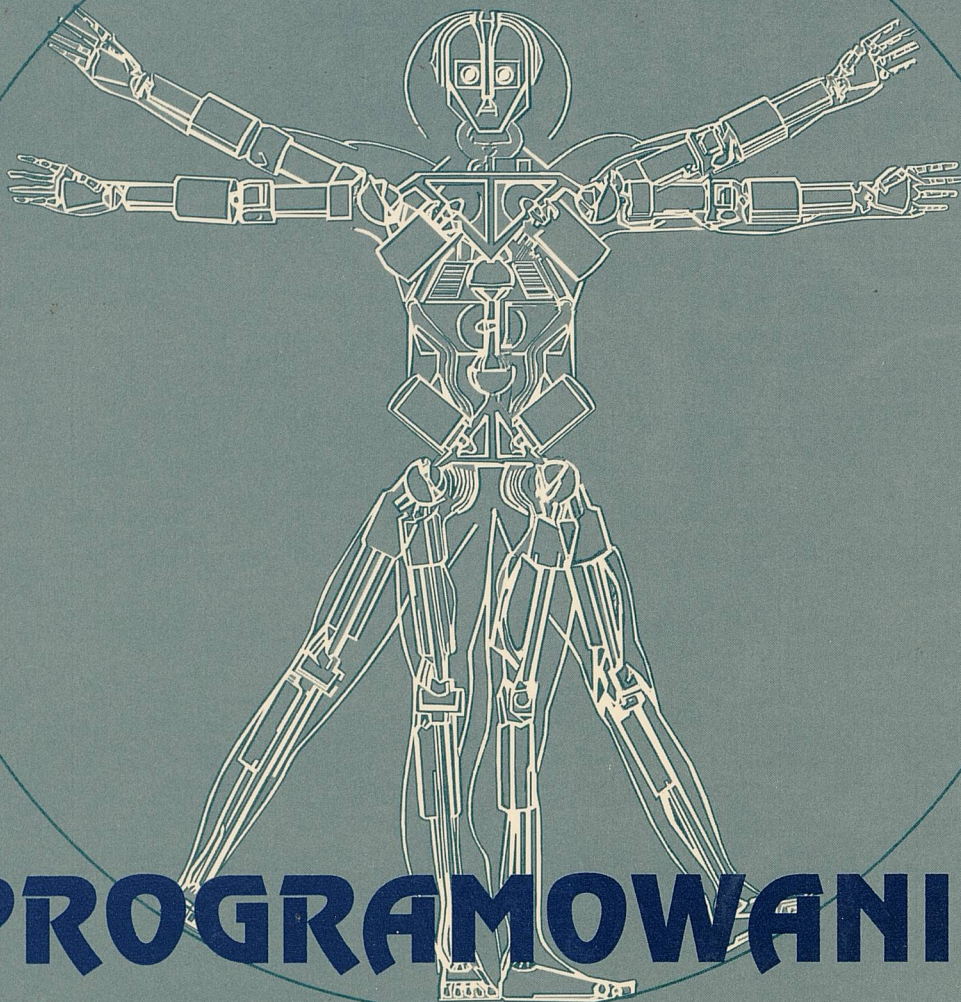


TOMASZ KOCH



**PROGRAMOWANIE
REDUNDANTNYCH
ROBOTÓW
PRZEMYSŁOWYCH**



Tomasz Koch

Programowanie redundantnych robotów przemysłowych



Oficyna Wydawnicza Politechniki Wrocławskiej · Wrocław 1996

Opiniodawcy
Józef KNAPCZYK
Andrzej MASŁOWSKI

Opracowanie redakcyjne i korekta
Alina KACZAK

Projekt graficzny okładki
Małgorzata BODAK

Biblioteka Główna i OINT
Politechniki Wrocławskiej



001799499



30 8244 L/V

© Copyright by Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1996

OFICyna WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

ISSN 0867-5325

CHiP Drukarnia Wrocław.

SPIS TREŚCI

1. WSTĘP	7
1.1. Robot redundantny	7
1.2. Cel pracy	10
2. PRZYKŁADY ZASTOSOWAŃ ROBOTÓW REDUNDANTNYCH	12
2.1. Zrobotyzowany montaż	12
2.2. Zrobotyzowane gniazdo spawalnicze	15
2.3. Zrobotyzowana kontrola pomiarowa skomplikowanego kształtu	18
2.4. Robot na wózku samojezdnym	19
2.5. Zrobotyzowane mycie samolotu	19
2.6. Robot w budownictwie	21
3. METODY STEROWANIA REDUNDANTNYMI ŁAŃCUCHAMI KINEMATYCZNYMI	23
3.1. Globalne i lokalne planowanie trajektorii ruchu	23
3.2. Kategorie redundantnych łańcuchów kinematycznych	26
3.3. Pseudoinwersja	31
3.4. Ważona pseudoinwersja	32
3.5. Wykorzystanie jądra macierzy Jacobiego	35
3.6. Inne formy rozwiązań za pomocą pseudoinwersji	42
3.7. Metoda tłumionych najmniejszych kwadratów odporna na konfiguracje osobliwe	44
3.8. Powtarzalność ruchu robota	45
3.9. Metoda rozszerzonego jakobianu	46
3.10. Sterowanie konfiguracją	48
3.11. Rozwiązanie w zamkniętej formie	51
3.12. Kompaktowa metoda programowania kwadratowego	52
3.13. Sterowanie bazujące na logice rozmytej	56
3.14. Sterowanie bazujące na sieciach neuronowych	56
3.15. Regulacja położenia	58
4. STEROWANIE ZORIENTOWANE NA PROGRAMOWANIE	62
4.1. Wymagania w stosunku do algorytmu sterowania	62
4.2. Kompaktowa metoda programowania kwadratowego w iteracyjnej metodzie Newtona-Raphsona	63
4.2.1. Utrzymanie wychyleń w połączeniach ruchowych i ich prędkości w zadanych granicach	65
4.2.2. Zapewnienie powtarzalności ruchu	67
4.2.3. Zmiana konfiguracji	68
4.2.4. Omijanie przeszkód	68
4.2.5. Zamrożenie ruchu w wybranych połączeniach ruchowych	73

4.3. Integracja modułu transformacji odwrotnej w sterowaniu robota o redundantnym łańcuchu kinematycznym	74
5. PROGRAMOWANIE ROBOTÓW O REDUNDANTNYCH ŁAŃCUCHACH KINEMATYCZNYCH	77
5.1. Programowanie robotów przemysłowych	77
5.2. Rozszerzenie tekstowego języka programowania dla programowania redundantnych łańcuchów kinematycznych	80
6. STEROWANIE 11-PRZEGUBOWYM ROBOTEM	90
6.1. Utrzymanie wychyleń w połączeniach ruchowych w zadanych granicach	94
6.2. Utrzymanie prędkości ruchu w połączeniach ruchowych w zadanych granicach	99
6.3. Utrzymanie przyspieszenia w połączeniach ruchowych w zadanych granicach	104
6.4. Zapewnienie powtarzalności ruchu	109
6.5. Zamrożenie ruchu w wybranych połączeniach ruchowych	109
6.6. Omijanie przeszkód	117
7. PODSUMOWANIE I WNIOSKI	124
8. LITERATURA	129

CONTENTS

1. INTRODUCTION	7
1.1. Robot with redundant kinematics	7
1.2. The aim of the work	10
2. EXAMPLES OF REDUNDANT ROBOT APPLICATIONS	12
2.1. Robotised assembly	12
2.2. Robotised welding cell	15
2.3. Robotised measurement control of a complicated shape	18
2.4. Mobile robot	19
2.5. Robotised airplanes washing	19
2.6. Robot in civil engineering	21
3. CONTROL METHODS OF REDUNDANT KINEMATICS	23
3.1. Global and local path planning	23
3.2. Categories of redundant kinematics	26
3.3. Pseudoinverse	31
3.4. Weighted pseudoinverse	32
3.5. Utilization of the Jacobian matrix kernel	35
3.6. Other solution forms using the pseudoinverse	42
3.7. Singularity robust damped least square method	44
3.8. Repeatability of the robot motion	45
3.9. Extended Jacobian method	46
3.10. Configuration control	48
3.11. Closed form solution	51
3.12. Compact quadratic programming method	52
3.13. Fuzzy logic based control	56
3.14. Neuronal network based control	56
3.15. Positioning control	58
4. CONTROL FOR PROGRAMMING	62
4.1. Requirements for control algorithm	62
4.2. Compact quadratic programming method in Newton-Raphson iteration scheme	63
4.2.1. Keeping the joints and their speeds within given limits	65
4.2.2. Assurance of motion repeatability	67
4.2.3. Change of the arms configuration	68
4.2.4. Obstacle,s avoidance	68
4.2.5. Moving the robot with some frozen joints	73
4.3. Integration of transformation module for the redundant kinematics in the robot control ..	74

5. PROGRAMMING OF ROBOTS WITH REDUNDANT KINEMATICS	77
5.1. Programming of industrial robots	77
5.2. Extension of textual programming language for programming of redundant kinematics ..	80
6. CONTROL OF 11-JOINT ROBOT	90
6.1. Keeping the joints within given limits	94
6.2. Keeping the joint's speeds within given limits	99
6.3. Keeping the joint's accelerations within given limits	104
6.4. Assurance of motion repeatability	109
6.5. Moving the robot with some frozen joints	109
6.6. Obstacle avoidance	117
7. SUMMARY AND CONCLUSIONS	124
8. REFERENCES	129

Tomasz KOCH*

PROGRAMOWANIE REDUNDANTNYCH ROBOTÓW PRZEMYSŁOWYCH

Przedstawiono liczne zastosowania robotów o redundantnych łańcuchach kinematycznych, a także przegląd metod sterowania tego typu łańcuchami, które należą do grupy metod lokalnego planowania trajektorii ruchu. Zestawiono też listę wymagań w stosunku do metod sterowania przemysłowymi robotami o redundantnych łańcuchach kinematycznych. Zaprezentowano następnie nowy oryginalny algorytm sterowania robotami redundantnymi, spełniający te wymagania. Polega on na zastosowaniu kompaktowej metody matematycznego programowania kwadratowego w iteracyjnej metodzie Newtona-Raphsona z elementami tzw. sterowania konfiguracją. Opisano kilka opcji tego algorytmu i sposób jego integracji w podstawowy tok przetwarzania danych w sterowaniu robota. Bazując na opracowanym algorytmie sterowania redundantnymi łańcuchami kinematycznymi, stworzono oryginalną koncepcję ich programowania, polegającą na rozszerzeniu tekstowego języka programowania robotów. Zaproponowane elementy tekstowego języka programowania zostały tak skonstruowane, że mogą stanowić rozszerzenie standardowego języka *PLR (Programming Language for Robots)* opracowywanego w ramach ISO, w celu umożliwienia programowania robotów redundantnych. Dla potwierdzenia efektywności opracowanej metody sterowania przetestowano kilka jej opcji na przykładzie stacjonarnego robota o sekwencyjnym, wielokrotnie redundantnym łańcuchu kinematycznym o 11 połączeniach ruchowych.

1. WSTĘP

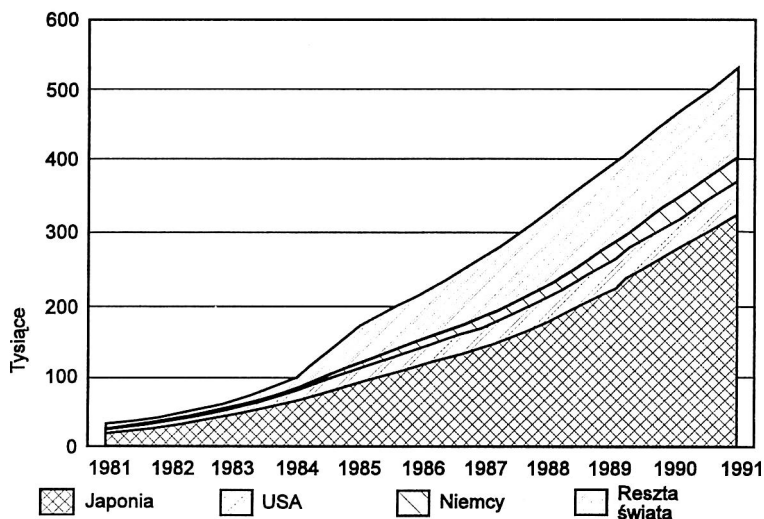
1.1. Robot redundantny

Robot jest to urządzenie manipulacyjne o wielu połączeniach ruchowych. Ruch względny elementów tych połączeń jest przy tym swobodnie programowalny, co odróżnia roboty od zwykłych manipulatorów. Ruch robota odbywa się według programu, w szczególności pod kontrolą sensorów. W ten sposób w zaleceniu niemieckiego VDI 2960 definiuje się robota.

*Instytut Technologii Maszyn i Automatyzacji Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

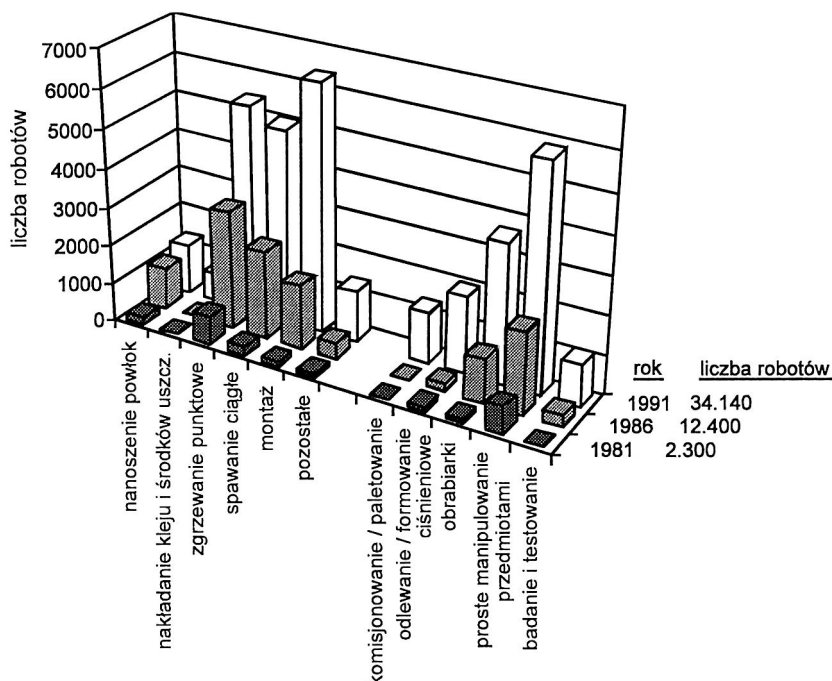
Roboty są znane w przemyśle od 1965 roku, a ich obszar zastosowań stale się powiększa i poczyną obejmować także takie dziedziny, jak usługi [Schraft 1993]. Źródłem rozwoju robotyki jest niewątpliwie stałe poszukiwanie coraz bardziej elastycznych sposobów automatyzacji. Na rysunku 1.1 pokazano wzrost liczby robotów na świecie.

Główne obszary zastosowania robotów przemysłowych to zgrzewanie punktowe, spawanie ciągłe, malowanie i nakładanie warstw oraz montaż (rys. 1.2). Wśród wielu zastosowań niektóre można nazwać już zastosowaniami tradycyjnymi. Obok nich pojawiają się nowe obszary, do których można zaliczyć: usuwanie zadziorów, szlifowanie taśmowe, oczyszczanie odlewów, nakładanie kleju i wytłaczanie uszczelki, montaż, w tym także montaż części miękkich, nakładanie taśmy oraz spawanie i cięcie laserem [N.N. 1993]. Niektóre z tradycyjnych obszarów zastosowań robotów przemysłowych wydają się osiągać punkt nasycenia. Jednocześnie w innych obszarach zastosowanie robotów natrafia na niewątpliwie trudności. Dalszy wzrost zastosowań robotów zależy z jednej strony od pojawiania się nowych technologii, które dają możliwość sprzęgnięcia robota z technicznym i technologicznym środowiskiem procesu wytwórczego [N.N. 1993] [Koch¹]. Z drugiej zaś strony wzrost ten zależy przede wszystkim od postępu w tych dziedzinach, które odpowiadają podstawowym zespołom składowym robota, a więc od nowych rozwiązań łańcuchów kinematycznych i metod ich sterowania, metod programowania i sensoryki. Rozwój w dziedzinie kinematyki i technik sterowania łańcuchami kinematycznymi wiąże się, bez wątpienia, przede wszystkim z zastosowaniem redundandantnych łańcuchów kinematycznych, co stanowi dalszy istotny krok na drodze poszukiwań coraz bardziej elastycznych urządzeń [Nakamura 1991].



Rys. 1.1. Wzrost liczby robotów na świecie (źródło: UN/ECE and IFR za [Morecki 1993])

Fig. 1.1. Increasing number of robots in the world (source: UN/ECE and IFR after [Morecki 1993])

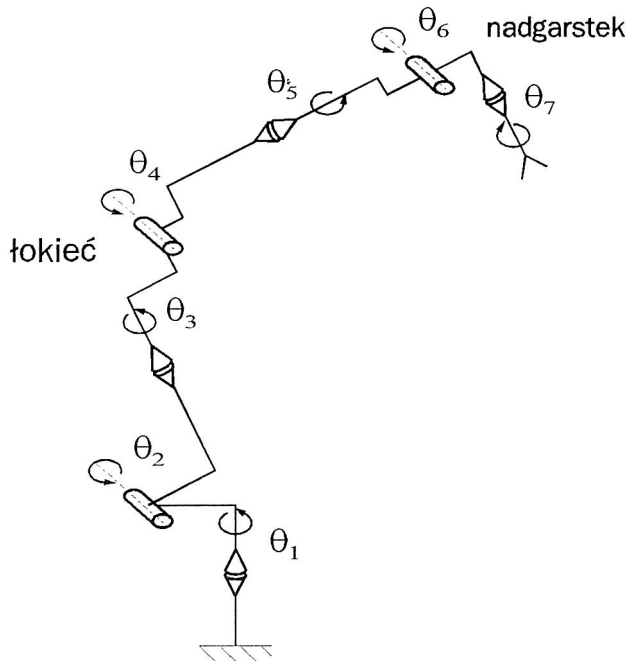


Rys. 1.2. Statystyka zastosowań w RFN robotów przemysłowych (źródło: Roboter za [N.N. 1993])

Fig. 1.2. Statistics of industrial robot application in Germany (source: Roboter after [N.N. 1993])

Redundantność w łańcuchu kinematycznym występuje wtedy, gdy robot ma więcej stopni swobody, co jest zwykle równoważne liczbie jego połączeń ruchowych, niż minimalna ich liczba potrzebna do zrealizowania określonego zadania. Ta definicja jest powszechnie używana w literaturze [Nenchev 1991] [Siciliano 1990]. Według tej definicji redundantność łańcucha kinematycznego robota musi być rozpatrywana w aspekcie realizowanego przez niego zadania. Jeśli zadanie polega przykładowo na manipulowaniu przedmiotem w przestrzeni, tak aby zajął dowolne położenie i orientację, to do realizacji takiego zadania wystarczy robot o 6 stopniach swobody. Robot o 7 stopniach swobody będzie dla takiego zadania redundantny. Przykładem może tu być niewątpliwie robot o łańcuchu kinematycznym, zbliżonym do łańcucha kinematycznego ludzkiej ręki, mający, upraszczając, trzy stopnie swobody w ramieniu, jeden w łokciu i trzy w nadgarstku [Kreutz-Delgado, Long, Seraji 1990] (rys. 1.3). Robot o takim łańcuchu kinematycznym odznacza się dużą zręcznością. Wiemy o tym z własnego doświadczenia, potrafiąc operować ręką w zadanej pozycji i przy tej samej orientacji na wiele różnych sposobów, tzn. przy wielu konfiguracjach ramienia i przedramienia.

Zręczność robotów o redundantnych łańcuchach kinematycznych jest źródłem ich większej elastyczności i wszechstronności niż robotów o konwencjonalnym łańcuchu kinematycznym i przyczyną coraz większego zainteresowania ich zastosowaniem.



Rys. 1.3. Model ludzkiej ręki [Kreutz-Delgado, Long, Seraji 1990]

Fig. 1.3. Model of human arm [Kreutz-Delgado, Long, Seraji 1990]

1.2. Cel pracy

Obiektem zainteresowania niniejszej pracy jest właśnie redundantny łańcuch kinematyczny, a zwłaszcza metody programowania robotów o takich łańcuchach kinematycznych. Duże zainteresowanie redundantnymi łańcuchami kinematycznymi, wsparte programami badawczymi w wielu krajach, przede wszystkim w USA i Japonii, dało wynik w postaci stworzenia wielu algorytmów sterowania. Niewiele z tych algorytmów zostało już praktycznie zastosowanych, a jedynie wypróbowano je w systemach symulacji. W istniejących przykładach robotów redundantnych bądź nie zastosowano żadnego takiego algorytmu i programowanie polega na mozolnym uczeniu metodą *teach-in*, bądź też zastosowano najprostsze i mało efektywne, nie stwarzając przy tym żadnego systemu programowania. Stan zaawansowania badań nad algorytmami sterowania redundantnymi łańcuchami kinematycznymi osiągnął już jednak taki poziom, że pozwala on na zrobienie następnego kroku, tzn. stworzenie koncepcji programowania ruchu redundantnych łańcuchów kinematycznych. Koncepcja programowania musi oczywiście bazować na przyjętej koncepcji sterowania, której wybór wpływa na możliwości i sposób programowania.

Celem pracy jest zaprezentowanie nowej oryginalnej koncepcji programowania robotów o redundantnych łańcuchach kinematycznych, dla czysto kinematycznie sformułowanych

zadań. Opracowana koncepcja obejmuje problem kompleksowo, proponując konkretne rozwiązania w sferach:

- algorytmu sterowania,
- regulacji położenia,
- metody programowania,
- integracji rozwiązań w otwarty system sterowania i programowania, nie tylko szeregowych łańcuchów kinematycznych, ale także robotów mobilnych i kooperujących manipulatorów.

W rozdziale 4. przedstawiono nowy oryginalny algorytm sterowania robotami redundantnymi, polegający na zastosowaniu kompaktowej metody matematycznego programowania kwadratowego w iteracyjnej metodzie Newtona-Raphsona z elementami tzw. sterowania konfiguracją. Punkt wyjścia do opracowania tego algorytmu stanowi lista wymagań w stosunku do metod sterowania robotami redundantnymi, którą autor zestawiał w podrozdziale 4.1, biorąc pod uwagę specyfikę zastosowań przemysłowych. W podrozdziale 4.2 przedstawiono szczegółowo opracowany algorytm, opisując kilka jego opcji, a w podrozdziale 4.3 sposób jego integracji w podstawowy tok przetwarzania danych w sterowaniu robota, na który składają się kolejno: interpretacja komend języka programowania, interpolacja, transformacja i regulacja położenia.

Bazując na opracowanym algorytmie sterowania redundantnymi łańcuchami kinematycznymi, autor stworzył oryginalną koncepcję ich programowania, polegającą na rozszerzeniu tekstowego języka programowania robotów. Uzasadnienie wyboru tej koncepcji znajduje się w podrozdziale 5.1, gdzie wskazano na zalety tekstowych języków programowania. Następnie w podrozdziale 5.2 przedstawiono oryginalnie opracowane elementy tekstowego języka programowania, które mogą stanowić rozszerzenie standardowego języka, np. *PLR* (*Programming Language for Robots*) opracowywanego w ramach ISO, w celu umożliwienia programowania robotów redundantnych.

Dla potwierdzenia efektywności opracowanej metody sterowania przetestowano jej kilka opcji na przykładzie stacjonarnego robota o sekwencyjnym wielokrotnie redundantnym łańcuchu kinematycznym o 11 połączeniach ruchowych. Wyniki tych testów, przeprowadzonych w systemie symulacji gniazda zrobotyzowanego, przedstawiono w rozdziale 6.

Integralną część tej pracy stanowią ponadto rozdziały 2. i 3. W rozdziale 2. przytoczono liczne zastosowania robotów redundantnych i stanowi to swoiste uzasadnienie ważności tematyki redundantnych łańcuchów kinematycznych, wynikające z potrzeb praktyki. Natomiast w rozdziale 3. dokonano przeglądu stworzonych już metod sterowania redundantnymi łańcuchami kinematycznymi, tych które należą do grupy metod lokalnego planowania trajektorii ruchu. Uzasadnienie wyboru lokalnego planowania trajektorii ruchu przedstawiono w podrozdziale 3.1. Ponadto, prócz metod sterowania, opisano także w rozdziale 3. kategorie redundantnych łańcuchów kinematycznych (podrozdział 3.2), jak również, nierozdzielnie łączący się ze sterowaniem manipulatorami, problem regulacji położenia (podrozdział 3.15).

2. PRZYKŁADY ZASTOSOWAŃ ROBOTÓW REDUNDANTNYCH

Zbudowano już wiele robotów o redundantnych łańcuchach kinematycznych. Duża liczba z nich to roboty skonstruowane w laboratoriach badawczych. Także pewna ich liczba jest używana w praktyce, np. 7-przegubowy robot K10AS firmy Motoman [Industrial Robot 1993]. Przykłady tutaj zamieszczone mają, między innymi, pokazać sytuacje, w których istnieje potrzeba zastosowania redundantnych łańcuchów kinematycznych.

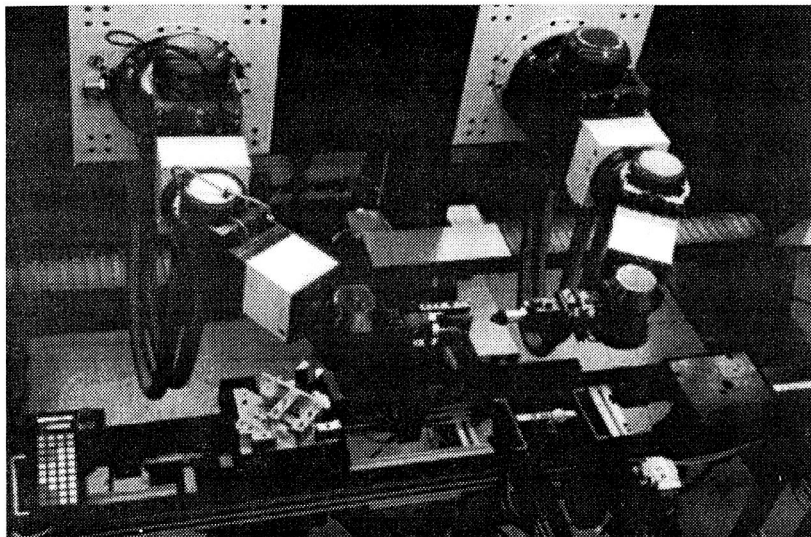
2.1. Zrobotyzowany montaż

Montaż jest czynnością, która występuje podczas produkcji prawie wszystkich produktów przemysłu maszynowego i wiąże się z dużym nakładem pracy. Z powodu skomplikowania procesów montażowych, pomimo znacznej elastyczności budowanych gniazd montażowych [Koch, Kozera, Kocełuch, Bielski, Dziuba 1993] [Bielski, Dziuba, Kocełuch, Koch, Kozera 1993], zautomatyzowany montaż znalazł tylko gdzieś tam przemysłowe zastosowanie. Na prostotę i efektywność ekonomiczną montażu, także wspartego robotem, zasadniczy wpływ ma konstrukcja wszystkich montowanych części. Z tego też powodu rozwijane są metodologie projektowania zorientowanego na montaż (Design for Assembly – DFA) [Boothroyd 1991] [Bielski, Koch, Chlebus 1993], których zadaniem jest doskonalenie montażowej technologiczności konstrukcji. Pomimo rozwoju i stosowania tego typu metodologii, konstrukcje wyrobów lub ich podzespołów wymagają niekiedy zastosowania na tyle skomplikowanych czynności montażowych, że ich zautomatyzowanie wymaga użycia robotów o redundantnych łańcuchach kinematycznych, przeważnie, jak pokazują podane dalej przykłady, robotów dwuramiennych.

Na rysunku 2.1 przedstawiono zrobotyzowane gniazdo montażowe stworzone na Uniwersytecie w Stuttgarcie [Pritschow, Bauder 1990]. Centralną częścią tego gniazda jest dwuramienny robot, którego każde z ramion ma 6 stopni swobody. Przykład montażu, którego automatyzacja wymaga zastosowania dwu ramion, to montaż wymagający równoczesnego skoordynowanego wsuwania w obudowę dwóch części (rys. 2.2). Efektywne wykorzystanie robota dwuramiennego i sprawne programowanie wymaga jednoczesnego sterowania wszystkimi 12 osiami i traktowania tego łańcucha kinematycznego jako redundantnego. Z braku dostępnych metod w tym zakresie, oba ramiona sterowane są tymczasem wg zasady *master-slave*, co oznacza sterowanie jednym ramieniem (*master*), a drugie (*slave*) nadąża zgodnie z zadaniem, w zależności od ruchu wykonywanego przez pierwsze ramię.

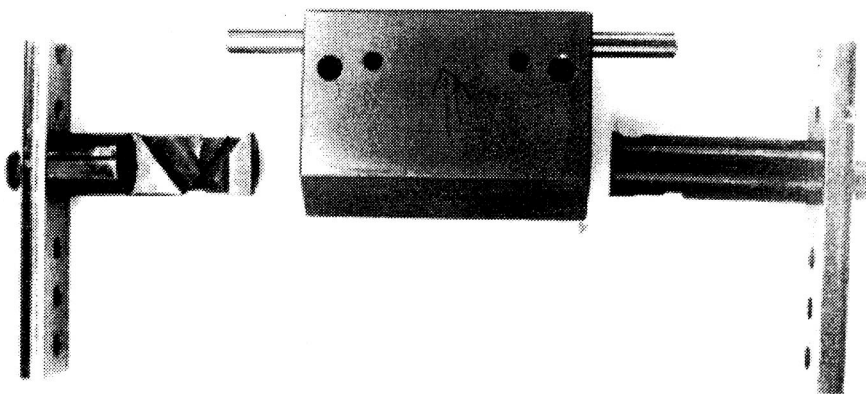
Również na Uniwersytecie w Karlsruhe zbudowano dwuramienny robot mobilny (rys. 2.3) [Roboter 1993].

Innym przykładem gniazda montażowego, w którym zastosowano także dwuramienny robot (rys. 2.4), pochodzi z firmy Toyota [Nagamatsu, Sugiura, Sato 1992]. Robot



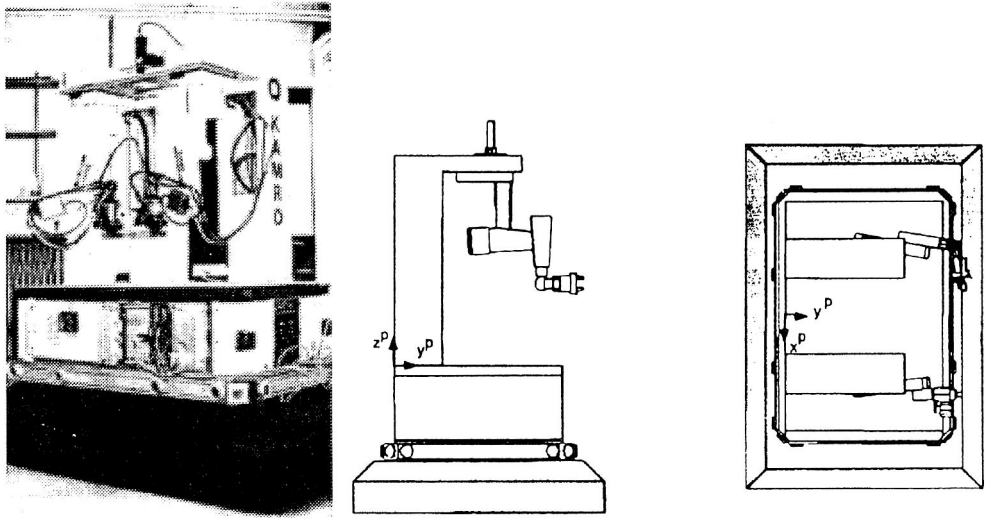
Rys. 2.1. Zrobotyzowane gniazdo montażowe z dwuramiennym robotem
(źródło: ISW, Uniwersytet Stuttgart)

Fig. 2.1. Robotised assembly cell with a dual arm robot (source: ISW, University of Stuttgart)



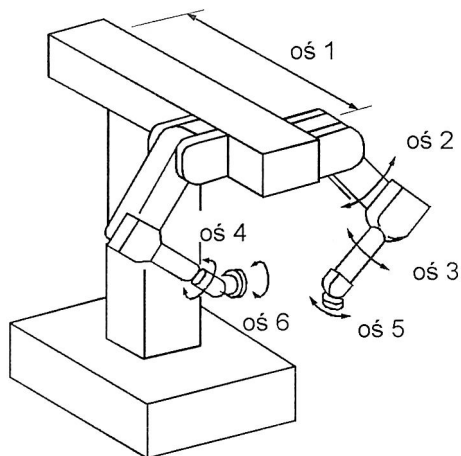
Rys. 2.2. Montaż chwytaka, który wymaga jednoczesnego skoordynowanego
wsuwania dwóch części w obudowę

Fig. 2.2. Assembly of gripper requiring simultaneous and coordinated
sneaking of two parts in the housing



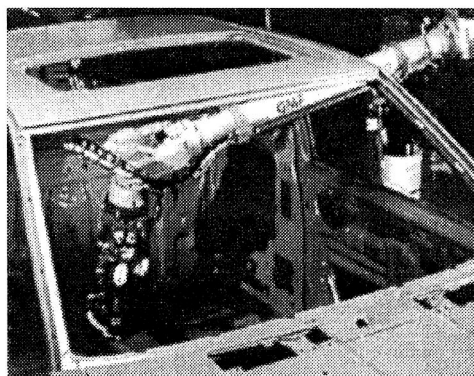
Rys. 2.3. Dwuramienny robot mobilny z Uniwersytetu w Karlsruhe [Roboter 1993]

Fig. 2.3. Dual arm mobile robot from University of Karlsruhe [Roboter 1993]



Rys. 2.4. Dwuramienny robot do montażu mechanizmu różnicowego i kolumny kierownicy z firmy Toyota [Nagamatsu, Sugiura, Sato 1992]

Fig. 2.4. Dual arm robot for assembly of differential gear and steering wheel column from Toyota company [Nagamatsu, Sugiura, Sato 1992]



Rys. 2.5. Robot manipulujący wewnątrz kabiny samochodu (źródło: GMFanuc)

Fig. 2.5. Robot manipulating inside the body of a car (source: GMFanuc)

dwuramienny okazał się tu niezbędny do automatyzacji montażu mechanizmu różnicowego i kolumny kierownicy.

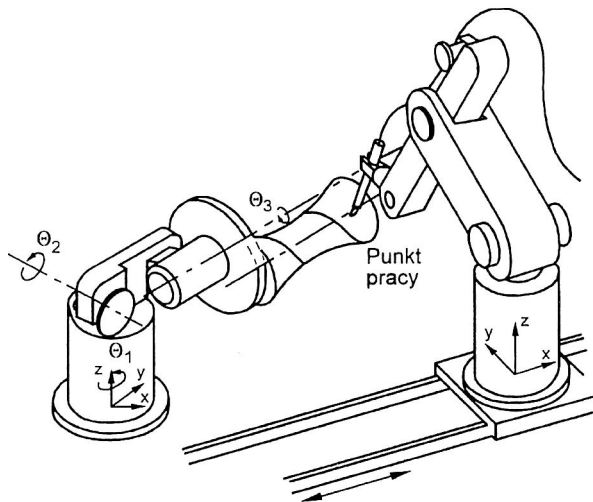
Użycie redundantnych łańcuchów kinematycznych jest niekiedy wymagane nie ze względu na konstrukcję wyrobu, lecz przez środowisko, w którym jest dokonywany montaż. Na przykład, konieczne może okazać się użycie robota redundantnego, o szeregowym otwartym łańcuchu kinematycznym, przy pracach montażowych oraz manipulacyjnych podczas montażu samochodu (wkładanie i mocowanie kół zapasowych, akumulatora czy innych części wewnątrz kabiny, jak to pokazano na rys. 2.5). Celem stosowania w tych zadaniach redundantnego łańcucha kinematycznego jest uniknięcie kolizji w mocno ograniczonych przestrzeniach.

2.2. Zrobotyzowane gniazdo spawalnicze

Na rysunku 2.6 pokazano schematycznie zrobotyzowane gniazdo spawalnicze składające się ze stołu, na którym mocuje się przedmiot spawany, i robota spawalniczego. Stół ma dwa stopnie swobody, których zadaniem jest odpowiednia orientacja chwilowego punktu spawania, tak aby chwilowa normalna wskazywała pionowo ku górze. Zastosowanie sześcioprzegubowego robota, umieszczonego dodatkowo na szynie, stwarza redundantność w całym systemie. Zręczność uzyskana przez redundantność jest potrzebna do spawania przedmiotu o długim i skomplikowanym kształcie.

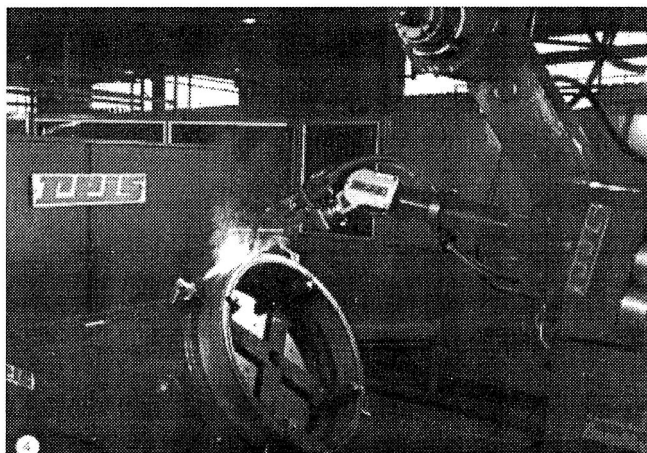
Podobne gniazdo spawalnicze, ze stołem do mocowania przedmiotu, przedstawiono na rysunku 2.7. Dziesięć sterowanych osi umożliwia tu ciągłe utrzymanie punktu spawania w ściśle określonej orientacji w przestrzeni.

Jeszcze bardziej uniwersalne i zarazem bardziej redundantne gniazdo przedstawia rysunek 2.8, gdzie rolę stołu do mocowania przedmiotu pełni drugi robot.



Rys. 2.6. Schematyczne przedstawienie zrobotyzowanego gniazda spawalniczego składającego się z robota i stołu, na którym mocuje się przedmiot spawany [Ahmad, Luo 1989]

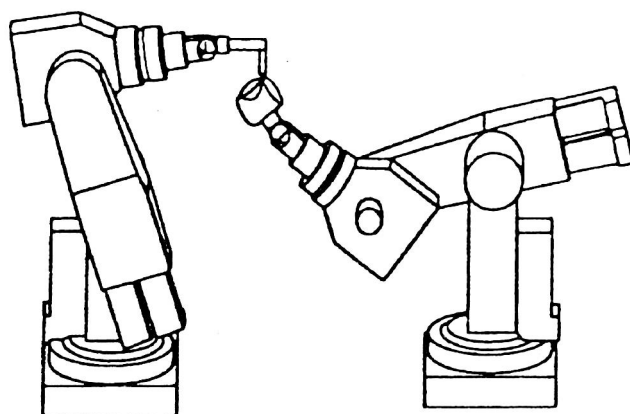
Fig. 2.6. Sketch of the robotised welding cell consisting of robot and table for fixing the workpiece [Ahmad, Luo 1989]



Rys. 2.7. Robot spawalniczy ze stołem do mocowania przedmiotu. System o 10 sterowanych osiach do spawania w osłonie gazowej (źródło: firma Reis)

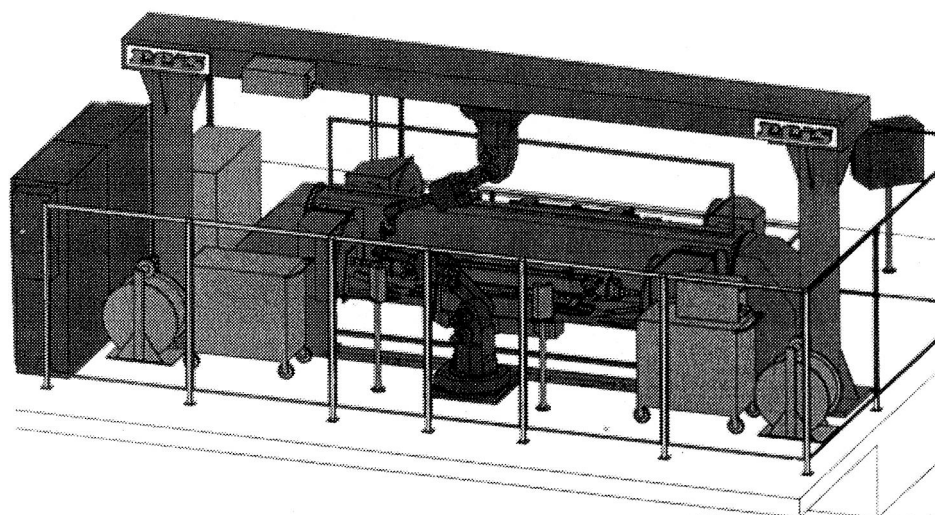
Fig. 2.7. System for electric welding with robot and table for fixing the workpiece. The system has 10 numerically controlled axis (source: Reis company)

Jak widać, systemy o redundantnym łańcuchu kinematycznym dwóch robotów są stosowane nie tylko do zautomatyzowanego montażu, jak to pokazano w podrozdziale 2.1, ale także w zadaniach spawalniczych, których jeszcze jednym przykładem jest system dwóch spawających robotów (rys. 2.9).



Rys. 2.8. System dwóch kooperujących sześcioprzegubowych robotów Manutec R2 zrealizowany na Technicznym Uniwersytecie w Berlinie [Münch 1991]

Fig. 2.8. System of two cooperating six joints robots Manutec R2 installed at the Technical University of Berlin [Münch 1991]



Rys. 2.9. System spawalniczy dwóch robotów, każdy o 6 stopniach swobody, pracujących w systemie master-slave (źródło: firma Reis)

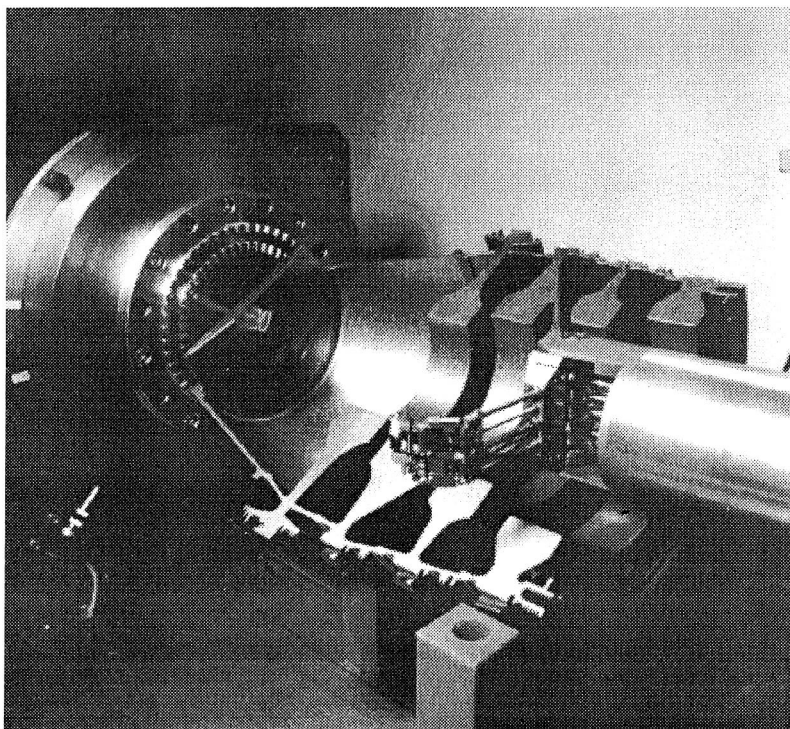
Fig. 2.9. Welding system consisting of two robots both with 6 degrees of freedom working in the master-slave system (source: Reis company)

Nowe rozwiązania w dziedzinie spawania ciągłego idą w kierunku lepszego dostosowania ruchu do parametrów procesu spawania poprzez obserwację jeziora ciekłego metalu i koordynację ruchu robota oraz osi pomocniczych stołu mocującego przedmiot, bądź też układu dwóch robotów.

2.3. Zrobotyzowana kontrola pomiarowa skomplikowanego kształtu

Inny rodzaj zadania, które często wymaga użycia robota o redundantnym łańcuchu kinematycznym, to zadanie określone w stosunku do przedmiotów o trudno dostępnych i skomplikowanych kształtach. Przykładem może tu służyć robot skonstruowany do celów kontroli pomiarowej rys w wirniku turbiny samolotowej [Angebauer, Rentschler 1991]. Na rysunku 2.10 przedstawiono przekrój wirnika turbiny z widocznym ramieniem robota.

Wirnik jest umocowany w uchwycie i może być obracany wokół swojej osi. Samo ramię robota ma sześć dalszych stopni swobody: dwie osie translacyjne i cztery rotacyjne. Całość urządzenia ma więc siedem osi sterowanych i jest wielokrotnie kinematycznie redundantne ze względu na to, że sześć osi samego ramienia pozycjonuje i nadaje orientację czujnikowi pomiarowemu umieszczonemu na końcówce w jednej tylko płaszczyźnie, a już trzy stopnie swobody wystarczają do pełnego określenia tego zadania. Tu również zaznaczył się deficyt efektywnych metod sterowania i programowania traktujących łańcuch kinematyczny tego urządzenia jako redundantny [Angebauer, Rentschler 1991]. Programowanie odbywa się przez mozolne uczenie metodą *teach-in*.



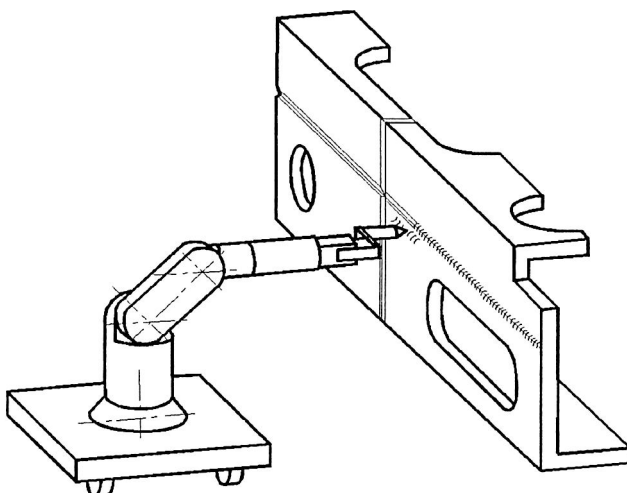
Rys. 2.10. Przekrój wirnika turbiny z widocznym ramieniem robota
(źródło: ISW, Uniwersytet Stuttgart)

Fig. 2.10. Section of a turbine rotor with robot arm (source: ISW, University of Stuttgart)

2.4. Robot na wózku samojezdnym

W przemysłowym zastosowaniu powszechne jest, zwłaszcza w elastycznych systemach produkcyjnych, posługiwanie się samojezdnymi systemami transportowymi (ang. Automated Guided Vehicle – AGV). Bardziej zaawansowane systemy w tym zakresie to roboty mobilne, stanowiące integrację wózka samojezdnego z robotem. Ponieważ wózek ma zazwyczaj trzy stopnie swobody, roboty mobilne stają się redundantnymi systemami kinematycznymi, które przewyższają zdecydowanie w elastyczności i ruchliwości inne urządzenia.

Konstruowane roboty mobilne (rys. 2.3 i rys. 2.11) znajdują zastosowanie zarówno w przemyśle, usługach [Roboter 1993] [Schraft 1993], jak i w budownictwie [Pritschow, Dalacker, Kurz 1993].



Rys. 2.11. Robot mobilny [Miksch, Schröder 1993]

Fig. 2.11. Mobile robot [Miksch, Schröder 1993]

2.5. Zrobotyzowane mycie samolotu

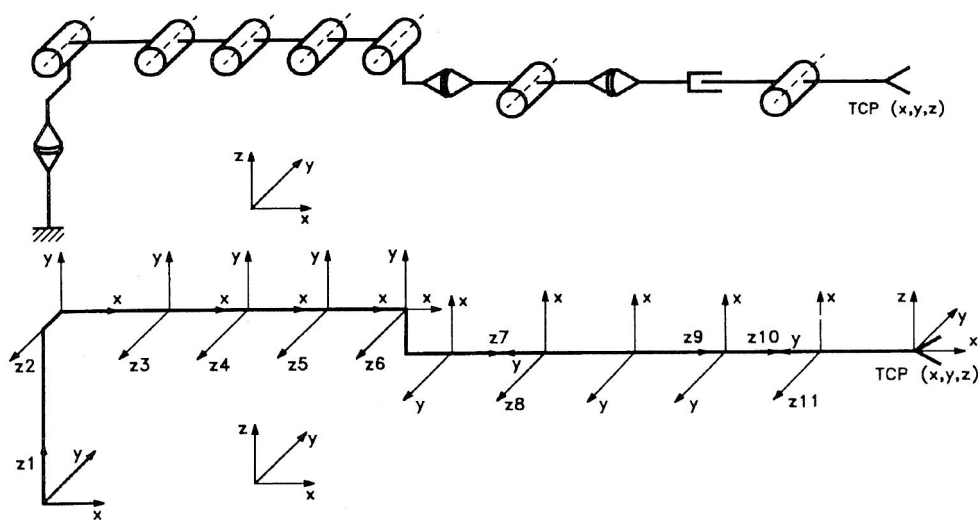
Zupełnie inne rozmiary ma robot pochodzący z firmy AEG. Na rysunku 2.12 przedstawiono robota olbrzyma o nazwie *Skywash*, służącego do automatyzacji mycia samolotów [Schumacher, Braun 93].

Zastosowanie pierwszych robotów *Skywash*, pochodzących z seryjnej produkcji, przewidywano na połowę 1994 roku. Robot *Skywash* ma aż 11 osi sterowanych (rys. 2.13) i przez to wysoce redundantny łańcuch kinematyczny. W sterowaniu tym robotem zastosowano już jedną z metod sterowania redundantnymi łańcuchami kinematycznymi, aczkolwiek jedną z najprostszych.



Rys. 2.12. Skywash - robot olbrzym służący do mycia samolotów (źródło: firma AEG)

Fig. 2.12. Skywash - giant robot for washing of airplanes (source: AEG company)



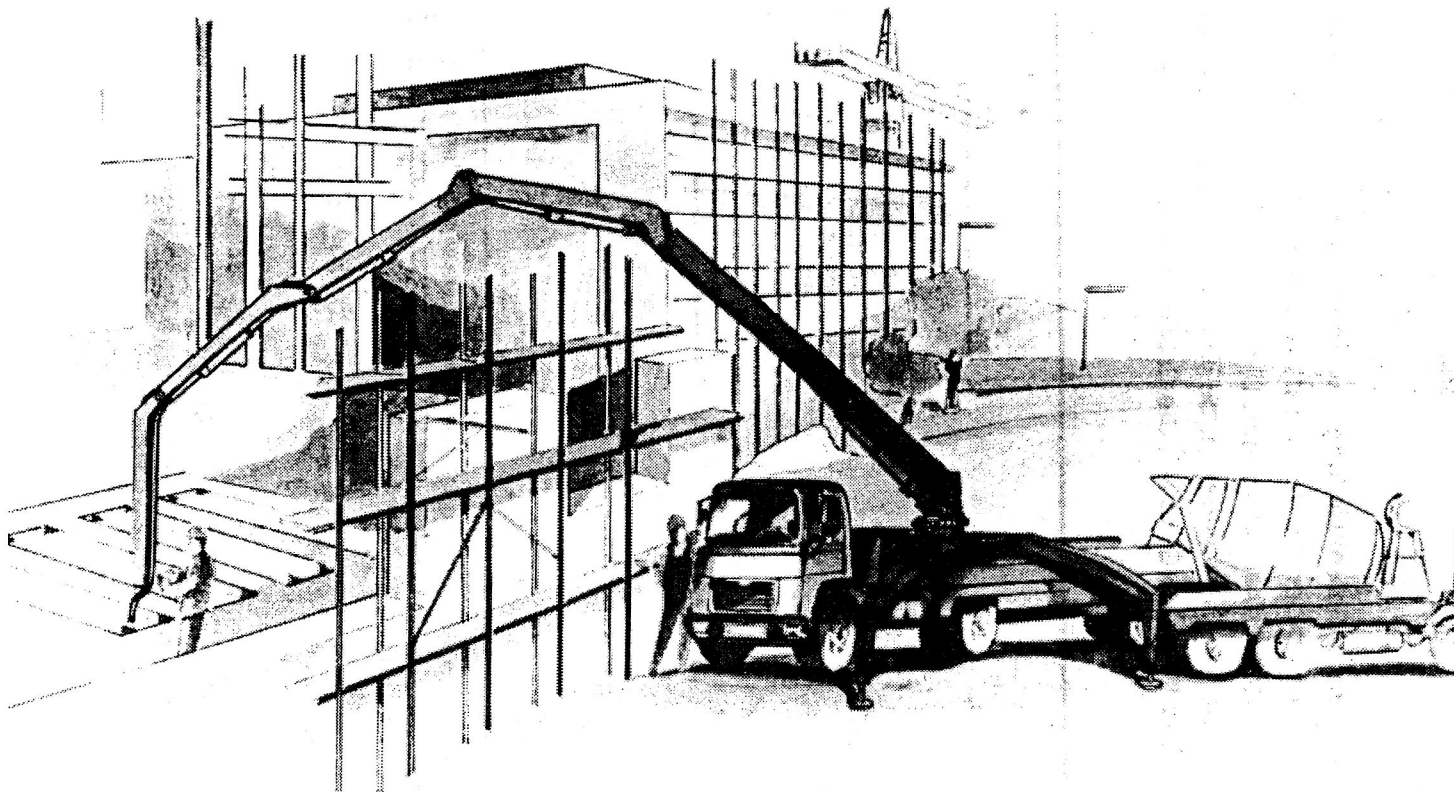
Rys. 2.13. Schemat kinematyczny robota *Skywash* [Schumacher, Braun 1993]

Fig. 2.13. Kinematics of Skywash robot [Schumacher, Braun 1993]

2.6. Robot w budownictwie

Podobnie olbrzymich rozmiarów jest robot EMJR (*Extended Multi Joint Robot*), skonstruowany dla niemieckiej firmy Putzmeister [Höfer 1992]. Na rysunku 2.14 przedstawiono szkic tego robota o zasięgu ramienia 22 metrów i udźwigu 1,4 tony.

Maszt tego robota, który może się obracać na platformie samochodu wokół pionowej osi, ma pięć dalszych obrotowych połączeń ruchowych o osiach równoległych. Ruch samego masztu odbywa się więc tylko w jednej płaszczyźnie, a powstała redundancja daje możliwość zastosowania robota do wielu zadań, w tym również do omijania przeszkód.



Rys. 2.14. Robot EMJR do stosowania w budownictwie [Höfer 1992]
Fig. 2.14. Robot EMJR for the application in civil engineering [Höfer 1992]

3. METODY STEROWANIA REDUNDANTNYMI ŁAŃCUCHAMI KINEMATYCZNYMI

3.1. Globalne i lokalne planowanie trajektorii ruchu

Robot o redundantnym łańcuchu kinematycznym, mający więcej stopni swobody aniżeli minimalna liczba potrzebna do zrealizowania postawionego zadania, może to zadanie wykonać na nieskończenie wiele sposobów – w sensie konfiguracji swojego łańcucha kinematycznego. Robot taki, prócz głównego zadania – prowadzenia swojego członu roboczego wzdłuż zadanego toru ruchu – może więc wykonać także inne zadania, wynikające z dodatkowych warunków lub otoczenia. Przykłady dodatkowych zadań to:

- omijanie przeszkód,
- utrzymanie wychyleń w połączeniach ruchowych w określonych granicach,
- unikanie konfiguracji osobliwych,
- minimalizacja zużycia energii

i wiele innych, które opisano w dalszej części tego rozdziału. W celu opracowania algorytmów sterowania dodatkowe zadania należy opisać matematycznie w postaci równań lub kryteriów optymalizacji. Opis matematyczny umożliwia wybór jednej spośród nieskończenie wielu konfiguracji łańcucha kinematycznego, które wszystkie zapewniają spełnienie głównego zadania.

Wśród algorytmów sterowania robotami o redundantnych łańcuchach kinematycznych można wyróżnić dwie podstawowe kategorie:

- metody globalne, które bazują na informacji o całej trajektorii członu roboczego (chwytaka lub narzędzia) robota,
- metody lokalne, które bazują na informacji o wąskim wycinku trajektorii z otoczenia aktualnego położenia członu roboczego.

Metody globalne umożliwiają znalezienie takiego ciągu konfiguracji łańcucha kinematycznego, który realizuje zadany tor ruchu członu roboczego i optymalizuje ten ruch wg zadanych kryteriów, takich jak np. całkowity czas realizacji zadania czy całkowita energia zużyta na zadanie. Zadanie globalnej optymalizacji jest zwykle formułowane następująco [Galicki 1992]: dla danego toru ruchu $\mathbf{x}(t)$, określonego w przestrzeni kartezjańskiej, należy znaleźć taką trajektorię $\boldsymbol{\theta}(t)$ w przestrzeni współrzędnych konfiguracyjnych, która optymalizuje kryterium w postaci całki po czasie. Można to opisać za pomocą następujących równań:

$$\mathbf{x}(t) = \mathbf{f}(\boldsymbol{\theta}(t)) \quad \text{ i } \quad \min \left\{ \int_{t_s}^{t_e} K(\ddot{\boldsymbol{\theta}}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}) dt \right\}, \quad (1)$$

gdzie: f – transformacja prosta,
 $[t_s, t_e]$ – przedział czasu,
 K – tzw. funkcja kosztu,
 $\theta, \dot{\theta}, \ddot{\theta}$ – zmienne konfiguracyjne i ich pochodne po czasie,
 x – współrzędne kartezjańskie.

Przykładami funkcji kosztu mogą być następujące dwie funkcje [Nakamura, Hanafusa 1987]:

$$K = \dot{\theta}^T \dot{\theta} + cV(\theta) \quad \text{lub} \quad K = T^T T + cV(\theta),$$

gdzie: T – wektor sił/momentów w połączeniach ruchowych,
 V – dodatkowa funkcja, np. przyjmująca coraz większe wartości wraz ze zbliżaniem się robota do przeszkody lub jego konfiguracji do konfiguracji osobliwej,
 c – stały współczynnik.

Pierwsza z tych funkcji kosztu przyczynia się do możliwie małych prędkości, a druga małych sił/momentów w połączeniach ruchowych w przedziale czasu $[t_s, t_e]$.

Niektórzy badacze [Kazerounian, Wang 1988] prównywali obie kategorie metod, globalnej i lokalnej optymalizacji, na przykładzie funkcji kosztu $K = \dot{\theta}^T \dot{\theta}$. Z porównania tego wynika, że globalna minimalizacja prędkości ruchu w połączeniach ruchowych, przy tak określonej funkcji kosztu, odpowiada lokalnej minimalizacji przyspieszeń ruchu połączeń ruchowych według kryterium optymalizacji $\ddot{\theta}^T \ddot{\theta}$.

Metody globalnej optymalizacji umożliwiają znalezienie trajektorii w przestrzeni współrzędnych konfiguracyjnych $\theta(t)$ z akceptowalnymi prędkościami ruchu w połączeniach ruchowych $\dot{\theta}(t)$, które nie są zbyt duże, by nie mogły być zrealizowane przez rzeczywiste układy napędów, połączeń ruchowych i ramion. Metody te ograniczone są w zasadzie do planowania *off-line* trajektorii i nie pozwalają na dokonywanie w czasie rzeczywistym korekcji toru ruchu, np. na podstawie sygnałów od sensorów [Baker, Wampler 1988].

W przeciwieństwie do nich, metody lokalnej optymalizacji optymalizują kryteria tylko np. wewnątrz jednego taktu interpolacji i umożliwiają dokonywanie korekcji toru ruchu w czasie rzeczywistym na podstawie sygnałów od sensorów. Metody lokalne nie są jednak w ogólności wolne od problemu zbyt dużych prędkości ruchu w połączeniach ruchowych, tzn. mogą prowadzić do znajdowania takich trajektorii $\theta(t)$ w przestrzeni współrzędnych konfiguracyjnych, że prędkości ruchu w połączeniach ruchowych $\dot{\theta}(t)$ mogą być zbyt duże do zrealizowania przez rzeczywiste układy napędów, połączeń ruchowych i ramion. Niemniej jednak, przy wielu zastosowaniach robotów, także przemysłowych, sprzężenie zwrotne od sensorów jest tak istotne, że monografia ta koncentruje się wyłącznie na metodach lokalnych, pomimo nie rozwiązanego w nich generalnie problemu zbyt dużych prędkości ruchu w połączeniach ruchowych.

Metody lokalnej optymalizacji formułują zadanie szukania trajektorii $\theta(t)$ w dziedzinie prędkości, różniczkując względem czasu równanie transformacji prostej $\dot{x}(t) = f(\theta(t))$, tak że

$$\dot{x}(t) = J(\theta) \dot{\theta}(t), \quad (2)$$

gdzie: $\dot{x}$ – m -wymiarowy wektor prędkości członu roboczego w przestrzeni kartezjańskiej,

$\dot{\theta}$ – wymiarowy wektor prędkości ruchu w połączeniach ruchowych,

J – macierz Jacobiego o wymiarach $m \times n$, powstała przez zróżniczkowanie m -wymiarowej transformacji prostej – funkcji n współrzędnych konfiguracyjnych, $\partial f(\theta)/\partial \theta$.

Dla redundantnych łańcuchów kinematycznych $n > m$ i rozwiązanie równania (2), polegające na znalezieniu wektora $\dot{\theta}(t)$, nie jest możliwe na drodze szukania macierzy odwrotnej do macierzy J . W latach siedemdziesiątych, a przede wszystkim w latach osiemdziesiątych i dziewięćdziesiątych, budowano różne algorytmy do rozwiązania problemu sterowania redundantnymi łańcuchami kinematycznymi, które najczęściej testowano w systemach symulacji. Podstawowe z tych algorytmów to:

– pseudoinwersja Moore-Penrose [Whitney 1969] z odmianą ważonej pseudoinwersji [Liegeois 1977] [Vukobratovic, Kircanski 1984] i uogólnionej pseudoinwersji [Wang 95],

– metoda tłumionych najmniejszych kwadratów (ang. *damped least squer*) [Wampler 1986] [Nakamura, Hanafusa 1986] [Seraji, Colbaugh 1990],

– metoda rozszerzonego jakobianu (ang. *extended Jacobian*) [Baillieul 1985],

– sterowanie konfiguracją (ang. *configuration control*) [Seraji 1989] [Seraji, Colbaugh 1990],

– rozwiązanie w zamkniętej formie (ang. *closed form solution*) [Chang 1987] [Galicki 1991],

– metoda bazująca na logice rozmytej (*fuzzy logic*) [Palm, Rehfueß 1992],

– metody bazujące na sieciach neuronowych [Hsia, Mao 1993] [Wloka² 1992],

– kompaktowa metoda programowania kwadratowego (ang. *compact QP method*) [Cheng, Chen, Sun 1992] [Cheng, Chen, Wang, Sun 1993].

Wiele z wyliczonych metod zastosowano w celu zwiększenia możliwości, zdolności i zręczności robotów, np. do:

– minimalizacji prędkości ruchu w połączeniach ruchowych,

– minimalizacji całkowitej lub kinetycznej energii [Vukobratovic, Kircanski 1984] [Hollerbach, Suh 1987],

– minimalizacji stosunku prędkości ruchu w połączeniach ruchowych do prędkości członu roboczego (ang. *Manipulator Velocity Ratio*) [Dubey, Luh 1988],

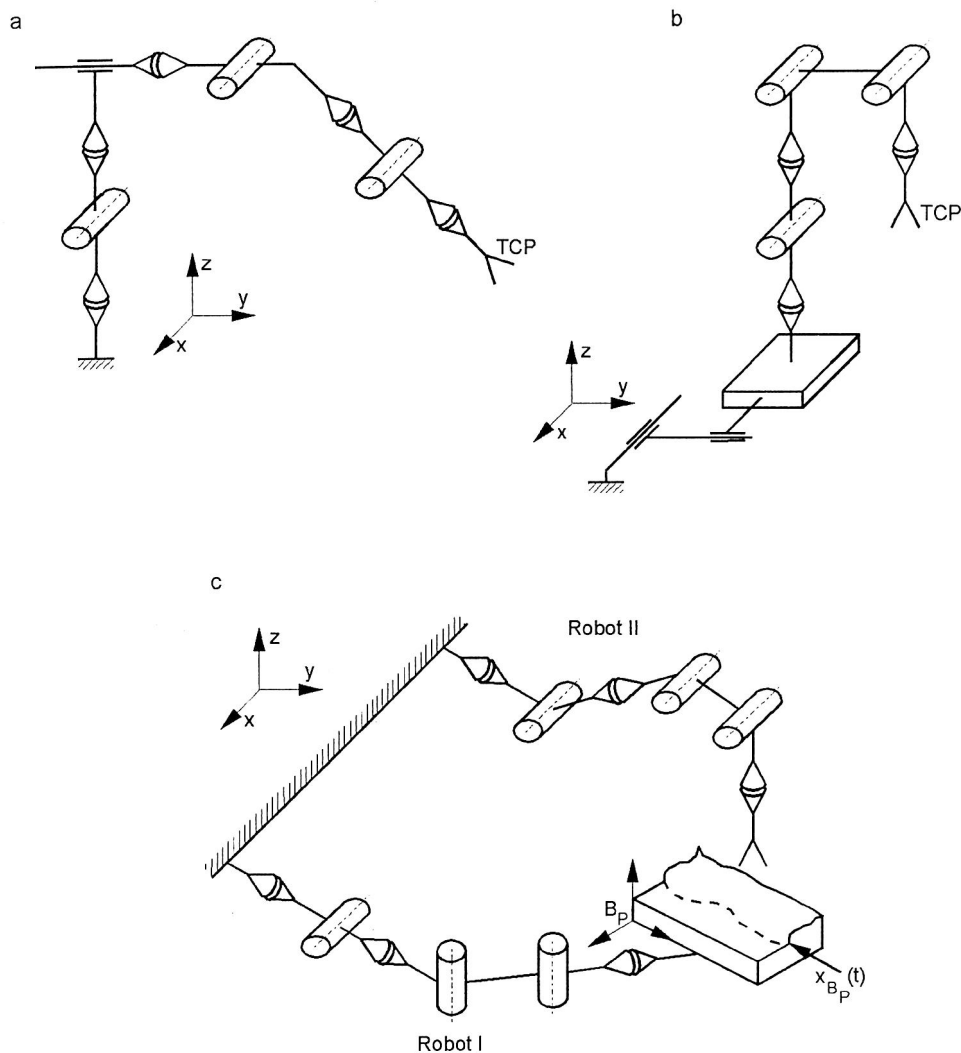
– minimalizacji sił/momentów w połączeniach ruchowych [Hollerbach, Suh 1987].

- minimalizacji stosunku sił/momentów w połączeniach ruchowych do sił/momentów na członie roboczym (ang. *Manipulator Mechanical Advantage*) [Dubey, Luh 1988] [Seraji 1992],
- utrzymania stałej podatności członu roboczego na zadanym torze ruchu [Seraji 1992],
- zmniejszenia obciążenia połączeń ruchowych, pochodzącego od impulsu siły przy wejściu członu roboczego w kontakt z przedmiotem [Seraji 1992],
- minimalizacji obciążenia połączeń ruchowych, pochodzącego od sił ciężkości [Seraji 1992],
- zwiększenia manipulacyjności (ang. *Manipulability*) [Yoshikawa 1985],
- zwiększenia dynamicznej manipulacyjności (ang. *Dynamic Manipulability*) [Yoshikawa 1985],
- zwiększenia odległości od najbliższej osobliwej konfiguracji ramion (zwiększanie wartości najmniejszej wartości osobliwej jakobianu) [Klein, Blaho 1987],
- utrzymania wychyleń połączeń ruchowych w określonych granicach [Liegeois 1977] [Pritschow, Koch 1991],
- omijania przeszkód [Maciejewski, Klein 1985] [Mitsi, Bouzakis 1993] [Cheng, Chen, Wang, Sun 1993],
- priorytetyzacji zadań [Nakamura, Hanafusa, Yoshikawa 1987] [Maciejewski, Klein 1985],
- zapewnienia powtarzalności ruchu przy zadaniach cyklicznych [Baker, Wampler 1988] [Roberts Maciejewski 1993] [Cheng, Chen, Wang, Sun 1993].

3.2. Kategorie redundantnych łańcuchów kinematycznych

Wymienione metody rozwiązywania problemu sterowania redundantnymi łańcuchami kinematycznymi oraz ich zastosowania w celu zwiększenia możliwości, zdolności i zręczności robotów, tworzą przede wszystkim dla szeregowego otwartego łańcucha kinematycznego. Nadają się one jednak także dla robotów mobilnych [Seraji 1993] i kooperujących łańcuchów kinematycznych w wersji dwóch robotów [Duelen, Kirchhoff, Held, Münch 1987] [Münch 1991] [Pritschow, Koch 1991] [Koch 1991] [Koch 1992], czy w wersji robota i stołu do mocowania obrabianego przedmiotu [Andersen, Cook, Zein-Sabattou, Fernandez 1989] [Cook, Andersen, Zein-Sabattou, Fernandez 1989] (rys. 3.1).

Roboty mobilne mogą występować w czterech podstawowych formach: robot umocowany na platformie poruszającej się na szynie w jednej osi, robot portalowy, robot na platformie poruszającej się swobodnie na kołach i robot umocowany na innym makrorobocie [Seraji 1993]. Chociaż formy te różnią się zasadniczo swoim zewnętrznym wyglądem, to przy jednakowym traktowaniu stopni swobody platformy i manipulatora opis matematyczny takich łańcuchów kinematycznych można sformułować w ten sam sposób. Robot mobilny może być traktowany jako kombinacja dwóch podsystemów



Rys. 3.1. Trzy podstawowe kategorie redundantnych łańcuchów kinematycznych: szeregowy otwarty łańcuch kinematyczny (a), mobilny robot (b), kooperujące roboty (c)

Fig. 3.1. Three main categories of redundant kinematics: sequential open kinematic chain (a), mobile robot (b), cooperating robots (c)

mów: mobilnej platformy o n_a stopniach swobody i manipulatora o n_b stopniach swobody. Zadaną trajektorię członu roboczego robota mobilnego $\mathbf{x}(t)$, wyrażoną we współrzędnych kartezjańskich, uzyskuje się ze złożenia ruchu połączeń ruchowych platformy $\boldsymbol{\theta}_a$ i połączeń ruchowych manipulatora $\boldsymbol{\theta}_b$ wg równania transformacji prostej

$$\mathbf{x}(t) = f_{a+b}(\boldsymbol{\theta}_a(t), \boldsymbol{\theta}_b(t)).$$

Po zróżniczkowaniu równania względem czasu

$$\dot{\mathbf{x}} = [\mathbf{J}_a(\boldsymbol{\theta}_a) \mathbf{J}_b(\boldsymbol{\theta}_b)] \begin{bmatrix} \dot{\boldsymbol{\theta}}_a(t) \\ \dot{\boldsymbol{\theta}}_b(t) \end{bmatrix}$$

uzyskuje się postać równoważną równaniu (2), która stanowi punkt wyjściowy dla lokalnych metod generowania trajektorii ruchu [Seraji 1993]. Na macierz Jacobiego $\mathbf{J}$ składają się przy tym dwie części: część związana z platformą $\mathbf{J}_a$ i część związana z manipulatorem $\mathbf{J}_b$, a na wektor współrzędnych konfiguracyjnych $\boldsymbol{\theta}$: współrzędne konfiguracyjne platformy $\boldsymbol{\theta}_a$ i manipulatora $\boldsymbol{\theta}_b$.

Z istoty macierzy Jacobiego, której kolumny wyrażają się następującym wzorem [Vukobratovic, Kircansky 1986]

$$\mathbf{j}_i = {}^0\mathbf{e}_i \times {}^0\mathbf{R}_{TCP,i}$$

gdzie ${}^0\mathbf{e}_i$ – jednostkowy wektor osi i -tego połączenia ruchowego opisany w bazowym układzie współrzędnych,

${}^0\mathbf{R}_{TCP,i}$ – wektor o początku w przegubie i , a końcu w punkcie pracy członu roboczego (*Tool Centre Point – TCP*), wyrażony w bazowym układzie współrzędnych,

wynika, że macierz $\mathbf{J}_b$ jest dokładnie macierzą Jacobiego manipulatora $\mathbf{J}_{\text{manip}}$, a macierz $\mathbf{J}_a$ macierzą Jacobiego platformy, ale wyznaczoną dla punktu *TCP* manipulatora.

Na ruch robota mobilnego nałożone są zwykle dodatkowe warunki, których przykładem może być poruszanie się platformy po wyznaczonym torze ruchu $\mathbf{x}_{\text{wózek}}(t)$, podczas realizacji zadania podstawowego, określonego dla członu roboczego całego robota mobilnego torem ruchu $\mathbf{x}(t)$. Warunek ten można wyrazić za pomocą funkcji wielowymiarowej

$$\Phi(\boldsymbol{\theta}_a(t)) = f_{\text{wózek}}(\boldsymbol{\theta}_a(t)) - \mathbf{x}_{\text{wózek}}(t),$$

gdzie $f_{\text{wózek}}(\boldsymbol{\theta}_a)$ – transformacja prosta platformy.

Warunek ten tworzy razem z równaniem transformacji prostej dla robota mobilnego następujący układ równań:

$$\mathbf{x}(t) = f_{a+b}(\boldsymbol{\theta}_a(t), \boldsymbol{\theta}_a(t)),$$

$$0 = \Phi(\boldsymbol{\theta}_a(t))$$

który po zróżniczkowaniu przyjmuje nadal formę równania (2)

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_a(\boldsymbol{\theta}_a) & \mathbf{J}_b(\boldsymbol{\theta}_b) \\ \mathbf{J}_{\text{wózek}}(\boldsymbol{\theta}_a) - \dot{\mathbf{x}}_{\text{wózek}}(t) & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\theta}}_a(t) \\ \dot{\boldsymbol{\theta}}_b(t) \end{bmatrix}.$$

Podobnie matematyczny opis łańcucha kinematycznego kooperujących robotów daje się sprowadzić do postaci równania (2). Na rysunku 3.1c przedstawiono kinematyczny model kooperujących robotów, dla których zadanie kooperacji polega na obróbce przedmiotu wzdłuż zadanej na nim trajektorii przez robota oznaczonego numerem II, podczas gdy robot I trzyma przedmiot. Trajektorja $\mathbf{x}_{Bp}(t)$ została przy tym określona w lokalnym układzie współrzędnych przedmiotu Bp . Zadanie kooperacji można sformułować, przyjmując za bazowy układ współrzędnych właśnie lokalny układ przedmiotu Bp i traktując łańcuchy kinematyczne obu robotów jako jeden szeregowy łańcuch kinematyczny, rozpoczynający się od członu roboczego robota I, poprzez jego stopę, następnie stopę robota II i kończąc na jego członie roboczym. Zadana trajektorię członu roboczego takiego łańcucha $\mathbf{x}_{Bp}(t)$, wyrażoną w układzie współrzędnych kartezjańskich przedmiotu Bp , uzyskuje się wtedy ze złożenia ruchu połączeń ruchowych obu robotów wg równania transformacji prostej

$$\mathbf{x}_{Bp}(t) = f_{I+II}(\boldsymbol{\theta}_I(t), \boldsymbol{\theta}_{II}(t))$$

Po zróżniczkowaniu równania

$$\dot{\mathbf{x}}_{Bp}(t) = [\mathbf{J}_I(\boldsymbol{\theta}_I) \mathbf{J}_{II}(\boldsymbol{\theta}_{II})] \begin{bmatrix} \dot{\boldsymbol{\theta}}_I(t) \\ \dot{\boldsymbol{\theta}}_{II}(t) \end{bmatrix}$$

uzyskuje się, podobnie jak w przypadku robota mobilnego, postać równoważną równaniu (2).

W zadaniach kooperacji dwóch robotów występują zwykle dodatkowe więzy, które daje się wyrazić za pomocą równań kinematyki jednego z robotów. Przykładem może być warunek, spotykany podczas spawania, aby końcówka technologiczna robota, pokrywająca się z normalną do powierzchni przedmiotu, na którym wyznaczono trajektorię, była skierowana pionowo w dół. W tym przypadku dwie współrzędne wektora $\mathbf{x}_{II} = f_{II}(\boldsymbol{\theta}_{II})$ (w globalnym układzie współrzędnych), np. piąta $x_{II,5}$ i szósta $x_{II,6}$, określające orientację członu roboczego robota II, powinny przyjmować pewną stałą wartość (wartość ta zależy od definicji kątów orientujących). Warunek ten można zapisać za pomocą wielowymiarowej funkcji

$$\Phi(\boldsymbol{\theta}_{II}(t)) = \begin{bmatrix} f_{II,5}(\boldsymbol{\theta}_{II}(t)) - x_{II,5} \\ f_{II,6}(\boldsymbol{\theta}_{II}(t)) - x_{II,6} \end{bmatrix} = 0.$$

Warunek ten, wraz z równaniem transformacji prostej, tworzy następujący układ:

$$\mathbf{x}_{Bp}(t) = f_{I+II}(\boldsymbol{\theta}_I(t), \boldsymbol{\theta}_{II}(t)),$$

$$0 = \Phi(\boldsymbol{\theta}_{II}(t)),$$

który po zróżniczkowaniu przyjmuje nadal formę równania (2):

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_I(\theta_I) & \mathbf{J}_{II}(\theta_{II}) \\ 0 & \mathbf{J}_{II,5,6}(\theta_{II}) \end{bmatrix} \begin{bmatrix} \dot{\theta}_I(t) \\ \dot{\theta}_{II}(t) \end{bmatrix}.$$

gdzie $\mathbf{J}_{II,5,6}$ jest piątym i szóstym wierszem macierzy Jacobiego robota II.

Zadanie dla dwóch kooperujących robotów może być też określone inaczej, np. wielowymiarową funkcją wiążącą ruchy członu roboczego

$$k(\mathbf{x}_I(t), \mathbf{x}_{II}(t)) = k(t).$$

Po zróżniczkowaniu tego równania

$$\frac{\partial k}{\partial \mathbf{x}_I} \frac{\partial \mathbf{x}_I}{\partial \theta_I} \dot{\theta}_I + \frac{\partial k}{\partial \mathbf{x}_{II}} \frac{\partial \mathbf{x}_{II}}{\partial \theta_{II}} \dot{\theta}_{II} = \dot{k}$$

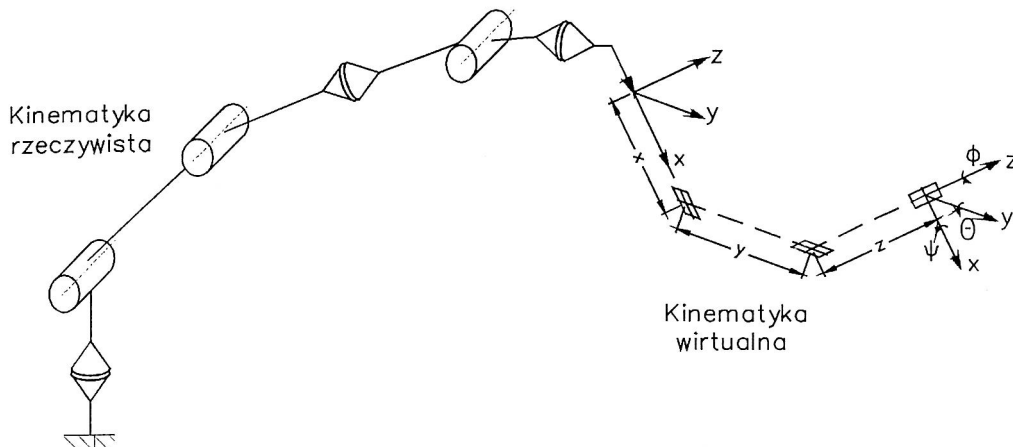
i przedstawieniu go w formie macierzowej

$$\begin{bmatrix} \frac{\partial k}{\partial \mathbf{x}_I} & \frac{\partial k}{\partial \mathbf{x}_{II}} \end{bmatrix} \begin{bmatrix} \mathbf{J}_I & 0 \\ 0 & \mathbf{J}_{II} \end{bmatrix} \begin{bmatrix} \dot{\theta}_I \\ \dot{\theta}_{II} \end{bmatrix} = [\dot{k}]$$

uzyskuje się także postać równoważną równaniu (2) [Koch 1992].

Przedstawione tu trzy kategorie redundantnych łańcuchów kinematycznych: szeregowego otwartego łańcucha kinematycznego, mobilnego robota i kooperujących robotów, przy czysto kinematycznie sformułowanym zadaniu, nie wyczerpują oczywiście wszystkich kategorii redundantnych łańcuchów kinematycznych i problemów ich sterowania. Do innych kategorii redundantnych łańcuchów kinematycznych można zaliczyć np.: zamknięte układy kinematyczne, maszyny kroczące [Morecki, Knapczyk 1993] [Nakamura 1991], koordynacja ruchu robotów z uwzględnieniem rozkładu sił lub obciążenia [Yoshikawa, Zheng 1993] [Shin, Chung 1993] [Choi, Lee, Ko 1993] [Luecke, Gardner 1993], czy też sterowanie ruchem obejmującym przedmioty przez hiperredundantny robot [Tsuji, Nakayama, Araki, Ito 1995]. Tych kategorii zadań w pracy nie rozpatrywano.

Wymienione algorytmy sterowania redundantnymi łańcuchami kinematycznymi, których przegląd znajduje się w dalszej części rozdziału 3, można zastosować jeszcze do rozwiązania tzw. wirtualnej redundancji, która może odnosić się do wszystkich trzech podstawowych kategorii łańcuchów kinematycznych. Chodzi tu o zdefiniowane dopuszczalne odchyłki od nominalnego toru ruchu członu roboczego, jak i o częściowe lub pełne symetrie w układzie: obrabiany przez robota przedmiot – końcówka technologiczna robota. Tolerancje te, określające warunki brzegowe zadania, modelowane są za pomocą wirtualnej redundancji [Duelen, Kirchoff, Held, Münch 1987] [Münch 1991], w ogólnym przypadku o sześciu stopniach swobody (rys. 3.2). Obszary tolerancji modelowane są przy tym jako obszary wychyleń wirtualnych połączeń ruchowych.



Rys. 3.2. Całkowity łańcuch kinematyczny robota składający się z części rzeczywistej i wirtualnej [Münch 1991]

Fig. 3.2. Total robot kinematics consisting of both real and virtual kinematics [Münch 1991]

3.3. Pseudoinwersja

Jednym z pierwszych i bardzo często proponowanych rozwiązań układu równań (2) jest rozwiązanie poprzez tzw. pseudoinwersję

$$\dot{\theta} = J^+ \dot{x}. \quad (3)$$

Macierz J^+ jest pseudoinwersją (zwaną też pseudoinwersją Moore'a-Penrose'a) macierzy J , jeśli spełnia następujące 4 równania [Nakamura 1991]:

$$J J^+ J = J, \quad J^+ J J^+ = J^+, \quad (J J^+)^T = J J^+, \quad (J^+ J)^T = J^+ J.$$

Pseudoinwersję można znaleźć na drodze tzw. rozkładu na wartości osobliwe (ang. *Singular Value Decomposition* – SVD) [Klema, Laub 1980], który rozkłada macierz na iloczyn trzech macierzy

$$J = U \Sigma V^T. \quad (4)$$

Macierze te służą do otrzymania pseudoinwersji

$$J^+ = V \Sigma^+ U^T, \quad (5)$$

przy czym:

$$\Sigma = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}_{m \times n}$$

$$\Sigma^+ = \begin{bmatrix} \mathbf{S}^+ & 0 \\ 0 & 0 \end{bmatrix}_{n \times n}$$

gdzie $\mathbf{S} = \text{diag}(\sigma_1, \dots, \sigma_r)$, $\mathbf{S}^+ = \text{diag}(1/\sigma_1, \dots, 1/\sigma_r)$, $r \leq m$ (r jest rzędem macierzy $\mathbf{J}$). Wartości σ_i są tzw. wartościami osobliwymi macierzy $\mathbf{J}$. Macierze $\mathbf{U}$ i $\mathbf{V}$ są macierzami unitarnymi (tzn. $\mathbf{U}\mathbf{U}^T = \mathbf{I}_m$ i $\mathbf{V}\mathbf{V}^T = \mathbf{I}_n$) o wymiarach odpowiednio $m \times m$ i $n \times n$. Do obliczania rozkładu na wartości osobliwe istnieją gotowe bardzo efektywne programy biblioteczne [Eispack 1977] [Koch 1992].

Jeśli wszystkie m wartości osobliwe jacobianu $\mathbf{J}(\theta)$ są większe od zera ($r = m$, tzn. rząd macierzy $\mathbf{J}$ jest równy m), oznacza to, że konfiguracja ramion, dla których ten jacobian został obliczony, nie jest osobliwa. Macierze $\mathbf{S}$ i $\mathbf{S}^+$ mają wówczas rząd równy także m , a macierze Σ , Σ^+ przyjmują następującą formę:

$$\Sigma = \begin{bmatrix} \mathbf{S} & 0 \end{bmatrix}_{m \times n}$$

$$\Sigma^+ = \begin{bmatrix} \mathbf{S}^+ \\ 0 \end{bmatrix}_{n \times m}$$

Gdy rząd macierzy $\mathbf{J}$ jest równy m , wtedy również $(\mathbf{J}\mathbf{J}^T)^{-1}$ istnieje i pseudoinwersję można obliczyć ze wzoru

$$\mathbf{J}^+ = \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1}. \quad (6)$$

Do obliczeń metodą rozkładu na wartości osobliwe tworzone są także hardwarowe rozwiązania w technologii VLSI [Walker, Cavallaro 1993], a także algorytmy i programy dla współbieżnych obliczeń na kilku procesorach [Maciejewski, Reagin 1992].

Rozwiązanie równania (2) poprzez pseudoinwersję jest rozwiązaniem, które minimalizuje kryterium w postaci sumy kwadratów prędkości ruchu w połączeniach ruchowych (kwadrat normy euklidesowej wektora prędkości), czyli w postaci formy kwadratowej $K = \dot{\theta}^T \dot{\theta}$ [Whitney 1969]. Kryterium takie wydaje się pożądane, jako że podczas ruchu robota przyczynia się do znajdowania takich konfiguracji, które wymagają minimalnych prędkości ruchu w połączeniach ruchowych (w sensie normy euklidesowej).

3.4. Ważona pseudoinwersja

Jeśli rozwiązanie równania (2) ma minimalizować kryterium w postaci ważonych iloczynów prędkości ruchu w połączeniach ruchowych $T = \dot{\theta}^T \mathbf{W} \dot{\theta}$, gdzie $\mathbf{W}$ jest symetryczną dodatnio określoną macierzą wag, to otrzymuje się je za pomocą tzw. ważonej pseudoinwersji

$$\dot{\theta} = \mathbf{J}_w^+ \dot{x}. \quad (7)$$

Rozwiązanie to można sprowadzić do poszukiwania rozwiązania przez zwykłą pseudoinwersję, dokonując podstawienia $\dot{\boldsymbol{\theta}} = \mathbf{W}^{-1/2}\dot{\mathbf{v}}$. Po tym podstawieniu minimalizowane kryterium przyjmuje postać $K = \dot{\mathbf{v}}^T \dot{\mathbf{v}}$, a zamiast równania (2) rozwiązać należy równanie $\dot{\mathbf{x}} = \mathbf{J}\mathbf{W}^{-1/2}\dot{\mathbf{v}}$.

Jeśli $\det(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T) \neq 0$, to ważoną pseudoinwersję można obliczyć z równania

$$\mathbf{J}_W^+ = \mathbf{W}^{-1} \mathbf{J}^T (\mathbf{J} \mathbf{W}^{-1} \mathbf{J}^T)^{-1} \quad (8)$$

Jeśli za macierz $\mathbf{W}$ zostanie podstawiona macierz bezwładności robota, to rozwiązanie metodą ważonej pseudoinwersji minimalizuje energię kinetyczną ruchu robota.

Ważona pseudoinwersja może być zastosowana także do omijania przeszkód [Vukobratovic, Kircansky 1986] [Münch 1991]. Macierz wag $\mathbf{W}$ wyraża się wtedy następująco

$$\mathbf{W} = \text{diag}(w_{11}, \dots, w_{nn}),$$

gdzie

$$w_{ii} = \begin{cases} \frac{d_{\max} - d_{\min}}{d(\mathbf{x}_k, H) - d_{\min}} & \text{dla } i = 1, \dots, l \\ 1 & \text{dla } i = l+1, \dots, n \end{cases} \quad (9)$$

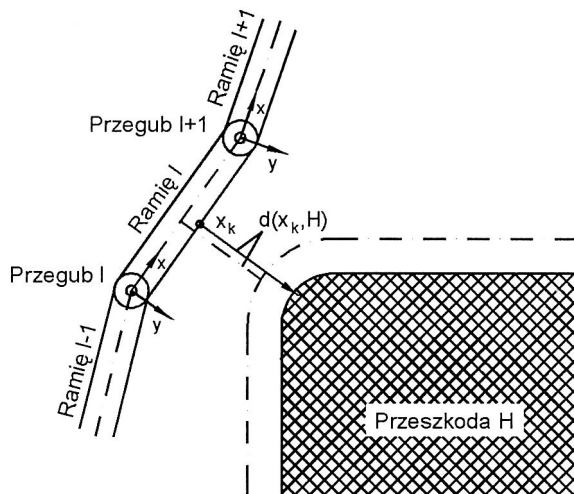
przy czym połączenie ruchowe l jest połączeniem poprzedzającym położenie na robocie krytycznego (najbliższego przeszkodzie) punktu $\mathbf{x}_k$ (rys. 3.3), a $d_{\min}$ i $d_{\max}$ określają obszar przestrzeni pokrywający przeszkodę, w którym poruszanie się robota oznacza włączenie proponowanej strategii omijania przeszkody.

Zbliżenie się punktu krytycznego do przeszkody na odległość $d_{\min}$ powoduje wzrost pierwszych l wag do nieskończoności, co w rezultacie preferuje ruch przede wszystkim połączeń ruchowych bliższych członowi roboczemu, począwszy od $l+1$. Problem znalezienia punktu krytycznego $\mathbf{x}_k$ jest osobnym problemem, który można rozwiązać przez zastosowanie sensorów lub innych metod (zob. rozdział 4.2.4).

Innym zastosowaniem ważonej pseudoinwersji jest utrzymywanie wychyleń w połączeniach ruchowych w ich granicach [Chan, Dubey 1993]. Skonstruowana w tym celu macierz wag ma postać

$$\mathbf{W} = \text{diag}(w_{11}, \dots, w_{nn}),$$

gdzie



Rys. 3.3. Położenie najbliższego przeszkodzie punktu krytycznego x_k na robocie [Münch 1991]

Fig. 3.3. Position of the critical point x_k on the robot which is closest to the obstacle [Münch 1991]

$$w_{ii} = \begin{cases} 1 + \left| \frac{\partial H(\theta)}{\partial \theta_i} \right| & \text{jeśli } \Delta \left| \frac{\partial H(\theta)}{\partial \theta_i} \right| \geq 0 \\ 1 & \text{jeśli } \Delta \left| \frac{\partial H(\theta)}{\partial \theta_i} \right| < 0 \end{cases} \quad (10)$$

$$H(\theta) = \sum_{i=1}^n \frac{1}{4} \frac{(\theta_{i,\max} - \theta_{i,\min})^2}{(\theta_{i,\max} - \theta_i)(\theta_i - \theta_{i,\min})}$$

Wartość funkcji $H(\theta)$ zdąży do nieskończoności przy zbliżaniu się wychylenia w którymkolwiek połączeniu ruchowym θ_i do którejkolwiek z granic wychyleń $\theta_{i,\min}$ lub $\theta_{i,\max}$. Pochodna $\partial H / \partial \theta_i$ jest równa zero, jeśli wychylenie w i -tym połączeniu ruchowym znajduje się pośrodku obszaru wychyleń i również wzrasta do nieskończoności przy zbliżaniu się wychylenia do którejkolwiek z granic. Waga w_{ii} przyjmuje natomiast duże wartości jedynie wówczas, gdy wychylenie w i -tym połączeniu ruchowym przesuwa się w kierunku którejkolwiek z granic wychyleń, przyjmując wartość 1 w pozostałych przypadkach.

3.5. Wykorzystanie jądra macierzy Jacobiego (część jednorodna rozwiązania)

Rozwiązanie poprzez pseudoinwersję, dane wzorem (3), może zostać uzupełnione tzw. jednorodną częścią rozwiązania

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^+ \dot{\mathbf{x}} + \alpha (\mathbf{I} - \mathbf{J}^+ \mathbf{J}) \mathbf{z}, \quad (11)$$

gdzie $\mathbf{z}$ jest dowolnym m -wymiarowym wektorem. Przekształcenie w postaci macierzy $(\mathbf{I} - \mathbf{J}^+ \mathbf{J})$ wykonuje rzut wektora $\mathbf{z}$ w tzw. jądro macierzy Jacobiego $\mathbf{J}$. Własność tę można zapisać równaniem

$$\mathbf{J} (\mathbf{I} - \mathbf{J}^+ \mathbf{J}) \mathbf{z} = \mathbf{0} \quad \text{dla dowolnego } \mathbf{z}.$$

Przekształcenie $(\mathbf{I} - \mathbf{J}^+ \mathbf{J})$ można obliczyć korzystając z rozkładu macierzy Jacobiego $\mathbf{J}$ metodą rozkładu na wartości osobiwe (zob. równania (4) i (5))

$$(\mathbf{I} - \mathbf{J}^+ \mathbf{J}) = \mathbf{V}_2 \mathbf{V}_2^T, \quad (12)$$

gdzie na $\mathbf{V}_2$ składa się $n - r$ dolnych wierszy macierzy $\mathbf{V}$ z rozkładu macierzy metodą *SVD*

$$\mathbf{J} = \mathbf{U} \Sigma \mathbf{V} = [\mathbf{U}_1 \mathbf{U}_2] \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix}.$$

Bardziej efektywny numerycznie algorytm znalezienia rozwiązania równania (2), zawierającego obie części dane wzorem (11), polega na rozwiązaniu dla zadanego wektora $\mathbf{z}$ następującego układu równań [Klein, Huang 1983]:

$$\mathbf{J} \mathbf{J}^T \mathbf{a} = \dot{\mathbf{x}} - \mathbf{J} \mathbf{z},$$

a następnie podstawieniu rozwiązania $\mathbf{a}$ do wzoru na $\dot{\boldsymbol{\theta}}$

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^T \mathbf{a} + \mathbf{z}.$$

Jednorodna część rozwiązania stosowana jest przez wielu badaczy do optymalizowania ruchu redundantnego łańcucha kinematycznego wg kryteriów sformułowanych jako funkcje potencjału. W tym celu w miejsce wektora $\mathbf{z}$ w równaniu (11) wstawiony zostaje gradient potencjału $\nabla H(\boldsymbol{\theta})$. Potencjał ten może pełnić rolę miary „dobroci” konfiguracji robota. Niektóre przykładowe funkcje potencjału opisano poniżej.

Stosunek prędkości ruchu w połączeniach ruchowych do prędkości członu roboczego (ang. Manipulator Velocity Ratio) [Dubey, Luh 1988]:

$$H = r_v = \|\dot{\mathbf{x}}\|_{\mathbf{W}_x} / \|\dot{\boldsymbol{\theta}}\|_{\mathbf{W}_\theta} \quad (13)$$

gdzie $\|\cdot\|_{\mathbf{W}}$ jest pierwiastkiem z formy kwadratowej z macierzą wag $\mathbf{W}$: $\|\mathbf{a}\|_{\mathbf{W}} = (\mathbf{a}^T \mathbf{W} \mathbf{a})^{1/2}$. Zastosowanie zamiast wektora $\mathbf{z}$ gradientu tego potencjału zapewnia, że robot podczas wykonywania swojego zadania przyjmuje takie konfiguracje ramion, które dla zadanej

prędkości członu roboczego wymagają minimalnych, w sensie normy $\|\cdot\|_W$, prędkości ruchu w połączeniach ruchowych.

Stosunek sił/momentów w połączeniach ruchowych do sił/momentów na członie roboczym (ang. Manipulator Mechanical Advantage) [Dubey, Luh 1988]:

$$H = r_m = \|\mathbf{F}\|_{W_F} / \|\mathbf{T}\|_{W_T}, \quad (14)$$

gdzie $\mathbf{F}$ – wektor sił/momentów na członie roboczym,

$\mathbf{T}$ – wektor sił/momentów w połączeniach ruchowych.

Zależność między obu wektorami wyraża się znaną zależnością za pomocą macierzy Jacobiego: $\mathbf{T} = \mathbf{J}^T \mathbf{F}$. Zastosowanie zamiast wektora $\mathbf{z}$, gradientu tego potencjału, zapewnia, że robot podczas wykonywania swojego zadania przyjmuje takie konfiguracje ramion, które umożliwiają mu realizację wymaganych sił/momentów na członie roboczym za pomocą minimalnych, w sensie normy $\|\cdot\|_W$, sił/momentów w połączeniach ruchowych.

Utrzymanie wychyleń w połączeniach ruchowych w ich granicach
wg [Liegois 1977]

$$H = \frac{1}{n} \sum_{i=1}^{i=n} \left(\frac{\theta_i - a_i}{\delta_i / 2} \right)^2, \quad (15)$$

wg [Pritschow, Koch 1991]

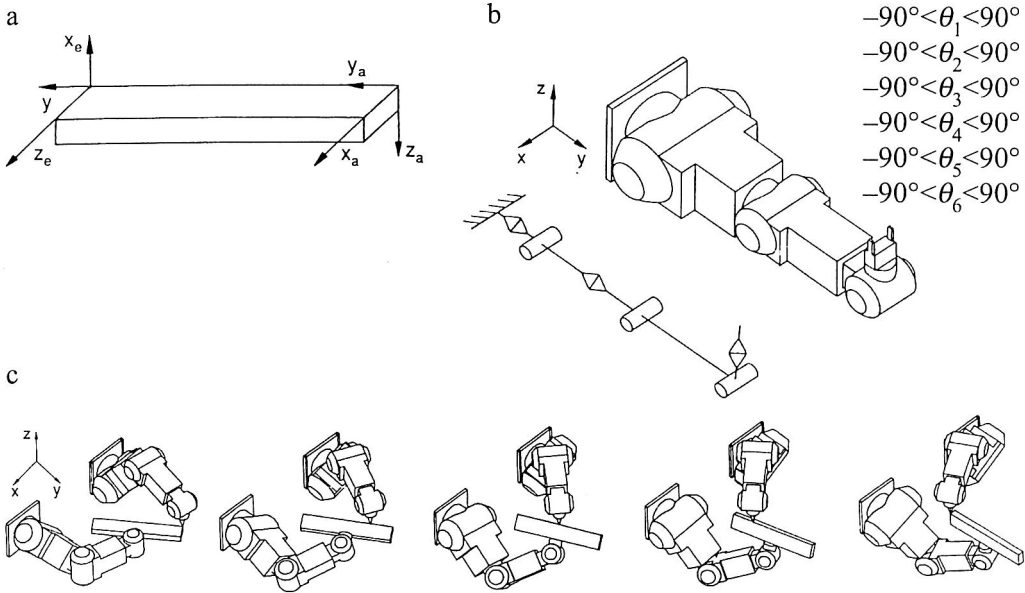
$$H = \sum_{i=1}^{i=n} \frac{1}{1 - \left(\frac{\theta_i - a_i}{\delta_i / 2} \right)^2}, \quad (16)$$

gdzie a_i – środek obszaru wychyleń w i -tym połączeniu ruchowym

$$(a_i = (\theta_{i,\max} - \theta_{i,\min})/2),$$

$$\delta_i - \text{zakres obszaru wychyleń } (\delta_i = \theta_{i,\max} - \theta_{i,\min}).$$

Wartości obu potencjałów wzrastają, gdy wychylenia w połączeniach ruchowych zbliżają się do granic. Pożądane jest więc minimalizowanie tych potencjałów, co wymaga ujemnego współczynnika α w równaniu (11). Różnica między tymi potencjałami polega na tym, że potencjał dany równaniem (16) wzrasta do nieskończoności, gdy wychylenia w połączeniach ruchowych zbliżają się do granic wychyleń. Przez zastosowanie tego potencjału uzyskuje się więc dalsze zwiększenie priorytetu oddalania się od granic wychyleń w połączeniach ruchowych, które są najbliższe tych granic. Zastosowanie potencjału z równania (16) dla kooperujących robotów (rys. 3.4), rozpatrywanych jako redundantny łańcuch kinematyczny o 12 stopniach swobody, ilustruje rysunek 3.5.



Rys. 3.4. Przedmiot, którego krawędź tworzy liniowo interpolowaną trajektorię ruchu narzędzia (zaznaczono początkowe i końcowe położenie lokalnego układu współrzędnych narzędzia) (a), schemat kinematyczny kooperujących robotów i granice wychyleń w połączeniach ruchowych (b), poszczególne fazy koordynacji ruchu (c) [Pritschow, Koch 1991]

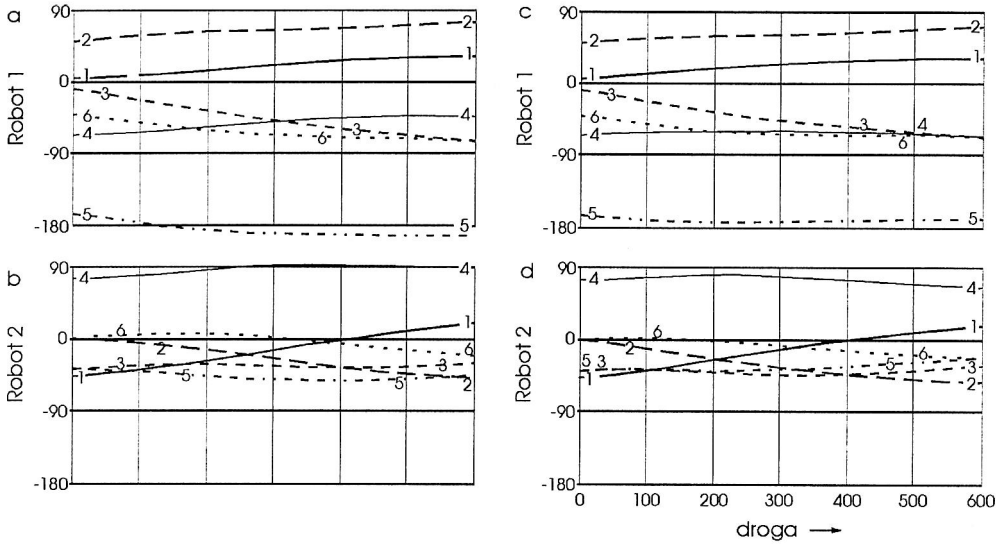
Fig. 3.4. Workpiece edge as a linear interpolated path for the tool (both start and end frames of the tool are indicated) (a), kinematics of both cooperating robots and specification of joints limits (b), stages of motion coordination (c) [Pritschow, Koch 1991]

Podczas ruchu, wg algorytmu przez pseudoinwersję danego równaniem (3), połączenie ruchowe nr 5 robota trzymającego przedmiot (rys. 3.5a) i połączenie ruchowe nr 4 robota trzymającego narzędzie (rys. 3.5 b), przekroczyły granice wychyleń (patrz określenie granic na rys. 3.4 b). Po dodaniu do algorytmu, jak w równaniu (11), składnika jednorodnego z wektorem z jako gradientem potencjału określonego równaniem (16), wychylenia w obydwu połączeniach ruchowych utrzymały się w zadanych granicach (patrz rys. 3.5c i d).

Manipulacyjność (ang. Manipulability) [Yoshikawa 1985]:

$$H = w = \sqrt{\det(\mathbf{J}\mathbf{J}^T)}. \quad (17)$$

Wskaźnik w znany jest jako miara manipulacyjności łańcucha kinematycznego i wskazuje na odległość aktualnej konfiguracji od konfiguracji osobliwej. Dla konfiguracji osobliwej $w = 0$. Zastosowanie tego potencjału powoduje przyjmowanie przez



Rys. 3.5. Przebiegi wychyleń [w stopniach] w połączeniach ruchowych o numerach od 1 do 6 kooperujących robotów podczas ruchu na trajektorii z rys. 3.4; robot trzymający przedmiot (a); robot trzymający narzędzie (b); (a) i (b) wg rozwiązania poprzez pseudoinwersję; (c) i (d) odpowiednio, ale po dodaniu do rozwiązania poprzez pseudoinwersję składnika jednorodnego z gradientem potencjału dla utrzymania wychyleń w połączeniach ruchowych w ich granicach [Pritschow, Koch 1991]

Fig. 3.5. Motion [in deg] of the joints 1–6 of the cooperating robots while moving along the path from fig. 3.4; robot holding the workpiece (a); robot holding the tool (b); (a) and (b) calculated using pseudoinverse; (c) and (d) respectively, after addition to the pseudoinverse solution of the homogeneous part with the gradient of potential for keeping the joints within their limits [Pritschow, Koch 1991]

robot takich konfiguracji, które są możliwie najbardziej „ruchowo zręczne” (w sensie wskaźnika manipulacyjności).

Dynamiczna manipulacyjność (ang. Dynamic Manipulability) [Yoshikawa 1985]:

$$H = w_d = \sqrt{\det[\mathbf{J}(\mathbf{M}^T \mathbf{M})^{-1} \mathbf{J}^T]}, \quad (18)$$

gdzie $\mathbf{M}$ – macierz bezwładności robota dla aktualnej konfiguracji.

Dynamiczną manipulacyjność należy interpretować jako miarę zdolności zmiany przyspieszenia członu roboczego robota dla zadanych sił/momentów w połączeniach ruchowych.

Omijanie przeszkód:

$$H = \sum_{i=1}^{i=l} \frac{1}{C_o(p_i) - 1}, \quad (19)$$

gdzie $C_o(p_i) = 1$ – równanie definiujące obrys przeszkody (ang. *contour*),

p_i – punkty na manipulatorze, które jako pierwsze mogą zetknąć się z przeszkodą,

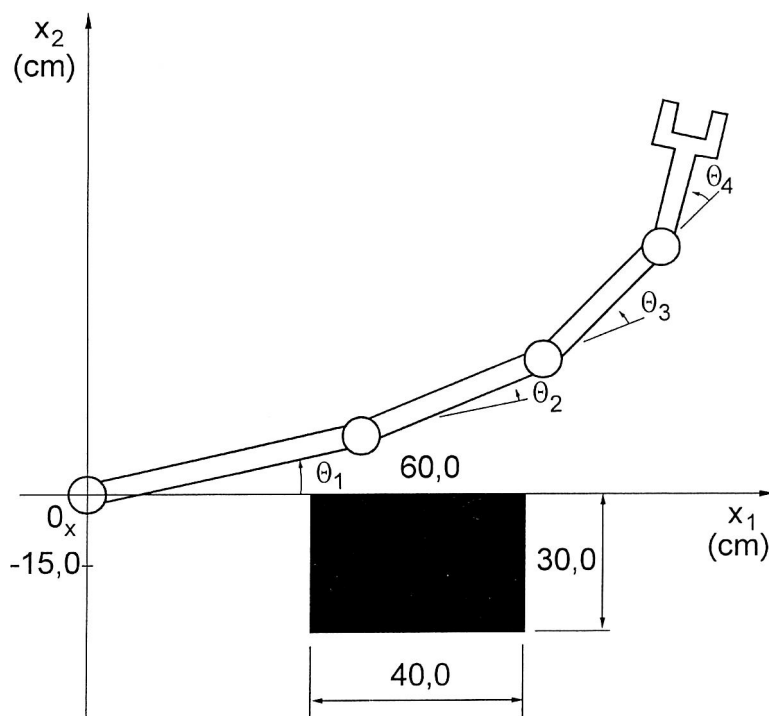
l – liczba tych punktów.

Ponieważ tak zdefiniowany potencjał wzrasta do nieskończoności przy zbliżaniu się robota do przeszkody, pożądany jest ruch, który by go minimalizował. Wymaga to ujemnego współczynnika α w równaniu (11).

Przykładowo, prostokąt o wymiarach 40×30 , którego środek znajduje się w punkcie $(60, -15)$ względem bazowego układu współrzędnych, będący przeszkodą dla płaskiego robota (rys. 3.6), można opisać funkcją [Nakamura, Hanafusa, Yoshikawa 1987]

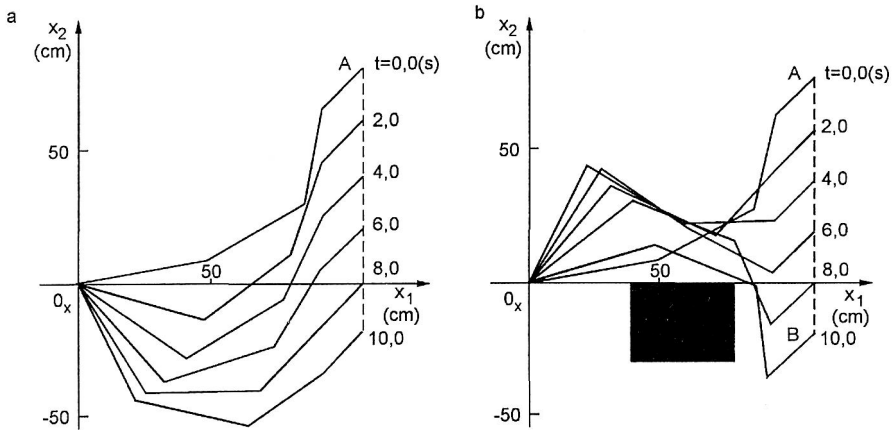
$$C_o(p_i) = \left(\frac{x_i - 60}{20} \right)^8 + \left(\frac{y_i + 15}{15} \right)^8.$$

Wyniki symulacji algorytmu korzystającego z przeformułowanego w dziedzinę przyspieszeń równania (11), z zastosowaniem gradientu potencjału określonego równaniem (19), pokazano na rysunku 3.7.



Rys. 3.6. Płaski robot o 4 połączeniach ruchowych i przeszkoda [Nakamura, Hanafusa, Yoshikawa 1987]

Fig. 3.6. Planar 4-degrees of freedom manipulator and an obstacle [Nakamura, Hanafusa, Yoshikawa 1987]



Rys. 3.7. Poszczególne fazy symulacji ruchu robota z rys. 3.6 na trajektorii wyznaczonej odcinkiem AB : bez przeszkody (a), z przeszkodą (b) [Nakamura, Hanafusa, Yoshikawa 1987]

Fig. 3.7. Simulation of the robot motion (robot from fig. 3.6) on the path along the line segment AB : without obstacle (a), with obstacle (b) [Nakamura, Hanafusa, Yoshikawa 1987]

Zamiast minimalizować potencjał H z równania (19), który wzrasta do nieskończoności wraz ze zbliżaniem się wybranych punktów robota do przeszkody, można także w celu uniknięcia kolizji maksymalizować pole powierzchni pomiędzy ramionami robota a odcinkami wyznaczonymi przez wybrane punkty na przeszkodzie i połączenia ruchowe robota (rys. 3.8) [Mayorga, Ma, Wong, Ressa 1993]

$$H = \sum V_i,$$

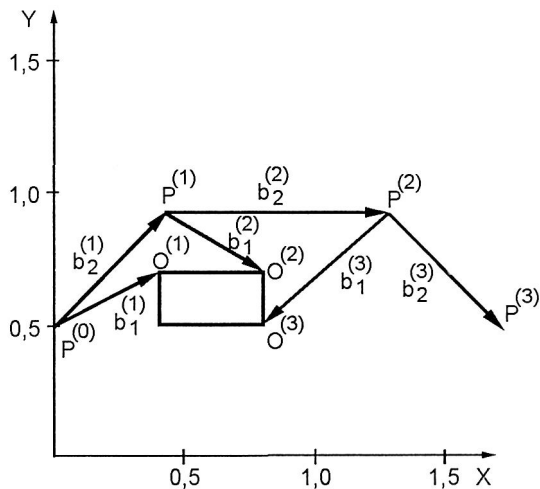
gdzie

$$V_i = |\mathbf{b}_1^{(i)} \times \mathbf{b}_2^{(i)}|.$$

Aby ominąć przeszkody, można także zastosować inny sposób, np. nie wyznaczać punktów na manipulatorze lub przeszkodzie, które jako pierwsze mogą wejść w kolizję (co może okazać się czasami zadaniem zbyt trudnym), lecz pokryć cały robot i przeszkody sferami (rys. 3.9) [Mitsi, Bouzakis 1993]. Funkcja C_o przyjmie wtedy postać

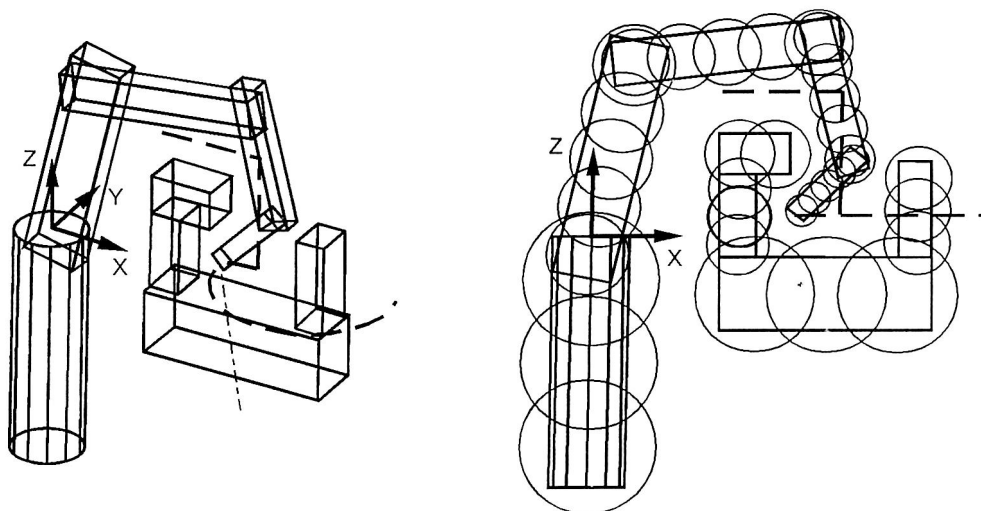
$$C_o = \sum_{j=1}^{j=p} [(x_i - x_j) / r_{ji}]^2 + [(y_i - y_j) / r_{ji}]^2 + [(z_i - z_j) / r_{ji}]^2,$$

gdzie p – liczba punktów (środków sfer) na przeszkodzie,
 (x_i, y_i, z_i) – współrzędne środków sfer na robocie,
 (x_j, y_j, z_j) – współrzędne środków sfer na przeszkodzie,
 $r_{ji} = r_i + r_j$ – r_i jest długością promienia sfery i -tej na robocie, a r_j jest długością promienia sfery j -tej na przeszkodzie.



Rys. 3.8. Wyznaczanie pola powierzchni pomiędzy robotem a przeszkodą
[Mayorga, Ma, Wong, Ressa 1993]

Fig. 3.8. Area determined between robot arms and obstacle [Mayorga, Ma, Wong, Ressa 1993]



Rys. 3.9. Pokrycie robota i przeszkody sferami celem wyznaczenia funkcji potencjału dla unikania kolizji [Mitsi, Bouzakis 93]

Fig. 3.9. Covering of robots and obstacle with spheres to calculate the potential function for obstacle avoidance [Mitsi, Bouzakis 93]

Priorytetyzacja zadań

Gradienty potencjałów, których przykłady zaprezentowano, mogą występować zamiast wektora $\mathbf{z}$ w równaniu (11) pojedynczo lub jako liniowa kombinacja kilku z nich [Nakamura, Hanafusa, Yoshikawa 1987] [Pritschow, Koch 1991]:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^+ \dot{\mathbf{x}} + \alpha (\mathbf{I} - \mathbf{J}^+ \mathbf{J}) (\alpha_1 \nabla H_1 + \dots + \alpha_k \nabla H_k). \quad (20)$$

Stałe wartości α_i odzwierciedlają kompromis między poszczególnymi celami (potencjałami).

Inny sposób polega na uwzględnieniu wielu zadań wg z góry przyjętych priorytetów. Zadanie o niższym priorytecie jest realizowane wówczas, gdy nie ogranicza to realizacji zadania o wyższym priorytecie [Maciejewski, Klein 1985] [Nakamura, Hanafusa, Yoshikawa 1987]. W przypadku dwóch zadań rozwiązanie tak sformułowanego zadania priorytetyzacji przyjmuje postać

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_1^+ \dot{\mathbf{x}}_1 + (\mathbf{I} - \mathbf{J}_1^+ \mathbf{J}_1) [\hat{\mathbf{J}}_2^+ (\dot{\mathbf{x}}_2 - \mathbf{J}_2 \mathbf{J}_1^+ \dot{\mathbf{x}}_1) + (\mathbf{I} - \hat{\mathbf{J}}_2^+ \hat{\mathbf{J}}_2) \mathbf{z}], \quad (21)$$

gdzie: $\hat{\mathbf{J}}_2 = \mathbf{J}_2 (\mathbf{I} - \mathbf{J}_1^+ \mathbf{J}_1)$,

$\dot{\mathbf{x}}_1 = \mathbf{J}_1^+ \dot{\boldsymbol{\theta}}$ – zadanie o niższym priorytecie,

$\dot{\mathbf{x}}_2 = \mathbf{J}_2 \dot{\boldsymbol{\theta}}$ – zadanie o wyższym priorytecie.

Udowodniono, że czynnik $(\mathbf{I} - \mathbf{J}_1^+ \mathbf{J}_1)$ w równaniu (21) może zostać opuszczony [Maciejewski, Klein 1985].

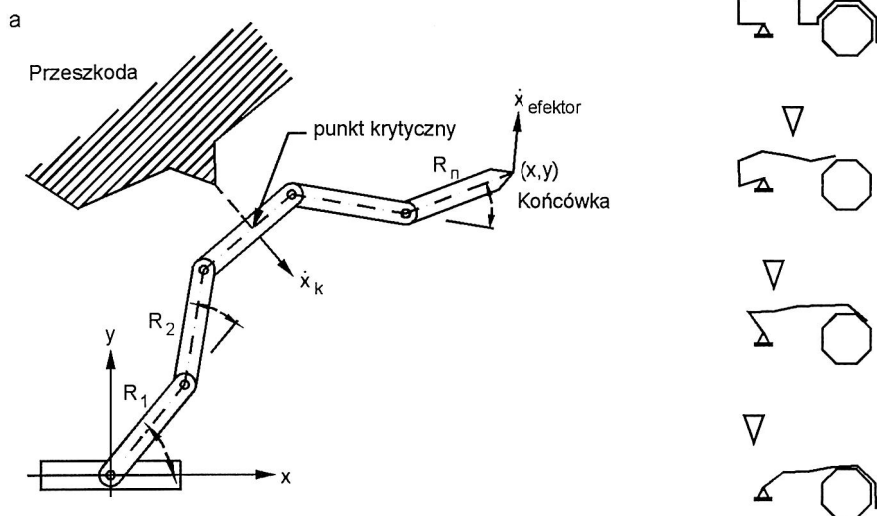
Wektor $\mathbf{z}$ w równaniu (21) może zostać użyty do realizacji następnego zadania, ale już o trzeciorzędnym priorytecie.

Ten sposób priorytetyzacji stosowano z powodzeniem do zadania omijania przeszkód [Nakamura, Hanafusa, Yoshikawa 1987] [Maciejewski, Klein 1985] [Risse, Fink, Hiller 1995]. Zadanie prowadzenia członu roboczego wzdłuż toru ruchu, zaznaczone na rys. 3.10 za pomocą $\dot{\mathbf{x}}_{\text{efektor}}$, było przy tym zadaniem o wyższym priorytecie. Zadanie o niższym priorytecie stanowiło omijanie przeszkody opisane przez wektor prędkości $\dot{\mathbf{x}}_k$, który oddala położony najbliżej przeszkody punkt na robocie.

3.6. Inne formy rozwiązań za pomocą pseudoinwersji

W niektórych zadaniach rozwiązanie równania (2) wiąże się z minimalizacją kryterium, które niekoniecznie przyjmuje postać samej jedynie formy kwadratowej, tak jak to było w poprzednich dwóch rozdziałach. Przykładem może tu być kryterium przyrostu zużycia energii na wykonanie ruchu w połączeniach ruchowych w przedziale czasu Δt , które przyjmuje postać [Vukobratovic, Kircanski 1984]

$$K = \Delta E = \frac{1}{2} \dot{\boldsymbol{\theta}}^T \mathbf{B} \dot{\boldsymbol{\theta}} + \mathbf{b}^T \dot{\boldsymbol{\theta}}, \quad (22)$$



Rys. 3.10. Płaski manipulator ilustrujący zadanie o wyższym priorytecie – zadany wektor prędkości członu roboczego $\dot{x}_{\text{efektor}}$ i zadanie o niższym priorytecie – wektor prędkości $\dot{x}_k$, oddalający manipulator od przeszkody (a); wynik symulacji rozwiązania wg równania (21), ilustrujący omijanie poruszającej się trójkątnej przeszkody podczas prowadzenia członu roboczego po zadanym torze (b) [Maciejewski, Klein 1985]

Fig. 3.10. Planar manipulator with both first-priority subtask specified by end-effector velocity vector $\dot{x}_{\text{efektor}}$ and second-priority subtask specified by velocity vector $\dot{x}_k$ moving the robot away from the obstacle (a); simulation results calculated according to equation (21) showing the avoidance of the moving triangle obstacle while moving the robot end-effector on the given path (b) [Maciejewski, Klein 1985]

przy czym macierz $\mathbf{B}$ i wektor $\mathbf{b}$ muszą zostać obliczone z dynamiki manipulatora i jego napędów. Optymalne rozwiązanie równania (2) przy tym kryterium przyjmuje postać (jeśli $\det(\mathbf{J}\mathbf{B}^{-1}\mathbf{J}^T) \neq 0$)

$$\dot{\theta} = \mathbf{G} \dot{\mathbf{x}} + (\mathbf{G}\mathbf{J} - \mathbf{I}) \mathbf{B}^{-1} \mathbf{b}, \quad (23)$$

gdzie $\mathbf{G} = \mathbf{B}^{-1} \mathbf{J}^T (\mathbf{J}\mathbf{B}^{-1}\mathbf{J}^T)^{-1}$.

Rozwiązanie w tej formie może być także zastosowane [Vukobratovic, Kircanski 1986] do ulepszenia algorytmu omijania przeszkód, przedstawionego już w rozdziale 3.4. W przedstawionym tam algorytmie macierz wag powoduje przyhamowanie, we wszystkich kierunkach, ruchu pierwszych l połączeń ruchowych poprzedzających położenie punktu krytycznego na robocie. W tej sytuacji położenie punktu krytycznego blisko członu roboczego mogłoby spowodować uniemożliwienie wykonania zadania podstawowego. Bardziej celowe byłoby, aby punkt krytyczny $\mathbf{x}_k$ poruszał się z pewną

prędkością $\mathbf{v}_c$ oddalającą go w kierunku prostopadłym od przeszkody. Zdefiniowana w równaniu (9) macierz wag pozwala na rozłożenie kryterium optymalizującego na część związaną z pierwszymi l połączeniami ruchowymi θ_L i pozostałymi θ_U aż po człon roboczy robota

$$K = \dot{\boldsymbol{\theta}}^T \mathbf{W} \dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_L^T \mathbf{W} \dot{\boldsymbol{\theta}}_L + \dot{\boldsymbol{\theta}}_U^T \mathbf{I} \dot{\boldsymbol{\theta}}_U.$$

Po dokonaniu podstawień: $\mathbf{W} = \mathbf{J}_k^T \mathbf{W}_k \mathbf{J}_k$ i $\dot{\mathbf{x}}_k = \mathbf{J}_k \dot{\boldsymbol{\theta}}_k$, gdzie $\mathbf{J}_k$ jest macierzą Jacobiego obliczoną dla punktu krytycznego x_k , można przedstawić kryterium K następująco

$$K = \dot{\mathbf{x}}_k^T \mathbf{W}_k \dot{\mathbf{x}}_k + \dot{\boldsymbol{\theta}}_U^T \mathbf{I} \dot{\boldsymbol{\theta}}_U.$$

Po wprowadzeniu wektora prędkości $\mathbf{v}_c$, który ma optymalny kierunek oddalania się punktu x_k od przeszkody, otrzymujemy ulepszoną wersję kryterium

$$K = (\dot{\mathbf{x}}_k - \mathbf{v}_c)^T \mathbf{W}_k (\dot{\mathbf{x}}_k - \mathbf{v}_c) + \dot{\boldsymbol{\theta}}_U^T \mathbf{I} \dot{\boldsymbol{\theta}}_U.$$

Po wymnożeniach i odrzuceniu stałych składników, które nie mają wpływu na optymalizację, rozważane kryterium przyjmuje końcową postać

$$K = \dot{\boldsymbol{\theta}}^T \mathbf{W} \dot{\boldsymbol{\theta}} + \mathbf{b}^T \dot{\boldsymbol{\theta}},$$

gdzie $\mathbf{b} = [-\mathbf{v}_c^T \mathbf{W}_k \mathbf{J}_k, 0, \dots, 0]^T$.

Rozwiązanie optymalne przyjmuje więc identyczną formę z kryterium minimalizacji zużycia energii daną równaniem (23).

3.7. Metoda tłumionych najmniejszych kwadratów odporna na konfiguracje osobliwe

W dotychczas przedstawionych metodach zadanie formułowano następująco: należy znaleźć taki wektor prędkości konfiguracji $\dot{\boldsymbol{\theta}}$, który spełniając równanie (2) optymalizuje zadane kryterium. Tak sformułowane zadanie nie zapewnia w ogólności ominięcia konfiguracji osobliwych robota, odznaczających się tym, że już w ich pobliżu występują znaczne prędkości ruchu w połączeniach ruchowych $\dot{\boldsymbol{\theta}}$. Istnieją jednak metody lokalne, które zapewniają omijanie konfiguracji osobliwych, choć może to spowodować błąd w dokładności prowadzenia członu roboczego. Przykładem takiej metody jest metoda tłumionych najmniejszych kwadratów (ang. *damped least squer*) [Wampler 1986], znana też pod nazwą metody odpornej na konfiguracje osobliwe (ang. *singularity robust inverse*) [Nakamura, Hanafusa 1986] [Nakamura 1991]. Sformułowanie zadania w tej metodzie polega na minimalizacji kryterium, które jest kompromisem między dokładnością rozwiązania a możliwością jego wykonania, tak aby prędkości ruchu w połączeniach ruchowych nie były zbyt duże

$$\min \{ \|\mathbf{J}\dot{\boldsymbol{\theta}} - \dot{\mathbf{x}}\|^2 + k^2 \|\dot{\boldsymbol{\theta}}\|^2 \}. \quad (24)$$

Jedyne rozwiązanie, minimalizujące to kryterium, ma postać [Wampler 1986]

$$\dot{\mathbf{\theta}} = (\mathbf{J}^T \mathbf{J} + k^2 \mathbf{I})^{-1} \dot{\mathbf{x}}. \quad (25)$$

Czynnik równoważący k może być dobierany w sposób adaptacyjny w zależności od bliskości aktualnej konfiguracji od konfiguracji osobiwej, co pozwala realizować dokładnie zadany ruch, nawet bardzo blisko takiej konfiguracji [Nakamura, Hanafusa 1986] [Nakamura 1991].

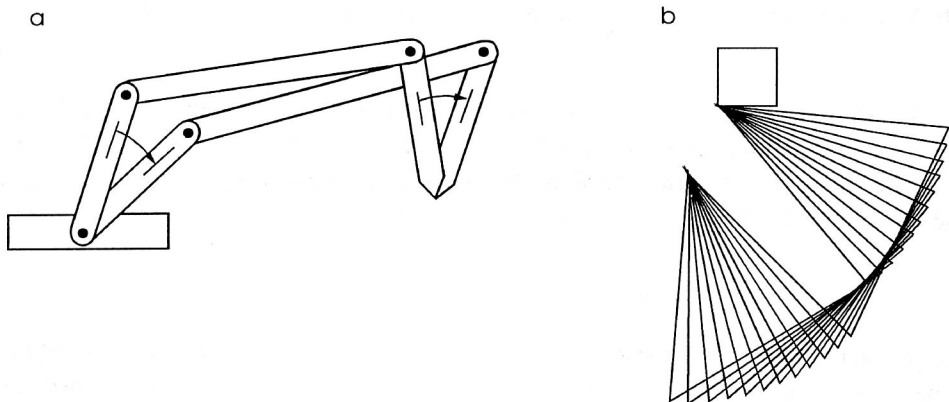
3.8. Powtarzalność ruchu robota

Z problemem tworzenia ruchu redundantnego łańcucha kinematycznego wiąże się nierozłącznie dość istotny problem powtarzalności tego ruchu. Powtarzalność ruchu najłatwiej wyjaśnić na przykładzie cyklicznego zadania wielokrotnego prowadzenia członu roboczego po zamkniętej trajektorii. Jeżeli podczas każdego powtórzenia robot przyjmuje te same konfiguracje ramion, w szczególności ma tę samą konfigurację na starcie każdego powtórzenia, to ruch taki i algorytm jego sterowania nazywamy powtarzalnym (ang. *repeatable control*). Sterowanie niepowtarzalne może w zasadzie uniemożliwić testowanie ruchu robota, np. w celu sprawdzenia czy nie zachodzi kolizja lub czy wychylenia w połączeniach ruchowych są utrzymywane w granicach wychyleń. Sterowanie powtarzalne ma następującą właściwość: blisko leżące trajektorie w przestrzeni kartezjańskiej odpowiadają blisko leżącym trajektoriom we współrzędnych konfiguracyjnych.

Wiele algorytmów, w tym również w ogólności rozwiązanie przez pseudoinwersję, to algorytmy niepowtarzalne. Na rysunku 3.11 pokazano zastosowanie rozwiązania poprzez pseudoinwersję dla cyklicznego zadania wielokrotnego prowadzenia członu roboczego po zamkniętej trajektorii w kształcie kwadratu [Klein, Huang 1983]. Za pomocą metod topologicznych można wykazać, że obszary, w których sterowanie jest powtarzalne, są ograniczone i że powtarzalny algorytm odpowiada transformacji odwrotnej [Baker, Wampler 1988]. Istnieje również metoda sprawdzenia, czy rozwiązanie wg pseudoinwersji w określonej przestrzeni współrzędnych konfiguracyjnych jest powtarzalne [Shamir, Yomdin 1988]. Niestety, metoda ta jest intensywna obliczeniowo.

Wiele algorytmów z natury swej są powtarzalne, a inne pozwalają się zmodyfikować tak, aby były powtarzalne. Przykładami tych metod, które zostaną w następnych podrozdziałach krótko opisane, są:

- metoda rozszerzonego jakobianu [Baillieul 1985],
- sterowanie konfiguracją [Seraji 1989] [Colbaugh, Seraji, Glass 1989] [Seraji 1992],
- rozwiązanie w tzw. zamkniętej formie [Chang 1987] [Galicki 1991].



Rys. 3.11. Płaski trójprzegubowy robot (a), pokazany na torze ruchu w kształcie kwadratu, przy sterowaniu wg pseudoinwersji podczas przechodzenia przez ten sam wierzchołek w kolejnych cyklach (b) [Klein, Huang 1983]

Fig. 3.11. Planar three joint robot (a) moving with the pseudoinverse control on a quadratic path shown while passing the same vertex in the successive cycles (b) [Klein, Huang 1983]

3.9. Metoda rozszerzonego jacobianu

Metoda rozszerzonego jacobianu (ang.: *Extended Jacobian* lub *Augmented Jacobian*) [Baillieul 1985] jest algorytmem powtarzalnym. Metoda ta wymaga zestawienia dodatkowych warunków na współrzędne konfiguracyjne w postaci układu $n-m$ równań więzów $h(\theta) = 0$. Razem z m równaniami określającymi tor ruchu członu roboczego $x(t)$ powstaje następujący układ n równań z n niewiadomymi współrzędnymi konfiguracyjnymi θ_i :

$$x = f(\theta)$$

$$0 = h(\theta).$$

Po zróżniczkowaniu układ ten ma postać

$$\begin{bmatrix} \dot{x} \\ 0 \end{bmatrix} = J_e \dot{\theta}, \quad (26)$$

gdzie

$$J_e = \begin{bmatrix} J \\ \frac{\partial h(\theta)}{\partial \theta} \end{bmatrix} = \begin{bmatrix} J \\ J_h(\theta) \end{bmatrix}.$$

Rozszerzony jacobian J_e jest macierzą kwadratową i jeśli jest odwracalny, to rozwiązanie układu równań (26) można znaleźć jako

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_e^{-1} \begin{bmatrix} \dot{\mathbf{x}} \\ 0 \end{bmatrix}.$$

Metoda rozszerzonego jacobianu może być zastosowana do optymalizacji kryteriów. W tym celu funkcję $h(\boldsymbol{\theta})$ należy skonstruować w następujący sposób [Seraji 1989]

$$h(\boldsymbol{\theta}) = \mathbf{J}^0 \nabla H(\boldsymbol{\theta}),$$

gdzie $H(\boldsymbol{\theta})$ – optymalizowane kryterium,

$\mathbf{J}^0$ – macierz o wymiarach $(n-m) \times n$, której wiersze tworzą przestrzeń jądra macierzy Jacobiego $\mathbf{J}$ (np. $\mathbf{J}^0$ może być utworzona z $n-m$ niezależnych wierszy macierzy $\mathbf{I} - \mathbf{J}^+ \mathbf{J}$).

Można dowieść, że warunkiem koniecznym, aby $\boldsymbol{\theta}^*$ było lokalnym ekstremum funkcji $H(\boldsymbol{\theta})$, jest to, że wektor gradientu $H(\boldsymbol{\theta}^*)$ nie należy do przestrzeni wyznaczonej przez jądro macierzy Jacobiego $\mathbf{J}$ [Baillieul 1985]. Warunek ten można matematycznie wyrazić równaniem: $\mathbf{J}^0 \nabla H(\boldsymbol{\theta}^*) = 0$ lub $(\mathbf{I} - \mathbf{J}^+ \mathbf{J}) \nabla H(\boldsymbol{\theta}^*) = 0$.

Przy tak przyjętej funkcji $h(\boldsymbol{\theta})$ macierz rozszerzonego jacobianu przyjmie postać:

$$\mathbf{J}_e = \begin{bmatrix} \mathbf{J} \\ \frac{\partial \mathbf{J}^0 \nabla H(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \mathbf{J} \\ \mathbf{J}_{\mathbf{J}^0 \nabla H(\boldsymbol{\theta})} \end{bmatrix}.$$

Aby znaleźć lokalne ekstremum funkcji $H(\boldsymbol{\theta})$ wystarczy znaleźć jakiegokolwiek rozwiązanie szczególne $\boldsymbol{\theta}_p$ i dokonać rzutu tego rozwiązania przez operator $\mathbf{P}$ [Chen, Walker 1993]

$$\boldsymbol{\theta}^* = \mathbf{P} \boldsymbol{\theta}_p,$$

gdzie

$$\mathbf{P} = \mathbf{I} - \mathbf{J}^0 (\mathbf{J}_{\mathbf{J}^0 \nabla H(\boldsymbol{\theta})} \mathbf{J}^0)^{-1} \mathbf{J}_{\mathbf{J}^0 \nabla H(\boldsymbol{\theta})}.$$

Metodę rozszerzonego jacobianu można zmodyfikować w taki sposób aby bardziej nadawała się do obliczeń numerycznych [Klein, Chu-Jeng, Ahmad 1993]

$$\mathbf{J}_e = \begin{bmatrix} \mathbf{J} \\ \mathbf{J}^0 \frac{\partial \nabla H(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \sum_{i=1}^r \mathbf{H}_i \mathbf{J}^+ \mathbf{Q}_i \end{bmatrix} = \begin{bmatrix} \mathbf{J} \\ \mathbf{J}^0 \mathbf{J}_{\nabla H(\boldsymbol{\theta})} - \sum_{i=1}^r \mathbf{H}_i \mathbf{J}^+ \mathbf{Q}_i \end{bmatrix},$$

gdzie $r = n - m$,

$\mathbf{Q}_i$ – macierz, której kolumna k jest równa: $[\partial \mathbf{J} / \partial \theta_k] \boldsymbol{\eta}_i$,

$\boldsymbol{\eta}_i$ – ortonormalna baza przestrzeni jądra macierzy Jacobiego $\mathbf{J}$,

$\mathbf{H}_i$ – macierz o wymiarach $r \times n$, której i -ty wiersz równy jest $\nabla^T H$, a pozostałe wiersze równe 0.

W ten sposób trudność w znalezieniu analitycznego wyrażenia na macierz $\mathbf{J}^0$, potrzebnego do obliczenia macierzy Jacobiego $\mathbf{J}_{j0 \nabla H(\theta)}$, została ominięta. Aby obliczyć $\partial \mathbf{J} / \partial \theta_k$ można zastosować dość prosty numerycznie algorytm [Klein, Chu-Jenq, Ahmed 1993].

Metoda rozszerzonego jakobianu, jako metoda powtarzalna, może posłużyć do poprawy rozwiązywania przez pseudoinwersję, tak aby rozwiązanie to stało się powtarzalne. Zaproponowany [Roberts, Maciejewski 1993] w tym celu sposób, wykorzystujący takie narzędzia matematyczne, jak przestrzeń Hilberta i szeregi Fouriera, polega na znalezieniu, w pierwszym kroku, rozwiązania równania (2) przez uogólnioną pseudoinwersję $\mathbf{G}$

$$\dot{\boldsymbol{\theta}} = \mathbf{G} \dot{\mathbf{x}},$$

które to rozwiązanie minimalizuje zadane kryterium w postaci formy kwadratowej. Z powodu braku powtarzalności tego rozwiązania szukane jest następnie rozwiązanie w postaci rozszerzonego jakobianu, które leżałoby „najbliżej” rozwiązania przez pseudoinwersję $\mathbf{G}$. Przykładowo, jeśli robot ma tylko jeden redundantny stopień swobody, poszukiwany jest taki wektor $\mathbf{v}$ dla rozszerzenia jakobianu

$$\begin{bmatrix} \dot{\mathbf{x}} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{J} \\ \mathbf{v}^T \end{bmatrix} \dot{\boldsymbol{\theta}},$$

który leży najbliżej jądra macierzy pseudoinwersji $\mathbf{G}$. W tym celu tworzy się podprzestrzeń dopuszczalnych, rozszerzających jakobian wektorów, rozpinając ją na ortonormalnej bazie wektorów $\mathbf{v}_j$, będących gradientami funkcji z szeregu Fouriera. Szukany wektor $\mathbf{v}$ można znaleźć jako ortogonalny rzut wektorów z przestrzeni jądra pseudoinwersji $\mathbf{G}$ na utworzoną przez bazę $\mathbf{v}_j$ podprzestrzeń.

3.10. Sterowanie konfiguracją

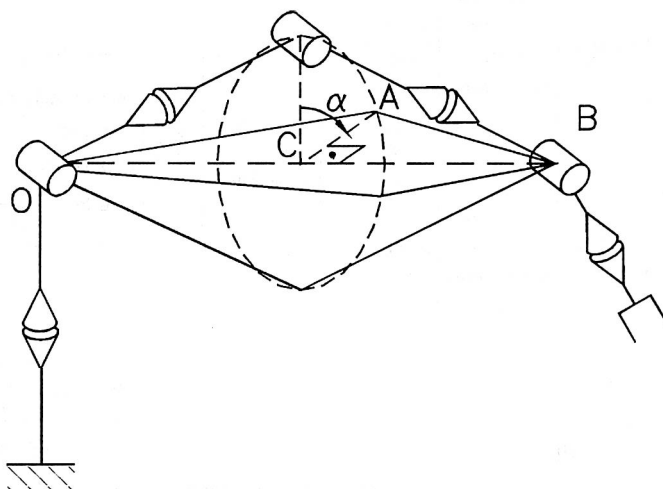
Metoda sterowania konfiguracją (ang. *configuration control*) [Seraji 1989] jest algorytmem powtarzalnym i wymaga, podobnie jak metoda rozszerzonego jakobianu, zestawienia dodatkowych warunków na współrzędne konfiguracyjne $\Phi(\boldsymbol{\theta})$, nazywanych więzami (*constraints*). Liczba więzów musi być równa $n-m$, dzięki czemu daje się zestawić układ n równań z n niewiadomymi θ_i

$$\mathbf{x}_e = \begin{bmatrix} \mathbf{x} \\ \Phi \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{\theta}) \\ \Phi(\boldsymbol{\theta}) \end{bmatrix}. \quad (27)$$

Tak opisany system nie jest już redundantny i może prosto realizować zadanie ruchu po trajektorii określonej w czasie

$$\mathbf{x}_e(t) = \begin{bmatrix} \mathbf{x}(t) \\ \Phi(t) \end{bmatrix}.$$

Prosty przykład siedmioprzegubowego robota, dla którego jako funkcję więzów wybrano pewien kąt α : $\Phi(\theta) = \alpha$, zamieszczono na rys. 3.12.



Rys. 3.12. Siedmioprzegubowy robot i definicja kąta α na potrzeby sterowania konfiguracją [Seraji 1989]

Fig. 3.12. Seven-joint robot and the definition of α angle for the purpose of configuration control [Seraji 1989]

Rozszerzenie systemu przez zdefiniowanie więzów pozbawia go redundantności, ale nie gwarantuje, że system nie wejdzie w konfigurację osobliwą. Same więzy w szczególności mogą taką osobliwość wprowadzić (nazywana jest ona wtedy osobliwością algorytmiczną).

Metoda sterowania konfiguracją może być także zastosowana do optymalizacji kryteriów. W tym celu funkcję więzów należy sformułować tak samo jak funkcję rozszerzającą jakobian (zob. rozdział 3.9)

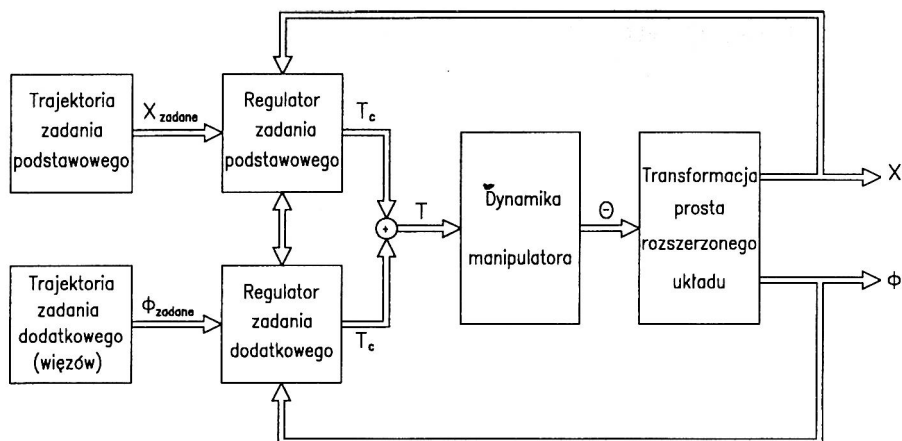
$$\Phi(\theta) = \mathbf{J}^0 \nabla H(\theta) = 0,$$

gdzie: $H(\theta)$ – optymalizowane kryterium,

$\mathbf{J}^0$ – macierz o wymiarach $(n-m) \times n$, której wiersze tworzą przestrzeń jądra macierzy Jacobiego $\mathbf{J}$.

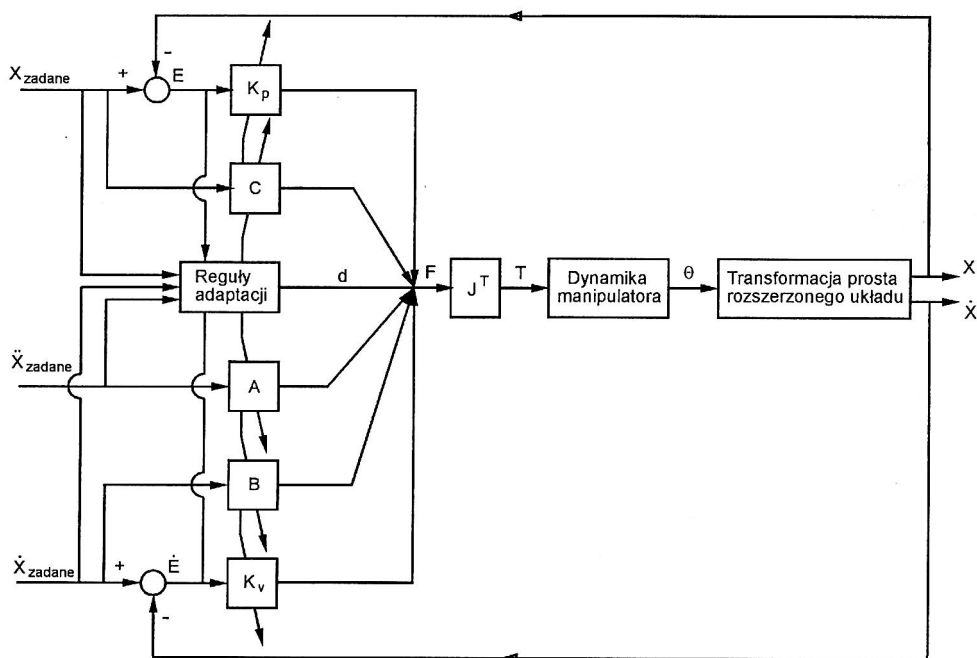
W systemie symulacji testowano metodę sterowania konfiguracją zastosowaną do omijania przeszkód [Colbaugh, Seraji, Glass 1989] oraz do optymalizacji takich kryteriów jak [Seraji 1992]: minimalizacja stosunku sił/momentów w połączeniach ruchomych do sił/momentów na członie roboczym (ang. *Manipulator Mechanical Advantage*), utrzymanie stałej podatności członu roboczego na zadanym torze ruchu, zmniejsz-

szanie obciążenia w połączeniach ruchowych pochodzącego od impulsu siły przy wejściu członu roboczego w kontakt z przedmiotem, minimalizacja obciążenia w połączeniach ruchowych, pochodzącego od sił ciężkości.



Rys. 3.13. Układ regulacji położenia do sterowania konfiguracją [Seraji 1989]

Fig. 3.13. Positioning control system for configuration control [Seraji 1989]



Rys. 3.14. Adaptacyjny układ regulacji położenia do sterowania konfiguracją [Seraji 1989]

Fig. 3.14. Adaptive positioning control system for configuration control [Seraji 1989]

Do sterowania konfiguracją stworzono układ regulacji w wersji kinematycznej [Seraji 1989] [Seraji 1990] (rys. 3.13) i wersji dynamicznej z adaptacją (rys. 3.14).

Równanie adaptacyjnego układu regulacji ma następującą postać:

$$\mathbf{T} = \mathbf{J}_e^T [\mathbf{d}(t) + K_p(t)\mathbf{E} + K_v(t)\dot{\mathbf{E}} + C(t)\mathbf{x}_{e,zadane} + B(t)\dot{\mathbf{x}}_{e,zadane} + A(t)\ddot{\mathbf{x}}_{e,zadane}]$$

gdzie $\mathbf{T}$ – wektor sił/momentów w połączeniach ruchowych, $\mathbf{E} = \mathbf{x}_{e,zadane} - \mathbf{x}_{e,rzeczywiste}$, a współczynniki $\mathbf{d}(t)$, $K_p(t)$, $K_v(t)$, $C(t)$, $B(t)$ i $A(t)$ wyznaczane są wg odpowiednich równań reguł adaptacyjnych [Seraji 89].

3.11. Rozwiązanie w zamkniętej formie

Rozwiązanie w tzw. zamkniętej formie (ang. *closed form solution*), które jest kolejnym algorytmem powtarzalnym, szuka wektora konfiguracji $\boldsymbol{\theta}$, który spełnia następujące warunki:

$$\begin{aligned} F(\boldsymbol{\theta}) &= f(\boldsymbol{\theta}) - \mathbf{x} = 0 \\ &\text{optimalizuje kryterium } H(\boldsymbol{\theta}). \end{aligned} \quad (28)$$

Tak sformułowane zadanie można rozwiązać za pomocą współczynników Lagrange'a [Chang 1987]. Polega to na znalezieniu wektora $\boldsymbol{\lambda}$ współczynników Lagrange'a λ_p , który spełnia równanie

$$\boldsymbol{\lambda}^T \frac{\partial F}{\partial \boldsymbol{\theta}} + \frac{\partial H}{\partial \boldsymbol{\theta}} = 0,$$

co można wyrazić z pomocą symbolu gradientu następująco

$$\boldsymbol{\lambda}^T \nabla F(\boldsymbol{\theta}) + \nabla H(\boldsymbol{\theta}) = 0.$$

Zadanie (28) można też rozwiązać metodą pochodnej Frecheta [Galicki 1989] [Galicki 1991]. W tym przypadku warunkiem koniecznym ekstremum funkcji $H(\boldsymbol{\theta})$ jest, by

$$\nabla H^T(\boldsymbol{\theta}) \delta \boldsymbol{\theta} = 0$$

z wektorem $\delta \boldsymbol{\theta}$ spełniającym

$$\mathbf{J} \delta \boldsymbol{\theta} = 0.$$

Obie metody prowadzą do zestawienia układu n równań z n niewiadomymi θ_i :

$$\begin{aligned} f(\boldsymbol{\theta}) - \mathbf{x} &= 0, \\ \mathbf{Z} \nabla H(\boldsymbol{\theta}) &= 0, \end{aligned} \quad (29)$$

gdzie $\mathbf{Z} = [(\mathbf{J}_m^{-1} \mathbf{J}_{n-m})^T, -\mathbf{I}_{n-m}]$, przy czym $\mathbf{J}_m$ i $\mathbf{J}_{n-m}$ składają się na macierz Jacobiego: $\mathbf{J} = [\mathbf{J}_m, \mathbf{J}_{n-m}]$. (We wzorze na $\mathbf{Z}$ przyjęto, że $\mathbf{J}_m$ jest odwracalna, tzn. że składa się z m niezależnych kolumn. Jeśli tak nie jest, należy zmienić permutację kolumn macierzy Jacobiego.) Można pokazać, że druga część układu równań (29) odpowiada rzutowi

gradientu $\nabla H(\theta)$ w przestrzeń jądra jacobianu $\mathbf{J}$ [Chang 1987]. Aby znaleźć wektor θ , układ równań (29) musi być rozwiązany numerycznie, co nie jest zadaniem nazbyt skomplikowanym.

Metodę rozwiązywania w tzw. zamkniętej formie można zastosować z powodzeniem do różnych zadań. Przykładem może tu być omijanie przeszkód. Wyniki symulacji omijania przeszkód według metody rozwiązywania w zamkniętej formie są podane w [Mitsi, Bouzakis 1993], gdzie robot i przeszkody pokryto sferami (zob. rys. 3.9 w rozdz. 3.5).

3.12. Kompaktowa metoda programowania kwadratowego

Do rozwiązywania problemu ruchu redundantnego łańcucha kinematycznego z powodzeniem zastosowano metodę programowania kwadratowego (ang. *quadratic programming* – *QP*) [Cheng, Chen, Sun 192] [Cheng, Chen, Wang, Sun 1993]. Metoda programowania kwadratowego rozwiązuje następująco sformułowany problem:

znajdź wektor α , który minimalizuje

$$\mathbf{g}^T \alpha + \frac{1}{2} \alpha^T \mathbf{H} \alpha$$

i spełnia równocześnie

$$\begin{aligned} \mathbf{A}_1 \alpha &= \mathbf{b}_1 \\ \mathbf{A}_2 \alpha &\geq \mathbf{b}_2 \end{aligned}$$

gdzie $\mathbf{b}_1$, $\mathbf{b}_2$ i $\mathbf{g}$ są zadanymi wektorami,

$\mathbf{A}_1$, $\mathbf{A}_2$ i $\mathbf{H}$ zadanymi macierzami, przy czym $\mathbf{H}$ jest dodatnio określona.

Metodę programowania kwadratowego można również zastosować do rozwiązania problemu ruchu redundantnych łańcuchów kinematycznych:

znajdź wektor prędkości przegubów $\dot{\theta}$, który minimalizuje

$$(a) \mathbf{g}^T \dot{\theta} + \frac{1}{2} \dot{\theta}^T \mathbf{H} \dot{\theta}, \quad (30)$$

i spełnia równocześnie

$$(b) \mathbf{J} \dot{\theta} = \dot{\mathbf{x}},$$

$$(c) \mathbf{A} \dot{\theta} \geq \mathbf{b},$$

przy czym forma kwadratowo-liniowa (30a) może służyć minimalizacji kryteriów, a nierówność (30c) ma zastosowanie do określenia maksymalnych prędkości ruchu w połączeniach ruchowych

$$\dot{\theta}_{i,\min} \leq \dot{\theta}_i \leq \dot{\theta}_{i,\max} \quad (31)$$

i granic ich wychyleń

$$\theta_{i,\min} \leq \theta_i \leq \theta_{i,\max}. \quad (32)$$

Macierz $\mathbf{A}$ i wektor $\mathbf{b}$ określa się po przekształceniu wpierw nierówności (32) w dziedzinę prędkości, wprowadzając takt interpolacji Δt :

$$\frac{\theta_{i,\min} - \theta_i}{\Delta t} \leq \dot{\theta}_i \leq \frac{\theta_{i,\max} - \theta_i}{\Delta t}, \quad (33)$$

a następnie po zintegrowaniu powstałej nierówności (33) i nierówności (31) w jedną

$$\max\left(\dot{\theta}_{i,\min}, \frac{\theta_{i,\min} - \theta_i}{\Delta t}\right) \leq \dot{\theta}_i \leq \min\left(\dot{\theta}_{i,\max}, \frac{\theta_{i,\max} - \theta_i}{\Delta t}\right). \quad (34)$$

Po rozłożeniu powstałej w ten sposób nierówność (34) na dwie:

$$\begin{aligned} \dot{\theta}_i &\geq \max\left(\dot{\theta}_{i,\min}, \frac{\theta_{i,\min} - \theta_i}{\Delta t}\right), \\ -\dot{\theta}_i &\geq -\min\left(\dot{\theta}_{i,\max}, \frac{\theta_{i,\max} - \theta_i}{\Delta t}\right) \end{aligned}$$

odnajdujemy $\mathbf{A}$ i $\mathbf{b}$

$$\mathbf{A} = \begin{bmatrix} I_n \\ -I_n \end{bmatrix}_{2n \times n} \quad (35)$$

$$\mathbf{b} = \begin{bmatrix} \max\left(\dot{\theta}_{1,\min}, \frac{\theta_{1,\min} - \theta_1}{\Delta t}\right) \\ \vdots \\ \max\left(\dot{\theta}_{n,\min}, \frac{\theta_{n,\min} - \theta_n}{\Delta t}\right) \\ -\min\left(\dot{\theta}_{1,\max}, \frac{\theta_{1,\max} - \theta_1}{\Delta t}\right) \\ \vdots \\ -\min\left(\dot{\theta}_{n,\max}, \frac{\theta_{n,\max} - \theta_n}{\Delta t}\right) \end{bmatrix}_{2n \times 1}$$

W przypadku gdy optymalizowana funkcja nie ma postaci formy kwadratowo-li-niowej, jak w równaniu (30a), i jest dowolną różniczkowalną funkcją nieliniową, za-

danie jej optymalizowania przy liniowych ograniczeniach (30b i c) jest problemem tzw. programowania nieliniowego. Do rozwiązania zadania programowania nieliniowego można zastosować metodę tzw. projekcji gradientu Rosena [Rosen 1957], która również nadaje się do sterowania redundantnymi łańcuchami kinematycznymi [Höfer 1992].

Przy rozwiązywaniu problemu programowania kwadratowego szybkość obliczeń zależy od wielkości zadania, tzn. od liczby zmiennych, liczby równań i liczby nierówności. W matematycznym opisie metody programowania kwadratowego, dla redundantnego łańcucha kinematycznego wg wzorów (30), liczba zmiennych jest równa n , liczba równań w układzie równań (30b) jest równa m , a liczba nierówności – biorąc pod uwagę równanie (35) – jest równa $2n$. Liczby te można zmniejszyć po przekształceniu postaci równania (30) na tzw. kompaktową metodę programowania kwadratowego (ang. *compact QP method*). Polega to na następującym ciągu przekształceń i podstawień:

Po pierwsze, równanie (30b) można przedstawić w następującej postaci (przy założeniu, że robot nie znajduje się w konfiguracji osobliwej):

$$\dot{\mathbf{x}} = [\mathbf{J}_m \mathbf{J}_{n-m}] \begin{bmatrix} \dot{\boldsymbol{\theta}}_m \\ \dot{\boldsymbol{\theta}}_{n-m} \end{bmatrix}, \quad (36)$$

gdzie: $\mathbf{J}_m$ – kwadratowa odwracalna macierz o wymiarach $m \times m$ (tzn., że $\mathbf{J}_m$ składa się z m niezależnych kolumn, a jeśli tak nie jest, należy zmienić permutację kolumn macierzy Jacobiego),

$\mathbf{J}_{n-m}$ – pozostałe $n-m$ kolumny macierzy Jacobiego,

$\dot{\boldsymbol{\theta}}_m$ – częściowy wektor prędkości ruchu w podstawowych połączeniach ruchowych, o wymiarze m ,

$\dot{\boldsymbol{\theta}}_{n-m}$ – częściowy wektor prędkości ruchu w połączeniach ruchowych redundantnych, o wymiarze $n-m$.

Układ równań (36) można rozwiązać następnie względem $\dot{\boldsymbol{\theta}}_m$

$$\dot{\boldsymbol{\theta}}_m = \mathbf{J}_m^{-1} \dot{\mathbf{x}} - \mathbf{J}_m^{-1} \mathbf{J}_{n-m} \dot{\boldsymbol{\theta}}_{n-m}$$

i określić na tej podstawie równanie przedstawiające cały wektor prędkości $\dot{\boldsymbol{\theta}}$ w funkcji częściowego wektora prędkości ruchu w połączeniach ruchowych redundantnych

$$\dot{\boldsymbol{\theta}} = \begin{bmatrix} \dot{\boldsymbol{\theta}}_m \\ \dot{\boldsymbol{\theta}}_{n-m} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_m^{-1} \dot{\mathbf{x}} \\ \mathbf{0}_{n-m} \end{bmatrix} + \begin{bmatrix} -\mathbf{J}_m^{-1} \mathbf{J}_{n-m} \\ \mathbf{I}_{n-m} \end{bmatrix} \dot{\boldsymbol{\theta}}_{n-m}.$$

Do równania tego można wprowadzić dla prostoty nowe oznaczenia wektora $\dot{\boldsymbol{\theta}}_p$ i macierzy $\mathbf{N}$:

$$\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_p + \mathbf{N} \dot{\boldsymbol{\theta}}_{n-m}. \quad (37)$$

Podstawienie równania (37) do wzorów (30) formułuje także problem programowania kwadratowego, ale względem zmiennych danych wektorem prędkości ruchu w połączeniach ruchowych redundantnych $\dot{\theta}_{n-m}$ [Cheng, Chen, Sun 1992] [Cheng, Chen, Wang, Sun 1993]:

Znajdź częściowy wektor prędkości w przegubach $\dot{\theta}_{n-m}$, który minimalizuje

$$\hat{\mathbf{g}}^T \dot{\theta}_{n-m} + \frac{1}{2} \dot{\theta}_{n-m}^T \hat{\mathbf{H}} \dot{\theta}_{n-m}, \quad (38)$$

i spełnia jednocześnie

$$\hat{\mathbf{A}} \dot{\theta}_{n-m} \geq \hat{\mathbf{b}},$$

przy czym:

$$\hat{\mathbf{g}}^T = (\mathbf{g}^T + \dot{\theta}_p^T \mathbf{H}) \mathbf{N},$$

$$\hat{\mathbf{H}} = \mathbf{N}^T \mathbf{H} \mathbf{N},$$

$$\hat{\mathbf{A}} = \mathbf{A} \mathbf{N},$$

$$\hat{\mathbf{b}} = \mathbf{b} - \mathbf{A} \dot{\theta}_p.$$

Po rozwiązaniu zadania programowania kwadratowego (38) podstawiamy znalezione rozwiązanie $\dot{\theta}_{n-m}$ do równania (37), by obliczyć prędkości we wszystkich połączeniach ruchowych robota.

Problem programowania kwadratowego, określony wzorami (38), zwany kompaktowym, ma tylko $n - m$ zmiennych i brak mu układu m -równań, co wyraźnie zwiększa efektywność obliczeniową podczas jego rozwiązywania.

Do obliczeń można zastosować efektywny algorytm Goldfarba i Idnani, rozwiązujący zadanie programowania kwadratowego [Goldfarb, Idnani 1983], a nawet użyć gotowych bibliotecznych programów napisanych wg tego algorytmu [IMSL 1987] [Cheng, Chen, Sun 1992] [Cheng, Chen, Wang, Sun 1993].

Jak pokazano na przykładzie symulacji ruchu płaskich robotów [Cheng, Chen, Wang, Sun 93] kompaktową metodę programowania kwadratowego można skutecznie zastosować do rozwiązania problemu ruchu redundantnego łańcucha kinematycznego, uwzględniając przy tym ograniczenia wynikające z granic wychyleń w połączeniach ruchowych i granic na prędkości ich ruchów, a także zapewniając omijanie przeszkód i powtarzalność ruchu po zamkniętej trajektorii. Metoda ta wykazuje przy tym wyższość w stosunku do innych metod, jeżeli chodzi o praktyczne sformułowanie zadania, jak i efektywność obliczeniową [Cheng, Chen, Wang, Sun 1993].

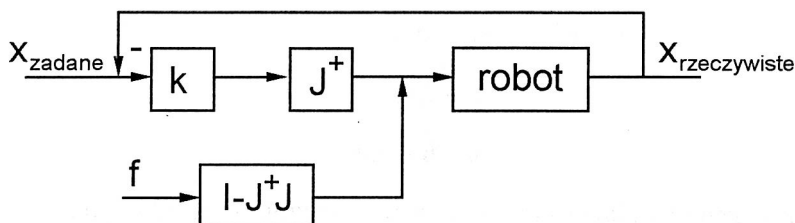
3.13. Sterowanie bazujące na logice rozmytej

Do rozwiązania problemu omijania przeszkód przez redundantny łańcuch kinematyczny można zastosować także sterowanie bazujące na logice rozmytej (ang. fuzzy logic) [Palm, Rehfuëß 1992]. Rysunek 3.15 pokazuje zastosowany układ regulacji położenia, którego równanie przedstawia się następująco:

$$\dot{\theta} = J^+ [k (x_{\text{zadane}} - x_{\text{rzeczywiste}})] + (I - J^+ J) f,$$

gdzie f – n -wymiarowy wektor tworzony na zasadach logiki rozmytej na podstawie sygnałów sensorycznych o przeszkodzie.

Zaletą tej metody jest to, że w przeciwieństwie np. do metod opartych na sztucznych polach potencjalnych, nie potrzeba tu tak dużej pracochłonności niezbędnej do geometrycznego modelowania ramion robota i przestrzeni roboczej (zob. rozdz. 3.5, a zwłaszcza rys. 3.9).



Rys. 3.15. Układ regulacji położenia z wejściem *fuzzy logic* f [Palm, Rehfuëß 1992]

Fig. 3.15. Positioning control system with fuzzy logic input f [Palm, Rehfuëß 1992]

3.14. Sterowanie bazujące na sieciach neuronowych

Do sterowania redundantnymi łańcuchami kinematycznymi próbuje się też zastosować sieci neuronowe. W robotyce znane są przykłady zastosowania sieci neuronowych do rozwiązywania takich zadań, jak: transformacja odwrotna, dynamika, planowanie trajektorii, sensoryka, rozpoznawanie obrazów i regulacja [Wloka² 1992]. Do rozwiązania transformacji odwrotnej konstruuje się zwykle sieci *feedforward* z połączeniami wyłącznie wprzód między neuronami ułożonymi w kilku warstwach (rys. 3.16).

Przykładem są sieci *backpropagation*, które charakteryzują się tym, że podczas fazy uczenia błędy są podawane jako sprzężenie zwrotne do sieci. Sieci takie nadają się do modelowania wielowymiarowych funkcji i stąd też próby ich stosowania do transformacji odwrotnej. Zaletami sieci neuronowych są po pierwsze, zbędność tworzenia algorytmu transformacji odwrotnej, gdyż rozwiązanie jest nauczone, po drugie, prędkość obliczeń jest znacznie większa, dzięki współbieżności operacji wykonywanych w sieciach. Zwykle jednak, jak dotychczas, rozwiązania dostarczane przez sieć

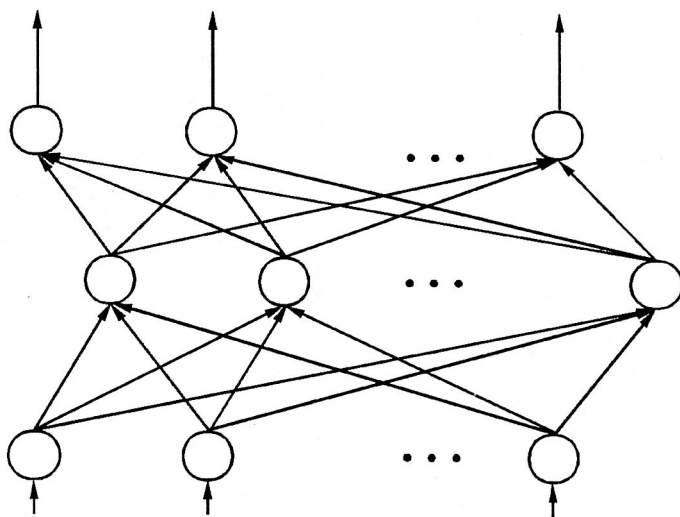
Rys. 3.16. Zasada budowy sieci neuronowej *feedforward*

Fig. 3.16. Construction principle of feedforward neuronal network

neuronową są niedokładne i trzeba uzupełnić je o fazę obliczeń numerycznych [Wloka² 1992]. Drugą wadą jest niezbędna duża liczba cykli uczących, idąca w setki tysięcy lub miliony.

Udało się z powodzeniem zastosować i wytestować w systemach symulacji sieci neuronowe rozwiązujące zadanie transformacji odwrotnej z omijaniem przeszkód dla prostych robotów redundantnych [Hsia, Mao 1993]. Na rysunku 3.17 przedstawiono przykładowy schemat takiej transformacji odwrotnej z użyciem wielowarstwowej sieci neuronowej *feedforward* z neuronami o funkcji sigmoidalnej w warstwach ukrytych i o funkcji liniowej w warstwie wyjściowej. Sieć neuronowa była w tym przypadku uczona jak minimalizować następujące kryterium:

$$H = \frac{1}{L} \sum_{p=1}^L [\alpha H_1(\theta^{(p)}) + (1 - \alpha) H_2(\mathbf{x}_{\text{zadane}}^{(p)}, \theta^{(p)}) + \mu H_3(\theta^{(p)})],$$

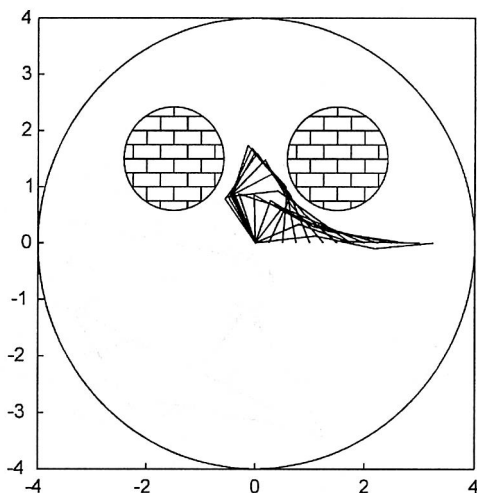
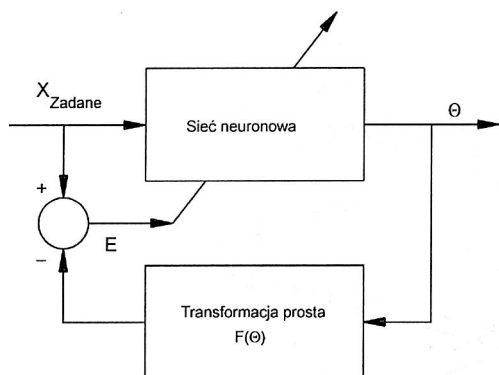
gdzie

$$H_1(\theta') = \frac{1}{2n} \sum_{k=1}^n (\theta'_k)^2$$

$$H_2(\mathbf{x}_{\text{zadane}}, \theta') = \frac{1}{2m} \sum_{j=1}^m (x_{j,\text{zadane}} - x_{j,\text{obliczone}}(\theta'))^2$$

$H_3(\theta') =$ najmniejsza odległość od przeszkody

$$\theta' = \theta - \frac{\theta_{\max} - \theta_{\min}}{2},$$



Rys. 3.17 Przykładowy schemat transformacji odwrotnej z użyciem sieci neuronowej [Hsia, Mao 1993]

Fig. 3.17. Example of inverse transformation scheme using the neuronal network [Hsia, Mao 1993]

przy czym suma od $p = 1$ do L jest sumowaniem po punktach uczących $\mathbf{x}_p$. Tak zaprojektowanym systemem udało się z powodzeniem sterować płaskim 4-przegubowym robotem w obecności przeszkód [Hsia, Mao 1993].

3.15. Regulacja położenia

Tradycyjne klasyczne układy regulacji położenia w urządzeniach wytwórczych, takich jak obrabiarki i roboty, są stosowane do każdej poszczególniej sterowanej osi [Weck 1989] [Dillman, Huck 1991] [Pritschow 1995]. W najprostszym, lecz powszechnie stosowanym przypadku, układ taki przyjmuje dla robotów przemysłowych postać kaskadową z wewnętrzną pętlą regulacji prędkości [Weck 1989] [Dillman, Huck 1991], jak to pokazano na rysunku 3.18 a.

W przedstawionych w rozdziale 4. koncepcjach sterowania redundantnymi łańcuchami kinematycznymi rozwiązanie redundancji następuje w przestrzeni prędkości, a nie położenia. Przy bezpośrednim zastosowaniu tych metod w układach regulacji położenia staje się nieuniknione to, że prędkość występuje jako wielkość sterująca, a regulacja w przestrzeni współrzędnych kartezjańskich. Przy sterowaniu przez pseudoinwersję równanie regulatora proporcjonalnego może wyglądać wtedy następująco [Nakamura, Hanafusa, Yoshikawa 1987] [Nakamura 1991] [Koch 1992] (rys. 3.18 b i c):

$$\dot{\mathbf{x}}_{\text{zadane}} - \dot{\mathbf{x}}_{\text{wejściowe}} + k(\mathbf{x}_{\text{zadane}} - \mathbf{x}_{\text{rzeczywiste}}) = 0,$$

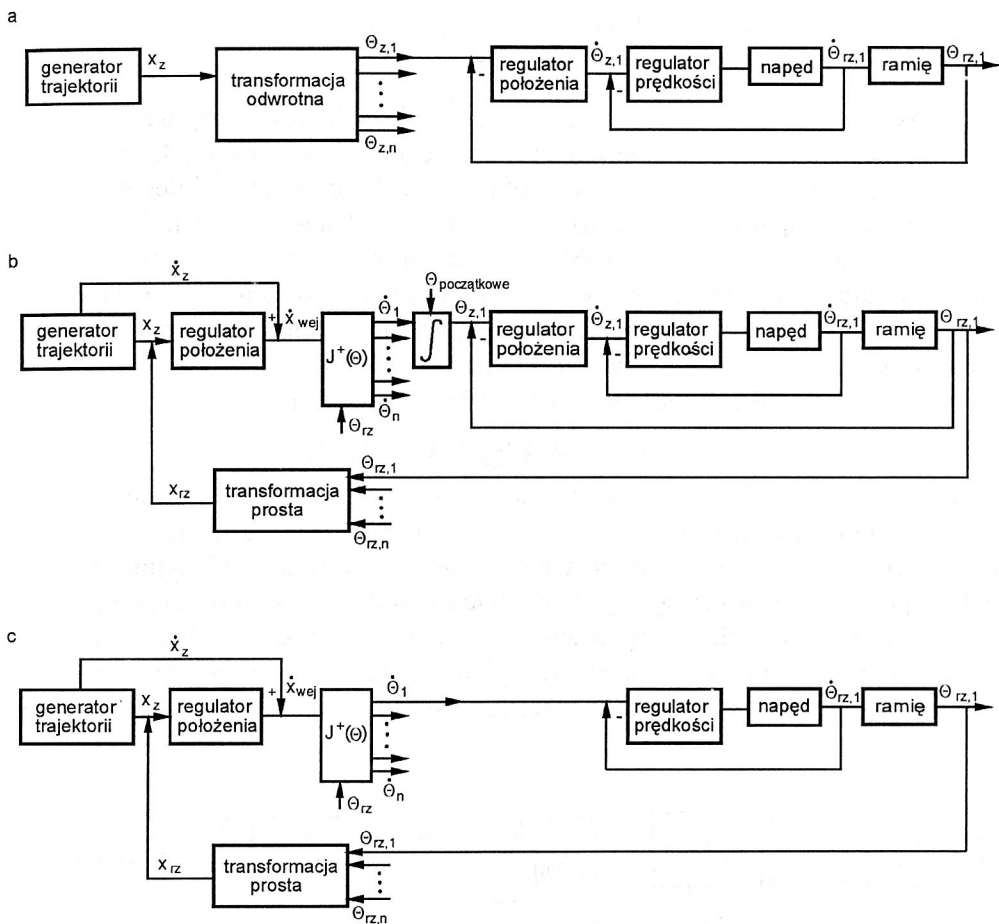
przy czym:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^+ \dot{\mathbf{x}}_{\text{wejściowe}},$$

lub inaczej podstawiając $\dot{\mathbf{x}}_{\text{wejściowe}}$ z równania regulatora:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^+ [\dot{\mathbf{x}}_{\text{zadane}} + k(\mathbf{x}_{\text{zadane}} - \mathbf{x}_{\text{rzeczywiste}})]$$

Możliwe jest również pominięcie pętli regulacji prędkości w poszczególnych osiach, tak jak to zaproponowano w metodzie sterowania konfiguracją (zob. rozdz. 3.10), gdzie



Rys. 3.18. Układy regulacji położenia: tradycyjny kaskadowy układ regulacji [Weck 1989] [Dillman, Huck 1991] (a), układ regulacji przy bezpośrednim zastosowaniu rozwiązania redundancji przez pseudoinwersję [Nakamura 91] (b), układ regulacji przy bezpośrednim zastosowaniu rozwiązania redundancji przez pseudoinwersję z pominięciem pętli regulacji położenia w osiach [Seraji, Colbaugh 1990] (c)

Fig. 3.18. Positioning control systems: conventional cascade positioning control system [Weck 1989] [Dillman, Huck 1991] (a), positioning control system while applying directly the redundancy resolution using pseudoinverse [Nakamura 91] (b), positioning control system while applying directly the redundancy resolution using pseudoinverse without joint position control loops [Seraji, Colbaugh 1990] (c)

wektor sił/momentów na członie roboczym zostaje obliczony bezpośrednio jako funkcja uchybu współrzędnych kartezjańskich $\mathbf{E} = \mathbf{x}_{\text{zadane}} - \mathbf{x}_{\text{rzeczywiste}}$ oraz zadanego przebiegu trajektorii $\mathbf{x}_{\text{zadane}}(t)$ wg następującego równania regulatora:

$$\mathbf{F} = K_p \mathbf{E} + K_v \dot{\mathbf{E}} + C \mathbf{x}_{\text{zadane}} + B \dot{\mathbf{x}}_{\text{zadane}} + A \ddot{\mathbf{x}}_{\text{zadane}}.$$

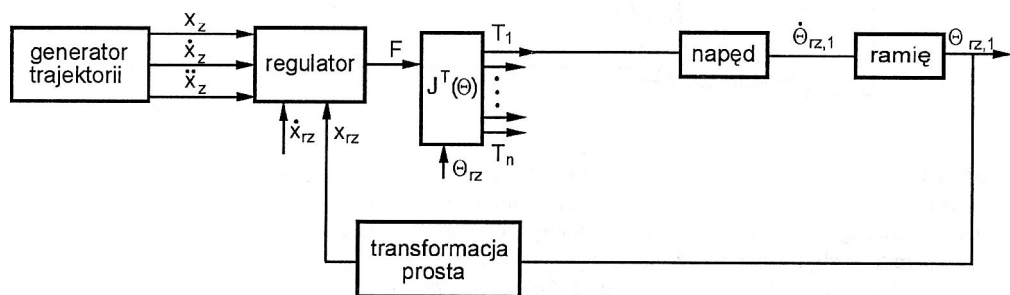
Podawane na napędy poszczególnych osi sygnały momentów są następnie obliczone wg znanej zależności (rys. 3.19)

$$\mathbf{T} = \mathbf{J}^T \mathbf{F}.$$

Alternatywne rozwiązanie problemu regulacji położenia, przy którym są zachowane tradycyjne układy regulacji położenia w poszczególnych osiach (rys. 3.18a), wymaga zastosowania metod iteracyjnych do obliczenia w każdym takcie interpolacyjnym poszczególnych współrzędnych konfiguracyjnych. Przykładem może tu być metoda Newtona-Raphsona. W metodzie tej oblicza się kolejne przybliżenia wektora współrzędnych konfiguracyjnych wg wzoru, który w przypadku rozwiązania redundancji poprzez pseudoinwersję przyjmuje postać [Duelen, Kirchoff, Held, Münch 1987] [Münch 1991] [Andersen, Cook, Zein-Sabattou, Fernandez 1989] [Cook, Andersen, Zein-Sabattou, Fernandez 1989] [Pritschow, Koch 1991] [Koch 1991]

$$\boldsymbol{\theta}_{\text{nowe}} = \boldsymbol{\theta}_{\text{stare}} - \mathbf{J}^+ (\boldsymbol{\theta}_{\text{stare}}) [\mathbf{x} - f(\boldsymbol{\theta}_{\text{stare}})].$$

Dla zadanego położenia $\mathbf{x}$ członu roboczego we współrzędnych kartezjańskich, $\boldsymbol{\theta}_{\text{nowe}}$ jest obliczane dopóki różnica $\mathbf{x} - f(\boldsymbol{\theta}_{\text{nowe}})$ nie będzie dostatecznie mała, co stanowi warunek na przerwanie procesu iteracji. Zaletą stosowania tradycyjnie układów regulacji położenia w poszczególnych osiach jest mniejszy czas opóźnienia wyregulowania uchybu niż w układzie regulacji we współrzędnych kartezjańskich, gdzie w obwodzie regulacji dodatkowo musi być wykonana transformacja prosta i obliczenie macierzy Jacobiego (rys. 3.19) bądź macierzy pseudoinwersji (rys. 3.18 b i c) i wy-



Rys. 3.19. Układ regulacji położenia we współrzędnych kartezjańskich bez pętli regulacji w poszczególnych osiach [Seraji 1989]

Fig. 3.19. Positioning control system in cartesian coordinates without any control loops in the joints [Seraji 1989]

mnożenie macierzy przez wektor. Inną zaletą tego rozwiązania jest utrzymanie rozdziału między transformacją odwrotną i regulacją położenia, co umożliwia jednolitą realizację tworzenia ruchu przy jego zaprogramowaniu we współrzędnych kartezjańskich i konfiguracyjnych. W układzie regulacji we współrzędnych kartezjańskich brak tego rozdziału i aby umożliwić tu programowanie ruchu we współrzędnych konfiguracyjnych, w sterowaniu robotem musi być zaprogramowany dodatkowo alternatywny algorytm przetwarzania danych z samymi układami regulacji położenia w osiach.

Przypadki układów regulacji pokazanych na rysunkach 3.18 b, 3.18 c i 3.18 a z transformacją odwrotną na drodze metody iteracyjnej Newtona-Raphsona przedstawiono na przykładzie rozwiązania przez pseudoinwersję. Można naturalnie zastosować tu każde inne rozwiązanie przedstawione w rozdziale 3, rozwiązujące redundancję w dziedzinie prędkości.

4. STEROWANIE ZORIENTOWANE NA PROGRAMOWANIE

4.1. Wymagania w stosunku do algorytmu sterowania

Stworzono wiele wartościowych algorytmów sterowania redundantnymi łańcuchami kinematycznymi (rozdz. 3). Algorytmy te stanowią wystarczającą bazę do stworzenia koncepcji programowania ruchu robotów redundantnych. Koncepcja programowania musi oczywiście bazować na przyjętej koncepcji sterowania, której wybór wpływa na możliwości i sposób programowania. Dlatego też należałoby najpierw ustalić katalog wymagań w stosunku do algorytmu sterowania. Uwzględnivszy specyfikę zastosowań przemysłowych zestawiono następujący katalog wymagań [Koch² 1994] [Koch² 1995] [Koch³ 1995]:

- algorytm powinien być lokalny, aby umożliwiać dokonywanie korekcji toru ruchu w czasie rzeczywistym, np. na podstawie sygnałów od sensorów (patrz rozdz. 3.1),
- algorytm powinien zapewniać powtarzalność ruchu przy cyklicznym powtarzaniu tej samej zamkniętej trajektorii (patrz rozdz. 3.8), co w zastosowaniach przemysłowych jest bardzo częstym przypadkiem,
- algorytm powinien umożliwiać opcjonalną i parametryzowaną realizację dodatkowych zadań, wynikających z samej istoty bądź z warunków brzegowych podstawowego zadania; do podstawowej biblioteki takich zadań powinno należeć:
 - * utrzymanie wychyleń w połączeniach ruchowych w granicach
 - * omijanie przeszkód,
- algorytm powinien być otwarty i umożliwiać uzupełnianie przez użytkownika robota wymienionej już biblioteki o dodatkowe zadania,
- algorytm powinien być zgodny z koncepcją tzw. sterowania o otwartej modułowej architekturze (OSACA – *Open Structure Architecture for Controls within Automation Systems*) [OSACA], gdzie każdy z bloków funkcyjnych jest niezależny od innych, co pociąga za sobą niezależność modułu transformacji odwrotnej od poprzedzającego go i następującego po nim przetwarzania danych w sterowaniu, tzn.:
 - * planowania trajektorii członu roboczego we współrzędnych kartezjańskich
 - * regulacji położenia,
- algorytm powinien umożliwiać mieszany sposób programowania ruchu, zarówno we współrzędnych kartezjańskich, jak i we współrzędnych konfiguracyjnych,
- algorytm powinien być możliwie stosowalny nie tylko do różnych szeregowych otwartych łańcuchów kinematycznych, ale także do koordynacji ruchu kilku robotów oraz robotów mobilnych,
- program sterujący napisany wg tego algorytmu powinien nadawać się do zastosowania jako standardowy moduł sterowania redundantnymi robotami przemysłowymi.

Zestawienie powyższego katalogu nasunęło pomysł konstrukcji nowego oryginalnego algorytmu sterowania, będącego integracją i zarazem modyfikacją kilku podstawowych algorytmów, przedstawionych w rozdziale 3. Algorytm ten opisano w podrozdziale 4.2. W skrócie, polega on na zastosowaniu kompaktowej metody programowania kwadratowego w iteracyjnej metodzie Newtona-Raphsona z elementami sterowania konfiguracją. Spełnia on wymagania zestawione w zamieszczonej liście, a utrzymanie przy jego zastosowaniu układów regulacji w poszczególnych osiach daje ponadto komfort małego opóźnienia wyregulowania uchybu położenia (patrz rozdział 3.15).

4.2. Kompaktowa metoda programowania kwadratowego w iteracyjnej metodzie Newtona-Raphsona

Punkt wyjściowy matematycznego opisu algorytmu stanowi problem transformacji odwrotnej. Dla zadanego wektora współrzędnych kartezjańskich członu roboczego robota $\mathbf{x}$ należy znaleźć wektor współrzędnych konfiguracyjnych $\boldsymbol{\theta}$, który spełnia równanie transformacji prostej

$$\mathbf{x} = f(\boldsymbol{\theta}) .$$

Dodatkowo może być pożądanе nałożenie więzów na współrzędne konfiguracyjne, tak jak w metodzie sterowania konfiguracją

$$\Phi(\boldsymbol{\theta}) = 0 .$$

Przykłady takich więzów dla robota mobilnego i kooperujących manipulatorów podano w rozdziale 3.2.

Oba równania można zintegrować w jedno:

$$F(\boldsymbol{\theta}) = \begin{bmatrix} f(\boldsymbol{\theta}) - \mathbf{x} \\ \Phi(\boldsymbol{\theta}) \end{bmatrix} = 0$$

i rozwiązać iteracyjnie w każdym takcie interpolacyjnym wg schematu Newtona-Raphsona:

$$\begin{aligned} \text{(a) } \mathbf{J} \Delta\boldsymbol{\theta} &= -F(\boldsymbol{\theta}_{\text{stare}}), \\ \text{(b) } \boldsymbol{\theta}_{\text{nowe}} &= \boldsymbol{\theta}_{\text{stare}} + \Delta\boldsymbol{\theta}, \end{aligned} \tag{39}$$

gdzie $\mathbf{J}$ – macierz Jacobiego wielowymiarowej funkcji $F(\boldsymbol{\theta})$ ($\mathbf{J} = \partial F / \partial \boldsymbol{\theta}$), której pierwsze m wierszy tworzą jacobian robota. Warunkiem przerwania iteracji jest spełnienie nierówności: $F(\boldsymbol{\theta}_{\text{nowe}}) < \varepsilon$.

Jeśli liczba więzów n_{Φ} , będąca wymiarem funkcji $\Phi(\boldsymbol{\theta})$, nie wyeliminowała redundancji ($m + n_{\Phi} < n$), układ równań (39a) jest rozwiązywany za pomocą metody programowania kwadratowego, która rozwiązuje następująco sformułowany problem:

znajdź wektor $\Delta\theta$, który minimalizuje

$$(a) \mathbf{g}^T \Delta\theta + \frac{1}{2} \Delta\theta^T \mathbf{H} \Delta\theta$$

i spełnia równocześnie

$$(b) \mathbf{J} \Delta\theta = -F(\theta_{\text{stare}}) \quad (40)$$

$$(c) \mathbf{A} \Delta\theta \geq \mathbf{b},$$

przy czym forma kwadratowo-liniowa (40a) może służyć minimalizacji kryteriów, a nierówność (40c) ma zastosowanie do określenia granic wychyleń w połączeniach ruchowych, granic ich prędkości i przyspieszeń.

Dla większej efektywności obliczeniowej zadanie programowania kwadratowego (40) należy przekształcić na postać kompaktową, podobnie jak w rozdziale 3.12: znajdź częściowy wektor $\Delta\theta_{n-m}$, który minimalizuje

$$\hat{\mathbf{g}}^T \Delta\theta_{n-m} + \frac{1}{2} \Delta\theta_{n-m}^T \mathbf{H} \Delta\theta_{n-m},$$

i spełnia równocześnie

$$\hat{\mathbf{A}} \Delta\theta_{n-m} \geq \hat{\mathbf{b}}, \quad (41)$$

przy czym:

$$\hat{\mathbf{g}}^T = (\mathbf{g}^T + \Delta\theta_p^T \mathbf{H}) \mathbf{N},$$

$$\hat{\mathbf{H}} = \mathbf{N}^T \mathbf{H} \mathbf{N},$$

$$\hat{\mathbf{A}} = \mathbf{A} \mathbf{N},$$

$$\hat{\mathbf{b}} = \mathbf{b} - \mathbf{A} \Delta\theta_p,$$

gdzie:

$$\Delta\theta_p = \begin{bmatrix} -\mathbf{J}_m^{-1} F(\theta_{\text{stare}}) \\ 0_{n-m} \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} -\mathbf{J}_m^{-1} \mathbf{J}_{n-m} \\ \mathbf{I}_{n-m} \end{bmatrix}.$$

Po rozwiązaniu zadania programowania kwadratowego (41) należy jeszcze obliczyć, na podstawie znalezionej rozwiązania $\Delta\theta_{n-m}$, pełny wektor $\Delta\theta$

$$\Delta\theta = \begin{bmatrix} \Delta\theta_m \\ \Delta\theta_{n-m} \end{bmatrix} = \Delta\theta_p + \mathbf{N} \Delta\theta_{n-m}.$$

Do rozwiązania zadania programowania kwadratowego można użyć efektywnego algorytmu Goldfarba i Idnani [Goldfarb, Idnani 1983], stosując np. gotową jego aplikację dokonaną przez Powella [Powell¹ 1983] [Powell² 1983].

4.2.1. Utrzymanie wychyleń w połączeniach ruchowych i ich prędkości w zadanych granicach

Granice wychyleń w połączeniach ruchowych są określone nierównościami:

$$\theta_{i,\min} \leq \theta_i \leq \theta_{i,\max}, \quad (42)$$

a granice prędkości przez nierówność

$$\dot{\theta}_{i,\min} \leq \dot{\theta}_i \leq \dot{\theta}_{i,\max}. \quad (43)$$

Aby dostosować te nierówności do postaci (40c), muszą one zostać przekształcone. Od stron nierówności (42) należy odjąć $\theta_{i,\text{stare}}$:

$$\theta_{i,\min} - \theta_{i,\text{stare}} \leq \theta_{i,\text{nowe}} - \theta_{i,\text{stare}} \leq \theta_{i,\max} - \theta_{i,\text{stare}},$$

co określa nierówność na $\Delta\theta_i$:

$$\theta_{i,\min} - \theta_{i,\text{stare}} \leq \Delta\theta_i \leq \theta_{i,\max} - \theta_{i,\text{stare}}. \quad (44)$$

Natomiast nierówność (43) można określić następująco, posługując się taktem interpolacji Δt :

$$\dot{\theta}_{i,\min} \leq \frac{\theta_{i,\text{nowe}} - \theta_{i,\text{początkowe}}}{\Delta t} \leq \dot{\theta}_{i,\max},$$

gdzie $\theta_{i,\text{początkowe}}$ – punkt startowy procesu iteracji dla i -tego połączenia ruchowego.

Ponieważ różnica $\theta_{i,\text{nowe}} - \theta_{i,\text{początkowe}}$ jest sumą przyrostu $\Delta\theta_i$ i wszystkich poprzednich przyrostów od początku aktualnego procesu iteracji dla danego taktu interpolacji, więc

$$\theta_{i,\text{nowe}} - \theta_{i,\text{początkowe}} = \Delta\theta_i + \sum \Delta\theta_i.$$

Tak więc nierówność (43) można przedstawić następująco

$$\Delta t \dot{\theta}_{i,\min} - \sum \Delta\theta_i \leq \Delta\theta_i \leq \Delta t \dot{\theta}_{i,\max} - \sum \Delta\theta_i. \quad (45)$$

Integrując nierówności (44) i (45) w jedną:

$$\begin{aligned} \Delta\theta_i &\geq \max(\theta_{i,\min} - \theta_{i,\text{stare}}, \Delta t \dot{\theta}_{i,\min} - \sum \Delta\theta_i), \\ -\Delta\theta_i &\geq -\min(\theta_{i,\max} - \theta_{i,\text{stare}}, \Delta t \dot{\theta}_{i,\max} - \sum \Delta\theta_i) \end{aligned}$$

można odnaleźć, podobnie jak w rozdziale 3.12, macierz $\mathbf{A}$ i wektor $\mathbf{b}$ dla nierówności (40c):

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_n \\ -\mathbf{I}_n \end{bmatrix}_{2n \times n}$$

$$\mathbf{b} = \begin{bmatrix} \max(\theta_{1,\min} - \theta_{1,\text{stare}}, \Delta t \dot{\theta}_{1,\min} - \sum \Delta \theta_1) \\ \vdots \\ \max(\theta_{n,\min} - \theta_{n,\text{stare}}, \Delta t \dot{\theta}_{1,\min} - \sum \Delta \theta_n) \\ -\min(\theta_{1,\max} - \theta_{1,\text{stare}}, \Delta t \dot{\theta}_{1,\max} - \sum \Delta \theta_1) \\ \vdots \\ -\min(\theta_{n,\max} - \theta_{n,\text{stare}}, \Delta t \dot{\theta}_{1,\max} - \sum \Delta \theta_n) \end{bmatrix}_{2n \times 1}$$

Można również uwzględnić ograniczenia na przyspieszenie

$$\ddot{\theta}_{i,\min} \leq \ddot{\theta}_i \leq \ddot{\theta}_{i,\max}.$$

Aby dostosować powyższą nierówność do postaci (40c), należy wykonać jeszcze następujące przekształcenia:

$$\ddot{\theta}_{i,\min} \leq \frac{\dot{\theta}_{i,\text{nowe}} - \dot{\theta}_{i,\text{poprzednie}}}{\Delta t} \leq \ddot{\theta}_{i,\max},$$

$$\ddot{\theta}_{i,\min} \leq \frac{\theta_{i,\text{nowe}} - \theta_{i,\text{początkowe}}}{\Delta^2 t} - \frac{\dot{\theta}_{i,\text{poprzednie}}}{\Delta t} \leq \ddot{\theta}_{i,\max},$$

$$\ddot{\theta}_{i,\min} \leq \frac{\Delta \theta_i + \sum \Delta \theta_i}{\Delta^2 t} - \frac{\dot{\theta}_{i,\text{poprzednie}}}{\Delta t} \leq \ddot{\theta}_{i,\max},$$

$$\Delta^2 t \left(\ddot{\theta}_{i,\min} + \frac{\dot{\theta}_{i,\text{poprzednie}}}{\Delta t} \right) - \sum \Delta \theta_i \leq \Delta \theta_i \leq \Delta^2 t \left(\ddot{\theta}_{i,\max} + \frac{\dot{\theta}_{i,\text{poprzednie}}}{\Delta t} \right) - \sum \Delta \theta_i.$$

Zintegrowana nierówność, uwzględniająca wszystkie trzy ograniczenia dotyczące wychYLENIA, jego prędkości i przyspieszenia, będąca podstawą sformułowania wektora $\mathbf{b}$, przedstawia się następująco:

$$\Delta \theta_i \geq \max(\theta_{i,\min} - \theta_{i,\text{stare}}, \Delta t \dot{\theta}_{i,\min} - \sum \Delta \theta_i, \Delta^2 t \ddot{\theta}_{i,\min}^{\text{zastępcze}} - \sum \Delta \theta_i)$$

$$-\Delta \theta_i \geq -\min(\theta_{i,\max} - \theta_{i,\text{stare}}, \Delta t \dot{\theta}_{i,\max} - \sum \Delta \theta_i, \Delta^2 t \ddot{\theta}_{i,\max}^{\text{zastępcze}} - \sum \Delta \theta_i)$$

gdzie

$$\ddot{\theta}_{i,\min}^{\text{zastępcze}} = \ddot{\theta}_{i,\min} + \frac{\dot{\theta}_{i,\text{poprzednie}}}{\Delta t}$$

$$\ddot{\theta}_{i,\max}^{\text{zastępcze}} = \ddot{\theta}_{i,\max} + \frac{\dot{\theta}_{i,\text{poprzednie}}}{\Delta t}.$$

4.2.2. Zapewnienie powtarzalności ruchu

Dla zapewnienia powtarzalności ruchu wystarczy zadbać o to, aby konfiguracja robota była podczas całego ruchu po zamkniętej trajektorii możliwie jak najbliższa konfiguracji początkowej θ^0 . Można to osiągnąć na drodze minimalizacji następującego kryterium

$$\frac{1}{2} (\theta_{\text{nowe}} - \theta^0)^T (\theta_{\text{nowe}} - \theta^0).$$

Kryterium to można jeszcze poprawić przez uzupełnienie go o macierz wag

$$\frac{1}{2} (\theta_{\text{nowe}} - \theta^0)^T \mathbf{W} (\theta_{\text{nowe}} - \theta^0), \quad (46)$$

przy czym $\mathbf{W}$ jest macierzą diagonalną, a elementy na diagonalu, w przypadku szeregowego otwartego łańcucha kinematycznego, należy dobrać tak, aby: $w_{11} \geq w_{22} \geq \dots \geq w_{mm} \geq 0$. Taki dobór wag powoduje, że wychylenia w pierwszych połączeniach ruchowych bliskich podstawie robota są na początku trajektorii możliwie niewielkie, podczas gdy na końcu ruchu, tzn. przy powrocie do początku zamkniętej trajektorii, możliwie duże w kierunku początkowej konfiguracji.

Kryterium wyrażone wzorem (46) musi jeszcze zostać przekształcone do postaci (40a). Przez wprowadzenie θ_{stare} :

$$\frac{1}{2} (\theta_{\text{nowe}} - \theta_{\text{stare}} + \theta_{\text{stare}} - \theta^0)^T \mathbf{W} (\theta_{\text{nowe}} - \theta_{\text{stare}} + \theta_{\text{stare}} - \theta^0),$$

kryterium uzyskuje formułę na $\Delta\theta$:

$$\begin{aligned} & \frac{1}{2} (\Delta\theta + \theta_{\text{stare}} - \theta^0)^T \mathbf{W} (\Delta\theta + \theta_{\text{stare}} - \theta^0) = \\ & = \frac{1}{2} \Delta\theta^T \mathbf{W} \Delta\theta + (\theta_{\text{stare}} - \theta^0)^T \mathbf{W} \Delta\theta + \frac{1}{2} (\theta_{\text{stare}} - \theta^0)^T \mathbf{W} (\theta_{\text{stare}} - \theta^0). \end{aligned}$$

Ostatni składnik powyższej sumy jest stały i nie ma wpływu na optymalizację, tak więc odnajdujemy $\mathbf{g}^T$ i $\mathbf{H}$ z (40a) jako:

$$\mathbf{g}^T = (\theta_{\text{stare}} - \theta^0)^T \mathbf{W}, \quad \mathbf{H} = \mathbf{W}.$$

4.2.3. Zmiana konfiguracji

Podczas zmiany konfiguracji chodzi o to, aby robot zmienił wychylenia w swoich połączeniach ruchowych na zadane w postaci jawnej współrzędne θ^z , nie zmieniając przy tym aktualnego położenia i orientacji członu roboczego. Można to osiągnąć na drodze minimalizacji kryterium zaproponowanego w poprzednim rozdziale 4.2.2 dla powtarzalności ruchu:

$$\frac{1}{2} (\theta_{\text{nowe}} - \theta^z)^T \mathbf{W} (\theta_{\text{nowe}} - \theta^z),$$

przy czym w równaniach (39a) oraz (40b) należy wtedy w pierwszej iteracji Newtona-Raphsona przyjąć $F(\theta_{\text{stare}}) = 0$. Wzory na $\mathbf{g}^T$ i $\mathbf{H}$ są więc w tym przypadku takie same jak w poprzednim rozdziale 4.2.2:

$$\mathbf{g}^T = (\theta_{\text{stare}} - \theta^z)^T \mathbf{W}, \quad \mathbf{H} = \mathbf{W}.$$

Ograniczenia dane algorytmem na utrzymanie wychyleń w połączeniach ruchowych i ich prędkości oraz ewentualnie przyspieszenia w zadanych granicach zapewniają, że zmiany konfiguracji w poszczególnych krokach interpolacji nie będą zbyt duże.

Jeśli współrzędne konfiguracyjne dane wektorem θ^z nie odpowiadają aktualnemu położeniu i orientacji członu roboczego ($\mathbf{x}_{\text{aktualne}} \neq f(\theta^z)$), zmiana konfiguracji polega na możliwie największym, w sensie ważonego kryterium kwadratowego, zbliżeniu się do zadanej konfiguracji θ^z . Polecenie zmiany konfiguracji zostaje przerwane, jeśli w kolejnym takcie interpolacji zmiany w konfiguracji są mniejsze niż zadana wartość $\varepsilon_{\text{conf}}$:

$$\|\theta(\text{takt}) - \theta(\text{takt} - 1)\| < \varepsilon_{\text{conf}}.$$

4.2.4. Omijanie przeszkód

Algorytm omijania przeszkód

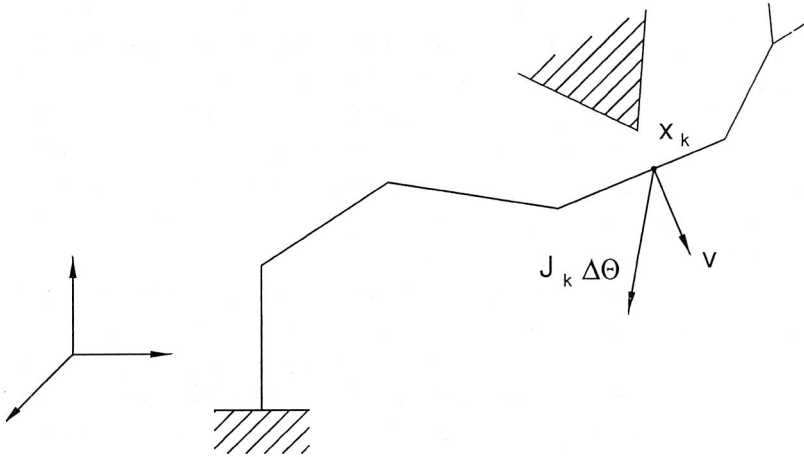
Punktem wyjściowym problemu omijania przeszkód jest informacja o punkcie na robocie leżącym najbliżej przeszkody, zwanym punktem krytycznym $\mathbf{x}_k$ oraz o kierunku ruchu $\mathbf{v}$ tego punktu, który by go najszybciej oddalał od przeszkody (rys. 4.1). Omijanie przeszkody można wtedy realizować minimalizując kryterium

$$\frac{1}{2} (\alpha \mathbf{v} - \mathbf{J}_k \Delta \theta)^T (\alpha \mathbf{v} - \mathbf{J}_k \Delta \theta), \quad (47)$$

przy czym $\mathbf{J}_k$ jest macierzą Jacobiego robota, policzoną dla punktu $\mathbf{x}_k$, a stała α powinna rosnąć wraz ze zbliżaniem się punktu krytycznego do przeszkody, np. wg zależności: $\alpha = 1/R$, gdzie R – odległość od przeszkody.

Przekształcając kryterium (47) do postaci

$$\frac{1}{2} \alpha^2 \mathbf{v}^T \mathbf{v} - \alpha \mathbf{v}^T \mathbf{J}_k \Delta \theta + \frac{1}{2} \Delta \theta^T \mathbf{J}_k^T \mathbf{J}_k \Delta \theta$$



Rys. 4.1. Omijanie przeszkody przez robota

Fig. 4.1. Robot avoiding obstacle

i pomijając pierwszy stały składnik, nie mający wpływu na optymalizację, odnajdujemy $\mathbf{g}^T$ i $\mathbf{H}$ z (40a) jako:

$$\mathbf{g}^T = -\alpha \mathbf{v}^T \mathbf{J}_k, \quad \mathbf{H} = \mathbf{J}_k^T \mathbf{J}_k.$$

Dla płynnego ruchu połączeń ruchowych celowe jest posłużenie się kryterium będącym ważoną sumą kryterium omijania przeszkód i kryterium minimalnej zmiany konfiguracji

$$\alpha_{\text{przeszk}} \frac{1}{2} (\alpha \mathbf{v} - \mathbf{J}_k \Delta\theta)^T (\alpha \mathbf{v} - \mathbf{J}_k \Delta\theta) + \alpha_{\text{konf}} \frac{1}{2} \Delta\theta^T \mathbf{W} \Delta\theta, \quad (48)$$

używamy przy tym np. diagonalnej macierzy wag $\mathbf{W}$ określonej jak w rozdziale 4.2.2. Odnajdujemy wtedy $\mathbf{g}^T$ i $\mathbf{H}$ z (40a) jako:

$$\mathbf{g}^T = -\alpha_{\text{przeszk}} \alpha \mathbf{v}^T \mathbf{J}_k, \quad \mathbf{H} = \alpha_{\text{przeszk}} \mathbf{J}_k^T \mathbf{J}_k + \alpha_{\text{konf}} \mathbf{W}. \quad (49)$$

Problem znalezienia punktu krytycznego na robocie $\mathbf{x}_k$

Odmienne do poprzednich dwóch zadań, omijanie przeszkód wymaga rozwiązania nietrywialnego problemu znalezienia punktu na ramieniu robota, leżącego najbliżej przeszkody, zwanego punktem krytycznym $\mathbf{x}_k$ oraz kierunku ruchu $\mathbf{v}$ tego punktu, który by go najszybciej oddalał od przeszkody.

Istnieje wiele podejść do tego zagadnienia. Prócz rozwiązań opartych na informacji o jednej parze punktów krytycznych, jako najbliższych punktów między robotem i jego gniazdem, proponuje się też rozwiązania uwzględniające informacje o kilku takich parach. Zaletą tego drugiego podejścia jest to, że w sytuacji, kiedy kilka prze-

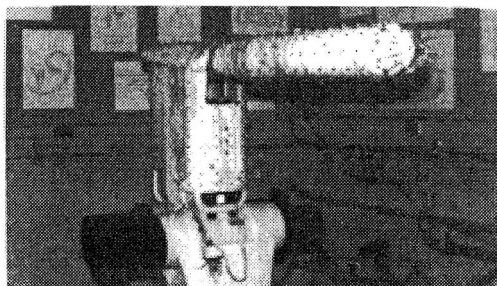
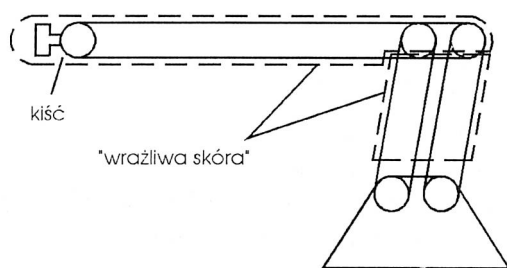
szkód jest blisko ramion robota, nie ma niebezpieczeństwa oscylacyjnych ruchów tych ramion pomiędzy przeszkodami. Tworzone algorytmy omijania przeszkód są albo uniwersalne, albo podają rozwiązanie dla pewnej konkretnej sytuacji, jak np. robot operujący przez okno [Zghal, Dubey, Euler 1992] [Glass, Colbaugh, Lim, Seraji 1993].

Różne są podejścia też do metod uzyskiwania informacji o przeszkodach i punkcie krytycznym. Informacje te można uzyskać od sensorów lub z systemu CAD, gdzie zamodelowano scenę gniazda zrobotyzowanego i jest dokonywana symulacja ruchu współbieżnie do ruchu rzeczywistego robota.

Użycie sensorów może oznaczać, w przypadku prostych dwustanowych sensorów zbliżeniowych, ich montaż, w dość dużej liczbie, na ramionach robota [Boddy, Taylor 1993] lub w mniejszej liczbie – w przypadku sensorów dostarczających informacji o odległości i kierunku od przeszkody [Novak, Feddama 1992]. Jeszcze inne rozwiązanie to pokrycie ramion wręcz wrażliwą skórą, np. stanowiącą siatkę par sensorów podczerwieni umieszczonych na elastycznej płycie drukowanej (rys. 4.2) [Cheung, Lumelsky 1990].

Inna metoda to umieszczenie jednej lub kilku kamer poza robotem, w jego gnieździe, w przypadku robota stacjonarnego, lub na robocie, w przypadku robota mobilnego. Dostępna od kilku lat technologia umożliwiła powstanie kamer w postaci skanerów laserowych, które dają rzeczywiste obrazy 3D (trójwymiarowe). Przykładem jest *Perceptron LASARTM*-System, który znalazł zastosowanie nie tylko w technologiach *Reverse Engineering* i *Rapid Prototyping*, ale także w sterowaniu robotami mobilnymi, robotami usługowymi w przemyśle nuklearnym, do automatycznego montażu w kosmosie, czy do pobierania ze skrzynki przez robota manipulacyjnego dowolnie wrzucanych elementów [Christensen 1993]. Produkowane są LASARTM-kamery o obszarze widzenia (rys. 4.3) sięgającym od $2\text{ m} \times 2\text{ m} \times 2\text{ m}$ do $20\text{ m} \times 20\text{ m} \times 20\text{ m}$.

Jeśli idzie o uzyskanie informacji o punkcie krytycznym z systemu CAD, to wiąże się to z obliczaniem odległości pomiędzy obiektami o postaci zdefiniowanych figur

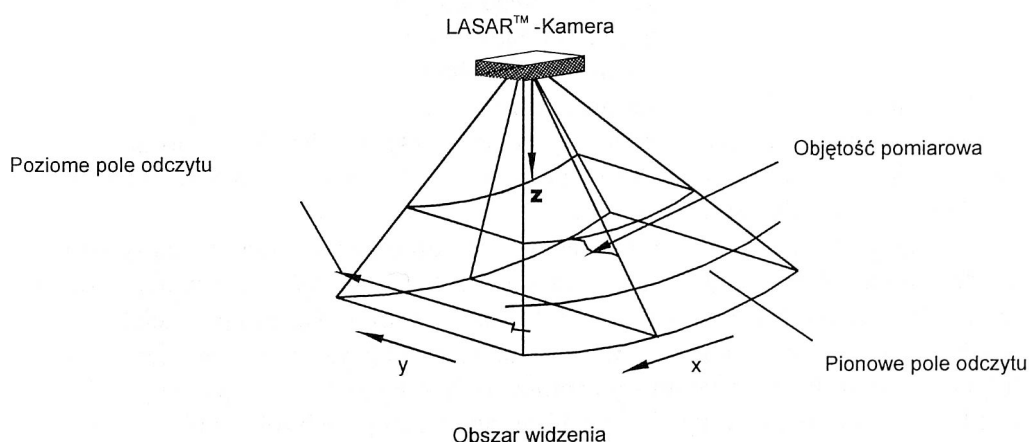


Rys. 4.2. Manipulator pokryty wrażliwą skórą, stanowiącą siatkę par sensorów podczerwieni umieszczonych na elastycznej płycie drukowanej [Cheung, Lumelsky 90]

Fig. 4.2. Manipulator covered by the sensitive skin made from elastic printed circuit board filled with the pairs of infrared sensors [Cheung, Lumelsky 90]

czy brył. Zadanie przecinania się obiektów ma natomiast zastosowanie do programowania czy planowania *off-line* i daje się je rozwiązać standardowymi procedurami dostępnymi w systemach CAD, na których bazie stworzono systemy symulacji robotów [ROBCAD] [AutoCAD].

Do obliczania odległości pomiędzy bryłami istnieje wiele algorytmów [Galicki 89] [Jacak 91]. W najprostszych przypadkach jest to poszukiwanie odległości pomiędzy dwiema kulami, co jest zadaniem trywialnym, lub między dwoma odcinkami, czy w bardziej skomplikowanym przypadku, między wielościanami. Zwykle postępuje się w ten sposób, że robot oraz przeszkody modelowane są określonymi bryłami, rzadziej natomiast, zamiast modelować robot bryłami, definiuje się na nim odpowiednią liczbę punktów kontrolnych, które potencjalnie najszybciej mogą kolidować z przeszkodami [Nakamura, Hanfusa, Yoshikawa 1987] [Mayorga, Ma, Wong 1993] [Böke, Miss, Deeg 1993]. Podczas ruchu robota są sprawdzane następnie wszystkie pary obiektów należących do robota i przeszkód, dla znalezienia najmniejszej odległości i punktu krytycznego, bądź tzw. wektora potencjalnej kolizji (najkrótszy wektor łączący robota i przeszkodę) w układzie: robot–przeszkody. Zbędne jest przy tym sprawdzanie par obiektów modelujących statyczne przeszkody. Kolizji pomiędzy ramionami robota można uniknąć, monitorując współrzędne konfiguracyjne robota na zbliżanie się do niedozwolonych konfiguracji ramion. Proces obliczeń, podczas którego są przeszukiwane wszystkie pary obiektów należących do robota i przeszkód, dla znalezienia najmniejszej odległości, jest intensywny obliczeniowo i czasochłonny. Dlatego też poszukuje się metod upraszczających i przyspieszających.



Rys. 4.3. Obszar widzenia LASAR™-kamery [Christensen 1993]

Fig. 4.3. Vision space of the LASAR™-camera [Christensen 1993]

Przyspieszenie procesu obliczeń ma tu cztery aspekty, które wynikają ze względu na czas obliczeń i metodę rozwiązania [Henrich, Cheng 1992]. Jeśli idzie o czasy, to obliczenia mogą być dokonywane *off-line* bądź *on-line*. Jeśli idzie natomiast o metodę, to przyspieszenie obliczeń można uzyskać na drodze redukcji liczby obiektów bądź redukcji czasu obliczania pojedynczej odległości między obiektami. Przykłady metod przyspieszających rozwiązanie problemu omijania przeszkód dla każdego z czterech aspektów przedstawia tabela 4.1 [Henrich, Cheng 1992].

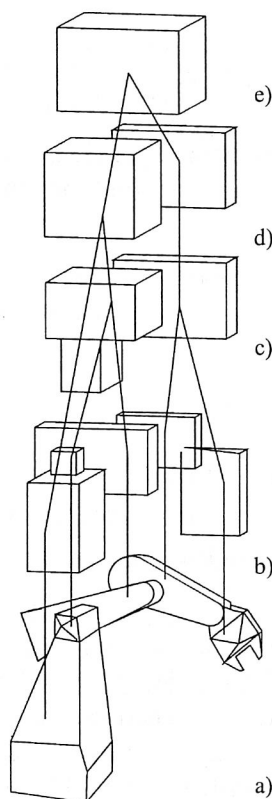
Tabela 4.1. Przykłady metod przyspieszenia obliczeń odległości między robotem a przeszkodami [Henrich, Cheng 1992]

Table 4.1. Examples of methods accelerating the calculation of distances between robot and obstacles [Henrich, Cheng 1992]

	off-line	on-line
Liczba obliczeń	Modelowanie przeszkód za pomocą tzw. statycznej hierarchii, gdzie na poszczególnych jej poziomach bryły aproksymujące przeszkody są łączone i aproksymowane jedną bryłą	Obliczanie odległości nie w każdym takcie ruchu, ale jej aktualizowanie w zależności od wystąpienia ustalonych warunków
Trudność obliczeń	Modelowanie jak najprostszymi bryłami, np. kulami czy prostopadłościanami o ścianach równoległych do osi bazowego układu współrzędnych	Użycie wyników poprzednich obliczeń, np. z poprzedniego kroku interpolacyjnego

Na Uniwersytecie w Karlsruhe opracowano metodę tzw. dynamicznej hierarchii, która przyspiesza obliczanie wektora potencjalnej kolizji we wszystkich wymienionych aspektach i nadaje się do zastosowania w czasie rzeczywistym, a więc *on-line*, w sterowaniu robotem [Henrich, Cheng 1992]. Metoda ta polega na hierarchicznym zamodelowaniu przeszkód i robota prostymi bryłami, możliwie prostopadłościanami o ścianach równoległych do osi bazowego układu współrzędnych. Na poszczególnych poziomach tej hierachii dwie sąsiednie bryły modelujące, np. robota, są łączone i aproksymowane jedną bryłą (rys. 4.4).

Na najwyższym poziomie robot jest np. reprezentowany przez jedną tylko bryłę. Modelowanie robota łączy się przy tym z przygotowaniem *off-line* pewnej liczby hierarchii dla różnych konfiguracji ramion. Podczas obliczeń wektora potencjalnej kolizji między robotem i przeszkodą, rozważa się najpierw najwyższy poziom hierarchii, czyli tylko dwie bryły. Na każdym z poziomów obliczany jest wektor potencjalnej kolizji między wszystkimi możliwymi parami brył reprezentujących robot i przeszkodę. Jeśli długość tego wektora, czyli odległość pomiędzy bryłami, jest mniejsza niż ustalony próg, to obie te bryły są rozkładane na bardziej dokładną reprezentację na niższym poziomie i tu obliczany jest znowu wektor potencjalnej kolizji. Obliczenia kończą się



Rys. 4.4. Modelowanie robota Puma 260 za pomocą dynamicznej hierarchii o 5 poziomach [Henrich, Cheng 1992]: wypukłymi wielościanami (a), prostopadłościanami o ścianach równoległych do osi bazowego układu współrzędnych (b)–(e)

Fig. 4.4. Modelling of robot Puma 260 using the dynamic hierarchy method with 5 levels [Henrich, Cheng 1992]: using the convex polyhedrons (a), using the rectangular prisms with the faces parallel to the base coordinate system (b)–(e)

w momencie, gdy najmniejsza odległość na którymś z poziomów jest większa niż ustalony próg, lub gdy na najniższym poziomie (najbardziej dokładnie modelującym) odległość ta jest mniejsza, co oznacza kolizję.

4.2.5. Zamrożenie ruchu w wybranych połączeniach ruchowych

Zamrożenie ruchu w wybranych połączeniach ruchowych oznacza, że wartość wychyleń w tych połączeniach jest stała podczas realizacji głównego zadania prowadzenia członu roboczego po zadanym torze ruchu. Oznacza to więc odebranie robotowi wybranych stopni swobody i zmniejszenie jego redundantności. W algorytmie sterowania można to osiągnąć przez odpowiednie sformułowanie wielowymiarowej funkcji $\Phi(\theta)$ opisującej więzy. Jeśli przykładowo zamrożenie dotyczy dwóch połączeń ruchowych o numerze 2 i 3, to

$$\Phi(\theta) = \begin{bmatrix} \theta_2 - \theta_2^0 \\ \theta_3 - \theta_3^0 \end{bmatrix} = 0,$$

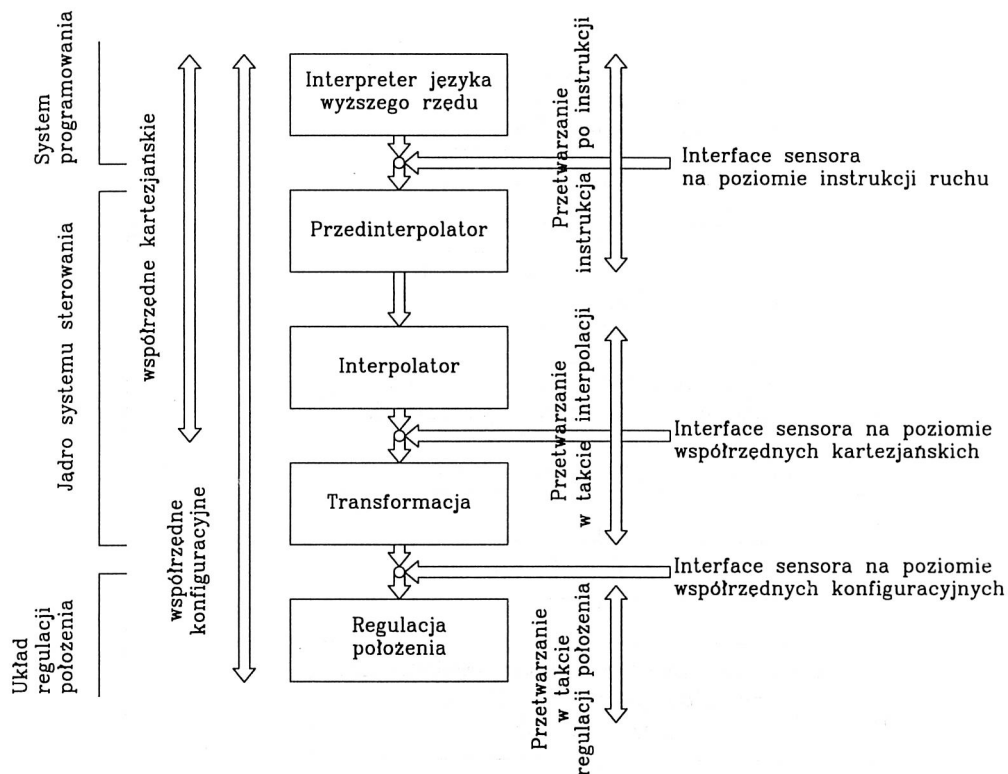
gdzie θ_3^0 i θ_2^0 są stałymi wartościami wychyleń, które mają być utrzymane podczas ruchu robota. Równanie (39a) i zarazem (40b), przyjmie wtedy postać

$$\begin{bmatrix} \mathbf{J}_{\text{robot}}(\theta_{\text{stare}}) & & \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \Delta\theta = - \begin{bmatrix} f(\theta_{\text{stare}}) - \mathbf{x} \\ \theta_{\text{stare},2} - \theta_2^0 \\ \theta_{\text{stare},3} - \theta_3^0 \end{bmatrix}.$$

4.3. Integracja modułu transformacji odwrotnej w sterowaniu robota o redundantnym łańcuchu kinematycznym

Na rysunku 4.5 przedstawiono przepływ danych i podstawowe etapy ich przetwarzania w sterowaniu robota. Program dla robota, napisany w języku wyższego rzędu (bądź powstały z niego kod pośredni), jest podczas jego uruchomienia najpierw analizowany przez interpreter. Interpreter generuje na podstawie programu sekwencję bloków funkcyjnych, czyli najmniejszych pojedynczych poleceń, których przykładami mogą być: polecenie ruchu do następnego punktu pośredniego wynikającego z typu interpolacji, sygnał do sterownika logicznego lub przeczytanie kanału wejściowego. Te poszczególne bloki funkcyjne są następnie analizowane w tzw. przedinterpolatorze, gdzie uwzględniane są różnego rodzaju ograniczenia, a także obliczane są parametry właściwej interpolacji. Zadaniem interpolatora jest obliczenie, dla aktualnego bloku funkcyjnego ruchu, nowego położenia i orientacji członu roboczego robota w przestrzeni kartezjańskiej, z uwzględnieniem zaprogramowanej prędkości na torze. Zadanie to jest wykonywane w każdym takcie interpolacyjnym, zwykle co 10–100 ms. Kolejny etap przetwarzania danych to obliczenie wychyleń w połączeniach ruchowych, czyli współrzędnych konfiguracyjnych, które odpowiadają nowemu położeniu i nowej orientacji członu roboczego, otrzymanym z interpolatora. Etap ten jest nazywany transformacją odwrotną. Zadane wychylenia w połączeniach ruchowych stanowią z kolei wielkości wejściowe na układy regulacji położenia w poszczególnych osiach robota. Na rysunku 4.5 zaznaczono również miejsce, w których zwykle może wystąpić dołączenie do głównego przepływu danych sygnałów pochodzących od sensorów.

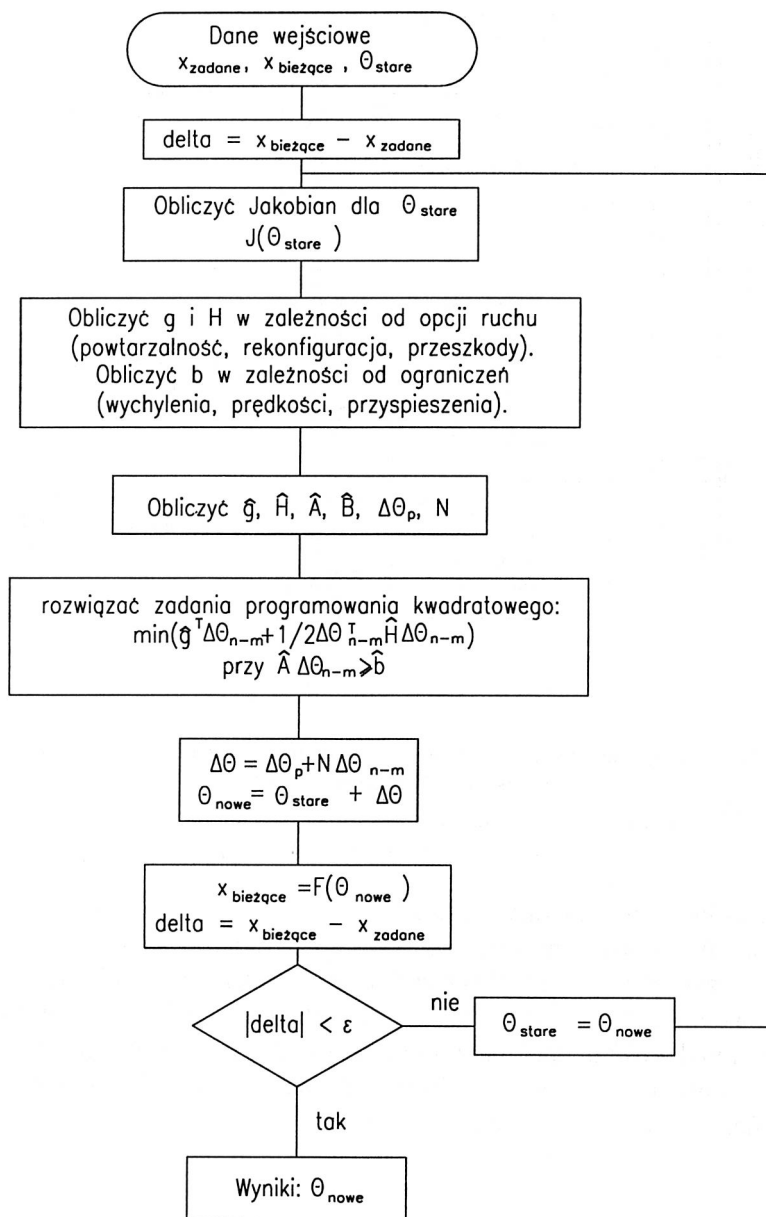
Opisany w rozdziale 4.2 algorytm sterowania robota o redundantnym łańcuchu kinematycznym przedstawia moduł transformacji odwrotnej. Aby móc go zintegrować w sterowaniu robota należy ściśle określić jego dane wejściowe i wyjściowe, tak aby mógł on być wprzęgnięty w podstawowy tok przetwarzania danych w sterowaniu po-



Rys. 4.5. Przepływ danych i podstawowe etapy ich przetwarzania w sterowaniu robota
[Pritschow, Gronbach, Uhl 1993]

Fig. 4.5. Data flow and main stages of their processing in the robot control system
[Pritschow, Gronbach, Uhl 1993]

między interpolatorem i układami regulacji poszczególnych osi. Należy także przewidzieć sposób sprzężenia danych od sensora, np. w postaci skanera laserowego, lub danych pochodzących z systemu CAD; zwłaszcza w zadaniu omijania przeszkód. Przedstawiony na rysunku 4.6 schemat blokowy uwzględnia wymienione powyżej aspekty, a także daje możliwość uzupełniania biblioteki zadań dodatkowych i może stanowić podstawę do oprogramowania modułu transformacji odwrotnej dla robota o redundantnym łańcuchu kinematycznym.



Rys. 4.6. Schemat blokowy algorytmu transformacji odwrotnej dla robota o redundantnym łańcuchu kinematycznym

Fig. 4.6. Block diagram of the inverse transformation algorithm for the robot with redundant kinematics

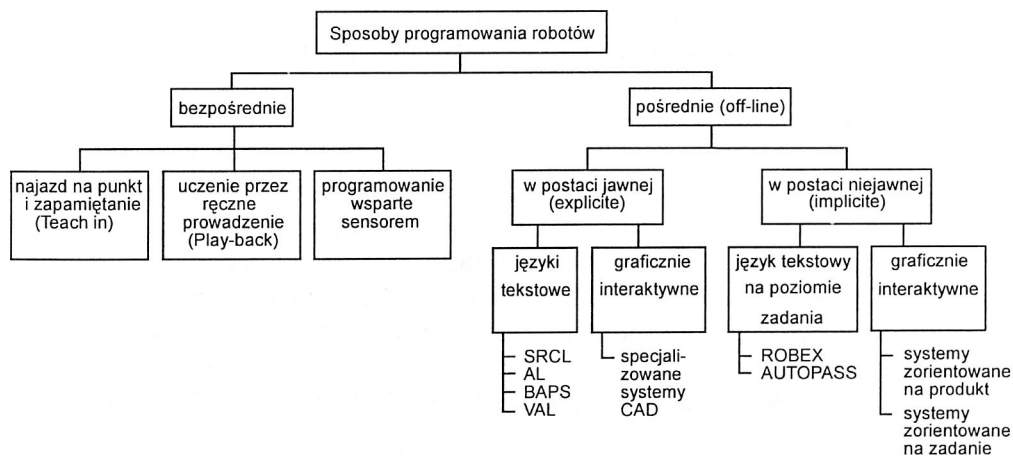
5. PROGRAMOWANIE ROBOTÓW O REDUNDANTNYCH ŁAŃCUCHACH KINEMATYCZNYCH

5.1. Programowanie robotów przemysłowych

Istnieje stosunkowo duża liczba metod programowania robotów przemysłowych, jeśli się ją porówna z dość ograniczoną liczbą sposobów programowania obrabiarek sterowanych numerycznie. Rysunek 5.1 przedstawia podstawowe kategorie metod programowania robotów.

Do metod programowania bezpośredniego (*on-line*) należą metody *teach-in* (uczenie pojedynczych pozycji), *play-back* (uczenie sekwencji pozycji na torze) i programowanie wsparte sensorem. Ich wspólną właściwością jest to, że do definiowania położenia członu roboczego, czy konfiguracji łańcucha kinematycznego, używa się robota. Wadami metod *teach-in* i *play-back* jest duża pracochłonność programowania i wyłączenie urządzeń z produkcji na czas jego trwania. To przyczyniło się do powstania metod programowania wspartych sensorem, których przykładem może być prowadzenie członu roboczego po konturze przedmiotu za pomocą sensora geometrii [Pritschow, Frager, Schumacher, Wieland 1989] [Pritschow 1995].

Do programowania robotów do zadań, które ze względu na stopień swojej złożoności stawiają większe wymagania, zyskały popularność tekstowe języki programowania. Języki te powstawały zazwyczaj na bazie języków programowania komputerów, które zmodyfikowano lub uzupełniono o konstrukcje niezbędne do zaprogramowania ruchu, położenia w przestrzeni i konfiguracji, a także do komunikacji z innymi urządzeniami



Rys. 5.1. Podstawowe kategorie metod programowania robotów [Schumacher 1991]

Fig. 5.1. Main categories of robot programming methods [Schumacher 1991]

i sensorami poprzez różnego rodzaju kanały komunikacyjne. Do zaprogramowania położeń pewnej liczby punktów sięga się przy tym jednak do możliwości programowania *teach-in*. Systemy programowania przedstawiają więc najczęściej metody hybrydowe. Powstała bardzo duża liczba języków programowania, które zostały wprowadzone przez producentów robotów czy ich sterowań, bądź powstały w ramach przedsięwzięć badawczych. Do tworzenia programu w postaci tekstowych języków programowania tworzone są mniej lub bardziej kompleksowe systemy programowania, które wszystkie zawierają jednak jako podstawę edytor tekstowy. Bardziej zaawansowane tego typu systemy, zwykle na komputery osobiste, wspierane są techniką okienek i rozwijanych menu, np. w środowisku *MS-Windows* oraz zawierają moduły testowania programu i komunikacji ze sterowaniem robota, jak np. system ROPS (*Robot Off-line Programming System*) pod nazwą *Robotwindows* firmy *Bosch* [Bosch 1989]. Tworzone są również systemy programowania, nazywane niekiedy układami programowania autonomicznego [Craig 1993], które są niezależne od języka programowania. Zwykle zanurzone są one w systemach CAD i umożliwiają zadawanie poleceń ruchu z jednoczesną ich graficzną wizualizacją, co można nazwać programowaniem graficznym, i końcową symulacją [Włoka 1991] [Dillmann, Huck 1991] [Włoka¹ 1992]. Stworzony przebieg ruchu tłumaczony jest następnie na tekstowy język programowania danego robota za pomocą odpowiedniego postprocesora, wybranego z wielu dostępnych w takim systemie. Przykłady takich systemów to CIMSTATION [Craig 1993] i ROBCAD [ROBCAD] [Włoka 1991].

Jakkolwiek dotychczas w przemysłowym zastosowaniu są prawie wyłącznie tekstowe języki programowania, obserwuje się dążenie do tworzenia języków i systemów programowania na poziomie zadania [Warnecke, Schraft 1990] [Craig 1993]. Przykładami mogą tu być system do programowania montażu układów scalonych na płytkach drukowanych [Pritschow, Frager, Schumacher, Wieland 1989] i do programowania montażu wiązki kabli we wtyczki [Warnecke, Schraft 1990].

W przemysłowych zastosowaniach roboty przeważnie występują w ramach tzw. zrobotyzowanych gniazd produkcyjnych i ich sterowania (*RC – Robot Control*) muszą współpracować z innymi maszynami wytwórczymi, jak obrabiarki sterowane numerycznie (*NC – Numerical Control*) i urządzenia sterowane programowalnymi sterownikami logicznymi (*PLC – Programmable Logic Controller*), czy też ze współrzędnościowymi maszynami pomiarowymi (*CMM – Coordinate Measuring Machine*). Niestety rozwój metod programowania tych sterowań następował zupełnie niezależnie. W tabeli 5.1 przedstawiono zgrubną klasyfikację metod programowania sterowań obrabiarek, robotów, sterowników logicznych i współrzędnościowych maszyn pomiarowych, podając stosowne przykłady. Dla ułatwienia i obniżenia kosztów programowania zrobotyzowanych gniazd produkcyjnych pojawiają się dążenia do tworzenia metod jednolitego programowania, na bazie tekstowych języków programowania, wszystkich urządzeń wchodzących w skład gniazda [Pritschow, Frager, Schumacher, Wieland 1989] [Schumacher 1991].

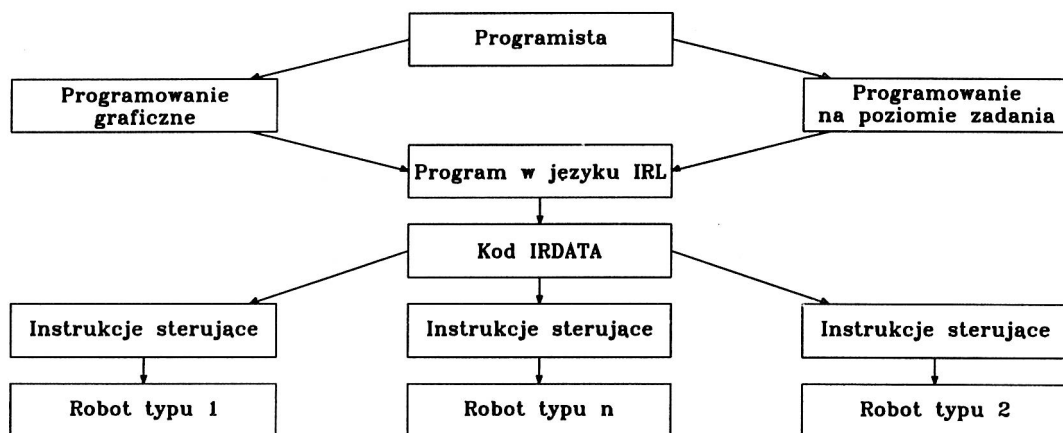
Tabela 5.1. Metody programowania i ich przykłady
Table 5.1. Programming methods and their examples

Metoda	NC	RC	PLC	CMM
Tekstowy język programowania	– DIN 66025	– VAL	– DIN 19239, – STEP 5	– DMIS
Język na poziomie zadania	– EXAPT	– ROBEX (Exapt)	– za pomocą bloków funkcyjnych	– NCMES (Exapt)
Graficzne interaktywne programowanie	– systemy CAD/CAM np. Personal Machinist	– ROBCAD (Tecnomatix)	– plan zestykowy (drabinkowy)	– Quantum (Zeiss), – CIMCMM (Cimlinc)
Programowanie specyficzne dla sterowania	– kopiowanie	– Teach-in, – Play-back, – wsparte sensorem		– Teach-in

Tekstowe języki programowania stanowią także od kilku lat przedmiot intensywnych dążeń normalizacyjnych, zarówno w niektórych krajach, jak i na arenie międzynarodowej, np. w ramach ISO. Motywacją tych dążeń jest wspomniana już z jednej strony powszechność posługiwania się w przemyśle tekstowymi językami programowania, a z drugiej duża liczba różniących się języków. Duża liczba języków oznacza dla przemysłowych użytkowników duże i z pewnością zbędne nakłady, wymaga bowiem utrzymywania wykwalifikowanego personelu dla każdego niemalże typu sterowania i nie pozwala przenosić istniejącego oprogramowania na inny typ sterowania robota. Poza tym tekstowe języki programowania nadają się do rozwiązania dużej liczby klas zadań i choć z pewnością nie są całkowicie uniwersalną metodą programowania, to jednak stanowią wypróbowany kompromis między metodami *teach-in*, *play-back* czy językami niskiego rzędu, a programowaniem graficznym, czy na poziomie zadania.

W Niemczech prace normalizacyjne w dziedzinie programowania robotów rozpoczęto na początku lat osiemdziesiątych i dotyczyły one kodu pośredniego. Pomysł polegał na stworzeniu standardowego interfejsu między systemem bądź językiem programowania a sterowaniem robota i jest analogią do kodu CLDATA dla sterowań numerycznych obrabiarek. Każdy robot, którego sterowanie rozumie standardowy kod pośredni, może być programowany dowolnym językiem bądź systemem programowania dającym po kompilacji ów kod (rys. 5.2).

Prace te zaowocowały powstaniem najpierw tzw. zaleceń VDI 2863 [VDI 2863] pod nazwą proponowanego kodu pośredniego IRDATA (*Industrial Robot Data*), a następnie w roku 1990 udało się zakończyć prace nad normą DIN 66313, również pod nazwą IRDATA [DIN 66313] [DIN 66314]. IRDATA zapoczątkowała od 1988 roku prace



Rys. 5.2. Możliwości programowania przy korzystaniu ze standardów IRDATA/IRL

Fig. 5.2. Programming possibilities using IRDATA/IRL standards

nad stworzeniem międzynarodowej normy pod nazwą ICR (*Intermediate Code for Robots*, ISO Committee Draft 10562.2), która w wielu istotnych fragmentach będzie zawierała części DIN 66313 [Pritschow, Frager 1992]. Jeśli idzie o normalizację języka wyższego rzędu, to w Niemczech opracowano język podobny do Pascala o nazwie IRL (*Industrial Robot Language*). Prace nad pierwszą częścią normy IRL DIN 66312 zakończono w 1992 roku [DIN 66312] [Borgolte 1993]. Język wyższego rzędu jest także przedmiotem prac normalizacyjnych w ramach ISO. Niemiecka norma IRL ma stanowić tu podstawę opracowania języka PLR (*Programming Language for Robots*, ISO Working Draft 11513.1) [Pritschow, Frager 1992] [Borgolte 1993].

Na rysunku 5.2 przedstawiono możliwości programowania przy korzystaniu ze standardów IRDATA/IRL. Kilku producentów wyposażyło już swoje sterowania w interfejs wg roboczej wersji ICR i w język programowania zgodny z IRL [Borgolte 1993].

5.2. Rozszerzenie tekstowego języka programowania dla programowania redundantnych łańcuchów kinematycznych

Dla programowania redundantnych łańcuchów kinematycznych, w ramach niniejszej pracy, rozszerzono tekstowy język programowania podobny do Pascala. Przyczyny bazowania na tekstowych językach programowania wynikają z rozważań w rozdziale 5.1 i można je zebrać w następującą listę [Koch¹ 1995]:

- powszechność stosowania w przemyśle tekstowych języków programowania,
- możliwość rozwiązania za ich pomocą programowania dużej liczby klas zadań (tekstowe języki programowania stanowią wypróbowany kompromis między prostymi i zaawansowanymi metodami),

– są podstawą prób tworzenia systemów jednolitego programowania urządzeń wytwórczych w ramach gniazda zrobotyzowanego,
 – będą stanowiły standardowe metody programowania w postaci języka PLR w ramach ISO i IRL w ramach DIN.

W tabelach od 5.2 do 5.9 zamieszczono podstawowe elementy języka IRL.

Tabela 5.2. Typy danych ogólne

Table 5.2. General data types

•	integer
•	real
•	character
•	string
•	text
•	boolean
•	array
•	record
•	file
•	signal

Tabela 5.3. Typy specyficzne dla robota

Table 5.3. Robot specific types position

•	orientation
•	pose
•	robtargt
•	main_joint
•	add_joint
•	joint

Tabela 5.4. Operatory geometryczne

Table 5.4. Geometrical operators

+	dodawanie
-	odejmowanie
/	dzielenie
*	mnożenie
@	przyrostowa translacja i rotacja

Tabela 5.5. Operatory boolowskie

Table 5.5. Boolean operators

AND	logiczne „i”
OR	logiczne „lub”
NOT	logiczne „nie”
EXOR	logiczna alternatywa wyłączna

W normie języka IRL pojawiły się już konstrukcje uwzględniające redundantne łańcuchy kinematyczne. Przykładem jest typ „robtargt” (tabela 5.3), zawierający pozycję, orientację i informacje o wieloznacznych konfiguracjach, redundantnych łańcuchach kinematycznych czy osiach dodatkowych. Typ ten zdefiniowano obok już istniejących „position” (położenie we współrzędnych kartezjańskich), „orientation” (orientacja we współrzędnych kartezjańskich) i „pose” (pozycja i orientacja).

Poniżej przedstawiono elementy tekstowego języka, które są tak skonstruowane, że mogą stanowić rozszerzenie np. języka IRL, w celu umożliwienia programowania robotów o redundantnych łańcuchach kinematycznych. Zamieszczone przykłady bazują na elementach języka IRL [DIN 66312].

Tabela 5.6. Zasadnicza budowa programu
w języku IRL

Table 5.6. Principal structure of the programme
in the IRL language

```

PROGRAM <name>;

SYSTEM SPECIFICATION <name>;
:
FROM <name> IMPORT <name>;
:
EXPORT <name> TO <name>;
:
IMPORT DATALIST <name>;
:
ROBOT <name> := <robot_name>;
:
STANDARD_IN := <name>;
:
STANDARD_OUT := <name>;
:
CONST
:
TYPE
:
VAR
:
PROCEDURE <name>;
:
BEGIN
:
ENDPROC;
:
BEGIN
:
ENDPROGRAM
:

```

Tabela 5.7. Typy instrukcji w języku IRL

Table 5.7. Instruction types in the IRL language

- przypisywanie wartości
– <object>:=<expression>
- wywołanie procedury
– <name>(<parametr>)
- wartość funkcji
– return <expression>
- sterowanie przebiegiem
– goto
– if then else
– case
– for
– repeat
– while
– wait
– pause
– return
– halt
- polecenia ruchu
– move
– move_inc
- komunikacja
– input
– output
– digital
– analog
- kanały i zbiory
– open
– close
– read
– write

Tabela 5.8. Operatory arytmetyczne

Table 5.8. Arithmetical operators

+	dodawanie
-	odejmowanie
/	dzielenie
*	mnożenie
DIV	dzielenie całkowito-liczbowe
MOD	modulo

Tabela 5.9. Operatory logiczne

Table 5.9. Logical operators

=	równy
>	większy
<	mniejszy
<>	nierówny
>=	większy lub równy
<=	mniejszy lub równy

Dopasowanie do łańcucha kinematycznego (definicja współrzędnych konfiguracyjnych)

Podany przykład definiuje nazwy współrzędnych konfiguracyjnych i informuje zarazem system sterowania, że sterowany łańcuch kinematyczny ma 8 głównych połączeń ruchowych.

Przykład: **TYPE**
 RECORD
 REAL: a1, a2, a3, a4, a5, a6, a7, a8;
 ENDRECORD = MAIN_JOINT;

Utrzymanie wychyleń w połączeniach ruchowych w zadanych granicach

W tym celu należałoby zdefiniować w języku dodatkowe dwa typy specyficzne dla robota: MAIN_JOINT_MAX i MAIN_JOINT_MIN:

Przykład: **TYPE**
 RECORD
 REAL: a1.max, a2.max, a3.max, a4.max, a5.max,
 a6.max, a7.max, a8.max;
 ENDRECORD = MAIN_JOINT_MAX;

 RECORD
 REAL: a1.min, a2.min, a3.min, a4.min, a5.min,
 a6.min, a7.min, a8.min;
 ENDRECORD = MAIN_JOINT_MIN;

Na wstępie programu należy określić granice, definiując zmienne typu MAIN_JOINT_MIN i MAIN_JOINT_MAX i odpowiednio je inicjując. Podczas ruchu wyrażenie kluczowe WITH JOINTLIMIT <zmienna typu MAIN_JOINT_MIN, zmienna typu MAIN_JOINT_MAX> oznaczałoby ruch wg algorytmu z rozdziału 4.2.1.

Przykład: **VAR**
 POSE: poz_1;
 MAIN_JOINT_MIN: jmin;
 MAIN_JOINT_MAX: jmax;
 ...

 MOVE LIN poz 1 WITH JOINTLIMIT jmin, jmax

Utrzymanie prędkości ruchu w połączeniach ruchowych w zadanych granicach

W tym przypadku należałoby postąpić tak samo jak przy utrzymaniu wychyleń połączeń ruchowych w zadanych granicach, definiując w języku dodatkowe dwa typy specyficzne dla robota: MAIN JOINT SPEED MAX i MAIN JOINT SPEED MIN:

Przykład: **TYPE**

RECORD

**REAL: a1.s.max, a2.s.max, a3.s.max, a4.s.max,
a5.s.max, a6.s.max, a7.s.max, a8.s.max;**

ENDRECORD = MAIN JOINT SPEED MAX;

RECORD

**REAL: a1.s.min, a2.s.min, a3.s.min, a4.s.min,
a5.s.min, a6.s.min, a7.s.min, a8.s.min;**

ENDRECORD = MAIN JOINT SPEED MIN;

Na wstępie programu należy określić granice, definiując zmienne typu MAIN_JOINT_SPEED_MIN i MAIN_JOINT_SPEED_MAX i odpowiednio je inicjując. Podczas ruchu wyrażenie kluczowe WITH JOINTSPEEDLIMIT <zmienna typu MAIN_JOINT_SPEED_MIN, zmienna typu MAIN_JOINT_SPEED_MAX> oznaczałoby ruch wg algorytmu z rozdziału 4.2.1.

Przykład: VAR

```
POSE:                                poz_1;
MAIN_JOINT_SPEED_MIN:               jsmin;
MAIN_JOINT_SPEED_MAX:               jsmax;
```

...

MOVE LIN p0z 1 WITH JOINTSPEEDLIMIT jmin, jmax

Łączne uwzględnienie obu ograniczeń musiałoby wyglądać następująco:

Przykład:

```
MOVE LIN poz_1 WITH JOINTLIMIT jmin, jmax  
WITH JOINTSPEEDLIMIT jsmin, jsmax
```

W przypadku, gdy ograniczenia mają być uwzględnione w całych fragmentach programu, celowe jest stworzenie możliwości oznaczenia tego fragmentu za pomocą słów kluczowych: JOINTLIMIT <zmienna, zmienna>, JOINTLIMITEND i odpowiednio JOINTSPEEDLIMIT <zmienna, zmienna>, JOINTSPEEDLIMITEND.

Utrzymanie przyspieszeń w połączeniach ruchowych w zadanych granicach

Na tej samej zasadzie można utrzymać przyspieszenia w połączeniach ruchowych w zadanych granicach tworząc typy MAIN_JOINT_ACCEL_MAX i MAIN_JOINT_ACCEL_MIN oraz wyrażenie kluczowe WITH JOINTACCELLIMIT i JOINTACCELLIMIT / JOINTACCELLIMITEND.

Powtarzalność ruchu na zamkniętym torze

W tym przypadku proponuje się zaznaczenie zamkniętego toru ruchu słowami kluczowymi: CYCLICSTART i CYCLICEND.

Przykład: **VAR**

POSE: poz_0, poz_1, poz_2, poz_3, poz_p;

...

MOVE LIN poz_0;

CYCLICSTART;

MOVE LIN poz_1, poz_2, poz_3;

MOVE CIRCLE poz_p, poz_0;

CYCLICEND;

W momencie wystąpienia słowa kluczowego CYCLICSTART konfiguracja przyjęta w momencie osiągnięcia punktu poz_0 staje się konfiguracją początkową, do której robot powinien powrócić przy ponownym dojściu do tego punktu wg algorytmu opisanego w rozdziale 4.2.2. Słowo kluczowe CYCLICEND wyłącza działanie algorytmu. W algorytmie opisanym w rozdziale 4.2.2 występuje diagonalna macierz wag, której wyrazy domyślnie ustawione są wg wzoru $w_{ii} = n - i + 1$ i mogą być zmieniane poza właściwym programem, przy parametryzowaniu systemu, lub w module, w którym ustalone są parametry systemu.

Zmiana konfiguracji z zachowaniem położenia i orientacji członu roboczego

W tym celu proponuje się polecenie RECONFIGURATION <punkt we współrzędnych konfiguracyjnych>, które spowoduje zmianę konfiguracji robota bez naruszania położenia i orientacji członu roboczego wg algorytmu z rozdziału 4.2.3.

Przykład: **VAR**

MAIN_JOINT: konf_1;

...

RECONFIGURATION konf_1;

Zgodnie z tym algorytmem, jeśli współrzędne konfiguracyjne dane zmienną konf_1 nie odpowiadają aktualnym położeniu i orientacji członu roboczego ($x_{\text{aktualne}} \neq f(\text{konf_1})$),

to zmiana konfiguracji polega na możliwie największym, w sensie ważonego kryterium kwadratowego, zbliżeniu się do zadanej konfiguracji `konf_1`.

Zmiana konfiguracji podczas ruchu członu roboczego

W tym przypadku chodzi o rozwiązanie problemu ruchu robota, gdy wskazane jest, aby jego łańcuch kinematyczny przyjmował na zaplanowanym torze z góry zaplanowane (w systemie symulacji bądź nauczone) konfiguracje. Jeśli przyjęcie tych konfiguracji nie jest możliwe, to chodziłoby tu o możliwie największe zbliżenie do nich. Proponowana komenda `WITH RECONFIGURATION` <punkt we współrzędnych konfiguracyjnych> powodowałaby zadziałanie algorytmu jak przy włączeniu opcji powtarzalności ruchu, z tą różnicą, że konfiguracja łańcucha kinematycznego dążyłaby nie do początkowej konfiguracji, ale do wskazanej podanym punktem we współrzędnych konfiguracyjnych.

```
Przykład:  VAR
            MAIN_JOINT:      konf_1, konf_2, konf_3;
            POSE:            poz_1, poz_2, poz_3;
            ...
            MOVE LIN poz_1 WITH RECONFIGURATION konf_1;
            MOVE LIN poz_2 WITH RECONFIGURATION konf_2;
            MOVE LIN poz_3 WITH RECONFIGURATION konf_3;
```

Korzystając z komendy definiującej tor ruchu poprzez podanie kolejnych punktów, powyższy przykład można sformułować następująco:

```
Przykład:  MOVE LIN PATH (poz_1, poz_2, poz_3) WITH
            RECONFIGURATION (konf_1, konf_2, konf_3);
```

Omijanie przeszkód

W tym celu fragmenty, w których ruch jest narażony na kolizję, muszą zostać oznaczone słowami kluczowymi `OBSTACLE` <name>, `OBSTACLEEND` <name>.

```
Przykład:  MOVE LIN poz_0;
            OBSTACLE ściana_1;
            MOVE LIN poz_1, poz_2, poz_3;
            MOVE CIRCLE poz_p, poz_0;
            OBSTACLEEND ściana_1;
```

Poprzez nazwę przeszkody proponuje się zidentyfikować, czy zdefiniować przeszkodę. To definiowanie przeszkody następowałoby poza programem, tak jak to jest w przypadku punktów *teach-in*. W tej jednak sytuacji nie chodzi o uczenie, ale o określenie

przeszkody lub grupy przeszkód w systemie *CAD*. Stanowiłoby to dane do przeprowadzania współbieżnie z uruchomionym programem dla robota, obliczeń punktu krytycznego $\mathbf{x}_k$ na robocie oraz kierunku $\mathbf{v}$ oddalającego go od przeszkody (patrz rozdział 4.2.4). Obliczenia te bazowałyby w prostych przypadkach na algorytmach liczenia odległości między figurami czy bryłami geometrycznymi, a w bardziej skomplikowanych przypadkach na algorytmie dynamicznej hierachii (patrz rozdział 4.2.4) lub na danych z obrazów 3D, dostarczonych przez skaner laserowy (patrz rozdział 4.2.4). Obliczone wielkości $\mathbf{x}_k$ i $\mathbf{v}$ dostarczane byłyby podstawowemu procesowi obliczeń transformacji odwrotnej wg schematu blokowego z rysunku 4.6.

Przy użyciu poleceń **OBSTACLE <name> / OBSTACLEEND <name>** na bardzo małym zamkniętym torze ruchu można dla danego położenia i orientacji członu roboczego znaleźć konfigurację, która najlepiej omija przeszkodę o nazwie identyfikacyjnej <name>. Może to być alternatywny sposób planowania *off-line*, czy wpływania *on-line* na ruch łańcucha kinematycznego robota. Najlepiej omijające przeszkodę konfiguracje mogą ewentualnie posłużyć do wpływania na ruch łańcucha kinematycznego dzięki komendzie **WITH RECONFIGURATION <punkt we współrzędnych konfiguracyjnych>**.

Zamrożenie ruchu w wybranych połączeniach ruchowych

W tym celu proponuje się polecenie **FREEZESTART <numery połączeń ruchowych>**, które spowoduje utrzymanie wychyleń w wymienionych połączeniach w bezruchu, aż do odwołania słowem kluczowym **FREEZEEND**.

Przykład: **MOVE LIN poz_0;**
 FREEZESTART 1, 2;
 MOVE LIN poz_1, poz_2, poz_3;
 MOVE CIRCLE poz_p, poz_0;
 FREEZEEND;

Powyższy przykład może się odnosić do robota mobilnego, którego pierwsze dwa połączenia ruchowe translacyjne realizują ruch platformy robota na płaszczyźnie. Wybrany fragment ruchu członu roboczego całego robota mobilnego, zaprogramowany pomiędzy poleceniami **FREEZESTART / FREEZEEND**, będzie zrealizowany bez przemieszczania się platformy po powierzchni.

Łączne uruchomienie różnych opcji ruchu redundantnego łańcucha kinematycznego

Możliwe jest uruchomienie kilku zaproponowanych opcji ruchu jednocześnie. Poniżej zamieszczony przykład ilustruje taką sytuację.

Przykład: VAR

```

MAIN_JOINT:  konf_0, konf_1, konf_2, konf_3;
POSE:        poz_p, poz_0, poz_1, poz_2, poz_3;
MAIN_JOINT_MAX:      jmax;
MAIN_JOINT_MIN:      jmin;
MAIN_JOINT_SPEED_MAX:  jsmax;
MAIN_JOINT_SPEED_MIN:  jsmin;

```

...

```

jmax:=(180., 180., 180., 180., 90., 270., 180., 180.);
jmin:=(-180, -180, 0., 0., -90., -90., -180., -180.);
jsmax:=(200., 200., 200., 200., 200., 200., 200., 200.);
jsmin:=(-200., -200., -200., -200., -200., -200., -200., -200.);

```

```

JOINTLIMIT jmin, jmax;
JOINTSPEEDLIMIT jsmin, jsmax;

```

```

MOVE LIN poz_0 WITH RECONFIGURATION konf_0;
OBSTACLE ściana_1;
  MOVE LIN poz_1;
  RECONFIGURATION konf_1;
  FREEZESTART 1, 2;
  CYCLICSTART
    OBSTACLE okno;
    MOVE LIN poz_2;
    MOVE CIRCLE poz_p, poz_3 WITH
      RECONFIGURATION konf_p, konf_3;
    MOVE LIN poz_2, poz_1;
    OBSTACLEEND okno;
  CYCLICEND;
  FREEZEEND;
OBSTACLEEND ściana_1

JOINTLIMITEND;
JOINTSPEEDLIMITEND;

```

W przykładzie tym na konfigurację łańcucha kinematycznego podczas zaprogramowanego ruchu wg interpolacji kołowej od punktu poz_2 do punktu poz_3 poprzez punkt pośredni poz_p mają jednocześnie wpływ następujące opcje:

– w punkcie pośrednim poz_p łańcuch kinematyczny przyjmuje konfigurację konf_p (ewentualnie jej bliską), a w punkcie poz_3 konfigurację konf_3 (ewentualnie jej bliską),

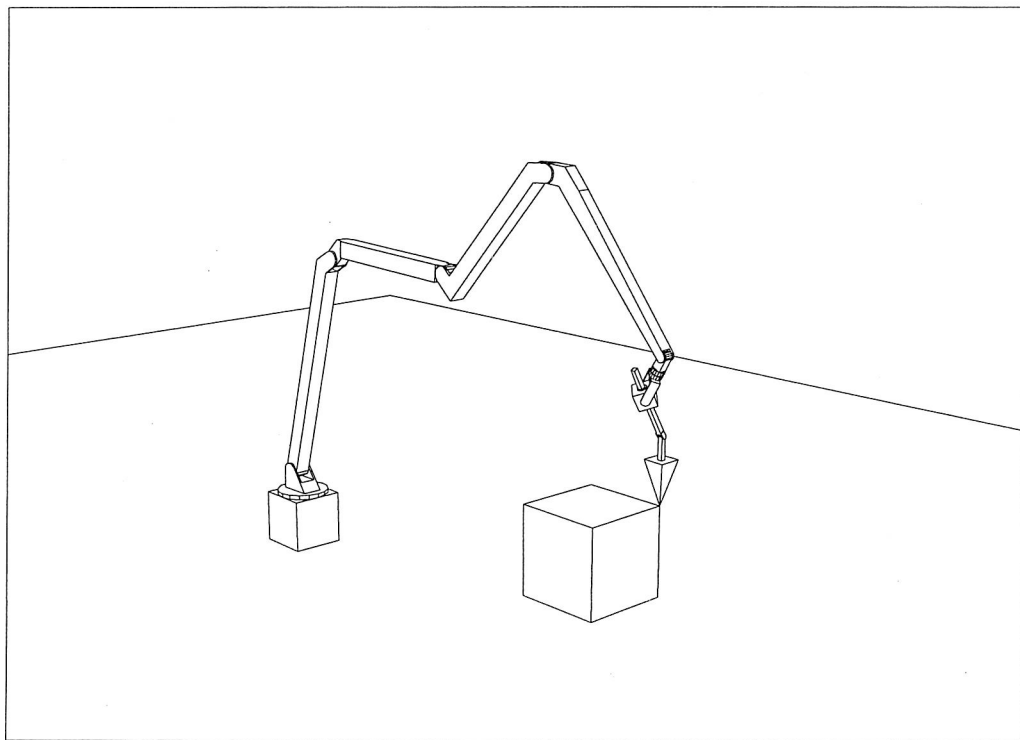
- łańcuch kinematyczny stara się ominąć przeszkody o nazwach okno i ściana_1,
- połączenia ruchowe pierwsze i drugie są w bezruchu, a wartości ich wychyleń równają się wartościom, które zostały przyjęte przez łańcuch kinematyczny w momencie wystąpienia polecenia FREEZESTART.
- wychylenia w połączeniach ruchowych i ich prędkości utrzymywane są w zadanych granicach.

Opcja powtarzalności ruchu jest automatycznie wyłączana podczas rozpatrywanego ruchu wg interpolacji kołowej. Ruch ten bowiem jest pod wpływem konfiguracji `konf_p` i `konf_3`. Natomiast wykonywane ruchy wg interpolacji liniowej, przed i po ruchu wg interpolacji kołowej, są pod wpływem konfiguracji `konf_1` z początku włączenia opcji ruchu cyklicznego.

6. STEROWANIE 11-PRZEGUBOWYM ROBOTEM

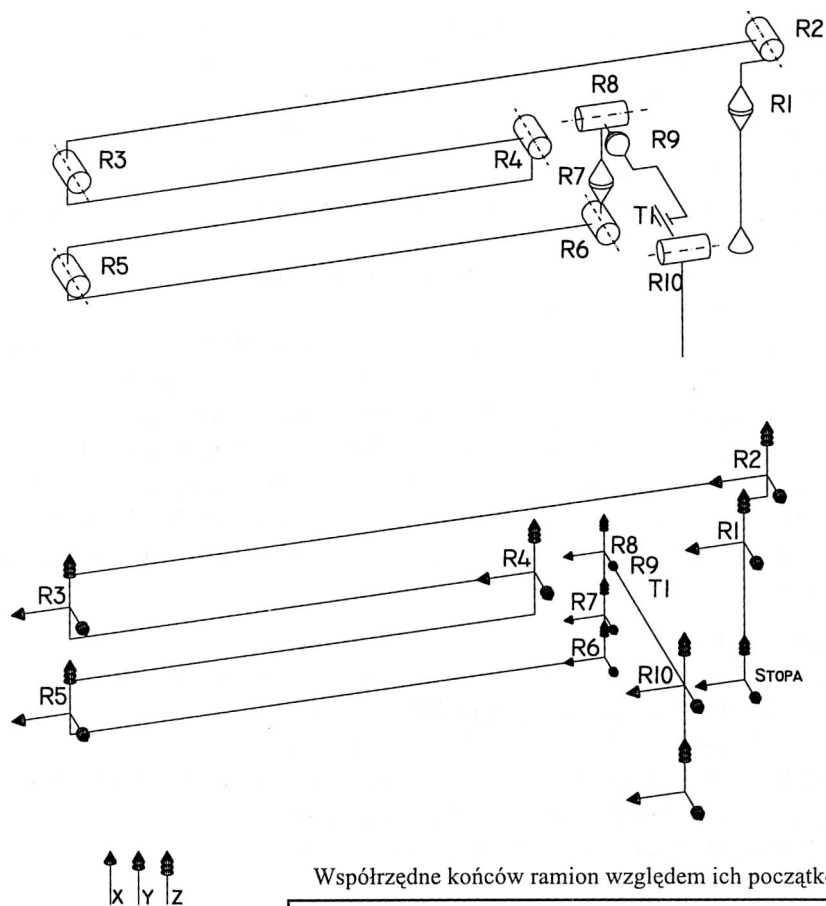
Testowania opracowanego algorytmu sterowania dokonano za pomocą stworzonego w Instytucie Technologii Maszyn i Automatyzacji Politechniki Wrocławskiej systemu RME (Robot Modelling Extension) służącemu planowaniu, programowaniu i symulacji zrobotyzowanych gniazd produkcyjnych [Koch, Kowalczewski, Dybała, Kocełuch, Szypicyn, Dutkowiak 1994]. System RME, będący aplikacją systemu AutoCAD napisaną w języku Metaware High C, umożliwia: symulację pracy robota wg zadanego w kodzie IRDATA programu, graficzną animację, rozpoznawanie kolizji [Koch, Dybała, Kowalczewski 1992] [Koch, Kowalczewski, Kocełuch, Dybała, Szypicyn 1993] i ma ponadto moduł do automatycznego generowania programów dla takich zadań, jak laserowe wycinanie, usuwanie zadziórów czy spawanie [Koch, Kowalczewski, Szypicyn, Kocełuch, Dutkowiak 1994].

Opracowany moduł programu, wg przedstawionego na rys. 4.6 algorytmu sterowania redundantnymi łańcuchami kinematycznymi, został włączony do systemu RME, a następnie za jego pomocą przetestowano sterowanie 11-przegubowym (10 połączeń



Rys. 6.1. 11-przegubowy robot użyty do testowania

Fig. 6.1. 11-joint robot used for the tests



Współrzędne końców ramion względem ich początków [m]

Nr ramienia	X	Y	Z
1	0	0	1,3
2	-0,2	0	0,6
3	6,0	0	-0,3
4	-4,0	0	-0,3
5	4,0	0	-0,7
6	-4,6	0	-0,2
7	0	0	0,4
8	0	0	0,6
9	0	0	0
10	0	0	0
11	0	1,0	0
człon roboczy	0	0	-1,5
położenie TCP	1,2	1,0	-0,1

Rys. 6.2. Schemat kinematyczny 11-przegubowego robota użytego do testowania

Fig. 6.2. Kinematics of 11-joint robot used for the tests

obrotowych i 1 połączenie wysuwne) stacjonarnym robotem, którego model geometryczny przedstawia rys. 6.1, a jego łańcuch kinematyczny rys. 6.2.

Przetestowano następujące opcje sterowania ruchem:

- utrzymanie wychyleń w połączeniach ruchowych w zadanych granicach,
- utrzymanie prędkości ruchu w połączeniach ruchowych w zadanych granicach,
- utrzymanie przyspieszenia ruchu w połączeniach ruchowych w zadanych granicach,
- zapewnienie powtarzalności ruchu,
- zamrożenie ruchu w wybranych połączeniach ruchowych,
- omijanie przeszkód.

Testowany tor ruchu dla pierwszych pięciu opcji sterowania stanowił kwadrat (o boku 2 m), którego każdy bok był liniowo interpolowany (rys. 6.3). Podczas ruchu wzdłuż pierwszego i czwartego boku nie było zmiany orientacji członu roboczego, wystąpiła dopiero podczas ruchu wzdłuż drugiego i trzeciego boku. Ponadto w trzecim wierzchołku człon roboczy robota zmieniał wyłącznie orientację bez ruchu translacyjnego. Tak opisana trajektoria została zaprogramowana w języku BAPS (język programowania stworzony przez firmę Bosch), a wersja niemiecka tego programu przedstawia się następująco:

PROGRAMM DEN1

FAHRE LINEAR NACH (11500,1000,3500,0,0,0) ;dojazd do pierwszego wierzchołka

FAHRE LINEAR NACH (9500,1000,3500,0,0,0)

FAHRE LINEAR NACH (9500,-1000,3500,-85,0,0)

FAHRE LINEAR NACH (9500,-1000,3500,0,85,0) ;zmiana orientacji w 3 wierzchołku

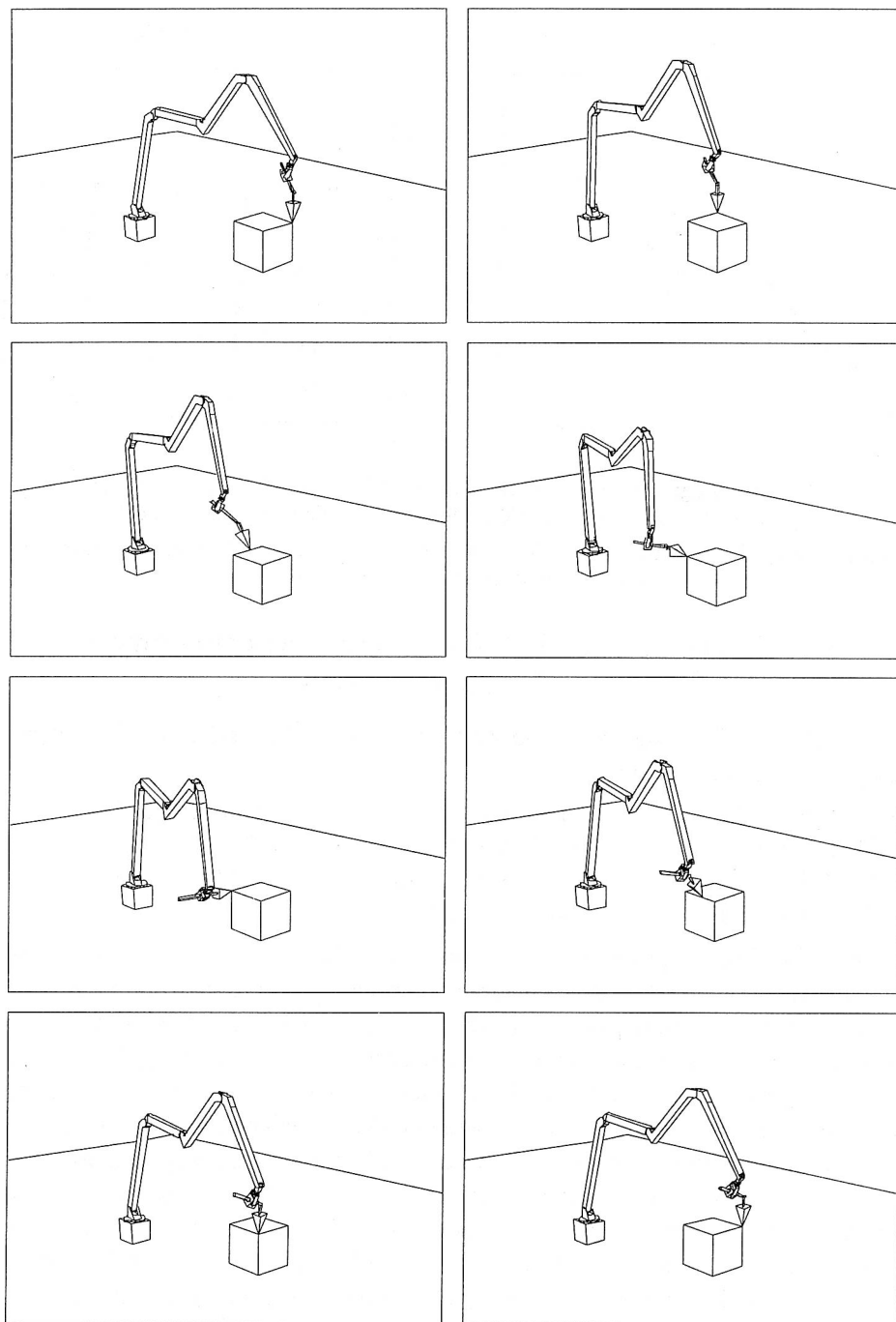
FAHRE LINEAR NACH (11500,-1000,3500,0,0,0)

FAHRE LINEAR NACH (11500,1000,3500,0,0,0)

HALT

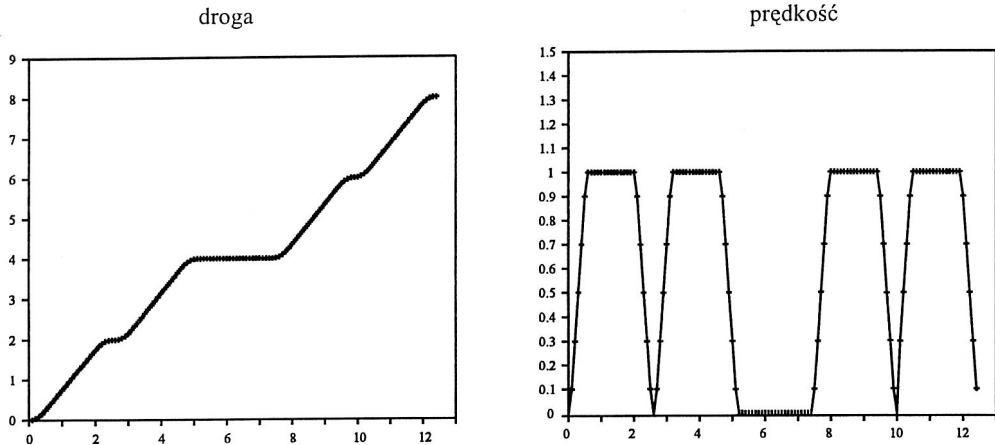
ENDE

Kod IRDATA, będący wynikiem kompilacji tego programu, został zadany do systemu symulacji RME, a poprawność wykonania programu, w tym również poprawność algorytmu sterowania redundantnymi łańcuchami kinematycznymi, potwierdzają zarówno graficzna animacja (rys. 6.3), jak i zarejestrowane wykresy przebiegu drogi i prędkości członu roboczego (rys. 6.4).



Rys. 6.3. Kolejne fazy symulowanego ruchu 11-przegubowego robota na trajektorii o kształcie kwadratu

Fig. 6.3. Simulation of motion of 11-joint robot on a square path



Rys. 6.4. Przebieg drogi [m] i prędkości [m/s] członu roboczego robota w czasie 12,5 sekund, podczas jego ruchu po zamkniętym torze o kształcie kwadratu z rys. 6.3

Fig. 6.4. Path [m] and speed [m/s] of the robot end-effector during 12,5 seconds while moving on the closed square path from fig. 6.3

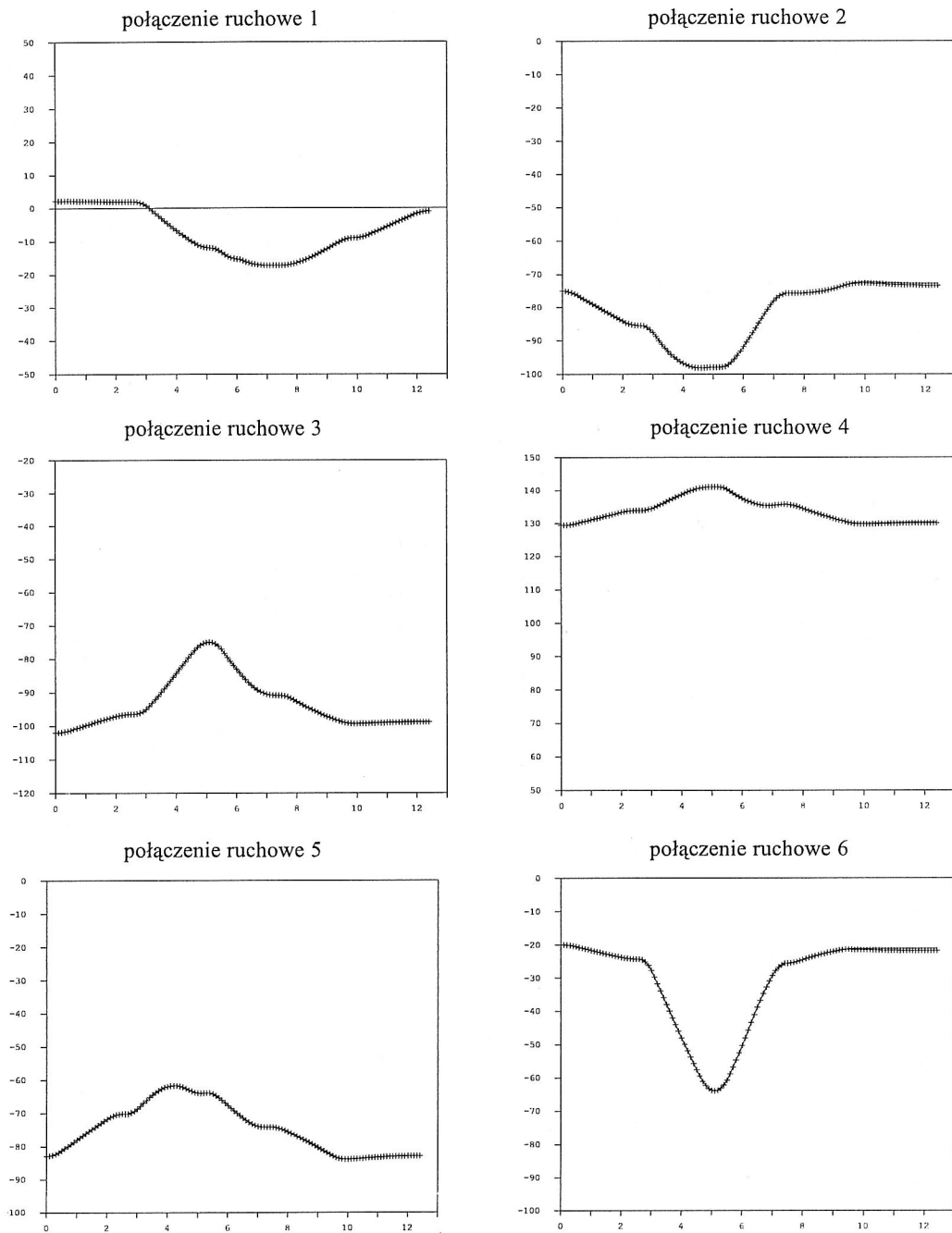
6.1. Utrzymanie wychyleń w połączeniach ruchowych w zadanych granicach

Dla każdego z 11 połączeń ruchowych określono jako podstawowe następujące dolne i górne granice wychyleń:

$$\begin{array}{lll}
 -180^\circ \leq \theta_1 \leq 180^\circ & -180^\circ \leq \theta_5 \leq 0^\circ & -180^\circ \leq \theta_9 \leq 180^\circ \\
 -150^\circ \leq \theta_2 \leq 0^\circ & -180^\circ \leq \theta_6 \leq 0^\circ & -0,5 \text{ m} \leq \theta_{10} \leq 0,5 \text{ m} \\
 -180^\circ \leq \theta_3 \leq 0^\circ & -180^\circ \leq \theta_7 \leq 180^\circ & -20^\circ \leq \theta_{11} \leq 60^\circ \\
 0^\circ \leq \theta_4 \leq 180^\circ & -60^\circ \leq \theta_8 \leq 60^\circ &
 \end{array}$$

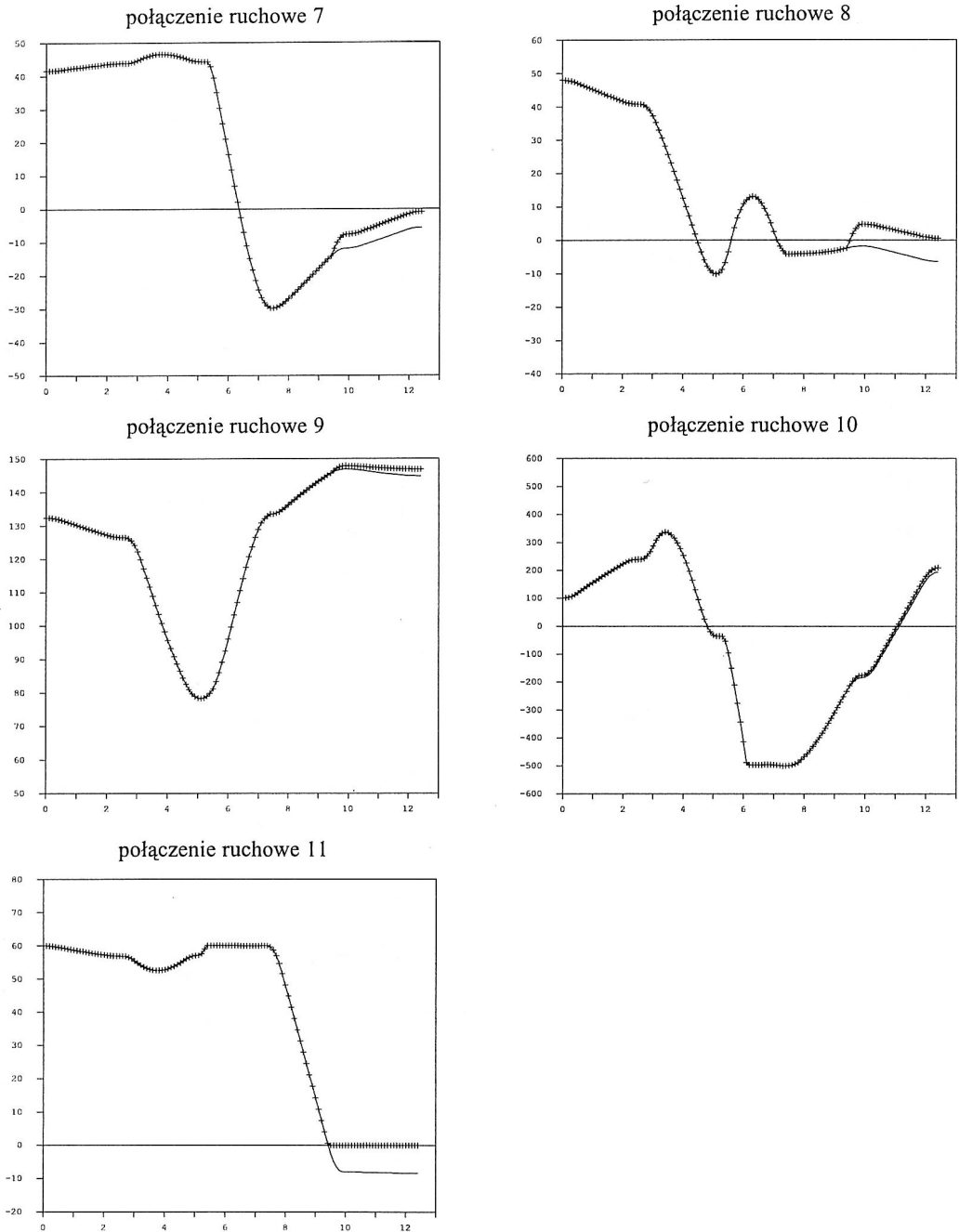
Dla tak określonych granic wychyleń przeprowadzono symulację ruchu robota na testowanym torze ruchu o kształcie kwadratu. Symulację przeprowadzono także po zmodyfikowaniu zestawu granic przez zawężenie obszaru zmienności wychyleń w 11 połączeniu ruchowym poprzez zmianę dolnej granicy z -20° na 0° ($0^\circ \leq \theta_{11} \leq 60^\circ$). Porównanie przebiegów wychyleń oraz prędkości ruchu w poszczególnych połączeniach ruchowych otrzymane z symulacji przedstawiono na kolejnych rysunkach 6.5–6.8.

Podczas ruchu z podstawowym zestawem granic aktywne było tylko górne ograniczenie wychyleń 60° w 11 połączeniu ruchowym (rys. 6.6). Po podniesieniu dolnej granicy wychyleń w tym połączeniu ruchowym na 0° , również i ta granica stała się aktywna i przebieg wychyleńa utrzymany został w przedziale $[0^\circ, 60^\circ]$. Zmiana przebiegu wychyleń w 11 połączeniu ruchowym spowodowała zmiany w przebiegach wychyleń w pozostałych połączeniach ruchowych (największe w 7 i 8 połączeniu ruchowym), tak aby złożenie ruchu wszystkich połączeń ruchowych zapewniło realizację za-



Rys. 6.5. Przebieg wychyleń [w stopniach] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy dwóch różnych ograniczeniach na wychylenie w 11 połączeniu ruchowym: — ograniczenie wychylenia ustawione na $[-20, 60]^\circ$, +++ ograniczenie wychylenia ustawione na $[0, 60]^\circ$

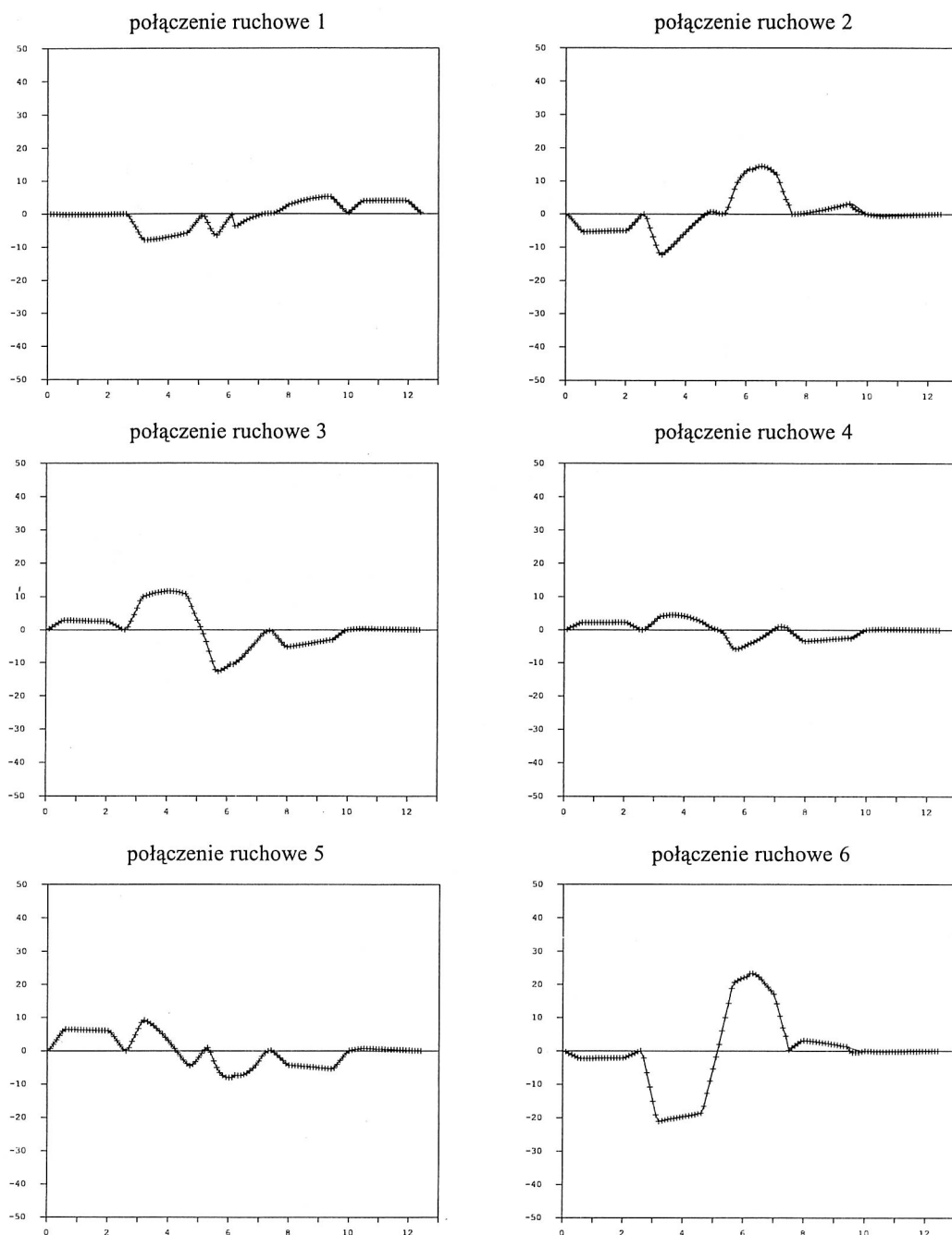
Fig. 6.5. Motion [in deg] of joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3, by different limits for joint 11: — limits set to $[-20, 60]^\circ$, +++ limits set to $[0, 60]^\circ$



Rys. 6.6. Przebieg wychyleń [w stopniach, tylko 10 połączenie ruchowe w mm] w połączeniach ruchowych o numerach 7–11 w czasie 1,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy dwóch różnych ograniczeniach na wychylenie w 11 połączeniu ruchowym: — ograniczenie wychylenia ustawione na $[-20, 60]^\circ$, +++ ograniczenie wychylenia ustawione na $[0, 60]^\circ$

Fig. 6.6. Motion [in deg, only joint 10 in mm] of joints 7–11 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3, by different limits for joint 11:

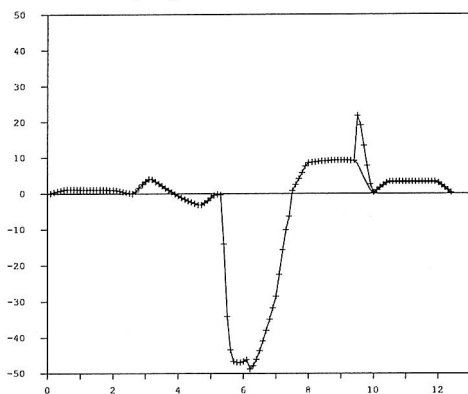
— limits set to $[-20, 60]^\circ$, +++ limits set to $[0, 60]^\circ$



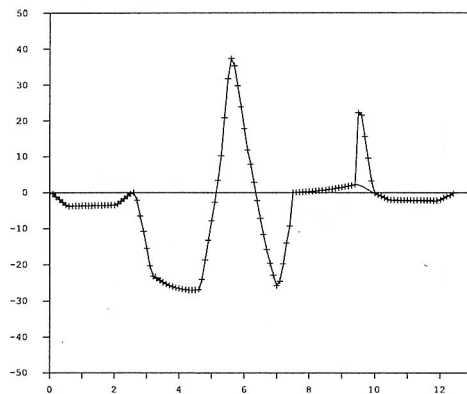
Rys. 6.7. Przebieg prędkości ruchu [w %/s] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy dwóch różnych ograniczeniach na wychylenie w 11 połączeniu ruchowym: — ograniczenie wychylenia ustawione na $[-20, 60]^\circ$, +++ ograniczenie wychylenia ustawione na $[0, 60]^\circ$

Fig. 6.7. Speed [in %/s] of joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3, by different limits for joint 11: — limits set to $[-20, 60]^\circ$, +++ limits set to $[0, 60]^\circ$

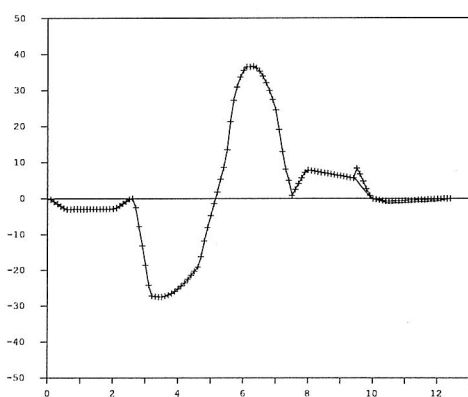
połączenie ruchowe 7



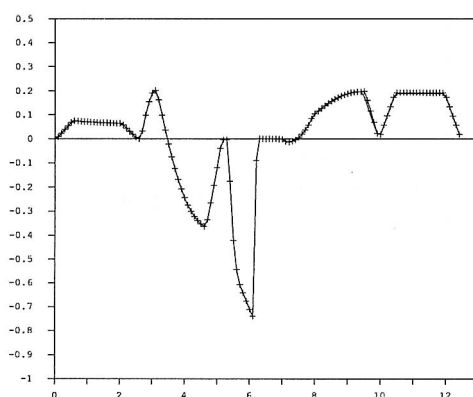
połączenie ruchowe 8



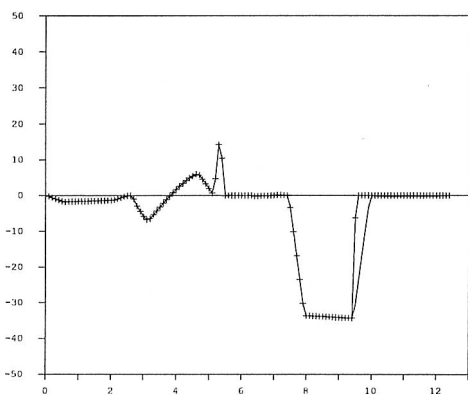
połączenie ruchowe 9



połączenie ruchowe 10



połączenie ruchowe 11



Rys. 6.8. Przebieg prędkości ruchu [w %/s, tylko 10 połączenie ruchowe w m/s] w połączeniach ruchowych o numerach 7–11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy dwóch różnych ograniczeniach na wychylenie w 11 połączeniu ruchowym: — ograniczenie wychylenia ustawione na $[-20,60]^\circ$, +++ ograniczenie wychylenia ustawione na $[0,60]^\circ$

Fig. 6.8. Speed [in %/s, only joint 10 in m/s] of joints 7–11 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3, by different limits for joint 11: — limits set to $[-20,60]^\circ$, +++ limits set to $[0,60]^\circ$

programowanego toru ruchu członu roboczego. Zmiany te można również dostrzec w przebiegach prędkości ruchu w poszczególnych połączeniach ruchowych (rys. 6.7 i 6.8).

Ruch symulowano zadając w algorytmie sterowania, określonym wzorem (40) z rozdziału 4.2, następujące wartości:

$$\mathbf{g} = 0, \quad \mathbf{H} = \text{diag}(11, 10, 9, 8, 7, 6, 5, 4, 3, 5 \cdot 10^{-7}, 1).$$

Taki dobór wag w macierzy $\mathbf{H}$ powoduje możliwie mniejszy ruch w połączeniach ruchowych bliższych podstawie robota. Przedostatni element wag dla połączenia ruchowego wysuwne zostało tak dobrany, aby zrównoważyć ze sobą przemieszczenia liniowe z kątowymi.

6.2. Utrzymanie prędkości ruchu w połączeniach ruchowych w zadanych granicach

Dla każdego z 11 połączeń ruchowych określono, oprócz granic wychyleń, także granice prędkości:

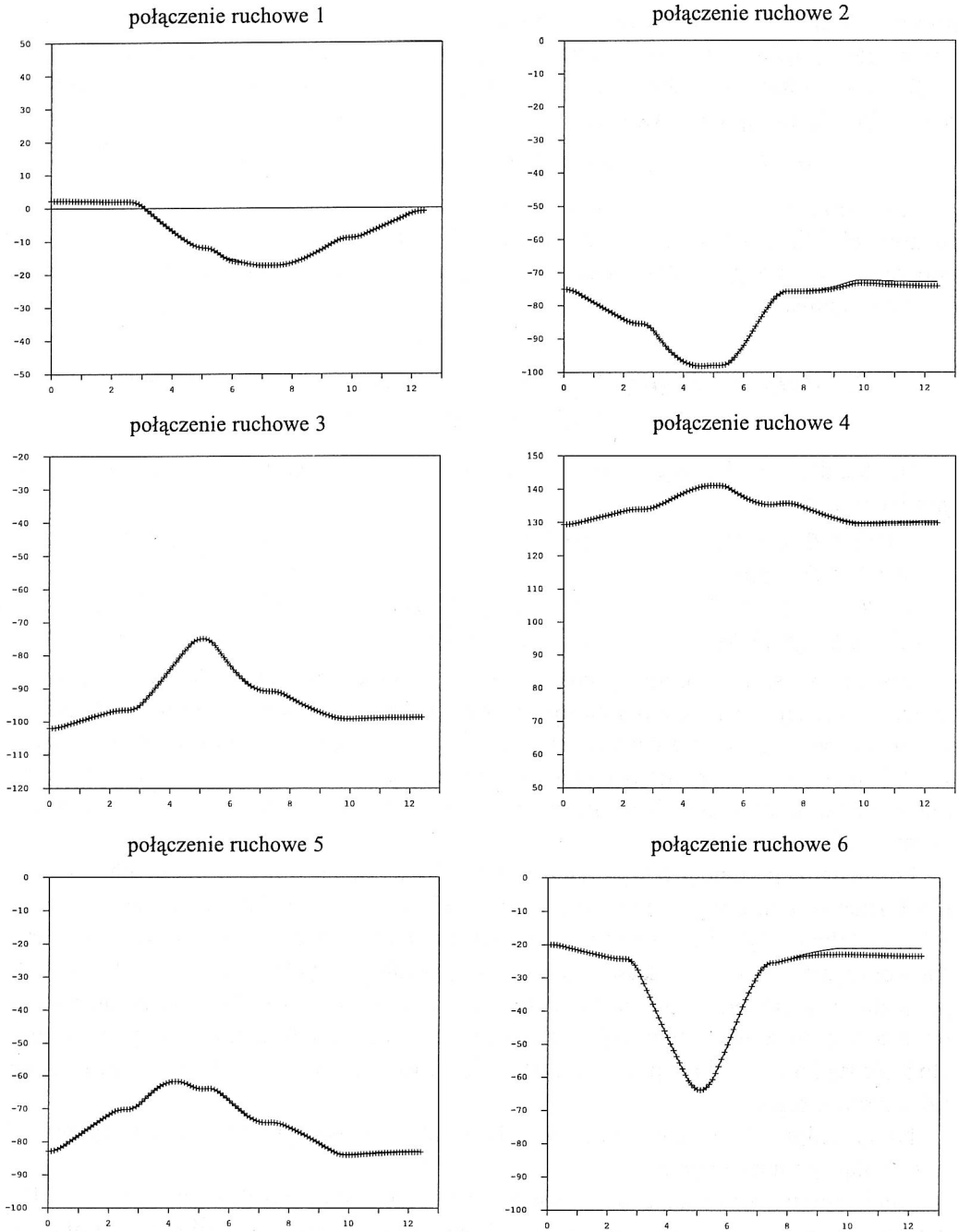
$$\begin{array}{lll} -50^\circ/\text{s} \leq \dot{\theta}_1 \leq 50^\circ/\text{s} & -50^\circ/\text{s} \leq \dot{\theta}_5 \leq 50^\circ/\text{s} & -50^\circ/\text{s} \leq \dot{\theta}_9 \leq 50^\circ/\text{s} \\ -50^\circ/\text{s} \leq \dot{\theta}_2 \leq 50^\circ/\text{s} & -50^\circ/\text{s} \leq \dot{\theta}_6 \leq 50^\circ/\text{s} & -0,5 \text{ m/s} \leq \dot{\theta}_{10} \leq 0,5 \text{ m/s} \\ -50^\circ/\text{s} \leq \dot{\theta}_3 \leq 50^\circ/\text{s} & -50^\circ/\text{s} \leq \dot{\theta}_7 \leq 50^\circ/\text{s} & -25^\circ/\text{s} \leq \dot{\theta}_{11} \leq 25^\circ/\text{s} \\ -50^\circ/\text{s} \leq \dot{\theta}_4 \leq 50^\circ/\text{s} & -50^\circ/\text{s} \leq \dot{\theta}_8 \leq 50^\circ/\text{s} & \end{array}$$

Dla tak określonych granic prędkości ruchu w połączeniach ruchowych przeprowadzono symulację ruchu robota na testowanym torze o kształcie kwadratu. Wyniki tej symulacji porównano z wynikami symulacji, gdy nie obowiązywały granice prędkości. W obu testowanych przypadkach obowiązywały te same ograniczenia na wychYLENIA w poszczególnych połączeniach ruchowych określone podstawowym zestawem granic (zob. rozdz. 6.1).

Porównanie przebiegów wychyleń oraz prędkości ruchu w poszczególnych połączeniach ruchowych, otrzymane z symulacji przedstawiono na kolejnych rysunkach 6.9–6.12. Ustalenie dolnej granicy prędkości ruchu w 11 połączeniu ruchowym na $-25^\circ/\text{s}$ spowodowało zmianę prędkości (rys. 6.12) i przebiegu wychyleń (rys. 6.10) w tym połączeniu ruchowym. Wymuszona zmiana przebiegu ruchu w 11 połączeniu ruchowym wywołała zmiany w przebiegach ruchu pozostałych połączeń ruchowych, tak aby złożenie ruchu wszystkich połączeń zapewniło realizację zaprogramowanego toru ruchu członu roboczego.

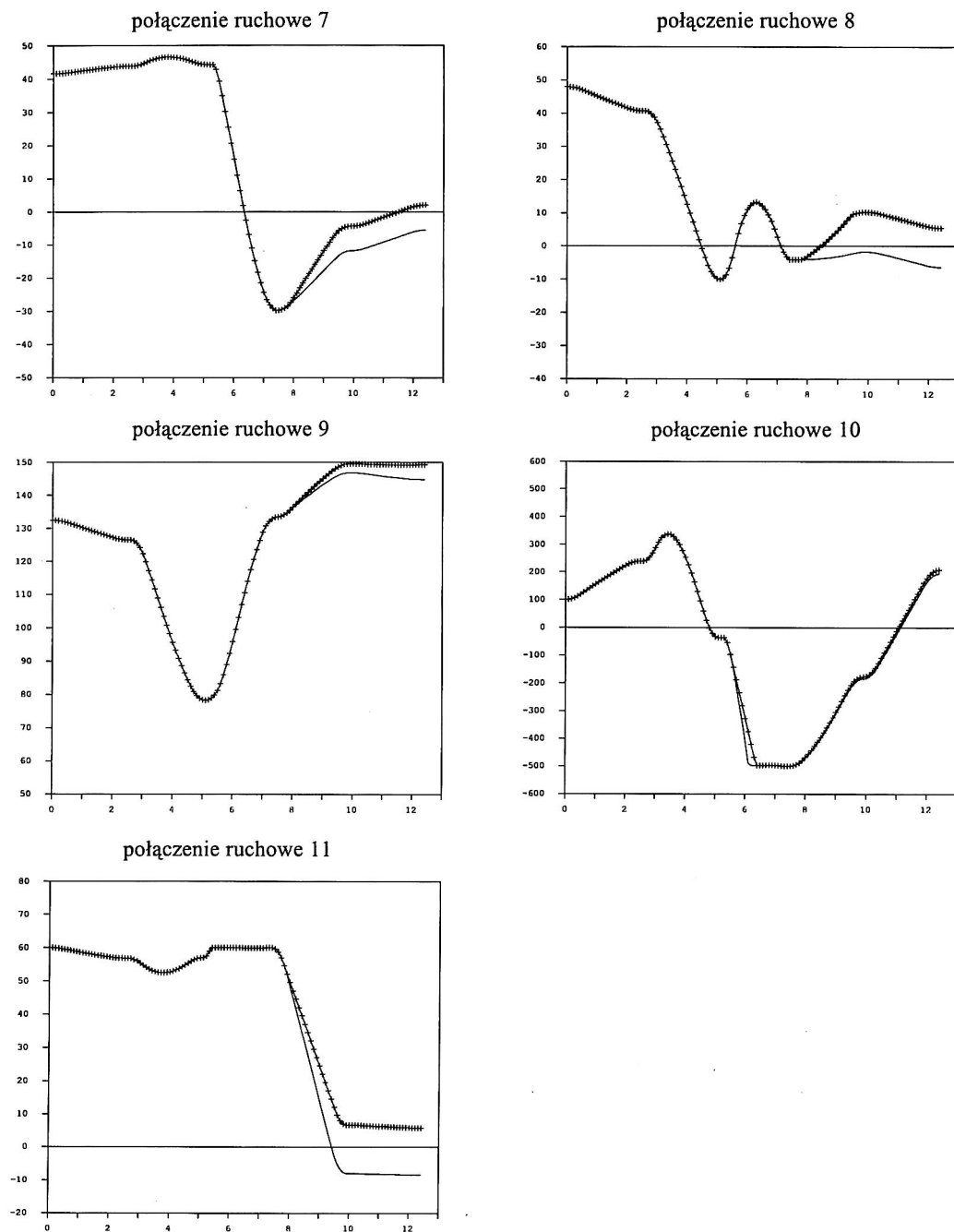
Na rysunku 6.10 widać, że aktywne było także górne ograniczenie wychyleń 60° w 11 połączeniu ruchowym.

Ruch symulowano zadając w algorytmie sterowania, określonym wzorem (40) z rozdziału 4.2, takie same wartości $\mathbf{g}$ i $\mathbf{H}$, jak przy utrzymaniu wychyleń w połączeniach ruchowych w zadanych granicach (patrz rozdz. 6.1)



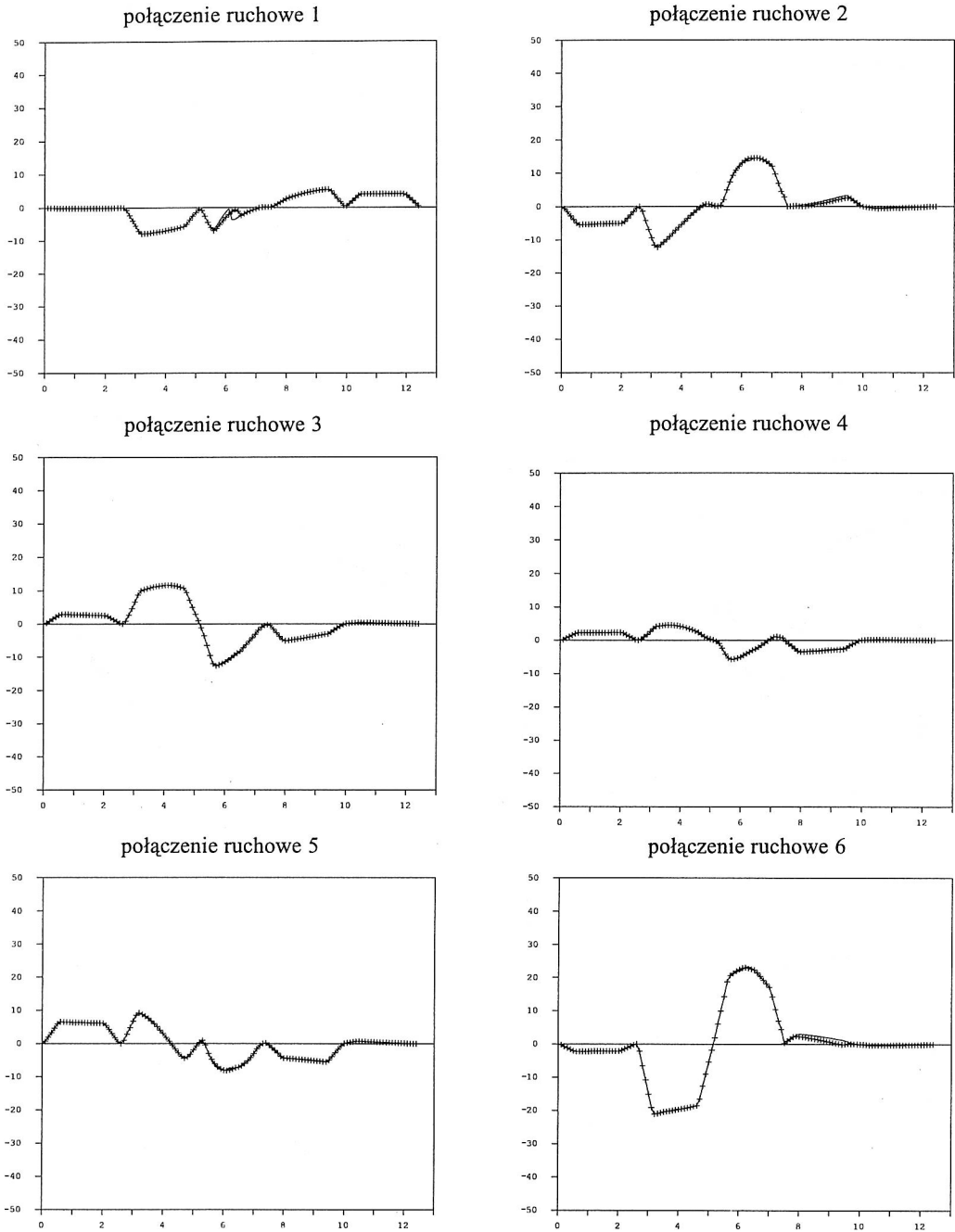
Rys. 6.9. Przebieg wychyleń [w stopniach] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3: — przy braku ograniczenia prędkości w połączeniach ruchowych, +++ przy ustaleniu ograniczeń prędkości (w tym w 11 połączeniu na $[-25,25]^{\circ}/s$)

Fig. 6.9. Motion [in deg] of joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3: — without speed limits for the joints, +++ by setting up speed limits (in joint 11 – $[-25,25]^{\circ}/s$)



Rys. 6.10. Przebieg wychyleń [w stopniach, tylko 10 połączenie ruchowe w mm] w połączeniach ruchowych o numerach 7–11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3: — przy braku ograniczenia prędkości w połączeniach ruchowych, +++ przy ustaleniu ograniczeń prędkości (w tym w połączeniu 11 na $[-25,25]^\circ/\text{s}$)

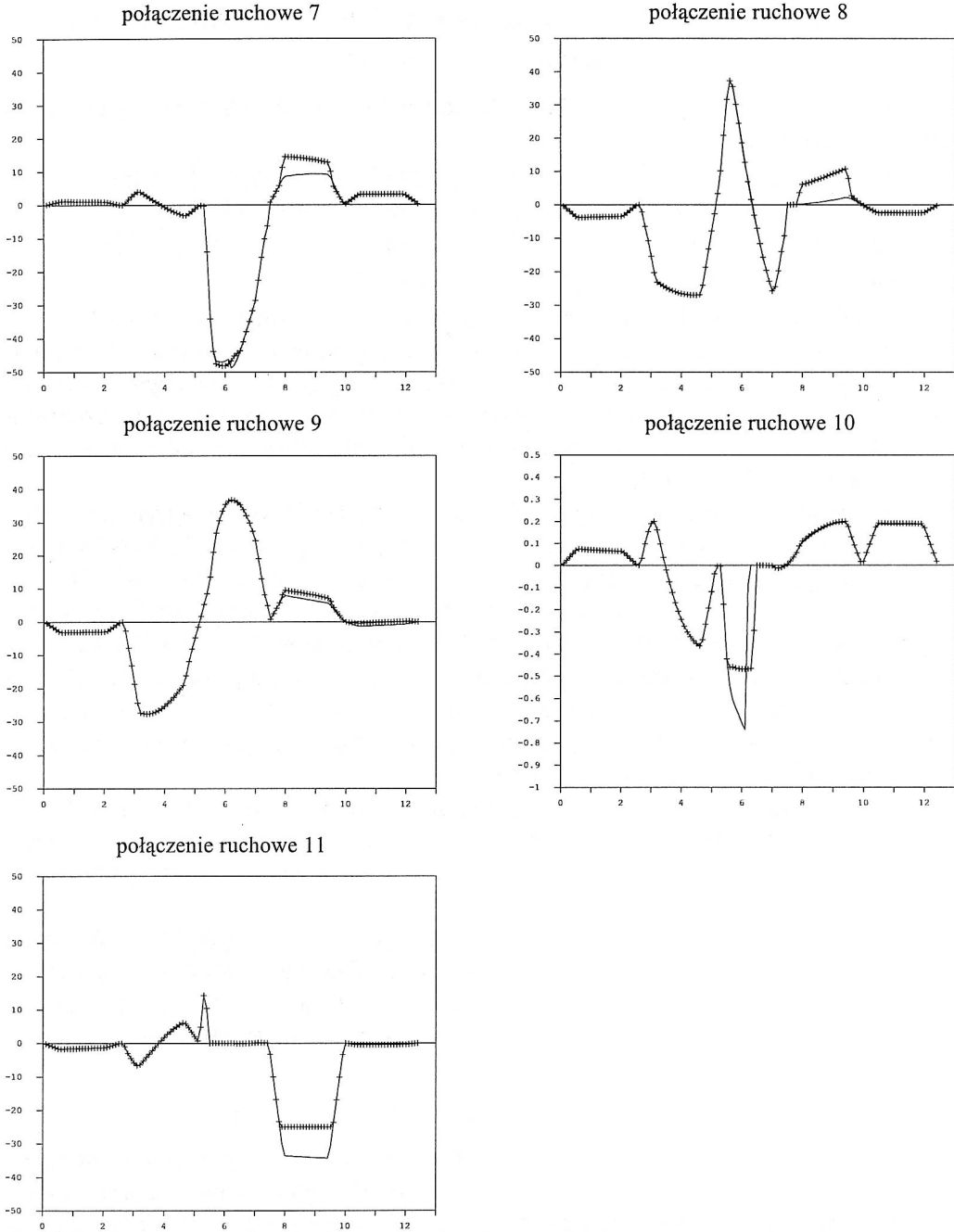
Fig. 6.10. Motion [in deg, only joint 10 in mm] of joints 7–11 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3: — without speed limits for the joints, +++ by setting up speed limits (in joint 11 – $[-25,25]^\circ/\text{s}$)



Rys. 6.11. Przebieg prędkości ruchu [$^{\circ}/s$] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3:

— przy braku ograniczenia prędkości w połączeniach ruchowych, +++ przy ustaleniu ograniczeń prędkości (w tym w połączeniu 11 na $[-25, 25]^{\circ}/s$)

Fig. 6.11. Speed [$^{\circ}/s$] of joints 1–6 during 12.5 sec. while moving the end-effector on the closed square path from fig. 6.3: — without speed limits for the joints, +++ by setting up speed limits (in joint 11 – $[-25, 25]^{\circ}/s$)



Rys. 6.12. Przebieg prędkości ruchu [w %s, tylko 10 połączenie ruchowe w m/s] w połączeniach ruchowych o numerach 7–11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3: — przy braku ograniczenia prędkości w połączeniach ruchowych, +++ przy ustaleniu ograniczeń prędkości (w tym w połączeniu 11 na $[-25,25]^{\circ}/s$)

Fig. 6.12. Speed [in %s, only joint 10 in m/s] of joints 7–11 during 12,5 sec while moving the end-effector on the closed square path from fig. 6.3: — without speed limits for the joints, +++ by setting up speed limits (in joint 11 – $[-25,25]^{\circ}/s$)

6.3. Utrzymanie przyspieszenia w połączeniach ruchowych w zadanych granicach

Dla każdego z 11 połączeń ruchowych określono, oprócz granic wychyleń, także granice przyspieszeń:

$$\begin{array}{lll}
 -350^\circ/\text{s}^2 \leq \ddot{\theta}_1 \leq 350^\circ/\text{s}^2 & -350^\circ/\text{s}^2 \leq \ddot{\theta}_5 \leq 350^\circ/\text{s}^2 & -350^\circ/\text{s}^2 \leq \ddot{\theta}_9 \leq 350^\circ/\text{s}^2 \\
 -350^\circ/\text{s}^2 \leq \ddot{\theta}_2 \leq 350^\circ/\text{s}^2 & -350^\circ/\text{s}^2 \leq \ddot{\theta}_6 \leq 350^\circ/\text{s}^2 & -50 \text{ m/s}^2 \leq \ddot{\theta}_{10} \leq 50 \text{ m/s}^2 \\
 -350^\circ/\text{s}^2 \leq \ddot{\theta}_3 \leq 350^\circ/\text{s}^2 & -350^\circ/\text{s}^2 \leq \ddot{\theta}_7 \leq 350^\circ/\text{s}^2 & -350^\circ/\text{s}^2 \leq \ddot{\theta}_{11} \leq 350^\circ/\text{s}^2 \\
 -350^\circ/\text{s}^2 \leq \ddot{\theta}_4 \leq 350^\circ/\text{s}^2 & -350^\circ/\text{s}^2 \leq \ddot{\theta}_8 \leq 350^\circ/\text{s}^2 &
 \end{array}$$

Dla tak określonych granic przyspieszeń ruchu w połączeniach ruchowych przeprowadzono symulację ruchu robota na testowanym torze o kształcie kwadratu. Wyniki tej symulacji porównano z wynikami symulacji, gdy nie obowiązywały granice przyspieszeń. W obu testowanych przypadkach obowiązywały te same ograniczenia na wychylenia w poszczególnych połączeniach ruchowych, określone podstawowym zestawem granic (zob. rozdz. 6.1).

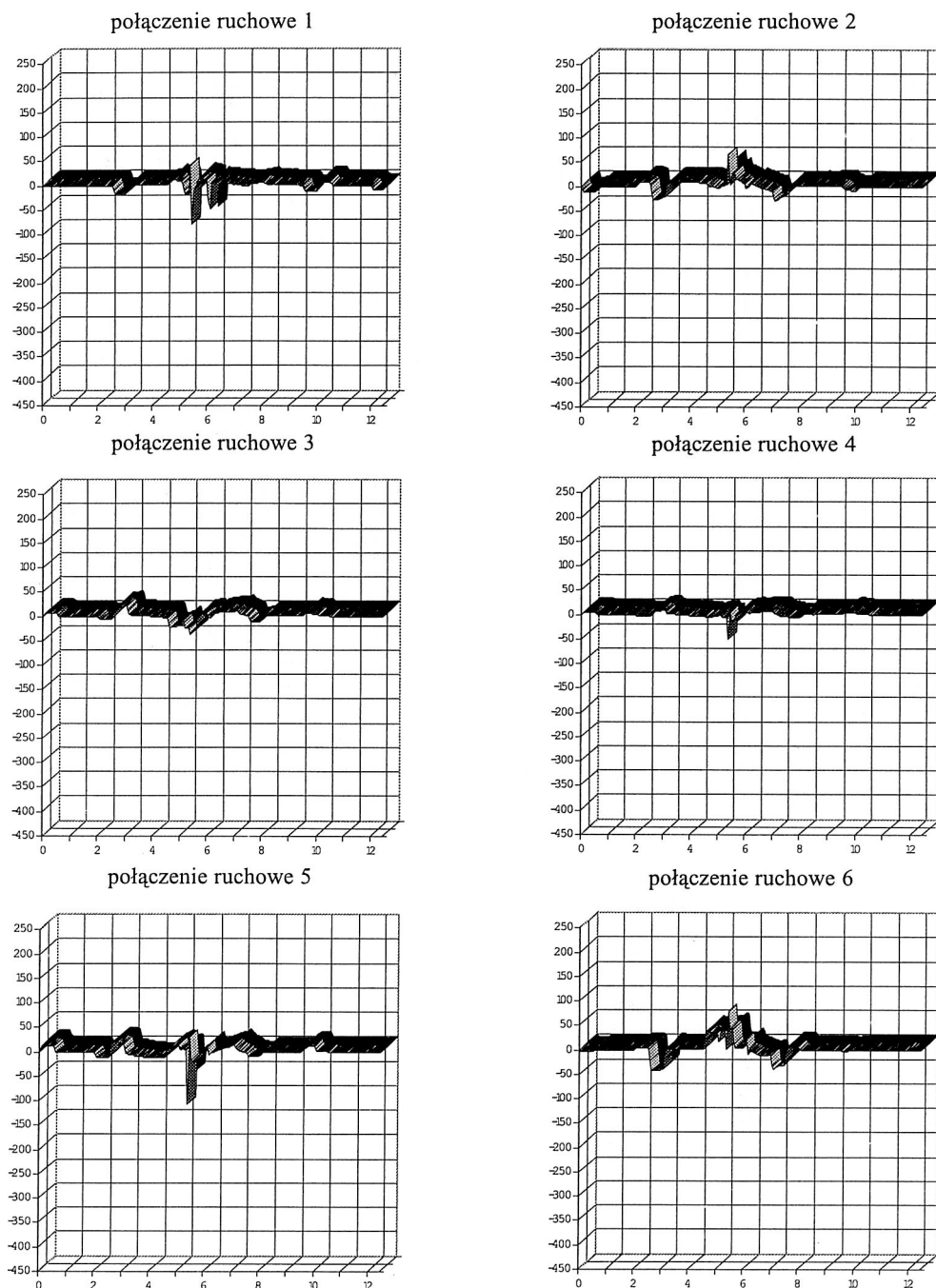
Porównanie przebiegów przyspieszeń ruchu w poszczególnych połączeniach ruchowych otrzymane z symulacji przedstawiono na kolejnych rysunkach 6.13 i 6.14. Ustalenie dolnej granicy przyspieszenia ruchu w 7 połączeniu ruchowym na $-350^\circ/\text{s}^2$ spowodowało zmniejszenie maksymalnego przyspieszenia z $-425^\circ/\text{s}^2$ na $-350^\circ/\text{s}^2$ (rys. 6.14). Wymuszona tym ograniczeniem zmiana przebiegu przyspieszenia ruchu w 7 połączeniu ruchowym wywołała zmiany w przebiegach przyspieszeń ruchu pozostałych połączeń ruchowych, tak aby złożenie ruchu wszystkich połączeń zapewniło realizację zaprogramowanego toru ruchu członu roboczego zadaną dynamiką. Największy wzrost przyspieszenia nastąpił w 10 połączeniu ruchowym.

Ruch symulowano zadając w algorytmie sterowania, określonym wzorem (40) z rozdziału 4.2, takie same wartości współczynników wektora $\mathbf{g}$ i macierzy $\mathbf{H}$, jak przy utrzymaniu wychyleń w połączeniach ruchowych w zadanych granicach (patrz rozdz. 6.1).

Zmiany przebiegu ruchu w połączeniach ruchowych, w tym ich przyspieszeń, można także dokonać zmieniając współczynniki macierzy $\mathbf{H}$. Dla zilustrowania tej możliwości symulowano ruch nie tylko z nałożonymi ograniczeniami na przyspieszenia (jak powyżej), ale także zwiększono czterokrotnie wagę przypadającą na 10 połączenie ruchowe, tak że macierz $\mathbf{H}$ wyrażała się następująco:

$$\mathbf{H} = \text{diag}(11, 10, 9, 8, 7, 6, 5, 4, 3, 2 \cdot 10^{-6}, 1).$$

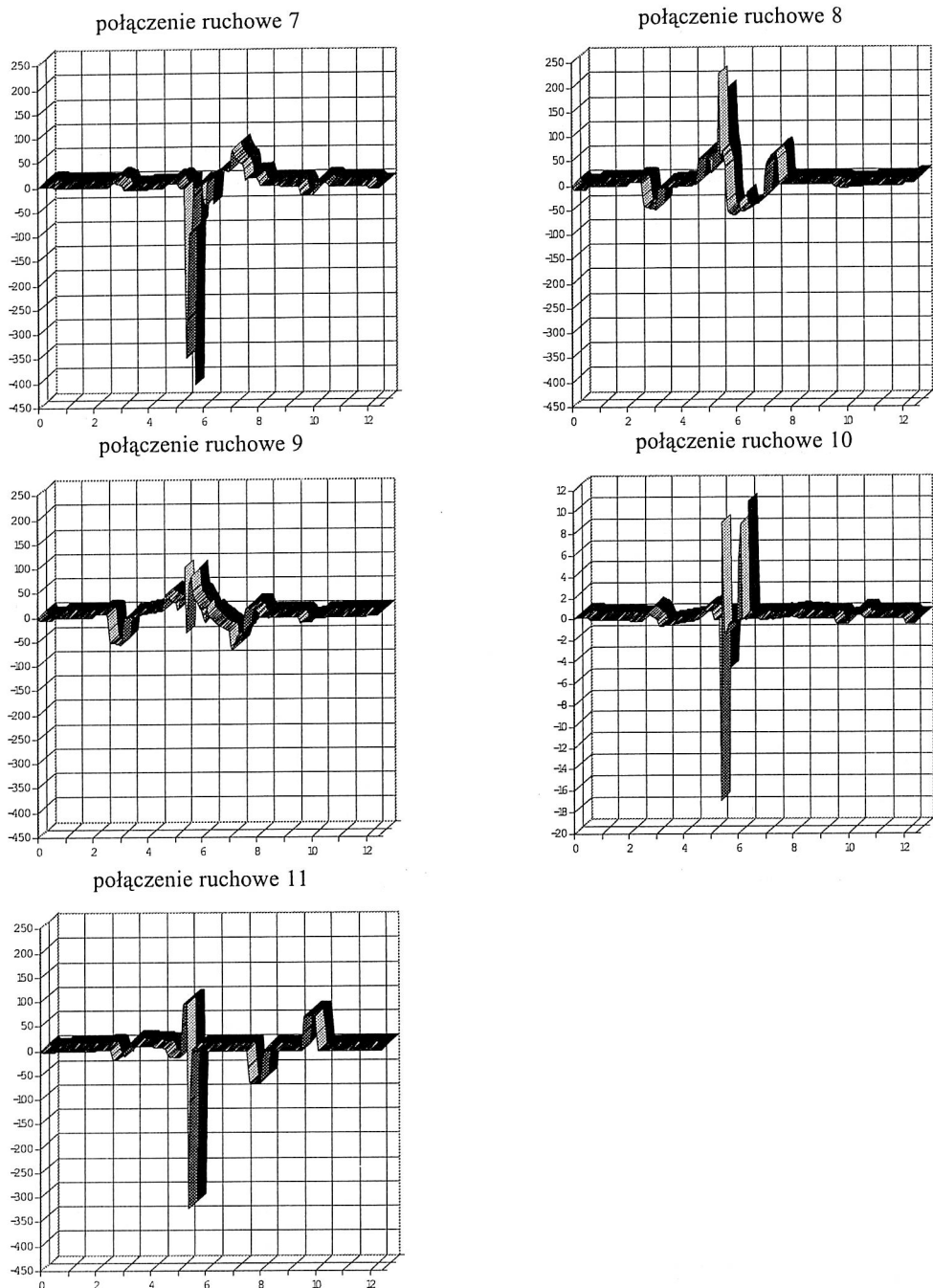
Takie zwiększenie wagi w macierzy $\mathbf{H}$, przypadające na 10 połączenie ruchowe, spowodowało ponad dwukrotne zmniejszenie maksymalnego przyspieszenia w tym połączeniu (rys. 6.16). Zwiększyły się przy tym przyspieszenia ruchu kilku innych połączeń ruchowych, tak aby złożenie ruchu wszystkich połączeń zapewniło realizację zaprogramowanego toru ruchu członu roboczego zadaną dynamiką.



Rys. 6.13. Przebieg przyspieszeń ruchu [w m/s^2] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3:

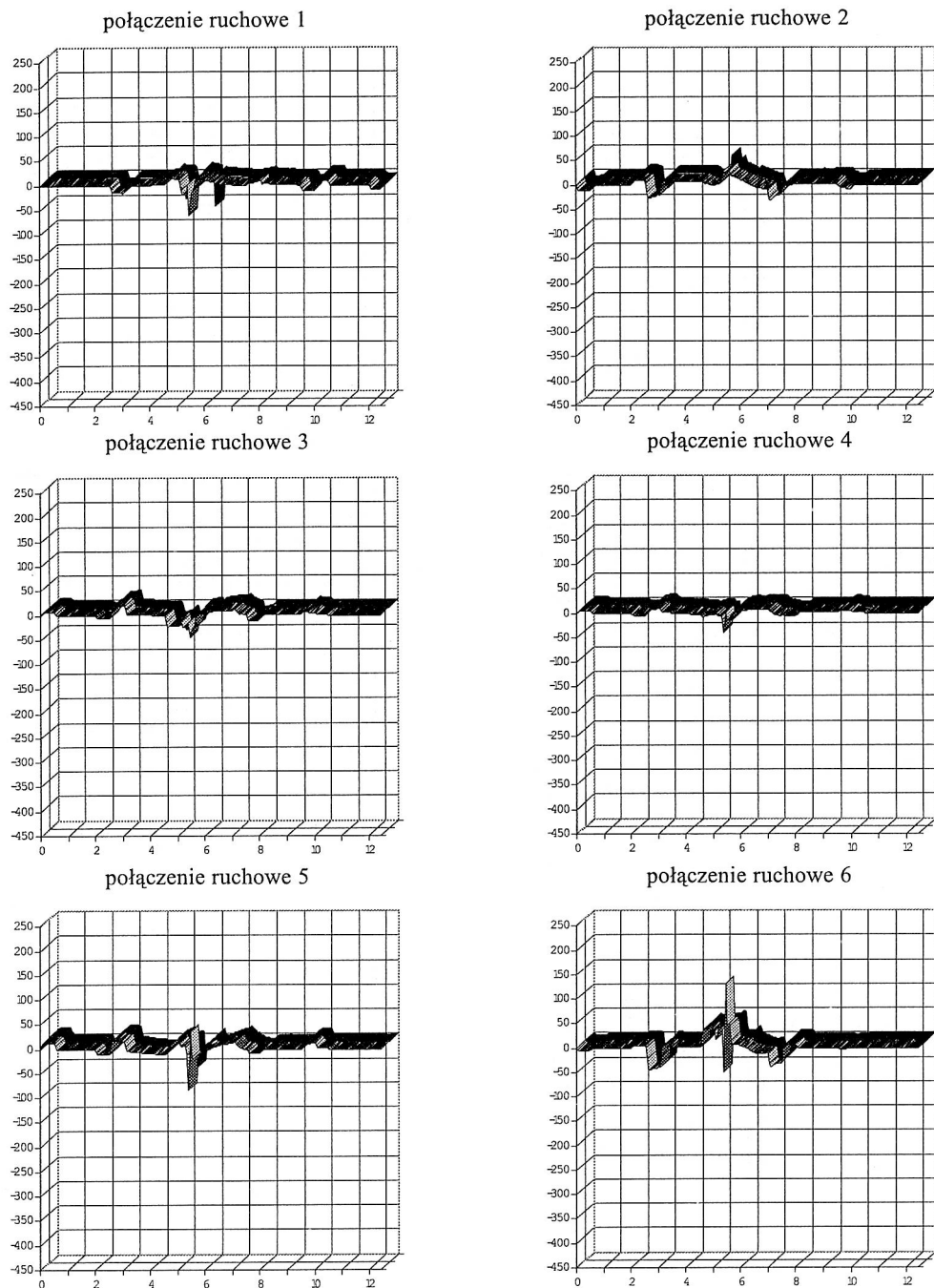
■ przy braku ograniczenia przyspieszeń w połączeniach ruchowych, ▨ przy ustaleniu ograniczeń przyspieszeń (w tym w 7 połączeniu na $[-350,350]\text{m/s}^2$)

Fig. 6.13. Acceleration [in m/s^2] of joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3: ■ without acceleration limits for the joints, ▨ by setting up acceleration limits (in joint 7 – $[-350,350]\text{m/s}^2$)



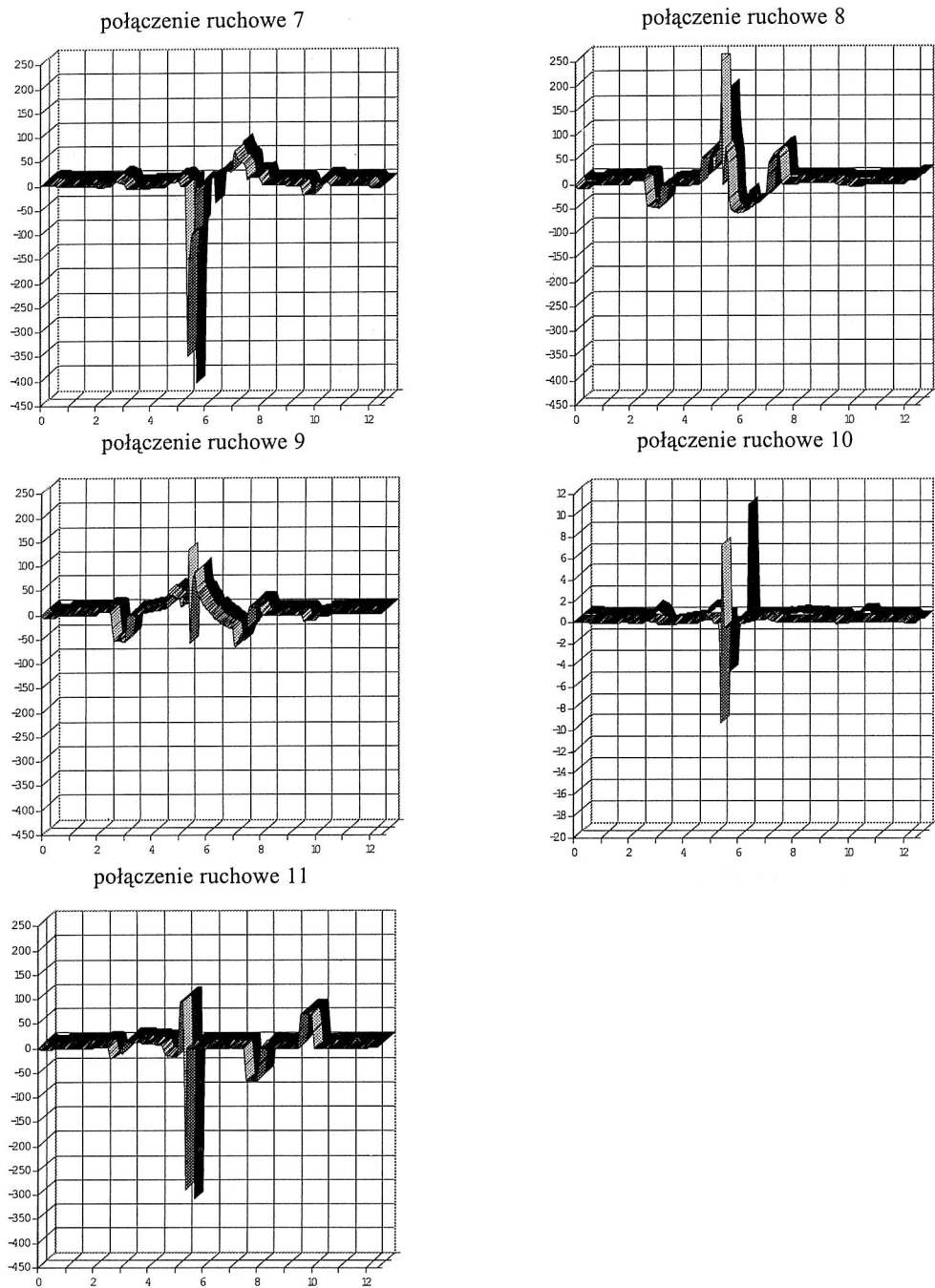
Rys. 6.14. Przebieg przyspieszeń ruchu [w m/s^2 , tylko 10 połączenie ruchowe w m/s^2] w połączeniach ruchowych o numerach 7–11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3: ■ przy braku ograniczenia przyspieszeń w połączeniach ruchowych, ▨ przy ustaleniu ograniczeń przyspieszeń (w tym w 7 połączeniu na $[-350,350]\text{m/s}^2$)

Fig. 6.14. Acceleration [in m/s^2 , only joint 10 in m/s^2] of joints 7–11 during 12,5 sec while moving the end-effector on the closed square path from fig. 6.3: ■ without acceleration limits for the joints, ▨ by setting up acceleration limits (in joint 7 – $[-350,350]\text{m/s}^2$)



Rys. 6.15. Przebieg przyspieszeń ruchu $[\text{m/s}^2]$ w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3: ■ przy braku ograniczenia przyspieszeń w połączeniach ruchowych, ▨ przy ustaleniu ograniczeń przyspieszeń (w tym w połączeniu 7 na $[-350,350]\text{m/s}^2$) i przy 4-krotnym zwiększeniu wagi w macierzy H dla 11 połączenia ruchowego

Fig. 6.15. Acceleration $[\text{m/s}^2]$ of joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3: ■ without acceleration limits for the joints, ▨ by setting up acceleration limits (in joint 7 – $[-350,350]\text{m/s}^2$) and by increasing the weighting coefficient in the matrix H for joint 11 by 4 times



Rys. 6.16. Przebieg przyspieszeń ruchu [$\text{w } ^\circ/\text{s}^2$, tylko 10 połączenie ruchowe w m/s^2] w połączeniach ruchowych o numerach 7–11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3: ■ przy braku ograniczenia przyspieszeń w połączeniach ruchowych, ■ przy ustaleniu ograniczeń przyspieszeń (w tym w połączeniu 7 na $[-350,350]^\circ/\text{s}^2$) i przy 4-krotnym zwiększeniu wagi w macierzy H dla 11 połączenia ruchowego

Fig. 6.16. Acceleration [$\text{in } ^\circ/\text{s}^2$, only joint 10 in m/s^2] of joints 7–11 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3: ■ without acceleration limits for the joints, ■ by setting up acceleration limits (in joint 7 – $[-350,350]^\circ/\text{s}^2$) and by increasing the weighting coefficient in the matrix H for joint 11 by 4 times

6.4. Zapewnienie powtarzalności ruchu

Jak wynika z rysunku 6.3, algorytm nie gwarantuje powtarzalności ruchu po zamkniętym torze, konfiguracja bowiem ramion robota na końcu ruchu różni się od tej na starcie, choć człon roboczy w obu przypadkach przyjmuje tę samą orientację i pozycję. Widoczne jest to też na wykresach przebiegu wychyleń poszczególnych połączeń ruchowych (rys. 6.5, 6.6 oraz 6.9, 6.10). Wykresy te nie kończą się po 12,5 sekundach z tymi wartościami wychyleń, z którymi się rozpoczęły. Po włączeniu opcji cykliczności ruchu w algorytmie, uzyskano wymaganą powtarzalność ruchu, co widać na rys. 6.17, gdzie konfiguracje ramion – początkowa i końcowa – są identyczne. Zarejestrowane przebiegi wychyleń poszczególnych połączeń ruchowych (rys. 6.18 i 6.19) przedstawiają porównanie obu wariantów ruchu: bez i przy zapewnieniu powtarzalności ruchu. W obu testowanych przypadkach obowiązywały także ograniczenia na wychylenia w poszczególnych połączeniach ruchowych, określone podstawowym zestawem granic (zob. rozdz. 6.1). Jak widać, w przypadku 10 i 11 połączenia ruchowego granice te są aktywne (rys. 6.19).

Ruch symulowano zadając wagowe kryterium do minimalizacji, określone wyrażeniem (46) z rozdziału 4.2.2, z macierzą wag $\mathbf{W}$ określoną jak w poprzednio opisanych opcjach ruchu

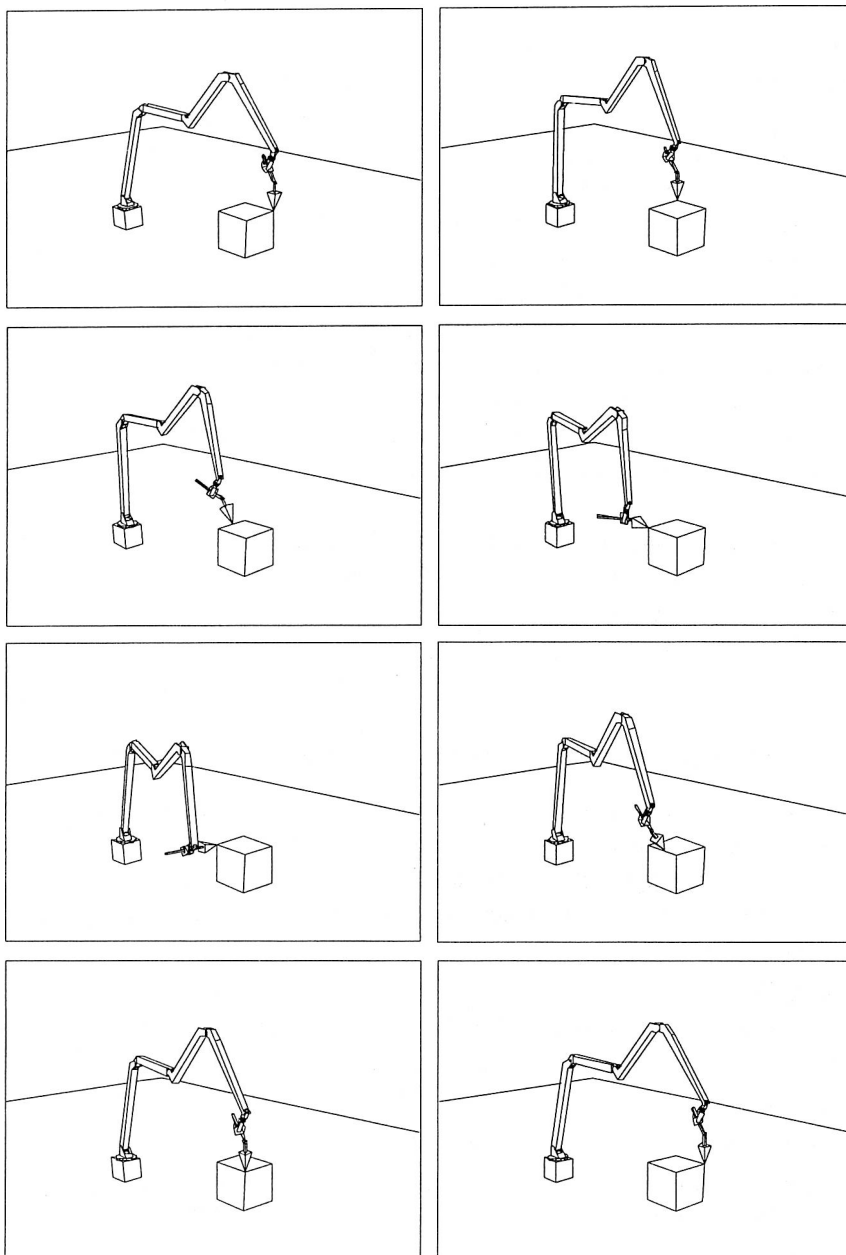
$$\mathbf{W} = \text{diag}(11, 10, 9, 8, 7, 6, 5, 4, 3, 5 \cdot 10^{-7}, 1).$$

6.5. Zamrożenie ruchu w wybranych połączeniach ruchowych

Zadanie testowane w rozdziale 6.2 utrzymania wychyleń i prędkości w połączeniach ruchowych w zadanych granicach zostało również przetestowane przy zmniejszonej redundancji robota, zamrażając ruch w jego drugim i trzecim połączeniu ruchowym. Symulację przeprowadzono przy następujących ograniczeniach na wychylenie i prędkości w połączeniach ruchowych:

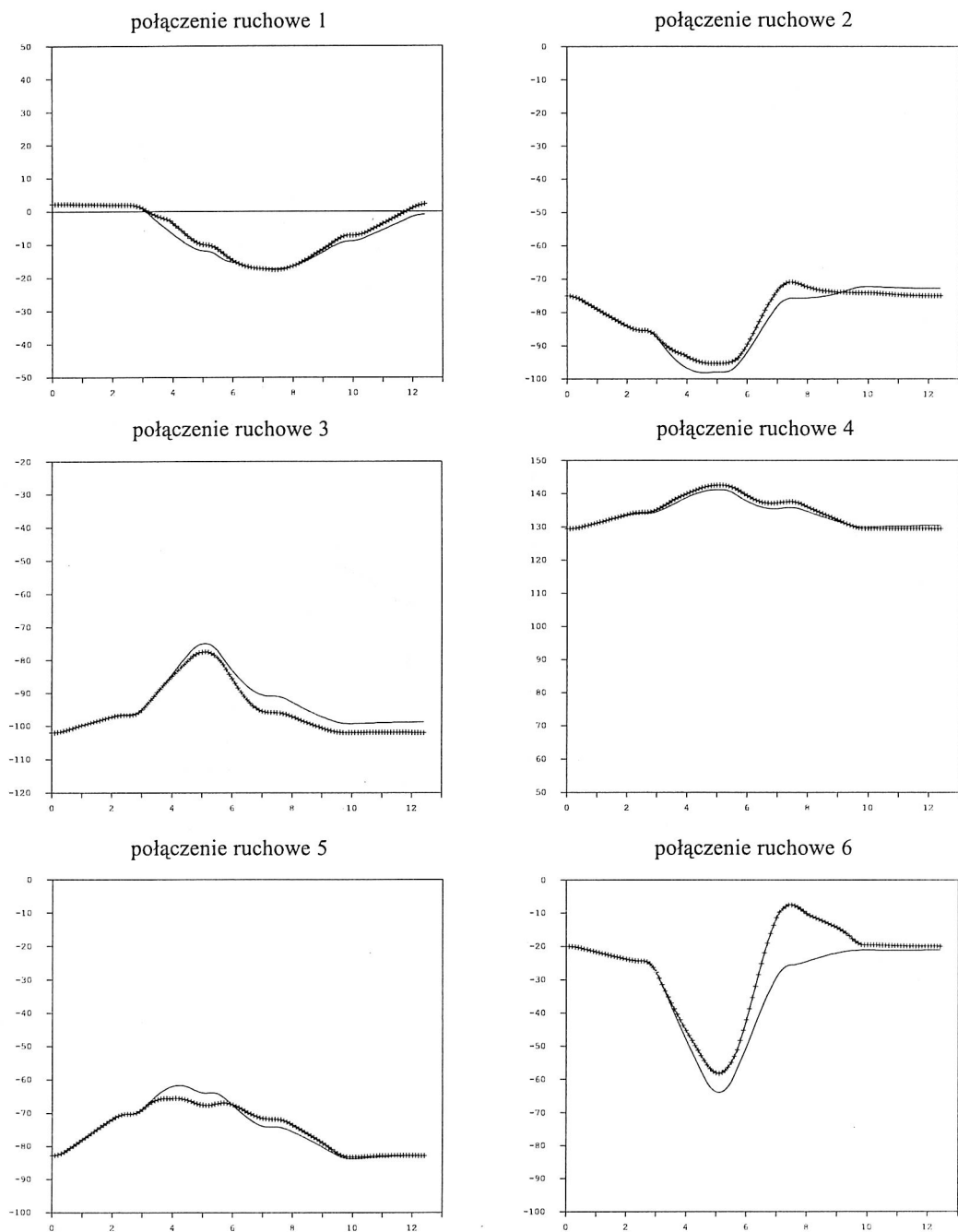
$-180^\circ \leq \theta_1 \leq 180^\circ$	$-180^\circ \leq \theta_5 \leq 0^\circ$	$-180^\circ \leq \theta_9 \leq 180^\circ$
$\theta_2 = -75^\circ \text{ (const)}$	$-180^\circ \leq \theta_6 \leq 0^\circ$	$-500 \text{ mm} \leq \theta_{10} \leq 500 \text{ mm}$
$\theta_3 = -102^\circ \text{ (const)}$	$-180^\circ \leq \theta_7 \leq 180^\circ$	$0^\circ \leq \theta_{11} \leq 60^\circ$
$0^\circ \leq \theta_4 \leq 180^\circ$	$-60^\circ \leq \theta_8 \leq 60^\circ$	
$-50^\circ/\text{s} \leq \dot{\theta}_1 \leq 50^\circ/\text{s}$	$-50^\circ/\text{s} \leq \dot{\theta}_5 \leq 50^\circ/\text{s}$	$-50^\circ/\text{s} \leq \dot{\theta}_9 \leq 50^\circ/\text{s}$
$\dot{\theta}_2 = 0^\circ/\text{s} \text{ (const)}$	$-50^\circ/\text{s} \leq \dot{\theta}_6 \leq 50^\circ/\text{s}$	$-500 \text{ mm/s} \leq \dot{\theta}_{10} \leq 500 \text{ mm/s}$
$\dot{\theta}_3 = 0^\circ/\text{s} \text{ (const)}$	$-50^\circ/\text{s} \leq \dot{\theta}_7 \leq 50^\circ/\text{s}$	$-25^\circ/\text{s} \leq \dot{\theta}_{11} \leq 25^\circ/\text{s}$
$-50^\circ/\text{s} \leq \dot{\theta}_4 \leq 50^\circ/\text{s}$	$-50^\circ/\text{s} \leq \dot{\theta}_8 \leq 50^\circ/\text{s}$	

Na rysunkach 6.20–6.23 przedstawiono przebiegi wychyleń i prędkości ruchu w połączeniach ruchowych otrzymane z symulacji. Na rys. 6.21 widać, że aktywne były obie granice na wychylenie w połączeniu ruchowym o numerze 11 i dolna granica na



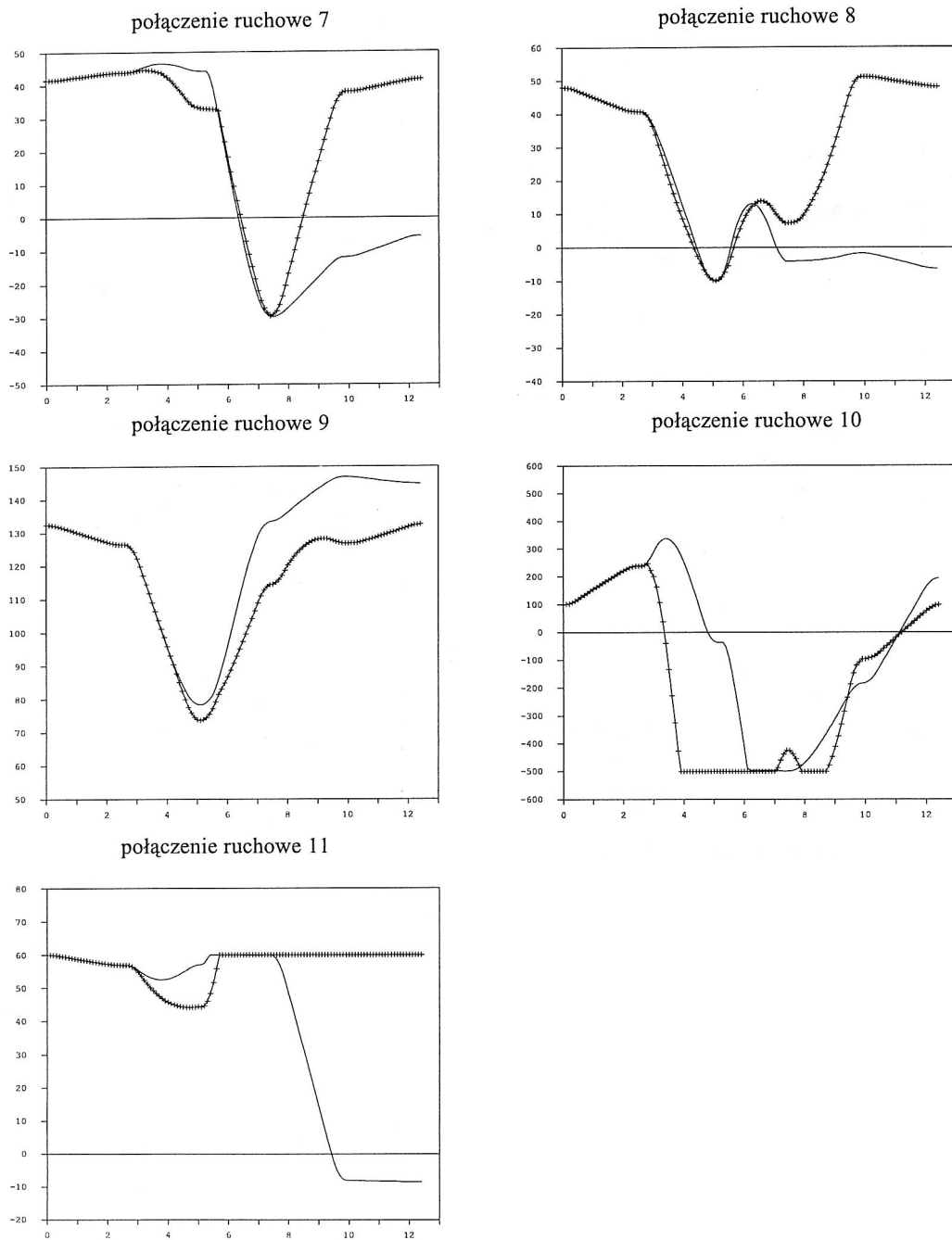
Rys. 6.17. Symulacja ruchu 11-przegubowego robota na trajektorii o kształcie kwadratu przy włączonej opcji powtarzalności ruchu po zamkniętej trajektorii

Fig. 6.17. Simulation of motion of 11-joint robot on a square path with the option of assurance of motion repeatability on a closed path



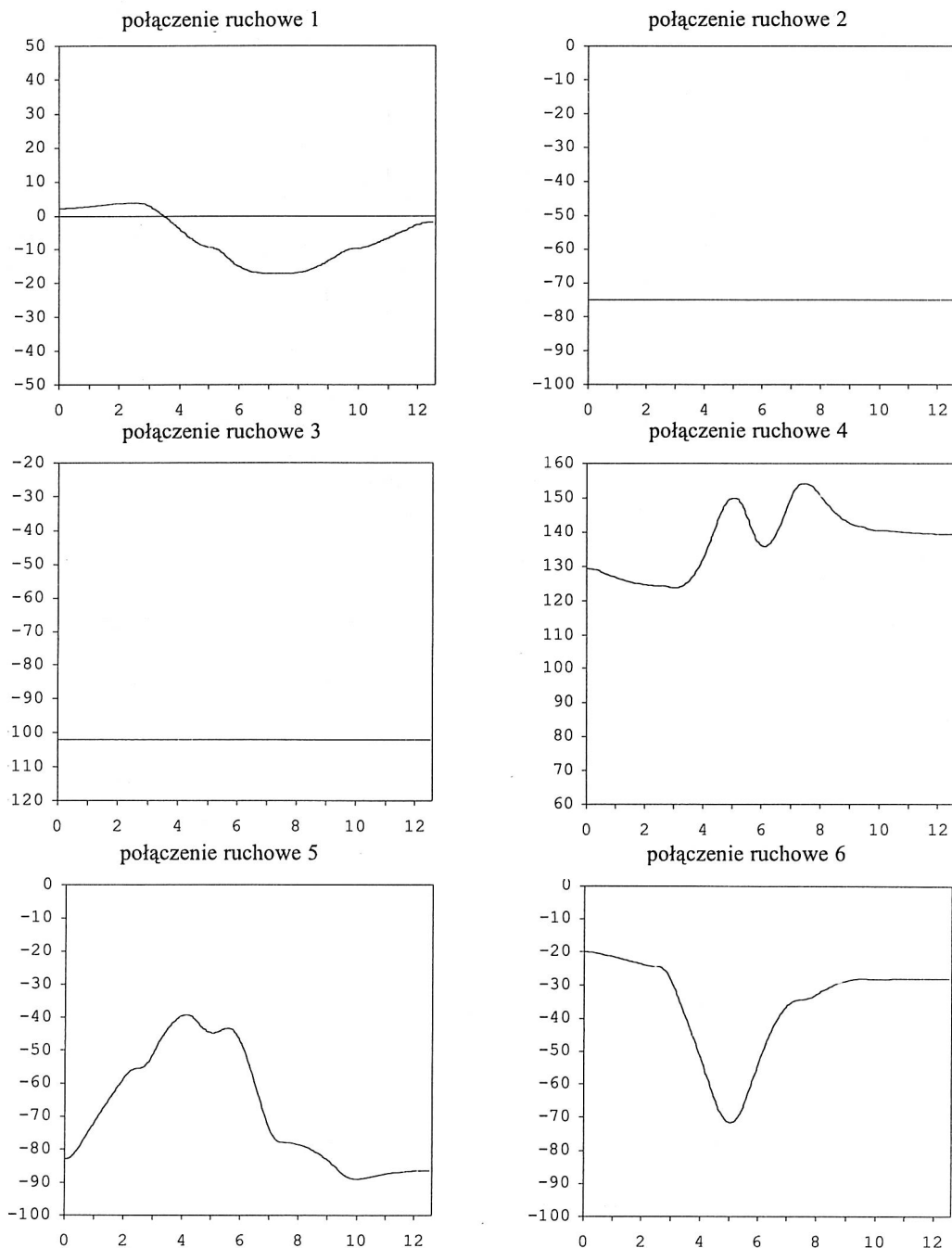
Rys. 6.18. Przebieg wychyleń [w stopniach] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy dwóch różnych opcjach ruchu: — bez powtarzalności ruchu, +++ przy włączonej opcji powtarzalności ruchu

Fig. 6.18. Motion [in deg] of joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3, for two different motion options: — without and +++ with the assurance of motion repeatability on a closed path



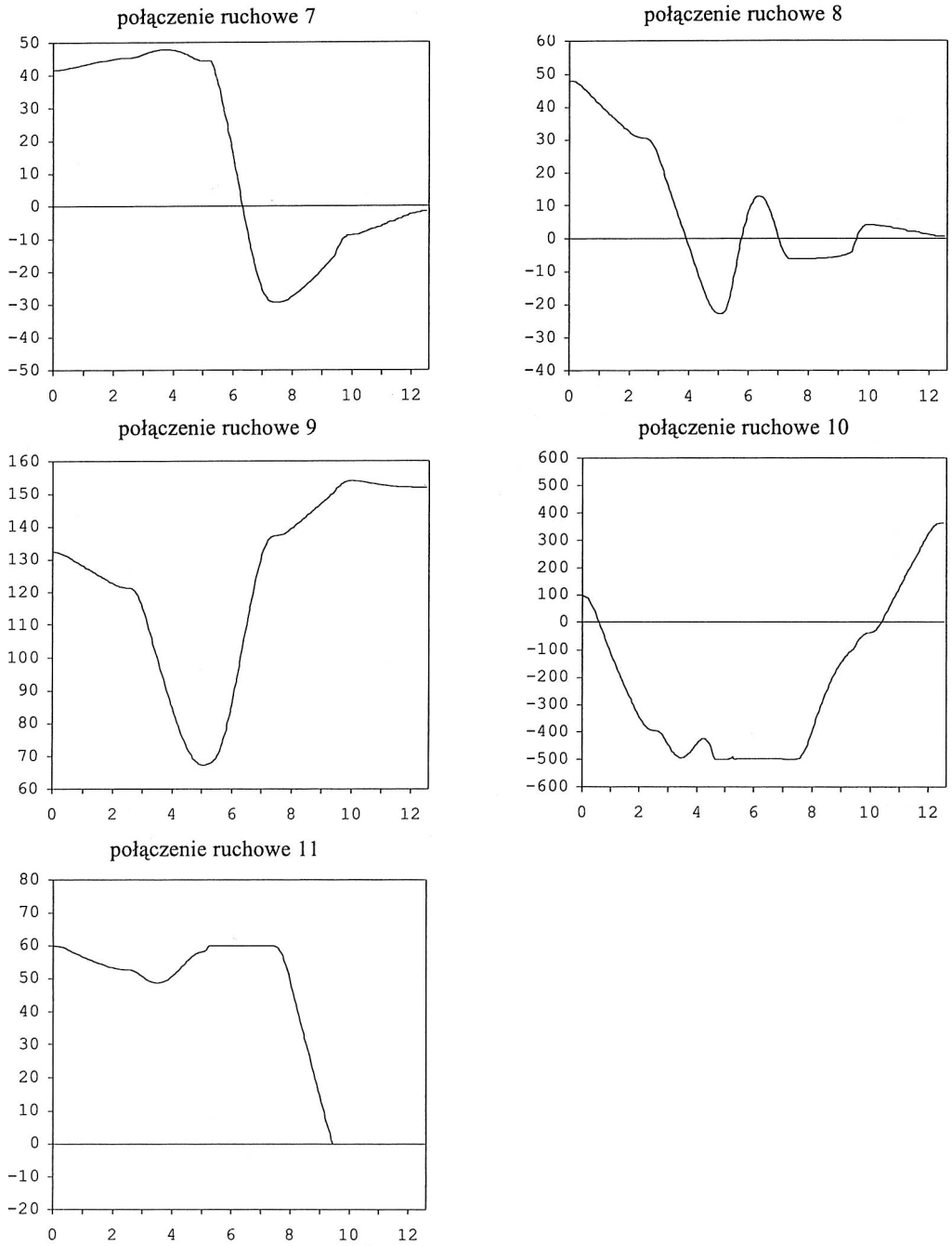
Rys. 6.19. Przebieg wychyleń [w stopniach, tylko 10 połączenie ruchowe w mm] w połączeniach ruchowych o numerach 7 – 11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy dwóch różnych opcjach ruchu: — bez powtarzalności ruchu, +++ przy włączonej opcji powtarzalności ruchu

Fig. 6.19. Motion [in deg, only joint 10 in mm] of joints 7–11 during 12,5 sec. while moving the endeffector on the closed square path from fig. 6.3, for two different motion options: – without and +++ with the assurance of motion repeatability on a closed path



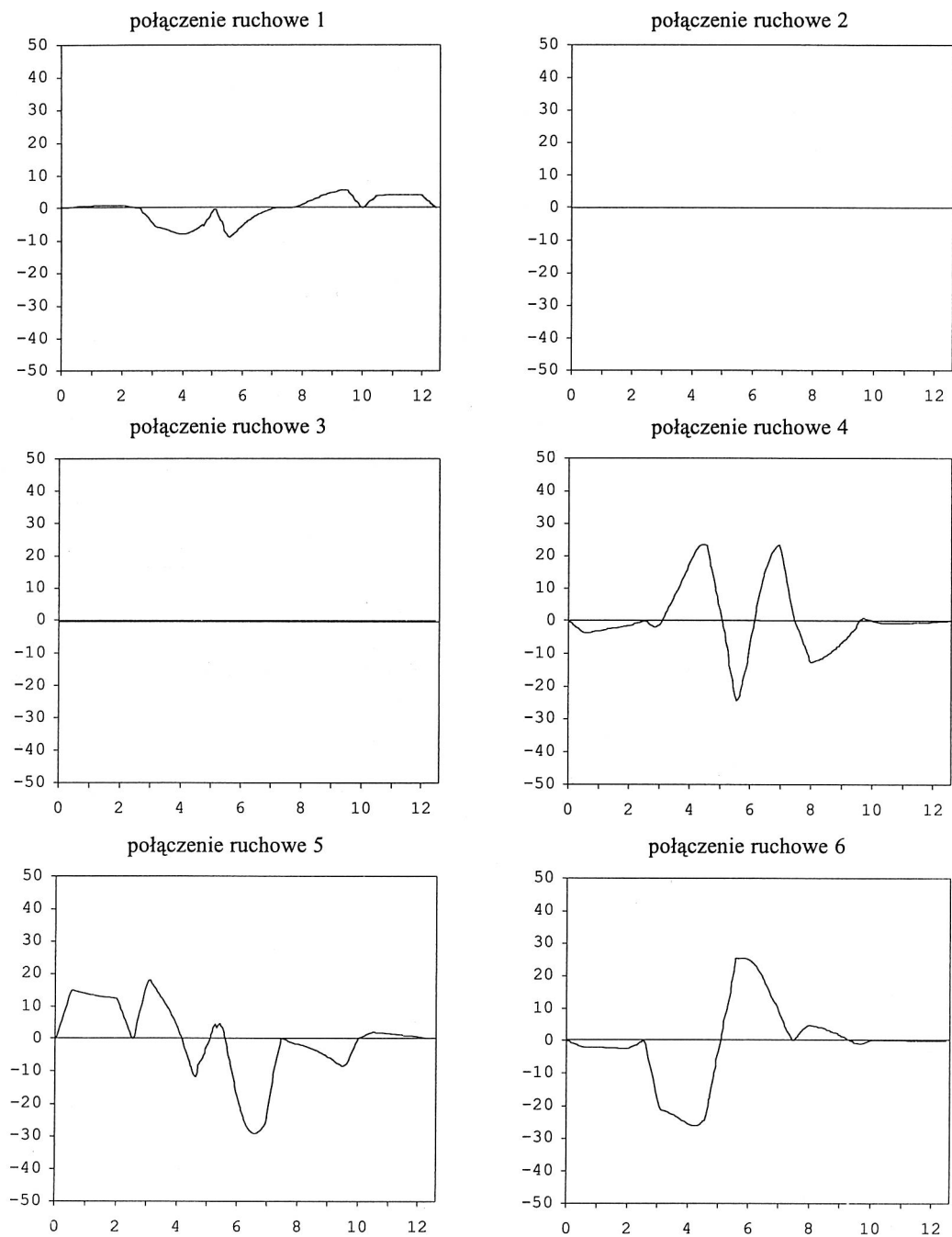
Rys. 6.20. Przebieg wychyleń [w stopniach] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy zamrożeniu ruchu w połączeniach ruchowych o numerach 2 i 3

Fig. 6.20. Motion [in deg] of joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3, by freezing the motion in 2nd and 3rd joints



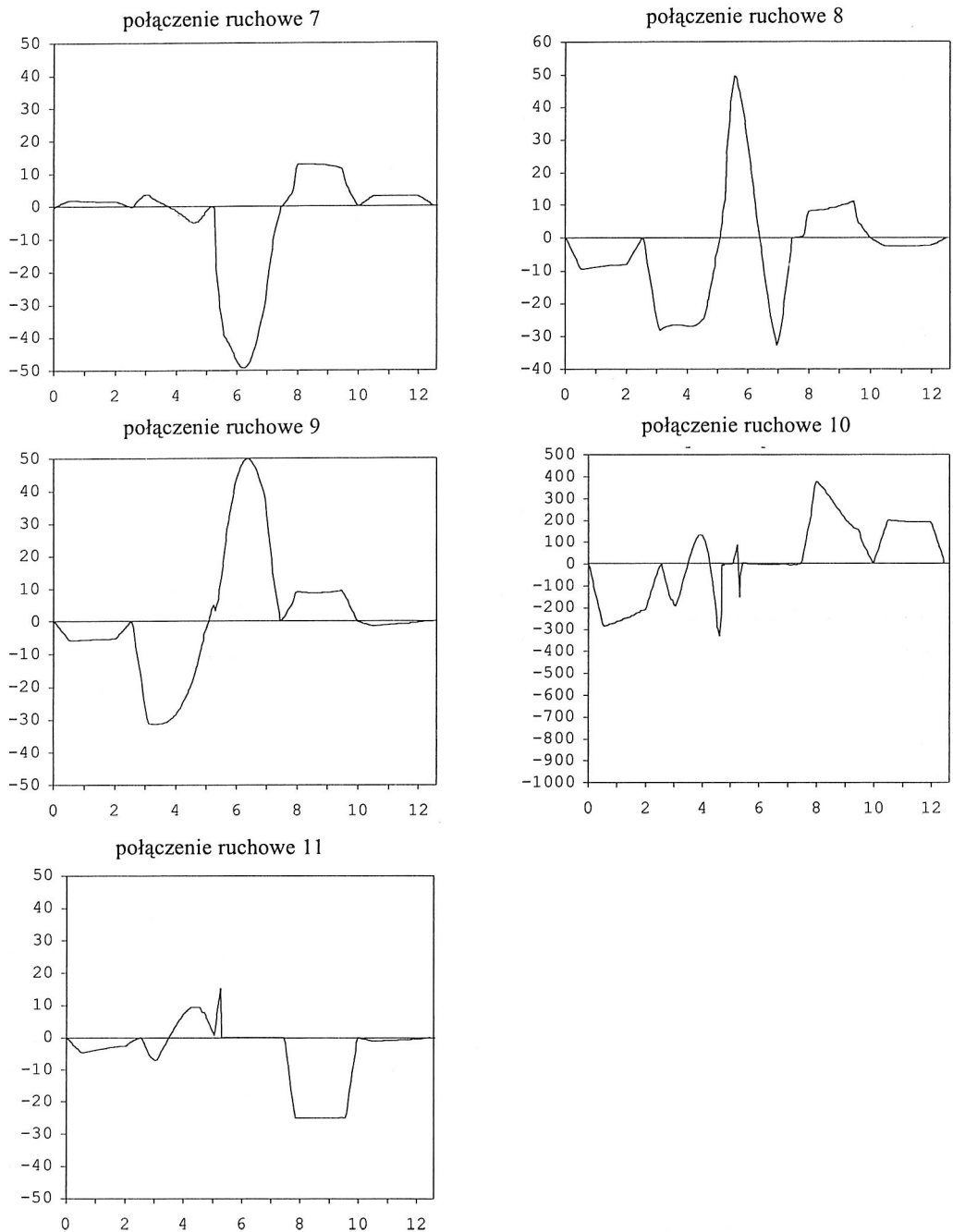
Rys. 6.21. Przebieg wychyleń [w stopniach, tylko 10 połączenie ruchowe w mm] w połączeniach ruchowych o numerach 7–11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy zamrożeniu ruchu w połączeniach ruchowych o numerach 2 i 3

Fig. 6.21. Motion [in deg, only joint 10 in mm] of joints 7–11 during 12,5 sec while moving the end-effector on the closed square path from fig. 6.3, by freezing the motion in 2nd and 3rd joints



Rys. 6.22. Przebieg prędkości ruchu [w %/s] w połączeniach ruchowych o numerach 1–6 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy zamrożeniu ruchu w połączeniach ruchowych o numerach 2 i 3

Fig. 6.22. Speed [in %/s] of the joints 1–6 during 12,5 sec. while moving the end-effector on the closed square path from fig. 6.3, by freezing the motion in 2nd and 3rd joints



Rys. 6.23. Przebieg prędkości ruchu [w %/s, tylko 10 połączenie ruchowe w mm/s] w połączeniach ruchowych o numerach 7–11 w czasie 12,5 sekund, podczas ruchu członu roboczego po zamkniętym torze o kształcie kwadratu z rys. 6.3, przy zamrożeniu ruchu w połączeniach ruchowych o numerach 2 i 3

Fig. 6.23. Speed [in %/s, only joint 10 in m/s] of joints 7–11 during 12,5 sec while moving the end-effector on the closed square path from fig. 6.3, by freezing the motion in 2nd and 3rd joints

wychylenie w połączeniu ruchowym o numerze 10, a także dolna granica na prędkość ruchu w połączeniu ruchowym o numerze 11.

Z porównania tych przebiegów z wykresami na rys. 6.9–6.12 wynika, że zamrożenie ruchu w dwóch przegubach spowodowało zwiększoną zmienność w pozostałych.

6.6. Omijanie przeszkód

W celu przetestowania sterowania ruchem robota przy opcji omijania przeszkód umieszczono w przestrzeni roboczej robota przeszkodę w postaci punktu oraz zdefiniowano tor ruchu będący odcinkiem o długości 5 m skierowanym pionowo w dół, równoległe do osi z . Przeszkoda została tak umieszczona w przestrzeni, aby podczas ruchu po zdefiniowanym torze nastąpiła kolizja z jednym z ramion. Tor ruchu zaprogramowano w interpolacji liniowej w języku BAPS następująco:

PROGRAMM OBST1

FAHRE LINEAR NACH (11500,1000,3500,0,0,0) ;dojazd do początku trajektorii

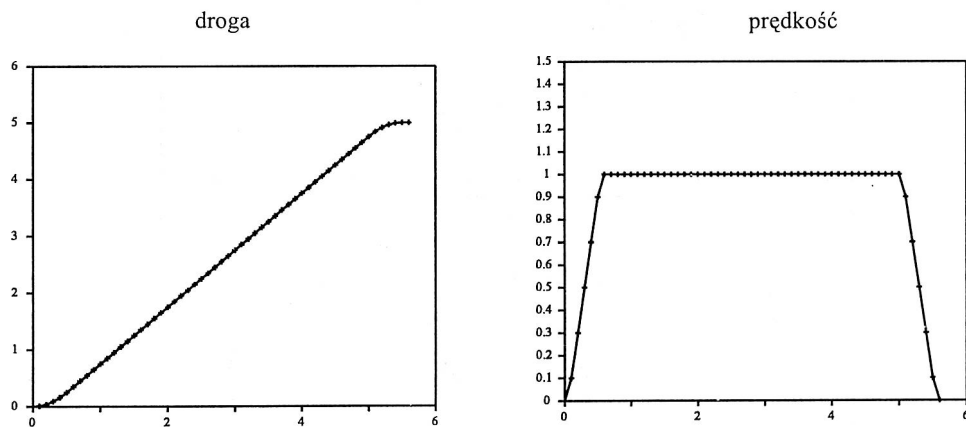
FAHRE LINEAR NACH (11500,1000,-1500,0,0,0)

HALT

ENDE

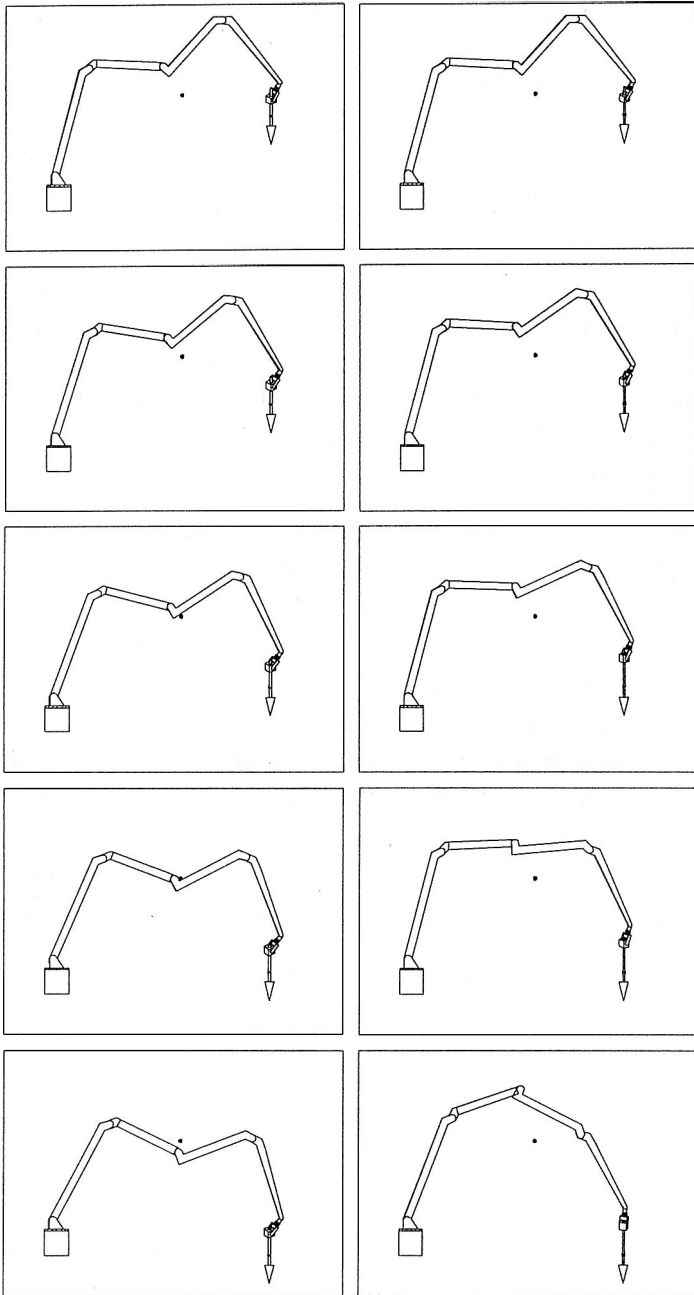
Zarejestrowane wykresy przebiegu drogi i prędkości członu roboczego podczas symulacji ruchu na tym torze przedstawiono na rysunku 6.24.

Na rysunku 6.25 przedstawiono porównanie kolejnych faz ruchu robota przy wyłączonej i włączonej opcji omijania przeszkód. Na rysunkach 6.26–6.29 przedstawiono



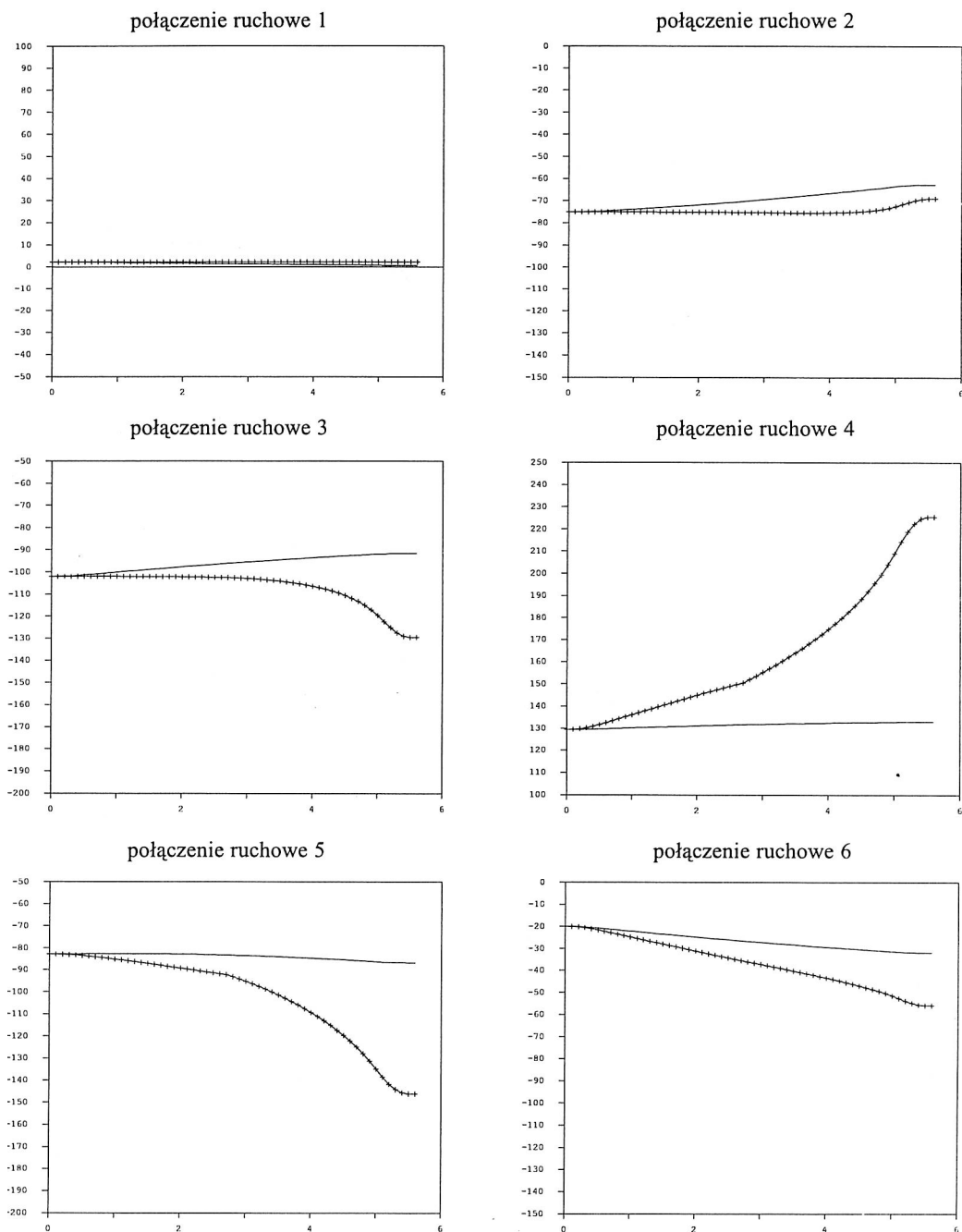
Rys. 6.24. Przebieg drogi [m] i prędkości [m/s] członu roboczego robota w czasie 5,5 sekund, podczas jego ruchu na odcinku prostej o długości 5 m

Fig. 6.24. Path [m] and speed [m/s] of the robot end-effector during 5,5 seconds while moving on the straight line segment 5 m long



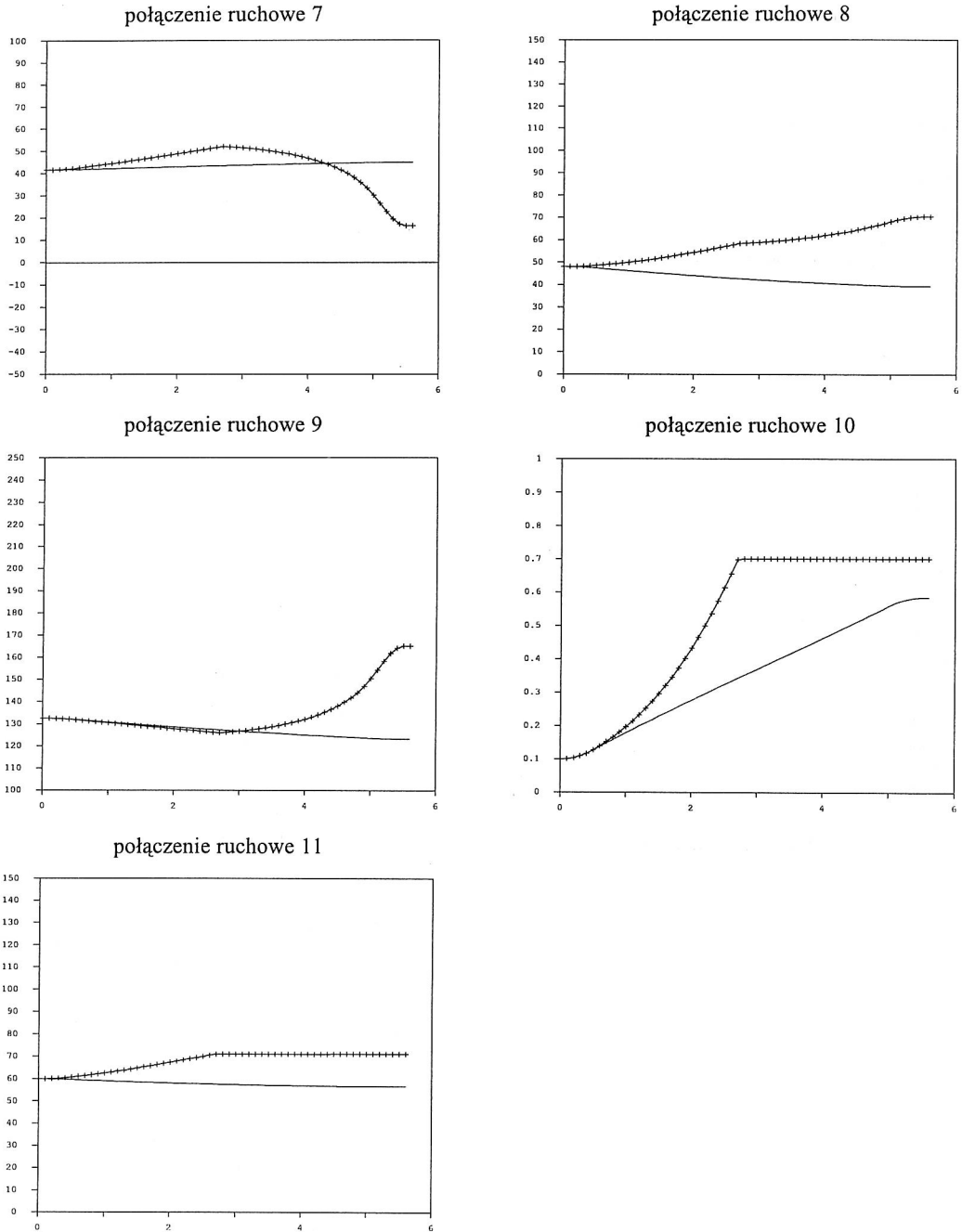
Rys. 6.25. Porównanie kolejnych faz symulowanego ruchu 11-przegubowego robota przy wyłączonej (po lewej) i włączonej (po prawej) opcji omijania przeszkód

Fig. 6.25. Comparison of simulated motion of 11-joint robot with the option for obstacle avoidance switched off (on the left) and switched on (on the right)



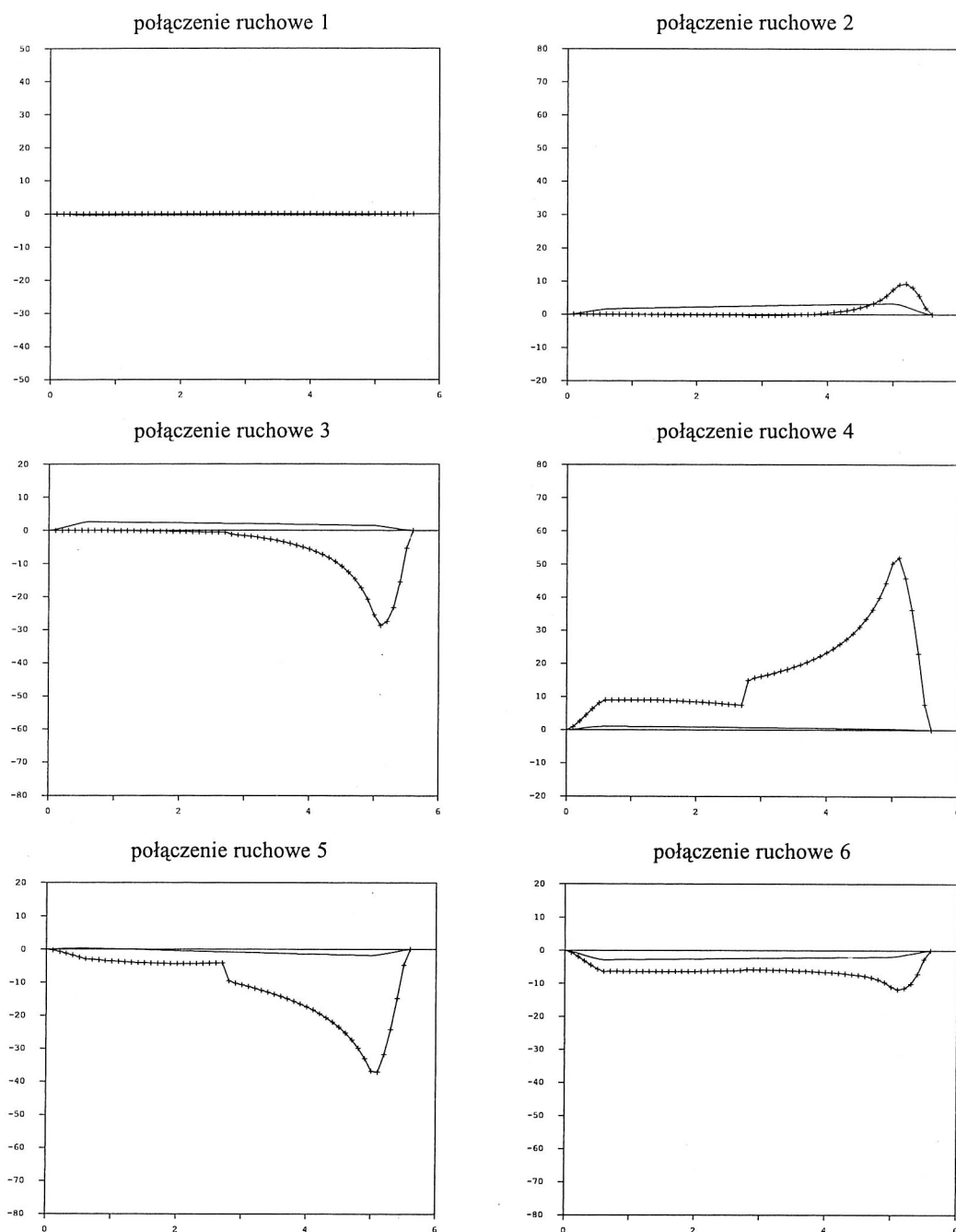
Rys. 6.26. Przebieg wychyleń [w stopniach] w połączeniach ruchowych o numerach 1–6 w czasie 5,5 sekund, podczas ruchu członu roboczego na odcinku prostej o długości 5 m (rys. 6.17), przy dwóch różnych opcjach ruchu: — bez omijania przeszkody, +++ przy włączonej opcji omijania przeszkody

Fig. 6.26. Motion [in deg] of joints 1–6 during 5,5 sec. while moving the end-effector on the straight line segment 5 m long (fig. 6.17), for two different motion option: — without obstacle avoidance and +++ with the obstacle avoidance



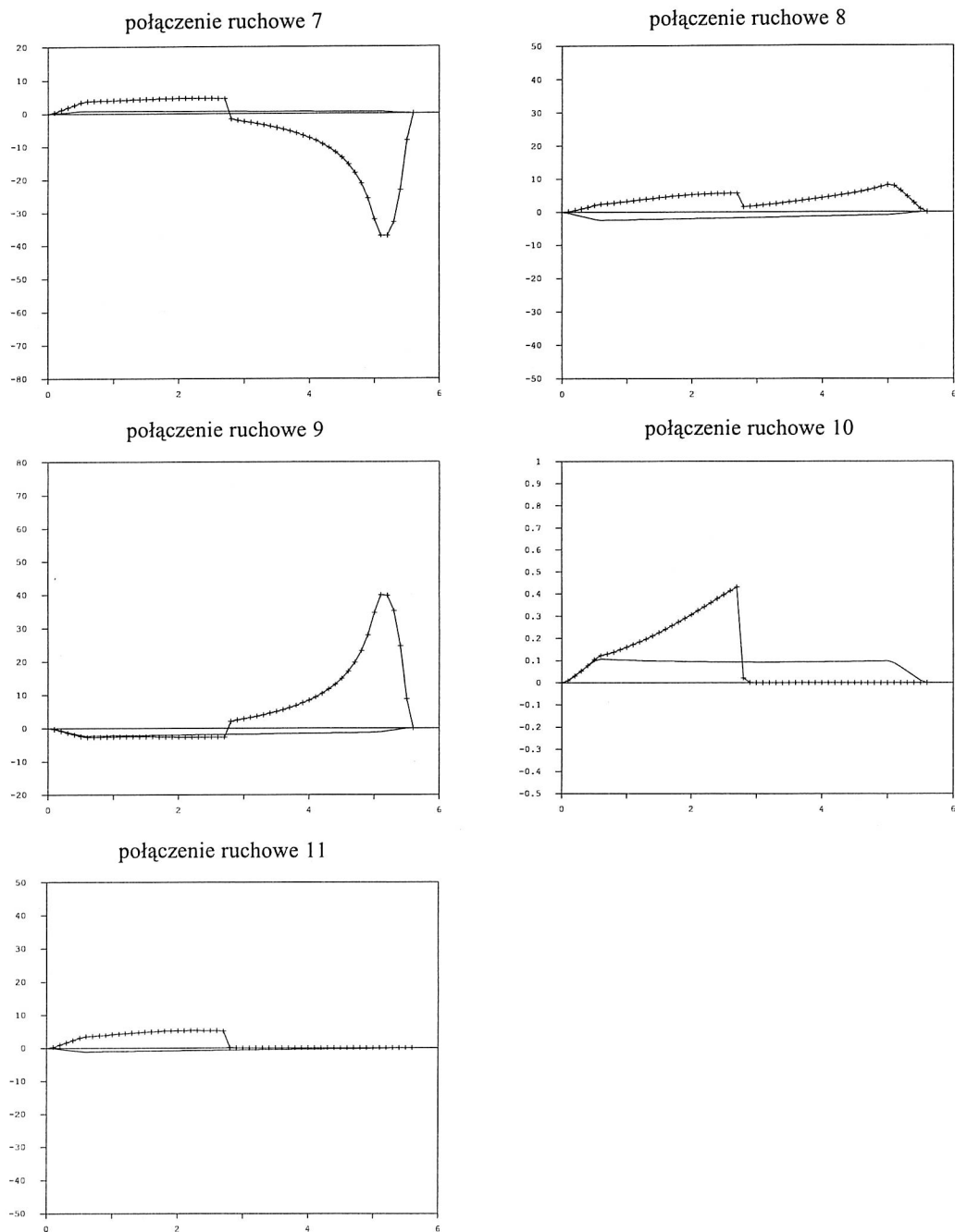
Rys. 6.27. Przebieg wychyleń [w stopniach, tylko 10 połączenie ruchowe w m] w połączeniach ruchowych o numerach 7–11 w czasie 5,5 sekund, podczas ruchu członu roboczego na odcinku prostej o długości 5 m (rys. 6.17), przy dwóch różnych opcjach ruchu: — bez omijania przeszkody, +++ przy włączonej opcji omijania przeszkody

Fig. 6.27. Motion [in deg, only joint 10 in m] of joints 7–11 during 5,5 sec. while moving the end-effector on the straight line segment 5 m long (fig. 6.17), for two different motion option: — without obstacle avoidance and +++ with the obstacle avoidance



Rys. 6.28. Przebieg prędkości ruchu [w °/s] w połączeniach ruchowych o numerach 1–6 w czasie 5,5 sekund, podczas ruchu członu roboczego na odcinku prostej o długości 5 m (rys. 6.17), przy dwóch różnych opcjach ruchu: — bez omijania przeszkody, +++ przy włączonej opcji omijania przeszkody

Fig. 6.28. Speed [in °/s] of joints 1–6 during 5,5 sec. while moving the endeffector on the straight line segment 5 m long (fig. 6.17), for two different motion options: — without obstacle avoidance and +++ with the obstacle avoidance



Rys. 6.29. Przebieg prędkości ruchu [w %/s, tylko połączenie ruchowe 10 w m/s] w połączeniach ruchowych o numerach 7–11 w czasie 5,5 sekund, podczas ruchu członu roboczego na odcinku prostej o długości 5 m (rys. 6.17), przy dwóch różnych opcjach ruchu: — bez omijania przeszkody, +++ przy włączonej opcji omijania przeszkody

Fig. 6.29. Speed [in %/s, only joint 10 in m/s] of joints 7–11 during 5,5 sec. while moving the end-effector on the straight line segment 5 m long (fig. 6.17), for two different motion option: — without obstacle avoidance and +++ with the obstacle avoidance

natomiast wykresy przebiegu wychyleń i prędkości ruchu, w poszczególnych połączeniach ruchowych, zarejestrowane podczas symulacji.

Podczas symulacji obowiązywał następujący zestaw ograniczeń na wychylenia w poszczególnych połączeniach ruchowych:

$$\begin{array}{lll}
 -180^\circ \leq \theta_1 \leq 180^\circ & -240^\circ \leq \theta_5 \leq 0^\circ & -180^\circ \leq \theta_9 \leq 180^\circ \\
 -150^\circ \leq \theta_2 \leq 90^\circ & -240^\circ \leq \theta_6 \leq 60^\circ & -700\text{mm} \leq \theta_{10} \leq 700\text{mm} \\
 -180^\circ \leq \theta_3 \leq 60^\circ & -240^\circ \leq \theta_7 \leq 240^\circ & -90^\circ \leq \theta_{11} \leq 90^\circ \\
 0^\circ \leq \theta_4 \leq 240^\circ & -90^\circ \leq \theta_8 \leq 90^\circ &
 \end{array}$$

W 10 połączeniu ruchowym ograniczenie górne 700 mm na wychylenie w tym połączeniu było aktywne podczas ruchu z opcją omijania przeszkód.

Ruch symulowano dla kryterium w postaci ważonej sumy kryteriów, określonego wyrażeniem (48) z rozdziału 4.2.4, z następującymi wartościami:

$$\mathbf{W} = \text{diag}(11, 10, 9, 8, 7, 6, 5, 4, 3, 5 \cdot 10^{-7}, 1),$$

$$\alpha = \frac{1000}{R}$$

gdzie R – odległość od przeszkody,

$$\alpha_{\text{przeszk}} = 0,7$$

$$\alpha_{\text{konf}} = 0,3.$$

7. PODSUMOWANIE I WNIOSKI

W pracy przedstawiono oryginalną koncepcję programowania robotów o kinematykach redundantnych dla zadań sformułowanych czysto kinematycznie. Zaproponowano przy tym rozwiązanie dwóch kluczowych problemów ruchu kinematyk redundantnych:

- algorytmu sterowania, zwłaszcza transformacji odwrotnej,
- metody programowania.

Proces znalezienia rozwiązania obu tych kluczowych problemów został wsparty metodycznym podejściem. W przypadku algorytmu sterowania

- sformułowano listę wymagań w stosunku do tego algorytmu uwzględniając specyfikę zastosowań przemysłowych,
- dokonano przeglądu metod zaproponowanych w ostatnich 20. latach,
- opracowano oryginalny nowy algorytm spełniający założone wymagania.

W przypadku rozwiązywania problemu programowania robotów przemysłowych o kinematykach redundantnych

- dokonano przeglądu metod programowania urządzeń wytwórczych, a zwłaszcza robotów,
- dokonano wyboru tekstowego języka programowania i uzasadniono ten wybór,
- zaproponowano oryginalne konstrukcje językowe, o które można rozszerzyć standardowe tekstowe języki programowania.

Opracowany nowy algorytm sterowania kinematykami redundantnymi polega na zastosowaniu kompaktowej metody matematycznego programowania kwadratowego w iteracyjnej metodzie Newtona-Raphsona, z elementami tzw. sterowania konfiguracją. Opracowane opcje tego algorytmu obejmują:

- utrzymanie wychyleń w połączeniach ruchowych, ich prędkości i przyspieszeń w zadanych granicach,
- zapewnienie powtarzalności ruchu,
- zmianę konfiguracji (tzw. *selfmotion*),
- omijanie przeszkód,
- zamrożenie ruchu wybranych połączeń ruchowych.

Algorytm ten ma wiele właściwości, które dają mu przewagę nad innymi metodami sterowania redundantnymi robotami.

• W opracowanym algorytmie **uwzględnienie ograniczeń wychyleń w połączeniach ruchowych**, a także ich prędkości i przyspieszeń nie wpływa na zadanie optymalizowania konfiguracji względem ustalonego kryterium. Ograniczenia ruchowe w połączeniach określają bowiem jedynie obszar, w którym poszukiwana jest optymalna konfiguracja minimalizująca zadane kryterium, w postaci formy kwadratowo-liniowej i zbędne jest dodawanie do tego kryterium składnika typu potencjał, który wzrasta przy zbliżaniu się wychyleń w połączeniach ruchowych do ich granic (jak to występuje w metodach opisanych w rozdziałach 3.4 i 3.5).

- Przez ograniczenie prędkości wychyleń nałożone w połączeniach ruchowych, można zapewnić **utrzymanie konfiguracji robota z dala od konfiguracji osobliwych**, do których zbliżanie się powoduje wzrost prędkości ruchu połączeń ruchowych aż do nieskończoności.

- Konstrukcja algorytmu oparta na iteracyjnej metodzie Newtona-Raphsona daje **możliwość klasycznego zastosowania układów regulacji położenia w poszczególnych osiach** (współrzędnych konfiguracyjnych). Jak to już stwierdzono w rozdziale 3.15, zaletą stosowania układów regulacji położenia w poszczególnych osiach jest mniejszy czas opóźnienia wyregulowania uchybu niż w układzie regulacji we współrzędnych kartezjańskich, gdzie w obwodzie regulacji dodatkowo musi być wykonanych kilka operacji z dużą liczbą obliczeń. Inną zaletą tego rozwiązania jest utrzymanie rozdziału między transformacją odwrotną i regulacją położenia, co umożliwia jednolitą realizację tworzenia ruchu przy jego zaprogramowaniu we współrzędnych kartezjańskich i konfiguracyjnych. W układzie regulacji we współrzędnych kartezjańskich brak tego rozdziału i aby umożliwić tu programowanie ruchu we współrzędnych konfiguracyjnych, musiałby w sterowaniu robota i tak zostać zaprogramowany dodatkowo alternatywny algorytm przetwarzania danych z samymi układami regulacji położenia w osiach.

- Algorytm **spełnia wymagania zestawione w rozdziale 4.1**, które uwzględniają specyfikę zastosowań przemysłowych.

- Zaproponowany algorytm **nadaje się do zastosowania w celu podniesienia zdolności i zręczności robotów redundantnych względem wielu kryteriów**, także i innych niż te przedstawione już w rozdziale 4. Jednym z ważniejszych kryteriów jest odległość konfiguracji łańcucha kinematycznego robota od konfiguracji osobliwych. Miarą tej odległości jest wskaźnik *MVR* wyrażający się stosunkiem prędkości ruchu w połączeniach ruchowych do prędkości członu roboczego (ang. *Manipulator Velocity Ratio*) [Dubey, Luh 1988]:

$$MVR = r_v = \|\dot{\mathbf{x}}\|_{\mathbf{W}_x} / \|\dot{\boldsymbol{\theta}}\|_{\mathbf{W}_\theta},$$

opisany w rozdziale 3.5. Wskaźnik *MVR* uważany jest nawet za lepszy od wskaźnika zwanego manipulacyjnością, który bada bezpośrednio wartości osobliwe macierzy Jacobiego:

$$w = \sqrt{\det(\mathbf{J}\mathbf{J}^T)}.$$

Zwiększanie manipulacyjności poprawia co prawda zdolność robota do poruszania się, ale nie konkretnie na zadanym w programie kierunku ruchu manipulatora, jak to jest właśnie w przypadku kryterium *MVR* [Dubey, Luh 1988].

Wskaźnik *MVR* można z powodzeniem zastosować do algorytmu kompaktowej metody matematycznego programowania kwadratowego w iteracyjnej metodzie Newtona-Raphsona, ponieważ MVR^{-2} przyjmuje postać formy kwadratowo-liniowej podanej równaniem (40 a), a minimalizowanie MVR^{-2} oznacza zwiększanie *MVR*:

$$\frac{1}{MVR^2} = \frac{\dot{\theta}^T \mathbf{W}_\theta \dot{\theta}}{\dot{\mathbf{x}}^T \mathbf{W}_x \dot{\mathbf{x}}}.$$

Dla sprowadzenia MVR^{-2} do postaci (40a) należy posłużyć się uśrednionymi prędkościami we współrzędnych konfiguracyjnych i kartezjańskich w obszarze jednego taktu interpolacji Δt . Zgodnie z przyjętymi oznaczeniami w rozdziale 4.2.1:

$$\dot{x}_i = \frac{\Delta x_i}{\Delta t},$$

$$\dot{\theta}_i = \frac{\Delta \theta_i + \sum \theta_i}{\Delta t}.$$

Następnie należy je podstawić do wzoru na MVR^{-2} :

$$\frac{1}{MVR^2} = \frac{(\Delta \theta + \sum \Delta \theta)^T \mathbf{W}_\theta (\Delta \theta + \sum \Delta \theta)}{\Delta \mathbf{x}^T \mathbf{W}_x \Delta \mathbf{x}},$$

$$\frac{1}{MVR^2} = \frac{\Delta \theta^T \mathbf{W}_\theta \Delta \theta}{\Delta \mathbf{x}^T \mathbf{W}_x \Delta \mathbf{x}} + 2 \frac{\sum \Delta \theta^T \mathbf{W}_\theta \Delta \theta}{\Delta \mathbf{x}^T \mathbf{W}_x \Delta \mathbf{x}} + \frac{\sum \Delta \theta^T \mathbf{W}_\theta \sum \Delta \theta}{\Delta \mathbf{x}^T \mathbf{W}_x \Delta \mathbf{x}}.$$

Ostatni składnik powyższej sumy jest stały w danej iteracji Newtona-Raphsona i przy szukaniu minimum MVR^{-2} może zostać odrzucony, tak więc $\mathbf{g}$ i $\mathbf{H}$ ze wzoru (40a) odnajdujemy jako:

$$\mathbf{H} = 2 \frac{\mathbf{W}_\theta}{\Delta \mathbf{x}^T \mathbf{W}_x \Delta \mathbf{x}},$$

$$\mathbf{g} = 2 \frac{\sum \Delta \theta^T \mathbf{W}_\theta}{\Delta \mathbf{x}^T \mathbf{W}_x \Delta \mathbf{x}}.$$

Kolejnym przykładem kryterium, które może być zastosowane w algorytmie jest kryterium minimalizacji energii kinetycznej:

$$\mathbf{E}_{\text{kinetyczna}} = \frac{1}{2} \dot{\theta}^T \mathbf{M} \dot{\theta},$$

gdzie $\mathbf{M}$ – macierz mas i momentów bezwładności manipulatora.

Posługując się uśrednionymi prędkościami we współrzędnych konfiguracyjnych i kartezyjskich w obszarze jednego taktu interpolacji Δt i dokonując podobnych podstawień i przekształceń jak poprzednio, wzór na energię kinetyczną przyjmuje następującą formę:

$$E_{\text{kinetyczna}} = \frac{1}{2} \frac{(\Delta\theta + \sum \Delta\theta)^T \mathbf{M} (\Delta\theta + \sum \Delta\theta)}{\Delta^2 t},$$

$$E_{\text{kinetyczna}} = \frac{1}{2} \frac{\Delta\theta^T \mathbf{M} \Delta\theta}{\Delta^2 t} + \frac{\sum \Delta\theta^T \mathbf{M} \Delta\theta}{\Delta^2 t} + \frac{1}{2} \frac{\sum \Delta\theta^T \mathbf{M} \sum \Delta\theta}{\Delta^2 t}.$$

Ostatni składnik powyższej sumy jest stały w danej iteracji Newtona-Raphsona i podczas szukania minimum może zostać odrzucony, tak więc $\mathbf{g}$ i $\mathbf{H}$ ze wzoru (40a) odnajdujemy jako:

$$\mathbf{H} = \frac{\mathbf{M}}{\Delta^2 t}, \quad \mathbf{g} = \frac{\sum \Delta\theta^T \mathbf{M}}{\Delta^2 t}.$$

Podobnie można przystosować do algorytmu kryterium minimalnego przyrostu zużycia energii całkowitej na wykonanie ruchu w połączeniach ruchowych w przedziale czasu Δt , które zostało przedstawione w rozdziale 3.6 [Vukobratovic, Kircanski 1984]:

$$K = \Delta E = \frac{1}{2} \dot{\boldsymbol{\theta}}^T \mathbf{B} \dot{\boldsymbol{\theta}} + \mathbf{b}^T \dot{\boldsymbol{\theta}},$$

gdzie macierz $\mathbf{B}$ i wektor $\mathbf{b}$ wynikają z modelu dynamicznego manipulatora i jego napędów.

- Konstrukcja algorytmu sterowania daje **łatwość jego integracji w podstawowy tok przetwarzania danych w sterowaniu robota**, na który składają się kolejno: interpretacja komend języka programowania, interpolacja, transformacja i regulacja położenia w poszczególnych osiach. Konstrukcja ta nie narusza tradycyjnej, ale i zarazem zalecanej budowy softwaru systemowego sterowania robotów z podziałem na poszczególne moduły funkcyjne.

- Do dalszych zalet algorytmu sterowania można zaliczyć **możliwość jego zastosowania do robotów mobilnych, a także kooperujących manipulatorów**, a więc sytuacji już dziś często spotykanych w przemyśle, na co wskazują rozważania z rozdziału 3.2.

Algorytm umożliwia dobór optymalnej, w sensie zadanego kryterium, konfiguracji dla danego położenia i orientacji członu roboczego, w tym dobór konfiguracji startowej dla danego zadania. Na przykład znalezienie najlepszej konfiguracji dla ominięcia przeszkody można dokonać programując ruch członu roboczego po okręgu o bardzo małej średnicy wokół punktu, dla którego poszukuje się tej optymalnej konfiguracji.

Do wad algorytmu należy bez wątpienia jego duża intensywność obliczeniowa, ponieważ w każdym takcie interpolacji wykonywanych jest kilka iteracji Newtona-Raphsona (zwykle od 1 do 3), w trakcie których każdorazowo obliczany jest jacobian manipulatora oraz wywoływana jest optymalizacja metodą kompaktowej metody programowania kwadratowego, będąca zresztą też metodą iteracyjną (patrz rozdział 4.3). Intensywność obliczeń wzrasta przy tym wraz ze wzrostem liczby redundancji manipulatora. Zastosowanie jednak nowoczesnego, aczkolwiek standardowego hardware i software (procesor pentium i kompilator C++ optymalizowany na pentium) pozwoliło w testowanym przypadku manipulatora o 11 stopniach swobody wykonać obliczenia w czasie rzeczywistym.

Opracowany algorytm, dzięki swej konstrukcji stanowi bazę do stworzenia koncepcji programowania. Zaproponowane elementy tekstowego języka programowania zostały przy tym tak skonstruowane, że mogą stanowić rozszerzenie standardowego języka *PLR (Programming Language for Robots)* opracowywanego w ramach ISO, w celu umożliwienia programowania robotów o kinematykach redundantnych.

Integracja algorytmu oraz metody programowania daje w rezultacie otwarty system sterowania i programowania, który nadaje się do wdrożenia praktycznego w sterowaniach robotów przemysłowych. Wyniki przetestowania algorytmu sterowania na wielokrotnie kinematycznie redundantnym trójwymiarowym robocie, na dość skomplikowanej trajektorii ruchu, wskazują na efektywność opracowanej metody sterowania. Praktyczne wdrożenie opracowanej koncepcji może stać się kolejnym krokiem na drodze budowania coraz bardziej zręcznych, wszechstronnych i elastycznych urządzeń.

LITERATURA

- Ahmad S., Luo S.: *Coordinated motion control of multiple robotic devices for welding and redundancy coordination through constrained optimization in cartesian space*. IEEE Transaction on Robotics and Automation, Dec. 1989, Vol. 5, No. 4.
- Andersen K., Cook G.E., Zein-Sabattou S. Fernandez K.R.: *Graphic simulation for robotic weld programming enhancement. Recent Trends in Welding Science and Technology*, TWR-89, Proceedings of the 2nd International Conference organised by ASM International, Gatlinburg, TN, USA, May 1989.
- Angebauer R., Rentschler U.: *Automatische Wirbelstromrißprüfung von Flugzeug Turbinen-Läufen – Ein Beispiel für die Schwachstellen konventioneller Robotersteuerungstechnik*, VDI Berichte 921 Industrieroboter Messen und Prüfen, Düsseldorf, VDI Verlag, 1991.
- AutoCAD: *Advanced Modelling Extension – Reference Manual*. Autodesk Ltd., 1990.
- Baillieul J.: *Kinematic programming alternatives for redundant manipulators*. IEEE International Conference on Robotics and Automation, St. Louis, USA, 1985.
- Baker D.R., Wampler II C.W.: *On the inverse kinematics of redundant manipulators*. The International Journal of Robotics Research, Mar. 1988, Vol. 7, No. 2.
- Bielski S., Koch T., Chlebus E.: *Nowe narzędzia efektywnego projektowania*. Design for Assembly, Technologia i Automatyzacja Montażu, Nr 2, OBR Tekoma, Warszawa 1993.
- Bielski S., Dziuba R., Kocęch A., Koch T., Kozera M.: *Elastyczne gniazdo montażowe oraz jego oprogramowanie*, Technologia i Automatyzacja Montażu, Nr 2, OBR Tekoma, Warszawa, 1993.
- Boddy C.L., Taylor J.D.: *Whole-arm reactive collision avoidance control of kinematically redundant manipulators*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Böke M., Miss R.W., Deeg H.-J.: *Offline-Programmierung und grafische Simulation als Planungsinstrument für den Robotereinsatz unter Wasser*. VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993
- Boothroyd G.: *Assembly automation and product design*, Marcel Dekker, Inc., New York, 1991.
- Borgolte U.: *IRL – Die deutsche Norm für explizite Roboterprogrammierung*. VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993.

- Bosch: *Handbuch Bosch robotwindows*. Robert Bosch GmbH Druckschrift Nr.: 3 842 394 235 IA 07/89 lub: *Manual Bosch robotwindows*. Robert Bosch GmbH Druckschrift Nr.: 3 842 394 497 IA 08/90.
- Chan T.F., Dubey R.V.: *A weighted least-norm solution based scheme for avoiding joint limits for redundant manipulators*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Chang P. H.: *A closed-form solution for inverse kinematics of robot manipulators with redundancy*. IEEE Journal of Robotics and Automation, Oct. 1987, Vol. 3, No. 5.
- Chen Y.-C., Walker I.D.: *A consistent null-space based approach to inverse kinematics of redundant robots*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Cheng F.T., Chen T.H., Sun Y.Y.: *Efficient algorithm for resolving manipulator redundancy – the compact QP method*. IEEE International Conference on Robotics and Automation, Nice, France, 1992.
- Cheng F.T., Chen T.H., Wang Y.S., Sun Y.Y.: *Obstacle avoidance for redundant manipulators using the compact QP method*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Cheung E., Lumelsky V.: *Motion planning for whole-sensitive robot arm manipulator*. IEEE International Conference on Robotics and Automation, Los Alamitos, USA, 1990.
- Choi M. H., Lee B. H., Ko M. S.: *Optimal load distribution for two cooperating robots using a force ellipsoid*. Robotica, 1993, Vol. 11.
- Christensen J.: *Dreidimensionale Roboterführung mit dem LASARTM-System*. VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993.
- Colbaugh R., Seraji H., Glass K.L.: *Obstacle avoidance for redundant robots using configuration control*, Journal of Robotic Systems, 1989, 6.
- Cook G.E., Andersen K., Zein-Sabattou S., Fernandez K.R.: *Multiple-robot programming for coordinated motion, end-effector calibration, and part location*. Conference Record of the IEEE Industry Application Society Annual Meeting, Oct. 1989, 1–5.
- Craig J.J.: *Wprowadzenie do robotyki*. Warszawa, WNT, 1993.
- Dillman R., Huck M.: *Informationsverarbeitung in der Robotik*. Berlin, Springer Verlag, 1991.

- DIN 66312 Teil 2: *Industrial Robot Language (IRL) – Language Description (Draft)*, Version 1.2, April 1994.
- DIN 66313 IRDATA: *Schnittstelle zwischen Programmiersystem und Robotersteuerung*. Allgemeiner Aufbau, Satztypen und Übertragung.
- DIN 66314-1 IRDATA: *Schnittstelle zwischen Programmierung und Robotersteuerung*. Teil 1: Allgemeiner Aufbau, Satztypen und Übertragung (Entwurf). Sept. 1994.
- Dubey R., Luh J.Y.S.: *Redundant robot control using task based performance measures*, Journal of Robotic Systems 5, 1988.
- Duelen G., Kirchoff U., Held J., Münch H.: *Automatische Bewegungssynthese fürbahnbezogen kooperierende Industrieroboter*. Robotersysteme 3, 1987.
- Eispack: *Matrix Eigensystem Routines Eispack*, Lecture Notes in Computer Sciences 51, New York, Springer Verlag 1977.
- Galicki M.: *Metody planowania zadań manipulatorów redundantnych w przestrzeni roboczej z przeszkodami*. Monografia 47, Instytut Matematyki Stosowanej i Informatyki, Wydawnictwo Wyższej Szkoły Inżynierskiej w Zielonej Górze, Zielona Góra 1989.
- Galicki M.: *A closed solution to the inverse kinematics of redundant manipulators*. Mechanism and Machine Theory, 1991, Vol. 26, No. 2.
- Galicki M.: *Applying the calculus of variations to redundancy resolution of manipulators*. W: Robotics research and application. Ed. by Morecki A., Muszyński W., Tchoń K., Warszawa, WNT, 1992.
- Glass K., Colbaugh R., Lim D., Seraji H.: *On-line collision avoidance for redundant manipulators*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Goldfarb D., Idnani A.: *A numerically stable dual method for solving strictly convex quadratic programs*, Mathematical Programming, 1983, Vol. 27.
- Henrich D., Cheng X.: *Fast distance computation for on-line collision detection with multiarm robots*. IEEE International Conference on Robotics and Automation, Nice, France, 1992.
- Hollerbach J.M., Suh K.C.: *Redundancy resolution of manipulators through torque optimization*, IEEE Journal of Robotics and Automation, Aug. 1987, Vol. 3, No. 4.
- Höfer A.: *Steuerung der Konfiguration eines redundanten Manipulators*. Fortschritte der Robotik 17, Wiesbaden, Vieweg, 1992.

- Hsia T.C., Mao Z.: *Obstacle avoidance inverse kinematics solution of redundant manipulators y neural networks*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- IMSL *Library User's Manual*, IMSL Mathematical Library, Version 1.0, IMSL Inc. Houston, USA, April 1987.
- Industrial Robot, 1993, Vol. 20, No. 2: Kinematic Advance.
- Jacak W.: *Roboty inteligentne: Metody planowania działań i ruchów*. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej Nr 85, seria Monografie nr 18, Wydawnictwo Politechniki Wrocławskiej, Wrocław 1991.
- Kazerounian K., Wang Z.: *Global versus local optimization in redundancy resolution of robotic manipulators*, The International Journal of Robotics Research, Oct. 1988, Vol. 7, No. 5.
- Klein C.A., Blaho B.E.: *Dexterity Measures for the design and control of kinematically redundant manipulators*, The International Journal of Robotics Research, Summer 1987, Vol. 6, No. 2.
- Klein C., Chu-Jenq C., Ahmed S.: *Use of an extended Jacobian method to map algorithmic singularities*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Klein C.A., Huang C.-H.: *Review of pseudoinverse control for use with kinematically redundant manipulators*, IEEE Transactions on System Man and Cybernetics, Mar. 1983, Vol. 13, No. 3.
- Klema V.C., Laub A.J.: *The singular value decomposition: its computation and some applications*, IEEE Transaction on Automatic Control, Apr. 198, Vol. 25, No. 20.
- Koch T.: *Metoda koordynacji ruchu po trajektorii dwóch robotów*. Konferencja: Podstawy technologii maszyn'91, Wrocław 1991.
- Koch T.: *Robots movement coordination*. W: Robotics research and application. Ed. by Morecki A., Muszyński W., Tchoń K., Warszawa, WNT, 1992.
- Koch T., Dybała B., Kowalczewski P.: *Robot workcell simulation system for personal computers*. International Conference on Computer Integrated Manufacturing, Zakopane, 1992.
- Koch T., Kowalczewski P., Kocełuch A., Dybała B., Szypicyn A.: *System symulacji gniazda zrobotyzowanego*, IV Krajowa Konferencja Robotyki, Wrocław 1993.
- Koch T., Kozera M., Kocełuch A., Bielski S., Dziuba R.: *Sprzężenie układów sterowania robota przemysłowego typu SCARA i transportowego systemu paletowego fir-*

my Bosch pracujących w gnieździe montażowym, IV Krajowa Konferencja Robotyki, Wrocław, 1993.

Koch T.¹: *Zastosowanie robotów w przemyśle*. Sympozjum: Automatyzacja produkcji. Prace Naukowe ITMiA Politechniki Wrocławskiej Nr 53, Seria: Konferencje Nr 21, Wrocław, 1994.

Koch T.²: *Steuerungskonzepte für Roboter mit redundanter Kinematiken*. International Conference on Computer Integrated Manufacturing, Zakopane, 1994.

Koch T., Kowalczewski P., Szypicyn A., Kocełuch A., Dutkowiak M.: *Automated robot program generation using the simulation system for robotised workcell*. International Conference on Computer Integrated Manufacturing, Zakopane, 1994.

Koch T., Kowalczewski P., Dybała B., Kocełuch A., Szypicyn A., Dutkowiak M.: *System do planowania, programowania i symulacji zrobotyzowanego stanowiska produkcyjnego na bazie komputera klasy PC*. Raport serii sprawozdania Nr 1/94, ITMiA Politechniki Wrocławskiej, Wrocław, 1994.

Koch T.¹: *Programming of redundant robots with textual programming language*. CIRP Second International Workshop on Learning in Intelligent Manufacturing Systems – LIMS'95, Budapest, 20–21 April 1995.

Koch T.²: *Algorithm for resolving manipulator redundancy – the compact QP method in Newton-Raphson iteration scheme*. IMEKO International Symposium on Measurement and Control in Robotics – ISMCR'95, Bratislava, 14–16 June 1995.

Koch T.³: *Programmierung von Industrierobotern mit redundanten kinematischen Ketten*. 6th International DAAAM Symposium: Intelligent Manufacturing Systems, Kraków 26–28th October 1995.

Kreutz-Delgado K., Long M., Seraji H.: *Kinematic analysis of 7-DOF anthropomorphic arms*. IEEE International Conference on Robotics and Automation, Los Alamitos, USA, 1990.

Liegeois A.: *Automatic supervisory control of the configuration and behaviour of multibody mechanisms*. IEEE Transactions on System Man and Cybernetics, Dec. 1977, Vol. 7, No. 12.

Luecke G. R., Gardner J. F.: *Local joint control in cooperating manipulators systems – force distribution and global stability*, Robotica, 1993, Vol. 11.

Maciejewski A.A., Klein C.A.: *Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments*. The International Journal of Robotics Research, Vol. 4, No. 3, Fall 1985.

- Maciejewski A.A., Reagin J.M.: *A parallel algorithm and architecture for the control of kinematically redundant manipulators*. IEEE International Conference on Robotics and Automation, Nice, France, 1992.
- Mayorga R.V., Ma K.S., Wong A.K.C., Ressa B.: *A fast approach for the path planning of telerobotic manipulators*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Miksch W., Schröder D.: *Koordinierung der Fahrzeug- und Roboterarmbewegung bei mobilen Robotersystemen*, Autonome Mobile Systeme, 9. Fachgespräch an der TU München, 1993.
- Miksch W., Schröder D.: *Methoden zur Regelung eines mobilen Robotersystems mit Redundanz*. VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993.
- Mitsi S., Bouzakis K.D.: *Simulation of redundant manipulators for collision avoidance in manufacturing and assembly environments*. Mechanism and Machine Theory, 1993, Vol. 28, No. 1.
- Morecki A.: *Zagadnienia współczesnej robotyki*. Biuletyn PIAP Nr 2-166/93.
- Morecki A., Knapczyk J. i inni: *Podstawy robotyki*. Warszawa, WNT, 1993.
- Münch H.: *Bewegungssynthese zur Steuerung redundanter und kooperierender Kinetiken*. Produktionstechnik – Berlin, Nr 94, Hanser, 1991.
- Nagamatsu S., Sugiura Y., Sato Y.: *Development of multi-arm assembly robot*. Toyota Technical Review, Vol. 41. No. 2, Apr. 1992.
- Nakamura Y.: *Advanced robotics: redundancy and optimization*. Addison Wesley, 1991
- Nakamura Y., Hanafusa H.: *Inverse kinematic solutions with singularity robustness for robot manipulator control*, Journal of Dynamic Systems, Measurement and Control, Sep. 198, Vol. 1086.
- Nakamura Y., Hanafusa H., Yoshikawa T.: *Task-priority based redundancy control of robot manipulators*. The International Journal of Robotics Research, Summer 1987, Vol. 6, No. 2.
- Nenchev D. N.: *Redundancy resolution through local optimization: a review*. Journal of Robotic Systems, 1989, 6.
- N.N.: *Der Roboter im Produktionstechnischen Umfeld*, Aachener Werkzeugmaschinen Kolloquium'93, Düsseldorf, VDI Verlag, 1993.
- Novak J.L., Feddama T.: *A capacitance-based proximity sensor for whole arm obstacle avoidance*. IEEE International Conference on Robotics and Automation, Nice, France, 1992.

OSACA: *Materialy informacyjne ESPRIT III Project 6379: Open System Architecture for Controls within Automation Systems.*

Palm R., Rehfuß U.: *Fuzzy-Steuerung in der Robotik.* Mikroelektronik, 1992, Bd. 6, H. 1.

Powell M.J.D.¹: *On the quadratic programming algorithm of Goldfarb and Idnani.* University of Cambridge, Department of Applied Mathematics and Theoretical Physics (DAMTP) Report NA19, Cambridge, England 1983.

Powell M.J.D.²: *ZQPCVX a Fortran subroutine for convex quadratic programming.* University of Cambridge, Department of Applied Mathematics and Theoretical Physics (DAMTP) Report NA17, Cambridge, England 1983.

Pritschow G.: *Sterowanie obrabiarkami i robotami przemysłowymi.* Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1995.

Pritschow G., Bauder M.: *Steuerungsstruktur und Programmierkonzept zur on line Bewegungskordinierung zweier Roboter in flexiblen Montagezelle.* Robotersysteme, 1990, 6.

Pritschow G., Dalacker M., Kurz J.: *Gesamtkonzept und praxisgerechte Realisierung eines mobilen Roboters zur automatischen Erstellung von Mauerwerk auf der Baustelle,* Autonome Mobile Systeme, 9. Fachgespräch an der TU München, 1993.

Pritschow G., Dalacker M., Kurz J.: *Ein mobiler Roboter zur automatisierten Erstellung von Mauerwerk auf der Baustelle.* VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993.

Pritschow G., Frager O.: *Roboterzellen-Programmierung: Die Sprache IRL und Zwischen-code ICR,* Robotersysteme, 1992, 8, No. 1.

Pritschow G., Frager O., Schumacher H., Wieland E.: *Programmierung von roboterbestückten Produktionsanlagen.* Robotersysteme, 1989, 5.

Pritschow G., Gronbach H., Uhl A.: *Dynamiksimulation als Komponente einer umfassenden Roboterentwicklungsumgebung.* VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993.

Pritschow G., Koch T.: *Koordinierte Bahnführung zweier Roboter,* Robotersysteme, 1991, 7.

Risse W., Fink B., Hiller M.: *Multisensor-based control of a redundant SCARA Robot.* 9th World Congress on the Theory of Machines and Mechanisms, Milano 1995.

ROBCAD: *Materialy informacyjne firmy Tecnomatix Automatisierungssysteme GmbH.*

- Roberts R.G., Maciejewski A.A.: *Repeatable generalized inverse control strategies for kinematically redundant manipulators*. IEEE Transaction on Automatic Control, May 1993, Vol. 38, No. 5.
- Roboter 93: *Zweiarmig und mobil: Grundlagenforschung für Service-Roboter*. Roboter, Aug. 1993, Verlag Moderne Industrie.
- Rosen J.B.: *Nonlinear programming. The gradient projection method*. American Mathematical Society, 63 (1957), Abstract No. 80.
- Schraft R.D.: *Vom Industrieroboter zum Serviceroboter – Stand der Technik, Chancen und Grenzen*. VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993.
- Schumacher H.: *Einheitliche Programmierung von Automatisierungskomponenten roboterbestückter Bearbeitungs- und Montagezellen*. ISW Forschung und Praxis Band 83, Berlin, Springer Verlag 1991.
- Schumacher H., Braun J.: *“Skywash”: Innovative Steuerungsfunktionen für Großroboter*. VDI Berichte 1094: Intelligente Steuerung und Regelung von Robotern, Düsseldorf, VDI Verlag, 1993.
- Seraji H.: *Configuration control of redundant manipulators: theory and implementation*, IEEE Transaction on Robotics and Automation, Aug. 1989, Vol. 5, No. 4.
- Seraji H., Colbaugh R.: *Improved configuration control for redundant robots*. Journal of Robotic Systems, 1990, 7.
- Seraji H.: *Task based configuration control of redundant manipulators*, Journal of Robotic Systems, 1992, 9(3).
- Seraji H.: *An on-line approach to coordinated mobility and manipulation*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Shamir T., Yomdin Y.: *Repeatability of redundant manipulators: mathematical solution of the problem*, IEEE Transaction on Automatic Control, Nov. 1988, Vol. 33, No. 11.
- Shin Y. D., Chung M. J.: *An optimal distribution scheme for cooperating multiple robot manipulators*, Robotica, 1993, Vol. 11.
- Siciliano B.: *Kinematic control of redundant robot manipulators: a tutorial*. Journal of Intelligent and Robotic Systems, 1990, 3.
- Tsuji T., Nakayama S., Araki A., Ito K.: *Instantaneous inverse kinematic solution for redundant manipulators based on virtual arms and its application to winding control*. JSME International Journal, Series C, 1995, Vol. 38, No. 1.

- VDI 2863: *Programmierung numerisch gesteuerter Handhabungseinrichtungen*. Allgemeiner Aufbau, Satztypen und Übertragung. Dez. 1987.
- Vukobratovic M., Kircanski M.: *A dynamic approach to nominal trajectory synthesis for redundant manipulators*. IEEE Transactions on System Man and Cybernetics, Jul. 1984, Vol. 14, No. 4.
- Vukobratovic M., Kircanski M.: *Scientific fundamentals of robotics 3: Kinematics and trajectory synthesis of manipulation robots*, Berlin, Springer Verlag 1986.
- Walker I.D., Cavallaro J.R.: *Parallel VLSI architectures for real-time kinematics of redundant robots*. IEEE International Conference on Robotics and Automation, Atlanta, USA, 1993.
- Wampler II C.W.: *Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods*. IEEE Transactions on System Man and Cybernetics, Jul. 1986, Vol. 16, No. 1.
- Wang J.T.: *Redundancy Resolution of Robotic Manipulators using Normalized Generalized Inverses*. Trans. ASME Journal of Mechanical Design, Sep. 1995, Vol. 117.
- Warnecke H.-J., Schraft R.D.: *Industrieroboter. Handbuch für Industrie und Wissenschaft*. Berlin, Springer Verlag 1990.
- Weck M.: *Werkzeugmaschinen*, Band 3: Automatisierung und Steuerungstechnik. Düsseldorf, VDI Verlag, 1989.
- Whitney D.E.: *Resolved Motion Rate Control of Manipulators and Human Prostheses*, IEEE Transactions on Man-Machine System, Jun. 1969, Vol. 10, No. 2.
- Wloka D.: *Robotersimulation*. Berlin, Springer Verlag, 1991.
- Wloka D.¹: *Robotersysteme*, Band 2: Graphische Simulation. Berlin, Springer Verlag, 1992.
- Wloka D.²: *Robotersysteme*, Band 3: Wissensbasierte Simulation. Berlin, Springer Verlag, 1992.
- Zghal H., Dubey R.V., Euler J.A.: *Collision avoidance of a multiple degree of redundancy manipulator operating through a window*. Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME, Dec. 1992, Vol. 114.
- Yoshikawa T.: *Manipulability of robotic mechanisms*. The International Journal of Robotics Research, Summer 1985, Vol. 4, No. 2.
- Yoshikawa T., Zheng X.: *Coordinated dynamic hybrid position/force control for multiple robot manipulators handling one constrained object*, The International Journal of Robotics Research, Jun. 1993, Vol. 12, No. 3.

PROGRAMMING OF INDUSTRIAL ROBOTS WITH REDUNDANT KINEMATICS

Developments in the field of robot kinematics relate mainly to applications of redundant kinematic chains. This should be seen as a tendency towards creation of more and more flexible machines. Kinematic redundancy occurs when a manipulator possesses more degrees of freedom than the minimum number required to execute a given task. The kinematically redundant robot offers greater flexibility, versatility and dexterity. It can simultaneously realize the main task and additional tasks like keeping joint movements within given limits, obstacle avoidance, singular configuration avoidance, etc.

One of the principal problems of redundant robot application is to find suitable control algorithm (kinematic inverse transformation scheme). There have been created many valuable algorithms for control of the redundant kinematics in the past two decades. However, only a few of these approaches consider particular requirements such an algorithm should fulfil for the industrial applications and they do not offer any general systematic approach. Taking into account characteristics of industrial applications, the following list of requirements for control algorithm for redundant robot has been set up:

- The algorithm should be designed for on-line robot control and should enable the path corrections in real time, e.g., on the basis of sensor signals.
- It should be possible to guarantee the motion repeatability (arm configuration repeatability) on the cyclic repeated closed path.
- The algorithm should enable optional and parameterized realization of additional tasks, which follow from the main task itself or from its boundary conditions. The main catalogue of these tasks could include: keeping the joints within given limits and obstacle avoidance.
- There should be given a possibility to extend the catalogue of additional tasks with user specific tasks.
- The algorithm should have open structure architecture according to the concept of Esprit OSACA (Open Structure Architecture for Control within Automation Systems) project, where each of the function modules is independent. It results in possible independence of the inverse transformation module from preceding operation of hand path generation and succeeding operation of positioning control of the axis.
- The algorithm should enable simple changing of motion programming between both space and joint coordinates.
- The algorithm should be applicable not only to the open serial kinematic chains but also to the coordination control of cooperating robots and mobile robots.
- The control programme implementing this algorithm should be suitable for the application in industrial robot controllers. It should form a standard module for the control of redundant robot kinematics.

Setting up these requirements and analysing several control algorithms investigated in the past two decades gave the idea of a new algorithm. The new algorithm relies on

the application of quadratic programming (QP) method in Newton-Raphson iteration scheme with some elements of the so-called configuration control. It can be applied for keeping the joints, their speed and acceleration within given limits, to assure the repeatability of robot motion on cyclic paths, to change the arms configuration without moving the robot hand (self motion), for obstacle avoidance and to move robot with some frozen joints (not moving some joints).

Development of the new algorithm fulfilling the requirements allowed to make next step in the field of redundant kinematics, i.e., to create a concept of their programming. Programming method is another principal problem of redundant robot application.

In comparison to the limited number of programming methods for machine tools there are a lot of robot programming techniques. However, certainly the most popular robot programming method is textual programming language. These languages originated usually from computer programming languages which had been modified and supplemented with constructions destined for programming of movements, position in space, arm configuration and communication with sensors and other devices. With the aid of textual programming languages a great number of task categories for robots can be solved. They are well proved compromise between simple techniques like Teach-in, Play-back or lower level languages and advanced techniques like graphical or task oriented programming methods. Therefore the textual programming languages are common programming methods of industrial robots and subject of intensive work towards standard textual programming language. Additional motivation for the creation of a standard language is a great number of languages provided by robot or robot controller manufacturers or research institutions causing superfluous costs for the industrial users if they must programme different robots with different languages. In Germany, the Pascal-like language IRL (Industrial Robot Language) was established as a DIN 66312 standard. The same efforts are made in ISO. The IRL language is the basis for ISO standard with the name of PLR (Programming Language for Robots, ISO Working Draft 11513.1).

Industrial robots are usually applied together with other manufacturing machines creating a so called robotised workcell. Therefore robot control must be programmed to cooperate with the other controls like numerical control (NC) of machine tools, programmable logic control (PLC) or control of coordinate measuring machine (CMM). Unfortunately programming methods of all these devices have been developed independently and they differ from each other considerably. Thus there are some trials to create uniform programming method for all manufacturing machines within a workcell, which is based on textual programming language for robots.

All the above-mentioned factors gave reasons to develop programming method of redundant robots on the basis of a textual programming language. The created concept relies on the set of language constructions proposed as an extension of PLR/IRL standards. The proposed constructions relate to:

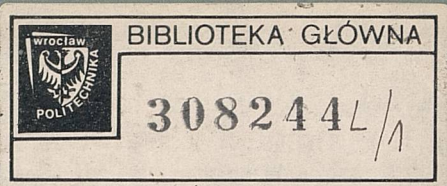
- definition of joint coordinates,
- setting up of limits for joint movement, speed and acceleration,

- repeatability on a closed path,
- change of the arms configuration without moving the robot hand (self motion),
- change of the arms configuration while moving the robot hand,
- obstacle avoidance,
- moving the robot with some frozen joints (not moving some joints) and
- integration of different movement option in the program.

To verify the created concept it was applied to control of 11-joint robot. For that reason the control system, the control algorithm, kinematics and geometry of the robot were modelled in the RME simulation system developed at the Technical University of Wrocław. Several options of additional tasks were successfully tested and proved the effectiveness of the created control method.

The practical implementation of the developed concept may become an important next step towards constructing more dexterous, versatile and flexible machines.

Verified by Halina Marciniak



**Przedstawiono przykłady zastosowań
tykach redundantnych i przegląd metod**

**Skoncentrowano się na rozwiązaniu dwóch kluczowych
problemów zastosowania przemysłowych robotów redun-
dantnych: zaprezentowano nowy oryginalny algorytm stero-
wania, polegający na zastosowaniu kompaktowej metody
matematycznego programowania kwadratowego oraz
koncepcję programowania, polegającą na rozszerzeniu
standardowego tekstowego języka programowania robotów.**

**Wydawnictwa Politechniki Wrocławskiej
są do nabycia w następujących księgarniach:
„Politechnika”**

**Wybrzeże Wyspiańskiego 27, 50-377 Wrocław
budynek A-1 PWr., tel. (0-71) 20 25 34;
„Tech”**

**plac Grunwaldzki 13, 50-377 Wrocław
budynek D-1 PWr., tel. (0-71) 20 32 52
Prowadzimy sprzedaż wysyłkową**

ISSN 087-5325