \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array} \tag{3}$$

Przy takich założeniach można zapisać proste zadanie kinematyczne DK (Direct Kinematics) jako:

$$DK: INT\_POS \longrightarrow EXT\_POS \tag{4}$$

czyli:

$$DK: (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \longrightarrow (x, y, z, nut, prec, rot) \tag{5}$$

i odwrotne zadanie kinematyczne IK (Inverse Kinematics) jako:

$$IK: EXT\_POS \longrightarrow INT\_POS \tag{6}$$

czyli:

$$IK: (x, y, z, nut, prec, rot) \longrightarrow (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \tag{7}$$

Przy wykorzystaniu zapisu macierzowego rozwiązaniem prostego zadania kinematycznego jest macierz pozycji manipulatora T, będąca iloczynem macierzy przejścia:

$$T = A_1 * A_2 * A_3 * A_4 * A_5 * A_6 \tag{8}$$

Postać macierzy przejścia, przy podanych wyżej założeniach jest następująca:

$$A_i = \begin{array}{|c|c|c|c|} \hline \cos(\theta_i) & -\sin(\theta_i)*\cos(\alpha_i) & \sin(\theta_i)*\sin(\alpha_i) & a*\cos(\theta_i) \\ \hline \sin(\theta_i) & \cos(\theta_i)*\cos(\alpha_i) & -\cos(\theta_i)*\sin(\alpha_i) & a*\sin(\theta_i) \\ \hline 0.0 & \sin(\alpha_i) & \cos(\alpha_i) & d \\ \hline 0.0 & 0.0 & 0.0 & 1.0 \\ \hline \end{array} \tag{9}$$

Schemat kinematyczny systemu manipulacyjnego zastosowanego w robocie 120/150 kg przedstawiono na rys. 1. System ten posiada sześć stopni swobody - wszystkie pary kinematyczne są obrotowe. Bazę stanowi nieruchoma podstawa kotwiczona do podłogi hali produkcyjnej. Łańcuch kinematyczny kończy się ostatnim członem robota - jest nim interfejs mechaniczny do montażu narzędzia. Parametry Denavita-Hartenberga dla takiej struktury kinematycznej przedstawia tabela 1.

Tabela 6.1. Parametry D-H dla systemu manipulacyjnego robota 120/150 kg.

i	*zmienna*	θ_i/θ_{i0}	d_i/d_{i0}	a_i	α_i	$\cos\alpha_i$	$\sin\alpha_i$
1	θ_1	$\theta_1/+\frac{\pi}{2}$	0.0	L_3	$+\frac{\pi}{2}$	0.0	+1.0
2	θ_2	$\theta_2/+\frac{\pi}{2}$	0.0	L_2	0.0	+1.0	0.0
3	θ_3	$\theta_3/0.0$	0.0	L_4	$+\frac{\pi}{2}$	0.0	+1.0
4	θ_4	$\theta_4/+\frac{\pi}{2}$	L_5	L_6	$+\frac{\pi}{2}$	0.0	+1.0
5	θ_5	$\theta_5/0.0$	0.0	$-L_6$	$-\frac{\pi}{2}$	0.0	+1.0
6	θ_6	$\theta_6/0.0$	0.0	0.0	0.0	+1.0	0.0

Na podstawie tabeli tworzymy macierze przejścia $A_1, A_2, A_3, A_4, A_5, A_6$:

$$A_1 = \begin{bmatrix} -s_1 & 0.0 & c_1 & -L_3 s_1 \\ c_1 & 0.0 & s_1 & L_3 c_1 \\ 0.0 & 1.0 & 0.0 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{10}$$

$$A_2 = \begin{bmatrix} -s_2 & -c_2 & 0.0 & -L_2 s_2 \\ c_2 & -s_2 & 0.0 & L_2 s_2 \\ 0.0 & 0.0 & 1.0 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{11}$$

$$A_3 = \begin{bmatrix} -c_3 & 0.0 & s_3 & L_4 c_3 \\ s_3 & 0.0 & c_3 & L_4 s_3 \\ 0.0 & 1.0 & 0.0 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{12}$$

$$A_4 = \begin{bmatrix} -s_4 & 0.0 & c_4 & -L_6 s_4 \\ c_4 & 0.0 & s_4 & L_6 c_4 \\ 0.0 & 1.0 & 0.0 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{13}$$

$$A_5 = \begin{array}{|c|c|c|c|} \hline c_5 & 0.0 & -s_5 & -L_6 c_5 \\ \hline s_5 & 0.0 & c_5 & -L_6 s_5 \\ \hline 0.0 & -1.0 & 0.0 & 0 \\ \hline 0.0 & 0.0 & 0.0 & 1.0 \\ \hline \end{array} \tag{14}$$

$$A_6 = \begin{array}{|c|c|c|c|} \hline c_6 & -s_6 & 0.0 & 0.0 \\ \hline s_6 & c_6 & 0.0 & 0.0 \\ \hline 0.0 & 0.0 & 1.0 & 0.0 \\ \hline 0.0 & 0.0 & 0.0 & 1.0 \\ \hline \end{array} \tag{15}$$

gdzie: $\sin(\theta_i) = s_i$, $\cos(\theta_i) = c_i$

Po wymnożeniu macierzy przejść otrzymujemy macierz T o postaci wg wzoru (3), gdzie:

$$n_x = [(-s_1 s_3 s_4 + c_1 c_4)c_5 - s_1 c_3 s_5]c_6 + (-s_1 s_3 c_4 - c_1 s_4)s_6 \tag{16}$$

$$n_y = [(c_1 s_3 s_4 + s_1 c_4)c_5 - c_1 c_3 s_5]c_6 + (c_1 s_3 c_4 - s_1 s_4)s_6 \tag{17}$$

$$n_z = (-c_3 s_4 c_5 + s_3 s_5)c_6 - c_3 c_4 s_6 \tag{18}$$

$$o_x = -[(-s_1 s_3 s_4 + c_1 c_4)c_5 - s_1 c_3 s_5]s_6 + (-s_1 s_3 c_4 - c_1 s_4)c_6 \tag{19}$$

$$o_y = -[(c_1 s_3 s_4 + s_1 c_4)c_5 + c_1 c_3 s_5]s_6 + (c_1 s_3 c_4 - s_1 s_4)c_6 \tag{20}$$

$$o_z = -(-c_3 s_4 c_5 + s_3 s_5)s_6 - c_3 c_4 c_6 \tag{21}$$

$$a_x = -(-s_1 s_3 s_4 + c_1 c_4)s_5 - s_1 c_3 c_5 \tag{22}$$

$$a_y = -(c_1 s_3 s_4 + s_1 c_4)s_5 + c_1 c_3 c_5 \tag{23}$$

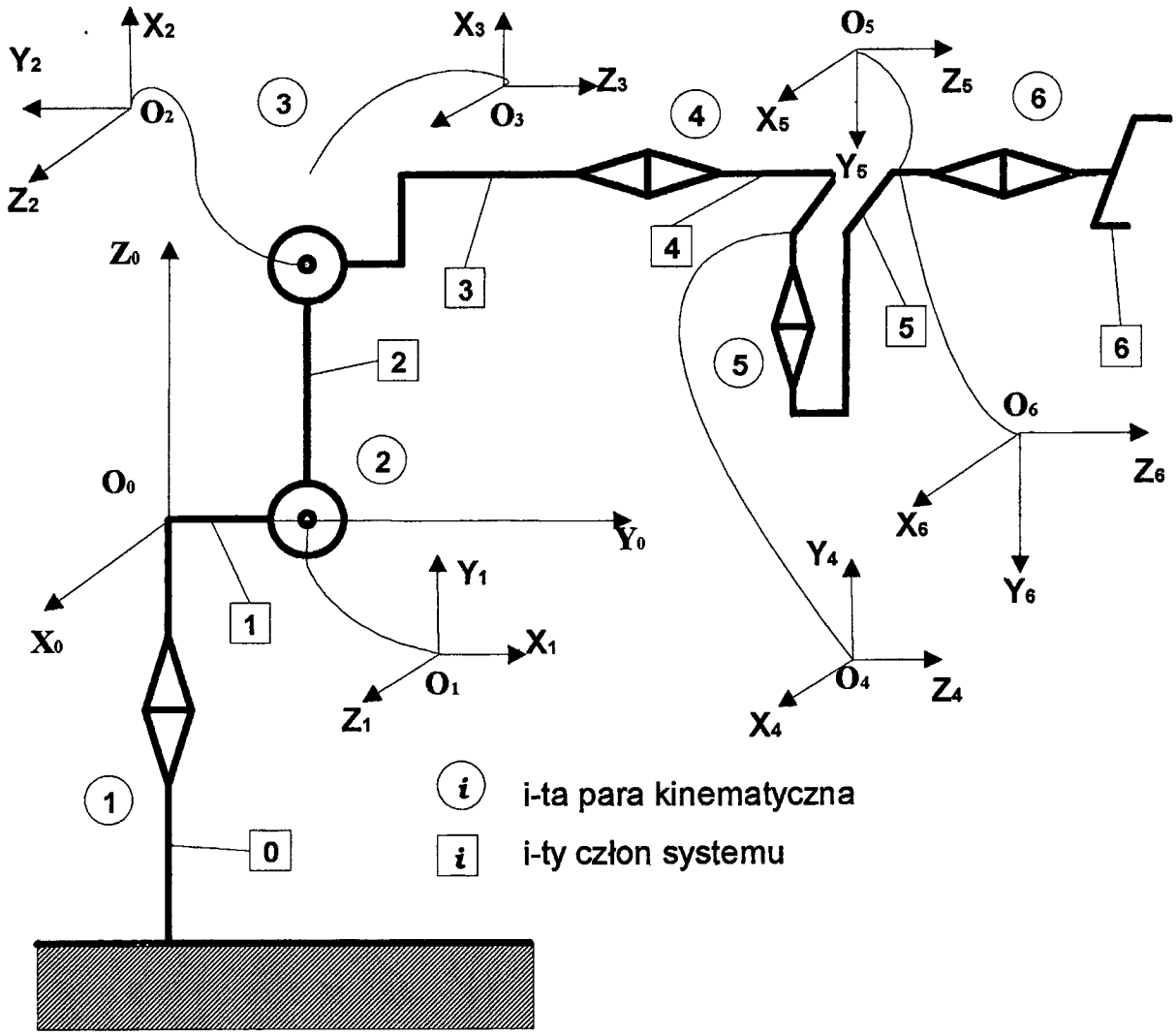
$$a_z = c_3 s_4 s_5 + s_3 c_5 \tag{24}$$

$$p_x = -(-s_1 s_3 s_4 + c_1 c_4)c_5 L_6 + s_1 c_3 s_5 L_6 - s_1 c_3 s_5 L_6 + c_1 c_4 L_6 - s_1 c_3 L_5 + s_1(-L_3 + s_3 L_4 + s_2 L_2) \tag{25}$$

$$p_y = -(c_1 s_3 s_4 + s_1 c_4)c_5 L_6 - c_1 c_3 s_5 L_6 + c_1 s_3 s_4 L_6 + s_1 c_4 L_6 + c_1 c_3 L_5 + c_1(L_3 - s_3 L_4 - s_2 L_2) \tag{26}$$

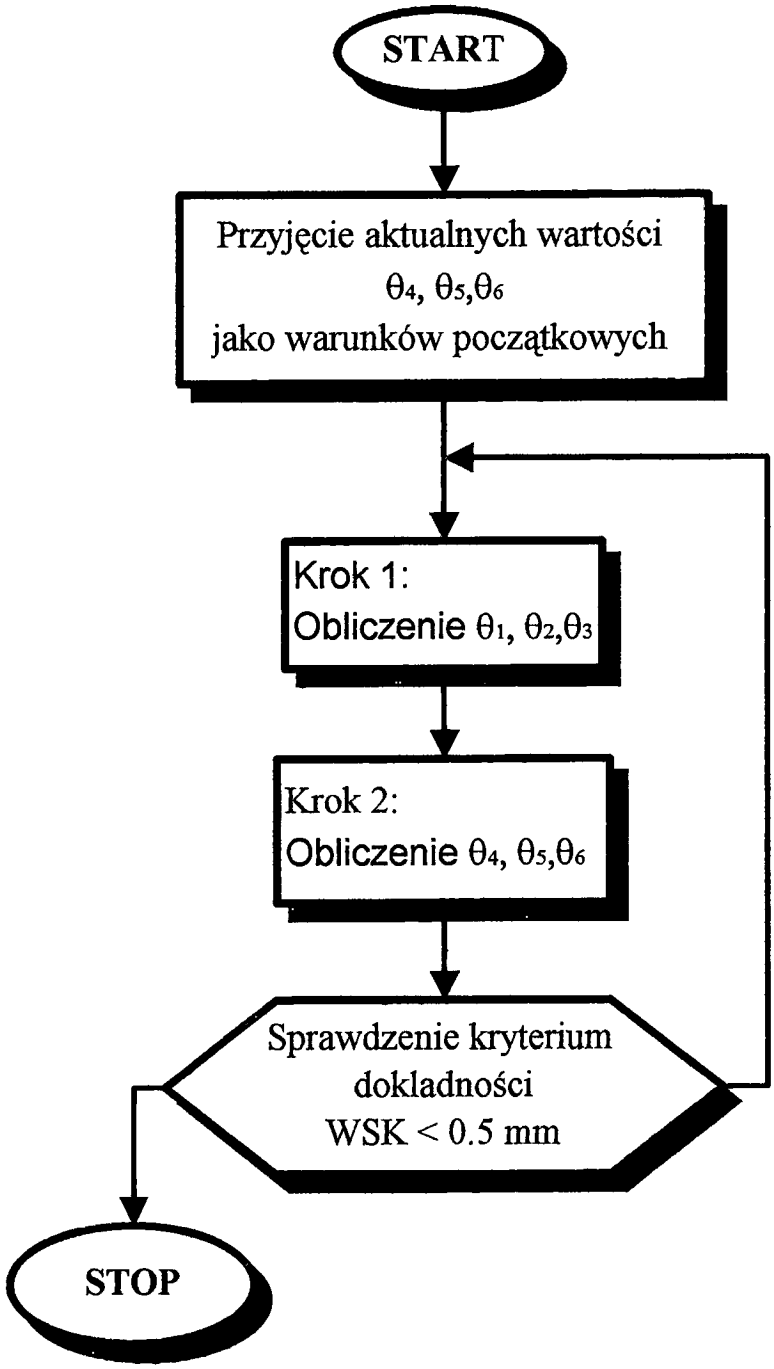
$$p_z = c_3 s_4 c_5 L_6 - s_3 s_5 L_6 - c_3 s_4 L_6 + s_3 L_5 + c_3 L_4 + c_2 L_2 \tag{27}$$

Równania (16) do (27) stanowią rozwiązanie prostego zadania kinematycznego dla systemu manipulacyjnego robota 120/150 kg.



Rys. 1. Schemat kinematyczny systemu manipulacyjnego robota 120/150 kg.

Rozwiązanie odwrotnego zadania kinematycznego polega, jak już wcześniej wspomniano, na obliczeniu współrzędnych wewnętrznych realizujących zadana pozycje zewnętrzna. Inaczej mówiąc należy rozwiązać układ równań (16-27) dla niewiadomych $\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6$, przy znajomości elementów macierzy T. Nie udało się uzyskać analitycznego rozwiązania tego układu. Patrząc od strony konstrukcji manipulatora, powodem było mimośrodowe ustawienie piątej osi. Spowodowało to powstanie w tym rejonie obszaru osobliwego. Po dokładnej analizie problemu zdecydowano spróbować wykorzystać iteracyjne metody optymalizacyjne. Proponowany algorytm, przedstawiony na rys. 3, zawiera dwa podstawowe kroki. W pierwszym, przy znanych katach $\theta_4,\theta_5,\theta_6$, oblicza się kąty $\theta_1,\theta_2,\theta_3$ na podstawie pozycji kołnierza robota. W drugim kroku, realizowanym przy obliczonych uprzednio $\theta_1,\theta_2,\theta_3$, wyznaczane są kąty: $\theta_4,\theta_5,\theta_6$. jest to więc klasyczny algorytm dwupoziomowy.



Rys. 3. Algorytm iteracyjnej metody rozwiązywania odwrotnego zadania kinematycznego dla manipulatora robota 120/150 kg.

Jako wskaźnik jakości proponuje się przyjęcie odległości liniowej zadanego TCP od punktu obliczonego w kolejnym kroku iteracji:

$$WSK = |P_0 - P| \qquad (28)$$

gdzie P jest wektorem położenia TCP zadanej macierzy T, zaś

$$P_0 = \begin{array}{|c|}\hline p_{x0} \\ \hline p_{y0} \\ \hline p_{z0} \\ \hline \end{array} \tag{29}$$

jest wektorem położenia TCP macierzy T_0 uzyskanej z obliczonych kątów $\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6$ (w/g wzorów 16-27). proponuje się na obecnym etapie przyjąć:

$$WSK < 0{,}5mm \tag{30}$$

Powinna to być wystarczająca dokładność, przy zadanej powtarzalności pozycjonowania robota ±1 mm. Wartość ta powinna zostać zweryfikowana podczas prób.

Metoda iteracyjna

Krok 1.

W Kroku 1 zakładamy, że kąty $\theta_4,\theta_5,\theta_6$, są znane, trzeba natomiast obliczyć kąty $\theta_1,\theta_2,\theta_3$. Zauważmy, że jeśli znamy kąty $\theta_4,\theta_5,\theta_6$, to znamy konfigurację nadgarstka, a więc możemy jednoznacznie wyznaczyć położenie końca ramienia górnego robota - por. rys. 1. Na podstawie jego współrzędnych x_0,y_0 w naturalny sposób można wyznaczyć kąt θ_1. Następnie na podstawie wysokości końca ramienia górnego oraz jego odległości od osi robota można wyznaczyć kąty pochylenia ramienia dolnego - θ_2, i górnego - θ_3. Dokładna procedura obliczenia tych katów została opisana w [12].

Krok 2.

W Kroku 2 zakładamy, że znane są kąty $\theta_1,\theta_2,\theta_3$, trzeba natomiast obliczyć kąty $\theta_4,\theta_5,\theta_6$. Zauważmy, że jeśli znamy kąty $\theta_4,\theta_5,\theta_6$, to znamy konfigurację nadgarstka, a więc możemy jednoznacznie wyznaczyć położenie końca ramienia górnego robota - por. rys. 1. Na podstawie jego współrzędnych x_0,y_0 w naturalny sposób można wyznaczyć kąt θ_1. Następnie na podstawie wysokości końca ramienia górnego oraz jego odległości od osi robota można wyznaczyć kąty pochylenia ramienia dolnego - θ_2, i górnego - θ_3. Dokładna procedura obliczenia tych kątów została opisana w [12].

3. Opis narzędzia robota 120/150 kg.

Przedstawione w poprzednim rozdziale proste zadanie kinematyczne pozwala obliczyć na podstawie współrzędnych wewnętrznych pozycje układu współrzędnych związanego z kołnierzem szóstej osi. Robot na stanowisku pracy operuje jednak nie swoim kołnierzem lecz konkretnym narzędziem. Aby obliczyć położenie narzędzia robota, trzeba jeszcze uwzględnić jego opis, wyznaczający pozycje narzędzia względem kołnierza szóstej osi. W tym celu wiążemy z narzędziem jego własny układ współrzędnych $O_T X_T Y_T Z_T$. Jeżeli przyjmiemy, że położenie narzędzia względem ostatniego członu robota, a więc położenie środka i orientację układu narzędzia względem układu ostatniego członu systemu manipulacyjnego, określa macierz T_TOOL, to pozycję narzędzia w bazowym układzie współrzędnych wyrazimy wzorem:

$$T\_POZ = T * T\_TOOL \tag{31}$$

Rozwiązując odwrotne zadanie kinematyczne trzeba najpierw obliczyć pozycję kołnierza robota:

$$T = T\_POZ * T\_TOOL^{-1} \tag{32}$$

a następnie na podstawie macierzy T wyliczyć poszczególne współrzędne wewnętrzne.

W praktycznej realizacji sterowania robotem trudno wymagać od użytkownika aby podawał jako dane wejściowe parametry macierzy *T_TOOL*. Potrzebny jest mechanizm umożliwiający wprowadzanie parametrów narzędzia w formie bardziej zrozumiałej i zapewniający proste przeliczenie takich danych do postaci macierzy transformacji. Analizując różne proponowane w literaturze sposoby opisu narzędzia robota, zdecydowano się na podawanie sześciu parametrów. Trzy pierwsze określają kąty Eulera pomiędzy układem $O_6 X_6 Y_6 Z_6$ a $O_T X_T Y_T Z_T$, trzy następne natomiast położenie środka O_T w układzie $O_6 X_6 Y_6 Z_6$:

- nut_t (nutacja) - obrót $O_6 X_6 Y_6 Z_6$ wokół $O_6 Z_6$ do układu $O_6' X_6' Y_6' Z_6'$,
- prec_t (precesja) - obrót $O_6' X_6' Y_6' Z_6'$ wokół $O_6' X_6'$ do układu $O_6'' X_6'' Y_6'' Z_6''$
- obr_t (obrót) - obrót $O_6'' X_6'' Y_6'' Z_6''$ wokół $O_6'' Z_6''$ do układu $O_6''' X_6''' Y_6''' Z_6'''$,
- x_t (x) - współrzędna x punktu O_T w układzie $O_6 X_6 Y_6 Z_6$,
- y_t (y) - współrzędna y punktu O_T w układzie $O_6 X_6 Y_6 Z_6$,
- z_t (z) - współrzędna z punktu O_T w układzie $O_6 X_6 Y_6 Z_6$,

przy czym mamy
$O_6 = O_6' = O_6'' = O_6'''$

oraz osie układu współrzędnych $O_6''X_6''Y_6''Z_6'''$ są równoległe do osi układu $O_TX_TY_TZ_T$. Wykorzystując podane wyżej oznaczenia można zapisać równania określające poszczególne elementy macierzy *T_TOOL* [11]:

$$t\_tool_{11} = \cos(nut\_t)*\cos(obr\_t) - \cos(prec\_t)*\sin(nut\_t)*\sin(obr\_t) \quad (33)$$

$$t\_tool_{12} = \sin(nut\_t)*\cos(obr\_t) + \cos(prec\_t)*\cos(nut\_t)*\sin(obr\_t) \quad (34)$$

$$t\_tool_{13} = \sin(prec\_t)*\sin(obr\_t) \quad (35)$$

$$t\_tool_{14} = x\_t \quad (36)$$

$$t\_tool_{21} = \cos(nut\_t)*\sin(obr\_t) - \sin(nut\_t)*\cos(prec\_t)*\cos(obr\_t) \quad (37)$$

$$t\_tool_{22} = -\sin(nut\_t)*\sin(obr\_t) + \cos(nut\_t)*\cos(prec\_t)*\cos(obr\_t) \quad (38)$$

$$t\_tool_{23} = \sin(prec\_t)*\cos(obr\_t) \quad (39)$$

$$t\_tool_{24} = y\_t \quad (40)$$

$$t\_tool_{31} = \sin(nut\_t)*\sin(prec\_t) \quad (41)$$

$$t\_tool_{32} = -\cos(nut\_t)*\sin(prec\_t) \quad (42)$$

$$t\_tool_{34} = \cos(prec\_t) \quad (43)$$

$$t\_tool_{41} = t\_tool_{42} = t\_tool_{43} = 0. \quad (44)$$

$$t\_tool_{44} = 1. \quad (45)$$

Trzeba zauważyć, że przyjmując pełny opis narzędzia, np. trzy współrzędne określające położenie TCP i trzy kąty Eulera określające orientację, pozwalające wyznaczyć kompletną macierz T_TOOL, można w sposób jednoznaczny obliczyć macierz T na podstawie znajomości aktualnej pozycji, opisanej jako T_POZ. Jest to niewątpliwie zaleta takiego podejścia.

4. Koncepcja realizacji generatora trajektorii liniowej dla robota 120/150 kg.

W docelowym oprogramowaniu posadowionym w sterowniku rho3 będzie wykorzystany najprawdopodobniej generator trajektorii interpolowanych (liniowa, kołowa) stosowany obecnie w innych aplikacjach firmy Bosch. Dla celów testów, głównie badań symulacyjnych modelu kinematyki, trzeba jednak taki generator przygotować również we własnym zakresie. Posłuży on nie tylko do weryfikacji modelu lecz także do porównania i oceny wyników prób na rzeczywistym robocie podczas uruchamiania kompletnego oprogramowania realizującego sterowanie CP. Generator taki będzie mógł być także wykorzystany przy pracach nad programem sterującym robota 120/150 kg połączonego z układem sterowania URP.

Ponieważ podstawowym problemem w opisie ruchu narzędzia robota we współrzędnych kartezjańskich (zewnętrznych) jest przejście z punktu do punktu po linii prostej, w niniejszym opracowaniu zajmiemy się tylko tym rodzajem interpolacji. W dalszych rozważaniach będzie wykorzystane podejście zaprezentowane w [11].

Postawienie problemu.

Dane są dwie pozycje narzędzia robota *P1* i *P2*:

$$P1 = T\_POS1 = T1 * T\_TOOL \tag{46}$$

$$P2 = T\_POS2 = T2 * T\_TOOL \tag{47}$$

Należy opisać taki ruch narzędzia z punktu P1 do P2 taki, że TCP narzędzia porusza się po linii prostej, a orientacja osi narzędzia zmienia się w sposób jednostajny.

Koncepcja rozwiązania.

Zakłada się, że pozycja narzędzia na trajektorii jest opisana zależnością:

$$T\_POS = T\_POS1 * DF(r) \tag{48}$$

przy czym:

T_POS1 - pozycja aktualna (startowa),

T_POS2 - pozycja docelowa,

$DF(r)$ - funkcja napędowa (Drive Function), jej postać zależy od typu interpolowanej trajektorii. Parametr r określa moment czasowy ruchu:

$r = t/t_k$, gdzie:

t - czas jaki upłynął od rozpoczęcia ruchu z P1 do P2,

t_k - całkowity czas ruchu z P1 do P2 wynikający z zadanej prędkości ruchu w programie robota.

Wtedy mamy oczywiście, że

r=0 - początek ruchu, a więc

$$DF(0) = I \tag{49}$$

r=1 - koniec ruchu, a więc

$$T\_POS1 * DF(1) = T\_POS2 \tag{50}$$

Jako przykład można podać dwie najpopularniejsze funkcje napędowe:

- DF_LIN - realizuje interpolację liniową - TCP porusza się po linii prostej w przestrzeni, a orientacja narzędzia zmienia się w sposób jednostajny od wyjściowej (w T_POS1) do końcowej (w T_POS2),
- DF_CIRC - realizuje interpolację kołową - TCP porusza się po okręgu w przestrzeni, a orientacja narzędzia względem toru ruchu jest stała.

Rozwiązanie.

Załóżmy, że pośrednie wartości DF(r) reprezentują translację i dwie rotacje układu współrzędnych. Wszystkie ruchy będą wtedy proporcjonalne do r, a więc jeśli r zmienia się liniowo w czasie, to DF(r) opisuje translację ze stałą prędkością liniową i dwie rotacje ze stałymi prędkościami kątowymi. Translacja odbywa się wzdłuż odcinka P1P2 - oznaczymy ją Tr(r). pierwsza rotacja ustawia odpowiednio wersor **a** (approach) układu współrzędnych narzędzia - oznaczymy ją Ra(r). druga rotacja odpowiada za właściwe ustawienie wektora **o** narzędzia - oznaczymy ją Ro(r). Stąd możemy zapisać:

$$DF(r) = Tr(r) * Ra(r) * Ro(r) \tag{51}$$

Rotacja Ra(r) odbywa się o kąt Θ wokół wektora **k**, otrzymanego przez obrót osi **y** w P1 o kąt ψ wokół osi **z** w P1. Rotacja Ro(r) odbywa się wokół wersora **a** układu współrzędnych narzędzia o kąt φ. Stąd [11, 12] postać macierzy DF (oczywiście cały czas mówimy o interpolacji liniowej) jest następująca:

$$dd_{12}(r) = -\sin(r\varphi) * [\sin^2(\varphi) * V(r\theta) + \cos(r\theta)] + \cos(r\varphi) * [-\sin(\varphi) * \cos(\varphi) V(r\theta)] \tag{52}$$

$$dd_{13}(r) = \cos(\psi) * \sin(r\theta) \tag{53}$$

$$dd_{14}(r) = rx \tag{54}$$

$$dd_{22}(r) = -\sin(r\varphi) * [-\sin(\varphi) * \cos(\psi) * V(r\theta)] + \cos(r\varphi) * [\cos^2(\varphi) V(r\theta) + \cos(r\theta)] \tag{55}$$

$$dd_{23}(r) = \sin(\psi) * \sin(r\theta) \tag{56}$$

$$dd_{24}(r) = ry \tag{57}$$

$$dd_{32}(r) = -\sin(r\varphi) * [-\cos(\psi) * \sin(r\theta)] + \cos(r\varphi) * [-\sin(\psi) * \sin(r\theta)] \tag{58}$$

$$df_{33}(r) = \cos(r\theta) \tag{59}$$

$$df_{34}(r) = rz, \tag{60}$$

gdzie:

$$V(r\theta)= vers(r\theta)=(1-\cos(r\theta)) \tag{61}$$

Pierwszą kolumnę macierzy DF obliczamy jako iloczyn wektorowy kolumny drugiej i trzeciej.

Mnożąc równanie (50) obustronnie przez T_POS1^{-1} otrzymujemy:

$$DF(1)= T\_POS1^{-1} T\_POS2 \tag{62}$$

Po wymnożeniu i porównaniu z wzorami (52-60) otrzymujemy wyrażenia określające parametry generatora trajektorii liniowej:

$$x={}^{P1}\mathbf{n}*({}^{P2}\mathbf{p}-{}^{P1}\mathbf{p}) \tag{63}$$

$$y={}^{P1}\mathbf{o}*({}^{P2}\mathbf{p}-{}^{P1}\mathbf{p}) \tag{64}$$

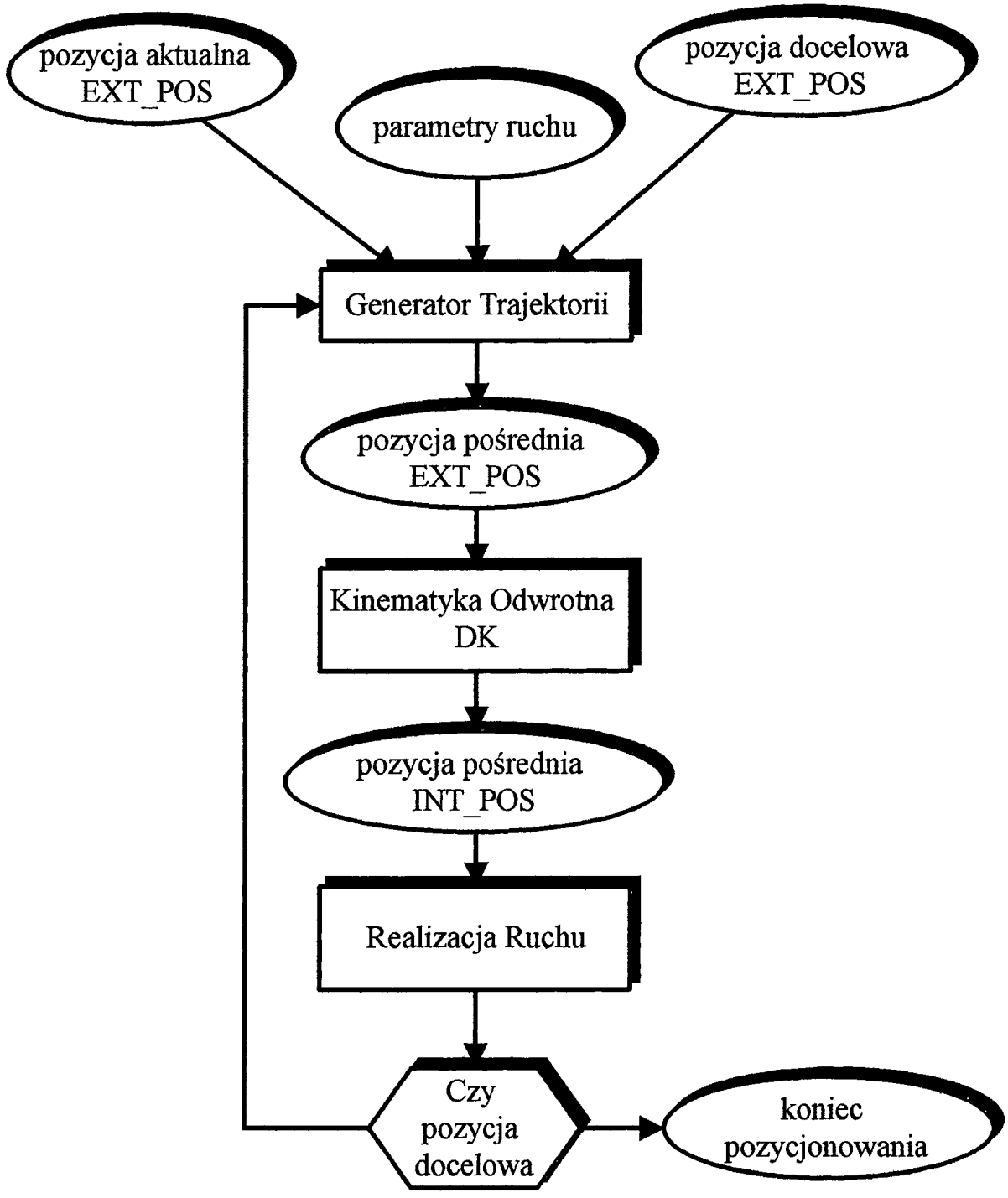
$$z={}^{P1}\mathbf{a}*({}^{P2}\mathbf{p}-{}^{P1}\mathbf{p}) \tag{65}$$

$$tg(\psi)=({}^{P1}\mathbf{o}*{}^{P2}\mathbf{a})/({}^{P1}\mathbf{n}*{}^{P2}\mathbf{a}) \tag{66}$$

$$tg(\theta)=(({}^{P1}\mathbf{n}*{}^{P2}\mathbf{a})^2+({}^{P1}\mathbf{o}*{}^{P2}\mathbf{a})^2)^{1/2}/({}^{P1}\mathbf{a}*{}^{P2}\mathbf{a}) \tag{67}$$

$$\sin(\varphi)= -\sin(\psi)*\cos(\theta)*V(\theta)({}^{P1}\mathbf{n}*{}^{P2}\mathbf{n})+(\cos^2(\psi)*V(\theta)+\cos(\theta)*({}^{P1}\mathbf{o}*{}^{P2}\mathbf{n})+ (-\sin(\psi)\sin(\theta)({}^{P1}\mathbf{a}*{}^{P2}\mathbf{n})) \tag{68}$$

Równania (52-60) wraz z (63-68) stanowią komplet zależności opisujących generator trajektorii liniowej dla robota 120/150 kg. Można na ich podstawie opracować oprogramowanie symulacyjne sprawdzające procedury zaimplementowane w sterowniku robota. Można również wykorzystać je przy uruchamianiu funkcji pozycjonowania liniowego dla robota 120/150 kg pracującego z układem sterowania URP. Ogólny algorytm tego pozycjonowania przedstawia rys. 2.



Rys. 2. Algorytm realizacji pozycjonowania z interpolacją trajektorii w przestrzeni kartezjańskiej.

5. Podsumowanie - możliwości realizacji sterowania CP w robocie 120/150 kg pracującym z układem sterowania Bosch.

Przedstawione rozważania pozwalają wykonać i uruchomić oprogramowanie symulacyjne kinematyki robota 120/150 kg zarówno w zakresie prostego i odwrotnego zadania kinematycznego jak i w zakresie generatora trajektorii prostoliniowej. Oprogramowanie to powinno być pisane z uwzględnieniem zaleceń programistów firmy Bosch (zał. 2). Z uwagi na wcześniejsze prace nad symulacją kinematyki robotów, wykorzystujące język C w wersji Borland, wydaje się wskazane pozostać przy tych narzędziach. Po przetestowaniu oprogramowania, o ile strona niemiecka w dalszym ciągu będzie żądać procedur napisanych w Pascalu, trzeba będzie nasze oprogramowanie przepisać i zweryfikować (porównanie wyników wersji C i Pascal). Gotowe procedury można będzie wysłać do firmy Bosch celem przygotowania wzorców pamięci EPROM zawierających program sterujący robota w wersji CP. Oczywiście prace te mają sens tylko wtedy, gdy będziemy dysponowali sprawnym zestawem robota 120/150 kg z układem sterowania rho3.

Drugim kierunkiem wykorzystania wyników tej pracy może być opracowanie programu sterującego robota 120/150 kg z funkcjami CP, z wykorzystaniem układu sterowania URP. W tym celu należy dodatkowo opracować przeliczanie współrzędnych wewnętrznych robota z miary kątowej na inkrementy. Biorąc pod uwagę, że w osiach drugiej i trzeciej robota mamy przekładnie nieliniowe, nie jest to zadanie trywialne. Podczas prac implementacyjnych należy pamiętać o rozwiązaniu problemów omijania osobliwości (kąt piąty równy zero). Niezbędne będą też narzędzia do kalibracji robota. Chodzi o narzędzia software'owe, być może z wykorzystaniem istniejącego w PIAP sprzętu do bezdotykowego pomiaru pozycji robota. Trzeba pamiętać, że elementy manipulatora są wykonywane z określoną dokładnością. Dlatego też jakość ruchu, szczególnie funkcji realizowanych w trybie CP, gdy zmienia się orientacja narzędzia, bardzo zależy od dokładności danych wymiarowych wprowadzonych do modelu robota. Ewentualne błędy mogą być identyfikowane i korygowane w procesie kalibracji on-line.

LITERATURA

[1] Baillieul J. "Kinematic Programming Alternatives for Redundant Manipulators." In Proc. of the IEEE Int. Conf. on Robotics and Automation, St. Louis USA, 1985.

[2] Baker D.R., Wampler II Ch. "On the Inverse Kinematics of Redundant Manipulators." The International Journal of Robotics Research, vol. 7, no. 2 March/Apr. 1988.

[3] Brady M., Robot Motion: Planning and Control. MIT Press, 1984r.

[4] Craig J. J., Introduction to Robotics. Mechanics and Control. Addison-Wesley Publishing Company, 1986r.

[5] International Symposium on Industrial Robots - materiały konferencyjne.

[6] Jacórzyńska M., Lichodziejewski C., Pilat Z. "Inverse Kinematics Methods for Redundant Manipulators and their Simulation and Verification." In Proc. of the ROBTEP'95 Conference, Presov Slovakia, Sept. 1995.

[7] Jacórzyńska M., Lichodziejewski C., Pilat Z.: Optimisation Methods in Inverse Kinematics Solution for Redundant Manipulators. International Symposium on Industrial Robots ISIR'96, Milano, Italy, 1996.

[8] Kaczmarczyk A., Opis kinematyczny robotow. Biuletyn PIAP, 1987.

[9] Manseur R., Doty K.L. "A Fast Algorithm for Inverse Kinematic Analysis of Robot Manipulators." The International Journal of Robotics Research, vol. 7, no. 3, June 1988.

[10] Pachuta M., Budowa ukladow sterowania napedow robotow. Stan obecny i kierunki rozwoju. Biuletyn PIAP, 1992.

[11] Paul R. P., Robot Manipulators: Mathematics, Programming and Control, MIT Press, Cambridge, Mass., 1981r.

[12] Pilat Z. i in. "Wykonanie programu sterujacego dla robota RP-120. Zadanie 3.2 Wykonanie i uruchomienie oprogramowania realizujacego sterowanie typu CP." spraw. PIAP nr rej. 6490, Warszawa 1990.

[13] Robert Bosch GmbH - Układ sterowania robotów rho3 - materiały firmowe.

[14] Sciavicco L., Siciliano B. "A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulators." IEEE Journal on Robotics and Automation, vol. 4, no. 4 Aug. 1988

[15] Seraji H., Long M.K., Lee T.S. "Motion Control of 7-DOF Arms: The Configuration Control Approach." IEEE Transactions on Robotics and Automation, vol. 9, no. 2 Apr. 1993.

[16] Spong M. W., Robotic Dynamocs and Control. John Wiley & Sons, 1989.

[17] Withney D.E. "Resolved Motion Rate Control of Manipulators and Human Prostheses." IEEE Transactions on Man-Machine Systems, vol. MMS-10, no. 2 June 1969.

Wykorzystanie metod optymalizacji do realizacji sterowania CP dla robota o udźwigu 120kg.

Z. Pilat*

(konspekt referatu na konferencję MECHATRONIKA'97)

Przy realizacji sterowania CP dla robotów przemysłowych należy rozwiązać dwa podstawowe problemy. Pierwszy to kinematyka robota. Musi być opracowane rozwiązanie prostego i odwrotnego zadania kinematycznego. Drugim problemem jest planowanie trajektorii. Robot wykonując zaprogramowane zadanie operuje narzędziem. Sterowanie CP umożliwia interpolację trajektorii punktu roboczego narzędzia (TCP). Najczęściej spotyka się trajektorie prostoliniowe i kołowe w przestrzeni kartezjańskiej. Część mechaniczna robota o udźwigu 120kg została opracowana przed kilku laty w PIAP. Ma ona 6 stopni swobody, a więc pełną konfigurację kinematyczną. Zakładając, że używamy pełnego opisu narzędzia, można zastosować gotowe, sprawdzone algorytmy generacji trajektorii w przestrzeni kartezjańskiej. Przy próbie rozwiązania kinematyki odwrotnej napotyka się jednak poważne kłopoty. Konstrukcja nadgarstka robota sprawia, że nie jest możliwe uzyskanie rozwiązania tego zadania w postaci zamkniętych formuł analitycznych. problem można jednak rozwiązać stosując metody optymalizacyjne, podobnie jak się to czyni w przypadku redundantnych struktur kinematycznych. Charakterystyczną cechą tych algorytmów jest iteracyjne dochodzenie do zadanej dokładności rozwiązania. To nakłada wysokie wymagania na moc obliczeniową komputera sterującego. Chodzi o to, aby czas wygenerowania kolejnej pozycji pośredniej na trajektorii interpolowanej i rozwiązania dla tej pozycji odwrotnego zadania kinematycznego był rzędu kilku, kilkunastu milisekund. W ostatnim czasie udało się pomyślnie sprząc robota 120kg z dwoma nowoczesnymi układami sterowania: URP produkcji krajowej i rho3 produkcji niemieckiej firmy Bosch. Oba sterowniki posiadają wystarczającą moc obliczeniową do efektywnej realizacji sterowania CP.

* PIAP Warszawa

Date: ☺Fri, 10 Nov 1995 08:25:18 GMT
☺From: ☺gottwald@16.124.decnet.bosch.de
☺To: ☺zpilat@sp.piap.waw.pl
☺Subject: ☺rho3 trafo proc. descrip.
☺

von: Robert Bosch GmbH
Industrieausruestung
Geschaeftsbereich
Industrielle Steuerungselktronik

an: Industrielle Research Institue
Institute for Automation and Meassurements PIAP
Robotics & Intelligent Systems Laboratory (ZSI)

to Mr. Zbigniew Pilat

Sehr geehrter Herr Pilat,

nachfolgend finden Sie die Declarationen und Procedure-Beschreibung der Transformation.
Sollten Sie weitere Fragen haben,senden Sie mir eine Mail oder rufen Sie mich an.

Mit freundlichen Gruessen

H. Gottwald

****

```
(*/-A/-C/+D/+H/+I/+L/+O/-S/-T/-X/+W*)

PROGRAM T35 (INPUT,OUTPUT);


CONST  GRAD_RAD_FAKTOR = 0.017453292;  (* (2*PI)/360 *)
       RAD_GRAD_FAKTOR = 57.29577951;  (* 360/(2*PI) *)
       GRAD_90         = 1.570796327;          (* 1/2*PI     *)
       GRAD_180        = 3.141592654;          (* PI         *)
       GRAD_360        = 6.283185307;          (* 2*PI       *)
       PI_     = 3.1415927;
       PI_HALBE= 1.5707963;

       RESET   = 0;
       NICHT_AUFGETRETEN       = 0;
       AUFGETRETEN     = 1000;
       EPS_M5  = 0.00001;        (*RECHENUNGENAUIGKEIT*)

       MAX_ACHS     = 20;
       MAX_ARMLANG          =  8;       (*MAX-ANZ/KIN VON ARMLAENGEN IN TRAFO*
       MAX_KIN      = 10;


TYPE
       BYTE          =    -128..127;
       WORD          =  -32768..32767;
       BYTEFELD      =  ARRAY [1..MAX_ACHS]     OF BYTE;
       KORRFELD      =  ARRAY [1..6]    OF REAL;
       RAUMFELD      =  ARRAY [1..6]    OF REAL;
```

```
                REALFELD      =  ARRAY [1..MAX_ACHS]      OF REAL;


TDREH_MATRIX=  RECORD  CASE BYTE OF
                0 : (MAT          :ARRAY[1..3,1..3] OF REAL);
                1 : (E11,E12,E13,
                     E21,E22,E23,
             E31,E32,E33          :REAL);
        2 : (SIN1,SIN2,SIN3,
             COS1,COS2,COS3,
             WIN1,WIN2,WIN3       :REAL);
        3 : (DREH1_SIN,DREH2_SIN,DREH3_SIN,
             DREH1_COS,DREH2_COS,DREH3_COS,
             SCW1,SCW2,SCW3            :REAL);
        END;

TPOS_VEKTOR=   RECORD  CASE BYTE OF
        0 : (VEC          :ARRAY[1..3] OF REAL);
        1 : (V1,V2,V3     :REAL);
        2 : (GR1,GR2,GR3  :REAL);
        END;

TARM_VEKTOR=   RECORD  CASE BYTE OF
        0 : (VEC        :ARRAY[1..MAX_ARMLANG] OF REAL);
        1 : (L1,L2, L3, L4, L5, L6, L7, L8
             (*,L9,L10,L11,L12,L13,L14,L15,L16*)  :REAL);
        2 : (F1,F2, F3, F4, F5, F6, F7, F8
             (*,F9,F10,F11,F12,F13,F14,F15,F16*)  :REAL);
        END;

THOMO_TRAMA=   RECORD
           ORI   :TDREH_MATRIX;
           POS   :TPOS_VEKTOR;
        END;

TTRAN=      RECORD       (*WIRD IN TMPP UND TDPP BENUTZT: NUR AUFWAERTS- *)
          RTYP   :BYTE;  (*KOMPATIBEL AENDERN!!! *)
          GK_ERLAUBT     :BYTE;
          RESERV1_BYTE   :BYTE;  (*RESERVE*)
          RESERV2_BYTE   :BYTE;  (*RESERVE*)
          RK_OFFSET      :RAUMFELD;
          FLANSCH_KORR   :KORRFELD;
          ACHS_LANG      :TARM_VEKTOR;
          KOPPL_FAKTOR   :TARM_VEKTOR;
          RKS_ZUORD      :ARRAY [1..6] OF BYTE;
          RESERVE        :ARRAY [7..16] OF BYTE; (*RESERVE*)
            END;


TTRAFO_GLOBAL=RECORD
        (*---    TRAFO-LINK-EXPORT-VARIABLE    ----------*)
        SEMA_TRAFO        :BOOLEAN;
        DUMMY    :WORD;   (*RE_LI :TWOBI;*)
        TRANS_ERR         :INTEGER;
        GK_ERR   :INTEGER;
        SINCOS_EXPORT     :TDREH_MATRIX;
        NULLPKT_GK_ADR    :INTEGER;
        ALTE_RK_ADR       :INTEGER;
        G_MATRIX:THOMO_TRAMA;
        IG_MATRIX         :THOMO_TRAMA;
```

```
        (*---   LOKALE TRAFO-LINK-VARIABLE   ----------*)
        AXANZ   :INTEGER;
        AIND    :INTEGER;
        AIND_M1 :INTEGER;

        ARM     :TARM_VEKTOR;
        KOPPL   :TARM_VEKTOR;
        X0,Y0,Z0,
        O10,O20,O30     :REAL;
        SICO    :TDREH_MATRIX;
        GK_ERLAUBT      :BYTE;

        RKS_ORI :BYTE;
        RKS_POS :BYTE;
        DUM2_BY4:BYTE;

        RKSYS   :THOMO_TRAMA;
        I_RKSYS :THOMO_TRAMA;
        RKSYS_VEC       :KORRFELD;
        RKS_SA  :REAL;
        RKS_CA  :REAL;

        CASE RTYP  :BYTE OF     (*--- RTYP-ABHAENGIGE VARIABLEN-DEF --*)
         0 : (LOC_RESERVE       :ARRAY[1..64] OF INTEGER);

         3 : (GANTRY    :BYTEFELD;
              GANTRY_FAK:REALFELD);

         10,11,60,70,71,94 :
             (TOLERANZ  :REAL;          (*TOL-GRENZE FUER SINGULAERE STELLUNG*
              KOPPLUNG_4_12     :REAL;  (*KOPPLUNG MK4 MIT MK1,MK2 (INVERS) *)
              TOL_EINGEKLAPPT   :REAL;  (*TOL ARM 1,2 EINGEKLAPPT*)
              TOL_AUSGESTRECKT  :REAL); (*TOL ARM 1,2 AUSGESTRECKT*)

         14,15,16,17 :
             (A02       :TDREH_MATRIX;  (*LOC_RESERVE [1..9] *)
              MK_KOEFF  :TDREH_MATRIX;  (*LOC_RESERVE[10..18]*)
              INV_MK_KOEFF      :TDREH_MATRIX;  (*LOC_RESERVE[19..27]*)
              INV_A_MK4 :TDREH_MATRIX;  (*LOC_RESERVE[28..36]*)
              MK_B      :TPOS_VEKTOR);  (*LOC_RESERVE[37..39]*)

         34:
             (L56       :REAL;  (* L5+L6 *)
                       L2Q      :REAL;  (* L2*L2 *)
                       L3Q      :REAL;  (* L3*L3 *)
                       L2_MAL_2 :REAL;  (* 2*L2  *)
              PM_1      :BYTE;  (* +1 ODER -1 WG. KUKA*)
              EPS_LHW   :REAL;  (*TOL LHW_QUAD*)
              EPS_RELI  :REAL;  (*TOL FUER RELI*)
              EPS_EINGEKLAPPT   :REAL;  (*TOL ARM 2,3 EINGEKLAPPT*)
              EPS_AUSGESTRECKT  :REAL); (*TOL ARM 2,3 AUSGESTRECKT*)

         80,81,82 :
             (EINHEITS_GR       :THOMO_TRAMA);  (*GREIFER MIT EINHEITS-*)
                (*MATRIX ALS ORIENTIERUNG*)
         91,92,93 :
             (MK_RK_DEF :BYTE); (*KOORD-DEF. UEBER KOPPL.F1*)
              END;

FTRAFO_GLOBAL  =ARRAY [1..MAX_KIN] OF TTRAFO_GLOBAL;
```

```
VAR     TRAFO_GLOBAL    :FTRAFO_GLOBAL;
       MK_FELD,RK_FELD :REALFELD;
       KIN      :BYTE;

       I,J,TRF_CNT      :INTEGER;
       TRANS_ERR        :INTEGER;



(*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*)
 EXPORT FUNCTION ARC_TAN (Z,N :REAL) :REAL;
(*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*)
(*      *)
(*      ERMITTELT DEN 4-QUADRANTEN ARCUS-TANGENS IM BEREICH      *)
(*      -180 <= ARC_TAN <= 180   *)
(*      *)
(*------------------------------------------------------------------------------*)

  NST  EPS_ARC = 0.000001;

{2}     BEGIN
       IF ABS(N) > EPS_ARC
        THEN
{2.1}    BEGIN
         IF N > 0.0
          THEN
{2.1.1}    BEGIN
           ARC_TAN := ARCTAN (Z/N);
{2.1.1}    END
          ELSE
{2.1.2}    BEGIN
           IF Z > 0.0
            THEN
{2.1.2.1}     BEGIN
             ARC_TAN := ARCTAN (Z/N) + PI_;
{2.1.2.1}     END
            ELSE
  .1.2.2}     BEGIN
             ARC_TAN := ARCTAN (Z/N) - PI_;
{2.1.2.2}     END;
{2.1.2}    END;
{2.1}    END
        ELSE
{2.2}    BEGIN
         IF ABS(Z) > EPS_ARC
          THEN
{2.2.1}    BEGIN
           IF Z > 0.0
            THEN ARC_TAN :=  PI_HALBE
            ELSE ARC_TAN := -PI_HALBE
{2.2.1}    END
          ELSE
{2.2.2}    BEGIN
           ARC_TAN := 0.0;
{2.2.2}    END
{2.2}    END;
{2}    END;    (* VON ARC_TAN *)
```

```
(*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 EXPORT PROCEDURE VTRANP35  (VAR MK,RK :REALFELD; KIN :BYTE);
(*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

LABEL  99;

VAR    I       :FAST INTEGER;

       MK2, MK3, MK4, MK5, MK6,(*MESS-SYSTEM-WERTE IN GRAD*)
       O, A, T,(*RAUM-ORIENTIERUNGEN*)
       O_ALT, A_ALT, T_ALT,     (*ALTE ORIENTIERUNGSWINKEL*)
       A_1, A_2, A_NORM_ALT,          (*ALTE ORIENT. A AUF +-180 NORMIERT*)
       G1,G2,G3,G4,G5,G6,       (*GELENK-KOORDINATEN*)

       SA,CA,CA_QUAD,

       SIN_G1, SIN_G2, SIN_G3, SIN_G4, SIN_G5, SIN_G6,  (*BERECHNETE SINUS-/
       COS_G1, COS_G2, COS_G3, COS_G4, COS_G5, COS_G6,  (*COSINUS-WERTE

       A02_11, A02_12, A02_14,          (*UEBERTRAGUNGS-MATRIZEN FUER VTRAN*)
       A02_21, A02_22, A02_24,
                       A02_34,

       A03_11, A03_13, A03_14,
       A03_21, A03_23, A03_24,
       A03_31, A03_33, A03_34,

       A04_11, A04_13, A04_14,
       A04_21, A04_23, A04_24,
       A04_31, A04_33, A04_34,

       A05_11,
       A05_21,
       A05_31,

       A06_11,
       A06_21,
       A06_31  :REAL;

       A0F,A0E :THOMO_TRAMA;


       BEGIN
       TRF_CNT := TRF_CNT+1;

       WITH TRAFO_GLOBAL[KIN] ,ARM  DO
{1.1}  BEGIN

        MK2    := MK[2] * GRAD_RAD_FAkTOR;
        MK3    := MK[3] * GRAD_RAD_FAkTOR;
        MK4    := MK[4] * GRAD_RAD_FAkTOR;
        MK5    := MK[5] * GRAD_RAD_FAkTOR;
        MK6    := MK[6] * GRAD_RAD_FAkTOR;

        (* GELENKWINKEL UND ENTKOPPLUNG DER ACHSEN *)
        G1     := PM_1 * MK[1] * GRAD_RAD_FAkTOR;
        G2     := PM_1 * MK2;
        G3     := PM_1 * (MK3 + KOPPL.F4*MK2);
       G4     := PM_1 * MK4;
        G5     := PM_1 * (MK5 + KOPPL.F1*MK4);
```

```
          G6     := PM_1 * (MK6 + KOPPL.F2*MK4 + KOPPL.F3*MK5);

 O_ALT := RK[4] * GRAD_RAD_FAkTOR;
 A_ALT := RK[5] * GRAD_RAD_FAkTOR;
 T_ALT := RK[6] * GRAD_RAD_FAkTOR;

A_NORM_ALT := A_ALT;     (*A_ALT AUF +-180 NORMIEREN*)
WHILE A_NORM_ALT >  GRAD_180 DO  A_NORM_ALT := A_NORM_ALT - GRAD_360;
WHILE A_NORM_ALT < -GRAD_180 DO  A_NORM_ALT := A_NORM_ALT + GRAD_360;


 (* SINUS UND COSINUS BERECHNEN *)
 SIN_G1 := SIN (G1);
 SIN_G2 := SIN (G2);
 SIN_G3 := SIN (G3);
 SIN_G4 := SIN (G4);
 SIN_G5 := SIN (G5);
 SIN_G6 := SIN (G6);
 COS_G1 := COS (G1);
 COS_G2 := COS (G2);
 COS_G3 := COS (G3);
 COS_G4 := COS (G4);
 COS_G5 := COS (G5);
 COS_G6 := COS (G6);



IF CA_QUAD > EPS_M5
  THEN
    BEGIN        (* NORMALER FALL: CA<>0 *)



    END
  ELSE
    BEGIN
    IF CA_QUAD > -EPS_M5
     THEN
      BEGIN      (* SINGULAERER FALL: CA=0 *)
      IF SA > 0
       THEN BEGIN
    END
       ELSE BEGIN
    END;
      END
     ELSE        (*ARBEITSRAUM-BESCHRAENKUNG*)
      BEGIN
      TRANS_ERR := 1;
      GOTO 99;
      END;
    END;(*OF SINGULAERER FALL*)


(*--- O,A,T IN DEN BEREICH VON O_ALT,A_ALT,T_ALT BRINGEN ---*)
WHILE (O - O_ALT) > GRAD_180 DO  O := O - GRAD_360;
WHILE (O_ALT - O) > GRAD_180 DO  O := O + GRAD_360;

WHILE (T - T_ALT) > GRAD_180 DO  T := T - GRAD_360;
WHILE (T_ALT - T) > GRAD_180 DO  T := T + GRAD_360;

WHILE (A - A_ALT) > GRAD_180 DO  A := A - GRAD_360;
WHILE (A_ALT - A) > GRAD_180 DO  A := A + GRAD_360;
```

```
      RK[1] := A0E.POS.V1 + X0;
      RK[2] := A0E.POS.V2 + Y0;
      RK[3] := A0E.POS.V3 + Z0;

      RK[4] := O * RAD_GRAD_FAKTOR;
      RK[5] := A * RAD_GRAD_FAKTOR;
      RK[6] := T * RAD_GRAD_FAKTOR;

      FOR I:=7 TO AXANZ DO
          RK[I]:=MK[I];

      99:     (*FEHLER-AUSSPRUNG*)
      END;    (*OF WITH*)
      END;    (*OF VTRANP35*)
```

```
(*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
EXPORT PROCEDURE RTRANP35 (VAR RK,MK :REALFELD; KIN :BYTE);
(*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

LABEL  99;

VAR    I                :FAST INTEGER;

       OO, AA, TT,                                    (*RAUM-KOORDINATEN*)
       G1, G2, G3, G4, G5 ,G6, (*GELENKWINKEL*)
       G1_1, G1_2,G11_DEL,G12_DEL,
       G2_1, G2_2,
       G5_1, G5_2,
       G1_ALT, G2_ALT, G3_ALT, (*ALTE WERTE DER*)
       G4_ALT, G5_ALT, G6_ALT,          (*GELENKWINKEL  *)
       G1_NORM_ALT,
       G3_NORM_ALT,G5_NORM_ALT,
       MK2, MK3, MK4, MK5, MK6,
       MK2_ALT, MK3_ALT,
       MK4_ALT, MK5_ALT, MK6_ALT,       (*ALTE MESS-SYSTEM-WINKEL*)
       XG, YG, ZG,      (*HANDWURZELPUNKT*)

       SIN_G1, SIN_G2, SIN_G3, SIN_G4, SIN_G5, SIN_G6,
       COS_G1, COS_G2, COS_G3, COS_G4, COS_G5, COS_G6,

       XGQPYGQ, LHW_QUAD, LHW, DIFF, C5_QUAD,
       L56YG, L56XG, XGLHW, YGLHW,
       ZSIN1_1, ZCOS1_1, ZSIN1_2, ZCOS1_2,
       ZSIN2_1, ZCOS2_1, ZSIN2_2, ZCOS2_2,

       F8Q, A, B, D, AABB, ARG_WURZ, WURZ,
        DA, DB, BWURZ, AWURZ, K1, K2, L8Q, F,

       D12I_21, D12I_22,
       D12I_31, D12I_32,

         D456_13,
       D456_21, D456_22, D456_23,
       D456_31, D456_32, D456_33        :REAL;

       D123I    :TDREH_MATRIX;
```

```
            R1, R2   :THOMO_TRAMA;


        BEGIN
        WITH TRAFO_GLOBAL[KIN] ,ARM DO
{2.1}   BEGIN
        L8Q      := L8*L8;

         MK2_ALT := MK[2] * GRAD_RAD_FAkTOR;
         MK3_ALT := MK[3] * GRAD_RAD_FAkTOR;
         MK4_ALT := MK[4] * GRAD_RAD_FAkTOR;
         MK5_ALT := MK[5] * GRAD_RAD_FAkTOR;
         MK6_ALT := MK[6] * GRAD_RAD_FAkTOR;

         (* ALTE GELENKWINKEL UND ENTKOPPLUNG DER ACHSEN *)
         G1_ALT  := PM_1 * MK[1] * GRAD_RAD_FAkTOR;
         G2_ALT  := PM_1 * MK2_ALT;
         G3_ALT  := PM_1 * (MK3_ALT + KOPPL.F4*MK2_ALT);
        G4_ALT  := PM_1 * MK4_ALT;
         G5_ALT  := PM_1 * (MK5_ALT + KOPPL.F1*MK4_ALT);
         G6_ALT  := PM_1 * (MK6_ALT + KOPPL.F2*MK4_ALT + KOPPL.F3*MK5_ALT);


        (* G1_ALT, G3_ALT AUF +-180 NORMIEREN:  *)
        (* NUR NOTENDIG, WENN UEBERDREHUNGEN MOEGLICH SIND.     *)
        G1_NORM_ALT := G1_ALT;
        WHILE G1_NORM_ALT >  GRAD_180 DO  G1_NORM_ALT := G1_NORM_ALT - GRAD_36
        WHILE G1_NORM_ALT < -GRAD_180 DO  G1_NORM_ALT := G1_NORM_ALT + GRAD_36

         G3_NORM_ALT := G3_ALT;
        WHILE G3_NORM_ALT >  GRAD_180 DO  G3_NORM_ALT := G3_NORM_ALT - GRAD_36
        WHILE G3_NORM_ALT < -GRAD_180 DO  G3_NORM_ALT := G3_NORM_ALT + GRAD_36
        IF (G3_NORM_ALT>0) AND (G3_NORM_ALT < GRAD_180)
         THEN EPS_RELI := EPS_AUSGESTRECKT
         ELSE EPS_RELI := EPS_EINGEKLAPPT;

        (* G5_ALT AUF +-180 NORMIEREN:  IMMER NOTWENDIG *)
        G5_NORM_ALT := G5_ALT;
        WHILE G5_NORM_ALT >  GRAD_180 DO  G5_NORM_ALT := G5_NORM_ALT - GRAD_36
        WHILE G5_NORM_ALT < -GRAD_180 DO  G5_NORM_ALT := G5_NORM_ALT + GRAD_36


        (* WINKEL -> RAD *)
        OO := RK[4] * GRAD_RAD_FAKTOR;
        AA := RK[5] * GRAD_RAD_FAKTOR;
        TT := RK[6] * GRAD_RAD_FAKTOR;


        IF LHW_QUAD > EPS_LHW
          THEN
           BEGIN(*NORMALFALL*)


          END    (*OF NORMALFALL*)

         ELSE
          BEGIN (*LHW_QUAD < +EPS_LHW *)
          IF LHW_QUAD > -EPS_LHW
            THEN
            BEGIN

             END
```

```
          ELSE
     BEGIN
     TRANS_ERR := 5;      (* LHW_QUAD < -EPS: ARBEITSRAUM-BESCHRAENKUNG*
     GOTO 99;
     END;
    END; (*OF LHW_QUAD < +EPS_LHW *)


IF AABB < EPS_RELI
 THEN
  BEGIN          (* -E < AABB < EPS : SINGULARITAET *)
  TRANS_ERR := 4;        (*          HW-PKT LIEGT IN GELENK 2 *)
   GOTO 99
  END;


ARG_WURZ := AABB - D*D;
IF ARG_WURZ > EPS_RELI
 THEN WURZ := SQRT (ARG_WURZ)    (*NORMALFALL*)
  ELSE
  BEGIN
  IF ARG_WURZ >-EPS_RELI
   THEN WURZ := 0  (*SINGULAERE STELLUNG: ARM 2,3 AUSGESTRECKT / EINGE
   ELSE BEGIN
         TRANS_ERR := 2;                (*ARBEITSRAUM-BESCHRAENKUNG*)
        GOTO 99
        END;
  END;


    BEGIN        (*ARBEITSRAUM-BESCHRAENKUNG*)
    TRANS_ERR := 3;
     GOTO 99
    END;



(* HAND-GELENKWINKEL IM BEREICH -N*PI .. +M*PI     ; N, M > 0 *)
WHILE (G4 - G4_ALT) > GRAD_180 DO  G4 := G4 - GRAD_360;
WHILE (G4_ALT - G4) > GRAD_180 DO  G4 := G4 + GRAD_360;

WHILE (G6 - G6_ALT) > GRAD_180 DO  G6 := G6 - GRAD_360;
WHILE (G6_ALT - G6) > GRAD_180 DO  G6 := G6 + GRAD_360;

WHILE (G5 - G5_ALT) > GRAD_180 DO  G5 := G5 - GRAD_360;
WHILE (G5_ALT - G5) > GRAD_180 DO  G5 := G5 + GRAD_360;



(*--- ENTKOPPLUNG DER ACHSEN ---*)
MK2 := PM_1*G2;
MK3 := PM_1*G3 - KOPPL.F4*MK2;
MK4 := PM_1*G4;
MK5 := PM_1*G5 - KOPPL.F1*MK4;
MK6 := PM_1*G6 - KOPPL.F2*MK4 - KOPPL.F3*MK5;


(* GEKOPPELTE ACHSEN IN DEN BEREICH DER ALTEN WERTE BRINGEN. *)
(* NUR NOTWENDIG, WENN GROSSE KOPPLUNGSFAKTOREN VORHANDEN    *)
WHILE (MK3 - MK3_ALT) > GRAD_180 DO  MK3 := MK3 - GRAD_360;
```

```
          WHILE (MK3_ALT - MK3) > GRAD_180 DO  MK3 := MK3 + GRAD_360;

WHILE (MK5 - MK5_ALT) > GRAD_180 DO  MK5 := MK5 - GRAD_360;
WHILE (MK5_ALT - MK5) > GRAD_180 DO  MK5 := MK5 + GRAD_360;

WHILE (MK6 - MK6_ALT) > GRAD_180 DO  MK6 := MK6 - GRAD_360;
WHILE (MK6_ALT - MK6) > GRAD_180 DO  MK6 := MK6 + GRAD_360;



(* RAD -> WINKEL *)
MK[1] := PM_1*G1 * RAD_GRAD_FAKTOR;
MK[2] := MK2 * RAD_GRAD_FAKTOR;
MK[3] := MK3 * RAD_GRAD_FAKTOR;
MK[4] := MK4 * RAD_GRAD_FAKTOR;
MK[5] := MK5 * RAD_GRAD_FAKTOR;
MK[6] := MK6 * RAD_GRAD_FAKTOR;

FOR I:=7 TO AXANZ DO
 MK[I] := RK[I];

99:      (*FEHLER-AUSSPRUNG*)
END;     (*OF WITH*)
END;     (*OF RTRANP35*)
```